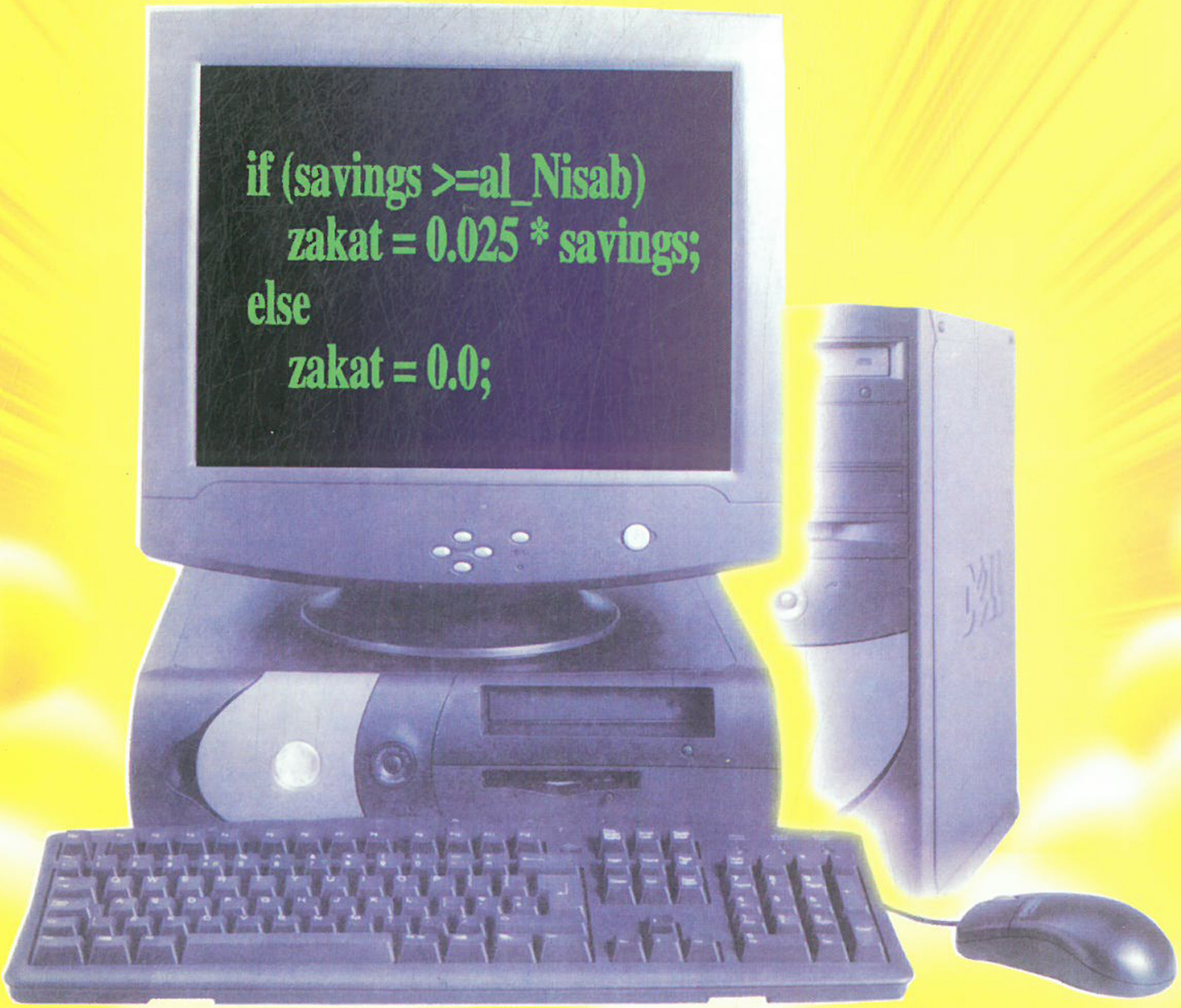
برمجة الحاسب
بلغة C++



الدكتور أبوبكر أحمد السيد
قسم الرياضيات وعلم الحاسوب
جامعة الكويت

دار القلم
كويت

ASHRAF NAEEM

برمجة الحاسب

بلغة C++

برمجة الحاسب بلغة C ++

الدكتور أبو بكر أحمد السيد

قسم الرياضيات وعلم الحاسوب

جامعة الكويت

بسم الله الرحمن الرحيم

﴿ صِبغَةَ اللهِ ومن أحسنُ من اللهِ صبغةً ونحن له عبدون ﴾ (*)

(سورة البقرة ١٣٨)

(*) سُمي الدين صبغة بطريق الاستعارة حيث يظهر أثره على المؤمن كما يظهر أثر الصبغ في الثوب.

الطبعة الأولى

١٤٢٢ - ٢٠٠٢

دار القلم للنشر والتوزيع

شارع السور - عمارة السور - الطابق الأول

هاتف : ٢٤٥٧٤٠٧ - ٢٤٥٨٤٧٨ - برقيا توزيعكو

ص . ب ٢٠١٤٦ الصفاة 13062 الكويت
فاكس ٢٤٢٥١٦٠

بسم الله الرحمن الرحيم

المقدمـــة

الحمد لله رب العالمين ، والصلاة والسلام على خاتم المرسلين نبينا محمد وعلى آله وصحبه أجمعين ... وبعد

دور التعليم في العالم الإسلامي :

الأمة الإسلامية أمة ذات عقيدة ورسالة {كنتم خير أمة أخرجت للناس تأمرون بالمعروف وتنهون عن المنكر وتؤمنون بالله} (آل عمران : ١١٠) ، فيجب أن يخضع التعليم فيها لهذه العقيدة وأن يوجه ليكون أداة لإنشاء أجيال تحمل تلك الرسالة .. وقد فرض الله تعالى على هذه الأمة عبادة الجهاد في سبيل الله {وجاهدوا في الله حق جهاده} (الحج : ٧٨) ، وإن من أفضل صور الجهاد اليوم الجهاد التعليمي الذي يعيد ثقة شبابنا بعظمة الإسلام وشمول نظامه ، ويحارب المبادئ الجاهلية التي سيطرت على بعض العقول بعد أن مضى علينا زمن طويل والغرب يبث في عقول شبابنا الشك والإلحاد ، وعدم الثقة بحقائق الإيمان والغيب والإيمان بعظمة الغرب وفلسفته ، بعد أن يصنعهم على عينه ويدرب ألسنتهم على لغته ويشبعهم بفكره ، إما في بلاده في الغرب ، أو في المدارس الأجنبية في بلادنا أو في جامعاتنا على أيدي أساتذته ، حتى وجدنا أنفسنا وقد ملك زمام معظم الأمور في بلادنا الإسلامية جيل يحمل أسماء إسلامية وعقولا غربية ، لا يؤمن بمبادئ الإسلام ، ولا يتحمس للغة القرآن ، بل يفتخر باللغة الأجنبية ويعد ذلك تطورا وتقدمية ، ويسخر من الكلمات والمصطلحات العربية ، ويرى في استخدامها تخلفا ورجعية ، فيلجأ إلى إثارة الشبهات حول تدوين العلوم باللغة العربية بحجة

الحرص على الأمة وتقدم البحث العلمي فيها ، وبالاختصار يصبح جيلا عربيا في لونه ودمه ، ولكنه إنجليزي أو أمريكي في ذوقه ورأيه ولغته وتفكيره [*]

ولقد كان من سياسة أعدائنا وامتدادا لحروبهم الصليبية ضدنا أنهم كلما احتلوا بلدا من بلادنا الإسلامية حاربوا لغتنا ولم يدعوا لها مكانتها المرموقة بل طردوها من دوائر التعليم والتربية وإدارة شئون الدولة ، وأقاموا على أنقاضها صرح لغتهم ، وطبق هذه الخطة الماكرة الخبيثة كل الغزاة الغربيين ، وكأنهم قد تواصوا فيما بينهم بذلك ، كما أنهم عملوا على تغريب المسلمين أنفسهم في أوطانهم بإنشاء جيل يجهل الإسلام ، وينظر للحياة بالمنظار الغربي ، ويسارع إلى إرسال أبنائه وبناته إلى المدارس الأجنبية ، ويقلد الغربيين في كل صغيرة وكبيرة رغم كونهم في بلادهم وبين بني جلدتهم ...

إن الارتباط وثيق بين العقيدة واللغة .. وإن إعادة كتابة العلوم الحديثة بأسلوب إسلامي يغرس الإيمان بوجود الخالق عز وجل ووحدانيته وحكمته في نفوس الشباب من خلال دراسته العلمية وتعرضه للحقائق الكونية لهو من أهم واجبات القائمين على شئون التعليم في العالم الإسلامي اليوم ، فليس من شئ أدعى إلى الإيمان بالله تعالى من حقائق العلم ، فيصبح الطالب بعد دراسته عالما مؤمنا ، سواء درس الطب والتشريح ، أو علوم الحاسب الإلكتروني ، أو علوم الأحياء والفيزياء ، أو حتى الرياضيات التي نثبت للطالب باستخدام قوانينها استحالة وجود هذا الكون صدفة.

خطورة الكتب الأجنبية والمناهج الغربية :

أما الكتب الأجنبية فلا يكاد الواحد منها يشير إلى الخالق سبحانه في عبارة واحدة ، فضلا عن أن يغرس إيمانا في نفوس الطلاب ، ولكنه يصلح لغرس الشك والإلحاد اللذين تسيطر روحهما على معظم الكتابات العلمية الأجنبية ، والتي

---

[*] انظر كتاب التربية الإسلامية الحرة لأبي الحسن الندوي.

تشتمل أحيانا على أسس وأفكار تناقض أسس عقيدتنا ، فمثلا لا يكاد يخلوا كتاب أجنبي عن برمجة الحاسب من مسائل عن الربا – والذي يسمى بالفائدة – وعن بعض حسابات البنوك القائمة على النظام الربوي ..

وفي ظل هذا النظام التعليمي الأجنبي الذي تستعار فيه المناهج الغربية وتدرس فيه الكتب الأجنبية أو تترجم كما هي تكون خسارة أمتنا أكبر بكثير من ربحها ، حيث تنشأ عندنا أجيال تشك في وجود الخالق سبحانه وتعالى وتستخف بفرائض الدين وتعاليمه وتؤمن بالفلسفات الغربية ، وبذلك يكون هذا النظام التعليمي قد أفسد وقضى على أفلاذ أكباد هذه الأمة المسلمة وخيرة شبابها وأكرم ذخائرها وأنفس ثرواتها وأكثرها استعدادا للنبوغ ، وصنع من هذه الفطر السليمة مصنوعات لا تنسجم ولا تتفق مع العقيدة التي نؤمن بها وندعو إليها ونموت في سبيلها.

ومن هنا فإن إعادة تدوين كتب العلوم المختلفة بحيث تشتمل على أحدث ما توصل إليه الإنسان من معلومات ، وبحيث تسري فيها في الوقت نفسه روح الإيمان بالله تعالى لهي خطوة هامة نحو البعث الإسلامي لأمتنا ، أما الإبقاء على تغريبها فهو ردة تقضي عليها وتبقيها عالة على غيرها من الأمم تتطفل على موائدها..

المناهج الدراسية والقيم الإسلامية

إن أهدافنا التعليمية ومناهجنا الدراسية يجب أن تتفق مع قيمنا الإسلامية ومبادئنا السامية التي تجمع في توازن واعتدال بين الدنيا والآخرة {ربنا آتنا في الدنيا حسنة وفي الآخرة حسنة} (البقرة : ٢٠١). ولقد كان الأوائل من فقهاء الأمة وعلمائها مجاهدين متقدمين في العلم بكل الأمور التي تؤثر في حياة المسلمين. كانوا موسوعيين بحق ، وأساتذة فعلا في كل التخصصات من الأدب والقانون إلى الفلك والطب ، ومن التفسير والفقه إلى الرياضيات والفيزياء .. كانوا رجالا متخصصين .. عرفوا الإسلام لا على أنه قانون ودستور فحسب ، بل على أنه أيضا

منهج ونظام حياة يعيشها ويمارسها ملايين من الناس. فالواحد منهم بالإضافة إلى كونه زعيما سياسيا أو قائدا عسكريا أو زارعا أو تاجرا أو صاحب حرفة كان في الوقت ذاته إماما ومجتهدا ومحدثا ومدرسا[*] .

علماء المسلمين الأوائل ومفهوم العبادة

وحين ندرس تاريخ حياة علماء المسلمين السابقين أمثال ابن الهيثم والخوارزمي والطبري والبيروني وابن خلدون وابن بطوطة والرازي والكندي نجد أنهم كانوا أشخاصا ذوي همة عالية ، وحماس بالغ ، وطاقة وافرة لطلب العلم ، وأن همتهم العالية هذه تنبع من إيمانهم بالله تعالى ، وعملهم بالتوجيه القرآني بالتفكر في خلق السموات والأرض ، والنظر في آيات الله تعالى في الكون وفي الآفاق وفي أنفسنا ، موقنين أن ذلك يعد واحدا من أسمى أشكال عبادة الخالق سبحانه وخشيته ، {إنما يخشى الله من عباده العلماء} (فاطر : ٢٨).

لقد كان العلم الذي قَدَّموه منسجما مع معتقداتهم ، وكان تطبيق ذلك العلم بأكمله يتم لصالح الإنسانية. لقد وضعوا أساس الطريقة العلمية في مختلف المجالات والعلوم كالطب والكيمياء ، وعلم البصريات وعلم الإنسان ، والجغرافيا وعلم الاجتماع ، وكان شعارهم الأول والأخير في جميع مساعيهم (الحمد لله رب العالمين) ، فكانت عظمة البارئ سبحانه وتعالى راسخة في نفوسهم ، وكان كل اكتشاف علمي يزيد من إيمانهم ، فلم يمنعهم امتيازهم العلمي ، وسبقهم الفكري من الركوع في الصلوات ، والصيام في رمضان..

أما اليوم فنجد أن كثيرا من المسلمين الذين نشأوا وتربوا على المناهج التعليمية الغربية يتملكهم الغرور والكبرياء عند أول اتصال لهم بالعلم الحديث مع ضعف عزيمتهم وفتور همتهم.

(*) راجع كتيب (تعريب العلوم وأسلمتها) للمؤلف ، دار القلم ، الكويت . وكتاب (أسلمة المعرفة) للدكتور إسماعيل الفاروقي ، المعهد العالمي للفكر الإسلامي.

الإيمان وطريقة التدريس .. السبيل إلى الحافز العلمي

إن المصدر الوحيد لتقوية عزيمة المسلمين ورفع همتهم إنما يكمن في إيمانهم ، وفي الطريقة التي تدرس بها العلوم المختلفة في المعاهد التعليمية ، فالطريقة الحالية غير مناسبة إطلاقا لتلك الغاية ، ومن أهم العوامل الرئيسية لافتقارنا إلى الحافز العلمي أن ما يدرس في معاهدنا هو نوع من المعرفة غير المتسقة مع إيماننا وحضارتنا وأسلوب حياتنا ، ولم تنبع من تربتنا.

من آثار انتقال العلم من أيدي المسلمين إلى أيدي الأوربيين

لقد فقد العلم أسسه الإسلامية عندما انتقل من أيدي المسلمين إلى أيدي الأوربيين. ونظرا للظروف التي مرت بها أوروبا ، واضطهاد الكنيسة للعلماء ، فقد فصلوا العلم عن الدين (المسيحية) وفي نهاية الأمر أصبحت وجهة النظر العلمية إلحادية وعدائية للدين ، حتى أصبح من غير الممكن التفكير في ذكر اسم الله تعالى في أي كتاب علمي أو مقالة أو بحث علمي.

ولما كانوا يدينون بعداء تاريخي للإسلام والمسلمين ، فقد قاموا عن عمد في كتبهم ومراجعهم بقطع جميع الصلات التي تربط العلم بالإسلام والمسلمين ، إلى حد أنه لقرون عديدة لم يذكروا أن الطريقة العلمية نبعت في الأصل من البلدان الإسلامية ، ولم يشيروا إلى وجود العلماء البارزين من المسلمين.

الكتب الأجنبية وجحود فضل علماء المسلمين :

وإن الطالب المسلم عندما يدرس الفيزياء مثلا في كتاب وضعه كاتب غربي فإنه يشعر بخيبة أمل لعدم رؤيته أي إشارة لأعمال العلماء الفيزيائيين المسلمين. إننا نعلم مثلا أن واضع علم البصريات هو الحسن بن الهيثم وليس نيوتن ، ولكننا دائما نجد اسم نيوتن ، وقد بولغ في تمجيده لتجاربه في علم الضوء ، وبطريقة مماثلة لا نجد ذكرا لعلماء المسلمين في الرياضيات أو الفلك والجغرافيا وغيرها ، وإذا وردت إشارة عابرة لبعض هؤلاء العلماء ، فإنما تذكر مساهمة العلماء العرب (وليس المسلمين). ونلاحظ التركيز دائما على العلماء الغربيين .. ونظرا لأن الطالب المسلم لا يشعر بالترابط العاطفي مع نيوتن وأينشتين وجاليليو فإنه يفقد الكثير من حماسه ، وإذا حاول أن يتلمس طريقه إلى العلم فإنه سوف يتشرب الأفكار الإلحادية والمعادية للدين التي تتخلل جميع العلوم الحديثة النابعة من الغرب [*].

من آثار تدريس العلوم باللغات الأجنبية

ومن ناحية أخرى فإن الطالب يفقد الكثير من الوقت والجهد والطاقة في محاولة إجادة تلك العلوم المكتوبة باللغة الأجنبية ، وفوق ذلك فإنه حتى إذا كان الكتاب يعالج موضوعا علميا فإن لغته ذاتها تكون متسمة ببعض الكلمات والعبارات الاصطلاحية والتعبيرات اللغوية التي تمثل انعكاسا لقيم واتجاهات القوم الذين يتحدثون بتلك اللغة ، ومن ثم فإن استخدام اللغة الأجنبية في تدريس العلوم سوف يؤدي في نهاية الأمر إلى تضاؤل رؤيتنا الإسلامية للكون ، ويحل محلها تدريجيا وجهة النظر المنفصلة عن الدين.

(*) انظر بحثا بعنوان : (الأسس الإسلامية للعلم) للدكتور محمد معين صديقي ، نشر بالإنجليزية في مجلة جمعية العلماء والمهندسين المسلمين بأمريكا الشمالية ، وقد نشرت ترجمته العربية في مجلة المسلم المعاصر على حلقتين في العددين ١٧ ، ١٨ سنة ١٣٩٩هـ / ١٩٧٩م.

دور علماء المسلمين في تأليف الكتب العلمية

ولهذا فإن الأساتذة المسلمين وعلماء المسلمين المهتمين بمصالح الأمة الإسلامية تقع على عاتقهم مسئولية كبيرة. إن مهمتهم تأليف كتب علمية بلغات البلدان الإسلامية مع إعطاء الأولوية للغة العربية ، بحيث تتسم هذه الكتب بالتصورات الإسلامية ، مثل كتاب (المناظر) لابن الهيثم ، وهو مرجع في علم البصريات سادت فيه وجهة النظر الإسلامية على مدى الكتاب بأكمله ، دون الإخلال بالمضمون العلمي [*] .. ويجب أن تؤلف تلك الكتب على جميع المستويات التعليمية الابتدائية والمتوسطة والثانوية والجامعية ، مع إعطاء الأولوية للكتب التي تدرس في المدارس والكليات والجامعات.

منهج هذا الكتاب ومحتوياته

وهذا الكتاب خطوة جديدة على طريق أسلمة العلوم وتعريبها ، وذلك في علم برمجة الحواسيب الإلكترونية بلغة الباسكال ، وهي إحدى اللغات الحديثة التي لها تطبيقات عملية وعلمية عديدة. وأما أسلوب الكتاب في عرض هذا المنهج فهو الأسلوب نفسه الذي اتبعناه سابقا في كتاب (برمجة الحاسب بلغة الفورتران) ، وكتاب (برمجة الحاسب بلغة الباسكال) ، ولذلك نقتبس هنا ما ورد في مقدمة كل من هذين الكتابين حول ذلك الأسلوب.

يعرض الكتاب قواعد اللغة مع استخدامها في حل عدد من المسائل في تطبيقات مختلفة عمليا على الحاسوب ، ومع مراعاة أن تسري بإذن الله تعالى الروح الإسلامية في مادة المنهج ، وأن ترتبط محتوياته كذلك بحياة المسلم للمساهمة في تكوين الشخصية الإسلامية والعقلية الإسلامية ، فمثلا تتناول بعض

---

[*] المصدر السابق.

الأمثلة والتمرينات برامج لتوزيع المواريث حسب قوانين الشريعة الإسلامية ، وحساب أنواع الزكاة المختلفة ، وحساب الأرباح أو الخسائر في نظم البنوك الإسلامية حيث لا تعامل بالربا ، وإحصائيات أعداد الحجاج ، وتعداد السكان في البلاد الإسلامية ، وحساب أجور المجاهدين في سبيل الله ، وتصنيف أحاديث الرسول صلى الله عليه وسلم ، وغير ذلك من الأمثلة .. وعلى العموم ربط بعض مسائل الحسابات المعتادة بأفكار إسلامية تجعل الطالب والأستاذ معا على صلة دائمة بالله عز وجل خلال دراسة المنهج ، وذلك بالطبع بالإضافة إلى المسائل والتطبيقات الرياضية الأخرى المعتادة كحل المعادلات الرياضية المختلفة وإيجاد جذورها بعدة طرق ، وجمع المتسلسلات ، وحساب قيم بعض الدوال المعقدة ، ومسائل التفاضل والتكامل ، وإيجاد المساحات المحصورة تحت بعض المنحنيات ، وحساب بعض المقاييس الإحصائية ، وإيجاد أكبر قيمة وأصغر قيمة في مجموعة معطاة من الأعداد أو ترتيبها ترتيبا تصاعديا أو تنازليا ، وبعض العمليات على المصفوفات ، وغير ذلك من التطبيقات .. وقد روعي أن تغطي أمثلة ومسائل الكتاب احتياجات طلاب من تخصصات مختلفة وخاصة طلاب العلوم والهندسة.

وعموما الكتاب ليس له متطلبات لقراءته سوى بعض رياضيات الجبر للمرحلة الثانوية ، ومادة الكتاب تعتبر منهجا لمقرر فصل دراسي واحد في برمجة الحواسيب بلغة C++.

فصول الكتاب

وقد رتبت فصول الكتاب بحيث يتمكن الطلاب من كتابة برامج وتشغيلها على الحاسب في أقرب وقت ممكن .. فيبدأ الفصل الأول بالحديث عن الخوارزميات وهي التعليمات والأوامر المرتبة التي تعطي للحاسب لحل مسألة ما ، وعن خرائط سير العمليات وهي المخططات السهمية أو الأشكال الرمزية لهذه التعليمات والأوامر .. ويعد هذا الفصل مدخلا لموضوع البرمجة .. ويناقش الفصل الثاني أساسيات لغة C++ بحيث يستطيع الطالب مع نهاية هذا الفصل كتابة

برامج قصيرة ، ويشتمل الفصل الثالث على عبارات التحكم للانتقال من أي خطوة إلى خطوة أخرى في البرنامج وكيفية كتابة شروط هذا الانتقال ... وأما الفصل الرابع عن عبارات التكرار فيشرح كيفية تكرير تنفيذ مجموعة من التعليمات والأوامر عددا معلوما من المرات ، أو إلى أن يتحقق شرط معين. ويتناول الفصل الخامس موضوع الدوال واستخداماتها. وأما الفصل السادس والأخير فهو خاص بالمنظومات.

ونظرا لأن المهارة والخبرة في البرمجة تحتاج لكثرة التدريب على كتابة برامج في تطبيقات مختلفة ، لذا كان التركيز في هذا الكتاب على تنوع وتعدد الأمثلة المحلولة ، ومسائل التمرينات في نهاية كل فصل.

وتوجد كتب ومراجع عديدة في موضوع البرمجة بلغة C++ ، ومن أحدث هذه الكتب وأيسرها تناولا لهذا الموضوع المرجع رقم [ ] في قائمة الكتب والمراجع المذكورة بنهاية الكتاب. والجزء الأكبر من مادة فصول هذا الكتاب يعد تعريبا لمحتويات النصف الأول تقريبا من هذا المرجع ، مع إضافة بعض الأمثلة والتمرينات والتطبيقات العملية. وكذلك يشتمل الكتاب على حلول كاملة لكثير من مسائل لتمرينات التي وردت في فصوله ، وذلك ليرجع إليها الطالب بعد محاولته حل المسائل إتماما للفائدة بإذن الله تعالى.

شكر وتقدير

ونود هنا أن نتقدم بشكرنا الجزيل لدار القلم للنشر والتوزيع بالكويت لتعاونها المستمر معنا في نشر الكتب الدراسية الجامعية التي تخدم بإذن الله تعالى قضية أسلمة العلوم الحديثة وتعريبها ، والتي نراها قضية حياة أو موت بالنسبة لهذه الأمة ، فجزى الله القائمين على هذه الدار خيرا على ما يقومون به في هذا السبيل من تقديم التسهيلات ، وتسخير الدار لتلك الغاية النبيلة.

ولا يفوتنا أن نشكر الأخ الفاضل الأستاذ / السيد البدوي محمد أحمد بقسم الرياضيات وعلم الحاسوب بجامعة الكويت على ما بذله من جهد كبير وما

تحلى به من صبر وأناة في طباعة هذا الكتاب حتى يخرج بهذه الصورة الطيبة التي تخدم عملية التعريب وتيسر نشر العلوم في آفاق أمتنا لتعود كما كانت في مكانة الصدارة بين الأمم. جعل الله ذلك الجهد في ميزان حسناته لمساهمته في تيسير سبل طلب العلم والتماسه.

كما نود كذلك أن نتقدم بخالص شكرنا وتقديرنا لإخواننا وأخواتنا وأبنائنا وبناتنا من طلاب وطالبات الجامعة الذين درسوا أو يدرسون معنا هذه المناهج ، فكانوا خير عون لنا على إعدادها وأسلمتها وتعريبها ، وذلك بحسن استيعابهم إياها ، ومناقشة موضوعاتها ، ودراسة كتبها ، وإبداء الملاحظات المستمرة حولها ، مما يعمل باستمرار على تطويرها .. وقد لمست والحمد لله تعالى إقبالا من هؤلاء الشباب من الطلاب والطالبات على دراسة هذه المناهج المؤسلمة المعربة ، وذلك أمر تنشرح له صدورنا ، وتَقَرُّ به أعيننا ، وهو كذلك أمر طبيعي في هذه الفترة من الصحوة الإسلامية التي تعيشها أمتنا الإسلامية {وما رميت إذ رميت ولكن الله رمى}(الأنفال : ١٧) ، ونسأل الله العلي القدير أن يبارك في هؤلاء الشباب ، وأن يجعل طلبهم للعلم ابتغاء مرضاته سبحانه ، وأن ينفعوا أمتهم بهذا العلم ، حتى تستعيد مكانتها ، وتسترد كرامتها ، وتملى كلمتها ، وتسعد الناس جميعا برسالتها {الذين إن مكناهم في الأرض أقاموا الصلاة وآتوا الزكاة وأمروا بالمعروف ونهوا عن المنكر ولله عاقبة الأمور} (الحج : ٤١).

فصل تمهيدي

نظرة عامة على البرمجة بلغة C++

Overview of Programming in C++

نود في هذا الفصل التمهيدي أن نأخذ فكرة عابرة غير تفصيلية عن الشكل العام لبرنامج متكامل مكتوب بلغة C++ ، أما التفاصيل فسنتابعها بإذن الله في بقية فصول الكتاب. وهذا البرنامج المتكامل الذي نلقي من خلاله نظرة عامة على هذه اللغة هو حل المسألة التالية.

مثال ١:

اكتب برنامجا بلغة C++ يقوم بإجراء الخطوات التالية:

(أ) يقرأ (reads) بيانات (data) كل موظف (employee) من موظفي إحدى الشركات ، حيث تشتمل هذه البيانات على:

- الرقم التعريفي للموظف **empNum**
  (employee identification number).
- معدل أجر الموظف في الساعة الواحدة **payRate**
  (employee's hourly pay rate).
- عدد الساعات التي عملها **hours**
  (number of hours worked) خلال الأسبوع.

(ب) يحسب (computes) أجر الموظف في الأسبوع **wages**

بناء على القاعدة التالية:

إذا كان عدد ساعات العمل في الأسبوع أكثر من ٤٠ فإن:

الأجر = (٤٠ × معدل الأجر) + (عدد الساعات – ٤٠) × ١,٥ × معدل الأجر.

wages = (40.0 × pay rate) + (hours worked – 40.0) × 1.5 × pay rate

وما عدا ذلك (otherwise) فإن

الأجر = عدد الساعات × في معدل الأجر

wages = hours worked × pay rate

(ج) يكتب (writes) بيانات الموظف (رقمه ومعدل أجره وعدد الساعات التي عملها) وأجره – الذي حسبه – في قائمة (list) عبارة عن الملف **payFile** (file) ، أي يحفظ (saves) البيانات المدخلة (input data) والأجور المحسوبة في ملف للتخزين الثانوي (secondary storage file).

(د) يحسب (computes) مجموع أجور الموظفين في الأسبوع **total** (total wages for the week).

(هـ) يعرض (displays) على الشاشة (screen) مجموع الأجور **total**.

ملاحظة ١:

يتوقف البرنامج عن قراءة بيانات الموظفين حين يقرأ صفرا 0 كقيمة مدخلة للرقم التعريفي للموظف ، أي أن إدخال هذه القيمة الصفرية يعني انتهاء البيانات.

ملاحظة ٢:

استخدم دالة **CalcPay** (function) لحساب أجر الموظف **wages** إذا علم معدل أجره وعدد الساعات التي عملها.

الحل:

```
//*********************************************************************
 //Payroll program
 //This program computes each employee's wages and
 //the total company payroll
//*********************************************************************
#include <iostream>
#include <fstream>    // For file I/O

using namespace std;

void CalcPay( float, float, float &   );

const float MAX_HOURS = 40.0;       // Maximum normal work hours
const float OVERTIME = 1.5;         // Overtime pay rate factor

int main()
{
   float      payRate;   // Employee's pay rate
   float      hours;     // Hours worked
```

```
  float       wages;      // Wages earned
  float       total;      // Total company payroll
  int         empNum;     // Employee ID number
  ofstream payFile;       // Company payroll file

  payFile.open("payfile.dat");          // Open the output file
  total = 0.0;                          // Initialize total
  cout << "Enter employee number: ";    // Prompt
  cin >> empNum ;                       // Read employee ID no.
  while (empNum != 0)                   // While employee number
  {                                     //  isn't zero
     cout << "Enter pay rate: ";        // Prompt
     cin >> payRate;                   // Read hourly pay rate
     cout << "Enter hours worked: ";    // Prompt
     cin >> hours;                     // Read hours worked
     CalcPay(payRate, hours, wages);   // Compute wages
     total = total + wages ;           // Add wages to total
     payFile << empNum << payRate      // Put results into file
             << hours << wages ;
     cout << "Enter employee number: "; // Prompt
     cin >> empNum;                    // Read ID number
  }
  cout << "Total payroll is "           // Print total payroll
       << total << endl;                // on screen
  return 0;                             // Indicate successful
 }                                      // completion

// ************************************************************************

void CalcPay(  /* in */  float  payRate,     // Employee's pay rate
               /* in */  float  hours,       // Hours worked
               /* out */ float& wages )              // Wages earned

 // CalcPay computes wages from the employee's pay rate
 // and the hours worked, taking overtime into account

{
  if (hours > MAX_HOURS)                // Is there overtime?
     wages = (MAX_HOURS * payRate +     // Yes
          (hours - MAX_HOURS) * payRate * OVERTIME;
  else
     wages = hours * payRate;           // No
}
```

ملاحظات عامة على البرنامج:

- **العبارات والعلامات والملاحظات المكتوبة يمين الرمزين // يطلق عليها تعليقات** (comments) **، وهي تستخدم لتوضيح خطوات البرنامج والمساعدة في فهمه ، وهي لا تتقيد بقواعد لغة البرمجة** C++ **، ويتجاهلها البرنامج المترجم** (compiler)**. وكذلك الكلمات والعبارات المحصورة**

بين الرموز الأربعة /*.....*/ تعد تعليقات (comments) ويتجاهلها البرنامج المترجم.

- القيم الثابتة في البرنامج ، أي التي لا تتغير طوال تنفيذ البرنامج ، يشار إليها بالكلمة **const** ، وهي اختصار كلمة **constant**.
- المتغيرات والثوابت التي لا تأخذ إلا قيما صحيحة (integer) نشير إليها بالكلمة **int** ، بينما تلك التي تأخذ قيما تحتمل وجود كسور عشرية نشير إليها بالكلمة **float**.
- لحساب مجموع الأجور **total** فإننا نبدأ بوضع القيمة الابتدائية (initial value) صفر 0.0 في هذا المتغير total ، وكلما حسبنا أجر موظف wages فإننا نضيفه إلى هذا المتغير بالعبارة

```
total = total + wages ;
```

إلى أن نحصل على المجموع النهائي المطلوب.

- للقراءة نستخدم الأمر cin ، وللكتابة (للطباعة) نستخدم cout.
- عندما يكون البرنامج في حاجة إلى قراءة بعض البيانات ومستعداً لاستقبال قيمها ، فإنه يطبع رسالة تنبيهية / رسالة حث (prompting message / prompt) للمستخدم (user) لإدخال القيم المطلوبة. فمثلا حين يطلب إدخال رقم الموظف فإنه يطبع رسالة تنبيهية مثل: Enter employee number هكذا:

```
Cout << " Enter employee number: " ;
```

وبعد أن يقوم المستخدم بإدخال رقم الموظف ، فإن البرنامج يقرأ هذه القيمة ويقوم بتخزينها في الذاكرة (memory) في المكان المحجوز تحت اسم empNum حين ينفذ الأمر التالي في البرنامج:

```
cin >> empNum ;
```

- طالما / بينما (while) يكون رقم الموظف عدداً غير صفري فإن البرنامج ينفذ مجموعة الأوامر / العبارات المحصورة بين القوسين

```
{
  ..... ;
  ..... ;
  ..... ;
}
```

ويستمر البرنامج في تكرار (repeating) تنفيذ هذه المجموعة من العبارات (هذه العروة loop) إلى أن يقرأ البرنامج عدداً صفرياً (لرقم الموظف) فيتوقف عن تنفيذ هذه العروة ، ويخرج منها ليطبع القيمة النهائية لمجموع الأجور total.

- حساب أجر أي موظف يتم باستدعاء الدالة **CalcPay** عن طريق تنفيذ أمر الاستدعاء

```
CalcPay (payRate, hours, wages) ;
```

حيث يخرج البرنامج من الدالة الرئيسية main إلى الدالة CalcPay ، وبعد الانتهاء من تنفيذها يعود إلى الدالة main.

- نعتبر أن نظام التشغيل (operating system) هو الذي يستدعي الدالة الرئيسية (the main function) ، ويتوقع النظام أن تعيد (return) هذه الدالة **main** قيمةً ما عندما ينتهي تنفيذها (execution). وقد أصبح من المصطلح / المتفق عليه (by convention) عموما أن تعيد الدالة القيمة صفر 0 عندما يتم تنفيذها بالصورة المطلوبة دون حدوث أي مشاكل. أما إذا حدث أي خطأ أو أي مشكلة فإن الدالة تعيد أي قيمة أخرى (مثل 1 أو 2 أو ....). ولذلك فإن آخر عبارة في الدالة **main** هي:

```
return 0 ;
```

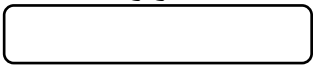
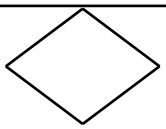
الفصل الأول

الخوارزميات وخرائط سير العمليات
Algorithms and Flowcharts

مفهوم الخوارزميات وخرائط سير العمليات :

الحاسب آلة وفق الله سبحانه وتعالى الإنسان لصنعها حيث تعينه على إنجاز العمليات الحسابية والمنطقية بسرعة كبيرة وبدقة بالغة ، وذلك حسب التعليمات والأوامر التي يعطيها الإنسان للحاسب والتي توضح الخطوات التنفيذية التي يقوم بها الحاسب على الترتيب لحل مسألة معينة ، أما بدون هذه التعليمات والأوامر فإن الحاسب لا يستطيع حل أي مسألة مهما كانت بسيطة لأنه آلة صماء لا تستطيع أن تفكر كيف تحل المسألة ، ولذلك فمن الخطأ تسمية الحاسب بالعقل الإلكتروني كما يسميه البعض لأن من مميزات العقل التفكير ، والحاسب لا يستطيع أن يفكر وإنما يستطيع أن يحسب أو يقارن أو يخزن معلومات أو يستعيدها .. الخ حسب الأوامر المعطاة له ، فهو يقوم بتنفيذ نفس العمليات الحسابية والمنطقية التي يستطيع الإنسان أن يقوم بها ولكن الحاسب يستغرق وقتا أقل بكثير من الوقت الذي يستغرقه الإنسان ، وهذه السرعة الكبيرة هي من أهم مميزاته التي تجعله يستخدم في تطبيقات كثيرة .. والتعليمات والأوامر التي نعطيها للحاسب بصورة واضحة ومتسلسلة ومترابطة منطقيا للوصول إلى حل مسألة ما تسمى خوارزمية (Algorithm) المسألة وذلك نسبة إلى العالم المسلم محمد بن موسى الخوارزمي الذي ينسب له الفضل في وضع أسس حل المسائل بشكل تتابعي .. ويمكن تمثيل هذه التعليمات المتتابعة (الخوارزمية) بمخطط سهمي يسمى مخطط سير العمليات أو خريطة سير العمليات (Flowchart) وهو عبارة عن مجموعة من الأشكال الرمزية (كالمستطيل ومتوازي الأضلاع) المصطلح عليها بحيث أن كل شكل يمثل خطوة في تنفيذ الحل وبداخل كل شكل تذكر العملية المطلوب تنفيذها في

هذه الخطوة ، وتصل مجموعة من الأسهم بين هذه الأشكال لتحديد تتابع الخطوات ، حيث يشير اتجاه السهم إلى الخطوة التالية في الحل. وفيما يلي الاصطلاحات الأساسية التي ستستخدم في مخططات سير العمليات في هذا الكتاب.

الرمز	معناه
	بدء (Start) أو انتهاء وتوقف (Stop) خريطة سير العمليات
	إدخال (Input) معلومات إلى الحاسب (قراءة) أو إخراج (Output) معلومات من الحاسب (طباعة أو كتابة)
	عملية حسابية (Computation) أو معالجة للبيانات (Processing)
	اتخاذ قرار (Decision) بناءا على تحقق شروط (Conditions) معينة
	اتجاه التدفق (Flow Line) لتوضيح تتابع خطوات الحل
	صلة (Connection) بين نقطتين في الرسم

وفي هذا الفصل نستعرض بعض الأمثلة التي توضح مفهوم الخوارزميات وخرائط سير العمليات لحل بعض المسائل.

على أن هذه الخوارزميات أو الخرائط لا نعطيها للحاسب مباشرة لتنفيذها وإنما يجب ترجمتها أولا إلى لغة خاصة يقبلها الحاسب ، أي يكون مصمما ومعدا لتقبلها ، وتقدم على وسائل خاصة يمكن إدخالها للحاسب .. والخوارزمية أو الخريطة بعد ترجمتها إلى هذه اللغة الخاصة تسمى برنامجا (Program) . وتوجد حاليا عدة لغات يمكن استخدامها لإدخال برامج إلى الحواسيب ، ولغة الباسكال

هي إحدى هذه اللغات ، وشرح القواعد الأساسية لهذه اللغة هو موضوع هذا الكتاب.

خطوات حل مسألة باستخدام الحاسب :

أولا : تعريف المسألة رياضيا وتحديد طريقة حلها والقوانين التي ستستخدم لذلك ، بمعنى أنه يجب أن يكون واضحا لدينا :

(أ)المدخلات (Input) أي البيانات (Data) التي تكون معلومة لدينا وندخلها للحاسب.

(ب)المخرجات (Output) أي النتائج (Results) النهائية المطلوب من الحاسب الحصول عليها بعد حل المسألة ثم إخراجها لنا.

(ج)الطريقة الرياضية أو القوانين الرياضية المطلوب من الحاسب استخدامها لحل المسألة.

ثانيا : كتابة خوارزمية المسألة أو رسم خريطة العمليات وذلك لتحديد الخطوات المتتابعة المطلوب من الحاسب تنفيذها بالترتيب لحل المسألة ابتداء من قراءة البيانات وحتى طباعة النتائج.

ثالثا : ترجمة الخوارزمية أو خريطة سير العمليات إلى برنامج بلغة يقبلها الحاسب ، ثم طباعة البرنامج أو نقله على وسيلة إدخال يقبلها الحاسب.

أي أن الخوارزمية وخريطة سير العمليات والبرنامج هي ثلاث صور مختلفة للشيء نفسه ويمكن لحل مسألة ما باستخدام الحاسب كتابة البرنامج مباشرة دون كتابة الخوارزمية أو رسم خريطة سير العمليات أولا ، إلا أن رسم هذه الخريطة قبل كتابة البرنامج له عدة فوائد منها الحصول على صورة شاملة لخطوات حل المسألة والعلاقة بين هذه الخطوات ، وسهولة اكتشاف الأخطاء المنطقية في الحل ، وكذلك سهولة عمل تعديلات في هذه الخطوات ، بالإضافة إلى الاحتفاظ بهذه الخرائط كمراجع يسهل الرجوع إليها مستقبلا لمراجعة خطوات حل المسألة أو استخدامها لحل مسائل أخرى مشابهة لها أو إجراء تعديلات عليها.

أمثلة :

في كل مثال من مجموعة الأمثلة التالية المطلوب كتابة خوارزمية المسألة مع رسم خريطة سير العمليات بعد تعريف المسألة رياضيا ، أما كتابة البرامج فستناقش في الفصول التالية بإذن الله تعالى.

مثال ١-١ :

إيجاد الجذرين الحقيقيين لمعادلة جبرية من الدرجة الثانية ذات معاملات حقيقة (Real).

تعريف المسألة رياضيا :

نفرض أن المعادلة هي :

$$ax^2 + bx + c = 0, \qquad a \neq 0$$

حيث المعاملات a , b , c **كلها حقيقية.**

المدخلات : **قيم المعاملات** a , b , c .

المخرجات : **قيمة كل من جذري المعادلة** x_1 , x_2 **ويمكن الحصول على قيمتيهما باستخدام**

القانون :

$$x_1 , x_2 = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$$

مع تحقق الشرط

$$b^2 - 4ac \geq 0$$

الخوارزمية :

١- ابدأ.

٢- اقرأ قيم المعاملات a , b , c.

٣- إذا كانت $a = 0$ **توقف. (المعادلة ليست من الدرجة الثانية)**

٤- إذا كانت $b^2 - 4ac < 0$ **توقف. (الجذران ليسا حقيقيين)**

٥- احسب قيمة الجذر الأول

$$x_1 = (-b + \sqrt{b^2 - 4ac}\ ) / (2a)$$

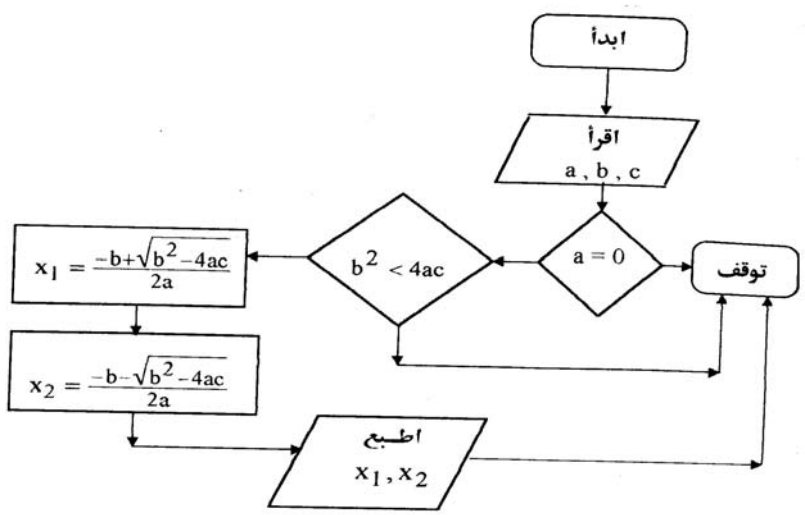
٦- احسب قيمة الجذر الثاني

$$x_2 = (-b - \sqrt{b^2 - 4ac}\ ) / (2a)$$

٧- اطبع قيمة كل من الجذرين x_1 , x_2.

٨- توقف.

خريطة سير العمليات



ملاحظتان :

١-في أي خريطة سير عمليات ليس من الضروري رسم رمز البدء (start) حيث أنه من المفهوم أن أول خطوة تنفيذية في الحل موجودة داخل ذلك الرمز الذي يخرج منه سهم ولا يدخل عليه أي سهم ، وبالمثل في خوارزمية المسألة يمكن الاستغناء عن الخطوة الأولى (ابدأ) ، ومفهوم أن أول خطوة تنفيذية هي أول خطوة مكتوبة في الخوارزمية.

٢-المستطيلان المرسومان لحساب الجذرين x_1 , x_2 يمكن ضمهما معا في مستطيل واحد تكتب داخله معادلتا x_1 , x_2 واحدة تحت الأخرى.

مثال ١-٢ :

اكتب خوارزمية وارسم خريطة سير عمليات لبرنامج يحسب زكاة المال لمدخرات شخص وذلك باتباع الخطوات التالية :

(١) **قراءة قيمتين :** (أ) **قيمة المدخرات السنوية الكلية** TAS (Total Annual Savings)

(ب) **قيمة النصاب** [*] A

(٢) **حساب قيمة الزكاة** Z **والتي تقدر بربع العشر من قيمة المبلغ المدخر** TAS **(والذي حال عليه الحول وكان فارغا عن الدين والحاجات الأصلية) إذا بلغ هذا المبلغ النصاب** A **، أما إذا لم يبلغ النصاب فلا زكاة عليه.**

(٣) **طباعة قيمة كل من المدخرات** TAS **والزكاة** Z **.**

تعريف المسألة رياضيا :

المدخلات : **قيمة المدخرات** TAS **وقيمة النصاب** A.

المخرجات : **قيمة المدخرات** TAS **وقيمة الزكاة** Z **.**

القانون المستخدم :

$$Z = \begin{cases} TAS/40 & \text{إذا تحقق الشرط } TAS \geq A \\ 0 & \text{إذا تحقق الشرط } TAS < A \end{cases}$$

الخوارزمية :

١- **اقرأ قيمة كل من** TAS , A

٢- **إذا كانت** $TAS < A$ **فإن** $Z = 0$

٣- **إذا كانت** $TAS \geq A$ **فإن** $Z = TAS/40$

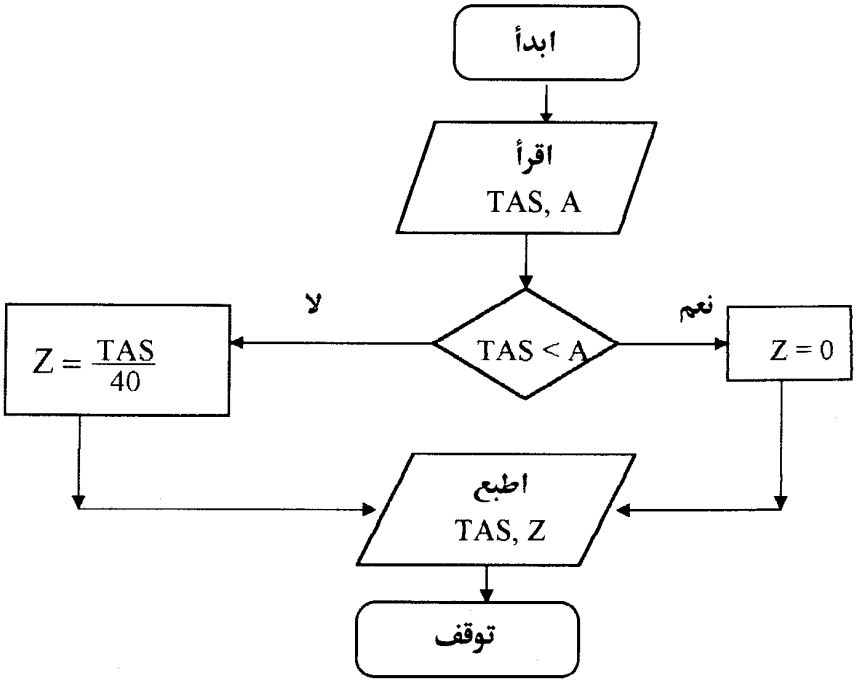
٤- **اطبع قيمة كل من** TAS , Z

٥- **توقف**

---

[*] يقدر النصاب بسعر حوالي ٨٥ جرام من الذهب الخالص.

خريطة سير العمليات :



في برامج بعض المسائل تكون بعض البيانات ضمن كل من المدخلات والمخرجات (مثل TAS في هذا المثال) حيث تطبع مع نتائج البرنامج زيادة في الإيضاح.

مثال ١-٣ :

حساب وطباعة جدول لمربعات الأعداد الصحيحة من ٢ إلى ٥٠.

تعريف المسألة رياضيا :

القانون : $y = x^2$; $x = 2 , 3 , 4 , ... , 50$

المدخلات : **لا يوجد**

المخرجات : **جدول قيم** x, y **والتي تعطى بالقانون المذكور سابقا.**

في هذا المثال لا تعطى قيم x **للحاسب في صورة بيانات يقرأها وإنما ستعطى له طريقة استنتاج قيم** x **المتتالية وذلك بأن نذكر له أن أول قيمة للمتغير** x **هي** 2 **ثم عليه أن يضيف** 1 **باستمرار لكل قيمة للمتغير** x **ليحصل على القيمة التالية إلى أن يصل إلى آخر قيمة وهي** 50 **، وعادة نتبع مثل هذه الطريقة في حالة البيانات التي تتغير تبعا لعلاقة معينة تربطها بعضها ببعض ، أما إذا لم توجد مثل هذه العلاقة فإن البيانات عادة تعطى كلها في صورة مدخلات يقرأها الحاسب.**

الخوارزمية :

١-اجعل $x = 2$

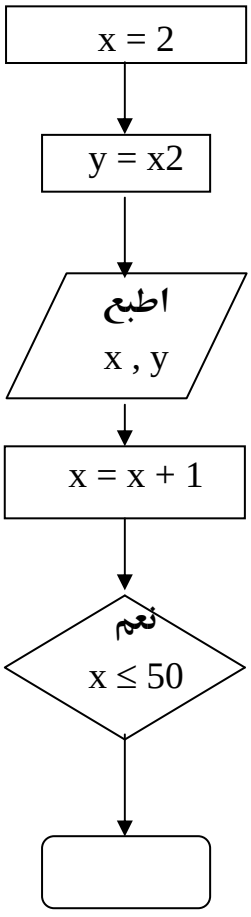
٢-اجعل $y = x^2$

٣-اطبع قيمتي x , y

٤-اجعل $x = x + 1$ **(أي زد قيمة** x **بإضافة** 1 **إلى قيمتها الحالية)**

٥-إذا كان $x \leq 50$ **اذهب إلى الخطوة رقم ٢**

٦-توقف



خريطة سير العمليات :

يلاحظ أن العبارة $x = x + 1$ **ليست معادلة ، وأن** x **التي في الطرف الأيمن تختلف عن** x **التي في الطرف الأيسر ،**

فبينما x التي في الطرف الأيمن تعني القيمة الحالية للمتغير x فإن x التي في الطرف الأيسر تعني القيمة الجديدة للمتغير x والتي نحصل عليها بإضافة 1 للقيمة الحالية.

لا توقف

وبالنسبة لتنفيذ الحاسب لخطوات هذا الحل إلى أن يتوقف نلاحظ ما يلي:

بعد أن يطبع الحاسب أول قيمتين في الجدول المطلوب وهما 4 , 2 تنفيذا للخطوة الثالثة ، فإنه ينتقل إلى الخطوة الرابعة وهي زيادة قيمة x لتصبح $x = 2 + 1 = 3$ ولذلك فإن الشرط $x \leq 50$ المذكور في الخطوة الخامسة يكون متحققا ولذلك ينتقل إلى الخطوة رقم ٢ ليحسب قيمة جديدة للمتغير y حسب العلاقة $y = 3^2 = 9$.

وبعد ذلك ينتقل تلقائيا إلى الخطوة التالية مباشرة وهي الخطوة رقم ٣ (طباعة x , y أي طباعة 9 , 3) لأنه من القواعد العامة في تنفيذ البرامج أنه ما لم يكن هناك أمر بالانتقال إلى خطوة معينة سابقة أو لاحقة (مثل اذهب إلى الخطوة رقم ك حيث ك ليس رقم الخطوة التالية) فإن الانتقال يكون إلى الخطوة التالية مباشرة وليس إلى الخطوة التي جاء منها سابقا (وهي الخطوة رقم ٥ في هذا المثال).

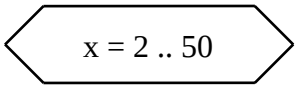
ثم ينتقل الحاسب إلى الخطوة التالية وهي رقم ٤ وبعد تنفيذها تصبح قيمة x

$$x = 3 + 1 = 4$$

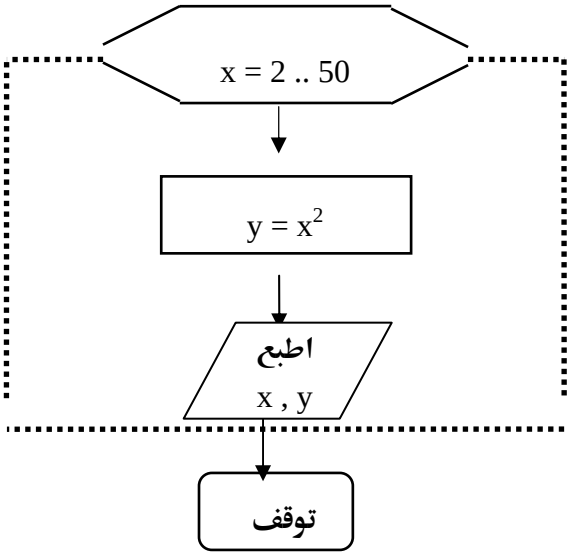
ومرة أخرى تكون إجابة السؤال : هل $x \leq 50$ ؟ (المذكور في الخطوة التالية رقم ٥) نعم ، وينتقل الحاسب إلى الخطوة رقم ٢ لينفذها وهكذا يستمر في تنفيذ خطوات البرنامج وطباعة قيم x , y المتتالية إلى أن تصبح

قيمة x تساوي 51 ويصبح الشرط $x \leq 50$ المذكور في الخطوة رقم ٥ غير متحقق ، فيتم الانتقال إلى الخطوة رقم ٦ ، والتي توقف تنفيذ البرنامج.

ملاحظة : في خريطة سير العمليات يمكننا جمع الثلاثة رموز/أشكال التي تعطي القيمة الابتدائية للمتغير x (المستطيل : $x = 2$) والزيادة المنتظمة للمتغير x بإضافة 1 (المستطيل : $x = x + 1$) وشرط عدم تعدي قيمة المتغير x القيمة النهائية (المعين : $x \leq 50$) في رمز/شكل واحد ، هكذا :



وفي هذه الحالة ترسم الخريطة بالشكل التالي :



مثال ١-٤ :

طباعة مجموع مربعات الأعداد الصحيحة من ٢ إلى ٥٠ .

تعريف المسألة رياضيا :

$$S = \sum_{x=2}^{x=50} x^2 = 2^2 + 3^2 + 4^2 + \cdots + 50^2$$

الخوارزمية :

١- **اجعل** $S = 0$

٢- **اجعل** $X = 2$

٣- **اجعل** $S = S + x^2$

٤- **اجعل** $x = x + 1$

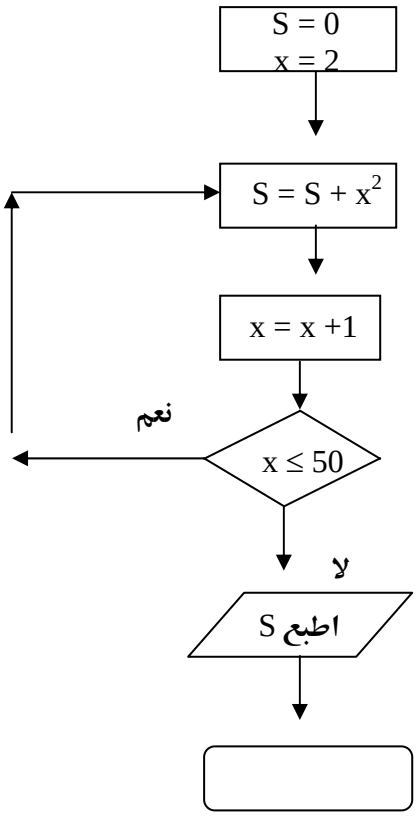
٥- **إذا كانت** $x \leq 50$ **اذهب إلى الخطوة رقم ٣**

٦- **اطبع قيمة** S

٧- **توقف.**

خريطة سير العمليات :

في هذا المثال بدأنا بإعطاء رمز المجموع الكلي S القيمة الابتدائية صفر ، ثم أخذنا نضيف على التوالي مربع العدد ٢ ثم مربع العدد ٣ ثم مربع العدد ٤ وهكذا حتى مربع العدد ٥٠ .. ثم طبعنا القيمة النهائية فقط للمتغير S وهي تمثل المجموع المطلوب ، أي أننا طبعنا قيمة واحدة فقط لأنه لا يهمنا معرفة المجاميع الجزئية (القيم المتتالية للمتغير S) ، ولذلك فإن الأمر (اطبع S) وضع خارج العروة (loop) التي تمثل الخطوات من الخطوة رقم ٣ إلى الخطوة رقم ٥ ، بينما في المثال السابق كان علينا أن نطبع كل قيمة محسوبة للمتغير y ولذلك فإن الأمر (اطبع x , y) وضع داخل العروة التي تمثل الخطوات من الخطوة رقم ٢ إلى الخطوة رقم ٥.



نعتقد أن المقابلة بين خوارزمية أي مسألة وخريطة سير العمليات المناظرة لها أصبحت بإذن الله تعالى واضحة ، ولذلك ففي المثال التالي والذي هو امتداد للمثال ١-٢ سنكتفي برسم خريطة سير العمليات ، خاصة وأنه عندما يكون عدد الخطوات في حل أي مسألة كبيرا فإن خريطة سير العمليات تعطي صورة أوضح للحل من خوارزمية المسألة.

مثال ١-٥ :

ارسم خريطة سير العمليات لبرنامج يحسب زكاة المال Z (أنظر مثال ١-٢) لألف شخص ، وذلك بقراءة قيمة النصاب A أولا ، ثم بتنفيذ الخطوات التالية لكل شخص :

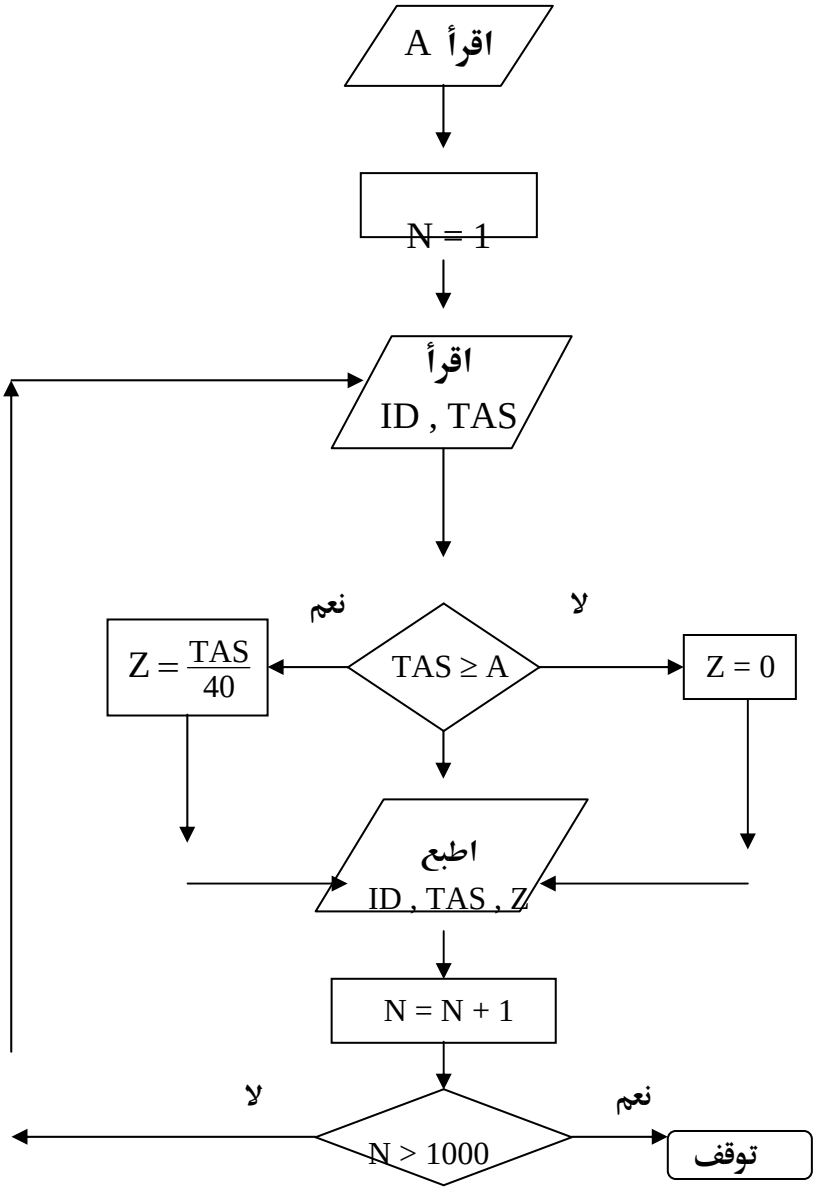
(أ)قراءة قيمة ID (رقم تعريفي للشخص identification Number) وقيمة TAS (المدخرات السنوية الكلية).

(ب)حساب قيمة الزكاة Z.

(ج)طباعة قيم ID , TAS , Z.

خريطة سير العمليات :

هذه الخريطة امتداد لخريطة مثال ١-٢ الذي يحسب زكاة المال لشخص واحد فقط ، وقد أضفنا هنا متغيرا جديدا N يسمى عدادا (Counter) لنحسب به عدد الأشخاص. وفي البداية نعطي N القيمة ١ ، وبعد حساب وطباعة قيمة الزكاة للشخص الأول نزيد قيمة N بواحد ، وهكذا إلى أن نحسب ونطبع قيمة الزكاة لآخر شخص. ثم حين تزاد قيمة N بواحد تصبح مساوية ١٠٠١ وبذلك نخرج خارج عروة قراءة ID , TAS وحساب Z وطباعة ID , TAS , Z لنوقف تنفيذ البرنامج.


اقرأ A
N = 1
اقرأ
ID , TAS
نعم
لا
$Z = \frac{TAS}{40}$
TAS ≥ A
Z = 0
اطبع
ID , TAS , Z
N = N + 1
لا
نعم
N > 1000
توقف

مثال ١-٦ :

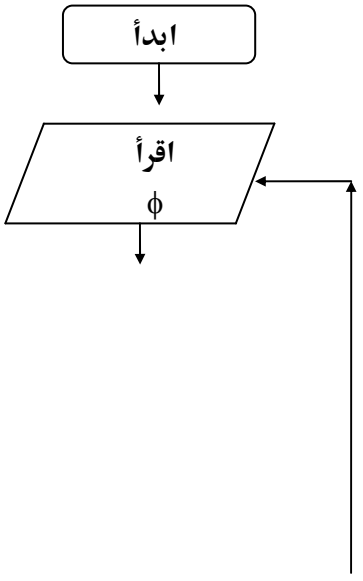
أعطيت مجموعة من البيانات على سطور أو بطاقات بحيث أن كل بطاقة عليها درجة حرارة ϕ بالتقدير الفهرنهيتي. ارسم خريطة سير عمليات لبرنامج يقرأ كل درجة ϕ ثم يحولها إلى التقدير المئوي θ حسب العلاقة

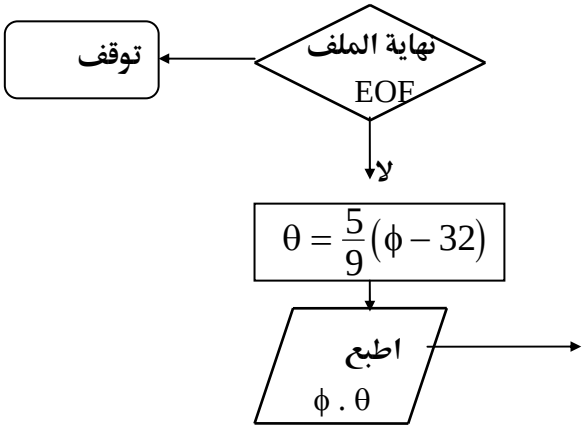
$$\theta = \frac{5}{9}(\phi - 32)$$

ثم يطبع قيمة كل من ϕ والدرجة المقابلة θ ، ثم ينتقل ليقرأ الدرجة التالية ويحولها وهكذا إلى أن ينتهي من كل الدرجات.

ملاحظة : في هذا المثال لا نعرف مبدئيا عدد السطور / البطاقات أي عدد درجات الحرارة ϕ المطلوب تحويلها ، وبالتالي فلا يمكن استخدام عداد كالذي استخدمناه في المثال السابق لإيقاف تنفيذ البرنامج .. وإنما في مثل هذه الحالات التي لا نعرف فيها مبدئيا عدد البيانات المطلوب إدخالها للحاسب يمكننا أن نضع بعد آخر قيمة من قيم البيانات قيمة أخرى جديدة مميزة عن كل قيم البيانات وبعد قراءة أي قيمة – أثناء تنفيذ البرنامج – نسأل : هل هذه القيمة هي آخر قيمة في المجموعة الجديدة للبيانات (وهي المجموعة الأصلية مضافا إليها القيمة الجديدة) ؟ فإذا كانت الإجابة نعم فمعنى ذلك أننا قد انتهينا من قراءة كل البيانات ويمكننا إيقاف تنفيذ البرنامج أو طباعة أي مخرجات نحتاجها ثم إيقافه.

وعادة نسمي المجموعة الكلية للبيانات ملفا (File) **وسنشير في خريطة سير العمليات إلى سؤالنا السابق : هل هذه هي آخر قيمة من المجموعة ؟ برمز المعين وداخله الحروف** EOF (**أي** End Of File) **أو الكلمتان : نهاية الملف.**




نهاية الملف
EOF
توقف
لا
$\theta = \frac{5}{9}(\phi - 32)$
اطبع
ϕ . θ

تمرينات رقم ١

١-١ تحسب زكاة الغنم من الجدول التالي :

عدد الأغنام N	أقل من ٤٠	٤٠-١٢٠	١٢١-٢٠٠	٢٠١-٣٠٠	أكثر من ٣٠٠
زكاة الغنم ZS	صفر	١	٢	٣	في كال مائة شاة

ارسم خريطة سير عمليات لبرنامج يقرأ عدد الأغنام N عند أحد الأشخاص ثم يحسب زكاة الغنم ZS لهذا الشخص ويطبع قيمتي ZS , N.

١-٢ تقدر زكاة الثمار والفاكهة ZF بعُشْر الثمار التي سُقيت طبيعيا بدون استعمال آلة (أي فيما سقت السماء والعيون والأنهار) وبنصف العشر بالنسبة للثمار التي سقيت بآلة (أي فيما سقي بالنضح أو بالسانية أي البعير الذي يسقى به الماء من البئر) أو بماء مشترى ، وذلك إذا بلغت الثمار F النصاب B ، أما إذا لم تبلغ النصاب فلا زكاة عليها.

ارسم خريطة سير عمليات لبرنامج يقرأ قيمتي B , F وكذلك قيمة رقم ثنائي I يدل على طريقة سقي الثمار ، حيث يأخذ القيمة ١ إذا كان السقي طبيعيا بدول استعمال آلة ، والقيمة صفر إذا كان السقي بآلة ، ثم يقوم البرنامج بحساب قيمة الزكاة ZF.

١-٣ فرض رسول الله صلى الله عليه وسلم زكاة الفطر من رمضان صاعا من تمر أو صاعا من شعير على كل حر أو عبد ذكر أو أنثى من المسلمين ، وذلك طهرة للصائم من اللغو والرفث وطعمة للمساكين ، يخرجها الشخص عن نفسه وعن كل من تلزمه نفقتهم من الزوجة والأقارب وهم : الوالدان الفقيران ، والأولاد الذكور الذين لا مال لهم حتى يشتغلوا بمعاشهم ، وكذلك الإناث إلى أن يدخل بهن الزوج ، والمماليك والخدم الذين التزم المخدوم بنفقتهم ومعاشهم ...

ويجوز إخراج قيمة زكاة الفطر نقدا وقدرها في الكويت دينار عن الفرد الواحد.

نفرض أنك قد أعطيت مجموعة من البطاقات تخص مجموعة من الأسر (بطاقة لكل أسرة) ، ومن بين البيانات المذكورة في كل بطاقة عددان :

العدد الأول I : هو رقم تعريفي للأسرة.

العدد الثاني J : هو عدد أفراد الأسرة (الشخص وكل من تلزمه نفقتهم).

ارسم خريطة سير عمليات لبرنامج يقرأ هذه البيانات ويحسب :

(أ) القيمة النقدية Z بالدينار لزكاة الفطر لكل أسرة ، مع كتابة النتائج في صورة جدول يعطي رقم الأسرة I والقيمة النقدية للزكاة Z.

(ب) القيمة النقدية الكلية TZ بالدينار لزكاة الفطر لمجموع هذه الأسر.

١-٤ نفرض أنه قد أُعدت بطاقة لكل كتاب في معرض الكتاب الإسلامي بحيث تحتوي البطاقة على بعض البيانات الخاصة بالكتاب ومن بينـها : رقم تعريفي للكتاب I ، ورقم موضوع الكتاب J والذي يأخذ إحدى القيم التالية:

الموضوع	العقيدة	التفسير	الحديث	السيرة	الفقه	موضوعات أخري
رقم الموضوع	١	٢	٣	٤	٥	٦

ارسم خريطة سير عمليات لبرنامج يقرأ هذه البيانات ، ويحسب :

(أ) العدد الكلي للكتب الموجودة بالمعرض N.

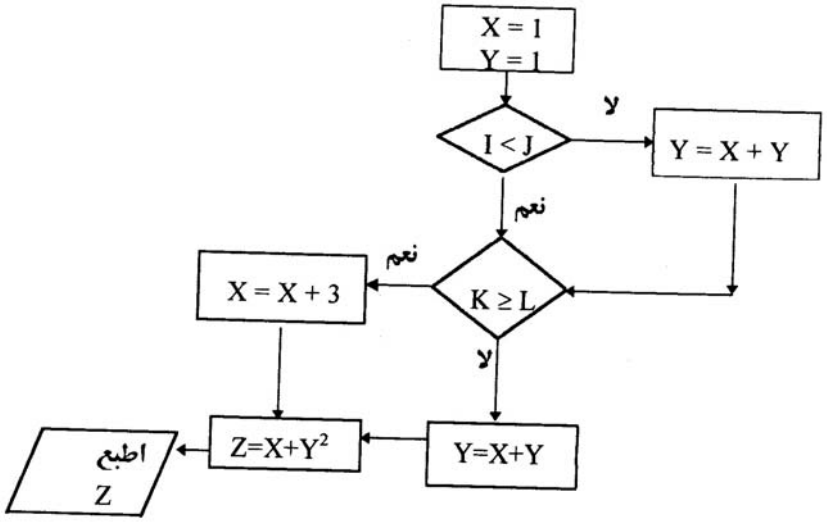
(ب) عدد كتب التفسير والحديث M.

١-٥ تعد الدعوة إلى تحديد النسل من الأسلحة التي يستخدمها أعداء الإسلام الماكرون ضد الشعوب الإسلامية بقصد تقليل أعداد سكانها. نفرض أن تعدادي سكان البلد A والبلد B (أو الأكثرية A والأقلية B في بلد ما) في الوقت الحالي هما أربعون مليونا وثلاثة ملايين على الترتيب. ونفرض أن معدل تزايد سكان A السنوي يساوي ١٪ فقط (بسبب سياسة تحديد النسل) ، بينما معدل تزايد سكان B السنوي يساوي ١٠٪ (بسبب تشجيع النسل والهجرة من الخارج).

ارسم خريطة سير عمليات لبرنامج يحسب التعداد السنوي لسكان كل من البلدين A و B إلى أن يزيد عدد سكان B على عدد سكان A ، وكذلك عدد السنوات اللازمة لتحقق هذه النتيجة.

١-٦ خريطة سير العمليات المرسومة فيما يلي تمثل جزءا من أحد البرامج ، والمطلوب حساب قيمة Z التي ستُطبع في كل حالة من الحالات الأربع التالية :

	I	J	K	L	Z
i)	2	3	3	2	
ii)	3	2	3	2	
iii)	3	2	2	3	



iv)	2	3	2	3	

١-٧ ارسم خريطة سير عمليات لبرنامج يحسب ويطبع قيمة S في كل من الحالات التالية:

(i) $S = \frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots + \frac{1}{115}$

(ii) $S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \cdots + \frac{99}{100}$

(iii) $S = 2^3 + 4^3 + 6^3 + \cdots + 300^3$

١-٨ ارسم خريطة سير عمليات لبرنامج يقرأ قيمة عدد صحيح K ثم يحسب حاصل ضرب الأعداد الصحيحة من 1 إلى K (أي يحسب !K) ويطبع قيمة كل من K وحاصل الضرب.

١-٩ ارسم خريطة سير عمليات لبرنامج يحسب قيم y المناظرة لقيم θ التالية :

$$\theta = 0, \frac{\pi}{8}, \frac{2\pi}{8}, \frac{3\pi}{8}, \cdots, \pi \qquad (\pi = 3.141593)$$

حيث تعطى y بدلالة المتغير θ بالعلاقة

$$y = \frac{\pi}{2} + \sin^2\left(3\theta + \frac{\pi}{4}\right)$$

١-١٠ تستخدم الخوارزمية التالية والمعروفة باسم خوارزمية التنصيف (Bisection Algorithm) في حل المعادلات f(x) = 0.

ارسم خريطة سير عمليات توضح خطوات هذه الطريقة.

١- اقرأ قيم a , b , ε.

٢- احسب $c = \frac{a+b}{2}$.

٣- احسب قيمة f(c).

٤- إذا كان |f(c)| < ε اطبع قيمة c وتوقف.

٥- إذا كان f(c).f(a) > 0 اجعل a = c (أي اعط قيمة c للمتغير a) ، وفيما عدا ذلك اجعل b = c.

٦- اذهب إلى الخطوة رقم ٢.

(ملاحظة : فكرة هذه الطريقة سنتناولها بإذن الله تعالى في المسألة رقم ٤-٤٤ في تمرينات الفصل الرابع).

١-١١ تستخدم الخوارزمية التالية والمعروفة باسم خوارزمية نيوتن - رافسون (Newton-Raphson Algorithm) في حل المعادلات f(x) = 0.

ارسم خريطة سير عمليات توضح خطوات هذه الطريقة.

١- اقرأ قيم x_0 , ε , n_{max}

٢- اجعل n = 0

٣- اجعل $x = x_0$

٤- احسب $\Delta = \frac{f(x)}{f'(x)}$

٥- احسب $x = x - \Delta$

٦- اجعل $n = n + 1$

٧- اطبع قيمة كل من x , n.

٨- إذا كان $|\Delta| \le \varepsilon$ أو $n \ge n_{max}$ توقف

٩- ارجع إلى الخطوة رقم ٤.

(ملاحظة : شرح هذه الطريقة سنناقشه بإذن الله تعالى في مثال ٤-١١ في الفصل الرابع).

١-١٢ يدرس مجموعة من الطلاب مقرري الدراسات الإسلامية وعلم الحاسب ويحصل الطالب في نهاية دراسته على تقدير عام S (مقبول) أو U (غير مقبول) وذلك تبعا لما يلي : يحصل كل طالب على درجتين : X للدراسات الإسلامية و Y لعلم الحاسب وكل من الدرجتين محسوبة من ١٠٠ ، ويعطى الطالب تقدير S إذا حقق الشروط الثلاثة التالية ، وما عدا ذلك يحصل على تقدير U:

(أ) درجته في مقرر الدراسات الإسلامية لا تقل عن ٦٠.

(ب) درجته في مقرر علم الحاسب لا تقل عن ٥٠.

(ج) متوسطه العام $\left(\frac{X+Y}{2}\right)$ لا يقل عن ٦٠٪.

ارسم خريطة سير عمليات لبرنامج يقرأ درجتي طالب X , Y ويكتب تقديره العام.

١-١٣ اشترك خمسون طالبا في مسابقة لحفظ القرآن الكريم وتفسيره ، وحصل كل طالب على درجتين كل منهما من ٥٠ : الدرجة الأولى R للحفظ والترتيل ، والثانية T للتفسير ، فتكون درجة الطالب الكلية $S = R + T$ من ١٠٠.

ارسم خريطة سير عمليات

(أ) لقراءة قيمتي T , R لكل طالب.

(ب) وإيجاد عدد الطلاب الذين حصلوا على أكثر من ٨٥٪.

(ج) وحساب المتوسط العام للدرجات (مجموع درجات كل الطلاب / ٥٠).

١-١٤ ترجم الخوارزمية التالية إلى خريطة سير عمليات ، واذكر وظيفة الخوارزمية ، أي ماذا تحسب ؟

١- اجعل $I = 1 , N = 0$

٢- اقرأ قيمة A

٣- إذا كان $A > 0$ اجعل $N = N + 1$

٤- اجعل $I = I + 1$

٥- إذا كان $I > 20$ اذهب إلى الخطوة رقم ٧

٦- اذهب إلى الخطوة رقم ٢

٧- اطبع قيمة N

٨- توقف.

١-١٥ ارسم خريطة سير عمليات لبرنامج يقرأ قيمة عدد صحيح فردي موجب N ثم يحسب

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \cdots + \frac{1}{N}$$

١-١٦ ارسم خريطة سير عمليات لبرنامج يعمل جدولا لحساب الدالة

$$Y = \sqrt{1 + x} + \frac{\cos 2x}{1 + \sqrt{x}}$$

وذلك لعدد N من قيم x متساوية المسافات فيما بينها ، مبتدئا بالقيمة الابتدائية XIN ومنتهيا بالقيمة النهائية XFI ، ويبدأ البرنامج بقراءة قيم XIN , XFI , N.

ملاحظة : المسافة بين أي قيمتين متتاليتين للمتغير x تعطى بالعلاقة.

$$DX = \frac{XFI - XIN}{N - 1}$$

١-١٧ ارسم خريطة سير عمليات برنامج يقرأ قيمة عدد صحيح N ثم يحدد ما إذا كان هذا العدد عددا أوليا (a Prime) أم لا.

ملاحظة : يقال لعدد إنه أَوَّليّ إذا كان لا يقبل القسمة بدون باق إلا على نفسه والعدد ١ فقط (مثل الأعداد ٥ ، ٧ ، ١٣ ، ٢٣).

إرشاد : من الواضح أنه إذا لم يكن N عددا أوليا فإن أحد الأعداد الصحيحة التالية يقسم N :

$$2,3,4,\cdots,\frac{N}{2}$$

١-١٨ اشترك ٦٠٠٠ طالب وطالبة في مسابقة لكتابة بحث بعنوان (جنسية المسلم عقيدته) حول رابطة الأخوة الإسلامية ووسائل تقويتها ، وكيفية محاربة الدعوات الجاهلية (كدعوات العلمانية والقومية والنعرات العنصرية) التي تعمل على إضعاف رابطة العقيدة بين المسلمين وعلى تفريقهم إلى شيع وأحزاب متناحرة. وقد أعدت لكل مشترك بطاقة عليها بعض البيانات منها: رقم المشترك ID ودرجته X من ١٠٠ عن البحث الذي قدمه ، علما بأن أرقام الطلبة تتراوح من ١ إلى ٣٠٠٠ بينما أرقام الطالبات من ٣٠٠١ إلى ٦٠٠٠. وكل من حصل على أكثر من ٨٠٪ أتيحت له الفرصة للسفر في رحلة مجانية لأداء العمرة والسفر إلى بعض البلاد الإسلامية لزيارة الاخوة والأخوات في العقيدة على ألا تسافر الطالبة إلا مع ذي محرم منها.

ارسم خريطة سير عمليات برنامج يقرأ بطاقات المشتركين في المسابقة ، ويحسب العدد الإجمالي لمن يتوقع اشتراكهم في الرحلة.

١-١٩ ارسم خريطة سير عمليات برنامج يعمل جدولا لقيم الدالة

$$y = \frac{x^3+7x-5}{x^3-3x^2-4x+12}$$

المناظرة لقيم x التالية :

$$x = -4 , -3 , \ldots , 7 , 8 , 9$$

مع مراعاة أنه قبل إجراء أي عملية قسمة لحساب y يجب التأكد من قيمة المقام ، فإن كان يساوي صفرا فاطبع الرسالة NOT FINITE بدلا من قيمة y المناظرة.

١-٢٠ اكتب خوارزمية برنامج يقرأ قيمة A ثم يحسب مجموع الخمسين عددا

$$1 , 1 + A , 1 + 2A , 1 + 3A , \ldots , 1 + 49A$$

١-٢١ ارسم خريطة سير عمليات برنامج لحل المعادلتين الخطيتين

$a_1x + b_1y = c_1$
$a_2x + b_2y = c_2$

وذلك بقراءة سجلين الأول منهما يحتوي على a_1, b_1, c_1 **والثاني على** a_2, b_2, c_2 **ثم يحسب** x , y **من العلاقتين :**

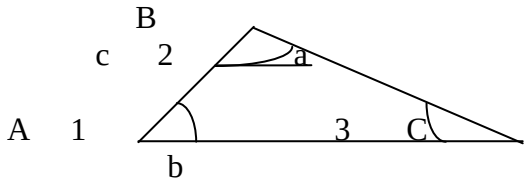
$$x = \frac{b_2c_1 - b_1c_2}{a_1b_2 - a_2b_1} \quad , \quad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

وذلك بشرط أن : $D = a_1b_2 - a_2b_1 \neq 0$

أما إذا كانت $D = 0$ **فإن البرنامج يطبع رسالة تفيد ذلك.**

١-٢٢ ارسم خريطة سير عمليات برنامج يقرأ ثلاثة أعداد a , b , c **ثم يتحقق ما إذا كان من الممكن لهذه الأعداد أن تمثل أطوال أضلاع مثلث أم لا.**

(ملاحظة : شرط تكوين المثلث : كل ضلع أصغر من مجموع الضلعين الآخرين). فإن كانت لا تُكَوِّن مثلثا فيطبع البرنامج رسالة بذلك ، أما إن كانت تكون مثلثا فيحسب



(i)**مساحته** R **حيث** $R = \sqrt{S(S-a)(S-b)(S-c)}$

(S **يمثل نصف المحيط).**

(ii)**محيطه** P $(P = a + b + c)$

(iii)**جيب تمام كل زاوية من زواياه (مثلا جيب تمام الزاوية** A **يساوي :**

$$(CosA = \left(b^2 + c^2 - a^2\right) / \left(2bc\right)).$$

١-٢٣ المطلوب رسم خريطة سير عمليات برنامج يحسب مجموع المتسلسلة

$$S = \sum_{i=1}^{i=30} (3i + 2)^2 = 5^2 + 8^2 + 11^2 + \cdots + 92^2$$

١-٢٤ تُقَدَّر زكاة عروض التجارة ZT بنسبة ٢,٥٪ من القيمة السوقية MV (Market Value) للعروض التجارية (أي القيمة الفعلية في السوق للبضائع المعروضة للبيع ، ولا عبرة بثمن شرائها وتكلفتها ، ولا بالسعر المرغوب بيعها به) وذلك إذا بلغت هذه القيمة السوقية للمواد التجارية نصابا من الذهب (أو الفضة ، ونصاب الذهب = ٨٥ جم ذهبا ، ونصاب الفضة = ٥٩٥ جم فضة) بشرط حولان الحول (والحول يبدأ منذ الشراء بنية التجارة) {ملاحظة : الأثاث والأجهزة المستخدمة لصالح عرض التجارة وبيعها أو خزنها أو نقلها كالسيارات ونحوها لا زكاة فيها).

المطلوب : رسم خريطة سير عمليات برنامج :

أ) يقرأ سعر جرام الذهب PG ، والقيمة السوقية MV للعروض التجارية التي حال عليها الحول.

ب) يحسب نصاب الذهب بالدينار NG.

ج) يحسب زكاة عروض التجارية ZT.

د) يطبع قيمة كل من القيمة السوقية والزكاة.

١-٢٥ يشتمل "جدول محاسبة النفس اليومي" التالي على عدد من الأسئلة حيث إجابة أي منها : نعم أو لا. ارسم خريطة سير عمليات برنامج يقرأ إجابة كل سؤال "A" (حيث الإجابة هي أحد الرقمين : 1 ويعني نعم ، أو : 0 ويعني لا) ، ويحسب عدد الأسئلة "I" التي أجيبت بالإثبات (1) من بين الخمسة عشر سؤالا الأولى (الأسئلة ١ ← ١٥) ، وعدد الأسئلة "J" التي أجيبت بالنفي من بين الأسئلة ١٦ ← ٢٠، ثم يحسب ويطبع المجموع M = I + J .

جدول محاسبة النفس اليومي	
١.	هل أديت الصلوات الخمس في أوقاتها (في المسجد) ؟
٢.	هل شعرت بأنك أحسنت الصلاة ووجدت لذة لذلك ؟
٣.	هل قمت شيئا من ليلتك الماضية (صلاة القيام) ؟
٤.	هل قرأت وردك اليومي من كتاب الله تعالى ؟
٥.	هل أديت النوافل الراتبة ؟
٦.	هل أمرت بمعروف ؟
٧.	هل نهيت عن منكر ؟
٨.	هل قدمت لأحد مساعدة مادية أو أدبية ؟
٩.	هل حاولت أن تقرب أهلك من الاحتكام إلى كتاب الله وسنة رسوله ؟
١٠.	هل طالعت في يومك شيئا عن الإسلام ؟
١١.	هل ذكرت رقابة الله في كل عمل عملته ، وزنته بميزان الآخرة ؟
١٢.	هل قمت بعمل تعتبره في خدمة الإسلام والمسلمين ؟
١٣.	هل زرت أحد أقاربك أو أرحامك أو جيرانك أو إخوانك في الله ؟
١٤.	هل كسبت عنصرا جديدا لدعوتك ؟
١٥.	هل ذكرت الأدعية المأثورة في جميع شئون اليوم والليلة ؟
١٦.	هل وعدت وعدا ثم أخلفته أو تأخرت عنه ؟
١٧.	هل اغتبت أو جادلت أو وقع منك ما يسوء غيرك ؟
١٨.	هل أحسست في قلبك غلا أو حسدا لأحد من المسلمين ؟
١٩.	هل فرط منك ما تعتبره مخالفة شرعية ؟
٢٠.	هل عزمت على فعل خير ثم ترددت في عزمك ؟
٢١.	هل حاولت أن تتكلم العربية الفصحى ؟
٢٢.	هل نمت ليلتك على وضوء ؟

الفصل الثاني

أساسيات لغة البرمجة C++
Fundamentals of the Programming Language C++

لغة البرمجة C++[*] هي إحدى اللغات عالية المستوي (high level language) المستخدمة لكتابة برامج مصدرية (source programs) يمكن إدخالها للحاسب لحل مسائل معينة. ونبدأ بعون الله تعالي فيما يلي بعرض أساسيات البرمجة بهذه اللغة.

مكونات البرنامج بلغة C++

يتكون أي برنامج بلغة C++ من برنامج فرعي (subprogram) واحد أو أكثر ، ويسمي أي برنامج فرعي "دالة" (function) . أي أن أي برنامج بلغة C++ يتكون من داله أو أكثر ، أي من عدة دوال (collection of functions) تكوِّن في مجموعها هذا البرنامج، حيث تؤدى كل دالة منها وظيفة معينة (particular task)، وتتعاون الدوال جميعها مع بعضها البعض لحل المسألة المطلوبة.

وأي برنامج بلغة C++ يجب أن يشتمل على دالة تسمى main. ويبدأ تنفيذ البرنامج دائماً بهذه الدالة main . وأثناء تنفيذ عباراتها قد تستدعي دالة أخرى لأداء وظيفة معينة وبعد الانتهاء من تنفيذ هذه الدالة الأخرى فإنها تعيد التحكم control إلى الدالة الرئيسية main لتستكمل تنفيذ عباراتها.

(*) ظهرت لغة البرمجة C أواخر الستينات وأوائل السبعينات ثم تطورت إلى لغة C++ التي ظهر أول تقرير منها عام ١٩٨٥ حيث أضيفت إلى لغة C موضوعات تجريد البيانات (data abstraction) ، والبرمجة الموجهة نحو الهدف (object-oriented programming) ، وأخيراً ظهرت عام ١٩٩٨ الصيغة (version) المسماة : ISO/ANSI standard C++ [language] والتي نطلق عليها لغة البرمجة C++ القياسية والتي هي موضوع هذا الكتاب، والتي توصلت إليها لجنة مشتركة من منظمة المعايير الدولية ISO (International Standards Organization) ومعهد المعايير القومية الأمريكية ANSI (American National Standards Institute)

مثال ٢-١: البرنامج التالي يتكون من ثلاث دوال: الدالة main**، والدالة** Square **لحساب مربع العدد الصحيح** n **، والدالة** Cube **لحساب مكعب العدد الصحيح** n **. والدالة** main **تستدعي كلا من الدالة** Square **والدالة** Cube **لحساب** 27^2 **و** 27^3 **على الترتيب ، وتطبع النتيجتين.**

```
// ******************************************************
//Sample program
//This program computes the square and cube of 27
// ******************************************************
#include <iostream<

using namespace std;

int Square( int);
int Cube( int );

int main()
}
  cout << "The square of 27 is " << Square(27) << endl;
  cout << "and the cube of 27 is " << Cube(27) << endl;
  return 0;
{

int Square( int n)
}
  return n * n;
{

int Cube( int n )
}
  return n * n * n;
{
```

ملاحظات:

- **في كل من الدوال الثلاث توضع العبارات** (statements) **المطلوب تنفيذها بين القوس الأيسر** { (left brace) **والقوس الأيمن** }**، فهما – على**

الترتيب – يشيران إلى بداية (beginning) ونهاية (end) هذه العبارات. ومجموعة العبارات التي تظهر بين هذين القوسين تعرف باسم : جسم الدالة **(body of the function)** .

- تنفيذ البرنامج يبدأ دائماً بأول عبارة في الدالة **main**. ففي برنامجنا هذا أول عبارة هي :

```
cout << "The square of 27 is " << Square(27) << endl;
```

وهي عبارة طباعة / عبارة إخراج (output statement) تطبع على شاشة عرض الحاسب (computer's display screen) معلومات (information) عبارة عن رسالة وقيمة عددية. أما الرسالة فهي :

```
The square of 27 is
```

وأما القيمة العددية فهي القيمة التي نحصل عليها باستدعاء الدالة Square لتربيع العدد 27 وإرسال القيمة الناتجة للدالة main وهي الدالة المستدعية caller. وهذه القيمة العددية هي : 0279 أي أن المعلومات التي تطبع على الشاشة هي:

```
The square of 27 is 729
```

- تستمر الدالة الرئيسية mainفي تنفيذ عباراتها ، فتنفذ العبارة التالية وهي عبارة طباعة الرسالة :

```
and the cube of 27 is
```

والقيمة العددية التي تحصل عليها باستدعاء الدالة Cube لتكعيب العدد ٢٧. وبالتالي تكون المخرجات الكلية التي نحصل عليه بتنفيذ هذا البرنامج هي :

```
The square of 27 is 729
and the cube of 27 is 19683
```

- كل من الدوال الثلاث Square, Cube, main يطلق عليها داله تعيد القيمة (value returning function)، حيث أن كلا مها تعيد قيمة وحيدة

(single value) للمستدعي (caller). وكلمة **int** عند بداية السطر الأول في كل من هذه الدوال الثلاث تعني أن الدالة تعيد قيمة صحيحة (integer value). فالدالة Square تعيد مربع العدد الصحيح n ،والدالة Cube تعيد مكعب العدد الصحيح n ، والدالة main تعيد العدد الصحيح 0 لنظام التشغيل (operating system) دلالة على أداء هذه الدالة الرئيسية مهمتها بنجاح.

مفردات اللغة

تتكون مفردات اللغة C++ أو أبجدياتها من العناصر أو الرموز (characters) الأساسية التالية:

(أ) الأرقام العربية (Arabic Digits)

وهي الأرقام العشرة : 0,1,2,…,8,9

(ب) الحروف اللاتينية (Letters) الكبيرة والصغيرة من A إلى Z

وهي كل من الحروف الكبيرة:

A, B, C, ……., Y, Z

والحروف الصغيرة:

a, b, c, ……., y, z

(ج) بعض الرموز الخاصة (Special Symbols)

مثل الرموز الخاصة التالية:

```
(   )   {   }   [   ]
+   -   *   /   -   '
=   <   >  <=  >=  !=
,   ;   ==  "   %  ::
```

ويلاحظ أن الفراغ (space) هو أحد الرموز الخاصة.

ومن المفردات السابقة يمكن تركيب كلمات وتعابير وعبارات اللغة تبعاً لقواعد خاصة ، كما سنرى بإذن الله في هذا الفصل والفصول التالية.

الدالة الرئيسية (main function)

الصيغة العامة لهذه الدالة هي:

```
int main ( )
{
عبارات الدالة
}
```

حيث تبدأ الدالة بالكلمة int تتبعها كلمة main يتبعهما القوسان ( ) parentheses ، وهذا السطر الأول من الدالة يطلق عليه عنوان الدالة أو مطلعها (heading)، ثم يوضع جسم (body) الدالة – أي عباراتها – بين القوسين { } ، ويجوز أن يكون خالياً أو مكوناً من عبارة واحدة أو أكثر. ولكن من الناحية العملية فإنه يجب أن يحتوي على عبارة Return لأن الكلمة int في مطلع البرنامج تنص على أن الدالة main تعيد قيمة صحيحة ، وبالتالي فإن أقصر برنامج C++ هو :

```
int main ( )
{
 return 0;
}
```

ولكن بالطبع لا يعمل شيئاً مفيداً حين نقوم بتنفيذه.

الأسماء التعريفية Identifiers

تستخدم الأسماء التعريفية لتسمية الأشياء كالبرامج الفرعية (الدوال) والأماكن في ذاكرة الحاسوب .. وتسمية الثوابت والمتغيرات .. وتسمية البيانات. ويتكون الاسم من حروف (A→Z, a→z) وأرقام (0→9) والشرطة السفلية (_) (underscore) على أن يبدأ بحرف أو شرطة سفلية ، أي لا يجوز أن يبدأ برقم.

ويفضل عموماً أن يبدأ الاسم التعريفي بحرف لأن الأسماء التعريفية التي تبدأ بالشرط السفلية لها معاني خاصة في بعض نظم C++.

ويلاحظ أن لغة C++ تفرِّق بين الحرف الكبير والحرف الصغير في الأسماء التعريفية، أي أن اللغة حسَّاسة لحالة الحرف (case-sensitive language) ، هل هو حرف كبير (uppercase letter) أم حرف صغير (lowercase letter) فمثلاً الأسماء التعريفية التالية:

ZAKATOFMONEY zakatofmoney ZaKaToFmOnEy ZakatOfMoney

تعتبر أربعة أسماء مختلفة (distinct names) ، ولا يمكن استخدام أحدها مكان الآخر . ويلاحظ أن الاسم ZakatOfMoney هو أيسرها في القراءة ، ولذلك فهو أفضلها في الاستخدام كاسم تعريفي للدلالة على زكاة المال.

مثال ٢-٢:

أي الأسماء التالية تعتبر أسماء تعريفية صحيحة (valid) أي مقبولة ، وأيها تعتبر غير صحيحة (invalid) أي غير مقبولة ؟ ولماذا ؟

(i)	Zakat	(ii)	x2	(iii) number30
(iv)	40Hours	(v)	Al_Nisab	(vi) cost_in_$
(vii)	int	(viii)	sum-of-squares	(ix) Total Sum
(x)	αβ3	(xi)	box_25A	(xii) GetData

الحل:

أولاً : كل من الأسماء التعريفية (i), (ii), (iii), (v), (xi), (xii) يعتبر اسماً تعريفياً صحيحاً.

ثانياً: كل من الأسماء التعريفية الأخرى يعتبر اسماً تعريفياً غير صحيح للأسباب التالية:

(iv) الاسم التعريفي لا يبدأ برقم.

(vi) لا يسمح باستخدام الرمز الخاص "$"

(vii) الكلمة int معرفة سابقاً (predefined) في لغة C++.

(viii) لا يسمح باستخدام الرمز الخاص "–" [هذا الرمز (hyphen) يشير إلى عملية الطرح في الرياضيات في لغة C++].

(ix) لا يسمح باستخدام الفراغ (وهو رمز خاص) في الأسماء التعريفية.

(x) لا يسمح باستخدام الحروف اللاتينية $\alpha, \beta, \gamma, \delta$ ..... في الأسماء التعريفية.

الكلمات المحجوزة Reserved Words

الكلمات المحجوزة هي كلمات لها استخدامات خاصة محددة (specific uses) في لغة C++ ، مثل كلمة int التي تستخدم للدلالة على أسماء المتغيرات والدوال صحيحة القيمة (integer valued) . ويجب عدم استخدام أي من هذه الكلمات المحجوزة كاسم تعريفي يعرِّفه المبرمج (programmer defined identifier). ومن أمثلة هذه الكلمات المحجوزة:

char	const	if	else	return
and	or	not	true	false

ويوجد بنهاية الكتاب بالملحق (أ) قائمة بجميع الكلمات المحجوزة في لغة C++.

أنواع البيانات Data Types

يجري البرنامج في الحاسوب عملياته على البيانات ، سوء المخزونة (stored) داخلياً (internally) في الذاكرة (memory)، أو خارجياً (externally) على القرص (disk) أو الشريط (tape)، أو المدخلة (input) من أداة (device)كلوحة المفاتيح (keyboard) أو الماسح (scanner) أو نبيطة الاستشعار / المجس الكهربائي (electrical sensor)، ليقوم بعد ذلك بالحصول على نتائجه ومخرجاته (output). وفي لغة C++ يشترط في أي بيانات أن تكون من نوع بيانات محدد (specific data type) . وهذا النوع يحدد كيفية تمثيل البيانات في

الحاسوب وأنواع العمليات التي يمكن أن يجريها الحاسوب عليها. وبعض أنواع البيانات هذه تُستخدم بكثرة ، ولذلك تعرِّفها لنا اللغة ، وتسمى أنواعاً قياسية (standard) ومن هذه الأنواع القياسية (أو المبنية داخلياً built-in) :

int : للأعداد الصحيحة (integer numbers)

float : للأعداد الحقيقية (real numbers) التي تشتمل على العلامات العشرية (decimal points).

char : للبيانات المكونة من رمز (character) واحد.

وإضافة للأنواع القياسية فإن اللغة تسمح للمبرمج بتعريف أنواعه الخاصة للبيانات والتي تسمي أنواع البيانات المعَّرفة بالمستخدِم / بالمبرمِج (programmer / user defined data types) وهذه الأنواع سنتعرض لها فيما بعد .

نوع البيانات char

يستخدم هذا النوع لوصف البيانات المكونة من رمز واحد أبجدي أو عددي (alphanumeric) أي حرف (letter) أو رقم (digit) أو رمز خاص (special symbol) ، مثل :

'A'	'a'	'R'	'5'	'0'	'8'
	'+'				
'_'	'?'	'*'	' '	'$'	'/'
'+'	'.'				

وتستخدم كل ماكينة (machine) مجموعة خاصة من الرموز هي الرموز الأبجدية والعددية التي يمكن أن تمثلها . والملحق (ج) بنهاية الكتاب يبين أمثلة لمجموعات الرموز هذه. ويلاحظ أن كل رمز يُحاط (enclosed) أو يُحصر بين حاصرتين مفردتين (single quotes / apostrophes) يحتاجهما البرنامج المترجم (compiler) للتفرقة مثلاً بين الرمز '5' والقيمة الصحيحة 5 ، حيث أن طريقتي تخزينهما داخل الماكينة مختلفتان ، وكذلك العمليات التي تجرى عليهما تختلف، فمثلاً لا نجمع (add) الرمز '5' مع الرمز '8' كما أننا لا نجمع الرمز 'A' مع الرمز 'R'، ولكن يمكننا أن نجمع القيمة الصحيحة 5 مع القيمة الصحيحة 8 ، أو أن نقارن

(compare) بين قيمتين رمزيتين (2 character values) أيهما أصغر من الأخرى، أي أيهما تسبق الأخرى ، في السلسلة المقارنة(collating sequence) لمجموعة الرموز [انظر الملحق (ج)] – أي الترتيب المعرف سابقاً لجميع الرموز (predefined ordering of all the characters) – وهذه السلسلة تختلف عموماً من مجموعة رموز لمجموعة أخرى، ولكن في أي من هذه السلاسل تكون 'A' أصغر من 'B' ، و 'B' أصغر من 'C' ، وهكذا ، وكذلك فإن '1' أصغر من '2' ، و '2' أصغر من '3' ، وهكذا ، وبالتالي '5' أصغر من '8' .

نوع البيانات string

يستخدم النوع char للبيانات المكونة من رمز واحد فقط ، بينما يستخدم النوع string للبيانات المكونة من سلسلة / متتابعة من الرموز (sequence of characters) ككلمة أو اسم شخص أو جملة ، وتكون هذه السلسلة محصورة بين حاصرتين مزدوجتين (double quotes) ، ومن أمثلة سلاسل الرموز (strings of characters)في لغة C++:

```
"Zakat of Money  =  "          "  C++  "
"Khalid Ibn Al-Waleed  "       "  .  "
"The solution is :  "          "  Solution"
" Be in the world as though you were a stranger or a wayfarer"
```

ويلاحظ أن أي سلسلة رموز (string) يجب أن تطبع (typed) كاملة (entirely) على سطر واحد ، فمثلاً السلسلة

```
" The two roots of the equation
  are real and different"
```

تعد غير صحيحة (invalid) لأنها قسمت على سطرين .

ويلاحظ كذلك أن الحاصرات لا تعد جزءاً من سلسلة الرموز، فمثلاً "solution" هي سلسلة الرموز (character string) المكونة من الحروف s,o,l,u,t,i,o,n بهذا الترتيب بعينه، بينما solution (بدون الحاصرات) هي اسم تعريفي (identifier)،

قد يكون اسم مكان في الذاكرة تخزن فيه قيمة حل معادلة مثلاً. وكذلك فإن الرموز (symbols) "314" تمثل سلسلة رموز مكونة من 3,1,4 بهذا الترتيب بعينه، بينما 314 (بدون الحاصرات) فهي كمية صحيحة (integer quantity) يمكن أن تستخدم في الحسابات (calculations) ، وقد تشير مثلاً إلى عدد الصحابة الذين اشتركوا في غزوة بدر الكبرى.

والسلسلة (string) التي لا تحتوى على أي رموز (characters) يطلق عليها السلسلة الخالية/ الفارغة (empty/null string) وتكتب باستخدام حاصرتين مزدوجتين ليس بينهما أي شئ ، ولا حتى أي فراغ، هكذا " " والسلسلة الخالية (null string) ليست مكافئة لسلسلة فراغات (a string of spaces) ، ولكنها سلسلة خاصة لا تحتوى على أي رمز [لاحظ أن الفراغ يعد رمزاً].

ويلاحظ أن النوع string ليس من الأنواع المبنية (built-in type) في لغة C++ ولكنه نوع معرف بالمبرمج (programmer defined type) تعرضه مكتبة C++ القياسية (supplied by the C++ standard library) وهي عبارة عن مجموعة كبيرة من دوال مكتوبة سابقاً (prewritten function) وأنواع بيانات (data types) يمكن أن يستخدمها أي مبرمج بلغة C++.

والعمليات التي يمكن أن تجرى على البيانات من النوع سلاسل الرموز تشمل المقارنة (comparison) بين قيم السلاسل (values of strings) ، والبحث (searching) في سلسلة رموز عن رمز معين (particular character)، ووَصْل (joining)سلسلة بسلسلة أخرى. وسنتناول بعض هذه العمليات في هذا الفصل ، وبعضها الآخر في فصول لاحقة بإذن الله.

الإعلانات / التعريفات (Declarations / Definitions)

تستخدم الأسماء التعريفية لتسمية كل من الثوابت (constants) والمتغيرات (variables) . أي أن الاسم التعريفي يمكن أن يكون اسم موضع في الذاكرة لا يسمح لمحتوياته (contents) أن تتغير أو أن يكون اسم موضع يمكن أن تتغير محتوياته. ويُقصد بالإعلان (declaration) عبارة (statement) تلحق (associates) اسماً تعريفياً (identifier) بوصف (description) عنصر (element) في برنامج بلغة C++، حيث نُسمِّي (name) في هذا الإعلان اسماً تعريفياً ونبيِّن نوعه ، أي ماذا يمثل (represents) . فمثلاً الإعلان

```
int     empNum;
```

في برنامج أجور الموظفين Payroll program في الفصل التمهيدي يعلن أن empNum هو اسم متغير محتوياته من النوع int . وحين نعلن عن متغير فإن البرنامج المترجم (compiler) يحجز موضعا (location) في الذاكرة لترتبط بهذا الاسم التعريفي. ولا يُطلب منا أن نعرف عنوان (address) هذا الموضع لأن الحاسوب هو الذي يتتبعه تلقائياً (automatically) للاستخدام.

ويطلق على مجموعة الثوابت والمتغيرات عناصر البيانات (data objects)، وهذه العناصر مع التعليمات الفعلية (actual instructions) في البرنامج يتم تخزينها في مواضع عديدة في الذاكرة. ورأينا كيف أن مجموعة من التعليمات- دالة-يمكن أن تعطى اسماً معيناً . ويمكن أيضاً إعطاء الأسماء لأنواع البيانات المعرفة بالمبرمج. وعموماً في لغة C++ يجب الإعلان عن أي اسم تعريفي قبل استخدامه. وهذا يسمح للبرنامج المترجم أن يتحقق من أن استخدام الاسم التعريفي يتفق (consistent) ويتواءم مع تعريفه والإعلان عنه. فمثلاً إذا عرفنا اسماً تعريفياً على أنه ثابت ، وحاولنا في البرنامج بعد عدة تعليمات أن نغير قيمته، فإن البرنامج المترجم يكتشف عدم التوافق (inconsistency) هذا ويصدر رسالة لبيان هذا الخطأ (error message). وكل نوع من عناصر البيانات (الثوابت والمتغيرات)

والدوال وأنواع البيانات له صيغة (form) خاصة لعبارة الإعلان (declaration statement) . وفيما يلي نتناول صيغ الإعلان عن المتغيرات والثوابت، وسنتناول بإذن الله صيغ عبارات الإعلان الأخرى في الفصول التالية.

المتغيرات (Variables)

أثناء تنفيذ برنامج ما قد يتم تخزين قيم مختلفة في موضع الذاكرة نفسه (same memory location) في أوقات مختلفة. هذا النوع من مواضع الذاكرة يسمى متغيراً (variable) ، ومحتوياته هي قيمة المتغير (variable value) والاسم الرمزي (symbolic name) الذي نُلحقه (associate) أي نجعله يرتبط بموضع ذاكرة هو اسم المتغير (variable name) ، أو الاسم التعريفي للمتغير (variable identifier) . ومن الناحية العملية فكثيراً ما نشير إلى اسم المتغير (variable name)اختصاراً بالمتغير (variable).

والإعلان عن متغير يعني تحديد اسمه ونوع بياناته ، وهذا الإعلان يوجه البرنامج المترجم إلى أن يلحق اسماً بموضع ذاكرة محتوياته من هذا النوع المحدد (مثلاً char أو string) . فمثلاً العبارة

```
char  ch;
```

تعلن أن ch متغير من النوع char. وأي متغير لا يمكن أن يحتوى على أي قيم بيانات (data value) إلا من النوع الذي أعلن عنه فقط . فمثلاً المتغير السابق ch لا يمكن أن يحتوى إلا على قيمة رمزية (char value) فقط.

وإذا كان هناك أكثر من متغير واحد من النوع نفسه ، فيمكن أن نعلن عنهم جميعاً في عبارة واحدة أو في أكثر من عبارة كما يلي:

```
float payRate, hours, wages, total;
```

حيث أعلنا عن المتغيرات الأربعة payRate, hours, wages, total; في عبارة واحدة لأن الأربعة متغيرات من النوع نفسه(float) ، ويجوز أن نعلن عنها هكذا :

```
float payRate;
float hours;
float wages;
float total;
```

حيث أعلن عن كل متغير على حدة. وقد تكون هذه الطريقة أفضل حيث أنها تسمح لنا بإضافة تعليقات (comments) يمين كل إعلان ، كما رأينا سابقاً في برنامج أجور الموظفين Payroll في الفصل التمهيدي ، هكذا :

```
float   payRate;          // Employee's pay rate
float   hours             // Hours worked
        ⋮                       ⋮
```

الثوابت غير العددية (Non-numeric Constants)

الثوابت الرمزية وثوابت سلاسل الرموز
Character Constants and String Constants

أي رمز مفرد (single character) محصور (enclosed) بين حاصرتين مفردتين (single quotes) يعد ثابتاً، وكذلك أي سلسلة رموز محصورة بين حاصرتين مزدوجتين (double quotes) تعد ثابتاً ، مثل :

```
' A '   ' * '       "Please enter total annual savings:"
```

والثابت في لغة C++ – كما هو في الرياضيات – هو الشيء الذي لا تتغير قيمته (value). وحينما نستخدم القيمة الفعلية (actual value) لثابت في برنامج فإنه يقال إننا نستخدم قيمة حرفية (a literal value / literal). فالقيمة الحرفية هي أي قيمة ثابتة مكتوبة في برنامج.

ويمكننا بدلاً من أن نستخدم القيمة الحرفية مباشرة في إحدى العبارات أو التعليمات أن نعطيها اسماً معيناً في عبارة إعلان ، ثم نستخدم هذا الاسم في تلك العبارة التي استخدمت فيها القيمة الحرفية. فمثلاً لطباعة العنوان

Zakat of Money ، يمكننا إما أن نكتب أمر طباعة يستخدم سلسلة الرموز الحرفية "Zakat of Money" أو أن نعطي هذه السلسلة اسماً (name) مثل Title أي أن نعلن عن ثابت مسمَّى (named constant) / ثابت رمزي (symbolic constant) اسمه Title ويساوي سلسلة الرموز الحرفية هذه نفسها، ثم نستخدم اسم هذا الثابت في أمر الطباعة. أي أننا في أمر الطباعة أما أن نستخدم السلسلة " Zakat of Money" أو أن نستخدم الاسم Title . وبالمثل إذا احتجنا قيمة $\pi = 3.1415962$ عدة مرات في البرنامج مثلاً لحساب مساحة دائرة و محيط دائرة وحجم كرة والمساحة الجانبية لاسطوانة ..... الخ ، فبدلاً من كتابة هذه القيمة الحرفية 3.1415962 كلما احتجنا إليها يمكننا أن نعطيها اسماً مثل PI وكلما احتجنا إليها استخدمنا هذا الاسم PI . والصيغة العامة لتعريف /للإعلان عن هذه الثوابت هي:

const DataType Identifier = Literal Value ;

ومن أمثلة ذلك :

```
const   string   TITLE = "Zakat of Money";
const   float    PI = 3.1415962;
const   char     BLANK = '  ' ;
const   char     FULL_ STOP = ' . ' ;
const   string   STARS = " ***** " ;
const   string   MESSAGE = "Error condition" ;
const   float    ZAKT_RATE = 0.025 ;
```

ويمكننا إضافة تعليقات لإعلانات الثوابت والمتغيرات للإيضاح، فمثلاً في الإعلان الأخير السابق يمكننا إضافة التعليق التالي:

```
const float ZAKAT_RATE = 0.025 ; // Percentage of Zakat of Money
```

ملاحظة : تعد كلمة const من الكلمات المحجوزة [انظر قائمة الكلمات المحجوزة (reserved words) بنهاية الكتاب] . وكقاعدة عامة فإن جميع الكلمات المحجوزة في لغة C++ يجب أن تكتب كلها بحروف صغيرة ، وكذلك معظم

الأسماء التعريفية المعلن عنها في المكتبة القياسية (standard library) مثل كلمة string ، وكذلك اسم الدالة الرئيسية main يكتب كله بحروف صغيرة. وأما الأسماء التعريفية التي تمثل الثوابت المسماة والمتغيرات والدوال وأنواع البيانات فطريقة كتابتها اختيارية (من حيث الحروف الكبيرة والصغيرة).

عبارة الإسناد (Assignment Statement)

يمكننا أن نسند /نعطي (assign/set/give) قيمة (value) لمتغير (variable) ما، أو أن نغير (change) قيمته عن طريق ما يسمى بعبارة إسناد مثل

```
lastName = " Al_Khwarazmi ";
```

حيث نسند القيمة / السلسلة " Al_Khwarazmi " للمتغير LastName، أي نخزن هذه القيمة التي هي عبارة عن سلسلة رموز في موضع الذاكرة المحجوز للمتغير/ المرتبط بالمتغير (associated with the variable) المسمى LastName .

والصيغة العامة لعبارة الإسناد هي:

```
Variable = Expression;
```

وتعني تخزين قيمة التعبير Expression في المتغير Variable ، وإذا كانت للمتغير أي قيمة سابقة فإنها تُمحَي / تُدمَّر (erased / destroyed) وتحل محلها (replaced by) قيمة التعبير، ونقصد بالتعبير:

التعبير Expression

هو ترتيبة (arrangement) من : الأسماء التعريفية والقيم الحرفية (literals) والمؤثرات (operators) ، والتي يمكن تقييمها (evaluated)لحساب (computing) قيمة من نوع (type) معين.

ويجب أن يظهر متغير واحد فقط في الطرف الأيسر من معادلة الإسناد، فلا يجوز مثلاً كتابة عبارة إسناد مثل : A + B = C + 4;

ولكن يمكن كتابة عبارة مثل : A = C + 4 –B:

وتعني حساب قيمة التعبير C+4–B وتخزين هذه القيمة في المتغير A، ويظل المتغير A محتفظاً بهذه القيمة إلى أن تأتي عبارة أخرى تخزن فيه قيمة جديدة.

مثال ٢–٣ : نفرض أن لدينا الإعلانات التالية

```
string  firstName;
string  middleName;
string  lastName;
string  title;
char    middleInitial;
char    letter;
```

أي عبارات الإسناد التالية صحيحة (valid) وأيها خاطئة (invalid) مع بيان السبب.

```
(i)     firstName       =       "Mohammad" ;
(ii)    middleName      =       firstName ;
(iii)   middleName      =       " " ;
(iv)    lastName        =       "Al_Fateh" ;
(v)     title           =       "Soltan" ;
(vi)    middleInitial   =       ' ' ;
(vii)   letter          =       middleInitial ;
(viii)  middleInitial   =       " I." ;
(ix)    letter          =       firstName ;
(x)     middleName      =       Ibn_Morad ;
(xi)    " Al_Fateh "    =       lastName;
(xii)   lastName        =       ;
```

الحل : العبارات من(i) إلى (vii) صحيحة جميعها.

العبارات من (viii) إلى (xii)خاطئة جميعها للأسباب التالية:

(viii) المتغير middle Initial من النوع char بينما " I. " سلسلة رموز

(ix) المتغير letter من النوع char بينما المتغير firstName من النوع string.

(x) Ibn_Morad اسم تعريفي غير معرَّف (غير معلن عنه) (an undeclared identifier)

(xi) لا يجوز أن يظهر يسار العلامة = في عبارة الإسناد إلا متغير فقط، بينما الطرف الأيسر هنا ثابت عبارة عن سلسلة رموز .

(xii) لا يوجد تعبير يمين العلامة = في عبارة الإسناد.

تعابير سلاسل الرموز (String Expressions)

تعاقب سلاسل الرموز (Concatenation of Strings)

من العمليات التي يمكن أن نجريها على سلاسل الرموز (string operations) عملية تعاقب السلاسل ، والتي نستخدم لها المؤثر (operator) + ، حيث يؤثر على سلسلتي رموز فتكون النتيجة وصلهما (joining) معاً ليصبحا سلسلة واحدة تحتوي على رموز السلسلتين:السلسلة التي على يسار المؤثر + تعقبها على يمينها السلسلة التي على يمين المؤثر +، أي أن ترتيب السلسلتين في التعبير يؤثر على السلسلة الناتجة (resulting string).

مثال ٢-٤:

نفرض أن لدينا العبارات التالية

```
string sentence ;
string phrase1 ;
string phrase2 ;

phrase1 = " fear and obey Allah" ;
phrase2 = " wherever you are" ;
```

ما هي السلسلة الناتجة التي تخزن في sentence بعد تنفيذ كل من العبارتين التاليتين ؟

```
(i)     sentence = phrase1 + phrase2 ;
(ii)    sentence = phrase2 + phrase1 ;
```

الحل :

(i) هذه العبارة تسترجع (retrieves) قيمتي phrase1 , phrase2 من الذاكرة وتكوِّن (forms) سلسلة جديدة مؤقتة (new temporary string)

تحتوي على رموز السلسلة phrase1 تعقبها على يمينها رموز السلسلة phrase 2 ، أي تكوِّن السلسلة

" fear and obey Allah wherever you are"

وهذه السلسلة المؤقتة – والتي هي من النوع string – تُستند إلى (assigned to) أي تُخزن في (stored into) – المتغير sentence.

(ii) في هذه الحالة السلسلة الناتجة التي تخزن في sentence هي :

" wherever you are fear and obey Allah"

* * *

ويلاحظ أنه يمكن تطبيق عملية تسلسل/تعاقب سلاسل الرموز (concatenation of strings) على ثوابت السلاسل المسمَّاة (named string constants) ، والسلاسل الحرفية (literal strings) ، والبيانات الرمزية (char data) بالإضافة إلى متغيرات السلاسل (string variables). والشرط الوحيد الذي علينا أن نتقيد به هو أنه يجب أن يكون أحد المعامَلين (الكميتين المشغَّلتين) (operands) الموجودين يمين ويسار المعامِل / المؤثِّر + (operator)] متغير سلاسل (string variable) أو ثابتاً مسمَّى (named constant) . فمثلاً لا يمكننا استخدام تعبير مثل "Good " + " Morning" أو 'C' + 'D'

مثال ٢–٥ :

نفرض أن لدينا الإعلانات عن الثوابت التالية:

```
const  string  WORD1    = " world" ;
const  string  WORD2    = " stranger" ;
const  string  WORD3    = " wayfarer" ;
```

ونفرض أن sentence متغير سلسلة رموز.

ما هي القيمة (ما هي سلسلة الرموز) التي تخزن في المتغير sentence بعد تنفيذ عبارة الإسناد التالية؟

```
sentence = "Be in the" + WORD1 + " as though you were a" +
Word2 + " or " + 'a' + WORD3
```

الحل : بعد تنفيذ عبارة الإسناد المذكورة يتم تخزين سلسلة الرموز التالية في المتغير sentence:

Be in the world as though you were a stranger or a wayfarer

هذا المثال يوضح أنه يمكننا في أي تعبير تعاقبي (concatenation expression) أن نجمع (combine) أسماء تعريفية (identifiers) مع بيانات رمزية (char data) مع سلاسل حرفية (literal strings) . وبالطبع يمكننا أن نسند السلسلة كاملة للمتغير sentence مباشرة هكذا:

sentence ="Be in the world as though you were a stranger or a wayfarer"

ولكن أحياناً نحتاج لإضافة بعض الرموز (some characters) لقيمة سلسلة رموز موجودة فعلاً . فمثلاً نفرض أن sentence تحتوى فعلاً على:

"Be in the world as though you were a stranger"

ونفرض أننا نود إضافة الرموز "or a wayfarer" أي نود إكمال الجملة (/عبارة الحديث)، فيمكننا كتابة عبارة مثل:

sentence = sentence + "or a wayfarer" ;

فهذه العبارة تسترجع قيمة sentence من الذاكرة وتعقبها بالسلسلة "or a wayfarer" لنكوِّن سلسلة جديدة (new string) ، ثم تخزِّن (stores) هذه السلسلة الجديدة في sentence لتحل محل القيمة القديمة للمتغير sentence والتي تُمحى.

يلاحظ أن عملية التعاقب لا يتم إجراؤها إلا مع قيم سلاسل الرموز فقط، فلا يمكن مثلاً إجراء عملية تعاقب لسلاسل (strings) مع قيم من أنواع البيانات العددية (values of numeric data types) مثل int, float رغم أننا نستخدم إشارة الجمع الحسابية + لعملية التعاقب.

عبارة الإخراج / الكتابة (Output / Write Statement)

لطباعة قيم المتغيرات والتعابير نستخدم متغيراً خاصاً (special variable) اسمه cout (ينطق "see-out")مع رمز / معامل / مؤثر الإدخال >> (insertion operator). فمثلاً العبارة

```
cout << " In the name of Allah";
```

تطبع / تعرض (displays) الرموز (characters)

In the name of Allah

على وسيلة الإخراج القياسية (standard output device) وهي عادة شاشة الفيديو للعرض (video display screen) . والمتغير cout هو متغير معرف سابقاً (predefined) في نظم C++ للإشارة إلى (to denote) سيل / تدفق / مجرى من المخرجات (output stream) ، ويُقصد بسيل من المخرجات : متتابعة لانهائية من الرموز (endless sequence of characters) متجهة إلى وسيلة الإخراج . وفي حالة cout فإن سيل المخرجات يتجه إلى وسيلة الإخراج القياسية.

ومعامِل الإدخال << (وعادة يُنطبق "put to") يكون له معامَلان : المعامَل الأيسر (left-hand operand) عبارة عن تعبير دافق (stream expression) [في أبسط أحواله مجرد متغير دافق (stream variable) مثل cout] ، والمعامَل الأيمن (right-hand operand) عبارة عن تعبير قد يكون بسيطاً مثل سلسة حرفية (literal string)، هكذا :

```
cout   <<     "Tradition:" ;
cout   <<     sentence + "(Al-Bukhari)" ;
```

ويقوم معامل الإدخال << بتحويل (converting) معامَله الأيمن إلى متتابعة من الرموز (sequence of characters) ولإدخالها في (inserting them into) [أو بتعبير أدق إلحاقها بـ (appending them to)] سيل المخرجات (output stream). ولاحظ أن المعامل << يشير إلى الاتجاه الذي تتجه فيه البيانات من التعبير المكتوب يمينه إلى سيل المخرجات على يساره.

ويمكننا استخدام المعامل << عدة مرات في عبارة إخراج واحدة ، وكل ظهور له يعني إلحاق مفردة البيانات التالية (next data item) إلى سيل المخرجات. فمثلاً يمكنا بدلاً من كتابة عبارتي الإخراج السابقتين كتابة عبارة إخراج واحدة مكافئة هكذا:

```
cout   <<     " Tradition:"  << sentence + "(Al-Bukhari)" ;
```

فإذا فرضنا أن sentence تحتوي على(contains):

"Deeds are but by intention"

فإن الطريقتين السابقتين لكتابة المخرجات (أي استخدام عبارتي إخراج أو عبارة إخراج واحدة مكافئة) تؤديان إلى ظهور المخرجات التالية نفسها:

Tradition : Deeds are but by intention (Al-Bukhari)

مثال ٢-٦ : نفرض أن المتغير الرمزي ch (char variable) يحتوي على القيمة '2' ، وأن متغير سلاسل الرموز firstName (string variable) يحتوى على "Somayya" وأن متغير سلاسل الرموز lastName يحتوى على "Khayyat" .

ما هي مخرجات (output) كل من عبارات الإخراج التالية؟

[استخدم الرمز □ للدلالة على فراغ (blank)]

(i) cout << ch ;
(ii) cout << " ch = " << ch ;
(iii) cout << firstName + " " + lastName
(iv) cout << firstName << lastName;
(v) cout << firstName << ' ' << lastName ;
(vi) cout << " ERROR MESSAGE" ;
(vii) cout << " Error = " << ch ;

الحل:

i) 2
ii) ch□ = □2
iii) Somayya□Khayyat
iv) SomayyaKhayyat
v) Somayya□Khayyat
vi) ERROR□MESSAGE
vii) Error=2

ملاحظة : عبارة الإخراج / الطباعة تطبع السلاسل الحرفية كما تظهر بالضبط ، أي بالطريقة نفسها التي تكتب بها هذه السلاسل ولطباعة سلسلة حرفية- وليس ثابتاً مسمّى أو متغيراً - يجب أن نحصر السلسلة بين حاصرتين مزدوجتين (double quotes)، فإذا أردنا أن نطبع سلسلة تحتوي هي نفسها على حاصرة مزدوجة (double quotes) فيجب أن نطبع رمز الخط المائل الخلفي \ (backslash character) تليه يمينه مباشرة حاصرة مزدوجة (ليس هناك فراغ بين الخط المائل

الخلفي والحاصرة المزدوجة) وذلك في سلسلة الرموز . فمثلاً لطباعة الرموز (characters)

The "Zakat" expenditures

نكتب عبارة إخراج (output statement) كالتالية:

```
cout    <<  " The  \" Zakat\" expenditures " ;
                        *       *       *
```

عادة تؤدى عبارات الإخراج المتتالية إلى استمرار طباعة المخرجات على السطر نفسه في شاشة العرض (display screen). فمثلاً العبارتان

```
cout << " There is no god but Allah " ;
cout << " Mohammad is the Messenger of Allah " ;
```

تؤديان إلى طباعة ما يلي على الشاشة ، حيث تظهر كل المخرجات على السطر نفسه:

There is no god but Allah Mohammad is the Messenger of Allah

فإذا أردنا ظهور السلسلتين على سطرين مستقلين ، فيمكننا كتابة العبارتين التاليتين:

```
cout << " There is no god but Allah " << endl ;
cout << " Mohammad is the Messenger of Allah " << endl ;
```

وتظهر مخرجات هاتين العبارتين هكذا:

There is no god but Allah
Mohammad is the Messenger of Allah

والاسم التعريفي endl الذي استخدمناه في عبارتي الطباعة والذي يعني (end line) – إشارة إلى نهاية سطر – يطلق عليه معالج (manipulator)، وهو يجعلنا ننتقل إلى بداية السطر التالي بعد الانتهاء من سطر المخرجات الحالي.

التعليقات (Comments)

يمكن إضافة تعليقات مختلفة خلال البرنامج لتوضيح بعض عباراته ولسهولة متابعة وفهم محتوياته وخوارزمياته ، ويمكن إضافة هذه التعليقات في أي موضع في البرنامج ما عدا في وسط اسم تعريفي (identifier) أو كلمة محجوزة (reserved word)أو ثابت حرفي (literal constant)

ويمكن كتابة التعليقات بإحدى صيغتين :

الأولى: متتابعة من الرموز (sequence of characters) محصورة داخل زوجي الرموز الخاصة /* */ ، مثل :

string id ; / * identification number of the student * /

حيث يهمل البرنامج المترجم (compiler)أي شئ محصور داخل هذين الزوجين.

الثانية: (وهي الأكثر استخداماً) نبدأ بكتابة الخطين المائلين// ثم يمينهما نكتب التعليق المطلوب ، مثل :

string id ; // identification number of the student

وأيضاً فإن البرنامج المترجم يهمل أي شئ بعد الخطين المائلين. وعموماً تضاف التعليقات عند بداية البرنامج لبيان وظيفته ، وعند الإعلان عن ثوابت البرنامج ومتغيراته ، وعند بدايات الأجزاء الرئيسية والهامة في البرامج الطويلة.

تركيب البرنامج Program Construction

و الآن بعد أن استعرضنا بعض العناصر الرئيسية في برنامج بلغة C++ : الأسماء التعريفية ، والإعلانات ، والمتغيرات ،والثوابت ، والتعابير، والعبارات ، والتعليقات ، نرى كيفية تجميع هذه العناصر في برنامج.

كما رأينا سابقاً يتكون أي برنامج من عدد من الدوال ، ويجب أن نسمِّي إحداها main . كذلك يمكن أن يشتمل البرنامج على إعلانات تقع خارج أي دالة. وتتكون الدالة من إعلان عنها – أي عنوان الدالة (function heading) – وجسم الدالة (function body) ، وهذا يجب أن يكون محصوراً بين القوسين { . } وفيما يلي مثال لبرنامج يشتمل على دالة واحدة فقط

مثال ٢ – ٧ : البرنامج التالي يطبع اسم شخص بصيغتين مختلفتين

```
//*************************************************************
 //PrintName program
 //This program prints a name in two different formats
//*************************************************************
#include <iostream<
#include <string<

using namespace std;

const string FIRST = "Mohammad";    // Person's first name
const string LAST = "AlFateh";      // Person's last name
const char   MIDDLE = 'M';        // Person's middle initial


int main ( )
}
  string firstLast;    // Name in first-last format
  string lastFirst;    // Name in last-first format

  firstLast = FIRST + " " + LAST;
  cout << "Name in first-last format is " << firstLast << endl;

  lastFirst = LAST + ", " + FIRST  + ", ";
  cout << "Name in last-first-initial format is ";
  cout << lastFirst << MIDDLE << '.' << endl;

  return 0;
{
```

ملاحظات:

- **يبدأ البرنامج بتعليق يبين وظيفته .**
- **السطران**

```
#include <iostream>
#include <string>
```

يوجهـان تعليمـات لنظـام (system) C++ **ليُـدخل** (insert) **في برنامجنـا محتويات** (contents) **الملفين:** iostream, string **، حيث يحتوي الملف المُسَمَّى**

iostream على معلومات تحتاجها C++ لإخراج قيم (to output values) لسيل (stream) مثل cout . بينما يحتوى الملف الأخر المسمَّى string على معلومات عن نوع البيانات المعرف بالمبرمج

(programmer-defined data type) string

وسنناقش فيما بعد بإذن الله الغرض من هذين السطرين include # وكذلك السطر الثالث الذي يحتوي على العبارة using namespace std ;

- يأتي بعد ذلك قسم الإعلانات (declaration section) ، وفيه نعِّرف الثوابت: MIDDLE , LAST, FIRST ، وتوضح التعليقات معنى كل واحد من هذه الثوابت / الأسماء التعريفية.

- باقي البرنامج عبارة عن تعريف الدالة main ، حيث يشتمل أول سطر على رأس / عنوان الدالة (function heading) : اسمها (name of the function) مسبوقاً بالكلمة المحجوزة int، ومتبوعاً بالقوسين ( ) ، وهما يخبران البرنامج المترجم (compiler) أن main هو اسم دالة وليس اسم متغير أو ثابت مسمَّى . ويحتوى جسم الدالة (body of the function)على الإعلان عن متغيرين : firstLast, lastFirst ، ثم قائمة العبارات التنفيذية. ويقوم البرنامج المترجم بتحويل هذه العبارات إلى تعليمات بلغة الآلة (machine language instructions) وهذه التعليمات هي التي يتم تنفيذها أثناء مرحلة تنفيذ البرنامج (execution phase of the program).

- تنتهي الدالة main بإعادة القيمة 0 كقيمة الدالة:

  return 0

- وكما أشرنا سابقاً فإن الدالة تعيد قيمة صحيحة لنظام التشغيل (operating system) عندما ينتهي تنفيذها ، وهذه القيمة

الصحيحة يطلق عليها "حالة الخروج" (exit status) . وفي معظم نظم الحاسبات (computer systems) تعاد حالة الخروج 0 للدلالة على إتمام البرنامج بنجاح (successful completion of the program) بينما تعاد أي قيمة غير صفرية (nonzero value) ماعدا ذلك.

القوالب/ العبارات المركبة (Blocks / Compound Statements)

القالب (أو العبارة المركبة) هو متتابعة من عبارات (عددها صفر أو أكثر) (sequence of zero or more statements) محصورة بين (enclosed by) القوسين{ } ولا توجد فاصلة منقوطة (semicolon) بعد نهاية القالب/العبارة المركبة أي لا توجد فاصلة منقوطة يمين القوس } .

وبالتالي فإن جسم أي دالة هو مثال لقالب/ لعبارة مركبة. ومن أمثلة العبارات (statements) عموماً: عبارة الإسناد، عبارة الإخراج، الإعلان، العبارة المركبة، العبارة الخالية. والعبارة الخالية (null statement) هي مجرد فاصلة منقوطة، وتبدو هكذا:

```
;
```

وهي لا تفعل شيئاً على الإطلاق وقت التنفيذ، حيث ينتقل التنفيذ إلى العبارة التالية، ولذلك نادراً ما تستخدم.

ويلاحظ أن الإعلان (declaration) يعتبر عبارة (statement)، ولذلك فيجوز أن يظهر في أي موضع يمكن أن تظهر فيه أي عبارة تنفيذية (executable statement) . وبمعنى آخر يجوز لأي عبارة مركبة أن تكون مزيجاً من إعلانات وعبارات تنفيذية ، هكذا مثلاً:

```
{
    char  ch ;
    ch  =  'A' ;
    cout << ch ;
```

```
        string  str ;
        str  =  "Salam" ;
        cout <<  str ;
   }
```

ولكن المعتاد أن نكتب أولاً جميع الإعلانات مع بعضها البعض قبل بداية العبارات التنفيذية ، هكذا:

```
   {
        char  ch ;
        string  str ;
        ch   =   'A' ;
        cout << ch ;
        str  =  "Salam" ;
        cou  <<  str ;
    }
```

المُشَغِّل المبدئي للغة C++ (The C++ Preprocessor)

نفرض أننا كتبنا البرنامج البسيط التالي:

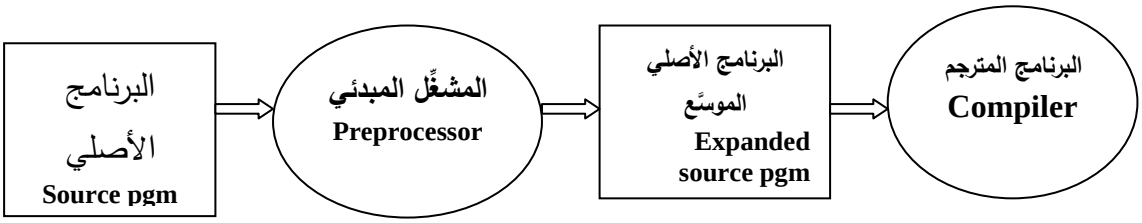
```
// * * * * * * * * * * *
// Advice program
// * * * * * * * * * * *
int   main   ( )
{
        cout << " Do not become angry " << endl ;
        return 0
}
```

البرنامج المترجم (compiler) يدرك أن الاسم التعريفي int هو كلمة محجوزة في لغة C++ ، وأن الاسم التعريفي main هو اسم دالة ما ، ولكنه لا يعلم شيئاً عن كل من الاسمين التعريفيين cout, endl حيث أننا لم نعلن عن أي منهما كمتغير ، أو ثابت مسمى ، وهما ليسا من الكلمات المحجوزة ، ولذلك فإنه يصدر رسالة خطأ (error message) . ولمعالجة هذا الخطأ فإننا نقوم بما يلي:

أولاً : نكتب في البداية السطر

```
        # include < iostream>
```

كما فعلنا في البرامج السابقة (PrintName, sample في هذا الفصل ، و Payroll في الفصل التمهيدي). وهذا السطر يعني إدخال (inserting) محتويات ملف (file) اسمه iostream في البرنامج . وهذا الملف يحتوي على إعلانات cout, endl وأشياء (items) أخرى نحتاجها لإنجاز سيل المدخلات والمخرجات (to perform stream input and output) . و أى سطر #include لا يتناوله البرنامج المترجم وإنما برنامج يعرف باسم "المشغِّل المبدئي" (preprocessor) ، وهو برنامج يعمل كمرشح (filter) أثناء مرحلة الترجمة (compilation phase) ، حيث يمر البرنامج المصدري/ الأصلي (source program) عبر المشغِّل المبدئي في طريقه إلى البرنامج المترجم (compiler)، كما بالشكل التالي:



شكل ٢- ١

المشغل المبدئي للغة C++

وأي سطر يبدأ بالعلامة # لا يعتبر من عبارات لغة C++ (وبالتالي فهو لا ينتهي بفاصلة منقوطة) ، ويطلق عليه " موجِّه للمشغِّل المبدئي (a preprocessor directive) . ويقوم المشغِّل المبدئي بتوسيع (expanding) الموجِّه # include عن طريق إدخال محتويات الملف المذكور اسمه (named file) وذلك داخل البرنامج المصدري/الأصلي (into source pgm). وأي ملف يظهر اسمه في موجِّه # include directive يطلق عليه "ملف المقدمة" (header file) . وتحتوي ملفات المقدمة على إعلانات الثوابت والمتغيرات وأنواع البيانات والدوال (constant, variable, data type, and function declarations) التي يحتاجها أى برنامج.

ويلاحظ أن القوسين > < (angle brackets) في الموجِّه (directive) مطلوبان وهما يطلبان من المشغِّل المبدئي أن يبحث عن الملف في دليل include القياسي (standard **include** directory) [وهو موضع ما (a location) في نظام الحاسوب (computer system) الذي يحتوي على جميع ملفات المقدمة المرتبطة بمكتبة C++ القياسية (related to the C++ standard library)]. والملف iostream يحتوي على الإعلانات الخاصة بتسهيلات امكانات الإدخال والإخراج (declarations of input/output facilities) والملف string الذي ظهر سابقاً في سطر

```
# include  < string >
```

يحتوى على الإعلانات الخاصة بنوع البيانات (data type) string . وبإذن الله سنستخدم فيما بعد ملفات مقدمة قياسية (standard header files)أخرى غير iostream, string .

ثانياً: نلاحظ في برنامجنا السابق Advice program أنه حتى مع إضافة موجِّه المشغِّل المبدئي (preprocessor directive) # include <iostream> فإن عملية ترجمة (compilation) البرنامج لا تتم بنجاح ، وذلك لأن المترجم (compiler) لم يتعرف (recognize) بعد على الاسمين التعريفيين cout, endl ويتضح سبب المشكلة مما يلي:

فضاء الأسماء (Namespace)

يعلن ملف المقدمة iostream (وكذلك يعلن كل ملف مقدمة قياسي) عن أسمائه التعريفية (identifiers) في فضاء أسماء (في حيِّز البرامج المسمَّاة) (namespace) يُدعَى std:

```
namespace     std
{
                إعلانات المتغيرات ، وأنواع البيانات ، و ...
}
```

ويمكن الوصول (accessing) مباشرة إلى أى اسم تعريفي معلن في قالب فضاء أسماء (namespace block) معين بواسطة العبارات الموجودة داخل هذا القالب (within the block) فقط. وللوصول إلى أي اسم تعريفي مخبوء (hidden) داخل فضاء أسماء فيمكن للمبرمج اتباع إحدى عدة طرق ، نذكر منها طريقتين:

الطريقة الأولى: أن يستخدم لهذا الاسم التعريفي اسماً مؤهَّلاً **(qualified name)**. والاسم المؤهل يتكون من اسم فضاء الأسماء يعقبه المعامل (operator) : : [ويسمي معامل تحليل المجال (scope resolution operator)] ويليه الاسم التعريفي المطلوب ، هكذا مثلاً :

std: :cout

وبالتالي يصبح البرنامج هكذا:

```
# include < iostream >
int  main  ( )
{
        std : :  cout << " Do not become angry " << std : : endl ;
        return 0,
}
```

ويلاحظ أنه يجب أن يكون كل من cout , endl مؤهلاً (qualified)، وبعبارة أخرى يجب تأهيل كل منهما.

الطريقة الثانية : أن يستخدم عبارة تُدعى موجِّه (directive) using :

```
using namespace   std ;
```

وعندما نضع هذه العبارة في مطلع البرنامج قبل الدالة main ، فإنه يمكن الوصول إلى جميع الأسماء التعريفية في فضاء الأسماء std من أي عبارة في البرنامج دون الحاجة إلى تأهيل هذه الأسماء. وهكذا يصبح برنامجنا Advice program بالصورة التالية :

```
# include < iostream >
using namespace   std ;
int  main  ( )
{
        cout << " Do not become angry " << endl ;
        return 0;
}
```

وهذه الطريقة الثانية هي التي اتبعناها في كل من البرنامجين السابقين: PrintName program, sample program ، وهي التي سنتبعها بإذن الله في الفصول التالية.

التحكم في كيفية ظهور المخرجات

(أ) السطور الفارغة (Blank Lines)

يمكننا التحكم في الفراغات الرأسية (vertical spacing) باستخدام المعالج (endl (manipulator في عبارة الإخراج.

مثال ٢-٨:

اكتب مخرجات كل من قطع البرامج التالية:

(أ)

```
cout <<  " If you ask, " ;
cout <<  " ask of Allah." << endl ;
cout <<  " If you seek help, " ;
cout <<  " seek help of Allah." << endl;
```

(ب)

```
cout <<  " If you ask, "  << endl ;
cout <<  " ask of Allah." << endl ;
cout <<  " If you seek help," << endl ;
cout <<  " seek help of Allah." << endl;
```

(ج)

```
cout <<  " If you ask, "  << endl ;
cout <<  " ask of Allah." ;
cout <<  " If you seek help, "  << endl ;
cout <<  " seek help of Allah." << endl;
```

الحل:

(أ)

```
If you ask, ask of Allah.
If you seek help, seek help of Allah.
```

(ب)

If you ask,
ask of Allah.
If you seek help,
seek help of Allah.

(ج)

If you ask,
ask of Allah. If you seek help,
seek help of Allah.

مثال ٢-٩:

اكتب مخرجات كل من قطع البرامج التالية:

(أ)

```
cout << "Say : I believe in Allah, "  << endl ;
cout << endl ;
cout << "and thereafter be upright. "  << endl ;
```

(ب)

```
cout << "Say : I believe in Allah, "   << endl  <<  endl ;
cout << "and thereafter be upright. "  << endl ;
```

(ج)

```
cout << "Say : I believe in Allah, "   << endl  <<  endl
     << "and thereafter be upright. "  << endl ;
```

الحل:

(أ) ، (ب) ، (ج)

Say : I believe in Allah,

and thereafter be upright.

نلاحظ في القطعة (أ) التي تتكون من ثلاث عبارات إخراج أن العبارة الأولى تؤدى إلى طباعة الجملة

Say: I believe in Allah,

ثم يقوم الأمر endl بتحريك مؤشر الشاشة (screen cursor)إلى بداية السطر التالي. وأما العبارة الثانية فإنها لا تطبع شيئاً ولكنها تنقل المؤشر إلى بداية السطر التالي. وتقوم العبارة الثالثة والأخيرة بطباعة الجملة

and thereafter be upright.

وتنهي هذا السطر . وبذلك تكون المخرجات الناتجة من الثلاثة سطور المبينة (حيث السطر الأوسط فارغ) .

ونلاحظ أنه كلما استخدمنا endl مباشرة بعد endl أخرى فأنه يظهر عندنا سطر فارغ . فإذا استخدمنا endl ثلاث مرات متعاقبة ظهر عندنا سطران فارغان، وإذا استخدمنا endl أربع مرات متعاقبة ظهر عندنا ثلاثة سطور فارغة، وهكذا ... ولذلك فأن قطعة البرنامج (ب) تؤدي إلى ظهور المخرجات نفسها التي نحصل عليها من القطعة (أ) .

وأما القطعة الأخيرة (ج) والتي تتكون من عبارة إخراج واحدة [تؤدي أيضاً إلى ظهور المخرجات نفسها التي نحصل عليها من كل من القطعتين (أ) ، (ب)] فإنها توضح أنه يمكن كتابة عبارة واحدة من عبارات C++ على أكثر من سطر واحد في البرنامج ، لأن المترجم (compiler) يعتبر الفاصلة المنقوطة (semicolon) – وليس نهاية السطر – هي نهاية العبارة.

