الخوارزميات والبرمجة بلغة الباسكال

الدكتور أبو بكر أحمد السيد

قسم الرياضيات وعلم الحاسوب

جامعة الكويت

دار القلم
الكويت

ALGORITHMS AND PROGRAMMING IN PASCAL

DR. ABU-BAKR AHMAD EL-SAYED

DEPT. OF MATHEMATICS AND COMPUTER SCIENCE
UNIVERSITY OF KUWAIT

الخوارزميات والبرمجة
بلغة الباسكال

الخوارزميات والبرمجة بلغة الباسكال

الدكتور أبو بكر أحمد السيد

قسم الرياضيات وعلم الحاسوب

جامعة الكويت

بسم الله الرحمن الرحيم

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

(سورة البقرة ١٣٨)

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

الطبعة الأولى

١٤١٨هـ - ١٩٩٧م

دار القلم للنشر والتوزيع

شارع السور - عمارة السور - الطابق الأول

هاتف : ٢٤٥٧٤٠٧ - ٢٤٥٨٤٧٨ - برقيا توزيعكو

ص . ب ٢٠١٤٦ الصفاة 13062 الكويت

بسم الله الرحمن الرحيم

المقدمـــة

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

دور التعليم في العالم الإسلامي :

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

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

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

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

خطورة الكتب الأجنبية والمناهج الغربية :

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

---

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

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

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

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

المناهج الدراسية والقيم الإسلامية

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

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

علماء المسلمين الأوائل ومفهوم العبادة

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

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

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

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

الإيمان وطريقة التدريس .. السبيل إلى الحافز العلمي

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

من آثار انتقال العلم من أيدي المسلمين إلى أيدي الأوربيين

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

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

الكتب الأجنبية وجحود فضل علماء المسلمين :

وإن الطالب المسلم عندما يدرس الفيزياء مثلا في كتاب وضعه كاتب غربي فإنه يشعر بخيبة أمل لعدم رؤيته أي إشارة لأعمال العلماء الفيزيائيين المسلمين. إننا نعلم مثلا أن واضع علم البصريات هو الحسن بن الهيثم وليس نيوتن ، ولكننا

دائما نجد اسم نيوتن ، وقد بولغ في تمجيده لتجاربه في علم الضوء ، وبطريقة مماثلة لا نجد ذكرا لعلماء المسلمين في الرياضيات أو الفلك والجغرافيا وغيرها ، وإذا وردت إشارة عابرة لبعض هؤلاء العلماء ، فإنما تذكر مساهمة العلماء العرب (وليس المسلمين). ونلاحظ التركيز دائما على العلماء الغربيين .. ونظرا لأن الطالب المسلم لا يشعر بالترابط العاطفي مع نيوتن وأينشتين وجاليليو فإنه يفقد الكثير من حماسه ، وإذا حاول أن يتلمس طريقه إلى العلم فإنه سوف يتشرب الأفكار الإلحادية والمعادية للدين التي تتخلل جميع العلوم الحديثة النابعة من الغرب [*].

من آثار تدريس العلوم باللغات الأجنبية

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

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

دور علماء المسلمين في تأليف الكتب العلمية

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

منهج هذا الكتاب ومحتوياته

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

---

[*] المصدر السابق.

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 الباسكال.

وهناك صيغة حديثة للغة الباسكال تسمى تربو باسكال ، وقد صممت لتلائم الحواسيب الشخصية ، وهي تتبع قواعد لغة الباسكال القياسية ، وتضيف إليها بعض التعديلات والتحسينات ، ونعرض في أحد ملاحق الكتاب هذه الإضافات والتعديلات.

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 الباسكال بحيث يستطيع الطالب مع نهاية هذا الفصل كتابة برامج قصيرة ، ويشتمل الفصل الثالث على عبارات التحكم للانتقال من أي خطوة إلى خطوة أخرى في البرنامج وكيفية كتابة شروط هذا الانتقال ... وأما الفصل الرابع عن عبارات التكرار فيشرح كيفية تكرير تنفيذ مجموعة من التعليمات والأوامر عددا معلوما من المرات ، أو إلى أن يتحقق شرط معين. ويتناول الفصل الخامس المتغيرات المؤشرة والمنظومات أي المجموعات المرتبة : الخطية (أي ذوات البعد الواحد أو المتجهات) وذوات البعدين (أي المصفوفات). أما الفصل السادس والأخير فهو خاص بالبرامج الفرعية وأنواعها المختلفة واستخداماتها.

وهناك ثلاثة ملاحق للكتاب : الأول منها خاص بالأنواع البسيطة للبيانات ، وهي تشمل كلا من البيانات الحقيقية ، والبيانات الترتيبية ، وهذه بدورها تشمل البيانات القياسية (من صحيحة ورمزية ومنطقية) ، والبيانات المعرفة بالمستخدم (من تعددية وجزئية). والملحق الثاني خاص بلغة التيربو باسكال ، والملحق الثالث خاص بإحدى العبارات قليلة الاستخدام في لغة الباسكال.

ونظرا لأن المهارة والخبرة في البرمجة تحتاج لكثرة التدريب على كتابة برامج في تطبيقات مختلفة ، لذا كان التركيز في هذا الكتاب على تنوع وتعدد الأمثلة المحلولة ، ومسائل التمرينات في نهاية كل فصل ، ثم مسائل التمرينات العامة على فصول الكتاب ، ثم مسائل الاختبارات العامة الفصلية والنهائية على كافة محتويات المنهج ، مع وجود حلول كاملة لتمرينات كل فصل في نهاية الكتاب وكذلك حلول كاملة للاختبارات ، وذلك ليرجع إليها الطالب بعد محاولته حل المسائل إتماما للفائدة بإذن الله تعالى.

شكر وتقدير

ونود هنا أن نتقدم بشكرنا الجزيل لدار القلم للنشر والتوزيع بالكويت لتعاونها المستمر معنا في نشر الكتب الدراسية الجامعية التي تخدم بإذن الله تعالى

قضية أسلمة العلوم الحديثة وتعريبها ، والتي نراها قضية حياة أو موت بالنسبة لهذه الأمة ، فجزى الله القائمين على هذه الدار خيرا على ما يقومون به في هذا السبيل من تقديم التسهيلات ، وتسخير الدار لتلك الغاية النبيلة.

ولا يفوتنا أن نشكر الأخ الفاضل الأستاذ / السيد البدوي محمد أحمد بقسم الرياضيات وعلم الحاسوب بجامعة الكويت على ما بذله من جهد كبير وما تحلى به من صبر وأناة في طباعة هذا الكتاب حتى يخرج بهذه الصورة الطيبة التي تخدم عملية التعريب وتيسر نشر العلوم في آفاق أمتنا لتعود كما كانت في مكانة الصدارة بين الأمم. جعل الله ذلك الجهد في ميزان حسناته لمساهمته في تيسير سبل طلب العلم والتماسه.

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

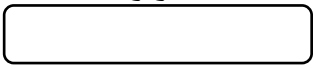
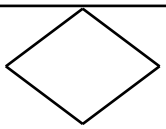
الفصل الأول

الخوارزميات وخرائط سير العمليات
Algorithms and Flowcharts

مفهوم الخوارزميات وخرائط سير العمليات :

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

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

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

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

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

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

خطوات حل مسألة باستخدام الحاسب :

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

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

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

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

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

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

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

أمثلة :

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

مثال ١-١ :

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

تعريف المسألة رياضيا :

نفرض أن المعادلة هي :

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

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

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

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

القانون :

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

مع تحقق الشرط

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

الخوارزمية :

١- ابدأ.

٢- اقرأ قيم المعاملات a , b , c.

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

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

٥- احسب قيمة الجذر الأول

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

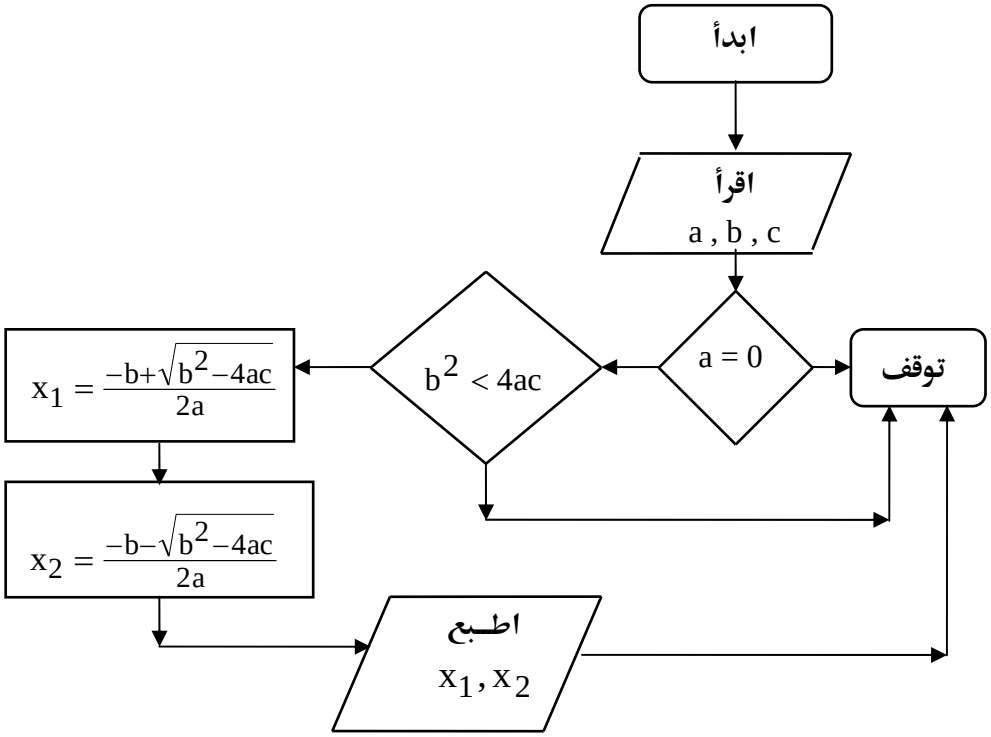
٦- احسب قيمة الجذر الثاني

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

٧- اطبع قيمة كل من الجذرين x_1 , x_2.

٨- توقف.

خريطة سير العمليات



ملاحظتان :

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

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

مثال ١-٢ :

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

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

(ب) **قيمة النصاب** [*] A

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

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

تعريف المسألة رياضيا :

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

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

القانون المستخدم : **إذا تحقق الشرط** $TAS \geq A$ $Z = TAS/40$

إذا تحقق الشرط $TAS < A$ $Z = 0$

الخوارزمية :

١- **اقرأ قيمة كل من** TAS , A

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

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

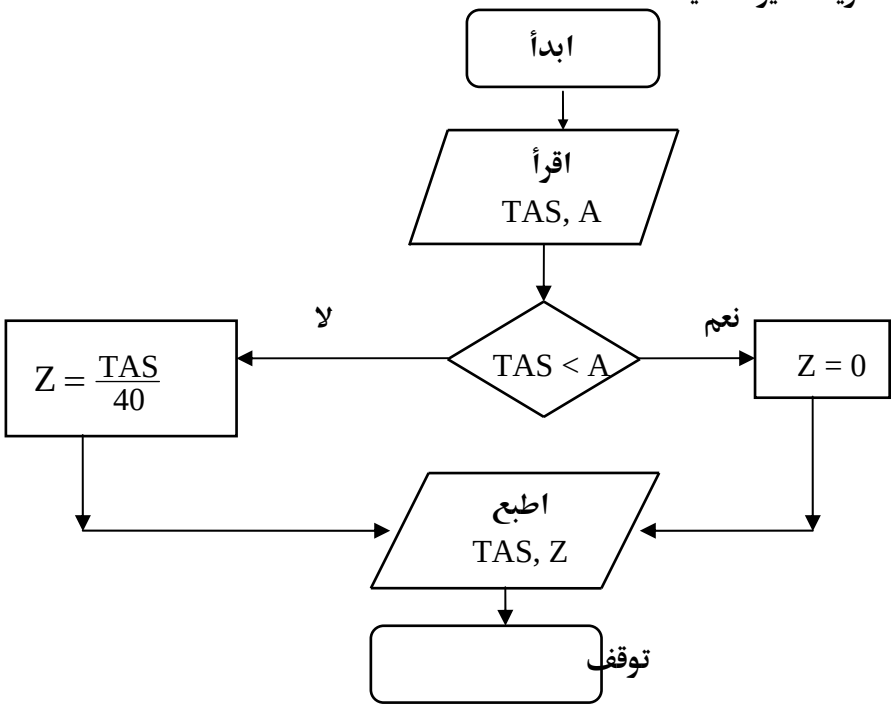
٤- **اطبع قيمة كل من** TAS , Z

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** $x = 2$

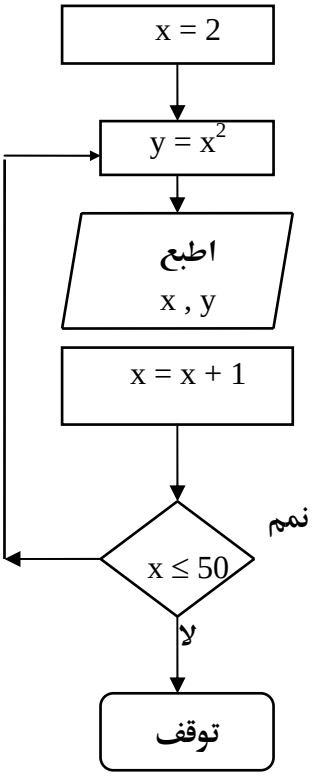
٢- **اجعل** $y = x^2$

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

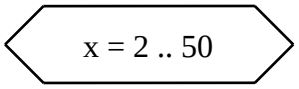
لتصبح $x = 2 + 1 = 3$ ولذلك فإن الشرط $x \leq 50$ المذكور في الخطوة الخامسة يكون متحققا ولذلك ينتقل إلى الخطوة رقم ٢ ليحسب قيمة جديدة للمتغير y حسب العلاقة $y = 3^2 = 9$.

- 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 قيمـة x

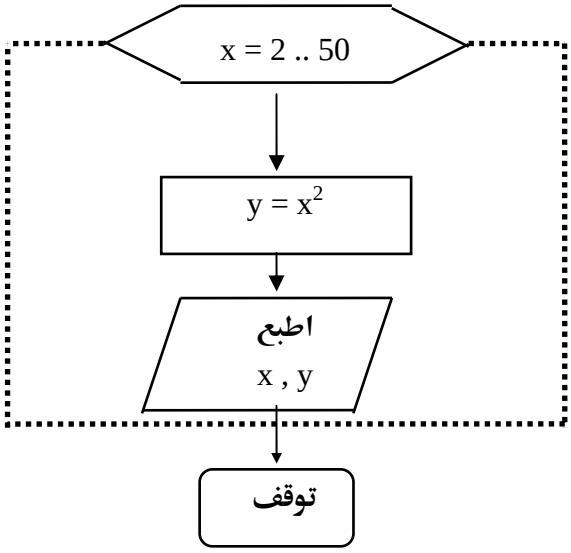
$$x = 3 + 1 = 4$$

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 :

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

الخوارزمية :

١- **اجعل** $S = 0$

٢- **اجعل** $X = 2$

٣- **اجعل** $S = S + x^2$

٤- **اجعل** $x = x + 1$

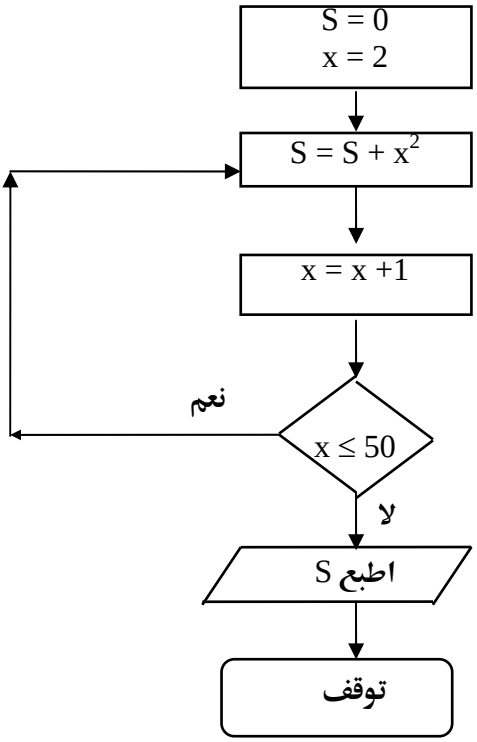
٥- **إذا كانت** $x \le 50$ **اذهب إلى الخطوة رقم ٣**

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 :

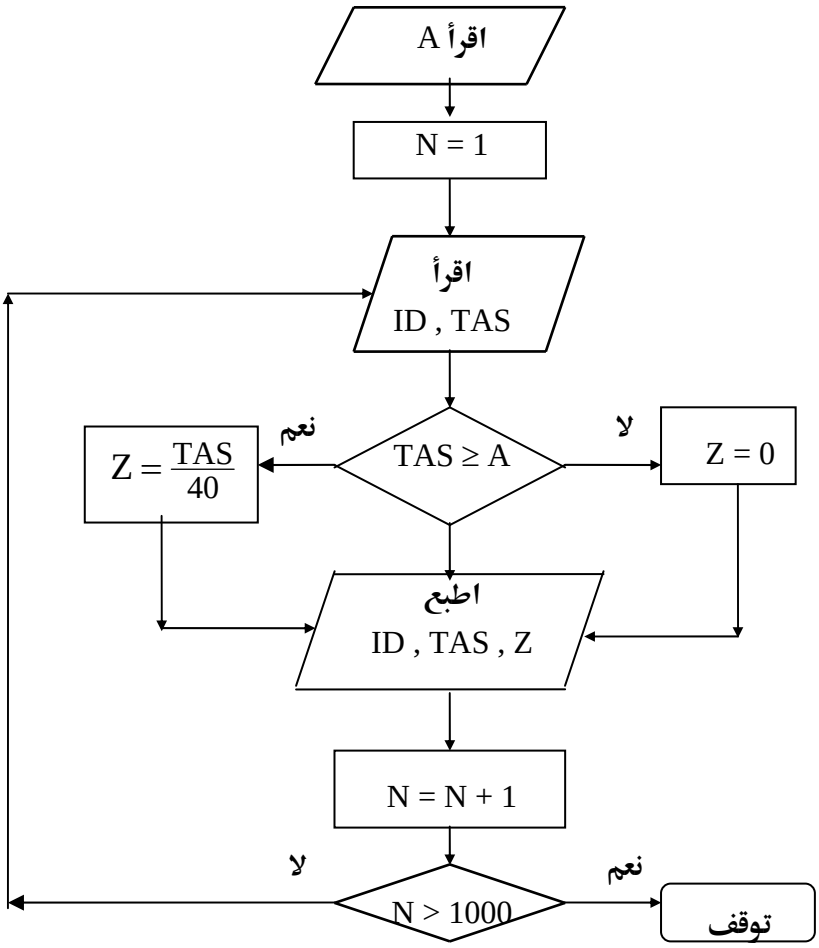
(أ) قراءة قيمة ID (رقم تعريفي للشخص identification Number**) وقيمة** TAS **(المدخرات السنوية الكلية).**

(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 :

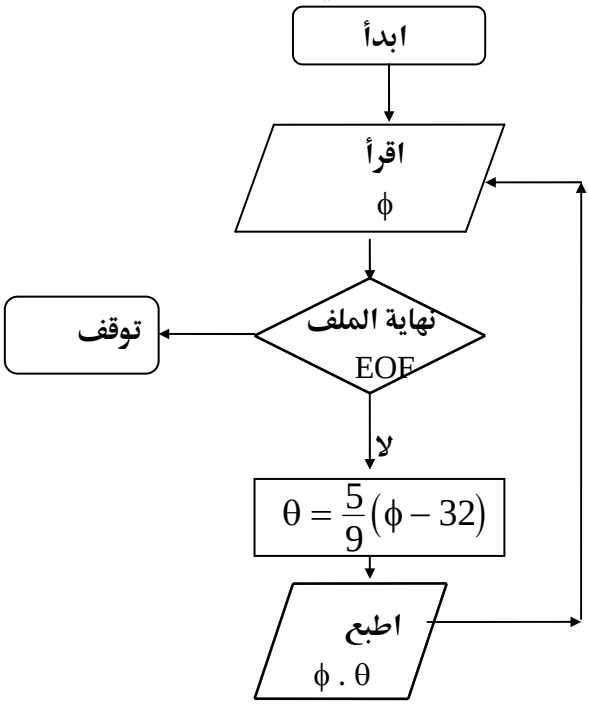
أعطيت مجموعة من البيانات على سطور أو بطاقات بحيث أن كل بطاقة عليها درجة حرارة ϕ بالتقدير الفهرنهيتي. ارسم خريطة سير عمليات لبرنامج يقرأ كل درجة ϕ ثم يحولها إلى التقدير المئوي θ حسب العلاقة

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

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 ZS , N.

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

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

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

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

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

العدد الأول I : هو رقم تعريفي للأسرة.

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

ارسم خريطة سير عمليات لبرنامج يقرأ هذه البيانات ويحسب :

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

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

١-٤ نفرض أنه قد أُعدت بطاقة لكل كتاب في معرض الكتاب الإسلامي بحيث تحتوي البطاقة على بعض البيانات الخاصة بالكتاب ومن 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.

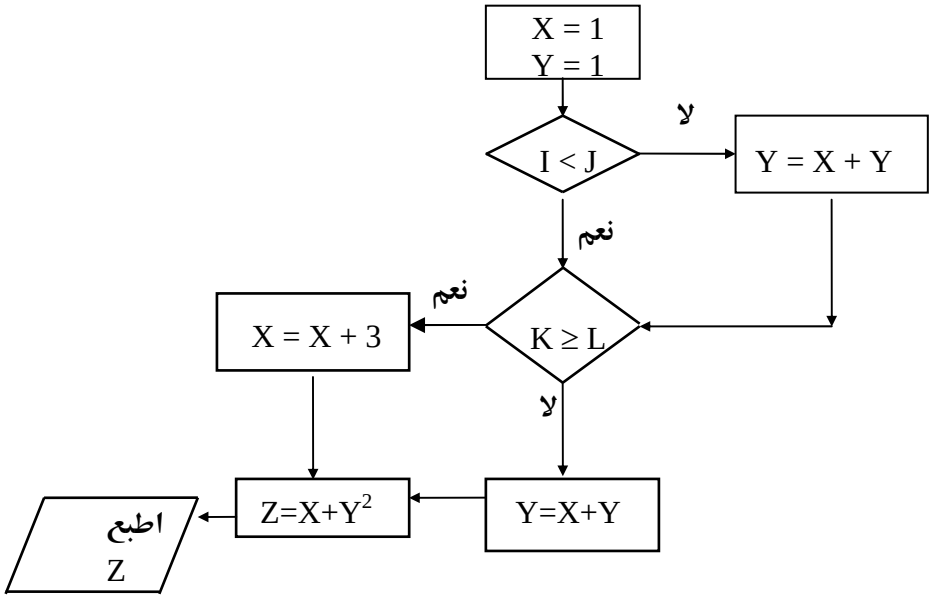
(ب) عدد كتب التفسير والحديث M.

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.

٦-١ 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	



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

$$S = \frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + L + \frac{1}{115} \quad (a)$$

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + L + \frac{99}{100} \quad \text{(b)}$$

$$S = 2^3 + 4^3 + 6^3 + L + 300^3 \quad \text{(i)}$$

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

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

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

حيث تعطى y بدلالة المتغير θ بالعلاقة

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

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

ارسم خريطة سير عمليات توضح خطوات هذه الطريقة.

١- اقرأ قيم a , b , ε.

٢- احسب $c = \frac{a+b}{2}$.

٣- احسب قيمة f(c).

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

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

٦- اذهب إلى الخطوة رقم ٢.

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

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

ارسم خريطة سير عمليات توضح خطوات هذه الطريقة.

١- اقرأ قيم x_0 , ε , n_{max}

٢- **اجعل** $n = 0$

٣- **اجعل** $x = x_0$

٤- **احسب** $\Delta = \frac{f(x)}{f'(x)}$

٥- **احسب** $x = x - \Delta$

٦- **اجعل** $n = n + 1$

٧- **اطبع قيمة كل من** x , n.

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

٩- **ارجع إلى الخطوة رقم ٤.**

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

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

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

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

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

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

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

ارسم خريطة سير عمليات

(أ) لقراءة قيمتي T , R لكل طالب.

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

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

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

١- اجعل $I = 1 , N = 0$

٢- اقرأ قيمة A

٣- إذا كان $A > 0$ اجعل $N = N + 1$

٤- اجعل $I = I + 1$

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

٦- اذهب إلى الخطوة رقم ٢

٧- اطبع قيمة N

٨- توقف.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

المناظرة لقيم x التالية :

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

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

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

$1 , 1 + A , 1 + 2A , 1 + 3A , ... , 1 + 49A$

١-٢١ ارسم خريطة سير عمليات برنامج لحل المعادلتين الخطيتين

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

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

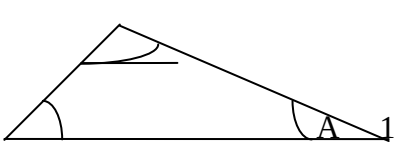
$$x = \frac{b_2c_1 - b_1c_2}{a_1b_2 - a_2b_1} \quad , \quad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

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

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

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

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



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

(S **يمثل نصف المحيط).**

(ii) **محيطه** P $(P = a + b + c)$

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

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

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 عروض التجارية ZT.

د) يطبع قيمة كل من القيمة السوقية والزكاة.

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

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

الفصل الثاني

أساسيات لغة الباسكال
Fundamentals of Pascal

الباسكال [*] هي إحدى اللغات المستخدمة لكتابة برامج يمكن إدخالها للحاسب لحل مسائل معينة. ونبدأ بعون الله تعالى فيما يلي بعرض أبجديات هذه اللغة.

العناصر الأساسية في لغة الباسكال :

تتكون مفردات هذه اللغة من العناصر أو الرموز (characters) **الأساسية التالية :**

(أ) الأرقام العربية العشرة من 0 إلى 9 (Digits) **:**

0 , 1 , 2 , 3 , ... , 8 , 9

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

A , B , C , ... , Y , Z

a , b , c , ... , y , z

(أ)بعض الحروف الخاصة (Special Symbols) **:**

```
(   )   {   }   [   ]
+   -   *   /   '

=   <   >   <=   >=   <>
.   ..  :   :=   ,    ;
```

ويلاحظ أن الفراغ (space) **هو أحد الرموز الخاصة. وهناك رموز أخرى قد توجد في بعض النظم ، مثل :**

\# $ ^

---

[*] ظهر أول تقرير عن هذه اللغة (Pascal, User manual and report) عام ١٩٧١ وقد سميت بهذا الاسم نسبة إلى العالم الرياضي باسكال الذي ابتكر إحدى الآلات الحاسبة الميكانيكية في منتصف القرن السابع عشر.

ومن المفردات السابقة يمكن تركيب كلمات وتعابير اللغة تبعا لقواعد خاصـة ، كما سنرى بإذن الله تعالى فيما يلي :

أولا : الثوابت (Constants) :

هي التي تحتفظ بقيمها دون أي تغيير طوال تنفيذ البرنامج.

والثوابت تنقسم إلى نوعين رئيسيين :

(أ) ثوابت عددية (Numeric Constants).

(ب) ثوابت غير عددية (Non Numeric Constants).

وتسمى أيضا ثوابت رمزية (String Constants).

١- الثوابت العددية :

الثابت العددي هو عدد يظهر بصورته الصريحة أي بقيمته التي لا تتغير

مثل :

4.795 -12.3 4

ويمكن أن تسبقه إشارة أو يكون بدون إشارة.

والثابت العددي في لغة الباسكال هو أحد نوعين :

(أ) **ثابت صحيح** (Integer Constant) :

وهو عدد صحيح بدون أي فاصلة عشرية أو أي رموز أخرى وقد يكون عددا موجبا أو سالبا أو صفرا.

أمثلة : 314- 300+ 5 0

وتختلف أكبر قيمة مطلقة مسموح بها للثابت الصحيح من حاسب لآخر.

(ب) ثابت حقيقي (Real Constant) :

وهو عدد نسبي حقيقي بالمفهوم الرياضي ، ويختلف المدى المسموح به لهذه الأعداد باختلاف الحاسب (مثلا في بعض النظم تكون القيم في المدى من حوالي $١٠^{-٤٠}$ إلى حوالي $١٠^{٤٠}$). ويمكن كتابة الثابت الحقيقي بإحدى صورتين :

(i) **الصورة العشرية** (Decimal Form) **:**

وهي عبارة عن عدة أرقام متتالية مع وجود الفاصلة العشرية ومع ضرورة وجود رقم واحد على الأقل على كل من يمين الفاصلة العشرية وشمالها.

أمثلة :

14.1 +5.0 5.00 13.25

ويختلف عدد الأرقام المسموح بها يمين العلامة العشرية من نظام لآخر (سبعة أرقام أو ثمانية ...).

(ii) **الصورة الأسية** (Exponential Form) **:**

وهي عبارة عن ثابت (صحيح أو حقيقي في الصورة العشرية) عن يمينه الحرف E **(إشارة إلى الأس** Exponent**) ، وعن يمين هذا الحرف عدد صحيح مكون من رقم أو رقمين ، وقد يكون له إشارة أو بدون إشارة (مثل** 3.2E5 **).**

معنى هذه الصورة : الثابت يضرب في عشرة مرفوعة لأس العدد الصحيح (القوة).

أمثلة :

3.2E5	**وتعني رياضيا**	3.2×10^{5}
4E-12	**وتعني رياضيا**	4×10^{-12}
-50E + 6	**وتعني رياضيا**	-5×10^{6}

ولاحظ مرة أخرى وجود رقم واحد على الأقل يمين الفاصلة العشرية – إن وجدت – في الصورة الأسية ، ورقم آخر على الأقل عن شمالها. وعموما تفيد الصورة الأسية ، والتي تسمى أيضا الصورة العلمية (Scientific Form) **في تمثيل الأعداد الكبيرة جدا والأعداد الصغيرة جدا.**

مثال ٢–١ :

فيما يلي مجموعة من المقادير ، والمطلوب بيان الثوابت الصحيحة منها والثوابت الحقيقية ، وما لا يقبل كثابت في لغة الباسكال مع بيان السبب.

(i) -12,000 (ii) 2.34 E 98 (iii) 5 E 4.5
(iv) +2515 (v) -6. E +23 (vi) E +15

(vii) 1234567891234 (viii) 24.123456 (ix) + .034
(x) -12.0

الحل :

(i) **غير مقبول لوجود الفاصلة.**

(ii) **غير مقبول لتجاوزه الحد الأعلى المسموح به لقيمة الثابت في معظم لغات الباسكال.**

(iii) **غير مقبول لأن القوة (العدد عن يمين حر E) يجب أن تكون عددا صحيحا.**

(iv) **ثابت صحيح.**

(v) **غير مقبول لعدم وجود رقم يمين الفاصلة العشرية.**

(vi) **غير مقبول لأنه غير مسموح بوجود القوة (الأس) بمفردها ولكن يسمح بكتابة** 1E + 15.

(vii) **غير مقبول لتجاوزه الحد الأعلى المسموح به لعدد الأرقام المعنوية.**

(viii) **ثابت حقيقي.**

(ix) **غير مقبول لعدم وجود رقم شمال الفاصلة العشرية.**

(x) **ثابت حقيقي.**

٢- الثوابت غير العددية (الثوابت الرمزية أو الأسماء) :

(String Constants)

(ونقصد بكلمة الرمزية أي المكونة من رموز وهي : الحروف والأرقام والرموز الخاصة).

الثابت غير العددي أو الثابت الرمزي هو سلسلة (String) **(أي مجموعة متتابعة) من الرموز** (Characters) **(الحروف والأرقام والرموز الخاصة) المسموح بها في لغة الباسكال بين حاصرتين علويتين** (Apostrophes) **، مثل :**

'Bilal'
'Omar Ibn Abdel Azeez'
'Omm Salama'

ويعرف طول الثابت الرمزي بأنه عدد الرموز المحصورة بين الحاصرتين العلويتين بما في ذلك الفراغات لأنها من الرموز المسموح بـها ، فمثلا أطوال

الثوابت الثلاثة السابقة هي على الترتيب 5 ثم 20 ثم 10 . وإذا كان الثابت الرمزي يشتمل على الرمز الخاص (′) أي الحاصرة العلوية (Apostrophe) **– والتي قد تكون جزءا من الاسم أو تفيد الاختصار أو الملكية – فإنها تكتب حاصرتين متتاليتين ، مثل :**

′ Ka ′ ′b Ibn Malik ′
′ I ′ ′ M Muslim ′
′ Khalid ′ ′ s Sword ′

ويلاحظ أنه عند حساب الطول أي حساب عدد الرموز فإن الحاصرتين المتتاليتين تحسبان كرمز واحد.

ويمكن استخدام الثوابت الرمزية كعناوين توضيحية أو تعريفية لنتائج البرنامج ، مثل :

′ Table of Zakat ′
′ Zakat of Money = ′
′ The Solution is :
′ Student ′ ′ s Average = ′

وعند طباعة مخرجات البرنامج (Output) **لا تظهر (أي لا تطبع) الحاصرتان العلويتان المتطرفتان (أقصى اليمين وأقصى اليسار) ، وتطبع الحاصرتان المتتاليتان حاصرة واحدة ، فمثلا يظهر العنوان الأول هكذا :**

Table of Zakat

ويظهر العنوان الأخير هكذا :

Student′s Average =

وعموما يكتب الثابت الرمزي على سطر واحد فقط.

وعند استخدام الثوابت في البرنامج يُعطَي الثابت – سواء كان عدديا أم رمزيا – اسما تعريفيا (Identifier) **ونعرِّف البرنامج في مطلعه بأن هذا الاسم سيمثل ثابتا عن طريق استخدام كلمة محجوزة** (Reserved Word) **وهي كلمة** Const **، فنكتب مثلا :**

```
Const
    Zakat Percent = 2.5 ;
    Date = ′ 17 Ramadan , 2 a.h. ′ ;
```

ولذلك نشير فيما يلي إلى الأسماء التعريفية والكلمات المحجوزة.

الأسماء التعريفية (Identifiers) **:**

الاسم التعريفي هو اسم يعطى لتعريف ثابت أو متغير أو منظومة أو دالة ... الخ في البرنامج. ويتكون الاسم التعريفي من حروف وأرقام متتالية (Alphanumeric Characters) **، على أن يبدأ بحرف وليس برقم. وهذه الحروف الأبجدية** (Alphabetic letters) **المستخدمة في تكوين الأسماء يمكن أن تكون حروفا كبيرة** (Upper case letters) **أو حروفا صغيرة** (Lower case letters) **دون أي فارق بالنسبة للبرنامج ، بل إن حروف الاسم الواحد قد يكون بعضها كبيرا والبعض الآخر صغيرا. ويختلف أقصى طول مسموح به للاسم التعريفي من نظام لآخر ، فيصل إلى ثمانية في بعض النظم ويزيد عن مائة في البعض الآخر. وفيما يلي أمثلة لبعض الأسماء التعريفية :**

ZAKAT	Zakat	AlNisab	X1	Z20
Sum	y5	root	NisabSheep	XY345
TAS	K	number30		

مثال ٢-٢ :

أي الأسماء التالية تعتبر أسماء تعريفية صحيحة أي مقبولة ، وأيها تعتبر غير مقبولة ، ولماذا ؟

(i)	Alzakat	(ii)	X12.4	(iii)	R (2)
(iv)	12XY	(v)	MAX	(vi)	I / 6
(vii)	α β 3	(viii)	Total sum	(ix)	$\overline{JJX}$
(x)	Y-12	(xi)	Totalsum		

الحل :

(i) **مقبول.**

(ii) **غير مقبول – النقطة غير مسموح بها ، ليست حرفا ولا رقما.**

(iii) **غير مقبول – الأقواس غير مسموح بها.**

(iv) **غير مقبول – لا يبدأ بحرف.**

(v) **مقبول.**

(vi) غير مقبول – الرمز الخاص ((/)) غير مسموح به.

(vii) غير مقبول – حروف لاتينية غير مسموح بها.

(viii) غير مقبول – الفراغ غير مسموح به.

(ix) غير مقبول – الرمز الخاص (الشرطة) غير مسموح به.

(x) غير مقبول – الرمز الخاص ((–)) غير مسموح به.

(xi) مقبول.

الأسماء التعريفية القياسية (Standard Identifiers):

هي أسماء تعريفية لها معان محددة معروفة للحاسب سلفا ولا نحتاج لتعريفها له ، مثل أسماء الدوال :

المثلثية :			ARCTAN	COS	SIN
والأسية :					EXP
واللوغاريتمية :					LN
والدوال :	SQRT	SQR		ROUND	ABS

وعموما يمكن استخدام هذه الأسماء القياسية لغير معانيها المحددة ، ويلزم في هذه الحالة إعادة تعريفها في البرنامج لتحديد معانيها الجديدة ، ولكن من الأفضل اجتناب هذا واحتفاظ هذه الأسماء بمعانيها درءا لأي التباس.

الكلمات المحجوزة (Reserved Words) :

هي كلما قياسية معرفة للحاسب لا تحتاج لتعريف ولها دلالات محددة تؤدي وظائف خاصة ، مثل

PROGRAM	BEGIN	END
AND	OR	NOT
IF	THEN	ELSE
CONST	VAR	TYPE

وتوجد بنهاية الكتاب قائمة بهذه الكلمات المحجوزة ، وهذه الكلمات لا يجوز استخدامها لغير دلالاتها المحددة أو إعادة تعريفها.

تعريف الأسماء التعريفية للثوابت (Constant Identifiers) **:**

جميع الأسماء التعريفية المستخدمة في البرنامج لتعريف الثوابت العددية والرمزية يجب أن تعرف للحاسب في مطلع البرنامج في جزء خاص بالتعريفات (Definition Part) **، وذلك بأن نكتب الكلمة المحجوزة** CONST **تليها هذه الأسماء التعريفية متعاقبة ، مع كتابة فاصلة منقوطة بين كل اسمين متتاليين ، وبعد آخر اسم ، مثل :**

```
CONST
        NisabSheep = 40 ; Max = 30 ;
                PI = 3.14159 ;
        FirstBattle = ' BADR ' ;
```

أي أن الصيغة العامة لتعريف الثوابت في البرنامج هي :

CONST $n_1 = c_1$; $n_2 = c_2$; ... ; $n_i = c_i$; ... ; $n_k = c_k$;

حيث :

n_i : **اسم تعريفي للثابت رقم** i **، والثابت عددي أو رمزي.**

c_i : **قيمة الثابت رقم** i.

وعموما يجب ملاحظة أن تظل قيم هذه الثوابت المعرفة ثابتة دون أي تغيير طوال تنفيذ البرنامج ، سواء كانت الثوابت عددية أو رمزية.

ملاحظة :

يوجد ثابت قياسي (Standard Constant) **اسمه** MAXINT **تمثل قيمته أكبر قيمة مسموح بها للأعداد الصحيحة.**

ثانيا : المتغيرات (Variables) **:**

المتغير هو كمية تعطَي اسما تعريفيا معينا مثل X **أو** Y **ويسمح لها بالتغير أي بأخذ قيم مختلفة أثناء تنفيذ البرنامج. والمتغيرات لها عدة أنواع:**

١- **المتغير الصحيح** (Integer Variable)

وهو المتغير الذي يسمح له بأخذ قيمة ثابت عددي صحيح فقط.

٢- **المتغير الحقيقي** (Real Variable)

وهو المتغير الذي يسمح له بأخذ قيمة ثابت عددي حقيقي.

٣- **المتغير الرمزي** (Character Variable)

وهو المتغير الذي يسمح له بأخذ قيمة رمزية (وسنرجع إلى هذا النوع من المتغيرات ببعض التفصيل فيما بعد بإذن الله).

٤- **المتغير المنطقي** (Boolean Variable)

وهو المتغير الذي يسمح له بأخذ قيمة منطقية (صحيح : True ، أو غير صحيح : False ، وسنشير أيضا إلى هذا النوع من المتغيرات فيما بعد).

تعريف الأسماء التعريفية للمتغيرات (Variable Identifiers)

عموما يُختار الاسم التعريفي للمتغير بحيث يساعد على تذكر معنى أو وظيفة ومهمة المتغير ، ولكن يجب تعريف الحاسب بنوع هذا المتغير ، وذلك بذكر اسم المتغير ونوعه في مطلع البرنامج في الجزء المخصص للإعلان عن عناصر البرنامج ومكوناته ، كما في المثال التالي :

```
VAR
    N1, N2, COUNT : INTEGER ;
    MONEY, Nisab , Zakat : REAL ;
    Name, ID : CHAR ;
    Switch : BOOLEAN ;
```

أي أن الصيغة العامة لتعريف أو الإعلان (Declaration) **عن جميع الأسماء التعريفية للمتغيرات الواردة في البرنامج وأنواعها هي :**

```
VAR
v1 , v2 , ... , vn : t1 ;
w1 , w2 , ... , wm : t2 ;
```

حيث $v_1,..., v_n$ **هي الأسماء التعريفية للمتغيرات من النوع** t_1 ،

$w_1,..., w_m$ **هي الأسماء التعريفية للمتغيرات من النوع** t_2 ، **وهكذا.**

وكما ذكرنا يأتي هذا الإعلان عن أسماء المتغيرات وأنواعها في مطلع البرنامج قبل استخدم هذه المتغيرات.

ثالثا : العمليات الحسابية (Arithmetic Operations) **:**

توجد أربع عمليات حسابية أساسية ، وهذه العمليات مع الرموز المقابلة لها في لغة الباسكال هي :

١-	الجمع	:	+	فمثلا	a + b	تعني	a + b
٢-	الطرح	:	-	فمثلا	a - b	تعني	a - b
٣-	الضرب	:	*	فمثلا	a * b	تعني	a . b
٤-	القسمة	:	/	فمثلا	a / b	تعني	b÷a

وبالنسبة للعمليات الثلاث الأولى * ، - , + فإنها تعطي نتيجة من النوع الصحيح إذا كانت كل من الكميتين a , b **من النوع الصحيح. أما إن كانت إحداهما أو كلاهما من النوع الحقيقي فإن النتيجة تكون من النوع الحقيقي. أما بالنسبة للعملية الرابعة : القسمة / فإنها تعطي دائما نتيجة من النوع الحقيقي حتى لو كان كل من الكميتين** a , b **من النوع الصحيح. وإذا أردنا في هذه الحالة الأخيرة (حالة قسمة عدد صحيح على عدد صحيح) الحصول على نتيجة من النوع الصحيح أي على عدد صحيح هو الجزء الصحيح من خارج القسمة الناتج ، فإننا نستخدم رمزا أو معاملا** (operator) **آخر هو** DIV.

تعريف :

عملية القسمة a Div b **تعطي عددا صحيحا هو الجزء الصحيح لخارج القسمة الناتج من قسمة** a ÷ b.

فمثلا :

20 Div 5 = 4 20 Div 8 = 2 ,

ولا يجوز استعمال هذا المؤثر (operator) **أي المؤثر** Div **إلا إذا كانت كل من الكميتين** a , b **من النوع الصحيح ، ولذلك فهو يسمي معامل القسمة الصحيحة** (Integer division operator).

وأما العدد الصحيح الباقي من القسمة a ÷ b **في هذه الحالة فيعطيه مؤثر حسابي آخر هو** MOD **(اختصار** Modulo).

تعريف :

العملية a MOD b **تعطي العدد الصحيح الباقي من قسمة الكمية الصحيحة** a **على الكمية الصحيحة** b.

أي أن المؤثر MOD **لا يستخدم أيضا إلا مع الكميات من النوع الصحيح.**

مثلا :

20 Div 5 = 0 20 Div 8 = 4 ,

ويمكننا تلخيص الملاحظات السابقة على أنواع الكميات الداخلة في العمليات الحسابية المذكورة ونتائجها مع المؤثرات المختلفة في الجدول التالي :

الكميات		المؤثر □		
a	b	+ , - , *	/	Div, MOD
صحيح	صحيح	صحيح	حقيقي	صحيح
صحيح	حقيقي	حقيقي	حقيقي	لا يجوز
حقيقي	صحيح	حقيقي	حقيقي	لا يجوز
حقيقي	حقيقي	حقيقي	حقيقي	لا يجوز

جدول نوع ناتج العملية الحسابية a □ b **المقابل لنوعي الكميتين** a , b

ملاحظات :

إذا كانت العملية الحسابية المطلوب إجراؤها هي عملية قسمة ، وكانت الكمية b **التي نريد القسمة عليها تساوي صفرا ، فإن ناتج عملية القسمة يكون غير معرف ، وبالتالي ففي هذه الحالة يكون ناتج كل من العمليات :** a / b , a MOD b , a DIV b **غير معرف.**

١- **إذا كانت** b **سالبة فإن العملية** a MOD b **تكون غير معرفة.**

٢- **لا يوجد في لغة الباسكال مؤثر حسابي خاص بعملية الرفع إلى أس (مثل :** a^b**) ، كما هو موجود في بعض اللغات الأخرى ، ولكن تجرى هذه العملية باستخدام الدالة اللوغاريتمية** LN **والدالة الأسية** EXP **[وسندرس موضوع الدوال واستخدامها فيما بعد] مع الاستعانة بالمتطابقة الرياضية :**

$$x = e^{\ln x}$$

أي أن : $a^b = e^{\ln(a^b)} = e^{b \ln a}$

ولذلك فإن a^b تترجم إلى :

Exp (b * LN (a)) ; a > 0

رابعا : التعابير الحسابية (Arithmetic Expressions) **:**

التعبير الحسابي عامة هو إجراء لعمليات حسابية على بعض الثوابت والمتغيرات والدوال (وسنتعرض لتعريف الدوال فيما بعد بإذن الله) أو بعض هذه الكميات ، وذلك لإيجاد قيمة عددية ، والتعبير الحسابي في صورته الخاصة البسيطة يتكون من ثابت واحد أو متغير واحد أو دالة واحدة.

أمثلة للتعابير الحسابية :

Y) DIV 3 , t - R3 Exp (a)–A / B * (x + 12.5) , Y , (X

قواعد كتابة وحساب قيمة التعبير الحسابي :

(أ) يجب ألا يظهر رمزان من رموز العمليات الحسابية بجانب بعضهما البعض مباشرة. فمثلا من الخطأ أن يترجم التعبير الجبري $\frac{A}{-B}$ إلى الباسكال بالتعبير الحسابي A/-B حيث ظهر الرمزان – ، / بجانب بعضهما دون أي فاصل بينهما ولكن يمكن أن يترجم إلى A / (-B) حيث استخدمنا القوس للفصل ، أو أن يترجم إلى A/B- وهذا يعطي القيمة المطلوبة نفسها.

(ب) قاعدة الأولوية : Rules of Priority (Order of Precedence)

يتم تنفيذ العمليات الحسابية وفق تسلسل يتبع قاعدة الأولوية والتي يمكن صياغتها كما يلي :

١- في حالة وجود قوسين ( ) فأولوية التنفيذ تكون لما هو داخل القوسين.

٢- تعطي عمليات الضرب × والقسمة / , MOD , DIV الأولوية بين العمليات الحسابية ، تليها عمليات الجمع + والطرح – .

٣- العمليات الحسابية التي لها الأولوية نفسها (مثل الضرب والقسمة) تنفذ على الترتيب من اليسار إلى اليمين.

٤- في حالة وجود (عدة) أقواس داخلية وخارجية ((( ))) فتعطى الأولوية لما هو داخل الأقواس الداخلية ثم الخارجية التي تليها ثم التي تليها وهكذا.

مثال ٢-٣ :

(لتوضيح قاعدة الأولوية) : الجدول التالي يعطي مجموعة من التعابير الحسابية في لغة الباسكال والتعابير الجبرية المقابلة لها ورقم البند من قاعدة الأولوية المذكورة سابقا الذي ينطبق في حالة كل تعبير حسابي.

التعبير الحسابي في الباسكال	رقم بند قاعدة الألوية	التعبير الجبري المقابل
a / (b * c)	١	$\frac{a}{b.c}$
a + b / c	٢	$a + \frac{b}{c}$
a * b - c	٢	a . b - c
a / b * c	٣	$\frac{a}{b}.c$
a * b / c	٣	$\frac{a.b}{c}$
a / b / c	٣	$\frac{\frac{a}{b}}{c} = \frac{a}{b.c}$
a - b + c	٣	a - b + c
a / (b * (c - d) + e)	٤ ، ٢ ، ١	$\frac{a}{b(c-d)+e}$

مثال ٢-٤ :

لتوضيح إمكانية الاستفادة من قاعدة الأولوية في حل بعض المشكلات التي تظهر أثناء تنفيذ بعض العمليات الحسابية في الحاسب ، نفرض أن المطلوب حساب قيمة التعبير A * B / C حيث قيم المتغيرات A , B , C كلها في حدود 10^{40} . تبعا لقاعدة الأولوية (البند ٣) فإن عملية الضرب A * B تنفذ أولا وهذه

تعطي قيمة كبيرة جدا في حدود 10^{80} (أكبر من الحد المسموح به وهو فرضا حوالي 10^{50}) ، وفي هذه الحالة فإن الحاسب إما أن يوقف تنفيذ البرنامج أو يعطي نتيجة خاطئة ، وكحل بسيط لهذه المشكلة يمكن وضع قوسين هكذا A * (B / C) مما لا يغير من قيمة المقدار ولكن يجعل عملية القسمة B / C تنفذ أولا (تبعا للبند ١ في قاعدة الأولوية) وهذه العملية تعطي نتيجة في حدود مدى القيم المسموح به ، وكذلك تكون النتيجة النهائية مقبولة.

خامسا : الدوال المُعَرَّفة (Predefined Functions) :

في التعابير الحسابية في لغة الباسكال يمكن استخدام بعض الدوال الرياضية المعلومة كالجذر التربيعي والقيمة المطلقة والجيب واللوغاريتم وغيرها ، فمثلا

SQRT (2.0 * X + 3.0)

هو تعبير حسابي في الباسكال ، ويقابل التعبير الرياضي

$$\sqrt{2X+3}$$

والصيغة العامة لاستخدام الدوال هي كتابة اسم الدالة وتعبير حسابي بين قوسين كما في المثال السابق.

وفيما يلي قائمة بأسماء بعض الدوال المعرفة التي تستخدم بكثرة في الباسكال

اسم الدالة في الباسكال	الدالة الرياضية المقابلة	معناها
ABS	$\lvert \ldots \rvert$	القيمة المطلقة
SQRT	$\sqrt{}$	الجذر التربيعي
EXP	e	الرفع إلى أس اللوغاريتم الطبيعي
LN	ln	اللوغاريتم الطبيعي (الأساس e)
SIN	sin	جيب الزاوية مقاسة بالتقدير الدائري
COS	cos	جيب تمام الزاوية مقاسة بالتقدير الدائري

TAN(*)	tan	ظل الزاوية مقاسة بالتقدير الدائري
ARCTAN	$\tan^{-1}$	قوس الظل حيث تحسب الزاوية بالتقدير الدائري
SQR	( )2	التربيع
ROUND	التقريب	تقريب الكمية الحقيقية إلى أقرب عدد صحيح
TRUNC	القطع	تحويل الكمية الحقيقية إلى عدد صحيح بحذف أي جزء كسري

عند استعمال أي من هذه الدوال – باستثناء الدالتين الأخيرتين – يمكن أن يكون التعبير المكتوب بين القوسين حقيقيا أو صحيحا ، فإن كان صحيحا فإن كلا من دالتي القيمة المطلقة والتربيع (ABS , SQR) تعطي عددا صحيحا ، أما الدوال الأخرى فتعطي عددا حقيقيا ، وإن كان التعبير حقيقيا فإن جميع هذه الدوال (القيمة المطلقة والتربيع والدوال الأخرى) تعطي أعدادا حقيقية. وواضح أن كلا من دالتي التقريب ROUND والقطع TRUNC تعطي عددا صحيحا ، فمثلا :

ROUND (-7.82) = -8	ROUND (7.82) = 8
TRUNC (-7.82) = -7	TRUNC (7.82) = 7
ROUND (-7.32) = -7	ROUND (7.32) = 7
TRUNC (-7.32) = -7	TRUNC (7.32) = 7
ROUND (-7.5) = -8	ROUND (7.5) = 8
TRUNC (-7.5) = -7	TRUNC (7.5) = 7

سادسا : عبارة الإسناد الحسابية (Arithmetic Assignment Statement) **:**

هي عبارة تستخدم لحساب قيمة تعبير حسابي ثم إعطاء أو إسناد هذه القيمة إلى متغير (فإذا كان لهذا المتغير قيمة سابقة فإنها تمحي وتحل محلها هذه

(*) **الدالة** TAN **غير معرفة في التيربوباسكال.**

القيمة الجديدة ، أي قيمة التعبير الحسابي) .. والصورة العامة للعبارة الحسابية هي :

تعبير حسابي =: اسم متغير

Variable name := arithmetic expression

فمثلا العبارة الحسابية A := B + C تُفقد A قيمتها السابقة لتحل محلها قيمة B + C ، أما قيمة كل B و C فإنها لا تتغير.

وعموما في العبارة الحسابية يكون المتغير والتعبير من النوع نفسه (صحيح أو حقيقي) ، ولكن يسمح للمتغير أن يكون حقيقيا إذا كان التعبير من النوع الصحيح ، وفي هذه الحالة فإن كل العمليات الحسابية في التعبير تجري بالأعداد الصحيحة ثم تحول النتيجة – وهي عدد صحيح – إلى الصورة الحقيقية قبل إعطائها للمتغير الموجود على يسار العبارة الحسابية.

ملاحظات :

يلاحظ أنه عندما نقول : " قيمة متغير ما" فإننا دائما نعني أحدث قيمة أعطيت لهذا المتغير...

كما يلاحظ أن العبارة الحسابية ليست معادلة بالمفهوم الرياضي ، فمثلا لإضافة ٢ إلى قيمة متغير اسمه I لزيادة قيمته بهذا القدر فإننا نكتب عبارة الإسناد الحسابية

I := I + 2

حيث I الموجودة في الطرف الأيمن تعني القيمة القديمة للمتغير I

بينما I الموجودة في الطرف الأيسر تعني القيمة الجديدة للمتغير I

كذلك نؤكد هنا على الصورة العامة للعبارة الحسابية حيث الطرف الأيمن عبارة عن تعبير حسابي والطرف الأيسر عبارة عن اسم متغير واحد فقط ، فمثلا العبارة :

X - Y := Z + T

لا يمكن أن تكون عبارة إسناد حسابية وذلك لأن الطرف الأيسر فيها ليس متغيرا واحدا فقط وإنما هو تعبير حسابي.

مثال ٢–٥ :

ترجم ما يلي إلى لغة الباسكال.

$$T = \ln\left|\sin\frac{x}{3}\right| + \frac{1}{e^{2x}+4}$$

$$y = \left(ax^4 + 10^{-5} + \alpha\cos^2 x\right)^{\frac{3}{4}}$$

$$\phi = \sqrt{|m-n|}$$

الحل :

```
T := LN (ABS(SIN(x/3.0))) + 1.0/(EXP(2.0 * x) + 4.0) ;
Y:= EXP (0.75 * LN (a * SQR(SQR (x)) + 1.0E -5 + alpha *
                                    SQR (COS(x)))) ;
FI = SQRT (ABS (M - N)) ;
```

مثال ٢-٦ :

افرض أن مجموعة العبارات الحسابية التالية سوف تنفذ بنفس الترتيب الذي كتبت به. ما هي قيمة كل من المتغير الصحيح I والمتغير الحقيقي X بعد تنفيذ كل عبارة ، وما هي القيمة النهائية لكل من I , X ؟

```
I := 7 Div 2 + 3 ;
X := 12 * (7 Div 12) + I / 2 + 1 ;
X := 8.0 * X/SQR(X) ;
I := (I-5 MOD 2) * 6 + ROUND (4.6 / X) ;
X : = I DIV 10 ;
X := X + 0.8 ;
I := TRUNC (X + 0.1) ;
```

الحل :

```
I = 3 + 3 = 6
X = 12*0 + 6/2 + 1 = 0 + 3.0 + 1 = 4.0
X = 8.0*4.0 / 16.0 = 32.0 / 16.0 = 2.0
I = (6-1) * 6 + ROUND (4.6/2.0) = 5 * 6 + ROUND (2.3)
  = 5*6 + 2 = 30 + 2 = 32
X = 32 DIV 10 = 3 → 3.0
```

(لأن X متغير حقيقي)

```
X = 3.0 + 0.8 = 3.8
I = TRUNC (3.9) = 3
```

وبالتالي فإن القيمتين النهائيتين هما :

I = 3 , X = 3.8

مثال ٢-٧ :

فيما يلي مجموعة من عبارات الإسناد الحسابية ، والمطلوب اكتشاف أي أخطاء في هذه العبارات.

a) X - Y := LN (T + Q)
b) R2 := SQRT (X + 3E3)
c) B := EXP (SIN (X/2.0)
d) Y := 3.2 X + A*3 / - 4.0
e) 12.5 := X - A
f) -A := 5.0 * D - F

الحل :

i) **الطرف الأيسر تعبير حسابي وليس متغيرا واحدا.**

ii) **لا توجد أخطاء.**

iii) **ناقص قوس على أقصى اليمين.**

iv) **لا يوجد رمز لعملية حسابية بين** 3.2 **والمتغير** X ، **وكذلك الرمزان / و – متجاوران مباشرة.**

v) **الطرف الأيسر ثابت وليس اسم متغير.**

vi) **لا يسمح بوجود إشارة مع اسم المتغير في الطرف الأيسر.**

سابعا : عبارات الإدخال والإخراج (القراءة والطباعة)
Input and Output Statements (Reading and Writing)

(أ) **عبارات الإدخال :**

البيانات أو المعطيات (data) **التي نحتاجها لتنفيذ برنامج ما يمكن أن يحصل عليها الحاسب بإحدى طريقتين :**

الأولى : أن نعطيها له مع البرنامج نفسه في بعض عباراته ، فمثلا قد نكتب بعض العبارات الحسابية لتعريف هذه البيانات حيث يكون الطرف الأيمن في العبارة الحسابية ثابتا مثل :

A := 300.0
N := 1000

TAS := 825.750

أو أن تظهر البيانات ضمن بعض التعابير الحسابية مثل

Z := 825.750/40.0

الثانية : أن تسجل قيم هذه البيانات في موضع ما خارج خطوات البرنامج : على سطور أو بطاقات أو شرائط أو أي وسائل (سجلات) أخرى ، ثم يشتمل البرنامج على عبارات خاصة لقراءة قيم هذه البيانات فيحصل عليها الحاسب أثناء تنفيذ البرنامج.

وفائدة هذه الطريقة الثانية أنه يمكن استخدام البرنامج نفسه بدون عمل أي تعديلات في خطواته لإجراء حسابات والحصول على نتائج خاصة بمجموعة أخرى من البيانات ، حيث أن كل المطلوب في هذه الحالة هو مجرد تغيير سطور البيانات فقط (أو قيم البيانات على الشريط أو الوسيلة الأخرى المستخدمة) أو إضافة سطور البيانات الجديدة ، أما في الطريقة الأولى فيجب تغيير بعض عبارات البرنامج الأصلي نفسه الذي ظهرت فيه البيانات القديمة لنضع مكانها عبارات جديدة تحمل البيانات الجديدة ، ولا شك أن عملية تتبع خطوات البرنامج خطوة خطوة لتغيير البيانات القديمة تستغرق وقتا وجهدا خاصة إذا كان البرنامج طويلا .. وكأمثلة على بعض المسائل العملية التي تتغير فيها البيانات :

- **عند حساب زكاة المال قد يتغير النصاب من عام لآخر ، وكذلك تتغير بيانات المبالغ المدخرة من شخص لآخر.**
- **عند حساب معدلات الطلاب تتغير بيانات الدرجات من طالب لآخر.**
- **في برامج حل المعادلات الجبرية تختلف المعاملات من معادلة لأخرى.**

عبارتا القراءة (READ & READLN Statements) **:**

(i) **عبارة (اقرأ)** READ

(ii) **عبارة (اقرأ ثم انتقل إلى سطر جديد)** READLN **ونختصرها إلى (اقرأ سطريا).**

إذا أردنا أن نقرأ قيم بعض المتغيرات مثل A , ID , TAS فيمكننا أن نكتب في البرنامج العبارة

READ (A , ID , TAS) ;

أو العبارة

READLN (A , ID , TAS) ;

وفي أي من الحالتين فيمكن أن تكون البيانات المقابلة هي :

300.0 3248 12800.250

وهذا يعني أن قيم المتغيرات الثلاثة هي :

A = 300.0 , ID = 3248 , TAS = 12800.250

وفي هذه الحالة فإن أي قيمة سابقة لأي من هذه المتغيرات تمحى وتحل محلها القيمة الجديدة.

ونلاحظ ما يلي :

i) أن تعطى قيم المتغيرات بنفس ترتيب أسمائها التي ظهرت في عبارة القراءة ، وبالطبع تكون البيانات المقابلة للمتغيرات من النوع نفسه على الترتيب ، فمثلا 300.0 من النوع الحقيقي مثل A ، و 3248 من النوع الصحيح مثل ID ، وهكذا.

ii) أن تترك مسافة واحدة على الأقل بين أي قيمتين متتاليتين في مجموعة البيانات.

وأما الفارق بين عبارة (اقرأ) READ وعبارة (اقرأ سطريا) READLN فهو أنه :

* بعد تنفيذ عبارة اقرأ READ يظل مؤشر قراءة البيانات (Cursor) واقفا في موضعه عند آخر سطر بيانات وصل إليه (Current line) ، ليستمر في قراءة البيانات التالية على هذا السطر نفسه عندما يأتي في البرنامج أمر آخر بالقراءة (سواء الأمر (اقرأ) أو الأمر (اقرأ سطريا)).
* بينما في حالة عبارة اقرأ سطريا READLN فبعد تنفيذ هذه العبارة ينتقل مؤشر القراءة إلى بداية السطر التالي للبيانات (New input line) ليقرأ هذه البيانات الموجودة على السطر الجديد عندما يأتي أمر بالقراءة ، وأما البيانات الزائدة المتبقية على السطر الحالي (Current line) فإنها تهمل.

* **وبأسلوب آخر ففي حالة عبارة READLN إذا أعطيت بيانات أكثر من البيانات المطلوبة للمتغيرات في عبارة القراءة (كأن تعطى ٧ قيم عددية في البيانات المقابلة للمتغيرات الثلاثة السابقة (A , ID , TAS) فإن هذه القيم الزائدة (٤ قيم) تهمل.**

وباختصار فإن عبارة READLN تجعل عبارة القراءة التي تأتي بعدها تأمر الحاسب أن يبدأ قراءة القيم من سطر جديد من سطور البيانات ، بينما عبارة READ تجعل عبارة القراءة التالية تأمر الحاسب أن يكمل أخذ القيم من السطر نفسه من حيث انتهى تنفيذ عبارة READ.

مثال ٢-٨ :

نفرض أن لدينا البيانات العددية التالية المكتوبة في سطرين :

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

في كل من الحالات التالية أوجد القيم التي تأخذها المتغيرات

X1 , X2 , … , X6

بعد تنفيذ عبارات القراءة المعطاة :

```
READ (X1 , X2 , X3) ;
READ (X4 , X5 , X6) ;

READLN (X1 ,X2 , X3) ;
READ (X4 , X5 , X6) ;

READLN (X1 ,X2 , X3) ;
READLN (X4 , X5 , X6) ;

READLN (X1 , X2 , X3 , X4 , X5 , X6) ;

READ (X1 ,X2) ;
READ (X3) ;
READ (X4 , X5 , X6);
```

الحل :

	(i)	(ii)	(iii)	(iv)	(v)
X1	2	2	2	2	2
X2	5	5	5	5	5
X3	8	8	8	8	8

X4	10	3	3	10	10
X5	3	7	7	3	3
X6	7	6	6	7	7

ملاحظة :

إذا كتبت عبارة READLN **بدون قائمة مدخلات** (Input list) **هكذا**

READLN ;

فإنها تعنى الانتقال إلى سطر المدخلات التالي.

ولا يسمح بكتابة عبارة القراءة READ **بدون قائمة مدخلات.**

عبارتا الكتابة (الطباعة) (WRITE & WRITELN Statements)

(i) **عبارة (اكتب)** WRITE

(ii) **عبارة (اكتب ثم انتقل إلى سطر جديد)** WRITELN **، ونختصرها إلى (اكتب سطريا).**

(أ) تستخدم أي عبارة من عبارتي الطباعة (i) , (ii) **لطباعة أي نتائج أو بيانات أو قيم متغيرات. فمثلا إذا أردنا أن نكتب قيم بعض المتغيرات مثل :** ID , TAS , Z **، فإننا نستخدم العبارة**

WRITE (ID , TAS , Z) ;

أو العبارة :

WRITELN (ID , TAS , Z) ;

وفي أي من الحالتين فإن الحاسب يكتب أو يطبع قيم هذه المتغيرات فنحصل مثلا على السطر التالي :

3248 1.2800250E + 04 3.2000625E + 02

حيث تظهر قيم المتغيرات بنفس ترتيب أسمائها.

ملاحظتان :

١- المتغيرات التي تطبع قيمها باستخدام عبارة الطباعة لا تتغير قيمها المخزونة داخل الحاسب.

٢- الكميات التي تظهر في عبارة الطباعة يمكن أن تكون على العموم ثوابت عددية أو رمزية أو متغيرات أو تعابير.

مثلا ، العبارة التالية صحيحة :

WRITELN (X , Y , SIN (x) + COS (2.0 * Y) , Z)

وأما الفارق بين عبارة (اكتب) WRITE وعبارة (اكتب سطريا) WRITELIN فهو شبيه بالفارق بين عبارة (اقرأ) وعبارة (اقرأ سطريا)، حيث أنه :

* بعد تنفيذ عبارة (اكتب) يظل مؤشر الكتابة / رأس الطباعة (Printer head) في موضعه عند آخر سطر طباعة وصله (السطر الحالي Current line) ليكمل الطباعة على بقية هذا السطر الحالي عندما يأتي أمر جديد بالطباعة (سواء الأمر (اكتب) أو الأمر (اكتب سطريا)).
* بينما في حالة الأمر (اكتب سطريا) WRITELN فبعد تنفيذ هذا الأمر ينتقل مؤشر الكتابة / رأس الطباعة إلى بداية السطر التالي ليبدأ عليه الطباعة عندما يأتي أمر جديد بالطباعة ، ويترك بقية السطر الحالي (إن كانت هناك بقية) دون أن يطبع عليه شيئا.

وباختصار فإن عبارة WRITELN تجعل عبارة الطباعة التالية تبدأ الطباعة من بداية سطر جديد ، بينما عبارة WRITE تجعل عبارة الطباعة التالية تبدأ الطباعة من بقية السطر الحالي حيث انتهت عبارة WRITE.

(ب) كما يمكن استخدام عبارة الطباعة كذلك في طباعة (رسائل) توضيحية لتوصيف نتائج البرنامج ومخرجاته وجعله أسهل وأوضح في القراءة والتمييز ، وذلك بوضع أي رسالة مطلوب طباعتها بين حاصرتين عاليتين في عبارة الطباعة فتظهر هذه الرسالة في المخرج كما هي مكتوبة تماما في عبارة الطباعة ولكن بدون الحاصرتين. فمثلا الثلاث قيم السابقة (ID , TAS , Z) تظهر في مخرج البرنامج دون أن يظهر معها ما يدل على ما هية هذه القيم وماذا تعنيه ، فإذا أردنا توصيف هذه النتائج وتوضيح معانيها فيمكن أن نعدِّل عبارة الطباعة السابقة لتصبح

WRITELN ('ID =' , ID , 'TAS = ' , TAS , 'Z = ' , Z)

فتظهر القيم الآن بالصورة التالية :

ID = 3248 TAS = 1.2800250E + 04 Z = 3.20006250E + 02

ولذلك نعرف بسهولة عند قراءة هذه النتائج أن القيمة الأولى من جهة اليسار هي (رقم تعريفي) ID والثانية هي (قيمة المدخرات السنوية الكلية) TAS والثالثة هي (قيمة الزكاة) Z .

ويمكن كذلك لعبارة الطباعة أن تستخدم لطباعة رسائل فقط كعناوين توضيحية دون طباعة أي قيم ، فمثلا العبارة

WRITELN ('TABLE OF ZAKAT')

تؤدي إلى طباعة هذا العنوان

TABLE OF ZAKAT

دون أن تطبع معه أي قيم.

ملاحظات :

١- **إذا كتبت عبارة الطباعة** WRITELN **دون قائمة مخرجات** (Output list)، **أي لم يطلب طباعة أي قيم أو أي رسائل توضيحية ، وبالتالي كتبت دون أقواس هكذا :**

WRITELN ;

فإنها تؤدي إلى ترك سطر فارغ عند الطباعة.

٢- **لا يسمح بكتابة عبارة الطباعة** WRITE **دون قائمة مخرجات هكذا:**

WRITE ;

٣- **إذا جاءت أي من عبارتي الإخراج** WRITELN , WRITE **بعد عبارة إدخال ، فإنها تبدأ سطرا جديدا.**

مثال ٢-٩ :

نفرض أن X , Y **عددان صحيحان قيمتاهما** X = 7 , Y = 3 **وأن** t **عدد حقيقي قيمته** 123.456. **ما هي صورة المخرجات بعد تنفيذ مجموعة الأوامر التالية ؟**

(أ)

```
WRITE ('X= ' , X , 'Y= ' , Y)
WRITE ('SUM = ' , X + Y) ;
WRITE ('DIFF = ' , X - Y) ;
```

(ب)

```
WRITELN ('X = ' , X , 'Y = ' , Y) ;
```

```
WRITELN ('SUM = ' , X + Y) ;
WRITE ('DIFF = ' , X - Y) ;
WRITE ('PI = ' , 3.14159) ;
WRITELN ('t = ' , t , 'W = ' , Y + t)
```

(ج)

الحل :

(أ) X = 7 Y = 3 SUM = 10 DIFF = 4

(ب)
```
X = 7    Y = 3
DIFF = 4
```

(ج) PI=3.141590E+0 t=1.2345600E+02 W=1.2645600E+02

عبارات طباعة المخرجات المصاغة Formatted Output Statements

إذا لم يحدد البرنامج بالتفصيل طريقة ظهور النتائج وشكلها فإن برنامج الترجمة (Compiler) يستعمل عددا محددا سابقا من الأعمدة لطباعة أي عدد ، وبالتالي يتحكم هذا البرنامج في الشكل العام للنتائج. وحتى نتحكم نحن في هذا الشكل لتظهر النتائج بصورة معينة فإننا نستخدم ما يسمى بعبارات الطباعة المصاغة مع أي من WRITE أو WRITELN.

فمثلا بدلا من أن نكتب عبارة طباعة غير مصاغة مثل

```
WRITE (ID , TAS , Z)
```

ونترك للحاسب كيفية طباعة قيم هذه المتغيرات حسب الطريقة المحددة له من قبل (وهذه تختلف من نظام لآخر وإن كانت معظم النظم تترك ٨ مواضع للأعداد الصحيحة ، و ١٤ موضعا للأعداد الحقيقية ، منها ٧ مواضع يمين العلامة العشرية ، و٦ مواضع للمتغيرات المنطقية وإذا كان عدد المواضع المسموح بها أكبر من العدد المطلوب للطباعة ، فدائما تطبع قيمة المتغير أقصى اليمين ويترك الفراغ إلى اليسار) فإنه يمكننا كتابة عبارة طباعة مصاغة كالتالية مثلا :

```
WRITE ('ΔID= ' , ID:5, 'ΔTAS = ' , TAS:10:3 , 'ΔZ= ' , Z:8:3)
```

حيث الرمز Δ يشير إلى فراغ واحد.

وهذه العبارة تعطى أمرا لطباعة قيم المتغيرات الثلاثة ID, TAS, Z **بالكيفية التالية ، وهي شرح ما بين القوسين.**

* **معنى** ′ = ∇ID′ **: اطبع هذه الرموز بين الحاصرتين كما هي ، كعنوان لما سيطبع بعدها.**

معنى ID : 5

الرقم 5 يشير إلى أن قيمة ID ستطبع في خمسة أعمدة (أي خمسة مواضع أو خمس مسافات) متتالية ، وبحيث تطبع القيمة في أقصى يمين هذه المسافات الخمس إذا كانت القيمة تشغل أقل من خمس مسافات كأن تكون ثلاثة أرقام فقط (مثلا 245) وتترك باقي المسافات الخمس على يسار القيمة فارغة (فتترك مثلا مسافتان فارغتان يسار القيمة 245) ، وإذا كانت القيمة مسبوقة بإشارة ، فإن الإشارة تشغل موضعا من الخمسة مواضع. وإذا كانت القيمة تشغل أكثر من خمس مسافات (مثل 245123) فإنها تطبع كاملة.

* **معنى** ′ = ∇TAS′ **: تطبع هذه الرموز بين الحاصرتين كما هي كعنوان لما سيطبع بعدها.**

معنى TAS : 10 : 3

أن قيمة TAS ستطبع في عشرة أعمدة (عشرة مواضع أو مسافات) متتالية ، منها ثلاث مسافات إلى أقصى اليمين مخصصة للكسر العشري مقربا لثلاثة أرقام تكتب على يمين العلامة العشرية (decimal point) والتي تشغل الموضع الرابع من اليمين من هذه المواضيع العشرة ، وبذلك تبقى ستة مواضع لطباعة الجزء الصحيح من القيمة وكذلك الإشارة إن كانت هناك إشارة.

* **معنى** ′ = Δ Z′ **: طباعة الرموز بين الحاصرتين كما هي.**
* **معنى** Z : 8 : 3 **: طباعة القيمة الحقيقية التالية – وهي قيمة المتغير Z – في ثمان مسافات منها ثلاث مسافات إلى أقصى اليمين مخصصة للكسر العشري مقربا لثلاثة أرقام ، ثم المسافة الرابعة من اليمين مخصصة للعلامة**

العشرية ثم الأربع مسافات الباقية على اليسار مخصصة للجزء الصحيح من القيمة وللإشارة إن وجدت.

فإذا فرضنا أن قيم المتغيرات هي :

ID = 245 , TAS = 2802.4 , Z = 70.06

فإن تنفيذ عبارة الطباعة السابقة مع عبارة الصيغة يؤدي إلى ظهور النتائج بالصورة التالية :

ΔID = ΔΔ245 ΔTAS = ΔΔ2803.400 ΔZ = ΔΔ70.060

ملاحظتان :

* إذا كانت قيمة متغير مثل Z هي 70.0657 مثلا وطلبنا طباعتها حسب المجال 8:3: أي بعد تقريب الكسر العشري لثلاثة أرقام ، فإن القيمة تظهر هكذا :

Z = ΔΔ70.066

* يجب على المبرمج أن يراعي إعطاء مجال (أي عدد من المسافات) يكفي لطباعة القيمة المطلوبة (بما فيها من علامة عشرية وكسر وجزء صحيح وإشارة) وإلا فإن الحاسب سيأخذ مجالا أكبر لطباعة القيمة ، فمثلا المجال 5:3: لا يكفي لطباعة قيمة Z لأن هذا يعني ترك مسافة واحدة فقط للجزء الصحيح بينما الجزء الصحيح (70) يشغل مسافتين.

مثال ٢-١٠ : (على الطباعة المصاغة للأعداد الصحيحة)

تحتوي ذاكرة الحاسب على القيم التالية للمتغيرات الصحيحة المذكورة :

ND = 17, NM = 9, NY = 2, M = 314, NM = 14, K = 950, KK = 70 , KP = 70.

أولا : صف نتيجة تنفيذ مجموعة العبارات الآتية حول بعض المعلومات عن غزوة بدر الكبرى.

```
WRITELN  ('ΔΔΔΔΔBATEL OF BADR') ;
WRITELN  ;
WRITELN  ('DATE : ', ND : 3, '/', NM:3, '/', NY:3);
WRITELN  ('NO. OF MUSLIMS = ', M : 6);
WRITELN  ('NO. OF MARTYRS = ', MM : 6);
WRITELN  ('NO. OF INFIDELS = ', K : 6);
WRITELN  ('NO. OF INFIDELS KILLED = ', KK : 6);
```

WRITELN ('NO. OF INFIDELS TAKEN PRISONERS = ', KP: 6);

ثانيا : ما هي قيمة كل من R_1 , R_2 بعد تنفيذ العبارتين الحسابيتين التاليتين بلغة الباسكال ؟

R1 := ROUND (K/M) ;
R2 := ROUND (KK/MM) ;

وماذا يمثل كل من هذين العددين ؟

الحل : أولا :

ΔΔΔΔΔBATEL OF BADR
DATE : Δ17 / ΔΔ9 / ΔΔ2
NO. OF MUSLIMS = ΔΔΔ314
NO. OF MARTYRS = ΔΔΔΔ14
NO. OF INFIDELS = ΔΔΔ950
NO. OF INFIDELS KILLED = ΔΔΔΔ70
NO. OF INFIDELS TAKEN PRISONERS = ΔΔΔΔ70

ثانيا :

R1 := ROUND (950/314) = 3
R2 := ROUND (70/14) = 5

العدد R1 **يمثل نسبة عدد الكفار إلى عدد المسلمين (ثلاثة أضعاف) {تقريبا} ،**

والعدد R2 **يمثل نسبة قتلى الكفار إلى شهداء المسلمين (خمسة أضعاف).**

* * *

ذكرنا سابقا طريقة لصياغة المخرجات حقيقية القيمة حيث تظهر المخرجات بالصورة العشرية ، وهناك طريقة أخرى نذكرها فيما يلي تظهر فيها المخرجات بالصورة الأسية.

فمثلا بدلا من استخدام عبارة الطباعة المصاغة

WRITE (TAS : 10 : 3)

يمكننا استخدام العبارة :

WRITE (TAS : 10)

والتي تعني حجز ١٠ مواضع لطباعة قيمة TAS **بالصورة الأسية ، على أن يظل الموضع الذي في أقصى اليسار شاغرا (أي فراغا) إذا كانت قيمة** TAS

موجبة ، ويطبع فيه إشارة (–) إذا كانت قيمة TAS سالبة. فإذا فرضنا مثلا أن TAS = 2802.400 ، فإن عبارة الطباعة الأخيرة تؤدي إلى ظهور قيمة TAS بالصورة التالية :

2.802E+03

بينما العبارة WRITE (TAS : 12) تؤدي إلى طباعة القيمة

2.80240 E + 03

وإذا استخدمنا مجالا ضيقا أي صغيرا غير كاف لطباعة القيمة كاملة ، مثل

WRITE (TAS : 4)

فإن الحاسب سيستخدم مجالا أكبر لطباعة القيمة بالصورة الأسية بحيث تشتمل على رقم عشري واحد على الأقل ، ويزيد عدد الأرقام العشرية المطبوعة كلما سمح بذلك المجال المعطى في عبارة الطباعة المصاغة. فمثلا يطبع الحاسب القيمة

٢,٨ E + 03

إذا استخدمنا أيا من العبارات التالية :

WRITE (TAS:1),WRITE (TAS:2),…,WRITE (TAS:8)

بينما إذا استخدمنا العبارة WRITE (TAS : 10)

فكما سبق ذكره تطبع القيمة :

2.802 E + 03

وهكذا.

ويمكننا أن نلخص في الجدول التالي المجالات المختلفة واستخداماتها:

وصف المجال	معناه واستخدامه
I : w	لطباعة العدد الصحيح I في عدد w من المواضع (w تشمل موضع الإشارة) ويطبع العدد في أقصى اليمين. ومن الممكن أن يكون I متغيرا أو ثابتا أو تعبيرا صحيحا.
R:w:d	لطباعة البيانات الحقيقية R (real) في صورة عشرية في أقصى اليمين في عدد w من المواضع ، حيث تكون قيمة البيانات مقربة إلى d رقم عشري كسري (w تشمل d وموضعا للعلامة العشرية وموضعا للإشارة السالبة إن وجدت ، بالإضافة

إلى عدد مواضع أرقام الجزء الصحيح من قيمة البيانات).
ومن الممكن أن يكون R متغيرا أو ثابتا أو تعبيرا حقيقيا وكل من w,d تعبير صحيح موجب.

R:w لطباعة البيانات الحقيقية R في صورة أسية في عدد w من المواضع ، بحيث تبدأ القيمة المطبوعة بفراغ أقصى اليسار إن كانت القيمة موجبة وبإشارة (–) إن كانت القيمة سالبة ، ولا يوجد إلا رقم عشري واحد قبل العلامة العشرية أي على يسارها. وكما سبق :
من الممكن أن يكون R متغيرا أو ثابتا أو تعبيرا حقيقيا ، و w تعبير صحيح موجب.

وفي أي من حالتي البيانات الصحيحة او الحقيقية ، إذا لم يخصص مجال يكفي لطباعة البيانات ، فإن الحاسب يأخذ مجالا أكبر لطباعتها ، بحيث تشتمل القيمة المطبوعة على رقم عشري واحد على الأقل (ويزيد عدد الأرقام العشرية المطبوعة كلما سمح بذلك المجال المعطى في عبارة الطباعة المصاغة).

مثال ٢–١١ :

نفرض أن

J = -2468
T = 123.45678
TN = -T

أوجد نتيجة تنفيذ عبارات الطباعة المصاغة التالية ، أي اكتب المخرجات.

(أ)
```
WRITE (J : W)
```
حيث

i) W = 1 ii) W = 4 iii) W = 5 iv) W = 8

(ب)
```
WRITE (T : W)
```
حيث

i) W = 1 ii) W = 5 iii) W = 8
iv) W = 12 v) W = 15

(ج) أعد حل (ب) بالنسبة للعبارة
```
WRITE (TN : W)
```

(د) WRITE (T : W : d)

حيث : i) W = 1 , d = 2 ii) W = 6 , d = 2

iii) W = 7 , d = 2 iv) W = 1 , d = 4

v) W = 8 , d = 4 vi) W = 12 , d = 2

vii) W = 13 , d = 6

(هـ) أعد حل (د) بالنسبة للعبارة

WRITE (TN : W : d)

الحل : (أ)

i) , ii) , iii) -2468 iv) ΔΔΔ-2468

(ب)	(ج)
i) , ii) , iii) Δ1.2E + 02	-1.2E + 02
iv) Δ1.234567E + 02	-1.23457E + 02
v) Δ1.23456780E + 02	-1.23456780E + 02

(د)	(هـ)
i) , ii) 123.46	-123.46
iii) Δ123.46	-123.46
iv) , v) 123.4568	-123.4568
vi) ΔΔΔΔΔ123.46	ΔΔΔΔΔ-123.46
vii) ΔΔΔ123.456780	ΔΔ-123.456780

ثامنا : عبارة التعليق (Comment Statement)

وهي عبارة غير تنفيذية تستخدم لتوضيح وشرح بعض خطوات البرنامج أو وظيفة البرنامج وبعض المعلومات عنه ، ولذلك فمن الممكن أن يوجد أكثر من عبارة تعليق واحدة في بداية البرنامج وفي ثناياه .. والصيغة العامة لعبارة التعليق أن يكتب التعليق المطلوب – وهو سلسلة متتابعة من الرموز characters **والفراغات – بين قوسين مموجين**(Curly braces)**:** {.....} **، أو بين نجمتين داخل قوسين عاديين :** (*.....*) **، كأن نكتب مثلا :**

{This program is for computing Zakat - U1 - Mal}

أو

(*Computing the area of a triangle *)

قواعد عامة لكتابة البرنامج :

قبل أن نبدأ كتابة بعض البرامج البسيطة نذكر فيما يلي :

الصيغة العامة لأي برنامج بلغة الباسكال:

يتكون البرنامج من الأجزاء الرئيسية التالية والمذكورة حسب ترتيب ورودها في البرنامج.

(أ) عنوان البرنامـج (Program Heading)

ويكتب في عبارة واحدة تبدأ بكلمة PROGRAM **يليها الاسم التعريفي للبرنامج** identifier **ثم قائمة وسطائه** Parameter list **بين قوسين** (.....). **أي أن الصيغة العامة لعنوان البرنامج هي :**

PROGRAM Identifier (Parameter list) ;

وقائمة الوسطاء هي عادة الكلمتان المحجوزتان INPUT , OUTPUT **أو إحداهما ، وهما يمثلان وسيلتى الإدخال والإخراج ، أو اسمي ملف الإدخال وملف الإخراج.**

ومن أمثلة عناوين البرامج :

PROGRAM Zakat (INPUT , OUTPUT) ;
PROGRAM Area (INPUT , OUTPUT) ;
PROGRAM FunctionTable (OUTPUT) ;

(ب) جزء الإعلانات والتعريفات
(Declaration part and Definitions)

يتكون هذا الجزء عموما من أربعة فروع لتعريف مكونات البرنامج وأنواعها من ثوابت ومتغيرات ودوال وغيرها.

ويجوز ألا يظهر فرع أو أكثر . والفروع الأربعة هي :

(١) جزء تعريف الثوابت (Constant Definition Part)

وصيغته العامة :

CONST .. = .. ; .. =.. ; ..... ;

(٢) جزء تعريف الأنواع (Type Definition Part)

(وهذا الجزء سنتكلم عنه في الفصل الخامس والملحق (أ)).

وصيغته العامة :

TYPE .. = .. ; .. = .. ; ..... ;

(٣) جزء الإعلان عن المتغيرات (Variable Declaration Part)

وصيغته العامة :

VAR .. , .. , ... , .. : .. ;

.....

.. , .. , ... : .. ;

(٤) جزء الإعلان عن الدوال والإجراءات

(FUNCTION and PROCEDURE declaration part)

(وسنتناول الدوال والإجراءات في الفصل السادس بإذن الله)

(ج) جسم أو عبارات البرنامج

Program body (Pascal Statements)

ويتكون هذا الجزء من العبارات التنفيذية (executable statements) **أي العبارات المطلوب تنفيذها ، وتكتب سلسلة هذه العبارات بين الكلمة المحجوزة** BEGIN **في البداية والكلمة المحجوزة** END **في النهاية ، مع وضع نقطة بعد كلمة** END **، أي في نهاية البرنامج. أي أن الصيغة العامة لهذا الجزء هي :**

BEGIN

..... ;

..... ;

..... ;

.....

END.

* **ويلاحظ دائما وجود فاصلة منقوطة " ; " (semicolon) بين أي عبارتي باسكال ، وهي ليست جزءا من عبارة باسكال ولكنها مجرد فاصلة بين أي عبارتين متتاليتين. وعادة لا نضع هذه الفاصلة / الفارزة المنقوطة بعد كلمة** BEGIN **أو قبل كلمة** END.
* **ويمكن أن تكتب أكثر من عبارة واحدة على السطر نفسه ، طالما أن هناك فاصلة منقوطة بين أي عبارتين متتاليتين.**
* **ويمكن أن تشتمل سلسلة عبارات البرنامج على عبارة مركبة وهي عبارة مكونة من مجموعة متتالية من العبارات (مع وجود فاصلة منقوطة بين كل**

عبارتين متتاليتين) بين كلمة BEGIN في البداية ، وكلمة END في النهاية. وهكذا يمكن أن يشتمل البرنامج على أكثر من كلمة END واحـدة ، ولكن لا توضع النقطة إلا بعد آخر كلمة END .

ملاحظات عامة :

يجب ترك فراغ واحد على الأقل بين أي كلمة محجوزة واسم تعريفي ، مثل : VAR Zakat

فلا يجوز أن نكتب : VARZakat دون ترك أي فراغ. وعموما يجب ترك فراغ واحد على الأقل بين أي اثنين من الكلمات المحجوزة والأسماء التعريفية والأرقام.

كما لا يجوز ترك فراغ حيث يجب ألا يكون له محل ، فلا يجوز مثلا ترك فراغ بين الرمزين = و : في مؤثر الإسناد =: كما لا يجوز كتابة الاسم التعريفي FunctionTable هكذا Function Table.

* * *

والآن يمكننا كتابة بعض البرامج البسيطة التي توضح المفاهيم السابقة لأساسيات لغة الباسكال ، والتي درسناها في هذا الفصل.

مثال ٢-١٢ :

اكتب برنامجا لقراءة قيم ثلاثة متغيرات X1 , X2 , X3 وحساب :

(أ) المتوسط الحسابي $AM = \frac{X_1+X_2+X_3}{3}$

(ب) المتوسط الهندسي $GM = \sqrt[3]{X_1 . X_2 . X_3}$

الحل :

```
PROGRAM ArithGeoMeans (INPUT , OUTPUT) ;
{Computing the arithmetic and geometric means of Three values
                                X1 , X2 , X3}
VAR X1 , X2 , X3, AM , GM : REAL ;
BEGIN
        READLN (X1 , X2 , X3) ;
WRITELN ('The three values are : ' , X1, X2, X3) ;
        AM := (X1+X2+X3)/3.0 ;
GM := EXP (1.0/3.0*LN (X1*X2*X3)) ;
```

```
WRITELN ('Arithmetic Mean = ' , AM) ;
WRITELN ('Geometric Mean = ' , GM)
END.
```

مثال ٢-١٣ :

مساحة مثلث أطوال أضلاعه A , B , C **تعطى بالعلاقة**

$$Area = \sqrt{S(S-A)(S-B)(S-C)}$$

حيث $$S = \frac{A+B+C}{2}$$

اكتب برنامجا يقرأ أطوال أضلاع مثلث A , B , C **ويحسب مساحته.**

الحل :

```
PROGRAM AreaTriangle (INPUT , OUTPUT) ;
VAR A, B, C, S, Area : REAL ;
BEGIN
    READLN (A,B,C) ;
WRITELN (' The Sides are : ' , A, B, C) ;
    S := (A + B + C) /2.0 ;
AREA := SQRT (S * (S-A) * (S-B) * (S-C)) ;
    WRITELN ('The area is : ' , AREA)
END.
```

* * *

البرنامج التبادلي والبرنامج المتكامل (Interactive Program and Batch Program)

مثل هذا البرنامج في مثال ٢-١٣ – وكذلك البرنامج المعطى في حال المثال السابق – يطلق عليه (برنامج متكامل) أو (برنامج دفعة واحدة) (batch program) **، حيث تعطى كافة البيانات المطلوبة دفعة واحدة (في ملف إدخال) مع البرنامج ، وأثناء تنفيذ خطوات البرنامج كلما جاء أمر بقراءة بيانات معينة أخذها الحاسب على الترتيب من مجموعة البيانات المعطاة ابتداء (في هذا الملف).**

وأما النوع الآخر من البرامج فيطلق عليه (برنامج تبادلي) (interactive program) **وهو برنامج لا تعطى معه مجموعة البيانات كاملة ابتداء ، وإنما يأخذ هذه البيانات على الشاشة مباشرة أولا بأول على دفعات كلما احتاج إليها ، أي**

كلما طلب مجموعة معينة من البيانات أعطيناها له على الشاشة. أي أننا نتبادل المعلومات مع البرنامج أثناء تنفيذه حيث أنه كلما احتاج بيانات معينة أعطيناها له ، ويعطينا هو النتائج. ولذلك لا نعطيه كافة البيانات قبل تنفيذه.

والبرامج التبادلية تعد شائعة الاستعمال هذه الأيام ، وتستخدم مع الحاسب الشخصي (personal computer) ، وكذلك مع الحاسوب متقاسَم الوقت (Large time-shared computer) ، ومثل هذا الحاسوب يستخدم عادة في الجامعات للأغراض التعليمية حيث يتم توصيل عدد كبير من المستخدمين (users) – عن طريق وحدات اتصال (terminals) – بحاسوب مركزي واحد ، ويتقاسم جميع المستخدمين الإمكانات والوسائل المركزية المتاحة.

ويعد البرنامج المتكامل أحد الخيارات (options) الممكنة المتاحة على معظم الحواسيب الشخصية أو الحواسيب متقاسَمة الوقت.

وبالنسبة للبرنامج التبادلي فإنه يشتمل على عبارات طباعة تحتوي على رسائل (messages) تعرض على الشاشة لتحث (prompt) أو تلقن الشخص مستخدم الجهاز أن يدخل البيانات المطلوبة في ذلك الموضع ، مثل إحدى العبارات التالية :

```
WRITELN ('Enter values of A,B,C') ;
WRITE      ('Input the Total Annual Savings in dinars') ;
WRITELN ('Enter the length of R in centimeter') ;
WRITELN ('Enter X1, X2, X3') ;
```

ويلي عبارة الحث هذه – التي تشتمل على رسالة تلقينية (prompting message) لإدخال البيانات – عبارة قراءة لقراءة هذه البيانات. وقد تشتمل الرسالة التلقينية ، على الصيغة المتوقعة للبيانات المطلوبة ، كأن يكون الطول مثلا بالسنتيمتر أو بالكيلومتر ، أو يكون المبلغ بالدينار أو الدرهم ، أو تحديد عدد الأرقام يمين الفاصلة العشرية ، ... وهكذا. وترجع أهمية طباعة الرسالة التلقينية قبل عبارة القراءة إلى أنه بدون هذه الرسالة قد لا يدرك الشخص أن البرنامج قد أوقف التنفيذ أو لا يدري ما هي البيانات التي عليه إدخالها.

ومن أمثلة عبارة القراءة بعد عبارة طباعة الرسالة التلقينية العبارات التالية (انظر عبارات الحث السابقة) :

```
READLN (A,B,C) ;
READLN (TAS) ;
READLN (R) ;
READLN (X1,X2,X3) ;
```

مثال ٢-١٤ :

أعد حل مسألة المثال السابق (٢-١٣) بكتابة برنامج تبادلي (interactive pgm.) **بدلا من البرنامج المتكامل** (batch pgm.)

الحل :

```
PROGRAM AreaTriangle (INPUT , OUTPUT) ;
VAR  A, B, C, S, AREA : REAL ;
BEGIN
    WRITELN ('Enter values of A,B,C') ;
    READLN (A,B,C) ;
    S := (A+B+C)/2.0 ;
AREA := SQRT (S*(S-A)*(S-B)*(S-C)) ;
    WRITELN ('The sides are : ' , A,B,C) ;
    WRITELN (' The area is : ' , AREA)
END.
```

لاحظ وجود عبارة طباعة الرسالة التلقينية قبل عبارة قراءة البيانات.

مثال ٢-١٥ :

توفى رجل عن زوجة وعدد من الأبناء الذكور NS **والإناث** ND **(بحيث أن** NS>0 , ND>0**) وترك ميراثا قدره** A **دينارا.**

اكتب برنامجا لتوزيع هذا الميراث على ورثته تبعا للقاعدتين التاليتين:

(أ) فإن كان لكم ولد فلهن الثمن مما تركتم.

(ب) يوصيكم الله في أولادكم للذكر مثل حظ الأنثيين.

ونفرض أننا سنرمز لنصيب الزوجة بالرمز W **ونصب الابن بالرمز** S **ونصيب البنت بالرمز** D **.**

ابدأ البرنامج بقراءة قيم البيانات A , NS , ND **واستخدم عبارة تعليق في مطلع البرنامج لبيان وظيفته ، وعبارات تعليق توضيحية في ثناياه.**

الحل :

```
PROGRAM Inheritance (INPUT , OUTPUT) ;
{program for distributing the estate of a deceased
on wife, sons and daughters}
VAR
A {the estate} : REAL ;
W {Wife's share} : REAL ;
S {Son's share} : REAL ;
D {Daughter's share} : REAL ;
NS {Number of sons} : INTEGER ;
ND {Number of daughters : INTEGER ;
BEGIN
{Input data}
WRITELN ('Enter the value of the estate') ;
READLN (A) ;
WRITELN('Enter No. of sons and No. of daughters');
READLN (NS , ND) ;
WRITELN ;
{Compute the different shares}
W := A/8.0 ;
D := (7.0/8.0*A)/(2.0*NS +ND) ;
S := 2.0*D ;
{Printing the read data}
WRITELN ('The Inheritance Problem') ;
WRITELN ;
WRITELN ('Value of the estate = ' , A:12:3, 'dinars');
WRITELN ('No. of sons = ' , NS);
WRITELN ('No. of daughters = ', ND);
WRITELN ;
{printing the results}
WRITELN ('Wife''s share = ', W:12:3, 'dinars');
WRITELN ('Daughter''s share = ' , D:12:3, 'dinars');
WRITELN ('Son''s share = ', S:12:3, 'dinars')
END {Inheritance}.
```

ملاحظات :

عند تنفيذ هذا البرنامج تظهر العبارة التالية على الشاشة مطالبة بإعطاء قيمة الميراث :

Enter the value of the estate

فندخل قيمة الميراث ، ولتكن مثلا 1284.400 (دينارا).

ثم تظهر العبارة التالية :

Enter No. of sons and No. of daughters

فندخل عددين صحيحين مثل 8 , 6 بينهما فراغ واحد على الأقل (أي أن عدد الأولاد الذكور 6 وعدد البنات 8) هكذا :

8 6

ثم تظهر المخرجات هكذا :

The Inheritance Problem

Value of the estate = ΔΔΔ12840.400 dinars
No. of sons = 6
No. of daughters = 8
Wife's share = ΔΔΔΔ1605.050 dinars
Son's share = ΔΔΔΔ1123.535 dinars
Daughter's share = ΔΔΔΔΔ561.768 dinars

أنواع العبارات في لغة الباسكال

يمكن تقسيم العبارات في لغة الباسكال إلى قسمين رئيسيين :

(أ) العبارات البسيطة (Simple Statements)

(ب) العبارات المبنية (Structured Statements)

وهناك أنواع مختلفة من العبارات في كل قسم من هذين القسمين.

(أ) العبارات البسيطة : وهي العبارات التالية :

(١) العبارة الخالية أو الفارغة (the empty statement) ، وهي عبارة عن فراغ (أو أكثر) ، وتستخدم حينما لا يُطلب عمل أي شئ. وسنرى أمثلة لها فيما بعد.

(٢) عبارة الإسناد (assignment statement).

(٣) عبارة الإجراء (procedure statement).

وسنتناولها بالتفصيل في الفصل السادس إن شاء الله ، ولكن من أمثلة عبارات الإجراء التي مرت بنا عبارات : READ , READLN , WRITE , WRITELN.

(ب) العبارات المبنية : وهي العبارات التالية :

(١) العبارة المركبة (Compound statement)

هي مجموعة عبارات محصورة بين الكلمة المحجوزة BEGIN في البداية والكلمة المحجوزة END في النهاية ، ومع وجود فاصلة منقوطة بين كل عبارتين متتاليتين.

(٢) العبارات الشرطية (Conditional statements)

وهي عبارات (إذا) (IF statements) ، وعبارات الحالة أو الانتقاء (Case statements).

وسندرس العبارات الشرطية في الفصل القادم بإذن الله.

(٣) العبارات التكريرية (repetitive statements)

وهي عبارات FOR ، وعبارات REPEAT ، وعبارات WHILE ، وسندرسها بإذن الله في الفصل الرابع.

(٤) عبارة WITH ونحتاجها عند دراسة أنواع بيانات السجلات (record data types).[*]

[*] انظر مثلا كتاب البرمجة المتقدمة بلغة الباسكال ، للمؤلف بالاشتراك مع د. حمزة رشوان ، دار القلم ، ١٩٩٤.

تمرينات رقم ٢

أولا : تدريبات

٢-١ حول كلا من الثوابت الحقيقية التالية من الصيغة الأسية إلى الصيغة العشرية :

3.0E4(a
5.08E0(b
-524.5E - 02(c
0.0293E - 4(i

٢-٢ حدد كلا مما يلي ما إذا كان ثابتا صحيحا (ث.ص) أو ثابتا حقيقيا (ث.ح) أو ثابتا رمزيا (ث.ر) أو متغيرا (م) أو ليس أيا من هذه (ل).

a)	Y2	b)	-45,000
c)	2.431	d)	125
e)	I25.4	f)	2Y
g)	$2.54	h)	294E-6
i)	'Enter Count'	j)	246.0
k)	'K3'	l)	COMPUTE X
m)	VALUE	n)	'LARGE'
p)	BIG	q)	GAMMA
r)	.8E10	s)	294.E-6
t)	0.	u)	8 3
v)	'There is no god but Allah'		

٢-٣ ترجم التعابير الرياضية التالية إلى تعابير حسابية بلغة الباسكال. استخدم العدد 3.14159 **لقيمة** π.

i) $(A+1)(A+B+6)$

ii) $\left(A-\frac{B}{C}\right)-3$

iii) $A^{i}-B^{c}+\sqrt{A+B}$

iv) $I^{j}+64$

v) $\frac{a+b}{c+\left[d/(e+f)\right]}$

vi) $\frac{(a/b)-1}{g\left[(g/d)-1\right]}$

vii) $\pi r^2 h$

viii) $\left(\frac{3V}{4\pi}\right)^{\frac{1}{3}}$

ix) $\frac{\pi h}{3}\left(R^2+Rr+r^2\right)$

x) $\frac{x+\pi}{12}$

xi) $\frac{-\cos^4 x}{x} + \frac{\sin^3 x \cos^2 x}{5}$

xii) $\ln|\sec x + \tan x| - \frac{1}{3\left(\sqrt{x^2 - a^2}\right)}$

xiii) $\frac{bc}{12}\left[6x^2\left(1-\frac{x}{a}\right) + b^2\left(1-\frac{x}{a}\right)^3\right]$

٢-٤ اكتب عبارات إسناد حسابية بلغة الباسكال تقابل العلاقات الرياضية التالية:

$$A = \left[\frac{a+b}{c+d}\right]^{k-1}$$

$$B = \left(\frac{a.b}{c+\frac{d}{3}+8}\right)^{t-1}$$

$$C = \frac{\alpha^3.10^{22}}{5!}\left|e^{\sin\phi}\right|$$

$$D = \frac{e^x.\sin x}{|y|+\cos Z}$$

$$E = \sqrt{|\cos(a - n.b)| + \ln|m - n|}$$

$$F = C \sin(\phi - t)$$

$$G = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$$

$$H = \sin X + \sin^2 X + \sin^3 X + ... + \sin^6 X$$

$$P = \left(\frac{i.k}{3L}\right)^{2m} + j^4$$

$$Q = a_0 + a_1X + a_2X^2 + ... + a_5X^5$$

$$R = \frac{\left(\frac{a}{b}\right)^{-p}}{1-\frac{c}{d}} + \sin\left|\frac{x-y}{x+y}\right|$$

٢-٥ ترجم ما يلى إلى عبارة بلغة الباسكال :

N = I ÷ J حيث كل من I و J عدد صحيح ، و N هي خارج القسمة I ÷ J مقربا إلى أقرب عدد صحيح.

٢-٦ اكتشف أي أخطاء موجودة في العبارات الحسابية التالية المكتوبة بلغة الباسكال.

V - 3.96 := X * 1.67(a
K12.3 := I * J(b
Z2 := A * -B + C * 4(c
R := 16.9 X + AB(i

٢-٧ فيما يلي مجموعة من العبارات بلغة الباسكال ، والمطلوب بيان أي هذه العبارات صحيحة وأيها غير صحيحة ، وبالنسبة للعبارات غير الصحيحة وضح الخطأ وحاول تصحيحه إن أمكن.

R := SQRT(YZ + 4E3 + 12.5)(One
S := 3E 4.0 + EXP(T + S) + 25Y / - 7.0(Two
SQRT (49.0) := 7.0(Three
I + I := I(Four
J := J + 1(Five
X := SQRT (SQRT(X))(Six
X1 := JI + K - 4(Seven
PI := 3.14159(Eight
R := X3 * (RL + RW)(Nine
J := SIN (A + COS(X)) / 4.0(Ten
WRITE (LN(Y * SIN(X)))(Eleven
READ (A,k,M,T).(a

٢-٨ اكتب بعض العبارات بلغة الباسكال كقطعة من برنامج تقوم بتبديل قيمتي المتغيرين X و Y.

٢-٩ نفرض أن A , B , C , X متغيرات حقيقية وأن I , J , M متغيرات صحيحة وأن

A = 4.0 , B = 3.2 , C = 1.5 , I = 3 , J = 2

ما هي قيمة كل من X , M بعد تنفيذ كل من العبارات الحسابية التالية :

a)	X := A + B * C	b)	X := A* I - C
c)	X := A * C / I * B	d)	X := SQR(SQR(A))
e)	X := A / B * J	f)	X := (A - B) * I
g)	X := 2.0 * (1.0E1 / 4.0)	h)	X := 19 / 4 + 5 / 3
i)	X := 4.0 * (3 / 2)	j)	M := 2 * I + I MOD J

k) M := I * J * (I DIV J) l) M := J * I - J

٢-١٠ نفرض أن I = 1 , A = 2.5 , B = 3.2 **. ما هي قيمة كل من** I , A , B , F **عند نهاية تنفيذ مجموعة العبارات التالية (حيث** I **متغير صحيح وكل من** A , B , F **متغير حقيقي) ؟**

```
I := I + 1 ;
C := B - A ;
D := C * C + 1.0E - 2 ;
I := 3 ;
F := (1.0 + D) * (I - 1) ;
A := A + B + F ;
```

٢-١١ ما هي القيم النهائية للمتغيرات الحقيقية I , X , Y , Z **بعد تنفيذ القطعة التالية من أحد البرامج :**

```
I := 3 * 4 / 5 + 3 / 5 ;
X := 5 / 2 * (I + 1) ;
Y := (X - 38.5) * I ;
Z := Y / 3.0 * I ;
X := 5.0 * Z + Y ;
```

٢-١٢ اكتب كلا من الأعداد التالية بالصيغة الأسية في لغة الباسكال :

534.964(One
66(Two
.00794(Three
469.55468(Four
-1465.386(a

٢-١٣ ماذا يُطبع بعد تنفيذ العبارتين التاليتين ؟ إذا كانت هناك أي فراغات فيجب إظهارها في مواضعها بالضبط.

```
num := -3.8625 ;
writeln (num : 9 : 2) ;
```

٢-١٤ ما هي القيمة التي تخزن في result **بعد تنفيذ مجموعة العبارات التالية ؟**

```
program one ;
                var
result : integer ;    x , y , b ,
                    begin
                x := 3 ;
                y := -2 ;
                b := 10 ;
                z := 2.5 ;
```

```
result := x * y - b/trunc (z) - y ;
```

٢-١٥ حول كلا من العلاقات الرياضية التالية إلى عبارة إسناد بلغة الباسكال. استخدم الدوال القياسية كلما أمكن ذلك. افرض أن جميع المتغيرات حقيقية. إذا ظهرت عدة رموز متتالية (تبدأ بحرف) - دون أي مؤثرات بينها - فهي تشير إلى متغير واحد وليس عدة متغيرات مضروبة في بعضها البعض ، فمثلا id متغير واحد وليس حاصل ضرب متغيرين i , d ، وكذلك eltrns متغير واحد وليس حاصل ضرب ٦ متغيرات ، وهكذا.

(a $eltrns = \frac{t^3 * b * e^{(-eg/2k)}}{k * \sqrt{id+is}}$

(b $t = \frac{\sin^3 w * \cos^2 v}{5p} + e^{-\left[\frac{v * w}{2p}\right]}$

(c $w = x * \arctan\left(x + \sqrt{a}\right) - \frac{a * \ln(x^2 + a^2)}{2}$

(d $den = \frac{f(1.015+0.85f)^2 * vis}{t}$

(e $freq = \frac{(1+\sin^3 a) * e^{(a(2+b))}}{b * \cos(a/2)}$

(f $j = \log_e h + 0.622p\left[\frac{1}{p-x} - \frac{1}{p-y}\right]$

(i $f = 0.35 \times 10^{-4} * \frac{x * \sqrt{2g} * (y2-y1)^2}{\log_e(y2+y1)}$

٢-١٦ ما هي أنواع المتغيرات التي تظهر في العبارات التالية حتى تكون هذه العبارات صحيحة بلغة الباسكال ؟

```
a := b <= c DIV 8 ;
e := b + c / (4 mod d) ;
f := (x = 4.5) or (x = 12.0) ;
g := Trunc(x) + SQR (Round (x)) ;
h := (k = 'i') or (k = 'j') ;
```

٢-١٧ اكتب عبارة اسناد واحدة - في أبسط صورة ممكنة - مكافئة في نتيجتها لقطعة البرنامج التالية المكونة من عبارتي اسناد متتاليتين :

```
a := 3 * (d + e) ;
```

```
a := a - (2 * e) ;
```

٢–١٨ اكتب المعادلة الجبرية المقابلة لكل من العبارات التالية المكتوبة بلغة الباسكال :

(i) `f := p * q / r * s / t`

(ii) `g := x * EXP (3 * LN(y))/(a  + 2 * b)`

(iii) `h := EXP (4 * LN(x) / (1 * 2 * 3 * 4)`

(iv) `p := r * SQR (Theta) / (a * SQR (SQR (b)))`

(هـ) `q := SQRT (SQR (x) + SQR (y))`

(و) `t := SQR (COS (x + 2 * y))`

٢–١٩ اكتشف أي أخطاء موجودة في عبارات الإدخال أو الإخراج التالية المكتوبة بلغة الباسكال :

(i) `READ a , b , c , d ;`

(ii) `WRITE    (x  y  z) ;`

(iii) `READLN (a1 , a2 , a3) ;`

(iv) `WRITELN (f := a * x + b) ;`

(هـ) `WRITELN (x , sqrt (x)) ;`

٢–٢٠ نفرض أن قيمة المتغير الحقيقي x هي 743.28 . اكتب نتيجة تنفيذ كل من عبارات الطباعة التالية :

(i) `WRITELN ('x = ' , x)`

(ii) `WRITELN ('x = ' , x : 9)`

(iii) `WRITELN ('x = ' , x : 8 : 2)`

(iv) `WRITELN ('x = ' , x : 5 : 1)`

(هـ) `WRITELN ('x = ' , x : 5)`

٢–٢١ نفرض أن لدينا عبارتي الإسناد :

```
sum := 6 ;          n := 15 ;
```

ما هي مخرجات عبارات الطباعة التالية على الترتيب ؟ وإن كان هناك خطأ تركيبي (syntax error) في أي عبارة ، فوضح الخطأ فيها بدلا من بيان مخرجاتها.

```
writeln ('n = ' , n , 'total = ' , sum + 2) ;(One
writeln ('result = ' , n mod sum) ;(Two
writeln ('R = ' , 'sum' div 3) ;(Three
```

writeln ('sum' , n , 10 div (sum mod 4)) ;(Four
writeln ('sum + n') ;(Five
writeln ('n = ' , n , sum = , 'sum') ;(a

٢-٢٢ i) صحح الأخطاء التركيبية (syntax errors) في تركيب عبارات البرنامج التالي بلغة الباسكال ، وأعد كتابته صحيحا بنفس شكله العام.

ii) ما وظيفة كل عبارة من عبارات البرنامج المصحح ؟

iii) إذا فرضنا أن عدد ساعات العمل (Hours) يساوي ١٠٠ ساعة وأن معدل أجر الساعة الواحدة (Rate) يساوي ٣ دنانير ، فماذا تكون مخرجات البرنامج ؟

```
Program Wages (InPut, OutPut)
Var Hours ; Rate ; G : Real ;
Const T : 25.0 ;
Begin
Write Ln ('Enter hours Worked) ;
ReadLn (Hours) ;
WriteLn ('Enter hourly Rate') ReadLn (Rate) ;
Hours * Rate := Gross ;
Net : Gross - Tax ;
WriteLn ('Gross Pay is KD , G) ;
WriteLn ('Net pay is KD' , Net)
END.
```

٢-٢٣ فيما يلي طائفة من القوانين العلمية ، والمطلوب ترجمة كل قانون منها إلى عبارة مناسبة (بسيطة أو مركبة) بلغة الباسكال :

أ) قانون زكاة الركاز والمعدن : "وفي الركاز الخُمس"

[الركاز هو المدفون من كنوز الجاهلية ، وما يخرج من الأرض مما له قيمة كالذهب والفضة والنفط والحديد ... ونحو ذلك].

ب) قانون كولوم في الفيزياء : $force = q_1.q_2/d^2$

ج) قانون الغاز المثالي : $P = 0.0299\ (T + 460) / V$

د) قانون الجاذبية لنيوتن : $f = g.\frac{m.n}{r^2}$

٢-٢٤ نفرض أن القيم المخزونة في المتغيرات الصحيحة y1 , y2 , y3 هي :

y1 = 1948 y2 = 1967 , y3 = 1982 ,

اكتب قطعة برنامج تستخدم المتغيرات y1 , y2 , y3 بحيث تجعل الحاسب يطبع السطور التالية :

Palestine has been occupied in 1948.
Golan Heights and West Bank have been occupied in 1967.
South of Lebanon has been occupied in 1982.
Jihad is the only way to liberate the occupied lands.

ثانيا : برامج :

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 π = 3.1415927 في قسم اعلانات البرنامج.**

٢-٢٧ اكتب برنامجا بلغة الباسكال يقوم بالتالي :

(أ) قراءة قيمتي DAYS و WAGE حيث :

DAYS تعني عدد أيام الجهاد التي قضاها الجندي مرابطا في سبيل الله تعالى ونفرض أن $DAYS \geq 40$

WAGE تعني الأجر اليومي بالدينار الذي يتقاضاه الجندي.

(ب) حساب قيمة كل من الأجر العادي للجندي REGPAY ، وأجهره الإضافي OVERTM ، وأجره الكليTOTPAY ، حسب القوانين التالية :

$$PEGPAY = 40 \times WAGE$$

$$OVERTM = (DAYS - 40)(WAGE \times 1\frac{1}{2})$$

$$TOTPAY = REGPAY + OVERTM$$

(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 Z , N .

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 $\pi = 3.1415927$)

(ب) حجم كرة نصف قطرها R.

