مبادئ
لغات البرمجة

١-١٣- عبارتي الإدخال والإخراج Read, Write. ٢٨
١-١٣-١- عبارة الإدخال Read. ٢٨
١-١٣-٢- عبارة الإخراج Write. ٢٩
١-١٣-٣- شكل آخر لعبارتي الإدخال والإخراج. ٢٩
١-١٣-٤- التحكم بشكل إخراج وطباعة المتحولات. ٣٠
١-١٤- عبارات USES، CRT، CLRSCR. ٣١
١-١٥- بيئة التطوير المتكاملة IDE في تربو باسكال Integrated Development Environment. ٣٥
١-١٥-١- فتح ملف. ٣٧
١-١٥-٢- تنفيذ البرنامج. ٣٩
١-١٥-٣- تعديل البرنامج. ٤٠
١-١٥-٤- نافذة التحرير. ٤١
١-١٦- التعمق في بيئة التطوير المتكاملة IDE. ٤١
١-١٦-١- التنقل ضمن الملف. ٤١
١-١٦-٢- إضافة وحذف نص ما. ٤٢
١-١٦-٣- معالجة كتل من النصوص. ٤٣
١-١٦-٤- وضع علامات مواضع. ٤٤
١-١٧- استخدام قوائم بيئة التطوير المتكاملة IDE. ٤٥
١-١٧-١- قائمة الملف. ٤٥
١-١٧-٢- قائمة التحرير. ٤٨
١-١٧-٣- قائمة البحث. ٥٠
١-١٧-٤- قائمة التنفيذ. ٥٣
١-١٧-٥- قائمة الترجمة. ٥٥

١-١٧-٦- قائمة التنقيح. ٥٧
١-١٧-٧- قائمة الأدوات. ٦٠
١-١٧-٨- قائمة الخيارات. ٦١
١-١٧-٩- قائمة النافذة. ٦٣
١-١٧-١٠- قائمة المساعدة ٦٦
١-١٨- الكلمات المحجوزة RESERVED WORDS ٧١
تمارين عامة ٧٣

الفصل الثاني: الخوارزمية ALGORITHM. ٧٥
٢-١٠- الخوارزمية ALGORITHM. ٧٥
٢-٢- خوارزمية حساب تابع رياضي. ٧٥
٢-٣- مفهوم الخوارزمية. ٨٠
٢-٤- الطرائق البيانية. ٨٤
٢-٥- البرمجة المهيكلة. ٨٥

الفصل الثالث: مبادئ أساسية في البرمجة بلغة تربو باسكال. ٨٧
٣-١- التعليمة الشرطية. ٨٧
٣-١-١- الشرط. ٨٧
٣-١-٢- الاختيار. ٨٧
٣-٢- تعليمة التكرار. ٨٩
٣-٣- أمثلة عامة. ٩٤
تمارين عامة. ١١١

الفصل الرابع: العمليات والتوابع الحسابية المعرفة في تربو باسكال. ١١٣
٤-١- العمليات الحسابية. ١١٣
٤-٢- الأولويات بين العمليات الحسابية. ١١٤
٤-٣- التوابع الحسابيـــة المعرفة على الأعـــداد الحقيقية في تربو باسكال. ١١٤
٤-٤- العمليـات الحســابيـــة المعرفـــة على الأعـــداد الصحيحة في باسكال. ١١٩
٤-٤-١- أمثلة. ١١٩
٤-٤-٢- ملاحظات. ١٢٠
٤-٥- التوابع المعرفة على الأعداد الصحيحة في تربو باسكال. ١٢٢
٤-٥-١- تابع القطع Trunc. ١٢٢
٤-٥-٢- تابع التدوير Round. ١٢٢
٤-٥-٣- توابع الترتيب الســلمي Succ، Pred. ١٢٣
٤-٦- مثال تعليمي. ١٢٤
تمارين عامة. ١٢٨

الفصل الخامس: نمط المتحولات المنطقية Boolean. ١٣١
٥-١- مقدمة. ١٣١
٥-٢- التصريح عن متحول من نمط منطقي Boolean. ١٣١
٥-٣- عمليات القراءة والكتابة والإسناد. ١٣٢
٥-٤- العمليات للمعرفة على النمط المنطقي. ١٣٢
٥-٤-١- العملية AND. ١٣٣
٥-٤-٢- العملية OR. ١٣٤

٥-٤-٣- العملية XOR. ١٣٥
٥-٤-٤- العملية NOT. ١٣٦
٥-٤-٥- أسبقية العمليات البوليانية. ١٣٧
٥-٥- قوانين الجبر البولي. ١٣٨
٥-٦- أمثلة عامة. ١٣٩
تمارين عامة. ١٤٩

الفصل السادس: نمط المتحول المحرفي CHAR. ١٥١
٦-١- مقدمة. ١٥١
٦-٢- شفرة أسكي ASCIT. ١٥١
٦-٣- رموز أسكي ASCIT . ١٥٢
٦-٤- التصريح عن متحول من نمط محرفي. ١٥٢
٦-٥- عملية الإسناد. ١٥٣
٦-٦- عمليات المقارنة. ١٥٣
٦-٧- جدول توصيف أسكي. ١٥٤
٦-٨- التوابع المعرفة على النمط المحرفي. ١٥٥
٦-٨-١- تابع الترتيب أو رقم الترميز ORD. ١٥٥
٦-٨-٢- تابع التحويل من رقم لمحرف CHR. ١٥٥
٦-٩- أمثلة. ١٥٦
تمارين عامة. ١٦٣

الفصل السابع: أشكال أخرى لبنى التكرار في لغة تربو باسكال. ١٦٥
٧-١- التعليمة التكرارية Repeat. ١٦٥

٧-٢- خواص التعليمة التكرارية Repeat. ١٦٥
٧-٣- أمثلة عامة. ١٦٦
٧-٤- التعليمة التكرارية For. ١٧٠
٧-٥- خواص التعليمة التكرارية For. ١٧١
٧-٦- أمثلة عامة. ١٧٢
تمارين عامة. ١٩١
الفصل الثامن: بنية التحكم الشرطية CASE والنمط المعدود. ١٩٣
٨-١- مقدمة. ١٩٣
٨-٢- التعليمة الشرطية Case. ١٩٣
٨-٣- خواص التعليمة الشرطية. ١٩٤
٨-٤- أمثلة عامة. ١٩٥
٨-٥- النمط المعدود لـ Enumerated. ١٩٩
٨-٦- العمليات والتوابع المعرفة على النمط المعدود. ٢٠١
أمثلة عامة. ٢٠١
تمارين عامة ٢٠٥
الفصل التاسع: نمط المعطيات مصفوفة Array. ٢٠٧
٩-١- مقدمة. ٢٠٧
٩-٢- المصفوفات. ٢٠٨
٩-٣- التصريح عن متحول من نمط مصفوفة أحادية البعد. ٢٠٨
٩-٤- ملاحظات على المصفوفات. ٢٠٩
٩-٥-أمثلة عامة. ٢١٠
٩-٦- التصريح عن مصفوفة ثنائية البعد. ٢١٤

٩-٧- أمثلة عامة. ٢١٥
٩-٨- تعريف الثوابت. ٢٢١
٩-٩- المصفوفة ذات الحجوم المختلفة. ٢٢٣
٩-١٠- ضرب مصفوفة بشعاع. ٢٢٥
٩-١١- خوارزمية الترتيب الفقاعي (خوارزمية الفرز بالتعويم) ٢٣٢
٩-١٢- نمط المعطيات سلسلة رمزية. ٢٣٦
٩-١٣- مصفوفات السلاسل الرمزية أحادية البعد في لغة تربو باسكال. ٢٣٨
تمارين عامة. ٢٤٤
الفصل العاشر: البرامج الجزئية Subprograms. ٢٥١
١٠-١- مقدمة. ٢٥١
١٠-٢- البرنامج الجزئي التابع Function. ٢٥٢
١٠-٢-١- تعريف التابع في لغة تربو باسكال. ٢٥٢
١٠-٢-٢- ملاحظات عامة حول تعريف التوابع. ٢٥٤
١٠-٣- البرنامج الجزئي الإجرائية Procedure. ٢٥٩
١٠-٣-١- تعريف الإجرائية Procedure. ٢٥٩
١٠-٣-٢- ملاحظات حول تعريف الإجرائية. ٢٦٠
١٠-٤- أمثلة. ١٦١
١٠-٥- الفارق بين الإجرائية والتابع. ٢٦٨
١٠-٦- البرامج الجزئية العودية. ٢٦٨
١٠-٧- مسألة أبراج هانوي. ٢٧١
١٠-٨- الإجرائيات والتوابع مسبقة التعريف. ٢٨٠

١٠-٨-١- التوابع والإجرائيات مسبقة التعريف في لغة باسكال القياسية. ٢٨١
١٠-٨-٢- الإجرائيات والتوابع مسبقة التعريف في لغة تربو باسكال. ٣٠٥
تمارين عامة ٣٢٦

الفصل الحادي عشر: نمط المعطيات مجموعة Set. ٣٣٥
١١-١- مقدمة. ٣٣٥
١١-٢- التصريح عن مجموعة Set. ٣٣٥
١١-٣- العمليات على المجموعات. ٣٣٦
١١-٣-١- الإسناد والتعبير عن الثوابت. ٣٣٦
١١-٣-٢- العمليات الجبرية على المجموعات. ٣٣٧
١١-٤- تطبيقات على المجموعات. ٣٣٧
١١-٤-١- اختبار الانتماء لمجموعة. ٣٣٧
١١-٤-٢- اختبار تقاطع مجموعتين. ٣٣٨
١١-٤-٣- الحصول على الأعداد الأولية. ٣٤٢
تمارين عامة محلولة ٣٥٥
دليل المصطلحات العلمية العربية. ٤٤١
دليل المصطلحات العلمية الإنكليزية. ٤٥٣
رسائل الخطأ. ٤٦٥
المخططات الكتابية. ٥٠٩
المراجع باللغة العربية. ٥٣١
المراجع باللغة الإنكليزية. ٥٣٣

مقدمة

خصص هذا الكتاب مبادئ لغات البرمجة لدراسة أساسيات ومفاهيم لغة التربو باسكال من أجل حل الكثير من المسائل الرياضية أوامر هذه اللغة، خصص هذا الكتاب لطلاب السنة الأولى في كلية العلوم قسم الرياضيات.

ابتدئ الكتاب بدراسة المفاهيم الأساسية للغة التربو باسكال، حيث تعتبر لغة التربو باسكال من اللغات التي تجمع بين منهجية البرمجة والاقتصاد في الجهد والوقت وأثبتت مدى صلاحيتها وملاءمتها للهدف.

تعتبر لغة تربو باسكال هي البداية العلمية الصحيحة لكل مبرمج أو طالب متخصص بالمعلوماتية. جاءت شركة بورلاند لتصدر بيئة البرمجة Delphi، والتي تعتمد على لغة تربو باسكال كخلفية لغوية لها. مما جعل لغة تربو باسكال لغة أساسية لدراسة غيرها من اللغات، ومن أراد أن يتعلم أساسيات البرمجة عليه أن يدرس لغة البرمجة التربو باسكال باحترافية.

قسمنا الكتاب إلى أحد عشر فصلا، تناولنا فيها أساسيات البرمجة بلغة التربو باسكال،ففي الفصل الأول تطرقنا إلى البرمجة بلغة التربو باسكال وتحدثنا فيه عن بيئة التطوير المتكاملة IDE وشرحنا فيه قوائم بيئة التطوير هذه، في الفصل الثاني تطرقنا إلى الخوارزمية وكيفية إنشاء الخوارزمية، في الفصل الثالث تناولنا أولى أوامر اللغة وهي التعليمة الشرطية،في الفصل الرابع درسنا العمليات والتوابع الحسابية المعرفة في لغة تربو باسكال، أما في الفصل الخامس فدرسنا نمط المتحولات المنطقية Boolean ، في الفصل السادس تم دراسة نمط المتحولات المحرفية Char. وفي الفصل السابع درسنا أشكال أخرى لبنى التكرار في لغة التربو باسكال، وفي الفصل الفصل الثامن تناولنا بدراستنا بنية التحكم الشرطية Case والنمط المعدود، وفي الفصل التاسع درسنا نمط

المعطـيـات مصفوفة Array ، أما في الفصل العاشر فقد تم دراسة البرامج الجزئـية وبنوع من التفصيل وذلك لحاجتنا إليها كثيرا في الرياضيات. أما في الفصل الحادي عشر فتحدثنا عن نمط المعطيات مجموعة Set .

وضعنا في الكتاب دليلا للمصطلحات العلمية العربية ودليلا آخر للمصطلحات العلمية الإنكليزية للتسهيل على الدارس معرفة ترجمة أي كلمة علمية تمر معه. وكـان لابد من إضافة رسـائل الخطأ التي تظهر عند ترجمة البرنامج إلى لغة الآلة وقبل تنفيذه، إذا كان هناك خطأ في كتابة البرنامج من قبل المستخدم، وذلك مـن أجـل مسـاعدة الطالب على إزالة الخطأ، حيث يعطى شـرح عن الخطأ المرتكـب، والهدف إغناء الكتاب، ولكي يسـتغني الطالب عن أي مرجع آخر، وأضـيفت أيضا المخططات الكتابيـة. وأخيرا تم ذكر أسـماء المراجع العلمية الأجنبية والعربية التي استخدمت.

لقـد بذلت فـي تأليف هذا الكتاب كل الجهد الممكن مـن أجل إيصال المعلومة إلى الطالب وبالشكل الأفضل، ومن أجل مساعدة الطالب على فهم هذه اللغة وبشكل جيد، تم إضافة عددا كبيرا من المسائل المحلولة في نهاية الكتاب، ولكي يتمرن الطالب على حل المسائل العلمية وليقوم بكتابة برامج بنفسه.

وأخيرا أتقدم بالشـكر إلى كل من ساهم بإخراج هذا الكتاب، راجيا من الله تعالى أن أكون قد وفقت إلى تحقيق الغاية المرجوة منه، والله ولي التوفيق.

المؤلف

الدكتور محمد هشام بوادقجي

حلب ٢٠٠٢/٨/١٧

الفصل الأول

المدخل إلى البرمجة بلغة تربو باسكال

١-١- مقدمة:

تطورت البرمجة بواسطة الكمبيوتر بشكل سريع منذ ظهوره لأول مرة عام ١٩٥٠ وأصبح الكمبيوتر يلعب دوراً هاماً في حل المسائل المعقدة.

تم برمجة حواسيب الجيل الأول بلغة الأرقام الثنائية أو ما يسـمى لغة الآلـة Machine Language، ثم انتقل العمل في حواسيب الجيل الثاني إلى لغة أكثر تطوراً فاستبدلت الأرقام الثنائية ببعض الحروف اللاتينية كاختصار لأسماء العمليات والمتحولات. وأضيف للحاسـوب برنامجاً يترجم هـذه الحروف إلى الأرقام الثنائية المقابلة لها، سمي هذا البرنامج بالمجمع Assembly. إن البرامج المكتوبة بلغة الآلـة المرمزة ترميزاً ثنائياً أي تتالي 1 و 0 هي برامج سـريعة التنفيذ ولكن كتابتها عملية صعبة ومتعبة جداً.

في لغة المجمع هناك تقابل حرفي بين التعليمات ولغة الآلة Assembly Language، إن هـذه اللغة غير قادرة على التعبيـر مبـاشرة عـن العمليات المجردة التي نحتاجها في حلنـا لمسائلنا بالحاسوب، فهي تعتبر لغـة متدنية المستوى، وهي تتعلق بنوع المعالج وبالتالي فهي ليسـت مستقلة عـن الجهاز المستخدم، إضافة إلى ذلك إن البرمجة بها صعبة.

يتم نقل برنامج من لغة المجمع إلى لغة الآلة باسـتخدام برنامج خاص يسمى المجمع Assembler.

المجمع (مترجم إلى لغة الآلة)

برنامج بلغة المجمع ←——— برنامج بلغة الآلة

إن صعوبة البرمجة بلغة الآلة ولغة المجمع أدت إلى اعتماد لغات أقرب إلى تفكير المبرمج منها إلى تعليمات الحاسوب وسميت هذه اللغات باللغات عالية المستوى (High Level Programming Languages). وأولى هذه اللغات هي لغة الفورتران ولغة كوبول. وقد تم اعتماد مترجمات Compilers من أجل نقل التعليمات من اللغات العالية المستوى إلى لغة الآلة.

مترجم للغة (Compiler)

برنامج بلغة المجمع ←——— برنامج بلغة الآلة

١-٢- أهم لغات البرمجة:

في أوائل الخمسينات من القرن الماضي تم تطوير لغات برمجية أكثر تطوراً وكانت أولى هذه اللغات هي لغة الفورتران Fortran (Formula Translation) أو لغة ترجمة العلاقات وهي لغة الأغراض العلمية، خصصت لحل المسائل الحسابية والفيزيائية.

ومع تقدم صناعة الحواسيب والتطبيقات المختلفة المنفذة عليها أصبحت لغة الفورتران غير قادرة على تلبية جميع التطبيقات حيث كانت مخصصة للتطبيقات العلمية، وظهرت لغة البيسك Basic (Beginners All-purposes Symbolic Instruction Code) المخصصة للأعمال التجارية، وظهرت أيضاً لغات أخرى مثل PL/1 (Programming Language No.1) وهي لغة متعددة الاستخدامات لكن عدد كلماتها كبير وبالتالي فإن مترجمها كبير ولا يمكن العمل بهذه اللغة إلا في الحواسيب الكبيرة، وظهرت لغة ALGOL (Algorithmic

Language) وهي لغة قريبة من اللغة الإنكليزية وتُستخدم لحل المسائل الرياضية والفيزيائية.

مع بداية السبعينات من القرن العشرين ظهرت حواسيب الجيل الثالث وأصبحت هناك ضرورة ملحة للحصول على لغات برمجية جديدة متطورة فتم تطوير لغتين هما ALGOL-68 التي اعتُمدت في شركة IBM ولغة الباسكال Pascal.

١-٣- بداية ظهور لغة الباسكال:

إن بداية ظهور لغة الباسكال كانت على يد العالم والبروفسور السويسري نيكلوس ويرث Niklaus Wirth في جامعة زويرخ بسويسرا عام ١٩٦٨ واعتمد في تصميمها على لغة ألغول-٦٠ (ALGOL-60).

كان ويرث من أعضاء اللجنة التي صممت لغة ألغول (ALGOL). ومن خلال التعقيد الذي أدخل على لغة ألغول شعر ويرث بشيء من الإحباط وقرر أن يطور لغة برمجة بنفسه، بحيث تكون سهلة التعلم أولاً ومتكاملة بالقدر الكافي وقد نجح فكانت لغة بسيطة سهلة التعلم والقراءة وهي في نفس الوقت لغة عالية المستوى.

كانت الحاجة إلى هذه اللغة هي الحاجة إلى إيجاد لغة متكاملة يمكن استخدامها في العديد من المجالات التطبيقية وإيجاد لغة سهلة للمبرمجين ولا يأخذ مترجمها جزءاً كبيراً من ذاكرة الحاسوب.

تم وضع أول مترجم لهذه اللغة عام ١٩٧٠، ثم تم تطويره عام ١٩٧٣. وفي عام ١٩٧٤ وفي مركز الأبحاث المعلوماتي في كاليفورنيا تم تشكيل لجنة من الأساتذة لتطوير هذه اللغة ووضع أساس قياسي(Standard) لها وهكذا في عام ١٩٧٨ ظهرت هذه اللغة وبشكل موسع وقياسي.

في عام ١٩٨٠ ظهر أول كتاب بها وسميت باسم العالم الرياضي الفرنسي Blaise Pascal الذي وضع أول آلة حساب ميكانيكية عام ١٦٩٠ وكان عمره ١٨ عاماً.

١-٤- أهمية اختيار لغة تربو باسكال:

تعتبر لغة تربو باسكال هي اللغة الأمثل والأفضل لتعلم أساليب البرمجة الصحيحة، وهي لغة سهلة وبسيطة إضافة إلى أنها قوية. وتختلف لغة باسكال عن باقي لغات البرمجة في كونها لغة تعليمية وليست لغة برمجة فقط وهي لغة برمجة متكاملة وتُستخدم لكتابة برامج طويلة وفعالة.

تمتلك لغة تربو باسكال بعض الإضافات والتعديلات عن لغة باسكال القياسية بحيث جعلت منها لغة أقوى فعالية وأبسط استخداماً. فعلى سبيل المثال استخدامها ومعالجتها للسلاسل الرمزية String أمر في غاية السهولة. إضافة إلى ذلك تملك تربو باسكال مجموعة كبيرة من الإجراءات والتوابع مسبقة التعريف والجاهزة للاستخدام، إن بيئة تربو باسكال التي نتحدث عنها هي بيئة التطوير المتكاملة IDE (Integrated Development Environment) وهي عبارة عن بيئة برمجية كاملة.

من أجل كل هذا قمنا باختيار لغة التربو باسكال من أجل تدريس مقرر لغات برمجة لطلاب سنة أولى رياضيات.

ننوه هنا إلى أن بيئة تربو باسكال تتألف من محرر النصوص (Text Editor) ومترجم (Compiler) ومنقح (Debugger) ونوافذ وشاشات مساعدة وهذا يميزها عن غيرها من لغات البرمجة.

إضافة لما تقدم نوجز فيما يلي أهم الأسباب التي أدت إلى اختيارنا للغة البرمجة التربو باسكال:

١- وجود البنية الهيكلية المنتظمة.
٢- شمولها لأغلب بنى التحكم وبنى المعطيات والتي يمكن أن نجدها في لغات البرمجة الأخرى.
٣- الانتظام القواعدي، إن البنية القواعدية يمكن تعميمها دون حالات استثنائية.
٤- خاصة التحقق من الأنماط، والتي تعطي إشارة إلى الأخطاء عند الترجمة (الانتقال من لغة المصدر أو نص البرنامج إلى لغة الآلة) وقبل تنفيذ البرنامج. مما يسمح بتصحيح الأخطاء مبكراً وبسهولة.
٥- البساطة القواعدية للغة والتي أتاحت توفرها على أغلب الحواسيب وخصوصاً الحواسيب الشخصية.

١-٥- أهمية خوارزمية الحل:

يجب التنويه إلى أن الخطة الأساسية في حل أي مسألة بالحاسوب هي كتابة خوارزمية الحل وبانتظام، أما نقل هذا الحل إلى برنامج بلغة برمجة معينة فيأتي في مرحلة ومرتبة ثانية، حيث يجب معرفة قواعد لغة البرمجة وكيفية مقابلة لغة الخوارزمية ببنى اللغة المطلوب كتابة البرنامج بها.

حيث تعتبر الخوارزمية واحداً من أوسع المفاهيم في الرياضيات المعاصرة، وكلمة خوارزمية تعني طريقة الحل الدقيقة والمفصلة، وذات الخطوات المتسلسلة لمسألة معينة حيث ليس من الضروري أن يتألف الحل من عمليات حسابية فقط.