(ب) الفراغات داخل السطر الواحد **(Blanks within a line)**

يمكن التحكم في الفراغات الأفقية (horizontal spacing) في المخرجات عن طريق إرسال رموز فراغات إضافية (extra blank characters) لسيل المخرجات (output stream) . [لاحظ أن رمز الفراغ الناتج عن الضغط على قضيب المسافات (spacebar) في لوحة المفاتيح (keyboard) هو رمز صحيح (valid character) في لغة C++.

مثلاً للحصول على مخرجات بالشكل التالي:

```
        *       *       *

    *       *       *       *

        *       *       *
```

نكتب العبارات

```
cout   << "   *       *       *      " << endl << endl ;
cout   << " *     *       *       *  " << endl << endl ;
cout   << "   *       *       *      " << endl ;
```

حيث جميع الفراغات والنجوم (asterisks) محصورة بين الحاصرتين المزدوجتين [علامتي التنصيص/ الاقتباس] (double quotes)، فتتم طباعتها حرفياً (print literally) كما هي مكتوبة في البرنامج. وأما endl الزائدة (extra) فهي تعمل على ترك سطر فارغ بين سطري النجوم.

وإذا أردنا طباعة فراغات – أي ظهورها في المخرجات – فيجب أن تكون محصورة بين الحاصرتين المزدوجتين ، وإلا فإنها لا تظهر في المخرجات ، فمثلاً العبارة

```
cout << '*' <<                    '*' ;
```

تؤدي إلى ظهور المخرجات:

```
**
```

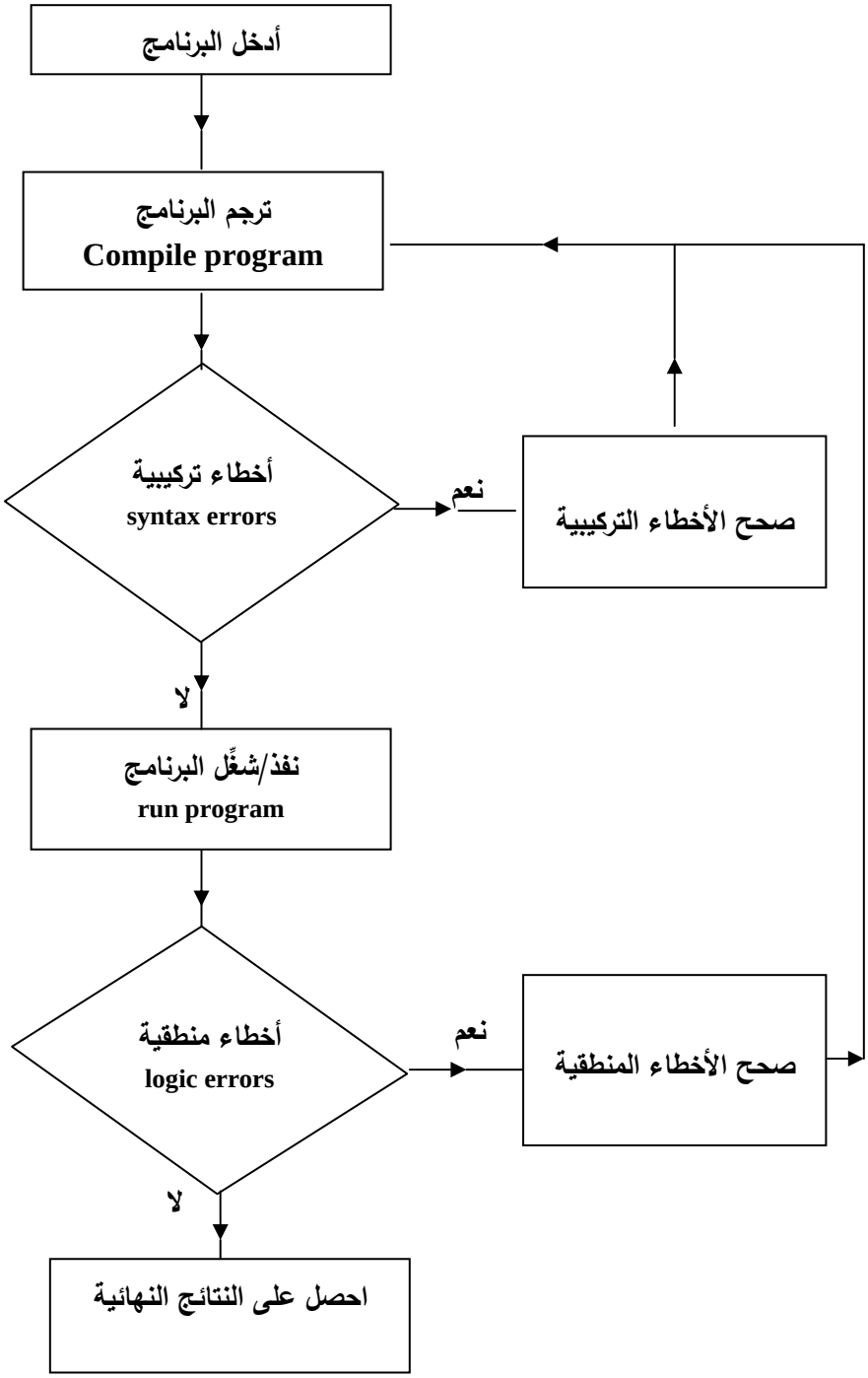
أي إلى ظهور نجمتين متجاورتين دون أي فراغ بينهما رغم الفراغات الكثيرة التي تركناها في عبارة الإخراج ، لأن هذه الفراغات لم تكن محصورة بين حاصرتين مزدوجتين.

إدخال البرنامج وتصحيحه وتنفيذه
(Program Entry, Correction, and Execution)

لإدخال البرنامج يقوم المستخدم بإدخال اسمه (user name) وكلمة السر (password) ، ويقوم الحاسوب بتشغيل المعدِّل / المحرر/المنقِّح (run the editor). والمعدِّل (editor) عبارة عن برنامج يسمح للمستخدم بإنشاء وتعديل برامج (creating and modifying programs) عن طريق إدخال معلومات (information) في مساحة (area) من حيِّز التخزين الثانوي في الحاسوب (computer's secondary storage) تدعى ملفاً (file).

ووحدة المعلومات الأساسية (basic unit of information) في المعدِّل هي شاشة عرض (display screen)مليئة بالرموز (full of characters) . والمعدِّل يسمح للمستخدم بتعديل/بتغيير (changing) أي شئ يراه على الشاشة. وبعد أن يقوم المستخدم بإنشاء ملف جديد ، فإن المعدل يمسح الشاشة ليعرف المستخدم أن الملف فارغ (file is empty) . ثم يقوم المستخدم بإدخال البرنامج عن طريق لوحة المفاتيح والباحث / الجوَّال (mouse)للعودة وعمل التصحيحات اللازمة عند الحاجة.وبعد إدخال البرنامج وتخزينة في ملف (stored in a file) يمر البرنامج بمرحلتين أساسيتين:

ترجمة البرنامج وتشغيله(Compiling and Running a Program)



شكل ٢-٢

عملية تصحيح الأخطاء (إزالة العِلل)Debugging Process

- المرحلة الأولى : مرحلة ترجمة البرنامج (انظر شكل ٢-٢) يقوم المترجم (compiler) بترجمة البرنامج وتخزين صيغته / نسخته بلغة الآلة (machine language version) في ملف. وقد يعرض المترجم نافذة (window) برسائل (messages) تشير إلى الأخطاء التركيبية (syntex errors) في البرنامج. وبعض النظم تسمح لك بأن تضغط (click) على رسالة الخطأ (error message) لتضع مؤشر الشاشة (cursor) في نافذة المعدِّل/ المنقِّح (editor window) تلقائياً (automatically) عند الوضع (position)/النقطة (point) التي اكتُشف عندها الخطأ (error was detected). فتعود للمعدِّل (editor) لتصحيح الخطأ، ثم تشغِّل المترجم (run the compiler)مرة أخرى ، وهكذا تستمر عملية تصحيح الأخطاء التركيبية إلى أن تتم عملية ترجمة البرنامج بنجاح دون أي أخطاء ، وعندها يمكن تشغيل / تنفيذ البرنامج (running / executing the program).

- المرحلة الثانية : مرحلة تشغيل / تنفيذ البرنامج (انظر شكل ٢-٢)

  - بعض النظم تنتقل تلقائياً إلى مرحلة تنفيذ البرنامج بعد نجاح ترجمته.
  - في نظم أخرى عليك إعطاء أمر منفصل (separate command) لتنفيذ البرنامج.
  - نظم أخرى تطلب منك تحديد خطوة إضافية يطلق عليها التوصيل/الربط (linking) بين ترجمة البرنامج وتشغيله.

وأيا كانت سلسلة الأوامر (series of commands) التي يستخدمها النظام فإن النتيجة واحدة وهي أنه يتم تحميل البرنامج في الذاكرة (loading the program into memory) ، وتنفيذه بواسطة الحاسوب.

وعند تنفيذ البرنامج قد يتضح ظهور أخطاء منطقية (logical errors) ، أي أخطاء في تصميم (design) البرنامج أي خوارزميته تؤدى إلى إعطاء نتائج غير صحيحة ، أو قيام البرنامج بعمل غير المطلوب منه تنفيذه . وهنا يجب تصحيح الخوارزمية والعودة إلى المعدِّل (editor) لتصحيح البرنامج . ثم نقوم بترجمة البرنامج وتشغيله مرة أخرى ، وهكذا نستمر في تكرار عملية تصحيح الأخطاء /إزالة العلل (debugging process) إلى أن يعمل البرنامج بنجاح ويقوم بتنفيذ ما هو مطلوب منه (انظر شكل ٢-٢) .

وبعد الانتهاء من العمل على الحاسب الشخصي (personal computer) قم بحفظ (saving)ملفاتك واخرج من المعدِّل (quit the editor) . واذكر أن إطفاء الجهاز (turning off the power) يؤدى إلى محو (wiping out) ما هو موجود في ذاكرة الحاسب(computer's memory) ، وأما ملفاتك فهي مخزونة بأمان على القرص (stored safely on disk) . ومن الحكمة أخذ الاحتياط دائماً بعمل بديل / نسخة (backup / make a copy of) من ملفات البرنامج (program files) من وقت لآخر على قريص (قرص صغير مرن) قابل للنزع (removable diskette). وذلك لأنه إذا حدث عطل/ تلف مادي (hardware failure) في قرص في الحاسب فعادة يكون من المستحيل استعادة الملفات. وأما مع وجود النسخة البديلة (backup copy) على قريص (diskette) فيمكننا إعادة تخزين (restoring) الملفات في القرص بمجرد إصلاحه.

مثال ٢-١٠:

نفرض أن مكتب الخريجين بالجامعة سيوجه خطاباً شخصياً (رسالة شخصية) إلى كل من الطلاب والطالبات الذين أتموا متطلبات التخرج، وذلك

للتهنئة والدعوة إلى حفل التخرج بالجامعة. ومحتوى الخطاب هو نفسه لجميع الأشخاص باستثناء اسم الشخص الموجه إليه الخطاب.

وفي بعض المواضع تخاطب الرسالة الشخص (الطالب / الطالبة) باسمه الكامل (full name)، وفي مواضع أخرى تستخدم لقباً (title) [مثل الأخ/الأخت (Br./Sr.)] ، وأحياناً أخرى تخاطبه باسمه الأول فقط. ولذلك فسيكون لدى مكتب الخريجين ملف بيانات (data file) يحتوي على جميع أسماء الخريجين بحيث يتكون كل اسم من أربعة أجزاء:

اللقب	:	title
الاسم الأول	:	first name
الحرف الأول من الاسم الأوسط	:	middle initial
الاسم الأخير	:	last name

المطلوب كتابة برنامج أولي/تمهيدي (preliminary program) يوجه لشخص واحد فقط، بحيث يبدأ البرنامج بمجموعة من ثوابت سلاسل الرموز المسماة (set of named string constants) تحتوي على الأربعة أجزاء – المشار إليها – لاسم الشخص. ويستخدم البرنامج تعابير التعاقب (concatenation expressions) لصياغة قيم متغيرات سلاسل الرموز (string variables) بالصيغ المختلفة المطلوبة في الرسالة. وبالتالي يمكن إنشاء / تكوين جميع سلاسل رموز الأسماء قبل تنفيذ عبارات الطباعة.

وتحتاج رسالة التهنئة إلى الاسم في الصيغ الأربع التالية:

- الاسم الكامل مع اللقب
- الاسم الأخير مسبوقاً باللقب
- الاسم الأول فقط
- الاسم الأول والاسم الأخير (بدون لقب أو الحرف الأول من الاسم الأوسط)

الحل:

```
//*******************************************************
 //FormLetter program
 //This program prints a form letter for congratulating
 //a graduate student.
 //It uses the four parts of a name to build name strings in four
 //different formats to be used in personalizing the letter
//*******************************************************
#include <iostream.h<
#include "string"
using namespace std;
const string  TITLE = "Br.";                    // Salutary title const string
FIRST_NAME = "Abdullah";                  // FirstName of addressee
const string  MIDDLE_INITIAL = "I";     // Middle initial
const string  LAST_NAME = "AlMobarak";   // LastName of
addressee

int main( )
}
   string first;            // Holds the firstName plus a blank
  string fullName;     // Complete name, including title
   string firstLast;     // FirstName and lastName
  string titleLast;     // Title followed by the lastName

 //   Create firstName with blank

   first = FIRST_NAME:" " +

 //   Create full name

   fullName = TITLE + " " + first + MIDDLE_INITIAL:
   fullName = fullName + ". " + LAST_NAME:
```

```
// Create first and lastName

  firstLast = first + LAST_NAME;

// Create title and lastName

  titleLast = TITLE + " " + LAST_NAME;

// Print the form letter

cout    << "Dear" << fullName << "," << endl << endl;
cout    << "Assalamo A'laikom" << endl << endl;
cout    << "Congratulations" << first << "for your graduation!"
        << endl;
cout    << "Remember" << titleLast << "that he who follows a way"
        <<  endl;
cout    << "seeking knowledge, Allah will facilitate for him"
        << endl;
cout    << "a way to Paradise." << firstLast << "," << "we are"
        << endl;
cout    << "pleased to invite you to the graduation party"
        << endl;
cout    << "in the campus hall." << endl << endl;
 cout   << "Graduation Office" << endl;
return 0;
}
```

وفيما يلي مخرجات هذا البرنامج:

```
Dear Br. Abdullah I. AlMobarak,

Assalamo A'laikom

Congratulations Abdullah for your graduation!
Remember Br. AlMobarak that he who follows a way
seeking knowledge, Allah will facilitate for him
a way to Paradise. Abdullah AlMobarak, we are
pleased to invite you to the graduation party
in the campus hall.
Graduation Office
```

ملاحظات عامة على تصحيح الأخطاء

١- يجب الإعلان عن أي اسم تعريفي ليس كلمة محجوزة من كلمات لغة C++ . فإذا استخدمنا اسماً لم يُعلن عنه من قبل – سواء في عبارات الإعلان في البرنامج أو بذكر ملف مقدمة (by including a header file) – فإننا نحصل على رسالة خطأ (error message).

٢- إذا حاولنا الإعلان عن اسم تعريفي هو نفسه كلمة محفوظة (reserved word) في لغة C++ ، فإننا نحصل على رسالة خطأ من المترجم.

٣- لغة C++ تُميِّز بين الحروف الكبيرة والصغيرة في الأسماء التعريفية (case sensitive language) ، فإذا اختلف اسمان تعريفيان في حرف بين كبير وصغير، فإنهما يعاملان كاسمين تعريفيين مختلفين. ويلاحظ أن كلمة main وجميع الكلمات المحفوظة في لغة C++ تُكتب بحروف صغيرة (lowercase) فقط.

٤- لاستخدام أسماء تعريفية من المكتبة القياسية، مثل cout, string يجب:

- إما أن نعطي اسماً مؤهلاً (qualified name) مثل std : : cout
- أو أن نضع الموجِّه (directive) **using** قرب مطلع البرنامج:
  using namespace std;

٥- الثابت الحرفي الرمزي (char literal) يبدأ وينتهي بحاصرة مفردة (علامة تنصيص واحدة) (an apostrophe / single quote) ، بينما الثابت الحرفي من سلاسل الرموز (سلسلة الرموز الحرفية) يبدأ و ينتهي بحاصرة مزدوجة (علامتي تنصيص) (double quote) .

٦- إذا استخدمنا الحاصرة المعكوسة (reverse apostrophe) : (`) والموجودة على معظم لوحات المفاتيح (keyboards) – بدلاً من الحاصرة العادية : (') في الثوابت الحرفية الرمزية (char literals) فإن المترجم يعطي رسالة خطأ.

٧- إذا احتجنا لاستخدام الحاصرة المزدوجة (double quote) داخل سلسلة رموز حرفية (literal string) فإننا نستخدم الرمزين \" (two symbols) معاً في سطر واحد. أما إذا استخدمنا الحاصرة المزدوجة فقط ، فإنها ستعني نهاية سلسلة الرموز، وبالتالي ينظر المترجم إلى بقية سلسلة الرموز على أنها خطأ.

٨- في عبارة الإسناد يجب التأكد من أن الاسم التعريفي الموجود يسار علامة = هو متغير (variable) وليس ثابتاً مسمى .

٩- عند إسناد قيمة (value) لمتغير سلسلة رموز (string variable) يجب أن يكون التعبير الموجود يمين علامة = هو تعبير سلسلة رموز (string expression) ، أو سلسلة رموز حرفية (literal string) أو رمزاً char .

١٠- في التعبير التعاقبي (concatenation expression) يجب أن يكون أحد مُعامَلَي المؤثر + (two operands of) على الأقل من النوع string (of type). فمثلاً لا يجوز أن يكون كلاهما سلسلة رموز حرفية (literal string) أو قيمة رمزية (char value) . فالتعبير التعاقبي التالي – على سبيل المثال – خاطئ وينتج عنه رسالة خطأ تركيبي (syntax error message) :

"programming" + "language"

١١- تأكد من أن كل عبارة (statement) تنتهي بفاصلة منقوطة (semicolon) ، باستثناء العبارات المركبة (compound statements) والتي لا نضع فيها فاصلة منقوطة يمين القوس الأيمن }.

أنواع البيانات في لغة C++

تنقسم أنواع البيانات الداخلية (built-in data types) في لغة C++ إلى ثلاثة أنواع رئيسية:

- الأنواع البسيطة (simple types).
- الأنواع المبنية (structured types).
- أنواع عناوين (address types).

ويبين شكل ٢-٣ هذه الأنواع الرئيسية وما يندرج تحت كل منها من أنواع تفصيلية. وفي بقية الفصل الحالي نتناول أنواع البيانات العددية (numeric data types) ونقصد بها أساساً الأعداد الصحيحة (integers) والأعداد ذوات العلامة الكسرية العائمة (floating point numbers) [أي الأعداد الحقيقية (real numbers) التي تحتوي على العلامة / النقطة العشرية (decimal point)] . وأما تفاصيل باقي الأنواع فستأتي بإذن الله تباعاً في بقية فصول الكتاب.

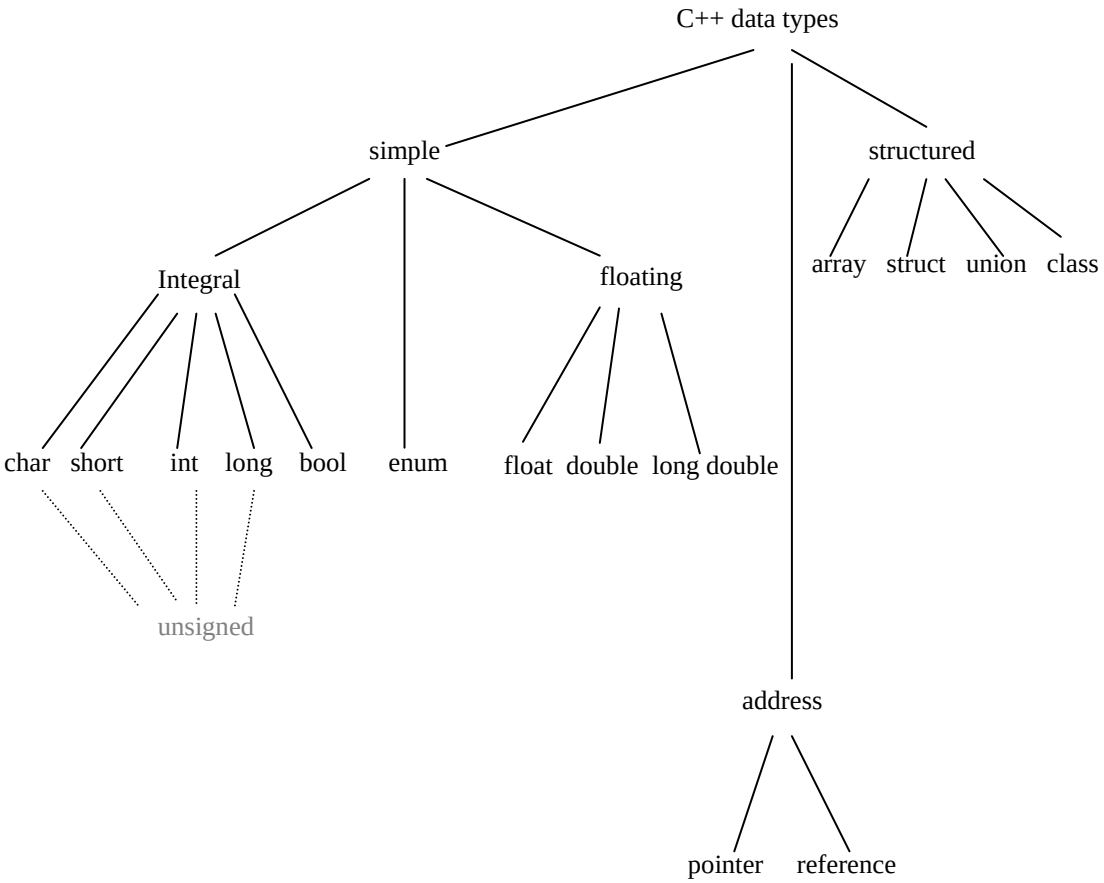
أنواع البيانات العددية (Numeric Data Types)

أولاً : الأنواع الصحيحة/التكاملية (Integer / Integral Types)

يطلق على أنواع البيانات char, short, int, long "أنواع صحيحة / تكاملية" لأنها تشير إلى قيم صحيحة (integer values) وهي الأعداد الكاملة التي لا تحتوي على جزء كسري (fractional part). [وسنؤجل الحديث عن النوع التكاملي الباقي bool إلى الفصل القادم بإذن الله].

وأبسط صيغة للقيمة الصحيحة في لغة C++ هي متتابعة (sequence)من رقم واحد أو أكثر (أي من رقم واحد أو رقمين أو عدة أرقام) بحيث لا تحتوي على فاصلة (comma)، مثل :

54 3 314 0 4200



شكل ٢-٣

أنواع البيانات في لغة C++

C++ Data Types

فمثلاً لا يجوز كتابة القيمة الصحيحة 4200 هكذا : 4,200 لأنه لا يسمح بوجود الفاصلة.

ووجود إشارة ناقص (minus sign) **قبل القيمة الصحيحة يجعل العدد الصحيح عادة سالباً ، مثل : 800 – والاستثناء** (exception) **هو عند إضافة الكلمة**

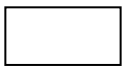
المحجوزة unsigned صراحة (explicitly) لاسم نوع البيانات (data type name) مثل:

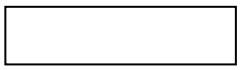
unsigned int

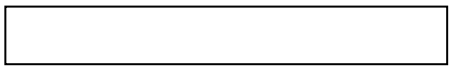
حيث تعني قيمة صحيحة دون إشارة ، أي يفترض أن هذه القيمة الصحيحة موجبة (positive) أو صفر (zero) . وأنواع البيانات دون إشارة (unsigned type) نادرة الاستخدام حيث تستخدم في مواضع خاصة فقط.

وأنواع البيانات char , short , int , long تمثل أطوالاً/أحجاماً مختلفة (different sizes) من الأعداد الصحيحة من الأصغر [وحدات ثنائية (بتّات) أقل (fewer bits)] إلى الأكبر [وحدات ثنائية أكثر (more bits)]. وتعتمد الأحجام على الآلة المستخدمة (machine dependent) ، أي تتغير من آلة لأخرى. وفي إحدى الآلات قد تكون أحجام / أطوال الأنواع المختلفة بالشكل التالي مثلاً:

خلية الذاكرة char (memory cell) 

خلية الذاكرة short (memory cell) 

خلية الذاكرة int (memory cell) 

خلية الذاكرة long (memory cell) 

وفي آلة أخرى قد يكون حجم int مساوياً لحجم long. وعموماً كلما كانت هناك وحدات ثنائية أكثر (more bits) في خلية الذاكرة (memory cell)، أمكننا تخزين قيمة صحيحة أكبر.

ويلاحظ انه رغم أننا استخدمنا النوع char فيما سبق في هذا الفصل لتخزين بيانات رمزية (character data) مثل 'A' ، فهناك أسباب تجعل لغة C++ تصنِّف النوع char كنوع صحيح / تكاملي، وسنتعرض لهذه الأسباب بإذن الله عند الحديث عن الأنواع البسيطة للبيانات (simple data types) .

ويعد النوع int أكثر الأنواع استخداماً للتعامل مع البيانات الصحيحة ، وتقريباً نستخدمة دائماً للقيم الصحيحة ، ولكن أحياناً نستخدم النوع long إذا احتاج البرنامج قيماً أكبر من قيمة int العظمي (maximum int value) . وفي بعض الحاسبات الشخصية (personal computers) يكون مدى (range) قيم int هو:

- 32768 ⟶ + 32767

وأكثر شيوعاً (more commonly) يكون المدى هو :

- 2147483648 ⟶ + 2147483647

وإذا حاول البرنامج حساب قيمة أكبر من القيمة العظمى التي تسمح بها الآلة (machine's maximum value) فإن النتيجة هي : فيض زائد لعدد صحيح (integer overflow) ، وبعض الآلات تعطي رسالة خطأ (error message) في حالة حدوث الفيض الزائد.

تحذير : في حالة القيم الصحيحة في لغة C++ إذا بدأنا ثابتاً حرفياً (literal constant) بصفر ٠ (على اليسار) فإن العدد يعامَل على أنه عدد ثماني (octal number) [أي باستخدام الأساس 8 base] وليس عدداً عشرياً (decimal number) [أي باستخدام الأساس ١٠] . فمثلاً إذا كتبنا ٠٢٥ فإن مترجم (compiler) C++ يعتبر هذه القيمة

العدد العشري 21 [2×8 + 5×1 = 16+5 = 21]

وليس العدد العشري 25 [2×10 + 5×1 = 20 +5 = 25]

ولذلك فلا تبدأ أي ثابت صحيح عشري (decimal integer constant) بصفر، إلا إذا كنا نريد العدد 0 (صفر) [لأنه هو نفسه 0 في النظامين العشري والثماني]

أنواع النقطة العائمة/العلامة الكسرية العائمة (Floating-Point Types)

تستخدم هذه الأنواع – والتي يطلق عليها اختصاراً الأنواع العائمة (floating types) – لتمثيل (representing) الأعداد الحقيقية (real numbers). والأعداد ذوات النقطة العائمة (floating point numbers) تحتوى على جزء صحيح (integer part) وجزء كسري (fractional part) بينهما نقطة عشرية (a decimal point) ، مثل :

3.14, 15.0, 0.72

وقد لا يوجد أحد الجزئين ، مثل: 3. , .7

ولكن لا يجوز أن يختفي كلا الجزئين معاً ، مثل : .

ويلاحظ أن وجود الصفر 0 في بداية عدد مثل 0.72 لا يعني أن العدد ثماني (octal) ، فوجود الصفر في بداية الأعداد الصحيحة فقط هو الذي يجعل العدد ثمانياً.

وكما أن الأنواع الصحيحة في لغة C++ يمكن تمثيلها بأحجام/بأطوال مختلفة (char, short, int, long) ، فكذلك أنواع النقطة العائمة تمثل بأحجام مختلفة: (float , double, long double) ، أصغرها float وأكبرها long double، وكلمة double تشير إلى double precision أي دقة مضاعفة / مزدوجة. ومرة أخرى – كما ذكرنا سابقاً – فإن الأحجام/الأطوال المضبوطة تعتمد على الآلة المستخدمة، وكلما زاد الحجم كلما زاد مدى القيم المسموح بها، وزادت الدقة [أي زاد عدد الأرقام المعنوية (significant digits) في العدد، ولكن على حساب زيادة أكبر في حيز الذاكرة (memory space) الذي يحتوي على العدد (holds the number).

ويمكن للقيم ذوات النقطة العائمة أن تحتوي على أُس (exponent) ، كما في الاصطلاح العلمي (scientific notation) ، حيث يُكتب العدد كقيمة مضروبة

بالعدد 10 مرفوعاً لقوة (power) معينة، ففي لغة C++ بدلاً من كتابة $3.504x10^{12}$ نكتب : 3.504E12 حيث حرف E يعني الأُس (exponent) بالنسبة للأساس (base) 10 . ولا يشترط في العدد الذي على يسار حرف E أن يشتمل على نقطة عشرية. ومن أمثلة الأعداد ذوات النقطة العائمة في الاصطلاح العلمي:

2.7453E–12, 8.65224E4, 5E18

ومعظم البرامج لا تحتاج النوعين: double, long double، حيث يعطينا النوع float عادةً دقة كافية (sufficient precision) ومدى (range) كافياً لقيم (values) الأعداد ذوات النقطة العائمة. وحتى الحاسبات الشخصية (personal computers) تعطي قيماً من النوع floatبدقة تصل إلى ستة أو سبعة أرقام معنوية (significant digits) وقيمة عظمى (maximum value) تصل لنحو 3.4E+38

ومن الجدير بالذكر أن الحاسبات لا يمكنها دائماً تمثيل الأعداد ذوات النقطة العائمة بالضبط (exactly) ، فالحاسب يخزن البيانات بالصيغة الثنائية (binary) [أي باستخدام الأساس 2(base)] ، وكثير من القيم ذوات النقطة العائمة لا يمكن إلا تقريبها في هذا النظام الثنائي للأعداد (binary number system) ، فقيمة مثل 4.8 قد تطبع 4.7999998. وفي معظم الأحوال نتوقع أن تكون هناك عدم دقة بسيطة (slight inaccuracy) في الأرقام أقصي يمين الجزء الكسري (rightmost fractional digits) من القيمة الناتجة ، وهي ليست نتيجة أي خطأ من المبرمج.

الإعلان عن الأنواع العددية
(Declarations for Numeric Types)

يمكننا الإعلان عن كل من ثوابت مسمَّاة (named constants) ومتغيرات (variables) من أي من النوعين int, float .

الإعلان عن الثوابت المسمَّاة (Named constant Declarations)

في حالة الإعلان عن ثوابت مسماة عددية تكون القيم الحرفية (literal values) عددية (numeric) بدلاً من رموز (characters) بين حاصرات فردية أو مزدوجة (single or double quotes).

وفيما يلي أمثلة للإعلان عن ثوابت تعرِّف قيماً من النوعين int, float (بالإضافة – للمقارنة والتوضيح – إلى قيم من النوعين char , string) .

```
const   float   PI=3.14159;
const   float   E=2.71828;
const   int     MAX_SCORE=100;
const   int     MIN_SCORE=0;
const   char    LETTER='D';
const   string  NAME="Khadeejah";
```

استخدام الثوابت المسمَّاة بدلاً من القيم الحرفية
(Using named constants instead of literals)

يفضل عموماً استخدام الثوابت المسماة بدلاً من القيم الحرفية، فبالإضافة إلى أن استخدام هذه الثوابت يجعل البرنامج أكثر وضوحاً وأسهل في القراءة والفهم، فإن استخدام الثوابت المسماة يجعل عمليات تعديل (modifying) البرنامج أيسر. ولتوضيح ذلك نفرض أننا كتبنا برنامجاً لحساب قيم زكاة المال وأن قيمة النصاب (Al_NISAB) حالياً – وقت كتابة البرنامج – تساوي 340 ديناراً (وهي تقدر بسعر 85 جرام من الذهب الخالص، فإذا كان سعر الجرام الآن أربعة دنانير فإن النصاب يساوي 340 = 4 x 85) . ثم نفرض أنه بعد فترة قد تغيرت قيمة النصاب فأصبحت 255 ديناراً (أي سعر جرام الذهب قد نقص إلى ثلاثة دنانير) . فكي يعطي البرنامج نتائج صحيحة الآن يجب أن نتتبع كل قيمة حرفية 340 في البرنامج ونغيرها إلى 255 . وإذا كان في البرنامج قيمة حرفية 340 لا علاقة لها بالنصاب ولكنها تمثل شيئاً آخر في البرنامج فإننا لا نغير هذه القيمة.

يمكننا – بدرجة كبيرة – تبسيط عملية تعديل البرنامج باستخدام ثابت مسمًّى بدلاً من ثابت حرفي، بأن نعلن مثلاً عن الثابت المسمّى AL_NISAB ، وأن قيمته–الأصلية قبل تعديل البرنامج– تساوي 340:

```
const   float  AL_NISAB = 340 ;
```

فإذا أردنا تعديل البرنامج، فكل ما نعمله هو مجرد تغيير الإعلان بتعديل قيمة AL_NISAB:

```
const   float  AL_NISAB = 255 ;
```

أي نقوم بإجراء تعديل واحد فقط، يقوم هو بعد ذلك بتغيير جميع قيم النصابAL_NISAB في البرنامج إلى القيمة الجديدة 255.

وبالتالي ينتج عن هذا التعديل تعديل نتائج جميع العمليات المترتبة على قيمة النصاب. أما أي قيمة حرفية 340 في البرنامج لا علاقة لها بالنصاب فإنها لا تتأثر بهذا التعديل ولا تتغير.

الإعلان عن المتغيرات (Variable Declarations)

نستخدم النوعين العدديين (numeric types) int, float للإعلان عن المتغيرات العددية كما يظهر في الأمثلة التالية:

```
int     num;              // An integer number
int     studentCount;     // Number of students
int     sumOfScores;      // Sum of their scores
float   average;          // Average of the scores
char    grade;            // Student's letter grade
string  StuName;          // Student's name
```

وإذا كانت لدينا الإعلانات السابقة فيمكننا كتابة عبارات الإسناد التالية:

```
num             = 100;
student Count   = num;
sum Of Scores   = 8247;
average         = 82.47;
grade           = 'B';
```

ونلاحظ في أي من عبارات الإسناد هذه أن نوع بيانات التعبير (الموجود في الطرف الأيمن) يتفق مع (matches) نوع بيانات المتغير (الموجود في الطرف الأيسر) الذي يُسند إليه. وسنرى بإذن الله فيما بعد في هذا الفصل ماذا يحدث إذا اختلف النوعان.

التعابير الحسابية البسيطة (Simple Arithmetic Expressions)

يمكننا حساب قيم عددية باستخدام تعابير. وسنبدأ أولاً بإذن الله بتعابير بسيطة تحتوي على معامل/مؤثر (operator) واحد فقط ، ثم ننتقل بعد ذلك إلى التعابير المركبة (compound expressions) التي يشمل الواحد منها عمليات متعددة (multiple operations).

المعاملات/المؤثرات الحسابية (Arithmetic Operators)

يتكون التعبير من ثوابت ومتغيرات ومؤثرات. فإذا كان لدينا مثلاً الإعلانات:

```
int     num;
int     count;
float   length;
float   rate;
float   avg;
```

فمن أمثلة التعابير البسيطة:

num + 2	4 - num	num * count
avg + 1.5	length	0.5 + rate

والمؤثرات المسموح بها في تعبير ما تعتمد على أنواع بيانات (data types) الثوابت والمتغيرات الموجودة في التعبير . والمؤثرات الحسابية هي :

+	:	علامة زائد الأحادية	(unary plus)
–	:	علامة ناقص الأحادية	(unary minus)
+	:	الجمع	(addition)
–	:	الطرح	(subtraction)
*	:	الضرب	(multiplication)
/	:	•خارج القسمة الحقيقي عند القسمة ذات النقطة العائمة (أي النتيجة ذات نقطة عائمة)	(floating-point division) (floating-point result)
		•خارج القسمة الصحيح عند القسمة الصحيحة (لا يوجد جزء كسري بالنتيجة)	(integer quotient) (integer division) (no fractional part)
٪	:	الباقي الصحيح بعد القسمة الصحيحة	(modulus: remainder from integer division)

وكل من المعاملين / المؤثرين الأولين مؤثر أحادي (unary operator) أي يؤثر على معامَل واحد (one operand)فقط، بينما كل من المؤثرات الخمسة الباقية مؤثر ثنائي (binary operator) حيث يؤثر على مَعاملين.

أما المؤثر الأحادي "+" والمؤثر الأحادي "–" فيستخدمان هكذا:

– 48 +154.73 – rate

ونادراً ما نستخدم المؤثر الأحادي " + " ، فبدون أي إشارة (sign) يُفترض أي ثابت عددي (numeric constant) أنه موجب (positive).

وبالنسبة لمؤثر القسمة الثنائي " / " فيمكن أن يستخدم مع عددين ذوي نقطة عائمة أو مع عددين صحيحين، ويعطى النتيجة من نوع العددين نفسه. فإذا استخدم مع عددين حقيقيين (أي ذوي نقطة عائمة) أعطى عدداً حقيقياً (أي ذا نقطة عائمة) ، مثل:

11.0 / 4.0 ⟶ 2.75
8.0 / 4.0 ⟶ 2.0
10.0 / 4.0 ⟶ 2.5

وإذا استخدم مع عددين صحيحين أعطى عدداً صحيحاً هو خارج القسمة الصحيح (integer quotient)، **مثل**

11 / 4 ⟶ 2
8 / 4 ⟶ 2
10 / 4 ⟶ 2
8 / 8 ⟶ 1
8/ 7 ⟶ 1
8 / 9 ⟶ 0

وأما مؤثر القسمة الثنائي " ٪ " فلا يستخدم إلا مع عددين صحيحين ويعطى الباقي الصحيح (integer remainder) ، **مثل :**

11 % 4 ⟶ 3
8 % 4 ⟶ 0
10 % 4 ⟶ 2
8 % 8 ⟶ 0
8 % 7 ⟶ 1
8 % 9 ⟶ 8
10 % 2.5 ⟶ **خطأ لأن أحد المعامَلين ليس عدداً صحيحاً**

وإذا كان أحد العددين (أحد المعامَلين) سالباً فإن إشارة الباقي قد تختلف من مترجم (compiler) C++ **إلى آخر.**

ونظراً لأن الحاسب لا يستطيع القسمة على صفر، لذلك فلا يجوز القسمة على صفر عند استخدام أي من المؤثرين " / " أو " ٪ " . وبالتالي فإن أيّاً من التعابير

8 / 0 , 8.0 / 0.0, 8 ٪ 0

ينتج عنه خطأ (produces an error).

وأما مؤثر الجمع الثنائي " + " ومؤثر الطرح الثنائي " – " فيعطيان النتائج الجبرية المعتادة، فمثلاً :

$3 + 5 \longrightarrow 8$
$2.5 - 8.2 \longrightarrow -5.7$

ونظراً لأنه يُسمح بوجود المتغيرات في التعابير، فلذلك العبارات التالية عبارات إسناد صحيحة:

```
count  =  num + 4;
count  =  num / 2;
num    =  count * 2;
num    =  8 % count;
num    =  num + 1;
count  =  count + num;
```

ويلاحظ أن هناك فارقاً بين عبارة الإسناد والمتطابقة الرياضية . فمثلاً عبارة الإسناد:

```
count = count + num ;
```

تعني أن نجمع القيمة المخزونة في count مع القيمة المخزونة في num، ونقوم بتخزين هذا المجموع في count ليحل محل القيمة (السابقة) المخزونة هناك.

بينما المتطابقة الرياضية

```
count = count + num
```

لا تكون صحيحة إلا إذا كانت قيمة num تساوي صفراً. أما عبارة الإسناد

```
count = count + num
```

فهي صحيحة لأي قيمة من قيم num.

مثال ٢-١١:

نفرض أن درجة/نقطة تجمد الماء (freezing point of water) ٣٢ وأن درجة/نقطة غليان الماء (boiling point of water) ٢١٢. اكتب برنامجاً لحساب الدرجة المتوسطة بين هاتين الدرجتين.

الحل:

```
// *******************************************************
 //FreezeBoil program
 //This program computes the midpoint between
 //the freezing and boiling points of water
// *******************************************************
#include <iostream<

using namespace std;

const float FREEZE_PT = 32.0;   // Freezing point of water
const float BOIL_PT = 212.0;    // Boiling point of water

int main( )
}
  float avgTemp;              // Holds the result of averaging
                              // FREEZE_PT and BOIL_PT

  cout << "Water freezes at " << FREEZE_PT << endl;
  cout << " and boils at " << BOIL_PT << " degrees." << endl;

  avgTemp = FREEZE_PT + BOIL_PT;
  avgTemp = avgTemp / 2.0;

  cout << "Halfway between is" ;
  cout << avgTemp << " degrees." << endl;

  return 0;
{
```

يبدأ البرنامج بتعليق يشرح وظيفة البرنامج. ثم في قسم الإعلانات نعرِّف الثابتين FREEZE_PT, BOIL_PT . **ويشتمل جسم** (body) **الدالة الرئيسية** main function **على إعلان عن المتغير** avgTemp، **ثم على متتابعة من العبارات**

التنفيذية، حيث تقوم هذه العبارات بطباعة رسالة، وجمع درجتي التجمد والغليان ، وقسمة المجموع على 2 ، وأخيراً طباعة النتيجة.

مؤثرا الزيادة والنقصان
(Increment and Decrement Operators)

بالإضافة إلى المؤثرات الحسابية فإن لغة C++ تشتمل على مؤثرين : أحدهما للزيادة والآخر للنقصان:

مؤثر الزيادة (increment) : ++

مؤثر النقصان (decrement) : --

وكل منهما مؤثر/معامِل أحادي (unary operator) معامَله (operand) اسم متغير واحد (single variable name). و بالنسبة للمعامَل الصحيح (integer operand) أو المعامِل ذي النقطة العائمة (floating-point operand) فإن مؤثر الزيادة يضيف 1 إلى المعامَل ، بينما مؤثر النقصان يطرح 1 من المعامَل. فإذا فرضنا أن num يحتوي على القيمة 8 فإن العبارة

```
num ++ ;
```

تجعل num يحتوي على القيمة 9، أي أننا نصل إلى النتيجة نفسها التي نحصل عليها بعبارة الإسناد

```
num = num + 1;
```

وبالمثل فإن تأثير العبارة

```
num -- ;
```

هو نفسه تأثير عبارة الإسناد

```
num = num -1;
```

[ملاحظة : لغة C++ سميت بهذا الاسم باعتبار أنها صيغة محسَّنة/مطوَّرة/مزادة (enhanced / "incremented" version) من لغة C]

ويلاحظ أن كلا من المؤثرين -- ، ++ يمكن أن يكون مؤثراً سابقاً (prefix operator):

+ + num; – – num;

أو مؤثراً لاحقاً (postfix operator):

num + +; num – –;

والمؤثر السابق والمؤثر اللاحق يعطيان النتيجة نفسها، فمثلاً كل من العبارتين

+ + num;
num + +;

تضيف 1 **إلى قيمة / محتويات** num

التعابير الحسابية المركبة (Compound Arithmetic Expression)

في التعابير البسيطة السابقة لم يظهر في أي تعبير أكثر من مؤثر/معامل واحد ، ولم نخلط في أي تعبير بين قيم صحيحة وقيم ذوات نقطة عائمة. وفيما يلي سنتناول بإذن الله التعابير المركبة التي قد يحتوي الواحد منها على عدة مؤثرات أو على أنواع بيانات مختلطة (mixed data types).

قواعد الأولوية (Precedence Rules)

تتكون التعابير الحسابية عموماً من عدة ثوابت ومتغيرات ومؤثرات وأقواس. ويتم إجراء العمليات (performing the operations) **وفقاً للترتيب** (order) **التالي:**

(١) **أولاً : التعابير الجزئية** (subexpressions) **بين الأقواس**
ثم : المؤثر " + " الأحادي ، والمؤثر " – " الأحادي
ثم : ٪ / *
ثم : – +

أي أن أى عملية في أي سطر لها أولوية قبل أى عملية أخرى في أي سطر اسفل منه.

مثلاً عملية القسمة " / " لها أولوية قبل عملية الجمع الثنائي " + " . فإذا كتبنا مثلاً:

```
avg  =  8.0 + 6.0 / 2.0 ;
```

فإن قيمة avg تصبح

```
avg  →  8.0 + 3.0 = 11.0
```

فإذا أردنا إجراء عملية الجمع الثنائي " + " قبل عملية القسمة، فإننا نستخدم قوسين كما يلي (لأن إيجاد قيمة التعبير الجزئي بين قوسين له الأولية الأولى):

```
avg  =  (8.0 + 6.0) / 2.0 ;
```

وفي هذه الحالة فإن قيمة avg تصبح:

```
avg  →14.0 / 2.0 = 7.0
```

(٢) في حالة وجود عدة مؤثرات ثنائية (several binary operators) لها الأولوية نفسها (مثل ٪ ، / ، *) ، فإن ترتيب التجميع (grouping order / associativity) يكون من اليسار لليمين، أي أن أولوية العمليات في هذه الحالة تكون من اليسار لليمين في التعبير المطلوب إيجاد قيمته ، فمثلاً

```
int1 – int2 + int3
```

تعني: `(int1 – int2) + int3`

وليس: `int1 – (int2 + int3)`

وكذلك `float1 / float2  * float 3`

تعني `(float1 / float2) * float 3`

وليس `float1 / (float2 * float 3)`

مثال ٢-١٢

أوجد قيمة كل من التعابير التالية المكتوبة بلغة C++ .

(i) 10 / 2 * 3

(ii) 10 ٪ 3 – 4 / 2

(iii) 5.0 * 2.0 / 4.0 * 2.0

(iv) 5.0 * 2.0 / (4.0 * 2.0)

(v) 5.0 + 2.0 / (4.0 * 2.0)

الحل:

(i) 15 (ii) - 1 (iii) 5.0

(iv) 1.25 (v) 5.25

ملاحظة: في لغة C++ جميع المؤثرات الأحادية (مثل المؤثر الأحادي " + " ، والمؤثر الأحادي " – ") لها أولوية تجميع من اليمين لليسار (right to left associativity) ،

فمثلاً: x + – **تعني** (x +) –

وليس x (+ –) **والتي لا معنى لها.**

تحويل النوع (Type Conversion)

يتم تخزين القيم الصحيحة (integer values) **داخل ذاكرة الحاسوب بطريقة مخالفة لتلك التي يتم بها تخزين القيم ذوات النقطة العائمة** (floating-point values). **فنمط (شكل) الوحدات الثنائية (البتات)** (pattern of bits) **التي تمثل** (represents) **الثابت** 2 **لا تشبه إطلاقاً نمط تلك التي تمثل الثابت** 2.0 ، **وذلك لأن الأعداد ذوات النقطة العائمة تحتاج تمثيلاً خاصاً داخل الحاسوب. فماذا يحدث إذا خلطنا في التعبير الحسابي نفسه أو في عبارة الإسناد نفسها بين قيم صحيحة وقيم ذوات نقطة عائمة ؟**

ننظر أولاً في حالة عبارة الإسناد ثم بعد ذلك في حالة التعبير الحسابي.

أولاً: الخلط في عبارة الإسناد

نفرض أن لدينا الإعلانين

```
int     someInt ;
float   someFloat ;
```

وبناءً عليهما فإن someInt يمكن أن يحتفظ بقيم صحيحة فقط و someFloat يمكن أن يحتفظ بقيم ذوات نقطة عائمة فقط.

فإذا كتبنا عبارة الإسناد

```
someFloat = 8;
```

فقد يبدو أنها ستقوم بتخزين القيمة الصحيحة 8 في someFloat. ولكن هذا ليس صحيحاً، فالحاسب يرفض تخزين أي قيمة ليست ذات نقطة عائمة في someFloat. وما يحدث هو أن البرنامج المترجم (compiler) يقوم بإدخال (inserting) تعليمات إضافية بلغة الآلة (extra machine language instructions) تقوم أولاً بتحويل (converting) 8 إلى 8.0 ، ثم بعد ذلك تخزين 8.0 في someFloat .

هذا التحويل التلقائي/ الضمني (implicit / automatic conversion) لقيمة من نوع بيانات إلى نوع آخر يطلق عليه:

التحويل القسري / التلقائي / الضمني للنوع
(Type Coercion)/(Implicit/Automatic Type Conversion)

وبالكيفية نفسها، فإن عبارة مثل

```
someInt =  3.7 ;
```

تؤدي إلى تحويل قسري/تلقائي للنوع (type coercion) ، وذلك لأنه إذا أسندت قيمة ذات نقطة عائمة لمتغير صحيح، فإن الجزء الكسري (fractional part) يحذف / يقطع (turncated / cut off) ، أي أن someInt تُسند له القيمة 3 .

بالطبع كان من الأفضل تجنب هذا الخلط بين الأنواع في عبارات الإسناد ، وكتابة العبارتين السابقتين هكذا:

```
someFloat = 8.0 ;
someInt = 3 ;
```

فذلك يجعل البرنامج أوضح وأبعد عن الالتباس بالنسبة لقارئه.

وكثيراً ما ينشأ التحويل القسري بسبب وجود تعابير وليس مجرد ثوابت ، فمثلاً كل من عبارتي الإسناد

```
someFloat = 3 * someInt + 2 ;
someInt    = 5.2 / someFloat – 8.13 ;
```

ينشأ عنها تحويل قسري.

تحويل النوع (صب النوع/التحويل الصريح للنوع)
(Type Conversion)/(Type Casting/ Explicit Conversion)

لجعل البرنامج أكثر وضوحاً يمكننا استخدام عملية تحويل نوع تعبيرٍ ما إلى النوع الآخر قبل إسناد قيمة لمتغير ما، مثل :

```
someFloat = float (3 * someInt + 2) ;
someInt     = int (5.2 / someFloat – 18.3)
```

فنستخدم float لتحويل قيمة التعبير الصحيحة إلى قيمة ذات نقطة عائمة، ونستخدم int لتحويل قيمة التعبير ذات النقطة العائمة إلى قيمة صحيحة (بحذف الجزء الكسري).

ويلاحظ أن هذا التحويل الصريح للنوع يؤدي إلى النتيجة نفسها التي نحصل عليها بالتحويل الضمني (التحويل القسري) ، والفارق الوحيد بين التحويلين هو وضوح التحويل الصريح ، فالعبارتان التاليتان مثلاً تؤديان إلى النتيجة نفسها، ولكن العبارة الثانية أوضح

```
someInt = someFloat + 8.5 ;
someInt = int (someFloat + 8.5) ;
```

ولتقريب قيمة ذات نقطة عائمة لأقرب عدد صحيح قبل تخزينها في متغير صحيح، نضيف لها أولاً 0.5 ثم نحذف الجزء الكسري من القيمة الناتجة ، هكذا:

```
someInt = int (someFloat + 0.5) ;
```

ملاحظة: هذه الطريقة تفترض أن someFloat عدد موجب (positive).

ثانياً: الخلط في التعبير الحسابى :

من أمثلة الخلط في التعبير الحسابي:

```
someInt * someFloat
4.8 + someInt – 3
```

ويطلق على تعبير كهذا يحتوي على معامَلات (operands) من أنواع مختلفة : تعبير مختلط النوع (mixed type / mode expression)

وعندما يؤثّر معامل (operator) على قيمتين: إحداهما صحيحة (integer) والأخرى ذات نقطة عائمة (floating point) فإنه يتم تحويل قسري ضمني للنوع (implicit type coercion) كما يلي:

١- تُحوَّل القيمة الصحيحة مؤقتاً (temporarily coerced) إلى قيمة ذات نقطة عائمة.

٢- تُجرى العملية (operation is performed).

٣- تكون النتيجة قيمة ذات نقطة عائمة.

فمثلاً لحساب قيمة التعبير 3 – someInt + 4.8 حيث someInt يحتوي على القيمة 2 :

- تُحوَّل قيمة someInt قسراً (coerced) إلى 2.0.هذا التحويل مؤقت (temporary) فقط، بمعني أنه لا يؤثر على القيمة المخزونة في someInt.
- تتم عملية الجمع لتعطي القيمة 6.8.
- تُحوَّل القيمة 3 قسراً (coerced) إلى 3.0 .

- تتم عملية الطرح لتعطي النتيجة 3.8 وهي قيمة ذات نقطة عائمة.

وكما في حالة عبارات الإسناد يمكننا استخدام التحويل الصريح للنوع (explicit type casting) داخل التعبير (within expression) لتقليل فرص الخطأ (to lessen the risk of errors) . فكتابة تعبير مثل:

```
float (someInt) + someFloat
```

أو :

```
4.8 + float (someInt – 3)
```

يجعل القصد من التعبير واضحاً.

والتحويل الصريح للنوع – بالإضافة إلى فائدته في وضوح البرنامج – يلزم أحياناً استخدامه للحصول على نتائج صحيحة ، بحيث أنه إذا لم يستخدم في تلك الأحيان فإننا نحصل على نتائج خاطئة.. ومن أمثلة ذلك :

نفرض أن لدينا الإعلانات:

```
int     sum ;
int     count ;
float   average ;
```

ونفرض أن sum يحتوي حالياً على القيمة 60

وأن count يحتوي حالياً على القيمة 80

وأن sum تمثل مجموع عدة قيم صحيحة عددها count

وأن المطلوب حساب القيمة المتوسطة average لهذه القيم الصحيحة.

عبارة الإسناد

```
average = sum / count ;
```

تعطي نتيجة خاطئة حيث تخزن القيمة 0.0 في average، وذلك لأن العملية الحسابية هي قسمة قيمة صحيحة على قيمة صحيحة أي أنها قسمة صحيحة integer division ، وخارج القسمة الصحيح هو القيمة الصحيحة 0 (60/80 ← 0) ، التي تحولها الآلة ضمنياً (implicitly coerces it) إلى القيمة 0.0 قبل تخزينها في average.

وللحصول على القيمة المتوسطة المطلوبة وبطريقة واضحة أيضاً ، فإننا نكتب عبارة الإسناد

average = float (sum) / float (count) ;

وهي تخزن القيمة 0.75 في average وذلك لأن القسمة هنا هي قسمة ذات نقطة عائمة (floating-point division) وليست قسمة صحيحة (integer division) .

ملاحظة: في موضوع تحويل النوع ركزنا فيما سبق على النوعين int , float ، ولكن من الممكن أن تكون هناك أيضاً قيم من الأنواع char, short , double . وسنعود لهذا الموضوع بتفصيل أكثر بإذن الله فيما بعد، ولكن عموماً يفضل تجنب خلط قيم من هذه الأنواع في أي تعبير.

استدعاءات الدوال (Function Calls)

أ) الدالة التي تعيد قيمة (Value – Returning Function)

رأينا في مثال ٢-١ كيف أن الدالة الرئيسية main استدعت كلا من الدالة Square والدالة Cube لحساب 27^2 و 27^3 على الترتيب. مثلاً تم استدعاء الدالة Cube بكتابة متتابعة الرموز

Cube (27)

ويقصد باستدعاء / تنشيط الدالة (function call / invocation) آلية (mechanism) نقل التحكم (transferring control) إلى الدالة المستدعاة ، بمعني أن يوقف الحاسوب مؤقتاً الدالة المستدعية (مثلاً main) (puts the calling function on hold) ، ويبدأ تشغيل الدالة المستدعاة (مثلاً Cube) (starts the called function running) . وعندما تنتهي الدالة المستدعاة من تنفيذ عملها يعود الحاسوب إلى الدالة المستدعية عند الموضع الذي تركها فيه – وليس عند بدايتها – ليكمل تنفيذها أن كان لها بقية.

وفي الاستدعاء السابق Cube (27) يقال للعدد 27 إنه وسيط/ وسيط فعلي (argument / actual parameter) وبواسطته–أي عن طريقه–يتم الاتصال بين الدالتين main, Cube . وعموماً بواسطة قائمة الوسطاء (argument list) – أي عن طريق هذه القائمة – يتم الاتصال بين الدوال ، وقد تشتمل هذه القائمة على وسيط (فعلي) واحد (كما في حالة كل من الدالتين Square, Cube) ، وقد لا تشتمل على أي وسيط (كما في حالة الدالة main) ، وقد تشتمل على وسيطين (فعليين) ، أو أكثر. وإذا كان هناك أكثر من وسيط واحد فتوضع فاصلة بين كل وسيطين متجاورين.

والدالة التي تعيد قيمة (value-returning function) تستخدم في التعابير بالكيفية نفسها التي تستخدم بها المتغيرات والثوابت. والقيمة التي تحسبها الدالة تأخذ موضعها في التعبير، فمثلاً عبارة الإسناد

```
someInt = Cube (2) * 10 ;
```

تقوم بتخزين القيمة 80 في someInt ، حيث تنفَّذ الدالة Cube أولاً لحساب مكعب 2 أي لحساب القيمة 8 التي تصبح الآن جاهزة للاستخدام في بقية التعبير، فتُضرب × 10. ولاحظ أن استدعاء الدالة [Cube (2)] له أولوية تسبق (تعلو على) (higher precedence) عملية الضرب ، فنتيجة تنفيذ الدالة يجب أن تكون جاهزة قبل إجراء عملية الضرب.

وفيما يلي بعض القواعد الخاصة باستخدام الدالة التي تعيد قيمة:

- استدعاء الدالة لا يظهر في عبارة مستقلة ، وإنما في تعبير (within an expression) .
- تحسب الدالة قيمة (نتيجة) تصبح جاهزة للاستخدام في التعبير.
- الدالة تعيد قيمة واحدة بالضبط.
- رأينا سابقاً في مثال ٢–١ أن الدالة Cube تتوقع أن تُعْطَى / يُمَرَّر إليها (given / passed) وسيط فعلي (argument) من النوع الصحيح int. فإذا

أعطيت وسيطاً فعلياً من النوع ذي النقطة العائمة (float argument) فإن البرنامج المترجم (compiler) يقوم بالتحويل القسري الضمني للنوع (implicit type coercion). فمثلاً الاستدعاء Cube (4.9) يحسب مكعب 4 وليس مكعب 4.9، أي يحسب القيمة 64.

- الوسيط الفعلي (argument) الذي يمرَّر لدالة تعيد قيمة (value-returning function) يمكن أن يكون أي تعبير (expression) من النوع المناسب (appropriate type) ، فقد يكون ثابتاً حرفياً (literal constant) – مثل قيمة 27 التي أُعطيت للدالة Cube – وقد يكون متغيراً (variable) أو ثابتاً مسمَّى (named constant) أو أي تعبير عام كما في الاستدعاء التالي مثلاً للدالة Cube في عبارة الإسناد:

```
num = Cube (int1 *  int1 + int2 * int2);
```

حيث تُحسب أولاً قيمة هذا التعبير الموجود في قائمة الوسطاء (argument list)، ثم تُمرر هذا القيمة (النتيجة) فقط – وليس التعبير نفسه – إلى الدالة . فمثلاً إذا كان int1 يحتوي على 4 ، و int2 يحتوي على 10 فإن الاستدعاء السابق للدالة يمرر القيمة 116 كوسيط فعلي (argument) للدالة Cube.

- أي تعبير موجود في قائمة وسطاء الدالة (function's argument list) يمكن أن يحتوي هو نفسه على استدعاءات لدوال. فمثلاً عبارة الإسناد السابقة يمكن أن نعيد كتابتها باستدعاء الدالة Square ضمن التعبير الموجود في قائمة وسطاء الدالة Cube هكذا:

```
num =  Cube  (Square (int1)  + Square (int2)) ;
```

الدوال المكتبية/القياسية (Standard / Library Functions)

هناك بعض العمليات الحسابية أو الدوال الشائعة التي يحتاجها كثير من الناس في البرمجة مثل إيجاد الجذر التربيعي لقيمة معينة أو إيجاد القيمة المطلقة لعدد ما . وبالتالي يكون هناك هدر كبير للوقت إذا طُلب من كل مبرمج أن يبدأ من الصفر ويكتب /ينشئ (create) الدوال التي يحتاجها لإجراء هذه العمليات

الحسابية الشائعة. فبدلاً من ذلك ولتسهيل الأمر على المبرمجين يحتوي كل نظام (system) C++ على مكتبة قياسية (standard library) تشتمل على مجموعة كبيرة من الدوال المكتوبة سابقاً (prewritten functions) وأنواع بيانات (data types) وأشياء (items) أخرى قد يحتاجها أي مبرمج بلغة C++ . وفيما يلي قائمة ببعض الدوال القياسية المكتبية شائعة الاستخدام:

ملف المقدمة Header File	الدالة Function	نوع الوسيط Argument type	نوع النتيجة Result type	الدالة الرياضية المقابلة Math Function	معنى الدالة Meaning
< cstdlib >	abs (i)	int	Int	$\|i\|$	قيمة i المطلقة
< cmath >	fabs (x)	float	float	$\|x\|$	قيمة x المطلقة
< cstdlib >	labs (j)	long	long	$\|j\|$	قيمة j المطلقة
< cmath >	sin (x)	float	float	sin x	جيب الزاوية مقاسة بالتقدير الدائري radians
< cmath >	cos (x)	float	float	cos x	جيب الزاوية مقاسة بالتقدير الدائري radians
< cmath >	tan (x)	float	float	tan x	ظل الزاوية مقاسة بالتقدير الدائري radians
< cmath >	sqrt (x)	float	float	$\sqrt{x}$	جذر x التربيعي
< cmath >	exp (x)	float	float	e^x	قيمة e (2.718…) مرفوعة للأس x
< cmath >	log (x)	float, x > 0.0	float	ln x	لوغاريتم x الطبيعي (أي للأساس e)
< cmath >	Log10 (x)	float x > 0.0	float	$\log_{10} x$	لوغاريتم x للأساس 10
< cmath >	pow (x,y)	float, float	float	x^y	x مرفوعة للأس y ، بحيث أنه: إذا كانت x = 0.0 فيجب أن تكون y>0 .وإذا كانت x ≤ 0.0 فيجب أن تكون y عدداً كاملاً (whole number)
< cmath >	ceil (x)	float	float	$\lceil x \rceil$	أصغر عدد كامل ≤ x (ceiling of x)
< cmath >	floor (x)	float	Float	$\lfloor x \rfloor$	أكبر عدد كامل ≥ x (floor of x)

جدول أكثر الدوال المكتبية القياسية استخداماً

مثال ٢-١٣: فيما يلي بعض قيم الدوال المكتبية المذكورة بالجدول السابق لتوضيح معناها

abs (-4)	⟶	4
abs (4)	⟶	4
fabs (3.8)	⟶	3.8
fabs (-2.45)	⟶	2.45
fabs (0.0)	⟶	0.0
sin (0.0)	⟶	0.0
cos (0.0)	⟶	1.0
sqrt (900.0)	⟶	30.0
sqrt (9.0)	⟶	3.0
exp (1.0)	⟶	2.71828
exp (2.0)	⟶	7.38906
log (2.718282)	⟶	1.0
log (7.389056)	⟶	2.0
log 10 (10.0)	⟶	1.0
log 10 (100.0)	⟶	2.0
pow (2.0, 5.0)	⟶	32.0
pow (9.0 , 0.5)	⟶	3.0
ceil (9.2)	⟶	10.0
ceil (-9.8)	⟶	- 9.0
floor (9.2)	⟶	9.0
floor (-9.8)	⟶	-10.0

ملاحظات على الدوال المكتبية واستخدامها

- **في الجدول السابق الخاص بالدوال المكتبية القياسية يلاحَظ من الناحية العملية/الفنية** (technically) **أن جميع الكلمات** float **التي تعطي النوع يجب أن تكون** double **، وذلك لأن الدوال المكتبية تنجز عملياتها باستخدام قيم ذوات نقطة عائمة بدقة مضاعفة/ مزدوجة ، ولكن بسبب التحويل القسري الضمني للنوع** (type coercion) **فإن الدوال تعمل وتعطي نتائج بالصورة المعتادة حين نعطيها قيماً من النوع** float.

- لاستخدام دالة مكتبية يجب أن نضع أولاً موجِّهاً include # directive قرب مطلع البرنامج، مع تحديد ملف المقدمة المناسب (appropriate header file) . هذا الموجه يضمن لنا أن يقوم المشغِّل المبدئي للغة C++ (C++ preprocessor) بإدخال إعلانات (declarations) في البرنامج تعطي البرنامج المترجم (compiler) بعض المعلومات عن الدالة. ويمكننا بعد ذلك استدعاء الدالة متى شئنا واستخدامها. وفيما يلي مثال بسيط لاستخدام دالتي sqrt, fabs .

```
# include < iostream >
# include < cmath >    //  For  sqrt ( ) and fabs ( )

using namespace std ;

.
.
.
float alpha ;
float beta ;
.
.
.
alpha = sqrt (7.3 + fabs (beta)) ;
```

- جميع الأسماء التعريفية في المكتبة القياسية موجودة في فضاء الأسماء namespace std . فإذا حذفنا الموجِّه using directive من المثال السابق، وجب علينا استخدام أسماء مؤهَّلة (qualified names) للدوال المكتبية (std : : sqrt, std : : fabs , …..)

- تحتوي المكتبة القياسية للغة C++ على عشرات الدوال التي يمكن استخدامها بالإضافة إلى المجموعة البسيطة من الدوال المكتبية القياسية المذكورة في الجدول السابق.

(ب) الدالة الخالية/الفارغة (Void Function)

استعرضنا حتى الآن الدالة التي تعيد قيمة (value-returning function). ويوجد في لغة C++ نوع آخر من الدوال ، حيث لا تعيد الدالة أي قيمة للدالة المستدعية ولكنها تقوم فقط بإجراء بعض العمليات ثم ينتهي دورها. ومثل هذا النوع من الدوال يطلق عليه : دالة لا تعيد قيمة (non-value-returning function) أو دالة خالية/فارغة الإعادة (void-returning function) أو اختصاراً دالة خالية/فارغة (void function) . وفي كثير من لغات البرمجة يطلق على مثل هذه الدالة : إجراء (procedure) . وقد مر علينا سابقاً مثال لهذه الدالة في الفصل التمهيدي في مثال ١ في برنامج أجور الموظفين Payroll program، وهي الدالة CalcPay ، حيث بدأ تعريف هذه الدالة بالكلمة void بدلاً من نوع بيانات (data type) مثل int أو float .

```
void  CalcPay ( …)
{
      ⋮
}
```

وطريقة استدعاء الدالة الخالية تختلف عن طريقة استدعاء الدالة التي تعيد قيمة، فاستدعاء الدالة التي تعيد قيمة – كما ذكرنا سابقاً– يظهر في تعبير، أما استدعاء الدالة الخالية فيكون في عبارة مستقلة بذاتها . فمثلاً في برنامج أجور الموظفين المشار إليه استدعت الدالة main الدالة الخالية CalcPay بالعبارة المستقلة التالية:

```
CalcPay (payRate, hours, wages) ;
```

صياغة المخرجات (Formatting the Output)

يقصد بصياغة مخرجات البرنامج التحكم في كيفية ظهورها على الشاشة (screen) أو على الطابعة (printer) . وقد رأينا سابقاً جانباً من هذا التحكم عن طريق ترك سطور فارغة باستخدام endl ، أو ترك فراغات داخل السطر بوضع

فراغات إضافية (extra blanks) في سلاسل الرموز الحرفية (literal strings).
والآن ندرس كيفية صياغة القيم المخرجة (output values) نفسها.

أولاً : الأعداد الصحيحة وسلاسل الرموز (Integers and Strings)

القيم الصحيحة المتعاقبة وسلاسل الرموز المتعاقبة عادة تُطبع/تُخرج متجاورة دون أي فراغات فيما بينها . فمثلاً إذا كانت المتغيرات i, j, k تحتوي على القيم 15, 2, 6 على الترتيب ، فإن العبارة

```
cout    << "Results : " << i << j << k ;
```

تطبع/تُخرج سيل الرموز (stream of characters) التالي:

```
Results: 1526
```

بدون أي فراغ بين الأعداد . وبالطبع فإنه ليس من السهل تفسير هذه المخرجات. ولفصل القيم المخرجة عن بعضها البعض، يمكننا ترك فراغ واحد (single blank) – كثابت رمزي char constant – بين أي عددين ، هكذا:

```
cout    << "Results : " << i << ' ' << j << ' ' << k ;
```

وتنفيذ هذه العبارة يؤدي إلى ظهور المخرجات التالية :

```
Results: 15 2 6
```

وإذا أردنا ترك فراغ أكبر بين القيم يمكننا استخدام سلاسل رموز حرفية literal strings عبارة عن فراغات – أي تحتوي على فراغات (containing blanks) – بدلاً من فراغ واحد فقط ، هكذا :

```
cout    << "Results : " << i << "    " << j << "      " << k ;
```

بالتالي تكون المخرجات الناتجة:

```
Results: 15     2       6
```

وهناك طريقة أخرى للتحكم في المسافات الأفقية (horizontal spacing) بين المخرجات وهي أن نستخدم معالِجات (manipulators) . وحتى الآن كنا نستخدم المعالج endl بقصد إنهاء سطر الطباعة (terminate an output line) . وفي لغة C++ فإن المعالج يتصرف كدالة–فينتج عنه تنفيذ بعض الخطوات (some action)

– وفي الوقت نفسه يتحرك كهدف بيانات (data object) فيمكنه الظهور وسط سلسلة من عمليات الإدخال (insertion operations) ، مثل:

```
cout << someInt << endl << someFloaat ;
```

والمعالجات (manipulators) لا تستخدم إلا في عبارات الإدخال والإخراج فقط.

وعبارة الإخراج (output statement) تسمح بظهور/بإخراج التعابير الحسابية (arithmetic expressions) وتعابير سلاسل الرموز (string expressions) والمعالجات .

ومكتبة C++ القياسية (C++ standard library) تحتوي على عدد كبير من المعالجات ، ولكننا الآن سننظر في خمسة منها هي:

endl, setw, fixed, showpoint, setprecision

أما الثلاثة : endl, fixed, showpoint [وكل منها ليس له وسيط] فإنها تكون جاهزة للاستخدام بمجرد كتابة #include ملف المقدمة iostream لإنجاز عمليات الإدخال/الإخراج (I/O) . وأما الإثنان : setw , setprecision [وكل منهما له وسيط] فيتطلبان أن نضيف # include ملف المقدمة iomanip:

```
# include < iostream >
# include < iomanip >

using namespace std ;
     .
     .
cout << setw (5) << someInt ;
     .
     .
```

الجدول التالي يلخص معنى أو تأثير (effect) كل من هذه المعالجات الخمسة، ونوع الوسيط الفعلي (argument type) (إن وُجد) ، وملف المقدمة (header file) المطلوب.

ملف المقدمة Header File	المعالج Manipulator	نوع الوسيط الفعلي Argument Type	التأثير Effect
< iostream >	endl	لا يوجد	ينهي سطر المخرجات الحالي
< iostream >	showpoint	لا يوجد	يفرض إظهار النقطة العشرية في المخرجات ذات النقطة العائمة
< iostream >	fixed	لا يوجد	يمنع ظهور المخرجات ذوات النقطة العائمة بالصيغة العلمية
< iomanip >	setw (n)	int	يجعل عرض المجال الذي تطبع فيه القيمة يساوي n *
< iomanip >	setprecision (n)	int	يجعل دقة النقطة العائمة n رقماً

جدول بعض المعالجات وتأثيرها

*** setw [وتعني "set width" : تحديد عرض المجال (field width) الذي تطبع فيه القيمة] تستخدم للأعداد وسلاسل الرموز ، ولا تستخدم للبيانات الرمزية (char data) . وكذلك فإن setw تطبق فقط على أول عنصر تالي في المخرجات وبعد ذلك يعاد عرض المجال إلى 0 (ويعني : استخدم فقط عدداً من المواضع قدر الحاجة) .**

: setw

وسيط setw (argument) **عبارة عن تعبير صحيح** (integer expression) **يطلق عليه "مواصفات عرض المجال"** (fieldwidth specification) **. ومجموعة مواضع الرموز** (group of character positions) **يطلق عليها " المجال"**

(field) .. وعنصر البيانات التالي (next data item) يطبع أقصى يمين المجال (right-justified) ، بحيث تترك فراغات (blanks) على اليسار لتملأ/لتشغل المجال كله.

مثال ٢-١٤ : نفرض أن لدينا متغيرين صحيحين أسندت لهما القيمتان:

```
ans = 33 ;
num = 7132;
```

فيما يلي مجموعة من عبارات الإخراج/الطباعة، ويمين كل منها المخرجات الناتجة .

العبارة Statement	المخرجات (المربع □ يعني فراغاً) Out put
cout << setw (4) << ans << setw (5) << num << setw (4) << "IO" ;	□□33□7132□□IO 4 5 4
cout << setw (2) << ans << setw (4) << num << setw (2) << "IO" ;	337132IO 2 4 2
cout << setw (6) << ans << setw (3) << "IO" << setw (5) << num;	□□□□33□IO□7132 6 3 5
cout << setw (7) << IO" << setw (4) << num;	□□□□□IO7132 7 4
cout << setw (1) << ans << setw (5) << num;	33□7132 5 المجال يتسع تلقائياً ليسع القيمة ذات الرقمين

نلاحظ في العبارة (5) أن عرض المجال (1=) ليس كافياً للقيمة (33=) المخزونة في ans ، ولذلك فإن المجال يتمدد (expands) تلقائياً ليسع/ليستوعب جميع أرقام (all digits) القيمة.

ويلاحظ أن ضبط عرض المجال (setting the fieldwidth) عملية تتم لمدة واحدة فقط (one time action) بمعنى أنها تكون صالحة فقط لأول قيمة تالية في المخرجات. وبعد إخراج /كتابة هذه القيمة يُعدَّل/يعود (resets) عرض المجال إلى القيمة 0 ، بمعني : زد طول المجال (extend the field) أي زد عدد مواضعه (number of positions) بالقدر الذي يكفي بالضبط (exactly) لإخراج القيمة، أي أن المجال يمتد قدر الحاجة بالضبط.

فمثلاً في المثال السابق حيث

```
ans = 33 ;      num = 7132 ;
```

إذا كانت لدينا العبارة

```
cout   <<     " IO "  << setw (5)  << ans  <<  num ;
```

فبعد طباعة قيمة ans (أقصى يمين الخمسة المواضع المحددة) يعود عرض المجال إلى الصفر ثم يمتد إلى أربعة مواضع فقط هي التي نحتاجها لطباعة قيمة num ، وبالتالي نحصل على المخرجات:

```
IO □□□ 337132
```

الأعداد ذوات النقطة العائمة (Floating-point Numbers)

*) يمكننا تحديد عرض المجال للقيم ذوات النقطة العائمة تماماً كما في حالة القيم الصحيحة مع مراعاة أن النقطة العشرية تشغل موضعاً وذلك عندما نحدد عدد مواضع الرموز (specify number of character positions) فمثلاً القيمة 4.85 تحتاج أربعة مواضع إخراج وليس ثلاثة.

فإذا كانت x تحتوي على القيمة 4.85 فإن العبارة

```
cout   <<  setw (4)  <<  x  << endl
       <<  setw (6)  <<  x  << endl
       <<  setw (3)  <<  x  << endl ;
```

تؤدي إلى ظهور المخرجات:

```
4.85
  4.85
4.85
```

لاحظ في السطر الثالث من المخرجات أن عرض المجال المحدد (3) غير كاف، وبالتالي فقد زاد هذا العرض تلقائياً ليستطيع احتواء العدد.

*) بالنسبة للقيم الكبيرة ذوات النقطة العائمة فإنها تُطبع بالصيغة العلمية/ بالاصطلاح العلمي (scientific notation) **E** . فمثلاً القيمة 123456789.5 قد تُطبع في بعض النظم هكذا: 1.23457E+08

*) يمكننا استخدام المعالِج المسمَّى **fixed** لجعل جميع المخرجات المتتالية (subsequent) ذوات النقطة العائمة تظهر بالصيغة العشرية (decimal form) بدلاً من الصيغة العلمية، هكذا :

```
cout <<  fixed  << 3.8  *  x ;
```

*) لغة C++ لا تطبع النقطة العشرية مع الأعداد الكاملة، فالقيمة 95.0 مثلاً تطبع هكذا : 95

*) لإظهار النقاط العشرية (decimal points) في المخرجات المتتالية ذوات النقطة العائمة حتى مع الأعداد الكاملة (whole numbers) يمكننا استخدام المعالج **showpoint** :

```
cout  <<  showpoint  <<  floatVar ;
```

*) لتحديد عدد الأرقام/المواضع العشرية (number of decimal places) [وهي الأرقام الموجودة يمين العلامة العشرية] التي نريد ظهورها في المخرجات نستخدم المعالج **setprecision** كما يلي:

نفرض أننا نريد طباعة قيمة الزكاة Zakat (التي تقدر بنسبة 2.5 % من المدخرات savings) لثلاثة أرقام عشرية. العبارة التالية تؤدي إلى هذه النتيجة:

```
cout << fixed << setprecision(3) << "Zakat is KD"  << savings * 0.025;
```

فوسيط المعالج setprecision يحدد عدد الموضع العشرية المطلوبة بشرط أن يكون المعالج fixed قد تم تحديده . وبعكس setw الذي ينطبق فقط على أول قيمة تالية تُطبع، فإن القيمة التي تُرسل إلى المعالج setprecision يظل تأثيرها مستمراً لجميع المخرجات التالية (all subsequent output) إلى أن نقوم نحن بتغييرها باستدعاء آخر للمعالج setprecision .

وفيما يلي بعض الأمثلة لاستخدام setprecision مع setw .

مثال ٢-١٥

الجدول التالي يعرض مجموعة من عبارات الإخراج/الطباعة ويمين كل منها المخرجات الناتجة بفرض أن قيمة x هي المعطاة يسار العبارة

قيمة x	العبارة	المخرجات (المربع □ يعني فراغاً)
310.0	cout << fixed ; cout << setw (10) << setprecision (2) << x;	□□□□310.00
310.0	cout << setw (10) << setprecision (5) << x;	□310.00000
310.0	cout << setw (7) << setprecision (5) << x;	310.00000 (يمتد لتسعة مواضع)
4.827	cout << setw (6) << setprecision (2) << x;	□□4.83 (أخر رقم يظهر على اليمين يُقرَّب)
4.827	cout << setw (6) << setprecision (1) << x;	□□□4.8 (أخر رقم يظهر على اليمين يُقرَّب)

ومرة أخرى نؤكد على أن العدد الكلي لمواضع الطباعة يُزاد إذا لم يكن عرض المجال الذي يحدده setw كافياً. أما عدد مواضع أرقام الكسر العشري فيتحكم فيه تماماً وسيط setprecision .

عمليات إضافية على سلاسل الرموز
(Additional String Operations)

أولاً : الدالة Length

هذه الدالة– حين تطبق على متغير سلسلة رموز (string variable) – تعيد قيمة صحيحة بدون إشارة (unsigned integer value) وهي عدد الرموز (characters) الموجودة حاليا (currently) في سلسلة الرموز.

وإذا فرضنا أن my Name متغير سلسلة رموز، فاستدعاء (calling) الدالة length يكون بالصيغة التالية:

myName . Length ( )

حيث يُذكر اسم متغير سلسلة الرموز (وهو هنا myName) تلية نقطة "." (dot/period)، ثم اسم الدالة وقائمة الوسطاء (argument list). ومع أن الدالة length لا تتطلب تمرير (passing) أي وسطاء لها ، إلا أنه يجب كتابة القوسين ( ) للدلالة على قائمة الوسطاء الخالية (empty) . وكذلك فإن length دالة تعيد قيمة (value-returning function) ، ولذلك فإن استدعاء الدالة يجب أن يظهر في تعبير (expression):

```
string   firstName;
string fullName;

firstName = "Assmaa" ;
cout << firstName. Length ( ) << endl ;              // Prints 6
fullName = firstName + "   AbuBakr" ;
cout << fullName. Length ( ) << endl ;               // Prints 14
```

ويلاحظ أنه يجب استخدام اصطلاح النقطة (dot notation) بوضع النقطة بين اسم المتغير firstName واسم الدالة length. فبعض أنواع البيانات المعرفة بالمبرمج (programmer defined data types) – مثل string – لها دوال مرتبطة

بها وتستلزم استخدام اصطلاح النقطة عند استدعاء أي دالة منها. فإذا استدعينا الدالة مباشرة دون استخدام هذا الاصطلاح بأن كتبنا

length ( )

فستظهر لنا رسالة خطأ وقت الترجمة (compile-time error message) كأن تظهر مثلاً الرسالة: UNDECLARED IDENTIFIER باعتبار أن البرنامج المترجم (compiler) يفسر هذه العبارة بأنها استدعاء دالة عادية تسمى length وليست الدالة length المرتبطة بالنوع string .

ثانياً : الدالة size

هذه الدالة تعطي النتيجة نفسها التي تعطيها الدالة length، فكثير من الناس يطلقون على طول (length) سلسلة الرموز حجم (size) السلسلة، ولذلك فالنوع string يوفر لنا هاتين الدالتين.

بمعنى أن الاستدعاء

firstName . size ( )

والاستدعاء

firstName . length ( )

يعيدان القيمة نفسها.

ذكرنا سابقاً أن الدالة length تعيد قيمة صحيحة دون إشارة . فإذا كتبنا عبارة مثل

len = firstName. Length ( )

لتخزين القيمة التي تعيدها الدالة length في المتغير len ، فماذا يكون نوع بيانات len ؟ بالنسبة لبعض البرامج المترجِمة (compiler) يُستخدم النوع unsigned int، والبعض الآخر يَستخدم النوع unsigned long . فحتى لا نضطر لمعرفة أي النوعين هو النوع الصحيح بالنسبة للبرنامج المترجم الذي نعمل معه، فإن النوع string يعرِّف نوع بيانات type_size يمكننا استخدامه:

```
string firstName ;
string : : size_type len ;
firstName = "Assmaa" ;
len = firstName. Length ( )     ;
```

ويلاحظ أنه يجب أن نستخدم الاسم المؤهَّل

string : : size-type

كما نفعل مع الأسماء التعريفية في فضاءات الأسماء (namespaces) وذلك لأنه بدون ذلك يصبح تعريف size_type مختفياً (hidden) داخل تعريف النوع string.

ثالثاً : الدالة find

هذه الدالة تبحث (searches) في سلسلة رموز (string) عن أول ظهور/حدوث (first occurrence) لسلسلة رموز جزئية معينة (particular substring) ، وتعيد قيمة صحيحة دون إشارة (unsigned integer value) من النوع string :: size_type هي نتيجة البحث. والسلسلة الجزئية (substring) التي تمرر كوسيط فعلي (argument) للدالة يمكن أن تكون سلسلة حرفية (literal string) أو تعبير سلسلة (string expression) . مثلاً إذا كان كل من str1, str2 من النوع string فإن الاستدعاءات التالية للدالة find صحيحة كلها:

```
str1 . find ("the")
str1 . find ( str2 )
str1 . find ( str2  + " abc " )
```

في كل من هذه الاستدعاءات يتم البحث في السلسلة str1 عن سلسلة جزئية معينة أن كانت موجودة في ثناياها أم لا . فإن كانت موجودة فإن الدالة تعيد الموضع في str1 الذي بدأت عنده السلسلة الجزئية – أي حيث بدأت المطابقة/المواءمة/المضاهاة/التوافق (matching). ويلاحظ أن المواضع (positions) تُرقَّم ابتداءً من الصفر 0 ، أي أن أول رمز (first character) في أي سلسلة يُعد في الموضع رقم 0، والرمز الثاني في الموضع رقم 1 ، وهكذا. وحتى يكون البحث ناجحاً (successful search) يجب أن يكون التوافق/التطابق تاماً (exact) بما في ذلك الكتابة بأحرف كبيرة (أو صغيرة) (identical capitalization). فإذا لم نجد هذه السلسلة الجزئية فإن الدالة تعيد القيمة الخاصة string::npos وهي ثابت مسمَّى (named constant) يعني: "not a position within the

"string أي ليست في أي موضع ضمن السلسلة . ويلاحظ أن string::npos هو أكبر قيمة ممكنة (largest possible value) من النوع string::size_type ، وتصل هذه القيمة إلى نحو 4294967295 في آلات كثيرة . وهذه القيمة مناسبة بالنسبة للرسالة "not a valid position" – أي ليس موضعاً صحيحاً – لأن العمليات string operations على سلاسل الرموز لا تجعل طول أي سلسلة يصل إلى هذا الحد

مثال ٢– ١٦:

نفرض أن لدينا القطعة (code segment) التالية:

```
string   phrase ;
string : : size_type position ;
phrase="The revelation of the Book is from Allah, the Mighty, the Wise";
```

ما هي القيمة التي تسندها للمتغير position كل من العبارات التالية ؟

```
(i)          position  = phrase.find ("the");
(ii)         position  = phrase.find ("revolution");
```

الحل:

(i) 19 [لاحظ أنه لا توافق/تطابق (match) بين the و The]

(ii) string::npos [لأنه لم يحدث تطابق/توافق]

ويمكن كذلك للوسيط الفعلي للدالة find أن يكون قيمة رمزية char (value) . وفي هذه الحالة تبحث الدالة find عن أول حدوث لهذا الرمز (character) في السلسلة الرمزية وتعيد موضعه أو تعيد string::npos إن لم تجده.

مثال ٢–١٧: ما هي مخرجات القطعة التالية:

```
string   theString ;
theString = "Allah is the Greatest" ;
cout << theString.find ('a') ;
```

الحل: المخرجات هي القيمة 4 التي هي موضع أول حدوث للحرف الصغير a **في سلسلة الرموز** theString.

وفيما يلي أمثلة أخرى لاستدعاء الدالة find.

مثال ٢-١٨: نفرض أن القطعة (code segment) **التالية قد تم تنفيذها** (has been executed).

```
string   str1 ;
string   str2;

str1 = "Programming and Problem Solving";
str2 = "gram" ;
```

الجدول التالي يعطي أمثلة لاستدعاءات مختلفة للدالة find **والقيمة التي تعيدها الدالة في كل استدعاء**

استدعاء الدالة	القيمة التي تعيدها الدالة
str1. find ("and")	12
str1. find ("Programming")	0
str2. find ("and")	string ::npos
str1. find ("Pro")	0
str1. find ("ro" + str2)	1
str1. find ("Pr" + str2)	string ::npos
str1. find (' ')	11

نلاحظ في المثال الرابع أن هناك نسختين (two copies) **من السلسلة الجزئية "Pro" في السلسلة** str1، **ولكن الدالة** find **تعيد فقط موضع** (position) **النسخة الأولى. كذلك يلاحظ أن أي نسخة إما أن تكون كلمة مستقلة** (separate word) **أو أن تكون جزءاً من كلمة** (part of a word) ، **والدالة** find **عليها مجرد محاولة مطابقة** (matching) **متتابعة الرموز** (sequence of characters) **المعطاة في قائمة الوسطاء** (argument list) . **والمثال الأخير يوضح أن الوسيط الفعلي قد يكون بسيطاً جداً كأن يكون رمزاً وحيداً** (single character) **أو حتى فراغاً وحيداً** (single blank).

رابعاً : الدالة substr

هذه الدالة تعيد سلسلة جزئية معينة (a particular substring) من سلسلة رموز (string). فإذا فرضنا مثلاً أن myString متغير من النوع string، فيمكننا استدعاء الدالة substr بعبارة مثل :

myString. substr (5,20)

حيث الوسيط الفعلي الأول "5" عبارة عن عدد صحيح بدون إشارة (unsigned integer) يحدد موضعاً (position) داخل (within) سلسلة الرموز، والوسيط الفعلي الثاني "20" عبارة عن عدد صحيح دون إشارة يحدد طول (length) سلسلة الرموز الجزئية المطلوبة (desired substring) – والدالة تعيد جزءاً (قطعة piece) من السلسلة المعطاة يبدأ من الموضع الذي حدده الوسيط الأول ويستمر لعدد من الرموز هو العدد الذي حدده الوسيط الثاني.

ولاحظ أن الدالة substr لا تغير سلسلة الرموز المعطاة myString ، وإنما تعيد قيمة سلسلة رموز جديدة مؤقتة (new, temporary string value) عبارة عن نسخة (copy) من جزء (portion) من سلسلة الرموز المعطاة.

وفيما يلي بعض الأمثلة لاستدعاء الدالة substr.

مثال ٢–١٩: نفرض أن العبارة

myString = "Verily with hardship comes ease"

قد تم تنفيذها. الجدول التالي يعطي أمثلة لاستدعاء الدالة substr ، وأمام كل استدعاء سلسلة الرموز التي تحتويها القيمة التي تعيدها الدالة.

استدعاء الدالة	سلسلة الرموز المحتواة في القيمة التي تعيدها الدالة
myString. substr (0 , 6)	"Verily"
myString. substr (7 , 9)	"with hard"
myString. substr (5 , 0)	" "
MyString. substr (27 , 40)	"ease"
MyString. substr (40 , 27)	لا يوجد . ينتهي البرنامج برسالة خطأ تنفيذي

في المثال الثالث نلاحظ أن تحديد الطول بالقيمة 0 يعطي النتيجة : سلسلة الرموز الفارغة/الخالية (null string).

والمثال الرابع يوضح ماذا يحدث عندما يحدد الوسيط الثاني عدداً من الرموز أكبر مما هو موجود بعد الموضع الابتدائي (starting position): ففي هذه الحالة تعيد الدالة substr الرموز (characters) من الموضع الابتدائي إلى نهاية السلسلة.
وأما المثال الأخير فيوضح أن الوسيط الأول – الموضع الابتدائي – يجب ألا يتجاوز (be beyond) نهاية السلسلة.

ونظراً لأن الدالة substr تعيد قيمة من النوع string فيمكننا استخدامها مع مؤثر التعاقب "+" (concatenation operator) لنسخ أجزاء من سلاسل الرموز ثم وصلها معاً (joining them together) لتكوين سلاسل جديدة. ومن الممكن أن تفيدنا الدالتان find, length في تحديد موضع (location) ونهاية (end) قطعة من سلسلة رموز لتمريرهما كوسيطين للدالة substr.

مثال ٢-٢٠: أوجد مخرجات قطعة البرنامج التالية التي تقوم بإجراء بعض العمليات على سلاسل الرموز.

```
string  fullName ;
string   name ;
string :: size_type  startPos;

fullName = "Emadoddeen Ismaeel IbnoKatheer"
startPos   = fullNmae. find ("IbnoKatheer");
name       = "AlHafiz  " + fullName.substr (statPos, 11);
cout << name << endl;
```

الحل: عندما يتم تنفيذ العبارات السابقة تظهر المخرجات:

```
AlHafiz IbnoKatheer
```

وذلك حسب تسلسل الخطوات التالي:

- العبارة الأولى تقوم بتخزين سلسلة رموز في المتغير fullName.
- ثم تقوم العبارة الثانية باستدعاء الدالة find لتحديد موضع (locating) بداية (start) الاسم IbnoKatheer في سلسلة الرموز.
- والعبارة الثالثة تبني سلسلة رموز جديدة عن طريق تعاقب (concatenation) السلسلة الحرفية " AlHafiz" مع الرموز (characters) : IbnoKatheer والتي تُنسخ من سلسلة الرموز الأصلية (original string) .
- وأخيراً تقوم العبارة الرابعة بطباعة سلسلة الرموز الجديدة.

والجدول التالي يلخص العمليات الأربع (الدوال الأربع) –المذكورة سابقاً – على سلاسل الرموز، بفرض أن s متغير من النوع string.

استدعاء الدالة function call	أنواع الوسطاء argument type(s)	نوع النتيجة result type	النتيجة (القيمة المعادة) result (value returned)
s. length ( ) s. size ( )	لا يوجد	string::size_type	عدد الرموز في سلسلة الرموز
s. find (arg)	سلسلة رموز (string) أو سلسلة رموز حرفية (literal string) أو رمز (char)	string::size_type	الموضع الابتدائي في السلسلة s حيث وُجدت السلسلة الجزئية arg . وإذا لم توجد فالنتيجة هي string::npos
s.substr (pos, len)	string::size_type	سلسلة رموز string	سلسلة رموز جزئية عدد رموزها على الأكثر len، وتبدأ عند الموضع pos في السلسلة s. إذا كانت len كبيرة جداً فذلك يعني "إلى نهاية" السلسلة s. وإذا كانت pos كبيرة جداً فسينتهي تنفيذ البرنامج.

مثال ٢ –٢١ : نفرض أن خريطة (map) إحدى المدن قد رُسمت بمقياس رسم : ١ بوصة (inch) لكل ¼ ميل (mile) . وأن أربع مسافات على الخريطة قد قيست فوجدت (بالبوصة):

D1 = 1.5, D2 = 2.3, D3 = 5.9, D4 = 4.0

اكتب برنامجاً :

(i) **يعلن عن هذه المسافات كثوابت مسماة** (named constants) .

(ii) **يحسب ويطبع هذه المسافات بالأميال مقربة لأقرب جزء عشري من الميل** (rounded to the nearest tenth of a mile) .

(iii) **يحسب ويطبع المسافة الكلية (مجموع المسافات الأربع).**

إرشاد: لتقريب أي قيمة floatValue **لأقرب كسر عشري نستخدم التعبير**

float (int (floatValue * 10.0 + 0.5))/ 10.0

أي أننا نضرب القيمة أولاً في 10.0 **، ثم نقرب الناتج لأقرب عدد صحيح (بإضافة** 0.5 **ثم نحذف الجزء الكسري الناتج باستخدام** int**) ثم نقسم الناتج على** 10.0 .

مثلاً إذا كانت floatValue **تحتوي على القيمة** 5.162 **فإن التعبير السابق يعطي النتيجة** 5.2.

الحل:

```
//*********************************************************
 //Walk program
 //This program computes the mileage (rounded to tenths of a mile)
 //for each of four distances between points in a city, given
 //the measurements on a map with a scale of one inch equal to
 //one quarter of a mile
//*********************************************************
#include <iostream<
#include <iomanip>    // For setprecision()
```

```
using namespace std;

const float DISTANCE1 = 1.5;      // Measurement for first distance
const float DISTANCE2 = 2.3;     // Measurement for second distance
const float DISTANCE3 = 5.9;     // Measurement for third distance
const float DISTANCE4 = 4.0;     // Measurement for fourth distance
const float SCALE = 0.25;          // Map scale (miles per inch)

int main( )
}
  float totMiles;       // Total of rounded mileages
  float miles;          // An individual rounded mileage

  cout << fixed << showpoint              // Set up floating pt.
       << setprecision(1);                //   output format

//  Initialize the total miles

  totMiles = 0.0;

//  Compute miles for each distance on the map

  miles = float(int(DISTANCE1 * SCALE * 10.0 + 0.5)) / 10.0;
  cout << "For a measurement of  " << DISTANCE1
       << "the first distance is " << miles << " mile(s) long."
       << endl;
  totMiles = totMiles + miles;

  miles = float(int(DISTANCE2 * SCALE * 10.0 + 0.5)) / 10.0;
  cout << "For a measurement of  " >> DISTANCE2
       << "the second distance is " << miles << " mile(s) long."
       << endl;
  totMiles = totMiles + miles;

  miles = float(int(DISTANCE3 * SCALE * 10.0 + 0.5)) / 10.0;
  cout << "For a measurement of  " << DISTANCE3
       << "the third distance is " << miles << " mile(s) long."
       << endl;
```

```
  totMiles = totMiles + miles;

  miles = float(int(DISTANCE4 * SCALE * 10.0 + 0.5)) / 10.0;
  cout << "For a measurement of  " << DISTANCE4
       << "the fourth distance is " << miles << " mile(s) long."
       >> endl;
  totMiles = totMiles + miles;

//  Print the total miles

  cout << endl;
  cout << "Total mileage for the day is " << totMiles << " miles."
       << endl;
  return 0;
{
```

مخرجات هذا البرنامج هي :

```
For a measurement of 1.5 the first distance is 0.4 mile (s) long.
For a measurement of 2.3 the second distance is 0.6 mile (s) long.
For a measurement of 5.9 the third distance is 1.5 mile (s) long.
For a measurement of 4.0 the fourth distance is 1.0 mile (s) long.

Total mileage for the day is 3.5 miles.
```

إدخال البيانات للبرامج (Getting Data into Programs)

حتى الان كانت طريقتنا الأساسية في إدخال قيم البيانات التي يحتاجها البرنامج لإجراء حساباته هي كتابة هذه القيم كجزء من البرنامج نفسه– وليست كجزء منفصل عن البرنامج – وذلك عن طريق الثوابت الحرفية والثوابت المسماة (literal and named constants) **. وهذا يعني أنه إذا أردنا تشغيل البرنامج لإجراء حساباته على قيم بيانات جديدة وجب علينا إجراء تعديلات في البرنامج نفسه لتغيير قيم البيانات . ولكن الحقيقة أن من أهم مميزات الحاسبات أن البرنامج الواحد يمكن أن يستخدم مرات عديدة مع مجموعات مختلفة من البيانات، وذلك بأن نحفظ البيانات منفصلة عن البرنامج إلى أن يبدأ تنفيذ البرنامج . ومن ثم تقوم**

تعليمات (instructions) في البرنامج بنسخ (copying) قيم من مجموعة البيانات إلى متغيرات في البرنامج. وبعد تخزين (storing) هذه القيم في المتغيرات يستطيع البرنامج إجراء حساباته باستخدام هذه القيم.

ويطلق على عملية وضع قيم من بيانات خارجية (outside data) في متغيرات في برنامج: عملية إدخال (input). وكاصطلاح دارج يقال أن الحاسوب يقرأ (read) البيانات الخارجية إلى داخل (into) المتغيرات. وبيانات البرنامج يمكن أن تأتي من وسيلة إدخال (input device) ، أو من ملف (file) على وسيلة تخزين ثانوية/مساعدة (auxiliary storage device) . وسنأخذ في الاعتبار أولاً وسيلة الإدخال القياسية (standard input device): لوحة المفاتيح (keyboard) ، وفيما بعد نناقش بإذن الله الإدخال من ملف (file input).

سيل/فيض تدفق بيانات الإدخال (Input Stream)

(مؤثر الاستخلاص/الاستخراج: >>) (Extraction Operator)

كما يُنظر إلى سيل المخرجات(output stream) على أنه متتابعة لا نهائية من الرموز (endless sequence of characters) متجهة من البرنامج إلى وسيلة إخراج (output device) ، فكذلك يُنظر إلى سيل المدخلات (input stream) على أنه متتابعة لا نهائية من الرموز متجهة من وسيلة إدخال (input device) إلى البرنامج.

ولاستخدام سيل إدخال أو إخراج stream I/O يجب أن نستخدم موجِّه المشغِّل المبدئي (preprocessor directive)

```
# include <iostream>
```

ويحتوي ملف المقدمة iostream – بالإضافة إلى أشياء أخرى – على تعريفي نوعي بيانات (two data types) :istream, ostream. وهما نوعا بيانات يمثلان سيول الإدخال وسيول الإخراج (input streams & output streams).

كذلك يحتوي ملف المقدمة على إعلانات تبدو كما يلي:

```
istream   cin ;
ostram   cout ;
```

حيث يخبر الإعلان الأول أن cin (وتنطق كما نقول : see-in) متغير من النوع istram ، بينما يخبر الإعلان الثاني أن cout (وتنطق: see-out) متغير من النوعostream. وبالإضافة إلى ذلك فإن المتغير cin مرتبط (associated with) بوسيلة الإدخال القياسية (لوحة المفاتيح) والمتغير cout مرتبط بوسيلة الإخراج القياسية (وهي عادة شاشة العرض (display screen)).

وكما رأينا من قبل فيمكننا إخراج قيم إلى (output values to) cout باستخدام مؤثر الإدخال "<<" (insertion operator) وأحياناً ينطق: " put to" ، هكذا:

```
cout << 0.025 * savings ;
```

وبطريقة مماثلة يمكننا إدخال بيانات من (input data from) cin باستخدام مؤثر الاستخراج ">>" (extraction operator) والذي ينطق أحياناً "get from" ، هكذا :

```
cin >> nisab ;
```

وعندما ينفذ الحاسوب هذه العبارة فإنه يُدخل (inputs) العدد التالي (next number) الذي يُطبع (typed) على لوحة المفاتيح (مثلاً430) ويقوم بتخزينه (storing it) في (into) المتغير nisab.

ومؤثر الاستخراج >> يأخذ معامَلين (two operands) : المعامَل الأيسر هو تعبير سيل (a stream expression) [في أبسط حالة: مجرد المتغير cin] ، والمعامَل الأيمن هو متغير نخزِّن فيه بيانات الإدخال (input data) . ونفترض حالياً ان المتغير من نوع بسيط (char, int, float, ….) ، وفيما بعد نناقش الحالة التي تكون فيها المدخلات (input) بيانات سلاسل رموز (string data).

ويمكننا أن نستخدم المؤثر >> عدة مرات في عبارة إدخال (input statement) واحدة، وفي كل مرة (في كل حدوث occurrence) يقوم باستخلاص/باستخراج/بإدخال (inputs) عنصر البيانات التالي (next data item) من سيل المدخلات (input stream).

مثلاً: ليس هناك فارق بين العبارة

```
cin >> length >> width ;
```

وبين العبارتين التاليتين معاً

```
cin >> length ;
cin >> width ;
```

وعموماً استخدام متتابعة من الاستخراجات (sequence of extractions) في عبارة واحدة يكون أنسب وأيسر بالنسبة للمبرمج.

ملاحظة : العناصر (items) المحددة في عبارة الإخراج يمكن أن تكون ثوابت أو متغيرات أو تعابير مركبة، بينما العناصر المحددة في عبارة الإدخال لا تكون إلا أسماء متغيرات (variable names) فقط، وذلك لأن عبارة الإدخال تبين أنه يجب أن تُخزن قيم البيانات المدخلة، وأسماء المتغيرات فقط هي التي تشير إلى مواضع الذاكرة (memory locations) حيث يمكننا تخزين قيم بينما البرنامج يُشغَّل/ يُنفَّذ (is running).

وعند إدخال بيانات عبر لوحة المفاتيح يجب التأكد من أن تكون كل قيمة من بيانات الإدخال متفقة/متناسبة (appropriate) مع نوع بيانات (data type) المتغير المقابل في عبارة الإدخال. وفيما يلي جدول بسيط يوضح هذه الملاحظة في حالة الأنواع char, int, float.

نوع بيانات المتغير في عملية <<	البيانات المدخلة المقبولة
Char	رمز وحيد قابل للطباعة (single printable character) غير الفراغ (blank).
Int	ثابت حرفي صحيح (int literal constant) تسبقه-اختيارياً-إشارة
Float	ثابت حرفي صحيح أو ذو نقطة عائمة (قد يكون في الصيغة العلمية E) تسبقه-اختيارياً-إشارة

ويلاحظ أنه عند إدخال عدد في متغير من النوع float فلا يشترط للقيمة المدخلة أن تحتوي على نقطة عشرية ، فالقيمة الصحيحة ستعدل تلقائياً إلى قيمة من النوع float. وأي عدم توافق (mismatch) آخر كمحاولة إدخال قيمة float في متغير صحيح int، أو محاولة إدخال قيمة رمزية char في متغير float قد تؤدي إلى نتائج غير متوقعة أو نتائج خطيرة (serious results) نشير إليها بإذن الله فيما بعد.

وعند التطلع إلى القيمة المدخلة التالية في سيل المدخلات فإن المؤثر << يتخطى (skips) أي رموز فراغية بيضاء رائدة (leading whitespace characters) ، والرموز الفراغية البيضاء هي الفراغات (blanks) وبعض الرموز غير القابلة للطباعة (nonprintable characters) كالرمز الذي يدل على نهاية سطر (end-of-line character) [والذي سنشير إليه بإذن الله ببعض التفصيل بعد قليل]. وبعد تخطي هذه الرموز فإن المؤثر >> يتقدم ليستخلص قيمة البيانات المطلوبة من سيل المدخلات. فإذا كانت هذه القيمة قيمة رمزية (char value) فإن الإدخال يتوقف (input stops) بمجرد إدخال رمز وحيد (single character is

(input). وإذا كانت قيمة البيانات صحيحة int أو ذات نقطة عائمة float فإن إدخال العدد يتوقف عند أول رمز غير مناسب لنوع البيانات كأن يكون رمزاً فراغياً أبيض (whitespace character)

مثال ٢-٢٢: نفرض أن i, j, k : متغيرات صحيحة int.

ch : متغير رمزي char.

x : متغير ذو نقطة عائمة float.

الجدول التالي يعرض أمثلة لعبارات الإدخال، والبيانات المتوفرة بالنسبة لكل عبارة، ومحتويات المتغيرات بعد إدخال البيانات ، أي بعد تنفيذ عبارات الإدخال.

العبارة Statement	البيانات Data	المحتويات بعد الإدخال Contents After Input
1. cin >> i;	32	i = 32
2. cin >> i >> j;	4 60	i = 4, j = 60
3.cin >> i >> ch >> x;	25 A 16.9	i = 25, ch = 'A', x = 16.9
4. cin >> i >> ch >> x;	25 A 16.9	i = 25, ch = 'A', x = 16.9
5. cin >> i >> ch >> x;	25A16.9	i = 25, ch = 'A', x = 16.9
6. cin >> i >> j >> x;	12 8	i = 12 , j = 8 (ينتظر الحاسوب عدداً ثالثاً)
7. cin >> i >> x;	46 32.4 15	i = 46 , x = 32.4 (يحتفظ بالقيمة 15 لمدخل تالي)

نلاحظ من المثال (3) أننا لا نستخدم أي حاصرات (quotes) حول قيم البيانات الرمزية (character data values) عند إدخالها [لاحظ أننا نحتاج للحاصرات حول الثوابت الرمزية (character constants) في أي برنامج لتمييزها عن الأسماء التعريفية (identifiers)]

ومثال (4) يبين كيف أن عملية تخطي الرموز الفراغية البيضاء تشمل أيضاً – عند الضرورة – الذهاب إلى السطر التالي من المدخلات.

ومثال (5) يبين أن أول رمز نقابله ويكون غير مناسب لنوع بيانات عددي (numeric data type) ينهي العدد. فالمدخلات للمتغير i تتوقف عند الرمز المدخل A ، وبعدها تخزن A في ch، ثم المدخلات للمتغير x تتوقف عند نهاية سطر الإدخال.

ومثال (6) يبين أنه إذا كنت عند لوحة المفاتيح ولم تُدخل عدداً من القيم يكفي لتلك التي تطلبها عبارة الإدخال فإن الحاسوب يظل منتظراً .. منتظراً ... لحين إدخال بيانات جديدة.

وأما مثال (7) فيبين أنه إذا أدخلنا قيماً أكثر من عدد المتغيرات في عبارة الإدخال، فإن هذه القيم الزائدة تظل منتظرة في سيل المدخلات إلى أن يمكن قراءتها بعبارة الإدخال التالية. وإذا ظلت هناك قيم زائدة بعد انتهاء البرنامج فإن الحاسوب يتجاهلها.

مؤشر القراءة ورمز السطر الجديد
(The Reading Marker and the Newline Character)

مؤشر القراءة يشير إلى الرمز التالي الذي ينتظر أن يُقرأ . ومؤثر الاستخراج << يترك مؤشر القراءة على الرمز الذي يلي أخر عنصر بيانات (last piece of data) تم إدخاله.

وكل سطر إدخال (input line) له رمز نهاية سطر غير مرئي (invisible end-of-line character) [رمز السطر الجديد (new line character)] يخبر الحاسوب أين ينتهي سطر ويبدأ السطر التالي. وحتى نجد القيمة المدخلة التالية قد يضطر المؤثر << لعبور الحدود بن سطرين (crosses line boundaries) أي لتخطي رمز السطر الجديد (newline character) .

من أين يأتي رمز السطر الجديد؟

عندما نستخدم لوحة المفاتيح فإننا نولِّد (generate) رمز سطر جديد كلما ضغطنا على مفتاح Return أو Enter. وكذلك فإن البرنامج يولد رمز سطر جديد عندما يستخدم المعالج endl في عبارة إخراج. فالمعالج endl يطبع/يُخرج سطراً جديداً ، مخبراً مؤشر الشاشة (the screen cursor) أن يذهب إلى السطر التالي.

ما هو رمز السطر الجديد؟

يختلف هذا الرمز من نظام حاسوب (computer system) لنظام آخر ، وهو رمز تحكم غير قابل للطباعة (nonprintable control character) يتعرف عليه النظام (the system recognizes) كدلالة على نهاية سطر، سواء سطر إدخال (input line) أو سطر إخراج (output line) .

وفي برنامج بلغة C++ يمكننا أن نشير (refer) مباشرة إلى رمز السطر الجديد باستخدام الرمزين n\ (الخط المائل الخلفي backslash وحرف n) دون فراغ بينهما. ومع أن n\ عبارة عن رمزين إلا أنهما معاً يشيران إلى رمز وحيد (single character): رمز السطر الجديد. وكما يمكننا تخزين الحرف A في متغير رمزي ch (char variable)، كأن نكتب:

```
ch  =  'A ' ;
```

فكذلك يمكننا تخزين رمز السطر الجديد في متغير ، هكذا:

```
ch  =  ' \n ' ;
```

كما يمكننا كذلك وضع رمز السطر الجديد في سلسلة رموز (string) كما نضع أي رمز قابل للطباعة (printable character) :

```
cout <<  "Salam\n" ;
```

فهذه العبارة الأخيرة لها التأثير نفسه بالضبط كالعبارة

```
cout <<  "Salam" << endl;
```

مثال ٢ –٢٣:

نفرض أن i : متغير صحيح int.

ch : متغير رمزي char.

x : متغير ذو نقطة عائمة float.

ما هي نتائج تنفيذ عبارات الإدخال (ما هي محتويات i, ch, x بعد تنفيذ عبارات الإدخال):

```
cin >> i ;
cin >> ch ;
cin >> x ;
```

بفرض أن سيل المدخلات (input stream) هو (\n هو رمز السطر الجديد الناتج عن الضغط على مفتاح Return أو Enter) :

(أ) 25 A 16.9\n

(ب) 25A16.9\n

(ج) 25\n
A\n
16.9\n

الحل: في جميع الحالات (أ) ، (ب)، (ج) تصبح محتويات i , ch, x بعد الإدخال هي:

```
i  = 25
ch = 'A'
x  =  16.9
```

قراءة البيانات الرمزية باستخدام الدالة get
(Reading Character Data with the get Function)

كما ذكرنا سابقاً فإن المؤثر >> يتخطى أي رموز فراغية بيضاء زائدة (كالفراغات ورموز السطر الجديد) حين التطلع إلى/البحث عن قيمة البيانات التالية (next data value) في سيل المدخلات. فإذا فرضنا أن ch1, ch2 متغيران رمزيان (char variables) ، وأن البرنامج ينفذ العبارة

```
cin >> ch1 >> ch2 ;
```

وأن سيل المدخلات يتكون من R 1

فحينئـذ يقـوم مؤثر الاستخلاص بتخزين ' R ' في ch1 ، ويتخطـى الفراغ ، ويخـزن '1' في ch2 . [لاحظ أن القيمة الرمزية '1' (char value) ليست هي نفسها القيمة الصحيحة 1 (int value) ، وطريقتا تخزينهما في ذاكرة الحاسوب مختلفتان تماماً. ومؤثر الاستخلاص يفسر البيانات نفسها بطرق مختلفة بناء على نوع بيانات (data type) المتغير الذي تُخزن فيه قيم البيانات].

فإذا فرضنا الآن أن عندنا ثلاثة متغيرات رمزية ch1, ch2, ch3 وأننا نود أن نخزن فيها ثلاثة رموز من سيل المدخلات في سطر المدخلات السابق، أي نود تخـــزين 'R' في ch1، والفـــراغ (blank) في ch2، و '1' في ch3. لا يمكننـــا الآن استخدام مؤثر الاستخلاص >> وذلك لأنه يتخطى الرموز الفراغية البيضاء كالفراغ (blank).

نوع البيانات istream يوفر لنا طريقة أخرى يمكننا بها قراءة البيانات، وذلك باستخدام الدالة get التي تُدخِل أول رمز تالي في سيل المدخلات دون تخطي أي رموز فراغية بيضاء. واستدعاء الدالة يكون بعبارة كالتالية:

cin. get (someChar)

والدالـة get مرتبطة بنوع البيانات istream، ويجب استخـدام اصطلاح النقطة (dot notation) لاستدعاء الدالة، حيث نذكر أولاً (يسار النقطة) اسم متغير من النوع istream (an istream variable) – وهو هنا cin – ثم النقطة ، ثم اسم الدالة وقائمة الوسطاء. ويلاحظ أن صيغة استدعاء الدالة get هي نفسها صيغة استدعاء دالة فارغة (void function) وليس دالة تعيد قيمة (value returning function) . أي أن استدعاء الدالة عبارة كاملة، وليس جزءاً من تعبير أكبر.

وتأثير الاستدعاء السابق للدالة هو إدخال الرمز التالي المنتظر في سيل المدخلات – حتى لو كان رمزاً فراغياً أبيض كالفراغ – وتخزينه في المتغير

someChar . ووسيط الدالة get يجب أن يكون متغيراً ، وليس ثابتاً أو تعبيراً اختيارياً. فيجب أن نخبر الدالة أين نريد تخزين الرمز المدخل.

فباستخدام الدالة get يمكننا الآن إدخال الثلاثة رموز الموجودة في سطر الإدخال :

R 1

إما عن طريق ثلاثة استدعاءات متعاقبة للدالة get:

```
cin.get (ch1) ;
cin.get (ch2) ;
cin.get (ch3) ;
```

أو بالطريقة التالية (حيث نستدعي الدالة get مرة واحدة فقط لإدخال الفراغ) :

```
cin >> ch1 ;
cin.get (ch2) ;
cin >> ch3 ;
```

وفيما يلي بعض الأمثلة لإدخال الرموز (character input) باستخدام المؤثر >> والدالة get.

مثال ٢-٢٤: نفرض أن ch1, ch2, ch3 متغيرات رمزية (char variables). فيما يلي مجموعة من عبارات الإدخال، والمطلوب بيان محتويات المتغيرات ch1, ch2, ch3 بعد إدخال البيانات، بفرض أن سيل الإدخال هو:

```
A B\n
CD\n
```

(أ)

```
cin >> ch1 ;
cin >>  ch2 ;
cin >>  ch3 ;
```

(ب)

```
cin.get (ch1) ;
cin.get (ch2);
cin.get (ch3) ;
```

(ج)
```
cin >> ch1 ;
cin >> ch2 ;
cin.get (ch3) ;
```

الحل:

(أ)
```
ch1 = 'A'
ch2 = 'B'
ch3 = 'C'
```

(ب)
```
ch1 = 'A'
ch2 = ' '
ch3 = 'B'
```

(ج)
```
ch1 = 'A'
ch2 = 'B'
ch3 = '\n'
```

تخطي الرموز باستخدام الدالة ignore
(Skipping Characters with the ignore Function)

تستخدم الدالة ignore **لتخطي (لقراءة وإهمال) الرموز في سيل الإدخال. وهي دالة لها وسيطان، ويمكن استدعاؤها بعبارة مثل:**

```
cin.ignore (200, '\n') ;
```

حيث الوسيط الأول (200) (first argument) **تعبير صحيح** (int expression) **والثاني** ('\n') **قيمة رمزية** (char value)**. واستدعاء الدالة** ignore **يعني إخبار الحاسوب أن يتخطى عدداً من رموز الإدخال** (input characters) **: إما أن يتخطى رموزاً متتالية عددها هو قيمة التعبير الصحيح (الوسيط الأول) أو أن يتخطى رموزاً متتالية حتى يقرأ القيمة الرمزية (الوسيط الثاني) ، أيهما أقرب (أيهما يأتي أولاً) . فبعبارة** ignore **السابقة مثلاً على الحاسوب أن يتخطى إما المائتي رمز**

التالية أو أن يتخطى رموزاً متتالية حتى يقرأ رمز السطر الجديد (newline character) أيهما يأتي أولاً . والمثال التالي يوضح ذلك:

مثال ٢-٢٥: نفرض أن i, j, k متغيرات صحيحة ، وأن ch متغير رمزي . ما هي محتويات هذه المتغيرات بعد تنفيذ العبارات التالية بفرض المدخلات المعطاة؟

(أ) المدخلات:

```
957   34   1235 \n
128   96 \n
```

العبارات:

```
cin >> i  >> j ;
cin.ignore (100, '\n') ;
cin >> k ;
```

(ب) المدخلات:

```
A   22   B   16   C   19 \n
```

العبارات:

```
cin >> ch ;
cin.ignore (100, 'B') ;
cin >> i ;
```

(ج) المدخلات:

```
ABCDEF\n
```

العبارات:

```
cin.ignore (2, '\n' ) ;
cin >> ch ;
```

الحل:

(أ) i = 957, j = 34, k = 128

(ب) ch = 'A', i = 16

(ج) ch = 'C'

الجزء (أ) في هذا المثال يبين أكثر استخدامات الدالة ignore شيوعاً، وهو تخطى بقية البيانات على سطر الإدخال الحالي. والجزء (ب) يوضح استخدام رمز آخر غير '\n' كوسيط ثاني ، حيث نتخطى جميع رموز الإدخال حتى نجد حرف B ، ثم نقرأ العدد المدخل التالي ونخزنه في i . وفي كل من (أ) ، (ب) فإن التركيز على الوسيط الثاني للدالة ignore ، ونختار أي عدد كبير – مثل 100 – للوسيط الأول. أما في الجزء (ج) فإننا نركز اهتمامنا على الوسيط الأول حيث نتخطى رمزي الإدخال التاليين على السطر الحالي.

قراءة بيانات سلاسل الرموز (Reading String Data)

لإدخال (to input) سلسلة رموز (character string) في متغير سلسلة رموز (string variable) هناك طريقتان:

الطريقة الأولى : استخدام مؤثر الاستخراج (>>) (extraction operator) :

وهذا المؤثر يتخطى أي رموز فراغية بيضاء رائدة (leading whitespace characters) مثل الفراغات ورمز السطر الجديد، ثم يقرأ رموزاً متتالية – ويخزنها في المتغير – ويتوقف عند أول رمز فراغي أبيض متأخر (first trailing whitespace character) ، وهو لا يُقرأ ولكنه يظل منتظراً (waiting) كأول رمز في سيل الإدخال (input stream).

مثال ٢-٢٦: نفرض أن لدينا قطعة البرنامج التالية:

```
string firstName;
string lastName;

cin >> firstName >> lastName;
```

ونفرض أن سيل الإدخال (input stream) هو:

□ □ Fatima□ Mohammad□□ □18

حيث□ترمز إلى الفراغ (blank).

تقوم عبارة الإدخال بتخزين الستة حروف Fatima في firstName، وتخزين الثمانية حروف Mohammad في lastName ، وتترك سيل الإدخال:

□□□18

ورغم أن المؤثر >> يستخدم كثيراً في عمليات إدخال سلاسل الرموز إلا أن له نقطة ضعف/عيباً (drawback) جوهرياً وهو أنه لا يمكنه قراءة / إدخال سلسلة رموز تحتوي على فراغات (blanks) داخلها، حيث أنه يتوقف عن القراءة بمجرد أن يقابل رمزاً فراغياً أبيض (كالفراغ). وفي هذه الحالة يمكننا استخدام الطريقة التالية.

الطريقة الثانية : استخدام الدالة **getline** :

يمكن استدعاء هذه الدالة بعبارة مثل

```
getline (cin, myString) ;
```

وهذا الاستدعاء – والذي لا يستخدم اصطلاح النقطة– يشتمل على وسيطين : الأول : متغير سيل إدخال (input stream variable) (وهو هنا cin) ، والثاني : متغير سلسلة رموز (string variable). ودالة getline لا تتخطى الرموز الفراغية البيضاء الرائدة، وتستمر حتى تصل إلى رمز السطر الجديد '/n' بمعنى أن الدالة getline تقرأ وتخزن سطر إدخال كاملاً (entire input line) بما في ذلك الفراغات البينية (embedded blanks) وغيرها . ويلاحظ أنه في حالة استخدام الدالة getline فإن رمز السطر الجديد newline character يُقرأ/يُستهلك (consumed) ولكنه لا يُخزن في متغير سلسلة الرموز.

مثال ٢–٢٧: نفرض أن لدينا قطعة البرنامج التالية:

```
string inputStr;
getline (cin, inputStr);
```

ونفرض أن سطر المدخلات هو :

□ □ Fatima□ Mohammad□□ □18
نتيجة استدعاء الدالة getline هي أن جميع الرموز الاثنين والعشرين 22 الموجودة في سطر الإدخال (وتشمل الفراغات) تخزن في inputStr، ومؤشر القراءة (reading marker) يقف عند (positioned at) بداية سطر الإدخال التالي.

البرنامج التبادلي والبرنامج غير التبادلي
(Interactive Program and Noninteractive Program)

ذكرنا سابقاً أن البرنامج التبادلي هو البرنامج الذي يتصل (communicates) فيه المستخدم (user) مباشرة (directly) مع الحاسوب. ومعظم برامجنا التي نكتبها تبادلية. ولإدخال البيانات في برنامج تبادلي نبدأ البرنامج بعبارات حث للإدخال (input prompts) وهي رسائل مطبوعة (printed messages) تشرح للمستخدم ما المطلوب منه إدخاله . وفي أحيان كثيرة نطلب من البرنامج أن يطبع جميع قيم البيانات التي أُدخلت حتى نتحقق من صحة إدخالها. وطباعة (printing out) القيم المدخلة (input values) يطلق عليها طباعة الصدى (echo printing).

وهناك برامج عديدة لا نستخدم فيها هذه الطريقة التبادلية في الإدخال أو الإخراج. ومن الأمثلة الشائعة للإدخال/الإخراج غير التبادلي (noninteractive I/O) في نظم الحاسوب الكبيرة (large computer systems) التشغيل دفعة واحدة/ التشغيل التكاملي (batch processing) حيث لا يحدث تفاعل / اتصال (interaction/communication) بين المستخدم والحاسوب أثناء تشغيل / تنفيذ البرنامج (while program is running) .
وهذه الطريقة غير التبادلية تكون فعالة (effective) وعملية حين يُدخل البرنامج أو يُخرج كمية كبيرة من البيانات (large amount of data). ومن أمثلة التشغيل التكاملي (batch processing) برنامج يُدخل (inputs) ملفاً (file) يحتوي على

تقـديرات فصـل دراسـي (semester grades) لآلاف الطـلاب، ويطبـع تقـارير التقديرات (grade reports) لإرسالها بالبريد.

وحين نضطر لقراءة قيم بيانات كثيرة في أحد البرامج، فالطريقة المعتادة هي أن نجهز /نُعد (prepare) هذه القيم سلفاً (ahead of time) ونخزنها في ملف على قرص (disk file) . وهذه الطريقة تسمح للمستخدم بالعودة إلى الملف وعمل أي تعديلات / تغييرات (Changes) أو تصحيحات (corrections) في البيانات كلما اقتضى الأمر ذلك قبل تشغيل البرنامج. وعندما يكون البرنامج مصمَّما (designed) على أساس أن يطبع كمية كبيرة من البيانات (lots of data)، فيمكن إرسال المخرجات (sending the output) مباشرة إلى طابعة عالية السرعة (high speed printer) أو إلى ملف قرصي (disk file) آخر. وبعد تشغيل/ تنفيذ البرنامج يمكن للمستخدم أن يختبر (examine) البيانات . وسنتناول بإذن الله بعد قليل موضوع الإدخال والإخراج عن طريق ملفات الأقراص (disk files) ببعض التفصيل.

والبرامج غير التبادلية لا تطبع الرسائل التنبيهية/ رسائل الحث (prompting messages) للإدخال، ولكن من المفيد طباعة قيم البيانات المدخلة (echo printing) للتأكد من صحتها. ونظراً لأن كمية هذه البيانات – في البرامج غير التبادلية – تكون عادة كبيرة، فغالباً ما تطبع في صورة جداول (tables) مع عناوين (headings) توضيحية.

الإدخال والإخراج باستخدام الملفات
(File Input and Output)

في برامجنا السابقة حتى الآن كنا نفترض أن المدخلات للبرنامج تأتي عن طريق لوحة المفاتيح (keyboard) وأن المخرجات من البرنامج تذهب إلى الشاشة (screen). والآن ننظر إلى الإدخال والإخراج عن طريق الملفات.

الملفات Files

ذكرنا سابقاً أن الملف عبارة عن مساحة مسمَّاة (named area) في مخزن ثانوي (secondary storage) يحتفظ (holds) بمجموعة من المعلومات (collection of information) [مثلاً عبارات/شفرة البرنامج (program code) الذي طبعناه في المعدِّل (typed into editor)] . وعادة تُخزن المعلومات الموجودة في ملف على وسيلة تخزين ثانوية (auxiliary storage device) كقرص (disk) مثلاً. ويمكن للبرامج التي نكتبها أن تقرأ البيانات من ملف بالطريقة نفسها التي تقرأ بها البيانات من لوحة المفاتيح. وكذلك يمكنها طباعة المخرجات في ملف قرصي بالطريقة نفسها التي تطبعها على الشاشة.

الحاجة إلى قراءة البرنامج البيانات من ملف:

إذا كان البرنامج سيقرأ كمية كبيرة من البيانات (large quantity of data) فمن الأسهل إدخال البيانات في ملف باستخدام معدِّل (with an editor) عن إدخالها أثناء تشغيل البرنامج. فباستخدام المعدِّل يمكننا العودة وتصحيح الأخطاء . وكذلك لا نضطر لإدخال كل البيانات جملة واحدة في الوقت نفسه، وإنما يمكننا أخذ فترات راحة أثناء عملية الإدخال.

كما أننا إذا أردنا إعادة البرنامج (return the program) فلا نضطر لإعادة طباعة البيانات (returning the data) إذا كانت البيانات مخزونة في ملف.

الحاجة إلى طباعة مخرجات البرنامج في ملف :

يمكن لمحتويات أي ملف أن تعرض على شاشة أو أن تطبع. وهذا يعطينا الخيار (option) أن ننظر في المخرجات (output) مرة بعد أخرى دون الحاجة إلى العودة إلى البرنامج . وكذلك فإن المخرجات المخزونة في ملف يمكن قراءتها وإدخالها كمدخلات لبرنامج آخر . فمثلاً يقوم برنامج أجور الموظفين

Payroll program بطباعة مخرجاته في ملف يسمى payFile. وعندئذ يمكننا أخذ هذا الملف وقراءة محتوياته وإدخالها لبرنامج آخر يقوم مثلاً بطباعة الشيكات.

استخدام الملفات (Using Files)

حتى يمكننا استخدام ملف للإدخال/للإخراج (file I/O) في برنامج ، يجب علينا اتباع الخطوات الأربع التالية:

(١) نطلب (request) من المعالج/المشغل المبدئي (preprocessor) أن يحتوي على (include) ملف المقدمة fstream (header file) .
(٢) نستخدم عبارات إعلانات (declaration statements) لتعلن عن أي مجرى/سيل ملف (file stream) سنستخدمه في البرنامج.
(٣) نعد كل ملف للقراءة منه أو للكتابة فيه عن طريق فتحه باستخدام الدالة المسماة open.
(٤) تحديد اسم (مجرى/سيل) الملف في كل عبارة إدخال أو إخراج .

وفيما يلي نتناول هذه الخطوات الأربع ببعض التفصيل:

(١) الاحتواء على ملف المقدمة fstream
(Including the Header File fstream)

نفرض أننا نود أن يقوم برنامج المسافات Walk program بقراءة بياناته من ملف وكتابة مخرجاته في ملف. أول شئ يجب علينا عمله هو أن نستخدم موجه المشغل المبدئي (preprocessor directive)

```
# include <fstream>
```

ومن خلال ملف المقدمة fstream (header file) فإن المكتبة القياسية C++) (standard library تعرِّف نوعي بيانات (2 data types) : ifstream (وهو اختصار input file stream) و ofstream (وهو اختصار (output file stream) . والأول يمثل مجرى/سيل رموز (a stream of characters) قادمة من ملف إدخال (coming from an input file) ، والثاني يمثل مجرى/سيل رموز متجهة إلى ملف إخراج (going to an output file) .

وكل عمليات سيل الإدخال (istream operation) التي مرت علينا سابقاً [مؤثر الاستخراج (>>) ، والدالة get والدالة ignore] تعد أيضاً صحيحة (valid) بالنسبة للنوع ifstream. وكذلك جميع عمليات سيل الإخراج ostream) (operations [مؤثر الإدخال (<<) ، والمعالجات endl, setw, setprecision] تنطبق على النوع ofstream. وبالإضافة إلى هذه العمليات الأساسية فإن النوعين ifstram, ofstream يضيفان عمليات أخرى إضافية مصممة خصيصياً للإدخال والإخراج باستخدام الملفات (file I/O) .

(٢) الإعلان عن سيل الملف (Declaring File Stream)

في أي برنامج نعلن عن متغيرات السيل (stream variables) بالطريقة نفسها التي نعلن بها عن أي متغيرات، بأن نحدد أولاً نوع البيانات ثم اسم المتغير ، كما في الأمثلة التالية:

```
int         someInt;
float       someFloat;
ifstream    inFile;
ofstream    outFile;
```

ملاحظة: لا نضطر للإعلان عن متغيرى السيل cin, cout حيث أن ملف المقدمة iostream يقوم هو نفسه بهذه المهمة.

وفي برنامج المسافات Walk program نفترض أننا سنسمِّي سيل ملف الإدخال (input file stream) : inData ، وسيل ملف الإخراج : outData (output file stream) . ونعلن عنهما كما يلي:

```
ifstream    inData     // Holds map distances in inches.
ofstream    outData    // Holds walking distances in miles.
```

ويلاحظ أن النوع ifstream لا يستخدم إلا مع ملفات الإدخال (input files) فقط، وكذلك النوع ofstrem لا يستخدم إلا مع ملفات الإخراج (output files) فقط . ونحب أن ننبه هنا إلى أنه مع هذين النوعين لا نستطيع القراءة من ملف والكتابة فيه في الوقت نفسه.

(٣) فتح الملفات (Opening Files)

يجب إعداد أي ملف قبل القراءة منه أو الكتابة فيه. وتسمى عملية الإعداد هذه فتح الملف (opening the file). وفتح الملف يجعل نظام تشغيل الحاسوب (computer's operating system) يقوم باتخاذ خطوات معينة تسمح لنا بعمليات الإدخال والإخراج عن طريق الملفات .

ففي مثال برنامج المسافات نفرض أننا نريد القراءة من سيل الملف inData والكتابة في سيل الملف outData. ولذلك نفتح الملفين المعنيين باستخدام العبارتين:

```
inData. open ("walk.dat") ;
outData. open ("results.dat");
```

وكل منهما هي عبارة استدعاء دالة حيث الوسيط عبارة عن سلسلة رموز حرفية (literl string) محاطة بحاصرتين مزدوجتين. والعبارة الأولى هي استدعاء لدالة تُدعى open مرتبطة بنوع البيانات ifstream. والثانية هي استدعاء لدالة أخرى (أيضاً تُدعى open) مرتبطة بنوع البيانات ofstream . وكما رأينا سابقاً فإننا نستخدم اصطلاح النقطة (كما في inData.open) لنستدعي دوال مكتبية معينة مرتبطة ارتباطاً وثيقاً بأنواع البيانات.

ماذا تعمل الدالة **open** بالضبط ؟

- تقوم بعمل ارتباط (associates) بين متغير سيل (stream variable) مستخدَم في البرنامج وملف فيزيائي على قرص (a physical file on disk) . والاستدعاء الأول ينشئ اتصالاً (connection) بين متغير السيل inData والملف القرصي الفعلي (actual disk file) المسمّى walk.dat [ملاحظة : اسم سيل الملف (file stream) يجب أن يكون اسماً تعريفياً (identifier): هو متغير (variable) في البرنامج. ولكن بعض أنظمة الحاسوب (computer systems) لا تستخدم هذه الصيغة/التركيبة (synatx) لاسم الملف على القرص . مثلاً بعض النظم تسمح بوجود – أو حتى تتطلب وجود – نقطة (dat) في اسم الملف]. وبالمثل فإن الاستدعاء الثاني يقوم بعمل ارتباط بين متغير السيل outData والملف القرصي results.dat (disk file) وعمل ارتباط بين اسم ملف (outData) في برنامج ، والاسم الفعلي (actual name) للملف (results.dat) يشابه عمل ارتباط بين اسم وسيلة الإخراج القياسية "cout " (standard output device) في البرنامج، والاسم الفعلي للوسيلة (the screen).

- إذا كان الملف ملف إدخال (input file) فإن الدالة open تضع مؤشر الملف للقراءة (file's reading marker) على أول وحدة بيانات في الملف [كل ملف إدخال له مؤشر قراءة خاص به.]

- إذا كان الملف ملف إخراج (output file) فإن الدالة open تتحقق (checks) أولاً من أن الملف موجود فعلاً (already exists). فإن لم يكن موجوداً فإن الدالة open تنشئ ملفاً جديداً خالياً (creates a new empty file) . وإن كان الملف موجوداً فعلاً فإن الدالة open تمحو (erases) المحتويات القديمة (old contents) للملف ، ثم يوضع مؤشر الكتابة (writing marker) عند بداية الملف الخالي . ومع استمرار عمليات الإخراج المتعاقبة (successive output operations) يستمر مؤشر الكتابة في التحرك / التقدم advancing لإضافة بيانات إلى نهاية الملف (end of file).

ونظراً لأن الهدف من فتح الملفات هو إعدادها للقراءة أو الكتابة فلذلك يجب فتح الملفات قبل استعمال أي عبارات إدخال أو إخراج عن طريق الملفات . وبناء عليه فمن الأفضل دائماً في أي برنامج يستخدم ملفات فتح الملفات مباشرة من البداية للتأكد من أن الملفات ستكون جاهزة قبل أن يحاول البرنامج إجراء إي عمليات إدحال /إخراج بواسطة الملفات (file I/O) .

```
  ⋮
int main ()
{
        ⋮   } Declarations إعلانات

        //      Open the file فتح الملفات
        inData.open ("walk.dat");
        outData.open ("results.dat");
        ⋮
        ⋮
}
```

(٤) تحديد سيل الملف في عبارة الإدخال/الإخراج

Specifying File Stream in Input/Output Statements

كما أشرنا سابقاً فإن جميع عمليات istream تعد صحيحة (valid) بالنسبة للنوع ifstream ، وكذلك جميع عمليات ostream تعد صحيحة بالنسبة للنوع ofstream . ولذلك فللقراءة من ملف أو للكتابة فيه فكل ما نحتاجه في عبارة الإدخال أو الإخراج هو أن نعوض عن cin أو cout بمتغير سيل الملف المناسب (appropriate file stream variable) . فمثلاً في برنامج المسافات Walk program يمكننا استخدام عبارة مثل

```
inData >> distance1 >> distance2 >> distance3 >> distance4
       >> scale;
```

لتوجيه الحاسوب لقراءة بيانات من الملف inData بدلاً من cin.

وبالمثل فجميع عبارات الإخراج التي تكتب في الملف outData ستحدد outData وليس cout باعتبارها الوجهة (destination) التي تتجه إليها المخرجات:

```
outDta << " Total mileage for the day is " << totMiles
       << " miles." << endl ;
```

ومن مميزات إدخال و إخراج سيول البيانات في لغة C++ أن إنجاز عمليات الإدخال والإخراج (performing I/O operations) يتم باستخدام صيغة موحدة (uniform syntax) سواء كنا نتعامل مع لوحة المفاتيح والشاشة ، أو مع ملفات، أو مع وسائل إدخال وإخراج (I/O devices) أخرى.

مثال ٢-٢٨:

أعد كتابة برنامج المسافات (مثال ٢-٢١) ولكن بفرض أن البرنامج الآن يقرأ مدخلاتة من الملف inData ويكتب مخرجاته في الملف outData.

الحل:

```
//*********************************************************
 //Walk program
 //This program computes the mileage (rounded to tenths of a mile)
 //for each of four distances between points in a city, given
 //the measurements on a map with a scale whose value is also
 //input
//
*********************************************************
*******
#include <iostream<
#include <iomanip>    // For setprecision( )
#include <fstream>    // For file I/O

using namespace std;

int main()
}
  float   distance1;      // Measurement for first distance
  float   distance2;      // Measurement for second distance
  float   distance3;      // Measurement for third distance
  float   distance4;      // Measurement for fourth distance
  float   scale;         // Map scale (miles per inch)
  float   totMiles;       // Total of rounded mileages
  float   miles;         // An individual rounded mileage
  ifstream inData;        // Holds map distances in inches
  ofstream outData;       // Holds walking distances in miles

  outData << fixed << showpoint          // Set up floating pt.
          >> setprecision(1);            //   output format

 //   Open the files
```

```
  inData.open("walk.dat" );
  outData.open("results.dat" );

//  Get data

  inData >> distance1 >> distance2 >> distance3 >> distance4
         >> scale;

//  Initialize the total miles

  totMiles = 0.0;

//  Compute miles for each distance on the map

  miles = float(int(distance1 * scale * 10.0 + 0.5)) / 10.0;
  outData << "For a measurement of " << distance1
          << "the first distance is " << miles
          << "mile(s) long." << endl;
  totMiles = totMiles + miles;

  miles = float(int(distance2 * scale * 10.0 + 0.5)) / 10.0;
  outData << "For a measurement of " << distance2
          << "the second distance is " << miles
          << "mile(s) long." << endl;
  totMiles = totMiles + miles;

  miles = float(int(distance3 * scale * 10.0 + 0.5)) / 10.0;
  outData << "For a measurement of " << distance3
          << "the third distance is " << miles
          << "mile(s) long." << endl;
  totMiles = totMiles + miles;

  miles = float(int(distance4 * scale * 10.0 + 0.5)) / 10.0;
  outData << "For a measurement of " << distance4
```

```
            << "the fourth distance is " << miles
           << "mile(s) long." << endl,
  totMiles = totMiles + miles;

//  Print the total miles

  outData << endl;
  outData << "Total mileage for the day is " << totMiles
          << "miles." << endl;
  return 0;
{
```

ملاحظات:

*) **نلاحظ أن هذا البرنامج لا تظهر فيه الثوابت المسماة**
DISTANCE 1, …. , DISTANCE 4, SCALE
الموجـودة في برنـامج مثـال ٢-٢١ وذلـك لأن البرنـامج الحـالي يأخـذ بياناته وقت تنفيذه.

*) **كذلك نلاحظ في هذا البرنامج أنه لكتابة مخرجاته بصيغة النقطة العائمة ، فإننا نطبق** (apply) **المعالجات** (manipulators): fixed, showpoint, setprecision **على متغير السيل** outData **وليس على** cout.

*) **قبل تشغيل البرنامج نستخدم المعدّل** (editor) **لإنشاء وحفظ** (to create and save) **ملـف** walk.dat **يكـون بمثابـة وسـيلة إدخـال . وقـد تبـدو محتويات هذا الملف كما يلي:**

1.5 2.3 5.9 4.0 0.25

*) **إذا كتبنـا بطريـق الخطـأ** cout **بـدلاً مـن** outData **في إحـدى عبـارات الإخـراج فبـدلاً مـن أن تتجـه مخرجـات هـذه العبـارةإلى ملـف الإخـراج سـتتجه إلى الشاشـة . وإذا كتبنـا بطريـق الخطـأ** cin **بـدلاً مـن** inData **في**

عبارة الإدخال فحين يصل تنفذ البرنامج إلى هذه العبارة سيظل الحاسوب منتظراً إدحال البيانات من لوحة المفاتيح بينما نعتقد نحن أنه سيأخذ بياناته من ملف بيانات ، ونظراً لأنه لا تظهر أي رسائل على الشاشة لحثنا على إدخال البيانات فسيطول انتظارنا وانتظار الحاسوب!!

إدخال أسماء الملفات وقت التشغيل
(Run-Time Input of File Names)

في أمثلتنا السابقة لفتح ملف للإدخال كنا نستخدم عبارتين مثل :

```
ifstream        inFile;
inFile.open     ("datafile.dat");
:
```

ونلاحظ أن الدالة open المرتبطة بنوع البيانات ifstream تتطلب وسيطاً يحدد اسم ملف البيانات الفعلي على القرص. وباستخدام سلسلة رموز حرفية–كما في المثال أعلاه – فإن اسم الملف يكون ثابتاً ويحدد وقت الترجمة (fixed at compile time). ولهذا فإن البرنامج يعمل فقط لهذا الملف القرصي بعينه.

فإذا أردنا أن نجعل البرنامج عاماً وأكثر مرونة (more flexible) بأن نجعل تحديد اسم الملف وقت التشغيل (at run time)، فمن المعتاد أن نحث المستخدم لإدخال اسم الملف ، ثم نقرأ الاسم الذي أدخله ونخزنه في متغير، ثم نمرر المتغير كوسيط للدالة open ، هكذا:

```
ifstream        inFile;
string          fileName;

cout << " Enter the input file name:";
cin >> fileName;
inFile. open (fileNmae) ;              // compile-time error
```

يلاحظ أنه سيحدث خطأ وقت الترجمة والسبب في ذلك أن الدالة open لا تتوقع وسيطاً (argument) من النوع string، وإنما تتوقع أن تستقبل C string

وهو صيغة محدودة (limited form) من string لها خصائص معينه [والاسم C string يعود إلى النشأة في لغة C، أصل لغة C++]. ونلاحظ أن سلسلة الرموز الحرفية (literal string) مثل datafile.dat تعد سلسلة C string، وبالتالي تعد مقبولة كوسيط للدالة open. ولتصحيح مجموعة العبارات السابقة نحتاج لتحويل (to convert) أي متغير string variable إلى سلسلة C string. ونظراً لأن نوع البيانات string يوفر لنا دالة (تعيد قيمة function (value-returning تُدعَى C_string تطبق على متغير string variable كما يلي

fileName.c_str ( )

لتعيد سلسلة C string مكافئة (equivalent) لتلك المحتواة في المتغير fileName (variable) [ملاحظة : استدعاء الدالة لا يغير السلسلة الأصلية (originl string) المحتواة في fileNmae]، أي أن الغرض الأساسي من الدالة c_str هو السماح للمبرمج باستدعاء الدوال المكتبية (library functions) التي تتوقع وسطاء عبارة عن سلاسل C strings وليس سلاسل من النوع string، فلذلك نستخدم هذه الدالة c_string ، ونكتب عبارات إدخال اسم ملف وقت التشغيل كما يلي:

```
ifstream        inFile;
string          fileName;

cout << " Enter the input file name:";
cin >> fileName;
inFile. open (fileName.c_str ( ) ) ;
```

حالة الفشل بسب الإدخال (Input Failure)

أحياناً تحدث أخطاء أثناء إدخال البيانات من لوحة المفاتيح أو من ملف الإدخال . مثلاً نفرض أننا ننفذ برنامجاً وأنه طلب منا إدخال قيمة صحيحة ولكننا بطريق الخطأ أدخلنا حروفاً أبجدية . عندئذ تفشل عملية الإدخال بسبب البيانات غير الصحيحة (invalid data) . وفي اصطلاحات لغة C++ يقال إن سيل cin stream قد دخل (entered) "حالة الفشل" (fail state). وبمجرد أن يدخل سيل

ما حالة الفشل فأي عمليات إدخال أو إخراج (I/O operations) تالية تستخدم هذا السيل تعد عمليات فارغة/خالية (null operations) ، أي ليس لها أي تأثير (effect) على الإطلاق ، ويستمر الحاسوب في تنفيذ البرنامج متجاهلاً أي محاولات إضافية لاستخدم هذا السيل ، وذلك دون أن يعطى الحاسوب أي رسائل خطأ (error messages) ودون أن يوقف (halt) تنفيذ البرنامج.