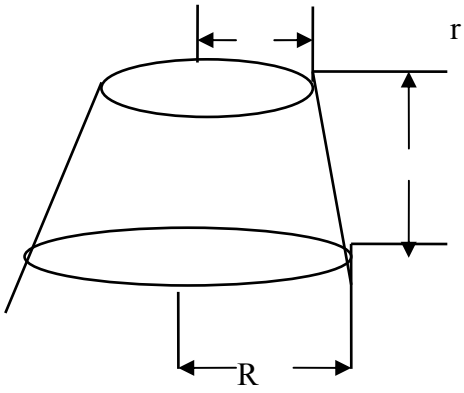
(حجم الكرة = $\frac{4}{3}\pi R^3$)

٢-٣١ تتحرك عربة بعجلة ثابتة a لمدة t ثانية. اكتب برنامجا لقراءة قيمة كل من t , a 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 بلغة الباسكال لقراءة قيمة كل من r , R , h وحساب الحجم V بفرض أن $\pi = 3.1415927$



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 : $Celsius = 5/9 \times (Fahrenheit - 32)$

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 :

Sum : عدد صحيح (مجموع x , y)

Difference : عدد صحيح (الفارق بين x , y)

Product : عدد صحيح (حاصل ضرب x , y)

Quotient : عدد حقيقي (حاصل قسمة x على y)

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)

٢-٣٦ 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 : Real      {Input-Angle (radians) of elevation}
Distance : Real   {Input - Distance (ft) to target}
Velocity : Real   {Input - Projectile Velocity (ft/sec)}
```

Program output مخرجات البرنامج

```
Time : Real       {output - Time (sec) of flight}
Height : Real     {output - height at impact}
```

Relevant Formulas العلاقات / القوانين المستخدمة

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

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 a acceleration) ، 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

عبارات التحكم
Control Statements

في البرامج البسيطة كتلك التي تعرضنا لها في الفصل السابق في الأمثلة والتمرينات يتم تنفيذ التعليمات أي الأوامر أو العبارات التنفيذية واحدة تلو الأخرى وفق الترتيب أو التسلسل نفسه الذي كتبت به ، إلا أن هناك مسائل كالتي ناقشناها في الفصل الأول (عن الخوارزميات وخرائط سير العمليات) تتطلب تغييرا في تسلسل تعليمات البرنامج أثناء التنفيذ (أي ترتيب تنفيذ التعليمات قد يختلف عن ترتيب كتابة التعليمات) حيث أن ترتيب التنفيذ قد يعتمد أحيانا على شروط معينة ، فبناءا على نتيجة إحدى العمليات الحسابية مثلا نود الانتقال لتنفيذ عبارة سابقة أو عبارة لاحقة تأتي بعد عدة عبارات ، وأحيانا نود الرجوع إلى بداية البرنامج لنعيد تنفيذ تعليماته مرة أخرى ولكن باستخدام بيانات جديدة ، وهكذا.

وفي هذا الفصل سندرس بإذن الله تعالى بعض العبارات الأساسية التي تتحكم في تسلسل خطوات تنفيذ تعليمات البرنامج أي في الانتقال من خطوة لأخرى أثناء التنفيذ ، وفي غياب عبارات التحكم فإن تنفيذ تعليمات البرنامج يتم وفق ترتيب كتابتها.

ويمكن تقسيم عبارات الانتقال عموما إلى قسمين :

(أ) عبارة الانتقال غير المشروط GOTO (unconditional transfer) :

[انظر ملحق (جـ)]. وعموما ينصح بعدم استخدام هذه العبارة حيث لا يسهل معها متابعة تسلسل خطوات البرنامج.

(ب) عبارات الانتقال المشروط (conditional transfer) :

(١) عبارات " إذا " (IF statements)

(٢) عبارة " الحالة " (Case statement) أو بنية الشرط الانتقائي.

توجد عدة أنواع من عبارات الانتقال إلى مواضع مختلفة في البرنامج عند تحقق شروط معينة ، ولدراسة هذه الأنواع ندرس أولا طريقة التعبير عن الشروط المنطقية

(logical conditions) **أو التعابير المنطقية** (logical expressions) ، **وأبسط صورة من صور التعابير المنطقية ما يسمى بالتعابير العلاقية.**

التعبير العلاقي (Relational Expression)

يتكون التعبير العلاقي من تعبيرين حسابيين يصل بينهما واحد من المؤثرات العلاقية الستة التالية :

المؤثر العلاقي Relational Operator	معناه الرياضي
<	<
<=	≤
=	=
<>	≠
>	>
>=	≥

مثال ٣ – ١ :

فيما يلي مجموعة من التعابير العلاقية بلغة الباسكال وما يكافئها من التعابير الرياضية٠

	التعبير العلاقي في الباسكال	التعبير الرياضي المكافئ
i.	x < y	$x < y$
ii.	(a+5.0) f >=SQR(b)	$(a+5) \geq b^2$
iii.	i - 3 <> j	$i - 3 \neq j$

التعبير المنطقي (Logical expression)

هو بصورته البسيطة تعبير أو شرط إما صادق (أي متحقق) أو خاطئ (أي غير متحقق) ، فإن كان صادقا قلنا إن قيمته TRUE **وإن كان خاطئا قلنا إن قيمته** FALSE.

والتعبيرات العلاقية السابقة كلها تعبيرات منطقية لأنها أما صادقة وإما خاطئة فمثلا إذا فرضنا أن

x = 2.0 , y = 3.0 , a = 4.0 , b = 3.0 , i = 5 , j = 2

فإن التعبير العلاقي (i) **صادق** TRUE **لأن** 2 < 3

والتعبير العلاقي (ii) **صادق** TRUE **لأن** 9 ≥ 9

والتعبير العلاقي (iii) **خاطئ** FALSE **لأن** 2 = 2

وسندرس فيما بعد إن شاء الله التعابير المنطقية المركبة ، ونحب أن نورد هنا الملاحظة التالية :

ملاحظة :

يجب أن يكون المرء حريصا عند استخدام المؤثرين = و <> لمقارنة عددين حقيقيين ، وذلك لأنه بسبب الأخطاء الداخلية في الحسابات – كأخطاء التقريب لدقة معينة داخل الحاسب – قد لا يتساوى داخل الحاسب عددان حقيقيان المفروض أنهما متساويان من الناحية النظرية. ولذلك فبدلا من كتابة الشرط A = B لاختبار صحة تساوى العددين الحقيقيين A,B يمكن أن نختبر صحة وقوع الفرق بينهما داخل حدود معينة من الدقة أو التجاوز كأن نكتب

ABS (A-B) < 0.000001

وتعني $|A - B| < 10^{-6}$ وذلك لتعويض أي خطأ محتمل بسبب التقريب.

وقبل إعطاء التعريف الشامل للتعبير المنطقي (العام أو المركب) نشير أولا إلى الثوابت والمتغيرات والمؤثرات المنطقية.

الثوابت المنطقية (Logical or Boolean Constants)

ليس هناك إلا ثابتان منطقيان مُعرَّفان هما TRUE (أي صحيح) و FALSE (أي خاطئ).

المتغيرات المنطقية (Logical or Boolean Variables)

المتغير المنطقي هو متغير لا يأخذ إلا إحدى القيمتين TRUE, FALSE. ويجب الإعلان عن المتغيرات المنطقية في جزء البرنامج الخاص بالإعلان عن المتغيرات ، وذلك بأن نذكر أسماء هذه المتغيرات ويليها الاسم التعريفي BOOLEAN ، كأن نكتب مثلا :

```
Flag, Switch , kk :  BOOLEAN ;
```

وقد تعطي هذه المتغيرات قيما للتخزين في الذاكرة من خلال عبارات مثل :

```
Flag := TRUE ;
Switch := FALSE ;
```

ويلاحظ أنه لا يجوز إعطاء قيمة للمتغير المنطقي من خلال عبارة قراءة (READ أو READLN) كما هو الحال مع المتغيرات الحسابية الصحيحة أو الحقيقية ، بل لابد من إعطاء القيم (TRUE/ FALSE) للمتغيرات المنطقية من خلال عبارات الإسناد فقط.

ومن أمثلة عبارات الإسناد المنطقية :

```
Flag   :=  x < 8.5 ;
Switch :=  SQR(b) > 4.0 * a * c ;
kk     :=   TRUE ;
```

وفي حالة الطباعة فعند كتابة اسم متغير منطقي ضمن قائمة مخرجات عبارة الطباعة (WRITE أو WRITELN) فإن القيمة المناظرة التي تطبع هي إما TRUE أو FALSE٠

المؤثرات المنطقية والتعابير المنطقية (المركبة)

Logical (or Boolean) Operators &
(compound) Logical (or Boolean) Expressions

ذكرنا سابقا أن التعبير المنطقي البسيط (simple logical expression) هو تعبير علاقي (relational expression) مثل x + 4.5 >= y

ويمكننا تركيب تعابير منطقية أكثر تعقيدا باستخدام المؤثرات المنطقية الثلاثة التالية :

AND
OR
NOT

فإذا أُعطينا تعبيرين منطقيين أمكننا تكوين تعبير منطقي مركب منهما باستخدام أي من المؤثرين AND و OR والقيمة المنطقية للتعبير المركب تعتمد على القيمة المنطقية لكل من التعبيرين كما يلي :

- التعبير المركب أو الشرط المركب باستخدام المؤثر AND يكون صادقا TRUE أي متحققا في حالة واحدة فقط وهي إذا كان كل من التعبيرين صادقا أي كان كل من الشرطين متحققا.
- التعبير المركب أو الشرط المركب باستخدام المؤثر OR يكون صادقا أي متحققا في حالة ما إذا كان أحد الشرطين متحققا أو كان كلاهما متحققين.

وأما المؤثر المنطقي NOT **فلا يؤثر إلا على تعبير منطقي واحد فقط ويغير قيمته المنطقية إلى عكسها (من صادق** TRUE **إلى كاذب أي خاطئ** FALSE **أو العكس) ، ولذلك فهو يسمى مؤثر النفي** (negation operator).

أمثلة للتعابير المنطقية المركبة :

(x >= y + z) AND (j< 9)
(i < j) OR (n > 25)
NOT (x < 20.0)

فالتعبير الأول يكون صادقا إذا كان $x \geq y + z$ **وأيضا** $j < 9$

والتعبير الثاني يكون صادقا إذا كان $i < j$ **أو** $n > 25$.

والتعبير الثالث يكون صادقا إذا كان $x < 20$.

أي أن التعبير المركب الثالث يكافئ التعبير البسيط

x >= 20.0

ولاحظ أننا استخدمنا الأقواس في كتابة التعابير المنطقية المركبة السابقة ، وعادة نستخدم الأقواس حين نحتاج إليها لتجنب أي أخطاء أو التباس كما سنرى بإذن الله بعد قليل حين ندرس قاعدة الأولويات بصورتها العامة.

بعد الأمثلة السابقة من التعابير المنطقية (البسيطة والمركبة) يمكننا إعطاء التعريف الشامل للتعبير المنطقي.

تعريف : التعبير المنطقي (المركب)

هو متتالية من الثوابت المنطقية والمتغيرات المنطقية والتعابير المنطقية البسيطة (أي التعابير العلاقية) والتي يفصل بينها جميعا مؤثرات منطقية.

ملاحظة :

لا يجوز ظهور مؤثرين منطقيين متجاورين مباشرة مثل :

…….. AND OR ……

إلا إذا كان المؤثـر الأيمـن هـو مؤثر النفـي NOT **فيسمح بذلك ، مثـــل**

L1 AND NOT L2 **حيث كل من** L1 , L2 **متغير منطقي.**

قاعدة الأولوية بالنسبة للمؤثرات المنطقية

(Precedence of Logical Operators)

- في حالة وجود (عدة) أقواس داخلية وخارجية ((( ))) فتعطى الأولوية لما هو داخل الأقواس الداخلية أولا ثم الخارجية التي تليها ثم التي تليها وهكذا..
- يعطى المؤثر NOT الأولوية الأولى بين المؤثرات المنطقية

  يليه المؤثر AND

  يليه المؤثر OR

مثال ٣ – ٢ :

إذا فرضنا أن A = 5.0 و B = 2.0 فأوجد القيمة المنطقية للتعبير المنطقي المركب التالي :

(B < A) AND (A + B = 8.0) OR (NOT(B = 4.0))

الحـــل :

- نبدأ بما هو داخل القوسين الداخليين B = 4.0

  فنجد أنه غير صحيح F (أي FALSE).
- ثم داخل القوسين الخارجيين نجد المؤثر NOT الذي يعكس القيمة إلى T (أي TRUE).
- ثم تعطى الأولوية للمؤثر AND قبل المؤثر OR.

  فعلى يسار AND نجد 5.0 < 2.0 وهذا صحيح T.

  وعلى يمينها نجد 8.0 = 7.0 وهذا غير صحيح F.

  أي أن أحد طرفي AND يعطي T والآخر يعطي F ولذلك فنتيجة تطبيق هذا المؤثر هيF .
- يبقى المؤثر OR وعلى يساره F وعلى يمينه T ولذلك فهو يعطي T أي أن النتيجة هي أن القيمة المنطقية للتعبير المنطقي المركب السابق هي TRUE.

القاعدة العامة للأولويات (General Precedence Rule)

عند إيجاد قيمة تعبير يشتمل على مؤثرات حسابية وعلاقية ومنطقية (وأقواس) أو على أي من هذه المؤثرات ، فإن الأولوية تعطي تبعا للترتيب التنازلي في السطور التالية ، حيث يعطي المؤثر الموجود في سطر علوي أولوية قبل المؤثر الموجود في سطر سفلي ، وأما المؤثرات ذوات الأولوية المتساوية – أي الموجودة على السطر نفسه – فإن المؤثر الذي يظهر على اليسار في التعبير المطلوب إيجاد قيمته يعطى أولوية قبل المؤثر الذي يظهر على اليمين في هذا التعبير.

أولا :	**الأقواس**				(	)
ثانيا :	NOT					
ثالثا :	*	/	DIV	MOD	AND	
رابعا :	+	–	OR			
خامسا :	<	<=	=	<>	>	>=

مثال ٣ – ٣ :

نفرض أن x , y , z , t **متغيرات حقيقية ، وأن** L1 , L2 , L3 **متغيرات منطقية.**

فيما يلي مجموعة من التعابير ، والمطلوب – بالنسبة لكل تعبير – بيان ما إذا كان التعبير صحيحا أم خاطئا ، مع بيان سبب الخطأ إن كان خاطئا ، وإعطاء تعبير مكافئ له يوضح تسلسل إجراء العمليات الواردة في التعبير وكيفية إيجاد قيمته إن كان صحيحا.

(أ) NOT L1 OR L2 AND L3

(ب) x > y OR z + t <= 1.5

(ج) x * y > z - 3.4

(د) L1 AND L2 OR NOT L3 AND NOT L1

(هـ) (x >= y * t) AND NOT L1 OR L2

(و) 12.4 < t AND L1

الحل :

(أ) التعبير منطقي صحيح

(NOT L1) OR (L2 AND L3)

(ب) التعبير منطقي خاطئ ، لأنه حسب قاعدة الأولويات يفسر هكذا :

x > (y OR z) + t <= 1.5

والتعبير y OR z **ليس له معنى لأن كلا من** y , z **متغير حقيقي وليس منطقيا ، بينما المؤثر** OR **لا يؤثر إلا على كميات منطقية.**

أما إذا كان التعبير الأصلي معطى بالصورة التالية :

(x > y) OR (z + t <= 1.5)

فانه يكون صحيحا ، وهنا المؤثر OR **يربط بين تعبيرين منطقيين بسيطين ، وفي هذه الحالة يكون التعبير مكافئا للتعبير التالي :**

(x > y) OR ((z + t) <= 1.5)

(جـ) التعبير منطقي صحيح ويكافئ

(x * y) > (z - 3.4)

(د) التعبير منطقي صحيح ويكافئ

(L1 AND L2) OR ((NOT L3) AND (NOT L1))

(هـ) التعبير منطقي صحيح ويكافئ

((x >= (y *t)) AND (NOT L1)) OR L2

(و) التعبير منطقي خاطئ لأنه حسب قاعدة الأولويات يكافئ التعبير

12.4 < (t AND L1)

وما بين القوسين تعبير خاطئ لأن t **متغير حقيقي وليس منطقيا. أما إن كان التعبير الأصلي بالصورة التالية :**

(12.54 < t) AND L1

فإنه يكون صحيحا.

ملاحظة :

يعرف الثابتان المنطقيان القياسيان TRUE , FALSE

بحيث أن[*] FALSE < TRUE

لذلك فإن أيا من المؤثرات العلاقية

< <= = <> > >=

يمكن أن يؤثر على كميات منطقية ويعطي نتيجة منطقية.

الدالة القياسية المنطقية ODD (دالة الفردية) : تعرف هذه الدالة كما يلي :

ODD (x) : تعطي القيمة المنطقية TRUE إذا كانت قيمة التعبير الصحيح x (integer) فردية بينما تعطي FALSE ماعدا ذلك.

* * *

والآن بعد أن درسنا الشروط المنطقية أو التعابير المنطقية ندرس عبارات الانتقال المشروط ، ونبدأ بعبارات " إذا ".

(١) **عبارات " إذا " (المنطقية)** (Logical) IF Statements

هناك ثلاث بنيات مختلفة لعبارة " إذا " المنطقية ، وهذه البنيات هي :

(أ) بنيـة IF .. THEN .. (Structure)

(ب) بنيـة IF .. THEN .. ELSE ..

(جـ) بنيـة IF المتداخلة (Nested IF Structure)

وفيما يلي بيان لهذه البنيات وشرح استخدامها.

(أ) بنيـة IF .. THEN ..

الصيغة العامة لهذه البنية هي :

IF e THEN T

---

[*] تستعمل لغات البرمجة جداول للرموز القياسية بحيث يعطي الجدول رقما خاصا لكل رمز (أي لكل رقم أو حرف أو رمز خاص) ، ويقال إن رمزا أصغر من رمز آخر إذا كان الرقم الخاص بالرمز الأول في الجدول أصغر من الرقم الخاص بهذا الرقم الآخر ، ومن هذه الجداول مثلا جدول ASCII وجدول EBSDIC .

حيث

e : **تعبير منطقي (شرط منطقي).**

T : **عبارة (بسيطة أو مبنية) ، وتسمى " عبارة جواب الشرط " أو اختصارا " العبارة الشرطية "** (conditional statement).

ومعنى هذه البنية : إذا كان التعبير e صادقا أي كان الشرط متحققا فإن العبارة T يتم تنفيذها. وإذا لم يتحقق الشرط فإن T لا تنفذ.

وفي أي من الحالتين فإن العبارة التالية التي تنفذ هي العبارة التي تلي بنية IF..THEN ..

مثال ٣ – ٤ :

فيما يلي بعض الأمثلة لعبارات IF .. THEN ..

```
IF TAS > A THEN Zakat := TAS / 40.0 ;
IF N = 1000 THEN average := sum/N ;
IF Switch THEN WRITELN ('NO Solution') ;
IF SQR (b) > 4.0 * a * c THEN
      BEGIN
            Discr :=  SQR (b) - 4.0 * a * c ;
            SD    :=  SQRT (Discr) ;
            X1    := (-b - SD) / (2.0 * a);
            X2    := (-b + SD) / (2.0 * a)
      END ;
IF (N = 30) AND (diff > EPS ) THEN
            WRITELN (' NO convergence in 30 iterations ') ;
```

مثال ٣ – ٥ :

ترجم إلى لغة الباسكال كلا من العبارات التالية :

i) **إذا كان** $x \geq y + z$ **وأيضا** $j < 9$ **فزد قيمة** N **بواحد.**

ii) **إذا كان** $3 < x < 5$ **فاحسب قيمة** y **من العلاقة** $y = x^4$

iii) **إذا كان** A **و** B **لهما الإشارة نفسها أو كان** $A \leq 50$ **فاعكس إشارة** A.

iv) **إذا كان** $I \neq K$ **أو** $C + D < F^3$ **فاحسب** t **باستخدام القانون** $t = e^{x+y}$

الحل :

```
a)IF (x >= y + z ) AND ( j < 9 ) THEN N := N+1 ;
b)    IF (3.0 < x )  AND ( x < 5.0 ) THEN
```

```
y := SQR(SQR(x)) ;
```

ملاحظة : من الخطأ ترجمة الشرط $3 < x < 5$ **إلى :**

$$3.0 < x < 5.0$$

ويلاحظ أن الشرط $3 < x < 5$ **هو في الحقيقة شرط مركب من شرطين** $3 < x$ & $x < 5$ **فيجب ترجمتها إلى الباسكال بالتفصيل كما هو مبين مع وجود المؤثر** AND **ودون تداخل بينهما كما هو مشار إليه في الحل الخطأ.**

```
iii)  IF (A * B > 0.0)  OR  (A<= 50.0) RHEN A := - A ;
iv)  IF (K < > I)  OR  (C + D < F * SQR (F))  THEN
     t := EXP (x + y ) ;
```

مثال ٣ – ٦

اكتب برنامجا لقراءة قيمتين X , Y **والمقارنة بينهما ، ثم وضع القيمة الكبرى في** X **والصغرى في** Y.

الحل :

من الواضح أنه إذا كانت Y **هي القيمة الكبرى فالمطلوب من البرنامج تبديل القيمتين (راجع المسألة ٢ – ٨ لكيفية عمل هذا التبديل) ، وما عدا ذلك فلا يكون هناك تبديل وتبقى كل قيمة محتفظة باسمها.**

```
PROGRAM Interchange (INPUT ,  OUTPUT) ;
VAR  X, Y, T : REAL ;
BEGIN
      WRITE (′ Enter X , Y ′) ;
      READLN (X,Y) ;
      IF Y > X THEN
            BEGIN { interchange the 2 values }
               T := X ;
               X := Y ;
               Y := T
            END ;
      WRITELN (′X = ′ , X , ′Y = ′ , Y) ;
END .
```

(ب) بنيـــة IF .. THEN .. ELSE ..

الصيغة العامة لهذه البنية هي :

```
IF e  THEN  T  ELSE  F
```

حيث e **تعبير منطقي (شرط منطقي)**

وكل من T , F عبارة (بسيطة أو مركبة)

وتسمى T عبارة جواب الشرط أو العبارة الشرطية(conditional statement) ، بينما تسمى F العبارة البديلة (alternate statement).

ومعنى هذه البنية : إذا كان الشرط e متحققا فإن العبارة T يتم تنفيذها وتهمل العبارة F ، وأما إذا لم يكن الشرط متحققا فيتم تنفيذ العبارة F وتهمل العبارة T.

وفي أي من الحالتين فإن العبارة التالية في التنفيذ هي العبارة التي تلي بنية IF.

ومن أمثلة هذه البنية لعبارة IF :

```
IF TAS >= Nisab THEN      Zakat  := TAS / 40 .0
ELSE       Zakat  :=  0.0 ;
```

(لاحظ عدم وجود فاصلة منقوطة قبل كلمة ELSE).

ويمكن كتابة هذا المثال في الصورة التوضيحية التالية :

```
IF TAS >= Nisab THEN
      Zakat := TAS/40.0
ELSE
      Zakat := 0.0
{End IF} ;
```

أي أن الصيغة العامة لهذه الصورة التوضيحية هي :

```
IF e   THEN
   T
ELSE
   F
{ END IF};
```

ويلاحظ أيضا أنه لا توجد فاصلة منقوطة قبل كلمة ELSE. وفي حالة ما إذا كانت F هي العبارة الفارغة – أي إذا كنا لا نريد عمل شيء في حالة عدم تحقق الشرط e – فإننا نحذف كلمة ELSE ، وبذلك تتحول هذه البنية إلى البنية السابقة .. IF .. THEN

وفي حالة ما إذا كانت كل من T , F عبارة مركبة (مبنية) ، فإن الصيغة العامة السابقة تأخذ الصورة التالية :

```
IF   e   THEN
       BEGIN
            .............
                  ⋮
                  ⋮
```

```
            .............
        END
ESLE
        BEGIN
            .............
                ⋮
                ⋮
            .............
        END
{ End IF } ;
```

ولاحظ مرة أخرى عدم وجود فاصلة منقوطة قبل كلمة ELSE.

مثال ٣ – ٧ :

اكتب برنامجا لقراءة قيمتين X , Y **ثم حساب قيمة** Z **والمعرفة بأنها تساوي خارج القسمة** $Z = \frac{Y}{X}$

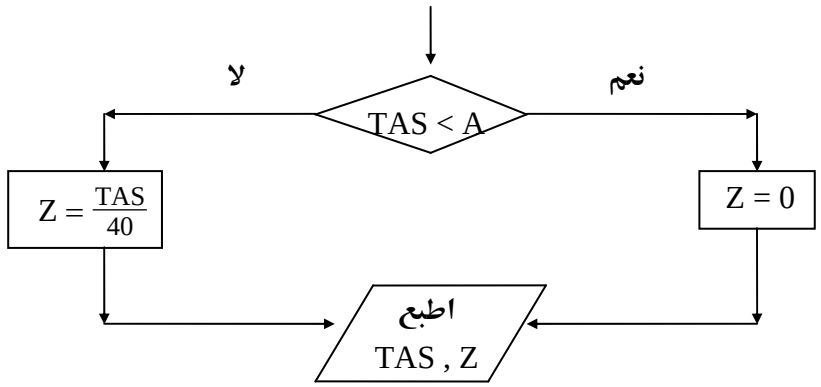
مع ملاحظة أن يعطي البرنامج رسالة تفيد أن Z **غير معرفة في حالة ما إذا وجد أن قيمة** X **تساوي صفرا.**

الحـــل :

```
PROGRAM Division (INPUT , OUTPUT ) ;
VAR X , Y , Z : REAL ,
BEGIN
        WRITE (′ Enter X , Y′) ;
        READLN (X , Y ) ;
        IF X < > 0.0 THEN
                BEGIN
                        Z := Y/ X ;
                        WRITELN (′X= ′, X, ′Y = ′ , Y , ′Z = ′ , Z)
                END
        ELSE
                WRITELN (′X = 0.0 , Z is undefined′)
        { End IF }
END.
```

مثال ٣ – ٨ :

ترجم إلى لغة الباسكال خريطة سير العمليات الجزئية الموضحة بالشكل ، والتي يمكن صياغة مدلولها بالعبارة التالية (راجع أيضا مثال ١ – ٢) :



إذا كانت المدخرات السنوية الكلية TAS أكبر من أو تساوي النصاب A فزكاتها تقدر بربع عشرها ، أما إذا لم تبلغ النصاب فلا زكاة عليها. وفي أي من الحالتين اطبع قيمة المدخرات TAS والزكاة Z.

الحل الأول :

رأينا سابقا (في مثال ١ – ٢) أنه يمكن صياغة معلومات المسألة في الصيغة الرياضية المختصرة التالية :

$$Z = \begin{cases} \dfrac{TAS}{40} & \text{if} \quad TAS \geq A \\ 0 & \text{if} \quad TAS < A \end{cases}$$

وباستخدام النوع الثاني (ب) من بنيات IF يمكن ترجمة هذه الصيغة مباشرة وحل المثال هكذا :

```
IF TAS >= A THEN
        Z := TAS / 40.0
ELSE
            Z := 0.0
{ End IF } ;
WRITELN (' TAS = ' , TAS , ' Zakat = ' , Z) ;
```

ويمكن كتابة هذا الحل في الصورة التالية :

```
IF  TAS >= A  THEN  Z := TAS / 40.0  ELSE  Z := 0.0 ;
WRITELN (′ TAS=′, TAS , ′ Zakat = ′ , Z) ;
```

الحل الثاني :

يمكن إدماج عبارة الطباعة وعبارة الإسناد التي قد تشتمل على إيجاد قيمة تعبير حسابي في عبارة واحدة هي عبارة طباعة قد تشتمل على التعبير الحسابي ، هكذا :

```
IF TAS >= A THEN
      WRITELN ('TAS = ′ , TAS , 'Zakat = ′ , TAS / 40.0)
ELSE
      WRITRLN ('TAS = ′ , TAS , ′ Zakat = 0.0′) ;
{END IF} ;
```

ويلاحظ أن الفرق بين هذا الحل والحل السابق أن الحل السابق يقوم أولا بتخزين قيمة الزكاة في ذاكرة الحاسب – عن طريق عبارة الإسناد – قبل طباعة هذه القيمة ، بينما هذا الحل الحالي يطبع قيمة الزكاة مباشرة دون تخزينها.

الحل الثالث :

باستخدام النوع الأول (أ) من بنيات IF ، يمكننا حل المثال هكذا :

```
IF  TAS >= A  THEN  Z := TAS / 40.0 ;
IF  TAS < A  THEN  Z := 0.0 ;
WRITELN  ('TAS = ′ , TAS , 'Zakat = ′ , Z ) ;
```

ملاحظة :

نحب أن نشير هنا إلى خطأ عام يقع فيه بعض الطلاب عند استخدام هذا النوع من عبارات IF :

فبعد كتابة عبارة IF في الحل السابق ، قد يقول الطالب إنه إذا لم يتحقق الشرط TAS >= A فإن العبارة الحسابية Z := TAS / 40.0 الموجودة بعد THEN سوف لا تنفذ ، وسينتقل الحاسب لينفذ العبارة التالية ولذلك فيكتبها الطالب Z := 0.0 مباشرة بدون ذكر الشرط TAS < A معها ، فيصبح حل الطالب هكذا :

```
IF TAS >= A THEN Z := TAS / 40.0 ;
Z := 0.0 ;
WRITELN ('TAS = ', TAS , 'Zakat = ′ , Z)
```

وهذا الحل خطأ لأنه إذا تحقق الشرط TAS >= A فإن العبارة Z := TAS / 40.0 سوف تنفذ ، ثم ينتقل الحاسب إلى العبارة التالية Z := 0.0 لينفذها وبالتالي يحصل على

قيمة جديدة للمتغير نفسه Z فيمحو القيمة القديمة الصحيحة (TAS / 40.0) ويحل محلها القيمة الجديدة الخاطئة (صفرا).

وبأسلوب آخر فإنه بناء على هذا الحل ستكون قيمة الزكاة المحسوبة مساوية صفرا دائما سواء كان TAS ≥ A أو TAS < A ، أي سواء كان المبلغ المدخر أكبر من النصاب أو يساويه أو أقل منه ، وهذا طبعا خطأ ...

ولكن إذا بدلنا ترتيب العبارتين الأولى والثانية في هذا الحل الخطأ فإنه يصبح صوابا كما يلي :

الحل الرابع :

```
Z := 0.0 ;
IF TAS >= A  THEN  Z := TAS / 40 .0 ;
WRITELN ('TAS = ' , TAS , 'Zakat = ' , Z) ;
```

في هذه الحالة : تعطى Z أولا القيمة صفر ، ثم تعدل هذه القيمة لتحل محلها قيمة المقدار TAS / 40.0 في حالة ما إذا كان TAS ≥ A ، أما في حالة ما إذا كان TAS < A فإن القيمة صفر تظل كما هي وهذا هو الحل المطلوب.

مثال ٣ – ٩ :

اكتب برنامجا كاملا يترجم إلى الباسكال خريطة سير العمليات في المثال ١ –٢ الذي يحسب زكاة المال لشخص واحد.

الحل :

```
PROGRAM Zakatl (INPUT , OUTPUT) ;
      {PROGRAM FOR COMPUTING ZAKAT OF MONEY
       FOR ONE PERSON }
VAR TAS , A  : REAL ;
BEGIN
      WRITE   ('Enter  TAS , A') ;
      READLN ( TAS , A ) ;
      IF TAS >= A  THEN
            WRITELN ('TAS = ' , TAS , 'Zakat =' , TAS / 40.0)
      ELSE
            WRITELN ('TAS = ' , TAS , 'Zakat = 0.0')
      {End IF}
END {Zakatl}.
```

مثال ٣ – ١٠ :

اكتب برنامجا لإيجاد الجذور الحقيقية لمعادلة جبرية من الدرجة الثانية

$$ax^2 + bx + c = 0$$

ذات معاملات حقيقية (أي ترجم إلى الباسكال خوارزمية أو خريطة سير العمليات في المثال ١ – ١).

الحل :

```
PROGRAM QuadraticEqn ( INPUT, OUTPUT ) ;
      {Finding the real roots of a quadratic equation .
       The coefficients a , b , c , are real numbers}
VAR
      a , b , c {Coefficients} ,
      xl , x2 {Roots} ,
      d {Discriminant} ,
      sd {Square root of d} : REAL ;
BEGIN
      WRITE (′Enter the coefficients a , b , c′) ;
      READLN ( a , b, c ) ;
            {Test for first degree eqn }
      IF    a = 0.0   THEN
            WRITELN (′First degree equation′)
      ELSE
            begin {seconel degree eqn}
                  d := b * b - 4.0 * a * c ;
                  IF   d < 0.0   THEN
                        WRITELN (′Complex  roots′)
                  ELSE
                        begin {find and print the real roots }
                              sd :=  SQRT (d ) ;
                              x1 :=  ( - b + sd ) / ( 2.0 * a ) ;
                              x2 :=  ( - b - sd  ) / ( 2.0 * a ) ;
                              WRITELN (′x1 = ′ , x1) ;
                              WRITELN (′x2 = ′ , x2 )
                        end
            end
END {QuadraticEqn}.
```

ملاحظة : نظرا لأننا نحتاج لكتابة كل من المقدار $b^2 - 4ac$ **(والذي يسمى بالمميز) وجذره التربيعي** $\sqrt{b^2 - 4ac}$ **أكثر من مرة أثناء خطوات الحل (انظر مثلا خوارزمية المسألة مثال ١ – ١) فإنه يفضل أن نعطي المقدار الذي يتكرر ظهوره في الحل اسما**

وذلك في عبارة إسناد حسابية أول مره يظهر فيها ثم نستخدم هذا الاسم في المرات التالية التي يظهر فيها ، وذلك لأننا إذا كررنا كتابة هذا المقدار الذي هو تعبير حسابي دون استخدام اسم له فإن الحاسب يبدأ في حساب قيمته من جديد كل مرة يظهر فيها ، أما إذا أعطيناه اسما في عبارة حسابية فإن قيمته تحسب مرة واحدة فقط أثناء تنفيذ هذه العبارة ، وحين يظهر الاسم مرة أخرى فإن الحاسب يحضر قيمته مباشرة من الذاكرة – دون أن يبدأ في حسابها من جديد – لأنه قام بتخزين هذه القيمة تحت هذا الاسم ... وعامة ينصح باستعمال هذا الأسلوب مع المقادير التي يتكرر ظهورها في أي برنامج.

(ج) بنيات IF المتداخلة :

(Nested IF Structures or Block IF)

الصيغة العامة لهذا النوع هي :

```
IF      e1   THEN
        T1
ELSE    IF   e2   THEN
        T2
ELSE    IF   e3   THEN
        T3
        ⋮
        ⋮
ELSE    IF   en   THEN
        Tn
ELSE
        F ;
```

حيث e_1, e_2..... e_n شروط أو تعابير منطقية وكل من T_1, T_2 ..... T_n , F عبارة (بسيطة أو مبنية) ، ومعنى هذه البنيات المتداخلة : تختبر أولا الشروط e_1 ... e_n على الترتيب حسب ظهورها في البرنامج (أي e_1 أولا ثم e_2 ثم e_3 وهكذا) ، فإذا وُجد أن الشرط e_i هو أول شرط متحقق ، فإن العبارة T_i تنفذ. وإذا لم يتحقق أي من هذه الشروط ، فإن العبارة F تنفذ. وعلى أي حال فبعد تنفيذ أي عبارة (سواء T_i أو F) فإن العبارة التالية التي تنفذ هي تلك العبارة التنفيذية التي تلي بنية IF مباشرة .

ملاحظات :

أي عبارة من العبارات السابقة يمكن أن تكون خالية ، وفي هذه الحالة فلا يتم عمل أي شيء عند تحقق الشرط المقابل. وإذا كانت العبارة F خالية ، فبالتالي لا يعمل

أي شيء عندما تكون كل الشروط غير متحققة. وعندما تكون أي عبارة خالية فيمكن حذف سطر ELSE IF (أو ELSE) المقابل ، ويلاحظ أن كلا من بنية IF....THEN وبنية IF ... THEN...ELSE تعتبر حالة خاصة من هذا النوع : بنيات IF المتداخلة.

مثال ٣ – ١١ :

اكتب برنامجا لقراءة عدد الأغنام N عند الأشخاص ، وحساب زكاة الغنم ZS لهذا الشخص من جدول المسألة رقم ١ – ١ ، ثم طباعة قيمة كل من عدد الأغنام و زكاتها٠

الحل:

يمكن الاستعانة بخريطة سير عمليات البرنامج كالمرسومة في حل المسألة رقم ١–١

```
PROGRAM ZakatSheep( INPUT , OUTPUT ) ;
      {computing Zakat of sheep of one person }
VAR
      N {Number of sheep},
      ZS {Zakat of sheep} : INTEGER ;
BEGIN
      WRITE  (′enter number of sheep′);
      READLN (N) ;
      IF  N < 40 THEN
            ZS := 0
      ELSE  IF N < 121 THEN
            ZS := 1
      ELSE  IF N  < 201 THEN
            ZS := 2
      ELSE IF N < 301 THEN
            ZS := 3
      ELSE
            ZS:= N DIV 100
      {end IF} ;
            {Output the results }
      WRITELN (′N = ′, N, ′ZS = ′, ZS);
END.  {ZakatSheep}
```

مثال ٣–١٢ (تعميم لبرنامج مثال ٣–١٠) :

اكتب برنامج لحساب جذور المعادلة التربيعية :

$$ax^2 + bx + c = 0$$

ذات المعاملات الحقيقية a, b, c باستخدام القانون

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

وباتباع الإرشادات التالية :

(أ) إذا كانت $a = 0$ فاطبع قيمة الجذر الحقيقي $-\frac{c}{b}$ (بفرض أن $b \neq 0$ عندما $a = 0$).

(ب) إذا كانت $d > 0$ (حيث d هو المميز $b^2 - 4ac$) فاطبع قيمتي الجذرين الحقيقيين المختلفين ، كل قيمة على سطر مستقل.

(ج) إذا كانت $d = 0$ فاطبع قيمة الجذر المكرر $\frac{b}{2a}$ – مرتين على سطرين مستقلين.

(i) إذا كانت $d < 0$ فاطبع قيمة كل من الجزء الحقيقي xreal و الجزء التخيلي ximag من الجذرين العقديين / المركبين (complex roots) حيث :

$$xreal = -\frac{b}{2a}, \quad ximag = \frac{\sqrt{-d}}{2a}$$

الحل:

```
PROGRAM QuadraticEqn( INPUT , OUTPUT );
    {program to find the different roots of a quadratic equation}
VAR
    a ,b , c, d , sd, x , x1 , x2 ,
    xreal , ximag :real ;
BEGIN
    WRITE  ('Enter the coefficients a ,b ,c ');
    READLN(a,  b, c);
        {Test for first degree equation}
    IF a = 0.0 THEN
        WRITELN ('single root: x = ' , - c/b );
    ELSE
        begin { Second degree equation }
            {Compute discriminant and test}
            d := b*b - 4.0*a*c;
            IF d > 0.0   THEN
                begin { 2 distinct real roots }
                    sd :=  SQRT (d) ;
                    x1 :=  ( -b + sd ) /(2.0 * a) ;
                    x2 :=  ( -b - sd ) /(2.0 * a) ;
```

```
                    WRITELN ('First root x1 = ' , x1);
                    WRITELN ('Second root x2 = ' , x2)
                end
            ELSE IF d = 0.0 THEN
                begin { repeated root }
                    x := -b / (2.0 * a ) ;
                    WRITRLN ('First root x = ', x);
                    WRITELN (' Repeated root x = ', x)
                end
            ELSE
                begin {conjugate complex roots}
                    xreal  := -b / (2.0 * a) ;
                    ximag := SQRT (-d) / (2.0 *a) ;
                    WRITELN('xreal = ', xreal,
                        'ximag = ',ximag)
                end
            {End Nested IF}
        end
    {End IF}
END{QuadraticEqn} .
```

(٢) عبارة الحالة (أو بنية الشرط الانتقائي)

Case Statement (structure)

الصيغة العامة لهذه البنية هي :

```
CASE  e   OF
      L1  :  S1 ;
      L2  :  S2 ;
      ⋮
      ⋮
      Ln  :  Sn
END {CASE}
```

حيث :

e : **تعبير [حسابي صحيح** (integer expression) **أو منطقي** (boolean) **أو رمزي** (character)]

L_i ; i = 1,2,...., n : **قائمة ثوابت [صحيحة** (list of integer constants) **أو منطقية أو رمزية] وتسمى قائمة أسماء الحالة** (case lable list) **والقائمة**

تشمل ثابتا أو أكثر من نفس نوع التعبير.

; S_i ; 1,2,....,n : عبارة (بسيطة أو مبنية).

ومعنى هذه البنية نفّذ العبارة S_i إذا كانت القيمة الحالية للتعبير e تساوي إحدى قيم القائمة L_i.أي أنه حسب حالة (أي قيمة) التعبير e نفذ العبارة المقابلة في البنية.

وبأسلوب آخر : إذا كانت القيمة الحالية للتعبير e تساوي إحدى قيم L_1 نفذ S_1 وإذا كانت قيمة التعبير تساوي إحدى قيم L_2 نفذ العبارة الثانية S_2 ، وهكذا.

أي أنه يتم انتقاء عبارة واحدة فقط لتنفذ بناء على قيمة التعبير e ، وبعد ذلك ينتقل التنفيذ إلى أول عبارة بعد {CASE} END.

مثال ٣ – ١٣ :

تُعَرِّف حدوديات (لاجاندر) (Legendre polynomials) الخمس الأولى بالعلاقات التالية :

$P_0(x) = 1$

$P_1(x) = x$

$P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}$

$P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x$

$P_4(x) = \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$

نفرض أننا أردنا عند نقطة ما في أحد البرامج قراءة قيمة كل من المتغير الحقيقي x والمتغير الصحيح n ، وحساب حدوديات لاجاندر $P_n(x)$ للمتغير x والمقابلة لقيمة n $(1 \le n \le 4)$ بحيث أنه إذا كانت n = 1 نحسب $P_1(x)$ ، وإذا كانت n = 2 نحسب $P_2(x)$ وهكذا.

استخدم عبارة الحالة لتنفيذ ذلك.

الحل :

```
READLN (x , n ) ;
CASE  n  OF
      1  :  P := x ;
      2  :  P := 1.5* x*x - 0.5 ;
```

```
    3  :  P := 2.5 * Exp(3.0 * LN(x))- 1.5* x ;
    4  :  P := 35.0 /8.0 * SQR(SQR(x)) - 15.0/4.0*x*x + 3.0/8.0
END {CASE} ;
```

ملاحظات على بنية الحالة (CASE structure)

(١) **لا يجوز أن يكون التعبير** e **حقيقيا** (real).

(١) **لا يشترط أن تأخذ ثوابت القوائم** $L_1, L_2, \ldots, L_n$ **القيم** 1,2,.... **(أي الأعداد الصحيحة مرتبة) كما في مثال ٣ – ١٣ ، ولكن يمكن أن تأخذ أي قيم وبأي ترتيب ، كما يتضح من حل المثال التالي.**

مثال ٣ – ١٤ :

المطلوب تعديل صيغة الدالة $Y \equiv Y(x)$ **بناء على قيمة التعبير الصحيح** $f(x) \equiv 3x + 4$ **كما يلي :**

(i) **إذا كانت قيمة التعبير** f **تساوي** 1 **فإن** Y = -5.

(ii) **إذا كانت** f = 2 **أو** f = 5 **فإن** $Y = x^2$.

(iii) **إذا كانت** f = 4 **أو** f = 10 **فإن** Y **تحتفظ بصيغتها الأصلية دون أي تعديل.**

(iv) **وأخيرا إذا كانت** f = 7 **فإن** Y = 2x.

استخدم بنية CASE **لعمل هذا التعديل المطلوب لصيغة** Y.

الحل :

```
CASE (3 * x + 4 ) OF
    1      :  Y := - 5 ;
    2,5    :  Y := SQR (x) ;
    4, 10  :  ;     {العبارة الخالية }
    7      :  Y := 2 * x
END;   { CASE}
```

تابع : ملاحظات على بنية الحالة

(٣) **أي عبارة** S_i **يمكن أن تكون مبنية أو بسيطة ، وبالتالي يمكن أن تكون خالية (كما في مثال ٣ – ١٤) حيث لا يُطلب تنفيذ أي شيء للقيم المناظرة للتعبير (للقيمتين ٤ ، ١٠ في هذا المثال).**

(٤) **من الخطأ أن يظهر أي ثابت أكثر من مرة واحدة في بنية الحالة (أي أن يظهر في أكثر من قائمة واحدة) فمثلا البنية التالية خاطئة لظهور** 4 **مرتين :**

```
CASE  i   OF
          4    :   m := 2 * i  ;
          2,7  :   m := i + 5 ;
          1,4  :   m := 10
END ;
```

(٥) من الممكن أن تكتب العبارة نفسها S مع قائمتين ($L_i , L_j ; i \neq j$) أو أكثر.

(٦) ترتيب الثوابت في أي قائمة L_i غير مهم ، وكذلك ترتيب القوائم (مع العبارات المقابلة لها) غير مهم. فمثلا من الممكن أن يكتب حل مثال ٣ – ١٤ كما يلي :

```
CASE   (3 * x + 4 ) OF
       10 ,4 :       ; { العبارة الخالية }
       2 , 5 :       Y := SQR (x) ;
       7     :       Y := 2 * x ;
       1     :       Y := - 5
END       { CASE }
```

(٧) يجب أن تحتوى إحدى قوائم الثوابت L_i على القيمة الحالية للتعبير e ، وإلا أعطى الحاسب " رسالة خطأ " (error message) وأوقف تنفيذ البرنامج.

(٨) كما ذكرنا في تعريف بنية الحالة يمكن أن يكون التعبير e حسابيا صحيحا أو منطقيا أو رمزيا كما في المثال التالي.

مثال ٣ – ١٥ : (على بنية CASE الرمزية)

نفرض أن لدينا التعريف التالي :

```
var
      DAY : char ;
```

ونفرض أن المتغير الرمزي DAY يمكن أن يأخذ إحدى القيم السبع التالية :

'F' , 'S' , 'U', 'M' , 'T', 'W' , 'H'

والتي تشير إلى أيام الأسبوع السبعة – على الترتيب – حيث :

'F' : تعني يوم الجمعة ، 'S' : يوم السبت ، ..... ، 'H' : يوم الخميس.

اكتب عبارة CASE تطبع اسم اليوم المقابل لقيمة المتغير DAY. مثلا إذا كانت قيمة DAY هي 'S' فإن العبارة تطبع كلمة Saturday.

الحل :

```
CASE DAY OF
      'F'    :  WRITELN ('Friday') ;
```

```
        'S'    :  WRITELN ('Saturday') ;
        'U'    :  WRITELN ('Sunday') ;
        'M'    :  WRITELN ('Monday') ;
        'T'    :  WRITELN ('Tuesday') ;
        'W'    :  WRITELN ('Wednesday') ;
        'H'    :  WRITELN ('Thursday')   ;
END ;  {CASE}
```

تمرينات رقم ٣

أولا : تدريبات

٣ – ١ ترجم إلى لغة الباسكال العبارات التالية :

(أ) إذا كان $a^2 \le b + c$ فإن $y = 3\ x$.

(ب) إذا كان $7 < x < 10$ فإن $t = \sin(x + ny)$ وإلا فإن $a = 0$.

(ج) إذا لم تكن a أكبر من b فاطبع قيمة a.

(د) إذا كان $f \le g^3 + h$ أو كانت d غير سالبة فإن $y = e^x$.

(ه) $k = 1$ إذا كان $a - b \le 100$ ، بينما $k = 2$ فيما عدا ذلك.

(و) اجعل $t = 2\ x$ إذا أصبحت قيمة x سالبة أو أصبحت أقل من قيمة المجموع $y + z$.

(ز) اعكس إشارة كل من y , z إذا كانا سالبين.

(ح) احسب x من العلاقة $x := 2\ x + 1$ إذا تحقق الشرط التالي

$$|y| < \varepsilon_2 \text{ و } |\Delta| < \varepsilon$$

إذا كان $f \ne g$ أو كان للمتغيرين a , b إشارتان متعاكستان فإن $f = \phi^{2a-1}$

٣–٢ اكتشف أي أخطاء موجودة في العبارات التالية المكتوبة بلغة الباسكال وصححها إن أمكن .

```
One.  IF x <= 14.0 write(x)  ;
Two. IF i := K  OR  i + j >8  THEN  I = 10 ;
Three.        IF a OR b < 82.0 THEN Z := Y ;
Four. IF ( a < 100.0 ) AND (J = -3)  THEN  K := K + 1 ;
Five. IF SQR(x) <= upper bound THEN y := x ;
Six.   IF x := 0.0 THEN XO = XO + 1.0 ;
Seven.        IF x > y THEN  x := x + t ;
Eight.        IF X > 0.0 THEN
                    Y := Y + 1.0 ;
                    WRITELN(X , Y)
        END IF
Nine. IF TAS >= A THEN
                Z = TAS * 2.5 / 100 . ;
        ELSE :
                Z = 0.0 .
Ten.  IF  a > 0.0  AND  IF a < b THEN
```

```
b := b + 1.0 ;
```

٣-٣ نفرض أن لدينا القيم التالية :

```
a =  -10
b =    0
c =   10
d = 100
```

ما هي قيمة كل من التعابير المنطقية التالية ؟

```
 One)(a > c) and ( d = 100)
Two) (a > c) or (d = 100)
Three)(a = c) or ( b = c)
Four) (not (a = c)) and (not (b = c))
a)    (a = c) or (b <> c) and (c < 10)
```

٣-٤ ما هي المخرجات الناتجة عن تنفيذ مجموعة العبارات التالية ؟

```
program one (output) ;
var
      a , b , c: real ;
begin
      a := 5.0 ;
      b := 5.5 ;
      c := 4.5 ;
      if a > b then
            writeln ('FIRST')
      else if a > c then
            writeln ('SECOND')
      else if b > a then
            writeln ('THIRD')
      else
            writeln ('FOURTH')
end.
```

٣-٥ ماذا تطبع قطعة البرنامج التالية ؟

```
var
      x , y : real ;
      over , out : boolean ;
begin
      x := 15.3 ;
      y := 51.0 ;
      if x > 5.0 then
            over := false ;
      if y < 0.0 then
```

```
        out := false
    else
        out := true ;
    if over and out then
        writeln ('DONE')
    else
        writeln ('NOT DONE') ;
end.
```

٣ – ٦ نفرض أن X = 7.0 , Y = 3.5 , Z = 6.0 , T = 3.4. أي التعبيرات المنطقية المركبة التالية صادقة وأيها خاطئة ، ولماذا ؟

a) (X / Y * Z > T) OR (Y + Z < X)
b) (X / Y * Z > T) AND (Y + Z < X)
c) (X / Y * Z < T) OR (Y + Z < X)

٣ – ٧ نفرض أن i , j , k ثلاثة متغيرات صحيحة. اكتب قطعة برنامج تطبع إحدى الرسائل الأربع التالية :

(i EQUAL **إذا كان** i = j = k.

(ii ASCENDING ORDER **إذا كان** $i \leq j \leq k$.

(iii DESCENDING ORDER **إذا كان** $i \geq j \geq k$.

(iv OUT OF ORDER **إذا لم تتحقق أي من الشروط السابقة.**

٣ – ٨ نفرض أن S متغير حقيقي يمثل (Score) مجموع درجات طالب محسوبا من ١٠٠ درجة. استخدم :

(أ) بنية IF المتداخلة

(ب) بنية CASE.

لطباعة التقدير الذي يحصل عليه هذا الطالب تبعا للجدول التالي

الدرجة	٩٠ – ١٠٠	٨٠ – ٨٩	٧٠ – ٧٩	٦٠ – ٦٩	٠ – ٥٩
التقدير	A	B	C	D	F

٣ – ٩ نفرض أن G متغير رمزي يمثل تقدير طالب بالحروف حيث يأخذ G القيم (الحروف) : A , B , C , D , F

استخدم بنية CASE

(أ) لطباعة التقدير بالألفاظ المقابل للتقدير بالحروف ، تبعا للجدول التالي :

التقدير بالحروف	A	B	C	D	F
التقدير بالألفاظ	ممتاز	جيد جدا	جيد	مقبول	راسب

(ب) لزيادة العدد المقابل للطلاب الذين حصلوا على هذا التقدير بواحد.

فمثلا في حالة تقدير B ، يطبع البرنامج الرسالة : very good ، ويزيد عدد الطلاب الحاصلين على تقدير جيد جدا NB بواحد ، هكذا : NB := NB + 1 ، وفي حالة التقدير D يطبع الرسالة PASS ويزيد ND بواحد ، وهكذا.

٣ – ١٠ ما هي مخرجات قطعة البرنامج التالية بالضبط عندما تكون قيمة X :

(أ) 5.53 (ب) 9.95

```
if X >= 7.5 then
      begin
            X := 90 . 0 ;
            Writeln (' X is' , X : 4 : 2 )
      end
else
            X := 25 . 0 ;
            writeln (' X is' ,  X : 3 : 1 ) ;
```

٣ – ١١

(أ) نفرض أن لدينا عبارة الإسناد :

```
t := Trunc (angle / 90) mod 4 ;
```

ما هي قيم t المقابلة لقيم angle التالية ؟

a)22.5 ii) 80.0 iii) 135.0 iv) 300.0 v) 400.0

(ب) تقوم عبارة CASE التالية بتحديد الربع الذي تقع فيه الزاوية angle.

```
CASE Trunc (angle/90) mod 4 of
          0:  writeln (' first quadrant ') ;
          1:  writeln ('second quadrant') ;
          2:  writeln ('third quadrant') ;
          3:  writeln ('fourth quadrant')
end ;
```

اكتب عبارة IF متداخلة مكافئة لعبارة CASE السابقة.

٣ – ١٢ هل كل قطعتين متقابلتين فيما يلي متكافئتان ؟

إن كانت الإجابة بالنفي فلماذا ؟

(a)
```
if response = ′ Q′ then
      switch := true
   else
      switch := false ;
```

```
switch := response = ′Q′ ;
```

(b)
```
if x > y then
      t := 0
else
      t := 5 ;
```

```
xGreater := x > y ;
Case xGreater of
      true :  t := 0 ;
      false : t := 5
end ;
```

(c)
```
if x  > 5 then
      begin
            if y > 0 then
                  z :=  1
            else
                  z := 2
      end ;
```

```
if x >  5  then
 begin
      if y > 0 then
            z :=  1
 end
      else
            z := 2 ;
```

٣ – ١٣ المفروض أن تقوم كل من القطعتين التاليتين بتعيين أصغر عنصر smallest **في مجموعة من ثلاثة أعداد حقيقية** a, b, c.

(a)
```
smallest := a ;
if  b < smalest then
smallest := b ;
if  c   < smallest  then
smallest := c;
writeln (smallest) ;
```

(ii)
```
if a <= b and a <= c then
      smallest := a;
  else  if  b <= c  and  b <= a then
smallest := b ;
  else
      smallest :=c ;
  writeln (smallest) ;
```

هل هناك أي أخطاء في أي من القطعتين ؟

إن كان هناك أخطاء فما تصحيحها ؟

٣–١٤ ما هي قيمة التعبير المنطقي التالي ؟

```
true and ( (30  mod  10 )  = 3 )
```

هل القوسان الخارجيان ضروريان ؟ و لماذا ؟

و هل القوسان الداخليان ضروريان ؟ و لماذا ؟

٣-١٥ نفرض أن كلا من I , J متغير صحيح و أن flag متغير منطقي.

اكتب عبارة بلغة الباسكال تعطي flag القيمة true إن كان I يقبل القسمة على J بدون باق و تعطيه القيمة false ما عدا ذلك٠

٣-١٦ ينص قانون الغرامات المالية لتجاوز الحد الأقصى للسرعة على ما يلي :

إذا تجاوزت السرعة ٦٠ كم/ ساعة فإن الغرامة تساوي ٢٠ دينارا , فاذا ما زادت السرعة عن ٨٠ كم/ساعة فان الغرامة تزيد إلى ٥٠ دينارا , و إذا زادت السرعة عن ١٠٠ كم/ساعة فان الغرامة تصل إلى ٧٠ دينارا٠

فيما يلي ترجمتان لهذا القانون باستخدام عبارة if :

(a)
```
if speed > 100 then
  fine := 70
else  if  speed > 80 then
  fine := 50
else if speed > 60 then
 fine := 20 ;
```

(ii)
```
if speed  >  60 then
    fine :=20
else if speed > 80 then
    fine := 50
else if  speed > 100 then
    fine := 70 ;
```

(أ)أي العبارتين تعد ترجمة صحيحة للقانون ؟ و أيهما أفضل إن كانتا صحيحتين ؟

(ب)إذا كانت السرعة = ١٠٠ كم/ساعة فكم تساوي الغرامة تبعا لكل من العبارتين ؟

(i) ترجم قانون الغرامات باستخدام عبارة case (بفرض أن السرعة عدد صحيح)٠

٣-١٧ ما هي مخرجات عبارة الطباعة التالية ؟

```
writeln ( 1= 2 , 2= 2 , ( 2 + 3) = 7 ) ;
```

٣-١٨ ما وظيفة عبارة case في قطعة البرنامج التالية ؟

```
write (n) ;
case  n of
      1 : writeln ('st') ;
      2 : writeln ('nd') ;
      3 : writeln ('rd') ;
      4 , 5 , 6 , 7 , 8 , 9 : writeln ('th') ;
end ;
```

ما هي مخرجات القطعة السابقة إذا كانت n = 8 ؟

٣-١٩ جاء انتصار المجاهدين في أفغانستان على قوى الكفر و الإلحاد ، و انهيار الشيوعية , و فتح كابل تصديقا لوعد الله الحق بنصر المجاهدين في سبيل الله بأموالهم وأنفسهم.

المطلوب : تبسيط عبارة case التالية إلى أبسط صوره ممكنة٠

```
CASE  number of
        0 : writeln (′Victory or Martyrdom′);
        1 : writeln (′ Jihad is the only way′);
        2 : writeln (′ Victory or Martyrdom′);
        3 : writeln (′ first opening of Afghanistan was in 21 a.h′);
        4 : writeln (′ Jihad is the only way′);
        5 : writeln (′Victory or Martyrdom′);
    END;
```

٣-٢٠ ما هي مخرجات البرنامج التالي المقابلة لكل من المدخلات المعطاة (بعد البرنامج) ؟

```
program student (input , output) ;
    var
        StudentID, FinalScore ,
        ExamsScore, Point       : integer ;

        FinalGrade :            : char ;
    begin
{= = = = = = = = = = = = = = = = = = = = = = = = }
    {read in the input data }

{= = = = = = = = = = = = = = = = = = = = = = = = }
    write (′enter student′′ s ID′) ;
    readln (StudentID) ;
    write (′enter student′′s exams score′) ;
    readln ( ExamsScore );
    write (′enter student′′s final  score′) ;
    readln ( FinalScore ) ;
{= = = = = = = = = = = = = = = = = = = = = = = = }
{ checking validity of  input data  }
{= = = = = = = = = = = = = = = = = = = = = = = = }
    if StudentID <= 0 then
        writeln (′error : invalid StudentID′)
    else if (FinalScore < 0 ) or (ExamsScore < 0) or
        (FinalScore >100 ) or (ExamsScore > 100)  then
        writeln (′error: invalid scores.′)
    else  {StudentID , FinalScore , and  ExamsScore  are valid}
        begin
{= = = = = = = = = = = = = = = = = = = = = = = = }
    { compute the final grade }
```

```
{= = = = = = = = = = = = = = = = = = = = = = = = }
     if (ExamsScore < 50 ) or (FinalScore < 60 ) then
          FinalGrade := ′F′
     else if (ExamsScore >= 70 ) and (FinalScore >= 90 ) then
          FinalGrade := ′A′
     else if (ExamsScore >= 65 ) and (FinalScore >= 80 ) and
          (FinalScore < 90) then
          FinalGrade := ′B′
     else if ( ExamsScore >= 60 ) and ( Finalscore >= 70 ) and
          (FinalScore < 80 ) then
          FinalGrade := ′C′
     else if (ExamsScore >= 50 ) and (FinalScore >= 60 ) and
               (FinalScore < 70 ) then
          FinalGrade := ′D′
     else  {undecided grades}
          begin
               writeln (′check StudentID number′ , studentID, ′record′);
               FinalGrade := ′U′
     end; {else }
{= = = = = = = = = = = = = = = = = = = = = = = = }
        { compute the earned point }
{= = = = = = = = = = = = = = = = = = = = = = = = }
          case Finalgrade of
               ′A′          : Point := 4 ;
               ′B′          : Point := 3 ;
               ′C′          : Point := 2 ;
               ′D′          : Point := 1 ;
               ′F′ , ′U′    : Point := 0
          end ; {case  ...}
{= = = = = = = = = = = = = = = = = = = = = = = = }
     {print the outputs }
{= = = = = = = = = = = = = = = = = = = = = = = = }
     write(′StudentID =′ , StudentID:9);
     write(′Grade = ′ , FinalGrade:2);
     writeln(′Point = ′, Point:2)
          end
     end.
```

1. Input : StudentID = -1, ExamsScore = 10, FinalScore = 20.
2. Input : StudentID = 1234, ExamsScore = 101 , FinalScore = 20.
3. Input : StudentID = 1235, ExamsScore = 10 , FinalScore = 201.

4. Input : StudentID = 1236, ExamsScore = 90 , FinalScore = 90.
5. Input : StudentID = 1237, ExamsScore = 60,FinalScore = 90.
6. Input : StudentID = 1238, ExamsScore = 70,FinalScore = 85.
7. Input : StudentID = 1239, ExamsScore = 61, FinalScore = 71.

٣ – ٢١ 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 – ٢٢ 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

استخدم عبارة CASE.

٣ – ٢٣ 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 ، ولاعبرة بثمن شرائها وتكلفتها ، ولا بالسعر المرغوب بيعها به) ، وذلك إذا بلغت هذه القيمة السوقية للمواد التجارية نصابا من الذهب (أو الفضة ، ونصاب الذهب = ٨٥ جم ذهبا ، ونصاب الفضة = ٥٩٥ جم فضة) بشرط حولان الحول (والحول يبدأ منذ الشراء بنية التجارة) [ملاحظة : الأثاث والأجهزة المستخدمة لصالح عرض التجارة وبيعها أو خزنها أو نقلها كالسيارات ونحوها لا زكاة فيها].**

المطلوب :

كتابة برنامج يقرأ سعر جرام الذهب PG ، والقيمة السوقية MV للعروض التجارية ، وقيمة متغير رمزي Year **(بحيث أن القيمة 'Y' تعني حولان الحول ،**

بينما القيمة 'N' تعني عدم حولان الحول) ، ثم يحسب قيمة زكاة عروض التجارة ZT.

٣ – ٢٤ تعطَى بعض المعادن تقديرا يعتمد على نتائج اختبارات ثلاثة. وهذه الاختبارت تحدد ما إذا كان المعدن يحقق المواصفات التالية :

(١) المحتوى الكربوني (Carbon Content) أقل من ٦,٧ : (T_1).

(٢) ثابت الصلابة (Rockwell Hardness Const.) ليس أقل من ٥٠ : (T_2).

(٣) مقاومة الشد (Tensile Strength) أكبر من ٧٠,٠٠٠ وحدة / بوصة^2 : (T_3).

ويعطَى المعدن تقديرا حسب الشروط المذكورة بالأولويات التالية :

	التقدير		الشروط
(أ)	١٠	.....	إذا نجح في الاختبارات الثلاثة.
(أ)	٩	.....	إذا نجح في الاختبارين (١) ، (٢) فقط.
(أ)	٨	.....	إذا نجح في الاختبار (١) فقط.
(أ)	٧	.....	ما عدا ما سبق.

اكتب برنامجا بلغة الباسكال 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 – ٢٥ 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 :

i) " ولأبويه لكل واحد منهما السدس مما ترك إن كان له ولد " ، أي أن :

نصيب الأب (f) = نصيب الأم (m) : $f = m = \frac{A}{6}$

ii) " فإن كان لكم ولد فلهن الثمن مما تركتم " ، أي أن :

نصيب الزوجة : $w = \frac{A}{8}$

iii) الباقي : $r = A - (f + m + w)$ يوزع على الأبناء والبنات تبعا للقاعدة :
" يوصيكم الله في أولادكم للذكر مثل حظ الأنثيين " ، أي أن :

iv) نصيب الإبن : $s = \frac{2r}{2NS+ND}$

v) 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 – ٢٦ اكتب برنامجا (يمكن أن يعد جزءا من برنامج عام لتوزيع الميراث) يحيث يقوم هذا البرنامج الجزئي بتحديد نصيب كل من الزوج H أو الزوجة W وكل واحد من الأبناء (نصيب الإبن S ونصيب البنت D) ، وذلك تبعا للقواعد التالية :

أولا : الأزواج :

(١) ولكم نصف ما ترك أزواجكم إن لم يكن لهن ولد ، فإن كان لهن ولد فلكم الربع مما تركن.

(٢) ولهن الربع مما تركتم إن لم يكن لكم ولد ، فإن كان لكم ولد فلهن الثمن مما تركتم.

ثانيا : الأبناء :

(أ) في حالة عدم وجود أبناء ذكور :

(١) فإن كن نساءً فوق اثنتين فلهن ثلثا ما ترك.

(وأيضا إن كانتا اثنتين فلهما ثلثا ما ترك)

(٢) وإن كانت واحدة فلها النصف

(ب) في حالة وجود أبناء ذكور :

يوصيكم الله في أولادكم للذكر مثل حظ الأنثيين.

ملاحظات :

(أ) يبدأ البرنامج بقراءة قيمة الميراث A ، وعدد الأبناء الذكور NS والإناث ND ، وقيمة رقم ثنائي I الذي يأخذ القيمة 1 إذا كان الشخص المتوفي هو الزوج ، والقيمة صفر إذا كان المتوفي هو الزوجة.

(ب) لا يحدد البرنامج نصيب بقية الأقارب غير المذكورين.

(ج) يراعى استخدام عناوين مناسبة عند طباعة النتائج.

٣ – ٢٧

– عند المقارنة بين عدة منازل لشراء منزل جديد ، يجب أن نأخذ في الاعتبار عدة عوامل وهي : التكلفة الأولية للمنزل ، وتكلفة الوقود السنوية التقديرية ، ومعدل الضريبة السنوية. اكتب برنامجا لتعيين التكلفة الكلية للمنزل بعد خمس سنين وذلك لكل مجموعة من مجموعات البيانات التالية ، ومن ثم يمكننا تحديد أفضل منزل للشراء.

تكلفة المنزل الأولية	تكلفة الوقود السنوية	معدل الضريبة
Initial house cost	Annual fuel cost	Tax Rate
$ 67, 000	$ 2,300	0.025
$ 62, 000	$ 2,500	0.025
$ 75, 000	$ 1,850	0.020

لحساب تكلفة المنزل الكلية أضف التكلفة الأولية الى تكلفة الوقود لخمس سنين ، ثم أضف الضريبة لخمس سنين ، علما بأن الضريبة لسنة واحدة تحسب بضرب معدل الضريبة في التكلفة الأولية.

٣ – ٢٨ 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).

٣-٢٩ – تقوم مؤسسة اتصالات تليفونية بالتعامل مع العملاء في تسديد الفواتير على النحو التالي :

- يعرض خصم 50% للمكالمات التي تبدأ بعد .6 p.m (الساعة 1800) وقبل 8: 00 a.m (الساعة 0800)
- ليس هناك أي خصم للمكالمات التي بعد .8.:00 a.m (الساعة 0800) وقبل 6:00 p.m (الساعة 1800)
- تفرض ضريبة على كل المكالمات مقدارها 4%
- سعر الدقيقة في المكالمة يساوي 0.40 $
- هناك خصم 15% من المبلغ إذا كانت مدة المكالمة أكثر من ٦٠ دقيقة (وذلك بعد طرح أي خصومات أخرى وقبل إضافة الضريبة)

اكتب برنامجا لقراءة وقت بداية المكالمة وطول المكالمة بالدقائق ، ومن ثم طباعة المبلغ الكلي (وهو المبلغ قبل أي خصم أو ضريبة) وطباعة المبلغ النهائي (بعد الخصم والضريبة).

٣ – ٣٠ 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 اليها 0.0005 $ لكل جالون.
- بالنسبة للشركة C : القيمة 1000$ لأول ٤ مليون جالون ويضاف اليها 0.00025$ لكل جالون زيادة عن ٤ مليون جالون.
- بالنسبة للمصنع I : القيمة 1000$ إذا كان عدد الجالونات لايزيد عن ٤ مليون جالون٠ وتساوي 2000$ إذا كان عدد الجالونات أكثر من ٤ مليون

جالون ولا يتعدى ١٠ مليون جالون. وتساوي 3000 $ إذا كان عدد الجالونات يزيد عن ١٠ مليون جالون.

ويقرأ البرنامج رقم الحساب (وهو عدد صحيح) وشفرة المستهلك (وهي من النوع الرمزي char) ، وعدد جالونات المياه (وهو عدد حقيقي) ، ثم يطبع هذه البيانات وقيمة الفاتورة.

٣ – ٣١ 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 الخالص.

ب) زكاة الغنم ZS و تحسب من الجدول التالي :

عدد الأغنام NS	أقل من ٤٠	٤٠–١٢٠	١٢١–٢٠٠	٢٠١–٣٠٠	أكثر من ٣٠٠
زكاة الغنم ZS	صفر	١	٢	٣	في كل مائة شاة

ج) زكاة الثمار والفاكهة ZF. و تقدر بالقاعدة التالية :

إذا كانت كمية الثمار F أقل من مقدار معين معطى AF (يسمى النصاب) فإن الزكاة تساوي صفرا : ZF = 0 ، و ما عدا ذلك فإنه

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

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 – ٣٢ اكتب برنامجا لحل المعادلتين الخطيتين الآتيتين :

$$a_1 x + b_1 y = c_1$$
$$a_2 x + b_2 y = c_2$$

بعد قراءة مجموعتي الثوابت a_1 , b_1 , c_1 **و** a_2 , b_2 , c_2 **(راجع المسألة رقم ١-٢١).**

الفصل الرابع

عبارات التكرار
Repetition Statements

في كثير من البرامج نحتاج أن نكرر عدة مرات تنفيذ عملية معينة – تعبِّر عنها عبارة بسيطة أو عبارة مركبة (مبنية) – كعمليات القراءة والطباعة والحساب والمقارنة ، كأن نريد مثلا عمل جدول يعطي قيم دالة f(x) المقابلة لقيم عديدة للمتغير x (مثلا من x = 2.5 إلى x = 8.5 بزيادة منتظمة تساوي 0.1) ، أو نريد حساب قيم زكاة المال لعدد كبير من الأشخاص ، أو نود حل معادلة في مجهول بتطبيق إحدى الطرق التكريرية (وسنرى بإذن الله في هذا الفصل مثالا لإحدى هذه الطرق) ، أو نود إيجاد قيمة مجموع متسلسلة معينة بجمع حدودها المتتالية ، طالما أن القيمة المطلقة للحد أكبر من قيمة محددة.

في مثل هذه الحالات نحتاج لعبارات "تكريرية" توجِّه الحاسب إلى أن يكرر تنفيذ مجموعة معينة من العبارات وذلك لبيانات مختلفة أو لقيم مختلفة لمتغيرات معينة. وقد يكون هذا التكرير عددا محددا من المرات ، أي عددا معلوما منذ البداية (ويسمى تكريرا غير مشروط) ، أو يكون التكرير مرهونا بتحقيق شرط معين (ويسمى تكريرا مشروطا)..

وفي هذا الفصل سندرس بإذن الله تعالى نوعين من العبارات التكريرية :

أولا : التكرير غير المشروط. حيث ندرس :

عبارة FOR ولها صيغتان :

(١) عبارة FOR .. TO .. DO ..

(٢) عبارة FOR .. DOWNTO .. DO

ثانيا : التكرير المشروط ، وفيه ندرس :

(١) عبارة WHILE .. DO ..

(٢) عبارة REPEAT .. UNTIL ..

* * *

أولا : التكرير غير المشروط :

(١) بنية FOR .. TO .. DO ..

والصيغة العامة لهذه البنية هي :

FOR i := initial TO final DO St

حيث :

i : **اسم تعريفي لمتغير صحيح يسمى متغير التحكم في العروة** (loop control variable)

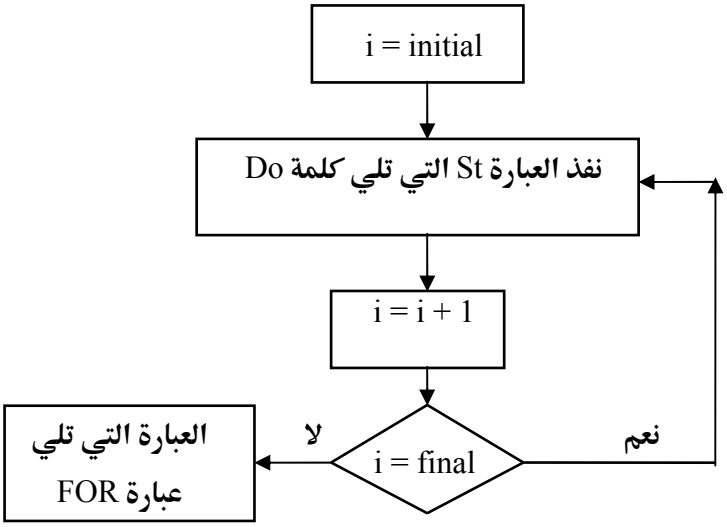
initial , final : **كل منهما تعبير حسابي صحيح ، ويسميان : (وسيطا العروة)** (loop parameters) **، حيث يعطي** initial **القيمة الابتدائية للمتغير** i **ويعطي** final **القيمة النهائية له.**

St : **عبارة بسيطة أو مركبة ، تسمى جسم العروة** (loop body).

ومعنى هذه البنية :

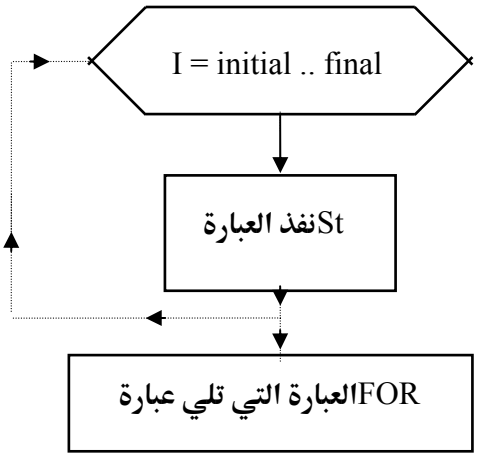
ضع أولا : i = initial **ثم نفذ العبارة** St **، ثم زد قيمة** i **بواحد ، ونفذ العبارة** St **مرة أخرى (ولكن بالطبع كلما تظهر** i **عوض عنها الآن بالقيمة:** initial + 1 **، ثم زد قيمة** i **ونفذ العبارة مرة ثالثة ، وهكذا إلى أن تنفذ العبارة** St **وقيمة** i **تساوي** final **، وبعدها ننتقل إلى العبارة التي تلي عبارة** FOR.

أي أن خريطة سير العمليات التي تقابل عبارة FOR .. TO.. DO .. **هي :**



خريطة سير العمليات المقابلة لتنفيذ عبارة .. FOR .. TO .. DO

ويمكن اختصار هذه الخريطة إلى الاصطلاح التالي :



مثال ٤-١ :

ما هي نتيجة تنفيذ العبارات التالية ؟

(أ) FOR i := 2 TO 50 DO WRITELN (2*i) ;

(ب)

```
FOR i := 2 TO 50 DO
        BEGIN
    Y := SQR(i) ;
WRITELN ('X= ', i, 'Y = ' , Y)
        END ;
```

الحل

(أ) يطبع البرنامج الأعداد الزوجية من 4 إلى 100 ، كل عدد على سطر مستقل ، هكذا

```
4
6
8
:
:
100
```

(ب) يطبع البرنامج جدولا لمربعات الأعداد الصحيحة من 2 إلى 50 في الصورة التالية :

```
X = 2      Y = 4
X = 3      Y = 9
X = 4      Y = 16
:
:
:
X = 50     Y = 2500
```

مثال ٤-٢ :

استخدم عبارة FOR **لقراءة وجمع درجات** item **ألفي طالب ، ثم احسب مجموعها** sum **ومتوسطها** ave .

الحل :

```
sum :=0.0 ;
ROR i := 1 TO 2000 DO
        BEGIN
 READLN (item) ;
sum := sum + item
        END ;
        {End FOR}
ave := sum / 2000.0 ;
WRITELN ('sum = ' , sum , 'ave = ' , ave)
```

ملاحظة :

من الأخطاء الشائعة التي نحب أن ننبه إليها وضع العبارة التي تعطي المتغير sum **قيمته الابتدائية** 0.0 := sum **داخل عروة** FOR **(بدلا من خارجها) هكذا :**

```
FOR i := 1 TO 2000 DO
    BEGIN
        sum := 0.0 ;
    READLN (item) ;
sum := sum + item
        END ;
    {End FOR}
```

وهذا خطأ لأنه لا يؤدي إلى جمع قيم item **كلها (أي درجات الطلاب) بإضافتها واحدة بعد الأخرى إلى** sum **، حيث أنه بعد إضافة أي واحدة من قيم** item **إلى** sum **، وقبل إضافة قيمة** item **التي تليها ، نعيد قيمة** sum **إلى قيمتها الابتدائية** sum = 0.0 **مرة أخرى ، حيث تزيد قيمة** i **بواحد ، ونبدأ في تنفيذ خطوات العبارة** (St) **داخل عروة** FOR **مرة جديدة ، ولذلك فيجب أن توضع العبارة** sum := 0.0 **قبل بنية (عبارة)** FOR **وليس داخلها (انظر حل مثال ٤-٢) حتى لا تنفذ هذه العبارة** sum := 0.0 **إلا مرة واحدة فقط لإعطاء** sum **القيمة الابتدائية.**

مثال ٤-٣ :

اكتب برنامجا لحساب زكاة المال لألف شخص وذلك على ضوء خريطة سير العمليات في المثال ١-٥ ، وباستخدام عبارة FOR **(أنظر مثال ٣-٩).**

الحل :

```
PROGRAM Zakat1000 (INPUT, OUTPUT) ;
                                VAR
                        i, ID : INTEGER ;
                    TAS, A, Z : REAL ;
                                BEGIN
                WRITE ('Enter Al-Nisab') ;
                        READLN (A) ;
                FOR i := 1 TO 1000 DO
                            BEGIN
            WRITE ('Enter ID, TAS') ;
                READLN (ID,TAS);
```

```
                    IF TAS >= A THEN
WRITELN('ID = ' , ID, 'TAS = ' , TAS ,
    'Z = ' , TAS/40.0)
                                ELSE
WRITELN('ID = ' , ID, 'TAS = ' , TAS ,
            'Z = 0.0')
                                    END ;
                                {End FOR loop}
                                END {Zakat1000}
```

ملاحظة :

من الأمثلة الثلاثة السابقة نلاحظ أن متغير التحكم في عروة FOR قد يدخل باسمه في الحسابات التي تجرى في البرنامج (كما في مثال ٤-١) ،أو أن يستخدم فقط لمجرد العد (كما في مثالي ٤-٢ ، ٤-٣).

ملاحظات وقواعد عامة لاستخدام عبارة FOR

- **يجوز أن تكون كل من القيمة الابتدائية initial والقيمة النهائية final ثابتا عدديا أو اسم متغير أو تعبيرا حسابيا ، مثل**

```
FOR j := k TO (2*K + 1) DO
```

- **بصورة عامة يجوز أن يكون متغير التحكم في العروة i (وبالتالي كل من القيمة الابتدائية initial ، والقيمة النهائية final) من النوع المنطقي أو الرمزي أو الحسابي الصحيح ، ولا يجوز أن يكون من النوع الحقيقي.**
- **إذا كانت القيمة الابتدائية أكبر من القيمة النهائية ، كأن نكتب**

```
FOR K := 8 TO 3 DO
```

فإن العروة (عبارة التكرار) لا تنفذ.

- **في حالة عدم تنفيذ العروة ينتقل التحكم (التنفيذ) إلى أول عبارة تلي عبارة FOR .**
- **إذا كانت القيمة الابتدائية تساوي القيمة النهائية ، فإن العروة تنفذ مرة واحدة فقط.**
- **إذا كانت القيمة الابتدائية أصغر من القيمة النهائية ، فإن :**

عدد مرات تنفيذ العروة = القيمة النهائية – القيمة الابتدائية + ١

فمثلا في الحالة :

```
FOR K := 3 TO 8 DO
```

تنفذ العروة ست مرات.

- لا يجوز أن تغيِّر أي عبارة في عروة FOR قيمة i (متغير التحكم في العروة).
- قيمة كل من التعبيرين initial , final تحسب مرة واحدة فقط عند دخول العروة ولا تتأثر بعد ذلك بأي تغيير يحدث في قيمة أي متغير من متغيراتهما عن طريق عبارة من عبارات العروة ، وبذلك فإن عدد مرات تنفيذ العروة لا يتأثر كذلك.
- بعد الانتهاء من تنفيذ العروة ينتقل التحكم إلى أول عبارة بعد العروة.
- بعد الانتهاء من تنفيذ العروة والخروج منها ، فإن قيمة متغير التحكم i تصبح عموما غير معرَّفة.
- إذا رغبنا في تنفيذ عروة FOR لكميات حقيقية بالنسبة لقيم متغير التحكم i ابتداء من القيمة الابتدائية initial وحتى القيمة النهائية final ، فيمكن تعديل هذه القيم إلى كميات صحيحة (بضربها مثلا في عدد ما) ليمكن استخدام بنية FOR ، ثم إجراء العملية العكسية (القسمة على هذا العدد) ، كما في المثال التالي.

مثال ٤-٤ :

استخدم عبارة FOR لحساب العلاقة

$$Y = 2.4 \sin X + 5X^2 - 7$$

للقيم :

$$X = 0 , 0.1 , 0.2 , ... , 10.0$$

الحل :

```
PROGRAM Fn (Output) ;
VAR
    i : INTEGER ;
    X , Y : REAL ;
BEGIN
  FOR i := 0 TO 100 DO
    BEGIN
      X := i / 10.0 ;
Y := 2.4 * SIN(X) + 5.0 * SQR(X) - 7.0 ;
```

```
WRITELN ('X = ' , X , 'Y = ' , Y)
                END {FOR}
            END {Fn}.
```

(٢) بنية FOR .. DOWNTO .. DO ..

والصيغة العامة لهذه البنية هي :

```
FOR i := initial DOWNTO final DO St
```

وكل ما ذكر سابقا عن بنية FOR .. TO .. DO من حيث تعريف صيغتها ومعناها وقواعد استخدامها ينطبق على بنية FOR .. DOWNTO .. DO باستثناء الاختلافات التالية :

في بنية (عروة / عبارة) FOR .. DOWNTO .. DO ..

- **عند تنفيذ العروة ننقص (بدل أن نزيد) قيمة i بواحد تدريجيا إلى أن تصبح قيمة i تساوي final.**

أي أن خريطة سير العمليات المقابلة لتنفيذ العروة هي نفسها الخريطة السابقة مع وضع $i = i - 1$ **بدلا من** $i = i + 1$

وكذلك $i \geq final$ **بدلا من** $i \leq final$ **.**

- **إذا كانت القيمة الابتدائية أصغر من القيمة النهائية ، مثل**

```
FOR K := 8 DOWNTO 3 DO
```

فإن العروة لا تنفذ.

- **إذا كانت القيمة الابتدائية أكبر من القيمة النهائية ، فإن**

عدد مرات تنفيذ العروة = القيمة الابتدائية – القيمة النهائية + ١

فمثلا في الحالة

```
FOR K := 8 DOWNTO 3 DO
```

تنفذ العروة ست مرات (٨ – ٣ + ١ = ٦).

مثال ٤–٥ :

اكتب برنامجا لقراءة قيمة عدد صحيح K ثم حساب

$K! = K(K - 1)(K - 2) \ldots \times 3 \times 2 \times 1$

باستخدام عبارة FOR .. DOWNTO .. DO

الحل :

```
PROGRAM Factorial (INPUT , OUTPUT) ;
```

```
VAR
    K , fact , i : INTEGER ;
BEGIN
    READLN (K) ;
    fact := K ;
FOR i := (K - 1) DOWNTO 1 DO
    fact := fact * i
    {End FOR}
WRITELN ('K = ', K , 'factorial K = ', fact)
END.
```

ملاحظة :

بالطبع يمكن حل المسألة باستخدام بنية FOR للعد التصاعدي بدل التنازلي ، كما يلي :

```
fact := K ;
FOR i := 1 TO (K - 1) DO
    fact := fact * i ;
```

أو كما يلي في حل المثال التالي :

مثال ٤-٦ :

أعد حل مثال ٤-٥ لحساب !K باستخدام عبارة FOR..TO..DO.

الحل :

```
BEGIN
    READLN (K) ;
    fact := 1 ;
FOR i := 2 TO k DO
    fact := fact * i
    {End FOR} ;
WRITELN ('K = ', K , 'factorial K = ', fact)
END.
```

ثانيا : التكرير المشروط

أحيانا نرغب في تكرير عملية معينة طالما أن شرطا معينا متحقق ، فإذا لم يتحقق هذا الشرط أوقفنا التكرير ، أو نرغب في تكرير عملية معينة إلى أن يتحقق شرط معين. في مثل هذه الحالات قد لا يكون عدد مرات التكرير معلوما منذ البداية ولكن يكون التكرير مرتبطا بذلك الشرط ، فتستخدم لذلك إحدى صيغتين :

(١) صيغة WHILE .. DO **(٢) صيغة REPEAT..UNTIL**

(١) بنية WHILE .. DO ..

الصيغة العامة لهذه البنية هي :

WHILE e DO St

حيث

e : **تعبير منطقي (شرط منطقي)**

St : **عبارة بسيطة أو مركبة.**

ومعنى هذه الصيغة :

تحقق أولا من صحة الشرط المنطقي ، فإن كان صحيحا (true) **نفذ العبارة** St **، ثم أعد اختبار صحة الشرط ، فإن كان صحيحا أعد تنفيذ العبارة** St **، وهكذا قم بتكرير تنفيذ العبارة** St **(أي استمر في تنفيذ العروة) ما دام الشرط متحققا ، إلى أن يصبح الشرط غير صحيح** (false) **وعندها انتقل خارج العروة لتنفذ أول عبارة تلي عبارة** WHILE.

مثال ٤-٧ :

اكتب برنامجا لقراءة ألفي قيمة item **وحساب مجموعها** sum **ومتوسطها الحسابي** ave **، وذلك باستخدام عروة** WHILE **(انظر مثال ٤-٢).**

الحل :

```
PROGRAM SumAve2000 (INPUT , OUTPUT) ;
                                    VAR
                          counter : INTEGER ;
                        item , sum , ave : REAL ;
                                          BEGIN
                                    sum := 0.0 ;
                                  counter := 0 ;
                        WHILE counter < 2000 DO
                                            BEGIN
                                 READLN (item) ;
                              sum := sum + item ;
                        counter := counter + 1
                                              END
                                          {End while} ;
                                ave := sum / 2000.0 ;
```

```
WRITELN ('sum = ', sum , 'ave = ', ave)
END. {SumAve2000}
```

ملاحظة على عروة WHILE :

دائما يتم اختبار الشرط e المذكور في عبارة WHILE قبل تنفيذ العروة (العبارة St) ، فإن تحقق الشرط نفذت العروة ، وإن لم يتحقق لم تنفذ ، ومعنى هذا :

(أ) يجب أن تكون قيم جميع المتغيرات المذكورة في الشرط معلومة قبل الوصول إلى عبارة While.

(ب) يجب أن تحتوي العروة على عبارة أو أكثر تغير من قيم بعض أو كل هذه المتغيرات حتى نضمن أنه عند لحظة ما سوف يصبح الشرط غير متحقق ونخرج من العروة ، وإلا أصبحت العروة غير منتهية (infinite loop).

(ج) من الجائز ألا تنفذ العروة إطلاقا ، وذلك إذا لم يتحقق الشرط e عند أول اختبار له.

* * *

ومن الممكن الاستفادة من عبارة WHILE بصورة خاصة لإيقاف تنفيذ البرنامج عند انتهاء ملف البيانات (EOF : End of File) التي يقرأها البرنامج ، وذلك في حالة عدم معرفة عدد سجلات البيانات ابتداء.

مثلا برنامج مثال ٤-٣ يصلح لحساب الزكاة إذا كان عدد الأشخاص ألف شخص ، أي إذا كانت مجموعة البيانات المعطاة تتكون بالضبط من ألف وحدة أو ألف سجل (ألف بطاقة بيانات مثلا أو ألف سطر بيانات) حيث السجل يمثل بيانات الشخص الواحد. ولكن في بعض المسائل لا نعرف عدد سجلات البيانات ، أو نطلب من الحاسب أن يقوم هو بعملية العد ، أو نرغب في كتابة برنامج عام يصلح لأي عدد من الأشخاص وليس لعدد معين فقط ، والمثال التالي يعالج حل مثل هذه المسألة.

مثال ٤-٨ (تعميم لمثال ٤-٣) :

اكتب برنامجا بلغة الباسكال لحساب زكاة المال لعدة أشخاص حيث بيانات كل شخص موضوعة على سجل منفصل (بطاقة مثلا أو سطر).

الحل :

نستخدم سجل بيانات إضافيا (أي بطاقة بيانات إضافية مثلا أو سطرا إضافيا) يوضع في النهاية بعد مجموعة سجلات البيانات ، ونضع عليه عددا أو قيمة خارج مدى قيم البيانات التي على سجلات البيانات ، ونسمي هذه القيمة "القيمة الحارسة" (sentinel value) ، فمثلا في مسألتنا هذه كل القيم في البيانات غير سالبة لأن المبالغ المدخرة غير سالبة (TAS ≥ 0) والأرقام التعريفية موجبة (ID > 0) ولذلك فيمكننا مثلا أن نضع قيمة سالبة للمبالغ المدخرة على السجل الإضافي وفي الوقت نفسه نضع في خطوات البرنامج عبارة تفيد الحاسب أنه عندما يقرأ قيمة سالبة للمبلغ المدخر فهذا يعني أن كل البيانات المطلوب قراءتها قد انتهت ، وذلك دون أن نعرف عدد سجلات هذه البيانات.

ولذلك فلتعديل برنامج مثال ٤-٣ ليصبح حلا لمسألة المثال الحالي:

- **نحذف عبارة FOR بالعداد i ، وكذلك نحذف i من جزء تعريف المتغيرات في قسم الإعلانات.**
- **نضيف عبارة WHILE لإيقاف تنفيذ البرنامج حين نجد أن قيمة TAS المقروءة سالبة. وبذلك يصبح البرنامج كما يلي :**

```
PROGRAM ZakatAnyNumber (INPUT , OUTPUT) ;
VAR
    ID : INTEGER ;
    TAS , A , Z : REAL ;
BEGIN
    WRITE ('Enter Al-Nisab') ;
    READLN (A) ;
    WRITE ('Enter ID , TAS') ;
    READLN (ID , TAS)
    WHILE TAS >= 0.0 DO
        BEGIN
        IF TAS >= A THEN
            Z := TAS / 40.0
        ELSE
            Z := 0.0 ;
        WRITELN ('ID = ' , ID , 'TAS = ', TAS ,
                 'Z = ' , Z) ;
        WRITE ('Enter ID , TAS')
```

```
        READLN (ID , TAS)
    END {WHILE}
END.
```

ملاحظة (١) :

إذا كنا لا نعلم مدى قيم البيانات فيمكننا أيضا استخدام طريقة السطر الإضافي (طريقة القيمة الحارسة) لإنهاء البرنامج وذلك بعمل تعديل طفيف مع الطريقة المذكورة ، فمثلا يمكننا أن نقرأ قيمة متغير إضافي ، وليكن K مثلا ، ونكتب قيمة ما ولتكن 1 مثلا مقابل موضع المتغير K على هذا السطر الإضافي ، بينما نكتب أي قيمة أخرى ولتكن صفرا على كل سطر من سطور البيانات. وبذلك يعرف الحاسب أنه قد انتهى من قراءة كل بيانات المسألة حينما يجد أن قيمة K تساوي 1 .

ولتوضيح معنى هذا الكلام فإن العبارتين التاليتين تحلان محل العبارتين المقابلتين لهما في البرنامج السابق.

```
READLN (ID , TAS , K)
WHILE K = 0 DO
```

ملاحظة (٢) :

في حالة ما إذا كنا نعرف عدد السجلات ولكن نود في الوقت نفسه أن يكون البرنامج عاما يصلح لأي عدد من هذه السجلات فنضع في بداية سجلات البيانات سجلا إضافيا عليه عدد معين يمثل عدد هذه السجلات ، ونضع في خطوات البرنامج خطوة لقراءة هذا العدد المعين الذي يمثل عدد الأشخاص ولنرمز له بالرمز NP ، وكذلك نضع خطوات العد (عبارة FOR مثلا ، أو عبارة WHILE) لإيقاف تنفيذ البرنامج حين نصل إلى هذا العدد NP.

أي أن البرنامج يصبح تعميما أيضا لبرنامج مثال ٤-٣ حيث أن NP تحل محل الرقم الخاص 1000. والمثال التالي يوضح بإذن الله تعالى ما نقصده.

مثال ٤-٩ (تعميم آخر لمثال ٤-٣) :

اكتب برنامجا بلغة الباسكال لحساب زكاة المال لعدد NP من الأشخاص ، علما بأن بيانات كل شخص موضوعة على سجل منفصل.

الحل :

```
PROGRAM ZakatNP (INPUT , OUTPUT) ;
```

```
VAR
    I , ID , NP : INTEGER ;
    TAS , A , Z : REAL ;
BEGIN
    WRITE ('Enter Al-Nisab') ;
    READLN (A) ;
    WRITE ('Enter Number of Persons') ;
    READLN (NP) ;
    FOR I := 1 TO NP DO
        BEGIN
        WRITE ('Enter ID , TAS') ;
        READLN (ID , TAS) ;
        IF TAS >= A THEN
        Z := TAS / 40.0
                ELSE
        Z := 0.0 ;
    WRITELN ('ID= ',ID,'TAS= ',TAS, 'Z=', Z)
                END
END.
```

(٢) بنية REPEAT .. UNTIL

الصيغة العامة لهذه البنية هي :

REPEAT Sts UNTIL e

حيث :

Sts : **مجموعة عبارات.**

e : **تعبير منطقي (شرط منطقي).**

ومعنى هذه البنية : نفذ مجموعة العبارات Sts **ثم اختبر الشرط** e **فإن لم يتحقق أعد تنفيذ مجموعة العبارات** Sts **ثم اختبر الشرط** e **، وهكذا استمر في تكرير تنفيذ العروة (العبارات** Sts**) طالما أن الشرط** e **غير متحقق ، إلى أن يتحقق هذا الشرط ، وعندها اخرج من العروة وانتقل إلى أول عبارة بعد بنية** REPEAT.

ملاحظات :

(أ) **بناء على هذا المعنى فإن مجموعة العبارات** Sts **تنفذ مرة واحدة على الأقل. وتكون هذه المرة هي المرة الوحيدة إذا تحقق الشرط** e **عند أول اختبار له.**

(ب) ويلاحظ أن الفارق بين عبارة WHILE وعبارة REPEAT أن الشرط في عبارة WHILE يختبر قبل تنفيذ عروتها ، بينما الشرط في عبارة REPEAT يختبر بعد تنفيذ عروتها ، ولذلك فمن الممكن ألا تنفذ عروة WHILE إطلاقا ، بينما يجب أن تنفذ عروة REPEAT مرة واحدة على الأقل.

(ج) من الممكن أن تكتب مجموعة العبارات Sts بين كلمتي BEGIN في البداية و END في النهاية ، ولكن هذا ليس ضروريا.

(د) حتى لا تصبح عروة REPEAT لانهائية (infinite loop) يجب أن تتغير قيم بعض أو كل متغيرات التعبير المنطقي e (بواسطة بعض عبارات المجموعة Sts) إلى أن يصبح التعبير e صادقا (متحققا) ، ونخرج عندئذ من العروة.

مثال ٤-١٠ :

أعد حل مثال ٤-٧ باستخدام عبارة REPEAT بدلا من عبارة WHILE (انظر أيضا مثال ٤-٢ للمقارنة مع الحل باستخدام عبارة FOR).

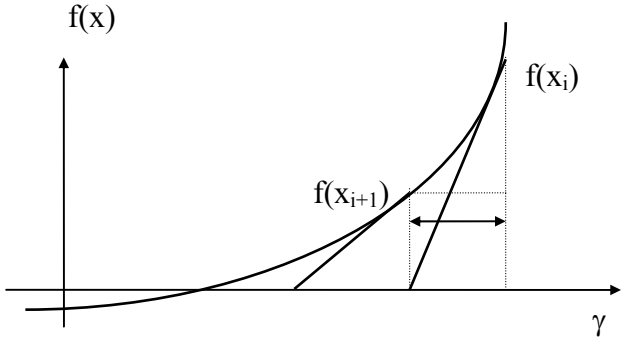
الحل :

```
PROGRAM SumAve2000 (INPUT , OUTPUT) ;
VAR
    counter : INTEGER ;
    item , sum , ave : real ;
BEGIN
    sum := 0.0 ;
    counter := 0 ;
    REPEAT
        READLN (item) ;
        sum := sum + item ;
        counter := counter + 1 ;
    UNTIL
        counter = 2000 ;
    ave := sum / 2000.0 ;
WRITELN ('sum = ', sum , 'ave = ', ave)
END {SumAve2000}
```

مثال ٤-١١ : (طريقة نيوتن رافسون لحل المعادلات)

فكرة الطريقة :

نفرض أن المطلوب حل المعادلة $f(x) = 0$ **أي إيجاد جذر هذه المعادلة.**



إذا فرضنا أن x_i **جذر تقريبي للمعادلة أي قيمة قريبة من الجذر الحقيقي** γ **ورسمنا مماسا للمنحنى** f **عند النقطة** $(x_i , f(x_i))$ **فتحت شروط خاصة ليس من اختصاص هذا الكتاب مناقشتها يقطع المماس المحور السيني عند نقطة** x_{i+1} **هي أقرب للجذر** γ **من** x_i **(أي أن** x_{i+1} **جذر تقريبي للمعادلة أفضل من الجذر التقريبي** (x_i)**.**

من الرسم نجد أن ميل المماس = $\dfrac{f(x_i)}{\Delta}$

ومن ناحية أخرى فإننا نعلم أن ميل المماس = $f'(x_i)$

وبذلك فإن $\dfrac{f(x_i)}{\Delta} = f'(x_i)$

أي أن $\Delta = \dfrac{f(x_i)}{f'(x_i)}$

ولكن $x_{i+1} = x_i - \Delta$

وبذلك فإن $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$

وتسمى هذه صيغة تكرارية (iterative formula) **حيث يمكن تكرار تطبيقها والحصول على قيم تقريبية أدق لجذر المعادلة ، أي أن القيم تتقارب نحو الجذر الحقيقي** γ **كلما كررنا تطبيق هذه الصيغة للقيم المتزايدة** $i = 1 , 2 , 3 , \ldots$ **هكذا :**

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

حيث x_0 هي أول قيمة تقريبية للجذر نبدأ بها.

وعادة نستمر في تطبيق هذه الصيغة التكرارية للحصول على القيم المتتالية

$$x_1 , x_2 , x_3 , \ldots$$

إلى أن يصل الفرق بين قيمتين متتاليتين إلى أقل من قيمة صغيرة ε تعتمد على درجة الدقة التي نريدها في حل المسألة

$$|x_{i+1} - x_i| \le \varepsilon$$

أو إلى أن يصل عدد مرات تكرار تطبيق الصيغة إلى عدد معين n_{max}.

المطلوب : كتابة برنامج لحل المعادلة

$$x - 0.2 \sin x - 0.5 = 0$$

بطريقة نيوتن رافسون (انظر خوارزمية الطريقة في المسألة رقم ١-١١ وخريطة سير العمليات في حل هذه المسألة).

الحل : نلاحظ في خوارزمية الطريقة أن الصيغة التكرارية مكتوبة في الصورة

$$x = x - \Delta$$

حيث x الموجودة في الطرف الأيسر هي قيمة x الجديدة (x_{i+1}) بينما x في الطرف الأيمن هي قيمة x القديمة (x_i).

وكذلك $\Delta = \frac{f(x)}{f'(x)}$ ، حيث

$$f(x) = x - 0.2 \sin x - 0.5$$
$$f'(x) = 1 - 0.2 \cos x$$

والآن يمكننا كتابة البرنامج الذي يقابل الخوارزمية أو خريطة سير العمليات لتطبيق الطريقة على المسألة المعطاة.

```
PROGRAM NewtonMethod (INPUT , OUTPUT) ;
{Finding the root of an equation by Newton - Rephson method}
VAR
    n , nmax : INTEGER ;
    x0 , eps , x , d : REAL ;
  {nmax : max allowed number of iterations
   n    : counter to count the number of iterations
   x0   : initial value as an approximate root
   eps  : required accuracy
   d    : difference bet. two successive roots}
BEGIN
    WRITELN ('Enter x0 , eps , nmax') ;
```

```
                    READLN (x0 , eps , nmax) ;
                                    n := 0 ;
                                    x := x0 ;
                                    REPEAT
        d := (x - 0.2*SIN (x) - 0.5) / (1.0 - 0.2*COS (x)) ;
                                    x := x - d ;
                                    n := n + 1 ;
                WRITELN ('X = ' , X , 'n = ' , n)
                                    UNTIL
            (ABS(d) <= eps) OR (n >= nmax)
                                {End REPEAT}
                            END {NewtonMethod}.
```

ملاحظة (١) :

عبارة FOR لا يجوز استخدامها هنا بدلا من عبارة REPEAT وذلك لأن عدد مرات التكرير غير معلوم سلفا. كذلك فإن عبارة REPEAT في هذا المثال أنسب من عبارة WHILE.

ملاحظة (٢) :

إذا فرضنا أن بيانات هذا البرنامج هي :

$x0 = 0.5$

$eps = 1.0E - 06 = 1 \times 10^{-6}$

فإن البرنامج يعطينا النتائج التالية :

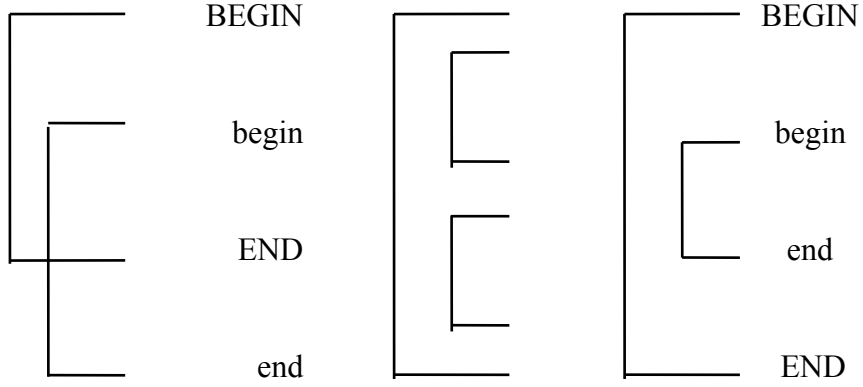
n = 1	x = 0.61629718
n = 2	x = 0.61546820
n = 3	x = 0.61546816

البنيات المتداخلة

وبنيات التكرار المتداخلة

Nested Structures
& Nested Repetition Structures

من الممكن عموما أن تكتب بعض البنيات بصورة متداخلة ، بمعنى أن توضع بنية ما داخل بنية أخرى ، بشرط أن تكون البنية الداخلية (inner structure) **بأكملها داخل البنية الخارجية** (outer structure) **، أي تحتوي البنية الخارجية احتواء كاملا على البنية الداخلية بدون أي تقاطع بين البنيتين ، كما يوضح ذلك الشكل التالي.**



تداخل صحيح **تداخل صحيح** **تداخل خاطئ (تقاطع)**

ومن الصور الخاصة للبنيات المتداخلة بنيات التكرار المتداخلة ، حيث نقوم بتكرار عملية معينة تشتمل إحدى خطواتها على تكرار عملية أخرى. وهكذا يشتمل البرنامج على تكرار لعملية مكررة كما يوضح ذلك المثال التالي (مثال ٤-١٢) ، وعموما يمكن أن تكون بنيتا التكرار المتداخلتان من النوع نفسه (الصيغة أو العبارة نفسها) أو من نوعين مختلفين (من العبارات FOR , WHILE , REPEAT**).**

مثال ٤-١٢ :

اكتب برنامجا يقوم بعمل جدول لحساب مضروب كل من الأعداد الصحيحة من ٢ إلى ١٢ (انظر مثال ٤-٦).

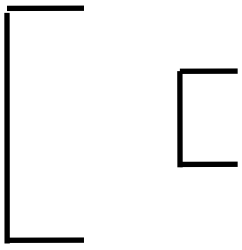
الحل :

في مثال ٤-٦ كان المطلوب حساب مضروب عدد واحد فقط k ، أما هنا فالمطلوب حساب المضروب لعدة أعداد (من ٢ إلى ١٢) ، ولذلك فكل ما سنعمله هنا هو إعطاء أمر – عن طريق عبارة FOR – بتكرار تنفيذ خطوات حل مثال ٤-٦ لجميع قيم k من ٢ إلى ١٢.

وبذلك يكون حل المثال الحالي كما يلي :

```
PROGRAM Factorial2To12 (INPUT , OUTPUT) ;
VAR k, fact , i : INTEGER ;
BEGIN
FOR k := 2 TO 12 DO
BEGIN
fact := 1 ;
FOR i := 2 To k DO
fact := fact * i
{End inner FOR} ;
WRITELN ('k = ', k, 'factorial k = ' , fact)
END
{End outer FOR}
END {Factorial2to12}
```

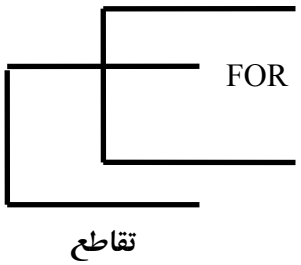
ملاحظة :



```
FOR k := 2 TO 12 DO

FOR i := 2 TO K DO

{End inner FOR}

{End outer FOR}
```

عندنا في هذا الحل عروتان من عرى FOR ، عروة خارجية وأخرى داخلية كما هو مبين في الرسم التوضيحي. العروة الخارجية فائدتها إعطاء قيم مختلفة

للمتغيرk ، والعروة الداخلية تستخدم لحساب المضروب لكل قيمة من قيم k التي تحددها العروة الخارجية. وكما ذكرنا سابقا فهناك قاعدة عامة في كتابة العرى وهي أنه يجب ألا يحدث تقاطع (أو تداخل) بينها بمعنى أنه إذا بدأت عروة قبل أن تنتهي عروة سابقة فيجب أن تنتهي العروة الجديدة اللاحقة قبل نهاية العروة السابقة (حتى لا يحدث تقاطع).



فمثلا الشكل المجاور يمثل استعمالا خاطئا لعرى FOR حيث بدأنا في FOR (جديدة) وانتهت بعد نهاية FOR سابقة أي هناك تقاطع كما هو مشار إليه.

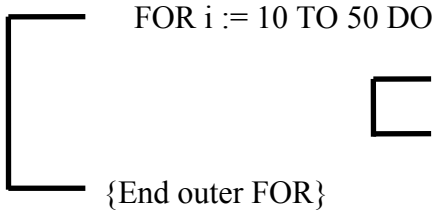
حل آخر مختصر :

يمكن حل هذه المسألة باستخدام عروة FOR واحدة فقط كما يلي بدلا من عروتين :

```
PROGRAM Factorial2To12 (INPUT , OUTPUT) ;
VAR k, fact : INTEGER ;
BEGIN
    fact := 1 ;
FOR k := 2 TO 12 DO
    BEGIN
        fact := fact * k ;
WRITELN ('k = ', k , 'factorial k = ', fact)
    END
    {End FOR}
END.
```

ملاحظة عامة بالنسبة لبنى FOR المتداخلة :

عند كتابة بنية (عروة) FOR داخل بنية أخرى من بِنَى (بنيات) FOR يجب مراعاة ألا يكون متغير التحكم (i مثلا) في العروة الخارجية هو نفسه متغير التحكم في العروة الداخلية ، فمن الخطأ أن نكتب مثلا



وذلك لأنه – كما ذكر سابقا ضمن قواعد استخدام عبارة FOR **– لا يجوز أن تغيِّر أي عبارة في عروة** FOR **قيمة متغير التحكم في العروة ، وفي المثال المذكور فإن عبارة** FOR **الداخلية تغير من قيمة متغير التحكم** i **في العروة الخارجية. وفي مثل هذه الحالة يجب استعمال متغير آخر كأن نكتب مثلا:**

```
FOR i := 10 TO 50 DO
    FOR j := 2 To n DO
```

تمرينات رقم ٤

أولا : تدريبات

٤-١ أوجد قيمة k بعد تنفيذ مجموعة العبارات التالية :

```
k := 3 ;
m := 2 ;
FOR j := 3 TO 5 DO
    BEGIN
t := m * j - 3 ;
k := k + t
    END ;
```

٤-٢ كم عدد القيم التي تضاف إلى sum في كل من قطع البرامج التالية ؟

```
var
    sum, xdata : real ;
    cnt : integer ;
begin
    sum := 0.0 ;
    cnt := 15 ;
    while cnt >= 0 do
        begin
        readln (xdata) ;
    sum := sum + xdata ;
        cnt := cnt - 2
        end
end
```

(i)

```
var
    sum, xdata : real ;
    cnt : integer ;
begin
    sum := 0.0 ;
    cnt := 0 ;
    while cnt < 15 do
        begin
        readln (xdata) ;
    sum := sum + xdata ;
        cnt := cnt + 2
        end
end
```

(ii)

```
var
    sum , xdata : real ;
    cnt : integer ;
begin
    sum : 0.0 ;
    cnt := 17 ;
        repeat
    readln (xdata) ;
sum := sum + xdata ;
    cnt := cnt - 3
    until cnt < 0
```

```
var
    sum , xdata : real ;
    cnt : integer ;
begin
    sum := 0.0 ;
    cnt := -1 ;
        repeat
    readln (xdata) ;
sum := sum + xdata ;
    cnt := cnt - 3
    until cnt < 0
```

```
end
```
(iii)

```
end
```
(iv)

٤-٣ ما هي قيمة counter بعد تنفيذ كل من قطع البرامج التالية :

```
var
    counter, i : integer ;
begin
    counter := 0 ;
    for i := -8 to 6 do
    counter := counter + 1
end
```
(i)

```
var
    counter, i : integer ;
begin
    counter := 0 ;
    for i := 6 downto -5 do
    counter := counter + 1
end
```
(ii)

```
var
    counter, i : integer ;
begin
    counter := 10 ;
    for i := 5 downto -5 do
    counter := counter - 1 ;
    for i := 4 to 16 do
    counter := counter + 1
end
```
(iii)

٤-٤ ما هي قيمة x1 بعد تنفيذ قطعة البرنامج التالية ؟

```
var
    i : integer ;
    x1 : real ;
begin
    x1 := 32.0 ;
    for  i := -2 to 2 do
        x1 := x1/2.0
end
```

٤-٥ ما هي قيمة counter بعد تنفيذ كل من عرى for المتداخلة التالية ؟

```
var
    i,j , counter : integer ;
begin
    counter := 0 ;
    for i := 40 downto 0 do
    for j := 5 to 10 do
    counter := counter + 1
end
```

```
var
    i,j , counter : integer ;
begin
    counter := 0 ;
    for i := 40 downto 0 do
    begin
    for j := 0 to 4 do
    counter := counter + 1;
```

```
counter := counter + 1
end
end
```

(i) (ii)

```
var
i , j , counter : integer ;
begin
counter := 0 ;
for i := 40 downto -1 do
begin
for j := -1 to 9 do
counter := counter + 1;
counter := counter - 1
end
end
```

(iii)

٤-٦ ما هي قيمة count بعد تنفيذ كل من قطع البرامج (المنفصلة) التالية؟

```
count := 0 ;
for i := 5 to 25 do
count := count - 2
```

(ii)

```
count := 0 ;
for i := -5 to 5 do
count := count + 1
```

(i)

```
i := 0 ;
j := 50 ;
count := 0 ;
while i <= 100 do
begin
count := count -20 ;
i := i + 10 ;
repeat
j := j - 10 ;
count := count + 20
until j <= 40
end
```

(iii)

٤-٧ ما هي قيمة كل من count1 , count2 بعد تنفيذ مجموعة العبارات التالية ؟

```
var
count1, count2, k, j : integer;
begin
count1 := 0 ;
count2 := 0 ;
for k := 5 down to -5 do
begin
if odd (k) then
count1 := count1 + k ;
for j := -2 to 4 do
count2 := count2 - 2
end
```

٤-٨ فيما يلي مجموعة من قطع برامج. إن كانت القطعة صحيحة فما مخرجاتها ؟ وإن كانت خاطئة فما سبب الخطأ ؟

(أ)

```
lower := 1 ;
upper := 3 ;
for counter := lower to upper do
begin
lower := 0 ;
upper := 2 ;
writeln ('Assalamo A''laikom')
end ;
```

(ب)

```
lower := 1 ;
upper := 3 ;
for counter := lower to upper do
begin
writeln (counter) ;
counter := counter + 1
end ;
```

(ج)

```
for i := 1 to 3 do
writeln ('Your Mother') ;
writeln ('Your Father') ;
```

```
writeln ('****') ;
```

٤-٩ كتب طالب مجموعة العبارات المبينة لحساب وطباعة مضروب العدد ٥ والذي يساوي ١ × ٢ × ٣ × ٤ × ٥.

```
Product := 1 ;
Counter := 2 ;
While Counter <= 5 do ;
Product := Product * Counter ;
Counter := Counter + 1 ;
Writeln (Product) ;
```

(أ) ما هي مخرجات هذه المجموعة ؟ ولماذا ؟

(ب) صحح أي أخطاء - إن وجدت - في هذه العبارات لتعطي المطلوب.

(i) أعد كتابة العبارات باستخدام عبارة Repeat لتعطي المطلوب.

٤-١٠ (أ) نفرض أن مدخلات البرنامج التالي هي N = 15 . تتبع تنفيذ البرنامج واكتب نتائج تنفيذ عباراته إلى أن يتوقف. ما هي مخرجات البرنامج ؟ وما وظيفته ؟

```
Program exam (input , output) ;
var
i , N , sum : integer ;
Begin
readln (n) ;
sum := 0 ;
for i := 1 to n do
if ( i mod 2) <> 0 then
sum := sum + i ;
writeln (sum : 5)
readln
end.
```

(ب) اكتب بالضبط مخرجات قطعة البرنامج التالية :

```
for i := 1 to 3 do
begin
for j := 1 to 4 do
write ('*') ;
writeln
end ;
```

(ج) ما هي مخرجات قطعة البرنامج المبينة إذا كانت قمية MAX تساوي 3؟

```
Count := 0 ;
while Count < MAX do
    begin
writeln ('Outer', Count) ;
 while Count < MAX do
     begin
writeln ('Inner' , Count) ;
 Count := Count + 1
     end
end ;
```

٤-١١ أ) اكتب بالضبط مخرجات (exact output) قطعة البرنامج التالية :

```
I := 30 ;
REPEAT
I := (I DIV 5) - 1 ;
I := I * 5 ;
WRITELN (I) ;
UNTIL (I <= 1) ;
```

(ب) أعد كتابة قطعة البرنامج السابقة باستخدام عبارة WHILE بدلا من عبارة REPEAT.

(ج) اكتب قطعة برنامج تستخدم عبارة FOR واحدة تعطي نفس مخرجات قطعة البرنامج السابقة في (أ).

٤-١٢ ما هي مخرجات قطعة البرنامج التالية بالضبط ؟

```
Limit := 7 ;
For i := 1 to Limit do
    begin
Write (i + Limit : 3) ;
Limit := 5
    end ;
writeln ;
```

٤-١٣ (أ) ما نتيجة تنفيذ العبارة التالية ؟

```
For i := 1 to 8 do
writeln (I * 10) ;
```

(ب) أعد كتابة العبارة باستخدام

(i) عبارة Repeat (ii) عبارة While

١٤-٤ تقوم عبارة Repeat التالية بقراءة أزواج متتالية من القيم N1 , N2 وطباعة كل زوج بترتيب معاكس ، إلى أن تصادف قيمة سالبة (N1 أو N2). صحح أي أخطاء في العبارة ، ثم اكتب قطعة مكافئة باستخدام عبارة While.

```
Repeat
    read (N1 , N2)
    writeln (N2 , N1)
untill
    N2 < 0 ;    or    N1 < 0
```

١٥-٤ المطلوب تتبع تنفيذ جميع العبارات في كل من البرنامجين التاليين ، وكتابة نتائج تنفيذها ، وكذلك المخرجات.

(أ)

```
program Rev (output) ;
    var n : integer ;
Begin n := 8453 ;
    repeat
        write (n mod 10 : 1) ;
        n := n div 10
    until
        n = 0
End.
```

```
Program Nestloop (output) ;
    var   i , j : integer ;
        begin
    writeln ('I': 12 , 'J' : 5) ; {print heading}
        for i := 1 to 3 do
        begin {outer loop}
    writeln ('outer' : 5 , i : 7) ;
        for j := 1 to i do
    writeln ('inner' : 7 , i :5 , j : 5)
        end {outer loop}
    end. {Nestloop}
```

١٦-٤ تتبع يدويا (Hand trace) البرنامج التالي واكتب نتائج تنفيذ عباراته. وبيِّن مخرجاته ، إذا أُعطيت البيانات الآتية :

4,2,8,4 1,4,2,1 9,3,3,1 -22,10,8,2

```
program SLOPE (INPUT , OUTPUT) ;
                                const
                                SENTINEL = 0 ;
                                var
                                SLOPE, Y2, Y1, X2 , X1 : REAL ;
                                begin
                                WRITELN ('Enter four real numbers') ;
                                READLN (Y2, Y1, X2 , X1) ;
                                SLOPE := (Y2 - Y1) / (X2 - X1) ;
                                while SLOPE <> SENTINEL do
                                begin
                                WRITELN ('Slope is' , SLOPE : 5 : 3) ;
                                WRITELN ;
                                WRITELN ('Enter four real numbers') ;
                                READLN (Y2 , Y1 , X2 , X1);
                                SLOPE := (Y2 - Y1) / (X2 - X1)
                                end {while}
                                end.
```

٤-١٧ اكتب بالضبط مخرجات العبارة التالية :

```
For i := 3 To 5 Do
case i of
1 : writeln ('In the Name of Allah') ;
2 : writeln ('Praise be to Allah') ;
3 : writeln ('O Mojahidoon of Afghanistan !') ;
4 : writeln ('Congratulations !') ;
5 : writeln ('May Allah bless you.') ;
6 : writeln ('Victory or Martyrdom')
end ;
```

٤-١٨ (أ) تتبع تنفيذ قطعة البرنامج التالية وذلك باستخدام مجموعة البيانات المعطاة أسفل القطعة. اكتب نتائج تنفيذ جميع عبارات الإسناد. ما هي القيمة النهائية لكل من x , c ؟

(ب) أعد كتابة القطعة باستخدام (i) عبارة Repeat (ii) عبارة For

```
c := 0 ;
i := 1 ;
while i <= 5 do
begin
read (x) ;
```

```
if (x < 0) or (x > 10) then
        c := c + 1 ;
            i := i + 1
          end {while}
input : 15   10   -10   5   20
```

٤-١٩ نفرض أن البيانات المدخلة (input) للبرنامج التالي هي :

n = 28693

أ) **اكتب نتيجة تنفيذ جميع عبارات الإسناد الحسابية في البرنامج إلى أن يتوقف.**

ب) **ما هي مخرجات (output) هذا البرنامج ؟**

ج) **ما هي وظيفة هذا البرنامج ؟ أي ماذا يحسب ؟**

د) **ما هي مخرجات البرنامج إذا كانت مدخلاته 20000 بدلا من 28693.**

```
PROGRAM Test (INPUT , OUTPUT) ;
VAR n , d , s : INTEGER ;
BEGIN
WRITE ('Enter an integer number') ;
READLN (n) ;
S := 0 ;
WHILE n <> 0 DO
BEGIN
d := n MOD 10 ;
WRITELN (d) ;
s := s + d ;
n := (n - d) DIV 10
END ;
WRITELN (s)
END.
```

٤-٢٠ كتب طالب مجموعة العبارات التالية لقراءة وَعَدِّ حروف متتالية إلى أن نصادف فراغا (blank). فهل تؤدي مجموعة العبارات هذه الوظيفة أم لا ؟ وإن كانت الإجابة : لا ، فما الخطأ في العبارات ؟

```
count := 0 ;
read (ch) ;
if (ch = ' ') then
repeat
read (ch) ;
```

```
        count := count + 1
                until
        (ch = ' ') ;
```

٤-٢١ نفرض أن لدينا التعريف :

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

ما نتيجة تنفيذ قطعة البرنامج المبينة ؟

```
For i := 1 To 4 do
        begin
write (ch) ;
    writeln
        end ;
```

٤-٢٢ تتبع تنفيذ مجموعة العبارات التالية من أحد البرامج للمدخلات المعطاة ، وأوجد مخرجات البرنامج. ما وظيفة هذا البرنامج ؟ أي ماذا تحسب هذه المجموعة من العبارات ؟

```
n := 6 ;
count := 0 ;
for i := 1 to n do
    begin
read (item ) ;
if (15 <= item) and (item <= 25) then
        count := count + 1
end : writeln (count) ;
```

المدخلات : input : 20 10 15 -30 25 33 18 40 22

٤-٢٣ (أ) ما مخرجات العبارة التالية ؟

```
for count := 10 downto 8 do
    writeln (count) ;
```

(ب) ما مخرجات القطعة التالية ؟

```
for i := 1 to 3 do ;
writeln ('Allah is the Greatest') ;
```

٤-٢٤ كتب طالب مجموعة العبارات التالية لجمع الأعداد الصحيحة من واحد إلى مائة ، أي لحساب sum = 1+2+3+….+99+100

```
sum := 0 ;
number := 1 ;
    repeat
sum := sum + number ;
```

```
        number := number + 1
                    until
            number >= 100 ;
        writeln ('sum = ' , sum) ;
```

أ) **هل هذه العبارات تعطي المجموع المطلوب ؟ ولماذا ؟**

ب) **أعد كتابة العبارات باستخدام** (i) **عبارة** while (ii) **عبارة** for.

٤-٢٥ تتبع تنفيذ البرنامج التالي ، واكتب مخرجاته.

```
                Program Tri (output) ;
        const   blank = ' ' ; star = '*' ;
                var i , j , k : integer ;
                        Begin
                    for i := 1 to 4 do
                        begin
            for j := 4 - i downto 1 do
                write (blank) ;
            for k := 1 to 2 * i - 1 do
                write (star) ;
                    writeln
                        end
                        End.
```

٤-٢٦ أ) تتبع تنفيذ خطوات البرنامج التالي مع البيانات المعطاة إلى أن يتوقف.

اكتب نتائج تنفيذ جميع عبارات الإسناد وكذلك المخرجات.

```
                    Program Test ;
                    const n = 8 ;
                        var
            i, sum , value : integer ;
                flag : boolean ;
                    Begin (* Test *)
        sum := 0 ; i := 1 ; flag := false ;
        while (i <= n) and not flag do
                        begin
                read (value) ;
            if value > 0 then
        sum := sum + value
            else if value = 0 then
            flag := true ;
                    i := i + 1
                        end ;
```

```
writeln (sum , value)
End. (* Test *)
```

البيانات : 9 8 5 0 -4 7 -3 6 5

(ب) ما وظيفة هذا البرنامج (بعبارة مختصرة) ؟

(ج) ما هي مخرجات البرنامج إذا كانت مدخلاته (البيانات) هي :

(i) 10 9 8 7 6 5 4 3 2 1

(ii) 8 -9 7 0 4 1 -6 3

٤-٢٧ سئل طالب أن يكتب برنامجا لحساب عدد الأرقام الزوجية في متتالية أرقام مدخلة ذات عدد معلوم من الأرقام ، فكتب البرنامج التالي. وجميع عبارات البرنامج صحيحة من حيث قواعد لغة الباسكال [أي أن البرنامج ليس به أخطاء تركيبية (syntax errors)] ، ولكن البرنامج لا يعطي نتائج صحيحة لوجود أخطاء منطقية (logical errors / semantics bugs) .

(أ) اكتشف أي أخطاء منطقية في البرنامج وأعد كتابته بعد تصحيح ما به من أخطاء.

(ب) اختبر صحة البرنامج (المصحح) بتتبع عباراته وكتابة مخرجاته عندما تكون مدخلاته هي :

6
8
7
3
6
4
5

```
program EvenCount (input , output) ;
{Find the number of even digits in a sequene of input numbers}
var
    Digit, Counter, EvenNumber, Limit : Integer;
begin
    writeln ('How many numbers should I look at? ') ;
    readln (Limit) ;
    writeln ('Enter a digit') ;
    readln (Digit) ;
    For Counter := 1 to Limit Do
        begin
```

```
                EvenNumber := 0 ;
              If (Digit div 2 = 0) then
EvenNumber := EvenNumber + 1 ;
              writeln ('Enter a digit') ;
                  readln (Digit)
                          end ;
writeln ('The number of even numbers was',EvenNumber) ;
                                  end.
```

٤-٢٨ البرنامج التالي يوجد أكبر عنصر في مجموعة مكونة من n عدد صحيح.

أ) تتبع خطوات تنفيذ البرنامج في الحالة n = 10 مع مجموعة البيانات التالية : 40 -25 70 -15 0 -20 45 -33 0 25

ب) عدل البرنامج بحيث يوجد أصغر عنصر في المجموعة وكذلك المتوسط الحسابي بالإضافة إلى أكبر عنصر.