والشيء الأساسي والضروري في مقرر لغات برمجة هو كتابة خوارزمية الحل للعديد من المسائل الرياضية وغيرها ثم كتابة برنامج الحل باستخدام لغة البرمجة تربو باسكال.

مثال: Program Goodby;.

يمكن تسمية البرنـــامج باختيار أي اســم حروفه تنتمي إلى مجموعـــة الحروف اللاتينية من A إلى Z ومجموعة الأرقام من 0 إلى 9 والحرف الخاص (_) شرط ابتداء الاسم بحرف أبجدي. تسمح لغة تربو باسكال أن يصل طول الاســم إلى أكثر من 100 محرف(١٢٦ على الأكثر)،في لغة باسكال القياسية لم يسمح بأكثر من ١٥ محرف.

أمثلة:

Byby, Seeyou, My_multiple, ...

وهي أسماء تعريف صحيحة أمـــا استخدام: 3Q كاسم تعريف فهو خطأ قواعدي لأنه لا يجوز أن يبتدئ الاسم برقم.

٢- قسم التصريحات (قسم إعلان المتحولات): Declaration Part

يتم في هذا القسم الإعلان عن المتحولات المستخدمة في البرنامج وذكر نوعها ويبدأ القسم بالكلمة المحجوزة Var. إن تعليمة التصريح تأخذ الشكل:

<L1>, <L2>, ... ,<LN>: <Type>;

حيث:

<L1>, ... ,<LN>: هي أسماء تعريف المتحولات.

<Type>: النمط.

مثال ١:

Var x: integer;

يعني أن x هو عدد من نمط صحيح.

مثال ٢:

Var y: real;

يعني أن y هو عدد من نمط حقيقي.

يعتـــبر قسم التصريح الجزء الأهم في نجاح البرنامج إذ تحدد من خلاله أنواع المعطيـــات التي ستستخدم لتخزين وحفظ عناصر وقيم المسألة البرمجية وكذلك مجموعة الأعمال الجزئية التي ستبسط خطوات البرنامج.

٣- قسم أوامر البرنامج: Order Part

يبتدئ هـــذا القســـم بالكلمة المحجوزة Begin وينتهي بالكلمة المحجوزة End يتبعها نقطة أي .End يسمى هذا القسم بجسم البرنامج وهو قسم التعليمات ويلي قسم التصريحات.

١-٩- ملاحظات عامة حول بنية البرنامج:

١- جميع الأوامر في لغة تربو باسكال تنتهي بفاصلة منقوطة.

٢- لا أهمية لعـــدد الفراغـــات الفاصلة بين كلمتين في نص البرنامج، يكفي وجود فراغ واحد على الأقل.

مثلاً كتابـــة Program Good; تكافئ كتابة Program Good;.

٣- لا يميز تربو باسكال بين الأحرف الكبيرة والأحرف الصغيرة فالشكلان متكافئان.

مثلاً الكلمات program، Program، PROGRAM متكافئة.

٤- بما أن .End ليس أمراً فإن السطر الذي يأتي قبل ;End يمكن أن يترك الأمر الذي يسبقه دون فاصلة منقوطة.

٥- بدء تسمية البرنامج برقم هو خطأ قواعدي وذلك لأن اسم البرنامج يجب أن يبدأ بحرف أبجدي. ولا يجوز إدخال الرموز الخاصة والمزدوجة في تسمية البرنامج عدا الرمز الخاص (_) والذي يمكن إدخاله في اسم البرنامج.

٦- تسمى الكلمات Program، Var، Begin بكلمات مفتاحية (Key words) أو كلمات محجوزة (Reserved words).

٧- يسمح بكتابة أكثر من تعليمة على سطر واحد في لغة تربو باسكال مثلاً:

```
Program by; Var n : Integer; Begin writeln(n);

End;
```

٨- يمكن إضافة التعليقات أو الملاحظات ضمن البرنامج وفي أي جزء منه شرط أن يتم وضعه بين قوسين على الشكل (* *) أو الشكل { }. وذلك لتسهيل قراءة البرنامج وفهمه. إن هذا التعليق لا يهم مترجم اللغة ومن ثم يمكن كتابة النص الذي نريد ضمنه.

مثلاً: (* This is my Program *)

أو بالشكل { This is my Program }

٩- يتم تحديد مجموعة من التعليمات بواسطة كلمة التحديد Begin في بداية الكتلة و End; في نهاية الكتلة.

١-١٠- الأنواع القياسية في تربو باسكال:

يجب أن يعلن عن المتحولات والأسماء المستخدمة في برنامج تربو باسكال وذلك في قسم إعلان المتحولات، أي يجب أن نذكر نوع المتحول الذي تنتمي إليه المتحولات.

إن الأنواع القياسية لهذه المتحولات هي:

١- Integer: وهي مجموعة كل الأعداد الصحيحة.

٢- Real: وهي مجموعة كل الأعداد الحقيقية ذات الفاصلة العائمة والفاصلة الثابتة.

٣- Char: وهي مجموعة كل رموز شيفرة الآلة أي مجموعة محارف ASCII.

٤- Byte: إن مجموعة تعريف لهذا النوع مؤلفة من 256 عدداً من 0 وحتى 255 وهي عبارة عن مجموعة جزئية من مجموعة الأعداد الصحيحة.

٥- Boolean: مجموعة التعريف تتألف من الرمزين المعرفين مسبقاً وهم True و False.

٦- String: وهي مجموعة كل رموز السلاسل الحرفية ويمكن أن يتبع كلمة String رقم بين قوسين متوسطين يدل على طول السلسة الحرفية.

مثال:

```
Var N1,N2: String [32];
```

إن المتحولات N1 و N2 هي متحولات سلاسل حرفية يمكن أن يكون أكبر طول لها 32 محرفاً. وإذا تركت String دون أقواس لتحديد الطول فإن أكبر طول يمكن أن تأخذها هو 255 محرفاً.

١-١١- المميزات غير المسموح بها في تسمية المتحولات:

هناك عدداً من الشروط يجب اتباعها في تسمية المتحولات وهي:

١- لا يسمح بوجود إشارة الناقص في تسمية المتحول:

مثال: لا يجوز الكتابة مثلاً D0-by كاسم للمتحول.

٢- لا يسمح بكتابة الرمز الأول عدد في تسمية المتحول:

مثال: لا يجوز كتابة 3XY ويسمح بكتابة X3Y أو XY3.

٣-لا يسمح بوجود فراغ في تسمية المتحول:

مثال: لا يجوز كتابة My Name كاسم للمتحول وإنما يسمح بكتابة MyName.

٤- لا يسمح بكتابة أكثر من 63 محرفاً (رمزاً) في تسمية المتحول. وذلك لأن مترجم التربو باسكال يقرأ فقط أول 63 محرفاً.

١-١٢- أنواع معطيات الأعداد الصحيحة:

تزودنا لغة تربو باسكال بعدد من أنواع الأعداد الصحيحة وتختلف هذه الأنواع وذلك حسب حجم مواضع التخزين المخزنة لكل نوع، وفي إمكانية تمثيل الأعداد السالبة في هذه الأنواع أم لا.

١-١٢-١- نوع المعطيات عدد صحيح: Integer

يستخدم هذا النوع لتمثيل الأعداد من 32.768- إلى 32.767. إذ يخصص المترجم 2 بايت أي 16 بت لكل قيمة من النوع الصحيح وبناءً عليه يمكننا تخزين قيم ضمن المجال من (-2^{15}) وحتى ($2^{15}-1$) حيث يُحجز البت السادس عشر لإشارة القيمة. فإذا كان هذا البت يساوي (1) تكون القيمة سالبة وإذا كانت قيمته (0) تكون القيمة موجبة.

وتمتلك لغة تربو باسكال أيضاً ثابتاً مسبق التعريف يدعى Maxint

قيمته تساوي أعظم قيمة صحيحة يمكن أن تستخدم في التطبيقات وهو في لغة تربو باسكال يساوي (32.767).

١-١٢-٢- نوع المعطيات عدد صحيح طويل: Longint

يمكننا نوع المعطيات هذا عدد صحيح طويل Longint من تمثيل مجال أكبر من القيم التي يمكن تمثيلها بواسطة نوع المعطيات (عدد صحيح Integer). حيث يحجز المترجم أربعة بايتات (32 بت) لكل متحول من النوع الصحيح الطويل Longint وهذا يعطينا مجالاً للقيم يتراوح بين (-2^{31}) و ($2^{31}-1$) أي بين 2.147، 483.648- و 2.147، 483.647 وهو الحد المسموح به. وأي قيمة أكبر من هذا المجال تعطينا حالة فيض (Overflow) وجواب خاطئ.

وتمتلك لغة تربو باسكال أيضاً ثابت مسبق التعريف هو Maxlongint الذي قيمته تساوي أعظم قيمة يمكن تمثيلها من النوع Longint.

١-١٢-٣- نوع المعطيات كلمة: Word

يُخصص لهذا النوع كلمة Word في لغة تربو باسكال 2 بايت (16 بت) ويستخدم لتمثيل القيم الموجبة فقط وتستثنى السالبة، لذلك يكون مجال الأعداد يتراوح ما بين 0 و 65535 أي بين (0) و ($2^{16}-1$). ويختلف عن نوع المعطيات الصحيح في كون الخانة اليسرى السادسة عشره تستخدم في الحسابات ولا تترك للإشارة.

١-١٢-٤- نوع المعطيات عدد صحيح قصير ونوع معطيات بايت:

Shortint and Byte

يخصص لكل نوع بايتاً واحداً أي حجرة ذاكرة واحدة أي (8 بت لكل واحد). يمثل نوع المعطيات عدد صحيح قصير Shortint الأعداد من (128-)

إلى (127). وفي هذا النوع يمثل البت الأعلى أو الخانة اليسرى إشارة العدد.

أما نوع المعطيات بايت Byte فهو لا يستخدم بتاً للإشارة وبالتالي فهو يمثل الأعداد من (0) وحتى (255). وهـذا النوع Byte مفيداً وذلك عند قراءة ملف مكتوب بالشفرة الثنائية. ولـن يعارض المترجـم في الإسناد للمتبادل بين هذين النوعين.

يلخص الجدول التالي أنواع المعطيات الصحيحة كلها ومجالات الأعداد المنتمية لكل نوع من هذه الأنواع وحجم التخزين اللازم للمتحولات الخاصة بكل نوع من أنواع المعطيات هذه:

نوع المعطيات	حجمه	القيمة الدنيا	القيمة العظمى
Shotint	1 byte	-128	127
Byte	1 byte	0	255
Integer	2 byte	-32768	32767
Word	2 byte	0	65535
Longint	4 byte	2,147, 483,647	-2,147, 483,648

١-١٣- عبارتي الإدخال والإخراج: Read, Write

١-١٣-١- عبارة الإدخال Read:

وظيفة هـذه العبارة إدخال المعلومات إلى الحاسوب عـن طريق لوحة المفاتيح ولها الشكل:

Read (x) ;

أو الشكل:

Read (X, Y, Z, ...) ;

إذ يمكن إدخال قيمة واحدة مـن على سطر واحـد أو مجموعة قيم من على سطر واحد وذلك حسب عدد المتحولات المطلوب إدخال قيمها.

١-١٣-٢- عبارة الإخراج Write:

وظيفة هذه العبارة إظهار المعلومات على شاشة الحاسوب ولها الشكل:

Write (x) ;

أو الشكل:

Write (X, Y, Z, ...) ;

وذلـك حسب الضرورة ويمكن طباعة قيمة واحـدة على سطر أو مجموعة قيم المتحولات على نفس السطر حسب ما هو مطلوب.

١-١٣-٣- شكل آخر لعبارتي الإدخال والإخراج:

يمكن إضافة الحرفين Ln إلى كل من عبارتي الإدخال والإخراج وذلك للبـدء في سطـر جديـد. وبالتـالي يصبح شكل التعليمتين بالشكل: Readln و Writeln.

إن تعليمة Read (x) تعني إقرأ قيمة المتحول x وابقى في نفس السطر أمـا تعليمة Writeln (x) فتعني إقرأ قيمة المتحول x وانتقل إلى سطر جديـد. وكذلـك أيضاً في تعليمة Write (x) تعني اكتب قيمة x على الشاشة وابقى في نفـس السطر أما تعليمة Writeln (x) فتعني اكتب قيمة x على الشاشة ثم انتقل إلى سطر جديد.

مثال ١: إذا كان لدينا في البرنامج الأوامر:

```
Read (x, y) ;

Read (z) ;
```

وإذا ما تم إدخال المعطيات بالشكل التالي:

```
1  4  5

6  7  8
```

فإن x سوف تأخذ القيمة 1 و y القيمة 4 أما z فسوف تأخذ القيمة 5.

مثال ٢: إذا كان لدينا في البرنامج الأوامر

```
Readln (x, y) ;

Readln (z) ;
```

وكانت القيم المدخلة هي:

```
1  4  5

6  7  8
```

فإن x سوف تأخذ القيمة 1 و y القيمة 4 أما z فسوف تأخذ القيمة 6، لأن الأمر السابق قد انتقل بعد القراءة إلى سطر جديد. وهو هنا يبتدئ بالرقم 6.

١-١٣-٤- التحكم بشكل إخراج وطباعة المتحولات:

يمكن التحكم بشكل إخراج وطباعة المتحولات وذلك من خلال حجز مجالات لهذه المتحولات مثلاً:


```
Write ('x = ', x : 8 : 3)
```

إن الكتابة السابقة تعني أنه سوف يتم طباعة x = ثم أمامها قيمة المتحول x على نفس السطر وسوف يبقى على نفس السطر حيث يتم حجز ثماني مواقع للمتحول X بكامله بما في ذلك الفاصلة والأعداد بعد الفاصلة، يحدد ثلاث أعداد لما بعد الفاصلة، طبعاً الحجز لما بعد الفاصلة فقط من أجل الأعداد الحقيقية حيث يفترض هنا أن X هو عدد حقيقي ويتم ملئ المواقع ابتداءً من اليمين وإلى اليسار وتبقى بقية الخانات الواقعة إلى اليسار فارغة إذا لم تستخدم. أما إذا كان طول المتحول أكبر من ثماني مواقع عندها يتم تدويره.

إن العدد 3 يعني أن عدد الأرقام بعد الفاصلة هو 3. وهذا مسموح به فقط في تمثيل الأعداد الحقيقية ذات الفاصلة الثابتة. أما إذا كان العدد حقيقي وكان أمر الطباعة بالشكل:

```
Write ('x = ', x)
```

أي لم يتم حجز أي مواقع عندها سوف يتم الإخراج على شكل عدد حقيقي ذو فاصلة عائمة أي من الشكل:

أس ± E . عدد ±

وإذا كان معلنا عن المتحول X أنه صحيح فيتم إخراجه على شكل عدد صحيح وبالطول المحجوز له. فمثلا في الأمر Write (x : 3) سوف يتم حجز ثلاث مواقع للعدد X أما إذا كان العدد x مؤلف من أكثر من ثلاث خانات عندها يقوم البرنامج بحجز مكان مناسب بشكل أوتوماتيكي وإذا كان طول X أقل من 3 فيكتب العدد ابتداءً من اليمين ويترك المجال المتبقي ومن اليسار فارغاً.

١-١٤- عبارات USES، CRT، CLRSCR:

تملك لغة تربو باسكال كماً كبيراً من الإجراءات والتوابع مسبقة

التعريف والتي يمكن استدعاؤها من خلال البرنامج، هـذه جميعها موجودة في مكتبة لغة تربـو باسكال وبالتحديـد في الملف Turbo.tpl وتمتلك أيضاً بعض الوحدات الجاهزة التي يمكن استخدامها في برامجنا وهـذه موجودة في المكتبة Turbo.tpl وهي:

١- الوحدة CRT.

٢- الوحدة DOS.

٣- الوحدة Graph.

٤- الوحدة System وغيرها ... .

إن الوحدة CRT هي وحـدة تحتوي على تعريفـات وروتينات لمعالجة الدخل والخرج ولإدارة الشاشـة النصية وما تحتاجه من عمليات. روتينات هذه الوحدة سريعة التنفيذ. وتستعمل هذه الوحدة مع USES بالشكل التالي:

```
USES CRT ;
```

وتكتب عادةً بعد رأس البرنامج. ومن أجل استعمال خصائص هذه الوحدة يمكن كتابة الأمر:

```
Clrscr ;
```

في قسم أوامر البرامج أي بعد كلمة Begin مباشرة وذلك من أجل مسح الشاشة. لكي تظهر نتائج البرنامج فقط على الشاشة.

أمثلة:

مثال١:

إن الجدول التالي يبين قيـم المـتحولات a,b,c في كل خطوة مـن خطوات البرنامج.

```
Program Trace;
Var
a,b,c,x:integer;
Begin
x:=5;
b:=1;
c:=x+b;
a:=x+4;
a:=c;
b:=c;
a:=a+b+c;
c:=c mod x;
c:=c*a;
a:=a mod b;
writeln(a,b,c);
End.
```

a	b	c
	1	
	1	6
9	1	6
6	1	6
6	0	6
18	6	6
18	6	1
18	6	18
0	6	18
0	6	18

مثال ٢:

اكتب نتائج تنفيذ البرنامج التالي:

```
Program Exercise;
Const
   L=10;
Var
  price,cost:integer;
       ch:char;
Begin
   price:=10;
   cost:=price*L;
```

```
    ch:='A';
    writeln('cost is');
    writeln(cost);
    writeln('price is '.price,' cost is ',cost);
    write('grade ',ch,' cost is');
    writeln(cost);
readln
End.
```

إن ناتج تنفيذ البرنامج هو:

```
cost is
100
price is 10 cost is 100
grade A cost is 100
```

مثال٣:

حدد أي من الكلمات التالية هي كلمات محجوزة أو مميزات مستخدمة في لغة تربو باسكال .

Temp
Var
Begin
Const
Constant
Flag
Program
End

مثال٤:

إن الجدول التالي يبين قيم المتحولات X و Y و Z في كل خطوة من خطوات البرنامج وذلك وفق القيم المعطاة أدناه.

```
Program trace ;                    X      Y      Z
Var
X,Y,Z :integer ;
Begin
   Readln(X,Y,Z);                  25     20     2
   X:=X+Y+Z;                       47     20     2
   X :=X mod Z ;                   1      20     2
   Z:=Y div 2;                     1      20     10
   Y :=X*X+10 ;                    1      11     10
   X :=(X-10)*3 ;                  -      11     10
                                   27
                                   -      11     10
Writeln(X,Y,Z);                    27
End.
```

علما أن المعطيات هي:

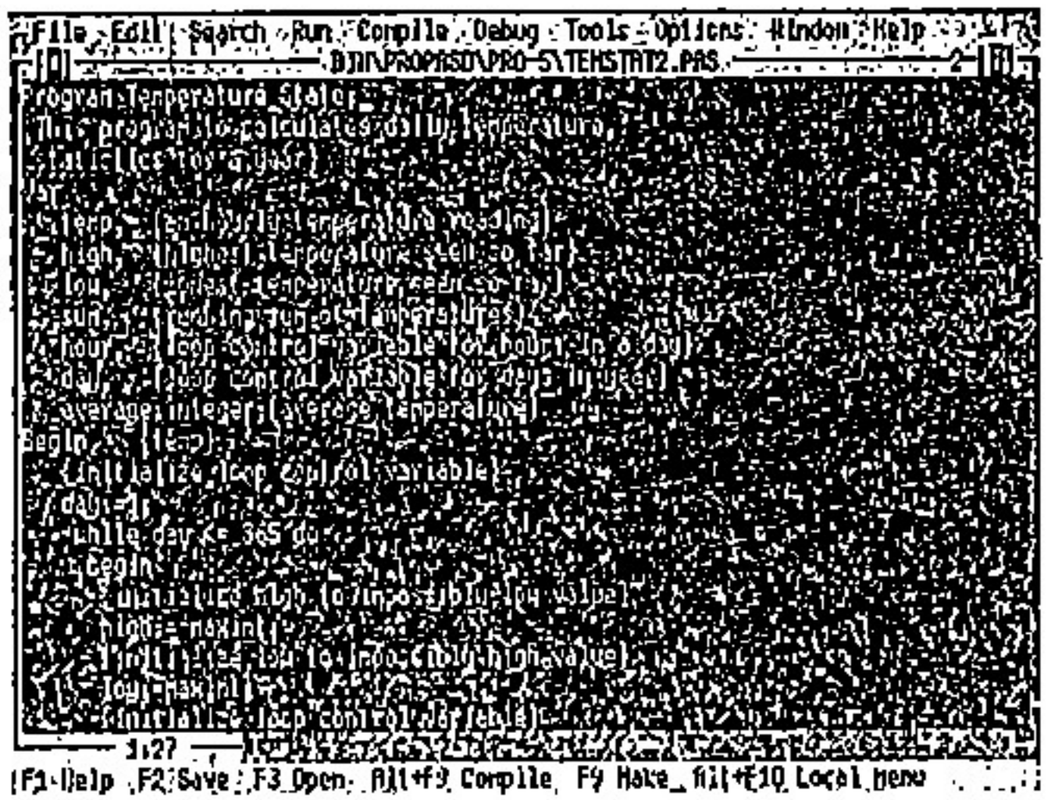
```
24  20
2   10
```

١-١٥- بيئة التطوير المتكاملة IDE في تربو باسكال:

Integrated Development Environment

تسهل بيئة تربو باسكال عمليات الترجمة Compile والتحرير Edit والتنفيذ Run والتنقيح Debug والتي لا غنى عنها عند كتابة البرامج.

إن الشكل التالي يرينا بيئة تربو باسكال موضحاً عليه عناصرها:



تتألف بيئة تربو باسكال من عدة عناصر نذكر منها ما يلي:

١- شريط القوائم Menu Bar:

يحتوي على مجموعة مـــن القوائم والتي تستخدم لاختيار الأوامر. يقع شريط القوائم في أعلى الشاشة وفيه عشر قوائم مـــن File وحتى Help. يمكن الوصول إلى قائمة ما بالضغط على المفتاح Alt مع أول حرف من حروف هذه القائمة. مثلاً لفتح القائمة Edit نضغط على المفتاحين Alt + E معاً.

٢- سطر الحالة Status Line:

يقع أسفل الشاشة ويحتوي على رسائل تشرح العمـــل الحالي أو أسماء مفاتيح يمكن أن تكون مفيدة ضمن الســياق الحالي، ويتـــم الانتقال بيـــن هاتين الوظيفيتين كلما انتقلنا من القوائم إلى أعمال أخرى.

٣- صندوق الإغلاق Close Bar وشريط التدرج Scroll Bar:

يستخدمان مـــن خلال الفأرة لإغلاق النوافذ وللتنقل ضمـــن الملف على الترتيب.

٤- شريط العنوان Title Bar:

ويقع في أعلى كل نافذة، ويحتوي على اسم الملف، وعند عدم تعيين اسم للملف يكون الاسم الافتراضي هو Nonamecoo.pas.