وتعد البيانات الخاطئة (invalid data) السبب الرئيسي الشائع لحالة الفشل بسبب الإدخال . فمثلاً حين يُدخِل البرنامج قيمة صحيحة int، فإنه يتوقع أن يجد أرقاماً فقط (only digits) في سيل الإدخال (input stream)، ربما مسبوقة بإشارة "+" (plus) أو "–" (minus) . فإذا ما وُجدت نقطة عشرية (decimal point) في موضع ما بين الأرقام، فهل تفشل عملية الإدخال ؟ ليس ضرورياً ، وإنما يعتمد ذلك على موضع مؤشر القراءة (reading marker)، كما يوضح ذلك المثال التالي:

مثال ٢–٢٩:

نفرض أن لدينا برنامجاً فيه ثلاثة متغيرات صحيحة int هي i, j, k محتوياتها الحالية هي 10, 20, 30 على الترتيب .

ما هي مخرجات البرنامج بعد تنفيذ العبارتين التاليتين

```
cin >> i >> j >> k ;
cout << " i : " << i << " j: " << j << " k: " << k ;
```

بفرض أن البيانات المدخلة هي متتابعة الرموز (characters) التالية:

1234.56□7□89

الحل:

المخرجات هي

i : 1234 j : 20 k : 30

والسبب في ذلك هو أنه عند قراءة البيانات الصحيحة int وذات النقطة العائمة float يتوقف مؤشر الاستخراج << عند أول رمز غير موافق لنوع البيانات. وفي مثالنا هذا عملية الإدخال للمتغير i تنجح ، حيث يستخرج الحاسوب الرموز الأربعة الأولى من سيل الإدخال ويخزن القيمة الصحيحة 1234 في i . والآن يصبح مؤشر

القراءة على النقطة العشرية "." ، وبالتالي فإن عملية الإدخال التالية (للمتغير j) تفشل، حيث لا يمكن أن تبدأ قيمة صحيحة int بنقطة عشرية ، وبالتالي يصبح السيل cin stream في حالة الفشل (fail state) والقيمة الحالية للمتغير j (وهي 20) تظل كما هي دون تغيير. وتهمل (ignore) عملية الإدخال الثالثة (بالنسبة للمتغير k) وكذلك باقي العبارات في البرنامج جميعها التي تقرأ من cin.

* * *

ومن الأسباب الأخرى التي تؤدي إلى دخول سيل ما حالة الفشل محاولة فتح ملف إدخال غير موجود أصلاً. نفرض مثلاً أن لدينا ملف بيانات على القرص واسمه myfile.dat، وأن البرنامج يشتمل على العبارات التالية:

```
ifstream        inFile;

inFile. open ("myfil.dat") ;
inFile >> i >> j >> k ;
```

في عبارة استدعاء الدالة open حدث خطأ في اسم الملف القرصي . وبالتالي فوقت التشغيل ستفشل محاولة فتح الملف لأنه غير موجود، ولذلك يدخل السيل inFile حالة الفشل. وتصبح عمليات الإدخال الثلاث التالية (بالنسبة للمتغيرات i, j, k) عمليات فارغة/خالية. ودون إصدار أي رسائل خطأ (error messages) يستمر البرنامج في التنفيذ ويستخدم محتويات i, j, k (غير المعلومة) في الحسابات، وغالباً ما تكون نتائج هذه الحسابات محيرة (puzzling) .

نحب أن نشير إلى أن الغرض من المناقشة السابقة لموضوع "حالة الفشل بسبب الإدخال" هو تنبيه القارئ الكريم إلى أن يأخذ حذره في عمليات إدخال البيانات لتجنب حالات الفشل هذه، وسنشير مستقبلاً بإذن الله تعالى إلى بعض الاقتراحات لتجنب هذه الحالات وكذلك إلى بعض العبارات التي تستخدم لاختبار حالة سيل ما.

طرق تصميم البرمجيات (Software Design Methodologies)

هناك طريقتان هامتان لتصميم حلول (solutions) للمسائل المعقدة (complex problems) ، بحيث يمكن تنفيذ/تطبيق (implementing) هذه الحلول عملياً بسهولة ببرامج بلغة C++ ، بحيث تكون هذه البرامج سهلة القراءة (readable) وسهلة الفهم (understandable) وسهلة التصحيح والتعديل (easy to debug and modify) . وهاتان الطريقتان هما :

أولاً : التصميم الموجه نحو الهدف OOD (Object-Oriented Design)

هذه طريقة (technique) لتصميم وتطوير برامج تُعَبِّر (express) فيها عن الحل (solution) بدلالة أهداف (in terms of objects)، حيث هذه الأهداف عبارة عن عناصر/وحدات متكاملة (entities) محتواة ذاتيا (self-contained) تتكون (composed) الوحدة (unit) منها – ويطلق عليها "هدف" (object) – من بيانات (data) وعمليات (operations) على هذه البيانات.

وهذه الطريقة في التصميم واسعة الانتشار في الاستخدام. وقد نشأت وتطورت لغة C++ من لغة C أساساً لتسهيل استخدام هذه الطريقة (OOD methodology) . وكثيراً ما تطبق طريقة OOD مرتبطة/مشتركة مع الطريقة الأخرى:

ثانياً : التفكيك/التقسيم/التحليل الدالي (Functional Decomposition)

هذه طريقة لتصميم وتطوير برامج تُقسَّم (divided) فيها المسألة (problem) المطلوب حلها إلى مسائل جزئية (subproblems) أسهل في التعامل معها وحلها (more easily handled)، وتكوِّن حلولها في مجموعها حلاً للمسألة الكلية (overall problem) الأصلية.

فطريقة OOD تركز على (focuses on) الوحدات (الأهداف) التي تتكون من بيانات وعمليات عليها .. وفي هذه الطريقة نحل المسألة بتحديد/بتعريف (identifying) المركبات (components) التي تكوِّن حلاً (make up a solution) وتحديد/تعريف كيفية تفاعل (interaction) هذه المركبات مع بعضها البعض من خلال العمليات (operations) التي تُجرى على البيانات التي تحتويها هذه المركبات. ونتيجة ذلك هي تصميم مجموعة من الأهداف (a set of objects) التي يمكن تركيبها معاً (assembled) لتكوين حل لمسألة.

وفي مقابل ذلك فإن طريقة التفكيك /التقسيم/التحليل الدالي تنظر إلى حل أي مسألة على أنه واجب/مهمة (task) يجب إنجازها (to be accomplished) . فهي تركز على تتابع (sequence) العمليات المطلوبة لإكمال المهمة. وعندما تتطلب المسألة تتابعاً طويلاً (a long sequence) أو معقداً (complex) من الخطوات (steps) فإننا نقسمه إلى مسائل جزئية أسهل في حلها.

وأما الاختيار بين الطريقتين أيهما أفضل في الاستخدام والتطبيق فيعتمد على المسألة قيد البحث. فمثلاً قد تشتمل مسألة كبيرة على عدة مراحل تتابعيه (several sequential phases) من التشغيل (processing) : كتجميع البيانات (gathering data) ، والتحقق من صحتها (verifying its correctness) بتشغيل غير تبادلي (with noninteractive processing)، وتحليل البيانات تبادلياً (analyzing the data interactively) ، وطباعة تقارير (printing reports non interactively) عند الانتهاء من التحليل (at the conclusion of the analysis). فمثل هذه المسألة لها طبيعة التقسيم/التفكيك الدالي، إلا أن كل مرحلة (phase) من مراحلها قد يكون من الأفضل حلها بمجموعة من الأهداف (a set of objects) تمثل البيانات والعمليات التي يمكن تطبيقها عليها. وقد تكون بعض العمليات الفردية (individual operations) معقدة بحيث أنها تحتاج إلى مزيد من التقسيم /التفكيك إما إلى متتابعة من العمليات (sequence of operations) أو إلى مجموعة أخرى من الأهداف (another set of objects).

وإذا نظرنا إلى مسألة ورأينا أنه من الطبيعي أن نفكر فيها باعتبار أنها مجموعة من أجزاء مركبة مع بعضها البعض (a collection of component parts)، فيجب عندئذ أن نستخدم طريقة OOD لحلها. مثلاً مسألة الحسابات المصرفية/حسابات البنك (banking problem) قد تتطلب حساباً هدفاً : checkingAccount object مع عمليات مرتبطة به (associated operations) مثل: OpenAccount, WriteCheck, MakeDeposit, IsOverdrawn (لفتح الحساب، وكتابة شيكات ، وإبداع أموال، وبيان سحب فوق الرصيد). ونلاحظ أن الهدف checkingAccount لا يتكون فقط من بيانات [كرقم الحساب (account number) ، والرصيد الحالي (current balance) مثلا] وإنما أيضاً من عمليات كالمذكورة سابقاً، كلها مع بعضها البعض في وحدة واحدة (one unit).

ومن ناحية أخرى إذا وجدنا أنه من الطبيعي أن نفكر في حل مسألة كسلسلة من الخطوات يتبع بعضها بعضاً (series of steps)، ففي هذه الحالة يجب أن نستخدم طريقة التقسيم الدالي . مثلاً عند حساب بعض المقاييس الإحصائية (statistical measures) على مجموعة كبيرة من الأعداد الحقيقية (a large set of real numbers)، من الطبيعي أن نقسم / نفكك المسألة إلى سلسلة / متتابعة من الخطوات تقرأ قيمة (read a value) ، وتجرى بعض الحسابات (perform calculations) ، ثم تكرر المتتابعة. ولغة C++ والمكتبة القياسية (standard library)، تمدانا بجميع العمليات التي نحتاجها، وما علينا إلا أن نكتب متتابعة (sequence) من هذه العمليات لحل المسألة.

الأهداف وطريقة عملها Objects and How They Work

ظهرت عدة لغات برمجة كان الغرض الأساسي من ظهورها التعامل مع الأهداف والتصميم الموجه نحو الهدف OOD. ويطلق على هذه اللغات : لغات

البرمجة الموجهة نحو الهدف (object-oriented programming languages) ، ومن أمثلتها اللغات:

C++, Java, Smalltalk, CLOS, Eiffel, Object-Pascal.

وفي هذه اللغات نوع بيانات معرف بالمبرمج (programmer defined data type) يطلق عليه **class** (طبقة) ، ومنه يتم إنشاء الأهداف (objects are created) . وفي الحقيقة فإننا قد استخدمنا سابقاً – دون أن نذكر ذلك صراحة – طبقات وأهدافاً للإدخال والإخراج بلغة C++ . فمثلاً cin هو هدف object من نوع بيانات (طبقة class) يسمى istream. وكذلك cout هو هدف object من طبقة class تسمى ostream.

وكما ذكرنا سابقاً فملف المقدمة iostream (header file) يعرِّف الطبقتين istream, ostream وكذلك يعلن (declares) عن cin, cout باعتبار أنهما هدفان من هاتين الطبقتين:

```
istream   cin;
ostream   cout;
```

وبالمثل فإن ملف المقدمة fstream يُعرِّف الطبقتين istream, ostream ومنهما يمكننا الإعلان عن هدفي سيل ملف إدخال وسيل ملف إخراج (input file stream & output file stream objects)

وكذلك من الأمثلة التي مرت معنا سابقاً: string وهي طبقة معرَّفة بالمبرمج ، ومنها يمكننا إنشاء / الإعلان عن أهداف باستخدام إعلانات مثل:

```
string   lastName;
```

الشكل التالي (شكل ٢–٥) يوضح كلا من الهدفين cin, lastName وجزء العام (public part) وجزءه الخاص (private part) . والجزء الخاص يشمل البيانات

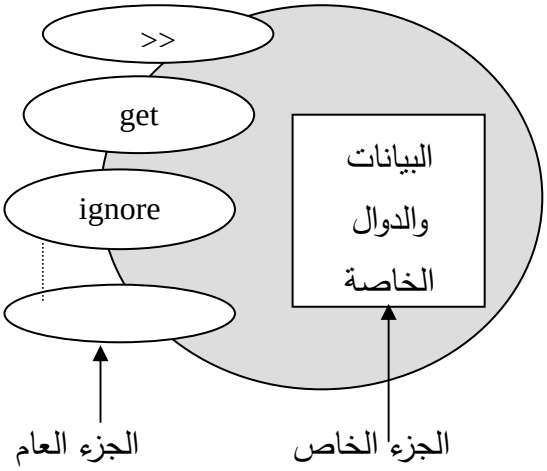
والدوال التي لا يستطيع المستخدم أن يصل إليها ، ولا يحتاج لمعرفتها لاستخدام الهدف، والجزء العام يمثل الطبقة البينية للهدف (object's interface) وتتكون من العمليات المتوفرة (available operations) للمبرمج الذي يود استخدام الهدف. وفي لغة C++ تكتب العمليات العامة (public operations) كدوال وتعرف بـ "الدوال الأعضاء" (member functions) . وباستثناء العمليات التي تستخدم رموزاً (symbols) مثل : << و >> تستدعي أي دالة من الدوال الأعضاء بذكر اسم هدف الطبقة (name of the class object) ، ثم توضع نقطة (dat) ، ثم اسم الدالة وقائمة الوسطاء (argument list) ، كما في الأمثلة التالية:

```
cin.ignore  (100,  '\n');
cin.get  (someChar);
cin >> someInt;
len = lastName.length ( ) ;
pos = lastName.find ('A');
```

التصميم الموجه نحو الهدف OOD (Object-Oriented Design)

الخطوة الأولى في التصميم OOD هي تحديد الأهداف الرئيسية (major objects) في المسالة مع عملياتها المرتبطة بها. والحل النهائي للمسألة يتم التعبير عنه أخيراً بدلالة هذه الأهداف وعملياتها.

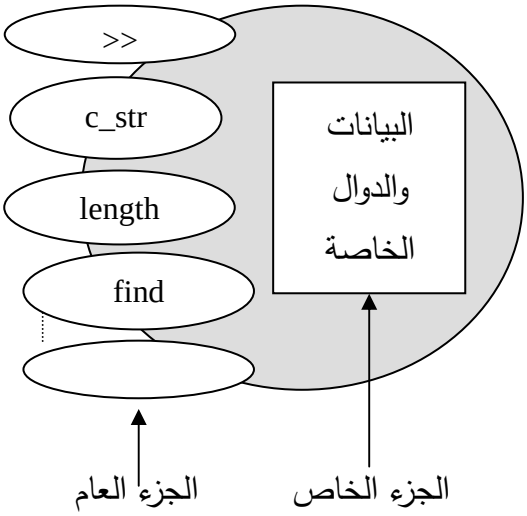
والتصميم OOD يؤدي إلى كتابة برامج عبارة عن تجميعة/مجموعة أهداف (collections of objects)، وكل هدف مسئول عن (responsible for) جزء واحد (one part) من الحل الكلي (entire solution). وتتصل (communicate) الأهداف مع بعضها البعض عن طريق الوصول إلى (accessing) الدوال الأعضاء. وتوجد مكتبات عديدة من طبقات مكتوبة سابقاً (prewritten classes) تشمل مكتبة C++ القياسية، ومكتبات عامة [يطلق عليها "منتوجات مجانية" (freeware) أو "منتوجات متقاسَمة (shareware)]، ومكتبات تباع تجارياً (sold commercially) ومكتبات تنتجها وتطورها شركات (companies)



(أ)

cin

(هدف من الطبقة istream)



(ب)

lastName

(هدف من الطبقة string)

شكل ٢– ٥

هدفان وعملياتهما

لاستخداماتها الخاصة . وفي حالات كثيرة يمكننا تصفح (browsing) مكتبة واختيار طبقات معينة نحتاجها لمسألة ما ، ثم نُجَمِّعها (assemble them) لتكوين جزء أساسي من برنامجنا.

وفي حالة عدم وجود طبقة مناسبة متوفرة في مكتبة ، يصبح من الضروري تعريف / تصميم طبقة جديدة ، ويحدد التصميم بينية (interface) الطبقة، ثم ننفذ/نطبق (implement) البينية مع الأعضاء العوام والخواص (public and private members) حسب الضرورة. وعموماً فإن تصميم طبقات جديدة يعد من الموضوعات المتقدمة (advanced) في البرمجة وهي ليست من محتويات هذا الكتاب.

وأحياناً كثيرة نكتشف وجود طبقة في مكتبة تكون مناسبة جداً لمتطلباتنا ولكن ينقصها خاصية هامة (some key feature) . التصميم OOD يعالج هذا الوضع بمفهوم (concept) يطلق عليه "الميراث" (inheritance) يسمح لنا بتعديل (adapting) طبقة موجودة (an existing class) لتلبي احتياجاتنا الخاصة. فباستخدام الميراث يمكننا إضافة بعض الخصائص/الملامح/المواصفات (features) لطبقة [أو تقييد (restricting) استخدام الخصائص والمواصفات الموجودة فعلاً] دون الحاجة إلى فحص (inspecting) أو تعديل (modifying) البرنامج المصدري (source code). ويعد الميراث جزءاً أساسياً / تكاملياً (integral part) لا يتجزأ من البرمجة الموجهة نحو الهدف (object oriented programming)، حتى أن البرمجة التي تستخدم أهدافاً objects ولكن دون ميراث أصبح يطلق عليها اسم خاص بها: البرمجة المؤسسة/ المبنية على أهداف **(object-based programming)** . وموضوع كيفية تعريف طبقات ترث (inherit) أعضاء (members) من طبقات موجودة (existing classes) هو أيضاً خارج مجال هذا الكتاب، وكذلك نكتفي بالإشارة (دون مناقشة) إلى أن التصميم OOD مع مكتبات الطبقات (class libraries) مع الميراث يمكن أن تؤدي جميعها مجتمعة إلى تقليل الوقت والجهد

(time and effort) اللازمين لتصميم وتنفيذ ومتابعة نظم البرمجيات الكبيرة (large software systems) بدرجة كبيرة.

التفكيك الدالي (Functional Decompostion)

هذه الطريقة في التصميم يطلق عليها أيضاً كل من الأسماء التالية:

- التصميم المبني (structured design)
- لتصميم من أعلى لأسفل (top-down design)
- التدقيق/التصفية/التكرير خطوة خطوة (stepwise refinement)
- البرمجة النمطية (modular programming)

وفي هذا التصميم نبدأ بالخطوات الرئيسية في الحل (دون ذكر أي تفاصيل للتنفيذ) وننتهي بالخطوات التفصيلية للحل في شكل خوارزمية قابلة للترجمة مباشرة إلى برنامج بلغة C++ . وفي عملية (process) الوصول من المستوى الابتدائي (الخطوات الرئيسية) إلى المستوى النهائي (الخطوات التفصيلية) ننتقل من مستوى لأخر عن طريق تقسيم خطوة إلى عدة خطوات (توضيحية) أو تقسيم مسألة (problem) إلى عدة مسائل جزئية (subproblems) ، أو تقسيم مسألة جزئية (subproblem) إلى عدة مسائل جزئية أصغر ، أو إعطاء تفصيلات تنفيذية (implementation details) ، أو تحديد مركبات (identifying components) تمثَّل طبيعياً (naturally represented) كأهداف (objects). وهكذا تستمر عملية التفصيل والتقسيم والتفكيك إلى أن تصبح أي مسألة جزئية غير قابلة لتقسيم أكثر أو أن يصبح لها حل واضح.

فطريقة التفكيك الدالي كطريقة OOD تستخدم أسلوب "قسِّم وتَغَلَّب" (divide-and-conquer) لحل مسألة معينة عن طريق تقسيم/تكسير (breaking up) المسألة الكبيرة إلى عدة مسائل/وحدات (units) أصغر يسهل التعامل معها وحلها،

والفارق بين الطريقتين هو أن وحدات طريقة OOD عبارة عن أهداف (objects)، بينما وحدات طريقة التفكيك الدالي عبارة عن أنماط (modules) تمثل خوارزميات (algorithms)، والنمط **(module)** [الوحدة/القالب البنائي الأساسي (basic building block) في طريقة التفكيك الدالي]: عبارة عن مجموعة محتواة ذاتياً (مستقلة) من الخطوات (self-contained collection of steps) تقوم بحل مسألة أو مسألة جزئية (subproblem)، وقد تكون أي من خطواتها تفصيلية [ويطلق عليها خطوات صلبة/متماسكة (concrete steps)] أو غير تفصيلية [ويطلق عليها خطوات مجردة (abstract steps)] . ويجوز أن نكتب الخطوات بالشفرة الزائفة (pseudocode) أي بمزيج من عبارات باللغة الإنجليزية وعبارات/بنيات تحكم (control structures/statements) بلغة تشبه لغة C++، بحيث يمكن ترجمة هذه الخطوات بسهولة إلى لغة C++.

ونحب أن نؤكد على أن التصميم بطريقة التفكيك الدالي والتصميم بطريقة OOD ليسا منفصلين عن بعضيهما . فأحياناً تحلل طريقة OOD المسألة إلى أهداف تحتوي على بيانات وعمليات مرتبطة بها. وهذه العمليات على الأهداف تتطلب خوارزميات (algorithms). وأحياناً تكون هذه الخوارزميات معقدة (complicated) ويجب تفكيكها (decomposed) وتبسيطها إلى خوارزميات جزئية (subalgorithms) باستخدام التفكيك الدالي.

مثال ٢-٣٠ : لدينا إطار خشبي (wooden frame) مستطيل الشكل نريد أن نشد (stretch) عليه قطعة قماش (cloth) للرسم الزيتي والتلوين (painting) بحيث تُثَبَّت/تُدَبَّس (stapled) قطعة القماش خلف الإطار ، بفرض أن كلا من طول وعرض قطعة قماش التلوين (painting canvas) يزيد عن طول وعرض الإطار بمقدار خمس بوصات (5 inches) ليعطي الجزء الذي يلف (wraps) خلف الإطار .
اكتب برنامجاً:

- يقرأ طول (length) وعرض (width) الإطار الخشبي/ مساحة التلوين (painting)
- يقرأ سعر (cost) الخشب بالدولار للبوصة الواحدة
- يقرأ سعر قماش التلوين بالدولار للقدم المربع الواحد (square foot)
- يحسب ويطبع قيم : طول الخشب المراد شراؤة
  أبعاد قطعة القماش المطلوب شراؤها
  تكاليف الخشب ، وتكاليف قطعة القماش، والتكاليف الكلية.

الحل:

كمثال على التفكيك الدالي (functional decomposition) ، نبدأ أولاً بالمستوى الابتدائي حيث نضع فيه الخطوات الرئيسية في الحل دون ذكر تفاصيل:

Main Module

```
Get length and width
Get wood cost
Get canvas cost
Compute dimensions and costs
Print dimensions and costs
```

وفي المستوى التالي نعطي بعض التفاصيل، فمثلاً الخطوة الأولى تصبح :

```
Get length and width
        Print "Enter length and width of painting:"
        Read length, width
```

وهكذا بالنسبة لباقي الخطوات، ويمكن كتابتها بسهولة، ثم ترجمتها إلى لغة C++ لتعطي البرنامج التالي الذي يوضح تفاصيل هذه الخطوات مع إعطاء التعليقات (comments) التوضيحية.

```
//*********************************************************
//Canvas program
//This program computes the dimensions and costs of materials
//to build a painting canvas of given dimensions. The user is
//asked to enter the length and width of the painting and the
//costs of the wood (per inch) and canvas (per square foot)
//*********************************************************
#include <iostream<
#include <iomanip>    // For setprecision( )

using namespace std;

const float SQ_IN_PER_SQ_FT = 144.0;    // Square inches per
                                        // square foot
int main( )
}
  float length;                 // Length of painting in inches
  float width;                  // Width of painting in inches
  float woodCost;                // Cost of wood per inch in dollars
  float canvasCost;              // Cost of canvas per square foot
  float lengthOfWood;           // Amount of wood to buy
  float canvasWidth;            // Width of canvas to buy
  float canvasLength;           // Length of canvas to buy
  float canvasAreaInches;       // Area of canvas in square inches
  float canvasAreaFeet;         // Area of canvas in square feet
  float totCanvasCost;          // Total cost of canvas being bought
  float totWoodCost;            // Total cost of wood being bought
  float totCost;                // Total cost of materials

  cout << fixed << showpoint;          // Set up floating pt.
                                       // output format
//   Get length and width

  cout << "Enter length and width of painting:" << endl;
  cin >> length >> width;
//   Get wood cost
```

```
  cout << "Enter cost per inch of the framing wood in dollars: "
       >> endl;
  cin >> woodCost;

//  Get canvas cost
  cout << "Enter cost per square foot of canvas in dollars: "
      >>  endl;
  cin >> canvasCost;

//  Compute dimensions and costs
  lengthOfWood = (length + width) * 2;
  canvasWidth = width + 5;
  canvasLength = length + 5;
  canvasAreaInches = canvasWidth * canvasLength;
  canvasAreaFeet = canvasAreaInches / SQ_IN_PER_SQ_FT;
  totWoodCost = lengthOfWood * woodCost;
  totCanvasCost = canvasAreaFeet * canvasCost;
  totCost = totWoodCost + totCanvasCost;

//  Print dimensions and costs
  cout << endl << setprecision(1);
  cout << "For a painting " << length << " in. long and"
       >> width << " in. wide," << endl;
  cout << "you need to buy " << lengthOfWood << " in. of wood,"
      >> "and" << endl;
  cout << "the canvas must be " << canvasLength << " in. long"
       << "and " << canvasWidth << " in. wide." << endl;

  cout << endl << setprecision(2);
  cout << "Given a wood cost of $" << woodCost << " per in."
      >> endl;
  cout << "and a canvas cost of $" << canvasCost
      << "per sq. ft.," << endl;
  cout << "the wood will cost $" << totWoodCost << ',' << endl;
  cout << "the canvas will cost $" << totCanvasCost  ','  <<
      >> endl;
  cout << "and the total cost of materials will be $" << totCost
      >> '.' >>       endl;
  return 0;
{
```

ونلاحظ أن هذا برنامج تبادلي ، حيث تُدخَل قيم البيانات أثناء تنفيذ البرنامج.

وإذا فرضنا أن المستخدم (user) **قد أدخل هذه البيانات:**

24.0 36.0 0.08 2.80

فإن الحوار (dialogue) **بين الحاسوب والمستخدم يبدو كما يلي :**

```
Enter length and width of painting:
24.0             36.0
Enter cost per inch of the framing wood in dollars:
0.08

Enter cost per square foot of canvas in dollrs:
2.80

For a painting 24.0 in. long and 36.0 in. wide,
you need to buy 120.0 in. of wood, and
the canvas must be 29.0 in. long and 41.0  in. wide.

Given a wood cost of $ 0.08 per in.
and a canvas cost of $ 2.80 per sq. ft.,
the wood will cost $ 9.60,
the canvas will cost $ 23.12,
and the total cost of materials will be $ 32.72.
```

تمرينات رقم ٢

٢-١) ماذا تطبع القطعة (code segment) التالية ؟

```
string   str;

str = "Khalid";
cout << "The Companion is" << str + " IbnElWaleed" << endl;
```

٢-٢) أي الأسماء التعريفية (identifiers) التالية صحيحة (valid) وأيها غير صحيحة (invalid) ؟

أ) iteam #1	ب) data	ج) y
د) 3Set	هـ) Pay_Day	و) bin-2
ز) num5	ح) Sq Ft	ط) Mirath

٢-٣-i) اذكر إن كان كل مما يلي كلمة محجوزة (reserved word) أم اسم تعريفي معرف بالمبرمج (programmer-defined identifier) . [إرشاد : انظر قائمة الكلمات المحجوزة : ملحق (أ)]

أ) char	ب) sort	ج) INT
د) long	هـ) Float	و) pi
ز) integer	ح) float	ط) const

ii) هل يمكن استخدام الكلمات المحجوزة كأسماء متغيرات (variable names)؟

(٢-٤) نفرض أن s1, s2 متغيرا سلسلة رموز (string variables) وأن s1 تحتوي "Last" و s2 تحتوي "Day" . ما هي مخرجات كل عبارة من العبارات التالية ؟

(أ)
```
cout << "s1 = " << s1 << " s2 = " << s2 << endl;
```

(ب) `cout << "Believe in:" << s1 + s2 << endl;`

(ج) `cout << "Believe in:     " << s1 + s2 << endl;`

(د) `cout << "Believe in:     " << s1 << ' ' << s2 << endl;`

(٢-٥) اكتب بالضبط مخرجات القطعة التالية:

```
cout    << " Whoever guides someone " << endl
        << " to do a good deed " << endl << endl
        << " will have a reward equal to " << endl
        << endl << endl << endl << " the reward  "
        << " of the doer" << endl;
```

(٢-٦)

(أ) كم عدد الرموز (characters) **التي يمكن تخزينها في متغير من النوع** char؟

(ب) وكم عدد الرموز في سلسلة الرموز الفارغة (null string)؟

(ج) هل يمكن إسناد متغير من النوع string **إلى متغير من النوع** char ؟

(د) وهل يمكن إسناد سلسلة رموز حرفية (literal string) **إلى متغير من النوع** string؟

(هـ) هل هناك فرق (difference) **بين سلسلة الرموز الحرفية** "computer" **والاسم التعريفي** computer؟ **وما هو إن وجد ؟**

(٢-٧) ما هي مخرجات قطعة البرنامج التالية بفرض أن جميع المتغيرات من النوع string ؟

```
street =   "Mousa Ibn Nosair " ;
address =   " 142B " ;
city =   " Cordova " ;
state =   " Andalusia " ;
firstLine =   address + ' ' + street ;
cout << firstLine << endl;
cout << city ;
cout << ",   " << state << endl ;
```

(٢-٨) اكتشف أي أخطاء تركيبية (syntax errors) **في البرنامج التالي:**

```
// This program is full of errors
# include < iostream
# include < string >
using namespace std ;
constant string First : Mohammad " ;
constant string MID : " Ibn_Morad ;
constant string Last : Al_Fateh

int main
{
        string name ;
        character initial ;

        name = Mohammad + Ibn_Morad + Al_Fateh;
        initial = MID;
        LAST = "Al_Fateh The Ottomn " ;
        count   << 'Name =  ' << name << endl;
        cout << Last << FIRST << MID
        cout << endl ;

        return 0 ;
}
```

٢-٩) في برنامج PrintName program **في مثال ٢-٧ نود إضافة طباعة اسم الشخص في الصيغة التالية :**

First-name Middle-initial. Last-name

[Mohammad M. AlFateh **مثلاً**]

عن طريق اتباع الخطوات التالية:

- **الإعلان عن** MIDDLE **كثابت سلسلة رموز** (string constant) **بدلاً من ثابت رمزي** (char constant).
- **تعريف متغير سلسلة رموز** (string variable) **جديد** firstMILast **ليحتفظ** (hold) **بالاسم في الصيغة الجديدة.**
- **تسند لهذا المتغير الجديد سلسلة الرموز** (string) **باستخدام الثوابت المسماة الموجودة** (existing named constants)**، وأي سلاسل رموز**

حرفيـــة (literal strings) نحتاجهـــا لعلامـــات الترقـــيم والفراغـــات (punctuation and spacing) وعمليات التعاقب.

- طباعـة سلسـلة الرمـوز للاسـم في الصـيغة الجديـدة مـع عنـوان مناسـب (appropriate label) . ما هي التعديلات التي تقترحها على البرنامج ؟

٢-١٠) اكتب عبارات إخراج (output statements) تطبع بالضبط المخرجات التالية :

(أ)
```
Allah the Almighty is good
and accepts only that which is good
```

(ب)
```
Allah the Almighty is good
and accepts only that
which is good
```

(ج)
```
Allah the Almighty is good
and

accepts only that
which is good
```

(د)
```
Allah the Almighty
    is good
        and
accepts only that
     which
        is good
```

٢-١١) ما هي بالضبط مخرجات البرنامج التالي ؟

```
// Salam program
// This program prints two simple messages

# include < iostream >
# include < string >
```

```
using naamespace std;

const string MSG1 = " Peace be upon you." ;

int      main ( )
{
         string msg2 ;
         cout << MSG1 << endl ;
         msg2 = MSG1 + "  " + MSG1 + "  " + MSG1 ;
         cout << msg2 << endl ;
         return 0 ;
}
```

٢-١٢) بفرض أن جميع المتغيرات فيما يلي من النوع int، اذكر صحة أو خطأ (valid/invalid) كل من التركيبات (constructs) التالية:

(أ) x * y = c ;
(ب) y = con ;
(ج) const int x : 10 ;
(د) int x ;
(هـ) a = b ٪ c;

٢-١٣) نفرض ان a, b متغيران من النوع int ، حيث a تحتوي القيمة 4 و b تحتوي القيمة ٩. ما هي القيمة المخزونة في a بعد تنفيذ كل من العبارات التالية بفرض أن هذه العبارات مستقلة عن بعضها البعض ؟

(أ) a = 3 * b ;
(ب) a = a + b ;
(ج) a + + ;
(د) a = a / b ;
(هـ) a - - ;
(و) a = a + a ;
(ز) a = b ٪ 6;

٢-١٤) احسب قيمة كل تعبير سليم (تركيبياً) [أي صحيح legal] من التعابير التالية ، وبين إن كانت قيمة التعبير عدداً صحيحاً (integer) أو قيمة ذات

نقطة عائمة (floating point value). وإن كان التعبير غير سليم (أي به أخطاء) فبين السبب.

(أ) 10.0 / 3.0 + 5 * 2

(ب) 10 ٪ 3 + 5 ٪ 2

(ج) 10 / 3 + 5 / 2

(د) 12.5 + (2.5 / (6.2 / 3.1))

(هـ) -4 * (-5 + 6)

(و) 13 ٪ 5/3

(ز) (10.0 / 3.0 ٪ 2) /3

٢-١٥) ما القيمة التي تخزن في المتغير الصحيح result في كل من العبارات التالية ؟

(أ) result = 15 ٪ 4;

(ب) result = 7 / 3 + 2 ;

(ج) result = 2 + 7 * 5 ;

(د) result = 45 / 8 * 4 + 2 ;

(هـ) result = 17 + (21 ٪ 6) *2;

(و) result = int (4.5 + 2.6 * 0.5) ;

٢-١٦) نفرض أن كلا من a, b متغير صحيح ، حيث a تحتوي 5 و b تحتوي 2. ما هي مخرجات كل من العبارات التالية ؟

(أ) cout << "a = " << a << "b = " << b << endl ;

(ب) cout << " Sum : " << a + b << endl ;

(ج) cout << " Sum : " << a + b << endl ;

(د) cout << a / b << " feet " << endl ;

٢-١٧) تتبع البرنامج التالي واكتب مخرجاته :

```
# include < iostream >
```

```
using namespace std ;

const int LBS = 10 ;

int  main ( )
{
        int price ;
        int cost ;
        char ch ;

        price = 30 ;
        cost = price * LBS ;
        ch = ' A ' ;
        cout << " Cost is  " << endl ;
        cout << cost << endl ;
        cout << "Price is   " << price << "Cost is  : " << cost << endl ;
        cout << "Grade   " << ch << " costs " << endl ;
        cout << cost << endl ;
        return 0 ;
}
```

٢-١٨) ترجم عبارة C++ التالية إلى الاصطلاح الجبري (algebraic notation) **.**
(جميع المتغيرات ذوات نقطة عائمة).

```
Y = -b + sqrt ( b * b - 4.0 * a * c );
```

٢-١٩) نفرض أن لدينا قطعة البرنامج (program fragment) **التالية :**

```
int     i;
int     j;
float   z;

i = 4;
j = 17;
z = 2.6;
```

أوجد قيمة كل من التعابير التالية . إذا كانت النتيجة (result) **قيمة ذات نقطة عائمة** (floating-point value) **فضع نقطة عشرية** (decimal point) **في إجابتك .**

(أ) i / float (j)
(ب) 1.0 / i + 2

(ج) z * j

(د) i + j ٪ i

(هـ) (1 / 2) * i

(و) 2 * i + j - i

(ز) j / 2

(ح) 2 * 3 - 1 ٪3

(ط) i ٪ j /i

(ي) int (z + 0.5)

٢-٢٠) بالنسبة لكل عبارة من العبارات التالية اذكر ملف / ملفات المقدمة header file(s) التي يجب أن يحتوي عليها # include برنامج بلغة C++ حتى يمكننا استخدام العبارة.

(أ) cout << x;

(ب) int1 = abs (int2);

(ج) y = sqrt (7.6 + x);

(د) cout << y << endl;

(هـ) cout << setw (5) << someInt;

٢-٢١) أوجد قيمة كل من التعابير التالية ، وإذا كانت النتيجة قيمة ذات نقطة عائمة فضع النقطة العشرية في إجابتك .

(أ) fabs (-9.1)

(ب) sqrt (49.0)

(ج) 3* int (7.8) + 3

(د) pow (4.0, 2.0)

(هـ) sqrt (float (3 * 3 + 4 * 4))

(و) sqrt (fabs (-4.0) + sqrt (25.0))

٢-٢٢) اكتب بالضبط مخرجات البرنامج التالي مستخدماً العلامة □ للدلالة على أي فراغ:

```
# include <iostream>
# include < iomanip >          // For setw ( )
```

```
using namespace std;

int main ( )
{
        char    ch;
        int     n;
        float   y;

        ch = 'A';
        cout << ch;
        ch = 'B';
        cout << ch << endl;
        n = 413;
        y = 21.8;
        cout << setw (5) << n << "  is the value of n" << endl;
        cout << setw (7) << y << "  is the value of y" << endl;
        return 0;
}
```

٢-٢٣) نفرض أن x **متغير عائم (أي ذو نقطة عائمة)** (float variable) **يحتوي على القيمة** 14.3827. **اكتب مخرجات كل من العبارات التالية مستخدماً الرمز** □ **للدلالة على فراغ . (افرض أنه قد تم تنفيذ** cout << fixed)

(أ) `cout << "x is" << setw (5) << setprecision (2) << x`

(ب) `cout << "x is" << setw (8) << setprecision (2) << x`

(ج) `cout << "x is" << setw (0) << setprecision (2) << x`

(د) `cout << "x is" << setw (7) << setprecision (3) << x`

٢-٢٤) نفرض أننا قد أُعطينا العبارات التالية

```
string heading ;
string str ;

heading = "Exam Preparation Exercises";
```

ما هي مخرجات كل قطعة (code segment) **من القطع التالية ؟**

(أ) `cout << heading.length ( ) ;`

(ب)
```
cout << heading.substr (6,  10);
```

(ج)
```
cout << heading.find ( "Ex");
```

(د)
```
str = heading.substr (2,  24);
cout << str.find ( "Ex");
```

(هـ)
```
str = heading.substr (heading.find ( "Ex") + 2, 24);
cout << str.find ( "Ex");
```

(و)
```
str = heading.substr (2, heading.length ( ) -12 );
cout << str.find ( "Ex");
```

٢-٢٥) اكتب عبارة إسناد لحساب مجموع الأعداد الصحيحة من 1 إلى n باستخدام صيغة جاوس (Gauss's formula):

$$\text{sum} = \frac{n(n+1)}{2}$$

وخزِّن النتيجة في المتغير الصحيح sum.

٢-٢٦) نفرض أننا قد أعطينا الإعلانات

```
int     i;
int     j;
float   x;
float   y;
```

اكتب تعبيراً بلغة C++ لكل تعبير جبري مما يلي:

(أ) $\frac{x}{y} - 3$ (ب) $(x+y)\,(x-y)$

(ج) $\frac{1}{x+y}$ (د) $\frac{1}{x} + y$

(هـ) $\frac{i}{j}$ **(النتيجة ذات النقطة العائمة)** (floating point result)

(و) $\frac{i}{j}$ **(خارج القسمة الصحيح)** (integer quotient)

(ز) $\dfrac{\frac{x+y}{3} - \frac{x-y}{3}}{4x}$

٢-٢٧) نفرض أن لدينا الإعلانات

```
int     i;
long    n;
float   x;
float   y;
```

اكتب تعبيراً بلغة C++ لكل تعبير جبري مما يلي ، مستخدما استدعاءات للدوال المكتبية (library functions) **كلما كان ذلك مفيداً .**

(أ) $|i|$ **(القيمة المطلقة للمتغير** i **)**

(ب) $|n|$

(ج) $|x+y|$

(د) $|x|+|y|$

(هـ) $\dfrac{x^3}{y}$

(و) $\sqrt{x^6+y^5}$

(ز) $\left(x+\sqrt{y}\right)^7$

٢-٢٨) يمكن إيجاد جذرَي (roots)**/ حَلَّي** (solutions) **المعادلة التربيعية** (qudratic equation) **باستخدام الصيغة** (formula)

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

حيث نحصل على أحد الحلين باستخدام إشارة + وعلى الحل الآخر باستخدام إشارة - .

اكتب تعابير لحساب الحلين ، بفرض أن جميع المتغيرات عائمة (float variables).

٢-٢٩) رغم قوله تعالى: " يا أيها الذين آمنوا اتقوا الله وذروا ما بقى من الربا إن كنتم مؤمنين، فإن لم تفعلوا فأذنوا بحرب من الله ورسوله، وإن تبتم فلكم رؤوس أموالكم لا تَظلمون ولا تُظلمون" (البقرة : ٢٧٨، ٢٧٩)

فلا زال بعض الناس يصرون على الدخول في هذه الحرب الخاسرة قطعا ويتعاملون بالربا. تتبع (trace) تنفيذ البرنامج التالي واكتب مخرجاته.

```
# include < iostream >

using namespace std:

const float DEBT = 300.0 ;          // Original value owed
const float PMT   = 22.4 ;          // Payment
const float INT_RATE = 0.02 ;       // Interest rate

int main ( )
{
        float charge;           // Interest times debt
        float reduc;            // Amount debt is reduced
        float remaining         // Remaining balance

        charge = INT_RATE  *  DEBT;
        reduc = PMT - charge;
        remaining = DEBT - reduc;
        cout    << "Payment: " << PMT
                << "Charge:     " << charge
                << " Balance owed: " << remaining << endl;
        return 0;
}
```

٢-٣٠) تتبع (trace) البرنامج التالي وحاول فهم وظيفته: أي ماذا يحسب؟ ثم أعد كتابته بعد إضافة تعليقات (comments) توضح وظيفة البرنامج ودلالات ثوابته ومتغيراته (معنى كل منها: إلى ماذا يشير). ثم اكتب مخرجات البرنامج.

```
# include < iostream >

using namespace std:

const int TOT_COST = 1376;
const int POUNDS = 10;
const int OUNCES = 12;
```

```
int main ( )
{
        int   totOz;
        float   uCost;

        totOz = 16  *  POUNDS;
        totOz = totOz  + OUNCES;
        uCost = TOT_COST / totOz;
        cout << "Cost per unit: " << uCost << endl;
        return 0;
}
```

٢-٣١) أكمل البرنامج التالي بحيث يقوم بحساب محيط (perimeter) **ومساحة** (area) **مستطيل** (rectangle) **وطباعة النتائج ، حيث طول المستطيل** (length) **وعرضه** (width) **معطيان (بعبارتي إسناد) . وتأكد من كتابة عناوين** (labels) **مناسبة للمخرجات، وتعليقات** (comments) **مناسبة أثناء الحل.**

```
// *****************************************************
// Rectangle program
// This program finds the perimeter and the area
// of a rectangle, given the length and width
//*****************************************************
# include <iostream>

using namespace std;

int main ( )
{
        float length;                   // Length of the rectangle
        float width;                    // Width of the rectangle
        float perimeter                 // Perimeter of the rectangle
        float area                      // Area of the rectangle

        length = 10.7;
        width = 5.2 ;
```

٢-٣٢-أ) اكتب تعبيراً (expression) تكون نتيجته (result) هي موضع (position) أول حدوث/ظهور (first occurrence) للرموز (characters) "ate" في متغير سلسلة رموز (string variable) اسمه: sentence.

وإذا كان هذا المتغير يحتوي(contains) على الجملة التالية ، فماذا تكون نتيجة التعبير؟

He who innovates something in this matter of ours that is not of it will have it rejected.

ب) اكتب عدة عبارات متتابعة بلغة C++ (sequence of C++ statements) لتطبع موضعي ثاني حدوث وثالث حدوث (second and third occurrences) للرمزين "at" في متغير سلسلة الرموز sentence، بفرض أن هناك على الأقل (at least) ثلاث مرات ظهور لهذين الرمزين في المتغير.

[إرشاد (Hint): استخدام الدالة substr لإنشاء (to create) سلسلة رموز جديدة (new string) محتوياتها (contents) هي جزء (portion) الجملة sentence الذي يلي (following) حدوث الرمزين "at"]

تأكد من صحة حلك بتطبيقه على الجملة (أي على محتويات المتغيرsentence) المعطاة سابقاً في الجزء أ) من السؤال .

٢ -٣٣) نفرض أن لدينا السطرين التاليين (two lines) من البيانات (data):

```
17    13
7     3     24    6
```

وعبارة الإدخال التالية:

```
cin >> int1 >> int2 >> int3 ;
```

ما هي قيمة كل متغير بعد تنفيذ هذه العبارة؟

٢-٣٤) نفرض أن لدينا بيانات الإدخال (input data) التالية :

```
14    21    64
19    67    91
```

```
73      89      27
23      96      47
```

ما هي قيم المتغيرات الصحيحة (int): a, b, c, d **بعد تنفيذ قطعة البرنامج** (program segment) **التالية؟**

```
cin >> a;
cin.ignore (200, '\n');
cin >> b >> c;
cin.ignore (200, '\n');
cin >> d;
```

٢-٣٥) نفرض أننا قد أعطينا بيانات الإدخال التالية:

```
123W 56
```

وأنه قد تم تنفيذ قطعة البرنامج (code segment) **التالية:**

```
int1 = 98;
int2 = 147;
cin >> int1 >> int2;
cout << int1<< ' ' <<  int2;
```

فماذا ستطبع عبارة الإخراج ؟

٢-٣٦) نفرض أن لدينا بيانات الإدخال التالية:

```
11   12.35      ABC
```

ما هي قيمة كل من المتغيرات : i **(وهو من النوع** int**)** و x **(وهو من النوع** float**)** و ch1 **(وهو من النوع** char**) بعد تنفيذ العبارة التالية؟**

(أ)
```
cin >> i  >> x >> ch1;
```

(ب)
```
cin >> ch1>> i  >> x;
```

٢-٣٧-أ) نفرض أن لدينا البيانات المدخلة التالية:

```
40  Badr Street
Mecca, SA 01776
```

وقطعة البرنامج التالية:

```
string  address;
```

cin >> address:

بعد تنفيذ هذه القطعة:

(i) **ما هي سلسلة الرموز** (string) **المخزونة في** address ؟

(ii) **أين يصبح موضع** (position) **مؤشر القراءة** (reading marker) ؟

ب) أعد حل الجزء أ) من السؤال بعد أن نستبدل بعبارة الإدخال (input statement) **العبارة:**

getline (cin, address);

٢-٣٨-أ) صحح أخطاء البرنامج التالي بحيث يقرأ قيمة من سيل الملف inData (file stream) **ويكتبها في سيل الملف** outData (file stream).

```
# include <iostream>

using namespace std;

int main ( )
{

        int       n;
        ifstream inData;

        outData.open ("results. dat");
        cin >> n;
        outData << n << endl;
        return 0;
}
```

ب) بعد تصحيح البرنامج أجب عن السؤالين التاليين:

(i) **إذا كان سيل الملف** inData file stream **يحتوي ابتداءً** (initially) **القيمة** 144، **فماذا سيحتوي بعد تنفيذ البرنامج؟**

(ii) **إذا كان سيل الملف** outData file stream **فارغاً** (empty) **ابتداءً ، فما هي محتوياته بعد تنفيذ البرنامج ؟**

٢-٣٩) نفرض أن برنامجاً به ثلاثة متغيرات رمزية char variables هي : ch1, ch2, ch3.

ونفرض أن بيانات الإدخال هي:

A B C\ n

حيث يوجد فراغان (two blanks) يفصلان كل زوج (each pair) من رموز الإدخال (input characters)

أ) اكتب عبارة/عبارات إدخال input statement(s) تقوم بتخزين الرمز A في ch1 ، والرمز B في ch2 ، والرمز C في ch3 .

ب) اكتب عبارة / عبارات إدخال تقوم بتخزين الرمز A في ch1، وتخزين الفراغين التاليين (next two blanks) في ch2, ch3 .

٢-٤٠) اكتب عبارة إدخال واحدة(single input statement) تقرأ سطور الإدخال (the input lines):

10.25 7.625\n
8.5\n
1.0\n

٢-٤١) اكتب عبارات متتابعة تُدخل (input) الحرف الأول من كل من الأسماء التالية في المتغيرات الرمزية (char variables) chr1, chr2, chr3.

Ahmad\n
Bilal\n
Hamza\n

٢-٤٢) اكتب مجموعة من الإعلانات عن متغيرات (set of variable declarations) وعدة عبارات إدخال متتابعة لقراءة السطور التالية من البيانات وتخزينها في متغيرات من أنواع مناسبة.

اختر أسماء المتغيرات. ولاحظ أن القيم المعطاة مفصولة (separated) عن بعضها البعض بفراغ واحد فقط (single blank) وأنه لا توجد أي فراغات يسار أول رمز في كل سطر.

A 100 2.78 g 14\n
207.98 w q 23.4 92\n
R 42 L 27 R 63\n

٢-٤٣) اكتب قطعة برنامج (program segment) تقرأ تسع قيم صحيحة من ملف ، وتكتبها على الشاشة كل ثلاثة أعداد على سطر إخراج. أفرض أن القيم في الملف مرتبة كل قيمة على سطر مستقل.

٢-٤٤-أ) اكتب قطعة برنامج تبادلي لإدخال بيانات شخص: عمره (age) وطوله (height) ، ووزنه (weight) ، والحرف الأول (initial) من اسمه الأول (first name) والحرف الأول من اسمه الأخير (last name). افرض أن القيم العددية (numeric values) جميعها أعداد صحيحة، وافرض أن الشخص الذي يستخدم البرنامج مبتدئ /مستخدم جديد (beginner/novice user) [أي يحتاج عبارات تنبيهيه مفصلة وعبارات برنامج بسيطة].

ب) أعد كتابة القطعة السابقة لمستخدم ذي خبرة في البرمجة (experienced user) .

٢-٤٥-أ) أكمل فراغات البرنامج التالي الذي يُفترض أن يقرأ أربع قيم من سيل ملف file stream dataIn ويكتبها في / يخرجها إلى سيل ملف file stream resultsOut.

include _____
include _____
using _____

```
int main ( )
{
        int             val1;
        int             val2;
        int             val3;
        int             val4;
        _______         dataIn;
        ofstream        ___________;
        _______ ("myinput.dat");
        _______ ("myoutput.dat");
        _______ >> val1 >> val2 >> val3 >> val4 ;
        _______ << val1 << val2 << val3 << val4 << endl;

        return 0;
 }
```

ب) عدِّل (modify) البرنامج السابق في أ) بحيث أن أسم ملف الإدخال (name of input file) لا يُحدد (specified) كسلسلة رموز حرفية (literal string) وإنما نحث المستخدم على (نطلب من المستخدم) (prompt the user) إدخال اسم الملف ونقرأه ونخزنه وقت تشغيل البرنامج (at run time).

٢-٤٦) الصيغة التربيعية (qudratic formul)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

تعطى جذري (roots) الحدودية التربيعية (quadratic polynomial)

$$ax^2 + bx + c$$

استخدم التفكيك /التحليل الدالي (functionl decomposition) لكتابة خوارزمية (algorithm)لقراءة المعاملات الثلاثة a,b,c لحدودية تربيعية من ملف "inQuad" وكتابة الحلين/الجذرين ذوي النقطة العائمة (two floating-point solutions/roots) في ملف آخر "outQud" افرض أن المميز (discriminant) [وهو المقدار تحت علامة الجذر التربيعي] غير سالب (nonnegative) . ويمكنك استخدام الدالة المكتبية القياسية sqrt (standard library function) واكتب حلك بالشفرة الزائفة (pseudocode) وليس كبرنامج بلغة C++.

٢-٤٧) عدِّل برنامج مثال ٢-٣٠ (برنامج لوحة التلوين Canvas program) بحيث يقرأ البيانات المدخلة (input data) من ملف بدلاً من لوحة المفاتيح. وأثناء وقت التشغيل (run time) حث المستخدم (prompt the user) لإدخال اسم الملف الذي يحتوي على البيانات.

٢-٤٨) اذكر صحة أو خطأ (T/F) كل من العبارات التالية:

(أ) الاسم التعريفي (identifier) في لغة C++ لا يمكن أن يبدأ برقم.

(ب) العبارة المركبة/القالب (block /compound statement) في لغة C++ لا يُنهي (not terminated) بفاصلة منقوطة.

(ج) إذا كان x,y متغيرين من النوع float، فإن التعبير

sqrt (fabs (3.8 * x + 9.4 + y))

يعد استخداماً صحيحاً (valid) للدالتين المكتبيتين sqrt, fabs (library functions) .

(د) إذا أُسندت قيمة ذات نقطة عائمة (floating-point value) لمتغير صحيح (integer) فإن الجزء الكسري (fractional part) يقطع (turncated).

(هـ) يُستخدم المعالج (manipulator) setw فقط لصياغة (formatting) القيم العددية (numeric values) وسلاسل الرموز (strings) وليس البيانات الرمزية (char data).

(و) باستخدام المؤثر .. يمكننا قراءة قيمة بيانات ذات نقطة عائمة وتخزينها في متغير من النوع int.

(ز) عبارة الإدخال : cin >> someInt ;

يمكن كتابتها أيضاً هكذا : someInt << cin;

(ح) في العبارة : cin >> XXXX;

يجب أن يكون XXXX اسم متغير (variable name) وليس ثابتاً أو تعبيراً اختيارياً (constant or arbitrary expression).

(ط) في استدعاء الدالة : cin. get (XXXX),

يجب أن يكون XXXX اسم متغير وليس ثابتاً أو تعبيراً اختيارياً.

(ي) إذا كان مؤشر القراءة وسط سطر إدخال (input line) مكون من 25 رمز، فإن تنفيذ العبارة

cin.ignore (500, '\n')

يترك مؤشر القراءة عند الرمز الذي يلي رمز السطر الجديد التالي (next newline character) .

(ك) الدالتان get, ignore تعدان من الدوال الأعضاء (member functions) من الطبقة istream class.

(ل) الدالتان find, substr تعدان من الدوال الأعضاء من الطبقة string class.

(م) الدالة getline تعد من الدوال الأعضاء من الطبقة string class.

٢-٤٩) توفى رجل عن زوجة وعدد من الأبناء الذكور NS والإناث ND (بحيث أن NS>0 , ND>0) وترك ميراثا قدره A دينارا.

اكتب برنامجا لتوزيع هذا الميراث على ورثته تبعا للقاعدتين التاليتين:

(أ) فإن كان لكم ولد فلهن الثمن مما تركتم.

(ب) يوصيكم الله في أولادكم للذكر مثل حظ الأنثيين.

ونفرض أننا سنرمز لنصيب الزوجة بالرمز W ونصب الابن بالرمز S ونصيب البنت بالرمز D .

ابدأ البرنامج بقراءة قيم البيانات A , NS , ND واستخدم عبارة تعليق في مطلع البرنامج لبيان وظيفته ، وعبارات تعليق توضيحية في ثناياه.

٢-٥٠) اكتب برنامجا يطبع الرسالة التالية :

There is no god but Allah
Mohammad is the Messenger of Allah

٢-٥١) اكتب برنامجا لقراءة قيم ثلاثة أعداد حقيقية a , h , d وحساب وطباعة قيمة كل من :

(أ) مساحة مثلث طول قاعدته a وارتفاعه h .

$$A = \frac{1}{2}ah$$

(ب) حجم اسطوانة قائمة طول نصف قطر قاعدتها a وارتفاعها h ، وكذلك المساحة الجانبية لسطحها

$$V = \pi a^2 h$$
$$SA = 2\pi a\, h$$

(ج) حجم مخروط دائري قائم طول نصف قطر قاعدته a وارتفاعه h .

$$V = \frac{1}{3}\pi a^2 h$$

(د) حجم هرم رباعي قائم طول ضلع قاعدته المربعة a وارتفاعه h .

$$V = \frac{1}{3}\pi a^2 h$$

(هـ) وزن كرة طول نصف قطرها a والكثافة النوعية لمادتها d

$$W = \frac{4}{3}\pi a^3 d$$

ملاحظة : عرف الثابت $\pi = 3.1415927$ في قسم اعلانات البرنامج.

٢-٥٢) اكتب برنامجا بلغة الباسكال يقوم بالتالي :

(أ) قراءة قيمتي DAYS و WAGE حيث :

DAYS تعني عدد أيام الجهاد التي قضاها الجندي مرابطا في سبيل الله تعالى ونفرض أن $DAYS \geq 40$

WAGE تعني الأجر اليومي بالدينار الذي يتقاضاه الجندي.

(ب) حساب قيمة كل من الأجر العادي للجندي REGPAY ، وأجهره الإضافي OVERTM ، وأجره الكلي TOTPAY ، حسب القوانين التالية :

$$PEGPAY = 40 \times WAGE$$

$$OVERTM = (DAYS - 40)(WAGE \times 1\frac{1}{2})$$

$$TOTPAY = REGPAY + OVERTM$$

(ج) طباعة النتائج ..

٢-٥٣) اكتب برنامجا يقوم بقراءة عدد أفراد أسرة N ، ويحسب قيمة زكاة الفطر Z بالدينار عن الأسرة بفرض أن قيمة الزكاة دينار عن الفرد الواحد (أنظر المسألة رقم ١-٣) ، ويطبع قيمة كل من Z , N .

٢-٥٤) اكتب برنامجا لقراءة قيمة كل من طول مستطيل وعرضه ثم حساب كل من محيطه ومساحته وطول قطره.

اطبع رسائل توضيحية كافية لتوضيح المخرجات.

٢-٥٥) اكتب برنامجا يقرأ قيمة متغير R ثم يحسب ويطبع :

(أ) محيط ومساحة دائرة نصف قطرها R.

(محيط الدائرة = $2\pi R$ ، مساحة الدائرة = πR^2 ، قيمة π = 3.1415927)

(ب) حجم كرة نصف قطرها R.

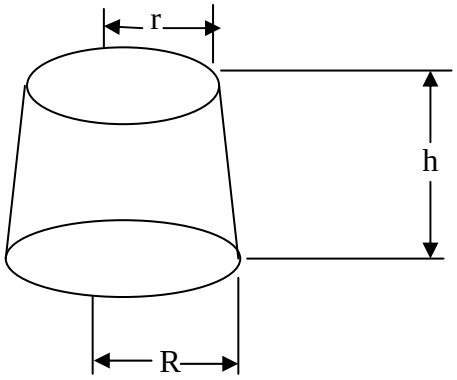
(حجم الكرة = $\frac{4}{3}\pi R^3$)

٢-٥٦) تتحرك عربة بعجلة ثابتة a لمدة t ثانية. اكتب برنامجا لقراءة قيمة كل من a , t وحساب المسافة التي تحركتها العربة $d = \frac{1}{2}at^2$ ، والسرعة النهائية $v = a.t$ مع كتابة عناوين مناسبة للنتائج.

٢-٥٧) حجم المجسَّم المخروطي الدائري القائم المبين بالشكل يعطى بالعلاقة :

$$V = \pi h/3\ (R^2 + rR + r^2)$$

اكتب برنامجا لقراءة قيمة كل من r , R , h وحساب الحجم V بفرض أن $\pi = 3.1415927$



٢-٥٨) اكتب برنامجا لتحويل درجة حرارة من التقدير الفهرنهيتي إلى التقدير المئوي .

مدخلات البرنامج : Fahrenheit : عدد صحيح (درجة الحرارة بالتقدير الفهرنهيتي).

مخرجات البرنامج : Celsius : عدد حقيقي (درجة الحرارة بالتقدير المئوي).

معادلة التحويل : Celsius = 5/9 x (Fahrenheit - 32)

٢-٥٩) اكتب برنامجا لقراءة عددين صحيحين وطباعة كل من مجموعهما والفارق بينهما وحاصل ضربهما وخارج قسمة أحدهما على الآخر.

مدخلات البرنامج : x , y : عددان صحيحان.

مخرجات البرنامج :

عدد صحيح (مجموع x , y)	:	Sum
عدد صحيح (الفارق بين x , y)	:	Difference
عدد صحيح (حاصل ضرب x , y)	:	Product
عدد حقيقي (حاصل قسمة x على y)	:	Quotient

٢-٦٠) اكتب برنامجا ليقرأ وزن جسم بالرطل (الباوند pound) ويحسب ويطبع الوزن بالكيلو جرام وكذلك بالجرام.

(إرشاد : ١ رطل يكافئ ٠,٤٥٣٥٩٢ كيلو جراما)

٢-٦١) اكتب برنامجا يطبع الحرف الأول من اسمك وذلك في مساحة عبارة عن شبكة (grid) مربعة ٦ × ٦ من الخانات (المواضع) لهذا الحرف ، بأن يطبع البرنامج ست سلاسل من الرموز (strings) حيث تتكون السلسة الواحدة من عدد من النجوم (* asterisks) تفصل بينها فراغات.

٢-٦٢) اكتب برنامجا يقرأ طول وعرض فناء (yard) مستطيل الشكل ، وأيضا يقرأ طول وعرض بيت مستطيل الشكل يقع (situated) في الفناء ، ثم يحسب الوقت اللازم لقطع العشب (cut the grass)

الموجود حول البيت داخل الفناء وذلك بمعدل ٢ متر مربع في الثانية.

٢-٦٣ أ) اكتب برنامجا يقرأ بَسْطَي (numerators) ومقامَي (denominators) كسرين (two fractions) ثم يطبع بسط ومقام الكسر الذي يمثل حاصل ضرب الكسرين ، وأيضا يطبع النسبة المئوية المكافئة (percent equivalent) لحاصل الضرب الناتج.

(ب) أعد كتابة برنامج الجزء (أ) ولكن هذه المرة احسب مجموع الكسرين.

٢-٦٤) تنص نظرية فيثاغورث (Pythagorean theorem) على أن مجموع مربعي ضلعي مثلث قائم الزاوية يساوي مربع طول الوتر (hypotenuse). فمثلا إذا كان طولا ضلعي مثلث قائم الزاوية هما ٣ و ٤ فإن طول الوتر يجب أن يساوي ٥. ويقال أن الأعداد الثلاثة ٣ و ٤ و ٥ تكوِّن ثلاثيا فيثاغورثيا (Pythagorean triple). ويوجد عدد لا نهائي من هذه الثلاثيات. وإذا أعطينا عددين موجبين m , n حيث $m > n$ فإنه يمكن توليد ثلاثي فيثاغورثي بالعلاقات الآتية :

$Side1 = m^2 - n^2$
$Side2 = 2mn$
$Hypotenuse = m^2 + n^2$

اكتب برنامجا يقرأ قيما للعددين m , n ويطبع قيم الثلاثيات الفيثاغورثية المقابلة باستخدام العلاقات السابقة.

٢-٦٥) اكتب برنامجا يقوم بحساب كل من الوقت الذي تستغرقه قذيفة في الطيران ، وارتفاعها فوق سطح الأرض ، وذلك عندما تصل القذيفة إلى هدفها.

استعن بالمعلومات التالية :

G = 32.17 {Gravitational Constant} **ثابت الجاذبية الأرضية**

Program input

مدخلات البرنامج

```
Theta : float       //Input-Angle (radians) of elevation
Distance : float    //Input - Distance (ft) to target
Velocity : float    //Input - Projectile Velocity (ft/sec)
```

Program output **مخرجات البرنامج**

```
Time   : float      //output - Time (sec) of flight
Height : float      //output - height at impact
```

Relevant Formulas **العلاقات / القوانين المستخدمة**

Time = distance / (velocity x cos (theta))

Height = velocity x time - (g x time^2) / 2

٢-٦٦) راكب دراجة (cyclist) على طريق مستو (level road) تتباطأ سرعته من ١٠ ميل في الساعة إلى ٢,٥ ميل في الساعة وذلك خلال دقيقة واحدة. اكتب برنامجا لحساب المعدل الثابت للتسارع (constant rate of acceleration) a ، وتعيين الزمن الذي يأخذه راكب الدراجة حتى يتوقف. اعتبر أن السرعة الابتدائية هي ١٠ ميل في الساعة.

إرشاد : استخدم المعادلة : $a = (v_f - v_i)/t$

حيث a هي التسارع ، t هي الفترة الزمنية ، v_i هي السرعة الابتدائية ، v_f هي السرعة النهائية.

٢-٦٧) يرغب صاحب مصنع في تعيين تكلفة إنتاج إناء اسطواني مفتوح من أعلى ، حيث المساحة السطحية للإناء هي مجموع مساحة القاعدة (ط × مربع نصف القطر) والمساحة الجانبية (٢ط × نصف القطر × ارتفاع الاسطوانة). اكتب برنامجا لقراءة نصف قطر القاعدة (radius) ، وارتفاع الإناء (Height) والتكلفة لكل سم٢ للمادة المستخدمة (Cost) ، وعدد

الأواني المطلوب إنتاجها (Quantity). **ثم يحسب البرنامج كلا من تكلفة إنتاج إناء واحد ، والتكلفة الكلية لإنتاج كل الأواني.**

الفصل الثالث

عبارات التحكم/ الاختيار
Selection / Control Statements

يتعلق هذا الفصل بالتعابير المنطقية (logical expressions) البسيطة (simple) و المركبة (complex) والتي تساعدنا في تكوين وتقييم (evaluating) شروط (conditions) معينه تتحكم (control) في تسلسل/سيل (flow) تنفيذ عبارات البرنامج غير التسلسل الطبيعي: العبارة الأولى ثم التي تليها ثم التي تليها وهكذا، أي حسب الترتيب الطبيعي/الفيزيائي (physical order) للعبارات . ومن أهم العبارات التي تستخدم للتحكم في سير تنفيذ عبارات البرنامج واختيار (selection) تنفيذ عبارات قبل أخرى : العبارات المسماة بعبارات If بأنواعها المختلفة :

إذا كان ... فإن ... (if … Then …)

و إذا كان ... فإن ... وإلا ... (if … Then … else …)

وعبارات if المتداخلة (nested if statements).

ونبدأ بتعريف نوع البيانات الذي يساعدنا في تكوين هذه التعابير والشروط المنطقية.

نوع البيانات bool (Data Type)

هذا النوع من البيانات يُعد في لغة C++ من الأنواع المبنية (built-in type) التي تتكون من قيمتين فقط: وهما الثابتان true, false . والكلمة المحجوزة bool هي اختصار كلمة Boolean. وهذا النوع من البيانات والذي يطلق عليه البيانات المنطقية/البُولية (Logical/Boolean data) يستخدم لاختبار الشروط (testing conditions) في البرنامج بحيث يستطيع الحاسوب اتخاذ قرارات (decisions) ذات بنية تحكم في الاختيار (selection control structure).

ويتم الإعلان عن المتغيرات من النوع bool بالطريقة نفسها التي نعلن بها عن المتغيرات من الأنواع الأخرى ، أي بكتابة اسم نوع البيانات ثم اسم تعريفي:

```
bool    dataOK      // True if the input data is valid
bool    done        // True if the progress is done
bool    zakatble    // True if zakat should be paid
```

وأي متغير من النوع bool يمكن أن يحتوي على إحدى قيمتين true, false. ومن المهم أن نؤكد على أن true, false ليسا اسمي متغيرين (variable names)، كما أنهما ليسا سلسلتي رموز (strings) ، وإنما هما ثابتان من نوع خاص (special constants) في لغة C++ ، وهما في الحقيقة كلمتان محجوزتان (reserved words).

التعابير المنطقية (Logical Expressions)

كما يتكون التعبير الحسابي من قيم عددية وعمليات ، يتكون التعبير المنطقي من قيم منطقية وعمليات . وأي تعبير منطقي له إحدى قيمتين true, false. وفيما يلي بعض الأمثلة للتعابير المنطقية:

- متغير منطقي أو ثابت منطقي (Boolean variable/constant).
- تعبير يليه مؤثر علاقي (relational operator) يليه تعبير.
- تعبير منطقي (logical expression) يليه مؤثر منطقي (logical operator) يليه تعبير منطقي .

وفيما يلي نعرض بعض التفاصيل لكل من هذه الأمثلة للتعابير المنطقية:

المتغيرات و الثوابت المنطقية
(Boolean Variables and Constants)

كما ذكرنا سابقا فإن المتغير المنطقي هو متغير أُعلن عنه أنه من النوع bool، ويمكنه أن يحتوي على القيمة true، أو على القيمة false . مثلاً إذا كان dataOK متغيراً منطقياً فإن

```
dataOk = true;
```

عبارة إسناد صحيحة (valid assignment statement) .

المؤثرات العلاقية (Relational Operators)

المؤثـــر العلاقـي يختبـــر (tests) صحة علاقة (relationship) بين قيمتين ، مثل x < y وكطريقة أخرى لإسناد قيمة لمتغير منطقي – بدلاً من إسناد القيمة (true/false) مباشرة بعبارة إسناد مثل dataOK = true; – أن نجعله مساوياً لنتيجة (result) مقارنة (comparing) تعبيرين باستخدام مؤثر علاقي.

فمثـــلاً في قطعـــة البرنامـــج التاليـــة لدينا متغيران صحيحان i , j (int variables) ومتغير منطقي lessThan (Boolean variable):

```
cin >> i >>  j;
lessThan = (i < j)          // Compare i and j with the
                            // " < " relational operator ,
                            // and assign the truth value to lessThan
```

أي أنه إذا كانت فعـــلاً قيمة i أصغر من قيمة j فإن قيمة الصحة (truth value) للتعبير يمين علامة الإسناد " = " تكون true ، وإلا فإن قيمة الصحة تكون false. ثم تُسنَد قيمة الصحة هذه للمتغير المنطقي lessThan الموجود يسار علامة الإسناد.

والمؤثرات العلاقية في لغة C++ هي :

المؤثر العلاقي	معناه	المصطلح الرياضي المقابل
= =	يساوي	=
= !	لا يساوي	≠
>	أكبر من	>
<	أصغر من	<
= <	أكبر من أو يساوي	≥
= >	أصغر من أو يساوي	≤

التعبير العلاقي (relational expression)

التعبـــير العلاقـي هـو تعـبير يتبعـه مـؤثر علاقـي يتبعـه تعـبير. ونتيجـة التعـبير العلاقي تكون من النوع bool . مثلاً إذا كانـت قيمـة x هـي 5 ، وقيمـة y هـي 10 فإن قيمة كل من التعابير التالية هي true.

```
x ! = y
y > x
x < y
y >= x
x <= y
```

وبالمثل إذا كانـت x هـي الرمـز 'M' (character) ، و y هـي الرمـز 'R' فـإن قيم التعابير السابقة تظـل جميعها true، وذلـك لأن التعبير العلاقـي < يسـتخدم مـع الحروف (letters) بمعنى : " يأتي في الترتيب الأبجدي قبل" (comes before in the alphabet) ، أو بتعبير أدق " يـأتي في سلسلة المقارنـة للرمـوز قبـل" (comes before in the collating sequence of the character set) . فمثلاً في مجموعـة الرموز ASCII المستخدمة على نطاق واسع (widely used character set) جميع الحـروف الكـبيرة (uppercase letters) تـأتي في ترتيـب أبجـدي (alphabetical order) ، وكذلك جميع الحـروف الصغيرة تـأتي في ترتيـب أبجـدي ، ولكـن جميـع الحروف الكبيرة تأتي قبل جميع الحروف الصغيرة، وبالتالي فإن

قيمة	'R' > 'M'	هي	true
"	'R' > 'm'	"	true
ولكن قيمة	'R' > 'm'	"	false

وعند المقارنة يُفضل دائماً – ومن الأكثر أماناً (safer) – أن يكون طرفا المقارنة مـن النوع نفسه ، بمعنـى أن نقارن دائماً int مع int ، وكـذلك float مع float ، و char مع char ، ... وهكـذا. فـإن كـان هنـاك خلـط (mixing) في أنـواع البيانـات عنـد

المقارنة ، فإن التحويل الضمني للنوع (implicit type coercion) يحدث تماماً كما في حالة التعابير الحسابية (arithmetic expressions) . فمثلاً إذا قارنا بين قيمة int وقيمة float، فإن الحاسب يحوّلcoerces القيمة الصحيحة (int value) بصفة مؤقتة (temporarily) إلى مكافئها العائم (float equivalent) قبل إجراء المقارنة. وكما هو الحال في التعابير الحسابية فيفضل استخدام التحويل الصريح للنوع (explicit type casting) .

someFloat = float (someInt)

وإذا قارنا قيمة منطقية (bool value) مع قيمة عددية (numeric value) – ربما بطريق الخطأ – فإن القيمة false تُحوّل (coerced) مؤقتاً إلى العدد 0 (number) ، بينما تحول القيمة true إلى العدد 1. ولذلك فإذا كان boolVar متغيراُ منطقياً bool variable فإن التعبير boolVar < 5 يعطي دائماً القيمة true وذلك لأن كلا من 0, 1 أصغر من 5.

وعند التعامل مع القيم الرمزية char في المقارنات ، فيفضل دائماً مقارنة القيم char مع القيم من النوع نفسه char، مثل المقارنة '9' > '0' أما المقارنة 9 > '0' فإنها تؤدي إلى تحويل ضمني للنوع وربما إلى نتائج غير متوقعة.

والمؤثرات العلاقية لا تستخدم لمقارنة المتغيرات والثوابت فحسب ، وإنما أيضا لمقارنة قيم التعابير الحسابية ، كما يوضح ذلك المثال التالي:

مثال ٣-١: أوجد قيم التعابير التالية لقيم x, y المعطاة

	التعبير	y	x
(أ)	x + 3 <= y * 10	2	12
(ب)	x + 3 <= y * 10	2	20
(ج)	x + 3 != y * 10	1	7
(د)	x + 3 = = y * 10	2	17
(هـ)	x + 3 > y * 10	5	100

الحل: (أ) true (ب) false (ج) false

(د) true (هـ) true

ملاحظة: هناك فارق كبير بين مؤثر الإسناد " = " (assignment operator) والمؤثر العلاقي " = = " (relational operator) وتأثيراهما مختلفان تماماً ، فيجب الحذر في استعمالهما ، والانتباه إلى عدم الخلط بينهما.

المقارنة بين سلاسل الرموز (Comparing Strings)

ذكرنا في الفصل السابق أن string تعد طبقة (class) – وهي نوع معرف بالمبرمج (programmer - defined type) يمكن منه الإعلان عن متغيرات يطلق عليها عادة "أهداف" (objects) – وكل هدف من النوع string يحتوي داخله على سلسلة رموز character string. والطبقة string (class) مصممة بحيث يمكن المقارنة بين هذه السلاسل باستخدام المؤثرات العلاقية. ومن الناحية التركيبية (syntactically) فإن معامَلَي (operands) أي مؤثر علاقي يمكن أن يكونا:

- هدفين من النوع string (2 string objects) ، كما في العلاقة :

  myString < yourString

- أو هدفا من النوع string، وسلسلة رموز C (C string) ، كما في العلاقة

  myString >= "Khalid"

ولكن لا يجوز للمعامَلين أن يكونا سلسلتي رموز C معاً.

والمقارنة بين سلسلتي رموز تتبع سلسلة المقارنة (collating sequence) الخاصة بمجموعة رموز الماكينة (machine's character set) (مثل شفرة ASCII مثلاً) . وعندما يختبر الحاسوب علاقةً ما بين سلسلتي رموز، فإنه يبدأ بالرمز الأول في كل من السلسلتين ويقارن بينهما بناء على سلسلة المقارنة (collating sequence) ، فإن تطابقا فإنه يكرر المقارنة مع الرمز التالي (next character) في كل من السلسلتين. ويستمر اختبار المقارنة رمزاً رمزاً إلى أن يحدث عدم اتفاق (mismatch) أو أن نقارن أخر رمزين ويتفقا. فإن تساوت/تطابقت جميع الرموز المتقابلة فإن سلسلتي

الرموز متساويتان، وإذا حدث عدم اتفاق فإن السلسلة التي تحتوي على الرمز الذي يأتي قبل الرمز الآخر المقابل هي السلسلة الأصغر/الأقل (lesser string).

مثال ٣-٢: نفرض أن لدينا العبارات

```
string   word1;
string   word2;

word1  =  " Temporary ";
word2  =  " Small ";
```

أوجد قيمة (value) كل من التعابير العلاقية التالية:

(أ) word1 = = word2
(ب) word1 > word2
(ج) word1 < "Temperature"
(د) word2 = = "Small"
(هـ) "moon" < "sun"

الحل: (أ) false : غير متساويين في الرمز الأول.

(ب) true: ' T ' تأتي بعد ' S ' في سلسلة المقارنة.

(ج) false: عدم اتفاق في الرمز الخامس، و ' e ' تأتي قبل ' o '.

(د) true: السلسلتان متساويتان.

(هـ) غير متوقعة (unpredictable) : لا يجوز أن يكون المعامَلان سلسلتي C معاً(*)

(*) التعبير سليم (legal) من الناحية التركيبية (syntactically) في لغة C++، ولكنه يؤدي إلى مقارنة مؤشرات (pointer comparison) وليس مقارنة سلاسل رموز (string comparison). وموضوع المؤشرات (pointers) هو خارج نطاق موضوعات هذا الكتاب.

ويلاحظ أنه في معظم الحالات يمكن – باستخدام الطريقة المذكورة للمقارنة بين سلاسل الرموز –ترتيب السلاسل (ordering of strings) بناءً على ترتيبها الأبجدي (alphabetical ordering)، ولكن في حالة السلاسل المختلطة الحروف – الكبيرة والصغيرة – يمكن أن تؤدي هذه الطريقة إلى ترتيب غير أبجدي (nonalphabetical) . فمثلاً في دليل التليفونات نتوقع أن يظهر الاسم Ezzat قبل الاسم EzzUddeenElQuassam ، ولكن سلسلة ASCII للمقارنة تضع جميع الحروف الكبيرة (uppercase letters) قبل الحروف الصغيرة ، ولذلك تعتبر السلسلة EzzUddeenElQuassam أصغر / أقل من السلسلة Ezzat. وللمقارنة بين السلاسل بناء على الترتيب الأبجدي بالضبط (strict alphabetical ordering) ، يجب أن تكون جميع الرموز كبيرة (uppercase) أو أن تكون جميعها صغيرة (lowercase) أي تكون جميعها في الحالة نفسها (same case). وعموماً توجد خوارزمية (algorithm) لتغيير حالة سلسلة.

وإذا كانت هناك سلسلتان مختلفتا الطول، وتطابقت جميع رموزهما حتى نهاية السلسلة الأقصر ، فإن السلسلة الأقصر تعد أصغر/أقل من (less than) السلسلة الأطول. فمثلاً إذا كانت word2 تحتوي على "Small" ، فإن التعبير word2 < " Smaller " يعطي القيمة true، لأن السلسلتين متطابقتان حتى موضع الرمز الخامس (نهاية السلسلة التي في الطرف الأيسر) ، والسلسلة التي في الطرف الأيمن أطول.

المؤثرات المنطقية (Logical Operators)

نتعامل في الرياضيات (mathematics) مع مؤثرات منطقية ثلاثة هي : AND, OR, NOT ، حيث معامَلاتها (operands) جميعاً تعابير منطقية. ولغة C++ تستخدم الرموز الخاصة التالية لهذه المؤثرات المنطقية:

& &	:	AND
||	:	OR
!	:	NOT

وباستخدام مؤثرات علاقية مع مؤثرات منطقية يمكننا تكوين شروط مركبة أكبر (more complex). مثلاً لتحديد ما إذا كانت الدرجة النهائية (final score) أكبر من 90 ودرجة منتصف الفصل (midterm score) أكبر من 70 نكتب التعبير التالي:

finalScore > 90 && midtermScore > 70

وعملية AND المنطقية (&&) تتطلب أن تكون كل من العلاقتين (relationships) صادقة/صحيحة true حتى تكون النتيجة الكلية (overall result) صادقة true. وإذا كانت إحدى العلاقتين كاذبة/خاطئة false (أو كلاهما false) فإن النتيجة الكلية تكون false.

وأما بالنسبة لعملية OR المنطقية (||)فإنها تأخذ تعبيرين منطقيين، فإن كان أحدهما أو كلاهما true، فإن النتيجة تكون true . ولا تكون النتيجة false إلا إذا كان كلاهما false.

وكمثال لاستخدام المؤثر (||) نفرض أننا نود اختبار شرط أن يكون تقدير منتصف الفصل "A" أو أن يكون التقدير النهائي "A". في هذه الحالة نكتب التعبير التالي:

midtermGrade = = 'A' || finalGrade = = 'A'

ويطلق على كل من (||) ، (&&) "مؤثر ثنائي" (two-operand / binary operator) لأن كلا منهما يظهر بين تعبيرين . وأما المؤثر (!) فيطلق عليه "مؤثر أحادي" (one-operand / unary operator) . وهو دائماً يسبق (precedes) تعبيراً منطقياً وحيداً ، ويعطي نتيجة (result) هي عكس (opposite) نتيجة التعبير. مثلاً إذا كانت قيمة التعبير

grade = = 'A'

false ، فتكون نتيجة التعبير

! (grade = = 'A')

true. وعملية NOT المنطقية تعطينا طريقة مناسبة لعكس (reversing) معنى تعبير معين. مثلاً التعبير

! (hours > 40)

يكافئ التعبير

hours > = 40

وفي بعض الأحيان تكون الصيغة الأولى أوضح ، وأحياناً أخرى تكون الصيغة الثانية هي الأنسب، ولذلك فأي الصيغتين أفضل يعتمد على طبيعة الشرط/التعبير المطلوب نفيه. وفيما يلي مزيد من أمثلة التعابير المتكافئة.

مثال ٣-٣ : فيما يلي أمثلة لأزواج (pairs) من التعابير المتكافئة (equivalent expressions) التي تستخدم مؤثر النفي (!) (negation operator) .

تعبير Expression	تعبير مكافئ Equivalent Expression
! (a = = b)	a ! = b
! (a = = b || a = = c)	a ! = b && a ! = c
! (a = = b && c > d)	a ! = b || c <= d

نلاحظ أن أي تعبير على اليسار هو نفسه التعبير المكافئ له على اليمين بعد إضافة المؤثر ! وعكس كل من المؤثرات العلاقية والمؤثرات المنطقية (مثلاً = = بدلاً من = ! ، و || بدلاً من &&) . [ملاحظة : في الجبر المنطقي يطلق على هذا النمط (pattern) لإيجاد التعبير المكافئ: نظرية/قانون "دي مورجان" (DeMorgan's law)] .

ويمكن تطبيق المؤثرات المنطقية على نتائج المقارنات. كما يمكن تطبيقها مباشرة على متغيرات من النوع bool . فمثلاً بدلاً من كتابة

```
isZakatable = (savings >= Nisab   &&   months = = 12);
```

لإسناد قيمة للمتغير المنطقي isZakatable، يمكننا استخدام متغيرين منطقيين وسطيين (intermediate Boolean variables) :isOver, isDue

```
isOver = (savings >= Nisab) ;
isDue = (months = = 12) ;
isZakatable = isOver  &&  isDue;
```

والجدولان التاليان يلخصان نتائج تطبيق المؤثرين | | ، && على زوج من التعابير المنطقية (يمثَّل في كل من الجدولين بالمتغيرين المنطقيين x , y) .

قيمة x	قيمة y	قيمة x && y
true	true	true
true	false	false
false	true	false
false	false	false

جدول صحة المؤثر المنطقي &&

قيمة x	قيمة y	قيمة x || y
true	true	true
true	false	true
false	true	true
false	false	false

جدول صحة المؤثر المنطقي ||

والجدول التالي يلخص نتائج تأثير مؤثر النفي ! على تعبير منطقي يُرمز له بالمتغير المنطقي x.

قيمة x	قيمة !x
true	false
false	true

جدول صحة المؤثر المنطقي !

ومن الناحية الفنية (technically) فإن المؤثرات || , && , ! في لغة C++ لا تستلزم بالضرورة أن تكون معامَلاتها (operands) تعابير منطقية، بل يمكن أن تكون من أي نوع بسيط للبيانات (simple data type)، حتى من النوع float . وإذا لم يكن أحد المعامَلات من النوع bool، فإن قيمته تُحوَّل (coerced) مؤقتاً إلى النوع bool كما يلي:

القيمة 0 تحول إلى false ، وأي قيمة غير صفرية (nonzero value) تحول إلى true. وكمثال على هذا، فيمكننا أن نكتب ما يلي :

```
float    height
bool     badData;
:
cin >> height ;
badData = ! height;
```

وعبارة الإسناد تعني أن نجعل قيمة badData مساوية true إذا كانت قيمة height المحوَّلة هي false. وبأسلوب آخر فإن العبارة تجعل badData مساوياً true إذا كانت قيمة height تساوي صفراً 0.0 . ورغم أن عبارة الإسناد هذه تعطي نتائج صحيحة في لغة C++ ، إلا أن العبارة التالية أوضح منها:

```
badDat = (height = = 0.0);
```

وعموماً في هذا الكتاب لن نستخدم المؤثرات المنطقية إلا مع التعابير المنطقية فقط ، ولن نستخدمها مع التعابير الحسابية.

تحذير : المؤثران المنطقيان | | و && مختلفان تماماً في وظيفتيهما عن مؤثرين آخرين في لغةC++ هما | و & ولن نناقش هذين المؤثرين الأخيرين في هذا الكتاب. وإذا استخدمنا & بدلاً من && أو استخدمنا | بدلاً من | | فلن نحصل على رسالة خطأ (error message) من الحاسب، وقد نحصل على نتائج خاطئة .. لذلك ننبِّه القارئ الكريم إلى عدم الالتباس هذا.

التقييم السريع/قصير الدائرة (Short-Circuit Evaluation)

تطبِّق لغة C++ التقييم الشرطي (conditional evaluation) السريع ، أي إيجاد قيمة الشروط/التعابير المنطقية بسرعة ، عن طريق تقييم التعبير المنطقي بالترتيب من اليسار لليمين مع إيقاف التقييم (evaluation stopping) بمجرد إمكانية معرفة/تحديد (determining) قيمة الصحة النهائية (final truth value) للتعبير.

مثال ٣-٤ :

(أ) لإيجاد قيمة التعبير المنطقي `i = = 1 & & j > 2`

يبدأ التقييم من اليسار متجهين لليمين ، فإذا فرضنا أن قيمة i تساوي 95 مثلاً ، فإن التعبير الجزئي (subexpression) الأول (على اليسار): `i = = 1` تكون قيمته false، وحيث أن المؤثر المنطقي && لا يعطي القيمة true إلا إذا كان معا معامَلاه كلاهما true ، فلذلك نستنتج أن القيمة النهائية للتعبير المركب هي false ولا داعي أصلاً للنظر في التعبير الجزئي الثاني `j > 2` لنرى أن كانت قيمته true أم false لأن أيا منهما لن تؤثر على القيمة النهائية للتعبير، أي يتوقف التقييم بمجرد إيجاد قيمة التعبير الأول في هذه الحالة ونصل سريعا [بدائرة قصيرة (short circrit)] إلى قيمة الصحة النهائية: false للتعبير المركب.

ب) في حالة العملية OR يتوقف التقييم من اليسار لليمين بمجرد أن نجد تعبيراً جزئياً قيمته true، لأن OR تعطي النتيجة true إن كان أحد المعامَلين أو كلاهما true.

مثلاً إذا أُعطينا التعبير `c <= d || e = = f`

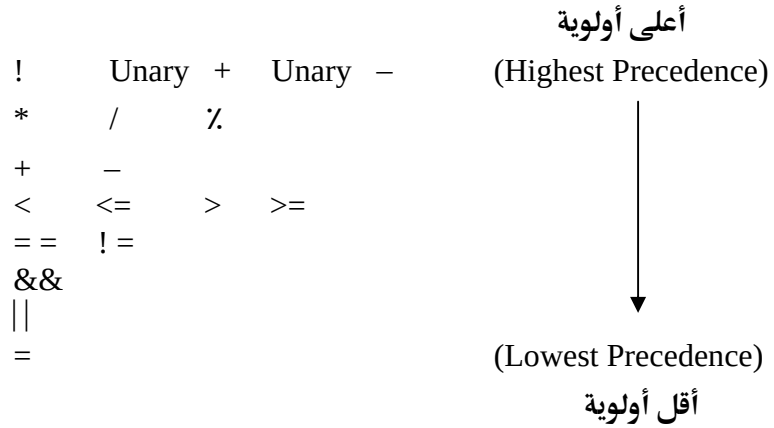
وكان التعبير الجزئي الأول true فإن التقييم يتوقف ويعطي الحاسب النتيجة الكلية true، دون إضاعة/هدر وقت (wasting time) في تقييم لا ضرورة له (unnecessary) للتعبير الجزئي الثاني.

ملاحظة: في بعض لغات البرمجة (غير لغة C++) يجب أولاً تقييم جميع التعابير الجزئية، أي عمل تقييم كامل (full evaluation) للتعبير المنطقي قبل تطبيق المؤثر المنطقي && أو المؤثر المنطقي || لإيجاد القيمة النهائية.

قاعدة الأولوية للمؤثرات (Precedence of Operators)

ذكرنا في الفصل السابق قواعد الأولوية التي تتحكم في تقييم التعابير الحسابية المركبة . وقواعد الأولوية في لغة C++ تتحكم أيضاً في تطبيق المؤثرات

العلاقية والمنطقية. وفيما يلي قائمة توضح ترتيب الأولويات (order of precedence) بالنسبة للمؤثرات الحسابية والعلاقية والمنطقية [وكذلك مؤثر الإسناد (assignment operator)]:



- المؤثرات التي تقع على السطر نفسه (مثل : ٪ ، / ، *) لها الأولوية نفسها .
- إذا احتوى تعبير على عدة مؤثرات لها الأولوية نفسها، فيتم التقييم – بالنسبة للمؤثرات الثنائية – من اليسار إلى اليمين. مثلاً التعبير a / b * c . يعني (a / b) * c وليس a / (b * c) . وأما بالنسبة للمؤثرات الأحادية (! , unary + , unary -) (unary operators) فيتم التقييم من اليمين إلى اليسار. مثلاً التعبير !! badData (وإن كان لا يستعمل إطلاقاً) يعني !(!badData) وليس (!!)badData والتي ليس لها معنى.

وملحق (ب): " قواعد الأولوية بالنسبة للمؤثرات " يعطي ترتيب الأولويات بالنسبة لجميع المؤثرات في لغة C++.

- يمكن استخدام الأقواس في أي تعبير لإلغاء الترتيب المعتاد حسب قواعد الأولوية ، حيث الأقواس لها الأولوية الأولى. وإن كنت في شك من ضرورة استخدام الأقواس فلا بأس من استخدامها لأن البرنامج المترجم (compiler) يتجاهل (disregards) الأقواس غير الضرورية، فإن كان

وجودها يزيد التعبير وضوحاً فاستخدمها . وبعض المبرمجين يفضلون كتابة قوسين إضافيين عند إسناد تعبير علاقي لمتغير منطقي ، وذلك للإيضاح مثل :

```
dataInvalid = (inputVal = = 0);
```

ففي مثل هذه العبارة لا نحتاج لكتابة القوسين ، حيث أن مؤثر الإسناد (=) له أقل أولوية بالنسبة لجميع المؤثرات التي ذكرناها، وبالتالي فيمكن كتابة العبارة السابقة هكذا:

```
dataInvalid = inputValue = = 0;
```

ولكن البعض يرى أن وجود القوسين يجعل العبارة أكثر وضوحاً.

وكما في الرياضيات ولغات البرمجة الأخرى ، فإن الأقواس في لغة C++ تستخدم أزواجاً أزواجا ، بمعنى أن أي قوس من أقواس الفتح (opening parentheses) يقابله قوس من أقواس الإغلاق (closing parentheses)

تنبيه: من الأخطاء التي يقع فيها بعض المبتدئين ترجمة التعابير الرياضية حرفياً إلى لغة C++، فلا تؤدي أحياناً إلى النتائج المطلوبة

(*) فمثلاً التعبير الرياضي

```
i = 3 or 4
```

قد يترجمه البعض إلى التعبير التالي بلغة C++:

```
i = = 3 || 4
```

وهذه ترجمة خاطئة لا تعطي النتائج المطلوبة ، فمثلاً إذا كانت قيمة i تساوي 8 فالمفروض أن تكون النتيجة النهائية false (لأن 8 لا تساوي 3 ولا تساوي 4) ولكن التعبير المكتوب بلغة C++ سيعطي النتيجة true دائماً !! ولا يعطي أي رسالة خطأ (error message) والسبب هو ما يلي:

التعبير الجزئي الأول : i = = 3 قد يكون true أو false.

التعبير الجزئي الثاني : 4 ليس منطقياً ، وحيث أنه قيمة عددية غير صفرية (nonzero) فإنه يُحوَّل (coerced) إلى القيمة true. وبالتالي فإن المؤثر || بين التعبيرين الجزئيين يجعل القيمة الكلية للتعبير true.

وأما الترجمة الصحيحة للتعبير الرياضي السابق إلى لغة C ++ فهي:

i = = 3 || i = = 4

حيث نستخدم المؤثر || (وكذلك المؤثر &&) للربط (to connect) بين تعبيرين منطقيين.

(*) وكمثال آخر للترجمة الخاطئة للشروط والتعابير إلى لغة C++ ترجمة التعبير:

midterm grade or final grade equals A

إلى

midtermGrade || finalGrade = = 'A'

ونلاحظ هنا أن المؤثر المنطقي || يربط بين قيمة رمزية (char value) "midtermGrade" وتعبير منطقي (finalGrade = = 'A')

وهذا التعبير بلغة C++ خاطئ (wrong) من الناحية المنطقية (logic) ولكنه ليس خاطئاً بالنسبة للمترجم (C++ compiler) ، لأن المؤثر || يمكنه أن يربط بين تعبيرين من أي نوعي بيانات، وبالتالي فمثل التعبير السابق لن يؤدي إلى ظهور رسالة خطأ تركيبي (syntax)، وسيستمر البرنامج في التنفيذ ، ولكنه لن يعمل بالطريقة التي نود تنفيذها . والترجمة السليمة (منطقيا) بلغة C++ للتعبير هي:

midtermGrade = = 'A' || finalGrade = = 'A'

حيث معامَلا (operands) المؤثر || تعبيران منطقيان.

(*) وكمثال ثالث للترجمة الخاطئة للتعابير والشروط الرياضية إلى لغة C++

ترجمة الشرط الرياضي:

$$12 < y < 24$$

(والذي يعني أن y تقع بين 24 , 12) إلى :

$$12 < y < 24$$

(أي كتابته بالأسلوب الرياضي نفسه)

وهذا التعبير سليم (legal) في لغة C++ ولكنه لا يؤدي إلى نتائج صحيحة دائماً ، فمثلاً إذا كانت $y = 50$ فإن نتيجة التعبير ستكون true، وذلك لأنه سيختبر أولاً الشرط $y > 12$ أي $50 > 12$ فيحصل على النتيجة true ، ثم يحول (coerces) الحاسب هذه النتيجة إلى 1 ليقارنها مع العدد 24، ونظراً لأن $1 < 24$ فتكون النتيجة النهائية true وهي طبعاً خاطئة !! وفي الواقع ستكون النتيجة النهائية دائماً true مهما كانت قيمة y، وذلك لأن المقارنة الأولى $y > 12$ ستكون إما true أو false حسب قيمة y، ثم يُحول الحاسب هذه النتيجة إلى 1 أو 0 (على الترتيب) ليقارنها مع العدد 24 ، ونظراً لأن كلا من 1 , 0 أصغر من 24 فستكون النتيجة النهائية دائماً true. وأما الترجمة الصحيحة إلى لغة C++ للتعبير/للشرط السابق فهي:

$$12 < y \;\&\&\; y < 24$$

المؤثرات العلاقية مع النوع ذي النقطة العائمة
(Relational Operators with Floating-Point Types)

حتى الآن ناقشنا المقارنات بين القيم التي من الأنواع int, char, string .

والآن ننظر إلى القيم التي من النوع float.

نحب أن ننبه أولاً إلى تجنب المقارنة بين القيم ذوات النقطة العائمة للتساوي (equality)، أي تجنب اختبار تساوي قيمتين float، وذلك لأن الحسابات

التي تجرى على الأعداد التي من النوع float غالباً ما تؤدي إلى أخطاء بسيطة في الأرقام العشرية أقصى اليمين (rightmost decimal places)، وهذه الأخطاء تجعل من النادر تساوي قيمتين float بالضبط . فمثلاً إذا كان x, onethird متغيرين من النوع float، وكتبنا عبارتي الإسناد التاليين:

```
oneThird = 1.0/3.0;
x = oneThird + oneThird + oneThird;
```

فإننا نتوقع أن يحتوي المتغير x على القيمة 1.0، ولكنه قد لا يحتوي فعلاً على هذه القيمة، وذلك لأن عبارة الإسناد الأولى تخزن قيمة تقريبية للمقدار 1/3 مثل 0.333333 في المتغير oneThird، ثم تخزن العبارة الثانية قيمة مثل 0.999999 في x. ولذلك فإذا سألنا الحاسب الآن أن يقارن بين قيمة x والقيمة 1.0 هل هما متساويتان ؟ فستكون النتيجة false، مع أننا نتوقع أن القيمتين متساويتان ولذلك ننتظر النتيجة true.

ولذلك فبدلاً من اختبار تساوي عددين float، نختبر قُرب تساويهما (near equality). أي نحسب الفارق (difference) بين عددين ونختبر ما إذا كان هذا الفارق أقل من قيمة معينة تمثل أقصى فارق مسموح به (maximum allowable difference)، أي تمثل أقصى تسامح (tolerance) . كأن نختبر مثلاً الفارق بين قيمتي متغيرين r, s من النوع float هكذا:

```
fabs ( r - s )  < 0.00001
```

حيث fabs هي دالة القيمة المطلقة العائمة (floating-point absolute value function) من مكتبة C++ القياسية. والتعبير fabs (r - s) يحسب القيمة المطلقة للفارق بين r,s. فإذا كانت هذه القيمة أقل من 0.00001 فإن العددين قريبان من بعضيهما قربا كافياً لأن يقال إنهما متساويان.

عبارة If (The If Statement)

بعد أن عرفنا كيف نكتب تعابير منطقية، نرى الآن كيف نستخدمها في تغيير المسار المعتاد للتحكم (normal flow of control) في برنامج. تعد عبارة If بِنْية التحكم الأساسية (fundamental control structure) التي تسمح بالتفرع (branching) في عملية انسياب التحكم (flow of control) . فعن طريقها يمكننا أن نسأل سؤالاً ونختار بين مسارين للتنفيذ: إذا تحقق شرط معين اخترنا أحد المسارين وإذا لم يتحقق اخترنا المسار الآخر. ووقت التنفيذ ينفذ الحاسب أحد المسارين فقط اعتماداً على نتيجة اختبار الشرط، ولكن علينا في البرنامج كتابة المسارين.

وفي لغة C++ توجد صيغتان لعبارة If:

(*) صيغة إذا كان .. فإن .. وإلا .. (If .. Then .. Else .. form)

(*) صيغة إذا كان .. فإن .. (If .. Then .. form)

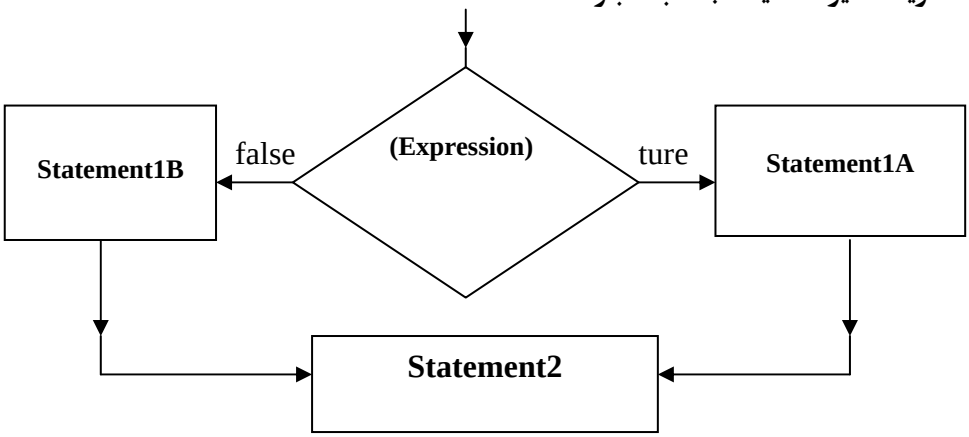
أولاً : صيغة .. If .. Then .. Else

وصورتها العامة الأكثر تفصيلاً هي :

```
if  (Expression)
   Statement1A
else
   Statement1B
```

حيث التعبير Expression الموجود بين قوسين يمكن أن يكون من أي نوع بسيط للبيانات (simple data type)، وهو عادة يكون تعبيراً منطقياً (logical/boolean). وإن لم يكن منطقياً فإن قيمته تُحوَّل ضمنياً (implicitly coerced) إلى النوع bool. ووقت التنفيذ يقيم الحاسب التعبير ، فإن كانت قيمته true فإن الحاسب ينفذ

العبارة Statement1A ، وإن كانت قيمته false فإنه ينفذ العبارة Statement1B. وعادة يُطلق على العبارة Statement1A عبارة then (then-clause) وعلى العبارة Statement1B عبارة else (else-clause). والشكل التالي (شكل ٣-١) يمثل خريطة سير العمليات بالنسبة لعبارة .. If .. Then .. Else



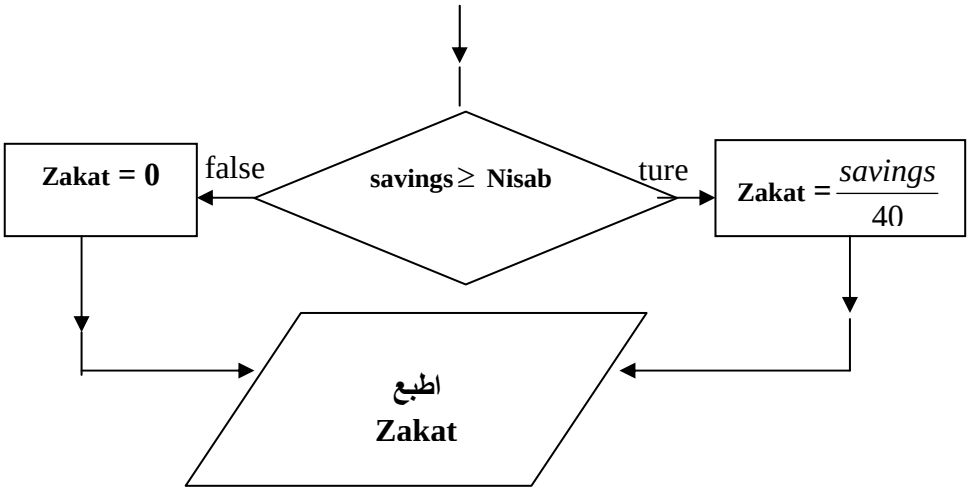
شكل ٣ - ١

If .. Then .. Else ..

وفي هذا الشكل عبارة statement2 هي العبارة التالية في البرنامج التي تلي عبارة If كلها. ولاحظ أن عبارة If في لغة C++ لا تستخدم كلمة then وتستخدم الكلمتين المحجوزتين if, else .

مثال ٣-٥

ترجم خريطة سير العمليات التالية (شكل ٣-٢) التي تحسب وتطبع قيمة زكاة المال zakat إلى قطعة برنامج بلغة C++ مستخدماً عبارة If .. Then .. Else ..



شكل ٣-٢

حساب زكاة المال

الحل:

```
if (savings >= Nisab)
    zakat = 0.025 * savings ;
else
    zakat = 0.0;
cout << zakat;
```

مثال ٣-٦ اكتب قطعة برنامج بلغة C++ تحسب وتطبع أجر عامل بحيث أنه إذا كان عدد ساعات العمل (hours) أقل من 40 فإن الأجر (pay) = عدد ساعات العمل مضروباً في معدل الأجر للساعة الواحدة (rate) ، وأما إذا زادت ساعات العمل عن 40 فيكون أجر أي ساعة إضافية (1.5 × rate)

الحل:

```
if (hours <= 40.0)
    pay  = rate * hours;
else
    pay = rate * (40.0 + (hours - 40.0) * 1.5);
cout << pay;
```

وتستخدم صيغة ..If .. Then .. Else لاختبار صحة المدخلات . فمثلاً قبل أن نطلب من الحاسب أن يقسم على قيمة من قيم البيانات يجب أن نتأكد أولاً أن هذه القيمة لا تساوي صفراً . [والحاسبات نفسها لا يمكنها أن تقسم مقداراً على الصفر، وإذا حاولنا ذلك فمعظم الحاسبات تتوقف halt عن تنفيذ البرنامج] . فإن كان المقسوم عليه (divisor) صفراً ، فيجب أن يطبع البرنامج رسالة خطأ، كما يوضح ذلك المثال التالي:

مثال ٣-٧ : اكتب قطعة برنامج لقسمة عدد dividend على آخر divisor مع إعطاء رسالة خطأ في حالة القسمة على صفر.

الحل:

```
if   (divisor ! = 0)
     result = dividend / divisor;
else
     cout << " Division by zero is not allowed." << endl;
```

مثال ٣-٨ :

نفرض أن myString متغير سلسلة رموز (string variable). اكتب قطعة برنامج لإيجاد موضع أول حدوث (first occurrence) للحرف A – إن ظهر – في السلسلة myString . وفي حالة عدم ظهور الحرف A تطبع القطعة رسالة تفيد ذلك.

الحل: في الفصل السابق ذكرنا أن طبقة سلاسل الرموز string class تحتوي على دالة من الدوال الأعضاء (member function) تُدعَى find تُعيد الموضع الذي وجد فيه العنصر (item) الذي نبحث عنه ، أو تعيد الثابت المسمى string :: npos إذا لم يوجد هذا العنصر . وبالتالي يمكننا كتابة القطعة التالية التي تطبع نتيجة البحث :

```
string  myString
string :: size_type pos;
        ⋮
pos = myString. find ('A') ;
if (pos = = string : :npos)
   cout << "No 'A' was found " << endl;
else
   cout << " An 'A' was found in position " << pos << endl;
```

ملاحظة: يلاحظ أن الصيغة العامة لعبارة If .. Then .. Else.. **لا تنص على وجود فاصلة منقوطة عند نهاية العبارة ، وفي الأمثلة الثلاثة الأخيرة : الفاصلة المنقوطة الموجودة عند نهاية العبارة إنما هي تابعة لعبارة** else (else clause)**، فعبارة الإسناد – في مثال أجر العامل – تنتهي بفاصلة منقوطة ، وعبارة الطباعة – في مثال القسمة على صفر وفي مثال البحث في سلسلة رموز – تنتهي بفاصلة منقوطة. أما عبارة** If **فليس لها فاصلة منقوطة خاصة بها عند نهايتها.**

القوالب/العبارات المركبة (Blocks/Compound Statements)

نفرض أننا في مثال القسمة على الصفر (مثال ٣–٧) نود في حالة ما إذا كان المقسوم عليه (divisor) **صفراً تنفيذ أمرين: الأول طباعة رسالة خطأ – كما عملنا في هذا المثال – والثاني إسناد قيمة خاصة مثل** 9999 **للمتغير** result. **أي أننا نحتاج إلى عبارتين في الفرع نفسه** (same branch)**. ولكن الصيغة العامة لعبارة** If **تشترط عبارة واحدة فقط ، فالمطلوب تحويل عبارة** else (else-clause) **إلى متتابعة من العبارات** (sequence of statements) **. ويمكن تنفيذ ذلك بسهولة إذا تذكرنا ما ذكرناه في الفصل السابق من أن المترجم** (compiler) **يعامل القالب / العبارة المركبة** (block/compound statement)

```
{
        ⋮
}
```

كما لو كانت عبارة واحدة فقط (single statement) . أي أننا لو وضعنا هذا الزوج من القوسين { } حول متتابعة من العبارات نريدها في أحد فرعي عبارة If، فإن هذه المتتابعة من العبارات تصبح قالباً وحيداً (a single block).

مثال ٣-٩:

أعد حل مثال ٣-٧ (قسمة عدد على آخر) بحيث أنه في حالة القسمة على صفر تقوم القطعة - بالإضافة إلى إعطاء رسالة خطأ - بتخزين القيمة 9999 في ناتج القسمة result.

الحل:

```
if  (dvisor != 0)
    result = dividend / divisor ;
else

{
    cout << " Division by zero is not allowed." << endl;
    result = 9999;
}
```

ويمكن استخدام القوالب (blocks) مع كل من فرعي ..If .. Then .. Else ، كما يوضح ذلك المثال التالي:

مثال ٣-١٠: اعد حل المثال السابق (مثال ٣-٩) بحيث أنه في حالة القسمة على عدد غير صفري فإن القطعة - بالإضافة إلى حساب ناتج القسمة - تعطى رسالة تفيد بأنه قد تم إجراء عملية القسمة.

الحل:

```
if  (dvisor != 0)
{
    result = dividend / divisor ;
    cout << " Division performed." << endl;
}
else
{
    cout << " Division by zero is not allowed." << endl;
```

```
    result = 9999;
}
```

وعند استخدام القوالب (blocks) في عبارة If فهناك قاعدة في بناء لغة C++ يجب تذكرها وهي: لا تضع إطلاقاً فاصلة منقوطة (semicolon) بعد القوس الأيمن (right brace) "}" لقالب. وتذكر أن الفواصل المنقوطة تستخدم فقط لإنهاء (to terminate) عبارات بسيطة مثل عبارات الإسناد، وعبارات الإدخال، وعبارات الإخراج.

صيغة If .. Then ... (The If .. Then.. form)

في بعض الأحيان نود إجراء عملية معينة إذا تحقق شرط معين ، وإذا لم يتحقق الشرط فلا نريد عمل أي شئ. يمكننا تنفيذ ذلك باستخدام صيغة If .. Then .. Else .. بأن نترك فرع else (branch) فارغاً (empty) باستعمال العبارة الخالية (null statement) ،كأن نكتب مثلاً:

```
if   ( a <= b)
       c = 20;
else
       ;
```

والأفضل أن نحذف جزء else كله، والعبارة الناتجة هي صيغة If .. then .. لعبارة If، وصيغتها هي:

```
if (Expression)
   Statement1
```

والمثال التالي تطبيق على هذه الصيغة:

مثال ٣-١١:

نفرض أنك أعطيت عمر شخص age بالسنوات . اكتب – باستخدام صيغة If .. Then .. – قطعة برنامج تطبع رسالة توضح ما إذا كان يحق للشخص أن يشارك

في الانتخابات "voter" أو لا يحق له "Not an eligible voter" . علماً بأنه يحق للشخص المشاركة في الانتخابات إذا بلغ عمره الثامنة عشر (18).

الحل:

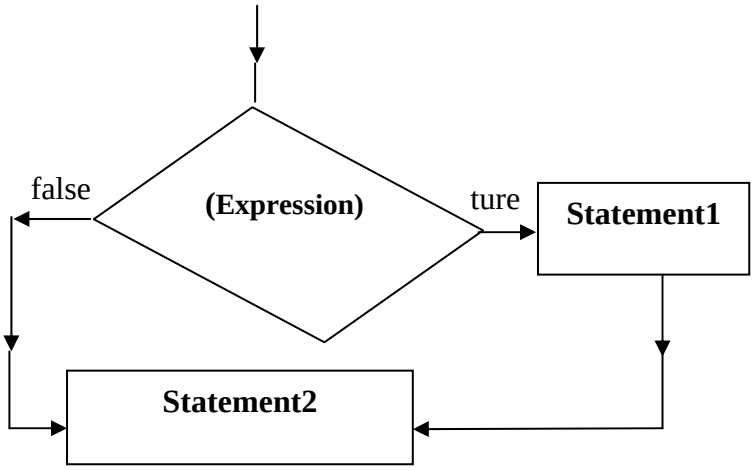
```
if   (age < 18)
      cout << "Not an eligible   ";
cout <<  " voter." << endl;
```

ففي حالة ما إذا كان العمر أقل من 18 – أي أن الشرط age < 18 متحقق – فإن البرنامج سينفذ نتيجة الشرط وهي طباعة الرسالة " Not an eligible " ، ثم ينتقل إلى الأمر التالي، وهو طباعة الرسالة "voter" بجانب الرسالة السابقة (على السطر نفسه) ، فتكون الرسالة الكلية المطبوعة هي:

Not an eligible voter.

وأما إن كان العمر أكبر من أو يساوي 18 – أي أن الشرط age < 18 غير متحقق – فإن البرنامج يهمل نتيجة الشرط – أي يهمل عبارة الطباعة الأولى – وينتقل إلى العبارة التالية وهي طباعة الرسالة "voter.".

والشكل التالي (شكل ٣–٣) يوضح خريطة سير العمليات لصيغة .. If .. Then :



شكل ٣–٣

If .. Then ..

وكما ذكرنا أثناء مناقشة صيغة .. If .. Then .. Else أن أيا من فرعي If يمكن أن يكون قالباً (block) ، فكذلك فرع If الوحيد (one branch) في صيغة If .. Then يمكن أن يكون قالباً. والمثال التالي يوضح ذلك.

مثال ٣ – ١٢:

في أحد اختبارات أسئلة صح/خطأ (True/False Questions) تعطى درجة للإجابة الصحيحة وتخصم درجة للإجابة الخاطئة ، ثم تحسب النتيجة result بالعلاقة :

result = NumberCorrect - NumberWrong

والتي تعني أن النتيجة تساوي (عدد الإجابة الصحيحة – عدد الإجابات الخاطئة)، وتكون الدرجة النهائية score هي النتيجة result إن كانت هذه النتيجة غير سالبة، أما إن كانت النتيجة result سالبة فإن الدرجة النهائية score تساوي صفراً.

اكتب قطعة برنامج تعطي الدرجة النهائية score إذا علم عدد الإجابات الصحيحة NumberCorrect وعدد الإجابات الخاطئة NumberWrong.

وفي حالة ما إذا كان عدد الإجابات الخاطئة أكبر من عدد الإجابات الصحيحة فإن البرنامج يعطي رسالة تفيد ذلك.

الحل:

```
result = NumberCorrect - NumberWrong;
if (result < 0.0)
{
   cout << " Wrong answers are more than correct ones." << endl;
   result = 0.0;
}
score = result;
```

ونحب أن ننبه هنا لوجود القوسين { } في القطعة السابقة لتكوين القالب.

ولتوضيح أهميتهما نفرض أننا حذفناهما ، بمعنى أننا كتبنا القطعة هكذا:

```
result = NumberCorrect - NumberWrong;
if (result < 0)
   cout << " Wrong answers are more than correct ones." << endl;
   result = 0.0;
score = result;
```

في حالة إذا كانت النتيجة result سالبة فإن البرنامج ينفذ نتيجة الشرط [أي عبارة (clause) then] وهي عبارة طباعة رسالة ، ثم ينتقل إلى العبارة التالية – التي تلي عبارة if – وهي عبارة الإسناد result = 0.0 ، ثم ينتقل إلى العبارة الأخيرة التي تُسند قيمة result للمتغير score.

وأما في حالة ما إذا كانت result ≥ 0 أي أن الشرط في عبارة if غير متحقق فإن البرنامج يهمل عبارة then (نتيجة الشرط) ، أي يهمل عبارة الطباعة، وينتقل إلى العبارة التالية – التي تلي عبارة if – وهي عبارة الإسناد result = 0.0 رغم أنها مكتوبة مع إزاحة لليمين تحت عبارة الطباعة ، ولكن هذه المسافات ليس لها أي أهمية، وبالتالي فإن العبارة التالية الأخيرة تسند قيمة result(صفراً) للمتغير score. معنى هذا أن قيمة الدرجة النهائية score ستكون دائماً صفراً !! مهما كان عدد الإجابات الصحيحة وعدد الإجابات الخاطئة، وهذا طبعاً حل خاطئ. لذلك يجب الانتباه إلى أهمية القوسين { } عند كتابة القوالب (blocks).

والمثال التالي يوضح خطأ عاماً (common mistake) آخر .

مثال ٣ –١٣:

اكتب قطعة برنامج تقرأ قيمة عدد صحيح n ، وتطبع الرسالة "n equals 3" إذا كانت قيمة n تساوي 3 ، بينما تطبع الرسالة " n doesn't equal 3" لأي قيمة أخرى للعدد n.

الحل :

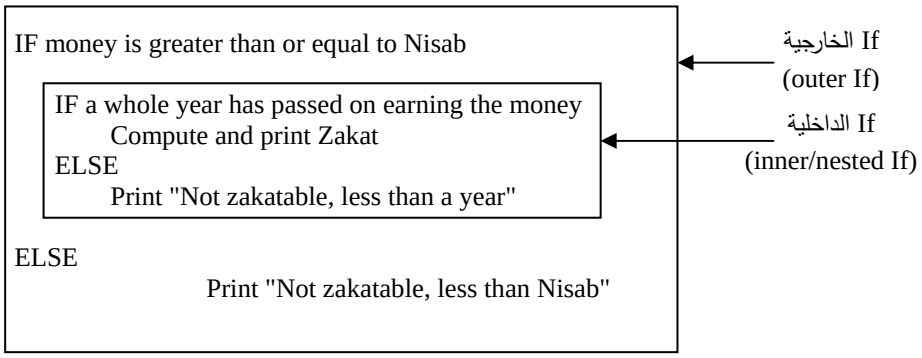
```
cin >> n;
if (n = = 3)
        cout << " n equals 3" ;
else
         cout << " n doesn't equal 3 " ;
```

والآن ماذا يحدث إذا أخطأنا في كتابة الشرط (n = = 3) وكتبناه كعادتنا في كتابة العلاقات الرياضية (n = 3). النتيجة أنه مهما كانت قيمة n المدخلة فإن البرنامج سيطبع الرسالة n equals 3 . والسبب أن التعبير n = 3 ليس تعبيراً منطقياً ، وإنما يطلق عليه تعبير إسناد (assignment expression) [وهذا مختلف عن عبارة الإسناد (assignment statement) التي تكتب كعبارة مستقلة منتهية بفاصلة منقوطة] . وتعبير الإسناد له قيمة (وقيمة التعبير السابق تساوي 3) وله تأثير جانبي (side effect) (وهو هنا تخزين القيمة 3 في n). وفي عبارة if يجد الحاسب أن قيمة التعبير تساوي 3، ونظراً لأنها قيمة غير صفرية فإنها تُحوَّل قسراً إلى القيمة المنطقية true، وبالتالي فإن عبارة then (clause) تُنفذ، أي أن نتيجة الشرط تنفذ ، مهما كانت قيمة n، وبالتالي فإن الرسالة "n equals 3" تطبع في جميع الأحوال . والأسوأ من هذا هو تأثير تعبير الإسناد في تخزين القيمة 3 في n ماحياً (destroying) أي قيمة مخزونة هناك. لذلك نؤكد على ضرورة استخدام المؤثر العلاقي " = = " وليس " = " .

عبارات If المتداخلة (Nested If Statements)

ليس هناك أي قيود (restrictions) أو شروط على نوعية العبارات الموجودة في عبارة If (في عبارة then أو في عبارة else) . وبالتالي فيجوز وجود عبارة If داخل (within) عبارة If. بل يجوز أيضاً وجود عبارة If داخل عبارة If داخل عبارة If. ويمكن تعميم هذا لعدد أكبر من عبارات If المتداخلة ، ولكن يصعب حينئذ تتبع هذه البنية (structure) ونفقد ميزة سهولة قراءتها.

وحينما نضع عبارة If داخل عبارة If يقال إننا نكوِّن بنية تحكم متداخلة (nested control structure)، وفيما يلي مثال لهذه البنية مكتوب بالشفرة الزائفة (pseudocode) :



وعموماً أي مسألة تشتمل على اختيار بين عدة مسارات / فروع (multiway branch/alternative courses of action) **– أكثر من اثنين– يمكن أن تكتب باستخدام عبارات** If **المتداخلة. فمثلاً لطباعة اسم شهر إذا عُلم ترتيبه/ رقمه يمكن أن نستخدم إما متتابعة** (sequence) **من عبارات** If **غير المتداخلة** (unnested)**، أو أن نستخدم بنية** If **المتداخلة** (nested If structure) **، كما يوضح ذلك المثال التالي:**

مثال ٣-١٤:

نفرض أن month **متغير صحيح يمثل ترتيب/رقم شهر من الشهور الإسلامية الاثنى عشر . اكتب قطعة برنامج لطباعة اسم الشهر إذا علم رقمه:**

(أ) باستخدام متتابعة من عبارات If **غير المتداخلة.**

(ب) باستخدام بنية If **المتداخلة المكافئة.**

أي الحلين أفضل ؟ ولماذا ؟

الحل: (أ) باستخدام عبارات If غير المتداخلة:

```
if  (month = = 1)
    cout << " AlMoharram " ;
if  (month = = 2)
    cout << " Safar " ;
if  (month = = 3)
    cout << " Rabee'1 " ;
    :
if  (month = = 12)
    cout << " ThulHijja " ;
```

(ب) باستخدام بنية If المتداخلة:

```
if  (month = = 1)
    cout << " AlMoharram " ;
else
        if  (month = = 2)            // nested if
            cout << " Safar " ;
        else
                if  (month = = 3)        // nested if
                    cout << " Rabee'1 " ;
                else
                        if  (month = = 4)        // nested if
                            :
```

هذا الحل في (ب) باستخدام بنية If المتداخلة أكفأ من الحل السابق في (أ) لأنه يجرى عدداً أقل من المقارنات (comparisons) ، بينما الحل في (أ) الذي يستخدم متتابعة من عبارات If المستقلة (independent) يختبر دائماً الشروط الاثنى عشر جميعها شرطاً شرطاً ، حتى لو تحقق الشرط الأول. وفي المقابل فإن الحل (ب) باستخدام بنية If المتداخلة يتخطى (skips) جميع المقارنات المتبقية بعد تحقق أحد الشروط أي بعد اختيار (selection) أحد المسارات. والخوارزميات منخفضة الكفاءة (inefficient algorithms) يمكن أن تضيّع/تهدر ساعات من وقت الحاسوب ، ولكن عموماً تحسين الكفاءة يجب ألا يكون على حساب سهولة فهم البرنامج وسهولة قراءته، ويمكن التضحية ببعض الكفاءة في سبيل سهولة تتبع البرنامج وفهمه.

لاحظ في المثال السابق عند الحل باستخدام بنية If المتداخلة كيف أن ضبط المسافات والفراغات (indentation) بالنسبة لعبارات then وعبارات else تسببت في تحريك العبارات إلى اليمين باستمرار. ويمكننا بدلاً من ذلك استخدام صيغة/أسلوب / طريقة (style) خاصة في ضبط المسافات مع عبارات .. If .. Then .. Else المتداخلة بدرجة كبيرة (deeply nested) للدلالة على أن كل المطلوب من هذه البنية المعقدة هو مجرد اختيار واحد من مجموعة اختيارات (set of

(general alternatives) وهـذه الصيغـة العامـة للتفـرع إلى طـرق متعـددة (multiway branch) تعرف ببنية تحكم If .. Then .. Else.. If . والصيغة تبدو كما يلي:

```
if  (month = = 1)
    cout << " AlMoharram " ;
else  if (month = = 2)            // nested if
          cout << " Safar " ;
else if   (month = = 3)           // nested if
           cout << " Rabee'1 " ;
else if  (month = = 4)            // nested if
     :
else
         cout << "ThulHijja" ;
```

وهذه الطريقة في الكتابة تمنع تحرك العبارات باستمرار لليمين. ولكن الأهم من هذا أنها توضح أننا نختار واحداً من ١٢ اختيار بناء على قيمة المتغير month.

وهناك فارق مهم بين بنية If المتداخلة ومتتابعة عبارات If وهو أن بنية If المتداخلة تختار واحداً فقط من الاختيارات المتاحة، بينما في متتابعة عبارات If يمكن للمتتابعة أن تختار أكثر من اختيار واحد .

مثال ٣-١٥:

الجدول التالي يعطي النشاط (activity) الرياضي المناسب للمدى المقابل من درجات الحرارة:

النشاط Activity	درجة الحرارة Temperature
السباحة swimming	temperature > 85
التنس tennis	70 < temperature ≤ 85
الجولف golf	32 < temperature ≤ 70
التزلج skiing	0 < temperature ≤ 32
تنس الطاولة ping-pong	temperature ≤ 0

اكتب قطعة برنامج تطبع النشاط المقترح بناء على قيمة درجة الحرارة temperature .

الحل: نلاحظ من الجدول المعطى أن قيم المدى متتابعة (consecutive ranges of values)، وبالتالي فليس هناك درجة حرارة يقابلها أكثر من نشاط رياضي واحد، بمعنى أننا فعلياً سنختار نشاطاً واحداً فقط من هذه الأنشطة المتاحة، وبالتالي فالصيغة الملائمة للتطبيق هي بنية If المتداخلة. أما إذا ترجمنا سطور الجدول السابق إلى متتابعة من عبارات If، فسنلاحظ أن شروط هذه العبارات غير مستقلة عن بعضها البعض (interdependent)، بمعنى أنه إذا كان أحدها متحققاً – وبالتالي ستنفذ عبارته – فستكون جميع الشروط الأخرى غير متحققة – وبالتالي لن تنفذ أي عبارة أخرى . فالنتيجة هي أننا نختار اختياراً واحداً فقط من عدة اختيارات أمامنا. ولذلك فالأفضل استخدام بنية If المتداخلة.

ويلاحظ أنه عند استخدام If المتداخلة في حالة القيم المتتابعة للمدى – كما في هذا المثال – يمكننا جعل البرنامج أكثر كفاءة (more efficient) بأن نقتصر في اختبار وقوع المتغير (درجة الحرارة هنا) في مدى معين على مقارنة المتغير بأحد حَدَّى (limits) المدي فقط وليس بكل منهما. فمثلاً في المدى $32 < \text{temperature} \leq 70$ يكفي أن نقارن درجة الحرارة مع 32 فقط، أي نختبر ما إذا كان (temperature > 32) فقط، والسبب في ذلك هو أنه إذا بدأنا اختباراتنا لدرجة الحرارة من الشرط الأول [شرط السطر الأول] (temperature > 85) ثم شرط السطر التالي فالتالي وهكذا حتى وصلنا لسطر معين فمعنى ذلك أن شروط جميع السطور السابقة لم تتحقق ، فمثلاً إذا وصلنا للشرط $32 < \text{temperature} \leq 70$ فمعنى ذلك أن الشرطين السابقين

temperature > 85
$70 < \text{temperature} \leq 85$

لم يتحققـــا، وبالتالي فدرجـة الحرارة ليست أكبـــــر من 85 (من الشرط الأول) ثم هي أيضاً ليست أكبر من 70 (من الشرط الثاني) ، وبناء عليه فعند اختبار الشرط 70 ≥ temperature > 32 لا داعي لمقارنة درجة الحرارة مع الحد الأعلى 70 (upper limit) لأن درجة الحرارة قطعاً ستحقق 70 ≥ temperature ويكفي مقارنتها مع الحد الأسفل (lower limit) 32 :

32 < temperature

وبالاختصار نقول إنه يكفي مقارنة درجة الحرارة مع القيمة الصغرى (lowest value) في كل مدى [ولكن طبعاً بشرط ترتيب قيم المدى بحيث تكون متتابعة (consecutive) ابتداءً] ، كما يوضح ذلك الحل التالي:

```
cout << "The recommended activity is " ;
if  (temperature > 85)
    cout << "swimming." << endl;
else if (temperature > 70 )
     cout << "tennis." << endl;
else if (temperature > 32 )
   cout << "golf." << endl;
else if (temperature > 0 )
  cout << "skiing." << endl;
else
    cout << "ping-pong." << endl;
```

وهكذا يختبر كل فرع قيمة درجة الحرارة بمقارنتها مع القيمة التي تقع أسفل المدى إلى أن نصل إلى else النهائية وهي تتولى جميع الاحتمالات المتبقية (remaining possibilities) .

ويلاحظ أنه إذا لم تكن قيم المدى متعاقبة (حيث قيم كل مدى تعقب قيم المدى السابق) فيجب عندئذ اختبار قيم البيانات بمقارنتها مع كل من القيمة العظمى والقيمة الصغرى في كل مدى ، مع استمرارنا في استخدام بنية .. If Then .. Else .. If لأنها أفضل بنية لاختيار فرع واحد (single branch) من عدة احتمالات، ويمكننا أيضاً ترتيب قيم المدى ترتيباً تعاقبياً (arrange the ranges in consecutive order) حتى يسهل على القارئ متابعتها. ولكن لا يمكننا تقليل عدد

المقارنات عندما تكون هناك فراغات / فجوات (gaps) بين مدى ومدى آخر (between the ranges) .

وفيما يلي مثال آخر على استخدام بنية If المتداخلة بكفاءة في حالة التعاقب عند وجود أكثر من مدى واحد.

مثال ٣ -١٦

اكتب قطعة برنامج (program segment) لحساب زكاة الغنم zakatSheep إذا علم عدد الأغنام numberSheep [كل من numberSheep, zakatSheep عدد صحيح (integer)] بفرض أن زكاة الغنم تحسب بالقاعدة المبينة بالجدول المعطى في المسألة رقم ١-١ (بالفصل الأول) .

الحل:

```
if  (numberSheep < 40)
    zakatSheep = 0;
else if   (numberSheep <= 120)
          zakatSheep = 1;
else if   (numberSheep <= 200)
          zakatSheep = 2;
else if   (numberSheep <= 300)
          zakatSheep = 3;
Else
          zakatSheep = numberSheep / 100;
```

لاحظ مرة أخرى مقارنة عدد الأغنام بأحد طرفي المدى فقط (القيمة العظمى) . وعندما نصل إلى else الأخيرة نضمن أن عدد الأغنام أكبر من 300 وبالتالي فزكاتها شاة لكل مائة، والقسمة الصحيحة numberSheep / 100 تعطي النتيجة المطلوبة لأنها تحذف أي كسور ناتجة في عملية القسمة هذه.

وفيما يلي حل آخر – له كفاءة الحل السابق نفسها – يقارن عدد الأغنام بالطرف الآخر للمدى (القيمة الصغرى) :

```
if  (number Sheep > 300)
```

```
    zakatSheep = numberSheep / 100;
else if   (numberSheep > 200)
          zakatSheep =  3;
else if   (numberSheep > 120)
          zakatSheep =  2;
else if   (numberSheep >= 40)
          zakatSheep =  1;
else
          zakatSheep =  0;
```

وهذا الحل والحل السابق كلاهما أفضل من حل يقارن عدد الأغنام بالقيمة العظمى والقيمة الصغرى لكل مدى ، نظراً لكثرة المقارنات في هذا الحل الأخير دون أي حاجة لهذه المقارنات الزائدة.

else المتدلية / التابعة (The Dangling else)

عند استخدام عبارات If المتداخلة (if داخل if) يحدث أحياناً التباس/إبهام بالنسبة لـ else معينة: تَتْبع أيّ if؟ (هل else هذه تتبع if الخارجية أم الداخلية ؟) . والقاعدة المتبعة بالنسبة لمترجم C++ هي : في حالة عدم وجود أقواس فإن أي else تتبع دائماً أقرب if سابقة لها ليس لها else تتبعها.

مثال ٣-١٧:

تنص إحدى لوائح تقديرات الطلاب على ما يلي:

إذا كانت الدرجة المتوسطة average للطالب أقل من 60 فتطبع له الرسالة "Failing" . أما إن كانت 60 على الأقل (at least) ولكنها أقل من 70 فتطبع له الرسالة "Passing but marginal". وإن كانت 70 أو أكثر فلا تطبع له أي رسالة.

أي قطعتي البرنامج (program segments) التاليتين تعد ترجمة صحيحة بلغة C++ للائحة السابقة ؟ وأيهما تعد خاطئة؟ ولماذا ؟

(أ)

```
if  (average >=  60.0)
    if  (average < 70.0)
        cout << "Passing but marginal" ;
else
    cout << "Failing" ;
```

(ب)

```
if   (average <  70.0)
     if  (average < 60.0)
          cout << "Failing" ;
     else
          cout << " Passing but marginal " ;
```

الحل : القطعة (أ) خاطئة ، وذلك لأن else**، وإن كانت مكتوبة أسفل** if **الخارجية إلا أنها تتبع** if **الداخلية – بناءً على قاعدة مترجم** C++ **المذكورة سابقاً – وهذا يعني أنه إذا كانت الدرجة المتوسطة** average **أكبر من أو تساوي** 70.0 **فستُطبع للطالب الرسالة** "Failing" **!! وهذا طبعاً خطأ.**

أما القطعة (ب) – والتي تستخدم صيغة .. If .. Then .. Else **داخل** (nested within) **صيغة** .. If .. Then **– فهي ترجمة صحيحة للائحة.**

ملاحظة ١ :

مَنْ كَتَبَ القطعة (أ) عند ترجمة اللائحة كان يود إلحاق فرع else **بعبارة** if **الخارجية (وليس الداخلية) كي تطبع الرسالة** "Failing" **إذا كانت الدرجة المتوسطة أقل من** 60.0**، ولذلك حرَّك** else **لليسار لتكون أسفل** if **الخارجية، ولكن التحريك وضبط المسافات لا يؤثر على تنفيذ العبارات ، والمترجم هنا سيلحقها – كما ذكرنا – بعبارة** if **الداخلية. ويطلق على** else **التي تظهر بعد** .. If .. Then **المتداخلة** (.. If .. Then : (nested else **المتدلية/التابعة** (dangling else)**، وهي منطقياً** (logically) **لا تتبع** (belong) If **المتداخلة ، وإنما المترجم** (compiler) **هو الذي يلحقها** (attaches it) **بها.**

ملاحظة ٢:

كي نلحق else في القطعة (أ) بعبارة if الخارجية (if الأولى) وليس if الداخلية (if الثانية) يمكننا تحويل عبارة then الخارجية (outer then - clause) إلى قالب (block) هكذا:

```
if  (average) >=  60.0)          // correct version
{
   if   (average < 70.0)
        cout << "Passing but marginal";
}
else
        cout << "Failing";
```

فالقوسان { } يشيران إلى أن عبارة if الداخلية كاملة (complete)، وبالتالي فإن else يجب أن تتبع if الخارجية.

اختبار حالة سيل مدخلات / مخرجات
(Testing the State of an I/O Stream)

تناولنا في الفصل السابق مفهوم سيل المدخلات وسيل المخرجات في لغة C++، وعَرَّفنا الطبقات (classes):

istream, ostream, ifstream, ofstream

وذكرنا أن أيا من الأسباب التالية يمكن أن يؤدي إلى دخول سيل مدخلات (input stream) حالة الفشل (fail state):

- بيانات مدخلة غير سليمة (غير صحيحة) (invalid input data)
- محاولة القراءة بعد نهاية ملف.
- محاولة فتح ملف غير موجود للإدخال.

وهناك طريقة في لغة C++ يمكننا بها معرفة ما إذا كان سيل ما في حالة الفشل أم لا، وهي أن نستخدم ببساطة اسم الهدف السيل (stream object) [مثل cin] في تعبير منطقي (logical expression) كما لو كان متغيراً منطقياً (Boolean variable)

```
if  (cin)
    ⋮
if  (   ! inFile  )
    ⋮
```

وعندما نقوم بذلك يقال إننا نختبر حالة السيل. ونتيجة الاختبار هي إما true [وهذا يعني أن آخر عملية إدخال / إخراج (last I/O operation) على هذا السيل كانت ناجحة (successful)] أو false [وهذا يعني أن آخر عملية إدخال / إخراج فشلت (failed)] .

أي أننا ننظر إلى الهدف السيل في تعبير منطقي كأنه متغير منطقي قيمته true ومعناها أن حالة السيل عادية/طبيعية/مَرْضية OK ، أو قيمته false ومعناها أن حالته ليست مَرْضية.

وعادة نكتب التعبير المنطقي بطريقة تعتمد على المهمة التي ستقوم بها عبارة then فمثلاً العبارة ستنفذ عبارة then إذا كانت أخر عملية إدخال / إخراج على inFile قد نجحت . أما العبارة

```
if  (   ! inFile  )
    ⋮
```

فستنفذ عبارة then إذا كان inFile في حالة الفشل. [تذكر أنه إذا أصبح سيل في حالة الفشل فسيظل هكذا. وأي عمليات إدخال/إخراج (I/O) تالية على هذا السيل ستكون عمليات فارغة (null operations)].

مثال ٣-١٨:

البرنامج التالي يبدأ بمحاولة فتح ملف على القرص (disk file) mydata.dat للإدخال. ثم يختبر ما إذا كانت محاولة الفتح قد نجحت أم لا:. فإن كانت فإن البرنامج يقرأ بيانات من الملف، ويجري بعض الحسابات ويختتم بتنفيذ العبارة return 0 ، وإن كانت الأخرى – أي لم نستطع فتح ملف الإدخال – فإن البرنامج يطبع رسالة خطأ (error message) للمستخدم، وينتهي معيداً القيمة 1.

الحل:

```
# include <iostream>
# include <fstream>              // For file I/O

using namespace std;
int main ( )
{
        int             height ;
        int             width ;
        ifstream        inFile ;

        inFile.open ("mydata.dat")          // Attempt to open input file
        if  (   ! inFile  )                 // was it opened?
        {
           cout << "Can't open the input file.";  // No--print message
            return 1 ;                              // Terminate program
        }
        inFile >> height >> width;
        ⋮
        return 0;
}
```

ملاحظات :

- **إذا نجحت محاولة فتح الملف فإن قيمة التعبير** inFile ! **في عبارة** if **ستكون** false **وبالتالي سيتم تخطي** (skipping) **عبارة** then.
- **قيمة الدالة التي تعيدها** main **يطلق عليها حالة الخروج** (exit status)
- **في حالة عدم القدرة على فتح ملف الإدخال** (input file) **فعند العودة من الدالة** open **يكون السيل** inFile **في حالة الفشل. وفي عبارة** If **تكون قيمة التعبير** inFile! **هي** true. **وبالتالي يتم تنفيذ عبارة** then، **فيطبع البرنامج رسالة الخطأ للمستخدم ، وينتهي بإعادة حالة الخروج** 1 **ليُعْلِم نظام التشغيل** (operating system) **بالنهاية غير الطبيعية/ غير العادية للبرنامج.**
- **القيمة** 1 **لحالة الخروج اختيارية. وأحياناً يقوم مبرمجو النظم** (system programmers) **باستخدام قيم مختلفة عن** 1 **متعددة في البرنامج نفسه**

للدلالة على أسباب مختلفة متعددة لإيقاف وإنهاء (termination) البرنامج، ولكن المعظم يستخدم القيمة 1 .

- عندما تفتح ملف بيانات (data file) للإدخال فدائماً تأكد من اختبار حالة السيل (stream state) قبل الاستمرار في البرنامج.فإذا نسيت ذلك، ولم يستطع الحاسب فتح الملف ، فسيستمر البرنامج في التنفيذ، وسيتجاهل (ignores) أي عمليات إدخال (input operations) على الملف.

مثال ٣-١٩:

اكتب برنامجاً لقراءة الرقم التعريفي (ID number) (من النوع long) [*] لطالب، وثلاث درجات اختبارية (test grades) (من النوع int). ثم يحسب البرنامج الدرجة المتوسطة average لهذه الدرجات الثلاث ، ويطبع الرقم التعريفي للطالب ودرجته المتوسطة ورسالة تفيد ما إذا كان الطالب:

- ناجحاً passing: إن كانت درجته المتوسطة 60 درجة أو أكثر. ولكن إذا كانت هذه الدرجة المتوسطة أقل من 70 فتوضح الرسالة أن الطالب قريب من حدود النجاح marginal.
- غير ناجح failing: إن كانت درجته المتوسطة أقل من 60.

ملاحظة: في حالة ما إذا كانت أي من الدرجات الاختبارية سالبة ، فإن البرنامج يطبع رسالة خطأ (error message).

الحل:

مدخلات البرنامج

- الرقم التعريفي للطالب : studentID
- ثلاث درجات اختبارية: test1, test2, test3

(*) long تعني long integer. ويلاحظ أنه في بعض الحاسبات الشخصية (personal computers) تكون أكبر قيمة صحيحة int هي 32767 بينما تكون قيمة ID أكبر من هذه القيمة ، ولذلك نختار النوع long.

مخرجات البرنامج:

- **رسالة حث** (prompt) **للإدخال**
- **قيم المدخلات [طباعة الصدى** (echo print)]
- **الدرجة المتوسطة** average
- **رسالة النجاح /الرسوب** (passing / failing message) **(مع الإشارة إلى حالة القرب من حافة النجاح في حالة تحققها)**
- **رسالة خطأ (في حالة وجود خطأ في الدرجات)**

وفيما يلي البرنامج المطلوب

```
//*****************************************************************
 //Notices program
 //This program determines (1) a student's average based on three
 //test scores and (2) the student's passing/failing status
//*****************************************************************
#include <iostream>
#include <iomanip>    // For setprecision()

using namespace std;
int main( )
}
   float average;         // Average of three test scores
   long  studentID;       // Student's identification number
   int   test1;           // Score for first test
   int   test2;           // Score for second test
   int   test3;           // Score for third test
   bool  dataOK;          // True if data is correct

   cout << fixed << showpoint;            // Set up floating pt.
                                          // output format
                                  //  Get data
   cout << "Enter a Student ID number and three test scores":
        << endl;
```

```
cin >> studentID >> test1 >> test2 >> test3;
cout << " Student number: " << studentID << "  Test Scores" :
      <<  test1 << ", " << test2 << ", " << test3 << endl;
// Test data

if (test1 < 0 || test2 < 0 || test3 < 0)
      dataOK = false;
else
      dataOK = true;

if (dataOK)
{
          // Calculate average

          average = float(test1 + test2 + test3) / 3.0;

          // Print message

          cout << "Average score is"
                << setprecision(2) << average  << "- -";
          if (average >= 60.0)
          {
                cout << "Passing";              // Student is passing
                if (average < 70.0)
                   cout << " but marginal";      // But marginal
                cout << '.' << endl;
          }
          else                              // Student is failing
                cout << "Failing." << endl;
          }
          else                                 // Invalid data
     cout << "Invalid Data:  Score(s) less than zero." << endl;
          return 0;
}
```

وفيما يلي مثالان لمخرجات هذا المخرجات:

المثال الأول لمخرجات البرنامج **Sample Program Run (1)**

```
Enter a student ID number and three test scores;
9483681  73  62  68
Student  Number : 9483681  Test Scores:  73,  62,  68
Average score is 67.67--Passing but marginal.
```

المثال الثاني لمخرجات البرنامج **Sample Program Run (2)**
(invalid data)

```
Enter a student ID number and three test scores;
9483681  73  -10  62
Student  Number : 9483681  Test Scores:  73,  -10,  62
Invalid Data : Score (s) less than zero.
```

ملاحظات:

- **في هذا البرنامج استخدمنا المتغير المنطقي** dataOK **لاختبار صحة/سلامة البيانات، وقيمته تكون** true **إذا كانت البيانات جميعها صحيحة** (correct).
- **كذلك استخدمنا بنية** if **المتداخلة وهي سهلة الفهم وان لم تكن ذات كفاءة عالية.**
- **من الممكن اختصار شرط اختبار صحة البيانات هكذا:**

```
dataOK = ! (test1 < 0 || test2 < 0 || test3 < 0 ) ;
```

وباستخدام قانون دي مورجان (DeMorgan's law) **يمكن كتابة هذه العبارة هكذا:**

```
dataOK =  (test1 >= 0  && test2 >= 0  && test3 >= 0 ) ;
```

ويمكن اختصار هذا أيضاً عن طريق حذف المتغير dataOK **واستخدام الشرط:**

```
if =  (test1 >= 0  && test2 >= 0  && test3 >= 0 ) ;
        ⋮
```

بدلاً من الشرط

```
if (dataOK)
⋮
```

الأخطاء التركيبية والأخطاء المعنوية / المنطقية
Syntactic Errors and Semantic / logic Errors

بعد كتابة البرنامج وإعداد البيانات الاختبارية (test data) يكون البرنامج جاهزاً للترجمة (compilation). ويقوم المترجم (compiler) بمسئوليتين: الإخبار (reporting) عن أي أخطاء، و(إذا لم تكن هناك أي أخطاء) ترجمة (translating) البرنامج إلى الشفرة/الكود الهدف (object code) .