```
                program example (input , output) ;
{this program finds the largest element in a set of n integer elements}
                                              var
                        : integer ;      n , i
                        : integer ;   x , large
                                            begin
                write ('enter the number of elements n') ;
                                  readln (n) ;
          writeln ('enter the elements to find their largest') ;
                                  large := - maxint ;
{we read the elements one by one and compare each with the
                        largest element found so far}
                                        i := 1 ;
                                  while i <= n do
                                      begin
                                read (x) ;
                              if x > large then
                             large := x ;
                                    i := i + 1
                                  end ; {while}
                                        readln ;
          writeln ('the largest element = ' , large) ;
          write ('press the enter key to continue') ;
                                        readln
                                  end. {example}
```

ثانيا : برامج

٤-٢٩ أ) اكتب برنامجا يقرأ قيم ١٠ أعداد صحيحة ويحدد لكل عدد إن كان فرديا أو زوجيا ، وذلك باستخدام بنية if .. then .. else **والمؤثر** mod **(ودون استخدام الدالة القياسية** odd**) ، ويطبع مخرجاته في الصورة التالية :**

المدخلات	المخرجات
Input	Output
69	69 IS ODD.
54	54 IS EVEN.
35	35 IS ODD.
⋮	⋮
⋮	⋮
⋮	⋮
62	62 IS EVEN.

ب) أعد حل الجزء أ) من السؤال ، وذلك باستخدام بنية case **والدالة القياسية** odd**.**

٤-٣٠ باستخدام عرى for **المتداخلة اكتب برنامجا يقرأ ٢٠ زوجا من الأعداد الصحيحة** a , b **، ويحسب لكل زوج قيمة** a **مرفوعة لأس** b **(أي يحسب** a^b**) ، ويطبع النتيجة في صورة مماثلة لما يلي :**

المدخلات	المخرجات
Input	Output
1 5	1 TO THE POWER 5 = 1
64 3	64 TO THE POWER 3 = 262144
12 4	12 TO THE POWER 4 = 20736
	⋮
	⋮

٤-٣١ اكتب برنامجا يقرأ ٢٠ عددا صحيحا موجبا num **بحيث أن أي عدد منها هو إحدى قوى ٢ (أي ١ ، ٢ ، ٤ ، ٨ ، ١٦ ، ٣٢ ، ...) ، ثم يعين لكل عدد** num **من هذه الأعداد قيمة العدد الصحيح** n **(القوة / الأس) الذي لو رفعت إليه ٢ لأعطت العدد المدخل** num **(أي أن** 2^n = num**) ، ويطبع النتائج في صورة مماثلة لما يلي :**

[ملاحظة : يمكن الحصول على قيمة n بالطريقة التالية :
إذا قسمنا العدد num على ٢ ، ثم قسمنا الناتج على ٢ ، وكررنا هذه القسمة عددا من المرات إلى أن نحصل على ناتج يساوي ١ ، فإن هذا العدد الكلي من المرات يساوي n].

المدخلات	المخرجات
Input	Output
1	2 TO THE POWER 0 = 1
2	2 TO THE POWER 1 = 2
4	2 TO THE POWER 2 = 4
1024	2 TO THE POWER 10 = 1024
	⋮
	⋮

٤-٣٢ سقط جسم من السكون من ارتفاع ٦٠٠٠٠ متر. اكتب برنامجا يعطي سرعة الجسم كل ١٠ ثواني وإلى أن يرتطم بالأرض ، علما بأن هذه السرعة تعطى بالعلاقة $v = g\,t$

حيث v : السرعة بالمتر / ثانية

g : عجلة الجاذبية وتساوي ٩,٨١ متر / ثانية ^{٢}

t : الزمن بالثانية

إرشاد :

يمكن استخدام العلاقة $t = \sqrt{\frac{2s}{g}}$ حيث s هي المسافة التي تحركها الجسم ابتداء من لحظة سقوطه إلى الزمن t ، وذلك لحساب الزمن limit الذي يستغرقه الجسم للوصول إلى الأرض ، ثم استخدام هذه القيمة للخروج من العروة التي تحسب السرعات.

٤-٣٣ 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

$$x = 1.0 , 1.1 , 1.2 , 1.3 , \dots , 2.0$$

وهكذا بالنسبة لباقي المتغيرات.

اكتب برنامجا يستخدم ثلاث عرى من عرى FOR 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).

٣٤-٤ 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 مستخدما عبارة FOR 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.

٣٥-٤ اكتب برنامجا لقراءة قيمة عدد صحيح موجب n ثم حساب مجموع المتسلسلة :

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots \pm \frac{1}{n}$$

وذلك باستخدام بنية FOR.

٣٦-٤ أعد حل المسألة السابقة (٤-٣٥) باستخدام

(أ) بنية WHILE (ب) بنية REPEAT.

٣٧-٤ 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.

٣٨-٤ 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)

٣٩-٤ 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 > 0 , DELTA > 0 , XIN < XFI)

(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.

٤٠-٤ اكتب برنامجا يترجم خريطة سير العمليات في المثال ١-٦ إلى لغة الباسكال ليقوم بقراءة مجموعة من درجات الحرارة ϕ بالتقدير الفهرنهيتي ويحولها إلى الدرجات المقابلة θ بالتقدير المئوي تبعا للعلاقة $\theta = \frac{5}{9}(\phi - 32)$.

٤١-٤ 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.

٤٢-٤ اكتب برنامجا لحساب مجموعة المتسلسلة التالية (انظر المسألة ١-٧-ii)

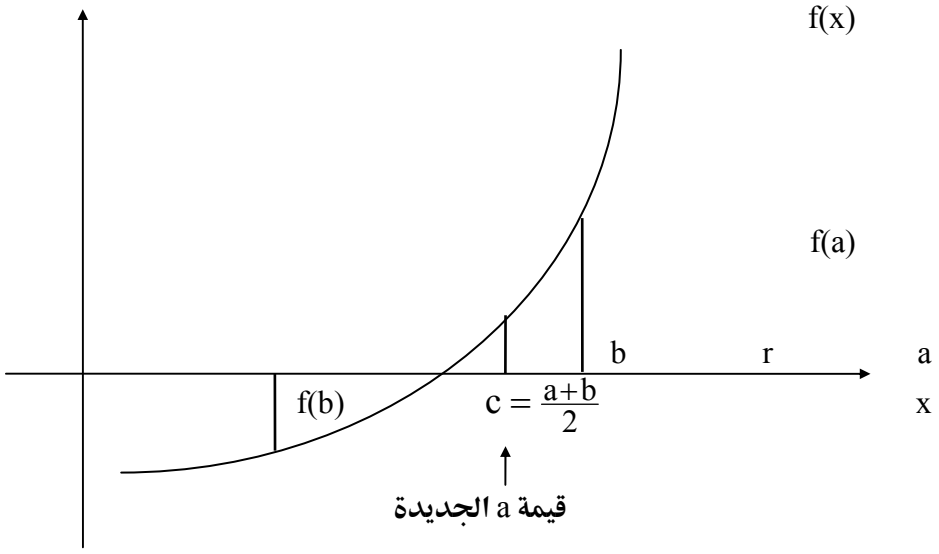
$$s = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \cdots + \frac{99}{100}$$

٤٣-٤ اكتب برنامجا لإيجاد جذر المعادلة $x = 1.4 \cos x$ بتطبيق طريقة نيوتن رافسون (انظر مثال ٤-١١). أوقف تطبيق الصيغة التكرارية عندما يصل

الفرق بين قيمتين متتاليتين إلى أقل من 10^{-5} ، واجعل البرنامج يطبع كل هذه القيم المتتالية إلى أن يصل إلى الجذر حسب الدقة المطلوبة.

٤٤-٤ 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 c , a على الجانب نفسه من الجذر الحقيقي) فاعط قيمة c للمتغير a ، أي أن قيمة a الجديدة هي $a = c$ ، وإلا فاعط قيمة c للمتغير b أي أن قيمة b

الجديدة تصبح c = b.. 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

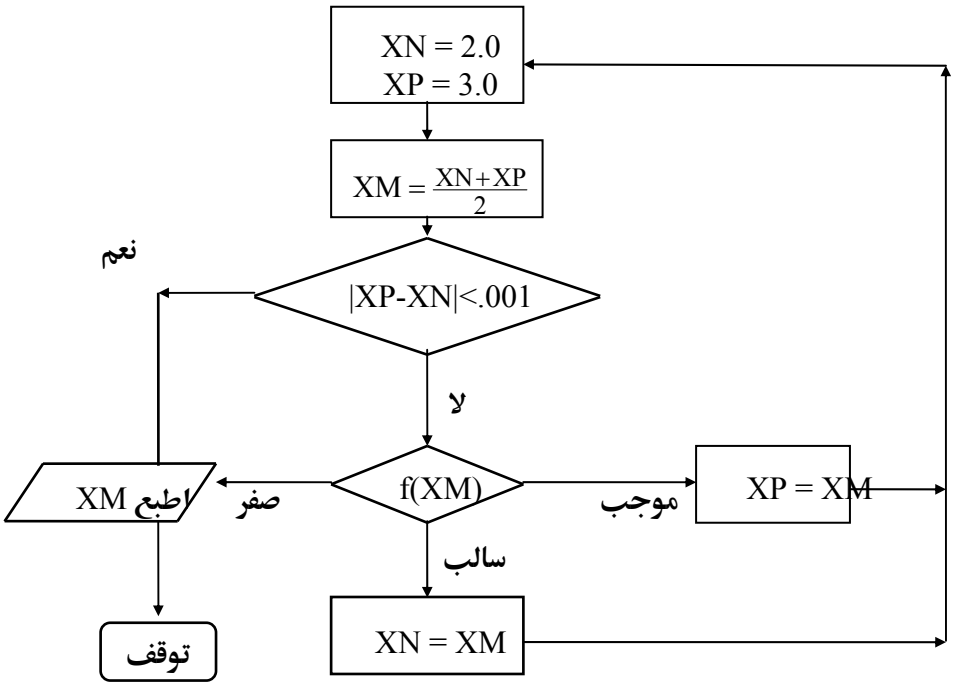
(إرشاد : $f(0) < 0$, $f(\pi) > 0$, $\pi = 3.14159$)

٤-٤٥ 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 :



٤-٤٦ 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.

٤٧-٤ يتجه المسلمون من جميع أنحاء العالم لأداء فريضة الحج حيث يقفون جميعا على عرفات ويؤدون مناسكهم بلا أدنى تفرقة بينهم بسبب اختلاف قومياتهم أو جنسياتهم أو ألوانهم أو ألسنتهم أو مراكزهم وإنما الجميع إخوة في الإسلام.

نفرض أن كل حاج قد أعدت له بطاقة عليها قيمة عدد صحيح K يشير إلى كيفية وصوله ، بحيث أن الشفرة المستخدمة هي أنK = ١ أو ٢ أو ٣ للدلالة على أنه وصل برا أو بحرا أو جوا على الترتيب. فإذا أُعطيت مجموعة بطاقات كل الحجاج فاكتب برنامجا يقرأ هذه البيانات ويحسب :

(أ) أعداد الحجاج NA , NS , NL الذين وصلوا برا وبحرا وجوا على الترتيب.

(ب) العدد الكلي للحجاج N.

(ج) النسب المئوية PA , PS , PL للحجاج الذين وصلوا برا وبحرا وجوا على الترتيب.

٤-٤٨ 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 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.

٤-٤٩ 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 $n = 9$.

٤-٥٠ 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.

٤-٥١ تحتوي كل بطاقة من مجموعة بطاقات على ثلاثة أعداد موجبة A,B,C. أضيف في نهاية المجموعة بطاقة عليها ثلاثة أعداد سالبة. اكتب برنامجا يحدد ما إذا أمكن للقيم A,B,C على كل بطاقة أن تكوِّن أطوال أضلاع مثلث أم لا. إذا كانت الإجابة نعم فاحسب طول محيط المثلث (P = A + B + C) وإذا كانت الإجابة لا فاطبع الرسالة 'NOT A TRIANGLE' ، وفي كلا الحالتين اطبع قيم A,B,C.

إرشاد : يمكن لأطوال الأضلاع A,B,C أن تكوِّن مثلثا إذا كان طول كل ضلع أصغر من مجموع طولي الضلعين الآخرين ، أي إذا كان :

$A < B + C$, $B < A + C$ & $C < A + B$

٤-٥٢ اكتب برنامجا مع رسم خريطة سير عملياته لحساب :

(أ) قوى العدد ٢ :

$y_i = 2^i$, $i = 1, 2, 3, \ldots$

(ب) مقلوبات هذه القوى ، أي الكسور :

$$z_i = \frac{1}{2^i} \qquad ; i = 1,2,3,.....$$

(ج) المجموعات المتراكمة الجزئية لهذه الكسور :

$$s_i = \sum_{k=1}^{i} \frac{1}{2^k}$$

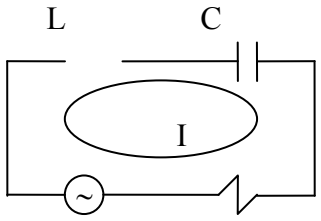
بحيث يطبع البرنامج النتائج في أربعة أعمدة كما يلي :

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 واحد من المليون $\left(\frac{1}{2^i} \le 0.000\,001\right)$

٤-٥٣ 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) $f_{max} , f_{min} , \Delta$

E R

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.

٤-٥٤ اكتب برنامجا يقرأ قيم عناصر مجموعة أعداد صحيحة عددها n ، ويوجد عدد العناصر الموجبة وحاصل ضربها.

٤-٥٥ اكتب برنامجا لإيجاد كل من عدد العناصر الموجبة ، وعدد العناصر السالبة ، وعدد العناصر الصفرية في مجموعة مدخلات تتكون من n عدد صحيح.

٤-٥٦ 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.

٤-٥٧ 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 : $sum = 9 + 15 = 24$

استخدم عبارة FOR.

٤-٥٨ نفرض أن كلا من i , j , k عدد صحيح محصور بين ١ و ٩ $[1 \le i , j , k \le 9]$. اكتب برنامجا يستخدم عرى FOR ليعطي جميع الثلاثيات (triplets) (i , j , k) التي تحقق الشرط $j = \frac{i+k}{2}$.

[من أمثلة هذه الثلاثيات : $(2,5,8) , (4,4,4) , (1,3,5)$]

٤-٥٩ 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.

٤-٦٠ 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.

٤-٦١ العلاقة بين الضغط والحجم ودرجة الحرارة لغاز مثالي يمكن تمثيلها بالعلاقة : $PV = KT$ ، وبصورة أدق يمكن تمثيل هذه العلاقة بالمعادلة :

$$PV = 0.0299\ (T + 460)$$

حيث :

P : هو الضغط (ووحدته : لكل بوصة مربعة psi)

V : هو الحجم (ووحدته : قدم^3 cubic feet)

t : هي درجة الحرارة (بالتقدير الفهرنهيتي ^oF)

والمطلوب حساب V لتوافقات مختلفة من P , T كما يلي :

اكتب برنامجا لطباعة جدول لقيم V للضغوط من ١٥٠ إلى ٢٠٠ وبزيادة ثابتة تساوي ٥ ، ولدرجات الحرارة من ١٠٠ إلى ٥٠٠ وبزيادة ثابتة تساوي ٥٠ ، مع كتابة عناوين مناسبة للجدول.

٤-٦٢ 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 (باستخدام عبارة for).

ب) في بداية كل عام من الأعوام التالية إلى أن تصل قيمة السيارة إلى أقل من ٧٠٠ دينار (باستخدام عبارة repeat).

٤-٦٣ 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.

(ب) عدد ناقص (deficient) : إذا كان العدد أقل من مجموع قواسمه SUM.

(ج) 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 بلغة الباسكال 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.

٦٤-٤ 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

٦٥-٤ يعتمد معدل تحلل أي عنصر من النظائر المشعة (radioactive isotopes) على عمره النصفي H (half - life) وهو الفترة الزمنية اللازمة للنظير ليتحلل إلى نصف كتلته الأصلية. مثلا H بالنسبة للنظير "سترونتيم ٩٠" (Strontium 90)(Sr^{90}) تساوي ٢٨ سنة.

اكتب برنامجا لحساب وطباعة جدول يعطي كمية هذا النظير المتبقية بعد كل سنة لمدة خمسين سنة بفرض أن الكمية الابتدائية تساوي ٥٠ جراما ، ويمكن حساب كمية النظير المتبقية باستخدام العلاقة :

$$r = a.c^{year/H}$$

حيث

a : الكمية الابتدائية = ٥٠ جراما

c : ثابت يساوي $e^{-.693}$

year : عدد السنوات التي انقضت

H : العمر النصفي للنظير بالسنوات

٤-٦٦ "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

$$(1\ldots10)\ ,\ (1\ldots11)\ ,\ (1\ldots12)\ ,\ \ldots\ ,\ (1\ldots20)$$

وذلك بأن يحسب ويطبع قيمتي طرفي العلاقة (*) ويعطي رسالة تفيد إن كانت هاتان القيمتان متساويتين ، وذلك لكل من القيم $n = 10,11,\ldots,20$.

٤-٦٧ نفرض أن شخصا عنده مدخرات تساوي S دينارا قد حال عليها الحول ، وأن سعر جرام الذهب يساوي P دينارا.

اكتب برنامجا يقرأ قيمة كل من P , S ويحسب ويطبع ما يلي :

(أ)) زكاة هذه المدخرات ، وقيمة المدخرات المتبقية بعد إخراج الزكاة.

i) زكاة المدخرات المتبقية عندما يحول عليها حول آخر ، والمدخرات المتبقية (بعد حولين) بعد إخراج هذه الزكاة. افرض أن سعر جرام الذهب لم يتغير.

ii) 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.

٤-٦٨ 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).

٤-٦٩ 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

٤-٧٠ 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.

٤-٧١ 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-o-data ومجموع مربعات القيم (sum-of-squares) في العروة (loop) الرئيسية ، وبعد انتهاء العروة استخدم المعادلة:

$$\text{sum-of-squares} = \text{variance} - (\text{sum-of-data})^2/N$$

(ج) الانحراف القياسي S (standard deviation) والذي هو عبارة عن قياس لكيفية انحراف البيانات عن المتوسط ، باستخدام المعادلة :

$$S^2 = \text{variance} / (N - 1)$$

٤-٧٢ 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 :

$$\text{cross - sectional area} = \text{base} \times \text{height}$$
$$\text{moment of inertia} = (\text{base} \times \text{height}^3) / 12$$
$$\text{section modulus} = (\text{base} \times \text{height}^2) / 2$$

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.

٤-٧٣ 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).

٤-٧٤ أ) اكتب برنامجا لقراءة مجموعة من درجات الامتحان التي تتراوح قيمها من ١ إلى ١٠٠ ، ثم حساب وطباعة كل من عدد الدرجات الممتازة وهي من ٩٠ إلى ١٠٠ ، وعدد الدرجات المقبولة وهي من ٦٠ إلى ٨٩ ، وعدد الدرجات غير المقبولة وهي من ١ إلى ٥٩. كما يعرض البرنامج الفئة التي تنتمي إليها كل درجة من الدرجات التي قرأها.

اختبر البرنامج بالبيانات الآتية :

63	75	72	72	78	67	80	63	
75	89	90	43	59	99	82	12	100

ب) عدل البرنامج ليعرض كذلك الدرجة المتوسطة للامتحان (عدد حقيقي) في نهاية مخرجات البرنامج.

٤-٧٥ اكتب برنامجا لقراءة البطاقات الزمنية الأسبوعية لموظفي إحدى المؤسسات. وكل موظف له ثلاث بيانات خاصة : رقم الهوية ، ومعدل الأجر في الساعة ، وعدد ساعات العمل في الأسبوع. وكل موظف يصرف له راتب مرة ونصف لكل ساعات العمل الزائدة عن ٤٠ ساعة ، وقيمة الضريبة تساوي ٣,٦٢٥ في المائة من الراتب الأصلي.

ومخرجات البرنامج يظهر فيها رقم الموظف وصافي راتبه. كذلك يطبع البرنامج إجمالي المدفوعات ، ومتوسط ما دفع للموظف الواحد.

٤-٧٦ 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.

٤-٧٧ عدل البرنامج السابق (٤-٧٦) 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.

٤-٧٨ اكتب برنامجا لإجراء عمليات حساب التوفير (savings account transaction program) ، بحيث يقوم بتشغيل (processing) المجموعات التالية من البيانات.

المجموعة الأولى			المجموعة الثانية		
I	1234	1054.07	I	5723	2008.24
W		25.00	W		15.55
D		243.35	Z		

W 254.55
Z

المجموعة الثالثة

I 2814 128.24
W 52.48
D 13.42
W 84.60
Z

المجموعة الرابعة

I 7234 7.77
Z

المجموعة الخامسة

I 9367 15.27
W 16.12
D 10.00
Z

المجموعة السادسة

I 1134 12900.00
D 9270.00
Z

السجل (السطر) الأول في كل مجموعة يحتوي على رمز الشفرة "I" ورقم الحساب الجديد والرصيد الابتدائي لهذا الحساب ، ثم السطور التالية تحتوي على سجلات العمليات المختلفة على هذا الحساب ، من سحب (W)ithdrawal **وإيداع** (D)eposit**. والسطر الأخير يحتوي على قيمة مميزة (حارسة)** (Z) (sentinel value) **تشير إلى انتهاء العمليات على هذا الحساب. ويقوم البرنامج بعرض رقم الحساب والرصيد النهائي بعد تشغيل جميع سجلات المجموعة. وإذا أصبح أي رصيد سالبا فاطبع رسالة تفيد ذلك ، واتخذ أي خطوات تصحيحية مناسبة. وإذا لم يكن هناك أي عمليات بالنسبة لحساب معين فاطبع رسالة تفيد ذلك. وأخيرا أدخل رمز الشفرة** (Q)uit **ليفيد الخروج من تنفيذ البرنامج.**

٧٩-٤ اشترك أربعة أشخاص في سباق الميل (mile race) **للجري ، حيث مسافة السباق تساوي ميلا واحدا. اكتب برنامجا لقراءة الوقت الذي استغرقه كل متسابق / عداء** (runner) **بالدقائق** (Minutes) **والثواني** (Seconds) **، ثم احسب واطبع قيمة السرعة بالقدم في الثانية** (FPS) **وبالمتر في الثانية** (MPS) **، وذلك لكل متسابق.**

(ملاحظة : الميل الواحد يساوي ٥٢٨٠ قدما ، والكيلومتر الواحد يساوي ٣٢٨٢ قدما)

Minutes Seconds

3	52.83
3	59.83
4	00.03
4	16.22

٨٠-٤ اكتب برنامجا يستخدم عرى FOR لطباعة جدول الضرب التالي بالصورة المعطاة :

$$\begin{array}{lllll} 1 & x & 1 & = & 1 \\ 1 & x & 2 & = & 2 \\ & \vdots & & & \\ 1 & x & 9 & = & 9 \\ 2 & x & 1 & = & 2 \\ & \vdots & & & \\ 2 & x & 9 & = & 18 \\ & \vdots & & & \\ 9 & x & 1 & = & 9 \\ & \vdots & & & \\ 9 & x & 9 & = & 81 \end{array}$$

٨١-٤ اكتب برنامجا لطباعة جدول لقيم الدالة :

$$f \equiv f(x,y) = \begin{cases} \dfrac{e^{x}-1}{x}.e^{y^2} & ; \quad x \neq 0 \\ e^{-y^2} & ; \quad x = 0 \end{cases}$$

وذلك لجميع قيم x , y التالية :

x = -5 , -4 , … , -1 , 0 , 1 , 2 , … , 6
y = 1 , 1.5 , 2 , 2.5 , … , 5

ملاحظة : البرنامج يقوم بتوليد (وليس قراءة) جميع قيم (x , y) :

(-5 , 1) , (-5 , 1.5) , … , (-5 , 5) ,
(-4 , 1) , (-4 , 1.5) , … , (-4 , 5) ,
…..
(6 , 1) , (6 , 1.5) , … , (6 , 5)

ويطبع القيم المقابلة للدالة f.

الفصل الخامس

المنظومات
Arrays

تمهيد

نفرض أن المطلوب قراءة درجات مائتي طالب في أحد الاختبارات وحساب المتوسط العام للدرجات وهو يساوي مجموع درجات كل الطلاب مقسوما على عددهم أي على مائتين.

يمكننا إعطاء هذه الدرجات أسماء مختلفة مثل

S200 , ….. , S3 , S2 , S1

ثم قراءة قيم هذه الأسماء – أي قراءة الدرجات – وجمعها ، ولكن هذه الطريقة تكون مملة جدا ومتعبة حيث أنه يلزمنا كتابة جميع هذه الأسماء في القراءة وفي الجمع ، بالإضافة إلى أننا نشغل من ذاكرة الحاسب حيزا كبيرا للتخزين لأن كل متغير (كل درجة) له اسم مختلف وبالتالي يشغل حيزا مختلفا.

إذا كان المطلوب هو حساب المتوسط فقط فإننا بعد قراءة أي درجة يمكننا إضافتها إلى المجموع وبعد ذلك لا نحتاج لهذه الدرجة في أي عملية أخرى ولذلك فيمكننا أن نسمي الدرجة التالية باسم الدرجة السابقة نفسه ، ونقرأها تحت هذا الاسم نفسه فتحل محل الدرجة السابقة في مكان تخزينها في الذاكرة ، ثم نضيف هذه الدرجة الجديدة إلى المجموع ، ونكرر هذه العملية مع الدرجة التالية ، ثم التي بعدها ، وهكذا إلى أن نقرأ كل الدرجات – تحت الاسم الواحد نفسه – ونضيفها جميعا إلى بعضها البعض لنحصل على المجموع الكلي الذي يقسم بعد ذلك على عدد هذه الدرجات ليعطي المتوسط .. وبذلك نكون قد حققنا أمرين : الأول : سهولة كتابة البرنامج حيث أعطينا كل الدرجات الاسم نفسه ، والثاني : شغل أقل حيز ممكن من ذاكرة الحاسب حيث استخدمنا موضعا واحدا فقط للدرجات حيث كانت كل درجة تحل محل الدرجة السابقة في الموضع نفسه. (وسبق أن عملنا خطوات شبيهة بتلك في مسائل سابقة) ويمكن أن يتمثل هذا الحل الذي ذكرناه في مجموعة العبارات التالية :

s , sum , avg : REAL ;VAR

```
BEGIN
                sum := 0.0 ;
        FOR i := 1 TO 200 DO
                BEGIN
        READLN (s) ;
        sum := sum + s
                END ;
        avg := sum / 200.0 ;
WRITELN ('average = ' , avg)
                END.
```

فإذا طلب منا – بالإضافة إلى حساب هذه الدرجة المتوسطة avg – حساب الفرق بين درجة كل طالب والدرجة المتوسطة ، فإنه يلزمنا قراءة درجات الطلاب مرة أخرى حيث أنها كانت تمحى أولا بأول كما سبق شرحه ، وهذا غير مقبول من الناحية العملية أن نعيد قراءة كل البيانات كلما احتجنا إليها وربما نحتاج إليها أكثر من مرتين أيضا فقد نرغب مثلا في معرفة أعلى درجة وأقل درجة أو ترتيب الدرجات ترتيبا تصاعديا أو تنازليا أو حساب الانحراف القياسي للدرجات (وهو أحد المقاييس في علم الإحصاء) أو تقسيم الدرجات إلى فئات مختلفة لإعطاء الطلاب التقديرات المقابلة (ممتاز – جيد جدا – جيد – مقبول – راسب) وعمل إحصائية لعدد الطلاب في كل فئة ، وهكذا ...

في مثل هذه الحالات التي نحتاج فيها إلى الاحتفاظ بالبيانات وعدم محوها من ذاكرة الحاسب ، يجب أن تعطى هذه البيانات أسماء مختلفة حتى لا تحل قيمة إحداها محل قيمة أخرى .. وهنا تظهر مرة أخرى مشكلة كيفية كتابة هذه الأسماء الكثيرة المختلفة أثناء كتابة البرنامج ، وهل نضطر لكتابة أسماء مجموعة الدرجات جميعها في البرنامج وتصبح بذلك كتابة البرنامج عملا شاقا ومملا. فكرة المنظومات / المجموعات المرتبة – وهي موضوع هذا الفصل – تحل هذه المشكلة ، وتتلخص في أننا نعطي المجموعة كلها (مجموعة الدرجات مثلا) اسما واحدا ولكن إذا أضفنا له قوسين مربعين ووضعنا بينهما قيمة تشير إلى ترتيب العنصر في المجموعة فإننا نحصل على اسم هذا العنصر ، فمثلا إذا أشرنا إلى مجموعة الدرجات بالاسم S ، فإن S[4] مثلا تعني درجة الطالب الرابع ، و S[140] تعني درجة الطالب رقم 140 وهكذا ، وعلى العموم فإن S[I] تعني درجة الطالب رقم I ، فإذا كتبنا في البرنامج بعض العبارات التي تشير إلى تغير منتظم في قيمة I فإنه يمكننا إجراء أي عمليات على كل الدرجات التي نحتاج إليها دون أن نضطر لكتابة أسماء كل هذه الدرجات..

على أنه يلزم قبل إجراء أي عمليات على هذه العناصر من المنظومة سواء قراءتها أو جمعها أو أي عملية أخرى أن نعطي الحاسب في برنامجنا إعلانا لتعريفه بهذه المنظومة (Array Declaration) : ما اسمها ؟ وكم عدد العناصر التي فيها ؟ وما نوع هذه العناصر ؟ فمثلا الإعلان

```
VAR   S : ARRAY [1 .. 200] OF REAL ;
```

يفيد أن هناك منظومة اسمها S ، وعدد عناصرها 200 ، وكلها أعداد حقيقية.

وفي حالة ما إذا كانت عناصر منظومة ما كلها أعدادا صحيحة فإنها نستخدم كلمة INTEGER بدلا من كلمة REAL ، مثل :

```
VAR Alfa : ARRAY [1 .. 500] OF INTEGER ;
```

وهذا الإعلان يطلب من الحاسب حجز خمسين موضعا في ذاكرته لعناصر المنظومة Alfa ، وهذه العناصر أعداد صحيحة ، بينما الإعلان السابق يطلب حجز مائتي موضع لعناصر حقيقية هي عناصر المنظومة S.

ويجوز في إعلان التعريف بمنظومة أن نعرف معها منظومة أخرى أو أكثر ، كأن نكتب مثلا :

```
VAR
S , D , R          :  ARRAY [1 .. 200] OR REAL ;
Alfa               :  ARRAY [1 .. 500] OF INTEGER ;
list , T           :  ARRAY [-10 .. 10] OF CHAR ;
Z , Q , X , Y      :  REAL ;
J , KL             :  INTEGER ;
```

وتعامل عناصر المنظومة في عبارات البرنامج معاملة المتغيرات البسيطة (simple variables) ، فمثلا العبارة :

```
sum := sum + S[200] ;
```

تعني إضافة العنصر رقم 200 في المنظومة S إلى المجموع sum ، ولاحظ أن S[200] هو اسم متغير مؤشَّر (subscripted variable) وهو الذي يرمز له رياضيا بالرمز S_{200}.

ولقراءة أو طباعة عناصر منظومة يمكننا استخدام إحدى عبارات التكرار وأنسبها عبارة FOR ، مثل

```
FOR i := 1 TO 200 DO
    READ (S[i]) ;
```

والآن يمكننا في المثال التالي صياغة وحل المسألة التي ناقشناها سابقا.

مثال ٥-١ :

اكتب برنامجا لقراءة درجات مائتي طالب

$S_1 , S_2 , \ldots , S_i , \ldots , S_{200}$

ثم حساب

(أ) الدرجة المتوسطة

$$AVG = \left(\sum_{i=1}^{200} S_i \right) / 200$$

(ب) الفرق بين درجة كل طالب والدرجة المتوسطة

$i = 1 , 2 , \ldots , 200 \qquad D_i = S_i - AVG ;$

الحل :

نلاحظ هنا أن D تمثل كذلك منظومة / مجموعة مرتبة وهي مجموعة الفروق ، ولكننا لا نستطيع أن نحسب عناصرها إلا بعد حساب المتوسط AVG.

```
PROGRAM Scores (INPUT , OUTPUT) ;
                                  i : INTEGER ; VAR
                        S , D : ARRAY [1 .. 200] OF REAL ;
                                    avg , sum : REAL ;
                                                  BEGIN
                                                sum := 0.0 ;
                                      FOR i := 1 TO 200 DO
                                                  BEGIN
                                          READ (S[i]) ;
                                  sum := sum + S[i]
                                                  END ;
                                     avg := sum / 200.0 ;
                              WRITELN ('Average = ' , avg) ;
                                      FOR i := 1 TO 200 DO
                                                  BEGIN
                                    D[i] := S[i] - avg ;
                          WRITELN ('D[' , i , '] = ' , D[i])
                                                  END
                                                  END.
```

مؤشرات المنظومات (Array Subscripts)

ذكرنا سابقا أنه يمكن تحديد عنصر ما في منظومة بذكر رقم ترتيبه في المنظومة بين قوسين مع اسم المنظومة ، مثل S[4] ، ويقال لهذا الرقم 4 إنه مؤشر (subscript).

والصيغة العامة لمؤشر منظومة هي أنه تعبير صحيح (integer expression) بحيث أن قيمته لا تخرج عن مدى قيم مؤشرات المنظومة المعلن عنه في قسم الإعلانات والتعريفات ، أما إذا وقعت القيمة خارج هذا المدى فعادة تعطى رسالة توضح هذا الخطأ.

وفيما يلي بعض الصور المقبولة للمؤشرات

S[2*I - 5] , Y[10] , Y[J] , X[I*I - 1]
Q[5 + K] , T[M*3 + 10]

والآن ندرس بعض الأمثلة التي توضح المفاهيم السابقة قبل الانتقال إلى مفاهيم أخرى .

مثال ٥-٢ :

اكتب برنامجا لقراءة قيم عناصر منظومة X مكونة من عشرين عنصرا ، وحساب مجموع مربعاتها

$$\sum_{i=1}^{20} X_i^2$$

الحل :

```
i : INTEGER ; VAR
X : ARRAY [1 .. 20] OF REAL ;
sumsq : REAL ;
BEGIN
sumsq := 0.0 ;
FOR i := 1 TO 20 DO
BEGIN
READLN (X[i]) ;
sumsq := sumsq + SQR(X[i])
END ;
WRITELN ('Sum of squares = ' , sumsq) ;
END.
```

مثال ٥-٣ :

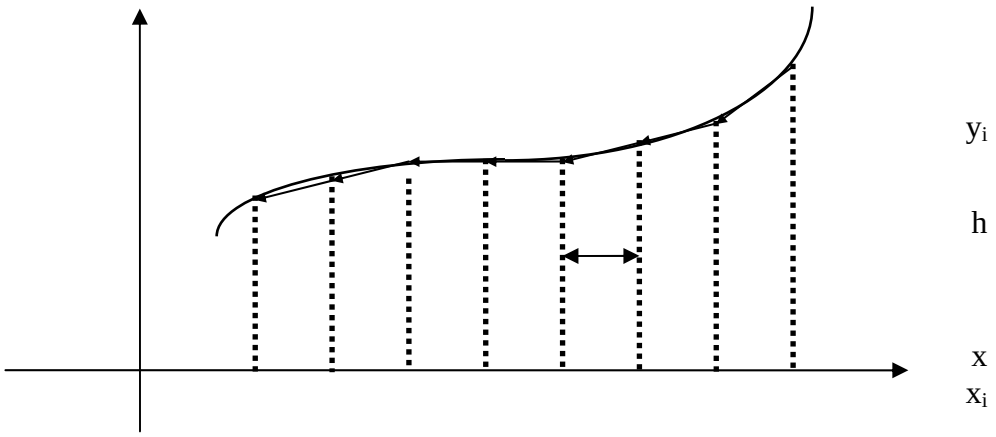
اكتب برنامجا لقراءة عناصر منظومة مكونة من سبعة وثلاثين عنصرا ، وأوجد مجموع العناصر الفردية الترتيب.

الحل :

```
i : INTEGER ; VAR
X : ARRAY [1 .. 37] OF REAL ;
Sodd : REAL ;
BEGIN   Sodd := 0.0 ;
FOR i := 1 TO 37 DO
READ (X[i]) ;
FOR i := 1TO 19 DO
Sodd := Sodd + X[2*i -1] ;
WRITELN ('Sum of odd-numbered elements = ' , Sodd)
END
```

مثال ٥-٤ : (قاعدة شبه المنحرف لحساب المساحة أو التكامل المحدود) :

يمكن تقريب المساحة المحصورة تحت منحنى $y = f(x)$ **بمجموعة مساحات أشباه منحرفات متساوية الارتفاع** (h) **كما هو مبين بالرسم.**



اكتب برنامجا لقراءة :

(أ) **قيمة الارتفاع الثابت** h **وهو المسافة بين كل قيمتين متتاليتين من قيم** x.

(ب) **٣٥ قيمة من قيم** y **المتتالية** $y_1 , y_2 , y_3 , \dots , y_{35}$

ثم لحساب المساحة المحصورة تحت المنحنى الذي تمثله هذه النقاط بتطبيق قاعدة شبه المنحرف التالية

$$\text{Area} = \frac{h}{2}\left[y_1 + 2\{y_2 + y_3 + y_4 + \text{L} + y_{34}\} + y_{35}\right]$$

الحل :

```
PROGRAM TrapezoidalRule (INPUT , OUTPUT) ;
                                  i : INTEGER ; VAR
                        y : ARRAY [1 .. 35] OF REAL ;
                                h : sum , area : REAL ;
                                                BEGIN
                                          READLN (h) ;
                                    FOR i := 1 TO 35 DO
                                  READLN (Y[i]) ;
                                          sum : 0.0 ;
                                    FOR i := 2 TO 34 DO
                                sum := sum + Y [i] ;
                    area := h / 2.0 * (Y [1] + 2.0 * sum + Y[35] ;
                  WRITELN ('Area by trapezoidal rule = ' , area)
                                                END.
```

مثال ٥–٥ (قاعدة سمبسون لحساب المساحة أو التكامل المحدود) :

حل مسألة المثال السابق ولكن بتطبيق قاعدة سمبسون التالية :

$$\text{Area} = \frac{h}{3}\Big[y_1 + y_{35} + 4\big(y_2 + y_4 + y_6 + \ldots\ldots + y_{34}\big) + 2\big(y_3 + y_5 + y_7 + \ldots\ldots + y_{33}\big)\Big]$$

الحل :

```
PROGRAM SimpsonRule (INPUT , OUTPUT) ;
                                  i   : INTEGER ;          VAR
                        Y : ARRAY [1..35] OF REAL ;
                          h , even , od , area  : REAL ;
                                                BEGIN
                                          READLN (h) ;
                                    FOR i := 1 TO 35 DO
                                  READLN (Y [i]) ;
                                          even := 0.0 ;
                                    FOR i := 1 TO 17 DO
                                even := even + Y[2*i] ;
                                          od := 0.0 ;
                                    FOR i := 1 TO 16 DO
                                od := od + Y[2*i + 1] ;
area := h/3.0 *(Y[1] + Y[35] + 4.0 * even + 2.0 * od) ;
                  WRITELN ('Area by Simpson Rule  = ' , area) ;
                                                END.
```

ملاحظة :

يمكن كتابة البرنامج باستخدام عروة FOR واحدة فقط بدلا من اثنتين بعمل التعديل التالي :

```
even := 0.0 ; od := 0.0 ;
FOR i := 1 TO 16 DO
    BEGIN
even := even + Y[2*i] ;
od := od + Y[2*i + 1]
    END ;
area := h/3.0 *(Y[1] + Y[35] + 4.0 * (even + Y[34]) + 2*od) ;
```

مثال ٥-٦ :

اكتب برنامجا لقراءة عناصر منظومة A مكونة من عشرين عددا حقيقيا ، وتعيين أصغر عدد.

الحل :

فكرة الحل أننا سنسمي أصغر عدد SMALL وسنبدأ بفرض أن أول عدد هو أصغر عدد ، أي أن SMALL هو A[1] ثم نقارن SMALL مع بقية العناصر كلها ابتداء من A[2] إلى A[20] ، فإذا وجدنا أن هناك عنصرا – A[I] مثلا – أصغر من SMALL محونا قيمة SMALL ووضعنا محلها قيمة A[I] ، ثم نستمر في مقارنة SMALL (القيمة الجديدة) مع بقية العناصر أي ابتداء من A[I+1] إلى A[20] وهكذا. أما إذا لم نجد أي عنصر أصغر من SMALL فإننا نستمر في المقارنة إلى ننتهي من كل عناصر المنظومة.

فمثلا بالنسبة للمنظومة المبينة فيما يلي والمكونة من ثمانية عناصر ، يأخذ المتغير SMALL القيم التالية على الترتيب

$a_1 = 5$	
$a_2 = 8$	
4	SMALL
10	5
7	
2	4
6	
$a_8 = 3$	2

وبالتالي يكون العدد 2 هو أصغر عنصر في المنظومة. والآن نكتب البرنامج المطلوب لحل مسألة المثال.

```
PROGRAM SmallestElement (INPUT , OUTPUT) ;
i : INTEGER ; VAR
A : ARRAY [1..20] OF REAL ;
```

```
                                SMALL : REAL ;
                                BEGIN
                    FOR i := 1 TO 20 DO READLN A[i] ;
                                SMALL := A[1] ;
                                FOR i := 2 TO 20 DO
                                IF SMALL > A[i] THEN
                                SMALL := A[i] ;
                    WRITELN ('The smallest number is : ' , SMALL)
                                END.
```

ملاحظة :

من الأخطاء التي تحدث أحيانا في كتابة مثل هذا البرنامج كتابة عبارة IF بالصورة التالية :

```
IF  A[1] > A[I]  THEN  SMALL := A[I] ;
```

اعتمادا على أن

```
SMALL = A[1]
```

كما ورد في عبارة سابقة في البرنامج ، ولكن الصحيح أن

```
SMALL = A[1]
```

في البداية ، ولكن بعد بعض المقارنات قد تختلف قيمة SMALL عن قيمة A[1] كما في هذا المثال ، وبالتالي فإن عبارة IF لا تؤدي عندئذ المقارنة المطلوبة.

وللتأكد من ذلك يمكن الرجوع إلى المثال التوضيحي الذي ذكرناه أثناء مناقشة فكرة الحل ، وتطبيق البرنامج بعد تغيير عبارة IF إلى الصورة المذكورة سابقا ، لنجد أن المتغير SMALL يأخذ القيم التالية على الترتيب.

```
SMALL
  5
  4
  2
  3
```

أي أنه ينتج أن العدد 3 هو أصغر عنصر في المنظومة وهذا غير صحيح.

مثال ٥–٧ :

اكتب برنامجا لقراءة عناصر منظومة A مكونة من عشرين عددا حقيقيا ، وإعادة ترتيب عناصرها – إذا لزم الأمر – بحيث يكون أصغر عنصر في الموضع A[1].

الحل :

في هذه المسألة سوف لا نستعمل اسما خاصا لأصغر عنصر مطلوب – وبالتالي سوف لا نحجز له مكانا جديدا خاصا في ذاكرة الحاسب – وإنما سنفترض أولا أن A[1] **هو أصغر عنصر ثم نقارنه ببقية العناصر على الترتيب ابتداء من** A[2] **فإذا وجدنا أن هناك عنصرا مثل** A[i] **أصغر من** A[1] **بدلنا العنصرين مكان بعضيهما البعض (انظر المسألة رقم ٢–٨) ، ثم نستمر في مقارنة العنصر** A[1] **الجديد مع بقية العناصر على الترتيب ابتداء من** A[i+1] **إلى** A[20] **فإذا وجدنا عنصرا أصغر من** A[1] **قمنا بعملية التبديل وهـكذا ، وفي النهاية يكون أصغر عنصر في الموضع** A[1] **، وفي الوقت نفسه لم نفقد أو لم نمح أي عنصر من المنظومة أي أنه قد أعيد ترتيبها (إذا حدثت أي عملية تبديل). وكمثال توضيحي نرجع أيضا لمجموعة العناصر الثمانية ونرى أن الطريقة المذكورة تؤدي إلى إعادة ترتيب المنظومة بالصور التالية على التوالي :**

أولا	ثانيا	ثالثا
5	4	2
8	8	8
4	5	5
10	10	10
7	7	7
2	2	4
6	6	6
3	3	3

والبرنامج التالي يقوم بهذه العملية :

```
PROGRAM SmallestFirst (INPUT , OUTPUT) ;
                    i : INTEGER ; VAR
          Y : ARRAY [1..20] OF REAL ;
                       Temp : REAL ;
                                BEGIN
                     FOR i := 1 TO 20 DO
                   READLN (A[i]) ;
                     FOR i := 2 TO 20 DO
              IF A[1] > A[i] THEN
                        BEGIN
          Temp := A[1] ;
           A[1] := A[i] ;
           A[i] := Temp
                           END ;
                     FOR i := 1 TO 20 DO
                  WRITELN (A[i]) ;
                                END.
```

ويقوم البرنامج بطباعة المنظومة بعد إعادة ترتيبها (إذا لزم الأمر) بصورة رأسية بحيث أن A[1] الآن هو أصغر عنصر.

مثال ٥-٨ : [ترتيب (Sorting) عناصر منظومة] :

اكتب برنامجا لقراءة عناصر منظومة مكونة من عشرين عددا حقيقيا وترتيب هذه العناصر ترتيبا تصاعديا (sorting in an ascending order).

الحل :

- **في المثال السابق وضعنا أصغر عنصر في الموضع A_1.**
- **نكرر الطريقة مع العناصر المتبقية من A_2 إلى A_{20}بحيث نضع أصغر عنصر من عناصر هذه المجموعة المتبقية في الموضع A_2.**
- **ثم نكرر الطريقة مع العناصر المتبقية من A_3 إلى A_{20} ونضع أصغرها في A_3 .**
- **ونستمر في هذه العملية إلى أن نضع أصغر العنصرين المتبقيين A_{19} و A_{20} في الموضع A_{19}.**
- **وبذلك تصبح المجموعة A مرتبة ترتيبا تصاعديا.**

يمكن تلخيص طريقة الحل في العبارة التالية

أعد ترتيب عناصر المجموعة $A_i, A_{i+1}, \dots , A_{20}$ لتضع أصغر عنصر في موضع A_i [حيث i = 1,2,3,...,19] وذلك بطريقة مثال ٥-٧ حيث تقارن أول عنصر A_i في المجموعة مع بقية العناصر A_j [حيث j = i+1 , i+2 , ... , 20] وتقوم بتبديل العنصرين A_i , A_j إذا كان A_j أصغر من A_i. واضح من هذه العبارة أننا نحتاج في البرنامج إلى عروتي FOR ، واحدة لتغيير i والثانية لتغيير j.

والبرنامج التالي يقوم بتطبيق الطريقة المذكورة.

```
PROGRAM SortingAscending (INPUT , OUTPUT) ;
VAR
        i,j : INTEGER ;
        A : ARRAY [1..20] OF REAL ;
        Temp : REAL ;
BEGIN
    FOR i := 1 TO 20 DO
        BEGIN
    WRITE ('Enter A[' , i : 2,'] ') ;
        READLN (A[i]) ;
```

```
                END;
            FOR i := 1 TO 19 DO
        FOR j := i+1 TO 20 DO
    If A[i] > A[j] THEN
        BEGIN
Temp := A[i] ;
  A[i] := A[j] ;
  A[j] := Temp
            END ;
        FOR i := 1 TO 20 DO
      WRITELN (A[i]) ;
END.
```

مثال ٥-٩ : [البحث (Searching) في منظومة عن عنصر معين] :

تتكون منظومة الأعداد الحقيقية A من ثمانين عنصرا A_{80} , ... , A_2 , A_1. اكتب برنامجا :

(أ) يقرأ قيم عناصر المنظومة A.

(ب) يقرأ قيمة عدد حقيقي Y (قد تكون هذه القيمة موجودة كعنصر في المنظومة A وقد لا تكون).

(ج) يحسب قيمة عدد صحيح J كالآتي :

- **إذا كان Y هو أحد عناصر المنظومة A فإن J هو ترتيب العنصر Y في هذه المنظومة.**
- **إذا لم يكن Y عنصرا في المنظومة A فإن J تساوي صفرا.**

الحل :

```
Program Search (input , output) ;
                                var
        i , J : integer ; found : boolean ;  Y : real ;
                A : array [1..80] of real ;
                                begin
        for i := 1 to 80 do readln (A[i]) ;
                        readln (Y) ;
                i := 1 ; found := false ;
while (i  <= 80) and (not found) do
                    if  Y = A [i]  then
                found : =  true
                            else
```

```
        i := i + 1 ;
            if found then
              J := i
                  else
             J := 0 ;
    WRITELN  ('J = ' , J)
                end.
```

المصفوفات أو المنظومات ذوات البعدين :
(Matrices - Two Dimensional Arrays)

تعد المصفوفة منظومة ذات بعدين حيث أنه يلزم لتعيين أي عنصر في المنظومة تحديد رقمين يمثلان رقم الصف ورقم العمود الذي يتواجد فيه العنصر. والعنصر a_{ij} في المصفوفة A والموجود في الصف رقم i وفي العمود رقم j يرمز له في الباسكال بالرمز A[i,j] ، فمثلا A[3,4] هو العنصر الموجود في الصف الثالث وفي العمود الرابع.

وكما في حالة المنظومة الخطية (linear array) أي ذات البعد الواحد ، يلزم أيضا تعريف المنظومة ذات البعدين أي المصفوفة في الإعلانات :

(اسم المصفوفة – نوع عناصرها – عدد صفوفها وعدد أعمدتها) ، فمثلا الإعلان

```
MARK : ARRAY [1..50 , 1..5] OF INTEGER
```

يفيد أن MARK هو اسم مصفوفة مكونة من خمسين صفا وخمسة أعمدة (وبالتالي فعدد عناصرها يساوي ٥٠ × ٥ = ٢٥٠) وأن كل عناصرها أعداد صحيحة. ومن ناحية المعنى الطبيعي لهذه الأعداد والمسألة العملية التي نحتاج في حلها إلى هذه المصفوفة فقد تمثل هذه الأعداد مثلا درجات طلاب في عدة مواد ، بحيث أن عندنا خمسين طالبا يدرسون خمس مواد ، فصفوف المصفوفة تمثل الطلاب وأعمدتها تمثل المواد. أي أن الصف رقم I يعطي درجات الطالب رقم I في الخمس مواد ، والعمود رقم J يعطي درجات الخمسين طالبا في المادة رقم J ، فالعنصر MARK[I,J] يعطي درجة الطالب رقم I في المادة رقم J.

وما قيل سابقا عن مؤشر (Subscript) العنصر في المنظومة ذات البعد الواحد ينطبق أيضا على كل من مؤشري العنصر في المنظومة ذات البعدين ، بمعنى أن كلا منهما تعبير صحيح (integer expression) بحيث أن قيمته لا تقل عن ١ ولا تزيد عن عدد

الصفوف أو عن عدد الأعمدة حسب ما إذا كان المؤشر يشير إلى رقم الصف أو رقم العمود على الترتيب ، وعموما قيمة التعبير لا تخرج عن المدى المسموح به لقيم المؤشر.

قراءة وطباعة عناصر المصفوفة :

(١) عادة نفضل قراءة أو طباعة عناصر المصفوفة صفا صفا وبحيث يكون كل صف على سطر مستقل ، حيث أنها بذلك تكون أكثر وضوحا وأكثر دلالة على المعنى الطبيعي الذي تمثله والمسألة التي تعبر عنها .. ويمكن تحقيق ذلك باستخدام عبارتي FOR :

تستخدم إحداهما للانتقال من صف إلى آخر ابتداء من الصف الأول إلى الصف الأخير ، وتستخدم الأخرى لكتابة عناصر الصف الواحد أي للانتقال من العمود الأول إلى الأخير للسطر نفسه. فمثلا لطباعة عناصر مصفوفة Q مكونة من صفين وثلاثة أعمدة ، نكتب

```
FOR i := 1 TO 2 DO
    BEGIN
        FOR j := 1 TO 3 DO
    WRITE (Q[i,j]) ;
            WRITELN
        END ;
```

فتكون النتيجة طباعة عناصر المصفوفة بالترتيب التالي

```
Q[1,1]    Q[1,2]    Q[1,3]
Q[2,1]    Q[2,2]    Q[2,3]
```

وذلك لأنه عندما نبدأ بوضع i = 1 فإن عروة FOR الداخلية تعني

```
FOR j :=1 TO 3 DO WRITE (Q[1,j]) ;
```

وهذه تكافئ :

```
WRITE (Q[1,1]) , WRITE (Q[1,2]) , WRITE (Q[1,3])
```

ولذلك نكتب هذه العناصر الثلاثة على السطر التالي .. وأما عبارة WRITELN فإنها تضمن لنا طباعة قيم العناصر الثلاثة الأخرى على سطر جديد.

(٢) أما إذا استخدمنا الأمر WRITELN في عروة FOR (الداخلية) هكذا :

```
FOR i := 1 TO 2 DO
    FOR j := 1 TO 3 DO
  WRITELN (Q[i,j]) ;
```

فإن العناصر تطبع بالصورة التالية

```
Q[1,1]
Q[1,2]
```

Q[1,3]
Q[2,1]
Q[2,2]
Q[2,3]

حيث يظهر كل عنصر على سطر واحد.

وفي هذه الحالة فإن FOR الخارجية تثبِّت أولا قيمة i لتساوي ١ ، ثم تقوم FOR الداخلية بتغيير j على الترتيب من ١ إلى ٢ إلى ٣ فيؤدي ذلك وبناء على الأمر WRITELN إلى طباعة الثلاث قيم الأولى (على الثلاثة سطور الأولى كما مبين سابقا) ، ثم تقوم FOR الخارجية بتغير قيمة i إلى القيمة التالية ٢ (وفي هذا المثال هي أيضا قيمة i الأخيرة) وبينما i ثابتة عند هذه القيمة ٢ تتغير قيمة j بفعل FOR الداخلية مرة أخرى من ١ إلى ٢ إلى ٣ فيؤدي ذلك إلى طباعة الثلاث قيم التالية ..

وعلى العموم فهذه الصورة من النتائج غير مقبولة بالنسبة للمصفوفات.

* * *

والآن ندرس بإذن الله تعالى بعض الأمثلة على البرمجة باستخدام المصفوفات.

مثال ٥-١٠ :

نفرض أن المصفوفة A تتكون من خمسة صفوف وسبعة أعمدة .. اكتب بعض العبارات التي تجعل كل عناصر المصفوفة A أصفارا.

الحل :

```
VAR
    i,j : INTEGER ;
  A : ARRAY [1..5 , 1..7] OF REAL ;
BEGIN
      FOR i := 1 TO 5 DO
  FOR j := 1 TO 7 DO A[i,j] := 0.0
END.
```

مثال ٥-١١ :

أوجد مجموع عناصر الصف الثالث ومجموع عناصر العمود الخامس في مصفوفة A مكونة من خمسة صفوف وثمانية أعمدة.

الحل :

مجموع عناصر الصف الثالث يعطى بالعلاقة

$$S_1 = a_{3,1} + a_{3,2} + a_{3,3} + L + a_{3,8}$$

$$= \sum_{j=1}^{8} a_{3,j}$$

بينما مجموع عناصر العمود الخامس يعطى بالعلاقة

$$S_2 = a_{1,5} + a_{2,5} + \cdots + a_{5,5}$$

$$= \sum_{i=1}^{5} a_{i,5}$$

والعبارات التالية تترجم هاتين العلاقتين إلى لغة الباسكال

```
VAR
    i,j : INTEGER ;
    A : ARRAY [1..5 , 1..8] OF REAL ;
    S1 , S2 : REAL ;
BEGIN
    .....
    .....
    .....
    S1 : 0.0 ;
    FOR j := 1 TO 8 DO
    S1 := S1 + A[3,j] ;
    S2 := 0.0 ;
    FOR i := 1 TO 5 DO
    S2 := S2 + A[i,5] ;
```

مثال ٥-١٢ :

نفرض أن كلا من A , B , C مصفوفة مربعة تحتوي على عدد N من الصفوف وN من الأعمدة ، حيث $N \leq 6$. ونفرض أن كلا من X , Y متجه / منظومة تحتوي على عدد N من العناصر.

اكتب برنامجا يقرأ قيمة N وقيم عناصر كل من المصفوفتين A , B والمتجه X ثم يستخدم عبارات FOR لحساب :

(أ) عناصر المتجه Y والذي ينتج من ضرب المصفوفة A والمتجه X (Y = AX)
حيث :

$$Y_i = \sum_{j=1}^{N} A_{ij} X_j \, , \; i = 1,2,...,N$$

(ب) عناصر المصفوفة C والتي تنتج من ضرب المصفوفتين A, B (C = AB) حيث :

$$C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj} \quad ; \quad i = 1,2,\ldots,N,$$

$$j = 1,2,\ldots,N$$

الحل :

```
PROGRAM MatrixMultiplication (INPUT , OUTPUT) ;
                                                VAR
                                    i , j , k , N : INTEGER ;
                          A , B , C : ARRAY [1..6 , 1..6] OF REAL ;
                                X , Y : ARRAY [1..6] OF REAL ;
                                        {input The data}      BEGIN
                                          WRITE (′Enter N′) ;
                                              READLN (N) ;
                          WRITELN (′Enter the matrix A row by row ′) ;
                                          FOR i := 1 TO N DO
                                                BEGIN
                          WRITE (′Row number′ , i :2 , ′ : ′) ;
                                        FOR j := 1 TO N DO
                                        READ (A [i,j]) ;
                                                READLN
                                                    END ;
                          WRITELN (′Enter the matrix B row by row′) ;
                                          FOR i := 1 TO N DO
                                                BEGIN
                          WRITE (′ROW number′ , i : 2 , ′ : ′) ;
                                        FOR j := 1 TO N DO
                                        READ (B[i,j]) ;
                                                READLN
                                                    END ;
                                    WRITELN (′Enter the vector X′) ;
                                FOR i := 1 TO N DO READ (X[i]) ;
                {Output the data : matrices A and B and the vector X}
                                    WRITELN (′The Matrix A′) ;
                                          FOR i := 1 TO N DO
```

```
                                        BEGIN
                                    WRITELN ;
                                FOR j := 1 TO N DO
                        WRITE (A [i,j] : 9 , ′  ′)
                                        END ;
                                WRITELN (′The Matrix B′) ;
                                    FOR i := 1 TO N DO
                                        BEGIN
                                    WRITELN ;
                                FOR j := 1 TO N DO
                        WRITE (B [i,j] : 9 , ′  ′)
                                        END ;
                                WRITELN (′The Vector X ′) ;
                        FOR i := 1 TO N DO WRITELN (X[i]) ;
                                {Compute Y = A . X
                    Multiplication of a matrix and a vector}
                                FOR i := 1 TO N DO
                                    BEGIN
                                Y[i] := 0.0 ;
                            FOR j := 1 TO N DO
                    Y[i] := Y[i] + A[i,j] * X[j]
                                        END ;
                                {Output the vector Y}
    WRITELN (′Multiplication of a matrix and a vector ′)
                            WRITELN (′The vector Y = A.X is′)
                                FOR i := 1 TO N DO
                                WRITELN (Y[i]) ;
                                {Compute C = A.B
                     Multiplication of two matrices}
                                FOR i := 1 TO N DO
                                FOR j := 1 TO N DO
                                    BEGIN
                                C [i,j] := 0.0 ;
                            FOR k := 1 TO N DO
                C[i,j] := C[i,j] + A[i,k]*b[k,j]
                                        END ;
                                {Output the matrix C}
             WRITELN (′Multiplication of two matrices′) ;
                    WRITELN (′The matrix C = A.B is ′) ;
                                FOR i := 1 TO N DO
                                    BEGIN
                                WRITELN ;
                    FOR j := 1 TO N DO
```

```
            WRITE (C[i,j] : 9 , ' ')
                        END
                    END.
```

ملاحظات :

١- نظرا لأن قيمة N غير معلومة قبل بدء تنفيذ البرنامج ولكننا نعلم أن $N \leq 6$ لذلك فإننا نحجز للمنظومات المذكورة في المسألة أكبر قدر ممكن من الأماكن التي يمكن أن تشغلها عناصرها ، فمثلا بالنسبة للمنظـومة (أو المصفوفة) A نحجز عدد 6 x 6 من الأماكن ، وبالنسبة للمنظومة X نحجز عدد 6 من الأماكن وهكذا كما هو واضح في قسم الإعلانات في بداية البرنامج ، ولا يجوز في هذا الموضع أن نستخدم الرمز N في حجز الأماكن لأن قيمة N غير معلومة الآن.

٢- هناك طريقة أخرى يمكن اتباعها في حالة عدم معرفة عدد عناصر المنظومة بالضبط قبل بدء تنفيذ البرنامج ، ونذكر هذه الطريقة فيما يلي :

بالنسبة للمثال السابق (مثال ٥-١٢) نعلم أن أقصى قيمة تصل إليها N هي 6 ، ولذلك فيمكننا أن نستعمل ثابتا جديدا نسميه max مثلا يشير إلى القيمة القصوى للمتغير N ، أي أن قيمة max تساوي 6 في هذا المثال ، ونستعيض عن عبارتي الإعلان عن المنظومات بعبارتين أخريتين ، كما يتضح فيما يلي :

```
CONST
    max = 6 ;
VAR
    i,j,k : INTEGER;
    A,B,C : ARRAY [1..max , 1..max] OF REAL ;
    X , Y : ARRAY [1..max] OR REAL
```

وتبقى بقية البرنامج كما هي.

وإذا أردنا تشغيل البرنامج نفسه ليتعامل مع مصفوفات ومتجهات أكبر (أي أن عدد الصفوف والأعمدة في المصفوفة مثلا أكبر من ستة) فما علينا إلا أن نغير عبارة الإعلان عن الثابت max بحيث تناسب المنظومات الجديدة كأن نكتب مثلا :

```
CONST max = 10 ;
```

ويبقى البرنامج كله باستثناء هذه العبارة كما هو.

٣- يمكننا اتباع طريقة أخرى للإعلان عن متغيرات البرنامج باستخدام ما يسمى بالنوع المعرف بواسطة المستخدم [*] (user defined type) ، وذلك في قسم الإعلانات كما في المثال التالي بالنسبة لمتغيرات برنامج مثال ٥-١٢ :

```
CONST
    max = 6 ;
TYPE
    Matrix = ARRAY [1..max , 1..max] OR REAL ;
    Vector = ARRAY [1..max] OF REAL ;
    Index = 1..max ;
VAR
    : Index ;  i,j,k
    : 1..max ;      n
    A,B,C : Matrix ;
    : Vector ;  X,Y
```

حيث نلاحظ أن 1..max يمثل مدى جزئيا من مدى الأعداد الصحيحة ، وهذا المدى الجزئي يشير إلى الأعداد الصحيحة من 1 إلى max ، وقد أطلقنا عليه الاسم التعريفي Index. ونلاحظ أن كلا من i,j,k هو من هذا النوع Index ، بينما كل من A,B,C هو من النوع Matrix (الذي هو مصفوفة أي منظومة ثنائية البعد) ، وكل من x,y هو من النوع Vector الذي هو متجه أي منظومة أحادية البعد.

وأما صلب (أو جسم) البرنامج ، وهو الجزء الذي يلي الإعلانات فيبقى كما هو في حل المثال.

ملاحظة حول استعمال الأسماء التعريفية للمنظومات :

(أ) إذا كان كل من A , B متغيرا يمثل منظومة من النوع نفسه فإنه يمكن كتابة عبارة الإسناد :

```
B := A ;
```

وتعني إسناد كل عنصر من عناصر المنظومة A إلى العنصر المقابل في المنظومة B. فمثلا إذا فرضنا أن كلا من A , B مصفوفة 6 x 8 ، فإن عبارة الإسناد السابقة تكافئ العروتين المتداخلتين :

```
FOR i := 1 TO 6 DO
FOR j := 1 TO 8 DO
```

---

[*] انظر الملحق (أ).

```
B[i,j] := A[i,j] ;
```

(ب) لا يسمح باستعمال الأسماء التعريفية للمنظومات في التعابير ، فلا يجوز مثلا كتابة عبارة مثل :

```
C := A + B
```

متغيرات سلسلة الرموز (String Variables) :

لم نستعمل المتغيرات من النوع الرمزي CHAR **حتى الآن إلا في نطاق ضيق ، واستعملنا المتغير بحيث تكون قيمته رمزا واحد فقط** (Single character value) **، ونود فيما يلي – فيما تبقى من هذا الفصل – دراسة معالجة المنظومات التي تشتمل عناصرها على رموز** (Character arrays) **وما يسمى بالمنظومات المُحْزَمَة أو المُحْكَمَة** (packed arrays of characters) **وأهميتها وميزتها.**

نفرض أن لدينا في قسم الإعلانات عن المتغيرات الإعلان

```
VAR
   Name : ARRAY [1..12] OF CHAR ;
   i    : INTEGER ;
```

فهذا يعني أن Name **منظومة رمزية من ١٢ عنصرا ، وفي كل عنصر منها يمكن تخزين رمز واحد. فمثلا العبارتان (العروتان) :**

```
FOR i := 1 TO 12 DO
   READ (Name [i]) ;
FOR i := 1 TO 12 DO
   WRITE (Name [i]) ;
```

تقومان بقراءة سلسلة من الرموز (sequence of characters) **رمزا رمزا** (one at a time) **في المنظـومة** Name **، ثم بطباعـة هـذه المنظـومة. فمثلا إذا أدخلنا الرموز** A.B. Al-Razi **، فإن المنظومة** Name **تصبح مُعَرَّفة كما هو مبين في الشكل التالي ، حيث يشتمل العنصر** Name [5] **على فراغ :** (blank character)

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
A	.	B	.		A	1	-	R	a	z	i

وكأي منظومة أخرى فإنه يجب استخدام عروة لقراءة وطباعة عناصر المنظومة. وفيما يلي نُعَرِّف نوعا خاصا من المنظومات يمكننا من طباعة المنظومة الرمزية بدون استخدام عروة.

تخزين السلاسل الرمزية في المنظومات المُحْزَمة (المُحْكَمة)

(Storing Character Strings in Packed Arrays)

الصيغة العامة لكتابة المنظومة المحزمة هي :

```
TYPE
Name = PACKED ARRAY [1..n] OR CHAR ;
(n > 1)
```

وتستخدم المنظومة المحزمة لتخزين سلسلة رموز (String of characters) ، وفائدة استخدام المنظومة المحزمة أنها تؤدي عموما إلى استخدام حَيِّز أقل في التخزين في ذاكرة الحاسب ، كما تؤدي إلى سهولة ومرونة أكبر في معالجة سلاسل الرموز (manipulating character strings) . فمثلا يمكن لكل من المنظومتين Name1 , Name2 المعلن عنهما فيما يلي تخزين سلسلة رموز طولها STRINGSIZE [١٢ رمزا 12 characters].

```
CONST
STRINGSIZE = 12 ;
TYPE
STRING = PACKED ARRAY [1..STRINGSIZE] OF CHAR ;
VAR
Name1 , Name2 : STRING ;
```

فمثلا عبارة الإسناد

```
Name1 := 'A.B. Al-Razi'
```

التي تسند سلسلة رموز لـ Name1 ، تخزِّن القيمة (قيمة سلسلة) 'A.B. Al-Razi' في المنظومة المحكمة name1 هكذا :

'A' في الموضع	Name1 [1]
'.' في الموضع	Name1 [2]
:	
:	
:	
'i' في الموضع	Name1 [12]

بينما إذا لم تكن Name1 منظومة محكمة فإن عبارة الإسناد السابقة تكون خاطئة (خطأ تركيبيا أو بنائيا Syntax error).

ويجب أن تكون قيمة السلسلة المسنَدة من نفس نوع (type) المنظومة المحكمة التي تستقبل السلسلة ، وهذا يعني أن عدد الرموز (characters) في قيمة السلسلة (string value) يجب أن يساوي بالضبط حجم (size) المنظومة (أي يساوي STRINGSIZE) ، فمثلا كل من العبارتين التاليتين لإسناد سلسلة :

```
Name1 := 'Al-Razi'
Name1 := 'Abu-Bakr Al-Razi'
```

تعد خاطئة خطأ تركيبيا ، وذلك لأن قيمة السلسلة المسندة في العبارة الأولى قصيرة (invalid string , too short) وفي الثانية طويلة (too long).

وعبارة الإسناد

```
Name2 := Name1 ;
```

تعني انسخ (copy) Name1 في Name2 ، أي انسخ في Name2 السلسلة المخزونة في Name1.

فإذا أتبعنا العبارة بعبارة الإسناد :

```
Name2 [7] := 'r' ;
```

فإنها تعني تغيير الرمز السابع فقط المخزون في المنظومة Name2 إلى 'r' . وبذلك فإن العبارتين :

```
WRITELN (Name1) ;
WRITELN (Name2) ;
```

تؤديان إلى ظهور السطرين :

```
A.B. Al-Razi
A.B. Al-Razi
```

أي أن المنظومة المحكمة من الرموز يمكن أن تستخدم كوسيط (parameter) في عبارة WRITELN ، وحينئذ تطبع الرموز كسلسلة واحدة. ولاحظ أن عبارات WRITELN هذه لا تكون صحيحة إلا مع المنظومات المحكمة. ولا توجد طريقة مختصرة لقراءة سلسلة رموز في منظومة محكمة ، بل يجب استخدام عروة لقراءة الرموز واحدا واحدا كما هو الحال مع المنظومات العادية.

خواص المتغيرات السلاسل (Properties of String Variables)

نجمع فيما يلي خواص المتغيرات السلاسل أي المتغيرات التي أعلن عنها أنها منظومات محكمة من الرموز :

١- تنطبق عليها جميع خواص المنظومات الأحادية البعد (أي المنظومات الخطية أو المتجهات) من حيث الوصول إليها واستعمال عناصرها ، وإسناد متغير سلسلة إلى متغير سلسلة آخر له الحجم نفسه.

٢- يمكن أن نسند لمتغير سلسلة ثابتا عبارة عن سلسلة (أي ثابت سلسلة ، أي متتالية من الرموز sequence of characters محاطة بحاصرتين) على أن يكون له الطول نفسه ، كما رأينا في المثال :

Name1 := 'A.B. Al-Razi '

حيث سبق الإعلان عن Name1 أنه String ، وسبق تعريف String أنه منظومة محكمة من ١٢ عنصرا. وكان من الممكن تعريف Name1 مباشرة هكذا.

```
VAR
Name1 : PACKED ARRAY [1..12] OF CHAR ;
```

ويجوز أن نسند للمتغير Name1 ثابتا بالصورة التالية :

Name1 := 'Al-RaziΔΔΔΔΔ'

حيث توجد ٥ فراغات أقصى يمين الثابت ، ويصبح طوله الإجمالي ١٢.

وكما أشرنا سابقا ، فإن عبارة الإسناد هذه تصبح غير صحيحة إذا لم نكتب كلمة PACKED في تعريف Name1 ، أي إذا كان Name1 منظومة عادية وليس محكمة ، وفي هذه الحالة يجب استخدام عروة لإسناد رموز الثابت واحدا تلو الآخر لعناصر Name1 المتتالية.

٣- الاسم التعريفي للمتغير السلسلة هو النوع الوحيد من متغيرات المنظومات الذي يمكن أن يظهر في قائمة إخراج (output list) عبارة طباعة WRITE / WRITELN .

أي أننا لا نحتاج لاستخدام عروة لطباعة قيمة متغير سلسلة.

أما في عبارة قراءة READ / READLN فكبقية الأسماء التعريفية لمتغيرات المنظومات لا يجوز ظهور الاسم التعريفي للمتغير السلسلة في قائمة الإدخال ، ويجب استخدام عروة لقراءة عناصره.

٤- **المؤثرات العلاقية** =< , < , <> , > , => , = , < **يمكن أن تستخدم لمقارنة المتغيرات السلاسل والثوابت التي لها الطول نفسه ، وذلك تبعا للترتيب المعمول به أي المستخدم في الآلة** machine - used **لمجموعة الرموز ، أي تبعا لسلسلة المقارنة** [*] (Collating sequence).

[*] انظر الملحق (أ).

تمرينات رقم ٥

أولا : تدريبات

٥-١ عين قيمة كل عنصر من عناصر المنظومة x فيما يلي بعد تنفيذ مجموعة العبارات المعطاة. وإذا لم توجد أي قيمة لعنصر ما في المنظومة فضع علامة استفهام في موضع هذا العنصر.

```
type
    index = 1 .. 8 ;
list = array [index] of integer ;
var
    x : list ;
    i : integer ;
begin
    for i := 8 downto 1 do
        if i > 1 then
x[i] := 9 - i ;
end
```

(i)

```
type
    index = 1 .. 8 ;
list = array [index] of integer ;
var
    x : list ;
    i : integer ;
begin
    for i := 1 to 8 do
        begin
    if i <= 3 then
x[i] := sqr (i)  ;
        else
    x[i] := -2
        end
end
```

(ii)

```
type
    index = 1 .. 8 ;
list = array [index] of integer ;
```

```
var
    x : list ;
  i , j : integer ;
begin
  for i := 1 to 8 do
    begin
  j := (2 * i - 1) ;
    if i <= 3 then
  x[j] :=  i-j
    end
end
```

(iii)

٥-٢ ما هي القيم المخزونة في المنظومة m بعد تنفيذ مجموعة العبارات التالية :

```
type
  index = 1 .. 10 ;
list = array [index] or integer ;
var
  m : list ;
  i : integer ;
begin
  for i := 1 to 10 do
m[i] := i + 10 ;
  m[2] := 6 ;
  for i := 1 to 10 do
if (3*i) < m[i] then
m[i] := m[i] - i
```

٥-٣ تتبع تنفيذ عبارات قطعة البرنامج التالية ، واكتب نتائج التنفيذ والمخرجات بفرض أنه قد تم تعريف المنظومة A والمتغير index بطريقة صحيحة.

```
For index := 1 to 4 do
A[index] := index ;
For index := 2 to 4 do
A[index] := A[index] + A[index - 1] ;
For index := 1 to 4 do
writeln ('A[', index : 1, ']= ' , A[index] : 3) ;
```

٥-٤ ما هي بالضبط مخرجات قطعة البرنامج التالية ؟

```
program four (output) ;
type
  index = 1 .. 5 ;
list : array [index] of integer ;
var
  k : list ;
```

```
i, j : integer ;
                                    begin
                                for i := 1 to 5 do
                                    begin
                                for j := 5 downto 1 do
                                    k[j] := i * j ;
                                k[i] := k[i] := k[i] - 10
                                    end ;
                                for i := 1 to 5 do
                            writeln ('k[' , i : 1 , '] = ' , k[i] : 1)
```

٥-٥ تتبع تنفيذ عبارات البرنامج التالي واكتب نتائج تنفيذها ومخرجات البرنامج.

```
                                program trace ;
                            var A : array [1 .. 6] of integer ;
                                i : integer ;
                                    begin
                                for i := 1 to 3 do
                                    begin
                                A[i] := 2 * i - 1 ;
                            A[i+3] := i * i - 2 ;
                                    end ;
                                for i := 1 to 3 do
                            A[i] := A[i+3] - A[i] ;
                                for i := 1 to 6 do
                    writeln ('for i = ' , i:2 , '  A = ' , A[i])
                                                        end.
```

٥-٦ صحح أي أخطاء في العبارات التالية عن المنظومات .

```
                        var A = array of [1 .. 8] of integer ;(a)
                    var New Array : array [1 .. 10] of real ;(b)
                        for j := 1 to n do score (j) := 0 ;(c)
                    var B , A : array [1..4 , 1..3] of real ;(d)
                    var A : array [1..5 , 1..6] of integer ;(e)
                        row , col : integer ;
                                    begin
                                for row := 1 to 6
                            for col := 1 to 5
                                begin
                        A [row,col] := 0 ;
                        B [row,col] := 0
                                end ;
                                    end ;
```

٥-٧ كُتب البرنامج التالي لحساب المجموع

$$\sum_{i=1}^{20} \frac{x_i}{y_i} = \frac{x_1}{y_1} + \frac{x_2}{y_2} + L + \frac{x_{20}}{y_{20}}$$

بفرض أن كلا من X , Y منظومة مكونة من عشرين عدد صحيح. صحح أي أخطاء في هذا البرنامج ليعطي المجموع المطلوب.

```
Program Series ;
Var i,sum,Y : integer ;
X : array [1..20] of integer ;
Begin
readln (X[i] , Y[i]) ;
sum := sum + x/y ;
writeln (sum)
End.
```

٨–٥ يقوم البرنامج التالي بقراءة أسماء وإدخالها في قائمة List. صحح أي أخطاء في البرنامج ، ثم اكتب عبارة While مكافئة لعبارة Repeat.

```
Program Names ;
Type
string15 = string [15] ;
NameList = array [1..10] of string15 ;
Var
List : NameList ;
Continue : char ;
Counter : integer ;
Begin
Continue := ' Y ' ; {initialize loop control variable}
Counter := 1 ; {initialize counter variable}
Repeat
write ('Enter a name for the list : ') ;
readdln (List[Counter]) ;
Counter := Counter + 1 ;
write ('Add more names ? Answer (Y or N) : ') ;
Readln (Continue)
Until
Continue <> 'Y'
End.
```

٩–٥ نفرض أن G مصفوفة مكونة من ستة صفوف وتسعة أعمدة. اكتب مجموعة عبارات لتنفيذ التعليمات التالية :

(أ) ضع مجموع عناصر العمود الثالث في SUM.

(ب) ضع مكان كل عنصر في الصف الثالث مجموع العنصرين المقابلين في الصفين الأول والثاني.

(ج) اجعل جميع عناصر الصف الرابع أصفارا.

٥-١٠ نفرض أن MAT مصفوفة مربعة فيها خمسة عشر صفا وخمسة عشر عمودا ، ونفرض أن DIAG منظومة من خمسة عشر عنصرا هي العناصر القطرية في المصفوفة MAT أي أن

DIAG(I) = MAT(I,I) ; I = 1,2,3,...,15

اكتب مجموعة عبارات حسابية لحساب كل من :

(أ) عناصر المنظومة DIAG

(ب) حاصل ضرب العناصر القطرية في المصفوفة MAT

$$PR = \prod_{I=1}^{15} MAT(I, I)$$

٥-١١ حوِّل أي عبارة FOR إلى عبارة WHILE في قطعة الباسكال التالية :

```
var
    t : ARRAY [1 .. n] of integer ;
    I,J,K : integer ;
begin
    t[1] := 1 ;
    for i := 2 to n do
        for j := i downto 1 do
            t[j] := t[j] + t[j-1] ;
end.
```

٥-١٢ أ) تتبع تنفيذ البرنامج التالي ، واكتب نتائج التتبع بفرض أن البيانات المدخلة هي : 6 2 0 8 4

ب) ماذا يعمل هذا البرنامج ؟ (أي ما وظيفته ؟)

```
Program ex ;
const
    n = 5 ;
type
    TAB = array [1..n] of integer ;
var
    v : TAB ;
    i,x,z,y : integer ;
    done : boolean ;
begin
```

```
                    for x := 1 to n do
                readln (v[x]) ;
                    for x := 2 to n do
                        begin
                done := false ;
                    z := x ;
            while (not done) and (z > 1) do
                        begin
                if v[z] < v[z-1] then
                        begin
                y := v[z] ;
            v[z] :=v[z-1] ;
              v[z-1] := y;
                        end
                else done := true ;
                        z := z-1 ;
                            end ;
                    for i := 1 to n do
              write (v[i]) ; writeln ;
                                end ;
                                                    end.
```

ثانيا : برامج

٥–١٣ اكتب برنامجا لقراءة مجموعة من درجات الحرارة

$T_1 , T_2 , T_3 , ... , T_{40}$

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 :

$$i = 1 , 2 , ... , 40 \qquad D_i = T_i - T_{av} ,$$

٥–١٤ اكتب برنامجا لقراءة قيم عناصر منظومة مكونة من مائة عنصر وحساب الجذر التربيعي لمجموع مربعات العناصر الفردية الترتيب أي لحساب :

$$S = \sqrt{A_1^2 + A_3^2 + A_5^2 + A_7^2+\cdots+A_{99}^2}$$

٥–١٥ نفرض أن A منظومة مكونة من أربعين عنصرا

$A_1 , A_2 , ... , A_i , ... , A_{40}$

اكتب برنامجا لقراءة قيم هذه العناصر وتعيين قيمة "i" حيث i هي ترتيب أول عنصر سالب في هذه المنظومة ، وإذا لم توجد أي قيمة سالبة فإن البرنامج يطبع القيمة صفرا (i = 0).

٥-١٦ تتكون المنظومة Y من خمسين عنصرا. اكتب برنامجا لقراءة قيمة عدد صحيح N وقيم أول N عنصر من المجموعة Y:

$$Y_1 , Y_2 , Y_3 , \dots , Y_N$$

ثم حساب (أ) القيمة المتوسطة (Average)

$$AVG = \sum_{i=1}^{N} Y_i / N$$

(ب) الانحراف القياسي (Standard Deviation)

$$SD = \sqrt{\frac{\sum_{i=1}^{N} Y_i^2}{N} - AVG^2}$$

٥-١٧ 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.

٥-١٨ اكتب برنامجا يقرأ قيم خمسة وأربعين عددا ثم يوجد ترتيب وقيمة العدد ذي أكبر قيمة مطلقة.

٥-١٩ 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.

٥-٢٠ نفرض أن كلا من L , M منظومة مكونة من أعداد صحيحة حيث تحتوي L على ٢٤ عنصرا مختلفا وتحتوي M على ٤٠ عنصرا مختلفا. اكتب برنامجا لقراءة قيم

عناصر المجموعتين ثم طباعة قيم الأعداد الصحيحة التي تظهر في كلا المجموعتين ، أي أن البرنامج يوجد تقاطع المجموعتين.

٥-٢١ نفرض أن Y المعرفة بالعلاقة

$$Y = F(x) = a_0 + a_1x + a_2x^2 + ... + a_ix^i + ... + a_nx^n$$

هي كثيرة حدود من درجة n في المتغير x.

اكتب برنامجا (أ) يقرأ قيم المعاملات

$$a_0 , a_1 , a_2 , ... , a_n$$

ويخزنها في منظومة ، بحيث أنه يمكن للبرنامج أن يعمل حسابات خاصة بكثيرات حدود لا تزيد درجة أي منها عن (١٥) خمس عشرة.

(ب) يقرأ قيمة للمتغير x ويحسب القيمة المقابلة لكثيرة الحدود.

٥-٢٢ نفرض أن خمسمائة شخص قدموا تبرعات في سبيل الله تعالى ، وأن قيم هذه التبرعات بالدينار قد وضعت كعناصر منظومة D :

$$D_1 , D_2 , ... , D_{500}$$

اكتب برنامجا لقراءة قيم هذه العناصر وحساب :

(أ) مجموع التبرعات SUM

(ب) أكبر قيمة تبرعات BIG

(ج) عدد الأشخاص N الذين تبرع الواحد منهم بأكثر من مائة دينار.

٥-٢٣ نفرض أن كلا من A , B منظومة مكونة من N عنصرا حيث $N \leq 20$. اكتب برنامجا :

(أ) يقرأ قيم كل من (i N

(ii عناصر كل من المنظومتين A , B

(iii متغير X

(ب) يحسب S مجموع حواصل ضرب A_iB_i العناصر المتقابلة في المنظومتين لقيم

i الفردية.

(ج) يحسب القيمة المتوسطة المركبة AV للقيم المطلقة لعناصر المنظومتين

والتي تعطى بالعلاقة

$$AV = \frac{\sum_{i=1}^{N}|A_i| + \sum_{i=1}^{N}|B_i|}{2N}$$

(د) يحسب قيمة XNEW باستخدام المعادلة

$$XNEW = X - P(X) / P'(X)$$

حيث

$$P(X) = A_1+A_2X+A_3X^2+...+A_{i+1}X^i+...+A_NX^{N-1} = \sum_{i=0}^{N-1} A_{i+1}X^i$$

$$P'(X) = A_2+2A_3X+3A_4X^2+...+iA_{i+1}X^{i-1}+...+(N-1)A_NX^{N-2} = \sum_{i=1}^{N-1} iA_{i+1}X^{i-1}$$

٢٤-٥ 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

(٥) 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).

٥-٢٥ يبين الشكل المرسوم خريطة سير عمليات برنامج يستخدم طريقة التبديل (Exchange) أو الترتيب الفقاعي (Bubble Sort) لترتيب عناصر منظومة ترتيبا تصاعديا.

المطلوب : ترجمة هذه الخريطة إلى برنامج لترتيب عناصر منظومة / مجموعة لا يزيد عدد عناصرها عن خمسين عنصرا ترتيبا تصاعديا.

ملاحظة : فكرة هذه الطريقة هي تنفيذ عدة مراحل : في كل مرحلة يقارن العنصر الأول في المجموعة مع العنصر الثاني ويوضع أصغرهما في الموضع الأول ، ثم يقارن العنصر الثاني – بعد ترتيب العنصرين الأوليين – مع الثالث ويوضع أصغرهما في الموضع الثاني ، وهكذا حتى ننتهي من كل المجموعة ، ونواصل في كل مرحلة ترتيب العناصر بهذه الكيفية إلى أن لا يحدث أي تبديل لأي عنصرين فتكون العناصر كلها حينئذ مرتبة تصاعديا (انظر المثال التوضيحي لهذه الطريقة).

٥-٢٦ المطلوب كتابة برنامج يحسب مجموع المتسلسلة $S = \sum_{i=1}^{100} T_i$ حيث حدود هذه المتسلسلة تُعَرَّف كما يلي :

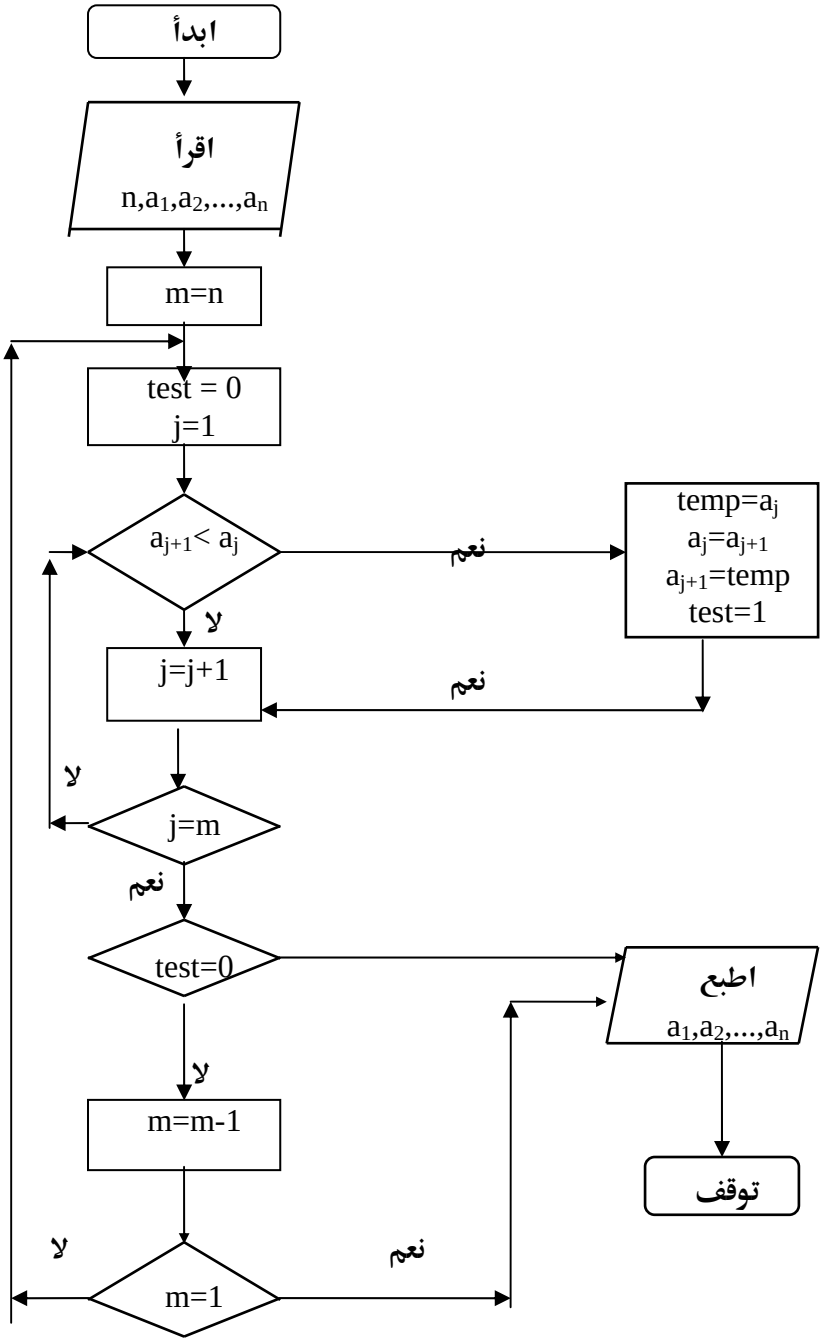
$$T_1 = 0$$
$$T_2 = 1$$
$$T_i = T_{i-1} + T_{i-2} \quad ; \quad i = 3 , 4 , 5 , \ldots$$

(ملاحظة : هذه المتسلسلة تعرف باسم متسلسلة فيبوناتشي Fibonacci).

٥-٢٧ 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.

2.0	-8.0
-17.0	7.0
25.0	3.0
3.0	25.0
7.0	-17.0
-8.0	2.0

المنظومة الأصلية المنظومة المعكوسة



طريقة الترتيب الفقاعي (المسألة ٥–٢٥)

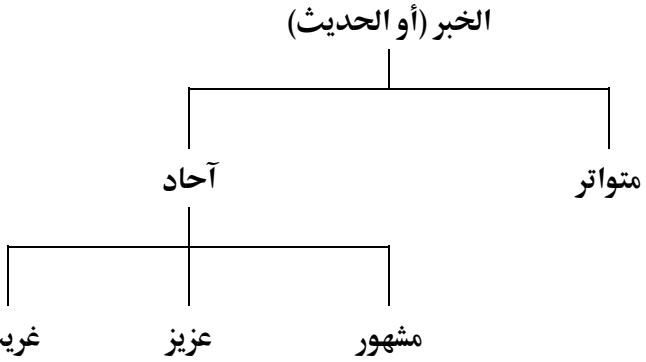
مثال :

المجموعة المعطاة	المرحلة الأولي	المرحلة الثانية	المرحلة الثالثة	المرحلة الرابعة	المرحلة الخامسة	المرحلة السادسة
٨	٦	٣	٣	٣	٢	٢
٦	٣	٥	٥	٢	٣	٣
٣	٥	٦	٢	٥	٤	٤
					.....	
٥	٨	٢	٦	٤	٥	٥
				.....		
١١	٢	٧	٤	٦	٦	٦
			.....			
٢	٧	٤	٧	٧	٧	٧
		.....				
٧	٤	٨	٨	٨	٨	٨
	.....					
٤	١١	١١	١١	١١	١١	١١

(تسمى الطريقة باسم الترتيب الفقاعي لأن العناصر الموجودة أسفل وضعها الصحيح تميل إلى التحرك لأعلى كالفقاعات. انظر مثلا إلى تحركات أصغر عنصر وهو الرقم ٢)

مثال على طريقة الترتيب الفقاعي (المسألة ٥-٢٥)

٥-٢٨ 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

(i) **يقرأ : – عدد طبقات السند : M (حيث $M \leq 6$).**

– الرقم التعريفي للحديث : ID

– عدد رواة الحديث في كل طبقة من طبقات السند :

$N_1\ N_2\ , \ldots , N_i , \ldots , N_M$

(ii) **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.

٥-٢٩ 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

٣٠-٥ 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.

٣١-٥ اكتب برنامجا لقراءة درجات طلاب الفصلين A , B في السؤال السابق ، وحساب متوسطي درجات الفصلين AvgA , AvgB ، وطباعة رسالة تفيد أي المتوسطين أعلى.

٣٢-٥ اكتب برنامجا يستخدم ثلاث منظومات (مجموعات مرتبة) 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 :

8811	90	C
8812	80	M
8813	75	B
8814	60	M
8815	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.

٣٣-٥ تقوم إحدى الجامعات بحساب المتوسطات الوزنية (weighted averages) للطلاب في نهاية كل فصل دراسي وذلك عن طريق قراءة ثلاث درجات , $score_1$ $score_2$, $score_3$ لكل طالب ، ومن ثم تقوم بحساب متوسطه الوزني.

اكتب برنامجا يقوم بقراءة عدد الطلاب n (حيث $n \leq 50$) ، والأوزان الاختبارية (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

$$score_3 \times score_2 + weight_3 \times score_1 + weight_2 \times \text{Weighted Average} = weight_1$$

وفي النهاية يطبع أكبر متوسط ومتوسط كل المتوسطات

شكل البيانات للطلاب كما يلي :

الأوزان الاختبارية : (test weights) 0.35 0.25 0.40

الدرجات الاختبارية ورقم الطالب : (test scores and ID) 100 76 88 1014

٣٤-٥ عمل مسح شامل للأسر الساكنة في إحدى المدن ، واشتمل سجل بيانات كل أسرة على : رقم تعريفي للأسرة عبارة عن عدد صحيح يتكون من ٤ أرقام ، والدخل السنوي للأسرة ، وعدد أفراد الأسرة.

اكتب برنامجا يقرأ هذه البيانات في ثلاث منظومات ، وذلك بعد قراءة العدد الإجمالي للأسر التي شملها المسح (افرض أن عدد هذه الأسر لا يزيد عن ٢٥).

أ) احسب متوسط دخل الأسر السنوي ، ثم أعد قائمة بالرقم التعريفي والدخل السنوي لكل أسرة يزيد دخلها عن الدخل المتوسط.

ب) احسب النسبة المئوية للأسر التي يقل دخلها عن حد الفقر P والذي يحسب بالعلاقة :

$$P = \$\ 6500.00 + \$\ 750.00 * (m-2)$$

حيث m تمثل عدد أفراد الأسرة. لاحظ من هذه العلاقة أن حد الفقر يعتمد على عدد أفراد الأسرة m ، وأن هذا الحد يزيد بزيادة m. شكل البيانات كما يلي :

عدد أفراد الأسرة	الدخل السنوي	الرقم التعريفي
4	12.180	1041
3	13.240	1062
2	19.800	1327

1483	22.458	8
1900	17.000	2
2112	18.125	7
2345	15.623	2
3210	3.200	6
3600	6.500	5
3601	11.970	2
4725	8.900	3
6217	10.000	2
9280	6.200	1

٣٥-٥ أعدت إجابات طلاب أحد فصول علم الحاسب على أحد الاختبارات ذي الأسئلة صح أم خطأ (T/F : true - false) **لإدخالها كبيانات لأحد البرامج. وبالنسبة لكل طالب تتكون المعلومات المتوفرة من رقم الطالب وإجاباته علي ١٠ من هذه الأسئلة** T/F. **والبيانات المتوفرة لدينا هي كما يلي :**

رقم الطالب	سلسلة الإجابات
0080	FTTFTFTTFT
0340	FTFTFTTTFF
0341	FTTFTTTTTT
0401	TTFFTFFTTT
0462	TTFTTTFFTF
0463	TTTTTTTTTT
0464	FTFFTFFTFT
0512	TFTFTFTFTF
0618	TTTFFTTFTF
0619	FFFFFFFFFF
0687	TFTTFTTFTF
0700	FTFFTTFFFT
0712	FTFTFTFTFT
0837	TFTFTTFTFT

اكتب برنامجا يقرأ أولا سلسلة الإجابات العشر الصحيحة (اعتبر أنها FTFFTFFTFT**) ثم يقرأ بعد هذا عدد الطلاب** N **(حيث** $N \leq 20$**) وبيانات كل طالب ويحسب ويخزن عدد الإجابات الصحيحة لكل طالب في منظومة ، ويخزن رقم الطالب** ID **في العنصر المقابل من منظومة أخرى. ثم يحدد البرنامج أفضل درجة** BEST**. ومن ثم يطبع جدولا من ثلاثة أعمدة تعرض رقم الطالب** ID **ودرجته وتقديره ، حيث التقدير يحسب كما يلي :**

- **إذا كانت الدرجة تساوي** Best **أو** (Best-1) **فإن التقدير** = A

- إذا كانت الدرجة تساوي (Best-2) أو (Best-3) فإن التقدير = C
- وما عدا ذلك فإن التقدير = F

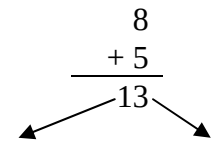
٥-٣٦ افرض أن حاسبك له قدرات محدودة إلى درجة كبيرة بحيث يستطيع قراءة وطباعة العدد الصحيح المكون من رقم واحد فقط ، وكذلك يستطيع جمع عددين صحيحين يتكون كل منهما من رقم عشري واحد فقط.

اكتب برنامجا يستطيع قراءة عددين صحيحين يتكون كل منهما من ٣٠ رقم على الأكثر ، ويجمع هذين العددين (أي يجمع هذه الأرقام المتقابلة) ويعرض النتيجة. اختبر البرنامج باستخدام أزواج أعداد ذوات أطوال مختلفة.

إرشاد : خزِّن العددين في منظومتين سعة كل منهما ٣٠ عنصرا ، بحيث تخزن رقما واحدا في العنصر الواحد من المنظومة. وإذا كان طول العدد أقل من ٣٠ رقما ، فأدخل عددا كافيا من الأصفار في البداية (على شمال العدد) لتجعل طول العدد ٣٠ رقما.

وستحتاج لعروة لجمع الأرقام في العناصر المتقابلة في المنظومتين مبتدئا بالموضع (المؤشر) رقم ٣٠. لا تنس أن تدخل في الاعتبار الرقم الحامل (الباقي) إذا كان هناك حامل. استخدم متغيرا منطقيا للدلالة على إذا ما كان مجموع آخر زوج من الأرقام أكبر من ٩ أم لا.

ملاحظة :



Carry — الباقي أو الحامل = 1

Sum — المجموع= 3

٥-٣٧ اشتمل كتاب "الإصابة في تمييز الصحابة" لابن حجر العسقلاني في جزئه الأخير على ذكر أسماء نحو ألف وخمسمائة من الصحابيات رضوان الله عليهن جميعا مع ذكر نبذة عن كل واحدة منهن.

نفرض أننا سنقوم بإدخال هذه الأسماء في الحاسب اسما اسما ، بحيث أن كل اسم يكتب على سطر مستقل ، ونفرض أن أي اسم لا يتجاوز ٢٠ رمزا (حرفا أو رمزا خاصا).

المطلوب :

(أ) كتابة برنامج بلغة الباسكال يقوم بما يلي :

١) قراءة الأسماء إلى أن تظهر العلامة * (والتي تعني انتهاء قائمة الأسماء) ، وتخزين هذه الأسماء في منظومة List من سلاسل رموز. أي أن العنصر List[I] يمثل اسم الصحابية التي ترتيبها i (في الإدخال).

٢) ترتيب أسماء المنظومة ترتيبا أبجديا.

٣) طباعة أسماء الصحابيات بعد الترتيب الأبجدي.

(ب) اختبار صحة البرنامج بإدخال نحو عشرة أسماء من الصحابيات مثل :

خديجة بنت خويلد عائشة بنت أبي بكر حفصة بنت عمر أم سلمه

سمية بنت خياط خولة بنت ثعلبة أسماء بنت أبي بكر نسيبة بنت كعب

أسماء بنت عميس أم سليم بنت ملحان (الرميصاء)

٥-٣٨ تحتوي كل بطاقة من مجموعة مكونة من خمسين بطاقة بيانات على رقم طالب ID ودرجاته في أربع مواد : الدراسات الإسلامية وعلم الحاسب والفيزياء والكيمياء. هذه الدرجات يمكن تخزينها في مصفوفة S مكونة من خمسين صفا وأربعة أعمدة (٥٠ × ٤) كما هو مبين بالشكل ، بحيث أن S(I,J) هي قيمة درجة الطالب رقم I في المادة رقم J.

Student	1 Islamic Studies	2 Computer Science	3 Physics	4 Chemistry
1	88	83	78	80
2	100	100	98	94
⋮	⋮	⋮	⋮	⋮
50	95	90	95	73

المطلوب كتابة برنامج يقوم بالتالي :

(أ) قراءة البيانات المذكورة.

(ب) حساب متوسط كل طالب (مجموع درجاته الأربعة مقسوما على ٤) وتخزين هذه المتوسطات في متجه جديد AVST مكون من خمسين عنصرا بحيث أن AVST(I) هو متوسط الطالب رقم I.

(ج) حساب متوسط كل مادة (مجموع درجات الخمسين طالبا في هذه المادة مقسوما على ٥٠) وتخزين هذه المتوسطات في متجه جديد AVSUB مكون من أربعة عناصر بحيث أن AVSUB(J) هو متوسط المادة رقم J.

(د) حساب المتوسط العام للفصل CLAV (إما بقسمة مجموع متوسطات الطلاب على خمسين ، أو بقسمة مجموع متوسطات المواد على أربعة).

٥-٣٩ نفرض أن لدينا ١٨٩٣ بطاقة بيانات ، تحتوي كل منها على :

(أ) رقم حديث من أحاديث رسول الله صلى الله عليه وسلم كما جاءت في كتاب الأحاديث (رياض الصالحين).

(ب) أربعة أرقام تشير إلى رواة هذا الحديث من بين الرواة : البخاري ومسلم وأبو داود والترمذي.

يمكن تخزين هذه المعلومات في مصفوفة ثنائية (١٨٩٣ × ٤) مكونة من ١٩٨٣ صف وأربعة أعمدة بحيث أن M(I,J) = 1 إذا كان الحديث رقم I قد رواه الراوي رقم J ، بينما M(I,J) = 0 إذا لم يرو هذا الراوي ذلك الحديث ، كما هو مبين في الجدول التالي:

Hadeeth Number	1 Al-Bukhari	2 Muslim	3 Abu-Dawood	4 Al-Tarmathi
1	1	1	0	0
2	1	1	0	0
⋮	⋮	⋮	⋮	⋮
157	0	0	1	1
⋮	⋮	⋮	⋮	⋮
1893	0	1	0	0

فمثلا يشير الصف الثاني في هذا الجدول إلى أن الحديث الثاني قد رواه البخاري ومسلم.

اكتب برنامجا يقرأ بطاقات البيانات ويوجد :

(أ) عدد الأحاديث – في هذا الكتاب – التي رواها كل واحد من الرواة الأربعة ، مع تخزين هذه الأعداد في منظومة N مكونة من أربعة عناصر ، بحيث يعطي العنصر N(I) عدد الأحاديث التي رواها المحدِّث رقم I.

٥-٤٠ نفرض أن A مصفوفة مكونة من سبعة صفوف وتسعة أعمدة. اكتب برنامجا يقرأ قيم عناصر هذه المصفوفة ، ثم يوجد :

(أ) عدد العناصر الموجبة ومجموع قيمها.

(ب) عدد العناصر السالبة ومجموع قيمها.

(ج) عدد العناصر الصفرية.

٥-٤١ نفرض أن A مصفوفة مربعة فيها تسعة صفوف وتسعة أعمدة. المطلوب كتابة برنامج

(أ) يحسب عناصر المصفوفة A وفقا للعلاقة :

$$a_{i,j} = \frac{1}{i+j-1} \quad \forall i,j$$

(ب) يوجد مجموع عناصر القطر الرئيسي.

(ج) يطبع العناصر الواقعة فوق القطر الرئيسي (أي عناصر المثلث الفوقي) بحيث تكون عناصر كل صف على سطر واحد مستقل.

٥-٤٢ نفرض أن MATH منظومة تشتمل على أرقام طلاب قسم الرياضيات وعددهم m طالبا ($m \le 600$) ، وأن SCIENCE منظومة تشتمل على أرقام الطلاب المتفوقين في كلية العلوم وعددهم n طالبا ($n \le 100$).

اكتب برنامجا يقرأ قيمتي m , n وكذلك قيم عناصر المنظومتين ، ثم يعطي أرقام الطلاب المتفوقين في قسم الرياضيات ، أي الأرقام التي تظهر في كلا المنظومتين (أي يوجد تقاطع المنظومتين).

الفصل السادس

البرامج الفرعية
Subprograms

الدوال والإجراءات
Functions and Procedures

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 طويلا كما أنه يستلزم حجز مواضع عديدة للتخزين في وحدة الذاكرة الرئيسية ، ولاختصار خطوات البرنامج وتحسين كفاءته وتوفير مواضع التخزين وأيضا لسهولة متابعة البرنامج وفهمه فإننا نقوم بفصل هذه العبارات – التي تنفذ خطوات هذه العملية المتكررة – خارج جزء عبارات البرنامج الرئيسي (Main program) ، ونضعها في جزء الإعلانات / التعريفات ، ونسميها برنامجا فرعيا Subprogram أو برنامجا مساعدا تابعا للبرنامج الرئيسي ، ونعطي هذا البرنامج الفرعي (أي العملية المراد تكرارها) اسما مميزا ، وكلما احتجنا تكرار هذه العملية في البرنامج الرئيسي ذكرنا الاسم المميز فقط دون كتابة كل خطوات العملـية ، أي أن هذه الخطوات لا تكتب إلا مرة واحدة فقط وذلك في البرنامج الفرعي وكلما احتجنا تنفيذها أشرنا إلى اسمها ، وعند كتابة البرنامج الفرعي نذكر مع اسمه أسماء بعض المتغيرات (وتسمى وسطاء parameters أو متغيرات وسيطة أو عمد arguments) والتي تظهر (إما كمدخلات أو مخرجات) عند تنفيذ عملية هذا البرنامج الفرعي ، وحين نشير في البرنامج الرئيسي إلى اسم البرنامج الفرعي نشير كذلك إلى القيم الفعلية أو الأسماء الفعلية التي نريدها لمتغيرات البرنامج الفرعي ، كما ستوضح ذلك بإذن الله تعالى الأمثلة التي ستذكر بعد قليل.

كذلك يمكن تسهيل حل بعض المسائل الطويلة بتقسيمها إلى مجموعة مسائل قصيرة يمكن حل بعضها ببرامج قصيرة (تسمى برامج فرعية) يكتبها الشخص نفسه أو يكتبها مبرمج آخر أو تكون جاهزة في مكتبة الحاسب (والتي تشتمل على عدة برامج جاهزة يحتاج إليها عدد كبير من الناس) ، وترتبط هذه البرامج الفرعية بالبرنامج الرئيسي لحل المسألة الأصلية الطويلة.

أنواع البرامج الفرعية :

سنذكر أولا نبذة مختصرة عن الأنواع المختلفة للبرامج الفرعية ثم نوضح بعد ذلك هذه الأنواع وكيفية استخدامها مع ذكر بعض الأمثلة التوضيحية.

تنقسم البرامج الفرعية عامة إلى قسمين رئيسيين :

أولا : البرامج الفرعية للدوال (البرامج الفرعية الدالية) :
Function Subprograms

ثانيا : البرامج الفرعية للإجراءات Procedure Subprograms :

ويستخدم البرنامج الفرعي لدالة عادة لإيجاد قيمة دالة معينة ، أما البرنامج الفرعي لإجراء فعادة يستخدم لإجراء عملية معينة (كترتيب عناصر مجموعة أو ضرب مصفوفة في متجه أو تبديل عنصرين) أو لإيجاد قيم عدة دوال (دالة أو دالتين أو أكثر).

وتكتب الإجراءات (Procedures) والدوال (functions) في الجزء الخاص بالإعلان عن الإجراءات والدوال ، وهو الذي يلي مباشرة الجزء الخاص بالإعلان عن المتغيرات (ويسبق مباشرة جسم البرنامج) ، وتوضع فاصلة منقوطة بين أي برنامجين فرعيين.

أولا : البرنامج الفرعي الدالي (Function Subprogram)

تنقسم البرامج الفرعية للدوال إلى نوعين :

(أ) البرنامج الفرعي لدالة قياسية (أو مكتبية أو معرفة سابقا) .
Standard (or Library or Predefined) Function

وهذا النوع يشمل البرامج التي تحسب قيم الدوال الرياضية القياسية التي تستخدم كثيرا مثل اللوغاريتم LN والجيب SIN والقيمة المطلقة ABS ، وقد سبق الحديث عن

استخدام هذه النوع من الدوال وذلك في الفصل الثاني (أساسيات لغة الباسكال) وذكرنا هناك مجموعة من أسماء هذه الدوال المعرفة. وتحتوي مكتبة الحاسب على عدة برامج (فرعية) معدة لحساب قيمة أي من هذه الدوال القياسية حين نذكر اسمها في البرنامج الرئيسي ، ولا داعي لتعريف هذه الدوال في البرنامج الرئيسي.

(ب) البرنامج الفرعي لدالة داخلية (Internal Function)

(أي دالة معرفة بالمستخدم) (User declared function)

وهذا النوع عبارة عن برنامج يعرِّف ويبيِّن كيفية حساب دالة ، وذلك داخل البرنامج الرئيسي وفي بدايته في الجزء الخاص بالإعلان عن الدوال (كما سبق ذكره) ، والصيغة العامة لهذا البرنامج الفرعي تشبه الصيغة العامة لبرنامج (رئيسي) بلغة الباسكال ، إلا أنه يبدأ في مطلعه بالكلمة المحجوزة FUNCTION يليها اسم الدالة ثم قائمة متغيراتها الوسيطة بين قوسين ، مع تحديد نوع هذه المتغيرات ونوع الدالة ، ثم توضع فاصلة منقوطة تفصل بين ما سبق (وهو الإعلان عن اسم الدالة ومتغيراتها) وما يلي ذلك وهو القسم الخاص بالإعلانات والتعريفات في البرنامج الفرعي وجسم (عبارات) البرنامج الفرعي ، أي أن الصيغة العامة هي :

FUNCTION name (v_1, v_2, ... , v_n : type) : Type ;

قسم الإعلانات والتعريفات المحلية

{Local declaration and definition part :
CONST , TYPE , VAR , ..}
BEGIN
.....
..... قسم عبارات البرنامج الفرعي الدالي
.....
.....
END;

حيث :

name : اسم تعريفي للدالة.

$v_1, v_2, \ldots, v_n$: قائمة المتغيرات الوسيطة (الشكلية) للدالة (formal parameter list).

type : نوع المتغيرات الوسيطة (النوع بسيط أو مركب / simple structured) وإذا لم تكن جميع المتغيرات من النوع نفسه ، فتوضع فواصل (commas) بين المتغيرات التي من النوع نفسه ، وفواصل منقوطة بين المتغيرات المختلفة الأنواع.
ويلاحظ أن جميع المتغيرات الوسيطة v_i متباينة (distinct) ، وتُعتبر جميعها بيانات (data) بالنسبة للدالة.

Type : اسم تعريفي يمثل نوع الدالة الناتجة ، ويجب أن يكون هذا النوع بسيطا [أي صحيحا أو حقيقيا أو منطقيا أو رمزيا أو تعدديا (معدِّدا) ومعرفا سابقا أو مدى جزئيا].
(INTEGER, REAL, BOOLEAN, CHARACTER, previously defined enumerated or subrange)

قسم عبارات البرنامج الفرعي يوضح العمليات التي تجري على المتغيرات الوسيطة ، والتي تؤدي إلى حساب قيمة الدالة. ويجب أن يشتمل هذا القسم على عبارة إسناد – واحدة على الأقل – تعطي الدالة قيمتها ، أي يكون اسم الدالة هو الطرف الأيسر في العبارة.

name := e ;

حيث name : اسم الدالة

e : تعبير حسابي.

وأما استخدام الدالة (أي استدعاؤها) في البرنامج الرئيسي فيتم بطريقة مشابهة لاستخدام أو استدعاء الدوال القياسية ، أي بكتابة اسم الدالة – في تعبير – مع وضع المتغيرات الوسيطة الفعلية (actual parameters / actual arguments) للدالة بين قوسيها بدلا من المتغيرات الوسيطة الشكلية (dummy / formal parameters). ويجب أن يكون المتغير الوسيط الفعلي تعبيرا ، أي ثابتا أو متغيرات بسيطا أو متغيرا مؤشرا أو دالة (إما الدالة نفسها أو دالة أخرى : قياسية أو دالة معرفة من قبل) أو تعبيرا حسابيا أو تعبيرا منطقيا. كما يجب ملاحظة :

(أ) أن يكون عدد المتغيرات الوسيطة الفعلية هو نفسه عدد المتغيرات الوسيطة الشكلية.

(ب) أن يكون ترتيب المتغيرات الوسيطة الفعلية هو نفسه ترتيب المتغيرات الوسيطة الشكلية المقابلة لها.

(ج) أن يكون نوع أي متغير وسيط فعلي هو نفسه نوع المتغير الوسيط الشكل المقابل ، وكذلك في حالة المنظومات أن يكون حجم المنظومة هو نفسه في المنظومة الفعلية والمنظومة الشكلية.

ملاحظة : إذا كان المتغير الوسيط الشكلي حقيقيا جاز أن يكون المتغير الوسيط الفعلي المقابل صحيحا ، ولكن العكس غير مسموح به.

وفيما يلي بعض الأمثلة التي توضح بإذن الله تعالى كتابة واستخدام البرامج الفرعية الدالية.

أمثلة على البرامج الفرعية الدالية Examples on Function Subprograms

أحيانا نحتاج لحساب قيمة دالة معينة – ليست من الدوال القياسية (المخزونة في مكتبة الحاسب) عدة مرات في البرنامج ، ويتطلب حساب قيمة الدالة في المرة الواحدة عدة خطوات أي تنفيذ عدة عبارات ، مثل حساب قيمة دالة المضروب factorial أي إيجاد عامليّ عدد صحيح (انظر المسألة رقم ١–٨) ، فقد نحتاج مثلا في البرنامج نفسه لحساب قيمة !N ، وقيمة !M وقيمة !(N-M) وذلك لقيمتين محددتين للمتغيرين N , M فبدلا من كتابة العبارات التي تحسب !N ، ثم إعادة كتابة العبارات نفسها مع تغيير كل N في هذه العبارات إلى M ، ثم إعادة كتابة العبارات نفسها مرة ثالثة مع وضع N-M بدلا من N أو M .. بدلا من ذلك نكتب برنامجا فرعيا Subprogram لدالة Factorial (K) تحسب مضروب أو عاملي عدد صحيح K أي تحسب !K حيث K متغير شكلي فقط وليس أيا من المتغيرات الفعلية أو التعابير الفعلية N, M, N-M المطلوب حساب مضروب كل منها .. ويكفي في البرنامج الرئيسي أن نذكر اسم الدالة Factorial التي عرفناها في البرنامج الفرعي وبعد اسم الدالة نضع بين قوسين المتغير أو التعبير المطلوب حساب مضروب قيمته

، أي حساب Factorial (N) أو Factorial (M) أو Factorial (N-M) وعند تنفيذ خطوات البرنامج الرئيسي فحينما يصل الدور إلى العبارة التي تشتمل على اسم الدالة كأن تشتمل مثلا على Factorial (N) فإن التحكم ينتقل من البرنامج الرئيسي إلى البرنامج الفرعي المسمَّى Factorial (K) وينفذ كل خطواته ولكن مع استخدام قيمة N بدلا من الرمز K ، وبعد أن ينتهي من تنفيذ البرنامج الفرعي يرجع أو يعود إلى البرنامج الرئيسي عند الموضع الذي تركه ليكمل تنفيذ خطواته بعد أن يكون قد حسب قيمة Factorial (N).

وإذا ورد اسم الدالة Factorial مرة أخرى عند تنفيذ خطوات البرنامج الرئيسي كأن يرد مثلا Factorial (M) فإن التحكم ينتقل مرة ثانية إلى البرنامج الفرعي لتنفيذ كل خطواته مرة أخرى ولكن مع استخدام M هذه المرة بدلا من K ، وبعد أن ينتهي من حساب Factorial (M) يرجع للبرنامج الرئيسي عند الموضع الذي تركه هذه المرة ليستكمل خطواته وهكذا .. وكلما ظهر اسم الدالة Factorial في البرنامج الرئيسي انتقل التحكم إلى البرنامج الفرعي لحساب القيمة المطلوبة للدالة..

مثال ٦-١ :

اكتب برنامجا فرعيا للدالة k! واستخدمه – كلما أمكن – في برنامج رئيسي يقرأ قيمتي n, m (حيث n > m) ويحسب قيمة كل من y , z حيث :

$$y = \frac{n!}{m!(n-m)!}$$

$$z = 1 + m + \frac{m^2}{2!} + \frac{m^3}{3!} + L + \frac{m^n}{n!}$$

الحل :

```
PROGRAM Compute (INPUT , OUTPUT) ;
    VAR   n , m , j : INTEGER ;
          y , z   : REAL ;
FUNCTION Factorial (k : INTEGER) : INTEGER ;
    VAR i , Fact : INTEGER ;
        BEGIN{Factorial}
          Fact := 1 ;
    FOR i := 2 TO k DO
  Fact := Fact * i ;
        Factorial := Fact
```

```
END {Factorial} ;
BEGIN
READLN (n , m) ;
y := Factorial (n) / Factorial (m) * Factorial (n-m)) ;
z := 1.0 + m ;
FOR j := 2 TO n DO
z := z + EXP (j * LN(m)) / Factorial (j) ;
WRITELN ('y = ' , y , 'z = ' , z)
END.
```

ملاحظة :

يمكن الحصول على قيمة Z بطريقة تستغرق وقتا أقل في التنفيذ حيث نحصل على قيمة كل حد من حدود متسلسلة Z من قيمة الحد الذي يسبقه مباشرة (انظر مثلا حل المسألة رقم ٤-٤٩ وحل المسألة رقم ٤-٥٠) ولكننا آثرنا إيجاد قيمة Z بالطريقة المذكورة في الحل لاستخدام البرنامج الفرعي كلما أمكن.

* * *

ليس من الضروري أن تكون الدالة التي نكتب لها برنامجا فرعيا دالة في متغير واحد فقط (مثل K في المثال السابق) وإنما يجوز أن تكون دالة في أكثر من متغير واحد كما في المثال التالي :

مثال ٦-٢ :

اكتب برنامجا فرعيا لدالة Larger توجد العدد الأكبر من عددين صحيحين i , j .

ثم اكتب برنامجا رئيسيا لقراءة قيم ثلاثة أعداد صحيحة t, m, n وإيجاد قيمة مربع أكبر المقدارين التاليين (بالاستعانة بالبرنامج الفرعي) :

$m + n^2$, $t + m - n$

الحل :

```
PROGRAM Test (INPUT , OUTPUT) ;
VAR t , m , n , k : INTEGER ;
FUNCTION Larger (i , j : INTEGER) : INTEGER ;
BEGIN
IF i > j THEN
Larger := i
ELSE
Larger := j
END ;
```

```
BEGIN
    READLN (t , m , n) ;
k := SQR (Larger (m + SQR (n) , t + m-n)) ;
    WRITELN ('k = ' , k)
END.
```

لاحظ في هذا المثال أننا احتجنا في قسم عبارات البرنامج الفرعي الدالي إلى عبارة واحدة فقط وهي عبارة IF. وكذلك البرنامج الفرعي في المثال التالي يحتاج إلى عبارة واحدة وهي عبارة إسناد حسابية.

مثال ٦-٣ :

اكتب نتيجة تنفيذ البرنامج التالي الذي يستخدم البرنامج الفرعي الدالي Avg.

```
PROGRAM Check (OUTPUT) ;
VAR x , y , u , v : INTEGER ;
FUNCTION Avg (x , y : INTEGER) : REAL ;
BEGIN
Avg := (x + y) / 2.0
END ;
BEGIN
x := 1 ; y := 2 ; u := 3 ; v := 4 ;
WRITELN ('a = ' , Avg (x , y) , ' b = ' , Avg (u , v)) ;
WRITELN ('c = ' , Avg (u + v , 3 * u - v)) ;
WRITELN ('d = ' , Avg (x + Avg (y , v) , 12))
END
```

الحل :

```
b = 3.5        a = 1.5
               c = 6.0
               d = 8.0
```

مثال ٦-٤ :

اكتب برنامجا فرعيا داليا لحساب قيمة الدالة

$$\text{Root}(A,B,C) = \frac{-B+\sqrt{B^2-4AC}}{2A}$$

الحل :

```
FUNCTION ROOT (A , B , C : REAL) : REAL ;
BEGIN
ROOT := (-B + SQRT(SQR(B) - 4.0 * A * C)) / (2*A)
END ;
```

وحين نستدعي (call) هذه الدالة في أي خطوة في البرنامج : أي حين نذكر اسمها مع أسماء المتغيرات الفعلية فإن قيم المتغيرات الفعلية تحل محل المتغيرات الشكلية المقابلة لها ، وتحسب قيمة التعبير وتعطي هذه القيمة للدالة.

مثال ٦-٥ :

فيما يلي أمثلة لبعض العبارات الحسابية لاستخدام الدالة ROOT التي سبق تعريفها في مثال ٦-٤ :

```
X1 := ROOT (12.4 , Y - Q , H) + 12.5 * SIN (Y)
VALUE := ROOT [I] , T [I + 1] , 0.087) * 3
G := H2 - ROOT (2.45 , A , B)
```

في العبارة الأولى يحل الثابت 12.4 محل المتغير الشكلي A ، ويحل التعبير Y - Q محل المتغير الشكلي B وكذلك يحل المتغير الفعلي H محل المتغير الشكلي C وبذلك فإن العبارة الحسابية الأولى تكافئ العبارة الحسابية

```
X1 := (-(Y-Q) + SQRT ((Y-Q) * (Y-Q) - 4.0 * 12.4 * H))/
* 12.4) + 12.5 * SIN (Y) ٢٠)
```

وبالمثل يمكن كتابة ما تكافئه كل من العبارتين الثانية والثالثة.

ويلاحظ أن المتغيرين (الفعليين) A,B في العبارة الثالثة لا علاقة لهما بالمتغيرين (الشكليين) A,B في تعريف الدالة ، وبعبارة أخرى فإن المتغيرات المستخدمة كمتغيرات وسيطة في تعريف الدالة هي مجرد متغيرات شكلية لا علاقة لها بأي متغيرات لها الأسماء نفسها والتي قد تظهر في موضع آخر في البرنامج.

مثال ٦-٦ :

اكتب برنامجا فرعيا لدالة منطقية F في ثلاثة متغيرات حقيقية a , b , c بحيث تكون الدالة صحيحة (true) عندما يكون المقدار (المميز) b^2 - 4ac موجبا ، وتكون خاطئة (false) ما عدا ذلك.

الحل :

```
FUNCTION F(a , b , c : REAL) : BOOLEAN ;
                    BEGIN
          F := b * b - 4.0 * a * c > 0.0
                    END ;
```

ملاحظة :

يقال للمتغيرات a , b , c في هذا المثال إنها متغيرات محلية (Local variables) خاصة بالدالة F أي متغيرات شكلية (dummy variables) ، أما إذا كتبنا الدالة F بدون هذه المتغيرات الوسيطة كما يلي :

```
FUNCTION F : BOOLEAN ;
    BEGIN
F := b * b - 4.0 * a * c > 0.0
    END.
```

فإن المتغيرات a , b , c في هذه الحالة تكون متغيرات عامة في البرنامج الرئيسي (global variables) أي ليست متغيرات شكلية. ويمكن أن تكون دالة ما – G مثلا – دالة في متغيرات بعضها محلي (Local) (أي شكلي) والبعض الآخر عام (global) ، كما في المثال التالي.

مثال ٦–٧ :

الدالة التالية مثال لدالة يظهر بها بعض المتغيرات التي لا يستخدم أي منها كمتغير وسيط للدالة ، أي لا يستخدم كمتغير شكلي ، أي أن صيغة الدالة قد يظهر فيها متغيرات عامة (global) (فعلية) بالإضافة إلى المتغيرات المحلية (Local) (الشكلية).

