برمجة الحاسب

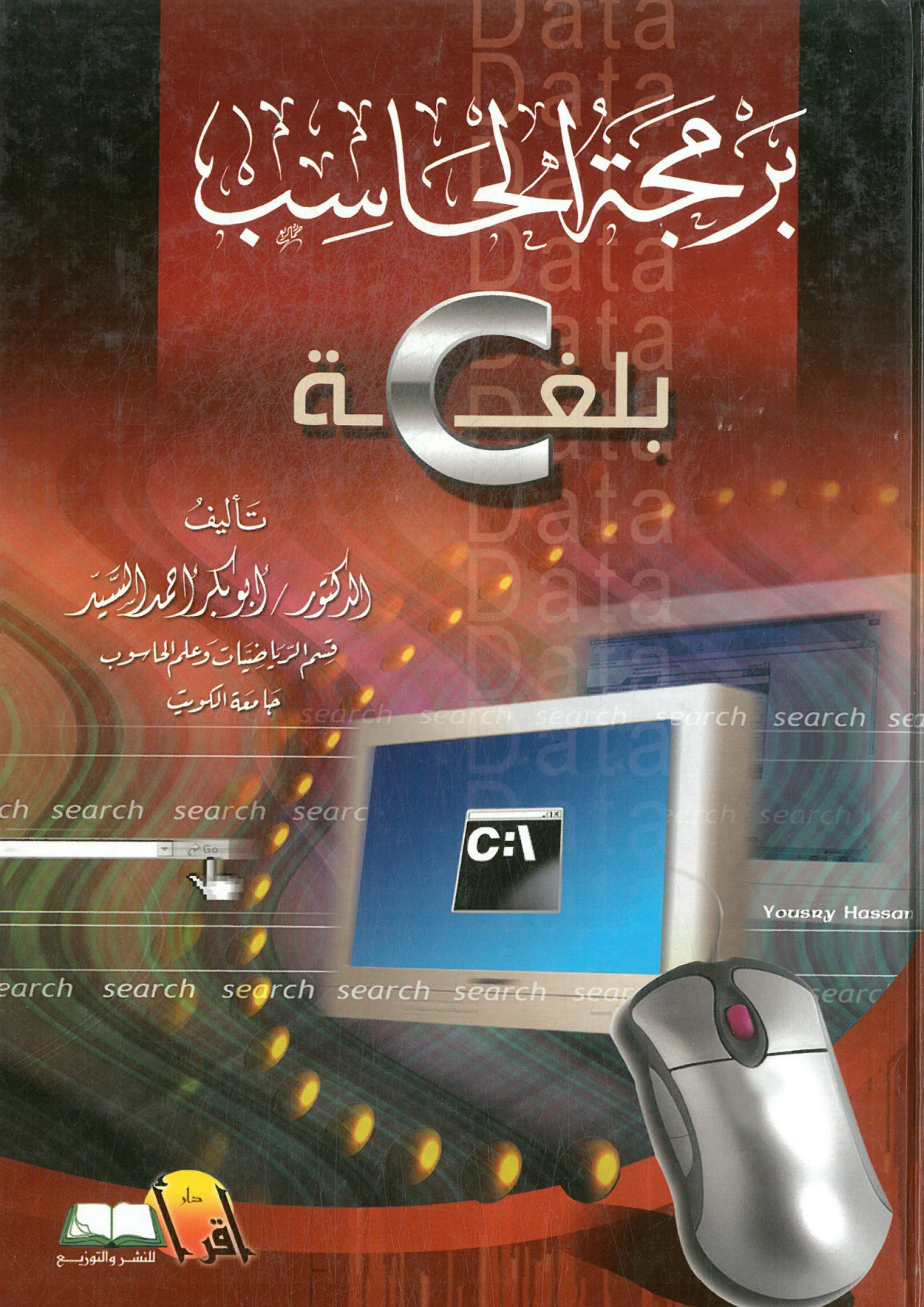
بلغة C

تأليف

الدكتور / أبو بكر أحمد السيد

قسم الرياضيات وعلم الحاسوب

جامعة الكويت



دار اقرأ للنشر والتوزيع

برمجة الحاسب

بلغة C

برمجة الحاسب بلغة C

الدكتور أبو بكر أحمد السيد

قسم الرياضيات وعلم الحاسوب
جامعة الكويت

الطبعة الأولى

١٤٢٨ هـ – ٢٠٠٧ م

دار اقرأ للنشر والتوزيع

الكويت – حولي – شارع المثنى

ص ـ ب: ١٩٣٧ حولي – الرمز البريدي 32020

هاتف : ٢٦٥٥٣٤٠ – ٢٦٥٥٣٥٠ فاكس : ٢٦٥٥٣٥٠

بِسْمِ اللّهِ الرَّحْمَنِ الرَّحِيمِ

﴿ صبغةَ 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 .

Computer Programming in

C

Dr. Abu-Bakr Ahmad El-Sayed

Dept. of Mathematics and Computer Science
University of Kuwait

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

---

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

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 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.

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 .

(*) 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 .

(*) المصدر السابق .

منهج هذا الكتاب ومحتوياته

وهذا الكتاب خطوة جديدة على طريق أسلمة العلوم وتعريبها ، وذلك في علم برمجة الحواسيب الإلكترونية بلغة C ، 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) ، وكتاب (برمجة الحاسب بلغة C++) ، ولذلك نقتبس هنا ما ورد في مقدمة كل من هذه الكتب 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 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 ومعالجة الشروط والعمليات الحسابية ، بحيث يستطيع الطالب مع نهاية هذا الفصل كتابة برامج قصيرة . ويشتمل الفصل الثالث على عبارات وبِنَى التحكم الأساسية (عبارات الاختيار `if` وعبارات التكرار `while`) . وأما الفصل الرابع فيعرض عبارات إضافية للتحكم والاختيار والتكرار (عبارات التكرار `for, do..while` ، وعبارة الاختيار `switch`) وكذلك المؤثرات المنطقية . ويتناول الفصل الخامس موضوع الدوال وتطبيقاتها ، بما في ذلك الدوال المكتبية القياسية ، والدوال التي يعرِّفها المستخدم . وأما الفصل السادس فهو خاص بالمنظومات أو مجموعات البيانات المرتبطة ببعضها والتي هي جميعها من النوع نفسه ، مع بعض التطبيقات الهامة كعمليات البحث عن عناصر معينة أو ترتيب منظومة العناصر ترتيبا معينا (تصاعديا مثلا أو تنازليا) . والفصل السابع والأخير يختص بكيفية التحكم في صياغة المدخلات والمخرجات .

شكر وتقدير

ونود هنا أن نتقدم بشكرنا الجزيل لدار اقرأ للنشر والتوزيع بالكويت لتعاونها 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 .

ولا يفوتنا أن نشكر الأخت الفاضلة / سالمة موسى بقسم الرياضيات وعلم الحاسوب بجامعة الكويت على ما بذلته من جهد كبير وما تحلت 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 حسناتها لما يساهم بإذن الله 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) .

الفصل الأول

الخوارزميات وخرائط سير العمليات
Algorithms and Flowcharts

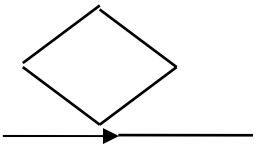
مفهوم الخوارزميات وخرائط سير العمليات :

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 C 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

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

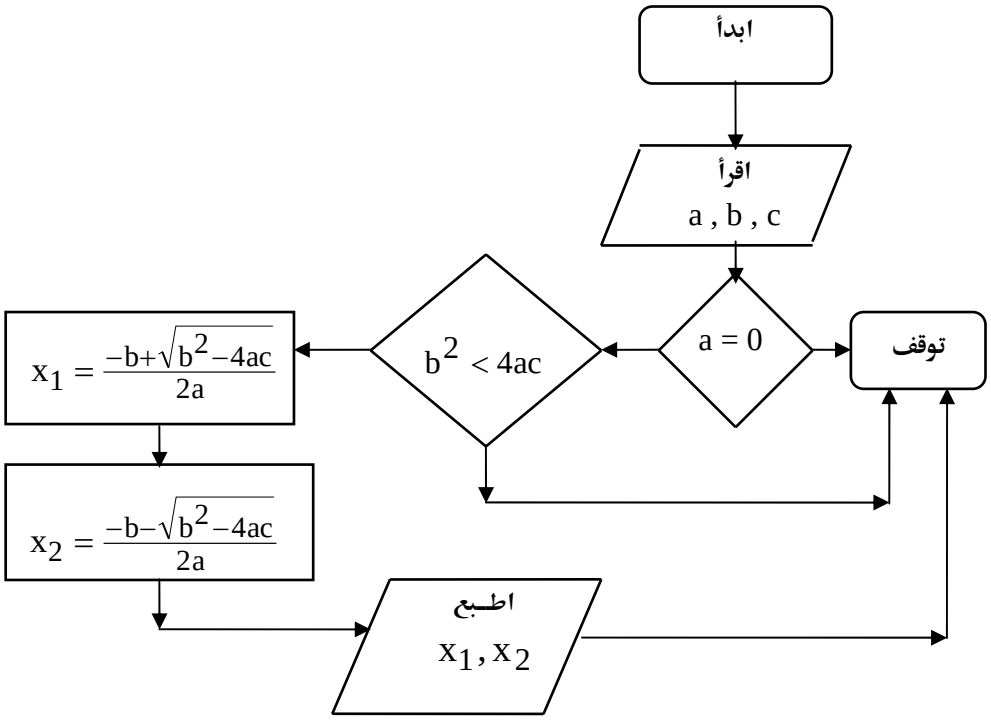
٦- احسب قيمة الجذر الثاني

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

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 :

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

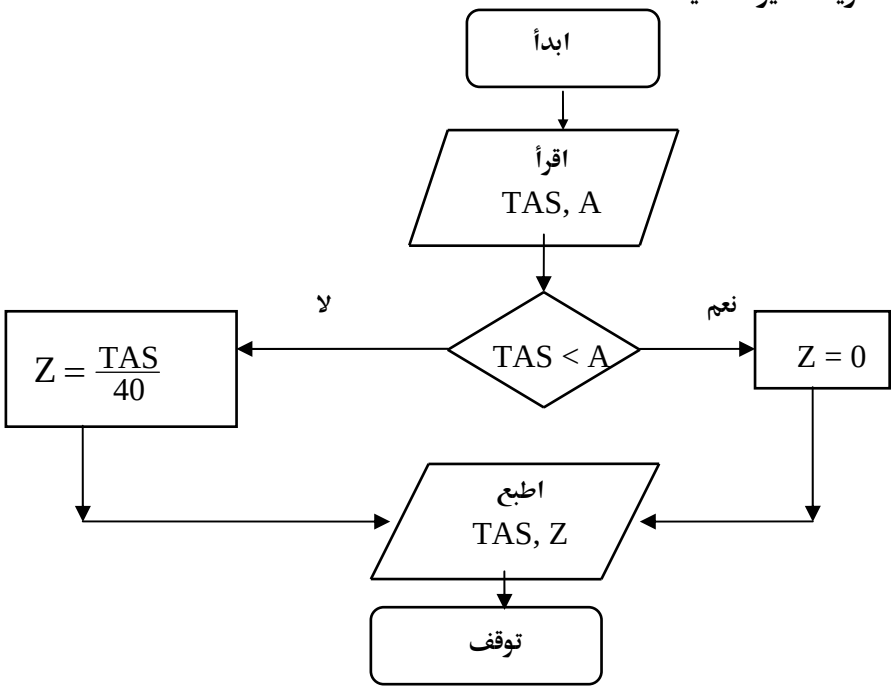
الخوارزمية :

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 :

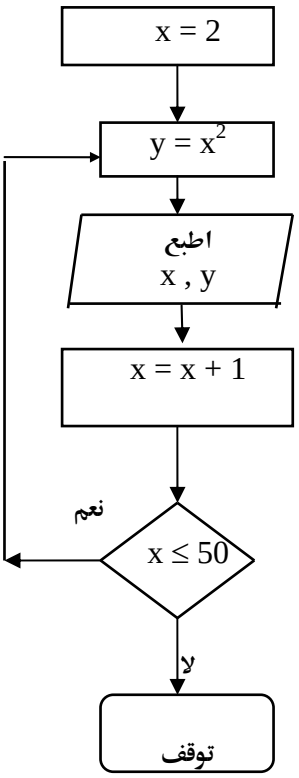
القانون : $y = x^2$; $x = 2 , 3 , 4 , ... , 50$

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 .

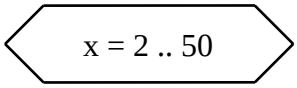
وبالنسبة لتنفيذ الحاسب لخطوات هذا الحل إلى أن يتوقف نلاحظ ما يلي:

- **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**

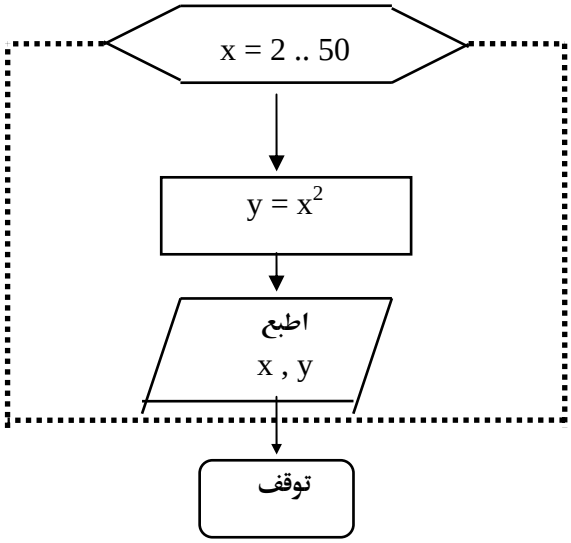
$$x = 3 + 1 = 4$$

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 $x = x + 1$

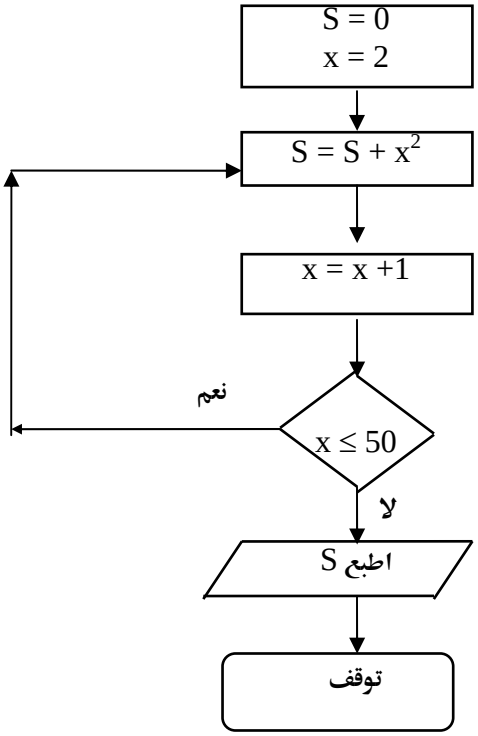
٥- إذا كانت $x \leq 50$ اذهب إلى الخطوة رقم ٣

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 :

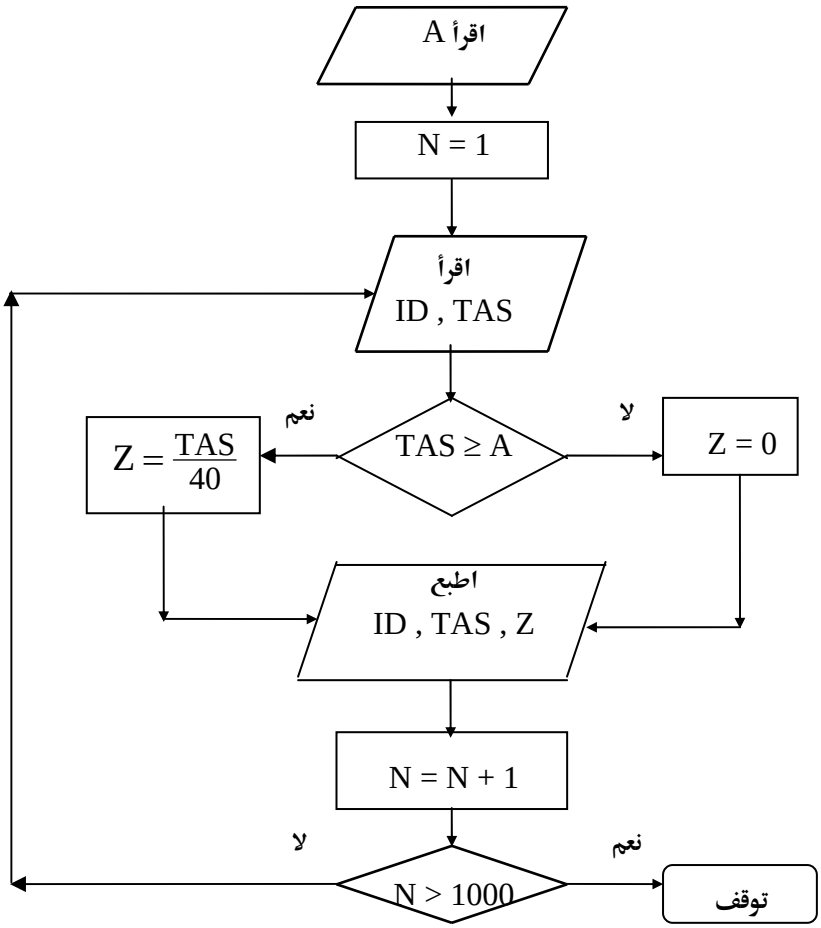
(أ) قراءة قيمة ID (رقم تعريفي للشخص identification Number) وقيمة TAS (المدخرات السنوية الكلية) .

(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 .



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

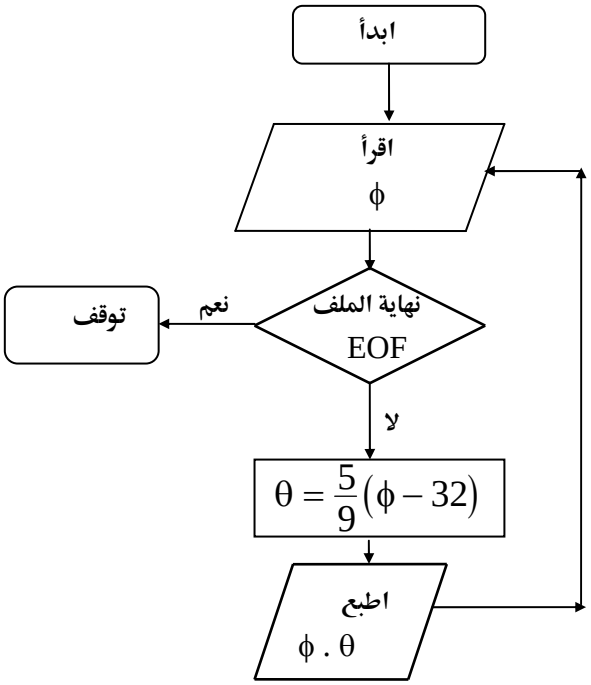
$$\theta = \frac{5}{9}(\phi - 32)$$

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 .



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 N , ZS .

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 بينها : 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 .

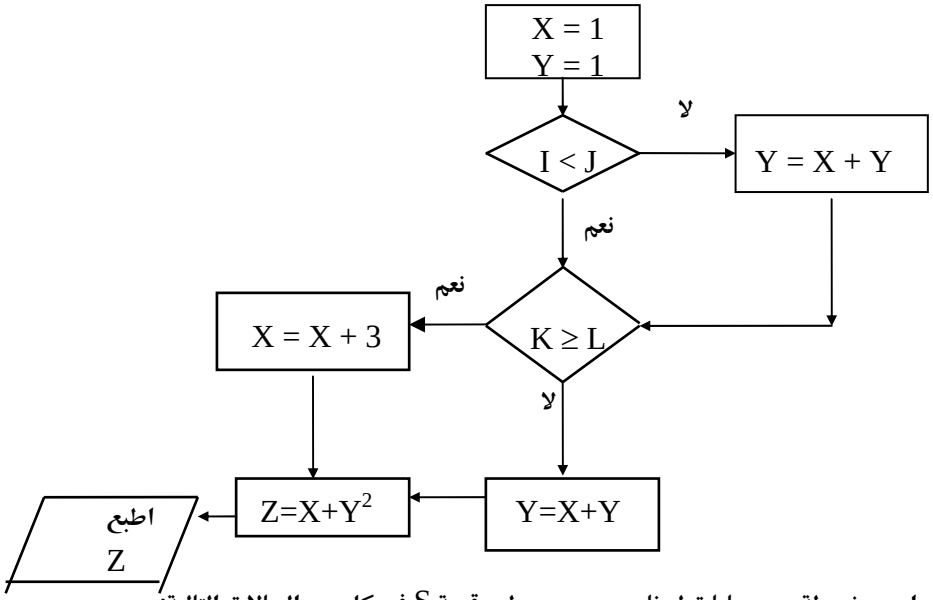
(ب) عدد كتب التفسير والحديث M .

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 $|f(c)| < \varepsilon$ 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 $x_0 , \varepsilon , n_{max}$

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 R , T 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 N , XFI , XIN .

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 :

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

١٨-١ 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.

١٩-١ 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 .

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

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

٢١-١ 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 :

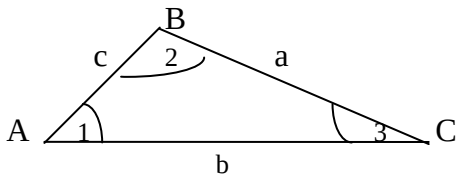
$$x = \frac{b_2c_1-b_1c_2}{a_1b_2-a_2b_1} \quad , \quad y = \frac{a_1c_2-a_2c_1}{a_1b_2-a_2b_1}$$

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

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 العروض 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

نبدأ بعون الله تعالى بعرض برنامج بسيط بلغة C يطبع سطرا نصّيا (a line of text) على الشاشة . ومع بساطة هذا البرنامج إلا أنه يوضح كثيرا من الملامح الأساسية لبرنامج مكتوب بلغة C .

مثال ٢-١ : يقول تعالى : " اليوم أكملت لكم دينكم ، وأتممت عليكم نعمتي ، ورضيت لكم الإسلام دينا " (المائدة : ٣) .

نفرض أن شخصاً ما قد هداه الله تعالى وشرح صدره للإسلام فاعتنقه . اكتب برنامجا يطبع على الشاشة الرسالة : Welcome to Islam ! للترحيب به وبدخوله في الإسلام .

الحل :

```
 */ Example 2-1
   A first program in C* /
#include <stdio.h>

 */function main begins program execution * /
int main ( )
{
   printf( "Welcome to Islam!\n" ) ;

   return 0; /* indicate that program ended successfully * /

  } / * end function main * /
```

```
Welcome to Islam!
```

برنامج طباعة نص

Text printing program

*) نلاحظ أن السطرين الأولين يبدآن بالرمزين المتعاقبين * / وينتهيان بالرمزين المتعاقبين / * ، وهذا يعني أن هذين السطرين عبارة عن تعليق (comment). وعموما تضاف التعليقات للبرامج لتوثيقها (documentation) وفهمها وقراءتها بسهولة (readability). وهذه التعليقات لا تجعل الحاسوب يقوم بتنفيذ أي عمليات أثناء تشغيل البرنامج . كما أن التعليقات يتم تجاهلها (ignored) بواسطة البرنامج المترجم (C compiler) ولا تؤدي إلى توليد (generating) أي برنامج / كود بلغة الآلة (machine language object code) . والتعليق المذكور في البرنامج السابق يذكر رقم المثال ، واسم الملف ، والغرض من البرنامج .

*) السطر # include <stdio.h>

يعتبر موجّها إلى المشغّل الابتدائي (directive to the C preprocessor) . وأي سطر يبدأ بالرمز # يتم تشغيله بالمشغّل الابتدائي قبل ترجمة البرنامج (compiling the program) . والسطر الحالي يطلب من المشغّل الابتدائي أن يدرج محتويات ملف المقدمة القياسي للإدخال والإخراج (standard input / output header) (stdio.h) في البرنامج . وهذه المقدمة (header) تحتوي على المعلومات التي يستخدمها البرنامج المترجم عند ترجمة الاستدعاءات (compiling calls) إلى الدوال المكتبية (library functions) القياسية للإدخال والإخراج كالدالة **printf** .

*) السطر int main ( )

هو جزء من أي برنامج بلغة C . والقوسان بعد كلمة **main** يدلان على أن **main** عبارة عن قالب بناء برنامج (program building block) يطلَق عليه دالة (a function). وأي برنامج بلغة C يحتوي على دالة أو أكثر ، على أن تكون إحداها **main** . ويبدأ تنفيذ أي برنامج عند الدالة **main** .

*) القوس الأيسر { يجب أن يبدأ " جسم " (body) أي دالة ، ويقابله قوس أيمن } (وهو الموجود في آخر سطر في البرنامج) لينهي الدالة . وهذان القوسان مع جزء البرنامج الموجود بينهما يطلق عليهم " قالب " (a block) . والقالب هو وِحدة (unit) هامة من وحدات البرنامج المكتوب بلغة c .

*) السطر printf ("Welcome to Islam!\n ") ;

يوجّه الحاسوب لأن يطبع على الشاشة سلسلة الرموز (string of characters) المحصورة بين علامتي الاقتباس (quotation marks) . وسلسلة الرموز يطلق عليها أحيانا " رسالة " (a message) ، أو " سلسلة حرفية " (a literal) . والسطر كله [بما في ذلك اسم الدالة **printf** ، ووسيطها (argument) المحصور بين قوسين ، والفاصلة المنقوطة ; (semicolon)] يطلق عليه " عبارة " (a statement) . وأي عبارة يجب أن تنتهي بفاصلة منقوطة [والتي تُعرَف أيضا باسم " مُنهية العبارة " (statement terminator)] .

وعندما تنفَّذ عبارة **printf** السابقة ، فإنها تطبع الرسالة Welcome to Islam! على الشاشة . وتتم طباعة الرموز كما تظهر بالضبط بين الحاصرتين المزدوجتين (double quotes) في عبارة **printf** . ولاحظ أن الرمزين \n لم يُطبعا على الشاشة . ورمز الشَّرطة المائلة الخلفية (\ ) (backslash) يطلق عليه " رمز الهروب " (escape character) . وهو يعني أن المفروض أن تعمل الدالة **printf** شيئا غير المعتاد . وعندما نصادف رمز الشرطة المائلة الخلفية في سلسلة (string) فإن البرنامج المترجم ينظر للأمام إلى الرمز التالي ويجمعه مع الشرطة المائلة الخلفية ليكوّن " متتابعة هروب " (escape sequence) . ومتتابعة الهروب n \ تعني " سطرا جديدا " (newline) . وعندما يظهر سطر جديد في السلسلة (string) التي تطبعها الدالة (**printf**) فإن السطر الجديد يجعل مؤشر الشاشة (cursor) يوضع عند بداية السطر التالي على الشاشة .

والجدول التالي يعرض بعض متتابعات الهروب الشائعة الاستخدام .

متتابعة الهروب **Escape sequence**	الوصف **Description**
\ n	سطر جديد (Newline) : ضع مؤشر الشاشة عند بداية السطر التالي .
\ t	جدولة أفقية (Horizontal tab) : حَرِّك مؤشر الشاشة إلى موقف الجدولة التالي (next tab stop) .
\ a	تنبيه (Alert) : أطلق جرس النظام (sound the system bell).
\\	الشرطة المائلة الخلفية (Backslash) : أدخل رمز الشَّرطة المائلة الخلفية في سلسلة رموز .
\ "	حاصرة مزدوجة (Double quote) : أدخل رمز الحاصرة المزدوجة في سلسلة رموز .

جدول ٢ –١

بعض متتابعات الهروب شائعة الاستخدام

Some common escape sequences

) عبارة return 0; / indicate that program ended successfully * / توضع في نهاية كل دالة رئيسية (function) **main** . وكلمة **return** سنستخدمها للخروج من عدة دوال ، وحينما تكون الدالة رئيسية **main** فإننا نضع 0 للدلالة على أن البرنامج قد انتهى بنجاح .

*) القوس الأيمن } – في السطر الأخير من البرنامج – يدل على أننا قد وصلنا إلى نهاية البرنامج / نهاية الدالة **main** .

ملاحظة : يمكن للدالة **printf** أن تطبع الرسالة Welcome to Islam! بعدة طرق مختلفة .

مثال ٢ -٢ : البرنامج التالي يطبع مخرجات برنامج مثال ٢-١ نفسها ولكن باستخدام عبارتَي **printf** المذكورتين في البرنامج بدلا من عبارة **printf** واحدة في برنامج مثال ٢-١ ، حيث تقوم العبارة الأولى بطباعة كلمة Welcome يليها فراغ واحد ، ثم تبدأ العبارة الثانية بطباعة الرسالة to Islam! على السطر نفسه مباشرة بعد الفراغ .

```
/* Example 2-2
  printing on one line with two printf statements */
#include <stdio.h>

/* function main begins program execution */
int main()
{
  printf( "Welcome " );
  printf( "to Islam!\n" );

  return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Welcome to Islam!
```

ويمكن لعبارة طباعة **printf** واحدة أن تقوم بطباعة عدة سطور باستخدام رموز سطر جديد إضافية (additional newline characters) .

مثال ٢-٣ : البرنامج التالي يطبع الرسالة نفسها الواردة في مخرجات برنامج مثال ٢-١ ولكن على ثلاثة سطور (بدلا من سطر واحد) وباستخدام عبارة **printf** واحدة .

```
/* Example 2-3
  Printing multiple lines with a single printf */
# include <stdio.h>

/* function main begins program execution */
int main()
{
  printf( "Welcome\nto\nIslam!\n" );

  return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Welcome
To
Islam!
```

نلاحظ أنه كلما قابلنا متتابعة الهروب \n [أي سطر جديد (newline)] – في عبارة **printf** – فإن المخرجات تستمر في الظهور عند بداية السطر التالي .

مثال ٢-٤ : اكتب برنامجا يستخدم الدالة المكتبية القياسية **scanf** لقراءة / للحصول على عددين صحيحين (integers 2) يدخلهما المستخدِم (user) عن طريق لوحة المفاتيح (keyboard) ، ثم يقوم البرنامج بحساب (computing) مجموع العددين / القيمتين ، وطباعة النتيجة باستخدام الدالة **printf** .

الحل :

```
/* Example 2-4
   Addition program */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int integer1; /* first number to be input by user  */
   int integer2; /* second number to be input by user */
   int sum;      /* variable in which sum will be stored */

   printf( "Enter first integer\n" ); /* prompt */
   scanf( "%d", &integer1 );         /* read an integer */

   printf( "Enter second integer\n" ); /* prompt */
   scanf( "%d", &integer2 );          /* read an integer */

   sum = integer1 + integer2; /* assign total to sum */

   printf( "Sum is %d\n", sum ); /* print sum */

   return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

*) السطور الثلاثة الأولى في جسم الدالة **main** هي تعريفات (definitions) لمتغيرات ثلاثة أسماؤها : **sum , integer2 , integer1** . والمتغير (variable) هو موقع (location) في الذاكرة (memory) حيث يمكننا تخزين (storing) قيمة (value) فيه ليستخدمها البرنامج . والتعريفات المذكورة تحدد أن المتغيرات الثلاثة جميعها من النوع int (type) ، والذي يعني أن هذه المتغيرات ستحتفظ بقيم صحيحة (integer values) ، أي أعداد كاملة (whole numbers) مثل : 31914 , 0 , - 11 , 7 . وجميع المتغيرات في أي برنامج يجب أن تُعرَّف بـ: اسم (a name)، ونوع بيانات (a data type) مباشرة بعد القوس الأيسر { (الذي يبدأ جسم الدالة **main**) وقبل إمكانية استخدامها في البرنامج . وهناك أنواع بيانات أخرى خلاف int في لغة C.

ويلاحظ أن التعريفات الثلاثة السابقة يمكن أن تُجمع في عبارة تعريف واحدة كما يلي :

```
int integer1,  integer2 ,  sum;
```

واسم المتغير في لغة C هو أي اسم تعريفي صالح / صحيح (any valid identifier). والاسم التعريفي (identifier) هو متسلسلة رموز (series of characters) مكونة من حروف (letters) ، وأرقام (digits) ، وشرطات سفلية (_) (underscores) ، على ألا تبدأ برقم . وطول الاسم التعريفي (أي عدد رموزه) اختياري ، ولكن الرموز الـ ٣١ الأولى فقط هي المطلوبة لتتعرَّف عليها البرامج المترجمة في C (C compilers) ،وذلك بناء على لغة ANSI C القياسية . ولغة C من اللغات الحساسة لحالة الرمز (case sensitive) بمعنى أن الحرف الكبير والحرف الصغير (uppercase and lowercase letters) يُعدَّان حرفين مختلفين في لغة C ، فمثلا A1 , a1 اسمان تعريفيان مختلفان . ومن الأخطاء الشائعة كتابة حرف كبير حيث يجب أن يكون الحرف صغيرا ، مثل كتابة اسم الدالة الرئيسية Main بدلا من main . ومن الأفضل عموما اختيار أسماء للمتغيرات تدل على معانيها ، حيث يساعد ذلك على أن يصبح البرنامج ذاتي التوثيق ويحتاج إلى تعليقات أقل .

ويجب وضع التعريفات بعد القوس الأيسر للدالة وقبل أي عبارات تنفيذية (executable statements) . فمثلا في البرنامج السابق إذا وضعنا التعريفات بعد عبارة الطباعة **printf** الأولى فسيؤدي ذلك إلى حدوث " خطأ تركيبي " (syntax error) .

والخطأ التركيبي يحدث عندما لا يستطيع البرنامج المترجم التعرُّف على عبارة . والبرنامج المترجم عادة يصدر رسالة خطأ (error message) ليساعد المبرمج على التعرف على موضع العبارة الخاطئة وتصحيحها . والأخطاء التركيبية تُعد خرقا لقواعد اللغة . ويُطلق على الأخطاء التركيبية أيضا " أخطاء الترجمة " (compile errors) ، أو " أخطاء وقت الترجمة " (compile - time errors) . ومن أخطاء البرمجة الشائعة التي تؤدي إلى حدوث أخطاء تركيبية وضع تعريفات المتغيرات بين العبارات التنفيذية .

*) عبارة

```
printf( "Enter first integer\n" ); /* prompt */
```

تطبع الرسالة الحرفية : Enter first integer

على الشاشة ، وتضع مؤشر الشاشة عند بداية السطر التالي . وهذه الرسالة يُطلق عليها : تنبيه (prompt) لأنها تطلب من المستخدم اتخاذ إجراء معين .

*) العبارة التالية

```
scanf( "%d", &integer1 );    /* read an integer */
```

تستخدم الدالة **scanf** للحصول على قيمة من المستخدِم . والدالة **scanf** تأخذ قيما مُدخلة من المدخلات القياسية (standard input) والتي هي عادة لوحة المفاتيح (keyboard) . والدالة scanf في هذه العبارة لها وسيطان (two arguments) : "%d", &integer1 . الوسيط الأول – وهو سلسلة التحكم في الصيغة (format control string) – يدل على نوع البيانات التي يجب أن يُدخلها المستخدِم . ومحدِّد التحويل (conversion specifier) ٪ d يدل على أن البيانات يجب أن تكون : عددا صحيحا (an integer) [حيث الحرف d يرمز إلى : عدد صحيح عشري (decimal integer)] . والرمز ٪ في هذا السياق يُنظَر إليه في الدالة **scanf** (وكذلك الدالة **printf** كما سنرى بإذن الله) على أنه رمز خاص يبدأ محددا للتحويل . وأما الوسيط الثاني في الدالة **scanf** فيبدأ بالعلامة (&) (ampersand) – والتي يطلق عليها :" معامِل / مؤثِّر العنوان في لغة C " (address operator in C) – يليها اسم المتغير . وعندما ترتبط علامة & مع اسم المتغير فإنها تخبر الدالة **scanf** الموقع في الذاكرة الذي يوجد عنده المتغير (وهو في هذه الحالة integer1) . ثم يقوم الحاسوب بتخزين قيمة integer1 في هذا الموقع . وعندما يقوم الحاسوب بتنفيذ عبارة **scanf** السابقة ، فإنه ينتظر من المستخدم أن يُدخِل قيمة للمتغير integer1 . فيقوم المستخدم بإدخال / بطباعة (typing) عدد صحيح ثم الضغط على مفتاح (key) **Return** [وأحيانا يطلق عليه مفتاح (key) Enter] لإرسال هذا العدد إلى الحاسوب . وعندئذ يقوم الحاسوب بإسناد هذا العدد / هذه القيمة للمتغير **integer1** . وأي إشارة تالية في البرنامج لهذا المتغير سوف تستخدم هذه القيمة

نفسها . والدالتان **scanf , printf** تسهِّلان التفاعل (interaction) بين المستخدم والحاسوب . ونظرا لأن هذا التفاعل يشبه الحوار (dialogue) ، فغالبا ما يطلق عليه الحساب الحِواري (conversational computing) أو الحساب التبادلي (interactive computing) .

*) العبارة

```
printf ( "Enter second integer\n" ); /* prompt */
```

تعرض الرسالة Enter second integer على الشاشة ، ثم تضع مؤشر الشاشة عند بداية السطر التالي .

*) العبارة

```
scanf ( "%d", &integer2 );    /* read an integer */
```

تحصل على قيمة للمتغير integer2 من المستخدم .

*) عبارة الإسناد :

```
sum = integer1 + integer2; /* assign total to sum */
```

تحسب مجموع المتغيرين **integer1 , integer2** ، وتسند النتيجة للمتغير **sum** باستخدام مؤثِّر / معامِل الإسناد = (assignment operator) . وكل من المؤثِّرين / المعامِلين = , + يطلق عليه " مؤثر / معامِل ثنائي " (binary operator) لأن له معامَلين (two operands) . أما معامَلا + هنا فهما : integer1 , integer2 . ومعامَلا = هنا هما : sum ، وقيمة التعبير integer1 + integer2 . ويفضل ترك فراغ يمين ويسار أي مؤثِّر ثنائي ، وذلك كي يظهر واضحا وتسهل قراءة البرنامج . وأي حسابات في عبارة الإسناد يجب أن تكون في الطرف الأيمن . وأما وَضع أي حسابات في الطرف الأيسر فيُعَدُّ خطأً تركيبيا .

*) عبارة الطباعة :

```
printf ( "Sum is %d\n", sum ); /* print sum */
```

تستدعي الدالة **printf** لطباعة السلسلة الحرفية **Sum is** تليها القيمة العددية للمتغير **sum** على الشاشة . والدالة **printf** هنا لها وسيطان : الأول "**Sum is %d\n**" والثاني : **sum** . أما الأول فهو سلسلة التحكم في الصيغة ، وتحتوي على بعض الرموز الحرفية (literal characters) المطلوب ظهورها ، ومحدِّد التحويل d % الذي يشير إلى أن عددا صحيحا سيُطبع . وأما الثاني فيحدد القيمة التي ستُطبع . ولاحظ أن محدد التحويل لعدد صحيح هو نفسه في كل من الدالتين scanf , printf . وهذا هو الحال بالنسبة لمعظم أنواع البيانات في لغة C .

ويمكن أيضا إجراء حسابات داخل عبارات printf . فيمكننا مثلا جمع العبارتين السابقتين (عبارة الإسناد وعبارة الطباعة) في عبارة واحدة هكذا :

```
printf ( "Sum is %d\n",   integer1 + integer2 ) ;
```

*) السطر

```
return 0; /* indicate that program ended successfully */
```

يمرر القيمة 0 إلى وسط نظام التشغيل (operating system environment) الذي يتم فيه تنفيذ البرنامج . وهذا يشير إلى نظام التشغيل أن البرنامج قد تم تنفيذه بنجاح .

*) القوس الأيمن } في آخر سطر يدل على وصولنا إلى نهاية الدالة main .

مفاهيم الذاكرة Memory Concepts

أسماء المتغيرات مثل integer1, integer2, sum تقابل فعليا مواقع في ذاكرة الحاسوب . وكل متغير له اسم ، ونوع ، وقيمة . فمثلا في برنامج المثال السابق (مثال ٢-٤) عندما يتم تنفيذ العبارة

```
scanf ( "%d" , &integer1 ) ; /* read an integer */
```

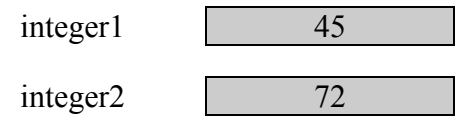
فإن القيمة التي يطبعها / يُدخِلها المستخدِم توضع في موقع الذاكرة الذي أُسند إليه الاسم integer1 . فإذا فرضنا أن المستخدِم قد أدخل العدد 45 كقيمة integer1 ، فإن الحاسوب يَضَع 45 في الموقع integer1 كما هو مبين في الشكل التالي :

وعندما توضَع قيمة في موقعٍ ما بالذاكرة فإن هذه القيمة تحل محل (replaces) القيمة السابقة الموجودة بهذا الموقع . ونظرا لأن هذه المعلومات السابقة تُمحَي / تُدَمَّر (destroyed) ، فإن عملية قراءة معلومات وإدخالها في موقع بالذاكرة يطلق عليها: " عملية قراءة وإدخال مدمِّرة " (destructive read – in process)

وعند تنفيذ العبارة

```
scanf ( "%d" , &integer2 ) ; /* read an integer */
```

في برنامج المثال السابق إذا فرضنا أن المستخدِم قد أدخل القيمة 72 ، فإن هذه القيمة توضع في الموقع integer2 ، وتظهر الذاكرة بالشكل التالي ، مع ملاحظة أن الموقعين المبينين بالشكل ليسا بالضرورة متجاورين في الذاكرة .



وبمجرد أن يحصل البرنامج على قيمتين للمتغيرين integer1 , integer2 فإنه يقوم بجمع هاتين القيمتين ووضع المجموع في المتغير sum .

والعبارة :

```
sum = integer1 +  integer2; /* assign total to sum * /
```

التي تقوم بعملية الجمع تشمل أيضا " عملية قراءة وإدخال مدمِّرة " والتي تحدث عندما توضع القيمة المحسوبة لمجموع integer1, integer2 في الموقع sum مدمِّرة بذلك أي قيمة قد تكون موجودة فعلا في sum . وبعد حساب sum تظهر الذاكرة بالشكل التالي :

integer1	45
integer2	72
sum	117

ولاحظ أن قيمتي integer1, integer2 تظهران بالضبط كما كانتا قبل استخدامهما في حساب قيمة sum . أي أن هاتين القيمتين قد تم استخدامهما ولكن لم تدمَّرا عندما قام الحاسوب بإجراء عملية الحساب هذه . أي أنه عندما تُقرأ قيمة من موقع بالذاكرة (read out of a memory location) ، فإن هذه العملية يُطلق عليها " عملية قراءة وإخراج غير مدمِّرة " (non destructive read – out process) .

العمليات الحسابية في لغة C (Arithmetic in C)

الجدول التالي يلخص المعامِلات / المؤثِّرات الحسابية في لغة C (C arithmetic operators)

العملية في لغة C	المؤثِّر الحسابي	التعبير الجبري	التعبير بلغة C
C Operation	Arithmetic operator	Algebraic expression	C expression
الجمع Addition	+	f+7	f + 7
الطرح Subtraction	-	p-c	p - c
الضرب Multiplication	*	bm	b * m
القسمة Division	/	x / y أو $\frac{x}{y}$ أو x ÷ y	x / y
الباقي Remainder	%	r mod s	r % s

جدول ٢ –٢

المؤثِّرات الحسابية في لغة C

C Arithmetic Operators

*) علامة النجمة (*) (asterisk) تعني الضرب . وفي الجبر إذا أردنا ضرب القيمتين a , b فيمكننا ببساطة وضع اسمي هذين المتغيرين (المكون أي منهما من حرف واحد) بجانب بعضيهما البعض هكذا : ab . ولكن في لغة C إذا فعلنا ذلك فإن ab سيفسَّر على أنه اسم متغير واحد [اسم تعريفي (identifier)] مكون من حرفين (two – letter name) . ولذلك فإن لغة C تحتم استخدام المؤثِّر *صراحة (explicitly) هكذا : a * b .

*) علامة الشرطة المائلة (/) (slash) تعني القسمة . وفي حالة قسمة عدد صحيح على عدد صحيح (integer division) فإن الناتج يكون عددا صحيحا هو خارج القسمة الصحيح . فمثلا التعبير 11/4 يعطي القيمة 2 ، والتعبير 17/5 يعطي القيمة 3 .

*) علامة النسبة المئوية (%) (percent sign) ترمز إلى " مؤثِّر الباقي " (remainder operator) ، والذي يعطي الباقي الصحيح بعد عملية القسمة الصحيحة (integer division) . أي أن التعبير x % y يعطي الباقي الصحيح بعد قسمة x على y . فمثلا 11 % 4 يعطي 3 ، بينما 17 % 5 يعطي 2 . ويلاحظ أن مؤثر الباقي لا يُستخدم إلا مع الأعداد الصحيحة فقط . أي أن كلاً من x , y في التعبير x % y يجب أن يكون عددا صحيحا .

*) جميع المؤثرات الحسابية في الجدول السابق هي مؤثرات ثنائية (binary operators) فمثلا التعبير 7 + 3 يحتوي على المعامِل / المؤثر الثنائي + والمعامَلان 7 , 3 .

*) أي محاولة للقسمة على الصفر تكون عادة غير معرَّفة في نظم الحاسوب ، وتؤدي عموما إلى خطأ قاتل / مميت (fatal error) ، ويُقصد به خطأ يؤدي إلى إنهاء (terminating) البرنامج فورا دون أن يؤدي وظيفته بنجاح . وأما الأخطاء غير القاتلة فهي تسمح للبرامج بالاستمرار في تشغيلها إلى نهايتها ، ولكن غالبا مع الوصول إلى نتائج خاطئة .

قواعد الأولوية بالنسبة للمؤثرات Rules of Operator Precedence /Priority

يتم حساب قيم التعابير الحسابية باتباع خطوات متتابعة بناءً على القواعد التالية :

١ – الأقواس (parentheses) لتجميع الحدود (grouping terms) لها الأولوية الأولى ، فمثلا التعبير a * (b + c) يشير إلى إجراء عملية الجمع قبل عملية الضرب في لغة C ، كما

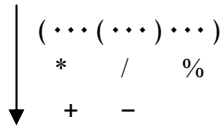
هو الحال بالنسبة للتعبير الجبر ي (a (b + c) algebraic expression) . وإن كانت هناك أقواس خارجية وأقواس داخلية فيتم أولا إجراء العمليات الحسابية بين الأقواس الداخلية .

٢ - عمليات الضرب (*) والقسمة (/) وحساب الباقي (%) لها الأولوية الثانية . وإذا احتوى التعبير المطلوب حساب قيمته على عدة عمليات من بين هذه العمليات (% , / , *) ، فإن حساب القيم المتتابعة يتجه من اليسار إلى اليمين (بالنسبة للعمليات المذكورة في التعبير المعطَي) . ويقال إن عمليات الضرب والقسمة وحساب الباقي لها مستوى الأولوية نفسه (same level of precedence) .

٣ - عمليات الجمع (+) والطرح (-) لها الأولوية الثالثة . وإذا احتوى التعبير المطلوب حساب قيمته على عدة عمليات جمع وطرح ، فإن حساب القيم المتتابعة يتجه من اليسار إلى اليمين . ويقال إن عمليتي الجمع والطرح لهما مستوى الأولوية نفسه ، والذي هو أقل من مستوى أولوية عمليات الضرب والقسمة وحساب الباقي .

ويلاحظ أننا حين نقول إن حساب القيم المتتابعة يتجه من اليسار إلى اليمين ، فإننا نشير إلى تجميع المؤثرات (associativity of the operators) .

الشكل التالي يلخص قواعد الأولوية المذكورة سابقا :



شكل ٢-١

قواعد الأولوية

حيث يتجه السهم من الأعلى أولوية إلى الأقل أولوية .

مثال ٢-٥ : اكتب تعبيرا بلغة C يكافئ كلا من التعابير الجبرية التالية :

أ) متوسط (mean / average) ثلاثة حدود :

$$m = \frac{a+b+c}{3}$$

ب) معادلة خط مستقيم (equation of a straight line)

$$y = m\ x + b$$

ج) z = pr % q + w / x - y

الحل : أ) m = (a + b + c) / 3 ;

نلاحظ هنا أنه من الضروري استخدام القوسين لأن أولوية القسمة أعلى من أولوية الجمع ، فبدون القوسين نحصل على التعبير a + b + c / 3 والذي يكافئ التعبير الجبري الخاطئ

$$a + b + \frac{c}{3}$$

ب) y = m * x + b ;

ونلاحظ هنا أننا لا نحتاج لأي أقواس ، وذلك لأن عملية الضرب ستُجري أولا ، لأن أولوية الضرب أعلى من أولوية الجمع .

ج) z = p * r % q + w / x - y;

6 1 2 4 3 5

الأعداد المحاطة بدوائر تحت العبارة تشير إلى الترتيب الذي ستجري به العمليات الحسابية : فالضرب وحساب الباقي والقسمة تُجري أولا من اليسار إلى اليمين . ثم تُجري بعد ذلك عمليتا الجمع والطرح من اليسار إلى اليمين . ثم تُسند القيمة المحسوبة النهائية إلى المتغير z .

مثال ٢-٦ : أ) اكتب تعبيرا بلغة C يكافئ حدودية الدرجة الثانية (second – degree polynomial) :

$y = a\ x^2 + b\ x + c$

واذكر الترتيب الذي تجري به العمليات الحسابية .

ب) نفرض أن a = 2 , b = 3 , c = 7 , x = 5

وضح كيفية إيجاد قيمة حدودية الدرجة الثانية المذكورة في أ) .

الحل : أ)

```
y  =  a  *  x  *  x  +  b  *  x  +  c;
      6     1     2     4     3     5
```

الأعداد المحاطة بدوائر تحت العبارة تشير إلى ترتيب إجراء العمليات الحسابية ، ثم تخزين القيمة النهائية للحدودية في y . ويلاحظ أنه لا يوجد مؤثِّر حسابي لعملية الرفع لأُسٍّ في لغة C ، ولذلك مثَّلنا x^2 بالتعبير x * x . ومكتبة C القياسية تحتوي على الدالة pow (وهي اختصار لكلمة power) لإجراء عملية الرفع لأُس ، ولكننا سنُرجئ الحديث عنها إلى الفصل الرابع وذلك لتوضيح ما تتطلبه من أنواع البيانات (data types) .

ب)

```
y = 2  *  5  *  5  +  3  *  5  +  7 ;
  =    10     *  5  +  3  *  5  +  7 ;
  =         50     +  3  *  5  +  7 ;
  =         50     +     15     +  7 ;
  =                65           +  7 ;
  =                             72  ;
```

اتخاذ القرار : مؤثرا التساوي والمؤثرات العلاقية

Decision Making : Equality and Relational Operators

تقوم عبارات C التنفيذية (executable statements) إما بتنفيذ أفعال (actions) [كإجراء حسابات أو إدخال بيانات أو طباعة بيانات] ، أو اتخاذ قرارات (making decisions) [وسنرى حالاً بإذن الله أمثلة لذلك] . فمثلا قد نختبر ما إذا كانت درجة طالب أكبر من أو تساوي 60 أم لا ، وفي حالة تحقق هذا الشرط فإننا نقرر طبع الرسالة " Congratulations ! You passed. " . وفي الجزء المتبقي من هذا الفصل نتناول صيغة بسيطة لعبارة **if** التي تسمح للبرنامج أن يتخذ قرارا بناءً على تحقُّق / صِدق (truth) أو عدم تحقق / خطأ (falsity) عبارة (statement) عن حقيقة (fact) يُطلَق عليها " شرط " (condition) .

فإذا تَحقَّق الشرط (أي كان صحيحا / صادقا true) فإن العبارة الموجودة في جسم عبارة **if** تُنَفَّذ . أما إذا لم يتحقق الشرط (أي كان خاطئا false) فإن تلك العبارة لا تنفَّذ . وسواء نُفِّذت تلك العبارة أم لم تُنفذ ، فبعد الانتهاء من عبارة if يبدأ تنفيذ العبارة التالية بعد عبارة if .

ويتم تكوين الشروط في عبارات if باستخدام مؤثري التساوي والمؤثرات العلاقية ، والملخصة في الجدول التالي :

المؤثرات الجبرية	المؤثرات بلغة C	مثال للشرط بلغة C
مؤثرا التساوي Equality operators		
=	==	x == y
≠	!=	x != y
المؤثرات العلاقية Relational operators		
>	>	x > y
<	<	x < y
≥	>=	x >= y
≤	<=	x <= y

جدول ٢-٣

مؤثرا التساوي والمؤثرات العلاقية

وجميع المؤثرات العلاقية (=> , =< , > , <) لها مستوى الأولوية نفسه ، ويتم تجميع (associating) هذه المؤثرات من اليسار إلى اليمين . وأما مؤثرا التساوي (!= , ==) فلهما مستوى أولوية أقل من مستوى أولوية المؤثرات العلاقية ، ويتم تجميعهما أيضا من اليسار إلى اليمين .

ملاحظات : ١) في لغة C الشرط (condition) قد يكون فعليا أي تعبير يولِّد (generates) صفرا 0 [وهذا يقابل أن الشرط : خاطئ (false)] أو قيمة غير صفرية (nonzero value) [وهذا يقابل أن الشرط : صحيح (true)] . وسنرى بإذن الله تعالى فيما بعد تطبيقات عديدة لذلك .

٢) تَركُ فراغ بين رَمزَي أي مؤثر من المؤثرات : => , =< , >= , =! , == يؤدي إلى خطأ تركيبي . وكذلك ينشأ خطأ تركيبي إذا عُكِس ترتيب رَمزَي أي من المؤثرات : =< , >= , =! فَكتِبا هكذا : <= , => , != على الترتيب .

٣) يجب الانتباه إلى أن مؤثر التساوي == يختلف عن مؤثر الإسناد = (assignment operator) ، فيلزم عدم الالتباس بينهما ، أي عدم استخدام أحدهما موضع الآخر . وكما سنرى بعد قليل بإذن الله فقد لا يؤدي هذا الالتباس بالضرورة إلى خطأ تركيبي يسهل التعرُّف عليه ، ولكنه قد يؤدي إلى أخطاء منطقية كبيرة .

٤) من الخطأ وَضع فاصلة منقوطة بعد القوس الأيمن بعد شرط عبارة **if** .

مثال ٢-٧ :

البرنامج التالي يستخدم عدة عبارات **if** للمقارنة بين عددين يُدخِلهما المستخدم , وإذا تحقق الشرط المذكور في أي من هذه العبارات فإن عبارة الطباعة **printf** المقابلة لعبارة **if** هذه يتم تنفيذها .

```
/* Example 2-7
Using if statements, relational
   operators, and equality operators */
#include <stdio.h>

/* function main begins program execution */
int main()
{

   int num1; /* first number to be read from user  */
   int num2; /* second number to be read from user */

   printf( "Enter two integers, and I will tell you\n" );
   printf( "the relationships they satisfy: " );

   scanf( "%d%d", &num1, &num2 ); /* read two integers */

   if ( num1 == num2 ) {
      printf( "%d is equal to %d\n", num1, num2 );
   } /* end if */
```

```
   if ( num1 != num2 ) {
      printf( "%d is not equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 < num2 ) {
      printf( "%d is less than %d\n", num1, num2 );
  } /* end if */

   if ( num1 > num2 ) {
      printf( "%d is greater than %d\n", num1, num2 );
   } /* end if */

   if ( num1 <= num2 ) {
      printf( "%d is less than or equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 >= num2 ) {
      printf( "%d is greater than or equal to %d\n", num1, num2 );
   } /* end if */

   return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is not equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

نلاحظ أن البرنامج يستخدم الدالة **scanf** لإدخال عددين وكل محدد تحويل (conversion specifier) له وسيط مقابل (corresponding argument) ستُخزَّن فيه قيمة . المحدِّد % الأول يحوِّل قيمة لتخزينها في المتغير num1 . والمحدِّد % الثاني يحوِّل قيمة لتخزينها في المتغير num2 . وكل عبارة **if** في هذا البرنامج لها عبارة واحدة فقط في جسمها (body) . وسنرى بإذن الله في الفصل القادم كيف نكتب عبارة **if** تحتوي على عدة عبارات في جسمها . ومن الخطأ وضع فواصل (commas) – عندما لا تكون هناك حاجة لأي منها – بين محدِّدي التحويل في سلسلة التحكم في الصيغة (format control string) في عبارة **scanf** .

أولوية المؤثرات Operators Precedence

الشكل التالي يوضح أولوية المؤثرات التي تم التعريف بها في هذا الفصل . والسهم في الشكل يتجه من الأعلى أولوية إلى الأقل أولوية . ولاحظ أن علامة التساوي = تُعَدُّ أيضا من المؤثرات ، ويطلق عليها " مؤثر الإسناد " (assignment operator) . وجميع المؤثرات – باستثناء مؤثر الإسناد – تتجمَّع (associate) من اليسار إلى اليمين ، بينما مؤثر الإسناد يتجمع من اليمين إلى اليسار .

المؤثرات Operators	التجميع Associatvity
* / %	من اليسار إلى اليمين
+ -	من اليسار إلى اليمين
< <= > >=	من اليسار إلى اليمين
== !=	من اليسار إلى اليمين
=	من اليمين إلى اليسار

شكل ٢-٢

أولوية المؤثرات

الكلمات المحجوزة / الكلمات المفاتيح Reserved Words / Keywords

هناك بعض الكلمات التي لها معاني خاصة بالنسبة للبرنامج المترجِم في لغة C (C compiler) ، ولا يجوز للمبرمج أن يستخدم أياً من هذه الكلمات كأسماء تعريفية

(identifiers) كأسماء متغيرات مثلا . ومن هذه الكلمات التي استخدمناها لـلآن : int , return , if . والجدول التالي يعطي هذه الكلمات والتي يطلق عليها " الكلمات المحجوزة" أو " الكلمات المفاتيح " .

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

جدول ٢-٤

الكلمات المحجوزة / الكلمات المفاتيح في لغة C

تمرينات رقم ٢

٢-١) اذكر صحة أو خطأ (T / F) كل من العبارات التالية ، وإن كانت العبارة خاطئة فوضح السبب .

أ) عندما نستدعي الدالة **printf** فإنها تبدأ الطباعة دائما عند بداية سطر جديد .

ب) التعليقات تجعل الحاسوب يطبع النص المحصور بين / * و */ على الشاشة عند تنفيذ البرنامج .

ج) متتابعة الهروب \n عندما تُستخدَم في سلسلة تحكم في الصيغة في عبارة **printf** ، فإنها تجعل مؤشر الشاشة ينتقل إلى بداية السطر التالي على الشاشة .

د) جميع المتغيرات يجب أن تُعرَّف قبل استخدامها .

هـ) جميع المتغيرات يجب أن تُعطَي نوعاً عند تعريفها .

و) لغة C تعتبر أن المتغيرين **number , NuMbEr** متطابقان .

ز) يمكن للتعريفات أن تظهر في أي موضع في جسم أي دالة .

ح) مؤثر الباقي ٪ يمكن أن يستخدم فقط مع المعامَلات الصحيحة .

ط) أي برنامج يطبع ثلاثة سطور من المخرجات يجب أن يحتوي على ثلاث عبارات **printf** .

٢-٢) اكتب عبارة C واحدة لتنفيذ كل مما يلي :

أ) عرِّف المتغيرات **number , q76354 , thisVariable , c** لتكون من النوع **int**.

ب) اطلب من المستخدِم أن يُدخل عددا صحيحا . واجعل رسالة الحث / الطلب (prompting message) تنتهي بالنقطتين الرأسيتين (رمز الوقف الاستدراكي) (:) (colon) يتبعهما فراغ واحد (a space) ، واترك مؤشر الشاشة في الموضع بعد هذا الفراغ .

ج) اقرأ عددا صحيحا من لوحة المفاتيح ، وقم بتخزين القيمة المُدخلة في المتغير الصحيح **a** .

د) إذا كان **number** لا يساوي 7 ، اطبع الرسالة "The variable number is not equal to7."

هـ) اطبع الرسالة " This is a C program." على سطر واحد .

و) اطبع الرسالة " This is a C program." على سطرين ، بحيث ينتهي السطر الأول عند C .

ز) اطبع الرسالة " This is a C program. " بحيث تكون كل كلمة على سطر مستقل.

ح) اطبع الرسالة " This is a C program. " بحيث تُفصل كل كلمة عن أخرى بجدولة أفقية \t (tab) .

٢-٣) اكتب عبارة (أو تعليقا) لتنفيذ ما يلي :

أ) بيان (تعليق) أن البرنامج سيحسب حاصل ضرب ثلاثة أعداد صحيحة .

ب) تعريف المتغيرات **result , z , y , x** لتكون من النوع **int** .

ج) حث المستخدِم لإدخال ثلاثة أعداد صحيحة .

د) قراءة ثلاثة أعداد صحيحة من لوحة المفاتيح وتخزينها في المتغيرات **x , y , z** .

هـ) حساب حاصل ضرب ثلاثة أعداد صحيحة تحتويها المتغيرات **x , y , z** ، وأسنِد النتيجة إلى المتغير **result** .

و) اطبع الرسالة " **The product is** " تتبعها قيمة المتغير الصحيح **result** .

٢-٤) باستخدام العبارات التي كتبتها في حل التمرين السابق (٢-٣) اكتب برنامجا كاملا يحسب حاصل ضرب ثلاثة أعداد صحيحة .

٢-٥) اكتشف وصحح أي أخطاء في كل من العبارات التالية :

أ)
```
printf( "The value is %d\n" , &number );
```

ب)
```
scanf( "%d%d" , &number1 , number2 ) ;
```

ج)
```
if ( c < 7 );
   printf( "C is less than 7\n" );
```

د)
```
if ( c => 7 )
   printf( "C is equal to or less than 7\n" );
```

٢-٦) اكتشف وصحح أي أخطاء في كل من العبارات التالية (ملاحظة : قد يكون هناك أكثر من خطأ واحد في أي عبارة) :

أ)
```
scanf( "d" , value );
```

ب)
```
printf( " The product of %d and %d is %d"\n, x, y );
```

ج)
```
firstNumber + secondNumber = sumOfNumbers
```

د)
```
if ( number => largest )
```

largest = = number;

هـ) * / program to determine the largest of three integers /*

و) Scanf("%d" , anInteger);

ز) printf("Remainder of %d divided by %d is\n", x , y , x % y);

ح) if (x = y);
printf(%d is equal to %d\n" , x, y)

ط) printf(" The sum is %d\n," x + y);

ي) Printf("The value you entered is: %d\n, &value);

٢-٧) اذكر صحة أو خطأ (T / F) كل مما يلي ، وإن كان خاطئا فاذكر السبب :

أ) يتم تقييم (evaluating) مؤثرات (**operator**s) **C** من اليسار إلى اليمين .

ب) الأسماء التالية تعد أسماء صالحة (valid) لمتغيرات :

under bar_ , m928134 , t5 , j7 , her_ sales , his_account_total,
a , b , c , z , z2

ج) الأسماء التالية جميعها غير صالحة (invalid) كأسماء متغيرات :

3g , 87 , 67h2 , h22 , 2h

٢-٨) نفرض أن y = 3, x = 2 . ماذا يُطبع – إن كان هناك شيء يُطبع – عند تنفيذ كل من العبارات التالية ؟ وفي حالة عدم طباعة أي شيء ، أجب : لا شيء .

أ) printf("%d", x);

ب) printf("%d", x + x);

ج) printf("x=");

د) printf("x=%d", x);

هـ) printf("%d = %d", x + y, y + x);

و) z = x + y;

ز) scanf("%d%d", &x, &y);

ح) /* printf("x + y = %d", x + y); * /

ط) printf("\n");

٢-٩) أي عبارات C التالية تحتوي على متغيرات تشملها عملية قراءة وإدخال مدمِّرة (destructive read-in) ؟

أ) scanf("%d%d%d%d%d", &b, &c, &d, &e, &f);

ب) p = i + j + k +7;

ج) printf("Destructive read-in");

د) printf("a = 5");

٢-١٠) وضِّح الترتيب الذي يتم به إيجاد قيم (evaluation) المؤثرات في كل من عبارات C التالية ، وأوجد قيمة x بعد تنفيذ كل عبارة .

أ) x = 7 + 3 * 6 / 2 – 1;

ب) x = 2 % 2 + 2 * 2 – 2 / 2;

ج) x = (3 * 9 * (3 + (9 * 3 / (3))));

٢-١١) اكتب برنامجا يطلب من المستخدِم إدخال عددين ، ثم يقوم البرنامج بالحصول على هذين العددين من المستخدِم ، وطباعة مجموع العددين ، وحاصل ضربهما ، والفارق بينهما ، وخارج القسمة والباقي عند قسمة العدد الأول على الثاني .

٢-١٢) اكتب برنامجا يطبع الأعداد من 1 إلى 4 على السطر نفسه ، وذلك باتباع كل من الطرق التالية :

أ) استخدام عبارة **printf** واحدة بدون محددات تحويل .

ب) استخدام عبارة **printf** واحدة ذات 4 محددات تحويل .

ج) استخدام أربع عبارات **printf** .

٢-١٣) اكتب برنامجا يطلب من المستخدِم إدخال عددين ، ثم يقوم البرنامج باستقبال هذين العددين من المستخدِم ، وطباعة العدد الأكبر باستقبال هذين العددين من المستخدِم ، وطباعة العدد الأكبر تليه الكلمات " is larger. " وإذا كان العددان متساويين فإن البرنامج يطبع الرسالة " These numbers are equal. " .

[ملاحظة : استخدم صيغة عبارة **if** البسيطة التي ذكرناها في هذا الفصل] .

٢-١٤) اكتب برنامجا يُدخِل ثلاثة أعداد صحيحة مختلفة من لوحة المفاتيح ، ثم يطبع مجموعها ، ومتوسطها ، وحاصل ضربها ، وأصغرها ، وأكبرها .

[ملاحظة : استخدم صيغة عبارة **if** البسيطة] . الحوار على الشاشة (screen dialogue) يبدو كما يلي :

```
Input three different integers:  13  27
14
Sum is  54
Average is  18
Product is  4914
Smallest is  13
Largest is  27
```

٢-١٥) اكتب برنامجا يقرأ قيمة نصف قطر دائرة ويطبع قيمة كل من قطر الدائرة ، ومحيطها ، ومساحتها . استخدم قيمة Π الثابتة 3.14159 . قم بإجراء حسابات هذه القيم – المطلوب طباعتها – داخل عبارة / عبارات الطباعة **printf** ، واستخدم محدِّد التحويل f% . [ملاحظة : في هذا الفصل درسنا فقط الثوابت الصحيحة والمتغيرات الصحيحة . وفي الفصل القادم سندرس بإذن الله الأعداد ذوات النقطة العائمة – floating) (point numbers ، وهي القيم التي يمكن أن تحتوي على العلامة العشرية ( ) . decimal point)

٢-١٦) ماذا تطبع العبارة التالية ؟

```
printf( "*\n**\n***\n****\n*****\n" );
```

٢-١٧) اكتب برنامجا يقرأ خمسة أعداد صحيحة ، ثم يحدد ويطبع أكبرها وأصغرها . استخدم فقط طرق البرمجة التي درسناها في هذا الفصل .

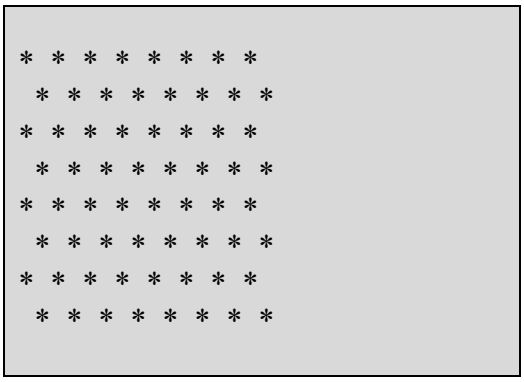
٢-١٨) اكتب برنامجا يقرأ عددا صحيحا ، ثم يحدد ويطبع ما إذا كان هذا العدد فرديا (odd) أم زوجيا (even) . [إرشاد : استخدم مؤثر الباقي . أيُّ عدد زوجي هو أحد مضاعفات 2 (a multiple of 2) . وأي قيمة من مضاعفات 2 إذا قُسمت على 2 فإنها تترك باقيا يساوي صفرا 0] .

٢-١٩) اكتب برنامجا يقرأ عددين صحيحين ، ويحدد ويطبع ما إذا كان العدد الأول هو أحد مضاعفات العدد الثاني . [إرشاد : استخدم مؤثر الباقي] .

٢-٢٠) اكتب برنامجا يعرض الشكل التالي [نموذج لوحة لعبة الداما checkerboard) (pattern] :

أ) باستخدام ثمان عبارات **printf** .

ب) باستخدام أقل عدد ممكن من عبارات **printf** .



٢-٢١) درسنا في هذا الفصل الأعداد الصحيحة والنوع int . ويمكن للغة C أن تمثِّل أيضا الحروف الكبيرة (uppercase letters) والحروف الصغيرة (lowercase letters) وعددا كبيرا من الرموز الخاصة (special symbols) . وتستخدم لغة C داخليا (internally) أعدادا صحيحة صغيرة لتمثيل كل رمز (character) من هذه الرموز المختلفة . ومجموعة الرموز التي يستخدمها الحاسوب والأعداد الصحيحة المقابلة التي تمثل هذه الرموز يُطلَق عليها " مجموعة رموز هذا الحاسوب " (that computer's character set) . ويمكنك طباعة العدد الصحيح المكافئ (integer equivalent) للحرف الكبير **A** مثلا بتنفيذ العبارة

```
printf( "%d", 'A' );
```

اكتب برنامجا بلغة C يطبع الأعداد الصحيحة المكافئة لبعض الحروف الكبيرة والحروف الصغيرة والأرقام والرموز الخاصة مثل :

```
A  B  C  a  b  c  0  1  2  $  *  +  /
```

ورمز الفراغ (blank character) .

٢-٢٢) اكتب برنامجا يقرأ عددا مكونا من خمسة أرقام (five – digit number) ، ثم يفصل العدد إلى أرقامه الخمسة ، ثم يطبع هذه الأرقام بحيث يترك ثلاثة فراغات بين أي رقمين متجاورين .

[إرشاد : استخدم عدة عمليات قسمة صحيحة (integer division) وحساب الباقي (remainder)] . مثلا إذا أدخل المستخدِم العدد 42139 ، فإن البرنامج يطبع

```
4    2    1    3    9
```

٢-٢٣) باستخدام الطرق التي درستها في هذا الفصل اكتب برنامجا يحسب مربعات ومكعبات الأعداد من 0 إلى 5 ، ويستخدم الجدولة الأفقية tabs لطباعة جدول القيم التالي :

```
number      square      cube
0           0           0
1           1           1
2           4           8
3           9           27
4           16          64
5           25          125
```

٢-٢٤) 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 ND , NS , A واستخدم عبارة تعليق في مطلع البرنامج لبيان وظيفته ، وعبارات تعليق توضيحية في ثناياه .

٢-٢٥) اكتب برنامجا يطبع الرسالة التالية :

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

٢-٢٦) 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 إعلانات البرنامج .

٢-٢٧) اكتب برنامجا بلغة C 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 ، وأجره الإضافي OVERTM ، وأجره الكلي TOTPAY ، حسب القوانين التالية :

$$PEGPAY = 40 \times WAGE$$
$$OVERTM = (DAYS - 40)(WAGE \times 1\tfrac{1}{2})$$
$$TOTPAY = REGPAY + OVERTM$$

(ج) طباعة النتائج ..

٢-٢٨) 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 N , Z .

٢-٢٩) اكتب برنامجا لقراءة قيمة كل من طول مستطيل وعرضه ثم حساب كل من محيطه ومساحته وطول قطره .

اطبع رسائل توضيحية كافية لتوضيح المخرجات .

٢-٣٠) 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 = $\frac{4}{3}\pi R^3$)

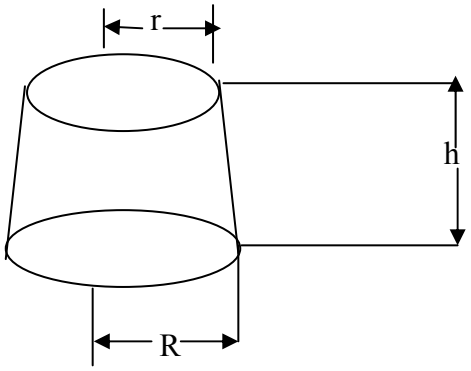
٢-٣١) تتحرك عربة بعجلة ثابتة a لمدة t ثانية. اكتب برنامجا لقراءة قيمة كل من a , t وحساب المسافة التي تحركتها العربة $d = \frac{1}{2} at^2$ ، والسرعة النهائية v = a.t مع كتابة عناوين مناسبة للنتائج .

٢-٣٢) حجم المجسَّم المخروطي الدائري القائم المبين بالشكل يعطى بالعلاقة :

$$V = \pi h/3\ (R^2 + rR + r^2)$$

اكتب برنامجا لقراءة قيمة كل من h , R , r وحساب الحجم V بفرض أن

π = 3.1415927



٢-٣٣) 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)

٢-٣٤) 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

٢-٣٥) اكتب برنامجا ليقرأ وزن جسم بالرطل (الباوند pound) ويحسب ويطبع الوزن بالكيلو جرام وكذلك بالجرام .
(إرشاد : ١ رطل يكافئ ٠,٤٥٣٥٩٢ كيلو جراما)

٢-٣٧) 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 .

٢-٣٧) 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 .

٢-٣٨ 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 .

٢-٣٩) 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 .

٢-٤٠) 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

$Time = distance / (velocity \times \cos(theta))$
$Height = velocity \times time - (g \times time^2) / 2$

٢-٤١) 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.

٢-٤٢) 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

بنَى التحكم

Control Structures

عادة يتم تنفيذ عبارات أي برنامج بالترتيب الذي كُتبت به هذه العبارات : عبارة تلو الأخرى . وهذا يطلَق عليه " تنفيذ تتابعي " (sequential execution) . وسندرس في هذا الفصل بإذن الله بعض عبارات لغة C التي تمكننا من تحديد العبارة التالية المطلوب تنفيذها والتي قد تختلف عن العبارة التالية في تتابع العبارات المكتوبة . وهذا الاختلاف يطلق عليه " انتقال التحكم " (transfer of control) .

وتحتوي لغة C على سبع بِنى / عبارات تحكم : التتابع (sequence) ، وثلاثة أنواع للاختيار (selection) ، وثلاثة أنواع للتكرار (repetition) . أما أنواع الاختيار فهي :

١) عبارة الاختيار **if** : وهذه العبارة إما تنفذ / تختار (performs/selects) أمراً / فعلاً (action) إذا تحقق شرط معين ، أي كان صحيحا / صادقا (true) ، أو تتخطاه (skips) أي لا تنفذه إذا لم يتحقق الشرط ، أي كان خاطئا (false) .

٢) عبارة الاختيار **if…else** : وهذه العبارة إما تنفذ أمراً ما إذا تحقق شرط معين ، أو تنفذ أمراً آخر إذا لم يتحقق الشرط .

٣) عبارة الاختيار **switch** : وهذه العبارة تنفذ واحدا من عدة أوامر / أفعال (actions) مختلفة بناءً على قيمة تعبير معين . وسندرس بإذن الله عبارة **switch** في الفصل القادم (الفصل الرابع) .

ويطلق على عبارة **if** : " عبارة اختيار مفرد " **(single – selection statement)** لأنها تختار أو تتجاهل (selects or ignores) أمرا مفردا . ويطلَق على عبارة **if…else** " عبارة اختيار مزدوج " **(double – selection statement)** لأن العبارة تختار بين فعلين مختلفين . وأما عبارة **switch** فيطلَق عليها عبارة اختيار متعدد " **(multiple – selection statement)**، لأن العبارة تختار بين عدة أفعال مختلفة .

وأما أنواع التكرار فهي :

١) عبارة **while** .

٢) عبارة **do..while** وهذه سندرسها في الفصل الرابع بإذن الله .

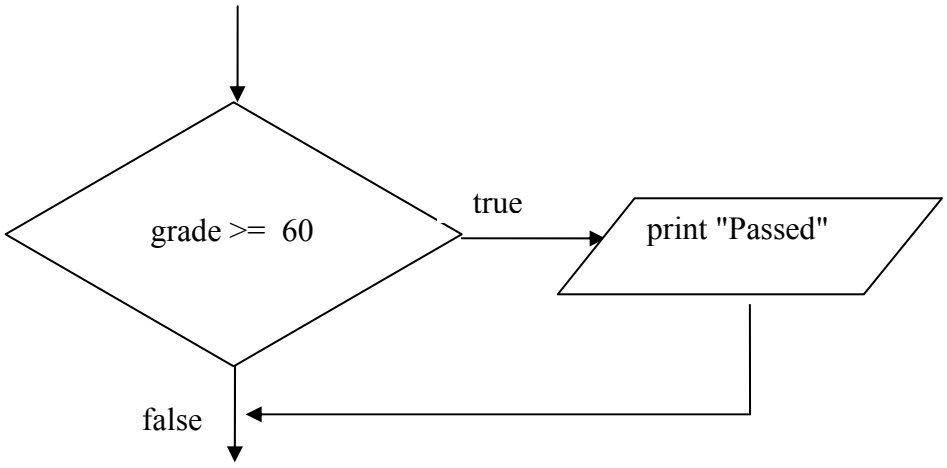
٣) عبارة **for** وهذه أيضا سندرسها في الفصل القادم (الرابع) بإذن الله .

عبارة الاختيار if

مثال ٣-١ : نفرض أن درجة النجاح في اختبار ما هي 60 . عبارة **if** التالية تختبر ما إذا كانت درجة طالب ما grade في هذا الاختبار قد بلغت أو تجاوزت 60 أم لا . فإذا تحقق هذا الشرط فإن العبارة تطبع الرسالة " Passed ". وإذا لم يتحقق فإنها لا تفعل شيئا . وفي أي من الحالتين فإن العبارة التالية التي تنفَّذ هي العبارة التي تلي عبارة **if**.

```
If ( grade >= 60 )
   printf ( "'Passed\n" );
```

وعبارة **if** هذه يمكن تمثيلها بخريطة سير العمليات التالية (شكل ٣-١) :



شكل ٣-١

ويلاحظ أن رمز المعيَّن (diamond symbol) وهو رمز اتخاذ القرار (decision symbol) يحتوي على تعبير (an expression) [كشرط (a condition) مثلا] يمكن أن يكون صحيحا أو خاطئا (true or false) . وقد رأينا في الفصل السابق أن اتخاذ القرار (decision) قد يكون مبنياً على شروط تحتوي على مؤثرات علاقية أو مؤثري التساوي .

وعموما فإن اتخاذ القرار قد يُبنى على أي تعبير ، بحيث أنه إذا كانت قيمة التعبير صفرا 0 فإننا نعتبر التعبير خاطئا (false) ، وإذا كانت قيمة التعبير أي قيمة أخرى غير صفرية (nonzero) فإننا نعتبر التعبير صحيحا (true).

عبارة الاختيار **if**...**else**

هذه العبارة تسمح للمبرمج بتنفيذ أوامر معينة في حالة ما إذا تحقق شرطٌ ما ، وتنفيذ أوامر أخرى في حالة عدم تحققه .

مثال ٣-٢ : في المثال السابق (مثال ٣-١) نفرض أننا أضفنا المطلوب التالي : إذا كانت درجة الطالب أقل من 60 فإن العبارة تقوم بطباعة الرسالة " Failed " . أي أن المطلوب من عبارة **if**...**else** أن تقوم بتنفيذ مايلي :

إذا كانت درجة الطالب أكبر من أو تساوي 60

فإن العبارة تطبع الرسالة " Passed "

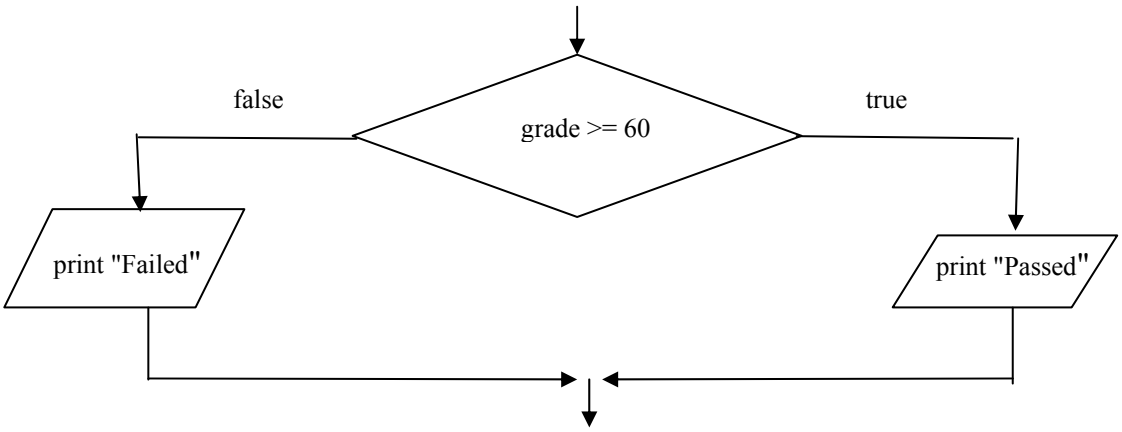
وما عدا ذلك

فإن العبارة تطبع الرسالة " Failed "

الحل :

```
if ( grade >= 60 )
   printf ( " Passed\n" );
else
   printf ( " Failed\n" );
```

وخريطة سير العمليات التالية (شكل ٣-٢) تمثل هذه العبارة وتوضحها :



شكل ٣ – ٢

المؤثر الشَّرطي : ? The conditional operator

هذا المؤثر يرتبط ارتباطا وثيقا بعبارة **if**...**else** . وهو المؤثر الثلاثي (ternary operator) الوحيد في لغة C . وله ثلاثة معامَلات (3 operands) تكوِّن مع المؤثر الشَّرطي تعبيرا شرطيا (conditional expression) :

- المعامَل الأول (first operand) : عبارة عن شرط .
- المعامَل الثاني (second operand) : عبارة عن قيمة التعبير الشَّرطي بأكمله (the value for the entire conditional expression) إذا كان الشرط صحيحا / صادقا (true) .
- المعامَل الثالث (third operand) : عبارة عن قيمة التعبير الشَّرطي بأكمله إذا كان الشَّرط خاطئا / كاذبا (false) .

مثال ٣-٣ : عبارة الطباعة (**printf** statement) :

```
printf ( " %s\n" , grade >= 60 ? "passed" : "Failed" );
```

تحتوي على تعبير شَرطي تكون قيمته (evaluates to) السلسلة الحرفية (string literal) "Passed" إذا كان الشرط grade >= 60 صحيحا (true) ، بينما تكون قيمته السلسلة الحرفية " Failed " إذا كان هذا الشرط خاطئا . وتحتوي سلسة التحكم في الصيغة (format control string) في عبارة الطباعة **printf** على تحديد التحويل ٪s (conversion specification) لطباعة سلسلة رموز (a character string) . وهكذا نرى أن عبارة **printf** السابقة تؤدي بطريقة مماثلة عمل عبارة **if**...**else** السابقة .

والقيم (values) التي تظهر في تعبير شَرطي (conditional expression) يمكن أن تكون أيضا أفعالا تُنفَّذ ، كما يوضِّح ذلك المثال التالي .

مثال ٣-٤ : التعبير الشَّرطي

```
grade >= 60 ? printf ("Passed\n" ) : printf ("Failed\n" );
```

يُقرأ هكذا :

"If grade is greater than or equal to 60 then printf ("Passed\n"), otherwise printf ("Failed\n");

ولـذلك فهـو أيضـا شـبيه بعبـارة **if**...**else** السـابقة . وسـنرى بـإذن الله مسـتقبلا أن المـؤثرات الشرطية يمكن أن تُستخدم في مواطن لا يمكننا فيها استخدام عبارات **if**...**else** .

عبارات **if**...**else** المتداخلة

(**Nested if** ...**else statements**)

تقـوم هـذه العبـارات باختبـار حـالات متعـددة (multiple cases) بوضـع عبـارات **if**...**else** داخل عبارات **if**...**else** .

مثال ٣–٥ : اكتب قطعة برنامج لطباعة تقرير الطالب الحرفي (letter grade) المقابل لدرجته grade بناءً على الجدول التالي :

grade	90 – 100	80 – 89	70 – 79	60 -69	0 -59
lerrer grade	A	B	C	D	F

أي أن عبارة الكـود الرمـزي (pseudocode statement) أو الخوارزمية (algorithm) المقابلة لهذه القطعة المطلوبة هي :

```
If student's grade is greater than or equal to 90
      Print "A"
else
      If student's grade is greater than or equal to 80
            Print "B"
       else
            If student's grade is greater than or equal to 70
                  Print "C"
            else
             If student's grade is greater than or equal to 60
                        Print "D"
             else
                         Print   "F"
```

الحل : الصيغة الأولى :

```
if ( grade >= 90 )
   printf( "A\n" );
else
   if ( grade >= 80 )
```

```
            printf( "B\n" );
      else
            if ( grade >= 70 )
               printf( "C\n" );
            else
                  if ( grade >= 60 )
                     printf( "D\n" );
                  else
                      printf( "F\n" );
```

نلاحظ أنه إذا كانت قيمة المتغير grade أكبر من أو تساوي 90 فإن الشروط الأربعة الأولى ستكون جميعها صحيحة (true) ، ولكن عبارة printf الموجودة بعد الشرط / الاختبار الأول هي فقط التي ستُنَفَّذ (وتطبع الحرف A) . فبعد تنفيذ عبارة printf هذه ، يتم تخطي (skipping) الجزء else من عبارة **else** ... **if** . ويُفَضِّل كثير من المبرمجين كتابة عبارة if السابقة (أي قطعة البرنامج السابقة) بالصيغة التالية :

الصيغة الثانية :

```
if ( grade >= 90 )
   printf( "A\n" );
else if ( grade >= 80 )
    printf( "B\n" );
else if ( grade >= 70 )
    printf( "C\n" );
else if ( grade >= 60 )
    printf( "D\n" );
else
    printf( "F\n" );
```

والتي يطلَق عليها أحيانا : عبارة **if** المتداخلة / القالبية / المركبة
(Compound / Block / Nested if statement)

وبالنسبة للبرنامج المترجِم (C compiler) فإن الصيغتين السابقتين للحل متكافئتان (equivalent) ، ولكن الصيغة الثانية تجد قبولا أكثر لأنها تتجنب الزحزحة (indentation) المتتالية للعبارات إلى اليمين مما يتسَبَّب في ترك حيِّز أصغر في السطور لكتابة العبارات والذي قد يؤدي بدوره إلى كتابة العبارة الواحدة على أكثر من سطر ، وهذا من شأنه أن يقلل من سهولة ووضوح قراءة البرنامج .

وعموما فإن عبارة الاختيار **if** تتوقع ظهور عبارة واحدة فقط في جسمها . فإذا أردنا إدراج عدة عبارات في جسم أي عبارة **if** فإننا نضع هذه المجموعة من العبارات بين القوسين : { , } (braces) . وأي مجموعة عبارات محصورة بين هذا الزوج من الأقواس : { , } يطلَق عليها : " عبارة مركبة " (compound statement) أو قالب (block) . وأي عبارة مركبة يمكن أن توضع في أي برنامج في موضع يمكن أن توضع فيه عبارة مفردة (بسيطة) (a single statement) .

مثال ٣-٦ :

قطعة البرنامج التالية تحتوي على عبارة مركبة (a compound statement) في الجزء else من عبارة **if** ... **else** :

```
if ( grade >= 60 )
   printf ( "Passed.\n" );
else  {
    printf ( "Failed.\n" );
    printf ( "You must take this course again.\n" );
                }
```

وفي هذه الحالة إذا كانت grade أصغر من 60 ، فإن البرنامج يقوم بتنفيذ عبارتي **printf** الاثنتين الموجودتين في جسم الجزء else ، أي يقوم بطباعة :

```
Failed.
You must take this course again.
```

ولاحظ أهمية وجود القوسين { , } اللذين يحيطان بعبارتي الجزء else ، فبدونهما ستصبح عبارة **printf** الثانية

```
printf ("You must take this course again.\n" );
```

خارج جسم الجزء **else** من عبارة **if** ، وبالتالي ستُنَفَّذ على أي حال ، سواء كانت درجة الطالب أقل من 60 أم لا ، وهذا بالطبع خطأ .

ملاحظة : وضْع فاصلة منقوطة (;) (semicolon) بعد شرط عبارة **if** يؤدي إلى :

- خطأ منطقي (logic error) في حالة عبارات **if** (البسيطة) للاختيار المفرد .
- خطأ تركيبي (syntax error) في حالة عبارات **else** ... **if** للاختيار المزدوج.

عبارة التكرار while

The while Repetition Statement

المقصود بعبارة التكرار : عبارة تسمح للمبرمج بتكرار تنفيذ أمر / فعل (action) معين طالما أن شرطا معينا لا يزال متحققا / صحيحا (true) ، فإذا أصبح الشرط خاطئا توقَّف تكرار تنفيذ هذا الأمر .

مثال ٣-٧ : أكتب قطعة برنامج لإيجاد أول قوة من قوى 2 (first power of 2) أكبر من 1000 .

الحل :

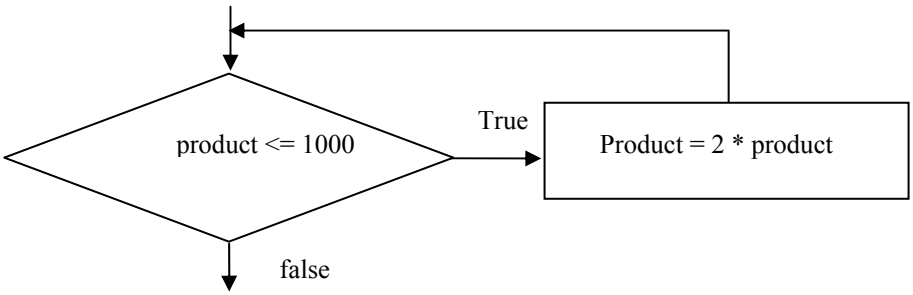
```
product = 2;
while ( product <= 1000 )
  product = 2 * product;
```

نبدأ بإعطاء قيمة ابتدائية تساوي 2 لمتغير صحيح (integer variable) يُدعى **product** . ونكرر عملية ضرب هذا المتغير في 2 طالما أن قيمته لا تزال أصغر من 1000 . فإذا أصبح هذا الشرط (product <= 1000) خاطئا، أي أصبح (product > 1000) فإن هذا التكرار يتوقف / ينتهي (repetition terminates) ، ويتوقف تنفيذ عبارة **while** ، وعندها سيحتوي **product** على القيمة المطلوبة . معنى هذا أن المتغير **product** سيحتوي على القيم المتتابعة التالية :

2 , 4 , 8 , 16 , 32 , 64 , 128 , 256 , 512 , 1024

وحينما يحتوي product على القيمة الأخيرة 1024 يصبح الشرط (product <= 1000) خاطئا ، ويتوقف التكرار ، وعندها يحتوي product على القيمة المطلوبة : 1024.