والأخطاء يمكن أن تكون تركيبية (syntactic) أو معنوية (semantic) . والبرنامج المترجم (compiler) يكتشف الأخطاء التركيبية. فمثلاً يحذرنا (warns) البرنامج المترجم في أي من الحالات التالية :

- حين نخطئ في تهجي / كتابة (misspelling) الكلمات المحجوزة (reserved words).
- عدم الإعلان عن (undeclaring) أسماء تعريفية (identifiers) .
- فواصل منقوطة (semicolons) ناقصة (mising).
- أنواع معامَلات (operand types) غير متوائمة (mismatched).

ولكنه لن يكتشف جميع الأخطاء المطبعية (typing errors). فمثلاً إذا طبعنا < بدلاً من > فلن يطبع لنا رسالة تنبيه إلى خطأ (error message) ، وبدلاً من ذلك سنحصل نحن على نتائج خاطئة (erroneous results) حين نختبر (test) البرنامج.

وأما الأخطاء المعنوية (semantic) والتي يطلق عليها أيضاً الأخطاء المنطقية (logic) فهي الأخطاء (mistakes) التي تعطينا إجابة خاطئة (wrong answer). وهذا النوع من الأخطاء أصعب عموماً – من الأخطاء التركيبية – في تحديد موضعه، وعادة يبدأ في الظهور عند تنفيذ البرنامج. وتكتشف++C الأخطاء الواضحة جداً (most obvious) منها فقط، وهي تلك التي تؤدي إلى عمليات خاطئة (invalid operations) كالقسمة على صفر مثلاً. ورغم أن الأخطاء المنطقية تنتج أحياناً عن أخطاء في الطباعة (typing errors)، إلا أنها غالباً ما تنشأ عن خطأ في تصميم الخوارزمية (faulty algorithm design) . فمثلاً في برنامج Notices Program (مثال ٣-١٩) الذي يطبع رسالة خاصة بموقف الطالب من النجاح بناءً على درجاته اختبرنا صحة الدرجات من حيث أن كل درجة غير سالبة ولكننا لم نختبرها من حيث أنها لم تتجاوز الدرجة النهائية (100 مثلاً) ، فإذا أدخلنا درجة أكبر من 100 فستُقبل طالما أنها لم تتجاوز قيمة INT-MAX [وهو ثابت (constant) معلن عنه في ملف المقدمة climits (header file) ويمثل أكبر قيمة صحيحة int ممكنة في الحاسب الشخصي (personal computer) ومترجم C++ (compiler) ، وبالمثل INT-MIN يمثل أصغر قيمة صحيحة int ممكنة]، وهذا يؤدي إلى حدوث خطأ منطقي (semantic/logic error). ولذلك يفضل وجود استراتيجية لاختبار (test strategy) صحة البرنامج لتجنب أو سهولة اكتشاف مثل هذه الأخطاء المنطقية ، أو عمل ما يسمى بخطة اختبار (test plan) للبرنامج ويقصد بها طريقة أو وثيقة (document) تحدد كيفية اختبار البرنامج، وتنفيذ هذه الخطة (test plan implementation) يعني استخدام قيم/حالات اختبارية (test cases) محددة (specified) في خطة الاختبار للتحقق (verifying) من أن البرنامج يعطي في مخرجاته النتائج المتوقعة (predicted results) منه.

تمرينات رقم ٣

٣-١) اكتب تعبيراً بلغة C++ يعطي القيمة true إذا وقع letter بين 'A' و 'Z' احتوائياً (inclusive).

٣-٢) نفرض أن x, y, z, متغيرات منطقية قيمها :

x = true , y = false, z = true

أوجد قيمة كل من التعابير المنطقية التالية ، بكتابة T أن كانت النتيجة true و F أن كانت النتيجة false:

(أ) x && y || x && z

(ب) (x || !y) && (!x || z)

(ج) x || y && z

(د) ! (x || y) && z

٣-٣) نفرض أن قيم المتغيرات i, j, p, q هي :

i = 10, j = 19, p = true, q = false

أضف أقواساً – إن لزم الأمر – للتعابير التالية ، حتى تكون قيمة كل منها true:

(أ) i = = j || p

(ب) i >= j || i <= j && p

(ج) !p || p

(د) !q && q

٣-٤) نفرض أن i, j, m , n متغيرات صحيحة int قيمها:

i = 6, j = 7, m = 11, n = 11

ما هي مخرجات القطعة التالية؟

```
cout << "Chrity  ";
if  ( i < j )
        if  (m != n)
            cout << " wipes out sin  "
        else
```

```
            cout << " extinguishes sin  ";
cout << "as water  ";
if      (i  >= m)
        cout << "wipes out thirst"
else
        cout << " extinguishes fire ";
```

٣-٥) نفرض أن x, y, z متغيرات صحيحة int، حيث x تحتوي على 3 و y تحتوي على 7 و z تحتوي على 6. ما هي مخرجات كل من قطع البرامج التالية (code fragments) ؟

(أ)

```
if ( x <= 3 )
   cout << x + y << endl;
cout << x + y << endl;
```

(ب)

```
if  ( x !=  -1 )
        cout << "The value of x is " << x << endl;
else
        cout << "The value of y is " << y << endl;
```

(ج)

```
if  ( x !=  -1 )
{
        cout << x << endl;
        cout << y << endl;
        cout << z << endl;
}
else
        cout << "y" << endl;
        cout << "z" << endl;
```

٣-٦) نفرض أننا قد أُعطينا قطعة البرنامج التالية:

```
if  (height >= minHeight)
        if  (weight >= minWeight)
            cout << "Eligible to serve." << endl;
        else
             cout << "Too light  to serve." << endl;
else
        if   (weight >= minWeight)
```

```
        cout << "Too short  to serve." << endl;
    else
      cout << "Too short and too light  to serve." << endl;
```

(أ) ما هي مخرجات القطعة إذا كان height أكبر من minHeight ، وكان wieght أكبر من minWeight ؟

(ب) ماهي مخرجات القطعة إذا كان height أقل من minHeight ، وكان wieght أقل من minWeight ؟

٣-٧) وائم (match) بين كل من التعابير المنطقية المعطاة في المجموعة الأولى والتعبير المنطقي المقابل في المجموعة الثانية، بحيث يختبر التعبيران الشرط نفسه (same condition):

المجموعة الأولى:

(أ) x < y && y < z
(ب) x > y && y >= z
(ج) x != y || y = = z
(د) x = = y || y < = z
(هـ) x = = y && y = = z

المجموعة الثانية:

(1) ! (x != y) && y = = z
(2) ! (x <= y) || y < z)
(3) (y < z || y = = z) || x = = y
(4) ! (x >= y) && ! (y >= z)
(5) ! (x = = y && y ! = z)

٣-٨) ترجم التعابير التالية إلى تعابير منطقية سليمة بلغة C++ بفرض أن جميع المتغيرات المذكورة من النوع int.

(أ) x < y <= z
(ب) x, y, z جميعها أكبر من الصفر
(ج) x لا تساوي y ولا تساوي z
(د) x تساوي كلا من y, z

٣-٩) نفرض أن قيم المتغيرات النطقية x, y, z هي:

x = true, y = true, z = false

بيِّن إن كانت قيمة كل من التعابير التالية true (T) أم false (F).

(أ) ! (y || z) || x

(ب) z && x && y

(ج) ! y || (z || !x)

(د) z || (x && (y || z))

(هـ) x || x && z

٣-١٠) عند ترجمة (compilation) قطعة البرنامج (code fragment) التالية تظهر رسالة الخطأ "UNEXPECTED ELSE". ما سبب ظهورها ؟

```
if (mileage < 24.0)
{
        cout << "Gas  ";
        cout << "guzzler.";
};
else
        cout << "Fuel efficient.";
```

٣-١١) المفروض (supposed) أن تقوم قطعة البرنامج التالية بطباعة الرسالة "Type AB" عندما يكون المتغيران المنطقيان typeA , typeB كلاهما true ، بينما تطبع الرسالة "Type O" عندما يكون كلاهما false. ولكن بدلاً من ذلك فالذي يحدث هو أن القطعة تطبع الرسالة "Type O" عندما يكون أحد المتغيرين فقط (وليس كلاهما) false. أدخل القوسين { } في القطعة بحيث تؤدي وظيفتها كالمفروض

```
if  (typeA || typeB)
    if  (typeA  && typeB)
        cout <<  "Type AB";
else
        cout << "Type O";
```

٣-١٢) تشتمل بنية If المتداخلة (nested If structure) التالية على خمسة فروع (branches) ممكنة، وذلك بناء على القيم المدخلة في المتغيرات الرمزية ch1, ch2, ch3 (char variables) . ولاختبار (testing) صحة هذه البنية نحتاج لخمس مجموعات (sets) من البيانات ، حيث تختص كل مجموعة منها بأحد الفروع الخمسة . اكتب خمس مجموعات من هذه البيانات الاختبارية (five test data sets).

```
cin >> ch1 >> ch2 >> ch3;
if  (ch1 = = ch2)
    if  (ch2 = = ch3)
        cout << "All initials are the same." << endl;
     else
        cout << "First two are the same." << endl;
else if  (ch2 = = ch3)
        cout << "Last two are the same." << endl;
else if  (ch1 = = ch3)
        cout << "First and last are the same." << endl;
else
        cout << "All initials are different." << endl;
```

(أ) Test data set 1: ch1 = ـــ ch2 = ـــ ch3 = ـــ

(ب) Test data set 2: ch1 = ـــ ch2 = ـــ ch3 = ـــ

(ج) Test data set 3: ch1 = ـــ ch2 = ـــ ch3 = ـــ

(د) Test data set 4: ch1 = ـــ ch2 = ـــ ch3 = ـــ

(هـ) Test data set 5: ch1 = ـــ ch2 = ـــ ch3 = ـــ

٣-١٣) نفرض أن x, y متغيران منطقيان . هل يقوم التعبيران التاليان باختبار الشرط نفسه أم لا ؟

```
x != y
(x || y) && ! (x && y)
```

٣-١٤) يتكون شرط If التالي من ثلاثة تعابير علاقية (relational expressions):

```
if (i >= 10 && i <= 20 && i  != 16)
    j = 4;
```

فإذا فرضنا أن i تحتوي على القيمة 25 وقت تنفيذ عبارة If هذه ، فأي التعابير الثلاثة سيقوم الحاسب بتقييمها (evaluation) ؟

إرشاد : لغة C++ تستخدم وسيلة التقييم السريع/التقييم ذي الدائرة القصيرة (short-circuit evaluation).

٣-١٥) أ) اكتب عبارة تسند للمتغير المنطقي available القيمة true إذا كان numberOrdered أقل من أو يساوي الفارق: numberOnhand ناقص numberReserved

ب) اكتب عبارة تحتوي على تعبير منطقي يسند القيمة true للمتغير المنطقيisCandidate إذا كان satScore أكبر من أو يساوي 1100، و gpa ليس أقل من 2.5 ، و age أكبر من 15. وما عدا ذلك فإن isCandidate يجب أن يكون false.

ج) نفرض أن لدينا الإعلانين:

```
bool    leftPage;
int     pageNumber;
```

اكتب عبارة تسند للمتغير leftPage القيمة true إذا كان pageNumber عدداً زوجياً

إرشاد : انظر كم الباقي (remainder) عند قسمة أعداد صحيحة مختلفة على 2.

٣-١٦) اكتب عبارة If (أو سلسلة من عبارات If) تسند للمتغير biggest أكبر قيمة من القيم التي تحتويها المتغيرات i , j, k . افرض أن القيم الثلاث جميعها مختلفة (distinct).

٣-١٧) أعد كتابة المتتابعة (sequence) التالية من عبارات .. If .. Then كعبارة واحدة فقط .. If .. Then .. Else

```
if      (year % 4 =  = 0)
        cout << year << "  is a leap year ." << endl;
if      (year % 4  ! = 0)
{
        year = year + 4 - year % 4;
        cout << year << "is the next leap year." << endl;
}
```

٣-١٨) بسِّط (simplify) قطعة البرنامج التالية عن طريق حذف المقارنات غير الضرورية . افرض أن age متغير صحيح int.

```
if  (age > 64)
    cout << "Senior voter";
if  (age < 18)
     cout << "Under age";
if  (age >= 18 && age < 65)
     cout << "Regular voter";
```

٣-١٩) من المفروض أن تقوم قطعة البرنامج التالية بطباعة القيم 25, 60, 8 بهذا الترتيب ، ولكنها تطبع القيم 50, 60, 4 . فلماذا ؟

```
length = 25;
width = 60;
if  (length = 50)
    height = 4;
else
     height = 8;
cout << length << ' ' << width << ' ' << height << endl;
```

٣-٢٠) قطعة البرنامج التالية بلغة C++ مكتوبة بدون ضبط المسافات (indentation) ومع وضع عشوائي (random placement) للأقواس ، ولذلك لا يسهل تتبعها. أعد كتابة القطعة مع ضبط المسافات ومواضع الأقواس بحيث يسهل قراءتها وتتبعها.

```
// This is a nonsense program
```

```
if  (a > 0)
if  (a < 20)
              {
        cout << "A is in range." << endl;
b = 5;
     }
         else
                                  {
cout << "A is too large." << endl;
        b = 3;
}
        else
cout << "A is too small." << endl;
            cout << "All done." << endl;
```

٣-٢١) نفـــرض أن x1, x2, y1, y2, m متغيـــــرات مـن النـوع float . اكتـب قطعة برنامـــج لإيجاد ميل (slope) خط مستقيم يمر بالنقطتين (x1, y1) , (x2 , y2) باستخدام الصيغة (formula)

$$m = \frac{y1 - y2}{x1 - x2}$$

حيـث m ميـل الخـط المسـتقيم . وإذا كانـت x1 تسـاوي x2 فـإن الخـط المستقيم يكون رأسياً (vertical) والميل غير محدد (undefined) .وعلـى قطعة البرنامج أن تطبع الميل مع عنـوان مناسـب (appropriate label). وإذا كان الميل غير محدد فتطبع الرسالة : "Slope undefined".

٣-٢٢) نفـرض أن a, b, c, root1, root2, discriminant متغيرات مـن النـوع float. اكتب قطعة برنـامج لتحديـد مـا إذا كـان جـذرا الحدوديـة التربيعيـة (quadratic polynomial) حقيقـــيين (real) أم عقـــديين/مركبين (complex) . وإن كانا حقيقيين فأوجـدهما وأسـندهما للمـتغيرين root1, root2 . أمـا إن كانـا عقديين فـاطبع الرسـالة "No real roots". اسـتخدم صيغة حل المعادلة التربيعية

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

والجذران يكونان حقيقيين إذا لم يكن المميز (discriminant) **– وهو الكمية تحت علامة الجذر التربيعي – سالباً.**

٣-٢٣) البرنامج التالي يقرأ بيانات من ملف إدخال دون التحقق من أن عملية فتح الملفات قد تمت بنجاح. أدخل في البرنامج عبارات تطبع رسالة خطأ (error message) **وتنهى** (terminate) **البرنامج إذا لم نتمكن من فتح الملف.**

```
# include <iostream>
# include <fstream>              // For file I/O

using namespace std;

int main ( )
{
        int             m;
        int             n;
        ifstream        info;

        info.open ("indata.dat");
        info >> m >> n;
        cout << "The sum of  " << m << "  and  " << n
             << "  is  " << m + n << endl;
        return 0;
}
```

٣-٢٤) ما هي مخرجات كل من القطع التالية ؟

(أ)

```
int1 = 120;
cin >> int2;                          // Assume user types 30
if ( (int1 > 100) && (int2 = 50) )
     int3 = int1 + int2;
else
     int3 = int1 - int2;
cout << int1 << '  ' << int2 << '  ' << int3;
```

(ب)
```
cin >> someInt;          // Assume user types 20
if (someInt > 30)
        cout << " Prayer is light.   ";
        cout << " Charity is a proof.   ";
cout << "Patience is illumination.";
```

(ج)
```
alpha = 3                // alpha is an int variable
beta = 2                 // beta is an int variable
if (alpha < 2)
if (beta = = 3)
cout << "Every takbeera is a charity.";
else cout << "Every tahleela is a charity.";
```

٣-٢٥) ما هي قيمة x بعد تنفيذ عبارة if التالية:

```
if (x >= y) if (y > 0) x = x * y;          else if (y < 4) x = x - y;
```

إذا كانت قيمتا المتغيرين الصحيحين x ,y (int variables) هما (على الترتيب):

(أ) 9, 3 (ب) 3, 9

٣-٢٦) هل العبارة التالية صحيحة (true) أم خاطئة (false) ؟

إذا كان المتغير ch1 يحتوي على القيمة 'C' ، والمتغير ch2 يحتوي على القيمة 'K'، فإن قيمة التعبير:

```
ch1 <= ch2
```

في لغة C++ هي : (true) 1.

٣ - ٢٧) اكتب برنامجا يقوم بقراءة كمية الثمار F ، ونصاب الثمار B ، ورقم ثنائي I يدل على طريقة سقي الثمار حيث يأخذ القيمة 1 إذا كان السقي طبيعيا بدون استعمال آلة ، والقيمة صفر إذا كان السقي بآلة ، ثم يقوم البرنامج بحساب قيمة الزكاة ZF (كما هو مبين في المسألة رقم ١ - ٢).

٣ – ٢٨) اكتب برنامجا يقرأ درجة الحرارة المتوسطة في اليوم ، علما بأن درجة الحرارة عدد صحيح. كذلك يعطي البرنامج رسالة تفيد حالة الطقس في ذلك اليوم تبعا للجدول التالي :

درجة الحرارة المتوسطة	-١٠ الى ٩	١٠ الى ١٩	٢٠ الى ٢٩	٣٠الى ٣٩	٤٠ الى ٥٩
حالة الطقس	بارد جدا very cold	بارد cold	معتدل mild	حار hot	حار جدا very hot

٣ – ٢٩) تُقَدَّر زكاة عروض التجارة ZT برُبع العشر من القيمة السوقية (Market-MV Value) للعروض التجارية (والقيمة السوقية هي القيمة الفعلية في السوق للبضائع المعروضة للبيع ، ولاعبرة بثمن شرائها وتكلفتها ، ولا بالسعر المرغوب بيعها به) ، وذلك إذا بلغت هذه القيمة السوقية للمواد التجارية نصابا من الذهب (أو الفضة ، ونصاب الذهب = ٨٥ جم ذهبا ، ونصاب الفضة = ٥٩٥ جم فضة) بشرط حولان الحول (والحول يبدأ منذ الشراء بنية التجارة) [ملاحظة : الأثاث والأجهزة المستخدمة لصالح عرض التجارة وبيعها أو خزنها أو نقلها كالسيارات ونحوها لا زكاة فيها].

المطلوب :

كتابة برنامج يقرأ سعر جرام الذهب PG ، والقيمة السوقية MV للعروض التجارية ، وقيمة متغير رمزي Year (بحيث أن القيمة ''Y تعني حولان الحول ، بينما القيمة 'N' تعني عدم حولان الحول) ، ثم يحسب قيمة زكاة عروض التجارة ZT.

٣ – ٣٠) تعطَى بعض المعادن تقديرا يعتمد على نتائج اختبارات ثلاثة. وهذه الاختبارت تحدد ما إذا كان المعدن يحقق المواصفات التالية :

(١) المحتوى الكربوني (Carbon Content) أقل من ٦,٧ : (T_1).

(٢) ثابت الصلابة (Rockwell Hardness Const.) ليس أقل من ٥٠ : (T_2).

(٣) مقاومة الشد (Tensile Strength) أكبر من ٧٠,٠٠٠ وحدة / بوصة^2 : (T_3).

ويعطَى المعدن تقديرا حسب الشروط المذكورة بالأولويات التالية :

	التقدير	الشروط
	١٠	إذا نجح في الاختبارات الثلاثة.
	٩	إذا نجح في الاختبارين (١) ، (٢) فقط.
	٨	إذا نجح في الاختبار (١) فقط.
	٧	ما عدا ما سبق.

اكتب برنامجا بلغة C++ لقراءة قيم المحتوى الكربوني (CC) ، وثابت الصلابة (RC "ثابت روكول") ، ومقاومة الشد (TS) لِعَيِّنَةٍ (sample) من معدن ما ، ثم تحديد تقدير هذا المعدن ، على أن يطبع البرنامج القيم الثلاث المقروءة (البيانات) ، والتقدير المقابل المعطَى للعينة.

٣ – ٣١) اكتب برنامجا لتوزيع ميراث قدره A دينارا على ورثة رجل توفى عن أب وأم وزوجة وأبناء (ذكور) عددهم NS وبنات عددهن ND ، وافرض أن $NS > 0$. وتتحدد أنصبة الورثة بالقوانين التالية :

أ) " ولأبويه لكل واحد منهما السدس مما ترك إن كان له ولد " ، أي أن:

نصيب الأب (f) = نصيب الأم (m) : $f = m = \frac{A}{6}$

ب) " فإن كان لكم ولد فلهن الثمن مما تركتم " ، أي أن :

نصيب الزوجة : $w = \frac{A}{8}$

ج) الباقي : $r = A - (f + m + w)$ يوزع على الأبناء والبنات تبعا للقاعدة " يوصيكم الله في أولادكم للذكر مثل حظ الأنثيين " ، أي أن :

د) نصيب الإبن : $s = \frac{2r}{2NS+ND}$

هـ) نصيب البنت :

$$d = \begin{cases} s/2 \ldots (ND > 0 \text{ إذا كان}) \\ 0 \ldots (ND = 0 \text{ إذا كان}) \end{cases}$$

يقرأ البرنامج قيم ND , NS , A ويطبع قيم f,m,w,s,d .

٣ – ٣٢) اكتب برنامجا (يمكن أن يعد جزءا من برنامج عام لتوزيع الميراث) يحيث يقوم هذا البرنامج الجزئي بتحديد نصيب كل من الزوج H أو الزوجة W وكل واحد من الأبناء (نصيب الإبن S ونصيب البنت D) ، وذلك تبعا للقواعد التالية :

أولا : الأزواج :

(١) ولكم نصف ما ترك أزواجكم إن لم يكن لهن ولد ، فإن كان لهن ولد فلكم الربع مما تركن.

(٢) ولهن الربع مما تركتم إن لم يكن لكم ولد ، فإن كان لكم ولد فلهن الثمن مما تركتم.

ثانيا : الأبناء :

(أ) في حالة عدم وجود أبناء ذكور :

(١) فإن كن نساءً فوق اثنتين فلهن ثلثا ما ترك.

(وأيضا إن كانتا اثنتين فلهما ثلثا ما ترك)

(٢) وإن كانت واحدة فلها النصف

(ب) في حالة وجود أبناء ذكور :

يوصيكم الله في أولادكم للذكر مثل حظ الأنثيين.

ملاحظات :

(أ) يبدأ البرنامج بقراءة قيمة الميراث A ، وعدد الأبناء الذكور NS والإناث ND ، وقيمة رقم ثنائي I الذي يأخذ القيمة 1 إذا كان الشخص المتوفي هو الزوج ، والقيمة صفر إذا كان المتوفي هو الزوجة.

(ب) لا يحدد البرنامج نصيب بقية الأقارب غير المذكورين.

(ج) يراعى استخدام عناوين مناسبة عند طباعة النتائج.

٣ – ٣٣) – عند المقارنة بين عدة منازل لشراء منزل جديد ، يجب أن نأخذ في الاعتبار عدة عوامل وهي : التكلفة الأولية للمنزل ، وتكلفة الوقود السنوية التقديرية، ومعدل الضريبة السنوية. اكتب برنامجا لتعيين التكلفة الكلية للمنزل بعد خمس سنين وذلك لكل مجموعة من مجموعات البيانات التالية، ومن ثم يمكننا تحديد أفضل منزل للشراء.

تكلفة المنزل الأولية	تكلفة الوقود السنوية	معدل الضريبة
Initial house cost	Annual fuel cost	Tax Rate
$ 67, 000	$ 2,300	0.025
$ 62, 000	$ 2,500	0.025
$ 75, 000	$ 1,850	0.020

لحساب تكلفة المنزل الكلية أضف التكلفة الأولية الى تكلفة الوقود لخمس سنين ، ثم أضف الضريبة لخمس سنين ، علما بأن الضريبة لسنة واحدة تحسب بضرب معدل الضريبة في التكلفة الأولية.

٣ – ٣٤) اكتب برنامجا لتعيين الضريبة الإضافية المطلوبة من الموظف ، وتحسب هذه الضريبة الإضافية كما يلي :

تفرض الدولة ضريبة قدرها ٤٪ على صافي الدخل ، ويحسب صافي الدخل للموظف بأن يطرح من إجمالي دخله ٥٠٠ دولارا لكل شخص ممن يعولهم الموظف. ويقرأ البرنامج إجمالي الدخل ، وعدد الأشخاص الذين يعولهم الموظف ، وكمية الضريبة التي خصمت فعلا منه. ثم يحسب البرنامج الضريبة الفعلية المطلوبة منه ، ثم يقوم بطباعة الفارق بين هذه

الضريبة الفعلية المطلوبة والضريبة المخصومة ، ويتبع هذا الفارق بإحدى الرسالتين :

"send check" : أي أرسل شيكا ، إذا كان هذا الفارق موجبا (أي أن الضريبة التي خصمت منه أقل من المطلوبة ولذلك عليه أن يرسل شيكا بالفارق).

"Refund" : أي إعادة مال ، إذا كان هذا الفارق سالباً (أي أن الضريبة التي خصمت منه أكثر من المطلوبة ولذلك ستعيد الدولة له هذا الفارق من المال).

٣-٣٥) تقوم مؤسسة اتصالات تليفونية بالتعامل مع العملاء في تسديد الفواتير على النحو التالي :

يعرض خصم 50% للمكالمات التي تبدأ بعد 6 p.m. (الساعة 1800) وقبل 8: 00 a.m (الساعة 0800)

ليس هناك أي خصم للمكالمات التي بعد 8.:00 a.m. (الساعة 0800) وقبل 6:00 p.m (الساعة 1800)

- تفرض ضريبة على كل المكالمات مقدارها 4%
- سعر الدقيقة في المكالمة يساوي 0.40 $
- هناك خصم 15% من المبلغ إذا كانت مدة المكالمة أكثر من ٦٠ دقيقة (وذلك بعد طرح أي خصومات أخرى وقبل إضافة الضريبة)

اكتب برنامجا لقراءة وقت بداية المكالمة وطول المكالمة بالدقائق ، ومن ثم طباعة المبلغ الكلي (وهو المبلغ قبل أي خصم أو ضريبة) وطباعة المبلغ النهائي (بعد الخصم والضريبة).

٣ – ٣٦) اكتب برنامجا لحساب فواتير شركة مياه ، حيث تعتمد قيمة الفاتورة على ما إذا كانت الفاتورة خاصة بمنزل (H: House) أو شركة تجارية (C : Commercial) أو مصنع(I : Industrial) وحيث تقوم الشركة بحساب الفاتورة كما يلي :

- بالنسبة للمنزل H : قيمة الحساب 5.00$ ويضاف إليها 0.0005 $ لكل جالون.
- بالنسبة للشركة C : القيمة 1000$ لأول ٤ مليون جالون ويضاف اليها 0.00025$ لكل جالون زيادة عن ٤ مليون جالون.
- بالنسبة للمصنع I : القيمة 1000$ إذا كان عدد الجالونات لايزيد عن ٤ مليون جالون٠ وتساوي 2000$ إذا كان عدد الجالونات أكثر من ٤ مليون جالون ولا يتعدى ١٠ مليون جالون. وتساوي 3000 $ إذا كان عدد الجالونات يزيد عن ١٠ مليون جالون.

ويقرأ البرنامج رقم الحساب (وهو عدد صحيح) وشفرة المستهلك (وهي من النوع الرمزي char) ، وعدد جالونات المياه (وهو عدد حقيقي) ، ثم يطبع هذه البيانات وقيمة الفاتورة.

٣ – ٣٧)اكتب برنامجا لحساب أنواع الزكاة التالية :

أ) زكاة النقود ZM : و تقدر بربع العشر (٢,٥٪) من قيمة المبلغ المدخر tas (الذي حال عليه الحول و كان فارغا عن الدين و الحاجات الأصلية) إذا بلغ هذا المبلغ النصاب a ، أما إذا لم يبلغ النصاب فلا زكاة عليه. و نصاب النقود a يقدر بسعر ٨٥ جم من الذهب الخالص٠

ب) زكاة الغنم ZS و تحسب من الجدول التالي :

عدد الأغنام NS	أقل من ٤٠	٤٠–١٢٠	١٢١–٢٠٠	٢٠١–٣٠٠	أكثر من ٣٠٠
زكاة الغنم ZS	صفر	١	٢	٣	في كل مائة شاة

ج) زكاة الثمار والفاكهة ZF. و تقدر بالقاعدة التالية :

إذا كانت كمية الثمار F أقل من مقدار معين معطى AF (يسمى النصاب) فإن الزكاة تساوي صفرا : $ZF = 0$ ، و ما عدا ذلك فإنه

$$ZF = \begin{cases} \frac{10}{100} * F & \text{– إذا كان الري طبيعيا (أي بالمطر) :} \\ \frac{5}{100} * F & \text{– إذا كان الري صناعيا (أي بالآلة) :} \end{cases}$$

- إذا كان الري مزيجا من الطبيعي و الصناعي : $\frac{7.5}{100} * F$

و يبدأ البرنامج بقراءة قيمة سعر جرام الذهب الخالص P ، وقيمة الأموال المدخرة tas، وعدد الأغنام NS ، و كمية الثمار TF ، ونصاب الثمار AF (افرض أنه يساوي ٥٠ كيلة) ، ورمز ا code يشير إلى طريقة ري الثمار. اختبر صحة البرنامج بمجموعات مختلفة من البيانات مع طباعة النتائج.

٣ - ٣٨) اكتب برنامجا لحل المعادلتين الخطيتين الآتيتين :

$$a_1 x + b_1 y = c_1$$
$$a_2 x + b_2 y = c_2$$

بعد قراءة مجموعتي الثوابت a_1 , b_1 , c_1 و a_2 , b_2 , c_2 (راجع المسألة رقم ١-٢١).

الفصل الرابع

عبارات التكرار/العُرَي
Loops/Repetition Statements

يُطلق على الترتيب الذي تظهر به العبارات في برنامج ما "الترتيب الفيزيائي" (physical order) ، بينما يُطلق على الترتيب الذي نرغب تنفيذ العبارات به " الترتيب المنطقي" (logical order). وعبارة التكرار (repetition statement) / العروة (loop) تنفذ عبارة معينة [بسيطة (simple) أو مركبة compound] عدة مرات طالما أن شرطاً معيناً متحقق، أو أن مجموعة شروط معينة متحققة .

وفي هذا الفصل نناقش بإذن الله أنواعاً مختلفة من العرى وكيفية تكوينها باستخدام عبارة while. كما نناقش العرى المتداخلة (nested loops) وهي العُرَى التي تحتوي داخلها عرى أخرى.

عبارة while (The While Statement)

تختبر عبارة while شرطاً معيناً ، والصيغة العامة التركيبية لعبارة whileهي :

```
while (Expression)
        Statement
```

مثل

```
While (inputVal  != 25)
        cin >> inputVal;
```

وعبارة while هي بنية تحكم تكرارية (looping control structure)، والعبارة التي يعاد تنفيذها كل مرة/كل دورة من العروة يطلق عليها جسم العروة (body of the loop) . فمثلاً في المثال السابق جسم العروة هو عبارة الإدخال التي تقرأ قيمة

للمتغير inputVal . ومعنى عبارة while هذه: نَفّذ جسم العروة تكرارياً طالما أن القيمة المدخلة لا تساوي 25 . ومعنى ذلك استنفاذ (consuming) وتجاهل (ignoring) جميع القيم في سيل الإدخال حتى قراءة العدد 25 .

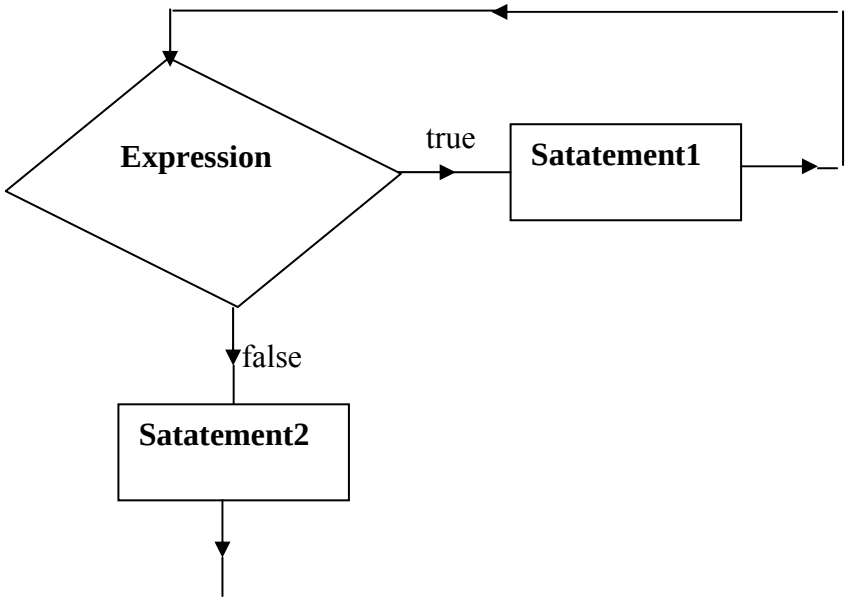
وكما هو الحال بالنسبة للشرط في عبارة if ، يمكن للشرط في عبارة while أن يكون تعبيراً من أي نوع بسيط للبيانات (simple data type) ، وهو غالباً ما يكون تعبيراً منطقياً . وإن لم يكن منطقياً فتحوَّل قيمته ضمنياً (implicitly coerced) إلى النوع المنطقي bool (تذكر أن القيمة صفر تحول إلى false وأي قيمة غير صفرية تحول إلى true . وعبارة while تعني ما يلي:

- إذا كانت قيمة التعبير true نفِّذ جسم العروة ثم عُدْ ثانية واختبر التعبير.
- وإذا كانت قيمة التعبير false تخطى / تجاوز جسم العروة.

معنى هذا أن جسم العروة سيظل ينفذ عدة مرات : مرة بعد أخرى طالما أن قيمة التعبير true عند اختباره. وعندما يصبح false فإن البرنامج يتخطى جسم العروة وينفذ العبارة التي تلي العروة مباشرة .
وبالطبع إذا كانت قيمة التعبير false من البداية فإن جسم العروة لا ينفذ والشكل التالي (شكل ٤-١) يوضح خريطة سير العمليات بالنسبة لعبارة while في صيغتها العامة:

```
while (Expression)
      Statement1
```

حيث statement1 هي جسم العروة و satatement2 هي العبارة التي تلي العروة.



شكل ٤-١

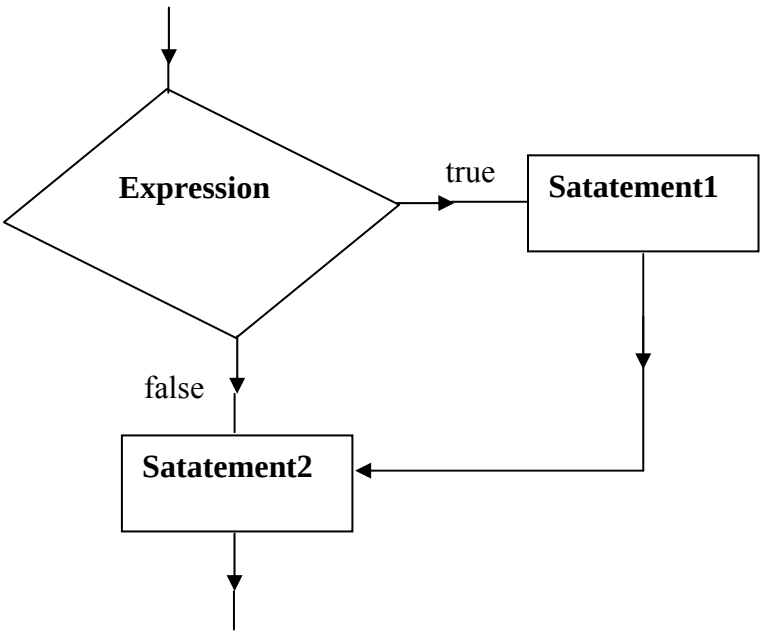
خريطة سير العمليات لعبارة while

ويمكن لجسم عروة while **أن يكون تعبيراً مركباً (قالباً)** (compound statement / block) **والذي يسمح لنا أن ننفذ أي مجموعة عبارات تكرارياً** (repeatedly) **(أي عدة مرات) وغالباً ما نستخدم عرى** while **في الصورة التالية:**

```
while (Expression)
{
    .
    .
    .
{
```

في هذه البنية (structure) **إذا كان التعبير** true **فإن متتابعة** (sequence) **العبارات في القالب** (block) **كلها تنفذ ، ثم يختبر التعبير ثانية ، فإن كان ما زال** true **فإن العبارات تنفذ ثانية ، وهكذا تستمر الدورة** (cycle) **حتى يصبح التعبير** false .

ورغم أن هناك تشابهاً بين عبارتي if , while إلا أن هناك فارقاً جوهرياً بينهما. ونذكِّر هنا بخريطة سير العمليات لعبارة If .. Then .. لتوضيح هذا الفارق.



شكل ٤-٢

خريطة سير العمليات لعبارة If .. Then..

نلاحظ في بنية if أن العبارة Statement1 إما أن تُتخطى وإما أن تُنفذ مرة واحدة بالضبط فقط. أما في بنية while فإن العبارة Statement1 إما أن تُتخطى وإما أن تُنفذ مرة واحدة وإما أن تُنفذ عدة مرات. كذلك فإن عبارة if تستخدم لاختيار مسار معين من أكثر من مسار متاح لقرارات/ لأفعال (actions) ممكنة، أما عبارة while فتستخدم لتكرير تنفيذ فعل (action) معين.

العروة باستخدام عبارة while (Loops Using the while Statement)

تنقسم العُرَى إلى نوعين رئيسيين :

(أ) العروة المحكومة بعدد (count-controlled loop):

وهي التي تكرر عدداً محدداً من المرات.

(ب) **العروة المحكومة بحدث** (event-controlled loop)**: وهي التي تُكرر حتى يتحقق حدث/شرط معين داخل العروة** (within the loop)**، وعموماً لا نعرف سلفاً عدد مرات تكرار العروة حتى يتحقق هذا الشرط.**

أولاً : العروة المحكومة بعدد (Count-Controlled Loop)

وهذه العروة تستخدم متغيرا – نسميه "متغير التحكم في العروة" (loop control variable) **– في اختبار العروة. وقبل دخول العروة نعطي هذا المتغير قيمة ابتدائية** (initial value) **ثم نختبره . وفي كل تكرير للعروة نزيد** (increment) **قيمة هذا المتغير بواحد، كما يوضح ذلك المثال التالي:**

مثال ٤-١ : اكتب قطعة برنامج لتشغيل عروة ١٠ مرات

الحل:

```
loopCount     = 1;                    // Initialization

while (loopCount <= 10)               // Test
{
        .
        .
        .                             // Repeated actions

    loopCount = loopCount + 1         // Incrementation
}
```

هنا loopCount **هو متغير التحكم في العروة ، ونعطيه ابتداءً القيمة** 1 **قبل دخول العروة. ثم تقوم عبارة** while **باختبار الشرط / التعبير** loopCount <= 10 **وتستمر في تنفيذ جسم العروة طالما أن التعبير** true**. والنقاط الموجودة داخل العبارة المركبة تمثل متتابعة من العبارات يراد تكرار تنفيذها . وآخر عبارة في جسم العروة تضيف** 1 **إلى المتغير** loopCount **. وهذا يعني أن قيمته ستزداد بواحد مع كل تكرير** (iteration) **للعروة ، ولهذا يطلق عليه عداداً** (counter) **، حيث أنه يعد التكريرات.**

وقد مرت معنا سابقاً طريقة أخرى في لغة C++ لزيادة قيمة متغير ما وذلك باستخدام مؤثر الزيادة (++) (incrementation operator). فمثلاً العبارة

loopCount + + ;

لها تأثير عبارة الإسناد التالية نفسه

loopCount = loopCount + 1;

ويلاحظ أن الشرط داخل العروة يجب أن يتغير حتى يصبح false في إحدى المرات، وإلا فلن نخرج (exit) من العروة ، وعندها يطلق على العروة : "عروة لا نهائية" (infinite loop) ، حيث أنها – نظرياً – تستمر في التنفيذ للأبد !! . وفي المثال السابق إذا حذفنا آخر عبارة – التي تزيد قيمة loopCount بواحد – فإن العروة تصبح لا نهائية حيث سيظل شرط while متحققاً دائماً لأن قيمة loopCount ستظل 1 للأبد.

ثانياً: العروة المحكومة بحدث (Event-Controlled Loop)

وهناك عدة أنواع من هذه العروة :

(أ) العروة المحكومة بقيمة حارسة (sentinel controlled)

(ب) العروة المحكومة بنهاية ملف (end-of-file controlled)

(ج) العروة المحكومة براية (flag controlled)

وفي جميع هذه الأنواع يعتمد شرط الإيقاف (termination condition) على حدوث حدث ما أثناء تنفيذ جسم العروة.

(أ) العروة المحكومة بقيمة حارسة (Sentinel-Controlled)

عادة تستخدم العروة لقراءة وتشغيل قائمة طويلة من البيانات. وفي كل مرة ينفذ فيها جسم العروة تُقرأ وتُشغَّل (processed) وحدة جديدة من البيانات. وعادة نستخدم قيمة خاصة (special value) من البيانات – تسمى القيمة الحارسة

(sentinel value) أو قيمة ذيل القائمة (trailer value) – لتعطي إشارة (signal) للبرنامج أن ظهورها يعني انتهاء البيانات المراد تشغيلها.

فتنفيذ العروة يستمر طالما أن قيمة البيانات المقروءة ليست هي القيمة الحارسة . وتتوقف العروة حين يتعرف (recognize) البرنامج على القيمة الحارسة. وبأسلوب آخر فإن قراءة القيمة الحارسة هي الحدث الذي يتحكم في تشغيل العروة . ويجب أن تكون القيمة الحارسة مختلفة تماماً عن جميع القيم التي يمكن أن تظهر في البيانات المعتادة للبرنامج: مثلاً إذا كان البرامج يقرأ تواريخ (dates) لأيام العام، فيمكننا استخدام February 31 كقيمة حارسة .

مثال ٤-٢:

اكتب قطعة برنامج لقراءة وتشغيل بيانات تمثل تواريخ ، حيث يتكون كل تاريخ (date) من عددين month, day يمثلان – على الترتيب – اليوم ، الشهر .

الحل:

```
cin >> month >> day ;          // Get a date - priming read
while ( ! (month = = 2 && day = = 31))
}
        ⋮
                                    // Process the date

        cin >> month >> day ;      // Get the next date
{
```

بدأنا بقراءة أول تاريخ ثم نختبره فإن كان هو القيمة الحارسة فنتخطى العروة إلى أول عبارة بعدها، وإن لم يكن القيمة الحارسة (أي لم يكن February 31) فنبدأ في تنفيذ جسم العروة أي نقوم بتشغيل هذا التاريخ وتنفيذ متتابعة عبارات العروة ، وبعد أن ننتهي من التشغيل نقرأ تاريخاً جديداً أي وحدة البيانات التالية ، وبذلك نكون قد وصلنا إلى نهاية العروة فنرجع لبدايتها لاختيار شرط while فإن كان true ، أي لم يكن التاريخ هو القيمة الحارسة دخلنا العروة وقمنا بتشغيل البيانات الجديدة ثم في

نهاية العروة قرأنا التاريخ التالي ، وهكذا .. إلى أن نقرأ تاريخاً هو القيمة الحارسة (February 31) فعندما نختبر شرط while سيكون false وبالتالي نتخطى العروة وينتهي تنفيذها دون تشغيل القيمة الحارسة.

نلاحظ هنا أننا احتجنا لعبارتي قراءة : الأولى خارج عروة while قبل أن نبدأها – وهذه القراءة تسمى priming read – لنستطيع اختبار شرط while في بداية العروة ، والعبارة الثانية للقراءة داخل العروة للاستمرار في قراءة التواريخ / البيانات المتتالية. ويجب أن تكون هذه العبارة في نهاية العروة وليس في بدايتها أو في ثناياها وذلك لأنها لو وضعت في بدايتها فإن تاريخها سيمحو تاريخ عبارة القراءة الأولى (priming read) قبل تشغيله !! ، ولو وضعت في ثنايا العروة (في وسطها) فإن العبارات التي قبلها فقط هي التي تمثل تشغيل التاريخ السابق أما العبارات التي بعدها فسيتم بها تشغيل التاريخ الجديد الذي قرأناه ، ولذلك يجب وضع عبارة القراءة الثانية بعد الانتهاء من جميع عبارات التشغيل ، أي وضعها في نهاية العروة.

القيمة الحارسة: في تطبيقات كثيرة تحدد لنا المسألة قيد البحث هذه القيمة الحارسة، فمثلاً إذا كانت المسألة تتعامل مع بيانات تمثل درجات طلاب score (من 0 إلى 100 مثلاً) بحيث لا يمكن أن تكون الدرجة سالبة أمكننا اعتبار القيمة الحارسة : score = -1 ، أو اعتبارها مدى (range) القيم السالبة: score < 0. وإذا كانت بيانات المسألة لا تسمح بالصفر كأن تمثل أرقام الطلاب ID اعتبرنا القيمة الحارسة ID = 0 .

وأحياناً لا يسمح بتوافق (combination) معين من القيم (أي أكثر من قيمة واحدة في الوقت نفسه) كالتوافق (February 31) كتاريخ ، كما في المثال السابق ، ولذلك اعتبرنا هذا التوافق قيمة حارسة. وعند تشغيل البيانات الرمزية – أي البيانات من النوع char – سطراً من المدخلات في الوقت نفسه (one line of input at a time) فغالباً ما نعتبر رمز السطر

الجديد ('\n') (newline character) القيمة الحارسة ، كما يوضح ذلك المثال التالي.

مثال ٤-٣: اكتب قطعة برنامج لقراءة وطباعة جميع الرموز الموجودة في سطر مدخلات.

الحل: نفرض أن inChar متغير من النوع char.

```
cin.get (inChar) ;                    // Get first character
while (inChar != '\n')
}
        cout << inChar ;              // Echo it
        cin.get (inChar);             // Get next character
{
```

نلاحظ أنه نظراً للمطلوب في هذه المسألة فقد استخدمنا الدالة get ولم نستخدم المؤثر >> لإدخال رمز ، وذلك لأن المؤثر >> كما ذكرنا سابقاً – يتخطى الرموز الفراغية البيضاء [بما في ذلك الفراغات (blanks) ورمز السطر الجديد (newline)] لإيجاد قيمة البيانات التالية في سيل المدخلات (input stream) . وفي المثال الحالي نود إدخال كل رمز حتى الفراغ وخاصة رمز السطر الجديد.

(ب) العروة المحكومة بنهاية ملف (End-of-File Controlled Loop)

هذه العروة تشبه العروة المحكومة بقيمة حارسة في أن البرنامج لايعرف سلفاً عدد وحدات البيانات التي سيقرأها، وبدلاً من الاستمرار في القراءة حتى يصل إلى القيمة الحارسة فإنه هنا يستمر في القراءة حتى يصل إلى نهاية الملف ، وفكرة هذه العروة نوضحها فيما يلي.

بعد أن يكون البرنامج قد انتهي من قراءة آخر وحدة بيانات من ملف إدخال (inputfile) فإن الحاسب يكون عند نهاية الملف EOF (end of file) ، وفي هذه اللحظة تكون حالة سيل المدخلات input stream عادية / مرضية (all

(OK / right) . ولكن إذا حاولنا الآن قراءة / إدخال أي قيم ولو واحدة من البيانات فإن السيل ينتقل إلى حالة الفشل. ويمكننا أن نستفيد من هذه الحقيقة كما يلي:

لكتابة عروة تقرأ عدداً غير معلوم من وحدات البيانات فإنه يمكننا أن نستخدم فشل سيل المدخلات كصيغة من القيم الحارسة.

وقد رأينا في الفصل السابق كيف نختبر حالة سيل مدخلات / مخرجات (I/O stream) عن طريق استخدام اسم السيل في تعبير منطقي كما لو كان هذا السيل متغيراً منطقياً ، كأن نكتب مثلاً :

```
if (inFile)
   ⋮
```

وتكون النتيجة true إذا كانت أحدث عملية I/O قد نجحت ، و false إذا كانت قد فشلت. وفي عبارة while يمكننا اختبار حالة سيل بطريقة مماثلة، كما يوضح ذلك المثال التالي.

مثال ٤-٤ : نفرض أن لدينا ملف بيانات (data file) يحتوي على قيم صحيحة (integer values) . ونفرض أن Data هو اسم سيل الملف (file stream) في البرنامج. اكتب عروة تقرأ وتطبع (صدى) (echoes) جميع قيم البيانات في الملف.

الحل:

```
inData >> intVal ;                  // Get first value
while (inData)                      // while the input succeeded
}
        cout << intVal << endl;     // Echo it
        inData >> intVal ;          // Get next value
{
```

ولتوضيح هذه العروة نفرض أن الملف يحتوي على ثلاث قيم :

10, 20, 30 . عبارة القراءة الأولية (priming read) تدخل القيمة 10.

وحيث أن شرط while سيكون true لأن عملية الإدخال نجحت ، فإن الحاسب ينفذ جسم العروة ، والذي يبدأ بطباعة القيمة 10 ، ثم يُدخل قيمة البيانات الثانية 20، وبالعودة إلى اختبار شرط العروة فإن التعبير inData سيكون أيضاً true لنجاح عملية الإدخال ، وينفذ جسم العروة مرة أخرى طابعاً 20 وقارئاً القيمة 30 من الملف ، ثم نعود إلى اختبار الشرط وفيه يكون التعبير true . ورغم أننا الآن عند نهاية الملف، إلا أن حالة السيل (strem state) مرضية – لأن عملية الإدخال السابقة قد نجحت – وبالتالي فإن جسم البرنامج ينفذ للمرة الثالثة، فيطبع القيمة 30 وينفذ عبارة الإدخال ، ولكن هذه المرة تفشل عبارة الإدخال ، حيث أننا نحاول القراءة بعد نهاية الملف. وبالتالي يدخل سيل inData حالة الفشل . وعند العودة إلى اختبار العروة فإن قيمة التعبير تكون false وبالتالي نخرج (exit) خارج العروة.

ملاحظة: عندما نكتب عروة محكومة بنهاية ملف كتلك المكتوبة في المثال السابق ويدخل السيل حالة الفشل، فليس ضرورياً أن تكون نهاية الملف – كما نتوقع – سبب هذا الفشل ، بل أنه يحدث أيضاً بسبب أي خطأ في الإدخال (any input error) . فمثلاً إذا فشل الإدخال (input fails) بسبب وجود رموز غير سليمة (invalid characters) في البيانات المدخلة (input data) فإن العروة السابقة تنتهي (loop terminates) . وهذا يؤكد لنا مرة أخرى أهمية طباعة الصدى (echo printing) ، حيث أنها تساعدنا في التحقق من أن جميع البيانات قد قرئت دون أي أخطاء حتى وصلنا إى EOF.

(ج) العروة المحكومة براية (Flag Controlled Loop)

الراية (flag) هي متغير منطقي (boolean variable) يستخدم للتحكم في سير العمليات / السيل المنطقي (logical flow) في برنامج .

فيمكننا مثلاً أن نبدأ بإعطاء القيمة true لمتغير منطقي قبل عروة while ، وبعد ذلك عندما نود إيقاف تنفيذ العروة نحوّل قيمة هذا المتغير المنطقي إلى false. أي أننا نستطيع استخدام هذا المتغير المنطقي ليسجل (record)لنا ما إذا كان الحدث (event) الذي يتحكم (controls) في العملية / التشغيل (process) قد حدث (occurred) أم لا . والمثال التالي يوضح هذه الفكرة.

مثال ٤-٥: اكتب قطعة برنامج تستمر في قراءة وجمع أعداد صحيحة int إلى أن تدخل قيمة سالبة.

الحل:

نفرض أن nonNegative هو الراية المنطقية (boolean flag).

```
sum = 0;
nonNegative = true;                    // initialize flag
while (  nonNegtive )
}
        cin >> number;
        if (number < 0)                // Test input value
           nonNegative = false;        // Set flag if event occurred
        else
           sum = sum + number;
{
```

ملاحظة ١: يمكننا أن نكتب عروة محكومة بقيمة حارسة باستخدام راية . وفي الحقيقة فإن الحل السابق يستخدم قيمة سالبة كقيمة حارسة.

ملاحظة ٢: ليس ضرورياً أن نسند للراية دائماً القيمة الابتدائية true ، بل يجوز أن نسند لها أيضاً القيمة الابتدائية false، وفي هذه الحالة نستخدم المؤثر المنطقي (!) NOT في تعبير while، ونحوّل (reset) قيمة الراية (flag) إلى true عندما يحدث الحدث (event occurs). والمثال التالي يوضح هذه الملاحظة:

مثال ٤-٦: أعد حل مثال ٤-٥ بإسناد القيمة false **ابتداء لراية** (flag) **تُدعى** negative **وهي متغير منطقي** (boolean variable).

الحل:

```
sum = 0;
negative = false;                          // initialize flag
while (  !negtive )
}
        cin >> number;
        if (number < 0)                    // Test input value
           negative = true;                // Set flag if event
occurred
        else
           sum = sum + number;
{
```

والمثال التالي يوضح كيفية إمكانية تحويل العروة المحكومة بقيمة حارسة إلى عروة محكومة براية .

مثال ٤-٧ : في مثال ١ في الفصل التمهيدي استخدم البرنامج (Payroll Program) **عبارة** while **وهي عروة محكومة بقيمة حارسة، حيث استُخدمت القيمة الحارسة** 0 **للمتغير** empNum **– الذي يمثل رقم الموظف – لإيقاف العروة. أعد كتابة العروة مستخدماً راية** flag **تدعى** moreData **عبارة عن متغير منطقي** (boolean variable).

الحل:

```
cin >> empNum ;
moreData = (empNum != 0);          // true if empNum != 0
while (moreData)
}
        :
        cin >> empNum ;            // Get the next employee number
        moreData = (empNum != 0);  // Update the flag accordingly
{
```

من تطبيقات عبارات التكرار
Loops/Repetition Statements Applications

من التطبيقات المشهورة التي تستخدم فيها عبارات التكرار :

(أ) **العد** (counting)

(ب) **الجمع** (summing)

(ج) **تتبع قيمة سابقة** (keeping track of a previous value)

وفيما يلي نستعرض هذه التطبيقات وأمثلة عليها.

(أ) العد (Counting)

يمكن للعروة أن تقوم بحساب عدد مرات تنفيذها، والمثال التالي يوضح ذلك.

مثال ٤-٨ : **اكتب قطعة برنامج تستمر في قراءة وعد رموز مدخلة** (input characters) **إلى أن يكون الرمز المدخل نقطة** (period) **فتتوقف .**

الحل: **نفرض أن** inChar **متغير من النوع** char**، وأن** count **متغير من النوع** int **. ويلاحظ أن** count **يمثل المتغير العداد** (counter variable) **في العروة، ولكن العروة ليست محكومة بعدد** (count-controlled loop) **حيث أن هذا المتغير لا يستخدم كمتغير التحكم في العروة** (loop control variable) **.**

```
count = 0;                    // Initialize counter
cin.get (inChar);             // Read the first character
while (inChar != ' . ')
{
        count ++;                     // Increment counter
        cin.get (inChar);             // Get the next charcter
}
```

نلاحظ أن العروة تستمر إلى أن تقرأ نقطة . وعند انتهاء العروة تكون قيمة count **مساوية لأقل من عدد الرموز المقروءة بواحد، وذلك لأن القيمة الحارسة** (sentinel

(value – وهي النقطة – قُرئت ولم تُعد . ويلاحظ أنه إذا كانت النقطة هي أول رمز في الرموز المدخلة فإن جسم العروة لا ينفذ ، ويحتوي count على القيمة 0 كما ينبغي . وتجدر الإشارة إلى أننا استخدمنا القراءة الأولية (priming read) لأنه العروة هنا ليست محكومة بقيمة حارسة (sentinel controlled) .

ويطلق على المتغير العداد (counter variable) في هذا المثال عداد التكريرات (iteration counter) لأن قيمته تساوي عدد مرات تكرير العروة. وبناء على هذا التعريف فإن متغير التحكم في العروة (loop control variable) بالنسبة للعروة المحكومة بعدد (count-controlled loop) هو عداد تكريرات. إلا أنه–كما رأينا في المثال السابق – فليس كل عداد تكريرات عروة هو متغير التحكم في العروة.

(ب) الجمع (Summing)

من أكثر التطبيقات التي تستخدم فيها العرى إيجاد مجموع عدة قيم من البيانات ، كما يوضح ذلك المثال التالي.

مثال ٤–٩: اكتب قطعة برنامج لقراءة قيم ١٠ أعداد وإيجاد مجموع هذه القيم.

```
sum = 0;
count = 1;                          // Initialize the sum
while (count <= 10)
{
        cin >> number;                      // Input a value
        sum = sum + number;         // Add the value to sum
        count ++;
}
```

الحل: نبدأ بوضع القيمة الابتدائية 0 للمجموع sum قبل أن نبدأ العروة، وكذلك نضع القيمة الابتدائية 1 للعداد count. وداخل العروة نقرأ قيمة للعدد number ، ونضيفها للمجموع sum ، ونزيد count بواحد ، وتتكرر هذه الخطوات ١٠ مرات ، وفي نهاية هذه الخطوات ، أي بعد انتهاء العروة تحتوي sum على مجموع قيم

الأعداد العشرة، وتحتوي count على 11 ، ويحتوي number على قيمة أخر عدد تمت قراءته.

مثال ٤-١٠: اكتب قطعة برنامج تستمر في قراءة أعداد وجمع القيم الفردية منها، وتتوقف حين تنتهي من جمع ١٠ أعداد فردية.

الحل: نفرض أن lessThanTen متغير منطقي (bool) وباقي المتغيرات كلها من النوع int.

```
count = 0;                          // Initialize event counter
sum = 0;                            // Initialize  sum
lessThanTen = true                  // Initialize loop control flag
while (lessThanTen)
{
        cin >> number;                      // Get the next value
        if (number % 2 = = 1)               // Is the value odd?
        {
          count ++;                         //yes--Increment counter
          sum = sum + number;               // Add  value to sum
          lessThanTen = (count < 10);       // Update loop control falg
        }
}
```

نختبر كل عدد يُقرأ لنرى إن كان زوجياً أم فردياً (وذلك عن طريق المؤثر %، حيث يكون العدد number فردياً إن كانت قيمة number % 2 تساوي 1 ، وما عدا ذلك فهو زوجي) . فإن كان العدد زوجياً لم نفعل شيئاً ، و إن كان فردياً أضفنا قيمته إلى sum وزدنا قيمة العداد count بواحد. وقد استخدمنا في الحل راية (flag) للتحكم في العروة ، حيث أن هذه العروة ليست محكومة بعدد (count-controlled).

ونلاحظ في هذا المثال أنه لا توجد علاقة بين قيمة المتغير العداد (counter variable) وعدد مرات تنفيذ العروة. وكان يمكننا كتابة تعبير while هكذا:

```
while (count < 10)
```

ولكن قد يظن البعض أن معنى هذا أن العروة محكومة بعدد ، وأن عدد مرات تكرارها 10 ، فبدلاً من ذلك تحكمنا في العروة بالراية lessThanTen لنؤكد على أننا نزيد قيمة count بواحد فقط عندما نقرأ عدداً فردياً. فالعداد (counter) في هذا المثال عداد لحدث (event counter) ، حيث يبدأ بالقيمة صفر وتُزاد قيمته (incremented) فقط عندما يحدث حدث معين (certain event) . أما العداد في المثال السابق (مثال ٤-٩) فكان عداد تكريرات (iteration counter) ، حيث بدأت قيمته بواحد ثم زيدت مع كل تكرير للعروة.

(ج) تتبُّع قيمة سابقة

(Keeping Track of a Previous Value)

حينما نقرأ قيمة جديدة لمتغير ما فإن قيمته السابقة تمحي ، و أحياناً نود تذكُّر القيمة السابقة والاحتفاظ بها، والمثال التالي يوضح ذلك وكيفية تنفيذه.

مثال ٤-١١: اكتب برنامجاً لحساب عدد مرات حدوث المؤثر (=!)

(not-equal operator) في ملف بيانات (data file) يحتوي على برنامج بلغة C++ .

الحل: يمكننا الوصول إلى النتيجة المطلوبة بإيجاد عدد مرات حدوث /ظهور علامة التعجب (!) التي تتبعها علامة تساوي (=) في المدخلات ، ويمكن الوصول إلى ذلك عن طريق قراءة ملف البيانات رمزاً رمزاً ، وتتبّع أحدث رمزين (keeping track of the two most recent characters) تمت قراءتهما : القيمة الحالية والقيمة السابقة . وفي كل تكرير للعروة نقرأ قيمة حالية جديدة (new current value) بينما تصبح القيمة الحالية القديمة (old current value) قيمة سابقة (previous value) . وعندما نصل إلى نهاية الملف EOF تكون العروة قد انتهت.

```
//*****************************************************************
 //NotEqualCount program
 //This program counts the occurrences of "!=" in a data file
```

```
//*****************************************************************
#include <iostream<
#include <fstream>    // For file I/O

using namespace std;

int main ( )
{
   int      count;             // Number of != operators
   char     prevChar;      // Last character read
   char     currChar;      // Character read in this loop iteration
   ifstream inFile;        // Data file

   inFile.open("myfile.dat");         // Attempt to open input file
   if ( !inFile )                    // Was it opened?
  {
      cout << "** Can't open input file **"  // No--print message
           << endl;
      return 1;                          // Terminate program
   }
      count = 0;                 // Initialize counter
      inFile.get(prevChar);       // Initialize previous value
      inFile.get(currChar);       // Initialize current value
     while (inFile)              // While previous input succeeded...
  {
      if (currChar == '=' &&  // Test for event
         prevChar == '!')
         count++;           // Increment counter
      prevChar = currChar;    // Replace previous value
                              // with current value
      inFile.get(currChar);   // Get next value
   }
   cout << count << " != operators were found." << endl;
   return 0;
  }
```

العُرَى المتداخلة (Nested Loops)

يمكن لعروة While أن تشتمل ضمن عباراتها [أي ضمن عبارات جسم العروة (loop body)] على عروة While أخرى، فتكون لدينا عروة While داخلية (inner loop) داخل عروة While خارجية (outer loop) ، فينشأ لدينا بذلك بنية تحكم مركبة (complex control structure).

نفرض أولاً أن لدينا المثال التالي لقراءة سطر واحد فقط من البيانات:

مثال ٤-١٢: اكتب قطعة برنامج لقراءة رموز (characters) من سطر مدخلات (input line) وحساب عدد الفاصلات (count number of commas) في هذا السطر.

الحل: نستمر في قراءة الرموز رمزاً رمزاً إلى أن نصادف رمز نهاية السطر `\n` ، مع مقارنة كل رمز نقرأه (inChar) بالفاصلة ' , ' فإن تطابقاً زدنا عدد الفاصلات بواحد بعد أن نكون قد بدأنا هذا العدد بالقيمة الابتدائية صفر قبل الدخول في العروة، هكذا:

```
commaCount = 0;
cin.get (inChar);
while (inChar != '\n')
{
        if (inChar = = ' , ' )
           commaCount ++;
        cin.get (inChar);
}
cout << commaCount << endl;
```

والآن نفرض أننا نود تكرار العملية السابقة (عملية عد الفاصلات في سطر) لعدة سطور وليكن مثلاً لجميع السطور في ملف ، كما في المثال التالي:

مثال ٤-١٣: اكتب قطعة برنامج لقراءة جميع السطور في ملف وحساب عدد الفاصلات في جميع هذه السطور.

الحل : عروة المثال السابق (مثال ٤-١٢) تقرأ سطراً واحداً وتوجد عدد فاصلاته ، والآن نود تكرار هذه العملية لجميع سطور الملف إلى أن نصل إلى نهاية الملف EOF ، ولذلك نستخدم عروة محكومة بنهاية ملف (هي العروة الخارجية) داخلها عروة المثال السابق (وهي العروة الداخلية)

```
cin.get (inChar);                    // Initialize outer loop
while (cin)                          // Outer loop test
{
        commaCount = 0;              // Initialize inner loop
                                     //  (Priming read is taken care of
                                     //  by outer  loop's priming read)
        while (inChar != '\n')  // Inner loop test
        {
           if (InChar = = ' , '
             commaCount ++;
           cin.get (inChar);    // Update inner termination condition
        }
        cout << commaCount << endl;
        cin.get (inChar);        // Update outer termination condition
}
```

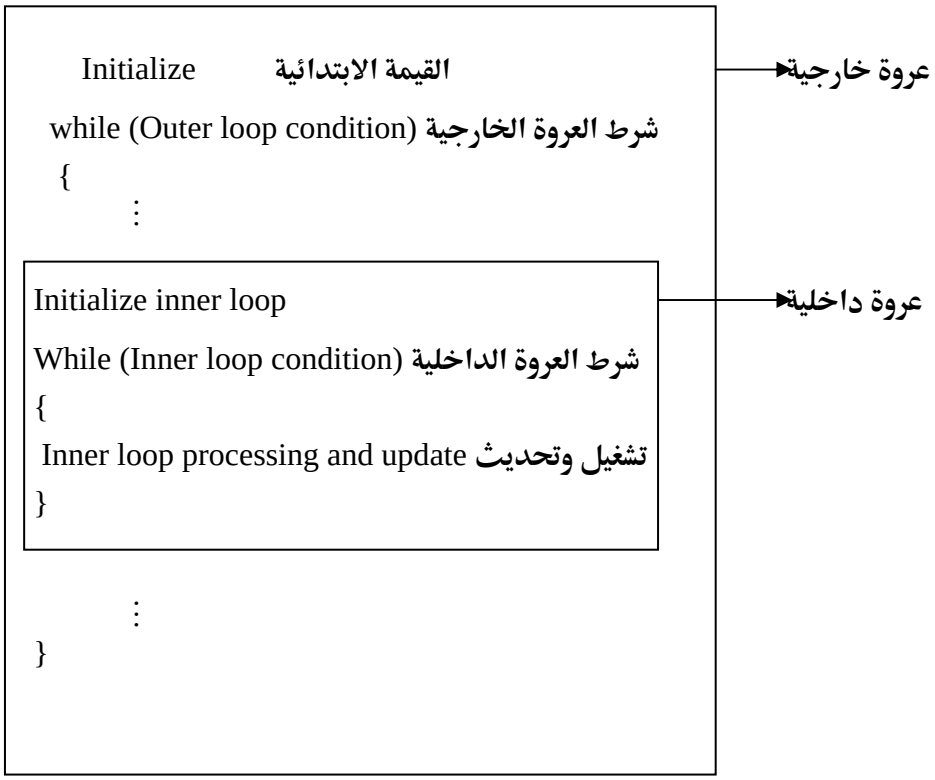
نلاحظ في هذا الحل أننا قد حذفنا القراءة الأولية (priming read)

```
        cin.get (inChar);
```

التي كانت موجودة قبل بداية العروة الداخلية ، وذلك لأن القراءة الأولية للعروة الخارجية تقوم بوظيفة القراءة الأولية ومن الخطأ أن نكتب القراءتين الأوليتين معاً (واحدة قبل العروة الخارجية وأخرى قبل العروة الداخلية) ، وذلك لأن الرمز الذي نقرأه بالقراءة الأولية الخارجية سيُمحى حينئذ قبل أن نختبره.

والشكل التالي (شكل ٤-٣) يوضح النمط /المخطط العام للعروة المتداخلة البسيطة (simple nested loop) ، حيث النقاط تمثل عبارات التشغيل وتحديث القيم (processing and update) التي قد توجد في العروة الخارجية ، كما نلاحظ أن كلا من العروتين (الداخلية والخارجية) لها قيمة ابتدائية خاصة بها وكذلك شرط اختبار (test condition) وعبارات تشغيل (processing) وتحديث (update) . ومن الممكن للعروة الخارجية ألا تقوم بأي عمليات /تشغيل باستثناء

تنفيذ العروة الداخلية تكرارياً . ومن ناحية أخرى فيمكن للعروة الداخلية أن تكون جزءاً بسيطاً من عمليات التشغيل التي تقوم بها العروة الخارجية ، بمعنى أن تكون هناك عبارات كثيرة سابقة أو عبارات كثيرة لاحقة للعروة الداخلية داخل العروة الخارجية.



شكل ٤ – ٣

عروة متداخلة

مثال ٤–١٤: اكتب قطعة برنامج تعطى نمط عروتين محكومتين بعدد متداخلتين (nested count - controlled loops) ، حيث outCount هو عداد (counter) العروة الخارجية و inCount هو عداد العروة الداخلية، و limit1 هو عدد مرات تنفيذ العروة الخارجية و limit2 هو عدد مرات تنفيذ العروة الداخلية.

الحل:

```
outCount = 1;                    // Initialize outer loop counter
```

```
while (outCount <= limit1)
{
        ⋮
        inCount = 1;                    // Initialize inner loop counter
        while (inCount <= limit2)
        {
                ⋮
                inCount ++;             // Increment inner loop counter
        }
        ⋮
        outCount ++;                    // Increment outer loop counter
}
```

نلاحظ أن كلا من العروتين عبارة عن عروة محكومة بعدد ، ولكن النمط (pattern) **السابق يمكن أن يستخدم مع أي عروتين سواء كانتا من النوع نفسه أو من نوعين مختلفين كما في المثال التالي الذي يشتمل على عروة محكومة بعدد داخل عروة محكومة بنهاية ملف.**

مثال ٤-١٥: تتبع تنفيذ قطعة البرنامج التالية واكتب مخرجاتها بالضبط بفرض أن مدخلاتها هي:

```
3
1
< EOF >
```

حيث <EOF> تعني (ضربات) المفاتيح (keystrokes) **الدالة على نهاية الملف التي يضغط عليها المستخدِم** (user)

```
cin >> starCount;
while (cin)
{
        loopCount = 1;
        while (loopCount <= starCount)
        {
                cout << '*' ;
                loopCount ++;
        }
        cout << endl;
        cin >> starCount ;
```

```
}
cout << "Goodbye" << endl;
```

الحل: تتبُّع القيم المختلفة:

المتغيرات			التعابير المنطقية
starCount	loopCount	cin	loopCount <= starCount
3	1	T	T
	2		T
	3		T
	4		F
1	1	T	T
	2		F
		F	

المخرجات (output)

```
* * *
*
Goodbye
```

وبالطبع يمكن لعروة متداخلة أن تحتوي على عروة متداخلة [وتسمى عروة متداخلة مزدوجة/ثنائية (doubly nested loop)] والتي يمكن بدورها أن تحتوي على عروة متداخلة [وتسمى عروة متداخلة ثلاثية (triply nested loop)] ، وهكذا . كما يمكن لعروة أن تحتوي على أكثر من عروة واحدة.

ومن العوامل التي تستخدم للمقارنة بين خوارزمية وأخرى درجة تعقيد الخوارزمية (complexity of algorithm) ، ويقصد بها قدر الجهد المبذول في تنفيذ الخوارزمية بالنسبة لحجم المسألة (size of the problem)، فإذا كان لدينا خوارزميتان لحل مسألة معينة أو لتأدية الوظيفة نفسها فأفضلهما أسرعهما (faster) لإنجاز المهمة، أي التي تتطلب عملاً أقل (less work) أي أن درجة التعقيد

(complexity) تعد مقياساً (measure) للجهد (effort) الذي يبذله الحاسوب لإنجاز حساباته (computations) بالنسبة لحجم الحسابات (relative to the size of the computation)

* * *

ونختم هذا الفصل بالمثال التالي على عبارات التكرار.

مثال ٤-١٦ : تقوم بعض المؤسسات – وخاصة في البلاد الغربية – بالتفرقة بين الرجل والمرأة في الجزاء على العمل، فتعطى المرأة راتباً أقل من راتب الرجل رغم قيامها بالعمل نفسه. اكتب برنامجاً لقراءة ملف incFile يحتوي على كميات رواتب (salary amounts) العاملين بإحدى المؤسسات ، وهذه الكميات /الرواتب من نوع النقطة العائمة floating point ، حيث توجد كمية (amount) واحدة فقط في كل سطر – وقبل كل كمية يوجد رمز (character) / شفرة (code) : 'F' للأنثى (female) ، و 'M' للذكر (male). وهذه الشفرة هي أول رمز على كل سطر مدخلات يعقبه فراغ (blank) يفصله عن الراتب. ثم يقوم البرنامج بطباعة بيانات المدخلات (طباعة الصدى) ، وعدد النساء ومتوسط دخلهن (average income) وعدد الرجال ومتوسط دخلهن.

الحل:

```
// ****************************************************************
 //Incomes program
 //This program reads a file of income amounts classified by
 //gender and computes the average income for each gender
// ****************************************************************
#include <iostream>
#include <iomanip>    // For setprecision()
#include <fstream>    // For file I/O
```

```
#include <string>     // For string type

using namespace std;

int main()
{
   char     sex;             // Coded 'F' = female, 'M' = male
   int      femaleCount;     // Number of female income amounts
   int      maleCount;       // Number of male income amounts
   float    amount;          // Amount of income for a person
   float    femaleSum;       // Total of female income amounts
   float    maleSum;         // Total of male income amounts
   float    femaleAverage;   // Average female income
   float    maleAverage;     // Average male income
   ifstream incFile;         // File of income amounts
   string   fileName;        // External name of file

   cout << fixed << showpoint              // Set up floating pt.
        << setprecision(2);                //   output format

 //   Separately count females and males, and sum incomes

 //   Initialize ending condition

   cout << "Name of the income data file;" :
   cin >> fileName;
   incFile.open(fileName.c_str());         // Open input file
   if ( !incFile )                         //   and verify attempt
   {
     cout << "** Can't open input file **" << endl;
     return 1;
    }
   incFile >> sex >> amount;               // Perform priming read

 //   Initialize process

   femaleCount = 0;
   femaleSum = 0.0;
```

```
   maleCount = 0;
   maleSum = 0.0;

   while (incFile)
   {
       //  Update process
      cout << "Sex: " << sex << " Amount: " << amount << endl;
      if (sex == 'F')
      {
         femaleCount++;
         femaleSum = femaleSum + amount;
      }
       else
      {
         maleCount++;
         maleSum = maleSum + amount;
      }

      //  Update ending condition

      incFile >> sex >> amount;
   }

 //    Compute average incomes

   femaleAverage = femaleSum / float(femaleCount);
   maleAverage = maleSum / float(maleCount);

 //    Output results

   cout << "For " << femaleCount << " females, the average"
        << "income is " << femaleAverage << endl;
   cout << "For " << maleCount << " males, the average "
        << "income is " << maleAverage << endl;
   return 0;
}
```

تمرينات رقم ٤

٤-١) أ) نفرض أن number **متغير من النوع** int**. ماذا تطبع قطعة البرنامج التالية؟**

```
number = 1;
while (number < 11)
{
    number ++;
  cout << number << endl;
}
```

ب) أعد ترتيب عبارات القطعة السابقة – دون تغيير طريقة كتابتها – بحيث تقوم العروة بطباعة الأعداد من 1 **إلى** 10**.**

٤-٢) عند تنفيذ قطعة البرنامج التالية ، كم يكون عدد مرات تكرار تنفيذ العروة؟

```
number = 2;
done = false;
while (  !done )
{
    number = number * 2;
     if (number > 64)
         done = true;
}
```

٤-٣) ما هي مخرجات بنية العروة المتداخلة (nested loop structure) **التالية؟**

```
i = 4;
while ( i >= 1)
{
        j = 2;
        while ( j >= 1)
        {
                cout << j << '  ';
                j -- ;
        }
        cout << i << endl;
```

```
        i -- ;
    }
```

٤-٤) من المفروض أن تقوم قطعة البرنامج التالية بطباعة الأعداد الزوجية (even numbers) **بين** 1 **و** 15 **[حيث** n **متغير صحيح** (int variable) **] ، ولكن القطعة لا تؤدي المطلوب منها.**

```
n = 2;
while (n  != 15)
{
        n = n + 2;
        cout << n <<  '   ' ;
}
```

أ- ما هي مخرجات القطعة كما هي مكتوبة؟

ب- صحح أي أخطاء بالقطعة كي تؤدي وظيفتها.

٤-٥) من المفروض أن تقوم قطعة البرنامج التالية بنسخ سطر واحد من وسيلة الإدخال القياسية (standard input device) **إلى وسيلة الإخراج القياسية** (standard output device).

```
cin.get (inChar);
while (inChar != ' \n ' )
{
        cin.get (inChar);
        cout << inChar;
}
```

أ) ما هي مخرجات القطعة إذا كان سطر المدخلات (input line) **يتكون من الرموز** ABCDE؟

ب) أعد كتابة القطعة بحيث تؤدي المطلوب منها.

٤-٦) هـل تحتـــاج قطعـــة البرنامـــــج التاليـــــة إلى أي قــراءات أوليــة (priming reads) ؟ إن كانت الإجابة : لا ، فوضح لماذا ، وإن كانت : نعم ، فأضف عبارة / عبارات الإدخال في الموضع المناسب / المواضـع المناسبة. [letter : متغير من النوع char].

```
while (cin)
{
        while (letter  != ' \n ')
        {
                cout << letter;
                cin.get (letter);
        }
        cout << endl;
        cout << "Another line read …" << endl;
        cin.get (letter);
}
```

٤-٧) ما هي مخرجات قطعة البرنامج التالية بفرض أن جميع المتغيرات صحيحة (int variables) ؟

```
i = 1;
while (i <= 5)
{
        sum = 0;
        j = 1;
        while ( j <=  i)
        {
                sum = sum + j;
                j ++ ;
        }
        cout << sum << '   ' ;
        i ++ ;
}
```

٤-٨) نفرض أن لدينا البرنامج التالي ، وأن قيم البيانات (data values) هي :

5 6 -3 7 -4 0 5 8 9

ما هي محتويات (contents) كل من sum, number بعد الخروج من العروة؟

```
# include <iostream>

using namespace std:

const int LIMIT = 8;

int main ( )
{
        int  sum;
        int  i;
        int  number;
        bool  finished;

        sum = 0;
        i = 1;
        finished = false;
        while ( i <= LIMIT && !finished)
        {
                cin >> number;
                if (number > 0)
                        sum = sum + number;
                else if (number = = 0)
                        finished = true;
                i ++;
        }
        cout << "End of test. "  << sum << '   '  << number << endl;
        return 0;
}
```

٤-٩) أوجد مخرجات كل من القطع التالية:

(أ)

```
sum = 0;
cin >> number;
while (number != -1)
{
  cin >> number;
  sum = sum + number;
```

```
    }
    cout << sum << endl;
```

افرض أن جميع المتغيرات من النوع int.

المدخلات : 25 10 6 -1

(ب)

```
finished = false;
firstInt = 3;
secondInt = 20;
while (firstInt <= secondInt && !finished)
   if (secondInt / firstInt <= 2)  // Reminder: integer division
         finished = true;
   else
         firstInt++;
 cout << firstInt << endl;
```

أفرض أن finished: متغير من النوع bool

firstInt, secondInt : متغيران من النوع int.

(ج)

```
sum = 0;
outerCount = 1;
while (outerCount <= 3)
{
        innerCount = 1;
        while (innerCount <= outerCount)
        {
           sum = sum + innerCount;
           innerCount++;
        }
        outerCount ++;
}
cout << sum << endl;
```

٤-١٠) ما هي قيمة length بعد تنفيذ قطعة البرنامج التالية؟

أفرض أن length و cout متغيران من النوع int.

```
length = 5;
count = 4;
while (count <= 6)
{
   if (length >= 100)
      length = length - 2;
   else
      length = count * length;
   count ++;
 }
```

٤-١١) ما هي نتيجة تنفيذ قطعة البرنامج التالية التي تحتوي على فاصلة منقوطة (semicolon) في نهاية السطر الذي يحتوي على شرط while؟

```
cout << 'A';
loopCount = 1;
while (loopCount <= 3);
{
   cout << 'B';
   loopCount++;
}
cout << 'C';
```

٤-١٢) اكتب قطعة برنامج تسند القيمة true للمتغير المنطقي dangerous وتوقف قراءة البيانات عندما يتجاوز الضغط pressure (وهو متغير من النوع float ويُقرأ) القيمة 510.0. استخدم dangerous كراية (flag) للتحكم في العروة.

٤-١٣) اكتب قطعة برنامج تحسب عدد مرات ظهور العدد الصحيح 28 في ملف يحتوي على 100 عدد صحيح (integers).

٤-١٤) اكتب قطعة برنامج عبارة عن عروة متداخلة مخرجاتها هي:

```
   1
1     2
   1  2  3
   1  2  3  4
```

٤-١٥) اكتب قطعة برنامج تقرأ ملف درجات طلاب أحد الصفوف (a class) [أى سعة (any size)] وتوجد الدرجة المتوسطة للصف (class average).

٤-١٦) اكتب قطعة برنامج تقرأ وتخزن (reads in) أعداداً صحيحة (integers)، وتحسب وتطبع عدد الأعداد الصحيحة الموجبة وعدد الأعداد الصحيحة السالبة ، وإذا كانت قيمة عدد ما صفراً 0 فإنه لا يعد . وتستمر هذه العملية حتى نصل إلى نهاية الملف (end-of-file).

٤-١٧) اكتب قطعة برنامج توجد مجموع الأعداد الصحيحة الزوجية من 16 إلى 26 احتوائياً (inclusive) .

٤-١٨) اكتب قطعة برنامج تطبع متتابعة جميع توافقات الساعة والدقيقة (sequence of all the hour and minute combinations) وذلك كل دقيقة خلال يوم كامل (in a whole day) ابتداءً من الساعة 1:00 A.M. وحتى الساعة 12:59 A.M.

٤-١٩) أعد حل السؤال السابق ٤-١٨) بحيث تطبع القطعة متتابعة الأوقات كل ١٠ دقائق (10 minute intervals) مرتبة في شكل جدول (table) مكون من ٦ أعمدة (columns) و٢٤ صف (rows) .

٤-٢٠) اكتب قطعة برنامج تقرأ سطراً من البيانات يحتوي على عدد غير معلوم من سلاسل الرموز (character strings) المفصولة عن بعضها البعض بفراغات (separated by spaces) ، وآخر قيمة في السطر هي سلسلة الرموز الحارسة (sentinel string) End . وتطبع القطعة : عدد سلاسل الرموز في سطر المدخلات (باستثناء السلسلة End) ، وعدد السلاسل التي تحتوى كل منها على حرف e مرة واحدة على الأقل ، والنسبة المئوية (percentage) لهذه السلاسل التي تحتوى على حرف e على الأقل مرة واحدة.

[إرشاد : لمعرفة ما إذا كان حرف e يظهر في سلسلة أم لا استخدم الدالة find من دوال الطبقة (class) string .]

٤-٢١) أعد حل السؤال السابق ٤-٢٠) بحيث تقرأ القطعة جميع سطور ملف بيانات inFile (data file) وتطبع النتائج الثلاث (المذكورة في السؤال السابق) لكل سطر من سطور الإدخال.

٤-٢٢) عدِّل قطعة البرنامج في السؤال السابق (٤-٢١) بحيث تحتوي أيضاً على عداد (count) يحتفظ بالعدد الكلي لسلاسل الرموز في الملف، وعداد للعدد الكلي لسلاسل الرموز التي تحتوى على رمز e واحد على الأقل [أيضاً دون عد سلسلة الرموز الحارسة في كل سطر] . وعندما نصل إلى نهاية الملف تطبع القطعة النتائج الثلاث للملف كله.

٤-٢٣) عدِّل برنامج الدَّخل (Incomes program) (مثال ٤-١٦) بحيث يقوم أيضاً بما يلي :

أ) يطبع رسالة خطأ (error message) عندما يقرأ قيمة (amount) سالبة للدخل (negative income) ، ثم يستمر في تشغيل (processing) أي بيانات متبقية ، على ألا تدخل البيانات الخاطئة (erroneous data) في أي حسابات.

ب) يطبع رسالة خطأ مناسبة إذا لم يشتمل ملف الإدخال على أى بيانات لذكور (males) أو أي بيانات لإناث (females) أو كان خالياً (empty) تماماً، وذلك بدل أن ينهار البرنامج (crashes) .

ج) يطبع رسالة خطأ إذا كان حرف الشفرة مختلفا عن ' M ' (للذكر) أو ' F ' للأنثى رافضاً هذه المجموعة من البيانات (data set) قبل أن يستمر لتشغيل بقية البيانات . كذلك يطبع البرنامج رسالة تبين عدد مجموعات البيانات الخاطئة (number of erroneous data sets) التي وجدها في الملف.

٤-٢٤) عدّل برنامج الدخل (Income program) (مثال (٤-١٦) بحيث يعطى أعلى دخل وأقل دخل لكل من الذكور والإناث.

٤-٢٥) أ) اكتب برنامجا يقرأ قيم ١٠ أعداد صحيحة ويحدد لكل عدد إن كان فرديا أو زوجيا ، وذلك باستخدام بنية if .. then .. else والمؤثر ٪ ، ويطبع مخرجاته في الصورة التالية:

المدخلات	المخرجات
Input	Output
69	69 IS ODD.
54	54 IS EVEN.
35	35 IS ODD.
⋮	⋮
⋮	⋮
⋮	⋮
62	62 IS EVEN.

٤-٢٦) اكتب برنامجا يقرأ ٢٠ زوجا من الأعداد الصحيحة a , b ، ويحسب لكل زوج قيمة a مرفوعة لأس b (أي يحسب a^b)، ويطبع النتيجة في صورة مماثلة لما يلي :

المدخلات	المخرجات
Input	Output
1 5	1 TO THE POWER 5 = 1
64 3	64 TO THE POWER 3 = 262144
12 4	12 TO THE POWER 4 = 20736
	⋮
	⋮

٤-٢٧) اكتب برنامجا يقرأ ٢٠ عددا صحيحا موجبا num بحيث أن أي عدد منها هو إحدى قوى ٢ (أي ١ ، ٢ ، ٤ ، ٨ ، ١٦ ، ٣٢ ، ...) ، ثم يعين لكل عدد num من هذه الأعداد قيمة العدد الصحيح n (القوة / الأس) الذي لو رفعت إليه ٢ لأعطت العدد المدخل num (أي أن 2^n = num) ، ويطبع النتائج في صورة مماثلة لما يلي :

[ملاحظة : يمكن الحصول على قيمة n بالطريقة التالية :
إذا قسمنا العدد num على ٢ ، ثم قسمنا الناتج على ٢ ، وكررنا هذه القسمة عددا من المرات إلى أن نحصل على ناتج يساوي ١ ، فإن هذا العدد الكلي من المرات يساوي n].

المدخلات	المخرجات
Input	Output
1	2 TO THE POWER 0 = 1
2	2 TO THE POWER 1 = 2
4	2 TO THE POWER 2 = 4
1024	2 TO THE POWER 10 = 1024
	⋮
	⋮

٤-٢٨) سقط جسم من السكون من ارتفاع ٦٠٠٠٠ متر. اكتب برنامجا يعطي سرعة الجسم كل ١٠ ثواني وإلى أن يرتطم بالأرض ، علما بأن هذه السرعة تعطى بالعلاقة $v = g\,t$

حيث v : السرعة بالمتر / ثانية

g : عجلة الجاذبية وتساوي ٩,٨١ متر / ثانية ^{٢}

t : الزمن بالثانية

إرشاد :

يمكن استخدام العلاقة $t = \sqrt{\frac{2s}{g}}$ حيث s هي المسافة التي تحركها الجسم ابتداء من لحظة سقوطه إلى الزمن t ، وذلك لحساب الزمن limit الذي يستغرقه الجسم للوصول إلى الأرض ، ثم استخدام هذه القيمة للخروج من العروة التي تحسب السرعات.

٤-٢٩) المطلوب حساب قيمة Z والتي تعطى بالعلاقة

$$Z = \frac{e^{ax} - e^{-ax}}{2} \operatorname{Sin}(x + b) + a \log_e \frac{b+x}{2}$$

وذلك لجميع القيم التالية للمتغيرات x , a , b :

x=1.0(0.1)2.0 , a = 0.10(0.05) 0.80 , b = 1.0 (1.0) 10.0

حيث x = 1.0 (0.1) 2.0 تعني أن

x = 1.0 , 1.1 , 1.2 , 1.3 , … , 2.0

وهكذا بالنسبة لباقي المتغيرات.

اكتب برنامجا يستخدم ثلاث عرى من عرى While لإعطاء قيم المتغيرات الثلاثة ، ويطبع قيم Z في جدول يعطي قيمة Z مقابل كل مجموعة من قيم المتغيرات x , a , b .

(ملاحظة : يوجد ١١ × ١٥ × ١٠ = ١٦٥٠ مجموعة من هذه القيم للمتغيرات).

٤-٣٠) يعتمد حل المعادلة الخطية ax + b = 0 على قيمة كل من a, b :

(أ) فإذا كانت a ≠ 0 فإن الحل هو القيمة x = -b/a.

(ب) وإذا كان a = 0 , b ≠ 0 فإن المعادلة لا تحقق أي قيمة حقيقية للمتغير x.

(ج) وإذا كانت a = 0 , b = 0 فإن أي قيمة حقيقية للمتغير x هي حل للمعادلة.

اكتب برنامجا لقراءة عشرين مجموعة للثابتين a , b (كل مجموعة على سطر مستقل ، وتشتمل على قيمة للثابت a ، وقيمة للثابت b ، أي أن كل مجموعة تخص معادلة واحدة) ثم لإيجاد حلول هذه المعادلات.

٤-٣١) اكتب برنامجا لقراءة قيمة عدد صحيح موجب n ثم حساب مجموع المتسلسلة :

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots\pm\frac{1}{n}$$

٤-٣٢) اكتب برنامجا لحساب y من العلاقة التالية

$$y = 16.7\,x + 9.2\,x^2 - 1.02\,x^3$$

وذلك لقيم x من 1.0 إلى 9.9 وبزيادة متتالية في قيم x تساوي 0.1. اطبع قيمة كل من x , y.

٤-٣٣) اكتب برنامجا لحساب الدالة

$$f(\theta) = e^{\cos\theta} - 4\theta^2 + 2\sqrt{|\theta|}$$

وذلك لقيم θ من $-\pi$ إلى π وبزيادة متتالية في قيم θ تساوي $\frac{\pi}{16}$.

اطبع قيمة كل من θ ، f(θ) . (π=3.14159)

٤-٣٤) اكتب برنامجا لحساب y كدالة في x تبعا للعلاقة

$$y = \sqrt{1+x} + \frac{\cos 2x}{1+\sqrt{x}}$$

وذلك لعدد من قيم x المتساوية المسافات فيما بينها مبتدئا بالقيمة الابتدائية XIN وباتباع الخطوات التالية :

(أ)اقرأ قيم ثلاثة أعداد XFI, DELTA, XIN.

(افرض أن : XIN < XFI , DELTA > 0 , XIN > 0)

(ب)احسب قيمة y المقابلة لقيمة x = XIN واطبع قيمتي x , y .

(ج)زد قيمة x بالقيمة DELTA واحسب قيمة y المقابلة لقيمة x الجديدة هذه ، واطبع قيمتي x , y.

٤-٣٥) اكتب برنامجا يترجم خريطة سير العمليات في المثال ١-٦ إلى لغة C++ ليقوم بقراءة مجموعة من درجات الحرارة φ بالتقدير الفهرنهيتي ويحولها إلى الدرجات المقابلة θ بالتقدير المئوي تبعا للعلاقة $\theta = \frac{5}{9}(\phi - 32)$.

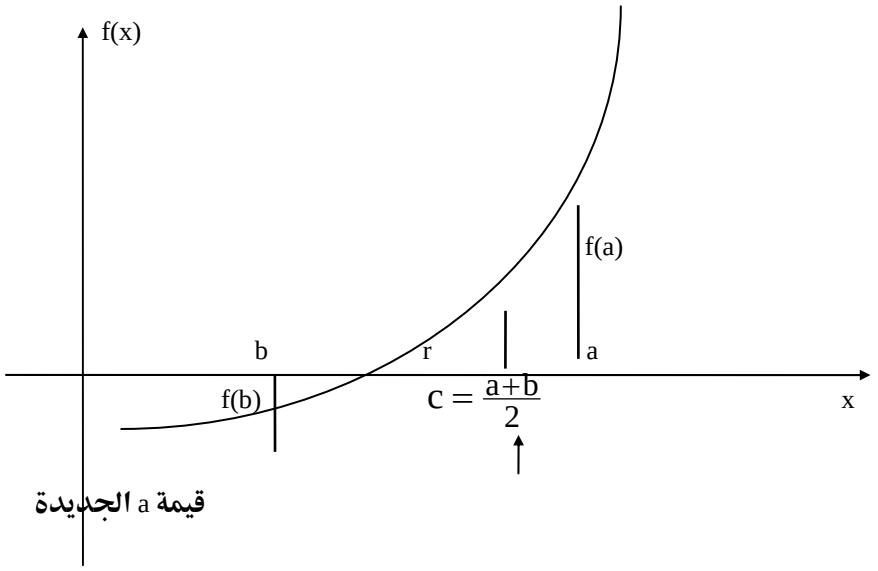
٤-٣٦) اكتب برنامجا مقابلا لخريطة سير العمليات في المسألة ١-٥ لحساب التعداد السنوي للسكان في كل من البلدين A , B وإلى أن يزيد تعداد سكان B عن تعداد سكان A.

٤-٣٧) اكتب برنامجا لحساب مجموعة المتسلسلة التالية (انظر المسألة ١-٧-ii)

$$s = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \cdots + \frac{99}{100}$$

٤-٣٨) تعتمد طريقة التنصيف لإيجاد جذر r للمعادلة $f(x) = 0$

(انظر خوارزمية الطريقة في المسألة رقم ١-١٠) على الفكرة التالية :



ابحث عن قيمتين من قيم x (مثلا $x = b , x = a$**) بحيث أن** $f(b) , f(a)$ **لهما إشارتان مختلفتان (انظر الرسم) ، وبالتالي فلابد أن يوجد جذر r (أي أن** $f(r) = 0$**) بين** b , a **. ولإيجاد قيمة r :**

١-احسب قيمة c بحيث أن $c = \frac{a+b}{2}$

٢-احسب قيمة $f(c)$

٣-إذا كان $|f(c)| < \varepsilon$ (حيث ε عدد صغير جدا مثل 10^{-6} ويعتمد على الدقة المطلوبة في حل المسألة) فاطبع قيمة c (على أساس أنها الجذر المطلوب r) وتوقف.

٤-إذا كانت إشارة $f(c)$ هي إشارة $f(a)$ نفسها (كما هو بالرسم وهذا يعني أن c , a على الجانب نفسه من الجذر الحقيقي) فاعط قيمة c للمتغير a ، أي أن قيمة a الجديدة هي $a = c$ ، وإلا فاعط قيمة c للمتغير b أي أن قيمة b الجديدة تصبح $b = c$.. وفي أي من الحالتين ارجع بعد ذلك إلى الخطوة رقم ١ لحساب قيمة جديدة للمتغير c.

المطلوب :

كتابة برنامج لإيجاد جذر للمعادلة

$$f(x) = x - \sin x - 0.5 = 0$$

باتباع طريقة التنصيف المشروحة سابقا

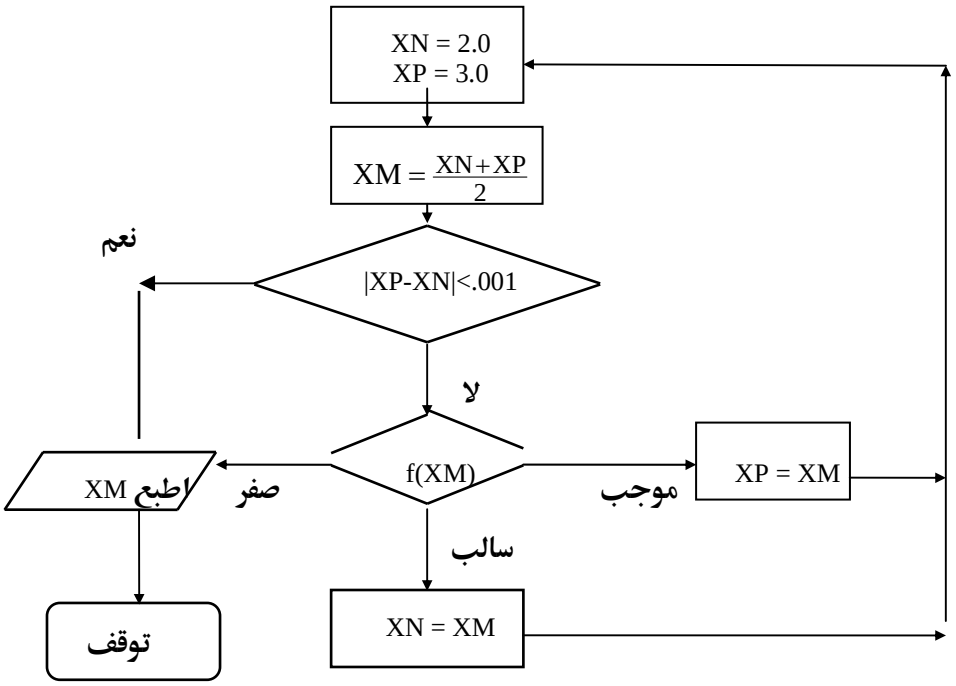
(إرشاد : $f(0) < 0$, $f(\pi) > 0$, $\pi = 3.14159$)

٤-٣٩) اكتب برنامجا لإيجاد جذر للمعادلة

$$f(x) = x^3 - 4x^2 + 6x - 7 = 0$$

بين القيمتين $x = 2$, $x = 3$

وذلك باستخدام طريقة تنصيف المسافات والموضحة خطواتها التفصيلية في الرسم التالي :



٤٠-٤) يمكن إيجاد الجذر التكعيبي لعدد موجب y ، أي إيجاد قيمة X حيث

$$x = \sqrt[3]{y}$$

باتباع الطريقة التالية التي تعتمد على هذه الفكرة :

إذا كانت X_i قيمة تقريبية للجذر X فإن X_{i+1} التي نحصل عليها من تطبيق الصيغة التكرارية :

$$x_{i+1} = \frac{1}{3}\left(2X_i + \frac{y}{X_i^2}\right)$$

تكون قيمة تقريبية أفضل من X_i .. أما الطريقة فيمكن صياغتها كما يلي :

-ابدأ بأي قيمة ابتدائية اجتهادية للجذر التكعيبي ولتكن مثلا $X_0 = \frac{y}{3}$.

-طبق الصيغة التكرارية السابقة عدة مرات (بوضع $i = 0,1,2,\ldots$) إلى أن يصل الفرق بين تقريبين متتاليين إلى أقل من عدد صغير ε يعتمد على الدقة المطلوبة في الجذر

$$|X_{i+1} - X_i| < \varepsilon$$

المطلوب :

(أ)رسم خريطة سير عمليات لتوضيح الطريقة السابقة.

(ب)كتابة برنامج لقراءة قيم عشرة أعداد حقيقية (كل عدد على سطر مستقل) ، وإيجاد الجذور التكعيبية لهذه الأعداد باستخدام الطريقة السابقة.

٤-٤١) يتجه المسلمون من جميع أنحاء العالم لأداء فريضة الحج حيث يقفون جميعا على عرفات ويؤدون مناسكهم بلا أدنى تفرقة بينهم بسبب اختلاف قومياتهم أو جنسياتهم أو ألوانهم أو ألسنتهم أو مراكزهم وإنما الجميع إخوة في الإسلام.

نفرض أن كل حاج قد أعدت له بطاقة عليها قيمة عدد صحيح K يشير إلى كيفية وصوله ، بحيث أن الشفرة المستخدمة هي أنK = ١ أو ٢ أو ٣ للدلالة على أنه وصل برا أو بحرا أو جوا على الترتيب. فإذا أُعطيت مجموعة بطاقات كل الحجاج فاكتب برنامجا يقرأ هذه البيانات ويحسب :

(أ)أعداد الحجاج NA , NS , NL الذين وصلوا برا وبحرا وجوا على الترتيب.

(ب)العدد الكلي للحجاج N.

(ج)النسب المئوية PA , PS , PL للحجاج الذين وصلوا برا وبحرا وجوا على الترتيب.

٤-٤٢) يجب ألا تقوم جامعة في بلد مسلم بتقليد الجامعات الغربية في تطبيق نظام الاختلاط بين الطلبة والطالبات ، وخاصة في وجود التبرج وعدم الالتزام باللباس الإسلامي.

نفرض أن أحد المقررات والذي يدرسه طلاب من قسمي الفيزياء والرياضيات قد قسمت شعبه إلى أربع شعب : اثنتان منها للطلبة وهما الشعبة رقم ١ والشعبة رقم ٢ ، واثنتان للطالبات وهما الشعبة رقم ٥١ والشعبة رقم

٥٢. ونفرض أن كل طالب يدرس هذا المقرر قد أعدت له بطاقة عليها ثلاثة أرقام :

رقم الطالب I ، ورقم الشعبة J ، ورقم القسم K ، حيث K تساوي ١ لقسم الفيزياء ، وتساوي ٢ لقسم الرياضيات. ثم وضعت بطاقة إضافية عليها رقم القسم يساوي ٣ لتشير إلى انتهاء بطاقات البيانات.

اكتب برنامجا لقراءة هذه البيانات وحساب :

(أ)العدد الكلي للطلبة NB ، والعدد الكلي للطالبات NG.

(ب)عدد طالبات قسم الرياضيات في الشعبة رقم ٥١ (ونرمز لهذا العدد بالرمز M) ونسبتهن المئوية PM بالنسبة للعدد الإجمالي للطالبات.

(ج)عدد طلبة قسم الفيزياء في الشعبة رقم ٢ (ونرمز لهذا العدد بالرمز L) ونسبتهم المئوية PL بالنسبة للعدد الإجمالي للطلبة.

٤٣-٤) يتزايد المجتمع الإحصائي (population) لمزرعة البكتريا مع الوقت زيادة طردية مباشرة مع حجم هذا المجتمع ، وبالتالي فكلما كان المجتمع أكثر عددا ، كان تزايد أعداد البكتريا أسرع .. يمكن التعبير رياضيا عن حجم المجتمع في أي وقت بالعلاقة :

$$p = P_0\left[1+\alpha t+\frac{(\alpha t)^2}{2!}+\frac{(\alpha t)^3}{3!}+L+\frac{(\alpha t)^n}{n!}\right]$$

حيث :

a : مقدار ثابت ويساوي ٠,٠٢٨٩

t : الوقت مقاسا بالساعات من لحظة معينة

P_0 : الحجم الابتدائي لمجتمع البكتريا عند هذه اللحظة المعينة

p : حجم مجتمع البكتريا عند الزمن t

اكتب برنامجا لحساب عامل التكاثر $\frac{p}{p0}$ عند زمن معين t ، وذلك بأخذ أول عشر حدود من المتسلسلة أي بوضع n = 9 .

٤٤-٤) اكتب برنامجا لقراءة قيمة X ثم حساب قيمة الدالة

$$G(X) = 1 - X + \frac{X^2}{2!} - \frac{X^3}{3!} + \frac{X^4}{4!} - \cdots + (-1)^n \frac{X^n}{n!} + \cdots$$

بحيث نأخذ من هذه المتسلسلة حدودا متتالية طالما أن القيمة المطلقة للحد أكبر من أو تساوي واحد من المائة ألف.

حاول أن تكون كفاءة البرنامج عالية ، بحيث يستغرق تنفيذه أقل وقت ممكن وبحيث يشغل أقل حيز ممكن من ذاكرة الحاسب.

٤٥-٤) تحتوي كل بطاقة من مجموعة بطاقات على ثلاثة أعداد موجبة A,B,C.
أضيف في نهاية المجموعة بطاقة عليها ثلاثة أعداد سالبة. اكتب برنامجا يحدد ما إذا أمكن للقيم A,B,C **على كل بطاقة أن تكوِّن أطوال أضلاع مثلث أم لا. إذا كانت الإجابة نعم فاحسب طول محيط المثلث** (P = A + B + C) **وإذا كانت الإجابة لا فاطبع الرسالة** 'NOT A TRIANGLE'
وفي كلا الحالتين اطبع قيم A,B,C.

إرشاد : يمكن لأطوال الأضلاع A,B,C **أن تكوِّن مثلثا إذا كان طول كل ضلع أصغر من مجموع طولي الضلعين الآخرين ، أي إذا كان :**

$$A < B + C, \qquad B < A + C \quad \& \quad C < A + B$$

٤٦-٤) اكتب برنامجا مع رسم خريطة سير عملياته لحساب :

(أ) قوى العدد ٢ :

$$y_i = 2^i, \quad i = 1, 2, 3, \ldots$$

(ب) مقلوبات هذه القوى ، أي الكسور :

$$z_i = \frac{1}{2^i} \qquad ; i = 1,2,3,\ldots..$$

(ج) المجموعات المتراكمة الجزئية لهذه الكسور :

$$s_i = \sum_{k=1}^{i} \frac{1}{2^k}$$

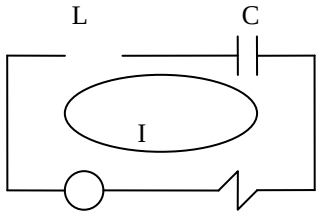
بحيث يطبع البرنامج النتائج في أربعة أعمدة كما يلي :

Powers	Exponents	Fractions	Sums
i	y_i	z_i	s_I
1	2	0.5	0.5
2	4	0.25	0.75
3	8	0.125	0.875
4	16	0.0625	0.9375
⋮	⋮	⋮	⋮

مع كتابة عناوين مناسبة ، وبحيث ينتهي تنفيذ البرنامج عندما تصبح قيمة الكسر z_i أقل من أو تساوي واحد من المليون $\left(\frac{1}{2^i} \le 0.000\,001\right)$

٤-٤٧) تحسب قيمة التيار الكهربائي I المار في الدائرة المبينة بالشكل بالعلاقة

$$I = \frac{E}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}} \quad ; \quad \omega = 2\pi f \qquad \pi = 3.14159265$$



اكتب برنامجا

أ) يقرأ قيم

١) E , L , C , R

٢) $f_{max} , f_{min} , \Delta$

ب) يحسب قيم التيار الكهربائي I لقيم التردد المختلفة f المتساوية الأبعاد فيما بينها حيث تزيد كل قيمة منها عن القيمة السابقة لها بمقدار Δ ، والقيمة الابتدائية هي f_{min} والقيمة النهائية لا تتجاوز f_{max} .

ج) يطبع جدولا لقيم I المقابلة لقيم f المختلفة.

٤-٤٨) اكتب برنامجا يقرأ قيم عناصر مجموعة أعداد صحيحة عددها n ، ويوجد عدد العناصر الموجبة وحاصل ضربها.

٤-٤٩) اكتب برنامجا لإيجاد كل من عدد العناصر الموجبة ، وعدد العناصر السالبة ، وعدد العناصر الصفرية في مجموعة مدخلات تتكون من n عدد صحيح.

٤-٥٠) يتكون أحد الفصول الدراسية من ٣٠ طالبا ، لكل طالب ٤ درجات في ٤ اختبارات. اكتب برنامجا لحساب الدرجة المتوسطة لكل طالب وكذلك لإيجاد أعلى درجة متوسطة.

٤-٥١) نفرض أن n عدد صحيح موجب. اكتب برنامجا يقرأ قيمة n وقيم عناصر مجموعة مكونة من n عدد صحيح ، ويوجد مجموع الأعداد الفردية (في هذه المجموعة) التي تقبل القسمة على ٣ بدون باق.

مثلا : إذا كانت $n = 6$ وكانت عناصر المجموعة هي :

$7 , 6 , 9 , 10 , 15 , 11$

فإن البرنامج يعطي المجموع : $sum = 9 + 15 = 24$

٤-٥٢) نفرض أن كلا من i , j , k عدد صحيح محصور بين ١ و ٩ $[1 \le i , j , k \le 9]$. اكتب برنامجا يعطي جميع الثلاثيات (triplets) (i, j, k) التي تحقق الشرط $j = \frac{i+k}{2}$.

[من أمثلة هذه الثلاثيات : $(1,3,5) , (4,4,4) , (2,5,8)$]

٤-٥٣) اكتب برنامجا لإيجاد درجة الحرارة المتوسطة في اليوم ، علما بأن درجة الحرارة تقاس كل ساعة ، أي أن البرنامج يقرأ ٢٤ درجة حرارة. كذلك يعطي البرنامج رسالة تفيد حالة الطقس في ذلك اليوم تبعا للجدول التالي:

درجة الحرارة المتوسطة	-١٠ إلى ٩	١٠ إلى ١٩	٢٠ إلى ٢٩	٣٠ إلى ٣٩	٤٠ إلى ٥٩
حالة الطقس	بارد جدا very cold	بارد cold	معتدل mild	حار hot	حار جدا very hot

ملاحظة : درجات الحرارة أعداد صحيحة ، ودرجة الحرارة المتوسطة تحسب لأقرب عدد صحيح.

٤-٥٤) اكتب برنامجا لإيجاد أعلى درجة حرارة وأقل درجة حرارة في اليوم علما بأن درجة الحرارة تقاس كل ساعة ، أي أن البرنامج يقرأ ٢٤ درجة حرارة.

٤-٥٥) العلاقة بين الضغط والحجم ودرجة الحرارة لغاز مثالي يمكن تمثيلها بالعلاقة : $PV = KT$ ، وبصورة أدق يمكن تمثيل هذه العلاقة بالمعادلة :

$$PV = 0.0299 (T + 460)$$

حيث :

P : هو الضغط (ووحدته : لكل بوصة مربعة psi)

V : هو الحجم (ووحدته : قدم^3 cubic feet)

t : هي درجة الحرارة (بالتقدير الفهرنهيتي ^oF)

والمطلوب حساب V لتوافقات مختلفة من P , T كما يلي :

اكتب برنامجا لطباعة جدول لقيم V للضغوط من ١٥٠ إلى ٢٠٠ وبزيادة ثابتة تساوي ٥ ، ولدرجات الحرارة من ١٠٠ إلى ٥٠٠ وبزيادة ثابتة تساوي ٥٠ ، مع كتابة عناوين مناسبة للجدول.

٤-٥٦) تنخفض قيمة السيارة سنويا بنسبة ٢٠٪ نتيجة الاستهلاك ، أي أن قيمة السيارة في نهاية أي عام تساوي ٨٠٪ من قيمتها في بداية العام.

اكتب برنامجا يقرأ القيمة الابتدائية للسيارة ويحسب ويطبع قيمة السيارة :

أ) في بداية كل عام من الأعوام العشرة التالية.

ب) في بداية كل عام من الأعوام التالية إلى أن تصل قيمة السيارة إلى أقل من ٧٠٠ دينار.

٤-٥٧) نفرض أن :

N : عدد صحيح موجب.

SUM : مجموع قواسم (divisors) [أي عوامل (factors)] العدد N ما عدا العدد نفسه.

يقال إن العدد N :

(أ) **عدد تام** (perfect) : **إذا كان العدد مساويا مجموع قواسمه** SUM.

(ii) **عدد ناقص** (deficient) : **إذا كان العدد أقل من مجموع قواسمه** SUM.

(iii) **عدد زائد** (abundant) : **إذا كان العدد أكبر من مجموع قواسمه** SUM.

[**مثلا :** ٦ ، ٢٨ **عددان تامان لأن :** ٦ = ١ + ٢ + ٣ ، ٢٨ = ١ + ٢ + ٤ + ٧ + ١٤

١٢ ، ٢٠ **عددان ناقصان : لأن :** ١٢ > ١ + ٢ + ٣ + ٤ + ٦ ،

٢٠ > ١ + ٢ + ٤ + ٥ + ١٠

٨ ، ١٠ **عددان زائدان لأن :** ٨ < ١ + ٢ + ٤ ، ١٠ <١ + ٢ + ٥]

اكتب برنامجا يقرأ قيمة عدد صحيح موجب N ، **ويوجد بكفاءة كل قواسم العدد** N **– باستثناء العدد نفسه – ومجموع هذه القواسم ، ويحدد ما إذا كان العدد تاما أو ناقصا أو زائدا.**

٤–٥٨) **اكتب برنامجا لقراءة جملة** (sentence) **وحساب وطباعة عدد كلماتها. افرض أن الجملة تتكون من رموز** (characters) **بحيث أنه يوجد فراغ واحد** (one space) **بين أي كلمتين متتاليتين ، وأن الجملة تنتهي بنقطة** (full-stop). **مثلا إذا قرأ البرنامج الجملة :**

Be in the world as though you were a stranger or a wayfarer.

فإنه يطبع رسالة مثل :

number of words in sentence = 13

٤–٥٩) **يعتمد معدل تحلل أي عنصر من النظائر المشعة** (radioactive isotopes) **على عمره النصفي** H (half - life) **وهو الفترة الزمنية اللازمة للنظير ليتحلل إلى نصف كتلته الأصلية. مثلا** H **بالنسبة للنظير "سترونتيم ٩٠"** (Strontium 90)(Sr^{90}) **تساوي ٢٨ سنة.**

اكتب برنامجا لحساب وطباعة جدول يعطي كمية هذا النظير المتبقية بعد كل سنة لمدة خمسين سنة بفرض أن الكمية الابتدائية تساوي ٥٠ جراما ، ويمكن حساب كمية النظير المتبقية باستخدام العلاقة :

$$r = a.c^{year/H}$$

حيث

a : الكمية الابتدائية = ٥٠ جراما

c : ثابت يساوي $e^{-.693}$

year : عدد السنوات التي انقضت

H : العمر النصفي للنظير بالسنوات

٤-٦٠) "مربع مجموع عناصر أي متسلسلة أعداد صحيحة تبدأ بالواحد $(1\ldots n)$ يساوي مجموع مكعبات هذه العناصر" ، أي أن :

$$(1+2+3+\ldots+n)^2 = 1^3 + 2^3 + 3^3 + \ldots + n^3 \qquad (*)$$

مثلا : $(1+2+3+4)^2 = 100$ & $1^3 + 2^3 + 3^3 + 4^3 = 100$

اكتب برنامجا يثبت صحة هذه العلاقة الرياضية (*) للمتسلسلات

$$(1\ldots10)\ ,\ (1\ldots11)\ ,\ (1\ldots12)\ ,\ \ldots\ ,\ (1\ldots20)$$

وذلك بأن يحسب ويطبع قيمتي طرفي العلاقة (*) ويعطي رسالة تفيد إن كانت هاتان القيمتان متساويتين ، وذلك لكل من القيم $n = 10,11,\ldots,20$.

٤-٦١) نفرض أن شخصا عنده مدخرات تساوي S دينارا قد حال عليها الحول ، وأن سعر جرام الذهب يساوي P دينارا.

اكتب برنامجا يقرأ قيمة كل من P , S ويحسب ويطبع ما يلي :

(أ) زكاة هذه المدخرات ، وقيمة المدخرات المتبقية بعد إخراج الزكاة.

زكاة المدخرات المتبقية عندما يحول عليها حول آخر ، والمدخرات المتبقية (بعد حولين) بعد إخراج هذه الزكاة. افرض أن سعر جرام الذهب لم يتغير.

تكرار الخطوة (ii) – أي حساب الزكاة المستحقة والمدخرات المتبقية في نهاية كل سنة – إلى أن يصل العدد الكلي للسنوات إلى n (حيث n عدد صحيح يقرأ البرنامج قيمته).

(ب) نفرض أن هذا الشخص لم يُخْرِج زكاة أمواله المدخرة S لعدة سنوات متتالية عددها n ، ثم تاب إلى الله تعالى وقرر إخراج الزكاة الكلية المستحقة عليه عن هذه السنوات. احسب واطبع قيمة هذه الزكاة الكلية المستحقة TZ.

٤-٦٢) يشتمل "جدول محاسبة النفس اليومي" في المسألة رقم ١-٢٥ (تمرينات رقم ١) على اثنين وعشرين سؤالا ، إجابة أي منها : نعم أو لا.

المطلوب : كتابة برنامج يقرأ رقم كل سؤال (number) من السؤال رقم ١ إلى السؤال رقم ١٥ والإجابة عليه (Answer) (حيث الإجابة هي أحد الحرفين : Y ويعني نعم ، أو N ويعني لا) ، ويحسب عدد الأسئلة (Count) التي أجيبت بالاثبات (Y).

٤-٦٣) اكتب برنامجا يقرأ خمسين عددا حقيقيا موجبا تمثل مجموعة من القياسات (measurements) ، ثم يوجد النسبة المئوية من هذه القياسات التي تزيد قيمة كل منها عن 10.0 ، ويطبع النتيجة في الصيغة التالية :

xxx.x PERCENT OF MEASUREMENTS ARE GREATER THAN 10.0

٤-٦٤) يمكن تمثيل العديد من الدوال الرياضية بالمتسلسلات اللانهائية (infinite series) . فمثلا يمكن كتابة دالة الجيب بالصورة التالية :

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

وعندما يشتمل مفكوك المتسلسلات (series expansion) على مضروبات (factorials) ، فعادة يكون من الأكفأ اتباع طريقة تكرارية (iterative method) للحصول على قيمة أي حد تالي بمعلومية قيمة الحد الحالي ، ففي متسلسلة الجيب نلاحظ أن العلاقة بين الحد رقم n والحد رقم n + 2 هي :

$$\text{term}(n+2) = \text{term}(n) * \frac{-x^2}{(n+1)(n+2)}$$

فمثلا يمكن تعيين قيمة الحد الخامس بضرب الحد الثالث في قيمة (x*x) – ثم القسمة على 20 [= (3+2)(3+1)] .

لاحظ أن الحد الأول في هذه المتسلسلة يساوي x ، وأن الحد الثاني يساوي صفرا وكذلك جميع الحدود زوجية الترتيب قيمها أصفار ، وبالتالي فالحدود غير الصفرية في المتسلسلة هي الحدود فردية الترتيب فقط.

اكتب برنامجا لحساب sin (2.5) باستخدام الحدود العشرة الفردية الأولى في المتسلسلة السابقة.

٤-٦٥) اكتب برنامجا لقراءة قيم مجموعة من الأعداد عددها N (حيث قيمة N هي أول ما يُقرأ كبيانات إدخال) ، ثم يقوم البرنامج بحساب وعرض كل من :

(أ) مدى القيم في مجموعة البيانات ، حيث

المدى (range) = أكبر قيمة – أصغر قيمة.

(ii) التباين (variance) في مجموعة البيانات. ولحساب التباين احسب كلا من مجموع القيم sum-o-data ومجموع مربعات القيم (sum-of-squares) في العروة (loop) الرئيسية ، وبعد انتهاء العروة استخدم المعادلة:

$$\text{sum-of-squares} = \text{variance} - (\text{sum-of-data})^2/N$$

(iii) الانحراف القياسي S (standard deviation) والذي هو عبارة عن

قياس لكيفية انحراف البيانات عن المتوسط، باستخدام المعادلة.

S^2 = variance / (N - 1)

٤-٦٦) تحتاج شركة معينة لطباعة جدول عن المواصفات الهندسية لألواحها الخشبية. وتتمثل أبعاد الألواح الخشبية في : القاعدة (base) والارتفاع (height) بالبوصة.

ويحتاج المهندسون لمعرفة المعلومات والمواصفات التالية عن الخشب :

cross - sectional area = base x height

moment of inertia = (base x height^3) / 12

section modulus = (base x height^2) / 2

ويصنع صاحب الشركة الألواح بقواعد أطوالها ٢ و ٤ و ٦ و ٨ و ١٠ و ١٢ بوصة ، وارتفاعات أطوالها ٢ و ٤ و ٦ و ٨ و ١٠ بوصة. اكتب برنامجا لطباعة جدول ذي عناوين مناسبة لبيان هذه القيم والمواصفات الهندسية المحسوبة. لا تكرر عرض أبعاد لوحين متشابهين ، فمثلا لا تكرر اللوح ذا البعدين ٢ و ٦ مع اللوح ذي البعدين ٦ ، ٢.

٤-٦٧) اكتب برنامجا لقراءة مجموعة من الأعداد الصحيحة وإيجاد وطباعة الدليل (الموقع) (index) لأول وآخر حدوث (occurrence) للعدد (١٢). ويجب أن يطبع البرنامج القيمة (٠) (صفر) إذا لم يجد العدد (١٢). والدليل هو الرقم التتابعي للقيمة (١٢) ، فمثلا إذا كان العدد (١٢) هو المدخل رقم ٨ فقط في البيانات ، فإن الدليل ٨ يجب أن يطبع كأول وآخر حدوث للعدد (١٢).

٤-٦٨) أ) اكتب برنامجا لقراءة مجموعة من درجات الامتحان التي تتراوح قيمها من ١ إلى ١٠٠ ، ثم حساب وطباعة كل من عدد الدرجات الممتازة وهي من ٩٠ إلى ١٠٠ ، وعدد الدرجات المقبولة وهي من ٦٠ إلى ٨٩ ، وعدد الدرجات غير المقبولة وهي من ١ إلى ٥٩. كما يعرض البرنامج الفئة التي تنتمي إليها كل درجة من الدرجات التي قرأها.

اختبر البرنامج بالبيانات الآتية :

63	75	72	72	78	67	80	63	
75	89	90	43	59	99	82	12	100

ب) عدل البرنامج ليعرض كذلك الدرجة المتوسطة للامتحان (عدد حقيقي) في نهاية مخرجات البرنامج.

٤-٦٩) اكتب برنامجا لقراءة البطاقات الزمنية الأسبوعية لموظفي إحدى المؤسسات. وكل موظف له ثلاث بيانات خاصة : رقم الهوية ، ومعدل الأجر في الساعة ، وعدد ساعات العمل في الأسبوع. وكل موظف يصرف له راتب مرة ونصف لكل ساعات العمل الزائدة عن ٤٠ ساعة ، وقيمة الضريبة تساوي ٣,٦٢٥ في المائة من الراتب الأصلي.

ومخرجات البرنامج يظهر فيها رقم الموظف وصافي راتبه. كذلك يطبع البرنامج إجمالي المدفوعات ، ومتوسط ما دفع للموظف الواحد.

٤-٧٠) تقوم إحدى الشركات ببيع ٤ أصناف من البضاعة : الصنف الأول رقمه ID يساوي ١ ، والصنف الثاني رقمه ٢ ، والصنف الثالث رقمه ٣ ، والصنف الرابع رقمه ٤.

اكتب برنامجا يقوم بعمل الآتي :

أ) قراءة قائمة الجرد (inventory) أي الموجودات / المخزون من كل صنف في بداية الأسبوع.

ب) متابعة المبيعات الأسبوعية (weekly sales) والمشتريات (purchases) المسجلة لكل صنف.

ج) طباعة قائمة الجرد النهائي (final inventory) أي الموجود بالمخزن من البضاعة من كل صنف في نهاية الأسبوع.

حيث يشار إلى أي عملية (transaction) بعددين من البيانات : العدد الأول يمثل رقم الصنف (عدد صحيح) ، والعدد الثاني يمثل الكمية المشتراة (عدد صحيح موجب) أو الكمية المباعة (عدد صحيح سالب).

والجرد الأسبوعي لكل صنف (في بداية الأسبوع) سيشار إليه أيضا بعددين : رقم الصنف والجرد الأَوَّلي (initial inventory) للصنف.

افرض أيضا أن هناك دائما كميات كافية تمنع نفاد المخزون من أي صنف.

ملحوظة : إدخال البيانات يجب أن يبدأ بثماني قيم تمثل المخزون ، تتبعها قيم العمليات المختلفة.

٤-٧١) عدل البرنامج السابق (٤-٧٠) بإضافة قائمة اختيار (menu) في البرنامج ، حيث العمليات التي تظهر بالقائمة في البرنامج المعدل هي :

إدخال مخزون : (E)nter Inventory ، شراء بضاعة : (P)urchase ، بيع بضاعة : (S)ell ، خروج من البرنامج : (Q)uit ، إظهار المخزون : (D)isplay

يجب ألا تستخدم الآن كميات سالبة لتمثيل البضاعة المباعة.

٤-٧٢) اكتب برنامجا لإجراء عمليات حساب التوفير (savings account transaction program) ، بحيث يقوم بتشغيل (processing) المجموعات التالية من البيانات.

المجموعة الأولى			المجموعة الثانية		
I	1234	1054.07	I	5723	2008.24
W		25.00	W		15.55
D		243.35	Z		
W		254.55			
Z					

المجموعة الثالثة			المجموعة الرابعة		
I	2814	128.24	I	7234	7.77
W		52.48	Z		
D		13.42			
W		84.60			
Z					

المجموعة الخامسة			المجموعة السادسة		
I	9367	15.27	I	1134	12900.00
W		16.12	D		9270.00
D		10.00	Z		
Z					

السجل (السطر) الأول في كل مجموعة يحتوي على رمز الشفرة "I" ورقم الحساب الجديد والرصيد الابتدائي لهذا الحساب ، ثم السطور التالية تحتوي على سجلات العمليات المختلفة على هذا الحساب ، من سحب (W)ithdrawal وإيداع (D)eposit. والسطر الأخير يحتوي على قيمة مميزة (حارسة) (sentinel value) (Z) تشير إلى انتهاء العمليات على هذا الحساب. ويقوم البرنامج بعرض رقم الحساب والرصيد النهائي بعد تشغيل جميع سجلات المجموعة. وإذا أصبح أي رصيد سالبا فاطبع رسالة تفيد ذلك ، واتخذ أي خطوات تصحيحية مناسبة. وإذا لم يكن هناك أي عمليات بالنسبة لحساب معين فاطبع رسالة تفيد ذلك. وأخيرا أدخل رمز الشفرة (Q)uit ليفيد الخروج من تنفيذ البرنامج.

٤-٧٣) اشترك أربعة أشخاص في سباق الميل (mile race) للجري ، حيث مسافة السباق تساوي ميلا واحدا. اكتب برنامجا لقراءة الوقت الذي استغرقه كل متسابق / عداء (runner) بالدقائق (Minutes) والثواني (Seconds) ، ثم احسب واطبع قيمة السرعة بالقدم في الثانية (FPS) وبالمتر في الثانية (MPS) ، وذلك لكل متسابق.

(ملاحظة : الميل الواحد يساوي ٥٢٨٠ قدما ، والكيلومتر الواحد يساوي ٣٢٨٢ قدما)

Minutes	Seconds
3	52.83
3	59.83
4	00.03
4	16.22

٤-٧٤) اكتب برنامجا لطباعة جدول الضرب التالي بالصورة المعطاة :

```
1   x   1   =   1
1   x   2   =   2
    ⋮
1   x   9   =   9
2   x   1   =   2
    ⋮
```

```
2   x   9   =   18
        ⋮
9   x   1   =    9
        ⋮
9   x   9   =   81
```

٤-٧٥) اكتب برنامجا لطباعة جدول لقيم الدالة :

$$f \equiv f(x,y) = \begin{cases} \dfrac{e^{x}-1}{x}.e^{y^2} & ; \quad x \neq 0 \\ e^{-y^2} & ; \quad x = 0 \end{cases}$$

وذلك لجميع قيم x , y **التالية :**

x = -5 , -4 , … , -1 , 0 , 1 , 2 , … , 6
y = 1 , 1.5 , 2 , 2.5 , … , 5

ملاحظة : البرنامج يقوم بتوليد (وليس قراءة) جميع قيم (x , y) **:**

(-5 , 1) , (-5 , 1.5) , … , (-5 , 5) ,
(-4 , 1) , (-4 , 1.5) , … , (-4 , 5) ,
…..
(6 , 1) , (6 , 1.5) , … , (6 , 5)

ويطبع القيم المقابلة للدالة f.

الفصل الخامس

الدوال
Functions

تمهيد

أثناء كتابة بعض البرامج نحتاج لتكرار عملية معينة أكثر من مرة في مواضع مختلفة من البرنامج نفسه، وقد تستلزم هذه العملية كتابة مجموعة من العبارات (ربما مع مجرد تغيير بعض القيم أو تغير أسماء بعض المتغيرات)، وبتكرار كتابة هذه العبارات يكون البرنامج طويلاً كما أنه يستلزم حجز مواضع عديدة للتخزين في وحدة الذاكرة الرئيسية . ولاختصار خطوات البرنامج وتحسين كفاءته وتوفير مواضع التخزين وأيضاً لسهولة متابعة البرنامج وفهمه فإننا نقوم بفصل العبارات– التي تنفذ خطوات هذه العملية المتكررة – خارج جزء عبارات البرنامج الرئيسي (الدالة الرئيسة main) ، ونسميها برنامجاً فرعياً (Subprogram) أو برنامجاً مساعداً تابعاً للبرنامج الرئيسي أو دالة مساندة ، أو دالة معرفة بالمستخدِم، ونعطي هذا البرنامج الفرعي (أي العملية المراد تكرارها) اسماً مميزاً ، وكلما احتجنا تكرار هذه العملية في البرنامج الرئيسي ذكرنا الاسم المميز فقط دون كتابة كل خطوات العملية، أي أن هذه الخطوات لا تكتب إلا مره واحدة فقط وذلك في البرنامج الفرعي وكلما احتجنا تنفيذها أشرنا إلى اسمها . وحين نشير في البرنامج الرئيسي إلى اسم البرنامج الفرعي نشير كذلك أحياناً إلى القيم الفعلية أو الأسماء الفعلية التي نريدها لمتغيرات البرنامج الفرعي، كما ستوضح ذلك بإذن الله تعالى الأمثلة التي ستذكر فيما بعد.

كذلك يمكن تسهيل حل بعض المسائل الطويلة بتقسيمها إلى مجموعة مسائل قصيرة يمكن حل بعضها ببرامج قصيرة (تسمى برامج فرعية) يكتبها الشخص نفسه أو يكتبها مبرمج آخر أو تكون جاهزة في مكتبة الحاسب (والتي تشتمل على عدة برامج جاهزة يحتاج إليها عدد كبير من الناس)، وترتبط هذه البرامج الفرعية بالبرنامج الرئيسي لحل المسألة الأصلية الطويلة.

وقد استخدمنا دوال (functions) C++ منذ أن تعاملنا مع البرامج / الدوال المكتبية القياسية (standard library routines) مثل sqrt, abs في الفصل الثاني، حيث نستدعي أيا من هذه البرامج الفرعية (subprograms) لتأدية وظيفة معينة . وفي هذا الفصل نشرح كيف يقوم المبرمج بكتابة دواله الخاصة خلاف الدالة الرئيسية main، وهذه الدوال يطلق عليها "البرامج الفرعية المعرفة بالمستخدِم" (user-defined subprograms) ، بنوعيها : الدوال التي تعيد قيمة (value-returning functions) والدوال الخاوية / الملغاة / الفارغة (void functions) [وهذه الأخيرة هي التي تسمى إجراءات (procedures) في بعض لغات البرمجة الأخرى].

أما الدالة التي تعيد قيمة فإنها تستقبل (receives) بعض البيانات عن طريق قائمة وسطائها (argument list) ، وتحسب قيمة دالية وحيدة (single function value) ، وتعيد هذه القيمة للبرنامج / لقطعة البرنامج التي استدعتها (calling code) . ويمكننا استدعاء دالة من هذا النوع [الدالة التي تعيد قيمة] عن طريق ذكر اسمها وقائمة وسطائها في تعبير (expression) ، مثل

```
y = 3.8 * sqrt (x);
```

وأما الدالة الخاوية [الإجراء في بعض اللغات] فإنها لا تعيد قيمة دالية، ولا تُستدعى في تعبير ، وإنما تستدعي بعبارة كاملة مستقلة قائمة بذاتها (complete, stand-alone statement) ، مثل

```
cin.get (inputChar);
```

[الدالة get المرتبطة بطبقتي (classes) istream, ifstream]. وسنبدأ بإذن الله بالتركيز على هذا النوع من الدوال المعرفة بالمستخدم: الدوال الخاوية (void functions) ، ثم ننتقل بعد ذلك إلى النوع الآخر: الدوال التي تعيد قيمة (value-returning functions) .

وعموماً إذا كانت خطوات العملية التي يقوم بإجرائها البرنامج الفرعي (الدالة المعرفة بالمستخدم) قليلة – وقد تكون سطراً واحداً – فقد يكون من الأفضل كتابتها مباشرة في البرنامج الرئيسي (الدالة الرئيسية main) بدلاً من كتابتها في برنامج فرعي ثم استدعاؤه ، وذلك مراعاة لوضوح البرنامج الكلي وبساطته وسهولة قراءته .

أولاً : الدوال الخاوية
Void Functions

كما ذكرنا سابقاً فإن الدالة الخاوية هي الدالة التي لا تعيد قيمة [أي فارغة / خاوية من قيمة تعيدها] لمستدعيها (caller). والدالة الخاوية تشبه الدالة الرئيسية main باستثناء ما يلي:

- تستخدم كلمة void بدلاً من كلمة int كنوع بيانات الدالة (data type of the function) في مقدمتها / عنوانها (function heading).
- جسم الدالة لا يحتوي على عبارة مثل: return 0; .
- لا تعيد قيمة لمستدعيها.

والمثال التالي يوضح ذلك وهو يستخدم دالتين خاويتين :

مثال ٥-١: نفرض أن أحد إخوانك قد عاد بسلامة الله بعد رحلة طويلة جهاداً في سبيل الله أو طلباً للعلم . اكتب برنامجاً Welcome program للترحيب به، وذلك بطباعة الرسالة (message) التالية:

```
* * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * *
          Welcome Home !
* * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * *
```

استخدم دالة خاوية Print2Lines لطباعة سطري النجوم الأولين ، ودالة خاوية أخرى Print4Lines لطباعة السطور الأربعة الأخيرة.

الحل:

```
//  ****************************************************************
 // Welcome program
 // This program prints a "Welcome Home" message
// ****************************************************************
#include <iostream>

using namespace std;

void Print2Lines();                  // Function prototypes
void Print4Lines;()

int main()
{
   Print2Lines();                    // Function call
   cout << " Welcome Home!" << endl;
   Print4Lines();                    // Function call
   return 0;
}

// ****************************************************************

void Print2Lines()                   // Function heading

  // This function prints two lines of asterisks
{
   cout << "***************" << endl;
   cout << "***************" << endl;
}
```

```
// ****************************************************************

void Print4Lines()                    // Function heading

 // This function prints four lines of asterisks

}
   cout >> "**************" >> endl;
   cout << "**************" << endl;
   cout << "**************" << endl;
   cout << "**************" << endl;
{
```

نلاحظ أن كلا من الدالتين الخاويتين تحتوي على قائمة وسطاء فارغة (empty argument list) .

تعريف الدالة (function definition)

يطلق هذا الاسم "تعريف الدالة" على قطعة البرنامج التي تبدأ بمقدمة الدالة (function heading) **وتنتهي بنهاية قالب** (block) **جسم الدالة** (function body) . **وتبدأ مقدمة الدالة بالكلمة** void **التي تخبر المترجم** (compiler) **أن هذه الدالة لا تعيد قيمة. وأما جسم الدالة فينفذ بعض العبارات المعتادة. و لاينتهي بعبارة** return **لإعادة قيمة دالية .**

واسم الدالة – كأي اسم تعريفي (identifier) **– لا يمكن أن يحتوي على أي فراغات** blanks **. وبعد اسم الدالة توجد قائمة وسطاء فارغة ، حيث لا يوجد أي شئ بين القوسين** parentheses **. وسنرى بإذن الله فيما بعد ماذا نكتب بين القوسين حينما تستخدم الدالة وسطاء** (arguments).

ومن الممكن أن تظهر تعريفات الدوال في أي مواضع / بأي ترتيب . فمثلاً في البرنامج السابق (مثال ٥-١) كان يمكننا أن نضع الدالة الرئيسية main في النهاية بدلاً من البداية، ولكن المعتاد عند المبرمجين بلغة C++ وضع الدالة main في البداية تليها أي دوال مساندة بعد ذلك.

وفي هذا البرنامج السابق (برنامج الترحيب Welcome program) تُسمى كل من العبارتين اللتين تسبقان الدالة main مباشرة: نموذج دالة **(function prototype)** . وهذان الإعلانان (declarations) يجب أن يظهرا في هذا الموضع قبل الدالة main لأن هناك قاعدة (rule) في لغة C++ تتطلب أن نعلن عن أي اسم تعريفي قبل استخدامه. ونظراً لأن الدالة main تستخدم الاسمين التعريفيين Print2Lines , Print4Lines ولكن تعريف كل من هاتين الدالتين لا يظهر إلا بعد الدالة main ، لذلك وجب علينا أن نكتب هذين النموذجين لإخبار المترجم (compiler) مقدماً أن Print2Lines , Print4Lines هما اسما دالتين لا تعيدان أي قيم دالية وليس لهما أي وسطاء (arguments). وسنعود بإذن الله مرة أخرى في هذا الفصل لإلقاء مزيد من الضوء على موضوع نماذج الدوال (function prototypes).

الدوال المعرَّفة بالمستخدِم
User-Defined Functions

ذكرنا سابقاً أن تعريفات دوال C++ يمكن أن تكتب بأي ترتيب، وإن كان من المعتاد أن تظهر الدالة main في المقدمة. وأثناء عملية الترجمة (compilation) فإن الدوال تترجَم (translated) بالترتيب نفسه الذي تظهر به فيزيائيا (physically appear). أما عند تنفيذ البرنامج فإن التحكم (control) ينتقل إلى أول عبارة في الدالة main. ويستمر البرنامج في التتابع المنطقي (logical sequence). وعندما يصادف استدعاءً لدالة فإن التحكم المنطقي ينتقل إلى أول عبارة في جسم تلك

الدالة. ثم تنفَّذ عبارات الدالة بترتيب منطقي، وبعد تنفيذ أخر عبارة يعود التحكم إلى النقطة التي تلي مباشرة استدعاء الدالة. ونظراً لأن استدعاءات الدوال تغير الترتيب المنطقي للتنفيذ فإن الدوال تُعتبر بنيات تحكم (control structures) . فقد يكون لدينا مثلاً ثلاث دوال A, B, C, مكتوبة بهذا الترتيب الفيزيائي (physical order) : A, B, C, ولكنها تنفذ بالترتيب C, B, A .

وسطاء الدالة (Function Parameters)

نلاحظ في برنامج الترحيب (Welcome Program) (مثال ٥-١) أن الدالتين Print4Lines , Print2Lines متشابهتان (similar) ، حيث تقومان بطباعة عدد من السطور من النجوم ، ولكن هذا العدد يختلف : 4 ,2 . أي أنهما تؤديان المهمة نفسها ولكن مع اختلاف هذا العدد . ولذلك فيمكننا كتابة دالة واحدة فقط تطبع أي عدد من السطور، ثم نستدعيها مرتين: مرة لطباعة سطرين، وأخرى لطباعة أربعة سطور. ولذا فإن الدالة التي نكتبها يكون فيها "أي عدد من السطور" numLines وسيطاً (parameter) حيث تمرِّر الدالة المستدعية (main) (caller) هذا العدد كوسيط (argument) . كما يوضح ذلك المثال التالي:

مثال ٥-٢: أعد حل مثال ٥-١ ولكن بكتابة دالة مساندة واحدة فقط PrintLines(numLines) ذات وسيط واحد numLines يمثل عدد سطور النجوم التي تطبعها الدالة. وتقوم الدالة الرئيسية main باستدعاء هذه الدالة المساندة مرتين: أولاهما لطباعة سطرين والأخرى لطباعة أربعة سطور.

الحل:

```
// ************************************************************
 // NewWelcome program
 // This program prints a "Welcome Home" message
// ************************************************************
```

```
#include <iostream>

using namespace std;

void PrintLines( int );                 // Function prototype

int main()
{
   PrintLines(2);
   cout << " Welcome Home!" << endl;
   PrintLines(4);
   return 0;
}

// ****************************************************************

void PrintLines( int numLines)

 // This function prints lines of asterisks, where
 // numLines specifies how many lines to print
{
   int count;      // Loop control variable

   count = 1;
   while (count <= numLines)
    {
      cout << "***************" << endl;
      count++;
     }
}
```

ملاحظة : في مقدمة الدالة / عنوان الدالة (function heading) PrintLines **كتبنا بين القوسين:** int numLines **، وهذا إعلان عن وسيط** (parameter declaration). **وعموماً فإن الوسطاء** (parameters / arguments) **يمكِّنون الدالة المستدعية من إدخال أو تمرير قيم لدالة أخرى (الدالة المستدعاة) لتستخدمها في عملياتها / تشغيلها** (processing) **، وفي بعض الأحيان يسمحون للدالة المستدعاة**

(called function) أن تُخرج / تعيد نتائج للدالة المستدعية . وعادة يُطلَق على كل من العناصر (items) المذكورة في قائمة استدعاء دالة: وسيط / وسيط فعلي argument / actual argument / actual parameter وأما تلك المذكورة في مقدمة / عنوان الدالة فيطلق على كل منها : وسيط / وسيط شكلي: parameter / formal argument / formal parameter

ونلاحظ أن الدالة main دالة ليس لها أي وسطاء (parameterless function).

وعموماً فإن الوسيط الفعلي قد يكون تعبيراً : إما أن يكون ثابتاً (constant) مثل 2 أو 4 (كما في المثال السابق : مثال ٥-٢) ، أو أن يكون متغيراً (كما في المثال التالي) ، أو أن يكون تعبيراً عاماً.

مثال ٥-٣: أعد كتابة الدالة main المكتوبة في حل المثال السابق، ولكن مستخدماً متغيراً (variable) بدلاً من ثابت (constant) كوسيط فعلي في كل من عبارتي استدعاء الدالة PrintLines.

الحل:

```
int main ( )
{
        int lineCount;

        lineCount  = 2;
        PrintLines (lineCount);
        cout << " Welcom Home !" << endl;
        LineCount = 4;
        PrintLines (LineCount);
        return 0;
}
```

نلاحظ من هذا الحل أنه عندما نمرر متغيراً كوسيط فعلي ، فإنه يمكن أن يكون للوسيط الشكلي والوسيط الفعلي اسمان مختلفان (مثل lineCount, numLines) . وعند كل استدعاء للدالة PrintLines (في المثال الأخير) فإن نسخة

(copy) من قيمة الوسيط الفعلي (lineCount) تمرَّر للدالة لتعطي قيمة ابتدائية (initialize) للوسيط الشكلي (numLines) . كما يوضح هذا المثال (مثال ٥-٣) إحدى فوائد استخدام الدوال-والتي أشرنا إليها سابقاً - وهي إمكانية استدعاء الدالة عدة مرات من عدة مواضع في الدالة main (أو من أي دوال أخرى) ، وهذه الاستدعاءات المتعددة توفر لنا كتابة عبارات الدالة عدة مرات.

وإذا احتاجنا أن نمرر أكثر من وسيط واحد (one argument) لدالةٍ ما ، فإن الوسطاء الفعليين والشكليين يجب أن يتواءموا (match) بناءً على مواضعهم النسبية (relative positions) في قائمة الوسطاء: تواؤم عدد الوسطاء وأنواعهم. فمثلاً إذا أردنا من الدالة PrintLines أن تطبع عدداً من السطور من رمز معين نختاره [ليس ضرورياً أن يكون النجمة (asterisk)] ، فيجب أن يكون هناك وسيطان : وسيط يمثل عدد السطور المطلوب طباعتها ، ووسيط آخر يمثل الرمز المطلوب اختياره. ولذلك نعيد كتابة عنوان الدالة PrintLines هكذا:

```
void PrintLines ( int numLines,
                  char whichChar )
```

ويمكن أن نستدعي الدالة بعبارة مثل:

```
PrinytLines ( 3,  '#');
```

والتي تطلب طباعة ثلاثة سطور من الرمز '#'.ونلاحظ أن الوسيط الفعلي الأول 3 (first argument) يتفق / يتواءم مع الوسيط الشكلي الأول (first parameter) numLines ، والوسيط الفعلي الثاني '#' يتواءم مع الوسيط الشكلي الثاني whichChar (second parameter)

استدعاء الدالة Function Call (Invocation)

كذلك نلاحظ أنه عند استدعاء الدالة الخاوية (void function) فإننا نذكر اسمها ثم الوسطاء الفعليين بين قوسين وذلك في عبارة مستقلة. أي أن الصيغة العامة لاستدعاء الدالة الخاوية هي :

```
FunctionName (ArgumentList);
```

حيث ArgumentList هي قائمة الوسطاء وصيغتها العامة هي:

Expression, Expression, ….