```
FUNCTION G(X : REAL) : REAL ;
    BEGIN
G := 1.0 / (EXP(5*LN(X)) * (EXP(1.432 / (T*X)) - 1.0))
    END ;
```

وهذه الدالة تقابل العلاقة الرياضية

$$G(X) = 1 / x^5 \ (e^{1.432/TX} - 1)$$

ونلاحظ أن X متغير شكلي حيث أنه متغير وسيط للدالة G بينما T ليس متغيرا شكليا وإنما هو متغير فعلي وهو المتغير T نفسه الذي يظهر في أي موضع آخر في البرنامج ، وأما لماذا لا نعتبر T متغيرا وسيطا آخر للدالة G فنكتبها في الصورة G(X , T) فلأننا سوف لا نحتاج لتعويض أي متغير أو أي قيمة بدلا من T ولكن المتغير T له قيمته الخاصة به أي أن T متغير عام (global) ، بينما X متغير محلي أو موضعي (local).

* * *

كما يمكن لدالة البرنامج الفرعي أن تكون دالة في عدة عناصر هي عناصر منظومة ، وفي هذه الحالة نذكر بين القوسين (الموضوعين يمين اسم الدالة) اسم المنظومة – وليكن A مثلا – واسما تعريفيا (Identifier) يدل على نوع هذا الاسم A ، أي يدل على أنه اسم منظومة ، كما يبين ذلك المثال التالي.

مثال ٦–٨ :

اكتب برنامجا فرعيا لدالة (Poly) تحسب مجموع حدود الحدودية التالية من الدرجة n :

$$\text{Poly} = a_1 + a_2x + a_3x^2 + \cdots + a_nx^{n-1}$$
$$= \sum_{i=1}^{n} a_i x^{i-1}$$

ثم استخدم هذا البرنامج الفرعي في برنامج رئيسي يقرأ قيمة عدد حقيقي t ، وعناصر منظومة $c_1 , c_2 , \ldots , c_8$ ، ثم يحسب قيمة المجموع التالي :

$$Z = \sum_{i=1}^{8} c_i t^{i-1}$$

الحل :

```
PROGRAM Example8 (INPUT , OUTPUT) ;
  TYPE Vector = ARRAY [1..8] OR REAL ;
  VAR i : INTEGER ; t , z : REAL ;
                C : Vector ;
FUNCTION Poly (n : integer ; x : REAL ; A : Vector) : REAL ;
                VAR i : INTEGER ;
                    BEGIN
                Poly := A [1] ;
              FOR i := 2 TO n DO
Poly := Poly + A[i] * EXP(i-1) * LN(x))
                    END ;
                  BEGIN
              READLN (t) ;
FOR i := 1 TO 8 DO READLN (C[i]) ;
              z := Poly (8 , t , C) ;
            WRITELN ('z = ' , z)
                  END.
```

ملاحظات :

١- لاحظ في هذا المثال أن i يظهر كمتغير محلي (في البرنامج الفرعي) وكذلك كمتغير عام وهذا جائز ، وهما يعتبران شيئين مختلفين.

٢- إذا استخدمنا عروة FOR في مجموعة عبارات البرنامج الفرعي الدالي ، فيجب الإعلان عن متغير التحكم في العروة (وهو i في هذا المثال) كمتغير محلي.

٣- نوع أي متغير وسيط شكلي وكذلك نوع الدالة يجب أن يحدده اسم تعريفي لنوع سبق الإعلان عنه ، فمثلا لا يجوز في المثال الحالي أن نعلن عن المنظومة A – التي هي متغير وسيط شكلي – بالصورة التالية :

```
FUNCTION Poly (n : INTEGER ; X : REAL ; A : ARRAY [1..8]
                OF REAL) : REAL ;
```

وإنما يمكن الإعلان عن المنظومة A بالطريقة التي اتبعت في حل المثال.

مثال ٦-٩ :

اكتب برنامجا فرعيا لدالة تحسب حاصل ضرب أول n عنصر من عناصر القطر الرئيسي (main diagonal elements) في مصفوفة مربعة A. افرض أن أي مصفوفة من المصفوفات التي سنستخدم لها هذا البرنامج الفرعي تشتمل على ثمانية صفوف وثمانية أعمدة.

الحل :

```
PROGRAM EX9 (INPUT , OUTPUT) ;
                                    TYPE
                .....
    Matrix = ARRAY [1..8 , 1..8] OF REAL ;
                .....
FUNCTION Prdiag (n : INTEGER ; A : Matrix) : REAL ;
                                VAR i : INTEGER ;
                                    BEGIN
                            Prdiag := A[1,1] ;
                        FOR i := 2 TO n DO
                Prdiag := Prdiag * A[i,i]
                                    END ;
```

مثال ٦-١٠ :

اكتب مجموعة عبارات من برنامج يقرأ قيم العناصر الموجودة في كل من الخمسة صفوف الأولى والخمسة أعمدة الأولى فقط من مصفوفة Z تتكون من ثمانية صفوف وثمانية أعمدة ، ويحسب الجذر التربيعي لحاصل ضرب أول خمسة عناصر في القطر الرئيسي ، وذلك باستخدام البرنامج الفرعي المذكور في المثال السابق.

(ملاحظة : بقية العناصر في المصفوفة إما أنه لا يهمنا معرفة قيمها في هذا الجزء من البرنامج أو أنه ربما ستحسب قيمها في جزء آخر من البرنامج).

الحل :

```
PROGRAM EX10 (INPUT , OUTPUT) ;
                                    .....          TYPE
        Matrix = ARRAY [1..8 , 1..8] OF REAL ;
                                    .....          VAR
                                      Z : Matrix ;
                                     SP : REAL ;
                                    .....
FUNCTION Prdiag (n : INTEGER ; A : Matrix) : REAL ;
                                    .....
                                          BEGIN
                                    .....
                                    .....          (مثل عبارات المثال السابق)
                                              END ;
                                                 BEGIN
                                          FOR i := 1 TO 5 DO
                                              BEGIN
                                    FOR j := 1 TO 5 DO
                                  READ (Z [i,j]) ;
                                            READLN
                                                END ;
                                  SP := SQRT (Prdiag (5,Z)) ;
                                    WRITELN ('SP = ' , SP)
                                                  END.
```

نلاحظ هنا أننا عندما استخدمنا دالة البرنامج الفرعي Prdiag عوضنا عن A بالرمز Z وكلاهما من النوع نفسه (منظومة ذات البعدين نفسيهما – ومكونة من أعداد حقيقية) وكذلك عوضنا عن n بالقيمة 5 وهما أيضا من نوع واحد (عدد صحيح).

وفي حالة ما إذا كنا نريد استخدام البرنامج الفرعي المذكور في مثال ٦-٩ لمصفوفات مربعة ولكنها ذات أبعاد مختلفة ، كأن تكون إحداهما (٥ × ٥) أي تشتمل على خمسة صفوف وخمسة أعمدة وأخرى (٨ × ٨) وثالثة (١٠ × ١٠) وهكذا .. ففي هذه الحالة نكتب البرنامج الفرعي بحيث يصلح لأي من هذه المصفوفات وذلك بأن تكون المصفوفة A في هذا البرنامج m x m مثلا حيث m متغير يوضع ضمن المتغيرات الوسيطة للبرنامج الفرعي كما في المثال التالي :

مثال ٦-١١ :

أعد حل مسألة ٦-٩ بعد حذف الفرض المذكور في المسألة أي بجعل البرنامج الفرعي عاما يصلح لاستخدامه لأي مصفوفة مربعة.

الحل :

نفرض أن المصفوفة A في هذا المثال m x m . والدالة Prdiag تحسب حاصل ضرب أول n عنصر من عناصر القطر الرئيسي.

```
FUNCTION Prdiag (n , m : INTEGER ; A : Matrix) : REAL ;
    VAR   i : INTEGER ;
    BEGIN
        .....
```

(كما في مثال ٦-٩)

```
    END :
```

وفيما يلي مجموعة عبارات تستخدم دالة هذا البرنامج الفرعي :

```
u := Prdiag (7 , 10 , Q) - SQRT (t + 12.5) ;
r3 := t2 / Prdiag (8 , 8 , DATA) ;
x := y + 4.0 * Prdiag (I - 1 , J , G) ;
```

حيث المصفوفة Q تحتوي على عشرة صفوف وعشرة أعمدة.

والمصفوفة DATA تحتوي على ثمانية صفوف وثمانية أعمدة.

والمصفوفة G تحتوي على J صف وكذلك J عمود.

واستخدمنا البرنامج الفرعي لإيجاد حاصل ضرب السبعة عناصر القطرية الرئيسية الأولى في Q ، وحاصل ضرب كل العناصر القطرية الرئيسية في المصفوفة DATA وحاصل ضرب العناصر من رقم ١ إلى رقم I-1 من العناصر القطرية الرئيسية في المصفوفة G.

مثال ٦-١٢ :

اكتب برنامجا فرعيا لدالة Big توجد قيمة العنصر ذي أكبر قيمة مطلقة من عناصر الصف رقم i في مصفوفة مربعة تشتمل على n صف و n عمود.

(فمثلا إذا كانت عناصر الصف رقم i هي : ٣ ، -١٢ ، ٧ ، ٩ ، -٤ فإن قيمة BIG تساوي -١٢).

الحل :

سنتبع في هذا الحل طريقة مماثلة للطريقة التي اتبعت عند حل مثال ٥-٦ لإيجاد أصغر عنصر.

```
FUNCTION Big (i , n : INTEGER ; A : Matrix) : REAL ;
{To find the element of the largest absolute value in the i-th
                    row of the n x n square matrix}
                                                    VAR
                                                j : INTEGER ;
                                                    BEGIN
                                              Big := A[i , 1] ;
                                            FOR j := 2 TO n DO
                                  IF ABS (A [i , j]) > ABS (Big) THEN
                                                Big := A [i , j]
                                                    END ;
```

ملاحظة :

إذا كان حجم أكبر مصفوفة تستخدم هذا البرنامج الفرعي هو ١٢ × ١٢ مثلا ، فإنه يمكننا كتابة ما يلي في قسم الإعلانات في البرنامج الرئيسي

```
CONST
    n = 12 ;
TYPE
Matrix = ARRAY [1..n , 1..n] OR REAL ;
```

وفيما يلي أمثلة لبعض العبارات التي نستخدم فيها هذا البرنامج الفرعي :

```
P := SQR (Big (3 , 10 , Q)) ;
R := SQRT (Big (4 , 8 , DATA)) / 4.0 ;
W := EXP (Big (k - 2 , 8 , DATA)) ;
```

الدوال الارتدادية (Recursive Functions)

يمكن لبرنامج فرعي (دالة أو إجراء) في لغة الباسكال أن يستدعي نفسه ، فيمكن عند تعريف دالة ما أن يظهر اسم الدالة في تعبير في إحدى عبارات البرنامج الفرعي الذي يعرِّف هذه الدالة نفسها ، ويسمى هذا ارتدادا (recursion) ويمكن أن يحدث هذا مع الدوال التي تعد هذه الظاهرة من طبيعتها ، مثل دالة المضروب (factorial) التي تعرف بالعلاقة

factorial (k) = k! = k x (k-1) x (k-2) x ... 3 x 2 x 1

حيث يمكن كتابة هذه العلاقة في الصورة الارتدادية التالية :

factorial (k) = k x factorial (k-1)

أي أنه يمكن كتابة برنامج فرعي لتعريف دالة المضروب باستخدام هذه العلاقة الارتدادية مع الشرط الابتدائي factorial (0) = 1 (انظر المسالة رقم ٦-٢٠) ، وعموما يمكن برمجة حل أي مسألة بدون استخدام العلاقات الارتدادية.

ثانيا : البرنامج الفرعي الإجرائي (البرنامج المساعد) (Procedure Subprogram)

عادة يستخدم البرنامج الفرعي الدالي لحساب قيمة واحدة فقط بينما نحتاج أحيانا لكتابة برنامج فرعي يعيد أكثر من قيمة واحدة أو يعيد مجموعة (مرتبة) من القيم ، كما نحتاج أحيانا لبرنامج فرعي لا يعيد أي قيمة ولكن يقوم بإجراء عملية ما مثل طباعة نتائج عملية سابقة أو تبديل بعض صفوف مصفوفة ما ، وفي مثل هذه الحالات يمكن أن نستخدم ما يسمى بالبرنامج الفرعي لإجراء ، وهو صورة أخرى من البرامج الفرعية ويشبه البرنامج الفرعي الدالي إلا أنه :

(أ) ليس له نوع (Type) معين (صحيح أو حقيقي مثلا) فهو من الممكن أن يحسب ويعيد قيما بعضها صحيح وبعضها حقيقي ، ولذلك فلا يذكر مع اسمه أي نوع.

(ب) القيم التي يعيدها تعطى للمتغيرات الفعلية وليس لاسم البرنامج الفرعي ، فهذا الاسم لا تعطى له قيمة وإنما هو مجرد اسم لتعريف البرنامج ، بينما اسم البرنامج الفرعي الدالي يشير إلى اسم الدالة التي تحسب قيمتها لتعاد للبرنامج الرئيسي.

(ج) عند استدعاء / مناداة (أي استخدام) البرنامج الفرعي الإجرائي يجب أن ينادى بعبارة نداء مستقلة وذلك عن طريق ذكر اسم البرنامج الفرعي وبين قوسين تكتب المتغيرات الوسيطة الفعلية ، ولا نكتب كلمة Procedure كما سنرى بإذن الله تعالى في الأمثلة التالية ، أما البرنامج الدالي فيمكن استدعاؤه بذكر اسمه (ضمن تعبير) في عبارة كعبارة إسناد أو عبارة طباعة.

ويبدأ البرنامج الفرعي الإجرائي بعبارة تعريف تشتمل على كلمة PROCEDURE يعقبها اسم البرنامج ثم بين قوسين نضع مدخلات البرنامج ومخرجاته (والترتيب هنا غير مهم) ، أما بقية البرنامج فهي كأي برنامج عادي ، ويجوز ألا يكون للبرنامج مدخلات أو مخرجات ، وإذا لم يشتمل على أي مدخلات وأي مخرجات فلا نضع القوسين ، أي أن الصيغة العامة للبرنامج الفرعي الإجرائي هي :

PROCEDURE name ((VAR) u_1 , u_2 , ... , u_n : Type) ;
قسم الإعلانات والتعريفات المحلية
{Local declaration and definition part :
CONST , TYPE , VAR , ... }
BEGIN
..... قسم عبارات
..... البرنامج الفرعي الإجرائي
.....
END ;

حيث :

u_1 , u_2 , ... أسماء متغيرات البرنامج الوسيطة الشكلية (dummy arguments) حيث بعضها مدخلات (input arguments) لإدخال بيانات للبرنامج ، وبعضها مخرجات (output arguments) لإعادة النتائج إلى البرنامج الرئيسي ، وأحيانا يستخدم نفس الرمز كمدخل ومخرج معا.

وقسم الإعلانات والتعريفات يستخدم لتوصيف وتحديد أنواع مدخلات البرنامج ومخرجاته وكذلك أنواع أي متغيرات أخرى تظهر في هذا البرنامج الفرعي.

أي أن الصيغة العامة للبرنامج الفرعي الإجرائي هي الصيغة العامة للبرنامج الفرعي الدالي نفسها ، مع وضع كلمة PROCDDURE بدلا من كلمة FUNCTION ، ودون

وضع النوع Type **لاسم الإجراء ، وأما بالنسبة لقائمة (الوسطاء) المتغيرات الوسيطة الشكلية (أو التمثيلية)** [dummy (formal) argument list] **فهناك – في حالة الإجراء – نوعان من الوسطاء :**

(أ) وسيط ذو قيمة (value parameter)

(ب) وسيط متغير (variable parameter)

وفيما يلي تعريف كل من هذين النوعين ، والفارق بينهما ، ثم نوضح هذا الفارق ببعض الأمثلة.

(أ) الوسيط ذو القيمة (value parameter)

هو وسيط خاص بالبيانات المدخلة فقط ، وهو متغير محلي (local) **للإجراء ، ولا يؤثر على المتغير الوسيط الفعلي حتى ولو كان له الاسم نفسه.**

(ب) الوسيط المتغير (variable parameter)

هو وسيط خاص بالمدخلات أو المخرجات ، ويتطابق مع المتغير الوسيط الفعلي ، أي أن أي حسابات تجري على الوسيط المتغير تؤثر مباشرة على المتغير الوسيط الفعلي. ويجب كتابة كلمة VAR **قبل أي وسيط متغير ، أو أي مجموعة وسطاء متغيرين من النوع نفسه مع وضع فاصلة بين أي وسيطين متتاليين.**

ملاحظة (١) :

إذا كان المتغير الوسيط الشكلي من النوع الأول – أي كان وسيطا ذا قيمة – فإن المتغير الوسيط الفعلي المقابل يمكن أن يكون تعبيرا (أي ثابتا أو متغيرا بسيطا أو متغيرا مؤشرا أو دالة ، أو تعبيرا حسابيا أو تعبيرا منطقيا).

أما إذا كان المتغير الوسيط الشكلي من النوع الثاني – أي وسيطا متغيرا (أي مسبوقا بكلمة VAR**) – فإن المتغير الوسيط الفعلي المقابل يجب أن يكون اسم متغير.**

ملاحظة (٢) :

المخرجات الشكلية (أي المتغيرات الوسيطة الشكلية التي تمثل مخرجات) يجب أن تكون من النوع الثاني ، أي يجب أن يكون كل منها وسيطا متغيرا (variable parameter).

أما المدخلات الشكلية (أي المتغيرات الوسيطة الشكلية التي تمثل مدخلات) فيمكن أن تكون من أي من النوعين ، ولكن الأفضل عموما أن تكون من النوع الأول ، أي يكون الواحد منها وسيطا ذا قيمة (value parameter) **، وذلك حتى يمكن أن نستبدل به متغيرا وسيطا فعليا ذا أي صورة نشاء من صور التعبير (أي ثابتا أو متغيرا بسيطا أو متغيرا مؤشرا أو دالة أو تعبيرا حسابيا أو تعبيرا منطقيا ، ويكون بالطبع من نوع المدخل الشكلي نفسه) ، أما إن كان من النوع الثاني (أي وسيطا متغيرا) فلا يمكن للمدخل الفعلي إلا أن يكون متغيرا بسيطا.**

مثال ٦-١٣ :

اكتب نتيجة تنفيذ البرنامج التالي :

```
PROGRAM Ex13 (INPUT , OUTPUT) ;
VAR x , z : INTEGER ;
PROCEDURE SUB13 (x : INTEGER ; VAR y : INTEGER) ;
                                    BEGIN
                                x := 2 * x ;
                                y := 2 * y ;
                            WRITELN (x , y)
                                    END ;
                                        BEGIN
                                    x := 5 ;
                                    z := 6 ;
                                SUB13 (x , z) ;
                            WRITELN (x , z)
                                    END.
```

الحل :

نلاحظ في هذا الإجراء (البرنامج الفرعي) أن x **وسيط ذو قيمة بينما** y **وسيط متغير ، ولذلك فحين يستدعي البرنامج الرئيسي** Ex13 **البرنامج الفرعي** sub13 **ويسند قيمتي**

المتغيرين الوسيطين الفعليين x , z **(أي يسند القيمتين** 6 , 5**) للمتغيرين الوسيطين الشكليين** x , y **على الترتيب ، يحدث ما يلي :**

أولا : تنفيذ البرنامج الفرعي

x := 2 * 5 = 10

y := 2 * 6 = 12

ولذلك فإن أمر الطباعة في البرنامج الفرعي يؤدي إلى طباعة العددين :

12 10

ثانيا : عند العودة من البرنامج الفرعي إلى البرنامج الرئيسي :

نظرا لأن x **وسيط ذو قيمة ، فإن المتغير الفعلي** x **(وإن كان له اسم المتغير الشكلي** x **نفسه) لا تتأثر قيمته الأصلية (أي لا تتغير من** 5 **إلى** 10**) ويبقى محتفظا بقيمته الأصلية** 5**.**

ونظرا لأن المتغير الشكلي y **وسيط متغير فإن قيمته تؤثر على المتغير الفعلي المقابل** z **، ولذلك تصبح قيمة** z **بعد تنفيذ البرنامج الفرعي** 12 **(بدلا من القيمة** 6 **قبل تنفيذ البرنامج الفرعي).**

ولذلك فإن أمر الطباعة WRITELN (x , z) **في البرنامج الرئيسي يؤدي إلى طباعة العددين:**

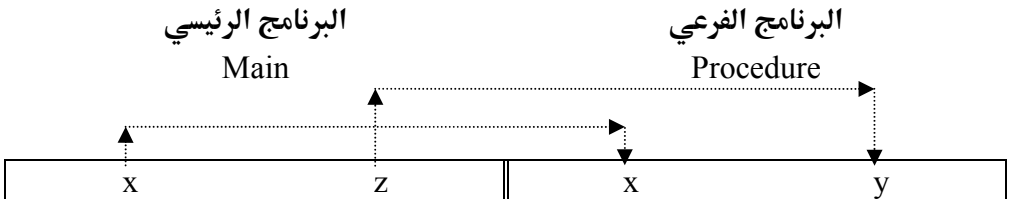
12 5

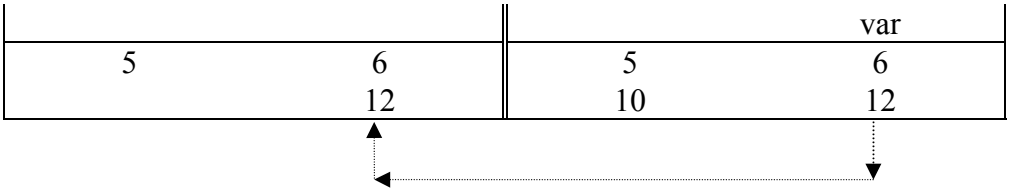
أي أن حل المثال يتلخص في طباعة السطرين :

12 10

12 5

ملاحظة (١) : يمكننا تلخيص القيم المتتابعة لمتغيرات البرنامج الرئيسي ومتغيرات البرنامج الفرعي في الجدول التالي :





ملاحظة (٢) : من البرامج الفرعية الإجرائية التي استخدمناها فعلا سابقا البرامج الفرعية :

READ , READLN , WRITE , WRITELN

فمثلا حين نكتب عبارة الطباعة :

WRITELN ('TAS = ' , TAS , 'Z = ' , 0.025 * TAS) ;

فإن WRITELN **هو اسم الإجراء ، والمتغيرات الوسيطة الفعلية هي :**

الثابت الرمزي (string constant) : 'TAS = '

المتغير (variable) : TAS

الثابت الرمزي : 'Z = '

التعبير الحسابي (arith. expression) : 0.025 * TAS

[انظر الملاحظة المكتوبة قبل هذا المثال (مثال ٦-١٣) مباشرة وبعد تعريف كل من الوسيط ذي القيمة والوسيط المتغير].

مثال ٦-١٤ :

تتبع البرنامج التالي وأوجد مخرجاته :

```
Program  main (input , output) ;
var  x , y , z : integer ;
Q(var a , b : integer ; c : integer) ;  Procedure
var  d : integer ;
begin
d := a+10 ;
a := a+b ;
b := 2*b+c ;
c := c+d ;
writeln ('a , b , c , d = ' , a , b , c , d)
end ;
begin
x := 10 ; y := 20 ; z := 30 ;
writeln ('x , y , z = ' , x , y , z) ;
Q (x,y,z) ;
writeln ('x , y , z = ' , x , y , z) ;
end.
```

الحل : أولا : التتبع : بدء تنفيذ البرنامج الرئيسي :

$z = 30 \quad y = 20 , \quad x = 10 ,$

الطباعة : انظر المخرجات فيما بعد.

استدعاء البرنامج الفرعي :

$$Q(x , y , z) \Rightarrow Q(10 , 20 , 30) \Rightarrow$$
$$(z \leftrightarrow c) = 30 , \quad (y \leftrightarrow b) = 20 , \quad (x \leftrightarrow a) = 10,$$
$$d = a + 10 = 10 + 10 = 20$$
$$a = a + b = 10 + 20 = 30$$
$$b = 2 * b + c = 2 * 20 + 30 = 70$$
$$c = c + d = 30 + 20 = 50$$

الطباعة : انظر المخرجات فيما بعد .

وعند الرجوع من البرنامج الفرعي إلى البرنامج الرئيسي :

*) **يأخـذ المتغـيران** x , y **– على الـترتيب – قيمـتي الوسيطين** a , b **نفسيهما (أي** 30 , 70**) لأن** a , b **وسيطان متغيران ، أي أن** $x = 30 , y = 70$.

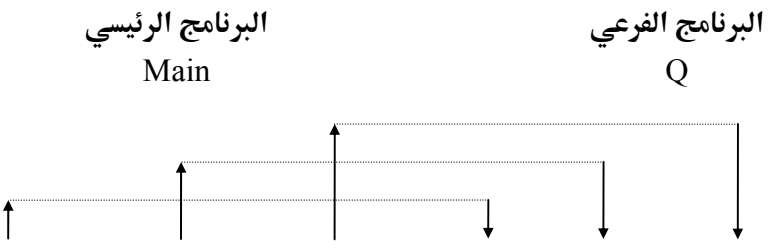
*) **بينما يحتفظ المتغـير** z **بقيمـته الأصلية قبل استدعاء وتنفيذ البرنامج الفرعي (أي** $z = 30$**) ، ولا يأخذ قيمة الوسيط** c **(أي القيمة** 50**) لأن** c **وسيط ذو قيمة.**

ثم الطباعة : انظر المخرجات فيما بعد.

ثانيا : المخرجات :

x , y , z =	10	20	30	
a , b , c , d =	30	70	50	20
x , y , z =	30	70	30	

ملاحظة : يمكننا تلخيص القيم المتتابعة لمتغيرات البرنامج الرئيسي ومتغيرات البرنامج الفرعي في الجدول التالي :



x	y	z	a var	b var	c	d
10 30	20 70	30	10 30	20 70	30 50	20

لاحظ أن d متغير محلي (local) في البرنامج الفرعي Q وليس وسيطا من وسطاء هذا البرنامج الفرعي ، وهو كذلك ليس من متغيرات البرنامج الرئيسي (غير معرف فيه).

مثال ٦-١٥ :

اكتب برنامجا فرعيا لإجراء يقوم بتبديل عنصرين حقيقيين x , y . وبيِّن كيفية استخدامه في برنامج رئيسي يقرأ قيمتي عددين حقيقيين a , b ويقوم بتبديل قيمتيهما.

الحل :

```
PROGRAM Example15 (INPUT , OUTPUT) ;
                                    VAR
                                        a , b : REAL ;
PROCEDURE Exchange (VAR x , y : REAL) ;
                                    VAR
                                        Temp : REAL ;
                                    BEGIN {Exchange}
                                        Temp := x ;
                                           x := y ;
                                        y := Temp
                                               END ;
                                     BEGIN {main pgm}
                                      REDLN (a , b) ;
                                    Exchange (a , b) ;
                          WRITELN('a = ' , a , 'b = ' , b) ;
                                    END.
```

ملاحظة :

في هذا المثال كل من a , b متغير عام (شامل) ، وكل من x , y متغير محلي (في البرنامج الفرعي) ، ومتغير وسيط شكلي (أي نعوض عنه بمتغير وسيط فعلي) ، بينما Temp متغير محلي ولكنه ليس متغيرا وسيطا (نعوض عنه بأي متغير آخر).

مثال ٦-١٦ :

نفرض أن X منظومة مكونة من n عنصر .

$$x_1 , x_2 , \ldots , x_n$$

اكتب إجراء Statistics يقوم بحساب :

(أ) المتوسط الحسابي

$$Ave = \left(\sum_{i=1}^{n} x_i \right) / n$$

(ب) الانحراف القياسي

$$SD = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - (Ave)^2}$$

ثم استخدم الإجراء في برنامج رئيسي يقرأ عناصر منظومة Y من ٣٠٠ عنصر ، ويحسب متوسطها الحسابي Average **وانحرافها القياسي** Stddev.

الحل :

```
PROGRAM Example16 (INPUT , OUTPUT) ;
                                    TYPE
                  Vector = ARRAY [1..300] OF REAL ;
                                     VAR
                                  i : INTEGER ;
                          Average , Stddev : REAL ;
                                  Y : Vector ;
: Vector; n : INTEGER; VAR Ave,  PROCEDURE Statistics (X
SD : REAL) ;
                                          VAR
                                    S1 , S2 : REAL ;
                                    i    : INTEGER ;
                                          BEGIN
                                 S1 := 0.0 ; S2 := 0.0 ;
                                 FOR i := 1 TO n DO
                                      BEGIN
                              S1 := S1 + X[i] ;
                        S2 := S2 + X[i] * X[i]
                                       END ;
                                      Ave := S1 / n
```

```
        SD := SQRT(S2 / n - Ave *Ave)
    END ;
BEGIN
    FOR i := 1 TO 300 DO
    READLN (Y[i]) ;
Statistics (Y , 300 , Average , Stddev) ;
WRITELN('Average = ', Average, 'Standard Deviation = ' ,
    Stddev)
END.
```

ملاحظة :

كل من S2 , S1 متغير محلي داخل البرنامج الفرعي ، وليس متغيرا وسيطا (أي ليس متغيرا شكليا نعوض عنه بمتغير فعلي في البرنامج الرئيسي).

مثال ٦-١٧ : (تعديل لمسألة مثال ٦-١٢).

اكتب برنامجا فرعيا يوجد القيمة Big وهي قيمة العنصر ذي أكبر قيمة مطلقة من عناصر الصف رقم i في مصفوفة مربعة A تشتمل على n صف وn عمود ، ويوجد كذلك قيمة k وهي ترتيب هذا العنصر في ذلك الصف ، أي يوجد رقم العمود الذي يقع فيه هذا العنصر.

الحل :

المطلوب الأول هنا هو المطلوب نفسه في مثال ٦-١٢ ، ثم أضيف هنا المطلوب الثاني وهو إيجاد قيمة k ، وفي حالة وجود أكثر من مطلوب واحد يفضل كتابة برنامج فرعي إجرائي.

```
PROCEDURE Large (i , n : INTEGER ; A : Matrix ;
VAR Big : REAL ; VAR k : INTEGER) ;
                                    : A , n , i
{input parameters
                                    : Big , k
output parameters
                                    : given n x n square matrix
A
                                    : given row number
i
                                    : required element of largest absolute value
Big
                                    : required column number of this element}
k
VAR
    j : INTEGER ;
BEGIN
    Big := A[i , 1] ;
```

```
                    k := 1 ;
                FOR j := 2 TO n DO
        IF ABS (A[i , j) > ABS (Big) THEN
                        BEGIN
            Big := A[i , j] ;
                    k := j
                        END
END ;
```

حينما فرضنا في بداية الحل أن العنصر المطلوب (Big) هو العنصر الأول في الصف وضعنا k := 1 ، ثم عند مقارنة قيمة Big المطلقة بالقيم المطلقة لبقية العناصر في الصف إذا وجدنا عنصرا قيمته المطلقة أكبر ، غيَّرنا قيمة Big لتصبح هي قيمة هذا العنصر وبالتالي تتغير قيمة k تبعا لترتيب هذا العنصر ولذلك كتبنا العبارة k := j.

عند استعمال هذا البرنامج الفرعي نناديه بعبارة كالتالية

```
Large (3 , 10 , Q , Big , k) ;
```

حيث عوضنا اسم المصفوفة Q بدلا من اسم المصفوفة A وطلبنا تعويض القيمتين i = 3 , n = 10 ، ثم البرنامج الفرعي يعطينا قيمة كل من k , Big. وليس من الضروري أن نتقيد بالاسمين k , Big ولكن يمكن اختيار أي اسمين آخرين مثل t , Great على الترتيب فنكتب :

```
Large (3 , 10 , Q , Great , t)
```

فإذا أردنا حساب مربع قيمة العنصر ذي أكبر قيمة مطلقة ، فإننا نكتب عبارة كالتالية بعد عبارة النداء السابقة :

```
P := SQR (Great) ;
```

(انظر للمقارنة أول عبارة كتبناها بعد حل مثال ٦-١٢ لشرح كيفية استخدام البرنامج الفرعي الدالي في ذلك المثال).

ويمكن كذلك عند كتابة عبارة النداء أن نكتب بعض التعابير لتحل قيمها محل المدخلات الشكلية للبرنامج الفرعي ، كأن نكتب مثلا :

```
Large (t-2 , 8 , DATA , Big , k) ;
Large (m+1 , t , Array3 , z , j2) ;
```

من أهم مميزات البرامج الفرعية عامة أن البرنامج الفرعي يعد وحدة مستقلة من حيث أنه يمكن كتابته وتصحيح أخطائه واختبار صحته أي استعماله لإعطاء نتائج صحيحة وذلك كله دون الرجوع إلى البرامج الرئيسية التي ستستخدمه فعلا.

ومن الممكن أن يشتمل حل مسألة معينة على كتابة واستعمال عدة برامج فرعية ، فإذا تأكدنا من صحة كل برنامج فرعي على حدة أصبح من السهل بإذن الله اختبار صحة البرنامج الرئيسي الكامل.

ولاختبار صحة عمل برنامج فرعي نكتب برنامجا رئيسيا قصيرا (وهذا غير البرنامج الرئيسي الفعلي الذي سنحتاج لتنفيذه إلى البرنامج الفرعي) وكل وظيفة هذا البرنامج القصير أن يقرأ بعض البيانات وينادي البرنامج الفرعي ثم يطبع النتائج التي يعيدها البرنامج الفرعي .. والمثال التالي يوضح بإذن الله تعالى هذه الفكرة.

مثال ٦-١٨ :

اكتب برنامجا فرعيا إجرائيا لإيجاد الجزء الصحيح intprt والجزء الكسري frcprt لعدد حقيقي A . ثم اختبر صحة البرنامج بكتابة برنامج رئيسي يقرأ قيم ثلاثة أعداد x,y,z, ويوجد لكل منها الجزء الصحيح والجزء الكسري وذلك بالاستعانة بالبرنامج الفرعي.

(مثلا إذا كان العدد يساوي 12.25 فإن الجزء الصحيح هو 12 والجزء الكسري 0.25).

الحل :

```
PROGRAM Test (INPUT , OUTPUT) ;
{For testing the procedure : Parts}
VAR
   X , Y , Z , FX , FY , FZ : REAL ;
   IX , IY , IZ             : INTEGER ;
PROCEDURE Parts (A : REAL : VAR intprt : INTEGER ;
   VAR frcprt : REAL ;
{For breaking a real number A into its integral and fractional parts}
   {input arguments : A (the given number) ;
      output arguments : intprt , frcprt}
         BEGIN
   intprt := TRUNC (A) ;
      frcprt := A - intprt
         END ;
BEGIN
   READLN (X , Y , Z) ;
   Parts (X , IX , FX) ;
   Parts (Y , IY , FY) ;
   Parts (Z , IZ , FZ)  ;
```

```
WRITELN ('X = ', X, 'INTEG.PART=' IX,'FRAC.PART=' , FX) ;
WRITELN ('Y = ', Y, 'INTEG.PART=' IY,'FRAC.PART=' , FY) ;
WRITELN ('Z = ', Z, 'INTEG.PART=' IZ,'FRAC.PART=' , FZ) ;
END.
```

وكمثال على نتائج هذا البرنامج نحصل على المخرجات التالية :

```
INTEG.PART = 15 FRAC.PART = 0.2400000  X = 15.2400000
INTEG.PART = -3 FRAC.PART = - .9480000  Y =  -3.9480000
INTEG.PART =  0 FRAC.PART =  0.7000000  Z =   0.7000000
```

* * *

سبق أن عرفنا أنه عند كتابة برنامج فرعي لحساب قيمة دالة واحدة فقط فإننا عادة نكتب برنامجا فرعيا داليا ، أما إذا أردنا حساب أكثر من قيمة واحدة فعادة نكتب برنامجا فرعيا إجرائيا ، على أنه يمكن أيضا في حالة حساب قيمة دالة واحدة فقط كتابة برنامج فرعي إجرائي ، كما يتضح من المثال التالي :

مثال ٦-١٩ :

أعد حل مسألة مثال ٦-١ ولكن بكتابة برنامج فرعي إجرائي بدلا من برنامج فرعي دالي.

الحل :

```
PROGRAM Computation (INPUT , OUTPUT) ;
VAR   n , m , y , i , Fn , Fm , Fnm , Fi : INTEGER ;
                                  : REAL ;     Z
                PROCEDURE Factor (k : INTEGER ;
  VAR   Fact : INTEGER) ;
                                      VAR   i : INTEGER ;
                                                  BEGIN
                                            Fact := 1 ;
                                  FOR i := 2 TO k DO
                                  Fact := Fact * i
                                                  END ;
                                                      BEGIN
                                          READLN (n , m) ;
                                            Factor (n , Fn) ;
                                           Factor (m , Fm) ;
                                        Factor (n-m , Fnm) ;
                                    Y := Fn / (Fm * Fnm) ;
                                          Z := 1.0 + m ;
```

```
                FOR i := 2 TO n DO
                    BEGIN
                Factor (i , Fi) ;
    Z := Z + EXP(i * LN (m)) / Fi
                    END ;
    WRITELN ('Y = ' , Y , 'Z = ' , Z)
                END.
```

مثال ٦-٢٠ :

اكتب برنامجا إجرائيا فرعيا يقوم بجمع كل عنصرين متقابلين في منظومتين X , Y تحتوي كل منهما على n عنصرا ويضع العناصر الناتجة في منظومة Z.

ثم اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومتين A , B تحتوي كل منهما على أربعة عناصر ، ثم ينادي البرنامج الفرعي ليخزن مجموع عناصر A ,B المتقابلة في منظومة جديدة SUMAB ... وكذلك يطبع البرنامج الرئيسي عناصر المنظومات الثلاث A,B, SUMAB .

الحل :

```
PROGRAM VectorSum (INPUT , OUTPUT) ;
                TYPE
    Vector = ARRAY [1..4] OR REAL ;
                VAR
        A , B , SUMAB : Vector ;
        : INTEGER ;     i
PROCEDURE Addvectors (X , Y : Vctor ; n : INTEGER ;
    VAR Z : Vector) ;
                {Input arguments :
        X , Y : arrays to be summed
            n  : size of arrays
(or number of top elements to be added)
                Output arguments :
    Z : array containing the sum X + Y}
                VAR
            i : INTEGER ;
                BEGIN
        FOR i := 1 TO n DO
    Z [i] := X[i] + Y[i] ;
                END ;
                BEGIN
        FOR i := 1 TO 4 DO
```

```
                    READLN (A[i]) ;
                      FOR i := 1 TO 4 DO
                    READLN (B[i]) ;
              Addvectors (A , B , 4 , SUMAB) ;
SUMAB') ;          B          WRITELN ('A
                      FOR i := 1 TO 4 DO
          WRITELN (A[i] , B[i] , SUMAB[i])
END.
```

وفيما يلي مثال لمدخلات ومخرجات البرنامج الرئيسي : إذا فرضنا أن المدخلات هي :

A:	4.5	12.0	15.4	20.0
B:	6.5	4.0	18.3	12.7

فتكون المخرجات هي :

A	B	SUMAB
4.5000000	6.5000000	11.0000000
12.0000000	4.0000000	16.0000000
15.4000000	18.300000	33.7000000
20.0000000	12.700000	32.7000000

نلاحظ في هذا المثال أن قراءة البيانات تمت في البرنامج الرئيسي ثم العمليات على هذه البيانات (الجمع في هذا المثال) قام بها برنامج فرعي ثم قام البرنامج الرئيسي بطباعة النتائج. وكثيرا ما يتبع هذا الأسلوب عند كتابة البرامج وإن كان من الممكن أيضا أن يقوم برنامج فرعي مستقل بقراءة البيانات وبرنامج فرعي آخر بطباعة النتائج.

كما نلاحظ أن عدد عناصر كل من المنظومات الفعلية كان ثابتا محددا (= ٤) من بداية البرنامج ، فإذا أردنا أن يكون البرنامج أعم من ذلك بحيث يقرأ أولا قيمة m تمثل عدد عناصر كل من المنظومتين A , B وفرضنا أن m لا تزيد عن ١٢ ($m \leq 12$) فإننا نعدل البرنامج الرئيسي ليصبح كما في حل المثال التالي بينما يبقى البرنامج الفرعي كما هو :

مثال ٦-٢١ :

أعد حل مسألة المثال السابق بعد فرض أن كلا من المنظومتين A , B تشتمل على عدد m من العناصر حيث $m \leq 12$ ومع فرض أن البرنامج الرئيسي سيقرأ أولا قيمة m.

الحل :

البرنامج الفرعي يبقى كما هو والبرنامج الرئيسي يصبح كما يلي :

```
PROGRAM VectorSum(INPUT , OUTPUT) ;
                                    CONST
                                  max = 12 ;
                                    TYPE
                Vector = ARRAY [1..max] OF REAL ;
                                    VAR
                          A , B , SUMAB : Vector ;
                           i : INTEGER ;
                                  PROCEDURE
                                  .....
                                  .....  كما هو في المثال السابق
                                  .....
                                  END;
                                  BEGIN
                                  READLN (m) ;
                                FOR i := 1 TO m DO
                              READLN (A[i]) ;
                                FOR i := 1 TO m DO
                              READLN (B[i]) ;
                        Addvectors (A , B , m , SUMAB) ;
                    WRITELN ('A      B      SUMAB') ;
                                FOR i := 1 TO m DO
              WRITELN (A[i] , B[i] , SUMAB [i])
                                  END.
```

مثال ٦-٢٢ :

نفرض أن A مصفوفة مكونة من n صف و t عمود.

اكتب برنامجا فرعيا يقوم بتبديل عناصر الصف رقم k مع العناصر المقابلة في الصف رقم m.

الحل :

هذا مثال لبرنامج فرعي لا يقوم بإجراء أي عمليات حسابية ولذلك فنكتبه برنامجا إجرائيا فرعيا كما يلي :

```
PROCEDURE Interchange (var A : matrix ; n , t , k , m : INTEGER) ;
                {input parameters : A , n , t , k , m ,
                No output parameters
```

```
    (or A : can be considered both
  input & output parameter}
                              VAR
   j : INTEGER ; temp : REAL ;
                            BEGIN
                  FOR j := 1 TO t DO
                        BEGIN
      temp := A[k,j] ;
    A[k,j] := A[m,j] ;
      A[m,j] := temp
                              END
                          END ;
```

مثال ٦-٢٣ :

اكتب برنامجا يقرأ قيم عناصر مصفوفة مكونة من عشرة صفوف وثمانية أعمدة ، ثم يقوم بتبديل عناصر كل صف فردي الترتيب مع عناصر الصف الذي يليه (أي تبديل الصف الأول مع الثاني ، والثالث مع الرابع .. وهكذا ، حتى التاسع مع العاشر) ، ثم يقوم البرنامج بطباعة عناصر المصفوفة الناتجة بعد الانتهاء من كل عمليات التبديل.

الحل :

نحتاج في هذا البرنامج لإجراء عملية تبديل صف مع صف خمس مرات ولذلك ننادي البرنامج الفرعي - المكتوب في حل المثال السابق - خمس مرات باستخدام عبارة FOR كما هو مبين في الحل التالي :

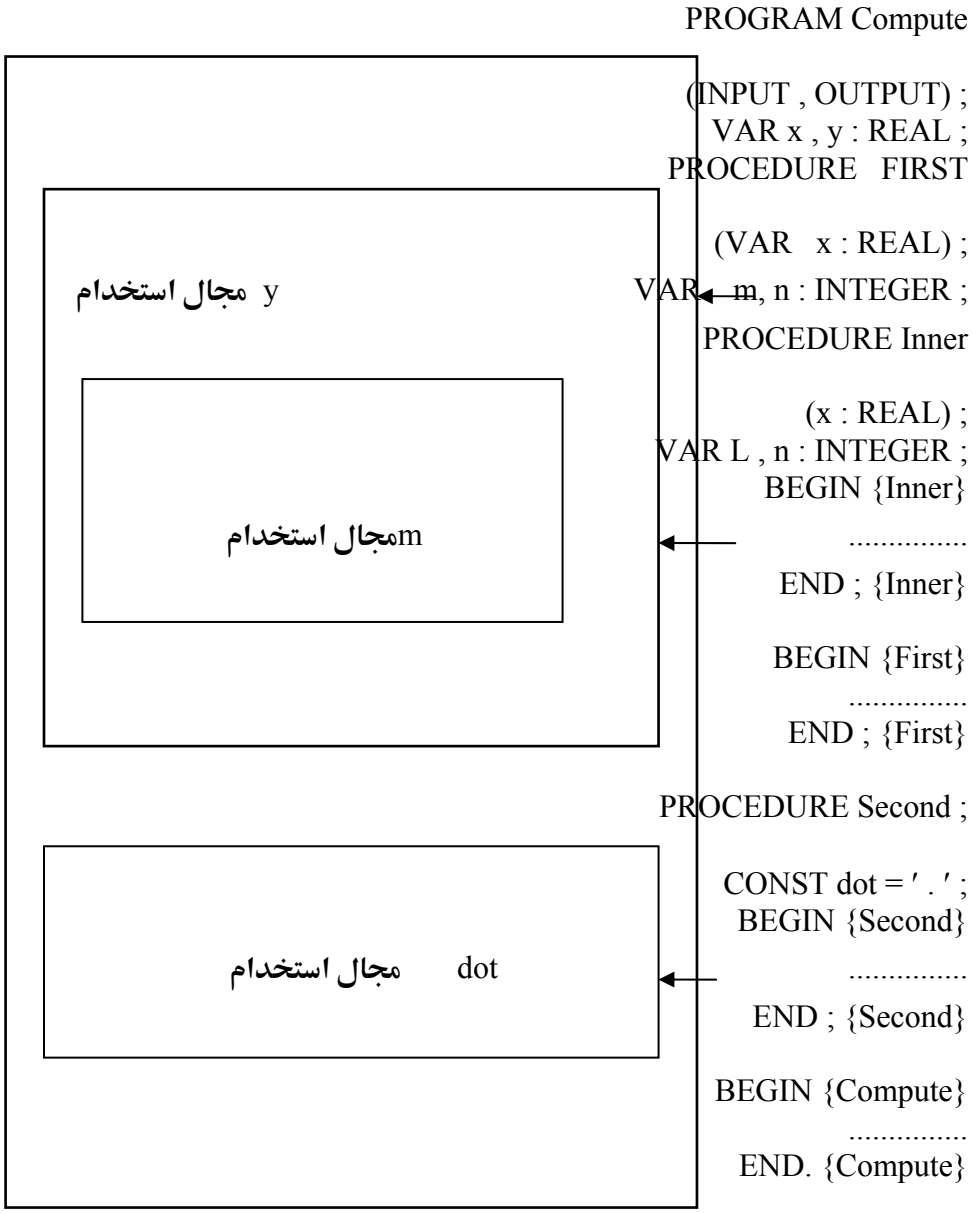
```
PROGRAM MatrixReorder (INPUT , OUTPUT) ;
                              TYPE
  Matrix = ARRAY [1..10 , 1..8] OF REAL ;
                              VAR
                  i , j : INTEGER ;
                      A   : Matrix ;
  PROCEDURE Interchange (.....)
                            BEGIN
                          .............
كما في المثال السابق      .............
                          .............
                              END ;
                            BEGIN
```

```
                    FOR i := 1 TO 10 DO
                        BEGIN
                    READLN ;
                FOR j := 1 TO 8 DO
            READ (A[i , j])
                        END ;
                    FOR j := 1 TO 5 DO
                        BEGIN
                    i := 2 * j-1 ;
        Interchange (A , 10 , 8 , i , i+1)
                        END ;
        WRITELN ('Matrix after interchanges') ;
                    FOR i := 1 TO 10 DO
                        BEGIN
                    WRITELN ;
                FOR j := 1 TO 8 DO
            WRITE (A [i , j])
                        END
                                                END.
```

البرامج الفرعية المتداخلة Nested Subprograms

من الممكن أن يُعَرَّف برنامج فرعي داخل برنامج فرعي آخر بطريقة مماثلة لتعريف برنامج فرعي داخل برنامج رئيسي ، فمثلا الشكل التالي يبين برنامجا رئيسيا compute **يعرِّف داخله برنامجين فرعيين (إجرائيين)** First , Second **وداخل البرنامج الفرعي** First **نعرِّف برنامجا فرعيا آخر** Inner**.**

وفي أي برنامج (رئيسي أو فرعي) يسمى الجزء الخاص بالتعريفات والإعلانات مع الجزء الخاص بالعبارات قالبا (block) **. وفي أي برنامج (فرعي) تعد الثوابت والأنواع والمتغيرات والدوال والإجراءات المعلنة (في جزء التعريفات والإعلانات) في هذا البرنامج (الفرعي) محلية بالنسبة له ، ولا معنى لها جميعا إلا في القالب المقابل [أي في جزء الإعلانات وجزء العبارات في هذا البرنامج (الفرعي)] ، وهذا القالب يسمى مجالها** (scope) **أو مجال استخدامها ، ولذلك فلا يمكن استخدام أي برنامج فرعي (دالة أو إجراء) إلا في داخل القالب المقابل الذي عُرِّف فيه ، أما خارج هذا القالب فلا يجوز استخدام هذا البرنامج الفرعي. أي أن :**



قاعدة المجال (Scope Rule) **في لغة الباسكال :**

(١) **أي اسم تعريفي** (identifier) **يمكن أن يستخدم (أي يشار إليه** referenced) **فقط في داخل القالب الذي عُرِّف فيه.**

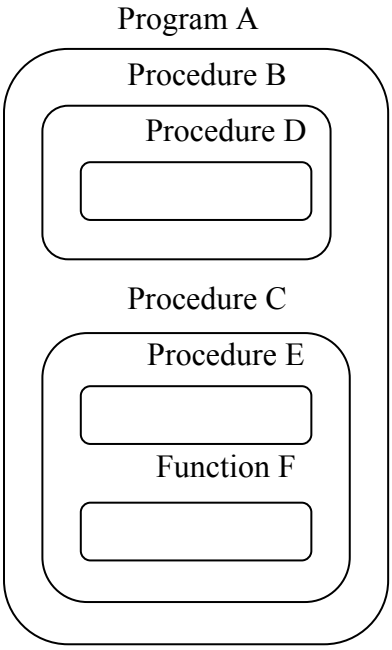
(٢) ويجب أن يعلن عن (أي يُعَرَّف) الاسم التعريفي قبل أول استخدام (إشارة) إليه.
ومجال أي اسم تعريفي هو القالب الذي عُرِّف فيه (أُعلن فيه عن) هذا الاسم.
فمثلا في الشكل السابق : كل مستطيل مرسوم يمثل قالبا لبرنامج فرعي (إجراء) أو للبرنامج الرئيسي ، واسم القالب مكتوب فوقه مباشرة. ونلاحظ ما يلي :

- مجال الثابت dot هو قالب الإجراء Second ، ولذلك لا يستخدم هذا الثابت إلا في الإجراء Second.
- حيث إن الإجراء Inner يقع داخل الإجراء First ، فإن مجال أي اسم تعريفي معلن عنه في الإجراء First سيشمل قالب الإجراء Inner. ولذلك فإن أي اسم تعريفي معلن عنه في الإجراء First (مثل المتغير m) يمكن الإشارة إليه (أي يمكن استخدامه) في أي من الإجراءين.
- وحيث أن جميع الإجراءات تقع داخل قالب البرنامج الرئيسي ، فإن أي اسم تعريفي معلن عنه في البرنامج الرئيسي يمكن استخدامه في أي موضع في البرنامج ولهذا تسمى متغيرات البرنامج الرئيسي متغيرات شاملة أو عامة (global variables).
- وقد يعلن عن الاسم التعريفي نفسه في أكثر من موضع ، فمثلا في الشكل السابق أعلن عن x كمتغير شامل في البرنامج الرئيسي ، وكمتغير وسيط شكلي في كل من الإجراءين First , Inner. وعموما في لغة الباسكال إذا كان هناك أكثر من إعلان واحد لاسم تعريفي ، فإن الإعلان الذي يُعمل به (أي الذي يسري مفعوله) هو أقرب إعلان في مجال يشتمل على موضع استخدام الاسم التعريفي ، ودائما يكون هذا الإعلان هو الإعلان المحلي local declaration إذا وُجد.
- إذا لم يعلن محليا عن الاسم التعريفي فإننا نأخذ بالإعلان الموجود في قالب خارجي يحتوي على موضع استخدام الاسم. فمثلا إذا استخدمنا x في عبارات الإجراء Second فإننا نأخذ بالإعلان عن x كمتغير شامل وذلك لأن الإجراء Second لا يوجد إلا داخل قالب البرنامج الرئيسي فقط.
- إذا أشرنا إلى الاسم التعريفي n في الإجراء Inner أو في الإجراء First ، فإننا نستخدم الإعلان المحلي المقابل عن n.

- إذا أشرنا إلى m في الإجراء Inner حيث لم يعلن عن m محليا فإننا نستخدم الإعلان عن m في الإجراء First.
- إذا أشرنا إلى m في البرنامج الرئيسي أو في الإجراء Second فإن هذا يؤدي إلى خطأ تركيبي (syntax error) حيث أننا استخدمنا اسما تعريفيا لم يعلن عنه (identifier not declared).
- نظرا لأن أسماء الإجراءات أسماء تعريفية فإن قاعدة المجال في لغة الباسكال (والمذكورة آنفا) تحدد لنا المواضع التي يمكن عندها استدعاء إجراء ما. فمثلا نظرا لأن الإجراء First معلن عنه في البرنامج الرئيسي فيمكن أن نستدعيه من أي موضع. ونظرا لأن الإجراء Inner معلن عنه في الإجراء First ، فيمكن أن يستدعى فقط بالإجراء First (أو الإجراء Inner نفسه). وإذا حاولنا استدعاءه من الإجراء Second أو من البرنامج الرئيسي نتج خطأ تركيبي.
- الإجراء Second (وهو معلن عنه أيضا في البرنامج الرئيسي) يمكن أن يستدعي الإجراء First ، ولكن العكس غير صحيح ، وذلك لأن أي اسم تعريفي يجب أن يعلن عنه قبل الإشارة إليه ، والإعلان عن الإجراء Second يأتي بعد قسم عبارات (أي بعد جسم) الإجراء First.
- بناء على ما سبق فإنه يمكن للإجراء First فقط أن يستدعى الإجراء Inner ، فإن كان من الضروري أن يستدعي الإجراء Second أيضا الإجراء Inner لزم أن نجعل الإجراء Inner عاما أو شاملا (global) معلنا عنه في البرنامج الرئيسي بدلا من الإعلان عنه في البرنامج الفرعي First. وحيث أنه في لغة الباسكال يجب أن يعلن عن الاسم التعريفي قبل الإشارة إليه ، فيجب في هذه الحالة أن يكون الإجراء Inner هو أول إجراء يعلن عنه في البرنامج الرئيسي ، قبل كل من الإجراءين First , Second.

مثال ٦-٢٤ :

نفرض أن لدينا برنامجا A يحتوي على إجراءين B , C (انظر الشكل). ويحتوي الإجراء B على إجراء آخر D ، بينما يحتوي الإجراء C على إجراء E ودالة F كما هو مبين بالشكل.



أ) مجال (scope) أي اسم تعريفي (identifier) معرف في البرنامج A هو (اختر الإجابة الصحيحة) :

(a) A , B , C
(b) A , B , C , D , E , F

(i) A فقط

(ii) A , B , C , D , E ولكن ليس F.

ب) (i) هل يمكن للإجراء D أن يستدعي الإجراء B ؟ (نعم / لا)

(ii) هل يمكن للدالة F أن تستدعي الإجراء B ؟ (نعم / لا)

(iii) هل يمكن للإجراء E أن يستدعي الدالة F ؟ (نعم / لا)

الحل :

أ) (ii) (متغيرات البرنامج الرئيسي متغيرات شاملة أو عامة global).

ب) (i) نعم.

(ii) نعم. B معلن عنه في البرنامج الرئيسي A فيمكن أن يستدعي من أي موضع (لاحق).

(iii) لا . أي اسم تعريفي يجب أن يعلن عنه قبل الإشارة إليه ، فلا يمكن استدعاء

الدالة F قبل الإعلان عنها.

الإعلان الأَوَّلي (FORWARD Declaration)

رأينا أنه يمكن لبرنامج فرعي (دالة أو إجراء) أن يستدعي برنامجا فرعيا (دالة أو إجراء) آخر بشرط أن يكون هذا البرنامج الآخر قد سبق الإعلان عنه ، فإذا كان لدينا برنامجان فرعيان A , B وأردنا أن يستدعي كل منهما الآخر ، فإننا نستخدم ما يسمى بالإعلان الأولي (FORWARD Declaration) بالصورة المبينة في المثال التالي :

```
PROCEDURE B (VAR Y : REAL) ; FORWARD ;
PROCEDURE A (VAR X : REAL) ;
                .....
        قسم إعلانات الإجراء A
                .....
                BEGIN
                .....
        استدعاء الإجراء B
                .....
                END ;
PROCEDURE B ;
                .....
        قسم إعلانات الإجراء B
                .....
                BEGIN
                .....
        استدعاء الإجراء A
                .....
                END ;
```

ونلاحظ ما يلي :

- الإعلان الأولي عن الإجراء B يتكون فقط من اسم الإجراء ومتغيراته الوسيطة الشكلية.
- ثم تأتي بعد ذلك كلمة FORWARD بين فاصلتين منقوطتين.

- **ثم يأتي الإجراء A كاملا بقسميه : قسم الإعلانات وقسم العبارات ، ويتضمن استدعاء الإجراء B.**
- **ثم يأتي بعد ذلك الإجراء B بقسميه ، ولكن بدون قائمة متغيراته الوسيطة الشكلية (والتي ظهرت في الإعلان الأولي عن الإجراء B). والإجراء B يتضمن استدعاء الإجراء A.**
- **يطلق على الإجراءين A , B اللذين يستدعي كل منهما الآخر أنهما : ارتداديان تبادليا** (mutually recursive).
- **يمكن لأي من الإجراءين A , B أن يكون دالة ، أي أن البرنامجين الفرعيين اللذين يستدعي كل منهما الآخر يمكن أن يكونا إجراءين ، أو دالتين أو أن يكون أحدهما إجراء والآخر دالة.**

تمرينات رقم ٦

أولا : تدريبات

٦-١ نفرض أن لدينا الإعلانات والتعريفات التالية :

```
var
    X, Y, Z : REAL ;
    M, N : INTEGER ;
procedure MESSAGE (var A,B : REAL ; X : INTEGER) ;
```

ضع خطا تحت أي وسيط فعلي خاطئ في كل مما يلي :

```
MESSAGE (X, Y, Z)(One
MESSAGE (X, Y, 8)(Two
MESSAGE (M, Y, N)(Three
MESSAGE (X, Y, M + N)(Four
MESSAGE (Y + Z, Y - Z, M)(Five
```

٦-٢ نفرض أن لدينا الإعلانات التالية في برنامج رئيسي :

```
VAR   A, B, C : INTEGER ;
      H, H1  : CHAR ;
PROCEDURE P (VAR X : INTEGER; Y : INTEGER ; CH : CHAR);
```

اكتشف أي أخطاء في العبارات التالية التي تستدعي (call) الإجراء P.

```
P(A, B/2, 'C') ;(One
P(A+1, B, H) ;(Two
P(A, A*B, CH) ;(Three
P(A, B, H, H1) ;(Four
```

٦-٣ ما هي القيمة التي تخزن في x بعد تنفيذ مجموعة العبارات التالية ؟

```
var
    x : real ;
function num (b, a : real) : real ;
                    begin
        num := a/(b * 2.0)
                    end ;
                begin {main}
    x := num (0.1 , 3.0) ;
                    :
                    :
                    end.
```

٦-٤ تتبع تنفيذ كل من البرامج التالية ، واكتب نتائج تنفيذ جميع عبارات البرنامج ، وكذلك اكتب مخرجاته.

أ)

```
PROGRAM TEST (INPUT , OUTPUT) ;
VAR A , B , C : INTEGER
PROCEDURE TRACE (D : INTEGER ; VAR E : INTEGER) ;
BEGIN
WRITELN ('A = ', A, 'D = ', D, 'E = ', E) ;
D := D + 1 ;
E := E + D ;
A := 2 * D ;
WRITELN ('A = ', A, 'D = ', D, 'E = ', E)
END ;
BEGIN
TRACE (B , C) ;  C := 3 ;  B := 2 ;  A := 1 ;
WRITELN ('A = ', A, 'B = ', B, 'C = ', C)
END.
```

ب)

```
program exam ;
var a , b , c : ingeger ;
procedure subprogram (d : integer ; var e : integer ; c : integer) ;
var a : integer ;
a := c + 1 ;  begin
e := d + c -1 ;
c := c*2
end ; {subprogram}
begin  {main}
a := 2 ;
b := 4 ;
c := 6 ;
subprogram (b , c , a) ;
writeln (a , b , c) ;
end. {exam}
```

ج)

```
program sample (input , output) ;
var A , B : integer ;
procedure show ( x : integer ; var y : integer) ;
var C : integer ;
begin
C := X ;
Y := Y + C ;
```

```
                X := 2*Y ;
            writeln (X , Y)
                        end ;
                begin {sample}
                    A := 2 ;
                      repeat
            B := A + 1
            A := A + 1
                until A > 5 ;
             writeln (A , B) ;
              show (A , B) ;
              writeln (A , B)
                    end. {sample}
```

د)

```
                PROGRAM Demo (OUTPUT) ;
            {Formal Versus Actual parameters}
                VAR a , b , x , y : INTEGER ;
        FUNCTION F(x , y : INTEGER) : INTEGER ;
{F = (square of first argument) + (second argument)}
                        BEGIN {F}
                    F := SQR(x) + y
                            END {F} ;
                BEGIN {Demo (main)}
                    a := 2 ; b := 1 ; x := 4 ; y := 3 ;
    ' , F(b,a) : 3) ;       WRITELN (F(a,b) : 3 , '
    ' , F(b,b) : 3) ;       WRITELN (F(a,a) : 3 , '
    ' , F(y,x) : 3) ;       WRITELN (F(x,y) : 3 , '
    ' , F(y,y) : 3) ;       WRITELN (F(x,x) : 3 , '
                WRITELN (F(x+y, -15*y) : 3)  ;
            WRITELN (F(-10, F(x,x) + 2) : 3)
                            END {Demo}.
```

هـ)

```
        PROGRAM ValueVariableParameters (OUTPUT) ;
            {Value Versus Variable parameters}
                VAR x , b : INTEGER ;
PROCEDURE Update (x : INTEGER ; VAR y : INTEGER) ;
                            BEGIN
                    x := x*x ;    y := y*y ;
            WRITELN (x : 3 , '   ' , y : 3)
                        END ; {update}
```

```
BEGIN {main}
x := 3 ;   b := 2 ;
Update (x , b) ; WRITELN (x : 3 , ' ' , b : 3) ;
Update (b , x) ; WRITELN (x : 3 , ' ' , b : 3)
{ValueVariableParameters}        END.
```

و)

```
program exam ;
var number1 , number2 : integer ;
procedure daa (var first, second : integer) ;
begin
first := 2 * first ;
second := 1 + second ;
{daa}  end ;
procedure aad (var first , second : integer) ;
begin
first := 1 + first ;
second := 2 * second ;
end ; {aad}
begin {main}
number2 := 4 ;          number1 := 3 ;
daa (number1 , number2) ;
writeln (number1 , number2) ;
aad (number1 , number2) ;
writeln (number1 , number2) ;
end.
```

٦-٥ لكل قالب (block) من القوالب التالية اذكر قائمة بالأسماء التعريفية التي يمكن أن تظهر في هذا القالب.

```
program a
procedure b
procedure c
procedure d
begin
:
:
:
end ;
begin
:
:
:
```

```
        end ;
    begin
        :
        :
        :
    end ;
procedure e
procedure f
    begin
        :
        :
        :
    end ;
procedure g
    begin
        :
        :
        :
    end ;
    begin
        :
        :
        :
    end ;
begin
    :
    :
    :
end.
```

٦-٦ لكل قالب من القوالب التالية اذكر قائمة بالأسماء التعريفية التي يمكن أن تظهر في هذا القالب.

```
program a
procedure b
procedure c
    begin
        :
        :
        :
    end ;
procedure d
```

```
procedure e
    begin
        :
        :
        :
    end ;
        begin
            :
            :
            :
        end ;
            begin
                :
                :
                :
            end ;
        procedure f
            begin
                :
                :
                :
            end ;
                begin
                    :
                    :
                    :
                end.
```

٦-٧ المفروض أن يقوم البرنامج التالي بقراءة ثلاثة أعداد صحيحة ، وترتيبها ترتيبا تصاعديا ، ثم طباعتها مرتبة ، ولكن البرنامج لا يؤدي إلى النتيجة المطلوبة. صحح أي أخطاء في البرنامج ليقوم بأداء وظيفته.

```
PROGRAM REORDER (INTPUT , OUTPUT) ;
    VAR A , B , C : INTEGER ;
PROCEDURE SORT (VAR X , Y , Z : INTEGER) ;
  PROCEDURE SWAP (X , Y : INTEGER) ;
                BEGIN
            X := Y ;
            Y := X ;
                END ;
                    BEGIN
    IF X > Y THEN SWAP (X , Y)
```

```
ELSE IF X > Z THEN SWAP (X , Z) ;
        IF Y > Z THEN SWAP (Y , Z) ;
                                END ;
                BEGIN (* REORDER ITSELF*)
                    READLN (A , B , C) ;
                      SORT (A , B , C) ;
                   WRITELN (A , B , C) ;
                                END.
```

٦-٨ ما هي مخرجات البرنامج التالي ؟(*)

```
                Program IslamicSong (output) ;
{Demonstrates declaration of procedures without parameters}
                procedure Chorus ; {Print the chours}
                                begin
                writeln (′Where do you come from′) ;
                writeln (′Where do you come from′) ;
                writeln (′North, South, East, or the West′) ;
        writeln (′It doesn′′t matter where you′′re born′) ;
                writeln (′Islam is the Best′)
                                end ; {Chorus}
                        procedure FirstVerse ;
                                begin
        writeln (′Where do you live, Where do you live′) ;
                writeln (′In a tent or in a town′) ;
        writeln (′It doesn″t matter where you live′) ;
                writeln (′You can soon be found′) ;
        writeln (′Because He is there , wherever you are′) ;
                writeln (′So, turn , turn to Allah′)
                                end ; {First Verse}
                        procedure SecondVerse ;
                                begin
writeln (′What are you saying, What are you saying′) ;
                writeln (′I′′m praying to Allah′) ;
        writeln (′To give us victory over the Koffar′) ;
                writeln (′Because He will be there′) ;
        writeln (′Where the good Muslims are′) ;
                writeln (′So, turn , turn to Allah′)
                                end ; {Second Verse}
```

(*) انشودة إسلامية من تأليف يوسف إسلام (" كات ستيفنز " سابقا)

```
Begin
    FirstVerse ;
    Chorus ;
    writeln ;
    SecondVerse ;
    Chorus
End.  {IslamicSong}
```

ثانيا : برامج

٦-٩ أ) اكتب برنامجا فرعيا إجرائيا

```
Break (x , whole , fraction)
```

يعطي قيمة الجزء الصحيح whole **والجزء الكسري** fraction **من قيمة أي عدد حقيقي مُعطَى** x.

(إرشاد : يمكن استخدام الدالة TRUNC **لتعيين قيمة الجزء الصحيح ، ثم الفرق بين هذه القيمة وقيمة العدد الأصلي يعطى الجزء الكسري).**

ب) اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة A **مكونة من خمسين عددا حقيقيا ، ويعيِّن لكل عدد جزأه الصحيح وجزأه الكسري باستخدام البرنامج الفرعي الإجرائي.**

٦-١٠ (١) المطلوب كتابة إجراء يقوم بضرب كل عنصرين متقابلين في منظومتين X,Y **تحتوي كل منهما على** n **عنصر ، ويضع العناصر الناتجة في منظومة** Z.

(ب) المطلوب كتابة برنامج رئيسي يقرأ قيم عناصر منظومتين A,B **تحتوي كل منهما على ثمانية عناصر ، ثم يستدعي الإجراء ليخزن حاصل ضرب عناصر** A,B **المتقابلة في منظومة جديدة** C **، وكذلك يطبع البرنامج الرئيسي عناصر المنظومات الثلاث** A , B , C.

٦-١١ نفرض أن لدينا الإعلانات والتعريفات التالية في البرنامج الرئيسي :

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    a : list ;
    n : integer ;
```

(أ) اكتب إجراء لتعديل (adjusting) قيم جميع عناصر منظومة أعداد حقيقية ، 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 الإجراء – من قبل البرنامج الرئيسي – بواسطة العبارتين.

```
n := 50 ;
adjust (a,n) ;
```

حيث a هي منظومة الأعداد الحقيقية المطلوب تعديل قيم عناصرها ، و n هو عدد عناصر المنظومة a.

(ب) أعد حل الجزء أ) ولكن بحيث تكون أكبر قيمة في المنظومة المعدلة تساوي صفرا ، وبحيث يتم استدعاء الإجراء بالعبارتين

```
n := 50 ;
shift (a,n) ;
```

(ج) أعد حل الجزء أ) ولكن بحيث تكون القيمة المتوسطة في المنظومة المعدلة تساوي صفرا ، وبحيث يتم استدعاء الإجراء بالعبارة

```
zeroav (a) ;
```

حيث منظومة الأعداد الحقيقية a تحتوي على 50 عنصرا.

٦–١٢ اكتب إجراء لطباعة قيم عناصر منظومة أعداد صحيحة عددها n في الصيغة التالية :

```
VALUES IN ARRAY
XXX    1.
XXX    2.
XXX    3.
```

وهكذا حتى تطبع جميع القيم.

نفرض أن k هي منظومة الأعداد الصحيحة ، وأنه يتم استدعاء الإجراء بواسطة العبارتين :

```
n := 50 ;
prtval (k , n) ;
```

وأن لدينا الإعلانات التالية في البرنامج الرئيسي :

```
type
    index = 1 .. 50 ;
    list = array [index] of integer ;
var
    k : list ;
    n : integer ;
```

٦-١٣ نفرض أن لدينا التعريفات التالية في البرنامج الرئيسي :

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    a, b , c : list ;
```

اكتب إجراء bigarr **له ثلاثة وسطاء** a , b , c **وكل منها عبارة عن منظومة أعداد حقيقية عددها** 50 **، حيث يستقبل الإجراء قيما لعناصر كل من المنظومتين** a , b **، ويعين قيم عناصر المنظومة** c **بناء على القاعدة التالية :**

إذا كان مجموع قيم عناصر a **أكبر من أو يساوي مجموع قيم عناصر** b **فإن الإجراء يسند لعناصر المنظومة** c **قيم العناصر المقابلة في المنظومة** a **وما عدا ذلك فإنه يسند لعناصر** c **قيم العناصر المقابلة في** b.

٦-١٤ نفرض أن لدينا التعريفات التالية في البرنامج الرئيسي :

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    b : list ;
    limit : real ;
```

اكتب إجراء bottom **له وسيطان :**

b : **منظومة أعداد حقيقية مكونة من خمسين عنصرا ،**

limit : **قيمة اختبارية** (test value) **وهي عدد حقيقي ،**

حيث يقوم الإجراء 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 البرنامج الفرعي بواسطة العبارة bottom (b , limit).

٦-١٥ نفرض أن البرنامج الرئيسي يحتوي على الإعلان التالي :

```
type
    index = 1 .. 50 ;
    list = array [index] of integer ;
var
    b : list ;
```

اكتب إجراء يقوم 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 الإجراء بالعبارة

```
chng (b) ;
```

تصبح المنظومة b مكونة من أصفار وآحاد فقط.

٦-١٦ اكتب إجراء يمكن استدعاؤه بالعبارة

```
modify (x) ;
```

حيث x منظومة مكونة من 50 عدد حقيقي. ويقوم الإجراء بعكس ترتيب (reversing the order) قيم المنظومة ، أي أنه يبدل القيمة الأولى مع القيمة الأخيرة ، والقيمة الثانية مع القيمة قبل الأخيرة ، وهكذا.

افرض أن البرنامج الرئيسي يحتوي على الإعلان التالي

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    x : list ;
```

٦-١٧ اكتب إجراء time يستقبل عددا صحيحا totmin يمثل العدد الكلي للدقائق ، ويعيد ثلاث قيم صحيحة تمثل عدد الأيام days وعدد الساعات hours وعدد الدقائق min المقابلة للقيمة المدخلة totmin

مثلا : إذا كانت totmin = 80 فإن :

days = 0 , hours = 1 , min = 20

ويمكن استدعاء الإجراء بالعبارة :

time (totmin , days , hours , min) ;

٦-١٨ اكتب إجراء length يستقبل عددا صحيحا totinc يمثل عددا من البوصات ، ويعيد ثلاث قيم صحيحة تمثل عدد الياردات yards ، وعدد الأقدام feet ، وعدد البوصات inches المقابلة للقيمة المدخلة totinc

مثلا : إذا كانت totinc = 80 فإن :

yards = 2 , feet = 0 , inches = 8

(ملاحظة : الياردة = ٣ أقدام ، والقدم = ١٢ بوصة)

ويمكن استدعاء الإجراء بالعبارة :

length (totinc , yards , feet , inches) ;

٦-١٩ اكتب إجراء time يستقبل عددا صحيحا totday يمثل عددا من الأيام ، ويعيد قيمتين صحيحتين تمثلان عدد الأسابيع weeks ، وعدد الأيام days المقابلة للقيمة المدخلة totday.

مثلا : إذا كانت totday = 80 فإن :

weeks = 11 , days = 3

ويمكن استدعاء الإجراء بالعبارة :

time (totday , weeks , days) ;

٦-٢٠ نفرض أن المنظومة y تتكون من مائة قيمة حقيقية 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 إجراء يستقبل منظومة y مكونة من مائة قيمة حقيقية موجبة ، ويستبدل أصفارا بالقيم التي تقل أي منها عن ٣٪ من أكبر قيمة بالمنظومة.

٦-٢١ اكتب إجراء trig يأخذ عددا صحيحا angle يمثل زاوية بالدرجات degrees ويطبع كلا من جيب sine وجيب تمام cosine الزاوية بالصيغة التالية :

SINE (xxx.xx DEGREES) = x.xxxx
COSINE (xxx.xx DEGREES) = x.xxxx

افرض أن الثابت pi قد تم الإعلان عنه في البرنامج الرئيسي وأن قيمته 3.14159.

يتم استدعاء الإجراء بالعبارة ; trig (angle)

أ) اكتب إجراء scale لتعديل قيم جميع عناصر منظومة أعداد حقيقية موجبة a عدد عناصرها n ، وذلك بقسمة كل قيمة في المنظومة على أكبر قيمة في المنظومة ، وبذلك تصبح جميع القيم محصورة بين القيمتين 1.0 , 0.0.

مثلا : المنظومة $\begin{bmatrix}4.0\\8.0\\0.0\\2.0\end{bmatrix}$ تصبح بعد التعديل $\begin{bmatrix}0.5\\1.0\\0.0\\0.25\end{bmatrix}$ حيث $n = 4$

ويتم استدعاء الإجراء scale بواسطة البرنامج الرئيسي بالعبارتين :

```
n := 50 ;
scale (a , n) ;
```

ونفرض أن لدينا التعريفات التالية في البرنامج الرئيسي :

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    a : list ;
    n : integer ;
```

ب) اكتب برنامجا رئيسيا يستخدم التعريفات السابقة ، ويقرأ قيم عناصر منظومة a من ٥٠ عدد حقيقي موجب ، ويستدعي الإجراء scale لتعديل قيم عناصر a ثم يطبع قيم عناصر a بعد التعديل.

٦-٢٣ اكتب إجراء (procedure) يعكس ترتيب القيم المخزونة في منظومة. المتغيرات الوسيطة (parameters) في الإجراء هي : منظومتان X , Y من الأعداد الحقيقية ، وعدد صحيح n يمثل طول (length) كل من المنظومتين.

مدخلات الإجراء (procedure inputs) : X , n

مخرجات الإجراء (procedure outputs) : Y

والإجراء يقوم بنسخ (copying) عناصر المنظومة X في المنظومة Y بترتيب معاكس (وذلك باستخدام عروة) ، أي أن :

$$Y[1] = X[n] , Y[2] = X[n-1] , \dots , Y[n-1] = X[2] , Y[n] = X[1]$$

٦-٢٤ أ) اكتب إجراء بلغة الباسكال يستقبل ثلاثة أعداد ويعيد أكبر عنصر في الثلاثة.

ب) اكتب برنامجا بلغة الباسكال يشتمل على الإجراء المذكور سابقا ويستمر البرنامج في قراءة ثلاثة أعداد مدخلة على سطر واحد ثم طباعة أكبر عنصر في الأعداد الثلاثة المدخلة ، ويتوقف البرنامج عندما تكون الأعداد الثلاثة المدخلة متساوية.

فيما يلي مثال لنتائج تشغيل البرنامج :

4	7	3	max = 7
8	9	0	max = 9
23	7	74	max = 74
1	0	4	max = 4
3	3	3	stop

٦-٢٥ أ) اكتب إجراء Subsequences يستقبل منظومة X من n عدد صحيح ، ويعطي منظومة Y من n عدد صحيح ، عناصرها هي مجاميع المتتاليات الجزئية (array subsequences) للمنظومة X ، أي أن :

$$y_1 = x_1$$
$$y_2 = x_1 + x_2$$
$$y_3 = x_1 + x_2 + x_3$$
$$\vdots$$
$$+ x_i \ldots + x_j + \ldots y_i = x_1 + x_2 +$$
$$\vdots$$
$$+ x_n \ldots\ldots\ldots\ldots y_n = x_1 + x_2 +$$

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 رئيسيا يعين قيم عناصر منظومة A من عشرة أعداد صحيحة باستخدام العلاقة

$$, 10 \ldots A_i = i \; ; \; i = 1 , 2 ,$$

ثم يستدعي الإجراء السابق لتعيين منظومة مجاميع المتتاليات الجزئية B للمنظومة A.

٦-٢٦ اكتب إجراءين أحدهما لإظهار / لعرض (displaying) مثلث (Triangle) والآخر لإظهار مستطيل (Rectangle) . استخدم هذين الإجراءين لكتابة برنامج كامل يعرض شكلا تخطيطيا لبيتين ، مستعينا بالخطوات الرئيسية التالية :

```
Program StackHouses (input , output) ;
Begin
    {1. Draw Triangle.}
    {2. Draw Rectangle.}
    {3. Print 2 blank lines.}
    {4. Draw Triangle.}
    {5. Draw Rectangle.}
End.
```

٦-٢٧ اكتب إجراءات لعرض الحروف الأولى من اسمك ، ثم اكتب برنامجا ينفذ هذه الإجراءات.

٦-٢٨ اكتب إجراءين لرسم مربع ومثلث. ثم اكتب برنامجا يقرأ حرف S أو حرف T وبناء على هذا الحرف الذي يقرأه يرسم مربعا (Square) أو مثلثا (Triangle) على الترتيب.

٦-٢٩ أضف إلى برنامج السؤال السابق إجراءين لرسم دائرة وخط مستقيم. ثم أعد كتابة البرنامج الرئيسي بحيث يقرأ ثلاثة حروف من المجموعة : C (تعني دائرة Circle) و L (تعني خطا مستقيما Line) و S (تعني مربعا Square) و T (تعني مثلثا Triangle) بحيث أن هذه الحروف تمثل متغيرات من النوع الرمزي char. ثم يرسم البرنامج الثلاثة أشكال المعنية. فمثلا إذا كانت البيانات هي CLT فإن البرنامج يرسم دائرة ثم مستقيما ثم مثلثا. استخدم عبارة حالة (Case Statement) لرسم كل من الأشكال الثلاثة.

٦-٣٠ اكتب برنامجا لقراءة رقم أحد الفصول (الشعب / الغرف) وسعة الفصل وعدد الطلاب المقيدين في الفصل ، ومن ثم يقوم البرنامج بطباعة رقم الفصل وسعته وعدد المقاعد الموجودة في الفصل والمشغولة بواسطة الطلاب (أي عدد الطلاب المقيدين) وعدد المقاعد الفارغة (غير المشغولة) ، وأيضا يقوم بطباعة رسالة تفيد ما إذا كان الفصل مشغولا بأكمله أم لا.

استخدم إجراء لطباعة العناوين التالية قبل سطر المخرجات.

Room Capacity Enrollment Empty Seats Filled Not Filled

واستخدم البيانات التالية كمدخلات للبرنامج.

Room	Capacity	Enrollment
426	25	25
327	18	14
420	20	15
317	100	90

٦-٣١ 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 برنامج فرعي إجرائي

Procedure LSTSQ (X , Y , N , A , B)

لإيجاد قيمة كل من الثابتين a , b **إذا علمت النقاط**

$$, (x_n , y_n) \ldots (x_1 , y_1) , (x_2 , y_2) ,$$

ب) كتابة برنامج رئيسي يقرأ إحداثيات عشرين نقطة ، ثم يستدعي البرنامج الفرعي لحساب قيمة كل من a , b **، ثم يطبع معادلة الخط المستقيم في الصورة** $y = a + bx$ **(بعد التعويض عن كل من** a , b**).**

٦-٣٢ أ) اكتب برنامجا فرعيا داليا Area **يحسب مساحة مثلث بدلالة أطوال أضلاعه** a , b , c **حيث :**

$$Area = \sqrt{s(s-a)(s-b)(s-c)}$$

حيث s **تمثل نصف المحيط :** $s = \frac{a+b+c}{2}$

ب) اكتب برنامجا رئيسيا يقرأ أطوال أضلاع مثلث side1 , side2 , side3 **ويحسب مساحته باستخدام الدالة** Area**.**

٦-٣٣ أ) اكتب دالة freq **تأخذ الوسيطين** (parameters / arguments) a , n

حيث a **: منظومة مكونة من ٥٠ عدد حقيقي.**

n **: عدد حقيقي**

وتعطي عدد مرات ظهور العدد الحقيقي n في المنظومة a.

ب) اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة X تمثل درجات طلاب فصل به خمسون طالبا ، ويستخدم الدالة freq لمعرفة عدد الطلاب الحاصلين على الدرجة النهائية (١٠٠).

٦-٣٤ 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).

٦-٣٥ اكتب دالة totdys توجد مجموع وسيطيها days (عدد الأيام) و weeks (عدد الأسابيع) محسوبا بالأيام.

مثلا : totdys (5 , 9) = 44

٦-٣٦ اكتب دالة منطقية bigeng لها ٣ وسطاء a , b , c عبارة عن أعداد صحيحة ، وتكون قيمة الدالة true عندما يتحقق (على الأقل) واحد من الشروط الثلاثة :

$a > bc$, $b > ac$, $c > ab$

وما عدا ذلك فإن قيمة الدالة تكون false.

٦-٣٧ اكتب دالة منطقية small لها ٣ وسطاء a , b , c عبارة عن أعداد حقيقية ، وتكون قيمة الدالة true إذا تحقق أحد الشروط الثلاثة :

$a < b/c$, $b < a/c$, $c < a/b$

وما عدا ذلك فإن قيمة الدالة تكون false.

٦-٣٨ 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.

٦-٣٩ نفرض أن لدينا الإعلانات التالية في البرنامج الرئيسي :

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    a : list ;
    n : integer ;
```

أ) اكتب دالة منطقية ascend تستقبل وسيطين n , a حيث a منظومة أعداد حقيقية عدد عناصرها n ، وتعيد الدالة القيمة true إذا كانت المنظومة a مرتبة ترتيبا تصاعديا ، وإلا فإنها تعيد القيمة false.

ملاحظة : الدالة لا تقوم بترتيب عناصر المنظومة ، ولا تقوم بتغيير أي عنصر من عناصرها ، ولا تقوم بتغيير قيمة n.

ب) أعد حل الجزء أ) ولكن بالنسبة لدالة منطقية descnd تعيد القيمة true عندما تكون المنظومة a مرتبة ترتيبا تنازليا.

٦-٤٠ اكتب دالة منطقية boundd تستقبل ٣ وسطاء : منظومة حقيقية a من النوع المعرف بالمستخدم (user defined type) list بها خمسون عنصرا ، وعددين حقيقيين rmin , rmax ، وتعيد القيمة true إذا وقعت جميع قيم عناصر المنظومة a في المدى ما بين الحد الأدنى (lower bound) rmin والحد الأقصى (upper bound) rmax ، أما إذا وقعت أي قيمة خارج هذا المدى فإن الدالة تعيد القيمة false.

٦-٤١ 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 صحيحة maxs تحسب عدد مرات ظهور أعلى قيمة في منظومة أعداد حقيقية ، حيث تستقبل الدالة وسيطين : المنظومة a وعدد عناصرها n. افرض أن لدينا التعريفات التالية في البرنامج الرئيسي.

```
type
    index = 1 .. 50 ;
    list = array [index] of real ;
var
    a : list ;
    n : integer ;
```

٦-٤٢ اكتب دالة حقيقية القيمة range توجد مدى قيم عناصر منظومة أعداد حقيقية a عدد عناصرها n ، حيث يعرف المدى هنا بأنه الفارق بين أكبر قيمة وأصغر قيمة من قيم عناصر المنظومة. الدالة تستقبل الوسيطين n , a ، وافرض أن المنظومة a من نوع معرف بالمستخدم list.

٦-٤٣ اكتب دالة حقيقية القيمة cubert تستقبل وسيطا واحدا x عبارة عن عدد حقيقي ، وتوجد جذره التكعيبي.

٦-٤٤ 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).

٦-٤٥ اكتب دالة حقيقية القيمة close تستقبل وسيطين : a , n ، حيث a : منظومة أعداد حقيقية ، و n : عدد عناصر المنظومة a ، ثم توجد قيمة أقرب عنصر / (nearest closest element) من عناصر المنظومة للقيمة المتوسطة (average value) لعناصر المنظومة. افرض أن المنظومة a من النوع المعرف بالمستخدم list.

٦-٤٦ اكتب دالة حقيقية القيمة تحسب مساحة شبه منحرف (trapezoid) إذا علم طولا قاعدتيه المتوازيتين b1 , b2 وارتفاعه h. الدالة لها ٣ وسطاء (b1 , b2 , h).

٦-٤٧ أ) اكتب دالة حقيقية القيمة 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 حقيقية القيمة 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.

٦-٤٨ اكتب دالة صحيحة القيمة 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 من نوع معرف بالمستخدم list.

٦-٤٩ اكتب دالة صحيحة تأخذ منظومة رموز ثنائية البعد (أي مصفوفة) جميع عناصرها حروف كبيرة ، وتعيد عدد مرات ظهور الحروف الساكنة في المنظومة ، ويمكن استدعاء الدالة بالعبارة

a := count (b , n , m) ;

حيث b : منظومة ثنائية البعد (مصفوفة) من نوع معرف بالمستخدم list ، n : عدد صفوف المصفوفة ، m : عدد أعمدتها.

ملاحظة : الحروف الساكنة هي جميع الحروف غير المتحركة ، والحروف المتحركة هي : A , E , I , O , U.

٦-٥٠ 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 برنامجا فرعيا داليا لحساب n! أي مضروب n ، حيث n عدد صحيح موجب.

ب) اكتب برنامجا يستخدم البرنامج الفرعي الدالي السابق 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.

٦-٥١ أ) اكتب برنامجا فرعيا لدالة 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).

ب) أوجد ناتج البرنامج (الجزئي) التالي الذي يستخدم البرنامج الفرعي السابق.

```
I := 2 ;
J := 4 ;
K := 3 ;
X := ADD(I,J) ;
Y := ADD(J,K) ;
Z := ADD(J,5) ;
WRITE ('X = ', X , 'Y = ', Y , 'Z = ' , Z) ;
```

٦-٥٢ أ) اكتب برنامجا مساعدا فرعيا (إجراء)

```
SUB (J, K, JUSM , JPROD, JDIFF)
```

حيث J , K : عددان صحيحان

JSUM : تمثل المجموع J + K

JPROD : تمثل حاصل الضرب J * K

JDIFF : تمثل الفارق J - K

ب) أوجد ناتج تنفيذ البرنامج الجزئي التالي الذي يستخدم البرنامج المساعد السابق.

```
J := 3 ;
K := 2 ;
SUB (J, K, L, M, N) ;
SUB (L+M , 3*N , J , N , M) ;
WRITE (J , K , L , M , N) ;
```

٦-٥٣ عَرِّف دالة لحساب

$$R = \sqrt{u^2 + v^2 + w^2}$$

ثم استخدمها لحساب :

$$A = \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

$$B = \sqrt{4x^4 + 9y^2 + 25z^2}$$

$$C = \sqrt{9 + \sin^2 y + u^2}$$

٦-٥٤ اكتب برنامجا مساعدا يقوم بتبديل عنصرين معينين من عناصر منظومة. افرض أن متغيرات البرنامج الوسيطة هي :

A : **اسم المنظومة**

i , j : **ترتيبا العنصرين المطلوب تبديلهما.**

افرض كذلك أن المنظومة لا تحتوي على أكثر من ٢٤ عنصرا ..

٦-٥٥ أ) نفرض أن كلا من A , B منظومة مكونة من عدد من العناصر يساوي N. المطلوب كتابة برنامج فرعي

FUNCTION PROD (A , B , N)

يحسب 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 رئيسيا يقرأ قيم عناصر كل من المنظومات X , Y , Z والتي تتكون كل منها من أربعين عنصرا ، ثم يستخدم البرنامج الفرعي السابق في حساب المقادير التالية :

$$\alpha = \left(\sum_{i=1}^{40} x_i z_i \right)^2$$

$$\beta = \frac{\sqrt{x_1^2 + x_2^2 + \cdots + x_{40}^2} \cdot \sqrt{z_1^2 + z_2^2 + \cdots + z_{40}^2}}{\sqrt{x_1 z_1 + x_2 z_2 + \cdots + x_{40} z_{40}}}$$

$$\gamma = x_1 (y_1 + z_1) + x_2 (y_2 + z_2) + \cdots + x_{20} (y_{20} + z_{20})$$

٦-٥٦ أ) نفرض أن X منظومة ، وأن عناصرها أعداد صحيحة ، وأن عدد هذه العناصر M. اكتب برنامجا فرعيا لدالة صحيحة القيمة :

FUNCTION NCOUNT (X , M , J , K , ITEM)

يحسب عدد مرات ظهور العنصر ITEM في المجموعة الجزئية للعناصر ابتداء من العنصر رقم J إلى العنصر رقم K (K ≥ J) من المجموعة X. فمثلا إذا كانت المجموعة X هي :

X = [0 , 1 , 1 , 2 , 0 , 0 , 0 , 0 , 2]

NCOUNT (X , 9 , 2 , 7 , 0) = 3 **فيكون**

ب) اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة A مكونة من ثمانين عنصرا من الأصفار والآحاد ثم يستخدم البرنامج الفرعي السابق لإيجاد :

١- عدد الأصفار في المنظومة A ونرمز له بالرمز N0.

٢- عدد الآحاد في المنظومة A ونرمز له بالرمز N1.

٣- عدد الأصفار في أول خمسين عنصرا من المنظومة A ونرمز له بالرمز N.

٤- عدد الآحاد في مجموعة العناصر ابتداء من العنصر الثلاثين إلى العنصر السبعين ونرمز له بالرمز L.

٦-٥٧ أ) اكتب برنامجا فرعيا لدالة حقيقية القيمة FUNCTION Y (X, N, MEAN) تحسب قيمة الدالة 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 :

$$Y = \begin{cases} \frac{X_1+X_2+\cdots+X_N}{N} & \text{if MEAN} = 1 \\ \left(X_1.X_2...X_N\right)^{\frac{1}{N}} & \text{if MEAN} = 2 \end{cases}$$

ب) اكتب برنامجا رئيسيا يقرأ قيم العناصر الثلاثين لمنظومة A وقيمة متغير K ثم يستخدم البرنامج الفرعي السابق لحساب المتوسط الحسابي أو المتوسط الهندسي لعناصر المنظومة A حسب ما إذا كانت قيمة K تساوي ١ أو ٢ على الترتيب.

٦-٥٨ أ) اكتب برنامجا فرعيا لدالة 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 رئيسيا :

١- يقرأ قيمة عدد صحيح n_1 (حيث $n_1 \le 30$) وقيم عناصر منظومة A مكونة من n_1 عنصرا.

٢- يقرأ قيمة عدد صحيح n_2 (حيث $n_2 \le 40$) وقيم عناصر منظومة B مكونة من n_2 عنصرا.

٣- يستخدم البرنامج الفرعي السابق 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.

٦-٥٩ أ) اكتب برنامجا فرعيا لإجراء

POLY (A , N , X , P , PD)

حيث A منظومة مكونة من N+1 عنصرا ، والبرنامج يحسب 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 رئيسيا :

١- يقرأ قيم أول عشرة عناصر من منظومة C ، أي قيم العناصر

$C_1 , C_2 , \dots , C_{10}$

وكذلك قيمة متغير Y.

٢- يستخدم البرنامج المساعد 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.

٦-٦٠ اكتب برنامجا مساعدا (إجراء)

MAXMIN (A , N , BIGA , SMALLA)

لإيجاد أكبر عنصر BIGA واصغر عنصر SMALLA في المنظومة / المجموعة الجزئية للعناصر التي عددها N والتي تأتي في مطلع منظومة / مجموعة العناصر المرتبة A.

ثم اكتب برنامجا رئيسيا 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 البرنامج المساعد MAXMIN لإيجاد أكبر عنصر (XHIGH) وأصغر عنصر (XLOW) في المجموعة $X_1 , X_2 , X_3 , \ldots , X_M$. افرض أن $M \leq 20$.

٦-٦١ أ) اكتب برنامجا فرعيا لدالة AVG (X , N) يحسب القيمة المتوسطة لعدد N من الأرقام $X_1 , X_2 , X_3 , \ldots , X_N$

أي يحسب $\left(\sum_{i=1}^{N} X_i \right) / N$

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 البرنامج الفرعي للدالة AVG لمعرفة أي الفصلين حصل طلابه على المعدل الأعلى لمتوسط الدرجات ، مع إعطاء رسالة توضح هذه النتيجة.

٦-٦٢ أ) اكتب برنامجا فرعيا لدالة 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 البرنامج الفرعي السابق في (أ).

٦-٦٣ أ) نفرض أن A مصفوفة مربعة عدد كل من صفوفها وأعمدتها N . اكتب برنامجا مساعدا

SYM (A , N , K)

بحيث يعطي K القيمة ١ (أي أن K = 1) إذا كانت المصفوفة A متماثلة (Symmetric) أي إذا كانت

لجميع قيم i , j $A_{ij} = A_{ji}$

ويعطى K القيمة صفر إذا كانت A غير متماثلة.

ب) اكتب برنامجا رئيسيا

١- يقرأ عناصر مصفوفة مربعة B مكونة من خمسة صفوف وخمسة أعمدة.

٢- يستخدم البرنامج المساعد SYM ليرى إذا كانت المصفوفة B متماثلة أم لا.

٣- إذا كانت B غير متماثلة يحسب مدورها B^T (transpose of B) ويخزنه في المصفوفة C ، أي أن

لجميع قيم i,j $C_{ij} = B_{ji}$

٦-٦٤ أ) اكتب برنامجا مساعدا (إجراء)

AVNZ (A , M , N , AVA , AVG , NZ)

حيث A : منظومة مكونة من M عنصر.

والبرنامج يحسب 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 يحسب كم عنصرا من هذه العناصر (التي عددها N) يساوي صفرا ، ويشير إلى عدد هذه الأصفار بالاسم NZ.

ب) اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة BETA مكونة من خمسين عنصرا ، ثم يستعين بالبرنامج المساعد السابق 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 برنامجا فرعيا

SEARCH (BUFFER , N , KEY , FOUND , INDEX)

يقوم بالبحث عن العنصر KEY في المنظومة BUFFER والمكونة من عدد من العناصر يساوي N ، فإذا وجد العنصر ضمن هذه المجموعة من العناصر فإنه يعطي FOUND القيمة 1 ويعطي INDEX قيمة ترتيب هذا العنصر الذي وجده في المنظومة BUFFER ، وإذا لم يجده فإنه يعطي FOUND القيمة صفر ، ولا يغير من قيمة INDEX.

ب) اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة A مكونة من خمسين عنصرا ومنظومة أخرى X مكونة من ثلاثين عنصرا ثم يستخدم البرنامج الفرعي السابق لمعرفة العناصر المشتركة في المنظومتين ، بحيث يطبع البرنامج هذه العناصر المشتركة وبجانب كل عنصر ترتيبه في المنظومة A.

٦-٦٦ أ) افرض أن JEST منظومة تحتوي على N1 عنصر عبارة عن أعداد صحيحة موجبة مختلفة .. وافرض كذلك أن KSET منظومة تحتوي على N2 من الأعداد الصحيحة الموجبة المتباينة. اكتب إجراء

INTER (JSET , N1 , KSET , N2 , JKSET , N3)

يقوم بتخزين تقاطع المنظومتين المذكورتين في المنظومة JKSET ، أي أن JKSET ستحتوي على N3 عنصر هي الأعداد الموجودة في كل من JSET , KSET.

ب) نفرض أن كل اسم من أسماء الصحابة رضوان الله عليهم جميعا قد أعطى رقما موجبا مميزا (أي مختلفا عن غيره من الأرقام) ، وأن المنظومة BADR

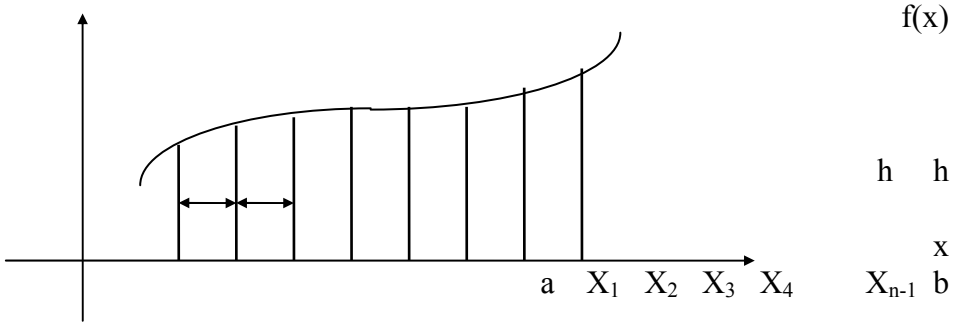
تحتوي على ٣١٤ رقما هي أرقام الصحابة الذين اشتركوا في غزوة بدر ، وأن المنظومة OHOD تحتوي على ٧٠٠ رقم هي أرقام الصحابة الذين اشتركوا في غزوة أحد.

اكتب برنامجا يقرأ أرقام كل من المجموعتين BADR و OHOD ثم يستخدم الإجراء السابق INTER لإيجاد المنظومة JBO 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 $x = a , x = b$ والتي تعطى بالتكامل

$$AREA = \int_a^b f(x)dx$$

يمكن حساب قيمة تقريبية لها بإيجاد مجموع مساحات عدد n من أشباه المنحرفات ، كما هو مبين بالشكل التالي :



فنحصل على القانون :

$$\text{Area} = \frac{h}{2}\left[f(a) + f(b) + 2\left\{f(x_1) + f(x_2) + \cdots + f(x_{n-1})\right\}\right]$$

$$= \frac{h}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{i=n-1} f(x_i)\right].$$

حيث :

$$h = \frac{b-a}{n}, x_i = a + ih;\ i = 1,2,\cdots,n-1$$

اكتب برنامجا :

١- يقرأ قيمة كل من a , b , n .

٢- يحسب قيمة h .

٣- يحسب المساحة تحت المنحنى

$$F(x)=e^{-x^2}+3x^3-4x^2+6x+5$$

والمحصورة بين $x = a$, $x = b$ وذلك باستخدام القانون السابق.

إرشاد : يمكن استخدام برنامج فرعي للدالة $F(x)$.

٦-٦٨ أ) اكتب برنامجا فرعيا داليا FUNCTION FACT (K)

لحساب قيمة الدالة $Fact(k) = k!$

ب) اكتب برنامجا فرعيا داليا آخر FUNCTION CNR (N , R) يستخدم البرنامج الفرعي السابق لحساب قيمة الدالة

$$C(n,r) = \frac{n!}{r!\ (n-r)!}$$

والتي تعطي عدد التوافقات أي المجموعات المختلفة الممكن تكوينها من عدد n من العناصر بحيث نأخذ r عنصرا في المرة الواحدة.

ج) اكتب برنامجا رئيسيا يستدعي البرنامج الفرعي CNR للقيم التالية :

أولا : $n = 4$, $r = 1$

ثانيا : $n = 5$, $r = 3$

ثالثا : $n = 7$, $r = 7$

ويطبع قيمة الدالة في كل حالة من الحالات الثلاث.

٦-٦٩ تستخدم طريقة (النقطة الثابتة) (Fixed point) والتي تسمى أيضا طريقة (التقريبات المتتابعة) (successive approximations) لإيجاد جذر للمعادلة $F(x) = 0$ وذلك بوضع المعادلة أولا في الصورة $x = G(x)$ والبدء بقيمة تقريبية X_1 للجذر ، ثم إيجاد قيمة أدق X_2 باستخدام العلاقة $X_2 = G(X_1)$ ، فإذا كانت القيمة المطلقة للفارق بينهما أقل من رقم معين ε - يعتمد على الدقة المطلوبة في الحل - نعتبر أن قيمة X_2 هي الجذر ، وإلا فإننا نعتبر أن X_2 هي القيمة التقريبية (أي نضع $X_1 = X_2$) ونحصل على قيمة أدق X_2 للجذر وهكذا .. وإذا لم يحدث تقارب للقيم بعد عدد N من المرات نطبع رسالة بهذا المعنى.

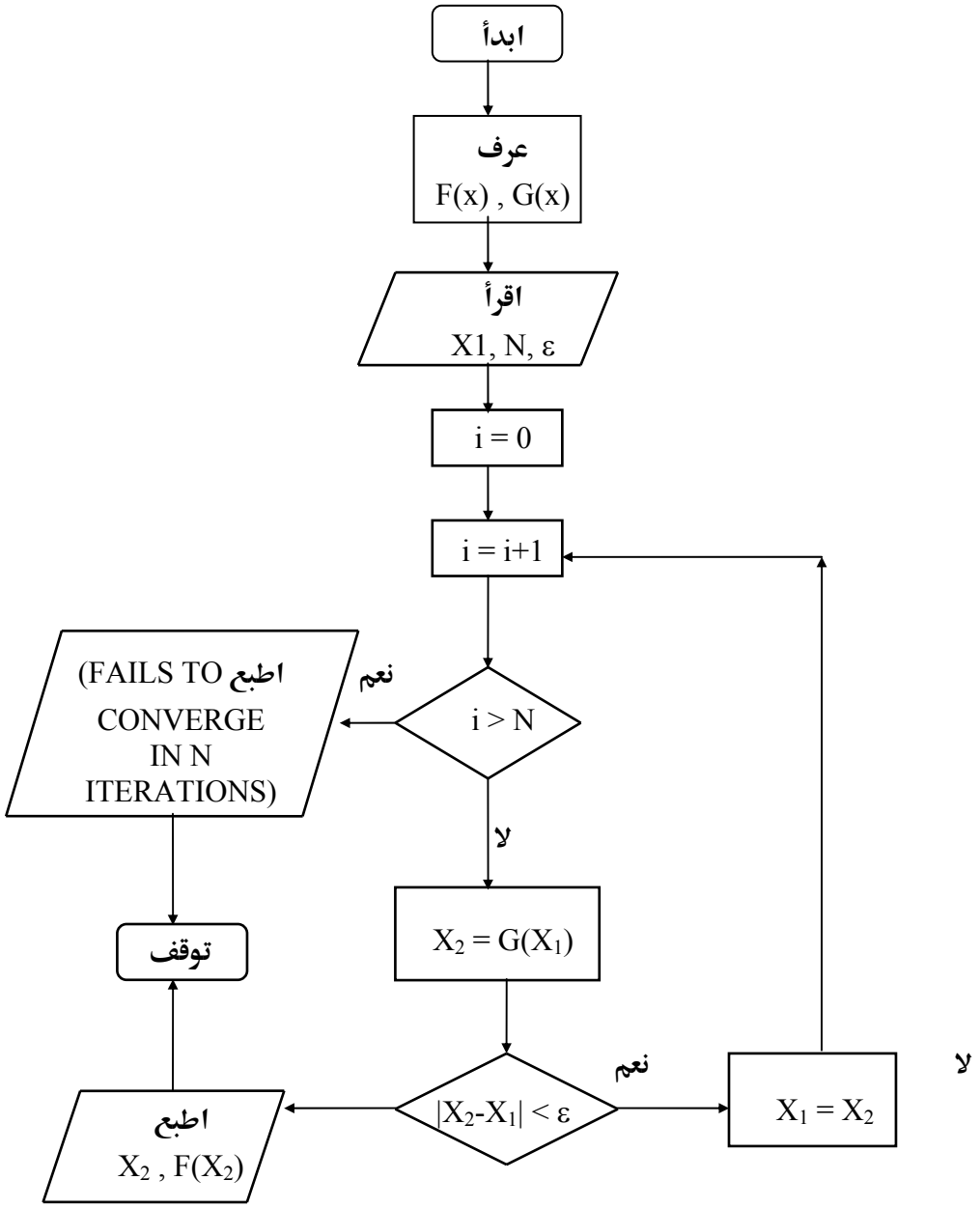
المطلوب : ترجمة خريطة سير العمليات المرسومة فيما يلي والتي توضح خطوات هذه الطريقة إلى برنامج لحل المعادلة $F(x) = 0$.

حيث :

$$F(x) = x^2 - \sqrt{x} - 3$$

بعد وضعها في الصورة $x = G(x)$ **حيث :** $G(x) = \left(\sqrt{x} + 3\right)^{\frac{1}{2}}$

على أن يبدأ البرنامج بتعريف كل من الدالتين $F(x)$ **و** $G(x)$.



٦-٧٠ يمكن لدالة المضروب $Fact(k) = k!$ **أن تعرف بالعلاقة الارتدادية التالية :**

$$Fact(0) = 1 \qquad Fact(k) = k * Fact(k-1) ; \qquad (1)$$

كذلك يمكن لدالة المجموع الجزئي (partial sum)

$$sum(k) = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{k!}$$

أن تعرف بالعلاقة الارتدادية :

$$sum(k) = sum(k-1) + \frac{1}{Fact(k)} \; ; \quad sum(0) = 1. \qquad (2)$$

١- اكتب برنامجا فرعيا داليا لتعريف الدالة Fact(k) **باستخدام العلاقة الارتدادية** (1).

٢- اكتب برنامجا فرعيا داليا لتعريف الدالة Sum (k) **باستخدام العلاقة الارتدادية** (2) **، والتي تستدعي البرنامج الفرعي الدالي السابق** Fact(k) .

٣- اكتب برنامجا رئيسيا يستخدم البرنامجين الفرعيين السابقين لحساب قيمة e **باستعمال العلاقة**

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{k!} + \cdots$$

بحيث توقف هذه المتسلسلة اللانهائية حينما تصبح قيمة الحد ($\frac{1}{k!}$) **صغيرة صغرا كافيا (افرض أن** $\varepsilon = 10^{-9}$; $\frac{1}{k!} < \varepsilon$ **).**

٦-٧١ اكتب برنامجا لحساب الضريبة (tax) **المفروضة على كل واحد من مجموعة من الممتلكات** (properties) **عددها ١٤ ، وكذلك الضريبة الكلية** (total tax) **، علما بأن معدل الضريبة هو ١٢,٥ سنت لكل دولار من القيمة المقدرة** (assessed value). **والقيمة المقدرة لكل من هذه الممتلكات تعادل ٢٨٪ من قيمته السوقية** (market value). **وفيما يلي القيم السوقية للممتلكات :**

	$ 167,000	$ 145,500	$ 148,000	$ 150,000
$ 153,350	$ 128,000	$ 165,000	$ 147,100	$ 137,600
$ 143,700	$ 156,500	$ 148,000	$ 148,100	$ 152,250

وتحتاج في حَلِّك لكتابة عدة إجراءات تقابل الإجراءات / الدوال المعرفة بالعناوين التالية :

```
Procedure PrintInstructions ;
{Display Instructions to the user}
```

```
Procedure ProcessProperties (Var TotalTax {output} : real) ;
{Reads market values and computes taxes on all properties}

Function ComputeTax (Market : Real) : Real ;
{Computes tax on a property with market value Market}

Procedure PrintSummary (TotalTax {input} : Real) ;
{Displays value of TotalTax}
```

٦-٧٢ أعد كتابة برنامج السؤال رقم (٦-٧١) بعد فرض ضريبة إضافية ٥٪ لجميع الممتلكات التي تزيد قيمتها السوقية عن ١٥٠,٠٠٠ دولار ، ومنح خصم ١٠٪ للمواطنين كبار السن (senior citizens) ، ولذلك تحتاج في البيانات لإدخال قيمة ثانية لكل من هذه الممتلكات تفيد ما إذا كان صاحبها من كبار السن أم لا. اكتب إجراء منفصلا يقرأ ويعيد القيمة السوقية الحالية للعقار ويعطي القيمة True لوسيط من النوع المنطقي إذا كان المالك من كبار السن.

٦-٧٣ تدرس اللجنة العليا للجامعة زيادة مرتبات ١٢ عضوا من أعضاء هيئة التدريس بنسبة قدرها ٥,٥٪ ، ولكنها تود أولا قبل إقرار هذه الزيادة معرفة كم سيكلفها ذلك. اكتب برنامجا لطباعة الزيادة في كل مرتب لـ ١٢ عضو ، وأيضا مجموع هذه الزيادات ، وكذلك طباعة مجموع المرتبات قبل وبعد الزيادة. اختبر البرنامج بقيم المرتبات التالية :

$ 43,250	$ 36,000	$ 24,029,50	$ 32,500
$ 28,900	$ 30,000,50	$ 22,800	$ 53,500
$ 24,100	$ 44,120,25	$ 47,300	$ 43,780

ملاحظة : استرشد بالسؤال رقم (٦-٧١) مثلا لمعرفة الإجراءات والدوال التي تحتاجها.

٦-٧٤ عدل البرنامج السابق بحيث تكون الزيادة قدرها ٧٪ للمرتبات التي هي أقل من 30,000 $ و ٤٪ للمرتبات التي هي أكثر من 40,000 $ و ٥,٥٪ لباقي المرتبات. اكتب دالة جديدة تحدد النسبة المئوية للزيادة. وأيضا بالنسبة لكل عضو هيئة تدريس اطبع النسبة المئوية للزيادة وكمية الزيادة.

٦-٧٥ يصعب على المرضى الذين يتعاطون أنواعا كثيرة من الأدوية تذكُّر متى يتناولون هذه الأدوية. نفرض انك أعطيت مجموعة الوصفات الطبية التالية. اكتب برنامجا

يطبع جدولا يعطي الأدوية التي يجب تناولها في أي ساعة معينة. استخدم متغيرا عدادا Hour للتعبير عن الأربع والعشرين ساعة خلال اليوم ، واستخدم إجراءات لطباعة العناوين والوصفات الطبية.

الدواء	مواعيد تعاطي الدواء
Iron	0800 , 1200 , 1800
Antibiotic	كل ٤ ساعات ابتداء من الساعة 0400
Vitamin	0800 , 2100
Calcium	1100 , 2000

٦-٧٦ مجلة شهرية تريد برنامجا لطباعة رسائل تجديد وإلغاء renewal and cancellation notices اشتراكات مشتركيها (subscribers). اكتب برنامجا يستخدم إجراءات كلما كان ذلك أفضل ، بحيث يقرأ البرنامج أولا الشهر الحالي (current month) (وهو عدد بين ١ و ١٢) والسنة الحالية (current year) ، ثم يقرأ لكل مشترك ٤ بيانات وهي :

رقم الحساب ، وشهر وسنة بداية الاشتراك ، وعدد السنوات المدفوع لها الاشتراك. اقرأ كل مجموعة من بيانات الاشتراك ثم اطبع رسالة تجديد الاشتراك إذا كان الشهر الحالي هو شهر انتهاء الاشتراك (expiration) أو الشهر السابق له ، ورسالة إلغاء الاشتراك إذا كان الشهر الحالي بعد شهر انتهاء الاشتراك.

نموذج للبيانات المدخلة :

96 10 : الشهر الحالي أكتوبر عام ١٩٩٦

3 94 4 1364 : الحساب رقم ١٣٦٤ بدأ صاحبه الاشتراك في أبريل عام ١٩٩٤ ولمدة ثلاثة أعوام.

٦-٧٧ 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 إجراء يطبق هذه الطريقة بحيث يشتمل الإجراء على ثلاثة وسطاء : الوسيط الأول : عدد حقيقي موجب (المطلوب إيجاد جذره التربيعي) ، والوسيط الثاني :

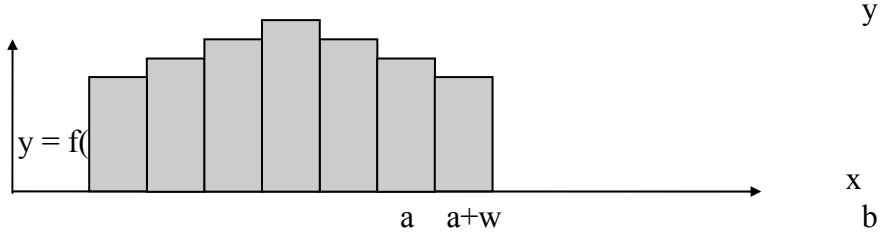
تخمين ابتدائي للجذر التربيعي ، الوسيط الثالث : النتيجة المحسوبة (الجذر التربيعي المطلوب).

التخمين الأول سيكون هو القيمة الابتدائية للمتغير LG. ثم يحسب الإجراء قيمة NG باستخدام العلاقة التكريرية ، ثم يوجد الفارق بين NG و LG ليتحقق ما إذا كانت القيمتان متساويتين تقريبا. فإن كانتا كذلك فإننا نخرج من الإجراء 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

٦-٧٨ نستطيع تقريب المساحة تحت المنحنى $y = f(x)$ بواسطة تقسيم المساحة إلى عدد من المستطيلات ، ثم حساب مجموع مساحات هذه المستطيلات. الشكل التالي مثال لطريقة " نقطة المنتصف " (midpoint method) لحساب المساحة وقد سميت الطريقة بهذا الاسم لأن المنحنى يقطع كل مستطيل عند منتصفه (مقاسا في اتجاه محور x) ، وتكون مساحة كل مستطيل عبارة عن ضرب العرض الثابت w في قيمة الدالة عند نقطة المنتصف (midpoint) (انظر الشكل).



مساحة المستطيل الذي له نقطة نهاية يسرى x_1 هي $w*f(x_1 + w/2)$.

إذا كانت الفترة [a , b] مقسمة إلى عدد n من المستطيلات ، فإن المساحة تحت المنحنى يمكن أن تمثل بالمجموع الآتي :

$$\text{Area} = w \sum_{i=0}^{n-1} f(a + i * w + w/2)$$

حيث w تساوي (b - a) / n ، والمستطيل الأول يبدأ عند x = a ، والثاني عند x = a + w ، والثالث عند x = a + 2w ، وهكذا.

اكتب برنامجا يستخدم طريقة " نقطة المنتصف " لإيجاد المساحة تحت المنحنى (قيمة التكامل المحدود) للدالة $f(x) = -3x^2 + 2x + 4$ على الفترة [3 , 2-] . اختبر البرنامج مستخدما قيما مختلفة لـ n. لاحظ أنه كلما زادت قيمة n ، كان التقريب أفضل.

٦-٧٩ أ) اكتب دالة Function Avg (X , n) تحسب القيمة المتوسطة Avg لعناصر منظومة X مكونة من قيم حقيقية عددها n.

$$Avg = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

ب) اكتب إجراء Procedure Big (A , m , key , count , sum) يستقبل منظومة A مكونة من قيم حقيقية عددها m ، ثم يوجد 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 رئيسيا يقرأ الأرقام التعريفية ودرجات ٤٠ طالبا وذلك في منظومتين : ID (أعداد صحيحة) و score (أعداد حقيقية) على الترتيب. ثم يستخدم بكفاءة البرنامجين الفرعيين السابقين لإيجاد وطباعة :

(i) الدرجة المتوسطة av للأربعين طالبا.

(ii) عدد الطلاب المتفوقين ngood ، وهم الحاصلون على درجة أعلى من الدرجة المتوسطة av.

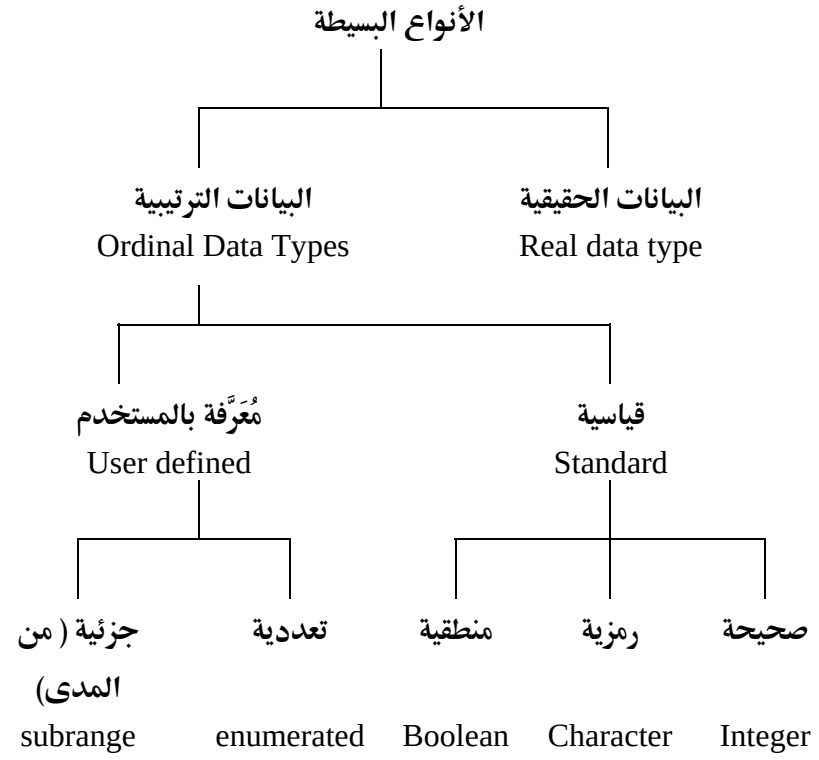
(iii) الدرجة المتوسطة للطلاب المتفوقين avgood.

ملحق (أ)

الأنواع البسيطة للبيانات

Simple Data Types

يحدد نوع البيانات مجموعة القيم أو الثوابت التي يمكن للمتغير أن يأخذها. والأنواع الأساسية للبيانات في لغة الباسكال هي الأنواع البسيطة ، والتي يوضحها الشكل التالي :



أنواع البيانات الترتيبية (Ordinal Data types)

يعد هذا النوع مجموعة قيم محدودة ومرتبة أو قابلة للترتيب ، ولذا فلها قيمة صغرى (minimum value) **وقيمة عظمى** (maximum value) **، ويمكن المقارنة بين عناصرها باستخدام المؤثرات العلاقية**

< , <= , = , <> , > , >=

وبالطبع يجب أن يكون طرفا المؤثر من نوع واحد.

وكما يبين الشكل السابق فإن البيانات الترتيبية إما قياسية أو مُعَرَّفَة بالمستخدم.

(أ) **البيانات القياسية** ، وهذه تنقسم إلى ثلاثة أنواع :

(i) صحيحة ، والفصل الثاني يتحدث عنها بتفصيل.

(ii) منطقية ، والفصل الثالث يتحدث عنها بتفصيل (ولاحظ أنه من المصطلح عليه أن False < True).

(iii) البيانات الرمزية Character (CHAR) Data Type.

وكل رمز من الرموز التي يتعامل معها الحاسب يقابله عدد صحيح خاص به ، أي يقابله عدد في شفرة خاصة بالجهاز (unique machine code). وبالتالي فيمكن ترتيب الرموز بناء على تمثيلها العددي هذا ، كما أنه بناء على هذا الترتيب المقابل للتمثيل العددي يمكن استخدام المؤثرات العلاقية , = , =< , < ... . وبالطبع يتغير هذا الترتيب من جهاز لآخر بناء على الشفرة المستخدمة ، ويسمى هذا الترتيب أو التسلسل لمجموعة الرموز كلها : " السلسلة المقارنة " (collating sequence) .

ومن أمثلة هذه الشفرات "الشفرة القياسية الأمريكية لتبادل المعلومات " (American Standard Code for Information Interchange) والتي تكتب اختصارا ASCII ، والجدول التالي المقابل لهذه الشفرة يعطي لكل رمز (حرف أو رقم رمز خاص) عددا خاصا به (كلمة شفرة).

والقيمة (Value) ذات النوع الرمزي (CHAR) هي عنصر في هذه المجموعة المرتبة (السلسلة) من الرموز ، ويكتب رمزا بين حاصرتين كما في الأمثلة التالية :

'A' 'd' '{' '/' '*'

وأما الرمز الخاص : الحاصرة ' فإنه يكتب حاصرتين متتاليتين ، هكذا.

''''

والمتغير الرمزي (Character variable) **يجب أن يعلن عنه في قسم الإعلان عن المتغيرات باستخدام الكلمة القياسية المحجوزة** CHAR **، مثل :**