ويمكن توضيح تنفيذ عبارة التكرار **while** السابقة بخريطة سير العمليات التالية (شكل ٣-٣) :



شكل ٣ - ٣

نلاحظ أنه عندما يصبح شرط التكرار (false)فإنه يتم الخروج من عروة / عبارة (statement / loop) **while** . وينتقل التحكم إلى العبارة التالية في البرنامج (أي التي تلي عبارة **while**) .

والعبارة أو العبارات التي تحتوي عليها عبارة التكرار **while** تكوِّن (constitute) ما يسمى بجسم (body) عبارة **while** . وجسم عبارة **while** هذا قد يكون عبارة مفردة (single statement) أو عبارة مركبة (compound statement) .

ملاحظة ١ : من الخطأ ألا يشتمل جسم عبارة / عروة **while** على أي عبارات تؤدي بعد حين إلى ما يجعل شرط عبارة **while** خاطئا ، حتى يتم الخروج من العروة . ومثل هذه البنية (repetition structure) للتكرار دون توقف يطلق عليها " عروة لا نهائية " (infinite loop) وتعد خطأ .

ملاحظة ٢ : من الخطأ كتابة كلمة **while** هكذا **While** أي باستخدام الحرف الكبير W . وعموما جميع الكلمات المفاتيح المحجوزة (reserved keywords) في لغة C تكتب كلها بحروف صغيرة ، مثل : while , if , else .

مثال ٣-٨ : (تكرار محكوم بعدّاد) **(Counter – Controlled Repetition)**

اكتب برنامجا لقراءة درجات 10 طلاب في صفٍّ ما في أحد الاختبارات ، حيث درجة أي طالب عبارة عن عدد صحيح (integer) في المدى من 0 إلى 100 . ثم يقوم البرنامج بحساب وطباعة الدرجة المتوسطة لطلاب هذا الصف في ذلك الاختبار .

الحل : الدرجة المتوسطة = مجموع درجات الطلاب مقسوما على عددهم (10) . سنستخدم أسماء المتغيرات التالية :

grade : درجة أي طالب .

total : مجموع درجات الطلاب .

counter : عداد يشير إلى ترتيب الدرجة التي سنُدخلها .

average : الدرجة المتوسطة .

سنُدخل درجات الطلاب درجة درجة . وبعد إدخال أي درجة " grade " نضيفها إلى المجموع " total " ونزيد العداد " counter " بواحد . ونكرر هذه الخطوات الثلاث [إدخال درجة – إضافتها إلى المجموع – زيادة العداد بواحد] إلى أن تزيد قيمة العداد عن 10 فيتوقف التكرار ، وعندها نقسم مجموع الدرجات الذي وصلنا إليه على 10 لنحصل على الدرجة المتوسطة ، ثم نطبعها .

وبالطبع نحتاج إلى قيمة ابتدائية لكل من counter , total .

الخوارزمية **Algorithm (Pseudocode)**

```
 total  = 0
counter  = 1

while ( counter <= 10 )
        input next grade
        add grade to total
        add 1 to counter

average = total / 10
print average
```

نلاحظ أننا استخدمنا " تكرارا محكوما بعداد " ، حيث أن التكرار توقف حينما تجاوز العداد قيمة معينة (10) . وهذا التكرار يطلَق عليه أيضا " تكرار محدَّد " (definite repetition) لأن عدد مرات التكرار معلوم سلفاً قبل بدء تنفيذ العروة .

وفيما يلي البرنامج المطلوب بلغة C مع مثال للبيانات والنتائج .

```
/* Example 3-8
   Class average program with counter-controlled repetition */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter; /* number of grade to be entered next */
   int grade;   /* grade value */
   int total;   /* sum of grades input by user */
```

```
   int average; /* average of grades */

   /* initialization phase */
   total = 0;   /* initialize total */
   counter = 1; /* initialize loop counter */

   /* processing phase */
   while ( counter <= 10 ) {     /* loop 10 times */
     printf( "Enter grade: " ); /* prompt for input */
     scanf( "%d", &grade );     /* read grade from user */
     total = total + grade;     /* add grade to total */
     counter = counter + 1;     /* increment counter */
   } /* end while */

   /* termination phase */
   average = total / 10; /* integer division */

   printf( "Class average is %d\n", average ); /* display result */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
Enter grade:          98
Enter grade:          76
Enter grade:          71
Enter grade:          87
Enter grade:          83
Enter grade:          90
Enter grade:          57
Enter grade:          79
Enter grade           82
Enter grade           94
Class average is  81
```

ملاحظة ١ أعطينا كلا من total , counter قيمة ابتدائية . وإذا لم نعط أي متغير قيمة ابتدائية فإنه سيحتوي على القيمة السابقة المخزونة في موقع الذاكرة (memory location) الخاص بهذا المتغير (total مثلا) – أي آخر قيمة مخزونة فيه – [وهذه القيمة يطلَق عليها عادة " قيمة من النفايات/قيمة عديمة المنفعة " (a garbage value)]،

وبالتالي فقد نحصل بذلك على نتائج خاطئة ، وهذا مثال لخطأ منطقي (a logical error) .

ملاحظة ٢ :في المثال المعطي بعد نهاية البرنامج :النتيجة النهائية الصحيحة (integer result) التي حصلنا عليها للدرجة المتوسطة هي : 81 . وقد كان مجموع الدرجات في هذا المثال 817 ، وعندما يُقسَم هذا المجموع على 10 فإنه يجب أن يعطي 81.7 ، أي قيمة تحتوي على علامة عشرية (decimal point) ، وكسر عشري . وسنرى بإذن الله في المثال القادم كيفية التعامل مع هذه الأعداد ، والتي يطلق عليها " أعداد ذوات نقطة عائمة " (floating – point numbers) .

مثال ٣-٩ :(تكرار محكوم بقيمة حارسة) (Sentinel – Controlled Repetition

اكتب برنامجا لقراءة درجات عدد اختياري (arbitrary number) من طلاب صفٍّ ما في أحد الاختبارات ، حيث درجة أي طالب عبارة عن عدد صحيح في المدى من 0 إلى 100 . ثم يقوم البرنامج بحساب وطباعة الدرجة المتوسطة للطلاب في هذا الاختبار . افرض أن البرنامج يتوقف عند إدخال البيانات (الدرجات) عندما يُدخِل المستخدِم القيمة 1 – للدلالة على نهاية البيانات .

الحل : عدد الدرجات في هذا المثال غير معلوم سلفاً . والقيمة الخاصة 1 – التي يُدخلها المستخدِم للدلالة على نهاية البيانات ، أي للدلالة على أن آخر درجة قد تم إدخالها ، يطلق عليها القيمة الحارسة (sentinel value) / قيمة الإشارة (signal value) / القيمة الدُّمية (dummy value) / قيمة الراية (flag value) . وواضح أن هذه القيمة يجب أن تُختار بحيث لا تلتبس (confused) مع أي قيمة من قيم البيانات المُدخَلة . وحيث أن أي درجة مُدخَلة تكون غير سالبة ، لذلك فإن 1 – قيمة حارسة مقبولة لهذه المسألة . وهكذا فإن تشغيل هذا البرنامج قد يؤدي إلى استقبال سيل مدخلات (a stream of input) مثل :

95 , 96 , 75 , 74 , 89 , – 1

ويكون مطلوبا من البرنامج حساب وطباعة الدرجة المتوسطة للدرجات : 89 , 74 , 75 , 96 , 95
ولاحظ أن القيمة الحارسة 1 – لا تدخل في حساب القيمة المتوسطة .
والتكرار المحكوم بقيمة حارسة يطلق عليه أيضا " تكرار غير محدد " (indefinite repetition) ، لأن عدد مرات التكرار غير معلوم قبل بدء تنفيذ العروة .

Algorithm (Pseudocode) **الخوارزمية**

```
 total  =  0
counter  =  0

input first grade
while sentinel has not yet been entered
        add grade to total
        add 1 to counter
        input next grade ( possibly the sentinel )

if counter  ≠  0
        average  =  total / counter
        print average
else
        print " No grades were entered "
```

لاحظ عند حساب الدرجة المتوسطة في الخوارزمية أننا قد أخذنا حذرنا كي لا نقسم على الصفر ، فهذه القسمة تُعَدُّ خطأ قاتلا / مميتا (fatal error) **يؤدي إلى فشل البرنامج ، وعادة يسمى "انفجارا/انهيارا"** (bombing / crashing) .

وفيما يلي البرنامج المطلوب بلغة C مع مثال للبيانات والنتائج .

```
/* Example 3-9
   Class average program with sentinel-controlled repetition */
#include <stdio.h>


/* function main begins program execution */
int main()
{
   int counter; /* number of grades entered */
   int grade;   /* grade value */
   int total;   /* sum of grades */


   float average; /* number with decimal point for average */


   /* initialization phase */
   total = 0;   /* initialize total */
   counter = 0; /* initialize loop counter */
```

```
   /* processing phase */
   /* get first grade from user */
   printf( "Enter grade, -1 to end: " ); /* prompt for input */
   scanf( "%d", &grade );               /* read grade from user */

   /* loop while sentinel value not yet read from user */
   while ( grade != -1 ) {
      total = total + grade; /* add grade to total */
      counter = counter + 1; /* increment counter */


      /* get next grade from user */
      printf( "Enter grade, -1 to end: " ); /* prompt for input */
      scanf("%d", &grade);                  /* read next grade */
   } /* end while */


   /* termination phase */
   /* if user entered at least one grade */
   if ( counter != 0 ) {


      /* calculate average of all grades entered */
      average = ( float ) total / counter; /* avoid truncation */


      /* display average with two digits of precision */
      printf( "Class average is %.2f\n", average );
   } /* end if */
   else { /* if no grades were entered, output message */
      printf( "No grades were entered\n" );
   } /* end else */


   return 0; /* indicate program ended successfully */


} /* end function main */
```

```
Enter grade ,   - 1  to  end :     75
Enter grade ,   - 1  to  end :     94
Enter grade ,   - 1  to  end :     97
Enter grade ,   - 1  to  end :     88
Enter grade ,   - 1  to  end :     70
Enter grade ,   - 1  to  end :     64
Enter grade ,   - 1  to  end :     83
Enter grade ,   - 1  to  end :     89
Enter grade ,   - 1  to  end :     -1
Class average is 82 . 50
```

```
Enter grade ,    1  to  end :     -1
No grades were entered
```

نظرا لأن الدرجة المتوسطة عادة تحتوي على جزء كسري – ومثل هذه القيم يطلَق عليها " أعداد ذوات نقطة عائمة " (floating point numbers) – فإن المتغير **average** يعرَّف أنه من النوع **float** ليستطيع احتواء القيمة الكسرية . ولكن نتيجة القسمة **total / counter** تكون عددا صحيحا لأن كلا من **counter** و **total** متغير صحيح ، ولذلك فإن أي جزء كسري يُحذف / يُقطع (lost / truncated) من النتيجة. أي أن الجزء الكسرى يُحذف من نتيجة القسمة قبل إسناد هذه النتيجة إلى المتغير **average** . وللحصول على نتيجة ذات نقطة عائمة من أعداد صحيحة فإننا نولِّد / نُنشئ (create) قيما مؤقتة (temporary values) عبارة عن أعداد ذوات نقطة عائمة ، وذلك باستخدام " مؤثِّر الصب الأحادي " **(float)** (unary cast operator) :

```
Average  =  ( float )  total  /  counter ;
```

والمؤثر **(float)** ينشئ نسخة (copy) مؤقتة ذات نقطة عائمة من معامَلها **total** (operand) . وأما القيمة المخزونة في **total** فهي لا تزال عددا صحيحا (integer) . واستخدام مؤثر صب (a cast operator) بهذه الكيفية يطلَق عليه " تحويل صريح " **(explicit conversion)** . وتصبح العمليات الحسابية الآن عبارة عن عملية قسمة قيمة ذات نقطة عائمة (وهي الصيغة **float** المؤقتة من **total**) على القيمة الصحيحة المخزونة في **counter**. ولكن البرنامج المترجم في لغة C (C compiler) يعرف كيف يوجد قيم

التعابير المكونة فقط من معامَلات (operands) من أنواع بيانات متطابقة (identical data types) . ولذلك حتى تكون جميع المعامَلات من النوع نفسه يقوم البرنامج المترجِم بإجراء عملية تسمَّى " ترقية " (promotion) [وتسمَّى أيضا " تحويل ضمني " (implicit conversion)] على بعض المعامَلات التي يختارها . فمثلا في تعبير يحتوي على نوعي البيانات **int , float** يقوم بعمل نسخ (copies) من المعامَلات التي من النوع **int** ويرقيها (promote) إلى **float** . ففي مثالنا الحالي بعد أن يعمل نسخة من **counter** ويرقيها إلى **float** ، تُجري العملية الحسابية ، وتُسنَد قيمة القسمة الناتجة والتي تكون من النوع **float** إلى **average** . ولغة C تحدد مجموعة من قواعد ترقية المعامَلات من الأنواع المختلفة . وسنناقش بإذن الله تعالى في الفصل الخامس (عن الدوال) الأنواع القياسية للبيانات (standard data types) ، وترتيب ترقيتها (order of promotion) .

ولدينا مجموعة من معامِلات / مؤثِّرات الصب (cast operators) لغالبية أنواع البيانات . ويتم تكوين / صياغة (forming) مؤثِّر الصب بوضع قوسين حول اسم نوع البيانات . ومؤثِّر الصب هو مؤثِّر أحادي (unary operator) أي يأخذ معامَلا واحدا فقط . وقد درسنا في الفصل الثاني المؤثرات الحسابية الثنائية (مثل – , +) . وهناك أيضا في لغة C صيغة أحادية من مؤثري الزائد (+) والناقص (–) ، بحيث يمكن للمبرمج أن يكتب تعبيرا مثل 7- أو 5 +. ويتم تجميع (associating) مؤثرات الصب من اليمين إلى اليسار ، ولها الأولوية نفسها الخاصة بالمؤثرات الأحادية الأخرى مثل (+) الأحادية و (–) الأحادية . وهذه الأولوية أعلى بمستوى واحد من مستوى المؤثرات الضربية * , / , ٪ (multiplicative operators).

وقد استخدم البرنامج السابق ٪.2f (محدد تحويل **printf**) لطباعة قيمة **average** . والمحدد f يحدد أن قيمة ذات نقطة عائمة ستُطبع . وأما 2 . فهي الدقة (precision) التي ستُعرض (displayed) بها القيمة ، وتعني رقمين يمين العلامة العشرية. وإذا استُخدم محدد التحويل ٪f دون تحديد الدقة ، فإن الدقة الافتراضية **(default precision)** 6 تُستخدم ، كما لو أننا استخدمنا محدد التحويل %.6f . وعند طباعة القيم ذوات النقطة العائمة بدقة معينة ، فإن القيمة المطبوعة تُقرَّب (rounded) إلى العدد المشار إليه من المواضع العشرية . وأما القيمة المخزونة في الذاكرة فإنها لا تتغير . فمثلا عند تنفيذ العبارتين التاليتين

```
printf ( "%.2f\n", 3.446 );    /* prints 3.45 */
printf ( "%.1f\n", 3.446 );    /* prints 3.4  */
```

تتم طباعة القيمتين 3.4 , 3.45 .

ملاحظات على استخدام القيم ذوات النقطة العائمة :

- كثيرا ما تنشأ القيم ذوات النقطة العائمة عند إجراء عمليات القسمة . فمثلا عند قسمة 10 على 3 فإن الناتج هو : ....3.3333333 حيث تتكرر متتالية الثلاثات إلى ما لا نهاية . ولكن الحاسوب يحدد لنا بالطبع حيزا محدودا ثابتا (fixed amount of space) (للاحتفاظ بهذه القيمة ، ولذلك فإن القيمة المخزونة ذات النقطة العائمة ستكون قيمة تقريبية فقط .
- وعموما فإن القيم ذوات النقطة العائمة تمثَّل بقيم تقريبية فقط في معظم الحاسبات .
- ولذلك يفضل عدم استخدام اختبار التساوي (equality) عند مقارنة (comparing) (القيم ذوات النقطة العائمة .
- يلاحظ أنه من الخطأ تعيين الدقة (precision) في محددات التحويل في سلسلة التحكم في الصيغة (format control string) في عبارة **scanf** . فالدقة يجوز تعيينها / استخدامها فقط في محددات التحويل في عبارة **printf** .

مثال ٣-١٠ : (بنيات تحكم متداخلة) (Nested Control Structures)

اكتب برنامجا لقراءة نتائج 10 طلاب في أحد الاختبارات ، بحيث أن النتيجة " 1 " تعني نجاح الطالب في الاختبار ، بينما النتيجة " 2 " تعني رسوبه . ويحسب البرنامج كلاً من عدد الطلاب الناجحين وعدد الطلاب الراسبين . ثم يطبع البرنامج ملخص النتائج ، أي يطبع هذين العددين . وإذا زاد عدد الطلاب الناجحين عن 8 ، فإنه يطبع كذلك الرسالة "Raise tuition " التي تعني طلب زيادة الرسوم .

الحل : سنستخدم أسماء المتغيرات التالية :

passes : عداد يعطي أخيرا عدد الطلاب الناجحين .

failures : عداد يعطي أخيرا عدد الطلاب الراسبين .

student : عداد يشير إلى ترتيب النتيجة التي سنُدخلها .

result : نتيجة طالب في الاختبار .

الخوارزمية **Algorithm** (Pseudocode)

```
passes = 0
failures = 0
student = 1
```

```
while ( student <= 10 )
        input next result
        if result = 1
                add 1 to passes
        else
                add 1 to failures
        add 1 to student

  print passes
  print failures
  if passes > 8
        print " Raise tuition "
```

وفيما يلي البرنامج المطلوب بلغة C مع مثالين للتنفيذ (البيانات والنتائج) . ولاحظ أننا قد استفدنا من إحدى خواص لغة C وهي السماح بإعطاء قيم ابتدائية للمتغيرات مع تعريفها / الاعلان عنها (initialization with declaration) وهذا يساعد على تقليل وقت تنفيذ البرنامج (program's execution time). وإعطاء القيم الابتدائية هذا (أي مع التعريف) يتم / يحدث وقت الترجمة (compile time) .

```
/* Example 3-10
   Analysis of examination results */
#include <stdio.h>


/* function main begins program execution */
int main()
{
   /* initialize variables in definitions */
   int passes = 0;   /* number of passes */
   int failures = 0; /* number of failures */
   int student = 1;  /* student counter */
   int result;       /* one exam result */


   /* process 10 students using counter-controlled loop */
   while ( student <= 10 ) {


     /* prompt user for input and obtain value from user */
```

```
      printf( "Enter result ( 1=pass,2=fail ): " );
      scanf( "%d", &result );

      /* if result 1, increment passes */
      if ( result == 1 ) {
         passes = passes + 1;
      } /* end if */
      else { /* otherwise, increment failures */
         failures = failures + 1;
      } /* end else */

      student = student + 1; /* increment student counter */
   } /* end while */

   /* termination phase; display number of passes and failures */
   printf( "Passed %d\n", passes );
   printf( "Failed %d\n", failures );

   /* if more than eight students passed, print "raise tuition" */
   if ( passes > 8 ) {
      printf( "Raise tuition\n" );
   } /* end if */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
Enter Result ( 1= pass, 2= fail ) :     1
Enter Result ( 1= pass, 2 =fail ) :     2
Enter Result ( 1= pass, 2 =fail ) :     2
Enter Result ( 1= pass. 2 =fail ) :     1
Enter Result ( 1= pass, 2 =fail ) :     1
Enter Result ( 1= pass, 2 =fail ) :     1
Enter Result ( 1= pass, 2 =fail ) :     2
Enter Result ( 1= pass, 2 =fail ) :     1
Enter Result ( 1= pass, 2 =fail ) :     1
Enter Result ( 1= pass, 2 =fail ) :     2
Passed 6
Failed  4
```

```
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    2
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Enter Result ( 1= pass, 2= fail ) :    1
Passed 9
Failed  1
Raise tuition.
```

مؤثّرات الإسناد (Assignment Operators)

توجد في لغة C عدة مؤثّرات إسناد لاختصار تعابير الإسناد (abbreviating assignment expressions).

فمثلا العبارة

```
c = c + 3;
```

يمكن اختصارها باستخدام مؤثر إسناد الجمع += (addition assignment operator) هكذا :

```
c += 3;
```

والمؤثر += يضيف قيمة التعبير الموجود يمين المؤثّر إلى قيمة المتغير الموجود يسار المؤثر ، ويخزن النتيجة (نتيجة الإضافة / الجمع) في المتغير الموجود يسار المؤثّر . وأي عبارة صيغتها :

```
variable = variable operator expression ;
```

حيث operator هو أحد المؤثرات الثنائية :

+ , - , * , / , ٪ (وهناك كذلك مؤثرات أخرى سيُشار إليها فيما بعد) يمكن أن تُكتب في الصيغة :

```
Variable operator = expression
```

وهكذا ، فالإسناد c += 3 يضيف 3 إلى c .

مثال ٣-١١ :

نفرض أن لدينا التعريفات التالية مع القيم الابتدائية المعطاة :

int c = 3 , d = 5 , e = 4 , f = 6 , g = 12;

الجدول التالي يبين بعض مؤثرات الإسناد ، وأمثلة لتعابير تستخدم هذه المؤثرات ، وبيان / شرح معنى كل من هذه التعابير ، والقيمة التي تُسند للمتغير في كل تعبير :

مؤثر الإسناد Assignment operator	تعبير مثال Sample expression	التفسير Explanation	القيمة المسنَدة Assigns
+ =	c += 7	c = c + 7	10 إلى c
- =	d - = 4	d = d – 4	1 إلى d
* =	e * = 5	e = e * 5	20 إلى e
/ =	f / = 3	f = f / 3	2 إلى f
% =	g % = 9	g = g % 9	3 إلى g

مؤثرات الإسناد الحسابية
Arithmetic assignment operators

مؤثرات الزيادة والنقصان Increment and Decrement Operators

الجدول التالي يوضح معنى كل من مؤثر الزيادة الأحادي (++) (unary increment operator) ، ومؤثر النقصان الأحادي (--) (unary decrement operator) . والمؤثر (+ +) يمكن أن يُستخدم لزيادة قيمة متغير مثل c بواحد [بدلا من استخدام التعبير c = c + 1 أو التعبير c + = 1] فنكتب c + + أو + + c . وإذا وُضع مؤثر الزيادة [أو مؤثر النقصان] قبل متغير (c + +) [أو (c - -)] فيُشار إلى المؤثر على أنه "مؤثر الزيادة السابق" (preincrement operator) [أو مؤثر النقصان السابق predecrement operator] . بينما إذا وُضع بعد المتغير فيُشار إليه على أنه "مؤثر الزيادة اللاحق" (postincrement operator) [أو مؤثر النقصان اللاحق postdecrement operator] . ووَضع مؤثر الزيادة [أو النقصان] قبل المتغير يؤدي إلى زيادة [أو نقصان] قيمة المتغير بواحد أولا ، ثم استخدم قيمة المتغير الجديدة في التعبير الذي يظهر فيه هذا المتغير . بينما وَضع مؤثر الزيادة [أو النقصان] بعد المتغير يودي إلى استخدام القيمة الحالية (current value) للمتغير في التعبير الذي يظهر فيه أولا ، ثم زيادة [أو نقصان] قيمة المتغير بواحد .

المؤثر Operator	تعبير مثال Sample expression	التفسير Explanation
+ +	+ + a	زِد قيمة a بواحد ، ثم استخدِم قيمة a الجديدة في التعبير الذي تظهر فيه a .
+ +	a + +	استخدِم قيمة a الحالية في التعبير الذي تظهر فيه a ، ثم زِد قيمة a بواحد .
– –	– – b	أنقِص قيمة b بواحد ، ثم استخدِم قيمة b الجديدة في التعبير الذي تظهر فيه b .
– –	b – –	استخدِم قيمة b الحالية في التعبير الذي تظهر فيه b ، ثم أنقِص قيمة b بواحد .

مؤثرا الزيادة والنقصان
increment and decrement operators

مثال ٣-١٢ :

البرنامج التالي يوضح الفارق بين صيغتى الزيادة السابقة والزيادة اللاحقة للمؤثِّر (+ +) . ولاحظ أن صيغة الزيادة اللاحقة للمتغير c (في البرنامج) تجعل قيمته تزداد بعد أن يُستخدم في عبارة printf . بينما صيغة الزيادة السابقة للمتغير c تجعل قيمته تزداد قبل أن يُستخدم في عبارة printf .

```
/* Examle 3-12
   Preincrementing and postincrementing */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int c;            /* define variable */

   /* demonstrate postincrement */
   c = 5;            /* assign 5 to c */
```

```
   printf( "%d\n", c );   /* print 5 */
   printf( "%d\n", c++ ); /* print 5 then postincrement */
   printf( "%d\n\n", c ); /* print 6 */

   /* demonstrate preincrement */
   c = 5;              /* assign 5 to c */
   printf( "%d\n", c );   /* print 5 */
   printf( "%d\n", ++c ); /* preincrement then print 6 */
   printf( "%d\n", c );   /* print 6 */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
5
5
6

5
6
6
```

والبرنامج يعرض قيمة c قبل وبعد استخدام المؤثر + + . ومؤثر النقصان يعمل بطريقة مماثلة .

ملاحظات :

*) عند زيادة (أو نقصان) متغير باستخدام عبارة مستقلة بذاتها تتساوى في التأثير صيغتا الزيادة السابقة والزيادة اللاحقة ، أي : c + + و + + c .
*) ولكن حينما يظهر متغير في سياق تعبير أكبر فعند ذلك فقط يختلف تأثير صيغة الزيادة السابقة عن تأثير صيغة الزيادة اللاحقة (وكذلك بالنسبة لصيغتي النقصان السابق والنقصان اللاحق) .
*) معامَل (operand) مؤثِّر الزيادة أو مؤثِّر النقصان يجب أن يكون اسم متغير بسيط (simple variable name) فقط .
*) محاولة التأثير بمؤثر الزيادة أو النقصان على تعبير ليس اسم متغير بسيط يُعَدُّ خطأ تركيبيا، كأن نكتب مثلا : (1 + x) + + .

قواعد الأولوية والتجميع Precedence and Associativity Rules

الجدول التالي يلخص قواعد أولوية وتجميع المؤثرات التي درسناها حتى الآن .
والأولوية تتناقص من أعلى لأسفل الجدول .

النوع Type		التجميع Associativity	المؤثرات Operators
الأحادية	(unary)	من اليمين إلى اليسار	++ -- + -
الضربية	(multiplicative)	من اليسار إلى اليمين	* / %
الجمعية	(additive)	من اليسار إلى اليمين	+ -
العلاقية	(relational)	من اليسار إلى اليمين	< <= > >=
التساوي	(equality)	من اليسار إلى اليمين	== !=
الشَّرطية	(conditional)	من اليمين إلى اليسار	?:
الإسناد	(assignment)	من اليمين إلى اليسار	= += -= *= /= %=

تمرينات رقم ٣

٣-١) اكتب أربع عبارات مختلفة بلغة C تضيف كلٌّ منها واحدا 1 إلى متغير صحيح x .

٣-٢) اكتب عبارة واحدة بلغة C لتنفيذ كل مما يلي :

أ) أسنِد مجموع قيمتى **y** , **x** إلى **z** ، ثم زِد قيمة x بواحد (بعد حساب المجموع) .

ب) اضرب قيمة المتغير **product** في 2 باستخدام المؤثر = * .

ج) اضرب قيمة المتغير **product** في 2 باستخدام المؤثرين = و * .

د) اختبر إن كانت قيمة المتغير **count** أكبر من 10 . فإن كانت كذلك ، فاطبع الرسالة ". Count is greater than 10" .

هـ) أَنقِص المتغير x بواحد ، ثم اطرحه من المتغير **total** .

و) أَضِف المتغير x إلى المتغير **total** ، ثم أَنقِص x بواحد .

ز) احسب الباقي (remainder) بعد قسمة **q** على **divisor** ، وأَسند النتيجة إلى **q** . اكتب هذه العبارة بطريقتين مختلفتين .

ح) اطبع القيمة 123.4567 بدقة تبلغ رقمين عشريين . ما هي القيمة التي ستُطبع ؟

ط) اطبع القيمة 3.14159 ذات النقطة العائمة بدقة تصل إلى ثلاثة أرقام يمين العلامة العشرية . ما هي القيمة التي ستُطبع ؟

٣-٣) اكتب عبارة واحدة بلغة c لتنفيذ كل مما يلي :

أ) عرِّف متغيرين **x** , **sum** من النوع **int** .

ب) أسنِد القيمة الابتدائية 1 إلى المتغير **x** .

ج) أسنِد القيمة الابتدائية 0 إلى المتغير **sum** .

د) أضِف المتغير **x** إلى المتغير **sum** ، وأسنِد النتيجة إلى المتغير **sum** .

هـ) اطبع الرسالة " The sum is : " تليها قيمة المتغير **sum** .

٣-٤) استخدم العبارات التي كتبتها عند إجابة السؤال السابق (٣-٣) في برنامج يقوم بحساب مجموع الأعداد الصحيحة من 1 إلى 10 . استخدم عبارة **while** في عروة تقوم بزيادة قيمة **x** وحساب المجموع المطلوب ، بحيث تنتهي العروة حينما تصبح قيمة **x** مساوية 11 .

٣-٥) ما هي قيمة كل من المتغيرين **x** , **product** بعد إجراء العملية الحسابية التالية ، بفرض أن قيمة كل من **x** , **product** تساوي 5 عند بدء تنفيذ العبارة

```
product *= x ++;
```

٣-٦) اكتب عبارات مفردة بلغة C تقوم بما يلي :

أ) أدخِل (input) / اقرأ متغيرا صحيحا **x** باستخدام **scanf** .

ب) أدخِل (input) / اقرأ متغيرا صحيحا **y** باستخدام **scanf** .

ج) اعط متغيرا صحيحا i القيمة الابتدائية 1 .

د) اعط متغيرا صحيحا **power** القيمة الابتدائية 1 .

هـ) اضرب المتغير **power** في **x** وأسند النتيجة إلى **power**.

و) زِد المتغير i بواحد .

ز) اختبر ما إذا كانت قيمة **i** أقل من أو تساوي **y** وذلك في شرط عبارة **while** .

ح) اطبع قيمة المتغير الصحيح **power** باستخدام **printf** .

٣-٧) اكتب برنامجا بلغة C يستخدم العبارات التي كتبتها عند إجابة السؤال السابق (٣-٦) وذلك لحساب قيمة **x** مرفوعة للأُسِّ **y** (**x** raised to the **y** power) ، وبحيث يحتوي البرنامج على عبارة تحكم **while** للتكرار .

٣-٨) اكتشف وصحح أي أخطاء في كل مما يلي :

أ)

```
while ( c <= 5 ) {
    product *=c;
    ++c;
```

ب)

```
scanf( "%. 4f" , &value ) ;
```

ج)

```
if ( gender == 1 )
    printf( "Woman\n" );
else ;
    printf( "Man\n" );
```

٣-٩) نفرض أن قيمة **z** تساوي 100 . ما الخطأ في عبارة التكرار **while** التالية ، والتي يُفترض أن تحسب مجموع الأعداد الصحيحة من 100 إلى 1 تنازليا ؟

```
while ( z >= 0 )
    sum += z;
```

٣-١٠) اكتشف وصحح أي أخطاء في كل من القطع التالية . (ملاحظة : قد يوجد أكثر من خطأ في أي قطعة .)

أ)

```
if ( age >= 65 );
   printf( "Age is greater than or equal to 65\n" );
else
    printf( "Age is less than 65\n" );
```

ب)

```
int x = 1, total;

while ( x <= 10 ) {
     total += x;
     ++x;
}
```

ج)

```
While ( x <= 100 )
     total += x;
     ++x;
```

د)

```
while ( y > 0 ) {
     printf( "%d\n", y );
     ++y;
}
```

٣-١١) ماذا يطبع البرنامج التالي ؟

```
#include <stdio.h>

int main()
 {
     int x = 1, total = 0, y;

    while ( x <= 10 ) {
        y = x * x;
        printf( "%d\n", y );
        total += y;
       ++x;
     }

     printf("Total is %d\n", total);
```

```
    return 0;
}
```

٣-١٢) اختر الإجابة الصحيحة في كل مما يلي :

i) أي العبارات التالية تطبع الرسالة " Passed " إذا كانت درجة الطالب **grade** أكبر من أو تساوي 60 بينما تطبع الرسالة " **Failed** " إذا كانت درجته أقل من 60 ؟

أ) `printf("%s\n",grade>=60:"Passed":"Failed");`

ب) `grade>=60:printf("Passed\n")?printf("Failed\n");`

ج) `printf("%s\n",grade>=60?"Passed":"Failed");`

د) `grade>=60?printf("Passed\n")?printf("Failed\n");`

ii) أي القطع التالية تولِّد خطأ ؟

أ)
```
if ( answer ==7 )
    printf( "correct" );
else
    printf( "incorrect" );
```

ب) `printf(answer==7?"correct":"incorrect");`

ج) `printf(answer==7?"correct":"incorrect");`

د) `answer ==7 ? printf( "correct" ):`

iii) ما هي قيمة p بعد الانتهاء من تنفيذ عروة while في القطعة التالية ؟

```
p = 2;
while ( p < 2000 )
  p = 2*p;
```

أ) 1023

ب) 1024

ج) 2047

د) 2048

iv) أي متغير لم يعطَ قيمة ابتدائية (an uninitialized variable) سيحتوي على :

أ) آخر قيمة مخزونة في موقع الذاكرة المحجوز لهذا المتغير .

ب) لا يحتوي على أي قيمة .

ج) القيمة صفر .

د) قيمة عشوائية (a randomly assigned value) .

v) ما هي قيمة x النهائية بعد تنفيذ العمليات التالية ؟

```
int x = 21;
double y = 6;
double z = 14;
y = x / z;
x = 5.5*y;
```

أ) 8.25

ب) 5.5

ج) 5

د) 8

vi) نفرض أن x تساوي 4 . في أي القطع التالية y لا تساوي 5 بعد التنفيذ ؟

أ) `y = 5;`

ب) `y = x++;`

ج) `y = ++ x;`

د) `y = x = 5;`

vii) أي العبارات التالية صحيحة (true) ؟

أ) التعبير (a + 1) ++ يضيف 2 إلى a .

ب) التعبير a** يضرب a في 1 .

ج) صيغة ANSI القياسية للغة C تحدد الترتيب الذي تُحسب به قيم معامَلات (operands) أي مؤثر (operator) .

د) التعبير (abc * 37) -- يُعَدُّ خطأً تركيبيا .

٣-١٣) اكتب برنامجا لحساب القيمة المتوسطة لعدد الأميال التي تقطعها السيارة لكل جالون من الوقود (البنزين / الجازولين) وذلك لعدد اختياري من خزانات الوقود (average miles / gallon for each tank of gas) ، وكذلك القيمة المتوسطة الكلية لعدد الأميال للجالون الواحد (overall miles / gallon) لهذه الخزانات ، وذلك باتباع الخطوات التالية :

- يقرأ البرنامج كلاً من عدد الجالونات وعدد الأميال المقطوعة المقابلة وذلك لعدد من خزانات الوقود ، وتنتهي هذه البيانات عندما يُدخل المستخدم القيمة 1- لعدد الجالونات .
- يحسب البرنامج ويطبع القيمة المتوسطة لعدد الأميال المقطوعة للجالون الواحد وذلك لكل خزان من هذه الخزانات .
- يحسب البرنامج ويطبع القيمة المتوسطة الكلية لعدد الأميال للجالون الواحد .

وفيما يلي مثال لمدخلات ومخرجات هذا البرنامج .

```
Enter the gallons used (-1 to end): 12.8
Enter the miles driven: 287
The miles / gallon for this tank was 22.421875

Enter the gallons used (-1 to end): 10.3
Enter the miles driven: 200
The miles / gallon for this tank was 19.417475

Enter the gallons used (-1 to end): 5
Enter the miles driven: 120
The miles / gallon for this tank was 24.000000

Enter the gallons used (-1 to end): -1

The overall average miles/gallon was 21.601423
```

٣-١٤) اكتب برنامجا يحدد ما إذا كان أي من زبائن / عملاء (customers) أحد المحال التجارية (department store) قد تجاوز الحد الأقصى المسموح له من المشتريات (credit limit) على رقم حسابه ، وذلك باتباع الخطوات التالية :

- يستمر البرنامج في قراءة البيانات التالية لكل عميل :
  * رقم الحساب (account number) .
  * الرصيد (balance) عند بداية الشهر .
  * مجموع مشترياته (على حسابه) (total charges) خلال الشهر .
  * مجموع المبالغ المستحقة له (total credits) خلال الشهر .
  * الحد الأقصى المسموح له (credit limit) .

ويتوقف البرنامج عن قراءة البيانات حين يُدخِل المستخدِم القيمة 1- لرقم الحساب .

- يحسب البرنامج لكل عميل رصيده الجديد باستخدام العلاقة :

new balance = balance + total charges - total credits

- إذا زاد الرصيد الجديد عن الحد الأقصى المسموح فإن البرنامج يطبع : رقم الحساب ، والحد الأقصى المسموح ، والرصيد ، والرسالة " Credit Limit Exceeded " .

وفيما يلي مثال لمدخلات ومخرجات البرنامج .

```
Enter account number ( -1 to end): 100
Enter beginning balance: 5394.78
Enter total charges: 1000.00
Enter total credits: 500.00
Enter credit limit: 5500.00
Account: 100
Credit limit: 5500.00
Balance: 5894.78
Credit Limit Exceeded.

Enter account number ( -1 to end ): 200
Enter beginning balance: 1000.00
Enter total charges: 123.45
Enter total credits: 321.00
Enter credit limit: 1500.00

Enter account number ( -1 to end ): 300
Enter beginning balance: 500.00
Enter total charges: 274.73
Enter total credits: 100.00
Enter credit limit: 800.00

Enter account number ( -1 to end ): -1
```

٣-١٥) تعطي إحدى الشركات عمولة (commission) تبلغ 9% من السعر الإجمالي للمبيعات (gross sales) خلال أسبوع التي يقوم ببيعها أي واحد من مندوبي المبيعات (salespersons) ، وذلك بالإضافة إلى راتبه الأسبوعي الثابت والذي يساوي 5000 $. مثلا إذا بلغت مبيعات أحد المندوبين خلال الأسبوع الماضي 2000 $ ، فإنه سيتقاضى عن هذا الأسبوع راتبا إجماليا يساوي

Salary = 200.0 + 0.09 * gross sales
= 200.0 + 0.09 * 5000 = $ 650

اكتب برنامجا يحسب الراتب الإجمالي الأسبوعي لعدد من مندوبي المبيعات بالشركة ، بحيث يقرأ البرنامج لكل واحد من هؤلاء جملة مبيعاته خلال الأسبوع الماضي ، ويتوقف البرنامج عن القراءة حينما يُدخل المستخدم القيمة 1- لجملة المبيعات .
وفيما يلي مثال لمدخلات ومخرجات البرنامج .

```
Enter sales in dollars ( -1 to end): 5000.00
Salary is: $650.00

Enter sales in dollars ( -1 to end ): 1234.56
Salary is: $311.11

Enter sales in dollars ( -1 to end ): 1088.89
Salary is: $298.00

Enter sales in dollars ( -1 to end ): -1
```

٣-١٦) من الوسائل الحديثة لتزيين المنكر وخداع الناس لدفعهم لارتكابه دون إحساسهم بمعصية الله عز وجل من أجل تحقيق مآرب دنيوية عاجلة والجري وراء تحصيل متاع الدنيا القليل الزائل وحطامها الفاني تسمية الأشياء بغير أسمائها الحقيقية كتسمية الربا (usury) فائدة (interest) ! فبدل أن يخاف المرء من الدخول يقينا في حرب خاسرة غير متكافئة لأنها في مواجهة الله تعالى ورسوله صلى الله عليه وسلم بسبب تعامله بالربا " يا أيها الذين آمنوا اتقوا الله وذروا ما بقي 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) يشعر بالفرح والارتياح لأنه سيستفيد من هذه " الفائدة" التي سيأخذها مثلا من بنك ربوي حين يضع رأس ماله فيه ، أو يأخذها من شخص أو مؤسسة حين يقرضها رأس ماله (بدل أن يكون القرض حسناً ، والأجر من الله تعالى مضاعفا أضعافا كثيرة) .

نفرض أن الربا [أو الفائدة البسيطة ! (simple interest)] على قرض (loan) [أو رأس المال (principal)] سيحسب بالصيغة

```
interest  = principal * rate * days / 365;
```

حيث rate : هو المعدَّل السنوي للربا (annual interest rate)
days: مدة القرض بالأيام .

اكتب برنامجا يقرأ قيم principal , rate , days لعدد من القروض ، ويحسب ويطبع قيمة ربا كل منها . ويتوقف البرنامج حين يقرأ قيمة رأس المال : 1- .
وفيما يلي مثال لمدخلات البرنامج ومخرجاته .

```
Enter loan principal ( -1 to end): 1000.00
Enter interest rate: .1
Enter term of the loan in days: 365
The interest charge is $100.00

Enter loan principal ( -1 to end ): 1000.00
Enter interest rate: .08375
Enter term of the loan in days: 224
The interest charge is $51.40

Enter loan principal ( -1 to end ): 10000.00
Enter interest rate: .09
Enter term of the loan in days: 1460
The interest charge is $3600.00

Enter loan principal ( -1 to end ): -1
```

٣-١٧) اكتب برنامجا يقرأ بيانات عدد من موظفي (employees) إحدى الشركات ، ويحسب ويطبع الأجر الإجمالي / راتب (salary) كل موظف . وتشتمل بيانات أي موظف على عدد ساعات العمل (hours worked) خلال الأسبوع الماضي ، وأجره في الساعة الواحدة (hourly rate) . وأما الأجر الإجمالي للموظف فيُحسب بناء على القاعدة التالية :

* إذا كان عدد ساعات العمل أقل من أو يساوي 40 ساعة :

الأجر الإجمالي = عدد ساعات العمل × الأجر في الساعة

* وإذا كان عدد ساعات العمل أكبر من 40 ساعة :

الأجر الإجمالي = (40 × الأجر في الساعة) + الأجر الإضافي

حيث :

الأجر الإضافي = (عدد ساعات العمل – 40) * الأجر في الساعة * 1.5

ويتوقف البرنامج عن قراءة البيانات حين يقرأ القيمة 1- لعدد ساعات العمل .
وفيما يلي مثال لمدخلات ومخرجات البرنامج .

```
Enter number of hours worked ( -1 to end ): 39
Enter hourly rate of the worker ( $00.00 ): 10.00
Salary is $390.00

Enter number of hours worked ( -1 to end ): 40
Enter hourly rate of the worker ( $00.00 ): 10.00
Salary is $400.00

Enter number of hours worked ( -1 to end ): 41
Enter hourly rate of the worker ( $00.00 ): 10.00
Salary is $415.00

Enter number of hours worked ( -1 to end ): -1
```

٣-١٨) اكتب برنامجا يوضح الفارق بين النقصان السابق (predecrementing) والنقصان اللاحق (postdecrementing) باستخدام مؤثّر النقصان (--) (decrement operator) .

٣-١٩) اكتب برنامجا يستخدم عروة loop لطباعة الأعداد من 1 إلى 10 بجانب بعضها البعض على السطر نفسه ، مع ترك 3 فراغات بين أي عددين متجاورين .

٣-٢٠) اكتب برنامجا يقرأ 10 أعداد ، ثم يوجد ويطبع أكبر عدد .
ويستخدم البرنامج المتغيرات الثلاثة التالية :
counter : عدَّاد يعد حتى 10 (أي يتابع كم عددا تمت قراءتها)
ويتأكد أن 10 أعداد قد تم تشغيلها) .
number : العدد الحالي (current number) الذي قرأه البرنامج .
largest : أكبر عدد وجدناه حتى الآن .
إرشاد : اتبع الخوارزمية التالية :

الخوارزمية **Algorithm** (Pseudocode)

```
input first number into largest
counter =

while ( counter <= 10
        input number
        if number > largest
                largest = number
```

counter ++

print largest

٣-٢١)

اكتب برنامجا يستخدم عروة لطباعة جدول القيم التالية :

N	10 * N	100 * N	1000 * N
1	10	100	1000
2	20	200	2000
3	30	300	3000
4	40	400	4000
5	50	500	5000
6	60	600	6000
7	70	700	7000
8	80	800	8000
9	90	900	9000
10	100	1000	10000

يمكنك استخدام رمز الجدولة \t (tab character) **في عبارة printf للفصل بين الأعمدة (باستخدام** tabs) .

٣-٢٢) **اكتب برنامجا يستخدم عروة لطباعة جدول القيم التالية :**

A	A+2	A+4	A+6
3	5	7	9
6	8	10	12
9	11	13	15
12	14	16	18
15	17	19	21

٣-٢٣) **اكتب برنامجا يقرأ** 10 **أعداد ، ثم يوجد ويطبع أكبر عددين .**

ملاحظة : يجب إدخال أي عدد مرة واحدة فقط .

إرشاد : استخدم طريقة مماثلة لتلك الموضحة في السؤال ٣-٢٠ .

٣-٢٤) **عدِّل برنامج مثال ٣-١٠) بحيث يتحقق من صحة المدخلات** (validate the inputs) **كما يلي : عند إدخال أي قيمة : إن لم تكن القيمة المدخَلة** 1 **أو** 2 **(وتعني – على الترتيب – نجاح الطالب أو رسوبه) استخدم عروة تظل تُخبر المستخدِم أن**

القيمة المدخلة غير صحيحة ، وتطلب منه إدخال قيمة صحيحة (correct value) :
1 أو 2 ، إلى أن يُدخِل قيمة صحيحة .

٣-٢٥) ما هي مخرجات البرنامج التالي ؟ أي ماذا يطبع ؟

```
#include <stdio.h>

/* function main begins program execution */
 int main()
 {
    int count = 1; /* initialize count */

    while ( count <= 10 ) { /* loop 10 times */

       /* output line of text */
       printf( "%s\n", count % 2 ? "****" : "++++++++" );
       count++; /* increment count */
    } /* end while */

    return 0; /* indicate program ended successfully */

  } /* end function main */
```

٣-٢٦) ما هي مخرجات البرنامج التالي ؟

```
#include <stdio.h>

/* function main begins program execution */
 int main()
 {
    int row = 10; /* initialize row */
    int column; /* define column */

    while ( row >= 1 ) { /* loop until row < 1 */
       column = 1; /* set column to 1 as iteration begins */

       while ( column <= 10 ) { /* loop 10 times */
          printf( "%s", row % 2 ? "<": ">" ); /* output */
          column++; /* increment column */
       } /* end inner while */

       row--; /* decrement row */
```

```
    printf( "\n" ); /* begin new output line */
  } /* end outer while */

  return 0; /* indicate program ended successfully */

} /* end function main */
```

٣-٢٧) [مشكلة **else** المتدلية (Dangling else problem)]

أوجد مخرجات كل من القطعتين التاليتين (أ ، ب) لكل من الحالتين

(i) $x = 9$, $y = 11$

(ii) $x = 11$, $y = 9$

لاحظ أن البرنامج المترجم (compiler) يتجاهل (ignores) مسألة ضبط المسافات (indentation) / الإزاحات / عدد الفراغات في أي برنامج لغة C . ولاحظ كذلك أن البرنامج المترجم دائما يُلحق (associates) أيَّ else بأقرب if سابقة (previous) (لم يُلحَق بها else) ما لم نخبره غير ذلك باستخدام القوسين { } . ونظرا لأنه قد لا يكون واضحا من النظرة الأولى بأي if ترتبط else معينة ، فيُشار إلى هذه المسألة على أنها مشكلة else المتدلية . وفي القطعتين التاليتين (أ ، ب) تعمَّدنا عدم ضبط المسافات والإزاحات (eliminated the indentation) لبيان هذه المشكلة .

أ)

```
if ( x < 10 )
if ( y > 10 )
printf( " Prayer is light.\n" );
else
printf( "Charity is a proof.\n" );
printf( "Patience is illumination.\n" );
```

ب)

```
if ( x < 10 ) {
if ( y > 10 )
printf( " Prayer is light.\n" );
}
else {
printf ("Charity is a proof.\n" );
printf( " Patience is illumination.\n" );
}
```

٣-٢٨) [مشكلة **else** المتدلية ، سؤال آخر]

عدِّل القطعة التالية بحيث تطبع المخرجات المعطاة (في كل من أ ، ب ، ج) وبحيث :

* تضبط المسافات والازاحات (proper indentation) في القطعة .
* لا يُسمح بعمل أي تغييرات في القطعة سوى إضافة أقواس { } .

[ملاحظة : من الممكن ألا تحتاج لعمل أي تعديل في القطعة] .

```
if ( y == 8 )
if ( x == 5 )
printf( "Every takbeera is a charity.\n" );
else
printf( " Charity is a proof.\n" );
printf( " Patience is illumination.\n" );
printf( "Every tahleela is a charity.\n" );
```

(أ) نفرض أن x = 5 , y = 8 والمخرجات الناتجة هي :

```
Every takbeera is a charity.
Patience is illumination.
Every tahleela is a charity.
```

(ب) نفرض أن x = 5 , y = 8 والمخرجات الناتجة هي :

```
Every takbeera is a charity.
```

(ج) نفرض أن x = 5 , y = 8 والمخرجات الناتجة هي :

```
Every takbeera is a charity.
Every tahleela is a charity.
```

(د) نفرض أن x = 5 , y = 7 والمخرجات الناتجة هي :

```
Charity is a proof .
Patience is illumination.
Every tahleela is a charity.
```

٣-٢٩) اكتب برنامجا يقرأ طول ضلع مربع (side of a square) ، ثم يطبع هذا المربع مكوَّنا من نجوم (asterisks) . مثلا إذا قرأ البرنامج طول الضلع 4 ، فإنه يطبع الشكل

```
****
****
****
****
```

ملاحظة : نفرض أن طول ضلع المربع [/ السعة / الحجم(size)] الذي يقرأه البرنامج هو أي عدد صحيح بين 1 و 20]

٣-٣٠) عدِّل برنامج السؤال السابق ٣-٢٩) بحيث يطبع مربعا أجوف (a hollow square) . مثلا إذا قرأ البرنامج الطول/ السعة 5 (size) ، فإنه يطبع الشكل

```
*****
*   *
*   *
*   *
*****
```

٣-٣١) السلسلة مزدوجة القراءة (palindrome) هي أي عدد (number) أو جملة نصية (text phrase) أو سلسلة رموز تُقرأ بالكيفية نفسها طردا وعكسا (forwards and backwards) (من اليمين إلى اليسار ومن اليسار إلى اليمين) . فمثلا كل من الأعداد الصحيحة خماسية الأرقام (five – digit integers) التالية يُعَدُّ سلسلة مزدوجة القراءة : 11611 , 45554 , 55555 , 12321 .

اكتب برنامجا يقرأ عددا صحيحا خماسي الأرقام ويحدد ما إذا كان هذا العدد سلسلة مزدوجة القراءة أم لا [إرشاد : استخدم مؤثِّري القسمة والباقي (division and remainder operators) لفَصْل / لتجزئة العدد إلى أرقامه المفردة (individual digits) .

٣-٣٢) اكتب برنامجا يقرأ عددا صحيحا ثنائيا (a binary integer) [أي يتكون من أصفار 0's وآحاد 1's فقط] – بحيث لا يزيد عن خمسة أرقام ثنائية (5 binary digits maximum) – ثم يحسب ويطبع مكافئه العشري (decimal equivalent) .

إرشاد : استخدم مؤثري الباقي والقسمة (remainder and division operators) للحصول على أرقام العدد الثنائية واحدا تلو الآخر من اليمين إلى اليسار ، ثم احسب المكافئ العشري بضرب كل رقم في قيمته المكانية (positional value) وجمع النواتج ، حيث :

القيمة المكانية للرقم الثنائي أقصى اليمين تساوي 1 ،

القيمة المكانية للرقم الثنائي التالي على يساره تساوي 2 ، ثم 4 ، ثم 8 ، ثم 16 .. وهكذا .

مثلا : المكافئ العشري للعدد الثنائي 1101 هو :

1 * 1 + 0 * 2 + 1 * 4 + 1 * 8 = 13

٣-٣٣) لتقدير مدى سرعة حاسوبك اكتب برنامجا يستخدم عروة while للعد من 1 إلى 300,000,000 بالآحاد . وكلما وصل العد إلى أحد مضاعفات (a multiple of) 100,000,000 اطبع هذا العدد (هذا المضاعف) على الشاشة . واستخدم ساعتك لتقدير الوقت الذي يستغرقه تكرار كل مليون من هذا العد .

٣-٣٤) اكتب برنامجا يطبع 100 نجمة (asterisks) واحدة واحدة . وبعد كل 10 نجوم يطبع البرنامج رمز سطر جديد (a newline character) . [إرشاد : عِد من 1 إلى 100 . واستخدم مؤثّر الباقي (remainder operator) لمعرفة متى وصل العد إلى أحد مضاعفات 10 وبالتالي نطبع عندها رمز السطر الجديد .]

٣-٣٥) اكتب برنامجا يقرأ عددا صحيحا مكونا من خمسة أرقام ، ويحدد ويطبع عدد الأرقام التي يساوي كل منها 7 في هذا العدد .

مثلا : إذا قرأ العدد 27747 فإنه يطبع رسالة مثل :

" The number 27747 has 3 sevens in it "

٣-٣٦) اكتب برنامجا يطبع / يعرض (displays) شكل / نموذج لوحة الداما (checkerboard pattern) التالي

```
********
 ********
********
 ********
********
 ********
********
 ********
```

بحيث يستخدم البرنامج ثلاث عبارات طباعة فقط ، واحدة من كل من الصيغ التالية

```
printf( "* " );
printf( " " );
printf( "\n" );
```

٣-٣٧) اكتب برنامجا يستمر في طباعة مضاعفات العدد الصحيح 2 ، أي طباعة

2 , 4 , 8 , 16 , 32 , 64 , 128 , ...

وبحيث لا تنتهي عروة الطباعة ، أي تكون عروة لا نهائية . ماذا يحدث عند تشغيل هذا البرنامج ؟

٣-٣٨) اكتب برنامجا لقراءة نصف قطر (radius) دائرة [كقيمة ذات نقطة عائمة (a float value)] وحساب وطباعة كل من القطر (diameter) ، والمحيط (circumference) ، والمساحة (area) ؟ . استخدم القيم3.14159 للثابت π

٣-٣٩) ما الخطأ في العبارة التالية ؟ أعد كتابة العبارة بحيث تُحَقِّق ما أراده المبرمج غالبا .

```
printf( "%d" , ++( x + y));
```

٣-٤٠) اكتب برنامجا يقرأ ثلاث قيم float غير صفرية ، ويطبع رسالة تفيد ما إذا كان ممكنا لهذه القيم أن تمثل أطوال أضلاع مثلث أم لا .

٣-٤١) اكتب برنامجا يقرأ ثلاثة أعداد صحيحة غير صفرية ، ويطبع رسالة تفيد ما إذا كان ممكنا لهذه الأعداد أن تمثل أطوال أضلاع مثلث قائم الزاوية (right triangle) أم لا .

٣-٤٢) ترسل إحدى الشركات بياناتها في صيغة أعداد صحيحة رباعية الأرقام (4 – digit integers) عبر التليفون . ولضمان الحفاظ على سِرِّيَّة / أَمن (security) هذه البيانات تود الشركة تشفير (encrypting) البيانات قبل إرسالها .

أ) اكتب برنامجا لتشفير البيانات باتباع الخطوات التالية :

- اقرأ عددا صحيحا رباعي الأرقام .
- استبدل بكل رقم : الباقي (remainder) بعد قسمة (مجموع هذا الرقم + 7) على 10 .
- بدِّل الرقم الأول مع الرقم الثالث .
- بدِّل الرقم الثاني مع الرقم الرابع .

- اطبع العدد الصحيح المشفر .

ب) اكتب برنامجا آخر لفك شفرة (decrypting) البيانات المشفَّرة المستقبلة ، أي يقرأ البرنامج عددا صحيحا مشفَّرا رباعي الأرقام ، ويفك شفرته أي يعيده إلى العدد الأصلي (original number) .

٣-٤٣) مضروب (factorial) أي عدد صحيح غير سالب **n** يعرَّف كما يلي :

$$n ! = n . (n - 1) . (n - 2) . \ldots . 1 \qquad (n \geq 1) ,$$
$$n ! = 1 \quad (n = 0)$$

مثلا :

$$5 ! = 5 . 4 . 3 . 2 . 1 = 120$$

أ) اكتب برنامجا يقرأ عددا صحيحا غير سالب ويحسب ويطبع مضروبه.

ب) اكتب برنامجا يُقدِّر قيمة الثابت الرياضي **e** باستخدام الصيغة

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

ج) اكتب برنامجا يحسب قيمة e^x باستخدام الصيغة

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

وذلك عندما $x = 3$.

٣-٤٤) اكتب برنامجا يحسب ويطبع القيمة المتوسطة لمتتابعة من أعداد صحيحة ، بفرض أن آخر قيمة تقرأها **scanf** هي القيمة الحارسة 9999 . فمثلا متتابعة الإدخال (input sequence)

10 8 11 7 9 9999

تعني أن المطلوب حساب القيمة المتوسطة لجميع القيم التي تسبق 9999 .

٣ - ٤٥) 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 – ٤٦) 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 – ٤٧) تُقَدَّر زكاة عروض التجارة ZT برُبع العشر من القيمة السوقية (Market-Value) MV للعروض التجارية (والقيمة السوقية هي 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 :

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 – ٤٨) تعطَى بعض المعادن تقديرا يعتمد على نتائج اختبارات ثلاثة. وهذه الاختبارات 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
(أ)	١٠	إذا نجح في الاختبارات الثلاثة .
(ب)	٩	إذا نجح في الاختبارين (١) ، (٢) فقط .
(ج)	٨	إذا نجح في الاختبار (١) فقط .
(د)	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 – ٤٩) 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 الابن : $s = \frac{2r}{2NS+ND}$

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 A , NS , ND ويطبع قيم f, m, w, s, d .

٣ – ٥٠) اكتب برنامجا (يمكن أن يعد جزءا من برنامج عام لتوزيع الميراث) بحيث يقوم هذا البرنامج الجزئي بتحديد نصيب كل من الزوج H أو الزوجة W وكل واحد من الأبناء (نصيب الابن 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 – ٥١) – 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 إلى تكلفة الوقود لخمس سنين ، ثم أضف الضريبة لخمس سنين ، علما بأن الضريبة لسنة واحدة تحسب بضرب معدل الضريبة في التكلفة الأولية .

٣ – ٥٢) 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) .

٣–٥٣) تقوم مؤسسة اتصالات تليفونية بالتعامل مع العملاء في تسديد الفواتير على النحو التالي :

يُعرض خصم 50% للمكالمات التي تبدأ بعد 6:00 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 6:00p.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 – ٥٤) 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 ذو نقطة عائمة) ، ثم يطبع هذه البيانات وقيمة الفاتورة .

٣ – ٥٥) 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	١٢٠–٤٠	٢٠٠–١٢١	٣٠٠–٢٠١	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

$$ZF = \begin{cases} \frac{10}{100} * F & \text{– إذا كان الري طبيعيا (أي بالمطر) :} \\ \frac{5}{100} * F & \text{– إذا كان الري صناعيا (أي بالآلة) :} \\ \frac{7.5}{100} * F & \text{– إذا كان الري مزيجا من الطبيعي و الصناعي :} \end{cases}$$

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 – ٥٦) 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 إضافية للتحكم والتكرار
Additional Control and Repetition Statements

درسنا في الفصل السابق عبارة التحكم **if** وعبارة التكرار **while** ، وفي الفصل الحالي ندرس بإذن الله تعالى عبارات إضافية للتحكم والتكرار حيث ندرس :

- عبارة التكرار **for** .
- عبارة التكرار **do … while** .
- عبارة التحكم **switch** للاختيار المتعدد (multiple selection) .
- عبارة **break** للخروج (exiting) فورا وبسرعة من عبارات تحكم وتكرار معينة .
- عبارة **continue** لتَخَطِّي (skipping) بقية جسم عبارة تكرار (remainder of the body of a repetition st.) والانتقال إلى التكرير التالي في العروة (next iteration of the loop) .

كما نناقش في هذا الفصل بإذن الله المؤثرات المنطقية (logical operators) التي تستخدم للجمع بين الشروط .

التكرار المحكوم بعَدَّاد Counter – Controlled Repetition

أي تكرار محكوم بعَدَّاد يتطلب :

١- اسم متغير التحكم (control variable) / عدَّاد العروة (loop counter).
٢- القيمة الابتدائية (initial value) لمتغير التحكم .
٣- الزيادة (increment) / النقصان (decrement) الذي يُعدِّل به متغير التحكم كل دورة أثناء تنفيذ العروة .
٤- الشرط الذي يختبر الوصول إلى القيمة النهائية (final value) لمتغير التحكم (أي يختبر استمرارية العروة) .

مثال ٤ – ١ : اكتب برنامجا يستخدم عبارة **while** لطباعة الأعداد الصحيحة من 1 إلى 10 .

الحل :

```
/* Example 4.1
   Counter-controlled repetition */
#include <stdio.h>

/* function main begins program execution */
int main ()
{
   int counter = 1; /* initialization */


  while ( counter <= 10 ) { /* repetition condition */
      printf ( "%d\n", counter ); /* display counter */
     ++counter; /* increment */
  } /* end while */


  return 0; /* indicate program ended successfully */


} /* end function main */
```

```
1
2
3
4
5
6
7
8
9
10
```

- **العبارة الأولى في جسم الدالة** main **تحدد اسم متغير التحكم :** **counter** **، وتُعرِّفه على أنه عدد صحيح** integer **، وتحجز له حيِّزا في الذاكرة ، وتعطيه قيمة ابتدائية** **1** **. وهذا التعريف لا يُعَدُّ عبارة تنفيذية** (executable statement) .

وكان من الممكن أيضا تعريف **counter** **وإعطاؤه قيمة ابتدائية باستخدام العبارتين :**

```
int     counter;
counter = 1;
```

والعبارة الأولى (التعريف) لا تُعد عبارة تنفيذية ، بينما الثانية (الاسناد) عبارة تنفيذية . وسوف نستخدم الطريقتين لإعطاء قيم ابتدائية للمتغيرات .

- العبارة ++counter; /* increment */
تزيد قيمة عداد العروة بواحد 1 كل مرة تُنفَّذ فيها العروة .

- شرط استمرار العروة المذكور في عبارة **while** يختبر ما إذا كانت قيمة متغير التحكم أقل من أو تساوي 10 [وهي آخر قيمة يكون عندها الشرط متحققا (true)] . ولاحظ أن جسم عروة while هذه يُنفَّذ أيضا عندما يكون متغير التحكم مساويا 10. وتنتهي العروة عندما يتجاوز متغير التحكم القيمة 10(أي عندما تصبح قيمة **counter** مساوية 11).

- يمكننا اختصار البرنامج السابق بأن نعطي **counter** القيمة الابتدائية 0 ، ونستبدل بعبارة while العبارة التالية :

```
while ( ++counter <= 10 )
   printf ( "%d\n", counter );
```

وبهذه الطريقة نوفِّر عبارة ، وذلك لأن الزيادة (incrementing) يتم اجراؤها مباشرة في شرط **while** قبل اختبار الشرط . كما أننا لا نحتاج الآن إلى القوسين { , } حول جسم **while** لأن **while** تحتوي الآن على عبارة واحدة فقط . وبالطبع تحتاج البرمجة بهذه الكيفية المختصرة إلى بعض الخبرة والتدريب .

ملاحظة : نظرا لأن القيم ذوات النقطة العائمة قد تكون تقريبية فإن التحكم في العد – بالنسبة للعُرَى المحكومة بعدَّاد – باستخدام قيم ذوات نقطة عائمة قد يؤدي إلى قيم غير دقيقة للعداد (imprecise counter values) واختبارات غير دقيقة لانتهاء العروة . ولذلك يفضل استخدام قيم صحيحة (integer values) للتحكم في عد دورات العُرى .

عبارة التكرار for (for Repetition Statement)

تحتوي عبارة التكرار **for** على جميع التفاصيل الخاصة بالتكرار المحكوم بعداد . ولتوضيح ذلك وقوة عبارة **for** نعيد فيما يلي كتابة البرنامج السابق (برنامج مثال ٤ – ١) ولكن باستخدام عبارة **for** .

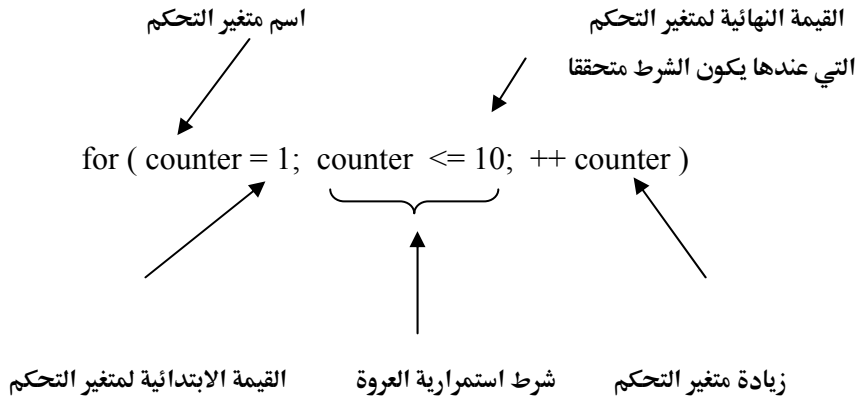
مثال ٤ – ٢ : اكتب برنامجا يستخدم عبارة **for** لطباعة الأعداد الصحيحة من 1 إلى 10 .

الحل :

```
/* Example 4.2
   Counter-controlled repetition with the for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter; /* define counter */

   /* initialization, repetition condition, and increment
      are all included in the for statement header. */
   for ( counter = 1; counter <= 10; counter++ ) {
      printf( "%d\n", counter );
   } /* end for */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

عندما يبدأ تنفيذ عبارة **for** فإن متغير التحكم **counter** يعطَى القيمة الابتدائية 1 . ثم يتم اختبار شرط استمرار العروة : counter <= 10 . ونظرا لأن القيمة الابتدائية للمتغير counter هي 1 ، فإن الشرط يكون متحققا ، وتقوم عبارة الطباعة **printf** بطباعة قيمة **counter** والتي هي 1 . ثم يقوم التعبير counter ++ بزيادة قيمة **counter** بواحد لتبدأ العروة مرة اخرى باختبار شرط استمرارها . ونظرا لأن قيمة **counter** الآن تساوي 2 فإن القيمة النهائية 10 لم يتم تجاوزها بعد ، ولذلك فإن البرنامج ينفذ عبارة الطباعة **printf** مرة أخرى . وهكذا تستمر هذه العملية إلى أن تُزاد قيمة متغير التحكم **counter** إلى قيمته النهائية 11 مما يؤدي إلى عدم تحقق شرط استمرار العروة وينتهي التكرار . ثم ينتقل البرنامج لتنفيذ أول عبارة بعد عبارة for ، وفي حالتنا هذه هي عبارة return الموجودة بنهاية البرنامج .

الشكل التالي يوضح تركيبة عبارة **for** التي تشتمل على كل ما يلزم للتكرار المحكوم بعداد (counter – controlled repetition) باستخدام متغير تحكم (a control variable) .



ملاحظة ١ : إذا كان هناك أكثر من عبارة واحدة في جسم عبارة **for** ، فإننا نستخدم قوسين : { } لتحديد جسم العروة .

ملاحظة ٢ : إذا كتبنا شرط استمرار العروة بطريق الخطأ counter < 10 بدلا من counter <= 10 ، فإن العروة ستُنفذ 9 مرات فقط ، وهذا من الأخطاء المنطقية العامة ويطلق عليه " خطأ عن التكرار الصحيح بواحد " (off – by – one error)

والصيغة العامة لعبارة **for** هي :

for (expression1; expression2; expression3) statement

حيث

التعبير expression1 : يعطي متغير التحكم في العروة قيمته الابتدائية .

التعبير expression2 : هو شرط استمرارية العروة .

التعبير expression3 : يزيد متغير التحكم .

وفي معظم الحالات فإن عبارة **for** يمكن تمثيلها بعبارة **while** مكافئة لها كما يلي :

```
expression1;
while ( expression2 ) {
    statement
    expression3;
}
```

وهناك استثناء لهذه القاعدة سنذكره بإذن الله فيما بعد في هذا الفصل .

ملاحظات على عبارة for

(١) قد يكون أي من التعبيرين expression1 , expression3 قائمة تعابير تفصل بينها فاصلات (comma – separated list of expressions) . مثلا قد يكون هناك متغيرا تحكم (two control variables) في عبارة **for** واحدة ، بحيث يجب إعطاؤهما قيمتين ابتدائيتين (initialized) ، وكذلك زيادتهما (incremented) . فيكون expression3 مثلا : i++ , j++

وفي مثل هذه الحالات تُستخدم الفاصلات كمؤثرات فواصل (comma operators) تضمن أن قائمة التعابير يتم إيجاد قيمها (evaluated) من اليسار إلى اليمين . وقيمة ونوع قائمة تعابير تفصل بينها فاصلات هي قيمة ونوع التعبير الموجود أقصى يمين القائمة . وغالبا ما يُستخدم مؤثر الفواصل (comma operator) في عبارة **for** . وفائدته الأساسية تمكين المبرمج من استخدام عدة تعابير لإعطاء قيم ابتدائية وكذلك عدة تعابير لزيادة المتغيرات (multiple initialization and/or multiple increment expressions) .

(٢) التعابير الثلاثة في عبارة **for** اختيارية (optional) . وإذا حُذف expression2 ، فإن لغة C تفترض أن الشرط متحقق (true) ، وبذلك تصبح العروة لا نهائية (infinite loop) . ويمكن حذف expression1 إذا تم إعطاء متغير التحكم قيمة ابتدائية في أي مكان آخر في البرنامج . وكذلك يمكن حذف expression3 إذا تم حساب الزيادة increment بعبارات في جسم عبارة **for** ، أو إذا لم نحتج إلى أي زيادة . وتعبير الزيادة في عبارة **for** يعمل عمل عبارة قائمة بذاتها في نهاية جسم عبارة **for** . ولذلك فإن التعابير

```
counter = counter + 1
counter += 1
++counter
counter++
```

جميعها متكافئة في جزء الزيادة (incrementing portion) في عبارة **for** . ويفضل كثير من المبرمجين الصيغة ++counter لأن الزيادة تحدث بعد تنفيذ جسم العروة ، وتبدو صيغة الزيادة اللاحقة (postincrementing form) طبيعية أكثر من غيرها . وصيغتا الزيادة السابقة والزيادة اللاحقة لهما التأثير نفسه .

(٣) الفاصلتان المنقوطتان في عبارة **for** ضروريتان . واستخدام فاصلة (comma) بدلا من فاصلة منقوطة (semicolon) يُعد خطأ تركيبيا (syntax error) . ووضع فاصلة منقوطة يمين عنوان / مقدمة / رأس عبارة for (for header) مباشرة (أي قبل جسم عبارة **for**) يجعل جسم عبارة **for** هذه عبارة خاوية (an empty statement) . وهذا يكون عادة خطأ منطقيا (a logic error) .

(٤) يمكن لأي من تعابير إعطاء القيمة الابتدائية (initialization) أو شرط استمرارية العروة (loop – continuation condition) أو الزيادة (increment) أن يحتوي على تعابير حسابية . مثلا إذا كانت x = 2 , y = 10 فإن العبارة

```
for ( j = x;  j <= 4 * x * y;  j += y / x )
```

تكافئ العبارة

```
for ( j = 2;  j <= 80;  j += 5 )
```

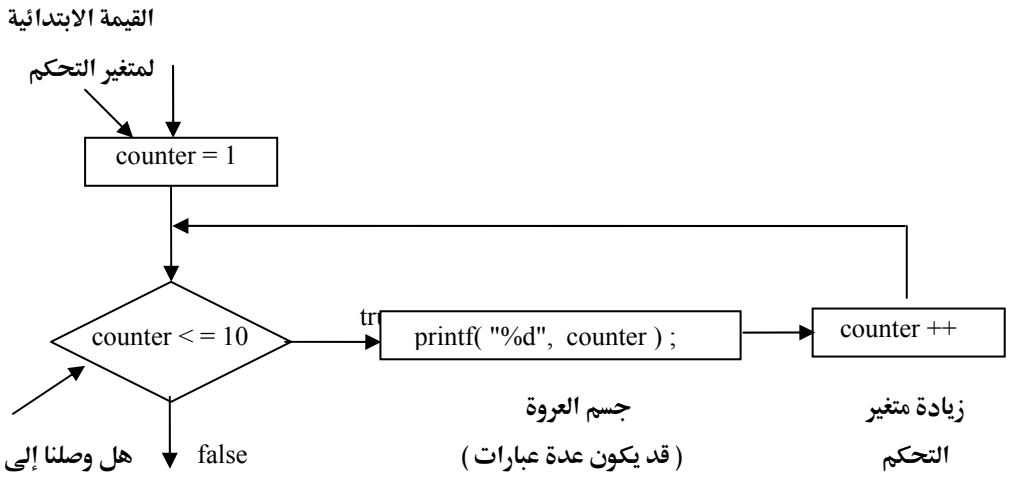
(٥) الزيادة قد تكون سالبة ، أي أنها في الحقيقة نقصان ، والعروة تتناقص تنازليا .

(٦) إذا كان شرط استمرار العروة غير متحقق (false) من البداية (initially) فإن جسم العروة لا ينفَّذ ، وينتقل التنفيذ إلى العبارة التي تلي عبارة for .

(٧) خريطة سير عمليات (flowchart) عبارة **for** تشبه إلى حد كبير تلك المقابلة لعبارة **while** . فمثلا عبارة **for** التالية :

```
for ( counter = 1; counter <= 10; counter++ )
    printf( "%d", counter );
```

تمثلها الخريطة المبينة بالشكل التالي :



القيمة النهائية
لمتغير التحكم ؟

(٨) رغم أن قيمة متغير التحكم يمكن تغييرها داخل جسم عروة **for** ، إلا أن ذلك قد يؤدي إلى أخطاء غير متوقعة ، ولذلك يفضل عدم تغييرها .

مثال ٤ – ٣ : الأمثلة التالية توضح طرق تغيير متغير التحكم في عبارة for .

أ) تغيير متغير التحكم من 1 إلى 100 بزيادة مضطردة (بخطوة ثابتة) تساوي 1.

```
for ( i = 1;  i<= 100;  i++ )
```

ب) تغيير متغير التحكم من 100 إلى 1 بزيادة ثابتة تساوي 1- (أي بنقصان ثابت يساوي 1) .

```
for ( i = 100;  i>= 1;  i-- )
```

ج) تغيير متغير التحكم من 7 إلى 77 بخطوة ثابتة 7 .

```
for ( i = 7;  i <= 77;  i += 7 )
```

د) تغيير متغير التحكم من 20 إلى 2 بخطوة ثابتة 2- .

```
for ( i = 20;  i >= 2;  i -= 2 )
```

هـ) تغيير متغير التحكم عبر تتابع القيم (sequence of values) :

```
2, 5, 8, 11, 14, 17, 20
for ( j = 2;  j <= 20;  j += 3 )
```

و) تغيير متغير التحكم عبر تتابع القيم :

```
99, 88, 77, 66, 55, 44, 33, 22, 11, 0
for ( j = 99;  j >= 0;  j -= 11 )
```

مثال ٤ – ٤ : اكتب برنامجا يقوم بحساب مجموع الأعداد الصحيحة الزوجية من 2 إلى 100 باستخدام عبارة **for** .

الحل :

```
/* Example 4.4
   Summation with for */
#include <stdio.h>

/* function main begins program execution */
```

```
int main()
{
  int sum = 0; /* initialize sum */
  int number;  /* number to be added to sum */

  for ( number = 2; number <= 100; number += 2 ) {
    sum += number; /* add number to sum */
  } /* end for */

  printf( "Sum is %d\n", sum ); /* output sum */

  return 0; /* indicate program ended successfully */

} /* end function main */
```

```
Sum is 2550
```

ملاحظة : جسم عبارة **for** في هذا البرنامج يمكن دمجه (merged) في الجزء الموجود أقصى اليمين في إعلان / مقدمة عبارة **for** (**for** header) باستخدام مؤثر الفواصل (comma operator) كما يلي :

```
for ( number = 2; number <= 100; sum += number, number +=  2 )
    ;  /* empty statement */
```

كما أن عبارة sum = 0 لإعطاء قيمة ابتدائية يمكن دمجها في قسم إعطاء القيم الابتدائية في عبارة **for** .

ورغم أن العبارات التي تسبق عبارة **for** ، والعبارات الموجودة في جسم عبارة **for** يمكن غالبا دمجها في إعلان / مقدمة عبارة **for** ، إلا أنه من الأفضل تجنب هذا الدمج لأنه يقلل من سهولة تتبُّع قراءة البرنامج بوضوح ويسر .

مثال ٤ – ٥ : رغم قوله تعالى : " يا أيها الذين آمنوا اتقوا الله وذروا ما بقي 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 .

نفرض أن شخصا ما يستثمر (invests) مبلغ 1000 دولار في حساب مدخرات (savings account) ربوي يعطي نسبة مئوية ربوية (usury percentage) يطلق

عليها " فائدة " (interest) [من باب التمويه والتحايل على الحرام ، حتى لا تبدو شيئا محرما بغيضا] تساوي 5% . وبفرض أن كل الربا الناتج / كل الفوائد تُترك في الحساب مضافة إلى المبلغ المودع (left on deposit in the account) ، احسب واطبع قيمة المبلغ الموجود بالحساب عند نهاية كل عام لمدة عشرة أعوام ، مستخدما الصيغة التالية لحساب قيم هذه المبالغ من الربا المركّب [الفائدة المركَّبة (compound interest)]

$$a = p (1 + r)^n$$

حيث :

p : المبلغ الأصلي المستثمر ، أي رأس المال (principal) .

r : النسبة الربوية السنوية [معدل الفائدة] (rate) .

n : عدد السنوات (number of years) .

a : كمية المبلغ المودع (amount of deposit) عند نهاية السنة رقم n .

الحل :

```
/* Example. 4.5
   Calculating compound usury / interest */
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main()
{
   double amount;               /* amount on deposit */
   double principal = 1000.0; /* starting principal */
   double rate = .05;           /* annual interest rate */
   int year;                    /* year counter */

   /* output table column head */
   printf( "%4s%21s\n", "Year", "Amount on deposit" );

   /* calculate amount on deposit for each of ten years */
   for ( year = 1; year <= 10; year++ ) {

      /* calculate new amount for specified year */
      amount = principal * pow( 1.0 + rate, year );
```

```
   /* output one table row */
   printf( "%4d%21.2f\n", year, amount );
  } /* end for */

  return 0; /* indicate program ended successfully */

} /* end function main */
```

```
Year        Amount    on    deposit
   1                          1050.00
   2                          1102.50
   3                          1157.63
   4                          1215.51
   5                          1276.28
   6                          1340.10
   7                          1407.10
   8                          1477.46
   9                          1551.33
  10                          1628.89
```

تقوم عبارة **for** بتنفيذ جسم العروة 10 مرات ؛ وذلك بتغيير متغير تحكم (a control variable) من 1 إلى 10 بزيادة ثابتة تساوي 1 . ورغم أنه لا يوجد مؤثر أسّ (exponentiation operator) في لغة C ، إلا أنه يمكننا استخدام الدالة المكتبية القياسية **pow** لهذا الغرض ، حيث تحسب الدالة pow (x , y) قيمة **x** مرفوعة للأس **y** . والدالة تأخذ وسيطين (2 arguments) من النوع **double** ، وتعيد قيمة من النوع **double** . والنوع **double** هو نوع ذو نقطة عائمة مثل **float** ولكن المتغير الذي نوعه **double** يمكنه تخزين قيمة (value) مقدارها أكبر بكثير (much greater magnitude) ودقتها (precision) أكبر من **float** . ولاحظ أن ملف المقدمة **math.h** يجب أن يحتوي عليه أي برنامج يستخدم دالة رياضية مثل **pow** . وكما ذكرنا فإن الدالة **pow** تأخذ وسيطين **double** . ولاحظ أن **year** عدد صحيح **integer** . والملف **math.h** يحتوي على معلومات تخبر البرنامج المترجِم (compiler) أن يقوم بتحويل قيمة **year** إلى تمثيل **double** مؤقت (a temporary double representation) قبل استدعاء الدالة. وهذه المعلومات موجودة فيما يطلق عليه " نموذج الدالة **pow** " (**pow**'s function prototype) . وسنناقش بإذن الله تعالى في

الفصل القادم " نماذج الدوال " . وسنعرض أيضا في الفصل القادم بإذن الله ملخصا للدالة **pow** ودوال مكتبية رياضية أخرى .

ولاحظ أننا عرَّفنا المتغيرات **amount, principal, rate** على أنها من النوع **double** وذلك لأننا نتعامل مع كميات فيها أجزاء كسرية من الدولارات .

وأما محدِّد التحويل 21.2f % الذي ورد في البرنامج فإنه يُستخدم لطباعة قيمة المتغير **amount** في مجال عرضه (field width) 21 موضع طباعة (print positions) ، والرقم 2 يمين النقطة يحدد الدِّقة (precision) ، أي عدد المواضع العشرية (decimal positions) ، أي عدد الأرقام يمين الفاصلة العشرية . وإذا كان عدد الرموز (characters) التي ستُطبع أقل من عرض المجال فإن القيمة تطبع تلقائيا أقصى اليمين (right justified) في المجال المحدد . وإذا أردنا طباعتها أقصى اليسار (left justified) فإننا نضع إشارة ناقص (-) بين علامة ٪ وعرض المجال . وهذه الإشارة السالبة يمكن أيضا أن تستخدم لطباعة الأعداد الصحيحة أقصى اليسار (مثل 6d - ٪) أو لطباعة سلاسل الرموز أقصى اليسار (مثل 8s - ٪) . وسنناقش بإذن الله تعالى في الفصل الأخير بعض التفاصيل الخاصة بقدرات كل من **printf, scanf** على صياغة المدخلات والمخرجات .

عبارة switch للاختيار المتعدِّد switch Multiple – Selection Statement

درسنا في الفصل السابق كلا من عبارة **if** للاختيار المفرد ، وعبارة **if ... else** للاختيار المزدوج . وأحيانا تحتوي خوارزمية مسألة ما على عدة اختيارات أو سلسلة قرارات (series of decisions) نختبر (test) فيها متغيرا ما أو تعبيرا ما بصورة منفصلة (separately) لكل قيمة من القيم التكاملية الثابتة (constant integral values) التي يمكن أن يأخذها ، وننفِّذ بناءً على ذلك أفعالا أو أوامر مختلفة (different actions) ويُطلَق على هذا " الاختيار المتعدد " (multiple selection) . وتُستخدم عبارة **switch** للاختيار المتعدد لاتخاذ قرار في مثل هذه الحالات .

وتتكون عبارة **switch** من سلسلة من عناوين الحالات (series of case labels) ، وحالة **default** اختيارية (optional) .

مثال ٤ – ٦ : اكتب برنامجا يقرأ عددا من التقديرات الحرفية (letter grades) لطلاب أحد الفصول ، ويستخدم عبارة **switch** لحساب أعداد الطلاب الحاصلين على كل من التقديرات المختلفة A, B, C, D, F . وتنتهي المدخلات حين يُدخل المستخدِم رمز نهاية الملف EOF character (end of file) .

الحل :

```
/* Example 4.6
   Counting letter grades */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int grade;      /* one grade */
   int aCount = 0; /* number of As */
   int bCount = 0; /* number of Bs */
   int cCount = 0; /* number of Cs */
   int dCount = 0; /* number of Ds */
   int fCount = 0; /* number of Fs */

   printf(  "Enter the letter grades.\n"  );
   printf(  "Enter the EOF character to end input.\n"  );

   /* loop until user types end-of-file key sequence */
   while ( ( grade = getchar() ) != EOF ) {

      /* determine which grade was input */
      switch ( grade ) { /* switch nested in while */

         case 'A': /* grade was uppercase A */
         case 'a': /* or lowercase a */
            ++aCount; /* increment aCount */
            break; /* necessary to exit switch */

         case 'B': /* grade was uppercase B */
         case 'b': /* or lowercase b */
            ++bCount; /* increment bCount */
            break; /* exit switch */

         case 'C': /* grade was uppercase C */
         case 'c': /* or lowercase c */
            ++cCount; /* increment cCount */
            break; /* exit switch */

         case 'D': /* grade was uppercase D */
         case 'd': /* or lowercase d */
```

```
            ++dCount; /* increment dCount */
            break; /* exit switch */

         case 'F': /* grade was uppercase F */
         case 'f': /* or lowercase f */
            ++fCount; /* increment fCount */
            break; /* exit switch */

         case '\n': /* ignore newlines, */
         case '\t': /* tabs, */
         case ' ':  /* and spaces in input */
            break; /* exit switch */

         default: /* catch all other characters */
            printf( "Incorrect letter grade entered." );
            printf( " Enter a new grade.\n" );
            break; /* optional; will exit switch anyway */
      } /* end switch */

   } /* end while */

   /* output summary of results */
   printf( "\nTotals for each letter grade are:\n" );
   printf( "A: %d\n", aCount ); /* display number of A grades */
   printf( "B: %d\n", bCount ); /* display number of B grades */
   printf( "C: %d\n", cCount ); /* display number of C grades */
   printf( "D: %d\n", dCount ); /* display number of D grades */
   printf( "F: %d\n", fCount ); /* display number of F grades */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
Enter the letter grades.
Enter the EOF character to end input.
a
b
c
C
A
d
f
C
E
```

```
Incorrect letter grade entered. Enter a new grade.
D
A
b
^Z

Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1
```

نلاحظ في مقدمة عبارة **while**

```
while ( ( grade = getchar () ) != EOF )
```

أن عبارة الإسناد المحاطة بقوسين (grade = getchar()) تُنفذ أولا ، حيث تقوم الدالة (getchar) [من المكتبة القياسية للمدخلات والمخرجات / standard input output Library] بقراءة رمز من لوحة المفاتيح وتخزينه في متغير صحيح (integer variable) **grade** . وعادة تُخَزَّن الرموز في متغيرات من النوع **char** . إلا أنه من الخصائص الهامة للغة C أنه يمكن تخزين الرموز في أي نوع بيانات صحيحة (any integer data type) وذلك لأن الرموز تمثَّل في الحاسوب كأعداد صحيحة أحادية البايتات (one-byte integers) . وبالتالي يمكننا معالجة (treating) / التعامل مع أي رمز إما على أنه عدد صحيح (integer) أو على أنه رمز (character) تبعا لاستخدامه . فمثلا العبارة

```
printf( " The character (%c) has the value %d .\n" , 'a', 'a' );
```

تستخدم محدِّدي التحويل %c, %d لطباعة الرمز **a** وقيمته الصحيحة (integer value) . ونتيجة تنفيذ هذه العبارة هي :

```
The character (a) has the value 97.
```

حيث أن العدد الصحيح 97 هو التمثيل العددي (numerical representation) للرمز a(character) في الحاسوب . وكثير من الحواسيب اليوم تستخدم مجموعة الرموز ASCII [وهي الشفرة الأمريكية القياسية لتبادل المعلومات American Standard Code) (for Information Interchange] . وفي هذه المجموعة العدد 97 يمثِّل الحرف الصغير 'a' (lower case letter) . وفي الملحق (ب) بنهاية الكتاب توجد قائمة برموز ASCII وقيمها

العشرية (decimal values). ويمكن قراءة الرموز باستخدام **scanf** مع محدِّد التحويل %c.

وأي عبارة إسناد – في مُجملها (as a whole) – لها قيمة . وهذه القيمة هي تلك التي تُسند إلى المتغير الموجود يسار علامة التساوي = . فمثلا قيمة (value) تعبير الإسناد (assignment expression) grade = getchar() هي الرمز الذي تعيده (returned by) **getchar** ويُسند إلى المتغير **grade** .

ويمكننا الاستفادة من تلك الحقيقة – وهي أن عبارات الإسناد لها قيم – في إعطاء عدة متغيرات القيمة الابتدائية نفسها (same initial value) . فمثلا التعبير

a = b = c = 0;

يحسب أولا قيمة عبارة الإسناد c = 0 [وذلك لأن المؤثِّر = يتم تجميعه (associates) من اليمين إلى اليسار] ، وهذه القيمة تساوي صفرا 0 . ثم تُسنَد هذه القيمة إلى المتغير b . ثم تُسنَد إلى المتغير a قيمة عبارة الإسناد (b = (c = 0) [والتي تساوي صفرا 0 أيضا] .

والبرنامج السابق يقارن بين قيمة عبارة الإسناد grade = getchar() وقيمة EOF التي تستخدمها كقيمة حارسة (sentinel value) [والتي عادة تساوي 1-] . والمستخدِم يقوم بطباعة توافق من مفاتيح لوحة المفاتيح يعتمد على النظام المستخدَم (system – dependent keystroke combination) ليعني EOF أي نهاية الملف (end of file)، أي أنه ليس عندي بيانات أخرى لإدخالها . و EOF يُعَدُّ ثابتا صحيحا رمزيا (a symbolic integer constant) معرَّفاً في ملف المقدمة (header) < stdio.h > . فإذا كانت القيمة المُسنَدة إلى **grade** مساوية لـ EOF ، فإن البرنامج يتوقف . وقد اخترنا أن نمثِّل الرموز (represent characters) في هذا البرنامج كأعداد صحيحة **ints** لأن EOF له قيمة صحيحة (integer value) [وهي – مرة أخرى – عادة تساوي 1- ، ولكن قد تختلف من نظام لآخر ، وفي لغة C القياسية (C standard) EOF له قيمة تكاملية / صحيحة سالبة ولكن ليس بالضرورة 1-] . واختيار الثابت الرمزي EOF (بدلا من قيمة معينة مثل 1-) عند اختبار نهاية البيانات يجعل البرنامج أكثر مرونة للاستخدام (more portable program) في أكثر من نظام .

وفي نظم UNIX ونظم أخرى كثيرة يتم إدخال EOF بطباعة المتتابعة (sequence) :

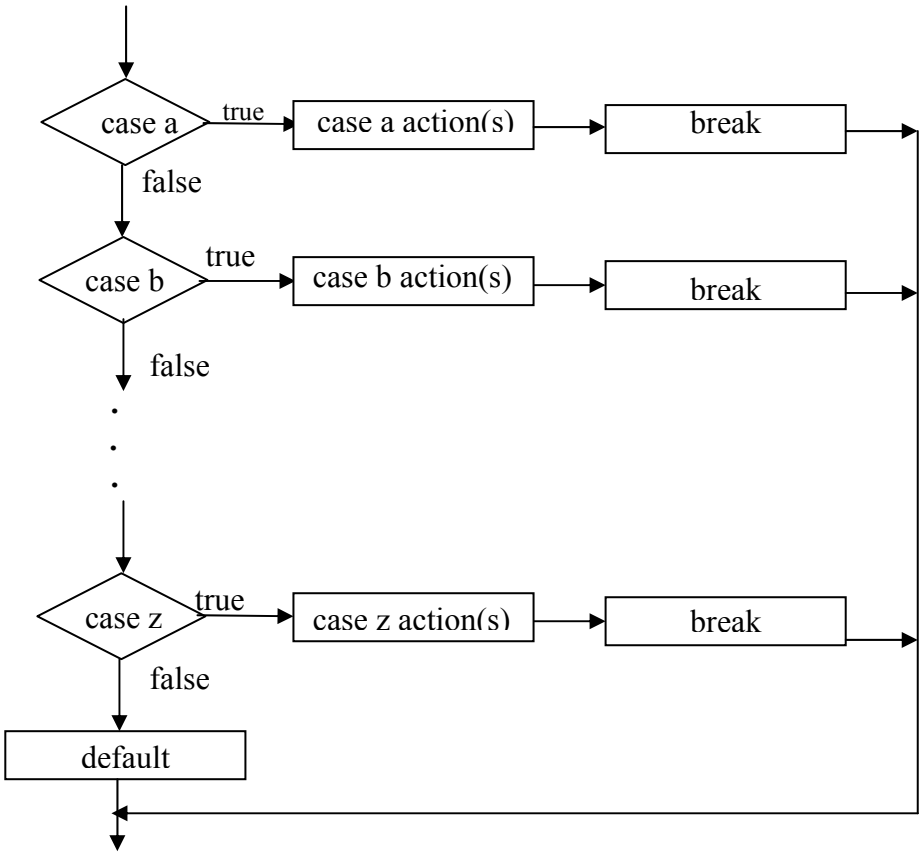
<return> <ctrl-d>

وهذا الاصطلاح (notation) يعني الضغط أولا على مفتاح return ، ثم الضغط – آنيا (simultaneously) ، أي في الوقت نفسه – على مفتاحي ctrl, d . وفي نظم أخرى مثل " نوافذ مايكروسوفت "(Microsoft Corporation's Windows) يمكن إدخال المؤشر (indicator) EOF بطباعة

<ctrl-z>

وفي البرنامج السابق يقوم المستخدِم بإدخال التقديرات من لوحة المفاتيح . وعندما يتم الضغط على مفتاح **Return** (أو **Enter**) تقوم الدالة **getchar** بقراءة الرموز (characters) رمزاً رمزا . فإذا لم يكن الرمز المُدخَل مساويا EOF ، فإنه يتم دخول عبارة **switch** . ونلاحظ أن كلمة **switch** يليها اسم المتغير **grade** بين قوسين . وهذا يُطلَق عليه تعبير التحكم (controlling expression). وقيمة هذا التعبير تقارَن مع كل واحدة من قيم/عناوين الحالات (case labels). فإذا فرضنا مثلا أن المستخدِم قد أدخل الحرف C كتقدير (grade) ، فإن التقدير C يقارَن تلقائيا (automatically) مع كل **case** في عبارة **switch** . فإذا حدث توافق (match) [case ' C ' :] فإن العبارات المقابلة لهذه الحالة يتم تنفيذها . وفي حالة إدخال الحرف C ، فإن قيمة **cCount** تُزاد بواحد **1**، ثم يتم الخروج فورا من عبارة **switch** بعبارة **break** . وعبارة **break** تؤدي إلى انتقال التحكم في البرنامج (program control) إلى أول عبارة بعد عبارة **switch** . وترجع أهمية استخدام عبارة **break** إلى أنه بدون استخدامها فإن الحالات المختلفة في عبارة **switch** ستعمل جميعها معاً ! وإذا لم نستخدم **break** في أي موضع في عبارة **switch** فإنه حينما يحدث توافق في العبارة في أي مرة ، فإن العبارات المقابلة لجميع الحالات **cases** المتبقية (أي المكتوبة أسفل حالة التوافق) سيتم تنفيذها . وإذا لم يحدث أي توافق (match) ، فإن حالة **default** تُنفَّذ ، وتُطبع " رسالة خطأ " (error message) .

وكل حالة **case** يمكن أن يقابلها أمر تنفيذي واحد أو أكثر . وتختلف عبارة **switch** عن جميع عبارات التحكم الأخرى في أنها لا تحتاج إلى قوسين { , } حول الأوامر المتعددة (multiple actions) المقابلة لأي **case** في عبارة **switch**. والشكل التالي يوضح خريطة سير العمليات المقابلة لعبارة **switch** العامة للاختيار المتعدد ، والتي تستخدم **break** مع كل **case**.



توضح الخريطة أن أي عبارة **break** في نهاية أي **case** تؤدي بالتحكم إلى الخروج فورا من عبارة **switch** .

ويلاحظ في عبارة **switch** أن ترتيب الحالات **case** المختلفة والحالة الافتراضية **default** اختياري ، ولكن المعتاد وضع حالة **default** في النهاية ، وحينئذ لا يكون من الضروري وضع **break** مع **default** ، ولكن البعض يفضل وضعها زيادة في الإيضاح وللتماثل مع الحالات الأخرى .

وفي عبارة **switch** في البرنامج السابق نلاحظ أن السطور

```
case '\n': /* ignore newlines, */
case '\t': /* tabs, */
case ' ':  /* and spaces in input */
        break; /* exit switch */
```

تجعل البرنامج يتخطى (skip) رموز السطر الجديد (newline) والفراغ(blank) . والسبب في إضافة هذه السطور هو أن قراءة الرموز رمزا رمزا قد تسبب بعض المشاكل : فكي نجعل البرنامج يقرأ الرموز ، يجب إرسال الرموز إلى الحاسوب عن طريق الضغط على مفتاح **Return** . وهذا يؤدي إلى وضع رمز السطر الجديد (newline character) في المدخلات بعد الرمز الذي نريد تشغيله (processing) . وغالبا نضطر إلى تشغيل رمز السطر الجديد هذا بطريقة خاصة حتى نجعل البرنامج يعمل بطريقة صحيحة ويؤدي المطلوب منه . فإذا أضفنا السطور / الحالات السابقة في عبارة **switch** فإننا نمنع بذلك طباعة رسالة الخطأ (error message) الموجودة في حالة **default** كلما قابلنا رمز السطر الجديد أو الفراغ في المدخلات . فعدم تشغيل رموز السطر الجديد الموجودة في المدخلات حين قراءة الرموز رمزا رمزا قد يؤدي إلى أخطاء منطقية .

ووَضْع عدة عناوين حالة (several case labels) معاً مثل :

```
case ' D ' : case ' d ' :
```

في البرنامج السابق يعني ببساطة تنفيذ مجموعة الخطوات نفسها في أي من هذه الحالات .

وعبارة **switch** لا تستخدم إلا عند اختبار تعبير صحيح / تكاملي ثابت (testing a constant integral expression) ، أَيْ تعبير مكوَّن من أَيِّ توافق (combination) من ثوابت رمزية (character constants) وثوابت صحيحة (integer constants) بحيث يكون للتعبير قيمة صحيحة ثابتة (constant integer value) . وأي ثابت رمزي يمثَّل بهذا الرمز المعيَّن محاطا بحاصرتين مفردتين (single quotes) ، مثل ' A ' . فأي رمز يجب أن يحاط بحاصرتين مفردتين للتعرف عليه كثابت رمزي . وأما الثوابت الصحيحة فهي ببساطة قيم (أعداد) صحيحة (integer values) . وفي مثالنا السابق (مثال ٤-٦) استخدمنا ثوابت رمزية . وتذكَّر أن الرموز هي فعليا قيم صحيحة صغيرة (small integer values) .

واللغات القابلة للنقل (portable languages) كلغة C يجب أن تحتوي على سعات متنوعة لأنواع البيانات (flexible data type sizes) . فبعض التطبيقات قد تتطلب أعدادا صحيحة ذوات سعات مختلفة . ولغة C تحتوي على أنواع متعددة للبيانات لتمثيل الأعداد الصحيحة . ويعتمد مدى القيم الصحيحة في كل نوع على المكوِّنات المادية (hardware) للحاسوب المستخدَم . وبالإضافة إلى النوعين **int , char** فإن لغة C تحتوي أيضا على النوعين **short** (وهو اختصار **short int**) و **long** (وهو اختصار **long int**) . ولغة C تقرر أن أصغر مدى (minimum range) لقيم **short** هو 32767± . وبالنسبة للغالبية العظمى من حسابات

الأعداد الصحيحة فإن الأعداد الصحيحة **long** تكون كافية . وأما لغة C القياسية (standard) فإنها تقرر أن أصغر مدى لقيم **long** هو 2147483647± ، وأن مدى قيم عدد **int** هو على الأقل مدى الأعداد الصحيحة **short** نفسه ، وعلى الأكثر مدى الأعداد الصحيحة **long** . ويمكن استخدام نوع البيانات **char** لتمثيل الأعداد الصحيحة الواقعة في المدى 127±، وكذلك أي رمز (character) في مجموعة رموز الحاسوب (computer's character set) .

عبارة التكرار (Repetition Statement) do … while

عبارة **do..while** تشبه عبارة **while** ، والفارق بينهما هو أن شرط استمرارية العروة (loop-continuation condition) يُختبر في عبارة **while** عند بداية العروة قبل تنفيذ جسم العروة ، وأما في عبارة **do..while** فيُختبر بعد تنفيذ جسم العروة . أي أنه في حالة عبارة **do..while** يجب أن ينفَّذ جسم العروة مرة واحدة على الأقل ، وعندما تنتهي العروة ينتقل التنفيذ إلى العبارة التي تلي جملة (clause) **while**. وكذلك إذا كانت هناك عبارة واحدة فقط في جسم عروة **do..while** فليس من الضروري وضع قوسين { } يحيطان بالعبارة ، ولكننا عادة نضعهما لتجنب أي التباس بين عبارة **while** وعبارة **do..while**. فمثلا

```
while ( condition )
```

تُعتبر عادةً عنوانا / مقدمة (header) لعبارة **while**. وعبارة **do..while** بدون قوسين حول العبارة الوحيدة في جسم العروة ستبدو هكذا

```
do
    statement
while ( condition );
```

وقد يُحدث هذا التباسا ، حيث أن آخر سطر `while ( condition );` ربما يفسره القارئ على أنه عبارة **while** تحتوي على عبارة خاوية (empty statement) . ولذلك فإن **do..while** ذات العبارة الواحدة تُكتب عادة هكذا :

```
do {
    statement
} while ( condition );
```

مثال ٤ – ٧ : اكتب برنامجا يستخدم عبارة **do..while** لطباعة الأعداد الصحيحة من 1 إلى 10 .

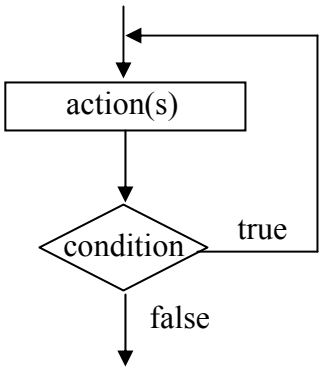
الحل :

```
/* Example. 4.7
   Using the do/while repetition statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter = 1;              /* initialize counter */

   do {
      printf( "%d  ", counter ); /* display counter */
   } while ( ++counter <= 10 ); /* end do...while */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
1 2 3 4 5 6 7 8 9 10
```

لاحظ أن متغير التحكم counter يُزاد مقدماً/سابقاً (preincremented) [++ counter] في اختيار استمرارية العروة . كذلك لاحظ وجود القوسين حول العبارة الوحيدة في جسم عروة do...while .

الشكل التالي يبين خريطة سير العمليات لعبارة do...while ، ومنها يتضح أن شرط استمرارية العروة لا يُختبر إلا بعد تنفيذ عبارات / أوامر (actions) جسم العروة مرة واحدة على الأقل .



عبارتا break و continue

تستخدم عبارتا **break** و **continue** لتغيير مسار التحكم . وعند تنفيذ عبارة **break** الموجودة داخل احدى العبارات / العُرى :

while, for, do…while, switch

فإن ذلك يؤدي إلى الخروج الفوري من تلك العبارة / العروة . ويستمر تنفيذ البرنامج ابتداءً من العبارات التالية . ومن الاستخدامات الشائعة لعبارة **break** الهروب (escaping) مبكرا من عروة ، أو تخطي (skipping) بقية عبارة **switch** (كما رأينا في مثال ٤ – ٦) . والمثال التالي يوضح استخدام عبارة **break** للهروب من عروة **for** .

مثال ٤ – ٨ : في البرنامج التالي عندما تكتشف عبارة **if** أن قيمة **x** أصبحت **5** ، يتم تنفيذ عبارة **break** ، وهذا ينهي عبارة **for** ، وينتقل البرنامج إلى تنفيذ عبارة **printf** الموجودة بعد عبارة **for** . ويتم تنفيذ العروة كاملة أربع مرات فقط .