٥- رقم النافذة Window Number:

عندما تُفتح عدة نوافذ ضمن البيئة، فإن كل نافـــذة تعرَّف برقـــم حيث تعطى النافذة الأولى الرقم (١) والثانية الرقم (٢) وهكذا.

٦- صندوق التكبير Zoom Box:

ويقع في الزاوية العليا اليمينية من الشاشة، والضغط عليه بالفأرة يسبب ملء النافذة الحالية للشاشة بأكملهـــا أو تعود لحالتهـــا الأولى.

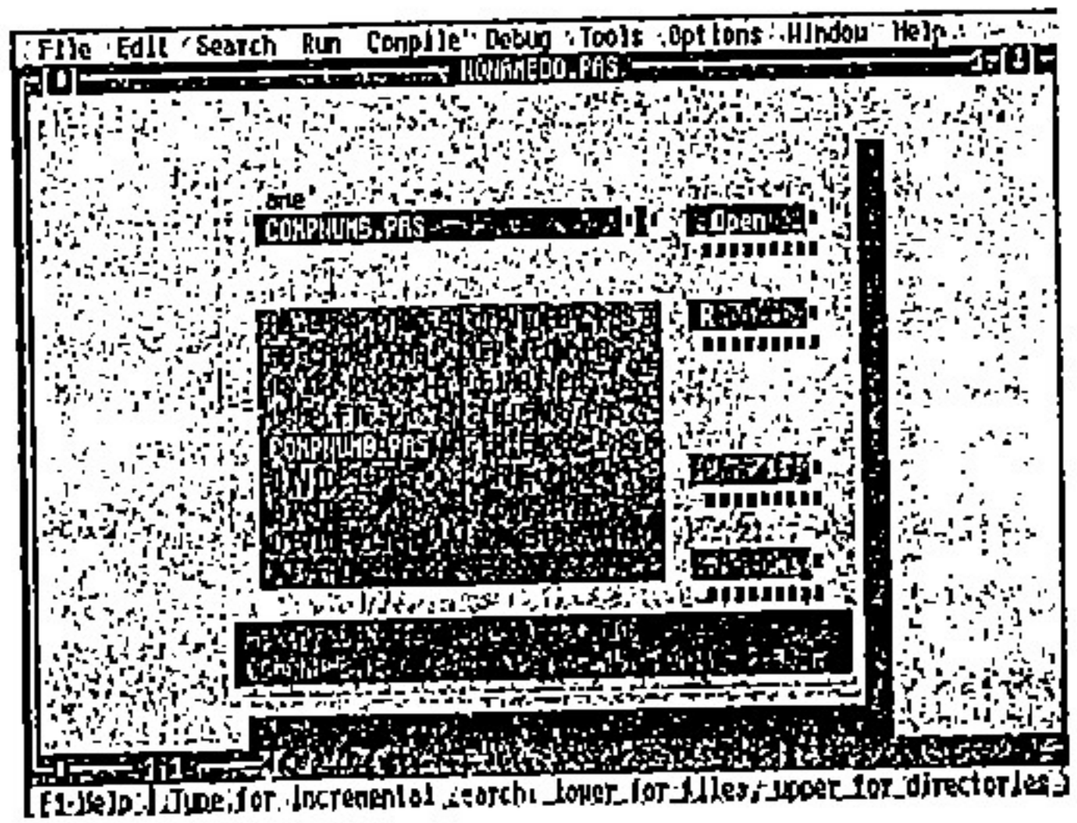
إن الضغطة الأولى عليه مـــع كون النافـــذة في حجمها الطبيعي يجعل النافذة تملأ الشاشة، وبضغطه ثانية عليه ستعود إلى حالتها السابقة.

٧- زاوية تغييـــر الحجم Resize Corner : يمكننا أيضـــاً تغيير حجم النافذة باستخدام الزاوية السفلية اليمينية، وذلك مـــن خلال الضغط عليها بالفأرة لتغيير حجم النافذة تغييراً طفيفاً.

١-١٥-١- فتح ملف:

حتى يتم فتح ملف لا بد في البداية مـــن فتح قائمة File إما بالنقر على File بواسطة الفأرة أو بالضغط على المفتاحين Alt + F معاً.

ومـــن ثم بالضغط على المفتاح O أو النقر بالفأرة على الرمـــز Open عـــندئذ تظهر شاشة تحتوي أسماء الملفـــات المكتوبة بلغة تربو باسكال، طبعاً إذا كانت هناك ملفات مكتوبة بلغة تربو باسكال، وسوف تبدو الشاشة كما يلي:



يجب اختيار الملف المطلوب فتحة بالنقر فوقـــه أو بكتابة اسمه مباشرة حيث يمكن كتابة اسم الملف مع امتداده أو بدونـــه وذلك لأن بيئة تربـــو باسكال

تفترض أن الأسماء لها الامتداد Pas.

بالإضافة إلى ذلك يمكن التنقل ضمن صندوق الملفات وانتقاء ملف معين ومن ثم الضغط على مفتاح الإدخال Enter لتحميل الملف.

يحوي صندوق الملفات كل الملفات في الدليل الحالي التي لهـــا الامتداد (Pas.) ويمكن أيضاً الوصول إلى كل أدلـــة القرص باستخدام مفاتيـــح الأسهم للانتقال إلى الملف المطلوب ومن ثم ضغط المفتاح Enter.

بعد تحديـــد اسم الملف والضغط على المفتاح Enter فإن محتويات هذا الملـــف ستُحمل إلى نافـــذة التحرير. إن نافذة التحرير هي النافذة التي يتم مـــن خلالها قراءة البرنامج أو كتابته.

١-١٥-٢- تنفيذ البرنامج:

بعد الانتهاء مـــن تحميل البرنامج الموجود في الملف المذكور إلى نافذة التحرير، يمكننا عندها النظر إلى محتويات هذا الملف.

يمكن أن يأخذ اسم الملف أي تتابع مـــن الرموز، شريطة أن يكون لـــه الامتداد (Pas.) وأما اسم البرنامج نفسه المخزن في الملف فيمكن أن يكون لـــه مختلفاً عن اسم الملف.

يتم تنفيـــذ أي برنـــامج مباشـــرة مـــن خلال الضغط على المفتاحيـــن Ctrl + F9 حيث سيترجم هذا البرنامج في الذاكرة أو على القرص ومن ثم ينفذ، ويعتمد موقع ترجمـــة البرنامج في الذاكرة أم في القرص على أحـــد الخيارات الموجودة في IDE هـــذا الخيار هو Destination أي الموقع الهدف الذي ستتم الترجمة فيه. هذا الخيار موجود ضمن قائمة الترجمة Compile، ويتم الوصول إليه بالضغط على المفاتيح ALT + C + D ومن خلالها يمكن التنقل بين الذاكرة والقرص، هـــذا يعني أنـــه إذا كانت قيمـــة هـــذا الخيـــار مثلاً هي القرص أي Destination Disk، عندئذ الضغط على المفاتيح السابقة سوف يحول هذه القيمة إلى الذاكرة Destination Memory، والعكس بالعكس. والقيـــم التي نحددهـــا كخيارات لبيئة التطويـــر المتكاملة تستمر حتى نغيِّرها، أو ننهي العمل مع هذه البيئة.

إذا أردنا المحافظة على هـذه الخيارات علينا تخزينها على القرص من خـلال الضغط على ALT + O + S والتي تعني اختيـار قائمـة الخيـارات Options واختيار أمر التخزين Save. يطلق على النافذة التي يظهر فيها خرج البرنامج بشاشة المستخدم User Screen أو شاشـة الخرج Out Put Screen. فعند تنفيذ البرنـامج يمكننا رؤية شاشـة الخرج بالضغط على ALT + F5 لأن وظيفة زوج المفاتيح هو التنقل بين شاشة الخرج ونافذة التحرير على التناوب.

١- ١٥- ٣- تعديل البرنامج:

عند قراءة برنامج نلاحظ وجود أحرف صغيرة وأحرف كبيرة، وهذا لا يسبب أية مشكلة أثناء الترجمة أو التنفيـذ وذلك لأن الحرفـان الصغير والكبير متكافئان في لغة باسكال. وإذا أردت إحداث تغير في بعض الكلمات فيمكن نقل المشيرة بواسطة مفتاح الأسهم إلى الكلمة المطلوبة.

بالضغط على المفتاح Insert يتغير شكل المشيرة ويتـم الكتابـة فوق الحروف السابقـة ويتم استبدالها بحروف أخرى كبيرة أو صغيرة. أما إذا أردنا حشر كلمـات إضافية فيكفي أن يكون مفتاح Insert غير فعال ووضع المشيرة فـي الحالة العاديـة وتنقل المشيرة إلى المكان المراد حشر كلمـات أو حروف إضافية فيه، يتم الكتابة مباشرة وتتم الإضافة.

وبعد الانتهاء مـن العمل مع بيئة تربـو باسكال نضغط على المفتاحين Alt + X أو اختيار الأمر Exit من القائمة File أو بالضغط على Alt + F + X مباشرة. ونتيجة التغييرات التي أحدثتها على البرنامج سوف تُسأل إذا كنت تريد حفظ التغييرات أم لا وفي حال الإيجاب اضغط Y أو Enter وإلا اضغط N.

١-١٥-٤- نافذة التحرير:

يوجد عددان في الزاوية السفلى اليسارية لنافـذة التحرير، هذان العددان يمـثلان إحداثيات السـطر والعمود للموقـع الحالي للمشـيرة، على اعتبار أن إحداثيـات الزاوية العليا اليسـارية للنافذة هي (1:1).

يمكن أن يحتوي السطر على أكثر من 249 رمزاً في محرر النصوص الخـاص بتربـو باسكال. وهـذه المعلومـات يمكن الحصول عليها من شاشة المساعدة التي تظهر بالضغط على F1، واختيار العبارة المضيئة والضغط على Enter.

١-١٦- التعمق في بيئة التطوير المتكاملة IDE:

سوف نتحدث في هـذه الفقرة عن أوامـر محرر النصوص، وحتى تتم الاستفادة من أوامر التحرير هذه يجب تجريب كل أمر من هذه الأوامر حتى يتم استخدامه عند الحاجة إلى كتابة البرامج وتنقيحه فيما بعد.

١-١٦-١- التنقل ضمن الملف:

يمكننا استخدام مفاتيح الأسهم العادية، أو الموجودة على لوحـة المفاتيح العددية بعد إطفاء المفتاح Num Lock، وإلا فإن مفاتيح لوحـة المفاتيح العددية ستمثل أرقاماً عوضاً عن الأسهم. نستخدم مفاتيح الأسـهم للتنقل ضمن الملف يمنة ويسرة، وللأعلى وللأسفل، حسب طبيعة السـهم.

ينقل المفتاح Home المشـيرة إلى بداية السـطر، على عكس المفتـاح End، الذي ينقـل المشـيرة إلى نهايـة السـطر، أما المفتاح Page Up فينقل المشـيرة مقدار شاشة إلى الأعلى وحتى بداية الملف، أما المفتاح Page Down فينقل المشيرة مقدار شاشة إلى الأسفل، وحتى نهاية الملف.

نذكر فيما يلي ملخص لأوامر تحريك المشيرة:

Ctrl + E أو السهم العلوي	ينقل المشيرة إلى السطر السابق لموقعها
Ctrl + X أو السهم السفلي	ينقل المشيرة إلى السطر التالي لموقعها
Ctrl + S أو السهم اليساري	ينقل المشيرة رمزاً واحداً لليسار
Ctrl + D أو السهم اليميني	ينقل المشيرة رمزاً واحداً لليمين
Ctrl + A أو السهم اليساري + Ctrl	ينقل المشيرة إلى بداية الكلمة الحالية أو السابقة
Ctrl + F أو السهم اليميني + Ctrl	ينقل المشيرة إلى بداية الكلمة التالية
Ctrl + Q +S أو Home	ينقل المشيرة إلى بداية السطر الحالي
Ctrl + Q +D أو End	ينقل المشيرة إلى نهاية السطر الحالي
Ctrl + R أو Page Up	ينقل المشيرة إلى أعلى الشاشة
Ctrl + C أو Page Down	ينقل المشيرة إلى أسفل الشاشة
Ctrl + Page Up	ينقل المشيرة إلى بداية الملف
Ctrl + Page Down	ينقل المشيرة إلى نهاية الملف
Ctrl + Home	ينقل المشيرة إلى السطر العلوي من الشاشة، مبقياً المشيرة في نفس العمود
Ctrl + End	ينقل المشيرة إلى السطر السفلي من الشاشة، مبقياً المشيرة في نفس العمود

١-١٦-٢- إضافة وحذف نص ما:

Ctrl + V أو Insert	يتنقل بين نمطي الحشر أو الكتابة فوق الحروف
Ctrl + N	يضيف سطراً فارغاً في موضع المشيرة

	الحالي، بعد إزاحة الأسطر إلى الأسفل
Del أو Ctrl + G	يحذف رمزاً في موضع المشيرة الحالي
Ctrl + H أو Backs Pace	يحذف رمزاً على يسار موضع المشيرة الحالي
Ctrl + Y	يحذف السطر الحالي
Ctrl + Q + Y	يحذف من موضع المشيرة الحالي إلى نهاية السطر الحالي
Ctrl + T	يحذف من موضع المشيرة الحالي إلى نهاية الكلمة الحالية
Ctrl + E + V أو Alt + Backs Pace	يلغي آخر حركة أو عملية حذف

١-١٦-٣- معالجة كتل من النصوص:

يمكننا من خلال محرر النصوص تحديد كتل كاملة ومن ثم معالجتها حذفاً أو نسخاً، الخ ... وهذه الكتل ستظهر على الشاشة مضاءة بشكل زائد أو بلون مختلف. وفيما يلي ملخص عن هذه الأوامر:

Ctrl + K + B	يُعلم بداية كتلة من النصوص
Ctrl + K + K	يعلم نهاية كتلة من النصوص
Ctrl + K + T	يُعلم كلمة واحدة، تكون المشيرة واقعة عليها، أما إذا كانت المشيرة واقعة بين كلمتين، فسيُعلم هذا الأمر الكلمة الواقعة يسار المشيرة.
Ctrl + K + Y	يحذف الكتلة المعلمة.
Ctrl + K + V	ينقل الكتلة المعلمة من موقعها الحالي إلى موقع المشيرة
Ctrl + K + C	ينسخ الكتلة المعلمة إلى موقع ثان، وبهذا تظهر الكتلة

نفسها مرتين ضمن الملف

Ctrl + K + W	يكتب الكتلة المعلمة في ملف جديد تحدّد اسمه مفترضاً أن امتداده (Pas.)، إن لم نحدّد غير ذلك.
Ctrl + K + R	يقرأ محتويات ملف يتم تحديد اسمه، ويضيف محتوياته في موقع المشيرة الحالي
Ctrl + K + P	يطبع الكتلة المعلمة، وفي حال عدم وجود أي كتلة معلمة يطبع الملف كاملاً

يمكن استخدام الرموز البديلة ضمن أسماء الملفات من أجل عرض الملفات ذات خصائص ما، مثلاً Pas.* تعرض الملفات ذات الامتداد Pas فقط مهما كان اسمها الأول، عندها تظهر هذه الملفات، بتحديد أحدها والضغط على Enter يظهر محتويات الملف على نافذة التحرير.

١-١٦-٤- وضع علامات مواضع:

يمكننا محرر نصوص تربو باسكال من تعليم عشرة مواضع ضمن الملف، وبعد وضع هذه العلامات، يمكننا الانتقال بسرعة قفزاً إلى هذه المواضع. هذه التقنية مفيدة جداً عندما يكون الملف طويلاً.

مثلاً إذا أردنا وضع علامة على بداية جسم البرنامج، علينا نقل المشيرة إلى ذلك الموقع، والضغط على Ctrl + K، ومن ثم أحد الأرقام من 0 وحتى 9، فإذا أردنا الانتقال إلى هذا الموضع من أي مكان في الملف، يكفي الضغط على Ctrl + Q ومن ثم الرقم الموافق.

Ctrl + K + n	يُعلم موضع المشيرة الحالي ويعطيه الرقم n، وهذا الرقم يجب أن يكون بين 0 و 9.
Ctrl + Q + n	ينقل المشيرة إلى الموقع ذي الرقم n.

١-١٧- استخدام قوائم بيئة التطوير المتكاملة IDE:

حتى نختار أو نلغي أي قائمة من قوائم IDE علينا الضغط على Alt، مع أول حرف من حروف اسم القائمة. مثلاً للوصول إلى قائمة المساعدة Help يكفي الضغط على Alt + H. تملك القوائم عدداً من الأوامر، تقسم هذه الأوامر إلى ثلاثة أنواع هي:

١- أوامر لا تتطلب معلومات إضافية حتى تنفذ، أي تنفذ مباشرة بعد طلبها مثل الأمر New من القائمة File.

٢- أوامر تتطلب معلومات إضافية تأخذها من خلال صندوق حوار مثال ذلك الأمر Open ضمن قائمة الملفات File.

٣- أوامر تستدعي قوائم فرعية أخرى وكمثال عليها الأمر Environment ضمن قائمة الخيارات Options.

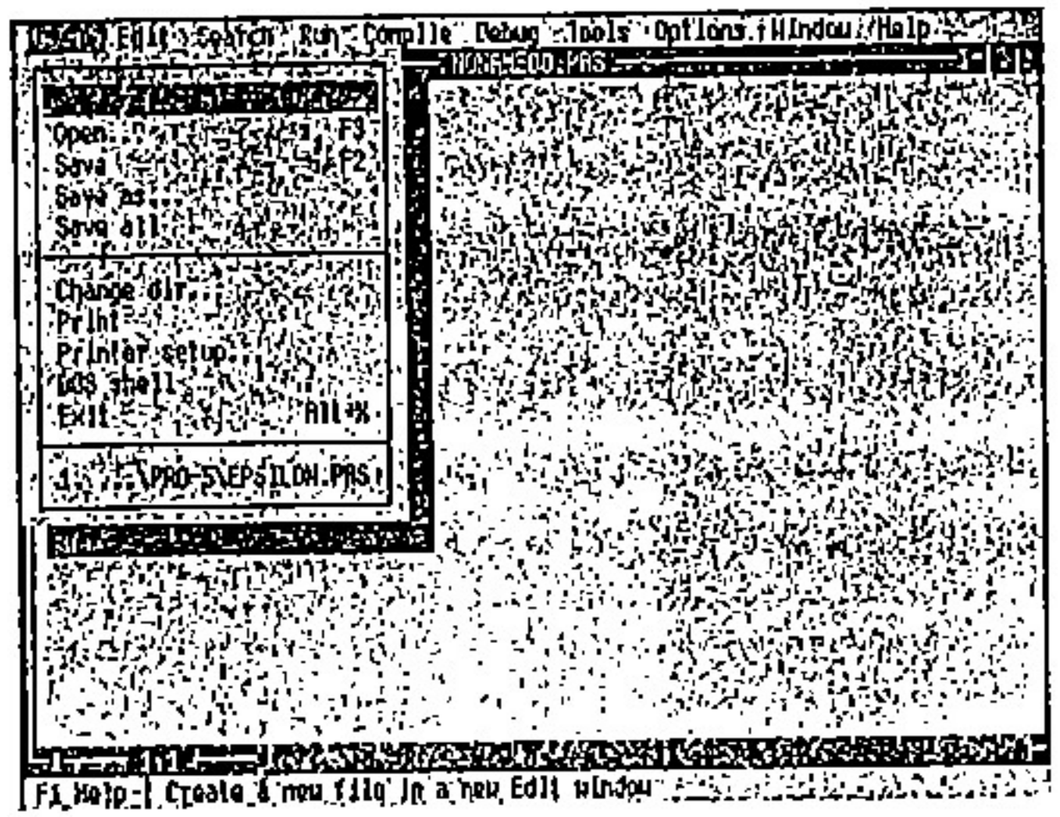
تملك بعض الأوامر مفاتيح مرتبطة بها تقوم بانتقاء الأمر مباشرة، دون العودة إلى القوائم، مثل المفتاح F3 الخاص بالأمر Open.

يمكن الحصول على معلومات حول أمر ما من خلال قراءة سطر الحالة أثناء الانتقال إلى الأمر أو من خلال طلب المساعدة أثناء ذلك بالضغط على F1 ، وفيما يلي شرح لأوامر كل قائمة من هذه القوائم.

١-١٧-١- قائمة الملف:

تحتوي قائمة الملف على أوامر لإنشاء وفتح وإغلاق وحفظ وحفظ باسم وطباعة الملفات، وأيضاً على إمكانية تغيير الدليل الحالي والخروج إلى شاشة دوس.

ومن أجل اختيار قائمة الملف نضغط على المفتاحين Alt + F، عندها ستبدو شاشة باسكال شبيهة بالشاشة التالية:



نعرض فيما يلي لأوامر هذه القائمة:

١- New (Alt + N):

يستخدم هذا الأمر لفتح ملف جديد ضمن نافذة التحرير.

٢- Open (Alt + O أو F3):

يُستخدم هذا الأمر لفتح ملف موجود وتحميله في نافذة التحرير، وهـذا الملـف يتم تحديده مـن خلال صندوق حوار. حيث يكتب اسم الملف ضمـن صندوق الحوار أو يكتب الامتداد Pas.* فقط لرؤية كل الملفات التي تحمل هذا الامتداد عندئذ يمكننا اختيار أحد هـذه الملفات بواسطة مفاتيح الأسهم، والضغط على Enter بعد اختيار الملف المطلوب.

إذا لـم يكـن الملف الذي تريد فتحه موجود فإن تربـو باسكال سوف

تنشئ ملفاً جديداً، تعطيه هذا الاسم.

٣- Save (Alt + F + S أو F2):

يستخدم هذا الأمر مـن أجل تخزين الملف الحالي، وسيتم التخزين تحت نفـس اسم الملف الحالي. أما إذا كان الملف جديداً عندها سيطلب تربو باسكال منك وضع اسم لهذه الملفات قبل تخزينها.

تجدر الإشارة هنا إلى أنـه عندما نقوم بتحرير ملف موجود فإن النسخة الأصلية لهذا الملف تخزن وفق الامتداد (.Bok).

٤- Save as (Alt + F + A):

يُخزن الملف الموجود في النافـذة تحت اسم جديد، يطلب تربو باسكال منك تحديـد الاسم الجديد مـن خلال صندوق حوار يظهر عند اختيارك لأمر Save as.

٥- Save all (Alt + F + L):

يُخـزن هذا الأمر محتويات كل الملفات التي طرأ عليها تعديل بعد آخر مرة خُزِّنت فيها. أي يمكن أن نكون قد فتحنا أكثر من ملف وعدلنا فيها ثم لكن لا نقوم بحفظ كل واحد منها على حدة. فإنه بهذا الأمر يمكن تخزين كل الملفات المعدلة دفعة واحدة.

٦- Change Dir... (Alt + F + C):

يمكننا هذا الأمر مـن تغيير الدليل الحالي الذي نحن عليه إلى أي دليل آخر، حيث أن الأمر Open سوف يفتح ملفات الدليل الحالي الجديد.