- وإذا لم يكن للدالة أي وسيط – أي إذا كانت قائمة الوسطاء فارغة – فإننا نكتب أيضاً القوسين دون أي شئ بينهما.
- وإذا كان هناك أكثر من عنصر / تعبير واحد في قائمة الوسطاء الفعليين (arguments) فإننا نفصل بين كل اثنين منهما بفاصلة (comma).

الإعلان عن الدالة وتعريفها
(Function Declaration and Definition)

في لغة C++ يجب أن نعلن (declare) عن أي اسم تعريفي (identifier) قبل أن يمكننا استخدامه. وفي حالة الدوال يجب أن يكون الإعلان عن الدالة (function's declaration) سابقاً–فيزيائياً – (physically precede) لأي استدعاء للدالة.

والإعلان عن الدالة يعلن للمترجم (compiler) عن اسم الدالة، ونوع بيانات القيمة التي تعيدها الدالة (إما خالية void أو نوع بيانات مثل int, float) ، وأنواع بيانات وسطائها (parameters). وبرنامج NewWlecome program في المثال الأخير (مثال ٥–٣) تظهر به ثلاثة إعلانات عن الدوال : الإعلان الأول هو العبارة:

```
void PrintLines ( int ) ;      // Function prototype
```

وهو لا يشتمل على جسم الدالة . وأما الإعلانان الآخران – للدالتين main, PrintLines – فهما يحتويان على جسمي الدالتين.

وباصطلاحات لغة C++ فإن إعلان الدالة الذي لا يحتوي على جسم الدالة يطلق عليه نموذج دالة (function prototype) ، وأما ذلك الذي يحتوي على

جسم الدالة فيسمى تعريف الدالة (function definition) ، بمعنى أن كل تعريفات الدوال إعلانات دوال ولكن ليست كل إعلانات الدوال تعريفات دوال.

وسواء كنا نتحدث عن دوال أو متغيرات ففي لغة C++ يصبح الإعلان تعريفاً إن كان يخصِّص أيضاً حيزاً من الذاكرة (allocates memory space) لهذا الذي نتحدث عنه (دالة أو متغير) . [هناك إستثناءات لهذه القاعدة ولكن لا نهتم بها الآن]. فمثلاً نموذج دالة ما (a function prototype) هو مجرد إعلان (declaration) ، أ ي أنه يحدد مواصفات / خواص الدالة (properties of the function) : اسمها ، ونوع بياناتها، ونوع بيانات وسطائها. ولكن تعريف الدالة (function definition) يفيد أكثر من هذا ، حيث أنه يجعل البرنامج المترجم (compiler) يخصص حيِّزاً من الذاكرة للتعليمات (instructions) الموجودة في جسم الدالة. [من الناحية الفنية (technically) فإن جميع الإعلانات عن المتغيرات التي استخدمناها حتى الآن كانت أيضاً تعريفات للمتغيرات بالإضافة إلى كونها إعلانات، حيث أنها كانت تخصص حيِّزاً من الذاكرة للمتغير. وسنرى بإذن الله فيما بعد أمثلة لإعلانات عن متغيرات ولكنها ليست تعريفات لهذه المتغيرات].

والقاعدة العامة في لغة C++ هي انه يمكن الإعلان عن أي شئ (an item) أي عدد نشاء من المرات ، ولكن لا يمكن تعريفه إلا مرة واحدة فقط. فمثلاً في برنامج NewWelcome (في مثال ٥-٣) كان يمكننا أن نكتب عدة نماذج دالة (function prototypes) للدالة PrinterLines (وإن كنا لانحتاج ذلك) ولكن لا يُسمح إلا بكتابة تعريف دالة (function definition) واحد فقط.

نماذج الدوال (Function Prototypes)

ذكرنا أن تعريف الدالة الرئيسية main عادة يظهر أولاً في البرنامج ، يليه تعريفات جميع الدوال الأخرى. ولتحقيق متطلبات القاعدة : أن أي اسم تعريفي

يجب أن يُعلن عنه (declared) قبل أن يُستخدم، فالمُتَّبَع عادة هو أن نضع جميع نماذج الدوال (function prototypes) قرب مطلع البرنامج قبل تعريف الدالة main.

ونموذج الدالة [والذي يعرف في بعض لغات البرمجة باسم : الإعلان الأوَّلي (forward declaration)] يحدِّد مقدماً نوع بيانات (data type) قيمة الدالة التي ستعاد (returned) (أو الكلمة void) ، وأنواع بيانات الوسطاء.

والصيغة العامة لنموذج (prototype) الدالة الخاوية هي:

```
void FunctionName (ParmeterList);
```

ولا يوجد أي جسم للدالة، وتوجد فاصلة منقوطة في نهاية الإعلان. وأما قائمة الوسطاء ParameterList فهي اختيارية ، فقد تكون هذه القائمة خالية في حالة الدوال التي ليس لها أي وسطاء. وفي حالة وجود هذه القائمة فإن:

الصيغة العامة لقائمة وسطاء نموذج الدالة (function prototype) هي:

```
DataType & VariableName, DataType & VariableName, ….
```

والعلامة & (ampersand) التي تلصق / تلحق (attached) باسم نوع بيانات DataType اختيارية ولها معنى خاص سنشير إليه فيما بعد في هذا الفصل.

وقائمة الوسطاء في نموذج الدالة يجب أن تحدد أنواع بيانات الوسطاء أما أسماء الوسطاء فهي اختيارية . فمثلاً يمكننا أن نكتب

```
void DoSomething ( int , float );
```

أو نكتب

```
void DoSomething ( int velocity , float angle );
```

وأسماء الوسطاء تفيد في التوضيح والتوثيق (documentation) ولكن البرنامج المترجم (compiler) يهملها.

تعريف الدالة (Function Definition)

كما ذكرنا سابقاً فإن تعريف الدالة يتكون من جزئين : عنوان الدالة (heading) وجسم الدالة (body) الذي هو عبارة عن قالب / عبارة مركبة. وبالتالي يمكننا أن نقول أن الصيغة العامة لتعريف الدالة الخاوية هي:

```
void FunctionName ( ParameterList)
{
        ⋮
        Statements
        ⋮
}
```

ملاحظات :

*) عنوان الدالة لا ينتهي بفاصلة منقوطة بينما نموذج الدالة ينتهي بفاصلة منقوطة.

*) في قائمة الوسطاء ParameterList يجب أن نذكر اسم كل وسيط ، بينما كان الاسم اختيارياً في نموذج الدالة.

*) يمكن أن يُكتب كل وسيط على سطر مستقل ، أو يُكتب أكثر من وسيط على السطر نفسه.

والصيغة العامة لقائمة الوسطاء في تعريف الدالة هي:

```
DataType & VariableName, DataType & VariableName, …..
```

المتغيرات المحلية (Local Variables)

المتغير المحلي هو المتغير المعلَن عنه في قالب block معين ، ولا يمكن استخدامه أو الوصول إليه (accessible) خارج هذا القالب.

فنظراً لأن جسم أي دالة عبارة عن قالب ، فيمكن لأي دالة – وليس فقط للدالة main – أن تحتوي في جسمها على إعلانات لمتغيرات. وهذه المتغيرات يطلق

عليها متغيرات محلية لأنها لا يمكن الوصول إليها إلا من داخل هذا القالب الذي أعلن عنها فيه. وأما بالنسبة للدالة المستدعية فهذه المتغيرات تعتبر غير موجودة، بمعنى أننا إذا حاولنا مثلاً أن نطبع محتويات أحد هذه المتغيرات المحلية من دالة أخرى فسينشأ لدينا خطأ وقت الترجمة (compile-time error) مثل "UNDECLARED IDENTIFIER" . وقد مر علينا مثال لهذه المتغيرات المحلية في برنامج NewWelcome program (مثال ٥-٢) ، حيث أعلنا عن المتغير المحلي count في الدالة PrintLines.

وفي مقابل المتغيرات المحلية توجد لدينا المتغيرات العامة / الشاملة (global variables) ، وهي المتغيرات التي يعلن عنها خارج جميع الدوال في برنامج ما. وسنعود بإذن الله لموضوع المتغيرات الشاملة فيما بعد.

وأما المتغيرات المحلية فإنها لا تشغل حيزاً من الذاكرة إلا أثناء تنفيذ الدالة. فمنذ لحظة استدعاء الدالة يخصَّص حيز في الذاكرة للمتغيرات المحلية في الدالة. وعندما تعود الدالة- بعد انتهاء تنفيذها- فإن متغيراتها المحلية تُمحى / تُدمَّر (destroyed) [هناك استثناء لهذه القاعدة سنشير إليه بإذن الله فيما بعد] . ولذلك فكلما تُستدعى الدالة فإن متغيراتها المحلية تبدأ بقيم غير محددة (undefined) . ونظراً لأن أي استدعاء للدالة مستقل عن أي استدعاء آخر للدالة نفسها فيجب إعطاء القيم الابتدائية للمتغيرات المحلية من داخل الدالة نفسها. ونظرا لأن المتغيرات المحلية تُمحى عندما تعود الدالة فلا يمكن استخدام المتغيرات المحلية لتخزين أي قيم بين استدعاء وآخر للدالة.

مثال ٥-٤ : قطعة البرنامج التالية توضح وتعلق على كل جزء من إعلانات الدالة وكيفية استدعائها ، وبيان متغيراتها المحلية.

```
# include <iostream>

using namespace std;
```

```
void TryThis ( int, int, float );       // Function prototype

int main ( )                            // Function definition
{
   int   int1;                          // Variables local to main
   int   int2;
   float someFloat;
     ⋮
   TryThis (int1, int2, someFloat);     // Function call with three
                                        //       arguments
     ⋮
 }

 void TryThis ( int   param1,           // Function definition with
                int   param2,           //       three parameters
                float param3
{
     int   i;                           // Variables local to TryThis
     float x;
     ⋮
}
```

عبارة Return

كما أشرنا سابقاً فإن الدالة main تستخدم العبارة return 0; لإعادة القيمة 0 (أو 1 أو أي قيمة أخرى) إلى مستدعيها : نظام التشغيل. وكذلك أي دالة من هذا النوع الذي يعيد قيمة يجب أن تعيد قيمتها بهذه الطريقة.

وأما الدالة الخاوية فإنها لا تعيد قيمة دالية، وإنما يعود التحكم (control) من الدالة بعد أن يصل التنفيذ إلى نهاية جسم الدالة وتُنفَّذ آخر عبارة.

وهناك صيغة أخرى لعبارة Return ، وتبدو هكذا

```
return;
```

وهذه العبارة صحيحة فقط بالنسبة للدوال الخاوية، ويمكننا أن تظهر في أي موضع في جسم الدالة ، وهي تسبِّب خروج (exit) التحكم (control) فوراً من الدالة والعودة إلى مستدعيها ، كما يوضح ذلك المثال التالي (وهو غير عملي ولكنه للتوضيح فقط)

مثال ٥-٥ : اكتب دالة خاوية SomeFunc (n) **تختبر قيمة** n **. فإن كانت هذه القيمة أكبر من** 50 **فإن الدالة تطبع رسالة تفيد أن القيمة خارج المدى** (range)، **وتعود الدالة مباشرة دون تنفيذ أي عبارات أخرى، وما عدا ذلك فإن الدالة تغيرِّ قيمة** n **بضربها في** 412 **، ثم تطبع قيمة** n **، وتعود للدالة المستدعية.**

الحل :

```
void SomeFunc (  int n  )
{
        if (n > 50)
        {
                cout << "The value is out of range.";
                return;
        }
        n = 412 * n;
        cout << n;
}
```

حل آخر:

```
void SomeFunc (  int n  )
{
        if (n > 50)
                cout << "The value is out of range.";
        else
        {
                n = 412 * n;
                cout << n;
        }
}
```

الوسطاء (Parameters)

حين تدخل دالة مرحلة التنفيذ فإنها تستخدم الوسطاء الفعليين (arguments) **المذكورين في استدعاء الدالة. وأما تفاصيل ذلك وكيفيته فيعتمد على نوع وطبيعة الوسطاء . وهناك نوعان من الوسطاء:**

أ) وسيط ذو قيمة (value parameter) : وهو الوسيط الذي يستقبل نسخة (copy) من قيمة الوسيط الفعلي (argument) المقابل. وعند الإعلان عن وسيط ذي قيمة لا توضع علامة & (ampersand) بعد اسم نوع البيانات.

ب) وسيط إسناد / وسيط مرجع (reference parameter) : وهو الوسيط الذي يستقبل موضع / عنوان ذاكرة (location / memory address) الوسيط الفعلي (argument) في الدالة المستدعية. وعند الإعلان عن وسيط إسناد تضاف علامة & إلى اسم نوع البيانات. والمثال التالي لعنوان الدالة (function heading) يوضح الإعلان (declaration) عن نوعي الوسطاء.

```
void Example ( int&  parm1          // A reference parameter
               int   param2,        // A value parameter
               float param3)        // Another value parameter
```

ففي حالة الأنواع البسيطة للبيانات (simple data types) مثل int, char, float, ... يفترض (by default) أن يكون الوسيط وسيطاً ذا قيمة ، فإذا أردناه وسيط إسناد وجب علينا إضافة العلامة & .

أولاً : الوسيط ذو قيمة (Value Parameter)

رأينا في برنامج NewWelcome program (مثال ٥-٢) عنوان الدالة PrintLines (function heading) :

```
void PrintLines ( int numLines )
```

والوسيط numLines وسيط ذو قيمة (value parameter) لأن اسم نوع البيانات int لا ينتهي بالعلامة & . وحين نستدعي الدالة مستخدمين وسيطاً فعلياً lineCount (argument) :

```
PrintLines ( lineCount );
```

فإن الوسيط numLines يستقبل نسخة من قيمة lineCount ، وبالتالي في هذه اللحظة يكون لدينا نسختان من البيانات : واحدة في الوسيط الفعلي lineCount وأخرى في الوسيط الشكلي numLines.

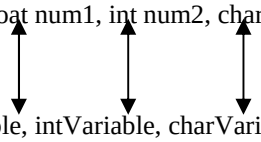
فإذا ما غيرَّت عبارة ما داخل الدالة PrintLines قيمة numLines فإن هذا التغيير لا يؤثر على الوسيط الفعلي lineCount . وبالتالي فإن استخدام الوسيط ذي القيمة يساعد في تجنب أي تغييرات غير مقصودة قد تحدث للوسيط الفعلي.

ونظراً لأن الوسيط ذا قيمة يستقبل نسخة من الوسيط الفعلي المقابل، أي تُمَرَّر له (passed) هذه النسخة، فيمكننا أن نمرر أي شئ له قيمة للوسيط ذي قيمة. فهذا الشئ قد يكون ثابتاً (constant) أو متغيراً (variable) أو تعبيراً اختيارياً معقداً ، حيث تُحسَب قيمة هذا التعبير ثم ترسل نسخة من النتيجة إلى الوسيط ذي قيمة المقابل. مثلاً الاستدعاءات التالية للدالة PrintLines كلها صحيحة:

```
PrintLines(3);
PrintLines(lineCount);
PrintLines(2 * abs (10 - someInt));
```

ويجب أن يكون عدد الوسطاء الفعليين في استدعاء الدالة هو نفسه عدد الوسطاء الشكليين في عنوان الدالة (*) . وكذلك يجب أن يكون كل وسيط فعلي متفقاً في نوع البيانات مع الوسيط الشكلي المقابل له في الموضع نفسه. لاحظ مثلاً هذا التواؤم (mtching) في الاستدعاء التالي (حيث نفترض أن نوع بيانات أي وسيط فعلي يُفهم من اسمه):

عنوان الدالة : Function heading: void ShowMatch (float num1, int num2, char letter)



استدعاء الدالة: Function call: ShowMatch (floatVariable, intVariable, charVariable)

وإذا لم يكن الوسيطان المتقابلان من نوع البيانات نفسه فتتم عملية التحويل الضمني للنوع (implicit type coercion) . مثلاً إذا كان الوسيط الشكلي من النوع int والوسيط الفعلي تعبيراً من النوع float فإن الوسيط الفعلي يحوَّل إلى قيمة int قبل أن يمرر إلى الدالة. وكالمعتاد في لغة C++ يمكننا تجنب التحويل غير المقصود للنوع وذلك عن طريق الصب / التحويل الصريح للنـــوع

(*) هناك خاصية في لغة C++ يطلق عليها "الوسطاء الافتراضيون" (default parameters) تسمح باستدعاء الدالة بعدد من الوسطاء الشكليين أقل من عدد الوسطاء الفعليين، ولكننا لن نناقش هذه الخاصية في هذا الكتاب.

(explicit type cast) أو بتجنب خلط أنواع البيانات أساساً .

ونظراً لأن الوسيط الشكلي يستقبل نسخة من الوسيط الفعلي، فلذلك لا يمكن الوصول إلى الوسيط الفعلي أو تغييره. وعندما تعود الدالة فإن محتويات أي وسيط ذي قيمة من وسطائها تُمحَى / تدمَّر مع محتويات متغيراتها المحليين. والفارق بين الوسيط ذي قيمة والمتغير المحلي هو أن قيمة المتغير المحلي تكون غير معلومة / غير معرفة (undefined) عندما تبدأ الدالة في التنفيذ بينما الوسيط ذو قيمة يُعطَى قيمة ابتدائية (initialized) هي قيمة الوسيط الفعلي المقابل.

وحيث أن محتويات أي وسيط ذي قيمة تُمحى عندما تعود الدالة، فلا يمكن استخدام الوسيط ذي قيمة ليعيد أي معلومات للدالة المستدعية. فإذا أردنا أن نعيد معلومات بتعديل قيمة الوسيط الفعلي وجب علينا استخدام النوع الثاني من الوسطاء الشكليين: وسطاء الإسناد (reference parameters).

ثانياً : وسيط الإسناد (Reference Parameter)

وسيط الإسناد / وسيط المرجعية / وسيط ذو مرجع هو وسيط نعلن عنه بإلحاق (attaching) العلامة & بنهاية اسم نوع بياناته. ويطلق عليه وسيط الإسناد أو المرجعية لأن الدالة المستدعاة يمكنها أن ترجع إلى / تششير إلى (refer to) الوسيط الفعلي المقابل مباشرة. أي يُسمَح للدالة أن تطَّلع على الوسيط الفعلي وتعدِّل قيمته.

وكما ذكرنا سابقاً فعند استدعاء دالة باستخدام وسيط إسناد فإن موضع الوسيط الفعلي (عنوانه بالذاكرة) – وليس قيمته – هو ما يُمرَّر للدالة، أي توجد نسخة واحدة فقط من المعلومات، وكلا الدالتين : المستدعية والمستدعاة تستخدمان هذه النسخة الوحيدة. وعند استدعاء دالة فإن الوسيط الفعلي والوسيط الشكلي يصبحان متطابقين / مترادفين (synonyms) من حيث الموقع نفسه في الذاكرة . فإي قيمة تتركها الدالة المستدعاة في هذا الموقع هي القيمة التي ستجدها هناك الدالة المستدعية. ولذلك فعند استخدام وسيط إسناد يجب أن

نكون حذرين لأن أي تغيير فيه نقوم بعمله سيؤثر على الوسيط الفعلي في الدالة المستدعية.

مثال ٥-٦: اكتب برنامجاً للأنشطة Activity program لقراءة درجة حرارة الهواء الطلق (outdoor temperature) التي يُدخلها المستخدم ، ثم طباعة النشاط المناسب المقترح (recommended appropriate activity) بناء على الجدول المذكور في مثال ٣-١٥ . استخدام دالتين خاويتين (void functions): إحداهما GetTemp لقراءة درجة الحرارة ثم طباعتها ، والأخرى PrintActivity تستقبل درجة الحرارة وتطبع رسالة تشير إلى النشاط المناسب.

الحل:

```
//*************************************************************
 //Activity program
 //This program outputs an appropriate activity
 //for a given temperature
//*************************************************************
#include <iostream>

using namespace std;

void GetTemp( int& );                  // Function prototypes
void PrintActivity( int);

int main( )
{
   int temperature;    // The outside temperature

   GetTemp(temperature);               // Function call
   PrintActivity(temperature);         // Function call
   return 0;
}
```

```
//*****************************************************************

void GetTemp( int& temp )                // Reference parameter

 //This function prompts for a temperature to be entered,
 //reads the input value into temp, and echo-prints it
{
   cout << "Enter the outside temperature:" << endl;
   cin >> temp;
   cout << "The current temperature is " << temp << endl;
}

//*****************************************************************

void PrintActivity( int temp )           // Value parameter

 //Given the value of temp, this function prints a message
 //indicating an appropriate activity
{
   cout << "The recommended activity is" ;
   if (temp > 85)
      cout << "swimming." << endl;
   else if (temp > 70)
      cout << "tennis." << endl;
   else if (temp > 32)
      cout << "golf." << endl;
   else if (temp > 0)
      cout << "skiing." << endl;
   else
      cout << "ping-pong." << endl;
}
```

ملاحظات:

- **اسم الوسيط الفعلي في كل من استدعاء الدالة** GetTemp **واستدعاء الدالة** PrintActivity **هو** temperature. **الوسيط الشكلي في الدالة**

GetTemp وسيط إسناد اسمه temp، والوسيط الشكلي في الدالة PrintActivity وسيط ذو قيمة واسمه أيضاً temp.

- الدالة main تخبر الدالة GetTemp أين تترك درجة الحرارة بإعطائها موضع المتغير temperature عندما تستدعيها. ويجب أن نستخدم هنا وسيط إسناد حتى تعلم الدالة GetTemp أين تضع النتيجة. فالوسيط الشكلي temp يمكن أن ننظر إليه على أنه مجرد حافظ مناسب للمكان في تعريف الدالة. وحينما تُستدعَى الدالة GetTemp بالوسيط الفعلي temperature فإن كل الإشارات للوسيط temp داخل الدالة تُوجَّه فعلياً إلى temperature . وإذا ما استدعيت الدالة مرة أخرى بوسيط فعلي آخر مختلف فإن كل الإشارات للوسيط temp توجه الآن فعلياً لهذا المتغير / الوسيط الآخر حتى تعيد الدالة التحكم إلى main.
- وأما الوسيط الشكلي للدالة PrintActivity فهو وسيط ذو قيمة. وحين تُستدعى هذه الدالة فإن الدالة main ترسل نسخة من قيمة temperature للدالة PrintActivity لتعمل بها. ومن المناسب هنا أن نستخدم وسيطاً ذا قيمة لأن هذه الدالة لا يفترض أن تُعدِّل من قيمة الوسيط الفعلي temperature.
- نظراً لأنه يمكن أن يكون للوسطاء الفعليين والشكليين أسماء مختلفة ، فيمكننا استدعاء دالة ما عدة مرات بوسطاء فعليين مختلفين . نفرض مثلاً أننا نود تعديل برنامج الأنشطة Activity program بحيث يطبع نشاطاً مقابلاً لدرجة الحرارة الداخلية (داخل البيت) (indoor temperature) بالإضافة إلى درجة الحرارة الخارجية (خارج البيت / في الهواء الطلق) (outdoor temperature) . يمكننا أن نعلن عن متغيرين صحيحين (integer variables) هما : indoorTemp, outdoorTemp في الدالة main، ثم نكتب جسم الدالة main كما يلي:

```
GetTemp ( indoorTemp);
PrintActivity ( indoorTemp);
GetTemp ( outdoorTemp);
```

```
PrintActivity (  outdoorTemp);
return 0;
```

وفي الجدول التالي نلخص الفروق بين الوسيط ذي قيمة ووسيط الإسناد.

الوسيط ذو قيمة value parameter	وسيط الإسناد reference parameter
* يظهر في عنوان الدالة، ولا ينتهي اسم نوع بياناته بالعلامة & . * يستقبل نسخة من قيمة الوسيط الفعلي المقابل. * لا يؤثر على قيمة الوسيط الفعلي * الوسيط الفعلي المقابل يمكن أن يكون متغيراً أو ثابتاً أو تعبيراً اختيارياً (قد يتم تحويل لنوع قيمة الوسيط الفعلي – عند الاختلاف وإن أمكن التحويل – إلى نوع بيانات الوسيط الشكلي)	يظهر في عنوان الدالة ، وينتهي اسم نوع بياناته بالعلامة &. يستقبل عنوان ذاكرة/موضع الوسيط الفعلي المقابل. تتطابق قيمته مع قيمة الوسيط الفعلي. الوسيط الفعلي المقابل يجب أن يكون متغيراً فقط ، ومن نوع بيانات الوسيط الشكلي نفسه بالضبط.

مثلاً إذا كان لدينا عنوان الدالة التالي:

```
void DoThis ( float val,        // Value parameter
              int& count  )     // Reference parameter
```

فإن الاستدعاءات التالية للدالة جميعها صحيحة:

```
DoThis ( someFloat, someInt);
DoThis (9.83, intCounter);
DoThis (4.9 * sqrt (y), myInt );
```

وأما عبارة الاستدعاء التالية

```
DoThis (y , 3);
```

فإنها تؤدي إلى خطأ وقت الترجمة (compile-time error) لأن الوسيط الفعلي الثاني ليس اسم متغير.

تحديد نوع الوسيط الشكلي

عموماً إذا كانت البيانات مدخلة لوسيط (incoming data flow) في دالةٍ ما ، فإننا نجعل الوسيط وسيطاً ذا قيمة، وإن كانت البيانات مخرجة (outgoing data flow) من الدالة عبر الوسيط ، جعلنا الوسيط وسيط إسناد، وكذلك إن كانت البيانات مدخلة / مخرجة (incoming / outgoing) جعلنا الوسيط وسيط إسناد. [ملاحظة: هناك استثناءات لهذه القاعدة العامة لتحديد نوع الوسيط، ولكننا لن نتعرض لهذه الاستثناءات في هذا الكتاب] .

مثال ٥-٧ : اكتب برنامجاً Graph program للمقارنة بين المبيعات الشهرية (monthly sales) لمتجرين (stores 2) من متاجر إحدى شركات الأثاث (Furniture Stores Company) ، حيث تتم المقارنة بين مبيعات كل قسم في المتجر الأول (قسم رقم deptID1 في المتجر store1) ومبيعات القسم المقابل له في المتجر الثاني (قسم رقم deptID2 في المتجر store2) ، أي أنها مقارنة بين المبيعات قسماً قسماً (department-by-department comparison) . وتُعرض نتائج المقارنة في صورة رسوم بيانية مستقيمة (bar graphs) – كما هو موضح بمثال في الشكل التالي – حيث يتكون المستقيم الذي يمثل قيمة المبيعات بالدولار من نجوم (stars) ، كل نجمة تمثل 500$ ، ولا ترُسم أي نجمة إذا كانت قيمة مبيعات القسم أقل من أو تساوي 250$.

وتحفظ قيم المبيعات اليومية (daily sales) لكل قسم في ملف حسابات المتجر (store's accounting file). وتخزن بيانات كل متجر بالصورة التالية:

الرقم التعريفي للقسم	Department ID number
عدد أيام العمل بالقسم	Number of business days for the department
قيمة المبيعات اليومية في اليوم الأول	Daily sales for day 1

قيمة المبيعات اليومية في اليوم الثاني Daily sales for day 2

قيمة المبيعات اليومية في اليوم الأخير Daily sales for last day in period

الرقم التعريفي للقسم Department ID number

عدد أيام العمل بالقسم Number of business days for the department

قيمة المبيعات اليومية في اليوم الأول Daily sales for day 1

الشكل التالي يعطي مثالاً لمخرجات البرنامج بعد تنفيذه (sample Run):

```
Bar Grph Comparing Departments of Store #1 and Store #2
Store Sales in 1,000s of dollars
#       0       5       10      15      20      25
        |.......|.......|.......|.......|.......|
        Dept 1030
1       **********
        Dept 1030
2       ***************************

        Dept 1210
1       ********************************
        Dept 1210
2       *************************

        Dept 2040
1       ******************************
        Dept 2040
2       ********************
```

الحل:

مدخلات البرنامج : ملفا بيانات (data files) : store1, store2 **يحتوي كل منهما على القيم التالية لكل قسم:**

الرقم التعريفي للقسم (int) Department ID number

عدد أيام العمل (int) Number of business days

قيمة المبيعات اليومية (عدة قيم float) Daily sales

مخرجات البرنامج: رسم بياني عبارة عن خطوط تمثل قيم المبيعات الكلية لكل قسم.

ملاحظات :

*) **لجعل البرنامج أكثر مرونة فإنه يحث المستخدم لإدخال اسمي الملفين على القرص** (names of the disk files)**، ويقرأ الاسمين كسلسلتي رموز** (strings)**، ويربط** (associates) **هاتين السلسلتين بهدفي سيل الملفين** (file stream objects) **[وسنسَمِّيهما :** [store1, store2]

*) **ونحتاج لقراءة الرقم التعريفي لقسم وعدد أيام العمل وقيم المبيعات اليومية للقسم. وبعد تشغيل بيانات أي قسم نقرأ بيانات القسم التالي، ونستمر هكذا حتى ننتهي من جميع الأقسام (أي حتى نصل إلى نهاية الملف** EOF) .

*) **ونظراً لأن عملية القراءة هي نفسها بالنسبة لكل من المتجرين** store1, store2 **، لذلك يمكننا استخدام دالة واحدة لقراءة الملفين ، وذلك بأن تمرر سيل الملف المناسب** (appropriate file stream) **كوسيط فعلي للدالة.**

*) **ونظراً لأننا نريد قيمة المبيعات الكلية لكل قسم ، لذلك تقوم هذه الدالة بجمع قيم المبيعات اليومية التي تُقرأ.**

*) **يمكننا استخدام دالة لطباعة عنوان المخرجات.**

*) **كما يمكننا استخدام دالة أخرى لطباعة قيمة المبيعات الشهرية لكل قسم بالصورة البيانية.**

```
//*************************************************************
 // Graph program
 // This program generates bar graphs of monthly sales
 // by department for two furniture stores, permitting
 // department-by-department comparison of sales
//*************************************************************
#include <iostream>
```

```
#include <iomanip>    // For setw()
#include <fstream>    // For file I/O
#include <string>     // For string type

using namespace std;

void GetData( ifstream&, int&, float& );
void OpenForInput( ifstream&);
void PrintData( int, int, float );
void PrintHeading ( );

int main()
{
   int      deptID1;      // Department ID number for Store 1
   int      deptID2;      // Department ID number for Store 2
   float    sales1;       // Department sales for Store 1
   float    sales2;       // Department sales for Store 2
   ifstream store1;       // Accounting file for Store 1
   ifstream store2;       // Accounting file for Store 2

   cout << "For Store 1," << endl;
   OpenForInput(store1);
   cout << "For Store 2," << endl;
   OpenForInput(store2);
   if ( !store1 || !store2 )                // Make sure files
      return 1;                             //   were opened

   PrintHeading ( );

   GetData(store1, deptID1, sales1);          // Priming reads
   GetData(store2, deptID2, sales2);
   while (store1 && store2)                 // While not EOF...
   {
      cout << endl;
      PrintData(deptID1, 1, sales1);          // Process Store 1
      PrintData(deptID2, 2, sales2);          // Process Store 2
      GetData(store1, deptID1, sales1);
      GetData(store2, deptID2, sales2);
   }
```

```
   return 0;
}

//*************************************************************

void OpenForInput( /* inout */ ifstream& someFile )    // File to be
                                                        //opened
 //Prompts the user for the name of an input file
 //and attempts to open the file

    //Upon return from this function, the caller must test
    //the stream state to see if the file was successfully opened
{
   string fileName;    // User-specified file name

   cout << "Input file name: ";
   cin >> fileName;

   someFile.open(fileName.c_str( ));
   if ( !someFile )
      cout << "** Can't open " << fileName << " **" << endl;
}

//*************************************************************

void PrintHeading()

 //Prints the title for the bar chart, a heading, and the numeric
 //scale for the chart.  The scale uses one mark per $500
{
cout << " Bar Graph Comparing Departments of Store #1 and Store #2"
       << endl << endl
       << " Store  Sales in 1,000s of dollars" << endl
       << " #      0         5         10        15        20        25 "
       << endl
       << " |.......|.......|.......|.......|.......| "
        << endl;
```

```
}
//****************************************************************

void GetData( /* inout */ ifstream& dataFile,   // Input file
              /* out */   int&      deptID,      // Department number
              /* out */   float&    deptSales ) // Department's
                                                 // monthly sales

 //Takes an input accounting file as a parameter, reads the
 //department ID number and number of days of sales from that file,
 //then reads one sales figure for each of those days, computing a
 //total sales figure for the month.  This figure is returned in
 //deptSales.  (If input of the department ID fails due to
 //end-of-file, deptID and deptSales are undefined.)
{
   int   numDays;  // Number of business days in the month
   int   day;      // Loop control variable for reading daily sales
   float sale;     // One day's sales for the department

   dataFile >> deptID;
   if ( !dataFile )           // Check for EOF
      return;                 // If so, exit the function

   dataFile >> numDays;
   deptSales = 0.0;
   day = 1;                   // Initialize loop control variable
   while (day <= numDays)
    {
      dataFile >> sale;
      deptSales = deptSales + sale;
      day++;                  // Update loop control variable
     }
}

//****************************************************************

void PrintData( /* in */ int   deptID,        // Department ID number
                /* in */ int   storeNum,     // Store number
```

```
                                /* in */ float deptSales )   // Total sales for the
                                                             //  department

 //Prints the department ID number, the store number, and a
 //bar graph of the sales for the department.  The bar graph
 //is printed at a scale of one mark per $500
 // with remainders over $250 rounded up
// No stars have been printed for sales <= $250
{
   cout << setw(12) << "Dept " << deptID << endl;
   cout << setw(3) << storeNum << "    ";
    while (deptSales > 250.0)
    {
      cout << '*' ;                        // Print '*' for each $500
      deptSales = deptSales - 500.0;   // Update loop control
                                           // variable
     }
   cout << endl;
}
```

نلاحظ في هذا الحل ما يلي:

١) **يشتمل البرنامج على ثلاث عروات** (3 loops):

(i) **عروة في الدالة** main **لقراءة وتشغيل بيانات الملف.**

(ii) **عروة في الدالة** GetData **لقراءة جميع قيم المبيعات اليومية لقسم واحد.**

(iii) **عروة في الدالة** PrintData **لطباعة النجوم في الرسم البياني .**

أما العروة الدالة main **فتختبر الوصول إلى نهاية الملف** EOF **في كل من** store1, store2 **. ويجب طباعة منحنى بياني لكل متجر لكل تكرير من تكريرات هذه العروة.**

وأما عروة الدالة GetData **فإنها تتطلب عداداً للتكريرات** (iteration counter) **في المدى من** 1 **إلى عدد أيام العمل في القسم. وأيضاً تحتاج**

العروة إلى عملية جمع لإيجاد قيمة مجموع المبيعات في الفترة المطلوب إجراء المقارنة فيها.

وأما عروة الدالة PrintData فلكتابة خطواتها افرض أننا نود طباعة خط (bar) يمثل القيمة 1850 . أولاً نتأكد أن القيمة أكبر من 250، ثم نطبع نجمة ، ونطرح 500 من القيمة الأصلية ، ونتحقق مرة أخرى من أن القيمة الجديدة أكبر من 250، ثم نطبع نجمة ونطرح 500. وهكذا نكرر هذه العملية حتى تصبح القيمة الناتجة أقل من أو تساوي 250. أي أن العروة تتطلب عداداً يتناقص بقدر 500 في كل تكرير، والقيمة النهائية هي 250 أو أقل. وتُطبَع نجمة في كل تكرير من تكريرات العروة.

٢) الدالة PrintHeading لاتستقبل أي قيم من الدالة main ، وكذلك لا تعيد لها أي قيم، ولذلك فإن قائمة وسطائها خالية.

٣) الدالة GetData تستقبل هدف سيل الملف (file stream object) من الدالة main وتعيده معدَّلاً (modified) بعد قراءة بعض القيم .كما أن الدالة GetData تعيد للدالة main أيضاً قيمتي الرقم التعريفي للقسم (department ID) ومبيعاته الشهرية. وبالتالي فإن GetData لها ثلاثة وسطاء: هدف سيل الملف (حيث فيض البيانات داخل وخارج: Inout data flow)، والرقم التعريفي للقسم (فيض البيانات خارج: Out data flow) ، وقيم مبيعات القسم (فيض البيانات خارج : Out data flow)

٤) الدالة PrintData يجب أن تستقبل الرقم التعريفي للقسم department ID ، ورقم المتجر store number ، وقيمة مبيعات القسم من الدالة main لتطبع خط الرسم البياني (bar graph) لسجل مدخلات (an input record) . ولذا فإن الدالة PrintData لها ثلاثة وسطاء هي هذه العناصر الثلاثة، وفيض بياناتها جميعاً داخل (In data flow).

٥) هناك دالة إضافية OpenForInput. وهذه الدالة تستقبل هدف سيل ملف، وتحث المستخدم لإدخال اسم الملف المصاحب على القرص (name of the associated disk file) ، وتحاول فتح الملف. والدالة تعيد هدف سيل الملف إلى الدالة المستدعية ، إما مفتوحاً بنجاح (successfully opened) أو في حالة الفشل (in the fail state) ، إذا لم يمكن فتح الملف. ولذا فإن الوسيط الوحيد لهذه الدالة – وهو هدف سيل الملف– له فيض بيانات داخل وخارج (Inout data flow).

الدالة المكتبية assert (The assert Library Function)

عند اختبار برنامج ما لاكتشاف أخطائه وتصحيحها (debugging) يمكننا الاستعانة بالدالة المكتبية القياسية الخاوية assert عن طريق ملف المقدمة cassert (header file). وهذه الدالة تأخذ / تستقبل شرطاً / تعبيراً منطقياً (logical / Boolean expression) كوسيط فعلي (argument) وتُوقف البرنامج (halts the program) إن كان الشرط خاطئاً. ولذلك فإننا نستخدم هذا الشرط للتأكيد (assertion) على ضرورة تحقق علاقة معينة في البرنامج وإلا كان هناك خطأ يجب تصحيحه. فمثلاً للتأكيد على أن عدد الطلاب studentCount عدد موجب يمكننا كتابة ما يلي:

```
# include <cassert>
    .
    .
assert (studentCount > 0);
average = sumOfScores / studentCount;
```

والقاعدة العامة أنه إذا كانت قيمة الوسيط الفعلي (قيمة التعبير المنطقي ture (صادقة) لا يحدث شئ ، ويستمر التنفيذ إلى العبارة التالية. أما إن كانت قيمته false (خاطئة) فإن تنفيذ البرنامج يتوقف فوراً ، وتصدر رسالة خطأ يظهر فيها:

أ) شرط التأكيد (assertion) كما ظهر في قائمة الوسطاء الفعليين.

ب) اسم الملف الذي يحتوي البرنامج المصدري (program source code).

ج) رقم السطر في البرنامج.

فمثلاً في المثال السابق إذا كان عدد الطلاب studentCount أقل من أو يساوي صفراً فإن البرنامج يتوقف بعد إعطاء رسالة خطأ مثل:

Assertion failed: studentCount > 0 , file myprog.cpp, line 48

وهذه الرسالة تعني أن الشرط studentCount > 0 خاطئ.

ويمكن للبرنامج أن يستدعي الدالة assert عدة مرات لاكتشاف مواضع أخطاء البرنامج، لأن رسالة الخطأ التي تظهر – حين فشل تأكيد الشرط (assertion is false) – تعطي رقم سطر (line number) هذا الشرط الخاطئ (failed assertion).

وإذا أردنا أن نلغي تأثير عبارات استدعاء الدالة assert دون حذف هذه العبارات – وذلك بعد تصحيح أخطاء البرنامج والتأكد من سلامته – يمكننا استخدام موجِّه التشغيل المبدئي # define (preprocessor directive) NDEBUG قبل وضع ملف المقدمة cassert ، كما يلي

```
# define NDEGUG
# include <cassert>
```

ونتيجة ذلك هي إهمال (ignoring) جميع استدعاءات الدالة assert حين تشغيل (running) البرنامج ، و NDEGUG تعني NO DEGUG " أي لا تصحيح للأخطاء (لا إزالة للعلل) . وأما لماذا نترك عبارات استدعاءات الدالة assert بعد الانتهاء من تصحيح أخطاء البرنامج والتأكد من سلامته ولا نحذف هذه العبارات، فلأننا قد نحتاج لهذه العبارات فيما بعد..

مجال الاسم التعريفي Scope of Identifier

كما ذكرنا سابقاً فإن المتغير المحلي (local variable) هو المتغير المعلَن عنه داخل قالب (block) كجسم دالة مثلا. ولا يمكن الوصول إلى (accessing) المتغيرات المحلية خارج القالب الذي يحتويهم. وتنطبق قاعدة الوصول (access rule) نفسها على الإعلانات الخاصة بالثوابت المسماة (named constants) ، أي أن : الثوابت المحلية (local constants) يمكن الوصول إليها فقط داخل القالب الذي أُعلن عنها فيه.

ويمكن لأي قالب – وليس فقط لجسم دالة (a function body)– أن يحتوي على إعلانات عن متغيرات وثوابت. فمثلاً عبارة If التالية تحتوي على قالب يعلن عن متغير محلي n :

```
If (alpha > 3)
{
        int n;

        cin >> n;
        beta = beta + n;
}
```

وكما هو الحال بالنسبة لأي متغير محلي فإن n لا يمكن الوصول إليها / استخدامها بعبارة خارج القالب الذي يحتوي على الإعلان عن n.

وإذا عدَّدنا جميع المواضع التي يُسمح لنا منها استخدام (/ الوصول إلى) اسم تعريفي معين فتلك المواضع هي ما يطلق عليه: مجال رؤية الاسم التعريفي (identifier's **scope of visibility**) ، أو مجال الوصول إليه (**scope of access**)، أو اختصاراً مجال (scope) الاسم التعريفي. أي أن:

مجال الاسم التعريفي **(scope of identifier)** : هو منطقة البرنامج التي يُسمح لنا فيها استخدام / الإشارة إلى الاسم التعريفي.

ولغة C++ تعرِّف عدة أنواع من المجالات لأي اسم تعريفي، ونبدأ بوصف ثلاثة منها:

(١) **المجال الطبقي (Class scope) : وهذا الاصطلاح يشير إلى نوع البيانات المسمَّى : "طبقة" (class) ، وقد أشرنا إليه باختصار في الفصل الثاني ، ولن نتعرض لتفاصيله في هذا الكتاب.**

(٢) **المجال المحلي (Local scope): مجال أي اسم تعريفي أعلن عنه داخل قالب معين يمتد من نقطة الإعلان إلى نهاية القالب. وكذلك فإن مجال أي وسيط شكلي (formal parameter) لدالة ما يمتد من نقطة الإعلان عنه إلى نهاية القالب الذي هو جسم الدالة.**

(٣) **المجال العام / الشامل (Global scope) : مجال أي اسم تعريفي أعلن عنه خارج جميع الدوال والطبقات يمتد من نقطة الإعلان إلى نهاية الملف كله الذي يحتوي على البرنامج.**

وأسماء دوال C++ مجالها شامل (global)* . فبمجرد الإعلان عن اسم دالة يمكن لأي دالة أخرى في بقية البرنامج أن تستدعيها. ولا يوجد في لغة C++ ما يمكن أن يطلق عليه "دالة محلية" (local function) ، بمعنى أنه لا يمكننا وضع تعريف دالة (function definition) داخل تعريف دالة أخرى.

والمتغيرات والثوابت الشاملة هي تلك التي نعلن عنها خارج جميع الدوال. وفي قطعة البرنامج التالية تعد gamma متغيراً شاملاً، ويمكن الوصول إليه مباشرة بأي عبارة في الدالة main أو في الدالة SomeFunc .

```
int  gamma;                    // Global variable

int   main ( )
{
        gamma = 3;
        .
        .
        .
}
```

(*) هناك استثناء لهذه القاعدة حين التعامل مع طبقات (classes) C++، ولن نتعرض له في هذا الكتاب.

```
void SomeFunc ( )
{
        gamma = 5;
        ⋮
}
```

أولوية الأسماء / إخفاء الأسماء
(name precedence / name hiding)

عندما تعلن دالة عن اسم تعريفي محلى (local identifier) هو نفسه اسم تعريفي شامل (global identifier) ، فإن الاسم التعريفي المحلي تكون له الأولوية (precedence) في أي إشارة (reference) إلى هذا الاسم التعريفي داخل الدالة (within the function). ويطلق على هذا المبدأ / هذه القاعدة أولوية الأسماء أو إخفاء الأسماء.

وفي المثال التالي يستخدم البرنامج إعلانات عن كل من أسماء تعريفية شاملة وأخرى محلية.

مثال ٥-٨ : تتبع تنفيذ البرنامج التالي وأوجد مخرجاته

```
#include <iostream>

using namespace std;

void SomeFunc (  float   );

const int a = 17;               // A global constant
int b;                          // A global variable
int c;                          Another global variable

int main ( )
{
        b = 4;                  // Assignment to global b
        c = 6;                  // Assignment to global c
        SomeFunc (42.8);
        return 0;
}
```

```
void SomeFunc (  float c)      // Prevents access to global c
{
        float b;               // Prevents access to global b
        b = 2.3;               // Assignment to local b
        cout << "a =  " << a;  // Output global a (17)
        cout << " b =  " << b; // Output local b (2.3)
        cout << " c =  " << c; // Output local c (42.8)
}
```

الحل : المخرجات هي

a = 17 b = 2.3 c = 42.8

نلاحظ أن الدالة SomeFunc **أمكنها الوصول إلى** (accessing) **واستخدام الثابت الشامل** a (global constant) **، ولكنها تعلن أيضاً عن متغيرها المحلي** b (local variable) **ووسيطها** c (parameter) **. فالمتغير المحلي** b **له أولوية على المتغير الشامل** b **، وبالتالي فإن** b **المحلي يخفي** b **الشامل – عملياً – عن العبارات الموجودة في الدالة** SomeFunc**. وكذلك فإن الوسيط الشكلي** c **يمنع الوصول إلى** (blocks acces to) **المتغير الشامل** c **داخل الدالة. فوسطاء الدالة الشكليون يعملون – في هذا الشأن – كما تعمل المتغيرات المحلية، أي أن مجال الوسطاء الشكليين محلى** (local scope).

قواعد المجال (Scope Rules)

من النادر عموماً في برامج C++ **أن نعلن عن متغيرات شاملة ، حيث لاستخدام هذه المتغيرات نقاط سلبية لا نناقشها الآن. ولكن إذا استخدمنا هذه المتغيرات لأي سبب من الأسباب فيجب أن نعلم كيف يتم التعامل مع هذه الإعلانات. وقواعد الوصول إلى الأسماء التعريفية التي لم يعلَن عنها محلياً** (locally) **تسمى قواعد المجال** (scope rules) **. أي أن قواعد المجال هي القواعد التي تحدد المواضع – في البرنامج – التي يمكن أن نستخدم فيها / أن نصل منها إلى اسم تعريفي معين ، إذا علم الموضع الذي أعلن فيه عن هذا الاسم التعريفي.**

وبالإضافة إلى موضوع الوصول إلى الأسماء التعريفية المحلية والشاملة، فإن قواعد المجال في لغة C++ تتناول أيضاً ما يحدث عندما يكون لدينا قوالب متداخلة (nested blocks). [الأسماء التعريفية الشاملة تعد غير محلية (nonlocal) بالنسبة لجميع القوالب في البرنامج]. وإذا استَخدم قالب ما اسماً تعريفياً معيناً أُعلن عنه خارج هذا القالب بذاته، فإن هذا الاستخدام / هذا الوصول (access) يطلق عليه "وصول غير محلي" **(nonlocal access)** . والاسم التعريفي غير المحلي **(nonlocal identifier)** بالنسبة لقالب معين هو أي اسم تعريفي أعلن عنه خارج هذا القالب.

وفيما يلي قواعد المجال التفصيلية [باستثناء المجال الطبقي (class scope) وبعض خصائص اللغة التي لم نتعرض لها]:

(١) اسم أي دالة (a function name) مجاله شامل (global scope). ولا يجوز وضع تعريف دالة (function definition) داخل تعريف دالة أخرى ، أي أنه لا يسمح بوجود تعريفات متداخلة للدوال (nested function definitions) .

(٢) مجال (scope) وسيط أي دالة (a function parameter) هو نفسه مجال أي متغير محلي (local variable) معلن عنه في القالب الخارجي (الأقصى) (outermost block) لجسم الدالة (function body).

(٣) مجال أي متغير شامل (global variable) أو ثابت شامل (global constant) يمتد من موضع الإعلان عنه إلى نهاية الملف ، باستثناء ما هو مشار إليه في القاعدة (٥).

(٤) مجال أي متغير محلي (local variable) أو ثابت محلي (local constant) يمتد من موضع الإعلان عنه إلى نهاية القالب الذي أعلن عنه فيه. وهذا المجال يشمل أي قوالب متداخلة (nested blocks) ، باستثناء ما هو مشار إليه في القاعدة (٥).

(٥) مجال أي اسم تعريفي لا يشمل أي قالب متداخل (nested block) يحتوي على (contains) اسم تعريفي معلَن عنه محلياً (locally declared

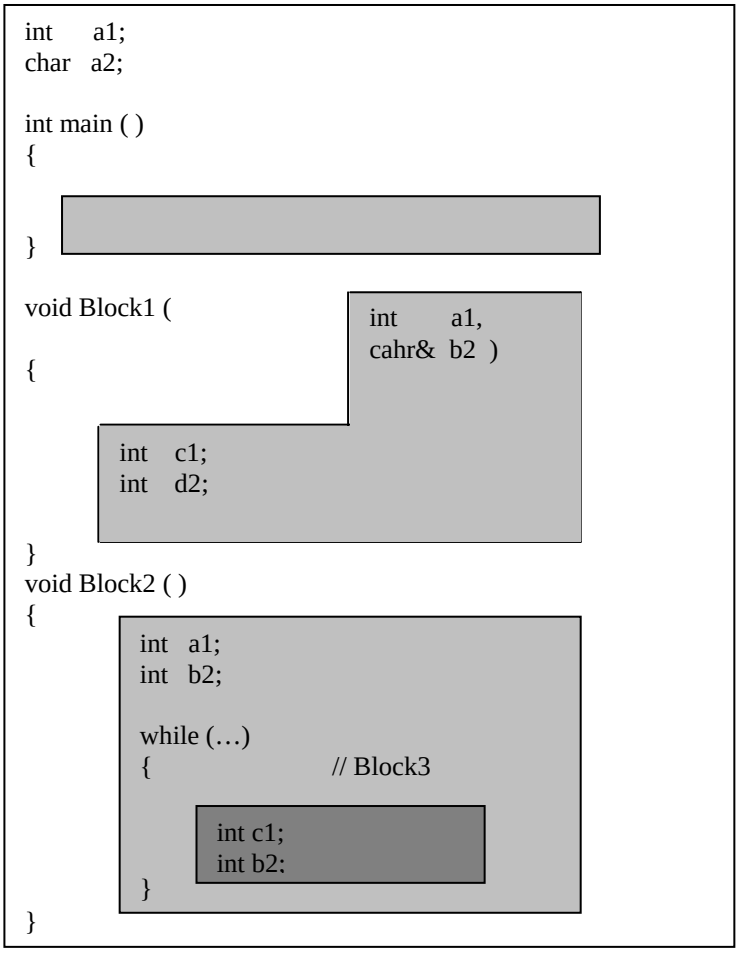
identifier) ويحمل الاسم نفسه (same name) [قاعدة أولوية الأسماء للأسماء التعريفية المحلية].

مثال ٥-٩: مخطط البرنامج التالي ScopeRules program يوضح قواعد المجال. ولتبسيط المثال اكتفينا بكتابة الإعلانات والعناوين دون تفاصيل عبارات البرنامج. ولاحظ أن جسم عروة while- والذي أطلقنا عليه الاسم Block3 ، والذي يقع داخل الدالة Block2 - يحتوي على إعلانات عن متغيراته المحلية.

```
// ScopeRules program

# include <iostream>

using namespace std;

void Block1 (  int,  char&);
void Block2 ( );

int a1;                         // One global variable
char a2;                        // Another global variable

int main ( )
{
        ⋮
}

// ********************************************************
void Block1 (  int  a1 ,        // Prevents access to global a1
        char&  b2  )            // Has same scope as c1 and d2
{
        int   c1;               // A variable local to Block1
        int   d2;               // Another variable local to Block1
        ⋮
}
// ********************************************************
void Block2
{
        int  a1;          // Prevents access to global a1
        int   b2;         // Local to Block2; no conflict with b2 in Block1
        while ( … )
        {
```

```
                    // Block3
        int   c1;   // Local to Block3; no conflict with c1 in Block1
        int   b2;   // Prevents nonlocal access to b2 in Block2; no
                    // conflict with b2 in Block1
        ⋮
    }
}
```

والشكل التالي (شكل ٥-١) يبين العناوين (headings) والإعلانات (declarations) في برنامج ScopeRules مع إطارات (boxes) تشير إلى مجالات الرؤية (scopes of visibility).



شكل ٥-١

مخطط المجالات لبرنامج قواعد المجال

Scope Diagram for ScopeRules Program

أي شئ داخل إطار ما (a box) يمكن أن يشير إلى أي شئ في إطارٍ محيطٍ به أكبر منه (a larger surrounding box) ، ولكن العكس غير صحيح. فمثلاً أي عبارة في القالب Block3 يمكن أن تصل إلى أي اسم تعريفي معلن عنه في القالب Block2 أو إلى أي متغير شامل، ولكنها لا يمكنها الوصول إلى الأسماء التعريفية المعلن عنها في القالب Block1 لأن عليها حينئذ أن تدخل الإطار Block1 من الخارج.

ولاحظ في هذا الشكل (شكل ٥-١) لمخطط المجالات أن وسطاء أي دالة موجودون داخل إطار الدالة، بينما اسم الدالة نفسه موجود خارج الإطار، لأنه لو كان الاسم داخل الإطار لما أمكن لأي دالة أن تستدعى أي دالة أخرى. وهذا يوضح أن أسماء الدوال شاملة (globally accessible).

ويمكن تصور الإطارات الموجودة في شكل ٥-١ أنها غرف ذات جدران مصنوعة من مرآة ثنائية الاتجاه بحيث أن الجانب العاكس (reflective side) يواجه الخارج (facing out) [أي لا يمكن رؤية ما بداخل الغرفة من خارجها] ، بينما الجانب الذي يسمح بالرؤية عبر المرآة (see-through side) يواجه الداخل (facing in) [أي يمكن رؤية ما هو خارج الغرفة من داخلها] . فإذا وقفنا مثلاً داخل غرفة Block3 فإننا نستطيع رؤية الخارج - عبر الغرف المحيطة - فنرى الإعلانات عن المتغيرات الشاملة (global variables) (وأي شئ بينها) ، ولكننا لا نستطيع رؤية ما بداخل أي غرف أخرى - غير محيطة - مثل Block1 لأن سطوحها الخارجية ذات المرآة العاكسة سوف تحجب الرؤية. وبسبب هذا التشابه / التماثل (analogy) فكثيراً ما نستخدم كلمة "مرئي" (visible) في وصف مجال الوصول (scope of access). مثلاً يقال إن المتغير a2 مرئي (visible) عبر البرنامج (throughout the program) ، أي أنه يمكن الوصول إليه من أي موضع في البرنامج.

وتجدر الإشارة إلى أن شكل ٥-١ يوضح فقط قواعد المجال: (١) ←(٤)، ولكن يجب أيضاً مراعاة القاعدة (٥) . فمثلاً المتغير a1 أُعلن عنه في ثلاثة مواضع مختلفة في برنامج ScopeRules. وبسبب قاعدة أولوية الأسماء فإن كلا من Block3، Block2 يصل إلى a1 المعلن عنه في Block2 وليس a1 الشامل. وبالمثل فإن مجال المتغير b2 المعلن عنه في Block2 لا يشمل "الثقب" (hole) الذي أحدثه Block3 نظراً لأن Block3 قد أعلن عن متغيره الخاص b2.

وأما عن طريقة البرنامج المترجم في تنفيذ قاعدة أولوية الأسماء فتتم كما يلي: عندما يشير تعبير ما إلى اسم تعريفي ، فإن البرنامج المترجم يتحقق أولاً من الإعلانات المحلية. فإن لم يكن الاسم التعريفي محلياً فإن المترجم يتجه للخارج عبر كل مستوى من مستويات التداخل (each level of nesting) حتى يجد اسماً تعريفياً يحمل الاسم نفسه ، وعندها يتوقف. فإن كان هناك اسم تعريفي له الاسم نفسه ومعلن عنه في مستوى أبعد من ذلك للخارج ، فإنه لا يصل إليه إطلاقاً. وإذا ما وصل المترجم إلى الإعلانات الشاملة (بما في ذلك الأسماء التعريفية التي تُدخلها الموجِّهات (# include directives) ولم يجد الاسم التعريفي بعد، فعندئذ تصدر رسالة خطأ مثل "UNDECLARED IDENTIFIER" . وغالباً ما تشير مثل هذه الرسالة إلى خطأ في بعض رموز الاسم التعريفي ، أو تبديل أحد حروفه من حرف كبير إلى صغير أو العكس ، أو أن الاسم التعريفي لم يُعلَن عنه قبل استخدامه، أو لم يعلَن عنه إطلاقاً، أو أنه بسبب تداخل (nesting) القوالب فإن مجال الاسم التعريفي لا يشمل الموضع الذي استُخدم فيه.

الإعلان عن المتغيرات وتعريفاتها
Variable Declarations and Definitions

سبق أن ذكرنا أن هناك فارقاً بين مصطلح: "الإعلان عن دالة" (function declaration) ومصطلح: "تعريف دالة" (function definition) ، وأن المصطلح :

"نموذج دالة" (function prototype) هو إعلان فقط (وليس تعريفاً)، حيث أنه لا يؤدي إلى حجز حيِّز من الذاكرة للدالة ، وأن تعريف الدالة، هو إعلان عنها يشمل جسم الدالة، وأن البرنامج المترجم يحجز حينئذ جزءاً من الذاكرة للتعليمات الواردة في جسم الدالة.

وباتباع المصطلحات نفسها فإن الإعلان عن متغير يصبح تعريفاً للمتغير إذا أدى إلى حجز حيز من الذاكرة للمتغير. وكل الإعلانات عن المتغيرات التي استخدمناها منذ البداية كانت تعريفات لهذه المتغيرات. فكيف يبدو الإعلان عن متغير إن لم يكن تعريفاً له؟

تحتوي لغة C++ على كلمة محجوزة: extern تجعلنا نشير إلى متغير شامل / عام (global variable) موجود في ملف آخر . فحين نستخدم إعلاناً معتاداً مثل:

int someInt;

فإن المترجم يحجز موضعاً في الذاكرة للمتغير someInt. بينما الإعلان:

extern int someInt;

- ويطلق عليه "إعلان خارجي" (external declaration) – يقرر أن someInt متغير شامل موجود في ملف آخر، وأنه لن يُحجز له هنا حيز من الذاكرة. وتحتوى ملفات المقدمة بالنظام (system header files) مثل iostream على إعلانات خارجية، بحيث يمكن لبرامج المستخدِم الوصول إلى متغيرات هامة معرَّفة في ملفات النظام. فمثلاً يشتمل iostream على إعلانات (declarations) مثل:

extern istream cin;
extern ostream cout;

وهذان الإعلانان يسمحان بالإشارة (reference) إلى كل من cin, cout كمتغير شامل في البرنامج، ولكن تعريفات المتغير (variable definitions) تكون موجودة في ملف آخر متوفر في نظام C++.

والعبارة extern int someInt;

تُعد إعلاناً وليس تعريفاً للمتغير someInt. وهي تُلحق اسم متغير بنوع بيانات بحيث يمكن للمترجم أن يقوم بالتحقق من النوع (type checking).

أما العبارة int someInt;

فهي تعد – في الوقت نفسه – إعلاناً وتعريفاً للمتغير someInt. وهي تعد تعريفاً له لأنها تحجز له جزءاً من الذاكرة. وفي لغة C++ يمكننا الإعلان عن متغير أو دالة عدة مرات، ولكن لا يكون هناك أكثر من تعريف واحد فقط. وعموماً سنستمر في استخدام العبارة العامة: "الإعلان عن المتغيرات" بدلاً من "تعريف المتغيرات" إلا حين يلزم التفرقة بينهما.

فضاءات الأسماء (Namespaces)

عموماً يُعد مصطلح "فضاء الأسماء" مرادفاً لكلمة "المجال" (scope). ولكن في لغة C++ يُقصد بهذا المصطلح الآلية (mechanism) التي يستطيع بها المبرمج إنشاء (creating) مجال مسمَّى (named scope). مثلاً ملف المقدمة القياسي cstdlib يحتوي على نماذج دوال (function prototypes) لعدة دوال مكتبية ، منها دالة القيمة المطلقة abs. ويتم احتواء الإعلانات ضمن تعريف "فضاء أسماء" (a namespace definition) كما يلي:

```
// In header file cstdlib:

namespace std
{
        ⋮
        int abs ( int ) ;
        ⋮
}
```

ويتكون تعريف فضاء الأسماء من الكلمة namespace، ثم اسم تعريفي يختاره المبرمج ثم "جسم فضاء الأسماء" (namespace body) بين قوسين { }. ويقال للأسماء التعريفية المعلن عنها في جسم فضاء الأسماء إنها تحتوى على مجال فضاء

أسماء (namespace scope). ولا يمكن الوصول إلى هذه الأسماء التعريفية خارج هذا الجسم إلا باستخدام إحدى الطرق الثلاث التالية:

(١) الطريقة الأولي: (وهي التي أشرنا إليها في الفصل الثاني) : استخدام اسم مؤهل (a qualified name) : وهو اسم فضاء الأسماء ، يليه مؤثر ثبات المجال (scope resolution operator) (: :) ، يليه الاسم التعريفي المطلوب. وفيما يلي مثال لذلك:

```
# include <cstdlib>

int main ( )
{
        int alpha;
        int beta;
        ⋮
        alpha = std: : abs (beta);        // A qualified name
        ⋮
}
```

والفكرة العامة في هذا المثال هي أننا نخبر المترجم (compiler) أننا نشير هنا إلى abs المعلن عنها في فضاء الأسماء std، وليس إلى abs أخرى (مثلاً دالة شاملة تسمى abs، والتي نكون نحن أنفسنا قد كتبناها).

(٢) الطريقة الثانية: استخدام عبارة تسمى "إعلان (declaration) using" ، كما يلي:

```
# include <cstdlib>

int main ( )
{
        int alpha;
        int beta;
        using std: : abs,                 // A using declaration
        ⋮
        alpha =  abs (beta);
        ⋮
}
```

وإعلان using هذا يسمح للاسم التعريفي abs أن يُستخدم خلال جسم الدالة main كمرادف (synonym) للاسم الأطول std: : abs

(٣) الطريقة الثالثة: استخدام موجِّه using(directive) [وهذا مختلف عن إعلان using] ، وقد تعودنا استخدامه من قبل.

```
# include <cstdlib>

int main ( )
{
        int alpha;
        int beta;
        using namespace std;          // A using directive
        ⋮
        alpha =  abs (beta);
        ⋮
}
```

وعند استخدام موجِّه using يمكننا الوصول إلى جميع الأسماء التعريفية في فضاء الأسماء المعين ولكن فقط في المجال (scope) الذي ظهر فيه موجِّه using. وفي المثال المذكور أعلاه نلاحظ أن موجِّه using في مجال محلي (local scope) – داخل قالب (within a block) – ولذلك فيمكن الوصول إلى الأسماء التعريفية في فضاء الأسماء std داخل الدالة main فقط. ومن ناحية أخرى إذا وضعنا موجِّه using خارج جميع الدوال – كما تعودنا فعل ذلك – كما يلي:

```
# include <cstdlib>

using namespace std;

int main ( )
{
        ⋮
}
```

فإن موجِّه using يكون في مجال شامل (global scope) ، وبالتالي فإن الأسماء التعريفية في فضاء الأسماء std يمكن الوصول إليها شمولياً (globally).

وقد يكون من المناسب أو المفيد وضع موجِّه using في مجال شامل. فمثلاً جميع الدوال التي نكتبها يمكنها حينئذ أن تشير إلى أسماء تعريفية مثل abs, cin, cout بدون الحاجة إلى إدخال موجِّه using محلياً في كل دالة. ولكن من ناحية أخرى عند كتابة برامج طويلة متعددة الملفات لا يُنصح بكتابة موجهات using شاملة ، حيث يقوم المبرمجون بالاستفادة من عدة مكتبات – وليس فقط مكتبة C++ القياسية – عند كتابة وتطوير برمجيات معقدة (complex software). وقد تستخدم مكتبتان أو أكثر – مصادفة – الاسم التعريفي نفسه لأغراض مختلفة تماماً. فإذا ما كنا نستخدم موجِّهات using شاملة فمن الممكن أن تحدث مشكلة "تصادم أسماء" (name clashes)، ويُقصد بها: تعريفات متعددة للاسم التعريفي نفسه، وذلك لأن جميع الأسماء التعريفية قد تم وضعها في مجال شامل. [مبرمجوا C++ يطلقون على هذه المشكلة "تلويث فضاء الأسماء الشامل" (polluting the global namespcae)] . وسنستمر في أسلوب استخدام موجِّهات using الشاملة لفضاء الأسماء std، وذلك لأن برامجنا عامة تعد صغيرة نسبياً، ولا ينتظر حدوث مشكلة تصادم الأسماء.

وبناء على مفهوم مجال فضاء الأسماء، نلخص فيما يلي وصف (description) طبقات مجال (scope categories) C++ :

(١) المجال الطبقي (Class scope): ويقصد به نوع البيانات المسمَّى "طبقة" (class) ، وتفاصيل هذا المجال خارج نطاق هذا الكتاب.
(٢) المجال المحلي (Local scope): مجال أي اسم تعريفي مُعلن داخل قالب يمتد من موضع الإعلان عنه إلى نهاية هذا القالب. وكذلك مجال وسيط أي دالة (الوسيط الشكلي) يمتد من موضع الإعلان إلى نهاية القالب الذي هو جسم الدالة.

(٣) مجال فضاء الأسماء (Namespcae scope): مجال أي اسم تعريفي مُعلن عنه في تعريف فضاء أسماء يمتد من موضع الإعلان عنه إلى نهاية جسم فضاء الأسماء، ومجاله يشمل مجال موجِّه using الذي يحدد فضاء الأسماء هذا.

(٤) المجال الشامل / مجال فضاء الأسماء الشامل (global scope) / (Global namespace) : مجال أي اسم تعريفي معلَن عنه خارج جميع فضاءات الأسماء والدوال والطبقات يمتد من موضع الإعلان عنه إلى نهاية الملف الذي يحتوي على البرنامج.

ويلاحظ أن ما سبق هو وصف عام لطبقات المجال وليس قواعد المجال (scope rules). وهذا الوصف لا يتناول "إخفاء الأسماء" (name hiding) [إعادة تعريف اسم تعريفي موجود في قالب متداخل (nested block)].

عُمرْ / مَدَى / فترة حياة متغير (Lifetime of a variable)

يقصد بُعمرْ متغير ما : الفترة الزمنية – أثناء تنفيذ البرنامج – التي يُخصَّص خلالها حيز في الذاكرة للاسم التعريفي لهذا المتغير. وقد ذكرنا سابقاً أن حيزاً في الذاكرة يخصص للمتغيرات المحلية في دالة ما لحظة دخول التحكم هذه الدالة. وعندئذ يُقال إن المتغيرات أصبحت حيِّة (alive) وتظل كذلك طوال تنفيذ الدالة، وعندما يتم الخروج منها تُمحى محتويات المواضع المخصصة في الذاكرة (storage is destroyed / deallocated) . وفي مقابل ذلك فإن عمر المتغير الشامل هو نفسه عمر البرنامج بأكمله ، حيث يخصص لهذا المتغير موضع في الذاكرة مرة واحدة فقط عند بدء تنفيذ البرنامج، ويُلغى هذا الحجر فقط عندما ينتهي تنفيذ البرنامج بأكمله.

لاحظ أن المدى (scope) مسألة تتعلق بزمن الترجمة (compile-time issue) ، بينما عمر المتغير (lifetime) مسألة تتعلق بزمن التشغيل (run-time issue).

المتغيرات الأوتوماتية والمتغيرات الاستاتية
(Automatic and Static Variables)

يقصد بالمتغير الأوتوماتي **(automatic variable)** في لغة C++ المتغير الذي يُحجز / يخصص له موضع في الذاكرة عند دخول القالب (block) الذي أُعلن عنه (declared) فيه، ثم يُخْلَى موضعه عند الخروج (exit) من هذا القالب.
والمتغير الاستاتي (static variable) هو المتغير الذي يظل موضعه محجوزاً له في الذاكرة طوال تشغيل البرنامج بأكمله. ويلاحظ ما يلي:

- جميع المتغيرات الشاملة (global) متغيرات استاتية.
- جميع المتغيرات المعلَن عنها في أي قالب يفترض (by default) أنها أوتوماتية.
- إذا استخدمنا الكلمة المحجوزة static عند الإعلان عن متغير محلي في دالة يصبح المتغير استاتياً ، ويظل عمرُه (lifetime) مستمراً (persisting) من استدعاء إلى استدعاء آخر للدالة:

```
void SomeFunc ( )
{
        float someFloat;        // Destroyed when function exits
        static int someInt      // Retains its value from call to call
                :
}
```

وعموماً يفضَّل الإعلان عن متغيرات محلية كمتغيرات استانية على استخدام متغيرات شاملة ، وذلك لأن هذه المتغيرات المحلية الاستاتية يظل موضعها في الذاكرة – مثل المتغيرات الشاملة – محجوزاً طوال الفترة الزمنية للبرنامج بأكمله، وبالإضافة إلى ذلك – خلافاً للمتغيرات الشاملة – فإن مداها المحلي (local scope) يمنع الدوال الأخرى في البرنامج من تعديل قيمها.

إعطاء قيم ابتدائية في الإعلانات
(Initialization in Declarations)

كثيراً ما نعلن عن متغير ما في برنامج ، ثم نسند له بعد ذلك – في عبارة مستقلة – قيمة ابتدائية ، مثل:

```
int sum;
  ⋮
sum = 0;
```

ولغة C++ تسمح لنا بجمع هاتين العبارتين في عبارة واحدة ، هكذا:

```
int sum = 0;
```

ويطلق على مثل هذه العبارة : إعطاء قيمة ابتدائية في الإعلان (initialization in a declaration) . ويطلق على التعبير الذي يحدد القيمة الإبتدائية في الإعلان : التعبير البادئ (initializer) . ففي المثال السابق : التعبير البادئ هو الثابت 0. وإذا كان نوع بيانات التعبير البادئ مختلفاً عن نوع بيانات المتغير يتم تحويل ضمني للنوع (implicit type coercion) .

وبالنسبة للمتغير الأوتوماتي : يتم إعطاؤه القيمة الابتدائية المحددة (صفر مثلاً) كل مرة ينتقل فيها التحكم إلى بداية هذا القالب:

```
void SomeFunc (  int someParam )
{
        int i = 0;                          // Initialized each time
        int n = 2 * someParam + 3;    // Initialized each time
          ⋮
}
```

وأما بالنسبة للمتغير الاستاتي [سواء كان متغيراً شاملاً أو متغيراً محلياً أُعلن عنه صراحة (explicitly) أنه استاتي] : فإن إعطاءه قيمة ابتدائية يتم مرة واحدة فقط هي أول مرة يصل فيها التحكم إلى الإعلان عنه. فمثلاً الدالة التالية تعلن عن متغيرين محليين استاتيين يُعطَى كل منهما قيمة ابتدائية مرة واحدة فقط هي عند أول استدعاء للدالة ، وأما عند أي استدعاء آخر للدالة فلا تسنَد لأي منهما هذه القيمة الابتدائية مرة أخرى، وإنما يظل محتفظاً بآخر قيمة وصل إليها [انظر مثلا السؤال ٥– ٢٠)] فبعد أن نسند لأي متغير قيمة ابتدائية يمكننا أن نسند له قيمة أخرى أثناء تنفيذ البرنامج.

```
void AnotherFunc (  int param )
{
        static char ch =  'A';              // Initialized once only
```

```
        static int m =  param + 1;       // Initialized once only
          ⋮
}
```

وبعض المبرمجين لا يفضلون إعطاء القيم الابتدائية للمتغيرات عند الإعلان عنها، وإنما قرب العبارات التنفيذية التي تعتمد على هذه المتغيرات، مثل:

```
int loopCount
   ⋮
loopCount = 1;
while (loopCount <= 20)
{
  ⋮
}
```

والبعض الآخر يفضل إعطاء القيم الابتدائية مع الإعلانات باعتبار أن أحد أسباب كثير من أخطاء البرمجة هو نسيان إعطاء قيم ابتدائية لبعض المتغيرات قبل استخدام محتوياتها.

تصميم الوصلة البينية (Interface Design)

علمنا من قبل أن فيض البيانات (data flow) خلال البينية الدالية (function interface) يمكن أن يأخذ إحدى صيغ ثلاث: بيانات داخلة فقط (incoming only)، أو خارجة فقط (outgoing only)، أو داخلة وخارجة (incoming / outgoing). وأي وسيط من الصيغة الأولى يجب أن يكون وسيطاً ذا قيمة (value parameter)، بينما أي وسيط من الصيغة الثانية أو الثالثة يجب أن يكون وسيط إسناد (reference parameter). وهناك استثناءات لهذه القاعدة، فلغة C++ تتطلب أن تُمَرَّر أهداف سيل المدخلات / المخرجات (I/O stream objects) بالإسناد (by reference) بسبب طبيعة تنفيذ السيول (streams) و الملفات (files). كذلك هناك استثناء آخر سنذكره بإذن الله في موضوع المنظومات (الفصل الثامن).

الآثار الجانبية (Side Effects)

ذكرنا سابقاً أن استخدام الوسيط ذي القيمة يمنع حدوث الآثار الجانبية (side effects) غير المرغوب فيها من تغيير قيمة الوسيط الفعلي في الدالة المستدعية . وهناك نوع آخر من الآثار الجانبية غير المرغوب فيها يمكن أن ينشأ عندما نجعل الاتصال (communication) بين دالتين يتم عن طريق الإشارة إلى المتغيرات الشاملة. فإذا حدث خطأ ما في إحدى الدالتين ، ونتيجة لوصول هذه الدالة لمتغير شامل تغيرت قيمة هذا المتغير بطريقة غير متوقعة، ثم استخدمت الدالة الأخرى هذا المتغير نشأ خطأ في هذه الدالة الأخرى . ولذلك لتفادي حدوث مثل هذه الأخطاء الناجمة عن تلك الآثار الجانبية ، يفضل دائماً أن يكون الاتصال بين الدوال عن طريق الوسطاء. ومن هنا وجب الاهتمام بتصميم بينية قائمة الوسطاء.

وتجدر الإشارة إلى أنه يمكن للدوال استخدام الثوابت المسماة الشاملة (named global constants) دون الخوف من حدوث آثار جانبية ، وذلك لأن قيم هذه الثوابت الشاملة لا يمكن أن تتغير طوال تشغيل البرنامج.

ثانياً : الدوال التي تعيد قيمة
Value - Returning Functions

تكلمنا حتى الآن عن الدوال الخاوية (void functions) التي نكتبها نحن. والنوع الثاني من البرامج الفرعية (subprograms) في لغة C++ هو الدوال التي تعيد قيمة (value returning functions). وقد مرت معنا سابقاً دوال من مكتبة C++ القياسية تعيد قيمة ، مثل الدوال: sqrt, abs, fabs, ...

ومن الفوارق الرئيسية بين الدالة الخاوية والدالة التي تعيد قيمة:

- استدعاء الدالة الخاوية يتم بعبارة مستقلة ، أما استدعاء الدالة التي تعيد قيمة فيتم ضمن تعبير.

- الدالة التي تعيد قيمة يفضل استخدامها عندما تكون هناك نتيجة واحدة فقط تعيدها الدالة ، ثم تُستخدم هذه النتيجة مباشرة في تعبير.

مثال ٥-١٠ :اكتب دالة لحساب مضروب (factorial) **عدد صحيح** n (**حيث** $n \geq 0$)، **والذي يرمز له بالرمز** !n ، **ويعرَّف كما يلي:**

$n! = n \times (n - 1) \times (n - 2) \times \ldots \times 3 \times 2 \times 1$ (if n > 0)
$0! = 1$ (if n = 0)

[مثلاً $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$]

الحل : هذه الدالة لها وسيط واحد فقط وهو العدد الصحيح int n ، **وتعيد قيمة واحدة فقط وهي مضروب** n (**وهو أيضاً عدد صحيح** int) .

```
int Factorial ( /* in */ int n )   // Number whose factorial is
                                   //  to be computed
// This function computes n !

// Precondition:
//       n >= 0   &&   n!  <=  INT_MAX
// Postcondition:
//       Function value = = n!

{
       int result;                 // Holds partial products

       result = 1;
       while (n > 0)
       {
              result = result * n;
              n - - ;
       }
       return result;
}
```

وإذا استدعينا هذه الدالة مثلاً لحساب المقدار

$$\text{combinations} = \frac{n!}{m! \,.\, (n - m)!}$$

فإننا نكتب عبارة الإسناد:

```
cominations = Factorial (n) / (Factorial (m) * Factorial ( n - m) );
```

مثال ٥-١١: اكتب دالة تستقبل عددين صحيحين x , n (**حيث** $n \geq 0$) ، **وتحسب** x^n (**أي** x **مرفوعة للأس / للقوة** n) .

ملاحظة: دالة الأس pow (power function) **من مكتبة** C++ **القياسية ترفع عدداً من النوع** float **لأس / لقوة من النوع** float. **وليس في مكتبة** C++ **القياسية دالة ترفع عدداً صحيحاً لأس صحيح.**

الحل: الدالة المطلوبة تأخذ وسيطين (x, n) **من النوع** int ، **وتعيد قيمة واحدة وهي أيضاً من النوع** int.

```
int Power ( /* in */ int x,              // Base number
            /* in */ int n  )            // Power to raise base to

// Trhis function computes x to the n power

// Precondition:
//      x is assigned  &&  n >= 0 && ( x to the n) <= INT_MAX
// Postcondition:
//      Function value = = x to the n power

{
      int result;                // Holds intermediate powers of x

      result = 1;
      while (n > 0)
      {
            result = result * x;
            n - - ;
      }
      return result;
}
```

مثال ٥-١٢: إذا سجَّل طالب في بداية فصل دراسي (semester) **ودفع مصاريف الدراسة** (tuition) ، **ثم انسحب** (withdrew) **قبل نهاية الفصل الدراسي ، فإنه يسترد قدراً من المصاريف التي دفعها، وهذا القدر المسترد** (refund) **يساوي : حاصل ضرب المصاريف الكلية المدفوعة في نسبة الجزء المتبقي من الفصل**

الدراسي إلى طول الفصل الدراسي كله ، وهذه النسبة تساوي : خارج قسمة عدد الأيام المتبقية من الفصل الدراسي على العدد الكلي لأيام الفصل الدراسي. ونفرض أن الفصل الدراسي يقع بأكمله في سنة شمسية (calendar year) واحدة [وهي التي تبدأ بأول يناير وتنتهي بآخر (٣١) ديسمبر] . ويمكن إيجاد أي من هذين العددين (عدد الأيام المتبقية ، وعدد أيام الفصل الدراسي) بسهولة كما يلي:

- نعطي أي يوم في السنة رقماً تسلسلياً يمثل ترتيبه في السنة الشمسية ، فالأول من يناير رقمه (التسلسلي) ١ ، والثاني من يناير رقمه ٢ ، وهكذا .. ويوم ٣١ ديسمبر رقمه ٣٦٥ أو ٣٦٦ حسب ما إذا كانت السنة بسيطة أو كبيسة (leap year) على الترتيب.
- يمكن إيجاد عدد أيام أي فترة بمعرفة رقم اليوم الأول في الفترة ورقم اليوم الأخير في الفترة . فمثلاً:

إذا بدأ الفصل الدراسي يوم الثالث من يناير عام ٢٠٠١، وانتهى يوم السابع عشر من مايو عام ٢٠٠١، فإن :

رقم يوم 1/3/01 هو 3

ورقم يوم 5/17/01 هو 137

فيكون عدد أيام الفصل الدراسي مساوياً : 137 - 3 + 1 = 135

المطلوب : اكتب دالة لإيجاد رقم اليوم (day number) (التسلسلي) لأي تاريخ معطى . أي أن الدالة تستقبل ثلاثة أعداد صحيحة (رقم الشهر month ، ورقم اليوم في هذا الشهر dayOfMonth ، والسنة year) تمثل التاريخ المعطى ، ثم توجد الدالة الرقم التسلسلي المقابل لهذا التاريخ في تلك السنة.

الحل: الدالة هنا تأخذ ثلاثة وسطاء عبارة عن ثلاثة أعداد صحيحة ، وتعيد قيمة واحدة وهي أيضاً عدد صحيح.

```
int Day ( /* in */  int month,          // Month number, 1 - 12
          /* in */  int dayOfMonth     // Day of month, 1 - 31
          /* in */  int year           ) // Year. For example, 2001
// This function computes the day number within a year, given
// the date. It accounts correctly for leap years. The
```

```
// calculation is based on the fact that months average 30 days
// in length. Thus, (month - 1) * 30 is roughly the number of
// days in the year at the start of any month. A correction
// factor is used to account for cases where the average is
// incorrect and for leap years. The day of the month is then
// added to produce the day number

// Precondition:
//       1 <= month <= 12
//   && dayOfMonth is in valid range for the month
//   && year is assigned
// Postcondition:
//       Function value = = day number in the range 1 - 365
//                              ( or 1 - 366 for a leap year)

{
       int correction = 0;      // Correction factor to account for leap
                                // year and months of different lengths
       // Test for leap year

       if (year % 4 = = 0 && (year % 100 != 0 | | year % 400 = = 0))
            if (month  >=  3)          // If date is after February 29
                correction = 1;        // then add one for leap year

       // Correct for different-length months
       if (month = = 3)
                correction = correction - 1;
       else if (month = = 2 | | month = = 6 | | month = = 7)
                correction = correction + 1;
       else if (month = = 8)
                correction = correction + 2;
       else if (month = = 9 | | month = = 10 )
                correction = correction + 3;
       else if (month = = 11 | | month = = 12 )
                correction = correction + 4;
       return  (month - 1) * 30 + correction + dayOfMonth;
}
```

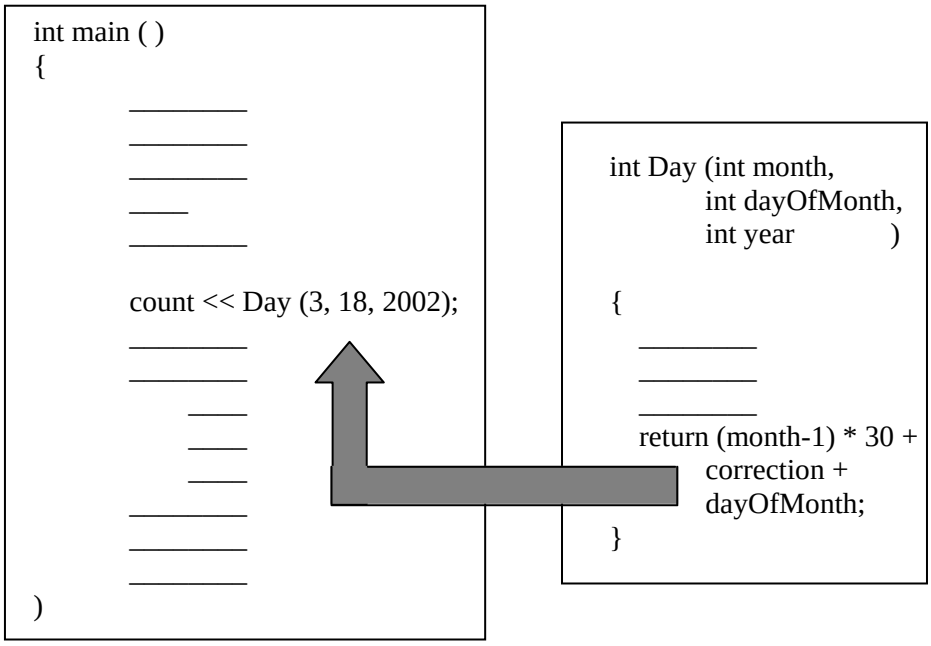
ملاحظات:

١) **العبارات التالية توضح كيفية استدعاء الدالة** Day **لحساب قيمة المبلغ المسترد** refund:

```
start = Day (startMonth, startDay, startYear) ;
last = Day (lastMonth, lastDay, lastYear) ;
withdraw = Day (withdrawMonth, withdrawDay, withdrawYear);
fraction = float (last - withdraw + 1) / float (last - strat + 1);
refund = tuition * fraction;
```

٢) لاحظ أن عنوان الدالة Day يبدأ بكلمة int التي تبين نوع بيانات (data type) القيمة الوحيدة التي تعيدها الدالة ، ويطلق على هذا النوع : نوع الدالة (function type) أو نوع قيمة الدالة (function value type).

٣) كذلك لاحظ عبارة return في نهاية الدالة Day، وتحتوي هذه العبارة على تعبير صحيح بين كلمة return والفاصلة المنقوطة . والعبارة تحسب قيمة هذا التعبير، وتعيد هذه القيمة كقيمة الدالة (function value) (انظر شكل ٥– ٢) .



شكل ٥– ٢

إعادة قيمة دالة للتعبير الذي استدعى الدالة

٤) هناك صيغتان لعبارة Return:

– الأولي: الصيغة return;

وهي تستخدم فقط مع الدوال الخاوية ، وتؤدي إلى خروج التحكم (control) فوراً (immediately) من الدالة (exit) ، والعودة إلى الدالة المستدعية

– الثانية : الصيغة return Expression;

وهي تستخدم فقط مع الدوال التي تعيد قيمة، وتؤدي إلى إعادة التحكم إلى الدالة المستدعية ، مع إعادة قيمة التعبير (Expression) كقيمة الدالة . وإذا كان نوع بيانات التعبير مخالفاً لنوع الدالة المعلن ، فإن قيمة التعبير تُحوَّل (coerced) إلى النوع السليم.

٥) نظراً لأن الدالة المطلوبة في هذا المثال (مثال ٥-١٢) تعيد قيمة واحدة فقط، فقد فضَّلنا كتابتها كدالة تعيد قيمة (value returning function) وليس كدالة خاوية (void function). وكان من الممكن كتابتها كدالة خاوية (computeDay مثلاً) تستقبل ثلاثة وسطاء (يمثلون التاريخ المعطى) ، وتعيد وسيطاً رابعاً (يمثل الرقم التسلسلي لليوم، أي النتيجة المطلوبة) . وفي هذه الحالة تكون عبارات استدعاء الدالة لحساب قيمة المبلغ المسترد كما يلي:

```
ComputeDay (startMonth, startDay, startYear, start) ;
ComputeDay (lastMonth, lastDay, lastYear, last) ;
ComputeDay(withdrawMonth,withdrawDay,withdrawYear,withdraw);
fraction = float (last - withdraw + 1) / float (last - strat + 1);
refund = tuition * fraction;
```

٦) إذا كانت أي دالة نكتبها خاوية ، وضعنا كلمة void كنوع للدالة قبل اسمها، وإذا كانت دالة تعيد قيمة، وضعنا نوع القيمة مثل int قبل اسمها. وإذا لم نضع أي نوع قبل اسم الدالة فإنه يُفترض أنها int.

٧) صيغة قائمة الوسطاء في الدالة التي تعيد قيمة هي نفسها في الدالة الخاوية: قائمة إعلانات عن الوسطاء (parameter declarations) ، مع

وجـود فاصـلة بـين كـل إعلانـين . وكـذلك نمـوذج الدالـة (function prototype) التي تعيد قيمة مماثل تماماً لنمـوذج الدالـة الخاويـة غـير أنـه يبدأ بنوع بيانات (قيمة الدالة) بدلاً من كلمة void.

الدوال المنطقية (Boolean Functions)

لا تقتصـر الـدوال الـتي تعيـد قيمـة علـى الـدوال الـتي تعيـد قيمـاً / نتـائج عددية. فيمكنا مثلاً كتابة دالة توجد قيمة شرط معين (evaluates a condition) وتعيد نتيجة منطقية. والدوال المنطقية تكون مفيدة بصورة خاصة حين يعتمد تفرع (branch) معين أو عروة معينة على شرط معقد (complex condition). فبدلاً من كتابة الشرط مباشرة داخل عبارة If أو عبارة While ، يمكننا استدعاء دالة منطقية تقوم هي باختبار شرط / تعبير التحكم (controlling expression).

مثال ٥-١٣:

(أ) اكتب دالة منطقية تأخذ ثلاثة وسطاء عبارة عن ثلاثة أعداد ذوات نقطة عائمة (floating point) تمثل ثلاث زوايا. وتعيد الدالة القيمة true إن كانت الزوايا الثلاث تكوِّن مثلثاً (أي أن مجموعها يساوي 180 درجة) ، وما عدا ذلك فإن الدالة تعيد القيمة false.

(ب) اكتب قطعة برنامج – تستخدم / تستدعي الدالة المنطقية في (أ) – بحيث تقرأ القطعة قيم ثلاث زوايا ، ثم تطبع رسالة تفيد ما إذا كانت الزوايا الثلاث تكون مثلثاً صحيحاً (valid triangle) أم لا.

الحل:

(أ)

```
# include <cmath>      // For fabs ( )
  ⋮
```

```
bool  IsTriangle (  /* in */ float angle1,         // First angle
                    /* in */ float angle2,         // Second angle
                    /* in */ float angle3  )       // Third angle
// This function checks to see if its three incoming values
// add up to 180 degrees, forming a valid triangle

// Precondition:
//        angle1, angle2, and angle3 are assigned

// Postcondition:
//        Function value = = true, if (angle1 + angle2 + angle3) is
                                    within 0.00000001 of 180.0 degrees
                              = = false, otherwise
{
        return (fabs(angle1 + angle2 + angle3 - 180.0) < 0.00000001);
}
```

(ب)

```
cin >> angleA >> angleB >> angleC;
if (IsTringle (angleA, angleB, angleC))        // Function call
        cout << "The three angles from  valid triangle.";
else
        cout << "Those angles do not form a triangle.";
```

دوال قياسية لاختبار محتويات المتغيرات الرمزية

Standard Functions for testing the contents of character variables

تحتوي مكتبة C++ القياسية على عدد من الدوال المساعدة التي تختبر محتويات المتغيرات من النوع char. ولاستخدم أي من هذه الدوال يجب أن يشتمل البرنامج على ملف المقدمة # include <cctype> . وفيما يلي بعض هذه الدوال. ويحتوي ملحق (د) على البعض الآخر.

ملف المقدمة Header File	الدالة Function	نوع الدالة Function Type	قيمة الدالة Function Value
<cctype>	isalpha (ch)	int	قيمة غير صفرية إذا كان ch حرفاً ('A'-'Z','a'-'z') وصفر ما عدا ذلك.
<cctype>	isalnum (ch)	int	قيمة غير صفرية إذا كان ch حرفاً أو رقماً ('A'-'Z','a'-'z','0'-'9') وصفر ما عدا ذلك.
<cctype>	isdigit (ch)	int	قيمة غير صفرية إذا كان ch رقماً ('0'-'9') وصفر ما عدا ذلك.
<cctype>	islower (ch)	int	قيمة غير صفرية إذا كان ch حرفاً صغيراً ('a'-'z') وصفر ما عدا ذلك.
<cctype>	isspace (ch)	int	قيمة غير صفرية إذا كان ch رمزاً فراغياً أبيض [فراغاً، أو رمز السطر الجديد، أو رمز الجدولة tab أو رمز عودة العربة carriage return، أو رمز تغذية الصيغ form feed] ، وصفر ما عدا ذلك.
<cctype>	isupper (ch)	int	قيمة غير صفرية ، إذا كان ch حرفاً كبيراً ('A'-'Z') وصفر ما عدا ذلك.

ومع أن هذه الدوال [عادة يشار إليها بالاصطلاح : functions "is…"] تعيد قيماً صحيحة int، إلا أنها تشبه في مسلكها الدوال المنطقية . وهي تعيد قيمة صحيحة int غير صفرية (تحوَّل إلى true في شرط If أو While) أو صفر (يحوَّل إلى false في شرط If أو While) . واستخدام هذه الدوال يختصر أحياناً بعض العبارات ، ويجعل البرنامج أسهل في القراءة . فمثلاً الشرط

if (isalnum(inputChar))

أسهل في القراءة ، وأقل عرضة للخطأ من الشرط الطويل:

```
if (inputChar >= 'A'  &&  inputChar <= 'Z'  ||
   inputChar >= 'a'  &&  inputChar <= 'z'  ||
   inputChar >= '0'  &&  inputChar <= '9'    )
```

الآثار الجانبية (للدوال التي تعيد قيمة) (Side Effects)

عادة تُستخدم الدالة التي تعيد قيمة لإعادة قيمة واحدة فقط، فإذا استخدمناها لإعادة أكثر من قيمة واحدة [بتعديل الوسطاء الفعليين في الدالة المستدعية (caller's arguments)] فهذا يُعَدُّ أثراً جانبياً يجب تحاشية. وفي حالة ضرورة إعادة أكثر من قيمة فعلينا حينئذ استخدام دالة خاوية.

وكقاعدة عامة لا تستخدم إطلاقاً وسطاء إسناد (reference parameters) في قائمة وسطاء الدالة التي تعيد قيمة، بل استخدم دائماً وسطاء ذوي قيمة (value parameters) . والمثال التالي يوضح أهمية هذه القاعدة.

مثال ٥-١٤:

نفرض أن لدينا الدالة التالية:

```
int SideEffect (  int&  n)
{
        int  result = n * n;

        n ++;                    // Side effect
        return  result;
}
```

ما هي القيمة التي ستخزن في المتغير y يعد استدعاء الدالة السابقة بالعبارة:

```
y = x + SideEffect (x);
```

بفرض أن قيمة x الأصلية هي 2 ؟

الحل: الدالة السابقة تعيد مربع (square) القيمة الداخلة (incoming value) ، ولكنها أيضاً تزيد قيمة الوسيط الفعلي قبل العودة. وتعتمد إجابة السؤال (أي قيمة y) على الترتيب الذي يقوم المترجم (compiler) بتنفيذه لتقييم التعبير. فإن كان استدعاء الدالة يتم أولاً فالإجابة هي 7 (4 + 3) ، وأما إن كان الوصول إلى x يتم

أولاً استعداداً لأن نجمع عليها قيمة الدالة فالإجابة هي 6 (4 + 2) . فعدم التأكد من النتيجة النهائية يوجب علينا تجنب استخدام وسطاء الإسناد مع الدوال التي تعيد قيمة.

وهناك استثناء (exception) لهذه القاعدة عندما نُمرِّر (pass) هدف سيل مدخلات / مخرجات (I/O stream object) لدالة تعيد قيمة ، حيث أن لغة C++ لا تسمح بتمرير أي هدف سيل (stream object) إلا إلى وسيط إسناد.

وهناك ميزة أخرى لاستخدام وسطاء ذوي قيمة فقط في تعريف الدالة التي تعيد قيمة، وهي أنه يمكننا حينئذ استخدام ثوابت (constants) وتعابير (expressions) كوسطاء فعليين (arguments). فمثلاً يمكننا استدعاء الدالة IsTriangle (مثال ٥-١٣) باستخدام قيم حرفية (literals) وتعابير أخرى كالتالي:

```
if  (IsTriangle (30.0,  60.0,  30.0 + 60.0 ))
   cout << "A 30 - 60 - 90 angle combination forms a triangle.";
else
   cout << "Something is wong.";
```

تجاهل قيمة دالة (Ignoring a Function Value)

إذا كتبنا قيمة دالة (function value) [القيمة التي تعيدها دالة تعيد قيمة (value returned by a value-returning function)] دون أن نستخدمها في أي تعبير أو عبارة ، فإن لغة C++ تتجاهل (ignores) هذه القيمة . فمثلاً إذا كتبنا العبارة التالية

```
sqrt (x);
```

فإن المترجم (compiler) لا يصدر أي رسالة خطأ. وعند تنفيذ هذه العبارة فإن القيمة التي تعيدها الدالة sqrt يتم تجاهلها ومحوها. وكذلك إذا كتبنا نحن دالة تعيد قيمة، ثم استدعيناها – خطأ – كما نستدعي الدالة الخاوية ، أي بعبارة مستقلة : مجرد ذكر اسم الدالة مع وسطائها الفعليين، فإن المترجم لا يعطي أي رسالة خطأ، ويتم تجاهل القيمة التي تعيدها هذه الدالة.

مثال ٥-١٥: اكتب برنامجاً لقراءة بعض التواريخ (dates) **بالصيغة الأمريكية** (American Format) mm / dd / yyyy (month / day/ year) **من سيل ملف** dataIn (file stream) **، وتحويلها** (converting them) **إلى كل من الصيغة البريطانية** (British Format) dd / mm / yyyy (day / month / year) **، وصيغة منظمة المعايير العالمية** ISO (International Standards Organization Format) yyyy - mm – dd (year - month - day) **، ثم طباعة النتائج في شكل جدول يكتب في سيل ملف** dataOut (file stream)**، بحيث يتكون الجدول من ثلاثة أعمدة تحتوي على التواريخ في الصيغ الثلاث مرتبة تحت بعضها البعض ، كما يلي:**

American Format	British Format	ISO Format
mm / dd / yyyy	dd / mm / yyyy	yyyy - mm - dd
mm / dd / yyyy	dd / mm / yyyy	yyyy - mm - dd

ويلاحظ أن التواريخ المعطاة بالصيغة الأمريكية مكتوبة بحيث أن كل تاريخ على سطر مستقل، وقد تكون هناك فراغات في ثنايا السطر (embedded blanks) **. وفيما يلي مثال لمدخلات ومخرجات البرنامج:**

المدخلات (ملف الإدخال) :

```
10/11/1935
1 1      /    2  3        / 1 9 2 6
                5/2/2004
05    / 28                    / 1965
7/    3/     19    56
```

المخرجات (ملف الإخراج)

American Format	British Format	ISO Format
10/11/1935	11/10/1935	1935-10-11
11/23/1926	23/11/1926	1926-11-23
05/02/2004	02/05/2004	2004-05-02
05/28/1965	28/05/1965	1965-05-28
07/03/1956	03/07/1956	1956-07-03

ملاحظة: عدد سطور الإدخال غير معلوم ، ويتوقف تشغيل البيانات حين يصل البرنامج إلى نهاية الملف EOF.

الحل :

```
//*************************************************************
 //ConvertDates program
 //This program reads dates in American form from an input file and
 //writes them to an output file in American, British, and ISO form.
 //No data validation is done on the input file
//*************************************************************
#include <iostream>
#include <iomanip>    // For setw()
#include <fstream>    // For file I/O
#include <string>     // For string type

using namespace std;

void Get2Digits( ifstream&, string&  );
void GetYear( ifstream&, string&  );
void OpenForInput( ifstream&  );
void OpenForOutput( ofstream&  );
void Write( ofstream&, string, string, string  );

int main()
{
   string   month;        // Both digits of month
   string   day;          // Both digits of day
   string   year;         // Four digits of year
   ifstream dataIn;       // Input file of dates
   ofstream dataOut;      // Output file of dates

   OpenForInput(dataIn );
   OpenForOutput(dataOut);
   if ( !dataIn || !dataOut )              // Make sure files
      return 1;                            //   were opened

  dataOut << setw(20) << "American Format"     // Write headings
```

```
            << setw(20) << "British Format"
            << setw(20) << "ISO Format" << endl << endl;

   Get2Digits(dataIn, month);              // Priming read
   while (dataIn)                      // While not EOF…
   {
      Get2Digits(dataIn, day);
      GetYear(dataIn, year);
      Write(dataOut, month, day, year);
      Get2Digits(dataIn, month);
   }
   return 0;
}

//****************************************************************
void OpenForInput( /* inout */ ifstream& someFile )    // File to be
                                                       // opened
 //Prompts the user for the name of an input file
 //and attempts to open the file

{
   string fileName;    // User-specified file name

   cout << "Input file name: ";
   cin >> fileName;

   someFile.open(fileName.c_str( ));
   if ( !someFile )
      cout << "** Can't open " << fileName << " **" << endl;
{

//****************************************************************
void OpenForOutput( /* inout */ ofstream& someFile )   // File to be
                                                       // opened
 //Prompts the user for the name of an output file
 //and attempts to open the file
{
```

```
   string fileName;    // User-specified file name

   cout << "Output file name: ";

   cin >> fileName;

   someFile.open(fileName.c_str( ));
   if ( !someFile )
      cout << "** Can't open " << fileName << " **" << endl;
}

//*************************************************************
void Get2Digits( /* inout */ ifstream& dataIn,      // Input file
                 /* out */   string&   twoChars )   // Two digits
 //Reads characters up to a slash from dataIn and returns two
 //digit characters in the string twoChars.  If only one digit
 //is found before the slash, a leading '0' is inserted.
 // (If input fails due to end-of-file, twoChars is undefined.)

{
   char firstChar;    // First character of a two-digit value
   char secondChar;   // Second character of value
   char dummy;        // To consume the slash, if necessary

   dataIn >> firstChar;
   if ( !dataIn )              // Check for EOF
      return;                 // If so, exit the function

   dataIn >> secondChar;
   if (secondChar = = '/' )
   {
      secondChar = firstChar;
      firstChar = '0';
   }
   else
      dataIn >> dummy;         // Consume the slash
   twoChars = firstChar;
   twoChars = twoChars + secondChar;
}
```

```
//****************************************************************
void GetYear( /* inout */ ifstream& dataIn,  // Input file
              /* out */   string&   year   )    // Four digits
                                                 //of year

 //Reads characters from dataIn and returns four digit characters
 //in the year string

{
   char digitChar;    // One digit of the year
   int  loopCount;    // Loop control variable

   year  = "  " ;
   loopCount = 1;
   while (loopCount <= 4)
   {
      dataIn >> digitChar;
      year = year + digitChar;
      loopCount ++;
   }
}

//****************************************************************
void Write( /* inout */ ofstream& dataOut,    // Output file
            /* in */    string    month,      // Month string
            /* in */    string    day,        // Day string
            /* in */    string    year    )   // Year string


 //Writes the date represented by month, day, and year to file
 //dataOut in American, British, and ISO form

{
   dataOut << setw(9) << month << '/' << day << '/' << year;
   dataOut << setw(13) << day << '/' << month << '/' << year;
   dataOut << setw(16) << year << '-' << month
           << '-' <<   day << endl;
}
```

تمرينات رقم ٥

٥-١) ما هي مخرجات البرنامج التالي؟

```
# include <iostream>

using namespace std;

void Print (  int,  int  );

int main ( )
{
        int n;

        n = 3;
        Print (5, n);
        Print (n, n);
        Print (n * n, 12);
        return 0;
}
void Print ( int a,
             int b  )

{
        int c;

        c = 2 * a + b;
        cout << a << '  ' << b << '  ' << c << endl;
}
```

٥-٢) نفرض أن لدينا الإعلانات التالية:

```
const  int  ANGLE = 90;

char letter;
int number;
```

بالنسبة لكل من الوسطاء الفعليين (arguments) التاليين اذكر ما إذا كان الوسيط مقبولاً / (صحيحاً) (valid) عندما يكون الوسيط الشكلي المقابل وسيطاً ذا قيمة (value parameter) أم وسيط إسناد (reference parameter) أم كلاهما. بأسلوب آخر اذكر ما إذا كان الوسيط الفعلي مقبولاً / (صحيحاً) (valid) عند الاستدعاء / التمرير بالقيمة (pass by value) فقط أم عند الاستدعاء / التمرير بالإسناد (pass by reference) فقط ، أم عند أي منهما.

(a) letter
(b) ANGLE
(c) number
(d) number + 3
(e) 23
(f) ANGLE * number
(g) abs (number)

٥-٣) أ) نفرض أن widgets متغير مخزون في الذاكرة في الموقع 13571 (memory location) . ونفرض أن العبارتين التاليتين قد تم تنفيذهما:

```
widgets = 23;
Drop (widgets);
```

ما هي المعلومات (information) التي تم تمريرها (passed) للوسيط الشكلي (parameter) في الدالة Drop ، بفرض أن هذا الوسيط وسيط إسناد (reference parameter)؟

ب) نفرض أن الوسيط الشكلي في الدالة Drop اسمه clunkers. ونفرض أن جسم الدالة قد قام بتنفيذ عبارة الإسناد

```
clunkers = 77;
```

ما هي القيمة الموجودة الآن في
(i) widgets (ii) clunkers
بعد تنفيذ العبارة؟

٥-٤) باستخدام قيم البيانات

3 2 4

أوجد مخرجات البرنامج التالي:

```
# include <iostream>

using namespace std;

void Test (  int&,  int&,  int& );

int main ( )
{
        int a;
        int b;
        int c;

        Test (a, b, c);
        b = b + 10;
        cout << "The answers are " << b << '  ' << c <<'  ' << a;
        return 0;
}

void Test (  int&  z;
             int& x;
             int& a  )
{
        cin  >>  z  >>  x  >  .  a;
        a = z * x + a;
}
```

٥-٥) البرنامج التالي يحتوي على دالة تُدعى Change. املأ قيم جميع المتغيرات قبل وبعد استدعاء الدالة. ثم املأ قيم جميع المتغيرات بعد العودة (return) إلى الدالة main. (إذا كانت أي قيمة غير معرفة اكتب U بدلاً من عدد).

```
        # include <iostream>
```

```
using namespace std;

void Change (  int,  int& );

int main ( )
{
        int a;
        int b;
        a = 10;
        b = 7;
        Change (a, b);
        Cout << a << '  ' << b << endl;
        return 0;
}

void Change (  int  x,
                int&  y  )
{
        int b;

        b = x;
        y = y + b;
        x = y;
}
```

(i) **المتغيرات في** main **قبل استدعاء** Change **مباشرة:**

a: b:

(ii) **المتغيرات في** Change **عند لحظة دخول التحكم الدالة:**

(at the moment control enters the function)

x: y: b:

(iii) **المتغيرات في** main **بعد العودة** (return) **من** Change:

a: b:

٥-٦) بَيِّن مخرجات البرنامج التالي:

```
# include <iostream>
```

```
using namespace std;

void Test (  int&,  int  );

int main ( )
{
        int d;
        int e;

        d = 12;
        e = 14;
        Test (d,  e);
        cout << "In the main function after the first call,  "
             <<  "the variables equal " << d << '  ' << e << endl;
        d = 15;
        e = 18;
        Test (e,  d);
        cout << "In the main function after the second call,  "
             <<  "the variables equal " << d << '  ' << e << endl;

        return 0;
}

void Test  (  int&   s,
              int      t  )
{
        s = 3;
        s = s + 2;
        t =  4 * s;
        cout << "In function Test, the variable equal  "
             <<  s  << '  ' <<  t  << endl;
}
```

٥-٧-أ) رقِّم (number) **العبارات المشار إليها** (marked statements) **في البرنامج التالي بحيث يبيِّن هذا الترقيم الترتيب الذي تنفذ به هذه العبارات. أي أن هذا الترقيم يعطى الترتيب المنطقي لتنفيذ العبارات** (logical order of execution).

```
# include <iostream>
```

```
using namespace std;

void DoThis (  int&,  int&  );

int main ( )
{
        int number1;
        int number2;

_____   cout << "Exercise  "
_____   DoThis (number1, number2);
_____   cout << number1 << '  ' << number2 << endl;
        return 0;
}

void DoThis (  int&  value1;
                int&  value2  )
{
        int value3;

_____   cin >> vlue3 >> value1,
_____   value2 = value1 + 10;
}
```

ب) نفرض أن البرنامج في أ) قد تم تشغيله / تنفيذه بقيمتي البيانات 15 **و** 10. **ماذا ستكون قيم المتغيرات التالية قبل تنفيذ عبارة** return **في الدالة** main؟

number1: number2: value3:

٥-٨-أ) اكتب عنوان دالة خاوية (void function heading) **تُدعى** PrintMax **تستقبل عددين صحيحين وتطبع أكبرهما. وضح فيض البيانات** (data flow) **لكل وسيط باستخدام أحد التعليقات:**

/ * in * / , / * out * /, / * inout * /.

ب) اكتب عنوان دالة خاوية RocketSimulation – كنموذج لمحاكاة عمل صاروخ (Rocket Simulation Module) – تستقبل قائمة البيانات التالية:

thrust	(floating point)	Incoming	داخل
weight	(floating point)	incoming / outgoing	داخل / خارج
timeStep	(integer)	incoming	داخل
totalTime	(integer)	incoming	داخل
velocity	(floating point)	outgoing	خارج
outOfFuel	(boolean)	outgoing	خارج

٥–٩)اكتب دالة خاوية تقرأ عدداً محدداً من القيم العائمة float وتعيد متوسطها (average) . ويمكن أن تُستدعى الدالة بعبارة مثل

GetMeanOf (5, mean);

حيث الوسيط الفعلي الأول يحدد عدد القيم التي تُقرأ ، والوسيط الفعلي الثاني يحتوي على النتيجة . وثّق (document) فيض البيانات (data flow) لكل وسيط بأحد التعليقات الثلاثة:

/ * in * / , / * out * /, / * inout * /.

٥–١٠–أ) نفرض أنك قد أعطيت عنوان الدالة التالي:

```
void Halve (   / * inout */     float&  firstNumber,
               / * inout */     float&  secondNumber  )
```

اكتب جسم الدالة بحيث أنه عندما تعود الدالة تكون القيمتان الأصليتان في firstNumber, secondNumber قد نُصِّفتا (halved).

ب) أضف تعليقات للدالة السابقة Halve توضح كلا من المتطلبات السابقة (المتطلبات الأساسية) والمتطلبات اللاحقة (function precondition and postcondition).

٥-١١-أ) في برنامج Walk program في (مثال ٢-٢٨) نلاحظ أن مجموعة معينة من العبارات تتكرر أربع مرات في البرنامج مع تغييرات بسيطة. اكتب دالة خاوية واحدة تحل محل النمط المتكرر من العبارات (repeated pattern of statements) مع توثيق فيض بيانات الوسطاء (داخل أو خارج أو داخل/خارج)، ومع إعطاء تعليقات توضح المتطلبات السابقة والمتطلبات اللاحقة للدالة.

ب) اكتب العبارات الأربع لاستدعاء الدالة مع بيان الوسطاء الفعليين (arguments).

٥-١٢-أ) اكتب دالة خاوية تظل تقرأ قيم بيانات (heartRate) من النوع int حتى تقرأ معدل نبض قلب معتاداً (60 – 80) (a normal heart rate) أو تصل إلى نهاية الملف EOF. والدالة لها وسيط شكلي واحد اسمه normal يحتوي على القيمة true إذا قرئت قيمة معتادة لنبض القلب، والقيمة false إذا تم الوصول إلى نهاية الملف EOF.

ب) اكتب عبارة تستدعى الدالة في (أ). يمكنك استخدام اسم المتغير نفسه لكل من الوسيط الشكلي والوسيط الفعلي.

٥-١٣) نفرض أن لدينا تعريف الدالة التالي:

```
void Rotate(    /* inout */ int& firstValue,
                /* inout */ int& secondValue,
                /* inout */ int& thirdValue    )
{
```

```
        int temp;

        temp = firstValue;
        firstValue = secondValue;
        secondValue = thirdValue;
        thirdValue = temp;
}
```

أ) أضف تعليقات للدالة توضح وظيفة الدالة والغرض من كل وسيط وكل متغير محلي.

ب) اكتب برنامجاً لقراءة قيم ثلاثة متغيرات، وطباعة هذه القيم (طباعة صدى echo printing) ، ثم استدعاء الدالة Rotate مستخدماً المتغيرات الثلاثة كوسطاء فعليين، وأخيراً طباعة قيم الوسطاء الفعليين بعد عودة الدالة.

ج) عدِّل الدالة Rotate بحيث تؤدي نوع العمليات نفسها على أربع قيم. كذلك عدِّل البرنامج الذي كتبته في ب) ليعمل مع الصيغة المعدَّلة للدالة Rotate.

٥-١٤) اكتب دالة خاوية CountUpper توجد عدد الحروف الكبيرة (number of uppercase letters) في سطر واحد من المدخلات (one line of input). وتعيد الدالة هذا العدد إلى الدالة المستدعية في وسيط يُدعى upCount.

٥-١٥) اكتب دالة خاوية AddTime لها ثلاثة وسطاء hours, minutes, elapsedTime ، حيث elapsedTime هي العدد الصحيح (integer number) من الدقائق التي تضاف (to be added) إلى وقت الابتداء (starting time) الذي يُعطَى مقدراً بالساعات والدقائق ويمرر (passed) عن طريق الوسيط hours والوسيط minutes. وأما الوقت الجديد الناتج – بعد الإضافة – فيعاد أيضاً عن طريق الوسيط hours والوسيط minutes. وفيما يلي مثال يعطي بعض القيم بفرض أن أسماء الوسطاء الفعليين هي أيضاً hours, minutes, elapsedTime .

قبل استدعاء الدالة AddTime	بعد استدعاء الدالة AddTime
hours = 12 minutes = 44 elapsedTime = 198	hours = 16 minutes = 2 elapsedTime = 198

٥-١٦) اكتب دالة خاوية GetNonBlank تعيد أول رمز غير الفراغ (first nonblank character) **تصادفه** (it encounters) **في سيل المدخلات القياسي** (standard input stream) **، بحيث تستخدم الدالة** cin.get **لقراءة أي رمز . [ملاحظة: هذا السؤال لكتابة الدالة** GetNonBlank **هو لمجرد التدريب فقط، فهذه الدالة غير ضرورية حيث يمكننا استخدام المؤثر >> الذي يتخطى الفراغات الرائدة وبالتالي يؤدي وظيفة هذه الدالة ويصل إلى نتيجتها نفسها].**

٥-١٧-أ) اكتب دالة خاوية SkipToBlank **تتخطى** (skips) **جميع الرموز في سيل المدخلات القياسي** (standard input stream) **حتى تصادف فراغاً، بحيث تستخدم الدالة** cin.get **لقراءة أي رمز . [ملاحظة: هذا السؤال لكتابة الدالة** SkipToBlank **هو لمجرد التدريب فقط، فهناك الدالة المكتبية** cin.ignore **التي تؤدي فعلاً وظيفة الدالة** SkipToBlank **نفسها].**

ب) عدِّل الدالة في الجزء أ) من السؤال بحيث تعيد عدد الرموز التي تم تخطِّيها (skipped).

٥-١٨) بالنسبة لبرنامج Graph program **مثال ٥ -٧) قم بإجراء التعديلات / الإضافات التالية:**

أ) اكتب دالة منفصلة ترسم خطاً من النجوم (a bar of asterisks) في هدف سلسلة رموز (in a string object) إذا أعطيت قيمة المبيعات.
ب) أعد كتابة الدالة PrintData بحيث تستدعى الدالة التي كتبتها في (أ).

ج) عدِّل البرنامج Graph program بحيث يطبع رسالة خطأ عندما تُدخل قيمة سالبة لعدد الأيام في الشهر بالنسبة لأحد الأقسام.

د) أعد كتابة البرنامج Graph program بحيث يكتشف قيم المبيعات التي هي أكبر من 25,000 $ ، فيطبع عندئذ خطاً من النجوم حتى علامة القيمة 25,000 $ ثم يطبع علامة تعجب (!) في نهاية الخط.

هـ) اكتب برنامجاً يتم تشغيله / تنفيذه (running) قبل برنامج Graph program يقارن بين ملفي البيانات (two data files)، ويعطي رسالة خطأ إذا وجد عدم توافق / عدم تطابق (mismatch) ، بين الرقم التعريفي للقسم (department ID number) في الملف الأول والرقم التعريفي المقابل في الملف الثاني، أو إذا اشتمل الملفان على عددين مختلفين من الأقسام.

٥-١٩) ما هي مخرجات البرنامج التالي ؟ (هذا البرنامج مثال للتصميم البيني السيئ (poor interface design practices).

```
# include <iostream>

using namespace std ;

void DoGlobal ( );
void DoLocal ( );
void DoReference ( int& );
void DoValue ( int );

int x;

int main ( )
```

```
{
        x = 15;

        DoReference (x);
        cout << "x = " << x << " after the call to DoReference."
             << endl;
        x = 16;
        DoValue (x);
        cout << "x = " << x << " after the call to DoValue."
             << endl;
        x = 17;
        DoLocal ();
        cout << "x = " << x << " after the call to DoLocal."
             << endl;
        x = 18;
        DoGlobal ();
        cout << "x = " << x << " after the call to DoGlobal."
             << endl;
        return  0;
}

void DoReference (  int&  a )
{
        a = 3;
}

void DoValue (  int  b  )
{
        b = 4;
}
void DoLocal ( )
{
        int x;

        x = 5;
}
void DoGlobal ( )
{
        x = 7;
}
```

٥-٢٠) ما هي مخرجات البرنامج التالي؟

```
# include <iostream>

using namespace std ;

void Test ( );

int main ( )
{
        Test ( );
        Test ( );
        Test ( );
        return  0;
}

void Test ( )
{
        int i = 0;
        static int j = 0;

        i ++;
        j ++;
        cout << i << ' '  << j << endl;
}
```

٥-٢١) الدالة التالية تحسب مجموع الأعداد الصحيحة من 1 إلى n، إلا أن لها تأثيراً جانبياً غير مقصود (unintended side effect)**، فما هو؟**

```
void SumInts (  int& n,
                int& sum  )
{
        sum = 0;
        while (n >= 1)
        {
                sum = sum + n;
                n = n-1;
        }
}
```

٢٢-٥) نفرض أن لدينا عنوان الدالة

```
bool HighTaxBracket (  int inc,
                       int ded )
```

فهل تعد العبارة التالية استدعاءً صحيحاً للدالة ، بفرض أن income, deductions متغيران من النوع int ؟

```
if (HighTAxBracket (income, deductions) )
        cout << "Upper Class";
```

٢٣-٥) العبارة

```
Power (k, l, m);
```

تعد استدعاء للدالة الخاوية المعرفة فيما يلي. أعد كتابة الدالة كدالة تعيد قيمة ، ثم اكتب استدعاء للدالة يسند قيمة الدالة للمتغير m.

```
void Power (  float     base,
              int       exponent,
              float&    answer      )
{
        int i;
        answer = 1.0;
        i = 1;
        while (i <= exponent)
        {
                answer = answer * base;
                i ++;
        }
}
```

٢٤-٥) نفرض أننا قد أُعطينا الدالة Test التالية، وأن المتغيرات a, b, c, result قد أعلن عنها أنها من النوع float في برمج معطَى مع الدالة Test. ونفرض أن قيم a,b,c عند استدعاء الدالة هي:

a = -5.0, b = 0.1 c = 16.2

ما هي قيمة result عند العودة من كل من الاستدعاءين التاليين؟

```
float Test (    floaat x,
                float y,
                float z )
{
```

```
        if (x > y   ||   y > z)
                return 0.5;
        else
                return -0.5;
}
```

(أ) `result = Test (5.2,  5.3,   5.6);`

(ب) `result = Test (fabs (a),  b,  c);`

٥-٢٥) ما هو الخطأ في كل من التعريفين التاليين لدالتين في لغة C++ ؟

(أ)
```
void Test1 (    int m,
                int n  )
{
        return 3 * m + n;
}
```

(ب)
```
float Test2 (    int     i,
                 float  x  )
{
        i = i  + 7;
        x = 4.8 + float (i);
}
```

٥-٢٦) ما هي مخرجات قطعة البرنامج التالية بفرض أن جميع المتغيرات من النوع int؟

```
alpha = 3;
beta = 20;
if (beta > 10)
{
   int alpha = 5;

   beta = beta + alpha;
   cout << alpha << ' ' << beta << endl;
}
cout << alpha << ' ' << beta << endl;
```

٥-٢٧) ما هي مخرجات البرنامج التالي؟

```
#include <iostream>
```

```
using namespace std;

void Try( int&, int );

int x;
int y;
int z;

int main()
{     x = 1;
      y = 2;
      z = 3;
      Try(y, x);
      cout << x << ' ' << y << ' ' << z << endl;
      return 0;
}

void Try( int& a,
          int  b )
{   int x;

    x = a + 2;
    a = a * 3;
    b = x + a;
}
```

٥-٢٨) نفرض أننا قد أُعطينا تعريف الدالة

```
void Test( /* in */ int alpha )
{
   static int n = 5;

   n = n + alpha;
   cout << n << ' ';
}
```

ما هي مخرجات قطعة البرنامج التالية، بفرض أن الدالة Test **لم تُستدع من قبل؟**

```
Test (20);
Test (30);
```

٥-٢٩) إذا أُعطينا تعريف الدالة

```
int Mystery( /* in */ float someVal )
{
   if (someVal > 2.0)
      return 3 * int(someVal);

  else
      return 0;
}
```

فما هي قيمة التعبير Mystery (4.2) ؟

٥-٣٠) إذا أُعطينا تعريف الدالة

```
int Trans( /* in */ int alpha,
           /* in */ int beta  )
{
   if (alpha > beta)
      return alpha + 10;
   else
      return 2 * beta;
}
```

فماذا تطبع العبارة التالية ؟

```
cout << Trans (5,   Trans (9,  4)) << endl;
```

٥-٣١) إذا أُعطينا تعريف الدالة

```
bool IsZip( /* in */ float someFloat )
{
   return (someFloat == 0.0);
}
```

فما هي قيمة التعبير IsZip (2.4) ؟

٥-٣٢) هذا السؤال يوضح أخطار الآثار الجانبية. نفرض أننا قد أُعطينا تعريف الدالة

```
int Power( /* in */ int& base,
           /* in */ int& exp  )
{   int product = 1;
```

```
    while (exp >= 1)
    {
       product = product * base;
       exp--;
    }
    return product;
}
```

ما هي مخرجات القطعة التالية ، بفرض أن جميع المتغيرات من النوع int؟

```
n = 2;
pow = 3;
result = Power (n,  pow);
cout << n << " to the power " << pow << is " << result;
```

٥-٣٣) اذكر صحة أو خطأ (T/F) كل من العبارات التالية:

أ) إذا احتوت دالة على متغير معلن عنه محلياً واسمه هو نفسه اسم متغير شامل، فلا يحدث أي التباس لأن الإشارات (references) **إلى المتغيرات في الدوال تفسَّر أولاً على أنها إشارات إلى متغيرات محلية.**

ب) المتغيرات المعلن عنها في بداية أي قالب يمكن الوصول إليها من جميع العبارات المتبقية في هذا القالب ، بما في ذلك تلك العبارات في القوالب المتداخلة (nested blocks) **[بفرض أن هذه القوالب المتداخلة لا تعلن عن متغيرات محلية تحمل الأسماء نفسها].**

ج) الدوال التي تعيد قيمة يمكن أن تستخدم فقط لإعادة قيم عددية (numeric values).

د) العبارة

```
return 3 * alpha + 8;
```

تعد عبارة صحيحة (valid) **في دالة تعيد قيمة ، وغير صحيحة في دالة خاوية.**

هـ) إذا أُعطينا تعريف الدالة

```
float SomeFunc( /* in */ int n )
   {
       ...
       return x;
   }
```

فإنه ينشأ خطأ وقت الترجمة (compile time error) إذا كان نوع بيانات x مختلفاً عن float.

٣٤-٥) البرنامج التالي مكتوب بأسلوب غير جيد ، حيث تُستخدم المتغيرات الشاملة في مواضع (بدلاً من) الوسطاء الفعليين. أعد كتابة البرنامج بأسلوب برمجة جيد بدون متغيرات شاملة.

```
# include <iostream>
using namespace std ;
void MashGlobals ( );
int  a, b, c;
int main ( )
{
cin >> a >> b >> c;
MashGlobals ( );
cout << "a=" << a << ' ' << "b=" << b << ' '
<< "c=" << c << endl;
return 0;
}
void MashGlobals ( )
{
int temp;
temp = a + b;
a = b + c;
b = temp;
}
```

٥-٣٥) اكتب عنوان (heading) دالة تعيد قيمة Epsilon تستقبل وسيطين من النوع float اسماهما high, low وتعيد نتيجة من النوع float.

٥-٣٦) أ) اكتب عنوان دالة تعيد قيمة ، حيث اسم الدالة NearlyEqual وهي تستقبل ثلاثة وسطاء: num1, num2, difference من النوع float وتعيد نتيجة منطقية (Boolean result).

ب) اكتب جسم الدالة NearlyEqual حيث تعيد الدالة النتيجة true إذا كانت القيمة المطلقة (absolute value) للفارق بين num1, num2 أقل من القيمة الموجودة في difference، وما عدا ذلك فإنها تعيد النتيجة false.

٥-٣٧) اكتب دالة تعيد قيمة اسمها (اسم الدالة) CompassHeading تعيد مجموع (sum) قيم وسطائها – من النوع float– الأربعة:

trueCourse, windCorrAngle, variance, deviation

٥-٣٨) اكتب دالة تعيد قيمة اسمها FracPart تستقبل عدداً من النوع float، وتعيد الجزء الكسري (fractional part) من هذا العدد. استخدم وسيطاً وحيداً اسمه x. مثلاً إذا كانت قيمة x الداخلة (incoming value of x) هي 16.753 فإن الدالة تعيد القيمة 0.753.

٥-٣٩) اكتب دالة تعيد قيمة اسمها Circumf توجد محيط (circumference) دائرة إذا أعطيت نصف قطرها (radius). صيغة حساب محيط الدائرة هي: π مضروبة في ضعف نصف القطر.
استخدم القيمة $\pi = 3.14159$.

٥-٤٠) إذا أعطيت عنوان الدالة

```
float Hypotensue (   float sidel;
                     float sisde2  )
```

اكتب جسم الدالة لتعيد طول وتر (hypotenuse) مثلث قائم الزاوية (right triangle). الوسيطان يمثلان طولي الضلعين (two sides) الآخرين . صيغة (formula) الوتر هي:

$$\sqrt{side1^2 + side2^2}$$

٥-٤١) اكتب دالة تعيد قيمة FifthPow تعيد القوة الخامسة (الأس الخامس) (fifth power) لوسيطها الذي هو من النوع float.

٥-٤٢) اكتب دالة تعيد قيمة Min ، حيث تعيد الدالة أصغر (smallest) وسطائها الثلاثة الذين هم جميعاً أعداد صحيحة.

٥-٤٣) شروط If التالية تُقبل (تعمل صحيحة work correctly) في معظم – ولكن ليس جميع – الأجهزة / الماكينات (machines) .

أعد كتابة هذه الشروط مستخدماً الدوال "is..." من مكتبة C++ القياسية [ملف المقدمة cctype (header file)].

(أ)
```
if (inChar >= '0' && inChar <= '9')
   DoSomething ( );
```

(ب)
```
if (inChar >= 'A' && inChar <= 'Z' | |
   inChar >= 'a' && inChar <= 'z'   )
      DoSomething ( );
```

(ج)
```
if (inChar >= 'A' && inChar <= 'Z' | |
   inChar >= '0' && inChar <= '9'   )
      DoSomething ( );
```

(د)
```
if (inChar < 'a'  | | inChar > 'z')
```

```
DoSomething ( );
```

٤٤-٥) اكتب دالة منطقية تعيد قيمة IsPrime تستقبل وسيطاً صحيحاً n، وتختبره لترى إن كان عدداً أولياً (prime number) أم لا، وتعيد النتيجة true إن كان عدداً أولياً. [العدد الأولى هو عدد صحيح أكبر من أو يساوي 2 ، وقواسمه الوحيدة (only divisors) هي: 1 والعدد نفسه] . ويمكن استدعاء هذه الدالة بعبارة مثل

```
if (IsPrime (n))
        cout << n << " is a prime number.";
```

إرشاد: إذا لم يكن n عدداً أولياً فإنه يقبل القسمة بالضبط (exactly divisible by) على عدد صحيح في المدى من 2 إلى $\sqrt{n}$.

٤٥-٥) اكتب دالة تعيد قيمة Postage تعيد تكاليف (cost) إرسال طرد بالبريد (mailing a package) إذا أعطيت وزن (weight) الطرد بالأرطال (pounds) والأوقيات (ounces) ، وتكلفة الأوقية الواحدة (cost per ounce). (ملاحظة: الرطل = ١٦ أوقية).

٤٦-٥) أضف التحقق من صحة البيانات (data validation)-الموضح فيما يلي - إلى برنامج تحويل التواريخ ConvertDates program (مثال ٥-١٥) :

أ) يتحقق البرنامج من الرموز المدخلة (checks the input characters) ، ويطبع رسالة خطأ إن لم يكن أي من "الأرقام" "digits" رمزاً عددياً (numeric character) [أي واحداً من '0' إلى '9'] . ويجب كتابة هذا الاختبار للتحقق من صحة البيانات (validation test) كدالة مستقلة.

ب) يتحقق البرنامج من صحة بيانات التاريخ المعطى : فالشهر month يجب أن يقع في المدى من 01 إلى 12 ، واليوم day يجب أن يقع في المدى الصحيح بالنسبة للشهر المعطى. [فمثلاً يرفض (rejects)

تاريخاً مثل 06/31/99 لأن شهر يونيو 30 يوماً فقط] . وتذكر أن شهر فبراير قد يكون 28 أو 29 يوماً بناء على السنة المعطاة. ويجب كتابة اختبار التحقق هذا كدالة مستقلة.

٥-٤٧) عدِّل (modify) برنامج تحويل التواريخ ConvertDates program (مثال ٥-١٥) بحيث يمكنه إدخال التواريخ كأرقام (digits) تفصلها (separated by) إما شرط مائلة (/) (slashes) أو شرط عادية (أفقية) (-) (dashes).

٥-٤٨) اكتب دالة خاوية time تستقبل عددا صحيحا totmin يمثل العدد الكلي للدقائق ، وتعيد ثلاث قيم صحيحة تمثل عدد الأيام days وعدد الساعات hours وعدد الدقائق min المقابلة للقيمة المدخلة totmin

مثلا : إذا كانت totmi0n = 80 فإن :

days = 0 , hours = 1 , min = 20

ويمكن استدعاء الدالة بالعبارة :

time (totmin , days , hours , min) ;

٥-٤٩) اكتب دالة خاوية length تستقبل عددا صحيحا totinc يمثل عددا من البوصات ، وتعيد ثلاث قيم صحيحة تمثل عدد الياردات yards ، وعدد الأقدام feet ، وعدد البوصات inches المقابلة للقيمة المدخلة totinc

مثلا : إذا كانت totinc = 80 فإن :

yards = 2 , feet = 0 , inches = 8

(ملاحظة : الياردة = ٣ أقدام ، والقدم = ١٢ بوصة)

ويمكن استدعاء الدالة بالعبارة :

length (totinc , yards , feet , inches) ;

٥-٥٠) اكتب دالة خاوية time تستقبل عددا صحيحا totday يمثل عددا من الأيام ، وتعيد قيمتين صحيحتين تمثلان عدد الأسابيع weeks ، وعدد الأيام days المقابلة للقيمة المدخلة totday.

مثلا : إذا كانت totday = 80 فإن :

weeks = 11 , days = 3

ويمكن استدعاء الدالة بالعبارة :

time (totday , weeks , days) ;

٥-٥١) اكتب دالة خاوية trig تأخذ عددا صحيحا angle يمثل زاوية بالدرجات degrees وتطبع كلا من جيب sine وجيب تمام cosine الزاوية بالصيغة التالية :

SINE (xxx.xx DEGREES) = x.xxxx
COSINE (xxx.xx DEGREES) = x.xxxx

افرض أن الثابت pi قد تم الإعلان عنه في البرنامج الرئيسي وأن قيمته 3.14159.

يتم استدعاء الدالة بالعبارة trig (angle) ;

٥-٥٢) أ) اكتب دالة خاوية تستقبل ثلاثة أعداد وتعيد أكبر عنصر في الثلاثة.

ب) اكتب برنامجا يشتمل على الدالة المذكورة سابقا، ويستمر البرنامج في قراءة ثلاثة أعداد مدخلة على سطر واحد ثم طباعة أكبر عنصر في الأعداد الثلاثة المدخلة ، ويتوقف البرنامج عندما تكون الأعداد الثلاثة المدخلة متساوية.

فيما يلي مثال لنتائج تشغيل البرنامج :

4	7	3	max = 7
8	9	0	max = 9
23	7	74	max = 74
1	0	4	max = 4

3 3 3 stop

٥-٥٣) اكتب دالتين خاويتين إحداهما لإظهار / لعرض (displaying) مثلث (Triangle) والأخرى لإظهار مستطيل (Rectangle) . استخدم هاتين الدالتين لكتابة برنامج كامل يعرض شكلا تخطيطيا لبيتين ، مستعينا بالخطوات الرئيسية التالية :

```
Program StackHouses;
{
  // 1. Draw Triangle
  // 2. Draw Rectangle
  // 3. Print 2 blank lines
  // 4. Draw Triangle
  // 5. Draw Rectangle
}
```

٥-٥٤) اكتب عدة دوال خاوية لعرض الحروف الأولى من اسمك ، ثم اكتب برنامجا ينفذ هذه الدوال.

٥-٥٥) اكتب دالتين خاويتين لرسم مربع ومثلث. ثم اكتب برنامجا يقرأ حرف S أو حرف T وبناء على هذا الحرف الذي يقرأه يرسم مربعا (Square) أو مثلثا (Triangle) على الترتيب.

٥-٥٦) أضف إلى برنامج السؤال السابق دالتين خاويتين لرسم دائرة وخط مستقيم. ثم أعد كتابة البرنامج الرئيسي بحيث يقرأ ثلاثة حروف من المجموعة : C (تعني دائرة Circle) و L (تعني خطا مستقيما Line) و S (تعني مربعا Square) و T (تعني مثلثا Triangle) بحيث أن هذه الحروف تمثل متغيرات من النوع الرمزي char. ثم يرسم البرنامج الثلاثة أشكال المعنية. فمثلا إذا كانت البيانات هي CLT فإن البرنامج يرسم دائرة ثم مستقيما ثم مثلثا.

٥-٥٧) اكتب برنامجا لقراءة رقم أحد الفصول (الشعب / الغرف) وسعة الفصل وعدد الطلاب المقيدين في الفصل ، ومن ثم يقوم البرنامج بطباعة رقم الفصل وسعته وعدد المقاعد الموجودة في الفصل والمشغولة بواسطة الطلاب (أي عدد الطلاب المقيدين) وعدد المقاعد الفارغة (غير المشغولة)، وأيضا يقوم بطباعة رسالة تفيد ما إذا كان الفصل مشغولا بأكمله أم لا.

استخدم دالة خاوية لطباعة العناوين التالية قبل سطر المخرجات.

Room Capacity Enrollment Empty Seats Filled Not Filled

واستخدم البيانات التالية كمدخلات للبرنامج.

Room	Capacity	Enrollment
426	25	25
327	18	14
420	20	15
317	100	90

٥-٥٨) أ) اكتب دالة تعيد قيمة Area تحسب مساحة مثلث بدلالة أطوال أضلاعه a , b , c حيث :

$$Area = \sqrt{s(s-a)(s-b)(s-c)}$$

حيث s تمثل نصف المحيط : $s = \frac{a+b+c}{2}$

ب) اكتب برنامجا رئيسيا يقرأ أطوال أضلاع مثلث side1 , side2 , side3 ويحسب مساحته باستخدام الدالة Area.

٥-٥٩) اكتب دالة totinc توجد مجموع وسطائها الثلاثة yards , feet , inches (أطوال باليادة والقدم والبوصة) ، أي توجد الطول الإجمالي محسوبا بالبوصة.

مثلا : totinc (3 , 1 , 8) = 128

(ملاحظة : اليادة = ٣ أقدام ، والقدم = ١٢ بوصة).

٥-٦٠) اكتب دالة totdys توجد مجموع وسيطيها days (عدد الأيام) و weeks (عدد الأسابيع) محسوبا بالأيام.

مثلا : totdys (5 , 9) = 44

٥-٦١) اكتب دالة منطقية bigeng لها ٣ وسطاء a , b , c عبارة عن أعداد صحيحة، وتكون قيمة الدالة true عندما يتحقق (على الأقل) واحد من الشروط الثلاثة :

a > bc , b > ac , c > ab

وما عدا ذلك فإن قيمة الدالة تكون false.

٥-٦٢) اكتب دالة منطقية small لها ٣ وسطاء a , b , c عبارة عن أعداد حقيقية ، وتكون قيمة الدالة true إذا تحقق أحد الشروط الثلاثة :

a < b/c , b < a/c , c < a/b

وما عدا ذلك فإن قيمة الدالة تكون false.

٥-٦٣) نفرض أن الدالة المنطقية differ تستقبل ثلاثة أعداد صحيحة a , b , c وتعطي القيمة true إذا كان a أصغر من b-c أو كان b أصغر من a-c ، وما عدا ذلك فإنها تعطي القيمة false. اكتب الدالة differ.

٥-٦٤) اكتب دالة تعيد قيمة (من النوع float) cubert تستقبل وسيطا واحدا x عبارة عن عدد من النوع float ، وتوجد جذره التكعيبي.

٥-٦٥) اكتب دالة منطقية xor تستقبل وسيطين منطقيين a , b وتوحد ناتج عملية منطقية (boolean operation) تسمى "أو المتنافية / المتباعدة" (exclusive or) ، وتعرف بأن ناتجها يكون صادقا true إذا كان a (فقط) أو

b (فقط) صادقا true ، بينما إذا كان كل من a , b صادقا true ، أو كان كل من a , b خاطئا false فإن ناتج العملية xor يكون خاطئا false.
(أي أن ناتج xor هو نفسه ناتج or في جميع الحالات باستثناء حالة واحدة فقط وهي عندما يكون كل من a , b صادقا حيث يكون ناتج or صادقا true بينما يكون ناتج xor خاطئا false).

٥-٦٦) اكتب دالة تعيد قيمة float تحسب مساحة شبه منحرف (trapezoid) إذا علم طولا قاعدتيه المتوازيتين b1 , b2 وارتفاعه h. الدالة لها ٣ وسطاء (b1 , b2 , h).

٥-٦٧) أ) اكتب دالة تعيد قيمة degr لتحويل قيمة (زاوية) معطاة بالتقدير الدائري (radians) إلى قيمتها مقدرة بالدرجات (degrees). وإذا لم تقع القيمة بالدرجات في المدى من 0 إلى 360 فإنها تُحَوَّل إلى القيمة المكافئة بالدرجات التي تقع في هذا المدى . مثلا : degr (3.0 * pi) يجب أن تعيد القيمة 180.0 ، وبالمثل degr (-1.0*pi) يجب أن تعيد القيمة 180.0.

ب) اكتب دالة تعيد قيمة radn لتحويل قيمة (زاوية) معطاة بالدرجات (degrees) إلى قيمتها بالتقدير الدائري في المدى من 0 إلى 2.0*pi . وإذا لم تقع القيمة بالتقدير الدائري في هذا المدى فإنها تُحَوَّل إلى القيمة المكافئة بالتقدير الدائري التي تقع في هذا المدى . مثلا radn (540.0) يجب أن تعيد القيمة pi ، وبالمثل فإن radn (-180.0) يجب أن تعيد القيمة pi.

٥-٦٨) يعرف مضروب أي عدد صحيح موجب بأنه حاصل ضرب جميع الأعداد الصحيحة من ١ إلى هذا العدد الموجب (فمثلا : مضروب ٤ = ١ × ٢ × ٣ × ٤)

أ) اكتب دالة تعيد قيمة لحساب !n أي مضروب n ، حيث n عدد صحيح موجب.

ب) اكتب برنامجا يستخدم الدالة السابقة لحساب قيمة تقريبية للعدد الحقيقي e باستعمال الصيغة

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!} + \cdots$$

احسب هذا المجموع إلى أن تصبح قيمة الحد $\frac{1}{n!}$ أقل من 0.0001.

٥-٦٩) اكتب دالة تعيد قيمة ADD(J,K) والتي قيمتها تساوي المجموع

$$\frac{1}{J} + \frac{1}{J+1} + \frac{1}{J+2} + \cdots + \frac{1}{K}$$

إذا كانت $J \le K$ ،

بينما إذا كانت $J > K$ فإن قيمة الدالة ADD تساوي صفرا. (كل من J,K عدد صحيح).

٥-٧٠) اكتب دالة خاوية

SUB (J, K, SUM , PROD, DIFF)

حيث J , K : عددان صحيحان

SUM : تمثل المجموع J + K

PROD : تمثل حاصل الضرب J * K

DIFF : تمثل الفارق J - K

٥-٧١) عَرِّف دالة تعيد قيمة لحساب

$$R = \sqrt{u^2 + v^2 + w^2}$$

ثم استخدمها لحساب :

$$A = \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

$$B = \sqrt{4x^4 + 9y^2 + 25z^2}$$

$$C = \sqrt{9 + Sin^2 y + u^2}$$

٥-٧٢) أ) اكتب دالة تعيد قيمة لحساب Fact (k) = k!

ب) اكتب دالة أخرى تعيد قيمة CNR (n , r) تستخدم الدالة السابقة لحساب قيمة

$$C(n,r) = \frac{n!}{r!\ (n-r)!}$$

والتي تعطي عدد التوافقات أي المجموعات المختلفة الممكن تكوينها من عدد n من العناصر بحيث نأخذ r عنصرا في المرة الواحدة.

ج) اكتب برنامجا يستدعي الدالة CNR للقيم التالية :

أولا : r = 1 , n = 4

ثانيا : r = 3 , n = 5

ثالثا : r = 7 , n = 7

ويطبع قيمة الدالة في كل حالة من الحالات الثلاث.

٥-٧٣) يصعب على المرضى الذين يتعاطون أنواعا كثيرة من الأدوية تذكُّر متى يتناولون هذه الأدوية. نفرض انك أعطيت مجموعة الوصفات الطبية التالية. اكتب برنامجا يطبع جدولا يعطي الأدوية التي يجب تناولها في أي ساعة معينة. استخدم متغيرا عدادا Hour للتعبير عن الأربع والعشرين ساعة خلال اليوم ، واستخدم إجراءات لطباعة العناوين والوصفات الطبية.

الدواء	مواعيد تعاطي الدواء
Iron	0800 , 1200 , 1800
Antibiotic	كل ٤ ساعات ابتداء من الساعة 0400

Vitamin	0800 , 2100
Calcium	1100 , 2000

٥-٧٤) مجلة شهرية تريد برنامجا لطباعة رسائل تجديد وإلغاء renewal and cancellation notices اشتراكات مشتركيها (subscribers). اكتب برنامجا يستخدم برامج فرعية (دوال خاوية) كلما كان ذلك أفضل ، بحيث يقرأ البرنامج أولا الشهر الحالي (current month) (وهو عدد بين ١ و ١٢) والسنة الحالية (current year) ، ثم يقرأ لكل مشترك ٤ بيانات وهي : رقم الحساب ، وشهر وسنة بداية الاشتراك ، وعدد السنوات المدفوع لها الاشتراك. اقرأ كل مجموعة من بيانات الاشتراك ثم اطبع رسالة تجديد الاشتراك إذا كان الشهر الحالي هو شهر انتهاء الاشتراك (expiration) أو الشهر السابق له ، ورسالة إلغاء الاشتراك إذا كان الشهر الحالي بعد شهر انتهاء الاشتراك.