```
/* Example. 4.8
   Using the break statement in a for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x; /* counter */

   /* loop 10 times */
   for ( x = 1; x <= 10; x++ ) {

      /* if x is 5, terminate loop */
      if ( x == 5 ) {
         break; /* break loop only if x is 5 */
      } /* end if */

      printf( "%d ", x ); /* display value of x */
   } /* end for */

   printf( "\nBroke out of loop at x == %d\n", x );

   return 0; /* indicate program ended successfully */
```

```
} /* end function main */
```

```
1 2 3 4
Broke out of loop at x == 5
```

وأما عبارة continue والتي قد تكون موجودة داخل احدى عبارات التكرار

while, for, do…while

فإن تنفيذها يـؤدي إلى تخطي (skipping) العبارات المتبقية في جسم تلك العروة / عبارة التكرار / ثم الانتقال لتنفيذ التكرير التالي(next iteration) في العروة . وفي كل من عبارتي while , do…while يُختبر شرط استمرارية العروة فورا بعد تنفيذ عبارة continue . وفي عبارة for يُنفذ تعبير الزيادة (increment expression) ، ثم يُختبر شرط استمرارية العروة . وقد ذكرنا سابقا أنه يمكن استخدام عبارة while في معظم الحالات لتمثيل عبارة for . والاستثناء الوحيد لذلك هو عندما يتبع تعبير الزيادة في عبارة while عبارة continue . ففي هذه الحالة لا تنفَّذ الزيادة قبل اختبار شرط استمرارية التكرار ، وبالتالي لا تُنفَّذ عبارة while بالكيفية نفسها كعبارة for .

مثال ٤ – ٩ : البرنامج التالي يستخدم عبارة continue داخل عبارة for لتخطي عبارة printf ثم الانتقال لتنفيذ التكرير التالي في العروة .

```
/* Example 4.9
   Using the continue statement in a for statement */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x; /* counter */

   /* loop 10 times */
   for ( x = 1; x <= 10; x++ ) {

      /* if x is 5, continue with next iteration of loop */
      if ( x == 5 ) {
         continue; /* skip remaining code in loop body */
      } /* end if */
```

```
   printf( "%d ", x ); /* display value of x */
  } /* end for */

  printf( "\nUsed continue to skip printing the value 5\n" );

  return 0; /* indicate program ended successfully */

} /* end function main */
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

المؤثرات المنطقية (Logical Operators)

درسنا حتى الآن شروطا بسيطة فقط مثل :

counter < = 10, total > 1000, number ! = sentinel Value

وقد كتبنا هذه الشروط بدلالة المؤثرات العلاقية :

<, <=, >, >= ومؤثّري التساوي : = ! ,== . وكل منها يمثّل شرطا واحدا بالضبط . فإذا أردنا اختبار عدة شروط (multiple conditions) في عمليةٍ ما لاتخاذ قرار ، فأننا نجري هذه الاختبارات بعبارات منفصلة ، أو بعبارات **if** أو **if…else** متداخلة .

وفي لغة C يمكننا أيضا استخدام ما يطلق عليها المؤثرات المنطقية لتكوين شروط مركبة (complex conditions) وذلك بالجمع بين شروط بسيطة . وهذه المؤثرات المنطقية هي :

&&: " و " المنطقية (logical AND)

|| : " أو " المنطقية (logical OR)

! : ليس المنطقية (logical Not) ، ويطلق عليها أيضا :

" النفي " المنطقي (logical negation)

وفيما يلي أمثلة لهذه المؤثرات المنطقية .

مثال ٤ – ١٠ :

(i) نفرض أن لدينا بيانات مجموعة من الطلاب والطالبات وتقديراتهم / درجاتهم (grades) . ونفرض أننا نود إيجاد عدد الطالبات المتفوقات / الممتازات وهن الحاصلات على درجة أكبر من أو تساوي 90 . لحساب هذا العدد علينا أن نضمن تحقق شرطين معاً وهما :

أ) جنس (gender) الشخص / الطالب : أنثى (female) .

ب) درجة الطالب / الطالبة ≥ 90 .

يمكننا استخدام المؤثر && في عبارة مثل :

```
if ( gender == 1 && grade >= 90 )
    ++excellentGirlStudents;
```

فعبارة **if** هنا تحتوي على شرطين بسيطين :

أ) الشرط gender == 1 يمكننا إيجاد قيمته لتحديد ما إذا كان الشخص أنثى أم لا .

ب) الشرط grade > = 90 يمكننا إيجاد قيمته لتحديد ما إذا كانت الدرجة قد بلغت (أو زادت عن) 90 أم لا ، أي ما إذا كان الشخص ممتازا أم لا .

والبرنامج يوجد أولا قيمتي الشرطين البسيطين ، وذلك لأن كلاً من = < , == أعلى أولوية من && . ثم يوجد قيمة الشرط المركب

gender == 1 && grade >= 90

هذا الشرط يكون صحيحا (true) أي متحققا إذا وفقط إذا كان كل من الشرطين البسيطين صحيحا . وأخيرا .. إذا كان هذا الشرط المركب صحيحا فإن عدد (count) الطالبات الممتازات يُزاد بواحد . وأما إذا كان أي من الشرطين أو كلاهما خاطئا (false) أي غير متحقق فإن البرنامج يتخطى أمر الزيادة ، وينتقل إلى العبارة التي تلي عبارة **if** .

الجدول التالي يلخص المؤثر && . والجدول يبين جميع التوافقات الأربعة الممكنة للقيم الصفرية (zero) [أي خاطئ (false)] وغيرالصفرية (nonzero) [أي صحيح (true)] للتعبيرين expression1, expression2 . وعادة يطلَق على مثل هذه الجداول " جداول الصحة " **(truth tables)** . ولغة C توجد قيم جميع التعابير التي تحتوي على مؤثرات علاقية ، ومؤثري التساوي ، والمؤثرات المنطقية بحيث تكون هذه القيم إما 0 أو 1 . ورغم أن لغة C تجعل القيمة true تقابل / تساوي القيمة 1 ، إلا أنها تقبل أي قيمة غير صفرية على أنها true .

expression1	expression2	expression1 && expression2
0	0	0
0	قيمة غير صفرية (nonzero)	0
قيمة غير صفرية (nonzero)	0	0

1	قيمة غير صفرية (nonzero)	قيمة غير صفرية (nonzero)

جدول الصحة للمؤثر المنطقي &&

Truth table for the && (logical AND) operator

مثال ٤ – ١١ : نفرض أن الطالب يعطَي التقدير A إذا تحقق أحد الشرطين التاليين أو كلاهما :

(أ) الشرط : الدرجة المتوسطة للفصل الدراسي ≥ 90 .

(ب) الشرط : درجة الاختبار النهائي ≥ 90 .

يمكننا استخدام المؤثر || في عبارة مثل :

```
if ( semesterAverage >= 90 || finalExam >= 90 )
    printf( " Student grade is A\n" );
```

ونلاحظ أن الرسالة " Student grade is A " لا تطبع في حالة واحدة فقط ، وهي عندما يكون كلا الشرطين (البسيطين) خاطئين (false) [أي صفرين (zero)] .

وفيما يلي جدول الصحة للمؤثر OR المنطقي (||) :

expression1	expression2	expression1 || expression2
0	0	0
0	قيمة غير صفرية (nonzero)	1
قيمة غير صفرية (nonzero)	0	1
قيمة غير صفرية (nonzero)	قيمة غير صفرية (nonzero)	1

جدول الصحة للمؤثر المنطقي ||

Truth table for the logical OR (||) operator

ويلاحظ أن أولوية المؤثر && أعلى من أولوية المؤثر || . ويتم تجميع (associativity) أي من هذين المؤثرين من اليسار إلى اليمين . وأي تعبير يحتوي على أي من هذين المؤثرين يتم تقييمه بطريقة التقييم السريع / قصير الدائرة (short-circuit evaluation) ، وهي أن يتوقف تقييم التعبير بمجرد الوصول إلى قيمته النهائية : صحيح (true / خاطئ (false) . فمثلا تقييم الشرط

```
gender == 1 && grade >= 90
```

يتوقف إذا لم يكن gender مساويا 1 [فالتعبير الكلي تكون قيمته حينئذ **false** ، ولا نحتاج لتقييم / لاختبار الشرط grade >= 90] ، ويستمر إذا كان gender مساويا 1 [حيث أن التعبير الكلي يمكن أن يكون true ، وذلك إذا كان grade >= 90] .

مثال ٤ – ١٢ : نفرض أننا نود إدخال وطباعة متتابعة من التقديرات grades إلى أن ندخل تقديرا مساويا sentinelValue ليعني إنهاء هذه المتتابعة . يمكننا كتابة عبارة if التالية حيث يستخدم شرطها مؤثر النفي المنطقي ! :

```
if ( ! ( grade == sentinelValue ) )
   printf ( " The next grade is %f\n", grade );
```

نلاحظ أن المؤثر ! يؤثر على شرط واحد فقط ، ولذلك يطلق عليه " مؤثر أحادي " (unary operator) ، بينما أي من || ,&& يؤثر على شرطين ، ولذلك يطلق على أي منهما " مؤثر ثنائي " (binary operator) . كما نلاحظ أن القوسين حول الشرط grade == sentinelValue ضروريان وذلك لأن أولوية المؤثر ! أعلى من أولوية مؤثر التساوي == . وعبارة if السابقة تكافئ عبارة if التالية :

```
if ( grade ! = sentinelValue )
printf ( " The next grade is %f\n", grade );
```

والجدول التالي هو جدول الصحة لمؤثر النفي المنطقي :

expression	expression
0	1
nonzero (قيمة غير صفرية)	0

جدول الصحة لمؤثر النفي المنطقي !

Truth table for operator ! (logical negation)

والجدول التالي يبين أولويات المؤثرات التي درسناها حتى الآن (حيث الأولوية تتناقص من أعلى الجدول إلى أسفله) . كما يبين الجدول كيفية تجميع كل من هذه المؤثرات .

المؤثرات Operators	التجميع Associativity	النوع Type
++ -- + - ! (type)	من اليمين إلى اليسار	أحادي (unary)
* / %	من اليسار إلى اليمين	ضربي (multiplicative)
+ -	من اليسار إلى اليمين	جمعي (additive)
< <= > >=	من اليسار إلى اليمين	علاقي (relational)
== !=	من اليسار إلى اليمين	تساوي (equality)
&&	من اليسار إلى اليمين	AND المنطقية (logical AND)
||	من اليسار إلى اليمين	OR المنطقية (logicalOR)
?:	من اليمين إلى اليسار	شَرطي (conditional)
= += -= *= /= %=	من اليمين إلى اليسار	إسناد (assignment)
,	من اليسار إلى اليمين	الفاصلة (comma)

جدول أولويات المؤثرات وتجميعها

الالتباس بين مؤثر التساوي (==) ومؤثر الإسناد (=)
(Confusing Equality and Assignment Operators)

يلاحظ أن مؤثر التساوي == يختلف عن مؤثر الإسناد = . واستخدام أحدهما (سهواً / خطأً) بدلا من الآخر لا يؤدي عادة إلى خطأ تركيبي (syntax error) ، ولكنه قد يؤدي إلى نتائج خاطئة (incorrect results) ناشئة عن أخطاء منطقية وقت التشغيل (runtime logic errors) . وترجع أسباب هذه المشكلة إلى خاصيتين من خصائص لغة C :

- الخاصية الأولى : يمكننا أن نستخدم في جزء الشروط / اتخاذ القرار (decision portion) في أي عبارة تحكم (control statement) أي تعبير (expression) بلغة C تكون له (يُنتج) قيمة (produces a value) . فإن كانت

هذه القيمة صفرا 0 ، فإنها تعامَل على أنها false (خاطئ) ، وإن كانت أي قيمة خلاف الصفر (nonzero) ، فإنها تعامَل على أنها true (صحيح) .

- الخاصية الثانية : أي عبارة إسناد في لغة C تكون لها (تُنتج) قيمة (produces a value) ، وهي القيمة التي تُسنَد إلى المتغير الموجود أيسر مؤثر الإسناد (assignment operator) .

مثال ٤ – ١٣ : نفرض أن الدرجة النهائية في اختبار مسابقة لحفظ القرآن الكريم هي 10 ، وأن كل من يحصل على الدرجة النهائية يُعطَي جائزة ، ولذلك فإننا نود كتابة العبارة :

```
if ( mark == 10 )
   printf( "You get a prize!" );
```

ولكننا بدلا من ذلك كتبنا العبارة (الخاطئة):

```
if ( mark = 10 )
   printf( "You get a prize!" );
```

عبارة **if** الأولى (الصحيحة) تمنح جائزة للشخص الذي درجته تساوي 10 . أما عبارة **if** الثانية (الخاطئة) فإنها توجد قيمة تعبير الإسناد في شرط **if** . وهذا التعبير هو إسناد بسيط قيمته هي الثابت 10 . وحيث أن أي قيمة غير صفرية تفسَّر على أنها " true " ، فلذلك سيكون الشرط في عبارة if هذه دائما true أي متحققا ، وبالتالي سيعطَى الشخص دائما جائزة بغض النظر عن درجته الفعلية !

تمرينات رقم ٤

٤ – ١) اكتب عبارة واحدة أو مجموعة عبارات لتنفيذ كل مما يلي :

أ) باستخدام عبارة **for** أوجد مجموع الأعداد الصحيحة الفردية من 1 إلى 99 . افرض أن المتغيرين الصحيحين count , sum قد تم تعريفهما .

ب) باستخدام عبارة **for** اطبع الأعداد الصحيحة من 1 إلى 20 . افرض أنه قد تم تعريف المتغير العداد **x** (counter variable) ، ولكنه لم يعطَ قيمة ابتدائية . اطبع خمسة أعداد صحيحة فقط في السطر الواحد . [إرشاد : استخدم العملية الحسابية x %5 . وعندما تكون قيمة هذه العملية صفرا اطبع رمز السطر الجديد (newline character) ، وإلا فاطبع رمز الجدولة (tab character) .

٤ – ٢) اكتشف الخطأ في كل من القطعتين التاليتين ، ووضح كيفية تصحيحه .

أ)
```
for ( y = .1; y != 1.0; y += .1 )
   printf( "%f \n", y );
```

ب)
```
switch ( n ) {
  case 1;
   printf( "The number is 1\n" );
  case 2;
      printf( "The number is 2\n" );
      break;
   default ;
       printf( "The number is not 1 or 2\n" );
      break;
   }
```

٤ – ٣) أوجد الخطأ في كل مما يلي ، مع ملاحظة احتمال وجود أكثر من خطأ واحد :

أ)
```
For ( x = 100, x >= 1,  x++ )
  printf( "%d \n", x );
```

ب) القطعة التالية يجب أن تطبع رسالة تفيد ما إذا كان العدد الصحيح المعطَى **value** فرديا أم زوجيا .

```
switch ( value % 2  ) {
   case 0;
     printf( "Even integer\n" );
   case 1;
```

```
        printf( "Odd integer\n" );
    }
```

ج)
```
for ( x = .000001; x <= .0001; x += .000001 )
   printf( "%.7f\n", x );
```

د) القطعة التالية يجب أن تطبع الأعدادالصحيحة الفردية من 999 إلى 1 .

```
for ( x = 999; x >= 1; x += 2 )
   printf( "%d \n", x );
```

هـ) القطعة التالية يجب أن تطبع الأعداد الصحيحة الزوجية من 2 إلى 100 .

```
counter = 2;
Do {
    if ( counter % 2 == 0 )
       printf( "%d\n", counter );

     counter += 2;
}While ( counter < 100 );
```

و) القطعة التالية يجب أن توجد مجموع الأعداد الصحيحة من 100 إلى 150 .
افرض أن total قد أُعطى القيمة الابتدائية 0 من قبل .

```
for ( x = 100; x <= 150; x ++ );
     total += x;
```

٤ – ٤) ما هي قيم متغير التحكم **x** التي تطبعها كل من عبارات **for** التالية ؟

أ)
```
for ( x = 2; x <= 13; x += 2 )
   printf( "%d\n", x );
```

ب)
```
for ( x = 5; x <= 22; x += 7 )
   printf( "%d\n", x );
```

ج)
```
for ( x = 3; x <= 15; x += 3 )
   printf( "%d\n", x );
```

د)
```
for ( x = 1; x <= 5; x += 7 )
   printf( "%d\n", x );
```

هـ)
```
for ( x = 12; x >= 2; x -= 3 )
   printf( "%d\n", x );
```

٤ – ٥) اكتب عبارات **for** تطبع متتابعات القيم التالية :

أ) 1, 2, 3, 4, 5, 6, 7

ب) 3, 8, 13, 18, 23

ج) 20, 14, 8, 2, -4, -10

د) 19, 27, 35, 43, 51

٤ – ٦) تتبع تنفيذ البرنامج التالي واكتب مخرجاته بالضبط ، بفرض أن مدخلاته هي : 4 3

```
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int x;
   int y;
   int i;
   int j;

   /* prompt user for input */
   printf( "Enter two integers in the range 1-20: " );
   scanf( "%d%d", &x, &y ); /* read values for x and y */

   for ( i = 1; i <= y; i++ ) { /* count from 1 to y */

      for ( j = 1; j <= x; j++ ) { /* count from 1 to x */
         printf( "@" ); /* output @ */
      } /* end inner for */

      printf( "\n" ); /* begin new line */
   } /* end outer for */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

٤ – ٧) اختر الاجابة الصحيحة في كل مما يلي :

i) أي مقدمات / عناوين (headers) **for** التالية غير صحيحة ؟

أ) `for( int i = 0; i < 10; i++ )`

ب)
```
int i;
for( ; i < 10; i++ )
```

ج)
```
int i;
for( ; ; i++ )
```

د)
```
int i;
for( ; i < 10; )
```

ii) ماذا ينتج عن عبارة **for** ذات جسم صحيح (correct body) والعنوان التالي ؟

```
for ( i = 20;   i > = 2;   i += 2 )
```

أ) خطأ تركيبي .

ب) خطأ القسمة على صفر .

ج) عروة لا نهائية .

د) قيم i الزوجية من 20 إلى 2 تنازليا .

iii) قطعة البرنامج التالية

```
int counter = 1;
do {
  printf ( "%i", counter);
}while ( ++counter <= 10 );
```

أ) تطبع الأعداد الصحيحة من 1 إلى 11 .

ب) تطبع الأعداد الصحيحة من 1 إلى 10 .

ج) تطبع الأعداد الصحيحة من 1 إلى 9 .

د) ينتج عنها خطأ تركيبي .

iv) نفرض أن **x** متغير صحيح قيمته الابتدائية 12. ما هي مخرجات (output) العبارة التالية؟

```
if( x = 6 )
  printf("%i", x);
```

أ) 6 ب) 12

ج) لا شئ د) ينتج خطأ تركيبي

٤ – ٨) اكتب برنامجا – يستخدم عبارة **for** – لإيجاد مجموع متتابعة (sequence) من الأعداد الصحيحة . افرض أن أول عدد صحيح تقرأه **scanf** يحدد عدد القيم المتبقية في المدخلات [وهو عدد القيم المطلوب جمعها] . وافرض كذلك أن البرنامج يقرأ قيمة واحدة فقط كلما نُفِّذت عبارة **scanf**. وفيما يلي مثال لمتتابعة الإدخال (input sequence) :

5 100 200 300 400 500

حيث العدد 5 يعني أن القيم الخَمس التالية هي المطلوب جمعها .

٤ – ٩) اكتب برنامجا لإيجاد أصغر قيمة من عدة قيم صحيحة . افرض أن أول قيمة تُقرأ تحدِّد عدد هذه القيم المطلوب إيجاد أصغرها .

٤ – ١٠) اكتب برنامجا لحساب وطباعة مجموع الأعداد الصحيحة الزوجية من 2 إلى 30 .

٤ – ١١) اكتب برنامجا لحساب وطباعة حاصل ضرب الأعداد الصحيحة الفردية من 1 إلى 15 .

٤ – ١٢) اكتب برنامجا لحساب قيم مضروب (factorial) العدد الصحيح الموجب n ، وذلك لقيم n من 1 إلى 5 . واطبع النتائج في صيغة جدول (tabular form) .

٤ – ١٣) عَدِّل برنامج مثال ٤ – ٥ [برنامج الربا المركب / الفائدة **المركبة** compound interest] بحيث يكرر البرنامج خطواته لِنِسَب ربوية / لمعدلات فائدة (interest rates) تساوي :

5%, 6%, 7%, 8%, 9%, 10%

استخدم عروة **for** لتغيير النسبة الربوية .

٤ – ١٤) اكتب برنامجا يطبع الأشكال / النماذج (patterns) التالية (منفصلة عن بعضها) واحدا أسفل الآخر . استخدم عُرَى **for** للحصول على هذه الأشكال . جميع النجوم (*) يجب أن تُطبع باستخدام عبارة **printf** مفردة صيغتها ; ("*") printf [هذا يؤدي إلى طباعة النجوم بجانب بعضها البعض (side by side)] .

إرشاد : في الشكلين الأخيرين (C), (D) يبدأ كل سطر بعدد مناسب من الفراغات .

(A)
```
*
**
***
****
*****
******
*******
********
*********
**********
```

(B)
```
**********
*********
********
*******
******
*****
****
***
**
*
```

(C)
```
**********
 *********
  ********
   *******
    ******
     *****
      ****
       ***
        **
         *
```

(D)
```
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

٤ – ١٥) من تطبيقات الحاسوب المفيدة رسم المنحنيات البيانية (graphs) و" مخططات الأعمدة " (bar charts) [والتي يطلق عليها أحيانا " المدرجات التكرارية " (histograms)] . اكتب برنامجا يقرأ خمسة أعداد صحيحة (كل منها بين 1 و 30). ولكل عدد من هذه الأعداد يقوم البرنامج بطباعة سطر يحتوي على هذا العدد من النجوم المتجاورة . فمثلاً إذا قرأ البرنامج العدد 7 ، فإنه يطبع ******* .

٤ – ١٦) الجدول التالي يعطي أرقام خمس منتوجات وسعر الواحدة منها . اكتب برنامجا يقرأ أزواجا من أعداد (pairs of numbers) يمثل كل زوج منها : رقم منتوجة (product number) ، والكمية (quantity) المباعة منها . وتنتهي هذه الأزواج بإدخال رقم منتوجة يساوي 1- . ويقوم البرنامج بحساب القيمة الكلية للمبيعات مستخدما عبارة **switch**.

Product number	Retail price
1	$ 2.98
2	$ 4.50
3	$ 9.98
4	$ 4.49
5	$ 6.87

٤ – ١٧) أعد كتابة برنامج مثال ٤ – ٢ بحيث يُعطَى المتغير **counter** قيمته الابتدائية عند تعريفه بدلا من إعطائه إياها في عبارة **for** .

٤ – ١٨) عدِّل برنامج مثال ٤ – ٦ بحيث يوجد التقدير المتوسط (average grade) للصف .

٤ – ١٩) نفرض أن $i = 1, j = 2, k = 3, m = 2$.

ماذا تطبع كل من العبارات التالية ؟

أ) `printf( "%d", i == 1 );`

ب) `printf( "%d", j == 3 );`

ج) `printf( "%d", i >= 1 && j < 4 );`

د) `printf( "%d", m < = 99 && k < m );`

هـ) `printf( "%d", j >= i || k == m );`

و) `printf( "%d", k + m < j || 3 - j >= k );`

ز) `printf( "%d", !m );`

ح) `printf( "%d", !( j - m ) );`

ط) `printf( "%d", !( k > m ) );`

ي) `printf( "%d", !( j > k ) );`

٤ – ٢٠) احسب قيمة Π من المتسلسلة اللانهائية التالية :

$$\Pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$$

اطبع جدولا يبين قيم Π المقرَّبة المحسوبة بأخذ حد واحد (one term) من المتسلسلة ، وحدين ، وثلاثة حدود ، ... الخ .

٤ – ٢١) (الثلاثيات الفيثاغورثية) (Pythagorian Triples) يمكن للأضلاع الثلاثة لمثلث قائم الزاوية أن تكون جميعها أعدادا صحيحة . ومجموعة الأعداد الصحيحة الثلاثة التي تمثل أطوال أضلاع مثلث قائم الزاوية يطلق عليها "ثلاثية فيثاغورثية" (a Pythagorian Triple) . وهذه الأضلاع الثلاثة يجب أن تحقق العلاقة أن مجموع مربعي ضلعين يساوي مربع الوتر (hypotenuse) . اكتب برنامجا لإيجاد

جميع الثلاثيات الفيثاغورثية التي لا يزيد طول أي من ضلعيها side1 , side2 ووترها عن 500 . استخدم ثلاث عرى **for** متداخلة تجرِّب (tries) جميع الاحتمالات المختلفة .

٤ – ٢٢) ينقسم الموظفون (employees) في إحدى الشركات إلى فئات أربع :

i) المديرون (managers) : وهؤلاء يتقاضون راتبا أسبوعيا ثابتا (a fixed weekly salary) .

ii) العمال بالساعة (hourly workers) : وهؤلاء يتقاضون أجرا ثابتا بالساعة (a fixed hourly wage) عن الساعات التي لا تزيد عن الأربعين ساعة الأولى التي يعملونها ، وأجرا مساويا مرة ونصف (1.5) للأجر في الساعة الواحدة عن أي ساعات إضافية (overtime hours) يعملونها .

iii) العمال بالعمولة (commission workers) : وهؤلاء يتقاضون 250 $ مضافا إليها نسبة 5.7% من جملة مبيعاتهم الأسبوعية (gross weekly sales) .

iv) العمال بالقطعة (pieceworkers) : وهؤلاء يتقاضون قدرا ثابتا من المال (fixed amount of money) عن كل قطعة / وحدة (piece / item) ينتجونها [وكل عامل من هؤلاء العمال بالقطعة في هذه الشركة يعمل لإنتاج نوع واحد فقط من الوحدات].

اكتب برنامجا لحساب الأجر الأسبوعي (weekly pay) لكل موظف . افرض أنك لا تعرف سلفا (in advance) عدد الموظفين ، وأن كل فئة من الموظفين لها شفرتها (pay code) الخاصة تبعا لما يلي : المديرون : 1 ، العمال بالساعة : 2 ، العمال بالعمولة : 3 ، العمال بالقطعة : 4 . استخدم عبارة **switch** لحساب ما يتقضاه كل موظف بناء على شفرته (pay code) . وفي داخل عبارة **switch** اطلب من المستخدم (user) [أي مسئول الميزانية / الرواتب (payroll clerk)] إدخال البيانات المناسبة التي يحتاجها البرنامج لحساب ما يتقاضاه الموظف بناءً على شفرته .

٤ – ٢٣) اكتب برنامجا يقوم بطباعة شكل المعيَّن (diamond) التالي . يمكنك استخدام عبارات **printf** لطباعة إما نجمة مفردة (*) أو فراغ مفرد (single blank) . استخدم أكبر قدر ممكن من التكرار (بعبارات **for** المتداخلة) وأقل عدد ممكن من عبارات **printf** .

```
        *
       ***
      *****
     *******
    *********
     *******
      *****
       ***
        *
```

٤ – ٢٤) عدِّل برنامج السؤال السابق ليقرأ أولا عددا فرديا في المدى من 1 إلى 19 يمثل عدد الصفوف في شكل المعيَّن . ثم يطبع البرنامج معينا بالحجم المطلوب .

٤ – ٢٥ – أ) نفرض أنك قد أُعطيت عروة **do…while** . وضِّح كيف تكتب عروة **while** مكافئة (equivalent) لها لتحل محلها (to replace it) .

ب) نفرض أنك قد أُعطيت عروة **while**. وضح كيف تكتب عروة **do…while** مكافئة لها لتحل محلها . ما هي المشكلة التي تظهر في هذه الحالة ؟ ما هي عبارة التحكم الإضافية التي تحتاجها هنا ، وكيف تستخدمها لتضمن أن سلوك البرنامج الناتج (الذي وضعت فيه عبارة **do…while** المكافئة) هو سلوك البرنامج الأصلي (الذي يحتوي على عبارة **while**) نفسه ؟

٤ – ٢٦) ما هي مخرجات قطعة البرنامج التالية ؟

```
for ( i = 1; i <= 5; i++ ) {
    for ( j = 1; j <= 3; j++ ) {
        for ( k = 1; k <= 4; k++ )
            printf( "*" );
        printf( "\n" );
    }
    printf( "\n" );
}
```

٤ – ٢٧) من الانتقادات التي تُوجَّه إلى كل من عبارة **break** وعبارة **continue** أن أيًّا منهما غير مبنية (unstructured) . وعموما يمكننا دائما أن نستبدل بعبارات , **break** **continue** عبارات مبنية .

أ) وضّح كيف يمكنك عموما حذف أي عبارة **break** من عروة في برنامج ووضع بنية مكافئة (equivalent structure) بدلا من هذه العبارة .

إرشاد : عبارة **break** تؤدي إلى الخروج من العروة من داخل جسم العروة . والطريقة الأخرى للخروج من العروة هي عدم تحقق شرط اختبار استمرارية العروة -loop) (continuation test . فيمكنك في هذا الاختبار وضع شرط آخر يفيد الخروج المبكر (early exit) من العروة بسبب شرط "break" . استخدم هذه الطريقة لحذف عبارة **break** من برنامج مثال ٤ – ٨ .

ب) وضّح كيف يمكنك عموما حذف أي عبارة **continue** من عروة في برنامج ووضع بنية مكافئة بدلا من هذه العبارة . استخدم هذه الطريقة لحذف عبارة **continue** من برنامج مثال ٤ – ٩ .

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 ^{2}

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

$$x = 1.0 , 1.1 , 1.2 , 1.3 , \ldots , 2.0$$

وهكذا بالنسبة لباقي المتغيرات.

اكتب برنامجا يستخدم ثلاث عرى 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 $a \neq 0$ فإن الحل هو القيمة $x = -b/a$.

(ب) وإذا كان $a = 0 , b \neq 0$ 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 ، $f(\theta)$. (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.

٣٥-٤) 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 $\theta = \frac{5}{9}(\phi - 32)$.

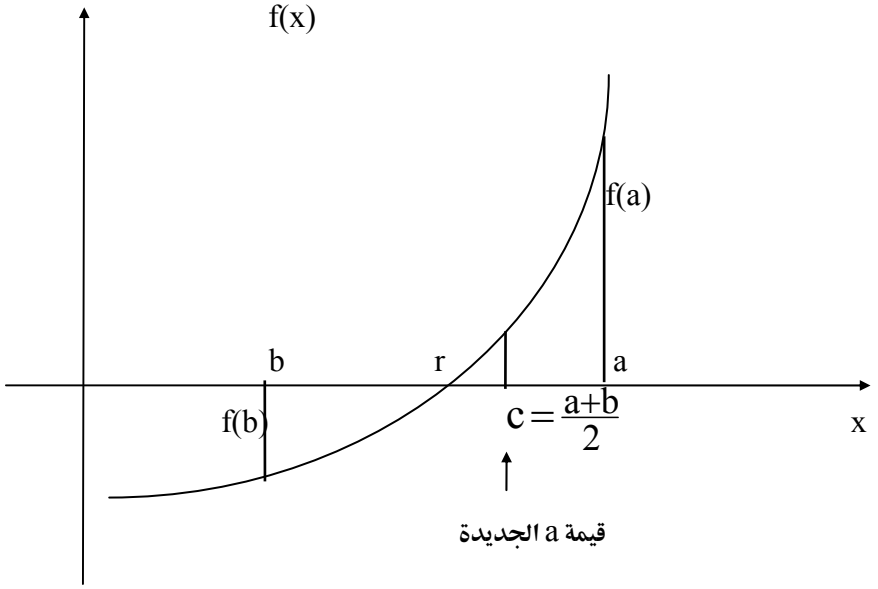
٣٦-٤) اكتب برنامجا مقابلا لخريطة سير العمليات في المسألة ١-٥ لحساب التعداد السنوي للسكان في كل من البلدين A , B وإلى أن يزيد تعداد سكان B عن تعداد سكان A.

٣٧-٤) اكتب برنامجا لحساب مجموعة المتسلسلة التالية (انظر المسألة ١-٧-ii)

$$s = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \cdots + \frac{99}{100}$$

٣٨-٤) 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 $x = a$, $x = b$) بحيث أن $f(a)$, $f(b)$ لهما إشارتان مختلفتان (انظر الرسم) ، وبالتالي فلابد أن يوجد جذر r (أي أن $f(r) = 0$) بين a , b .
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 a , c 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

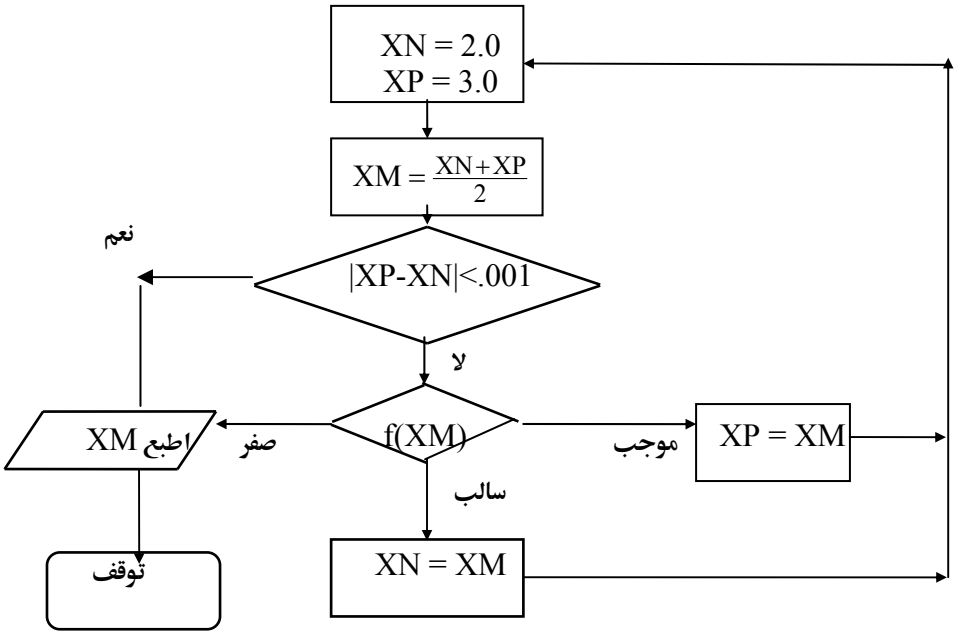
(إرشاد : $f(0) < 0$, $f(\pi) > 0$, $\pi = 3.14159$)

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:



٤-٤٠) 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 NL , NS , NA الذين وصلوا برا وبحرا وجوا على الترتيب.

(ب)العدد الكلي للحجاج N.

(ج)النسب المئوية PL , PS , PA 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.

٤-٤٣) 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 $\frac{p}{p_0}$ عند زمن معين t ، وذلك بأخذ أول عشر حدود من المتسلسلة أي بوضع $n = 9$.

٤-٤٤) 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.

٤-٤٥) 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 كلتى 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 :

$$A < B + C , \qquad B < A + C \quad \& \quad C < A + B$$

٤-٤٦) اكتب برنامجا مع رسم خريطة سير عملياته لحساب :

(أ) قوى العدد ٢ :

$$y_i = 2^i , \qquad i = 1 , 2, 3, \ldots$$

(ب) مقلوبات هذه القوى ، أي الكسور :

$$z_i = \frac{1}{2^i} \qquad ; i = 1,2,3,.....$$

(ج) المجموعات المتراكمة الجزئية لهذه الكسور :

$$s_i = \sum_{k=1}^{i} \frac{1}{2^k}$$

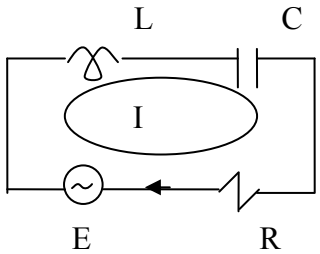
بحيث يطبع البرنامج النتائج في أربعة أعمدة كما يلي :

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 واحدا 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 : الضغط (ووحدته : لكل بوصة مربعة psi)

V : الحجم (ووحدته : قدم^{٣} cubic feet)

t : درجة الحرارة (بالتقدير الفهرنهيتي ^{o}F)

والمطلوب حساب V لتوافقات مختلفة من T , P 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 (half - life) H وهو الفترة الزمنية اللازمة للنظير ليتحلل إلى نصف كتلته الأصلية. مثلا H بالنسبة للنظير "سترونتيم ٩٠" (Sr^{90})(Strontium 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…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

$(1\ldots10)$, $(1\ldots11)$, $(1\ldots12)$, ... , $(1\ldots20)$

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 S , P ويحسب ويطبع ما يلي :

(أ) (i) زكاة هذه المدخرات ، وقيمة المدخرات المتبقية بعد إخراج الزكاة.

(ii) زكاة المدخرات المتبقية عندما يحول عليها حول آخر ، والمدخرات المتبقية (بعد حولين) بعد إخراج هذه الزكاة. افرض أن سعر جرام الذهب لم يتغير.

(iii) 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.

(ب) التباين (variance) في مجموعة البيانات. ولحساب التباين احسب كلا من مجموع القيم sum-of-data ومجموع مربعات القيم (sum-of-squares) في العروة (loop) الرئيسية ، وبعد انتهاء العروة استخدم المعادلة:

$$\text{variance} = \text{sum-of-squares} - (\text{sum-of-data})^2/N$$

(ج) الانحراف القياسي S (standard deviation) والذي هو عبارة عن قياس لكيفية انحراف البيانات عن المتوسط، باستخدام المعادلة.

$$S^2 = \text{variance} / (N - 1)$$

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 ذو نقطة عائمة) في نهاية مخرجات البرنامج.

٤-٦٩) اكتب برنامجا لقراءة البطاقات الزمنية الأسبوعية لموظفي إحدى المؤسسات. وكل موظف له ثلاثة 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 الأوَّلي (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) 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

٤-٧٣) 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
```

٤-٧٤) 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.

٤-٧٥) اكتب برنامجا يقرأ جملة (sentence) تنتهي بنقطة (a period) ، ويطبع عدد مرات ظهور النمط (pattern) "an" [أي الحرف a يليه مباشرة الحرف n] . مثلا إذا قرأ البرنامج الجملة

Ar-Rahman has taught the Quran, has created man, and has taught him utterance.

فإنه يطبع رسالة مثل :

Number of occurrences of the pattern "an" is: 5

إرشاد : يمكنكم اتباع الخوارزمية التالية :

```
int count = 0
char  ch, prev_ch
read prev_ch
read ch
while ( ch ≠ '.' ) {
      if ( ch == 'n'  and  prev_ch == 'a' )
            count++
      prev_ch = ch
      read ch }
print count
```

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 تغيير 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 الرئيسية 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 تُكتب 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 الحاسوب (والتي تشتمل على عدة برامج جاهزة يحتاج إليها عدد كبير من الناس)، وترتبط هذه البرامج الفرعية بالبرنامج الرئيسي لحل المسألة الأصلية الطويلة.

وهناك دوال مكتبية قياسية رياضية (mathematical standard library routines) مثل sin, cos, sqrt ، حيث نستدعي أيا من هذه الدوال أو 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 / الفارغة (void functions) .

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 * square (x);
```

وأما الدالة الخاوية فإنها لا تعيد قيمة دالية، ولا تُستدعى في تعبير ، وإنما تستدعى بعبارة كاملة مستقلة قائمة بذاتها (complete, stand-alone statement) ، مثل

```
printf( "%d%d", m, n);
```

وسنبدأ بإذن الله بالحديث عن الدوال المكتبية القياسية الرياضية ، ثم ننتقل بعد ذلك إلى الدوال المعرَّفة بالمستخدِم .

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 .

الدوال المكتبية الرياضية Math Library Functions

الدوال المكتبية الرياضية تسمح للمبرمج بإجراء بعض الحسابات الرياضية الشائعة . وهنا نستخدم عددا من هذه الدوال المكتبية الرياضية لتقديم مفهوم الدوال ، وفي بقية الكتاب والكتاب اللاحق (البرمجة المتقدمة بلغة C) نناقش بإذن الله تعالى عددا كبيرا من الدوال الأخرى في مكتبة C القياسية .

وتُستخدم الدوال عادة في أي برنامج بكتابة اسم الدالة يليه قوس أيسر يليه وسيط (argument) الدالة [أو قائمة (list) وسطاء تفصل بينها فاصلات (commas)] يليه قوس أيمن. فمثلا إذا أردنا أن نحسب ونطبع الجذر التربيعي للقيمة 900.0 فيمكننا أن نكتب العبارة

```
printf ( "%.2f", sqrt( 900.0 ) );
```

وعند تنفيذ هذه العبارة يتم استدعاء (calling) الدالة **sqrt** لحساب الجذر التربيعي للعدد الموجود بين القوسين (900.0) . فالعدد 900.0 هو وسيط الدالة **sqrt**. والعبارة السابقة ستقوم بطباعة 30.00 . والدالة **sqrt** تأخذ وسيطا من النوع **double** ، وتعيد نتيجة من النوع **double**. وجميع الدوال في المكتبة الرياضية تعيد نوع البيانات **double**. ولاحظ أن قيم النوع **double** يمكن طباعتها – مثل قيم النوع **float** – باستخدام محدِّد التحويل (conversion specifier) %f . ويفضَّل عند استخدام دوال من المكتبة الرياضية أن يحتوي البرنامج على ملف

المقدمة الرياضي (math header) باستخدام موجِّه قبل التشغيل (preprocessor directive) # **include < math.h >** وذلك منعا لحدوث أخطاء .

ووسطاء الدالة قد يكونون ثوابت أو متغيرات أو تعابير . فمثلا إذا كانت

c1=13, d=3.0, f=4.0

فإن العبارة

```
printf( "%.2f", sqrt( c1 + d * f ) );
```

تحسب وتطبع الجذر التربيعي للمقدار

13.0 + 3.0 * 4.0 = 25.0

أي تحسب وتطبع القيمة 5.0 .

والجدول التالي يلخص بعض دوال المكتبة الرياضية في لغة C ، مع ملاحظة أن المتغيرين y , x في الجدول هما من النوع **double**.

الدالة Function	وصف الدالة Description	أمثلة Examples
sqrt(x)	الجذر التربيعي للعدد x square root of x	sqrt(900.0) = 30.0 sqrt(9.0) = 3.0
exp(x)	الدالة الأسِّيَّة e^x exponential function e^x	exp(1.0) = 2.718282 exp(2.0) = 7.389056
log (x)	لوغاريتم x الطبيعي (الأساس e) natural logarithm of x (base e)	log(2.718282) = 1.0 log(7.389056) = 2.0
Log10 (x)	لوغاريتم x (للأساس 10) logarithm of x (base 10)	log10(1.0) = 0.0 log10(10.0) = 1.0 log10(100.0) = 2.0
fabs (x)	القيمة المطلقة للعدد x absolute value of x	fabs (5.0) = 5.0 fabs (0.0) = 0.0 fabs (-5.0) = 5.0
ceil (x)	سقف x : أي تقريبها إلى أصغر عدد صحيح ليس أصغر من x	ceil (9.2) = 10.0 ceil (-9.8) = -9.0

floor(x)	أرضية x : أي تقريبها إلى أكبر عدد صحيح ليس أكبر من x	floor(9.2) = 9.0 floor(-9.8) = -10.0
pow(x, y)	x مرفوعة للأس y (x^y) x raised to power y	pow(2, 7) = 128.0 pow(9,.5) = 3.0
fmod(x, y)	باقي x/y كعدد ذي نقطة عائمة	fmod(13.657, 2.333) = 1.992
sin(x)	sin(x) حيث x بالتقدير الدائري	sin(0.0) = 0.0
cos(x)	cos(x) حيث x بالتقدير الدائري	cos(0.0) = 1.0
tan(x)	tan(x) حيث x بالتقدير الدائري	tan(0.0) = 0.0

جدول الدوال المكتبية الرياضية القياسية شائعة الاستخدام

تعريفات الدوال Function Definitions

المتغيرات المعرَّفة في تعريفات الدوال تعد جميعها متغيرات محلية (local variables)، بمعنى أن هذه المتغيرات تكون معروفة فقط في الدالة التي تم فيها تعريف المتغيرات . ومعظم الدوال تحتوي على قائمة من الوسطاء (parameters) . ويتم تبادل المعلومات (communicating information) بين الدوال عن طريق الوسطاء . ووسطاء أي دالة يُعتبرون أيضا متغيرات محلية لهذه الدالة .

وفي جميع البرامج التي ذكرناها سابقا نلاحظ أن البرنامج يتكون من دالة تُدْعَى **main** تستدعي " دوال مكتبية قياسية " لإنجاز مهام معينة مثل الدوال : printf, scanf, pow . والآن نرى كيف يقوم المبرمج بكتابة دواله الخاصة .

مثال ٥ – ١ :

اكتب برنامجا يستخدم دالة **square** لحساب وطباعة مربعات الأعداد الصحيحة من 1 إلى 10 .

الحل :

```
/*Example 5.1
  Creating and using a programmer-defined function*/
#include <stdio.h>

int square( int y ); /* function prototype*/
```

```
 /*function main begins program execution*/
int main()
{
  int x; /* counter*/

 /*   loop 10 times and calculate and output square of x each time*/
  for ( x = 1; x <= 10; x++ )  {
    printf( "%d  ", square( x ) ); /* function call*/
 } */    end for*/

  printf( "\n;" );

  return 0; /* indicates successful termination*/

} /* end main */

 /* square function definition returns square of parameter*/
int square( int y ) /* y is a copy of argument to function*/
{
  return y * y; /* returns square of y as an int*/

} /* end function square*/
```

```
1  4  9  16  25  36  49  64  81  100
```

نلاحظ أن الدالة **square** قد تم استدعاؤها (invoked/called) في الدالة الرئيسية **main** داخل عبارة **printf**. والدالة **square** تستقبل نسخة (copy) من قيمة **x** في وسيطها y . ثم تقوم الدالة **square** بحساب **y** * **y** ، ويتم (إعادة (returning) /) تمرير (passing back) النتيجة (result) إلى الدالة **printf** في الدالة **main** عند الموضع الذي تم فيه استدعاء الدالة**square** . وتقوم الدالة **printf** بطباعة النتيجة . وهذه العملية تكرَّر 10 مرات باستخدام عبارة التكرار **for** .

وتعريف الدالة **square** (المذكور بعد الدالة **main**) يبين أن هذه الدالة تتوقع استقبال وسيط صحيح **y** (integer parameter) . وأما كلمة **int** التي تسبق اسم الدالة **square** فتعني أن هذه الدالة تعيد نتيجة صحيحة (integer result) أي تعيد عددا صحيحا . وعبارة **return** الموجودة في الدالة **square** تمرِّر نتيجة الحسابات إلى الدالة المستدعية (calling function) .

وأما السطر

```
int square( int y ); /* function prototype */
```

الموجود قبل الدالة **main** فيُعْرَف بـ " نموذج الدالة " (function prototype) . وكلمة **int** الموجودة بين القوسين تُخبر البرنامج المترجِم (compiler) أن الدالة **square** تتوقع استقبال قيمة صحيحة (integer value) من الدالة المستدعية . وكلمة **int** الموجودة يسار اسم الدالة **square** تخبر البرنامج المترجِم أن الدالة **square** تعيد نتيجة صحيحة إلى الدالة المستدعية . والبرنامج المترجِم يرجع إلى نموذج الدالة ليتحقق من أن استدعاءات الدالة **square** تحتوي على النوع المضبوط للقيمة / للنتيجة المعادة ، والعدد المضبوط من الوسطاء ، والأنواع المضبوطة للوسطاء ، والترتيب المضبوط للوسطاء . وسنتعرض بإذن الله لموضوع نماذج الدوال بتفصيل أكثر فيما بعد في هذا الفصل .

الصيغة العامة لتعريف أي دالة هي :

```
return-value-type function-name( parameter-list )
{
    definitions
    statements
}
```

حيث :

: function name — اسم الدالة ، وهو أي اسم تعريفي سليم (any valid identifier)

: return-value type — نوع القيمة المعادة ، وهو نوع بيانات (data type) النتيجة المعادة (returned result) إلى الدالة المستدعية (caller) . وإذا كان هذا النوع **void** فإن هذا يعني أن الدالة – التي نعرِّفها – لا تعيد قيمة . وإذا لم نحدِّد (specify) هذا النوع ، فإن البرنامج المترجِم (compiler) يفترض أنه **int** . وعموما يُفضَّل ذكر هذا النوع (return type) صراحةً (explicitly)، وعدم حذفه .

: parameter – list — قائمة وسطاء الدالة .

وأحيانا يطلق على مجموعة نوع القيمة المعادة ، واسم الدالة ، وقائمة وسطائها : " مقدِّمة الدالة " (function header) .

: definitions — تعريفات (أيّ متغيرات محلية)

: statements — عبارات الدالة .

ملاحظات على تعريفات الدوال :

- **حذف نوع القيمة المعادة في تعريف الدالة يُعد خطأً تركيبيا (syntax error) إذا حدَّد نموذج الدالة نوعا للقيمة المعادة مختلفا عن int .**

- **إذا نسينا إعادة قيمة من دالة يُفترض أن تعيد قيمة ، فإن ذلك قد يؤدي إلى أخطاء غير متوقعة . ولغة C القياسية تقرر أن نتيجة هذا الحذف غير معلومة / غير معرَّفة (undefined) .**

- **إعادة قيمة من دالةٍ نوع قيمتها المعادة (return type) : void يُعد خطأ تركيبيا .**

[ملاحظة : سنستخدم الاصطلاح " دالة void " للإشارة إلى الدالة التي نوع قيمتها المعادة void] .

- **قائمة الوسطاء [وهي قائمة تَفْصِل بين عناصرها فاصلات (comma-separated list)] تحدِّد الوسطاء الذين تستقبلهم الدالة حين تُستدعي . وحين لا تستقبل الدالة أي قيمة فإن قائمة الوسطاء هذه تكون خاوية void.**

- **يجب ذكر نوع (type) كل وسيط صراحةً إلا إذا كان الوسيط من النوع int . وإذا لم يُذكر نوع أي وسيط فيُفترض أنه int.**

- **إذا كان هناك أكثر من وسيط من وسطاء الدالة من النوع نفسه ، فإنَّ ذِكْر النوع مرةً واحدةً قبل أسماء الوسطاء متعاقبة – بدلا من ذِكر النوع قبل اسم كل وسيط – قد يؤدي إلى أخطاء في البرنامج . فمثلا إذا كتبنا**

```
double x, y
```

بدلا من :

```
double x,  double y
```

فإن الإعلان الأول `double x, y` **لوسطاء الدالة سيجعل y وسيطا من النوع int ، لأن نوع y لم يُذكر صراحة ، والنوع الافتراضي (default) هو int. وعموما يُفضل ذكر نوع كل وسيط حتى لو كان هو النوع الافتراضي int.**

- **وَضْع فاصلة منقوطة بعد القوس الأيمن الذي يحيط بقائمة الوسطاء في تعريف دالة يُعد خطأً تركيبيا .**

- تعريف وسيط دالة مرة أخرى كمتغير محلي داخل الدالة يُعد خطأ تركيبيا .

- من الجائز أن تتطابق أسماء الوسطاء الفعليين (arguments) الذين نمررهم إلى الدالة وأسماء الوسطاء " الشكليين " المقابلين (corresponding formal parameters) في تعريف الدالة . والبعض يفضل عدم استخدام الأسماء نفسها منعاً للالتباس (ambiguity) .

- التعريفات (definitions) والعبارات (statements) – في الصيغة العامة لتعريف الدالة – بين القوسين { } يكوّنان (form) ما يُعْرَف بـ " جسم الدالة " (function body). وأحيانا يُشار أيضا إلى جسم الدالة بـ " قالب " (block) . ويمكن الإعلان (declaration) عن متغيرات في أي قالب ، ويمكن كتابة قوالب متداخلة (nested). ولكن لا يجوز تعريف أي دالة داخل أي دالة أخرى في أي حال من الأحوال . وتعريف دالة داخل دالة أخرى يُعد خطأ تركيبيا .

- يجب أن يتفق نموذج الدالة (function prototype) ومقدمة الدالة (function header) واستدعاءات الدالة (function calls) في كلٍّ من عدد (number) ونوع (type) وترتيب (order) الوسطاء الفعليين والشكليين (arguments and parameters) ونوع القيمة المعادة .

وهناك ثلاث طرق لإعادة التحكم (returning control) من دالة مستدعاة إلى النقطة التي تم عندها استدعاء الدالة :

- إذا كانت الدالة لا تعيد نتيجة ، فإن التحكم يُعاد ببساطة عندما نصل إلى القوس الأيمن "} " الذي ينهي الدالة ،
- أو بتنفيذ العبارة

return;

- وإذا كانت الدالة تعيد نتيجة ، فإن العبارة

return expression;

تعيد قيمة التعبير expression إلى الدالة المستدعية .

مثال ٥ – ٢ : اكتب برنامجا يقرأ قيم ثلاثة أعداد صحيحة ، ويستدعي دالة **maximum** لإيجاد أكبر الأعداد الثلاثة ، ثم يطبع هذه القيمة الكبرى .

الحل :

```
/* Example 5.2
   Finding the maximum of three integers */
#include <stdio.h>

int maximum( int x, int y, int z ); /* function prototype */

/* function main begins program execution */
int main()
{
   int number1; /* first integer */
   int number2; /* second integer */
   int number3; /* third integer */

   printf( "Enter three integers: " );
   scanf( "%d%d%d", &number1, &number2, &number3 );

   /* number1, number2 and number3 are arguments
      to the maximum function call */
   printf( "Maximum is: %d\n", maximum( number1, number2, number3 ) );

   return 0; /* indicates successful termination */

} /* end main */

/* Function maximum definition */
/* x, y and z are parameters */
int maximum( int x, int y, int z )
{
   int max = x;     /* assume x is largest */

   if ( y > max ) { /* if y is larger than max, assign y to max */
      max = y;
   } /* end if */

   if ( z > max ) { /* if z is larger than max, assign z to max */
      max = z;
   } /* end if */
```

```
   return max;      /* max is largest value */

} /* end function maximum */
```

```
Enter three integers:  22  85  17
Maximum is:85
```

```
Enter three integers:  85  22  17
Maximum is:85
```

```
Enter three integers:  22  17  85
Maximum is:85
```

نلاحظ أن الأعداد الصحيحة الثلاثة يتم إدخالها بعبارة **scanf** . ثم تُمرَّر هذه الأعداد الثلاثة إلى الدالة **maximum** التي توجد أكبرها ، وتعيد هذه القيمة الكبرى إلى الدالة **main** بعبارة **return**. ثم تُطبع هذه القيمة المعادة بواسطة عبارة **printf** في الدالة main .

نماذج الدوال Function Prototypes

نموذج أي دالة يخبر البرنامج المترجم : نوع البيانات التي تعيدها الدالة ، وعدد الوسطاء الذين تتوقع الدالة استقبالهم ، وأنواع الوسطاء ، وترتيبهم الذي تتوقع الدالة وصولهم به . ويستخدم البرنامج المترجم نماذج الدوال للتحقق من صحة (validation) استدعاءات الدوال .

وفي المثال السابق (مثال ٥ – ٢) رأينا أن نموذج الدالة **maximum** هو

```
int maximum( int x, int y, int z ); /* function prototype */
```

وهذا النموذج يعني أن الدالة **maximum** تأخذ ثلاثة وسطاء من النوع int ، وتعيد نتيجة من النوع **int**. ولاحظ أن نموذج الدالة هو نفسه أول سطر في تعريف الدالة **maximum** .

وفي نموذج أي دالة يجوز أن نذكر أو لا نذكر أسماء الوسطاء (parameter names) . والبرنامج المترجم (compiler) يتجاهل هذه الأسماء .

وتوجد فاصلة منقوطة (semicolon) في نهاية نموذج أي دالة (على اليمين) . وعدم وضع هذه الفاصلة المنقوطة يُعد خطأً تركيبيا .

وأي استدعاء دالة لا يتوافق / لا يتواءم (does not match) مع نموذج الدالة يعد خطأ تركيبيا . وكذلك ينشأ خطأ إذا حدث اختلاف (disagreement) بين نموذج الدالة وتعريف الدالة . مثلا إذا استبدلنا بنموذج الدالة maximum في برنامج المثال السابق (مثال ٥ – ٢) النموذج التالي

```
void maximum( int x, int y, int z );
```

فإن البرنامج المترجم سيولِّد (generate) خطأً لأن نوع القيمة المعادة **void** المذكور في نموذج الدالة يختلف عن نوع القيمة المعادة **int** في مقدمة / عنوان الدالة (function header) .

ومن الخصائص الهامة لنماذج الدوال تشكيل / صب الوسطاء الفعليين (coercion of arguments) ، أي إجبار (forcing) هؤلاء الوسطاء على التقيُّد بالنوع المناسب (appropriate type) . فمثلا الدالة المكتبية الرياضية **sqrt** يمكن أن تُستدعي بوسيط فعلي من النوع الصحيح (integer argument) رغم أن نموذج الدالة في الملف math.h يحدِّد نوع الوسيط الفعلي (argument) **double** ، وستؤدي الدالة وظيفتها بطريقة صحيحة (correctly) .

مثال ٥ – ٣ : العبارة

```
printf( "%.3f\n", sqrt(4) );
```

تحسب قيمة sqrt (4) بطريقة صحيحة ، وتطبع القيمة 2.000 . فنموذج الدالة يجعل البرنامج المترجم يقوم بتحويل القيمة الصحيحة (integer) 4 إلى القيمة المضاعفة (double)4.0 قبل تمرير القيمة (passing the value) إلى الدالة **sqrt** . وعموما فإن قيم الوسطاء الفعليين التي لا تتفق في أنواعها بالضبط مع أنواع الوسطاء الشكليين في نموذج الدالة تُحوَّل إلى الأنواع المضبوطة قبل استدعاء الدالة . وهذه التحويلات يمكن أن تؤدي إلى نتائج خاطئة إذا لم نتبع " قواعد الترقية " في لغة C (C`s promotion rules) أي درجات تسلسل المراتب في لغة C . وقواعد الترقية تحدد كيفية تحويل الأنواع إلى أنواع أخرى بدون فقدان أي بيانات . ففي مثالنا السابق (مثال ٥ –٣ : الدالة **sqrt**) يُحوَّل أي عدد صحيح **int** تلقائيا (automatically) إلى عدد **double** بدون تغيير قيمته . إلا أن تحويل عدد **double** إلى عدد **int** يحذف الجزء الكسري من القيمة المضاعفة **double** . وكذلك تحويل أنواع صحيحة كبيرة (large integer types) إلى أنواع صحيحة صغيرة (small integer type) [مثلا تحويل **long** إلى **short**] قد يؤدي إلى تغيير القيم .

وقواعد الترقية تنطبق تلقائيا على التعابير التي تحتوي على قيم من نوعين أو أكثر من أنواع البيانات [ويطلَق أيضا على هذه التعابير : " تعابير مختلطة الأنواع" (mixed-type expressions)] . ونوع أي قيمة في تعبير مختلط الأنواع يُرَقَّى تلقائيا إلى " أعلى نوع " (highest type) في التعبير . [ما يحدث فعليا هو أنه يتم إنشاء صيغة مؤقتة (temporary version) من كل قيمة ، وتُستخدم هذه الصيغة في التعبير ، بينما تبقى القيم الأصلية دون تغيير] . الجدول التالي يعطي أنواع البيانات مُرَتَّبة حسب تسلسل مراتبها / رتبها من الأعلى إلى الأدنى ، وكذلك ما يقابل كل نوع من محددات التحويل / مواصفات التحويل (conversion specifications) مع كل من الدالتين **printf , scanf** .

نوع البيانات	محددات التحويل مع printf	محددات التحويل مع scanf
Data type	printf conversion specification	scanf conversion specification
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
short	%hd	%hd
char	%c	%c

جدول درجات ترقية / تسلسل مراتب أنواع البيانات
Promotion hierarchy for data types

وتحويل القيم إلى أنواع أدنى يؤدي عادة إلى قيم خاطئة . ولذلك فيمكن تحويل قيمة إلى نوع أدنى فقط عن طريق إسناد القيمة صراحة (explicitly) إلى متغير من نوع أدنى ، أو باستخدام مؤثر/ معامل صب (a cast operator) . ويتم تحويل قيم وسطاء الدالة الفعليين إلى أنواع الوسطاء الشكليين في نموذج الدالة كما لو أنه يتم إسنادهم مباشرة إلى متغيرات من هذه الأنواع . فلو أن دالتنا **square** التي تستخدم وسيطا صحيحا (integer parameter) [مثال ٥ – ٢] تم استدعاؤها بوسيط فعلي ذي نقطة عائمة (a floating paint argument) ، فإن هذا الوسيط الفعلي يُحوَّل إلى **int** (وهو نوع أدنى)، والدالة **square** تعيد عادة قيمة خاطئة (incorrect value) . فمثلا **square (4.5)** تعيد 16 وليس 20.25 .

وإذا لم نضع نموذج دالة (function prototype) لدالةٍ ما في البرنامج ، فإن البرنامج المترجِم يصيغ لنفسه نموذج دالة مستخدما أول ظهور / حدوث (first occurrence) للدالة : إما تعريف الدالة (function definition) أو استدعاء (call) للدالة . وافتراضيا (by default) فإن البرنامج المترجم يفترض أن الدالة تعيد قيمة **int** ، ولا يفترض أن شيء بخصوص الوسطاء الفعليين (arguments) . ولذلك فإذا كان الوسطاء الفعليون خاطئين (incorrect) ، فإن البرنامج المترجم لا يمكنه اكتشاف الأخطاء .

ونسيان كتابة نموذج دالة يؤدي إلى خطأ تركيبي إذا لم يكن نوع القيمة المعادة من الدالة هو **int**، وظهر تعريف الدالة بعد استدعاء الدالة في البرنامج . وما عدا ذلك فإن نسيان كتابة نموذج الدالة قد يؤدي إلى خطأ وقت التشغيل (run-time error) أو إلى نتائج غير متوقعة .

ويلاحظ أن وَضْع نموذج دالة خارج تعريف أي دالة يجعل تطبيق هذا النموذج يَسْرِي على جميع الاستدعاءات (لهذه الدالة) التي تظهر بعد نموذج الدالة في الملف . بينما وضع نموذج دالة داخل دالة أخرى يجعل تطبيق النموذج يَسْرِي فقط على الاستدعاءات التي تظهر في هذه الدالة الأخرى .

المقدِّمات Headers

أي مكتبة قياسية (standard library) يكون لها مقدِّمة مقابلة (corresponding header) تحتوي على نماذج دوال لجميع الدوال في هذه المكتبة ، وكذلك على تعريفات أنواع البيانات المختلفة والثوابت (constants) التي تحتاجها تلك الدوال . والجدول التالي يعرض بترتيب أبجدي بعض مقدمات المكتبات القياسية (standard library headers) التي قد تحتويها البرامج. وكلمة " ماكرو " [أي كبير أو مُوَسَّع] (macro) التي تظهر عدة مرات في الجدول عبارة عن اسم تعريفي (identifier) يُعرَّف في مُوجِّه قبل التشغيل # define (preprocessor directive)، ويُستخدم لتسمية (naming) عبارة (statement) معينة أو عملية (operation) معينة يشيع استخدامها (commonly used) .

مقدمة المكتبة القياسية Standard library header	المحتويات Contents
<assert.h>	تحتوي على الأسماء التعريفية الماكرو (macros) والمعلومات اللازمة لإضافة المشخّصات (diagnostics) التي تساعد في تصحيح أخطاء البرنامج (program

	. debugging)
<ctype.h>	تحتوي على نماذج دوال للدوال التي تختبر (test) الرموز (characters) للتحقق من خواص (properties) معينة، ونماذج دوال للدوال التي يمكن استخدامها لتحويل الحروف الصغيرة (lowercase letters) إلى حروف كبيرة (uppercase letters) ، والعكس .
<errno.h>	تعرِّف (defines) الأسماء الماكرو (macros) التي تفيد في تقرير / الدلالة على شروط الأخطاء (reporting error conditions) .
<float.h>	تحتوي على مدى / حدود سعة النقطة العائمة (floating point size limits) في النظام (system) .
<limits.h>	تحتوي على مدى السعة التكاملية (integral size limits) في النظام .
<locale.h>	تحتوي على نماذج دوال ومعلومات أخرى تساعد البرنامج على إمكانية تعديله (to be modified) بالنسبة للموضع الحالي (current locale) الذي يتم تشغيله (running) فيه. واصطلاح الموضع (notion of locale) تمكِّن نظام الحاسوب من التعامل مع اصطلاحات مختلفة (different conventions) للتعبير عن بيانات مثل التواريخ (dates) والأوقات (times) ومقادير الدولارات (dollar amounts) والأعداد الكبيرة (large numbers) في العالم .
<math.h>	تحتوي على نماذج دوال لدوال المكتبة الرياضية .
<setjmp.h>	تحتوي على نماذج دوال لدوال تسمح بتخطي / بتجنب

	(bypassing) التتابع المعتاد لاستدعاء دالة والعودة (usual function call and return sequence).
<signal.h>	تحتوي على نماذج دوال وأسماء ماكرو للتعامل مع أحوال مختلفة قد تنشأ أثناء تنفيذ البرنامج .
<stdarg.h>	تعرِّف أسماء ماكرو للتعامل مع قائمة من الوسطاء (الفعليين) لدالة غير معلوم عددهم وأنواعهم .
<stddef.h>	تحتوي على تعريفات مشتركة (common definitions) لأنواع تستخدمها لغة C لإنجاز حسابات معينة .
<stdio.h>	تحتوي على نماذج دوال لدوال مكتبة المدخلات والمخرجات القياسية (standard input/output library functions) .
<stdlib.h>	تحتوي على نماذج دوال لتحويلات (conversions) أعداد إلى نص (numbers to text) ، ونص إلى أعداد ، وحجز مواقع في الذاكرة (memory allocation) ، والأعداد العشوائية (random numbers) ، ودوال خدمات / منافع عامة (utility functions) أخرى .
<string.h>	تحتوي على نماذج دوال لدوال تشغيل سلاسل الرموز .
<time.h>	تحتوي على نماذج دوال وأنواع لمعالجة (manipulating) الوقت (time) والتاريخ (date).

بعض المقدِّمات المكتبية القياسية

Some standard library headers

ويستطيع المبرمج أن ينشئ مقدِّمات حسب طلبه (custom headers) . والمقدِّمات المعرَّفة بالمبرمج (programmer- defined headers) يجب أن تنتهي أيضا بالرمزين ". h" . والمقدِّمة المعرفة بالمبرمج يمكن أن يشتمل عليها (include) البرنامج باستخدام الموجِّه سابق التشغيل (preprocessor directive) #include. فمثلا إذا كان نموذج الدالة **square** التي ناقشناها سابقا موجودا (located) في المقدِّمة **square.h** ، فإننا نشمل (include) هذه المقدمة في برنامجنا باستخدام الموجِّه (directive) التالي في مطلع البرنامج

```
# include "square.h"
```

استدعاء الدوال : الاستدعاء بالقيمة والاستدعاء بالإسناد
Calling Functions : Call by Value and Call by Reference

هناك طريقتان لاستدعاء الدوال في كثير من لغات البرمجة ، وهما :
الاستدعاء بالقيمة ، والاستدعاء بالإسناد . وعندما يمرَّر الوسطاء الفعليون (arguments) بالقيمة ، فإن نسخة (copy) من قيمة الوسيط الفعلي تُعمل وتمرَّر إلى الدالة المستدعاة . والتغييرات التي تحدث في النسخة لا تؤثر على قيمة المتغير الأصلية في الدالة المستدعية . أما إذا مُرِّر الوسيط الفعلي بإسناد فإن الدالة المستدعية تسمح للدالة المستدعاة أن تُعدِّل (modify) من قيمة المتغير الأصلية . ويجب استخدام الاستدعاء بالقيمة عندما لا تحتاج الدالة المستدعاة إلى تعديل قيمة المتغير الأصلية في الدالة المستدعية ، فهذا يمنع حدوث الآثار الجانبية العرضية (accidental side effects) والتي تعوق بدرجة كبيرة تطوير (development) نظم برمجيات صحيحة عالية الوثوقية (correct and reliable software systems) . أما الاستدعاء بالإسناد فيجب أن يُستخدم فقط مع الدوال المستدعاة الموثوق بها (trusted) التي تحتاج إلى تعديل المتغير الأصلي .

وفي لغة C تتم جميع الاستدعاءات بالقيمة . ومن الممكن محاكاة (simulating) الاستدعاء بالإسناد باستخدام مؤثرات العناوين (address operators) ، ومؤثرات التوجيه غير المباشر (indirection operators) ، وهذا الموضوع خارج نطاق هذا الكتاب (*) . وفي الفصل القادم " المنظومات " سنرى باذن الله أن المنظومات تُمرر تلقائيا محاكاةً للاستدعاء بالإسناد . وسنركز حاليا على الاستدعاء بالقيمة .

(*) انظر الفصل الثالث " المؤشرات " في كتاب : البرمجة المتقدمة بلغة C ، دار اقرأ للنشر والتوزيع ، الكويت ، للمؤلف .

توجد في مكتبة C القياسية دالة تُدْعَى **rand** تقوم بتوليد عدد صحيح (integer) بين 0 و RAND_MAX [وهو عبارة عن ثابت رمزي (symbolic constant) مُعَرَّف في المقدمة (header) <stdlib.h>] . ويمكن استخدام الدالة **rand** في عبارة مثل :

```
i = rand( );
```

وتقرر ANSI القياسية أن قيمة RAND_MAX يجب أن تكون على الأقل 32767 ، والتي هي أكبر قيمة (maximum value) لعدد صحيح ذي بايتين (2-byte integer) ، أي ذي 16 رقم ثنائي / وحدة ثنائية (16-bit integer) . والبرامج التالية قد تم اختبارها على نظام C قيمة RAND_MAX العظمى فيه تساوي 32767 .

وإذا كانت الدالة **rand** تُولِّد فعلا أعدادا صحيحة عشوائية ، فإن أي عدد بين 0 و RAND_MAX تكون له فرصة متساوية (equal chance) – أي احتمال (probability) متساوي – لأن يتم اختياره / توليده كلما استُدعيت الدالة **rand** .

وعادة يكون مدى القيم (range of values) التي تُنتجها مباشرة الدالة **rand** مختلفا عما نحتاجه في تطبيق معين . فمثلا إذا كان البرنامج يحاكي (simulates) عملية قذف قطعة معدنية (coin tossing) ، فإنه يحتاج فقط لرقمين 0,1 لتمثيل أحد الوجهين (H/heads مثلا) والوجه الآخر (T/tails) ، وإذا كان البرنامج يحاكي عملية قذف قطعة نرد (dice-rolling) ذات ستة أوجه ، فإنه يحتاج فقط لأعداد صحيحة عشوائية في المدى من 1 إلى 6 (أي يحتاج فقط لستة أرقام : 1, 2, …, 6).

مثال ٥ – ٤ : [أعداد عشوائية (random numbers)] ، البرنامج التالي يحاكي تكرار عملية قذف قطعة النرد (dice-rolling) ذات الستة أوجه 20 مرة ، مع طباعة القيمة الناتجة كل مرة . ونموذج الدالة **rand** موجود في <stdlib.h> . وسنستخدم مؤثر الباقي (٪) مع الدالة **rand** هكذا :

```
rand( ) % 6
```

للحصول على أعداد صحيحة في المدى من 0 إلى 5 . وهذه العملية يُطلَق عليها "تغيير أبعاد" أو "وزن" أو "تحديد مقياس" (scaling) . والعدد 6 يطلق عليه "عامل القياس" (scaling factor) . ثم نزيح (shift) مدى الأعداد الناتجة بإضافة 1 إلى النتيجة السابقة . ومن مخرجات البرنامج التالي نلاحظ أن النتائج تقع في المدى من 1 إلى 6 .

```
/* Example 5.4
   Shifted, scaled integers produced by 1 + rand() % 6 */
#include <stdio.h>
#include <stdlib.h>

/* function main begins program execution */
int main()
{
   int i; /* counter */

   /* loop 20 times */
   for ( i = 1; i <= 20; i++ ) {

      /* pick random number from 1 to 6 and output it */
      printf( "%10d", 1 + ( rand() % 6 ) );

      /* if counter is divisible by 5, begin new line of output */
      if ( i % 5 == 0 ) {
         printf( "\n" );
      } /* end if */

   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

```
6       6       5       5       6
5       1       1       5       3
6       6       2       4       2
6       2       3       4       1
```

أعداد صحيحة عشوائية مُزاحة وموزونة/مقاسة بالدالة 1+rand( )%6

Shifted, scaled random integers produced by 1+ rand ( ) %6

مثال ٥ – ٥ :

يُفترض أن الأعداد العشوائية تظهر باحتمالات متساوية (equal likelihood) تقريبا . ولبيان ذلك فإن البرنامج التالي يحاكي عملية قذف قطعة النرد ذات الستة أوجه 6000 مرة . ومن مخرجات البرنامج نرى أن أي عدد صحيح في المدى من 1 إلى 6 قد نتج/ظهر (appeared) نحو 1000 مرة .

```
/* Example 5.5
   Roll a six-sided die 6000 times */
#include <stdio.h>
#include <stdlib.h>

/* function main begins program execution */
int main()
{
   int frequency1 = 0; /* rolled 1 counter */
   int frequency2 = 0; /* rolled 2 counter */
   int frequency3 = 0; /* rolled 3 counter */
   int frequency4 = 0; /* rolled 4 counter */
   int frequency5 = 0; /* rolled 5 counter */
   int frequency6 = 0; /* rolled 6 counter */

   int roll; /* roll counter, value 1 to 6000 */
   int face; /* represents one roll of the die, value 1 to 6 */

   /* loop 6000 times and summarize results */
   for ( roll = 1; roll <= 6000; roll++ ) {
      face = 1 + rand() % 6; /* random number from 1 to 6 */

      /* determine face value and increment appropriate counter */
      switch ( face ) {

         case 1: /* rolled 1 */
            ++frequency1;
            break;
```

```
         case 2: /* rolled 2 */
            ++frequency2;
            break;

         case 3: /* rolled 3 */
            ++frequency3;
            break;

         case 4: /* rolled 4 */
            ++frequency4;
            break;

         case 5: /* rolled 5 */
            ++frequency5;
            break;

         case 6: /* rolled 6 */
            ++frequency6;
            break; /* optional */
      } /* end switch */

   } /* end for */

   /* display results in tabular format */
   printf( "%s%13s\n", "Face", "Frequency" );
   printf( "    1%13d\n", frequency1 );
   printf( "    2%13d\n", frequency2 );
   printf( "    3%13d\n", frequency3 );
   printf( "    4%13d\n", frequency4 );
   printf( "    5%13d\n", frequency5 );
   printf( "    6%13d\n", frequency6 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Face        Frequency
   1             1003
   2             1017
   3              983
   4              994
```

5	1004
6	999

قذف قطعة نرد ذات ستة أوجه 6000 مرة
Rolling a six-sided die 6000 times

كما نرى من مخرجات هذا البرنامج فعن طريق تغيير الأبعاد / الوزن (scaling) **والإزاحة** (shifting) **أمكننا الاستفادة من الدالة rand في محاكاة عملية قذف قطعة النرد ذات الستة أوجه . ولاحظ أنه لا توجد حالة default في عبارة switch . كذلك لاحظ استخدام مُحدِّد التحويل %s لطباعة سلسلتي الرموز "Face" , "Frequency" كعنواني عمودين . وسنرى بإذن الله تعالى حين ندرس "المنظومات" في الفصل القادم (السادس) كيف يمكننا أن نستبدل بعبارة switch بأكملها عبارة ذات سطر واحد فقط .**

ونلاحظ أننا إذا قمنا بتشغيل / بتنفيذ (executing) **برنامج مثال ٥ – ٤ مرة أخرى ، فإننا سنحصل على النتائج / المخرجات نفسها (أي القيم المتتابعة نفسها) المبينة بعد البرنامج . فهل يمكننا اعتبار هذه الأعداد عشوائية وهي تتكرر بالتتابع نفسه ؟! الحقيقة أن هذا التكرار** (repeatability) **هو إحدى الخصائص الهامة للدالة rand . فالدالة rand تولِّد فعليا** (actually generates) **" أعدادا شبه عشوائية "** (pseudo-random numbers) **وليس أعدادا عشوائية . وتكرار استدعاء الدالة rand يؤدي إلى ظهور متتابعة** (sequence) **أعداد تبدو وكأنها عشوائية . إلا أن المتتابعة تكرر نفسها كلما نفذنا البرنامج. وفي الواقع إذا تمت عملية تصحيح أخطاء أي برنامج بدقة** (thoroughly debugged) **فإنه يمكننا ضبط البرنامج ليولِّد متتابعة مختلفة من أعداد عشوائية كلما نفذنا البرنامج . ويطلق على هذه العملية " ضبط العشوائية "** (randomizing) **، ويمكننا إنجازها باستخدام الدالة المكتبية القياسية srand . والدالة srand تأخذ وسيطا صحيحا بدون إشارة** (unsigned integer argument) **، وتَبْذر** (seeds) **الدالة rand [أي تعطيها البذرة** (seed) **] لتولِّد / لتنتج متتابعة مختلفة من أعداد عشوائية عند كل تنفيذ للبرنامج . والمثال التالي يوضح استخدام الدالة srand .**

مثال ٥ – ٦ : (ضبط عشوائية برنامج قذف قطعة النرد)
(Randomizing the die- rolling program)
في البرنامج التالي نستخدم نوع البيانات unsigned والذي هو اختصار unsigned int . وأي int يتم تخزينه في بايتين (2 bytes) **على الأقل من الذاكرة ، ويمكن أن يحتوي على قيم موجبة أو سالبة . وأي متغير من النوع unsigned يتم تخزينه أيضا في بايتين على الأقل من**

الذاكرة . وأي **unsigned int** في بايتين يمكنه أن يحتوي على قيم موجبة فقط في المدى من 0 إلى 65535. بينما أي **unsigned int** في 4 بايتات فيمكنه أن يحتوي على قيم موجبة فقط في المدى من 0 إلى 4294967295 . والدالة **srand** تأخذ قيمةً دون إشارة (an unsigned value) كوسيط (argument) لها . ونستخدم محدِّد التحويل u% لقراءة قيمةٍ unsigned بواسطة **scanf** . ونموذج الدالة **srand** موجود في الملف <stdlib.h> .