٧- Print (Alt + F + P):

يقوم هذا الأمر بطباعة محتويات النافذة وبشكل كامل.

٨- Printer Setup... (Alt + F + R):

يتم التحكم مـــن خلال هذا الأمـــر ببعض خواص الطباعـــة، مثلاً طلب طباعة الكلمات المحجوزة وفق لون أكثر سواداً عـــن باقي الكلمـــات وذلك عبر الخيار PRNFLTR.

٩- Dos Shell (Alt + F + D):

يُستخدم هذا الأمر للخروج المؤقت مـــن بيئة تربـــو باسكال IDE إلى النظام DOS. ويتم العودة إلى بيئة تربـــو باسكال IDE بكتابة Exit عند إشارة الجاهزية والضغط على Enter.

١٠- Exit (Alt + F + X أو Alt + X):

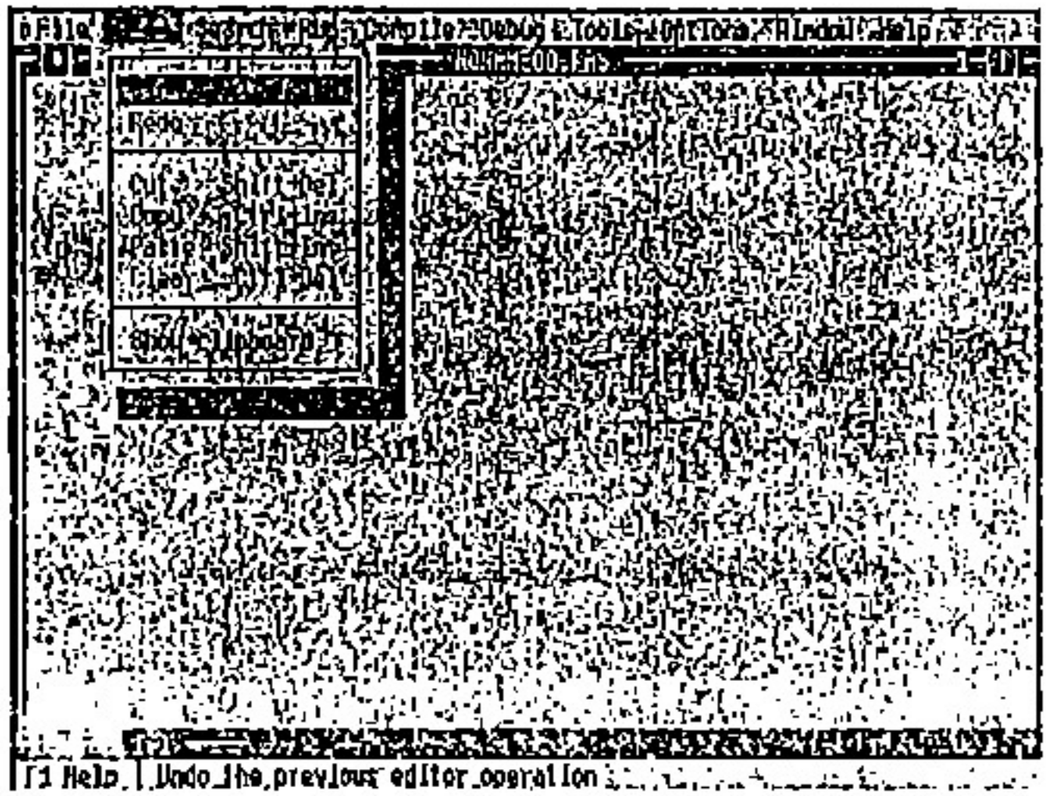
ينهي العمل في بيئة تربو باسكال IDE ويعيدنا إلى النظام الذي انطلقنـــا منه.

١١- يوجد في أسفل هذه القائمة عدداً من المسارات وأسماء الملفات مرقمة من 1 وحتى 5. وهذه الملفات ما هي إلا ملفات قد استخدمناها وأغلقناها. ومن أجل تحميل ملف منها يكفي اختياره بالضغط على مفتاح Enter. حيث يتـــم الاحتفاظ بآخر خمس ملفات في هذه اللائحة.

١-١٧-٢- قائمة التحرير:

تسمح أوامر قائمـــة التحرير بالقيام بعمليـــات على النصوص مثل نقل النصوص بين الملفات، أو نسخ فقرات من النصوص أو قص فقرات منها.

إن نافذة التحرير تبدو بالشكل التالي:



تُعلم النصوص بعدة طرق. في البداية ننقل المشيرة إلى بداية النصوص التي نريد تعليمها. وهذه الطرق هي:

أ- عن طريق لوحة المفاتيح، نضغط Shift ونبقيه مضغوطاً حتى ننقل المشيرة إلى نهاية كتلة النصوص التي نريد تحديدها، باستخدام مفاتيح الأسهم.

ب- الضغط على Ctrl + K + B لتحديد بداية كتلة النصوص، ثم ننقل المشيرة إلى نهاية كتلة النصوص ونضغط Ctrl + K + K. أمـــا إذا أردنـــا تحديد كلمة واحدة فيكفي الضغط على Ctrl + K + T.

ج- يمكن استخدام الفأرة، ووضع المؤشر فوق النص الذي نريـــد ونضغط على الـــزر اليســـاري، ونسحب المؤشر إلى نهاية كتلة النصوص التي نريد تحديدها ومن ثم نحرر الزر اليساري.

وأوامر هذه القائمة هي:

١- Undo (ALT + E + U أو ALT + Backs Pace):

التراجع عن آخر عملية تحرير.

٢- Redo (ALT + E + R):

استعادة المعلومات الملغاة بآخر عملية إلغاء تمت باستخدام الأمر Undo، أي يعيد الأمر إلى ما كان عليه قبل استعمال الأمر Undo.

٣- Cut (ALT + E + T أو Shift + Del):

يقتطع كتلة النصوص المحددة من النافذة الحالية، وينسخها إلى الحافظة. حيث يمكن نقلها إلى عدة نوافذ.

٤- Copy (ALT + E + C أو Ctrl + Ins):

ينسخ هذا الأمر كتلة النصوص المحددة إلى الحافظة، حيث يمكن استخدامها في عدة نوافذ.

٥- Paste (ALT + E + P أو Shift + Ins):

يقوم هذا الأمر بلصق النصوص المحددة الموجودة في الحافظة، إلى موضع المشيرة في النافذة الفعالة.

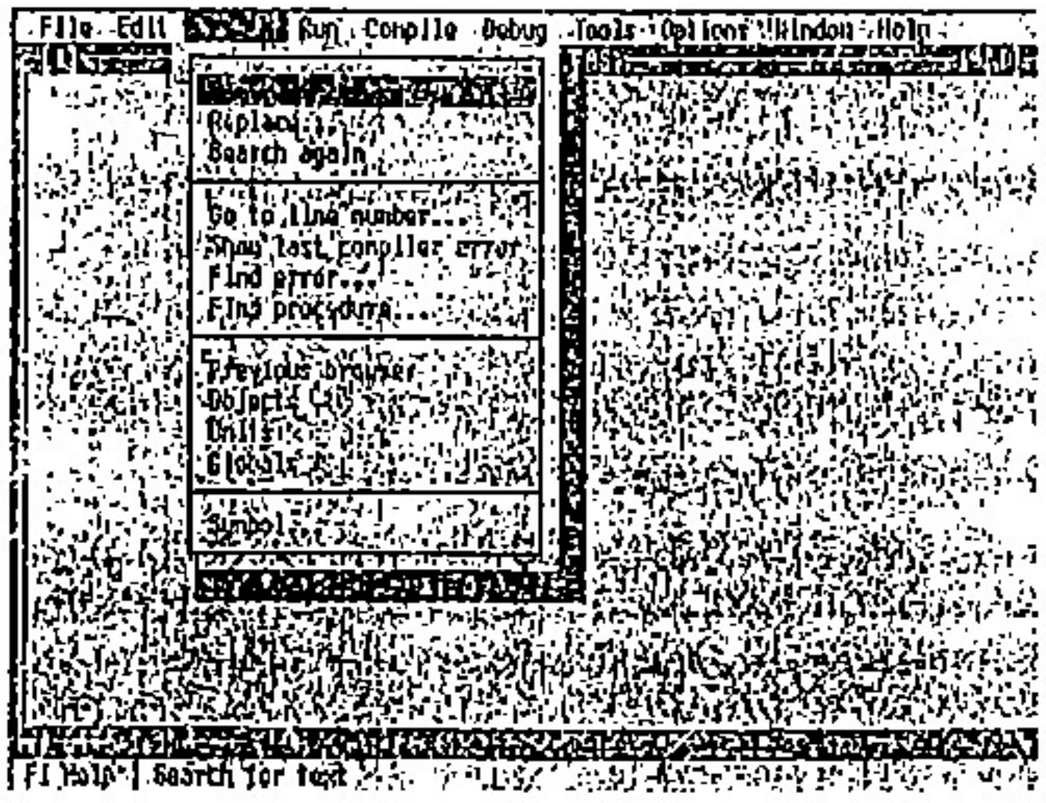
٦- Clear (ALT + E + L أو Ctrl + Del):

يحذف هذا الأمر كتلة النصوص المحددة ضمن النافذة الفعالة لكن لا ينسخها إلى الحافظة.

١-١٧-٣- قائمة البحث:

تتضمن قائمة البحث أوامر البحث عن النصوص أو الانتقال إلى موقع محدد ضمن الملف.

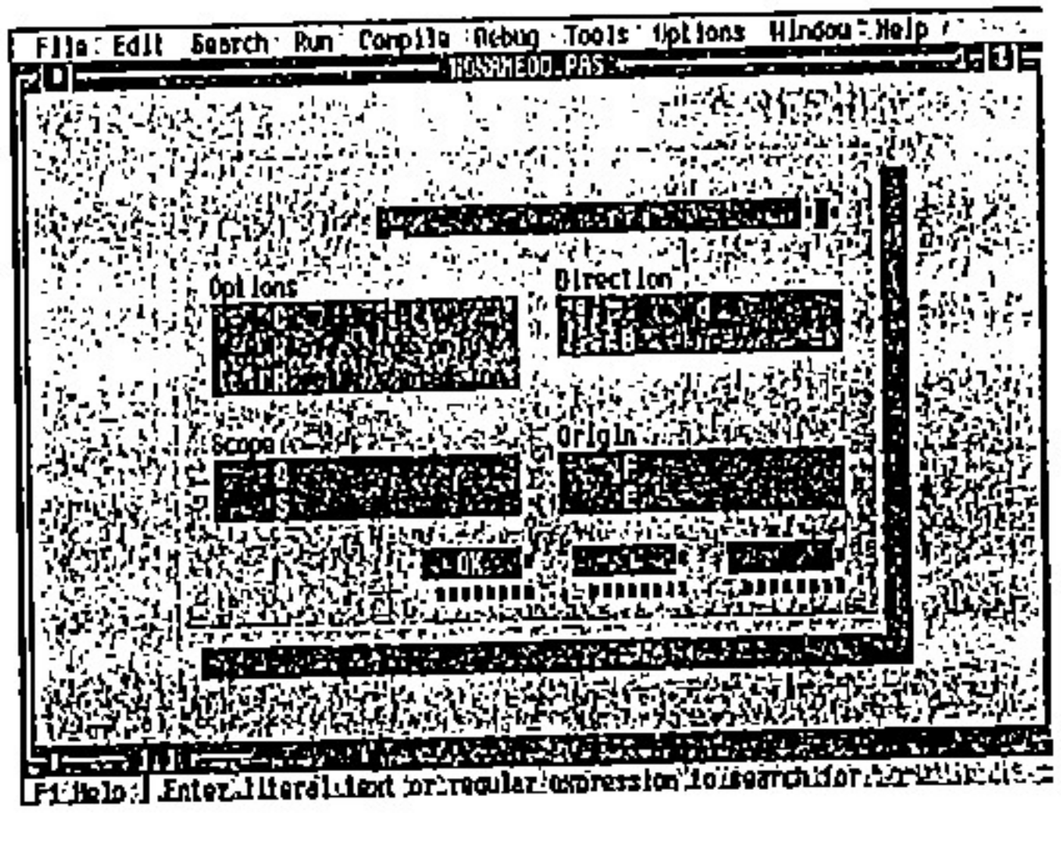
إن نافذة البحث تبدو بالشكل التالي:



أوامر هذه القائمة هي:

١- Find ... (Alt + S + F):

يبحث هذا الأمر عن نص معين، وتحدد خيارات هذا الأمر من خلال صندوق حوار البحث التالي:



يمكن البدء بعملية البحث من عدة مواضع هي:

- الموقع الحالي للمشيرة.
- بداية الملف.
- نهاية الملف.
- بداية كتلة نصوص محددة.

٢- Replace ... (Alt + S):

يُستخدم هذا الأمر للبحث عن نص محدد واستبداله بنص آخر.

٣- Search Again (Alt + S + S أو Ctrl + L):

يكرر آخر عملية بحث أو عملية استبدال.

٤- Goto Line Number ... (Alt + S + G):

ينقل المشيرة إلى سطر يتم تحديد رقمه.

٥- Show Last Compiler Error (Alt + S + C):

يظهر هـذا الأمر آخر خطأ عثر المترجـم عليـه، وينقل المشيرة إلى موقع ذلك الخطأ.

٦- Find Error (Alt + S + E):

نستطيع البحث عـن موضع خطأ التنفيذ باستخدام هذا الأمر، وذلك من خلال تحديد قيمة المقطع والإزاحة لهذا الخطأ.

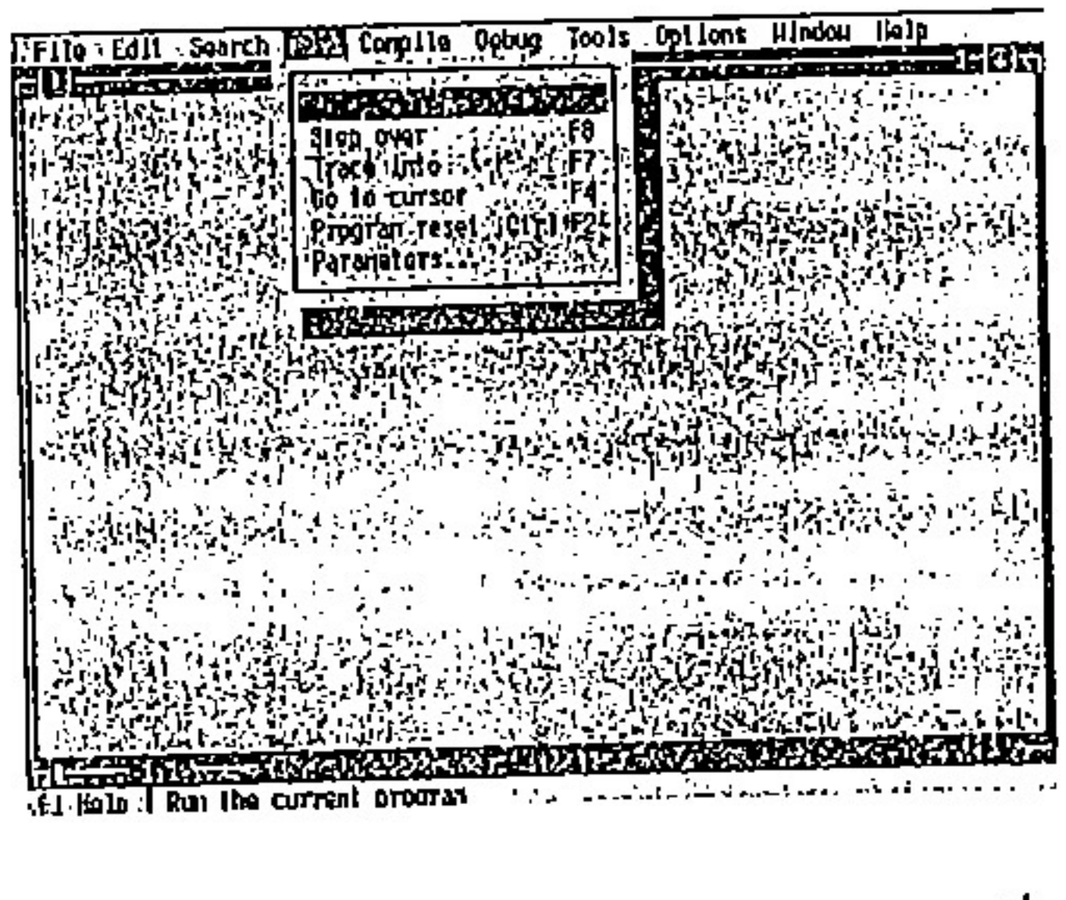
٧- Find Procedure (Alt + S + F):

يُستخدم هذا الأمر للبحث عن تعريف الإجراء أو التابع الذي نحدد، وهذا الأمر يسمح به فقط أثناء التنقيح.

١-١٧-٤- قائمة التنفيذ:

حتى نصل إلى قائمـة التنفيذ نضغط على Alt + R أو بالضغـط على الكلمة Run في شريط القوائم. تمكننا هـذه القائمة من ترجمة البرنامج وتنفيذه.

إن نافذة التنفيذ تبدو كما في الشكل التالي:



أوامر هذه القائمة هي:

١- Run (Alt + R + R أو Ctrl + F9):

الأمـــر Run يقوم بتنفيذ البرنامج وترجمة قبل ذلك إذا ما احتاج الأمر إلى ذلك أي إذا لم يكن مترجماً.

٢- Step Over (Alt + R + S أو F8):

يتم تتبع تنفيذ تعليمات البرنامج بشكل مشابه لأمر التتبع Trace.

٣- Trace (Alt + R + T أو F7):

يقوم هذا الأمر بتنفيـــذ تعليمـــات البرنامج خطوة خطوة، عند تنفيذ هذا الأمر فإن المشـــيرة ســـوف تنتقل إلى التعليمـــة التالية الواجب تنفيذهـــا، وتنفذ التعليمـــة الحاليـــة. وفي حال كون التعليمة التالية اســـتدعاء برنامج جزئي فإن المشيرة سوف تنتقل إلى بداية البرنامج الجزئي الواجب تنفيذه.

٤- Goto Cursor (Alt + R + G أو F4):

بتنفيذ هـــذا الأمر سيتوقف التنفيذ مؤقتاً في مكان توقف المشيرة. أي أن البرنامج يجري تنفيذه إلى أن يصل إلى موقع المشيرة فيتوقف التنفيذ.

٥- Program Reset (Alt + R + P أو Ctrl + F2):

ينهي هذا الأمـــر عملية التنفيح، محرراً أي مساحـــات مخصصة مـــن الذاكرة. ومغلقاً الملف المفتوح لهذه العملية.

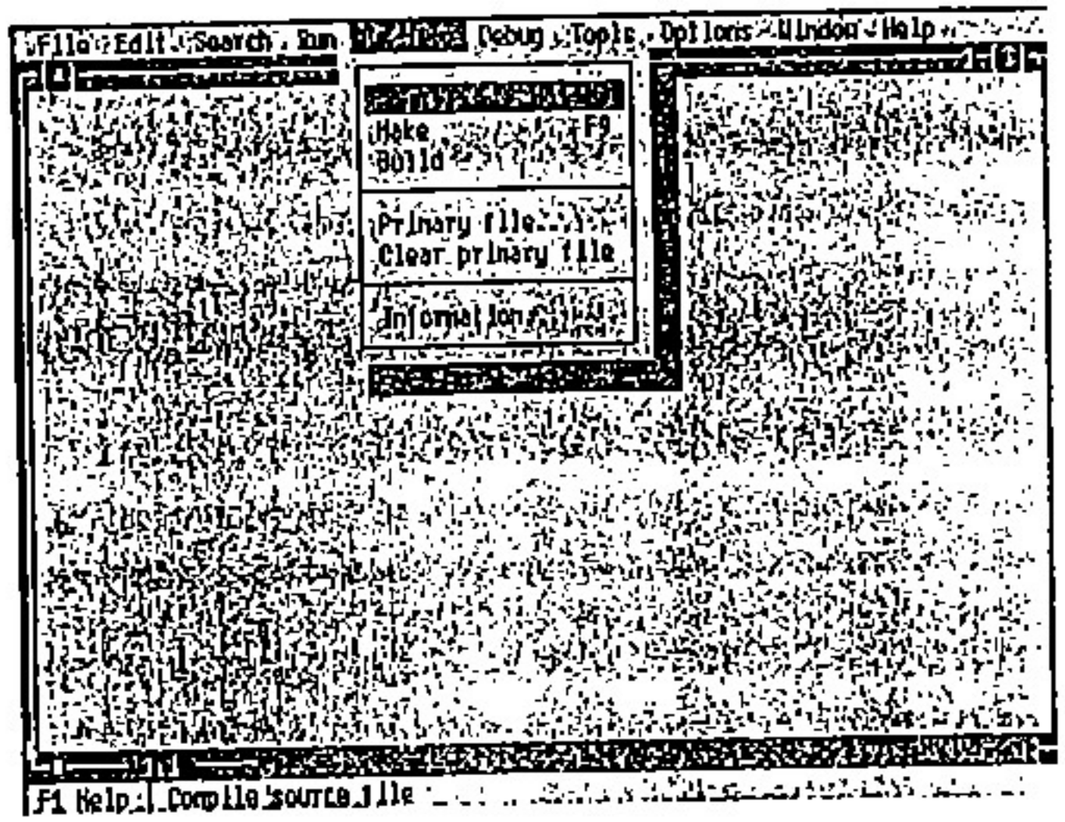
٦- Parameters (Alt + R + A):

يحدد هـــذا الأمر متحولات وسيطة للأوامر السطرية قبل تنفيذ البرنامج من خلال بيئة تربو باسكال IDE أي يمكن تمرير قيم أو متحولات إلى البرنامج في بداية التنفيذ وضمن بيئة IDE.

١-١٧-٥- قائمة الترجمة:

وظيفة أوامر هذه القائمة هي ترجمة البرامج فقط، يتم اختيار هذه القائمة إما بالضغط عليها في شريط القوائم أو بالضغط على المفتاحين Alt + C.

إن نافذة قائمة الترجمة هي بالشكل التالي:



أوامر هذه القائمة هي:

١- Compile (Alt + C + C أو Alt + F9):

يُستخدم هـذا الأمر لترجمة الملف الموجود ضمن النافذة الفعالة، وأثناء عملية التنفيذ يظهر صندوق حوار يعرض عدداً من المعلومات عن إنجاز عملية الترجمة.

أن الملف الناتج بعد عمليـة الترجمة يُكتب على القرص و يصبح مـن النوع التنفيذي EXE أو يكتب في الذاكرة حسب قيمة الأمر Destination وإذا ما وجد المترجم خطأ فسوف تتوقف عمليـة الترجمة ويُضاء السطر الذي سبب الخطأ وتظهر رسالة في أعلى النافذة فيها رقم الخطأ وشرح عنه.