```
VAR  x , y , letter : CHAR ;
```

قيمة ASCH	الرمز Character	قيمة ASCH	الرمز Character	قيمة ASCH	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

ويمكن أن تسند للمتغير الرمزي قيمة ما إما عن طريق :

عبارة إسناد ، مثل X := 'd' أو عبارة إدخال مثل READLN (X).

ويلاحظ أنه في الحالة الأخيرة ، أي عند ظهور المتغير الرمزي في قائمة عناصر مدخلة ، فإن عنصر البيانات المقابل (أي الرمز المقابل) يعطى في البيانات بدون الحاصرتين.

وعند طباعة قيمة المتغير الرمزي فإن الحاصرتين الخارجتين لا تطبعان ، وعند طباعة القيمة '''' التي تمثل حاصرة ، فإنها تطبع حاصرة واحدة. وعند طباعة المخرجات المصاغة (formatted output) فإن الرمز يطبع أقصى يمين المجال المسموح به ، فمثلا أمر الطباعة

WRITELN ('A' : 5)

يؤدي إلى طباعة :

ΔΔΔΔA

وفي الصيغة القياسية للغة الباسكال – أيا كان الجهاز المستخدم أو الشفرة المستخدمة – فإن :

(أ) الأرقام :

'0' , '1' , … , '9'

يجب أن تكون مرتبة الترتيب المعتاد ، ومتعاقبة (أي متتابعة دون وجود رموز أخرى بينها).

(ب) الحروف الكبيرة

'A' , 'B' , … , 'Z'

إذا وجدت ، يجب أن تكون مرتبة ترتيبا أبجديا ، ولكن لا يشترط فيها التعاقب (يسمح مثلا بوجود أرقام أو رموز خاصة بين حرفين متتاليين).

(ج) الحروف الصغيرة

'a' , 'b' , … , 'z'

إذا وجدت ، يجب أن تكون مرتبة أبجديا ، ولا يشترط في حالتها التعاقب.

وبناء على هذا فالعلاقات التالية صحيحة TRUE

'A' < 'E' , '8' > '5' , 'y' > 'x'

أما العلاقات التالية فقد تكون صحيحة أو خاطئة تبعا للجهاز أو الماكينة المستخدمة (machine used) **أو الشفرة المستخدمة.**

'8' < 'A' , 'G' > 'g' , '+' < '*'

وللحصول على السلسلة المقارنة (collating sequence) **لأي ماكينة ، يمكننا استخدام :**

الدالة القياسية CHR

تعرَّف هذه الدالة كما يلي :

CHR (i): **هي قيمة الرمز المصاحبة للعدد الصحيح** i **(أي للعدد الترتيبي** i**).**

وباختصار CHR (i) **: تعطي الرمز الذي ترتيبه** i**. وعلى سبيل المثال العبارة التالية يؤدي تنفيذها إلى الحصول على السلسلة المقارنة كاملة :**

```
FOR i := 0 TO 255 DO
    WRITELN (i : 3 , CHAR(i) : 3) ;
```

ولمعرفة موضع أو ترتيب رمز ما في السلسلة المقارنة ، يمكننا استخدام :

الدالة القياسية ORD

تعرف هذه الدالة كما يلي :

ORD (x) : **العدد الترتيبي** (ordinal number) **للرمز** x **في السلسلة المقارنة ، أي ترتيب الرمز** x.

أي أن الدالة ORD **هي عكس الدالة** CHR **، حيث أن**

CHR(i) : **تعطي الرمز** (x) **الذي ترتيبه** i **، بينما**

ORD (x) : **تعطي الترتيب** (i) **للرمز** x

(ب) البيانات المعرفة بالمستخدم (User defined data types)

وهذه نوعان :

(i) **البيانات التعددية** enumerated data type

وهي عبارة عن مجموعة مفردات من البيانات من نوع واحد ، وتحمل اسما عاما لها ، وتعرف في قسم الإعلانات في البرنامج بالصيغة العامة التالية :

```
TYPE
    name = (data1 , data2 , … , data n)
```

حيث :

name : **يمثل اسما تعريفيا عاما لهذه المجموعة المعددة من البيانات.**

data1, …, data n : **قائمة أسماء تعريفية لمفردات مجموعة البيانات ، والتي تعرَّف بترتيب تصاعدي من اليسار إلى اليمين جميع قيم (ثوابت) هذا النوع.**

ملاحظة : من الممكن أن نعرف أكثر من مجموعة واحدة من البيانات التعددية باستخدام كلمة TYPE **واحدة – في قسم الإعلانات – مع كتابة فاصلة منقوطة بين أي مجموعتين متتاليتين ، كما في المثال التالي.**

مثال :

```
TYPE
    DAYS = (Fri , Sat, Sun, Mon, Tue, Wed, Thu) ;
    Months = (Moharram, Safar, … , ThulHijja) ;
                {تكتب أسماء الاثني عشر شهرا}
    SacredMonths = (Moharram, Rajab, Thulquida, ThulHijja) ;
                { الأربعة أشهر الحرم }
```

ملاحظات على البيانات التعددية وصيغتها العامة :

* **العدد الترتيبي** (ordinal number) **لأول اسم تعريفي يظهر في قائمة الأسماء التعريفية هو صفر ، يليه العدد ١ لثاني اسم تعريفي ، وهكذا. فمثلا العدد الترتيبي ليوم الاثنين هو ٣ ، والعدد الترتيبي ليوم الخميس هو ٦. وعموما العدد الترتيبي للاسم التعريفي رقم** i **في الظهور في قائمة الأسماء التعريفية هو** i-1 .
* **لا يمكن أن يظهر أي اسم تعريفي في أكثر من قائمة واحدة.**

* نوع البيانات التعددية المنطقية المعرفة سابقا (predefined boolean type) يعرَّف هكذا :

```
TYPE
BOOLEAN = (FALSE , TRUE) ;
```

* أي متغير من النوع التعددي يمكن أن تسند له فقط إحدى قيم (ثوابت) القائمة المقابلة للأسماء التعريفية ، ولا يكون ذلك إلا عن طريق عبارة إسناد v := e حيث e ثابت أو متغير من النوع نفسه.
* أي متغير من النوع التعددي لا يمكن أن يظهر في قائمة أسماء مدخلات عبارة READ أو عبارة READLN ، أو في قائمة أسماء مخرجات عبارة WRITE ، أو عبارة WRITELN.

(ii) البيانات الجزئية (من المدى) (Subrange Type)

وهذه تسمى جزئية لأنها تمثل جزءا من مجموعة من مفردات البيانات القياسية أو التعددية ، أي أن البيانات الجزئية تعرَّف بأنها مجموعة جزئية من مدى بيانات ترتيبية (قياسية أو تعددية) معرَّفة سابقا. ويسمى نوع هذه البيانات المعرفة سابقا : "النوع المُضيف للمدى الجزئي" (host type of the subrange).

وتعرَّف بيانات المدى الجزئي في قسم الإعلانات الخاص بالنوع TYPE بالصيغة العامة التالية :

```
TYPE
    name = minvalue .. maxvale ;
```

وتعني أن المتغير من النوع name يمكن أن تسند إليه أي قيمة من بين مجموعة قيم (ثوابت) النوع المُضيف المحصورة بين القيمة minvalue والقيمة maxvalue ، بما في ذلك هاتان القيمتان (لاحظ وجود نقطتين في هذه الصيغة بين القيمة الأولى والقيمة الأخيرة).

مثال :

```
TYPE
    Days = (Fri, Sat, Sun, Mon, Tue, Wed, Thu) ;
    Workdays = Sat..Wed;
```

وهذا يعني أن أي متغير من النوع DAYS يمكن أن يسند إليه أي يوم من أيام الأسبوع كلها : من الجمعة إلى الخميس (بيانات تعددية) ، بينما أي متغير من النوع Workdays يمكن أن يسند إليه أي يوم من السبت إلى الأربعاء فقط (بيانات جزئية من البيانات التعددية).

مثال آخر :

```
CONST
    max = 100 ;
TYPE
    Count = 0..max ;
    Temp = -18..80 ;
    SmallLet = 'a' .. 'z' ;
    Months = (Moh, Saf, Rab1, Rab2, Jum1, Jum2, Raj, Shb,
              Ram, Shl, ThQ, ThH) ;
    Hajjmonths = Sha .. ThH ;
```

في هذا المثال نلاحظ ما يلي :

- كل من البيانات الجزئية من النوع Count والبيانات الجزئية من النوع Temp تعد مدى جزئيا (subrange) من الأعداد الصحيحة (INTEGER) التي هي بيانات قياسية.
- البيانات الجزئية من النوع SmallLet (الحروف الصغيرة) تعد مدى جزئيا من الرموز (CHAR) والتي هي بيانات قياسية.
- البيانات الجزئية من النوع Hajjmonths (أشهر الحج) تعد مدى جزئيا من البيانات التعددية من النوع Months (الشهور).

ملاحظات على البيانات الجزئية وصيغتها العامة :

* يجب أن تتحقق العلاقة التالية بين القيمة الأولى والقيمة الأخيرة للمدى الجزئي :

```
minvalue <= maxvalue
```

* لا توجد بيانات جزئية من النوع الحقيقي REAL لأن البيانات الحقيقية ليست قياسية ولا تعددية.

* البيانات الجزئية (وكذلك التعددية) تجعل البرنامج أسهل عموما في القراءة ، وقد تؤدي إلى شغل حيز أقل في الذاكرة.

الدوال القياسية ORD , PRED , SUCC

هناك ثلاث دوال قياسية على البيانات الترتيبية ، أي أن وسيط أي دالة منها من النوع الترتيبي (ordinal type) ، وقد مرت معنا سابقا الدالة ORD حيث ORD (x) تعطي ترتيب x .

وفيما يلي تعريف الدالتين الأخريتين :

PRED (x) : تعطي القيمة الترتيبية (ordinal value) السابقة مباشرة (predecessor) للقيمة x في القائمة المناظرة.

SUCC (x) : تعطي القيمة الترتيبية التالية مباشرة (successor) لـ x في القائمة المناظرة.

وبالطبع لا توجد قيمة سابقة لأول قيمة ولا قيمة لاحقة (تالية) لآخر قيمة في القائمة.

مثال :

```
TYPE
    Days = (Fri, Sat, Sun, Mon, Tue, Wed, Thu) ;
ORD (Mon) = 3
PRED (Mon) = Sun
SUCC (Mon) = Tue
ORD (Fri) = 0
SUCC (Fri)  = Sat
```

ملاحظة : كل من الحروف الكبيرة $A \rightarrow Z$ والحروف الصغيرة $a \rightarrow z$ بيانات ترتيبية (ordered) ولكنها ليست بالضرورة تعاقبية (consecutive) فمثلا 'j' > 'i' ولكن ليس ضروريا أن 'j' = SUCC ('i') .

استخدام عبارات التحكم والتكرار مع البيانات الترتيبية :

يجوز استخدام عبارات التحكم والتكرار المذكورة في الفصلين الثالث والرابع مع البيانات التي من النوع الترتيبي.

- فمثلا يجوز استخدام عروة FOR بحيث يكون متغير التحكم في العروة ووسطاء العروة من أي نوع ترتيبي (على أن يتحقق التواؤم بين الجميع).
- ويجوز استخدام بنية CASE بحيث يكون (التعبير) المنتقِي (selector) والثوابت أسماء الحالة (case labels) من أي نوع ترتيبي (النوع نفسه).

مثال :

إذا اشتمل قسم الإعلانات على التعريفات التالية :

```
TYPE
Days = (Fri, Sat, Sun, Mon, Tue, Wed, Thu) ;
VAR
    x , y , day : Days ;
    Score       : REAL ;
    Switch      : BOOLEAN ;
    Operator    : CHAR ;
```

فيجوز أن يشتمل قسم العبارات على العبارات التالية :

```
x := Fri ;
y := SUCC (x) ;
FOR x := Sat TO Wed DO
        BEGIN
        .............
        .............
        .............
        END ;
WHILE (x <= Wed) AND Switch DO statement ;
IF x <> y THEN Statement ;
        REPEAT
        .............
        .............
        .............
        x = y ;      UNTIL
CASE ROUND (Score) OF
    0 .. 60   :  WRITELN ('Fail') ;
    61 .. 70  :  WRITELN ('Pass') ;
    71 .. 80  :  WRITELN ('Good') ;
    81 .. 90  :  WRITELN ('Very Good') ;
    91 .. 100 :  WRITELN ('Distinction') ;
END ;
```

ونظرا لأنه لا يُسمح بظهور متغير من النوع التعددي في عبارة طباعة ، فيمكن استخدام بنية CASE للتغلب على هذه الصعوبة ، كما في المثال التالي :

```
CASE Day OF
    Fri : WRITELN ('Friday') ;
    Sat : WRITELN ('Saturday') ;
    Sun : WRITELN ('Sunday') ;
    Mon : WRITELN ('Monday') ;
    Tue : WRITELN ('Tuesday') ;
    Wed : WRITELN ('Wednesday') ;
    Thu : WRITELN ('Thursday') ;
END;
```

مثال آخر :

```
CASE Operator OF
    '+' : WRITELN ('Add') ;
    '-' : WRITELN ('Subtract') ;
    '*' : WRITELN ('Multiply') ;
    '/' : WRITELN ('Divide') ;
END ;
```

ملاحظة عامة على تعريف البيانات الترتيبية :

يجوز في تعريف البيانات الترتيبية في قسم الإعلانات أن نجمع بين الإعلان عن النوع TYPE والإعلان عن المتغيرات VAR في إعلان واحد كما في المثال التالي :

مثال :

المجموعات الثلاث التالية للإعلان جميعها متكافئة :

(أ) المجموعة الأولى

```
TYPE
    Grade = (Fail , Pass , Good) ;
    Temp = -18 .. 80 ;
    Count = 1 .. 120 ;
VAR
    G : Grade ;
    t1 , t2 : Temp ;
    i , j , k : Count ;
```

(ب) المجموعة الثانية :

```
VAR
    G  : (Fail , Pass , Good) ;
    t1 , t2 :  -18 .. 80 ;
    i , j , k :  1 .. 120 ;
```

(ج) المجموعة الثالثة :

```
TYPE
    Grade = (Fail , Pass , Good) ;
VAR
    G  : Grade ;
    t1 , t2 :  -18 .. 80 ;
    i , j , k :  1 .. 120 ;
```

مثال ١ :

ما نوع وما قيمة كل من التعبيرين التاليين :

(أ) SUCC (CHR (ORD ('B') + 2)) ;

(ب) (0 < 1) OR (5 <= 12) AND ('A' > 'G')

الحل :

(أ) Succ ('D') = 'E'

(ب) true or true and false = true or false
= true

مثال ٢ :

أ) **اكتب إجراء** encode **يستقبل منظومة رموز** (character array) **ثم يعيدها بعد تشفيرها ، بأن يستبدل بكل رمز الرمز الذي يلي الذي يليه (أي الرمز الذي يأتي بعده والفرق بينهما موضعان) في السلسلة المقارنة** (collating sequence).

مثلا : إذا استقبل المنظومة MATHEMATICS

فإنه يعيد / يخرج المنظومة OCVJGOCVKEU

حيث أن رموز المنظومة المخرجة تختلف بحرفين (2 letters) (مزاحة موضعين) عن رموز المنظومة المستقبلة في السلسلة الأبجدية (alphabetical sequence).

ويمكن استدعاء الإجراء بالعبارة

encode (word , n) ;

حيث word : منظومة رموز من نوع معرف بالمستخدم list ،

و n : عدد رموز المنظومة word.

ب) اكتب إجراء decode يستقبل منظومة رموز مشفرة (encoded character array) ، ثم يعيدها بعد فك / ترجمة شفرتها ، بأن يستبدل بكل رمز الرمز الذي يسبق الذي يسبقه في السلسلة المقارنة.

مثلا : إذ استقبل المنظومة OCVJGOCVKEU

فإنه يعيد المنظومة MATHEMATICS

ويمكن استدعاء الإجراء بالعبارة

decode (word , n) ;

حيث word : منظومة رموز من نوع معرف بالمستخدم list ،

و n : عدد رموز المنظومة word.

الحل :

أ)

```
procedure encode (var word : list ; n : integer) ;
    {this procedure encodes an array of characters.}
                                                    var
                                                        i : integer ;
                                                    begin
                                                        for i := 1 to n do
                                                    word [i] := succ (succ (word [i]))
                                                    end ;
```

ب)

```
procedure decode (var word : list ; n : integer) ;
    {this procedure decodes an array of characters.}
                                                    var
                                                        i : integer ;
                                                    begin
                                                        for i := 1 to n do
```

```
word [i] := pred (pred (word [i]))
end ;
```

مثال ٣ :

أعد حل السؤال (٥-١٣) بقراءة سبع درجات حرارة فقط تمثل درجات حرارة أحد الأسابيع (درجة لكل يوم) ، وطباعة جدول يعطي اسم كل يوم من أيام هذا الأسبوع ، ودرجة حرارته ، وانحراف هذه الدرجة عن الدرجة المتوسطة. استخدم البيانات التعددية في تعريف نوع أيام الأسبوع.

الحل :

```
PROGRAM Temperatures ;
TYPE
WEEK = (SAT , SUN , MON , TUE , WED , THU , FRI) ;
ARRAY_TYPE1 = ARRAY [WEEK] OF INTEGER ;
ARRAY_TYPE2 = ARRAY [WEEK] OF REAL ;
VAR
DAY : WEEK ;
{TO STORE THE TEMPERATURE READINGS} TEMP : ARRAY_TYPE1 ;
DIFFERENCE : ARRAY_TYPE2 ; {TO STORE THE DIFFERENCES}
{SUM OF TEMPERATURE READINGS} SUM : INTEGER ;
{AVERGE TEMPERATURE OF THE WEEK} AVERAGE : REAL ;
BEGIN
SUM := 0 ;
WRITELN ('ENTER TEMPERATURE READINGS OF THE WEEK : ') ;
FOR DAY := SAT TO FRI DO
BEGIN
READ (TEMP [DAY]); {READS A TEMPERATURE}
SUM := SUM + TEMP [DAY] {ACCUMULATES TEMPERATURE}
END ;
AVERAGE := SUM /7 ; {COMPUTES AVERAGE}
FOR DAY := SAT TO FRI DO {STORES THE DIFFERENCES}
DEFFERENCE [DAY] := TEMP [DAY] - AVERAGE ;
```

```
                                        {DISPLAYING OUTPUT}
WRITELN (′AVERAGE TEMPERATURE IS′ , AVERAGE : 2:2) ;
                                                WRITELN ;
WRITELEN (′DAY′:5,′TEMPERATURE′:16, ′DIFFERENCE′:16);
                WRITELN (′---′:5, ′-----------′ : 16, ′----------′: 16) ;
                                    FOR DAY := SAT TO FRI DO
                                                    BEGIN
                        {DISPLAYING DAY}  CASE DAY OF
                                    :  WRITE (′SAT′ : 5) ;      SAT
                                    :  WRITE (′SUN′ : 5) ;      SUN
                                   :  WRITE (′MON′ : 5) ;     MON
                                    :  WRITE (′TUE′ : 5) ;      TUE
                                   :  WRITE (′WED′ : 5) ;     WED
                                    :  WRITE (′THU′ : 5) ;      THU
                                     :  WRITE (′FRI′ : 5) ;       FRI
                                            {CASE}       END ;
WRITELN(TEMP [DAY]: 11,DIFFERENCE [DAY]:18:2)
                                                    END ;   {FOR}
                            WRITELN (′-----------------------------------′)
                                                    END.

                                                Program Run
                            ENTER TEMPERATURE READINGS OF THE WEEK :
                                    16     20     14     19     18     17     15
                                AVERAGE TEMPERATURE IS 17.00
```

DAY	TEMPERATURE	DIFFERENCE
SAT	15	-2.00
SUN	17	0.00
MON	18	1.00
TUE	19	2.00
WED	14	-3.00
THU	20	3.00
FRI	16	-1.00

مثال ٤ :

أ) يقبل أي عدد صحيح القسمة على ٩ إذا كان مجموع أرقامه يقبل القسمة على ٩. اكتب برنامجا لتحديد ما إذا كانت الأعداد التالية تقبل القسمة على ٩ أم لا.

[إرشاد : راجع استخدام Mod في السؤال رقم ٤-١٩ لإيجاد أرقام عدد صحيح ومن ثم إيجاد مجموعها]

N = 154368 , N = 621594 , N = 123456

ب) أعد حل الجزء (أ) عن طريق قراءة كل رقم (على حدة) من أرقام العدد N المراد اختباره ، وذلك باعتبار هذا الرقم متغيرا اسمه Digit من النوع الرمزي (الحرفي) Char . ثم احسب مجموع القيم العددية لهذه الأرقام.

[إرشاد : القيمة العددية للمتغير Digit (الذي هو من النوع الرمزي) تعطي بالتعبير :

Ord (Digit) - Ord ('0')

الحل : أ)

```
program Nines ;
{This program determines whether numbers read from
Input are divisible by 9. The program terminates after reading
sentine1 0.}
var
{input - number to test}      Number : LongInt ;
{interm - 9 divides Number?}  IsDivisible : Boolean ;
procedure TestDivisibility
(Number {input} : LongInt ;
var IsDivisible {output} : Boolean) ;
{Determine whether Number is divisible by 9.
Pre : Number > 0.
IsDivisible = True if and only if Number is Post :
divisible by 9.}
Var
{of Number} SumofDigits : Integer ;
Rest : LongInt
{succesive quotients
dividing Number by 10}
begin {TestDivisibility}
Sumofdigits := 0 ;
Rest := Number ;
while Rest > 0 do
begin
{Add next digit}
SumOfDigits := SumOfDigits + Rest mod 10 ;
```

```
                Rest := Rest div 10
                    {while}  end ;
        IsDivisible := (SumOfDigits mod 9 = 0)
                    {TestDivisibility}  end ;

procedure GetNumber (var Number {output} ; LongInt) ;
        {Read a nonnegative number from Input.}
                    Begin {GetNumber}
        Write (′Enter a nonnegative integer : ′) ;
                    ReadLn (Number) ;
                  while Number < 0 do
                        begin
    WriteLn (′Number cannot be negative.′) ;
     Write (′Enter a nonnegative integer : ′) ;
                  ReadLn (Number)
                    {while}   end
                  {GetNumber}  end ;
                        {Digits}        begin
  WriteLn (′To terminate the program, enter zero after′ ,
                    ′the prompt.′) ;
                {Get first number to be tested}
                  GetNumber (Number) ;
{Repeatedly test number and get another, until sentinel.}
                  While Number <> 0 do
                        begin
     TestDivisibility (Number, IsDivisible) ;
                  if Isdivisible then
  WriteLn (Number:1, ′is divisible by 9′)
                        else
 WriteLn (Number:1, ′is not divisible by 9′) ;
                  GetNumber (Number)
                    {while}   end.
                                    end.
```

ب)

```
                Program Nines ;
                                    var
        {input - number to test}        Number : Integer ;
 {interm - 9 divides Number?} IsDivisible : Boolean ;
                        procedure TestDivisibility
      (var Isdivisible {output} : Boolean) ;
```

```
{Determine whether a number read from standard input
                                        is divisible by 9.
  The user enters a legal positive integer at the terminal,  Pre :
                                  terminated by a period.
    IsDivisible = True if and only if the number is   Post:
                                    divisible by 9.}
                                                          Var
{Sum of digits of Number}        SumOfDigits : Integer ;
      {Digits read from Input}              Digit : Char;
                                             begin {TestDivisibility}
                                           SumOfDigits := 0 ;
          Write (′Enter a positive number, terminated by a′ ,
                        ′period and RETURN : ′) ;
                                                Read (Digit) ;
                                          while Digit <> ′.′ do
                                                  begin
                    if Digit in [′0′ .. ′9′] then
          SumOfDigits := SumOfDigits +
            Ord (Digit) - Ord (′0′) ;
                                 Read (Digit)
                                       {while}  end ;
                                             ReadLn ;
                IsDivisible := (SumOfDigits mod 9 = 0)
                                    {TestDivisibility}  end ;

                                          {Digits}        begin
                              TestDivisibility (IsDivisible) ;
                                          if IsDivisible then
              WriteLn (′The number is divisible by 9′)
                                                        else
          WriteLn (′The number is not divisible by 9′) ;
                                                  {Digits}  end.
```

ملحق (ب)

تيربو باسكال
Turbo Pascal

تيربو باسكال هو اسم صيغة حديثة للغة الباسكال صممت لتلائم الحاسبات الشخصية ، فهي تمد المبرمج بصيغة تبادلية (interactive) وتؤدي إلى سرعة كبيرة بالنسبة للترجمة (compilation) والتنفيذ (execution) في هذه الحاسبات. وتتبع هذه الصيغة قواعد لغة الباسكال العادية (القياسية Standard Pascal) وتضيف إليها عدة زيادات مفيدة ، وبعض التعديلات والتحسينات ، وفيما يلي في هذا الملحق نتناول بعض هذه الإضافات والتعديلات التي تتعلق بفصول هذا الكتاب.

أساسيات اللغة

* **يبلغ مدى الأعداد الصحيحة (integers) في لغة التيربو باسكال من 32768- إلى 32767.**
* **بالنسبة للأعداد الحقيقية (real) يتغير الأس (exponent) من 38- إلى 38 ، ويخزن العدد الحقيقي باستخدام ١١ رقما معنويا.**
* **بالنسبة للمتغيرات يمكن أن يصل طول الاسم التعريفي (identifier) للمتغير إلى ١٢٧ رمزا (characters) أي أن عدد حروف وأرقام الاسم التعريفي يمكن أن يصل إلى ١٢٧ ، ويسمح للرمز الخاص (_) أن يكون ضمن رموز الاسم التعريفي ، فمثلا الاسم Zakat_ul_mal يسمح به في لغة التيربو باسكال.**
* **الثابت القياسي MAXINT الذي يمثل قيمة أكبر عدد صحيح مسموح به يساوي ٣٢٧٦٧ في التيربو باسكال.**

* في لغة التيربو باسكال يوجد ثابت قياسي يدعى PI وهو غير موجود في لغة الباسكال العادية ، وقيمته تساوي 3.1415926536E+00 وهو يقابل الرمز الرياضي Π.

* لإرسال المخرجات output إلى آلة الطباعة printer نستخدم عبارة الطباعة

WRITE (LST , output list) ;

أو عبارة الطباعة

WRITELN (LST , output list) ;

* في التيربو باسكال تطبع القيمة الصحيحة (integer value) كما هي بدون أي فراغات إضافية بجانب العدد ، فإذا اشتمل أمر الطباعة مثلا على طباعة عددين صحيحين بدون أن يشتمل أمر الطباعة على ترك فراغ بين العددين طبع العددان متجاورين فيظهر أنهما عدد واحد ، وليس هو المطلوب. مثلا إذا كانت x = 123 وكان y = 4567 فإن أمر الطباعة WRITE (x,y) يؤدي إلى طباعة العدد 1234567.

* وتطبع القيمة الحقيقية (أو تظهر على الشاشة) في الصيغة الأسية في مجال يتسع لثمانية عشر موضعا (رمزا) في الصورة

$\nabla\nabla\underbrace{dd.....d}E \pm dd$ للأعداد الموجبة

والصورة

$\nabla - d.\overbrace{dd.....d}E \pm dd$ للأعداد السالبة

حيث الحرف d يشير إلى digit أي رقم.

فمثلا أمر الطباعة

WRITE (PI) ;

في التيربو باسكال يؤدي إلى طباعة ما يلي :

$\nabla\nabla 3.1415926536E + 00$

* يسمح التيربو باسكال بأن تكون علامات (labels) أو أرقام العبارات (statement numbers) أسماء تعريفية (identifiers) بالإضافة إلى كونها أعدادا ، فيجوز مثلا أن نكتب :

```
LABEL 24, Start, Loop ;
.....
.....
BEGIN
.....
24 : t := u + 10.0 ;
.....
.....
Start : z := x + y ;
.....
.....
Loop : FOR j := 4 TO 20 DO
BEGIN
.....
.....
END ;
```

* **اسم البرنامج أو عنوانه** (program name / heading) **اختياري في التيربو باسكال بينما هو إجباري في الباسكال القياسي. وكذلك أسماء ملفات الإدخال والإخراج في عبارة اسم البرنامج اختيارية في التيربو باسكال.**
* **قسم الإعلانات في الباسكال القياسي يتكون من خمسة أجزاء :**

LABEL, CONST, TYPE, VAR, PROCEDURE & FUNCTION

وأي جزء منها يجب ألا يظهر أكثر من مرة واحدة فقط ، وبنفس هذا الترتيب المذكور ، بينما في التيربو باسكال في قسم الإعلانات يجوز أن يظهر أي جزء من هذه الأجزاء أي عدد من المرات وبأي ترتيب.

* **في التيربو باسكال يجوز أن يظهر قوسا عبارة التعليق {.....} داخل القوسين (*.....*) ، والعكس صحيح.**

عبارات التحكم

* **الصيغة العامة لبنية الحالة** CASE Structure **(أو عبارة الاختيار الانتقائي كما تسمي أيضا) في التيربو باسكال هي :** CASE - ELSE - END **، كما يلي :**

```
CASE e OF
L1 : s1 ;
L2 : s2 ;
  :
Ln : sn ;
     ELSE ;
     St
END ;
```

أي أنه في التيربو باسكال يسمح بكتابة ELSE **والعبارة أو الاختيار البديل** St **، بينما في الباسكال القياسي لا يسمح بكتابة** ELSE**. ويجوز ألا تشتمل عبارة الحالة في التيربو باسكال على عبارة** ELSE **، وفي هذه الحالة إذا لم تكن قيمة التعبير المنتقى** e **هي أي قيمة من قيم قوائم الثوابت** L1,L2, … , Ln **فإنه لا يتم تنفيذ شئ ولا يعد هذا خطأ ، بينما في الباسكال القياسي يعد هذا خطأ ، إذ يجب أن تكون قيمة التعبير** e **إحدى قيم قوائم الثوابت** L1, … , Ln**.**

مثال :

```
CASE TRUNC (Score / 10) OF
  : WRITELN ('Grade = A') ; 10,9
  : WRITELN ('Grade = B') ;   8
  : WRITELN ('Grade = C') ;   7
  : WRITELN ('Grade = D') ;   6
    ELSE
      WRITELN ('Grade = F') ;
    END ;
```

عبارات التكرار

عند الخروج العادي من عروة FOR

```
FOR v := initial TO final DO st ;
```

بعد انتهائها تكون قيمة v **(متغير التحكم في العروة) هي القيمة النهائية** final **بينما في الباسكال القياسي تكون قيمة** v **غير معرفة.**

البرامج الفرعية

فيما يلي بعض البرامج الفرعية (الإجراءات والدوال) الإضافية في التيربوباسكال ووظيفة كل منها :

أولا : الإجراءات :

CLRSCR : يمسح الشاشة ويضع المؤشر في بداية الصفحة (الركن الشمالي العلوي).

DELAY(i) : يؤخر تنفيذ البرنامج نحو i ميللي ثانية ، حيث i عدد صحيح فمثلا DELAY (4000) يؤخر البرنامج نحو ٤ ثواني.

DELLINE : يمسح السطر الحالي وينتقل إلى السطر التالي ، أي يمسح السطر الذي عنده المؤشر ، وبالتالي تتحرك جميع السطور السفلية مسافة سطر واحد إلى أعلى.

EXIT : إذا استخدم في برنامج فرعي فإنه يعيد التحكم إلى البرنامج الرئيسي ، وإذا استخدم في قسم العبارات في البرنامج الرئيسي فإنه ينهي تنفيذ البرنامج.

GOTO (x,y) : يحرك المؤشر إلى الإحداثي (x,y) حيث كل من x,y عدد صحيح ، فمثلا GOTO (2,5) يعنى وضع المؤشر في الموضع x = 2, y=5 على الشاشة. ونلاحظ أن x تمثل رقم العمود ، و y تمثل رقم الصف ، حيث

$$1 \le x \le 80 \text{ أو } 1 \le x \le 40$$

$$1 \le y \le 25$$

وحيث الإحداثي (1,1) هو موضع الركن الشمالي العلوي وفي حالة ما إذا كان x أو y خارج المدى المسموح به فإن المؤشر ينتقل إلى الموضع (1,1).

HALT : يوقف تنفيذ البرنامج ويرجع إلى نظام التشغيل.

INSLINE : يضيف سطرا خاليا في المكان الحالي للمؤشر ، وبالتالي تتحرك جميع السطور السفلية مسافة سطر واحد لأسفل.

ثانيا : الدوال :

FRAC (x) : تعطي الجزء الكسري من العدد (الحقيقي أو الصحيح) x ، وتكون النتيجة حقيقية. مثلا :

FRAC (4.25) = 0.25
FRAC (4) = 0.0

INT (x) : تعطي الجزء الصحيح من العدد x . أي أنها تؤدي نفس عمل الدالة TRUNC إلا أن النتيجة هنا تكون من النوع الحقيقي.

UPCASE(c) : تعطي الرمز العلوي المقابل للرمز c ، وإذا لم يوجد رمز علوي مقابل فإن الدالة تعطي نفس الرمز c ، فمثلا

UPCASE ('t') = 'T'
UPCASE ('*') = '*'

WHEREX : تعطي الإحداثي x لموضع المؤشر ، والنتيجة من النوع الصحيح.

$1 \le x \le 80$)أو($1 \le x \le 40$

WHEREY : تعطي الإحداثي y لموضع المؤشر ، والنتيجة من النوع الصحيح.

$1 \le y \le 25$

أما الدالة TAN (x) الموجودة في الباسكال العادي فإنها غير معرفة في التيربو باسكال ، ويمكن تعريفها كما يلي :

```
FUNCTION TAN (x : REAL) : REAL ;
BEGIN
   TAN := SIN (x) / COS (x)
END ;
```

ملحق (ج)

عبارة GOTO وأرقام العبارات

GOTO Statement and Statement Numbers/Labels

رقم العبارة هو عدد صحيح بدون إشارة ولا يزيد عن أربعة أرقام عشرية. ويجب ألا يكون لعبارتين الرقم نفسه. وليس من الضروري أن نعطي كل عبارة رقما. ورقم العبارة هو مجرد تعريف لها ، بحيث يمكن للعبارات الأخرى في البرنامج أن تشير إليها .. وأرقام العبارات أرقام اختيارية بمعنى أنه ليس من الضروري أن تسير وفق تسلسل معين ، فليس من الضروري أن يكون رقم أول عبارة 1 ، وإنما أي ترقيم اختياري للعبارات المتتالية في البرنامج يعد مقبولا (وليس شرطا أن يكون تصاعديا ، فمثلا قد تكون أرقام العبارات هكذا : (35,21,800,797,19,100).

وأي رقم عبارة يعطَى يجب أن يعلن عنه في الجزء الخاص بالإعلان عن أرقام العبارات (Label decleration part) **قبل استخدامه. وهذا الجزء يبدأ بالكلمة المحجوزة** LABEL **تليها قائمة هذه الأرقام وبحيث توجد فاصلة بين أي رقمين متتاليين ، كأن نكتب مثلا :**

```
LABEL 19 , 21 , 35 , 100 , 797 , 800
```

ورقم أي عبارة يكتب على يسارها تليه نقطتان ، مثل :

```
35 : Z := X + Y ;
```

وعموما قليلا ما نحتاج إلى أرقام العبارات في البرنامج.

والصيغة العامة لعبارة GOTO **هي :**

```
GOTO  n
```

حيث n **تمثل رقم** (label) **عبارة تنفيذية قبل أو بعد عبارة** GOTO **، وعبارة** GOTO **السابقة تعني إعطاء الأمر للحاسب ليذهب مباشرة ودون أي شرط لينفذ العبارة رقم** n **، وكمثال على عبارة الانتقال غير المشروط :**

```
GOTO  25
```

وعموما يفضل عدم استخدام عبارة GOTO في البرامج لأنه لا يسهل مع وجودها متابعة تسلسل خطوات البرنامج وتصحيح أخطائه.

مثال :

أعد حل مثال ٣–٨ باستخدام عبارة GOTO .

الحل :

```
IF TAS >= A THEN GOTO 3
                Z := 0.0 ;
            GOTO 100 ;
Z := TAS / 40.0 ;    3 :
WRITELN ('TAS = ' , TAS , ' Zakat = ' , Z) ;  100:
```

استخدمنا في هذا الحل عبارة الانتقال GOTO مرتين ، وكما ذكرنا سابقا فإن وجود هذه العبارة في ثنايا البرنامج يقلل من سهولة متابعة تسلسل خطوات البرنامج وتصحيح أخطائه ، ولذلك يفضل عدم استخدامها أو التقليل من استخدامها قدر الاستطاعة ، وتسمى البرمجة التي تخلو من هذه العبارات الانتقالية : البرمجة المبنية أو البرمجة البنائية (structured programming).

تمرينات عامــة

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 بلغة الباسكال.

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 بلغة الباسكال يقرأ سعر الجرام من الذهب الخالص p بالدينار ، ويحسب زكوات ألف شخص باتباع الخطوات التالية :

(أ) يقرأ بيانات كل شخص على سطر مستقل وهي :

NAME : اسم الشخص .
GOLD : كمية الذهب بالجرام التي حال عليها الحول.
STANDARD : عيار الذهب.
MONEY : قيمة النقود المدخرة بالدينار التي حال عليها الحول.

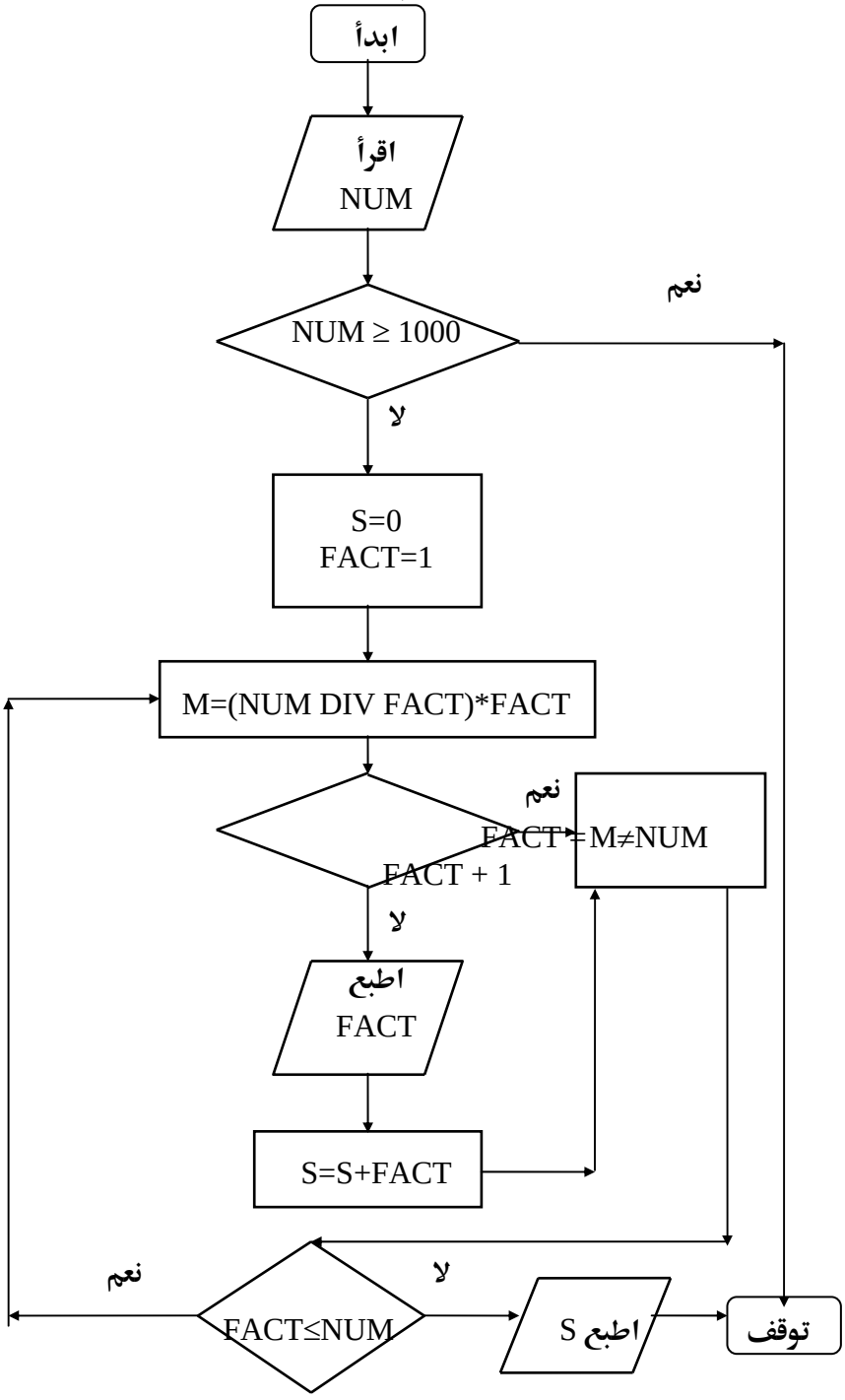
(ب)يحسب نصاب النقود بالدينار NB.

(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 :

عيار الذهب × كمية الذهب بالجرام
٢٤

(هـ) يحسب زكاة الذهب ZG بالجرام ، وهي تساوي ٢,٥٪ من كمية الذهب الخالص بالجرام إذا بلغ النصاب.



٣- الرسم المبين بالصفحة التالية يوضح خريطة سير عمليات إيجاد القاسم المشترك الأعظم GCD لعددين صحيحين N1 , N2 ، وهو أكبر عدد صحيح يقبل كل من العددين N1 , N2 القسمة عليه بدون باق (مثلا القاسم المشترك الأعظم للعددين ٢٤ ، ٤٠ هو ٨).

(أ) اكتب برنامجا فرعيا داليا لإيجاد القاسم المشترك الأعظم لعددين صحيحين N1 , N2.

(ب) اكتب برنامجا رئيسيا يقرأ قيمتي عددين صحيحين M , N ، ثم يستدعي البرنامج الفرعي لحساب القاسم المشترك الأعظم MAX للعددين M , N ثم يحسب الدالة

$$G = \sin(2.MAX) + \sqrt{MAX + 5} + e^{MAX}$$

٤- اكتب برنامجا لقراءة قيمة عدد صحيح n وقيمتي متغيرين a , x ثم حساب المفكوك التالي للدالة a^x وذلك بكفاءة عالية أي بحيث يستغرق تنفيذ البرنامج أقل وقت ممكن وبحيث يشغل أقل حيز ممكن من ذاكرة الحاسب.

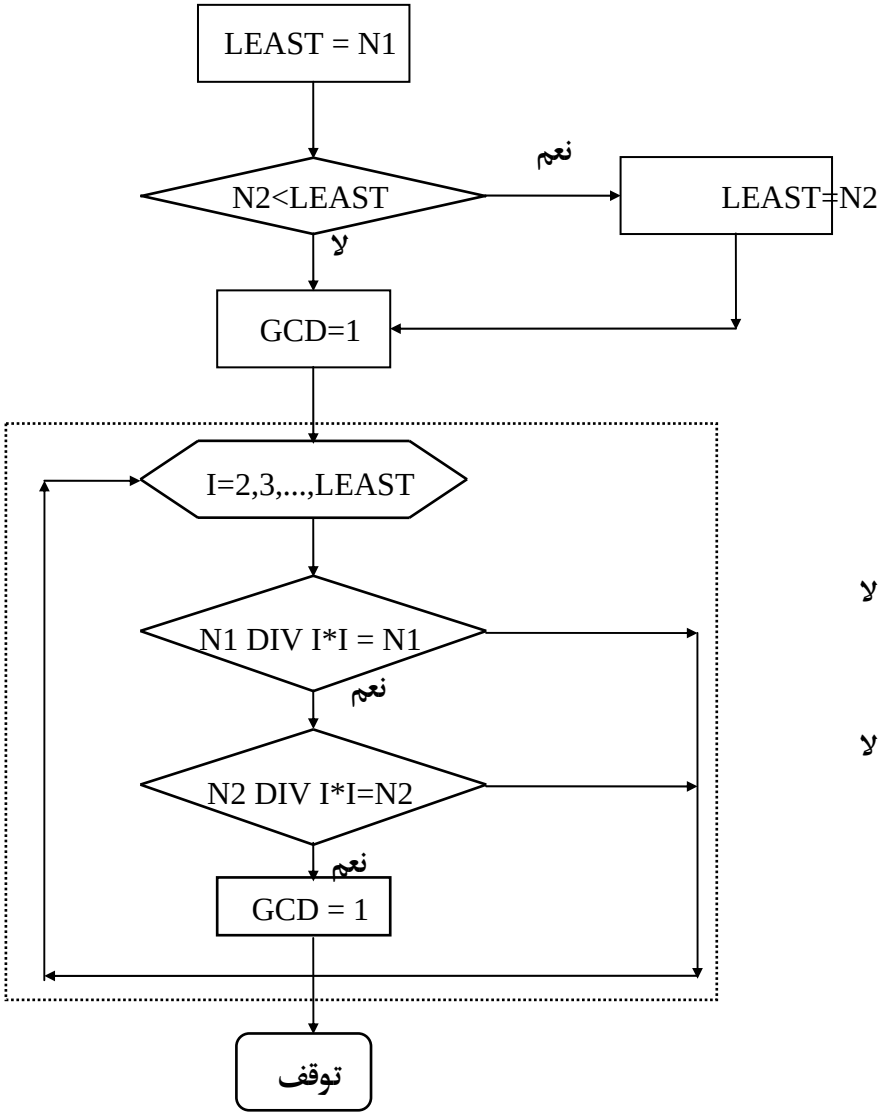
$$a^x = 1 + \frac{x \log a}{1!} + \frac{(x \log a)^2}{2!} + \cdots + \frac{(x \log a)^n}{n!}$$

ثم قارن الجواب الناتج من هذا المفكوك مع الجواب الذي نحصل عليه مباشرة باستخدام الدالة a^x (أي احسب القيمة المطلقة للفارق بين الجوابين).

٥- تستخدم الطرق التكرارية iterative techniques لحل مجموعة المعادلات الخطية. فمثلا بالنسبة للمجموعة التالية والمكونة من معادلتين فقط :

$$\left.\begin{aligned} 2x + y &= 3 \\ x - 3y &= 2 \end{aligned}\right\} \quad (*)$$

يمكن حلها باتباع الخطوات التالية :



(أ) أوجد x (**بدلالة** y) **من المعادلة الأولى ، وأوجد** y (**بدلالة** x) **من المعادلة الثانية ، هكذا :**

$$x = (3 - y) /2$$
$$y = (x - 2) /3$$

(ب) ابدأ بقيمة ابتدائية تقريبية للحل مثل :

XOLD = 1
YOLD = 1

(ج) احصل على قيمة أدق من هذه القيمة التقريبية ، بحساب XNEW , YNEW **كما يلي :**

$$\left.\begin{aligned} XNEW &= (3 - YOLD)/2 \\ YNEW &= (XOLD - 2)/3 \end{aligned}\right\} \quad (**)$$

(د) كرر هذه الطريقة بجعل القيمة التقريبية الجديدة تصبح قيمة تقريبية قديمة

$$XOLD = XNEW$$
$$YOLD = YNEW$$

وحساب قيمة جديدة لكل من XNEW , YNEW **بدلالة** XOLD , YOLD **وذلك باستخدام العلاقتين (**).**

(هـ) استمر في تكرار الخطوة السابقة (رقم د) إلى أن تتحقق – تقريبا – المعادلتان الأصليتان (*) حين تعويض XNEW , YNEW **فيهما (والتقريب هنا يعتمد على الدقة المطلوبة) ، أي إلى أن يتحقق لدينا :**

$$2\ XNEW + YNEW \equiv 3$$
$$XNEW - 3\ YNEW \equiv 2$$

وبصورة أخرى : إلى أن يتحقق الشرطان :

$$|2\ XNEW + \ YNEW - 3| < \varepsilon$$
$$|\ XNEW - 3\ YNEW - 2| < \varepsilon$$

حيث ε **درجة الدقة المطلوبة (مثلا** $\varepsilon = 10^{-4}$**).**

اكتب برنامجا يقرأ قيم الثوابت a, b, c, d, e, f, ε **ويستخدم الطريقة المشروحة في هذا السؤال لحل المعادلتين الخطيتين التاليتين :**

$$a\ x + b\ y = c$$
$$d\ x + e\ y = f$$

٦– أفتى العلماء بأنه لا يجوز شرعا في التعاقد العام – كما في قطاعات التعليم والصحة مثلا – استخدام أجيرين بأجرين مختلفين مع تساويهما في الكفاءة والمؤهلات والجهد والخدمات ، بل يجب تحقيق العدل المطلق الكامل بينهما تحقيقا للمصلحة العامة ، أما المخالفة بينهما لاعتبارات شخصية أو عائلية أو قومية أو جنسية فإنها توقظ الأحقاد وتمزق الأمة ، والله سبحانه وتعالى يقول : {وأُمِرْتُ لأعدل بينكم} ، وهو سبحانه خلق الناس شعوبا وقبائل ليتعارفوا لا ليتخذوا ذلك أساسا للتمييز العنصري.

نفرض أنه قد أُعِدَّ سجل (بطاقة) لكل مدرس (أو مُدَرِّسة) في مرحلة تعليمية معينة في وزارة التربية ، حيث وضع على السجل :

رقم تعريفي ID للمدرس

وحرف يشير إلى جنسه (M للذكر وF للأنثى)

وراتبه الشهري S بالدينار.

وأضيف في النهاية سجل يحمل قيمة سالبة للمتغير S ليشير إلى انتهاء سجلات البيانات.

اكتب برنامجا بلغة الباسكال يقرأ هذه المجموعة من السجلات ويوجد :

(أ) عدد المدرسين والمدرسات (N1) الذين يحصلون على راتب شهري أقل من ٢٥٠ دينارا.

(ب) عدد المدرسين والمدرسات (N2) الذين يحصلون على راتب شهري أعلى من ٤٠٠ دينارا.

(ج) العدد الإجمالي للمدرسين والمدرسات (N)

(د) النسبة المئوية لكل من المدرسين والمدرسات من العدد الإجمالي.

٧- نفرض أن A منظومة فيها أربعون قيمة حقيقية. المطلوب كتابة برنامج بلغة الباسكال :

(أ) يقرأ قيم عناصر المنظومة A.

(ب) يوجد أقرب قيمة – في المنظومة A – للقيمة المتوسطة AV (التي تساوي مجموع كل القيم مقسوما على عددها).

٨- اكتب برنامجا لإيجاد جميع الأزواج المرتبة (i , j) التي تحقق آنيا المتباينات التالية ، حيث كل من i , j عدد صحيح :

$$2i - j < 3$$
$$i + 3j \geq 1$$
$$-6 \leq i \leq 6$$
$$-10 \leq j \leq 10$$

٩- إذا عُلِّقت كتلة من زنبرك فإنها تتذبذب بتردد يُعْطَى بالعلاقة :

$$F = 0.1592\sqrt{K / M}$$

حيث :

F : التردد

K : ثابت الزنبرك

M : الكتلة

والمطلوب كتابة برنامج يحسب ويطبع قيم التردد F وذلك لقيم K المختلفة: ١ ، ٣ ، ٥ ، ... ، ١١. وعند كل قيمة من قيم K فإن M تأخذ القيم: ١, ، ٢, ، ٣, ، ... ، ١. وبحيث يطبع النتائج في شكل جدول من ثلاثة أعمدة تعطي قيم K, M, F ، وبحيث يطبع F صحيحا لثلاثة أرقام عشرية فقط.

١٠- حث الإسلام على الزواج المبكر طلبا للإحصان والعفاف ، ولكن بعض عادات المجتمع وتقاليده تحول دون تحقيق ذلك كعادات التغالي في المهور ، والتباهي بالأثاث والدور والقصور ، والتكالب على المظاهر الجوفاء والمتاع الدنيوي الزائل ، وحب المال حبا جما ، والعصبية القبلية التي تمنع الزواج إلا من عائلات معينة ، وضرورة الحصول على شهادات تعليمية ودراسات عليا أولا ، ... الخ. نفرض أن لدينا مجموعة من سجلات البيانات في ملف حيث يشتمل كل سجل على البيانات التالية :

اسم الشخص (NAME).

حرف يشير إلى الجنس (M لذكر وF للأنثى)

عمر الشخص (AGE).

رقم يشير إلى حالته الزوجية (١ للأعزب و ٢ للمتزوج)

وعدد هذه السجلات غير معلوم ، وقد استخدم الرقم ٩ في الموضع المخصص للحالة الزوجية ليشير إلى نهاية الملف.

ارسم خريطة سير عمليات ، واكتب برنامجا بلغة الباسكال لحساب وطباعة ما يلي :

(أ) النسبة المئوية للذكور (PM) ، والنسبة المئوية للاناث (PF).

(ب) العدد الإجمالي للذكور فوق سن السادسة عشرة (N). والنسبة المئوية – من هذا العدد N – للذكور الأيامى [*] فوق سن الخامسة والعشرين (PUM).

(ج) العدد الإجمالي للنساء فوق سن السادسة عشرة (M) ، والنسبة المئوية – من هذا العدد M – للنساء الأيامى فوق سن الثانية والعشرين (PUF).

١١- نفرض أن كلا من X , Y منظومة مكونة من عشرين عنصرا. اكتب برنامجا

(أ) يقرأ قيمة عدد صحيح N.

(ب) يقرأ قيم أول N عنصر من كل من المنظومتين X , Y.

(ج) يحسب قيم N عنصر من منظومة Z تبعا للقاعدة التالية :

قيمة العنصر z_i تساوي ١ أو صفر أو –١ إذا كانت قيمة العنصر x_i أكبر من أو تساوي أو أصغر من قيمة العنصر y_i على الترتيب.

(د) يطبع جدولا من ثلاثة أعمدة يعطي القيم المتقابلة لعناصر المنظومات X , Y , Z.

(ه) يطبع عدد عناصر المنظومة X التي تزيد قيمة كل عنصر منها عن قيمة العنصر المقابل في المنظومة Y. وكذلك عدد عناصر المنظومة X التي تقل قيمة كل عنصر منها عن قيمة العنصر المقابل في المنظومة Y.

١٢- من المعلوم أن مجموع مربعات الأعداد الصحيحة من ١ إلى N يساوي

$$1^2 + 2^2 + 3^2 + \cdots + N^2 = \frac{N(N+1)(2N+1)}{6}$$

اكتب برنامجا :

(أ) يعرف دالة SUMSQ (N) تعطي قيمة هذا المجموع.

(ب) يقرأ قيمة عدد صحيح LIMIT.

---

[*] الأيامى : جمع أَيِّم ، وهو الرجل الذي لا زوجة له ، أو المرأة التي لا زوج لها.

(ج) يستخدم الدالة المعرفة SUMSQ في حساب مجموع مربعات الأعداد الصحيحة من ١ إلى M وذلك لكل قيمة من قيم M الصحيحة ابتداء من ٢ وحتى قيمة LIMIT.

١٣- تتكون المنظومة A من عدد – لا يزيد عن مائة – من الأعداد الصحيحة غير الصفرية ، وبحيث أن كل عدد منها مكتوب على سطر مستقل.
ويستخدم العدد صفر كقيمة تشير إلى انتهاء عناصر المنظومة.
اكتب برنامجا لقراءة عناصر هذه المنظومة وإيجاد :

(أ) عدد العناصر N في المنظومة.

(ب) عدد العناصر الزوجية NE ، وعدد العناصر الفردية NO في المنظومة.

(ج) عدد العناصر مضاعفات العدد ٣ (أي العناصر التي يقبل كل منها القسمة على ٣ بدون باق) ومجموع قيمها.

١٤- (أ) اكتب برنامجا فرعيا لإيجاد عناصر المتجه $\underline{y}$ الذي يعطى بالعلاقة

$$\underline{y} = A.\underline{x}$$

حيث :

A : مصفوفة n x n من عناصر حقيقية.

$\underline{x}$: متجه مكون من n عنصر حقيقي.

(ب)اكتب برنامجا فرعيا لإيجاد عناصر المتجه $\underline{z}$ الذي يعطى بالعلاقة

$$\underline{z} = \underline{x} - \underline{y}$$

حيث كل من $\underline{y}$ و $\underline{x}$ متجه مكون من n عنصر حقيقي.

(ج)اكتب برنامجا فرعيا داليا لحساب طول متجه $\underline{x}$ مكون من n عنصر حقيقي حيث يعرف الطول بالعلاقة :

$$L = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

(د)اكتب برنامجا رئيسيا يقرأ عناصر كل من :

متجه $\underline{x}$ من خمسة عناصر (كلها على سطر واحد)

متجه $\underline{b}$ من خمسة عناصر (كل عنصر على سطر مستقل)

مصفوفة A من خمسة صفوف وخمسة أعمدة (كل صف على سطر مستقل)

ثم يستخدم البرامج الفرعية السابقة لإيجاد :

(i) عناصر المتجه $\underline{r}$ حيث $\underline{r} = \underline{b} - A.\underline{x}$

(ii) طول المتجه r .

١٥- (أ) اكتب برنامجا فرعيا يحذف كل العناصر الصفرية من منظومة X تحتوي على N عدد حقيقي ، ويحتفظ بالعناصر الأخرى من X في منظومة اسمها Y.

(ب) اكتب برنامجا رئيسيا :

١- يقرأ عناصر منظومة A فيها L عدد حقيقي.

ويقرأ عناصر منظومة Z فيها M عدد حقيقي.

٢- يستخدم البرنامج الفرعي - في (أ) - ليحذف العناصر الصفرية في المنظومة A ويخزن العناصر الأخرى في منظومة ANEW ، وكذلك ليحذف العناصر الصفرية في المنظومة Z ويخزن العناصر الأخرى في منظومة ZNEW.

٣- يطبع فقط المنظومتين ZNEW , ANEW مستخدما عناوين مناسبة.

١٦- منحنى المعادلة $x^2 + y^2 = 50$ عبارة عن دائرة مركزها نقطة الأصل ونصف قطرها يساوي $\sqrt{50}$. اكتب برنامجا بلغة الباسكال 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 $\sqrt{50}$.

١٧- اكتب برنامجا بلغة الباسكال :

(أ) يقرأ بيانات جدول (كالمبين فيما يلي) يعطي قراءات لكميات سقوط المطر على أربع مناطق (الشمال والشرق والجنوب والغرب)

خلال مدة اثني عشر شهرا ، ويخزن هذه البيانات في مصفوفة PRECIP من ١٢ صف و ٤ أعمدة ، بحيث أن العنصر في الصف رقم R والعمود رقم C يعطي كمية المطر التي سقطت خلال الشهر رقم R على المنطقة رقم C (والمناطق مرقمة حسب ترتيبها المذكور سابقا ، وتبدأ بمنطقة الشمال).

(ب) يحسب متوسط كمية سقوط المطر في كل شهر.

(ج) يحسب كمية المطر الكلية التي سقطت على كل منطقة خلال الاثني عشر شهرا ، وبالتالي :

(د) يوجد رقم المنطقة التي سقطت عليها أكبر كمية كلية من الأمطار.

اطبع النتائج بحيث تظهر بالصورة المبينة فيما يلي :

أولا : البيانات Input file :

12.4	13.5	12.1	12.6
8.2	9.5	8.0	7.5
14.9	13.5	12.9	15.3
30.0	35.5	40.2	37.7
24.5	26.1	23.4	24.0
10.0	9.0	8.5	7.9
4.0	.5	2.3	3.2
3.4	0.0	4.5	6.1
7.8	7.9	7.6	7.8
10.1	10.6	10.9	12.0
12.3	12.5	12.1	12.6
14.1	15.2	12.8	9.5

ثانيا : النتائج Output file :

MONTH	NORTH	EAST	SOUTH	WEST
١	12.40000	13.50000	12.10000	12.60000
٢	8.200000	9.500000	8.000000	7.500000
٣	14.90000	13.50000	12.90000	15.30000
٤	30.00000	35.50000	40.20000	37.70000
٥	24.50000	26.10000	23.40000	24.00000
٦	10.000000	9.00000	8.500000	7.900000
٧	4.000000	0.5000000	2.300000	3.200000

٨	3.400000	0.0000000E+00	4.500000	6.100000
٩	7.800000	7.900000	7.600000	7.800000
١٠	10.10000	10.60000	10.90000	12.00000
١١	12.30000	12.50000	12.10000	12.60000
١٢	14.10000	15.20000	12.80000	9.500000

```
 12.65000   IS    1   THE AVERAGE FOR MONTH
 8.300000   IS    2   THE AVERAGE FOR MONTH
 14.15000   IS    3   THE AVERAGE FOR MONTH
 35.85000   IS    4   THE AVERAGE FOR MONTH
 24.50000   IS    5   THE AVERAGE FOR MONTH
 8.850000   IS    6   THE AVERAGE FOR MONTH
 2.500000   IS    7   THE AVERAGE FOR MONTH
 3.500000   IS    8   THE AVERAGE FOR MONTH
 7.775001   IS    9   THE AVERAGE FOR MONTH
 10.90000   IS   10   THE AVERAGE FOR MONTH
 12.37500   IS   11   THE AVERAGE FOR MONTH
 12.90000   IS   12   THE AVERAGE FOR MONTH

151.7000    IS    1  THE TOTAL RAINFALL FOR LOCATION
153.8000    IS    2  THE TOTAL RAINFALL FOR LOCATION
155.3000    IS    3  THE TOTAL RAINFALL FOR LOCATION
156.2000    IS    4  THE TOTAL RAINFALL FOR LOCATION
                 4   THE WETTEST LOCATION IS
```

١٨- اكتب برنامجا بلغة الباسكال يقرأ قيمة عدد صحيح N (حيث $N \le 20$) ، وكذلك إحداثيات رؤوس مُضَلَّع (polygon) عدد رؤوسه N ، ويُخَزِّن هذه الإحداثيات في منظومتين X , Y. ثم يحسب ويطبع :

(أ) مساحة المضلع والتي تعطى بالعلاقة :

$$AREA = \frac{1}{2}\left[y_1 x_N - x_1 y_N\right] + \sum_{i=2}^{N}\left(y_i x_{i-1} - x_i y_{i-1}\right)$$

(ب) إحداثيي مركز ثقله (Center of gravity) ، ويعطيان بالعلاقتين :

$$XG = \left(\sum_{i=1}^{N} x_i\right) / N \quad , \qquad YG = \left(\sum_{i=1}^{N} y_i\right) / N$$

١٩- **(أ)** **نفرض أن A مصفوفة مربعة N x N من أعداد حقيقية ، وأن K عدد صحيح موجب. اكتب برنامجا فرعيا داليا يحسب قيمة الدالة LARGE للمصفوفة A باستخدام العلاقة :**

$$\mathrm{LARGE}(A, N, K) = \left(\sum_{i,j=1}^{N} \left| a_{ij} \right|^{K} \right)^{\frac{1}{k}}$$

(ب) **نفرض أن X منظومة مكونة من n عنصرا (من الأعداد الحقيقية). اكتب برنامجا فرعيا إجرائيا يحسب قيمة كل من الدالتين التاليتين :**

$$\alpha = |x_1| + |x_2| + \ldots + |x_n|$$
$$\beta = \mathrm{Max}\,(|x_1|, |x_2|, \ldots, |x_n|)$$

(ج) **اكتب برنامجا رئيسيا يقرأ قيم عناصر منظومة A مكونة من خمسين عددا حقيقيا ، وقيم عناصر مصفوفة B مكونة من ثمانية صفوف وثمانية أعمدة من أعداد حقيقية ، ثم يحسب باستخدام البرنامجين الفرعيين السابقين :**

(١) قيمة كل من الدالتين α , β للمنظومة A.

(٢) قيمة الدالة LARGE للمصفوفة B ، وذلك للقيم K=1,3,5,7,9.

اختبارات عامة

فيما يلي نماذج لمجموعة من الاختبارات العامة الفصلية (Mid_Term Tests) والنهائية (Final Exams.) ، حيث يشتمل أي نموذج للاختبار الفصلي الأول على أسئلة في الموضوعات من أساسيات البرمجة إلى عبارات التكرار ، بينما يشتمل الاختبار الفصلي الثاني على أسئلة في عبارات التكرار والمنظومات والبرامج الفرعية. أما أسئلة الاختبار النهائي فتشمل كافة محتويات منهج لغة الباسكال بالكتاب. وقد جمعت هذه النماذج من الاختبارات العامة الموحدة في لغة الباسكال بقسم الرياضيات وعلم الحاسوب بجامعة الكويت. وتوجد بنهاية الكتاب حلول نموذجية كاملة لهذه الاختبارات.

أولا : نماذج للاختبار الفصلي الأول

الاختبار الفصلي الأول
رقم (١)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (٣٠ درجة) :

أ) أوجد قيمة كل من التعابير التالية المكتوبة بلغة الباسكال ، وحدد نوع كل منها :

a) 2 + 5/2 * 3

b) 2 * 3 + 6 mod 2 div 2

c) trunc (2 * 0.43E1) + sqr (round 1.6))

i) (('A' < 'Z') or not X) and ('5' < '8')

ب) ترجم الصيغ الرياضية التالية إلى تعابير بلغة الباسكال ، حيث x , y , z متغيرات.

i) $2.0 \cdot 10^{-1} \sqrt{\frac{x-y}{x+y}}$

ii) $\frac{x}{y} + z^3$

iii) $(x - \sqrt{\frac{x}{y+z}})^2$

ج) اكتب تعابير منطقية مقابلة للشروط التالية :

i) **قيمة متغير صحيح m تقع في المدى من 10- إلى 10+ احتوائيا.**

ii) **قيمة المتغير الصحيح m هي إحدى مضاعفات قيمة المتغير الصحيح n.**

iii) **القيمة المطلقة للفارق بين قيمتي متغيرين حقيقيين x , y أقل من $.10^{-3}$**

السؤال الثاني (٢٥ درجة) :

اكتب بالضبط مخرجات كل من قطعتي البرنامج التاليتين بعد تتبع تنفيذ كل منهما. مَثِّل الفراغ بالرمز ∇.

أ)

```
var x , y : real ;
begin
    x := 2.0 ;
    if x >= 0.0 then
        x := x + 1.0
    else
        writeln ('∇x∇ = ∇' , x : 6 :2)
    (*end if *) ;
    y := 2.0 * x - 1.0 ;
    if x < y then
        if round (y) mod round (x) = 1 then
            writeln ('∇y∇ = ∇' , y : 6 :2)
        else writeln ('∇y-x∇ = ∇' , y-x : 6 :2)
            (*end if*)
    (*end if*) ;
end.
```

ب) نفرض أن مدخلات البرنامج هي : ↵ b

↵ 35.4

```
var x : char ;
    t : real ;
begin
    readln (x) ;
    readln (t) ;
    if ('a' <= x) and (x <= 'c') then
        case x of
'a' : write ('rainy weather ∇') ;
'b' : write ('partly cloudy ∇') ;
'c' : write ('sunshine ∇') ;
        end
        (*end case *)
    else
        writeln ('error : no forecast')
    (*end if*)
writeln ('temperature is ' , t : 5 : 1 , '∇C') ;
end.
```

السؤال الثالث (٢٥ درجة) :

اكتب برنامجا يحسب إما حجم كرة V نصف قطرها R (R مدخل للبرنامج) او مساحة سطحها S ، ثم يطبع النتيجة. ويتم اختيار حساب الحجم V أو حساب مساحة السطح S إذا كانت قيمة مدخل آخر هي الرمز 'V' او الرمز 'S' على الترتيب. وينبغي أن يكتشف البرنامج أي رمز اختيار خاطئ مدخل للبرنامج.

ملاحظة : حجم الكرة : $V = \frac{4}{3}\pi R^3$

مساحة سطح الكرة : $S = 4\pi R^2$

السؤال الرابع (٢٠ درجة) :

اكتب برنامجا يحسب ويطبع مجموع جميع الأعداد الصحيحة في المدى من m إلى n احتوائيا (حيث $m \le n$). العددان الصحيحان m , n مدخلان للبرنامج.

ملاحظة : المجموع المطلوب : $S = \sum_{i=m}^{n} i$

الاختبار الفصلي الأول

رقم (٢)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة وربع

السؤال الأول (٢٠ درجة) :

نفرض أن لدينا الإعلانات والتعريفات التالية بلغة الباسكال:

```
const
    B = True ;          R = 2.71728 ;
    C = ' ? ' ;             I = 6 ;
var
    B1 :  Boolean ; I1 , I2 : Integer ;
    C1,C2 :  Char ;   R1 , R2:  Real ;
```

اذكر صحة أو خطأ كل من العبارات التالية المكتوبة بلغة الباسكال ، وإن كانت العبارة خاطئة فاذكر الأسباب :

```
I1 := I2 + R + 2*I ;(One)
B1 := C1 and C2 ;(Two)
if R1 + I1 > R2(Three)
    then I := 5 ;
B1 : = B and (R1 + R2 > R) ;(Four)
I2 := (R1 mod I) * I2 ;(Five)
while I do (Six)
    R1 := R1 + 1.0 ;
readln (B1 , C1 , I1) ;(Seven)
writeln (C:1 , R1 : 6 : 2 , I : 2) ;   (a)
```

السؤال الثاني (٢٥ درجة) :

اكتب برنامجا بلغة الباسكال يقوم بإجراء العمليات التالية :

- **قراءة قيم ستة أعداد حقيقية** $x_1 , x_2 , x_3 , y_1 , y_2 , y_3$
- **حساب قيم**

$$LX = \sqrt{x_1^2 + x_2^2 + x_3^2} \quad , \quad LY = \sqrt{y_1^2 + y_2^2 + y_3^2}$$

$$XdotY = x_1y_1 + x_2y_2 + x_3y_{31},$$

وحساب قيمة $\cos(\alpha)$ **حيث**

$$) = \begin{cases} 1 & \cdots LX \text{ or } LY \le 0.0001 \quad \textbf{إذا كان} \\ \dfrac{XdotY}{LX \times LY} & \cdots\cdots \quad \textbf{ما عدا ذلك} \end{cases}$$

- **طباعة قيم** LX , LY , XdotY , cos (α)

السؤال الثالث (١٥ درجة) :

اكتب مخرجات البرنامج التالي المكتوب بلغة الباسكال :

```
program Question3 (Input , Output) ;
                                  var
                        I , J , Sum : Integer ;
                              begin {Question3}
                                      I := 7 ;
                                      J := 6 ;
                                    Sum := 0 ;
                             while (I+J >= 2) do
                                      begin
                          if (I mod 2 = 1) then
                                      begin
                                  I := I-1 ;
                                  J := J-1
                                    end {if}
                     else if ((J div 2) >= 2) then
                                      begin
                                  I := I-2 ;
                                  J := J-2
                                  end {else if}
                                      else
                                      begin
                                  I := I-3 ;
                                  J := J-3
                                  end ; {else}
                            Sum := Sum + I + J ;
                Writeln (I : 2 , '--' , J : 3 , ' -- ' , Sum : 4) ;
                                    end ; {while}
                                  end. {Question3}
```

السؤال الرابع (٤٠ درجة) :

اكتب برنامجا بلغة الباسكال يستمر في حث (prompting) **المستخدم** (user) **على إدخال عدد صحيح** ID **يمثل الرقم التعريفي لطالب في أحد الصفوف وكذلك الدرجات الثلاث للطالب في ثلاثة اختبارات ، ويطبع البرنامج الرقم**

التعريفي ID والدرجة المتوسطة في الاختبارات الثلاثة ، وذلك لكل طالب في الصف ، ويتوقف البرنامج حين تكون قيمة ID المدخلة صفرا. كذلك يطبع البرنامج الدرجة المتوسطة للصف في كل اختبار من الاختبارات الثلاثة والدرجة المتوسطة للصف في الاختبارات الثلاثة جميعها.

الاختبار الفصلي الأول

رقم (٣)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة وربع

السؤال الأول (٨ درجات):

حدد نوع كل من الثوابت التالية : صحيح (integer) ، أو حقيقي (real) ، أو سلسلة رموز (string) ، أو خاطئ (invalid):

(a) 1,015
(b) .00045
(c) '12 + 34'
(d) $3.90
(e) 'A"A"A'
(f) -4.5
(g) E12
(h) 14E2.5

السؤال الثاني (١٤ درجة):

نفرض أن لدينا التعريفات التالية لبعض المتغيرات.

```
var
  A,B,C:integer;
  X,Y,Z:real;
```

ما هي القيم التي يتم إسنادها لهذه المتغيرات ، أو ما هي الأخطاء التي تحدث وذلك عند تنفيذ مجموعة العبارات التالية باستخدام بيانات الإدخال المعطاة؟

```
  readln(A);          A =
  read(B,C);          B =      C =
  readln;
  readln(X,Y);        X =      Y =
  read(Z);            Z =
  read(A);            A =
input:  1  2  3
        4  5  6
        7.1  8  9
        4.5  5.5
```

السؤال الثالث (١٢ درجة) :

أوجد قيمة كل من التعابير التالية ، أو وضح لماذا يعد التعبير خاطئا.

(a) 7.0 / 2 + (12 * abs(-3)) / (7 div 2)
(b) 7 * 10 - 5 mod 2 * 4 + round(8.95)

(c) sqrt (sqrt (64) - 2 * 4 / 2 + 5)

السؤال الرابع (١٢ درجة) :

ترجم كلا من التعبيرين الرياضيين التاليين إلى تعبير بلغة الباسكال.

(a) $\left|\frac{a-b}{2}\right| \neq c^{-1}$

(b) $\pi \cdot \sqrt{(x-\alpha)^2+(y-\beta)^2}$

السؤال الخامس (٢٠ درجة) :

نفرض أن لدينا البرنامج التالي بلغة الباسكال. اكتشف أي أخطاء في البرنامج مع بيان سبب الخطأ.

```
program errors (input,output);
const
    pi = 3.1415;
    j := 6;
    no = true;
    star:char;
var
    end,x,y:integer;
    p   : real;
    2q  : real;
    c1,c2:char;
    flag:boolean;
begin
    x:= 4;    p:= 5.0;   c1:= 'A';
    x := x mod p;
    c1:= c1 * 1;
    flag:= no;
    if flag and x + y > 4 then
       writeln (it is true)
    else
       writeln (x:4);
    if (x * y) > 5 then
       no:=false
   else
```

```
        x:=p;
end.
```

السؤال السادس (١٠ درجات) :

نفرض أن A,B,R,S **متغيرات حقيقية. اكتب قطعة برنامج تجعل قيمة** R **صفرا وتضيف** 1 **إلى قيمة** S **وذلك إذا كانت** $A > B$ **، وأما ما عدا ذلك فإنها بالعكس تضيف** 1 **إلى قيمة** R **وتجعل قيمة** S **صفرا.**

السؤال السابع (٢٤ درجة) :

اكتب برنامجا بلغة الباسكال يقرأ عددا صحيحا (grade) **يمثل درجة طالب ، ثم يطبع رسالة مناسبة تبعا لما يلي :**

إذا كانت الدرجة في المدى من 0 **الى** 100 **(أى أن** $0 \leq \text{grade} \leq 100$**) فإن البرنامج يطبع الرسالة** 'valid grade' **، وما عدا ذلك فإنه يطبع الرسالة** 'invalid grade' **تتبعها إما الرسالة** 'less than zero' **أو الرسالة** 'greater than 100' **تبعا لقيمة الدرجة.**

الاختبار الفصلي الأول

رقم (٤)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (١٠ درجات):

نفرض أن القيم المخزونة في المتغيرين الصحيحين I , J والمتغير الحقيقي P هي :

$$I = 10 , \quad J = 20 , \quad P = 2.5$$

اكتب قطعة برنامج تعطي المخرجات التالية وهي عبارة عن نص مكون من أربعة سطور. استخدم في القطعة المتغيرات I , J, P لتمثل القيم العددية التي تظهر بالنص.

Zakat of Money is 2.5 percent of savings.
Zakat of Gold is 2.5 percent of the pure amount of gold.
Zakat of Fruit is 10 percent of the amount of fruits.
Zakat of Rikaz (buried treasures, minerals, petroleum) is 20 percent.

السؤال الثاني (١٠ درجات) :

تتبع تنفيذ قطعة البرنامج التالية وأوجد القيم النهائية للمتغيرات الحقيقية i, x, y, z .

```
i := 2 * 6/5 + 3/5 ;
x := ABS(-5.2) + SQRT(ROUND(15.6)) ;
y := 11 MOD (TRUNC(TRUNC(x)/i)) ;
z := SQR (17 DIV 5*2) ;
x := z/6*y ;
```

السؤال الثالث (١٠ درجات) :

ترجم العلاقات التالية إلى عبارات إسناد حسابية بلغة الباسكال :

i) **طاقة الحركة لجسم متحرك :** $KE = \frac{1}{2}mv^2$

ii) **المسافة بين نقطتين :** $D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

iii) **حجم اسطوانة :** $V = \pi r^2 h$

iv) **مساحة مثلث بدلالة أطوال أضلاعه** a , b , c **:**

$s = \frac{a+b+c}{2}$ $Area = \sqrt{s(s-a)(s-b)(s-c)}$;

v) درجة الحرارة بالتقدير المئوي المقابلة للدرجة بالتقدير الفهرنهيتي

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

السؤال الرابع (١٢ درجة):

أ) باستخدام عبارة .. if .. then .. else والدالة MOD اكتب قطعة برنامج تتحقق ما إذا كان عدد صحيح معطى N فرديا أم زوجيا ، وتطبع رسالة مقابلة تفيد ذلك.

ب) ما هي القيمة النهائية للمتغير x في كل من قطعتي البرنامج التاليتين إذا كانت القيمة الابتدائية للمتغير x هي x = 1 ؟

(1)

```
if x >= 0 then
   x := x+1
else if x >= 1 then
   x := x+2 ;
```

(2)

```
if x >= 0 then
   x := x+1 ;
if x >= 1 then
   x := x+2 ;
```

السؤال الخامس (١١ درجة) :

نفرض أن قيم المتغيرات الحقيقية X , Y , Z هي :

X = 3.0 , Y = 4.0 , Z = 2.0

وأن قيمة المتغير المنطقي FLAG (boolean varible) هي FLAG = false .

أوجد قيمة كل من التعابير المنطقية (boolean expressions) التالية :

١) (X > Z) and (Y > Z)
٢) (X + Y / Z) <= 3.5
٣) (Z > X) or (Z > Y)
٤) not FLAG
٥) (X = 1.0) or (X = 3.0)
٦) (0.0 < X) and (X < 3.5)
٧) (X <= Y) and (Y <= Z)
٨) not FLAG or ((Y + Z) >= (X - Z))
٩) not (FLAG or ((Y + Z) >= (X - Z)))

السؤال السادس (١٧ درجة) :

من المفروض أن يقوم البرنامج التالي بإجراء العمليات التالية :

i) قراءة قيمتي مدخرات أحد الأشخاص وصدقاته.

ii) حساب قيمة زكاته.

iii) حساب وطباعة المبلغ الصافي الباقي من مدخراته بعد دفع زكاته وصدقاته.

أعد كتابة البرنامج بعد تحديد وتصحيح أخطائه التركيبية (syntax errors)

```
Brogram Money (input, output)
Variables : Savings ; Charities ; Zakat ; Net : real
Constant : Precent : 0.025 ;
Begen
    Writeln (′Enter amount of Savings) ;
    Readln (Savings) ;
Writeln (′ Enter amount of Charities′) Readln (Charities) ;
    Savings * Percent := Zakat ;
    Net = Savings - Zakat - Charities ;
    Writeln (′Savings = K.D. , S) ;
    Writeln (′Net = K.D. , Net′)
End ;
```

السؤال السابع (١٤ درجة) :

تطفو (float) **مادة على سطح الماء إذا كانت كثافتها** (density) **(الكثافة = الكتلة/الحجم** density = mass/volume**) أقل من** 1 g/cc **(أي جرام واحد لكل سنتيمتر مكعب) بينما تغوص** (sink) **إذا كانت كثافتها** 1 g/cc **أو أكثر.**

اكتب برنامجا يقرأ كتلة جسم m **وحجمه** v **، ويطبع رسالة تفيد ما إذا كان الجسم سيطفو أم يغوص في الماء.**

السؤال الثامن (١٦ درجة) :

اكتب برنامجا يحسب عدد دقات قلب الإنسان HB (Human Heart Beats) **خلال فترة حياته ، بفرض أن :**

- **عدد أيام السنة :** 365.25 DPY :
- **معدل دقات القلب :** 75 **دقة / دقيقة** RATE :

حيث يقرأ البرنامج عمر الإنسان AGE **بالسنوات ، ويحسب ويطبع عدد دقات قلبه** HB **. مثلا : إذا قرأ القيمة** 65 **فإنه يطبع رسالة مثل**

For an age of 65, the number of heart beats is: 2.564055 E + 09.

اجعل كلا من DPY , RATE **ثابتا** (const) **ضمن قسم إعلانات البرنامج.**

الاختبار الفصلي الأول

رقم (٥)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (٦ درجات):

ما هي قيمة (value) ونوع (type) كل من التعابير التالية ؟

a) Round (9.8) div trunc (3.85) / 2 ;

b) (0 < 1) or (5 <= 12) and (sqr (4) < 10) ;

c) 43 mod 7 * 14 div 2 - abs (-43) ;

السؤال الثاني (٧ درجات):

نفرض أن لدينا الإعلانات التالية :

```
const
   Pi = 3.14159 ;
var
   X , Y  :  real ;
   A, B, I : integer ;
```

ونفرض أن

Y = 1.0 B = 4 , A = 3 ,

لكل عبارة من العبارات التالية اذكر قيمة I أو X إن كانت العبارة صحيحة ، أو اذكر سبب الخطأ إن كانت العبارة خاطئة.

i) I := A mod B ; ii) I := A mod Y ;

iii) I := A / B ; iv) X := A / Y ;

v) X := Pi div A ; vi) X := A div B ;

vii) I := A mod 0 ;

السؤال الثالث (١٢ درجة):

ما هي القيمة النهائية للمتغير الصحيح x بعد تنفيذ كل من قطعتي البرنامج التاليتين المنفصلتين ، إذا كانت القيمة الابتدائية للمتغير x هي :

x = 9 (c) x = 1 (b) x = 0 (a)

(i)
```
if x >= 1 then
   x := x + 1
else if x >= 2 then
   x := x + 12
else
   x := 12;
```

(ii)
```
if x >= 1 then
   x := x + 1 ;
if x >= 2 then
   x := x + 12 ;
if x >= 12 then
   x := 12 ;
```

السؤال الرابع (٨ درجات):

اكتب بالضبط مخرجات البرنامج التالي المقابلة للمدخلات المعطاة .

استخدم الرمز Δ للدلالة على وجود فراغ.

```
Program Exchange (input, output) ;
                                    var
                                A , B : integer ;
                                  : real ;   C
                                begin
  writeln (′Please type in 2 integer numbers and a real number′)
                                readln (A , B , C) ;
  writeln (′Backwards they are′ , C : 8 : 3 , B : 5 , A : 5) ;
    writeln (′In order they are′ , A : 1 , B : 4 , C : 6 : 1)
                                end.
                    55.38   34   -12      input :
```

السؤال الخامس (٢٢ درجة):

صحح الأخطاء التركيبية (syntax errors) **في البرنامج التالي (اكتب العبارات الخاطئة بعد تصحيح أخطائها) ، ثم اكتب مخرجات البرنامج المقابلة للمدخلات المعطاة .**

```
Program JihadPayRoll (input ; output).
                                    var
                                Days = integer,
WageRate,RegPay,OverTime,GrossPay,Donations,NetPay
                                =real,
                                begin
  writeln (′enter number of days of Jihad for the sake of Allah′) ;
                                readln (Days) ;
                  writeln (′enter wage rate per day′) ;
                                readln (WageRate) ;
  writeln (′enter amount of donations for the sake of Allah′) ;
                                readln (Donations) ;

                                if Days ≤ 40 then
                        GrossPay = Days * WageRate ;
                                        else
                                        begin
                        RegPay = 40 * WageRate,
      Over Time = (Days - 40) * WageRate * 1.5 ,
```

```
                    GrossPay = RegPay + Over Time ,
writeln(RegPay=′,RegPay,Over  Time  =′  Over
                                                    Time)
                                          end ;
                              NetPay = GrossPay - Donations ;
          writeln (′GrossPay =′ , GrossPay , ′NetPay =′ , NetPay)
                                                      end ;
   30 input    (ii)                    100 input    (i)
   20                                   20
  300                                   600
```

السؤال السادس (١٠ درجات):

اكتب برنامجا يقرأ طولي ضلعي (two sides) **مثلث قائم الزاوية** A , B ، **ثم يحسب ويطبع طول الوتر** H (hypotenuse) ، **حيث** $A^2 + B^2 = H^2$.

السؤال السابع (١٥ درجة):

اكتب برنامجا يقرأ ارتفاع مخروط دائري قائم h (right circular cone) **ونصف قطر قاعدته** r ، **ثم يحسب ويطبع حجمه** (volume = $1/3\ \pi\ r^2\ h$) ، **على أن يعطي البرنامج رسالة تنبيه إلىخطأ** (error message) **في حالة إدخال أي قيمة سالبة ، كما يقوم البرنامج بتقريب النتيجة المطلوبة إلى أقرب عدد صحيح.**

السؤال الثامن (٢٠ درجة):

اكتب برنامجا يقرأ عددين صحيحين N1 , N2 **ورمزا** OP (character) **يشير إلى عملية حسابية معينة ، حيث هذا الرمز** OP **هو أحد الرموز الأربعة** {+ , - , * , /} **، ثم يقوم البرنامج بإجراء العملية الحسابية المقابلة لهذا الرمز ، ويطبع المخرجات في صورة مماثلة لما يلي ، مع التأكد أولا (في حالة القسمة) من عدم القسمة على صفر.**

```
65 + 20 = 85
65 - 20 = 45
65 * 20 = 1300
65 / 20 = 3.25
```

الاختبار الفصلي الأول

رقم (٦)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (٥درجات):

ترجم كلا من التعبيرين الرياضيين التاليين إلى لغة الباسكال.

a) $\frac{K(R-J)+H}{1+\frac{9}{|K+J|}}$ b) $\frac{-B\pm\sqrt{B^2-4AC}}{2A}$

السؤال الثاني (١٠ درجات):

(i) **(٥ درجات) أي الأسماء التعريفية** (identifiers) **التالية مقبولة (صحيحة** Valid**) وأيها غير مقبولة** (Invalid) **كأسماء متغيرات (مع بيان السبب إن كانت غير مقبولة) ؟**

Identifier	Valid	Invalid	Reason **السبب**
Minimum			
I/O			
13B			
Else			
Constant			
student's			
output			
code-name			
var			
number2			

(ii) **(٥ درجات) نفرض أن لدينا الإعلانات والتعريفات التالية :**

```
Const
    h = 5 ;
Var
    A , B : real ;
    J , K , L : integer ;
    grade , let : char ;
    flag , ok : boolean ;
```

أي عبارات الباسكال التالية صحيحة وأيها خاطئة (مع بيان السبب إن كانت خاطئة) ؟

Statement	Valid	Invalid	Reasonالسبب
J := A mod 2;			
h := h + 1 ;			
flag := (5 <> 4 + 1) ;			
ok := 'true' ;			
grade := B + ;			

السؤال الثالث (١٥ درجة):

(i) **(١٠ درجات) نفرض أن**

A = 2.4 , B = 8.6 , C = 4 , D = 3

أوجد قيمة ونوع كل من التعابير الحسابية التالية :

١. Sqrt(c) * (D mod C)
٢. Round(B) div (Trunc(A) + 6)
٣. C/D * (A + B) + Sqr(D)
٤. (C div D) / Abs (2 - Trunc(B))
٥. D * Trunc (2 * B - D) div C

(ii) **(٥ درجات) نفرض أن** a = 4 , b = 3

ما هي قيمة كل من التعابير المنطقية التالية ؟

١. (B > A) and (B = 3) or ((A - B) <> 0) and not (B = 0)
٢. (A mod 2 <> 0) or (((B + 3) div A) > 5)
٣. (A + 5) >= sqr(B)

السؤال الرابع (١٥ درجة):

(i) **(٥ درجات) ما هي مخرجات قطعة البرنامج التالية :**

```
A := 1 ;
B := 2 ;
C := 3 ;
if A > B then
    if B > C then
        if A > C then
            writeln(A)
        else
            writeln(B)
    else
        writeln(C) ;
writeln(A + B + C) ;
```

(ii) (١٠ درجات) ما هي مخرجات البرنامج التالي ؟

```
Program Test (input , output) ;
var
   A , B , C , D : integer ;
   flag : boolean ;
begin
   read(A) ;                          input data : المدخلات
   readln(B , C) ;                    24   72
   read(D) ;                          12   33
   flag := (D >= A + B) ;             9    2
   Writeln(A , B) ;
   Writeln(C , D , flag) ;
end.
```

السؤال الخامس (١٠ درجات):

اكتب برنامجا يطبع السطور التالية :

Fear Allah wherever you are,
follow up an evil deed with a good one
and it will wipe it out,
and behave well towards people.

السؤال السادس (١٠ درجات):

اكتب برنامجا يقرأ قيم ثلاثة أعداد حقيقية ، ويطبع منها القيم التي تزيد عن القيمة المتوسطة لهذه الأعداد الثلاثة.

السؤال السابع (١٥ درجة):

اكتب برنامجا يقرأ قيمة عدد صحيح NUM **ذي أربعة أرقام** (4 digits) ، **ثم يحسب البرنامج ويطبع قيمة مجموع هذه الأرقام الأربعة. مثلا إذا كان** NUM = 4723 **فإن البرنامج يطبع القيمة** 16 ، **وذلك لأن** 4 + 7 + 2 + 3 = 16 . **ويتحقق البرنامج أولا من أن العدد** NUM **يحتوى على أربعة أرقام. (إرشاد : استخدم المؤثرين** div , mod)

السؤال الثامن (٢٠ درجة):

تعطي إحدى الشركات تخفيضا في قيمة المبيعات يعتمد على هذه القيمة كما يلي : إذا كانت قيمة المبيعات (purchase amount) **تزيد عن أو تساوي ١٠٠٠ دينار فإن معدل التخفيض** (discount rate) **يساوي ٢٠٪ ، وإذا كانت قيمة المبيعات أقل من ١٠٠٠ دينار وأكبر من أو تساوي ٥٠٠ دينار فإن معدل التخفيض يساوي ١٥٪ ، وإذا كانت قيمة المبيعات أقل من ٥٠٠ دينار وأكبر من أو تساوي ٢٥٠ دينارا فإن معدل التخفيض يساوي ١٠٪ ، بينما إذا قلت قيمة المبيعات عن ٢٥٠ دينار فلا يعطى أي تخفيض.**

اكتب برنامجا يقرأ قيمة المبيعات ، ويحسب قيمة التخفيض ، ويطبع فاتورة العميل (coustomer bill) **في الصورة التالية :**

Purchase Amount	:	KD
Discount Rate	:	%
Discount Amount	:	KD
Net Purchase Amount	:	KD

الاختبار الفصلي الأول

رقم (٧)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (١٠ درجات) :

نفرض أن A , B , C , X **متغيرات حقيقية ، وأن** I , J , M **متغيرات صحيحة. ونفرض أن**

A = 4.0 , B = 2.5 , C = 1.5 , I = 3 , J = 2

أوجد قيمة كل من X , M **في العبارات المستقلة التالية :**

1) M:=J - I * J ; M = ———

2) X:= A+C / I+B; X = ———

3) M:= I*J DIV (Round(I/J)); M = ———

4) X := SQR(SQR(A)); X = ———

5) M:= 2* I+J MOD I M = ———

السؤال الثاني (١٢ درجة) :

استخدم القيم المعطاة فيما يلي لتحديد نتائج التعابير المنطقية التالية :

A = 6.0 , B = 1.5 , C = 5.0 , D = 3.5 , I = -3 , Done = false

التعبير	النتيجة
1) not(A<3.0*B)	——
2) -I<=I+6	——
3) (Abs(I)>3) or Done	——
4) (A/B*C>D) or (B+C<A)	——
5) (A/B*C>D) and (B+C<A)	——
6) (A/B*C<D) or (B+C<A)	——

السؤال الثالث (٢٠ درجة) :

أوجد وصحح الأخطاء في العبارات المستقلة التالية المكتوبة بلغة الباسكال. وإن كانت العبارة صحيحة فاكتب ذلك (اكتب .O.K).

العبارة	التصحيح إن وجد
1) K12.3:= I * J;	
2) R:= SQRT(YZ + 4E3 + 12.5);	
3) S:= 3E4.0 + SQR(T+S) + 25Y/-7.0;	
4) SQRT(49.0):= 7.0;	
IF SQR(X) <= upper bound THEN (	
Y:=X;	
6) J := J + 1;	
7) X:= SQRT(SQRT(X));	
8) X1:= JI + K – 4;	
9) R:=X3*(RL + RW);	
10) READLN (A,k,M,T).	
11) IF I:=K OR I+J>8 THEN N=10;	
12) IF A OR B <82.0 THEN Z:=Y;	

السؤال الرابع (١٢ درجة) :

(أ) (٦ درجات) اكتب عبارات بلغة الباسكال تقوم بتنفيذ الأوامر التالية :

١) إذا كانت قيمة Den **أقل من** 0.005 **فاطبع الرسالة**

DENOMINATOR IS TOO SMALL

٢) إذا كانت القيمة المطلقة للمتغير x **أكبر من أو تساوي ٣ ، فاجعل** Time **يساوي صفرا وزد قيمة** Count **بواحد.**

٣) إذا كانت Distance **أقل من** 50.0 **و** Time **أكبر من** 10.0 **فزد قيمة** Time **بـ** 0.05 **، وإلا فزد قيمة** Time **بـ** 2.0.

(ب) (٦ درجات) اكتب مخرجات قطعة البرنامج التالية :

```
program test (input,output);                    OUTPUT
var
  a,b,c,d: integer;
begin
  a:= 15;  b:= 12;  c:= 4;  d:= 20;
  if (a < b) or (c <> d) then
     if a mod 2 = 0 then
      if c mod 2 = 0 then
         writeln(c,' is even')
     else if a < d then
           writeln(d div 4)
```

```
  else
     writeln('d = ',d);
     writeln('c = ',c);
end.
```

السؤال الخامس (١٠ درجات) :

اكتب برنامجا يقرأ قيمة عدد حقيقي ويطبع :

i) **مربعه** ii) **جذره التربيعي** iii) **قيمته المطلقة**

iv) **العدد مقربا لأقرب عدد صحيح** v) **جزءه الصحيح** vi) **مكعبه**

استخدم عناوين مناسبة في طباعة النتائج ، واطبع الأعداد الحقيقية في النتائج بصيغة عشرية (decimal format) **مناسبة.**

السؤال السادس (١٦ درجة) :

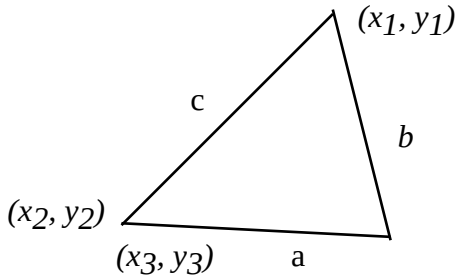
المعادلة التالية تعطي المسافة بين نقطتين إحداثياتهما (X_1,Y_1) , (X_2,Y_2)

$$\text{dist} = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

اكتب برنامجا يقرأ إحداثيات ثلاث نقط $(X_1,Y_1,X_2,Y_2,X_3,Y_3)$ **ثم يحسب ويطبع مساحة المثلث المكون من هذه النقاط الثلاث باستخدام العلاقة التالية :**

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$$

حيث a , b , c **تمثل أطوال أضلاع المثلث ، و** s **هي نصف مجموع هذه الأطوال (انظر الشكل).**



السؤال السابع (٢٠ درجة) :

تقوم إحدى دور نشر الكتب الإسلامية ببيع ثلاث مجموعات من كتب التفسير بالأسعار التالية :

المجموعة الأولى (تفسير القرطبي) سعرها ٤٥,٧٥٠ دينارا

المجموعة الثانية (تفسير ابن كثير) سعرها ١٠,٥٠٠ دينارا

المجموعة الثالثة (في ظلال القرآن) سعرها ١٦ دينارا

وفي معرض الكتاب الإسلامي تقوم دار النشر بتقديم عرض خاص على هذه المجموعات بناءا على القيمة الكلية للمبيعات منها ، كما يلي :

إذا كانت القيمة الكلية للمبيعات >= ١٠٠٠ دينارا فإن معدل الخصم (discount rate = ١٥٪)

إذا كانت القيمة الكلية للمبيعات < ١٠٠٠ و >= ٥٠٠ دينارا فإن معدل الخصم = ١٠٪

إذا كانت القيمة الكلية للمبيعات < ٥٠٠ و >= ٢٥٠ دينارا فإن معدل الخصم = ٥٪

اكتب برنامجا يقرأ الكمية المطلوبة من كل من هذه المجموعات الثلاث (NSet1, NSet2, NSet3) **، ويحسب ويطبع القيمة الكلية** (total purchase) **، وقيمة الخصم** (discount amount) **، والقيمة النهائية بعد الخصم** (net price) **وجميع القيم بالدينار.**

ثانيا : نماذج للاختبار الفصلي الثاني

الاختبار الفصلي الثاني

رقم (١)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول :

أ) (١٠ درجات) ما هي مخرجات قطعة البرنامج التالية :

```
Sum := 0 ;
A := 7 ;
while A < 10 do
begin
for K := A to 10 do
Sum := Sum + K ;
A := A+1
end ;
writeln (Sum)
```

ب) (١٥ درجة) اكتب بالضبط مخرجات البرنامج التالي :

```
Program Add (output) ;
var
x,y,sum : integer ;
Begin
write ('+' : 4) ;
for x := 1 to 3 do
write (x : 4) ;
writeln ;
for y := 4 to 6 do
begin
write (y : 4) ;
for x := 1 to 3 do
begin
sum := x + y ;
write (sum : 4)
end ;
writeln
end
End.
```

السؤال الثاني (٢٥ درجة) :

(أ) اكتب إجراء يستقبل طول نصف قطر دائرة r ، ويحسب كلا من مساحتها (πr^2) وطول محيطها ($2\pi r$).

(ب) اكتب برنامجا يقرأ عدة قيم لطول نصف قطر دائرة ، ويحسب لكل قيمة من هذه القيم – باستخدام الإجراء السابق – كلا من مساحة الدائرة وطول محيطها. ويتوقف البرنامج حين يقرأ قيمة سالبة لطول نصف القطر.

السؤال الثالث (٢٠ درجة) :

تتبع تنفيذ البرنامج التالي وأوجد مخرجاته :

```
PROGRAM TEST2 ;
VAR A , B , G : INTEGER ;
    YES : BOOLEAN ;
PROCEDURE CHECK (X : INTEGER ; VAR Y : INTEGER ; VAR
    C : BOOLEAN) ;
    VAR A : INTEGER ;
        BEGIN
            A := X + Y ;
            X := 2*Y ;
            C := X = Y ;
            G := A ;
        WRITELN ('X = ' , X) ;
    END ; {PROCEDURE CHECK}

BEGIN {MAIN PROGRAM}
        A := 6 ;
        B := 5 ;
        G := 3 ;
    YES := TRUE ;

WRITELN ('A = ' , A) ;      WRITELN ('B = ' , B) ;
WRITELN ('G = ' , G) ;      WRITELN ('YES = ' , YES) ;

CHECK (A , B , YES) ;
WRITELN ('A = ' , A) ;      WRITELN ('B = ' , B) ;
WRITELN ('G = ' , G) ;      WRITELN ('YES = ' , YES) ;
END.
```

السؤال الرابع (٣٠ درجة) :

تحسب قيمة زكاة المال (Zakat) على المدخرات الكلية (total savings) لأي شخص كما يلي :

- إذا لم يحل على المدخرات حول (أي لم تنقض عليها سنة) فإن Zakat = 0.
- إذا حال عليها حول فإن :

إذا كان TotalSavings >= Alnisab ... Zakat = 0.025*TotalSavings

إذا كان TotalSavings < Alnisab ... Zakat = 0

اكتب دالة ZAKAT تستقبل

Alnisab , TotalSavings , YearPassed

حيث

إذا حال الحول ..... YearPassed = 'Y'

إذا لم يحل الحول ..... YearPassed = 'N'

وتعطي الدالة قيمة الزكاة.

ثم استخدم هذه الدالة في برنامج :

١) يقرأ البيانات التالية لكل شخص في قائمة تحتوي على عدد M من الأشخاص (حيث M أحد المدخلات) :
   - الرقم التعريفي IDnumber
   - قيمة المدخرات الكلية TotalSavings
   - رمز YearPassed للدلالة على حولان أو عدم حولان الحول
٢) يستدعي الدالة ZAKAT لحساب قيمة الزكاة لكل شخص في القائمة.
٣) يطبع الرقم التعريفي والمدخرات الكلية وزكاة المال لكل شخص.

(افرض أن قيمة النصاب حاليا تساوي Alnisab = 500 K.D.)

الاختبار الفصلي الثاني

رقم (٢)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة وربع

السؤال الأول : (١٤ درجة)

تقدر زكاة الثمار ZF بالقاعدة التالية :

إذا كانت كمية الثمار F أقل من مقدار معين معطى A (يسمى النصاب) فإن الزكاة تساوي صفرا : ZF = 0

وما عدا ذلك فإنه :

- **إذا كان الري طبيعيا (أي بالمطر) فإن :** 10/100*F
- **إذا كان الري صناعيا (أي بالآلة) فإن :** 5/100*F
- **إذا كان الري مزيجا من الطبيعي والصناعي فإن :** 7.5/100*F

ZF =

البرنامج التالي يحسب زكاة الثمار باستخدام القاعدة السابقة.

المطلوب : إعادة كتابة البرنامج باستخدام عبارة if بدلا من عبارة case.

```
program ZakaFruit (input , output) ;
                                    var
                                      A , F , ZF : real ;
                                      W          : char ;
                                    begin
                                      readln (A , F , W) ;
                                        if F < A then
                                          ZF := 0.0
                                              else
                                            case W of
                                    'N' : ZF := 0.1*F ;
                                    'A' : ZF := 0.05*F ;
                                    'C' : ZF := 0.075*F
                                            end ;
                          writeln ('F = ' , F ,'W = ', W , 'ZF = ' , ZF)
                                    end.
```

السؤال الثاني : (١٠ درجات)

نفرض أن لدينا الإعلانات والتعريفات التالية في برنامج رئيسي

```
var
```

```
x , y : integer ;
r     : real ;
c     : char ;
procedure p (i : integer ; var a : real ; var ch : char) ;
begin
...
end ;
function f (i : integer ; a : real ; ch : char) : real ;
begin
...
end ;
```

ضع دائرة حول أي وسيط فعلى خاطئ أو أي استخدام خاطئ للدالة أو الإجراء وذلك في مجموعة الاستدعاءات التالية ، مع بيان سبب الخطأ.

```
a) ;p (x+y , 2*r , c)
b) ;p (5 , r , 'x')
c) ;p (x , r , ch)
d) ;f (x , r , c)
i) ;y := f (x , r , c)
```

السؤال الثالث : (١٦ درجة)

أوجد مخرجات البرنامج التالي :

```
program main (input , output) ;
var
a , b : integer ;
procedure p(var x , y : integer) ;
begin
x := 2*x ;
y := y+1 ;
writeln('p : ' , 'x = ' , x , 'y = ' , y) ;
end ; {p}
procedure q(x : integer ; var y : integer) ;
begin
x := x+1 ;
y := 2*y ;
writeln ('q : ', 'x = ' , x , 'y = ' , y) ;
end ; {q}
begin
a := 3 ;
b := 4 ;
p(a , b) ;
```

```
writeln ('main : ' , 'a = ' , a , 'b = ' , b) ;
q(a , b) ;
writeln ('main : ' , 'a = ' , a , 'b = ' , b) ;
end.
```

السؤال الرابع : (٢٨ درجة)

(أ) (١٤ درجة)

اكتب إجراء يستقبل عددين حقيقيين a , b **ويعيد متوسطهما الحسابي** av **، ومتوسطهما الهندسي** gm **والذي يعطى بالعلاقة** $gm = \sqrt{ab}$ **، وكبرى القيمتين** a , b.

(ب) (١٤ درجة)

اكتب برنامجا رئيسيا :

- **يستمر في قراءة عددين مكتوبين على سطر واحد ،**
- **ويستدعي الإجراء السابق لحساب المتوسط الحسابي للعددين ومتوسطهما الهندسي وأكبر العددين قيمة**
- **ثم يطبع هذه النتائج ،**

ويتوقف البرنامج حين يقرأ عددا سالبا.

السؤال الخامس : (٣٢ درجة)

(أ) (١٢ درجة)

اكتب دالة رمزية القيمة (character valued function) **تأخذ ثلاث درجات صحيحة** m1 , m2 , m3 **وتعطي التقدير بالحروف** (letter grade) **المقابل لمتوسط الدرجات الثلاث مقربا** (rounded) **لأقرب عدد صحيح ، وذلك بناء على المقياس التالي :**

A: 90-100 , B: 80-89 , C: 70-79 , D: 60-69 , F: 0-59

(ب) (٢٠ درجة)

اكتب برنامجا رئيسيا يقرأ رقما تعريفيا ID **ودرجات ثلاث ويستدعي الدالة السابقة لإيجاد التقدير المقابل بالحروف ، وذلك لكل طالب في فصل به عدد** N **من الطلاب. كذلك يوجد البرنامج أعداد الطلاب الحاصلين على**

التقديرات المختلفة المذكورة (أى عدد الطلاب الحاصلين على A ، وعدد الطلاب الحاصلين على B ، ... الخ)

الاختبار الفصلي الثاني

رقم (٣)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة وربع

السؤال الأول (٣٠ درجة):

اكتب بالضبط مخرجات كل من قطع البرامج التالية :

(أ) (١٠ درجات)

```
WriteLn('In the Name of Allah') ;
FOR i := 1TO 3 DO
CASE (2*i - 1) OF
1 : Write ('O Mojahidoon of Bosnia') ;
2 : Write ('Praise be to Allah') ;
3 : WriteLn ('Assalamo A''laikom') ;
4 : WriteLn ('Victory of Martyrdom') ;
5 : Write ('May Allah bless you')
END ;
```

(ب) (١٠ درجات)

```
Sum := 0 ;
Product := 1 ;
Count := 1 ;
REPEAT
Sum := Sum + Count mod 2 ;
Product := Product * Count ;
WriteLn (Sum : 5 , Product : 5) ;
Coun := Count + 1
Until Count = 5 ;
```

(ج) (١٠ درجات)

```
FOR i := 1 TO 3 DO
BEGIN
FOR j := 1 TO 4 DO
Write (i*j : 5) ;
WriteLn
END ;
```

السؤال الثاني (١٢ درجة)

تتبع تنفيذ البرنامج التالي ، واذكر نتائج تنفيذ عباراته ، واكتب مخرجاته.

```
PROGRAM Trace (Input , Output) ;
                                VAR
                A : ARRAY [1..6] OF Integer ;
                                  i  : Integer ;
                                              BEGIN
                                    FOR i := 1 TO 3 DO
                                                BEGIN
                                A[i] := 2*i - 1 ;
                                A[i+3] := i*i-2
                                                END ;
                                FOR i := 1 TO 6 DO
                        WriteLn ('A(',i:2,') = ', A[i])
                                                END.
```

السؤال الثالث (١٦ درجة)

(أ) تتبع تنفيذ قطعة البرنامج التالية وذلك باستخدام مجمـوعة البيانات المعطاة أسفل القطعة. اكتب نتائج تنفيذ جميع عبارات الإسـناد. ما هـي القيمة النهائية لـكل من x و c ؟

```
                                c := 0 ;
                                i := 1 ;
                        WHILE i <= 5 DO ;
                                    BEGIN
                                Read (x) ;
                IF (x < 0) or (x > 10) THEN
                                c := c + 1 ;
                                    i := i + 1 ;
                                    END ; {While}
```

(ب) أعد كتابة القطعة السابقة باستخدام عبارة Repeat.

السؤال الرابع (١٦ درجة)

اكتب برنامجا يحسب ويطبع مجموعة المتسلسلة

$$\sum_{i=1}^{10}(4i+1)^2 = 5^2 + 9^2 + 13^2 + 17^2 + \cdots + 41^2.$$

السؤال الخامس (26 درجة)

نفرض أن كل حاج قد أعطي رقما تعريفيا ID (عبارة عن عدد صحيح) وشفرة رمزية (character code) CODE (عبارة عن حرف) للإشارة إلى كيفية وصوله لأداء فريضة الحج ، حيث CODE تساوي ' L ' أو ' S ' أو ' A ' للدلالة على أنه وصل برا أو بحرا أو جوا على الترتيب.

اكتب برنامجا لقراءة هذه البيانات الخاصة بالحجاج بفرض أن قيمة نهائية (حارسة) (sentinel value) للشفرة ' E ' = CODE ستستخدم للدلالة على انتهاء البيانات. ثم يقوم البرنامج بإيجاد وطباعة :

(أ) الأرقام التعريفية ID للحجاج الذين وصلوا جوا.

(ب) أعداد الحجاج NA , NS , NL الذين وصلوا برا وبحرا وجوا على الترتيب.

(ج) العدد الكلي للحجاج N.

(د) النسب المئوية PA , PS , PL للحجاج الذين وصلوا برا وبحرا وجوا على الترتيب .

(مثلا : $PL = \frac{NL}{N} x100\%$)

الاختبار الفصلي الثاني

رقم (٤)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (١٥ درجة) :

(أ) (٥ درجات) أوجد بالضبط مخرجات قطعة البرنامج التالية :

```
FOR i := 1 TO 5 DO
    BEGIN
        FOR j := i TO 5 DO
        Write ('*' : 5) ;
            WriteLn
    END;
```

(ب) (١٠ درجات) تتبع تنفيذ قطعة البرنامج التالية مستخدما البيانات المدخلة المعطاة واكتب المخرجات المقابلة.

Input 77.2 94.9 66.1 59.4 89.45 -1

```
Write('Enter Score >') ;
Read (Score) ;
WHILE Score >= 0.0 DO
    BEGIN
    CASE Round (Score) of
        0..59 : BEGIN
            Grade := 'F'
            Points := 0 ;
WriteLn('Score=',Score:5:2,'Grade=', Grade,
    'Points=',Points:3)
            END;
        60..69 : BEGIN
            Grade := 'D' ;
            Points := 1 ;
WriteLn('Score=',Score:5:2,'Grade=', Grade,
    'Points=',Points:3)
            END ;
        70..79 : BEGIN
            Grade := 'C' ;
            Points := 2 ;
WriteLn('Score=',Score:5:2,'Grade=', Grade,
```

```
        'Points=',Points:3)
                                    END ;
                                   80..89 : BEGIN
                              Grade := 'B' ;
                                Points := 3 ;
    WriteLn('Score=',Score:5:2,'Grade=', Grade,
        'Points=',Points:3)
                                    END ;
                                  90..100 : BEGIN
                              Grade := 'A' ;
                                Points := 4 ;
    WriteLn('Score=',Score:5:2,'Grade=', Grade,
        'Points=',Points:3)
                                    END ;
                                        END ;
  Write('Enter next score (when done enter a negative score) > ') ;
                                      Read (Score)
                                    END ;
```

السؤال الثاني (٢٠ درجة):

(أ) أوجد بالضبط مخرجات قطعة البرنامج التالية باستخدام بيانات الإدخال المعطاة.

```
                                i := 1 ; Sum := 0 ; Product := 1 ;
                                    WHILE i <= 11 DO
                                        BEGIN
                                      Read (x) ;
                            IF (x < 10) or (x > 20) THEN
                               Product := product * i ;
                                    Sum := Sum + x ;

    WriteLn('i=',i:5,'x=',x:5,'Sum='sum:5,'Product=',Product:5);
                                                i := i + 2
                                      END ;  {While}
```

Input -1 9 15 22 10 30

(ب) أعد كتابة قطعة البرنامج السابقة مستخدما عبارة REPEAT...UNTIL .

السؤال الثالث (١٥ درجة) :

اكتب برنامجا يقرأ قيمة عدد حقيقي x وقيمة عدد صحيح n ، ثم يحسب ويطبع قيمة مجموع المتسلسلة

$$\sum_{i=1}^{n} \frac{\tan^2(2i-1)x}{(2i-1)x} = \frac{\tan^2 x}{x} + \frac{\tan^2 3x}{3x} + \frac{\tan^2 5x}{5x} + \cdots + \frac{\tan^2(2n-1)x}{(2n-1)x}.$$

*** لاحظ أن tan t ليست دالة قياسية في التيربو باسكال ، ويمكن استخدام العلاقة**

$$\tan t = \sin t / \cos t$$

السؤال الرابع (٢٢ درجة) :

اكتب برنامجا يسمح للمستخدم بإدخال رقم تعريفي ID للطالب ، ودرجاته الاختبارية الثلاث (S1 , S2 , S3) وذلك لعدد من الطلاب. ويحسب البرنامج ويطبع الدرجة الاختبارية المتوسطة Ave لكل طالب ، ويحسب كذلك ويطبع عدد الطلاب Naالحاصلين على درجة متوسطة > ٩٠ وعدد الطلاب Nf الحاصلين على درجة متوسطة < ٦٠. ويتوقف البرنامج حين يدخل المستخدم رقما تعريفيا ID صفريا.

السؤال الخامس (٢٨ درجة) :

(أ) اكتب إجراء يقرأ طول ضلعي مستطيل A , B ، ويحسب ويطبع كلا من محيطه ومساحته.

(ب) اكتب إجراء يقرأ طول ضلع مربع D ، ويحسب ويطبع كلا من محيطه ومساحته.

(ج) اكتب إجراء يقرأ طول نصف قطر دائرة R ، ويحسب ويطبع كلا من محيطها ومساحتها.

(د) اكتب برنامجا يقرأ عددا شفريا لشكل هندسي (1 : مستطيل ، 2 : مربع ، 3 : دائرة) ، ثم يستدعي الإجراء المناسب من بين الإجراءات (أ) ، (ب) ، (ج) لحساب وطباعة كل من محيط الشكل ومساحته.

الاختبار الفصلي الثاني

رقم (٥)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (١٢ درجة):

تتبع تنفيذ البرنامج التالي واكتب مخرجاته بالضبط.

```
program tradition (output) ;

procedure QuesAns1 ;
begin
writeln('Q: Who is the most deserving of my good company?')
;
writeln('A: Your mother.')
end;

procedure QuesAns2 (n: integer) ;
var i: integer ;
begin
for i := 1 to n do
begin
writeln ('Q: Who is next?') ;
writeln ('A: Your father.')
end
end ;

procedure QuesAns3 (n: integer) ;
var i : integer ;
begin
for i := 1 to n do
begin
writeln ('Q: Who is next?') ;
writeln ('A: Your mother.')
end
end ;
begin (* main program *)
QuesAns1 ;
QuesAns3 (2) ;
QuesAns2 (1)
end.
```

السؤال الثاني (١٢ درجة):

اكتب بصورة واضحة ودقيقة مخرجات قطعة البرنامج التالية ، مستخدما الرمز □ للدلالة على فراغ ، والرمز ↵ للدلالة على سطر فارغ ، مع بيان خطوات تتبع تنفيذ القطعة.

```
...
                for k := 5 downto 1 do
                                begin
                    for i := 1 to 5 - k do
                        write ('□')
                            (*end for*)
                        writeln ('*') ;
                    for j := 1 to k - 1 do
                    write ('□□') ;
                            (*end for*)
                        writeln ('*') ;
                                    end
                                (*end for*)
...
```

السؤال الثالث (١٢ درجة) :

نفرض أن لدينا التعريفات التالية :

```
var i , j : integer ;
    a , b : real ;
    c : boolean ;
procedure F(t:boolean; m:integer; var n:integer; x:real; var y:real) ;
```

أي استدعاءات الإجراء التالية صحيحة وأيها غير صحيحة (أجب : نعم أو لا على الترتيب) ؟ وإن كانت الإجابة لا ، فضع خطا أسفل الوسيط الفعلي الخاطئ.

a) F(c , i , a , j , b)
b) F(not c , i+j , i , j , b)
c) F(c , a+j , a , j , b)
d) F(3 < 2 , 2 , i , j , a)
e) F(c and (3>1) , 2 , 2 , 3 , a)
f) F(true , 3 , i , a , b)

السؤال الرابع (١٢ درجة) :

هل قطعتا البرنامج المتقابلتان فيما يلي متكافئتان أم لا (أجب : نعم أو لا) ؟

اكتب مخرجات كل من القطعتين مستخدما الرمز □ للدلالة على فراغ.

a) Yes / No ?

(ii)

```
power := 1 ;
repeat
write (power : 3) ;
power := power +1
until power < 10
(*end repeat*);
```

(i)

```
power := 1 ;
while power < 10 do
begin
write (power : 3) ;
power := power +1
end
(*end while*);
```

b) Yes / No ?

(ii)

```
n := 3 ;
for line := 1 to n do
writeln ('***')
(*end for*) ;
```

(i)

```
n := 3 ;
line := 1 ;
repeat
writeln ('***')
until line <= n
(*end repeat*) ;
```

السؤال الخامس (٢٠ درجة) :

اكتب برنامجا يقرأ درجات طلاب إحدى الشعب ، ويطبع الدرجة المتوسطة لكل طالب. يبدأ البرنامج بقراءة عددين صحيحينM , N حيث N هو عدد الطلاب في الشعبة ، و M هو عدد الدرجات لكل طالب. ثم يقرأ البرنامج درجات الطالب الأول (وعددها M درجة) ، ويحسب ويطبع درجته المتوسطة ، ثم يقرأ درجات الطالب الثاني ، ... وهكذا لجميع الطلاب (وعددهم N).

السؤال السادس (٢٠ درجة) :

اكتب برنامجا يقرأ متتابعة (Sequence) من الأعداد الصحيحة الموجبة ، وتنتهي المدخلات بإدخال أي قيمة سالبة. ويطبع البرنامج ثلاث قيم :

*** أكبر عدد * أصغر عدد * المتوسط الحسابي**

السؤال السابع (١٢ درجة) :

اكتب برنامجا يوجد عدد جميع الأعداد الصحيحة التي يقبل كل منها القسمة على كل من ٣ و ٧ (في الوقت نفسه) بدون باق ، وذلك في مدى الأعداد الصحيحة ما بين ١ و ١٠٠٠.

مثلا : العدد ٢١ يدخل في هذا العد ، بينما العدد ١٤ أو العدد ٣٠ لا يدخل في هذا العد.

الاختبار الفصلي الثاني

رقم (٦)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة ونصف

السؤال الأول (١٠ درجات):

اكتب بالضبط مخرجات البرنامج التالي :

```
PROGRAM Q1a (Input, Output);
VAR
   I, J, Counter : Integer;
BEGIN
   Counter := 0;
   FOR  I := 3  DOWNTO  -1  DO
     BEGIN
          FOR  J := 0  TO  4  DO
           Counter := Counter + 1;
          Counter := Counter + 1
     END;
   Writeln ( 'Counter = ', Counter:4 )
END.
```

السؤال الثاني (٢٠ درجة):

(أ) (٥ درجات) ما هي قيمة Sum بعد تنفيذ قطعة البرنامج التالية ؟

```
Sum := 0;
FOR L := 1 TO 11 DO
   IF L MOD 5 = 0 THEN
     Sum := Sum + L;
```

(ب) (٥ درجات) ما هي قيمة K بعد تنفيذ قطعة البرنامج التالية ؟

```
K := 3;
M := 2;
J := 3;
WHILE J <= 5 DO
   BEGIN
     N := M * J - 3;
     K := K + N;
     J := J + 2
   END;
```

(ج) (١٠ درجات) اكتب بالضبط مخرجات قطعة البرنامج التالية :

```
K := 5;
```

```
WHILE  K >= 1  DO
    BEGIN
      FOR  I := 1  TO  5 - K  DO
            Write( '.' );
      FOR  J := 1  TO  2 * K - 1  DO
            Write( 'B' );
      Writeln;
      K := K - 1
    END;
```

السؤال الثالث (١٠ درجات):

(أ) ما هي مخرجات قطعة البرنامج التالية لكل قيمة من قيم المدخلات الثلاث :

(i) X = 1 , (ii) X = 3 , (iii) X = 5

```
Readln  ( X );
WHILE X <= 3 DO
    BEGIN
      Writeln ( 'Number = ', X:2 );
      X := X + 1
    END;
```

(ب) أعد كتابة قطعة البرنامج السابقة باستخدام عروة REPEAT **بدلا من عروة** WHILE **. ما هي المخرجات الآن المقابلة لقيم** X **السابقة المذكورة في (أ) ؟**

السؤال الرابع (١٠ درجات):

اكتب برنامجا يقرأ ١٠ أعداد صحيحة ، ثم يطبع كل عدد من هذه الأعداد ، تتبعه رسالة تفيد ما إذا كان العدد فرديا odd **أم زوجيا** even **. استخدم بنية** CASE.

السؤال الخامس (١٥ درجة):

اكتب برنامجا يشتمل على إجرائين : أحدهما لجمع عددين والآخر لطرح عددين . ويقوم البرنامج الرئيسي بقراءة عددين Num1 , Num2 **ورمز** OP (character) **للدلالة على اختيار للمستخدم** (user's choice)**. وبناء على هذا الاختيار فإن البرنامج يقوم إما بجمع العددين أو طرحهما. كما يقوم البرنامج الرئيسي بطباعة النتيجة.**

السؤال السادس (١٥ درجة):

اكتب برنامجا لتعيين أصغر قيمة للعدد الصحيح N بحيث يكون المجموع

$$1 + 2 + 3 + ... + N$$

أكبر من أو يساوي ١٠٠.

السؤال السابع (٢٠ درجة):

اكتب برنامجا لقراءة متتابعة (sequence) من أعداد صحيحة موجبة ، وإيجاد وطباعة كل من :

i. **عدد مرات ظهور العدد** 3 **في هذه المتتابعة.**

ii. **ترتيب** (index / position) **آخر ظهور للعدد** 3 **في هذه المتتابعة ،**

على أن يطبع البرنامج قيمة الترتيب صفرا إذا لم يظهر العدد 3 **في المتتابعة. وتنتهي المدخلات عند إدخال عدد صحيح سالب.**

مثال : بالنسبة لمتتابعة الأعداد الصحيحة :

1 , 9 , 12 , 4 , 3 , 8 , 3 , 6 , -1

عدد مرات ظهور العدد 3 **هو** 2

وترتيب آخر ظهور للعدد 3 **هو** 7

الاختبار الفصلي الثاني

رقم (٧)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعة وربع

السؤال الأول (أ) (٥ درجات) :

أعد كتابة عبارة IF التالية باستخدام عبارة CASE مكافئة.