```
/* Example 5.6
   Randomizing die-rolling program */
#include <stdlib.h>
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int i;               /* counter */
   unsigned seed; /* number used to seed random number generator */

   printf( "Enter seed: " );
   scanf( "%u", &seed ); /* note %u for unsigned */

   srand( seed ); /* seed random number generator */

   /* loop 10 times */
   for ( i = 1; i <= 10; i++ ) {

      /* pick a random number from 1 to 6 and output it */
      printf( "%10d", 1 + ( rand() % 6 ) );

      /* if counter is divisible by 5, begin a new line of output */
      if ( i % 5 == 0 ) {
         printf( "\n" );
      } /* end if */

   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter seed: 67
     6    1    4    6    2
     1    6    1    6    4
```

```
Enter seed: 867
     2    4    6    1    6
     1    1    3    6    2
```

```
Enter seed: 67
     6    1    4    6    2
     1    6    1    6    4
```

إذا قمنا بتشغيل البرنامج عدة مرات ونظرنا إلى النتائج التي نحصل عليها ، فسنلاحظ أننا نحصل على متتابعة مختلفة من أعداد عشوائية كلما قمنا بتشغيل البرنامج بشرط أن نعطي بذرة مختلفة (different seed) .

وإذا أردنا أن نضبط عشوائية البرنامج دون الحاجة إلى إدخال بذرة كل مرة ، فيمكننا استخدام عبارة مثل

```
srand( time( NULL ) );
```

وهذه تجعل الحاسوب يقرأ ساعته (clock) ليحصل تلقائيا (automatically) على قيمة البذرة . والدالة **time** تعيد الوقت الحالي (current time) في اليوم بالثواني . وهذه القيمة تُحوَّل (converted) إلى عدد صحيح دون إشارة (unsigned integer) وتُستخدم على أنها البذرة لمولِّد الأعداد العشوائية (random number generator) . والدالة **time** تأخذ **NULL** كوسيط (an argument) . [الدالة **time** قادرة على أن تمد المبرمج بسلسلة (string) تمثل الوقت في اليوم (time of day) ، و **NULL** تُفقدها / تلغي هذه المقدرة (disables this capability) لاستدعاء محدد للدالة time] . ونموذج الدالة **time** موجود في <time.h> .

والقيم التي تنتج مباشرة من الدالة **rand** تكون دائما في المدى :

$$0 \leq \text{rand}\ (\ ) \leq \text{RAND\_MAX}$$

رأينا سابقا كيف يمكننا كتابة عبارة واحدة لمحاكاة (simulating) عملية قذف قطعة النرد ذات الأوجه الستة :

```
face = 1 + rand ( ) % 6;
```

فهذه العبارة تُسند دائما قيمة صحيحة – عشوائيا – إلى المتغير **face** في المدى $1 \leq face \leq 6$. ولاحظ أن عرض هذا المدى (width of range) [أي عدد الأعداد الصحيحة المتتالية في المدى (number of consecutive integers in the range)] يساوي 6 ، وعدد الابتداء (starting number) في المدى يساوي 1 . وبالرجوع إلى العبارة السابقة ، نرى أن عرض المدى يتحدد بالعدد المستخدم لوزن / لتحديد قياس (scaling) الدالة **rand** باستخدام مؤثر الباقي [أي 6] ، وأن عدد الابتداء في المدى يساوي العدد الذي يضاف إلى rand % 6 [أي 1] . ويمكننا تعميم هذه النتيجة كما يلي :

```
n = a + rand( ) % b;
```

حيث :

a : قيمة الإزاحة (shifting value)

(والتي تساوي العدد الأول في المدى المطلوب للأعداد الصحيحة المتتالية)

b : معامل القياس / الوزن (scaling factor)

(والذي يساوي عرض المدى المطلوب للأعداد الصحيحة المتتالية) وعموما يمكننا اختيار أعداد صحيحة عشوائيا من مجموعات قيم خلاف مدى أعداد صحيحة متتالية .

ونحب أن نشير هنا إلى أنه من الأخطاء الشائعة استخدام **srand** بدلا من **rand** لتوليد أعداد عشوائية .

طبقات التخزين Storage Classes

في الفصول الثلاثة السابقة استخدمنا أسماء تعريفية (identifiers) لأسماء متغيرات . وخصائص / صفات المتغيرات (attributes of variables) تشمل أسماءها وأنواعها وسعاتها وقيمها (names, types, sizes, values). وفي الفصل الحالي نستخدم الأسماء التعريفية أيضا كأسماء للدوال المعرَّفة بالمستخدم . وأي اسم تعريفي في برنامج يكون له خصائص / صفات أخرى تشمل : طبقة التخزين (storage class)، وأَمَد/مدة/فترة التخزين (storage duration)، والمجال (scope)، والارتباط (linkage).

وفي لغة C توجد أربع طبقات تخزين تشير إليها محدِّدات طبقات التخزين (storage class specifies) :

auto, register, extern, static

وطبقة تخزين أي اسم تعريفي تحدِّد فترة تخزينه ، ومجاله ، وارتباطه .

- أما فترة تخزين الاسم التعريفي فهي الفترة التي يوجد خلالها هذا الاسم التعريفي في الذاكرة . وبعض الأسماء التعريفية توجد لفترة قصيرة ، وبعضها الآخر يتم إنشاؤها (created) وتدميرها (destroyed) بصفة متكررة (repeatedly) ، بينما أسماء أخرى تظل موجودة في الذاكرة طوال فترة تنفيذ البرنامج .
- وأما مجال الاسم التعريفي فهو المواضع التي يمكن عندها الإشارة إلى / استخدام الاسم التعريفي في البرنامج . فبعض الأسماء التعريفية يمكن الإشارة إليها من أي موضع في البرنامج ، بينما أسماء أخرى لا يمكن الإشارة إليها إلا من أجزاء محددة فقط في البرنامج .
- وأما ارتباط الاسم التعريفي [وهو موضوع خارج منهج هذا الكتاب] فيعني تحديد ما إذا كان الاسم التعريفي معروفا فقط في الملف المصدري الحالي (current source file) أم في أي ملف مصدري بإعلانات معينة ، وذلك في حالة البرنامج متعدد الملفات المصدرية (multiple – source – file program) .

وفيما يلي نتناول موضوع طبقات التخزين وفترة التخزين .

يمكننا تقسيم محددات طبقة التخزين الأربعةإلى فئتين حسب فترة التخزين :

*) فترة التخزين الأوتوماتيكي/الأوتوماتي/الديناميكي/المتحرك
(automatic storage duration)

*) فترة التخزين الاستاتيكي/الاستاتي/الساكن
(static storage duration)

(أ) الكلمتان **auto, register** تُستخدمان للإعلان عن متغيرات ذوات فترة تخزين أوتوماتيكي . وهذه المتغيرات يتم إنشاؤها (created) عند دخول القالب (block) الذي تم تعريفها فيه ، وتظل موجودة (exist) طوال فترة نشاط/ عمل القالب (block is active)، ويتم تدميرها (destroyed) عند الخروج من القالب (block is exited) .

والمتغيرات (variables) فقط هي التي يمكن أن يكون لها فترة تخزين أوتوماتيكي . والمتغيرات المحلية (local variables) في أي دالة [وهي المتغيرات المعلَن عنها في قائمة الوسطاء أو في جسم الدالة] تكون لها عادةً فترات تخزين أوتوماتيكية .

الكلمة **auto** تعلن صراحةً (explicitly) عن متغيرات ذوات فترة تخزين أوتوماتيكي . فمثلا الإعلان التالي

```
auto double x, y;
```

يشير إلى أن المتغيرين x, y من النوع **double** متغيران محليان أوتوماتيكيان ، وموجودان فقط في جسم الدالة التي ظهر فيها هذا الإعلان .

وافتراضيا (by default) فإن المتغيرات المحلية يكون لها فترة تخزين أوتوماتيكي . لذلك فإن كلمة **auto** نادرا ما تُستخدم. وسنشير بعد ذلك إلى المتغيرات ذوات فترة التخزين الأوتوماتيكي بـ "المتغيرات الأوتوماتيكية" اختصارا .

ونلاحظ أن التخزين الأوتوماتيكي هو إحدى وسائل الحفاظ على الذاكرة (conserving memory) وتوفيرها ، وذلك لأن المتغيرات الأوتوماتيكية تكون موجودة فقط حين نحتاج إليها . فهي تُنشأ عندما ندخل الدالة التي تم تعريفها فيها ، وتُدمَّر حينما نخرج من الدالة.

والبيانات (data) في صيغة لغة الماكينة لبرنامج ما (machine language version of a program) يتم تحميلها (loaded) في مسجِّلات (registers) وذلك لإجراء الحسابات وعمليات التشغيل الأخرى .

ويلاحظ أنه يمكن وضع محدِّد طبقة التخزين **register** قبل الإعلان عن متغير أوتوماتيكي لحث البرنامج المترجم (compiler) على الاحتفاظ بالمتغير في أحد مسجلات الحاسوب عالية السرعة (computer's high-speed hardware registers) . وحين نحتفظ بالمتغيرات التي تُستخدم بكثرة – كالعدادات (counters) والمجاميع (totals) – في مسجلات الحاسوب فإننا نستريح من تكرار عمليتي تحميل (loading) المتغيرات من الذاكرة إلى المسجلات ، ثم إعادة النتائج لتخزينها في الذاكرة .

وقد يتجاهل البرنامج المترجِم الإعلانات **register** . فمثلا قد لا يكون هناك عدد كاف من المسجلات المتوفرة ليستخدمها البرنامج المترجم . والإعلان التالي

```
register int counter = 1;
```

يطلب وضع المتغير الصحيح **counter** في أحد مسجلات الحاسوب، وإعطاءه القيمة الابتدائية 1.

وكلمة **register** لا تُستخدم إلا مع المتغيرات الأوتوماتيكية فقط . وغالبا ما تكون إعلانات **register** غير ضرورية ، حيث أن البرامج المترجِمة المُثْلَى (optimizing compilers) اليوم تكون قادرة على التعرُّف (recognizing) على المتغيرات المستخدَمة بكثرة (frequently used) ، كما يمكنها أن تقرر وضعها – بالتالي – في مسجلات (registers) دون الحاجة إلى إعلان **register** من المبرمج .

(ب) الكلمتان **extern, static** تُستخدمان للإعلان عن أسماء تعريفية لمتغيرات ودوال ذوات فترة تخزين استاتيكي . والأسماء التعريفية لفترة التخزين الاستاتيكي تتواجد (exist) من

لحظة بداية تنفيذ البرنامج . وبالنسبة للمتغيرات فإن حيّز التخزين يتم تخصيصه (storage is allocated) ويُعطَى القيم الابتدائية مرة واحدة (initialized once) عندما يبدأ تنفيذ البرنامج . وبالنسبة للدوال فإن اسم الدالة يتواجد (exist) عندما يبدأ تنفيذ البرنامج . ورغم أن المتغيرات وأسماء الدوال تتواجد من بداية تنفيذ البرنامج ، إلا أن ذلك لا يعني أن هذه الأسماء التعريفية يمكن الوصول إليها من أي موضع في البرنامج بأكمله ، ففترة التخزين (storage duration) والمجال (scope) موضوعان مختلفان كما سنرى بإذن الله بعد قليل حين ندرس قواعد المجال .

وهناك نوعان من الأسماء التعريفية ذوات فترة التخزين الاستاتيكي :

- أسماء تعريفية خارجية (external identifiers) : مثل المتغيرات الشاملة (global variables) وأسماء الدوال .
- المتغيرات المحلية (local variables) التي يعلَن عنها باستخدام محدِّد طبقة التخزين **static** .

والمتغيرات الشاملة وأسماء الدوال تعد من طبقة التخزين **extern** افتراضا (by default) . والمتغيرات الشاملة يتم إنشاؤها (created) بوضع الإعلانات عن المتغيرات خارج تعريف أي دالة ، وتحتفظ هذه المتغيرات الشاملة بقيمها خلال / طوال تنفيذ البرنامج . ويمكن الإشارة (referencing) إلى المتغيرات الشاملة والدوال من أي دالة تتبع (follows) إعلاناتها (declarations) أو تعريفاتها (definitions) في الملف (file) . وهذا هو أحد أسباب استخدام نماذج الدوال – فعندما نشمل stdio.h (include) في برنامج يستدعي **printf**، فإن نموذج الدالة يوضع في بداية الملف لنجعل الاسم **printf** معلوماً لبقية الملف .

والمتغيرات المحلية التي نعلن عنها باستخدام كلمة **static** تظل معروفة فقط في الدالة التي تم تعريفها فيها ، ولكنها [أي المتغيرات المحلية الاستاتيكية (**static** local variables)] – بخلاف المتغيرات الأوتوماتيكية – تحتفظ (retain) بقيمها عندما يتم الخروج من الدالة (function is exited) . وعند استدعاء الدالة في المرة التالية ، فإن المتغير المحلي الاستاتيكي (**static** local variable) يحتوي عندئذ على القيمة التي كان يحتوي عليها عندما تم الخروج من الدالة آخر مرة . العبارة التالية

```
static int count = 1;
```

تعلن عن متغير محلي اسمه **count** وأنه استاتيكي **static** وتعطيه القيمة الابتدائية 1 .

وجميع المتغيرات العددية (numeric variables) ذوات فترة التخزين الاستاتيكي تعطَى قيما ابتدائية صفرية إذا لم يعطها المبرمج صراحة (explicitly) قيما ابتدائية . ومن الخطأ استخدام أكثر من محدِّد طبقة تخزين واحد لأي اسم تعريفي .

قواعد المجال Scope Rules

مجال (scope) أي اسم تعريفي هو جزء البرنامج الذي يمكن فيه الإشارة إلى هذا الاسم التعريفي . مثلا : حين نعرِّف متغيرا محليا في قالبٍ ما ، فلا يمكن الإشارة إلى هذا المتغير المحلي إلا في هذا القالب أو في قوالب متداخلة (nested blocks) داخل هذا القالب . والمجالات الأربعة لأي اسم تعريفي هي :

(أ) مجال دالة (function scope)
(ب) مجال ملف (file scope)
(ج) مجال قالب (block scope)
(د) مجال نموذج دالة (function – prototype scope)

(أ) " العلامات " (labels) [العلامة (label) هي اسم تعريفي يليه نقطتان إحداهما فوق الأخرى (colon) مثل **start:**]هي الأسماء التعريفية الوحيدة التي لها مجال دالة **(function scope)**. والعلامات يمكن أن تُستخدم في أي موضع في الدالة التي ظهرت فيها ، ولكن لا يمكن الإشارة إليها من خارج جسم الدالة . وتُستخدم العلامات في عبارات **switch** كعلامات الحالة labels **case** وفي عبارات **goto** .

(ب) وأي اسم تعريفي تم الإعلان عنه خارج أي دالة يكون له مجال ملف (file scope) . ومثل هذا الاسم التعريفي يكون "معروفا" (known) [أي "يمكن الوصول إليه" (accessible)] في جميع الدوال من النقطة التي تم عندها الإعلان عنه وحتى نهاية الملف . وجميع المتغيرات الشاملة وتعريفات الدوال ونماذج الدوال الموضوعة خارج دالة يكون لها مجال ملف (file scope) .

(ج) والأسماء التعريفية المعرَّفة داخل قالب يكون لها مجال قالب **(block scope)** . وينتهي مجال القالب عند القوس الأيمن (}) لإنهاء القالب (terminating right brace of the block) . والمتغيرات المحلية المعرَّفة في بداية أي دالة يكون لها مجال قالب ، كما أن وسطاء الدالة (function parameters) [والذين تعتبرهم الدالة متغيرات محلية] يكون لهم مجال قالب . وقد يحتوي أي قالب على تعريفات متغيرات . وعندما تكون هناك قوالب متداخلة (nested

(outer blocks) ، وهنـاك اسـم تعريفـي (identifier) في قالـب خـارجي (outer block) واسم تعريفي في قالب داخلي (inner block) والاسمان التعريفيان متطابقان (أي لهما الاسم نفسه) فإن الاسم التعريفي في القالب الخارجي يُخْفَى (is hidden) حتى ينتهي (terminates) القالب الداخلي . وهذا يعني أنه طوال تنفيذ القالب الداخلي فإن القالب الداخلي يرى قيمة الاسم التعريفي المحلـي الخـاص بـه (its own local identifier) ولـيس قيمـة سَـمِيِّهِ (identically named) الاسـم التعريفـي في القالـب الخـارجي المحـيط (enclosing block) بالقالب الداخلي . والمتغيرات المحلية المعلَنة **static** يكون لها أيضا مجال قالب (block scope) رغم أنها تتواجد (exist) من لحظة بداية تنفيذ البرنامج . وهكذا فإن فترة التخزين لا تؤثر على مجال أي اسم تعريفي.

(د) الأسمـاء التعريفيـة الوحيـدة الـتي لهـا مجـال نمـوذج دالـة **(function prototype scope)** هي تلك المستخدمة في قائمة وسطاء نموذج دالة . وكما ذكرنا سابقا فإن نماذج الدوال لا تتطلب أسماء (names) في قائمة الوسطاء ، وإنما الأنواع (types) فقط هي المطلوبة . وإذا استُخدم / ذُكر أي اسم في قائمة وسطاء نموذج دالة فإن البرنامج المترجم (compiler) يتجاهل (ignores) هـذا الاسـم . والأسمـاء التعريفيـة المسـتخدمة في نمـوذج دالـة يمكـن أن يعـاد اسـتخدامها (reused) في أي موضـع آخـر في البرنـامج بـدون أي التبـاس (ambiguity) .

مثال ٥ – ٧ : البرنامج التالي يوضح تطبيق قواعد المجال باستخدام متغير شامل ، ومتغير محلي أوتوماتيكي ، ومتغير محلي استاتيكي ، وجميع المتغيرات تحمل الاسم نفسه x .

```
/* Example 5.7
   A scoping example */
#include <stdio.h>

void useLocal( void );         /* function prototype */
void useStaticLocal( void ); /* function prototype */
void useGlobal( void );       /* function prototype */

int x = 1; /* global variable */

/* function main begins program execution */
```

```
int main()
{
  int x = 5; /* local variable to main */

  printf("local x in outer scope of main is %d\n", x );

  { /* start new scope */
    int x = 7; /* local variable to new scope */

    printf( "local x in inner scope of main is %d\n", x );
  } /* end new scope */

  printf( "local x in outer scope of main is %d\n", x );

  useLocal();             /* useLocal has automatic local x */
  useStaticLocal();   /* useStaticLocal has static local x */
  useGlobal();          /* useGlobal uses global x */
  useLocal();           /* useLocal reinitializes automatic local x */
  useStaticLocal(); /* static local x retains its prior value */
  useGlobal();        /* global x also retains its value */

  printf( "\nlocal x in main is %d\n", x );

  return 0; /* indicates successful termination */

} /* end main */

/* useLocal reinitializes local variable x during each call */
void useLocal( void )
{
  int x = 25;  /* initialized each time useLocal is called */

  printf( "\nlocal x in useLocal is %d after entering useLocal\n", x );
  x++;
  printf( "local x in useLocal is %d before exiting useLocal\n", x );
} /* end function useLocal */

/* useStaticLocal initializes static local variable x only the first time
   the function is called; value of x is saved between calls to this
```

```
   function */
void useStaticLocal( void )
{
   /* initialized only first time useStaticLocal is called */
   static int x = 50;

   printf( "\nlocal static x is %d on entering useStaticLocal\n", x );
   x++;
   printf( "local static x is %d on exiting useStaticLocal\n", x );
} /* end function useStaticLocal */

/* function useGlobal modifies global variable x during each call */
void useGlobal( void )
{
   printf( "\nglobal x is %d on entering useGlobal\n", x );
   x *= 10;
   printf( "global x is %d on exiting useGlobal\n", x );
} /* end function useGlobal */
```

```
Local x in outer scope of main is 5
Local x in inner scope of main is 7
Local x in outer scope of main is 5

Local x in a is 25 after entering useLocal
Local x in a is 26 before exiting useLocal

Local  static x is 50 on entering useStaticLocal
Local  static x is 51 on exiting useStaticLocal

global x is 1 on entering useGlobal
global x is 10 on exiting useGlobal

Local x in a is 25 after entering useLocal
Local x in a is 26 before exiting useLocal

Local  static x is 51 on entering useStaticLocal
Local  static x is 52 on exiting useStaticLocal

global x is 10 on entering useGlobal
global x is 100 on exiting useGlobal

local x in main is 5
```

- البرنامج يعرِّف في البداية متغيرا شاملا x ، ويعطيه القيمة الابتدائية 1 . وهذا المتغير الشامل يُخفَى (is hidden) في أي قالب (أو أي دالة) يُعرَّف فيه متغير اسمه x .
- وفي الدالة الرئيسية **main** يُعرَّف متغير محلي **x** ويُعطَى القيمة الابتدائية 5. ثم نطبع هذا المتغير **x** لنبيِّن أن المتغير الشامل **x** مخفي في الدالة **main** .
- ثم نعرِّف قالبا جديدا في الدالة **main** وفيه متغير محلي آخر **x** يعطَى القيمة الابتدائية 7 . وهذا المتغير يُطبع ليبيِّن أنه يُخفِي المتغير **x** في القالب الخارجي في **main** . والمتغير **x** الذي قيمته 7 يُدمَّر تلقائيا / أوتوماتيكيا بمجرد الخروج من القالب الداخلي (الجديد) ، ونَطبع مرة أخرى المتغير المحلي **x** الموجود في القالب الخارجي في **main** لنبيِّن أنه لم يعد مخفيا .
- ويعرِّف البرنامج ثلاث دوال لا تأخذ أي منها وسيطا ولا تعيد شيئا .
- الدالة **uselocal** تعرِّف متغيرا أوتوماتيكيا **x** وتعطيه القيمة الابتدائية 25 . وعندما تُستدعَى الدالة **uselocal** تتم طباعة المتغير ، وزيادة قيمته (incrementing) بواحد ، وطباعته مرة أخرى قبل الخروج من الدالة . وكلما تُستدعي هذه الدالة فإن المتغير الأوتوماتيكي **x** يعاد إعطاؤه القيمة الابتدائية 25 .
- الدالة **useStaticLocal** تعرِّف متغيرا استاتيكيا **x** (**static** variable) ، وتعطيه القيمة الابتدائية 50 . ومعلوم أن المتغيرات المحلية المعلَن عنها أنها متغيرات استاتيكية **static** تحتفظ بقيمها حتى عندما تكون خارج المجال (out of scope) . وعندما تُستدعَى الدالة **useStaticLocal** فإنها تطبع قيمة **x** ، وتزيدها بواحد ، وتطبعها مرة أخرى قبل أن نخرج من الدالة . وعند الاستدعاء التالي لهذه الدالة فإن المتغير المحلي الاستاتيكي x **(static)** سيكون محتويا القيمة 51 .
- الدالة **useGlobal** لا تعرِّف أي متغيرات . ولذلك فإنها حين تشير إلى متغير **x** فإن المتغير الشامل **x** هو الذي يُستخدم . وحينما تُستدعى الدالة **useGlobal** فإن المتغير الشامل يُطبع ، ويُضرب في 10 ، ويُطبَع مرة أخرى قبل أن نخرج من الدالة . وحينما تُستدعى الدالة **useGlobal** مرة أخرى ، فإن المتغير الشامل يكون محتفظا بقيمته المعدَّلة 10 .
- وأخيرا يطبع البرنامج المتغير المحلي **x** في الدالة **main** مرة أخرى ليبين أن أيًّا من استدعاءات الدوال لم يغير من قيمة **x** لأن جميع الدوال كانت تشير إلى متغيرات في مجالات أخرى .

تمرينات رقم ٥

٥-١) اختر الإجابة الصحيحة فيما يلي :

i) **جميع دوال المكتبة الرياضية تعيد نوع البيانات:**

أ) long double　　ب) int

ج) float　　د) double

ii) **ماذا يحدث عندما لا يشتمل (include) البرنامج على** math.h **بينما يستخدم دوال من المكتبة الرياضية ؟**

أ) **خطأ ترجمة** (compilation error).

ب) **خطأ تنفيذ** (execution error).

ج) **خطأ منطقي** (logic error).

د) **قد تحدث نتائج غريبة .**

iii) **نفرض أن** a = 7.0,　b = 7.0,　c = 6.0

ماذا تطبع العبارة التالية؟

```
printf( "%.2f", sqrt( a + b * c));
```

أ) 49　　ب) 7.00

ج) 7　　د) 49.00

iv) **ما هي قيمة** fabs (-5.0) **؟**

أ) 5　　ب) 5.0　　ج) -5　　د) -5.0

v) **ما هي الدالة التي لا يشملها ملف المقدمة** < math.h > **؟**

أ) pow　　ب) floor

ج) ln　　د) log10

vi) **أي الدوال التالية لا تحتوي على أخطاء ؟**

أ)

```
void printnum ( int x )
{
        print ( "%i", x );
        return x;
}
```

ب)
```
int cube( int s )
{
        int s;
        return ( s * s * s );
}
```

ج)
```
double triple ( float n )
        return ( 3 * n  );
```

د)
```
double circumference ( int r );
        return (5.14 * 2 * r );
```

vii) نوع (type) أي وسيط (parameter) حُذِف نوعه في تعريف دالة يكون :

أ) int
ب) double
ج) long
د) float

viii) أكثر الاصطلاحات اختصارا لتعريف وسيطي دالة (function perameters) x , y من النوع double هو:

أ) x, y
ب) x, double y
ج) double x, y
د) double x, double y

ix) وضع فاصلة منقوطة بعد القوس الأيمن الذي يحيط بقائمة الوسطاء في تعريف دالة يُعَدُّ خطأ

أ) منطقيا (logic)
ب) تركيبيا (syntax)
ج) قاتلا وقت التنفيذ (fatal runtime)
د) غير قاتل وقت التنفيذ (nonfatal runtime)

x) بالنسبة للبرنامج التالي اذكر ما إذا كان مجال (scope) كل عنصر من العناصر التالية هو :

أ) مجال دالة (function scope)
أم ب) مجال ملف (file scope)
أم ج) مجال قالب (block scope)

أم د) مجال نموذج دالة (function prototype scope)

(١) المتغير **x** في الدالة **main** .

(٢) المتغير **y** في الدالة **cube** .

(٣) الدالة **cube** .

(٤) الدالة **main** .

(٥) نموذج الدالة **cube** .

(٦) الاسم التعريفي **y** في نموذج الدالة **cube** .

```
#include <stdio . h>
int cube( int y ) ;

int main ( )
{
   int x ;

   for ( x = 1; x <= 10; x++ )
      printf ( "%d\n" , cube ( x )  ) ;
   return  0 ;
}

int cube ( int y )
{
      return y * y * y ;
}
```

٥-٢) اكتب برنامجا يختبر صحة نتائج استدعاءات دوال المكتبة الرياضية (math library function calls) المبينة في "جدول الدوال المكتبية الرياضية القياسية شائعة الاستخدام" ، أي يختبر ما اذا كانت هذه الاستدعاءات تُنتج فعلا القيم المبينة .

٥-٣) اكتب مقدمة دالة (function header) لكل من الدوال التالية:

أ) دالة **hypotenuse** تأخذ وسيطين **side1, side2** ذَوَي نقطة عائمة من الدِّقة المضاعفة (double – precision floating point arguments) ، وتعيد نتيجة ذات نقطة عائمة من الدقة المضاعفة .

ب) دالة **smallest** تأخذ ثلاثة أعداد صحيحة **x , y , z** ، وتعيد عددا صحيحا .

ج) دالة **instructions** لا تستقبل أي وسيط ، ولا تعيد أي قيمة . [ملاحظة : مثل هذه الدوال تُستخدم عامة لعرض تعليمات للمستخدِم] .

د) دالة **intToFloat** تأخذ وسيطا صحيحا **number** ، وتعيد نتيجة ذات نقطة عائمة .

٥-٤) اكتب نموذج دالة (function prototype) لكل من الدوال الأربع المذكورة في السؤال السابق ٥ - ٣ - أ ، ب ، ج ، د .

٥-٥) اكتب إعلانا (declaration) لكل مما يلي :

أ) عدد صحيح **count** يجب أن يُحفظ في مسجِّل (maintained in a register) . واعط **count** القيمة الابتدائية 0 .

ب) متغير **lastVal** ذو نقطة عائمة يجب أن يحتفظ بقيمته (retains its value) بين الاستدعاءات المختلفة للدالة التي تم تعريفه فيها .

ج) عدد صحيح خارجي (external integer number) يجب أن يكون مجاله (scope) مقيدا في بقية الملف (restricted to the remainder of the file) الذي تم تعريف العدد فيه .

٥-٦) أوجد الخطأ في كل من قطع البرامج التالية ، ووضح كيف يمكن تصحيح الخطأ .

أ)
```
int g( void )
{
  printf ( “inside function g\n” );
  int h( void)
  {
    printf ( “inside function h\n” );
  }
}
```

ب)
```
int sum( int x, int y )
{
  int result;
  result = x + y;
}
```

ج)
```
void f( float a ) ;
{
```

```
  float a ;
  printf ( “%f”, a ) ;
}
```

د)
```
void product ( void )
{
  int a, b, c, result ;
  printf ( "enter three integers :  "  )
  scanf (  "%d%d%d" , &a ,  &b ,  &c  ) ;
  result = a * b * c ;
  printf (  "result  is  %d " , result  ) ;
  return result ;
}
```

هـ)
```
double cube ( float ) ;    /* function prototype */
...
cube ( float number )    /* function definition  */
{
  return number * number * number :
}
```

و)
```
register auto int x = 7 ;
```

ز)
```
int randomNumber  = srand ( ) ;
```

ح)
```
double  y  = 123.45678;
int  x;
x  =  y ;
printf (  "%f\n",  (double)  x  ) ;
```

ط)
```
double square ( double number )
{
  double number ;

  return number  *  number ;
}
```

٥-٧) اكتب قيمة x الناتجة بعد تنفيذ كل من العبارات التالية:

أ)
```
x  =  fabs( 7.5 ) ;
```

ب)
```
x  =  floor( 7.5 ) ;
```

ج)
```
x  =  fabs( 0.0 ) ;
```

د) `x = ceil( 0.0 ) ;`

هـ) `x = fabs( -6.4 ) ;`

و) `x = ciel( -6.4 ) ;`

ز) `x = ceil( - fabs ( -8 + floor ( -5.5 ) ) ) ;`

٥-٨) أحد تطبيقات الدالة floor تقريب (rounding) أي عدد x إلى أقرب عدد صحيح y (nearest integer) باستخدام العبارة :

```
y = floor ( x + .5 ) ;
```

اكتب برنامجا يقرأ قيم خمسة أعداد، ويستخدم العبارة السابقة لتقريب كل من هذه الأعداد إلى أقرب عدد صحيح . واطبع كلا من العدد الأصلي والعدد المقرَّب .

٥-٩) يمكن استخدام الدالة floor لتقريب أي عدد x إلى عدد مُحَدَّد من الأرقام / المواضع العشرية (a specific decimal place) . فمثلا العبارة

```
y = floor ( x * 10 + .5 ) / 10 ;
```

تُقَرِّب x إلى رقم عشري واحد [الموضع العشري (tenths position)] ، أي إلى أول موضع يمين العلامة العشرية (decimal point) . والعبارة

```
y = floor ( x * 100 + .5 ) / 100 ;
```

تُقَرِّب x إلى رقمين عشريين [الموضع المئوي (hundredths position)] ، أي إلى الموضع الثاني يمين العلامة العشرية .

اكتب برنامجا يعرِّف الدوال الأربع التالية لتقريب عدد x بالطرق المختلفة المبينة فيما يلي :

أ) تقريب لأقرب عدد صحيح `roundToInteger ( number )`

ب) تقريب لأقرب موضع عشري `roundToTenths ( number )`

ج) تقريب لأقرب موضع مئوي `roundToHundredths ( number )`

د) تقريب لأقرب موضع ألفي `roundToThousandths ( number )`

ويقرأ البرنامج بعض القيم ويقرب كلاً منها إلى أقرب عدد صحيح ، وإلى أقرب موضع عشري (nearest tenth) ، وإلى أقرب موضع مئوي (nearest hundredth) ، وإلى أقرب موضع ألفي (nearest thousandth) .

يبدأ البرنامج بقراءة عدد القيم (count) المطلوب تقريبها ، ثم يطبع كلا من القيمة الأصلية – التي قرأها – وقيمها المقربة .

٥-١٠) اكتب عبارة تُسند عددا صحيحا عشوائيا (a random integer) للمتغير **n** في المدى (range) المبيَّن فيما يلي :

أ) $1 \le n \le 2$

ب) $1 \le n \le 100$

ج) $0 \le n \le 9$

د) $1000 \le n \le 1112$

هـ) $-1 \le n \le 1$

و) $-3 \le n \le 11$

٥-١١) لكل مجموعة من مجموعات الأعداد الصحيحة التالية اكتب عبارة واحدة تطبع عددا عشوائيا من المجموعة :

أ) 2, 4, 6, 8, 10

ب) 3, 5, 7, 9, 11

ج) 6, 10, 14, 18, 22

٥-١٢) عَرِّف دالة تُدعى **hypotenuse** تحسب طول الوتر في مثلث قائم الزاوية (right triangle) إذا أُعطِي ضلعاه الآخران .

استخدم هذه الدالة في برنامج يقوم بحساب طول وتر كل من المثلثات التالية . الدالة تأخذ وسيطين من النوع **double** وتعيد الوتر المقابل من النوع **double** .

المثلث triangle	الضلع الأول Side 1	الضلع الثاني Side 2
1	3.0	4.0
2	5.0	12.0
3	8.0	15.0

٥-١٣) اكتب دالة **integerPower (base, exponent)** تعيد قيمة $\mathbf{base}^{\mathbf{exponent}}$

مثلا : **integerPower (3, 4) = 3 * 3 * 3 *3**

افرض أن **exponent** عدد صحيح موجب ، وأن **base** عدد صحيح .

والدالة **integerPower** يجب ان تستخدم عروة **for** للتحكم في إجراء الحسابات ، ولا تستخدم أي دالة مكتبية رياضية .

٥-١٤) اكتب دالة **multiple** تأخذ وسيطين عددين صحيحين ، وتحدد ما إذا كان العدد الثاني أحد مضاعفات العدد الأول أم لا . فإن كان أحد مضاعفاته فإن الدالة تعيد 1 (true) ، وإن لم يكن فإن الدالة تعيد 0 (false) . واستخدم هذه الدالة في برنامج يقرأ ثلاثة أزواج من أعداد صحيحة (pairs of integers) ، ويحدد لكل زوج ما إذا كان العدد الثاني أحد مضاعفات العدد الأول أم لا .

٥-١٥) اكتب برنامجاً يقرأ سلسلة (series) من ثلاثة أعداد صحيحة، ويمررها واحدا واحدا إلى دالة تُدعى **even** تستخدم مؤثر الباقي (remainder operator) لتحديد ما إذا كان أي عدد صحيح زوجيا أم لا . والدالة تأخذ وسيطا عددا صحيحا وتعيد 1 إن كان العدد الصحيح زوجيا ، وما عدا ذلك تعيد 0.

٥-١٦) اكتب دالة **square** تعرض على الحافة اليسرى (left margin) من الشاشة مربعا مصمتا (solid square) من نجوم (asterisks) طول ضلعه يحدده وسيط صحيح **side** (integer parameter) . واختبر الدالة **square** بدالة رئيسية تقرأ طول ضلع المربع وتستدعي الدالة **square** .

مثلا: اذا كان **side** مساويا 4 فإن الدالة **square** تعرض الشكل التالي:

```
Enter side :  4
****
****
****
****
```

٥-١٧) اعد حل السؤال السابق مع تعديل الدالة **square** بحيث تعرض مربعا من أي شكل اختياري يحدده رمز (character) يحتويه وسيط رمزي **fillCharacter** (character parameter) .

مثلا : إذا كان **side** مساويا 5 ، وكان **fillCharacter** هو " # " ، فإن الدالة **square** تطبع الشكل التالي:

```
Enter a character and the side length :  #  5
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

٥-١٨) اكتب الدوال الصحيحة (integer functions) التالية :

أ) الدالة **celsius** التي تعيد درجة الحرارة المئوية المكافئة(Celsius equivalent) لدرجة حرارة فهرنهيتية (a Fahrenheit temperature) .

ب) الدالة **fahrenheit** التي تعيد درجة الحرارة الفهرنهيتية المكافئة (Fahrenheit equivalent) لدرجة حرارة مئوية (a Celsius temperature) .

ج) الدالة الرئيسة **main** التي تستخدم الدالتين السابقتين لطباعة جدولين :

I) الاول يعرض درجات الحرارة الفهرنهيتية المكافئة لجميع درجات الحرارة المئوية من 0 إلى 100 درجة .

II) الثاني يعرض درجات الحرارة المئوية المكافئة لجميع درجات الحرارة الفهرنهيتية من 32 إلى 212 درجة .

٥-١٩) اكتب دالة **smallest3** تعيد أصغر قيمة من قيم ثلاثة أعداد ذوي نقطة عائمة (3 floating point numbers) . واكتب دالة رئيسية تقرأ ثلاثة أعداد ذوي نقطة عائمة وتوجد أصغرها باستخدام الدالة **smallest3** .

٥-٢٠) يقال لعدد صحيح إنه " عدد تام " (perfect number) إذا كان مجموع عوامله/ قواسمه (factors / divisors) [بما فيهم 1 ، ولكن باستثناء العدد نفسه] مساويا العدد نفسه . فمثلا العدد 6 عدد تام لأن :

$$1 + 2 + 3 = 6$$

اكتب دالة **perfect** تحدد ما إذا كان وسيطها (parameter) عددا تاما أم لا. واستخدم هذه الدالة في برنامج يحدِّد ويطبع قيم جميع الأعداد التامة بين 1 و 1000.

٥-٢١) يُقال لعدد صحيح إنه " عدد أوَّلي" (prime number) اذا كان يقبل القسمة – بدون باق – (divisible by) على 1 ونفسه فقط . فمثلا 2, 3, 5, 7 أعداد أولية ، بينما 4, 6, 8, 9 ليست أعدادا أولية .

اكتب دالة **prime** تحدد ما إذا كان عددٌ ما أوليا أم لا .

ثم استخدم هذه الدالة في برنامج يحدد ويطبع جميع الأعداد الأولية بين 1 و 10,000.

٥-٢٢) القاسم المشترك الأعظم (greatest common divisor) (gcd) لعددين صحيحين هو أكبر عدد صحيح يَقْسِم بالتساوي (evenly divides) كلا من العددين. مثلا القاسم المشترك الأعظم للعددين 20 , 12 هو 4 . اكتب دالة **gcd** تعيد القاسم المشترك الأعظم لعددين صحيحين . واكتب برنامجا يقرأ خمسة أزواج من أعداد صحيحة ، ويعين لكل زوج القاسم المشترك الأعظم باستخدام الدالة **gcd** .

٥-٢٣) اكتب دالة **qualityPoints** تستقبل عددا صحيحا **average** يمثل الدرجة المتوسطة لطالب (student's average) [بين 0 و 100] ، وتعيد عددا صحيحا [بين 0 و 4] يمثل درجة الطالب (عدد النقاط) المقابلة على مقياس الأربع نقاط (4 point scale) بناءً على الجدول التالي :

عدد النقاط المقابلة	الدرجة المتوسطة
4	90 - 100
3	80 - 89
2	70 - 79
1	60 - 69
0	0 - 59

واكتب برنامجا يقرأ الدرجات المتوسطة لخمسة طلاب ، ويستخدم الدالة السابقة لتعيين وطباعة الدرجة المقابلة لكل من هذه الدرجات المتوسطة الخمس على مقياس الأربع نقاط .

٥-٢٤) اكتب دالة **distance** تحسب طول المسافة بين النقطتين (x2 , y2) و (x1 , y1)، بحيث تكون جميع الأعداد والقيمة المعادة من النوع **double** . واكتب برنامجا يقرأ إحداثيات نقطتين ، ويستخدم الدالة **distance** لحساب وطباعة طول المسافة بين النقطتين .

٥-٢٥) يفرض موقف للسيارات (parking garage) حدّاً أدنى (minimum fee) قدره 2.00$ لرسوم (charges) إيقاف سيارة لفترة لا تتجاوز ثلاث ساعات . ويفرض رسما إضافيا قدره 0.50 $ (additional) في الساعة ، وذلك لكل ساعة أو جزء من الساعة في الفترة التي تتجاوز الثلاث ساعات . والحد الأقصى للرسوم (maximum charge) خلال أي فترة 24 ساعة هو 10.00$. وافرض أن أي سيارة لاتُترك في الموقف لفترة تزيد عن 24 ساعة متصلة .

اكتب دالة **calculateCharges** تحسب رسوم إيقاف سيارة إذا عُلم عدد الساعات التي تُرِكت خِلالها في الموقف . واكتب برنامجا يقرأ عدد ساعات إيقاف كل من ثلاث سيارات في الموقف أمس ، ويستخدم الدالة **calculateCharges** لحساب وطباعة رسوم إيقاف (parking charges) كل من السيارات الثلاث . كذلك يوجد البرنامج ويطبع العدد الإجمالي لساعات إيقاف السيارات والرسوم الإجمالية (total charges) التي تم تحصيلها أمس من هذه السيارات . ويُراعَى طباعة النتائج في صيغة جدولية (tabular format) بحيث تظهر المخرجات في الصيغة التالية :

```
Enter the hours parked for 3 cars: 1.5 4.0 24.0
  Car        Hours        Charge
    1          1.5          2.00
    2          4.0          2.50
    3         24.0         10.00
TOTAL         29.5          14.50
```

٥-٢٦) اكتب دالة **integerPower (b, e)** تعيد قيمة b^e . فمثلا :

integerPower (3, 4) = 3^4 = 3*3*3*3 = 81

افرض أن الأس **e** (exponent) عدد صحيح غير صفري موجب (positive integer) ، والأساس **b** (base) عدد صحيح (integer) . الدالة **integerPower** يجب أن تستخدم عروة **for** لإجراء حساباتها ، ولا تستخدم أي دالة مكتبية رياضية .

اكتب كذلك دالة رئيسية تقرأ عددين صحيحين **exponent** , **base** وتستخدم الدالة **integerPower** لحساب

$$base^{exponent}$$

٥-٢٧-أ) اكتب دالة **quotient** تعيد الجزء الصحيح (integer part) من خارج قسمة العدد الصحيح **a** على العدد الصحيح **b** .

ب) اكتب دالة **remainder** تعيد الباقي الصحيح عند قسمة العدد الصحيح **a** على العدد الصحيح **b** .

ج) اكتب دالة رئيسية تقرأ عددا صحيحا بين 1 و 32767 وتطبعه – باستخدام الدالتين السابقتين – كمتسلسلة أرقام (series of digits) ، يفصل بين كل رقمين متتاليين منهما فراغان (2 spaces) . مثلا العدد الصحيح 4562 يجب أن يُطبع هكذا :

4 5 6 2

٥-٢٨- أ) اكتب دالة **seconds** تأخذ الوقت (time) ممثَّلاً بثلاثة وسطاء / أعداد صحيحة (للساعات والدقائق والثواني) ، وتعيد عدد الثواني المقابلة (منذ آخر مرة كانت الساعة فيها 12) .

ب) استخدم الدالة السابقة لحساب كمية الوقت بالثواني بين وقتين (تتم قراءتهما) يقعان خلال دورة زمنية واحدة تبلغ 12 ساعة (one 12- hour cycle of the clock) .

٥-٢٩) تلعب الحواسيب اليوم دورا متزايدا في التعليم . اكتب برنامجا يساعد تلميذا في مدرسة ابتدائية يتعلم جدول الضرب .

استخدم الدالة **rand** لتوليد عددين صحيحين موجبين يتكون كل منهما من رقم واحد (two positive one – digit integers) . ثم يطبع البرنامج سؤالا مثل:

How much is 6 times 7 ?

ثم يطبع التلميذ الإجابة . ويختبر البرنامج إجابة التلميذ . فإن كانت صحيحة فإن البرنامج يطبع الرسالة "very good"، ثم يسأل سؤالا آخر لعملية ضرب أخرى. وإن كانت إجابة التلميذ خاطئة فإن البرنامج يطبع الرسالة "No. Please try again." ، ويسمح للتلميذ بتكرار الإجابة على السؤال نفسه إلى أن يجيب عليه الإجابة الصحيحة .

[ملاحظة: استخدام الحواسيب في التعليم يطلق عليه "التعليم بمساعدة الحاسوب" (Computer – Assisted Instruction) (CAI).

٥-٣٠) عدِّل برنامج السؤال السابق ٥-٢٩) بحيث يطبع تعليقات / رسائل مختلفة للإجابات الصحيحة ، وكذلك يطبع تعليقات / رسائل مختلفة للإجابات الخاطئة، كالتعليقات التالية:

تعليقات للإجابات الصحيحة :

Very good !
Excellent !
Nice work !
Keep up the good work !

تعليقات للإجابات الخاطئة:

No. Please try again.

Wrong. Try once more.
Don't give up !
No. keep trying.

استخدم مولِّد الأعداد العشوائية (random number generator) لاختيار عدد من 1 إلى 4 لينتقي تعليقا مناسبا لكل إجابة . واستخدم عبارة **switch** مع عبارات **printf** لطباعة التعليق .

٥-٣١) أ) اكتب دالة **Area** تحسب مساحة مثلث بدلالة أطوال أضلاعه a , b , c حيث :

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

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**.

٥-٣٢) اكتب دالة **totinc** توجد مجموع وسطائها الثلاثة **yards , feet , inches** (أطوال باليادرة والقدم والبوصة) ، أي توجد الطول الإجمالي محسوبا بالبوصة.

مثلا : totinc (3 , 1 , 8) = 128

(ملاحظة : اليادرة = ٣ أقدام ، والقدم = ١٢ بوصة).

٥-٣٣) اكتب دالة **totdys** توجد مجموع وسيطيها **days** (عدد الأيام) و **weeks** (عدد الأسابيع) محسوبا بالأيام.

مثلا : totdys (5 , 9) = 44

٥-٣٤) 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 :

$$a > bc , \quad b > ac , \quad c > ab$$

وما عدا ذلك فإن قيمة الدالة تكون false.

٥-٣٥) 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**.

٥-٣٦) اكتب دالة تعيد قيمة (من النوع **float**) **cubert** تستقبل وسيطا واحدا x عبارة عن عدد من النوع **float** ، وتوجد جذره التكعيبي .

٥-٣٧) 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).

٥-٣٨) اكتب دالة تعيد قيمة **float** تحسب مساحة شبه منحرف (trapezoid) إذا علم طولا قاعدتيه المتوازيتين **b2 , b1** وارتفاعه **h**. الدالة لها ٣ وسطاء (**b1 , b2 , h**).

٥-٣٩) 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.

٥-٤٠) 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.

٥-٤١) 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).

٥-٤٢) عَرِّف دالة تعيد قيمة لحساب

$$R = \sqrt{u^2 + v^2 + w^2}$$

ثم استخدمها لحساب :

$$A = \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

$$B = \sqrt{4x^4 + 9y^2 + 25z^2}$$

$$C = \sqrt{9 + Sin^2 y + u^2}$$

٥-٤٣) 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.

٤٤-٥) 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).

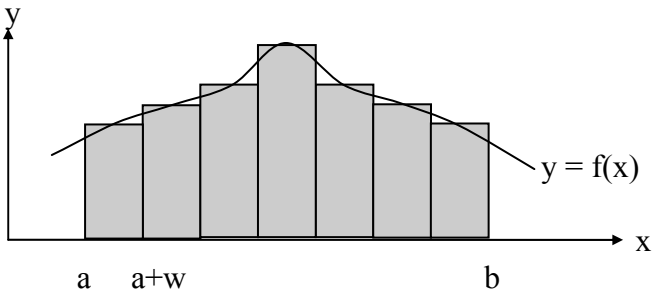
اكتب دالة تطبق هذه الطريقة بحيث تستقبل الدالة وسيطين : الوسيط الأول : عدد موجب (من النوع **float**) (المطلوب إيجاد جذره التربيعي) ، والوسيط الثاني : تخمين ابتدائي للجذر التربيعي.

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

٤٥-٥) 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 $[-2 , 3]$. اختبر البرنامج مستخدما قيما مختلفة لـ n. لاحظ أنه كلما زادت قيمة n ، كان التقريب أفضل.

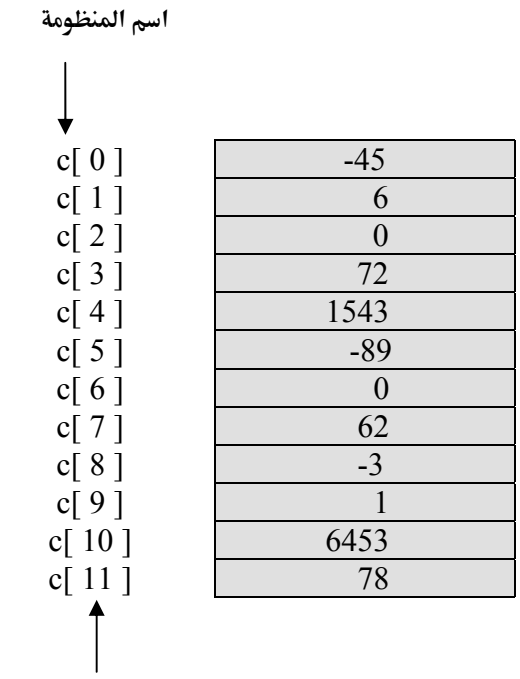
الفصل السادس

المنظومات

Arrays

المنظومة هي مجموعة مواقع في الذاكرة (group of memory locations) تحتوي عل مجموعة عناصر/وحدات بيانات (data items) مرتبطة ببعضها البعض ولها جميعها الاسم نفسه (same name) والنوع نفسه (same type) . وللإشارة إلى موقع / عنصر معين في المنظومة نحدِّد اسم المنظومة ورقم موقع (position number) العنصر المعين في المنظومة .

مثال ٦ – ١ : الشكل التالي (شكل ٦–١) يبيِّن منظومة صحيحة (integer array) تُدْعَى c . وهذه المنظومة تحتوي على 12 عنصر (element) . ورقم موقع أي عنصر يُكتب بين قوسين مربعين. والعنصر الأول في أي منظومة هو العنصر الصفري (zeroth element) ، أي أن العنصر الأول في المنظومة c يُشار إليه على أنه العنصر [0] c ، والعنصر الثاني هو [1] c، والعنصر السابع هو[6] c ، وعموما العنصر الذي ترتيبه i هو [1-i] c.



شكل (٦ – ١)

منظومة أعداد صحيحة مكونة من ١٢ عنصر

وأسماء المنظومات – كأسماء المتغيرات الأخرى – يمكن أن تشتمل فقط على حروف وأرقام وشُرط سفلية (letters, digits and underscores) .ولا يمكن لاسم منظومة أن يبدأ برقم .

ورقم موضع العنصر الذي يوضع بين قوسين مربعين يُطلق عليه "مؤشِّر" (subscript) . والمؤشر يجب أن يكون عددا صحيحا (integer) أو تعبيرا صحيحا . وإذا استخدم البرنامج تعبيرا حسابيا كمؤشر ، فإن قيمة التعبير تُحسب أولا لتحديد المؤشر . فمثلا إذا فرضنا أن

a = 5, b = 6

فإن العبارة c[a + b] += 2;

تضيف 2 إلى عنصر المنظومة [11] c .

وفي مثال ٦ – ١ تتكون المنظومة c من 12 عنصر نشير إليها بالأسماء c[0], c[1], c[2],…, c[11] . والقيمة المخزونة في c[0] (انظر شكل ٦ – ١) تساوي 45- ، وقيمة c[1] تساوي 6 ، وقيمة c[2] تساوي 0 ، وقيمة c[7] تساوي 62 ، وقيمة c[11] تساوي 78 . ولطباعة مجموع القيم المحتواة في العناصر الثلاثة الأولى في المنظومة c نكتب

printf("%d", c[0] + c[1] + c[2]);

ولقسمة قيمة العنصر السابع في المنظومة c على 2 ، وإسناد النتيجة إلى المتغير x ، فإننا نكتب

x = c[6] / 2;

ملاحظة : هناك فارق بين "العنصر السابع" (seventh element) (مثلا) في المنظومة c (وهو العنصر c[6]) ، و" العنصر رقم سبعة " (element seven) – أي العنصر الذي رقم موضعه 7 في المنظومة c – (وهو العنصر c[7] ، وهو العنصر الثامن في المنظومة c ، لأن مؤشرات المنظومة تبدأ بالصفر 0) .

قاعدة أولويات المؤثرات (Operator Precedence Rule)

يُعَدُّ القوسان [ ] (brackets) اللذان يحيطان بمؤشر منظومة (subscript of an array) أحد المؤثرات (operators) في لغة C ، ولهذا المؤثر مستوى الأولوية (level of precedence) نفسه الخاص بمؤثر استدعاء دالة (function call operator) [أي القوسان اللذان يوضعان بعد اسم الدالة لاستدعاء هذه الدالة] . والجدول التالي يبيِّن أولويات وطرق

تجميع (associativity) المؤثرات التي درسناها حتى الآن . والجدول يعرض المؤثرات بترتيب تنازلي للأولويات (decreasing order of precedence) من أعلى الجدول لأسفله (top to bottom) .

المؤثرات Operators	التجميع Associativity	النوع Type
[ ] ( )	من اليسار إلى اليمين	(الأعلى) (highest)
++ -- ! (type)	من اليمين إلى اليسار	أحادي (unary)
* / %	من اليسار إلى اليمين	ضَربي (multiplicative)
+ -	من اليسار إلى اليمين	جَمعي (additive)
< <= > >=	من اليسار إلى اليمين	علاقي (relational)
== !=	من اليسار إلى اليمين	تساوي (equality)
&&	من اليسار إلى اليمين	"و" المنطقية (logical and)
||	من اليسار إلى اليمين	"أو" المنطقية (logical or)
?:	من اليمين إلى اليسار	شَرطي (conditional)
= += -= *= /= %=	من اليمين إلى اليسار	إسناد (assignment)
,	من اليسار إلى اليمين	فاصلة (comma)

جدول أولويات المؤثرات

تعريف المنظومات Defining Arrays

تحتل المنظومات حيِّزا في الذاكرة . وعلى المبرمج أن يحدد نوع وعدد عناصر أي منظومة حتى يمكن للحاسوب أن يحجز الحيِّز المناسب في الذاكرة . فمثلاً كي نخبر الحاسوب أن يحجز (reserve) مواقع لمنظومة c مكونة من ١٢ عنصر عبارة عن أعداد صحيحة نستخدم التعريف

```
int c[ 12 ];
```

والتعريف التالي

```
int b[ 100 ], x[ 27 ];
```

يحجز مواقع لمائة 100 عنصر لمنظومة أعداد صحيحة **b** ، وسبعة وعشرين 27 عنصرا لمنظومة أعداد صحيحة **x** . ويمكن تعريف منظومات لتحتوي على عناصر من أنواع بيانات أخرى . فمثلا يمكن استخدام منظومة من النوع **char** لتخزين سلسلة رموز (a character string)[*].

أمثلة على المنظومات

الأمثلة التالية توضح بإذن الله تعالى كيفية تعريف المنظومات ، وكيفية إعطائها قيما ابتدائية (initializing arrays) ، وكيفية إجراء عمليات عامة لتشغيل ومعالجة المنظومات (manipulating arrays) .

مثال ٦ – ٢ : نفرض أن n منظومة أعداد صحيحة (integer array) مكونة من 10 عناصر . اكتب برنامجا يعطي عناصر n قيما ابتدائية صفرية ، ويطبع المنظومة بصيغة جدولية (tabular format) تبين رقم / موضع العنصر وقيمته .

الحل :

```
/*Example 6.2
  initializing an array */
#include <stdio.h<

/* function main begins program execution*/
int main()
}
  int n[ 10 ]; /* n is an array of 10 integers*/
  int i; /* counter */

 /*   initialize elements of array n to 0 */
  for ( i = 0; i < 10; i ++) {
    n[ i ] = 0; /* set element at location i to 0 */
  } /* end for */
```

(*) سنناقش بإذن الله في كتابنا التالي " البرمجة المتقدمة بلغة C " موضوع سلاسل الرموز والتشابه (similarity) بينها وبين المنظومات ، وكذلك العلاقة بين المنظومات وما يُعْرَف بالمؤشرات (pointers) .

```
   printf( "%s%13s\n", "Element", "Value" );

  /*  output contents of array n in tabular format */
   for ( i = 0; i < 10; i ++) {
     printf( "%7d%13d\n", i, n[ i ] );
   } /* end for */

   return 0; /* indicates successful termination */

}/* end main */
```

```
Element         Value
      0             0
      1             0
      2             0
      3             0
      4             0
      5             0
      6             0
      7             0
      8             0
      9             0
```

استخدمنا في الحل عبارتي **for** لإعطاء عناصر المنظومة **n** القيم الابتدائية الصفرية ، ولطباعة المنظومة بالصيغة الجدولية المطلوبة . وعبارة **printf** الأولى تطبع عنواني عمودي جدول المخرجات : أرقام العناصر وقيمها .

* * *

ويمكننا إعطاء القيم الابتدائية لعناصر منظومة عند تعريفها بأن نُتبع التعريف بعلامة التساوي "=" يليها القوسان { } وبينهما قائمة (list) القيم الابتدائية (initializers) المطلوب إعطاؤها بالترتيب وبحيث تفصل (separates) فاصلة (comma) بين أي قيمتين متتاليتين ، كأن نكتب مثلا

```
int  x[ 3 ] = { 5, 12, 4 };
```

مثال ٦ – ٣ : البرنامج التالي يعطي 10 قيم ابتدائية لمنظومة أعداد صحيحة ، ويطبع المنظومة بصيغة جدولية .

```
/* Example 6.3
   Initializing an array with a initializer list */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   /* use initializer list to initialize array n */
   int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
   int i; /* counter */

   printf( "%s%13s\n", "Element", "Value" );

   /* output contents of array in tabular format */
   for ( i = 0; i < 10; i++ ) {
      printf( "%7d%13d\n", i, n[ i ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

```
Element         Value
      0            32
      1            27
      2            64
      3            18
      4            95
      5            14
      6            90
      7            70
      8            60
      9            37
```

ويلاحظ أنه إذا كان عدد القيم الابتدائية المعطاه (initializers) أقل من عدد عناصر المنظومة ، فإن العناصر المتبقية تُعطَى تلقائيا القيمة الابتدائية "صفر" 0 . فمثلاً عناصر المنظومة **n** في مثال ٦ – ٢ يمكن أن نعطيها قيما ابتدائية صفرية كما يلي :

```
int n[ 10 ] = { 0 };
```

فهذه العبارة تقوم صراحة (explicitly) بإعطاء العنصر الأول في المنظومة القيمة الابتدائية 0 ، ثم تعطي العناصر التسعة المتبقية قيما ابتدائية صفرية 0 لأن عدد القيم الابتدائية المعطاه (1) أقل من عدد عناصر المنظومة (10) . ومن الجدير بالذكر أن المنظومات لا تُعطَى تلقائيا قيما ابتدائية صفرية . ولكن على المبرمج أن يُعطِي على الأقل العنصر الأول في المنظومة قيمة ابتدائية صفرية كي تُعطَى العناصر المتبقية تلقائيا قيما ابتدائية صفرية . وهذه الطريقة لإعطاء منظومة قيما ابتدائية صفرية يتم تنفيذها وقت الترجمة (compilation time) بالنسبة للمنظومات الاستاتيكية (**static** arrays) ووقت التشغيل (run time) بالنسبة للمنظومات الأوتوماتيكية .

وإذا كان عدد القيم الابتدائية المعطاة أكبر من عدد عناصر المنظومة كأن نكتب مثلا تعريف المنظومة التالي :

```
int n[ 5 ] = { 32, 27, 64, 18, 95, 14 };
```

فإن ذلك يؤدي إلى خطأ تركيبي (syntax error) ، حيث أننا أعطينا 6 قيما ابتدائية بينما هناك 5 عناصر فقط في المنظومة .

وإذا لم نذكر حجم المنظومة (array size) أي عدد عناصرها في تعريف منظومة مع إعطاء قائمة قيمها الابتدائية (initializer list) ، فإن عدد عناصر المنظومة سيكون هو عدد القيم الابتدائية في هذه القائمة .

مثال ٦ – ٤ : التعريف التالي للمنظومة **n**

```
int n[] = { 1, 2, 3, 4, 5 };
```

سيُنشئ (create) منظومة من خمسة عناصر .

مثال ٦ – ٥ : البرنامج التالي يعرِّف منظومة أعداد صحيحة **s** ، ويحدِّد سعتها (array's size) (10) عن طريق ثابت رمزي **SIZE** (symbolic constant) ، ويعطي عناصر المنظومة القيم الابتدائية 2, 4, 6, …, 20 عن طريق عبارات إسناد حسابية ، ثم يطبع المنظومة بصيغة جدولية .

```
/* Example 6.5
   Initialize the elements of array s to
   the even integers from 2 to 20 */
```

```
#include <stdio.h>
#define SIZE 10

/* function main begins program execution */
int main()
{
   /* symbolic constant SIZE can be used to specify array size */
   int s[ SIZE ]; /* array s has 10 elements */
   int j; /* counter */

   for ( j = 0; j < SIZE; j++ ) { /* set the values */
      s[ j ] = 2 + 2 * j;
   } /* end for */

   printf( "%s%13s\n", "Element", "Value" );

   /* output contents of array s in tabular format */
   for ( j = 0; j < SIZE; j++ ) {
      printf( "%7d%13d\n", j, s[ j ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

```
Element         Value
      0             2
      1             4
      2             6
      3             8
      4            10
      5            12
      6            14
      7            16
      8            18
      9            20
```

نلاحظ أن موجِّه المشغِّل المبدئي (**#define** preprocessor directive)

```
#define SIZE 10
```

يعرِّف ثابتا رمزيا **SIZE** (symbolic constant) قيمته 10. والثابت الرمزي هو اسم تعريفي (identifier) يُستبدل به (replaced with) نص استبدال (replacement text) بواسطة مشغِّل C المبدئي (C preprocessor) قبل ترجمة (compiling) البرنامج . وعندما يُشغَّل البرنامج مبدئيا (program is preprocessed) فإن جميع مواطن ظهور الثابت الرمزي **SIZE** يُستبدَل بها نص الاستبدال 10 (replacement text) .

ومن الجدير بالذكر أن استخدام الثوابت الرمزية لتحديد سعات المنظومات يجعل البرامج أكثر قابلية لتعديل القياس / الوزن (more scalable) . ففي البرنامج السابق (مثال ٦ – ٥) عروة **for** الأولى يمكنها أن تملأ منظومة مكونة من 1000 عنصر بمجرد تغيير قيمة **SIZE** في الموجِّه (directive) **#define** ببساطة من 10 إلى 1000 . أما إذا لم نستخدم الثابت الرمزي **SIZE** فسيكون من الواجب علينا تغيير البرنامج في ثلاثة مواضع مختلفة لتعديل البرنامج لمعالجة منظومة من 1000 عنصر. وكلما كانت البرامج أكبر كلما ظهرت بصورة أوضح فائدة استخدام الثوابت الرمزية في كتابة برامج واضحة.

ومن الأخطاء الشائعة وضع فاصلة منقوطة في نهاية موجِّه المشغِّل المبدئي #include (preprocessor directive) أو #define . وتذكَّر أن موجِّهات المشغِّل المبدئي ليست عبارات في لغة C . وإذا وضعنا فاصلة منقوطة في نهاية موجِّه المشغِّل المبدئي #define في البرنامج السابق ، فإن جميع مواطن ظهور الثابت الرمزي **SIZE** في البرنامج يُستبدَل بها النص 10; (text) بواسطة المشغِّل المبدئي . وهذا قد يؤدي إلى أخطاء تركيبية (syntax errors) أثناء وقت الترجمة (compile time) ، أو أخطاء منطقية (logic errors) وقت التنفيذ (execution time) . ومرة أخرى تذكَّر أن المشغِّل المبدئي ليس من لغة C ، وإنما هو مجرد معالج نص (text manipulator) .

وإسناد قيمة إلى ثابت رمزي في عبارة تنفيذية يُعَدُّ خطأً تركيبيا . فالثابت الرمزي ليس متغيرا ، ولا يُحجز له (reserved) حيِّز معيَّن بواسطة المترجِم (compiler) كما يُحجز للمتغيرات التي تتلقى قيما وقت التنفيذ .

مثال ٦ – ٦ : البرنامج التالي يعرِّف منظومة أعداد صحيحة **a** من 12 عنصر ، ويعطِي عناصرها قيما ابتدائية عند تعريفها . ثم يوجد مجموع القيم التي تحتويها عناصر المنظومة باستخدام عبارة **for** .

```
/* Example 6.6
   Compute the sum of the elements of the array */
#include <stdio.h>
#define SIZE 12

/* function main begins program execution */
int main()
{
   /* use initializer list to initialize array */
   int a[ SIZE ] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
   int i; /* counter */
   int total = 0; /* sum of array */

   /* sum contents of array a */
   for ( i = 0; i < SIZE; i++ ) {
      total += a[ i ];
   } /* end for */

   printf( "Total of array element values is %d\n", total );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Total of array element values is 383
```

مثال ٦ – ٧ :

اشترك أربعون طالبا في استبيان / اقتراع (poll) لتقييم (rating) مدى جودة الطعام الذي تقدِّمه كافيتريا الطلاب ، بحيث يكون تقييم (rating) / تقدير / إجابة الطالب عددا صحيحا على مقياس (scale) من 1 إلى 10 [بحيث أن 1 تعني "ردىء جداً" و 10 تعني "ممتاز"] .
اكتب برنامجا يضع إجابات / تقديرات الطلاب الأربعين في منظومة responses عن طريق إعطاء عناصرها قيما ابتدائية تمثل إجابات الطلاب . ثم يلخص البرنامج نتائج الاقتراع بأن يحصى عدد الطلاب الذين أعطوا كل تقدير من التقديرات العشر ، أي عدد الطلاب الذين أعطوا التقدير 1 ، وعدد الطلاب الذين أعطوا التقدير 2 ، ... ، وعدد الطلاب الذين أعطوا التقدير 10 ، ويضع هذه النتائج في منظومة frequency ، بحيث أن frequency [i] تعني عدد الطلاب الذين أعطوا التقدير i (rating) .

الحل :

```
/* Example 6.7
   Student poll program */
#include <stdio.h>
#define RESPONSE_SIZE 40 /* define array sizes */
#define FREQUENCY_SIZE 11

/* function main begins program execution */
int main()
{
   int answer; /* counter to loop through 40 responses */
   int rating; /* counter to loop through frequencies 1-10 */

   /* initialize frequency counters to 0 */
   int frequency[ FREQUENCY_SIZE ] = { 0 };

   /* place the survey responses in the responses array */
   int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
       1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
       5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

   /* for each answer, select value of an element of array responses
            and use that value as subscript in array frequency to
            determine element to increment */
   for ( answer = 0; answer < RESPONSE_SIZE; answer++ ) {
      ++frequency[ responses [ answer ] ];
   } /* end for */

   /* display results */
   printf( "%s%17s\n", "Rating", "Frequency" );

   /* output the frequencies in a tabular format */
   for ( rating = 1; rating < FREQUENCY_SIZE; rating++ ) {
      printf( "%6d%17d\n", rating, frequency[ rating ] );
   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

Rating	Frequency
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

نلاحـظ أن العنصر frequency [i] فـي المنظومـة **frequency** يعتـبر عـدَّادا (counter) يحصي عدد الطلاب الذين أعطوا التقدير **i** لجودة الطعام ، أي أن لـدينا 10 عـدادات (counters) هـي : frequency [1], frequency [2], …., frequency [10] حيـث المؤشِّر i (subscript) يمثل التقدير i لجـودة الطعام ، ولـذلك عرَّفنا المنظومة **frequency** على أنها منظومة أعداد صحيحة من 11 عنصر متجاهلين frequency [0] (ignoring) لأنه لا يوجد في هذه المسالة التقدير 0 .

وإذا أَعْطَى طالبٌ ما التقدير i لجـودة الطعام [أي أن responses [answer] == i لقيمـةٍ مـا مـن قـيم **answer** حيـث 40 > answer ≥ 0]،فإننـا عندئـذ نزيـد العـدَّاد frequency[i] بواحد ، أي أن

```
frequency [i] = frequency [i] + 1;
```

أو بأسلوب آخر

```
++ frequency [i] ;
```

وحيث أن

```
i == responses [ answer ]
```

فإننا يمكننا كتابة أمر زيادة العداد frequency [i] بواحد هكذا :

```
++ frequency [ responses [ answer ] ];
```

وهذه هـي العبارة الأساسية في عروة **for** الأولى في البرنـامج ، والـتي تزيـد عـدَّاد **frequency** المناسب المقابل لإجابة الطالب . فمثلا عندما answer == 2 ، فإن

```
responses [ answer ] = responses [2] = 6
```

ولذلك فإننا نزيد العدَّاد frequency [6] بواحد ، أي

++ frequency [responses [answer]] == ++ frequency [6]

وهكذا .

ثم تقوم عبارة **for** الثانية بطباعة القيم النهائية للعدادات

frequency [i]; i = 1, 2, …, 10

ملاحظة : إذا احتوت البيانات (أي تقديرات الطلاب i لدرجة جودة الطعام) على قيم خاطئة (invalid) مثل 13 (فهي ليست بين 1 و 10) فإن البرنامج سيحاول إضافة 1 إلى frequency [13]. وهذا سيكون خارج حدود (bounds) المنظومة . وللأسف فإن لغة C ليس من آلياتها / قدراتها اختبار / التحقق من حدود المنظومة (array bounds checking) لتمنع الحاسوب من الإشارة إلى (referring to) عنصر غير موجود . وبالتالي فإن أي برنامج أثناء التنفيذ قد يتجاوز حدود / نهاية المنظومة (walk off the end of an array) بدون إنذار سابق . ولذلك فهي مسئولية المبرمج أن يتأكد / يضمن أن جميع الإشارات إلى عناصر أي منظومة تبقى داخل حدود المنظومة . أما الإشارة إلى عنصر خارج حدود المنظومة فيُعَدُّ من أخطاء البرمجة التي يجب التنبه إليها.

مثال ٦ – ٨ : (الرسم البياني لقيم عناصر منظومة باستخدام المدرج التكراري)

(Graphing Array Element Values with Histograms)

البرنامج التالي يعرِّف منظومة أعداد صحيحة **n** من 10 عناصر ، ويعطيها قيما عند تعريفها. ثم يرسم بيانيا هذه المعلومات في صيغة مدرج تكراري (histogram) / مخطط أعمدة/ أعمدة بيانية (bar chart / diagram / graph) ، حيث يطبع :

رقم العنصر (في المنظومة) : i ،

وقيمة العنصر : n[i] ،

وعمودا / صفا / قضيبا (bar) من نجوم (asterisks) عددها يساوي قيمة العنصر (بجانب رقم العنصر وقيمته) .

الحل :

```
/* Example 6.8
   Histogram printing program */
#include <stdio.h>
#define SIZE 10
```

```
/* function main begins program execution */
int main()
{
  /* use initializer list to initialize array n */
  int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
  int i; /* outer for counter for array elements */
  int j; /* inner for counter counts *s in each histogram bar */

  printf( "%s%13s%17s\n", "Element", "Value", "Histogram" );

  /* for each element of array n, output a bar of the histogram */
  for ( i = 0; i < SIZE; i++ ) {
    printf( "%7d%13d        ", i, n[ i ]) ;

    for ( j = 1; j <= n[ i ]; j++ ) {   /* print one bar */
      printf( "%c", '*' );
          } /* end inner for */

    printf( "\n" ); /* end a histogram bar */
  } /* end outer for */

  return 0; /* indicates successful termination */

} /* end main */
```

Element	Value	Histogram
0	19	**********
1	3	***
2	15	***************
3	7	*******
4	11	***********
5	9	*********
6	13	*************
7	5	*****
8	17	*****************
9	1	*

تقوم عبارة **for** المتداخلة (nested **for** statement) برسم الأعمدة / الصفوف / القضبان (bars) المطلوبة ، حيث تقوم **for** الخارجية بتحديد أرقام الصفوف / العناصر ، بينما تقوم **for**

الداخلية بطباعة النجوم المقابلة لكل عنصر ، أي نجوم الصف المقابل للعنصر . وأما عبارة printf("\n"); فإنها تنهي سطرا من المدرج التكراري .

مثال ٦ – ٩ : (قذف قطعة النرد 6000 مرة وتلخيص النتائج في منظومة)
(Rolling a die 6000 times and summarizing the results in an array)

قُذفت قطعة نرد ذات ستة أوجه (single six-sided die) 6000 مرة . اكتب برنامجا يستخدم دالة توليد الأعداد العشوائية (random number generator) لمحاكاة عملية القذف . ويحسب البرنامج عدد مرات ظهور كل من الأرقام الستة 1, 2, …, 6 ، ويضع هذه الأعداد في منظومة **frequency** ، بحيث أن frequency[i] تمثل عدد مرات ظهور الرقم i .

ملاحظة ١ : سبق حل هذه المسألة في الفصل السابق (في مثال ٥ – ٥) باستخدام عبارة **switch** ، ودون استخدام منظومات .

ملاحظة ٢ : هذه المسألة تختبر ما إذا كانت الدالة المستخدَمة تولِّد فعلاً أعدادا عشوائية ، حيث يجب أن نحصل على النتائج

frequency[i] ≈ 1000 ; i=1, 2, …, 6

الحل:

```
/* Example 6.9
   Roll a six-sided die 6000 times */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 7

/* function main begins program execution */
int main()
{
   int face; /* random die value 1 - 6 */
   int roll; /* roll counter */
   int frequency[ SIZE ] = { 0 }; /* clear counts */

   srand( time( NULL ) ); /* seed random-number generator */

   /* roll die 6000 times */
   for ( roll = 1; roll <= 6000; roll++ ) {
      face = 1 + rand() % 6;
      ++frequency[ face ]; /* replaces 26-line switch of Example 5.5 */
```

```
  } /* end for */

  printf( "%s%17s\n", "Face", "Frequency" );

  /* output frequency elements 1-6 in tabular format */
  for ( face = 1; face < SIZE; face++ ) {
    printf( "%4d%17d\n", face, frequency[ face ] );
  } /* end for */

  return 0; /* indicates successful termination */

} /* end main */
```

Face	Frequency
1	1029
2	951
3	987
4	1033
5	1010
6	990

استخدام منظومات الرموز في تخزين ومعالجة سلاسل الرموز
Using Character Arrays to Store and Manipulate Strings

ناقشنا حتى الآن منظومات الأعداد الصحيحة فقط . وعموما يمكن للمنظومات أن تحتفظ ببيانات من أي نوع . ونناقش الآن تخزين سلاسل الرموز (strings) في منظومات الرموز (character arrays) . العملية الوحيدة التي درسناها للآن لتشغيل (processing) سلسلة رموز هي طباعة سلسلة رموز باستخدام **printf** . وفي الحقيقة فإن أي سلسلة رموز مثل "salam" تُعَدُّ منظومة استاتيكية (**static** array) من رموز مفردة (individual characters) في لغة C.

ومنظومات الرموز لها عدة خصائص مميزة ، منها أن أي منظومة رموز يمكن أن تُعطَى قيما ابتدائية باستخدام سلسلة رموز حرفية (a string literal) . فمثلا الإعلان / التعريف

```
char string1[ ] = "first";
```

يعطِي عناصر المنظومة **string1** قيماً ابتدائية هي الرموز المفردة (individual characters) في سلسلة الرموز الحرفية "first" . وفي هذه الحالة تتحدد سعة المنظومة **string1** بواسطة

البرنامج المترجِم (compiler) بناءً على طول سلسلة الرموز . ومن الجدير بالذكر أن سلسلة الرموز "first" تحتوي على خمسة رموز يضاف إليها رمز خاص لإنهاء السلسلة (a special string termination character) يُدْعَى " الرمز الخالي / الخاوي / الفارغ" (null character) . وهكذا فإن المنظومة **string1** تحتوي فعليا على ستة عناصر . والثابت الرمزي (character constant) الذي يمثِّل الرمز الخالي هو '\0' . وجميع سلاسل الرموز في لغة C تنتهي بهذا الرمز . وأي منظومة رموز تمثِّل سلسلة رموز يجب أن تُعرَّف دائما بحيث تكون كبيرة كبرا كافيا للاحتفاظ بعدد من الرموز يساوي عدد رموز السلسلة والرمز الخالي لإنهائها (terminating null character).

ومنظومات الرموز يمكن أيضا إعطاؤها قيما ابتدائية بواسطة ثوابت رمزية مفردة (individual character constants) في قائمة قيم ابتدائية (initializer list) . فالتعريف السابق مثلا للمنظومة **string1** يكافئ التعريف

```
char string1[ ] = { 'f', 'i', 'r', 's', 't', '\0' };
```

ونظرا لأن سلسلة الرموز هي في الواقع منظومة رموز ، فيمكننا الوصول (accessing) مباشرة (directly) إلى الرموز المفردة (individual characters) في أي سلسلة باستخدام اصطلاح مؤشر المنظومة (array subscript notation) . فمثلا string1[0] هو الرمز 'f' ، وكذلك string1[3] هو الرمز 's' .

كما أنه يمكننا إدخال سلسلة رموز (to input a string) مباشرة في منظومة رموز (a character array) من لوحة المفاتيح باستخدام **scanf** ومحدِّد التحويل %s . فمثلا الإعلان

```
char string2[ 20 ];
```

ينشئ منظومة رموز قادرة على تخزين سلسلة رموز – مكوَّنة من 19 رمز على الأكثر – ورمز الإنهاء الخالي . والعبارة

```
scanf( "%s", string2 );
```

تقرأ سلسلة رموز من لوحة المفاتيح لتخزنها في **string2** . ولاحظ أن اسم المنظومة يُمرَّر إلى **scanf** بدون أن تسبقه العلامة & التي تُستخدم مع المتغيرات غير السلاسل (non-string variables) . والسبب في ذلك هو أن العلامة & عادةً تُستخدم لتنبئ **scanf** بموقع متغيرٍ ما (a variable's location) في الذاكرة (memory) حتى يمكن تخزين قيمة هناك . وسنرى بإذن الله بعد قليل – في هذا الفصل – حين الحديث عن "تمرير المنظومات إلى الدوال" أن اسم أي منظومة هو عنوان بداية المنظومة . ولذلك فلا نحتاج إلى الرمز & .

وتُعَدُّ من مسئوليات المبرمج أن يضمن أن المنظومة التي نقوم بتخزين سلسلة رموز فيها قادرة على استيعاب / الاحتفاظ بـ (holding) أي سلسلة رموز يقوم المستخدم بطباعتها (typing) على لوحة المفاتيح . وذلك لأن الدالة **scanf** تقوم بقراءة الرموز من لوحة المفاتيح إلى أن تصادف أول رمز فراغ أبيض (whitespace character) ، دون أن تختبر / تتحقق من (check) سعة المنظومة ، ولذلك فقد تقوم **scanf** بالكتابة / بالتخزين بعد نهاية المنظومة . وقد يؤدي هذا إلى ضياع / تدمير (destruction) بعض البيانات من البرنامج ، أو إلى أخطاء أخرى وقت التنفيذ (run-time errors) .

ويمكننا طباعة أي منظومة رموز تمثّل سلسلة رموز باستخدام عبارة الطباعة **printf** مع محدد التحويل %s . فنستطيع مثلا طباعة المنظومة **string2** باستخدام العبارة

```
printf( "%s\n", string2 );
```

ولاحظ أن الدالة **printf** مثل الدالة **scanf** لا تتحقق من سعة منظومة الرموز . وتظل تطبع رموز السلسلة حتى تقابل رمز الإنهاء الخالي .

البرنامج التالي يوضح عمليات : إعطاء منظومةِ رموزٍ قيماً ابتدائية بواسطة سلسلة رموز حرفية (a string literal) ، وقراءة سلسلة رموز وتخزينها في منظومة رموز ، وطباعة منظومة رموز كسلسلة رموز ، والوصول إلى (accessing) الرموز المفردة (individual characters) في سلسلة رموز .

مثال ٦ – ١٠ : (معالجة منظومات الرموز كسلاسل رموز)

(Treating character arrays as strings)

اكتب برنامجا يعرِّف منظومتي رموز : الأولى **string1** وسعتها 20 رمزا ، والثانية **string2** يعطيها عند الإعلان عنها قيما ابتدائية بواسطة سلسلة الرموز الحرفية "string literal" . ويطلب البرنامج من المستخدِم (user) إدخال سلسلة رموز فيقوم البرنامج بتخزينها في المنظومة **string1** بواسطة **scanf** . ثم يطبع البرنامج كلا من السلسلتين **string1, string2** ، ثم يعيد طباعة السلسلة **string1** ولكن مع ترك فراغ (space) بين كل رمزين متجاورين في السلسلة .

الحل :

```
/* Example 6.10
   Treating character arrays as strings */
#include <stdio.h>

/* function main begins program execution */
int main()
```

```
{
  char string1[ 20 ]; /* reserves 20 characters */
  char string2[] = "string literal"; /* reserves 15 characters */
  int i; /* counter */

  /* read string from user into array string1 */
  printf("Enter a string: ");
  scanf( "%s", string1 ); /* input ended by whitespace character */

  /* output strings */
  printf( "string1 is: %s\nstring2 is: %s\n"
        "string1 with spaces between characters is:\n",
        string1, string2 );

  /* output characters until null character is reached */
  for ( i = 0; string1[ i ] != '\0'; i++ ) {
    printf( "%c ", string1[ i ] );
  } /* end for */

  printf( "\n" );

  return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a string: Salam brothers
string1 is : Salam
string2 is : string literal
string1 with spaces between characters is :
S a l a m
```

- لاحظ أنه عند تخزين سلسلة رموز باستخدام **scanf** ، تُنْهَى المدخلات برمز الفراغ الأبيض ، ولذلك فإن **strng1** تصبح السلسلة "Salam" فقط .
- البرنامج يستخدم عبارة **for** للمرور بعروة عبر رموز المنظومة **string1** وطباعة هذه الرموز المفردة رمزا رمزا مع ترك فراغ بين كل رمزين باستخدام محدِّد التحويل %c . وأما الشرط الموجود في عبارة **for**

```
string1[ I ] != '\0'
```

فيظل صحيحا / متحققاً طالما أننا لم نصل إلى رمز الإنهاء الفارغ في سلسلة الرموز .

المنظومات المحلية الاستاتيكية والمنظومات المحلية الأوتوماتيكية
Static Local Arrays and Automatic Local Arrays

ناقشنا في الفصل السابق محدِّد طبقة التخزين **static** . ورأينا أن المتغير المحلي الاستاتيكي (**static** local variable) يظل موجودا طوال فترة / أمد (duration) البرنامج ، ولكنه يكون مرئياً (visible) فقط في جسم الدالة (function body) . ويمكننا تطبيق / استخدام المحدِّد **static** مع تعريف أي منظومة محلية ، وبالتالي فإن المنظومة لا تُنشأ (created) وتُعطَى قيما ابتدائية كلما استدعينا الدالة ، وكذلك لا تُمحَى / لا تُدَمَّر المنظومة كلما خرجنا (exited) من الدالة في البرنامج . وهذا يقلل من وقت تنفيذ (execution time) البرنامج ، وخاصة بالنسبة للبرامج التي تُستدعَى فيها مرارا (frequently) دوال تحتوي على منظومات كبيرة .

والمنظومات الاستاتيكية **static** تُعطَى تلقائيا (automatically) قيما ابتدائية مرة واحدة وقت الترجمة (at compile time) . وإذا لم يعطِ المبرمج صراحةً (explicitly) قيماً ابتدائية لمنظومة استاتيكية ، فإن البرنامج المترجِم (compiler) يعطِي عناصر هذه المنظومة قيماً ابتدائية صفرية . والمثال التالي يوضح ذلك .

مثال ٦ – ١١ : البرنامج التالي يعرض دالتين :

- الدالة الأولى : **staticArrayInit** ولها منظومة استاتيكية محلية .
- الدالة الثانية : **automaticArrayInit** ولها منظومة أوتوماتيكية محلية .

– الدالة الأولى تُستدعَى مرتين :

١) في المرة الأولى : يقوم البرنامج المترجِم (compiler) بإعطاء المنظومة الاستاتيكية المحلية في الدالة قيما ابتدائية صفرية

(عند السطر : **static int** array1[3];)

وتقوم الدالة بطباعة المنظومة ، ثم إضافة 5 إلى كل عنصر من عناصر المنظومة ، وطباعة المنظومة مرة أخرى .

٢) وفي المرة الثانية : تكون المنظومة الاستاتيكية محتوية على القيم المخزونة من خلال الاستدعاء الأول للدالة .

- والدالة الثانية تُستدعَى أيضا مرتين :

١) في المرة الأولى : تُعطَى عناصر المنظومة الأوتوماتيكية المحلية في الدالة القيم الابتدائية {1, 2, 3} .

(عند السطر int array2[3] = { 1, 2, 3 };)

وتقوم الدالة بطباعة المنظومة ، ثم إضافة 5 إلى كل عنصر من عناصر المنظومة ، وطباعة المنظومة مرة أخرى .

٢) وفي المرة الثانية : تُعطَى عناصر المنظومة القيم الابتدائية {1, 2, 3} مرة أخرى نظراً لأن المنظومة لها فترة تخزين أوتوماتيكي (automatic storage duration) .

(ملاحظة : من الأخطاء الشائعة افتراض أن عناصر أي منظومة استاتيكية محلية ستُعطَى قيما ابتدائية صفرية كلما استُدعيت الدالة التي عُرِّفت فيها المنظومة)

```
/* Example 6.11
   Static arrays are initialized to zero */
#include <stdio.h>

void staticArrayInit( void );    /* function prototype */
void automaticArrayInit( void ); /* function prototype */

/* function main begins program execution */
int main()
{
   printf( "First call to each function:\n" );
   staticArrayInit();
   automaticArrayInit();

   printf( "\n\nSecond call to each function:\n" );
   staticArrayInit();
   automaticArrayInit();

   return 0; /* indicates successful termination */

} /* end main */

/* function to demonstrate a static local array */
void staticArrayInit( void )
{
   /* initializes elements to 0 first time function is called */
```

```
   static int array1[ 3 ];
   int i; /* counter */

   printf( "\nValues on entering staticArrayInit:\n" );

   /* output contents of array1 */
   for ( i = 0; i <= 2; i++ ) {
      printf( "array1[ %d ] = %d  ", i, array1[ i ] );
   } /* end for */

   printf( "\nValues on exiting staticArrayInit:\n" );

   /* modify and output contents of array1 */
   for ( i = 0; i <= 2; i++ ) {
      printf( "array1[ %d ] = %d  ", i, array1[ i ] += 5 );
   } /* end for */

} /* end function staticArrayInit */

/* function to demonstrate an automatic local array */
void automaticArrayInit( void )
{
   /* initializes elements each time function is called */
   int array2[ 3 ] = { 1, 2, 3 };
   int i; /* counter */

   printf( "\n\nValues on entering automaticArrayInit:\n" );

   /* output contents of array2 */
   for ( i = 0; i <= 2; i++ ) {
      printf("array2[ %d ] = %d  ", i, array2[ i ] );
   } /* end for */

   printf( "\nValues on exiting automaticArrayInit:\n" );

   /* modify and output contents of array2 */
   for ( i = 0; i <= 2; i++ ) {
      printf( "array2[ %d ] = %d  ", i, array2[ i ] += 5 );
   } /* end for */

} /* end function automaticArrayInit */
```

```
First call to each function :

Values on entering staticArrayInit:
array1[ 0 ] = 0  array1[ 1 ] = 0  array1[ 2 ] = 0
Values on exiting staticArrayInit:
array1[ 0 ] = 5  array1[ 1 ] = 5  array1[ 2 ] = 5

Values on entering automaticArrayInit:
array2[ 0 ] = 1  array2[ 1 ] = 2  array2[ 2 ] = 3
Values on exiting automaticArrayInit:
array2[ 0 ] = 6  array2[ 1 ] = 7  array2[ 2 ] = 8

Second call to each function:

Values on entering staticArrayInit:
array1[ 0 ] = 5  array1[ 1 ] = 5  array1[ 2 ] = 5
Values on exiting staticArrayInit:
array1[ 0 ] = 10  array1[ 1 ] = 10  array1[ 2 ] = 10

Values on entering automaticArrayInit:
array2[ 0 ] = 1  array2[ 1 ] = 2  array2[ 2 ] = 3
Values on exiting automaticArrayInit:
array2[ 0 ] = 6  array2[ 1 ] = 7  array2[ 2 ] = 8
```

تمرير المنظومات إلى الدوال Passing Arrays to Functions

لتمرير منظومة كوسيط فعلي (an array argument) إلى دالة نحدِّد اسم المنظومة بدون أي أقواس . فإذا فرضنا مثلا أن المنظومة **hourlyTemperatures** قد تم تعريفها كما يلي :

```
int hourlyTemperatures[ 24 ];
```

فإن الاستدعاء الدالِّي (function call) :

```
modifyArray( hourlyTemperatures, 24 )
```

يُمرِّر المنظومة **hourlyTemperatures** وسعتها إلى الدالة **modifyArray** . وعلى عكس منظومات الرموز التي تحتوي على سلاسل رموز ، فإن الأنواع الأخرى من المنظومات لا تشتمل على رمز خاص للإنهاء (special terminator) . ولهذا السبب فإن سعة المنظومة تمرر إلى الدالة ، حتى تستطيع الدالة تشغيل العدد المضبوط من العناصر .

ولغة C تمرر المنظومات تلقائيا / أوتوماتيكيا إلى الدوال بالإسناد (by reference) : أي أن الدالة المستدعاة تستطيع تغيير / تعديل (modifying) قيم العناصر في المنظومات الأصلية في الدالة المستدعية (caller's original arrays) . واسم المنظومة هو في الحقيقة / فعلياً (actually) عنوان أول عنصر في المنظومة . ونظرا لأننا نمرِّر عنوان المنظومة الابتدائي (starting address of the array) ، فإن الدالة المستدعاة تَعْرِف بالضبط أين موقع تخزين المنظومة . ولذلك فحين تُعدِّل الدالة المستدعاة عناصر المنظومة في جسم الدالة الخاص بها (its function body) ، فإنها فعليا تعدِّل العناصر الأصلية / الفعلية (actual elements) في المنظومة في مواقعها الأصلية بالذاكرة (in their original memory locations) .

مثال ٦ – ١٢ : البرنامج التالي يبين أن اسم أي منظومة هو في الحقيقة عنوان أول عنصر في المنظومة ، وذلك بطباعة كل من

اسم المنظومة : array

عنوان أول عنصر في المنظومة : &array[0]

عنوان المنظومة : &array

وذلك باستخدام محدِّد التحويل p% وهو محدِّد تحويل خاص لطباعة العناوين (addresses) .