٢- Make (Alt + C + M أو F9):

ينشئ هـذا الأمر برنامجاً مترجماً مـن خلال ترجمة كل الملفات التي يتطلبها هذا البرنامج والتي طرأ عليها تعديل منذ آخر عملية ترجمة للملفات التي يعتمد عليها هذا الملف في حين أن الأمر Compile يترجم فقط الملف في النافذة الفعالة.

٣- Build (Alt + C + B):

يشبه الأمر Make إلا أنـه يعيد ترجمـة كل الملفـات التي يحتاجهـا البرنامج، عُدلت أم لم تُعدل.

٤- Destination (Alt+ C + D):

يُستخدم هـذا الأمر لتحديد فيما إذا كانت وجهة البرنامج ستكون إما في القرص أو في الذاكرة.

٥- Primary File ... (Alt + C + P):

يُسـتخدم هذا الأمر لتحديد الملف الرئيسـي، الذي سيتم ترجمتـه عند استدعاء الأمر Make أو الأمر Build.

٦- Clear Primary File:

يلغي تخصيص الملف الذي حدد كملف أساسي أو أولي وبذلك تتم عملية الترجمة ابتداءً مـن الملف في النافذة الفعالة.

٧- Information (Alt + C + I):

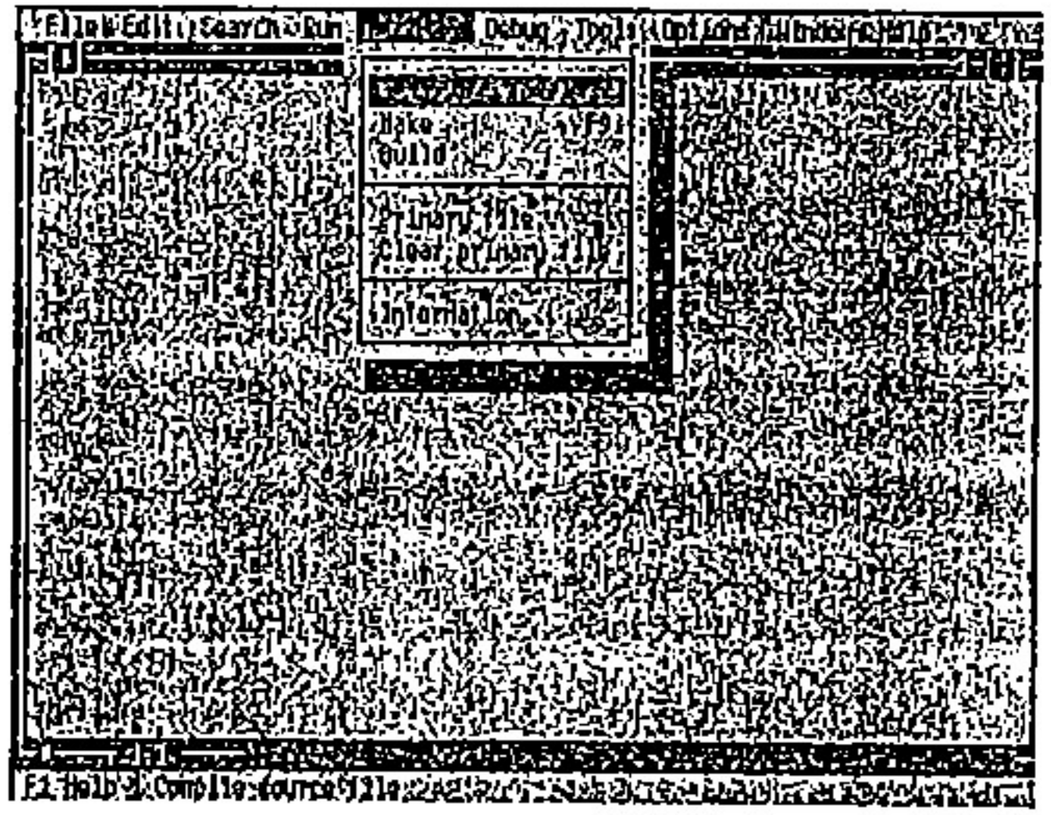
يظهر المعلومات حول الملف الحالي، والبيئة الخاصة به.

١-١٧-٦- قائمة التنقيح:

تزودنا بيئة التطوير المتكاملة لتربو باسكال، إمكانية تنقيح وفحص لتنفيذ البرنامج، حيث يمكن إضافة نقاط توقف ضمن عبارات البرنامج ليتوقف تنفيذه

مؤقتاً عندها، نستطيع فحص المتحولات وقيمتهـا عند نقـاط التوقف تلك، نفتح قائمة التنقيح بالضغط على Alt + D أو على القائمة Debug في شريط القوائم.

إن نافذة قائمة الترجمة هي بالشكل التالي:



وأوامر هذه القائمة هي:

١- Break Points (Alt + D + B):

هـذا الأمر يفتح صندوق الحـوار الخاص بنقـاط التوقف، حيث يمكن إضافة أو حذف أو تحرير نقاط التوقف تلك، من خلال هذا الصندوق.

٢- Call Stack (Alt + D + C أو Ctrl + F3):

يفـتح هـذا الأمر نافذة تحتوي على قائمة مـن الإجرائيات والتوابـع المستدعاة والتي توقف تنفيذها مؤقتاً، ليتم تنفيذ روتين آخر.

٣- Register (Alt + D + R):

يظهر هذا الأمر محتويات مسجلات الحاسب ضمن نافذة مستقلة.

٤- Watch (Alt + D + W):

يفتح نافذة مراقبـة تظهر فيها قيـم المتحولات المخصصـة كمتحولات مراقبة أو قيم تعابير معينة.

٥- Out Put (Alt + D + O):

يفتح نافذة يظهر فيها خرج البرنامج.

٦- User Screen (Alt + D + U أو Alt + F5):

يسمح هذا الأمر بمشاهدة خرج البرنامج ضمن نافذة تملأ الشاشة كاملة.

٧- Evaluate / Modify (Alt + D + E أو Ctrl + F4):

يُستخدم هـذا الأمر لتحديد متحول أو تعبير لحساب قيمته، حيث يمكن تغيير قيمة هذا المتحول واستئناف تنفيذ البرنامج اعتماداً على هذه القيمة. ويمكن استخدام الأمر Evaluate كآلة حاسبة صغيرة.

٨- Add Watch ... (Alt + D + A أو Ctrl + F7):

يسـتخدم هذا الأمـر لإضافة متحولات لتراقب ضمـن نافذة المراقبة، بتحديد اسم المتحول المراد مراقبته.

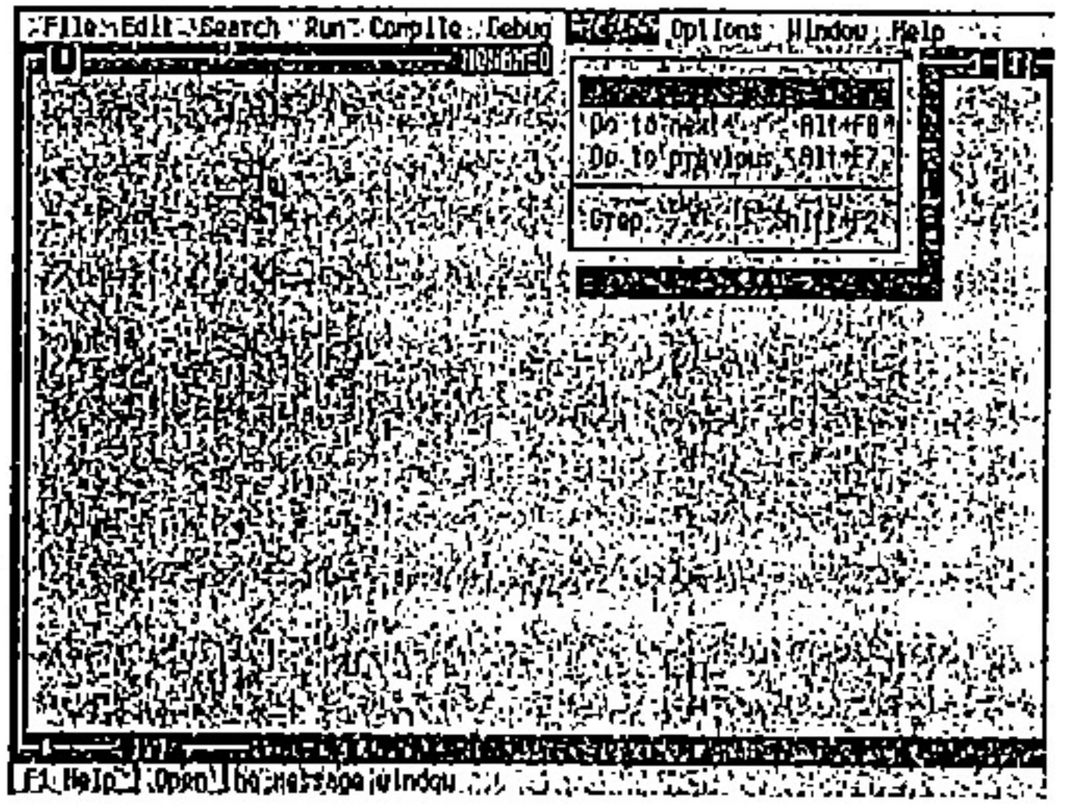
٩- Add Break Point (Alt + D + P أو Ctrl + F8:

يُستخدم هذا الأمر لتحديد نقاط توقف، حيث ينتقل هذا الأمـر بين تحديد نقطة توقف، أو إلغاء هذه النقطة.

١-١٧-٧- قائمة الأدوات:

في الإصدار السابع من تربو باسكال تم إضافة هذه القائمة والتي تُزودنا بإمكانية الوصول إلى عدة برامج وسيطة، لاستخدامها في عملنا، ومن خلال بيئة IDE مثلاً يمكننا تنفيذ البرنامج لتحديد جزء البرنامج الذي يستهلك معظم زمن تنفيذ البرنامج.

إن نافذة قائمة الأدوات لها الشكل التالي:



وأوامر هذه القائمة هي:

١- Messages (Alt + T + M):

يفتح هذا الأمر نافذة الرسائل.

٢- Goto Next (Alt +T + N أو Alt + F8):

إذا احتوت نافذة الرسائل على عدة رسائل فإن هذا الأمر ينقلنا إلى الرسالة التالية.

٣- Goto Previous (Alt + T + P أو Alt + F7):

إذا احتوت نافذة الرسائل على عدة رسائل فإن هذا الأمر ينقلنا إلى الرسالة السابقة.

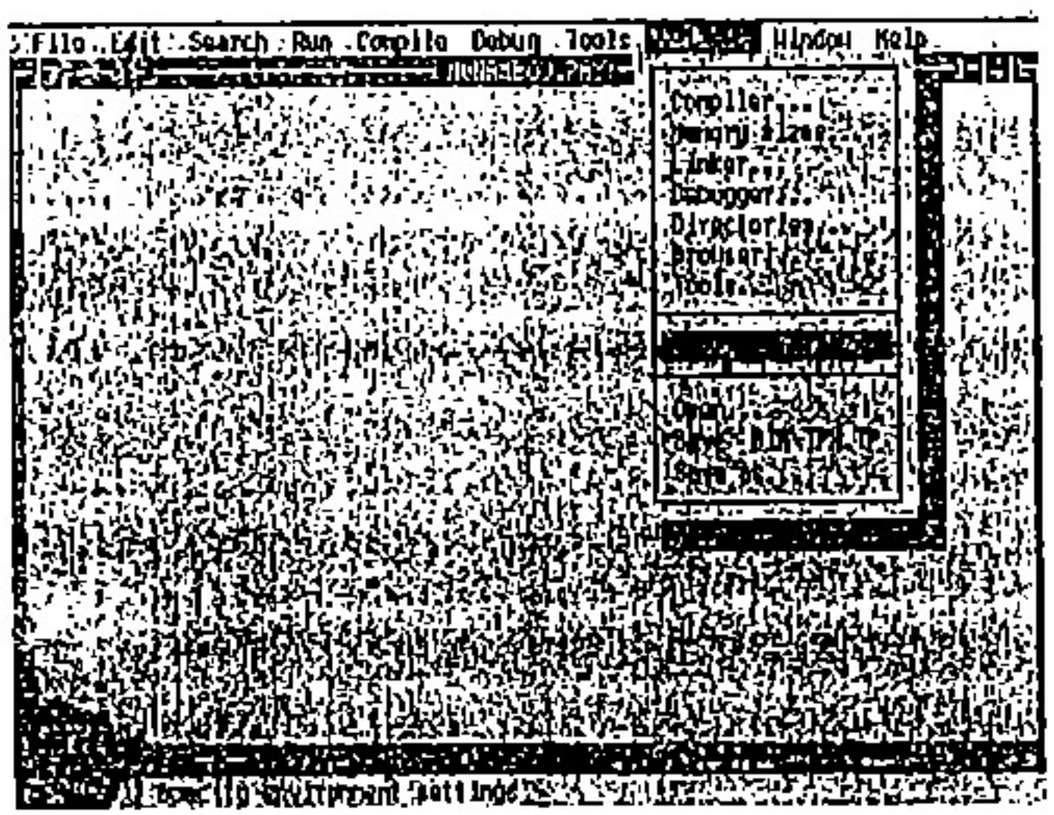
٤- Grep (Alt + T + G أو Shift + F2):

يبحث هذا الأمر عن سلسلة رمزية معينة أو تعبير نظامي ضمن ملف أو عدة ملفات يأخذ الشكل:

Grep [خيارات] سلسلة رمزية [ملفات]

١-١٧-٨- قائمة الخيارات:

يتم تعديل بيئة العمل هذه بواسطة أوامر هذه القائمة ويتم حفظ هذه التغيرات. نضغط على هذه القائمة من شريط القوائم لفتحها أو نضغط المفتاحين Alt + O.



وأوامر هذه القائمة هي:

١- Compiler (Alt + O + C):

من خلال صندوق الحوار الذي يفتحه هـذا الأمر تتحكم في سير عملية الترجمة.

٢- Memory Sizes (Alt + O + M):

يحدد هذا الأمر حجم المكدس والكومة المخصصان للبرنامج.

٣- Linker (Alt + O + L):

يتحكم هذا الأمر بسلوك عملية ربط البرنامج.

٤- Debugger (Alt + O + B):

هذا الأمر يمكننا من التحكم بسلوك المنقح ضمن بيئة التطوير المتكاملة IDE: حيث يمكننا تحديد فيما إذا كنا نريد التبديل بين شاشة الخرج ونافذة الملف الأصلي أثناء عملية التنقيح، إذ يحدد الخيار Smart أن عملية التبديل تتم فقط إذا كُتب شيء في شاشة الخرج.

٥- Directories (Alt + O + D):

يستخدم هذا الأمر لتحديد أدلة معينة، يمكن أن يوجد فيها عدة أنواع من الملفات.

٦- Tools (Alt + O + T):

يقـوم هـذا الأمر بعمليات تحديد لصندوق الأدوات Tool Bar وإضافة برامج جديدة له، وتحديد خواص كل أداة مـن هـذه الأدوات أو إضافة برامج جديدة إلى صندوق الأدوات.

٧- Environment → (Alt + O + O):

يقوم هـذا الأمر بتجهيز قائمة فرعية من الأوامر، وهـذه الأوامر تقوم بتهيئة وتشكيل بيئة العمل وجعلها اكثر فعالية وإثارة.

٨- Open (Alt +O + O):

يقوم هـذا الأمر بإخبار تربـو باسكال بقراءة معلومات تهيئة البيئة من الملف المخصص لذلك.

٩- Save (Alt + O + S):

يقوم هـذا الأمر بتخزين القيم التي حددت كخيارات. وهـذا الأمر مهم عندما نريد العمل مع عدة خيارات لبيئة تربو باسكال.

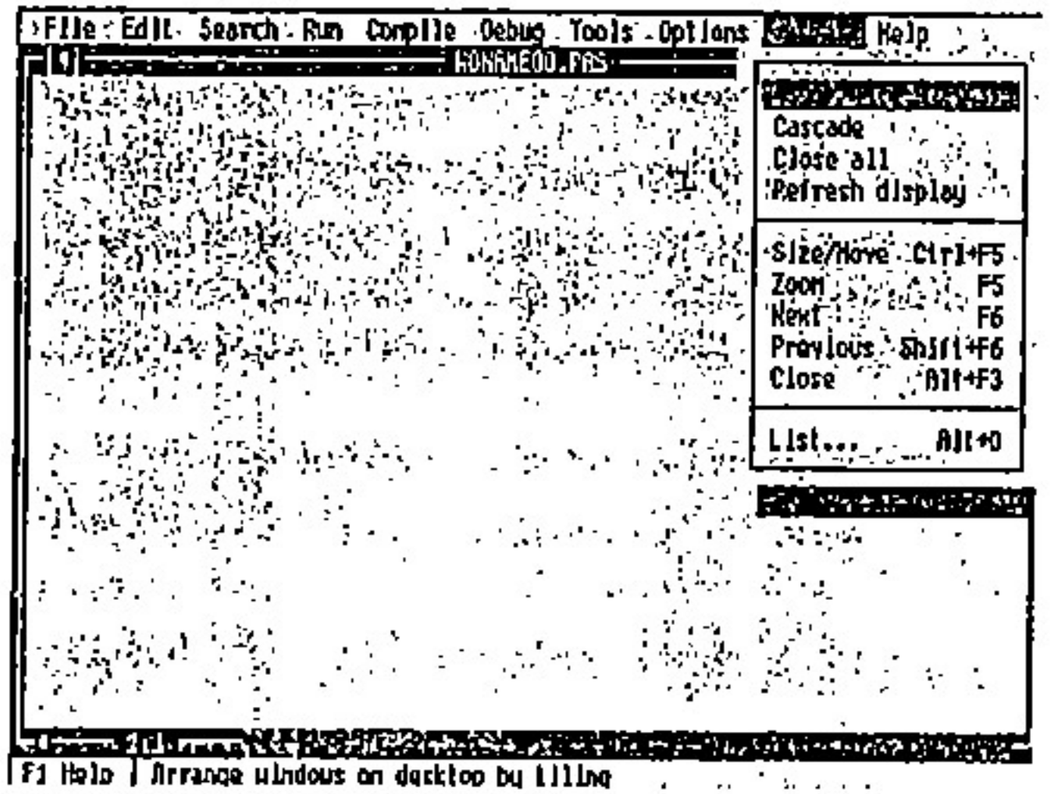
١٠- Save as ... (ALT + O + S):

يحفظ هذا الأمر القيم التي حددت ضمن ملف جديد.

١-١٧-٩- قائمة النافذة:

تستخدم أوامر هذه القائمة من أجل التحكم بطريقة ترتيب وإظهار النوافذ على الشاشة، نضغط على القائمة في شريط القوائم بزر الفأرة الأيسر أو نضغط المفتاحين ALT + W.

إن قائمة هذه النافذة سوف تبدو كما في الشكل التالي:



أوامر هذه القائمة هي التالية:

١- (ALT + W + T) Tile:

يقوم هـذا الأمر بوضع كل النوافذ المفتوحة بحيث تراها جميعها حيث تشغل كل واحدة منها جزءاً صغيراً من الشاشة، وتكون النافذة الفعالة هي النافذة المضاءة.

٢- (ALT + W + A) Cascade:

يقوم هـذا الأمـر برصف النوافذ المفتوحة خلف بعضها البعض حيث يظهر عنوانها فقط.

٣- (Alt +W + O) Close All:

يغلق هذا الأمر كل النوافذ المفتوحة.

٤- (Alt + W + R) Refresh Display:

يعيد هذا الأمر رسم الشاشة من جديد.

٥- (Alt + W + S أو Ctrl + F5) Size / Move:

يستخدم هذا الأمر لتحريك النافذة الفعالة وتغيير حجمها.

٦- (Alt + W + Z أو F5) Zoom:

يجعل النافذة تملأ الشاشة بأكملها أو يعيدها إلى حجمها الأصلي.

٧- (Alt + W + N أو F6) Next:

يهيئ هذا الأمر النافذة التالية في قائمة النوافذ المفتوحة.

٨- (Alt + W + P أو Shift + F6) Previous:

يهيئ هذا الأمر النافذة السابقة للنافذة الحالية في قائمة النوافذ الدائرة.

٩- (Alt + W + L أو Alt + F3) Close:

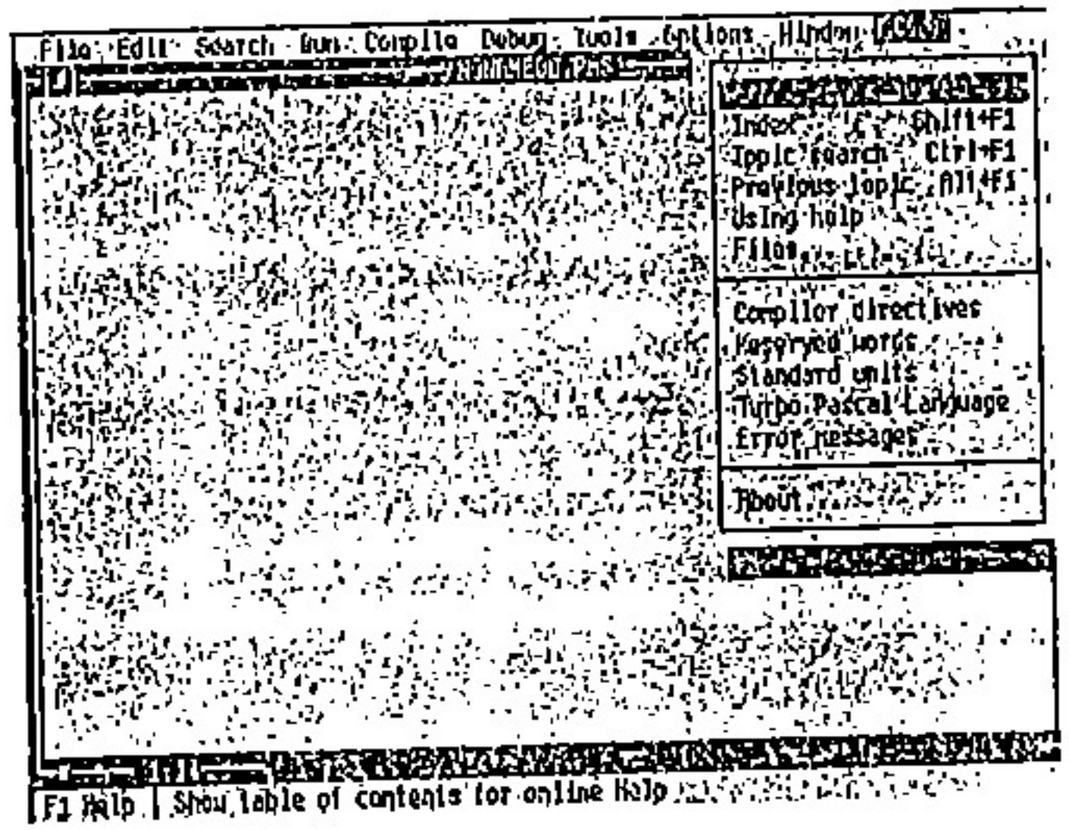
يغلق هذا الأمر النافذة الحالية.