```
VAR
    Number, Value : integer; .
      .
      .
IF ( 0 <=  Number ) AND ( Number < 2 )  THEN
    Value := Number
ELSE IF ( Number = 2 ) OR (Number = 3) THEN
          Value := 2 * Number
ELSE IF Number  = 5  THEN
               Value := 3 * Number
ELSE  IF ( 6 <=  Number ) AND ( Number <= 9)  THEN
            Value := 4 * Number;
```

(ب) (١٥ درجة)

(i) **تتبع تنفيذ البرنامج التالي وأوجد مخرجاته .**

```
PROGRAM Q1b(Input, Output);
VAR
  X, Y, Z, Sum: integer;
BEGIN
   readln( X );
  Y := 0;    Sum := 0;
   IF X  > 0 THEN
     REPEAT
      Z := X MOD 10;
      X := X DIV 10;
      Sum := Sum + Z;
      Y := Y * 10 + Z;
      writeln (' X = ', X: 4, 'Y = ',Y :4, Z = ', Z : 4)
     UNTIL   X = 0;
      writeln (' Sum = ', Sum : 4)
END.
```

```
INPUT  163
```

(ii) أعد كتابة البرنامج السابق باستخدام عبارة WHILE .. DO

(ج) (١٠ درجات) ما هي مخرجات البرنامج التالي ؟

```
PROGRAM Q1c(Input, Output);
VAR
   i, j :integer;
   Flag : boolean;
BEGIN
   j := 20;   Flag := TRUE;
   WHILE ( j > 0 ) AND Flag DO
       BEGIN
      readln ( i );
      IF ( i MOD 2 ) <> 0 THEN
         WHILE ( j - i ) > 0 DO
            BEGIN
               writeln ('  i = ', i , ' j = ', j );
               i := Sqr( i ) + 1
            END
          ELSE
            Flag := FALSE;
        j := j - 5
       END
       writeln ( FLAG)
END.
```

```
INPUT  3
       5
       4
```

السؤال الثاني (١٠ درجات) :

تتبع مخرجات البرنامج التالي .

```
PROGRAM Q2 (Input, Output);
VAR
  X, Y, Z, Sum : integer;
PROCEDURE P1( N1, N2, N3 : integer; VAR S : integer);
BEGIN
   N1 := N1 * 2;
   N2 := N1 + 1;
   S := N1 + N2 + N3 + S;
   writeln ( ' P1 : ', N1 : 4, N2 : 4, N3 : 4, S : 4)
END;
```

```
PROCEDURE P2( A, B : integer; VAR C : integer);
VAR
   Total : integer;
BEGIN
    A := ( A * 2 ) MOD 2;
    B := B DIV 2;
    C :=  C + 1;
    Total := A + B  + C ;
    writeln ( ' P2 : ', A : 4, B : 4, C : 4, Total : 4)
END;
BEGIN   { Main }
    X := 1; Y := 3; Z := 8; Sum := 1;
    P1(X, Y, Z, Sum);
    writeln(' Main: X : 4, Y : 4, Z : 4, Sum : 4);
    P2(X, Y, Z);
    writeln(' Main: X : 4, Y : 4, Z : 4, Sum : 4)
END.
```

السؤال الثالث (١٠ درجات) :

نفرض أننا قد أعطينا الإعلانات التالية والإجراء Check.

```
VAR
      i,j :integer;
      A :real;
      B :boolean;
      C : char;
PROCEDURE Check ( M : integer; VAR X : real; VAR N : integer ;
                  VAR Ch: char;  Y:  boolean);
BEGIN
          .
          .
 END ;
```

أي الاستدعاءات التالية لهذا الإجراء صحيحة ، وأيها غير صحيحة ، مع بيان السبب إن كان الاستدعاء خاطئا.

الاستدعاء	T/F	سبب الخطأ
Check (i, j, i ,C, B);		
Check (4+i, A, j, C, TRUE);		
Check (i, A, j, C);		
Check (i, 4/2, j, C, FALSE);		
Check (5 DIV 2, A, i, C, j > 0);		

السؤال الرابع (١٥ درجة) :

اكتب برنامجا يقرأ متتابعة (sequence) من أعداد صحيحة موجبة. ويتم إدخال القيمة صفر كعلامة على انتهاء المدخلات. ولكل عدد صحيح مدخل يقوم البرنامج بحساب مجموع مربعات الأعداد الصحيحة من 1 إلى هذا العدد الصحيح. مثلا إذا كان العدد الصحيح يساوي 4 فإن البرنامج يحسب المجموع $1^2 + 2^2 + 3^2 + 4^2 = 30$ **، ويطبع البرنامج كلا من قيمة العدد الصحيح ومجموع المربعات المقابل.**

السؤال الخامس (١٥ درجة) :

اكتب برنامجا لحساب حجم ومساحة سطح اسطوانة مصمتة ، حيث يقرأ البرنامج كلا من نصف القطر r والارتفاع h .

استخدم إجراء لحساب الحجم وآخر لحساب مساحة السطح. ويجب إعادة القيمتين المحسوبتين إلى البرنامج الرئيسي الذي يقوم بطبعهما.

إرشاد : **حجم الاسطوانة المصمتة** $= \pi r^2 h$

مساحة سطح الاسطوانة $= 2 \pi r (r+h)$

استخدم القيمة التقريبية $\pi = 3.14159$

السؤال السادس (٢٠ درجة) :

اكتب برنامجا يقرأ قيمة عدد صحيح موجب N يمثل عدد الطلاب في أحد الفصول. ويقرأ البرنامج لكل طالب رقما تعريفيا صحيحا (integer ID identification number) **ودرجاته في ثلاثة اختبارات** Mark1 , Mark2 , Mark3.

ويطبع البرنامج لكل طالب رقمه التعريفي ID **وتقديره** Grade **الذي يحسب كمتوسط درجاته الثلاث. كما يقوم البرناج بطباعة أعلى تقدير في الفصل** Max **، وأقل تقدير** Min **، والتقدير المتوسط للفصل** Ave.

ثالثا : نماذج للاختبار النهائي

الاختبار النهائي

رقم (١)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

السؤال الأول (١٠ درجات) :

نفرض أن i , j , k **متغيرات صحيحة ، حيث** i = 4 , j = 5 , k = 6 **. ونفرض أن** B **متغير منطقي ، حيث** B = False.

أوجد نوع (type) **وقيمة** (value) **كل من التعابير التالية المكتوبة بلغة الباسكال.**

أ) i + j div 3 * k - 1

ب) suce (sqr(i)) + ord (k)

ج) (j + 1 >= k) and not B

د) round (9.4) + sqrt (i)

هـ) pred (chr (ord('A') + 2))

السؤال الثاني (١٦ درجة) :

تحسب الدرجة النهائية Score **للطالب على أنها مجموع ثلاث درجات :**

Test1 : **درجة الاختبار الفصلي الأول**

Test2 : **درجة الاختبار الفصلي الثاني**

Final : **درجة الاختبار النهائي**

ويعطى الطالب تقديرا حرفيا (letter grade) Grade **تبعا للجدول التالي :**

الدرجة Score	٩٠–١٠٠	٨٠–٨٩	٧٠–٧٩	٦٠–٦٩	صفر–٥٩
التقدير Grade	A	B	C	D	F

اكتب برنامجا يقرأ

أ) **عدد الطلاب** M

ب) **بيانات كل طالب ، وهي :**

رقمه التعريفي : IdNo

درجاته الثلاث : Test1 , Test2 , Final

ويطبع البرنامج الرقم التعريفي والتقدير الحرفي لكل طالب.

السؤال الثالث (١٥ درجة) :

اكتب برنامجا يقرأ متتابعة من الأعداد الصحيحة الموجبة ، وتنتهي المدخلات بإدخال قيمة سالبة ، ويطبع البرنامج :

أ) عدد "الأعداد الفردية" في هذه المدخلات.

ب) عدد "الأعداد الزوجية التي تقبل القسمة على ١١ بدون باقي "في هذه المدخلات.

السؤال الرابع (١٠ درجات) :

تتبع تنفيذ البرنامج التالي وأوجد مخرجاته بفرض أن مدخلاته هي :

2 3 2 5 1 ↵

```
program main (input , output) ;
var
        c1 , c2 : char ;
        cno : integer ;
        i    : integer ;
begin
        c2 := 'a' ;
        for i := 1 to 5 do
        begin
                read (cno) ;
                if (cno >= 1) and (cno <= 4) then
                case cno of :
                        1 : c1 := 'd' ;
                        2 : c1 := 'c' ;
                        3 : c1 := 'b' ;
                        4 : c1 := 'a'
                end ;
        if c2 = pred (c1) then
        begin
                writeln ('right' , c1) ;
                c2 := c1
        end
        else
                writeln ('wrong' , c1)
```

```
    end {for}
end.
```

السؤال الخامس (١٦ درجة) :

تتبع تنفيذ البرنامج التالي ، واكتب نتائج تنفيذ جميع عبارات الاسناد ، وأوجد مخرجاته.

```
program main (output) ;
var
    w , x , y , z : integer ;
procedure P1 (a : integer ; var x : integer) ;
    var y : integer ;
    begin {P1}
        w := a ;
        a := 2*a ;
        y := w + z ;
        writeln ( 'P1 : y = ' , y , 'w = ' , w , 'a = ' , a) ;
        x := x + y
    end ; {P1}
begin {main}
    w := 40 ;
    x := 5 ;
    y := 10 ;
    z := 20 ;
    P1 (x , y) ;
    writeln ('w = ' , w , 'x = ' , x , 'y = ' , y ,'z = ' , z)
end. {main}
```

السؤال السادس (١٦ درجة) :

أ) اكتب برنامجا فرعيا داليا (function) Average **يستقبل كوسيط** (parameter) **منظومة** A **من النوع** (type) ArrayType **(المعرَّف فيما يلي)، ويعيد القيمة المتوسطة لعناصر هذه المنظومة.**

إفرض أن لدينا التعريفات والإعلانات التالية :

```
const
    SIZE = 50 ;
type
    ArrayType = array [1..SIZE] of integer ;
```

ب) اكتب برنامجا

i) يقرأ عناصر منظومة L من النوع ArrayType.

ii) يستخدم البرنامج الفرعي الدالي Average لإيجاد القيمة المتوسطة لعناصر المنظومة L.

iii) يطبع عناصر L التي تزيد قيمة كل منها عن القيمة المتوسطة.

السؤال السابع (١٧ درجة) :

تتبع تنفيذ البرنامج التالي ، واكتب نتائج تنفيذ جميع عبارات الإسناد ، وأوجد مخرجاته.

```
program main (output) ;
const
  MaxSize = 6 ;
type
  AType = array [1..MaxSize] of integer ;
var
  j : integer ;
  X , Y : AType ;
procedure Double (A: AType ; var B: AType ; SA, SB, Len : integer)
;
var
  N , I , J : integer ;
begin
  I := SA , J := SB , N := 1 ;
  While N <= Len do
    begin
      B[J] := 2*A[I] ;
      writeln ('J = ' , J , 'B[J] = ', B[J]) ;
      I := I + 1 ;
      J := J + 1 ;
      N := N + 1
    end
end ;
begin {main}
  for i := 1 to 3 do
    X[i] := i + 1 ;
  for i := 4 to 6 do
    X[i] := 0 ;
  for i := 1 to 6 do
    write (X[i] : 3) ;
```

```
                writeln ;
            for i := 1 to 6 do
             Y[i] := 0 ;
    Double (X , Y , 1 , 2 , 3) ;
            for i := 1 to 6 do
      write (Y[i] : 3) ;
                              end.
```

الاختبار النهائي

رقم (٢)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

السؤال الأول : (١٠ درجات) تتبع تنفيذ البرنامج التالي وأوجد مخرجاته :

```
program test (input, output);
type
    Index = 1..3;
    List  = Array[Index] of Integer;
var
    K    : List;
    I, J : Integer;
begin
    for I := 1 to 3 do
        begin
            for J := 3 downto I do
                K[J] := I * J;
            K[I] := K[I] - 10
        end;
    for I := 1 to 3 do
        writeln (K[I])
end.
```

السؤال الثاني : (١٠ درجات) تتبع تنفيذ البرنامج التالي وأوجد مخرجاته :

```
Program main ( input, output );
var
    a, b, c, d : integer;

procedure P ( x : integer; var y : integer );
var
    c : integer;
begin {P}
    x := x - 1;
    c := 5;
    d := d - c;
writeln ( 'P:', 'a=', a, 'b=', b, 'c=', c, 'd=', d, 'x=', x, 'y=', y );
    y := y + 2
end; { P }
begin {main}
    a := 10;    b := 20;
```

```
                d := 40;          c := 30;
                                P ( a, b );
        writeln ( 'main:', 'a=', a, 'b=', b, 'c=', c, 'd=', d )
                                end. { main }
```

السؤال الثالث : (١٣ درجة)

اكتب برنامجا يقرأ متتابعة (sequence) من أعداد صحيحة موجبة يعقبها صفر للدلالة على انتهاء المتتابعة ، ويطبع البرنامج عدد الأعداد الزوجية التي تقبل القسمة على ٣ بدون باق وذلك من بين أعداد متتابعة الإدخال (input sequence) .

السؤال الرابع : (١٧ درجة)

اكتب برنامجا يستمر في قراءة ثلاثة أعداد حقيقية a,b,c ، ثم يحدد – ويطبع رسالة تفيد – ما إذا كانت الأعداد الثلاثة تمثل أطوال أضلاع مثلث أم لا (طول أي ضلع في مثلث يجب أن يكون أقل من مجموع طولي الضلعين الآخرين). وإذا كانت الأعداد الثلاثة تمثل أطوال أضلاع مثلث ، فإن البرنامج يطبع رسالة تفيد ما إذا كان المثلث متساوي الأضلاع (equilateral ، جميع الأضلاع متساوية) ، أو متساوي الساقين (isosceles ضلعان متساويان) ، أو مختلف الأضلاع (scalene ، جميع الأضلاع مختلفة الأطوال) . ويتوقف البرنامج حين يقرأ أي عدد سالب.

السؤال الخامس : (٢٥ درجة)

اكتب برنامجا يستخدم منظومتين : إحداهما لتخزين رموز شفرية للأقسام العلمية (department codes) والأخرى لتخزين درجات صحيحة ، وذلك لطلاب شعبة بها عدد n من الطلاب (حيث n <= 100) . ويستخدم البرنامج حرفا للدلالة على رمز القسم ، حيث 'M' ترمز الى قسم الرياضيات ، و 'C' ترمز الى قسم علم الحاسب. ويجب أن يشتمل البرنامج على :

أ) إجراء لقراءة وتخزين رموز الأقسام والدرجات في المنظومتين.

ب) دالة لحساب أعلى درجة في الشعبة كلها.

ج) دالة لحساب الدرجة المتوسطة لطلاب علم الحاسب.

ويطبع البرنامج نتائج كل من ب) و ج).

السؤال السادس : (٢٥ درجة)

أ) نفرض أن X منظومة أعداد صحيحة مختلفة عددها n . اكتب إجراء
SEARCH (X , N , Key , Index)
يبحث عن عدد صحيح Key في المنظومة X ، فإن وجده فإنه يعطي المتغير الصحيح Index قيمة ترتيب Key في المنظومة X ، وإن لم يجده فإنه يعطى Index القيمة صفرا.

ب) نفرض أن كل واحد من أصحاب رسول الله صلى الله عليه وسلم قد أعطي رقما تعريفيا صحيحا. ونفرض أن Badr منظومة تحتوي على ٣١٤ رقما تعريفيا هي أرقام الصحابة الذين اشتركوا في غزوة بدر ، وأن Ohod منظومة تحتوي على ٧٠٠ رقم تعريفي هي أرقام الصحابة الذين اشتركوا في غزوة أحد. اكتب برنامجا يقرأ قيم عناصر المنظومتين Badr , Ohod ، ويستدعي الإجراء SEARCH لمعرفة الصحابة الذين اشتركوا في كل من الغزوتين ، وطباعة أرقامهم التعريفية.

الاختبار النهائي

رقم (٣)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

(١) (١٨ درجة) نفرض أن لدينا الاعلانات التالية :

```
const i=4 , j=5 , k=6 ;
const b = false ;
(green , yellow , red) ;=type colour
```

أوجد نوع وقيمة كل من التعابير التالية المكتوبة بلغة الباسكال

```
i+j div 2-1(One
i+j / 2-1(Two
pred (sqr (i) - sqrt (i))(Three
(j+1 >= k) and not b(Four
round (9.4) * sqrt (i)(Five
succ (ord (yellow)) = ord (red).(Six
```

(٢) (٩ درجات) اكتب تعبيرا بلغة الباسكال لكل من التعابير الرياضية التالية:

(One $-x^2\frac{\sqrt{y}+2}{y}$

(Two $\frac{x*y}{x+y}-\frac{x-y}{x*y}$

(Three $\left(x^2+y^2\right)^{\frac{1}{2}}$

(٣) (١٣ درجة) أ) تتبع تنفيذ قطعة البرنامج التالية واكتب مخرجاتها.

(ملاحظة : استخدم الرمز □ للإشارة إلى الفراغ كما هو مبين فى الصيغة المعطاة).

```
var x , y : integer ;
    s : real ;
        x := 2 ;
        y := 3 ;
    if x >= 0 then
if 2*x - y > 0 then
  x := x + 1
   else x := x + 2
  (*end if *)
            (*end if *)
writeln ('x □ = □' , x : 4) ;
```

```
s := 0.0 ;
while x > 0 do
  begin
s := s + x * x ;
    x := x - 2
  end
(* end while *) ;
writeln ('s□ = □' , s : 8 : 4) ;
```

ب) تتبع تنفيذ قطعة البرنامج التالية واكتب مخرجاتها. حدد مواضع السطور الفارغة.

```
var i,n : integer ;
  line : integer ;
n := 5 ;
 repeat
line := n ;
if line mod 2 <> 0 then
for i := 1 to line do
write ('*')
(* end for *)
(* end if *) ;
writeln ;
n := n - 1
until line < 1
(* end repeat *) ;
```

(٤) (١٠ درجات) اكتب عبارة case مكافئة لعبارة if التالية :

```
var digit , value : integer ;
if (0 <= digit) and (digit <= 2) then
value := digit
else if (digit = 3) or (digit = 4) then
value := 2*digit
else if digit = 5 then
value := 3 * digit
else value ;= 4 * digit
(* end if *)
(* end if *)
(* end if *)
```

(٥) (١٠ درجات) اكتب برنامجا يحسب ويطبع كلا من حجم ومساحة سطح اسطوانة نصف قطرها r وارتفاعها h ، علما بأن r , h مدخلان للبرنامج .

ملاحظة : الحجم : $V = \pi r^2 h$

مساحة السطح : $S = 2 \pi r h$

(٦) (١٠ درجات) اكتب برنامجا لتعيين أصغر قيمة للعدد الصحيح n بحيث يكون حاصل الضرب 1*2*3*...*n أكبر من أو يساوى 10000 .

(٧) (١٠ درجات) اكتب دالة منطقية تستقبل عددين صحيحين x ,y وتعيد القيمة true إذا كان x أحد مضاعفات y أو إذا كان y أحد مضاعفات x وما عدا ذلك فإنها تعيد القيمة false .

(٨) (٢٠ درجة) أ) نفرض أننا قد أعطينا الإعلان التالي :

```
const max = 100 ;
type array_type = array [1..max] of real ;
```

اكتب دالة تسمى is_positive_array تستقبل وسيطين : متغير منظومة x من النوع array_type ، وقيمة صحيحة m تمثل العدد الحالى لمركبات (عناصر) المنظومة (حيث m أقل من أو يساوى max) ، وتعيد الدالة القيمة true إذا كان كل عنصر من العناصر –التى عددها m – فى بداية المنظومة أكبر من أو يساوى صفرا (ويقال إن المنظومة موجبة) ، وما عدا ذلك تعيد الدالة القيمة false .

ii) اكتب برنامجا رئيسيا يستخدم الدالة المعرفة في (أ) . مدخلات البرنامج : عدد عناصر المنظومة وقيم هذه العناصر. ويطبع البرنامج معلومات تفيد ما إذا كانت المنظومة المعطاة موجبة أم لا.

الاختبار النهائي

رقم (٤)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

السؤال الأول :

(أ) (٦ درجات) أوجد قيمة كل من التعابير التالية :

i) 14 * (3 + 18 DIV 4) - 50
ii) 4 * 11 MOD (trunc(trunc(8.9) / sqrt(16)))
iii) NOT (((3 - 4 MOD 3) < 5) AND (round(6 / 4) <> 3))

(ب) (٦ درجات) حوِّل عروة while **التالية إلى عروة** repeat **مكافئة لها:**

```
read(ch);
count := 0;
while (ch <> '*') and (count <= 8) do
begin
write(ch);
count := count + 1;
read(ch) ;
end ;
```

السؤال الثاني :

أ) (درجتان) أي الأنواع (types) **التالية لا يجوز أن يكون نوع التعبير الانتقائي** (selector) x **في عبارة الحالة** case x of

i) Integer ii) Real iii) Char iv) Enumerated

(ب) (درجتان) أي الأنواع التالية لا يجوز أن يكون نوع مؤشر نوع منظومة (subscript type of an array type)

i) Integer ii) Real iii) Char iv) Boolean

(ج) (درجتان) أي الأنواع التالية لا يعد من أنواع البيانات الترتيبية (ordinal data types)؟

i) Boolean ii) Real iii) Char iv) Integer

السؤال الثالث (أ) (٧ درجات) تتبع تنفيذ البرنامج التالي وأوجد مخرجاته:

```
program test (input,output);
    const
        max = 3;
    var
        k: array[1..max] of integer;
        i,j: integer;
```

```
begin
    for i:=1 to max do
    begin
        j:=max-i+1;
        repeat
            k[j]:= j + i;
            j:= j - 1
        until j = 0
    end;
    for i:=1 to max do
        writeln(k[i]);
end.
```

(ب) (٧ درجات) ما هي قيم المتغيرات t , i , s , g , z **بعد تنفيذ عروة** while

التالية بفرض أن لدينا المدخلات : input: -3 5 4 0 -4 4

```
z := 0; g := 0; s := 0; i := 1;
while (i < 7) do
begin
read( t );
s := s + t;
if ( t >= 0 ) then
g :=g + 1
else
z := z + 1;
i := i + 1
end;
```

(ج) (٦ درجات) تتبع واكتب قيم متغيرات قطعة البرنامج التالية ، واكتب مخرجاتها.

```
Count := 0;
Stop := 4;
while Count < Stop do
begin
for K := 1 to Count do
write(K:3);
writeln;
Count := Count + 1;
end;
writeln('All Done');
```

السؤال الرابع :

(أ) (٥ درجات) نفرض أن لدينا الدالة التالية حيث ArrayType **معرفة أنها منظومة أعداد صحيحة.**

```
function Result (A:ArrayType; N : integer) : integer;
                                                    var
                                                  k, i : integer;
                                                  begin
                                                    k := 1;
                                                for i := k+1 to N do
                                              if A[i] > A[k] then
                                                  k := i;
                                                  Result := k
                                                  end;
```

i) **ما هي القيمة التي يعيدها استدعاء الدالة** Result (X,5) **حيث** X **هي المنظومة** 3 8 6 2 5 ؟

ii) **ما وظيفة الدالة** Result **(أي ماذا توجد) ؟**

(ب) (١٢ درجة) تتبع تنفيذ البرنامج التالي وأوجد مخرجاته .

```
program test(output);
                                                    var
                                                  a, b : integer;
procedure One (x : integer; var y : integer);
                                                  begin
                                                    x := 2;
                                                  y := y + 1;
                                              writeln('1: ', x:5, y:5);
                                                  end;
procedure Two (var a : integer; b : integer);
                                          procedure Three (a, b : integer)
                                                  begin
                                                    a := 10;
                                                    b := 11;
                                              writeln('2: ', a:5, b:5);
                                                  end;
                                                  begin
                                                  One(b, a);
                                                  a := a + 1;
                                                  b := b * 2;
```

```
    writeln('3: ', a:5,b:5);
        Three(a, b);
    writeln('4: ', a:5, b:5);
end;
begin
    a := 4;
    b := 5;
writeln('5: ', a:5, b:5);
    One(a,b);
writeln('6: ', a:5, b:5);
    Two(a,b);
writeln('7: ', a:5, b:5);
end.
```

السؤال الخامس (١٥ درجة) :

اكتب دالةSECOND **لها وسيطان شكليان** (two formal parameters) **:**

D : **منظومة أعداد صحيحة مختلفة.**

N : **حجم المنظومة (أي عدد عناصرها).**

وتعيد الدالة قيمة ثاني أكبر عنصر (second largest element) **في المنظومة. مثلا إذا كانت عناصر** D **هي :** 4 10 6 2 3 5 **فإن الدالة** SECOND **تعيد القيمة** 6.

إرشاد : يمكن حل هذا السؤال بترتيب (sorting) **عناصر المنظومة أولا.**

السؤال السادس (١٥ درجة) :

يقال لمنظومة من الرموز (array of characters) **إنها منظومة مزدوجة القراءة** (Palindrome) **إن أمكن قراءتها طردا أو عكسا ، أي إن كانت تقرأ بالكيفية نفسها سواء من أول المنظومة لآخرها (من اليسار لليمين) أو العكس من آخرها لأولها (من اليمن لليسار). فمثلا كل من الكلمات التالية تعد منظومة مزدوجة القراءة:**

RADAR NOON POP

وذلك لأن الرمز (الحرف) الأول هو نفسه الرمز الأخير ، والرمز الثاني هو نفسه الرمز قبل الأخير ، ... وهكذا.

اكتب دالة تستقبل وسيطين : منظومة A من الرموز ، وعددا صحيحا N يمثل عدد عناصر المنظومة ، وتعيد نتيجة منطقية (boolean result) تفيد ما إذا كانت المنظومة A مزدوجة القراءة أم لا .

السؤال السابع (١٥ درجة) :

أ) اكتب برنامجا فرعيا (دالة أو إجراء) يتحقق ما إذا كان عدد معطى زوجيا أم لا. (لا تستخدم دالة التيربو باسكال ODD).

ب) اكتب برنامجا فرعيا يطبع جميع عوامل عدد صحيح معطى.

ملاحظة : يقال لعدد صحيح x إنه أحد عوامل العدد الصحيح n إذا كان x يقسم n (أي أن n يقبل القسمة على x بدون باق) ، فمثلا عوامل العدد ١٢ هي : ١ ، ٢ ، ٣ ، ٤ ، ٦ ، ١٢

ج) اكتب برنامجا يقرأ قيمة عدد صحيح N ويستخدم البرنامجين الفرعيين السابقين لطباعة جميع عوامل العدد N إن كان N زوجيا أو طباعة رسالة مناسبة ما عدا ذلك.

الاختبار النهائي

رقم (٥)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

السؤال الأول (١٠ درجات) :

نفرض أن i , j , k **متغيرات صحيحة ، حيث** i = 8 , j = 10 , k = 11

ونفرض أن Flag **متغير منطقي ، حيث** Flag = True.

أوجد نوع (type) **وقيمة** (value) **كل من التعابير التالية :**

a) i + j div 3 * j - k mod 3 b) (k + 1 > j + i) or (i + 2 = j)
c) (j - i < k) and not Flag d) round (i/j) + trunc (j+k/j)
e) pred (chr(ord('A')+2))

السؤال الثاني (١٠ درجات):

اكتب إجراء PrintPrayer (InPrayer : PRAYERS) **يطبع قيمة الوسيط المدخل** InPrayer (input parameter) **، حيث النوع** PRAYERS **معرف كما يلي :**

```
TYPE
PRAYERS = (FAJR , ZUHR , ASR , MAGHREB , ISHA) ;
```

السؤال الثالث (١٥ درجة):

تتبع تنفيذ قطعة البرنامج التالية وأوجد مخرجاتها :

```
FOR  i := 1  to  8  DO
  BEGIN
      A[i] := i +1 ;
     Write( A[i]:5)
  END;
Writeln;   i :=  2;
WHILE  i  <=  6 DO
   BEGIN
      A[ i ]  :=  A[ i + 2 ];
      A[ i - 1 ] := A[ i +1];
      For  j := 1  to  i   DO  Write( A[ j ]:5);
      Writeln;
      i :=  i + 2
   END;
 FOR  j := 1  to  8  DO  Write( A[ j ]:5);
```

السؤال الرابع (١٥ درجة):

اكتب برنامجا يقوم بإجراء الخطوات التالية :

(i) **يقرأ مجموعة من الأعداد الحقيقية غير السالبة ، ولإنهاء المدخلات نقوم بإدخال عدد سالب.**

(ii) **يطبع أصغر قيمة.**

(iii) **يطبع القيمة المتوسطة.**

السؤال الخامس (١٥ درجة):

تتبع تنفيذ قطعة البرنامج التالية واكتب مخرجاتها.

```
FOR  K := 5 DOWNTO 1 DO
            BEGIN
                  FOR  I := 1 to 5 - K DO
                        Write ( '*' );
                  FOR J := 1 to 2 * K - 1 DO
                        Write ( 'B' );
                  Writeln
            END;
```

السؤال السادس (١٥ درجة):

تتبع تنفيذ البرنامج التالي وأوجد مخرجاته :

```
PROGRAM Q6(OUTPUT);
          VAR
             a, b, c, d : Integer;
           PROCEDURE   P1( a : Integer ; VAR  b : Integer);
             VAR
             c : Integer;
             PROCEDURE P2( x, y : Integer);
               BEGIN
                  c := x + y + a;
                  writeln (' P2: a = ',a: 3,' b = ', b:3,' c = ', c:3, 'd = ',d:3)
             END;
            BEGIN  {P1}
              P2( a ,  2 );
                   a :=  a + c;          b :=  b + c;          d :=  2 * c;
                  writeln (' P1: a = ',a: 3,' b = ',b:3,'c = ',c:3,'d = ', d:3);
             END; {P1}
          BEGIN   {Main}
            a := 4;  b := 3;  c :=10;  d :=20;
```

```
    P1 (   b ,  a );
    writeln ('Main : a = ',a:3,' b = ', b:3, 'c = ', c:3, ' d = ', d:3 );
END.
```

السؤال السابع (٢٠ درجة):

اكتب برنامجا يقوم بتنفيذ الخطوات التالية :

١- **يقرأ قيمة عدد صحيح** N ، **بحيث أن أكبر قيمة مسموح بها هي** 100.

٢- **يقرأ قيم عناصر منظومة** A **مكونة من أعداد صحيحة عددها** N .

٣- **يقوم بتخزين عناصر** A **زوجية القيمة في منظومة** EVARRAY.

٤- **يقوم بتخزين عناصر** A **فردية القيمة في منظومة** ODARRAY.

٥- **يطبع عناصر المنظومة** EVARRAY.

٦- **يطبع عناصر المنظومة** ODARRAY.

مثال : إذا كانت القيمة المدخلة للعدد الصحيح N **هي** 10 ، **وكانت القيم المدخلة لعناصر المنظومة** A **هي :**

30 50 23 100 45 47 13 20 17 1

فإن البرنامج يطبع ما يلي :

```
Even array is :   30   50   100   20
Odd array is  :   23   45   47   13   17   1
```

الاختبار النهائي

رقم (٦)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

السؤال الأول (١٠ درجات) :

اكتب برنامجا يقرأ الرقم التعريفي (identification number) **لأحد الموظفين** (an employee) **وراتبه الأساسي** (basic salary) **، وحالته الاجتماعية (الرمز 'M' للمتزوج أو الرمز 'S' للأعزب) ، وعدد أولاده (صفرا إن لم يكن له أولاد). ويطبع البرنامج الرقم التعريفي للموظف ، وراتبه الأساسي وراتبه الاجمالي** (total salary). **والراتب الإجمالي هو مجموع راتبه الأساسي وعلاوته الاجتماعية. والعلاوة الاجتماعية** (social allowance) **تساوي ١٠٠ دينار للموظف و٥٠ دينار للزوجة ، ١٥,٥ دينارا لكل واحد من الأولاد.**

السؤال الثاني (١٥ درجة) :

ما هي مخرجات كل من البرنامجين التاليين ؟

(a)
```
Program Q2A (Output) ;
var i : integer
Begin
    Write ('He whose') ;
    For i := 0 to 6 do
        case i mod 4 of
            3 : write (' his') ;
            2 : writeln (' and his Messenger,') ;
            1 : write (' for Allah') ;
            0 : write (' migration was')
        end ;
    writeln ('Agreed upon')
End.
```

(b)
```
Program Q2B (output) ;
Type Colour = (Red , Blue , Yellow , Green , Black) ;
var  C1 : Colour ;
     I , J : Integer ;
begin
     C1 : Blue ;
     While C1 <> Black do
          begin
```

```
                J := ord (C1) ;
                I := J mod (ord (Black)) ;
                C1 := succ (C1) ;
                Writeln (′The values of I and J are ′ , I:4 , J:4)
            end
    end.
```

السؤال الثالث (١٥ درجة) :

تتبع تنفيذ البرنامج التالي وأوجد مخرجاته بفرض أن مدخلاته هي 3 5 7

.2

```
                                Program Q3 (Input , Output) ;
                                    var X , Y , Z , G : nteger ;
                                Procedure Outer (var X , Y : Integer) ;
                                        var Z : Integer ;
                                    Procedure Inner (var X : Integer) ;
                                            Begin
                                                X := 6 ;
                                            G := X - Y ;
Writeln ('In Inner: G=',G,'X=',X,'Y=',Y,'and
                                                                Z=',Z)
                                                    end ;
                                                        Begin
                                                    G := G + 2 ;
                                                        X := 5 ;
                                                    Y := X + 1 ;
                                                        Z := 4 ;
                                                    Inner (Z) ;
    Writeln ('In Quter : G=',G,'X=',X,'Y=',Y,'and Z=',Z)
                                                            end ;
                                                                Begin
                                                    Writeln ('Enter 4 values :')
                                                    Readln (X , Y , Z , G) ;
                                    Writeln ('G=',G,'X=',X, 'Y=', Y,'and Z=',Z) ;
                                                        Outer (Y , X) ;
                                    Writeln ('G=',G,'X=',X,'Y=',Y,'and Z=',Z)
                                                                end.
```

السؤال الرابع (١٥ درجة) :

أوجد مخرجات البرنامج التالي

```
program Q4 (input , output) ;
```

```
Type
Arr = array [1..8] of Integer ;
var
x : Arr ;
j : Integer ;
procedure Arrange (b , e : Integer : var a : Arr) ;
var
i , temp : Integer ;
begin
i := b ;
while i <= e do
begin
temp := a[i] ;
a[i] := a[i + 2] ;
a[i + 2] := temp ;
i := i + 2
end
end ;
begin
for j := 1 to 8 do
if j mod 2 = 0 then
x[j] := j
else
x[j] := j - 2 ;
for j := 1 to 8 do
writeln (x[j]) ;
Arrange (2 , 6 , x) ;
for j := 1 to 8 do
writeln (x[j])
end.
```

السؤال الخامس (١٥ درجة) :

اكتب دالة توجد ترتيب / مؤشر (index) أصغر عنصر في المدى الواقع بين العنصر A[L] والعنصر A[U] ، وذلك في منظومة A مكونة من ١٠٠ عدد صحيح. ثم اكتب برنامجا يقرأ عناصر منظومة A مكونة من ١٠٠ عدد صحيح ، ويستدعي الدالة السابقة حيث يعطيها : المنظومة A ، والمدى المطلوب (الحد الأسفل L والحد الأعلى U). [ملاحظة : إذا كانت U = 100 ، L = 1 فإن المدى المطلوب هو

المنظومة A كلها]. ويطبع البرنامج : المدى المطلوب ، وقيمة أصغر عنصر في هذا المدى ، وترتيب هذا العنصر.

السؤال السادس (١٥ درجة) :

اكتب برنامجا يقرأ متتابعة sequence من الأعداد الحقيقية غير السالبة ، ويقوم بتخزينها في منظومة. ويتم إنهاء المتتابعة إما بإدخال قيمة سالبة أو إذا بلغ عدد عناصرها الحد الأقصى لسعة المنظومة (array size) والذي يساوي ١٠٠ عنصر. وسنطلق على الفترة [70.5 , 0.0] " الفترة الجيدة". وكي نضمن وقوع جميع قيم عناصر المنظومة في مدى هذه الفترة ، فإننا نستبدل بأي قيمة تقع خارج هذا المدى القيمة المتوسطة average لعناصر المنظومة ، إذا كانت هذه القيمة المتوسطة واقعة في هذا المدى. أما إذا وقعت هذه القيمة المتوسطة خارج هذا المدى فإننا نستبدل بهذه العناصر الخارجة عن المدى أصفارا.

يطلب من البرنامج أن يطبع عدد القيم التي قرأها ، وعدد هذه القيم التي وقعت في مدى الفترة الجيدة المذكورة ، وترتيب / مؤشرات Indices العناصر التي بدلناها.

السؤال السابع (١٥ درجة) :

اكتب برنامجا يقرأ أعدادا صحيحة متتابعة يتكون كل منها من رقم واحد (consecutive single digit integers) ، وتنتهي هذه الأعداد بإدخال 1- . ثم يحسب البرنامج ويطبع قيمة العدد الصحيح المكافئ لهذه الأرقام المتتابعة التي قرأها ، كما يوضح ذلك المثال التالي :

إذا فرضنا أن المدخلات هي :

3
4
6
9
-1

فإن القيمة الناتجة التي يحسبها البرنامج ويطبعها هي 3469

إرشاد :

3469 = 3 * 1000 + 4 * 100 + 6 * 10 + 9

= (((((3 * 10) + 4) * 10) + 6) * 10) + 9

الاختبار النهائي

رقم (٧)

مجموع الدرجات : ١٠٠ درجة الزمن : ساعتان

السؤال الأول (٢٠ درجة):

(أ) (١٠ درجات) ما هي مخرجات البرنامج التالي :

```
Program Caliphate (output) ;
type
Caliphs = (AbuBakr, Omar, Othman, Ali) ;
var
    Caliphs ;      C , C1 :
    integer ;      :   i , j
begin
C := AbuBakr ;  C1 := Ali ;
    j := ord (C1) + 1 ;
    for i := 1 to j do
        begin
    write (′ The′ , i : 2) ;
        case i of
1 : write (′st′ : 3) ;
2 : write (′nd′ : 3) ;
3 : write (′rd′ : 3) ;
4,5 : write (′th′ : 3) ;
        end ;
    write (′ Caliph is : ′) ;
        case C of
AbuBakr  :  writeln (′AbuBakr Assiddeek′);
Omar     :  writeln (′Omar Ibn El Khattab′);
Othman   :  writeln (′Othman Ibn Affan′);
Ali      :  writeln (′Ali Ibn Abi Taleb′);
        end ;
    C := succ (C)
        end
end.
```

(ب) (١٠ درجات) نفرض أننا قد أعطينا الإعلانات التالية والإجراء encode **لصياغة الشفرات.**

```
const     max = 10;
type      list = array [1..max] of char;
var       A, B : list;
procedure encode (var word : list ; n : integer);
{This procedure encodes an array of characters}
var   i : integer;
begin
    for i := 1 to n do
      begin
word [i] := succ (succ(word [i]));
        write (word [i] : 1)
      end ;
    writeln
end ;
```

ونفرض أيضا أن القيم التي قرئت في منظومة الرموز A (character array) **هي** Badr **، وأن تلك التي قرئت في منظومة الرموز** B **هي** Lkjcf.

i) **ما نتيجة تنفيذ الاستدعاء** encode (A,4) ؟

ii) **اكتب إجراء** decode (var word : list ; n : integer) **لفك شفرة منظومة رموز مستقبلة** word **طولها** n **، أي أن وظيفة هذا الإجراء هي عكس وظيفة الإجراء** encode.

iii) **ما نتيجة تنفيذ الاستدعاء** decode (B,5) ؟

السؤال الثاني (٢٠ درجة) :

تتبع كلا من البرامج التالية ، واكتب نتيجة تنفيذ عبارات البرنامج ومخرجاته.

(أ) **(٥ درجات)**

```
Program Rev (output) ;
Var
    n : integer ;
Begin
    n := 8453 ;
    repeat
write (n mod 10 : 1) ;
    n := n div 10
    until   n = 0
End .
```

(ب) (٧ درجات)

```
Program Test (input , output) ;
                                const
                                    n = 8 ;
                                  Var
                        i,sum,value : integer ;
                              flag : boolean ;
                                 Begin
                sum := 0 ;   i := 1 ;   flag := false ;
                   while (i <= n)  and  not flag  do
                                      begin
                              read (value) ;
                            if  value > 0  then
                        sum := sum + value
                           else if  value = 0  then
                              flag := true ;
                                    i := i + 1
                                         end ;
                              writeln (sum , value)
                                              End.
9     8     5     0     -4     7     -3     6     5     input :
```

(ج) (٨ درجات)

```
                          Program Trace (input , output) ;
                  Type  Vector = Array [1..10] of integer ;
                                  Var     List : Vector ;
                                Size : integer ;     i ,
Function Result (A : Vector ; L : integer) : integer ;
                                Var   i , index : integer ;
                                                 Begin
                                                index := 1 ;
                                             For i := 2  to  L  Do
                                    if A[i] > A[index] then
                                          index := i ;
                                                Result := index
                                                             End ;
                                                 Begin
                                                    Read (Size) ;
                                              For i := 1 to Size Do
                                              Read (List[i]) ;
```

```
        Writeln (Result (List , Size))
    End
5   4   9   1   3   5   input :
```

السؤال الثالث (١٥ درجة) :

تتبع تنفيذ البرنامج التالي وأوجد مخرجاته.

```
Program Trace ;
    var x , y , z , sum : integer ;
Procedure P1 (n1 , n2 , n3 : integer ; var s : integer) ;
                                                    begin
                                                n1 := x + 1 ;
                                                n2 := y + 2 ;
                                                n3 := z + 3 ;
                                              s  := n1 + n2 + n3 ;
                                    writeln ('P1 :' , n1:4 , n2:4 , n3:4 , s:4) ;
                                                    end ;
procedure P2 (var a , b : integer ; c: integer) ;
                                                    begin
                                                    a := a*2 ;
                                                b := b mod 2 ;
                                                c := c + 10 ;
                                              sum := a + b + c ;
                                    writeln ('P2 :' , a:4 , b:4 , c:4 , sum:4) ;
                                                    end ;
                                                Begin (Trace)
                                            x := 2 ; y := 4 ; z := 10 ;
                                                P1 (x , y , z , sum) ;
                                    writeln ('Main :' , x:4 , y:4 , z:4 , sum:4) ;
                                                P2 (x , y , z) ;
                                    writeln ('Main :' , x:4 , y:4 , z:4 , sum:4) ;
                                                End. {Trace}
```

السؤال الرابع (١٥ درجة) :

في أحد النظم المصرفية يتم إدخال البيانات التالية لكل شخص (عميل (customer) **: رقم الحساب** A (account number) **، الرصيد الحالي** (current balance) B **، قيمة المعاملة** T (transaction amount) **، الرمز الشفري للمعاملة** code (transaction code) **(حيث يستخدم الرمز** D **للإيداع** deposit **والرمز** W **للسحب** withdrawal).

(أ) اكتب إجراء BALANCE يستقبل الرمز الشفري للمعاملة ، وقيمة المعاملة ، والرصيد الحالي ، ويعيد الرصيد الجديد (new balance) تبعا للعلاقات التالية

- في حالة الإيداع :

الرصيد الجديد = الرصيد الحالي + قيمة المعاملة

new balance = current balance + transaction amount

- في حالة السحب :

إذا كان الرصيد الحالي أقل من قيمة المعاملة فإنه يطبع رسالة تفيد عدم إمكانية السحب ، مثل "Sorry, no withdrawal" ، ويكون :

الرصيد الجديد = الرصيد الحالي

new balance = current balance

وما عدا ذلك ، فإن :

الرصيد الجديد = الرصيد الحالي – قيمة المعاملة

new balance = current balance - transaction amount

(ب) اكتب برنامجا يقوم بتحديث (updating) أرصدة عملاء المصرف كما يلي :

- يقرأ البيانات السابقة (A, B, T, code) لقائمة من الأشخاص (العملاء).
- يستدعي الإجراء BALANCE .
- يطبع رقم الحساب والرصيد الجديد لكل واحد من هؤلاء الأشخاص.
- يتوقف البرنامج حين يتم إدخال قيمة سالبة لرقم الحساب.

السؤال الخامس (١٥ درجة) :

(أ) اكتب دالة freq (a,n) تستقبل منظومة أعداد حقيقية a عدد عناصرها n ، وتعطي عدد مرات ظهور أكبر قيمة في المنظومة a .

(ب) اكتب برنامجا رئيسيا يقرأ عناصر منظومة X تمثل درجات خمسين طالبا 50) (secores في أحد الفصول الدراسية ، ويستخدم الدالة freq لإيجاد عدد الطلاب الذين حصلوا على أعلى درجة في الفصل.

السؤال السادس (١٥ درجة) :

اكتب برنامجا لقراءة الأرقام التعريفية ID's **(الرموز الشفرية للبلدان وهي أعداد صحيحة)** (integer identification numbers / country codes) **وتعداد السكان** (populations) **(بالمليون) للبلاد العربية وعددها** 21 **بلدا ، وتخزين هذه المعلومات في منظومتين متوازيتين** (2 parallel arrays) : ID **(للأرقام التعريفية) و** Pop **(لتعداد السكان) ، ثم يقوم البرنامج بما يلي :**

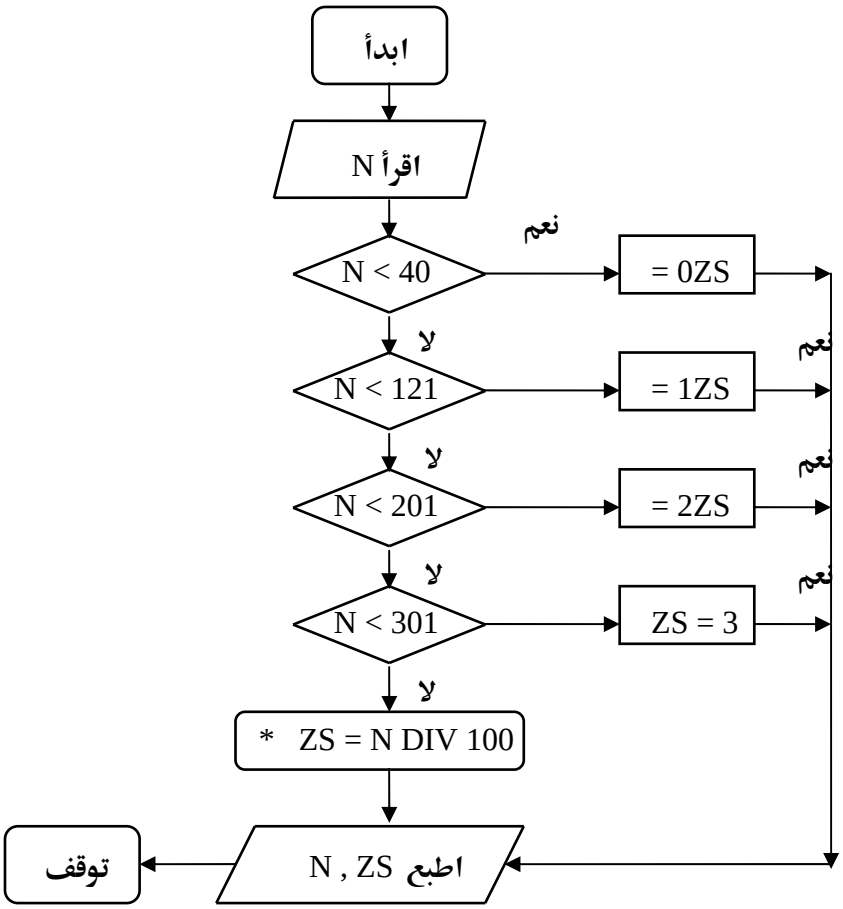
i) **إيجاد المجموع الكلي** TOT **لتعداد السكان.**

ii) **إيجاد أعلى تعداد سكان** MAX **والرقم التعريفي للبلد المقابل.**

iii) **ترتيب منظومة تعداد السكان** Pop **ترتيبا تصاعديا.**

أجوبة تمرينات الفصل الأول

١-١

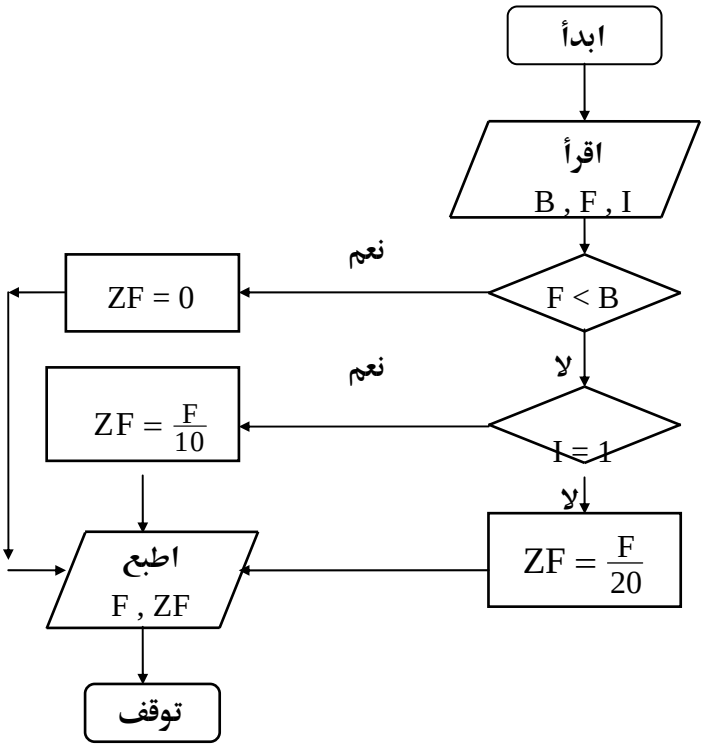


*** ملاحظة : يمكن التعبير رياضيا عن العبارة : ZS تساوي في كل مائة شاة بالعلاقة :**

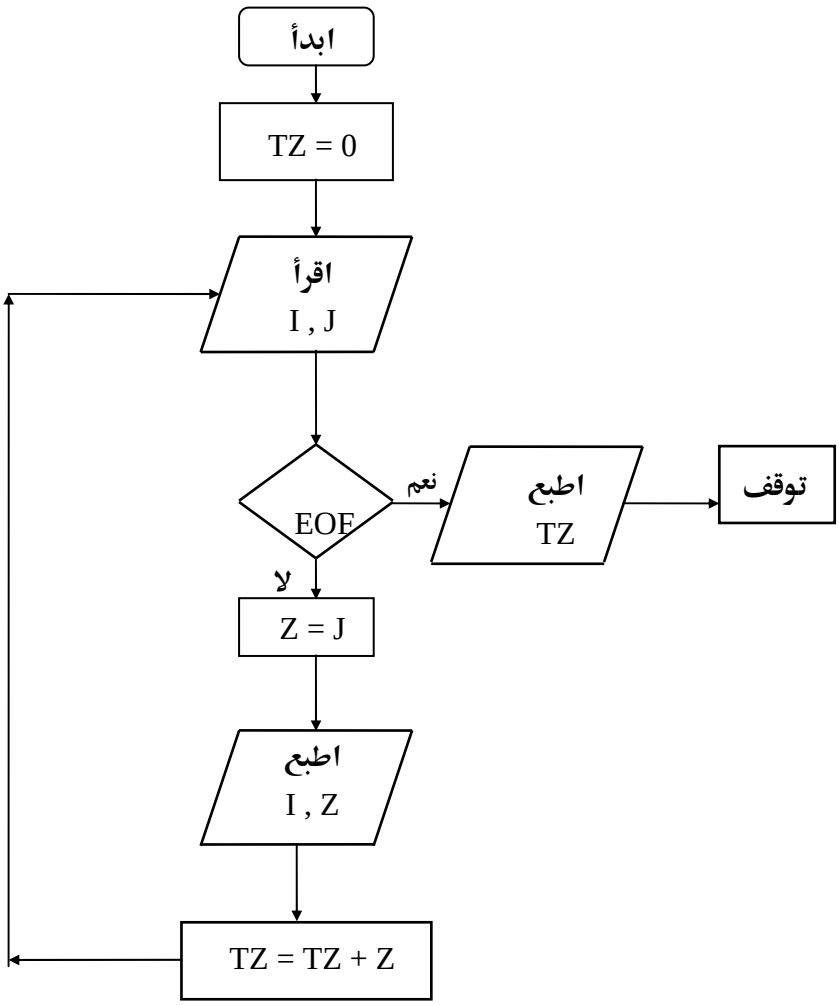
ZS = N DIV 100 والتي تعطي العدد الصحيح فقط في خارج قسمة $\frac{N}{100}$ مع حذف الكسور العشرية ، وذلك لأن هذه العلاقة تعطي النتيجة التالية :

N	٣٠١-٣٩٩	٤٠٠-٤٩٩	٥٠٠-٥٩٩	٠٠٠
ZS	٣	٤	٥	٠٠٠

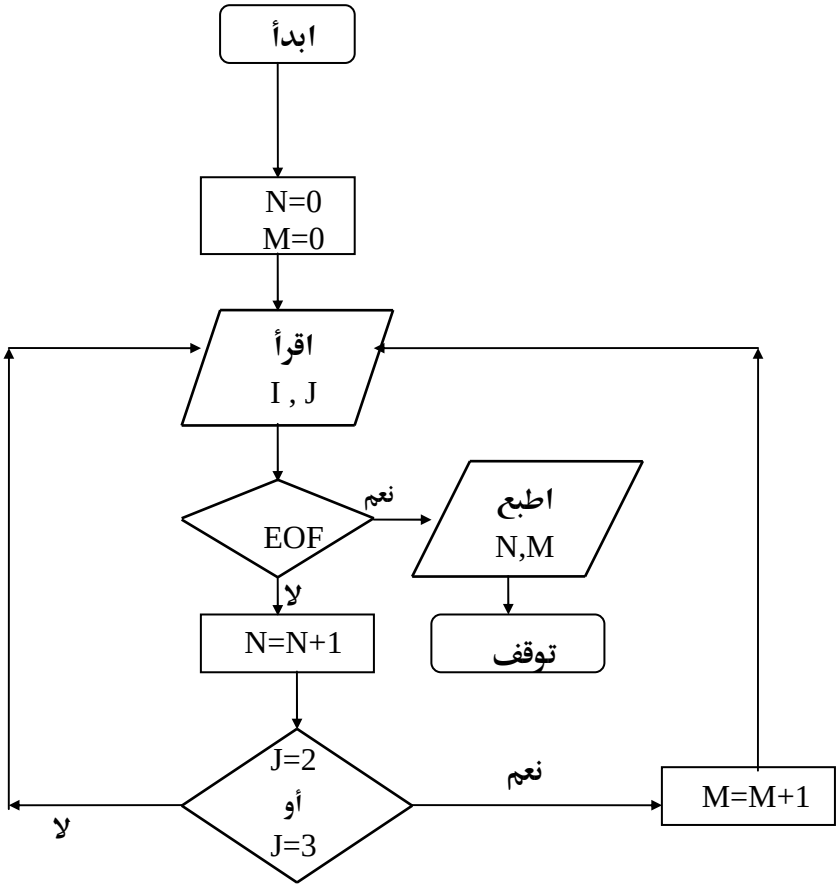
٢-١



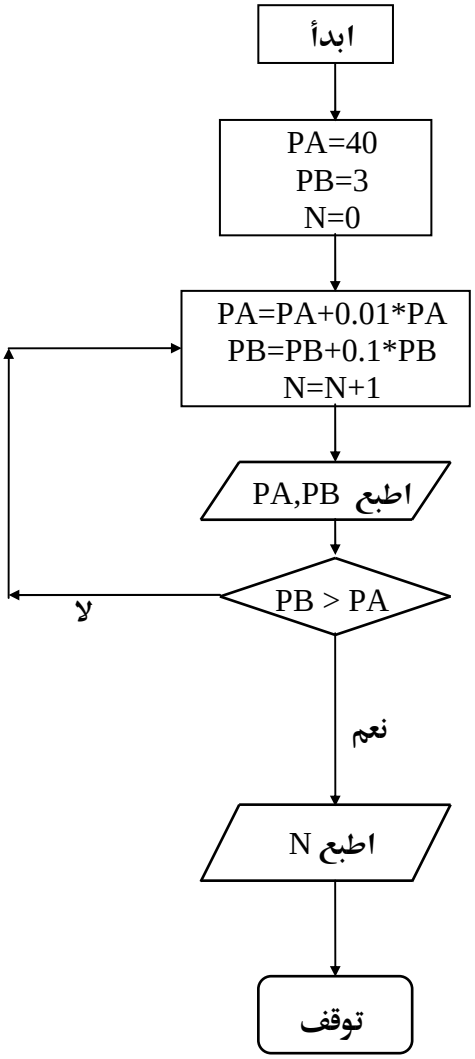
٣-١



٤-١



١-٥

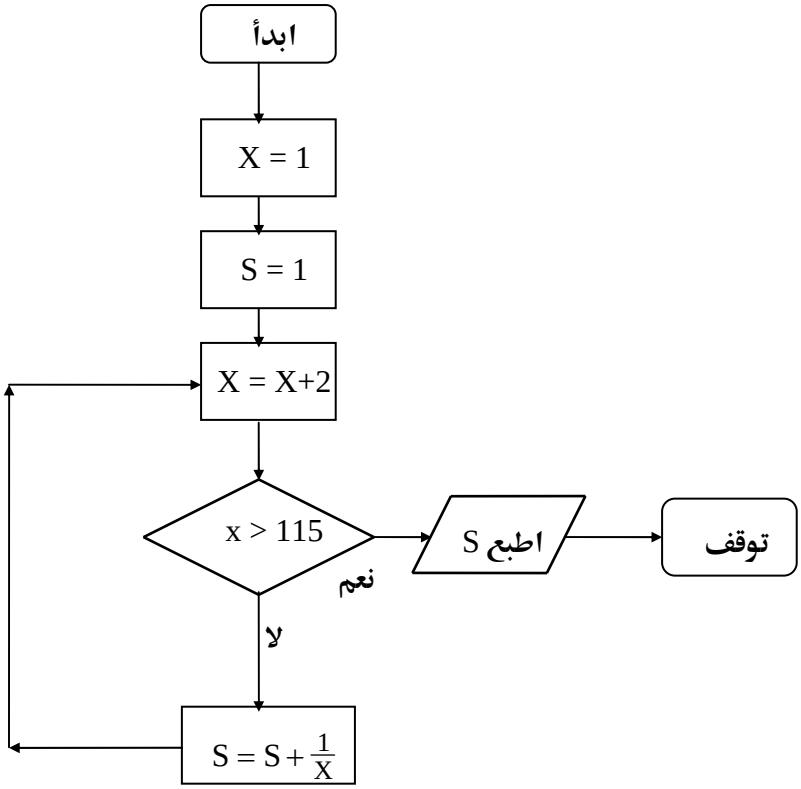


ملاحظة : يمكن أيضا طباعة عدد السنوات N – كل سنة – مع PA , PB.

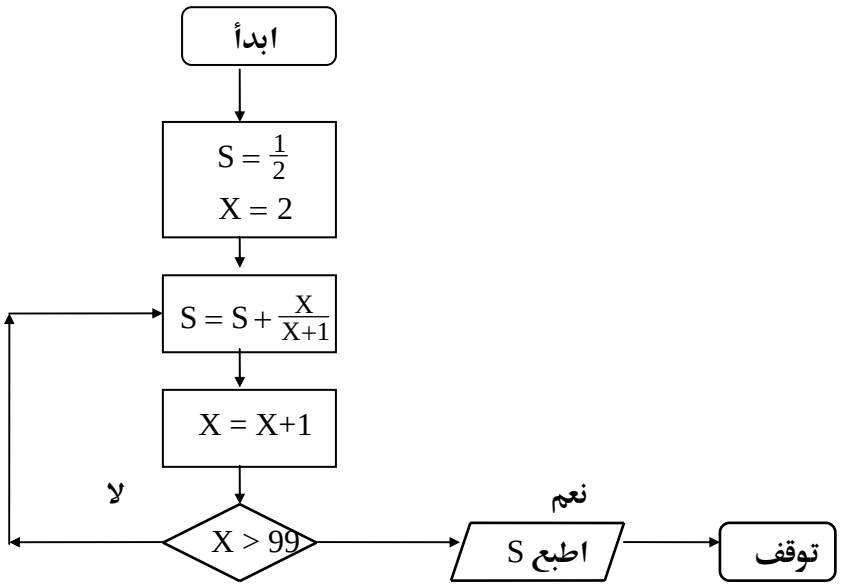
٦-١

iv) 5 iii) 10 ii) 8 i) 5

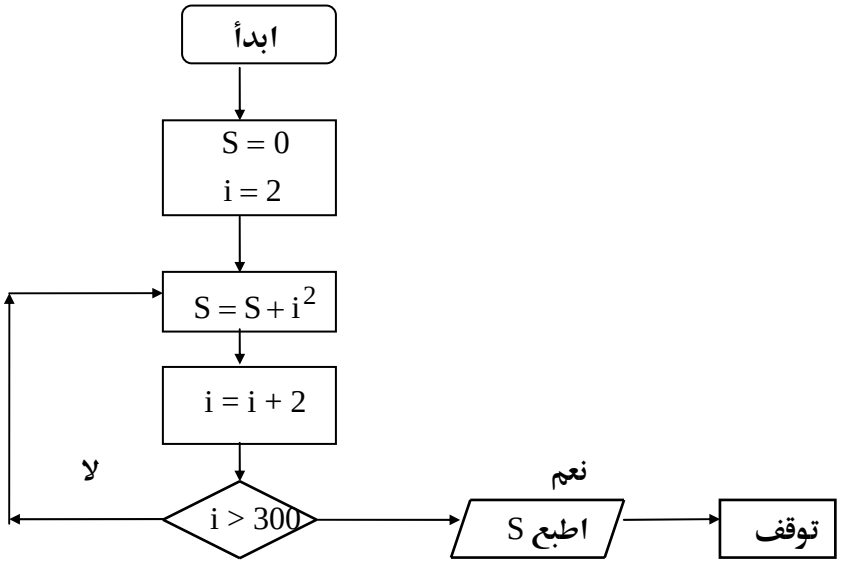
٧-١ i)



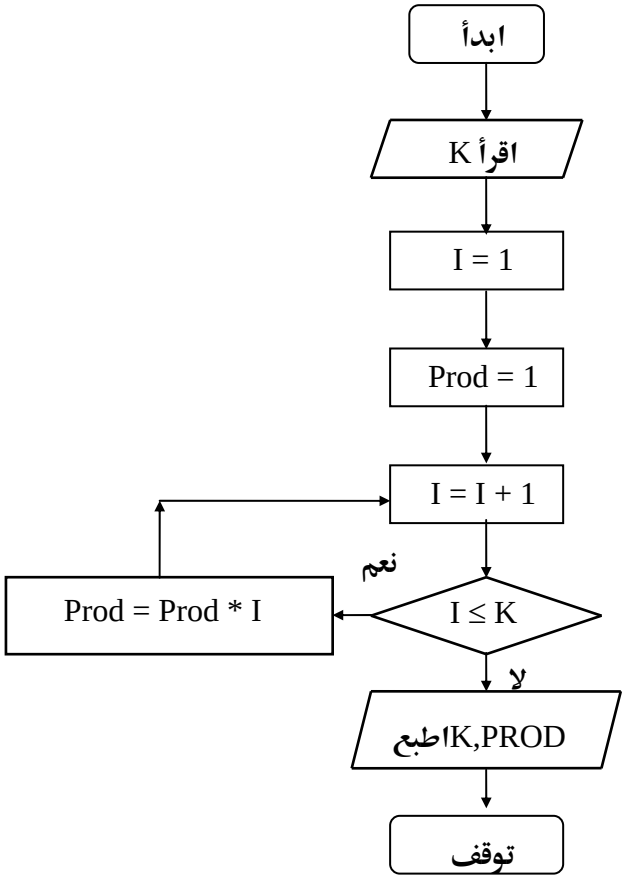
(ii ٧–١


ابدأ
$S = \frac{1}{2}$
X = 2
$S = S + \frac{X}{X+1}$
X = X+1
لا
X > 99
نعم
اطبع S
توقف

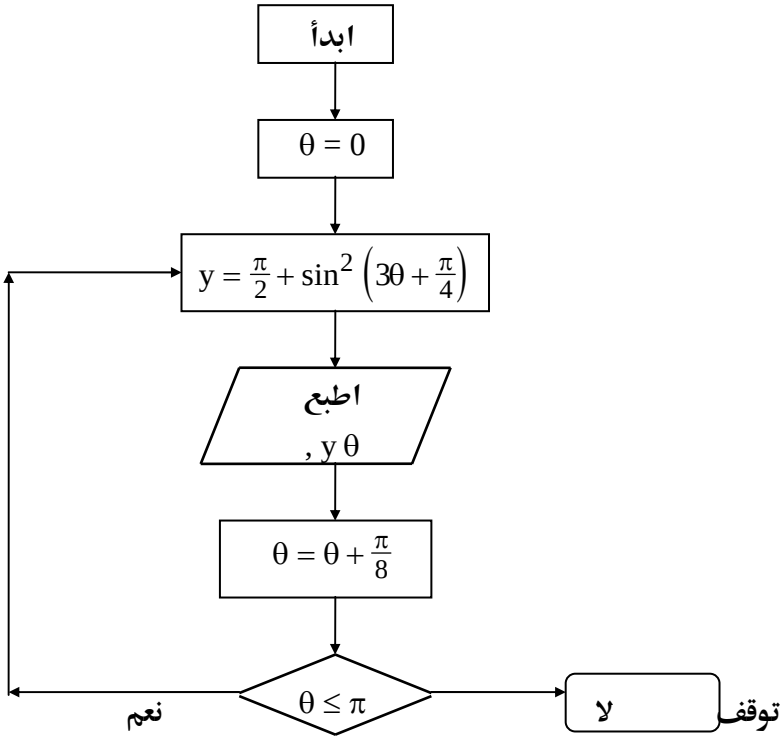
(iii ٧–١


ابدأ
S = 0
i = 2
$S = S + i^2$
i = i + 2
لا
i > 300
نعم
اطبع S
توقف

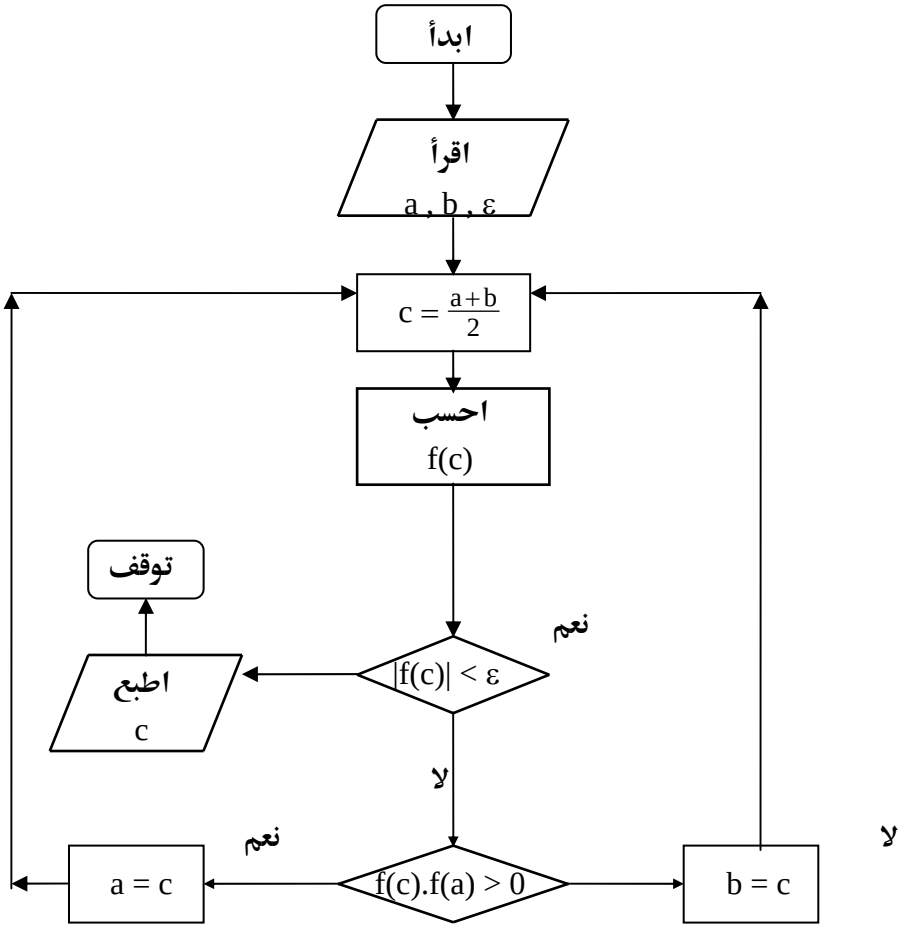
٨-١

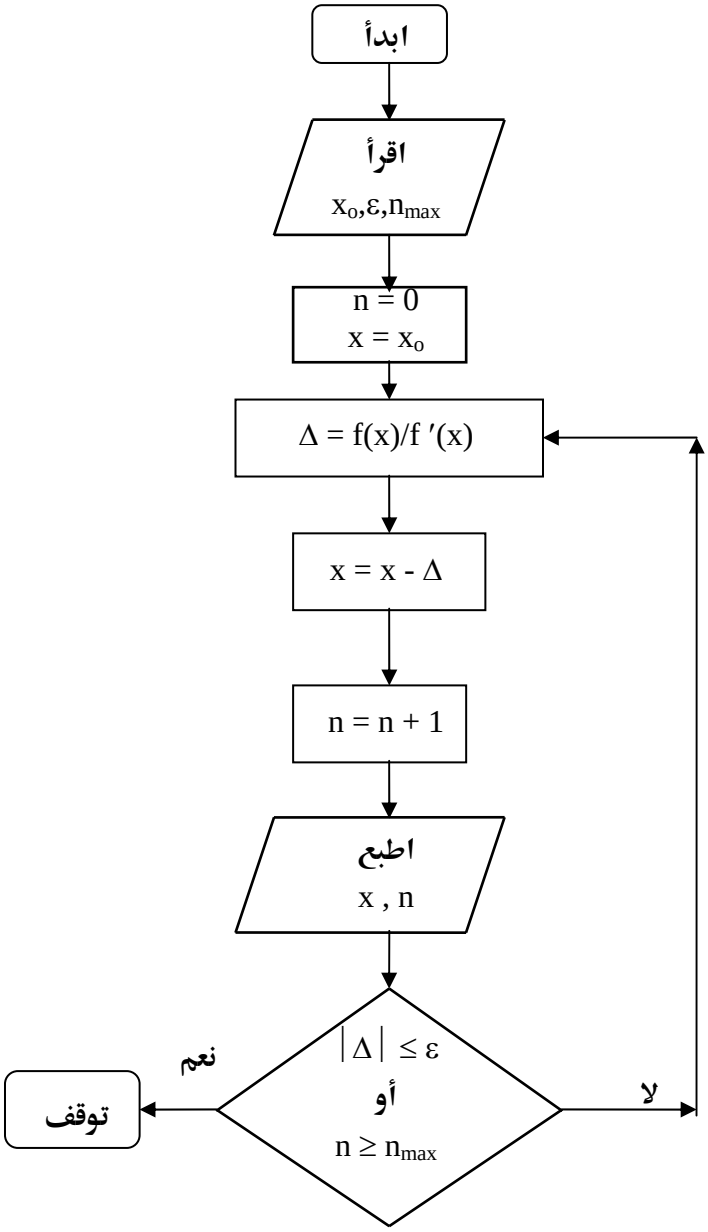


٩-١

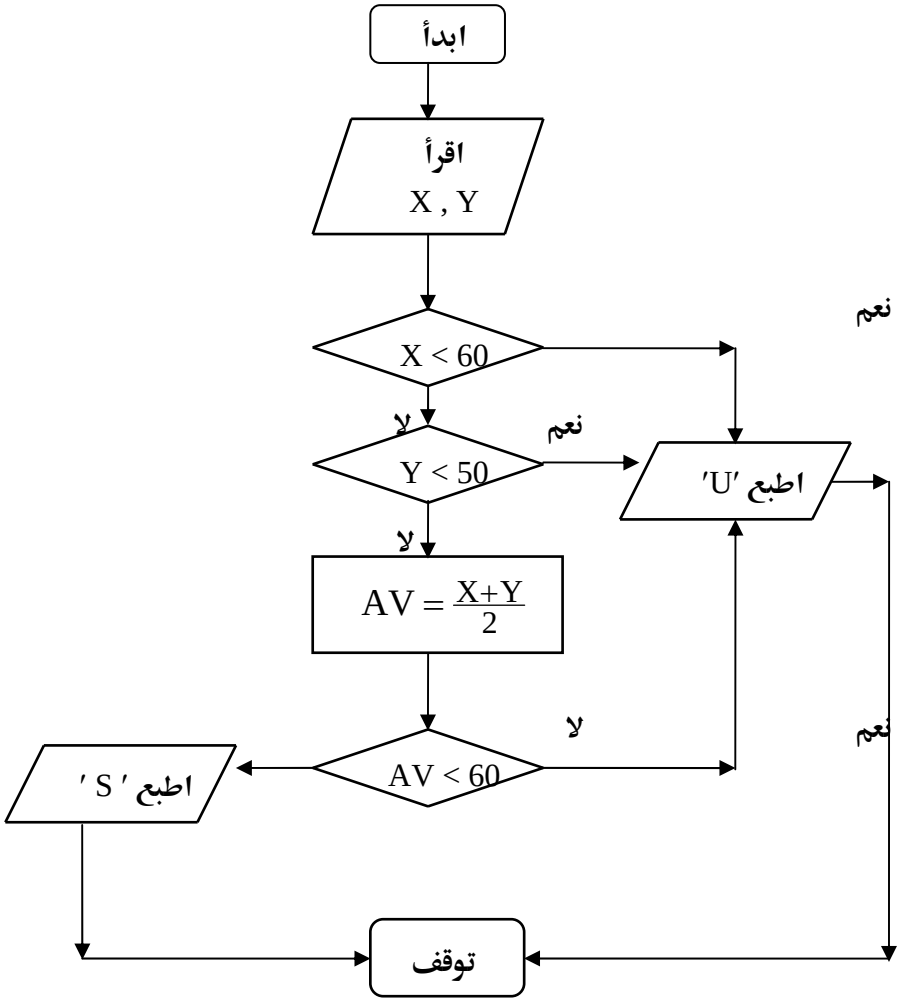

ابدأ
θ = 0
$y = \frac{\pi}{2} + \sin^2\left(3\theta + \frac{\pi}{4}\right)$
اطبع
, y θ
$\theta = \theta + \frac{\pi}{8}$
$\theta \le \pi$
نعم
لا
توقف

١٠-١

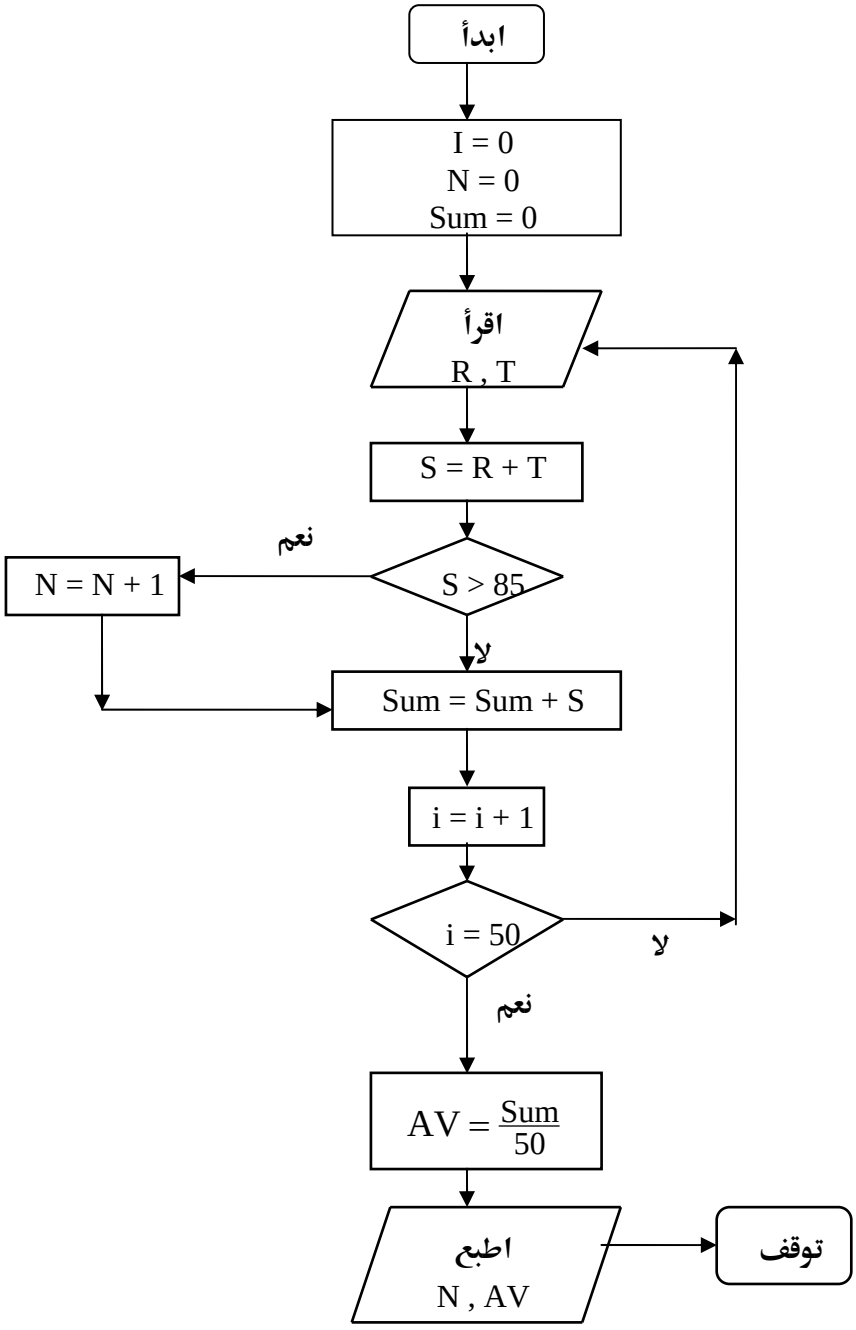

ابدأ
اقرأ
a , b , ε
c = (a+b)/2
احسب
f(c)
توقف
نعم
اطبع
c
|f(c)| < ε
لا
نعم
لا
a = c
f(c).f(a) > 0
b = c


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
$|\Delta| \le \varepsilon$
أو
$n \ge n_{max}$
نعم
لا
توقف

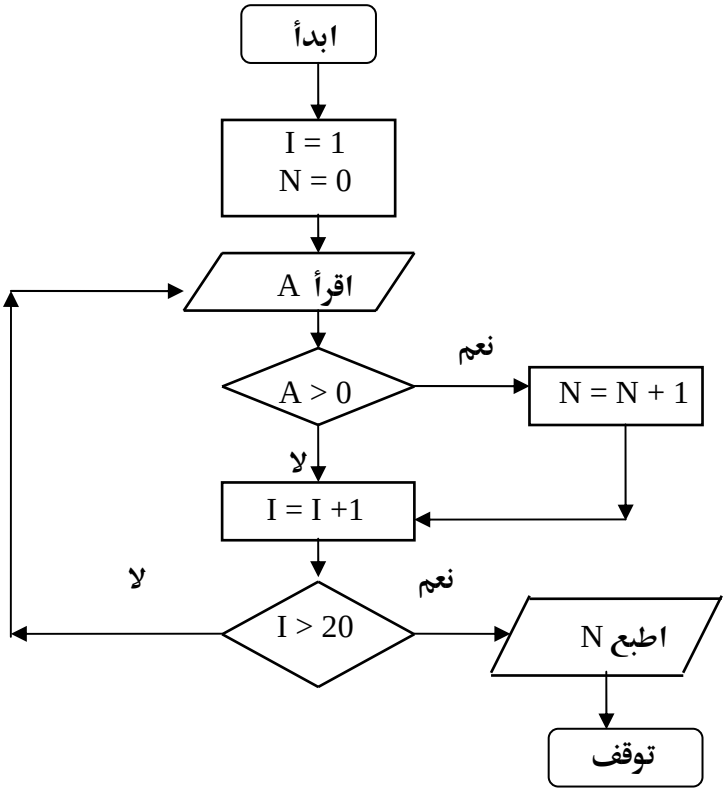
١٢-١


ابدأ
اقرأ
X , Y
X < 60
نعم
لا
Y < 50
نعم
اطبع 'U'
لا
AV = X+Y / 2
لا
AV < 60
نعم
اطبع ' S '
توقف

١٣-١

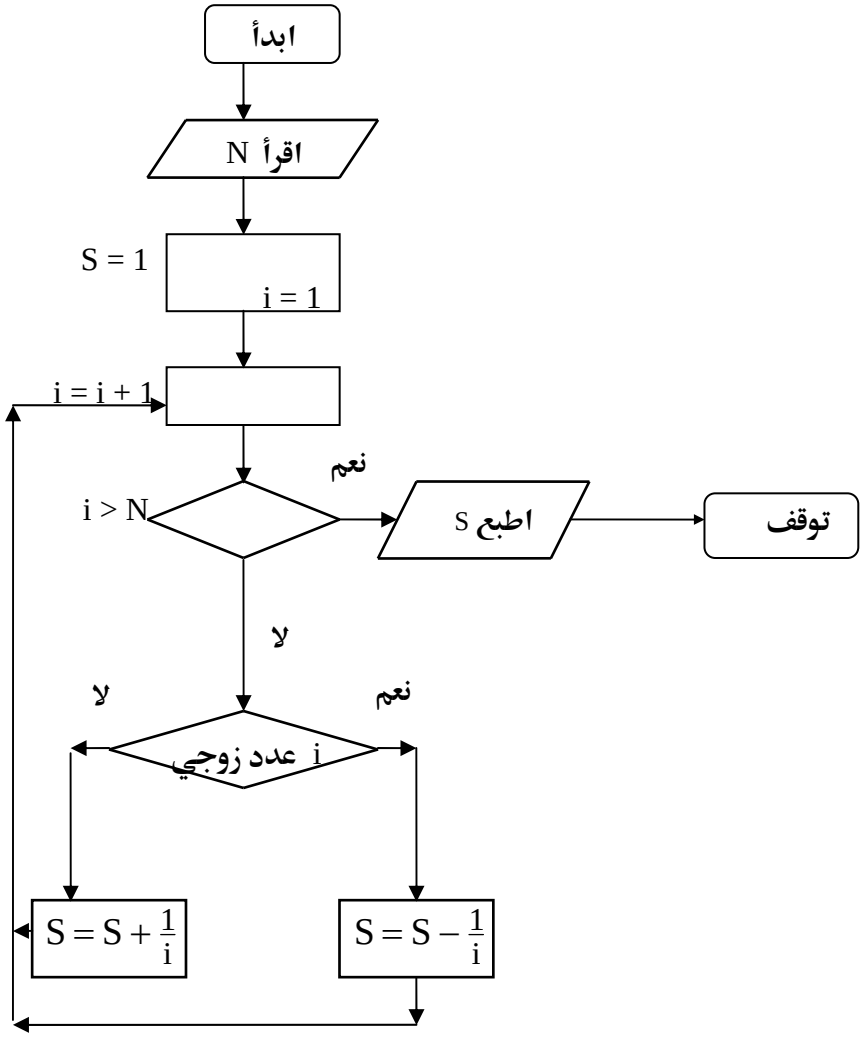

ابدأ
I = 0
N = 0
Sum = 0
اقرأ
R , T
S = R + T
نعم
N = N + 1
S > 85
لا
Sum = Sum + S
i = i + 1
i = 50
لا
نعم
AV = Sum / 50
اطبع
N , AV
توقف

١٤-١

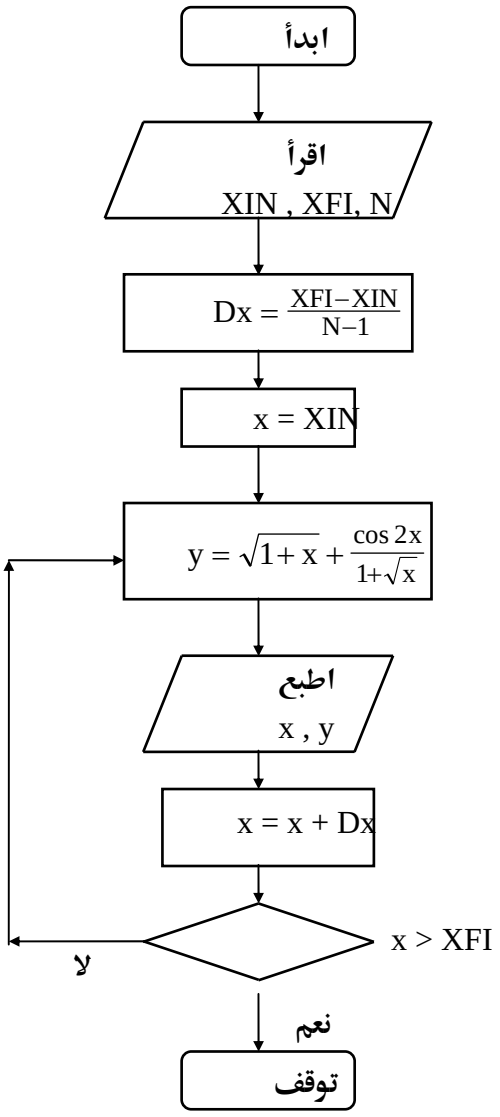


الخوارزمية تحسب عدد العناصر الموجبة في مجموعة مكونة من عشرين عددا.

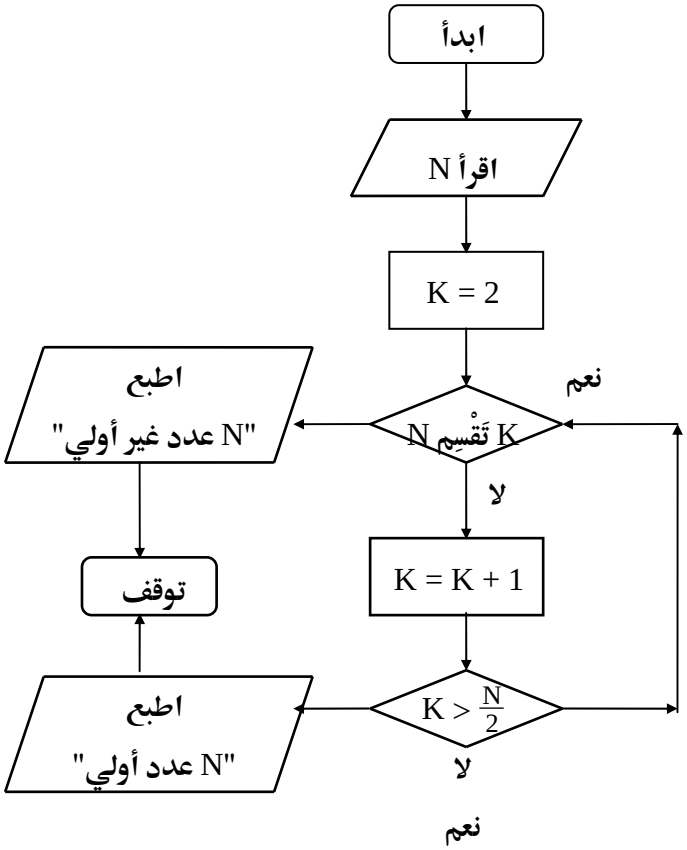
١٥-١


ابدأ
اقرأ N
S = 1
i = 1
i = i + 1
نعم
i > N
اطبع S
توقف
لا
لا
نعم
i عدد زوجي
$S = S + \frac{1}{i}$
$S = S - \frac{1}{i}$

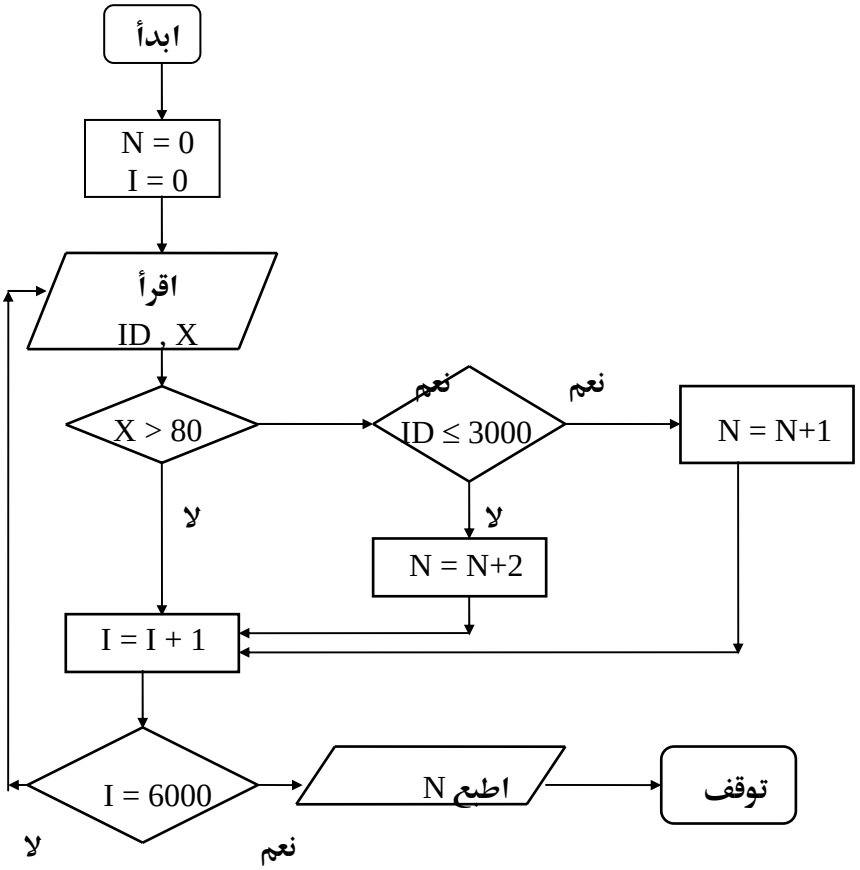
١٦-١


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

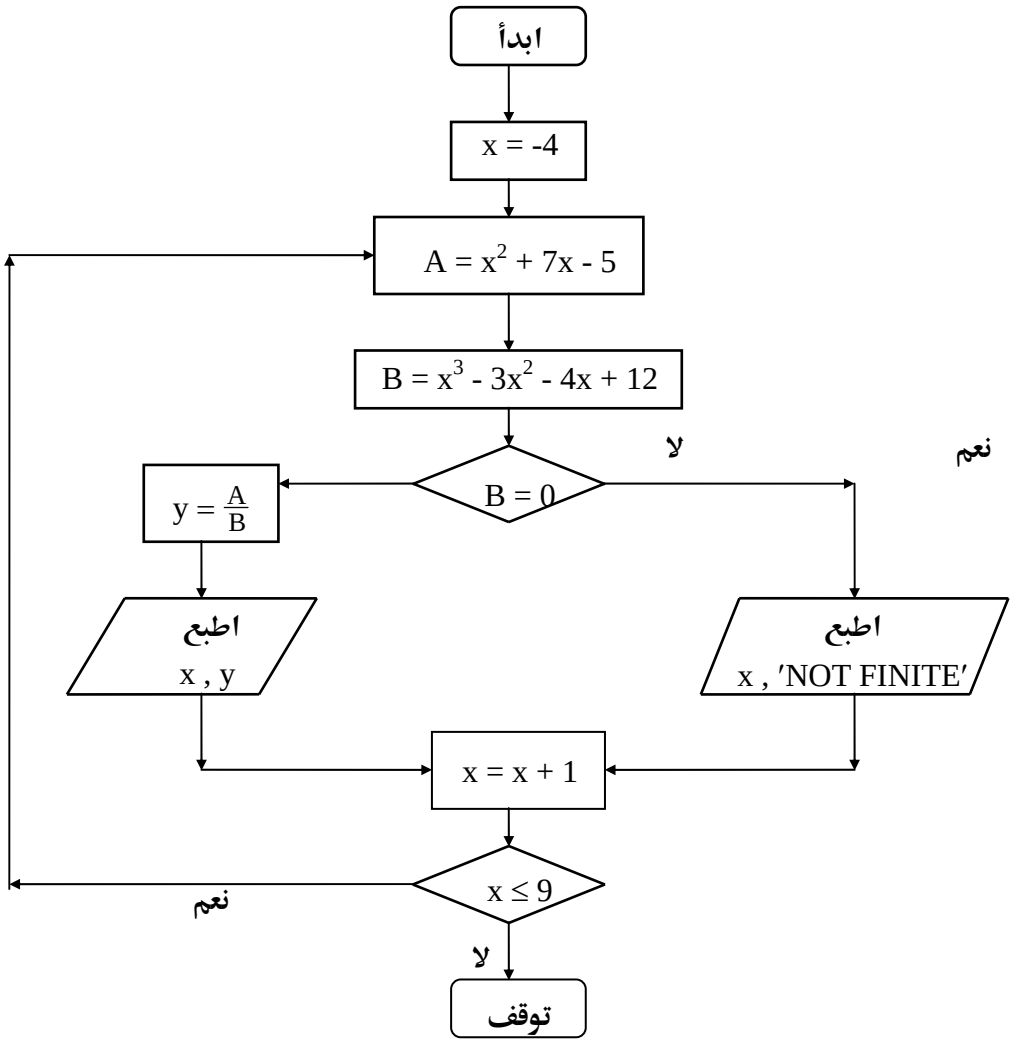
١٧-١


ابدأ
اقرأ N
K = 2
K تَقْسِم N
نعم
اطبع
"N عدد غير أولي"
لا
K = K + 1
توقف
K > N/2
اطبع
"N عدد أولي"
لا
نعم

١٨-١


ابدأ
N = 0
I = 0
اقرأ
ID , X
X > 80
نعم
ID ≤ 3000
نعم
N = N+1
لا
N = N+2
لا
I = I + 1
I = 6000
اطبع N
توقف
لا
نعم

١٩-١


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

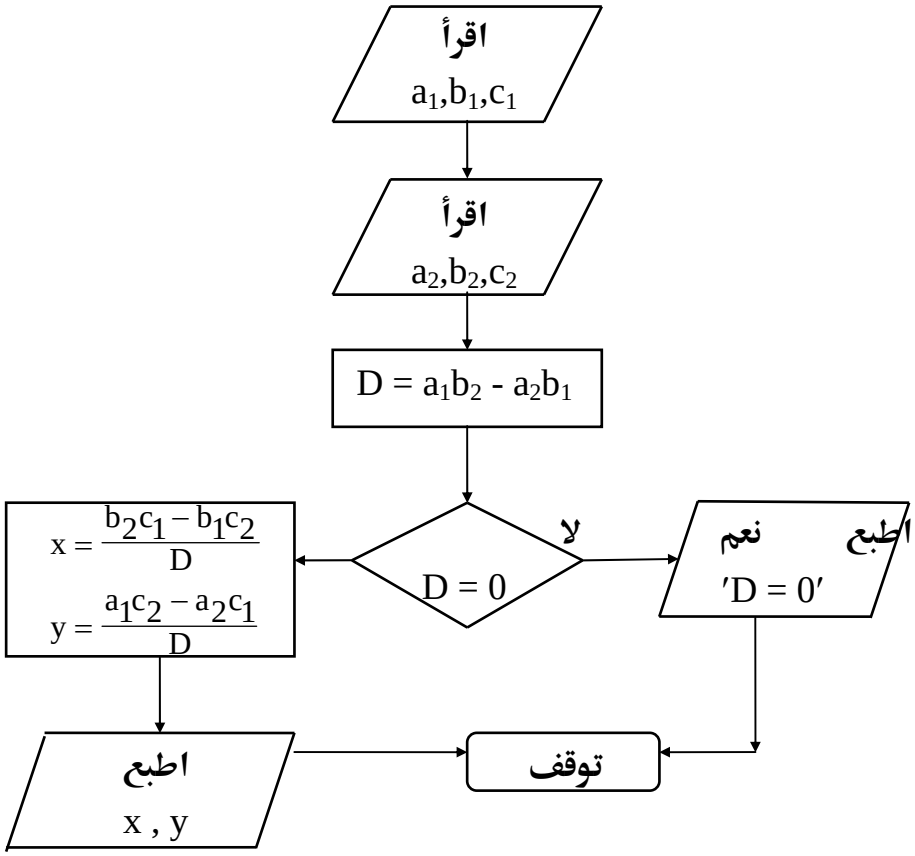
٥- S = S + 1 + n A

٦- إذا كانت $n \leq 48$ اذهب إلى ٤

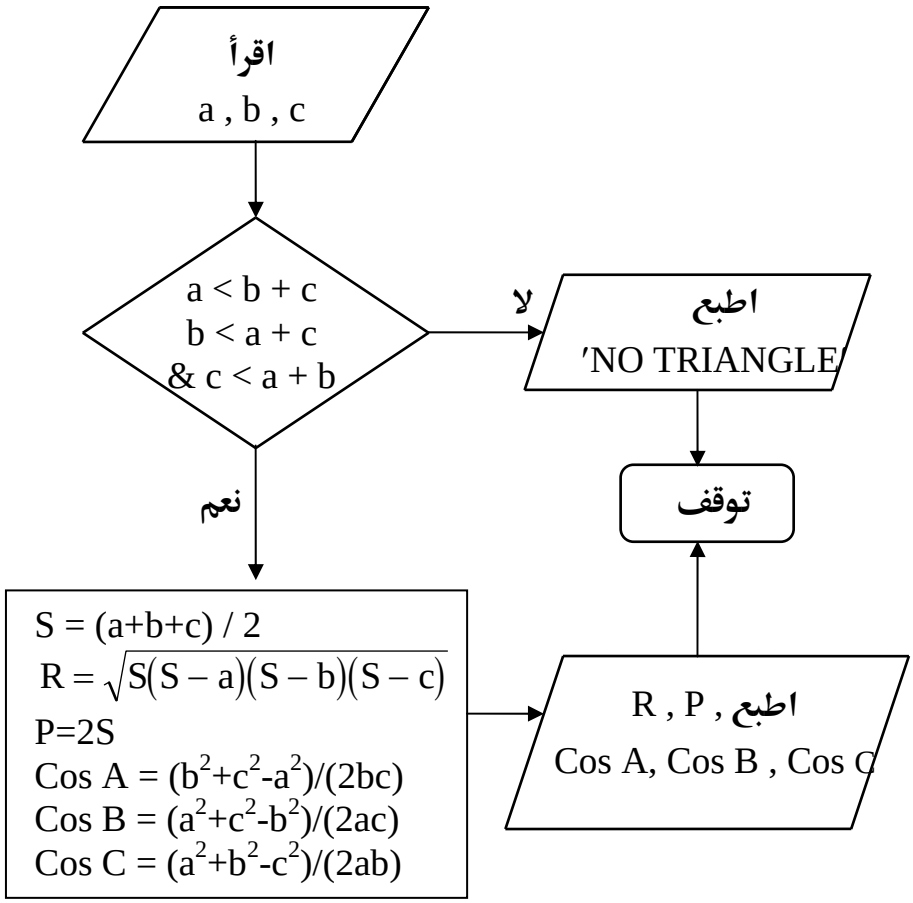
٧- اطبع S

٨- توقف

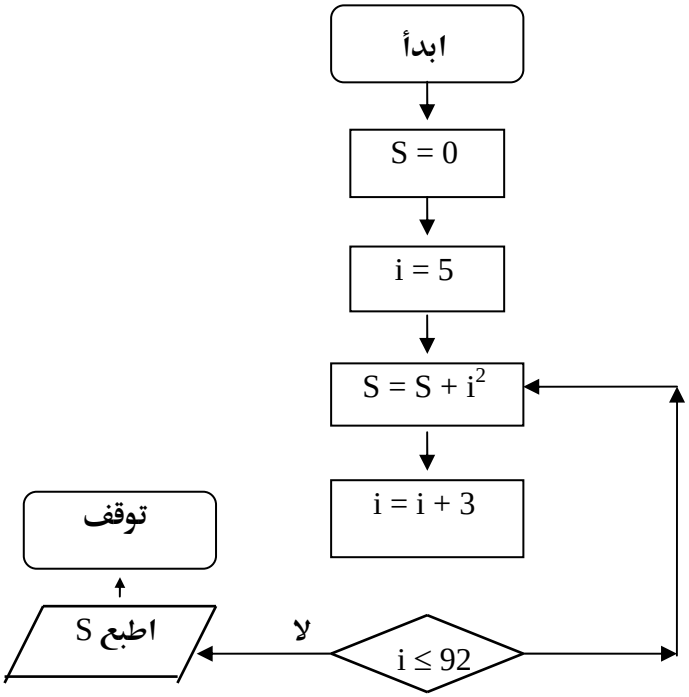
٢١-١



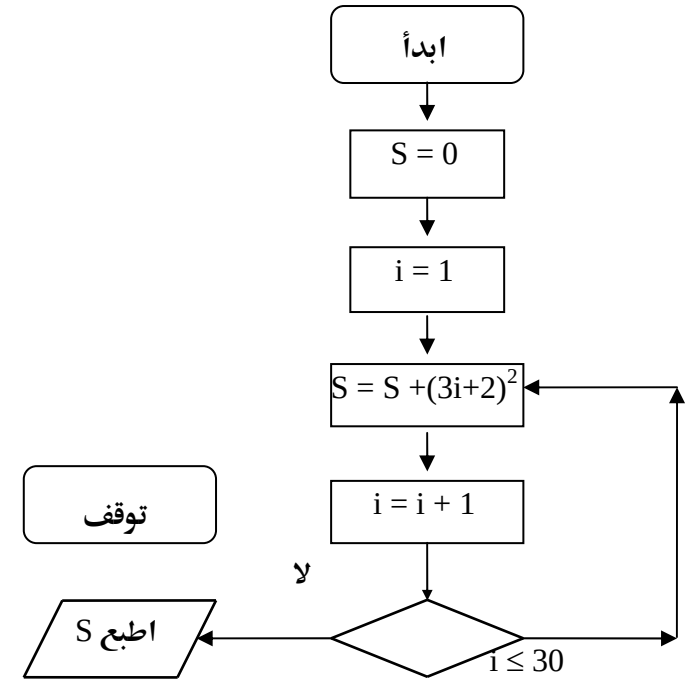
١-٢٢


اقرأ
a , b , c
a < b + c
b < a + c
& c < a + b
لا
اطبع
'NO TRIANGLE'
توقف
نعم
S = (a+b+c) / 2
R = √S(S − a)(S − b)(S − c)
P=2S
Cos A = (b²+c²-a²)/(2bc)
Cos B = (a²+c²-b²)/(2ac)
Cos C = (a²+b²-c²)/(2ab)
اطبع R , P ,
Cos A, Cos B , Cos C

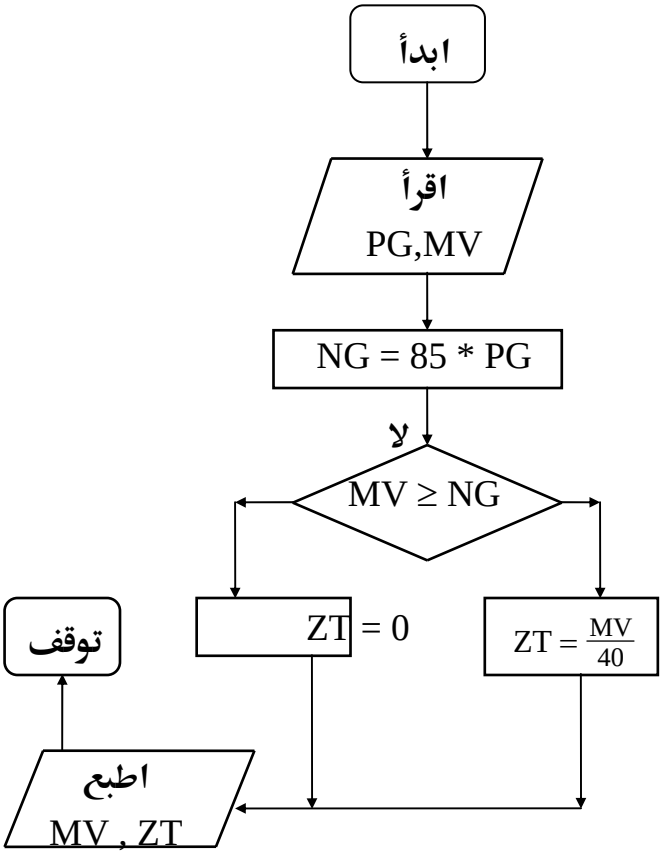
١-٢٣


ابدأ
S = 0
i = 5
S = S + i²
i = i + 3
i ≤ 92
نعم
لا
اطبع S
توقف

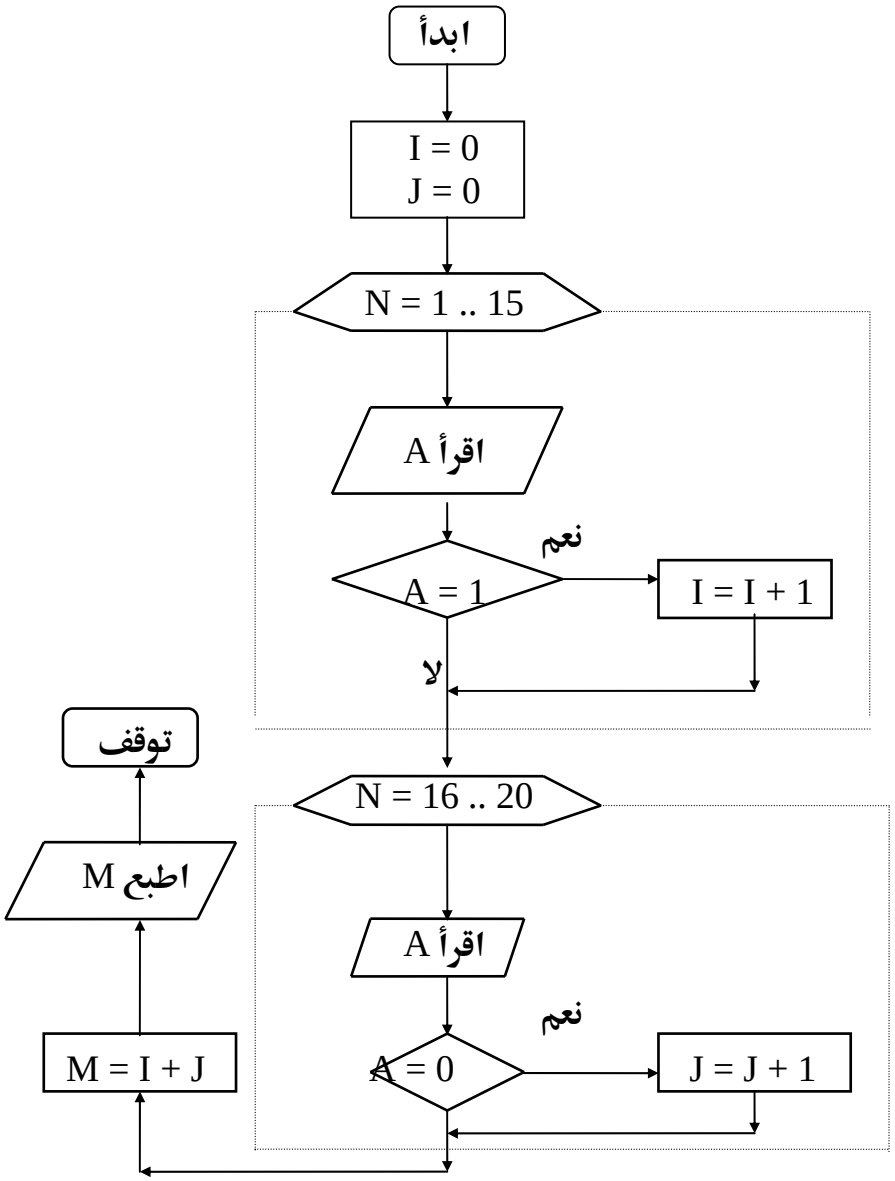
حل آخر


ابدأ
S = 0
i = 1
S = S +(3i+2)²
i = i + 1
i ≤ 30
نعم
لا
اطبع S
توقف

١-٢٤



١-٢٥


ابدأ
I = 0
J = 0
N = 1 .. 15
اقرأ A
نعم
A = 1
I = I + 1
لا
N = 16 .. 20
اقرأ A
نعم
A = 0
J = J + 1
M = I + J
اطبع M
توقف

أجوبة تمرينات الفصل الثاني

١-٢

a)30000.0
b) 5.08
c) -5.245
d) 0.00000293

٢-٢

ث.ص :	d
ث.ح :	c , h , j
ث.ر :	i , k , n , v
م :	a , m , p , q
ل :	b , e , f , g l , r , s , t , u

٣-٢

a.(A + 1.0) * (A + B + 6.0)
b. (A - B / C) - 3.0
c. EXP (I*LN(A)) - EXP (C*LN(B) + SQRT (A + B)
d. EXP (J*LN(I)) + 64
e. (a + b) / (c + d / (e + f))
f. (a / b - 1) / (g * (g / d - 1))
g. 3.14159 * SQR (r) * h
h. EXP (1.0 / 3.0 * LN (3.0 * V / (4.0 * 3.14159)))
i. 3.14159 * h / 3.0 * (SQR (R) + R * R1 + SQR (R1)) (R1 ≡ r)
j. (X + 3.14159) / 12.0
k. - SQR(SQR(COS(x))/x + SIN(x)*SQR(SIN(x)*COS(x)) / 5.0
l. LN(ABS((1.0 + SIN(x))/COS(x))) - 1.0/(3.0*SQRT(SQR(x)-SQR(a)))
m. B*C/12.0*(6.0*SQR(x)*(1-x/a) + SQR(b)*EXP(3*LN(1.0-x/a)))

٤-٢

```
A := EXP ((k-1)*LN ((a1 + b) / (c + d)))
B := EXP ((t-1)*LN (a * b / (c + d/3.0 + 8)))
C := EXP (3*LN (alfa)) * 1.0 E 22 * ABS (EXP(SIN(fi))) / 120.0
D := EXP (x) * SIN (x) / (ABS (y) + COS (z))
E := SQRT (ABS (COS (a - n * b)) + LN (ABS (m - n)))
F := C * SIN (fi - t)
G := 1.0 + x + SQR (x) / 2.0 * (1.0 + x / 3.0 + SQR (x) / 12.0)
```

```
H := SIN (x)*(1.0 + SIN (x)*(1.0 + SIN (x)*(1.0 + SIN (x)*(1.0 +
        SIN (x)*(1.0 + SIN (x))))))
P := EXP (2*m*LN(i*k/(3*L))) + SQR (SQR(j))
Q := a0 + x*(a1 + x*(a2 + x*(a3 + x*(a4 +a5*x))))
R := EXP (-p*LN(a / b)) / (1 - c/d) + SIN (ABS ((x - y) / (x + y)))
```

٢-٥ N := ROUND (I / J)

٢-٦

i) الطرف الأيسر ليس اسم متغير واحد (بل هو تعبير يشتمل على علامة الطرح).

ii) الطرف الأيسر ليس اسم متغير (بل يشتمل على نقطة).

iii) رمزان حسابيان متجاوران (-*).

iv) X 16.9 لا يقبل ، ربما رمز حسابي ناقص بين16.9 , X .

٢-٧

a)صحيحة

One) S = 3E4 + EXP (T + S) + 25*Y / (-7.0)

a) الطرف الأيسر ليس اسم متغير (بل دالة)

b) الطرف الأيسر ليسم اسم متغير (بل تعبير يشتمل على علامة الجمع)

One) e), f), g), h), i), j), k) هذه العبارات كلها صحيحة

a) تحذف النقطة في نهاية العبارة

٢-٨

```
T := X ;
X := Y ;
Y := T ;
```

حيث T متغير (مؤقت) تحفظ فيه قيمة X الأصلية قبل أن تمسح لتحل محلها قيمة Y.

ملاحظة : الحل التالي

```
X := Y ;
Y := X ;
```

والذي قد يتبادر إلى الذهن مباشرة غير صحيح ، وذلك لأن العبارة الثانية تخزن في Y قيمة X الجديدة والتي هي نفسها قيمة Y الأصلية.

٢-٩

a)	8.8	b)	10.5	c)	6.4
d)	256.0	e)	2.5	f)	2.4
g)	5.0	h)	6.4166667	i)	6.0
j)	7	k)	6	l)	4

٢-١٠

I = 3 , A = 8.7 , B = 3.2 , F = 3.0

٢-١١

I = 3 , X = -513.0 Y = -85.5 Z = -85.5

٢-١٢

One)5. 34964E + 03
Two) 6.6E + 01
Three)7.94E - 03
Four) 4.6955468E + 02
a) -1.465386E + 03

٢-١٣

bbbb-3.86

٢-١٤ -9

٢-١٥

a)eltrns := sqr(t)*t*b*exp(-eg/(2.0 * k)) / (k*sqrt(id+is)) ;
b) t := sqr(sin(w))*sin(w)*sqr(cos(w))) / (5.0*p) +
(exp(-v*w/(2.0*p))) ;
c) w := x*arctan(x+sqrt(a))-a*ln(sqr(x)+sqr(a)) / 2.0 ;
d) den := f*sqr(1.015+0.85*f)*vis/t ;
e) freq := (1+sqr(sin(a))*sin(a))*exp(a*(2.0+b)) / (b*cos(a/2.0)) ;
f) j := ln(h)+0.622*p*(1.0/(p-x)-1.0/(p-y)) ;
g) f := 0.35E-4.0*x*sqrt(2.0*g)*sqr(y2-y1)/ln(y2+y1) ;

٢-١٦

```
Var
      b , c , d , g : integer ;
      e , x         : real ;
      a , f , h     : boolean ;
      k             : char ;
```

٢-١٧

a := 3*(d+e) - (2*e)

تبسط إلى :

a := 3*d+e

٢-١٨

(أ) $f = \frac{pq}{r} \cdot \frac{s}{t}$

(ب) $g = \frac{xy^3}{a+2b}$

(ج) $h = \frac{x^4}{4!}$

(د) $p = \frac{r\theta^2}{ab^4}$

(هـ) $q = \sqrt{x^2 + y^2}$

(و) $t = Cos^2(x + 2y)$

٢-١٩

(أ) ناقص قوسان يحيطان بأسماء المتغيرات.

(ب) يجب كتابة فاصلة بين أي اسمي متغيرين متتاليين.

(ج) العبارة صحيحة وليس بها أخطاء.

(د) لا يجوز كتابة عبارة إسناد داخل عبارة الطباعة.

(هـ) العبارة صحيحة وليس بها أخطاء. لاحظ أنه يجوز كتابة تعبير حسابي داخل عبارة الطباعة.

٢-٢٠

(i) x = 7.4328E + 02

(ii) x = Δ7.43E + 02

(iii) x = ΔΔ743.28

(iv) x = 743.3

(هـ) x = Δ7.4E + 02

٢-٢١

One) n = 15 total = 8

Two) result = 3

Three) خطأ لأن عملية القسمة div غير معرفة بصورة صحيحة

Four) sum 15 5

Five) sum + n

Six) خطأ لأن sum = ليس اسم متغير وليس تعبيرا

٢-٢٢

```
Program Wages (Input , Output) ;
Const Tax = 25.0 ;
Var  Hours , Rate , Gross , Net : real ;
Begin
      Writeln ('Enter hours worked') ;
      Readln (Hours) ;
      Writeln ('Enter hourly Rate') ;
      Readln (Rate) ;
      Gross := Hours * Rate ;
      Net := Gross - Tax ;
```

```
        Writeln (′Gross Pay is KD′, Gross)
        Writeln (′Net pay is KD′, Net)
END.
```

Output مخرجات البرنامج

```
Gross Pay is KD 300
Net pay is KD 275
```

٢٣-٢

```
    1)Z := M/5 ;
2)      f := q1 * q2 / SQR(d) ;
3)      P := 0.0299 * ( (t + 460) / V ;
4)      f := g * m * n / SQR (r) ;
```

٢٤-٢

```
writeln (′Palestine has been occupied in ′, y1:5) ;
writeln (′Golan Heights and West Bank have been occupied in′,y2:5);
writeln (′South of Lebanon has been occupied in ′, y3:5);
writeln (′Jihad is the only way to liberate the occupied lands. ′) ;
```

٢٥-٢

```
Program Monotheism ;
Begin
        Writeln (′There is no god but Allah′) ;
        Writeln (′Mohammad is the Messenger of Allah′)
End.
```

٢٦-٢

```
Program Shapes ;
        {Computing areas , voumes and weights}
Const
        Pi = 3.1415927 ;
var
        a , h , d , Area , V , SA , W : real ;
begin
        writeln (′enter 3 real numbers : a , h , d′) ;
        readln (a , h , d) ;
        Area := 0.5 * a * h ;
        write (′Area of a triangle of base a and height h = ′) ;
        writeln (Area : 10 : 3) ;
        V := Pi * a * a * h ;
        SA := 2.0 * Pi * a * h ;
        writeln (′A cylinder of base radius a and height h has : ′) ;
        writeln (′Volume = ′ , V : 10 : 3) ;
```

```
        writeln (′and Surface Area = ′, SA : 10 : 3) ;
        V := 1/3 * Pi * a * a * h ;
        write (′Volume of a right circular cone of base radius a′) ;
        writeln (′ and height h is : ′, V : 10 : 3) ;
        write (′Volume of a right square pyramid of base length a′) ;
        writeln (′ and height h is : ′, V : 10 : 3) ;
        W := 4/3 * Pi * a * a * a * d ;
        write (′Weight of a sphere of radius a and specific′) ;
        writeln (′ density d is : ′, W : 10 : 3) ;
end.
```

٢٧–٢

```
PROGRAM JihadPay (INPUT , OUTPUT) ;
VAR Days : INTEGER ;
        Wage , Regpay , Overtm : REAL ;
        Totpay                 : REAL ;
BEGIN
        WRITELN (′Enter Days , Wage ′) ;
        READLN (Days , Wage) ;
        Regpay := 40.0 * Wage ;
        Overtm := (Days - 40.0) * Wage * 1.5 ;
        Totpay := Regpay + Overtm ;
        WRITELN (′Regpay = ′ , Regpay) ;
        WRITELN (′Overtm = ′ , Overtm) ;
        WRITELN (′Totpay = ′ , Totpay) ;
END.
```

٢٨–٢

```
PROGRAM ZakatulFitr (INPUT , OUTPUT)
VAR N , Z : INTEGER ;
BEGIN
        WRITELN (′Enter N′) ;
        READLN (N) ;
        Z := N ;
        WRITELN (′Number of persons = ′, N) ;
        WRITELN (′Zakat-Ul-Fitr = ′, Z)
END.
```

٢٩–٢

```
PROGRAM Rectangle (INPUT , OUTPUT) ;
VAR         L , W , C , A , D : REAL ;
BEGIN
        WRITELN (′ENTER L , W′) ;
```

```
        READLN (L , W) ;
        C := 2.0*(L + W) ;
        A := L*W ;
        D := SQRT (SQR(L) + SQR(W)) ;
        WRITELN (′Cirumference = ′, C) ;
        WRITELN (′Area = ′, A) ;
        WRITELN (′Diagonal = ′, D) ;
END.
```

٣٠–٢

```
PROGRAM Computations (INPUT , OUTPUT) ;
CONST     P1 = 3.1415927 ;
VAR       R , C , A , V : REAL ;
BEGIN
        WRITELN (′Enter R′) ;
        READLN (R) ;
        C := 2.0 * P1 * R ;
        A := P1 * SQR (R) ;
        WRITELN (′For R =′, R , ′:′) ;
        WRITELN (′Circumference = ′, C) ;
        WRITELN (′Area = ′, A) ;
        WRITELN (′Volume = ′, V) ;
END.
```

٣١–٢