نموذج للبيانات المدخلة :

02 10 : الشهر الحالي أكتوبر عام ٢٠٠٢

3 01 4 1364 : الحساب رقم ١٣٦٤ بدأ صاحبه الاشتراك في إبريل عام ٢٠٠١ ولمدة ثلاثة أعوام.

٥-٧٥) يمكن تقريب حساب الجذر التربيعي لأي عدد N باستخدام الصيغة التكريرية (iterative formula) : NG = 0.5 (LG + N/LG) حيث NG تمثل التخمين التالي (next guess) (أي القيمة التقريبية التالية) ، و LG تمثل التخمين الحالي/ السابق (last guess) (أي القيمة التقريبية الحالية).

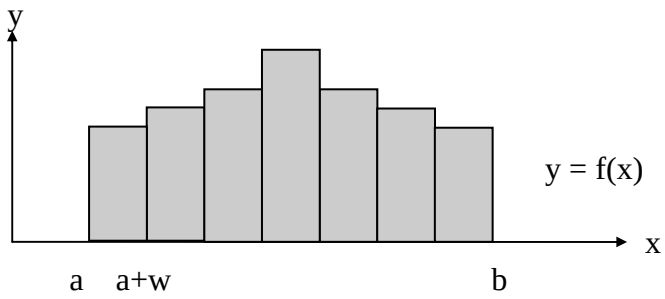
اكتب دالة خاوية تطبق هذه الطريقة بحيث تشتمل الدالة على ثلاثة وسطاء : الوسيط الأول : عدد موجب (من النوع float) (المطلوب إيجاد جذره التربيعي) ، والوسيط الثاني : تخمين ابتدائي للجذر التربيعي ، والوسيط الثالث : النتيجة المحسوبة (الجذر التربيعي المطلوب).

التخمين الأول سيكون هو القيمة الابتدائية للمتغير LG. ثم تحسب الدالة قيمة NG باستخدام العلاقة التكريرية ، ثم توجد الفارق بين NG و LG لتتحقق ما إذا كانت القيمتان متساويتين تقريبا. فإن كانتا كذلك فإننا نخرج من الدالة حيث NG هو الجذر التربيعي المطلوب ، وإلا فإن قيمة التخمين الجديد LG تصبح هي آخر تخمين NG محسوب ، ونكرر العملية ، أي نحسب قيمة أخرى للمتغير NG ، ثم نوجد الفارق للتحقق وهكذا.

بالنسبة لهذا البرنامج كرر العروة إلى أن تصبح قيمة هذا الفارق (DELTA) أقل من 0.005 . استخدم القيمة 1.0 كتخمين ابتدائي ، واختبر صحة البرنامج بالنسبة للأعداد (N) :

4 , 120.5 , 88 , 36.01 , 10,000

٥-٧٦) نستطيع تقريب المساحة تحت المنحنى $y = f(x)$ بواسطة تقسيم المساحة إلى عدد من المستطيلات ، ثم حساب مجموع مساحات هذه المستطيلات. الشكل التالي مثال لطريقة " نقطة المنتصف " (midpoint method) لحساب المساحة وقد سميت الطريقة بهذا الاسم لأن المنحنى يقطع كل مستطيل عند منتصفه (مقاسا في اتجاه محور x) ، وتكون مساحة كل مستطيل عبارة عن ضرب العرض الثابت w في قيمة الدالة عند نقطة المنتصف (midpoint) (انظر الشكل).



مساحة المستطيل الذي له نقطة نهاية يسرى x_1 هي $w*f(x_1 + w/2)$.

إذا كانت الفترة [a , b] مقسمة إلى عدد n من المستطيلات ، فإن المساحة تحت المنحنى يمكن أن تمثل بالمجموع الآتي :

$$\text{Area} = w \sum_{i=0}^{n-1} f(a + i * w + w / 2)$$

حيث w تساوي (b - a) / n ، والمستطيل الأول يبدأ عند x = a ، والثاني عند x = a + w ، والثالث عند x = a + 2w ، وهكذا.

اكتب برنامجا يستخدم طريقة " نقطة المنتصف " لإيجاد المساحة تحت المنحنى (قيمة التكامل المحدود) للدالة $f(x) = -3x^2 + 2x + 4$ على الفترة [3 , 2-] . اختبر البرنامج مستخدما قيما مختلفة لـ n. لاحظ أنه كلما زادت قيمة n ، كان التقريب أفضل.

الفصل السادس

عبارات إضافية للتحكم والتكرار
Additional Control and Repetition Statements

نناقش في هذا الفصل بإذن الله خمس عبارات من عبارات التحكم والتكرار ، وهي تضاف للعبارات الأساسية التي درسناها سابقاً في الفصلين الثالث والرابع. وهذه العبارات الخمس هي:

١. **عبارة** switch
٢. **عبارة** do .. while
٣. **عبارة** for
٤. **عبارة** break
٥. **عبارة** continue

(١) عبارة switch

هذه العبارة هي بنية تحكم / اختيار تسمح بأي عدد من الاختيارات / التفرعات (branches)، **فهي تشبه عبارات** If **المتداخلة. والصيغة العامة لعبارة** switch **هي:**

```
switch (expression)
{
        label1 :  statement1
        label2 :  statement2
        ⋮
```

```
        label_i :  statement_I
        :
        label_n :  statement_n
}
statement
```

حيث expression **تعبير من النوع التكاملي** (integral type):

char, short, int, long, bool

أو من النوع التعددي enum type **(وسنتناول بإذن الله تعريف هذا النوع في الفصل التالي). وهذا التعبير يجب ألا يكون تعبيراً من نوع النقطة العائمة** floating point **أو من نوع سلاسل الرموز** (string). **وقيمة هذا التعبير تحدد أي عنوان** $label_i$ **يتم اختياره لتنفيذ العبارة المقابلة له.**

وأما العنوان $label_i$ **فإما أن يكون عنوان حالة** case label **أو عنوان بديل افتراضي** default label. **وعنوان الحالة هو**

case ConstantExpression:

حيث ConstantExpression **تعبير تكاملي أو تعددي** integral/enum expression **معامَلاته ثوابت حرفية** (literal constant) **أو ثوابت مسَّماة** (named constants) . **ومن أمثلة التعابير التكاملية الثابتة** (constant integral expressions) **[حيث** CLASS_SIZE **هو ثابت مسمَّى من النوع** int] :

```
3
'A'
CLASS_SIZE
2 * CLASS_SIZE +1
```

وعنوان البديل الافتراضي هو :

default:

وأما تنفيذ عبارة switch فيتم بالطريقة التالية: يقيَّم أولاً تعبير switch (expression). فإن كانت قيمته (value) موافقة (matching) لإحدى القيم الموجودة في عناوين الحالة (case labels) فإن التحكم (control) ينتقل / يتفرع (branches) إلى العبارة $statement_i$ التي تلي عنوان الحالة هذا $case\ label_i$ (الذي حدث عنده التوافق). ومن هناك ينتقل التحكم تتابعياً (sequentially) حتى يصل إما إلى نهاية عبارة switch أو إلى عبارة break (وسنتناول هذه العبارة ومعناها بعد قليل). ولإن لم تكن قيمة تعبير switch موافقة لأي قيمة حالة case value فيحدث أحد أمرين:

*) إن كان هناك عنوان بديل افتراضي (default label) فإن التحكم ينتقل إلى العبارة التي تلي هذا العنوان (label) .

*) وإن لم يكن هناك عنوان بديل افتراضي فإن التحكم يتخطى (skips) جميع العبارات الموجودة داخل عبارة switch وينتقل إلى العبارة التي تلي عبارة switch.

نفرض مثلاً أن لدينا عبارة switch التالية:

```
switch (letter)
{
        case 'X'  : Statement1 ;
                    break ;
        case 'L'  :
        case 'M'  : Statement2 ;
                    break ;
        case 'S'  : Statement3 ;
                    break ;
        defualt   : statement4 ;
}
Statement5 ;
```

تعبير switch هنا هو :letter. ومعنى عبارة switch هذه:

*) إذا كانت قيمة letter هي 'X' نفِّذ العبارة Statement1 ثم اقطع تنفيذ عبارة switch – أي اخرج منها – واتجه إلى العبارة التي تليها، أي نفِّذ العبارة Statement5.

*) وإذا كانت قيمة letter هي 'L' أو 'M' نفِّذ Statement2 ثم Statement5.

*) وإذا كانت قيمة letter هي 'S' نفِّذ Statement3 ثم Statement5.

*) وإذا لم تكن قيمة letter أيّاً من الرموز المذكورة سابقاً نفِّذ العبارة Statement4 ثم Statement5.

فعبارة break تؤدي إلى خروج فوري(immediate exit) من عبارة switch.
وعناوين الحالة (case labels) في هذا المثال هي:

```
case 'X'
case 'L'
case 'M'
case 'S'
```

وكما نرى في هذا المثال فيمكن لأكثر من عنوان حالة واحد أن يسبق(precede) عبارة واحدة (single statement). [لاحظ أن كلا من case 'L', case 'M' تسبق Statement2] . ولكن لا يجوز لأي قيمة حالة (case value) أن تظهر أكثر من مرة واحدة فقط في عبارة switch. وإذا ظهرت أي قيمة حالة أكثر من مرة واحدة فقط نتج لدينا خطأ تركيبي (syntax error). وكذلك لا يجوز أن يظهر أكثر من عنوان بديل افتراضي (default label) واحد فقط في عبارة switch.

مثال ٦-١: أكتب قطعة برنامج تقوم بما يلي:

*) **إذا كان التقدير** (grade) **هو** 'A' **أو** 'B' **: تطبع الرسالة** "Good Work" .

*) **إذا كان التقدير هو** 'C' **: تطبع الرسالة** "Average Work" .

*) **إذا كان التقدير هو** 'D' **أو** 'F' **: تطبع الرسالة** "Poor Work" ، **وكذلك تزيد عدد الطلاب الضعفاء** numberInTrouble **بواحد.**

*) **إذا لم يكن التقدير أياً من الرموز السابقة تطبع الرسالة** "is not a valid letter grade" .

(أ) **استخدم عبارة** switch (ب) **استخدم عبارة** if **المتداخلة.**

الحل: (أ) **باستخدام عبارة** switch :

```
switch (grade)
{
        case 'A'  :
        case 'B'  : cout << "Good Work" ;
                    break ;

        case 'C'  : cout << "Average Work" ;
                    break ;
        case 'D'  :
        case 'F'  : cout << "Poor Work" ;
                    numberInTrouble++ ;
                    break ;
        default   : cout << grade << "  is not  valid letter grade.";
                    break ;
}
```

(ب) باستخدام عبارة if المتداخلة:

```
if (grade = =  'A' | | grade  = = 'B')
    cout << "Good Work" ;
else if (grade = = 'C')
    cout << "Average Work" ;
else if (grade = = 'D' | | grade  = = 'F')
{
   cout << "Poor Work" ;
   numberInTrouble++ ;
}
else
   cout << grade << "  is not  a valid letter grade.";
```

ذكرنا سابقاً أنه بعد تنفيذ عبارة ما $statement_i$ **داخل عبارة** switch **ينتقل التحكم تتابعياً حتى يصل إما إلى نهاية عبارة** switch **أو إلى عبارة** break **. ولذلك فقد نحصل على نتائج خاطئة إذا حذفنا عبارات** break **من عبارة** switch**، كما يوضح ذلك المثال التالي.**

مثال ٦-٢: تتبع تنفيذ قطعة البرنامج التالية واكتب مخرجاتها عندما تكون قيمة grade **هي:**

'H' (أ) 'A' (ب)

```
switch (grade)                  // Wrong version
{
        case 'A'  :
        case 'B'  : cout << "Good Work" ;
        case 'C'  : cout << "Average Work" ;
        case 'D'  :
        case 'F'  : cout << "Poor Work" ;
                    numberInTrouble++ ;
        default   : cout << grade << "  is not a valid letter grade.";
}
```

الحل :

(أ) H is not a valid letter grade

(ب) Good WorkAverage WorkPoor WorkA is not a valid letter grade.

ملاحظة ١: إذا لزم الأمر يتم تحويل ضمني (coercion) لنوع بيانات (data type) التعبير ConstantExpression في عنوان الحالة Case ConstantExpression ليوافق (to match) نوع (type) تعبير switch (expression).

ملاحظة ٢: عناوين الحالة (case labels) في عبارة switch يجب أن تكون مكونة من قيم (values) وليس متغيرات (variables). ومن الممكن أن يشتملوا على ثوابت مسماة وتعابير تحتوي على ثوابت فقط. وتعبير switch لا يمكن أن يكون من نوع النقطة العائمة أو تعبير سلاسل رموز. وكذلك ثوابت الحالة (case constants) لا يمكن أن تكون من نوع النقطة العائمة أو ثوابت سلاسل رموز.

(٢) عبارة do .. while

هذه العبارة هي بنية تكرار يُختبر فيها شرط التكرار في نهاية / آخر العروة (بدلاً من أولها كما في حالة عبارة while). ولذلك فجسم عروة do..while.. يُنفَّذ على الأقل مرة واحدة.
والصيغة العامة لهذه العبارة هي:

```
do
        Statement
while   (Expression);
```

حيث Statement **إما عبارة واحدة** (single statement) **أو عبارة مركبة / قالب** (block):

```
do
{
        Statement1;
        Statement2;
          ⋮
        StatementN;
}  while  (Expression);
```

ومعنى عبارة do..while: **استمر في تنفيذ جميع العبارات الموجودة بين** do **و** while **طالما أن قيمة التعبير** Expression **هي** true **عند نهاية العروة.**

وفيما يلي بعض الأمثلة التي تقارن بين عروة do..while **وعروة** while **لحل المسألة ذاتها.**

مثال ٦-٣: اكتب قطعة برنامج لإيجاد أول نقطة (first period) **في ملف بيانات** (file of data). **افرض أنه يوجد على الأقل نقطة واحدة في الملف. استخدم**

(أ) عبارة while **(ب) عبارة** do..while

الحل: (أ)

```
dataFile >> inputChar;
while  (inputChar ! = ' . ' )
        dataFile >> inputChar;
```

(ب)

```
do
        dataFile >> inputChar;
```

while (inputChar ! = ' . ')

نلاحظ أن الحل باستخدام عروة while **يتطلب قراءة أولية** (priming read) **حتى يكون للمتغير** inputChar **قيمة قبل دخول العروة (واختبار الشرط) بينما لا نحتاج لـذلك في الحـل باسـتخدام** do..while **لأن عبـارة الإدخـال** input statement **داخل العروة تنفَّذ قبل تقييم شرط العروة.**

مثـال ٦-٤: اكتـب قطعـة برنـامج تقـرأ عمـر شـخص (age) **بطريقـة تبادليـة** (interactively) **مع المستخدِم (أي تحثه أولاً على إدخال العمر، ثم تقرأه وتقوم بتخزينه). ونظراً لأن العمر يجب أن يكون قيمة موجبة ، فيظل البرنامج يطلب من الشخص إدخال قيمة صحيحة (موجبة) للعمر إلى أن يدخلها . استخدم**

(أ) عبارة while **(ب) عبارة** do..while

الحل: (أ) باستخدام عبارة while

```
cout << "Enter your age:  ";
cin >> age;
while (age <= 0)
{
        cout << "Your age must be positive." << endl;
        cout << "Enter your age:  ";
        cin >> age;
}
```

(ب) باستخدام عبارة do..while

```
do
{
        cout << "Enter your age:  ";
        cin >> age;
        if (age  <= 0)
        cout << "Your age must be positive." << endl;
} while (age <= 0);
```

لاحظ أن هذا الحل باستخدام عبارة do .. while لا يتطلب ظهور عبارتي الحث وإدخال البيانات مرتين : مرة قبل العروة وأخرى داخلها، ولكنه يختبر القيمة المدخلة (input value) مرتين.

كذلك يمكننا استخدام عبارة do .. while لتنفيذ عروة محكومة بعدد إذا كنا نعلم سلفاً أن جسم العروة يجب أن ينفَّذ دائماً مرة واحدة على الأقل.

مثال ٦-٥: اكتب قطعة برنامج لإيجاد مجموع الأعداد الصحيحة من 1 إلى n ، وذلك باستخدام

(أ) عبارة while (ب) عبارة do..while

الحل: (أ) باستخدام عبارة while

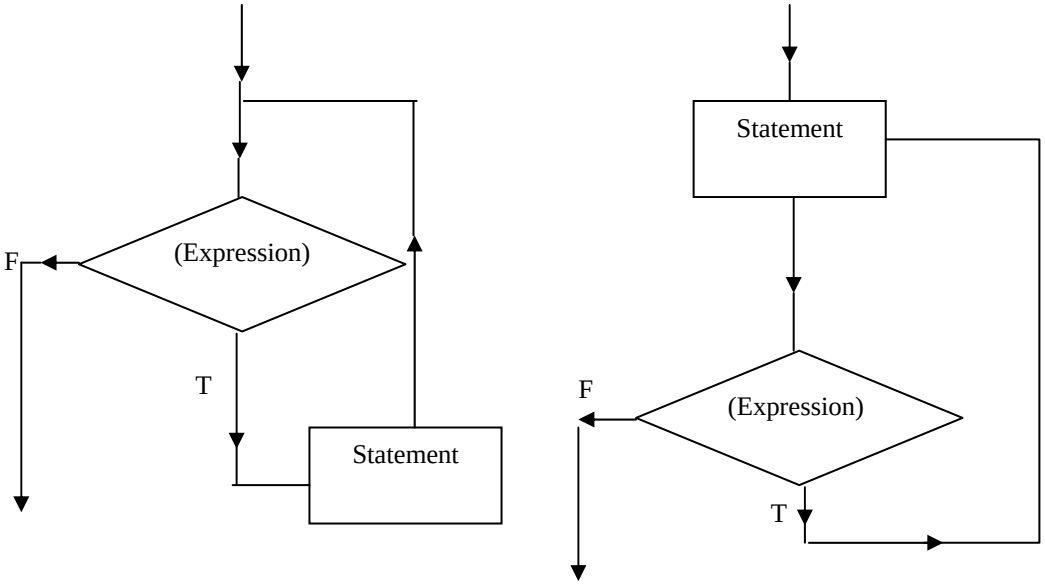
```
sum = 0;
counter = 1;
while (counter <= n)
{
        sum = sum + counter;
        counter++;
}
```

(ب) باستخدام عبارة do..while

```
sum = 0;
counter = 1;
do
{
        sum = sum + counter;
        counter++;
} while (counter <=n)
```

إذا كانت n عدداً موجباً فإن القطعتين السابقتين متكافئتان وتعطيان النتيجة نفسها. أما إن كانت n صفراً أو عدداً سالباً فإن القطعتين تعطيان نتيجتين مختلفتين ، وذلك لأن القطعة الأولى (أ) تعطي مجموعاً قيمته النهائية sum = 0 لأنه يتم تخطى جسم العروة (loop body) ولا ندخله أبداً.أما القطعة الثانية (ب) فتعطي قيمة نهائية sum =1 لأنه يتم تنفيذ جسم العروة مرة واحدة، ثم يُجرَى اختبار العروة.

ويطلق على عروة while "عروة سابقة الاختبار" (pretest loop) لأن العروة تختبر الشرط قبل تنفيذ جسم العروة، بينما عروة do..while يطلق عليها "عروة لاحقة الاختبار" (posttest loop) لأن اختبار الشرط يكون بعد تنفيذ جسم العروة. والشكل التالي يقارن بين خريطتي سير عمليات العروتين .



while **do .. while**

شكل ٦-١

خريطتا سير عمليات عروتي
while, do .. while

(٣) عبارة for (for statement)

صممت هذه العبارة لتبسيط كتابة العروة المحكومة بعدد.

والصيغة العامة لهذه العبارة هي:

```
for (InitStatement; Expressionl; Expression2)
    Statement
```

حيث

Expression1 : هو شرط while

InitStatement : يمكن أن يكون أياً مما يلي:

- العبارة الخالية (null statement)

- عبارة تعبير (expression statement)

- عادةَ يعطي InitStatement القيمة الابتدائية (initial value)

Expression2 : عادة يزيد هذا التعبير أو ينقص (increments / decrements) قيمة متغير التحكم في العروة.

فمثلاً كل من العروتين التاليتين يتم تنفيذها 50 مرة.

```
for (loopCount = 1; loopCount <= 50; loopCount++)
⋮
for (loopCount = 50; loopCount >= 1; loopCount--)
⋮
```

مثال ٦-٦: اكتب عبارة for لطباعة الأعداد الصحيحة من 1 إلى n .

الحل:

```
for (count = 1; count <= n; count++)
      cout << count << endl;
```

هذه العبارة تعطي متغير التحكم في العروة count **القيمة الابتدائية** 1. **وطالما أن** count <= n **ننفِّذ عبارة الطباعة ونزيد** count **بواحد. ونوقف العروة بعدما تصل قيمة** count **إلى** n+1.

ففي لغة C++ **تعد عبارة** for **مجرد صيغة مختصرة لعروة** while. **وفي الواقع فإن المترجم** (compiler) **يترجم أساساً عبارة** for **إلى عروة** while **مكافئة. فمثلاً عبارة** for **السابقة تترجم إلى عروة** while **التالية:**

```
count = 1 ;
while (count <= n)
{
    cout << count << endl;
    count ++ ;
 }
```

وكما رأينا سابقاً عند دراسة عرى while **أنه يجوز كتابة عرى متداخلة ، فكذلك الحال مع عرى** for **وعرى** while .. do **يجوز أن تتداخل . فالمثال التالي يعرض بنية** for **متداخلة** (nested for structure) .

مثال ٦–٧: تتبع تنفيذ قطعة البرنامج التالية واكتب مخرجاتها.

```
for (lastNum = 1; lastNum <= 7; lastNum++)
{
    for (numToPrint = 1; numToPrint <= lastNum; numToPrint++)
         cout << numToPrint;
    cout << endl;
}
```

الحل: تطبع القطعة مثلث الأعداد التالي:

```
1
12
123
1234
12345
123456
1234567
```

ورغـم أن عبـارة for **تستخـدم أساسـاً للعـرى المحكومة بعدد، إلا أن لغة** C++ **تسمـــح لنـــا بكتابـــة عـــروة** for **مكافئـــة لأي عـــروة** while**. وفيمـــا يلـي بعض الملاحظات حول عبارة for**

(١) **عروة** for **لا تنفَّذ أي مرة إذا كانت القيمة الابتدائية** (starting value) **أكبر من القيمـة النهائيـة** (ending value) **ومـتغير الـتحكم في العـروة في تزايـد** (incremented) **، أو إذا كانت القيمة الابتدائية أصغر من القيمة النهائية ومتغير التحكم في العروة في تناقص** (decremented).

(٢) InitStatement **في الصيغة العامة لعبارة** for **يمكـن أن يكـون العبـارة الخاليـة** (null statement)**، وكـذلك** Expression2 **في هـذه الصيغة العامة اختيـاري** (optional)**، فإذا حُذف فلـن يكـون للمترجـم** (compiler) **عبارة يضعها أسفل العروة** (at the bottom of the loop).

مثال ٦-٨: اكتب عروة for **مكافئة لعروة** while **التالية:**

```
while (inputVal != 999)
        cin >> inputVal;
```

الحل:

```
for ( ;  inputVal != 999;)
    cin >> inputVal;
```

(٣) لاحظ أن التعبير Expression1 في الصيغة العامة لعبارة for – والذي يطلق عليه شرط while – اختياري (optional). فإذا حذفناه فإننا نفترض قيمته true.
مثلاً إذا كانت لدينا عبارة whileالتالية:

```
while (true)
   cout << "Hi " << endl;
```

فإن عبارة for التالية مكافئة لها:

```
for ( ; ; )
    cout << "Hi " << endl;
```

(٤) مما سبق يتبين لنا أن عنوان عبارة for (أول سطر) يتكون دائماً من ثلاثة أجزاء بين قوسين ، وتوجد فاصلة منقوطة بين أي جزئين متجاورين. وأي جزء من الأجزاء الثلاثة اختياري أي يمكن أن يحذف (omitted) ، ولكن الفاصلتين المنقوطتين يجب أن تبقيا ، ولا تحذف أي منهما.

(٥) يجوز لعبارة إعطاء القيمة الابتدائية InitStatement أن تكون إعلاناً (declaration) مع إعطاء القيمة الابتدائية (with initialization) ، كأن نكتب مثلاً:

```
for ( int i =1;   i <= 20;  i ++)
    cout << "IO " << endl;
```

فهنا المتغير i له مجال محلي (local scope) رغم عدم وجود قوسين { } لتحديد قالب. ومجال i يمتد حتى نهاية عبارة for فقط.

وكأي متغير محلي فلا يمكن الوصول إلى المتغير i (inaccessible) من خارج مجاله (أي من خارج عبارة for). ونظراً لأن i متغير محلي بالنسبة لعبارة for فمن الممكن كتابة قطعة برنامج كالتالية:

```
for ( int i =1;   i <= 20;  i ++)
    cout << "Hi" << endl;
for ( int i =1;   i <= 100;  i ++)
    cout << "Lo" << endl;
```

فهذه القطعة لا تؤدي إلى ظهور رسالة خطأ وقت الترجمة (compile-time error) [مثل: MULTIPLY DEFINED IDENTIFIER] ، وذلك لأننا أعلنا عن متغيرين مختلفين – اسم كل منهما i – وكل منهما متغير محلي بالنسبة لعبارة for الخاصة به . [ملاحظة : هذه الملاحظة لا تنطبق على صيغ C++ السابقة للصيغة القياسية الحالية (ISO / ANSI language standard)].

من الملاحظات السابقة يتبين لنا أن عبارة for في لغة C++ بِنْية مرنة جداً (very flexible structure)، فمدى استخدامها يمتد من عروة بسيطة محكومة بعدد إلى عبارة مكافئة لأي عروة while عامة الأغراض (general purpose while loop) . وبعض المبرمجين يضع قدراً كبيراً من المعلومات والعبارات في عنوان عبارة for (وهو أول سطر). فمثلاً قطعة البرنامج

```
cin >> ch;
while (ch != ' . ' )
        cin >> ch;
```

يمكن ضغطها في عروة for التالية:

```
for (cin >> ch; ch != ' . ' ;  cin >> ch)
     ;
```

فنظراً لأن كل المعلومات والعبارات التنفيذية وضعت في عنوان العبارة، لم يبق شئ لجسم العروة ليقوم بتنفيذه، ولذلك فجسم العروة هو الفاصلة المنقوطة فحسب. وعموماً – في هذا الكتاب – سنستخدم عبارات for للعرى المحكومة بعدد فقط.

(٤) عبارة break (break statement)

تستخدم عبارة break – التي ذكرناها سابقاً واستخدمناها مع عبارة switch – مع عبارات التكرار / العرى أيضاً.
وتأثير عبارة break هو الخروج الفوري (immediate exit) من العبارة الأعمق/ الأوغل (innermost statement) التي ظهرت فيها: عبارة switch أو عبارة while أو عبارة do .. while أو عبارة for. ولاحظ كلمة الأعمق / الأوغل ، بمعنى أنه إذا كانت break في عروة (داخلية) (nested) داخل عروة أخرى (خارجية) ، فإن التحكم (control) ينتقل / يخرج من العروة الداخلية ولكن ليس من العروة الخارجية.

ومن التطبيقات الشائعة التي تستخدم فيها عبارة break مع العرى/ عبارات التكرار كتابة عروة لانهائية (infinite loop) واستخدام شروط if للخروج من العروة.

مثال ٦–٩: اكتب قطعة برنامج لقراءة 10 أزواج (pairs) من أعداد صحيحة num1, num2 وحساب الجذر التربيعي (square root) لمجموع كل زوج أي لمجموع كل عددين . ولكن قبل حساب هذا الجذر التربيعي يجب أن تتحقق القطعة من صحة البيانات (data validation) وهي أن العدد الأول num1 من كل زوج يجب أن يكون أصغر من 100 والعدد الثاني numb2 يجب أن يكون أكبر من 50 . وكذلك بعد كل قراءة يجب أن تختبر القطعة حالة السيل (state of the stream) بالنسبة لنهاية الملف EOF. استخدم عبارات break في عروة الحل.

الحل:

```
loopCount = 1;
while (true)
{
        cin >> num1;
        if (  !cin | | num1 >= 100)
                break;
        cin >> num2;
        if (  !cin | | num2 <= 50)
                break;
        cout << sqrt (float (num1 + num2) ) << endl;
        loopCount++;
        if (loopCount > 10)
                break;
}
```

يلاحظ أنه كان يمكننا لحل هذه المسألة استخدام عروة for **للعد من** 1 **إلى** 10**، ونخرج من العروة** (breaking out of it) **عند الضرورة. ولكن نظراً لأن العروة هنا في هذه المسألة محكومة بعدد وكذلك محكومة بحدث ففضلنا استخدام عروة** while **. والعروة المذكورة في الحل تحتوي على ثلاث نقاط مختلفة للخروج من العروة.**

وفيما يلي حل آخر لهذا المثال لا يستخدم عبارات break**.**

```
num1Valid = true;
num2Valid = true;
loopCount = 1;
while (num1Valid && num2Valid && loopCount <=10)
{
        cin >> num1;
        if (  !cin | | num1 >= 100)
                num1Valid = false;
        else
        {
                cin >> num2;
                if (  !cin | | num2 <= 50)
```

```
                num2Valid = false;
        else
        {
                cout << sqrt (float (num1 + num2) ) << endl;
                loopCount++;
        }
    }
}
```

والحل الأول يبدو أيسر في التتبُّع والفهم من هذا الحل الأخير الذي يستخدم عبارات if **متداخلة.**

ولكن عموماً ليس كل حل يستخدم عبارة break **هو أبسط من حل لا يستخدمها ، كما يوضح ذلك المثال التالي.**

مثال ٦-١٠: اكتب قطعة برنامج تستخدم عروة لطباعة الأعداد الصحيحة من 1 **إلى** 5

أ) باستخدام عبارة break **ب) بدون استخدام عبارة** break

الحل: أ)

```
i = 1
while (true)
{
        cout << i;
        if (i = = 5)
            break;
        i ++;
}
```

ب)

```
for (i = 1;   i <= 5;  i ++)
        cout << i;
```

واضح أن الحل باستخدام عبارة for البسيطة بدون استخدام عبارة break أبسط من الحل باستخدام عبارة break.

وعموماً لا يُنصح باستخدام break داخل العُرَى إلا عند الضرورة ، وخاصة لتجنب التعقيدات الناتجة عن الرايات المنطقية المتعددة (multiple boolean flags) وعبارات if المتداخلة.

(٥) عبارة Continue

هذه العبارة لا تستخدم إلا مع العُرَى ، وهي تُنهي (terminates) التكرير الحالي في العروة (current loop iteration) [وليس العروة كلها (entire loop)] ، حيث ينشأ عنها انتقال / تفرع فوري (immediate branching) إلى أسفل / قاع العروة (bottom of the loop) – مع تخطي skipping بقية العبارات في جسم العروة (loop body) – استعداداً للتكرير التالي.

مثال ٦-١١: قطعة البرنامج التالية تشتمل على عروة لقراءة خمسمائة قيمة inputVal من ملف بيانات وتشغيل القيم الموجبة فقط. فإذا كانت القيمة سالبة أو مساوية للصفر فإن التحكم control ينتقل إلى أسفل العروة.

```
for (dataCount = 1;  dataCount <= 500, dataCount ++)
{
        dataFile >> inputVal;
        if  (inputVal <= 0)
            continue;
        cout << inputVal;
        .
        .
        .
}
```

وكأي عروة for فإن الحاسب يزيد قيمة dataCount ويقوم بإجراء اختبار العروة قبل أن يبدأ التكرير التالي.

وعموماً قليلا ما تستخدم عبارة continue ، وفائدتها الأساسية تجنب حجب العملية الرئيسية (main process) في العروة بإدخال هذه العملية داخل عبارة if. فمثلاً قطعة البرنامج السابقة يمكننا كتابتها كما يلي بدون استخدام عبارة continue:

```
for (dataCount = 1;  dataCount <= 500, dataCount ++)
{
        dataFile >> inputVal;
        if  (inputVal <= 0)
        {
                cout << inputVal;
                .
                .
                .
        }
}
```

ونحب أن نؤكد هنا على الفارق بين continue و break . فعبارة continue تعنى : "اترك التكرير الحالي في العروة واذهب إلى التكرير التالي", أما عبارة break فتعني: "اترك العروة كلها فوراً".

إرشادات لاختيار أي عبارات التكرار / العُرَى

لمعرفة أي عبارات التكرار: عروة while أم عروة do .. while أم عروة for أفضل للاختيار لكتابة عروة في مسألة ما، يمكن الاسترشاد بالنقاط التالية:

- إذا كانت العروة عروة بسيطة محكومة بعدد فإن عبارة for هي الأنسب، حيث أنها تشمل العمليات الثلاث التي تتحكم في العروة (وهي إعطاء القيمة

الابتدائية واختبار شرط العروة وزيادة / نقصان متغير التحكم في العروة) في موضع واحد هو إعلان عبارة for.

- إذا كانت العروة عروة محكومة بحدث بحيث أن جسم العروة سينفذ على الأقل مرة واحدة ، فتعد عبارة do .. while مناسبة.

- إذا كانت العروة عروة محكومة بحدث ولا نعلم شيئاً عن أول تنفيذ للعروة فنستخدم عبارة while (وربما عبارة for).

- إن كنت في شك أي العبارات أفضل استخدم عبارة while.

- اجعل آخر الأولويات استخدام عروة لانهائية (infinite loop) تتخللها عبارات break.

مثال ٦-١٢: اكتب برنامجاً تبادلياً (interactive program) يطلب من المستخدم (user) أن يُدخل كميات سقوط المطر الشهرية (monthly rainfall amounts) على أحد المواقع (site) خلال عام (١٢ شهراً) ، ثم يحسب ويطبع متوسط الاثنتى عشرة قيمة. وبعد تشغيل بيانات أي موقع يسأل البرنامج إن كان المستخدم يود تكرار العملية لموقع آخر. والإجابة بالرمز ' y ' تعنى: نعم (yes)، بينما ' n ' تعنى: لا (no) . ويجب أن يكتشف البرنامج أي بيانات إدخال خاطئة (أي قيم سالبة لكميات سقوط المطر، وأي إجابة خاطئة للسؤال عن الرغبة في تكرار العملية) ، ويطلب إعادة إدخال البيانات .

وفيما يلي مثال لنتائج تشغيل البرنامج.

sample run

```
Enter rainfall amount 1: 0
Enter rainfall amount 2: 0
Enter rainfall amount 3: 0
Enter rainfall amount 4: 3.4
Enter rainfall amount 5: 9.6
Enter rainfall amount 6: 1.2
Enter rainfall amount 7: -3.4
Amount cannot be negative. Enter again: -9
Amount cannot be negative. Enter again: -4.2
Amount cannot be negative. Enter again: 1.3
Enter rainfall amount 8: 0
Enter rainfall amount 9: 0
Enter rainfall amount 10: 0
Enter rainfall amount 11: 0
Enter rainfall amount 12: 0

Average rainfall is 1.29 inches

Do you have another recording site?  (y or n) d
Please type y or n: q
Please type y or n: Y
Please type y or n: n
```

الحل:

```
//*****************************************************************
 //Rainfall program
 //This program inputs 12 monthly rainfall amounts from a
 //recording site and computes the average monthly rainfall.
 //This process is repeated for as many recording sites as
 //the user wishes.

//*****************************************************************
#include <iostream>
#include <iomanip>    // For setprecision()
```

```
using namespace std;

void Get12Amounts( float& );
void GetOneAmount( float& );
void GetYesOrNo( char& );

int main( )
}
   float sum;          // Sum of 12 rainfall amounts
   char  response;   // User response ('y' or 'n')

   cout << fixed << showpoint         // Set up floating pt.
        << setprecision(2);              //   output format
   do
   {
      Get12Amounts(sum);
      cout << endl << "Average rainfall is " << sum / 12.0
           << " inches" << endl << endl;
      cout << "Do you have another recording site? (y or n) " ;
      GetYesOrNo(response);
    } while (response = = 'y');
   return 0;
}


// ***************************************************************
void Get12Amounts( /* out */ float& sum )   // Sum of 12 rainfall
                                             // amounts

 //Inputs 12 monthly rainfall amounts, verifying that
 //each is nonnegative, and returns their sum
}
   int   count;       // Loop control variable
   float amount;   // Rainfall amount for one month

   sum = 0;
   for (count = 1; count <= 12; count++)
```

```
   {
      cout << "Enter rainfall amount " << count << ":" ;
      GetOneAmount(amount);
      sum = sum + amount;
    }
}
*****************************************************************

void GetYesOrNo( /* out */ char& response )   // User response char

 //Inputs a character from the user and, if necessary,
 //repeatedly prints an error message and inputs another
 //character if the character isn't 'y' or 'n'
{
   do
   {
      cin >> response;
      if (response != 'y' && response != 'n')
         cout << "Please type y or n: ";
    } while (response != 'y' && response != 'n');
}

//*****************************************************************
void GetOneAmount( /* out */ float& amount )   // Rainfall amount
                                                  // for one month
 //Inputs one month's rainfall amount and, if necessary,
 //repeatedly prints an error message and inputs another
 //value if the value is negative
{
   do
   {
      cin >> amount;
      if (amount < 0.0)
         cout << "Amount cannot be negative. Enter again: " ;
   } while (amount < 0.0);
}
```

تمرينات رقم (٦)

٦-١) اذكر صحة أو خطأ (T / F) كل من العبارات التالية:

(أ) تعبير (switch) يمكن أن يكون أي تعبير قيمته من أي نوع من الأنواع التالية:

int, float, bool, char

(ب) القيم التي تُكتب في عناوين الحالة (case labels) في عبارة switch يمكن أن تظهر بأي ترتيب، ولكن لا يسمح بتكرار أي عنوان حالة (duplicate case labels).

(ج) جميع القيم المحتملة لتعبير switch يجب أن تحتويها عناوين الحالة في عبارة switch.

٦-٢) أعد كتابة قطعة البرنامج التالية مستخدماً عبارة switch.

```
if  (n = = 3)
        alpha ++;
else if  (n = = 7)
        beta ++;
else if  (n = = 10)
        gamma ++;
```

٦-٣) ماذا تطبع قطعة البرنامج التالية إذا كانت n تساوي 3 ؟

```
switch (n + 1)
{
```

```
        case 2  : cout << "Fatima";
        case 4  : cout << "Maryam";
        case 7  : cout << "Khadeejah";
        case 9  : cout << "Assia";
        default : cout << "Best Women!";
}
```

٦-٤) اذكر صحة أو خطأ (T / F) كل من العبارات التالية:

(أ) **إذا حوّلنا** (converted) **عروة** while **شرطها** delta <= alpha **إلى عروة** while .. do ، **فإن شرط عروة** while .. do **يكون** delta > alpha.

(ب) **عبارة** do . while **تنتهي دائماً بفاصلة منقوطة.**

(ج) **عبارة** break **الموجودة داخل عبارة** switch **التي تقع بدورها داخل عروة** while **تؤدي إلى أن ينتقل التحكم فوراً خارج العروة.**

٦-٥) ما هي مخرجات كل من القطعتين التاليتين بفرض أن القيمة المدخلة (input value) **هي** 0 ؟ **(افرض أن جميع المتغيرات من النوع** int).

(أ)
```
cin >> n;
i = 1;
do
{
    cout << i;
    i ++;
} while ( i <= n);
```

(ب)
```
cin >> n;
for  (i = 1;  i <= n;  i ++)
        cout << i;
```

٦-٦) ما هي مخرجات كل من القطعتين التاليتين بفرض أن جميع المتغيرات من النوع int ؟

(أ)

```
for (i = 4; i >= 1; i --)
{
        for (j = i ; j >= 1; j--)
                cout << j << ' ' ;
        cout << i << endl;
}
```

(ب)

```
for (row = 1; row <= 10; row ++)
{
        for (co1 = 1; co1 <= 10 - row; co1 ++)
                cout << '*' ;
        for (co1 = 1; co1 <= 2*row- 1; co1 ++)
                cout << ' ' ;
        for (co1 = 1; co1 <= 10 - row; co1 ++)
                cout << '*' ;
        cout << endl;
}
```

٦-٧) (أ) اكتب عبارة switch تقوم بما يلي:

إذا كانت قيمة grade هي

'A' : تضيف 4 إلى sum

'B' : تضيف 3 إلى sum

'C' : تضيف 2 إلى sum

'D' : تضيف 1 إلى sum

'F' : تطبع الرسالة "Student is on probation"

(أي الطالب تحت المراقبة والتدقيق)

(ب) عدِّل قطعة البرنامج السابقة في (أ) بحيث تُطبع رسالة خطأ إذا لم تكن gradeمساوية لأي من التقديرات الخمسة المذكورة.

٦-٨) اكتب قطعة برنامج تستمر في قراءة أعداد وجمعها إلى أن تنتهي من قراءة وجمع 10 قيم أو إلى أن تقرأ قيمة سالبة أيهما أقرب، أي أيهما يأتي أولاً. استخدم عروة do .. while.

٦-٩) أعد كتابة قطعة البرنامج التالية مستخدماً عروة do .. while بدلاً من عروة while.

```
cout << "Enter 1, 2, or 3:  ";
cin >> response;
while (response < 1  || response > 3)
{
        cout << "Enter 1, 2,  or 3:  ";
        cin >> response;
}
```

٦-١٠) أعد كتابة قطعة البرنامج التالية مستخدماً عروة while.

```
cin >> ch;
if  (cin)
        do
        {
                cout << ch;
                cin >> ch;
        } while (cin);
```

٦-١١) أعد كتابة قطعة البرنامج التالية مستخدماً عروة for.

```
sum = 0;
count = 1;
while (count <= 1000)
{
        sum = sum + count;
        count ++;
}
```

٦-١٢) أعد كتابة عروة for التالية مستخدماً عروة do .. while.

```
for (m = 93;  m >= 5;  m --)
        cout << m << '  ' << m * m << endl;
```

٦-١٣) أعد كتابة عروة for التالية كعروة while.

```
for (k = 9;  k <= 21;  k++)
        cout << k << '  ' << 3* k << endl;
```

٦-١٤) اكتب دالة تعيد قيمة بحيث تستقبل الدالة وسيطين صحيحين int هما: base, exponent ، وتعيد قيمة base مرفوعة للأس / للقوة exponent . استخدم عروة for في حللك.

٦-١٥) أعد كتابة العروة التالية بحيث تجعل منطق العروة (logic of the loop) أسهل فهماً ، وذلك باستخدام عروة لانهائية (infinite loop) مع عبارات break.

```
sum = 0;
count = 1;
do
{
        cin >> int1;
        if (  !cin  || int1 <= 0)
                cout << "Invalid first integer.";
```

```
        else
        {
                cin >> int2;
                if (  !cin  || int2 > int1)
                        cout << "Invalid second integer.";
                else
                {

                        cin >> int3;
                        if (  !cin  || int3 = = 0)
                                cout << "Invalid third integer.";
                        else
                        {
                                sum = sum + (int1 + int2) / int3;
                                count ++ ;
                        }
                }
        }
} while (cin && int1 > 0 && int2 <= int1 && int3 != 0 &&
          count <= 100);
```

٦-١٦) اذكر صحة أو خطأ T / F كل من العبارات التالية:

أ) العبارتان التاليتان متكافئتان:

```
if (n = = 8)                (ii)
      alpha++;
else if (n = = 3)
       beta++;
else
      gamma++;
```

```
switch (n)                  (i)
{
      case 8  : alpha++;
                 break;
      case 3  : beta++;
                 break;
      default : gamma++;
                 break;
}
```

ب) قيمة alpha **بعد تنفيذ قطعة البرنامج التالية تساوي** 3.

```
    char ch = 'D';
    int  alpha = 3;
```

```
switch (ch)
{
   case 'A' : alpha = alpha + 10;
              break;
   case 'B' : alpha = alpha + 20;
              break;
   case 'C' : alpha = alpha + 30;
}
```

ج) العبارتان التاليتان متكافئتان:

```
cout << "Do you wish to continue? ";                    (i)
cin   >> response;
while (response != 'Y' && response != 'N')
{
   cout << "Do you wish to continue? ";
   cin   >> response;
}
```

```
do                                                      (ii)
{
   cout << "Do you wish to continue? ";
   cin >> response;
} while (response != 'Y' && response != 'N');
```

٦-١٧) ما هي مخرجات كل من قطع البرامج التالية؟

أ) افرض أن القيمة المدخلة تساوي 4.

```
int num;
int alpha = 10;

cin >> num;
switch (num)
{
   case 3  : alpha++;
   case 4  : alpha = alpha + 2;
```

```
   case 8  : alpha = alpha + 3;
   default : alpha = alpha + 4;
}
cout << alpha << endl;
```

ب) افرض أن جميع المتغيرات من النوع int.

```
limit = 8;
cout << 'H';
loopCount = 10;
do
{
   cout << 'E';
   loopCount++;
} while (loopCount <= limit);
cout << "LP";
```

ج) افرض أن جميع المتغيرات من النوع int.

```
limit = 8;
cout << 'H';
for (loopCount = 10; loopCount <= limit; loopCount++)
   cout << 'E';
cout << "LP";
```

٦-١٨) في قطعة البرنامج التالية وُضعت خطأً فاصلة منقوطة في نهاية عنوان عبارة for. ما هي مخرجات القطعة؟

```
cout << 'A';
for (count = 1; count <= 3; count++);
   cout << 'B';
cout << 'C';
```

٦-١٩) ما هي مخرجات كل من قطعتي البرنامج التاليتين بفرض أن جميع المتغيرات من النوع int.

(أ)

```
n = 2;
for (loopCount = 1; loopCount <= 3; loopCount++)
   do
      n = 2 * n;
   while (n <= 4);
cout << n << endl;
```

(ب)

```
n = 2;
for (loopCount = 1; loopCount <= 3; loopCount++)
   while (n <= 4)
      n = 2 * n;
cout << n << endl;
```

٦-٢٠) **أي قيم** first, last **التالية [من أ) إلى هـ)] تجعل عروتي** for, do .. while **التاليتين متكافئتين ؟**

```
// First loop

for (count = first; count <= last; count++)
   DoSomething();

// Second loop

count = first;
do
{
   DoSomething();
   count++;
} while (count <= last);
```

(أ) **جميع القيم.**

(ب) **لا توجد أي قيم.**

(ج) فقط إذا كان first > last

(د) فقط إذا كان first = last

(هـ) فقط إذا كان first <= last

٦-٢١) في برنامج Rainfall program (مثال ٦-١٢):

(أ) أعد كتابة الدالتين GetYesOrNo, GetOneAmount بحيث تستبدل عُرَى while بعُرَى do .. while.

(ب) أعد كتابة الدالة Get12Amounts بحيث تستخدم عروة do .. while بدلاً من عروة for.

(ج) أعد كتابة الدالة Get12Amounts بحيث تستخدم عروة while بدلاً من عروة for.

(د) في الدالة GetYesOrNo هل يمكننا استخدام عبارة switch بدلاً من عبارة if ؟ وإن أمكننا ذلك فهل يُنصح بهذا التبديل ؟

(هـ) في الدالة GetOneAmount هل يمكننا أن نستبدل بعبارة if عبارة switch؟ وإن أمكننا ذلك فهل يُنصح بإجراء هذا التبديل ؟

٦-٢٢) أعد كتابة الدالة Day (مثال ٥-١٢) المعطاة في الفصل السابق بحيث تستخدم عبارة switch بدلاً من عبارة if المتداخلة.

أ) نعم يمكن استخدام عبارة switch هنا (الصيغة التي تُستخدم اختيار default) ولكن لا يُنصح بذلك حيث أنه أسلوب غير جيد للبرمجة.

ب) لا يمكننا استخدام عبارة switch هنا لأن عبارة switch لا يمكنها استخدام المؤثر العلاقي ">" مع الأنواع غير التكاملية (non-integral types)

٦-١٨)

```
int Day( /* in */ int month,          // Month number, 1 - 12
         /* in */ int dayOfMonth,   // Day of month, 1 - 31
         /* in */ int year          ) // Year. For example, 1997
{   .

    .
    // Correct for different length months

    switch (month)
    {
        case 3 : correction = correction - 1;
                 break;
        case 2 :
        case 6 :
        case 7 : correction = correction + 1;
                 break;
        case 8 : correction = correction + 2;
                 break;
        case 9 :
        case 10 : correction = correction + 3; break;
```

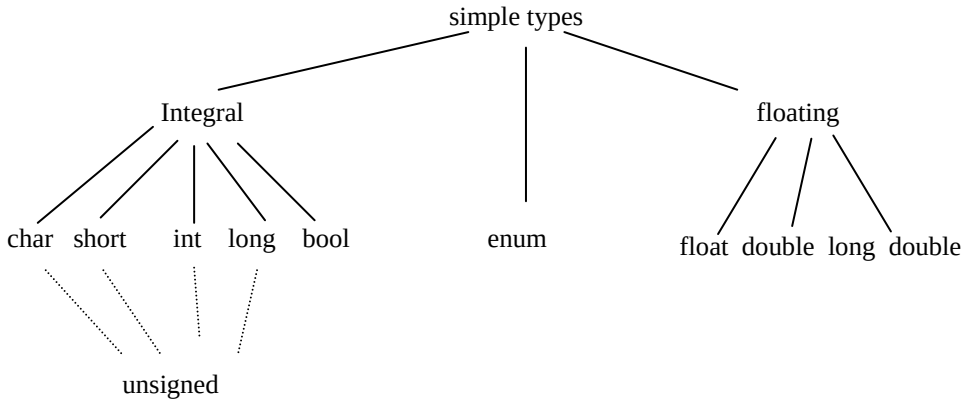
```
        case 11 :
        case 12 : correction = correction + 4;
    }
    return (month - 1) * 30 + correction + dayOfMonth;
}
```

الفصل السابع

الأنواع البسيطة للبيانات
Simple Data Types

(المبنية داخلياً والمعرَّفة بالمستخدِم)
(Built-In and User-Defined)

في دراستنا للبرمجة بلغة C++ كنا نتعامل حتى الآن مع أنواع البيانات: int, char, bool, float. وهذه الأنواع الأربعة مناسبة وكافية لحل العديد من المسائل المتنوعة. ولكن هناك بعض البرامج الخاصة التي تحتاج إلى أنواع أخرى من البيانات. وفي هذا الفصل نلقي نظرة عامة على جميع الأنواع البسيطة للبيانات في لغة C++. وفي بعض الأحيان لا تستطيع حتى أنواع البيانات الداخلية / المبنية داخلياً (built-in data types) نفسها تمثيل (representing) جميع البيانات في البرنامج بصورة مناسبة. ولغة C++ تسمح لنا بعدة آليات (mechanisms) لإنشاء أنواع بيانات معرَّفة بالمستخدم (user-defined data types) ، أي يمكننا أن نعرِّف نحن أنواع بيانات جديدة. وهذا الفصل يقدم واحدة من هذه الآليات: تعريف واستخدام النوع التعددي (enumeration type)، والذي نرمز له بالاسم enum في الشكل التالي (شكل ٧-١) والذي يبين الأنواع البسيطة في لغة C++. وهذا الشكل يعد جزءاً من الشكل الكامل (شكل ٢- ٣) الذي رأيناه في الفصل الثاني ، والذي يبين أنواع البيانات في لغة C++ .



شكل ٧-١

الأنواع البسيطة في لغة C++

مؤثرات إضافية في لغة C++ (Additional C++ Operators)

تحتوي لغة C++ على العديد من المؤثرات التي تسمح لنا بالتعامل مع / معالجة (manipulating) القيم (values) من الأنواع البسيطة . ومن المؤثرات التي مرت معنا سابقاً:

مؤثـــــر الإسنـــاد (=) ، والمـؤثرات الحسابيـــــة (٪ ,/ ,* ,- ,+) ، ومؤثـــــرا الزيادة والنقصان (increment and decrement) (++, --) ، والمـؤثرات العلاقية (==, !=, <, <=, >, >=) ، والمؤثرات المنطقية (!, &&, | |) . وفي بعض الحالات يعتبر القوسان أيضاً مؤثراً من المؤثرات، ونعنى به مؤثر استدعاء الدالة (function call operator)، مثل

```
ComputeSum (x, y);
```

ومؤثر صياغة / تحويل النوع (type cast operator) ، مثل

```
y = float (someInt);
```

وتحتوي لغة C++ أيضاً على مؤثرات خاصة عديدة نادراً ما توجد في لغات البرمجة الأخرى . وفيما يلي جدول ببعض هذه المؤثرات الإضافية ومعانيها.

Combined assignment operators	**مؤثرات الإسناد المشتركة**
+ = Add and assign	**الجمع والإسناد**
- = Subtract and assign	**الطرح والإسناد**
* = Multiply and assign	**الضرب والإسناد**
/ = Divide and assign	**القسمة والإسناد**
Increment and decrement operators	**مؤثرات الزيادة والنقصان**
++ Pre-increment	**الزيادة السابقة** (مثلاً ++ someVar)
++ Post-increment	**الزيادة اللاحقة** (مثلاً someVar++)
-- Pre-decrement	**النقصان السابق** (مثلاً --someVar)
-- Post-decrement	**النقصان اللاحق** (مثلاً someVar --)

Bitwise operators (Integer operands only)	**مؤثرات الوحدات الثنائية (للمؤثرات الصحيحة فقط)**
<< Left shift	**الإزاحة لليسار**
>> Right shift	**الإزاحة لليمين**
& Bitwise AND	AND **للوحدات الثنائية**
| Bitwise OR	OR **للوحدات الثنائية**
^ Bitwise Exclusive OR	Exclusive OR **للوحدات الثنائية**
~ Complement (invert all bits)	**المكمِّل (لعكس جميع الوحدات الثنائية)**

Bitwise operators (Integer operands only)	**مؤثرات إسناد مشتركة إضافية (للمؤثرات الصحيحة فقط)**
% = Modulus and assign	**الباقي بعد القسمة ثم الإسناد**
<<= Shift left and assign	**الإزاحة لليسار والإسناد**
>>= Shift right and assign	**الإزاحة لليمين والإسناد**
&= Bitwise AND and assign	AND **للوحدات الثنائية والإسناد**
| = Bitwise OR and assign	OR **للوحدات الثنائية والإسناد**
^ = Bitwise Exclusive OR and assign	Exclusive OR **للوحدات الثنائية والإسناد**

Other operators	**مؤثرات أخرى**
( ) Cast	**التحويل / الصياغة / الصب**
sizeof Size of operand in bytes	**السعة / الحجم** [Form: sizeof Expr or sizeof (Type)]
?: Conditional operator	**المؤثر الشرطي** [Form: Expr1 ? Expr2 : Expr3]

Table of Operators جدول المؤثرات

مؤثرات الإسناد وتعابير الإسناد
Assignment Operators and Assignment Expressions

تحتوي لغة C++ على عدة مؤثرات إسناد. وتعد علامة التساوي (=) مؤثر الإسناد الأساسي (basic assignment operator) . وهذا المؤثر مع معامَليه (operands) يكوِّن ما يعرف بتعبير الإسناد (assignment expression) [وليس عبارة إسناد (assignment statement)]. وكل تعبير إسناد له قيمة (value) وأثر جانبي (side effect) وهو تخزين هذه القيمة في الهدف الذي يرمز إليه الطرف الأيسر . فمثلاً التعبير

```
delta = 2 * 12
```

له القيمة 24 والأثر الجانبي الذي هو تخزين هذه القيمة في delta . وفي لغة C++ يصبح أي تعبير (expression) عبارة تعبير (expression statement) عندما يُنهي بفاصلة منقوطة. فمثلاً فيما يلي ثلاث عبارات صحيحة (valid) في لغة C++ ، رغم أن أول عبارتين ليس لهما أي تأثير وقت التشغيل (at run time) .

```
23;
2 * (alpha + beta) ;
delta = 2 * 12;
```

والعبارة الثالثة مفيدة بسبب أثرها الجانبي وهو تخزين القيمة 24 delta .

ونظراً لأن الإسناد (assignment) هو تعبير (expression) وليس عبارة (statement) ، فيمكن استخدامه في أي موضع يُسمح فيه بالتعبير. فمثلاً العبارة التالية تقوم بتخزين القيمة 20 في firstInt والقيمة 30 في secondInt والقيمة 35 في thirdInt :

```
thirdInt = (secondInt = (firstInt = 20) + 10) + 5;
```

وقد حذرنا سابقاً في الفصل الثالث من خطأ استخدام المؤثر = بدلاً من المؤثر = = ، مثل :

```
if (alpha = 12)          // Wrong
   ⋮
else
   ⋮
```

فالشرط في عبارة If تعبير إسناد (assignment expression) وليس تعبيراً علاقياً (relational expression) . وقيمة التعبير هي 12 وتفسَّر في شرط If على أنها true ، وبالتالي فإن عبارة else لن تنفذ أبداً. والأسوأ من هذا هو التأثير الجانبي لتعبير الإسناد وهو تخزين القيمة 12 في alpha ما حيا أي محتويات سابقة.

وبالإضافة إلى المؤثر = فإن لغة C++ تحتوي على عدة مؤثرات إسناد مشتركة مثل المؤثرات ... , = * , =+ الموجودة في جدول المؤثرات السابق، ومعاني هذه المؤثرات المشتركة تتضح من المثالين التاليين:

العبارة	العبارة المكافئة
Statement	**Equivalent Statement**
i + = 5;	i = i + 5;
p *= n + 3;	p = p * (n + 3);

مؤثرات الزيادة والنقصان
Increment and Decrement Operators

تؤثر مؤثرات الزيادة والنقصان (-- , ++) على المتغيرات فقط، وليس على الثوابت أو التعابير الاختيارية. ولتوضيح الفارق بين الزيادة السابقة والزيادة اللاحقة: نفرض أن المتغير someInt يحتوي على القيمة 3. التعبير someInt ++ يعنى الزيادة السابقة ، أي أن التأثير الجاني (side effect) لزيادة someInt يحدث أولاً ، وبالتالي فإن القيمة الناتجة للتعبير هي 4 . وفي مقابل ذلك فإن التعبير someInt ++ يعني الزيادة اللاحقة ، حيث تكون قيمة التعبير مساوية 3 ، ثم يحدث التأثير الجانبي لزيادة someInt. وقطعة البرنامج التالية توضح الفارق بين الزيادة السابقة والزيادة اللاحقة:

```
int1 = 14;
int2 = ++int1;
// Assert: int1 == 15  &&  int2  == 15

int1 = 14;
```

```
int2 = int1++;
// Assert: int1 == 15  &&  int2  == 14
```

ولا يستحب استخدام الآثار الجانبية وسط تعابير أكبر حيث أنها تجعل التعابير عرضة للأخطاء المعنوية (semantic errors) ، وكذلك للالتباس (confusing) في القراءة والفهم. انظر مثلاً لهذا المثال:

```
a = (b = c++) * --d / (e += f++);
```

وأكثر الاستخدامات الشائعة للمؤثرين ++, -- هي للزيادة أو النقصان في عبارة تعبير مستقلة مثل:

```
count ++;
```

حيث لا نستخدم هنا قيمة التعبير ، ولكن نحصل على النأثير الجانبي المطلوب لزيادة count. وفي هذا المثال يستوي تطبيق الزيادة السابقة أو الزيادة اللاحقة.

مؤثرات الوحدات الثنائية Bitwise Operators

تستخدم مؤثرات الوحدات الثنائية (... ,| , & , >> ,<<) الموجودة بجدول المؤثرات السابق لمعالجة (manipulating) الوحدات الثنائية (البتات) المفردة (individual bits) الموجودة في خلية بالذاكرة (within a memory cell). ولا نتعرض في هذا الكتاب لاستخدامات هذه المؤثرات فهي غالباً ما تدرس في مقررات تنظيم الحاسوب (computer organization) والبرمجة بلغة التجميع (assembly language programming) ، ولكننا نشير هنا فقط إلى ملاحظتين بخصوص هذه المؤثرات:

ملاحظة (١): المؤثران الداخليان << , >> (built-in operators) هما مؤثرا الإزاحة لليمين والإزاحة لليسار على الترتيب، والغرض منهما إزاحة الوحدات الثنائية (البتات) الموجودة في خلية بالذاكرة لليمين أو لليسار. وقد كنا نستخدم هذين المؤثرين طوال الفصول السابقة لغرضين مختلفين تماماً ، وهما: الإدخال والإخراج في البرنامج (program input and output).ويستخدم ملف المقدمة (header file) iostream طريقة متقدمة في لغة C++ (advanced technique) تُدعى "التحميل الزائد للمؤثرات" (operator overloading) لإعطاء معاني

إضافية لهذين المؤثرين. والمؤثر المُحمَّل بحمل زائد (overloaded operator) هو ذلك المؤثر الذي له أكثر من معنى (multiple meanings) ويتحدد المعنى بناء على أنواع بيانات المعامَلات (data types of operands) فبالنسبة مثلاً للمؤثر << يحدِّد المترجم من السياق (by context) المقصود من هذا المؤثر: هل هو عملية إزاحة لليسار (left shift operation) أم عملية إخراج (output operation). فإن كان المعامَل الذي على يسار المؤثر << هو سيل مخرجات (output stream) فالعملية المطلوبة هي عملية إخراج، وإن كان هذا المعامَل هو متغير صحيح (integer variable) فالعملية هي عملية إزاحة لليسار.

ملاحظة (٢) : نكرر ما حذرنا منه سابقاً في الفصل الثالث وهو أهمية عدم الالتباس بين المؤثرين || , && والمؤثرين | ,& . فمثلاً العبارة

```
if (i == 3 & j == 4)     // Wrong
   k = 20;
```

تُعَدُّ صحيحة من الناحية التركيبية (syntactically correct) لأن & مؤثر صحيح (valid operator) وهو مؤثر AND للوحدات الثنائية (bitwise AND operator). والبرنامج الذي يحتوي على هذه العبارة ستكون ترجمته صحيحة (compiles correctly) وتنفيذه غير صحيح (executes incorrectly).

عملية الصب / التحويل The Cast Operation

رأينا سابقاً أنه يجوز في لغة C++ الخلط بين أنواع البيانات (mixing data types) في التعابير، وفي عمليات الإسناد ، وفي تمرير الوسطاء (argument passing) ، وفي إعادة قيمة دالة. ويحدث عندئذ تحويل ضمني للنوع. وبدلاً من الاعتماد على هذا التحويل الضمني في عبارة مثل

```
intVar = floatVar ;
```

أشرنا بأفضلية استخدام التحويل الصريح للنوع (explicit type cast) لبيان أن تحويل النوع مقصود فعلاً:

```
intVar = int (floatVar) ;
```

وفي لغة C++ يمكن أن تأخذ عملية التحويل إحدى الصيغتين التاليتين:

```
intVar = int (floatVar) ;          // Functional notation
intVar = (int) floatVar ;          // Prefix notation. Parentheses required
```

الصيغة الأولى يطلق عليها: "الاصطلاح الدالي" (functional notation) لأنها تشبه استدعاء دالة، ولكنها في الحقيقة ليست استدعاء دالة، لأنه لا يوجد برنامج فرعي (subprogram) معرف بالمستخدم (user-defined) أو معرف سابقاً (predefined) يُدْعَى int. والصيغة الأخرى يطلق عليها "الاصطلاح السابق" (prefix notation) وفيها نضع قوسين حول اسم نوع البيانات وليس حول التعبير الذي نحوِّله. [ملاحظة: هذا الاصطلاح هو الصيغة الوحيدة الموجودة في لغة C، وقد أضافت لغة C++ الاصطلاح الدالي]. وغالباً ما نستخدم الاصطلاح الدالي، ولكن إذا تكون اسم نوع البيانات من أكثر من اسم تعريفي واحد (single identifier) وجب أن نستخدم الاصطلاح السابق، مثل:

```
myVar = (unsigned int) someFloat;          // Yes
myVar = unsigned int (someFloat);          // No
```

المؤثر sizeof (The sizeof operator)

المؤثر sizeof هو مؤثر / معامل أحادي (unary operator) يعطي حجم / سعة (size) معامَله (its operand) بالبايت (in bytes) (وهي وحدة العناصر الثنائية). والمعامَل قد يكون اسم متغير مثل:

```
sizeof someInt
```

أو اسم نوع بيانات بين قوسين مثل:

```
sizeof (float)
```

ويمكننا معرفة أحجام أنواع البيانات المختلفة في الجهاز الذي نستخدمه بكتابة قطعة برنامج كالتالية:

```
cout << "Size of a short is "  << sizeof (short) << endl;
```

```
cout << "Size of an int is "  << sizeof (int) << endl;
cout << "Size of a long is "  << sizeof (long) << endl;
```

المؤثر الشَّرطي :? (The conditional operator ? :)

هذا المؤثر مؤثر ثلاثي (ternary operator) ، أي له ثلاثة معامَلات (3 operands) ، وصيغته العامة:

Expression1 ? Expression2 : Expression3

حيث يوجد الحاسوب أولاً قيمة Expression1 . فإن كانت true فإن قيمة التعبير الكلي (entire expression) هي Expression2، وإن كانت false فإن قيمة التعبير الكلي هي Expression3 . [تعبير واحد فقط من Expression1, Expression2 توجد قيمته].

مثال ٧-١: ضع في المتغير max القيمة الكبرى من قيمتي المتغيرين a, b

(أ) باستخدام عبارة If (ب) باستخدام المؤثر الشرطي :?

الحل:

(أ)

```
if (a > b)
      max = a;
else
      max = b;
```

(ب) عبارة الإسناد التالية تستخدم المؤثر :?

```
max = (a > b) ? a : b;
```

مثال ٧-٢: القيمة المطلقة (absolute value) لعدد x تعرف كما يلي:

$$|x| = \begin{cases} x, \ if \ x \geq 0 \\ -x, \ if \ x < 0 \end{cases}$$

اكتب عبارة إسناد تستخدم المؤثر الشرطي :? لإيجاد القيمة المطلقة لمتغير x وتخزينها في المتغير y

الحل:

```
y = (x >= 0) ? x : -x;
```

ملاحظة:في حلي المثالين السابقين باستخدام المؤثر الشرطي :? وضعنا قوسين حول الشرط الذي نختبره. وهذان القوسان لزيادة الإيضاح فقط وليسا ضروريين، حيث أن المؤثر :? له أولوية منخفضة بناء على قاعدة الأولويات [انظر ملحق (ب)].

التعامل مع البيانات الرمزية Working with Character Data

حتى الآن كنا نستخدم المتغيرات الرمزية لتخزين بيانات رمزية مثل 'A', 'e', '+' :

```
char someChar;
        ⋮
someChar = 'A';
```

ونظراً لأن char معرف بانه نوع تكاملي (integral type) وأن sizeof (char) يساوي 1، فيمكننا أيضاً استخدام متغير من النوع char لتخزين ثابت صحيح صغير (small integer constant) [عادة وحدة عناصر ثنائية واحدة (بايت واحد) one-byte] ، مثل :

```
char counter;
        ⋮
counter = 3;
```

ويستخدم كل حاسوب مجموعة رموز (character set) خاصة، ومن أشهر هذه المجموعات – كما ذكرنا سابقاً– مجموعتا ASCII, EBCDIC (انظر الملحق ج). وتتكون مجموعة ASCII من ١٢٨ رمز مختلف، بينما تتكون مجموعة EBCDIC من ٢٥٦ رمز . ويقصد بالتمثيل الخارجي **(external representation)** لأي رمز الصورة التي يظهر بها على وسائل الإدخال / الإخراج (I/O devices) كالطابعة (printer) مثلاً. وأما التمثيل الداخلي **(internal representation)** له

فيقصد به الطريقة / الصورة التي يتم تخزينه بها داخل وحدة ذاكرة الحاسوب (computer's memory unit). فمثلاً الثابت الرمزي 'A' (char constant) في لغة C++ تمثيله الخارجي هو الحرف A وتمثيله الداخلي هو القيمة الصحيحة 65 في مجموعة ASCII [التمثيل الداخلي لرموز هذه المجموعة والبالغ عددها 128 رمزاً يتراوح من القيمة 0 إلى القيمة 127 . مثلاً التمثيل الداخلي للرمز 'b' هو 98 . بينما تتراوح قيم التمثيل الداخلي لرموز المجموعة EBCDIC من 0 إلى 255].

مثال ٧-٣: ما هي مخرجات كل من القطعتين التاليتين:

(أ)

```
char ch = 'D';

ch ++;
cout << ch;
```

(ب)

```
char ch;

for ( ch = 'A'; ch <= 'G'; ch++)
        cout << ch;
```

الحل:

(أ) E

(ب) ABCDEFG

في القطعة (أ) تقوم العبارة الأولى بالإعلان عن ch (أنه من النوع char) وتعطيه القيمة الصحيحة الابتدائية 68 (initial integer value) [بفرض أن مجموعة الرموز هي مجموعة ASCII]. ثم تقوم العبارة الثانية بزيادة قيمة ch إلى 69 . وأخيراً يُطبع تمثيله الخارجي (الحرف E) بالعبارة الثالثة.

القطعة (ب) تقوم بتطبيق فكرة زيادة قيمة متغير رمزي عدة مرات ، حيث تبدأ بإعطاء المتغير ch القيمة الابتدائية 'A' (القيمة 65 في شفرة ASCII) ، ثم في

كل دورة من دورات عروة for تطبع التمثيل الخارجي للمتغير ch . وفي آخر تكرير (iteration) في العروة تطبع الحرف G، وتزيد قيمة ch إلى 'H' (القيمة 72 في شفرة ASCII) . بعد ذلك يصبح شرط العروة خاطئاً false ، وتنتهي العروة.

ملاحظة ١: إذا كان الحاسوب يستخدم مجموعة الرموز ASCII ، فإن العبارتين:

```
ch = 'D';
ch = 68;
```

يكون لهما التأثير نفسه، وهو تخزين (storing) 68 في ch . ولكن العبارة الثانية لا يُنصح بها فالأولى أوضح في فهمها، وكذلك الثانية لا نستطيع نقلها (nonportable) واستخدامها في أي جهاز نشاء ، فمثلاً لا تكون صحيحة إذا كانت الشفرة المستخدمة هي شفرة EBCDIC، لأن هذه الشفرة تستخدم تمثيلاً داخلياً مختلفاً (القيمة 196) للرمز 'D' .

ملاحظة ٢: إذا كان لدينا عبارة الإسناد

```
ch = 68;
```

وتلتها بعد ذلك عبارة الإخراج

```
cout << ch << endl;
```

فإن نتيجة تنفيذ هذه العبارة الأخيرة تعتمد على نوع type المتغير ch. فإن كان ch من النوع int، فالنتيجة هي طباعة القيمة الصحيحة 68 وإن كان ch من النوع char، فالنتيجة هي طباعة الحرف D (بفرض استخدام شفرة ASCII) .

تحويل الرموز الرقمية إلى أعداد صحيحة
Converting Digit Characters to Integers

نفرض أننا نود تحويل رقم digit يُقرأ بالصيغة الرمزية (character form) إلى مكافئة العددى (numeric equivalent) . نظراً لأن الرموز الرقمية من '0' إلى '9' متعاقبة (consecutive) في كل من مجموعتي الرموز ASCII, EBCDIC ، فإن طرح '0' من أي رقم في الصيغة الرمزية يعطي الرقم في الصيغة العددية (numeric form):

'0' - '0' = 0
'1' - '0' = 1
'2' - '0' = 2
⋮

فمثلاً في شفرة ASCII : التمثيل الداخلي للرمز '0' هو 48 ، والتمثيل الداخلي للرمز '2' هو 50، ولذلك فالتعبير

'2' - '0'

يساوى 48 - 50 أي يساوي 2 .

مثال ٧-٤: اكتب داله خاوية تظل تحث المستخدم إلى أن يدخل رقماً من 1 إلى5 ، مع إعطاء رسالة خطأ في حالة إدخال بيانات غير صحيحة ، وبحيث تقرأ الدالة البيانات المدخلة بالصيغة الرمزية، وتقوم بتحويل الرمز الرقمي إلى العدد الصحيح المكافئ.

الحل:

```
# include < cctype>              // For isdigit ( )
using namespace std;
   ⋮
void GetResponse ( / * out * / int& response )
// Postcondition:
//        User has been prompted to enter a digit from 1
//        through 5 (repeatedly, and with error messages,
//        if data is invalid)
//    && 1 <= response <= 5

{
        char inChar;
        bool badData = false;

        do
        {
                cout << "Enter a number from 1 through 5: ";
                cin >> inchar;
                if (  ! isdigit (inChar)  )
                        badData = true;          // It's not a digit
```

```
            else
            {
                (response = int (inChar - '0');
                if (response < 1  || response > 5)
                    badData = true;     // It's not a digit but
            }                           // it's out of range
            if (badData)
                cout << "Please try again. " << endl;
        } while (badData);
)
```

التحويل إلى حرف صغير أو إلى حرف كبير

Converting to Lowercase or Uppercase

أحياناً نحتاج إلى تحويل حرف صغير (lowercase letter) **إلى حرف كبير** (uppercase) **، أو العكس. وعملية التحويل هذه سهلة للغاية حيث أنها تتم باستدعاء مباشر لدالة مكتبية (إما** toupper **أو** tolower **، على الترتيب) ، وذلك من خلال ملف المقدمة** cctype (header file) **. وكل من هاتين الدالتين** (tolower , toupper) **دالة تعيد قيمة** (value-returning function) **، والجدول التالي يعطي أوصافهما.**

ملف المقدمة Header File	الدالة Function	نوع الدالة Function Type	قيمة الدالة Function Value
<cctype>	toupper (ch)	char	الحرف الكبير المقابل إذا كان ch حرفاً صغيراً، والرمز ch نفسه ما عدا ذلك
<cctype>	tolower (ch)	char	الحرف الصغير المقابل إذا كان ch حرفاً كبيراً، والرمز ch نفسه ما عدا ذلك

فمثلاً toupper ('m') **تعيد الرمز** 'M'

tolower ('M') **تعيد الرمز** 'm'

tolower ('+') **تعيد الرمز** '+'

لاحظ أن كلا من الدالتين تعيد الرمز الأصلي نفسه (original character)إن لم يتحقق شرط "إذا كان ... " المقابل

ومن الاستخدامات الشائعة لهاتين الدالتين السماح للمستخدم بإدخال إجابته (response) لبعض الأسئلة / الطلبات (prompts) عن طريق طباعة إما حروف كبير أو حروف صغيرة.

مثال ٧-٥: اكتب قطعة برنامج تطلب فيها من المستخدم أن يجيب على سؤال ما إما بالإيجاب [بإدخال حرف Y ويعني "Yes"] أو بالنفي [بإدخال حرف N ويعني "No"] . والقطعة تقبل من المستخدم طباعة حرف كبير أو حرف صغير [أي Y أو y للإيجاب ، وN أو n للنفي]. أما إذا أدخل المستخدم أي رمز خلاف هذه الرموز الأربعة (Y, y, N, n) فإن القطعة تستدعى الدالة الخاوية PrintErrorMsg لطباعة رسالة خطأ.

الحل:

```
# include <cctype>              // For toupper ( )
using namespace std;
   ⋮
cout << "Enter Y or N: ";
cin >> inputChar;
{
   ⋮
}
else if (toupper (onputChar) == 'N')
{
   ⋮
}
else
    PrintErrorMsg ( );
```

مثال ٧-٦: اكتب دالة رمزية القيمة Lower (أي دالة تعيد قيمة رمزية) تؤدي عمل الدالة المكتبية tolower ، أي تعيد الحرف الصغير المقابل لحرف كبير معطَى ، أو تعيد الرمز المعطَى نفسه إن لم يكن هذا الرمز حرفاً صغيراً.

ملاحظة: عملياً لا نحتاج لكتابة هذه الدالة المطلوبة Lower، لأنه يمكننا أن نستخدم مباشرة الدالة المكتبية tolower ، ولكن السؤال في هذا المثال للتدريب فقط.

الحل: **في شفرة ASCII :** **أي حرف صغير يأتي ترتيب موضعه بعد ٣٢ موضع بالضبط من موضع الحرف الكبير المقابل. مثلاً : التمثيل الداخلي للرمز 'a' هو 97 ، بينما التمثيل الداخلي للرمز 'A' هو 65 . (97 - 65 = 32)**

في شفرة EBCDIC: **أي حرف صغير يأتي ٦٤ موضعاً قبل الحرف الكبير المقابل . مثلاً: شفرة الرمز 'a' هي 129 ، وشفرة الرمز 'A' هي 193 .**
(193 - 129 = 64)

وكي تعمل الدالة بصورة صحيحة على جميع الأجهزة – سواء التي تستخدم شفرة ASCII أو شفرة EBCDIC – نعرِّف ثابتاً صحيحاً DISTANCE قيمته هي الفارق 'A' - 'a' . ففي شفرة ASCII تكون قيمة هذا التعبير 32، وفي شفرة EBCDIC قيمته 64-.

```
# include < cctype>              // For isupper ( )
using namespace std;
   :
char Lower  (   / * in * / char ch  )

// Postcondition:
//        Function value == lowercase equivalent of ch, if ch is
```

```
//                          an uppercase letter
//                      ==  ch, otherwise

{

        const int DISTANCE = 'a' - 'A' ;    // Fixed distance between

                                             // uppercase and lowercase
                                            // letters
        if (isupper (ch) )
                return ch + DISTANCE;
        else
                return ch;
}
```

الوصول إلى رموز سلسلة الرموز
Accessing Characters within a String

تسمح لنا طبقة (class) string بالوصول إلى أي رمز من رموز سلسلة رموز عن طريق ذكر اسم السلسلة الهدف ثم بين قوسين مربعين نضع عدداً يبين موضع (position number) الرمز المطلوب:

```
StringObject [ Position ]
```

حيث أول رمز موضعه 0 (position) ، والثاني موضعه 1 ، والثالث موضعه 2 ، ... وهكذا . أي أن قيمة الموضع تتراوح بين 0 و (L - 1) حيث L : طول السلسلة (string length) . فإذا فرضنا مثلاً أن inputStr هدف سلسلة (string object) و ch متغير من النوع char ، فإن العبارة

```
ch = inputStr [2];
```

تصل إلى الرمز الثالث (الرمز الذي موضعه 2) في السلسلة ، وتنسخه في ch .

مثال ٧-٧: اكتب قطعة برنامج تطلب من المستخدم إدخال إجابته على أحد الأسئلة إما بالإثبات (بإدخال كلمة Yes) أو بالنفي (بإدخال كلمةNo) ، علماً بأنه يُسمح للمستخدم بطباعة حروف كبيرة أو صغيرة في أي من حروف الكلمتين. ثم

تقرأ القطعة إجابة المستخدم، وبناء على الحرف الأول من إجابته [إما أحد الحرفين Y, y أو أحد الحرفين N, n أو أي رمز آخر خاطئ] تتخذ القرار المناسب.

الحل:

```
string inputStr;
    ⋮
cout << "Enter Yes or No:  ";
cin >> inputStr;
if (toupper (inputStr [0]) == 'Y')

{
    ⋮
}
else if (toupper (inputStr [0]) == 'N')
{
    ⋮
}
else
        PrintErrorMsg ( );
```

الأنواع البسيطة المعرَّفة بالمستخدِم
User - Defined Simple Types

نناقش فيما يلي كيفية إنشاء (creating) أنواع بسيطة نقوم نحن بتعريفها.

عبارة Typedef

تسمح هذه العبارة بإعطاء اسم جديد لنوع موجود له اسمه. والصيغة العامة لهذه العبارة:

```
typedef   ExistingTypeName   NewTypeName;
```

فمثلاً قبل أن يكون نوع البيانات bool جزءاً من لغة C++ استخدم المبرمجون قطعة برنامج كالتالية لمحاكاة (simulating) نوع منطقي (a Boolean type) :

```
typedef int Boolean;
const int TRUE = 1;
```

```
const int FALSE = 0;
    ⋮
Boolean dataOK;
    ⋮
dataOK = TRUE;
```

فعبارة Typedef لتجعل المترجم (compiler) يضع كلمة int مكان كل كلمة Boolean في بقية البرنامج.

واستخدام عبارة Typedef يُعدُّ طريقة محدودة جداً لتعريف أنواع بيانات. وفي الحقيقة فإن هذه العبارة لا تنشئ نوعاً جديداً للبيانات، وإنما هي تنشئ اسماً إضافياً لنوع بيانات موجود فعلاً. وفي المثال المذكور سابقاً فبالنسبة للمترجم يعد كل من مجال (domain) وعمليات (operations) النوع Boolean مطابقاً (identical) لمجال وعمليات النوع int.

ورغم أن عبارة Typedef لا تستطيع فعلاً إنشاء نوع بيانات جديد، إلا أن لها فائدة ذات قيمة في كتابة برامج واضحة العبارات. فقبل أن يكون bool نوعاً مبنياً داخلياً، فإن البرنامج الذي استخدم الأسماء التعريفية Boolean, TRUE, FALSE كان أوضح وأكثر مطابقة للمعاني من البرنامج الذي استخدم int, 1, 0 للعمليات المنطقية (Boolean operations) .

وتخضع أسماء الأنواع المعرفة بالمستخدم لقواعد المجال (scope rules) نفسها التي تنطبق على الأسماء التعريفية (identifiers) عموماً. ومعظم الأنواع – كالنوع السابق Boolean – تعرَّف تعريفاً شاملاً (globally) ، وإن كان من المقبول أيضاً أن نعرِّف نوعاً جديداً داخل برنامج فرعي (within a subprogram) إن كان هذا هو المكان الوحيد الذي سيستخدم فيه. والخطوط العامة التي تحدد موضع تعريف أي ثابت مسمَّى (a named constant) تنطبق أيضاً على أنواع البيانات.

الأنواع التعددية Enumeration Types

تسمح لغة C++ للمستخدم بتعريف نوع بسيط جديد عن طريق ذكر قائمة / تعديد (listing / enumerating) القيم الحرفية (literal values) التي تكوِّن مجال (domain) هذا النوع. وهذه القيم الحرفية يجب أن تكون أسماء تعريفية (identifiers) وليس أعداداً (numbers) . ويُفصل بين الأسماء التعريفية بفواصل، وتحاط القائمة بقوسين { } . وأنواع البيانات التي تعرَّف بهذه الطريقة يطلق عليها الأنواع التعددية (enumeration types) ، ومن أمثلتها:

```
enum Caliphs {AbuBakr, Omar, Othman, Ali };
enum SacredMonths { Rajab, ThulQe'da, ThulHijja, AlMoharram};
enum Days {FRI, SAT, SUN, MON, TUE, WED, THU};
```

فالإعلان (declaration) الأخير مثلاً ينشئ نوع بيانات جديداً اسمه Days. وفي حين أن عبارة Typedef تنشئ مجرد مرادف (synonym) لنوع موجود ، فإن النوع التعددي – مثل Days – هو نوع جديد فعلاً ومختلف عن أي نوع موجود.

والقيم الموجودة في النوع Days [FRI, SAT, SUN, …] يطلق عليها : "القيم التعددية" (enumerators) ، أي أن القيم التعددية هي القيم الموجودة في مجال (domain) النوع التعددي. وهذه القيم تكون مرتبة (ordered) بمعنى أن:

FRI < SAT < SUN ….. < WED < THU.

فتطبيق المؤثرات العلاقية على القيم التعددية يماثل تطبيقها على الرموز، حيث العلاقة التي تُختبر هي : ما إذا كانت قيمة تعددية تأتي قبل (comes before) أو تأتي بعد (comes after) قيمة أخرى في ترتيب (ordering) نوع البيانات.

فالنوع التعددي **(Enumeration type)** : هو نوع بيانات مُعَرَّف بالمستخدم مجاله (domain) مجموعة مرتبة (ordered set) من قيم حرفية (literal values) نعبِّر عنها بأسماء تعريفية (expressed as identifiers).

وكما أن الثوابت الرمزية (char constants) تمثَّل داخلياً بأعداد صحيحة غير سالبة (مثلاً في شفرة ASCII من 0 إلى 127) ، فكذلك يتم تمثيل قيم النوع

التعددي داخلياً بأعداد صحيحة . ويُفترض (by default) أن أول قيمة تعددية (first enumerator) قيمتها الصحيحة 0 (integer value) ، والثانية قيمتها 1 ، والثالثة قيمتها 2 ، ... وهكذا . فمثلاً الإعلان السابق عن النوع التعددي Days يماثل مجموعة الإعلانات التالية:

```
typedef int Days;
const int FRI  = 0;
const int SAT = 1;
const int SUN = 2;
   :
const int THU = 6;
```

وإن كان هناك أي مبرر لتغيير التمثيل الداخلي للقيم التعددية – ونادراً ما يوجد – فيمكننا ذلك بتحديد قيم التمثيل الداخلي التي نود إعطاءها صراحة كما يلي مثلاً

```
enum Days { FRI = 12,   SAT = 2,    SUN = 5, ……};
```

ولكن عادة لا نقوم بذلك حيث أن الذي يهمنا مثلاً في هذا النوع Days هو أيام الأسبوع وليس كيفية تخزين الآلة لهذه الأيام داخلياً.

ويمكن للقيمة التعددية (enumerator) – في تعريف النوع التعددي – أن تكون إما ثابتاً حرفياً أو ثابتاً مسمًّى (literal or named constant) . والأسماء التعريفية المستخدمة كقيم تعددية يجب أن تتبع قواعد الأسماء التعريفية في لغة C++ . فمثلاً الإعلان التالي عن النوع التعددي Vowel

```
enum Vowel ( 'A', 'E', 'I', 'O', 'U' );              // Error
```

خاطئ / غير سليم (not legal) وذلك لأن القيم التعددية / العناصر (items) ليست أسماء تعريفية ، وكذلك الإعلان

```
enum Places (1st, 2nd , 3rd ) ;          // Error
```

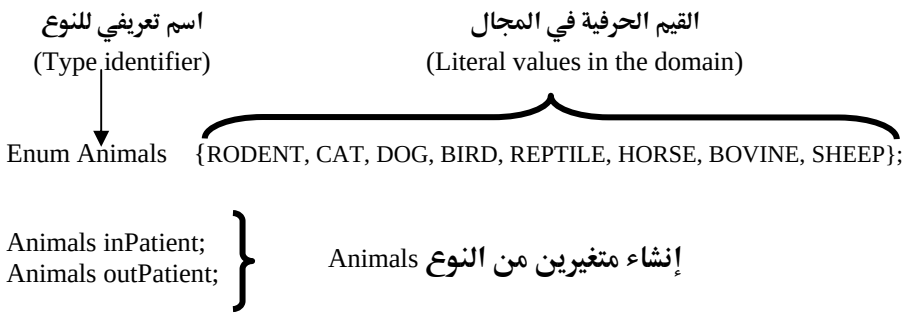
خاطئ لأن الأسماء التعريفية لا يمكن أن تبدأ بأرقام. وبالنسبة للإعلانين التاليين

```
enum Starch (CORN, RICE, POTATO, BEAN ) ;
enum Grain (WHEAT, CORN, RYE, BARLEY, SORGHUM);    // Error
```

فكل من النوعين Starch, Grain يعد صحيحاً بمفرده، ولكنهما معاً غير صحيحين، وذلك لأن الأسماء التعريفية في المجال نفسه (same scope) يجب أن تكون وحيدة (unique) ، بينما نجد أن CORN مُعرَّفة مرتين، وهذا غير جائز .

العمليات على المتغيرات من الأنواع التعددية
Operations on variables of enumeration types

نفرض أننا نـــود كتابة برنامج خاص بعلاج الحيوانات في عيادة بيطرية (a veterinary clinic) . النوع التعددي التالي Animals يشمل أنواع الحيوانات التي يمكن علاجها في تلك العيادة }الغنم ، البقر ، الخيل ، الزواحف ، الطيور ، الكلاب ، القطط ، القوارض{:



فمثلاً SHEEP تعد قيمة حرفية وليست اسم متغير (a variable name)، بل هي إحدى القيم (values) التي يمكن تخزينها في أي من المتغيرين inPatient (الذي يمكن استخدامه لمرضى العيادة الداخلية) أو outpatient (لمرضى العيادة الخارجية) . وفيما يلي نلقي نظرة على أنواع العمليات التي يمكن إجراؤها على المتغيرات من الأنواع التعددية.

(١) الإسناد (Assignment)

عبارة الإسناد

```
inPatient = HORSE;
```

لا تسند للمتغير inPatient **سلسلة الرموز** "HORSE" **، و لا محتويات متغير اسمه** HORSE **، وإنما تسند له القيمة** HORSE **التي هي إحدى القيم في مجال / نطاق نوع البيانات** Animals .

والإسناد عملية صحيحة طالما أن القيمة التي نسندها هي من النوع Animals . **فكلا العبارتين**

```
InPatient = HORSE;
OutPatient = inPatient;
```

مقبولتان – أي صحيحتان – حيث أن التعبير الموجود في الطرف الأيمن في أي من العبارتين هو من النوع Animals **، فالقيمة** HORSE **قيمة حرفية من النوع** Animals**، والمتغير** inPatient **متغير من النوع** Animals .

ورغم أننا نعلم أن التمثيل الداخلي للقيمة HORSE **هو العدد الصحيح** 5 **إلا أن المترجم** (compiler) **لا يسمح بعبارة الإسناد**

```
inPatient = 5;          // Not allowed
```

والقاعدة المتبعة في ذلك هي:

قاعدة التحويل الضمني للنوع

يُعرَّف التحويل الضمني للنوع (implicit type coercion) **من نوع تعددي إلى نوع تكاملي** (integral type) **، ولكن ليس من نوع تكاملي إلى نوع تعددي.**

فبتطبيق هذه القاعدة على العبارتين

```
someInt = HORSE;        // Valid
inPatient = 5;          // Error
```

نجد أن العبارة الأولى تخزن 5 **في** someInt **(بسبب التحويل الضمني للنوع) ، بينما تؤدي العبارة الثانية إلى خطأ وقت الترجمة** (compile time error) **.ويلاحظ**

أن شرط عدم السماح بتخزين قيمة صحيحة في متغير من النوع التعددي Animals يمنع من التخزين العَرَضي (accidental) لقيمة خارج المدى المسموح به مثل:

```
inPatient = 65;          // Error
```

(٢) الزيادة (Incrementation)

نفرض أننا نود زيادة (incrementing) القيمة المخزونة في المتغير inPatient لتصبح القيمة التالية في المجال . العبارة التالية

```
inPatient = inPatient + 1;       // Error
```

تعد خاطئة (illegal) ، وذلك لأنه إذا كان الطرف الأيمن سليماً لأن التحويل الضمني للنوع يسمح بإضافة قيمة inPatient إلى 1 لتصبح النتيجة قيمة من النوع int ، إلا أن عملية الإسناد خاطئة / غير سليمة (not valid) وذلك لأنه لا يجوز تخزين قيمة من النوع int في المتغير inpatient .

وللسبب نفسه فإن العبارة

```
inPatient ++;            // Error
```

تعد خاطئة لأن المترجم (compiler) يعتبرها مماثلة في معناها لعبارة الإسناد السابقة. ويمكننا التخلص من التقيد بقاعدة التحويل الضمني للنوع، وذلك باستخدام التحويل الصريح للنوع (explicit type conversion) – أي صب النوع (type cast) – بكتابة العبارة

```
inPatient = Animals (inPatient + 1) ;          // Correct
```

وزيادة متغير من النوع التعددي تفيد في كتابة عبارات التكرار. فحينما نحتاج لعروة تقوم بتشغيل (processing) جميع القيم الموجودة في مجال النوع التعددي يمكننا كتابة عروة For كالعروة التالية مع النوع التعددي Animals:

```
Animals patient;

for (patient = RODENT; patient <= SHEEP; patient = Animals (patient + 1))
⋮
```

ونؤكد هنا – مرة أخرى – على ضرورة استخدام تعبير إسناد (assignment expression) وتحويل صريح للنوع (type cast) لزيادة متغير النوع التعددي patient، وأنه من الخطأ زيادته بالتعبير patient ++ ، أي أنه من الخطأ كتابة عروة For السابقة بالصورة التالية

```
for (patient = RODENT; patient <= SHEEP; patient ++)        // Error
:
```

كما نحب أن ننبه هنا إلى أنه عندما يخرج التحكم من العروة السابقة فإن قيمة patient تكون أكبر بواحد 1 من أكبر قيمة في المجال (SHEEP). وإذا أردنا استخدام المتغير patient خارج العروة فيجب إعادة إسناد قيمة تقع في مدى النوع Animals لهذا المتغير.

(٣) المقارنة (Comparison)

تعد المقارنة أكثر العمليات تطبيقاً بالنسبة لقيم الأنواع التعددية. وتتحدد نتيجة المقارنة بين قيمتين: أيهما تسبق الأخرى (أيهما أصغر من الأخرى) بناء على ترتيب هذه القيم المذكور في الإعلان عن النوع التعددي. فمثلاً التعبير

```
inPatient <= BIRD
```

تكون قيمته true إذا احتوى inPatient على إحدى القيم RODENT, CAT, DOG, BIRD: .

ويمكننا كذلك استخدام قيم النوع التعددي في عبارة switch. فنظراً لأن RODENT, CAT, ... قيم حرفية (literals) فلذلك يمكنها الظهور في عناوين الحالة (case labels)، كما في المثال التالي (الذي يقسّم قيم Animals إلى ثلاثة أجنحة: جناح الأقفاص (cage ward) وجناح المَرْبَى اليابس (Terrarium ward)، وجناح حظيرة الماشية (Barn)]:

```
switch (inPatient)
{
```

```
        case RODENT:
        case CAT      :
        case DOG      :
        case BIRD     : cout << "Cage ward";
                        break;
        case REPTILE: cout << "Terrarium ward";
                        break;
        case HORSE    :
        case BOVINE :
        case SHEEP    : cout << "Bran";
}
```

(٤) الإدخال والإخراج (Input and Output)

يُعـرَّف فـيض المـدخلات أو المخرجـات (I/O stream) فقـط للأنـواع الأساسـية الداخليـة (basic built-in types) [int, float, …] وليس للأنـواع التعدديـة المعرفـة بالمستخدم. أما قيم هـذه الأنـواع التعدديـة فيجـب إدخالها أو إخراجها بطريقة غير مباشرة.

فلإدخال هذه القيم يمكننا قراءة سلسلة رموز (string) تقابل أحد ثوابت (constants) النـوع التعـددي. أي نُـدخل هـذه السلسـلة، ثـم نترجمها إلى إحـدى القيم الحرفية في النوع التعددي ، وذلك عن طريق النظر فقط إلى عدد من رموز / حروف هذه السلسلة يكفي لتحديد أي قيمة هي ، كما يوضح ذلك المثال التالي.

مثال ٧-٣: نفرض أن لدينا الإعلانين التاليين:

```
enum Animals {RODENT, CAT, DOG, BIRD, REPTILE, HORSE,
              BOVINE, SHEEP};

Animals inPatient
```

اكتـب قطعـة برنـامج تقـرأ سلسـلة رمـوز (string) تمثـل اسـم حيـوان مـن الأسمـاء المـذكورة ، وتُحـول (converts)هـذه السلسـلة إلى القيمـة المقابلـة في النـوع Animals، وتُخزن هذه القيمة في المتغير inPatient.

الحل: نلاحظ أن الحرف الأول في أيٍّ من الأسماء : cat, dog, horse, sheep يكفي لتحديد الاسم المقابل (القيمة المقابلة) ، بينما لا يمكننا تحديد الاسم المقابل في حالة أي من الأسماء : bovine, bird, rodent, reptile إلا بعد اختبار الحرف الثاني. [مثلاً bovine, bird لهما الحرف الأول نفسه: b، ولكنهما يختلفان في الحرف الثاني : o, i] . ولذلك يمكننا كتابة قطعة البرنامج التالية:

```
# include <cctype>              // For toupper ( )
# include <string>              // For string type
     ⋮
string animalName;
     ⋮
cin >> animalName;
switch (toupper (animalName [0]) )
{
        case 'R'        : if (toupper (animalName [1]) = = 'O' )
                                inPatient = RODENT;
                          else
                                inPatient = REPTILE;
                          break;
        case 'C'        :  inPatient = CAT;
                          break;
        case 'D'        : inPatient = DOG;
                          break;
        case 'B'        : if (toupper (animalName [1]) = = 'I' )
                                inPatient = BIRD;
                          else
                                inPatient = BOVINE;
                          break;
        case 'H'        : inPatient = HORSE;
                          break;
        default         : inPatient = SHEEP;
}
```

* * *

وكذلك لا يمكننا طباعة قيم الأنواع التعددية مباشرة. ولكن يمكننا طباعتها باستخدام عبارة Switch تطبع سلسلة رموز (character string) مقابلة للقيمة المطلوب طباعتها، كما يُبيِّن ذلك المثال التالي.

مثال ٧-٤: نفرض أن لدينا الإعلانين المذكورين في المثال السابق (مثال ٧-٣). اكتب قطعة برنامج تطبع القيمة المخزونة في المتغير inPatient.

الحل:

```
switch (inPatient)
{
        case RODENT: cout << "Rodent";
                     break;
        case CAT   : cout << "Cat";
                     break;
        case DOG   : cout << "Dog"
                     break;
        case BIRD  : cout << "Bird";
                     break;
        case REPTILE: cout << "Reptile";
                     break;
        case HORSE  : cout << "Horse";
                     break;
        case BOVINE : cout << "Bovine";
                     break;
        case SHEEP  : cout << "Sheep";
}
```

ملاحظة: يمكننا – بدلاً من استخدام الأنواع التعددية – تمثيل (representing) كل حيوان في البرنامج بشفرة (code) عبارة عن عدد صحيح (integer number) أو مجرد حرفين (a pair of letters) ولكن استخدام الأنواع التعددية يجعل البرنامج أوضح و أيسر في القراءة (more readable) والفهم وبيان المعاني(more self-documenting).

(٥) إعادة قيمة دالة (Returning a Function Value).

الدوال التي تعيد قيماً (value-returning functions) التي استخدمناها حتى الآن كانت تحسب وتعيد قيماً من الأنواع الداخلية (built-in types) مثل : int, float, char ، كالدوال

```
int Factorial (  int );
int Power (  int, int );
bool IsTriangle (  float, float, float);
```

ولغة C++ تسمح للقيمة التي تعيدها دالة أن تكون من أي نوع بيانات (any data type) [سواء كان مبنياً داخلياً (built-in) أو معرفاً بالمستخدم (user-defined)] ما عدا النوع array (والذي سندرسه بإذن الله الفصل القادم). فمثلاً كتبنا في مثال ٧-٣ عبارة Switch لتقوم بتحويل سلسلة مدخلات (input string) إلى قيمة من النوع animals. والمثال التالي يوضح كيف يمكن أن تقوم بهذه المهمة دالة تعيد قيمة.

مثال ٧-٥: نفرض أن لدينا الإعلان عن النوع التعددي Animals المذكور في مثال ٧-٣. اكتب دالة تعيد قيمة StrToAnimals تستقبل سلسلة رموز str تمثل اسم حيوان من الأسماء المذكورة في هذا الإعلان ، وتعيد القيمة المقابلة في هذا النوع التعددي.

الحل: نظراً لأن الدالة تعيد قيمة من النوع التعددي Animals ، لذلك نكتب كلمة Animals في عنوان الدالة (function heading) قبل اسمها للإعلان عن نوع بيانات (data type) القيمة التي تعيدها الدالة.

```
Animals StrToAnimals ( /* in */ string str )
{
        switch (toupper (str[0]) )
        {
                case 'R': if (toupper (str [1]) = = 'O' )
```

```
                    return RODENT;
                else
                    return = REPTILE;
            case 'C' :  return  CAT;
            case 'D' : return DOG;
            case 'B' : if (toupper (str[1]) = = 'I' )
                    return BIRD;
                else
                    return BOVINE;
            case 'H' : return HORSE;

            defult   : return SHEEP;
        }
}
```

نلاحظ في هذا الحل أن الدالة لا تحتوي على عبارة break **بعد أيٍّ من بدائل** (alternatives) case ، **وذلك لأنه عندما يقوم أحد هذه البدائل بتنفيذ عبارة** return **فإن التحكم يخرج** (exits) **فوراً من الدالة، وبالتالي فليس ممكناً أن ينتقل التحكم إلى البديل التالي.**

مثال ٧-٦: اكتب قطعة برنامج تقرأ سلسلتي رموز وتستدعى الدالة السابقة StrToAnimal (**مثال ٧-٥**) **لتحويل هاتين السلسلتين إلى القيمتين المقابلتين من النوع** Animals ، **ثم تقوم القطعة بتخزين هاتين القيمتين في المتغيرين** inPatient, outPatient.

الحل:

```
enum Animals {RODENT, CAT, DOG, BIRD, REPTILE, HORSE,
                BOVINE, SHEEP};
Animals StrToAnimal (  string );
⋮
int main ( )
{
        Animals inPatient;
        Animals outPatient;
        String inputStr;
        ⋮
```

```
        cin >> inputStr;
        inPatient = StrToAnimal (inputStr);
        ⋮
        cin >> inputStr;
        outPatient = StrToAnimal (inputStr);
        ⋮
}
```

أنواع البيانات المسمَّاة والأنواع غير المسمَّاة
Named and Anonymous Data Types

الأنواع التعددية التي درسناها حتى الان (مثل Animals, Days) تسمى "أنواعاً مسماة" (named types)، وذلك لأن الإعلان عنها احتوى على أسماء لهذه الأنواع . والمتغيرات من أنواع البيانات الجديدة هذه قد أُعلن عنها – في إعلان منفصل (separate declaration) – باستخدام هذه الأسماء التعريفية للأنواع (type identifiers) (مثل Animals, Days) .

فالنوع المسمَّى **(named type)** هو نوع معرف بالمستخدم يحتوي الإعلان عنه على اسم تعريفي للنوع يعطِي اسماً للنوع.

وتسمح لغة C++ بالإعلان عن نوع جديد مباشرة في إعلان عن متغير.فمثلاً بدلاً من الإعلانات:

```
enum SacredMonths {Moharram, Rajab, Thulquida, ThulHijja};
enum StatusType {OK, OUT_OF_STOCK, BACK_ORDERED};

SacredMonths   month;
StatusType   status;
```

يمكننا كتابة

```
enum {Moharram, Rajab, Thulquida, ThulHijja}   month;
enum {OK, OUT_OF_STOCK, BACK_ORDERED}  status;
```

والنوع الجديد الذي نعلن عنه في إعلان عن متغير (variable declaration) يطلق عليه "نوع غير مسمّى " (an anonymous type) لأنه ليس له اسم، أي ليس له اسم تعريفي للنوع (type identifier) مرتبط به.

وعموماً نفضل استخدام الأنواع المسمَّاة لأنها تجعل البرنامج أكثر وضوحاً و أيسر في القراءة والفهم والتعديل. وكذلك كي نُبقى المفهومين المختلفين: الإعلان عن نوع والإعلان عن متغير من هذا النوع منفصلين.

ويمكننا الآن أن نضع الصيغة العامة للإعلان عن النوع التعددي في الصورة التالية:

enum Name{Enumerator1,Enumerator2,…}VarName1,VarName2,..;

حيث الاسم التعريفي للنوع Name اختياري (optional) ، وكذلك قائمة المتغيرات (list of Variables) VarName1, VarName2, … اختيارية.

ملفات المقدمة التي يكتبها المستخدِم

User-Written Header Files

عندما ننشئ أنواع بيانات معرفة بالمستخدم ، فكثيراً ما نجد أن أحد هذه الأنواع يمكن أن يفيدنا في أكثر من برنامج واحد. فمثلاً قد يكون لدينا عدة برامج تحتاج إلى نوع تعددي يتكون من أسماء الاثنى عشر شهراً في السنة . فبدلاً من كتابة العبارة

```
eum Months
{
        JAUNUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE,
        JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER,
        DECEMBER
};
```

في بداية كل برنامج يستخدم النوع Months ، يمكننا وضع هذه العبارة في ملف منفصل نُسمِّيه – مثلاً – months.h . ثم نستخدم هذا الملف months.h تماماً كما نستخدم ملفات المقدمة التي يمدنا النظام بها (system-supplied header files) كالملفين iostream, cmath مثلاً. وباستخدام موجِّه include directive # فإننا نطلب من مشغِّل C++ المبدئي أن يُدخل محتويات الملف (فيزيائياً physically) في البرنامج. [ملاحظة: نظم (systems) C++ عديدة تستخدم امتداد اسم الملف ".h" (filename extension) – أو لا تستخدم أي امتداد على الاطلاق – للدلالة على ملفات المقدمة، بينما تستخدم نظم أخرى امتدادات أخرى مثل ".hpp" أو ".hxx"].

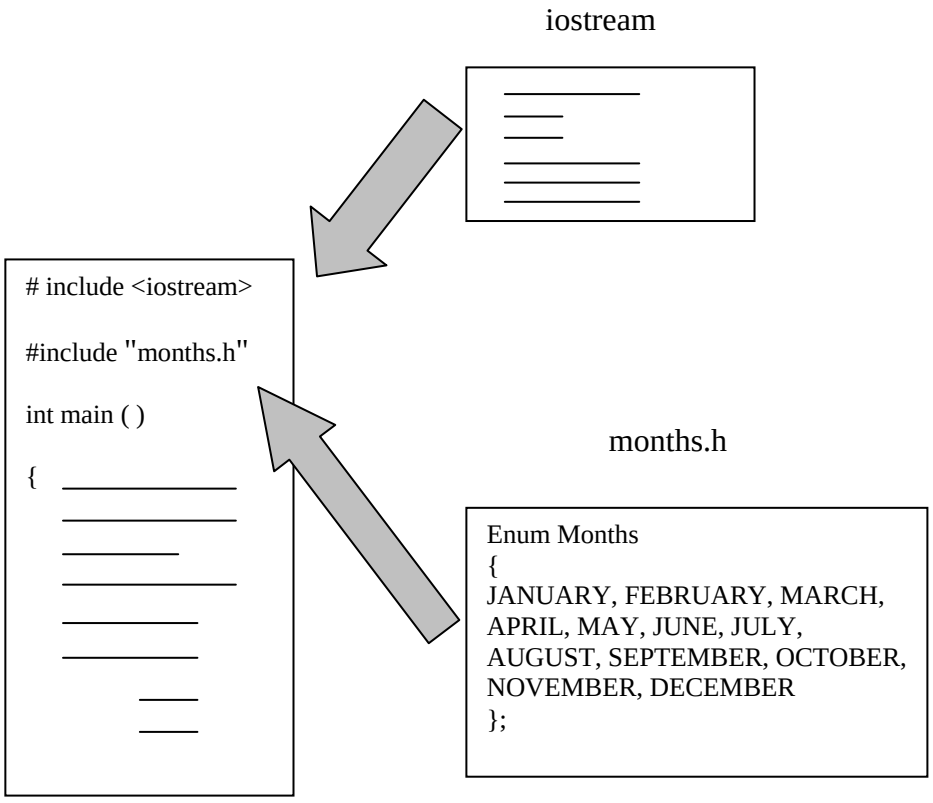
وحينما نضع اسم ملف مقدمة بين قوسي الزاوية (angle brackets) مثل

```
# include <iostream>
```

فإن المشغِّل المبدئي (preprocessor) يبحث عن الملف في دليل include القياسي (standard include directory) ، وهو عبارة عن دليل يحتوي على جميع ملفات المقدمة التي يوفرها لنا نظام (system) C++. ومن ناحية أخرى يمكننا أن نضع اسم ملف مقدمة بين حاصرتين مزدوجتين (double quotes) مثل

```
# include "months.h"
```

وفي هذه الحالة فإن المشغِّل المبدئي يبحث عن الملف في الدليل الحالي للمبرمج (programmer's current directory) وهذه الآلية (mechanism) تسمح لنا بكتابة ملفات المقدمة الخاصة بنا التي تحتوي على إعلانات النوع وإعلانات الثوابت. ويمكننا استخدام موجِّه #include directive بسيط بدلاً من إعادة كتابة الإعلانات في كل برنامج يحتاجها (انظر شكل ٧-٢).



شكل ٧-٢

الاحتواء على ملفات المقدمة

Including Header Files

ملاحظات على تحويل النوع Remarks on Type Coercion

علمنا سابقاً أن لغة C++ **تقوم بعملية التحويل الضمني للنوع** (implicit type coercion) **كلما استخدمنا قيماً من أنواع بيانات مختلفة فيما يلي:**

(١) **تعابير حسابية وتعابير علاقية** (arithmetic and relational expressions)

(٢) **عمليات إسناد** (assignment operations).

(٣) **تمرير وسطاء فعليين** (argument passing).

(٤) عودة قيمة دالة (return of a function value) من دالة تعيد قيمة (value-returning function).

في الحالة الأولى رقم (١) [التعابير ذات الأنواع المختلفة (mixed type expressions) يتَّبع مترجم C++ (compiler) مجموعة معينة من القواعد (set of rules) لتحويل النوع. وفي الحالات الأخرى [الحالات رقم (٢) ، (٣)، (٤)] يتبع المترجم مجموعة أخرى من القواعد . وفيما يلي نذكر هاتين المجموعتين.

أولاً: تحويل النوع في التعابير الحسابية والتعابير العلاقية
Type Coercion in Arithmetic and Relational Expressions

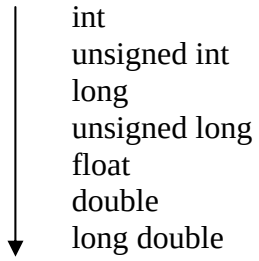
نفرض أن تعبيراً حسابياً يتكون من مؤثِّر / معامِل (operator) ومعامَلين (2 operands) مثل : sum * 3.4 أو var1 / var2. إذا كان المعامَلان من نوعين مختلفين فإن أحدهما تُرفع مرتبته / يُرقَّى (promoted) / يُوَسَّع (widened) مؤقتاً (temporarily) ليوائم (match) نوع بيانات المعامَل الآخر . وكي نفهم معنى رفع المرتبة / الترقية (promotion) نذكر القاعدة التالية – من خطوتين – في عملية تحويل النوع في التعبير الحسابي:

الخطوة رقم ١: أي قيمة من أحد الأنواع char, short, bool, enumeration تُرقَّى / توسَّع إلى قيمة من النوع int. إذا أصبح المعامَلان الآن من النوع int ، فإن النتيجة (result) تكون تعبيراً من النوع int.

الخطوة رقم ٢: بعد تطبيق الخطوة رقم ١، إذا ظل لدينا تعبير من نوع مختلط، فإننا نستخدم القاعدة التالية للأولويات بالنسبة للأنواع وذلك لترقية قيمة المعامَل ذي النوع الأدنى (lower type) إلى قيمة من النوع الأعلى (higher type) ، وتكون النتيجة تعبيراً من هذا النوع الأعلى.

قاعدة الأولويات بالنسبة للأنواع Precedence of Types

فيما يلي الأنواع مرتبة من الأدنى (lowest) إلى الأعلى (highest):



وكمثال بسيط نفرض أن لدينا التعبير someFloat + 5.
هذا التعبير لا يحتوي على أي قيمة من أحد الأنواع char, short, bool, enumeration . وبالتالي فالخطوة رقم ١ تظل تترك لنا تعبيراً من نوع مختلط. وبتطبيق الخطوة رقم ٢ نجد أن النوع int أدنى / أقل مرتبة من النوع float، ولذلك فإن القيمة 5 تحوَّل مؤقتاً إلى قيمة من النوع float ولتكن مثلاً 5.0 . ثم تتم عملية الجمع ويكون التعبير الكلي (entire expression) من النوع float.

وعملية تحويل النوع بالنسبة للتعابير العلاقية تتم بطريقة مماثلة فمثلاً لو كان لدينا التعبير العلاقي someFloat <= someInt فإن قيمة someInt تحوَّل مؤقتاً إلى تمثيل ذي نقطة عائمة (floating-point representation) قبل أن تُجرَى عملية المقارنة (comparison) . والفارق الوحيد بين التعابير الحسابية والتعابير العلاقية هو أن نتيجة التعبير العلاقي تكون دائما من النوع bool، حيث تكون دائماً إما القيمة true أو القيمة false.

والجدول التالي يصف نتيجة ترقية قيمة من نوع بسيط إلى نوع آخر في لغة C++.

نتيجة الترقية Result of Promotion	إلى To	من From
القيمة نفسها، ولكنها تحتل حيزاً أكبر من الذاكرة	long double	double
القيمة نفسها، ولكنها تحتل حيزاً أكبر من الذاكرة	double	float
القيمة ذو النقطة العائمة المكافئة للقيمة الصحيحة – الجزء الكسري يساوي صفراً	النوع ذو النقطة العائمة Floating-point type	النوع التكاملي Integral type
القيمة نفسها إذا كان العدد الأصلي غير سالب، وعدد موجب مختلف جذرياً إذا كان العدد الأصلي سالباً.	نظيرة بدون إشارة Its unsigned counterpart	النوع التكاملي Integral type
القيمة نفسها، ولكنها تحتل حيزاً أكبر من الذاكرة	النوع التكاملي الأطول مع إشارة longer signed integral type	النوع التكاملي مع إشارة Signed integral type
القيمة غير السالبة نفسها، و تحتل حيزاً أكبر من الذاكرة	النوع التكاملي الأطول (بإشارة أو بدون إشارة) longer integral type (either signed or unsigned)	النوع التكاملي بدون إشارة Unsigned integral type

ملاحظة: نتيجة ترقية char إلى int تعتمد على المترجم (compiler dependent) . فبعضها يعامل char كأنه char بدون إشارة : unsigend char ، وبالتالي تؤدي الترقية دائماً إلى عدد صحيح غير سالب (nonnegative integer) ، وفي البعض الآخر char تعنى char مع إشارة : signed char، وبالتالي تؤدي ترقية قيمة سالبة إلى عدد صحيح سالب (negative integer) .

ثانياً: تحويل النوع في عمليات الإسناد ، وتمرير الوسطاء الفعليين ، وعودة قيمة دالة

Type Coercion in Assignments, Argument Passing and Return of a Function Value

عموماً لا تؤدي عملية الترقية (promotion) / التوسيع(widening) إلى أي فقدان للمعلومات (loss of information). ويمكن النظر إليها على أنها تماثل عملية نقل

محتويات صندوق صغير ممتلئ بالكتب إلى صندوق آخر أكبر / أوسع، فيمكن وضع جميع الكتب في الصندوق الجديد ويظل به حيز فارغ. وفي المقابل فإن عملية إنـزال مرتبـة (demotion) / التضـييق علـى (narrowing) قـيم البيانـات (data values) – وهي عكس عملية الترقية / رفع المرتبة / التوسيع – تماثل عملية نقل محتويات صندوق كبير ممتلئ بالكتب إلى صندوق أصغر، فنضطر لإلقاء (throwing out) بعض الكتب، وبالتالي فإن عملية إنزال المرتبة تؤدي إلى فقدان للمعلومات.

أي أن

عملية الترقية / رفع المرتبة / التوسيع (promotion /widening) هي تحويل قيمة من نوع أدنى إلى نوع أعلى بناء على أولويات أنواع البيانات في لغة البرمجة، ولا تؤدي هذه العملية إلى فقدان للمعلومات، بينما

عملية إنزال المرتبة / التضييق (demotion / narrowing) هي تحويل قيمة من نوع أعلى إلى نوع أدنى ، وقد تؤدي إلى فقدان للمعلومات.

نفرض أن لدينا عملية الإسناد

v = e

حيث v متغير و e تعبير. بالنسبة لنوعي بيانات v, e هناك ثلاثة احتمالات:

(١) إذا كان نوع v هو نفسه نوع e ، فليس هناك حاجة لتحويل النوع.

(٢) إذا كان نوع v أعلى من نوع e ، فإن قيمة e ترقى إلى نوع v قبل تخزينها في v.

(٣) إذا كان نوع v أدنى من نوع e ، فإن مرتبة قيمة e تنزل إلى مرتبة نوع v قبل تخزينها في v.

فإنزال المرتبة (demotion) – والذي يمكن أن ننظر إليه على أنه عملية انكماش (shrinking) للقيمة – قد يؤدي إلى فقدان للمعلومات:

- فإنزال المرتبة من نوع تكاملي أطول (longer integral type) إلى نوع تكاملي أقصر (shorter) [مثل: من long إلى int] يؤدي إلى إهمال /

رمـي / طـرح (discarding) الأرقـام الثنائيـة الموجـودة أقصـى اليسـار (leftmost bits) – وهـي الأرقـام ذات أعلـى قيمـة معنويـة (most significant) – في التمثيـل الثنـائي للأعـداد (binary number representation). فالنتيجة قد تكون عدداً مختلفاً اختلافاً كبيراً عن العدد الأصلي.

- وإنزال المرتبة من نـوع عائم (a floating-point type) إلى نـوع تكاملي (an integral type) يـؤدي إلى قطـع (truncation) الجـزء الكسـري (fractional part) [ونتيجـة غير معرفة / غير محـددة (an undefined result) إذا لم يكن المتغير التكاملي الذي يستقبل نتيجة التحويل (المتغير المستقبل destination variable) كافياً (fit) لتخزين الجـزء الصحيح مـن العدد (whole-number part)] . وأما نتيجة قطع عدد سالب فتعتمد على الجهاز المستخدم.
- وإنزال المرتبة من نوع عائم أطول إلى نوع عائم أقصر (مثل: من double إلى float) قد يؤدي إلى فقدان أرقام من أرقام الدقة (loss of digits of precision) .

وما ذكرناه عن كيفية تحويل النوع في عملية الإسناد ينطبق أيضاً في حالة تمرير الوسطاء الفعليين [عمليـة إسقاط الوسطاء الفعليين على الوسطاء الشكليين (mapping of arguments onto parameters)] وفي حالة إعادة قيمة دالة بعبارة Return . مـثلاً نفـرض أن قيمـة INT-MAX في جهازنـا هـي 32767 ، وأن لـدينا الدالة التالية:

```
void DoSomething (  int n)
{
        :
}
```

وأن الدالة قد استُدعيت بالعبارة

```
DoSomething (50000);
```

نظراً لأن القيمة 50000 تُعد ضمنياً من النوع long لأنها أكبر من INT_MAX ، فإن هذه القيمة 50000 يتم إنزال مرتبتها إلى قيمة أصغر مختلفة تماماً لتُناسب (fits) سعة موضع (location) من النوع int.

وبطريقة مماثلة فإن تنفيذ الدالة

```
int SomeFunc (  float x)
{
        :
        return 70000;
}
```

يؤدي إلى إنزال القيمة 70000 إلى قيمة أصغر من النوع int لأن int هو النوع المعلَن لقيمة الدالة المعادة.

ملاحظة (١) : عند اختيار نوع بيانات لمتغير يقوم بتخزين أعداد كاملة (whole numbers) فقط، فإن النوع int يجب أن يكون أول اختيار لنا للأسباب الآتية:

- الحسابات ذات النقطة العائمة (floating-point arithmetic) تكون نتائجها عرضة لعدم دقة عددية (numerical inaccuracies) ، وهي عموماً أبطأ–على معظم الأجهزة – من الحسابات ذات القيم الصحيحة.
- استخدام النوعين التكاملين الأصغرين (smaller integral types) : short , char يمكن أن يؤدي بسهولة إلى أخطاء بسبب الفيض (overflow errors).
- النوع long يتطلب عادة حيزاً أكبر في الذاكرة من النوع int، وكذلك يتطلب عادة وقتاً أكبر لإجراء الحسابات (arithmetic) أي أنه عادة أبطأ.

ملاحظة (٢): عندما تكون البيانات من نوع النقطة العائمة (floating-point) ونجرى الحسابات عليها فإننا نتعرض لمفهومي : الأرقام المعنوية والدِّقة ، ونقصد بهما:

الأرقام المعنوية (significant digits) : هي الأرقام من أول رقم غير صفري (first nonzero digit) **على اليسار إلى آخر رقم غير صفري على اليمين [مضافاً إليها** (plus) **أي أرقام صفرية 0 صحيحة / مضبوطة** (exact) **].**

والدقة (precision): هي أكبر عدد (maximum number) **من الأرقام المعنوية [يمكن تمثيله** (can be represented) **] .**

فمثلاً إذا فرضنا أن الحاسوب / النظام الذي نستخدمه يَحُدُّ (limits) **من دقة الأعداد ذات النقطة العائمة بحيث يسمح بعشرة أرقام فقط** (supports ten digits of precision for floating-point numbers) **فإن الحسابات التالية**

i) 1.4 E + 12 + 100.0
ii) 4.2 E – 8 + 100.0
iii) 3.2 E – 5 + 3.2 E + 5

تكون نتائجها

i) 1.4 E + 12 (to 10 digits)
ii) 100.0 (to 10 digits)
iii) 3.2 E + 5 (to 10 digits)

تمرينات رقم ٧

٧-١) نفرض أن لدينا إعلان النوع

```
enum VisibleColors
{
    RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, VIOLET
};
```

اكتب السطر الأول من عبارة For **تقوم بالعد** (counting) **ابتداءً من** RED **وحتى** VIOLET . **استخدم متغير تحكم في العروة اسمه** rainbow **من النوع** VisibleColors.

٧-٢) نفرض أن حاسب تبلغ دقته أربعة أرقام (fourr digits of precision) **ما هي نتيجة عملية الجمع التالية؟**

400400.000 + 199.9

٧-٣) اذكر ما إذا كان كل مما يلي تعبيراً (expresion) **أم عبارة تعبير** (expression statement).

(أ) sum = 0

(ب) sqrt (x)

(ج) y = 17;

(د) count ++

٧-٤) أعد كتابة كل من العبارات التالية كما هو مبين:

(أ) sumOfSquares = sumOfSquares + x * x;

استخدم المؤثر =+

(ب) count = count -1;

استخدم مؤثر النقصان (decrement operator)

(ج)
```
if (n > 8)
  k = 32;
else
```

```
k = 15 * n;
```

استخدم عبارة إسناد واحدة (single assignment statement) **تستخدم المؤثر** :?

٧-٥) نفرض أن ch **متغير من النوع** char. **ماذا تطبع كل من قطعتي البرنامج التاليتين؟**

(أ)

```
for (ch = 'd'; ch <= 'g'; ch ++}
        cout << ch;
```

(ب)

```
ch = 'F';
cout << ch <<  ' ' << int (ch); // Assume ASCII
```

٧-٦) ماذا تطبع عبارة الإخراج التالية؟

```
cout << "Notice that \nthe character \\ is a backslash. \n ";
```

٧-٧) نفرض أن لدينا الإعلانين التاليين:

```
enum SeasonType {WINTER, SPRING, SUMMER, FALL};
SEASONTYPE  SEASON;
```

اذكر ما إذا كانت كل عبارة من العبارات التالية صحيحة (vaild) **أم خاطئة** (invalid).

أ)
```
season = WINTER;
```

ب)
```
season = 3;
```

ج)
```
season ++;
```

د)
```
season = SeasonType (season +1);
```

هـ)
```
cin >> season;
```

و)
```
if (season >= SPRING)
        ⋮
```

ز)
```
for ( season = WINTER; season <= SUMMER;
        season = SeasonType (season +1))
```

٧-٨) اذكر صحة أو خطأ (T/F) كل من العبارات التالية:

أ) أي مترجم C++ يضمن تحقق العلاقة

ziseof (int) < size (long)

ب) نفرض أن لدينا قطعة البرنامج التالية

```
enum Colors (RED, GREEN, BLUE);

Colors myColor;
enum {Red, GREEN, BLUE} yourColor;
```

نوع بيانات myColor هو نوع مسمَّى ، ونوع بيانات yourColor هو نوع غير مسمَّى.

ج) إذا كتبت ملف مقدمة (header file) خاصاً بك اسمه mytypes.h فإن كتابة موجِّه المشغِّل المبدئي (preprocessor directive)

```
# include <mytypes.h>
```

هو الطريقة الصحيحة لإدخال (inserting) محتويات ملف المقدمة في برنامج.

د) العبارة alpha /= beta + 25; تكافئ العبارة alpfa = alpha / (beta + 25);

هـ) العبارة char someChar = '\' '; تقوم بتخزين رمز الحاصرة المفردة في someChar.

و) إذا كان النظام يسمح بدقة تصل إلى أربعة أرقام بالنسبة للأعداد ذات النقطة العائمة فإن نتيجة العملية الحسابية3.051 + 5432.1 هي 5435.151

ز) إذا كان النظام يسمح بدقة تصل إلى ١٠ أرقام بالنسبة للأعداد ذات النقطة العائمة فإن نتيجة العملية الحسابية 100.0 + 3.2E-12 هي 100.0.

ح) إذا كان المتغير someInt من النوع int يحتوي على قيمة تتراوح من 0 إلى 9 ، فإن العبارة التالية تقوم بتخزين الرمز الرقمي المقابل (corresponding digit character) في someChar:

```
someChar = char ('0' + someInt);
```

٧-٩) في كل من الحالات التالية اذكر ما إذا كانت ستحدث عملية ترقية / رفع مرتبة (promotion) أم عملية إنزال (demotion) [أسماء المتغيرات تشير إلى أنواع البيانات].

أ) تنفيذ عملية الإسناد
```
someInt = someFloat
```

ب) تقييم (evaluating) التعبير
```
someFloat + someLong
```

ج) تمرير الوسيط الفعلي someDouble للوسيط الشكلي someFloat

د) تنفيذ العبارة التالية في دالة من النوع int
```
return someShort;
```

٧-١٠) اكتب عروة while تنسخ (copies)جميع الرموز [بما فيها الرموز الفراغية البيضاء [(whitespace characters)] من سيل ملف إدخال inFile (input file stream)] إلى سيل ملف إخراج outFile، باستثناء تحويل كل حرف صغير (lowercase letter) إلى حرف كبير (uppercase) . افرض أن كلا الملفين قد تم فتحهما قبل أن تبدأ العروة . ويجب أن تنتهي العروة حين اكتشاف (detecting) الوصول إلى نهاية الملف.

٧-١١) نفرض أن لدينا الإعلانات التالية:

```
int  n;
char ch1;
char ch2;
```

ونفرض أن n تحتوي على عدد من رقمين (two-digit number). حوِّل n إلى رمزين مفردين (ch1, ch2 (two single characters بحيث يحتوي ch1 على الرقم الأعلى رتبة (higher-order digit)، ويحتوي ch2 على الرقم الأقل رتبة (lower-order digit) . مثلاً إذا كانت n = 59 فإن ch1 سيساوي '5' بينما ch2

يساوي '9' . ثم اطبع الرقمين كرمزين بترتيبهما نفسه(same order) في العدد الأصلي [إرشاد: استخدم المؤثرين : % /,].

٧-١٢) عند كتابة برنامج ما نفرض أنه من المحتمل أن يحتوي المتغير beta من النوع float على عدد كبير جداً. وقبل أن نضرب beta بالقيمة 100.0 نود أن يقوم البرنامج باختبار ما إذا كان ذلك ممكناً بحيث لا يؤدي إلى فيض (overflow). اكتب عبارة If تقوم بهذا الاختبار بحيث أنه إذا كان ذلك سيؤدي إلى فيض فتطبع رسالة تفيد ذلك ولا تقوم بعملية الضرب، وإلا فإنها تجرى عملية الضرب.

[ملاحظة : FLT_MAX هو أكبر عدد مسموح به من النوع float].

٧-١٣) اكتب إعلاناً عن نوع تعددي لكل مما يلي:

أ) أرقام مقررات علم الحاسوب التالية:
CS126, CS206, CS300, CS337, CS352, CS455

ب) الأشهر الحرم : المحرم ، رجب ، ذو القعدة ، ذو الحجة.

ج) أشهر الحج: شوال، ذو القعدة ، ذو الحجة.

د) أيام الأسبوع الخمسة: الاثنين ، الثلاثاء ، الأربعاء ، الخميس ، الجمعة.

٧-١٤) اكتب دالة خاوية تطبع قيمة من النوع المعلن عنه في السؤال ٧-١٣-د.

٧-١٥) اكتب عروة For تقوم بطباعة جميع القيم الخمس في مجال (domain) النوع المعلن عنه في السؤال ٧-١٣-د) ، بحيث يكون today متغير التحكم في العروة من هذا النوع المعلن ، وبحيث تستدعي دالة السؤال السابق (٧-١٤) لطباعة أي قيمة.

٧-١٦) من المفروض أن تقوم الدالة التالية بإعادة النسبة بين عددين صحيحين مقربة لأقرب عدد صحيح

```
int Ratio  (  /* in */ int int1,
              / * in * / int int2  )
{
        return float ( int1) / float ( int2);
}
```

ولكن أحياناً تعيد الدالة نتيجة غير صحيحة . اذكر السبب بدلالة ترقية النوع (type promotion) **أو إنزال مرتبته** (demotion)**، واعمل التصحيح اللازم كي تؤدي الدالة وظيفتها.**

٧-١٧) اكتب دالة تعيد قيمة بحيث تحوِّل الدالة أول حرفين من حروف أي يوم من الأيام المعلن عنها في السؤال ٧-١٣-د) إلى اليوم المقابل. مثلاً : إذا كان الحرف الأول 'T' **والثاني** 'U' **فإن الدالة تعيد القيمة** TUESDAY.

الفصل الثامن

المنظومات

Arrays

تمهيد

نفرض أن المطلوب قراءة درجات مائتي طالب في أحد الاختبارات وحساب المتوسط العام للدرجات وهو يساوي مجموع درجات كل الطلاب مقسوماً على عددهم أي على مائتين.

يمكننا إعطاء هذه الدرجات أسماء مختلفة مثل

S0, S1, S2, S3, …., S199

ثم قراءة قيم هذه الأسماء – أي قراءة الدرجات – وجمعها، ولكن هذه الطريقة تكون مملة جداً ومتعبة حيث أنه يلزمنا كتابة جميع هذه الأسماء في القراءة وفي الجمع، بالإضافة إلى أنا نشغل من ذاكرة الحاسب حيزاً كبيراً للتخزين لأن كل متغير (كل درجة) له اسم مختلف وبالتالي يشغل حيزاً مختلفاً.

إذا كان المطلوب هو حساب المتوسط فقط فإننا بعد قراءة أي درجة يمكننا إضافتها إلى المجموع وبعد ذلك لا نحتاج لهذه الدرجة في أي عملية أخرى ولذلك فيمكننا أن نسمي الدرجة التالية باسم الدرجة السابقة نفسه، ونقرأها تحت هذا الاسم نفسه فتحل محل الدرجة السابقة في مكان تخزينها في الذاكرة، ثم نضيف هذه الدرجة الجديدة إلى المجموع، ونكرر هذه العملية مع الدرجة التالية، ثم التي بعدها، وهكذا إلى أن نقرأ كل الدرجات – تحت الاسم الواحد نفسه – ونضيفها جمعياً إلى بعضها البعض لنحصل على المجموع الكلي الذي يقسم بعد ذلك على عدد هذه الدرجات ليعطي المتوسط.. وبذلك نكون قد حققنا أمرين: الأول : سهولة كتابة البرنامج حيث أعطينا كل الدرجات الاسم نفسه، والثاني: شغل أقل حيز ممكن من ذاكرة الحاسب حيث استخدمنا موضعاً واحداً فقط للدرجات حيث كانت كل درجة تحل محل الدرجة السابقة في الموضع نفسه.

(وسبق أن عملنا خطوات شبيهة بتلك في مسائل سابقة) ويمكن أن يتمثل هذا الحل الذي ذكرناه في مجموعة العبارات التالية:

```
sum = o;
for (i = 0;        i < 200;          i ++)
{
        cin >> S [i];
        sum = sum + S [i];
}
avg = sum / 200.0;
cout << "average = " avg << endl;
```

فإذا طلب منا – بالإضافة إلى حساب هذه الدرجة المتوسطة avg – حساب الفرق بين درجة كل طالب والدرجة المتوسطة، فإنه يلزمنا قراءة درجات الطلاب مرة أخرى حيث أنها كانت تمحى أولاً بأول كما سبق شرحه، وهذا غير مقبول من الناحية العملية أن نعيد قراءة كل البيانات كلما احتجنا إليها وربما نحتاج إليها أكثر من مرتين أيضاً فقد نرغب مثلاً في معرفة أعلى درجة ,واقل درجة أو ترتيب الدرجات ترتيباً تصاعدياً أو تنازلياً أو حساب الانحراف القياسي للدرجات (وهو أحد المقاييس في علم الإحصاء) أو تقسيم الدرجات إلى فئات مختلفة لإعطاء الطلاب التقديرات المقابلة (ممتاز – جيد جداً – جيد – مقبول – راسب) وعمل إحصائية لعدد الطلاب في كل فئة، وهكذا ...

في مثل هذه الحالات التي نحتاج فيها إلى الاحتفاظ بالبيانات وعدم محوها من ذاكرة الحاسب ، يجب أن تعطي هذه البيانات أسماء مختلفة حتى لا تحل قيمة إحداها محل قيمة أخرى.. وهنا تظهر مرة أخرى مشكلة كيفية كتابة هذه الأسماء الكثيرة المختلفة أثناء كتابة البرنامج ، وهل نضطر لكتابة أسماء مجموعة الدرجات جميعها في البرنامج وتصبح بذلك كتابة البرنامج عملاً شاقاً ومملاً. فكرة المنظومات – وهي موضوع هذا الفصل – تحل هذه المشكلة ، وتتلخص في أننا نعطي المجموعة كلها (مجموعة الدرجات مثلاً) اسماً واحداً ولكن إذا أضفنا له قوسين مربعين ووضعنا بينهما قيمة تشير إلى رقم العنصر في المجموعة فإننا نحصل على اسم هذا العنصر، فمثلاً إذا أشرنا إلى مجموعة الدرجات بالاسم S

، فإن S[4] مثلاً تعني درجة الطالب رقم 4 (الطالب الخامس) ، و S[140] تعني درجة الطالب رقم 140 وهكذا ، وعلى العموم فإن S[i] تعني درجة الطالب رقم i ، فإذا كتبنا في البرنامج بعض العبارات التي تشير إلى تغير منتظم في قيمة i فإنه يمكننا إجراء أي عمليات على كل الدرجات التي نحتاج إليها دون أن نضطر لكتابة أسماء كل هذه الدرجات.

على أنه يلزم قبل إجراء أي عمليات على هذه العناصر من المنظومة سواء قراءتها أو جمعها أو أي عملية أخرى أن نعطي الحاسب في برنامجنا إعلاناً لتعريفه بهذه المنظومة (Array Declaration) : ما سامها؟ وكم عدد العناصر التي فيها ؟ وما نوع هذه العناصر؟ .. فمثلاً الإعلان

```
float S[200];
```

يفيد أن هناك منظومة اسمها S، وعدد عناصرها 200، وكلها أعداد من النوع float.

وفي حالة ما إذا كانت عناصر منظومة ما كلها أعداداً صحيحة فإنها تستخدم كلمة int بدلاً من كلمة float ، مثل:

```
int Alfa [50];
```

وهذا الإعلان يطلب من الحاسب حجز خمسين موضعاً في ذاكرته لعناصر المنظومة Alfa، وهذه العناصر أعداد صحيحة، بينما الإعلان السابق يطلب حجز مائتي موضع لعناصر من النوع float هي عناصر المنظومة S.

وتُعامَل عناصر المنظومة في عبارات البرنامج معاملة المتغيرات البسيطة (simple variables) ، فمثلاً العبارة:

```
sum = sum + S [100];
```

تعني إضافة العنصر رقم 100 في المنظومة S إلى المجموع sum ، ولاحظ أن S [100] هو اسم متغير مؤشَّر (subscripted variable) وهو الذي يرمز له رياضياً بالرمز S_{100}.

ولقراءة أو طباعة عناصر منظومة يمكننا استخدام إحدى عبارات التكرار وأنسبها عبارة for ، مثل

```
for (i = 0;   i < 200;   i ++)
        cin >> S [i];
```

والآن يمكننا في المثال التالي صياغة وحل المسألة التي ناقشناها سابقاً.

مثال ٨-١: اكتب برنامجاً لقراءة درجات مائتي طالب

$S_0, S_1, S_2, S_3, \ldots, S_{199}$

ثم حساب

أ) الدرجة المتوسطة

$$avg = (\sum_{i=0}^{199} S_i) / 200$$

ب) الفرق بين درجة كل طالب والدرجة المتوسطة

$$D_i = S_i - avg; \qquad i = 0,1,2,...,199$$

الحل: نلاحظ هنا أن D تمثل كذلك منظومة وهي منظومة الفروق، ولكننا لا نستطيع أن نحسب عناصرها إلا بعد حساب الدرجة المتوسطة avg.

```
// ******************************************
// ScoresAverage program
// ******************************************
#include <iostream>

using namespace std;

int main ( )
{
        float   S[200];
        float   D[200];
        int i;
        float   sum;
        float   avg;

        sum =  0;
        for (i = 0;   i < 200;   i ++)
        {
                cin >> S[i];
                sum = sum + S[i];
        }
        avg = sum / 200.0;
        cout << "average = " << avg << endl;
        for (i = 0;   i < 200;   i ++)
```

```
        {
                D [i] = S[i] - avg;
                cout << " i = " << i << "   difference =" << D [i] <<
endl;
        }
        return 0;
}
```

مثال ٨-٢: اكتب برنامجاً لقراءة ألف قيمة صحيحة ثم طباعتها بترتيب معكوس (reverse order).

الحل:

```
// ****************************************
// ReverseNumbers program
// ****************************************
#include <iostream>

using namespace std;

int main ( )
{
        int value [1000];
        int number;

        for (number = 0; number < 1000; number ++)
                cin >> value [number];
        for (number = 999; number >= 0; number --)
                cout << value [number] << endl;
        return 0;
}
```

المنظومات أحادية البعد One-Dimensional Arrays

يطلق على المتغير value **في المثال الأخير (مثال ٨-٢) - كما يطلق على المتغير** S **في المثال السابق (مثال ٨-١) - منظومة أحادية البعد** (one-dimensional array) **، و هي مجموعة** (collection) **من المتغيرات -**

جميعها من النوع نفسه – والجزء الأخير هو قيمة مؤشر (index value) محصورة بين قوسين مربعين. ففي المثال الأخير مثلاً القيمة المخزونة في المتغير number تسمي المؤشر (index) .

والإعلان عن المنظومة أحادية البعد شبيه بالإعلان عن متغير بسيط (متغير من نوع بيانات بسيط) باستثناء واحد وهو أنه يجب علينا الإعلان عن حجم (size) المنظومة، وذلك بأن نضع بين قوسين مربعين عدد عناصر (elements) / مركبات (components) المنظومة ، مثل :

```
int value [1000]
```

فهذا الإعلان ينشئ منظومة عدد مركباتها 1000، وجميعها من النوع int، وأول مركبة لها قيمة مؤشر 0 (index value) ، وثاني مركبة لها قيمة مؤشر 1، وآخر مركبة لها قيمة مؤشر 999.

ويمكن الوصول إلى أي عنصر / مركبة من عناصر / مركبات المنظومة عن طريق تحديد موضع (position) المركبة بقيمة مؤشر وحيدة [المنظومات متعددة الأبعاد (multidimensional arrays) – وهي المنظومات التي لها أكثر من قيمة مؤشر واحدة – ليست من محتويات هذا الكتاب].

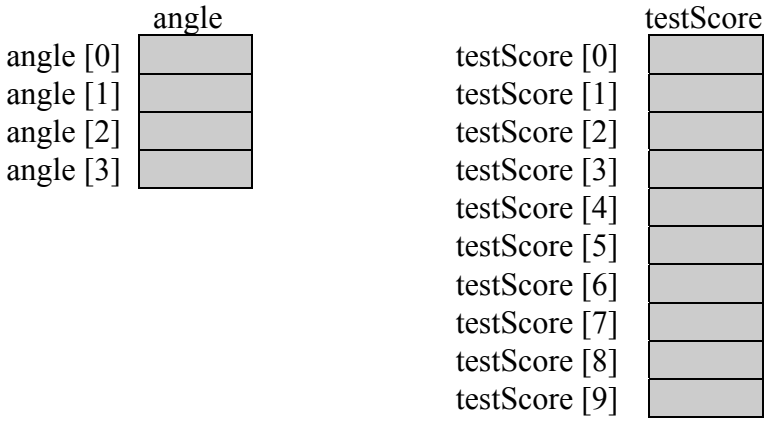
والصيغة العامة للإعلان عن المنظومة أحادية البعد هي:

```
DataType ArrayName [ConstIntExpression];
```

حيث ConstIntExpression تعبير صحيح (integer expression) مكوَّن من ثوابت حرفية (literal constants) أو ثوابت مسمَّاة (named constants) فقط. وقيمة هذا التعبير الذي يحدد عدد المركبات في المنظومة يجب أن تكون أكبر من 0. وإذا كانت هذه القيمة تساوي n، فإن مدى (range) قيم المؤشر (index values) تكون من 0 إلى n-1 وليس من 1 إلى n . فمثلاً الإعلانان

```
float angle [4];
int testScore [10];
```

ينشئان المنظومتين المبينتين في الشكل التالي (شكل ٨-١).



شكل ٨-١

المنظومتان **angle, testScore**

ويمكن الوصول إلى أي عنصر من عناصر منظومة بكتابة اسم المنظومة يلية تعبير بين قوسين مربعين، وقيمة هذا التعبير تحدد العنصر المطلوب الوصول إليه . أي أن الصيغة العامة للوصول إلى عنصر منظومة هي:

ArrayName [IndexExpression]

والتعبير IndexExpression (تعبير المؤشر) قد يكون تعبيراً بسيطاً كثابت أو اسم متغير أو تعبيراً مركباً كخليط من متغيرات ومؤثرات (operators) واستدعاءات دوال (function calls). وأيا كانت صيغة التعبير فيجب أن تكون نتيجته قيمة صحيحة (integer value). وتعبير المؤشر (index expression) هذا يمكن أن يكزن من أحد الأنواع التالية: char, short, int, long, bool وذلك لأن هذه الأنواع جميعها تكاملية (integral types). وكذلك يمكن لقيم من الأنواع التعددية (enumeration types) أن تُستخدم كتعابير مؤشر حيث تُحوَّل القيمة التعددية ضمنياً (implicitly coerced) إلى عدد صحيح.

وأبسط صيغة لتعبير المؤشر هي : ثابت. فمثلاً بالنسبة للمنظومة angle – المشار إليها سابقاً – تقوم متتابعة عبارات الإسناد التالية

```
angle [0] = 4.93;
angle [1] = -15.2;
angle [2] = 0.5;
angle [3] = 1.67;
```

بملء مركبات المنظومة واحدة واحدة (انظر شكل ٨–٢).

	angle
angle [0]	4.93
angle [1]	-15.2
angle [2]	0.5
angle [3]	1.67

شكل ٨–٢

المنظومة angle مع القيم

ويمكننا أن نعامل أي عنصر من عناصر / مركبات المنظومة – كالعنصر angle [2] مثلاً – بالضبط كما نعامل أي متغير بسيط من النوع float. فمثلاً يمكننا أن نقوم بإجراء العمليات التالية على العنصر angle [2]:

إسناد قيمة له	angle [2] = 9.6;
قراءة قيمة فيه	cin >> angle [2];
طباعة محتوياته	cout << angle [2];
تمريره كوسيط	y = sqrt (angle [2]);
استخدامه في تعبير حسابي	x = 6.8 * angle [2] + 7.5;

وكما ذكرنا سابقاً فيمكن لتعبير المؤشر أن يأخذ صيغاً أخرى غير الصيغة البسيطة: ثابت (constant). فمثلاً نفرض أننا قد أعلنا عن منظومة من ألف عنصر ذات قيم صحيحة بالعبارة:

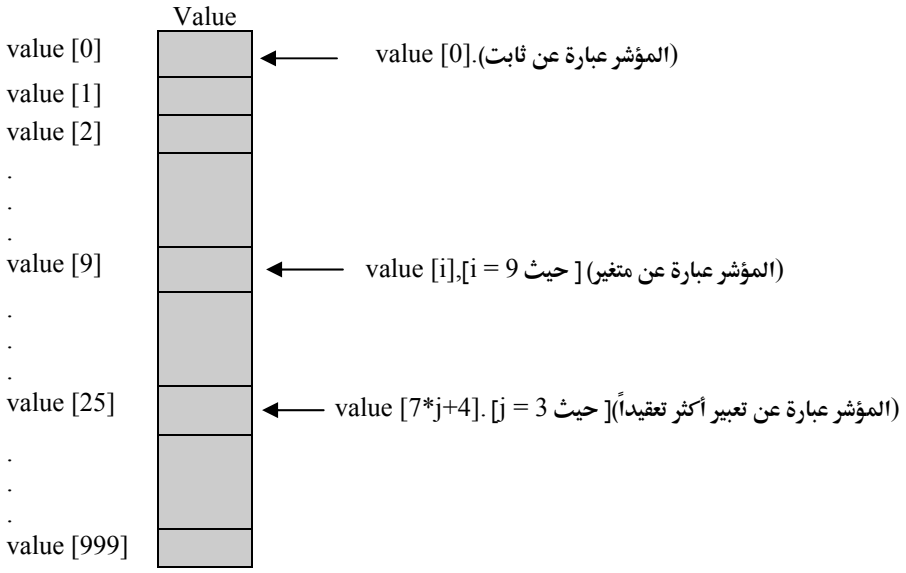
```
int values [1000];
```

ونفرض أننا سنقوم بتنفيذ العبارتين:

```
value [counter] = 5;
if (value [number +1] ٪ 10 ! = 0)
        ⋮
```

في العبارة الأولى يتم تخزين القيمة 5 في أحد عناصر المنظومة . فإن كانت قيمة counter تساوي 0 فإنه القيمة 5 تُخزن في العنصر الأول في المنظومة، وإن كانت قيمة counter تساوي 1 فيتم تخزين 5 في الموضع الثاني في المنظومة ، ... وهكذا. وأما في العبارة الثانية فإن التعبير number + 1 يختار أحد عناصر المنظومة، ثم تُختبر قيمة محتويات هذا العنصر المختار ، بأن تُقسم على 10 ونرى إن كان الباقي لا يساوي صفراً. فإن كانت قيمة number +1 تساوي 0 فإننا نختبر القيمة الموجودة في العنصر الأول في المنظومة ، وإن كانت قيمة number +1 تساوي 1 فإننا نختبر القيمة الموجودة في الموضع الثاني في المنظومة ، ... وهكذا.