١٠- (Alt + W + C أو Alt + O) List ...:

يعطينا هـذا الأمر لائحة بأسـماء الملفات الموجـودة ضمن النوافـذ المفتوحة، حيث يمكن اختيار ملف من هذه الملفات لتقوم البيئة بتهيئة نافذته.

١-١٧-١٠- قائمة المساعدة:

تساعد هذه القائمة في الوصول إلى فهرس المحتويات، نفتح هذه القائمة بالضغط عليهـا في شريـط القوائـم بزر الفأرة الأيســر أو بضغط المفتاحين ALT + H كما في الشكل التالي:



وأوامر هذه القائمة هي:

١- Contents (ALT + H + C):

هــذا الأمر يعطينا جدول بالمحتويــات التي تستطيع المســاعدة Help الوصول إليها.

٢- Index (ALT + H + I أو Shift + F1):

يعطينا فهرس بالمصطلحات التي تستطيع المساعدة الوصول إليها.

٣- Topic Search (ALT + H + T أو Ctrl + F1):

إذا كانت المشيرة واقفــة على مصطلح فإن هــذا الأمر يعطينا مساعدة حول هذا المصطلح وإلا فإن هذا الأمر سوف يظهر فهرس الموضوعات إذا لم تكن المشيرة واقفة على مصطلح.

٤- Previous Topic (ALT + H + P أو ALT + F1):

يعيد إظهار آخر شاشة مساعدة تم إظهارها قبل الشاشة الحالية.

٥- Using Help (ALT + H + H):

يعطينا مساعدة حول إمكانيات المساعدة نفسها.

٦- Files ... (ALT + H + F):

يسمح هذا الأمر بتأسيس أو تركيب ملفات مساعدة إضافية.

٧- Compiler Directives (ALT + H + D):

يعطيــنا هذا الأمر معلومــات حــول توجيهات المترجم الخاصة بتربو باسكال.

٨- Reserved Words (ALT + H + R):

يعطينا هذا الأمر معلومات حول الكلمات المحجوزة في تربو باسكال.

٩- Standard Units (ALT + H + U):

يعطينا هذا الأمر معلومات حول الوحدات المتضمنة في المكتبة التنفيذية لتربو باسكال.

١٠- Turbo Pascal Language (ALT + H + L):

يعطينا هذا الأمر معلومات حول تعريفات لغة تربو باسكال.

١١- Error Messages (ALT + H + E):

يعطينا هذا الأمر مساعدة حول رسائل الخطأ التي يمكن أن نصادفها.

١٢- About (ALT + H + A):

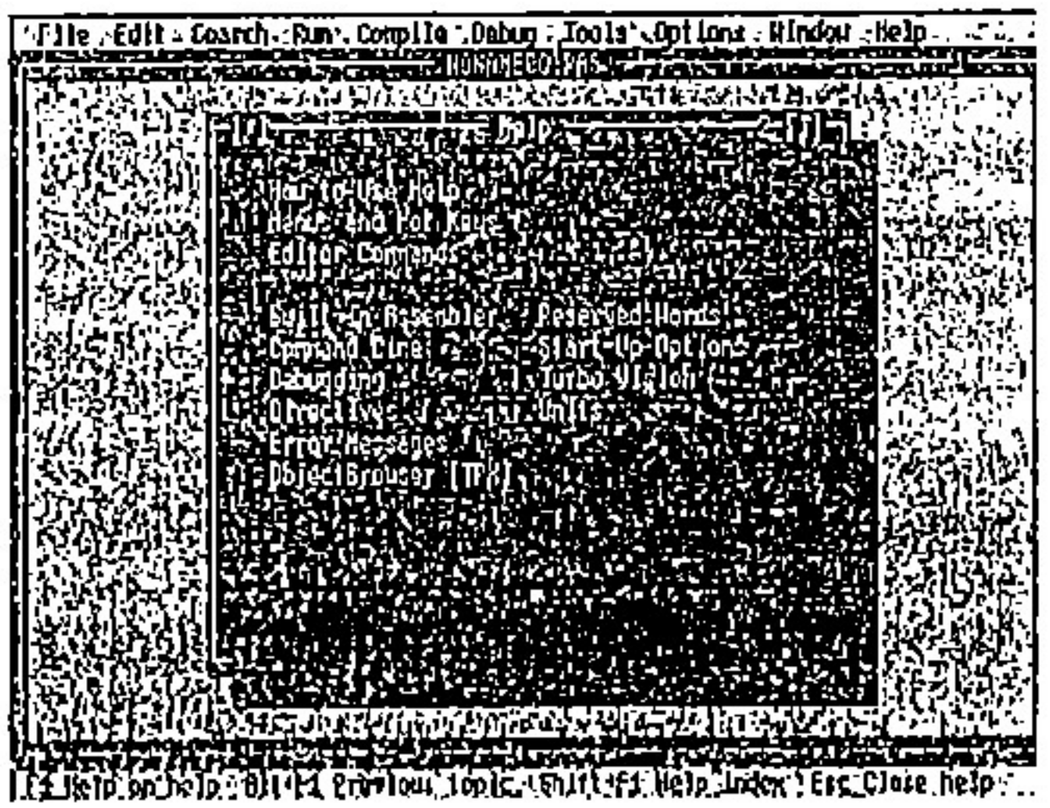
يعطينا معلومات حول إصدار لغة تربو باسكال الذي نعمل فيه.

بالإضافة لما تقدم من أوامر المساعدة هناك المفتاح F1 الذي يعطينا مساعدة حول العمل الحالي الذي نعمل فيه. مثلاً إذا كانت المشيرة في سطر القوائم فإن الضغط على المفتاح F1 سيعطينا مساعدة حول القائمة التي تقف عندها أما إذا كنا نكتب برنامجاً ما وكانت المشيرة متوضعة فوق كلمة فإن الضغط على المفتاح F1 سوف يعطينا معلومات حول هذه الكلمة وهكذا.

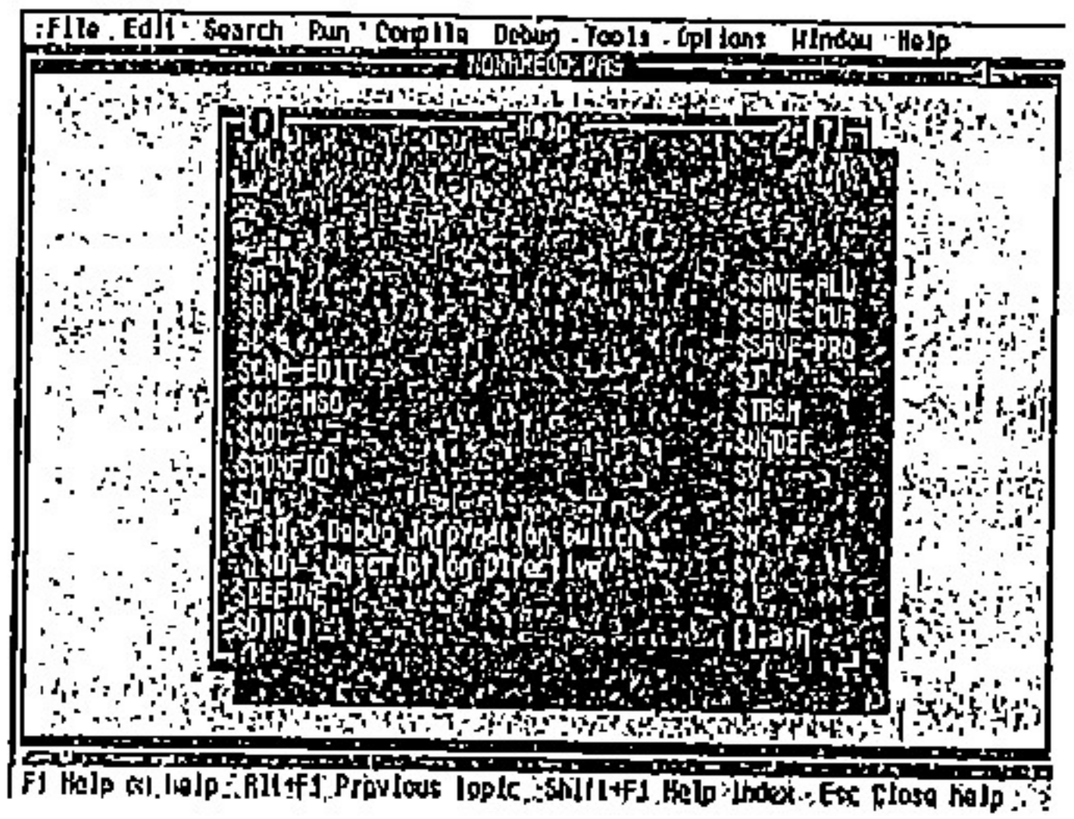
ملاحظة:

يمكن الاستعانة بقائمة المساعدة من أجل الحصول على معلومات عن أوامر اللغة وكيفية استخدام هذه الأوامر.

على سبيل المثال ننقر بزر الفأرة الأيسر على كلمة Contents لظهور لوح المحتويات كما في الشكل التالي:

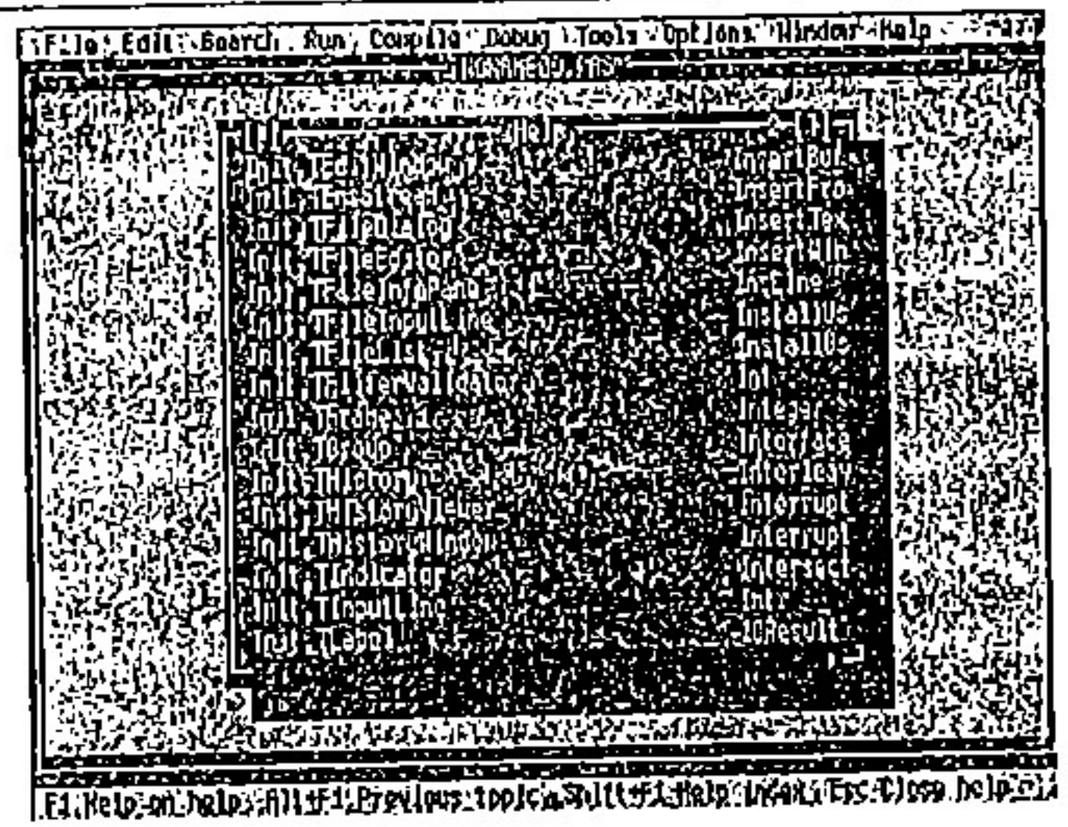


يمكن النقر بزر الفأره الأيسر على كلمة Index لظهور دليل أوامر اللغة كما في الشكل:

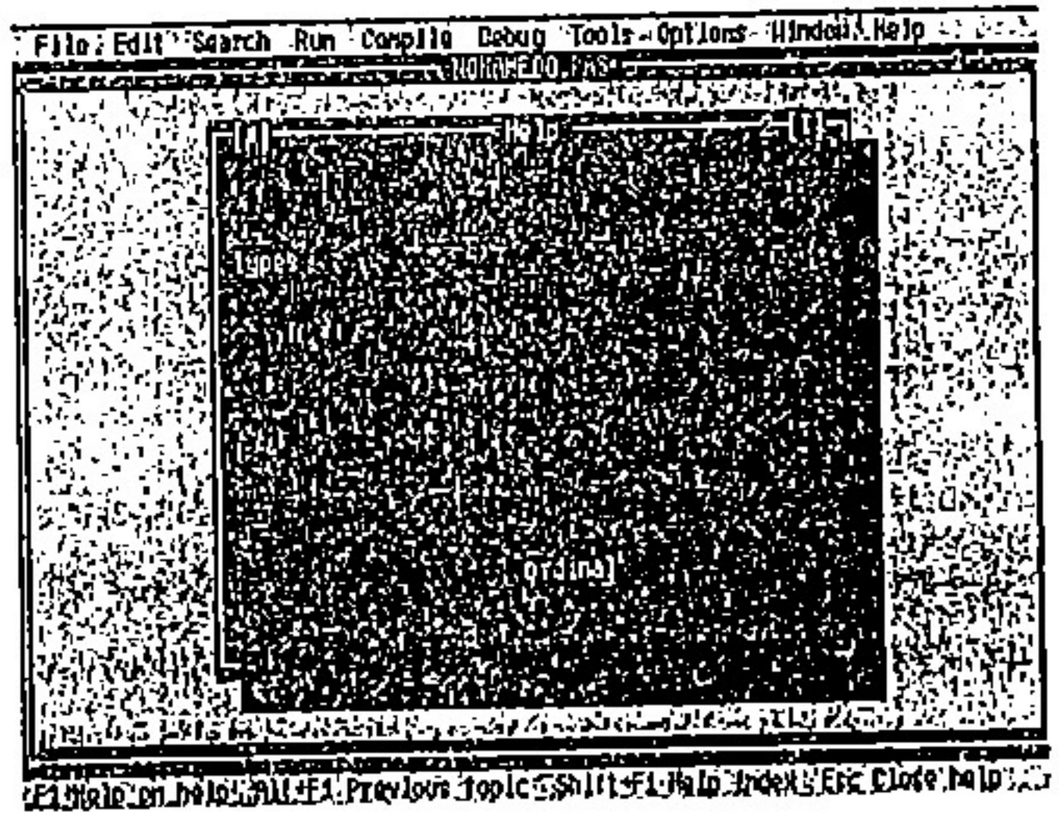


نقوم بتمرير شريط التحريك العمودي للوصول إلى أي كلمة محجوزة والنقر فوقها للحصول على معلومات عن هذه الكلمة وشروط وكيفية استخدامها، أو نقوم بكتابة الكلمة من لوحة المفاتيح لنصل إلى موقع الكلمة على نافذة المساعدة.

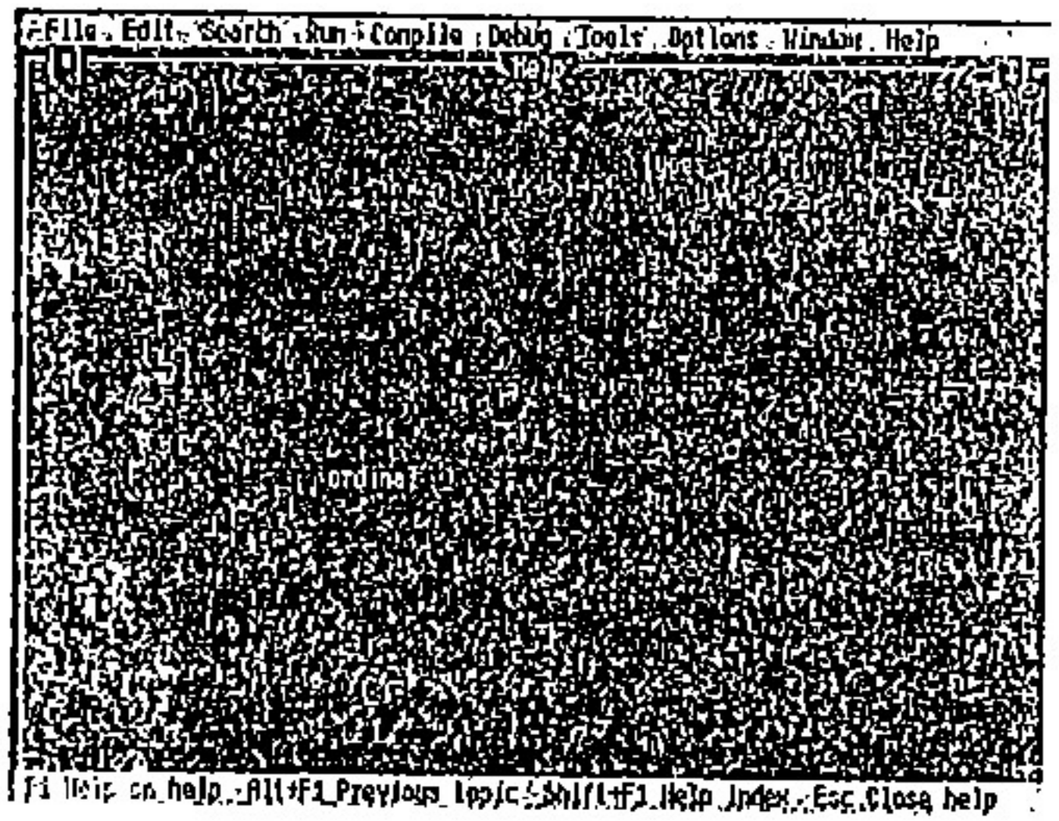
على سبيل المثال نكتب كلمة Integer ، لنصل إلى النافذة التالية كما في الشكل:



ننقر بزر الفأره على كلمة Integer نجد نافذه المساعدة تحوي معلومات عن نمط الأعداد الصحيحة Integer كما في الشكل التالي:



نكبر نافذة المعلومات بالنقر فوق السهم العلوي في الجهة العلوية اليمنى من النافذة السابقة لنحصل على النافذة بالشكل التالي:



وهو المطلوب.

١-١٨- الكلمات المحجوزة RESERVED WORDS:

يملك تربو باسكال مجموعة من الكلمات المحجوزة والتي لا يمكن استخدامها كمميزات أو كأسماء للمتحولات الاختيارية وذلك لأن مترجم تربو يتعامل معها وفق تعريفها ولا يمكننا استخدامها إلا في موضعها كأمر من أوامر اللغة،حيث لا أهمية لحالة الأحرف هل هي صغيرة أم كبيرة،لأن مترجم اللغة لا يميز بين حالة الأحرف.

الكلمات المحجوزة هي:

and	goto	program
asm	if	record
array	implementation	repeat
begin	in	set
case	inherited	shl
const	inline	shr
constructor	interface	string
destructor	label	then
div	library	to
do	mod	type
downto	nil	unit
else	not	until
end	object	uses
exports	of	var
file	or	while
for	packed	with
function	procedure	xor

تمارين عامة

١- لدينا مجموعة من المتحولات كما في الشكل:

```
Var
    A,B,C,D:integer;
Begin
    Read(A);Readln(B,C);Read(D);
```

ولدينا معطيات الدخل بالشكل:

```
25  72
46  55
18  4
```

والمطلوب: اكتب القيم التي تأخذها المتحولات A,B,C,D .

٢- اكتب قيم كل من المتحولات A و B و C في كل خطوة من خطوات البرنامج التالي:

A B C

```
Program trace ;
Var
     A,B,C,N :integer ;
Begin
    N:=4;
    A:=5;
    B:=A*N;
    C:=A mod N;
    A:=C;
    B:=A+C;
   Writeln(A,B,C);
End.
```

الفصل الثاني

الخوارزمية ALGORITHM

٢-١- الخوارزمية ALGORITHM:

في البداية وقبل التطرق إلى أوامر تربو باسكال لا بد من تعريف الخوارزمية والتي ستكون الحل البرمجي للمسألة الرياضية حيث سنقوم بوضع خوارزمية الحل ثم نكتب البرنامج الموافق لها.

يعد مفهوم الخوارزمية من أوسع المفاهيم في الرياضيات المعاصرة وجاءت كلمة خوارزمية نسبة إلى العالم الإسلامي محمد بن موسى الخوارزمي وهو رياضي من القرن التاسع الميلادي ومبتكر علم الجبر والمقابلة وقد عبرت هذه الكلمة سابقاً في أوربا عن الحساب بالنظام العربي وذلك وفق المفاهيم التي وضعها الخوارزمي في علم الجبر.

إن الخوارزمية هي طريقة الحل الدقيق والمفصل وذات الخطوات المتسلسلة لمسألة معينة. إن خوارزمية حل المسألة تعني العمليات اللازمة للحل وفق التسلسل المطلوب لتنفيذ هذه العمليات وبشكل يؤدي إلى حل المسألة.

٢-٢- خوارزمية حساب تابع رياضي:

لتكن لدينا المسألة التالية: بفرض x عدداً صحيحاً والمطلوب إيجاد قيمة التابع

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

إن حل هذه المسألة سهل ويمكن بسهولة إجراء العمليات الحسابية اللازمة لذلك، لكن هل يمكن اعتبار هذه العلاقة خوارزمية يمكن بواسطتها

حساب قيمة التابع Y من أجل قيمة معطاة للمتحول x. طبعاً الجواب هو لا لأن الحـل لا يعطى هنا على شكل متسلسل لتنفيذ العمليات وسيكون تنفيذ الحل هذا اختـيارياً، مثلاً يمكن أن نحسب البسط أولاً ثم المقام ثانياً ويمكن أن يتم العكس وفـي كلتا الحالتين نحصل على النتيجة ذاتها، وهـذا يعني أن العلاقة تترك لنا حرية اختيار تسلسل عمليات الحل، وبالتالي فهي لا تعين خوارزمية.

إن مفهوم الخوارزمية يتطلب تحديد تسلسـل تنفيذ العمليات إضافة إلى العمليات اللازمة للحل. وحتى تصبح العلاقة السابقة خوارزميـة يجب إضافـة بعض الظروف إليها، بحيث تحدد هذه الشروط التسلسل المتتالي لتنفيذ العمليات اللازمة للحصول على قيمة التابع Y انطلاقاً من قيمة للمتحول x.

مما تقـدم نكتب قائمـة عمليات هـذه الخوارزمية وفق هـذا المفهوم الخوارزمي:

١- عمليـة تحديد قيمة المتحول x والتي سوف ندعوهـا عملية إدخـال قيمة للمتحول x وتكتب بالشكل إقرأ x.

٢- عملية حساب البسط والتي سوف تحسب بالشكل:

A : = 3 * x + 7

نرمز لعملية الضرب بالإشارة *.