```
/* Example 6.12
   The name of an array is the same as &array[ 0 ] */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   char array[ 5 ]; /* define an array of size 5 */

   printf( "    array = %p\n&array[0] = %p\n"
      "   &array = %p\n",
      array, &array[ 0 ], &array );

   return 0; /* indicates successful termination */

} /* end main */
```

```
      array = 0012FF78
&array[0] = 0012FF78
   &array = 0012FF78
```

ملاحظـة : محـدِّد التحويـل p% عـادة يطبـع العنـاوين كأعـداد في النظـام السداسـي عشـر (hexadecimal system) . والأعداد في هذا النظام [الذي أساسه 16 base] تتكـون من الأرقام 0 إلى 9 ، والحروف من A إلى F [وهذه الحروف هي المكافئات السداسـي عشـرية (hexadecimal equivalents) للأعـداد مـن 10 إلى 15] . والأعداد في هذا النظام غالبا ما تستخدم كاصطلاح مختصر (shorthand notation) لقيم الأعداد الصحيحة الكبيرة .

ومـن مخرجـات البرنـامج (مثـال ٦ – ١٢) نـرى أن قيمـة **array** هـي نفسـها قيمـة **&array[0]** وهي 0012FF78 . ومخرجات هذا البرنامج لا تعتمد على النظام المستخدم (system dependent) ، ولكن العناوين ستكون دائما متطابقة (identical) لتنفيذ خاص (a particular execution) لهـذا البرنـامج باسـتخدام حاسـوب خـاص (a particular computer) .

ذكرنا سابقا أن المنظومات تُمرر عادة بالإسناد . ولو كانت تمرر بالقيمة (by value) لكان لزاماً تمرير نسخة من كل عنصر . وواضح أنه بالنسبة للمنظومات الكبيرة التي تُمرر بكثرة (frequently passed large arrays) فـإن ذلـك سيسـتغرق وقتـا طـويلا (time consuming) ويشغل حيزا كبيرا في التخزين (consume considerable storage) لِنُسَخِ المنظومات (copies of the arrays) .

ورغم أن المنظومات بأكملها (entire arrays) تُمرَّر بالإسناد ، إلا أن العناصر المفردة في المنظومات تمرر بالقيمة كالمتغيرات البسيطة تماما . وهذه القطع / الوحدات المفردة البسيطة من البيانات (simple single pieces of data) [كالوحدات المفردة من الأنواع **int, float, char**] يطلق عليها "أعداد قياسية" (scalar) . ولتمرير عنصر من منظومة إلى دالة نستخدم اسم عنصر المنظومة (الذي يحتوي على مؤشر) (subscripted name of the array element) (مثلا a [3]) كوسيط فعلي (argument) في استدعاء الدالة .(*)

(*) سنرى بإذن الله في كتابنا التالي "البرمجة المتقدمة بلغة C " دار اقرأ للنشر ، الكويت ، وبعد دراسة ما يُعرف بالمؤشرات (pointers) كيفية تمرير الوحدات القياسية (المتغيرات المفردة وعناصر المنظومات) إلى الدوال بالإسناد ، وكيفية تمرير منظومة بالقيمة .

وكي تستقبل أي دالة منظومةً – عبر استدعاء دالة – يجب أن تحدِّد قائمة وسطاء الدالة (function's parameter list) أن منظومةً سوف تُستقبل . فمثلا مقدمة / عنوان الدالة (function header) بالنسبة للدالة **modifyArray** التي استدعيناها سابقا – عند بداية الحديث عن تمرير المنظومات إلى الدوال – يمكن أن تُكتب كما يلي :

```
void modifyArray( int b[ ], int size )
```

حيث تشير إلى أن الدالة **modifyArray** تتوقع أن تستقبل منظومة أعداد صحيحة في الوسيط **b** (parameter) ، وعدد عناصر المنظومة في الوسيط **size** . وليس مطلوبا (ليس ضروريا) أن نكتب سعة المنظومة بين قوسي المنظومة [ ] . وإذا كتبنا هذه السعة فإن البرنامج المترجِم يتحقق من أنها أكبر من الصفر ثم يتجاهلها . وتحديد سعة سالبة يعد خطأً وقت الترجمة (compile error) . ونظراً لأن المنظومات تُمرَّر تلقائيا بالإسناد ، فعندما تَستخدم الدالة المستدعاة اسم المنظومة **b** فسيُعلم أنها تشير إلى المنظومة في الدالة المستدعية [أي إلى المنظومة **hourlyTemperatures** في الاستدعاء السابق] .

المثال التالي يوضح الفارق بين تمرير منظومة بأكملها ، وتمرير عنصر من منظومة .

مثال ٦ – ١٣ :

أ) اكتب دالة خاوية **modifyArray** تستقبل منظومة أعداد صحيحة **b** سعتها **size** ، وتضاعف قيم عناصرها .

ب) اكتب دالة خاوية **modifyElement** تستقبل عددا صحيحا **e** وتضاعف قيمته وتطبع القيمة المضاعفة .

ج) اكتب دالة رئيسية

(i) تعرِّف منظومة أعداد صحيحة **a** من خمسة عناصر وتعطيها القيم الابتدائية {0, 1, 2, 3, 4 } ، وتطبع قيم عناصر **a** . ثم تمرِّر **a** وسعتها إلى الدالة **modifyArray** ، ثم تعيد طباعة قيم عناصر **a** . هل هناك تغيَّر في قيم هذه العناصر المطبوعة ؟

(ii) تطبع قيمة العنصر a[3] ، ثم تمرِّره إلى الدالة **modifyElement** ، ثم تعيد طباعة قيمة العنصر a[3] . هل هناك تغير في هذه القيمة المطبوعة ؟

الحل :

```
/* Example 6.13
   Passing arrays and individual array elements to functions */
#include <stdio.h>
#define SIZE 5

/* function prototypes */
```

```
void modifyArray( int b[], int size );
void modifyElement( int e );

/* function main begins program execution */
int main()
{
  int a[ SIZE ] = { 0, 1, 2, 3, 4 }; /* initialize a */
  int i; /* counter */

  printf( "Effects of passing entire array by reference:\n\nThe "
       "values of the original array are:\n" );

  /* output original array */
  for ( i = 0; i < SIZE; i++ ) {
    printf( "%3d", a[ i ] );
  } /* end for */

  printf( "\n" );

  /* pass array a to modifyArray by reference */
  modifyArray( a, SIZE );

  printf( "The values of the modified array are:\n" );

  /* output modified array */
  for ( i = 0; i < SIZE; i++ ) {
    printf( "%3d", a[ i ] );
  } /* end for */

  /* output value of a[ 3 ] */
  printf( "\n\n\nEffects of passing array element "
       "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );

  modifyElement( a[ 3 ] ); /* pass array element a[ 3 ] by value */

  /* output value of a[ 3 ] */
  printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );

  return 0; /* indicates successful termination */

} /* end main */
```

```
/* in function modifyArray, "b" points to the original array "a"
   in memory */
void modifyArray( int b[], int size )
{
   int j; /* counter */

   /* multiply each array element by 2 */
   for ( j = 0; j < size; j++ ) {
      b[ j ] *= 2;
   } /* end for */

} /* end function modifyArray */

/* in function modifyElement, "e" is a local copy of array element
   a[ 3 ] passed from main */
void modifyElement( int e )
{
   /* multiply parameter by 2 */
   printf( "Value in modifyElement is %d\n", e *= 2 );
} /* end function modifyElement */
```

```
Effects of passing entire array by reference:

The values of the original array are:
   0  1  2  3  4
The values of the modified array are:
   0  2  4  6  8


Effects of passing array element by value:

The value of  a[3]  is  6
Value in modifyElement  is 12
The value of  a[ 3 ]  is  6
```

نلاحظ من المخرجات أن هناك تغيُّرا في قيم عناصر المنظومة **a** بعد تمريرها إلي الدالة **modifyArray** (التي تقوم بمضاعفة قيم العناصر) ثم طباعة قيم العناصر بعد استدعاء الدالة ، حيث تغيرت القيم من : {0, 1, 2, 3, 4} قبل الاستدعاء إلى : {0, 2, 4, 6, 8} بعد الاستدعاء. بينما نلاحظ عدم تغير قيمة العنصر a[3] بعد تمريره إلى الدالة **modifyElement** (التي تقوم بمضاعفة قيمة العنصر وطباعة القيمة الجديدة) ثم طباعة قيمة العنصر a[3] في الدالة

الرئيسية **main**) بعد استدعاء الدالة ، حيث كانت قيمة العنصر a[3] قبل الاستدعاء : 6 ، وأيضا بعد الاستدعاء :6. والسبب هو أن عنصر المنظومة a[3] قد تم تمريره بالقيمة وليس بالإسناد .

* * *

قد تكون هناك في بعض البرامج حالات يجب ألا نسمح فيها لدالةٍ ما بعمل أي تغيير / تعديل في قيم عناصر منظومةٍ ما . ونظرا لأن المنظومات تُمرَّر دائما بالإسناد ، فمن الصعب التحكم في التعديلات التي تُجرى على القيم في المنظومة . ولغة C تسمح لنا باستخدام مؤهِّل نوع (type qualifier) يُدْعَى **const** يمنع حدوث أي تعديلات لقيم المنظومة في دالةٍ ما . وعندما نكتب **const** قبل أي وسيط منظومة (array parameter) ، فإن عناصر المنظومة تصبح ثابتة في جسم الدالة (function body) ، وأي محاولة لتعديل أي عنصر من المنظومة في جسم الدالة سينتج عنها عندئذ خطأ وقت الترجمة (compile time error) ، وبالتالي يمكننا تصحيح البرنامج بحيث لا يحاول تعديل قيم عناصر المنظومة .

مثال ٦ – ١٤ : البرنامج التالي يوضح استخدام المؤهل **const** ، ونتيجة محاولة تعديل قيم عناصر منظومة كتبنا **const** قبل اسم وسيطها . الدالة **tryToModifyArray** معرَّفة بوسيط (parameter)

const int b[ ]

أي أنه يحدِّد أن المنظومة **b** ثابتة ولا يمكن تعديلها . ونلاحظ في المخرجات ظهور رسائل الخطأ الثلاث التي يصدرها البرنامج المترجِم (compiler) [قد تختلف رسائل الخطأ هذه عن تلك التي تظهر في نظامكم] ، حيث تحاول الدالة ثلاث مرات تعديل قيم عناصر المنظومة ، وكل مرة تظهر الرسالة "1-value specifies const object"

```
/* Example 6.14
   Demonstrating the const type qualifier with arrays */
#include <stdio.h>

void tryToModifyArray( const int b[] ); /* function prototype */

/* function main begins program execution */
int main()
{
   int a[] = { 10, 20, 30 }; /* initialize a */

   tryToModifyArray( a );

   printf("%d %d %d\n", a[ 0 ], a[ 1 ], a[ 2 ] );
```

```
   return 0; /* indicates successful termination */

} /* end main */

/* in function tryToModifyArray, array b is const, so it cannot be
   used to modify the original array a in main. */
void tryToModifyArray( const int b[] )
{
   b[ 0 ] /= 2;    /* error */
   b[ 1 ] /= 2;    /* error */
   b[ 2 ] /= 2;    /* error */
} /* end function tryToModifyArray */
```

```
Compiling…
error C2166:  1-value specifies const object
error C2166:  1-value specifies const object
error C2166:  1-value specifies const object
```

ترتيب عناصر المنظومات Sorting Arrays

يُعَدُّ ترتيب البيانات (sorting data) [أي وَضْع البيانات بترتيب خاص كترتيب تصاعدي (ascending order) أو ترتيب تنازلي (descending order)] في منظومةٍ ما من أهم التطبيقات (applications) باستخدام الحاسوب . فقد يقوم أحد البنوك مثلاً بترتيب جميع الشيكات بناءً على رقم الحساب (account number) حتى يمكنه إعداد التقارير المصرفية الفردية (individual bank statements) نهاية كل شهر . وتقوم بعض شركات التليفونات بترتيب قوائم المشتركين بناءً على الاسم الأخير (last name) ، ثم داخل هذا تقوم بترتيب آخر بناءً على الاسم الأول (first name) حتى يكون من السهل الوصول إلى أي أرقام تليفونات مطلوبة . وافتراضيا فإن أي منظمة يجب أن تقوم بترتيب بعض البيانات ، وفي حالات كثيرة تكون كميات هذه البيانات كبيرة جدا . وهناك خوارزميات عديدة في علم الحاسوب لعملية الترتيب . وفيما يلي سندرس بإذن الله واحدة من أبسط هذه الخوارزميات . وفي التمرينات بنهاية الفصل نناقش بإذن الله خوارزميات أكثر تعقيدا ولكن أفضل أداءً وكفاءة .

مثال ٦ – ١٥ : خوارزمية الترتيب الفقاعي (bubble sort algorithm)

تقوم هذه الخوارزمية بترتيب عناصر منظومة ترتيبا تصاعديا (ascending order) عن طريق المرور (passing) على جميع عناصر المنظومة عدة مرات / أشواط / دورات / مراحل (passes / phases) وفي كل مرة / مرحلة من هذا المرور (on each pass) نقارن (compare) بين كل عنصرين متتاليين (successive pairs of elements) . فإن كانا مرتبين تصاعدياً أو كانا متطابقين تركناهما كما هما دون أي تعديل . أما إن كانا مرتبين تنازليا / تناقصيا (in decreasing order) فإننا نبدلهما (swap them) مكان بعضيهما البعض ليصبحا مرتبين تصاعديا . ونستمر في تطبيق هذه الطريقة إلى أن نصل إلى مرحلة لا يحدث فيها أي تبديل بين أي عنصرين ، فتكون العناصر كلها عندئذ مرتبة ترتيبا تصاعديا ، أو إلى أن يصل عدد المراحل إلى SIZE-1 حيث SIZE هو سعة المنظومة .

تتبَّع خطوات تنفيذ هذه المراحل لترتيب عناصر المنظومة التالية ترتيبا تصاعديا .

{ 8, 6, 3, 5, 11, 2, 7, 4 }

الحل :

نقارن العنصر الأول 8 مع العنصر الثاني 6 ونضع أصغرهما 6 في الموضع الأول . ثم نقارن العنصر الثاني – بعد ترتيب العنصرين الأوليين – (أي نقارن العنصر 8) مع العنصر الثالث 3 ، ونضع أصغرهما 3 في الموضع الثاني . ثم نقارن 8 مع 5 ، ونضع 5 في الموضع الثالث . ثم نقارن 8 مع 11 ، ونتركهما في موضعيهما الرابع والخامس دون أي تبديل , ثم نقارن 11 مع 2 ، ونضع 2 في الموضع الخامس . وهكذا .. إلى أن تنتهي المرحلة الأولى . ثم نكرر الطريقة في المراحل التالية . وفيما يلي نتائج المراحل المختلفة إلى أن نصل إلى الترتيب التصاعدي المطلوب .

المنظومة المعطاة	المرحلة الأولى	المرحلة الثانية	المرحلة الثالثة	المرحلة الرابعة	المرحلة الخامسة	المرحلة السادسة
8	6	3	3	3	2	2
6	3	5	5	2	3	3
3	5	6	2	5	**4**	**4**
5	8	2	6	**4**	5	5
11	2	7	**4**	6	6	6
2	7	**4**	7	7	7	7
7	**4**	8	8	8	8	8
4	11	11	11	11	11	11

ملاحظة : تُسَمَّى هذه الطريقة " الترتيب الفقاعي" (bubble sort) أو "الترتيب الغَطْسِي / الغَوْصي / الرسوبي" (sinking sort) لأن العناصر الصغرى الموجودة أسفل وضعها

الصحيح (النهائي) تتجه إلى التحرك لأعلى نحو قمة المنظومة (top of the array) كالفقاعات (bubbles) التي تتحرك في الماء لأعلى متجهة نحو السطح [انظر مثلا إلى تحركات العنصر 4 ، أو إلى تحركات أصغر عنصر 2] ، وكذلك العناصر الكبرى تتجه إلى التحرك لأسفل / الغَطْس / الغَوْص / الرسوب نحو قاع المنظومة (bottom of the array) .

مثال ٦ – ١٦ : اكتب برنامجا يعطِي قيما ابتدائية لمنظومة أعداد صحيحة مكونة من 10 عناصر ، ثم يستخدم طريقة الترتيب الفقاعي لترتيب عناصر المنظومة ترتيبا تصاعديا .

الحل :

```
/* Example 6.16
   This program sorts an array's values into ascending order */
#include <stdio.h>
#define SIZE 10

/* function main begins program execution */
int main()
{
   /* initialize a */
   int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
   int pass; /* passes counter */
   int i;    /* comparisons counter */
   int hold; /* temporary location used to swap array elements */

   printf( "Data items in original order\n" );

   /* output original array */
   for ( i = 0; i < SIZE; i++ ) {
      printf( "%4d", a[ i ] );
   } /* end for */

   /* bubble sort */
   /* loop to control number of passes */
   for ( pass = 1; pass < SIZE; pass++ ) {

      /* loop to control number of comparisons per pass */
      for ( i = 0; i < SIZE - 1; i++ ) {

         /* compare adjacent elements and swap them if first
```

```
      element is greater than second element */
      if ( a[ i ] > a[ i + 1 ] ) {
         hold = a[ i ];
         a[ i ] = a[ i + 1 ];
         a[ i + 1 ] = hold;
      } /* end if */

   } /* end inner for */

} /* end outer for */

printf( "\nData items in ascending order\n" );

/* output sorted array */
for ( i = 0; i < SIZE; i++ ) {
   printf( "%4d", a[ i ] );
} /* end for */

printf( "\n" );

return 0; /* indicates successful termination */
```

```
Data items in original order
   2   6   4   8   10   12   89   68   45   37
Data items in ascending order
   2   4   6   8   10   12   37   45   68   89
```

يقارن البرنامج أولا بين a[0], a[1] ، ثم بين a[1], a[2] ، ثم بين a[2], a[3] ، ...، وهكذا حتى تنتهي المرحلة الأولى بالمقارنة بين a[8], a[9] . ونظرا لأن لدينا 10 عناصر ، فهناك 9 مقارنات . وبناءً على الطريقة التي ذكرناها لتبديل العناصر عند المقارنات المتتابعة ، فإن أي قيمة كبرى قد تتحرك نحو أسفل المنظومة عدة مواضع في المرحلة الواحدة ، بينما قد تتحرك أي قيمة صغرى لأعلى موضعا واحدا فقط . وبعد انتهاء المرحلة الأولى سنضمن هبوط أكبر قيمة بالمنظومة إلى قاعها ، أي إلى الموقع a[9] . وبعد انتهاء المرحلة الثانية سنضمن هبوط ثاني أكبر قيمة إلى الموقع a[8] . وبعد انتهاء المرحلة التاسعة سنضمن هبوط تاسع أكبر قيمة إلى الموقع a[1] . وبذلك تَتبقى لنا أصغر قيمة في الموقع a[0] . أي أننا احتجنا إلى تسع مراحل لترتيب المنظومة المكونة من 10 عناصر ترتيبا تصاعديا .

ونلاحظ أن عملية الترتيب قد تمت باستخدام عروة **for** متداخلة . وإذا احتجنا إلى عملية تبديل (swapping) قيمتين ، فإننا نجريها بثلاث عبارات إسناد هي :

```
hold = a[ i ];
a[ i ] = a[ i + 1 ];
a[ i + 1 ] = hold;
```

حيث يقوم المتغير الإضافي **hold** بتخزين إحدى القيمتين – المطلوب تبديلهما – مؤقتا . ومن الأخطاء التي يقع فيها بعض المبتدئين كتابة العبارتين التاليتين فقط لتبديل القيمتين :

```
a[ i ] = a[ i + 1 ];
a[ i + 1 ] = a[ i ];
```

ولبيان الخطأ نفرض أن قيمة a[i] تساوي 7 ، وقيمة a[i+1] تساوي 5 . بعد تنفيذ عبارة الإسناد الأولى تصبح قيمة كل من a[i], a[i+1] تساوي 5 ، والقيمة 7 تُفقد تماما . وبعد تنفيذ عبارة الإسناد الثانية تظل قيمة كل من a[i], a[i+1] تساوي 5 . أي أننا لم نبدل قيمتي العنصرين . ولذلك فإننا نحتاج إلى المتغير الإضافي **hold** .

والميزة الأساسية لخوارزمية الترتيب الفقاعي هي سهولة برمجتها . إلا أنها بطيئة في التنفيذ ، ويتضح هذا العيب حين نقوم بترتيب منظومات كبيرة السعة . وفي التمرينات بنهاية الفصل سنعرض بإذن الله صورا أخرى من خوارزمية الترتيب الفقاعي أكثر كفاءة . وفي التمرينات أيضا سنعرض بإذن الله طرقا أخرى أكثر كفاءة من الترتيب الفقاعي .

مثال ٦ – ١٧ : (تحليل بيانات / نتائج استبيان / مسح / فحص عام) (Survey Data Analysis)

أَعْطَى 99 شخصا آراءهم حول مشروعٍ ما عن طريق إعطاء تقييم / تقدير (عدد صحيح) من 1 إلى 9 . اكتب برنامجا يستخدم منظومة **response** يعطيها 99 قيمة ابتدائية تمثل تقديرات الأشخاص ، ثم يحلل هذه البيانات عن طريق إيجاد كل من :

المتوسط **(mean)** : وهو القيمة المتوسطة (average) وتساوي مجموع جميع القيم / التقديرات مقسوما على عددها (99) .

الأوسط **(median)** : وهو العنصر الأوسط (middle element) (أي العنصر رقم 49) بعد ترتيب جميع العناصر / التقديرات – وعددها 99 – ترتيبا تصاعديا .

المنوال **(mode)** : وهو القيمة / التقدير الأكثر تكرارا (most frequent) .

الحل :

```
/* Example 6.17
   This program introduces the topic of survey data analysis.
   It computes the mean, median and mode of the data */
#include <stdio.h>
#define SIZE 99

/* function prototypes */
void mean( const int answer[] );
void median( int answer[] );
void mode( int freq[], const int answer[] ) ;
void bubbleSort( int a[] );
void printArray( const int a[] );

/* function main begins program execution */
int main()
{
   int frequency[ 10 ] = { 0 }; /* initialize array frequency */

   /* initialize array response */
   int response[ SIZE ] =
      { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
        7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
        6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
        7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
        6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
        7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
        5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
        7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
        7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
        4, 5, 6, 1, 6, 5, 7, 8, 7 };

   /* process responses */
   mean( response );
   median( response );
   mode( frequency, response );

   return 0; /* indicates successful termination */

} /* end main */

/* calculate average of all response values */
```

```
void mean( const int answer[] )
{
  int j; /* counter for totaling array elements */
  int total = 0; /* variable to hold sum of array elements */

  printf( "%s\n%s\n%s\n", "********", "  Mean", "********" );

  /* total response values */
  for ( j = 0; j < SIZE; j++ ) {
    total += answer[ j ];
  } /* end for */

  printf( "The mean is the average value of the data\n"
          "items. The mean is equal to the total of\n"
          "all the data items divided by the number\n"
          "of data items ( %d ). The mean value for\n"
          "this run is: %d / %d = %.4f\n\n",
          SIZE, total, SIZE, (  double  ) total / SIZE );
} /* end function mean */

/* sort array and determine median element's value */
void median( int answer[] )
{
  printf( "\n%s\n%s\n%s\n%s",
          "********", " Median", "********",
          "The unsorted array of responses is" );

  printArray( answer ); /* output unsorted array */

  bubbleSort( answer ); /* sort array */

  printf( "\n\nThe sorted array is" );
  printArray( answer ); /* output sorted array */

  /* display median element */
  printf( "\n\nThe median is element %d of\n"
          "the sorted %d element array.\n"
          "For this run the median is %d\n\n",
          SIZE / 2, SIZE, answer[ SIZE / 2 ] );
} /* end function median */

/* determine most frequent response */
```

```
void mode( int freq[], const int answer[] )
{
  int rating; /* counter for accessing elements 1-9 of array freq */
  int j; /* counter for summarizing elements 0-98 of array answer */
  int h; /* counter for diplaying histograms of elements in array freq */
  int largest = 0; /* represents largest frequency */
  int modeValue = 0; /* respesents most frequent response */

  printf( "\n%s\n%s\n%s\n",
        "********", "  Mode", "********" );

  /* initialize frequencies to 0 */
  for ( rating = 1; rating <= 9; rating++ ) {
    freq[ rating ] = 0;
  } /* end for */

  /* summarize frequencies */
  for ( j = 0; j < SIZE; j++ ) {
    ++freq[ answer[ j ] ];
  } /* end for */

  /* output headers for result columns */
  printf( "%s%11s%19s\n\n%54s\n%54s\n\n",
        "Response", "Frequency", "Histogram",
        "1    1    2    2", "5    0    5    0    5" );

  /* output results */
  for ( rating = 1; rating <= 9; rating++ ) {
    printf( "%8d%11d          ", rating, freq[ rating ] );

    /* keep track of mode value and largest frequency value */
    if ( freq[ rating ] > largest ) {
      largest = freq[ rating ];
      modeValue = rating;
    } /* end if */

    /* output histogram bar representing frequency value */
    for ( h = 1; h <= freq[ rating ]; h++ ) {
      printf( "*" );
    } /* end inner for */

    printf( "\n" ); /* being new line of output */
```

```
   } /* end outer for */

   /* display the mode value */
   printf( "The mode is the most frequent value.\n"
           "For this run the mode is %d which occurred"
           " %d times.\n", modeValue, largest );
} /* end function mode */

/* function that sorts an array with bubble sort algorithm */
void bubbleSort( int a[] )
{
   int pass; /* pass counter */
   int j;    /* comparison counter */
   int hold; /* temporary location used to swap elements */

   /* loop to control number of passes */
   for ( pass = 1; pass < SIZE; pass++ ) {

      /* loop to control number of comparisons per pass */
      for ( j = 0; j < SIZE - 1; j++ ) {

         /* swap elements if out of order */
         if ( a[ j ] > a[ j + 1 ] ) {
            hold = a[ j ];
            a[ j ] = a[ j + 1 ];
            a[ j + 1 ] = hold;
         } /* end if */

      } /* end inner for */

   } /* end outer for */

} /* end function bubbleSort */

/* output array contents (20 values per row) */
void printArray( const int a[] )
{
   int j; /* counter */

   /* output array contents */
   for ( j = 0; j < SIZE; j++ ) {
```

```
      if ( j % 20 == 0 ) { /* begin new line every 20 values */
         printf( "\n" );
      } /* end if */

      printf( "%2d", a[ j ] );
   } /* end for */

} /* end function printArray */
```

```
********
   Mean
********
The mean is the average value of the data
items. The mean is equal to the total of
all the data items divided by the number
of data items ( 99 ). The mean value for
this run is: 681 / 99 = 6.8788


********
  Median
********
The unsorted array of responses is
6 7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 8
6 7 8 9 3 9 8 7 8 7 7 8 9 8 9 8 9 7 8 9
6 7 8 7 8 7 9 8 9 2 7 8 9 8 9 8 9 7 5 3
5 6 7 2 5 3 9 4 6 4 7 8 9 6 8 7 8 9 7 8
7 4 4 2 5 3 8 7 5 6 4 5 6 1 6 5 7 8 7

The sorted array is
1 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5 5
5 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

The median is element 49 of
The sorted 99 element array.
For this run the median is 7


********
   Mode
```

```
********
Response  Frequency       Histogram
                                1   1   2   2
                            5   0   5   0   5

       1          1       *
       2          3       ***
       3          4       ****
       4          5       *****
       5          8       ********
       6          9       *********
       7         23       ***********************
       8         27       ***************************
       9         19       *******************
The mode is the most frequent value.
For this run the mode is 8 which occurred 27 times.
```

تقوم الدالة **mean** بحساب القيمة المتوسطة (average value) بجمع الـ 99 عنصر في المنظومة وقسمة الناتج على 99 . وتقوم الدالة **median** بإيجاد القيمة الوسطى (middle value) عن طريق استدعاء الدالة **bubbleSort** لترتيب عناصر منظومة التقديرات (responses) [المنظومة **answer**] ترتيبا تصاعديا ، ومن ثم انتقاء (picking) العنصر الأوسط answer [SIZE/2] في المنظومة المرتبة . ولاحظ أنه إذا كان عدد العناصر زوجيا فإن العنصر الأوسط (median) يُحسَب على أنه القيمة المتوسطة (mean) للعنصرين الأوسطين (2 middle elements) . والدالة **median** المكتوبة في البرنامج السابق لا تعالج هذه الحالة . وقد استدعينا الدالة **printArray** لطباعة المنظومة **response** .

والمنوال **(mode)** هو القيمة الأكثر تكرارا / حدوثا بين القيم / التقديرات / الإجابات (responses) التي عددها 99 . وتقوم الدالة **mode** بتحديد المنوال عن طريق إحصاء / عد (counting) عدد مرات تكرار (frequency) كل تقدير من التقديرات / الإجابات (responses)، ثم اختيار القيمة الأكثر تكرارا . والدالة **mode** في هذا البرنامج لا تعالج حالة تساوي قيمتين في أعلى تكرار (وسنعالج هذه الحالة بإذن الله في التمرينات بنهاية الفصل) . وتقوم الدالة **mode** برسم مدرج تكراري (histogram) ليساعد في تحديد المنوال بيانيا .

البحث في المنظومات Searching Arrays

عادة يتعامل المبرمجون مع كميات كبيرة من البيانات المخزونة في منظومات . وقد يكون من الضروري تحديد ما إذا كانت منظومةٌ ما تحتوي على قيمة تتطابق مع قيمة معينة **key** . وعملية إيجاد عنصر معين في منظومة يطلق عليها عملية "بحث" (searching) .

طريقة البحث الخطي البسيط
(Simple Linear Search Technique)

في هذه الطريقة نقارن كل عنصر من عناصر المنظومة مع قيمة البحث (مفتاح البحث) (search key) . ونظرا لأن عناصر المنظومة غير مرتبة بأي ترتيب معين فإن احتمال وجود قيمة البحث في العنصر الأول كاحتمال وجودها في العنصر الأخير . ولذلك ففي المتوسط (on average) سيقارن البرنامج قيمة البحث مع نصف عناصر المنظومة .

مثال ٦ – ١٨ :

أ) اكتب دالة صحيحة القيمة **int linearSearch** تبحث عن قيمة صحيحة **int key** في منظومة صحيحة **int array** عدد عناصرها **size** ، وتعيد موضع (location) العنصر المقابل في المنظومة إن وجدت القيمة ، بينما تعيد 1- إن لم تجدها .

ب) اكتب دالة رئيسية

(i) تنشئ منظومة صحيحة **a** عدد عناصرها / حجمها size = 100 ، باستخدام العلاقة

$$a_x = 2 \times ; \qquad o \le \times < size$$

(ii) تقرأ قيمة صحيحة **searchKey** .

(iii) تبحث عن هذه القيمة **searchKey** في المنظومة **a** باستخدام الدالة **linearSearch** .

الحل :

```
/* Example 6.18
   Linear search of an array */
#include <stdio.h>
#define SIZE 100
```

```
/* function prototype */
int linearSearch( const int array[], int key, int size );

/* function main begins program execution */
int main()
{
   int a[ SIZE ]; /* create array a */
   int x; /* counter for initializing elements 0-99 of array a */
   int searchKey; /* value to locate in array a */
   int element; /* variable to hold location of searchKey or -1 */

   /* create data */
   for ( x = 0; x < SIZE; x++ ) {
      a[ x ] = 2 * x;
   } /* end for */

   printf( "Enter integer search key:\n" );
   scanf( "%d", &searchKey );

   /* attempt to locate searchKey in array a */
   element = linearSearch( a, searchKey, SIZE );

   /* display results */
   if ( element != -1 ) {
      printf( "Found value in element %d\n", element );
   } /* end if */
   else {
      printf( "Value not found\n" );
   } /* end else */

   return 0; /* indicates successful termination */

} /* end main */

/* compare key to every element of array until the location is found
   or until the end of array is reached; return subscript of element
   if key or -1 if key is not found */
int linearSearch( const int array[], int key, int size )
{
   int n; /* counter */

   /* loop through array */
```

```
   for ( n = 0; n < size; ++n ) {

      if ( array[ n ] == key ) {
         return n; /* return location of key */
      } /* end if */

   } /* end for */

   return -1; /* key not found */

} /* end function linearSearch */
```

```
Enter integer search key:
36
Found value in element 18
```

```
Enter integer search key:
37
Value not found
```

تمرينات رقم ٦

٦-١) اكتشف الخطأ في كل من قطع البرامج التالية ، وصحح الخطأ :

أ)
```
#define SIZE 100;
```

ب)
```
SIZE = 10;
```

ج)
```
int b[ 10 ] = { 0 }, i;
for ( i = 0; i <= 10; i++)
   b[ i ] = 1;
```

د)
```
#include <stdio.h>;
```

هـ)
```
#define VALUE = 120
```

٦ – ٢) اكتب عبارات تقوم بتنفيذ ما يلي :

أ) عرض / طباعة قيمة العنصر السابع في منظومة الرموز f .

ب) إدخال قيمة في العنصر رقم 4 في منظومة الأعداد ذوات النقطة العائمة (floating-point array) b .

ج) اعط القيمة الابتدائية 8 لكل عنصر من العناصر الخمسة في منظومة الأعداد الصحيحة g .

د) أوجد مجموع قيم الـ 100 عنصر في منظومة الأعداد ذوات النقطة العائمة c .

هـ) انسخ المنظومة a في الجزء الأول من المنظومة b . افرض :

```
double  a[11],  b[34]
```

و) أوجد واطبع أصغر قيمة وأكبر قيمة في منظومة أعداد ذوات نقطة عائمة w مكونة من 99 عنصرا .

٦ – ٣) اكتب عبارة واحدة لتنفيذ كل من العمليات التالية على المنظومات :

أ) اعط قيما ابتدائية صفرية لعناصر منظومة الأعداد الصحيحة counts التي سعتها 10

ب) أضف 1 لكل عنصر من عناصر منظومة الأعداد الصحيحة bonus وعددهم 15 عنصرا .

ج) اقرأ من لوحة المفاتيح 12 قيمة لعناصر منظومة أعداد ذوات نقطة عائمة monthlyTemperatures .

د) اطبع الخَمس قيم من منظومة الأعداد الصحيحة bestScores في صيغة عمود (in column format) .

٦ – ٤) اكتشف الخطأ / الأخطاء في كل من العبارات التالية ، وصحح الخطأ إن أمكن :

أ) افرض أن لدينا الإعلان :

```
char str[5];
scanf( "%s", str );  /* User types salam */
```

ب) افرض أن لدينا الإعلان :

```
int a[ 3 ];
printf( "$d  %d  %d\n", a[1], a[2], a[3] );
```

ج)

```
double f[3] = { 1.1, 10.01, 100.001, 1000.0001 };
```

٦ – ٥) اختر الإجابة الصحيحة في كل مما يلي :

(i) نفرض أن المتغير **int a** قيمته 3 ، وأن منظومة الأعداد الصحيحة **b** لها سبعة عناصر . ما هي الطريقة الصحيحة لإسناد ناتج جَمْع 3 والعنصر الثالث إلى العنصر الخامس في المنظومة ؟

أ) `b[a+1] = b[a] +3;`
ب) `b[a+1] = b[a-1] +3;`
ج) `b[a] +1 = b[a+3];`
د) `b[a+2] = b[a] +3;`

(ii) أي الطرق التالية غير صحيحة لإعطاء عناصر منظومة قيما ابتدائية ؟

أ) `int n[5] = {0,7,0,3,8,2};`
ب) `int n[] = {0,7,0,3,8,2};`
ج) `int n[5] = {7};`
د) `int n[5] = {6,6,6};`

(iii) نفرض أن **string1** منظومة رموز . أي العمليات التالية لا تولِّد / لا تُنتج (does not produce) سلسلة رموز (a string) ؟

أ) `string1[] = "test";`
ب) `string1[] = { 't', 'e', 's', 't', '\0' };`
ج) `string1[] = { 't', 'e', 's', 't' };`
د) `string1[] = " ";`

(iv) إذا كان عدد القيم الابتدائية (initializers) أقل من عدد العناصر في المنظومة ، فإن العناصر المتبقية :

أ) تُحذَف (deleted)
ب) يتم تجاهلها (ignored)
ج) تُعطَى قيما ابتدائية خالية (initialized to empty)
د) تُعطَى قيما ابتدائية صفرية (initialized to zero)

(v) أي العبارات التالية صحيحة (true) بخصوص العبارة

```
++frequency[ responses[ answer ] ];
```

أ) العبارة تزيد العداد **frequency** المناسب بناءً على قيمة responses[answer] .
ب) العبارة تزيد العداد **answer** المناسب بناءً على قيمة frequency [responses] .
ج) العبارة تزيد العداد **responses** المناسب بناءً على قيمة frequency [answer] .
د) العبارة ينتج عنها خطأ تركيبي (syntax error) نظراً لأن المؤشرات (subscripts) لا يمكن تَداخُلها (cannot be nested) .

(vi) أي التعريفات التالية تكافئ التعريف

```
char string1[] = "first";
```

أ) `character string1[] = { 'f, 'i', 'r', 's', 't', '\0' };`
ب) `char string1 = { 'f, 'i', 'r', 's', 't', '\0' };`
ج) `char string1[] = { 'f, 'i', 'r', 's', 't' };`
د) `char string1[] = { 'f, 'i', 'r', 's', 't', '\0' };`

(vii) عندما نستخدم طريقة الترتيب الفقاعي (bubble sort) لترتيب عناصر منظومة مكونة من 1000 عنصر ، فإننا نحتاج على الأكثر عددا من الأشواط / الدورات / المراحل / مرات المرور (passes) يساوي

أ) 1001 ب) 1000
ج) 999 د) 998

٦-٦) تدفع إحدى الشركات رواتب بائعيها (salespeople) على أساس العمولة (on a commission basis) ، حيث يتقاضى البائع في الأسبوع أجرا يساوي 200$ مضافا إليها 9% من إجمالي مبيعاته (gross sales) خلال هذا الأسبوع . فمثلا إذا بلغ إجمالي مبيعات أحد الباعة في أسبوع مبلغ 3000$ ، فإنه يتقاضى في هذا الأسبوع راتبا يساوي :

$$200+\frac{9}{100}\times 3000=200+270=470$$

أي يساوي 470 $.

اكتب برنامجا يقرأ إجمالي مبيعات عدد من الباعة خلال أحد الأسابيع . ويتوقف إدخال البيانات حين يُدخل المستخدم القيمة 1- .

ويحسب البرنامج لكل بائع راتبه خلال هذا الأسبوع ، ويوجد عدد الباعة الذين تقع رواتبهم في كل فئة / مدى (range) من الفئات التالية :

أ)		$200-299
ب)		$300-399
ج)		$400-499
د)		$500-599
هـ)		$600-699
و)		$700-799
ز)		$800-899
ح)		$900-999
ط)	أكبر من أو يساوي	$1000

افرض أن راتب أي بائع يُقطَع (truncated) إلى عدد صحيح . واستخدم منظومة **salaries** للاحتفاظ بأعداد الباعة في الفئات المختلفة [أي أنها منظومة عدَّادات (counters)] ، فمثلا : العنصر salaries [5] يشير إلى عدد الباعة الذين يتقاضون راتبا في المدى : 500-599$ ، والعنصر salaries[9] يشير إلى الباعة الذين يتقاضون راتبا في المدى 900-999$.

٦ – ٧) تعد خوارزمية الترتيب الفقاعي التي وضَّحها برنامج مثال ٦ – ١٦ منخفضة الكفاءة بالنسبة للمنظومات الكبيرة . سنقوم في الخطوتين التاليتين بعمل تعديلات بسيطة لتحسين كفاءة هذه الخوارزمية .

أ) بعد المرحلة الأولى من المرور (after first pass) نكون قد ضَمِنَّا أن أكبر عدد موجود في عنصر المنظومة ذي أكبر رقم . وبعد المرحلة الثانية نكون قد ضَمِنَّا أن أكبر عددين في موضعيهما (in place) الصحيحين ، ... وهكذا . فبدلا من عمل 9 مقارنات في كل مرحلة من مراحل المرور (on every pass) [بالنسبة لمنظومة مثال ٦ – ١٦ المكونة من 10 عناصر] ، عَدِّل الخوارزمية بحيث نعمل 8 مقارنات في المرحلة الثانية ، و 7 مقارنات في المرحلة الثالثة ، وهكذا .

ب) بيانات المنظومة المعطاة قد تكون مُرَتَّبة أصلا ترتيبا تصاعديا ، أو قد تكون قريبة من الترتيب التصاعدي المطلوب . فلماذا نُنفِّذ 9 مراحل إذا كان يكفينا عدد أقل من المراحل ؟ عَدِّل الخوارزمية بحيث نتحقق في نهاية كل مرحلة ما إذا كانت قد عُملت أي عملية تبديل (swapping) في هذه المرحلة . فإذا لم تُعمَل أي عملية تبديل ، فمعنى ذلك أن البيانات / الأعداد مرتَّبة فعلا بالترتيب التصاعدي المطلوب ، ويجب عندئذ إنهاء البرنامج . وإذا عُمِلت أي تبديلات فإننا نحتاج على الأقل لمرحلة أخرى من المرور .

٦–٨– أ) عَدِّل برنامج مثال ٦ – ١٧) بحيث تعالج الدالة **mode** حالة تساوي قيمتين أو أكثر (to handle a tie) في الحدوث أعلى تكرارا ، أي حالة وجود منوالين أو أكثر (modes) .

ب) عَدِّل كذلك الدالة **median** بحيث تعالج حالة احتواء المنظومة على عدد زوجي من العناصر ، وبالتالي يُحسَب العنصر الأوسط (median) على أنه القيمة المتوسطة للعنصرين الأوسطين .

٦ – ٩) اكتب برنامجا يستخدم منظومة لحل السؤال التالي : اقرأ 20 عددا صحيحا ، بحيث يقع أي عدد في المدى من 10 إلى 100 احتوائيا (inclusive) . وكلما قُرِأ عدد فإنه يُطبع فقط إن لم يكن تكرارا (duplicate) لعدد سبقت فعلا قراءته . اكتب البرنامج بحيث :

i) يراعى "أسوأ حالة" (worst case) وهي عندما تكون جميع الأعداد المقروءة مختلفة.

ii) يستخدم أصغر منظومة ممكنة لحل السؤال .

٦ – ١٠) اكتب برنامجا يحاكي عملية قذف قطعتي نرد (rolling two dice) ، بحيث يستخدم البرنامج الدالة **rand** لقذف القطعة الأولى ، ثم يستخدم **rand** مرة أخرى لقذف القطعة الثانية . ويحسب البرنامج مجموع القيمتين الناتجتين من قذف قطعتي النرد . وحيث أن القيمة الصحيحة الناتجة من قذف أي من القطعتين تقع في المدى من 1 إلى 6 ، فمجموع القيمتين سيتراوح من 2 إلى 12 ، بحيث يكون المجموع 7 هو أكثر المجاميع تكرارا (most frequent sum) ، والمجموع 2 أو 12 أقلها تكرارا (least frequent sum) . والشكل / الجدول التالي يعرض التوافقات (combinations) المختلفة لقطعتي النرد – وعددها 36 توافقا – والمجموع المقابل لكل توافق . [لاحظ أن المجموع 7 يحدث 6 مرات ، بينما كل من المجموع 2 والمجموع 12 يظهر مرة واحدة فقط] .

اكتب البرنامج بحيث يقذف قطعتي النرد 36,000 مرة . واستخدم منظومة لتسجيل (to tally) عدد مرات ظهور كل مجموع من المجاميع المحتملة : (each possible sum:) 2, 3, 4, …, 12 . واطبع النتائج في صيغة جدولية تبين : المجموع المحتمل ، وعدد مرات ظهوره ، والنسبة المئوية لهذا العدد من العدد الإجمالي (36,000) ، والنسبة المئوية المتوقعة لهذا العدد . [مثلا النسبة المئوية المتوقعة للعدد / للمجموع 7 هي 16.667% (= 1/6 = 6/36) ، والنسبة المئوية المتوقعة للمجموع 12 هي 2.778% (= 1/36)] .

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

٦ – ١١) (نظام الحجز على خطوط الطيران) (Airline Reservation System)

اشترت إحدى شركات الطيران الصغيرة حاسوبا لنظام الحجز الآلي / الأوتوماتي الجديد على خطوطها . وقد طلب منك المسئول أن تبرمج النظام الجديد ، بحيث تكتب برنامجا يعطِي / يحدد (أرقام) المقاعد (to assign seats) على كل رحلة طيران للطائرة الوحيدة للشركة ، والتي سعتها 10 مقاعد . ويعرض البرنامج قائمة الاختيارات التالية :

Please type 1 for "first class"
Please type 2 for "economy"

والتي تعني طباعة الرقم 1 لاختيار الدرجة الأولى ، أو طباعة الرقم 2 لاختيار الدرجة الاقتصادية / السياحية . فإذا طبع الشخص الرقم 1 ، فإن البرنامج يحدد له مقعدا في قسم الدرجة الأولى (المقاعد من 1 إلى 5) . وإذا طبع الرقم 2 ، فإن البرنامج يحدد له مقعدا في قسم الدرجة السياحية (المقاعد من 6 إلى 10) . ثم يطبع البرنامج بطاقة صعود الطائرة (boarding pass) ، والتي تبيِّن رقم مقعد الشخص ، وما إذا كان في قسم الدرجة الأولى أو في قسم الدرجة السياحية .

استخدم منظومة لتمثِّل خارطة مقاعد الطائرة . وأسند قيما ابتدائية صفرية لجميع عناصر المنظومة للدلالة على أن جميع المقاعد خالية / غير مشغولة (empty) . وكلما حُدِّد / اختير /

أُسنِد مقعد (a seat is assigned) لشخصٍ ما فإن العنصر المقابل في المنظومة تُغَيَّر قيمته إلى 1 للدلالة على أن المقعد لم يعد شاغرا / متوفرا (available) .

ويجب ألا يقوم البرنامج – بالطبع – بإسناد مقعد سبق إسناده . وعندما تُشغَل / تمتلئ جميع مقاعد الدرجة الأولى ، فإن البرنامج يسأل الشخص – الذي يرغب في حجز مقعد في الدرجة الأولى – إن كان مقبولا لديه أن يُعطَى مقعدا في الدرجة السياحية ، [والعكس أيضا صحيح : عندما تمتلئ مقاعد الدرجة السياحية] . فإن أجاب بنعم فإن البرنامج يُسنِد له المقعد المناسب ، وإن أجاب بلا فإن البرنامج يطبع الرسالة "Next flight leaves in 3 hours" .

٦-١٢) (إيجاد الأعداد الأولية بطريقة "منخل إراستوشينز")
(Finding Prime Numbers using Sieve of Erastosthenes)

العدد الصحيح الأولي (a prime integer) هو أي عدد صحيح يقبل القسمة (بالتساوي) (can be divided evenly) على نفسه والواحد 1 فقط ، مثل العدد 7 .

اكتب برنامجا يستخدم منظومة من 1000 عنصر لإيجاد وطباعة الأعداد الأولية بين 1 و 999، مستخدما طريقة "منخل إراستوشينز" الموضحة فيما يلي ، ومتجاهلا العنصر 0 في المنظومة.

فكرة الطريقة :

من الواضح أن أي عدد صحيح لا يكون أوليا إذا كان من مضاعفات (multiples) عدد آخر (أصغر منه) . ولذلك فإننا نبدأ بمجموعة جميع الأعداد الصحيحة بين 1 و 999 ، ثم نحذف منها – على التوالي – الأعداد التي هي مضاعفات أعداد أخرى ليبقى لنا في النهاية (بعد عملية النخل هذه) مجموعة الأعداد الأولية المطلوبة .

تطبيق الفكرة :

أ) أنشئ منظومة **array** مكوَّنة من 1000 عنصر ، وأعط جميع عناصرها القيمة الابتدائية 1 (true) . عناصر المنظومة التي مؤشراتها أعداد أولية (with prime subscripts) ستبقى 1 إلى نهاية تطبيق الفكرة ، وبالتالي ستكون مؤشراتها هي الأعداد الأولية المطلوبة . أما باقي عناصر المنظومة فستتغير قيمها في النهاية إلى أصفار (false) .

ب) مبتدئين بمؤشر المنظومة 2 (array subscript) [المؤشِّر 1 يجب أن يكون أوليا] كلما وجدنا أن عنصرا من عناصر المنظومة قيمته 1 ، فإننا نمر في عروة (loop) عبر بقية عناصر المنظومة ونغيِّر إلى الصفر 0 (set to zero) قيمة أي عنصر مؤشِّرُه هو أحد مضاعفات a) (multiple of مؤشِّر ذلك العنصر الذي قيمته 1 . فمثلا بالنسبة لمؤشر المنظومة 2 فإن جميع عناصر المنظومة بعد العنصر (رقم) 2 التي أرقامها / مؤشراتها هي مضاعفات 2 [أي : 4, 6, 8, 10, …] تُغيَّر قيمها إلى أصفار . وبالنسبة لمؤشِّر المنظومة 3 فإن جميع العناصر

بعد العنصر 3 التي مؤشراتها هي مضاعفات 3 [أي : ... ,15 ,12 ,9 ,6] تُغيَّر قيمها إلى أصفار ، وهكذا .

ج) عندما تنتهي هذه العملية فإن عناصر المنظومة التي بقيت قيمها آحادا 1 تعني أن مؤشراتها هي الأعداد الأولية المطلوبة ، وبالتالي فإننا نطبع هذه المؤشرات .

٦-١٣) نفرض أن لدينا الإعلانات التالية :

```
const int MAX_STUD = 100;// Max.Number of students

int failing [MAX_STUD];
int passing [MAX_STUD];
int grade;
int score [MAX_STUD];
```

أ) اكتب دالة تعطي قيماً ابتدائية -جميعها 0 (false)- لجميع عناصر المنظومة المنطقية **failing** ، حيث المنظومة **failing** تعد وسيطاً (parameter) .

ب) اكتب دالة تأخذ المنظومة **failing** ، والمنظومة **score** كوسيطين . وتقوم الدالة بإعطاء عناصر **failing** القيمة 1 (true) كلما كانت القيمة المقابلة في **score** أقل من 60 . افرض أن العدد الفعلي للطلاب هو **length** حيث (length ≤ MAX_STUD) .

ج) اكتب دالة تأخذ المنظومة **passing** ، والمنظومة **score** كوسيطين. وتقوم الدالة بإعطاء عناصر **passing** القيمة 1 (true) حينما تكون القيمة المقابلة في **score** أكبر من أو تساوي من 60 . افرض أن العدد الفعلي للطلاب هو MAX_STUD .

د) اكتب دالة **PassTally** تأخذ المنظومة **passing** كوسيط ثم تخبرنا (reports) كم عدد العناصر التي قيمتها **1 (true)** في المنظومة **passing** .

هـ) اكتب دالة **Error** تأخذ الوسيطين **passing, failing** وتعيد القيمة 1 (true) إذا تطابق أي عنصرين متقابلين (corresponding elements) في المنظومتين **passing, failing** .

و) اكتب دالة تأخذ الوسيطين **grade, score** ، وتعيد عدد قيم **score** التي هي أكبر من أو تساوي **grade** .

ز) 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]

٦ – ١٤) 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 :

$$D_i = T_i - T_{av} , \quad i = 1 , 2 , \dots , 40$$

٦ – ١٥) اكتب برنامجا لقراءة قيم عناصر منظومة مكونة من مائة عنصر وحساب الجذر التربيعي لمجموع مربعات العناصر الفردية الأرقام أي لحساب :

$$S = \sqrt{A_1^2 + A_3^2 + A_5^2 + A_7^2 + \cdots + A_{99}^2}$$

٦ – ١٦) 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 رقم أول عنصر سالب في هذه المنظومة ، وإذا لم توجد أي قيمة سالبة فإن البرنامج يطبع القيمة 1- (i = -1) .

٦ – ١٧) تتكون المنظومة **Y** من خمسين عنصرا. اكتب برنامجا لقراءة قيمة عدد صحيح N وقيم أول **N** عنصر من المنظومة **Y**:

$$Y_1 , Y_2 , Y_3 , \dots , Y_N$$

ثم حساب (أ) القيمة المتوسطة (Average)

$$AVG = \sum_{i=1}^{N} Y_i / N$$

(ب) الانحراف القياسي (Standard Deviation)

$$SD = \sqrt{\frac{\sum_{i=1}^{N} Y_i^2}{N} - AVG^2}$$

٦ – ١٨) 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** .

٦ – ١٩) اكتب برنامجا يقرأ قيم خمسة وأربعين عددا ثم يوجد ترتيب وقيمة العدد ذي أكبر قيمة مطلقة .

٦ – ٢٠) 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 .

٦ – ٢١) نفرض أن كلا من **M** , **L** 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.

٦ – ٢٢) نفرض أن **Y** المعرفة بالعلاقة

$$Y = F(x) = a_0 + a_1x + a_2x^2 + \ldots + a_ix^i + \ldots + a_nx^n$$

هي كثيرة حدود من درجة **n** في المتغير **x** .

اكتب برنامجا (أ) يقرأ قيم المعاملات

$a_0 , a_1 , a_2 , \ldots , a_n$

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 .

٦ – ٢٣) نفرض أن خمسمائة شخص قدموا تبرعات في سبيل الله تعالى ، وأن قيم هذه التبرعات بالدينار قد وضعت كعناصر منظومة **D** :

$D_1 , D_2 , \ldots , D_{500}$

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 .

٦ – ٢٤) 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

$$P(X) = A_1+A_2X+A_3X^2+\ldots+A_{i+1}X^i+\ldots+A_NX^{N-1}$$
$$= \sum_{i=0}^{N-1} A_{i+1}X^i$$

$$P'(X) = A_2+2A_3X+3A_4X^2+...+iA_{i+1}X^{i-1}+...+(N-1)A_NX^{N-2}$$
$$= \sum_{i=1}^{N-1} iA_{i+1}X^{i-1}$$

٦ – ٢٥) 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) .

٦ – ٢٦) المطلوب كتابة برنامج يحسب مجموع المتسلسلة $S = \sum_{i=1}^{100} T_i$ حيث حدود هذه المتسلسلة تُعَرَّف كما يلي :

$T_1 = 0$
$T_2 = 1$
$T_i = T_{i-1} + T_{i-2}$; $i = 3 , 4 , 5 , \ldots$

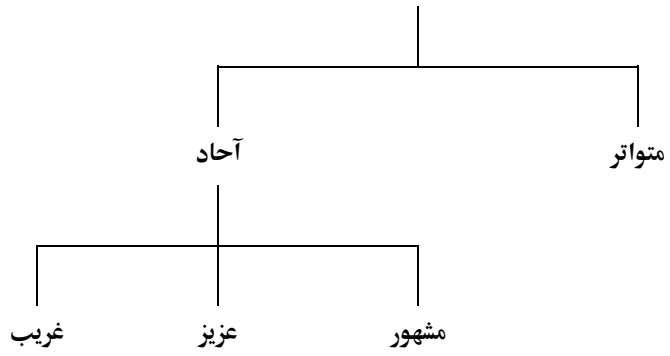
(ملاحظة : هذه المتسلسلة تعرف باسم متسلسلة فيبوناتشي Fibonacci) .

٦ – ٢٧) 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

٦ – ٢٨) 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 $M \leq 6$) .

– الرقم التعريفي للحديث : **ID**

– عدد رواة الحديث في كل طبقة من طبقات السند :

$N_1, N_2 , \ldots , N_i , \ldots , N_M$

(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** .

٦ – ٢٩) 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

٦ – ٣٠ –أ) 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 .

ب) اكتب برنامجا لقراءة درجات طلاب الفصلين **A , B** في الجزء أ) من السؤال ، وحساب متوسطي درجات الفصلين **AvgA , AvgB** ، وطباعة رسالة تفيد أي المتوسطين أعلى .

٦ – ٣١) اكتب برنامجا يستخدم ثلاث منظومات لتخزين معلومات عن عدد **n** من الطلاب (حيث $n \leq 100$) :

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 :

0711	90	C
0712	80	M

0713 75 B
0714 60 M
0715 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 .

٦ – ٣٢) 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 كما هو مبين فيما يلي :

اكتب برنامجا يقوم بقراءة عدد الطلاب **n** (حيث $n \leq 50$) ، والأوزان الاختبارية (test weights) $weight_1$, $weight_2$, $weight_3$ 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)

٦ – ٣٣) عُمل 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

٦ – ٣٤) أُعدت 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 :

سلسلة الإجابات	رقم الطالب
FTTFTFTTFT	0080
FTFTFTTTFF	0340
FTTFTTTTTT	0341
TTFFTFFTTT	0401
TTFTTTFFTF	0462
TTTTTTTTTT	0463
FTFFTFFTFT	0464
TFTFTFTFTF	0512
TTTFFTTFTF	0618
FFFFFFFFFF	0619
TFTTFTTFTF	0687
FTFFTTFFFT	0700
FTFTFTFTFT	0712
TFTFTTFTFT	0837

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**

٦ – ٣٥) 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 .

المطلوب :

(أ) كتابة برنامج 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 رقمها 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)		

٦ – ٣٦ أ) اكتب دالة

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 الدالة **Break** .

٦ – ٣٧) (١) المطلوب كتابة دالة **ProdArrays** 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 **ProdArrays** ليخزن حاصل ضرب عناصر **A, B** المتقابلة في منظومة جديدة **C** ، وكذلك يطبع البرنامج الرئيسي عناصر المنظومات الثلاث **A , B , C** .

٦ – ٣٨) (أ) اكتب دالة 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) ;
```

٦ – ٣٩) اكتب دالة 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);

٦ – ٤٠) اكتب دالة **bigarr** لها 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 الدالة تسند لعناصر المنظومة **c** قيم العناصر المقابلة في المنظومة **a** ، وما عدا ذلك فإنها تسند لعناصر **c** قيم العناصر المقابلة في **b** .

٦ – ٤١) اكتب دالة **bottom** لها 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); .

٦ – ٤٢) اكتب دالة 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 .

٦ – ٤٣) اكتب دالة 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 .

٦ – ٤٤) 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 تستقبل منظومة **y** مكونة من مائة قيمة **float** موجبة ، وتستبدل أصفارا بالقيم التي تقل أي منها عن ٣٪ من أكبر قيمة بالمنظومة.

٦ – ٤٥) أ) اكتب دالة **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 .

$$\left\{ \text{مثلا : المنظومة} \begin{bmatrix} 4.0 \\ 8.0 \\ 6.0 \\ 2.0 \end{bmatrix} \text{تصبح بعد التعديل} \begin{bmatrix} 0.5 \\ 1.0 \\ 0.75 \\ 0.25 \end{bmatrix} \text{حيث } n = 4 \right\}$$

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 .

٦ – ٤٦) اكتب دالة 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 : **X , n**

مخرجات الدالة : **Y**

والدالة تقوم بنسخ (copying) عناصر المنظومة **X** في المنظومة **Y** بترتيب معاكس (وذلك باستخدام عروة) ، أي أن :

$$Y[1] = X[n] , Y[2] = X[n-1] , \dots , Y[n-1] = X[2] , Y[n] = X[1]$$

٦ – ٤٧ أ) اكتب دالة **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** .

٦ – ٤٨) 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

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**) .

٦ – ٤٩) أ) اكتب دالة **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) .

٦ – ٥٠) أ) اكتب دالة منطقية **ascend** تستقبل وسيطين **n , a** حيث **a** منظومة أعداد **float** عدد عناصرها **n** ، وتعيد الدالة القيمة 1 (true) إذا كانت المنظومة **a** مرتبة ترتيبا تصاعديا ، وإلا فإنها تعيد القيمة 0 (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 1 (true) عندما تكون المنظومة **a** مرتبة ترتيبا تنازليا .

٦ – ٥١) اكتب دالة منطقية **boundd** تستقبل ٣ وسطاء : منظومة **a** من النوع **float** بها خمسون عنصرا ، وعددين **rmin , rmax** من النوع **float** ، وتعيد القيمة 1 (true) إذا وقعت جميع قيم عناصر المنظومة **a** في المدى ما بين الحد الأدنى (lower bound) **rmin** والحد الأقصى **rmax** (upper bound) ، أما إذا وقعت أي قيمة خارج هذا المدى فإن الدالة تعيد القيمة 0 (false) .

٦ – ٥٢) 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 **maxs** تحسب عدد مرات ظهور أعلى قيمة في منظومة أعداد **float** ، حيث تستقبل الدالة وسيطين : المنظومة **a** وعدد عناصرها **n** .

٦ – ٥٣) اكتب دالة 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 **n , a** .

٦ – ٥٤) اكتب دالة **close** تستقبل وسيطين : **n , a** ، 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 .

٦ – ٥٥) اكتب دالة 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 .

٦ – ٥٦) اكتب دالة تقوم بتبديل عنصرين معينين من عناصر منظومة. افرض أن متغيرات الدالة الوسيطة هي :

A : اسم المنظومة

i , j : رقما العنصرين المطلوب تبديلهما .

افرض كذلك أن المنظومة لا تحتوي على أكثر من ٢٤ عنصرا ..

٦ – ٥٧) أ) نفرض أن كلا من **A , B** منظومة مكونة من عدد من العناصر يساوي **N** . المطلوب كتابة دالة **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 :

$$\alpha = \left(\sum_{i=1}^{40} x_i z_i \right)^2$$

$$\beta = \frac{\sqrt{x_1^2 + x_2^2 + \cdots + x_{40}^2} \cdot \sqrt{z_1^2 + z_2^2 + \cdots + z_{40}^2}}{\sqrt{x_1 z_1 + x_2 z_2 + \cdots + x_{40} z_{40}}}$$

$$\gamma = x_1 (y_1 + z_1) + x_2 (y_2 + z_2) + \cdots + x_{20} (y_{20} + z_{20})$$

٦ – ٥٨) أ) اكتب دالة **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 .

٦ – ٥٩) أ) اكتب دالة تحسب مجموع القيم المطلقة لعدد **n** من الأعداد

$$X_1 , X_2 , \dots , X_n$$

أي تحسب $\sum_{i=1}^{n} |X_i|$

ب) اكتب برنامجا :

١- يقرأ قيمة عدد صحيح $\mathbf{n_1}$ (حيث $n_1 \le 30$) وقيم عناصر منظومة **A** مكونة من $\mathbf{n_1}$ عنصرا .

٢- يقرأ قيمة عدد صحيح $\mathbf{n_2}$ (حيث $n_2 \le 40$) وقيم عناصر منظومة **B** مكونة من $\mathbf{n_2}$ 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 .

٦ – ٦٠) أ) اكتب دالة

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** ، وسنرمز لها بالرموز

$$C_1 , C_2 , \ldots , C_{10}$$

وكذلك قيمة متغير **Y** .

٢- يستخدم الدالة **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 .

٦ – ٦١) اكتب دالة

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 **A** .

ثم اكتب برنامجا يقرأ قيمة متغير **M** ، وقيم أول **M** عنصر في مطلع منظومة **X** ، أي قيم العناصر

$$X_1 , X_2 , X_3 , \ldots , X_M$$

ثم يستدعي الدالة **MAXMIN** لإيجاد أكبر عنصر **XHIGH** وأصغر عنصر **XLOW** في المجموعة $X_1 , X_2 , X_3 , \ldots , X_M$. افرض أن $M \leq 20$.

٦ – ٦٢) أ) اكتب دالة **AVG (X , N)** تحسب القيمة المتوسطة لعدد **N** من الأرقام

$$X_1 , X_2 , X_3 , \ldots , X_N$$

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 .

٦ – ٦٣) أ) اكتب دالة **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) .

٦ – ٦٤) أ) اكتب دالة

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 .

٦ – ٦٥) أ) اكتب دالة

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** رقم 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 رقمه في المنظومة **A** .

٦ – ٦٦) أ) افرض أن **SET1** منظومة تحتوي على **N1** عنصر عبارة عن أعداد صحيحة موجبة مختلفة .. وافرض كذلك أن **SET2** منظومة تحتوي على **N2** من الأعداد الصحيحة الموجبة المتباينة . اكتب دالة

INTER (SET1, N1 , SET2, N2 , SET3, N3)

تقوم بتخزين تقاطع المنظومتين المذكورتين في المنظومة **SET3** ، أي أن **SET3** ستحتوي على **N3** عنصر هي الأعداد الموجودة في كل من **SET1, SET2** .

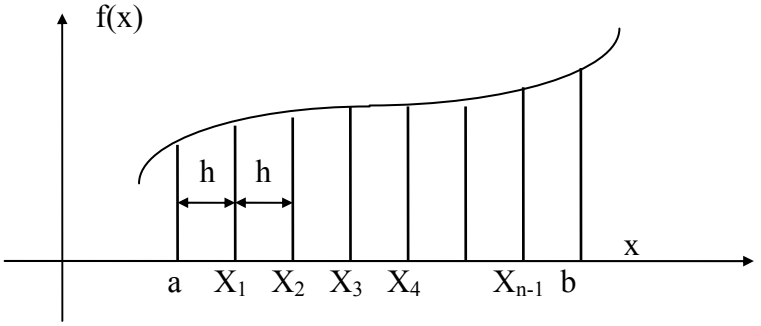
ب) نفرض أن كل اسم من أسماء الصحابة رضوان الله عليهم جميعا قد أعطى رقما موجبا مميزا (أي مختلفا عن غيره من الأرقام) ، وأن المنظومة **BADR** تحتوي على ٣١٤ رقما هي أرقام الصحابة الذين اشتركوا في غزوة بدر ، وأن المنظومة **OHOD** تحتوي على ٧٠٠ رقم هي أرقام الصحابة الذين اشتركوا في غزوة أحد .

اكتب برنامجا يقرأ أرقام كل من المجموعتين **BADR** و **OHOD** ثم يستخدم الدالة السابقة **INTER** لإيجاد المنظومة **BO** التي تحتوي على أرقام الصحابة الذين اشتركوا في كل من الغزوتين .

٦ – ٦٧) نفرض أن الدالة **f** موجبة في الفترة $a \le x \le b$. المساحة الواقعة تحت المنحنى $y = f(x)$ والمحصورة بين الخطين $x = a$ و $x = b$ 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 :

$$h = \tfrac{b-a}{n}, x_i = a+ih;\ i = 1,2,\cdots,n-1$$

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 $x = a$ و $x = b$ وذلك باستخدام القانون السابق .

إرشاد : يمكن استخدام دالة تعيد قيمة $f(x)$.

٦٨-٦) أ) اكتب دالة **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 **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** .

الفصل السابع
صياغة المدخلات والمخرجات
Formatting Input and Output

نناقش في هذا الفصل بإذن الله تعالى بعض تفاصيل صياغة المدخلات والمخرجات باستخدام الدالتين **printf, scanf** حيث تقوم الدالة **scanf** بإدخال البيانات من سيل / فيض المدخلات القياسي (standard input stream)، بينما تقوم الدالة **printf** بإخراج البيانات إلى سيل / فيض المخرجات القياسي (standard output stream)* . وقد درسنا في الفصول السابقة بعض خصائص (features) الدالتين **printf, scanf** . وفي هذا الفصل نلخص هذه الخصائص ، ونضيف إليها خصائص أخرى .

السيول Streams

تتم جميع عمليات الإدخال والإخراج عن طريق ما يُعرَف بالسيول (streams) وهي متتابعات من البايتات (sequences of bytes). ففي عمليات الإدخال تسيل / تفيض (flow) البايتات من نبيطة / جهاز (a device) [مثل لوحة مفاتيح (a keyboard) ، أو محرِّك أقراص (a disk drive) ، أو وصلة شبكة (a network connection)] إلى الذاكرة الرئيسية (main memory) . بينما في عمليات الإخراج تتجه البايتات من الذاكرة الرئيسية إلى جهاز [مثل شاشة عرض (a display screen) ، أو طابعة (a printer) ، أو محرِّك أقراص ، أو وصلة شبكة ، ... إلخ]

وعندما يبدأ تنفيذ البرنامج يتم تلقائيا (automatically) توصيل ثلاثة سيول بالبرنامج : فعادة (normally) يتم توصيل سيل المدخلات القياسي بلوحة المفاتيح ، وسيل المخرجات القياسي بالشاشة ، وغالبا ما تسمح نظم التشغيل (operating systems) بإعادة توجيه (redirecting) هذين السيلين إلى أجهزة أخرى . وكذلك يتم توصيل سيل ثالث ، وهو سيل الأخطاء القياسي (standard error stream) بالشاشة . ويتم إخراج "رسائل خطأ" (error messages) إلى سيل الأخطاء القياسي .

(*) سنناقش بإذن الله تعالى – في كتابنا التالي "البرمجة المتقدمة بلغة C" ، دار اقرأ للنشر ، الكويت – أربع دوال أخرى (gets, puts, getchar, putchar) تَستخدِم المدخلات القياسية والمخرجات القياسية ، وذلك في الفصل الرابع "الرموز والسلاسل في لغة C ".

صياغة المخرجات باستخدام printf (Formatting Output with printf)

يحتوي اي استدعاء (call) للدالة **printf** على "سلسلة تحكم في الصيغة" (a format control string) تصف (describes) صيغة المخرجات (output format) . وهذه السلسلة تتكون من "محددات تحويل" (conversion specifiers) ، و "رايات / أعلام / إشارات تمييز" (flags) ، و "محددات لعرض المجال" (field widths) ، و "محددات للدِّقة" (precisions)، و "رموز حرفية" (literal characters) . وهذه جميعها مع علامة النسبة المئوية "٪" (percent sign) تكوِّن ما يُعرف بـ "مواصفات التحويل" (conversion specifications) . والدالة **printf** يمكنها إجراء عمليات الصياغة التالية والتي سنتناولها بإذن الله في هذا الفصل :

١ - تقريب (rounding) القيم ذوات النقطة العائمة (floating-point values) لعدد نحدِّده من المواضع العشرية (decimal places) .

٢ - ضبط / محاذاة (aligning) عمود (column) من الأعداد بحيث تظهر النقاط العشرية (decimal points) واحدة فوق الأخرى .

٣ - ضَبْط يمين (ضبط / محاذاة نحو جهة اليمين) (right-justification) ، وضَبْط يسار(left-justification) للمخرجات .

٤ - إدخال (inserting) رموز حرفية (literal characters) عند مواضع محدَّدة بالضبط (precise locations) في سطر من المخرجات .

٥ - تمثيل (representing) الأعداد ذوات النقطة العائمة (floating-point numbers) بالصيغة الأسِّيَّة (exponential format) .

٦ - تمثيل الأعداد الصحيحة دون إشارة (unsigned integers) بالصيغة الثمانية والصيغة السداسي عشرية (octal and hexadecimal format) .

٧ - عرض جميع أنواع البيانات بدقةٍ وعرضِ مجالٍ محدَّدين (fixed-size field widths and precisions) .

صيغة الدالة printf :

```
printf( format-control-string, other-arguments );
```

حيث :

format-control-string : "سلسلة التحكم في الصيغة" تصف صيغة المخرجات (output format)

other-arguments : "الوسطاء الفعليون الآخرون" [ووجودهم في الصيغة اختياري (optional)] وهم في تقابل مع (correspond to) توصيفات / مواصفات التحويل (conversion specifications) الموجودة في "سلسلة التحكم في الصيغة" واحدا واحدا .

ويبدأ كل توصيف تحويل (conversion specification) بعلامة النسبة المئوية وينتهي بمحدِّد تحويل (conversion specifier) . ومن الممكن وجود عدة مواصفات / توصيفات تحويل في سلسلة واحدة (سلسلة التحكم في الصيغة) . وعدم وضع سلسلة التحكم في الصيغة بين حاصرتين مزدوجتين / علامتي تنصيص (quotation marks) يعد خطأً تركيبيا (syntax error)

طباعة الأعداد الصحيحة Printing Integers

العدد الصحيح هو عدد كامل (a whole number) مثل 52- ، 0 ، 776 ولا يحتوي على أي علامة / نقطة عشرية (decimal point) . ويمكن عرض قيم الأعداد الصحيحة بأكثر من صيغة . والجدول التالي يصف محددات التحويل للأعداد الصحيحة .

محدِّد التحويل conversion specifier	الوصف description
d	يعرض عددا صحيحا عشريا ذا إشارة (a signed decimal integer)
i	يعرض عددا صحيحا عشريا ذا إشارة (ملاحظة : المحدِّدان **d, i** مختلفان حين يُستخدمان مع scanf)
o	يعرض عددا صحيحا ثمانيا دون إشارة (an unsigned octal integer)
u	يعرض عددا صحيحا عشريا دون إشارة (an unsigned decimal integer)
x أو **X**	يعرض عددا صحيحا سداسيا عشريا دون إشارة (an unsigned hexadecimal integer) **X** : تؤدي إلى طباعة الأرقام 9←0 والحروف A←F، **x** : تؤدي إلى طباعة الأرقام 9←0 والحروف a←f .

h أو l	h : توضع قبل أي محدد تحويل أعداد صحيحة ليعنى عرض عدد صحيح قصير **short** . l : توضع قبل أي محدد تحويل أعداد صحيحة ليعنى عرض عدد صحيح طويل **long** .

محددات تحويل الأعداد الصحيحة

Integer conversion specifiers

مثال ٧ – ١ :

البرنامج التالي يطبع عددا صحيحا (موجبا أو سالبا) باستخدام كل من محددات تحويل الأعداد الصحيحة .

```
/* Example 7.1 */
/* Using the integer conversion specifiers */
#include <stdio.h>

int main()
{
  printf( "%d\n", 455 );
  printf( "%i\n", 455 );  /* i same as d in printf */
  printf( "%d\n", +455 );
  printf( "%d\n", -455 );
  printf( "%hd\n", 32000 );
  printf( "%ld\n", 2000000000 );
  printf( "%o\n", 455 );
  printf( "%u\n", 455 );
  printf( "%u\n", -455 );
  printf( "%x\n", 455 );
  printf( "%X\n", 455 );

  return 0; /* indicates successful termination */

} /* end main */
```

```
455
455
455
```

```
-455
32000
2000000000
707
455
4294966841
1c7
1C7
```

لاحظ أن الإشارة السالبة فقط هي التي تُطبع ، أما الإشارة الموجبة فلا . وسنرى بإذن الله فيما بعد كيف نفرض طباعة الإشارة الموجبة . ولاحظ كذلك أن القيمة السالبة 455- حين نقرأها بالتوصيف u% تُحَوَّل إلى القيمة دون إشارة (unsigned value) 4294966841 . فطباعة قيمة سالبة باستخدام محدِّد تحويل يتوقع قراءة قيمة دون إشارة يُعَدُّ من الأخطاء التي يجب الانتباه إليها .

طباعة الأعداد ذوات النقطة العائمة

Printing Floating – Point Numbers

القيمة ذات النقطة العائمة تحتوي على علامة / نقطة عشرية (decimal point) مثل : 657.983- ,0.0 ,33.5 . ويمكن عرض هذه القيم ذوات النقطة العائمة بأكثر من صيغة . والجدول التالي يصف محددات تحويل النقطة العائمة .

محدِّد التحويل conversion specifier	الوصف description
e أو **E**	يعرض قيمة ذات نقطة عائمة بالاصطلاح / بالصيغة الأُسِّيَّة (in exponential notation)
f	يعرض قيمة ذات نقطة عائمة بصيغة النقطة الثابتة (in fixed-point notation)
g أو **G**	يعرض قيمة ذات نقطة عائمة إما بصيغة النقطة الثابتة **f** أو بالصيغة الأُسِّيَّة **e** (أو **E**) بناءً على مقدار (magnitude) القيمة .

L	**توضع قبل أي محدِّد تحويل نقطة عائمة للدلالة على أن قيمة ذات نقطة عائمة طويلة مضاعفة** -floating **long double** (point value) **ستُطبع .**

محددات تحويل النقطة العائمة

Floating-point conversion specifiers

- **محدِّدا التحويل E , e يعرضان القيمة ذات النقطة العائمة بالصيغة الأُسِّيَّة ، وهي صيغة الحاسوب المكافئة للصيغة العلمية / للاصطلاح العلمي** (scientific notation) **في الرياضيات . فمثلا القيمة**

150.4582

تمثَّل في الصيغة العلمية هكذا

1.504582×10^2

وتمثَّل في الصيغة الأُسِّيَّة بالحاسوب هكذا

1.504582E+02

وتعني أن 1.504582 **تُضرب في** 10 **مرفوعة للأُس** 2 **/ للقوة الثانية** (second power) (E+02) **. وحرف E يشير إلى كلمة** Exponent .

- **وافتراضيا** (by default) **فإن القيم التي تُطبع باستخدام محددات التحويل** **e, E, f** **تُطبع بدقَّة** (precision) **ستة أرقام** (6 digits) **يمين العلامة / النقطة العشرية . ويمكننا تحديد أي دقة أخرى صراحة** (explicitly) .

- **محدِّد التحويل f يطبع دائما رقما واحدا على الأقل يسار النقطة العشرية .**

- **المحدِّدان E و e يطبعان – على الترتيب – الحرف الكبير E والحرف الصغير e قبل الأُس** (exponent) **، ودائما يطبعان رقما واحدا بالضبط** (exactly one digit) **يسار النقطة العشرية .**

- **المحدِّد g (أو G) يطبع بالصيغة e (E) أو f بدون أصفار خلفية (في أقصى اليمين)** (with no trailing zeros) **[فمثلا** 1.234000 **يُطبع هكذا :** 1.234 **] .**

i) **وتُطبع القيم بالصيغة e (E) إذا كان أُس القيمة** (value's exponent) **– بعد تحويل القيمة إلى الصيغة الأُسِّيَّة – أصغر من** 4- **، أو كان الأُس أكبر من أو مساويا للدقة**

المحدَّدة (specified precision) [وافتراضيا (by default) هي ستة أرقام معنوية (six significant digits) لكل من **g** و **G**] .

ii) وما عدا ذلك ، فإن القيم تُطبع بالصيغة **f** .

والأصفار الخلفية لا تُطبع في الجزء الكسري (fractional part) من قيمة مطلوب طباعتها باستخدام **g** أو **G** . ولطباعة النقطة العشرية يُطلب على الأقل رقم عشري واحد .

مثال ٧ – ٢ : القيم

i) 0.0000875 ii) 8750000.0
iii) 8.75 iv) 87.50 v) 875

تُطبع – على الترتيب – باستخدام المحوِّل **g** هكذا :

i) 8.75e-05 ii) 8.75e+06
iii) 8.75 iv) 87.5 v) 875

لاحظ في (i أن القيمة 0.0000875 قد استخدمت الاصطلاح **e** لأنه بعد تحويلها إلى الصيغة الأُسِّيَّة فإن الأُس (5-) يكون أصغر من 4- . بينما في (ii القيمة 8750000.0 تستخدم الاصطلاح **e** لأن أُسَّها (6) مساوٍ للدقة الافتراضية (default precision) .

- ودقَّة (precision) محدِّدي التحويل **G** , **g** هي أكبر عدد من الأرقام المعنوية (max. no. of significant digits) المطبوعة ، بما في ذلك الرقم الموجود يسار النقطة العشرية . فالقيمة 1234567.0 تُطبع هكذا : 1.23457e+06 باستخدام محدد التحويل g% [تذكَّر أن الدقة الافتراضية لجميع محددات تحويل النقطة العائمة تساوي 6] . ولاحظ وجود 6 أرقام معنوية في النتيجة المطبوعة .
- الفارق بين **G** و **g** مطابق للفارق بين **E** و **e** عندما تُطبع القيمة بالصيغة الأُسِّيَّة : حيث يؤدي الحرف الصغير **g** إلى طباعة الحرف الصغير **e** في المخرجات ، بينما يؤدي الحرف الكبير **G** إلى طباعة الحرف الكبير **E** في المخرجات .

وعموما عند طباعة البيانات يجب الانتباه إلى الحالات التي قد تظهر فيها البيانات بصورة غير دقيقة (imprecise) بسبب الصياغة (formatting) [مثلا : أخطاء التقريب (rounding errors) الناتجة عن تحديد الدقة] .

مثال ٧ – ٣ :

البرنامج التالي يطبع عددا ذا نقطة عائمة باستخدام كل من محددات تحويل النقطة العائمة .

```
/* Example 7.3 */
/* Printing floating-point numbers with
   floating-point conversion specifiers */

#include <stdio.h>

int main()
{
   printf( "%e\n", 1234567.89 );
   printf( "%e\n", +1234567.89 );
   printf( "%e\n", -1234567.89 );
   printf( "%E\n", 1234567.89 );
   printf( "%f\n", 1234567.89 );
   printf( "%g\n", 1234567.89 );
   printf( "%G\n", 1234567.89 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

لاحظ أن محددات التحويل **%g** ,**%e**, **%E** تؤدي إلى تقريب (rounding) القيمة في المخرجات ، بعكس محدد التحويل **%f** الذي لا يؤدي إلى ذلك التقريب .

طباعة الرموز وسلاسل الرموز Printing Characters and Strings

يُستخدم محددا التحويل **c** و **s** لطباعة الرموز المفردة (individual characters) وسلاسل الرموز (strings) على الترتيب . ومحدد التحويل **c** يتطلب وسيطا رمزيا (a **char**

(argument) ، بينما يتطلب محدد التحويل **s** مؤشّرا (*) (a pointer) إلى رمز **(char)** كوسيط (an argument) . ويؤدي محدد التحويل **s** إلى طباعة الرموز المتتالية إلى أن نقابل رمز الإنهاء الخالي (terminating null character) ('\0') .

مثال ٧ – ٤ :

البرنامج التالي يطبع رموزا وسلاسل رموز باستخدام محدِّدي التحويل c و s .

```
/* Example 7.4 */
/* Printing strings and characters */
#include <stdio.h>

int main()
{
  char character = 'A'; /* initialize char */
  char string[] = "This is a string"; /* initialize char array */
  const char *stringPtr = "This is also a string"; /* char pointer */

  printf( "%c\n", character );
  printf( "%s\n", "This is a string" );
  printf( "%s\n", string );
  printf( "%s\n", stringPtr );

  return 0; /* indicates successful termination */

} /* end main */
```

```
A
This is a string
This is a string
This is also a string
```

ومن الأخطاء التي نود التنبيه إليها حين التعامل مع الرموز وسلاسل الرموز وطباعتها :

- **استخدام %c لطباعة سلسلة رموز يُعد خطأ ، فمحدد التحويل %c يتوقع وسيطا رمزيا char [بينما سلسلة الرموز تُعد مؤشرا إلى رمز ، أي : * char a] .**

(*) سنتناول بإذن الله موضوع "المؤشرات" (pointers) بالتفصيل في الكتاب التالي "البرمجة المتقدمة بلغة C " ، دار اقرأ للنشر ، الكويت .

- في بعض النظم يؤدي استخدام **%s** لطباعة وسيط رمزي **char** إلى خطأ قاتل وقت التنفيذ (fatal execution-time error) يطلق عليه "تَعَدِّي / انتهاك لقواعد الوصول / التناول" (access violation) ، فمحدد التحويل **%s** يتوقع وسيطا من نوع : مؤشر إلى رمز **char** .
- استخدام حاصرتين مفردتين (single quotes) حول سلسلة رموز يُعد خطأ تركيبيا ، فيجب استخدام حاصرتين مزدوجتين (double quotes) حول سلسلة الرموز .
- استخدام حاصرتين مزدوجتين حول ثابت رمزي (a character constant) يؤدي إلى إنشاء (creating) سلسلة رموز مكونة من رمزين ، حيث الرمز الثاني هو رمز الإنهاء الخالي (terminating null character) . فالثابت الرمزي هو رمز مفرد (single character) يحاط بحاصرتين مفردتين .

محدد التحويل ٪

يؤدي محدد التحويل ٪ إلى طباعة رمز النسبة المئوية ٪ (percent character).

فمثلا عبارة الطباعة

```
printf( "Printing a %% in a format control string\n" );
```

تستخدم ٪ ٪ لطباعة الرمز ٪ في سلسلة رموز ، حيث تؤدي هذه العبارة إلى طباعة الرسالة

```
Printing a % in a format control string
```

ومن الخطأ محاولة طباعة رمز النسبة المئوية الحرفي (literal percent character) باستخدام ٪ بدلا من ٪ ٪ في سلسلة التحكم في الصيغة (format control string) . فعندما يظهر الرمز ٪ في أي سلسلة تحكم في الصيغة يجب أن يتبعه محدد تحويل .

الطباعة بتحديد عرض المجال والدقة
Printing with Field Widths and Precision

يمكننا تحديد السعة المضبوطة للمجال (exact size of the field) الذي تُطبع فيه البيانات بتحديد ما يسمَّى "عرض المجال" (field width) ، وذلك بإدخال عدد صحيح (an integer) – يمثل عرض المجال – بين علامة النسبة المئوية (٪) ومحدد التحويل ، كأن نكتب مثلا: **%4d** . وإذا كان عرض المجال المحدَّد أكبر من السعة اللازمة لطباعة البيانات ، فعادةً تُطبَع البيانات في أقصى يمين (right-justified) هذا المجال المحدَّد . وإذا كان عرض المجال المحدَّد أصغر من السعة اللازمة لطباعة البيانات فإن عرض المجال يُزاد (increased) لطباعة البيانات . ويلاحظ أن إشارة الناقص (minus sign) لأي قيمة سالبة (a negative value)

تحتل موضع رمز واحد (one character position) في عرض المجال . وعموما يمكن استخدام عرض المجال مع جميع محددات التحويل .

مثال ٧ – ٥ :

البرنامج التالي يطبع مجموعتين من الأعداد ، تتكون كل مجموعة منهما من خمسة أعداد .

```
/* Example 7.5 */
/* Printing integers right-justified */
#include <stdio.h>

int main()
{
   printf( "%4d\n", 1 );
   printf( "%4d\n", 12 );
   printf( "%4d\n", 123 );
   printf( "%4d\n", 1234 );
   printf( "%4d\n\n", 12345 ); /* data too large */

   printf( "%4d\n", -1 );
   printf( "%4d\n", -12 );
   printf( "%4d\n", -123 );
   printf( "%4d\n", -1234 ); /* data too large */
   printf( "%4d\n", -12345 ); /* data too large */

   return 0; /* indicates successful termination */

} /* end main */
```

```
   1
  12
 123
1234
12345

  -1
 -12
-123
-1234
-12345
```

ويلاحظ أن عدم تحديد / ترك عرض مجال (field width) كبير كافي لطباعة قيمةٍ ما يمكن أن يؤدي إلى زحزحة (offset) بيانات أخرى تُطبع ، وقد يؤدي إلى التباس في قيم المخرجات (confusing outputs) .

والدالة **printf** يمكنها أيضا تحديد الدِّقَّة (precision) التي تُطبع بها البيانات . والدقة تُمَثَّل بعدد صحيح (an integer) له معاني مختلفة بالنسبة لأنواع البيانات (data types) المختلفة :

- فعند استخدام الدقة (العدد الصحيح) مع محددات تحويل الأعداد الصحيحة ، فإن الدقة تعني : أقل عدد من الأرقام (minimum number of digits) التي تُطبع . وإذا احتوت القيمة المطبوعة على عدد من الأرقام أقل من الدقة المحددة ، فإن أصفارا تضاف قبل (prefixed to) القيمة المطبوعة ، حتى يصبح العدد الكلي للأرقام مساويا للدقة المحددة [مثلا : 00891 ، إذا كانت الدقة تساوي 5] . والدقة الافتراضية (default precision) للأعداد الصحيحة هي 1 .
- وعند استخدام الدقة مع محددات تحويل النقطة العائمة **e, E, f** ، فإن الدقة تعني عدد الأرقام التي تظهر يمين النقطة العشرية .
- وعند استخدام الدقة مع محَدِّدَي التحويل **g, G** ، فإن الدقة تعني : أكبر عدد من الأرقام المعنوية (max. no. of significant digits) التي تُطبع .
- وعند استخدام الدقة مع محدد التحويل **s** ، فإن الدقة تعني أكبر عدد من الرموز التي تُطبع من سلسلة الرموز .

وأما كيفية استخدام أو تحديد الدقة فهي : أن نضع نقطة عشرية (.) (decimal point) يمينها عدد صحيح يمثل الدقة ، وذلك بين علامة النسبة المئوية (٪) ومحدِّد التحويل ، كأن نكتب مثلا : **%.4f** أو **%.8d** .

ويلاحظ أنه في حالة طباعة قيمة ذات نقطة عائمة باستخدام دقة أصغر من العدد الأصلي للأرقام / للمواضع العشرية (original number of decimal places) في القيمة ، فإن القيمة تُقرَّب (rounded) .

مثال ٧ – ٦ :

البرنامج التالي يوضح استخدام الدقة في سلاسل التحكم في الصيغة (format control strings) لطباعة بيانات من أنواع مختلفة .