```
PROGRAM Motion (INPUT , OUTPUT) ;
VAR       a , t , d , v   : REAL ;
BEGIN
        WRITELN (′Enter acceleration and time ′) ;
        READLN (a , t) ;
        d := 0.5 * a * SQR (t) ;
        v := a * t ;
        WRITELN (′Distance covered = ′, d) ;
        WRITELN (′Ultimate velocity = ′, v) ;
END.
```

٣٢–٢

```
PROGRAM RightCircularCone (INPUT , OUTPUT)
VAR       R , R1 , h , v : REAL ;
BEGIN
        WRITELN (′Enter 2 radii and height′) ;
        READLN (R , R1 , h) ;
        v := 3.1415927 / 3.0 * h * (SQR(R) + R * R1 + SQR (R1))
```

```
        WRITELN (′Volume = ′, v) ;
END.
```

٣٣–٢

```
program TempConvert ;
{
Program prompts user for Fahrenheit temperature and displays
Celsius equivalent.
}
var
        Fahrenheit ,        {input - temperature}
        Celsius : Real ;    {output - temperature}
begin
        Write (′Please enter the temperature in Fahrenheit : ′) ;
        ReadLn (Fahrenheit) ;
        Celsius := 5 * (Fahrenheit - 32) / 9 ;
        WriteLn (Fahrenheit:5:2,′degrees F. =′,Celsius:5:2,′degrees
C.′);
end.
```

٣٤–٢

```
program SimpleMath ;
{Program performs simple arithmetic on Integer values.}
var
        X , Y ,                          {input}
        Sum , Difference ,
        Product , Quotient : Integer ; {output}
begin
        write (′Please enter two integer values : ′) ;
        ReadLn (X , Y) ;
        Sum := X + Y ;
        Difference := X - Y ;
        Product := X * Y ;
        Quotient := X / Y ;
        WriteLn (′Sum of numbers is ′, Sum : 1) ;
        WriteLn (′Difference of numbers is ′, Difference : 1) ;
        WriteLn (′Product of numbers is ′, Product : 1) ;
        WriteLn (′Quotient of numbers is ′, Quotient :4 : 2) ;
end.
```

٣٥–٢

```
program Weight ;
{
```

```
Program prompts user for weight in pounds and displays kilogram
and gram equivalents.
}
const
    KgPerPound = 0.453592 ;
var
    Pounds ,                    {input}
    Kilograms , Grams : Real ;  {output}
begin
    Write (′Please enter a weight in pounds : ′) ;
    ReadLn (Pounds) ;
    Kilograms := Pounds * KgPerPound ;
    Grams := Kilograms * 1000 ;
    Write (Pounds : 1 : 2 , ′ pounds is ′ , Kilograms : 1 : 6) ;
    WriteLn (′Kilograms or ′, Grams : 1 : 5 , ′grams.′)
end.
```

٣٦-٢

```
program Initials ;
{Program displays the block letter E.}
begin
    WriteLn (′******′) ;
    WriteLn (′*     ′) ;
    WriteLn (′****  ′) ;
    WriteLn (′*     ′) ;
    WriteLn (′*     ′) ;
    WriteLn (′******′) ;
end.
```

٣٧-٢

```
program CutGrassTime ;
{
Program prompts user for dimensions of rectangular yard and house,
displays time required to cut grass.
}
Const
Rate = 2 ;              {cutting rate of mower}
var
    Length, Width,      { input - dimensions rectangle}
    AreaYard ,          {area of rectangular yard}
    AreaHouse,          {area of yard covered by house}
    Time : Real ;       {output - time to cut grass}
```

```
begin
        Write {′Please enter the length of yard in meters : ′) ;
        ReadLn (Length) ;
        Write (′Please enter the width of yard in meter : ′) ;
        ReadLn (Width) ;
        AreaYard := Length * Width ;
        Write (′Please enter the length of house in meters : ′) ;
        ReadLn (Length) ;
        Write (′Please enter the Width of house in meters : ′) ;
        ReadLn (Width) ;
        AreaHouse := Length * Width ;
        Time := (AreaYard - AreaHouse) / Rate ;
        Write (′Lawn cutting time is ′, Time : 5 : 0 , ′seconds′) ;
        Time := Time / 60 ;
        WriteLn (′or approximately′, Time :2 : 0, ′minutes.′) ;
end.
```

(أ ٣٨–٢

```
program FractionProduct ;
{
Program displays the product of two fractions and its percent
equivalent.
}
var
        X1 , X2 , Y1 , Y2 : Integer ;  {input - fraction parts)
        Num : Integer ;                {output - product numerator}
        Den : Integer ;                {output - product denom}
        Percent : Real ;               {output - % equivalent}
begin
        Write (′Enter numerator of first fraction : ′) ;
        ReadLn (Y1) ;
        Write (′Enter denominator of first fraction : ′) ;
        ReadLn (X1) ;
        Write (′Enter numerator of second fraction : ′) ;
        ReadLn (Y2) ;
        Write (′Enter denominator of second fraction : ′) ;
        ReadLn (X2) ;
        {compute product of the fractions}
        Num := Y1 * Y2 ;
        Den := X1 * X2 ;
        Percent := Num / Den * 100 ;
```

```
        WriteLn (′The product numerator and denominator are′,
                    Num, ′  ′ , Den) ;
        WriteLn (′and the percent equivalent is′, Percent : 4 : 2,′%.′) ;
end.
```

ب)

```
program FractionSum ;
{
Program displays the sum of two fractions and its percent equivalent.
}
var
        M ..... (أ كما سبق في
begin
        M ..... (أ كما سبق في
        {compute sum of the fractions}
        Num := Y1 * X2 + Y2 * X1 ;
        Den := X1 * X2 ;
        Percent := Num / Den * 100 ;
        WriteLn (′The sum numerator and denominator are′,
                    Num, ′   ′, Den) ;
        WriteLn (′and the percent equivalent is′, Percent :4:2, ′ % .′) ;
end.
```

٣٩-٢

```
program Pythagorean ;
{
Program generates Pythagorean triples based on user input values.
}
var
        M , N ,              {input - two integers}
        Side1 , Side2 ,
        Hypot : Integer ;  {output - triangle dimensions}
begin
        Write (′Please enter an integer : ′) ;
        ReadLn (N) ;
        Write (′Please enter an integer larger than ′, N , ′ : ′) ;
        ReadLn (M) ;
        {compute values of triangle sides}
        Side1 := M * M - N * N ;
        Side2 := 2 * M * N ;
        Hypot := M * M + N * N ;
```

```
        Write (′Pythagorean triple is the integers ′ ;
        WriteLn (Side1 : 3 , Side2 : 3 , Hypot : 3 , ′ . ′) ;
end.
```

٤٠–٢

```
program Projectile ;
        {This program computes a projectile's time of flight and height
        at impact.}
const
                G = 32.17 ;         {gravitational constant}
var
        Theta ,             {input - angle of elevation , radians}
        Distance ,          {input - distance (ft) to target}
        Velocity ,          {input - projectile vel. (ft/sec)}
        Time ,              {output - duration (sec) of flight}
        Height : Real ;     {output - height at impact}
begin {Projectile}
        {Get projectile information}
        WriteLn (′Angle of elevation in radians > ′) ;
        ReadLn (Theta) ;
        WriteLn (′Distance to target in feet > ′) ;
        ReadLn (Distance) ;
        WriteLn (′Projectile velocity (ft/sec) > ′) ;
        ReadLn (Velocity) ;
{
Compute and display time of flight and height at impact.
}
        Time := Distance / (Velocity * Cos (Theta)) ;
        Height := Velocity * Sin (Theta) * Time - (G * Sqr (Time)) / 2 ;
        WriteLn (′Time of flight = ′ , Time : 4 : 2, ′seconds.′) ;
        WriteLn (′Height at impact = ′, Height : 4 : 2 , ′feet.′)
end. {Projectile}
```

٤١–٢

```
program Cyclist ;
{
A cyclist decelerates at a constant rate. The program reads initial
velocity, a time interval, and the velocity at the end of the time
interval. It then computes and prints the cyclist's acceleration and the
length of time for him or her to come to rest.
}
Const
```

```
        SecPerHour = 3600 ; {Number of seconds in one hour}
var
        InitVel,            {input - Initial velocity}
        FinalVel,           {input - Final velocity}
        TimeInt,            {input - Time interval}
        Acceleration,       {output - Acceleration}
        StopTime : Real; {output - Time to come to rest}
begin {Cyclist}
        WriteLn (′Enter starting and ending velocities (mph), ′) ;
        Write (′and time interval (seconds) : ′) ;
        Readln (InitVel, FinalVel, TimeInt) ;
        {Compute output values}
        {To compute StopTime , use final velocity of zero}
        Acceleration := (FinalVel - InitVel) / (TimeInt / SecPerHour) ;
        StopTime := (0 - InitVel) / Acceleration * SecPerHour ;
        {Print results}
        WriteLn (′The cyclist ′ ′s acceleration rate is′ ,
                    Acceleration :5 : 3, ′miles per hour squared. ′) ;
        WriteLn (′He or she will come to rest after ′,
                    StopTime : 4 : 2, ′seconds.′)
end. {Cyclist}
```

٤٢–٢

```
program Manufacture ;
{
Calculates and prints the cost of producing open topped cylindrical
containers
}
const
        MyPi = 3.14159 ;
Var
        Radius ,                {input - radius of base}
        Height ,                {input - height of cylinder}
        Surface ,               {surface area of cyilnder}
        Cost ,              {input - cost per square cm}
        UnitCost ,              {output - one container}
        TotalCost : Real ;      {output - all containers}
        Quantity : Integer ;    {input - # of containers}
bgein {Manufacture}
        Write (′Enter the radius of the top > ′) ;
        ReadLn (Radius) ;
        Write (′Enter the height of the cylinder >′) ;
```

```
        ReadLn (Height) ;
        Write (′Enter the cost per square cm > ′) ;
        ReadLn (Cost) ;
        Write (′Enter the quantity to be made > ′) ;
        ReadLn (Quantity) ;
        {calculate the surface and costs}
        Surface := MyPi*Radius*Radius + 2*MyPi*Radius*Height ;
        UnitConst := Cost * Surface ;
        TotalCost := UnitCost * Quantity ;
        {display the unit cost and the total cost}
        WriteLn (′The unit cost of a container is $′, UnitCost :4 : 2, ′.′)
;
        WriteLn (′The cost of ′, Quantity , ′ containers is $ ′,
                                                    TotalCost : 5 : 2 , ′.′);
end. {Manufacture}
```

أجوبة تمرينات الفصل الثالث

١-٣

(i)IF SQR(a) <= b + c THEN y := 3 * x ;

(ii) IF (7.0 < x) AND (x < 10.0) THEN
t := SIN (x + n * y)
ELSE a := 0.0 ;

(iii) IF a <= b THEN write (′a = ′, a) ;

(iv) IF (f <= EXP (3.0 * LN (g)) + h) OR (d >= 0.0)
THEN y := EXP (-x) ;

(هـ) IF a - b <= 100.0 THEN K := 1
ELSE K := 2 ;

(vi) IF (x < 0.0) OR (x < y +z) THEN t := 2.0 * x ;

(vii) IF (x < 0.0) AND (y < 0.0) THEN
BEGIN
x := - x ;
y := - y
END.

(viii) IF (ABS(Delta)<eps1)AND(ABS(y)<eps 2)THEN x := 2x + 1 ;

(ix) IF (f <> g) OR (a * b < 0.0) THEN
f := EXP ((2.0 * alpha - 1.0) * LN (Fi)) ;

٢-٣

One)IF x <= 14.0 THEN write (x) ;

Two) IF (i = K) OR (i + j > 8) THEN I := 10 ;

Three)IF (a < 82.0) OR (b < 82.0) THEN z := y ;

a) ليس فيها أخطاء

One) اسم لمتغير ، فيجب أن يكون كلمة واحدة : upper bound

هكذا upperbound

Two) IF X = 0.0 THEN XO := XO + 1.0 ;

a) ليس بها أخطاء

One) IF X > 0.0 THEN
BEGIN
Y := Y + 1.0 ;
WRITELN (X , Y)
END
{END IF} ;

Two) IF TAS >= A THEN
Z := TAS * 2.5 / 100.0

```
        ELSE
                Z := 0.0 ;
Three)IF (a > 0.0) AND (a < b) THEN
                b := b + 1.0 ;
```

٣-٣

One) (False) and (True) = False
Two) (False) or (True) = True
Three)(False) or (False) = False
Four) (not (False)) and (not (False)) = (True) and (True) = True
Five) (False) or (True) and (False) = (False) or (False) = False

٤-٣ SECOND

٥-٣ NOT DONE

٦-٣

(i) صادق لأن التعبير الأيسر صادق.

(ii) خاطئ لأن التعبير الأيمن خاطئ.

(iii) خاطئ لأن التعبيرين خاطئان.

٧-٣

```
if ((i = j) and (j = k)) then
        writeln (′EQUAL′)
else if ((i <= j) and (j <= k)) then
        writeln (′ASCENDING ORDER′)
else if ((i >= j) and (j >= k)) then
        writeln (′DESCENDING ORDER′)
else
        writeln (′OUT OF ORDER′) ;
```

٨-٣

(أ) باستخدام بنية IF المتداخلة :

```
IF S < 60.0 THEN
        WRITELN (′S = ′, S, ′Grade = F′)
ELSE IF S < 70.0 THEN
        WRITELN (′S = ′, S, ′Grade = D′)
ELSE IF S < 80.0 THEN
        WRITELN (′S = ′, S, ′Grade = C′)
ELSE IF S < 90.0 THEN
        WRITELN (′S = ′, S, ′Grade = B′)
```

```
ELSE
      WRITELN (′S = ′, S, ′Grade = A′)
{End IF} ;
```

(ب) استخدام بنية CASE :

```
CASE ROUND ((S + 5.0) / 10.0) OF
      10 , 11       : WRITELN (′S = ′, S, ′Grade = A′)
      9             : WRITELN (′S = ′, S, ′Grade = B′)
      8             : WRITELN (′S = ′, S, ′Grade = C′)
      7             : WRITELN (′S = ′, S, ′Grade = D′)
      6,5,4,3,2,1   : WRITELN (′S = ′, S, ′Grade = F′)
END ;
```

٩-٣

```
CASE   G   OF
      ′A′   :  BEGIN
                  WRITELN (′excellent′) ;
                  NA := NA + 1
               END ;
      ′B′   :  BEGIN
                  WRITELN (′Very Good′) ;
                  NB := NB + 1
               END ;
      ′C′   :  BEGIN
                  WRITELN (′Good′) ;
                  NC := NC + 1
               END ;
      ′D′   :  BEGIN
                  WRITELN (′Pass′) ;
                  ND := ND + 1
               END ;
      ′F′   :  BEGIN
                  WRITELN (′Fail′) ;
                  NF := NF + 1
               END ;
END {CASE} ;
```

١٠-٣ (أ) X is 25.0

(ب) X is 90.00

X is 90.0

١١-٣

(أ) i) 0 ii) 0 iii) 1 iv) 3 v) 0

(ب)

```
t := Trunc (angle /90) mod 4 ;
if t = 0 then writeln ('first quadrant')
else if t = 1 then writeln ('second quadrant')
else if t = 2 then writeln ('third quadrant')
else if t = 3 then writeln ('fourth quadrant')
```

ملاحظة : يمكن الاستغناء عن الشرط الأخير (if t = 3 then)

٣-١٢ (i) متكافئتان (ii) متكافئتان (iii) غير متكافئتين.

السبب : في حالة $x \leq 5$ فإن $z := 2$ في القطعة اليمنى بينما تكون z غير معرفة في القطعة اليسرى.

٣-١٣ القطعة الأولى (i) ليس بها أي أخطاء (منطقية أو تركيبية)

القطعة الثانية (ii) ليس بها أخطاء منطقية ولكن بها أخطاء تركيبية ، وتصحيحها هو :

```
if (a <= b) and (a <= c) then
      smallest := a
else if (b <= c) and (b <= a) then
      smallest := b
else
      smallest := c ;
writeln (smallest) ;
```

٣-١٤ True and (0 = 3) $\Rightarrow$ True and false $\Rightarrow$ false

القوسان الخارجيان ضروريان لأن and لها أولوية قبل = ولا معنى للتعبير True and 0

القوسان الداخليان غير ضروريين لأن mod لها أولوية قبل = فلن يحدث التباس بحذفهما.

٣-١٥

```
if I mod J = 0 then
      flag := true
else
      flag := false ;
```

٣-١٦

(أ) العبارة (i) ترجمة صحيحة للقانون والعبارة (ii) ترجمة خاطئة

(ب) speed = 100 => fine = 50 in (i) & fine = 20 in (ii)

(i)

```
CASE speed of
      0 .. 60   : fine := 0 ;
     61 .. 80   : fine := 20 ;
     81 .. 100 :  fine := 50
else
      fine := 70
```

```
end ;
```

٣-١٧ false true false

٣-١٨ وظيفة عبارة case :

تحويل الرقم المكتوب إلى الترتيب المقابل :

مثلا 1 إلى 1st

2 إلى 2nd

3 إلى 3rd .. وهكذا.

إذا كانت n = 8 فإن المخرجات هي : 8th

٣-١٩

```
CASE number of
      0 , 2 , 5  : writeln (′Victory or Martyrdom′) ;
         1 , 4  : writeln (′Jihad is the only way′) ;
            3  : writeln (′First opening of Afghanistan was in 21a.h.′);
End ;
```

٣-٢٠

```
    1.ERROR :  Invalid Student ID.
2.     ERROR :  Invalid Scores.
3.     ERROR :  Invalid Scores.
4.     Student ID = ∇∇∇∇∇1236 Grade ∇ = ∇∇A Point = ∇4
5.     check student ID number       1237 record
       Student ID =  ∇∇∇∇∇1237 Grade = ∇U Point = ∇0
6.     Student ID =  ∇∇∇∇∇1238 Grade = ∇B Point = ∇3
7.     Student ID =  ∇∇∇∇∇1239 Grade = ∇C Point = ∇2
```

٣-٢١

```
PROGRAM ZakatFruit (INPUT , OUTPUT) ;
VAR
      I : INTEGER ;
      F , B , ZF : REAL ;
BEGIN
      READLN (B , F , I) ;
      IF F < B THEN
            ZF := 0.0
      ELSE IF I = 1 THEN
            ZF := F / 10.0
      ELSE
            ZF := F / 20.0 ;
      WRITELN (′F = ′, F , ′ZF = ′, ZF)
END.
```

٢٢–٣

```
Program Weather (input , output) ;
Var
      AvTmp : integer ;
Begin
      Writeln (′Enter avg. temp.′) ;
      readln (AvTmp) ; writeln (′AvTmp = ′, AvTmp) ;
      Case AvTmp of
            -10 .. 9    :  writeln (′very cold′) ;
             10 .. 19  :  writeln (′cold′) ;
             20 .. 29  :  writeln (′mild′) ;
             30 .. 39  :  writeln (′hot′) ;
             40 .. 59  :  writeln (′very hot′) ;
      end
End.
```

٢٣–٣

```
Program ZakatOfTrade (input , output) ;
VAR
      PG , A , MV , ZT        :  real ;
      Year          : char ;
Begin
      Writeln (′Enter PG , Year , MV ′) ;
      Readln (PG , Year , MV) ;
      A := 85 * PG ;
      If (MV >= A) and (Year = ′Y′) then
            ZT := MV / 40
      else
            ZT := 0.0 ;
      Writeln (′MV = ′, MV , ′ZT = ′, ZT)
End.
```

٢٤–٣

```
PROGRAM Tests (INPUT , OUTPUT) ;
VAR
      CC , RC , TS : REAL ;
      G : INTEGER ;
      T1 , T2 , T3 : BOOLEAN ;
BEGIN
      READLN (CC , RC , TS) ;
      T1 := CC < 6.7 ;
      T2 := RC >= 50.0 ;
```

```
        T3 := TS > 70000.0 ;
        IF T1 AND T2 AND T3 THEN G := 10
        ELSE IF T1 AND T2 THEN G := 9
        ELSE IF T1              THEN G := 8
        ELSE                            G := 7 ;
        WRITELN ('CC = ', CC , 'RC = ', RC , 'TS = ', TS) ;
        WRITELN ('Grade = ', G)
END.
```

٣–٢٥

```
Program Estate (INPUT , OUTPUT) ;
VAR
        NS , ND : integer ;
        A , f , m , w , s , d , r : real
Begin
        Writeln ('Enter the estate A in K.D. ') ;
        Readln (A) ;
        Writeln ('Enter the no. of sons and no. of daughters ') ;
        Readln (NS , ND) ;
        f := A / 6 ;
        m := f ;
        w := A / 8.0 ;
        r := A - (f + m + w) ;
        S := 2 * r / (2 * NS + ND) ;
        IF ND > 0 then
                d := S / 2.0
        Else d := 0.0 ;
        Writeln ('f = ', f , 'm = ', m , 'w = ', w) ;
        writeln ('s = ', s , 'd = ', d) ;
End.
```

٣–٢٦

```
PROGRAM DistributionOfEstate (INPUT , OUTPUT) ;
        {Distribution of Al-Mirath (Inheritance) in different cases}
VAR
        NS , ND , I , N : INTEGER ;
        A , H , W , S , D , A1 : REAL ;
BEGIN
        WRITE ('Enter A, NS , ND , I , N ') ;
        READLN (A , NS , ND , I , N) ;
        H := 0.0 ;          W := 0.0 ;
        S := 0.0 ;          D := 0.0 ;
```

```
        {N is the total number of children}
        N := NS + ND ;
        {Computing the share of husband / wife}
        IF (I = 0) AND (N = 0) THEN H := A / 2.0 ;
        IF (I = 0) AND (N <> 0) THEN H := A / 4.0 ;
        IF (I = 1) AND (N = 0) THEN W := A / 4.0 ;
        IF (I = 1) AND (N <> 0) THEN W := A / 8.0 ;
        {Computing the share of daughter / son}
        IF (NS = 0) AND (ND = 1) THEN D := A / 2.0 ;
        IF (NS = 0) AND (ND >= 2) THEN D := 2.0 / 3.0 * A / ND ;
        IF NS <> 0 THEN
                BEGIN
                        A1 := A - H - W ;
                          S := 2.0 * A1 / (2 * NS + ND) ;
                        IF ND > 0 THEN D := S / 2.0
                END ;
        WRITELN ('A = ', A , 'NS = ', NS , 'ND = ', ND , 'I = ', I,
                                'H = ', H , 'W = ', W, 'S = ', S , 'D = ', D)
END {DistributionOfEstate}.
```

وفيما يلي حل آخر للمسألة يعتمد على استخدام البنية IF...THEN...ELSE عدة مرات :

```
BEGIN
        {Husband/wife share}
        IF I = 1 THEN
                BEGIN
                        H := 0.0 ;
                        IF N = 0 THEN
                                W := A/4.0
                        ELSE
                                W := A/8.0
                        {End IF} ;
                        A1 := A - W
                END
        ELSE
                BEGIN
                        W := 0.0 ;
                        IF N = 0 THEN
                                H := A/2.0
                        ELSE
                                H := A/4.0
                        {End IF} ;
                END ;
```

```
        {Son/daughter share}
        IF NS = 0 THEN
                BEGIN
                        S := 0 ;
                        IF ND = 1 THEN
                                D := A/2.0
                        ELSE
                                D := 2.0/3.0*A / ND
                        {End IF} ;
                END
        ELSE
                D := A1/(2*NS + ND) ;
                S := 2.0*D ;
                IF ND = 0 THEN D := 0.0
                {End IF} ;
                {Printing the Results}
                WRITELN ('A = ', A, 'NS = ', NS, 'ND = ', ND, 'I = ', I,
                        'H = ', H, 'W = ', W, 'S = ', S, 'D = ', D)
END {Distribution Of Estate}.
```

٢٧–٣

```
program HouseCosts ;
{
        This program computes the total cost of a house after a number
        of years. It reads the initial cost of the house, the annual fuel
        cost, the annual tax rate and the number of years, and computes
        the total cost over these years.
}
var
        HouseCost ,         {input - Initial cost of house}
        Fuelcost ,          {input - Annual fuel cost}
        TaxRate ,           {input - Annual tax rate}
        Years ,             {input - Number of years}
        Totalcost : Real ; {output - Total cost of house}
begin {HouseCosts}
        Write ('Please enter the initial cost of the house ; $') ;
        ReadLn (Housecost) ;
        Write ('Enter the annual fuel cost : $') ;
        ReadLn (FuelCost) ;
        Write ('Enter the annual tax rate as a decimal fraction') ;
        ReadLn (TaxRate) ;
        Write ('Enter number of years over which to compute : ') ;
```

```
        ReadLn (Years) ;
        {Compute total cost}
        Totalcost := HouseCost +
                        (FuelCost + TaxRate * HouseCost) * Years ;
        {Print total cost}
        WriteLn (′The total cost is $′, TotalCost : 4 : 2)
end. {HouseCosts}
```

٢٨–٣

```
Program StateTax ;
{
        This program determines the additional tax due or the tax
        refund for an employee. Inputs are the employee's gross
        income, number of dependents, and the tax already deducted.
        The actual tax owed is computed by subtracting a fixed
        deduction for each dependent, and then taking a fixed
        percentage of the net. The exact amounts are given in the
        constants below. The difference between tax owed and tax
        deducted is printed with a message indicating refund or
        additional tax due.
}
        Const
        TaxRate = 0.04 ;        {percent tax on net income}
        Allowance = 500.00 ;    {allowance per dependent}
var
        GrossIncome : Real ;    {input - gross income}
        Dependents : Integer ;  {input - number of dependents}
        TaxPaid : Real ;        {input - tax deducted}
        TaxOwed : Real ;        {output - tax if underpaid}
        TaxRefund : Real ;      {output - refund if overpaid}
        NetIncome : Real ;      {interm - net income}
        TaxesDue : Real ;       {interm - tax due}
begin {StateTax}
        Write (′Enter gross income : ′) ;
        ReadLn (GrossIncome) ;
        Write (′Enter number of dependents : ′) ;
        ReadLn (Dependents) ;
        Write (′Enter tax amount deducted : ′) ;
        ReadLn (TaxPaid) ;
        {Compute net income and tax due}
        Netincome := GrossIncome - Dependents * Allowance ;
        TaxesDue := TaxRate * NetIncome ;
```

```
        {
        Compute tax due or refund as appropriate, and print proper
        massage.
        }
        WriteLn ;
        if TaxPaid > TaxesDue then
              begin {Refund}
                    TaxRefund := TaxPaid - TaxesDue ;
                    WriteLn (′Refund $′, TaxRefund : 4 : 2)
              end {Refund}
        else if TaxPaid < TaxesDue then
              begin {Owe}
                    TaxOwed := TaxesDue - TaxPaid ;
                    WriteLn (′Send check for $′, TaxOwed : 4 : 2)
              end {Owe}
           else {Tax due equals tax paid}
              WriteLn (′Tax due exactly equals tax paid.′)
        end. {StateTax}
```

٢٩–٣

```
Program PhoneCall ;
{
This program reads the start time and length of a phone call. It prints
the gross cost and net cost of the call, based on the rate structure
given in the problem statement.
}
const
        NightDiscount = 0.50 ;      {discount offhours}
        TaxRate = 0.04 ;            {tax on calls}
        RegRate = 0.40 ;            {regular call rate}
        LongDiscount = 0.15 ;       {discount for long calls}
        LengthLimit = 60 ;          {
                                    length for long call
                                    discount to apply
                                    }
var
        StartTime : Integer ;   {input - time of call start}
        Length : Integer ;      {input - length of call}
        GrossCost : Real ;      {output - gross cost of call}
        NetCost : Real ;        {output - net cost of call}
        Tax : Real ;            {interm - tax on call}
begin {PhoneCall}
```

```
        Write (′Enter the starting time of the call′) ;
        Write (′  (24 hr clock) : ′) ;
        ReadLn (StartTime) ;
        Write (′Enter the length of the call in minutes : ′) ;
        ReadLn (Length) ;
        {compute call gross cost}
        GrossCost := Length * RegRate ;
        {Subtract evening rate discount if necessary}
        if StartTime < 800 then
                NetCost := GrossCost - GrossCost * NightDiscount
        else if StartTime > 1800 then
                NetCost := GrossCost - GrossCost * NightDiscount
        {compute discount for lengthy calls if applicable}
        if Length > LengthLimit then
                NetCost := NetCost - NetCost*LongDiscount ;
        {Compute fax and add to net cost}
        Tax := NetCost*TaxRate ;
        NetCost := NetCost + Tax ;
        {Print output data}
        WriteLn ;
        WriteLn (′The gross cost is $ ′ , GrossCost : 4 : 2) ;
        WriteLn (′The net cost is $ ′ , NetCost : 4 : 2)
end. {PhoneCall}
```

٣٠–٣

```
Program WaterBills ;
{
        Program Computes and prints water bill based on calss of user
        and gallons used.
}
var
        Code        : Char ;      {input - user code}
        Account     : Integer ;   {input - user account #}
        Gallons ,                 {input - quantity used}
        AmountDue: Real ;         {output - amount billed}
begin {WaterBills}
        {Prompt user for billing information}
        WriteLn (′Account number >′) ;
        ReadLn (Account) ;
        WriteLn (′Enter Gallons used > ′) ;
        ReadLn (Gallons) ;
        WriteLn (′Enter user code : H = Home, C = Commercial , ′,
```

```
                ′1 = Industrial > ′) ;
        ReadLn (Code) ;
        {Echo input data}
        WriteLn (′Data Entered ′) ;
        WriteLn (′Account = ′ , Account : 8) ;
        WriteLn (′Gallons used = ′ , Gallons : 4 : 2) ;
        WriteLn (′Code = ′ , Code) ;
        if (Code <> ′h′) and (Code <> ′H′) and
          (Code <> ′c′) and (Code <> ′C′) and
          (Code <> ′i′) and (Code <> ′I′)
        then
        WriteLn (′Bad code, amount due not computed′)
else
        begin        {Determine amount due}
                case Code of
                        ′h′ , ′H′ : {home use}
                                AmountDue := 5.0 + 0.0005 * Gallons ;
                        ′c′ , ′C′ : {commercial use}
                                if Gallons <= 4.0E+6 then
                                        AmountDue := 1000.0
                                else
                                        AmountDue := 1000.0 +
                                                (Gallons - 4.0E+6) * 0.00025 ;
                        ′i′ , ′I′ : {industrial use}
                                if Gallons > 10.0E6 then
                                        AmountDue := 3000.0
                                else if Gallons > 4.0E6 then
                                        AmountDue := 2000.0
                                else
                                        AmountDue := 1000.0
                        end ; {case}
                WriteLn (′Amount due = $ ′ , Amountdue : 4 : 2)
        end {else}
end {WaterBills}
```

أجوبة تمرينات الفصل الرابع

٤-١ k = 18

٤-٢

i) 8 ii) 8 iii) 6 iv) 1

٤-٣

i) 15 ii) 12 iii) 12

٤-٤ 1.0

٤-٥

i) 246 ii) 246 iii) 420

٤-٦

i) 11 ii) -42 iii) 0

٤-٧ count1 = 0 , count2 = -154

٤-٨

(أ)
Assalamo A′laikom
Assalamo A′laikom
Assalamo A′laikom

(ب) خاطئة لأن عبارة

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

تغيِّر قيمة counterوهو متغير عروة for

(ج)
Your Mother
Your Mother
Your Mother
Your Father

٤-٩

(أ) ليس للمجموعة أي مخرجات لأن عروة While (التي تنتهي عند ; بعد do مباشرة) لا نهائية (infinite loop) (لتحقق شرطها باستمرار 5 => 2 وعدم تغيره) والعروة لا تعمل شيئا ولا تطبع أي قيم.

(ب) التصحيح

```
Product := 1 ;
Counter := 2 ;
While Counter <= 5 do
      begin
            Product := Product * Counter ;
            Counter := Counter + 1
```

```
        end ;
    Writeln (Product) ;
```

(ج)

```
    Product := 1 ;
    Counter := 2 ;
    Repeat
        Product := Product * Counter ;
        Counter := Counter + 1
    Until
        Counter > 5 ;
    Writeln (Product) ;
```

٤-١٠ (أ)

N : 15

i :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
sum : 0	1		4		9		16		25		36		49		64

المخرجات : ∇∇∇64

وظيفة البرنامج : إيجاد مجموع الأعداد الفردية من 1 إلى N.

(ب)

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

٤-١١

(أ)

```
I := 30 ;
while I > 1 DO
   begin
      I := (1 DIV 5) - 1 ;
      I := I*5 ;
     WRITELN (1)
   end ;
```

(ب)

25
20
15
10
5
0

(ج)

```
   For i := 5 Down to 0 DO
        begin
             j := 5 * i ;
             writeln (j)
        end ;
```

٤-١٢

i	Limit	i + Limit
1	7	8
2	5	7
3	5	8
4	5	9
5	5	10
6	5	11
7	5	12

المخرجات : ∇∇8∇∇7∇∇8∇∇9∇10∇11∇12

١٣-٤

(أ) المخرجات

10
20
30
40
50
60
70
80

(ب)

```
i := 1 ;
Repeat
    writeln (i * 10) ;
    i := i + 1
until i > 8 ;
```

(ii)

```
i := 1 ;
while i <= 8 do
   begin
      writeln (i * 10) ;
       i := i + 1
   end ;
```

١٤-٤

التصحيح

```
Repeat
    read (N1 , N2) ;
    writeln (N2 , N1)
until
    (N1 < 0) Or (N2 < 0) ;
```

القطعة المكافئة

```
read (N1 , N2) ;
While (N1 >= 0) and (N2 >= 0) do
   begin
      writeln (N2 , N1)
      read (N1 , N2)
   end ;
```

١٥-٤ (أ) 3548 **(أي عكس ترتيب أرقام العدد n)**

(ب) IJ

∇∇∇∇∇∇∇∇∇∇∇I∇∇∇∇J

i	j	outer	1	
1	1	inner	1	1
2	1	outer	2	
	2	inner	2	1
3	1	inner	2	2
	2	outer	3	
	3	inner	3	1
		inner	3	2

inner 3 3

٤-١٦ المخرجات

```
Enter four real numbers
Slope is 0.500
Enter four real numbers
Slope is - 3.000
Enter four real numbers
Slope is 3.000
Enter four real numbers
Slope is -5.333
```

٤-١٧

```
O Mojahidoon of Afghanistan !
Congratulations ! May Allah bless you.
```

٤-١٨ (أ)

c	i	x
0	1	15
1	2	10
2	3	-10
3	4	5
	5	20
	6	

القيمة النهائية c = 3

x = 20

(ب) – (i)

```
c := 0 ;
i := 1 ;
repeat
        real (x) ;
        If (x < 0) or (x > 10) then
                c := c + 1 ;
        i := i + 1
until i > 5 ;
```

(ب) – (ii)

```
c := 0 ;
For i := 1 to 5 do
        begin
        read (x) ;
                If (x < 0) or (x > 10) then
                   c := c + 1 ;
```

```
end ;
```

٤-١٩

(أ)

```
S = 0
d = 3
s = 0+3=3
n = 2869
d = 9
s = 3+9 = 12
n = 286
d = 6
s = 12+6 = 18
n = 28
d = 8
s = 18+8 = 26
n = 2
d = 2
s = 26+2 = 28
n = 0
```

(ب)

المخرجات

قيم d : 3

```
9
6
8
2
```

قيمة s :

```
28
```

(ج)

يحسب البرنامج مجموع أرقام العدد المعطى n

(د)

n = 20000 ⇒

المخرجات

```
0
0
0
0
2

2
```

٤-٢٠

لا تؤدي العبارات الوظيفة المطلوبة لأن عبارة if غير صحيحة

التصحيح : `if (ch <> ' ') then`

٤-٢١

```
A
A
A
A
```

٤-٢٢ المخرجات : 3

الوظيفة : حساب عدد القيم item المحصورة بين 25 , 15 أي

$15 \le item \le 25$ من بين القيم الست التي يقرأها

٤-٢٣ (أ)

```
10
9
8
```

(ب) Allah is the Greatest

(لاحظ أن عبارة الطباعة تقع خارج عروة for التي لا تعمل شيئا لوجود ; بعد do مباشرة).

٤-٢٤

(أ) العبارات لا تعطي المجموع المطلوب لأن الشرط number >= 100 عندما يتحقق ينقل التنفيذ إلى عبارة طباعة المجموع sum خارج العروة دون إضافة العدد 100

(ب)

i)
```
sum := 0 ;
number := 1 ;
while number <= 100 do
  begin
    sum := sum + number ;
    number := number + 1
  end ;
writeln ('sum = ', sum) ;
```

ii)
```
sum := 0 ;
For number := 1 to 100 do
    sum := sum + number
;
writeln ('sum = ', sum) ;
```

٤-٢٥

```
∇ ∇ ∇ *
∇ ∇ * * *
∇ * * * * *
* * * * * * *
```

٤-٢٦ (أ)

n	i	sum	value	flag
8	1	0	5	false
	2	5	6	true
	3	11	-3	
	4	18	7	
	5		-4	
	6		0	
	7			
		18	0	المخرجات :

(ب) وظيفة البرنامج : قراءة قيم أعداد صحيحة إلى أن يصادف القيمة صفرا أو إلى أن ينتهي من قراءة ثمان قيم ، ثم يعطي مجموع القيم الموجبة التي قرأها وكذلك يعطي آخر قيمة قرأها.

(ج) (i) 1+2+3+4+5+6+7+8 = 36

المخرجات : 36 8

(ii) 3+1+4 = 8

المخرجات : 8 0

٤-٢٧ (أ)

```
begin
    writeln ('How many numbers ... ? ') ;
    readln (Limit) ;
```

```
        EvenNumber := 0 ;
        For Counter := 1 to Limit Do
                begin
                        Writeln (′Enter a digit′) ;
                        readln (Digit) ;
                        If (Digit mod 2 = 0) then
                                EvenNumber := EvenNumber + 1
                end ;
        writeln (′The number of even Numbers was ′, EvenNumber)
end.
```

(ب)

Limit	:	6				
EvenNumber	:	0	1		2	3
Counter		1	2	3	4	5	6
Digit		8	7	3	6	4	5

The number of even numbers was 3

٤-٢٨ (ب) تضاف بعض العبارات في المواضع المبينة فيما يلي :

```
.....
large := - maxint ;
small :=  maxint ;
sum := 0
.....
if x > large then large := x ;
if x < small then small := x ;
sum := sum + x ;
.....
end ; {while}
average := sum / n ;
.....
writeln (′the largest element = ′, large) ;
writeln (′the smallest element = ′, small) ;
writeln (′the average = ′, average) ;
.....
```

٤-٢٩ (أ)

```
program odd_even (input , output) ;
        {This program reads 10 numbers and determines if they are
        odd or even.}
var
        num , i : integer ;
begin
```

```
        for i := 1 to 10 do
                begin
                        readln (num) ;
                        if num mod 2 = 0 then
                                writeln (num : 1, 'IS EVEN.')
                        else
                                writeln (num : 1, 'IS ODD.')
                end
end.
```

(ب)

```
program odd_even (input , output) ;
        {This program reads 10 numbers and determines if they are
        odd or even.}
var
        num , i : integer ;
begin
        for i := 1 to 10 do
                begin
                readln (num) ;
                case odd (num) of
                        false : writeln (num : 1 , 'IS EVEN') ;
                        true : writeln (num : 1 , 'IS ODD')
                end
        end
end.
```

٣٠–٤

```
 program power (input , output) ;
        {This program reads a and b and then computes and prints
        a^b}
var
        a , b , ab , i , j : integer ;
begin
        for i := 1 to 20 do
                begin
                        ab := 1 ;
                        readln (a , b) ;
                        for j := 1 to b do
                                ab := ab * a ;
                        writeln (a:2, 'TO THE POWER' , b:2, '=' , ab:6)
                end
```

```
end.
```

٣١–٤

```
program root (input , output) ;
        {This program computes n so that 2^n is equal to the number
        read.}
var
        num , i , n : integer ;
begin
        for i := 1 to 20 do
                begin
                        n := 0 ;
                        readln (num) ;
                        while num <> 1 do ;
                                begin
                                        n := n + 1 ;
                                        num := num div 2
                                end ;
                        writeln ('2 TO THE POWER' , n:2 , ' = ' , num:6)
                end
end.
```

٣٢–٤

```
PROGRAM Velocity (OUTPUT) ;
        {Time - Velocity Table of a falling body}
VAR
        t , L : INTEGER ;
        v , limit : REAL ;
BEGIN
        limit := SQRT (2.0 * 60000.0 / 9.81) ;
        L := TRUNC (limit) ;
        t := 0 ;
        WHILE t <= L DO
                BEGIN
                        v := 9.81 * t ;
                        WRITELN ('t = ', t, 'v = ', v) ;
                        t := t + 10
                END ;
        WRITELN ('t = ', limit , 'v = ', 9.81 * limit)
END.
```

٣٣–٤

```
PROGRAM ValuesOfZ (OUTPUT) ;
```

```
        {Computing Z for different combinations of x , a , b}
VAR
        i , j , b : INTEGER ;
        z , x , a : REAL ;
BEGIN
        FOR i := 10 TO 20 DO
                BEGIN
                        x := i / 10.0 ;
                        FOR j := 2 TO 16 DO
                                BEGIN
                                        a := j / 20.0 ;
                                        FOR b := 1 TO 10 DO
                                                BEGIN
                                                        Z := (EXP(a*x)
                                        -Exp (-a*x)) / 2.0 *SIN (x+b)
                                        + a*LN ((b+x) / 2.0) ;
                                        WRITELN (′x = ′, x, ′a = ′, a, ′b = ′, b,
                                                        ′Z = ′, Z)
                                                END
                                END
                END
END.
```

٣٤–٤

```
PROGRAM Equations (INPUT , OUTPUT) ;
        {Solution of 20 first - degree equations}
VAR
        i : INTEGER ;
        a , b , x : REAL ;
BEGIN
        FOR i := 1 TO 20 DO
                BEGIN
                        WRITE (′Enter a , b′) ;
                        READLN (a , b) ;
                        IF a <> 0.0 THEN
                                BEGIN
                                        x := -b / a ;
                                        WRITELN (′a = ′, a, ′b = ′, b,
                                        ′Solution is′, x)
                                END
                        ELSE IF b = 0.0 THEN
                                WRITELN (′a = ′, a, ′b = ′, b,
```

```
                    'Any real x is a solution')
            ELSE
                WRITELN ('a = ', a, 'b = ', b,
                    'No real solution')
        END
END.
```

٤–٣٥

```
PROGRAM Series (INPUT , OUTPUT) ;
VAR
    i , n : INTEGER ;
    sum : REAL ;
BEGIN
    WRITE ('Enter n') ;
    READLN (n) ;
    sum := 0.0 ;
    FOR i := 1 TO n DO
        IF i MOD 2 = 0 THEN
            sum := sum - 1 / i
        ELSE
            sum := sum + 1 / i ;
    WRITELN ('sum = ', sum)
END.
```

٤–٣٦

(أ) باستخدام عبارة WHILE :

```
PROGRAM SeriesWithWhile (INPUT , OUTPUT) ;
VAR
    i , n : INTEGER ;
    sum : REAL ;
BEGIN
    WRITE ('Enter n') ;
    READLN (n) ;
    SUM := 0.0 ;
    i := 1 ;
    WHILE i <= n DO
        BEGIN
            IF i MOD 2 = 0 THEN
                sum := sum - 1 / i
            ELSE
                sum := sum + 1 / i ;
            i := i + 1
```

```
        END ;
    WRITELN (′sum = ′, sum)
END.
```

(ب) باستخدام عبارة REPEAT :

```
PROGRAM SeriesWithRepeat (INPUT , OUTPUT) ;
VAR
    i , n : INTEGER ;
    sum : REAL ;
BEGIN
    WRITE (′Enter n′) ;
    READLN (n) ;
    SUM := 0.0 ;
    i := 1 ;
    REPEAT
        IF i MOD 2 = 0 THEN
            sum := sum - 1 / i
        ELSE
            sum := sum + 1 / i ;
        i := i + 1
    UNTIL    i > n ;
    WRITELN (′sum = ′, sum)
END.
```

٤–٣٧

```
FOR i = 10 TO 99 DO
    BEGIN
        x := i / 10 ;
        y := (16.7 + 9.2 * x - 1.02 * SQR(x)) * x ;
        WRITELN (′x = ′, x , ′y = ′, y)
    END ;
```

٤–٣٨

```
PROGRAM FnOf Theta (INPUT , OUTPUT) ;
CONST Pi = 3.14159 ;
VAR
    T , F : REAL ;
BEGIN
    T := -Pi ;
    WHILE T <= Pi DO
    BEGIN
        F := EXP(COS(t)) - 4.0*T*SQR(T) + 2.0*
                        SQRT(ABS(t)) ;
```

```
            WRITELN ('Theta = ', T , 'F = ', F) ;
            T := T + Pi / 16.0
        END
END.
```

٣٩-٤

```
READLN (xin , delta , xfi) ;
x := xin ;
REPEAT
        y := SQRT (1.0 + x) + COS (2.0 * x) / (1.0 + SQRT (x)) ;
        WRITELN ('x = ', x , 'y = ', y) ;
        x := x + delta
UNTIL       x > xfi
```

٤٠-٤

نظرا لأن عدد الدرجات المطلوب قراءتها – أي عدد البيانات – غير محدد في السؤال فسنقوم باتباع طريقة كالمشروحة في حل مثال ٤-٨ حيث سنضيف سجلا بعد سجلات البيانات ونضع عليه العدد ١ ليقابل قيمة متغير K نعرف به انتهاء البيانات ، كما يلي :

```
{CALCULATING THE CENTIGRADE EQUIVALENT OF SOM
GIVEN TEMPERATURES IN FAHRENHEIT DEGREES}
WRITE ('Enter F1 , K') ;
READLN (FI , K) ;
WHILE K = 0 DO
        BEGIN
            T := 5.0 / 9.0 * (FI - 32.0) ;
            WRITELN ('FI = ', FI , 'T = ', T) ;
            WRITE ('Enter FI, K') ;
            READLN (FI , K)
        END
```

٤١-٤

```
PA := 40.0 ;        PB := 3.0 ;          N := 0 ;
REPEAT
        PA := PA + 0.01 * PA ;
        PB := PB + 0.1 * PB ;
        N := N + 1 ;
        WRITELN ('PA = ', PA , 'PB = ', PB)
UNTIL       PB > PA ;
WRITELN ('N = ', N)
```

٤٢-٤

```
S := 0.5 ;
FOR i = 2 TO 99 DO
```

```
        S := S + i / (i + 1) ;
WRITELN (′S = ′, S)
```

٤٣–٤

```
{Solving the equation x = 1.4 cos x by Newton - Raphson Method}
x := 1.0 ;
REPEAT
        d := (x - 1.4 * COS (x)) / (1.0 + 1.4 * SIN (X)) ;
        x := x - d ;
        WRITELN (′x = ′, x)
UNTIL       ABS (d) < 1.0E - 5 ;
```

٤٤–٤

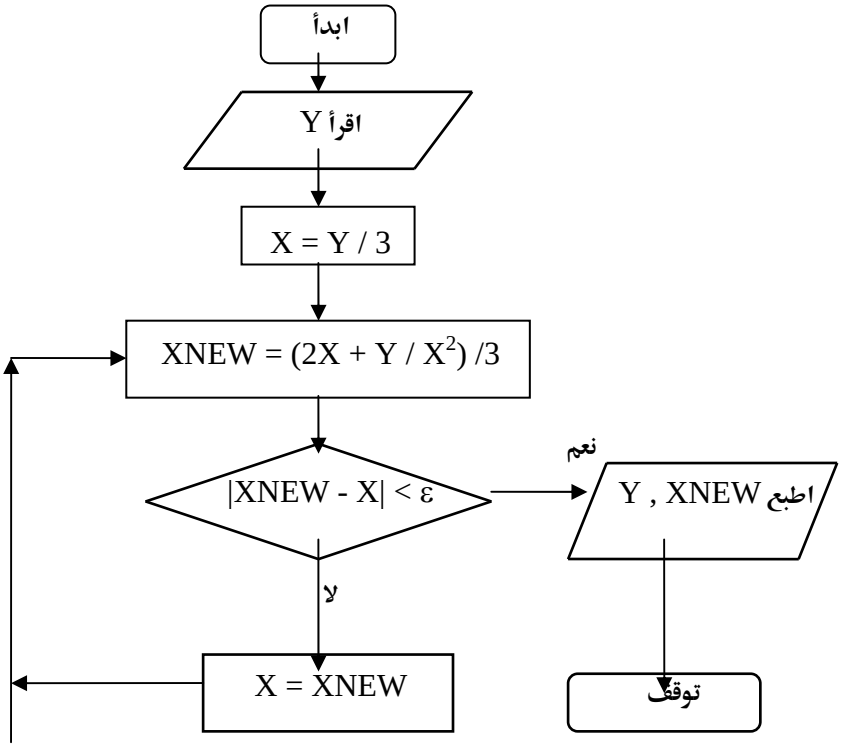
```
{Bisection Method for Solving the equation x - SIN (x) - 0.5 = 0}
a := 3.14159 ; b := 0.0 ;
REPEAT
        c := (a + b) / 2.0 ;
        fa := a - SIN (a) - 0.5 ;
        fc := c - SIN (c) - 0.5 ;
        IF fc * fa > 0.0 THEN
                a := c
        ELSE
                b := c
UNTIL       ABS (fc) < 1E - 6 ;
WRITELN (′c = ′, c)
```

٤٥–٤

```
PROGRAM BisectionCubicEqn (INPUT , OUTPUT) ;
        {Finding the root of a cubic equation by the Bisection
Method′}
VAR   XN , XP , XM , F : REAL ;
BEGIN
        XN := 2.0 ;  XP := 3.0 ;
        REPEAT
                XM := (XN + XP) / 2.0 ;
                F := XM * (SQR (XM) - 4 * XM + 6.0) - 7.0 ;
                IF F > 0.0 THEN
                        XP := XM
                ELSE IF F < 0 THEN
                        XN := XM
        UNTIL       (ABS (XP - XN) < 0.001) or (F = 0.0) ;
        WRITELN (′ROOT = ′ , XM)
END.
```

٤٦–٤

```
{Finding the cubic roots of some positive numbers
  by a given iterative formula}
FOR i := 1 TO 10 DO
BEGIN
      WRITE (′Enter y′) ;
      READLN (y) ;
      x := y / 3.0 ;
      REPEAT
            xnew := (2.0 * x + y / SQR (x)) / 3.0 ;
            d := xnew - x ;
            x := xnew ;
      UNTIL      ABS (d) < 1E - 5 ;
      WRITELN (′y = ′, y , ′cubic root = ′, xnew)
END ;
```



٤٧–٤

```
PROGRAM Pilgrimage (INPUT , OUTPUT) ;
   {To find the number of pilgrims arriving by land, by sea, and by
air}
VAR
```

```
        NL , NS , NA , N , K : INTEGER ;
        PL , PS , PA : REAL ;
BEGIN
        NL := 0 ; NS := 0 ; NA := 0 ;
        REPEAT
                WRITE (′Enter K′) ;
                READLN (K) ;
                IF K = 1 THEN NL := NL + 1
                ELSE IF K = 2 THEN NS := NS + 1
                ELSE IF K = 3 THEN NA := NA + 1
        UNTIL K = 4 ; {K = 4 indicates the end of data}
        N := NL + NS + NA ;
        WRITELN (′NL = ′, NL , ′NS = ′, NS , ′NA = ′, NA , ′N = ′, N)
;
        {Calculate the Percentages}
        PL := NL / N ;
        PS := NS / N ;
        PA := NA / N
        WRITELN (′PL = ′, PL , ′PS = ′, PS , ′PA = ′, PA)
END.
```

٤٨–٤

```
{To calculate the numbers of students in different sections}
NB := 0 ; NG := 0 ; M := 0 ; L := 0 ;
WRITE (′Enter I , J , K ′) ;
READLN (I , J , K) ;
WHILE K <> 3 DO
        BEGIN
                IF (J = 1) OR (J = 2) THEN NB := NB + 1 ;
                IF (J = 51) OR (J = 52) THEN NG := NG + 1 ;
                IF (J = 51) AND (K = 2) THEN M := M + 1 ;
                IF (J = 2) AND (K = 1) THEN L := L + 1 ;
                WRITE (′Enter I , J , K′) ;
                READLN (I , J , K) ;
        END ;
PM := M / NG * 100.0 ;
PL := L / NB * 100.0 ;
WRITELN (′NB = ′, NB , ′NG = ′, NG , ′M = ′, M ,
                        ′L = ′, L , ′PM = ′, PM , ′PL = ′, PL)
```

٤٩–٤

```
PROGRAM Bacteria (INPUT , OUTPUT) ;
```

```
        {Population of a bacteria culture.
         f : is the population multiplication factor at time t}
CONST ALFA = 0.0289 ;
VAR i : INTEGER ;
      t , q , f , term : REAL ;
BEGIN
      READLN (t)
      q := Alfa * t ;
      term := 1.0 ; f := 1.0 ;
      FOR i := 1 TO 9 DO
            BEGIN
                  term := term * q / i ;
                  f := f + term
            END ;
      WRITELN ('Pop. Mul. Factor = ', f)
END.
```

٤–٥٠

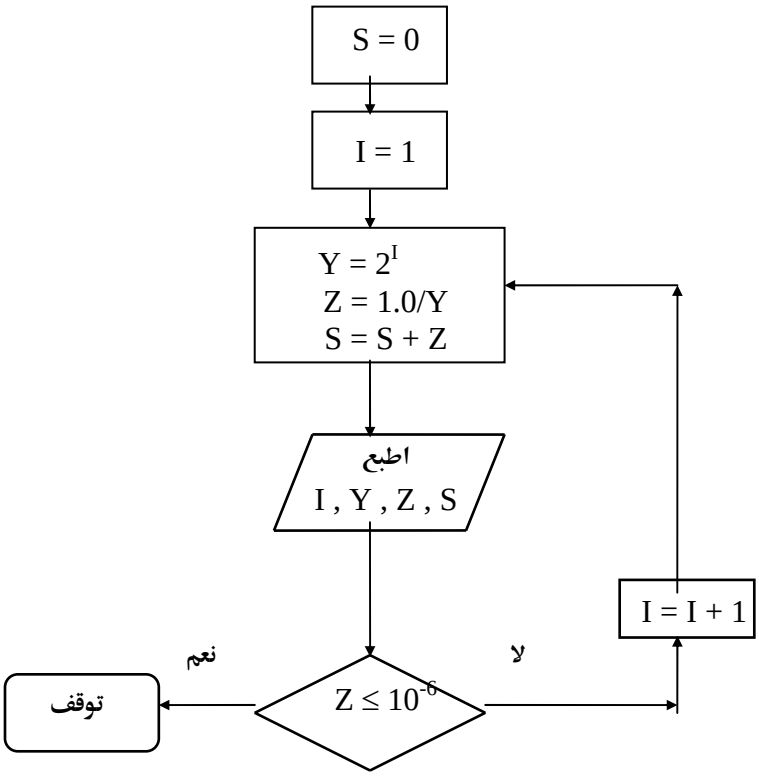
```
PROGRAM Series (INPUT , OUTPUT) ;
      {Evaluating a convergent series with a high efficiency
        (short execution time and small memory space)}
VAR i : INTEGER ;
      x , sum , term : REAL ;
BEGIN
      READLN (x) ;
      sum := 0.0 ; term := 1.0 ; i := 1 ;
      WHILE ABS (term) >= 1E - 5 DO
            BEGIN
                  sum := sum + term ;
                  term := -term * x / i
                  i := i + 1
            END ;
      WRITELN ('G (x) = ', sum)
END.
```

٤–٥١

```
PROGRAM Triangle (INPUT , OUTPUT) ;
      {To determine whether or not three given numbers can
        form the sides of a triangle}
VAR
      A , B , C : REAL ;
BEGIN
      WRITELN ('Enter A , B , C') ;
```

```
        READLN (A , B , C) ;
        REPEAT
            IF (A < B + C) AND (B < A + C) AND (C < A + B)
            THEN WRITELN ('A = ', A , 'B = ', B , 'C = ', C ,
                                'P = ', A + B + C)
            ELSE
                WRITELN ('A = ', A , 'B = ', B , 'C = ', C ,
                                'Not a triangle') ;
            WRITELN ('Enter A , B , C') ;
            READLN (A , B , C)
        UNTIL
            A < 0.0
END.
```

٥٢–٤




```
{CALCULATING POWERS OF TWO, THEIR RECIPROCALS,
AND PARTIAL SUMS OF THE RECIPROCALS}
VAR
    I , Y : INTEGER ;
    S , Z : REAL ;
BEGIN
```

```
        S := 0.0 ;  I := 1 ;  Z := 1.0 ;
        WRITELN ('POWERS      EXPONENTS       ',
                    'FRACTIONS       SUMS') ;
        WHILE  Z > 1E - 6 DO
            BEGIN
                Y := EXP (I * LN (2)) ;
                Z := 1.0 / Y ;
                S := S + Z ;
                WRITELN (I , Y , Z , S) ;
                I := I + 1
            END
END.
```

٥٣–٤

```
PROGRAM Current (INPUT , OUTPUT) ;
        {Current - Frequency Relation in an Electric Circuit}
CONST PI = 3.14159265 ;
VAR   E , L , C , R , W , D , Fmin , Fmax , F , I  : REAL ;
BEGIN
        READLN (E , L , C , R , Fmin , Fmax , D) ;
        F := Fmin ;
        WHILE F <= Fmax DO
            BEGIN
                W := 2 * PI * F ;
                I := E / SQRT (R*R + SQR (W*L - 1.0 / (W*C))) ;
                WRITELN ('F = ', F , 'I = ', I) ;
                F := F + D
            END
END.
```

٥٤–٤

```
Program Positive (input, output);
var
        elt, i, n, np, P : integer;
Begin np := 0; P:= 1;
        writeln ('Enter n');
        readln (n);
        For i := 1 to n do
            begin
                Writeln ('enter an elt');
                readln (elt);
                if elt > 0 then
```

```
                    begin
                        np := np + 1;
                        P := P * elt
                    end
            end ;
        writeln (′np = ′, np);
        writeln (′P = ′ P)
End.
```

٥٥–٤

```
PROGRAM Counting (INPUT, OUTPUT);
VAR
        n, np, nn, nz, m, i : integer;
BEGIN
        np := 0; nn := 0; nz := 0;
        Writeln (′enter the number of elements′);
        Readln (n);
        For i := 1 to n Do
                begin
                        Writeln (′Enter an element′);
                        Readln (m);
                        IF m > 0 then np := np + 1
                        Else if m < 0 then nn := nn + 1
                        Else nz := nz + 1
                end;
        Writeln (′number of positive elts = ′, np);
        Writeln (′number of negative ellts = ′, nn);
        Writeln (′number of zero elts = ′, nz)
END.
```

٥٦–٤

```
Program Marks (INPUT, OUTPUT);
VAR
        m1, m2, m3, m4, avg, large : REAL;
        i :  INTEGER;
BEGIN
        large := 0.0;
        For i := 1 To 30 Do
                begin
                        Writeln (′Enter the 4 marks of a student′);
                        Readln (m1, m2, m3, m4);
                        avg := (m1 + m2 + m3 + m4) / 4.0;
```

```
                Writeln (′avg (′, i : 2, ′) = ′, avg);
                IF avg > large then
                        large := avg
            end;
        Writeln (′highest average mark = ′, large)
END.
```

٥٧–٤

```
Program Odd (input, output);
Var n, i, number, sum : integer;
Begin
        Writeln (′Enter the value of n′) ;
        Readln (n);
        sum := 0;
        For i := 1  To  n  Do
                begin
                        Writeln (′Enter a number′) ;
                        Readln (number ) ;
                        if (number mod 2 = 1) and
                          (number mod 3 = 0) then
                                sum := sum + number
                end;
        Writeln (′sum = ′, sum)
End.
```

٥٨–٤

```
Program Triplets (output);
Var   i , j , k  :  integer;
Begin
        For i := 1  To  9  Do
                For j := 1  To  9  Do
                        For k := 1  To  9  Do
                                if 2*j  =  i + k  then
                                        Writeln (′(′, i , ′ , ′ , j , ′ , ′ , k , ′)′)
End.
```

٥٩–٤

```
Program Temperature (input, output);
Var
        Temp, Avg, i , sum : integer;
Begin sum := 0;
        For i := 1  To  24  Do
                begin
```

```
                Writeln (′Enter a temp′);
                readln (Temp);
                sum := sum + Temp;
            end;
        Avg := round (sum / 24);
        Writeln (′Avg. Temp. = , Avg) ;
        Case Avg of
            -10..9   :  Writeln (′Avg. temp. = ′, Avg, ′ very cold′) ;
             10..19 :  Writeln (′Avg. temp. = ′, Avg, ′ cold′) ;
             20..29 :  Writeln (′Avg. temp. = ′, Avg, ′ mild′) ;
             30..39 :  Writeln (′Avg. temp. = ′, Avg, ′ hot′) ;
             40..59 :  Writeln (′Avg. temp. = ′, Avg, ′ very hot′)
        end ;
End.
```

٦٠–٤

```
Program Temperature (input, output) ;
Var
        i, temp, max, min : integer ;
Begin
        max := -maxint; min := maxint ;
        For i := 1  to  24  do
        begin
            writeln (′Enter a temp′) ;
            readln (temp) ;
            if temp > max then max := temp ;
            if temp < min then min := temp
        end ;
        Writeln (′max. temp. is : ′, max) ;
        Writeln (′min. temp. is : ′, min)
End.
```

٦١–٤

```
PROGRAM PVT (OUTPUT) ;
CONST      a = 0.0299 ;
VAR  V  :  REAL ;
        i, j, P, T : INTEGER ;
BEGIN WRITELN (′P          T          V′) ;
        FOR  i := 30  To  40  DO
            BEGIN
                P := 5 * i ;
                FOR  j := 2  To  10  Do
```

```
                    BEGIN
                        T := 50 * j ;
                        V := a * (T + 460) / P ;
                        WRITELN (P, T, V)
                    END
            END
END.
```

٤-٦٢ أ)

```
Program CarValues (input, output) ;
Var  i :  integer ;
     value  :  real ;
Begin
     Writeln (′Enter initial value′) ;
     readln (value) ;
     For i := 1  to  10  do
          begin
               value := 0.8 * value ;
               writeln (′year = ′, i : 3, ′value = ′, value : 10 : 3)
          end
End.
```

ب)

```
Begin
     Writeln (′Enter initial value′) ;
     readln (value) ;
     i := 0 ;
     repeat
          i := i+ 1;
          value := 0.8 * value
          writeln (′year = ′, i : 3, ′value = ′, value : 10:3)
     until  value < 700.0
End.
```

٤-٦٣

```
PROGRAM NumberType (INPUT, OUTPUT);
VAR
     N, i, sum : INTEGER;
BEGIN
     WRITELN (′ENTER AN INTEGER NUMBER′) ;
     READLN (N) ;
     WRITELN (′Divisors of N are′) ;
     sum := 0.0 ;
```

```
        For i := 1  To  N Div  2  Do
                IF  N mod i = 0 THEN
                        BEGIN
                                WRITELN (i) ;
                                sum := sum + i
                        END ;
        WRITELN (′sum of divisors = ′, sum);
        IF n = sum THEN
                        WRITELN (′N is a perfect number′)
        ELSE IF n < sum THEN
                        WRITELN (′N is a deficient number′)
        ELSE
                        WRITELN (′N is an abundant number′)
END.
```

٦٤-٤

```
Program Word Count (input, output) ;
Const
        Space = ′ ′;
        FullStop = ′ . ′ ;
Var
        NextChar  :  char ; {character input from keyboard}
        Words  :  integer ; { number of words in sentence}
begin
        Words := 1 ; {initialise word count to 1}
        Writeln (′input sentence - terminate with full stop′);
        read (NextChar);
        while NextChar <> Fullstop do
                begin
                        {count number of spaces between words}
                        if NextChar  =  space then
                                words := words + 1;
                        {end if}
                        read (NextChar)
                end;
        Writeln;
        Writeln (′number of words in sentence ⇒ ′, words : 3)
end.
```

ملاحظة : نتائج تشغيل البرنامج :

```
input sentence - terminate with full stop
Be in the world as though you were a stranger or a wayfarer.
```

number of words in sentence ⇒ 13

٦٥-٤

```
Program Isotope ;
Const a = 50 ;  H = 28 ;
Var
        year : integer ;
        r      : real ;
Begin
        For year := 1 To 50 Do
                begin
                        r := a * Exp (-0.693 * year / H) ;
                        writeln (year, r)
                end
End.
```

٦٦-٤

```
Program MathRelation ;
Var    i, n, S, L, R : integer ;
Begin
        For n := 10 to 20 do
                begin
                        S := 0; R := 0 ;
                        For i := 1  to  n  do
                                begin
                                        S := S + i ;
                                        R := R + i * i * i
                                end ;
                                L := S * S ;
                                Writeln ('L= ', L , 'R = ', R) ;
                                If L = R then
                                Writeln ('For n= ', n, 'relation is true')
                end
End.
```

حل آخر أكثر كفاءة :

```
Begin S := 0 ; R := 0 ;
        For i := 1  to  9  do
                begin
                        S := S + i ;
                        R := R + i * i * i
                end ;
                For i := 10  to  20  do
```

```
            begin
                S := S + i ;
                R := R + i * i * i ;
                L := S * S ;
                Writeln (′L = ′, L , ′ R = ′, R) ;
                If L = R then
                    Writeln (′For n = ′, i , ′relation is true′)
            end
End.
```

٦٧–٤

```
Program Zakat (input, output) ;
Var
    S, P, A, Z, TZ : real; n, i : integer ;
Begin
    Writeln (′Enter P, S, n′) ;
    readln (P, S, n) ;
    A := 85 * P ;
    TZ := 0.0 ;
    For i := 1 to n do
        begin
            if  S > = A then
                Z := S/40.0
            else
                Z :=  0.0 ;
            S := S - Z ;
            TZ := TZ + Z ;
            writeln (′i = ′, i , ′Z = ′, Z , ′S = ′, S)
        end ;
    Writeln (′TZ = ′, TZ)
End.
```

٦٨–٤

```
Program SelfReckoning ;
Var
    i, Number, Count : integer ;
    Answer : Char ;
Begin
    Count := 0 ;
    For i := 1  To 15 do
        begin
            Writeln (′Enter Number, Answer′) ;
```

```
                readln (Number, Answer) ;
                if Answer = 'Y' then
                        Count := Count + 1
        end ;
    Writeln ('Count = ', Count)
End.
```

٤–٦٩

```
program Measurements, (input, output);
    {This program finds the percentage of measurements
    that are greater than 10.0}
Var
    value, count : real ;
    i : integer ;
begin
    count := 0.0 ;
    for i := 1  to  50  do
        begin
            readln (value) ;
            if value > 10.0 then
                count := count + 1.0
        end ;
    value :=  (count / 50.0) * 100.0 ;
    writeln (value: 5: 1,  PERCENT OF THE MEASUREMENTS
          ARE GREATER THAN 10.0')
end.
```

٤–٧٠

```
program series (input, output) ;
    {This program approximates sin(x) using an infinite series.}
Var
    sine, x, term, index: real ;
    i : integer ;
begin
    x := 2.5 ;
    term := x ;
    sine := term ;
    index := 2.0 ;
    for i := 2  to  20  do
        begin
            if not odd (i) then
                begin
```

```
                                term :=
term*sqr(x)/(index*(index+1.0));
                                sine := sine + term
                            end ;
                        index := index + 1.0
                end ;
        Writeln ;
        Writeln (′THE APPROIXMATION OF SINE (2.5) IS , sine)
end.
```

٧١–٤

```
Program Numbers ;
{
        This program accepts numbers from the user and finds the
largest, smallest, the average, range, variance, and standard deviation
of the numbers typed.
}
Var
        N,                  {input    - number of data items}
        Number,             {input    - data item}
        Largest,            {output -  largest value}
        Smallest,           {output -  smallest value}
        Range : Integer;    {output -  range of values}
        Variance,           {output -  variance of data}
        StdDev,             {output -  standard deviation}
        Average : Real;     {output -  average value}
        I,                  {loop control}
        Sum,                {sum of values}
        SumSqr : Integer;{sum of squared values}
begin {Numbers}
{Get N, make sure it′s positive.}
Write (′enter the number of values to be read: ′) ;
ReadLn (N) ;
while N < = 0 do
        begin
                WriteLn (′The number must be positive. ′) ;
                Write (′Enter the number of values to be read: ′) ;
                ReadLn (N)
        end; {while}
{Get first value, initialize counters}
Sum := 0 ;
SumSqr := 0 ;
```

```
Largest := -MaxInt ;
Smallest := MaxInt ;
{Get the N values, update Largest, Smallest, Sum}
for I := 1  to  N  do
        begin
                {Get next value}
                Write (′Enter a value: ′) ;
                ReadLn (Number) ;
                {Check for new max or min}
                if Number > Largest then
                        Largest := Number ;
                if Number < Smallest then
                        Smallest := Number ;
                Sum := Sum + Number ;
                SumSqr := SumSqr + Sqr (Number) ;
        end; {for}
{Compute statistics and print the results}
Average := Sum / N ;
Range :=  Largest - Smallest ;
Variance := SumSqr - Sqr (Sum) / N ;
StdDev := SqRt (Variance / (N - 1)) ;
WriteLn ;
WriteLn    (′Largest value:          ′,    Largest : 6) ;
WriteLn    (′Smallest value:         ′,    Smallest : 6) ;
WriteLn    (′Range of values:        ′,    Range : 6) ;
WriteLn    (′Average value:          ′,    Average : 6: 2) ;
WriteLn    (′Variance:               ′,    Variance : 6: 2) ;
WriteLn    (′Standard Deviation      ′,    StdDev : 6: 2) ;
WriteLn
end. {Numbers}
```

٧٢–٤

```
program LumberTable ;
{
Calculates the engineering properties of lumber and displays the
information in a  table.
}
Var
        Base,                   {width of board in inches}
        Height  :  Integer;     {length of board in inches}
        Cross,                  {cross sectional area}
        Moment,                 {moment of inertia}
```

```
        Section  :  Real ;   {section modulus}
begin {LumberTable}
        {Display table headings}
        WriteLn (′Engineering Properties of Lumber ′ : 50) ;
        WriteLn (′Board′ :15, ′Cross′ :15, ′Moment′ :15,′Section′ :15) ;
        WriteLn (′(inches) ′ : 15, ′Sectional′ : 15, ′Of Inertia′ : 15,
                                        ′Modulus′ : 15);
{Calculate and display properties for each board}
Height := 2;
while Height <= 10 do
        begin
                Base := Hieght;
                while Base <= 12 do
                        begin
                                Write(Base : 8, ′-by-′, Hieght) ;
                                Cross := Base * Height ;
                                Section := Cross * Height / 6 ;
                                Moment := Section * Height / 2 ;
                                WriteLn (Cross : 15: 2, Moment : 15: 2,
                                        Section : 15: 2);
                                Base := Base + 2 {next board Height}
                        end;  {while}
                WriteLn;
                Height := Height + 2            {next board Height}
        end;  {while}
end.  {LumberTable}
```

٧٣–٤

```
program FindTheNumber ;
{
This program finds the position of the first and last occurrence in a
sequence of integer input values of a target value. If the target value
is not present, 0 is displayed as position. Input terminated by sentinel
-1.
}
const
        Target  =  12;      {The number to search for}
        Sentinel  =  -1;    {Terminates data input}
var
        Item,               {input    - the input values}
```

```
        FirstIndex,         {output - index of first Target}
        LastIndex,          {output - index of last Target}
        Index  :  Integer;  {position of current Item}
begin {FindNumber}
        {Initialize loop variables}
        FirstIndex := 0 ;
        LastIndex := 0 ;
        Index := 1 ;
        {Get first Item}
        Write (′Enter first item (′, Sentinel : 1, ′ to quit ) : ′);
        ReadLn (Item) ;
        {Process each Item until sentinel}
        while Item <> Sentinel do
                begin
                        {
Check current Item for target, update LastIndex, but update First
Index only if no prior target values were found
                        }
                        if Item  =  Target then
                                begin
                                        LastIndex := Index ;
                                        if FirstIndex = 0 then
                                                FirstIndex := Index
                                end; {if Item = Target}
                        {Get next item, update Index}
                        Write (′Enter next item (′, Sentinel : 1, ′ to quit) : ′);
                        ReadLn (Item) ;
                        Index := Index + 1
                end; {while}
        if FirstIndex = 0 then
                WriteLn (′The value ′, Target : 1, ′was not entered, ′,
                                ′ LastIndex = ′, LastIndex : 1, ′,
                                FirstIndex = ′, FirstIndex : 1, ′.′)
        else
                begin
                        WriteLn (′The value ′, Target : 1, ′first occurred′,
                                        ′ at position ′, FirstIndex : 1);
                        WriteLn (′and last occurred at position ′,
                                        LastIndex : 1, ′.′)
                end {else}
end. {FindTheNumber}
```

```
program Grades;
{
        This program reads a collection of exam scores in the range 1
to 100, and counts        and prints the number of scores in the ranges
90 - 100 (outstanding) , 60 - 89 (satisfactory), and 1-59
(unsatisfactory). It also displays the average exam score. As each
score is read, the category of each score is displayed. 0 is used as a
sentinel for input.
}
const
        OutstMin = 90 ; {cutoff  for  outstanding  score}
        SatisMin  = 60 ; {cutoff  for  satisfactory  score}
        Sentinel   = 0 ; {terminates input of scores}
var
        Score,                  {input - test scores}
        NumOutst,               {output - # outstanding}
        NumSatis,               {output - # satisfactory}
        NumUnsat : Integer;     {output - # unsatisfactory}
        AveScore : Real;        {output - average score}
        NumScores : Integer;    {number of scores}
        SumScores : Real;       {sum of scores}
begin {Grades}
        {Initialize counters and accumulator}
        NumOutst  := 0 ;
        NumSatis   := 0 ;
        NumUnsat  := 0 ;
        NumScores := 0 ;
        SumScores := 0.0 ;
        {Get first score}
        Write (′Enter first score (′, Sentinel : 1, ′ to quit ) : ′);
        ReadLn (Score) ;
        {Process scores until Sentinel is encountered}
        while Score <> Sentinel do
                begin
                        {Update totals}
                        NumScores := NumScores + 1 ;
                        SumScores := SumScores + Score ;
                        {
        Update appropriate category counter and write category of
        score.
```

```
                    }
                    if Score >= OutstMin then
                        begin
                            WriteLn (′Outstanding′: 20) ;
                            NumOutst := NumOutst + 1
                        end {if outstanding}
                    else if Score >= SatisMin then
                        begin
                            WriteLn (′Satisfactory′: 20) ;
                            NumSatis := NumSatis + 1
                        end {else if satisfactory}
                    else
                        begin
                            WriteLn (′Unsatisfactory′ : 20) ;
                            NumUnsat := NumUnsat + 1
                        end ; {else unsatisfactory}
                    WriteLn ;
                    {Get next score}
                    Write (′enter next score (′, Sentinel : 1, to quit): ′);
                    ReadLn (Score)
                end ; {while}
        {
        Check for empty input; if nonempty, compute average and
        display results.
        }
        if NumScores = 0 then
            WriteLn (′No scores were entered. ′)
        else
            begin
                AveScore  :=  SumScores / NumScores ;
                WriteLn;
                WriteLn (′Number Outstanding: ′, NumOutst : 6) ;
                WriteLn (′Number Satisfactory: ′, NumSatis : 6) ;
                WriteLn (′Number Unsatisfactory: ′, NumUnsat: 6)
;
                WriteLn ;
                WriteLn (′Average Score: ′, AveScore : 6: 2) ;
                WriteLn
            end {else}
end. {Grades}
```

```
program Payroll ;
{
This program processes the payroll for a company.
INPUTS :
        for each employee, his or her id number, hours worked, and
        hourly wage rate. An id of zero terminates input.
OUTPUTS:
        For each employee, id number and net pay. Total payroll and
        average amount paid.
SPECIFICATION:
        Net pay is computed based on time-and-a-half for hours worked
        over 40, and a deduction of 3.625 percent tax on gross salary.
}
const
        OTLimit =  40;          {Overtime paid for excess}
        OTRate  =  1.5;         {Rate for overtime}
        TaxRate =  0.03625;     {Tax rate}
        Sentinel =  0;          {Terminates input}
var
        ID : Integer ;                  {input - employee id number}
        Hours ,                 {input - empl. hours worked}
        Rate ,              {input - empl. hourly wage}
        NetPay ,                {output - empl. net salary}
        TotalGross ,            {output - total gross payroll}
        AveNet : Real ;         {output - average amount paid}
        NumEmpl : Integer ;     {number of employees}
        TotalNet : Real ;       {total net payroll}
        GrossPay : Real ;       {employee gross pay}
begin {Payroll}
        {Initialize counters and accumulators}
        TotalNet    := 0.0 ;
        TotalGross := 0.0 ;
        NumEmpl  := 0 ;
        {get first employee data}
        Write (′Enter employee id (′,Sentinel : 1, ′ to quit) : ′) ;
        ReadLn (ID) ;
        {Process employees}
        while ID <> Sentinel do
                begin
                        {Read input data for current employee}
```

```
                Write (′enter hours worked for employee′,
                        ID:1, ′: ′) ;
                ReadLn (Hours) ;
                Write (′Enter hourly wage rate for employee ′,
                        ID : 1, ′: ′);
                ReadLn (Rate) ;
                {Find gross and net pay for this employee}
                if Hours > OTLimit then
                    GrossPay := Rate * 40 +
                        OTRate * Rate * (Hours - OTLimit)
                else
                    GrossPay := Rate * Hours ;
                NetPay := GrossPay - GrossPay * TaxRate ;
                WriteLn (′Net pay for employee ′, ID : 1, ′ is $′,
                        NetPay : 4: 2) ;
                WriteLn;
                {update counters and accumulators}
                TotalNet := TotalNet + NetPay ;
                TotalGross := TotalGross + GrossPay ;
                NumEmpl := NumEmpl + 1 ;
                {Get next id}
                Write (′Enter next employee id (′, Sentinel : 1,
                    ′ to quit): ′) ;
                ReadLn (ID)
            end; {while}
            {Compute average and print summary data}
            if  NumEmpl = 0 then
                WriteLn (′No employees were processed. ′)
            else
                begin
                  AveNet  :=  TotalNet / NumEmpl ;
                  WriteLn ;
                  WriteLn (′Total gross payroll: ′,TotalGross:7:2)
;
                  WriteLn  (′Average net pay: ′, AveNet:7:2) ;
                  WriteLn
                end {else}
end. {Payroll}
```

٧٦–٤

```
program Inventory ;
```

```
{
This program updates inventory records in a distributing company.
INPUTS
        Initial inventory for each of 4 items. Each item is coded by a
        number: 1, 2, 3, and 4. Each sale or purchase is read, a purchase
        amount being positive and a sale being negative. A zero code
        terminates input.
OUTPUTS
        Final inventory for the week.
ASSUMPTION
        Inventory is always sufficient to cover all sales for the week.
}
const
        Sentinel = 0;       {terminates input}
var
        ID,                 {input - item code}
        Transaction,        {input - sale or purchase amount}
        It1Inv,             {input / output - Item 1 inventory}
        It2Inv,             {input / output - Item 2 inventory}
        It3Inv,             {input / output - Item 3 inventory}
        It4Inv              {input / output - Item 4 inventory}
                            : Integer;
begin {Inventory}
        {Read initial inventories}
        Write (′Starting inventory for Item 1: ′) ;
        Readln (It1Inv) ;
        Write (′Starting inventory for Item 2: ′) ;
        Readln (It2Inv) ;
        Write (′Starting inventory for Item 3: ′) ;
        Readln (It3Inv) ;
        Write (′Starting inventory for Item 4: ′) ;
        Readln (It4Inv) ;
        writeLn ;
{Process transactions}
Write (′Enter ID of transaction (′,Sentinel:1, ′ to quit) : ′) ;
ReadLn (ID) ;
while ID <> Sentinel do
        begin
                {Get transaction amount}
                Write (′Enter transaction amount: ′) ;
                readLn (Transaction) ;
```

```
            {Process transaction, ingnoring illegal ID's}
            if ID = 1 then
                  It1Inv := It1Inv + Transaction
            else if ID = 2 then
                  It2Inv := It 2 Inv + Transaction
            else if ID = 3 then
                  It3Inv := It3Inv + Transaction
            else if ID = 4 then
                  It4Inv := It4Inv + Transaction
            else
                  WriteLn ('Invalid Code, transaction ignored. ') ;
            WriteLn ;
            {Get next item ID}
            Write ('Enter item ID (', Sentinel : 1, ' to quit): ') ;
            ReadLn (ID)
      end; {while}
      {Display final inventories}
      WriteLn ;
      WriteLn (' End - of -Week Inventories') ;
      WriteLn ('--------------------------------') ;
      WriteLn ('Item 1     ', It1Inv : 8) ;
      WriteLn ('Item 2     ', It2Inv : 8) ;
      WriteLn ('Item 3     ', It3Inv : 8) ;
      WriteLn ('Item 4     ', It4Inv : 8)
end. {Inventory}
```

٧٧–٤

```
program Inventory ;
{
A menu driven program updates inventory records for a distributing
company
}
var
      Choice : Char;    {input - menu selection}
      ID,               {input - item code}
      Transaction,      {input - sale or purchase amount}
      It1Inv,           {input / output - Item 1 inventory}
      It2Inv,           {input / output - Item 2 inventory}
      It3Inv,           {input / output - Item 3 inventory
      It4Inv            {input / output - Item 4 inventory}
            : Integer ;
```

```
begin {Inventory}
        repeat
{Display program menu choices to the user.}
        WriteLn (′ INVENTORY ANALYZER: ′) ;
        WriteLn (′(E)nter Inventory′) ;
        WriteLn (′(P)urchase an item′) ;
        WriteLn (′(S)ell an item′) ;
        WriteLn (′(D) isplay Inventory′) ;
        WriteLn (′(Q)uit Program′) ;
        WriteLn ;
        Write (′Type first letter of your choice > ′)
ReadLn (Choice) ;
WriteLn ;
if (Choice = ′E′) or (Choice = ′e′) then
        begin
                Write (′Starting inventory for Item1: ′) ;
                Readln (It1Inv) ;
                Write (′Starting inventory for Item2: ′) ;
                Readln (It2Inv) ;
                Write (′Starting inventory for Item3: ′) ;
                Readln (It3Inv) ;
                Write (′Starting inventory for Item4: ′) ;
                Readln (It4Inv) ;
                WriteLn
        end {Enter Inventory}
else if (Choice = ′P′) or (Choice  = ′p′) then
        begin
                Write (′enter ID of item to purchase > ′) ;
                ReadLn (ID) ;
                Write (′enter amount purchased > ′) ;
                ReadLn (Transaction) ;
        {Process transaction, ignoring illegal ID′s}
                if ID = 1 then
                        It1Inv := It1Inv + Transaction
                else if ID = 2 then
                        It2Inv := It2Inv + Transaction
                else if ID = 3 then
                        It3Inv := It3Inv + Transaction
                else if ID = 4 then
                        It4Inv := It4Inv + Transaction
```

```
                else
                        WriteLn (′Invalid Code, transaction ignored. ′) ;
                WriteLn ;
        end {Purchase an item}
else if (Choice = ′S′) or (choice = ′s′) then
        begin
                WriteLn ;
                Write (′Enter ID of item to sell > ′) ;
                ReadLn (ID) ;

                Write (′Enter amount sold > ′) ;
                ReadLn (Transaction) ;
                {Process transaction , ignoring illegal ID′s }
                If ID = 1 then
                        It1Inv := It1Inv - Transaction
                else if ID = 2 then
                        It2Inv := It2Inv - Transaction
                else if ID = 3 then
                        It3Inv := It3Inv - Transaction
                else if ID = 4 then
                        It4Inv := It4Inv - Transaction
                else
                        WriteLn (′Invalid Code , transaction ignored.′) ;
                WriteLn ;
        end {Sell an item}
else if (Choice = ′D′) or (Choice = ′d′) then
        begin
                WriteLn ;
                WriteLn (′ End-of-Week Inventories′) ;
                WriteLn (′----------------------------′) ;
                WriteLn (Item1     ′ , It1Inv : 8) ;
                WriteLn (Item2     ′ , It2Inv : 8) ;
                WriteLn (Item3     ′ , It3Inv : 8) ;
                WriteLn (Item4     ′ , It4Inv : 8) ;
                WriteLn
        end {Display Inventory}
else if (Choice = ′Q′) or (Choice = ′q′) then
        WriteLn (′Program execution terminated. ′)
else
        begin
```

```
            WriteLn  (′Bad menu code. ′) ;
            WriteLn
        end
until (Choice = ′Q′) or (Choice = ′q′)
end. {Inventory}
```

٧٨–٤

```
program Savings ;
{
This program reads weekly transactions for all savings accounts and
prints a summary report.
INPUT FORMAT :
Each savings account is represented as follows:
        Line 1:     I                 --  Column 1
                    Account Num       --  Columns 3-9
                    Starting Balance  --  Columns 11-20
        Line 2, 3, ..
        (possibly none if no transactions for the week)
                    Transaction type      --  Column 1
                    Transaction amount    --  Columns 11-20
        Last Line:  Sentinel ′′Z′′        --  Column 1
OUTPUT:
A summary table showing the account number and the balance
following each transaction. If the account balance becomes negative
after any transaction that fact is noted, and the transaction is
processed.
Note: A bad transaction code will cause a data line to be discarded.
}
const
        Max = 6;                {# accounts to process}
var
        Code : Char;            {input - transaction code}
        I,                      {loop control}
        Account : Integer ;     {input / output -  account number}
        Amount,                 {input - transaction amount}
        Balance : Real ;        {input / output - account balance}
begin {Savings}
        {Display table heading}
        WriteLn (′ SAVINGS ACCOUNT WEEKLY SUMMARY′);
        for I := 1 to Max do
            begin
                {Get initial data}
```

```
            WriteLn;
            WriteLn (′Enter initial account data. ′) ;
            Read (Code) ;
            {Discard bad records.}
            while (Code <> ′I′)do
                  begin
                        ReadLn;
                        WriteLn (′Bad initial data, re-enter. ′) ;
                        Read (Code)
                  end; {while}
            {Get account number and initial balance.}
            ReadLn (Account, Balance) ;
            WriteLn (′Account: ′, Account : 6,
                  ′ Balance: ′, Balance : 4: 2);
{Process remaining transactions.}
repeat
      WriteLn ;
      WriteLn (′Enter next transaction record. ′) ;
      Read (Code) ;
      if Code in [′D′, ′W′, ′Z′] then
            begin
                  case Code of
                        ′D′ :  begin
                                    ReadLn (Amount) ;
                                    Balance := Balance +Amount
                                    end; {Deposit}
                        ′W′ :  begin
                                    ReadLn (Amount) ;
                                    Balance := Balance - amount
                                    end; {Withdrawal}
                        ′Z′ :  begin
                                    ReadLn;
                                    WriteLn(′No more changes.′)
                                    end {Last Transaction}
                  end; {case}
                  {Display account balance}
                  WriteLn (′ ′ : 14, ′Balance: ′, Balance : 4: 2) ;
                  {Check for negative account balance}
                  if Balance < 0 then
                        WriteLn (′Your balance is negative. ′) ;
                  WriteLn
```

```
                end {if}
            else
                begin
                    WriteLn ('Bad transaction code. ') ;
                    ReadLn
                end {else}
            until code = 'Z'
        end {for}
end. {Savings}
```

٧٩–٤

```
program TrackSpeed ;
{
This program computes speeds for a runner in a mile race. It reads the
time in minutes and seconds, and then computes and prints the speed
in feet per second and meters per second.
}
const
    FtPerMile   = 5280 ;      {Conversion - feet per mile}
    FtPerKm     = 3282 ;      {Conversion - feet per km}
    SecPerMin = 60 ;          {Conversion - Seconds per min}
    MetersPerKm =   1000 ; {Conversion - meters per km}
var
    MinuteTime,               {input - runner's time (minutes)}
    SecondTime,               {input - runner's time (seconds)}
    Time,                     {interm - total time in seconds}
    SpeedFPS,                 {output - speed in feet per sec}
    SpeedMPS : Real;          {output - speed in meters per sec}
begin {TrackSpeed}
    Write ('Enter the runner ''s time in minutes and ') ;
    Write (' seconds : ') ;
    Readln (MinuteTime ; SecondTime) ;

    { Compute the time in seconds and the two speeds }
    Time := SecPerMin * MinuteTime + SecondTime ;
    SpeedFPS := FtPerMile / Time ;
    SpeedMPS := (SpeedFPS / FtPerKm) * MetersPerKm ;

    { Display the results }
    WriteLn ('If a runner finishes a mile run in ' ,
                MinuteTime:3:1, ' minutes and ',
                SecondTime:4:2, ' seconds, ') ;
```

```
        Write ('his or her speed was ', SpeedFPS:4:2,
                    ' feet per second, or ') ;
        Writeln (SpeedMPS:4:2, ' meters per second. ')
end. {TrackSpeed}
```

أجوبة تمرينات الفصل الخامس

٥-١

(	?	7	6	5	4	3	2	1
(	1	4	9	-2	-2	-2	-2	-2
(	0	?	-1	?	-2	?	?	?

٥-٢

20 19, 18, 17, 16, 15, 10, 10, 6, 10,

٥-٣ عبارات الإسناد

A [1] := 1 ;
A [2] := 2 ;
A [3] := 3 ;
A [4] := 4
A [2] := A [2] + A [1] := 2+1 = 3 ;
A [3] := A [3] + A [2] := 3+3 = 6 ;
A [4] := A[4] + A[3] := 4 + 6 = 10

المخرجات

A [1] = ∇∇ 1
A [2] = ∇∇ 3
A [3] = ∇∇ 6
A [4] = ∇ 1 0

٥-٤

k [1] = 5
k [2] = 10
k [3] = 15
k [4] = 20
k [5] = 15

٥-٥ عبارات الإسناد

A [1] := 1
A [4] := -1
A [2] := 3
A [5] := 2
A [3] := 5
A [6] := 7
A [1] := A [4] - A [1] = -1-1 = -2 ;
A [2] := A [5] - A [2] = 2-3 = -1 ;
A [3] := A [6] - A [3] = 7-5 = 2

المخرجات

```
for   i = 1      A = -2
for   i = 2      A = -1
for   i = 3      A = 2
for   i = 4      A = -1
for   i = 5      A = 2
for   i = 6      A = 7
```

٥-٦

a) var A : array [1..8] of integer ;

b) var NewArray : array [1..10] of real ;

c) for j := 1 to N do score [j] := 0 ;

d) **ليس بها أخطاء**

e) .....

```
for row := 1 To 5 Do
 for col := 1 To 6 Do
.....
```

٥-٧

```
Program Series ;
Var   i : integer ;
      X,Y : array [1..20] of integer ;
      sum : real ;
Begin        sum := 0.0 ;
      For i := 1  To  20  Do  begin
            readln (X[i], Y[i]) ;
            sum := sum + X[i] / Y[i]
            end ;
      Writeln (sum)
End.
```

٥-٨ التصحيح :

```
Until (Continue <> 'Y' ) or (Counter  >  10)
```

العبارة المطلوبة :

```
While       (Continue  =  'Y') and (Counter  <=  10) do
      begin
            Write('Enter a name for the list : ') ;
            .............
            .............
            readln (Continue)
                                                end
```

٥–٩ (أ)

```
sum := 0.0  ;
FOR i := 1 TO 6 DO
sum := sum + G[i , 3] ;
```

(ب)

```
FOR j := 1 TO 9 DO
G[3 , j] := G[1 , j] + G[2 , j] ;
```

(ج)

```
FOR j := 1 TO 9 DO
G[4 , j] := 0.0 ;
```

٥–١٠ (أ)

```
FOR i := 1 TO 15 DO
Diag [i] := MAT [i , i]
```

(ب)

```
PR := 1.0 ;
FOR i := 1 TO 15 DO
PR := PR * Diag [i] ;
```

٥–١٣ انظر حل مثال ٥ – ١

٥–١٤

```
sum := 0.0 ;
FOR i := 0 TO 49 DO
sum := sum + SQR (A[2 * i + 1) ;
S  =  SQRT (sum) ;
WRITELN (′S =′, S ) ;
```

٥–١٥

```
j := 1
while (A[j] >= 0) and (j <= 40) do
j := j + 1 ;
if j <= 40 then
i := j
else
i := 0 ;
writeln (′i = ′, i) ;
```

حل آخر

```
var
```

```
found :  boolean ;
--------
--------
begin
found :=  false ;
j :=  1 ;
While ( j <= 40 ) and (not found) do
if  A[ j] < 0 then
found := true
else
j := j + 1 ;
if found then
i := j
else
i := 0 ;
Writeln (′i = ′ , i) ;
```

١٦–٥

```
READLN (N)  ;
FOR i  := 1 TO N DO
READ ( Y [i])  ;
sum := 0.0 ;
FOR i :=  1  TO  N  DO
sum  := sum + Y [i]  ;
Avg  := sum / N ;
{ calculate the standard deviation }
sum  :=  0.0 ;
FOR i := 1 TO N DO
sum := sum + SQR ( Y[i]) ;
SD := SQRT (sum / N - SQR (Avg)) ;
WRITELN (′Avg = ′, Avg , ′SD = ′, SD)
```

١٧–٥

```
smsqrt := 0.0 ;
FOR i := 1 TO 50 DO
smsqrt := smsqrt + SQRT(A[i]) ;
WRITELN (′smsqrt = ′, smsqrt)
{Find largest element}
Amax := A[1] ;
FOR i := 2 to 50 DO
IF Amax < A[i] THEN Amax := A[i] ;
WRITELN (′largest element = ′, Amax) ;
```

```
                    {Normalized Values}
                    FOR i := 1 TO 50 DO
                        BEGIN
            B[i] := A[i] / Amax ;
WRITELN ('B[' , i : 2 , '] = ', B[i])
                        END ;
```

۱۸–۵

```
                    Big := A[1] ; N := 1 ;
                   FOR i :  = 2 TO 45 DO
        IF ABS (BIG) < ABS (A[i]) THEN
                        BEGIN
        BIG := A[i] ; N := i
                        END ;
       WRITELN ('BIG = ', BIG , 'N = ' , N)
```

۱۹–۵

```
                         S :=  0.0 ;
                      FOR i := 1 TO 5 DO
                   S := S + Score [i] ;
                      small := Score [1] ;
                     FOR  i := 2 TO 5 DO
            IF small > Score [i] THEN
             small :=  Score [i] ;
                    Av := (S - small) / 4.0  ;
             WRITELN ('Average  = ', AV)
```

۲۰–۵

```
                    FOR i := 1 TO 24 DO
                FOR j := 1 TO 40 DO
            IF L [i] = M[j] THEN
          WRITELN (L [i]) ;
```

۲۱–۵

```
PROGRAM Polynomial (INPUT , OUTPUT) ;
                        VAR
                  i  , n : INTEGER  ;
          A     : array [0..15] OF REAL ;
                    x , y : REAL ;
                        BEGIN
                    READLN (n) ;
FOR i :=  0 TO n DO READ (A[i]) ;
                    READLN (x) ;
```

```
                    y :=  A[0] ;
                FOR i := 1 TO n DO
        y := y + A[i] * EXP (i * LN (x))  ;
            WRITELN (′x = ′, x , ′y = ′, y)
                                END.
```

٢٢–٥

```
                    sum := 0.0  ;
                FOR i := 1 TO 500 DO
            sum := sum +  D [i] ;
                        BIG := D [1] ;
                FOR i := 2 TO 500  DO
        IF BIG < D [i] THEN BIG := D [i] ;
                                N := 0 ;
                FOR i := 1 TO 500 DO
        IF D [i] > 100.0 THEN N := N + 1 ;
WRITELN (′sum = ′, sum, ′BIG = ′, BIG, ′N = ′, N)
```

٢٣–٥

```
{sum of products of corresponding elements of odd order}
                    S := 0.0 ; i := 1 ;
                WHILE i <= N DO
                        BEGIN
            S := S + A [i] * B[i] ;
                        i := i + 2
                        END ;
        {Composite Average of Absolute Values}
                    sum := 0.0 ;
                FOR i := 1 TO N DO
    sum := sum + ABS(A[i] ) + ABS (B[i]) ;
                    AV := sum / (2.0 * N) ;
        {Newton's Formula for computing xnew}
                    PX := A[1] ;
                FOR i := 1 TO (N-1) DO
    PX := PX + A[i+1] * EXP (i * LN(X)) ;
                    PDX := 0.0 ;
                FOR i := 1 TO (N-1) DO
PDX := PDX + i * A[i+1] * EXP((i-1) * LN(X)) ;
                    xnew := X - PX / PDX ;
WRITELN (′S = ′, S , ′AV = ′, AV, ′xnew = ′, xnew)
```

٢٤–٥

```
                                        {Islamic Bank Program}
                          ND := 0 ;  NL := 0 ; TD := 0.0 ; TL := 0.0 ;
                                          FOR j := 1 TO 1000 DO
                                        IF A [j] > 0.0 THEN
                                              BEGIN
                                    ND := ND + 1 ;
                                  TD :=  TD + A [j]
                                              END
                                                   ELSE
                                              BEGIN
                                    NL :=  NL + 1 ;
                                    TL :=  TL - A[j]
                                              END ;
WRITELN (′ND =′, ND,′ NL = ′, NL , ′TD = ′, TD,′TL = ′,TL)
                                        {Compute Profits / losses}
                                          FOR j := 1 TO 1000 DO
                                                   BEGIN
                                        IF A[j] > 0.0 THEN
                                  S[j] := TPL * A[j] / TD
                                              ELSE
                                    S[j] := 0.0 ;
  WRITELN (′A[′, j : 4,′] = ′, A[j] , ′S[′ , j : 4,′] = ′, S[j])
```

٢٥–٥

```
                                        {Bubble Sort Algorithm}
                                              READ (n) ;
                          FOR i := 1 TO n DO READ (a [i])  ;
                                          m := n ;  test := 1 ;
                                While (m <> 1) and (test <> 0) do
                                              BEGIN
                                        test := 0 ;
                                FOR j := 1 TO m-1 DO
                          IF a [j + 1] < a [j] THEN
                                        BEGIN
                          temp := a [j] ;
                        a [j] := a [j+1] ;
                      a [j +1] := temp ;
                                test :=  1
                                        END ;
                                          m := m - 1
                                              END ;
                              FOR i := 1 TO n DO WRITELN (a[i])
```

ملاحظـــة :

المتغير test **يأخذ القيمة صفرا** test = 0 **في بداية كل مرحلة ، وإذا كانت قيمته** =1 **في نهاية أي مرحلة فهذا يعني أنه قد تمت عملية تبديل (أو أكثر) بين عنصرين من عناصر المجموعة في هذه المرحلة ، أما إذا كانت قيمته صفرا في نهاية المرحلة فهذا يعني أنه لم تحدث أي عملية تبديل في هذه المرحلة ، وبالتالي فلا داعي للانتقال إلى مرحلة تالية ، بل تتوقف المقارنات لأن جميع العناصر قد تم ترتيبها .**

٢٦-٥

```
PROGRAM  Fibonacci (OUTPUT) ;
VAR i : INTEGER ;
T : ARRAY [1..100] OF REAL ;
S : REAL ;
BEGIN
T[1] := 0.0 ; T[2] := 1.0 ;
S := 1.0 ;
FOR i := 3 TO 100 DO
BEGIN
T[i] := T[i-1] + T[i-2] ;
S := S + T [i]
END ;
WRITELN ('S = ' , S)
END.
```

٢٧-٥

```
m := n DIV 2 ;
FOR i := 1 TO m DO
BEGIN
Temp := A[i] ;
A[i] := A[n + 1 - i] ;
A [n + 1 - i] :=  Temp
END ;
FOR i := 1 TO n DO
WRITELN (A[i]) ;
```

٢٨-٥

```
PROGRAM HadeethType (INPUT , OUTPUT) ;
VAR
i , ID , M , small : INTEGER ;
```

```
N : ARRAY [1..6] OF INTEGER  ;
                                    BEGIN
                                READLN (M , ID) ;
                               FOR i := 1 TO M DO
                             READLN (N[i]) ;
                                    small := N [1] ;
                               FOR i := 2 TO M DO
                        IF N [i] < small THEN
                           small := N [i] ;
                                IF small >= 10 THEN
                     WRITELN (′Mutawatir′)
                        ELSE IF small >= 3 THEN
                  WRITELN (′Ahad, Mashhoor′)
                        ELSE IF small = 2 THEN
                   WTIYRLN (′Ahad , A′ ′zeez′)
                                                ELSE
                   WRITELN (′Ahad , Ghareeb′)
                                                    END.
```

٢٩–٥

```
Program Compare ;
Var
      A, B : array [1..10] of integer ;
      sum, i,n : integer ;
Begin readln (n) ;
      For  i:= 1 to n do
            readln (A[i]) ;
      For i := 1 to n do
            readln (B[i]) ;
      sum := 0 ;
      For i := 1 to n do
            if  A[i] > 2 then
            sum := sum + B [i] ;
      writeln (′sum = ′, sum)
End.
```

٣٠–٥

```
Program MaxScores ;
Var   i, MaxA, MaxB : integer ;
      SA, SB : array [1..20] of integer ;
```

```
Begin
        For i := 1 To 20 Do
                readln (SA [i]) ;
        For i := 1 To 20 Do
                readln (SB [i]) ;
        MaxA := SA [1] ; MaxB := SB [1] ;
        For i := 2 To 20 Do begin
                if MaxA < SA [i] then
                        MaxA := SA [i] ;
                if  MaxB < SB [i]then
                        MaxB := SB[i] end ;
                IF MaxA > MaxB then
                        Writeln (′Class A has the higher Max′)
                else if MaxB > MaxA then
                        Writeln (′Class B has the higher Max′)
                else
                        Writeln (′Both classes have the same Max′)
End.
```

٣١–٥

```
Program Averages ;
Var
        i : integer ;
        SA, SB : array [1..20] of real ;
        AvgA, AvgB : real ;
Begin
        AvgA := 0.0 ; Avg B := 0.0 ;
        For i := 1 To 20 do
                readln (SA [i]) ;
        For i := 1 To 20 Do
                readln (SB [i]) ;
        For i := 1 To 20 Do begin
                AvgA := AvgA + SA [i] ;
                AvgB := AvgB + SB [i] end ;
                AvgA := AvgA / 20 ;
                AvgB := AvgB / 20 ;
                IF AvgA > AvgB then
                        Writeln (′AvgA is higher′)
                else if AvgB > AvgA then
                        Writeln (′AvgB is higher′)
                else
```

```
        Writeln (′Both Averages are equal′)
End.
```

٣٢–٥

```
Program Students (input, output) ;
Var
    id, mark    :    array [1 .. 100]    of integer ;
    dept        :    array [1 .. 100]    of char ;
    high, sum   :    integer ;
    n, i, ncomputer, high_index : integer ;
begin
    readln (n) ; {assume    1 <= n <= 100}
    {loop to input}
    for    i := 1 to n do
    readln (id [i], mark [i], dept [i]) ;
    {loop for best in Math and sum all in computer}
    {we may use one or two loops}
    high := 0    ;    high_index := 0 ;
    ncomputer := 0   ;  sum := 0 ;
    for i := 1  to  n  do
        begin
            if dept [i] = ′C′ then
                begin
                    sum := sum + mark [i] ;
                    ncomputer = ncomputer + 1
                end ;  {if dept ...}
            if (dept [i] = ′M′) and (mark [i] > high) then
                begin
                    high := mark [i] ;
                    high_index := i
                end {if dept ...}
        end ; { for  i := ....}
    {output}
    if ncomputer > 0 then
       Writeln (′average of computer students = ′,sum / ncomputer)
    else
       Writeln (′Number of computer students = 0′) ;
    if high_index > 0 then
        begin
            Writeln (′id and mark of best math. student′) ;
            Writeln (′id = ′, id [high_index], ′mark = ′,
```

```
                                    mark [high_index])
            end
        else
            Writeln (′sorry, there are no Math. students′)
end. {Students}
```

٣٣–٥

```
Program Students  (input, output) ;
Var
        ID : array [ 1.. 50] of integer ;
        WAV : array [ 1.. 50] of real ;
        w1, w2, w3,  s1, s2, s3  :  real ;
        n, i : integer  ; Big, Av  :  real ;
Begin
        Writeln (′Enter number of students & weights′) ;
        readln (n, w1, w2, w3 ) ;
        for  i := 1  To  n  do
            begin
                Writeln (′Enter 3 scores of a student and his ID′) ;
                readln  (s1, s2, s3, ID [i]) ;
                WAV [i] := w1 * s1 + w2 * s2 + w3 * s3
            end ;
        Big := WAV [1] ;  Av := 0.0 ;
        For  i :=  1  To  n  do
            begin
                Av := Av + WAV [i],
                if WAV [i] > Big then Big := WAV [i]
            end ;
        Writeln (′Highest average = ′, Big) ;
        Writeln (′overall averge = ′, Av / n)
End.
```

٣٤–٥

```
Program Families  (input, output) ;
Var
        n, i, m, no : integer ;
        m, ID : array [1..25] of integer ;
        P, income : array [1 .. 25] of real ;
            AvIncome , sum , percent : real ;
Begin
        Writeln (′Enter number of families′) ;
        readln (n) ; sum := 0 ;
```

```
        For i := 1  To  n  do
                begin
                        Writeln (′Enter id, income, m of a family′) ;
                        readln (ID [i], income [i], m [i]) ;
                        sum := sum + income [i]
                end ;
        AvIncome := sum / n ;
        For  i := 1  To  n  do
                if income [i] > AvIncome then
                        Writeln (ID [i], income [i]) ;
        no := 0 ;
        For i := 1  To  n  do
                begin
                        P[i] := 6500.00 + 750.00 * (m[i] - 2) ;
                        if income [i] < P[i] then
                                no := no + 1
                end ;
        percent := no / n * 100.00 ;
        writeln (′percentage of poor families =′, percent)
End.
```

۳۵–۵

```
program Tests ;
{
This program reads data for a true-false test and computes the score
                                        for each student, and his or her grade.
INPUT FORMAT: The first line contains the answer key, a string of
T's or F's. Then follows number of students, and then one line for
each student, containing the student id, one space, and then test
                                        answers (a string of T's or F's).
OUTPUT: A table of student ids, scores and grades.
}
const
        MaxSize  =  20 ;        {maximum number of students}
        NumQuest  =  10 ;       {number of test questions}
type
        IndexType  =  1..MaxSize ;
        CountRange  =  0..MaxSize ;
        ArrayType  =  array [IndexType] of Integer ;
var
        ID : ArrayType ;                {input  -  student ID numbers}
        Score: ArrayType ;              {output - test scores}
```

```
        N : CountRange ;                {number of scores processed}
procedure ReadTestData
        (var ID      {output}    : ArrayType ;
        var Score    {output}    : ArrayType ;
        var N        {output}    : CountRange) ;
type
        QuestRange  =  1..NumQuest ;
        QuestList    =  String[NumQuest] ;
var
        Junk : Char;        {to skip space between id and answers}
        Answers,            {Student answers to questions}
        Key : QuestList ; {holds answer key}
        I : QuestRange ;  {loop control}
        Count : CountRange ;  {loop control}
begin {ReadTestData}
        {Get answer key}
        ReadLn (Key) ;
        {Get number of students}
        Readln (N) ;
        {Get each test data}
        for count := 1  to  N  do
        begin
                {Get next ID, skip space and read answers}
                ReadLn (ID [Count], Junk, Answers) ;
                {compute score}
                Score [Count] := 0 ;
                for I :=  1  to NumQuest do
                        if Answers [I] = Key [I] then
                                Score [Count] := Score [Count] + 1
        end ; {while}
end ; {ReadTestData}
procedure PrintTable  (ID {input} : ArrayType ;
                Score{input}      :      ArrayType ;
                N      {input}    :      IndexType) ;
var
        I  :  IndexType ;           {loop control and subscript}
        Best  :  Integer ;          {highest score}
function MaxArray (X : ArrayType ;
                                Count  :  Integer)  :  Integer ;
{
Find the largest array member.
```

```
var
        I  :  Integer ;                {loop control}
        Biggest  :  Integer ;  {biggest found so far }
begin {MaxArray}
        Biggest := X[1] ;
        for I := 2 to Count do
                if Biggest < X[I] then
                        Biggest := X[I] ;
        MaxArray := Biggest
end ; {MaxArray}
begin {PrintTable}
        Best := MaxArray (Score, N) ;
        WriteLn (′D′ : 4, ′Score′ :12, ′Grade′ : 8) ;
        for I := 1 to N do
                begin
                        Write (ID[I] : 4, Score [I] : 10, ′    ′) ;
                        if Score [I] >= Best - 1 then
                                WriteLn (′A′)
                        else if Score [I] >= Best -3 then
                                WriteLn (′C′)
                        else
                                WriteLn (′F′)
                end {for}
end ; {PrintTable}
begin {Tests}
        ReadTestData (ID, Score, N) ;
        if N = 0 then
                WriteLn (′No tests to process. ′)
        else
                PrintTable (ID, Score, N)
end. {Tests}
```

٣٦–٥

```
program ThirtypDigitNums ;
{
Simulate a computer that can only read, write and add single digits.
                Program can read, add and display 30 digit numbers.
}
const
        Max = 30 ;                {maximum number of digits}
type
```

```
        IndexType = 1..Max ;
        NumArray = Array [IndexType] of Integer ;
var
        Num1,                   {input      -       30 digit integer}
        Num2,                   {input      -       30 digit integer}
        Num3 : NumArray ;       {output     -       Num1 + Num2}
        Overflow : Boolean ;    {output     -       overflow flag}
procedure ReadLn30Digits (var Num : NumArray) ;
{Read 30 digit integer into array.}
var
        I : Integer ;           {interm - loop counter}
begin {Get30Digits}
        for I := Max downto 1 do
                Read (Num [I]) ;
        ReadLn ;
end ; {ReadLn30Digits}
procedure Add30Digits (var Num1, Num2, {input}
                                Num3  {output}  :  NumArray ;
                        var Overflow:  Boolean) ;
{
Add two NumArray arrays together. If their sum is greater than 30
digits then set Overflow to True.
}
var
        Sum,                    {interm     -       Sum of two digits}
        Carry,                  {interm     -       if sum  >  9}
        I  :  Integer ;         {interm     -       loop counter}

begin {Add30Digits}
        I := 1 ;
        Carry := 0 ;
        while I <= Max do
                {
                Carry is set to 1 if Num1 [I] + Num2 [I] + Carry > 9.
                }
                begin
                        Sum := Num1 [I]  +  Num2 [I] + Carry ;
                        Num3 [I] := Sum mod 10 ;
                        Carry := Sum div 10 ;
                        I := I + 1
                end ; {while}
```

```
        Overflow := Carry <> 0                {return overflow flag}
end ; {Add30Digits}
procedure Display 30Digits {Num : NumArray) ;
{Display a thirty digit integer.}
var
        I : Integer ;                {interm - loop counter}
begin {Display30Digits}
        for I := Max downto 1 do
                Write (Num[I]) ;
        WriteLn
end ; {Display30Digits}
begin {ThirtyDigitNums}
        WriteLn     (′Enter two 30 digit integers. Digits must ′) ;
        WriteLn     (′be separated from one another by a single′) ;
        WriteLn     (′blank. Use leading zeros, if number has ′) ;
        WriteLn     (′fewer than 30 digits. ′) ;
        WriteLn ;
        WriteLn     (′First addend′) ;
        ReadLn30Digits (Num1) ;
        WriteLn ;
        WriteLn     (′Second addend′) ;
        ReadLn30Digits (Num2) ;
        Add30Digits (Num1, Num2, Num3, Overflow) ;
        WriteLn ;
        WriteLn     (′Sum is′) ;
        WriteLn ;
        { Display carry digit if there is one }
        if Overflow then
                Write (1 : 1)
        {display rest of sum}
        Display30Digits (Num3) ;
end. {ThirtyDigitNums}
```

۳۷–۵

```
PROGRAM SortingNames  (INPUT, OUTPUT) ;
CONST
        max = 2000 ;  {max  no. of names}
        length = 20 ; {max length of a mame}
TYPE
        Name = {PACKED ARRAY [1 .. length] OF CHAR ;
VAR
```

```
    n : 1..max {actual no. of names} ;
    LIST : ARRAY [1..max] OF Name {list of names} ;
    Temp : Name ; {temporary or auxiliary variable for exchanging
                   names}
    i,j,k:1..max {counters} ;
BEGIN
    {Read the names until * appears.
    Store the names in an array of strings}
    i := 1 ;
    WRITE (′enter a′, length:3, ′character name′) ;
    READ (List [i,1]) ; {read the first character of the name}
    WHILE List [i, 1] <> ′*′ DO
        BEGIN
            FOR  j :=2 TO length DO
                READ (List [i,j]) ;
            READLN ;
            i := i + 1 ;
            WRITE (′Enter a name or * to end  : .. ′) ;
            READ  (List [i,1])
        END ;
    n := i - 1 ;
    {Sort the names alphabetically}
    FOR  i := 1  TO  n-1  DO
        {first find the order j of the smallest element, then
            interchange with the first one}
        BEGIN
            j := i ;
            FOR  k := i+1  TO  n  DO
                If  List  [k] < List [j]  then j := k ;
            Temp := List [i] ;
            List [i] := List[j] ;
            List [j] := Temp
        END ;
    {print the sorted names}
    WRITELN (′The sorted names are : ′) ; WRITELN ;
    FOR  i := 1  TO  n  DO
        WRITELN (LIST [i])
END. {SortingNames}
```

ملاحظة : في هـذا الحل اتبعنا الخوارزمية التالية لترتيب الأسماء .. List [1]

أبجديا : list [n]

ضع أصغر اسم في الموضع List [i] [حيث i = 1 , 2 , 3 , ... ,n-1] ،
وذلك بأن تبحث أولا عن أصغر عنصر في المجموعة الجزئية
List [i] , List [i+1] , .. , List [n-1] , List [n]
ثم تقوم بتبديله مع العنصر الأول List [i] في هذه المجموعة الجزئية.

حل آخر : أما في الحل التالي فإننا سنقوم باتباع الخوارزمية التالية :

ضع أصغر عنصر في الموضع List [i] [حيث i = 1 , 2 , 3 , ... , n-1] ،
وذلك بأن تقارن كل عنصر من عناصر المجموعة الجزئية
List [i+1] , ... , List [n-1] , List [n]
مع العنصر List [i] ، وكلما وجدنا عنصرا (اسما) List [j] أصغر من List [i]
بدلناهما معا ، أي أن الحل السابق يبحث أولا عن أصغر عنصر في المجموعة الجزئية ثم يبدله مع العنصر الأول فيها List [i] ، أما الحل التالي فإنه كلما وجد عنصرا أصغر من العنصر الأول List [i] بدلهما معا. (انظر حل مثال ٥-٨).

M
```
{Sort the names alphbetically}
        FOR i := 1 TO n-1 DO
      FOR j := i + 1 TO n DO
  If List [j] < List [i] THEN
          BEGIN
Temp := List [i]  ;
List [i] := List [j] ;
  List [j] := Temp
          END ;
          {print the sorted names}
```
M

٥-٣٨

```
{Calculate Students ' Averages}
 FOR  i  := 1  TO  50  DO
     BEGIN
        sum := 0.0  ;
  FOR j :=  1  TO  4  DO
sum := sum + S [i,j] ;
```

```
AvSt [i]  := sum / 4.0
                    END ;
          {Calculate averages of Subjects}
               FOR j :=  1  TO  4  DO
                         BEGIN
                    sum := 0.0  ;
               For i := 1 to 50 do
          sum := sum + S[i,j] ;
            AvSub [j] :=  sum /50.0
                              END ;
            {Calculate Class Average }
                         sum :=  0.0
               FOR j :=  1  TO  4  DO
          sum := sum + AvSub [j] ;
                CLAV :  = sum / 4.0 ;
                    {Print the results}
     WRITELN ( ` Averages of students `)
                    FOR i  :=  1  TO  50  DO
       WRITELN( ID [i] ,  AvSt[i])  ;
     WRITELN ( ` Averages of Subjects ` )
               FOR j  :=  1  TO  4  DO
     WRITELN ( j , AvSub [j] )  ;
WRITELN ( ` Class Average  =  ` , CLAv)
```

٣٩–٥

```
PROGRAM Ahadeeth ( INPUT , OUTPUT)  ;
   {Program of Compilation of Ahadeeth
     (Traditions of the Prophet p.b.u.h.)
        in the book :  ` Riyad - Us- Saliheen`}
                              VAR
ARRAY [ 1 .. 1893 , 1 .. 4] OF INTEGER ;  M   :
   ARRAY [ 1 .. 1893 ] OF INTEGER ;  ID  :
    ARRAY [ 1..4] OF INTEGER ;  N   :
          i , j , NBM :  INTEGER ;
                              BEGIN
          FOR i  := 1  TO  1893  DO
                         BEGIN
     READ ( ID[ i ] ;
  for j :=  1  TO  4 DO
READ ( M[i,j]) ;
          READLN
                    END ;
```

```
{Compute number of Ahadeeth narrated by each compiler }
                                    FOR j := 1  TO  4  DO
                                                BEGIN
                                N[j] := 0 ;
                        FOR i :=  1  TO 1893 DO
                            N[j] := N[j] + M [i,j]
                                                END ;
        {Compute number of Ahadeeth narrated by both Bukhari and
                                                Muslim }
                                                NBM :=  0 ;
                                    FOR i := 1 TO 1893 DO
                    IF (M[i,1] = 1) AND (M[i,2] = 1) THEN
                                NBM := NBM + 1 ;
                                                {Print the results}
                                    FOR j :=  1  TO  4  DO
                        WRITELN (′N [′, j:2 , ′] = ′, N [j]) ;
                WRITELN (′ Number of Ahadeeth compiled by both
                        Bukhari and Muslim = ′ , NBM )
                                                END .
```

٤٠–٥

```
NP := 0 ;    NN := 0 ;    N0 := 0 ;
SP := 0.0 ;  SN := 0.0 :
FOR i := 1 TO 7 DO
        FOR j := 1 TO 9 DO
                IF A[i,j] > 0.0 THEN
                        BEGIN
                                NP := NP + 1 ;
                                SP := SP + A[i,j] ;
                        END
                ELSE IF A[i,j] < 0.0 THEN
                        BEGIN
                                NN := NN + 1 ;
                                SN := SN + A[i,j]
                        END
                ELSE
                        N0 := N0 + 1
```

٤١–٥

```
FOR i := 1 TO 9 DO
        FOR j := 1 TO 9 DO
                A[i,j] := 1.0 / (i + j - 1) ;
```

```
FOR i := 1 TO 9 DO
        BEGIN
                FOR J := 1 TO 9 DO
                        WRITE (A[i,j]) ;
                WRITELN
        END ;
SD := 0.0 ;
FOR i := 1 TO 9 DO
        SD := SD + A[i,i] ;
WRITELN ('SD = ', SD) ;
FOR i := 1 TO 8 DO
BEGIN
        FOR j := i + 1 TO 9 DO
                WRITE (A[i,j]) ;
        WRITELN
END.
```

٥-٤٢ انظر حل السؤال رقم ٥-٢٠.

أجوبة تمرينات الفصل السادس

١-٦

a. MESSAGE (X , Y , Z)
b. MESSAGE (M , Y , N)
e. MESSAGE (Y+Z , Y-Z , M)

٢-٦

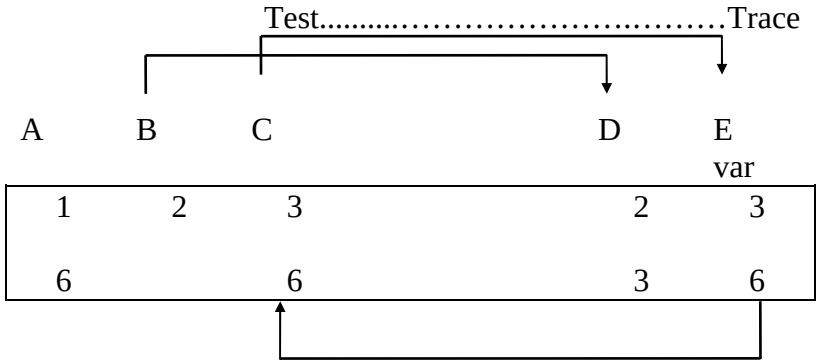
On P(A , B/2 , 'C') ;
Tw P(A+1 , B , H) ;

a. P(A , A*B , CH) ... **لاحظ أن المتغير** CH **غير معرف في البرنامج الرئيسي**

b. **الاستدعاء خاطئ في** d **لأن عدد الوسطاء الفعليين (٤) لا يتفق مع عدد الوسطاء الشكليين (٣).**

٣-٦ x = 15.0

٤-٦ أ)



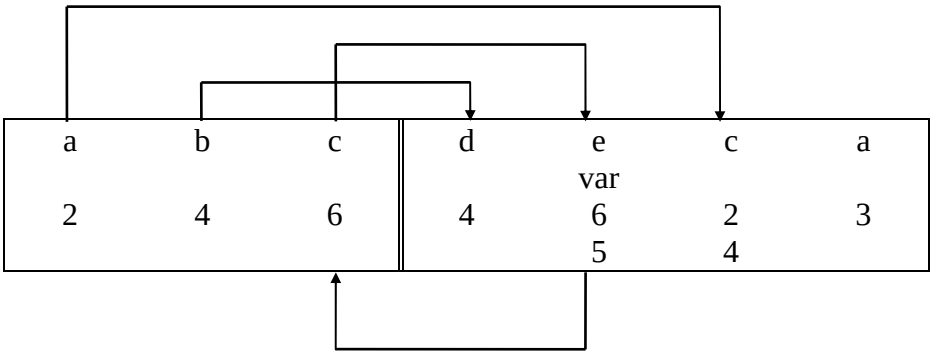
D = 2 + 1 = 3
E = 3 + 3 = 6
A = 2 * 3 = 6

المخرجات

A = 1	D = 2	E = 3
A = 6	D = 3	E = 6
A = 6	B = 2	C = 6

ب)

exam Subprogram



a = 2 + 1 = 3
e = 4 + 2 - 1 = 5
c = 2 * 2 = 4

المخرجات

5 4 2

ج) **المخرجات :**

6 6
12 24
12 6

٦-٥

program a - a , b , c , d , e , f , g
procedure b - b , c , d
procedure c - c , d
procedure d - d
procedure e - e , f , g
procedure f - f
procedure g - f , g

٦-٦

program a - a , b , c , d , e , f
procedure b - b , c , d , e
procedure c - c
procedure d - c , d , e
procedure e - e
procedure f - b , f

٦–٧ تصحيح العبارات الخاطئة :

```
PROCEDURE SWAP (VAR  X, Y : INTEGER)  ;
      VAR
            TEMP : INTEGER  ;
      BEGIN
            TEMP := X  ;
            X := Y  ;
            Y := TEMP
      END  ;
BEGIN
      IF X > Y   THEN   SWAP (X,Y)  ;
      IF X > Z   THEN   SWAP (X, Z)  ;
      IF Y > Z   THEN   SWAP (Y, Z)
END  ;
```

٦–٨

```
Where do you live, Where do you live
In a tent or in a town
It doesn't matter where you live
⋮
So, turn, turn to Allah
Where do you come from
⋮
Islam is the Best
```

سطر فارغ

```
What are you saying, What are you saying
I'm praying to Allah
⋮
So, turn, turn to Allah
Where do you come from
⋮
Islam is the Best
```

٦–٩

```
PROGRAM Main (INPUT, OUTPUT)  ;
TYPE
      Vector = ARRAY [ 1..50] OF REAL   ;
VAR A : Vector  ;
      Whole, fraction  :  REAL  ;
```

```
        i       :       INTEGER,
PROCEDURE  Break  (X : REAL   ;
                    VAR  whole : INTEGER ; VAR fraction : REAL)  ;
        BEGIN
                Whole       :=      TRUNC (X)  ;
                fraction    :=      X - Whole
        END  ;
BEGIN
        FOR  i := 1  To  50  Do
                BEGIN READLN A[i]  ;
                        Break (A[i], whole, fraction)  ;
                        WRITELN (′A(′, i : 2, ′) =′, A[i],
                                ′whole =′, whole, ′ fraction = ′, fraction )
                END
END.
```

٦–١٠

```
Program Main  ;
Type vector = array [1..8] of real  ;
Var
        A, B, C  :  vector  ;
        i : integer  ;
Procedure Multiply (X, Y : vector  ; n : integer  ; var Z : vector)  ;
Var   i  :  integer  ;
begin
        for  i :=  1  to  n  do
                Z[i] :=  X[i] * Y[i]
end  ;
Begin
        for  i :=  1  to  8  do
                readln (A[i])  ;
        for i := 1  to  8  do
                readln (B[i])  ;
        Multiply (A, B, 8, C)  ;
        for i := 1  to  8  do
                writeln (A[i], B[i], C[i])
End.
```

٦–١١ (أ)

```
procedure adjust (var a: list  ; n: integer)  ;
        {This procedure adjusts values in an array}
var
```

```
        least : real  ;
        i : integer  ;
begin
        least := a[1]  ;
        for i := 2 to n do
                if a[i] < least then
                        least := a[i]  ;
        for i := 1 to n do
                a[i] := a[i] - least
end  ;
```

(ب)

```
procedure shift (var a: list  ; n: integer)  ;
        {This procedure adjusts values in an array.}
var
        most: real  ;
        i : integer  ;
begin
        most := a[1]  ;
        for i := 2 to n do
                if a[i] > most then
                        most := a[i]  ;
        for i := 1 to n do
                a[i] := a[i] - most
end  ;
```

(ج)

```
procedure zeroav (var a: list)  ;
        {This procedure adjusts the values in an array}
var
        sum, ave  ; real  ;
        i : integer  ;
begin
        sum := 0 ;
        for i := 1  to  50  do
                sum := sum + a[i]  ;
        ave := sum / 50.0  ;
        for i := 1 to  50  do
                a[i] := a[i] - ave
end  ;
```

١٢–٦

```
procedure prtval (k : list  ; n: integer)
```

```
        {This procedure prints an array k with n elements.}
var
        i : integer  ;
begin
        writeln (′VALUES IN ARRAY′)  ;
        for i := 1  to  n  do
                writeln (i : 2, ′      ′, a[i] : 3)
end  ;
```

١٣–٦

```
procedure bigarr (a, b: list  ; var c: list)  ;
        {This procedure determines the larger array to put into array c.}
var
        asum, bsum: real  ;
        i : integer  ;
begin
        asum := 0  ;
        bsum := 0  ;
        for  i := 1  to  50  do
                begin
                        asum := asum + a[i]  ;
                        bsum := bsum + b[i]
                end  ;
        if asum > = bsum then
                for i := 1  to  50  do
                        c[i] := a[i]
        else
                for i := 1  to  50  do
                        c[i] := b[i]
end  ;
```

١٤–٦

```
procedure bottom (var b : list  ; limit: real)  ;
        {This procedure adjusts values in an array.}
var
        i: integer  ;
        begin
                for  i :=  1  to  50  do
                        if      b[i]  < limit then
                                b[i] := limit
        end  ;
```

١٥–٦

```
procedure chng (var b : list)  ;
        {This procedure converts values to zeros and ones}
var
        i : integer  ;
begin
        for   i := 1  to  50  do
                if odd (b[i]) then
                        b[i] := 1
                else
                        b[i] := 0
end  ;
```

١٦–٦

```
procedure modify (var x: list)  ;
        {This procedure reverses the order of values in an array.}
var
        temp : real  ;
        i : integer  ;
begin
        for i := 1  to  25  do
                begin
                        temp := x[i]  ;
                        x[i] := x[51 - i]  ;
                        x[51 - i] := temp
                end
end  ;
```

١٧–٦

```
procedure time (totmin: integer  ; var days, hours, min: integer )  ;
        {This procedure converts minutes to days, hours, and minutes.}
begin
        days := totmin div 1440  ;
        hours := (totmin - days*1440) div 60  ;
        min := totmin - days*1440 - hours*60
end  ;
```

١٨–٦

```
procedure length (totinc: integer  ; var yards, feet, inches: integer)  ;
        {This procedure converts inches to yards, feet, and inches}
```

```
begin
        yards := totinc div 36  ;
        feet := (totinc - yards*36) div 12  ;
        inches := totinc - yards*36 - feet*12
end  ;
```

٦–١٩

```
procedure time (totday: integer  ; var weeks, days: integer )  ;
        {This procedure converts days to weeks and days.}
begin
        weeks := totday div 7  ;
        days := totday - weeks*7
end  ;
```

٦–٢٠

```
Type    list  =  array [1..100] of real  ;
procedure smooth (var y : list )  ;
        {This procedure smoothes noise in data}
var
        max: real  ;
        i : integer  ;
begin
        max := y[1]  ;
        for i := 2  to  100  do
                if y[i] > max then
                max := y[i]  ;
        for i := 1  to  100  do
                if y[i] < (0.03 * max ) then
                y[i]: = 0.0
end  ;
```

٦–٢١

```
procedure trig (angle: real)  ;
        {This procedure finds the sine and cosine of angle.}
begin
        writeln (′ SINE    (′, angle : 6: 2, ′   DEGREES)  =  ′,
                                        sin(angle * pi/180.0) : 6: 4)  ;
        writeln (′COSINE (′, angle : 6: 2, ′DEGREES) = ′,
                                        cos(angle * pi / 180.0) :6 : 4)  ;
end  ;
```

٦–٢٢

```
program ScalingArrays  ;
```

```
type index  =  1..50 ;
        list  =  array [index ] of real ;
var
        a : list ;
        i, n : integer ;
procedure scale (var a: list ; n: integer) ;
        {This procedure adjusts values in an array.}
var
        maxim: real ;
        i : integer ;
begin
        maxim := a[1] ;
        for  i := 2  to  n  do
                if a[i] > maxim then
                        maxim := a[i] ;
        for i := 1  to  n  do
                a[i] := a[i] / maxim
end ;

Begin
        for i := 1  to  50  do
                readln (a[i])
        n := 50 ;
        scale (a, n) ;
        for i := 1  to  50  do
                writeln (a[i])
end.
```

٦–٢٣

```
Type
        vector  =  array [1..max] of real ;
procedure Reverse (X : vector  ; n : integer ;
                                var Y : vector) ;
var
        i : integer ;
begin
        for i := 1  to  n  do
                Y[i] := X[n-i+1]
end ;
```

٦–٢٤ أ)

```
procedure MAX (x, y, z  :  real ; var big : real) ;
```

```
begin
        if x > y then big := x
        else big := y ;
        if z > big then big := z
end ;
```

ب)

```
program Main (input, output) ;
VAR
        a, b, c, large  :  real ;
procedure MAX .....
......
......
......
BEGIN
        readln (a, b, c) ;
        REPEAT
                MAX (a, b, c, large) ;
                Writeln (a, b, c, max = ′, large) ;
                readln (a, b, c)
        UNTIL
                (a = b) AND  (a = c) ;
        writeln (a, b, c, ′stop′)
END.
```

٦–٢٥

```
Program Main (output) ;
Type
        Vector  =  array [1..10] of integer ;
Var
        A, B : Vector ;
        i : integer ;
Procedure Subsequences (X : vector ; n : integer ; var Y : vector) ;
Var
        i, j : integer ;
begin
        For  i := 1  to  n  do
        begin
                Y [i] := 0 ;
                for j := 1  to  i  do
                        Y[i] := Y[i] + X[j] ;
        end
```

```
end ;
Begin
        For i := 1  to  10  do
                A [i] := i ;
        Subsequences (A, 10, B) ;
        For i := 1  to  10  do  writeln (B[i])
End.
```

ملاحظة : حل آخر مختصر للإجراء

```
begin
        Y[1] := X [1] ;
        for i := 2  to  n  do
                Y [i] := Y[i-1] + X[i]
end ;
```

٢٦-٦

```
program StackHouses ;
        {Program draws two houses stacked on top of one another.}
        procedure DrawTriangle ;
        {procedure draws a triangle.}
        begin {DrawTriangle}
                WriteLn     (′   *′) ;
                WriteLn     (′ ***′)
        end ;  {DrawTriangle}

procedure DrawRectangle ;
{Procedure draws a rectangle.}
        begin {DrawRectangle}
                WriteLn     (′*****′) ;
                WriteLn     (′*****′) ;
                WriteLn     (′*****′)
        end    {DrawRectangle}
begin {StackHouses}
        {Draw first house.}
        DrawTriangle ;
        DrawRectangle ;
        {Skip two lines.}
        WriteLn ;
        WriteLn ;
        {Draw Second house.}
        DrawTriangle ;
        DrawRectangle
```

```
end.  {StackHouses}
```

٦-٢٧ نفرض أن اسم الشخص هو :

ادريس بلال خالد ، وأن الحروف الأولى من اسمه هي : E. B. K.