٣- عملية حساب المقام، تتم بالشكل التالي:

B : = 5 – 2 + x

٤- عملية تقسيم البسط على المقام تأخذ الشكل التالي:

Y : = A / B

نرمز لعملية القسمة بالإشارة / حيث يمكن كتابة البسط والمقام على سـطر

واحد

٥- عملية إخراج النتيجة وتكتب على الشكل:

اكتب Y

مما تقـدم يمكن كتابـة قائمة العمليـات هـذه وبشـكل متتال لتشكيل الخوارزمية المطلوبة بالشكل التالي:

١) إقرأ x

٢) A : = 3 * x + 7

٣) B : = 5 – 2 + x

٤) Y : = A / B

٥) اكتب Y

إن قائمة العمليات السابقة تسمى خوارزمية حـل المسألة المعطاة. وإن هذه القائمة قد حددت جميع الخطوات اللازمـة لحل المسألـة وأعطت التسلسل اللازم.

يمكن شرح خطوات كل من العمليات السابقة وفق التالي:

- في العملية الأولى تمت عمليـة إدخال المتحول x وهي عملية إعطاء قيمـة محـددة للمتحول x. بحيث يصبح هـذا المتحول حاملاً لقيمة محددة ومن النوع المحدد بنص المسألة.

- فـي العملية الثانية عملية حساب البسط وتعني ضرب القيمة المدخلة للمتحول x في العملية الأولى بالعدد 3 ثم جمع العدد 7 إلى الناتج، ووضع الناتج النهائي الحاصل في متحول آخر هو A وبالتالي يعني أن قيمة A سـتكون هي قيمـة البسط بعد تنفيذ العملية الثانية.

- في العملية الثالثة عملية حساب المقام ووضع القيمة في B ليصبح B بعد تنفيذ العملية مساوياً لمقام العلاقة.

- وفي العملية الرابعة والتي هي عملية تقسيم البسط على المقام أي عملية تقسيم

قيمـــة المتحول A الناتجة من العملية الثانية على قيمـــة المتحول B الناتجة من العملية الثالثة وسيعطى الناتج إلى المتحول Y. وبالتالي قيمة هذا المتحول Y هو مساوي لقيمة العلاقة المطلوبة.

- أما في العملية الخامسة فسيتم إخراج النتيجة أي كتابة Y.

مثال:

اكتب برنامجاً يقوم بقراءة مجموعة من الأعداد الحقيقية عددها N ويقوم البرنامج بحساب العدد الأصغر والأكبر والمتوسط الحسابي لهذه الأعداد.

المعطيات:

عدد الأعداد وهو N حيث N أكبر من 0 والأعداد وهي Xi .

النتائج:

Xmin العدد الأصغر .

Xmax العدد الأكبر .

Xmean المتوسط الحسابي .

الخوارزمية:

١- قراءة عدد الأعداد N .

٢- إعطاء قيم إبتدائية للمتحولات Xmin، Xmax، Xmean .

٣- كرر N مرة :

قراءة عدد ومقارنته بالأعداد السابقة وجمعه لما سبقت قراءته.

٤- كتابة النتائج.

الخوارزمية بالتفصيل:

١- إقرأ N (عدد الأعداد)

٢- إقرأ X (العدد الأول)

٣- Xmin : = X

Xmax : = X

Xmean : = X

٤- I : = 1

٥- ما دام I أصغر من N كرر

أ- إقرأ X

ب- إذا X < Mmin نفذ

Xmin : = X

ج- إذا X > Mmax نفذ

Xmax : = X

د- Xmean : = Xmean + x

هـ- I : = I + 1

٦- اكتب

'Xmax = ' , Xmax

'Xmin = ' , Xmin

'Xmean = 'Xmean /N

وهو المطلوب.

٢-٣- مفهوم الخوارزمية:

نورد فيما يلي بعض الخصائص الأساسية للخوارزميات:

١- توزع العمليات المكونة للخوارزمية على خطوات منتهية تهدف كل خطوة منها إلى تنفيذ عمل معين.

٢- يجب أن يكون عمل كل خطوة بسيطاً وهذا يضمن التنفيذ الصحيح للخوارزمية.

٣- إن عبارة اختر شيئاً أو سم شيئاً من شيئين لا يمكن أن تدخل في عبارة خوارزمية لأن ذلك يعطي نتائج مختلفة عند التنفيذ من قبل منفذين مختلفين.

٤- تحدد الخوارزمية تسلسل تنفيذ العمليات البسيطة المكونة لها.

أمثلة:

مثال١:

المطلوب حل معادلة الدرجة الثانية

$Ax^2 + Bx + C = 0$

من أجل قيم معطاة للمتحولات A, B,C.

تحليل المسألة:

المعطيات هي معاملات المعادلة A, B,C.

النتائج:

هي جذور المعادلة.

طريقة الحل:

إذا كانت المعادلة من الدرجة الثانية،تحسب الجذور اعتمادا على قيمة المعين:

$D := B^2-4*A*C$

الخوارزمية:

١- إقرأ A, B,C

٢- إذا a = 0 نفذ:

أ- اكتب المعادلة ليست من الدرجة الثانية

ب- إذا b <> 0 نفذ:

اكتب الحل = -c/b

وإلا اكتب المعادلة غير صحيحة

٣- وإلا

أ- حساب المعين $D := B^2-4*A*C$

ب- حساب الجذور

إذا D = 0 نفذ:

- جذران متماثلان

- X := -b/2*a

-اكتب " للمعادلة جذران متماثلان =",x

وإلا إذا D > 0 نفذ:

- جذران حقيقيان

- RD := sqrt (D)

- x1 := (-b-RD)/(2*a)

x2 := (-b+RD)/(2*a)

- اكتب جذران حقيقيان

- اكتب "x1 =", x1, "x2 =", x2

وإلا

- جذران عقديان

- RD := sqrt (|D|)

Rx := -b/(2*a) –

Ry := RD/(2*a)

– اكتب جذران عقديان

– اكتب "x1 =", Rx, "+i", Ry

"x2 =", Rx, "-i", Ry

مثال٢:

المطلــوب كتابة خوارزمية، تقرأ درجة الحرارة وتطبع نوع الرياضة المــوافقة لدرجة الحرارة وفق الجدول التالي:

نوع الرياضة	درجة الحرارة
Swimming	Temp>50
Tennis	40<temp<=50
golf	30<temp<=40
skiing	10<temp<=30
Chinese checkers	Temp<=10

إن خوارزمية الحل هي بالشكل التالي:

١-قراءة درجة الحرارة.

٢-طباعة نوع الرياضة.

من أجل طباعة درجة الحرارة لابد من ملاحظة المجال التي تنتمي إليه:

٢-١- إذا كانت درجة الحرارة >50 نفذ:

اطبع "swimming"

٢-٢- إذا كانت درجة الحرارة <= و >= 50 نفذ:

اطبع "tennis"

٢-٣- إذا كانت درجة الحرارة <= و >= 40 نفذ:

اطبع "golf"

٢-٤- إذا كانت درجة الحرارة <= 30 و >= 10 نفذ:

اطبع "sking"

٢-٥- إذا كانت درجة الحرارة < 10 نفذ:

اطبع "Chinese chechers"

مثال٣:

المطلوب إيجاد جميع قواسم العدد الصحيح n ولتكن m حيث $n \neq m$.

المعطيات:

العدد الصحيح n .

النتائج:

هي قواسم العدد n ،التي هي m وهي عبارة عن أعداد صحيحة.

القاسم:

نقــول عن العدد m أنه قاســم للعدد n إذا كان باقي قســمة n على m يساوي الصفر أي باقي قسمة n على m =0 .وحاصل القسمة يكون d=n%m .

الخوارزمية:

١- إقرأ العدد n .

٢- إسناد القيمة 2 إلى المتحول m .

٣- مناقشة حالة m هل هي أصغر من n بالشكل:مادام m<n كرر:

٣-١- إسناد القيمة d إلى ناتج القسمةأي:d:=n%m .

٣-٢- إذا كانت قيمة المتحول d تساوي الصفر أى d=0 نفذ:

اكتب قيمة m التي هي قاسم للعد n .

٣-٣- أعط تزايد مقدارة 1 للعدد m .

نستطيع الآن كتابة الحل بالشكل التالي:

١- إقرأ n .

٢- m := 2 .

٣- مادام m < n كرر:

٣-١- d := n % m .

٣-٢- إذا d = 0 نفذ:

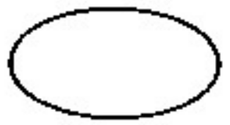
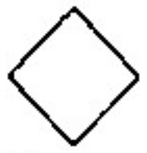
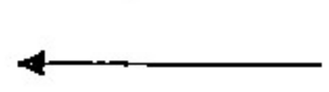
اكتب m، قاسم للعدد n .

٣-٣- m := m + 1 .

٢-٤- الطرائق البيانية:

تعتمد الطريقة البيانية لصياغة الخوارزميات على توضيح خطوات تنفيذ الخوارزمية باستخدام أشكال هندسية خاصة وأسهم تصل بينهما إضافة إلى عبارات باللغة الطبيعية وتعابير رياضية أو منطقية. وبذلك نحصل على ما يسمى بالمخطط التدفقي Flowchart Diagram.

إن الأسهم تفصل العمليات اللازمة لإنجاز العمل وتبين تسلسلها. ويستخدم المخطط التدفقي أشكالاً هندسية متفقاً عليها كما في الجدول التالي:

الشكل	الوصف
	الشكل البيضوي: لتحديد بداية الخوارزمية ونهايتها
	المستطيل: للعمليات التنفيذية العادية والحسابية وغيرها
	المعين: للعمليات التي ترتبط باختبار تحقق شرط ما وتتطلب قراراً منطقياً
	متوازي الأضلاع: لعمليات الإدخال والإخراج
	السهم: لاتجاه التنفيذ

٢-٥- البرمجة المهيكلة:

يمكن الاكتفاء بالتعليمات الأساسية للغة الخوارزميات من أجل توصيف الحل. أما في المسائل المعقدة والصعبة فلا بد من بناء الحل تدريجياً. ويُنصح بتطبيق مبادئ البرمجة المهيكلة Structured Programming.

تهدف البرمجة المهيكلة إلى كتابة برامج مقروءة ومفهومة. وتعتمد هذه البرمجة على استخدام ثلاثة أنواع من عمليات التحكم في مسار البرنامج:

١- التتالي.

٢- التفرع الشرطي.

٣- التكرار.

حيث يمكن تجزئة الحل إلى خطـوات تمثل الوظائف الأسـاسية وكل وظيفة معقدة بدورها تتألف من عدة خطوات جزئية وبهـذا الشكل يمكن تبسيط الحل حتى نصل إلى خطوات قابلة للتحقق.

اعـتماداً على مـا تقدم نكتب النقاط الرئيسية في منهجية حـل المسائل بالشكل التالي:

١- فهم المسألة وتحليلهـا بدقـة أي معرفة المدخلات المطـلوب معـالجتها والمخرجات المطلوب الحصول عليها.

٢- وضع تصور لحل المسـألـة، أي كتابة الحل بطريقـة التبسيط المتتـالي وباستخدام هياكل التعليمات الأساسية حين الحاجة.

٣- عند الوصول إلى أدنى مسـتوى نكون قد حصلنا على الخوارزمية كاملة وبلغة الخوارزمية.

اعتماداً على هذه الخوارزمية يجري الترميز باللغة المطلوبة ولتكن هنا Pascal.

الفصل الثالث

مبادئ أساسية في البرمجة بلغة تربو باسكال

٣-١- التعليمة الشرطية:

للتعليمة الشرطية حالتين الشرط والاختيار:

٣-١-١- الشرط:

- لغة الخوارزميات:

إذا A < 0 نفذ: < مجموعة تعليمات >

يجري هنا اختبار الشرط فإذا كان محققاً عندها تنفذ < مجموعة تعليمات > وإلا يتم الانتقال إلى التعليمة التالية في البرنامج.

- لغة تربو باسكال:

```
If A < 0 Then
  Begin
  ...
  End;
```

٣-١-٢- الاختيار:

- لغة الخوارزميات:

إذا A < 0 نفذ:

< مجموعة تعليمات ١ >

وإلا

< مجموعة تعليمات ٢ >

يجري هنا الاختيار بين طريقتي تنفيذ وذلك وفق شرط محدد، إذا تحقق الشرط عندها تنفذ < مجموعة تعليمات ١ > وإلا تنفذ < مجموعة تعليمات ٢ >.

- لغة تربو باسكال:

```
If A < 0 Then
  Begin
  ...
  End;
Else
  Begin
  ...
  End;
```

مثال:

اكتب برنامجاً لترتيب عددين تصاعدياً، يجري إدخال العددين عبر لوحة المفاتيح.

الحل:

- لغة الخوارزمية:

١- إقرأ العددين A, B

٢- إذا A > = B نفذ:

اكتب B, A

وإلا

اكتب A, B

- لغة تربو باسكال:

```
Program ArrangementNumbers;
var A,B:integer;
Begin
  writeln('input A,B');
  readln(A,B);
  if A>=B then
    writeln(B:8,A:8)
  else
    writeln(A:8,B:8);
readln
End.
```

٣-٢- تعليمة التكرار:

يتم تكرار تنفيذ مجموعة من التعليمات ما دام الشرط محققاً أي ما دامت الصيغة المنطقية للشرط صحيحة.

- لغة الخوارزميات:

مادام I < = N كرر

< مجموعة تعليمات >

يجري في هذه الخوارزمية اختبار الشرط فإذا كان محققاً تنفذ < مجموعة تعليمات > ويكرر الأمر ويُختبر الشرط من جديد وتنفذ التعليمات ما دام الشرط محققاً وعندما يصبح الشرط غير محقق تنتقل إلى التعليمة التالية.

ملاحظة:

يجب أن تتضمن < مجموعة تعليمات > عمليات تساهم في تغيير متحولات الشرط وذلك للخروج من الحلقة وإلا بقي الشرط صحيحاً دائماً وتم تكرار < مجموعة تعليمات > بشكل دائم ودون توقف.

- لغة تربو باسكال:

```
While I < = N do
  Begin
  ...
  End;
```

أمثلة:

مثال ١:

احسب مجموع الأعداد 1, 2, 3, ... , n أي المطلوب حساب المجموع:

$1 + 2 + 3 + 4 + ... + n$.

الحل:

- لغة الخوارزمية:

نفرض قيمة ابتدائية للمجموع S ولتكن 0 ونستخدم متحول عداد I لعد عدد مرات الجمع وبالتالي الخوارزمية تأخذ الشكل:

١- إقرأ M

٢- I : = 0

S : = 0

٣- ما دام I < = M كرر

أ- S : = S + I

ب- I : = I + 1

٤- اكتب S

- لغة تربو باسكال:

```
PROGRAM SUM_NUMBERS;
VAR I,S,M:INTEGER;
BEGIN
   WRITELN('ENTER NUMBER ');
   READLN(M);
   I:=0;S:=0;
  WHILE I<=M DO
     BEGIN
        S:=S+I;
        I:=I+1
     END;
  WRITELN('S=',S);
READLN
END.
```

مثال ٢:

اكتب برنامجاً لحساب مربعات الأعداد من 5 وحتى 40. بشرط في الخرج أن يكون جدول الأعداد ومربعاتها مسبوقاً بترويسة.

الحل:

- لغة الخوارزمية:

١- اكتب " قيمة x "، " قيمة y "

٢- x : = 5

٣- ما دام x < = 40 كرر

أ- y : =x * x

ب- اكتب x، " "، y

ج- x : = x + 1

- لغة تربو باسكال:

```
PROGRAM SQR_NUMBERS;
VAR
  X,Y:INTEGER;
BEGIN
  WRITELN('X VALUE','  ','Y VALUE');
  X:=5;
  WHILE X<=40 DO
   BEGIN
    Y:=X*X;
```

نتيجة تنفيذ البرنامج:

```
X VALUE  Y VALUE
   5        25
   6        36
   7        49
   8        64
   9        81
  10        100
```



```
  11        121
  12        144
  13        169
  14        196
  15        225
  16        256
  17        289
  18        324
  19        361
  20        400
  21        441
  22        484
  23        529
  24        576
  25        625
  26        676
  27        729
  28        784
  29        841
  30        900
  31        961
  32        1024
  33        1089
  34        1156
  35        1225
  36        1296
  37        1369
  38        1444
  39        1521
  40        1600
```

٣-٣- أمثلة عامة:

مثال ١:

أوجد القاسم المشترك الأعظم لعددين صحيحين موجبين.

الحل:

لنسم هذا القاسم GCD اختصاراً للكلمات Greater Common Divider يمكـن إيجاد القواسم بأن نوجد جميع القواسم لكل مـن العددين ونختار القواسم المشتركة فيكون GCD هو أكبر. طبعاً الواحد هـو قاسم مشترك لكل الأعداد. لكن طريقة الحل هذه طويلة. توجد طريقة أسهل تنسب إلى اليوناني إقليدس تقوم فكرة الحل على طرح العدد الأصغر من العـدد الأكبر وجعل حاصـل الطرح مكان الأكبر حتى تصبح القيمتان متساويتان فتكون قيمة التساوي هذه هي قيمة القاسم المشترك الأعظم GCD.

مثلاً:

8	5	
3	5	3 = 8 - 5
3	2	2 = 5 - 3
1	2	
1	1	

ويكون القاسم لـ 8 و 5 هو 1

وأيضاً مثلاً

15	9
6	9
6	3
3	3

ويكون القاسم لـ 15 و 9 هو 3

اعتماداً على ما تقدم نكتب خوارزمية الحل.

- لغة الخوارزمية:

١- إقرأ A، B

٢- ما دام A لا يساوي B كرر

أ- إذا A > B نفذ

A : = A - B

ب- وإلا B : = B - A

٣- GCD : = A

٤- اكتب القاسم المشترك الأعظم GCD.

- لغة تربو باسكال:

```
Program find_gcd;
(*find greater common divider gcd of A,B*)
Var
    A,B:integer;
    GCD:integer;
Begin
   readln(A,B);
```

```
  While(A<>B) do
   begin
    if(A>b)then
        A:=A-B
    else
        B:=B-A;
   end;
  GCD:=A;
  writeln('Greater Common Divider=',GCD);
readln
End.
```

مثال ٢:

اكتب برنامجاً يقوم بقراءة مجموعة من الأعداد الحقيقية عددها N ويقوم البرنامج بحساب العدد الأصغر والأكبر والمتوسط الحسابي لهذه الأعداد.

الحل:

- لغة الخوارزمية:

١- إقرأ N (عدد الأعداد)

٢- إقرأ X (العدد الأول)

٣- Xmin : = X

Xmax : = X

Xmean : = X

٤- I : = 1

٥- ما دام I أصغر من N كرر

أ- إقرأ X

ب- إذا X < Mmin نفذ

Xmin : = X

ج- إذا X > Mman نفذ

Xmax : = X

د- Xmean : = Xmean + x

هـ- I : = I + 1

٦- اكتب

'Xmax = ' , Xmax

'Xmin = ' , Xmin

'Xmean = 'Xmean /N

- لغة تربو باسكال:

```
Program MinMaxMean(input,output);
(*Find Min,Max and Mean of input numbers *)
Var
    N,i:integer;
    X,Xmin,Xmax,Xmean:real;
Begin
 Writeln('Enter The Value of N');
 readln(N);
 Writeln('Enter The numbers');
 readln(X);
 Xmin:=X;Xmax:=X;Xmean:=X;
 i:=1;
 While(i<N) do
   begin
```

```
    readln(X);
    if(X<Xmin)then
      Xmin:=X;
    if(X>Xmax)then
      Xmax:=X;
    Xmean:=Xmean+X;
    i:=i+1;
  end;
writeln('Max=',Xmax:8:2);
writeln('Min=',Xmin:8:2);
writeln('Mean=',Xmean/N:8:2);
readln
End.
```

مثال ٣:

اكتب برنامجاً لإيجاد المضـــاعف المشترك الأصغـــر لعددين صحيحين موجبين.

الحل:

يعتمـــد الحل على المثال ١ حيث أن المضـــاعف المشترك الأصغر هو عبارة عن ناتج قسمة جداء العددين على القاسم المشترك الأعظم للعددين وذلك لأن:

جداء العددين = القاسم المشترك الأعظم × المضاعف المشترك الأصغر.

- لغة تربو باسكال:


```
Program find_scm;
(*find small common multiple scm of A,B*)
Var A,B,C:integer;
   GCD.integer;SCM:real;
Begin
 writeln('input tow integer numbers');
 readln(A,B);
 C:=A*B:
   While(A<>B) do
  begin
   if(A>b)then
     A:=A-B
   else
     B:=B-A;
 end;
GCD:=A;
SCM:=C/GCD;
writeln('Small Common Multiple=',SCM:8:2);
readln
End.
```

مثال ٤:

اكتب برنامجاً لحساب المتوسط الحسابي لعدد غير محدد من الأعداد.

الحل:

- لغة الخوارزمية:

١- إقرأ N (عدد الأعداد).

٢- Sum : = 0 ، Average : = 0

٣- I : = 1

٤- مادام I < = N كرر

أ- إقرأ Number

ب- Sum : = Sum + Number

ج- I : I + 1

٥- Average : = Sum / N

٦- اكتب Average

- لغة تربو باسكال:

```
PROGRAM AVERAGE_NUMBERS;
VAR
    I,N:INTEGER;
    AVERAGE,NUMBER,SUM:REAL;
BEGIN
   WRITELN('INPUT N');
   READLN(N);
   SUM:=0;AVERAGE:=0;I:=1;
   WHILE I<=N DO
    BEGIN
     WRITELN('INPUT NUMBER');
     READLN(NUMBER);
     WRITELN('THE VALUE OF NUMBER IS =
                                  ',NUMBER:8:2);
     SUM:=SUM+NUMBER;
```



```
     I:=I+1
    END;
   AVERAGE:=SUM/N;
   WRITELN('THE AVERAGE IS = ',AVERAGE:8:2);
READLN
END.
```

مثال٥:

اكتب برنامجا لحساب قيمة التابع:

$$y=\begin{cases}|x| & ; x<0\\ e^{x} & ; 0\le x\le 1\\ \dfrac{1}{(2x+1)} & ; x>1\end{cases}$$

البرنامج بلغة تربو باسكال:

```
Program Culculate_Function;
Var
    i:integer;
   x,y:real;
Begin
    writeln('Input x Value');
    readln(x);
    if x<0 then
       y:=abs(x)
```

```
    else
      if x<=1 then
        y:=exp(x)
    else
       y:=1/(2*x+1);
    writeln('x=',x:8:1,' y=',y:8:1);
readln
End.
```

مثال٦:

إكتب برنامجا يقرأ درجة الحرارة ويطبع نوع الرياضة الموافقة لدرجة الحرارة وفق الجدول التالي:

نوع الرياضة	درجة الحرارة
Swimming	Temp>50
Tennis	40<temp<=50
golf	30<temp<=40
skiing	10<temp<=30
Chinese checkers	Temp<=10

إن خوارزمية الحل هي بالشكل التالي:

١-قراءة درجة الحرارة.