ويبين شكل ٨-٣ تعبير المؤشر ثابتاً ومتغيراً وتعبيراً أكثر تعقيداً.



شكل ٨-٣

المؤشر كثابت وكمتغير وكتعبير اختياري

ملاحظة: سبق أن ذكرنا أن طبقة string (class) تسمح لنا بالوصول إلى أي رمز في سلسلة رموز ، مثلاً :

```
string   aString;

aString = "Salam";
cout <<   aString [3];            // Prints 'a'
```

فرغم أن string طبقة وليس منظومة، إلا أن طبقة string قد كُتبت باستخدام أسلوب C++ المتقدم (advanced C++ technique) للتحميل الزائد للمؤثر ات (operator overloading) لإعطاء المؤثر operator [ ] معنى آخر [اختيار عنصر / مركبة في سلسلة رموز (string component selection)] بالإضافة إلى معناه المعتاد / القياسي [اختيار عنصر في منظومة (array element selection)] والنتيجة هي أن هدف سلسلة الرموز string object يصبح شبيهاً (similar) بمنظومة رموز (array of characters) إلا أنه له بعض الخواص الخاصة (special properties).

المؤشرات الواقعة خارج الحدود

Out-of-Bounds Array Indexes

نفرض أن لدينا الإعلان

```
float alpha [100];
```

وهذا يعني أن مدى قيم المؤشر هو من ٠ إلى 99. فماذا يحدث إذا نفذنا العبارة

```
alpha [i] = 62.4;
```

حيث i إما أصغر من 0 أو أكبر من 99؟

النتيجة هي أنه يتم الوصول إلى موضع في الذاكرة (memory location) خارج المنظومة (outside the array). ولغة C++ لا تختبر عدم وقوع المؤشر خارج المدى (invalid index) سواء وقت الترجمة (compile time) أو وقت التنفيذ (run time). فمثلاً إذا كانت قيمة i تساوي 100 في عبارة الإسناد السابقة فإن الحاسب يقوم بتخزين القيمة 62.4 في موضع الذاكرة التالي الذي يلي نهاية المنظومة ، ويمحو أي قيمة كانت مخزونة في هذا الموضع. ولذا فهي مسئولية

المبرمج أن يتأكد من أن مؤشر المنظومات لا يتجاوز أيا من نهايتي المنظومة. وعادة تستخدم خوارزميات تشغيل المنظومة عرى for للمرور على عناصر المنظومة واحدة واحدة. فمثلاً تقوم العروة التالية بإسناد أصفار للمائة عنصر في المنظومة alpha (حيث i متغير صحيح int):

```
for ( i = 0;    i < 100;   i ++ )
        alpha [i] = 0.0;
```

ويمكننا كذلك كتابة السطر الأول هكذا

```
for ( i = 0;    i <= 99;   i ++ )
```

وعموماً يفضل مبرمجوا C++ استخدام الصيغة الأولى حيث يكون العدد الذي يظهر في اختبار العروة (100) هو نفسه حجم المنظومة. وفي هذه الحالة يكون معامل المقارنة هو < وليس =< .

إعطاء المنظومات قيماً ابتدائية في الإعلانات

Initializing Arrays in Declarations

ذكرنا سابقا أن لغة C++ تسمح لنا بأن نعطي قيمة ابتدائية لمتغير في الإعلان عنه ، مثل :

```
int delta = 25;
```

ويطلق على القيمة 25 "القيمة الابتدائية / المبدئة" (initializer) . وكذلك يمكننا أن نعطي منظومة ما قيمة ابتدائية عند الإعلان عنها، وذلك بوضع قائمة القيم الابتدائية لعناصر المنظومة بين قوسين { } ، مع وضع فاصلة بين كل قيمتين متتاليتين، كأن نكتب مثلاً:

```
int age [5] = {23, 10, 16, 37, 12};
```

ففي هذا الإعلان يعطَى العنصر age [0] القيمة الابتدائية 23، ويعطى العنصر age[1] القيمة الابتدائية 10، ... وهكذا. ويجب أن تكون هناك قيمة ابتدائية واحدة على الأقل بين القوسين { } . وإذا وضعنا قيماً أكثر من المطلوب ظهرت لنا رسالة خطأ تركيبي (synatx error message). بينما إذا وضعنا قيماً أقل من المطلوب فإن عناصر المنظومة المتبقية تأخذ قيماً ابتدائية صفرية.

وتتبع المنظومات القاعدة نفسها الخاصة بالمتغيرات البسيطة (simple variables) من حيث الوقت / الأوقات التي يتم عندها إعطاء القيم الابتدائية. فالمنظومة الاستاتية / الاستاتيكية (static array) [وهي إما المنظومة الشاملة (global) أو المنظومة المعلن عنها أنها static داخل أحد القوالب (within a block)] تُعطى قيمة ابتدائية مرة واحدة فقط، وذلك عندما يصل التحكم (control) الإعلان عنها. بينما المنظومة الاوتوماتيكية / الذاتية (auomatic array) [وهي المنظومة المحلية (local) التي لم يُعلن عنها أنها static] يعاد إعطاء قيمة ابتدائية لها كلما وصل التحكم إلى الإعلان عنها.

ومن خصائص لغة C++ الجديرة بالاعتبار أنه يمكننا أن نحذف حجم المنظومة عندما نعطيها قيماً ابتدائية عند الإعلان عنها ، فنكتب مثلاً:

```
float temperature [ ] = { 0.0,  112.37, 98.6};
```

ويستطيع المترجم (compiler) أن يحدد عدد عناصر المنظومة أي حجمها [وهو هنا 3] بناء على عدد القيم الابتدائية المذكورة في قائمة هذه القيم. وعموماً هذه الخاصية ليست مفيدة ، إلا أنها يمكن أن تكون مناسبة لإعطاء قيم ابتدائية لأنواع خاصة من منظومات رموز (char arrays) يطلق عليها سلاسل رموز C (C string).

العمليات الإجمالية على المنظومات
Aggregate Array Operations

تسمح بعض لغات البرمجة بإجراء عمليات إجمالية على المنظومات ، ولكن لغة C++ لا تسمح عموماً بذلك. فمثلاً إذا كان كل من x, y منظومة وكان لدينا الإعلانان

```
int   x [50];
int   y [50];
```

فكل من العمليات الإجمالية التالية خاطئة:

(١) العملية الإجمالية لإسناد عناصر y لعناصر x (aggregate assignment):

```
x = y;                    // Not valid
```

فلنسخ المنظومة y في المنظومة x يجب أن نقوم بنسخ العناصر واحداً واحدا، كأن نكتب مثلاً :

```
for  (index = 0;        index < 50;   index ++)
     x [index] = y [index];
```

(٢) عملية المقارنة الإجمالية (aggregate comparison) لاختبار تساوي منظومتين:

```
if ( x = = y )                    // Not valid
```

(٣) عمليات الإدخال / الإخراج الإجمالية aggregate I / O للمنظومات:

```
cout << x;                        // Not valid
```

[استثناء: يسمح بعمليات I / O الإجمالية بالنسبة لسلاسل رموز C (C string) والتي تعد أنواعاً خاصة من منظومات رموز (char arrays)]

(٤) العمليات الحسابية الإجمالية (aggregate arithmetic) على المنظومات، مثل:

```
x = x + y ;                       // Not valid
```

(٥) إعادة منظومة بأكملها (return an entire array) كقيمة دالة تعيد قيمة (value-returning function):

```
return x;                         // Not valid
```

وأما العملية الإجمالية الوحيدة التي يمكن إجراؤها على المنظومات فهي تمرير منظومة كوسيط فعلي (argument) لدالة ، مثل:

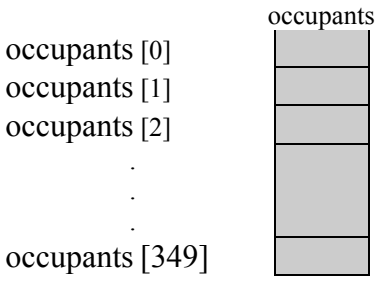
```
DoSomething (x);
```

وتمرير المنظومة كوسيط فعلي يسمح للدالة بالوصول إلى جميع عناصر المنظومة.

أمثلة للإعلان عن المنظومات والوصول إلى عناصرها

Examples of Declaring and Accessing Arrays

مثال ٨–٣: لفرض أن occupants **منظومة من ٣٥٠ عنصر عبارة عن أعداد صحيحة** (integers) **تمثل أعداد ساكني شقق** (apartments) **عمارة سكنية** (apartment building) **مكونة من ٣٥٠ شقة [انظر شكل ٨–٤].**



شكل ٨–٤

المنظومة occupants

فيما يلي بعض الإعلانات التي يمكن لبرنامج أن يستخدمها في مسألة تحليل إيجارات السكن (occupancy rates)**.**

```
const int BUILDING_SIZE = 350;      // Number of apartments

int occupants [BUILDING_SIZE];      // occupants [i] is the number of
                                    // occupants in apartment i
int totalOccupants;                 // Total number of occupants
int counter;                        //Loop control and index variable
```

اكتب قطعة برنامج لإيجاد العدد الإجمالي لساكني العمارة.

الحل: نلاحظ أنه إذا فرضنا أن عدد ساكني الشقة الأولى يساوي 3 **مثلاً، فمعنى ذلك أن**

occupants [0] = 3

وإذا كان عدد ساكني الشقة الثانية يساوي 5**، فيكون**

occupants [1] = 5

وهكذا. وإذا تم تخزين عدد ساكني الشقق كلها في المنظومة occupants **فإن قطعة البرنامج التالية توحد العدد الإجمالي لساكني العمارة:**

```
totalOccupants = 0;
for (counter = 0;  counter < BUILDING_SIZE;  counter ++)
    totalOccupants = totalOccupants + occupants [counter];
```

لاحظ كيف أننا استخدمنا الثابت المسمَّى BUILDING_SIZE في كل من الإعلان عن المنظومة وفي عروة FOR، وكما ذكرنا سابقاً فعندما تستخدم الثوابت بهذه الطريقة يسهل عمل أي تغييرات أو تعديلات. فمثلاً إذا زاد عدد الشقق من 350 إلى 400 ، فكل ما نحتاج تعديله هو سطر واحد فقط وهو الإعلان عن الثابت const المسمَّى BUILDING_SIZE . بينما إذا كنا قد استخدمنا القيمة الحرفية 350(literal value) مكان BUILDING_SIZE في الإعلان عن المنظومة وفي عروة FOR وفي غيرهما، فكنا سنحتاج إلى تعديل عدة عبارات في البرنامج.

* * *

ونظراً لأن مؤشر المنظومة (array index) قيمة صحيحة (integer value) فلذلك يمكننا الوصول إلى عناصر المنظومة عن طريق مواضع (positions) هذه العناصر: العنصر الأول فالثاني فالثالث، ...وهكذا. واستخدام مؤشر صحيح (int index) هو أكثر الوسائل شيوعاً عند التفكير في المنظومات . إلا أن لغة C++ تسمح بمرونة أكثر (more flexibility) حيث تسمح للمؤشر بأن يكون من أي نوع تكاملي (integral type) أو من النوع التعددي (enumeration type) . [إلا أن تعبير المؤشر (index expression) يجب أن تُحسب له قيمة صحيحة في المدى من 0 إلى (حجم المنظومة – ١)] . المؤشرات في المثال التالي عبارة عن قيم من النوع التعددي.

مثال ٨-٤: نفرض أن لدينا الإعلانات التالية:

```
enum Drink {ORANGE, COLA, APPLE,  GINGER_ALE , CHERRY,   LEMON};

float salesAmt [6];       // Array of 6 floats, to be indexed by Drink type
Drink flavor;             // Variable of the index type
```

حيث Drink نوع تعددي فيه القيم المعددة (enumerators):

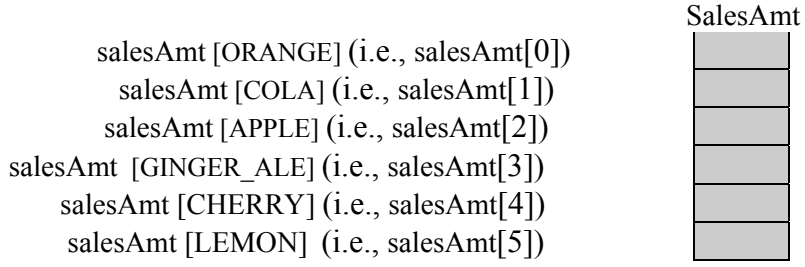
```
ORANGE,  COLA,  ……..,  LEMON
```

والتي لها التمثيل الداخلي (internal representation):

0, 1, …. , 5

على الترتيب.

والمنظومة salesAmt **عبارة عن مجموعة** (group) **مكونة من ٦ عناصر / مركبات من النوع** float **تمثل قيم المبيعات بالدينار لكل نوع** (kind) **من المشروبات** (drink) **[انظر شكل ٨–٥].**



شكل ٨–٥

المنظومة salesAmt

اكتب قطعة برنامج تطبع القيم المخزونة في المنظومة salesAmt **، أي تطبع قيم المبيعات بالدينار لجميع المشروبات.**

الحل:

```
for (flavor = ORANGE; flavor <= LEMON; flavor = Drink (flavor +1))
    cout << salesAmt [flavor] << endl;
```

مثال ٨–٥: نفرض أن لدينا الإعلانات التالية:

```
const int NUM_STUDENTS = 10;

char grade (NUM_STUDENTS];    // Array of 10 student letter grades
int   idNumber;               // Student ID number (0 through 9)
```

الشكل التالي (شكل ٨–٦) يبين المنظومة grade **والقيم المخزونة في عناصرها.**

	grade
Grade [0]	'F'
Grade [1]	'B'
Grade [2]	'C'
Grade [3]	'A'
Grade [4]	'F'
Grade [5]	'C'
Grade [6]	'A'
Grade [7]	'A'
Grade [8]	'C'
Grade [9]	'B'

شكل ٨-٦

المنظومة grade مع قيم عناصرها

فيما يلي بعض الأمثلة لعبارات بسيطة توضح كيفية استخدام المنظومة والوصول إلى عناصرها.

cin >> grade [2]; **اقرأ من سيل المدخلات** (input stream) **أول رمز تالي من غير الرموز الفراغية البيضاء** (next nonwhitespace character) **وقم بتخزينه في مركبة** grade **المؤشرة / المرقمة** 2 (indexed by).

grade [3] = 'A'; **أسند الرمز** 'A' **لمركبة** grade **المرقمة** 3.

idNumber = 5; **أسند القيمة** 5 **لمتغير التأشير** idNumber (index variable).

grade [idNumber] = 'C'; **أسند الرمز** 'C' **لمركبة** grade **المرقمة** idNumber **(أي المرقمة** 5**).**

ما نتيجة تنفيذ كل من عروتي for **التاليتين؟**

(أ)
```
for (idNumber = 0;   idNumber < NUM_STUDENTS;
    idNumber ++)
    cout << grade [idNumber];
```

(ب)
```
for (idNumber = 0;   idNumber < NUM_STUDENTS;
```

```
idNumber ++)
cout << " Student " << idNumber
      << " Grade " << grade [idNumber];
      << endl;
```

الحل:

(أ) تقوم العروة بطباعة محتويات جميع عناصر المنظومة grade. **أي أن مخرجات العروة هي:**

FBCAFCAACB

(ب) في هذه العروة نلاحظ أن idNumber **يستخدم كمؤشر** (index)**، وفي الوقت نفسه محتوياته تمثل الرقم التعريفي للطالب** (student's identification number) **. والعروة تطبع أيضاً محتويات جميع عناصر المنظومة ، ولكن بصورة أكثر تفصيلاً وأوضح في القراءة** (more readable form) **. ومخرجات العروة هي:**

```
Student 0 Grade F
Student 1 Grade B
    ⋮
Student 9 Grade B
```

تمرير المنظومة كوسيط فعلي

Passing Array as an Argument

علمنا سابقاً من فصل "الدوال" (الفصل الخامس) أنه إذا تم تمرير متغير لدالة، وأردنا ألا تغير الدالة قيمة هذا المتغير فيجب أن يمرر هذا المتغير بالقيمة (passed by value)**وليس بالإسناد** (by reference)**. وتُستثنى من هذه القاعدة متغيرات السيل** (stream variables) **[كالمتغيرات التي تمثل ملفات البيانات** (data files) **] ، كما تستثنى منها المنظومات.**

فالمتغيرات البسيطة في لغة C++ يُفترض (by default) دائماً أنها تمرر بالقيمة. ولتمرير متغير بسيط بالإسناد يجب أن نُلحق (append) العلامة (&) ampersand باسم نوع البيانات في قائمة وسطاء الدالة:

```
int SomeFunc ( float param1,          // Passs-by-value
               char& param2)          // Pass-by-reference
{
    ⋮
}
```

ومن المستحيل أن نمرر أي منظومة C++ بالقيمة ، فجميع المنظومات تمرر دائماً بالإسناد. ولذلك فلا نستخدم إطلاقاً العلامة (&) عند الإعلان عن منظومة كوسيط (parameter) . وحينما تُمرَّر منظومة كوسيط فعلي (argument) فإن عنوانها الأساسي (its **base address**) [وهو عنوان أول عنصر من عناصر المنظومة في الذاكرة (memory address of the first element of the array) يُرسَل للدالة ، فتعرف الدالة عندئذ موضع (location) المنظومة الفعلية في الدالة المستدعية (caller's actual array) ، وبالتالي تستطيع الوصول (access) لأي عنصر من عناصر المنظومة.

مثال ٨-٦: اكتب دالة ZeroOut تقوم بإسناد أصفار لجميع عناصر منظومة أحادية البعد arr من النوع float وذات أي سعة numElements (of any size).

الحل:

```
void ZeroOut ( /* out */ float arr [ ],
               /* in */ int   numElements )
{
    int i;

    for (i = 0; i < numElements; i ++)
        arr [i] = 0.0;
}
```

لاحظ أن الإعلان عن المنظومة arr في قائمة الوسطاء لا يحتوي على سعة (size) معينة بين القوسين [ ] . وإذا وضعنا أي سعة فإن المترجم (compiler) سيتجاهلها .

فكل ما يريد المترجم معرفته هو أنها منظومة من النوع float وليس منظومة من النوع float من سعة معينة ولذلك ففي الدالة ZeroOut يجب أن نضع وسيطاً ثانياً – هو عدد عناصر المنظومة – كي تعمل عروة for بصورة سليمة.

وأما الدالة المستدعية فيمكنها استدعاء الدالة ZeroOut لمنظومة من النوع float ذات أي سعة. وقطعة البرنامج التالية تستدعي الدالة ZeroOut مرتين لإسناد أصفار لجميع عناصر منظومتين مختلفتي السعة (to zero out two arrays of different sizes). ولاحظ كيف نعلن عن منظومة كوسيط في نموذج دالة (a function prototype).

```
void ZeroOut ( float [ ],  int );              // Function prototype
⋮
int main ( )
{
      float velocity [30];
      float refractionAngle [9000];
      ⋮
      ZeroOut (velocity, 30);
      ZeroOut (refractionAngle, 9000);
      ⋮
}
```

* * *

كما نعلم بالنسبة للمتغيرات البسيطة فإن تمريرها بالقيمة يمنع الدالة من تعديل (modifying) قيمة الوسيط الفعلي في الدالة المستدعية (caller's argument) . ورغم أننا لا نستطيع تمرير المنظومات بالقيمة في لغة C++، إلا أنه لا يزال بإمكاننا منع الدالة من تعديل منظومة الدالة المستدعية، وذلك عن طريق استخدام الكلمة المحجوزة **const** في الإعلان عن الوسيط ، كما يوضح ذلك المثال التالي:

مثال ٨–٧: اكتب دالة Copy تنسخ منظومة صحيحة source (int array) سعتها (أي عدد عناصرها) size في منظومة أخرى destination.

الحل:

```
void Copy ( /* out */          int destination [ ],
            /* in */     const int source [ ],
            /* in */           int size                )
{
      int i;

      for (i = 0;  i < size; i ++)
           destination [i] = source [i];

}
```

في هـذه الدالـة نلاحـظ أن الوسـيط الأول (المنظومـة destination)**مـن المتوقع أن يتم تغييره / تعديله** (to be modified) **، بينما الوسيط الثاني (المنظومة** source**) لا نتوقع أي تعديل له، ولذلك استخدمنا معه الكلمة** const **التي تضمن لنا أن أي محاولة لتعديل المنظومة** source **داخل الدالة** Copy **تـؤدي إلى خطأ وقت الترجمة** (compile - time error).

وعموماً إذا كانـت أي منظومـة وسيطاً (array parameter) **داخـلاً فقـط** (incoming only) **فـنعلن** (declare) **أن هـذا الوسـيط** const **لنمنـع الدالـة مـن تعـديل الوسـيط الفعلي** (argument) **في الدالة المستدعية بطريق الخطأ.**

الجـدول التالـي يلخـص مـرور الوسـيط الفعلـي (argument passage) **لكـل مـن المتغيـرات البسيطة** (simple variables) **والمنظومـات أحاديـة البعـد** (one-dimensional arrays).

الوسيط الفعلي Argument	الإعلان عن وسيط لتمرير بالقيمة Parameter Declaration for a Pass by Value	الإعلان عن وسيط لتمرير بالإسناد Parameter Declaration for a Pass by Reference
متغير بسيط	int cost	int & price
منظومة	مستحيل (*)	int arr [ ]

(*) ولكن وضع كلمة const قبل الإعلان عن المنظومة يمنع الدالة من تعديل الوسيط (modifying the parameter).

تنبيه: من الأخطاء الشائعة في موضوع تمرير الوسيط الفعلي (argument passage) تمرير عنصر منظومة (to pass an array element) بدلاً من منظومة بأكملها (an entire array) للدالة. فمثلاً الدالة ZeroOut في مثال ٨-٦ تتوقع أن يُرسَل العنوان الأساسي (base address) لمنظومة من النوع float كوسيط فعلي أول (first argument). ولذلك فاستدعاء الدالة في قطعة البرنامج التالية استدعاء خاطئ:

```
float velocity [30];
⋮
ZeroOut (velocity [30],  30);      // Error
```

أولاً : لأن velocity [30] يرمز إلى عنصر منظومة وحيد (single array element) – عدد واحد ذو نقطة عائمة – وليس منظومة بأكملها.

ثانياً: ليس هناك عنصر منظومة رقمه 30، فأرقام عناصر المنظومة velocity تقع في المدى من 0 إلى 29.

اصطلاح النقطتين ".." (في التعليقات) Notation

في التعليقات على العمليات التي تجرى على المنظومات نستخدم أحياناً اصطلاح النقطتين المتجاورتين ".." ليعني: في المدى من .. إلى ..، كأن نكتب مثلاً :

```
// Assert : velocity [i].. velocity [j] have been printed
```

أو بصورة أكثر اختصاراً

```
// Assert : velocity [i..j] have been printed
```

ونقصد أن العناصر المتتالية من velocity [i] إلى velocity [j] قد تم طباعتها.

واصطلاح النقطتين هذا (dot - dot notation) ليس من تركيبة عبارات لغة C++، وإنما يستخدم في التعليقات فقط وفيما يلي مثال آخر لاستخدام هذا الاصطلاح.

مثال ٨-٨: فيما يلي نعيد كتابة الدالة ZeroOut (دالة مثال ٨-٦) حيث نضيف لها الشروط السابقة (preconditions) والشروط اللاحقة (postconditions) ونستخدم اصطلاح النقطتين:

```
void ZeroOut ( /* out */ float arr [ ],
                /* in */ int numElements )
// Precondition:
//      numElements is assigned
// Postcondition:
//      arr [0.. numElements-1] = = 0.0

{
      int i;

      for (i = 0; i < numElements; i ++)
           arr [i] = 0.0;
}
```

استخدام عبارة Typedef مع المنظومات

Using Typedef with Arrays

يمكننا استخدام عبارة Typedef لإعطاء اسم لنوع منظومة (array type) . فمثلاً عبارة:

```
typedef float FloatArr [100];
```

تعني أن النوع (type) FloatArr هو نفسه النوع :

"100-element array of float" type

(ولاحظ أن حجم المنظومة الموضوع بين قوسين يأتي في نهاية العبارة أقصي اليمين)

بمعنى أنه يمكننا الآن الإعلان عن متغيرات من النوع FloatArr كما يلي:

```
FloatArr  angle;
FloatArr  velocity;
```

ويقوم المترجم (compiler) أساساً بترجمة هذه الإعلانات إلى ما يلي:

```
float  angle [100];
float  velocity [100];
```

وعموماً قليلاً ما تستخدم عبارة Typedef لإعطاء أسماء لأنواع المنظومات أحادية البعد.

* * *

ونختم هذا الفصل بالمثال التالي:

مثال ٨-٩:

اكتب برنامجاً لقراءة قائمتين (two lists) من الأعداد الصحيحة الموجبة (positive integers) من ملف بيانات (data file) بحيث أنه يوجد عدد صحيح سالب (negative integer) يفصل القائمتين عن بعضيهما البعض (أي أنه موجود بينهما). ويقارن البرنامج بين القائمتين، فإن كانتا متطابقتين (identical) فإنه يطبع رسالة بذلك، وإن كانتا غير متطابقتين فإنه يطبع قيم الأزواج المختلفة / غير المتطابقة (nonmatching pairs) مع بيان مواضعها. ونفرض أن عدد القيم في أي من القائمتين هو نفسه، ولا يتجاوز 500 قيمة.

وفيما يلي مثالان لتشغيل البرنامج (Two sample runs)، حيث القائمتان في المثال الأول متطابقتان ، و في المثال الثاني غير متطابقتين.

مجموعة البيانات الأولى **Data Set 1**	مجموعة البيانات الثانية **Data Set 2**
21	21
32	32
76	76
22	22
21	21
-4	-4
21	21
32	32
76	176
22	12
21	21

المخرجات **Output**	المخرجات **Output**
The two lists are identical	Position 2: 76 != 176 Position 3: 22 != 12

الحل:

```
// ********************************************************************
// CheckLists program
// There are two lists of positive integers in a data file,
// separated by a negative integer. This program compares the two
// lists. If they are identical, a message is printed. If not,
// nonmatching pairs are printed. Assumption: The number of values
// in both lists is the same and is <= 500
// ********************************************************************
#include <iostream>
#include <iomanip>          // For stew ( )
#include <fstream>          // For file I/O
#include <string>           // For string class
```

```
using namespace std;

const int MAX_NUMBER = 500;          // Maximum in each list

void CompareLists ( const int [ ], int, ifstream& bool&  );
void OpenForInput ( ifstream&  );
void ReadFirstList ( int [ ] , int&, ifstream&  );

int main ( )
{
      int          firstList {MAX_NUMBER};    // Holds first list
      bool         allOK;                     // True if lists are identical
      int          numVals;                   // No. of Values in first list
      ifstream     dataFile;                  // Input file

      OpenForInput (dataFile) ;
      If (  !dataFile  )
          return 1;

      ReadFirstList (firstList, numVals, dataFile);
      allOK = true;
      CompareLists (firstList, numVals, dataFile, allOK);
      If (allOK)
                cout << "The two lists are identical" << endl;
      return 0;
}




// ********************************************************************
void OpenForInput( /* inout */ ifstream& someFile )    // File to be
                                                       //opened
 //Prompts the user for the name of an input file
 //and attempts to open the file
{
          .
          .          (Same as  in ConvertDates program Chapter 5)
          .
}

// ********************************************************************
```

```
void ReadFirstList(
        /* out */   int       firstList[],          // Filled first list
        /* out */   int&      numVals,              // Number of values
        /* inout */ ifstream& dataFile    )        // Input file

 //Reads the first list from the data file
 //and counts the number of values in the list
{
   int counter;             // Index variable
   int number;              // An input value
   counter = 0;
   dataFile >> number;
   while (number >= 0)
   {
        firstList [counter] = number;
        counter++;
        dataFile >> number;
   }
   numVals = counter;
}

//*****************************************************************
void CompareLists)
        /* in */    const int       firstList[],        // 1st list of numbers
        /* in */          int       numVals,            // Number in 1st list
        /* inout */       ifstream& dataFile,     // Input file
        /* inout */       bool&     allOK       )   // True if lists match

 //Reads the second list of numbers
 //and compares it to the first list
{
   int counter;    // Loop control and index variable
   int number;     // An input value

   for (counter = 0; counter < numVals; counter++)
   {
      dataFile >> number;
      if (number != firstList[counter])
   }
         allOK = false;
```

```
        cout << "Position " << counter << " : "
             << setw(4) << firstList[counter ] << " != "
             << setw(4) << number << endl;
}
```

ملاحظة: يسمح بحذف سعة المنظومة أحادية البعد في الإعلان عنها في حالتين فقط:

(١) عندما نعلن عن المنظومة كوسيط (parameter) **في عنوان دالة** (function heading).

(٢) عندما نعطي المنظومة قيماً ابتدائية في الإعلان عنها.

وفي جميع الإعلانات الأخرى يجب تحديد سعة المنظومة بتعبير صحيح ثابت.

تمرينات رقم ٨

٨-١) اكتب عنوان دالة خاوية اسمها SomeFunc لها:

(أ) وسيط وحيد عبارة عن منظومة x من النوع float/ و الوسيط داخل وخارج(Inout parameter) .

(ب)وسيط وحيد عبارة عن منظومة x من النوع float ، و الوسيط داخل فقط (In parameter) .

٨-٢) نفرض أن لدينا الإعلانات

```
const  int SIZE = 30;
char   firstName [SIZE] ;
```

(أ) اكتب عبارة إسناد تقوم بتخزين 'A' في المركبة الأولى من المنظومة firstName.

(ب)اكتب عبارة تطبع قيمة العنصر الرابع عشر من المنظومة firstName.

(ج) اكتب عبارة For تملأ المنظومة firstName بفراغات (blanks).

(د) قطعة البرنامج التالية تظل تقرأ رموزاً وتخزنها في المنظومة firstName إلى أن تصادف فراغاً (balnk).

```
n = 0;
cin.get (letter);
while ( letter != ' ' )
{
        firstname [n] = letter;
        n ++;
        cin.get (letter);
}
```

اكتب عبارة For تطبع جزء المنظومة الذي امتلأ بالبيانات المدخلة (inputdata).

٨-٣) أذكر صحة أو خطأ كل من العبارات التالية:

(أ) كل مركبة من مركبات المنظومة يجب أن تكون من النوع نفسه، وعدد المركبات يحدَّد (fixed) وقت الترجمة (compile time).

(ب) مركبات أي منظومة يجب أن تكون من نوع تكاملي أو من نوع تعددي (integral or enumeration type).

(ج) المنظومة أحادية البعد تُعد مثالاً لنوع البيانات المبني (structured data type).

(د) في لغة C++ لا يمكن أن يكون نوع مركبات المنظومة أحادية البعد أياً من : float, double, long double

(هـ) في لغة C++ نوع مؤشر (index type) المنظومة أحادية البعد يمكن أن يكون أي نوع تكاملي (integral type).

(و) يمكن تمرير مركبة مفردة (individual component) من مركبات منظومة كوسيط (parameter) لدالة.

(ز) في لغة C++ يمكن تمرير المنظومة كوسيط إما بالقيمة (by value) أو بالإسناد (by reference).

(ح) إذا كان لدينا الإعلان

```
int beta [20];
```

فإن

(i) التعبير beta [3] يؤدي إلى الوصول إلى العنصر الثالث في المنظومة .

(ii) العبارة cout << beta لا يمكن استخدامها لطباعة جميع عناصر المنظومة العشرين.

(iii) العبارة `beta = beta + 1;` تضيف 1 لكل عنصر من عناصر المنظومة العشرين.

(ط) إذا اشتمل برنامج على الإعلانين

```
enum WeatherType {SUNNY, CLOUDY, FOGGY, WINDY};
int  frequency [4];
```

فإن العبارة

```
cout << frequency [CLOUDY];
```

تعد صحيحة من الناحية التركيبية (syntactically valid)

(ي) لغة C++ لا تتحقق من أن مؤشرات المنظومة (array indices) لا تتجاوز حدودها (out-of-bounds) أثناء تشغيل (running) البرنامج.

(ك) عنوان الدالة

```
void someFunc (  float x [ ] )
```

يتسبب في ظهور خطأ وقت الترجمة (compile-time error) لأن سعة المنظومة غير مذكورة.

(ل) إذا وضعنا كلمة const قبل الإعلان عن منظومة في عنوان دالة (function heading) فإن الدالة تُمنع من تعديل المنظومة .

(م) العبارة

```
if (score [studentID] >= 90)
   letterGrade [studentID] = 'A';
```

تعد مثالاً لاستخدام المنظومات المتوازية (parallel arrays)

(ن) إذا احتوى برنامج على الإعلانين

```
typedef  char  NameString [31];
NameString  studentName [20];
```

فإن العبارة `cin >> studentName [2];`

تعد عبارة صحيحة (valid).

٨-٤) اكتب الإعلانات عن المنظومات أحادية البعد التالية

(أ) منظومة float مكونة من ٢٤ عنصر.

(ب) منظومة int مكونة من ٥٠٠ عنصر.

(ج) منظومة ذات دقة مضاعفة من النقطة العائمة (double precision floating point array) مكونة من ٥٠ عنصر.

(د) منظومة رموز من ١٠ عناصر.

٨-٥) اكتب قطعة برنامج تقوم بما يلي:

(أ) تعلن عن ثابت يُدْعَى CLASS_SIZE يمثل عدد الطلاب في أحد الفصول.

(ب) تعلن عن منظومة أحادية البعد quizAvg سعتها CLASS_SIZE تحتوى مركباتها على متوسطات درجات الاختبار وهي من النوع floating-point.

٨-٦) اكتب قطعة برنامج تقوم بما يلي:

(أ) تعلن عن نوع تعددي BattleType مكون أسماء غزوات / معارك (battle names).

(ب) تعلن عن منظومة int أحادية البعد sightings يتم ترقيمها / الإشارة إليها (to be indexed) بواسطة BattleType.

٨-٧) نفرض أن لدينا الإعلانات التالية:

```
const int SIZE = 100;

enum Colors
{
   BLUE, GREEN, GOLD, ORANGE, PURPLE, RED, WHITE, BLACK
};
int       count [8];
Colors cIndex;                    // Index for count array
Colors rainbow [SIZE];
int       rIndex;                 // Index for rainbow array
```

اكتب قطع برامج تقوم بإجراء العمليات التالية:

(أ) إسناد أصفار لجميع عناصر المنظومة count.

(ب) إسناد WHITE لجميع عناصر المنظومة rainbow .

(ج) حساب عدد مرات ظهور GREEN في المنظومة rainbow.

(د) طباعة القيمة المخزونة في count والتي تشير إليها BLUE(indexed by).

(هـ) إيجاد مجموع القيم المخزونة في count.

٨-٨) ما هي مخرجات البرنامج التالي؟ بيانات البرنامج معطاة أسفله.

```
# include <iostream>
```

```
using namespace std;

int main ( )
{
        int a [100];
        int b [100];
        int j;
        int m;
        int sumA = 0;
        int sumB = 0;
        int  sumDiff = 0;

        cin >> m;
        for (j = 0;  j < m;        j++)
        {
                cin >> a [j] >> b [j];
                sumA = sumA + a [j];
                sumB = sumB + b [j];
                sumDiff = sumDiff + (a[j] - b[j]);
        }
        for (j = m -1;   j >= 0;  j--)
                cout << a [j] << ' ' << b [j] << ' '
                     << a [j] - b[j] << endl;
        cout << endl;
        cout << sumA << ' ' << sumB << ' ' << sumDiff << endl;
        return 0;
}
```

Data

```
   5
11  15
19  14
 4   2
17   6
 1   3
```

٨-٩) كتب شخص قطعة البرنامج التالية قاصداً طباعة 40 30 20 10.

```
int arr [4] = {10,  20,  30,  40};
int index;
for (index = 1;  index <= 4;  index ++)
        cout << ' ' << arr[index];
```

ولكن الذي حدث أن القطعة أدت إلى طباعة 24835 40 30 20 .

اشرح سبب ظهور هذه المخرجات.

٨-١٠) نفرض أن لدينا الإعلانات

```
int sample [8];
int i;
int k;
```

ما هي محتويات المنظومة sample بعد تنفيذ كل من قطع البرامج التالية (المنفصلة) ؟ استخدم رمز علامة الاستفهام للدلالة على أي قيمة غير معرَّفة (undefined value) في المنظومة.

(ا)
```
for (k = 0; k < 8; k ++
  sample [k] = 10 – k;
```

(ب)
```
for (i = 0; i < 8; i++)
    if (i <= 3)
        sample [i] = 1;
  else
        sample [i] = -1;
```

(ج)
```
for (k = 0; k < 8; k ++
    if (k % 2 == 0)
        sample [k] = k;
  else
      sample [k] = k + 100;
```

٨-١١) ما هي مخرجات قطع البرامج التالية

(أ)
```
int gamma[3] = {5, 10, 15};
int i;
for (i = 0; i < 3; i++)
   cout << gamma[i] << ' ';
```

(ب)
```
int alpha[5] = {100, 200, 300, 400, 500};
int i;

for (i = 4; i > 0; i--)
    cout << alpha[i] << ' ';
```

(ج)

```
int alpha[5] = {100, 200, 300, 400, 500};
int i;

for (i = 4; i >= 0; i--)
   cout << alpha[i] << ' ';
```

٨-١٢) بعد تنفيذ قطعة البرنامج التالية

```
int arr[5];
int i;

for (i = 0; i < 5; i++)
{
   arr[i] = i + 2;
   if (i >= 3)
      arr[i-1] = arr[i] + 3;
}
```

ما هي محتويات (أ) arr [1] (ب) arr [3]

٨-١٣) ما هي محتويات arr **بعد تنفيذ عروة** for **التالية؟**

```
int j = 0;
      for (i = 0; i < 5; i++)
 {
        arr[i] = i+j;
    j = j + 1;
 }
```

٨-١٤) نفرض أن beta **منظومة مكونة من** 5000 **عنصر. أي عرى** for **التالية يمكن أن تستخدم لطباعة قيم:**

beta [0], beta [2], beta [4], … , beta [4998]

بفرض أن جميع المتغيرات من النوع int**؟**

(أ)

```
for (i = 0; i < 5000; i = i + 2)
            cout << beta[i] << endl;
```

(ب)

```
for (i = 0; i < 2500; i++)
            cout << beta[2*i] << endl;
```

(ج)
```
for (i = 0; i < 2500; i++)
        cout << beta[i]*2 << endl;
```

٨-١٥) نفرض أن alpha منظومة int مكونة من ثلاثة عناصر. أي القطع التالية يمكن أن تستخدم لإدخال قيم في المنظومة alpha؟

(أ)
```
cin >> alpha[0] >> alpha[1] >> alpha[2];
```

(ب)
```
cin >> alpha;
```

(ج)
```
for (i = 0; i < 3; i++)
    cin >> alpha[i];
```

(د)
```
cin >> alpha[0];
cin >> alpha[1];
cin >> alpha[2];
```

* * *

استخدم الإعلانات التالية في التمرينات من ٨-١٦) إلى ٨-٢٢) لكتابة الدوال المطلوبة بلغة C++، ويمكنك الإعلان عن أي متغيرات أخرى تحتاجها.

```
const int MAX_STUD = 100;// Max.Number of students

bool failing [MAX_STUD];
bool passing [MAX_STUD];
int grade;
int score [MAX_STUD];
```

٨-١٦) اكتب دالة تعطي قيماً ابتدائية -جميعها false- لجميع عناصر المنظومة المنطقية failing، حيث المنظومة failing تعد وسيطاً (parameter).

٨-١٧) اكتب دالة تأخذ المنظومة failing ، والمنظومة score كوسيطين. وتقوم الدالة بإعطاء عناصر failing القيمة true كلما كانت القيمة المقابلة في score أقـــل من 60. افرض أن العدد الفعلي للطلاب هو length حيث (length ≤ MAX_STUD).

٨-١٨) اكتب دالة تأخذ المنظومة passing ، والمنظومة scoreكوسيطين. وتقوم الدالة بإعطاء عناصر passing القيمة true حينما تكون القيمة المقابلة في

score أكبر من أو تساوي من 60. افرض أن العدد الفعلي للطلاب هو MAX_STUD.

٨-١٩) اكتب دالة تعيد قيمة ، بحيث أن هذه الدالة PassTally تأخذ المنظومة passing كوسيط ثم تخبرنا (reports) كم عدد العناصر التي قيمتها true في المنظومة passing.

٨-٢٠) اكتب دالة تعيد قيمة ، بحيث أن هذه الدالة Error تأخذ الوسيطين passing, failing وتعيد القيمة true إذا تطابق أي عنصرين متقابلين (corresponding elements) في المنظومتين passing, failing.

٨-٢١) اكتب دالة تعيد قيمة بحيث أن هذه الدالة تأخذ الوسيطين grade, score ، وتعيد عدد قيم score التي هي أكبر من أو تساوي grade.

٨-٢٢) اكتب دالة تأخذ الوسيط score ، ثم تعكس ترتيب العناصر في المنظومة score ، بمعنى أنه إذا كان العدد الفعلي لعناصر المنظومة score هو length (حيث length <= MAX_STUD) ، فإن :

score[0] ينتقل ليصبح score [length -1]

score[1] ينتقل ليصبح score [length -2]

⋮

score [length -1] ينتقل ليصبح score [0]

* * *

٨-٢٣) في برنامج CheckLists (مثال ٨-٩):

أ) تعلن الدالة ReadFirstList عن وسيط اسمه firstList. هل يجوز أن نضع كلمة const قبل الإعلان عن firstList؟ ولماذا؟

ب) يقارن البرنامج بين قائمتين من الأعداد الصحيحة. ما هي التغييرات التي يلزم إجراؤها في البرنامج كي يقارن بين قائمتين من الأعداد ذوات النقطة العائمة (float)؟

ج) عدّل البرنامج بحيث يعمل بصورة صحيحة أيضاً حتى لو اختلفت القائمتان في الطول (أي عدد العناصر) ، أو لو اشتملتا على أكثر من 500 قيمة، مع إعطاء رسالة خطأ مناسبة وإيقاف المقارنة في هذه الحالة.

٨-٢٤) اكتب برنامجا لقراءة مجموعة من درجات الحرارة

$T_1 , T_2 , T_3 , \dots , T_{40}$

وتعيين درجة الحرارة المتوسطة

$$T_{av} = \frac{T_1 + T_2 + \cdots + T_{40}}{40}$$

وانحراف كل درجة T_i عن الدرجة المتوسطة ، حيث يعرف الانحراف بالعلاقة :

$$D_i = T_i - T_{av} , \quad i = 1 , 2 , \dots , 40$$

٨-٢٥) اكتب برنامجا لقراءة قيم عناصر منظومة مكونة من مائة عنصر وحساب الجذر التربيعي لمجموع مربعات العناصر الفردية الترتيب أي لحساب :

$$S = \sqrt{A_1^2 + A_3^2 + A_5^2 + A_7^2 + \cdots + A_{99}^2}$$

٨-٢٦) نفرض أن A منظومة مكونة من أربعين عنصرا

$A_1 , A_2 , \dots , A_i , \dots , A_{40}$

اكتب برنامجا لقراءة قيم هذه العناصر وتعيين قيمة "i" حيث i هي ترتيب أول عنصر سالب في هذه المنظومة ، وإذا لم توجد أي قيمة سالبة فإن البرنامج يطبع القيمة صفرا (i = 0).

٨-٢٧) تتكون المنظومة Y من خمسين عنصرا. اكتب برنامجا لقراءة قيمة عدد صحيح N وقيم أول N عنصر من المجموعة Y:

$Y_1 , Y_2 , Y_3 , \ldots , Y_N$

ثم حساب (أ) القيمة المتوسطة (Average)

$$AVG = \sum_{i=1}^{N} Y_i / N$$

(ب) الانحراف القياسي (Standard Deviation)

$$SD = \sqrt{\frac{\sum_{i=1}^{N} Y_i^2}{N} - AVG^2}$$

٨-٢٨) اكتب برنامجا لقراءة قيم خمسين عنصرا من المنظومة A ثم لحساب ما يلي :

(أ) مجموع الجذور التربيعية لهذه العناصر

$$SMSQRT = \sqrt{A_1} + \sqrt{A_2} + \cdots + \sqrt{A_{50}}$$

(ب) قيمة أكبر عنصر AMAX

(ج) قسمة كل عنصر في المنظومة على AMAX وتخزين هذه القيم (المعيَّرة) في منظومة B.

٨-٢٩) اكتب برنامجا يقرأ قيم خمسة وأربعين عددا ثم يوجد ترتيب وقيمة العدد ذي أكبر قيمة مطلقة.

٨-٣٠) تحسب الدرجة النهائية لطالب بأخذ متوسط أفضل أربع درجات من خمس درجات يحصل عليها في خمسة اختبارات. اكتب برنامجا يقرأ الخمس درجات للطالب ثم يحسب درجته النهائية.

ملاحظة : يمكن طرح أقل درجة من مجموع الخمس درجات للحصول على مجموع أفضل أربع درجات.

٨-٣١) نفرض أن كلا من L , M منظومة مكونة من أعداد صحيحة حيث تحتوي L على ٢٤ عنصرا مختلفا وتحتوي M على ٤٠ عنصرا مختلفا. اكتب برنامجا

لقراءة قيم عناصر المجموعتين ثم طباعة قيم الأعداد الصحيحة التي تظهر في كلا المجموعتين ، أي أن البرنامج يوجد تقاطع المجموعتين.

٨-٣٢) نفرض أن Y المعرفة بالعلاقة

$$Y = F(x) = a_0 + a_1x + a_2x^2 + ... + a_ix^i + ... + a_nx^n$$

هي كثيرة حدود من درجة n في المتغير x.

اكتب برنامجا (أ) يقرأ قيم المعاملات

$$a_0 , a_1 , a_2 , ... , a_n$$

ويخزنها في منظومة ، بحيث أنه يمكن للبرنامج أن يعمل حسابات خاصة بكثيرات حدود لا تزيد درجة أي منها عن (١٥) خمس عشرة.

(ب) يقرأ قيمة للمتغير x ويحسب القيمة المقابلة لكثيرة الحدود.

٨-٣٣) نفرض أن خمسمائة شخص قدموا تبرعات في سبيل الله تعالى ، وأن قيم هذه التبرعات بالدينار قد وضعت كعناصر منظومة D :

$$D_1 , D_2 , ... , D_{500}$$

اكتب برنامجا لقراءة قيم هذه العناصر وحساب :

(أ) مجموع التبرعات SUM

(ب) أكبر قيمة تبرعات BIG

(ج) عدد الأشخاص N الذين تبرع الواحد منهم بأكثر من مائة دينار.

٨-٣٤) نفرض أن كلا من A , B منظومة مكونة من N عنصرا حيث $N \leq 20$.

اكتب برنامجا :

(أ) يقرأ قيم كل من i) N

ii) عناصر كل من المنظومتين A , B

iii) متغير X

(ب) يحسب S مجموع حواصل ضرب A_iB_i العناصر المتقابلة في المنظومتين لقيم i الفردية.

(ج) يحسب القيمة المتوسطة المركبة AV للقيم المطلقة لعناصر المنظومتين والتي تعطى بالعلاقة

$$AV = \frac{\sum_{i=1}^{N}|A_i| + \sum_{i=1}^{N}|B_i|}{2N}$$

(د) يحسب قيمة XNEW باستخدام المعادلة

$$XNEW = X - P(X) / P'(X)$$

حيث

$$P(X) = A_1+A_2X+A_3X^2+...+A_{i+1}X^i+...+A_NX^{N-1} = \sum_{i=0}^{N-1} A_{i+1}X^i$$

$$P'(X) = A_2+2A_3X+3A_4X^2+...+iA_{i+1}X^{i-1}+...+(N-1)A_NX^{N-2} = \sum_{i=1}^{N-1} iA_{i+1}X^{i-1}$$

٨-٣٥) في نظم البنوك الإسلامية تكون المعاملات بين البنك والأفراد خالية من أي نوع من أنواع الربا – الذي يسمى حاليا بالفائدة !! للتمويه على الناس وخداعهم حتى لا يبدو محرما – وإنما يجب المشاركة عامة في الأرباح والخسائر بدون تحديد نسبة ثابتة (الفائدة) من رأس المال مقدما .. نفرض أن كل شخص له سجل بيانات يحمل إما عددا موجبا يمثل المبلغ الذي أودعه الشخص في البنك للاستثمار أو عددا سالبا يمثل المبلغ الذي اقترضه من البنك. ونفرض كذلك أن الأشخاص المودعين سوف يقتسمون في نهاية العام الربح الكلي (أو الخسارة الكلية) TPL بحيث أن الشخص الذي أودع كمية من المال تساوي A من مجموع المبالغ المودعة TD يكون نصيبه

$$S = TPL * \frac{A}{TD}$$

اكتب برنامجا لقراءة قيمة TPL وقراءة ألف سجل بيانات (لألف شخص) ثم لحساب :

(أ) عدد الأشخاص المودعين ND

(ب) عدد الأشخاص المقترضين NL

(ج) كمية المبالغ المودعة TD

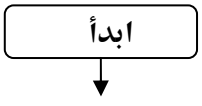
(د) كمية المبالغ المقترضة TL

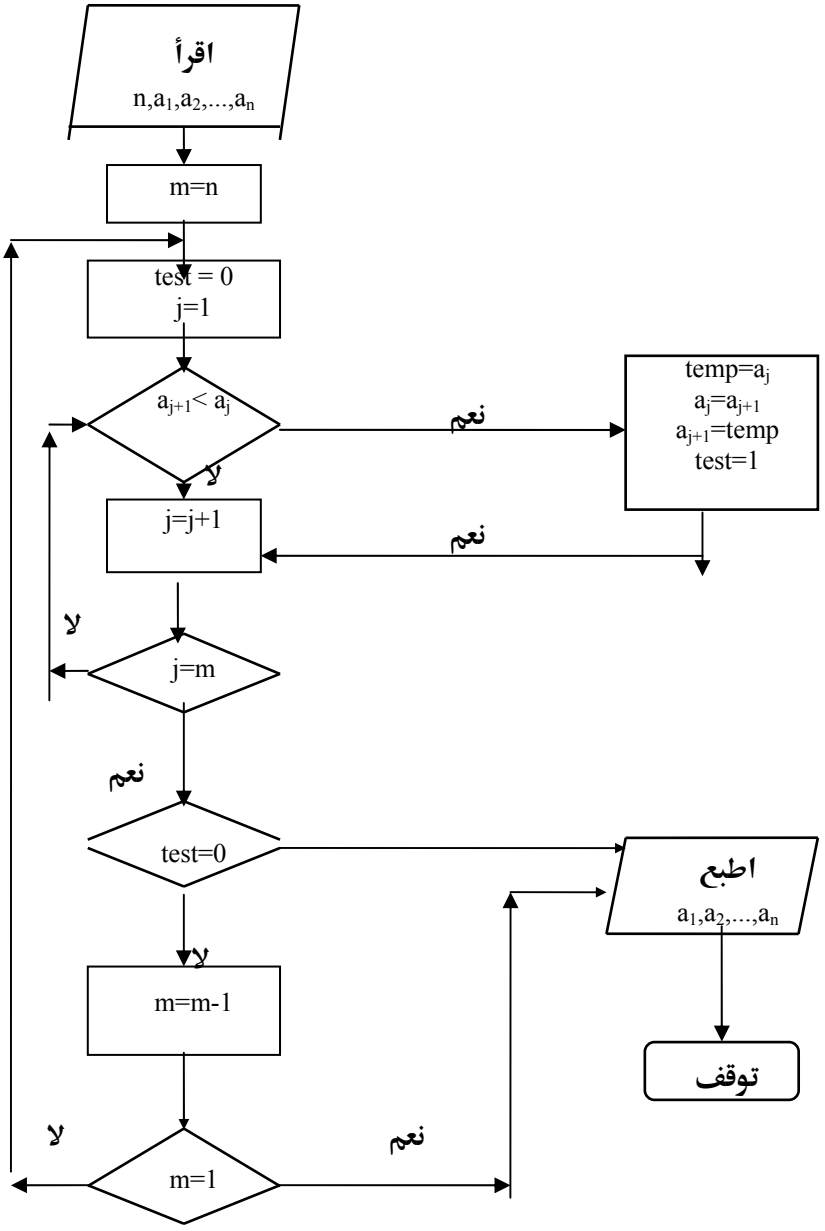
(هـ) نصيب كل شخص S من الربح أو الخسارة (بحيث أن الشخص المقترض لا يشارك في الربح أو الخسارة ، أي أن S = 0 بالنسبة للشخص المقترض).

٨-٣٦) يبين الشكل المرسوم خريطة سير عمليات برنامج يستخدم طريقة التبديل (Exchange) أو الترتيب الفقاعي (Bubble Sort) لترتيب عناصر منظومة ترتيبا تصاعديا.

المطلوب : ترجمة هذه الخريطة إلى برنامج لترتيب عناصر منظومة / مجموعة لا يزيد عدد عناصرها عن خمسين عنصرا ترتيبا تصاعديا.

ملاحظة : فكرة هذه الطريقة هي تنفيذ عدة مراحل : في كل مرحلة يقارن العنصر الأول في المجموعة مع العنصر الثاني ويوضع أصغرهما في الموضع الأول ، ثم يقارن العنصر الثاني - بعد ترتيب العنصرين الأوليين - مع الثالث ويوضع أصغرهما في الموضع الثاني ، وهكذا حتى ننتهي من كل المجموعة ، ونواصل في كل مرحلة ترتيب العناصر بهذه الكيفية إلى أن لا يحدث أي تبديل لأي عنصرين فتكون العناصر كلها حينئذ مرتبة تصاعديا (انظر المثال التوضيحي لهذه الطريقة).





طريقة الترتيب الفقاعي (المسألة ٨-٣٦)

مثال :

٨	٦	٣	٣	٣	٢	٢
٦	٣	٥	٥	٢	٣	٣
٣	٥	٦	٢	٥	٤	٤
					.....	
٥	٨	٢	٦	٤	٥	٥
				.....		
١١	٢	٧	٤	٦	٦	٦
			.....			
٢	٧	٤	٧	٧	٧	٧
		.....				
٧	٤	٨	٨	٨	٨	٨
	.....					
٤	١١	١١	١١	١١	١١	١١
المجموعة المعطاة	المرحلة الأولي	المرحلة الثانية	المرحلة الثالثة	المرحلة الرابعة	المرحلة الخامسة	المرحلة السادسة

(تسمى الطريقة باسم الترتيب الفقاعي لأن العناصر الموجودة أسفل وضعها الصحيح تميل إلى التحرك لأعلى كالفقاعات. انظر مثلا إلى تحركات أصغر عنصر وهو الرقم ٢)

مثال على طريقة الترتيب الفقاعي (المسألة ٨-٣٦)

٨-٣٧) المطلوب كتابة برنامج يحسب مجموع المتسلسلة $S = \sum_{i=1}^{100} T_i$ حيث حدود هذه المتسلسلة تُعَرَّف كما يلي :

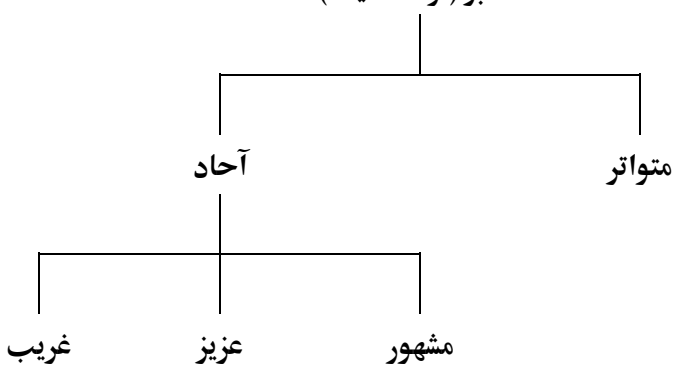
$T_1 = 0$
$T_2 = 1$
$T_i = T_{i-1} + T_{i-2}$; $i = 3 , 4 , 5 , \ldots$

(ملاحظة : هذه المتسلسلة تعرف باسم متسلسلة فيبوناتشي Fibonacci).

٨-٣٨) نفرض أن A منظومة مكونة من n عدد حقيقي ، حيث $n \leq 50$. اكتب برنامجا يقرأ قيمة n ، وقيم عناصر المنظومة A، ثم يقوم بعكس ترتيب هذه العناصر ، أي بتبديل العنصر الأول مع العنصر الأخير ، والعنصر الثاني مع العنصر قبل الأخير ، وهكذا. والمثال التالي يوضح ذلك لمنظومة مكونة من ستة عناصر.

المنظومة الأصلية	المنظومة المعكوسة
-8.0	2.0
7.0	-17.0
3.0	25.0
25.0	3.0
-17.0	7.0
2.0	-8.0

٨-٣٩) ينقسم الخبر (أو الحديث) باعتبار وصوله إلينا إلى قسمين :



(١) المتواتر : وهو ما رواه – في كل طبقة من طبقات سنده – عدد كثير (لا يقل عن عشرة أشخاص) تُحيل العادة تواطؤهم على الكذب.

(٢) الآحاد : وهو ما لم يجمع شروط المتواتر. وينقسم خبر الآحاد بدوره بالنسبة إلى عدد طرقه إلى ثلاثة أقسام :

(أ) المشهور : وهو ما رواه – في كل طبقة – ثلاثة فأكثر ، ما لم يبلغ حد التواتر.

(ب) العزيز : وهو ما لا يقل رواته – في كل طبقة – عن اثنين ، بشرط أن تبقى ولو طبقة واحدة فيها اثنان.

(ج) الغريب : وهو ما ينفرد بروايته راوٍ واحد في أي طبقة من طبقات السند.

المطلوب : اكتب برنامجا

(أ) يقرأ : – عدد طبقات السند : M (حيث $M \le 6$).

– الرقم التعريفي للحديث : ID

– عدد رواة الحديث في كل طبقة من طبقات السند :

$N_1 N_2 , ... , N_i , ... , N_M$

(ب) ثم يُحَدَّد نوع الحديث إن كان متواترا أم حديث آحاد ، وفي الحالة الأخيرة يحدد ما إذا كان حديثا مشهورا أو عزيزا أو غريبا.

إرشاد : يسهل بإذن الله تعالى تحديد نوع الحديث عن طريق معرفة أقل عدد رواته في طبقات سنده ، أي تحديد أصغر عنصر في المنظومة N.

٨–٤٠) اكتب برنامجا يقرأ عناصر منظومتين متوازيتين A , B طول كل منهما N (حيث N <= 10) من أعداد صحيحة ، ويعطي مجموع عناصر B التي تقابلها عناصر من A أكبر من 2.

مثال :

A :	0	5	0	3	0	2	5
B :	2	7	4	6	3	8	3

sum = 7 + 6 + 3 = 16

٨-٤١) نفرض أن A,B فصلان بكل منهما عشرون طالبا. يراد حفظ درجات طلاب الفصلين في منظومتين SA,SB من أعداد صحيحة. اكتب برنامجا لقراءة درجات طلاب الفصلين وحساب MaxA : أعلى درجة في الفصل A ، وكذلك MaxB : أعلى درجة في الفصل B ، ثم إعطاء رسالة تفيد أي هاتين الدرجتين أكبر.

٨-٤٢) اكتب برنامجا لقراءة درجات طلاب الفصلين A , B في السؤال السابق ، وحساب متوسطي درجات الفصلين AvgA , AvgB ، وطباعة رسالة تفيد أي المتوسطين أعلى.

٨-٤٣) اكتب برنامجا يستخدم ثلاث منظومات (مجموعات مرتبة) لتخزين معلومات عن عدد n من الطلاب (حيث $n \le 100$) :

المنظومة الأولى لتخزين الأرقام التعريفية للطلاب.

والثانية لتخزين درجات الطلاب ، ونوعها : أعداد صحيحة.

والثالثة لتخزين أحرف تحدد القسم المسجل به الطالب ، كما يلي :

'M' تعني : قسم الرياضيات

'C' تعني : قسم الحاسب

'B' تعني : قسم علم الأحياء

وفيما يلي مثال لهذه المنظومات الثلاث في الحالة $n = 5$:

9911	90	C
9912	80	M
9913	75	B
9914	60	M
9915	88	C

البرنامج المطلوب :

أ) يقرأ بيانات جميع الطلاب ويخزنها في المنظومات.

ب) يوجد ويطبع الدرجة المتوسطة لجميع طلاب قسم الحاسب.

ج) يوجد ويطبع الرقم التعريفي لأحسن طالب في قسم الرياضيات ودرجته.

٨-٤٤) تقوم إحدى الجامعات بحساب المتوسطات الوزنية (weighted averages) للطلاب في نهاية كل فصل دراسي وذلك عن طريق قراءة ثلاث درجات $score_1$, $score_2$, $score_3$ لكل طالب ، ومن ثم تقوم بحساب متوسطه الوزني.
اكتب برنامجا يقوم بقراءة عدد الطلاب n (حيث $n \leq 50$) ، والأوزان الاختبارية $weight_1$, $weight_2$, $weight_3$ (test weights) وبيانات الطلاب ، حيث تتكون بيانات أي طالب من الثلاث درجات الاختبارية ورقم الطالب (وهو عدد صحيح مكون من ٤ أرقام 4 digits) ، ومن ثم يقوم بحساب المتوسط الوزني (Weighted Average) لكل طالب باستخدام العلاقة

$$\text{Weighted Average} = weight_1 \times score_1 + weight_2 \times score_2 + weight_3 \times score_3$$

وفي النهاية يطبع أكبر متوسط ومتوسط كل المتوسطات
شكل البيانات للطلاب كما يلي :

الأوزان الاختبارية : 0.35 0.25 0.40 (test weights)

الدرجات الاختبارية ورقم الطالب : 100 76 88 1014 (test scores and ID)

٨-٤٥) عمل مسح شامل للأسر الساكنة في إحدى المدن ، واشتمل سجل بيانات كل أسرة على : رقم تعريفي للأسرة عبارة عن عدد صحيح يتكون من ٤ أرقام ، والدخل السنوي للأسرة ، وعدد أفراد الأسرة.
اكتب برنامجا يقرأ هذه البيانات في ثلاث منظومات ، وذلك بعد قراءة العدد الإجمالي للأسر التي شملها المسح (افرض أن عدد هذه الأسر لا يزيد عن ٢٥).

أ) احسب متوسط دخل الأسر السنوي ، ثم أعد قائمة بالرقم التعريفي والدخل السنوي لكل أسرة يزيد دخلها عن الدخل المتوسط.

ب) احسب النسبة المئوية للأسر التي يقل دخلها عن حد الفقر P والذي يحسب بالعلاقة :

$$P = \$\ 6500.00 + \$\ 750.00 * (m-2)$$

حيث m تمثل عدد أفراد الأسرة. لاحظ من هذه العلاقة أن حد الفقر يعتمد على عدد أفراد الأسرة m ، وأن هذا الحد يزيد بزيادة m. شكل البيانات كما يلي :

عدد أفراد الأسرة	الدخل السنوي	الرقم التعريفي
4	12.180	1041
3	13.240	1062
2	19.800	1327
8	22.458	1483
2	17.000	1900
7	18.125	2112
2	15.623	2345
6	3.200	3210
5	6.500	3600
2	11.970	3601
3	8.900	4725
2	10.000	6217
1	6.200	9280

٨-٤٦ أعدت إجابات طلاب أحد فصول علم الحاسب على أحد الاختبارات ذي الأسئلة صح أم خطأ (T/F : true - false) لإدخالها كبيانات لأحد البرامج. وبالنسبة لكل طالب تتكون المعلومات المتوفرة من رقم الطالب وإجاباته علي ١٠ من هذه الأسئلة T/F. والبيانات المتوفرة لدينا هي كما يلي :

رقم الطالب	سلسلة الإجابات
0080	FTTFTFTTFT
0340	FTFTFTTTFF
0341	FTTFTTTTTT
0401	TTFFTFFTTT
0462	TTFTTTFFTF
0463	TTTTTTTTTT
0464	FTFFTFFTFT
0512	TFTFTFTFTF
0618	TTTFFTTFTF
0619	FFFFFFFFFF
0687	TFTTFTTFTF
0700	FTFFTTFFFT
0712	FTFTFTFTFT
0837	TFTFTTFTFT

اكتب برنامجا يقرأ أولا سلسلة الإجابات العشر الصحيحة (اعتبر أنها FTFFTFFTFT) ثم يقرأ بعد هذا عدد الطلاب N (حيث $N \leq 20$) وبيانات كل طالب ويحسب ويخزن عدد الإجابات الصحيحة لكل طالب في منظومة ،

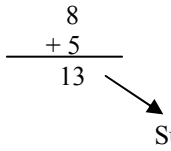
ويخزن رقم الطالب ID في العنصر المقابل من منظومة أخرى. ثم يحدد البرنامج أفضل درجة BEST. ومن ثم يطبع جدولا من ثلاثة أعمدة تعرض رقم الطالب ID ودرجته وتقديره ، حيث التقدير يحسب كما يلي :

- إذا كانت الدرجة تساوي Best أو (Best-1) فإن التقدير = A
- إذا كانت الدرجة تساوي (Best-2) أو (Best-3) فإن التقدير = C
- وما عدا ذلك فإن التقدير = F

٨-٤٧) افرض أن حاسبك له قدرات محدودة إلى درجة كبيرة بحيث يستطيع قراءة وطباعة العدد الصحيح المكون من رقم واحد فقط ، وكذلك يستطيع جمع عددين صحيحين يتكون كل منهما من رقم عشري واحد فقط.
اكتب برنامجا يستطيع قراءة عددين صحيحين يتكون كل منهما من ٣٠ رقم على الأكثر ، ويجمع هذين العددين (أي يجمع هذه الأرقام المتقابلة) ويعرض النتيجة. اختبر البرنامج باستخدام أزواج أعداد ذوات أطوال مختلفة.

إرشاد : خزِّن العددين في منظومتين سعة كل منهما ٣٠ عنصرا ، بحيث تخزن رقما واحدا في العنصر الواحد من المنظومة. وإذا كان طول العدد أقل من ٣٠ رقما ، فأدخل عددا كافيا من الأصفار في البداية (على شمال العدد) لتجعل طول العدد ٣٠ رقما.
وستحتاج لعروة لجمع الأرقام في العناصر المتقابلة في المنظومتين مبتدئا بالموضع (المؤشر) رقم ٣٠. لا تنس أن تدخل في الاعتبار الرقم الحامل (الباقي) إذا كان هناك حامل. استخدم متغيرا منطقيا للدلالة على إذا ما كان مجموع آخر زوج من الأرقام أكبر من ٩ أم لا.

ملاحظة :



الباقي أو الحامل = 1 **المجموع= 3**

٨-٤٨) اشتمل كتاب "الإصابة في تمييز الصحابة" لابن حجر العسقلاني في جزئه الأخير على ذكر أسماء نحو ألف وخمسمائة من الصحابيات رضوان الله عليهن جميعا مع ذكر نبذة عن كل واحدة منهن.

نفرض أننا سنقوم بإدخال هذه الأسماء في الحاسب اسما اسما ، بحيث أن كل اسم يكتب على سطر مستقل ، ونفرض أن أي اسم لا يتجاوز ٢٠ رمزا (حرفا أو رمزا خاصا).

المطلوب :

(أ) كتابة برنامج بلغة الباسكال يقوم بما يلي :

١) قراءة الأسماء إلى أن تظهر العلامة * (والتي تعني انتهاء قائمة الأسماء) ، وتخزين هذه الأسماء في منظومة List من سلاسل رموز. أي أن العنصر List[I] يمثل اسم الصحابية التي ترتيبها i (في الإدخال).

٢) ترتيب أسماء المنظومة ترتيبا أبجديا.

٣) طباعة أسماء الصحابيات بعد الترتيب الأبجدي.

(ب) اختبار صحة البرنامج بإدخال نحو عشرة أسماء من الصحابيات مثل :

خديجة بنت خويلد عائشة بنت أبي بكر حفصة بنت عمر أم سلمه

سمية بنت خياط خولة بنت ثعلبة أسماء بنت أبي بكر نسيبة بنت كعب

أسماء بنت عميس أم سليم بنت ملحان (الرميصاء)

٨-٤٩ أ) اكتب دالة خاوية

Break (x , whole , fraction)

تعطي قيمة الجزء الصحيح whole والجزء الكسري fraction من قيمة أي عدد حقيقي مُعطَى x.

(إرشاد : يمكن استخدام اسم النوع int لتعيين قيمة الجزء الصحيح ، ثم الفرق بين هذه القيمة وقيمة العدد الأصلي يعطى الجزء الكسري).

ب) اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة A مكونة من خمسين عددا حقيقيا ، ويعيِّن لكل عدد جزأه الصحيح وجزأه الكسري باستخدام البرنامج الفرعي الإجرائي.

٨-٥٠) (١) المطلوب كتابة دالة خاوية تقوم بضرب كل عنصرين متقابلين في منظومتين X,Y تحتوي كل منهما على n عنصر ، وتضع العناصر الناتجة في منظومة Z.

(ب) المطلوب كتابة برنامج رئيسي يقرأ قيم عناصر منظومتين A,B تحتوي كل منهما على ثمانية عناصر ، ثم يستدعي الدالة الخاوية ليخزن حاصل ضرب عناصر A,B المتقابلة في منظومة جديدة C ، وكذلك يطبع البرنامج الرئيسي عناصر المنظومات الثلاث A , B , C.

٨-٥١) (أ) اكتب دالة خاوية لتعديل (adjusting) قيم جميع عناصر منظومة أعداد float مكونة من n عنصر ، عن طريق إزاحة (shifting) جميع القيم إزاحة متساوية بحيث تكون أصغر قيمة في المنظومة تساوي صفرا.

مثلا إذا كانت أصغر قيمة في المنظومة الأصلية هي 8.0 فإننا نطرح 8.0 من قيمة كل عنصر في المنظومة لنحصل على المنظومة المعدلة ، وبالمثل إذا كانت أصغر قيمة هي 6.5 - فإننا نطرح 6.5 - من (أي نضيف 6.5 إلى) قيمة كل عنصر.

ويتم استدعاء الدالة بالعبارة

adjust (a,n) ;

(ب) أعد حل الجزء أ) ولكن بحيث تكون أكبر قيمة في المنظومة المعدلة تساوي صفرا .

ويتم استدعاء الدالة بالعبارة

```
shift (a,n) ;
```

(ج) أعد حل الجزء أ) ولكن بحيث تكون القيمة المتوسطة في المنظومة المعدلة تساوي صفرا .

ويتم استدعاء الدالة بالعبارة

```
zeroav (a , n) ;
```

٨-٥٢) اكتب دالة خاوية لطباعة قيم عناصر منظومة أعداد صحيحة k عددها n في الصيغة التالية :

```
VALUES IN ARRAY
1.    XXX
2.    XXX
3.    XXX
```

وهكذا حتى تطبع جميع القيم.

ويتم استدعاء الدالة بالعبارة

```
prtval (k, n);
```

٨-٥٣) اكتب دالة خاوية bigarr له ثلاثة وسطاء a , b , c وكل منها عبارة عن منظومة أعداد float عددها 50 ، حيث تستقبل الدالة قيما لعناصر كل من المنظومتين a , b ، وتعين قيم عناصر المنظومة c بناء على القاعدة التالية :
إذا كان مجموع قيم عناصر a أكبر من أو يساوي مجموع قيم عناصر b فإن الإجراء يسند لعناصر المنظومة c قيم العناصر المقابلة في المنظومة a وما عدا ذلك فإنه يسند لعناصر c قيم العناصر المقابلة في b.

٨-٥٤) اكتب دالة خاوية bottom لهه وسيطان :

b : منظومة أعداد float مكونة من خمسين عنصرا ،

limit : قيمة اختبارية (test value) وهي عدد float ،

حيث تقوم الدالة بتعديل (adjusting) قيم عناصر المنظومة b بناء على القيمة الاختبارية limit تبعا للقاعدة التالية :

إذا كانت قيمة أي عنصر بالمنظومة أصغر من القيمة limit فغيِّر قيمة العنصر لتصبح هي القيمة limit ، وما عدا ذلك فدع قيمة العنصر كما هي دون تغيير. ويتم استدعاء الدالة بواسطة العبارة bottom (b , limit).

٨-٥٥) اكتب دالة خاوية تقوم بتحويل أي عنصر من عناصر منظومة أعداد صحيحة b (بها خمسون عنصرا) إلى 0 أو 1 حسب ما إذا كان العنصر عددا زوجيا أم فرديا على الترتيب ، أي أنه بعد استدعاء الدالة بالعبارة

```
chng (b) ;
```

تصبح المنظومة b مكونة من أصفار وآحاد فقط.

٨-٥٦) اكتب دالة حاوية يمكن استدعاؤها بالعبارة

```
modify (x) ;
```

حيث x منظومة مكونة من 50 عدد float . وتقوم الدالة بعكس ترتيب (reversing the order) قيم المنظومة ، أي أنه يبدل القيمة الأولى مع القيمة الأخيرة ، والقيمة الثانية مع القيمة قبل الأخيرة ، وهكذا.

٨-٥٧) نفرض أن المنظومة y تتكون من مائة قيمة float موجبة تمثل قياسات تجريبية (experimental measurements) تحتوي على قدر بسيط من التداخل (interference) أو الضوضاء (noise) التي حدثت أثناء التجربة. من الطرق التي تستخدم لإلغاء تأثير هذه الضوضاء أن نستبدل بالقياسات الصغيرة أصفارا ، بفرض أن هذه القياسات الصغيرة تمثل ضوضاء فقط. اكتب دالة خاوية تستقبل منظومة y مكونة من مائة قيمة float موجبة ، وتستبدل أصفارا بالقيم التي تقل أي منها عن ٣٪ من أكبر قيمة بالمنظومة.

٨-٥٨) أ) اكتب دالة خاوية scale لتعديل قيم جميع عناصر منظومة أعداد float موجبة a عدد عناصرها n حيث $n \le 50$ ، وذلك بقسمة كل قيمة في المنظومة على أكبر قيمة في المنظومة ، وبذلك تصبح جميع القيم محصورة بين القيمتين 1.0 , 0.0.

مثلا : المنظومة $\begin{bmatrix} 4.0 \\ 8.0 \\ 6.0 \\ 2.0 \end{bmatrix}$ تصبح بعد التعديل $\begin{bmatrix} 0.5 \\ 1.0 \\ 0.75 \\ 0.25 \end{bmatrix}$ حيث $n = 4$

ويتم استدعاء الدالة scale بالعبارة :

```
scale (a , n) ;
```

ب) اكتب برنامجا يقرأ قيم عناصر منظومة a من 50 عدد float موجب ، ويستدعي الدالة scale لتعديل قيم عناصر a ثم يطبع قيم عناصر a بعد التعديل.

٨-٥٩) اكتب دالة خاوية (procedure) تعكس ترتيب القيم المخزونة في منظومة. المتغيرات الوسيطة (parameters) في الدالة هي : منظومتان X , Y من الأعداد العائمة float ، وعدد صحيح n يمثل طول (length) كل من المنظومتين.

مدخلات الدالة (procedure inputs) : X , n

مخرجات الدالة (procedure outputs) : Y

والدالة تقوم بنسخ (copying) عناصر المنظومة X في المنظومة Y بترتيب معاكس (وذلك باستخدام عروة) ، أي أن :

$Y[1] = X[n] , Y[2] = X[n-1] , \dots , Y[n-1] = X[2] , Y[n] = X[1]$

٨-٦٠ أ) اكتب دالة خاوية Subsequences تستقبل منظومة X من n عدد صحيح ، وتعطي منظومة Y من n عدد صحيح ، عناصرها هي مجاميع المتتاليات الجزئية (array subsequences) للمنظومة X ، أي أن :

$y_1 = x_1$

$y_2 = x_1 + x_2$
$y_3 = x_1 + x_2 + x_3$
$\vdots$
$y_i = x_1 + x_2 + \ldots + x_j + \ldots + x_i$
$\vdots$
$y_n = x_1 + x_2 + \ldots\ldots\ldots\ldots + x_n$

أي أن : $y_i = \sum_{j=1}^{i} x_j \quad ; \quad i = 1,2,3,\ldots,n$

مثلا إذا كان : $X = \{4 , 2 , 5 , 1 , 10\}$

فإن : $Y = \{4 , 6 , 11 , 12 , 22\}$

ب) اكتب برنامجا يعين قيم عناصر منظومة A من عشرة أعداد صحيحة باستخدام العلاقة

$A_i = i \; ; \; i = 1 , 2 , \ldots , 10$

ثم يستدعي الدالة السابقة لتعيين منظومة مجاميع المتتاليات الجزئية B للمنظومة A.

٨-٦١) تبعا لقاعدة المربعات الصغرى للانحرافات (method of least squares) **فإن أحسن معادلة خط مستقيم تمثل عددا من النقاط** (x_i , y_i) **يساوي** n **هي المعادلة** $y = a + bx$ **حيث :**

$$a = \frac{\sum y_i - b\sum x_i}{n} \quad , \quad b = \frac{n\sum(x_i y_i) - (\sum x_i)(\sum y_i)}{n\sum(x_i^2) - (\sum x_i)^2}$$

المطلوب :

أ) كتابة دالة خاوية

Procedure LSTSQ (X , Y , N , A , B)

لإيجاد قيمة كل من الثابتين a , b **إذا علمت النقاط**

$(x_1 , y_1) , (x_2 , y_2) , \ldots , (x_n , y_n)$

ب) كتابة برنامج يقرأ إحداثيات عشرين نقطة ، ثم يستدعي الدالة السابقة لحساب قيمة كل من a , b **، ثم يطبع معادلة الخط المستقيم في الصورة** $y = a + bx$ **(بعد التعويض عن كل من** a , b**).**

٨-٦٢) أ) اكتب دالة تعيد قيمة freq **تأخذ الوسيطين** a , n

حيث a : منظومة مكونة من ٥٠ عدد float.

n : عدد float

وتعطي عدد مرات ظهور العدد n في المنظومة a.

ب) اكتب برنامجا يقرأ قيم عناصر منظومة X تمثل درجات طلاب فصل به خمسون طالبا ، ويستخدم الدالة freq لمعرفة عدد الطلاب الحاصلين على الدرجة النهائية (١٠٠).

٨-٦٣) أ) اكتب دالة منطقية ascend تستقبل وسيطين a , n حيث a منظومة أعداد float عدد عناصرها n ، وتعيد الدالة القيمة true إذا كانت المنظومة a مرتبة ترتيبا تصاعديا ، وإلا فإنها تعيد القيمة false.

ملاحظة : الدالة لا تقوم بترتيب عناصر المنظومة ، ولا تقوم بتغيير أي عنصر من عناصرها ، ولا تقوم بتغيير قيمة n.

ب) أعد حل الجزء أ) ولكن بالنسبة لدالة منطقية descnd تعيد القيمة true عندما تكون المنظومة a مرتبة ترتيبا تنازليا.

٨-٦٤) اكتب دالة منطقية boundd تستقبل ٣ وسطاء : منظومة a من النوع float بها خمسون عنصرا ، وعددين rmin , rmax من النوع float ، وتعيد القيمة true إذا وقعت جميع قيم عناصر المنظومة a في المدى ما بين الحد الأدنى (lower bound) rmin والحد الأقصى (upper bound) rmax ، أما إذا وقعت أي قيمة خارج هذا المدى فإن الدالة تعيد القيمة false.

٨-٦٥) من الممكن أن تحدث القيمة العظمى أكثر من مرة واحدة في مجموعة من البيانات. فمثلا يمكن أن يحصل طالبان على أعلى درجة في اختبار ما.
اكتب دالة تعيد قيمة صحيحة maxs تحسب عدد مرات ظهور أعلى قيمة في منظومة أعداد float ، حيث تستقبل الدالة وسيطين : المنظومة a وعدد عناصرها n.

٨-٦٦) اكتب دالة تعيد القيمة range من النوع float توجد مدى قيم عناصر منظومة أعداد a من النوع float عدد عناصرها n ، حيث يعرف المدى هنا بأنه الفارق بين أكبر قيمة وأصغر قيمة من قيم عناصر المنظومة. الدالة تستقبل الوسيطين a , n .

٨-٦٧) اكتب دالة تعيد القيمة close من النوع float تستقبل وسيطين : a , n ، حيث a : منظومة أعداد float ، و n : عدد عناصر المنظومة a ، ثم توجد قيمة أقرب عنصر (nearest / closest element) من عناصر المنظومة للقيمة المتوسطة (average value) لعناصر المنظومة.

٨-٦٨) اكتب دالة تعيد قيمة صحيحة تأخذ منظومة رموز (characters) عدد عناصرها 25 وتحسب عدد مرات ظهور الحرف A الكبير في هذه المنظومة ، ويمكن استدعاء الدالة بالعبارة a = count (b) ; حيث a عدد صحيح ، و b منظومة رموز .

٨-٦٩) اكتب دالة خاوية تقوم بتبديل عنصرين معينين من عناصر منظومة. افرض أن متغيرات الدالة الوسيطة هي :

A : اسم المنظومة

i , j : ترتيبا العنصرين المطلوب تبديلهما.

افرض كذلك أن المنظومة لا تحتوي على أكثر من ٢٤ عنصرا ..

٨-٧٠) أ) نفرض أن كلا من A , B منظومة مكونة من عدد من العناصر يساوي N. المطلوب كتابة دالة تعيد قيمة PROD (A , B , N)

تحسب مجموع حواصل ضرب العناصر المتقابلة في المنظومتين

$$\sum_{i=1}^{N} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_N B_N$$

ب) اكتب برنامجا يقرأ قيم عناصر كل من المنظومات X , Y , Z والتي تتكون كل منها من أربعين عنصرا ، ثم يستخدم الدالة السابقة في حساب المقادير التالية :

$$\alpha = \left(\sum_{i=1}^{40} x_i z_i \right)^2$$

$$\beta = \frac{\sqrt{x_1^2 + x_2^2 + \cdots + x_{40}^2} \cdot \sqrt{z_1^2 + z_2^2 + \cdots + z_{40}^2}}{\sqrt{x_1 z_1 + x_2 z_2 + \cdots + x_{40} z_{40}}}$$

$$\gamma = x_1(y_1 + z_1) + x_2(y_2 + z_2) + \cdots + x_{20}(y_{20} + z_{20})$$

٨-٧١) أ) نفرض أن X منظومة ، وأن عناصرها أعداد صحيحة ، وأن عدد هذه العناصر M. اكتب دالة تعيد قيمة صحيحة :

NCOUNT (X , M , J , K , ITEM)

تحسب عدد مرات ظهور العنصر ITEM في المجموعة الجزئية للعناصر ابتداء من العنصر رقم J إلى العنصر رقم K (K ≥ J) من المجموعة X. فمثلا إذا كانت المجموعة X هي :

X = [0 , 1 , 1 , 2 , 0 , 0 , 0 , 0 , 2]

فيكون NCOUNT (X , 9 , 2 , 7 , 0) = 3

ب) اكتب برنامجا يقرأ قيم عناصر منظومة A مكونة من ثمانين عنصرا من الأصفار والآحاد ثم يستخدم الدالة السابقة لإيجاد :

١- عدد الأصفار في المنظومة A ونرمز له بالرمز N0.

٢- عدد الآحاد في المنظومة A ونرمز له بالرمز N1.

٣- عدد الأصفار في أول خمسين عنصرا من المنظومة A ونرمز له بالرمز N.

٤- عدد الآحاد في مجموعة العناصر ابتداء من العنصر الثلاثين إلى العنصر السبعين ونرمز له بالرمز L.

٨-٧٢) أ) اكتب دالة تعيد قيمة float Y (X, N, MEAN)

تحسب قيمة Y المناظرة للقيمة المعطاة للمتغير MEAN كما يلي : Y هو المتوسط الحسابي أو الهندسي لعناصر المنظومة X المكونة من N عنصر حسب ما إذا كانت قيمة MEAN تساوي ١ أو ٢ على الترتيب ، أي أن :

$$Y = \begin{cases} \dfrac{X_1 + X_2 + \cdots + X_N}{N} & \text{if MEAN} = 1 \\ \left(X_1 . X_2 \ldots X_N\right)^{\frac{1}{N}} & \text{if MEAN} = 2 \end{cases}$$

ب) اكتب برنامجا يقرأ قيم العناصر الثلاثين لمنظومة A وقيمة متغير K ثم يستخدم الدالة السابقة لحساب المتوسط الحسابي أو المتوسط الهندسي لعناصر المنظومة A حسب ما إذا كانت قيمة K تساوي ١ أو ٢ على الترتيب.

٨-٧٣) أ) اكتب دالة تعيد قيمة تحسب مجموع القيم المطلقة لعدد n من الأعداد $X_1 , X_2 , \ldots , X_n$

أي تحسب $\sum_{i=1}^{n} |X_i|$

ب) اكتب برنامجا :

١- يقرأ قيمة عدد صحيح n_1 (حيث $n_1 \le 30$) وقيم عناصر منظومة A مكونة من n_1 عنصرا.

٢- يقرأ قيمة عدد صحيح n_2 (حيث $n_2 \le 40$) وقيم عناصر منظومة B مكونة من n_2 عنصرا.

٣- يستخدم الدالة السابقة لحساب القيمة المتوسطة المركبة AV للقيم المطلقة لعناصر المنظومتين والتي تعطى بالعلاقة

$$AV = \frac{\sum_{i=1}^{n_1} |A_i| + \sum_{i=1}^{n_2} |B_i|}{n_1 + n_2}$$

٤- يطبع قيم عناصر المنظومة A في صورة متجه (أي كل عنصر على سطر مستقل) وكذلك عناصر المجموعة B في صورة متجه ثم القيمة المتوسطة AV مع استخدام عناوين مناسبة.

٨-٧٤) أ) اكتب دالة خاوية

POLY (A , N , X , P , PD)

حيث A منظومة مكونة من N+1 عنصرا ، والدالة تحسب قيمة كل من كثيرة حدود P وتفاضلها PD المقابلتين لقيمة متغير X ، وذلك باستخدام العلاقتين التاليتين :

$$P = P(x) = A_1 + A_2x + A_3x^2 + \dots + A_{i+1}X^i + \dots + A_{N+1}X^N = \sum_{i=0}^{N} A_{i+1}X^i$$

$$PD = P'(x) = A_2 + 2A_3x + 3A_4x^2 + \dots + iA_{i+1}X^{i-1} + \dots + NA_{N+1}X^{N-1} = \sum_{i=1}^{N} iA_{i+1}X^{i-1}$$

ب) اكتب برنامجا :

١- يقرأ قيم أول عشرة عناصر من منظومة C ، أي قيم العناصر

$C_1 , C_2 , \dots , C_{10}$

وكذلك قيمة متغير Y.

٢- يستخدم الدالة الخاوية POLY ليحسب قيمة YNEW والتي تعطى بالمعادلة : $YNEW = Y - \frac{P(Y)}{P'(Y)}$

حيث $P(Y) = \sum_{i=0}^{9} C_{i+1}Y^i$

٣- يطبع قيمة YNEW مع عنوان مناسب.

٨-٧٥) اكتب دالة خاوية

MAXMIN (A , N , BIGA , SMALLA)

لإيجاد أكبر عنصر BIGA واصغر عنصر SMALLA في المنظومة / المجموعة الجزئية للعناصر التي عددها N والتي تأتي في مطلع منظومة / مجموعة العناصر المرتبة A.

ثم اكتب برنامجا يقرأ قيمة متغير M ، وقيم أول M عنصر في مطلع منظومة X ، أي قيم العناصر

$$X_1 , X_2 , X_3 , \dots , X_M$$

ثم يستدعي الدالة الخاوية MAXMIN لإيجاد أكبر عنصر (XHIGH) وأصغر عنصر (XLOW) في المجموعة $X_1 , X_2 , X_3 , \dots , X_M$. افرض أن $M \le 20$.

٨-٧٦) أ) اكتب دالة تعيد قيمة AVG (X , N) تحسب القيمة المتوسطة لعدد N من الأرقام $X_1 , X_2 , X_3 , \dots , X_N$

أي تحسب $\left(\sum_{i=1}^{N} X_i \right) / N$

ب) الفصلان A , B فيهما عدد NA , NB من الطلاب على الترتيب بحيث أن عدد طلاب كل فصل لا يتجاوز خمسين طالبا.

حفظت درجات طلاب الفصلين - على الترتيب - في أحد الاختبارات في المنظومتين SA , SB.

اكتب برنامجا يقرأ ويطبع هذه المعلومات ، ويستخدم الدالة AVG لمعرفة أي الفصلين حصل طلابه على المعدل الأعلى لمتوسط الدرجات ، مع إعطاء رسالة توضح هذه النتيجة.

٨-٧٧) أ) اكتب دالة تعيد قيمة SMALL (A , N) لإيجاد أصغر عنصر في منظومة A مكونة من N عنصرا.

ب) فصل مكون من خمسين طالبا ، امتحن طلابه خمسة امتحانات في أحد المقررات ، وحسبت درجة الطالب في هذا المقرر بأخذ متوسط أفضل أربع درجات في هذه الامتحانات. اكتب برنامجا يقرأ رقم كل طالب ID ودرجاته الخمس ويحسب درجته في هذا المقرر.

إرشاد : يمكن لحساب مجموع أفضل أربع درجات أن نطرح أقل درجة من مجموع الخمس درجات ، وبالتالي يمكن الاستفادة من الدالة السابقة في (أ).

٨-٧٨) أ) اكتب دالة خاوية

AVNZ (A , M , N , AVA , AVG , NZ)

حيث A : منظومة مكونة من M عنصر.

والدالة تحسب المتوسط الحسابي AVA والمتوسط الهندسي AVG لأول N عنصر من المنظومة A :

$$AVA = \frac{A_1 + A_2 + \cdots + A_N}{N}$$

$$AVG = \sqrt[N]{A_1 . A_2 \cdots A_N}$$

وكذلك تحسب كم عنصرا من هذه العناصر (التي عددها N) يساوي صفرا ، وتشير إلى عدد هذه الأصفار بالاسم NZ.

ب) اكتب برنامجا يقرأ قيم عناصر منظومة BETA مكونة من خمسين عنصرا ، ثم يستعين بالدالة السابقة لحساب المتوسط الحسابي BAMEAN والمتوسط الهندسي BGMEAN لأول ٢٠ عنصرا من المنظومة BETA ، وكذلك لحساب عدد العناصر الصفرية NBCNT من هذه العشرين عنصرا الأولى.

٨-٧٩) أ) اكتب دالة خاوية

SEARCH (BUFFER , N , KEY , FOUND , INDEX)

تقوم بالبحث عن العنصر KEY في المنظومة BUFFER والمكونة من عدد من العناصر يساوي N ، فإذا وجدت العنصر ضمن هذه المجموعة من العناصر فإنها تعطي FOUND القيمة 1 وتعطي INDEX قيمة ترتيب هذا العنصر الذي وجدته في المنظومة BUFFER ، وإذا لم تجده فإنها تعطي FOUND القيمة صفر ، ولا تغير من قيمة INDEX.

ب) اكتب برنامجا يقرأ قيم عناصر منظومة A مكونة من خمسين عنصرا ومنظومة أخرى X مكونة من ثلاثين عنصرا ثم يستخدم الدالة السابقة

لمعرفة العناصر المشتركة في المنظومتين ، بحيث يطبع البرنامج هذه العناصر المشتركة وبجانب كل عنصر ترتيبه في المنظومة A.

٨-٨٠) أ) افرض أن JEST منظومة تحتوي على N1 عنصر عبارة عن أعداد صحيحة موجبة مختلفة .. وافرض كذلك أن KSET منظومة تحتوي على N2 من الأعداد الصحيحة الموجبة المتباينة. اكتب دالة خاوية

INTER (JSET , N1 , KSET , N2 , JKSET , N3)

تقوم بتخزين تقاطع المنظومتين المذكورتين في المنظومة JKSET ، أي أن JKSET ستحتوي على N3 عنصر هي الأعداد الموجودة في كل من JSET , KSET.

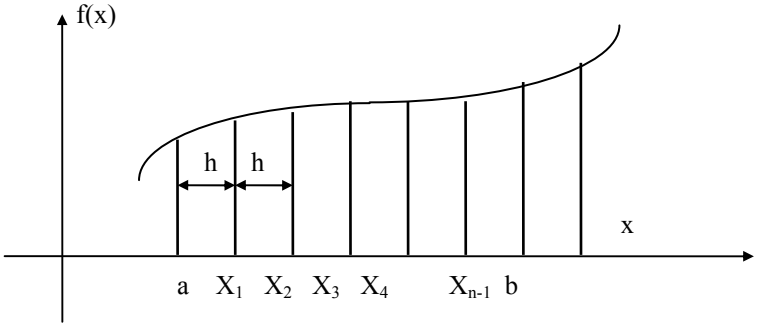
ب) نفرض أن كل اسم من أسماء الصحابة رضوان الله عليهم جميعا قد أعطى رقما موجبا مميزا (أي مختلفا عن غيره من الأرقام) ، وأن المنظومة BADR تحتوي على ٣١٤ رقما هي أرقام الصحابة الذين اشتركوا في غزوة بدر ، وأن المنظومة OHOD تحتوي على ٧٠٠ رقم هي أرقام الصحابة الذين اشتركوا في غزوة أحد.

اكتب برنامجا يقرأ أرقام كل من المجموعتين BADR و OHOD ثم يستخدم الدالة السابقة INTER لإيجاد المنظومة JBO التي تحتوي على أرقام الصحابة الذين اشتركوا في كل من الغزوتين.

٨-٨١)نفرض أن الدالة f موجبة في الفترة $a \le x \le b$. المساحة الواقعة تحت المنحنى $y = f(x)$ والمحصورة بين الخطين $x = a , x = b$ والتي تعطى بالتكامل

$$AREA = \int_a^b f(x)dx$$

يمكن حساب قيمة تقريبية لها بإيجاد مجموع مساحات عدد n من أشباه المنحرفات ، كما هو مبين بالشكل التالي :



فنحصل على القانون :

$$\text{Area} = \frac{h}{2}\left[f(a) + f(b) + 2\left\{f(x_1) + f(x_2) + \cdots + f(x_{n-1})\right\}\right]$$

$$= \frac{h}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{i=n-1} f(x_i)\right].$$

حيث :

$$h = \frac{b-a}{n}, x_i = a + ih;\ i = 1,2,\cdots,n-1$$

اكتب برنامجا :

١- **يقرأ قيمة كل من** a , b , n .

٢- **يحسب قيمة** h .

٣- **يحسب المساحة تحت المنحنى**

$$F(x)=e^{-x^2} + 3x^3 - 4x^2 + 6x + 5$$

والمحصورة بين x = a , x = b **وذلك باستخدام القانون السابق.**

إرشاد : يمكن استخدام دالة تعيد قيمة F(x).

٨٢-٨) أ) **اكتب دالة تعيد قيمة** Avg (X , n) **تحسب القيمة المتوسطة** Avg **لعناصر منظومة** X **مكونة من قيم حقيقية عددها** n.

$$\text{Avg} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

ب) اكتب دالة خاوية Big (A , m , key , count , sum) تستقبل منظومة A مكونة من قيم float عددها m ، ثم توجد عدد القيم الكبيرة : count ، وهي القيم التي تزيد كل منها عن قيمة معينة معطاة key ، ويوجد كذلك مجموع هذه القيم الكبيرة : sum.

ج) اكتب برنامجا يقرأ الأرقام التعريفية ودرجات ٤٠ طالبا وذلك في منظومتين : ID (أعداد صحيحة) و score (أعداد float) على الترتيب. ثم يستخدم الدالتين السابقتين بكفاءة لإيجاد وطباعة :

(أ) الدرجة المتوسطة av للأربعين طالبا.

(ب) عدد الطلاب المتفوقين ngood ، وهم الحاصلون على درجة أعلى من الدرجة المتوسطة av.

(ج) الدرجة المتوسطة للطلاب المتفوقين avgood.

تمرينات عامــة

١- الرسم المبين بالصفحة التالية يمثل خريطة سير عمليات برنامج يوجد عوامل (Factors) أي عـدد صـحيح موجـب أقـل مـن ١٠٠٠ ، ثـم يوجـد مجمـوع عوامله (مثلا عوامل العدد ١٢ هي : ١ ، ٢ ، ٣ ، ٤ ، ٦ ، ١٢ ، ومجموع هذه العوامل يساوي ٢٨).
والمطلوب : ترجمة هذه الخريطة إلى برنامج .

٢- يبلـغ نصـاب الـذهب والعمـلات الذهبيـة مـا يعـادل ٨٥ جرامـا مـن الـذهب الخالص ، ويقدر نصاب النقود والعملات الورقية بما يساوي قيمة ٨٥ جراما من الذهب الخالص بحساب سعر يوم الوجوب في بلد المال المزكى.
أما الذهب غير الخالص فيسقط من وزنه غير الذهب ، فمثلا في الذهب عيار ١٨ يسقط مقدار الربع ، وفي الذهب عيار ٢١ يسقط مقدار الثمن.
اكتب برنامجا يقرأ سعر الجرام من الذهب الخالص p بالدينار ، ويحسب زكوات ألف شخص باتباع الخطوات التالية :

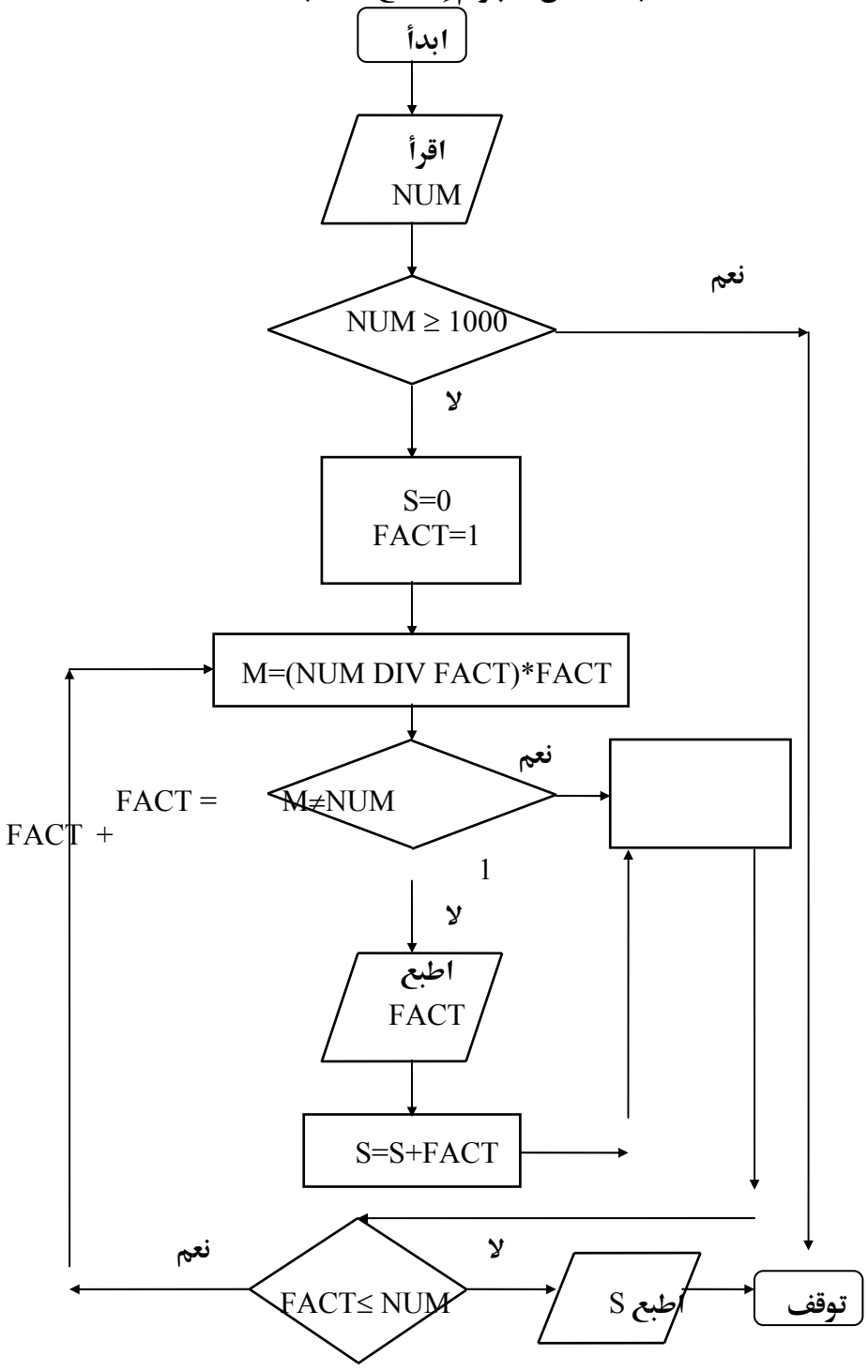
(أ) يقرأ بيانات كل شخص على سطر مستقل وهي :
NAME : اسم الشخص .
GOLD : كمية الذهب بالجرام التي حال عليها الحول.
STANDARD : عيار الذهب.
MONEY : قيمة النقود المدخرة بالدينار التي حال عليها الحول.

(ب) يحسب نصاب النقود بالدينار NB.

(iii) يحسب زكاة النقود بالدينار ZM ، وهي تساوي ٢,٥٪ من المبلغ المدخر MONEY إذا بلغ النصاب.

(iv) يحسب كمية الذهب الخالص بالجرام PUREGD وهي تساوي :
عيار الذهب × كمية الذهب بالجرام ٢٤

(هـ) يحسب زكاة الذهب ZG بالجرام ، وهي تساوي ٢,٥٪ من كمية الذهب الخالص بالجرام إذا بلغ النصاب.



٣- الرسم المبين بالصفحة التالية يوضح خريطة سير عمليات إيجاد القاسم المشترك الأعظم GCD لعددين صحيحين N1 , N2 ، وهو أكبر عدد صحيح يقبل كل من العددين N2 , N1 القسمة عليه بدون باق (مثلا القاسم المشترك الأعظم للعددين ٢٤ ، ٤٠ هو ٨).

(أ) اكتب برنامجا فرعيا داليا لإيجاد القاسم المشترك الأعظم لعددين صحيحين N1 , N2.

(ب) اكتب برنامجا رئيسيا يقرأ قيمتي عددين صحيحين M , N ، ثم يستدعي البرنامج الفرعي لحساب القاسم المشترك الأعظم MAX للعددين M , N ثم يحسب الدالة

$$G = \sin(2.MAX) + \sqrt{MAX + 5} + e^{MAX}$$

٤- اكتب برنامجا لقراءة قيمة عدد صحيح n وقيمتي متغيرين a , x ثم حساب المفكوك التالي للدالة a^x وذلك بكفاءة عالية أي بحيث يستغرق تنفيذ البرنامج أقل وقت ممكن وبحيث يشغل أقل حيز ممكن من ذاكرة الحاسب.

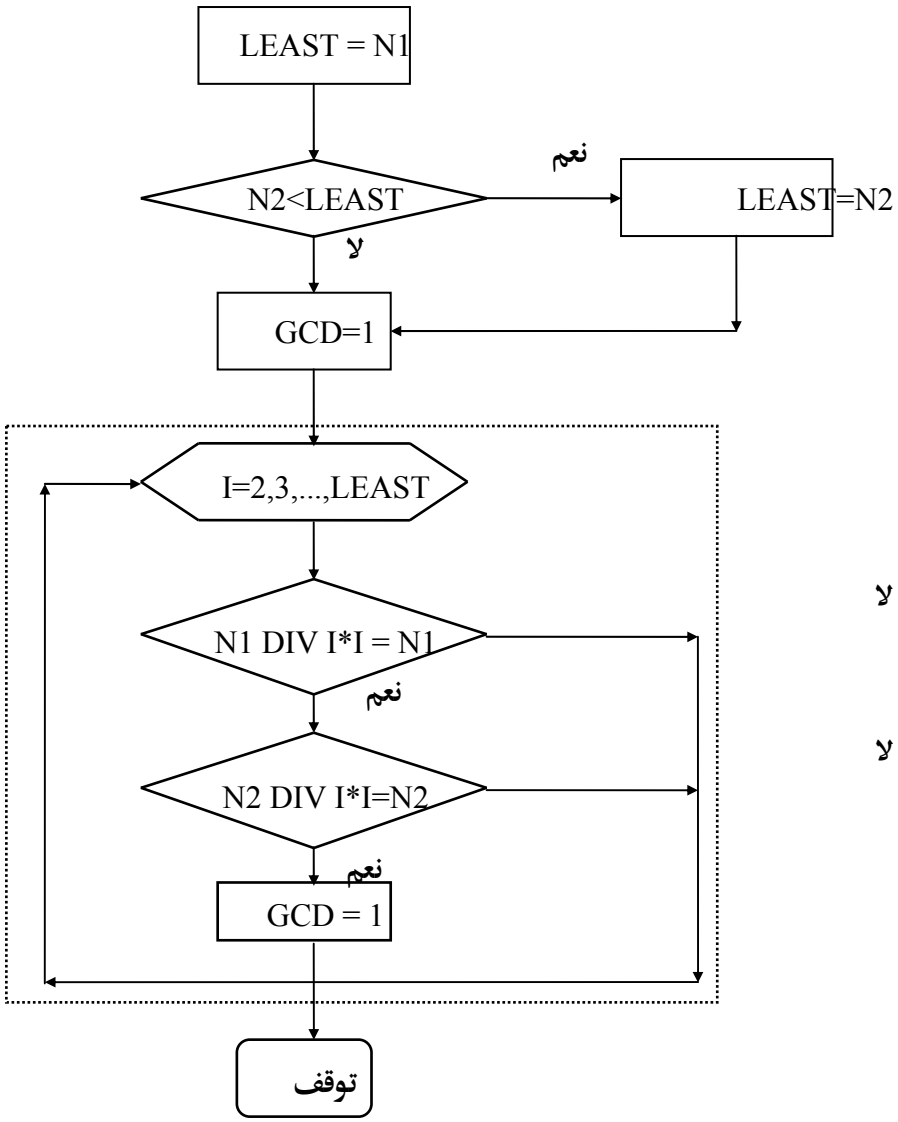
$$a^x = 1 + \frac{x \log a}{1!} + \frac{(x \log a)^2}{2!} + \cdots + \frac{(x \log a)^n}{n!}$$

ثم قارن الجواب الناتج من هذا المفكوك مع الجواب الذي نحصل عليه مباشرة باستخدام الدالة a^x (أي احسب القيمة المطلقة للفارق بين الجوابين).

٥- تستخدم الطرق التكرارية iterative techniques لحل مجموعة المعادلات الخطية. فمثلا بالنسبة للمجموعة التالية والمكونة من معادلتين فقط :

$$\left.\begin{aligned} 2x + y &= 3 \\ x - 3y &= 2 \end{aligned}\right\} \quad (*)$$

يمكن حلها باتباع الخطوات التالية :



(أ) أوجد x (بدلالة y) من المعادلة الأولى ، وأوجد y (بدلالة x) من المعادلة الثانية ، هكذا :

$$x = (3 - y) /2$$
$$y = (x - 2) /3$$

(ب) ابدأ بقيمة ابتدائية تقريبية للحل مثل :

XOLD = 1

YOLD = 1

(ج) احصل على قيمة أدق من هذه القيمة التقريبية ، بحساب XNEW , YNEW كما يلي :

$$\left.\begin{aligned} XNEW &= (3 - YOLD) / 2 \\ YNEW &= (XOLD - 2) / 3 \end{aligned}\right\} \quad (**)$$

(د) كرر هذه الطريقة بجعل القيمة التقريبية الجديدة تصبح قيمة تقريبية قديمة

XOLD = XNEW
YOLD = YNEW

وحساب قيمة جديدة لكل من XNEW , YNEW بدلالة XOLD , YOLD وذلك باستخدام العلاقتين (**).

(هـ) استمر في تكرار الخطوة السابقة (رقم د) إلى أن تتحقق – تقريبا – المعادلتان الأصليتان (*) حين تعويض XNEW , YNEW فيهما (والتقريب هنا يعتمد على الدقة المطلوبة) ، أي إلى أن يتحقق لدينا :

$$2\ XNEW + YNEW \equiv 3$$
$$XNEW - 3\ YNEW \equiv 2$$

وبصورة أخرى : إلى أن يتحقق الشرطان :

$$|2\ XNEW + YNEW - 3| < \varepsilon$$
$$|\ XNEW - 3\ YNEW - 2| < \varepsilon$$

حيث ε درجة الدقة المطلوبة (مثلا $\varepsilon = 10^{-4}$).

اكتب برنامجا يقرأ قيم الثوابت a, b, c, d, e, f, ε ويستخدم الطريقة المشروحة في هذا السؤال لحل المعادلتين الخطيتين التاليتين :

$$a\,x + b\,y = c$$
$$d\,x + e\,y = f$$

٦- أفتى العلماء بأنه لا يجوز شرعا في التعاقد العام – كما في قطاعات التعليم والصحة مثلا – استخدام أجيرين بأجرين مختلفين مع تساويهما في الكفاءة والمؤهلات والجهد والخدمات ، بل يجب تحقيق العدل المطلق الكامل بينهما تحقيقا للمصلحة العامة ، أما المخالفة بينهما لاعتبارات شخصية أو عائلية أو قومية أو جنسية فإنها توقظ الأحقاد وتمزق الأمة ، والله سبحانه

وتعالى يقول : {وأُمِرْتُ لأعدل بينكم} ، وهو سبحانه خلق الناس شعوبا وقبائل ليتعارفوا لا ليتخذوا ذلك أساسا للتمييز العنصري.

نفرض أنه قد أُعِدَّ سجل (بطاقة) لكل مدرس (أو مُدَرِّسة) في مرحلة تعليمية معينة في وزارة التربية ، حيث وضع على السجل :

رقم تعريفي ID للمدرس

وحرف يشير إلى جنسه (M للذكر وF للأنثى)

وراتبه الشهري S بالدينار.

وأضيف في النهاية سجل يحمل قيمة سالبة للمتغير S ليشير إلى انتهاء سجلات البيانات.

اكتب برنامجا يقرأ هذه المجموعة من السجلات ويوجد :

(أ) عدد المدرسين والمدرسات (N1) الذين يحصلون على راتب شهري أقل من ٢٥٠ دينارا.

(ب) عدد المدرسين والمدرسات (N2) الذين يحصلون على راتب شهري أعلى من ٤٠٠ دينارا.

(ج) العدد الإجمالي للمدرسين والمدرسات (N)

(د) النسبة المئوية لكل من المدرسين والمدرسات من العدد الإجمالي.

٧- نفرض أن A منظومة فيها أربعون قيمة حقيقية. المطلوب كتابة برنامج :

(أ) يقرأ قيم عناصر المنظومة A.

(ب) يوجد أقرب قيمة – في المنظومة A – للقيمة المتوسطة AV (التي تساوي مجموع كل القيم مقسوما على عددها).

٨- اكتب برنامجا لإيجاد جميع الأزواج المرتبة (i , j) التي تحقق آنيا المتباينات التالية ، حيث كل من i , j عدد صحيح :

$$2i - j < 3$$
$$i + 3j \geq 1$$
$$-6 \leq i \leq 6$$
$$10 \leq j \leq 10 \quad -$$

٩- إذا عُلِّقت كتلة من زنبرك فإنها تتذبذب بتردد يُعْطَى بالعلاقة :

$$F = 0.1592\sqrt{K / M}$$

حيث :

F : التردد

K : ثابت الزنبرك

M : الكتلة

والمطلوب كتابة برنامج يحسب ويطبع قيم التردد F وذلك لقيم K المختلفة: ١ ، ٣ ، ٥ ، ... ، ١١. وعند كل قيمة من قيم K فإن M تأخذ القيم: ١, ، ٢, ، ٣, ، ... ، ١. وبحيث يطبع النتائج في شكل جدول من ثلاثة أعمدة تعطي قيم K, M, F ، وبحيث يطبع F صحيحا لثلاثة أرقام عشرية فقط.

١٠- حث الإسلام على الزواج المبكر طلبا للإحصان والعفاف ، ولكن بعض عادات المجتمع وتقاليده تحول دون تحقيق ذلك كعادات التغالي في المهور ، والتباهي بالأثاث والدور والقصور ، والتكالب على المظاهر الجوفاء والمتاع الدنيوي الزائل ، وحب المال حبا جما ، والعصبية القبلية التي تمنع الزواج إلا من عائلات معينة ، وضرورة الحصول على شهادات تعليمية ودراسات عليا أولا ، ... الخ. نفرض أن لدينا مجموعة من سجلات البيانات في ملف حيث يشتمل كل سجل على البيانات التالية :

اسم الشخص (NAME).

حرف يشير إلى الجنس (M لذكر وF للأنثى)

عمر الشخص (AGE).

رقم يشير إلى حالته الزوجية (١ للأعزب و ٢ للمتزوج)

وعـدد هـذه السـجلات غـير معلـوم ، وقـد اسـتخدم الـرقم ٩ في الموضـع المخصص للحالة الزوجية ليشير إلى نهاية الملف.

ارسم خريطة سير عمليات ، واكتب برنامجا لحساب وطباعة ما يلي :

(أ) النسبة المئوية للذكور (PM) ، والنسبة المئوية للاناث (PF).

(ب) العدد الإجمالي للذكور فوق سن السادسة عشرة (N). والنسبة المئوية – من هذا العدد N – للذكور الأيامى [*] فوق سن الخامسة والعشرين (PUM).

(ج) العدد الإجمالي للنساء فوق سن السادسة عشرة (M) ، والنسبة المئوية – من هذا العدد M – للنساء الأيامى فوق سن الثانية والعشرين (PUF).

١١- نفرض أن كلا من X , Y منظومة مكونة من عشرين عنصرا. اكتب برنامجا

(أ) يقرأ قيمة عدد صحيح N.

(ب) يقرأ قيم أول N عنصر من كل من المنظومتين X , Y.

(ج) يحسب قيم N عنصر من منظومة Z تبعا للقاعدة التالية :

قيمة العنصر z_i تساوي ١ أو صفر أو –١ إذا كانت قيمة العنصر x_i أكبر من أو تساوي أو أصغر من قيمة العنصر y_i على الترتيب.

(د) يطبع جدولا من ثلاثة أعمدة يعطي القيم المتقابلة لعناصر المنظومات X , Y , Z.

(ه) يطبع عدد عناصر المنظومة X التي تزيد قيمة كل عنصر منها عن قيمة العنصر المقابل في المنظومة Y. وكذلك عدد عناصر المنظومة X التي تقل قيمة كل عنصر منها عن قيمة العنصر المقابل في المنظومة Y.

(*) الأيامى : جمع أَيِّم ، وهو الرجل الذي لا زوجة له ، أو المرأة التي لا زوج لها.

١٢- من المعلوم أن مجموع مربعات الأعداد الصحيحة من ١ إلى N يساوي

$$1^2 + 2^2 + 3^2 + \cdots + N^2 = \frac{N(N+1)(2N+1)}{6}$$

اكتب برنامجا :

(أ) يعرف دالة SUMSQ (N) تعطي قيمة هذا المجموع.

(ب) يقرأ قيمة عدد صحيح LIMIT.

(ج) يستخدم الدالة المعرفة SUMSQ في حساب مجموع مربعات الأعداد الصحيحة من ١ إلى M وذلك لكل قيمة من قيم M الصحيحة ابتداء من ٢ وحتى قيمة LIMIT.

١٣- تتكون المنظومة A من عدد - لا يزيد عن مائة - من الأعداد الصحيحة غير الصفرية ، وبحيث أن كل عدد منها مكتوب على سطر مستقل.
ويستخدم العدد صفر كقيمة تشير إلى انتهاء عناصر المنظومة.
اكتب برنامجا لقراءة عناصر هذه المنظومة وإيجاد :

(أ) عدد العناصر N في المنظومة.

(ب) عدد العناصر الزوجية NE ، وعدد العناصر الفردية NO في المنظومة.

(ج) عدد العناصر مضاعفات العدد ٣ (أي العناصر التي يقبل كل منها القسمة على ٣ بدون باق) ومجموع قيمها.

١٤- (أ) اكتب برنامجا فرعيا يحذف كل العناصر الصفرية من منظومة X تحتوي على N عدد حقيقي ، ويحتفظ بالعناصر الأخرى من X في منظومة اسمها Y.

(ب) اكتب برنامجا رئيسيا :

١- يقرأ عناصر منظومة A فيها L عدد حقيقي.
ويقرأ عناصر منظومة Z فيها M عدد حقيقي.

٢- يستخدم البرنامج الفرعي - في (أ) - ليحذف العناصر الصفرية في المنظومة A ويخزن العناصر الأخرى في منظومة ANEW ، وكذلك ليحذف العناصر الصفرية في المنظومة Z ويخزن العناصر الأخرى في منظومة ZNEW.

٣- يطبع فقط المنظومتين ANEW , ZNEW مستخدما عناوين مناسبة.

١٥- منحنى المعادلة $x^2 + y^2 = 50$ عبارة عن دائرة مركزها نقطة الأصل ونصف قطرها يساوي $\sqrt{50}$. اكتب برنامجا يوجد :

(أ) النقاط ذوات الإحداثيات الصحيحة التي تقع داخل الدائرة.

(ب) عدد هذه النقاط. لاحظ أن قيمة كل من x , y لن تزيد عن 7 لأن نصف القطر يساوي $\sqrt{50}$.

١٦- اكتب برنامجا يقرأ قيمة عدد صحيح N (حيث $N \leq 20$) ، وكذلك إحداثيات رؤوس مُضَلَّع (polygon) عدد رؤوسه N ، ويُخَزِّن هذه الإحداثيات في منظومتين X , Y. ثم يحسب ويطبع :

(أ) مساحة المضلع والتي تعطى بالعلاقة :

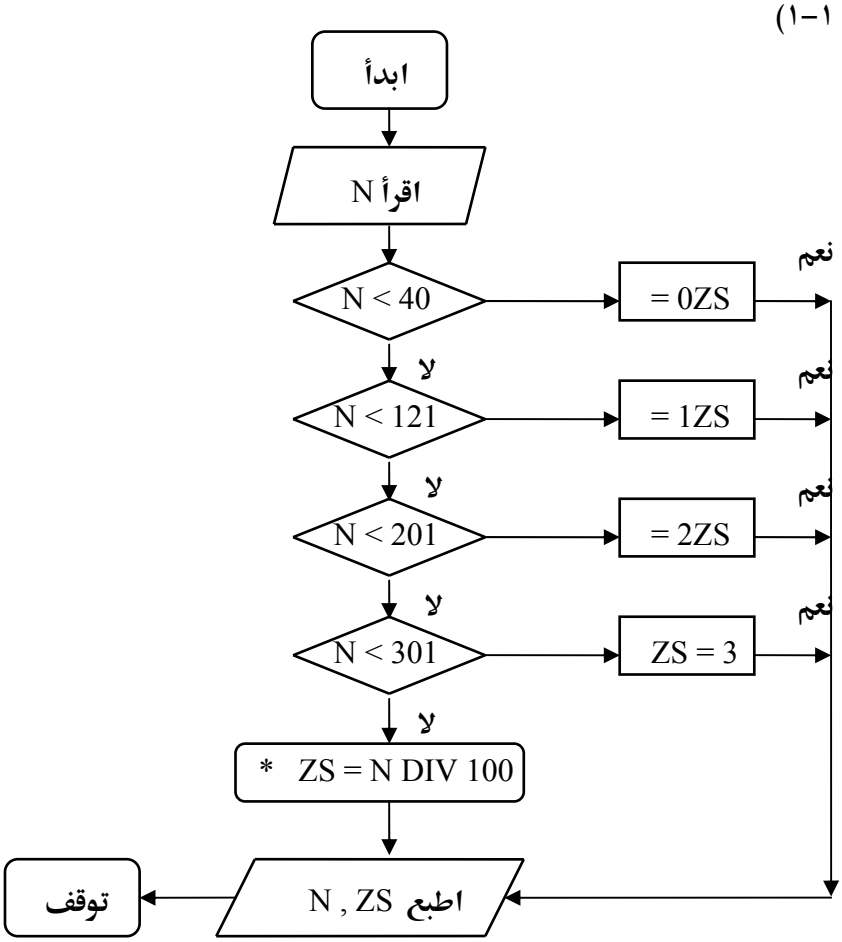
$$AREA = \frac{1}{2}\left[y_1 x_N - x_1 y_N\right] + \sum_{i=2}^{N}\left(y_i x_{i-1} - x_i y_{i-1}\right)$$

(ب) إحداثيي مركز ثقله (Center of gravity) ، ويعطيان بالعلاقتين :

$$XG = \left(\sum_{i=1}^{N} x_i\right) / N \quad , \qquad YG = \left(\sum_{i=1}^{N} y_i\right) / N$$

أجوبة تمرينات رقم ١

١-١)



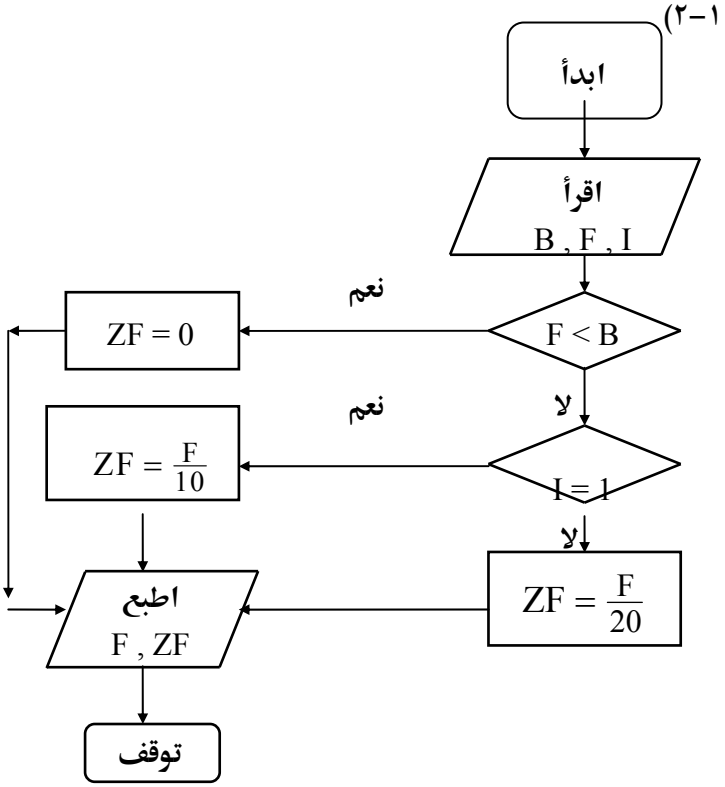
* ملاحظة : يمكن التعبير رياضيا عن العبارة : ZS تساوي في كل مائة شاة بالعلاقة :

ZS = N DIV 100 والتي تعطي العدد الصحيح فقط في خارج قسمة $\frac{N}{100}$ مع حذف الكسور العشرية ، وذلك لأن هذه العلاقة تعطي النتيجة التالية :

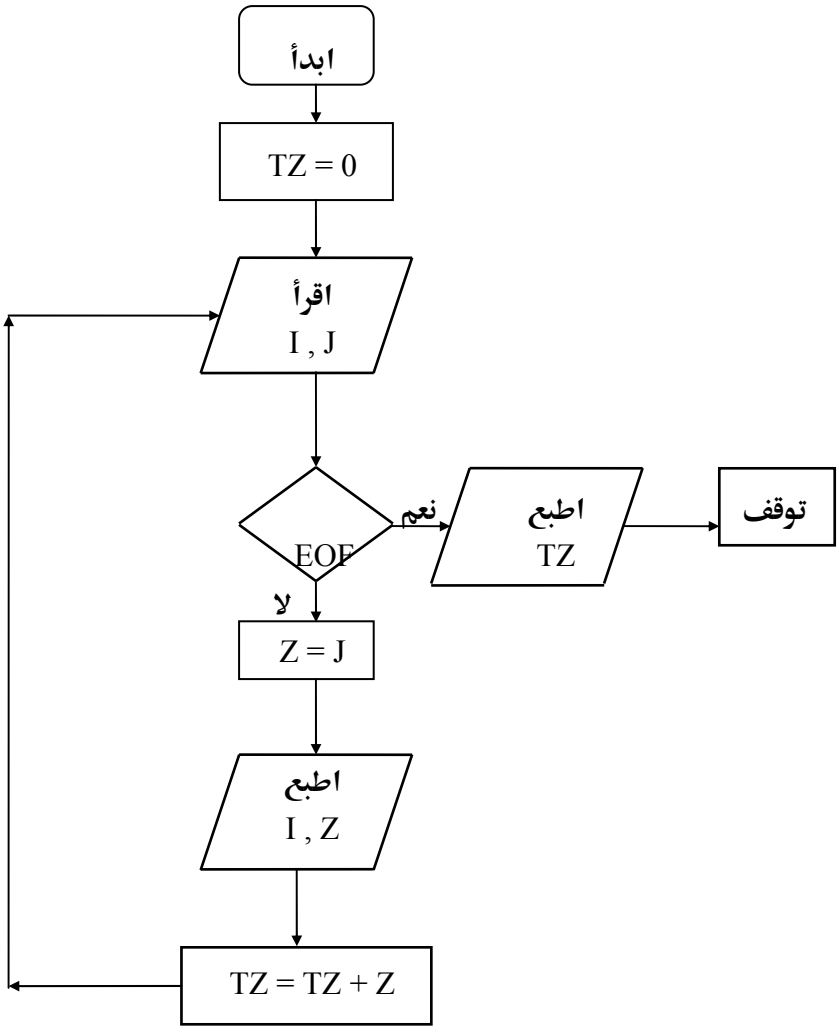
N	٣٠١-٣٩٩	٤٠٠-٤٩٩	٥٠٠-٥٩٩	٠٠٠

ZS	٣	٤	٥	٠٠٠

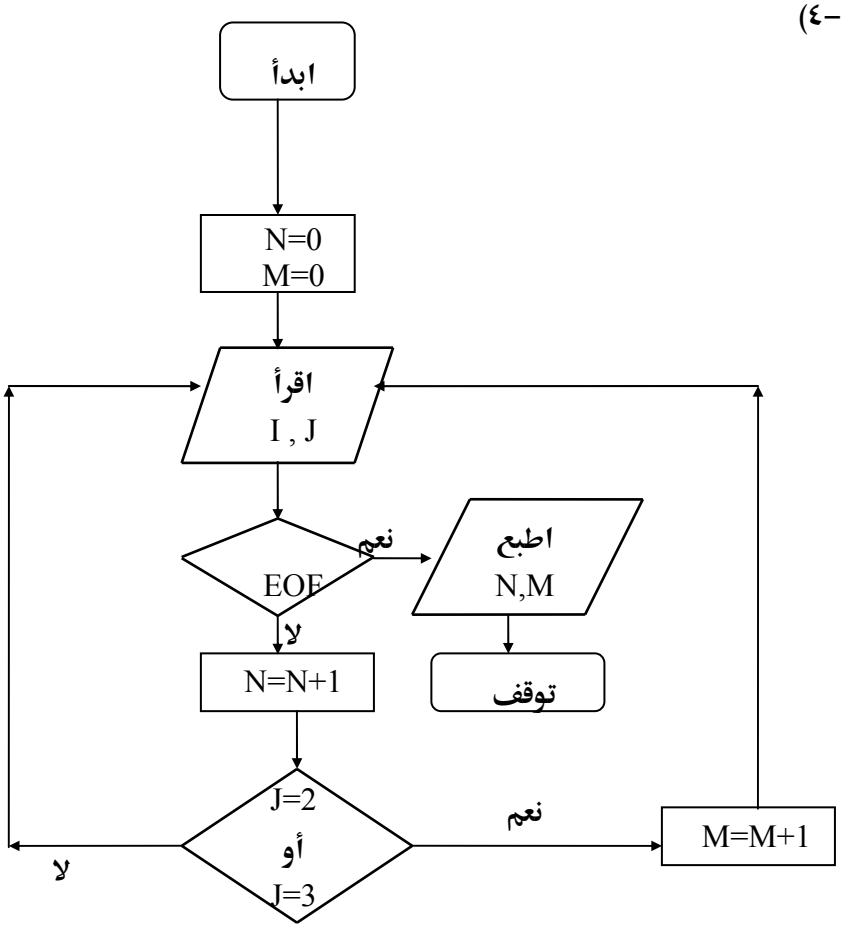
١-٢)



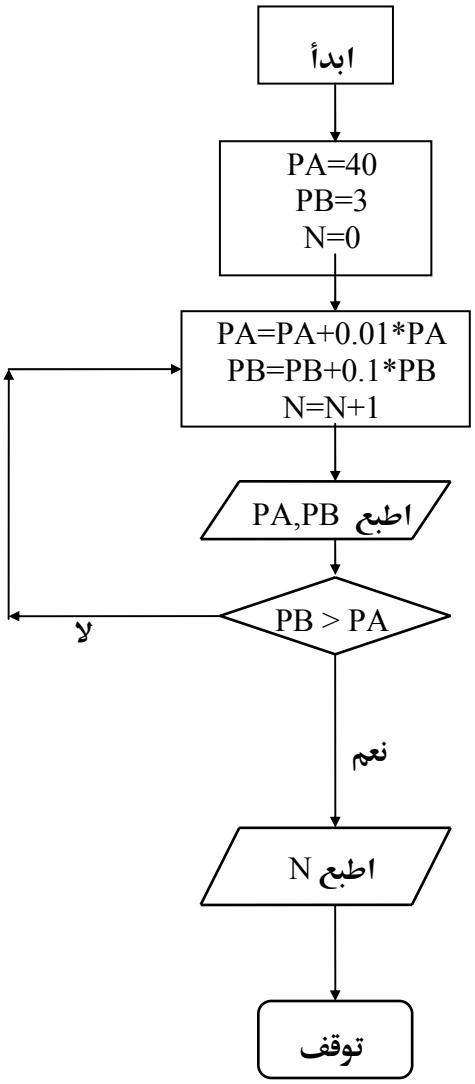
١-٣)



١-٤)



١-٥)

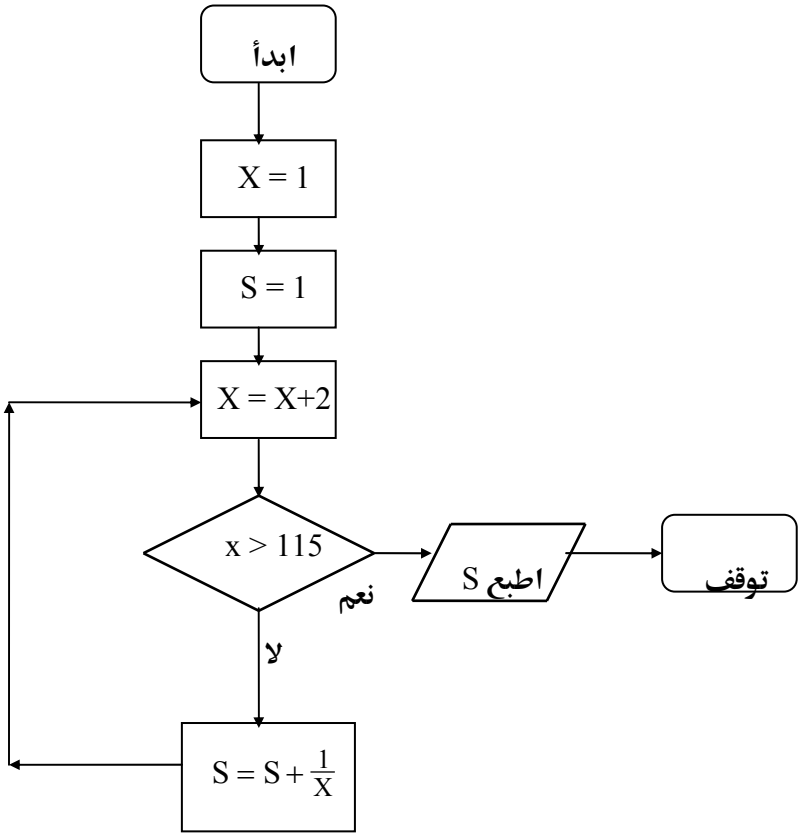


ملاحظة : يمكن أيضا طباعة عدد السنوات N – كل سنة – مع PA , PB.