```
/* Example 7.6 */
/* Using precision while printing integers,
   floating-point numbers, and strings */
#include <stdio.h>

int main()
{
   int i = 873;                /* initialize int i */
   double f = 123.94536;       /* initialize double f */
   char s[] = "Happy EidulFitr"; /* initialize char array s */

   printf( "Using precision for integers\n" );
   printf( "\t%.4d\n\t%.9d\n\n", i, i );

   printf( "Using precision for floating-point numbers\n" );
   printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );

   printf( "Using precision for strings\n" );
   printf( "\t%.9s\n", s );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Using precision for integers
        0873
        000000873
Using precision for floating-point numbers
        123.945
        1.239e+002
        124
Using precision for strings
        Happy Eid
```

ويمكننا الجمع بين (combining) عرض المجال (field width) والدقة (precision) ، وذلك بوضع عرض المجال تليه نقطة عشرية تليها الدقة ، وذلك بين علامة النسبة المئوية ومحدِّد التحويل ، كما هو موضح في المثال التالي .

مثال ٧ – ٧ : العبارة

```
printf( "%9.3f", 123.456789 );
```

تؤدي إلى طباعة القيمة 123.457 [بثلاثة أرقام (عشرية) يمين النقطة العشرية] في أقصى اليمين (right justified) في مجال عرضه تسعة أرقام (nine-digit field) .

ومن الممكن تحديد عرض المجال والدقة باستخدام تعابير صحيحة (integer expressions) في قائمة الوسطاء (argument list) التي تلي سلسلة التحكم في الصيغة (format control string) . ولاستخدام هذه الإمكانية / الخاصية (feature) : نُدخِل (insert) علامة النجمة (*) (an asterisk) في موضع (place) عرض المجال أو الدقة أو كليهما . فيتم حساب قيمة (evaluating) الوسيط الصحيح **int** الموائم (matching **int** argument) في قائمة الوسطاء (argument list) ويُستخدَم في موضع النجمة . وقيمة عرض أي مجال قد تكون موجبة أو سالبة . وإن كانت سالبة فإنها تعني طباعة المخرجات في أقصى اليسار (left-justified) في المجال (كما سنوضح ذلك بإذن الله بعد قليل) .

مثال ٧ – ٨ :

العبارة

```
printf( "%*.*f", 7, 2, 98.736 );
```

تستخدم 7 لعرض المجال ، و 2 للدِّقة ، وتطبع القيمة 98.74 في أقصى يمين المجال (right justified) .

استخدام الرايات/الأعلام/إشارات التمييز في سلسلة التحكم في الصيغة في الدالة printf
Using Flags in the printf Format Control String

الدالة **printf** يمكنها أيضا استخدام ما يسمَّى بالرايات / الأعلام / إشارات التمييز (flags) والتي تزيد من قدراتها على صياغة المخرجات (output formatting capabilities) .

الجدول التالي يعرض الخمس رايات / إشارات التمييز التي يمكننا استخدامها في سلاسل التحكم في الصيغة .

الراية / إشارة التمييز Flag	الوصف / المعنى Meaning / Description
- (إشارة ناقص)	اطبع المخرجات في أقصى يسار (left justified) المجال المحدد .

+ (إشارة زائد)	اطبع إشارة + قبل القيم الموجبة ، وإشارة – قبل القيم السالبة .
فراغ (space)	اطبع / اترك فراغا قبل أي قيمة موجبة لم تُطبع باستخدام إشارة التمييز + .
#	• ضع صفرا **0** قبل القيمة المطبوعة عند استخدام محدد التحويل الثماني **o**(octal conversion specifier) • ضع **0x** أو **0X** قبل القيمة المطبوعة عند استخدام محدد التحويل السداسي عشري **x** أو **X** (hexadecimal conversion specifiers) . • ضع نقطة عشرية (decimal point) لأي عدد ذي نقطة عائمة تتم طباعته بواسطة أي من **e, E, f, g, G** ولا يحتوي على جزء كسري (a fractional part) . [عادة تُطبع النقطة العشرية فقط إذا كان هناك رقم (عشري) يليها] . وفي حالة المحددين **g, G** فإن الأصفار الخلفية (في أقصى اليمين) (trailing zeros) لا تُحذف .
0 (صفر)	أكمل (حَشْو) المجال (pad the field) بأصفار رائدة / متقدمة (في البداية على اليسار) (leading zeros) .

الرايات / إشارات التمييز في سلسلة التحكم في الصيغة
Format control string flags

ولاستخدام راية / إشارة تمييز (flag) في سلسلة تحكم في الصيغة فإننا نضع إشارة التمييز يمين علامة النسبة المئوية ٪ مباشرة . ويمكننا الجمع بين (combining) عدة إشارات تمييز في محدِّد تحويل واحد .

مثال ٧ – ٩ :

البرنامج التالي يعرض كلا من الضبط جهة اليمين (right-justification) والضبط جهة اليسار (left-justification) لكل من سلسلة رموز ، وعدد صحيح ، ورمز ، وعدد ذي نقطة عائمة .

```
/* Example 7.9 */
/* Right justifying and left justifying values */
#include <stdio.h>
```

```
int main()
{
  printf( "%10s%10d%10c%10f\n\n", "salam", 7, 'a', 1.23 );
  printf( "%-10s%-10d%-10c%-10f\n", "salam", 7, 'a', 1.23 );

  return 0; /* indicates successful termination */

} /* end main */
```

```
    salam         7         a    1.230000

salam      7         a         1.230000
```

مثال ٧ – ١٠ :

البرنامج التالي يطبع عددا موجبا وآخر سالبا ، وكلاهما مرة باستخدام إشارة التمييز (flag)+ وأخرى دون استخدامها . لاحظ أن إشارة الناقص تُطبع في الحالتين ، بينما إشارة الزائد لا تُطبع إلا مع استخدام إشارة التمييز + .

```
/* Example 7.10 */
/* Printing numbers with and without the + flag */
#include <stdio.h>

int main()
{
  printf( "%d\n%d\n", 786, -786 );
  printf( "%+d\n%+d\n", 786, -786 );

  return 0; /* indicates successful termination */

} /* end main */
```

```
786
-786
+786
-786
```

مثال ٧ – ١١ :

البرنامج التالي يستخدم راية الفراغ (space flag) لوضع فراغ قبل العدد الموجب. ولاحظ فائدة ذلك في ضبط محاذاة (aligning) الأعداد الموجبة والسالبة – التي تحتوي على عدد الأرقام نفسه – فوق بعضها البعض . لاحظ أن القيمة 547- لا يسبقها فراغ في المخرجات بسبب إشارتها السالبة (الناقص) .

```
/* Example 7.11 */
/* Printing a space before signed values
   not preceded by + or - */
#include <stdio.h>

int main()
{
   printf( "% d\n% d\n", 547, -547 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
 547
-547
```

مثال ٧ – ١٢ :

البرنامج التالي يستخدم إشارة التمييز (flag) # لوضع 0 قبل القيمة الثمانية ، ووضع 0x و 0X قبل القيمتين السداسي عشريتين ، ولفرض إظهار النقطة العشرية على أي قيمة تطبع بواسطة المحدِّد g .

```
/* Examle 7.12 */
/* Using the # flag with conversion specifiers
   o, x, X and any floating-point specifier */
#include <stdio.h>

int main()
{
   int c = 1427;      /* initialize c */
   double p = 1427.0; /* initialize p */
```

```
   printf( "%#o\n", c );
   printf( "%#x\n", c );
   printf( "%#X\n", c );
   printf( "\n%g\n", p );
   printf( "%#g\n", p );

   return 0; /* indicates successful termination */

} /* end main */
```

```
02623
0x593
0X593

1427
1427.00
```

مثال ٧ – ١٣ :

البرنامج التالي يجمع بين (combines) إشارة التمييز (flag) + وإشارة التمييز (flag) 0 لطباعة القيمة 452 في مجال عرضه 9 فراغات (9-space field) مع إشارة + وأصفار متقدمة ، ثم طباعة 452 مرة أخرى باستخدام إشارة التمييز 0 فقط في مجال عرضه 9 فراغات .

```
/* Example 7.13 */
/* Printing with the 0( zero ) flag fills in leading zeros */
#include <stdio.h>

int main()
{
   printf( "%+09d\n", 452 );
   printf( "%09d\n", 452 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
+00000452
000000452
```

طباعة الرموز الحرفية ومتتابعات الهروب
Printing Literals and Escape Sequences

معظم الرموز الحرفية (literal characters) التي نود طباعتها في عبارة **printf** يمكن أن تحتويها ببساطة سلسلة التحكم في الصيغة . إلا أن هناك بعض الرموز التي تؤدي إلى "مشاكل" كالحاصرة المزدوجة / علامة الاقتباس (") (quotation mark) التي تَحُدُّ من (delimits) سلسلة التحكم في الصيغة نفسها . وهناك عدة رموز تحكم (control characters) – كرمز السطر الجديد (newline) ورمز الجدولة (tab) – التي يجب أن تمثَّل بمتتابعة هروب (escape sequence) . وأي متتابعة هروب تُمَثَّل بشَرْطة مائلة خلفية (\) (backslash) يليها رمز هروب معين . وأي محاولة لطباعة حاصرة مفردة (a single quote) أو حاصرة مزدوجة (a double quote) أو علامة استفهام أو رمز الشرطة المائلة الخلفية (backslash) دون أن نضع الشرطة المائلة الخلفية قبل هذا الرمز المطلوب طباعته – كبيانات حرفية (literal data) في عبارة **printf** – لتكوِّن متتابعة هروب / إفلات صحيحة (a proper escape sequence) تُعَدُّ خطأً .

والجدول التالي يعرض قائمة بمتتابعات الهروب وتأثير كل منها .

متتابعة الهروب Escape sequence		التأثير / الوصف Description / Effect
حاصرة مفردة	\' (single quote)	تطبع رمز الحاصرة المفردة (')
حاصرة مزدوجة	\" (double quote)	تطبع رمز الحاصرة المزدوجة (")
علامة استفهام	\? (question mark	تطبع رمز علامة الاستفهام (?)
شَـــرطة مائلـــة خلفية	\\ (backslash)	تطبع رمز الشرطة المائلة الخلفية (\)
تنبيه أو جرس	\a (alert or bell)	تؤدي إلى (جرس) يُسمع(audible) أو علامة تنبيه تُرى (visual)
خطوة للخلف	\b (backspace)	تحرِّك المؤشر(cursor) موضعا واحدا للخلف على السطر الحالي
صفحة جديدة أو تغذية صيغة	\f (new page or form feed)	تحرِّك المؤشر إلى بداية الصفحة المنطقية (logical) التالية
سطر جديد	\n (newline)	تحرِّك المؤشر إلى بداية السطر التالي
عودة العربة	\r (carriage return)	تحرِّك المؤشر إلى بداية السطر الحالي

جدولة أفقية	\t (horizontal tab)	تحرِّك المؤشر إلى موضع الجدولة الأفقية التالي
جدولة رأسية	\v (vertical tab)	تحرِّك المؤشر إلى موضع الجدولة الرأسية التالي

متتابعات الهروب
Escape Sequences

صياغة المدخلات باستخدام scanf (Formatting Input with scanf)

يمكن صياغة المدخلات بدقة (precise input formatting) باستخدام **scanf** .
وأي عبارة **scanf** تحتوي على سلسلة تحكم في الصيغة تصف صيغة البيانات التي ستُدخل . وتتكون سلسلة التحكم في الصيغة من محددات تحويل ورموز حرفية . وتتمتع الدالة **scanf** بالقدرات (capabilities) التالية على صياغة المدخلات :

١) إدخال جميع أنواع (all types) البيانات .
٢) إدخال رموز معينة (specific characters) من سيل المدخلات (input stream) .
٣) تخطِّي (skipping) رموز معينة في سيل المدخلات .

والدالة **scanf** تُكتب بالصيغة التالية :

scanf(format-control-string , other-arguments);

حيث

format-control-string : تصف صيغ (formats) المدخلات .

other-arguments : مؤشرات (pointers) إلى المتغيرات التي سيتم تخزين المدخلات فيها.

الجدول التالي يلخص محددات التحويل المستخدمة لإدخال جميع أنواع البيانات . وبعد الجدول نستعرض عددا من الأمثلة لبرامج توضح قراءة البيانات باستخدام محددات تحويل **scanf** المختلفة .

الوصف **Description**	محدِّد التحويل **Conversion specifier**
	الأعداد الصحيحة (Integers)
اقرأ عددا صحيحا عشريا ذا إشارة اختيارية (an optionally signed decimal integer). **الوسيط الفعلي** (argument) **المقابل يكون مؤشرا** (a pointer) **إلى عدد صحيح .**	**d**
اقرأ عددا صحيحا سداسيا عشريا أو ثمانيا أو عشريا ذا إشارة اختيارية (an optionally signed decimal, octal or hexadecimal integer) . **الوسيط الفعلي المقابل يكون مؤشرا إلى عدد صحيح .**	**i**
اقرأ عددا صحيحا ثمانيا . الوسيط الفعلي المقابل يكون مؤشرا إلى عدد صحيح دون إشارة (an unsigned integer) .	**o**
اقرأ عددا صحيحا عشريا دون إشارة . الوسيط الفعلي المقابل يكون مؤشرا إلى عدد صحيح دون إشارة.	**u**
اقرأ عددا صحيحا سداسيا عشريا . الوسيط الفعلي المقابل يكون مؤشرا إلى عدد صحيح دون إشارة.	**x** أو **X**
يوضع قبل أي محدد تحويل أعداد صحيحة للدلالة على أن عددا صحيحا قصيرا short أو طويلا long سيُدخَل (to be input).	**h** أو **l**
	الأعداد ذوات النقطة العائمة **Floating-point numbers**
اقرأ قيمة ذات نقطة عائمة . الوسيط الفعلي المقابل يكون مؤشرا إلى متغير ذي نقطة عائمة .	**أي من: e, E, f, g, G**

l أو **L**	يوضع قبل أي محدد تحويل قيم ذوات نقطة عائمة للدلالة على أن قيمة مضاعفة **double** أو قيمة مضاعفة طويلة **long** (value **double**) ستُدخَل . الوسيط الفعلي المقابل يكون مؤشرا إلى متغير **double** أو متغير **long double** .
الرموز وسلاسل الرموز **Characters and strings**	
c	اقرأ رمزا . الوسيط الفعلي المقابل يكون مؤشرا إلى رمز **char** a. ولا يُضاف الرمز الخالي ('\0') (null) .
s	اقرأ سلسلة رموز . الوسيط الفعلي المقابل يكون مؤشرا إلى منظومة من النوع **char** ، وبحيث تكون كبيرة بما يكفي للاحتفاظ بسلسلة الرموز ورمز الإنهاء الخالي ('\0') الذي يضاف تلقائيا .
مجموعة المسح **Scan set**	
[رموز المسح] [scan characters]	قم بعملية مسح لسلسة رموز (scan a string) بحثا عن مجموعة رموز لتُخزَّن في منظومة.
محددات تحويل متنوعة **Miscellaneous**	
p	اقرأ عنوانا (an address) صيغته (form) هي الصيغة نفسها التي نحصل عليها عندما نطبع عنوانا باستخدام **%p** في عبارة **printf** .
n	قم بتخزين عدد الرموز التي تم إدخالها حتى الآن في عبارة **scanf** هذه . الوسيط الفعلي المقابل يكون مؤشرا إلى عدد صحيح .

%	قم بتخطِّي (skip) علامة نسبة مئوية (%) في المدخلات .

جدول محددات التحويل المستخدمة مع scanf
scanf conversion specifiers

مثال ٧ – ١٤ : البرنامج التالي يقرأ أعدادا صحيحة باستخدام محددات التحويل المختلفة للأعداد الصحيحة، ويطبع الأعداد الصحيحة كأعداد عشرية (decimal numbers) . لاحظ أنه بإمكان %i إدخال (inputting) أعداد عشرية وثمانية وسداسية عشرية .

```
/* Example 7.14 */
/* Reading integers */
#include <stdio.h>

int main()
{
   int a;
   int b;
   int c;
   int d;
   int e;
   int f;
   int g;

   printf( "Enter seven integers: " );
   scanf( "%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g );

   printf( "The input displayed as decimal integers is:\n" );
   printf( "%d %d %d %d %d %d %d\n", a, b, c, d, e, f, g );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter seven integers:   -70 -70 070 0×70  70 70 70
The input displayed as decimal integers is:
-70 -70 56 112 56 70 112
```

وعند إدخال الأعداد ذوات النقطة العائمة يمكننا استخدام أي من محددات تحويل النقطة العائمة **e**, **E**, **f**, **g**, **G** .

مثال ٧ – ١٥ : البرنامج التالي يقرأ ثلاثة أعداد ذوات نقطة عائمة **a, b, c** بالأنواع الثلاثة لمحددات تحويل النقطة العائمة **e**, **f**, **g** . ثم يطبع الأعداد الثلاثة كلها باصطلاح النقطة العائمة البسيط بمحدد التحويل **f** .

```
/* Example 7.15 */
/* Reading floating-point numbers */
#include <stdio.h>

/* function main begins program execution */
int main()
{
  double a;
  double b;
  double c;

  printf( "Enter three floating-point numbers: \n" );
  scanf( "%le%lf%lg", &a, &b, &c );

  printf( "Here are the numbers entered in plain\n" );
  printf( "floating-point notation:\n" );
  printf( "%f\n%f\n%f\n", a, b, c );

  return 0; /* indicates successful termination */

} /* end main */
```

```
Enter three floating point numbers:
1.27987 1.27987e+03 3.38476e-06
Here are the numbers entered in plain
floating- point notation:
1.279870
1279.870000
0.000003
```

يلاحَظ أن مخرجات هذا البرنامج تؤكد حقيقة أن القيم ذوات النقطة العائمة تكون عموما غير دقيقة (imprecise) . وهذا واضح في القيمة الثالثة المطبوعة [0.000003 ، وقيمتها المقروءة 3.38476e-06] .

ويمكننا إدخال الرموز والسلاسل باستخدام محددي التحويل **c** و **s** على الترتيب .

مثال ٧ – ١٦ : البرنامج التالي يطلب من المستخدم إدخال سلسلة رموز . ثم يقوم البرنامج بإدخال الرمز الأول في السلسلة باستخدام **%c** ، ويخزنه في المتغير الرمزي (character variable)**x** ، ثم يُدخِل بقية السلسلة باستخدام **%s** ، ويخزنها في منظومة الرموز **y** (character array) .

```
/* Example 7.16 */
/* Reading characters and strings */
#include <stdio.h>

int main()
{
  char x;
  char y[ 9 ];

  printf( "Enter a string: " );
  scanf( "%c%s", &x, y );

  printf( "The input was:\n" );
  printf( "the character \"%c\" ", x );
  printf( "and the string \"%s\"\n", y );

  return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a string: Friday
The input was:
the character "F" and the string "riday"
```

ويمكننا إدخال (inputting) متتابعة من الرموز (sequence of characters) باستخدام "مجموعة مسح" (**scan set**) . ومجموعة المسح هي مجموعة من الرموز محاطة بقوسين مربعين [ ] وتسبقها علامة النسبة المئوية (مثل "[kmtw] % ") في سلسلة التحكم في الصيغة (format control string) . وتقوم مجموعة المسح بعملية مسح للرموز في سيل الإدخال (input stream) ، باحثة فقط عن تلك الرموز التي تتفق / تتواءم (match) مع تلك الرموز الموجودة في مجموعة المسح (كرموز مفردة في المجموعة وليس كسلسلة رموز) . وكلما وجدت رمزا متوائما (أي موجودا في مجموعة المسح) فإن هذا الرمز يُخَزَّن في الوسيط الفعلي المقابل (corresponding argument) لمجموعة المسح – وهو مؤشر إلى منظومة رموز (a pointer to a character array) . وتتوقف مجموعة المسح عن إدخال أي رمز عندما تقابل رمزا ليس موجودا في مجموعة المسح . وفي حالة ما إذا كان أول رمز في سيل الإدخال (input stream) ليس متوائما (أي ليس موجودا في مجموعة المسح) فإن الرمز الخالي (null character) فقط هو الذي يُخَزَّن في المنظومة .

مثال ٧ – ١٧ : البرنامج التالي يستخدم مجموعة المسح [aeiou] { أي مجموعة الحروف المتحركة (vowels)} لمسح سيل المدخلات بحثا عن الحروف المتحركة . لاحظ أن الحروف السبعة الأولى من المدخلات قد تمت قراءتها (لأنها جميعا من الحروف المتحركة) ، بينما الحرف الثامن (h) ليس في مجموعة المسح (لأنه ليس حرفا متحركا) ولذلك فعنده تنتهي عملية المسح .

```
/* Example 7.17 */
/* Using a scan set */
#include <stdio.h>

/* function main begins program execution */
int main()
{
  char z[ 9 ];   /* define array z */

  printf( "Enter string: " );
  scanf( "%[aeiou]", z ); /* search for set of characters */

  printf( "The input was \"%s\"\n", z );

  return 0; /* indicates successful termination */

} /* end main */
```

```
Enter string: ooeeooahah
The input was "ooeeooa"
```

ويمكن لمجموعة المسح أن تُستخدم للقيام بعملية مسح لسيل المدخلات بحثا عن الرموز غير الموجودة في مجموعة المسح ، وفي هذه الحالة نَستخدم فعليا ما يُطلق عليها "مجموعة المسح المعكوسة / المقلوبة" **(inverted scan set)** ، وتتوقف عملية المسح حينئذ حين نقابل رمزا موجودا في مجموعة المسح . وأما كيفية إنشاء (creating) مجموعة المسح المعكوسة فهي ببساطة أن نضع علامة "الإقحام" (^)(caret) قبل رموز المسح (scan characters) داخل القوسين المربعين { مثلا: [^grqmr] } . وهذا يؤدي إلى تخزين (storing) الرموز – في سيل الإدخال – التي لا تظهر في مجموعة المسح ، إلى أن نقابل رمزا موجودا في مجموعة المسح (المعكوسة) ، وعندها ينتهي الإدخال (input terminates) .

مثال ٧ – ١٨ : البرنامج التالي يستخدم مجموعة المسح المعكوسة [^aeiou] للبحث عن الحروف الساكنة (consonants) ، أي للبحث عن الحروف غير المتحركة (nonvowels) .

```
/* Example 7.18 */
/* Using an inverted scan set */
#include <stdio.h>

int main()
{
  char z[ 9 ];

  printf( "Enter a string: " );
  scanf( "%[^aeiou]", z ); /* inverted scan set */

  printf( "The input was \"%s\"\n", z );

  return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a string: String
The input was "Str"
```

ويمكن استخدام "عرض مجال" (a field width) في محدِّد تحويل **scanf** لقراءة عدد معين من الرموز (a specific number of characters) من سيل الإدخال .

مثال ٧ – ١٩ : البرنامج التالي يُدخِل (inputs) متسلسلة (a series) من الأرقام المتتابعة / المتعاقبة (consecutive digits) كعدد صحيح من رقمين (a two-digit integer) وعدد صحيح آخر مكون من الأرقام المتبقية (remaining digits) في سيل المدخلات (input stream) .

```
/* Example 7.19 */
/* inputting data with a field width */
#include <stdio.h>

int main()
{
  int x;
  int y;

  printf( "Enter a six digit integer: " );
  scanf( "%2d%d", &x, &y );

  printf( "The integers input were %d and %d\n", x, y );

  return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a six digit integer: 123456
The integers input were 12 and 3456
```

في أحيان كثيرة يكون من الضروري تَخَطِّي (skipping) رموز معينة في سيل المدخلات . فمثلا يمكننا إدخال تاريخٍ ما هكذا :

9-17-1424

ونحتاج لتخزين كل عدد في هذا التاريخ ، ولكن الشَّرْطَتين (-,-) (2 dashes) اللتين تفصلان الأعداد عن بعضها البعض يمكننا تجاهلهما . ولحذف (to eliminate) أي رموز غير ضرورية ، فإننا نضعها في سلسلة التحكم في الصيغة في عبارة **scanf** [ملاحظة : رموز الفراغ البيضاء

(whitespace characters) – كالفراغ (space) ، والسطر الجديد (newline) ، والجدولة (tab) – تتخطَّى جميع رموز الفراغ البيضاء الرائدة / المتقدمة (leading whitespace)] .

مثال ٧ – ٢٠ : كي نتخطَّى (to skip) الشَّرطتين (2 dashes) في المدخلات

9-17-1424

نستخدم العبارة

```
scanf( "%d-%d-%d", &month, &day, &year );
```

ورغم أن **scanf** هذه تلغي (eliminates) فعلا الشَّرطتين في المدخلات السابقة إلا أنه من الممكن عند إدخال التاريخ أن يتم إدخاله هكذا :

9/17/1424

وفي هذه الحالة فإن **scanf** السابقة سوف لا تلغي الرموز غير الضرورية (unnecessary characters) . ولهذا السبب فإن **scanf** تستخدم ما يُعرف بـ " رمز كَبْت / حَذْف / إلغاء الإسناد / التخصيص " (*)(assignment suppression character) . ورمز الكبت هذا يُمَكِّن **scanf** من قراءة (reading) أي نوع من البيانات من المدخلات ، وإهماله / تجاهله (discarding it) دون إسناده إلى أي متغير .

مثال ٧ – ٢١ : البرنامج التالي يستخدم رمز كَبْت / حَذْف / إلغاء الإسناد (assignment suppression character)(*) في محدد التحويل **%c** للدلالة على أن أي رمز يظهر في سيل المدخلات يجب أن يُقرأ ويُهمَل (discarded) . فالأعداد التي تمثِّل الشهر واليوم والسنة (month, day, year) هي فقط التي يتم تخزينها . والبرنامج يطبع أيضا قيم المتغيرات لتوضيح أن هذه القيم قد تم فعلا إدخالها بطريقة صحيحة . ولاحظ أن أي قائمة وسطاء فعليين (argument list) لأي استدعاء (call) للدالة **scanf** لا تحتوي على متغيرات (variables) لمحددات التحويل التي تستخدم رمز كَبْت / حَذْف الإسناد . والرموز المقابلة (corresponding characters) يتم تجاهلها ببساطة .

```
/* Example 7.21 */
/* Reading and discarding characters from the input stream */
#include <stdio.h>

int main()
{
  int month1; /* define month1 */
```

```
   int day1;   /* define day1 */
   int year1;  /* define year1 */
   int month2; /* define month2 */
   int day2;   /* define day2 */
   int year2;  /* define year2 */

   printf( "Enter a date in the form mm-dd-yyyy: " );
   scanf( "%d%*c%d%*c%d", &month1, &day1, &year1 );

   printf( "month = %d  day = %d  year = %d\n\n", month1, day1, year1 );

   printf( "Enter a date in the form mm/dd/yyyy: " );
   scanf( "%d%*c%d%*c%d", &month2, &day2, &year2 );

   printf( "month = %d  day = %d  year = %d\n", month2, day2, year2 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a date in the form mm-dd-yyyy: 9-17-1424
month = 9  day = 17  year = 1424

Enter a date in the form mm/dd/yyyy: 9/17/1424
month = 9  day = 17  year = 1424
```

تمرينات رقم ٧

٧-١) اكتشف الخطأ في كل من العبارات التالية ، ووضِّح كيف يمكن تصحيح الخطأ .

أ) العبارة التالية يجب أن تطبع الرمز 'c' .

```
printf( "%s\n", 'c' );
```

ب) العبارة التالية يجب أن تطبع 9.375% .

```
printf( "%.3f%", 9.375 );
```

ج) العبارة التالية يجب أن تطبع الرمز الأول من السلسلة "Monday" .

```
printf( "%c\n", "Monday" );
```

د)
```
printf( ""A string in quotes"" );
```

هـ)
```
printf( "%d%d, 12, 20 );
```

و)
```
printf( "%c", "x" );
```

ز)
```
printf( "%s\n", 'Sohayb' );
```

٧ – ٢) اكتب عبارة لتنفيذ كل مما يلي :

أ) اطبع 1234 أقصى اليمين (right-justified) في مجال عرضه 10 أرقام (10 digit field) .

ب) اطبع 123.456789 بالاصطلاح / بالصيغة الأسِّيَّة (in exponential notation) مع إشارة (+ أو –) ودِقَّة تبلغ 3 أرقام (3 digits of precision) .

ج) اقرأ قيمة مضاعفة **double** (وقم بتخزينها) في المتغير **number** .

د) اطبع 100 بالصيغة الثمانية مسبوقة بالصفر 0 .

هـ) اقرأ سلسلة (string) (وقم بتخزينها) في منظومة الرموز **string** (character array) .

و) اقرأ رموزا (وقم بتخزينها) في منظومة **n** إلى أن تقابل رمزا غير رقمي (nondigit character) .

ز) استخدم المتغيرين الصحيحين **x, y** (integer variables) لتحديد (to specify) عرض المجال (field width) والدقة (precision) المُسْتَخْدَمَيْن لطباعة القيمة المضاعفة 87.4573 (value) **double** .

ح) اقرأ قيمة صيغتها 3.5% (of the form) . ثم قم بتخزين (storing) النسبة المئوية (percentage) في متغير من النوع **float** اسمه **percent** ، واحذف (eliminate) علامة النسبة المئوية % من سيل المدخلات (input stream) . لا تستخدم رمز كبت / حذف الإسناد .

ط) اطبع 3.333333 كقيمة مضاعفة طويلة (a **long double** value) مع إشارة (+ أو–) في مجال عرضه 20 رمزا ، ودقة تبلغ 3 .

٧ – ٣) اكتب عبارة **printf** أو عبارة **scanf** لكل مما يلي :

أ) اطبع العدد الصحيح 40000 دون إشارة (unsigned integer) أقصى اليسار (left justified) في مجال عرضه 15 رقما (15-digit field) ودقة 8 أرقام .

ب) اقرأ قيمة سداسية عشرية وخزِّنها في المتغير **hex** .

ج) اطبع القيمة 200 مع إشارة ودون إشارة .

د) اطبع القيمة 100 بالصيغة السداسية العشرية بحيث يسبقها **0×** .

هـ) اقرأ رموزا وقم بتخزينها في منظومة **s** ، إلى أن تقابل الحرف **p** .

و) اطبع 1.234 في مجال عرضه 9 أرقام مع أصفار سابقة / متقدمة (preceding zeros).

ز) اقرأ زمناً (a time) في الصيغة **hh : mm : ss** ، وقم بتخزين أجزاء الزمن / الوقت في المتغيرات الصحيحة **hour, minute, second** ، بحيث تتخطَّى أي نقطتين (:) (colon) في سيل المدخلات . استخدم رمز كبت / حذف الإسناد .

ح) اقرأ سلسلة صيغتها " **characters**" (of the form) من المدخلات القياسية ، وقم بتخزين السلسلة في منظومة رموز **s** . احذف (eliminate) علامتي التنصيص / الاقتباس (quotation marks) من سيل المدخلات .

ط) اقرأ زمنا صيغته **hh : mm : ss** ، وقم بتخزين أجزاء الزمن في المتغيرات الصحيحة **hour, minute, second** ، بحيث تتخطَّى أي نقطتين (:) (colon) في سيل المدخلات. لا تستخدم رمز كبت / حذف الإسناد .

٧ – ٤) وضِّح ماذا يُطبع بكل عبارة من العبارات التالية . وإذا كانت أي عبارة خاطئة (incorrect) فبين لماذا .

أ) `printf( "%-10d\n", 10000 );`

ب) `printf( "%c\n", "This is a string" );`

ج) `printf( "%*.*lf\n", 8, 3, 1024.987654 );`

د) `printf( "%#o\n%#X\n%#e\n", 17, 17, 1008.83689 );`

هـ) `printf( "% ld\n%+ld\n", 1000000, 1000000 );`

و) `printf( "%10.2E\n", 444.93738 );`

ز) `printf( "%10.2g\n", 444.93738 );`

ح) `printf( "%d\n", 10.987 );`

٧ – ٥) اكتشف الخطأ / الأخطاء في كل من قطع البرامج التالية . ووضح كيفية تصحيح أي خطأ .

أ) `printf( "%s\n", 'Happy Eid' );`

ب) `printf( "%c\n", 'Salam' );`

ج) `printf( "%c\n", "This is a string" );`

د) العبارة التالية يجب أن تطبع **"See You"** :

```
printf( ""%s"", "See Yoo" );
```

هـ)
```
char day[] = "Friday";
printf( "%s\n", day[ 3 ] );
```

و) `printf( 'Enter your name: ' );`

ز) `printf( %f, 123.456 );`

ح) العبارة التالية يجب أن تطبع الرمزين 'O', 'K' :

```
printf( "%s%s\n", 'O', 'K' );
```

ط)
```
char s[ 10 ];
scanf( "%c", s[ 7 ] );
```

٧ – ٦) اكتب برنامجا يملأ المنظومة **number** المكونة من 10 عناصر باعداد صحيحة عشوائية (random integers) من 1 إلى 1000 . ولكل قيمة (value) من هذه القيم / الأعداد يطبع البرنامج : القيمة ، والعدد الإجمالي الجاري (running total number) للرموز (characters) التي تمت طباعتها (printed) لجميع القيم حتى

لحظة الانتهاء من طباعة هذه القيمة الحالية . استخدم محدد التحويل **%n** لتحديد عدد الرموز المطبوعة لكل قيمة . وفيما يلي مثال لصيغة طباعة مخرجات البرنامج .

Value	Total characters
342	3
1000	7
963	10
6	11
etc.	

٧ – ٧) اكتب برنامجا ليختبر الفارق بين محدِّدي التحويل **%i , %d** عندما يُستخدمان في عبارات **scanf** . استخدم العبارتين

```
scanf( "%i%d", &x, &y );
printf( "%d %d\n", x, y );
```

لإدخال وطباعة القيم . اختبر البرنامج بمجموعات البياناتالمُدْخَلة (sets of input data) التالية :

10	10
-10	-10
010	010
0x10	0x10

٧ – ٨) اكتب برنامجا يختبر نتائج طباعة القيمة الصحيحة 12345 والقيمة ذات النقطة العائمة 1.2345 في مجالات مختلفة السعات . ماذا يحدث عندما تُطبع القيم في مجالات عدد أرقام سعاتها أقل من عدد أرقام القيم المطبوعة (المطلوبة طباعتها) ؟

٧ – ٩) اكتب برنامجا لطباعة القيمة 100.453627 مقربةً إلى أقرب رقم (rounded to the nearest digit) ، وإلى أقرب رقم عشري (nearest tenth) ، وإلى أقرب رقم مئوي (hundredth) ، وإلى قرب رقم ألفى (thousandth) ، وإلى أقرب رقم من عشرة آلاف (ten thousandth) .

٧ – ١٠) اكتب برنامجا لإدخال سلسلة رموز من لوحة المفاتيح ، وتعيين طول السلسلة . واطبع السلسلة باستخدام عرض مجال (field width) يساوي ضعف طول السلسلة .

٧ – ١١) اكتب برنامجا يحوّل (converts) درجات الحرارة الفهرنهيتية الصحيحة (integer Fahrenheit temperatures) التي تتراوح من 0 إلى 212 درجة (degrees) إلى درجات حرارة مئوية ذوات نقطة عائمة (floating-point Celsius temperatures) بدقة تبلغ 3 أرقام (3 digits of precision) . استخدم الصيغة

celsius = 5.0 / 9.0 * (fahrenheit – 32);

لإجراء الحسابات . واطبع المخرجات أقصى يمين (right-justified) عمودين يتكون كل منهما من 10 رموز . واطبعدرجات الحرارة المئوية مسبوقة (preceded by) بإشارة لكل من القيم الموجبة والقيم السالبة .

٧ – ١٢) اكتب برنامجا يختبر جميع متتابعات الهروب المذكورة في "جدول متتابعات الهروب" المعطَى في هذا الفصل . وبالنسبة لمتتابعات الهروب التي تحرِّك المؤشر (move the cursor) : اطبع رمزا قبل وبعد طباعة متتابعة الهروب ، حتى يتضح لنا أين تحرك المؤشر .

٧ – ١٣) اكتب برنامجا يحدِّد ما إذا كان ممكناً طباعة ? كجزء من (part of) سلسلة التحكم في الصيغة في عبارة **printf** كرمز حرفي (as a literal character) بدلا من استخدام متتابعة الهروب ?\ .

٧ – ١٤) اكتب برنامجا لإدخال القيمة 437 باستخدام كل من محددات تحويل الأعداد الصحيحة مع **scanf** . اطبع كل قيمة مُدْخَلة باستخدام جميع محددات تحويل الأعداد الصحيحة .

٧ – ١٥) اكتب برنامجا يستخدم كلا من محددات التحويل **e**, **f**, **g** لإدخال القيمة 1.2345 . اطبع قيم جميع المتغيرات لإثبات أنه يمكن استخدام أي من محددات التحويل هذه لإدخال تلك القيمة نفسها .

٧ – ١٦) في بعض لغات البرمجة يتم إدخال سلسلة الرموز محاطة إما بحاصرتين مفردتين (single quotes) أو بحاصرتين مزدوجتين (double quotes) . اكتب برنامجا يقرأ السلاسل الثلاث : 'Omar' , "Omar", Omar . هل لغة C تتجاهل (ignores) الحاصرتين المفردتين ، والحاصرتين المزدوجتين أم تقرأهما كجزء من السلسلة ؟

٧ – ١٧) اكتب برنامجا يحدد ما إذا كان ممكنا طباعة ? باعتبار أنها الثابت الرمزي (character constant) بدلا من متتابعة هروب الثابت الرمزي (character constant) '?'

'\?' (escape sequence) باستخدام محول التحويل **%c** في سلسلة التحكم في لاصيغة في عبارة **printf** .

٧ – ١٨) اكتب برنامجا يستخدم محدد التحويل **g** لطباعة القيمة 9876.12345 . اطبع القيمة عدة مرات بدقة تتراوح من 1 إلى 9 .

تمرينات عامة

١- الرسم المبين بالصفحة التالية يمثل خريطة سير عمليات برنامج يوجد عوامل (Factors) أي عدد صحيح موجب أقل من ١٠٠٠ ، ثم يوجد مجموع عوامله (مثلا عوامل العدد ١٢ هي : ١ ، ٢ ، ٣ ، ٤ ، ٦ ، ١٢ ، ومجموع هذه العوامل يساوي ٢٨) .
والمطلوب : ترجمة هذه الخريطة إلى برنامج .

٢- يبلغ نصاب الذهب والعملات الذهبية ما يعادل ٨٥ جراما من الذهب 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 :

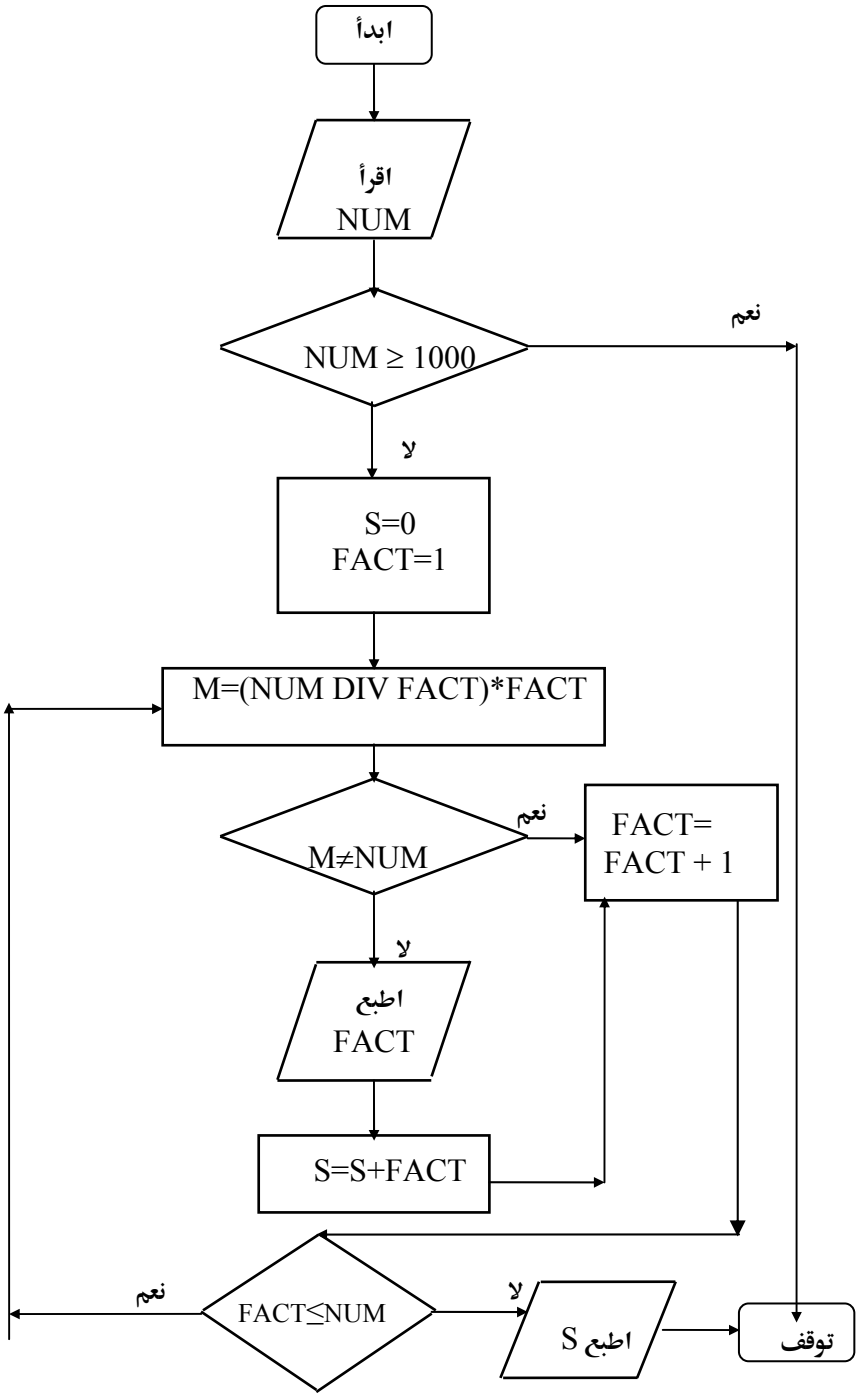
(أ) يقرأ بيانات كل شخص على سطر مستقل وهي :
NAME : اسم الشخص .
GOLD : كمية الذهب بالجرام التي حال عليها الحول .
CARATS : عيار الذهب .
MONEY : قيمة النقود المدخرة بالدينار التي حال عليها الحول .

(ب) يحسب نصاب النقود بالدينار **NB** .

(ج) يحسب زكاة النقود بالدينار **ZM** ، وهي تساوي ٢,٥٪ من المبلغ المدخر **MONEY** إذا بلغ النصاب .

(د) يحسب كمية الذهب الخالص بالجرام **PUREGD** وهي تساوي : $\frac{\text{عيار الذهب}}{٢٤} \times$ كمية الذهب بالجرام

(هـ) يحسب زكاة الذهب **ZG** بالجرام ، وهي تساوي ٢,٥٪ من كمية الذهب الخالص بالجرام إذا بلغ النصاب .


ابدأ
اقرأ
NUM
NUM ≥ 1000
نعم
لا
S=0
FACT=1
M=(NUM DIV FACT)*FACT
M≠NUM
نعم
FACT=
FACT + 1
لا
اطبع
FACT
S=S+FACT
FACT≤NUM
نعم
لا
اطبع S
توقف

٣- الرسم المبين بالصفحة التالية يوضح خريطة سير عمليات إيجاد القاسم المشترك الأعظم **GCD** لعددين صحيحين **N2 , N1** ، وهو أكبر عدد صحيح يقبل كل من العددين **N2 , N1** القسمة عليه بدون باق (مثلا القاسم المشترك الأعظم للعددين ٢٤ ، ٤٠ هو ٨) .
اكتب دالة لإيجاد القاسم المشترك الأعظم لعددين صحيحين **N2 , N1** .

٤- اكتب برنامجا لقراءة قيمة عدد صحيح **n** وقيمتي متغيرين **x , a** ، 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 :

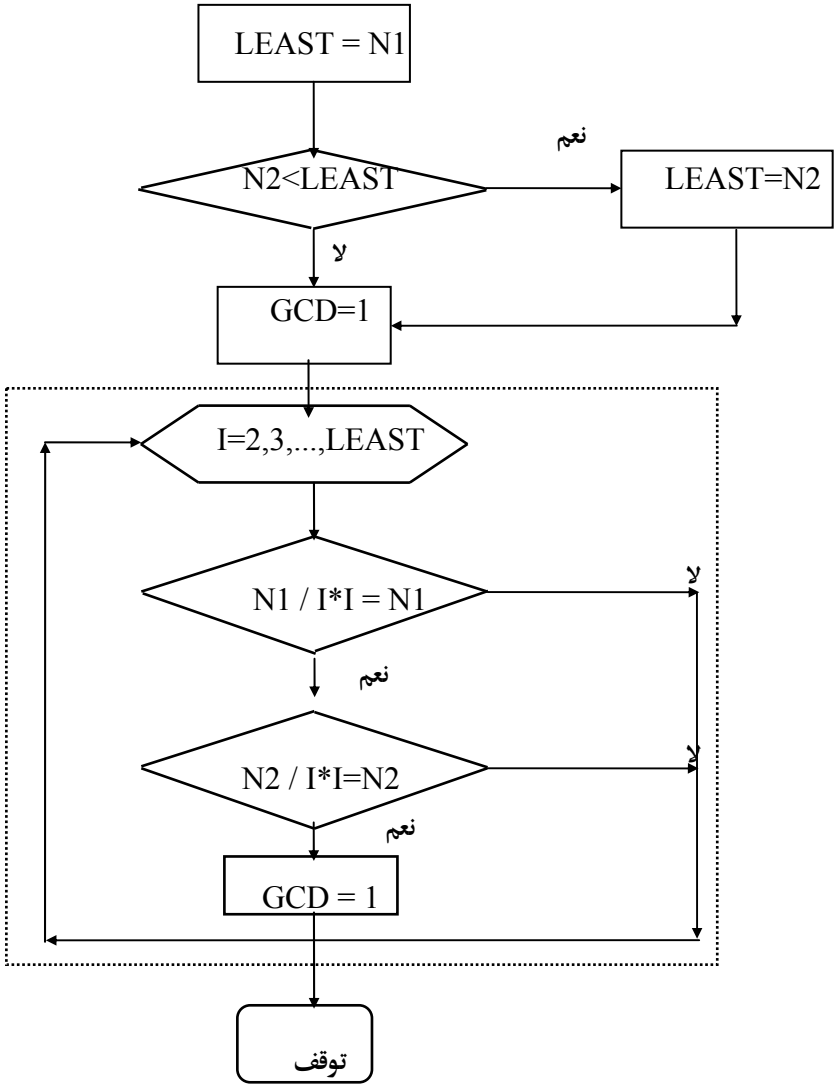
$$x = (3 - y) / 2$$
$$y = (x - 2) / 3$$

(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 :

$$2\ XNEW + YNEW \cong 3$$

$$XNEW - 3\ YNEW \cong 2$$

وبصورة أخرى : إلى أن يتحقق الشرطان :

$$|2\ XNEW + \quad YNEW - 3| < \varepsilon$$
$$|\ \ XNEW - 3\ YNEW - 2| < \varepsilon$$

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 و**F** 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 ٤٠٠ دينارا .

(ب) عدد المدرسين والمدرسات (**N2**) الذين يحصلون على راتب شهري أعلى من ٧٠٠ 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 **float** . المطلوب كتابة برنامج :

(ه) يقرأ قيم عناصر المنظومة **A** .

(و) 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 :

$$2i - j < 3$$
$$i + 3j \geq 1$$
$$-6 \leq i \leq 6$$
$$-10 \leq j \leq 10$$

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 و**F** للأنثى)

عمر الشخص (**AGE**).

رقم يشير إلى حالته الزوجية (١ للأعزب و ٢ للمتزوج)

وعدد هذه السجلات غير معلوم ، وقد استخدم الرقم ٩ في الموضع المخصص للحالة الزوجية ليشير إلى نهاية الملف .

ارسم خريطة سير عمليات ، واكتب برنامجا لحساب وطباعة ما يلي :

(أ) النسبة المئوية للذكور (**PM**) ، والنسبة المئوية للإناث (**PF**).

(ب) العدد الإجمالي للذكور فوق سن السادسة عشرة (**N**). والنسبة المئوية – من هذا العدد **N** – للذكور الأيامى (*) فوق سن الثامنة والعشرين (**PUM**) .

(ج) العدد الإجمالي للنساء فوق سن السادسة عشرة (**M**) ، والنسبة المئوية – من هذا العدد **M** – للنساء الأيامى فوق سن الخامسة 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** .

(هـ) 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**.

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) يعرِّف 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** .

(*) الأيامى : جمع أيِّم ، وهو الرجل الذي لا زوجة له، أو المرأة التي لا زوج لها

١٣- تتكون المنظومة **A** من عدد – لا يزيد عن مائة – من الأعداد الصحيحة غير الصفرية . ويُستخدم العدد صفر كقيمة تشير إلى انتهاء عناصر المنظومة.
اكتب برنامجا لقراءة عناصر هذه المنظومة وإيجاد :
(أ) عدد العناصر **N** في المنظومة .
(ب) عـدد العناصـر الزوجيـة **NE** ، وعـدد العناصـر الفرديـة **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 دالة تحذف كل العناصر الصفرية من منظومة **X** تحتوي على **N** عدد float، ويحتفظ بالعناصر الأخرى من **X** في منظومة اسمها **Y**.
(ب) اكتب برنامجا رئيسيا :
١- يقرأ عناصر منظومة **A** فيها **L** عدد float .
ويقرأ عناصر منظومة **Z** فيها **M** عدد float .
٢- يستخدم الدالة – 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 :

$$\text{AREA} = \frac{1}{2}\left[y_1 x_N - x_1 y_N\right] + \sum_{i=2}^{N}\left(y_i x_{i-1} - x_i y_{i-1}\right)$$

(ب) **إحداثيي مركز ثقله** (Center of gravity) **، ويعطيان بالعلاقتين :**

$$XG = \left(\sum_{i=1}^{N} x_i\right) / N \quad , \qquad YG = \left(\sum_{i=1}^{N} y_i\right) / N$$

١٧- **اكتب برنامجا يقرأ متتابعة من خمس سلاسل رموز يُدخلها المستخدِم . ويقوم البرنامج بتخزين هذه السلاسل في منظومة سلاسل رموز** (array of strings) **. ثم يطبع البرنامج تلك السلاسل فقط التي تبدأ بالحرف "b" .**

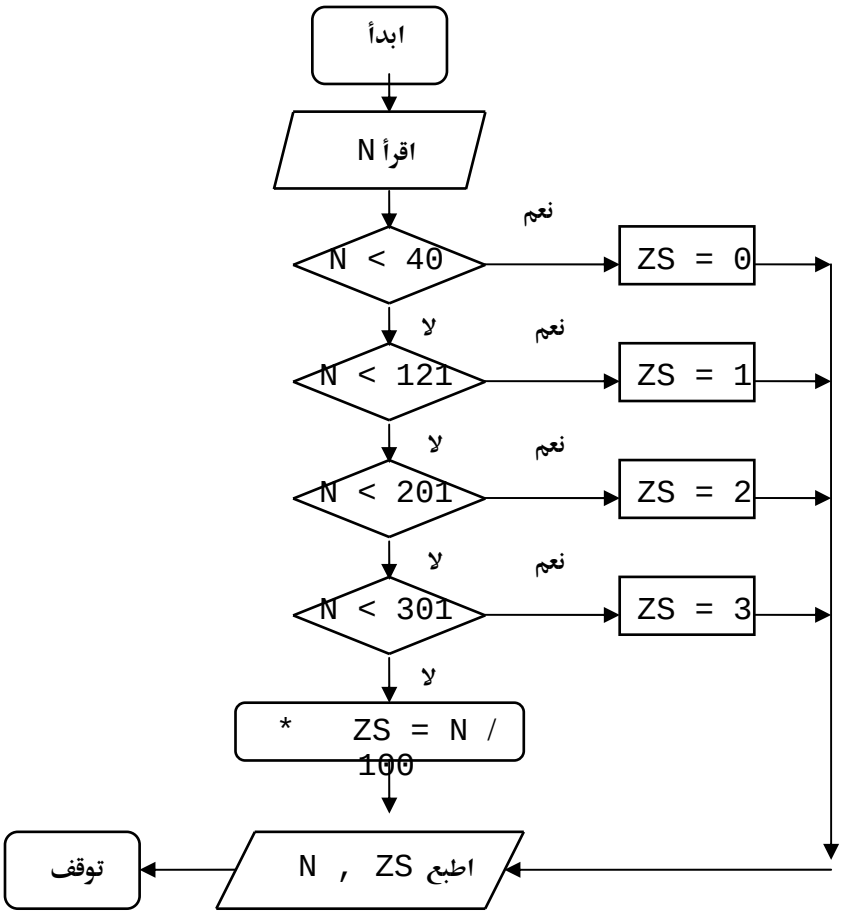
مثال لمدخلات ومخرجات البرنامج :

Sample Run :

```
Enter a string: the
Enter a string: believers
Enter a string: are
Enter a string: but
Enter a string: brothers

The strings starting with 'b' are:
believers
but
brothers
```

أجوبة تمرينات رقم ١

١-١

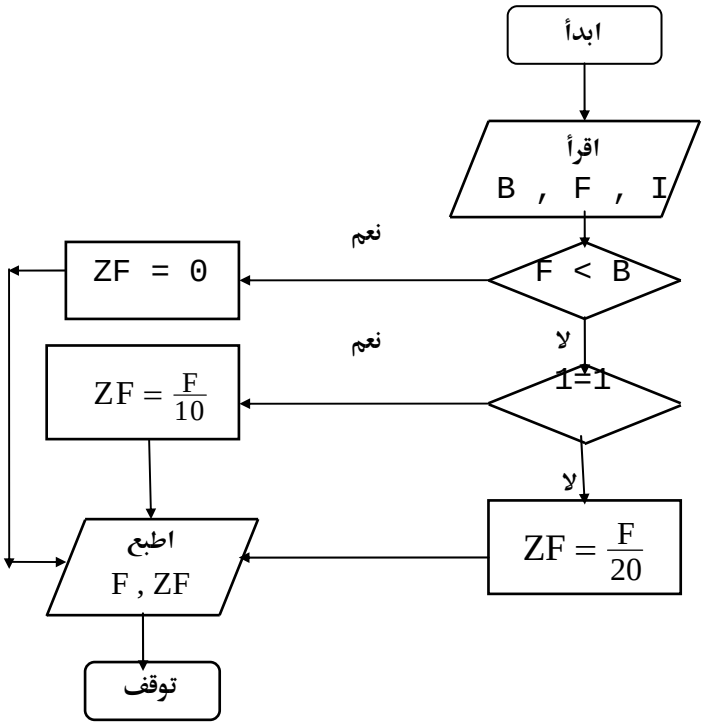


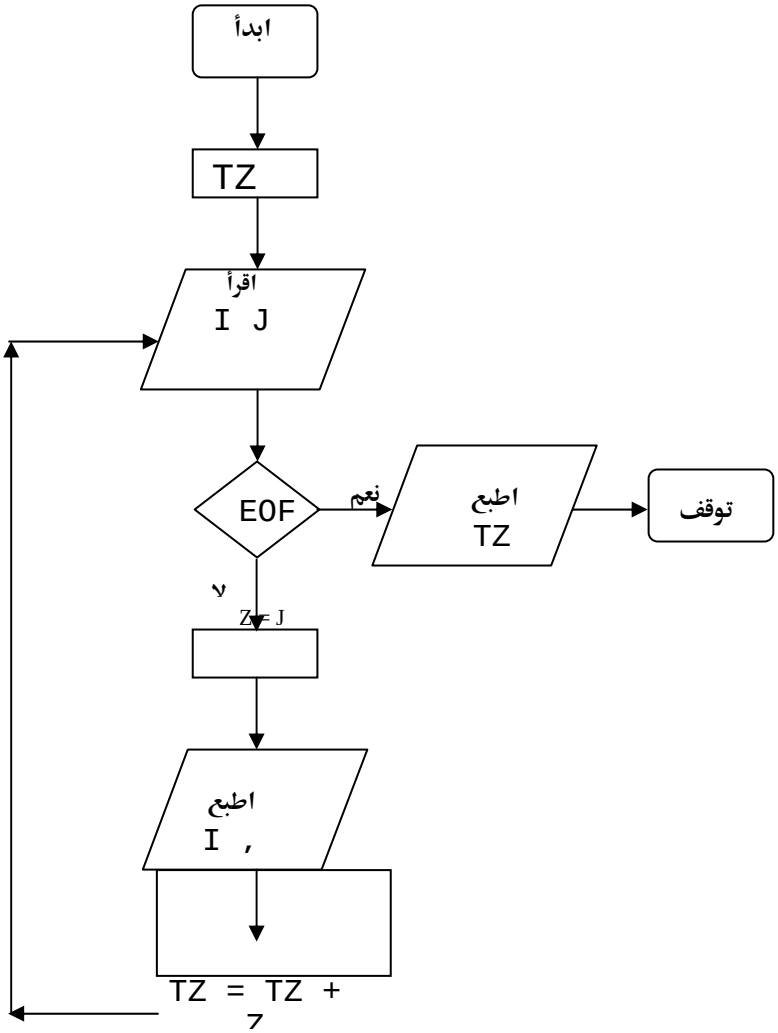
* ملاحظة : يمكن التعبير عن العبارة : ZS تساوي في كل مائة شاة بالعلاقة :

ZS = N / 100 والتي تعطي العدد الصحيح فقط في خارج قسمة $\frac{N}{100}$ مع حذف الكسور العشرية ، وذلك لأن هذه العلاقة تعطي النتيجة التالية :

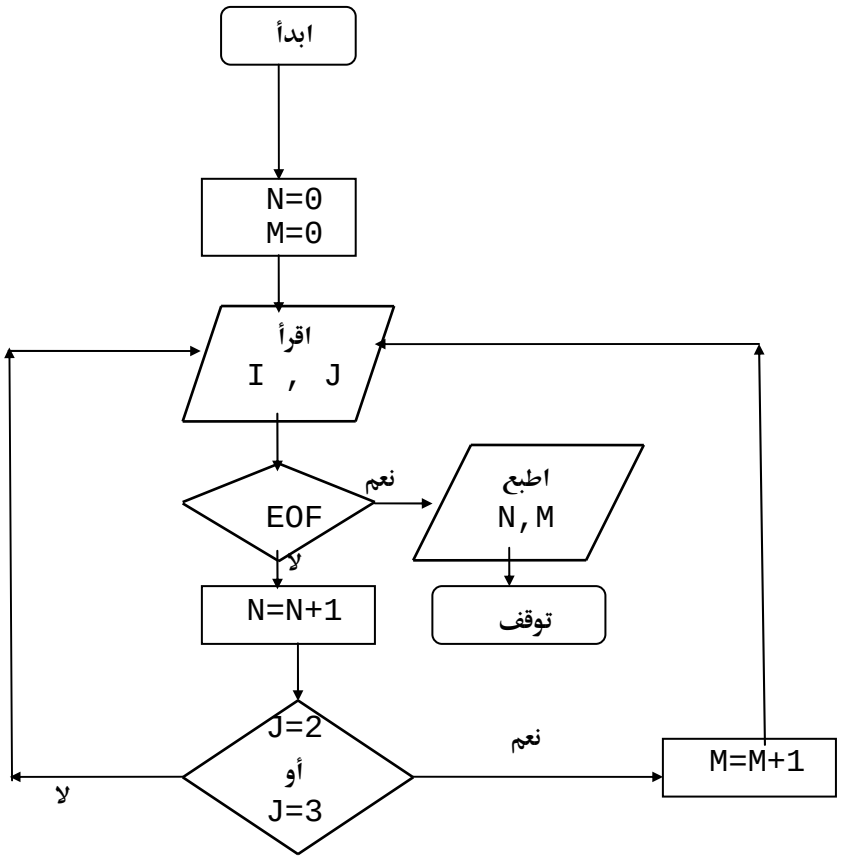
N	٣٠١-٣٩٩	٤٠٠-٤٩٩	٥٠٠-٥٩٩	...
ZS	٣	٤	٥	...

٢-١

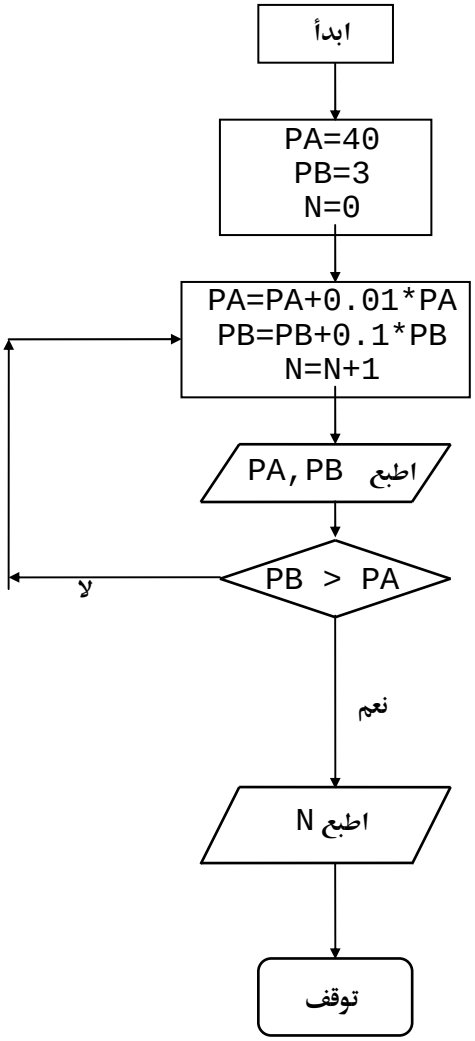



ابدأ
TZ
اقرأ
I J
EOF
نعم
اطبع
TZ
توقف
لا
Z = J
اطبع
I ,
TZ = TZ +
Z

٤-١



١-٥

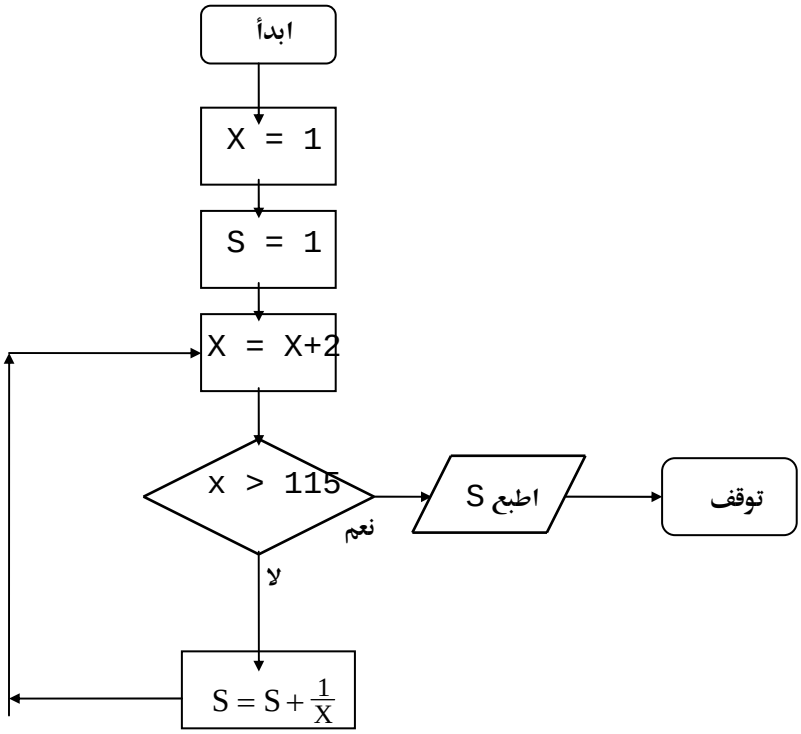


ملاحظة : يمكن أيضا طباعة عدد السنوات N – كل سنة – مع PB , PA .

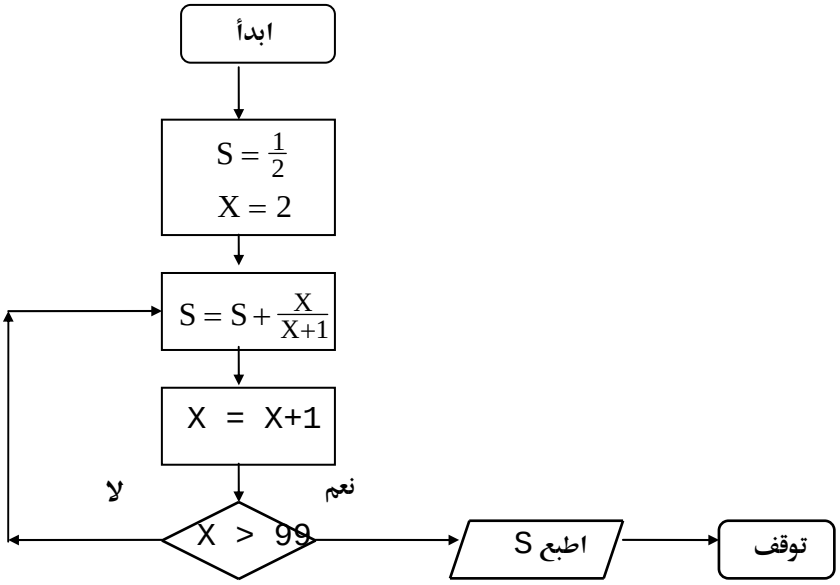
١-٦

iv) 5 iii) 10 ii) 8 i) 5

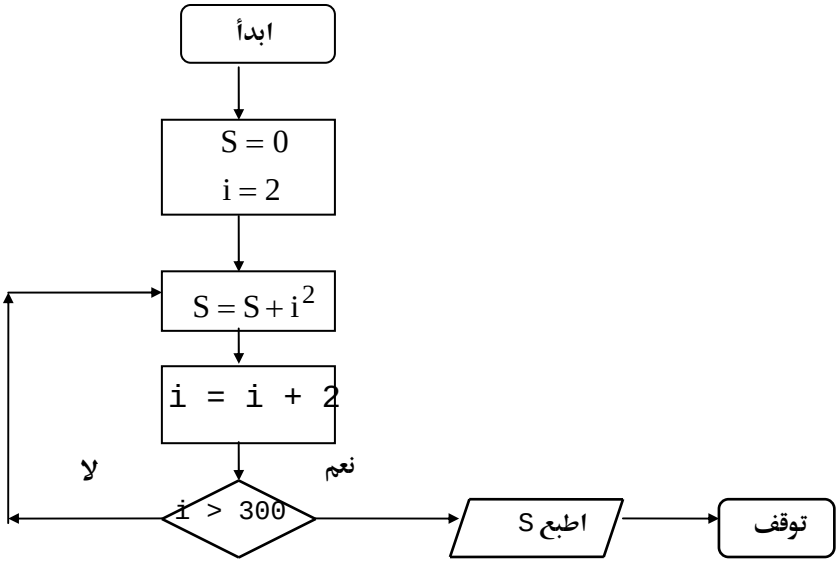
١-٧ i)

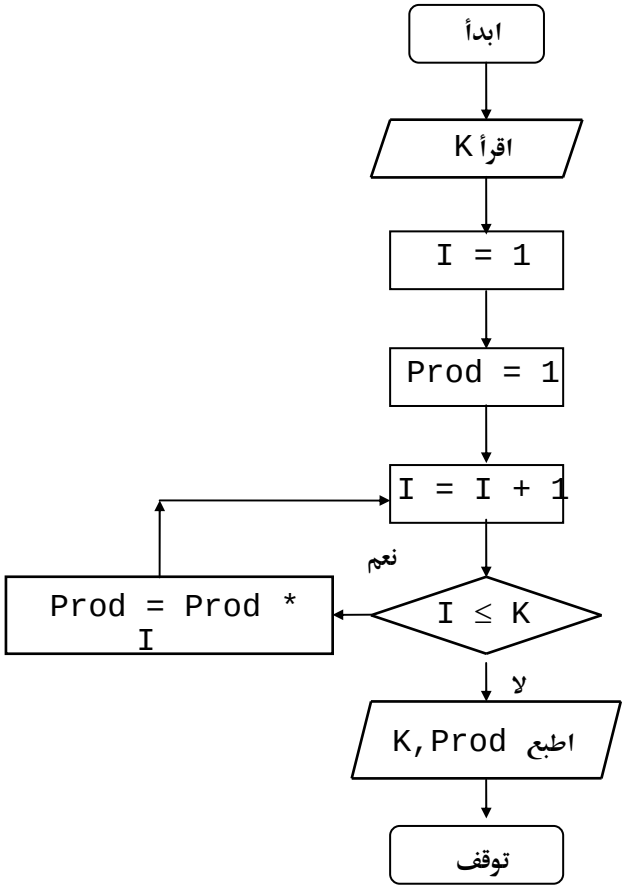


(ii ٧-١

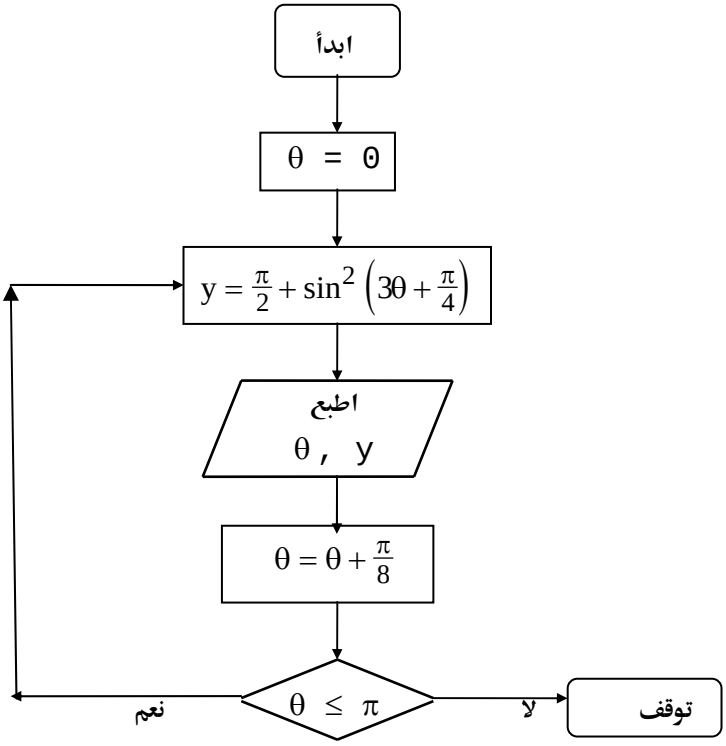

ابدأ
$S = \frac{1}{2}$
$X = 2$
$S = S + \frac{X}{X+1}$
X = X+1
لا
نعم
X > 99
اطبع S
توقف

(iii ٧-١

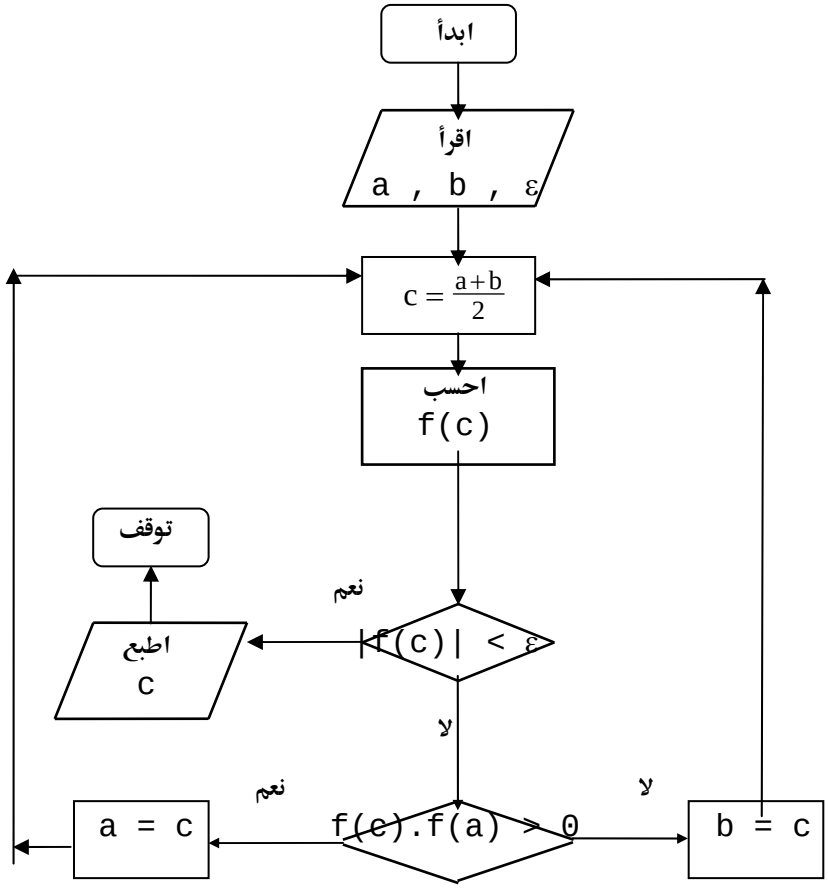

ابدأ
$S = 0$
$i = 2$
$S = S + i^2$
i = i + 2
لا
نعم
i > 300
اطبع S
توقف

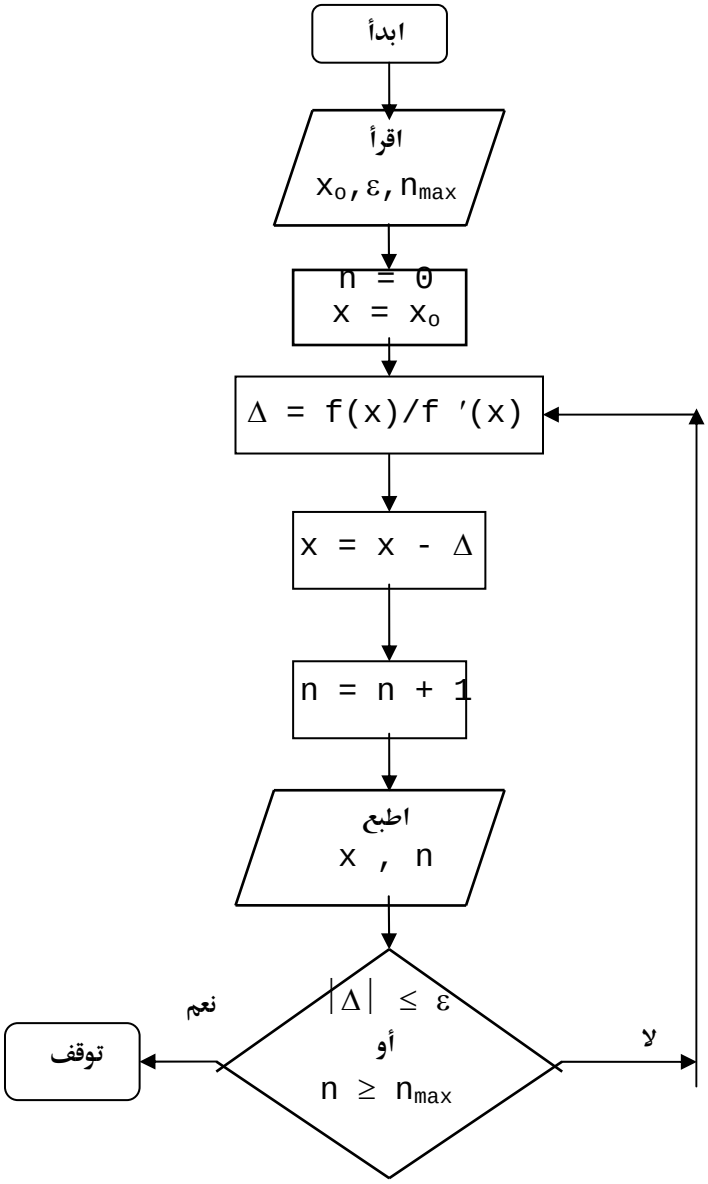

ابدأ
اقرأ K
I = 1
Prod = 1
I = I + 1
نعم
I ≤ K
Prod = Prod * I
لا
اطبع K,Prod
توقف

٩-١

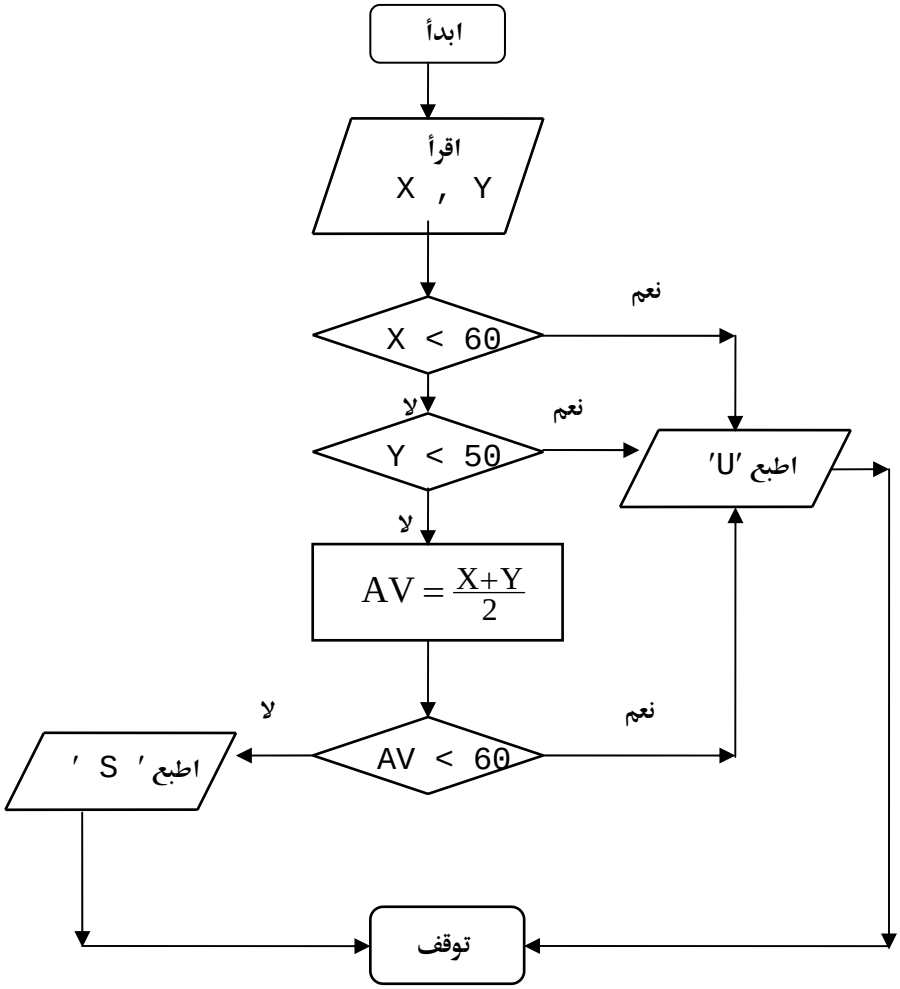

ابدأ
θ = 0
$y = \frac{\pi}{2} + \sin^2\left(3\theta + \frac{\pi}{4}\right)$
اطبع
θ , y
$\theta = \theta + \frac{\pi}{8}$
$\theta \leq \pi$
نعم
لا
توقف

١٠-١

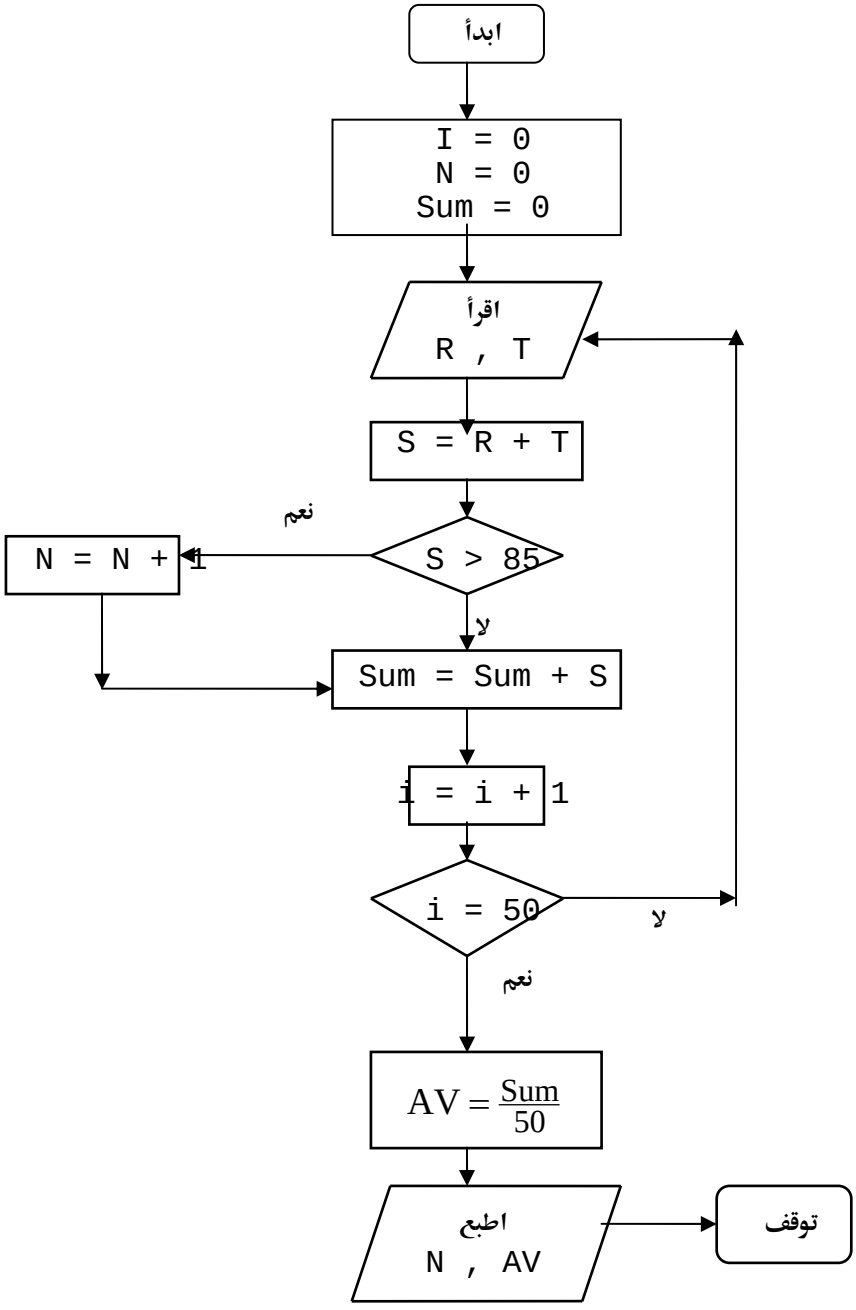

ابدأ
اقرأ
a , b , ε
$c = \frac{a+b}{2}$
احسب
f(c)
توقف
نعم
|f(c)| < ε
اطبع
c
لا
نعم
f(c).f(a) > 0
لا
a = c
b = c


ابدأ
اقرأ
$x_o, \varepsilon, n_{max}$
n = 0
x = x_o
Δ = f(x)/f ′(x)
x = x - Δ
n = n + 1
اطبع
x , n
$|\Delta| \leq \varepsilon$
أو
$n \geq n_{max}$
نعم
لا
توقف

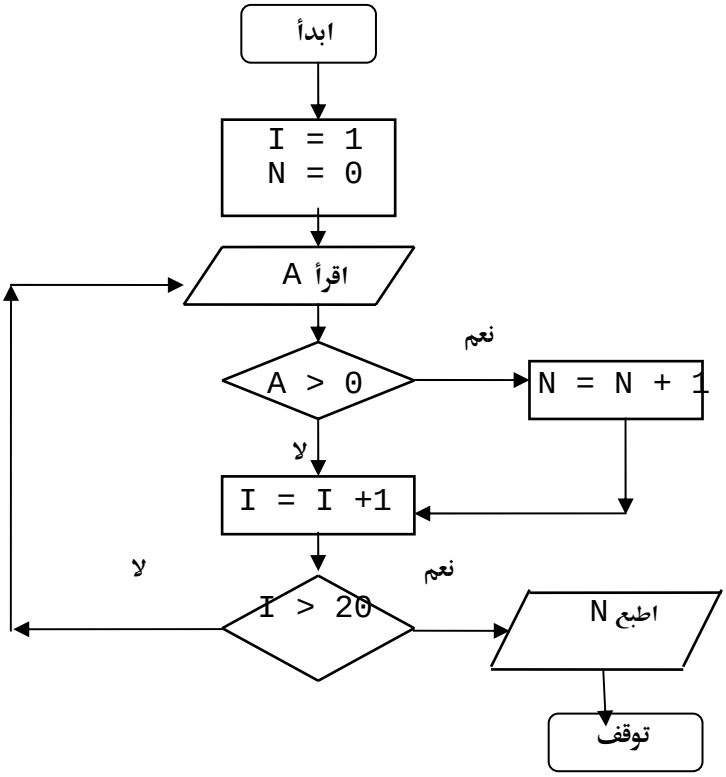
١٢-١


ابدأ
اقرأ
X , Y
X < 60
نعم
لا
Y < 50
نعم
اطبع 'U'
لا
AV = (X+Y)/2
AV < 60
لا
نعم
اطبع ' S '
توقف

١٣-١

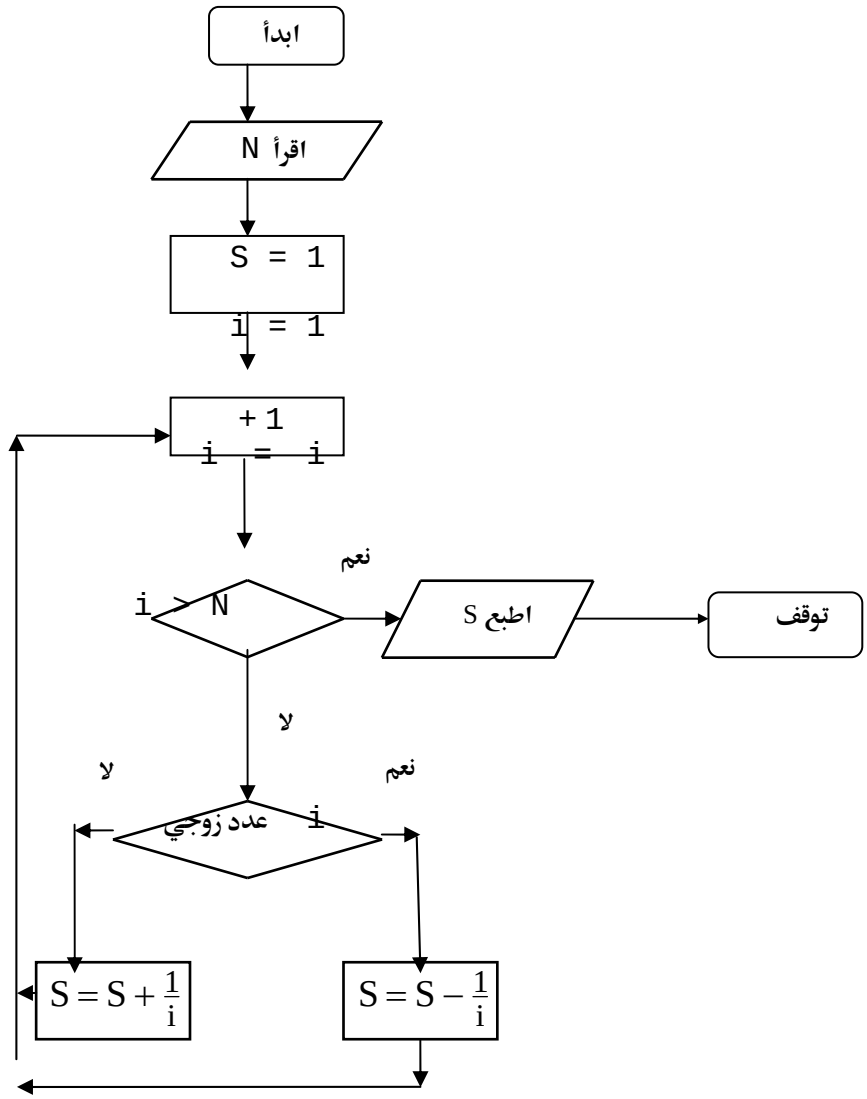

ابدأ
I = 0
N = 0
Sum = 0
اقرأ
R , T
S = R + T
نعم
N = N + 1
S > 85
لا
Sum = Sum + S
i = i + 1
i = 50
لا
نعم
$AV = \frac{Sum}{50}$
اطبع
N , AV
توقف

١٤-١

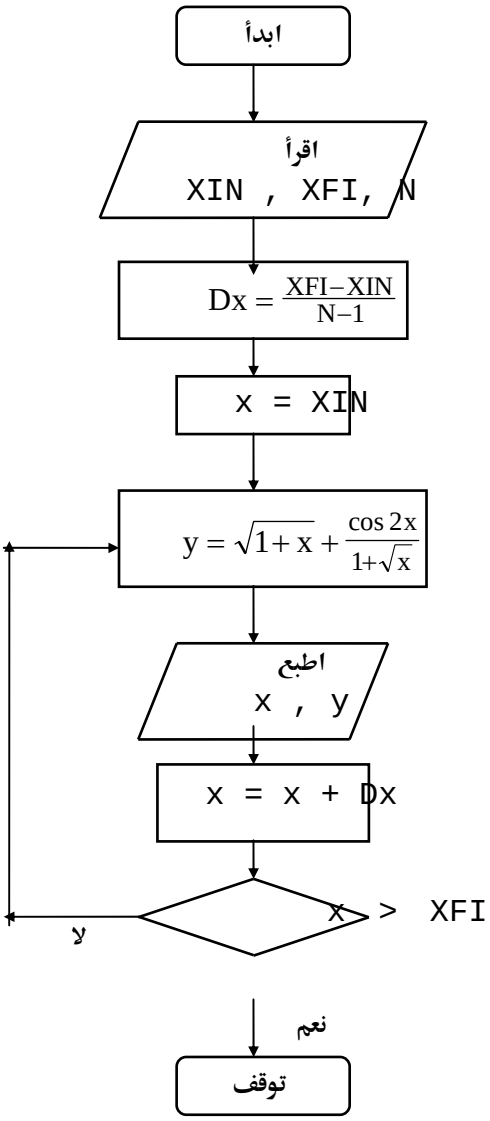


الخوارزمية تحسب عدد العناصر الموجبة في مجموعة مكونة من عشرين عددا .

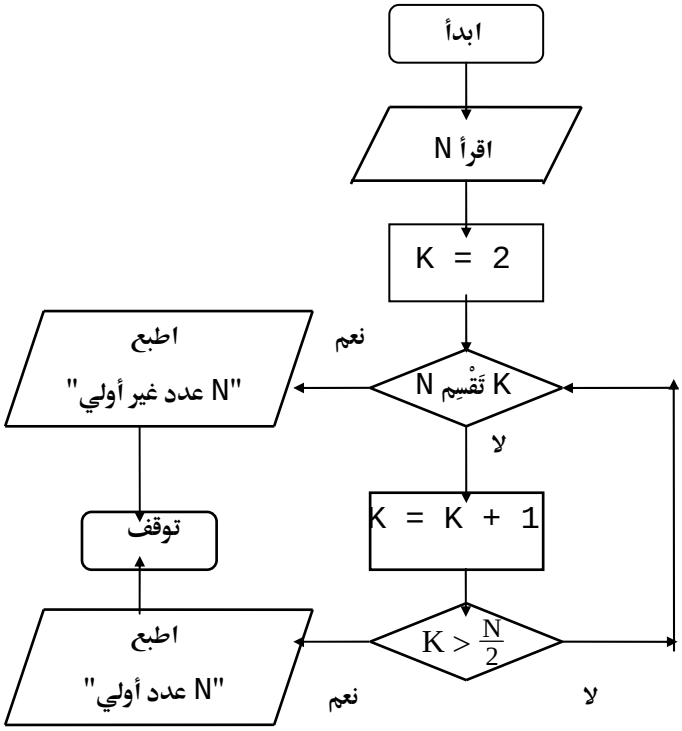
١-١٥


ابدأ
اقرأ N
S = 1
i = 1
i = i + 1
نعم
i > N
اطبع S
توقف
لا
لا
نعم
i عدد زوجي
$S = S + \frac{1}{i}$
$S = S - \frac{1}{i}$

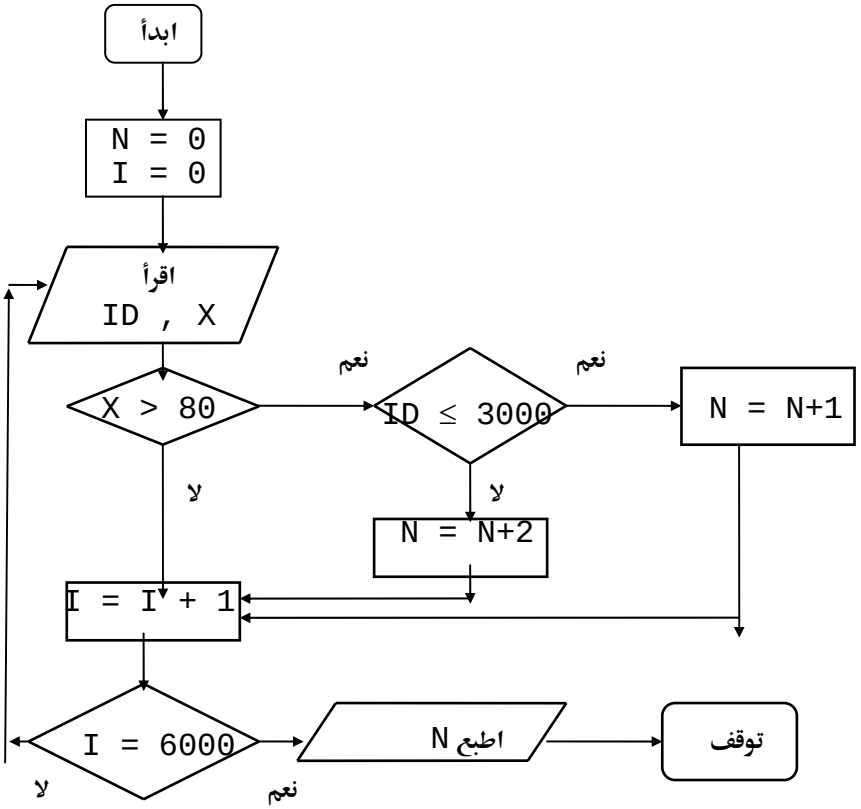
١٦-١


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

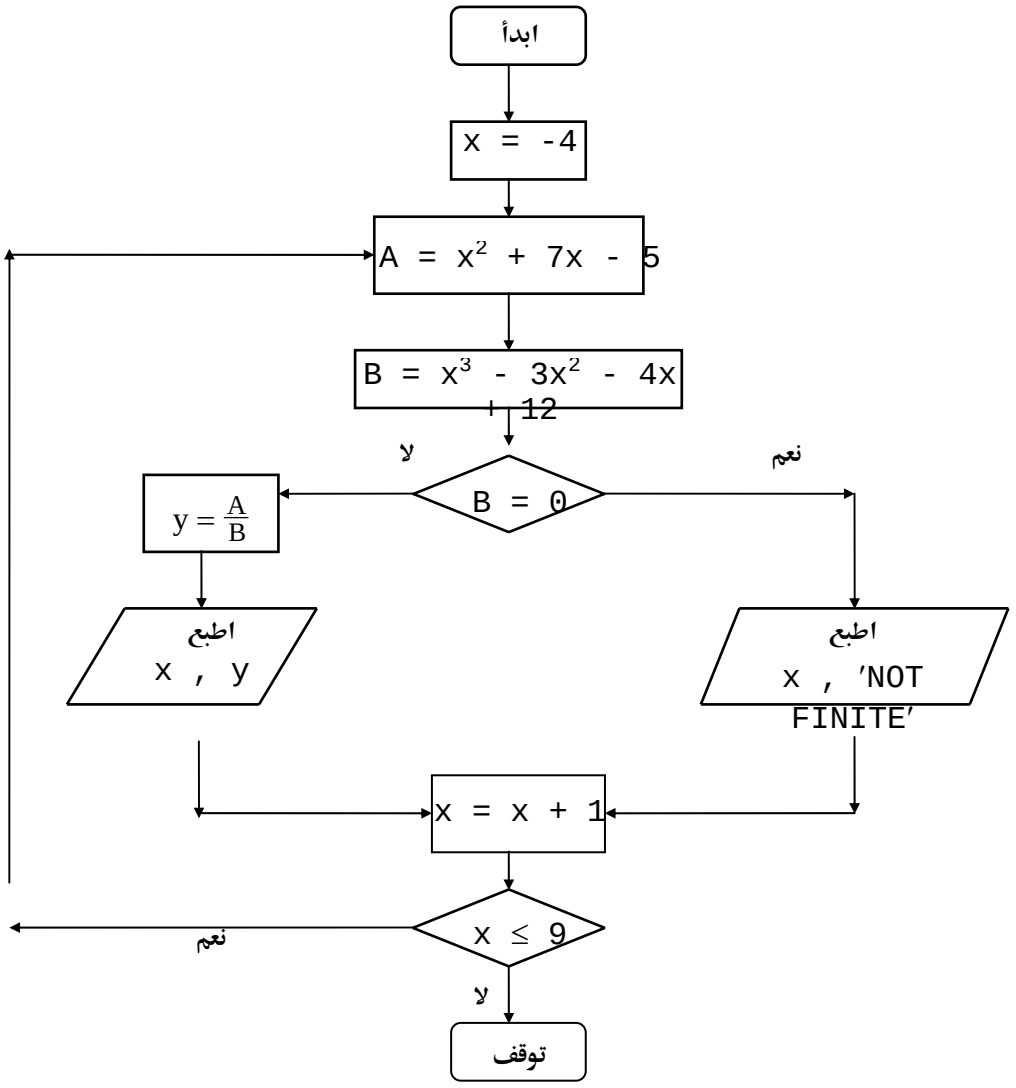
١-١٧


ابدأ
اقرأ N
K = 2
K تَقْسِم N
نعم
اطبع
"N عدد غير أولي"
لا
K = K + 1
$K > \frac{N}{2}$
نعم
لا
اطبع
"N عدد أولي"
توقف

١-١٨


ابدأ
N = 0
I = 0
اقرأ
ID , X
X > 80
نعم
لا
ID ≤ 3000
نعم
لا
N = N+1
N = N+2
I = I + 1
I = 6000
لا
نعم
اطبع N
توقف

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

٢٠-١

١- اقرأ قيمة A

٢- اجعل S = 1

٣- اجعل n = 0

٤- n = n + 1

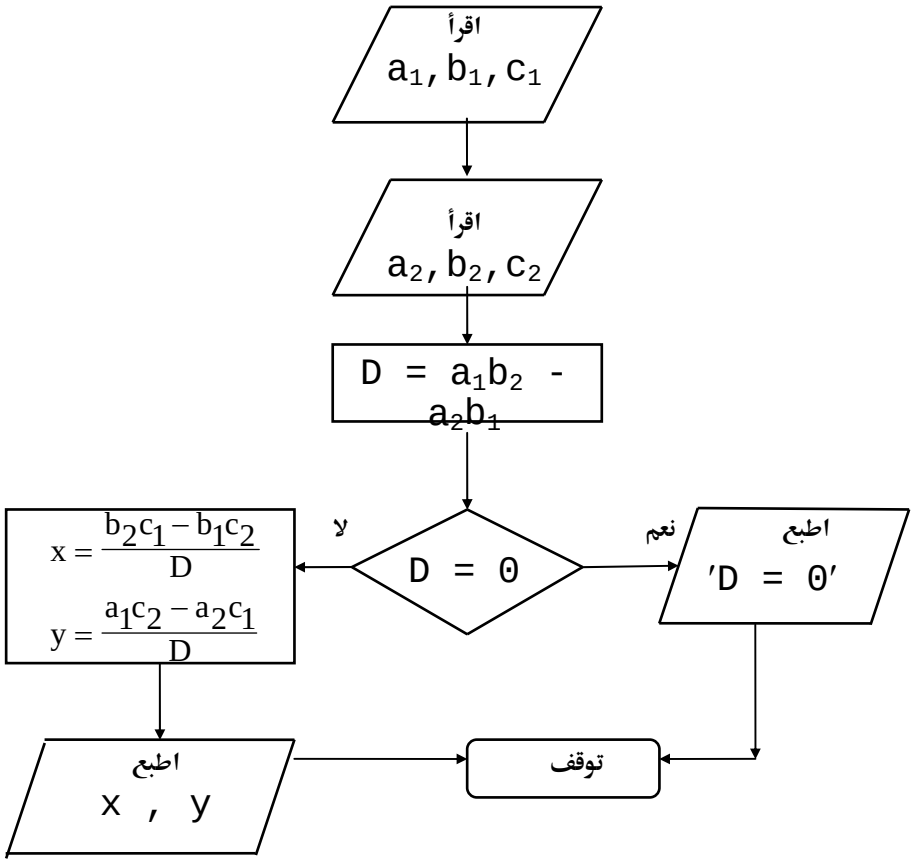
٥- S = S + 1 + n A

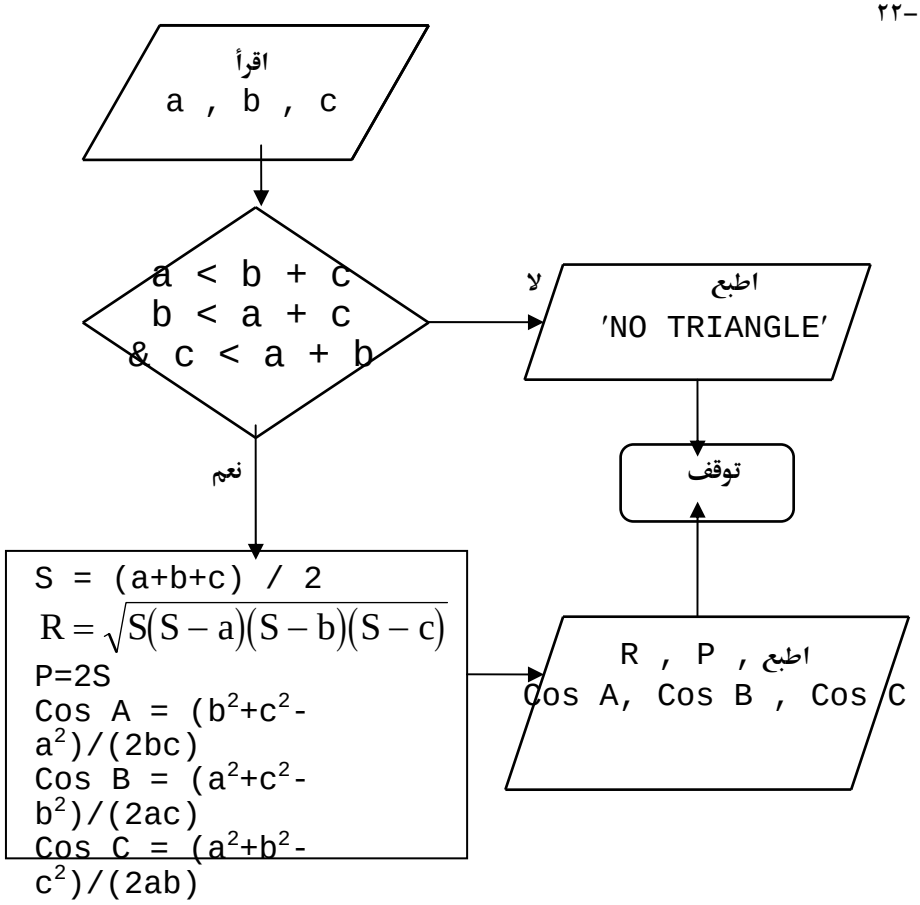
٦- إذا كانت n ≤ 48 اذهب إلى ٤

٧- اطبع S

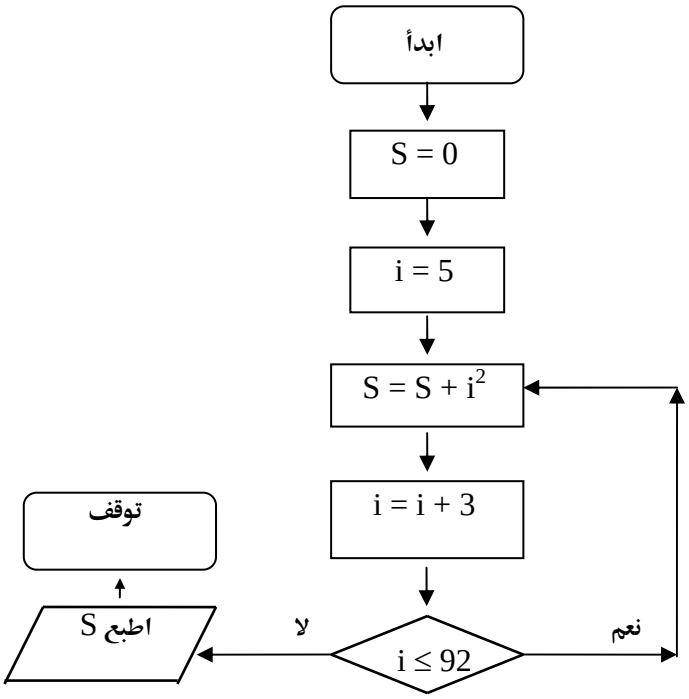
٨- توقف

٢١-١

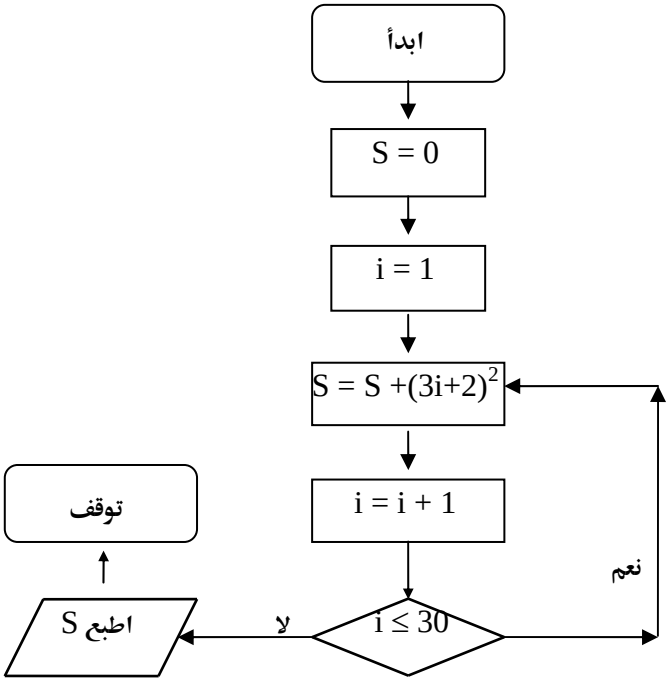


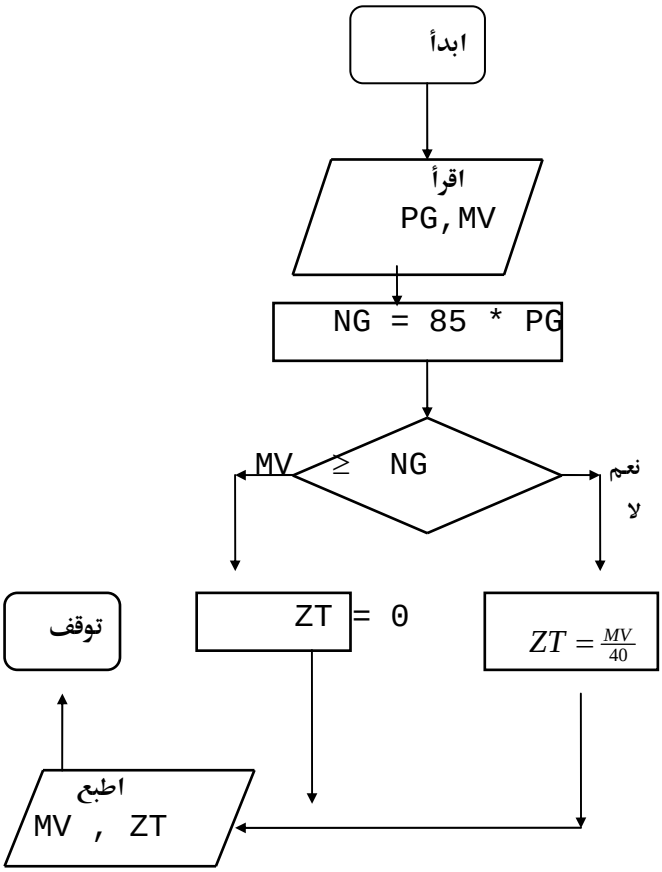

اقرأ
a , b , c
a < b + c
b < a + c
& c < a + b
لا
اطبع
'NO TRIANGLE'
توقف
نعم
S = (a+b+c) / 2
$R = \sqrt{S(S-a)(S-b)(S-c)}$
P=2S
Cos A = (b²+c²-a²)/(2bc)
Cos B = (a²+c²-b²)/(2ac)
Cos C = (a²+b²-c²)/(2ab)
اطبع , P , R
Cos A, Cos B , Cos C

١-٢٣

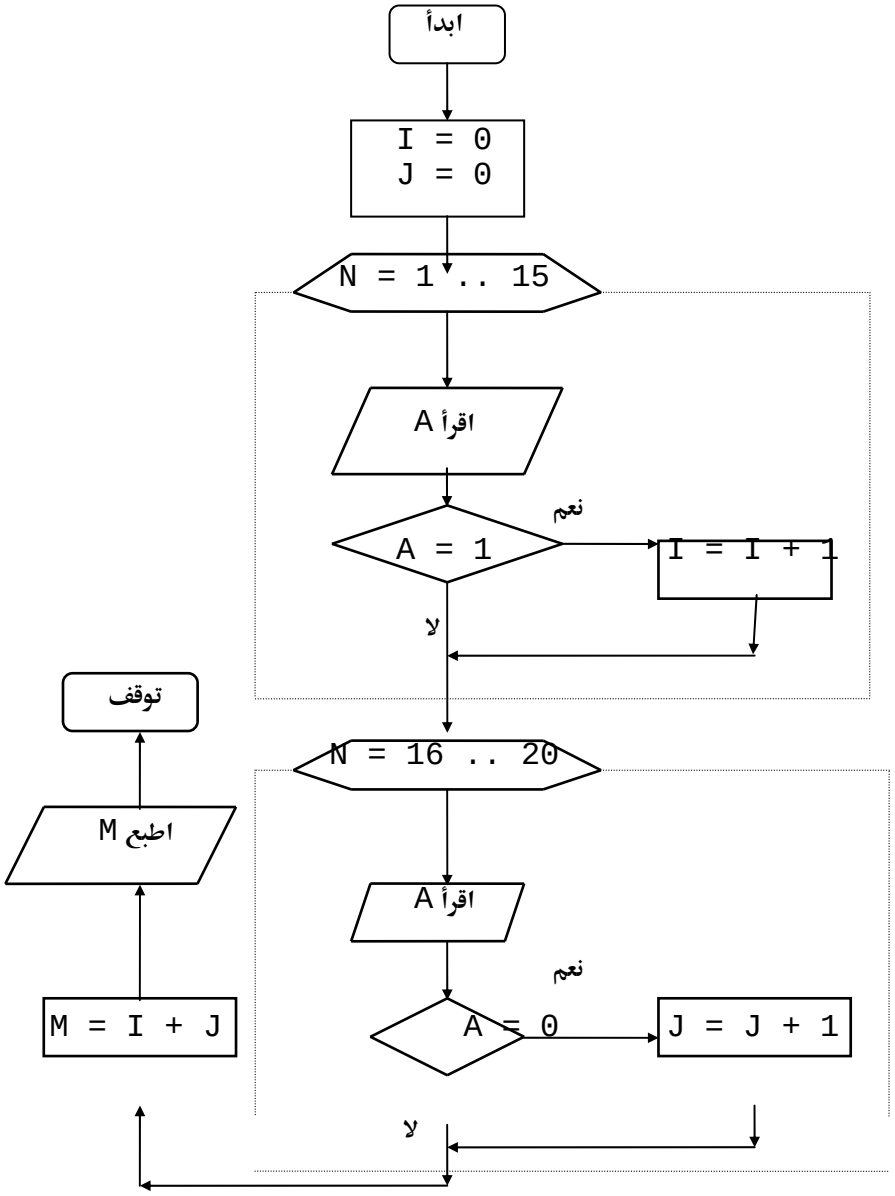

ابدأ
S = 0
i = 5
S = S + i²
i = i + 3
i ≤ 92
نعم
لا
اطبع S
توقف

حل آخر


ابدأ
S = 0
i = 1
S = S +(3i+2)²
i = i + 1
i ≤ 30
نعم
لا
اطبع S
توقف


ابدأ
اقرأ
PG,MV
NG = 85 * PG
MV ≥ NG
نعم
لا
توقف
ZT = 0
$ZT = \frac{MV}{40}$
اطبع
MV , ZT

١-٢٥


ابدأ
I = 0
J = 0
N = 1 .. 15
اقرأ A
A = 1
نعم
I = I + 1
لا
N = 16 .. 20
اقرأ A
A = 0
نعم
J = J + 1
لا
M = I + J
اطبع M
توقف

أجوبة تمرينات رقم ٢

٢-١- أ) F . الدالة **printf** تبدأ الطباعة دائما من الموضع الذي يوجد عنده مؤشر الشاشة.

ب) F . التعليقات لا تؤدي إلى عمل أي شئ عند تنفيذ البرنامج ، وإنما تُستخدم لتوثيق البرامج وتوضيحها وإمكانية قراءتها بسهولة .

ج) T د) T هـ) T

و) F . المتغيران مختلفان لأن لغة C تميِّز بين الحرف الكبير والحرف الصغير .

ز) F . التعريفات يجب أن تظهر بعد القوس الأيسر لجسم الدالة وقبل أي عبارات تنفيذية .

ح) T

ط) خ . عبارة **printf** واحدة تحتوي على عدة متتابعات هروب \n يمكن أن تطبع عدة سطور .

٢-٢- أ)
```
int c,  thisVarible, q76354,  number;
```

ب)
```
printf (  "Enter an integer:  ");
```

ج)
```
scanf(  "%d" , &a );
```

د)
```
if ( number != 7 )
```

هـ)
```
   printf( "The variable number is not equal to 7 .\n" );
   printf( "This is a C program.\n" );
```

و)
```
printf( "This is a C\nprogram.\n" );
```

ز)
```
printf( "This\nis\na\nC\nprogram.\n" );
```

ح)
```
printf( "This\tis\ta\tC\tprogram.\n" );
```

٢-٣- أ)
```
/* Calculate the product of three integers */
```

ب)
```
int  x, y, z, result;
```

ج)
```
printf (  "Enter three integers:  ");
```

د)
```
scanf(  "%d%d%d" , &x, &y, &z  );
```

هـ)
```
result  =  x  *  y  *  z;
```

و)
```
printf( "The product is %d\n", result  );
```

٢-٤)