```
program Initials ;
{This program displays my initials vertically.}
procedure DrawE ;
{Draw an 'E' in block letter form.}
begin {DrawE}
      WriteLn    ('*******  ') ;
      WriteLn    ('*        ') ;
      WriteLn    ('*        ') ;
      WriteLn    ('***      ') ;
      WriteLn    ('*        ') ;
      WriteLn    ('*        ') ;
      WriteLn    ('*******  ')
end ; {DrawE}
procedure DrawB ;
{Draw a 'B' in block letter form.}
begin {DrawB}
      WriteLn    ('******    ') ;
      WriteLn    ('*       *') ;
      WriteLn    ('*       *') ;
      WriteLn    ('******    ') ;
      WriteLn    ('*       *') ;
      WriteLn    ('*       *') ;
      WriteLn    ('******   ')
end ; {DrawB}

procedure DrawK ;
{Draw a 'K' in block letter form.}



begin {DrawK}
      WriteLn    ('*         *') ;
      WriteLn    ('*       *' ) ;
      WriteLn    ('*    *      ') ;
      WriteLn    ('****        ') ;
      WriteLn    ('*    *      ') ;
```

```
        WriteLn    ('*        *    ') ;
        WriteLn    ('*            *')
end ; {DrawK}

begin {Initials}
        DrawE ;
        WriteLn ;
        DrawB ;
        WriteLn ;
        DrawK
end.  {Initials}
```

٢٨–٦

```
program DrawFig ;
var
        FigType : Char ;  {input - figure code S or T}
procedure DrawTriangle ;
{Procedure draws a triangle.}
begin {DrawTriangle}
        WriteLn    ('   *    ') ;
        WriteLn    ('  ***   ') ;
        WriteLn    ('***** ') ;
end ; {DrawTriangle}

procedure DrawSquare ;
{Procedure draws a square.}
begin {DrawSquare}
        WriteLn   ('*****') ;
        WriteLn   ('*****') ;
        WriteLn   ('*****') ;
end ; {DrawSquare}

begin {DrawFig}
        {Read figure code.}
WriteLn ('Type figure code, S for square or T for triangle > ') ;
ReadLn (FigType) ;
{Display figure.}
if FigType = 'S' then
        DrawSquare
else if FigType = 'T' then
        DrawTriangle
```

```
else
        WriteLn (figType, ′ is a bad figure code. ′)
end.  {DrawFig}
```

٦-٢٩

```
program Draw3Fig ;
{Program allows user to display 3 figures.}
var
        Fig1, Fig2, Fig3 : Char ; {input - figure code}
procedure DrawTriangle ;
.....
```

كما حل السؤال السابق

```
.....

procedure DrawSquare ;
.....
```

كما حل السؤال السابق

```
.....

procedure DrawCircle ;
        {Procedure draws a circle.}
begin {DrawCircle}
        WriteLn     (′      *    ′) ;
        WriteLn     (′*         *′) ;
        WriteLn     (′   *    * ′) ;
end ;  {DrawCircle}

 procedure DrawLine ;
        {Procedure draws a line.}
begin {DrawLine}
        WriteLn (′*****′)
end ;  {DrawLine}

begin {Draw3Fig}
        {Read figure code.}
        WriteLn (′Type 3 figure codes′) ;
        WriteLn (′S for square or T for Triangle′) ;
        WriteLn (′or L for Line or C for Circle > ′) ;
        ReadLn (Fig1, Fig2, Fig3) ;
```

```
        {Display figures.}
        Case Fig1 of
                'C' : DrawCircle ;
                'S' : DrawSquare ;
                'T' : DrawTriangle ;
                'L' : DrawLine
        end ; {case}

        case  Fig2 of
                'C' : DrawCircle ;
                'S' : DrawSquare ;
                'T' : DrawTriangle ;
                'L' : DrawLine
        end ; {case}

        case Fig3 of
                'C' : DrawCircle ;
                'S' : DrawSquare ;
                'T' : DrawTriangle ;
                'L' : DrawLine
        end ; {case}
end.  {Draw3Fig}
```

٣٠–٦

```
Program Classroom ;
{
This program prints a report of enrollments for a classroom.
Given a room's number, capacity, and number of students enrolled, it
prints this information, the number of seats still available, and
                        indication of whether or not the room is filled.
}
var
        RoomNumber,             {input - Classroom number}
        Capacity,               {input - Room capacity}
        Enrollment,                     {input - Number enrolled}
        Available : Integer ;   {output - Seats left}
procedure PrintHeader ;
{Print data header line}
begin {PrintHeader}
        WriteLn ;
        Write ('Room  Capacity  Enrollment  Empty  Seats  ') ;
```

```
        WriteLn (′Filled / Not Filled′) ;
        WriteLn
end ; {PrintHeader}

begin {Classroom}
        {Display instructions and read room data}
        Write (′Enter room number: ′) ;
        ReadLn (RoomNumber) ;
        Write (′Enter room capacity: ′) ;
        ReadLn (Capacity) ;
        Write (′Enter current enrollment: ′) ;
        ReadLn (Enrollment) ;

        {Compute the number of empty seats}
        Available := Capacity - Enrollment ;

        {Print header and all data}
        PrintHeader ;
        Write (RoomNumber: 4, Capacity : 10, enrollment : 12,
                                Available : 13) ;
        if Available <= 0 then
                WriteLn (′Filled′ : 13)
        else
                WriteLn (′Not Filled′ : 15)
end . {Classroom}
```

٣١–٦

```
Program LeastSquares (input, output) ;
Type
        Vector  =  array [1..20] of real ;
Var
        X, Y  :        Vector ;
        a, b   :       real ;
        i       :      integer ;
Procedure LSTSQ (X, Y  : Vector   ;  n  :  integer ;
                                var  a, b  :  real ) ;
Var
        sx, sy, sxy, sxx  :  real ;
        i  :  integer  ;
begin
        sx := 0.0  ;  sy := 0.0 ; sxy := 0.0 ; sxx := 0.0 ;
```

```
        For  i := 1  To  n  Do
                begin
                        sx := sx + X[i] ;
                        sy := sy + Y[i] ;
                        sxy := sxy + X[i] * Y[i] ;
                        sxx := sxx + X[i] * X[i]
                end ;
        b := (n * sxy - sx * sy) / (n * sxx - sx* sx) ;
        a := (sy - b * sx ) / n
end ;
Begin
        For  i := 1  To  20  Do
                begin writeln (′Enter the coordinates of a point′) ;
                        readln (X[i],  Y[i])  end ;
        LASTSQ (X, Y, 20, a, b )  ;
        writeln (′y = ′, a : 8 : 3, ′+′, b : 8 : 3, ′ x′)
End.
```

٣٢–٦

```
Program Main ;
Var
        side1, side2, side3, A : real ;
Function Area (a, b, c  :  real )  :  real ;
var
        s : real ;
begin
        s := (a + b + c) / 2.0  ;
        Area := SQRT (s * (s-a) * (s-b) * (s-c))
end ;
Begin
        readln (side1, side2, side3) ;
        A := Area (side1, side2, side3)
        writeln (′Area = ′, A)
End.
```

٣٣–٦

```
Program Marks ;
type vector  =  array  [1..50] of real ;
Var
        x : vector  ; i : integer  ;
function freq (a : vector   ;  n : real) : integer ;
var
```

```
        i : integer  ;  count : integer ;
begin
        count := 0 ;
        for  i := 1  to  50  do
                if  n  =  a[i]  then
                        count := count + 1 ;
        freq := count
end ;
Begin
        for  i := 1  to  50  do
                readln (X[i]) ;
        writeln (′no. of students having full mark is : ′, freq (x, 100))
End.
```

٣٤–٦

```
function totinc (yards, feet, inches: integer ): integer ;
        {This function converts yards, feet, and inches to inches.}
begin
        totinc := (yards*3 + feet)* 12 + inches
end ;
```

٣٥–٦

```
function totdys (weeks, days: integer): integer ;
        {This function converts weeks and days to days.}
begin
        totdys := weeks*7 + days
end ;
```

٣٦–٦

```
function bigeng (a, b, c: integer) : boolean ;
        {This function returns a boolean value based on a, b, and c.}
begin
        if (a > (b * c)) or (b > (a * c)) or (c > (a * b)) then
                bigeng := true
        else
                bigeng := false
end ;
```

٣٧–٦

```
function small (a, b, c: real ): boolean ;
        {This function returns a boolean value based on a, b, and c.}
begin
        if (a < (b/c)) or (b < (a/c)) or (c < (a/b)) then
```

```
            small := true
        else
            small := false
end ;
```

٦–٣٨

```
function differ (a, b, c: integer): boolean ;
        {This function returns a boolean value based on a, b, and c.}
begin
        if (a < (b - c)) or (b < (a - c)) or c < (a - b)) then
            differ := true
        else
            differ := false
end ;
```

٦–٣٩ أ)

```
function ascend (a: list ; n: integer) : boolean ;
        {This function returns a true value if a is in ascending order.}
var
        i : integer ;
begin
        ascend := true ;
        for i := 1  to  (n - 1) do
            if a[i] > a [i + 1] then
                ascend := false
end ;
```

ب)

```
function descnd (a : list  ; n: real): boolean:
        {This function returns a true value if a is in descending order.}
var
        i : integer ;
begin
        descnd := true ;
        for i := 1  to  (n - 1) do
            if a[i] < a[i + 1] then
                descnd := false
end ;
```

٦–٤٠

```
function boundd (a: list ; rmin, rmax: real ):  boolean ;
        {This function returns a true value if a lies between rmin and
          rmax.}
```

```
var
        i : integer ;
begin
        boundd := true ;
        for i := 1  to  50  do
                if (a[i] < rmin) or (a[i] > rmax ) then
                        boundd := false
end ;
```

٦–٤١

```
function maxs (a :  list ; n: integer): integer  ;
        {This function counts the occurrences of the maximum value.}
var
        i,  ms ;  integer ;
        maxim : real ;
begin
        maxim := a[1] ;
        ms := 1 ;
        for i := 2  to  n  do
                if a[i] > maxim then
                        begin
                                maxim := a[i] ;
                                ms: = 1
                        end
                else if a[i] = maxim then
                        ms := ms + 1 ;
        maxs := ms
end ;
```

٦–٤٢

```
function range (a: list ; n: integer): real ;
        {This function finds the range of the array.}
var
        maxim, minim: real ;
        i: integer ;
begin
        maxim := a[1] ;
        minim := a[1] ;
        for  i := 2  to  n  do
                if  a[i]  >  maxim then
                        maxim  :=  a[i]
```

```
            else if a[i] < minim then
                    minim  :=  a[i] ;
        range  :=  maxim  -  minim
end ;
```

٤٣–٦

```
function cubert (x: real) : real
        {This function returns the cube root of x.}
begin
        if abs(x)  =  x then
                cubert :=  exp  ((1.0 / 3.0) * ln(x))
        else
                cubert := -1.0*exp  ((1.0 / 3.0)*ln(abs(x)))
end ;
```

٤٤–٦

```
function xor  (a,  b:  boolean) :  boolean ;
        {This function performs an exclusive or on a and b.}
begin
        xor := a or b ;
        if  (a  and  b)  then
                xor := false
end ;
```

٤٥–٦

```
function close (a: list ; n: integer): real ;
        {This function finds a value closest to the average.}
var
        sum, avg, diff : real ;
        i : integer ;
begin
        sum := a[1] ;
        for i := 2  to  n  do
                sum := sum + a[i] ;
        avg := sum / n ;
        diff :=  abs (avg - a[1]) ; close := a[1] ;
        for i := 2  to  n  do
                if diff > (abs (avg - a[i])) then
                        begin
                                diff  := abs (avg - a[i]) ;
                                close := a[i] ;
                        end
end ;
```

٤٦–٦

```
function trap (b1, b2, h: real): real ;
        {This function calculates the area of a trapezoid.}
begin
        trap := ((b1 + b2 ) / 2.0)*h
end ;
```

٤٧–٦ أ)

```
function degr (rad: real): real ;
        {This function converts radians to degrees.}
var
        deg: real ;
begin
        deg := rad * 180.0 / 3.141593 ;
        while deg < 0.0 do        {Could also use mod function}
                deg := deg + 360.0 ;
        while deg > 360.0 do
                deg := deg - 360.0 ;
        degr := deg
end ;
```

ب)

```
function radn (deg: real) :  real ;
        {This function converts degrees to radians.}
var
        rad: real ;
begin
        rad := deg  *  3.141593 / 180.0 ;
        while rad < 0.0 do
                rad := rad + 2.0  *  3.141593 ;
        while rad > 2.0  *  3.141593 do
                rad := rad  -  2.0 * 3.141593 ;
        radn := rad
end ;
```

٤٨–٦

```
function count (b: list): integer ;
        {This function counts the number of A's in an array.}
var
        i :  integer ;
begin
        count := 0 ;
```

```
        for i :=  1  to  25  do
                if b[i]  =  'A' then
                        count := count + 1
end ;
```

٦–٤٩

```
function count (b: list ; n, m: integer): integer ;
        {This function counts consonants in a 2-dimensional array.}
var
        i , j :  intger ;
begin
        count := 0 ;
        for  i :=  1  to  n   do
                for  j :=  1  to  m  do
        if (b[i, j] <> 'A') and (b[i, j] <> 'E')  and
                (b[i, j] <> 'I') and (b[i, j] <> 'O') and
                b[i, j] <> 'U') then
                count := count + 1
end ;
```

٦–٥٠ أ)

```
Function Factorial (n: Integer ) : Real ;
        {The function computes n factorial for n > = 0}
var
        I : Integer ;
        Product : Real ;
Begin
        Product := 1.0 ;
For    I := 2  To  n  Do
        Product := Product * I ;
Factorial := Product
END ;
```

ب)

```
Program approximation (output) ;
        {program computes an approximation to the real number e}
Const
        Delta := 0.0001 ;
Var
        I : integer ; Sum, Term: Real ;
Function     Factorial (n : Integer) : Real ;
        ⋮
```

```
Begin        {main program}
      Sum := 0 ;  I := 0 ;  Term := 1.0 / Factorial (I) ;
Repeat
      Sum := Sum + Term ;
      I := I + 1 ;
      Term :=  1.0 / Factorial (I) ;
UNTIL      Term < Delta ;
Writeln ;
Writeln  ('The number e is', Sum)
END.
```

(أ) ٥١–٦

```
FUNCTION ADD (J , K : INTEGER) : REAL ;
VAR  i , sum : INTEGER ;
BEGIN
      IF J > K THEN
            ADD := 0.0
      ELSE
            BEGIN
                  sum := 1.0 / J ;
                  FOR i := 1 TO K-J DO
                        sum := sum + 1.0 / (J + i) ;
                  ADD := sum
            END
END ;
```

(ب)

X = 1.0833333 Y = 0.0000000 Z = 0.4500000

(أ) ٥٢–٦

```
PROCEDURE SUB (J , K : INTEGER ;
                  VAR JSUM , JPROD , JDIFF : INTEGER) ;
BEGIN
      JSUM := J + K ;
      JPROD := J * K ;
      JDIFF := J - K
END ;
```

(ب)

14 2 5 8 33

٥٣–٦

```
FUNCTION R (u , v , w : REAL) : REAL ;
        BEGIN
                R := SQRT (u * u + v * v + w * w)
        END ;
BEGIN
        .................
        .................
        A := x / R (x , y , z) ;
        B := R (2 * x * x , 3 * y , 5 * z) ;
        C := R (3.0 , SIN (y) , u) ;
        .................
        .................
END.
```

٥٤–٦

```
        .................
CONST
        max = 24 ;
TYPE
        Vector = Array [1..max] OF REAL ;
        .................
        .................
PROCEDURE Interchange (i , j : INTEGER ; VAR A : Array) ;
{Procedure for interchanging two elements of an array}
VAR temp : REAL ;
BEGIN
        temp := A[i] ;
        A[i] := A[j] ;
        A[j] := temp
END ;
        .................
```

٥٥–٦

```
PROGRAM P655 (INPUT , OUTPUT) ;
CONST
        m = 40 ;
TYPE
        Vector = ARRAY [1..m] OF REAL ;
VAR             i : INTEGER ;
        X , Y , Z : Vector ;
        alfa , beta , gamma : REAL ;
```

```
FUNCTION Prod (A , B : Vector ; n : INTEGER) : REAL ;
      {function subprogram to find sum of products of corresponding
            elements of two arrays}
      VAR
            i : integer ; P : REAL ;
      BEGIN
            P := 0.0 ;
            FOR i := 1 TO n DO
            P := P + A[i] * B[i] ;
            Prod := P
      END ;
BEGIN
      FOR i := 1 TO 40 DO READLN (X[i]) ;
      FOR i := 1 TO 40 DO READLN (Y[i]) ;
      FOR i := 1 TO 40 DO READLN (Z[i]) ;
      alfa := SQR (Prod (X , Z , 40)) ;
      beta := SQRT (Prod (X , X , 40)) *
                SQRT (Prod (Z , Z , 40)) /
                SQRT (Prod (X , Z , 40)) :
      gamma := Prod (X , Y , 20) + Prod (X , Z , 20) ;
      WRITELN (′alfa = ′, alfa , ′beta = ′, beta , ′gamma = ′, gamma)
END.
```

٥٦–٦

```
PROGRAM P656 (INPUT , OUTPUT) ;
CONST
      max = 80 ;
TYPE
      Vector = ARRAY [1..max] OF INTEGER ;
VAR
      N0 , N1 , N , L , i : INTEGER ;
      A : Vector ;
FUNCTION Ncount (X : Vector ; m , j , k , item : INTEGER) :
                                                INTEGER ;
      {To count the number of times an item appears between the
            elements indexed by j and k in an integer array X}
      VAR
            i , count : INTEGER ;
      BEGIN
            count := 0 ;
            FOR i :=  j TO k DO
                  IF X[i] = item THEN
```

```
                    count = count + 1 ;
            Ncount := count
        END ;
BEGIN
        FOR i := 1 TO 80 DO
            READLN (A[i]) ;
        N0 := Ncount (A , 80 , 1 , 80 , 0) ;
        N1 := Ncount (A , 80 , 1 , 80 , 1) ;
        N  := Ncount (A , 80 , 1 , 50 , 0) ;
        L  := Ncount (A , 80 , 30 , 70 , 1) ;
        WRITELN ('N0 = ', N0 , 'N1 = ', N1 , 'N = ', N , 'L = ', L)
END.
```

٥٧–٦

```
PROGRAM P657 (INPUT , OUTPUT) ;
CONST
        m = 30 ;
TYPE
        Vector = ARRAY [1..m] OR REAL ;
VAR
        k  :  INTEGER;
        t  :  REAL ;
        A :  Vector ;
FUNCTION Y(X : Vector ;
                n , mean : INTEGER) : REAL ;
        {To compute arithmetic mean or geometric mean}
        VAR
            i : INTEGER ;
            P , Sum : REAL ;
        BEGIN
            IF mean = 1 THEN
                BEGIN
                    Sum := 0.0 ;
                    FOR i := 1 TO n DO
                        Sum := Sum + X[i] ;
                    Y := Sum / n
                END
            ELSE
                BEGIN
                    P := 1.0 ;
                    FOR i := 2 TO n DO
                        P := P * X[i] ;
```

```
                    Y := EXP (1.0 / n * LN(P))
                END
    END ;
BEGIN
    FOR i := 1 TO 30 DO ;
        READLN (A[i]) ;
    READLN (k) ;
    t := Y(A , 30 , k) ;
    WRITELN (′k = ′, k , ′mean = ′, t)
END.
```

٥٨–٦

```
PROGRAM P658 (INPUT , OUTPUT) ;
    {main program to compute the composite average of
        the absolute values of the elements of two arrays}
CONST
    max = 40 ;
TYPE
    Vector = ARRAY [1..max] OF REAL ;
VAR
    n1 , n2 , i : INTEGER ;
    A , B : Vector ;
    av    :  REAL ;
FUNCTION Sum (X : Vector ; n : INTEGER) : REAL ;
    {for computing sum of absolute values of n numbers}
    VAR
        i : INTEGER ; S : REAL ;
    BEGIN
        S := 0.0 ;
        FOR i := 1 TO n DO
            S := S + ABS (X[i]) ;
        Sum := S
    END ;
BEGIN
    READLN (n1) ;
    FOR i := 1 TO n1 DO ;
        READLN (A[i]) ;
    READLN (n2) ;
    FOR i := 1 TO n2 DO ;
        READLN (B[i]) ;
    av := (Sum (A , n1) + Sum (B , n2)) / (n1 + n2) ;
    WRITELN (′Array A′) ;
```

```
    FOR i := 1 TO n1 DO ;
        WRITELN (A[i]) ;
    WRITELN (′Array B′) ;
    FOR i := 1 TO n2 DO ;
        WRITELN (B[i]) ;
    WRITELN (′Composite Average = ′, av)
END.
```

٥٩–٦

```
PROGRAM P659 (INPUT , OUTPUT) ;
    {A program that applies the iterative
        Newton - Raphson method using procedure Poly}
TYPE
    Vector = ARRAY [1..10] OF REAL ;
VAR
    Y , PY , PDY , YNEW : REAL ;
    C : Vector ;
PROCEDURE Poly (A : Vector , n : INTEGER ;
                X : REAL ;
               VAR P , PD : REAL) ;
    {procedure for evaluating a polynomial and its derivative}
    VAR
        i : INTEGER ;
    BEGIN  {evaluating the polynomial}
        P := A[1] ;
        FOR i := 1 TO n DO
            P := P + A[i+1] * EXP (i * LN (x)) ;
        {evaluate the derivative of the polynomial}
        PD := 0.0 ;
        FOR i := 1 TO n DO
            PD := PD + i * A[i+1] * EXP ((i-1) * LN(x)) ;
    END ;
BEGIN
    FOR i := 1 TO 10 DO ;
        READLN (C[i]) ;
    READLN (Y) ;
    Poly (C , 9 , Y , PY , PDY) ;
    YNEW := Y - PY / PDY ;
    WRITELN (′YNEW = ′, YNEW)
END.
```

٦٠–٦

```
PROGRAM P660 (INPUT , OUTPUT) ;
CONST
      max = 20 ;
TYPE
      Vector = ARRAY [1..max] OF REAL ;
VAR
      m , i : INTEGER ;
      xhigh , xlow : REAL ;
      X : Vector ;
PROCEDURE MaxMin (A : Vector ; n : INTEGER ;
                              VAR bigA, smallA : REAL) ;
      VAR
            n , i : INTEGER ;
      BEGIN
            bigA := A[1] ;
            smallA := A[1] ;
            FOR i := 2 TO N DO
                  BEGIN
                        IF bigA < A[i] THEN
                              bigA := A[i] ;
                        IF smallA > A[i] THEN
                              smallA := A[i]
                  END
      END ;
BEGIN
      READLN (m) ;
      FOR i := 1 TO m DO ;
            READLN (X[i]) ;
      MaxMin (X , m , xhigh , xlow) ;
      WRITELN ('xhigh = ', xhigh , 'xlow = ', xlow)
END.
```

٦١-٦

```
PROGRAM P661 (INPUT , OUTPUT) ;
      {To find which of 2 classes has the higher average}
CONST
      max = 50 ;
TYPE
      Vector = ARRAY [1..max] OF REAL ;
VAR
      nA , nB , i : INTEGER ;
      A1 , A2 : REAL ;
```

```
        SA , SB : Vector ;
FUNCTION AVG (X : Vector ; n : INTEGER) : REAL ;
        {function subprogram to compute the average of n numbers}
        VAR
                i : INTEGER ;
                Sum : REAL ;
        BEGIN
                Sum := 0.0 ;
                FOR i := 1 TO n DO
                        Sum := Sum + X[i] ;
                AVG := Sum / n
        END ;
BEGIN
        READLN (nA , nB) ;
        FOR i := 1 TO nA DO
                READ (SA[i]) ;
        FOR i := 1 TO nB DO
                READ (SB[i]) ;
                A1 := AVG (SA , nA) ;
                A2 := AVG (SB , nB) ;
                IF A1 > A2 THEN
                        WRITELN ('Class A has the higher average')
                ELSE IF A2 > A1 THEN
                        WRITELN ('Class B has the higher average')
                ELSE
                        WRITELN ('Both averages are equal')
END.
```

٦٢–٦

```
PROGRAM P62 (INPUT , OUTPUT) ;
        {to get the average of the best four scores out of five using
                the subprogram Small}
TYPE
        Vector = ARRAY [1..5] OF REAL ;
VAR
        i , j , ID : INTEGER ;
        sum , av : REAL ;
        S : Vector ;
FUNCTION Small (A : Vector ; n : INTEGER) : REAL ;
        VAR
                i : INTEGER ;
        BEGIN
```

```
            Small := A[i] ;
            FOR i := 2 TO n DO
                  IF Small > A[i] THEN
                        Small := A[i]
      END ;
BEGIN
      FOR j := 1 TO 50 DO
            BEGIN
                  READ (ID) ;
                  FOR i := 1 TO 5 DO ;
                        READ (S[i]) ;
                  sum := 0.0
                  FOR i := 1 TO 5 DO
                        sum := sum + S[i] ;
                  av := (sum - Small (S , 5)) / 4.0 ;
                  WRITELN ('ID = ', ID , 'Average = ', av)
            END
END.
```

٦٣–٦

```
PROGRAM P663 (INPUT , OUTPUT) ;
      {To check the symmetry of a matrix and find its
            transpose if not symmetric}
TYPE
      Matrix = ARRAY [1..5 , 1..5] OF REAL ;
VAR
      i , j : INTEGER ;
      B , C : Matrix ;
PROCEDURE SYM (A : Matrix , n : INTEGER ;
                              VAR K : INTEGER ;
      {subprogram to check whether a square matrix is symmetric}
      VAR
            i , j : INTEGER ;
      Begin
            i := 1 ; K := 1 ;
            While (i <= n) and (K <> 0) do
                  begin
                        j := 1 ;
                        while (j <= n) and (K <> 0) do
                              begin
                                    if A[i,j] <> A[j,i] then
                                          K := 0 ;
```

```
                                    j := j+1
                                end ;
                            i := i+1
                        end
        End ;
BEGIN
        FOR i := 1 TO 5 DO
                BEGIN
                        FOR j := 1 TO 5 DO
                                READ (B[i,j]) ;
                        READLN
                END
                SYM (B , 5 , K) ;
                IF K = 0 THEN
                        BEGIN
                                FOR i := 1 TO 5 DO
                                FOR j := 1 TO 5 DO
                                        C[i,j] = B[j,i] ;
                                WRITELN (′B is not symmetric′) ;
                                WRITELN (′Its transpose is : ′) ;
                                FOR i := 1 TO 5 DO
                                        BEGIN
                                                WRITELN ;
                                                FOR j := 1 TO 5 DO
                                                        WRITE (C[i,j])
                                        END
                        END
                ELSE
                        WRITELN (′B is symmetric′)
END.
```

٦٤-٦

```
PROGRAM P664 (INPUT , OUTPUT) ;
TYPE
        Vector = ARRAY [1..50] OF REAL ;
VAR
        i , nbcnt : INTEGER ;
        bamean , bgmean : REAL ;
        beta : Vector ;
PROCEDURE AVNZ (A : Vector ; m , n : INTEGER ;
                                VAR Ava , Avg : REAL ;
                                VAR nz : INTEGER) ;
```

```
{to calculate the arith. and geom. averages of the first n
 elements of an array A of m numbers, and to count the
 number of zero elements of these n numbers}
VAR
      i : INTEGER ;
      sum , prod : REAL ;
BEGIN
      {calculate arith. average}
      sum := 0.0 ;
      FOR i := 1 TO n DO
            sum := sum + A[i] ;
      Ava := sum / n ;
      {Calculate geometric average}
      prod := 1.0 ;
      FOR i := 1 TO n DO
            prod := prod * A[i] ;
      Avg := EXP ((1.0 / n) * LN (PROD)) ;
      {count number of zero elements}
      nz := 0 ;
      FOR i := 1 TO n DO
            IF A[i] = 0.0 THEN
                  nz := nz + 1
END ;
```

ملاحظة : يمكن كتابة عروة FOR واحدة فقط – بدلا من ثلاث عرى –

لإيجاد كل من مجموع العناصر وحاصل ضربها وعدد العناصر التي تساوي صفرا.

```
                                                BEGIN
                                                  FOR i := 1 TO 50 DO ;
                                                READLN (beta [i]) ;
                                    AVNZ (beta , 50 , 20 , bamean , bgmean , nbcnt) ;
      WRITELN ('bamean = ', bamean , 'bgmean = ', bgmean ,
                                                'nbcnt = ', nbcnt)
                                                                END.
```

٦٥–٦

```
PROGRAM Intersection (INPUT , OUTPUT) ;
      {program to determine the intersection of two arrays using
        the subprogram : Search}
TYPE
      Vector = ARRAY [1..50] OF REAL ;
      Vector1 = ARRAY [1..30] OF REAL ;
VAR
```

```
        i , found , index : INTEGER ;
        A      :  Vector ;
        X      :  Vector1 ;
PROCEDURE Search (Buffer : Vector ;
                n : INTEGER ; Key : REAL ;
                VAR found , index : INTEGER) ;
        {this subprogram searches the array : Buffer
        (of n elements) for the given element : key}
        VAR   i : INTEGER ;
        BEGIN
                found := 0 ; i := 1 ;
                While (i <= n) and (found = 0) do
                        IF Key = Buffer [i] THEN
                                BEGIN
                                        found := 1 ;
                                        index := i ;
                                END
                        ELSE
                                        i := i + 1
        END ;
BEGIN
        FOR i := 1 TO 50 DO
                READLN (A[i]) ;
        FOR i := 1 TO 30 DO
                READLN (X[i]) ;
        FOR i := 1 TO 30 DO
                BEGIN
                        Search (A , 50 , X[i] , found , index) ;
                        IF found = 1 THEN
                                WRITELN (X[i] , index)
                END
END.
```

٦-١١

```
PROGRAM P666 (INPUT , OUTPUT) ;
TYPE
        Vector = ARRAY [1..700] OF INTEGER ;
        Vector1 = ARRAY [1..314] OF INTEGER ;
VAR   i , n3 : INTEGER ;
        Badr , BO : Vector1 ;
        Ohod : Vector ;
PROCEDURE Inter (Jset : Vector1 , Kset : Vector ;
```

```
                    n1 , n2 : INTEGER ;
                    VAR JKset : Vector1 ;
                    VAR n3 : INTEGER) ;
    {Subprogram to find intersection of two arrays and
        number (n3) of common elements}
    VAR
        i , j : INTEGER ; found : boolean
    BEGIN
        n3 : = 0 ;
        FOR i := 1 TO n1 DO
            BEGIN
                j := 1 ; found := false ;
                While (j <= n2) and (not found) do
                    IF Jset[i] = Kset[j] THEN
                        BEGIN
                            n3 := n3 + 1 ;
                            JKset[n3] := Jset[i] ;
                            found := true
                        END ;
                    ELSE
                            j := j+1 ;
            END
    END ;
BEGIN
    FOR i := 1 TO 314 DO
        READLN (Badr[i]) ;
    FOR i := 1 TO 700 DO
        READLN (Ohod[i]) ;
    Inter (Badr , Ohod , 314 , 700 , BO , n3)
    FOR i := 1 TO n3 DO
        WRITELN (BO[i])
END.
```

٦٧–٦

```
PROGRAM P667 (INPUT , OUTPUT) ;
    {evaluating the area under a curve by applying the
        trapezoidal rule and using a function subprogram}
VAR
    i , n : INTEGER ;
    a , b , x , h , area : REAL ;
```

```
FUNCTION F (x : REAL) : REAL ;
        BEGIN
                F := EXP (-x * x) + 3.0 * EXP (3 * LN(x))
                        - 4.0 * SQR (x) + 6.0 * x + 5.0
        END ;
BEGIN
        READLN (a , b , n) ;
        h := (b - a) / n ;
        sum := 0.0 ;
        FOR i := 1 TO (n-1) DO
                sum := sum + F (a + i * h) ;
        area := h / 2.0 * (F(a) + F(b) + 2.0 * sum) ;
        WRITELN (′area = ′ , area)
END.
```

٦٨–٦

```
PROGRAM P668 (OUTPUT) ;
        {main program that calls the function subprogram CNR ,
          which in turn calls the function subprogram Fact}
VAR     a , b , c : REAL ;
FUNCTION Fact (k : INTEGER) : REAL ;
        {Function subprogram for computing the factorial}
        VAR
                i : INTEGER ;  F : real ;
        BEGIN
                F := 1.0 ;
                FOR i := 2 TO k DO
                        F := F * i ;
                Fact := F
        END ;
FUNCTION CNR (n , r : INTEGER) : REAL ;
        {Function subprogram that calls the function subprogram Fact}
        BEGIN
                CNR := Fact(n) / (Fact(r) * Fact (n-r))
        END ;
BEGIN
        a := CNR (4,1) ;
        b := CNR (5,3) ;
        c := CNR (7,7) ;
        WRITELN (′a = ′, a , ′b = ′, b , ′c = ′, c)
END.
```

٦٩–٦

```
PROGRAM FixedPointMethod (INPUT , OUTPUT) ;
        {Also called method of successive approximations ,
          for solving the equation f(x) = 0}
VAR i , n : INTEGER ;
        x1 , x2 , eps : REAL ;
FUNCTION F(x : REAL) : REAL ;
        BEGIN
                F := SQR(x) - SQRT(x) - 3.0
        END ;
FUNCTION G (x : REAL) : REAL ;
        BEGIN
                G := SQRT(SQRT(x) + 3.0)
        END ;
BEGIN
        READLN (x1 , n , eps) ;
        i := 1 ;
        x2 := G(x1) ;
        while (i <= n) and (abs(x2 - x1) >= eps) do
                begin
                        x1 := x2 ;
                        i := i + 1;
                        x2 := G(x1)
                end ;
        if i <= n then
                WRITELN (x2 , F(x2))
        else
                WRITELN (′Fails to converge in n iterations′) ;
END.
```

٧٠–٦

```
PROGRAM Recursion (INPUT , OUTPUT) ;
CONST
        eps = 1E - 09 ;
VAR
        k : INTEGER ;
        term : REAL ;
FUNCTION Fact (k : INTEGER) : REAL ;
        BEGIN
                IF k = 0 THEN
                        Fact := 1.0
```

```
        ELSE
            Fact := k * Fact (k-1)
    END ;
FUNCTION Sum (k : INTEGER) : REAL ;
    BEGIN
        IF k = 0 THEN
            Sum := 1.0
        ELSE
            Sum := Sum (k-1) + 1.0 / Fact (k)
    END ;
BEGIN
    k := 0 ;
    REPEAT
        k := k + 1 ;
        term := 1.0 / Fact (k)
    UNTIL
        term < eps ;
    WRITELN ('e = ', Sum (k))
END.
```

٧١–٦

```
Program PropertyValues ;
                                                    {
This program determines the tax owed on each of several properties.
Each property's market value is read, its assessed value computed as
28% of the estimated market value and its tax computed and
displayed at the rate of 12.5 cents per dollar of assessed value. Input
is terminated by a sentinel of 0. After all values are read, the total tax
                                                    to be collected is printed.
                                                    }
                                                  const
                                {terminates input}   Sentinel = 0.0  ;
                                                    var
                            {sum of all property taxes} TotalTax : Real ;
                                                procedure PrintInstructions ;
                                              {Display instructions to the user.}
                                                  begin {PrintInstructions}
        WriteLn ('When prompted, please enter property' 's') ;
          WriteLn ('market value in dollars and then press') ;
          WriteLn ('return. To terminate the program, enter') ;
                    WriteLn ('a value of ', Sentinel : 4:2, '.') ;
```

```
                                        WriteLn
                                    end  ; {PrintInstructions}
                            function ComputeTax (Market : Real) : Real ;
                        {Compute tax on property with market value Market.}
                                                                    const
            {tax rate in $}                          TaxRate = 0.125 ;
        {% market assessed val}          AssessedPercent = 0.28 ;
                                                                      var
                {assessed value}            AssessedValue : Real ;
                                                  begin {ComputeTax}
                    AssessedValue := Market * AssessedPercent ;
                        ComputeTax := AssessedValue * TaxRate
                                                  end ; {ComputeTax}
    procedure ProcessProperties (var TotalTax {output} : Real) ;
                        {Read and compute taxes for all properties.}
                                                                      var
                {input - property value}     MarketValue : Real ;
                {output - computed tax}                Tax : Real ;
                                               begin {ProcessProperties}
                                        {Read first MarketValue}
    Write (′Enter first value (′,Sentinel : 4 : 2, ′to quit) : ′) ;
                                            ReadLn (MarketValue) ;
                                                {Initialize TotalTax}
                                                    TotalTax := 0.0 ;
                                    {Process all values until Sentinel is read}
                                        while MarketValue <> Sentinel do
                                                                begin
                                            {Compute and display tax}
                                    Tax := ComputeTax (MarketValue) ;
                                    WriteLn (′The tax is $′, Tax: 4: 2) ;
                                                {Update TotalTax}
                                        TotalTax := TotalTax + Tax ;
                                                {Get next MarketValue}
    Write (′Enter next value (′, Sentinel:4:2, ′to quit): ′);
                                            ReadLn (MarketValue)
                                                            end {while}
                                                    end ; {ProcessProperties}
                                procedure PrintSummary (TotalTax : Real) ;
                                                {Print the value of TatalTax.}
                                                    begin {PrintSummary}
                                WriteLn (′The total tax due is $′, TotalTax: 4: 2)
```

```
end ; {PrintSummary}
begin {MarketValues}
PrintInstructions ;
ProcessProperties (TotalTax) ;
PrintSummary (TotalTax)
end. {MarketValues}
```

٦-٧٢

فيما يلي التعديلات المطلوبة على حل السؤال السابق.

```
function ComputeTax (Market : Real ;
Senior : Boolean) : Real ;
const
TaxRate = 0.125 ;          {tax rate in $}
AssessedPercent = 0.28 ;   {% market - assessed}
SurchargeRate = 0.05 ;     {% market}
SeniorDiscount = 0.10 ;    {% tax due}
Var
TaxDue,
AssessedValue : Real ;
begin {ComputeTax}
AssessedValue : = Market * AssessedPercent ;
TaxDue := AssessedValue * TaxRate ;
{add surcharge on expensive properties}
if Market > 50000.00 then
TaxDue := TaxDue + SurchargeRate * Market ;
{apply senior citizen discount if applies}
if Senior then
TaxDue := TaxDue - TaxDue * SeniorDiscount ;
ComputeTax := TaxDue
end ; {ComputeTax}

Procedure ReadProperty
(var MarketValue {output} : Real ;
var Senior {output} : Boolean) ;
var
Ch : Char ; {input - question response}
begin {ReadProperty}
Write {'Enter market value (', Sentinel : 4: 2, ' to quit): ') ;
ReadLn (MarketValue) ;
Senior := False ;
{Check for Senior citizen discount}
```

```
                    if MarketValue <> Sentinel then
                                        begin
  Write ('Is owner 65+ years old (Y/N)? ') ;
                                    ReadLn (Ch) ;
        Senior := (Ch = 'Y') or (Ch = 'Y')
                                          end ;
                                  end ; {ReadProperty}
procedure ProcessProperties (var TotalTax {output} : Real) ;
                                                  var
    {input - property value}          MarketValue,
    {output - computed tax}               Tax : real ;
       {age over 65?}               Senior : Boolean ;
                                {ProcessProperties} begin
                            {Read first MarketValue}
                    Readproperty (MarketValue, Senior) ;
                                {Initialize TotalTax}
                                    TotalTax := 0.0 ;
                {Process all values until Sentinel is read}
                                  while MarketValue <> Sentinel do
                                                      begin
                                    {Compute and display tax}
          Tax := ComputeTax (MarketValue, Senior) ;
                    WriteLn ('The tax is $', Tax: 4: 2) ;
                                        {Update TotalTax}
                                TotalTax := TotalTax + Tax ;
                                    {Get next MarketValue}
                    ReadpProperty (MarketValue, Senior) ;
                                                    end {while}
                                  end ; {ProcessProperties}
```

٦–٧٣

```
                                  program RaiseAnalysis ;
                                                      {
This program analyzes the effect of granting a 5.5% pay raise
to the faculty. Each salary is input and the individual raise is printed.
After all salaries are read, the total payroll before and after the raise,
                                  and the total amount of raises, are displayed.
                                                      }
                                                  const
    {terminates input}              Sentinel = 0.0  ;
                                                  var
  {output - payroll before raise}       OldTotalPay,
```

```
{output - payroll after raise}          NewTotalPay,
 {output - sum of all raises}       TotalRaise : Real ;
                                 procedure ProcessSalaries
                        (var OldTotalPay {output} : Real ;
                        var NewTotalPay {output} : Real ;
                        {output} : Real) ;    var TotalRaise
                                {Read and process all salaries.}
                                                        var
  {input - individual salary}           Salary : Real ;
 {output - Computed Raise}               Raise : Real ;
          procedure ComputeRaise (Salary {input} : Real ;
        var Raise {output} : Real) ;
              {Compute raise for one faculty salary.}
                                                      const
 {uniform raise fraction}              RaiseRate = 0.055 ;
                                        begin {ComputeRaise}
                              Raise := Salary * RaiseRate ;
                                       end ; {ComputeRaise}
                                     begin {ProcessSalaries}
                                  {Initialize accumulators}
                                  := 0.0 ;       OldTotalPay
                                  := 0.0 ;     NewTotalPay
                                  := 0.0 ;          TotalRaise
                                          {Get first salary}
  Write (′Enter first salary (′, Sentinel : 4:2, ′ to quit) : ′) ;
                                        ReadLn (Salary) ;
                                {process salaries until sentinel}
                                   while Salary <> Sentinel do
                                                      begin
                          {Compute raise and update totals}
                            ComputeRaise (Salary, Raise) ;
                  OldTotalPay := OldTotalPay + Salary ;
  NewTotalPay := NewTotalPay + Salary + Raise ;
                  TotalRaise := TotalRaise + Raise ;
                                        {Display raise}
                WriteLn {′Raise amount is $′, Raise: 4: 2) ;
                                        {Get next salary}
 Write (′Enter next salary (′, Sentinel :4:2, ′ to quit): ′) ;
                                        ReadLn (Salary)
                                                 end {while}
                                      end ; {ProcessSalaries}
```

```
procedure PrintReport (OldTotalSalary {input} : Real ;
NewTotalSalary {input} : Real ;
TotalRaise {input}  ; Real) ;
{Print a summary report of the effect of the raise.}
begin {PrintReport}
WriteLn ;
WriteLn (′total payroll before raise = $′, OldTotalSalary : 4: 2)
;
WriteLn (′Total Payroll after raise = $′, NewTotalSalary : 4:2) ;
WriteLn (′Total amount of raise = $′, TotalRaise : 4:2)
end ; {PrintReport}
begin {RaiseAnalysis}
{Read Salaries, compute and print raises, and compute totals}
ProcessSalaries (OldTotalPay, NewTotalPay, TotalRaise) ;
{Display Summary Report}
PrintReport (OldTotalPay, NewTotalPay, TotalRaise)
end. {RaiseAnalysis}
```

٧٤–٦

```
procedure ComputeRaise (Salary {input} : Real ;
var Raise {output} : Real) ;
{Compute raise for one faculty salary.}
const
{7 percent raise cutoff}    LowSalary = 30000 ;
{4 percent raise cutoff}    MidSalary = 40000 ;
{low salary raise}   LowRaisePct = 0.07 ;
{middle salary raise}  MidRaisePct = 0.055 ;
{high salary raise}   HighRaisePct = 0.04 ;
begin {ComputeRaise}
if Salary < LowSalary then
Raise := Salary * LowRaisePct
else if Salary <= MidSalary then
Raise := Salary * MidRaisePct
else
Raise := Salary * HighRaisePct
end ; {ComputeRaise}
```

٧٥–٦

```
program Medications ;
{
This program prints a table showing what medication to take at
any given hour in a 24-hour day, according to the prescription
```

```
specified in the problem statement. The table is printed with 24 rows,
                                                    one for each hour.
                                                                    }
                                                                const
    {Width of a column of the table}     ColumnWidth = 12 ;
    {Width of hour column of table}          HourWidth = 4 ;
                                                                  var
                        Hour : Integer ; {Output - hour of day}
                                                 procedure PrintHeader ;
                                                     {print table header}
                                                   begin {PrintHeader}
                                                        WriteLn ;
                    MEDICATION CHART′) ;          WriteLn (′
                                                       WriteLn
                                                  end ; {PrintHeader}
                          procedure PrintHour (Hour {input} : Integer) ;
                                                                    {
Assuming Hour is given in military time, it is converted to AM-PM
                                                        time and printed.
                                                                    }
                                                    begin {PrintHour}
                                                  if Hour < 12 then
                                Write (Hour: HourWidth, ′ AM′
                                                 else if Hour = 12 then
                                                       {noon}
                                Write (Hour : HourWidth, ′ PM′)
                                                               else
                                                {Assert: 13 < = Hour <= 24}
                                                              begin
                                                 Hour := Hour - 12 ;
                                                   if Hour < 12 then
                                Write (Hour: HourWidth, ′ PM′)
                                                           else
                                                       {midnight}
                                Write (Hour: HourWidth, ′ AM′)
                                                        end {else}
                                                    end ; {PrintHour}
                                 procedure Iron (Hour {input} : Integer) ;
                               {print message if iron pill prescribed at this hour.}
                                                                begin{Iron}
                                                  if Hour = 8 then
```

```
                Write ('iron' : ColumnWidth)
                    else if Hour = 12 then
                Write ('Iron' : ColumnWidth)
                    else if Hour = 18 then
                Write ('Iron' : ColumnWidth)
                                    end ; {Iron}
            procedure Antibiotic (Hour {input} : Integer) ;
                                    {
        Print message if antibiotic prescribed at this hour.
            Hour has been assigned a value.   Pre:
Print 'Antibiotic' if hour is every 4 hours starting at 0400.  Post:
                                    }
                                begin {Antibiotic}
                                if Hour = 4 then
                Write ('Antibiotic' : ColumnWidth)
                                  else if Hour = 8 then
                Write ('Antibiotic' : ColumnWidth)
                                else if Hour = 12 then
                 Write ('antibiotic' : ColumnWidth)
                                else if Hour = 16 then
                 Write ('antibiotic' : ColumnWidth)
                                else if Hour = 20 then
                Write ('Antibiotic' : ColumnWidth)
                                else if Hour = 24 then
                Write ('Antibiotic' : ColumnWidth)
                                    end ; {Antibiotic}
            procedure Vitamin (Hour {input} : Integer) ;
                                    {
    Print message if vitamin pill is prescribed at this hour
                                    }
                                  begin {Vitamin}
                                if Hour = 8 then
                Write ('Vitamin' : ColumnWidth)
                                else if Hour = 21 then
                Write ('Vitamin' : ColumnWidth)
                                    end ; {Vitamin}
            procedure Calcium (Hour {input} : Integer) ;
                                    {
      Print message if Calcium pill prescribed this hour.
            Hour has been assigned a value.   Pre:
            Print 'Calcium' if hour is 1100 or 2000.  Post:
```

```
                                                    }
                                                begin {Calcium}
                                                    if Hour = 11 then
                                        Write ('Calcium' : ColumnWidth)
                                                    else if Hour = 20 then
                                        Write ('Calcium' : Columnwidth)
                                                end ; {Calcium}
                                            begin {Medications}
                                                PrintHeader ;
                                        {Print each line of table}
                                                Hour : = 1 ;
                                            while Hour <= 24 do
                                                begin
                                    PrintHour (Hour) ;
                                        Iron (Hour) ;
                                    Antibiotic (Hour) ;
                                      Vitamin (Hour) ;
                                      Calcium (Hour) ;
                                            WriteLn ;
                                     Hour := Hour + 1
                                            end {while}
                                        end. {Medications}
```

٧٦–٦

```
                                        Program Subscriptions ;
                                                {
This program accepts information about magazine subscriptions, and
                                                prints a message about the subscription.
Inputs are:  Current month (1-12) and year (2 digits), and for each
subscription the account number, the starting month and year and
                                                number of years subscribed.
Output is either a warning message if the subscription expires this
month or next month, a cancellation notice if the subscription has
                                                expired, or no message otherwise.
                                                }
                                                var
                {input - Current  Month}     CurMonth : Integer ;
                  {input - Current Year}       CurYear : Integer ;
                                            procedure ReadCurrentDate
                    {output} : Integer ; CurMonth    (var
                   {output} : Integer) ;     CurYear    var
                        {Get current month and year from the user.}
```

```
begin {ReadCurrentDate}
Write ('Enter Current Month (1-12) and year') ;
('(2 digits) : ') ;Write
ReadLn (CurMonth, CurYear)
end {ReadCurrentDate}

procedure ProcessSubscriptions
(CurMonth {input},
CurYear {input} : Integer) ;
var
{input - account number} AccountNo,
{input - starting month} Month,
{input - starting year} Year,
{input - number of years} YearCount : Integer ;

procedure ReadAccountNo (var AccountNo {output} : Integer) ;
{Get account number of subscription.}
begin {ReadAccountNo}
Write ('Enter account number (0 to quit): ') ;
ReadLn (AccountNo)
end ; {ReadAccountNo}

procedure ReadSubRest
(var Month, Year {output} : Integer ;
var YearCount {output} : Integer) ;
{Get remainder of subscription data from user.}
begin {ReadSubRest}
Write ('Enter month subscription began (1-12) : ') ;
ReadLn (Month) ;
Write ('Enter Year subscription began (2 digit): ') ;
ReadLn (Year) ;
Write ('Enter number of years subscribed: ') ;
ReadLn (YearCount)
end ; {ReadSubRest}

procedure PrintMessage
(CurMonth, CurYear {input} : Integer ;
AccountNo {input} : Integer ;
Month, Year, YearCount {input} : Integer) ;
{
```

```
Print Appropriate Message. Implementation note: we compute
the number of months since the start of the century until (a) now and
(b) the end of the subscription. The two numbers are compared and
the appropriate message is printed. This trick gets around the messy
                    cases when a subscription expires in January.
                                                                  }
                                                                var
        {Months from 1/1/1900 to end}          EndingMonth,
      {Months from 1/1/1900 to now}   ThisMonth : Integer ;
                                              begin {PrintMessage}
    EndingMonth := 12 * (Year + YearCount) + Month ;
          ThisMonth := 12 * CurYear + CurMonth ;
                  if EndingMonth = ThisMonth then
     WriteLn ('Print Renewal Notice for account',
                        AccountNo: 1, '.')
             else if EndingMonth = ThisMonth + 1 then
     WriteLn ('Print Renewal Notice for account',
                        AccountNo: 1, '.')
               else if EndingMonth < ThisMonth then
      WriteLn ('Cancel Subscription for account',
                        AccountNo: 1, '.')
                                    end ; {PrintMessage}
                                begin {ProcessSubscriptions}
                        ReadAccountNo {AccountNo} ;
                            While AccountNo <> 0 do
                                        begin
        ReadSubRest (Month, Year, YearCount) ;
PrintMessage {CurMonth, CurYear, AccountNo,
    Month, Year, yearCount) ;
                ReadAccountNo (AccountNo)
                                    end {while}
                                end ; {ProcessSubscriptions}
                                  begin {Subscriptions}
                ReadCurrentDate {CurMonth, CurYear) ;
              ProcessSubscriptions (CurMonth, CurYear)
                                  end. {Subscriptions}
```

٦–٢٢

```
                                  program Driver ;
                                                  {
```

```
This program is a driver to test procedure SquareRoot. user allowed
                                                    to input several sets of test data.
                                                                                    }
                                                                               const
                                                        Sentinel = 0.0 ; {terminates input}
                                                    Delta = 0.005 ; {tolerance for square root}
                                                                                  var
                {input - number to find root of}                        N,
                    {input - guess of root of N}                    Guess,
            {output - approx square root of N}   SqrRoot : Real ;
function FindSqrRoot (N {input} : Real ; Guess {input} : real) : Real
                                                                                    ;
                    {Compute an approximation to the square root of N.}
                                                                               const
                    Delta = 0.005 ; {tolerance for difference of terms}
                                                                                  var
                        NextGuess, {computed approx next iteration}
                        {approx computed previously} LastGuess,
                        Difference : Real ; {diff of successive approx}
                                                              begin {FindSqrRoot}
                                {Initialize first two terms of the sequence}
                                                            LastGuess := Guess ;
                                NextGuess : = 0.5 * (LastGuess + N / LastGuess) ;
                                    Difference := NextGuess - LastGuess ;
                                                        {Make difference positive}
                                                            if Difference < 0 then
                                                Difference := - Difference ;
{Repeat unitl difference between successive approximations is small}
                                                        while difference > Delta do
                                                                              begin
                                                    {Update LastGuess, NextGuess}
                                                        LastGuess := NextGuess ;
                                NextGuess := 0.5 * (LastGuess + N / LastGuess) ;
                                    {Compute absolute value of the difference}
                                    Difference := NextGuess - LastGuess ;
                                                            if Difference < 0 then
                                                Difference := - Difference
                                                                        end ; {while}
                                                    FindSqrRoot := NextGuess
                                                            end ; {FindSqrRoot}
                                                                begin {Driver}
```

```
                                {Get first number}
        Write (′Enter Number whose square root is to be′,
                                ′ found (0 to quit) : ′) ;
                                ReadLn (N) ;
                                while N <> 0.0 do
                                    begin
    Write (′Enter guess for square root of number: ′) ;
                                ReadLn (Guess) ;
        SqrRoot := FindSqrRoot (N, Guess) ;
    WriteLn (′The square root is approximately′,
                            SqrRoot : 7: 5) ;
    Write (′Enter Number whose square root is to be′,
                        ′ found (0 to quit): ′) ;
                                ReadLn (N)
                                end {while}
                                end. {Driver}
```

٧٨–٦

```
                                program Midpoint ;
                                {
Program uses the midpoint method to determine an approximateion
                    to the definite integral of f(X), over the interval [A, B].
                                }
                                var
                        N : Integer ; {Input - # rectangles used}
                            A, B, {input - bounds of integration}
                        Area : Real ; {output - area under F(X)}
                                function F (X : Real) : Real ;
                        {Computes value of function under study.}
                                begin {F}
                                F := -3 * Sqr(X) + 2 * X + 4
                                end ; {F}

        function ComputeArea (A, B : Real ; N : Integer) : Real ;
                                {
Uses midpoint method to approximate area under F(X) over [A, B]
                    (the definite integral) using N rectangles.
                                }
                                var
                    {loop control}      I : Integer ;
            {size of rectangle base}        W,
```

```
TempSum : Real ;  {partial sum of rectangles}
                                    begin {ComputeArea}
                                  {Compute width}
                                 W :=  (B - A) / N ;
                               {Initialize accumulator}
                                    TempSum := 0 ;
                          {Compute sum of rectangle areas}
                                  for I := 0 to N - 1 do
                        TempSum := TempSum +
                            Abs (W * F(A + I * W + W / 2.0)) ;
                              ComputeArea := TempSum
                                    end ; {ComputeArea}
                                       begin {Midpoint}
                           {Get approximation parameters.}
                  Write (′enter left endpoint of interval > ′) ;
                                    ReadLn (A) ;
                 Write (′Enter right endpoint of interval > ′) ;
                                    ReadLn (B) ;
               Write (′Enter number of rectangles to use > ′) ;
                                    ReadLn (N) ;
                        {Compute value of approximation.}
                          Area := ComputeArea (A, B, N) ;
                           WriteLn (′Area is′, Area : 6 : 4)
                                     end. {Midpoint}
```

أجوبة الاختبارات العامة

حل الاختبار الفصلي الأول رقم (١)

1) a) i) 9.5 , real.
ii) 6 , integer.
iii) 12 , integer.
iv) true , boolean.

b) i) 2.0E-1 * sqrt ((x - y) / (x + y)) ;
ii) x / y + z * z * z ;
iii) sqr (x - sqrt (x / (y + z))) ;

c) i) (m >= -10) and (m <= 10)
ii) m mod n = 0
iii) Abs (x - y) < 1E - 3

2) a) ∇y - x∇ = ∇∇∇2.00

b) partly cloudy ∇temperature is ∇35.4∇C

3)
```
program VolSurface (input , output) ;
    const Pi = 3.14159 ;
    var R , V , S : real ;
        ch        : char ;
    begin
        writeln (′enter R , ch′) ;
        readln (R , ch) ;
        if (ch = ′V′) or (ch = ′v′) then
            begin
                V := 4/3 * Pi * R * R * R ;
                writeln (′Volume = ′, V)
            end
        else if (ch = ′S′) or (ch = ′s′) then
            begin
                S := 4*Pi*R*R ;
                writeln (′Surface Area = ′, S)
            end
        else
            writeln (′wrong selection character′)
    end.
```

4)
```
program Sum (input , output)
    var
        i , m , n , s : integer ;
    begin
```

```
S := 0 ;
writeln (′enter m , n′) ;
readln (m , n) ;
for i := m to n do
S := S + i ;
writeln (′s = ′ , s)
end.
```

حل الاختبار الفصلي الأول رقم (٢)

1)

a) **خاطئة : لا يجوز إسناد قيمة تعبير حقيقي لمتغير صحيح**

b) **خاطئة : المؤثر المنطقي** and **يربط بين شرطين وليس بين رمزين**

c) **خاطئة : الطرف الأيسر في عبارة الاسناد يجب أن يكون متغيرا ، بينما** I **ثابت**

d) **صحيحة**

e) **خاطئة :** mod **تستخدم مع صحيحين بينما** R1 **حقيقي**

f) **خاطئة :** while **يأتي بعدها شرط منطقي بينما** I **ثابت صحيح**

g) **خاطئة : لا يجوز قراءة قيمة متغير منطقي** B1 (boolean)

h) **صحيحة**

2)

```
program Question2 (Input , Output) ;
const
Small = 0.0001 ;
var
x1 , x2 , x3 , y1 , y2 , y3 ,
LX , LY , XdotY , CosAlpha : real ;
begin
write (′Enter in the 6 real numbers : ′) ;
readln (x1 , x2 , x3 , y1 , y2 , y3) ;
LX := sqrt (x1*x1 + x2*x2 + x3*x3) ;
LY := sqrt (y1*y1 + y2*y2 + y3*y3) ;
XdotY := x1*y1 + x2*y2 + x3*y3 ;
if (LX <= Small) or (LY <= Small) then
CosAlpha := 1.0
else
CosAlpha := XdotY / (LX*LY) ;
writeln (′LX = ′, LX : 9 : 3) ;
```

```
writeln (′LY = ′, LY : 9 : 3) ;
writeln (′XdotY = ′, XdotY : 9 : 3) ;
writeln (′cos(alpha) = ′, CosAlpha : 9 : 6) ;
end.
```

∇6--∇∇5--∇∇11 3)
∇4--∇∇3--∇∇18
∇1--∇∇0--∇∇19

4)

```
program Question4 (Input , Output) ;
var
NumSt , StID : Integer ;
Score1 , Score2 , Score3 , Average : Real ;
TotalScore1 , TotalScore2 , TotalScore3 : Real ;
AvgSore1 , AvgScore2 , AvgScore3 , AvgAll : Real ;
begin
write (′Enter in the Student ID : ′) ;
readln (StID) ;
NumSt := 0 ;
TotalScore1 := 0.0 ;
TotalScore2 := 0.0 ;
TotalScore3 := 0.0 ;
while (StID <> 0) do
begin
write (′Enter 3 acores for Student ID ′, StID,′ : ′) ;
readln (Score1 , Score2 , Score3) ;
Average := (Score1 + Score2 + Score3) / 3 ;
writeln (′student ID =′, StID:5′Average =′, Average:6:2);
NumSt := NumSt + 1 ;
TotalScore1 := TotalScore1 + Score1 ;
TotalScore2 := TotalScore2 + Score2 ;
TotalScore3 := TotalScore3 + Score3 ;
write (′Enter in the Student ID : ′) ;
readln (StID) ;
end ;
AvgScore1 := TotalScore1 / NumSt ;
AvgScore2 := TotalScore2 / NumSt ;
AvgScore3 := TotalScore3 / NumSt ;
AvgAll := (AvgScore1 + AvgScore2 + AvgScore3) / 3 ;
writeln (′Average of the first test is : ′ , AvgScore1 : 6 : 2) ;
writeln (′Average of the second test is : ′ , AvgScore2 : 6 : 2) ;
writeln (′Average of the third test is : ′ , AvgScore3 : 6 : 2) ;
```

end.

حل الاختبار الفصلي الأول رقم (٣)

1)

a) خاطئ
b) خاطئ
c) سلسلة رموز
d) خاطئ
e) سلسلة رموز
f) حقيقي
g) خاطئ
h) خاطئ

2)

A = 1
B = 4
C = 5
X = 7.1
Y = 8.0
Z = 4.5

يحدث خطأ عند محاولة إعطاء القيمة الحقيقية 5.5 للمتغير الصحيح A

3)

a) 15.5
b) 75
c) 3.0

4)

a) Abs ((a-b) / 2) <> 1/c ;
b) Const Pi = 3.14159 ;
Pi * sqrt (sqr(x - Alfa) + sqr (y - Beta))

5)

لا يجوز تعريف ثابت معين بعبارة إسناد — j := 6 ;

هذا تعريف متغير وليس تعريف ثابت — star : char ;

لا يجوز استخدام الكلمة المحجوزة end كاسم متغير — end : integer ;

لا يجوز أن يبدأ اسم متغير برقم — 2q : real ;

p متغير حقيقي فلا يجوز استخدامه مع المؤثر mod — x := x mod p ;

لا معنى لإجراء عملية ضرب على المتغير الرمزي C1(char) — C1 := C1*1 ;

قيمة y غير معلومة ، وكذلك يجب وضع قوسين حول الشرط x + y > 4 — if flag and x+y > 4 then

يجب وضع حاصرتين ('it is true') — writeln (it is true)

no ثابت وليس متغيرا فلا يجوز أن يكون الطرف no := false

الأيسر في عبارة إسناد

لا يجوز إسناد قيمة متغير حقيقي p لمتغير صحيح x x := p

6)

```
Begin
    if A > B then
        begin
            R := 0 ;
            S := S+1
        end
    else
        begin
            R := R+1 ;
            S := 0
        end ;
```

7)

```
Program check (input , output) ;
Var
    grade : integer ;
Begin
    writeln (′enter grade′) ;
    readln (grade) ;
    if (grade >= 0) and (grade <= 100) then
        writeln (′valid grade′)
    else if grade < 0 then
        writeln (′invalid grade , less then zero′)
    else
        writeln (′invalid grade , greater then 100′)
End.
```

حل الاختبار الفصلي الأول رقم (٤)

(١

```
Writeln (′Zakat of Money is′ , P : 4 : 1, ′ percent of savings′) ;
Writeln (′Zakat of God is′,P:4:1,′ percent of the pure amount of gold′);
Writeln (′Zakat of Fruit is′,I:3,′ percent of the amount of fruits′);
Writeln (′Zakat of Rikaz (buried treasures, minerals, petroleum) is ′,
                                                    J:3,′ percent ′);
```

2) i = 3.0 , x = 12.0 , y = 2.0 , z = 36.0

3) i) KE := 0.5 * m * v * v ;

ii) D := sqrt (sqr(x1 - x2) + sqr (y1 - y2)) ;
iii) V := 3.14159 * r * r * h ;
iv) s := (a + b + c) / 2.0 ;
Area := sqrt (s*(s-a)*(s-b)*(s-c)) ;
v) Theta := 5/9 * (Fi - 32) ;

4) a)
```
if N mod 2 = 0 then
        writeln (′N is even′)
else
        writeln (′N is odd′) ;
```

b) 1) x = 2 القيمة النهائية

2) x = 4 القيمة النهائية

5)
(true
(false
(false
(true
(true
(true
(false
(true
(false

6)
```
Program Money (input , output) ;
Const Percent = 0.025 ;
Var   Savings , Charities , Zakat , Net : real
Begin
      Writeln (′Enter amount of Savings′) ;
      Readln (Savings) ;
      Writeln (′Enter amount of Charities′) ;
      Readln (Chairties) ;
      Zakat := Savings * Percent ;
      Net := Savings - Zakat - Charities ;
      Writeln (′Savings = K.D.′, Savings) ;
      Writeln (′Net = K.D.′, Net)
End.
```

7)
```
Program FloatOrSink (input , output) ;
Var   m , v , d : real ;
begin
      writeln (′enter mass and volume ′) ;
      readln (m , v) ;
```

```
        d := m / v ;
        if d < 1 then
                writeln (′object will float′)
        else
                writeln (′object will sink′)
end.
```

8)

```
Program HeartBeats (input , output) ;
Const
        DPY = 365.25 ;
        Rate = 75 ;
Var
        Age : integer ;     HB : real ;
Begin
        Writeln (′Enter age in years′) ;
        Readln (Age) ;
        HB := Age*DPY*24*60*Rate ;
        Write (′For an age of ′, Age :3 ,′,the number′) ;
        Writeln (′of heart beats is : ′, HB)
End.
```

حل الاختبار الفصلي الأول رقم (٥)

1) (1.5 , real .
(true , boolean.
(-36 , integer.

2) i) I = 3
(mod خطأ : لا يسمح بمتغير حقيقي (Y) مع المؤثر
(خطأ : لا يجوز إسناد قيمة تعبير حقيقي لمتغير صحيح
(X = 3.0
(div خطأ : لا يسمح بثابت حقيقي (Pi) مع المؤثر
(X = 0.0
(خطأ : لا يجوز القسمة على صفر

3)

القيمة الابتدائية	القيمة النهائية
$X_{initial}$	X_{final}

	(i)	(ii)
0	12	0
1	2	12
9	10	12

4) Backwards they are ΔΔ55.380ΔΔΔ34ΔΔ-12
In order they are -12ΔΔ34ΔΔ55.4

5) i) RegPay = 8E + 2 OverTime = 1.8E + 3
GrossPay = 2.6E + 3 NetPay = 2.0E + 3
ii) GrossPay = 6.0E + 2 NetPay = 3.0E + 2

6)

```
Program RightTriangle (input , output) ;
var
      A , B , H : real ;
begin
      writeln (′enter A , B′) ;
      readln (A , B) ;
      H := sqrt (sqr(A) + sqr (B)) ;
      writeln (′hypotenuse = ′, H)
end.
```

7)

```
Program Cone (input , output) ;
const Pi = 3.14159 ;
var
      h , r , v : real ;
begin
      writeln (′enter h , r′) ;
      readln (h , r) ;
      if (h < 0) or (r < 0) then
            writeln (′invalid input  values′)
      else
            begin
                  v := Pi/3 * sqr (r) * h ;
                  writeln (′volume = ′ , round (v))
            end
end.
```

8)

```
Program ArithOperations (input , output) ;
var   N1 , N2 : integer ;
      OP : char ;
begin
      writeln (′enter two integers and an operation symbol′) ;
```

```
        readln (N1 , N2 , OP) ;
        case OP of
                '+' : writeln (N1 : 5 , '+' : 2 , N2 :5 , '=' : 2, N1 + N2 :6) ;
                '-' : writeln (N1 : 5 , '-' : 2 , N2 :5 , '=' : 2 , N1 - N2 :6) ;
                '*' : writeln (N1 : 5 , '*' : 2 , N2 :5 , '=' : 2 , N1 * N2 :6) ;
                '/' : begin
                        if N2 = 0 then
                                writeln ('Division by zero')
                        else
                                writeln (N1:5, '/':2, N2:5, '=':2, N1/N2:8:3)
                      end
        end
end.
```

حل الاختبار الفصلي الثاني رقم (١)

1) a)

Sum	A	K
0	7	7
7	8	8
15	9	9
24	10	10
34		8
42		9
51		10
61		9
70		10
80		

المخرجات : 80

b)

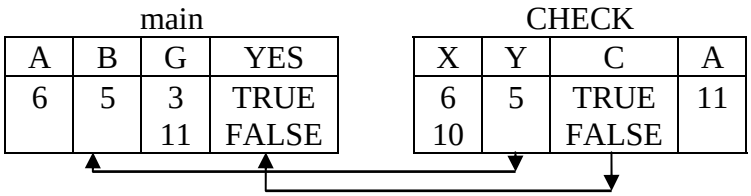
			+				1				2				3
			4				5				6				7
			5				6				7				8
			6				7				8				9

2)

```
program circle (input , output) ;
        var r , a , c : real ;
procedure circle_a_c (r : real ; var a , c : real) ;
        begin
                a := pi * r * r ;
```

```
            c := 2 * pi * r
        end
(* end circle_a_c *) ;
begin
        write (′Enter radius : ′) ;    readln (r) ;
        while r >= 0 do
            begin
                circle_a_c (r , a , c) ;
                writeln (′area = ′ , a) ;
                writeln (′circumference = ′ , c) ;
                write (′Enter radius : ′) ; readln (r)
            end
            (* end while *) ;
        end.
```

3)



المخرجات

A = 6
B = 5
G = 3
YES = TRUE
X = 10
A = 6
B = 5
G = 11
YES = FALSE

4)

```
program Zakat_Calc (input , output) ;
        const Alnisab = 500 ;
        var    Id_number : integer ;
               T , Z         :  real;
               Y_N           :  char ;
               i , M         :  integer ;
        function Zakat (Alnisab , TotalSavings : real ;
```

```
                    YearPssed : char) : real ;
        begin
            if YearPssed = 'Y' then
                if TotalSavings >= Alnisab then
                    Zakat := 0.025 * TotalSavings
                else Zakat := 0.0
                (* end if *)
            else Zakat := 0.0
            (* end if *)
        end
        (* end Zakat *) ;
    begin
    writeln ('enter number of people') ;
    readln (M) ;
    for i := 1 to M do
        begin
            write ('Id_Number :') ; readln (Id_number) ;
            write ('Total Savings : ') ; readln (T) ;
            write ('Year Passed : ') ; readln (Y_N) ;
            Z := Zakat (Alnisab , T , Y_N) ;
            writeln ('Id_Number : ', Id_number : 6,
                    'Total Savings : ' , T : 8 : 3 ,
                    'Zakat : ' , Z : 8 : 3)
        end
        (* end for *)
    end.
```

حل الاختبار الفصلي الثاني رقم (٢)

1)
```
    .
    else
        if W = 'N' then
           ZF := 0.1 * F
    else if W = 'A' then
           ZF := 0.05 * F
    else if W = 'C' then
           ZF := 0.075 * F ;
    writeln .....
end.
```

2) i) a **تعبير (ليس متغيرا بسيطا) يحل محل الوسيط المتغير** 2 * r

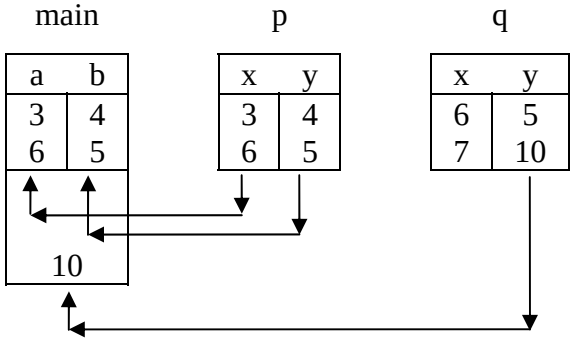
ii) ch **ثابت رمزي (ليس متغيرا بسيطا) يحل محل الوسيط المتغير** 'x'

iii) ch غير معرف في البرنامج الرئيسي

iv) لا يجوز استدعاء الدالة بمجرد ذكر اسمها في عبارة مستقلة

v) y لمتغير صحيح f(x , r , c) لا يجوز اسناد قيمة حقيقية

3)



المخرجات

p: x = 6 y = 5
main : a = 6 b = 5
q: x=7 y=10
main: a=6 b=10

4)

```
procedure Q4Pa (a , b : real ; var av , gm , max : real) ;
begin
        av := (a+b) / 2 ;
        gm := sqrt (a*b) ;
        if a > b then
                max := a
        else
                max := b ;
end ; {Q4Pa}

program Q4 (input , output) ;
var
        a , b , av , gm , max : real ;
{Insert the above procedure Q4Pa here}
begin
        write ('Enter two real numbers > ') ;
        readln (a , b) ;
        while (a >= 0.0) and (b >= 0.0) do
        begin
                Q4Pa (a , b , av, gm , max) ;
                writeln ('a = ', a , ' b = ', b , 'average = ', av ,
                        'geometric mean = ', gm , 'max of a and b = ', max);
                readln (a , b) ;
        end ; {while}
end. {Q4}
```

5)

a)

```
Function G( m1 , m2 , m3 : integer) : char ;
var
    avg : real ;
begin
    avg := (m1 + m2 + m3) / 3.0 ;
    case round (avg) of
        90..100 : G := 'A' ;
        80..89  : G := 'B' ;
        70..79  : G := 'C' ;
        60..69  : G := 'D' ;
          0..59 : G := 'F'
    end
end ;
```

b)

```
program Q5 (input , output) ;
var
    N , ID , m1 , m2 , m3 , I ,
    CountA , CountB , CountC , CountD , CountF : integer ;
    Grade : char ;
{Insert the above function G here}
begin
    CountA := 0 ;
    CountB := 0 ;
    CountC := 0 ;
    CountD := 0 ;
    CountF := 0 ;
    write ('Enter the number of students > ') ;
    readln (N) ;
    for I := 1 to N do
    begin
        write ('Enter student ID and three marks > ') ;
        readln (ID , m1 , m2 , m3);
        Grade := G (m1 , m2 , m3) ;
        writeln ('ID = ',ID, 'mark1 = ', m1 , 'mark2 = ', m2 ,
                        'mark3 = ', m3 , 'Grade = ', Grade) ;
        case Grade of
            'A' : CountA := CountA + 1 ;
            'B' : CountB := CountB + 1 ;
            'C' : CountC := CountC + 1 ;
            'D' : CountD := CountD + 1 ;
```

```
            'F' : CountF  := CountF + 1
        end ; {case}
    end ; {for}
    writeln ('The number of students receiving A is ', CountA) ;
    writeln ('The number of students receiving B is ', CountB) ;
    writeln ('The number of students receiving C is ', CountC) ;
    writeln ('The number of students receiving D is ', CountD) ;
    writeln ('The number of students receiving F is ', CountF) ;
end. {Q5}
```

حل الاختبار الفصلي الثاني رقم (٣)

1) a) In the Name of Allah
O Mojahidoon of Bosnia Assalamo A'laikom
May Allah bless you

b)

Sum	Product	Count
0	1	1
1	1	2
1	2	3
2	6	4
2	24	5

المخرجات

∇∇∇∇1∇∇∇∇1
1 2
2 6
2∇∇∇24

c)

∇∇∇∇1	∇∇∇∇2	∇∇∇∇3	∇∇∇∇4
2	4	3	4
3	6	9	12

2)

عبارات إسناد	مخرجات
A[1] := 1	A(∇1) = 1
A[4] := -1	A(∇2) = 3
A[2] := 3	A(∇3) = 5
A[5] := 2	A(∇4) = -1
A[3] := 5	A(∇5) = 2
A[6] := 7	A(∇6) = 7

3) a)

c	i	x
0	1	15
1	2	10
	3	-10
2	4	5

القيمة النهائية لـ x : 20

القيمة النهائية لـ c : 3

```
                5       20
        3       6
b)      c := 0 ;
        i := 1 ;
        Repeat
                Read (x) ;
                IF (x < 0) or (x > 10) THEN
                        c := c + 1 ;
                i := i + 1
        until i > 5 ;
```

4)

```
Program Series (output) ;
var  i , s : integer ;
begin
        s := 0 ;
        for i := 1 to 10 do
            s := s + sqr (4 * i + 1) ;
        writeln (′sum = ′, s)
end.
```

5)

```
Program Pilgrimage (input , output) ;
var
        ID , NL , NS , NA , N : integer ;
        PL , PA , PS : real ;
        CODE : char ;
begin
        NL := 0 , NS := 0 , NA := 0 ;
        writeln (′enter id , code′) ;
        readln (ID , CODE) ;
        while CODE <> ′E′ do
                begin
                        case CODE OF
                                ′L′ : NL := NL + 1 ;
                                ′S′ : NS := NS + 1 ;
                                ′A′ : begin
                                        writeln (′ID = ′ , ID) ;
                                        NA := NA + 1
                                      end
                        end ; {case}
                        readln (ID , code)
                end {while} ;
```

```
    N := NL + NS + NA ;
    writeln (′NL = ′, NL , ′NS = ′, NS , ′NA = ′, NA , ′N = ′, N) ;
    PL := NL/N * 100 ;
    PS := NS/N * 100 ;
    PA := NA/N * 100 ;
    writeln (′PL = ′, PL , ′PS = ′ , PS , ′PA = ′ , PA) ;
End.
```

حل الاختبار الفصلي الثاني رقم (٤)

1) a)
```
*  *  *  *  *
*  *  *  *
*  *  *
*  *
*
```

b)
```
Score = 77.20 Grade = C Points = 2
Score = 94.90 Grade = A Points = 4
Score = 66.10 Grade = D Points = 1
Score = 59.40 Grade = F Points = 0
Score = 89.45 Grade = B Points = 3
```

2) a)
```
i =  1 x =  -1 Sum = -1 Product =   1
i =  3 x =   9 Sum =  8 Product =   3
i =  5 x = 15 Sum = 23 Product =   3
i =  7 x = 22 Sum = 45 Product =  21
i =  9 x = 10 Sum = 55 Product =  21
i = 11 x = 30 Sum = 85 Product = 231
```

b)
```
i := 1 ; Sum := 0 ; Product := 1 ;
REPEAT
      Read (x) ;
      IF (x < 10) or (x > 20) THEN
            Product := Product * i ;
      Sum := Sum + x ;
      WriteLn(′i = ′, i:5 ,′x=′, x:5, ′Sum=′, Sum:5,
                        ′Product=′, Product:5) ;
      i := i + 2
UNTIL i > 11 ;
```

3)
```
PROGRAM Q3 (input , output) ;
VAR
      x , Sum , Term : Real ;
      i , n : Integer ;
BEGIN
      Write (′Enter x , n > ′) ;
      ReadLn (x , n) ;
      Sum := 0.0 ;
```

```
        FOR i := 1 to n DO
            BEGIN
                Term := (2 * i - 1) * x ;
                Sum := Sum + Sqr(Sin(Term)/Cos(Term))/Term
            END ;
        WriteLn ('The sum of the series is ' , Sum : 5 : 2)
    END.
```

4)

```
    PROGRAM Q4 (Input , Output) ;
                                                    CONST
                                                Sentinel = 0 ;
                                                    VAR
                                            Na , Nf , ID : Integer ;
                                          S1 , S2 , S3 , Ave : Real ;
                                                    BEGIN
                                    Write ('Enter ID , S1 , S2 , S3 > ') ;
                                            Read (ID , S1 , S2 , S3) ;
                                                Na := 0 ; Nf := 0 ;
                                          WHILE ID <> Sentinel DO
                                                    BEGIN
                                        Ave := (S1 + S2 + S3) / 3 ;
                    WriteLn ('ID = ', ID, 'Average = ', Ave:5:2) ;
                                                IF Ave > 90 Then
                                            Na := Na + 1
                                                        ELSE
                                        IF Ave < 60 Then
                                        Nf := Nf + 1
                Write ('Enter ID,S1,S2,S3(when done ID = 0)>');
                        Read (ID , S1 , S2 , S3)
                    END
            Writeln ('Na =', Na :5, 'Nf = ', Nf :5)
    END.
```

5) (a)

```
PROCEDURE Rectangle ;
VAR   A , B , P , Area : Real ;
BEGIN
    Read (A , B) ;
    Area := A * B ;
    P    := 2 * (A + B) ;
    WriteLn('Perimeter of the Rectangle = ', P : 7 : 3 ,
                                        'Its Area = ', Area:7:3)
```

```
END ;
(b)
PROCEDURE Square ;
VAR   D , P , Area : Real ;
BEGIN
      Read (D) ;
      Area := D * D ;
      P     := 4 * D ;
      WriteLn('Perimeter of the Square = ', P : 7 : 3 ,
                                          'Its Area = ' , Area:7:3)
END ;
(c)
PROCEDURE Circle ;
CONST
      Pi = 3.14159 ; {In Turbo Pascal Pi is a standard constant}
VAR   R , C , Area : Real ;
BEGIN
      Read (R) ;
      Area := Pi * R * R ;
      C     := 2 * Pi * R ;
      WriteLn ('Circumference of the circle = ', C:7:3,
                                          'Its Area=', Area:7:3)
END ;
(d)
PROGRAM Q4 (Input , Output) ;
VAR   Code : Integer ;
Procedures .....
BEGIN
      Write('Enter Code 1 for Rectangle,2 for Square,3 for Circle >');
      ReadLn (Code) ;
      IF (Code < 1) or (Code > 3) Then
            WriteLn ('Invalid CODE')
      ELSE
            CASE Code of
                  1 : Rectangle ;
                  2 : Square ;
                  3 : Circle
            END
END.
```

حل الاختبار الفصلي الثاني رقم (٥)

1) Q : Who is the most deserving of my good company ?
A : Your mother.
Q : Who is next ?
A : Your mother.
Q : Who is next ?
A : Your mother.
Q : Who is next ?
A : Your father.

2)

Tracetable: **جدول تتبع القيم**

k	i	j
5	1..0	1..4
4	1..1	1..3
3	1..2	1..3
2	1..3	1..1
1	1..4	1..0

Output : **المخرجات**

```
*  □  □  □  □  □  □  □  □  *
□  *  □  □  □  □  □  □  *
□  □  *  □  □  □  □  *
□  □  □  *  □  □  *
□  □  □  □  *  *
```

3)

a) F(c, i, a, j, b) **لا**

b) F(not c, i+j, i, j, b) **نعم**

c) F(c, a+j, a, j, b) **لا**

d) F(3<2, 2, i, j, a) **نعم**

e) F(c and (3>1), 2, 2, 3, a) **لا**

f) F(true, 3, i, a, b) **نعم**

4)

a) **لا**

المخرجات

□□1□□2□□3□□4□□5□□6□□7□□8□□9

(i)

b) **لا**

المخرجات

```
□□1
```

(ii)

المخرجات

```
***
```

(i)

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

(ii)

5)

```
program mark (input , output) ;
```

```
        var n , m , i , j , s , sm : integer ;
begin
        writeln (′Enter the number n of students′) ;
        writeln (′and the number m of marks per student : ′) ;
        readln (n , m) ;
        for i := 1 to n do
                begin
                        s := 0 ;
                        writeln (′Enter the student marks : ′) ;
                        for j := 1 to m do
                                begin
                                        readln (sm) ;
                                        s := s + sm
                                end
                        (* end for *) ;
                        writeln (′The average for this student is : ′, s/m :6:2)
                end
        (* end for *)
end.
```

6)

```
program mark (input , output) ;
        var  ma , mi , numb , n , s : integer ;
begin
        writeln (′Enter numbers : ′) ;
        s := 0 ;
        numb := 0 ;
        readln (n) ;
        mi := n ;
        ma := n ;
        repeat
                if n < mi
                        then mi := n
                (* end if *) ;
                if n > ma
                        then ma := n
                (* end if *) ;
                s := s + n ;
                numb := numb + 1 ;
                readln (n)
        until n < 0 ;
```

```
        writeln (′Result: max=′, ma, ′ min=′, mi, ′
average=′,s/numb:3:2)
end.
7)
program number_check (output) ;
        var i , j : integer ;
        begin
                j := 0 ;
                for i := 1 to 1000 do
                        if (i mod 3 = 0) and (i mod 7 = 0) then
                                j := j + 1
                        (* end if *) ;
                (* end for *)
                writeln (′Count : ′, j) ;
                readln
        end.
```

حل الاختبار النهائي رقم (١)

1) a) integer , 9.
   b) integer , 23.
   c) boolean , true.
   d) real , 11.0.
   e) char , ′B′.

2)
```
        program Students (input , output) ;
        var
                i , M , IdNo : integer ;
                Test1 , Test2 , Final , Score : real ;
                Grade : char ;
        Begin
                writeln (′enter number of students′) ;
                readln (M) ;
                for i := 1 to M do
                        begin
                                writeln (′enter information of a student :
                                        IdNo, Test1, Test2, Final′) ;
                                readln (IdNo , Test1, Test2, Final) ;
                                Score := Test1 + Test2 + Final ;
                                Case Score of
                                        90..100 :  Grade := ′A′ ;
                                        80..89   :  Grade := ′B′ ;
```

```
                    70..79   :  Grade := 'C' ;
                    60..69   :  Grade := 'D' ;
                     0..59   :  Grade := 'F' ;
                end ;
                writeln ('IdNo = ', IdNo, 'Grade = ', Grade)
            end
    End.
```

3)

```
    Program Numbers ;
    var num , od , ev11 : integer ;
    Begin
        od := 0 ;  ev11 := 0 ;
        writeln ('enter a number') ;
        readln (num) ;
        while num >= 0 do
            begin
                if num mod2 = 1 then
                    od := od + 1
                else if num mod11 = 0 then
                    ev11 := ev11 + 1 ;
                writeln ('enter a number') ;
                readln (num)
            end ;
        writeln ('number of odd numbers = ', od) ;
        writeln ('number of even numbers divisible by 11 is:'
ev11)
    End.
```

4) cno = 2

المخرجات

```
wrong c
right b
right c
wrong c
right d
```

```
c1 = c
cno = 3
c1 = b
c2 = b
cno = 2
c1 = c
c2 = c
cno = 5
cno = 1
c1 = d
c2 = d
```

5)

Main

w	x	y	z
40	5	10	20
5		35	

P1

a	x	y
5	10	
10	35	25

المخرجات

P1 : y=25 w=5 a=10
w=5 x=5 y=35 z=20

6)
```
Program Av ;
const size = 50 ;
type ArrayType = array [1..size] of integer ;
var
        L : ArrayType ;
        i : integer ;
        mean : real ;
function Average (A : ArrayType) : real ;
var sum : integer ;
begin
        sum := 0 ;
        for i := 1 to 50 do
                sum := sum + A[i] ;
        Average := sum / 50
end ;
Begin {main}
        for i := 1 to 50 do
                readln (L[i]) ;
        mean := Average (L) ;
        for i := 1 to 50 do
                if (L[i]> mean) then
                        writeln (L[i])
End.
```

7)
```
X[1] = 2
X[2] = 3
X[3] = 4
X[4] = 0
X[5] = 0
X[6] = 0
Y[1] = 0
Y[2] = 0
Y[3] = 0
```

المخرجات

```
∇∇2∇∇3∇∇4∇∇0∇∇0∇∇0
   J = 2   B[J] = 4
   J = 3   B[J] = 6
   J = 4   B[J] = 8
∇∇0∇∇4∇∇6∇∇8∇∇0∇∇0
```

Y[4] = 0
Y[5] = 0
Y[6] = 0

Double (X , Y , 1 , 2 , 3) ;
SA = 1 , SB = 2 , Len = 3
I = 1 J = 2 N = 1

B[2] = 2A[1] = 2 X [1] = 4
I = 1 + 1 = 2
J = 2 + 1 = 3
N = 1 + 1 = 2

B[3] = 2A[2] = 2 X [2] = 6
I = 2 + 1 = 3
J = 3 + 1 = 4
N = 2 + 1 = 3

B[4] = 2A[3] = 2 X [3] = 8
I = 3 + 1 = 4
J = 4 + 1 = 5
N = 3 + 1 = 4
B $\rightarrow$ Y

حل الاختبار النهائي رقم (٢)

1)

I	J		المخرجات
1	3	K[3] = 3	
	2	K[2] = 2	
	1	K[1] = 1	
		K[1] = 1 - 10 = -9	-9
2	3	K[3] = 6	
	2	K[2] = 4	
		K[2] = 4 - 10 = -6	-6
3	3	K[3] = 9	
		K[3] = 9 - 10 = -1	-1

2)

Main

a	b	c	d
10	20	30	40
	22		35

P

x	y	c
10	20	5
9	22	

P : a = 10 b = 20 c = 5 d = 35 x = 9 y = 20
main : a = 10 b = 22 c = 30 d = 35

3)
```
program Q3 (input , output) ;
var n , count : integer ;
begin
    count := 0 ;
    readln (n) ;
    while n <> 0 do
        begin
            if (n mod 2 = 0) and (n mod 3 = 0) then
                count := count + 1 ;
            readln (n)
        end ;
    writeln (′count = ′, count)
end.
```

4)
```
program Q4 (input , output) ;
var a , b , c : real ;
begin
    readln (a , b , c) ;
    while (a >= 0.0) and (b >= 0.0) and (c >= 0.0) do
        begin
            if (a+b > c) and (a+c > b) and (b+c > a) then
                begin
                    if (a=b) and (b=c) then
                        writeln (′Triangle with sides′,
                            a,b,c,′ is equilateral′)
                    else if (a=b) or (b=c) or (a=c) then
                        writeln (′Triangle with sides′,
                            a,b,c,′ is isosceles′)
                    else
                        writeln (′Triangle with sides′,
                            a,b,c,′ is scalene′)

                end
            else
                writeln (a,b,c, ′ do not form a triangle′) ;
            readln (a,b,c)
        end
end.
```

5)
```
program Q5 (input , output) ;
type
        vector1 = array [1..100] of char ;
        vector2 = array [1..100] of integer ;
var
        n : integer ;
        Depts : vector1 ;
        Marks : vector2 ;
procedure ReadAndStore (var n : integer ; var Depts : vector1 ;
                                        var Marks : vector2) ;
var     i : integer ;
begin
        readln (n) ;
        for i := 1 to n do
                readln (Depts [i] , Marks [i])
end ;
function HighestMark (n : integer ; Marks : vector2) : integer ;
var i , max : integer ;
begin
        max := Marks [1] ;
        for i := 2 to n do
                if Marks [i] > max then
                        max := Marks [i] ;
        HighestMark := max
end ;
function AvComp(n:integer; Depts:vector1; Marks:vector2):real;
var i , sum , ncs : integer ;
begin
        sum := 0 ; ncs := 0 ;
        for i := 1 to n do
                if (Depts [i] =′C′) or (Depts [i] = ′c′) then
                        begin
                                sum := sum + Marks [i] ;
                                ncs := ncs + 1
                        end ;
                AvComp := sum / ncs
        end ;
Begin
        ReadAndStore (n , Depts , Marks) ;
        writeln (′The highest mark is′, HighestMark (n,Marks)) ;
        writeln (′The avg. mark of cs students is′,
```

```
                                        AvComp (n,Depts, Marks))
        End.
```

6) a)

```
procedure Search (X : vector ; n , key : integer ; var index : integer) ;
var i : integer ; found : boolean ;
begin
        i := 1 ; found := false ;
        while (i <= n) and (not found) do
                if X[i] = key then
                        found := true
                else
                        i := i + 1 ;
        if found then
                index := i
        else
                index := 0
end ;
```

حل آخر :

```
begin
        i := 1 ;
        while (i <= n) and (X [i] <> key) do
                i := i + 1 ;
        if i <= n then
                index := i
        else
                index := 0
end ;
```

b)

```
        Program Q6 (input , output) ;
        type    vector = array [1..700] of integer ;
        var Badr : array [1..314] of integer ;
                Ohod : array [1..700] of integer ;
                i , index : integer ;
        procedure Search
                :
        Begin
                for i := 1 to 314 do
                        readln (Badr [i]) ;
                for i := 1 to 700 do
                        readln (Ohod [i]) ;
                for i := 1 to 314 do
                        begin
```

```
                Search (Ohod, 700, Badr [i], index) ;
                if index <> 0 then
                    writeln (Badr [i])
            end
End.
```

حل الاختبار النهائي رقم (٣)

1) a) 5 integer
   b) 5.5 real
   c) 13 integer
   d) true boolean
   e) 18 integer
   f) true boolean

2) a) -x*x* (sqrt (y) + 2) / y
   b) x * y / (x + y) - (x - y) / (x * y)
   c) sqrt (x * x + y * y)

3) a) x□ = □□□□3
      s□ = □□10.0000
   b) ***** ↱
      ↱
      ***↱
      ↱
      *

4)
```
case digit of
        0 , 1 , 2 : value := digit ;
        3 , 4     : value := 2 * digit ;
        5         : value := 3 * digit ;
        else        value := 4 * digit ;
end
(* end case *) ;
```

حل آخر :

```
if 0 ≤ digit ≤ 5 then
  case digit of
    0 , 1 , 2 : value := digit ;
    3 , 4     : value := 2 * digit ;
    5         : value := 3 * digit ;
                                end
```

```
                                        (* end case *) ;
                                    else value := 4 * digit
                                            (* end if *) ;
```

5)
```
program cylinder (input , output) ;
const Pi = 3.14159 ;
var h , r : real ;
begin
    write (′height : ′) ; readln (h) ;
    write (′radius : ′) ; readln (r) ;
    writeln (′volume = ′, pi * r * r * h) ;
    writeln (′surface = ′, 2 * pi * r * h)
end.
```

6)
```
program product (input , output) ;
var n , p : integer ;
begin
    n := 1 ;
    p := 1 ;
    while p < 10000 do
        begin
            n := n + 1 ;
            p := p * n
        end
    (* end while *) ;
    writeln (′smallest value of n is ′, n)
end ;
```

7)
```
function multiple (x , y : integer ) : boolean ;
    begin
        if (x mod y = 0) or (y mod x = 0) then
            multiple := true
        else multiple := false
        (* end if *) ;
    end
(* end multiple *) ;
```

8) a)
```
function is_positive_array (var x : array_type; m : integer) : boolean ;
var i : integer ;
    positive : boolean ;
begin
    positive := true ;
    for i := 1 to m do
        if x[i] < 0 then
```

```
                positive := false
            (* end if *)
        (* end for *) ;
        is_positive_array := positive
end
(* end is_positive_array *) ;
```

حل آخر :

```
begin
        positive := true ;
            i := 1 ;
            while (i <= m) and positive do
                begin
                    if x[i] < 0 then
                        positive := false
                    (* end if *)
                    i := i + 1
                end
            (* end while *).
            is_positive_array := positive;
        b)
program array (input , output) ;
        const max = 100 ;
        type array_type = array [1..max] of real ;
        var    i , m : integer ;
            x : array_type ;
        function is_positive_array (var x : array_type ;
                                    m : integer) : boolean ;
                    :
        (* end is_positive_array *) ;
        begin
            write (′number of array - components : ′) ; readln (m) ;
            for i := 1 to m do
                readln (x[i])
            (* end for *) ;
        if is_positive_array (x , m) then
            writeln (′array is positive′)
        else writeln (′array is not positive′)
        (* end if *)
end.
```

حل الاختبار النهائي رقم (٤)

1) a) i) 48 ii) 0 iii) false

b)
```
read(ch) ;  count := 0 ;
repeat
        write (ch) ;
        count := count + 1 ;
        read (ch)
until (ch = '*') or (count > 8) ;
```

2) a) ii) real

b) ii) real

c) ii) real

3) a) **المخرجات**

```
4
4
4
```

b)

z	g	s	i	t
0	0	0	1	-3
1	1	-3	2	5
2	2	2	3	4
	3	6	4	0
	4	2	5	-4
		6	6	4
			7	

z=2, g=4, s=6, i=7, t=4.

c)

count	stop	k	**المخرجات**
0	4	$1 \rightarrow 0$	**سطر فارغ**
1		$1 \rightarrow 1$	∇∇1
2		$1 \rightarrow 2$	∇∇1∇∇2
3		$1 \rightarrow 3$	∇∇1∇∇2∇∇3
4			All Done

4) a) i)

k	i	x
1	2	5
3 ←	3	2
4 ←	4	6
	5	8
		3

Result = 4

ii) **وظيفة الدالة : إيجاد موضع (أي ترتيب) أكبر عنصر**

b)

المخرجات

5: ∇∇∇∇4∇∇∇∇5
1: 2 6
6: 4 6
1: 2 5
3: 6 12
2: 10 11
4: 6 12
7: 6 6

Main

a	b
4	5
	6
6	

One

x	y
4	5
2	6
6	4
2	5

Two

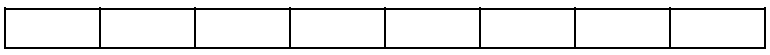
a	b
4	6
5	
6	12

Three

a	b
6	12
10	11

5)
```
type
        vector = array [1..max] of integer ;
function second (D : vector ; N : integer) : integer ;
var
        i , j : integer ; temp : integer ;
begin
        for i := 1 to N-1 do
                for j := i+1 to N do
                        if D[i] < D[j] then
                                begin
                                        temp := D[i] ;
                                        D[i] := D[j] ;
                                        D[j] := temp
                                end ;
                second := D[2]
end ;
```

6)
```
type
        vector = array [1..max] of char ;
function Palindrome (A : vector ; N : integer) : boolean ;
var
        i : integer ;
begin
        Palindrome := true ;
        for i := 1 to N div 2 do
                if A[i] <> A[n - i + 1] then
                        Palindrome := false
end ;
```



A_1 A_2 A_{n-1} A_n

$A_1 = A_n$

$A_2 = A_{n-1}$

$\vdots$

$A_i = A_{n-i+1}$

7) a)
```
function checkNo (N : integer) : boolean ;
begin
        if N mod 2 = 0 then
                checkNo := true
        else
                checkNo := false
end ;
```

b)
```
procedure factors (n : integer) ;
var
        i : integer ;
begin
        for i := 1 to n do
                if n mod i = 0 then
                        writeln (i)
end ;
```

c)
```
program Numbers (input , output) ;
var
        N : integer ;
function checkNo ...
:
:
procedure factors ...
:
:
Begin
        writeln (′enter N′) ;
        readln (N) ;
        if CheckNo (N) then
                begin
                        writeln (′factors of ′, N : 4, ′are : ′) ;
                        factors (N)
                end
        else
                writeln (N : 4, ′is an odd number ′)
End.
```

حل الاختبار النهائي رقم (٥)

1) النوع القيمة

a) Integer 36
b) Boolean True
c) Boolean False
d) integer 12
e) character B

2)
```
Procedure PintPrayer (InPrayer : PRAYERS) ;
begin
    case InPrayer of
        FAJR : write (′FAJR′) ;
        ZUHR : write (′ZUHR′) ;
        ASR : write (′ASR′) ;
        MAGHREB : write (′MAGHREB′) ;
        ISHA : write (′ISHA′)
    end
end ;
```

3)
```
----2----3----4----5----6----7----8----9
----4----5
----4----5----6----7
----4----5----6----7----8-----9
----4----5----6----7----8-----9----8----9
```

4)
```
program Q4 ;
var
    X : real ;
    Min , Sum , Ave : real ;
    N : integer ;
begin
    read (X) ;
    N := 0 ; Sum := 0 ; Min := X ;
    while X >= 0 do
    begin
        Sum := Sum + X ;
        N := N + 1 ;
        if X < Min then Min := X ;
        read (X)
    end ;
    if N > 0 then
      begin
        Ave := Sum / N ;
        Writeln (′Minimum = ′, Min : 10 : 3) ;
        Writeln (′Average = ′, Ave : 10 : 3)
      end.
```

5)

	K	I	J
BBBBBBBBB	5	1..0	1..9
*BBBBBBB	4	1..1	1..7
**BBBBB	3	1..2	1..5
***BBB	2	1..3	1..3
****B	1	1..4	1..1
		*	B

6)

Main

a	b	c	d
4	3	10	20
12			16

P1

a	b	c
3	4	8
11	12	

P2

x	y
3	2

ملاحظة : بعد الانتهاء من تنفيذ الاستدعاء

P1(b , a) ≡ P1 (3 , 4)

غيَّرنا قيمة (الوسيط الفعلي) a في البرنامج الرئيسي من 4 إلى 12 لأن الوسيط الشكلي المقابل (b) في البرنامج الفرعي P1 – وهو وسيط متغير – قد تغيرت قيمته من 4 إلى 12.

P2 :	a = 3	b = 4	c = 8	d = 20
P1 :	a = 11	b = 12	c = 8	d = 16
Main :	a = 12	b = 3	c = 10	d = 16

7)

```
program Q7 (input , output) ;
const
MaxSize = 100 ;
type
AT = array [1..MaxSize] of integer ;
var
A , EVARRAY , ODARRAY : AT ;
i , N : 1 .. MaxSize ;
IEVEN , IODD : 0 .. MaxSize ;
begin
Write (′Enter N′) ; Readln (N) ;
for i := 1 to N do read (A[i]) ;
IEVEN := 0 ; IODD := 0 ;
for i := 1 to N do
begin
if A[i] and 2 = 0 then
begin
IEVEN:=IEVEN+1; EVARRAY[IEVEN]:=A[i]
```

```
                                        end
                                        else
                                    begin
IODD:=IODD+1; ODARRAY[IODD]:=A[i]
                                    end
                                end ;
                            if IEVEN > 0 then
                                begin
                        write (′Even array is : ′) ;
for i := 1 to IEVEN do write (EVARRAY [i] :3)
                                end
                                else
                    Writeln (′No even-valued elements′) ;
                            if IODD > 0 then
                                begin
                        write (′Odd array is : ′) ;
for i := 1 to IODD do write (ODARRAY [i] : 3)
                                end
                                else
                    Writeln (′No odd-valued elements ′)
                                                end.
```

حل الاختبار النهائي رقم (٦)

```
                    Program Employee (input , output) ;      1)
                                                var
                                id , nchild : integer ;
                                BS , TS , SA : real ;
                                    code : char ;
                                                begin
                        writeln (′enter id , BS , code , nchild′) ;
                            readln (id , BS , code , nchild) ;
                        if (code = ′S′) or (code = ′s′) then
                                        SA := 100
                    else if (code = ′M′) or (code = ′m′) then
                        SA := 100 + 50 + 15.5 * nchild
                            else writeln (′an error code′) ;
                                    TS := BS + SA ;
                        writeln (′id = ′, id , ′Basic Salary =′, BS ,
                    ′Total Salary =′, TS)
                                                end.
```

He whose migration was for Allah and His Messenger, 2) a)

his migration was for Allah and his Messenger,
Agreed upon.

The values of I and J are ---1---1 b)
The values of I and J are ---2---2
The values of I and J are ---3---3

3)

Main

X	Y	Z	G
3 6	5	7	2 4 0

Outer

X var	Y var	Z
5	3 6	4 6

Inner

X var
4 6

المخرجات

G=2 X=3 Y=5 and Z=7
In Inner G=0 X=6 Y=6 and Z=4
In Outer G=0 X=5 Y=6 and Z=6
G=0 X=6 Y=5 and Z=7

4) **المخرجات**

-1
2
1
4
3
6
5
8
-1
4
1
6
3
8
5
2

```
Program Minimum (input , output) ;      5)
                                    Type
                vector = array [1..100] of integer ;
                                     var
                        i , j , L , U : integer ;
                                A : vector ;
                             small : integer ;
function index (A : vector ; L , U : integer) : integer ;
                                     var
                            i , small : integer ;
                                   begin
                                small := A[L] ;
                                   index := L ;
                            for i := L+1 to U do
                        if small  > A[i] then
                                  begin
                    small := A[i]
                        index := i
                                    end
                                  end ;
            Begin writeln(′enter elements of array A′) ;
                                for i := 1 to 100 do
                               readln (A[i]) ;
writeln (′enter lower and upper indices of range′) ;
                                readln (L , U) ;
                            j := index (A , L , U) ;
                                small := A[j] ;
writeln (′Required range : L = ′, L , ′U = ′, U) ;
            writeln (′smallest element is : ′, small) ;
                             writeln (′Its index is : ′, j)
                                    End.
Program Sequence (input , output) ;     6)
                                    type
                vector = array [1..100] of real ;
                                     var
                        i , j , n , m : integer ;
                                A : vector ;
                        sum , average : real ;
                                   begin
                                    i := 1 ;
```

```
                                    readln (A[i]) ;
                    while (i <= 100) and (A[i] >= 0) do
                                        begin
                                    i := i + 1 ;
                                readln (A[i])
                                        end ;
        n := i-1 ; {n is the number of elts in the array}
                                    sum := 0 ;
                                for i := 1 to n do
                            sum := sum + A[i] ;
                        average := sum / n ;
    m := 0 ; {m is the number of elts in the good interval}
                                for i := 1 to n do
                            if A[i] > 70.5 then
                                begin
writeln ('we changed elt. of index ', i) ;
                    if average <= 70.5 then
                      A[i] := average
                                else
                          A[i] := 0
                                    end
                                        else
                                m := m + 1 ;
                    writeln ('no of read elements = ', n) ;
            writeln ('no of elts in the good interval = ', m) ;
                    writeln ('final elts of the array A are : ') ;
                        for i := 1 to n do writeln (A[i])
                                                    end.
                        Program Numbers (input , output) ;      7)
                                                    var
                                x , sum : integer ;
                                                  begin
                                        sum := 0 ;
                                        readln (x) ;
                                    while x <> -1 do
                                        begin
                            sum := sum * 10 + x ;
                                    readln (x)
                                        end ;
        writeln ('The equivalent number is : ', sum)
                                                    end.
```

بعض المراجع عن البرمجة بلغة الباسكال
References on Programming in Pascal

فيما يلي قائمة بأسماء بعض الكتب عن البرمجة بلغة الباسكال ، والقائمة مرتبة ترتيبا أبجديا بالنسبة لأسماء المؤلفين.

١- الخوارزميات والبرمجة بلغة الباسكال ، د. أبو بكر أحمد السيد . دار القلم ، الكويت ، ١٩٩٧.

٢- البرمجة المتقدمة بلغة الباسكال ، د. حمزة سيد رشوان – د. أبو بكر أحمد السيد . دار القلم ، الكويت ، ١٩٩٤.

3- Ashry (El-Sayed), Z. H. ;
Practical Programming in Pascal ,
Sanad Publishers, Kuwait, 1997.

4- Cooper, D. and Clancy, M. ;
Oh ! Pascal !, second edition,
Norton W. W. & Company.

5- Crawley, J. W. and McArthur, W. G. ;
Structured Programming using Pascal,
Prentic-Hall, Inc., Englewood Cliffs,
New Jersey 07632.

6- Dale, N. ;
Programming in Pascal with an early introduction to Procedures,
D. C. Heath and Company, Lexington, Massachusetts,
Toronto, 1990.

7- Etter, D. M ;
Problem Solving in Pascal for Engineers and Scientists
The Benjamin / (Cummings Publishing Company, Inc.

8- Holmes, B. J. ;
Pascal Programming,
DP.Publications Limited, Grand Union Ind. Est.-Unit 6

Abbey Road, London NW10 7UL, 1988.

9- Koffman, E. B.
Pascal Problem Solving and Program Design,
fifth edition,
Addison-Wesley Publishing company, Inc., 1995.

10- Leestma, S. and Nyhoff, L. ;
Pascal Programming and Problem Solving,
fourth editon,
Macmillan Publishing Company, 1993.

11- Meyers, R. A. ;
Pascal, The Software Fundamentals of Computer Science,
Prentice Hall, Inc., Englewood Cliffs,
NJ 07632 , 1992.

12- Nance, D. W. ;
Understanding Turbo Pascal : Programming and Problem Solving,
West Publishing Company, 1994.

13- Niedelman, M. S. and Carnine, D. W. ;
Learning Pascal,
Scott, Foresman and Company, Glenview,
Illinois, Boston, London.

14- O'Brien, S. and Nameroff, S. ;
Turbo Pascal 7 : The Complete Reference,
Obsorne McGraw-Hill, Inc., 2600 Tenth Street,
Berkeley, California 94710 , 1993.

15- Palmer, S. D. ;
Mastering Turbo Pascal,
Sybex, Inc., 2021 Challenger Drive, Alameda, CA 94501,
1991

الكلمات المحجوزة
Reserved Words

فيما يلي قائمة بالكلمات المحجوزة المعرفة سابقا للحاسب ولا يجوز إعادة تعريفها وهي مرتبة ترتيبا أبجديا.

AND	FORWARD	PROCEDURE
ARRAY	FUNCTION	PROGRAM
BEGIN	GOTO	RECORD
CASE	IF	REPEAT
CONST	IN	SET
DIV	LABEL	THEN
DO	MOD	TO
DOWNTO	NIL	TYPE
ELSE	NOT	UNTIL
END	OF	VAR
FILE	OR	WHILE
FOR	PACKED	WITH

ويضاف إلى هذه القائمة كلمة STRING **في تيربو باسكال.**

دليل المصطلحات العربية والإنجليزية
Index

فيما يلي قائمة بالمصطلحات الإنجليزية مرتبة ترتيبا أبجديا ، والمصطلحات العربية المقابلة لها ، وكذلك فهرس لهذه المصطلحات ، مع ملاحظة أن الكلمات التي تستخدم في قاموس لغة الباسكال قد كتبت كل حروفها حروفا كبيرة ..

محتويات الكتاب

كتب للمؤلف

من منشورات دار القلم .. الكويت

١-برمجة الحاسب بلغة الفورتران ، ط ٤ ، ١٩٩٢.

٢-مقدمة في نظرية المعلومات ، ط ٢ ، ١٩٩٣.

٣-الشبكات الرقمية ، ١٩٨٦.

٤-التحليل العددي ، ١٩٨٨.

٥-الجبر الخطي ، ط ٢ ، ١٩٩٥.

٦-برمجة الحاسب بلغة الباسكال ، ١٩٨٩.

٧-البرمجة المتقدمة بلغة الباسكال ، مع د. حمزة رشوان ، ١٩٩٤.

٨-الدوائر المتكاملة الرقمية (ترجمة) ، ١٩٩٣.

٩-الخوارزميات والبرمجة بلغة الباسكال ، ١٩٩٧.

﴿ سُبحَنكَ لاَ عِلْمَ لنا إلاّ مَا عَلَّمتَنَا إنّكَ أَنت العليمُ الحَكِيمُ ﴾

(سورة البقرة : ٣٢)

Algorithms and Programming in

PASCAL

Dr. Abu-Bakr Ahmad El-Sayed

Dept. of Mathematics and Computer Science
University of Kuwait

661