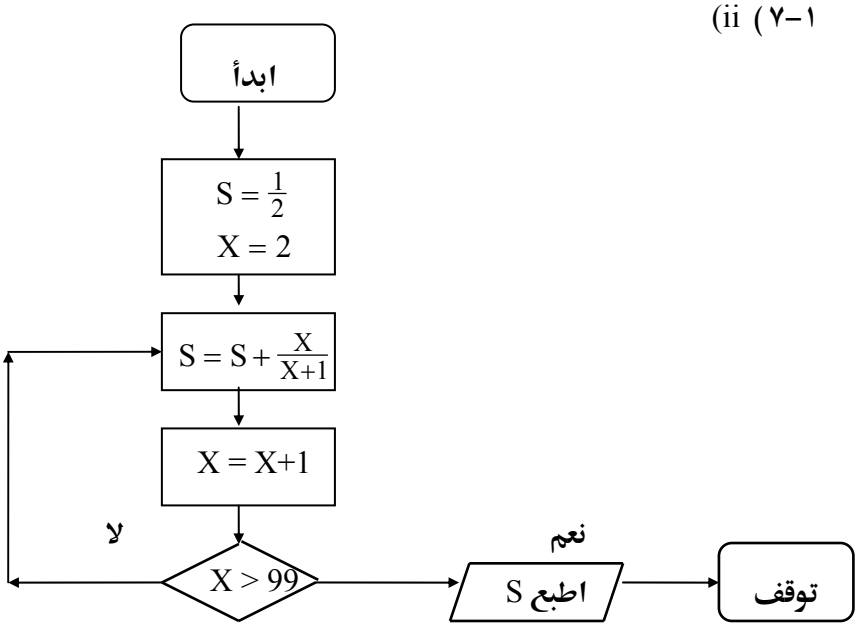
(١-٦

i) 5 ii) 8 iii) 10 iv) 5

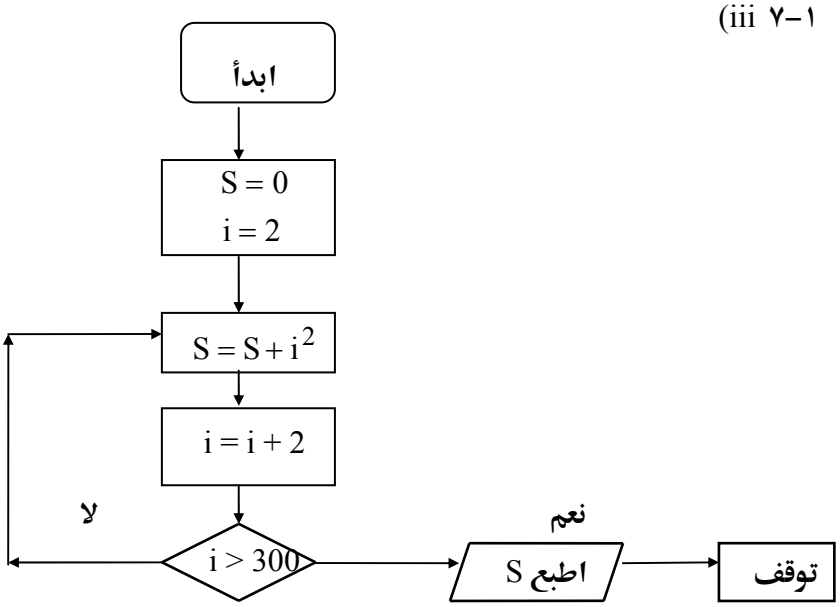
(i (١-٧



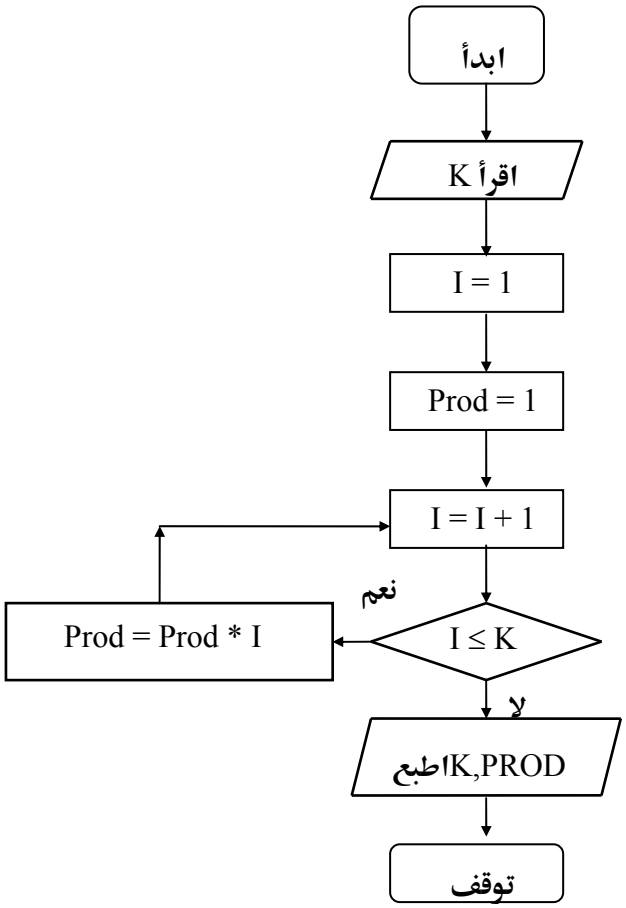
(ii (٧-١


ابدأ
$S = \frac{1}{2}$
$X = 2$
$S = S + \frac{X}{X+1}$
$X = X+1$
لا
$X > 99$
نعم
اطبع S
توقف

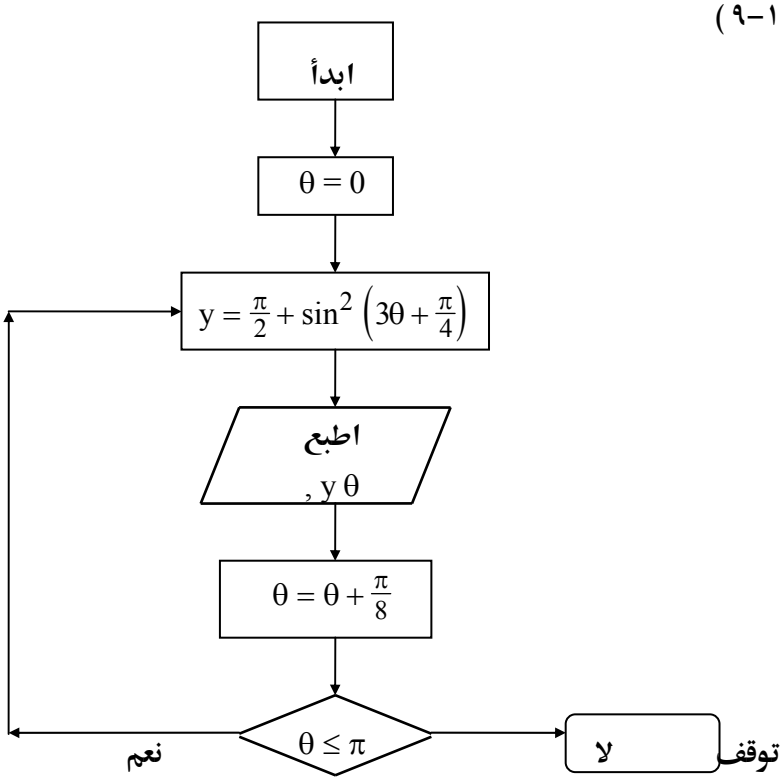
(iii ٧-١


ابدأ
$S = 0$
$i = 2$
$S = S + i^2$
$i = i + 2$
لا
$i > 300$
نعم
اطبع S
توقف

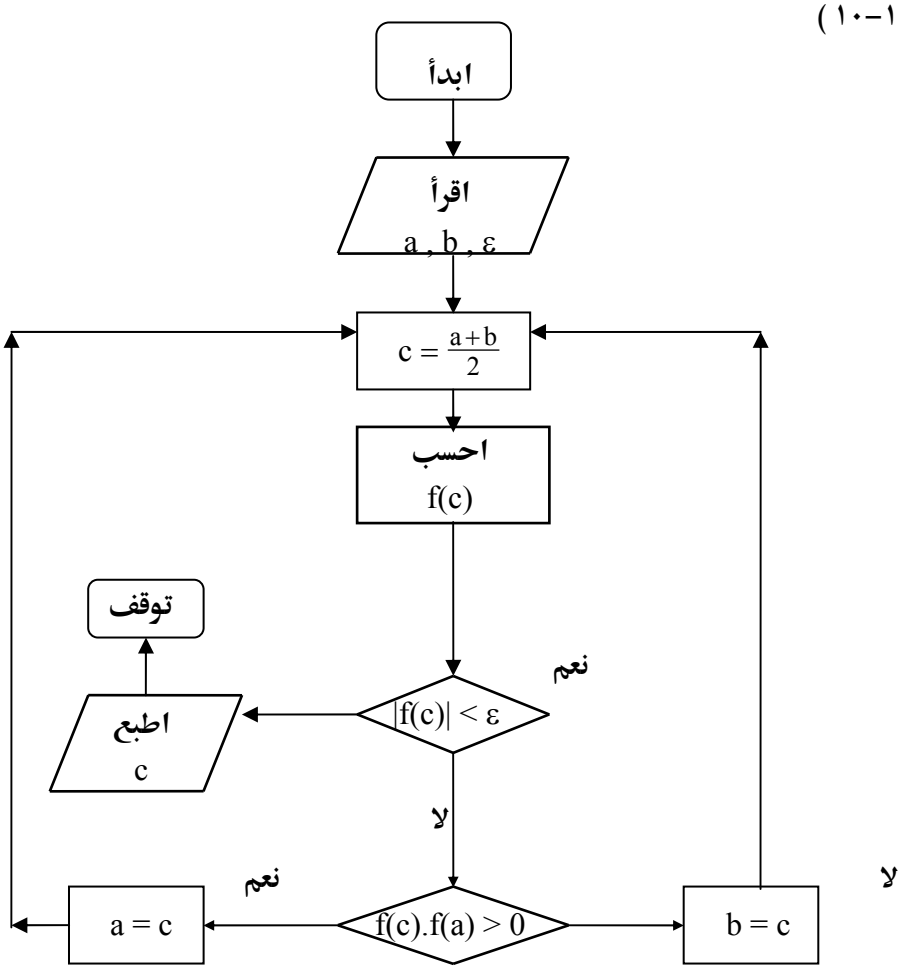
(١-٨



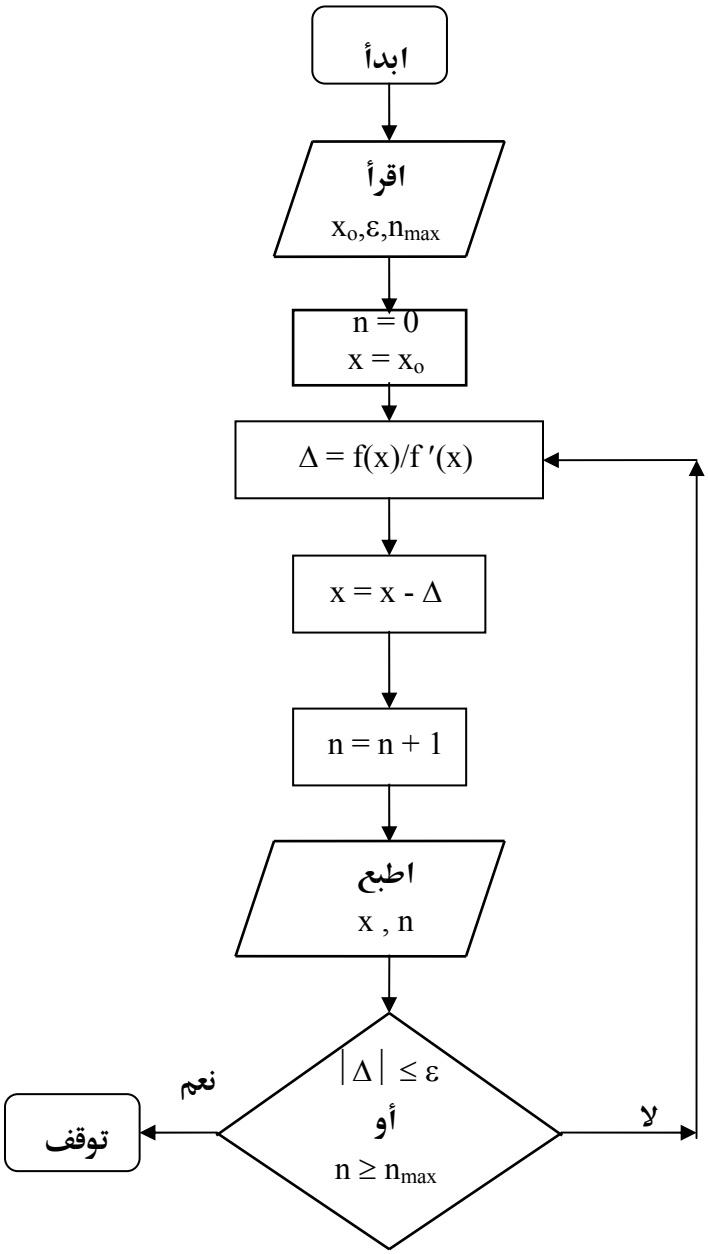
(٩-١



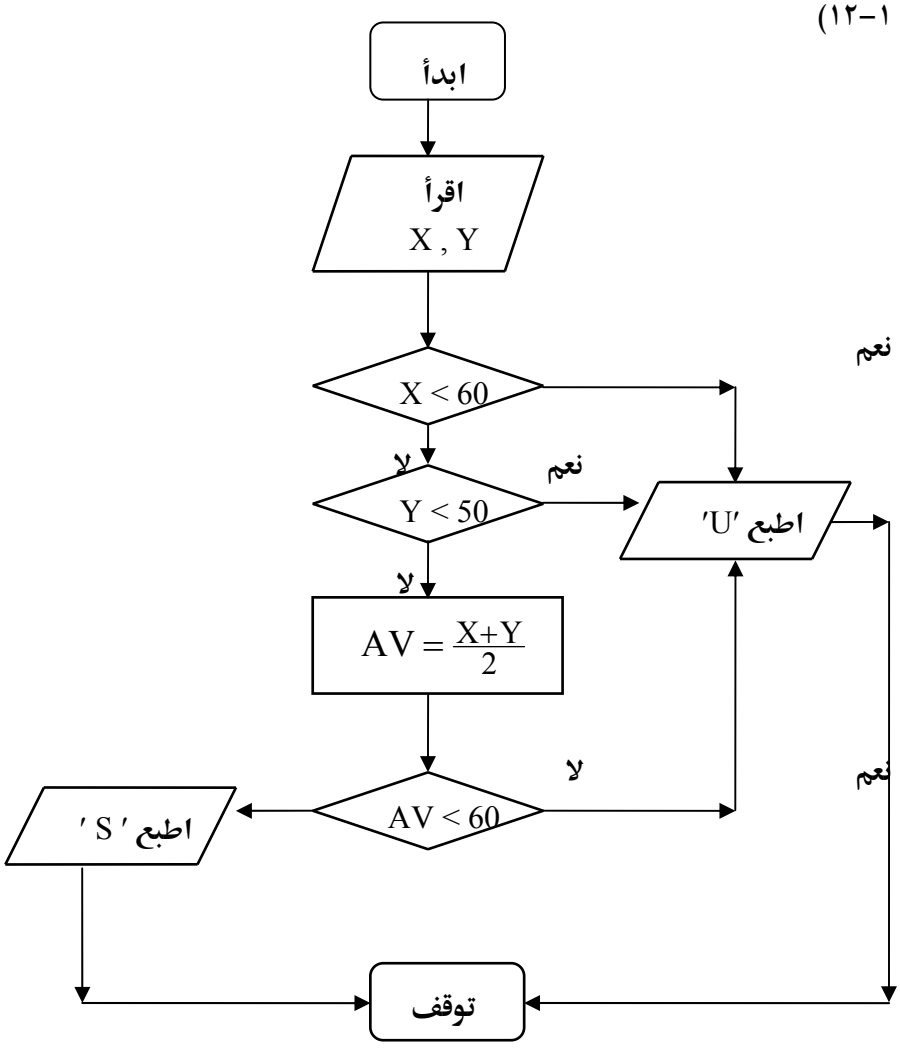
١-١٠)


ابدأ
اقرأ
a , b , ε
$c = \frac{a+b}{2}$
احسب
f(c)
توقف
نعم
اطبع
c
|f(c)| < ε
لا
نعم
لا
a = c
f(c).f(a) > 0
b = c

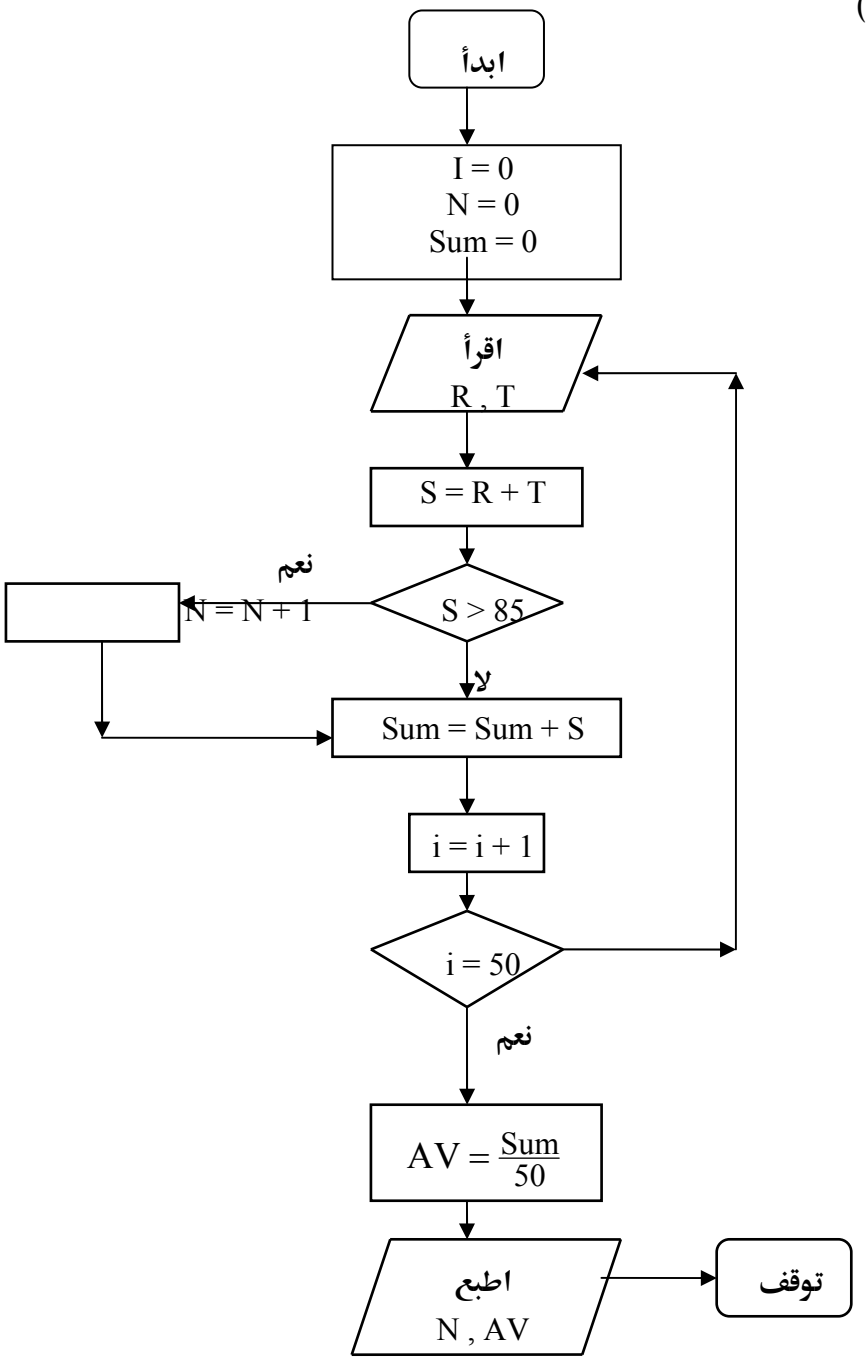
١-١١)


ابدأ
اقرأ
x_o,ε,n_{max}
n = 0
$x = x_o$
$\Delta = f(x)/f'(x)$
$x = x - \Delta$
n = n + 1
اطبع
x , n
$|\Delta| \le \varepsilon$
أو
$n \ge n_{max}$
نعم
توقف
لا

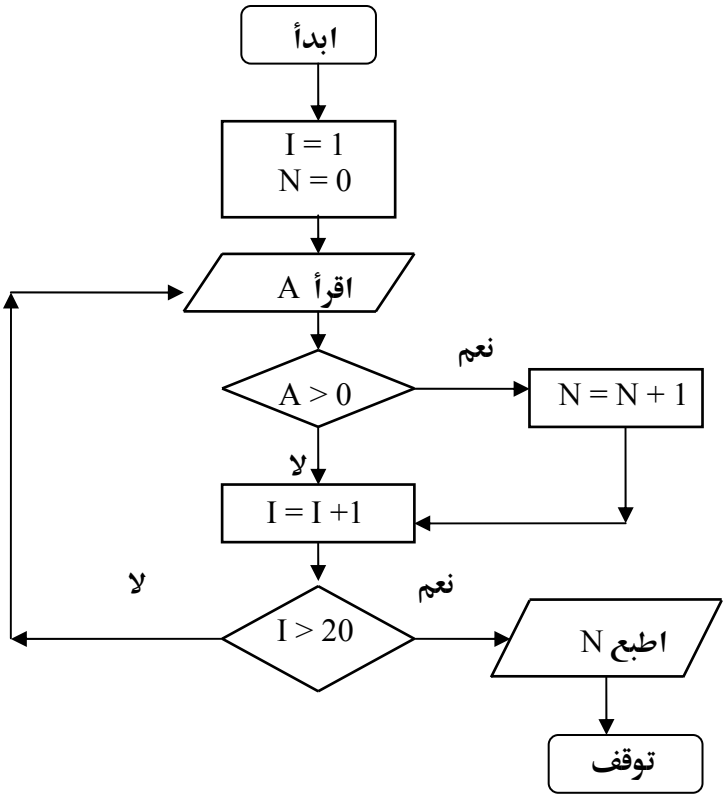
١-١٢)


ابدأ
اقرأ
X , Y
X < 60
نعم
لا
Y < 50
نعم
اطبع 'U'
لا
AV = (X+Y)/2
لا
AV < 60
نعم
اطبع ' S '
توقف

١-١٣)

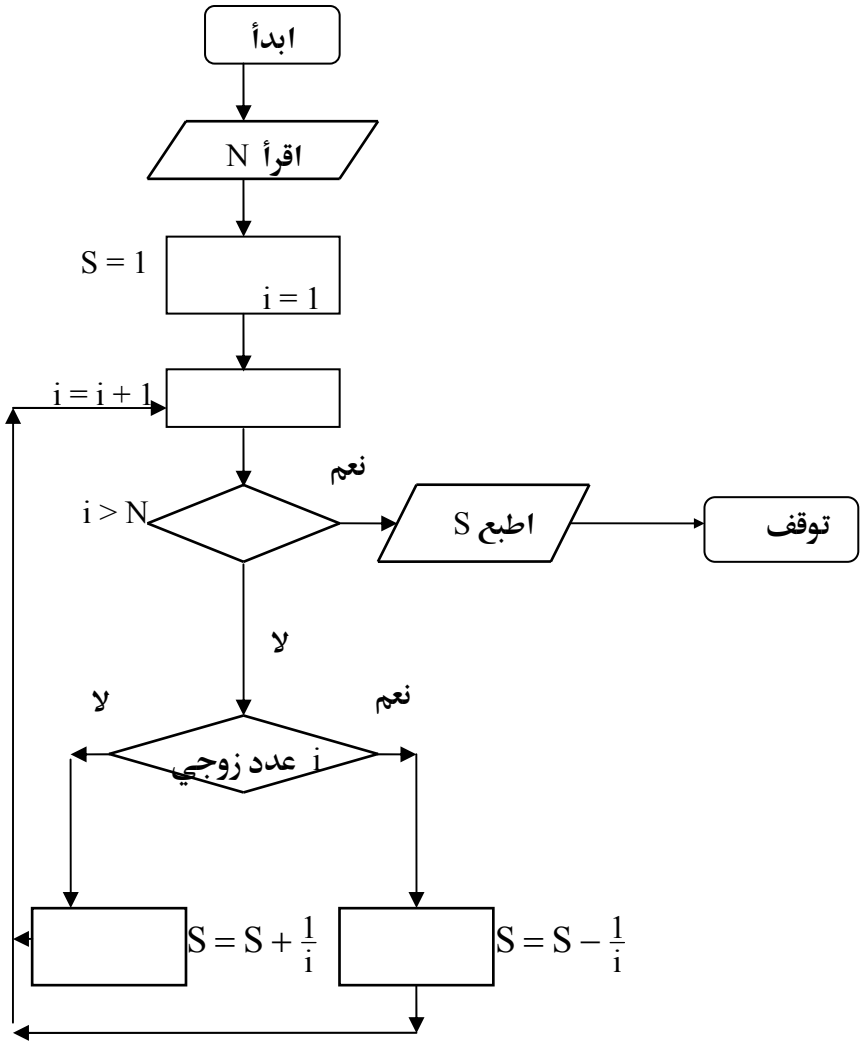

ابدأ
I = 0
N = 0
Sum = 0
اقرأ
R , T
S = R + T
نعم
N = N + 1
S > 85
لا
Sum = Sum + S
i = i + 1
i = 50
لا
نعم
AV = Sum / 50
اطبع
N , AV
توقف

١-١٤)

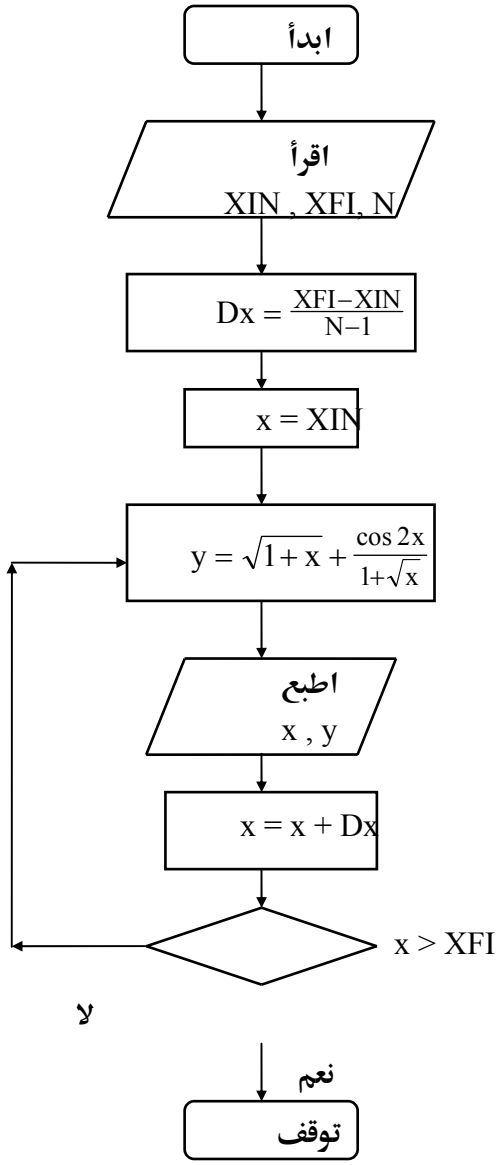


الخوارزمية تحسب عدد العناصر الموجبة في مجموعة مكونة من عشرين عددا.

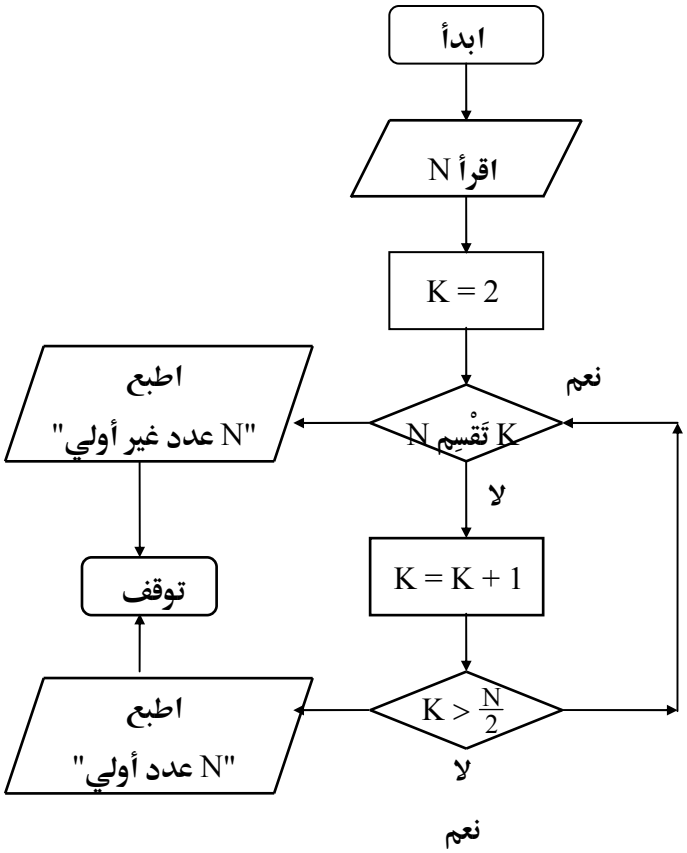
١-١٥)


ابدأ
اقرأ N
S = 1
i = 1
i = i + 1
نعم
i > N
اطبع S
توقف
لا
لا
نعم
i عدد زوجي
S = S + 1/i
S = S - 1/i

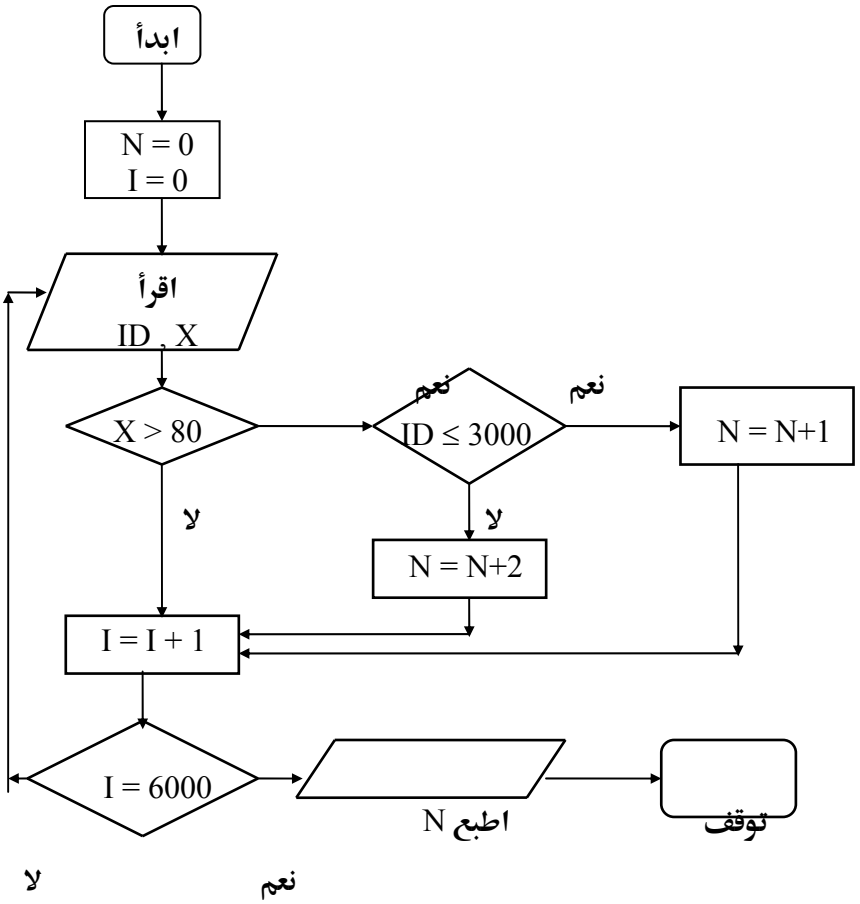
١-١٦)


ابدأ
اقرأ
XIN , XFI, N
Dx = (XFI−XIN)/(N−1)
x = XIN
y = √(1+x) + cos 2x/(1+√x)
اطبع
x , y
x = x + Dx
x > XFI
لا
نعم
توقف

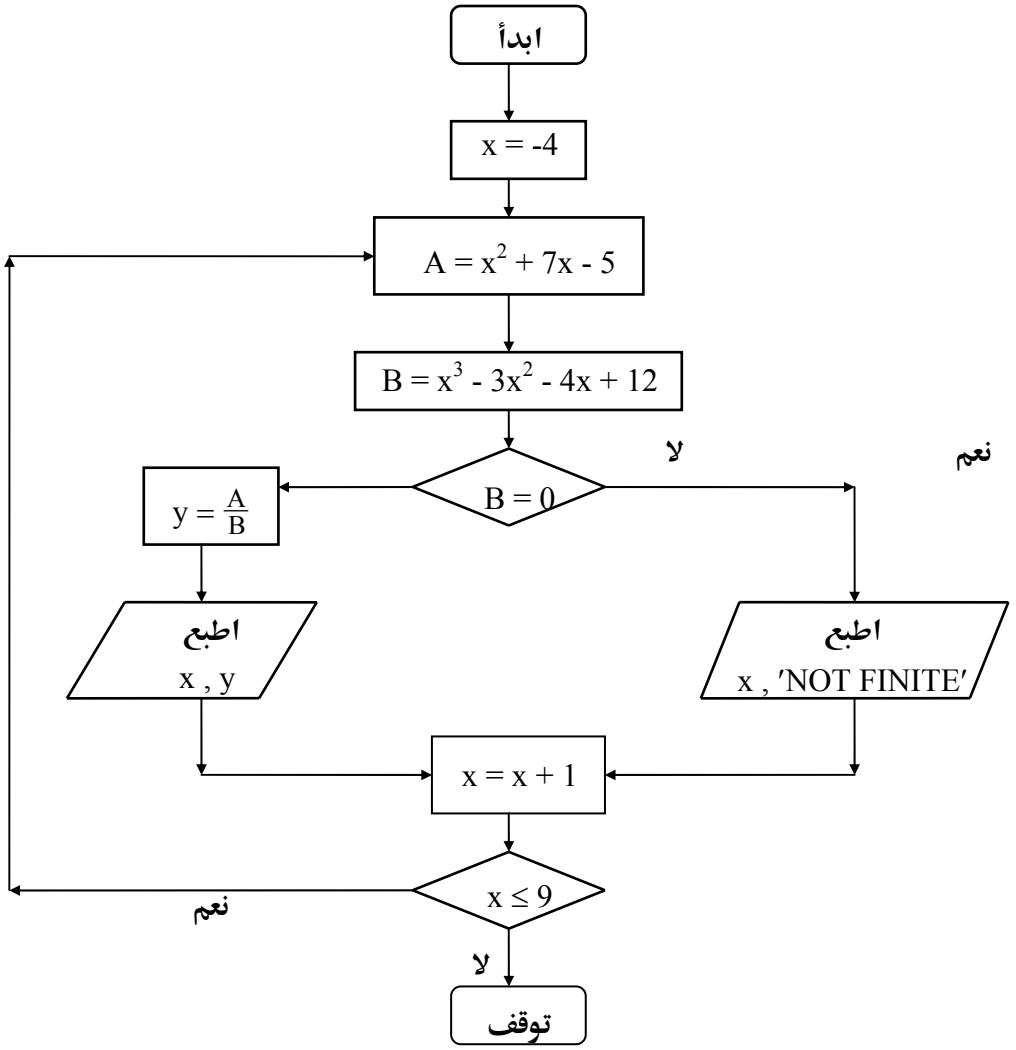
١-١٧)


ابدأ
اقرأ N
K = 2
نعم
K تَقْسِم N
اطبع
"N عدد غير أولي"
لا
K = K + 1
توقف
K > N/2
اطبع
"N عدد أولي"
لا
نعم

١-١٨)


ابدأ
N = 0
I = 0
اقرأ
ID , X
X > 80
نعم
ID ≤ 3000
نعم
N = N+1
لا
N = N+2
لا
I = I + 1
I = 6000
اطبع N
توقف
لا
نعم

١-١٩)


ابدأ
x = -4
A = x^2 + 7x - 5
B = x^3 - 3x^2 - 4x + 12
B = 0
لا
نعم
y = A/B
اطبع
x , y
اطبع
x , 'NOT FINITE'
x = x + 1
x ≤ 9
نعم
لا
توقف

١-٢٠)

١- اقرأ قيمة A

٢- اجعل $S = 1$

٣- اجعل $n = 0$

٤- $n = n + 1$

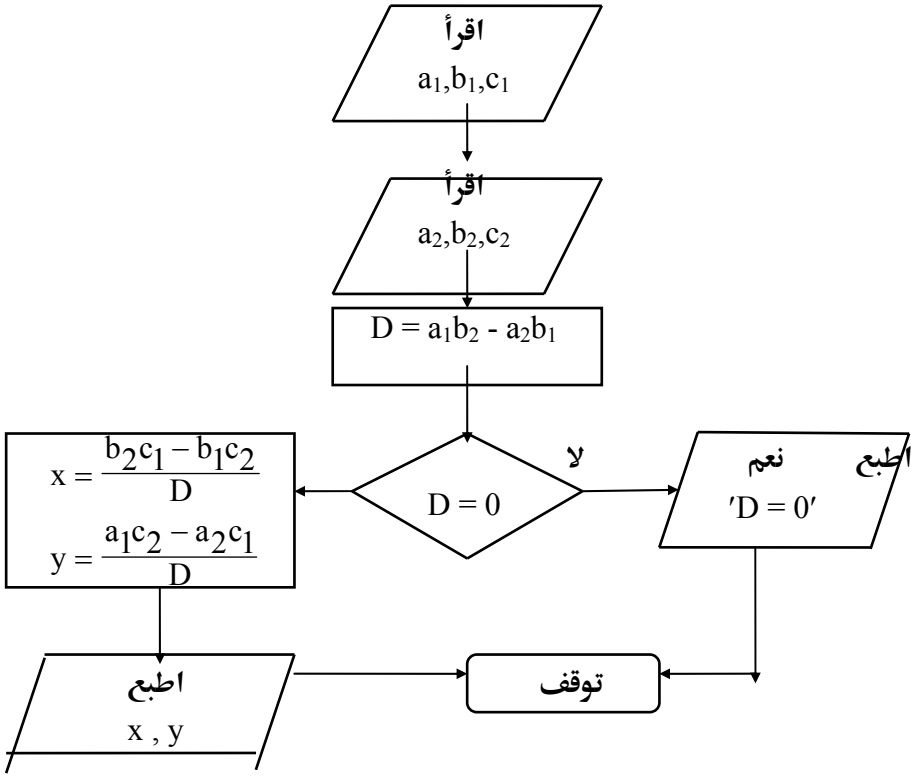
٥- $S = S + 1 + n\ A$

٦- إذا كانت $n \leq 48$ اذهب إلى ٤

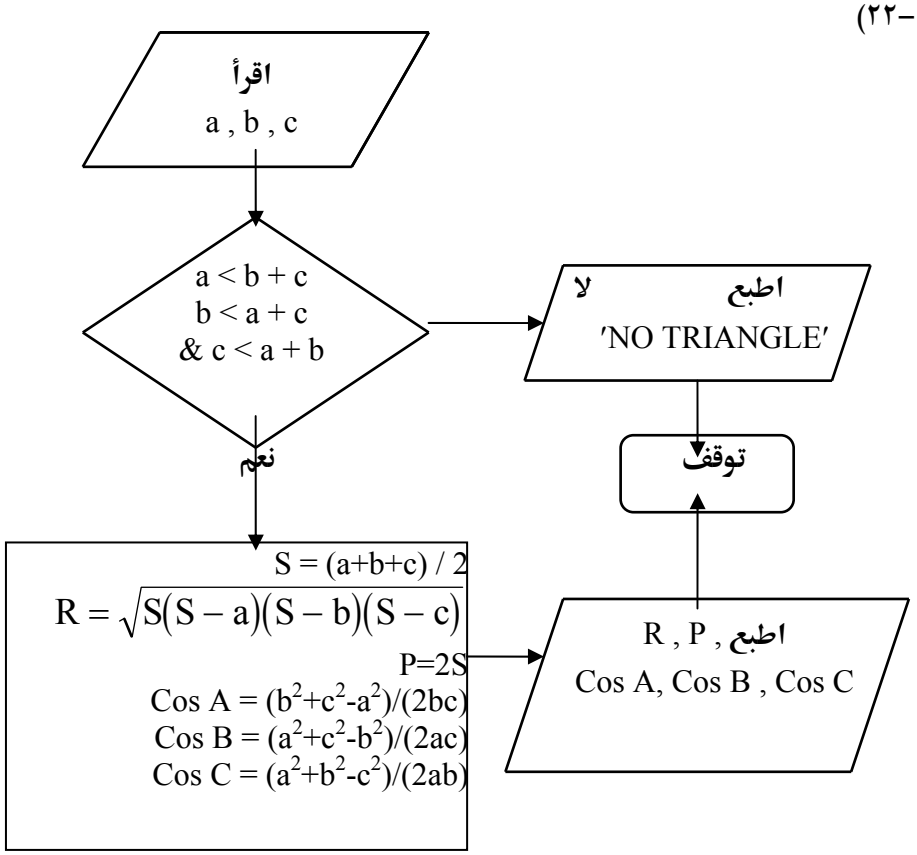
٧- اطبع S

٨- توقف

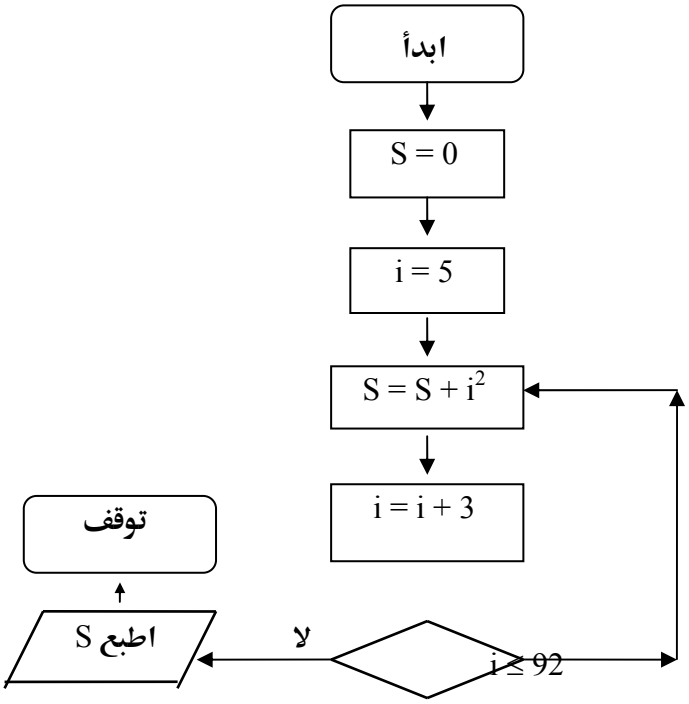
١-٢١



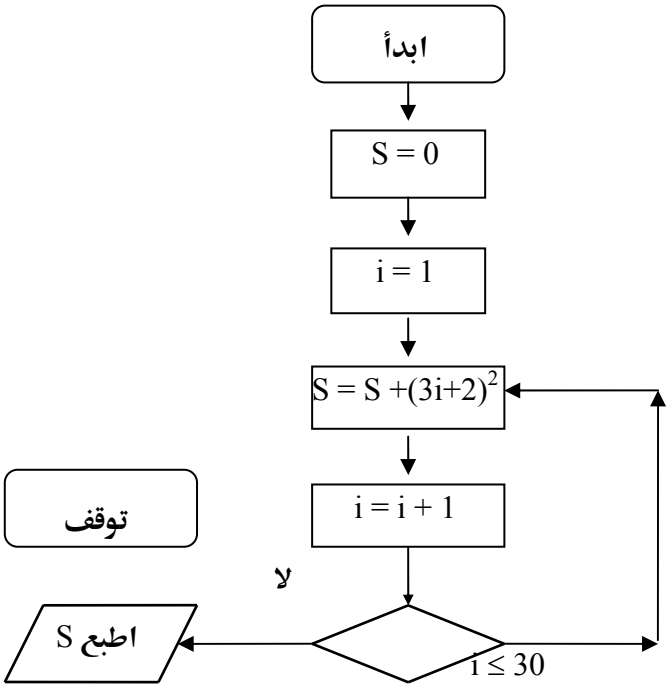
(١-٢٢


اقرأ
a , b , c
a < b + c
b < a + c
& c < a + b
لا
اطبع
'NO TRIANGLE'
توقف
نعم
S = (a+b+c) / 2
$R = \sqrt{S(S-a)(S-b)(S-c)}$
P=2S
Cos A = (b²+c²-a²)/(2bc)
Cos B = (a²+c²-b²)/(2ac)
Cos C = (a²+b²-c²)/(2ab)
اطبع , R , P
Cos A, Cos B , Cos C

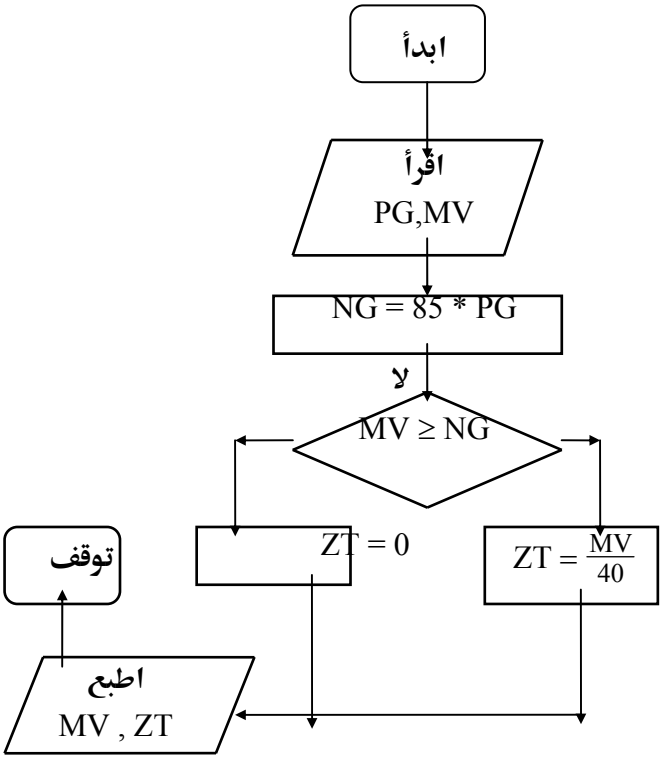
١-٢٣)


ابدأ
S = 0
i = 5
S = S + i²
i = i + 3
i ≤ 92
نعم
لا
اطبع S
توقف

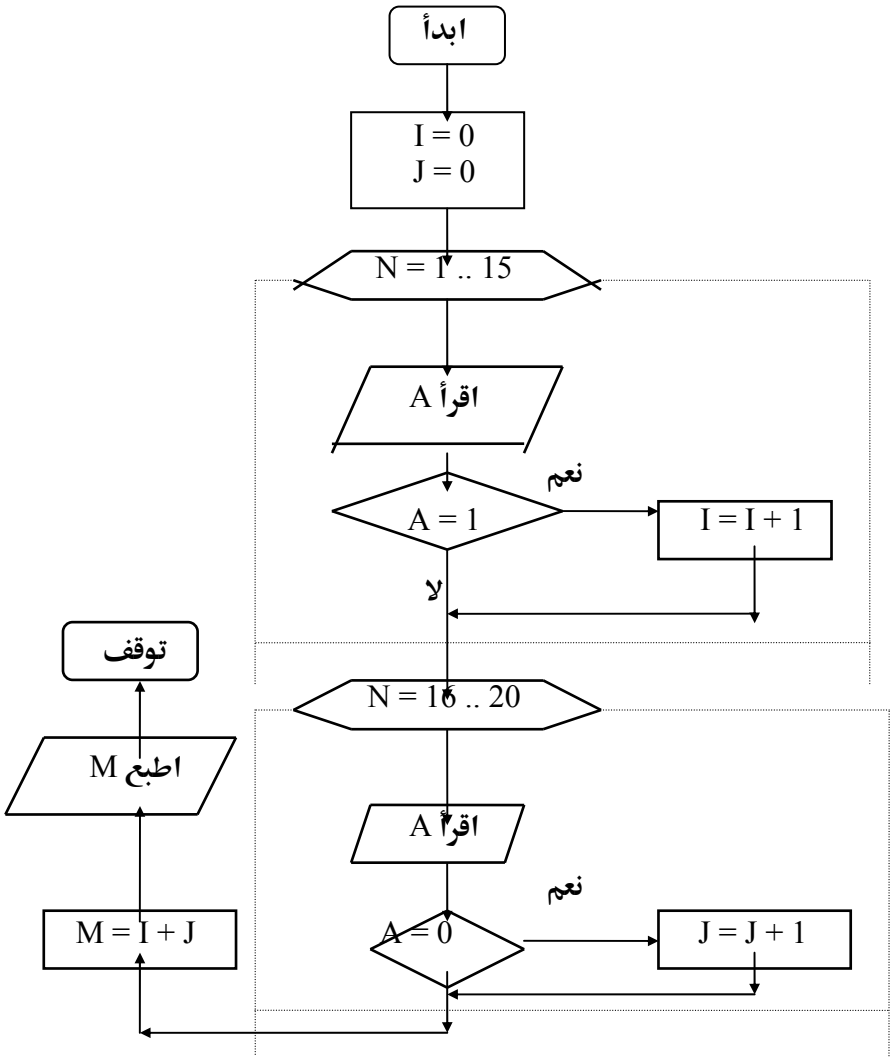
حل آخر


ابدأ
S = 0
i = 1
S = S +(3i+2)²
i = i + 1
i ≤ 30
نعم
لا
اطبع S
توقف

١-٢٤)



١-٢٥)


ابدأ
I = 0
J = 0
N = 1 .. 15
اقرأ A
نعم
A = 1
I = I + 1
لا
N = 16 .. 20
اقرأ A
نعم
A = 0
J = J + 1
M = I + J
اطبع M
توقف

أجوبة تمرينات رقم ٢

٢-١) The Companion is KhlidIbnElWleed

٢-٢) أ) غير صحيح ب) صحيح ج) صحيح
د) غير صحيح هـ) صحيح و) غير صحيح
ز) صحيح ح) غير صحيح ط) صحيح

٢-٣- i) أ) كلمة محجوزة ب) اسم تعريفي ج) اسم تعريفي
د) كلمة محجوزة هـ) اسم تعريفي و) اسم تعريفي
ز) اسم تعريفي ح) كلمة محجوزة ط) كلمة محجوزة

ii) لا يمكن استخدام الكلمات المحجوزة كأسماء متغيرات.

٢-٤) (أ) s1 = Lasts2 = Day

(ب) Believe in : LastDay

(ج) Believe in LastDay

(د) Believe in Last Day

٢-٥) Whoever guides someone
to do a good

□ ← سطر فارغ

will have reward equal to

□

ثلاثة سطور فارغة

the reward of the doer

٢ -٦ - أ) رمز واحد.

ب) صفر. سلسلة الرموز الفارغة سلسلة خالية (empty string).

ج) لا.

د) نعم.

هـ) نعم . سلسلة الرموز الحرفية "computer" عبارة عن قيمة سلسلة رموز (string value) بينما الاسم التعريفي computer عبارة عن اسم name يمكن أن تُسند إليه قيمة.

٢-٧)

142B Mousa Ibn Nosair
Cordova, Andalusia

٢-٨) الأخطاء التركيبية:

- ناقص ">" في < iostream
- يجب وضع "=" بدلاً من "." في الإعلانات عن سلاسل الرموز الثابتة.
- يجب وضع "const" بدلاً من كل كلمة "constant"
- يجب وضع الحاصرات المزدوجة "double quotes" حول جميع قيم سلاسل الرموز المسندة.
- ناقص قوسان ( ) بعد كلمة main.
- يجب كتابة كلمة char بدلاً من كلمة character.
- العبارة name = ... تصحح إلى : name = FIRST + MID + LAST
- لا يجوز إعادة إسناد قيمة للثابت LAST لأنه ثابت
- عبارة count >> تصحح إلى cout >>
- الحاصرتان المفردتان في سلسلة المخرجات ' Name =' يجب أن يكونا مزدوجتين " Name = "
- يجب وضع فاصلة منقوطة ";" في نهاية عبارة الإخراج قبل الأخيرة.

٢-٩)

- نستبدل بعبارة const chr MIDDLE = 'M'; ….

العبارة

```
const string MIDDLE = 'M';          // Person's middle initial
```

- نضيف الإعلان

```
string fistMILast;              // Name in first-initial-last format
```

- نضيف العبارات

```
fistMILast = FIRST + "   " + MIDDLE + " . " + LAST;
cout << " Name in firstMILast formt is " ;
cout << firstMILaast << endl;
```

٢ – ١٠ – أ)

```
cout << "Allah the Almighty is good" << endl ;
cout << " and accepts only that which is good ";
```

ب)

```
cout << "Allah the Almighty is good" << endl ;
cout << " and accepts only that " << endl ;
cout << " which is good " << endl ;
```

ج)

```
cout << "Allah the Almighty is good" << endl ;
cout << " and " << endl ;
cout <<  endl ;
cout << " accepts only that " << endl ;
cout << " which is good " << endl ;
```

د)

```
cout << "Allah the Almighty " << endl ;
cout << "     is good " << endl ;
cout << "          and " << endl ;
cout << " accepts only that " << endl ;
cout << "    which " << endl ;
cout << "      is good " << endl ;
```

٢–١١) مخرجات البرنامج:

Peace be upon you.

Peace be upon you. Peace be upon you. Peace be upon you.

٢-١٢) (أ) خاطئة (ب) صحيحة (ج) خاطئة
(د) صحيحة (هـ) صحيحة

٢-١٣) (أ) ٢٧ (ب) ١٣ (ج) ٥
(د) ٠ (هـ) ٣ (و) ٨
(ز) ٣

٢-١٤) (أ) 13.3333 (f.p.) (ب) 2 (integer)
(ج) 5 (integer) (د) 13.75(f.P.)
(هـ) -4(integer) (و) 1 (integer)
(ز) غير سليم لأن 10.0/3.0 تعبير f.p.، بينما المؤثر/ يتطلب أن يكون كلا معامَليه (operands) صحيحين (integer).

٢-١٥) (أ) ٣ (ب) ٤ (ج) ٣٧
(د) ٢٢ (هـ) ٢٣ (و) ٥

٢-١٦) (أ) a = 5b = 2
(ب) Sum : 7
(ج) Sum : 7
(د) 2 feet

٢-١٧) cost is
300
Price is 30Cost is 300
Grade A costs
300

(٢–١٨) ¶ ⊥ EMBED Equation.3 □

(٢–١٩)
(أ) .235 (ب) 2.25 (ج) 44.2
(د) 5 (هـ) 0 (و) 21
(ز) 8 (ح) 5 (ط) 1
(ي) 3

(٢–٢٠)
(أ) iostream (ب) cstdlib
(ج) cmath (د) iostream
(هـ) iostream, iomanip

(٢–٢١)
(أ) 9.1 (ب) 7.0 (ج) 24
(د) 16.0 (هـ) 5.0 (و) 3.0

(٢–٢٢)
AB
□□413□□is the value of n
□□□21.8□□is the value of y

(٢–٢٣)
(أ) 14.38 (ب) □□□14.38
(ج) 14.38 (د) □14.383

(٢–٢٤)
(أ) 26 (ب) reparation (ج) 0
(د) 15 (هـ) 15 (و) string: : npos

(٢–٢٥) sum = n * (n+1) / 2;

(٢–٢٦)
(أ) x / y - 3

(ب) (x + y) * (x - y)

(ج) 1 / (x + y)

(د) 1 / x + y

(ه) i / float (j)

(و) i / j

(ز) ((x + y) / 3 - (x - y) / 3) / (4 * x)

(٢٧–٢) (أ) abs(i) (ب) labs (n)

(ج) fabs (x + y) (د) fabs (x) + fabs (y)

(هـ) pow (x , 3) /y

(و) sqrt (pow (x,6) + pow (y,5)

(ز) pow (x + sqrt (y) , 7)

(٢٨–٢)
```
discriminant = sqrt (b * b - 4.0 * a * c);
denominator = 2.0 * a;
  solution 1 = (-b + discriminant) / denominator;
  solution 2 = (-b - discriminant) / denominator;
```

(٢٩–٢) Payment : 22.4 Charge : 6.0 Balance owed : 283.6

(٣٠–٢)
```
// This program compute the cost per ounce, given total cost
// and the weight expressed in pounds and ounces
// ***************************************
# include <iostraem>

using namespace std ;
const int T_COST      = 1376;        // Total cost
const int POUNDS = 10;               // Weight in pounds
const int OUNCES = 12;               // Additional weight in ounces

int main ( )
{       int totOz;                   // Total weight in ounces
```

```
float uCost;              // Computed cost per ounce
totOz = 16 * POUNDS;
totOz = totOz + OUNCES
uCost = T_COST / totOz;
cout << "Cost per unit: " << uCost << endl;
return 0;         }
```

مخرجات البرنامج :

Cost per unit: 8.0

٢-٣١)

```
// Calculate perimeter
   perimeter = 2.0 * (length + width) ;
   // Calculate are
  area = length *  width ;
  // Output results
  cout << "Rectangle length = " << length
          << " and width = " << width << endl ;
   cout << "Rectangle perimeter is " << perimeter << endl ;
   cout << "Rectangle area is " << area << endl ;
    return 0;
}
```

٢ - ٣٢ -أ) التعبير هو :

sentence. find ("ate")

ونتيجتـه - أي موضـع أول حـدوث للسلسـلة "ate" **- بالنسـبة للجملـة المعطاة:** 12

(أي أن الحرف a **في السلسلة** "ate" **هو الرمز الثالث عشر في الجملة، لأن موضع الرمز الأول في الجملة هو** 0 **وليس** 1**)**

ب)

```
pos1 = sentence.find ("at");
strleft = sentence.substr (pos1 + 2 , sentence.length ( ) - 2) ;
pos2 = strleft.find ("at");
cout << "second position of 'at' : " << pos2 + pos1 + 2
strleft = strleft.substr (pos2 + 2 , strleft.length ( ) - 2) ;
pos3 = strleft.find ("at");
```

cout << "third position of 'at' : " << pos3 + pos2 + pos1 + 4;

وبالنسبة للجملة المعطاة نحصل على النتائج التالية:

pos1 = 12 **(موضع "at" في الجملة المعطاة في أول حدوث)**

pos2 = 22 **(موضع "at" في السلسلة الجزئية التي تلي أول حدوث)**

pos3 = 14 **(موضع "at" في السلسلة الجزئية التي تلي ثاني حدوث)**

second position of "at" : 36 **(موضع ثاني حدوث في الجملة المعطاة)**

third position of "at" : 52 **(موضع ثالث حدوث في الجملة المعطاة)**

٢-٣٣) int1: 17, int2: 13, int3: 7

٢-٣٤) a: 14, b: 19; c: 67, d: 73

٢-٣٥) 123 147

٢-٣٦ – أ) i: 11; x: 12.35, ch1 : 'A'

ب) i: 1; x: 12.35, ch1 : '1'

٢- ٣٧ -أ) i) **المتغير** address **سيحتوي على سلسلة الرموز** "40" **(فقط ، لأن المؤثر** << **سيتوقف عند أول رمز فراغي أبيض)**

ii) **مؤشر القراءة** (reader marker) **سيقف مشيراً إلى** (pointing to) **رمز الفراغ** (blank character) **الموجود قبل حرف** 'B' .

ب) i) **المتغير** address **سيحتوي على سلسلة الرموز** "40 Badr Street"

(لأن الدالة getline **تقرأ وتخزن سطر إدخال كاملاً بما في ذلك الفراغات البينية)**

ii) **مؤشر القراءة سيقف مشيراً إلى حرف** 'M' .

٢ – ٣٨ -أ) **أخطاء البرنامج هي:**

- **ناقص الإعلان عن** outData:

```
ofstream        outData:
```

- **ناقص فتح ملف الإدخال:**

```
InData.open ( " myfile.dat ") ;
```

- **العبارة**

```
cin >> n ;
```

لا تقرأ من ملف الإدخال . فيجب تغيير cin **إلى** inData

ب) i) **سيل الملف** inData **سيظل يحتوي على القيمة** 144 **بعد تنفيذ البرنامج.**

ii) **سيل الملف** outData **سيحتوي على القيمة** 144 **يتبعها رمز السطر الجديد** (newline character).

٢ – ٣٩ –أ)

```
cin >> ch1 >> ch2 >> ch3 ;
```

ب)

```
cin.get (ch1) ;          أو          cin >> ch1 ;
cin.get (ch2) ;
cin.get (ch3) ;
```

```
cin >> x1 >> x2 >> x3 >> x4 ;
```
٢–٤٠)

٢–٤١) في الحل التالي القيمة ١٠٠ اختيارية ، وأي قيمة أخرى أكبر من 4 **تصلح أيضاً .**

```
cin.get (chr1) ;
cin.ignore (100,   '\n') ;
cin.get (chr2) ;
cin.ignore (100,   '\n') ;
cin.get (chr3) ;
cin.ignore (100,   '\n') ;
```

٢–٤٢) نفرض أن لدينا الإعلانات

```
char   ch1,   ch2,   ch3,   ch4,   ch5,   ch6,   ch7;
int    n1,    n2,    n3,    n4,    n5,    n6;
float  x1,    x2,    x3;
```

هناك حلول عديدة ممكنة، وفيما يلي أحد هذه الحلول:

```
cin >> ch1 >> n1 >> x1 >> ch2 >> n2 ;
cin >> x2 >> ch3 >> ch4 >> x3 >> n3;
cin >> ch5 >> n4 >> ch6 >> n5 >> ch7 >> n6;
```

وفيما يلي حل آخر يستخدم عبارة إدخال واحدة فقط:

```
cin >> ch1 >> n1 >> x1 >> ch2 >> n2 ;
    >> x2 >> ch3 >> ch4 >> x3 >> n3;
    >> ch5 >> n4 >> ch6 >> n5 >> ch7 >> n6;
```

٢-٤٣) يمكننا استخدام تسعة متغيرات ، وفي هذه الحالة نحتاج فقط لعبارة إدخال واحدة ، وعبارة إخراج واحدة كما يلي:

```
inFile >> int1 >> int2 >> int3 >> int4 >> int5
       >> int6 >> int7 >> int8 >> int9;
cout   << int1 << '  ' << int2 << '  ' << int3 << endl
       << int4 << '  ' << int5 << '  ' << int6 << endl
       << int7 << '  ' << int8 << '  ' << int9 << endl;
```

وكحل آخر يمكننا كتابة كل عدد صحيح بمجرد قراءته، وفي هذه الحالة نحتاج لمتغير واحد فقط (بدلاً من تسعة متغيرات) ، ولكننا عندئذ نحتاج لتسع عبارات إدخال مستقلة، وتسع عبارات إخراج مستقلة.

٢ - ٤٤-أ) قطعة برنامج تبادلي لمستخدم مبتدئ:

```
cout << "Enter your age:  " ;
cin >> age;
cout << "Enter your height:  " ;
cin >> height ;
cout << "Enter your weight:  " ;
cin >> weight ;
cout << "Enter the initial of your first name:  " ;
cin >> first ;
cout << "Enter the initial of your last name:  " ;
cin >> last ;
```

ب) قطعة برنامج تبادلي لمستخدم ذي خبرة:

```
cout << "Enter age, height, and weight:  " ;
cin >> age >> height >> weight;
cout << "Enter initials:  " ;
cin >> first >> last;
```

(٢ –٤٥ –أ

8.
```
# include <iostream>
# include <fstream>

using namespace std;

int main ( )
{
        int             val1;
        int             val2;
        int             val3;
        int             val4;
        ifstream dataIn;
        ofstream resultsOut;

        dataIn.open ("myinput.dat") ;
        resultsOut.open ("myoutput.dat") ;

        dataIn >> val1 >> val2 >> val3 >> val4;
        resultsOut << val1 << val2 << val3 << val4 << endl;
        return 0;
}
```

ب) أضف العبارات التالية للبرنامج:

```
String InFileName;
istream dataIn;
.
.
cout << "Enter file name:  " ;
cin << InFileName;
.
.
dataIn.open (inFileName);
.
.
```

(٢–٤٦

Main Module

Open files

Read 3 coefficients from input file

```
        Calculate 2 floating-point solutions
        Write solutions to output file

    Open files
        Open file inQuad for input
        Open file outQuad for output
  Read 3 coefficients from input file
        Read a from inQuad
        Read b from inQuad
        Read c from inQuad

Calculate 2 floating-point solutions
        Set solution1 = (-b + sqrt (b * b - 4.0 * a * c)) / (2.0 * a)
        Set solution2 = (-b - sqrt (b * b - 4.0 * a * c)) / (2.0 * a)

Write solutions to output file
        Write solution1 to outQuad
        Write '  ' to outQuad
        Write solution2 to outQuad
```

(٤٧–٢

```
//************************************************
// Canvas program
// This program computes the dimensions and costs of materials
// to build a painting canvas of given dimensions. The user is
// asked to enter the length and width of the painting and the
// costs of the wood (per inch) and canvas (per square foot)
//************************************************
# include <iostream>
# include <iomanip>          // For setprecision ( )

using namespace std;

const float SQ_IN_PER_SQ_FT = 144.0;     // Square inches per
istream InFile;                          // square foot
string InFileName;

int main ( )
{
        float length;          // Length of painting in inches
        float width;           // Width of painting in inches
```

```
float woodCost;          // Cost of wood per inch in dollars
.
.
cout << "Enter name of file to read:  ";
cin >> InFileName;
InFile.open (InFileName) ;

cout << fixed << showpoint;          // Set up floating-pt.
                                     // output format
// Get length and width
InFile >> length >> width;

// Get wood cost
InFile >> woodCost;

// Get canvas cost
InFile >> canvasCost;
```

٢-٤٨) T (أ) T (ب) T (ج)

T (د) T (هـ) F (و)

F (ز) T (ح) T (ط)

T (ي) T (ك) T (ل)

(م) F . **الـدوال الأعضـاء في طبقـة** (class member functions) **يـتم استدعاؤها** (invoked) **باستخدام اصطلاح النقطة** (dot notation) .

أجوبة تمرينات رقم ٣

٣-١) letter >= 'A' && letter <= 'Z'

٣-٢) (أ) T (ب) T (ج) T (د) F

٣-٣) (أ) لا نحتاج لأي أقواس (ب) لا نحتـــاج لأي أقواس
(ج) لا نحتاج لأي أقواس (د) ! (q && q)

٣-٤) Charity extinguishes sin as water extinguishes fire

٣-٥) (أ) 10
10
(ب) The value of x is 3
(ج) 3
7
6
z

لاحظ أن أخر عبارة طباعة ليست جزءً من عبارة else- else) (clause **رغم تحويلها لليمين، وليست جزءاً من عبارة** if **المعطاة.**

٣-٦) (أ) Eligible to serve
(ب) Too short and too light to serve

٣-٧) (أ) 4 (ب) 2 (ج) 5
(د) 3 (ه) 1

٣-٨) (أ) x < y && y <= z

(ب) x > 0 && y > 0 && z > 0

(ج) x != y && x != z

(د) x = = y && x = = z

٣-٩) (أ) T (ب) F

(ج) F

(د) T (ه) T

٣-١٠) تظهر رسالة الخطأ الفاصلة المنقوطة بعد القوس الأيمن " } " لقالب (العبارة المركبة) (block / compound statement).

٣-١١) نحل مشكلة else المتدلية / التابعة بإدخال القوسين { } في القطعة كما هو موضح فيما يلي:

٣-١٢)

	ch1	ch2	ch3	Expected Output	
Set 1	'A'	'A'	'A'	All initials are the same	(أ)
Set 2	'A'	'A'	'B'	First two are the same	(ب)
Set 3	'B'	'A'	'A'	Last two are the same	(ج)
Set 4	'A'	'B'	'A'	First and Last are the same	(د)
Set 5	'A'	'A'	'C'	All initials are different	(ه)

٣-١٣) نعم

٣-١٤) عند التقييم من اليسار لليمين يتم تقييم أول تعبيرين علاقيين فقط، وذلك لأن التعبير الأول يعطي true فيستمر التقييم ، ثم

نعطي الثاني fals فيتوقف التقييم ، ويعطي الحاسب النتيجة النهائية false ، حيث لا داعي لتقييم التعبير الثالث لأن النتيجة لن تؤثر على هذه النتيجة النهائية.

٣-١٥) بعد استدعاء الدالة open أدخل القطعة التالية:

```
if ( ! info )
{
        cout << "Cannot open the input file." << endl;
        return 1 ;
}
```

٣-١٦) أ) المخرجات 120 50 170

لاحظ أن الشرط الثاني في عبارة if هو تعبير الإسناد (int2 = 50)

وليس التعبير العلاقي (int2 = = 50)

ب) المخرجات:

```
Charity is a proof. Patience is
                              illumination
```

ج) لا تظهر أي مخرجات

٣-١٧) (أ) 27 (ب) 3

٣-١٨) صحيحة (true)

أجوبة تمرينات رقم (٤)

٤-١) **أ) تطبع القطعة الأعداد الصحيحة من 2 إلى 11 ، كل عدد على سطر مستقل.**

(ب)

```
number = 1;
while (number < 11)
{
        cout << number << endl;
        number ++ ;
}
```

٤-٢) **٦ تكريرات**

٤-٣)

4	1	2
3	1	2
2	1	2
1	1	2

٤-٤) **أ)**

4 6 8 10 12 14 16 18 20 ..

تستمر القطعة في طباعة الأعداد الزوجية ابتداءً من 4 دون توقف.

ب) الخطأ الأول في القطعة أنها تبدأ بطباعة العدد 4 بدلاً من 2 .ويمكن تصحيح هذا الخطأ إما بجعل القيمة الابتدائية قبل العروة n = 0 ، أو بطباعة قيمة n الابتدائية أولاً قبل زيادتها إلى4 عن طريق تبديل عبارة الطباعة وعبارة زيادة n.

والخطأ الثاني هو أن العروة لا تتوقف أبداً وتظل لانهائية لأن شرط العروة (n لا تساوي 15) يظل صحيحاً true دائما لأن n تحتوي دائماً على عدد زوجي ، ويمكن تصحيح الشرط ليصبح مثلاً n < 15 .

وبالتالي يمكن تصحيح القطعة لتصبح كما يلي :

```
n = 2;
while (n < 15)
{
     cout << n <<  '   ' ;
     n = n + 2;
}
```

٤-٥) أ) مخرجات القطعة هي BCDE يتبعها – يمينها– صدى لرمز السطر الجديد (echo of the newline character). وأما رمز 'A' الذي يُدخَل بالقراءة الأولية (priming read) قبل العروة فإنه يُهمل و لا يطبع لأن العروة تبدأ بقراءة رمز جديد قبل طباعة الرمز الحالي ، أي أن عبارتي العروة معكوستان.

ب) تصحيح القطعة:

```
cin.get (inChar);
while (inChar != '\n')
{
        cout << in Char;
        cin.get (inChar);
}
```

٤-٦) تحتاج القطعة إلى قراءة أولية (واحدة فقط) قبل عروة while الخارجية ، هكذا :

```
cin.get (letter);
while (cin)
   .
   .
```

وعبارة الإدخال هذه التي أضفناها تصلح كقراءة أولية لكل من العروة الخارجية والعروة الداخلية.

٤-٧) 1 3 6 10 15

٤-٨) بعد الخروج من العروة : sum = 18
number = 0

٤-٩) أ) 15

ب) 7

ج)

sum	innerCount	outerCount
0	1	1
1	2	2
2	1	3
4	2	4
5	3	
7	1	
10	2	
	3	
	4	

المخرجات : 10

٤-١٠) 98

٤-١١) **العروة لا نهائية**

٤-١٢)

```
dangerous = false;
while (  !dangerous)
{
        cin >> pressure;
        if (pressure > 510.0)
                dangerous = true
}
```

أو بصورة أخرى

```
dangerous = false;
while (  !dangerous)
{
        cin >> pressure;
        dangerous =  (pressure > 510.0);
}
```

٤-١٣)

```
count28 = 0;
loopCount = 1;
while (loopCount <= 100)
{
        inputFile >> number;
        if  (number = = 28)
                count28 ++ ;
        loopCount ++ ;
}
```

٤-١٤)

```
line = 1;
while (line <= 4)
{
    number = 1;
    while (number <= line)
    {
        cout << number << ' ';
        number ++;
    }
    cout << endl;
    line ++;
}
```

٤-١٥) نفرض في الحل التالي أن :

sum , count, score : **متغيرات من النوع** int،

average : **متغير من النوع** float.

```
  count = 0;
  sum = 0;
  scoreFile >> score;
  while (scoreFile)
  {
      sum = sum + score;
      count++;
      scoreFile >> score;
  }
  if (count > 0)
      average     =     float(sum)     /
float(count);
```

(١٦-٤

```
positives = 0;
negatives = 0;
cin >> number;
while (cin)                // while NOT EOF …
{
        if (number > 0)
                positives ++;
        else if (number < 0)
                negatives ++;
        cin>> number;
}
cout << " Number of positive numbers: " << positives << endl;
cout << " Number of negative numbers: " << negatives << endl;
```

(١٧-٤

```
sum = 0;
evenInt = 16;
while (evenInt <= 26)
{
        sum = sum + evenInt;
        evenInt = evenInt + 2;
}
```

(١٨-٤

```
hour = 1;
minute = 0;
am = true;
done = false;
while ( !done )
{
    cout << hour << ':';
    if (minute < 10)
        cout << '0';
     cout << minute;
   if (am)
      cout << " A.M. " << endl;
   else
      cout << " P.M. " << endl;
```

```
     minute++;
     if (minute > 59)
     {
          minute = 0;
           hour++;
           if (hour == 13)
               hour = 1;
           else if (hour == 12)
               am = !am;
      }
      if (hour == 1 && minute == 0 && am)
           done = true;
  }
```

(٤–١٩

```
hour = 1;
tenMinute = 0;
am = true;
done = false;
while ( !done )
{
   cout << setw(2) << hour << ':' <<
tenMinute << '0';
   if (am)
           cout << " A.M. ";
   else
           cout << " P.M. ";

      tenMinute++;
    if   (tenMinute > 5)
    {
           cout << endl;
           tenMinute = 0;
           hour++;
           if (hour == 13)
               hour = 1;
           else if (hour == 12)
               am = !am;
     }
```

```
     if (hour == 1 && tenMinute == 0 &&
am)
          done = true;
     }
```

(٤–٢٠

```
int NumStrings = 0;
int NumStringsWithE = 0;

string InputLine;
string CurrentString;

cout << "Enter line (terminated with 'End'): " << endl;
cin >> InputLine;

while (CurrentString != "End")
{
      NumStrings++;
      if (CurrentString.find("e") != string:npos)
          NumStringsWithE++;

      cin >> InputLine;

}

output << NumStrings << " strings read" << endl;
output << NumStringsWithE << " strings contained an 'e' "
        << endl;
output << setw(2) << setprecision(0)
       << (NumStringsWithE/float(NumStrings)))*100
       << " %" << " contained an 'e' " << endl;
```

(٤–٢١

```
int NumStrings = 0;
int NumStringsWithE = 0;

int TotalNumStrings = 0;
int TotalNumStringWithE = 0;

string InputLine;
```

```
string CurrentString;
 string InputFileName;

istream InputFile;

cout << "Enter file name: ";
cin >> InputFileName;

InputFile.open(InputFileName.c_str());

InputFile >> CurrentString;

while (InputFile)
{
    while (CurrentString != "End”)
    {
        NumStrings++;
        if (CurrentString.find("e”) != string:npos)
            NumStringsWithE++;

        InputFile >> CurrentString;
    }
    output << NumStrings << " strings read" << endl;
    output << NumStringsWithE << " strings contained an
        'e'" << endl;
    output << setw(2) << setprecision(0)
           << (NumStringsWithE/float(NumStrings)))*100
           << " %" << " contained an 'e'" << endl;
    output << endl;  // spacer

    NumString = 0;            // reset
    NumStringWithE = 0;  // reset
}
```

(٤–٢٢

```
int NumStrings = 0;
int NumStringsWithE = 0;

int TotalNumStrings = 0;
int TotalNumStringWithE = 0;
```

```
string InputLine;
string CurrentString;
string InputFileName;

istream InputFile;

cout << "Enter file name: ";
cin >> InputFileName;

InputFile.open(InputFileName.c_str());

While (InputFile)
{
    while (CurrentString != "End”)
    {
        NumStrings++;
        if (CurrentString.find("e”) != string:npos)
            NumStringsWithE++;
    }

    output << NumStrings << " strings read" << endl;
    output << NumStringsWithE << " strings contained an
        'e'" << endl;
    output << setw(2) << setprecision(0)
           << (NumStringsWithE/float(NumStrings)))*100
           << " %" << " contained an 'e'" << endl;
    output << endl;  // spacer

    TotalNumStrings = TotalNumStrings + NumStrings;
    TotalNumStringWithE = TotalNumStringsWithE +
                                  NumStringWithE;

    NumStrings = 0;          // reset
    NumstringWithE = 0;   //   reset
}

    output  <<  TotalNumStrings << " strings read"
            <<    endl;
    output  <<  TotalNumStringsWithE << "strings contained an 'e'"
            << endl;
    output << setw(2) << setprecision(0)
           << totalNumStringsWithE/float (TotalNumStrings))) *100
```

```
<< " %" << " contained an 'e'" << endl;
```

٤- ٢٣) أ ، ب، ج) في عروة while **نختبر إن كانت كمية الدخل سالبة** (negative income amount) **وذلك قبل تشغيلها:**

```
int main()
{     .
      .
    incFile.open(fileName.c_str());
    if ( !incFile )
    {
        cout << "** Can't open input file **" << endl;
         return 1;
    }

    incFile >> sex >> amount;
    femaleCount = 0;
    femaleSum = 0.0;
    maleCount = 0;
    maleSum = 0.0;

    while (incFile)
    {
        cout << "Sex: " << sex << " Amount: " << amount
             <<  endl;

        if (amount < 0.0)              // Check for invalid salary
          cout << "** Bad data--negative salary **" << endl;
        else
            if (sex = = 'F')
            {
               femaleCount++;
               femaleSum = femaleSum + amount;
            }
            else if (sex == 'M')
            {
               maleCount++;
               maleSum = maleSum + amount;
            }
            else                       // Reject invalid sex code
```

```
                cout << "** Bad data--invalid sex code **" <<
                endl;

        incFile >> sex >> amount;
    }
    if (femaleCount <= 0)
       cout << "No females" << endl;  // Avoid division by zero
    else
    {
        femaleAverage = femaleSum / float(femaleCount);
        cout << "For " << femaleCount << " females, the
        average "
             << "income is " << femaleAverage << endl;
     }
        if (maleCount <= 0)
          cout << "No males" << endl; //Avoid division by zero
        else
        {
           maleAverage = maleSum / float(maleCount);
           cout << "For " << maleCount << " males, the
           average "
                << "income is " << maleAverage << endl;
         }
        return 0;
    }
```

٤-٢٤) لحساب أعلى دخل وأقل دخل لكل من الذكور والإناث نقوم بعمل التغييرات التالية:

(i) **أضف الإعلانات التالية:**

```
int  femaleHighest,      // highest income for females
     femaleLowest,       // lowest income for females
     maleHighest,        // highest income for males
     makeLowest;         // lowest income for males
```

(ii) **أضف القيم الابتدائية التالية:**

```
femaleHighest = 0;
femaleLowest = 0;
```

```
maleHighest = 0;
maleLowest = 0;
```

(iii) ضع العروة التالية بدلاً من العروة الموجودة بالبرنامج:

```
while (incFile)
{
    incFile << sex << amount;
    if (sex == 'F')
    {
       femaleCount++;
       femaleSum = femaleSum + amount;
       if (amount > femaleHighest)
          femaleHighest = amount;
       if (amount < femaleLowest)
          femaleLowest = amount;
 }
 else
 {
         maleCount++;
         maleSum = maleSum + amount;
         if (amount > maleHighest)
            maleHighest = amount;
         if (amount < maleLowest)
            maleLowest = amount;
   }

}
```

(iv) ضع عبارات الطباعة التالية بدلاً من عبارات الطباعة الموجودة بنهاية البرنامج:

```
cout << "For " << femaleCount
     << " females, the average income is " << femaleAverage
     << " the highest income is " << femaleHighest
     << " and the lowest income is " << femaleLowest << endl;
cout << "For " << maleCount
     << " males, the average income is " << maleAverage
     << " the highest income is " << maleHighest
     << " and the lowest income is " << maleLowest << endl;
```

أجوبة تمرينات رقم ٥

٥-١)

```
 5  3 13
 3  3  9
 9 12 30
```

٥-٢) *) عند التمرير بالقيمة (passing by value): جميع الأجزاء (g ← a) صحيحة (valid).

*) عند التمرير بالإسناد (passing by reference) : (a , c) فقط صحيحان (valid).

٥-٣-أ) 13571 [عنوان الذاكرة (memory address) الخاصة بالمتغير widgets (لأن الوسيط الشكلي وسيط إسناد)]

ب) نظراً لأن الوسيط الشكلي وسيط إسناد فلذلك كل من widgets, clunkers يشير إلى موقع الذاكرة نفسه (same memory location). وبالتالي فبعد تنفيذ عبارة الإسناد تكون قيمة كل من widgets, clunkersهي 77.

٥-٤) the answers are 12 10 3

٥-٥)

(i) a = 10 b = 7

(ii) x = 10 y = 7 b = U

(هذه هي قيم المتغيرات في Change عند لحظة دخول الدالة ، وقبل تنفيذ أي عبارات).

(iii) a = 10 b = 17

لاحظ أن x في الدالة Change وسيط ذو قيمة ، ولذلك فإن قيمة الوسيط الفعلي a لا تتغير.

٥-٦)

```
In function Test, the variables equal 5 20
In the main function after the first call, the variables equal 5 14
In function Test, the variables equal 5 20
In the main function after the second call, the variables equal
15 5
```

٥-٧- أ)

```
 1   cout << "Exercise ";
 2   DoThis(number1, number2);
 5   cout << number1 << ' ' << number2 << endl;
 3   cin >> value3 >> value1;
4     value2 = value1 + 10;
```

ب) في الدالة main قبل تنفيذ عبارة return مباشرة تكون

number2 number2 = 25

و أما value3 فإنها غير موجودة / غير معرفة حيث أنها متغير محلي في الدالة DoThis

٥-٨) أ)

```
void PrintMax( /* in */ int number1,
          /* in */ int number2 )
```

ب)

```
void RocketSimulation (    /* in */          float thrust,
                           /* inout */ float weight,
                           /* in * /     int       timestep,
                           /* in * /     int       totalTime,
                           /* out */     float& velocity,
                           /* out */     bool&  outOfFuel  )
```

(٥–٩

```
void GetMeanOf( /* in */  int    howMany,
                /* out */ float& avg     )

// Precondition:
// howMany > 0
// Postcondition:
// avg == average of howMany values read from standard input
device

{
        int   loopCount;       // Loop control variable
        float sum;              // Sum of the input values
        float inputValue;      // One input value

        sum = 0.0;
        loopCount = 1;
        while (loopCount <= howMany)
        {
           cin >> inputValue;
           sum = sum + inputValue;
           loopCount++;
        }
        avg = sum / float(howMany);
}
```

(٥–١٠–أ

```
void Halve( /* inout */ float& firstNumber,
            /* inout */ float& secondNumber )
{
        firstNumber = firstNumber / 2;
        secondNumber = secondNumber / 2;
}
```

(ب

```
void Halve( /* inout */     float&       firstNumber,
            /* inout */     float&       secondNumber )
// Precondition:
//    firstNumber  and  secondNumber are assigned
```

```
//Postcondition:
//    firstNumber = = firstNumber @ entry / 2
//    secondNumber = = secondNumber @ entry / 2

{
□
}
```

٥-١١-أ) نلاحظ في البرنامج Walk program **أن المعالج** setprecision **يظهر فقط في مخرجات المسافة الأولى ، ولكن لا بأس من استخدامه في مخرجات كل من المسافات الأربع.**

```
void ComputeMiles( /* in */    int    measurementNumber,
                   /* inout */ float& totalMiles       )

// Precondition:
//     1 <= measurementNumber <= 4
//  && totalMiles is assigned
// Postcondition:
//     Distance in miles for measurement number
measurementNumber
//       has been output
//  && totalMiles == totalMiles@entry + miles for
measurement
//                number measurementNumber

{
        float distance;    // Distance as measured
        float miles;       // Rounded distance in miles

        if (measurementNumber == 1)
           distance = DISTANCE1;
        else if (measurementNumber == 2)
           distance = DISTANCE2;
        else if (measurementNumber == 3)
           distance = DISTANCE3;
        else
           distance = DISTANCE4;

   miles = float(int(distance * SCALE * 10.0 + 0.5)) / 10.0;
```

```
cout << "For a measurement of " << setprecision(1) << distance;

            if (measurementNumber == 1)
               cout << " the first ";
            else if (measurementNumber == 2)
               cout << " the second ";
            else if (measurementNumber == 3)
               cout << " the third ";
            else
               cout << " the fourth ";

            cout << "distance is " << miles << " mile(s) long." <<
endl;
            totalMiles = totalMiles + miles;
        }
```

ب)

```
        ComputeMiles(1, totMiles);
            ComputeMiles(2, totMiles);
            ComputeMiles(3, totMiles);
            ComputeMiles(4, totMiles);
```

٥-١٢-أ)

```
        void ScanHeart (    /* out */     bool&  normal  )

        // Postcondition:
        //    normal = = true, if a normal heart rate (60-80) was input
        //                        before EOF occurred
                    = = false, otherwise
        {
            int  heartRate;

            cin >> heartRate;
            while ( ( heartRaate < 60 ۱۱ heartRate > 80) && cin)
                cin >> heartRate;

            // At loop exit, either (heartRate >= 60 && heartRate <=
        80)
            // or EOF occurred
```

```
    normal = (heartRate >= 60 && heartRate <= 80);
}
```

ب)

```
ScanHeart (normal);
```

٥-١٣-أ)

```
void Rotate(    /* inout */  int&  firstValue,
                /* inout */  int&  secondValue,
            /* inout */ int&  thirdValue,
// This function takes three parameters and returns their values
// in a shifted order


// Precondition:
    firstValue , secondValue and  thirdValue are assigned
// Postcondition:
//    firstValue == secondValue@entry
//  && secondValue == thirdValue@entry
//  && thirdValue == firstValue@entry

{
    int temp;      // Temporary holding variable

    // Save value of first parameter

    temp = firstValue;

    // Shift values of next two parameters

    firstValue = secondValue;
    secondValue = thirdValue;

    // Replace value of final parameter with saved value

    thirdValue = temp;
}
```

ب)

```
#include <iostream>

using namespace std;

void Rotate( int&, int&, int&,  );

int main()
{
        int int1;    // First input value
        int int2;    // Second input value
        int int3;    // Third input value

        cout << "Enter three values: ";
        cin >> int1 >> int2 >> int3 ;

        cout << "Before: " << int1 << '  ' << int2 << '   '
             << int3 << '  ' << endl;
        Rotate (int1, int2, int3);
        cout << "After: " << int1 << '  ' << int2 << '  '
              << int3 << '  ' << endl;
   return 0;
}
// The Rotate function, as above, goes here
```

ج)

```
#include <iostream>

using namespace std;

void Rotate( int&, int&, int&, int& );

int main()
{
        int int1;    // First input value
        int int2;    // Second input value
        int int3;    // Third input value
        int int4;    // Fourth input value

        cout << "Enter four values: ";
        cin >> int1 >> int2 >> int3 >> int4;
```

```
        cout << "Before: " << int1 << ' ' << int2 << ' '
             << int3 << ' ' << int4 << endl;
        Rotate(int1, int2, int3, int4);
        cout << "After: " << int1 << ' ' << int2 << ' '
             << int3 << ' ' << int4 << endl;
   return 0;
}

//*************************************************
*******

void Rotate(    /* inout */ int& firstValue,
                /* inout */ int& secondValue,
            /* inout */ int& thirdValue,
            /* inout */ int& fourthValue  )

// Precondition:
//    All parameters are assigned
// Postcondition:
//    firstValue == secondValue@entry
//  && secondValue == thirdValue@entry
//  && thirdValue == fourthValue@entry
//  && fourthValue == firstValue@entry

{
    int temp;    // Temporary holding variable

    // Save value of first parameter
    temp = firstValue;

    // Shift values of next three parameters

    firstValue = secondValue;
    secondValue = thirdValue;
    thirdValue = fourthValue;

    // Replace value of final parameter with saved value

    fourthValue = temp;
}
```

(١٤-٥

```
void CountUpper( /* out */ int& upCount )

// Precondition:
//      Reading marker for standard input stream is at beginning
of
//     a line terminated by '\n'
// Postcondition:
//     upCount == number of uppercase letters in the input line

{
        char ch;    // An input character

        upCount = 0;
        cin.get(ch);
        while (ch != '\n')
        {
            if (ch >= 'A' && ch <= 'Z')
                upCount++;
            cin.get(ch);
        }
}
```

(١٥-٥

```
void AddTime(      /* inout */ int& hours,
                   /* inout */ int& minutes,
                   /* in */    int  elapsedTime )

// Precondition:
//     hours >= 0  &&  0 <= minutes <= 59
//  && elapsedTime >= 0
// Postcondition:
//     hours and minutes represent the time obtained by adding
//          elapsedTime to the starting time (hours@entry and
minutes@entry)

{
    minutes = minutes + elapsedTime;
```

```
    hours = hours + minutes / 60;
    minutes = minutes % 60;
}
```

(٥–١٦

```
void GetNonBlank( /* out */ char& ch )

// Precondition:
//    At least one nonblank character exists in the
//    remaining input stream
// Postcondition:
//    Any blank input characters have been skipped
//  && ch == first nonblank character

{
       cin.get(ch);
       while (ch == ' ')
          cin.get(ch);
}
```

(٥–١٧–أ

```
void SkipToBlank( /* out */ char& ch )

// Precondition:
//    At least one blank exists in the remaining input stream
// Postcondition:
//    Any nonblank input characters have been skipped
//  && ch == first blank character
{
    cin.get(ch);
    while (ch != ' ')
        cin.get(ch);
}
```

(ب

```
void SkipToBlank (/* out */ char& ch,
                  /* out */ int&  skipped )

// Precondition:
//    At least one blank exists in the remaining input stream
```

```
// Postcondition:
//    Any nonblank input characters have been skipped
//  && ch == first blank character
//  && skipped == number of nonblank characters skipped

{
        skipped = 0;
        cin.get(ch);
        while (ch != ' ')
        {
              skipped++;
              cin.get(ch);
        }
}
```

(٥-١٨-أ

```
void PrintBarGraph( /* in */ float deptSales )

// Precondition:
//    0.0 <= deptSales <= 25000.0
// Postcondition:
//    A line of the bar chart has been printed with one * for
//    each $500 in sales, with fractions over $250 rounded up
//  && No stars have been printed for sales <= $250

{
        while (deptSales > 250.0)
        {
           cout << '*' ;                              // Print '*' for
each $500
           deptSales = deptSales - 500.0;     //     Update     loop
control
        }                                          //  variable
}
```

(ب

```
void PrintData (          /* in */ int   deptID,
                  /* in */ int   storeNum,
                  /* in */ float deptSales )
```

```
{
        string bar;                 // Bar of asterisks

        cout << setw (12) <<   "Dept  " << deptID << endl;
        cout << setw (3) << storeNum << "       " ;
        CreateBar (deptSales, bar);
        cout << bar << endl;
```

ج) نضيف العبارة التالية – في الدالة GetData – بعد عبارة الإدخال (input statement) التي تقرأ قيمة numDays:

```
if   (numDays < 0)
        cout << "** Data error -- number of days for dept.  " deptId
             << "  is negative **" << endl;
```

د)

```
//*************************************************
void PrintData( /* in */ int   deptID,            // Department ID
number
                /* in */ int   storeNum,          // Store number
                /* in */ float deptSales )        // Total sales for the
                                                  //   department
{
        bool printExclamation = false;       // initialize
        if (deptSales > 25000.00)
            printExclamation = true;

        cout << setw(12) << "Dept " << deptID << endl;
        cout << setw(3) << storeNum << "     ";
        while (deptSales > 250.0)
        {
            cout << '*' ;                    // Print '*' for each $500
            deptSales = deptSales - 500.0;  // Update loop control
        }                                    //   variable

        if (printExclamation)
            cout << "!" << endl;
        else
            cout << endl;
}
```

هـ)

```
//*************************************************
// This program check the two data files for Graph
// The errors that it looks for are:
//  1) Different department ids
//  2) Different number of departments
//*************************************************
#include <iostream>
#include <fstream>

using namespace std;

void main ()
{
      int deptID1,          // Department ID for Store 1
          deptID2,          // Department ID for Store 2
          numDays1,         // Number of Days for Store 1
          numDays2,         // Number of Days for Store 2
          day;              // Used to read in the sales per day
      bool dataError;       // file error flag

      float sales;              // Used to read in the sales per day
      string fileName;      // Holds user-entered file names
      ifstream store1,      // file for store 1
               store2;      // file for store 2

 // Get data file names from user
 cout << "Enter the name of the data file for the first store.";
 cin >> fileName;
 store1.open(fileName.c_str());
 cout << "Enter the name of the data file for the second store.";
 cin >> fileName;
 store2.open(fileName.c_str());

 dataError = false;  // Assume that data has no errors
 while (store1 && store2 && !dataError)
 {
       // Read in department ids and check for equivalence
       store1 >> deptID1 >> numdays1;
      store2 >> deptID2 >> numDays2;
      if (deptID1 != dept_ID2)
      {
            cout << "Error in data: Department IDs are different";
            dataError = true;
       }
       else    // Department IDs equivalent, continue
      {
           // Read in Sales for Store 1
           day = 1;
           while (day <= numDays1)
           {
            store1 >> sales;
```

```
            day++;
          }
          // Read in sales for Store 2
          day := 1;
          while (day <= numDays2)
          {
           store2 >> sales;
           day++;
          }

      }
   }  // end while
   if (!dataError && Store1 || Store2)

        cout << "Error in data: Files have an unequal number of"
             << " departments")
}
```

(٥-١٩

```
x = 3 after the call to DoReference.
x = 16 after the call to DoValue.
x = 17 after the call to DoLocal.
x = 7 after the call to DoGlobal.
```

(٥-٢٠

```
1 1
1 2
1 3
```

(٥-٢١ في داخل الدالة يتم تنقيص قيمة n إلى أن تصل إلى الصفر. ونظراً لأنه يتم تمرير n كوسيط إسناد وليس كوسيط ذي قيمة فعند العودة من هذه الدالة التي تعيد مجموع الأعداد الصحيحة من 1 إلى n ستصبح محتويات الوسيط الآخر – في الدالة المستدعية – صفراً !!

(٥-٢٢ نعم

(٥-٢٣

تعريف الدالة

```
float Power( float base,
             int   exponent )
{
   float answer = 1.0;
```

```
   int   i = 1;

   while (i <= exponent)
   {
      answer = answer * base;
      i++;
   }
   return answer;
}
```

Function call:

استدعاء الدالة

```
m = Power(k, l);
```

٥-٢٤) أ) -0.5
ب) 0.5

٥-٢٥) **أ) لا تستخدم عبارة** return **مع وجود تعبير بعد كلمة** return **في الدوال الخاوية ولكنها تستخدم في الدوال التي تعيد قيمة.**
ب) هذه الدالة تعيد قيمة ، ولذا ينقصها عبارة return **لتعيد قيمة الدالة.**

٥-٢٦) 5 25
3 25

٥-٢٧) 1 6 3

٥-٢٨) 25 55

٥-٢٩) 12

٥-٣٠) 38
٥-٣١) FALSE

٥-٣٢) 2 to the power 0 is 8

٥-٣٣) أ) T ب) T ج) F

د) T هـ) F

٥-٣٤) ملاحظة: عند إعادة كتابة البرنامج بدون متغيرات شاملة، لم يعد اسم الدالة MashGlobals (والذي يعني مزيجاً من المتغيرات الشاملة) يعكس ما يحدث ، وقد يكون من الأفضل إعادة تسميتها MashParameters.

```
#include <iostream>

using namespace std;

  void MashGlobals( int&, int&, int );

  int main()
  {
     int a;
     int b;
     int c;

     cin >> a >> b >> c;
     MashGlobals(a, b, c);
     cout << "a = " << a << ' ' << "b = " << b << ' '
          << "c = " << c << endl;
     return 0;
  }

  void MashGlobals( /* inout */ int& a,
                    /* inout */ int& b,
                    /* in */    int  c )
  {
     int temp = a + b;

     a = b + c;
     b = temp;
  }
```

٥-٣٥)

```
  float Epsilon( /* in */ float high,
                 /* in */ float low  )
```

(٣٦–٥

```
#include <cmath>   // For fabs()
    .
    .
    bool NearlyEqual( /* in */ float num1,
                      /* in */ float num2,
                      /* in */ float difference )
    {
       return (fabs(num1 - num2) < difference);
    }
```

or حل آخر

```
    {
       if (fabs(num1 - num2) < difference)
          return true;
       else
          return false;
    }
```

(٣٧–٥

```
flot CompassHeading (  / * in * / float trueCourse,
                       / * in * / float windCorrAngle,
                       / * in * / float variance,
                       / * in * / float deviation          )
// Precondition:
//        All parameters are assigned
// Postcondition:
//        Function value = = trueCourse + windCorrAngle +
//                                variance + deviation
{
        return trueCourse + windCorrAngle + variance + deviation;
}
```

(٣٨–٥

```
    float FracPart( /* in */ float x )

    // Precondition:
    //    x >= 0.0
    // Postcondition:
```

```
//    Function value == fractional part of x

{
  return x - float(int(x));
}
```

ملاحظة: افترضنا في الحل أن EMBED Equation.3 **. وفي حالة ما إذا كانت** x **عدداً سالباً في التعبير** int (x) **فإن اتجاه القطع** (trunction) **(نحو الصفر أو بعيداً عنه) يعتمد على الآلة المستخدمة. وإذا أردنا حذف هذا الافتراض لهذه الدالة ، فيمكننا أن نشمل** (include) **ملف المقدمة** cmath (header file) **ونستخدم التعبير** int(fabs (x)).

٥-٣٩)

```
float Circumf( /* in */ float radius )

// Precondition:
//    radius is assigned (in principle, radius >= 0.0)
// Postcondition:
//    Function value == 2.0 * pi * radius

{
  const float PI = 3.14159;

  return 2.0 * PI * radius;
}
```

٥-٤٠) بفرض أن البرنامج يشتمل على ملف المقدمة cmath **ليمكننا استخدام الدالة** sqrt**، فإن جسم الدالة** hypotenuse **هو:**

```
{
        return sqrt (side1*side1 + side2*side2);
}
```

٥-٤١) يمكن كتابة الدالة FifthPow **باستخدام عروة، ولكن أبسط من ذلك يمكننا كتابة تعبير بسيط مباشر يستخدم الضرب المتكرر كما يلي:**

```
float FifthPow( /* in */ float x )

// Precondition:
```

```
//    x is assigned
// Postcondition:
//    Function value == x to the power 5

{
   return x * x * x * x * x;
}
```

(٤٢-٥

```
int Min( /* in */ int int1,
         /* in */ int int2,
         /* in */ int int3 )

// Precondition:
//    int1, int2, and int3 are assigned
// Postcondition:
//    Function value == smallest of int1, int2, and int3

{
   int smallest = int1;

   if (int2 < smallest)
      smallest = int2;
   if (int3 < smallest)
      smallest = int3;
   return smallest;
}
```

حل آخر: *or*

```
{
   int smallest;

   if (int1 < int2)
      if (int1 < int3)
         smallest = int1;
      else
         smallest = int3;
   else
      if (int2 < int3)
         smallest = int2;
```

```
        else
            smallest = int3;
    return smallest;
}
```

(٤٣-٥

a.
```
if (isdigit(inChar))
    DoSomething();
```

b.
```
if (isalpha(inChar))
    DoSomething();
```

c.
```
if (isupper(inChar) || isdigit(inChar))
    DoSomething();
```

or

```
if (isalnum(inChar) && !islower(inChar))
    DoSomething();
```

d.
```
if ( !islower(inChar) )
    DoSomething();
```

(٤٤-٥

```
#include <cmath>   // For sqrt()
    .
    .
bool IsPrime( /* in */ int n )

// Precondition:
//    n is assigned
// Postcondition:
//    Function value == true, if n is a prime number
//                   == false, otherwise

{
  int trialDivisor;    // A trial divisor to test
  int limit;           // Greatest divisor to test

  if (n < 2)                  // If n < 2, it's not prime
    return false;
```

```
  limit = int(sqrt(float(n)));   // Only need to check up to sqrt(n)
   trialDivisor = 2;

   // Test divisors starting with 2. If we exceed limit, n is prime

   while (trialDivisor <= limit && n % trialDivisor != 0)
      trialDivisor++;
   return (trialDivisor > limit);
}
```

٤٥-٥)

```
float Postage ( /* in */ int      pounds,
                /* in */ int      ounces,
                /* in */ int      costPerOunces  )

// Precondition:
//        pounds >= 0 && ounces >= 0 && costPerOunce >= 0.0

// Postcondition
//        Function vlue = = ( pounds * 16 + ounces) * costPerOunce

{
        return ( pounds * 16 + ounces) * costPerOunce;
}
```

٤٦-٥) أ) أضفنا الدالة CheckDigits **التي تعيد القيمة** true **إذا كانت جميع رموز سلسلة الرموز** (character string) **التي تم تمريرها** (passed) **رموزاً رقمية** (digit characters)**. وما عدا ذلك فإنها تعرض رسالة خطأ وتعيد القيمة** false**. ويتم استدعاء الدالة** CheckDigits **بعد كل إدخال لسلسلة رموز رقمية (مثل** year**).**

```
//*******************************************************
// ConvertDates program
// This program reads dates in American form from an input file and
// writes them to an output file in American, British, and ISO form.
// No data validation is done on the input file
//*******************************************************
#include <iostream>
```

```
#include <iomanip>    // For setw()
#include <fstream>    // For file I/O
#include <string>     // For string type

using namespace std;

void Get2Digits( ifstream&, string& );
void GetYear( ifstream&, string& );
void OpenForInput( ifstream& );
void OpenForOutput( ofstream& );
bool CheckDigits ( string );
void Write( ofstream&, string, string, string );

int main()
{
   string   month;        // Both digits of month
   string   day;          // Both digits of day
   string   year;         // Four digits of year
   ifstream dataIn;       // Input file of dates
   ofstream dataOut;      // Output file of dates

   OpenForInput(dataIn);
   OpenForOutput(dataOut);
   if ( !dataIn || !dataOut )                // Make sure files
      return 1;                          //   were opened

   dataOut << setw(20) << "American Format"     // Write headings
           << setw(20) << "British Format"
           << setw(20) << "ISO Format" << endl << endl;

   Get2Digits(dataIn, month);                // Priming read

   if (!CheckDigits(month)) return 0;

   while (dataIn)                         // While not EOF...
   {
      Get2Digits(dataIn, day);
      if (!CheckDigits(day)) return 0;

      GetYear(dataIn, year);
      if (!CheckDigits(year)) return 0;
```

```
      Write(dataOut, month, day, year);
      Get2Digits(dataIn, month);
      if (!CheckDigits(month)) return 0;
    }
    return 0;
}

//*****************************************************

int CheckDigits( /* in */ digitChars)

// This function checks to see if the string passed contains non-digit
characters
//
// preconditions: none
//
// postconditions: An error message is output if the string contains non-digit
// characters, and false returned. Otherwise, a value of true is returned.
{
  for (int i = 0; digitChars.length(); i++)
    if ((digitChars.substr(0,1) < '0') || digitchars.substr(0,1) > '9')
  {
     cout << i << "th character is not a valid digit";
     return false;
  }

  return true;
}
```

ب) أضفنا الدالتين checkDate, LeapYear **للتحقق من صحة قيمتي اليوم** day **والشهر** month **. والدالة المنطقية المساعدة** LeapYear **تعيد القيمة** true **إذا كانت السنة كبيسة، والقيمة** false **إذا كانت السنة بسيطة، مع ملاحظة أن السنة تكون كبيسة إذا كانت تقبل القسمة على ٤ (بدون باق) بينما لا تقبل القسمة على ١٠٠ ، أو إذا كانت تقبل القسمة على ٤٠٠. ويتم استدعاء الدالة** CheckDate **من المواضع المناسبة في الدالة الرئيسية** main.

```
bool LeapYear ( /* in */ int year )

// Returns true if year passed is a leap year, otherwise,
// returns false.
```

```
{ if (((year % 4) == 0) && (year % 100) != 0)) ||
                (year % 400 == 0) )
return true
else
return false;
}

//*****************************************************

bool CheckDate ( /* in */ string days, /* in / string month,
                        /* in */ string year )

// This procedure checks to see that the month and year digit strings are
// valid, and that the number of days for the month/year passed is also valid.
// preconditions: days and month are digit character strings of length two.
// year is a string of length four.
// postconditions: An error message is printed if month is invalid, or days is
// invalid for month, and false returned. Otherwise, a value of true is returned.

{
  int intMonth, intDays, intYear;

  intMonth = (month.substr(0,1) – int('0'))*10 + (month.substr(1,0) –
  int('0'));
  intDays = (days.substr(0,1) – int('0'))*10 + (days.substr(1,0) –
int('0'));
  intYear = (year.substr(0,1) – int('0'))*1000 +
               (year.substr(1,0) – int('0'))*100 +
               (year.substr(2,0) – int('0'))*10 +
                    (year.substr(3,0) – int('0'));
if (intMonth < 1 || intMonth > 12)
{
    cout << "Month invalid!";
     return false;
}

// Check Days
   switch (intdays)
   {
    case 4: case 6: case 9:
    case 11: if (intDays == 30)
               return true;
             else
               return false;
    case 1: case 3: case 5: case 7: case 8: case 10:
    case 12: if (intDays == 31)
               return true;
             else
               return false;

    case 2: if ((LeapYear(intYear) && intDays == 29) ||
```

```
    (!LeapYear(intYear) && intDays == 28))
    return true;
  else;
    return false;

  }
```

٥-٤٧) نحتاج فقط لتعديل الدالة Get2Digits. **وذلك بتغيير شرط اختبار ما إذا كان الرمز الثاني شرطة مائلة** '/' (slash) **أم لا،**

من :

```
if (secondChar = = ' / ' )
```

إلى:

```
if (secondChar = = ' / '   | | secondChar = = ' - ' )
```

أجوبة تمرينات رقم ٦

٦-١) أ) F ب) T ج) F

٦-٢)

```
switch (n)
{
        case 3  : alpha ++;
                  break;
        case 7  : beta ++;
                  break;
        case 10: gamma ++;
                  break;          // Optional
}
```

٦-٣) MaryamKhadeejahAssiaBest Women !

٦-٤) أ) F (**يظل الشرط** (delta <= alpha)

ب) T

ج) F

٦-٥) أ) 1

ب) **لا يُطبع شئ ، حيث أن الشرط يكون** false **فوراً ونتخطى جسم العروة.**

٦-٦) أ)

```
4 3 2 1 4
3 2 1 3
2 1 2
1 1
```

(ب)

```
********* *********
********   ********
*******     *******
******       ******
*****         *****
****           ****
***             ***
**               **
*                 *
```

ويلاحظ أن هذه المخرجات تحتوي على سطر فارغ (blank line) مطبوع أسفل آخر سطر من النجوم.

(٦-٧)

(أ)

```
switch (grade)
{
        case 'A' :  sum = sum + 4;
                    break;
        case 'B' :  sum = sum + 3;
                    break;
        case 'C' :  sum = sum + 2;
                    break;
        case 'D' :  sum++;
                    break;
        case 'F' :  cout << "Student is on probation" << endl;
                    break;                  // Not required
}
```

(ب)

```
switch (grade)
{
        case 'A' :  sum = sum + 4;
                    break;
        case 'B' :  sum = sum + 3;
                    break;
        case 'C' :  sum = sum + 2;
                    break;
        case 'D' :  sum++;
                    break;
```

```
        case 'F' :  cout << “Student is on probation” << endl;
                    break;
        default  :  cout << “Invalid letter grade” << endl;
                    break;                  // Not required
}
```

(٦–٨

```
count = 0,
sum = 0;
do
{
        count++;
        cin >> dataValue;
        if (dataValue >= 0)
            sum = sum + dataValue;
} while (dataValue >= 0 && count  < 10);
```

(٦–٩

```
do
{
        cout << “Enter 1, 2, or 3: “;
        cin >> response;
} while (response < 1 | | response > 3);
```

(٦–١٠

```
cin >> ch;
while (cin)
{
        cout << ch;
        cin >> ch;
}
```

(٦–١١

```
sum = 0;
for (count = 1; count <= 1000; count ++)
        sum = sum + count;
```

(٦–١٢

```
m = 93;
while (m >= 5)
{
        cout << m << '  ' << m * m << endl;
        m--;
}
```

(١٣-٦

```
k = 9;
do
{
        cout << k << '  ' << 3 * k << endl;
        k++;
} while (k <= 21);
```

(١٤-٦

```
int Power (  /* in */  int base,
             /* in */  int exponent  )

// precondition:
//       base is assigned && exponent >= 0
// && (base to the exponent power) <= INT_MAX
// Postcondition:
??       Function value = = base to the exponent power

{
        int result = 1;          // Holds intermediate powers of base
        int count;               // loop control variable

        for (count = 1;  count <= exponent; count ++)
                result = result *  base;
        return result;
}
```

(١٥-٦

```
sum = 0;
count = 1;
do
{
      cin >> int1;
      if ( !cin || int1 <= 0)
      {
         cout << "Invalid first integer.";
         break;
      }
      cin >> int2;
      if ( !cin || int2 > int1)
```

```
    {
       cout << "Invalid second integer.";
       break;
    }
    cin >> int3;
    if ( !cin || int3 == 0)
    {
       cout << "Invalid third integer.";
       break;
    }
    sum = sum + (int1 + int2) / int3;
    count++;
  } while (count <= 100);
```

٦–١٦) أ) T ب) T ج) T

٦–١٧) أ) 19 ب) HELP ج) HLP

٦–١٨) ABC

٦–١٩) أ) 32 ب) 8

٦–٢٠) (هـ)

٦–٢١) أ)

```
void GetYesOrNo (  /* out */ char& response  ) // User response char
{
        cin >> response;
        while (response != 'y' && response != 'n')
        {
                cout << "Please type y or n: ";
                cin >> response;
        }
}
void GetOneAmount ( /* out */ float& response )   // Rainfall amount
                                                  // for one month
{
```

```
        cin >> amount;
        while (amount < 0.0)
        {
                cout << "Amount cannot be negative. Enter again: ";
                cin >> amount;
        }
}
```

ب)

```
count = 1;
do
{
      cout << "Enter rainfall amount " << count << ": ";
      GetOneAmount(amount);
      sum = sum + amount;

      count++;
}
while (count <= 12);
```

ج)

```
   count = 1;
   while (count <= 12)
   {
      cout << "Enter rainfall amount " << count << ": ";
      GetOneAmount(amount);
      sum = sum + amount;

      count++;
   }
```

د) نعم يمكن استخدام عبارة switch هنا (الصيغة التي تَستخدِم اختيار default) ولكن لا يُنصح بذلك حيث أنه أسلوب غير جيد للبرمجة.

هـ) لا يمكننا استخـــدام عبــارة switchهنا لأن عبارات switch لا يمكنها استخدام المؤثر العلاقي "<" مع الأنواع غير التكاملية (non-integral types)

(٦-٢٢

```
int Day( /* in */ int month,            // Month number, 1 - 12
         /* in */ int dayOfMonth,   // Day of month, 1 - 31
         /* in */ int year          )  // Year. For example, 2002
{   .

    .
    // Correct for different length months

    switch (month)
    {
        case 3 : correction = correction - 1;
                 break;
        case 2 :
        case 6 :
        case 7 : correction = correction + 1;
                 break;
        case 8 : correction = correction + 2;
                 break;
        case 9 :
        case 10 : correction = correction + 3; break;
        case 11 :
        case 12 : correction = correction + 4;
      }
      return (month - 1) * 30 + correction + dayOfMonth;
    }
```

أجوبة تمرينات رقم ٧

٧-١)
```
for (rainbow = RED; rainbow <= VIOLET;
rainbow = VisibleColors (ainbow +1))
```

٧-٢) 400500.000

٧-٣) أ) تعبير ب) تعبير ج) عبارة تعبير د) تعبير

٧-٤) أ)
```
sumOfSquares + = x * x ;
```
ب)
```
count --;
```
أو
```
-- count
```
ج)
```
k = (n > 8) ? 32 : 15 * n;
```

٧-٥) أ) defg ب) F 70

٧-٦)
```
Notice that
The character \ is a backslash
```

٧-٧) أ) صحيحة ب) خاطئة ج) خاطئة
د) صحيحة هـ) خاطئة و) صحيحة ز) صحيحة

٧-٨) أ) F ب) T
ج) F [قوسا الزاوية < > (angle brackets) يجب أن يكونا علامتي اقتباس " " (quotation marks)]
د) T هـ) T و) F ز) T ح) T

٧-٩) أ) إنزال مرتبة ب) ترقية

ج) إنزال مرتبة د) ترقية

٧-١٠)

```
# include <cctype>     // For toupper ( )
        .
        .
inFile.get (inputChar);
while (inFile)
{
        outFile << toupper (inputChar);
        inFile.get (inputChar);
}
```

٧-١١)

```
ch1 = '0' + n / 10;
ch2 = '0' + n % 10;
cout << ch1 << ch2;
```

لا حظ أن كلا من عبارتي الإسناد في هذا الحل تتضمن عمليتي تحويل ضمني، وذلك لأن '0' الموجود في الطرف الأيمن يُرقَّى أولاً إلى int قبل إجراء عملية الجمع، وبعدها تكون النتيجة من النوع int . ثم تؤدي عملية الإسناد إلى تحويل النتيجة من النوع int إلى النوع char.

الحل التالي الذي يستخدم التحويل الصريح للنوع يؤدي إلى عملية تحويل نوع واحدة فقط في كل من عبارتي الإسناد.

```
ch1 = '0' + char ( n / 10);
ch1 = '0' + char ( n % 10);
cout << ch1 << ch2;
```

٧-١٢)

```
 # iclude <float>      // For FLT_MAX
        .
        .
if (beta > FLT_MAX / 100.0)
        cout << "Value too large to multiply by 100" << endl;
else
        someFloat = beta * 100.0;
```

(٧–١٣) أ) enum CourseType {CS126, CS206, CS300, CS337, CS352, CS455}

ب) enum SacredMonths{ALMOHARRAM,RAJAB,THULQUEDA, THULHIJJA}

ج) enum HajjMonths{SHAWWAL, THULQUEDA, THULHIJJA}

د) enum DayType{MONDAY,TUESDAY,WEDNESDAY,THURSDAY, FRIDY};

(٧–١٤

```
void PrintDay ( /* in */ DayType weekDay )  // Day to be printed
// Precondition
//        weekDay is assigned
// Postcondition:
//        String corresponding to weekDay has been printed
{
        switch (weekDAy)
        {
                case MONDAY       : cout << "Monday";
                                    break;
                case TUESDAY      : cout << "Tuesday";
                                    break;
                case WEDNESDAY    : cout << "Wednesday";
                                    break;
                case THURSDAY     : cout << "Thursday";
                                    break;
                case FRIDAY       : cout << "Friday";
        }
}
```

(٧–١٥

```
for (today = MONDAY; today <= FRIDAY; today = DayType (today
+ 1))
{
        PrintDay (today) ;
        Cout << '  ';
}
```

١٦-٧) التعبير في عبارة return من النوع float بينما نوع الدالة المعلن هو int. ولذلك فإن قيمة التعبير float تُنْزَل مرتبتها إلى قيمة int معادة (int return value) بقطع (turncating) الجزء الكسري (fractional part) ، وقد يؤدي ذلك إلى نتيجة غير صحيحة. فلضمان الحصول على التقريب لأقرب عدد صحيح نغيِّر عبارة return إلى

```
return float (int1) / float (int2) + 0.5;
```

[مثلاً إذا كانت النسبة بين العددين هي 6.85 فإن عبارة return الأصلية في الدالة المعطاة ستؤدي إلى إعادة النتيجة 6 ، بينما عبارة return الجديدة ستؤدي إلى إعادة النتيجة 7].

والأفضل أن نستخدم تحويلاً صريحاً لزيادة الإيضاح:

```
return int (float (int1) / float (int2) + 0.5);
```

١٧-٧)

```
DayType CharToday  (  /* in */ char ch1,
                         / * in * / char ch2  )

// Precondition:
//       ch1 = = 'M' OR ch1 = = 'W' OR ch1 = = 'F' OR
       (ch1 = = 'T' && ch2 = = 'U' ) OR (ch1 = = 'T' && ch2 = = 'H')
// Postcondition:
//       Function value  = = MONDAY. If ch1 = = 'M'
                        = = TUESDAY, if (ch1 = = 'T' && ch2 = = 'U')
                        = = WEDNESDAY , if ch1 = = 'W'
                        = = THURSDAY, if (ch1 = = 'T' & & ch2 = = 'H')
                        = = FRIDAY, if ch1 = = 'F'
{
       switch ( ch1)
       {
              case 'M' : return MONDAAY;
              case 'T'  : if (ch2 = = 'U')
                                return TUESDAY
                          else
                                return THURSDAY;
              case 'W' : return WEDNESDAY;
              case 'F'   : return FRIDAY;
       }
}
```

أجوبة تمرينات رقم ٨

٨-١)

(أ)
```
void SomeFunc (  float x [ ]  )
```

(ب)
```
void SomeFunc (  const float x [ ]  )
```

٨-٢)

(أ)
```
firstName [0] = 'A';
```

(ب)
```
cout << firstName [13];
```

(ج)
```
for (index = 0;   index < SIZE;  index++)
```

(د)
```
for (index = 0;   index < n;  index++)
        cout << firstName [index];
```

٨-٣) (أ) T (ب) F (ج) T (د) F

(هـ) T (و) T (ز) F (ح-i) F

(ح-ii) T (ح-iii) F (ط) T (ى) T

(ك) F (ل) T (م) T (ن) T

(٤-٨

```
float  floatArray [24];
```
(أ)

```
int  intArray [500];
```
(ب)

```
double  doubleArray [50];
```
(ج)

```
char  cahrArray [10];
```
(د)

(٥-٨

```
const int CLASS_SIZE = 30;
```
(أ)

```
float  quizAvg [CLASS_SIZE]
```
(ب)

(٦-٨

```
enum BattleType {BADR,  OHOD,  Khaybar,  Tabook}
```
(أ)

```
int sightings [4];
```
(ب)

(٧-٨

```
for (cIndex = BLUE; cIndex <= BLACK; cIndex = Colors (cIndex +1))
     cout [cIndex] = 0;
```
(أ)

```
for (rIndex = 0; rIndex < MAX_LENGTH;  rIndex++)
     rinbow [rIndex] = WHITE;
```
(ب)

```
int counter = 0;
for (rIndex = 0; rIndex < MAX_LENGTH;  rIndex++)
    if (rinbow [rIndex] = = GREEN;
        counter++;
```
(ج)

```
cout << count [BLUE} << endl;
```
(د)

```
int sum = 0;
for  (cIndex = BLUE; cIndex <= BLACK; cIndex = Colors (cIndex +1))
    sum = sum + count [cIndex];
```
(هـ)

٨-٨)

```
1   3   -2
17  6   11
4   2   2
19  14  5
11  15  -4

52  40  12
```

٨-٩) عبارة for كُتبت خطأً بحيث أنها جعلت قيم المؤشر index تتغير في المدى من 1 إلى 4 بدلاً من المدى من 0 إلى 3. و الأمر بطباعة قيمة عنصر المنظومة arr [4] غير الموجود أصلاً أدى إلى الوصول إلى موضع الذاكرة الذي يلي مباشرة نهاية المنظومة. وواضح أن هذا الموضع كان يحتوي على القيمة 24835.

٨-١٠)

(أ)

sample	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
	10	9	8	7	6	5	4	3

(ب)

sample	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
	1	1	1	1	-1	-1	-1	-1

(ج)

sample	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
	0	101	2	103	4	105	6	107

٨-١١) (أ) 5 10 15

(ب) 500 400 300 200

(ج) 500 400 300 200 100

٨-١٢) (أ) 3 (ب) 9

٨-١٣) 0 2 4 6 8

٨-١٤) يمكن استخدام العروة (أ) أو العروة (ب)

٨-١٥) يمكننا استخدام (أ) أو (ج) أو (د)

٨-١٦)

```
void Initialize (  /* out */ bool failing [ ]  )

// Postcondition;
//    failing [0.. MAX_STUD -1] = = false

{
    int index;      // Loop control and index variable

    for (index = 0;  index < MAX_STUD;  index++)
        failing [index] = false;
}
```

٨-١٧)

```
 void SetFailing(    /* inout */     bool    failing[],
                     /* in */   const int     score[],
                     /* in */        int     length    )
// Precondition:
//    length <= MAX_STUD
//  && score[0..length-1] are assigned
// Postcondition:
//    For all i, where 0 <= i <= length-1,
//       IF score[i] < 60, THEN failing[i] == true

{
       int index;   // Loop control and index variable

       for (index = 0; index < length; index++)

                  // (optional)
                  // Invariant (prior to test):
```

```
                //     For all i, where 0 <= i <= index-1,
                //       IF score[i] < 60, THEN failing[i] == true
                //  && 0 <= index <= length

          if (score[index] < 60)
             failing[index] = true;
}
```

(١٨–٨

```
void SetPassing (  /* inout */        bool passing [],
                        /* in */   const int score []    )
// Precondition:
//    score [0.. MAX_STUD-1] are assigned
// Postcondition:
//    For all i, where 0 <= i <= MAX_STUD-1,
//       IF score[i] >= 60, THEN passing [i] = = true
{
        int index;   // Loop control and index variable

        for (index = 0; index < MAX_STUD; index++)
              if (score [index] >= 60)
                   passing [index] = true;
```

(١٩–٨

```
int PassTally( /* in */ const bool passing[],
                  /* in */       int     length    )
// Precondition:
//    length <= MAX_STUD
//  && passing[0..length-1] are assigned
// Postcondition:
//   Function value = = count of true values in passing[0..length-1]

{
        int index;       // Loop control and index variable
        int count = 0;   // Count of TRUE values

        for (index = 0; index < length; index++)

              // (optional)
            // Invariant (prior to test):
```

```
            //  count = = count of TRUE values in passing[0..index-1]
            //  && 0 <= index <= length

          if (passing[index])
            count++;
        return count;
    }
```

(٢٠–٨

```
    bool Error( /* in */ const bool passing[],
                /* in */ const bool failing[],
                /* in */       int     length    )

    // Precondition:
    //     length <= MAX_STUD
    //  && passing[0..length-1] are assigned
    //  && failing[0..length-1] are assigned
    // Postcondition:
  //    Function value = = true, if there is some i (0 <= i <= length-1)
 //                          such that passing[i] = = failing[i]
//                  = = false, otherwise

{
     int index = 0;    // Loop control and index variable

    while (index < length && passing[index] != failing[index])

            // (optional)
            // Invariant (prior to test):
            //     For all i, where 0 <= i <= index-1,
            //        passing[i] != failing[i]
            //  && 0 <= index <= length
        index++;
      return (index < length);
}
```

(٢١–٨

```
    int NumberGreater( /* in */       int grade,
                       /* in */ const int score[],
                       /* in */       int length  )
```

```
// Precondition:
//    length <= MAX_STUD
//  && score[0..length-1] are assigned  &&  grade is assigned
// Postcondition:
//    Function value == count of values in score[0..length-1]
//                      that are >= grade

{
        int index;       // Loop control and index variable
        int count = 0;   // Count of values that are >= grade

        for (index = 0; index < length; index++)
              // (optional)
              // Invariant (prior to test):
              //    count == count of values in score[0..index-1]
              //             that are >= grade
              //  && 0 <= index <= length

          if (score[index] >= grade)
             count++;
       return count;
}
```

(٢٢–٨

```
void Reverse( /* inout */ int score[],
                 /* in */    int length  )

// Precondition:
//    length <= MAX_STUD
//  && score[0..length-1] are assigned
// Postcondition:
//    score contains the same values as score@entry,
//    rearranged into reverse order

{
     int index;                    // Loop control and index variable
     int tempScore;                // Used for swapping
     int halfLength = length / 2;  // Upper limit for array index

    for (index = 0; index < halfLength; index++)
    {          // (optional)
```

```
        // Invariant (prior to test):
        //   For all i, where 0 <= i <= index-1,
        //     score[i] and score[length - (i + 1)] have
        //     been swapped
        //  && 0 <= index <= halfLength

        tempScore = score[index];
        score[index] = score[length - (index + 1)];
        score[length - (index + 1)] = tempScore;
    }
}
```

ملاحظة: إذا نفذنا عروة for للمدى من 0 إلى length (بدلاً من المدى من 0 إلى halflength فإن عناصر المنظومة score تعود إلى وضعها الأصلي.

٨-٢٣) (أ) لا يجوز . لأننا لو أعلنا عن المنظومة firstList أنها وسيط ثابت const parameter، فإن المترجم compiler سوف لا يسمح لجسم الدالة بتعديل المنظومة . وبالتالي فالعبارة:

```
firstList [counter] = number
```

سوف تؤدي إلى خطأ وقت الترجمة (compile-time error) .

(ب) كي يقارن البرنامج بين قائمتين من الأعداد من النوع float (بدلاً من النوع int) تبقي عبارات البرنامج كما هي، ولكن نحتاج فقط لتعديل الإعلان عن نوع firstList وبعض أنواع الوسطاء في الدالتين ReadFirstList, CompareLists ونوع القيمة التي تفصل بين القائمتين من النوع int إلى النوع float.

(ج) لا نحتاج لتغيير الدالة ReadFirstList. وأما الدالة compareLists فيتم تغييرها إلى ما يلي:

```
//*********************************************************
void CompareLists(
    /* in */   const int      firstList[],   // 1st list of numbers
```

```
        /* in */        int      numVals,      // Number in 1st list
        /* inout */     ifstream& dataFile,    // Input file
        /* inout */     bool&    allOK       ) // True if lists match

// Reads the second list of numbers
// and compares it to the first list

// Precondition:
//    numVals <= MAX_NUMBER && firstList[0..numVals-1] are assigned
// Postcondition:
//    Values from the second list have been read from the
//    input file
//  && If lists are identical,
//         allOK == true
//    ELSE
//         allOK == false
//      && Appropriate error message have been output

{
        int counter;    // Loop control and index variable
        int number;     // An input value
        bool listsEqualLength;

counter = 0;
        listsEqualLength = true;

while ((counter < numvalues) && listsEqualLength)
        {
```

```
dataFile >> number;
if (!dataFile)
{
        listsEqualLength = false;
        cout << “Second list shorter” << endl;
}
else
if (number != firstList[counter])
{
         allOK = false;
         cout << "Position " << counter << ": "
              << setw(4) << firstList[counter] << " != "
             << setw(4) << number << endl;
}
         }

         if (dataFile)
         {
listsEqualLength = false;
cout << “Second list is shorter.” << endl;
         }
         allOK = (allOK && listsEqualLength);
}
```

بعض المراجع عن البرمجة بلغة C+ +

References on Programming in C++

فيما يلي قائمة بأسماء بعض الكتب عن البرمجة بلغة C++ والقائمة مرتبة ترتيباً أبجدياً بالنسبة لأسماء المؤلفين

1. Astrachan, O.L.;
Computer Science Tapestry
Exploring Programming and Computer Science with C++
Second edition
McGraw Hill, 2000.

2. Dale, N. – Weems, C. – Headington, M.
Programming and Problem Solving with C++ (Text)
Second edition
Jones and Bartlett publishers, 2000.

3. Eitel, H.M. – Deitel, P.J.
C++ How to Program
Third edition
Prentice Hall, 2001

4. Friedman, F.L.-Koffman, E.B.
Problem Solving, Abstraction, Design using C++
Third Edition
Addison wesley Longman, 2000

5. Horstmann, C.
Computing Concepts with C++ Essentials
Second edition
John Wiely & Sons, 1999

ملحق (أ)

الكلمات المحجوزة

Reserved Words

فيما يلي قائمة بأسماء تعريفية (identifiers) تعد كلمات محجوزة ، أي أسماء تعريفية لها معاني معرَّفة سابقاً (predefined meanings) في لغة C++ . فلا يجوز للمبرمج أن يعلن (declare) عن أي منها لأي غرض آخر [كاسم متغير (variable name) مثلاً] في برنامج بلغة C++.

and	double	not	this
and_eq	dynamic_cast	not_eq	throw
asm	else	operator	true
auto	enum	or	try
bitand	explicit	or_eq	typedef
bitor	export	private	typeid
bool	extern	protected	typename
break	false	public	union
case	float	register	unsigned
catch	for	reinterpret_cast	using
char	friend	return	virtual
class	goto	short	void
compl	if	signed	volatile
const	inline	sizeof	wchar_t
const_cast	int	static	while
continue	long	static_cast	xor
default	mutable	struct	xor_eq
delete	namespace	switch	
do	new	template	

الجدول التالي يلخص أولويات المؤثرات في لغة C++ ، ومنها مؤثرات لم نشر إليها في هذا الكتاب . وقد تم تجميع المؤثرات في الجدول بناء على مستوى الأولوية (precedence level) من الأعلى أولوية إلى الأقل أولوية، بحيث يفصل خط مستقيم أفقي بين أي مستوى أولوية معين والمستوى الأقل الذي يليه.

وعموماً – في غياب الأقواس – يكون تجميع المؤثرات الثنائية (binary operators) من اليسار لليمين، وتجميع المؤثرات الأحادية (unary operators) من اليمين لليسار ، والمؤثر :? من اليمين لليسار. وهناك استثناء وهو أن تجميع مؤثرات الإسناد (assignment operators) يكون من اليمين لليسار.

ملحق (ب)

قاعدة الأولوية بالنسبة للمؤثرات

Operator Precedence

الأولوية من الأعلى إلى الأدنى

المؤثر Operator	التجميع Associative	ملاحظات Remarks
: :	من اليسار إلى اليمين	تحليل المجال (ثنائي) (scope resolution (binary)
: :	من اليمين إلى اليسار	وصول شامل (الأحادي) (global access (unary)
( )	من اليسار إلى اليمين	استدعاء دالة (function call) ، وتحويل / صب
[ ]	من اليسار إلى اليمين	
++ - -	من اليمين إلى اليسار	صياغة دالة (function – style cast)
typeid dynamic_cast static_cast const_cast reinterpret_cast	من اليمين إلى اليسار من اليمين إلى اليسار	++ و – كمؤثرين لاحقين (postfix operators)
++ -- ! Unary+ Unary- - Unary * Unary & (cast) sizeof new delete	من اليمين إلى اليسار من اليمين إلى اليسار من اليمين إلى اليسار	++ و – كمؤثرين سابقين (prefix operators)
->* .*	من اليسار إلى اليمين	
* / %	من اليسار إلى اليمين	
+ -	من اليسار إلى اليمين	
<< >>	من اليسار إلى اليمين	
< <= > >=	من اليسار إلى اليمين	
== !=	من اليسار إلى اليمين	
&	من اليسار إلى اليمين	
^	من اليسار إلى اليمين	
|	من اليسار إلى اليمين	
&&	من اليسار إلى اليمين	
||	من اليسار إلى اليمين	

?:	من اليمين إلى اليسار	
= += -= *= /= %= <<= >>= &= |= ^=	من اليمين إلى اليسار من اليمين إلى اليسار	
throw	من اليمين إلى اليسار	
,	من اليسار إلى اليمين	مـؤثر التتـابع (sequencing operator) ، ولـيس الفصل (separator)

ملحق (ج)

مجموعات/شفرت الرموز

Character Sets / Codes

الجداول التالية تبيِّن ترتيب الرموز (ordering of characters) في مجموعتي رموز (character sets) مستخدمتين على نطاق واسع لتشفير الرموز:

١) مجموعة (American Standard Code for Information Interchange) ASCII "الشفرة القياسية الأمريكية لتبادل المعلومات"

٢) مجموعة (Extended Binary Coded Decimal Interchange Code) EBCDIC "الشفرة التبادلية العشرية الثنائية الممتدة"

وتبين الجداول التمثيل الداخلي (internal representation Code) EBCDIC لكل رمز (character) في النظام العشري (decimal) . فمثلاً الحرف A يمثل داخلياً بالعدد الصحيح 65 (representered internally as the integer) في المجموعة ASCCII وبالعدد الصحيح 193 في المجموعة EBCDIC . ويرمز للفراغ (space / blank character) بالعلامة "□ ".

Right Digit / Left Digit	ASGII									
	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	Cr	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	□	!	"	#	$	%	&	`
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	G	h	i	j	k	l	M
11	n	o	p	Q	r	s	t	u	v	W
12	x	y	z	{	ا	}	~	DEL		

جدول الشفرة ASCII

الشفرات من 00 إلى 31 والشفرة 127 هي المقابلة لرموز التحكم التالية:

وهي رموز غير قابلة للطباعة (nonprintable control characters):

NUL	Null character	VT	Vertical tab	SYN	Synchronous idle
SOH	Start of header	FF	Form feed	ETB	End of transmitted block
STX	Start of text	CR	Carriage return	CAN	Cancel
ETX	End of text	SO	Shift out	EM	End of medium
EOT	End of transmission	SI	Shift in	SUB	Substitute
ENQ	Enquiry	DLE	Data link escape	ESC	Escape
ACK	Acknowledge	DC1	Device control one	FS	File separator
BEL	Bell character (beep)	DC2	Device control two	GS	Group separator
BS	Back space	DC3	Device control three	RS	Record separator
HT	Horizontal	DC4	Device control four	US	Unit separator
LF	Line feed	NAK	Negative acknowledge	DEL	Delete

والجدول التالي لشفرة ASCII يحتوي على معظم الرموز وقيم شفرة ASCII المقابلة لها بصورة مباشرة، والجدول لا يحتوي على رموز التحكم.

قيمة ASCII	الرمز Character	قيمة ASCII	الرمز Character	قيمة ASCII	الرمز Character
032	blank	063	?	093	]
033	!	064	@	094	^
034	"	065	A	095	_
035	#	066	B	096	`
036	$	067	C	097	a
037	%	068	D	098	b
038	&	069	E	099	c
039	'	070	F	100	d
040	(	071	G	101	e
041	)	072	H	102	f
042	*	073	I	103	g
043	+	074	J	104	h
044	,	075	K	105	i
045	-	076	L	106	j
046	.	077	M	107	k
047	/	078	N	108	l
048	0	079	O	109	m
049	1	080	P	110	n
050	2	081	Q	111	o
051	3	082	R	112	p
052	4	083	S	113	q
053	5	084	T	114	r
054	6	085	U	115	s
055	7	086	V	116	t
056	8	087	W	117	u
057	9	088	X	118	v
058	:	089	Y	119	w
059	;	090	Z	120	x
060	<	091	[	121	y
061	=	092	\	122	z
062	>				

جدول الشفرة ASCII

ملاحظة : هناك بعض الرموز (رموز تحكم) غير موجودة بالجدول

Right Digit / Left Digit	EBCDIC									
	0	1	2	3	4	5	6	7	8	9
6					□					
7					¢	.	<	(	+	|
8	&									
9	!	$	*	)	;	¬	-	/		
10							^	,	%	_
11	>	?								
12		`	:	#	@	`	=	"		a
13	b	c	d	e	f	g	h	i		
14						j	k	l	m	n
15	o	p	q	r						
16		~	s	t	u	v	w	x	y	z
17								\	{	}
18	[	]								
19				A	B	C	D	E	F	G
20	H	I								J
21	K	L	M	N	O	P	Q	R		
22							S	T	U	V
23	W	X	Y	Z						
24	0	1	2	3	4	5	6	7	8	9

جدول الشفرة EBCDIC

ملاحظة : رموز التحكم غير القابلة للطباعة (nonprintable control characters) **[المقابلة للشفرات من** 00 **إلى** 63 **وكذلك من** 250 **إلى** 255**] وأيضاً الرموز التي يظهر مكانها في الجدول فراغات خالية** (empty spaces) **غير مبينة بهذا الجدول.**

ملحق (د)

بعض البرامج المكتبية القياسية
Some Standard Library Routines

تحتوي مكتبة C++ القياسية على عدد كبير من أنواع البيانات والدوال والثوابت المسماة . وفي هذا الملحق نشير إلى أكثرها استخداماً. والملحق مرتب أبجدياً بالنسبة لملفات المقدمة (header files) التي يجب أن يشتمل عليها # include البرنامج قبل الوصول إلى الاستخدام هذا العناصر (أنواع البيانات والدوال والثوابت المسماة) . فمثلاً لاستخدام دالة رياضية [برنامج رياضي (a mathematics routine)] مثل sqrt يجب أن نشمل ملف المقدمة cmath كما يلي:

```
# include <cmath>
using namespace std;
  ⋮
y = sqrt (x);
```

ولا حظ أن أي اسم تعريفي (identifier) في المكتبة القياسية مُعَرَّف (defined) بأنه في فضاء الأسماء namespace std. وبدون الموجِّه using directive السابق علينا أن نكتب:

```
y = sqrt (x);
```

(١) ملف المقدمة cassert

```
cassert (boolleanExpr)
```

(٢) ملف المقدمة cctype

islnum (ch)	isalpha (ch)	iscntrl (ch)
isdigit (ch)	isgraph (ch)	islower (ch)
isprint (ch)	ispunct (ch)	isspace (ch)
isupper (ch)	isxdigit (ch)	tolower (ch)
toupper (ch)		

(٣) ملف المقدمة cfloat

FLT_DIG	FLT_MAX	FLT_MIN
DBL_DIG	DBL_MAX	DBL_MIN
LDBL_DIG	LDBL_MAX	LDBL_MIN

(٤) ملف المقدمة climits

CHAR_BITS	CHAR_MAX	CHAR_MIN
SHRT_MAX	SHRT_MIN	SHRT_MAX
INT_MIN	LONG_MAX	LONG_MIN
UCHAR_MAX	USHRT_MAX	UNIT_MAX
ULONG_MAX		

(٥) ملف المقدمة cmath

acos (x)	asin (x)	atan (x)
ceil (x)	cos (angle)	cosh (x)
exp (x)	fabs (x)	floor (x)
log (x)	log10 (x)	pow (x,y)
sin (angle)	sinh (x)	sqrt (x)
tan (angle)	tenth (x)	

(٦) ملف المقدمة cstddef

NULL

(٧) ملف المقدمة cstdlib

abs (i)	atof (str)	atoi (str)
atol (str)	exit (exitStatus)	labs (i)
rand ( )	srand (seed)	system (str)

(٨) ملف المقدمة cstring

strcat (toStr, formStr)	strcmp (str1 , str2)
strcpy (toStr, fromStr)	strlen (str)

(٩) ملف المقدمة string

string :: size_type	string :: npos	
s.c_str ( )	s.find (ard)	getline (inStream, s)
s.length ( )	s.size ( )	s.substr (pos, len)

دليل المصطلحات العربية والإنجليزية
Index

فيما يلي قائمة بالمصطلحات الإنجليزية مرتبة ترتيباً أبجدياً، والمصطلحات العربية المقابلة لها، وكلك فهرس لهذه المصطلحات.

المصطلح العربي	الصفحة	المصطلح الإنجليزي
عمليات إجمالية	٥٢٤	aggregate operations
خوارزمية	٢١	algorithms
ترتيب أبجدي	٢١٢	alphabetical order
أنواع بيانات غير مسَّماة	٤٩٥	anonymous data types
يلحق	٦٤	append
وسيط فعلي	١١٠	argument
قائمة الوسطاء	١١١	argument list
مؤثر حساب	٩٦	arithmetic operator
منظومة	٥١٣	array
اسم دالة مكتبية	٣٥٧	assert
تعبير إسناد	٤٦٨	assignment expression
عبارة إسناد	٥٩	assignment statement
الخط المائل الخلفي	٦٥	backslash
نسخة بديلة	٨٢	backup copy
الأساس	٩٣	base
عدد ثنائي	٩٣	binary number
قالب	٧٠	block
اسم عبارة	٤٢٧	break
رمز	٥٣	character

comment	٦٦	تعليق
compiler	١٧	البرنامج المترجم
compound statement	٧٠	عبارة مركبة
concatenation	٦١	تعاقب
conditional operator	٤٧٣	المؤثر الشَّرطي
continue	٤٢٧	اسم عبارة
control statement	٢٠٥	عبارة تحكم
count-controlled loop	٢٧٣	عروة محكومة بعدد
dangling (else)	٢٤٠	(else) المتدلية / التابعة
data types	٥١	أنواع البيانات
debugging	٨٢	تصحيح الأخطاء/إزالة العلل
decimal number	٩١	عدد عشري
declaration	٥٥	إعلان
decrement operator	١٠١	مؤثر النقصان
default	٤٢٨	افتراض / بديل افتراضي
directive	٧٢	موجِّه
directory	٧٣	دليل
diskette	٨٢	قريص
do..while	٤٢٧	اسم عبارة
double precision	٩٢	دقة مضاعفة / مزدوجة
editor	٨٢	المعدِّل
end-of-file controlled loop	٢٧٧	عروة محكومة بنهاية ملف
enumeration type	٤٨٣	نوع تعددي
error message	٥٥	رسالة خطأ
event-controlled loop	٢٧٤	عروة محكومة بحدث
exit status	٧٠	حالة الخروج

explicit conversion	١٠٦	تحويل صريح
exponent	٩٣	الأس
expression	٥٩	تعبير
extraction operator	١٣٥	مؤثر الاستخلاص / الاستخراج
failure	٨٢	عطل / تلف
field width	١١٩	عرض المجال
file	١٥١	ملف
find	١٢٦	اسم دالة (البحث)
flag controlled loop	٢٧٩	عروة محكومة براية
floating point number	٨٨	عدد ذو علامة كسرية عائمة
flowchart	٢١	خريطة سير العمليات
FOF (End of File)	٣٤	نهاية الملف
for	٤٢٧	اسم عبارة
fractional part	٨٨	الجزء الكسري
function call	١٠٩	استدعاء الدالة
function heading	٦٩	عنوان الدالة
functional decomposition	١٦٥	التفكيك/التقسيم/التحليل الدالي
get	١٤٢	اسم دالة (لقراءة الرموز)
getline	١٤٨	اسم دالة (القراءة سطر المدخلات)
hardware	٨٢	مكونات مادية
header file	٧٢	ملف المقدمة
identification number	١٨	رقم تعريفي
If statements	٢٢٣	عبارات If (إذا كان)
ignore	١٤٥	اسم دالة (لتخطي الرموز)
implicit conversion	١٠٥	تحويل ضمني
increment operator	١٠١	مؤثر الزيادة

Index	٥٢٢	مؤشر
input stream	١٣٥	سيل المدخلات
integer	٨٨	عدد صحيح
integral type	٨٨	نوع تكاملي
interactive	١٤٩	برنامج تبادلي
interface	١٧١	بينية
length	١٢٤	اسم دالة (الطول)
lifetime of variable	٣٧٣	عمر / مدى / فترة حياة متغير
literal string	٦٢	سلسلة حرفية
local variable	٣٣٨	متغير محلي
logic error	٨٢	خطأ منطقي
logical expression	٢٠٦	تعبير منطقي
logical operator	٢١٢	مؤثر منطقي
loop	٢٦٩	عروة
main function	٤٩	الدالة الرئيسية
manipulator	١١٧	معالج
memory	٨٢	ذاكرة
name hiding	٣٦١	إخفاء الأسماء
name precedence	٣٦١	أولوية الأسماء
named data types	٤٩٥	أنواع بيانات مسماة
namespace	٧٣	فضاَء الأسماء
nested If statements	٢٣٣	عبارات If المتداخلة
nested loop	٢٨٧	عُرَى متداخلة
newline character	١٤٠	رمز السطر الجديد
null statement	٧٠	العبارة الخالية
object	١٦٧	هدف

Object_Oriented Design (OOD)	١٦٩	تصميم موجه نحو الهدف
one-dimensional array	٥١٧	منظومة أحادية البعد
open	١٥٥	اسم دالة (لفتح الملف)
operand	٦٢	معامَل (كمية مشغَّلة)
operating system	٤٨	نظام التشغيل
operator	٦٠	مؤثر
output stream	٦٩	سيل المخرجات
overflow	٩١	فيض زائد
parameter	١١٠	وسيط
postfix operator	١٠١	مؤثر لاحق
precedence rule	١٠٢	قاعدة الأولوية
prefix operator	١٠١	مؤثر سابق
preprocessor	٧١	المشغل المبدئي
processing	٢٢	معالجة / تشغيل
prompting message	١٨	رسالة تنبيهيه / رسالة حث
prototype	٣٣٦	نموذج (دالة)
qualified name	٧٤	اسم مؤهَّل
range	٩١	المدى
reference parameter	٣٤٤	وسيط إسناد
relational expression	٢٠٨	تعبير علاقي
relational operator	٢٠٧	مؤثر علاقي
repetition statements	٢٦٩	عبارة تكرار
reserved word	٥١	كلمة محجوزة
retrieve	٦١	يسترجع
running a program	٧٩	تشغيل برنامج
scientific notation	٩٣	الاصطلاح العلمي

scope of identifier	٣٥٩	مجال الاسم التعريفي
scope rules	٣٦٢	قواعد المجال
selection	٢٠٥	اختبار
semantic error	٢٤٩	خطأ معنوي (منطقي)
sentinel-controlled loop	٢٧٤	عروة محكومة بقيمة حارسة
sequence	٨٨	متتابعة
short-circuit evaluation	٢١٦	تقييم سريع / قصير الدائرة
side effect	٣٧٧	أثر جانبي
significant digit	٩٢	رقم معنوي
size	١٢٥	اسم دالة (السعة)
sizeoff	٤٧٢	اسم مؤثر (للسعة)
software	١٥٥	برمجيات
standard library	١١٨	مكتبة قياسية
string of characters	٥٣	سلسلة رموز
structured type	٨٨	نوع مبني
substr	١٢٩	اسم الدالة (السلسلة الجزئية)
substring	١٢٦	سلسلة رموز جزئية
switch	٤٢٧	اسم عبارة
syntax error	٨١	خطأ تركيبي
truth table	٢١٥	جدول الصحة
type casting	١٠٦	صب النوع / تحويل صريح للنوع
type coercion	١٠٥	تحويل قسري/تلقائي/ضمني
type conversion	١٠٤	تحويل النوع
Typedef	٥٣٥	اسم عبارة
unary	٩٧	أحادي
user-defined function	٣٣٠	دالة معرفة بالمستخدم

value parameter	٣٤٢	وسيط ذو قيمة
value returning function	٤٧	دالة تعيد قيمة
variable	٥٦	متغير
void function	١١٦	دالة خاوية
void function	٣٢٧	دالة خاوية
While statements	٢٦٩	عبارة while (بينما / طالما)
whitespace characters	١٣٨	الرموز الفرعية البيضاء
write statement	٦٣	عبارة الكتابة

محتويات الكتاب

الموضوع	صفحة
المقدمة	٥
فصل تمهيدي: نظرة عامة على البرمجة بلغة C++	١٥
الفصل الأول: الخوارزميات وخرائط سير العمليات	٢١
الفصل الثاني: أساسيات لغة البرمجة C++	٤٥
الفصل الثالث: عبارات التحكم / الاختيار	٢٠٥
الفصل الرابع: عبارات التكرار / العُرَى	٢٦٩
الفصل الخامس : الدوال	٣٢٥
الفصل السادس: عبارات إضافية للتحكم والتكرار	٤٢٧
الفصل السابع: الأنواع البسيطة للبيانات	٤٦٥
الفصل الثامن: المنظومات	٥١٣
تمرينات عامة	٥٧٩
أجوبة تمرينات الكتاب	٥٨٩
بعض المراجع عن البرمجة بلغة C++	٦٩٠
ملاحق الكتاب	٦٩١
- ملحق (أ): الكلمات المحجوزة	
- ملحق (ب): قاعدة الأولوية للمؤثرات	
- ملحق (ج): مجموعات الرموز / شفرات الرموز	
- ملحق (د): بعض البرامج المكتبية القياسية	
دليل المصطلحات العربية والإنجليزية	٦٩٩

كتب للمؤلف

من منشورات دار القلم .. الكويت

١- برمجة الحاسب بلغة الفورتران ، ط ٤ ، ١٩٩٢.

٢- مقدمة في نظرية المعلومات ، ط ٢ ، ١٩٩٣.

٣- الشبكات الرقمية ، ١٩٨٦.

٤- التحليل العددي ، ١٩٨٨.

٥- الجبر الخطي ، ط ٢ ، ١٩٩٥.

٦- برمجة الحاسب بلغة الباسكال ، ط٢ ، ١٩٩٩.

٧- البرمجة المتقدمة بلغة الباسكال ، مع د. حمزة رشوان ، ١٩٩٤.

٨- الدوائر المتكاملة الرقمية (ترجمة) ، ١٩٩٣.

٩- الخوارزميات والبرمجة بلغة الباسكال ، ١٩٩٧.

١٠- بنى المعطيات ، مع د. حمزة رشوان ، ١٩٩٨.

١١- الحلول العددية للمعادلات التفاضلية العادية ، ٢٠٠٠.

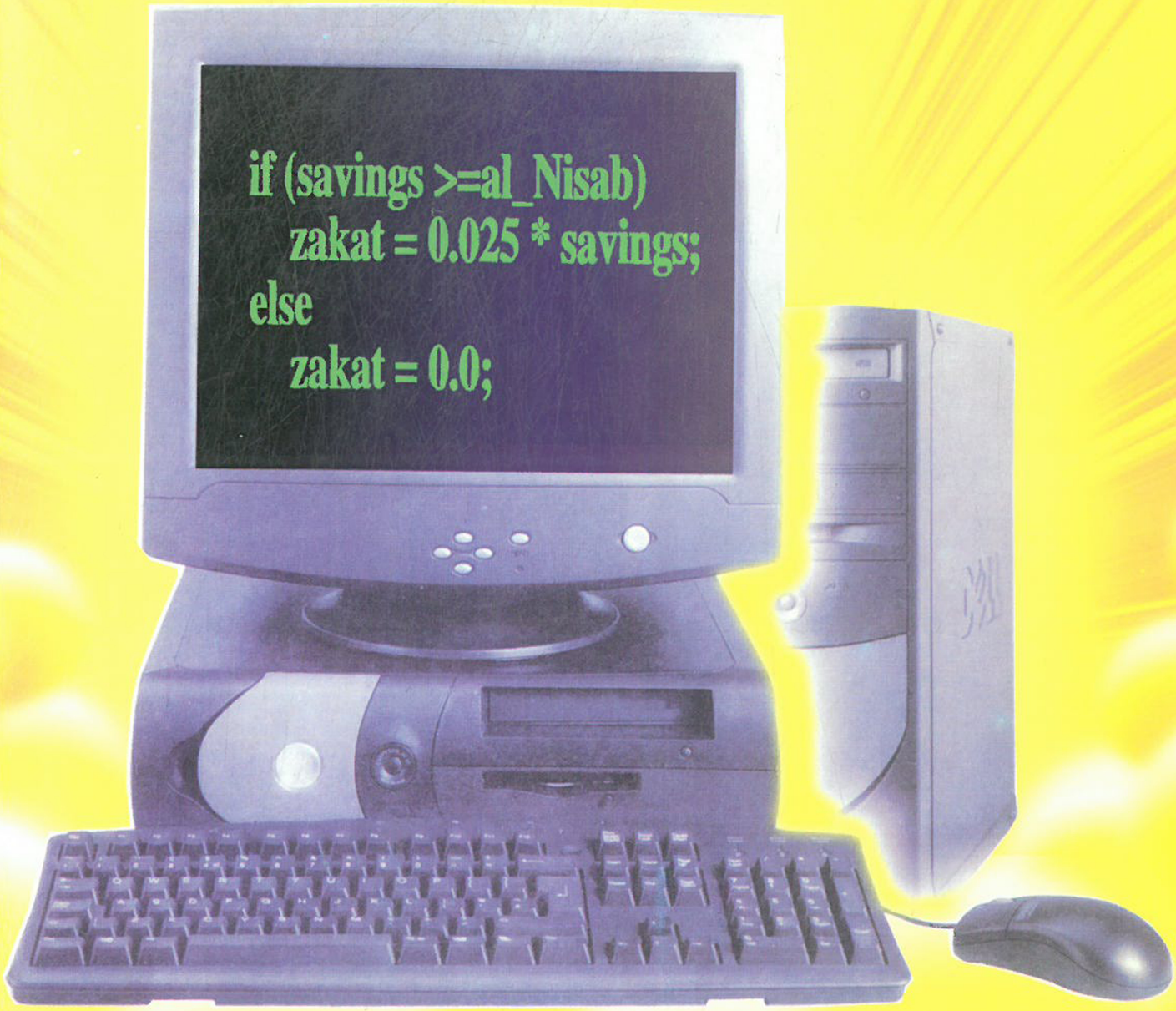
١٢- الحول العددية للمعادلات التفاضلية الجزئية ، ٢٠٠١.

١٣- برمجة الحاسب بلغة C++ ، ٢٠٠٢.

﴿ سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا إِنَّكَ أَنْتَ الْعَلِيمُ الْحَكِيمُ ﴾

(سورة البقرة : ٣٢)

Computer Programming in C++



Dr. Abu-Bakr Ahmed El-Sayed
Department of Mathematics and Computer Science
University of Kuwait