```
/* Calculate the product of three integers */
#include <stdio.h>

int main()
{
   int x, y, z, result; /* declare variables */

   printf (  "Enter three integers;  ");       /* prompt */
   scanf("%d%d%d" , &x, &y, &z ); /* read three integers  */
   result  =  x  *  y  *  z;   /* multiply values  */
   printf( "The product is %d\n",  result  ); /* display result  */

   return 0;
  }
```

٢-٥- أ) الخطأ : &number . التصحيح : احذف علامة & .

[ملاحظة : سنناقش بإذن الله في الكتاب فيما بعد استثناءات لذلك] .

ب) الخطأ : number2 لا تسبقها علامة & . التصحيح : &number2 .

[ملاحظة : سنناقش بإذن الله في الكتاب فيما بعد استثناءات لذلك] .

ج) الخطأ : توجد فاصلة منقوطة بعد القوس الأيمن في شرط عبارة **if** .

التصحيح احذف هذه الفاصلة المنقوطة .

ملاحظة : نتيجة هذا الخطأ هي أن عبارة **printf** ستُنَفَّذ سواء تحقق شرط عبارة **if** أم لم يتحقق ، لأن الفاصلة المنقوطة الموجودة بعد القوس الأيمن تُعتبر عبارة خاوية (an empty statement) ، أي عبارة لا تفعل شيئا .

د) الخطأ : => ، التصحيح : >= أو <=

٢-٦- أ) `scanf( “%d”, &value );`

ب) `printf( “The product of %d and %d is %d\n”, x, y, z );`

`sumOfNumbers=firstNumber +secondNumber;`

ج) `if ( number >= largerst )`

د)

`largest = number;`

هـ) `/* Program to determine the largest of three integers */`

scanf(“%d”, &anInteger); و)

printf(“Remainder of %f divided by %d is %d\n”, x, y, x % y) ; ز)

if (x == y) ح)
  printf(“%d is equal to %d\n”, x, y);
printf(“The sum is %d\n”, x + y); ط)
printf(“The value you entered is: %d\n”, value); ي)

٢-٧- أ) F . بعض المؤثرات يتم تقييمها من اليسار إلى اليمين ، وبعضها الآخر من اليمين إلى اليسار ، بناء على خاصيتها التجميعية (associativity) .

ب) T

ج) F . الاسم h22 صالح بينما باقي الأسماء جميعها غير صالحة لأنها تبدأ برقم .

٢-٨ أ) 2 ب) 4 ج) x =

د) x = 2 هـ) 5 = 5

و) لا شئ . قيمة x + y تُسند إلى z .

ز) لا شئ . تتم قراءة قيمتين صحيحتين وتخزينهما في موقعي x , y .

ح) لا شئ . هذا تعليق .

ط) رمز السطر الجديد يُطبع ، ومؤشر الشاشة يوضع عند بداية السطر التالي على الشاشة .

٢-٩) أ

٢-١٠-أ)

* → / → + → −

x = 15

ب)

% → * → / → + → −

x = 3

ج)

x = (3 * 9 * (3 + (9 * 3 / (3)))) ;
5 6 4 2 3 1

x = 324

(١١-٢

```
/* Exercise 2-11 Solution */
#include <stdio.h>

int main()
{
  int x; /* define first number */
  int y; /* define second number */

  printf( "Enter two numbers: "); /* prompt user */
  scanf( "%d%d", &x, &y );       /* read values from keyboard */

  /* output results */
  printf( "The sum is %d\n", x + y );
  printf( "The product is %d\n", x * y );
  printf( "The difference is %d\n", x - y );
  printf( "The quotient is %d\n", x / y );
  printf( "The modulus is %d\n", x % y );

  return 0; /* indicate successful termination */

} /* end main */
```

```
Enter two numbers: 20 5
The sum is 25
The product is 100
The difference is 15
The quotient is 4
The modulus is 0
```

(١٢-٢

```
/* Exercise 2-12 Solution */
#include <stdio.h>

int main()
{
  printf( "1 2 3 4\n\n" ); /* part a */
```

```
   printf( "%d %d %d %d\n\n", 1, 2, 3, 4 ); /* part b */

   printf( "1 " ); /* part c */
   printf( "2 " );
   printf( "3 " );
   printf( "4\n" );

   return 0; /* indicates successful termination */

} /* end main */
```

(١٣-٢

```
/* Exercise 2-13 Solution */
#include <stdio.h>

int main()
{
   int x; /* define first number */
   int y; /* define second number */

   printf( "Enter two numbers: " ); /* prompt */
   scanf( "%d%d", &x, &y );        /* read two integers */

   /* compare the two numbers */
   if ( x > y ) {
      printf( "%d is larger\n", x );
   } /* end if */

   if ( x < y ) {
      printf( "%d is larger\n", y );
   } /* end if */

   if ( x == y ) {
      printf( "These numbers are equal\n" );
   } /* end if */

   return 0; /* indicate successful termination */

} /* end main */
```

```
/* Exercise 2-14 Solution */
#include <stdio.h>

int main()
{
  int a;       /* define first integer */
  int b;       /* define second integer */
  int c;       /* define third integer */
  int smallest; /* smallest integer */
  int largest;  /* largest integer */

  printf( "Input three different integers: " ); /* prompt user */
  scanf( "%d%d%d", &a, &b, &c ); /* read three integers */

  /* output sum, average and product of the three integers */
  printf( "Sum is %d\n", a + b + c );
  printf( "Average is %d\n", ( a + b + c ) / 3 );
  printf( "Product is %d\n", a * b * c );

  smallest = a; /* assume first number is the smallest */

  if ( b < smallest ) { /* is b smaller? */
    smallest = b;
  } /* end if */

  if ( c < smallest ) { /* is c smaller? */
    smallest = c;
  } /* end if */

  printf( "Smallest is %d\n", smallest );

  largest = a; /* assume first number is the largest */

  if ( b > largest ) { /* is b larger? */
    largest = b;
  } /* end if */

  if ( c > largest ) { /* is c larger? */
    largest = c;
  } /* end if */
```

```
   printf( "Largest is %d\n", largest );

   return 0; /* indicate successful termination */

} /* end main */
```

(١٥–٢

```
/* Exercise 2-15 Solution */
#include<stdio.h>

int main()
{
   int radius; /* circle radius */

   printf( "Input the circle radius: " ); /* prompt user */
   scanf( "%d", &radius ); /* read integer radius */

   /* calculate and output diameter, circumference and area */
   printf( "\nThe diameter is %d\n", 2 * radius );
   printf( "The circumference is %f\n", 2 * 3.14159 * radius );
   printf( "The area is %f\n", 3.14159 * radius * radius );

   return 0; /* indicate successful termination */

} /* end main */
```

(١٦–٢

(١٧–٢

```
/* Exercise 2-17 Solution */
#include <stdio.h>

int main()
```

```
{
  int largest;  /* largest integer */
  int smallest; /* smallest integer */
  int int1;     /* define int1 for user input */
  int int2;     /* define int2 for user input */
  int int3;     /* define int3 for user input */
  int temp;     /* temporary integer for swapping */

  printf( "Input 5 integers: " ); /* prompt user and read 5 ints */
  scanf("%d%d%d%d%d", &largest,&smallest,&int1, &int2, &int3);

  if ( smallest > largest ) { /* make comparisons */
    temp = largest;
    largest = smallest;
    smallest = temp;
  } /* end if */

  if ( int1 > largest ) {
    largest = int1;
  } /* end if */

  if ( int1 < smallest ) {
    smallest = int1;
  } /* end if */

  if ( int2 > largest ) {
    largest = int2;
  } /* end if */

  if ( int2 < smallest ) {
    smallest = int2;
  } /* end if */

  if ( int3 > largest ) {
    largest = int3;
  } /* end if */

  if ( int3 < smallest ) {
    smallest = int3;
  } /* end if */

  printf( "The largest value is %d\n", largest );
```

```
   printf( "The smallest value is %d\n", smallest );

   return 0; /* indicate successful termination */

} /* end main */
```

(٢-١٨

```
/* Exercise 2-18 Solution */
#include <stdio.h>

int main()
{
   int integer; /* integer input by user */

   printf( "Input an integer: " ); /* prompt */
   scanf( "%d", &integer );        /* read integer */

   /* test if integer is even */
   if ( integer % 2 == 0 ) {
      printf( "%d is an even integer\n", integer );
   } /* end if */

   /* test if integer is odd */
   if ( integer % 2 != 0 ) {
      printf( "%d is an odd integer\n", integer );
   } /* end if */

   return 0; /* indicate successful termination */

} /* end main */
```

(٢-١٩

```
/* Exercise 2-19 Solution */
#include <stdio.h>

int main()
{
   int integer1; /* first integer */
   int integer2; /* second integer */

   printf( "Input two integers: " );      /* prompt user */
```

```
   scanf( "%d%d", &integer1, &integer2 ); /* read two integers */

   /* use remainder operator */
   if ( integer1 % integer2 == 0 ) {
      printf( "%d is a multiple of %d ", integer1, integer2 );
      printf( "by a factor of %d\n", integer1 / integer2 );
   } /* end if */

   if ( integer1 % integer2 != 0 ) {
      printf( "%d is not a multiple of %d\n", integer1, integer2 );
   } /* end if */

   return 0; /* indicate successful termination */

} /* end main */
```

(٢٠-٢

```
/* Exercise 2-20 Solution */
#include<stdio.h>

int main()
{
   printf( "With eight printf() statements: \n" );

   printf( "* * * * * * * *\n" );
   printf( " * * * * * * * *\n" );
   printf( "* * * * * * * *\n" );
   printf( " * * * * * * * *\n" );
   printf( "* * * * * * * *\n" );
   printf( " * * * * * * * *\n" );
   printf( "* * * * * * * *\n" );
   printf( " * * * * * * * *\n" );

   printf( "\nNow with one printf() statement: \n" );

   printf( "* * * * * * * *\n * * * * * * * *\n"
         "* * * * * * * *\n * * * * * * * *\n"
         "* * * * * * * *\n * * * * * * * *\n"
         "* * * * * * * *\n * * * * * * * *\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

(٢١-٢

```
/* Exercise 2-21 Solution */
#include <stdio.h>

int main()
{
  char intEquivalent; /* letter, digit or character */

  printf( "Input a letter, digit, or character: " ); /* prompt */
  scanf( "%c", &intEquivalent ); /* read user input */

  printf( "%c's integer equivalent is %d\n", intEquivalent,
       intEquivalent );

  return 0; /* indicate successful termination */

} /* end main */
```

(٢٢-٢

```
/* Exercise 2-22 Solution */
#include<stdio.h>

int main()
{
  int number; /* number input by user */
  int temp1;  /* first temporary integer */
  int temp2;  /* second temporary integer */

  printf( "Enter a five-digit number: " ); /* prompt user */
  scanf( "%d", &number ); /* read integer */

  printf( "%d ", number / 10000 ); /* print left-most digit */
  temp2 = number % 10000;

  printf( "  %d ", temp2 / 1000 );
  temp1 = temp2 % 1000;

  printf( "  %d ", temp1 / 100 );
  temp2 = temp1 % 100;

  printf( "  %d ", temp2 / 10 );
  temp1 = temp2 % 10;

  printf( "  %d\n", temp1 ); /* print right-most digit */

  return 0; /* indicate successful termination */
```

```
} /* end main */
```

(٢-٢٣)

```
/* Exercise 2-23 Solution */
#include <stdio.h>

int main()
{
  int count = 0; /* initialize count to zero */

  /* calculate the square and cube for the numbers 0 to 10 */
  printf( "\nnumber\tsquare\tcube\n" );
  printf( "%d\t%d\t%d\n", count, count * count,
    count * count * count );

  count = count + 1; /* increment count by 1 */
  printf( "%d\t%d\t%d\n", count, count * count,
    count * count * count );

  count = count + 1;
  printf( "%d\t%d\t%d\n", count, count * count,
    count * count * count );

  count = count + 1;
  printf( "%d\t%d\t%d\n", count, count * count,
    count * count * count );

  count = count + 1;
  printf( "%d\t%d\t%d\n", count, count * count,
    count * count * count );

  count = count + 1;
  printf( "%d\t%d\t%d\n", count, count * count,
    count * count * count );

  return 0; /* indicate successful termination */

} /* end main */
```

أجوبة تمرينات رقم ٣

٣-١)

```
x = x + 1;
x += 1;
++x;
x++;
```

٣-٢-أ)

```
z = x++ + y;
```

ب)

```
product *= 2 ;
```

ج)

```
product = product * 2 ;
```

د)

```
if ( count > 10 )
    printf ( " Count is greater than 10. \n" );
```

هـ)

```
total -= --x;
```

و)

```
total += x --;
```

ز)

```
q %= divisor;
q = q % divisor;
```

ح)

```
printf ( "%.2f" , 123.4567 );
```

123.46

ط)

```
printf ( "%.3f\n" , 3.14159 );
```

3.142

٣-٣-أ)

```
int sum, x;
```

ب)

```
x = 1;
```

ج)

```
sum = 0;
```

د)

```
sum += x; or sum = sum + x;
```

هـ)

```
printf ( "The sum is: %d \n", sum );
```

٣-٤)

```
/* Calculate the sum of the integers from 1 to 10 */
  # include <stdio.h>

int main ()
{
```

```
  int sum, x; /* define variables sum and x */

        x = 1;      /* initialize x */
       sum = 0; /* initialize sum */

     while ( x <= 10 ) { /* loop while x is less than or equal to 10 */
             sum += x; /* add x to sum */
             ++x; /* increment x */
     } /* end while */
     printf ( "The sum is: %d \n", sum ); /* display sum */

       return 0;
} /* end main function */
```

٣-٥) `product = 25, x = 6;`

٣-٦-أ) `scanf ( "%d" , &x );`

ب) `scanf ( "%d" , &y );`

ج) `i = 1;`

د) `power = 1;`

هـ) `power *= x;`

و) `y++;`

ز) `if ( y <= x )`

ح) `printf( "%d" , power );`

٣-٧)

```
 /* raise x to the y power */
 # include <stdio.h>

 int main ()
 {
       int x,  y,  i,  power;  /* define variables */

      i = 1;       /* initialize i */
     power = 1; /* initialize power */
     scanf( "%d", &x ); /* read value for  x  from user */
     scanf( "%d", &y ); /* read value for  y  from user */
```

```
        while ( i <= y ) { /* loop while i is less than or equal to y */
            power *= x; /* multiply power by x *
             ++i; /* increment i */
      } /* end while */
      printf ( "%d", power ); /* display power */

        return 0;
} /* end main function */
```

٣-٨ - أ) الخطأ : ناقص قوس الإغلاق الأيمن "}" في جسم عبارة **while** .

التصحيح : إضافة القوس "}" يمين العبارة ++c;

ب) الخطأ استخدام الدقة في تحديد تحويل **scanf** .

التصحيح : حذف 4 . من تحديد التحويل .

ج) الخطأ : وجود الفاصلة المنقوطة بعد **else** يؤدي إلى خطأ منطقي ، حيث أن عبارة **printf** الثانية ستُنفذ دائما .

التصحيح : احذف الفاصلة المنقوطة بعد **else** .

٣-٩) قيمة المتغير **z** تظل 100 دائما ولا تتغير طوال تنفيذ عبارة **while** ، وبذلك تصبح العروة لا نهائية . التصحيح : يجب إنقاص قيمة **z** (مثلا : إضافة العبارة **z--;** إلى جسم **while**) بحيث تصبح قيمة **z** في النهاية 0 .

٣-١٠) أ)

```
if ( age >= 65 ) /* : removed */
   printf( "Age is greater than or equal to 65\n" );
else
    printf( "Age is less than 65\n" );
```

ب)

```
int x = 1, total = 0;
while ( x <= 10 ) {
     total += x;
     ++x;
}
```

ج)

```
while ( x <= 100 )  {
     total += x;
     ++x;
}
```

د)

```
while ( y > 0 ) {
    printf( "%d\n", y );
    --y;
}
```

٣-١١)

```
1
4
9
16
25
36
49
64
81
100
Total is 385
```

٣-١٢ – i) جـ ii) ب iii) د

iv) أ v) د vi) جـ

vii) د

٣-١٣)

```
/* Exercise 3.13 Solution */
#include <stdio.h>

int main()
{
  double gallons;             /* gallons used for current tank*/
  double miles;               /* miles driven for current tank*/
  double totalGallons = 0.0; /* total gallons used */
  double totalMiles = 0.0;   /* total miles driven */
  double totalAverage;       /* average miles/gallon */
```

```
   /* get gallons used for first tank */
   printf( "Enter the gallons used ( -1 to end): " );
   scanf( "%lf", &gallons );

   /* loop until sentinel value read from user */
   while ( gallons != -1.0 ) {
      totalGallons += gallons; /* add current tank gallons to total */

      printf( "Enter the miles driven: " ); /* get miles driven */
      scanf( "%lf", &miles );
      totalMiles += miles; /* add current tank miles to total */

      /* display miles per gallon for current tank */
      printf( "The Miles / Gallon for this tank was %f\n\n",
         miles / gallons );

      /* get next tank's gallons */
      printf( "Enter the gallons used ( -1 to end ): " );
      scanf( "%lf", &gallons );
   } /* end while */

   /* calculate average miles per gallon over all tanks */
   totalAverage = totalMiles / totalGallons;
   printf( "\nThe overall average Miles/Gallon was %f\n",
totalAverage );

   return 0; /* indicate successful termination */

} /* end main */
```

(٣-١٤

```
/* Exercise 3.14 Solution */
#include <stdio.h>

int main()
{
```

```
int accountNumber; /* current account's number */
double balance;      /* current account's starting balance */
double charges;    /* current account's total charges */
double credits;    /* current account's total credits */
double limit;      /* current account's credit limit */


/* get account number */
printf( "\nEnter account number ( -1 to end): " );
scanf( "%d", &accountNumber );


/* loop until sentinel value read from user */
while ( accountNumber != -1 ) {
   printf( "Enter beginning balance: " );
   scanf( "%lf", &balance );


   printf( "Enter total charges: " );
   scanf( "%lf", &charges );


   printf( "Enter total credits: " );
   scanf( "%lf", &credits );


   printf( "Enter credit limit: " );
   scanf( "%lf", &limit );


   balance += charges - credits; /* calculate balance */


   /* if balance is over limit, display account number
      with credit limit and balance to two digits of precision */
   if ( balance > limit ) {
      printf( "%s%d\n%s%.2f\n%s%.2f\n%s\n",
         "Account:      ", accountNumber, "Credit limit: ", limit,
         "Balance:      ", balance, "Credit Limit Exceeded." );
   } /* end if */


   /* prompt for next account */
   printf( "\nEnter account number ( -1 to end ): " );
   scanf( "%d", &accountNumber );
```

```
   } /* end while */

   return 0; /* indicate successful termination */

} /* end main */
```

(٣-١٥

```
/* Exercise 3.15 Solution */
#include <stdio.h>

int main()
{
   double sales; /* gross weekly sales */
   double wage;  /* commissioned earnings */

   /* get first sales */
   printf( "Enter sales in dollars ( -1 to end): " );
   scanf( "%lf", &sales );

   /* loop until sentinel value read from user */
   while ( sales != -1.0 ) {
      wage = 200.0 + 0.09 * sales; /* calculate wage */

      /* display salary */
      printf( "Salary is: $%.2f\n\n", wage );

      /* prompt for next sales */
      printf( "Enter sales in dollars ( -1 to end ): " );
      scanf( "%lf", &sales );
   } /* end while */

   return 0; /* indicate successful termination */

} /* end main */
```

```
/* Exercise 3.16 Solution */
#include <stdio.h>

int main()
{
  double principal; /* loan principal */
  double rate;      /* interest rate */
  double interest;  /* interest charge */
  int term;         /* length of loan in days */

  /* get loan principal */
  printf( "Enter loan principal ( -1 to end): " );
  scanf( "%lf", &principal );

  /* loop until sentinel value is read from user */
  while ( principal != -1.0 ) {
    printf( "Enter interest rate: " ); /* get rate */
    scanf( "%lf", &rate );

    printf( "Enter term of the loan in days: " ); /* get term */
    scanf( "%d", &term );

    /* calculate interest charge */
    interest = principal * rate * term / 365.0;
    printf( "The interest charge is $%.2f\n\n", interest );

    /* get next loan principal */
    printf( "Enter loan principal ( -1 to end ): " );
    scanf( "%lf", &principal );
  } /* end while */

  return 0; /* indicate successful termination */

} /* end main */
```

```
/* Exercise 3.17 Solution */
#include <stdio.h>

int main(  void  )
{
  double hours;  /* total hours worked */
  double rate;    /* hourly pay rate */
  double salary; /* gross pay */

  /* get first employee's hours worked */
  printf(  "Enter number of hours worked (  -1 to end  ): "  );
  scanf(  "%lf", &hours  );

  /* loop until sentinel value read from user */
  while (  hours != -1.0  ) {

    /* get hourly rate */
    printf(  "Enter hourly rate of the worker (  $00.00  ): "  );
    scanf(  "%lf", &rate  );

    /* if employee worked less than 40 hours */
    if (  hours <= 40  ) {
      salary = hours * rate;
    } /* end if */
    else { /* compute "time-and-a-half" pay */
      salary = 40.0 * rate + (  hours - 40.0  ) * rate * 1.5;
    } /* end else */

    /* display gross pay */
    printf(  "Salary is $%.2lf\n\n", salary  );

    /* prompt for next employee's data */
    printf(  "Enter number of hours worked (  -1 to end  ): "  );
    scanf(  "%lf", &hours  );
  } /* end while */

  return 0; /* indicate successful termination */
```

```
} /* end main */
```

(٣-١٨

```
/* Exercise 3.18 Solution */
#include <stdio.h>

int main()
{
   int c; /* define c to use decrement operator */

   c = 5;
   printf( "%d\n", c );
   printf( "%d\n", --c ); /* predecrement */
   printf( "%d\n\n", c );

   c = 5;
   printf( "%d\n", c );
   printf( "%d\n", c-- ); /* postdecrement */
   printf( "%d\n\n", c );

   return 0; /* indicate successful termination */

} /* end main */
```

```
5
4
4

5
5
4
```

(٣-١٩

```
/* Exercise 3.19 Solution */
#include <stdio.h>

int main()
{
   int i = 0; /* initialize i */
```

```
   /* loop while i is less than 11 */
   while ( ++i < 11 ) {
      printf( "%d   ", i );
   } /* end while */

   return 0; /* indicate successful termination */

} /* end main */
```

```
1      2      3      4      5      6      7      8      9      10
```

(٢٠–٣

```
/* Exercise 3.20 Solution */
#include <stdio.h>

int main()
{
   int counter; /* counter for 10 repetitions */
   int number; /* current number input */
   int largest; /* largest number found so far */

   /* get first number */
   printf( "Enter the first number: " );
   scanf( "%d", &largest );
   counter = 2;

   /* loop 9 more times */
   while ( counter <= 10 ) {
      printf( "Enter next number: " ); /* get next number */
      scanf( "%d", &number );

      /* if current number input is greater than largest number,
         update largest */
      if ( number > largest ) {
         largest = number;
      } /* end if */

      counter++;
   } /* end while */

   printf( "Largest is %d\n", largest ); /* display largest number */

   return 0; /* indicate successful termination */
```

```
} /* end main */
```

```
Enter the first number: 7
Enter next number: 37
Enter next number: 78
Enter next number: 2
Enter next number: 437
Enter next number: 72
Enter next number: 1
Enter next number: 4
Enter next number: 36
Enter next number: 100
Largest is 437
```

(٣–٢١

```
/* Exercise 3.21 Solution */
#include <stdio.h>

int main()
{
  int n = 0; /* counter */

  /* display table headers */
  printf( "\tN\t\t10 * N\t\t100 * N\t\t1000 * N\n\n" );

  /* loop 10 times */
  while ( ++n <= 10 ) {

    /* calculate and display table values */
    printf( "\t%-2d\t\t%-5d\t\t%-5d\t\t%-6d\n",
          n, 10 * n, 100 * n, 1000 * n );
  } /* end while */

  return 0; /* indicate successful termination */

} /* end main */
```

(٣–٢٢

```
/* Exercise 3.22 Solution */
#include <stdio.h>
```

```
int main()
{
  int a = 3; /* counter */


  /* display table headers */
  printf( "A\tA+2\tA+4\tA+6\n\n" );


  /* loop 5 times */
  while ( a <= 15 ) {


    /* calculate and display table values */
    printf( "%d\t%d\t%d\t%d\n", a, a + 2, a + 4, a + 6 );
    a += 3;
  } /* end while */


  return 0; /* indicate successful termination */


} /* end main */
```

(٣-٢٣

```
/* Exercise 3.23 Solution */
#include <stdio.h>

int main()

{
  int counter;          /* counter for 10 repetitions */
  int number;           /* current number input */
  int largest;          /* largest number found */
  int secondLargest = 0; /* second largest number found */

  printf( "Enter the first number: " ); /* get first number */
  scanf( "%d", &largest );
  counter = 2;

  /* loop 9 more times */
  while ( counter <= 10 ) {
    printf( "Enter next number: " ); /* prompt for next number */
```

```
      scanf( "%d", &number );

      /* if current number is greater than largest */
      if ( number > largest ) {

         /* update second largest with previous largest */
         secondLargest = largest;

         /* update largest with current number */
         largest = number;
      } /* end if */
      else {

         /* if number is between secondLargest and largest */
         if ( number > secondLargest ) {
            secondLargest = number;
         } /* end if */

      } /* end else */

      ++counter;
   } /* end while */

   /* display largest two numbers */
   printf( "Largest is %d\n", largest );
   printf( "Second largest is %d\n", secondLargest );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the first number: 100
Enter next number: 102
Enter next number: 83
Enter next number: 3883
Enter next number: 328
Enter next number: 28
Enter next number: 839
Enter next number: 2398
Enter next number: 182
Enter next number: 0
Largest is 3883
Second largest is 2398
```

(٣-٢٤

```
/* Exercise 3.24 Solution */
#include <stdio.h>

int main()

{
  int passes = 0;   /* number of passes */
  int failures = 0; /* number of failures */
  int student = 1;  /* student counter */
  int result;       /* one exam result */

  /* process 10 students using counter-controlled loop */
  while ( student <= 10 ) {

    /* prompt user for input and obtain value from user */
    printf( "Enter result ( 1=pass, 2=fail ): " );
    scanf( "%d", &result );

    /* loop until valid input */
    while ( result != 1 && result != 2 ) {
      printf( "Invalid result\nEnter result ( 1=pass, 2=fail ): " );
      scanf( "%d", &result );
    } /* end inner while */

    /* if result 1, increment passes */
    if ( result == 1 ) {
      ++passes;
    } /* end if */
    else { /* if result is not 1, increment failures */
      ++failures;
    } /* end else */

    ++student;
  } /* end while */
```

```
   printf( "Passed %d\nFailed %d\n", passes, failures );

   /* if more than eight students passed, print "raise tuition" */
   if ( passes >= 8 ) {
      printf( "Raise tuition\n" );
   } /* end if */

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter result ( 1=pass, 2=fail ): 1
Enter result ( 1=pass, 2=fail ): 2
Enter result ( 1=pass, 2=fail ): 3
Invalid result
Enter result ( 1=pass, 2=fail ): 4
Invalid result
Enter result ( 1=pass, 2=fail ): 2
Enter result ( 1=pass, 2=fail ): 2
Enter result ( 1=pass, 2=fail ): 2
Enter result ( 1=pass, 2=fail ): 1
Enter result ( 1=pass, 2=fail ): 1
Enter result ( 1=pass, 2=fail ): 1
Enter result ( 1=pass, 2=fail ): 1
Enter result ( 1=pass, 2=fail ): 1
Passed  6
Failed  4
```

(٢٥–٣

```
****
++++++++
****
++++++++
****
++++++++
****
++++++++
****
++++++++
```

(٣–٢٦

```
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
```

(٣–٢٧ – أ

x = 9, y = 11

```
Prayer is light.
Patience is illumination.
```

x = 11, y = 9

```
Patience is illumination.
```

(ب

x = 9, y = 11

```
Prayer is light.
```

x = 11, y = 9

```
Charity is a proof.
Patience is illumination.
```

(٣–٢٨ – أ

```
if ( y == 8 ) {
  if ( x == 5 )
    printf( “Every takbeera is a charity.\n” );
  else
```

```
    printf( “Charity is a proof.\n” );
  printf( “ Patience is illumination.\n” );
  printf( “Every tahleela is a charity.\n” );
}
```

ب)

```
if ( y == 8 )
  if ( x == 5 )
    printf( “Every takbeera is a charity.\n” );
  else {
    printf( “Charity is a proof.\n” );
    printf( “Patience is illumination.\n” );
    printf( “Every tahleela is a charity.\n” );
  }
```

ج)

```
if ( y == 8 )
  if ( x == 5 )
    printf( “Every takbeera is a charity.\n” );
  else {
    printf( “Charity is a proof.\n” );
    printf( “Patience is illumination.\n” );
  }
printf( “Every tahleela is a charity.\n” );
```

د)

```
if ( y == 8 ) {
  if ( x == 5 )
    printf( “Every takbeera is a charity.\n” );
}
else {
  printf( “Charity is a proof.\n” );
  printf( “Patience is illumination.\n” );
  printf( “Every tahleela is a charity.\n” );
}
```

٣-٢٩)

```
/* Exercise 3.29 Solution */
#include<stdio.h>

int main()
{
  int side;     /* side counter */
```

```
   int temp;     /* temporary integer */
   int asterisk; /* asterisk counter */

   printf( "Enter the square side: " ); /* get size of square */
   scanf( "%d", &side );

   temp = side;

   /* loop through rows of square */
   while ( side-- > 0 ) {
      asterisk = temp;

      /* loop through columns of square */
      while ( asterisk-- > 0 ) {
         printf( "*" );
      } /* end inner while */

      putchar( '\n' );
   } /* end outer while */

  return 0; /* indicate successful termination */

} /* end main */
```

(٣٠-٣

```
/* Exercise 3.30 Solution */
#include<stdio.h>


int main()
{
   int side;       /* side counter */
   int rowPosition; /* row counter */
   int size;       /* length of side */


   printf( "Enter the square side: " ); /* prompt for side length */
   scanf( "%d", &side );


   size = side; /* set size counter to length of side */
```

```
  /* loop side number of times */
  while ( side > 0 ) {
    rowPosition = size; /* set row counter to length of size */

    /* loop rowPosition number of times */
    while ( rowPosition > 0 ) {

      /* if side or row counter is 1 or size print an '*' */
      if ( size == side ) {
        printf( "*" );
      } /* end if */
      else if ( side == 1 ) {
        printf( "*" );
      } /* end else if */
      else if ( rowPosition == 1 ) {
        printf( "*" );
      } /* end else if */
      else if ( rowPosition == size ) {
        printf( "*" );
      } /* end else if */
      else { /* otherwise, print a space */
        printf( " " );
      } /* end else */

      --rowPosition; /* decrement row counter */
    } /* end inner while */

    printf( "\n" ); /* new line for next row */
    --side; /* decrement side counter */
  } /* end outer while */

  return 0; /* indicate successful termination */

} /* end main */
```

(٣١-٣

```
/* Exercise 3.31 Solution */
#include<stdio.h>

int main()
{
```

```
int number;      /* input number */
int temp1;       /* first temporary integer */
int temp2;       /* second temporary integer */
int firstDigit;  /* first digit of input */
int secondDigit; /* second digit of input */
int fourthDigit; /* fourth digit of input */
int fifthDigit;  /* fifth digit of input */

printf( "Enter a five-digit number: " ); /* get number */
scanf( "%d", &number );


temp1 = number;


/* determine first digit by integer division by 10000 */
firstDigit = temp1 / 10000;
temp2 = temp1 % 10000;


/* determine second digit by integer division by 1000 */
secondDigit = temp2 / 1000;
temp1 = temp2 % 1000;


temp2 = temp1 % 100;


/* determine fourth digit by integer division by 10 */
fourthDigit = temp2 / 10;
temp1 = temp2 % 10;


fifthDigit = temp1;


/* if first and fifth digits are equal */
if ( firstDigit == fifthDigit ) {


  /* if second and fourth digits are equal */
  if ( secondDigit == fourthDigit ) {

    /* number is a palindrome */
    printf( "%d is a palindrome\n", number );
  } /* end if */
  else { /* number is not a palindrome */
```

```
      printf( "%d is not a palindrome\n", number );
    } /* end else */

  } /* end if */
  else { /* number is not a palindrome */
    printf( "%d is not a palindrome\n", number );
  } /* end else */

  return 0; /* indicate successful termination */

} /* end main */
```

```
Enter a five digit number: 18181
18181 is a palindrome
```

```
Enter a five digit number: 16738
16738 is not a palindrome
```

(٣٢-٣

```
/* Exercise 3.32 Solution */
#include<stdio.h>

int main()
{
  int binary;         /* current value of binary number */
  int number;         /* input binary number */
  int decimal = 0;    /* current value of decimal number */
  int highBit = 16;   /* value of highest bit */
  int factor = 10000; /* factor of 10 to pick off digits */

  /* prompt for binary input */
  printf( "Enter a binary number ( 5 digits maximum ): " );
  scanf( "%d", &binary );

  number = binary; /* save in number for final display */

  /* loop 5 times using powers of 2 */
  while ( highBit >= 1 ) {
```

```
      /* update decimal value with decimal value corresponding
         to current highest binary bit */
      decimal += binary / factor * highBit;

      /* reduce high bit by factor of 2, i.e.,
         move one bit to the right */
      highBit /= 2;

      /* reduce binary number to eliminate current highest bit */
      binary %= factor;

      /* reduce factor by power of 10, i.e.,
         move one bit to the right */
      factor /= 10;
   } /* end while */

   /* display decimal value */
   printf( "The decimal equivalent of %d is %d\n", number, decimal );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter a binary number ( 5 digits maximum ):: 10111
The decimal equivalent of 10111 is 23
```

```
Enter a binary number ( 5 digits maximum ):: 1101
The decimal equivalent of 1101 is 13
```

(٣٣-٣

```
/* Exercise 3.33 Solution */
#include<stdio.h>

int main()
{
   long int count = 1; /* counter */

   /* loop to 300,000,000 */
   while( count <= 300000000 ) {
```

```
    if ( count % 100000000 == 0 ) {
      printf( "Multiple is %d\n", count / 100000000 );
    } /* end if */

    ++count; /* increment count */
  } /* end while */

  return 0; /* indicate successful termination */

} /* end main */
```

```
Multiple is 1
Multiple is 2
Multiple is 3
```

(٣٤-٣

```
/* Exercise 3.34 Solution */
#include <stdio.h>

int main()
{
 int count = 0; /* counter */

 /* loop to 100 */
 while( ++count <= 100 )

   /* print a new line after every 10th asterisk */
   count % 10 == 0 ? printf( "*\n" ) : printf( "*" );

 return 0; /* indicate successful termination */

} /* end main */
```

```
**********
**********
**********
**********
**********
**********
```

```
**********
**********
**********
**********
```

(٣٥–٣

```
/* Exercise 3.35 Solution */
#include <stdio.h>

int main()
{
  int number;         /* user input */
  int numCopy;        /* copy of number */
  int factor = 10000; /* set factor to pick off digits */
  int digit;          /* individual digit of number */
  int sevens = 0;     /* sevens counter */

  printf( "Enter a 5-digit number: " ); /* get number from user */
  scanf( "%d", &number );

  numCopy = number;

  /* loop through each of the 5 digits */
  while ( factor >= 1 ) {
    digit = numCopy / factor; /* pick off next digit */

    if ( digit == 7 ) { /* if digit equals 7, increment sevens */
      ++sevens;
    } /* end if */

    numCopy %= factor;
    factor /= 10;
  } /* end while */

  /* output number of sevens */
  printf("The number %ld has %d seven(s) in it\n", number, sevens );

  return 0; /* indicate successful termination */
```

```
} /* end main */
```

```
Enter a 5- digit number: 17737
The number 17737 has 3 seven (s) in it
```

```
Enter a 5- digit number: 11727
The number 11727 has 2 seven (s) in it
```

(٣-٣٦)

```
/* Exercise 3.36 Solution */
#include <stdio.h>

int main()
{
  int side = 8; /* side counter */
  int row;      /* row counter */
  int mod;      /* remainder */

  /* loop 8 times */
  while ( side >= 1 ) {
    row = 8; /* reset row counter */
    mod = side % 2;

    /* loop 8 times */
    while ( row >= 1 ) {

      /* if odd row, begin with a space */
      if ( mod != 0 ) {
        printf( " " );
        mod = 0;
      } /* end if */

      printf( "* " );
      --row;
    } /* end while */
```

```
    printf( "\n" ); /* go to next line */
    --side;
  } /* end while */

  return 0; /* indicate successful termination */

} /* end main */
```

٣-٣٧) يتوقف تنفيذ البرنامج عندما يتم تجاوز أكبر عدد صحيح (largest integer is exceeded) **، أي أن شرط استمرار العروة يصبح خاطئا (غير متحقق) عندما نتجاوز أكبر قيمة مسموح بها للعدد الصحيح في الحاسوب وفي نظام الأربعة بايتات** (4 – byte system) **فإن قيمة أكبر عدد صحيح هي** 2147483647 **، وأي قيمة أكبر من هذه فإنها تُمثَّل بعدد سالب مما يجعل شرط استمرار العروة غير متحقق .**

```
/* Exercise 3.37 Solution */
#include <stdio.h>

int main()
{
  int multiple = 1; /* counter */

  /* infinite loop */
  while ( multiple > 0 ) {

    /* calculate the next power of two */
    multiple *= 2;
    printf( "%d\n", multiple );
  } /* end while */

  return 0; /* indicate successful termination */

} /* end main */
```

```
2
4
8
16
32
64
128
256
```

```
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
16777216
33554432
67108864
134217728
268435456
536870912
1073741824
-2147483648
```

(٣٨–٣

```
/* Exercise 3.38 Solution */
#include<stdio.h>

int main()
{
   float radius;       /* input radius */
   float pi = 3.14159; /* value for pi */

   printf( "Enter the radius: "); /* get radius value */
   scanf( "%f", &radius );

   /* compute and display diameter */
   printf( "The diameter is %.2f\n", radius * 2 );
```

```
   /* compute and display circumference */
   printf( "The circumference is %.2f\n", 2 * pi * radius );

   /* compute and display area */
   printf( "The area is %.2f\n", pi * radius * radius );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the radius: 4.7
The diameter is 9.40
The Circumference is 29.53
The area is 69.40
```

(٣٩-٣

```
printf ( "%d" , 1 + x + y );
```

(٤٠-٣

```
/* Exercise 3.40 Solution */
#include <stdio.h>

int main()
{
   double a; /* first number */
   double b; /* second number */
   double c; /* third number */

   /* input 3 numbers */
   printf( "Enter three doubleing point numbers: " );
   scanf( "%lf%lf%lf", &a, &b, &c);

   if (( a < b + c) && ( b < a + c ) &&( c < a + b )) {
      printf( "The three numbers could be sides of a triangle\n" );
   } /* end if */
   else {
      printf( "The three numbers probably");
      printf( " are not the sides of a triangle\n" );
   } /* end if */
```

```
  return 0; /* indicate successful termination */

} /* end main */
```

```
Enter three doubleing point numbers: 5.7 3.6 1.2
The three numbers probably are not the sides of a triangle
```

```
Enter three doubleing point numbers: 3.0 4.0 5.0
The three numbers could be sides of a triangle
```

(٣-٤١)

```
/* Exercise 3.41 Solution */
#include <stdio.h>

int main()
{
  int a; /* first number */
  int b; /* second number */
  int c; /* third number */

  /* input three numbers */
  printf( "Enter three integers: ");
  scanf( "%d%d%d", &a, &b, &c );

  /* use Pythagorean Theorem */
  if (( c * c == a * a + b * b )| | ( b * b = = a * a + c * c| |
                                    ( a * a = = b * b + c * c)) {
    printf( "The three integers are the sides of");
    printf( " a right triangle\n" );
  } /* end if */
  else {
    printf( "The three integers are not the sides");
    printf( " of a right triangle\n" );
  } /* end else */

  return 0; /* indicate successful termination */

} /* end main */
```

```
Enter three integers: 3 4 5
The three integers are the sides of a right triangle
```

```
Enter three integers: 9 4 1
The three integers are not the sides of a right triangle
```

(٣-٤٢-أ)

```
/* Exercise 3.42 Part A solution */
#include <stdio.h>

int main()
{
   int first;  /* first digit replacement */
   int second; /* second digit replacement */
   int third;  /* third digit replacement */
   int fourth; /* fourth digit replacement */
   int digit;  /* input number */
   int temp1;  /* temporarily hold digit */
   int temp2;  /* temporarily hold digit */
   int encryptedNumber; /* resulting encrypted number */

   /* prompt for input */
   printf( "Enter a four digit number to be encrypted: " );
   scanf( "%d", &digit );

   temp1 = digit;

   /* retrieve each digit and replace with
      (sum of digit and 7) mod 10 */
   first = ( temp1 / 1000 + 7 ) % 10;
   temp2 = temp1 % 1000;

   second = ( temp2 / 100 + 7 ) % 10;
   temp1 = temp2 % 100;

   third = ( temp1 / 10 + 7 ) % 10;
   temp2 = temp1 % 10;

   fourth = ( temp2 + 7 ) % 10;
```

```
   /* swap first and third */
   temp1 = first;
   first = third * 1000; /* multiply by 1000 for 1st digit component */
   third = temp1 * 10; /* multiply by 10 for 3rd digit component */

   /* swap second and fourth */
   temp1 = second;
   second = fourth * 100; /* multiply by 100 for 2nd digit component
*/
   fourth = temp1 * 1;

   /* add components to obtain encrypted number */
   encryptedNumber = first + second + third + fourth;

   /* display encrypted number */
   printf( "Encrypted number is %d\n", encryptedNumber );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter a four digit number to be encrypted: 5678
Encrypted number is 4523
```

ب)

```
/* Exercise 3. 42 Part B Solution */
#include <stdio.h>

int main()
{
   int first;     /* first decrypted digit */
   int second;    /* second decrypted digit */
   int third;     /* third decrypted digit */
   int fourth;    /* fourth decrypted digit */
   int decrypted; /* decrypted number */
   int temp1;     /* temporarily hold digit */
   int temp2;     /* temporarily hold digit */
   int encryptedNumber; /* input number */

   /* prompt for input */
   printf( "Enter a four digit encrypted number: " );
   scanf( "%d", &encryptedNumber );
```

```
   temp1 = encryptedNumber;

   /* retrieve each digit and decrypt by
      (sum of digit and 3) mod 10 */
   first = ( temp1 / 1000 );
   temp2 = temp1 % 1000;

   second = ( temp2 / 100 );
   temp1 = temp2 % 100;

   third = ( temp1 / 10 );
   temp2 = temp1 % 10;

   fourth = temp2;

   temp1 = ( first + 3 ) % 10;
   first = ( third + 3 ) % 10;
   third = temp1;

   temp1 = ( second + 3 ) % 10;
   second = ( fourth + 3 ) % 10;
   fourth = temp1;

   /* add components to obtain decrypted number */
   decrypted = ( first * 1000 ) + ( second * 100 ) +
               ( third * 10 ) + fourth;

   /* display decrypted number */
   printf( "Decrypted number is %d\n", decrypted );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter a four digit encrypted number: 4523
Decrypted number is 5678
```

(٣-٤٣-أ

```
/* Exercise 3.43 Part A Solution */
#include <stdio.h>


int main()
{
```

```
   int n;                    /* current multiplication factor */
   int number = -1;          /* input number */
   unsigned factorial = 1; /* resulting factorial */


   /* loop until valid input */
   do {
      printf( "Enter a nonnegative integer: " );
      scanf( "%d", &number );
   } while ( number < 0 ); /* end do...while */


   n = number;


   /* compute factorial */
   while( n >= 0 ) {


      if ( n == 0 ) {
         factorial *= 1;
      } /* end if */
      else {
         factorial *= n;
      } /* end else */


      --n;
   } /* end while */


   /* display factorial */
   printf( "%d! is %u\n", number, factorial );


   return 0; /* indicate successful termination */


} /* end main */
```

```
Enter a nonnegative integer: 5
5! is 120
```

```
Enter a nonnegative integer: 9
9! is 362880
```

```
Enter a nonnegative integer: -8
Enter a nonnegative integer: 0
0! is 1
```

ب)

```
/* Exercise 3.43 Part B Solution */
#include <stdio.h>

int main()
{
  int n = 0;         /* loop counter for accuracy */
  int fact = 1;      /* current n factorial */
  int accuracy = 10; /* degree of accuracy */
  double e = 0;      /* current estimated value of e */

  /* loop until degree of accuracy */
  while( n <= accuracy ) {

    if ( n == 0 ) {
      fact *= 1;
    } /* end if */
    else {
      fact *= n;
    } /* end else */

    e += 1.0 / fact;
    ++n;
  } /* end while */

  printf( "e is %f\n", e ); /* display estimated value */
```

```
  return 0; /* indicate successful termination */

} /* end main */
```

```
e is 2.718282
```

(ج

```
/* Exercise 3.43 Part C Solution */
#include <stdio.h>

int main()
{
  int n = 0;         /* counter */
  int accuracy = 15; /* degree of accuracy */
  int x = 3;       /* exponent */
  int times = 0;     /* counter */
  int count;         /* copy of n */
  double e = 1.0;    /* e raised to the x power */
  double exp = 0.0;  /* x raised to the n power */
  double fact = 1.0; /* n factorial */

  /* loop while less than degree of accuracy */
  while( n <= accuracy ) {
    count = n;

    /* update n! */
    if ( n == 0 ) {
      fact *= 1.0;
    } /* end if */
    else {
      fact *= n;
    } /* end else */

    while ( times < count ) {

      /* calculate x raised to the n power */
      if ( times == 0 ) {
```

```
            exp = 1.0;
            exp *= x;
          } /* end if */
          else {
            exp *= x;
          } /* end else */


          ++times;
        } /* end while */


        e += exp / fact; /* update e raised to the x power */
        ++n;
      } /* end while */


      /* display result */
      printf( "e raised to the %d power is %f\n", x, e );


      return 0; /* indicate successful termination */


    } /* end main */
```

```
e raised to the 3 power is 20.085534
```

(٤٤ – ٣

```
/* Exercise 3.44 Solution */
#include <stdio.h>

int main( void )
{
    int value; /* current value */
    int count = 0; /* number of values */
    int total = 0; /* sum of integers */

    /* display prompt */
    printf( "Enter an integer ( 9999 to end ): " );
    scanf( "%d", &value );
```

```
   /* loop while sentinel value not read from user */
   while ( value != 9999 ) {
      total += value; /* update total */
      ++count;

      /* get next value */
      printf( "Enter next integer ( 9999 to end ): " );
      scanf( "%d", &value );
   } /* end while */

   /* show average if more than 0 values entered */
   if ( count != 0 ) {
      printf( "\nThe average is: %.2f\n", ( double ) total / count );
   } /* end if */
   else {
      printf( "\nNo values were entered.\n" );
   } /* end else */

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter an integer ( 9999 to end ): 1
Enter next integer ( 9999 to end ): 2
Enter next integer ( 9999 to end ): 3
Enter next integer ( 9999 to end ): 4
Enter next integer ( 9999 to end ): 5
Enter next integer ( 9999 to end ): 6
Enter next integer ( 9999 to end ): 9999
The average is: 3.50
```

أجوبة تمرينات رقم ٤

٤ – ١ – أ)

```
sum = 0;
for ( count = 1; count <= 99; count += 2 )
    sum += count;
```

ب)

```
for ( x = 1; x <= 20; x++ ) {
    printf( "%d" , x );
     if ( x % 5 == 0 )
          printf( "\n" );
     else
          printf("\t" );
}
```

أو

```
for ( x = 1; x <= 20; x++ )
    if ( x % 5 == 0 )
          printf( "%d\n" , x );
     else
          printf("%d\t" , x );
```

٤ – ٢ – أ) الخطأ : استخدم عدد ذي نقطة عائمة (floating point number) للتحكم في عبارة **for** للتكرار .

التصحيح : استخدم عددا صحيحا (an integer) للتحكم في عبارة **for** ، واكتب العمليات الحسابية بحيث تضمن لنا الحصول على القيم المطلوبة . مثلا :

```
for ( y = 1; y != 10; y++ )
    printf( "%f\n" , ( float ) y / 10 );
```

ب) الخطأ : عدم وجود عبارة **break** في عبارات **case** الأولى ، إلا إذا كان المبرمج يريد تنفيذ عبارات **case** الثانية كلما نُفِّذت عبارات **case** الأولى ، ففي هذه الحالة لا يكون هناك خطأ .

التصحيح : إضافة عبارة **break** في نهاية عبارات **case** الأولى .

٤ – ٣ – أ) ثلاثة أخطاء :

- حرف F في كلمة For يجب أن يكون حرفا صغيرا لا كبيرا .

- شروط for بين القوسين يجب أن تفصل بينها فاصلات منقوطة (semicolons)
  (وليس فاصلات عادية (commas) .
- العروة لا نهائية .

التصحيح يمكن أن يكون كما يلي :

```
for ( x = 1; x <= 100; x++ )
   printf( "%d\n", x);
```

ب) الخطأ : عدم وجود عبارة **break** في الحالة case 0 ، وذلك يؤدي إلى طباعة العبارتين المذكورتين في القطعة .

التصحيح :

```
switch ( value % 2 ) {
     case 0:
          printf( "Even integer\n" );
          break;
     case 1:
          printf( "Odd integer\n" );
```

ج) الخطأ : استخدم عدد ذي نقطة عائمة للتحكم في العروة ، فاستخدام الأعداد ذوات النقطة العائمة غالبا ما يصاحبه عدم دقة (imprecision) مما يتسبب عادة في حدوث عُرَى لا نهائية (infinite loops) .

التصحيح : استخدم عددا صحيحا في عروة **for** .

د) الخطأ : العروة متزايدة (incrementing) وليست متناقصة (decrementing) .

التصحيح :

```
for ( x = 999; x >= 1; x -= 2 )
    printf( "%d\n", x );
```

هـ) الأخطاء وتصحيحها :

- حرف D في كلمة **Do** يجب أن يكون حرفا صغيرا : **do** .
- حرف W في كلمة **While** يجب أن يكون حرفا صغيرا : **while** .
- المؤثر العلاقي > يجب أن يكون => حتى تتم طباعة 100 .

ملاحظة : الاختبار / الشرط في عبارة **if** لا ضرورة له ، لأن المتغير **counter** تتم زيادته دائما بـ 2 ، ولذلك سيظل دائما عددا زوجيا داخل جسم عبارة التكرار **do..while** ، أي أن شرط **if** سيكون دائما متحققا .

و) الخطأ : وجود فاصلة منقوطة في نهاية عبارة **for** .

التصحيح : حذف هذه الفاصلة المنقوطة بحيث تصبح العبارة total += x; في جسم العروة ، هكذا :

```
for ( x = 100; x <= 150; x++ ) /* ; removed */
   total += x;
```

٤ – ٤ – أ) 2, 4, 6, 8, 10, 12

ب) 5, 12, 19

ج) 3, 6, 9, 12, 15

د) 1

هـ) 12, 9, 6, 3

٤ – ٥ – أ)

```
for ( i = 1; i <= 7; i++ )
   printf( "%d ", i );
```

ب)

```
for ( i = 3; i <= 23; i += 5 )
   printf( "%d ", i );
```

ج)

```
for ( i = 20; i >= -10; i -= 6 )
   printf( "%d ", i );
```

د)

```
for ( i = 19; i <= 51; i += 8 )
   printf( "%d ", i );
```

٤–٦)

```
Enter integers in the range 1-20: 3 4
@@@
@@@
@@@
@@@
```

٤–٧–i) ج ii) ج iii) ب iv) أ

```
/* Exercise 4.8 Solution */
#include <stdio.h>

int main( void )
{
   int sum = 0; /* current sum */
   int number; /* number of values */
   int value;   /* current value */
   int i;       /* counter */

   /* display prompt */
   printf( "Enter the number of values"
           " to be processed: " );
   scanf( "%d", &number ); /* input number of values */

   /* loop number times */
   for ( i = 1; i <= number; i++ ) {
      printf( "Enter a value: " );
      scanf( "%d", &value );
      sum += value; /* add to sum */
   } /* end for */

   /* display sum */
   printf( "Sum of the %d values is %d\n", number, sum );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the number of values to be processed: 5
Enter a value: 10
Enter a value: 15
Enter a value: 20
Enter a value: 25
Enter a value: 30
Sum of the 5 values is 100
```

```
/* Exercise 4.9 Solution */
#include <stdio.h>

int main( void )
{
    int number; /* number of integers */
    int value; /* value input by user */
    int smallest; /* smallest number */
    int i; /* counter */

    /* prompt user for number of integers */
    printf( "Enter the number of integers to be processed: " );
    scanf( "%d", &number );

    /* prompt user for an integer */
    printf( "Enter an integer: " );
    scanf( "%d", &smallest );

    /* loop until user has entered all integers */
    for ( i = 2; i <= number; i++ ) {
        printf( "Enter next integer: " ); /* get next integer */
        scanf( "%d", &value );

        /* if value is smaller than smallest */
        if ( value < smallest ) {
            smallest = value;
        } /* end if */

    } /* end for */

    printf( "\nThe smallest integer is: %d\n", smallest );

    return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the number of integers to be processed: 5
Enter an integer: 372
Enter next integer: 920
Enter next integer: 73
```

```
Enter next integer: 8
Enter next integer: 3433

The smallest integer is: 8
```

(١٠ – ٤

```
/* Exercise 4.10 Solution */
#include <stdio.h>

int main( void )
{
    int i; /* counter */
    int sum = 0; /* current sum of integers */

    /* loop through even integers up to 30 */
    for ( i = 2; i <= 30; i += 2 ) {
        sum += i; /* add i to sum */
    } /* end for */

    printf( "Sum of the even integers from 2 to 30 is: %d\n", sum );

    return 0; /* indicate successful termination */

} /* end main */
```

```
Sum of the even integers from 2 to 30 is: 240
```

(١١ – ٤

```
/* Exercise 4.11 Solution */
#include <stdio.h>

int main( void )
{
    long i; /* counter */
    long product = 1; /* current product */

   /* loop through odd integers up to 15 */
   for ( i = 3; i <= 15; i += 2 ) {
       product *= i; /* update product */
   } /* end for */

    printf( "Product of the odd integers from 1 to 15 is: %ld\n", product );
```

```
    return 0; /* indicate successful termination */

} /* end main */
```

```
Product of the odd integers from 1 to 15 is: 2027025
```

(٤ – ١٢

```
/* Exercise 4.12 Solution */
#include <stdio.h>

int main( void )
{
    int i; /* outer counter */
    int j; /* inner counter */
    int factorial; /* current factorial value */

    printf( "X\tFactorial of X\n" ); /* display table headers */

    /* compute the factorial for 1 to 5 */
    for ( i = 1; i <= 5; i++ ) {
        factorial = 1;

        /* calculate factorial of current number */
        for ( j = 1; j <= i; j++ ) {
            factorial *= j;
      } /* end inner for */

      printf( "%d\t%d\n", i, factorial );
} /* end outer for */

return 0; /* indicate successful termination */

} /* end main */
```

```
X       Factorial of X
1       1
2       2
3       6
4       24
5       120
```

(٤ – ١٣

```
/* Exercise 4.13 Solution */
#include <stdio.h>
#include <math.h>

int main( void )
{
    int year; /* year counter */
    int rate; /* usury/interest rate */
   double amount; /* amount on deposit */
   double principal = 1000.0; /* starting principal */

    /* loop through usury/interest rates 5% to 10% */
    for ( rate = 5; rate <= 10; rate++ ) {

        /* display table headers */
        printf( "Usury/Interest Rate: %f\n", rate / 100.0 );
        printf( " %s%21s\n", "Year","Amount on deposit" );

        /* calculate amount on deposit for each of ten years */
        for ( year = 1; year <= 10; year++ ) {

            /* calculate new amount for specified year */
            amount = principal * pow( 1 + ( rate / 100.0 ), year );

            /* output one table row */
            printf( "%4d%21.2f\n", year, amount );
        } /* end for */

        printf( "\n" );
 } /* end for */

 return 0; /* indicate successful termination */

} /* end main */
```

```
Usury/Interest Rate:  0.050000
Year    Amount on deposit
    1              1050.00
    2              1102.50
    3              1157.63
    4              1215.51
```

```
    5              1276.28
    6              1340.10
    7              1407.10
    8              1477.46
    9              1551.33
   10              1628.89
...
Usury/Interest Rate: 0.080000
Year    Amount on deposit
    1              1080.00
    2              1166.40
    3              1259.71
    4              1360.49
    5              1469.33
    6              1586.87
    7              1713.82
    8              1850.93
    9              1999.00
   10              2158.92
...
Usury/Interest Rate: 0.100000
Year    Amount on deposit
    1              1100.00
    2              1210.00
    3              1331.00
    4              1464.10
    5              1610.51
    6              1771.56
    7              1948.72
    8              2143.59
    9              2357.95
   10              2593.74
```

(٤ – ١٤

```
/* Exercise 4.14 Solution */
#include <stdio.h>

int main( void )
{
    int row; /* row counter */
    int col; /* column counter */
    int space; /* spaces counter */
```

```
/* Pattern A, loop 10 times for rows */
for ( row = 1; row <= 10; row++ ) {

    /* print row asterisks */
    for ( col = 1; col <= row; col++ ) {
        printf( "*" );
    } /* end for */

    printf( "\n" );
 } /* end for */

 printf( "\n" );

 /* Pattern B, loop 10 times for rows
    row counts down to correspond to number of asterisks */
 for ( row = 10; row >= 1; row-- ) {

    /* print row asterisks */
    for ( col = 1; col <= row; col++ ) {
        printf( "*" );
    } /* end for */

    printf( "\n" );
} /* end for */

printf( "\n" );

/* Pattern C, loop 10 times for rows
   row counts down to correspond to number of asterisks */
for ( row = 10; row >= 1; row-- ) {

   /* print (10 - row) number of preceding spaces */
   for ( space = 1; space <= 10 - row; space++ ) {
       printf( " " );
   } /* end for */

   /* print row asterisks */
   for ( col = 1; col <= row; col++ ) {
       printf( "*" );

   } /* end for */
```

```
      printf( "\n" );
   } /* end for */

   printf( "\n" );

   /* Pattern D, loop 10 times for rows */
   for ( row = 1; row <= 10; row++ ) {

      /* print (10 - row) number of preceding spaces */
      for ( space = 1; space <= 10 - row; space++ ) {
         printf( " " );
      } /* end for */

      /* print row asterisks */
      for ( col = 1; col <= row; col++ ) {
         printf( "*" );
      } /* end for */

      printf( "\n" );
   } /* end for */

   printf( "\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

(٤ – ١٥

```
/* Exercise 4.15 Solution */
#include <stdio.h>

int main( void )
{
   int i; /* outer counter */
   int j; /* inner counter */
   int number; /* current number */

   printf( "Enter 5 numbers between 1 and 30: " );

   /* loop 5 times */
   for ( i = 1; i <= 5; i++ ) {
```

```
      scanf( "%d", &number );

       /* print asterisks corresponding to current input */
      for ( j = 1; j <= number; j++ ) {
         printf( "*" );
      } /* end for */

      printf( "\n" );
   } /* end for */

    return 0; /* indicate successful termination */

} /* end main */
```

```
Enter 5 numbers between 1 and 30: 28 5 13 24 7
****************************
*****
**********
************************
*******
```

(١٦ – ٤

```
/* Exercise 4.16 Solution */
#include <stdio.h>

 int main( void )
{
    int product; /* current product number */
    int quantity; /* quantity of current product sold */
   double total = 0.0; /* current total retail value */

    /* prompt for input */
    printf( "Enter pairs of item numbers and quantities.\n");
    printf( "Enter -1 for the item number to end input.\n" );
    scanf( "%d", &product );

    /* loop while sentinel value not read from user */
    while ( product != -1 ) {
        scanf( "%d", &quantity );

        /* determine product number and corresponding retail price */
```

```
        switch ( product ) {

            case 1:
                total += quantity * 2.98; /* update total */
                break;

            case 2:
                total += quantity * 4.50; /* update total */
                break;

            case 3:
                total += quantity * 9.98; /* update total */
                break;

            case 4:
                total += quantity * 4.49; /* update total */
                break;

            case 5:
                total += quantity * 6.87; /* update total */
                break;

            default:
                printf( "Invalid product code: %d\n", product );
                printf( "           Quantity: %d\n", quantity );
        } /* end switch */

        scanf( "%d", &product ); /* get next input */
    } /* end while */

    /* display total retail value */
    printf( "The total retail value was: %.2f\n", total );

    return 0; /* indicate successful termination */

} /* end main */
```

```
Enter pairs of item numbers and quantities.
Enter -1 for the item number to end input.
1 1
2 1
```

```
3 1
4 1
5 1
6 1
Invalid product code: 6
                Quantity: 1
1 1
-1
The total retail value was: 31.80
```

(٤ – ١٧

```
/* Exercise 4.17 Solution */
#include <stdio.h>

int main( void )
{
   int counter = 1; /* initialize counter */

   /* leave first statement empty */
   for ( ; counter <= 10; counter++ ) {
      printf( "%d\n", counter );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */
```

```
1
2
3
4
5
6
7
8
9
10
```

```
/* Exercise 4.18 Solution */
#include <stdio.h>

int main( void )
{
    int grade; /* current grade */
    int aCount = 0; /* total a grades */
    int bCount = 0; /* total b grades */
    int cCount = 0; /* total c grades */
    int dCount = 0; /* total d grades */
    int fCount = 0; /* total f grades */
   double averageGrade; /* average grade for class */

    /* prompt user for grades */
    printf( "Enter the letter grades.\n");
    printf( "Enter the EOF character to end input.\n" );

    /* loop while not end of file */
    while ( ( grade = getchar() ) != EOF ) {

        /* determine which grade was input */
        switch ( grade ) {

            case 'A': /* grade was uppercase A */
            case 'a': /* grade was lowercase a */
                ++aCount; /* update grade A counter */
                break; /* exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* grade was lowercase b */
                ++bCount; /* update grade B counter */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* grade was lowercase c */
                ++cCount; /* update grade C counter */
                break; /* exit switch */

            case 'D': /* grade was uppercase C */
            case 'd': /* grade was lowercase c */
```

```
                ++dCount; /* update grade C counter */
                break; /* exit switch */

            case 'F': /* grade was uppercase C */
            case 'f': /* grade was lowercase c */
                ++fCount; /* update grade C counter */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional, will exit switch anyway */
   } /* end switch */

} /* end while */

/* output totals for each grade */
printf( "\nThe totals for each letter grade are:\n" );
printf( "A: %d\n", aCount );
printf( "B: %d\n", bCount );
printf( "C: %d\n", cCount );
printf( "D: %d\n", dCount );
printf( "F: %d\n", fCount );

/* calculate average grade */
averageGrade =
    ( 4 * aCount + 3 * bCount + 2 * cCount + dCount ) /
    ( aCount + bCount + cCount + dCount + fCount );

/* output appropriate message for average grade */
if ( averageGrade > 3.4 ) {
   printf( "Average grade is A\n" );
} /* end if */
else if ( averageGrade > 2.4 ) {
    printf( "Average grade is B\n" );
} /* end else if */
else if ( averageGrade > 1.4 ) {
```

```
    printf( "Average grade is C\n" );
 } /* end else if */
 else if ( averageGrade > 0.4 ) {
    printf( "Average grade is D\n" );
 } /* end else if */
 else {
    printf( "Average grade is F\n" );
 } /* end else */

  return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the letter grades.
Enter the EOF character to end input.
A B B B C D F
C C C D D D C B
B A A B C C C
^Z

The totals for each letter grade are:
A: 3
B: 6
C: 8
D: 4
F: 1
Average grade is C
```

٤ – ١٩) أ) 1 ب) 0 ج) 1 د) 0

هـ) 1 و) 0 ز) 0 ح) 1

ط) 0 ي) 1

٤ – ٢٠)

```
/* Exercise 4.20 Solution */
#include<stdio.h>

int main( void )
{
   long double pi = 0.0; /* approximated value for pi */
```

```
   long double num = 4.0; /* numerator */
   long double denom = 1.0; /* denominator of current term */
   long int loop; /* loop counter */
   long int accuracy; /* number of terms */

   accuracy = 400000; /* set decimal accuracy */

   /* display table headers */
   printf( "Accuracy set at: %ld\n", accuracy );
   printf( "term\t\t pi\n" );

   /* loop through each term */
   for ( loop = 1; loop <= accuracy; loop++ ) {

      /* if odd-numbered term, add current term */
      if ( loop % 2 != 0 ) {
         pi += num / denom;
      } /* end if */
      else { /* if even-numbered term, subtract current term */
          pi -= num / denom;
      } /* end else */

      /* display number of terms and approximated
          value for pi with 8 digits of precision */
      printf( "%ld\t\t%Lf\n", loop, pi );

      denom += 2.0; /* update denominator */

   } /* end for */

   return 0; /* indicate successful termination */

   } /* end main */
```

```
Accuracy set at: 400000
term                pi
1                   4.000000
2                   2.666667
3                   3.466667
4                   2.895238
5                   3.339683
```

```
6               2.976046
...

995             3.142598
996             3.140589
997             3.142596
998             3.140591
999             3.142594
...

399998          3.141590
399999          3.141595
400000          3.141590
```

(٤ – ٢١

```
/* Exercise 4.21 Solution */
#include<stdio.h>

 int main( void )
{
    int count = 0; /* number of triples found */
    long int side1; /* side1 value counter */
    long int side2; /* side2 value counter */
    long int hypotenuse; /* hypotenuse value counter */
    long int hyptSquared; /* hypotenuse squared */
    long int sidesSquared; /* sum of squares of sides */

    /* side1 values range from 1 to 500 */
    for ( side1 = 1; side1 <= 500; side1++ ) {

        /* side2 values range from 1 to 500 */
        for ( side2 = 1; side2 <= 500; side2++ ) {

            /* hypotenuse values range from 1 to 500 */
           for ( hypotenuse = 1; hypotenuse <= 500; hypotenuse++ ) {

               /* calculate square of hypotenuse value */
               hyptSquared = hypotenuse * hypotenuse;

               /* calculate sum of squares of sides */
               sidesSquared = side1 * side1 + side2 * side2;
```

```
            /* if hypotenuse squared = side1 squared + side2 squared,
               Pythagorean triple */
            if ( hyptSquared == sidesSquared ) {

               /* display triple */
               printf( "%ld %ld %ld\n", side1, side2, hypotenuse );
               ++count; /* update count */

            } /* end if */

         } /* end for */

      } /* end for */

   } /* end for */

   /* display total number of triples found */
   printf( "A total of %d triples were found.\n", count );

   return 0; /* indicate successful termination */

} /* end main */
```

```
3 4 5
4 3 5
5 12 13
6 8 10
7 24 25
8 6 10
...
476  93  485
480  31  481
480  88  488
480  108  492
480  140  500
483  44  485
A total of 772 triples were found.
```

(٤ – ٢٢

```
/* Exercise 4.22 Solution */
#include<stdio.h>

 int main( void )
{
     int payCode; /* current employee's pay code */
     int managers = 0; /* total number of managers */
     int hWorkers = 0; /* total number of hourly workers */
     int cWorkers = 0; /* total number of commission workers */
     int pWorkers = 0; /* total number of pieceworkers */
     int pieces; /* current pieceworker's number of pieces */
    double mSalary; /* manager's salary */
    double hSalary; /* hourly worker's salary */
    double cSalary; /* commission worker's salary */
    double pSalary; /* pieceworker's salary */
    double hours; /* total hours worked */
    double otPay; /* overtime pay */
    double otHours; /* overtime hours */
    double pay; /* current employee's weekly pay */

    /* prompt for first employee input */
    printf( "Enter paycode ( -1 to end): " );
    scanf( "%d", &payCode );

    /* loop while sentinel value not read from user */
    while ( payCode != -1 ) {

        /* switch to appropriate computation according to pay code */
        switch ( payCode ) {

            /* pay code 1 corresponds to manager */
            case 1:

                /*prompt for weekly salary */
                printf( "Manager selected.\n" );
                printf( "Enter weekly salary: " );
                scanf( "%lf", &mSalary );

                /* manager's pay is weekly salary */
                printf( "The manager's pay is $%.2f\n", mSalary );

                ++managers; /* update total number of managers */
```

```
      break; /* exit switch */

/* pay code 2 corresponds to hourly worker */
case 2:

      /* prompt for hourly salary */
      printf( "Hourly worker selected.\n" );
      printf( "Enter the hourly salary: " );
      scanf( "%lf", &hSalary );

      /* prompt for number of hours worked */
      printf( "Enter the total hours worked: " );
      scanf( "%lf", &hours );

      /* pay fixed for up to 40 hours, 1.5 for hours over 40 */
      if ( hours > 40.0 ) {

         /* calculate OT hours and total pay */
         otHours = hours - 40.0;
         otPay = hSalary * 1.5 * otHours + hSalary * 40.0;

      printf( "Worker has worked %.1f overtime hours.\n", otHours );
      printf( "Workers pay is $%.2f\n", otPay );
      } /* end if */
      else { /* no overtime */
          pay = hSalary * hours;
          printf( "Worker's pay is $%.2f\n", pay );
      } /* end else */

      ++hWorkers; /* update total number of hourly workers */
      break; /* exit switch */

 /* pay code 3 corresponds to commission worker */
 case 3:

      /* prompt for gross weekly sales */
      printf( "Commission worker selected.\n" );
      printf( "Enter gross weekly sales: " );
      scanf( "%lf", &cSalary );

      /* pay $250 plus 5.7% of gross weekly sales */
      pay = 250.0 + 0.057 * cSalary;
```

```
            printf( "Commission Worker's pay is $%.2f\n", pay );

            ++cWorkers; /* update total number of commission workers */
            break; /* exit switch */

         /* pay code 4 corresponds to pieceworker */
         case 4:

            /* prompt for number of pieces */
            printf( "Piece worker selected.\nEnter number of pieces: " );
            scanf( "%d", &pieces );

            /* prompt for wage per piece */
            printf( "Enter wage per piece: " );
            scanf( "%lf", &pSalary );

            pay = pieces * pSalary; /* compute pay */
            printf( "Piece Worker's pay is $%.2f\n", pay );

            ++pWorkers; /* update total number of pieceworkers */
            break; /* exit switch */

         /* default case */
         default :
            printf( "Invalid pay code.\n" );
            break;
         } /* end switch */

         /* prompt for next employee input */
         printf( "\nEnter paycode ( -1 to end ): " );
         scanf( "%d", &payCode );
} /* end while */

/* display total counts for each type of employee */
printf( "\n" );
printf( "Total number of managers paid : %d\n", managers );
printf( "Total number of hourly workers paid : %d\n", hWorkers );
printf( "Total number of commission workers paid: %d\n", cWorkers );
printf( "Total number of piece workers paid : %d\n", pWorkers );

return 0; /* indicate successful termination */
```

```
} /* end main */
```

```
Enter paycode ( -1 to end): 4
Piece worker selected.
Enter number of pieces: 200
Enter wage per piece: 20
Piece Worker's pay is $4000.00

Enter paycode ( -1 to end ): -1

Total number of managers paid : 0
Total number of hourly workers paid : 0
Total number of commission workers paid: 0
Total number of piece workers paid : 1
```

```
Enter paycode ( -1 to end): 1
Manager selected.
Enter weekly salary: 2500
The manager's pay is $2500.00

Enter paycode ( -1 to end ): 2
Hourly worker selected.
Enter the hourly salary: 10.50
Enter the total hours worked: 75
Worker has worked 35.0 overtime hours.
Workers pay is $971.25

Enter paycode ( -1 to end ): 3
Commission worker selected.
Enter gross weekly sales: 9000
Commission Worker's pay is $763.00

Enter paycode ( -1 to end ): 4
Piece worker selected.
Enter number of pieces: 200
Enter wage per piece: 20
Piece Worker's pay is $4000.00

Enter paycode ( -1 to end ): -1

Total number of managers paid : 1
Total number of hourly workers paid : 1
Total number of commission workers paid: 1
```

```
Total number of piece workers paid : 1
```

(٤ – ٢٣

```
/* Exercise 4.23 Solution */
#include <stdio.h>

int main( void )
{
    int line; /* line counter */
    int space; /* space counter */
    int asterisk; /* asterisk counter */

    /* top half */
    for ( line = 1; line <= 9; line += 2 ) {

        /* print preceding spaces */
        for ( space = ( 9 - line ) / 2; space > 0; space-- ) {
            printf( " " );
        } /* end for */

        /* print asterisks */
        for ( asterisk = 1; asterisk <= line; asterisk++ ) {
            printf( "*" );
        } /* end for */

        printf( "\n" );
    } /* end for */

    /* bottom half */
    for ( line = 7; line >= 0; line -= 2 ) {

        /* print preceding spaces */
        for ( space = ( 9 - line ) / 2; space > 0; space-- ) {
            printf( " " );
        } /* end for */

        /* print asterisks */
        for ( asterisk = 1; asterisk <= line; asterisk++ ) {
            printf( "*" );
        } /* end for */

        printf( "\n" );
```

```
    } /* end for */

    return 0; /* indicate successful termination */

} /* end main */
```

(٤ – ٢٤

```
/* Exercise 4.24 Solution */
#include<stdio.h>

int main( void )
{
    int line; /* line counter */
    int space; /* space counter */
    int asterisk; /* asterisk counter */
    int size; /* number of rows in diamond */

    /* prompt for diamond size */
    printf( "Enter an odd number for the diamond size ( 1-19 ):\n" );
    scanf( "%d", &size );

    /* top half */
    for ( line = 1; line <= size - 2; line += 2 ) {

        /* print preceding spaces */
        for ( space = ( size - line ) / 2; space > 0; space-- ) {
            printf( " " );
        } /* end for */

        /* print asterisks */
        for ( asterisk = 1; asterisk <= line; asterisk++ ) {
            printf( "*" );
        } /* end for */

        printf( "\n" );
    } /* end for */

    /* bottom half */
    for ( line = size; line >= 0; line -= 2 ) {

        /* print preceding spaces */
        for ( space = ( size - line ) / 2; space > 0; space-- ) {
            printf( " " );
```

```
        } /* end for */

        /* print asterisks */
        for ( asterisk = 1; asterisk <= line; asterisk++ ) {
            printf( "*" );
        } /* end for */

            printf( "\n" );
        } /* end for */

        return 0; /* indicate successful termination */

} /* end main */
```

```
Enter an odd number for the diamond size ( 1-19 ):
13
        *
       ***
      *****
     *******
    *********
   ***********
  *************
 ***********
  *********
   *******
    *****
     ***
      *
```

٤ – ٢٥ – أ) جسم عروة **do…while** يصبح هو جسم عروة **while** [والشرط في نهاية عروة **do…while** يصبح هو نفسه الشرط في بداية عروة **while**] . ومحتويات الجسم تكرَّر قبل عروة **while** (لأن الجسم في عروة **do…while** يجب أن يُنفَّذ مرة واحدة على الأقل ، بينما تنفيذ الجسم في عروة **while** يعتمد على شرط الاستمرارية) .

ب) المشكلة هي [التي أشرنا إليها في الجزء أ)] أن عروة **while** (المعطاة) قد لا تنفَّذ إطلاقا (بناءً على شرط الاستمرارية) ، بينما عروة **do…while** (المكافئة المطلوبة) لا بد أن تنفَّذ مرة واحدة على الأقل ، ولذلك فإننا نحتاج قبل عروة **do…while** إلى عبارة **if** للاختيار ، بحيث تكون عروة **do…while** هي جسم عبارة **if** ، وشرط **if** هو نفسه شرط استمرارية عروة **do…while**.

(٤ – ٢٦)

```
****
****
****

****
****
****

****
****
****

****
****
****

****
****
****
```

(٤ – ٢٧ – أ)

```
/* Exercise 4.27.a Solution */
#include <stdio.h>

int main( void )
{
   int x; /* loop counter */
   int breakOut = 1; /* breakout condition */

   /* test for breakout condition */
   for ( x = 1; x <= 10 && breakOut == 1; x++ ) {

      /* break out of loop after x = 4 */
      if ( x == 4 ) {
```

```
            breakOut = -1;
         } /* end if */

         printf( "%d ", x );
      } /* end for */

      printf( "\nBroke out of loop at x = %d\n", x );

      return 0; /* indicate successful termination */

   } /* end main */
```

```
1 2 3 4
Broke out of loop at x = 5
```

(ب

```
/* Exercise 4.40 Solution */
#include <stdio.h>

int main( void )
{
   int x; /* loop counter */

   /* loop 10 times */
   for ( x = 1; x <= 10; x++ ) {

      /* if x == 5, skip to next interation */
      if ( x == 5 ) {
         ++x;
      } /* end if */

      printf( "%d ", x );
   } /* end for */

   printf( "\nUsed ++x to skip printing the value 5\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

```
1 2 3 4 6 7 8 9 10
Used ++x to skip printing the value 5
```

أجوبة تمرينات رقم ٥

٥ – ١ – i) د
ii) د
iii) ب
iv) ب
v) ج
vi) ج
vii) أ
viii) د
ix) ب
x)
(١) ج
(٢) ج
(٣) ب
(٤) ب
(٥) ب
(٦) د

٥ – ٢)

```
/* Exercise5.2 */
/* Testing the math library functions */
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main()
{
   /* calculates and outputs the square root */
   printf( "sqrt(%.1f) = %.1f\n", 900.0, sqrt( 900.0 ) );
   printf( "sqrt(%.1f) = %.1f\n", 9.0, sqrt( 9.0 ) );

   /* calculates and outputs the exponential function e to the x */
   printf( "exp(%.1f) = %f\n", 1.0, exp( 1.0 ) );
   printf( "exp(%.1f) = %f\n", 2.0, exp( 2.0 ) );

   /* calculates and outputs the logorithm (base e) */
   printf( "log(%f) = %.1f\n", 2.718282, log( 2.718282 ) );
   printf( "log(%f) = %.1f\n", 7.389056, log( 7.389056 ) );

   /* calculates and outputs the logorithm (base 10) */
   printf( "log10(%.1f) = %.1f\n", 1.0, log10( 1.0 ) );
```

```
   printf( "log10(%.1f) = %.1f\n", 10.0, log10( 10.0 ) );
   printf( "log10(%.1f) = %.1f\n", 100.0, log10( 100.0 ) );

   /* calculates and outputs the absolute value */
   printf( "fabs(%.1f) = %.1f\n", 13.5, fabs( 13.5 ) );
   printf( "fabs(%.1f) = %.1f\n", 0.0, fabs( 0.0 ) );
   printf( "fabs(%.1f) = %.1f\n", -13.5, fabs( -13.5 ) );

   /* calculates and outputs ceil( x ) */
   printf( "ceil(%.1f) = %.1f\n", 9.2, ceil( 9.2 ) );
   printf( "ceil(%.1f) = %.1f\n", -9.8, ceil( -9.8 ) );

   /* calculates and outputs floor( x ) */
   printf( "floor(%.1f) = %.1f\n", 9.2, floor( 9.2 ) );
   printf( "floor(%.1f) = %.1f\n", -9.8, floor( -9.8 ) );

   /* calculates and outputs pow( x, y ) */
   printf( "pow(%.1f, %.1f) = %.1f\n", 2.0, 7.0, pow( 2.0, 7.0 ) );
   printf( "pow(%.1f, %.1f) = %.1f\n", 9.0, 0.5, pow( 9.0, 0.5 ) );

   /* calculates and outputs fmod( x, y ) */
   printf( "fmod(%.3f/%.3f) = %.3f\n", 13.675, 2.333,
          fmod( 13.675, 2.333 ) );

   /* calculates and outputs sin( x ) */
   printf( "sin(%.1f) = %.1f\n", 0.0, sin( 0.0 ) );

   /* calculates and outputs cos( x ) */
   printf( "cos(%.1f) = %.1f\n", 0.0, cos( 0.0 ) );

   /* calculates and outputs tan( x ) */
   printf( "tan(%.1f) = %.1f\n", 0.0, tan( 0.0 ) );

   return 0; /* indicates successful termination */

} /* end main */
```

```
sqrt (900.0) = 30.0
sqrt (9.0) = 3.0
exp(1.0) = 2.718282
```

```
exp(2.0) = 7.389056
log(2.718282) = 1.0
log(7.389056) = 2.0
log10(1.0) = 0.0
log10(10.0) = 1.0
log10(100.0) = 2.0
fabs(13.5) = 13.5
fabs(0.0) = 0.0
fabs(-13.5) = 13.5
ceil(9.2) = 10.0
ceil(-9.8) = -9.0
floor(9.2) = 9.0
floor(-9.8) = -10.0
pow(2.0, 7.0) = 128.0
pow(9.0, 0.5) = 3.0
fmod(13.675/2.333) = 2.010
sin(0.0) = 0.0
cos(0.0) = 1.0
tan(0.0) = 0.0
```

٥ – ٣ – أ) `double hypotenuse( double side1, double side2 )`

ب) `int smallest( int x, int y, int z )`

ج) `void instructions( void )`

د) `float intToFloat( int number )`

٥ – ٤ – أ) `double hypotenuse( double side1, double side2 );`

ب) `int smallest( int x, int y, int z );`

ج) `void instructions( void );`

د) `float intToFloat( int number );`

٥ – ٥ – أ) `register int count = 0;`

ب) `static float lastVal;`

ج) `static int number;`

(ملاحظة : هذا سيظهر خارج تعريف أي دالة)

٥ – ٦ – أ) الخطأ : الدالة h معرَّفة داخل الدالة g .

التصحيح : انقل تعريف الدالة **h** إلى خارج تعريف الدالة **g** .

ب) الخطأ : المفروض أن جسم الدالة (body of the function) يعيد عددا صحيحا ، ولكنه لا يقوم بذلك .

التصحيح : أضف العبارة التالية في الدالة :

return result;

ج) الخطأ الأول : وجود فاصلة منقوطة بعد القوس الأيمن الذي يحيط بقائمة الوسطاء .

تصحيحه : احذف هذه الفاصلة المنقوطة .

الخطأ الثاني : إعادة تعريف الوسيط **a** (parameter) في تعريف الدالة .

تصحيحه : احذف الإعلان float a;

الموجود في جسم الدالة .

د) الخطأ : الدالة تعيد قيمة بينما المفروض ألا تعيد قيمة .

التصحيح : احذف عبارة return .

٥-٧- أ) 7.5 ب) 7.0 ج) 0.0

د) 0.0 هـ) 6.4 و) -6.0

ز) -14.0

٥-٨)

```
/* Exercise 5.8 Solution */
 #include <stdio.h>
 #include <math.h>

void calculateFloor( void ); /* function prototype */

int main()
{
  calculateFloor(); /* call function calculateFloor */

  return 0; /* indicate successful termination */

 } /* end main */
```

```
/* calculateFloor rounds 5 inputs */
void calculateFloor( void )
{
  double x; /* current input */
  double y; /* current input rounded */
  int loop; /* loop counter */

  /* loop for 5 inputs */
  for ( loop = 1; loop <= 5; loop++ ) {
     printf( "Enter a floating point value: " );
     scanf( "%lf", &x );

     /* y holds rounded input */
    y = floor( x + .5 );
    printf( "%f rounded is %.1f\n\n", x, y );
  } /* end for */

} /* end function calculateFloor */
```

```
Enter a floating point value: 1.5
1.500000 rounded is 2.0

Enter a floating point value: 5.55
5.550000 rounded is 6.0

Enter a floating point value: 73.2341231432
73.234123 rounded is 73.0

Enter a floating point value: 9.0
9.000000 rounded is 9.0

Enter a floating point value: 4
4.000000 rounded is 4.0
```

(٥ – ٩

```
/* Exercise 5.9 Solution */
#include <stdio.h>
#include <math.h>
```

```
double roundToInteger( double n ); /* function prototype */
double roundToTenths( double n ); /* function prototype */
double roundToHundredths( double n ); /* function prototype */
double roundToThousandths( double n ); /* function prototype */

int main()
{
   int i; /* loop counter */
   int count; /* number of values to process */
   double number; /* current input */

   printf( "How many numbers do you want to process? " );
   scanf( "%d", &count );

   /* loop for inputs */
   for ( i = 0; i < count; i++ ) {
      printf( "Enter number: " );
      scanf( "%lf", &number );

      /* display number rounded to nearest integer */
      printf( "%f rounded to an integer is %f\n",
            number, roundToInteger( number ) );

      /* display number rounded to nearest tenth */
      printf( "%f rounded to the nearest tenth is %f\n",
            number, roundToTenths( number ) );

      /* display number rounded to nearest hundredth */
      printf( "%f rounded to the nearest hundredth is %f\n",
            number, roundToHundredths( number ) );

      /* display number rounded to nearest thousandth */
      printf( "%f rounded to the nearest thousandth is %f\n\n",
            number, roundToThousandths( number ) );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */
```

```
/* roundToInteger rounds n to nearest integer */
double roundToInteger( double n )
{
  return floor( n + .5 );

} /* end function roundToInteger */

/* roundToTenths rounds n to nearest tenth */
double roundToTenths( double n )
{
  return floor( n * 10 + .5 ) / 10;

} /* end function roundToTenths */

/* roundToHundredths rounds n to nearest hundredth */
double roundToHundredths( double n )
{
  return floor( n * 100 + .5 ) / 100;

} /* end function roundToHundredths */

/* roundToThousandths rounds n to nearest thousandth */
double roundToThousandths( double n )
{
  return floor( n * 1000 + .5 ) / 1000;

} /* end function roundToThousandths */
```

```
How many numbers do you want to process? 1
Enter number: 8.54739
8.547390 rounded to an integer is 9.000000
8.547390 rounded to the nearest tenth is 8.500000
8.547390 rounded to the nearest hundredth is 8.550000
8.547390 rounded to the nearest thousandth is 8.547000
```

٥-١٠- أ)
```
n = 1 + rand() % 2;
```
ب)
```
n = 1 + rand() % 100;
```
ج)
```
n = rand() % 10;
```

د) n = 1000 + rand() % 113;

هـ) n = -1 + rand() % 3;

و) n = -3 + rand() % 15;

٥-١١- أ) printf(“%d\n”, 2 * (1 + rand() % 5));

ب) printf(“%d\n”, 1 + 2 * (1 + rand() % 5));

ج) pritnf(“%d\n”, 6 + 4 * (rand() % 5));

(٥-١٢

```
/* Exercise 5.12 Solution */
#include <stdio.h>
#include <math.h>

double hypotenuse( double s1, double s2 ); /* function prototype */

int main()
{
  int i; /* loop counter */
  double side1; /* value for first side */
  double side2; /* value for second side */

  /* loop 3 times */
  for ( i = 1; i <= 3; i++ ) {
      printf( "Enter the sides of the triangle: " );
      scanf( "%lf%lf", &side1, &side2 );

      /* calculate and display hypotenuse value */
      printf( "Hypotenuse: %.1f\n\n", hypotenuse( side1, side2 ) );
  } /* end for */

  return 0; /* indicate successful termination */

} /* end main */

/* hypotenuse calculates value of hypotenuse of
  a right triangle given two side values */
double hypotenuse( double s1, double s2 )
{
```

```
   return sqrt( pow( s1, 2 ) + pow( s2, 2 ) );

} /* end function hypotenuse */
```

```
Enter the sides of the triangle: 3.0 4.0
Hypotenuse: 5.0

Enter the sides of the triangle: 5.0 12.0
Hypotenuse: 13.0

Enter the sides of the triangle: 8.0 15.0
Hypotenuse: 17.0
```

(٥–١٣

```
/* Exercise 5.13 Solution */
#include <stdio.h>

int integerPower( int b, int e );

int main()
{
   int exp; /* integer exponent */
   int base; /* integer base */

   printf( "Enter integer base and exponent: " );
   scanf( "%d%d", &base, &exp );

   printf( "%d to the power %d is: %d\n",
           base, exp, integerPower( base, exp ) );

   return 0; /* indicate successful termination */

} /* end main */

/* integerPower calculates and returns b raised to the e power */
int integerPower( int b, int e )
{
   int product = 1; /* resulting product */
   int i; /* loop counter */
```

```
   /* multiply product times b (e repetitions) */
   for ( i = 1; i <= e; i++ ) {
      product *= b;
   } /* end for */

   return product; /* return resulting product */

} /* end function integerPower */
```

```
Enter integer base and exponent: 5 3
5 to the power 3 is: 125
```

(١٤–٥

```
/* Exercise 5.14 Solution */
#include <stdio.h>

int multiple( int a, int b ); /* function prototype */

int main()
{
   int x; /* first integer */
   int y; /* second integer */
   int i; /* loop counter */

   /* loop 3 times */
   for ( i = 1; i <= 3; i++ ) {
      printf( "Enter two integers: " );
      scanf( "%d%d", &x, &y );

      /* determine if second is multiple of first */
      if ( multiple( x, y ) ) {
         printf( "%d is a multiple of %d\n\n", y, x );
      } /* end if */
      else {
         printf( "%d is not a multiple of %d\n\n", y, x );
      } /* end else */

   } /* end for */
```

```
   return 0; /* indicate successful termination */

} /* end main */

/* multiple determines if b is multiple of a */
int multiple( int a, int b )
{
   return !( b % a );

} /* end function multiple */
```

```
Enter two integers: 2 10
10 is a multiple of 2

Enter two integers: 5 17
17 is not a multiple of 5

Enter two integers: 3 696
696 is a multiple of 3
```

(٥–١٥

```
Exercise 5.15 Solution */
#include <stdio.h>

int even( int a ); /* function prototype */

int main()
{
   int x; /* current input */
   int i; /* loop counter */

   /* loop for 3 inputs */
   for ( i = 1; i <= 3; i++ ) {
      printf( "Enter an integer: " );
      scanf( "%d", &x );

      /* determine if input is even */
      if ( even( x ) ) {
         printf( "%d is an even integer\n\n", x );
```

```
      } /* end if */
      else {
         printf( "%d is not an even integer\n\n", x );
      } /* end else */

   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */

/* even returns true if a is even */
int even( int a )
{
   return !( a % 2 );

} /* end function even */
```

```
Enter an integer: 7
7 is not an even integer

Enter an integer: 6
6 is an even integer

Enter an integer: 10000
10000 is an even integer
```

(١٦–٥

```
/* Exercise 5.16 Solution */
#include <stdio.h>

void square( int s ); /* function prototype */

int main()
{
   int side; /* input side length */

   printf( "Enter side: " );
   scanf( "%d", &side );
```

```
   square( side ); /* display solid square of asterisks */

return 0; /* indicate successful termination */

} /* end main */

/* square displays solid square of asterisks with specified side */
void square( int s )
{
   int i; /* outer loop counter */
   int j; /* inner loop counter */

   /* loop side times for number of rows */
   for ( i = 1; i <= s; i++ ) {

      /* loop side times for number of columns */
      for ( j = 1; j <= s; j++ ) {
         printf( "*" );
      } /* end for */

      printf( "\n" );
   } /* end for */

} /* end function square */
```

(١٧–٥

```
/* Exercise 5.17 Solution */
#include <stdio.h>

void square( int side, char fillCharacter ); /* function prototype */

int main()
{
   int s; /* side length */
   char c; /* fill character */

   printf( "Enter a character and the side length: " );
   scanf( "%c%d", &c, &s );

   square( s, c ); /* display solid square of input character */
```

```
   return 0; /* indicate successful termination */

} /* end main */

/* square displays solid square of fillCharacter with specified side */
void square( int side, char fillCharacter )
{
     int loop; /* outer loop counter */
     int loop2; /* inner loop counter */

     /* loop side times for number of rows */
     for ( loop = 1; loop <= side; loop++ ) {

        /* loop side times for number of columns */
        for ( loop2 = 1; loop2 <= side; loop2++ ) {
        printf( "%c", fillCharacter );
        } /* end for */

        printf( "\n" );
     } /* end for */

} /* end function square */
```

(٥ – ١٨

```
/* Exercise 5.18 Solution */
#include <stdio.h>

int celcius( int fTemp ); /* function prototype */
int fahrenheit( int cTemp ); /* function prototype */

int main()
{
   int i; /* loop counter */

   /* display table of Fahrenheit equivalents of Celsius temperature */
   printf( "Fahrenheit equivalents of Celcius temperatures:\n" );
   printf( "Celcius\t\tFahrenheit\n" );

   /* display Fahrenheit equivalents of Celsius 0 to 100 */
   for ( i = 0; i <= 100; i++ ) {
      printf( "%d\t\t%d\n", i, fahrenheit( i ) );
```

```
   } /* end for */

   /* display table of Celsius equivalents of Fahrenheit temperature */
   printf( "\nCelcius equivalents of Fahrenheit temperatures:\n" );
   printf( "Fahrenheit\tCelcius\n" );

   /* display Celsius equivalents of Fahrenheit 32 to 212 */
   for ( i = 32; i <= 212; i++ ) {
      printf( "%d\t\t%d\n", i, celcius( i ) );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */

/* celsius returns Celsius equivalent of fTemp,
given in Fahrenheit */
int celcius( int fTemp )
{
   return ( int ) ( 5.0 / 9.0 * ( fTemp - 32 ) );

} /* end function celsius */

/* fahrenheit returns Fahrenheit equivalent of cTemp,
   given in Celsius */
int fahrenheit( int cTemp )
{
   return ( int ) ( 9.0 / 5.0 * cTemp + 32 );

} /* end function fahrenheit */
```

```
Fahrenheit equivalents of Celcius temperatures:
Celcius          Fahrenheit
0                32
1                33
2                35
3                37
4                39
5                41
6                42
```

```
7               44
8               46
9               48
.
.
.
Celcius equivalents of Fahrenheit temperatures:
Fahrenheit      Celcius
32              0
33              0
34              1
35              1
36              2
37              2
38              3
39              3
40              4
41              5
.
.
.
```

(٥ – ١٩

```
/* Exercise 5.19 Solution */
#include <stdio.h>

/* function prototype */
double smallest3( double a, double b, double c );

int main()
{
   double x; /* first input */
   double y; /* second input */
   double z; /* third input */

   printf( "Enter three doubleing point values: " );
   scanf( "%lf%lf%lf", &x, &y, &z );

   /* determine smallest value */
```

```
   printf( "The smallest value is %f\n", smallest3( x, y, z ) );

   return 0; /* indicate successful termination */

} /* end main */

/* smallest3 returns the smallest of a, b and c */
double smallest3( double a, double b, double c )
{
   double smallest = a; /* assume a is the smallest */

   if ( b < smallest ) { /* if b is smaller */
      smallest = b;
   } /* end if */

   if ( c < smallest ) { /* if c is smaller */
      smallest = c;
   } /* end if */

   return smallest; /* return smallest value */

} /* end function smallest3 */
```

```
Enter three doubleing point values: 3.3 4.4 5.5
The smallest value is 3.300000
```

```
Enter three doubleing point values: 4.4 5.5 3.3
The smallest value is 3.300000
```

```
Enter three doubleing point values: 4.4 3.3 5.5
The smallest value is 3.300000
```

(٢٠ – ٥

```
/* Exercise 5.20 Solution */
#include <stdio.h>
```

```
int perfect( int value ); /* function prototype */

int main()
{
   int j; /* loop counter */

   printf( "For the integers from 1 to 1000:\n" );

   /* loop from 2 to 1000 */
   for ( j = 2; j <= 1000; j++ ) {

      /* if current integer is perfect */
      if ( perfect( j ) ) {
         printf( "%d is perfect\n", j );
      } /* end if */

   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */

/* perfect returns true if value is perfect integer,
   i.e., if value is equal to sum of its factors */
int perfect( int value )
{
   int factorSum = 1; /* current sum of factors */
   int i; /* loop counter */

   /* loop through possible factor values */
   for ( i = 2; i <= value / 2; i++ ) {

      /* if i is factor */
      if ( value % i == 0 ) {
         factorSum += i; /* add to sum */
      } /* end if */

   } /* end for */

   /* return true if value is equal to sum of factors */
   if ( factorSum == value ) {
```

```
    return 1;
  } /* end if */
  else {
    return 0;
  } /* end else */

} /* end function perfect */
```

```
For the integers from 1 to 1000:
6 is perfect
28 is perfect
496 is perfect
```

(٢١ – ٥

```
/* Exercise 5.21 Solution*/
#include <stdio.h>

int prime( int n );

int main()
{
  int loop; /* loop counter */
  int count = 0; /* total number of primes found */

  printf( "The prime numbers from 1 to 10000 are:\n" );

  /* loop through 1 to 10000 */
  for ( loop = 1; loop <= 10000; loop++ ) {

    /* if current number is prime */
    if ( prime( loop ) ) {
       ++count;
       printf( "%6d", loop );

       /* new line after 10 values displayed */
       if ( count % 10 == 0 ) {
          printf( "\n" );
       } /* end if */
```

```
      } /* end if */

   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */

/* prime returns 1 if n is prime */
int prime( int n )
{
   int loop2; /* loop counter */

   /* loop through possible factors */
   for ( loop2 = 2; loop2 <= n / 2; loop2++ ) {

      /* if factor found, not prime */
      if ( n % loop2 == 0 ) {
         return 0;
      } /* end if */

   } /* end for */

   return 1; /* return 1 if prime */

} /* end function prime */
```

```
The prime numbers from 1 to 10000 are:
   1    2    3    5    7   11   13   17   19   23
  29   31   37   41   43   47   53   59   61   67
  71   73   79   83   89   97  101  103  107  109
 113  127  131  137  139  149  151  157  163  167
.
.
.
9733 9739 9743 9749 9767 9769 9781 9787 9791 9803
9811 9817 9829 9833 9839 9851 9857 9859 9871 9883
9887 9901 9907 9923 9929 9931 9941 9949 9967 9973
```

ملاحظة : في الدالة prime عند اختبار العدد n لمعرفة إن كان "أوليا" أم لا اعتبرنا أن n/2 هو الحد الأعلى (upper limit) للاختبار بعروة for :

```
for ( loop2 = 2; loop2 <= n/2; loop2 ++) {
```

ولكن فعليا يكفينا أن نعتبر أن $\sqrt{n}$ [الجذر التربيعي (square root) للعدد n] هو الحد الأعلى للاختبار بعروة **for** :

```
for ( loop2 = 2; loop2 <= ( int ) sqrt (n); loop2 ++) {
```

وفي هذه الحالة نضع في مطلع البرنامج :

```
#include <stdio.h>
#include <math.h>
```

(٥ – ٢٢

```
/* Exercise 5.22 Solution */
#include <stdio.h>

int gcd( int x, int y ); /* function prototype */

int main()
{
  int j; /* loop counter */
  int a; /* first number */
  int b; /* second number */

  /* loop for 5 pairs of inputs */
  for ( j = 1; j <= 5; j++ ) {
    printf( "Enter two integers: " );
    scanf( "%d%d", &a, &b );

    /* find greatest common divisor of a and b */
    printf( "The greatest common divisor "
       "of %d and %d is %d\n\n", a, b, gcd( a, b ) );
  } /* end for */

  return 0; /* indicate successful termination */

} /* end main */

/* gcd finds greatest common divisor of x and y */
int gcd( int x, int y )
```

```
{
   int i;
   int greatest = 1; /* current gcd, 1 is minimum */

   /* loop from 2 to smaller of x and y */
   for ( i = 2; i <= ( ( x < y ) ? x : y ); i++ ) {

      /* if current i divides both x and y */
      if ( x % i == 0 && y % i == 0 ) {
         greatest = i; /* update greatest common divisor */
      } /* end if */

   } /* end for */

   return greatest; /* return greatest common divisor found */

} /* end function gcd */
```

```
Enter two integers: 75 225
The greatest common divisor of 75 and 225 is 75

Enter two integers: 99 30
The greatest common divisor of 99 and 30 is 3

Enter two integers: 17 22
The greatest common divisor of 17 and 22 is 1


Enter two integers: 100 92
The greatest common divisor of 100 and 92 is 4

Enter two integers: 10005 15
The greatest common divisor of 10005 and 15 is 15
```

(٥ – ٢٣

```
/* Exercise 5.23 Solution */
#include <stdio.h>
```

```
int qualityPoints( int average ); /* function prototype */

int main()
{
   int average; /* current average */
   int loop; /* loop counter */

   /* loop for 5 inputs */
   for ( loop = 1; loop <= 5; loop++ ) {
      printf( "\nEnter the student's average: " );
      scanf( "%d", &average );

      /* determine and display corresponding quality points */
      printf( "%d on a 4 point scale is %d\n",
         average, qualityPoints( average ) );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */

/* qualityPoints takes average in range 0 to 100 and
      returns corresponding quality points on 0 to 4 scale */
int qualityPoints( int average )
{

   /* 90 <= average <= 100 */
   if ( average >= 90 ) {
      return 4;
   } /* end if */
   else if ( average >= 80 ) { /* 80 <= average <= 89 */
      return 3;
   } /* end else if */
   else if ( average >= 70 ) { /* 70 <= average <= 79 */
      return 2;
   } /* end else if */
   else if ( average >= 60 ) { /* 60 <= average <= 69 */
      return 1;
   } /* end else if */
   else { /* 0 <= average < 60 */
```

```
    return 0;
  } /* end else */

} /* end function qualityPoints */
```

```
Enter the student's average: 92
92 on a 4 point scale is 4

Enter the student's average: 87
87 on a 4 point scale is 3

Enter the student's average: 75
75 on a 4 point scale is 2

Enter the student's average: 63
63 on a 4 point scale is 1

Enter the student's average: 22
22 on a 4 point scale is 0
```

(٥ – ٢٤

```
/* Exercise 5.24 Solution */
#include <stdio.h>
#include <math.h>

/* function prototype */
double distance( double xOne, double yOne, double xTwo, double yTwo );

int main()
{
  double x1; /* x coordinate of first point */
  double y1; /* y coordinate of first point */
  double x2; /* x coordinate of second point */
  double y2; /* y coordinate of second point */
  double dist; /* distance between two points */

  /* prompt for first point coordinates */
  printf( "Enter the first point: " );
  scanf( "%lf%lf", &x1, &y1 );
```

```
   /* prompt for second point coordinates */
   printf( "Enter the second point: " );
   scanf( "%lf%lf", &x2, &y2 );

   dist = distance( x1, y1, x2, y2 ); /* calculate distance */

   printf( "Distance between ( %.2f, %.2f )"
         " and ( %.2f, %.2f ) is %.2f\n",
         x1, y1, x2, y2, dist );

   return 0; /* indicate successful termination */

} /* end main */

/* distance calculates distance between 2 points
   given by (xOne, yOne) and (xTwo, yTwo) */
double distance( double xOne, double yOne, double xTwo, double yTwo )
{
   double distance; /* distance between two points */

   distance = sqrt( pow( xOne - xTwo, 2 ) + pow( yOne - yTwo, 2 ) );

   return distance;

} /* end function distance */
```

```
Enter the first point: 3 4
Enter the second point: 0 0
Distance between ( 3.00, 4.00 ) and ( 0.00, 0.00 ) is 5.00
```

(٢٥ – ٥

```
/* Exercise 5.25 Solution */
#include <stdio.h>
#include <math.h>

double calculateCharges( double hours ); /* function prototype */

int main()
{
```

```
   double h;                   /* number of hours for current car */
   double currentCharge;       /* parking charge for current car */
   double totalCharges = 0.0;  /* total charges */
   double totalHours = 0.0;    /* total number of hours */
   int i;                      /* loop counter */
   int first = 1;              /* flag for printing table headers */

   printf( "Enter the hours parked for 3 cars: " );

   /* loop 3 times for 3 cars */
   for ( i = 1; i <= 3; i++ ) {
      scanf( "%lf", &h );
      totalHours += h; /* add current hours to total hours */

      /* if first time through loop, display headers */
      if ( first ) {
         printf( "%5s%15s%15s\n", "Car", "Hours", "Charge" );

         /* set flag to false to prevent from printing again */
         first = 0;
      } /* end if */

      /* calculate current car's charge and update total */
      totalCharges += ( currentCharge = calculateCharges( h ) );

      /* display row data for current car */
      printf( "%5d%15.1f%15.2f\n", i, h, currentCharge );
   } /* end for */

   /* display row data for totals */
   printf( "%5s%15.1f%15.2f\n", "TOTAL", totalHours, totalCharges );

   return 0; /* indicate successful termination */

} /* end main */

/* calculateCharges returns charge according to number of hours */
double calculateCharges( double hours )
{
   double charge; /* calculated charge */
```

```
   /* $2 for up to 3 hours */
   if ( hours < 3.0 ) {
      charge = 2.0;
   } /* end if */

   /* $.50 for each hour or part thereof in excess of 3 hours */
   else if ( hours < 19.0 ) {
      charge = 2.0 + .5 * ceil( hours - 3.0 );
   } /* end else if */
   else { /* maximum charge $10 */
      charge = 10.0;
   } /* end else */

   return charge; /* return calculated charge */

} /* end function calculateCharges */
```

(٥–٢٦)

```
/* Exercise 5.26 Solution */
#include <stdio.h>

int integerPower( int b, int e );

int main()
{
   int exp; /* integer exponent */
   int base; /* integer base */

   printf( "Enter integer base and exponent: " );
   scanf( "%d%d", &base, &exp );

   printf( "%d to the power %d is: %d\n",
         base, exp, integerPower( base, exp ) );

         return 0; /* indicate successful termination */

} /* end main */

/* integerPower calculates and returns b raised to the e power */
int integerPower( int b, int e )
{
```

```
   int product = 1; /* resulting product */
   int i; /* loop counter */

   /* multiply product times b (e repetitions) */
   for ( i = 1; i <= e; i++ ) {
      product *= b;
   } /* end for */

   return product; /* return resulting product */

} /* end function integerPower */
```

```
Enter integer base and exponent: 5 3
5 to the power 3 is: 125
```

(٢٧ – ٥

```
/* Exercise 5.27 Solution */
#include <stdio.h>

int quotient( int a, int b ); /* function prototype */
int remainder( int a, int b ); /* function prototype */

int main()
{
   int number; /* input number */
   int divisor = 10000; /* current divisor */

   printf( "Enter an integer between 1 and 32767: " );
   scanf( "%d", &number );

   printf( "The digits in the number are:\n" );

   /* determine and print each digit */
   while ( number >= 10 ) {

      /* if number is >= current divisor, determine digit */
      if ( number >= divisor ) {

         /* use quotient to determine current digit */
```

```
         printf( "%d ", quotient( number, divisor ) );

         /* update number to be remainder */
         number = remainder( number, divisor );

         /* update divisor for next digit */
         divisor = quotient( divisor, 10 );
      } /* end if */
      else { /* if number < current divisor, no digit */
         divisor = quotient( divisor, 10 );
      } /* end else */

   } /* end while */

   printf( "%d\n", number );

   return 0; /* indicate successful termination */

} /* end main */

/* Part A: determine quotient using integer division */
int quotient( int a, int b )
{
   return a / b;

} /* end function quotient */

/* Part B: determine remainder using the remainder operator */
int remainder( int a, int b )
{
   return a % b;

} /* end function remainder */
```

(٢٨ – ٥

```
/* Exercise 5.28 Solution */
#include <stdio.h>
#include <math.h>

/* function prototype */
unsigned seconds( unsigned h, unsigned m, unsigned s );
```

```
int main()
{
   int hours; /* current time's hours */
   int minutes; /* current time's minutes */
   int secs; /* current time's seconds */
   int first; /* first time, in seconds */
   int second; /* second time, in seconds */
   int difference; /* difference between two times, in seconds */

   printf( "Enter the first time as three integers: " );
   scanf( "%d%d%d", &hours, &minutes, &secs );

   /* calculate first time in seconds */
   first = seconds( hours, minutes, secs );

   printf( "Enter the second time as three integers: " );
   scanf( "%d%d%d", &hours, &minutes, &secs );

   /* calculate second time in seconds */
   second = seconds( hours, minutes, secs );

   /* calculate difference */
   difference = fabs( first - second );

   /* display difference */
   printf( "The difference between the times is %d seconds\n",
         difference );

   return 0; /* indicate successful termination */

} /* end main */

/* seconds returns number of seconds since clock "struck 12"
   given input time as hours h, minutes m, seconds s */
unsigned seconds( unsigned h, unsigned m, unsigned s )
{
   return 3600 * h + 60 * m + s;

} /* end function seconds */
```

```
Enter the first time as three integers: 4 20 39
Enter the second time as three integers: 7 20 39
The difference between the times is 10800 seconds
```

(٥ – ٢٩)

```
/* Exercise 5.29 solution */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void multiplication( void ); /* function prototype */

int main( void )
{
  srand( time( NULL ) ); /* seed random number generator */
  multiplication(); /* begin multiplication practice */

  return 0; /* indicate successful termination */

} /* end main */

/* multiplication produces pairs of random numbers and
  prompts user for product */
void multiplication( void )
{
  int x; /* first factor */
  int y; /* second factor */
  int response = 0; /* user response for product */

  /* use sentinel-controlled repetition */
  printf( "Enter -1 to end.\n" );

  /* loop while sentinel value not read from user */
  while ( response != -1 ) {
    x = rand() % 10; /* generate 1-digit random number */
    y = rand() % 10; /* generate another 1-digit random number */

    printf( "How much is %d times %d? ", x, y );
    scanf( "%d", &response );
```

```
      /* loop while not sentinel value or correct response */
      while ( response != -1 && response != x * y ) {
         printf( "No. Please try again.\n? " );
         scanf( "%d", &response );
      } /* end while */

      /* correct response */
      if ( response != -1 ) {
         printf( "Very good!\n\n" );
      } /* end if */

   } /* end while */

   printf( "That's all for now. Salam.\n" );

} /* end function multiplication */
```

```
Enter -1 to end.
How much is 0 times 7? 0
Very good!

How much is 0 times 0? 0
Very good!

How much is 2 times 6? 18
No. Please try again.
? 12
Very good!

How much is 5 times 0? 0
Very good!

How much is 9 times 2? 18
Very good!

How much is 6 times 1? -1
That's all for now. Salam.
```

(٣٠ – ٥

```
/* Exercise 5.30 Solution */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void correctMessage( void ); /* function prototype */
void incorrectMessage( void ); /* function prototype */
void multiplication( void ); /* function prototype */

int main()
{
  srand( time( NULL ) ); /* seed random number generator */
  multiplication(); /* begin multiplication practice */

  return 0; /* indicate successful termination */

} /* end main */

/* correctMessage randomly chooses response to correct answer */
void correctMessage( void )
{

  /* generate random number between 0 and 3 */
  switch ( rand() % 4 ) {

    /* display a random response */
    case 0:
      printf( "Very good!\n\n" );
      break; /* exit switch */

    case 1:
      printf( "Excellent!\n\n" );
      break; /* exit switch */

    case 2:
      printf( "Nice work!\n\n" );
      break; /* exit switch */

    case 3:
      printf( "Keep up the good work!\n\n" );
```

```
         break; /* exit switch */
   } /* end switch */

} /* end function correctMessage */

/* incorrectMessage randomly chooses response to incorrect answer
*/
void incorrectMessage( void )
{

   /* generate random number between 0 and 3 */
   switch ( rand() % 4 ) {

      /* display random response */
      case 0:
         printf( "No. Please try again.\n? " );
         break; /* exit switch */

      case 1:
         printf( "Wrong. Try once more.\n? " );
         break; /* exit switch */

      case 2:
         printf( "Don't give up!\n? " );
         break; /* exit switch */

      case 3:
         printf( "No. Keep trying.\n? " );
         break; /* exit switch */
   } /* end switch */

} /* end function incorrectMessage */

/* multiplication produces pairs of random numbers and
   prompts user for product */
void multiplication( void )
{
   int x;                  /* first factor */
   int y;                 /* second factor */
   int response = 0; /* user response for product */

   /* use sentinel-controlled repetition */
   printf( "Enter -1 to end.\n" );
```

```
   /* loop while sentinel value not read from user */
   while ( response != -1 ) {
      x = rand() % 10; /* generate 1-digit random number */
      y = rand() % 10; /* generate another 1-digit random number */

      printf( "How much is %d times %d? ", x, y );
      scanf( "%d", &response );

      /* loop while not sentinel value or correct response */
      while ( response != -1 && response != x * y ) {
         incorrectMessage();
         scanf( "%d", &response );
      } /* end while */

      /* correct response */
      if ( response != -1 ) {
         correctMessage();
      } /* end if */

   } /* end while */

   printf( "That's all for now. Salam.\n" );
} /* end function multiplication */
```

```
Enter -1 to end.
How much is 7 times 6? 42
Very good!

How much is 8 times 5? 40
Excellent!

How much is 7 times 2? 15
No. Please try again.
? 14
Keep up the good work!

How much is 9 times 6? 54
Keep up the good work!

How much is 3 times 7? -1
That's all for now. Salam.
```

أجوبة تمرينات رقم ٦

٦ – ١ – أ) الخطأ : وجود فاصلة منقوطة في نهاية موجِّه المشغِّل المبدئي **define#** .
التصحيح : احذف الفاصلة المنقوطة .

ب) الخطأ : إسناد قيمة لثابت رمزي باستخدام عبارة إسناد .
التصحيح : أسند قيمة للثابت الرمزي في موجِّه المشغِّل المبدئي **define#** بدون استخدام مؤثر الإسناد كما يلي :

```
#define SIZE 10
```

ج) الخطأ : الإشارة إلى عنصر منظومة خارج حدود المنظومة (b[10]) .
التصحيح : غيِّر القيمة النهائية لمتغير التحكم إلى 9 .

د) الخطأ : وجود فاصلة منقوطة في نهاية موجِّه المشغِّل المبدئي **include#** .
التصحيح : احذف الفاصلة المنقوطة .

هـ) الخطأ : إسناد قيمة لثابت رمزي باستخدام عبارة إسناد .
التصحيح : أسند قيمة للثابت الرمزي في موجِّه المشغِّل المبدئي **define#** بدون استخدام مؤثر الإسناد كما يلي :

```
#define VALUE 120
```

٦ – ٢ – أ)
```
printf( "%c\n", f[ 6 ] );
```

ب)
```
scanf( "%f", &b[ 4 ] );
```

ج)
```
for ( loop = 0; loop <= 4; loop++ )
  g[ loop ] = 8;
```

د)
```
for ( loop = 0; loop <= 99; loop++ )
  sum += c[ loop ];
```

هـ)
```
for ( loop = 0; loop <= 10; loop++ )
  b[ loop ] = a[ loop ];
```

و)
```
smallest = largest = w[ 0 ];

for ( loop = 1; loop <= 98; loop++ )
  if ( w[ loop ] < smallest )
    smallest = w[ loop ];
  else if ( w[ loop ] > largest )
    largest = w[ loop ];
```

٦ – ٣ –أ)
```
for ( i = 0; i <= 9; i++ )
  counts[ i ] = 0;
```

ب)
```
for ( i = 0; i <= 14; i++ )
  ++bonus[ i ];
```

ج)
```
for ( i = 0; i <= 11; i++ ) {
  printf( “Enter a temperature: ” );
  scanf( “%f”, &monthlyTemperatures[ i ] );
}
```

د)
```
for ( i = 0; i <= 4; i++ ) {
  printf( “%d\t”, bestScores[ i ] );
```

٦ – ٤ – أ) **str تحتاج إلى طول لا يقل عن 6 : عنصر واحد لكل حرف في كلمة** salam ، **وعنصر واحد لرمز الإنهاء الخالي** (terminating NULL character) .

ب)
```
printf( "%d %d %d\n", a[ 0 ], a[ 1 ], a[ 2 ] );
```

ج) **تم تعريف عدد كبير من المتغيرات**
```
double f[ 3 ] = { 1.1, 10.01, 100.001 };
```

٦ – ٥ – i) ب (ii) أ (iii) ج

(iv) د (v) أ (vi) د

(vii) ج

٦ – ٦)
```
/* Exercise 6.6 Solution */
#include <stdio.h>

int main()
{
  int salaries[ 11 ] = { 0 }; /* array to hold salary counts */
  int sales;              /* current employee's sales */
  double salary;          /* current employee's salary */
  double i = 0.09;        /* commission percentage */

  /* prompt user for gross sales */
  printf( "Enter employee gross sales ( -1 to end ): " );
```

```
   scanf( "%d", &sales );

   /* while sentinel value not read from user */
   while ( sales != -1 ) {

      /* calculate salary based on sales */
      salary = 200.0 + sales * i;
      printf( "Employee Commission is $%.2f\n", salary );

      /* update appropriate salary range */
      if ( salary >= 200 && salary < 1000 ) {
         ++salaries[ ( int ) salary / 100 ];
      } /* end if */
      else if ( salary >= 1000 ) {
         ++salaries[ 10 ];
      } /* end else if */

      /* prompt user for another employee sales amount */
      printf( "\nEnter employee gross sales ( -1 to end ): " );
      scanf( "%d", &sales );
   } /* end while */

   /* display table of ranges and employees in each range */
   printf( "\nEmployees in the range:\n" );
   printf( "$200-$299 : %d\n", salaries[ 2 ] );
   printf( "$300-$399 : %d\n", salaries[ 3 ] );
   printf( "$400-$499 : %d\n", salaries[ 4 ] );
   printf( "$500-$599 : %d\n", salaries[ 5 ] );
   printf( "$600-$699 : %d\n", salaries[ 6 ] );
   printf( "$700-$799 : %d\n", salaries[ 7 ] );
   printf( "$800-$899 : %d\n", salaries[ 8 ] );
   printf( "$900-$999 : %d\n", salaries[ 9 ] );
   printf( "Over $1000: %d\n", salaries[ 10 ] );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter employee gross sales ( -1 to end ): 3000
Employee Commission is $470.00
```

```
Enter employee gross sales ( -1 to end ): 1000
Employee Commission is $290.00

Enter employee gross sales ( -1 to end ): 10000
Employee Commission is $1100.00

Enter employee gross sales ( -1 to end ): 8000
Employee Commission is $920.00

Enter employee gross sales ( -1 to end ): 200
Employee Commission is $218.00

Enter employee gross sales ( -1 to end ): 7000
Employee Commission is $830.00

Enter employee gross sales ( -1 to end ): -1

Employees in the range:
$200-$299 : 2
$300-$399 : 0
$400-$499 : 1
$500-$599 : 0
$600-$699 : 0
$700-$799 : 0
$800-$899 : 1
$900-$999 : 1
Over $1000: 1
```

(٦-٧)

```
/* Exercise 6.7 Solution */
#include <stdio.h>
#define MAX 10

int main()
{

   /* initialize array a with initializer list */
   int a[ MAX ] = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
   int i;    /* loop counter */
   int pass; /* loop counter */
   int hold; /* temporary variable for swapping */
   int swap; /* flag to break loop if elements are sorted */
```

```
printf( "Data items in original order\n" );

/* display original, unsorted array */
for ( i = 0; i < MAX; i++ ) {
  printf( "%4d", a[ i ] );
} /* end for */

printf( "\n\n" );

/* begin sorting the array */
for ( pass = 1; pass < MAX; pass++ ) {
  swap = 0;

  /* traverse and compare unsorted part of array */
  for ( i = 0; i < MAX - pass; i++ ) {

    /* compare adjacent array elements */
    if ( a[ i ] > a[ i + 1 ] ) {
      swap = 1; /* raise flag if any elements are swapped */
      hold = a[ i ];
      a[ i ] = a[ i + 1 ];
      a[ i + 1 ] = hold;
    } /* end if */

  } /* end for */

  printf( "After Pass %d: ", pass );

  /* display array after each pass */
  for ( i = 0; i <= MAX-pass; i++ ) {
    printf( "  %d", a[ i ] );
  } /* end for */

  printf( "\n" );

  /* break loop if array is sorted */
  if ( !swap ) {
    break;
  } /* end if */

} /* end for */
```

```
   printf( "\nData items in ascending order\n" );

   /* display array in sorted order */
   for ( i = 0; i < 10; i++ ) {
      printf( "%4d", a[ i ] );
   } /* end for */

   printf( "\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Data items in original order
   10   9    8   7   6   5   4   3   2   1

After Pass 1:   9   8   7   6   5   4   3   2   1   10
After Pass 2:   8   7   6   5   4   3   2   1   9
After Pass 3:   7   6   5   4   3   2   1   8
After Pass 4:   6   5   4   3   2   1   7
After Pass 5:   5   4   3   2   1   6
After Pass 6:   4   3   2   1   5
After Pass 7:   3   2   1   4
After Pass 8:   2   1   3
After Pass 9:   1   2

Data items in ascending order
    1   2   3   4   5   6   7   8   9   10
```

(٨-٦

```
/* Exercise 6.8 Solution */
#include <stdio.h>
#define SIZE 100

void mean( int answer[] );             /* function prototype */
void median( int answer[] );           /* function prototype */
void mode( int freq[], int answer[] ); /* function prototype */

int main()
{
```

```
   /* array of responses */
   int response[ SIZE ] = { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
                            7, 8, 9, 5, 9, 8, 7, 8, 7, 1,
                            6, 7, 8, 9, 3, 9, 8, 7, 1, 7,
                            7, 8, 9, 8, 9, 8, 9, 7, 1, 9,
                            6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
                            7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
                            5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
                            7, 8, 9, 6, 8, 7, 8, 9, 7, 1,
                            7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
                            4, 5, 6, 1, 6, 5, 7, 8, 7, 9};
   int frequency[ 10 ] = { 0}; /* array of response frequencies */

   mean( response ); /* process mean */
   median( response ); /* process median */
   mode( frequency, response ); /* process mode */

   return 0; /* indicates successful termination */

} /* end main */

/* calculate average of all response values */
void mean( int answer[] )
{
   int j;        /* loop counter */
   int total = 0; /* total of all response values */

   printf( "%s\n%s\n%s\n", "******", " Mean", "******" );

   /* total response values */
   for ( j = 0; j <= SIZE - 1; j++ ) {
      total += answer[ j ];
   } /* end for */

   /* output results */
   printf( "The mean is the average value of the data\n" );
   printf( "items. The mean is equal to the total of\n" );
   printf( "all the data items divided by the number\n" );
   printf( "of data items ( %d ). ,", SIZE );
   printf( "The mean value for this run is: " );
   printf( "%d / %d = %.4f\n\n", total, SIZE, ( double ) total / SIZE );
} /* end function mean */
```

```
/*sort an array and determine median element's value */
void median( int answer[] )
{
  int loop;     /* loop counter */
  int pass;     /* loop counter */
  int hold;     /* temporary variable for swapping */
  int firstRow; /* flag to indicate first row of array */

  printf( "\n%s\n%s\n%s\n", "******", "Median", "******" );
  printf( "The unsorted array of responses is\n" );

  /* display unsorted array */
  for ( loop = 0, firstRow = 1; loop <= SIZE - 1; loop++ ) {

    /* start a new line */
    if ( loop % 20 == 0 && !firstRow ) {
      printf( "\n" );
    } /* end if */

    printf( "%2d", answer[ loop ] );
    firstRow = 0;
  } /* end for */

  printf( "\n\n" );

  /* sort array */
  for ( pass = 0; pass <= SIZE - 2; pass++ ) {

    /* compare elements and swap if necessary */
    for ( loop = 0; loop <= SIZE - 2; loop++ ) {

      /* swap elements */
      if ( answer[ loop ] > answer[ loop + 1 ] ) {
        hold = answer[ loop ];
        answer[ loop ] = answer[ loop + 1 ];
        answer[ loop + 1 ] = hold;
      } /* end if */

    } /* end for */

  } /* end for */
```

```
   printf( "The sorted array is\n" );

   /* display sorted array */
   for ( loop = 0, firstRow = 1; loop <= SIZE - 1; loop++ ) {

      /* start a new line */
      if ( loop % 20 == 0 && !firstRow ) {
         printf( "\n" );
      } /* end if */

      printf( "%2d", answer[ loop ] );
      firstRow = 0;
   } /* end for */

   printf( "\n\n" );

   /* even number of elements */
   if ( SIZE % 2 == 0 ) {
      printf( "The median is the average of elements %d",
         ( SIZE + 1 ) / 2 );
      printf( " and %d of", 1 + ( SIZE + 1 ) / 2 );
      printf( " the sorted %d element array.\n", SIZE );
      printf( "For this run the median is " );
      printf( "%.1f\n\n", ( double )( answer[ ( SIZE + 1 ) / 2 ] +
         answer[ ( SIZE + 1 ) / 2 + 1 ] ) / 2 );
   } /* end if */
   else { /* odd number of elements  */
      printf( "The median is element %d of ", ( SIZE + 1 ) / 2 );
      printf( "the sorted %d element array.\n", SIZE );
      printf( "For this run the median is " );
      printf( "%d\n\n", answer[ ( SIZE + 1 ) / 2 - 1 ] );
   } /* end else */

} /* end function median */

/* determine most frequent response */
void mode( int freq[], int answer[] )
{
   int rating;          /* loop counter */
   int loop;            /* loop counter */
   int largest = 0;     /* represents largest frequency */
```

```
int array[ 10 ] = { 0}; /* array used to hold largest frequencies */
int count = 0;          /* flag to count number of modes */

printf( "\n%s\n%s\n%s\n", "******", " Mode", "******" );

/* set all frequencies to 0 */
for ( rating = 1; rating <= 9; rating++ ) {
  freq[ rating ] = 0;
} /* end for */

/* traverse array and increment corresponding frequency */
for ( loop = 0; loop <= SIZE - 1; loop++ ) {
  ++freq[ answer[ loop ] ];
} /* end for */

printf( "%s%11s%19s\n\n", "Response", "Frequency", "Histogram" );
printf( "%54s\n", "1    1    2    2" );
printf( "%54s\n\n", "5    0    5    0    5" );

/* display values and frequency */
for ( rating = 1; rating <= 9; rating++ ) {
  printf( "%8d%11d          ", rating, freq[ rating ] );

  /* test if current frequency is greater than largest frequency */
  if ( freq[ rating ] > largest ) {
    largest = freq[ rating ];

    /* set values of array to 0 */
    for ( loop = 0; loop < 10; loop++ ) {
      array[ loop ] = 0;
    } /* end for */

    /* add new largest frequency to array */
    array[ rating ] = largest;
    ++count;
  } /* end if */
  /* if current frequency equals largest, add current to array */
  else if ( freq[ rating ] == largest ) {
    array[ rating ] = largest;
    ++count;
  } /* end else if */
```

```
      /* display histogram */
      for ( loop = 1; loop <= freq[ rating ]; loop++ ) {
         printf( "*" );
      } /* end for */

      printf( "\n" );
   } /* end for */

   printf( "\n" );

   /* if more than one mode */
   if ( count > 1 ) {
      printf( "The modes are:  " );
   } /* end if */
   else { /* only one mode */
      printf( "The mode is: " );
   } /* end else */

   /* display mode(s) */
   for ( loop = 1; loop <= 9; loop++ ) {

      if ( array[ loop ] != 0 ) {
         printf( "%d with a frequency of %d\n\t\t", loop, array[ loop ] );
      } /* end if */

   } /* end for */

   printf( "\n" );
} /* end function mode */
```

```
******
 Mean
******
The mean is the average value of the data
items. The mean is equal to the total of
all the data items divided by the number
of data items ( 100 ). ,The mean value for this run is: 662 / 100 = 6.6200


******
Median
```

```
******
The unsorted array of responses is
  6 7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 1
  6 7 8 9 3 9 8 7 1 7 7 8 9 8 9 8 9 7 1 9
  6 7 8 7 8 7 9 8 9 2 7 8 9 8 9 8 9 7 5 3
  5 6 7 2 5 3 9 4 6 4 7 8 9 6 8 7 8 9 7 1
  7 4 4 2 5 3 8 7 5 6 4 5 6 1 6 5 7 8 7 9

The sorted array is
  1 1 1 1 1 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5
  5 5 5 5 5 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7
  7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8
  8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
  9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

The median is the average of elements 50 and 51 of the sorted 100 element array.

For this run the median is 7.0


******
 Mode
******
Response    Frequency       Histogram
                                1   1   2   2
                            5   0   5   0   5
       1        5       *****
       2        3       ***
       3        4       ****
       4        5       *****
       5        8       ********
       6        9       *********
       7       23       ***********************
       8       23       ***********************
       9       20       ********************
The modes are:   7 with a frequency of 23
                 8 with a frequency of 23
```

(٦-٩)

```
/* Exercise 6.9 Solution */
#include <stdio.h>
#define MAX 20
```

```
int main()
{
   int a[ MAX ] = { 0 }; /* array for user input */
   int i;              /* loop counter */
   int j;              /* loop counter */
   int k = 0;          /* number of values currently entered */
   int duplicate;      /* flag for duplicate values */
   int value;          /* current value */

   printf( "Enter 20 integers between 10 and 100:\n" );

   /* get 20 integers from user */
   for ( i = 0; i <= MAX - 1; i++ ) {
      duplicate = 0;
      scanf( "%d", &value );

      /* test if integer is a duplicate */
      for ( j = 0; j < k; j++ ) {

         /* if duplicate, raise flag and break loop */
         if ( value == a[ j ] ) {
            duplicate = 1;
            break;
         } /* end if */

      } /* end for */

      /* if number is not a duplicate enter it in array */
      if ( !duplicate ) {
         a[ k++ ] = value;
      } /* end if */

   } /* end for */

   printf( "\nThe nonduplicate values are:\n" );

   /* display array of nonduplicates */
   for ( i = 0; a[ i ] != 0; i++ ) {
      printf( "%d  ", a[ i ] );
   } /* end for */

   printf( "\n" );
```

```
   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter 20 integers between 10 and 100:
10 11 12 13 14 15 16 17 18 19 20 21 10 11 12 13 14 15 16 17

The nonduplicate values are:
10  11  12  13  14  15  16  17  18  19  20  21
```

(٦ – ١٠

```
/* Exercise 6.10 Solution */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
   long i;              /* loop counter */
   int j;              /* loop counter */
   int x;               /* first die */
   int y;               /* second die */
   int sum[ 13 ] = { 0 }; /* count occurrences of each combination */

   /* array expected contains counts for the expected
      number of times each sum occurs in 36 rolls of the dice */
   int expected[ 13 ] = { 0, 0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1};

   srand( time( NULL ) ); /* seed random number generator */

   /* roll dice 36,000 times */
   for ( i = 1; i <= 36000; i++ ) {
      x = 1 + rand() % 6;
      y = 1 + rand() % 6;
      ++sum[ x + y ];
   } /* end for */

   printf( "%10s%10s%10s%10s\n", "Sum", "Total", "Expected", "Actual" );
```

```
   /* display results of rolling dice */
   for ( j = 2; j <= 12; j++ ) {
      printf( "%10d%10d%9.3f%%%9.3f%%\n", j, sum[ j ],
         100.0 * expected[ j ] / 36, 100.0 * sum[ j ] / 36000 );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */
```

Sum	Total	Expected	Actual
2	1018	2.778%	2.828%
3	2008	5.556%	5.578%
4	3020	8.333%	8.389%
5	4024	11.111%	11.178%
6	4891	13.889%	13.586%
7	6011	16.667%	16.697%
8	5065	13.889%	14.069%
9	3984	11.111%	11.067%
10	2970	8.333%	8.250%
11	1989	5.556%	5.525%
12	1020	2.778%	2.833%

(٦ – ١١

```
/* Exercise 6.11 Solution */
#include <stdio.h>
#include <ctype.h>

int main()
{
   int plane[ 11 ] = { 0 }; /* seats on the plane */
   int i = 0;               /* counter */
   int firstClass= 1;       /* first class seats start at 1 */
   int economy = 6;         /* economy seats start at 6 */
   int choice;              /* user's choice */
   char response[ 2 ];      /* user's response */

   /* loop 10 times */
   while ( i < 10 ) {
```

```
printf( "\n%s\n%s\n? ", "Please type 1 for \"first class\"",
      "Please type 2 for \"economy\"" );
scanf( "%d", &choice );


/* if user selects first class */
if ( choice == 1 ) {


  /* if seats are available in first class */
  if ( !plane[ firstClass ] && firstClass <= 5 ) {
    printf( "Your seat assignment is %d\n", firstClass );
    plane[ firstClass++ ] = 1;
    i++;
  } /* end if */
  /* if no first class seats, but economy seats available */
  else if ( firstClass > 5 && economy <= 10 ) {


    /* ask if passenger would like to sit in economy */
    printf( "The first class section is full.\n" );
    printf( "Would you like to sit in the economy" );
    printf( " section ( Y or N )? " );
    scanf( "%s", response );


    /* if response is yes, then assign seat */
    if ( toupper( response[ 0 ] ) == 'Y' ) {
      printf( "Your seat assignment is %d\n", economy );
      plane[ economy++ ] = 1;
      i++;
    } /* end if */
    else { /* print next departure */
      printf( "Next flight leaves in 3 hours.\n" );
    } /* end else */


  } /* end else if */
  else { /* print next departure */
    printf( "Next flight leaves in 3 hours.\n" );
  } /* end else */


} /* end if */
```

```
      else { /* if user selects economy */

         /* if seats available, assign seat */
         if ( !plane[ economy ] && economy <= 10 ) {
            printf( "Your seat assignment is %d\n", economy );
            plane[ economy++ ] = 1;
            i++;
         } /* end if */
         /* if only first class seats are available */
         else if ( economy > 10 && firstClass <= 5 ) {

            /* ask if first class is suitable */
            printf( "The economy section is full.\n" );
            printf( "Would you like to sit in first class" );
            printf( " section ( Y or N )? " );
            scanf( "%s", response );

            /* if response is yes, assign seat */
            if ( toupper( response[ 0 ] ) == 'Y' ) {
               printf( "Your seat assignment is %d\n", firstClass );
               plane[ firstClass++ ] = 1;
               i++;
            } /* end if */
            else { /* print next departure */
               printf( "Next flight leaves in 3 hours.\n" );
            } /* end else */

         } /* end else if */
         else { /* print next departure */
            printf( "Next flight leaves in 3 hours.\n" );
         } /* end else */

      } /* end else */

   } /* end while */

   printf( "\nAll seats for this flight are sold.\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Please type 1 for "first class"
Please type 2 for "economy"
? 2
Your seat assignment is 6

Please type 1 for "first class"
Please type 2 for "economy"
? 1
Your seat assignment is 1

Please type 1 for "first class"
Please type 2 for "economy"
? 2
Your seat assignment is 7
.
.
.
Please type 1 for "first class"
Please type 2 for "economy"
? 1
The first class section is full.
Would you like to sit in the economy section ( Y or N )? n
Next flight leaves in 3 hours.

Please type 1 for "first class"
Please type 2 for "economy"
? 1
The first class section is full.
Would you like to sit in the economy section ( Y or N )? y
Your seat assignment is 9

Please type 1 for "first class"
Please type 2 for "economy"
? 2
Your seat assignment is 10

All seats for this flight are sold.
```

(٦ – ١٢

```
/* Exercise 6.12 Solution */
#include <stdio.h>
#define SIZE 1000
```

```
int main()
{
  int array[ SIZE ]; /* array to indicate prime numbers */
  int loop;          /* loop counter */
  int loop2;         /* loop counter */
  int count = 0;     /* total prime numbers */

  /* set all array elements to 1 */
  for ( loop = 0; loop < SIZE; loop++ ) {
    array[ loop ] = 1;
  } /* end for */

  /* test for multiples of current subscript */
  for ( loop = 1; loop < SIZE; loop++ ) {

    /* start with array subscript two */
    if ( array[ loop ] == 1 && loop != 1 ) {

      /* loop through remainder of array */
      for ( loop2 = loop; loop2 <= SIZE; loop2++ ) {

        /* set to zero all multiples of loop */
        if ( loop2 % loop == 0 && loop2 != loop ) {
          array[ loop2 ] = 0;
        } /* end if */

      } /* end for */

    } /* end if */

  } /* end for */

  /* display prime numbers in the range 2 - 197 */
  for ( loop = 2; loop < SIZE; loop++ ) {

    if ( array[ loop ] == 1 ) {
      printf( "%3d is a prime number.\n", loop );
      ++count;
    } /* end if */

  } /* end for */
```

```
   printf( "A total of %d prime numbers were found.\n", count );

   return 0; /* indicate successful termination */

} /* end main */
```

```
  2 is a prime number.
  3 is a prime number.
  5 is a prime number.
  7 is a prime number.
11 is a prime number.
13 is a prime number.
17 is a prime number.
19 is a prime number.
.
.
.
971 is a prime number.
977 is a prime number.
983 is a prime number.
991 is a prime number.
997 is a prime number.
A total of 168 prime numbers were found.
```

أجوبة تمرينات رقم ٧

٧ – ١ – أ) الخطأ : محدد التحويل **s** يتوقع وسيطا فعليا من النوع : مؤشر إلى رمز (pointer to) **char** .

التصحيح : لطباعة الرمز **'c'** استخدم محدد التحويل **%c** ، أو غيّر **'c'** إلى **"c"** .

ب) الخطأ : محاولة طباعة الرمز الحرفي **%**(literal character) دون استخدام محدد التحويل **% %** .

التصحيح : استخدم **% %** لطباعة الرمز الحرفي **%** .

ج) الخطأ : محدد التحويل **c** يتوقع وسيطا فعليا من النوع **char** .

التصحيح : لطباعة الحرف الأول من "Monday" استخدم محدد التحويل %1s.

د) الخطا : محاولة طباعة الرمز الحرفي " دون استخدام متتابعة الهروب "\ .

التصحيح : استبدل بكل حاصرة في مجموعة الحاصرات الداخلية "\ .

هـ) الخطأ : سلسلة التحكم في الصيغة ليست محاطة بحاصرتين مزدوجتين .

التصحيح : ضع %d%d بين حاصرتين مزدوجتين .

و) الخطأ : الرمز × محاط بحاصرتين مزدوجتين .

التصحيح : الثوابت الرمزية التي تُطبع باستخدام **%c** يجب أن تحاط بحاصرتين مفردتين .

ز) الخطأ السلسلة المطلوب طباعتها محاطة بحاصرتين مفردتين .

التصحيح : استخدم حاصرتين مزدوجتين بدلا من المفردتين لتمثيل سلسلة .

٧ – ٢ – أ) `printf( "%10d\n", 1234 );`

ب) `printf( "%+.3e\n", 123.456789 );`

ج) `scanf( "%lf", &number );`

د) `printf( "%#o\n", 100 );`

هـ) `scanf( "%s", string );`

و) scanf("%[0123456789]", n);

ز) printf("%*.*f\n", x, y, 87.4573);

ح) scanf("%f%%", &percent);

ط) printf("%+20.3l_f\n", 3.333333);

٧ – ٣ – أ) printf(“%-15.8u”, (unsigned) 40000);

ب) scanf(“%x”, hex);

ج) printf(“%+d %d\n”, 200, 200);

د) printf(%#x\n”, 100);

هـ) scanf(“%[^p]”, s);

و) printf(“%09.3f\n”, 1.234);

ز) scanf(“%d%*c%d%*c%d”, &hour, &minute, &second);

ح) scanf(“\”%[^\”]”, s);

ط) scanf(“%d:%d:%d:”, &hour, &minute, &second);

٧ – ٤ – أ) 10000

ب) **لا يمكن طباعة سلسلة رموز باستخدام المحدد %c .**

ج) 1024.988

د) 021
0X11
1.008837e+03

هـ) 1000000
+1000000

و) 4.45E+02 **(قبلها على اليسار فراغان)**

ز) 4.4e+02 **(قبلها على اليسار 3 فراغات)**

ح) **القيمة ذات النقطة العائمة لا يمكن طباعتها باستخدام محدد التحويل** %d .

٧ – ٥ – أ) printf(“%s\n”, “Happy Eid”);

ب) printf(“%s\n”, “Salam”);

ج) printf(“%s\n”, “This is a string”);

د) printf(“\“%s\””, “See You”);

هـ) printf(“%s\n”, day);

و) printf(“Enter your name: “);

ز) printf(“%f”, 123.456);

ح) printf(“%c%c\n”, ‘O’, ‘K’);

ط) scanf(“%c”, &s[7]);

٧-٦)

```
/* Exercise 7.6 Solution */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
   int a[ 10 ] = { 0 }; /* random integers from 1 to 1000 */
   int i;               /* loop counter */
   int count;           /* number of characters in current value */
   int totalCount = 0;  /* total characters in array */

   srand( time( NULL ) );

   /* fill the array with random numbers */
   for ( i = 0; i <= 9; i++ ) {
      a[ i ] = 1 + rand() % 1000;
   } /* end for */

   /* print table headers */
   printf( "%s\t%s\n", "Value", "Total characters" );

   /* loop through 10 elements */
   for ( i = 0; i <= 9; i++ ) {
      printf( "%d%n", a[ i ], &count );
      totalCount+= count; /* update totalCount */
      printf( "\t%d\n", totalCount );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */
```

```
Value     Total characters
842       3
18        5
220       8
658       11
275       14
647       17
657       20
623       23
242       26
471       29
```

(٧–٧

```
/* Exercise 7.7 Solution */
#include <stdio.h>

int main()
{
   int i; /* loop counter */
   int x; /* first integer from user */
   int y; /* second integer from user */

   /* loop four times */
   for ( i = 1; i <= 4; i++ ) {
      printf( "\nEnter two integers: " );
      scanf( "%i%d", &x, &y );
      printf( "%d %d\n", x, y );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter two integers: 10  10
10  10

Enter two integers: -10  -10
-10  -10
```

```
Enter two integers: 010  010
8  10

Enter two integers: 0×10  0×10
16  0
```

(٨-٧

```
/* Exercise 7.8 Solution */
#include <stdio.h>

int main()
{

   /* print the integer 12345 */
   printf( "%10d\n", 12345 );
   printf( "%5d\n", 12345 );
   printf( "%2d\n\n", 12345 );

   /* print the floating-point value 1.2345 */
   printf( "%10f\n", 1.2345 );
   printf( "%6f\n", 1.2345 );
   printf( "%2f\n", 1.2345 );

   return 0; /* indicate successful termination */

} /* end main */
```

```
       12345
12345
12345

   1.234500
 1.234500
 1.234500
```

(٩-٧

```
/* Exercise 7.9 Solution */
#include <stdio.h>
```

```
int main()
{
   printf( "%.0f\n", 100.453627 );
   printf( "%.1f\n", 100.453627 );
   printf( "%.2f\n", 100.453627 );
   printf( "%.3f\n", 100.453627 );
   printf( "%.4f\n", 100.453627 );

   return 0; /* indicate successful termination */

} /* end main */
```

```
100
100.5
100.45
100.454
100.4536
```

(٧-١٠

```
/* Exercise 7.10 Solution */
#include <stdio.h>

int main()
{
   int count;         /* length of string */
   char string[ 20 ]; /* string entered by user */

   /* read string from user and find length */
   printf( "Enter a string:\n" );
   scanf( "%s%n", string, &count );

   printf( "%*s\n", 2 * count, string ); /* print the string */

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter a string:
```

```
salam
        salam
```

(١١ – ٧

```
/* Exercise 7.11 Solution */
#include <stdio.h>

int main()
{
  int fahrenheit; /* holds fahrenheit temperature */
  double celcius; /* holds celcius temperature */

  printf( "%10s%12s\n", "Fahrenheit", "Celcius" );

  /* convert fahrenheit to celsius and display temperatures
    showing the sign for celsius temperatures */
  for ( fahrenheit = 0; fahrenheit <= 212; fahrenheit++ ) {
    celcius = 5.0 / 9.0 * ( fahrenheit - 32 );
    printf( "%10d%+12.3f\n", fahrenheit, celcius );
  } /* end for */

  return 0; /* indicate successful termination */

} /* end main */
```

```
Fahrenheit   Celcius
         0    -17.778
         1    -17.222
         2    -16.667
         3    -16.111
         4    -15.556
         5    -15.000
         6    -14.444
         7    -13.889
         .
         .
         .
       204    +95.556
       205    +96.111
```

```
206    +96.667
207    +97.222
208    +97.778
209    +98.333
210    +98.889
211    +99.444
212   +100.000
```

(٧ – ١٢

```
/* Exercise 7.12 Solution */
#include <stdio.h>

int main()
{

   /* test all escape sequences */
   printf( "The single quote : \'\n" );
   printf( "The double quote : \"\n" );
   printf( "The question mark: \?\n" );
   printf( "The backslash    : \\\n" );

   printf( "The bell. \a\n\n" );

   printf( "Move cursor back one position on current line. *\b*\n" );
   printf( "Move cursor to start of next logical page. *\f*\n" );

   printf( "Move cursor to the beginning of next line. *\n*\n" );
   printf( "Move cursor to the beginning of current line. *\r*\n" );

   printf( "Move cursor to the next horizontal tab position. *\t*\n" );
   printf( "Move cursor to the next vertical tab position. *\v*\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

```
The single quote    : '
The double quote  : "
The question mark: ?
The backslash       : \
The bell .
```

```
Move cursor back one position on current line. *
Move cursor to start of next logical page. *?*
Move cursor to the beginning of next line. *
*
*ove cursor to the beginning of current line. *
Move cursor to the next horizontal tab position. *       *
Move cursor to the next vertical tab position. *?*
```

(٧ – ١٣

```
/* Exercise 7.13 Solution */
#include <stdio.h>

int main()
{
   printf( "Did the \? print at the end of the sentence?\n" );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Did the ? print at the end of the sentence?
```

(٧ – ١٤

```
/* Exercise 7.14 Solution */
#include <stdio.h>

int main()
{
   int array[ 5 ]; /* holds the value 437 five times */
   int loop;       /* loop counter */

   /* array of table headers */
   char *s[] = { "Read with %d:", "Read with %i:", "Read with %o:",
                 "Read with %u:", "Read with %x:"};

   /* prompt the user and read 5 values */
   printf( "Enter the value 437 five times: " );
```

```
   scanf( "%d%i%o%u%x", &array[ 0 ], &array[ 1 ], &array[ 2 ],
      &array[ 3 ], &array[ 4 ] );

   /* loop through all 5 values */
   for ( loop = 0; loop <= 4; loop++ ) {

      /* print each of the 5 values */
      printf( "%s\n%d %i %o %u %x\n\n", s[ loop ], array[ loop ],
         array[ loop ], array[ loop ], array[ loop ], array[ loop ] );
   } /* end for */

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the value 437 five times: 437 437 437 437 437
Read with %d:
437 437 665 437 1b5

Read with %i:
437 437 665 437 1b5

Read with %o:
287 287 437 287 11f

Read with %u:
437 437 665 437 1b5

Read with %x:
1079 1079 2067 1079 437
```

(١٥ – ٧

```
/* Exercise 7.15 Solution */
#include <stdio.h>

int main()
{
   float a[ 3 ]; /* holds the value 1.2345 three times */
```

```
   /* array of table headers */
   char *s[] = { "Read with %e:", "Read with %f:", "Read with %g:"
};

   /* prompt the user and read 3 values */
   printf( "Enter the value 1.2345 three times: " );
   scanf( "%e%f%g", &a[ 0 ], &a[ 1 ], &a[ 2 ] );

   printf( "%s%e\n\n", s[ 0 ], a[ 0 ] );
   printf( "%s%f\n\n", s[ 1 ], a[ 1 ] );
   printf( "%s%g\n\n", s[ 2 ], a[ 2 ] );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the value 1.2345 three times: 1.2345 1.2345 1.2345
Read with %e: 1.234500e+000

Read with %f: 1.234500

Read with %g: 1.2345
```

(٧-١٦

```
/* Exercise 7.16 Solution */
#include <stdio.h>

int main()
{
   char a[ 10 ]; /* first string */
   char b[ 10 ]; /* second string */
   char c[ 10 ]; /* third string */

   /* prompt user and read three strings */
   printf( "Enter the strings omar, \"omar\", and 'omar':\n" );
   scanf( "%s%s%s", a, b, c );

   printf( "%s %s %s\n", a, b, c ); /* display strings */
```

```
   return 0; /* indicate successful termination */

} /* end main */
```

```
Enter the strings omar, "omar", and 'omar' :
omar
"omar"
'omar'
omar "omar" 'omar'
```

(١٧ – ٧

```
/* Exercise 7.17 Solution */
#include <stdio.h>

int main()
{
   const char questionMark = '?'; /* define '?' as a char constant */

   printf( "This %c can be printed without using the \\\?\n",
      questionMark );

   return 0; /* indicate successful termination */

} /* end main */
```

```
This ? can be printed without using the \?
```

(١٨ – ٧

```
/* Exercise 7.18 Solution */
#include <stdio.h>

int main()
{

   /* output the value 9876.12345 with precisions from 1 to 9 */
```

```
   printf( "Precision: %d, value = %.1g\n", 1, 9876.12345 );
   printf( "Precision: %d, value = %.2g\n", 2, 9876.12345 );
   printf( "Precision: %d, value = %.3g\n", 3, 9876.12345 );
   printf( "Precision: %d, value = %.4g\n", 4, 9876.12345 );
   printf( "Precision: %d, value = %.5g\n", 5, 9876.12345 );
   printf( "Precision: %d, value = %.6g\n", 6, 9876.12345 );
   printf( "Precision: %d, value = %.7g\n", 7, 9876.12345 );
   printf( "Precision: %d, value = %.8g\n", 8, 9876.12345 );
   printf( "Precision: %d, value = %.9g\n", 9, 9876.12345 );

   return 0; /* indicate successful termination */

} /* end main */
```

```
Precision: 1, value = le)004
Precision: 2, value = 9.9e+003
Precision: 3, value = 9.88e+003
Precision: 4, value = 9876
Precision: 5, value = 9876.1
Precision: 6, value = 9876.12
Precision: 7, value = 9876.123
Precision: 8, value = 9876.1234
Precision: 9, value = 9876.12345
```

ملحق (أ)

قاعدة الأولوية بالنسبة للمؤثرات
Operator Precedence

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 إلى اليمين ، وتجميع المؤثرات الأحادية (unary operators) من اليمين إلى اليسار ، والمؤثر :? من اليمين إلى اليسار. وهناك استثناء وهو أن تجميع مؤثرات الإسناد (assignment operators) يكون من اليمين إلى اليسار .

الأولوية تتناقص من الأعلى إلى الأسفل

التجميع Associativity	ملاحظات Remarks	المؤثر Operator	
من اليسار إلى اليمين	parentheses (function call operator)	القوسان (مؤثر استدعاء دالة)	( )
من اليسار إلى اليمين	array subscript	مؤشر منظومة	[ ]
من اليسار إلى اليمين	member selection via object	اختيار عنصر عن طريق هدف	.
من اليسار إلى اليمين	member selection via pointer	اختيار عنصر عن طريق مؤشر	->
من اليمين إلى اليسار	unary preincrement	الزيادة السابقة الأحادية	++
من اليمين إلى اليسار	unary predecrement	النقصان السابق الأحادي	--
من اليمين إلى اليسار	unary plus	زائد الأحادية	+
من اليمين إلى اليسار	unary minus	ناقص الأحادية	-
من اليمين إلى اليسار	unary logical negation	النفي المنطقي الأحادي	!
من اليمين إلى اليسار	unary bitwise complement	المكمِّل البتّي الأحادي	~
من اليمين إلى اليسار	C-style unary cast	الصب الأحادي	(type)
من اليمين إلى اليسار	dereference	الإطلاق	*
من اليمين إلى اليسار	address	العُنوان	&

sizeof	تحديد الحجم بالبايتات	determine size in bytes	من اليمين إلى اليسار
*	الضرب	multiplication	من اليسار إلى اليمين
/	القسمة	division	من اليسار إلى اليمين
%	ايجاد الباقي الصحيح	modulus	من اليسار إلى اليمين
+	الجمع	addition	من اليسار إلى اليمين
-	الطرح	subtraction	من اليسار إلى اليمين
<<	الإزاحة لليسار بتًّيا	bitwise left shift	من اليسار إلى اليمين
>>	الإزاحة لليمين بتًّيا	bitwise right shift	من اليسار إلى اليمين
<	أصغر من العلاقية	relational less than	من اليسار إلى اليمين
<=	أصغر من أو يساوي العلاقية	relational less than or equal to	من اليسار إلى اليمين
>	أكبر من العلاقية	relational greater than	من اليسار إلى اليمين
>=	أكبر من أو يساوي العلاقية	relational greater than or equal to	من اليسار إلى اليمين
==	يساوي العلاقية	relational is equal to	من اليسار إلى اليمين
!=	لا يساوي العلاقية	relational is not equal to	من اليسار إلى اليمين
&	و : البتية	bitwise AND	من اليسار إلى اليمين
^	أو : المتباعدة البتية	bitwise exclusive OR	من اليسار إلى اليمين
|	أو : الاحتوائية البتية	bitwise inclusive OR	من اليسار إلى اليمين
&&	و: المنطقية	logical AND	من اليسار إلى اليمين
||	أو: المنطقية	logical OR	من اليسار إلى اليمين
?:	الشَّرطي الثلاثي	ternary conditional	من اليسار إلى اليمين
=	الإسناد	assignment	من اليمين إلى اليسار
+=	إسناد الجمع	addition assignment	من اليمين إلى اليسار
-=	إسناد الطرح	subtraction assignment	من اليمين إلى اليسار
*=	إسناد الضرب	multiplication assignment	من اليمين إلى اليسار
/=	إسناد القسمة	division assignment	من اليمين إلى اليسار
%=	إسناد الباقي الصحيح	modulus assignment	من اليمين إلى اليسار

&=	إسناد و : البتية	bitwise AND assignment	من اليمين إلى اليسار
^=	إسناد أو : المتباعدة البتية	bitwise excusive OR assignment	من اليمين إلى اليسار
|=	إسناد أو : الاحتوائية البتية	bitwise inclusive OR assignment	من اليمين إلى اليسار
<<=	إسناد الإزاحة لليسار بتيا	bitwise left shift assignment	من اليمين إلى اليسار
>>=	إسناد الإزاحة لليمين بتيا بإشارة	bitwise right shift with sign	من اليمين إلى اليسار
,	الفاصلة	comma	من اليسار إلى اليمين

جدول أولوية المؤثرات

C operator precedence chart

ملحق (ب)

مجموعة رموز ASCII

ASCII Character Set

الجدول التالي يبيِّن ترتيب الرموز (ordering of characters) في مجموعة رموز ASCII ("الشفرة القياسية الأمريكية لتبادل المعلومات" American Standard Code for Information Interchange) لتشفير الرموز. ويعطي الجدول التمثيل الداخلي (internal representation code) لكل رمز (character) في النظام العشري (decimal) . ولاحظ أن الأرقام التي على يسار الجدول هي الأرقام اليسرى في المكافئ العشري (decimal equivalent) [من 0 إلى 127] لشفرة الرمز (character code) ، بينما الأرقام الموجودة أعلى الجدول هي الأرقام اليمنى في المكافئ العشري لشفرة الرمز . فمثلاً الحرف A يمثل داخلياً بالعدد الصحيح 65 ، بينما 116 هو المكافئ العشري لشفرة الحرف t . ويرمز للفراغ / space) blank character) بالعلامة " □ ".

Right Digit / Left Digits	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	{	|	}	~	del		

مجموعة رموز ASCII

ASCII Character Set

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

دليل المصطلحات العربية والإنجليزية
Index

فيما يلي قائمة بالمصطلحات الإنجليزية مرتبة ترتيباً أبجدياً، والمصطلحات العربية المقابلة لها، وكذلك فهرس لهذه المصطلحات .

interactive	٣٠	تبادلي
inverted scan set	٣٠٧	مجموعة المسح المعكوسة / المقلوبة
iterative technique	٣١٩	طريقة تكرارية
label	١٨٦	علامة
leading zeros	٢٩٥	أصفار رائدة / متقدمة (في البداية على اليسار)
left - justification	٢٨٢	ضبط يسار / محاذاة نحو جهة اليسار
library function	١٦٠	دالة مكتبية
linear search	٢٤٩	البحث الخطي
literal string	٢٤	سلسلة حرفية
loading	١٨٤	تحميل
local variable	١٨٥	متغير محلي
logic error	٦١	خطأ منطقي
logical expression	١٢٦	تعبير منطقي
logical operator	١٢٦	مؤثر منطقي
loop	٦٣	عروة
main function	٢٥	الدالة الرئيسية
manipulate	١٧٣	يعالج
match	٢٩٤	يوائم / يتفق
median	٢٤٢	العنصر الأوسط
memory	٣١	ذاكرة
mode	٢٤٢	المنوال / العنصر الأكثر تكرارا
nested control structures	٧١	بنيات تحكم متداخلة
nested if statements	٥٩	عبارات if المتداخلة
nested loops	٢٢٢	عُرَى متداخلة
newline character	٢٥	رمز السطر الجديد
null character	٢٢٥	الرمز الخالي
null / empty statement	١٢٢	العبارة الخالية / الخاوية
offset	٢٩٢	زحزحة
operand	٣٠	معامَل (كمية مشغَّلة)
operating system	٣١	نظام التشغيل

بعض المراجع عن البرمجة بلغة C

References On C Programming Language

[1] Dale, N.
Programming and Problem Solving with C
Programming and Computer Science with C++
Jones and Bartlett publishers.

[2] H.M. Deitel, P.J Deitel
C How To Program
Prentice-Hall,Inc.

[3] J.R. Hanly, E.B> Koffman
Problem Solving and Program Design in C
Addison Wesley, Inc.

[4] L.H. Miller, A.E. Quilici
The Joy of C
John Wiley and Sons, Inc.

[5] Y. Uckan
Problem Solving Using C
McGraw-Hill Companies, Inc.

كتب للمؤلف
في الرياضيات وعلم الحاسوب

---

١ – برمجة الحاسب بلغة الفورتران ، ط ٤ ، دار القلم .. الكويت ١٩٩٢ .

٢ – مقدمة في نظرية المعلومات ، ط ٢ ، دار القلم .. الكويت ١٩٩٣ .

٣ – الشبكات الرقمية ، دار القلم .. الكويت ١٩٨٦ .

٤ – التحليل العددي ، دار القلم .. الكويت ١٩٨٨ .

٥ – الجبر الخطي ، ط ٢، دار القلم .. الكويت ١٩٩٥ .

٦ – برمجة الحاسب بلغة الباسكال ، ط٢ ، دار القلم .. الكويت ١٩٩٩ .

٧ – البرمجة المتقدمة بلغة الباسكال ، مع د. حمزة رشوان، دار القلم .. الكويت ١٩٩٤.

٨ – الدوائر المتكاملة الرقمية (ترجمة) ، دار القلم .. الكويت ١٩٩٣.

٩ – الخوارزميات والبرمجة بلغة الباسكال ، دار القلم .. الكويت ١٩٩٧.

١٠– بنى المعطيات ، مع د. حمزة رشوان ، دار القلم .. الكويت ١٩٩٨.

١١– الحلول العددية للمعادلات التفاضلية العادية ، دار القلم .. الكويت ٢٠٠٠ .

١٢– الحلول العددية للمعادلات التفاضلية الجزئية ، دار القلم .. الكويت ٢٠٠١ .

١٣– برمجة الحاسب بلغة C++ ، دار القلم .. الكويت ٢٠٠٢ .

١٤– البرمجة المتقدمة بلغة C++ ، دار القلم .. الكويت ٢٠٠٣ .

١٥– الرياضيات المتقطعة في علم الحاسوب ، مكتبة الفلاح .. الكويت ٢٠٠٥ .

١٦– هياكل البيانات بلغة C++ ، مع د. حمزة رشوان ، مكتبة الفلاح .. الكويت ٢٠٠٦ .

١٧– برمجة الحاسب بلغة C ، دار اقرأ .. الكويت ٢٠٠٧ .

١٨ – البرمجة المتقدمة بلغة C ، دار اقرأ .. الكويت ٢٠٠٦ .

﴿ سُبْحَانَكَ لاَ عِلْمَ لَنَا إِلاَّ مَا عَلَّمْتَنَا إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ ﴾

(سورة البقرة :٣٢)

محتويات الكتاب

Computer Programming In C

Dr. Abu_Bakr Ahmad El_Sayed

Department Of Mathematics and Computer Science
University of Kuwait

دار اقـــرأ للنشر والتوزيع
الكويت / حولي - شارع المثني - ص . ب ١٩٣٧ حولي - الرمز البريدي : ٣٢٠٢٠
هاتف / ٢٦٥٥٣٤٠ - ٢٦٥٥٣٥٠ - فاكس / ٢٦٥٥٣٥٠ - نقال / ٧٥٧٩٢٧١
email: dariqraa@hotmail.com - email: dariqraa@yahoo.com
website: www.dar_iqraa.tk