٢-طباعة نوع الرياضة.

٢-١- إذا كانت درجة الحرارة >50 نفذ:

اطبع "swimming"

٢-٢- إذا كانت درجة الحرارة <= و >= 50 نفذ:

اطبع "tennis"

٢-٣- إذا كانت درجة الحرارة <= و >= 40 نفذ:

اطبع "golf"

٢-٤- إذا كانت درجة الحرارة <= 30 و >= 10 نفذ:

اطبع "sking"

٢-٥- إذا كانت درجة الحرارة < 10 نفذ:

اطبع " Chinese chechers"

البرنامج بلغة تربو باسكال:

```
Program Temperature;
Var
   temp:integer;
Begin
    writeln('Input Temperature');
    readln(temp);
    if temp>=50 then
       writeln('swimming')
    else
    if temp>=40 then
       writeln('tennis')
    else
```

```
    if temp>=30 then
       writeln('golf')
    else
    if temp>=10 then
       writeln('skiing')
    else
       writeln('chinese chechers');
readln
End.
```

مثال٧:

بســط العلاقة التالية إلى أقل ما يمكن مــن العبارات الضرورية:

```
If AGE > 60 then
     writeln('Coaial Security');
If AGE < 20 then
     writeln('Exempt');
If (AGE >= 20) and (AGE <=60) then
     writeln('Tax_Able');
```

الحل:

```
If AGE > 60 then
     writeln('Coaial Security');
If AGE < 20 then
     writeln('Exempt');
else
     writeln('Tax_Able');
```


مثال٨:

لتكن A,B,C ثلاث متحولات والمطلوب باستخدام تعليمة if-then-else أوجد القيمــة الأكبر مــن بين هذه المتحولات وذلك بثلاث طرق:

الحل:

a)
```
If  A > B then
    If  A > C then  Largest :=A
    Else   Largest :=C
Else
   If  B > C then  Largest:=B
    Else   Largest :=C
```

b)
```
If  A > B then  Largest :=A
Else   Largest:=B
If  C > Largest then  Largest :=C
Else   Largest :=C
```

c)
```
If  (A > B) and (A>C) then
            Largest :=A
Else
  If  (B > A) and (B>C) then
         Largest :=B
Else   Largest :=C
```

مثال٩:

اكتـب برنامجا يقوم بحسـاب مـعدل درجة الحرارة، درجة الحرارة العظمـى ودرجة الحرارة الدنيا خلال 24 سـاعة،حيث توجد قراءة لكل ساعة من ساعات اليوم.

مـن أجـل كـتابة برنامج لحساب درجات الحرارة لابد في البداية من التصريح عن المتحولات المستخدمة وهي :

١- متحول قراءة درجة الحرارة temp.

٢- متحول درجة الحرارة العظمى high.

٣- متحول درجة الحرارة الدنيا low.

٤- متحول يمثل مجموع درجات الحرارة sum.

٥- متحول يمثل معدل درجات الحرارة average.

في البداية يجب إسناد قيم ابتدائية للمتحولات كما يلي:

```
high:=-maxint;
low:=maxint;
hour:=1;
sum:=0;
```

ثم ننشـئ حلقة قراءة درجات الحرارة في كل ساعة من ساعات اليوم. يتم الخروج من الحلقة عندما يصبح قيمة العداد أكبر من 24 وهي عدد ساعات اليوم.

فـي الحلقة تتم عمـلية المقارنة لاختيار درجة الحرارة الدنيا و درجة الحرارة العظمى وإيجـاد مجموع درجـات الحرارة في 24 سـاعة. بعد تحقق الشـرط والخروج من الحلقة يتم حسـاب معدل درجات الحرارة،ثم تتم طباعة

المعدل ودرجة الحرارة الدنيا ودرجة الحرارة العظمى على الشـاشـة.

وهو المطلوب.

- البرنامج بلغة تربو باسكال:

```
Program Temperature_State;
{This program to calculates the average temperature,
 the high temperature and the low temperature from 24
 houtly temperature readings}

Var
  temp,  {an hourly temperature reading}
  high,  {highest temperature seen so far}
  low,   {lowest temperature seen so far}
  sum,   {running sum of temperatures}
  hour,  {loop control variable for hours in a day}
  average:integer;{average temperature}
Begin   {temp}
  {initialize high to impossibly low value}
  high:=-maxint;
  {initialize low to impossibly high value}
  low:=maxint;
  {initialize loop control variable}
  hour:=1;
  {initialize running sum}
  sum:=0;
  writeln('Input Temperatures');
  while hour <= 24 do
    begin
      {get temp}
      readln(temp);
```

```
        {add temp to sum}
        sum:=sum+temp;
        {check for new low}
        if temp < low then
          low:=temp;
        {check for new high}
        if temp > high then
          high:= temp;
        {increment loop control variable}
        hour:=hour+1;
      end;
  average:=sum div 24;
  {print}
  writeln('Average Temperature is ',average:6);
  writeln('High Temperature is ',high:8);
  writeln('Low Temperature is ',low:8);readln
End.
```

إن نتيجة تنفيذ البرنامج وفق القيم المعطاة هي بالشكل التالي:

```
Input Temperatures
22
21
23
55
44
42
41
51
52
32
```



```
33
34
35
45
46
13
14
12
22
21
40
18
55
3
41
```

```
Average Temperature is    31
High Temperature is       55
Low Temperature is        3
```

طريقة أخرى لحل البرنامج بالشكل:

```
Program Temperature_State;

Var
  temp,high,low,sum,hour,average:integer;
```

```
Begin
   ReadIn(temp);
   high:=temp;
  low:=temp;
  hour:=1;
  sum:=temp;
  writeln('Input Temperatures');
  while hour < 24 do
     begin
        readln(temp);
        sum:=sum+temp;
        if temp < low then
          low:=temp;
        if temp > high then
          high:= temp;
        hour:=hour+1;
      end;


  average:=sum div 24;
  {print}
  writeln('Average Temperature is ',average:6);
  writeln('High Temperature is ',high:8);
  writeln('Low Temperature is ',low:8);
readln
End.
```


تمارين عامة

١- اكتب برنامجاً لحساب مجموع السلسلة:

$$S=\sum_{i=1}^{50}\frac{1}{2i}$$

٢- اكتب برنامجاً لحساب سلسلة الذبذبات:

$$H=1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$$

٣- اكتب برنامجاً لحساب مجموع السلسلة:

$$S=\sum_{i=1}^{n}\frac{1}{3i}$$

٤- اكتب برنامجاً لحساب عاملي عدد أي حساب n!.

٥- اكتب برنامجا لحساب مجموع الأعداد الصحيحة الزوجية من 12 حتى 28 .

- ملاحظة: الحل يستخدم حلقة While- do فقط في الحل.

٦- صحح الأخطاء في البرنامج التالي:

```
Program Exercise;
Const
     A=10     B=5      C=6
Var
     D;E;F:integer
Begin
     Read(D,E F)
     If  (D>A) then  D= A+D;
```

```
    Elze D=A
    E:+D+F
    Writeln('This program does
            not make any sense,E,FD);
End.
```


الفصل الرابع

العمليات والتوابع الحسابية المعرفة في تربو باسكال

٤-١- العمليات الحسابية:

تسمح لنا لغة البرمجة تربـــو باسكال كغيرها من لغات البرمجة بإجراء العمليات الحسابية الأســاسية مـــن جمع وطرح وضرب وقســمة +، -، *، /. إضافة إلى عمليات أخرى، أما عملية الرفع إلى قوة فهي غير متاحة.

تعد المتحولات والأعــداد المكونـــات الأساسية للتعبير العددي إذ يتألف التعبير المركب مـــن أعداد ومتحولات تربط بينها عمليات حسابيـــة كالضرب والقسمة والجمع والطرح. إضافـــة إلى ذلك يمكن استخدام أقواس دائرية ضمن أي تعبير وأيضاً بعض التوابع المعرفة مسبقاً.

يستخدم الرمز / لعملية القسمة حيث يمكن كتابة بسط كسر ومقامه على سطر واحد، مثلاً: X/Y أو B/12 وهكذا...

ويجب كتابـــة إشارة الضرب على الشكل *، مثلاً: X*Y أو B*12.

مثلاً العلاقة الرياضية التالية:

$$\frac{0.5(A+1)}{2X}+(Y+2)(Z-3)$$

تكتب برمجياً على الشكل التالي:

$$0.5*(A+1)/2*X+(Y+2)(Z-3)$$

نشـــير هنا إلى أنه يمنع كتابة إشارتين متتاليتين لعمليتين حسابيتين على التتالي كأن نكتب مثلاً: 3-*7 أو 8-/7، إنما تكتب هــذه العبارات على الشكل التالي:

7/(-8), 7*(-3)

٤-٢- الأولويات بين العمليات الحسابية:

معــنى الأولوية بين العمليات هو تحديد أي من العمليات الحسابية ستنفذ أولاً في حال عدم وجود أقواس بين العمليات.

تنفذ العمليات الحسابية مــن اليسار وإلى اليمين آخذيــن بعين الاعتبار الأولوية، حيث تكون للضرب والقسمة وهي على نفس المستوى أيهما يأتي أولاً ثم للجمع والطرح وهي على نفس المستوى أيضاً أيهما يأتي أولاً.

أمثلة:

١- إن التعبير البرمجي A*B+C يكافئ التعبير البرمجي (A*B)+C وذلك لأن للضرب * أولوية على الجمع +.

٢- إن التعبير البرمجي A/B*C يكافئ التعبيــر (A/B)*C وذلك لأن القسمــة والضــرب متساويان في الأولوية والحساب يجري من اليسار وإلى اليمين أيهما يأتي أولاً.

٤-٣- التوابع الحسابيــة المعرفة على الأعــداد الحقيقية في تربو باسكال:

تسمى هذه التوابع أيضاً توابع حقيقية وذلك لأن هذه التوابع تطبق على

المتحولات الحقيقية وهي ذات قيم حقيقية.

ليكن x متحولاً من الأعــداد الحقيقية أي x: real; عندها نكتب الجدول التالي:

مسلسل	التابع الرياضي	التابع بلغة تربو باسكال
١	مربع العدد x	sqr(x)
٢	الجذر التربيعي للعدد x	sqrt(x)
٣	القيمة المطلقة للعدد x	abs(x)
٤	جيب الزاوية x	sin(x)
٥	تجيب الزاوية x (جيب التمام)	cos(x)
٦	القوس الذي ظله x (الظل العكسي)	arctan(x)
٧	الأس النيبري (تابع القوى بأساس نيبري e^x)	exp(x)
٨	اللوغاريتم الطبيعي $\log_e(x)$	ln(x)
٩	عدد عشوائي منتظم التوزيع بالمجال [0,1]	random(x)
١٠	ثابت عددي قيمته 3.1415	PI(x)
١١	قوة عدد ما a^x	exp(x*ln(a))

ملاحظة:

يمكن الاعتمــاد على تابــع القوى بأســاس نيبري لحساب a^x بالشكل التالي:

لدينا

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

أي:

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

إذاً يمكن التعبير عن a^x في صيغة تربو باسكال بالشكل:

exp(x*ln(a))

مثال:

اكتب برنامجاً لحل معادلة الدرجـــة الثانيـــة $Ax^2 + Bx + C = 0$ مهما كانت قيم A, B,C.

نحن نعلم حل معادلة الدرجة الثانيـــة بالشكل الرياضي التقليـــدي ولكن ســـوف نكتب الآن خوارزمية الحل والتي عليها سوف نكتب البرنامج بلغة تربو باسكال.

- لغة الخوارزمية:

١- إقرأ A, B,C

٢- إذا a = 0 نفذ:

أ- اكتب المعادلة ليست من الدرجة الثانية

ب- إذا b <> 0 نفذ:

اكتب الحل = -c/b

وإلا اكتب المعادلة غير صحيحة

٣- وإلا

أ- حساب المعين $D := B^2-4*A*C$

ب- حساب الجذور

إذا D = 0 نفذ:

- جذران متماثلان

X := -b/2*a -

-اكتب " للمعادلة جذران متماثلان =",x

وإلا إذا D > 0 نفذ:

- جذران حقيقيان

RD := sqrt (D) -

x1 := (-b-RD)/(2*a) -

x2 := (-b+RD)/(2*a)

- اكتب جذران حقيقيان

- اكتب "x1 =", x1, "x2 =", x2

وإلا

- جذران عقديان

RD := sqrt (|D|) -

Rx := -b/(2*a) -

Ry := RD/(2*a)

- اكتب جذران عقديان

- اكتب "x1 =", Rx, "+i", Ry

"x2 =", Rx, "-i", Ry

- لغة تربو باسكال:

-

```
Program quation2E(input,output);
(*find root of second degree equation:parameters
                                        a,b,c *)

Var
   a,b.c:real;
   D,RD,RX,RY,X,X1,X2:real;
```

```
Begin
  Writeln('Enter The Numbers a,b,c');
  readln(a,b,c);
  if (a=0) then
   begin
    Writeln('Equation is not second degree');
    if(b<>0)then
     Writeln('solution=',-c/b:8:3)
    else
     Writeln('Equation is trivial or not correct')
   end
  else
   begin
    D:=b*b-4*a*c;
    if (D=0) then
   begin
     X:=-b/(2*a);
     Writeln('Identical roots=',X);
   end
  else
  if (D>0) then
   begin
     RD:=sqrt(D);
     X1:=(-b-RD)/(2*a);
     X2:=(-b+RD)/(2*a);
     writeln('Tow real roots:');
     writeln('X1=',X1:8:3,'  ',' X2=',X2:8:3);
   end
  else
```



```
    begin
      RD:=sqrt(abs(D));
      RX:=-b/(2*a);
      RY:=RD/(2*a);
      writeln('Complex root:');
      writeln('X1=',RX:8:3,' +i',RY:8:3);
      writeln('X2=',RX:8:3,' -i',RY:8:3):
    end;
   end;
readln
End.
```

٤-٤- العمليات الحسابية المعرفة على الأعداد الصحيحة في باسكال:

إضافة إلى العمليات الحسابية +، -، *، /. المعرفة في تربو باسكال تُعرف أيضاً عمليتي القسمة الصحيحة div وباقي القسمة mod.

ليكن A و B عددين صحيحين أي

```
A, B : integer;
```

إن التعبير الحسابي A div B هو تعبير حسابي قيمته عدد صحيح مساوٍ لناتج قسمة A على B.

إن التعبير الحسابي A mod B هو تعبير حسابي قيمته عدد صحيح مساوٍ لباق قسمة A على B.

٤-٤-١- أمثلة:

- إن 15 div 6 هو ناتج قسمة 15 على 6 وهو 2 أي نستطيع أن نكتب:

15 div 6 =2

- وإن 15 mod 2 هو باقي قسمة 15 على 6 وهو 3 أي نستطيع أن نكتب:

15 mod 6 = 3

- ومن أجل القيم السالبة لدينا

-7 div 2 = -3

و

-7 div (-2) = 3

- إن

3 div 5 = 0

- إن A mod B هو دوماً غير سالب:

3 mod 2 = 1

(-10) mod 5 = 0

(-14) mod 3 = 1

3 mod 5 = 0

٤-٤-٢- ملاحظات:

- إن قيمة A mod B تُحدد فقط من أجل B > 0.

- وإذا كانت A ≥ 0 عندها:

A mod B = A – ((A div B)*B)

من أجل A = 8 و B = 3 نجد:

8 mod 3 = 8 – ((8 div 3)*3)

2 = 8 – (2 * 3)

2 = 8 –6 = 2

وهو محقق دوماً.

- إذا كانت A < 0 عندها نقبل أن:

A mod B = A + ((A div B)*B) + B

حيث أن قيمة A mod B هي دوما ليست سالبة

من أجل A = -7 و B = 2 نجد:

-7 mod 2 = -7 + (((-7) div 2)*2) + 2

1 = -7 + (3 * 2) + 2

1 = -7 + 8 = 1

وهو محقق دوماً.

- مما تقدم نستنتج دوماً أنه إذا كانت A ≥ 0 فإن:

A mod B = A – ((A div B)*B)

وإذا كانت A < 0 فإن:

A mod B = A + ((A div B)*B) + B

- نقول عن عدد أنه زوجي إذا كان يقبل القسمة الصحيحة على 2 دون باقٍ أي إذا كان A عدداً ما فإن:

A mod 2 = 0

يعني أنه عدد زوجي ويمكن اختيار ومعرفة العدد الزوجي بالشكل:

If ((A mod 2) = 0) then

٤-٥- توابع المعرفة على الأعداد الصحيحة في تربو باسكال:

سوف نتطرق في هذه الفقرة إلى بعض التوابع المعرفة على الأعداد الصحيحة وهم تابع القطع Trunc و وتابع التدوير Round وتابع الترتيب السلمي Pred، Succ.

٤-٥-١- تابع القطع Trunc: (القسم الصحيح من عدد حقيقي)

تابع القطع يطبق على الأعداد الحقيقية ليأخذ الجزء الصحيح من العدد الحقيقي أي الدخل من نمط حقيقي والخرج من نمط صحيح بالشكل:

Trunc : Real ⟶ Integer

مثلاً:

Trunc (-12.7) = -12

Trunc (15.32) = 15

٤-٥-٢- تابع التدوير Round: (تقريب عدد حقيقي لأقرب عدد صحيح)

إن تابع التدوير هو تابع يقرب العدد إلى أقرب عدد صحيح حيث أن الدخل هو من نمط حقيقي والخرج هو من نمط صحيح كما في الشكل:

Round : Real ⟶ Integer

يصمم هذا التابع (تابع التدوير) تابع القطع في عملياته بالشكل التالي:

Round (x) = Trunc (x + 0.5) if x > 0

Round (x) = Trunc (x - 0.5) if x < 0

أمثلة:

Round (-7.5) = -8

Round (4.5) = 5

Round (14.6) = 15

Round (-12.4) = -12

٤-٥-٣- توابع الترتيب السلمي Pred، Succ: (العنصر السابق والعنصر اللاحق)

إن التابع Pred يعني ما قبل وهو اختصار للكلمة Predecessor. أما التابع Succ يعني ما بعد وهو اختصار للكلمة Successor.

ويعرفان بالشكل:

Succ (I) = I + 1

Pred (I) = I - 1

أي أن Pred (I) يعطي ما قبل العنصر I و Succ (I) يعطي ما بعد العنصر I.

إن هذه التوابع معرفة على جميع الأنماط المتقطعة حيث أن الأنماط المتقطعة هي الأنماط المرتبة، التي يمكن فيها تحديد العنصر السابق والعنصر اللاحق. إن الأعداد الحقيقية مع كونها مرتبة لكن ليست نمطاً سلمياً إذ لا يمكن تحديد العدد التالي لعدد حقيقي أو العدد السابق لعدد حقيقي.

مثلاً:

العدد التالي للعدد 5.2 هو 5.215.201 أو 5.001 وهكذا ... أو هو 5.221 ... الخ.

أي لا يمكن تطبيق هذه التوابع على المتحولات الحقيقية.

٤-٦- مثال تعليمي:

اكتب برنامجاً لحساب معكوس عدد صحيح. المطلوب من البرنامج هو قـراءة مجموعة من الأعداد وكتابة كل منها بوجه معكوس. يتـم تحديـد عدد الأعداد المطلوب إيجاد معكوسها بشكل مسبق.

نلاحظ أن معكوس العدد 543 هو 345 ومعكوس العدد 1243- هو العدد 3421- ومعكوس 300 هو 3.

- خوارزمية البرنامج:

١- إقرأ Q (يمثل عدد الأعداد).

٢- I := 1

٣- مادام N => I كرر

أ- إقرأ N (العدد المطلوب إيجاد معكوسه)

ب- احسـب معكوس N وضعـه في inv (مجموعة الأعـداد المعكوسة)

ج- اكتب N, inv (العدد ومعكوسه)

د- I := I + 1

إن الخطوات السابقة تمثل مراحل عمل الخوارزمية وبالتالي البرنـامج، أما حساب معكوس N فتحتاج إلى شرح آخر.

- <u>عملية حساب معكوس عدد:</u>

لشرح هذه العملية لا بد من أخذ المثال التالي:

ليكن لدينا العـدد 1245 إن الخانة الأولى والتي هي هنا 5، هي عبارة عن الباقي من قسمة العدد 1245 على 10، أي نستطيع أن نكتب أن:

1245 mod 10 = 5

أمـا العدد المتبقي أو المؤلف مـن الخانات المتبقية أي 124 فهو عبارة عن ناتج قسمة العدد الأصلي على 10 أي هو

1245 div 10 = 124

ولإزاحة خانة في العدد إلى اليسار لحساب المعكوس تتم بعملية الضرب بـ 10. ولنفرض أن Rest هو العدد المؤلف من الخانات المتبقية أي التي لم نستخرجها بعد فالوصول إلى المعكوس يتم كما في الجدول التالي:

Rest	Rest div 10	Rest mod 10	Inv
1245	124	5	0
124	12	4	0 * 10 + 5 = 5
12	1	2	5 * 10 + 4 = 54
1	0	1	54 * 10 + 2 = 542
0	-	-	542 + 10 + 1 = 5421

من الجدول السابق نكتب:

نبدأ بإسناد قيمـة العـدد المـراد عكسه للمتحول Rest ونسند القيمة 0 للمعكوس inv. ثم نكرر إضافـة الخانة (محسوب بعملية mod) إلى inv بعـد ضربه بـ 10 (إزاحة) ثم نضع الخانات المتبقيـة بـ Rest (محسوبـة بعملية div). يتم تكرار تنفيذ حساب Rest و inv حتى لا يبقى خانـات، (ما دام العدد المكافئ للخانات المتبقية مختلفاً عن الصفر) أي حتى يصبح:

Rest div 10 = 0

- خوارزمية حساب المعكوس:

61

Rest := N -١

Inv := 0 -٢

٣- مادام

Rest div 10 <> 0 كرر

Inv := inv * 10 + (Rest mod 10)

Rest := Rest div 10

Inv := inv * 10 + (Rest mod 10)

وهو المطلوب.

- لغة تربو باسكال:

```
PROGRAM INVERS;
(*invers integer number,examples: 1357 to 7531,23 to 32
*)
VAR
  LL,L:INTEGER;
  INV,REST,I:INTEGER;
BEGIN
  WRITELN('How many numbers to invers:');
  READLN(LL);
  I:=1;
  WHILE (I<=LL) DO
   BEGIN
    WRITELN('Give the number:');
    READLN(L);
    REST:=L;
    INV:=0;
    WHILE ((REST DIV 10)<>0) DO
     BEGIN
```



```
      INV:=INV*10+(REST MOD 10);
      REST:=REST DIV 10;
     END;
    INV:=INV*10-(REST MOD 10);
    WRITELN(L:8,INV:8);
    I:=I+1;
   END;
READLN
END.
```

مثال:

احسب قيمة كل من التعبيرات التالية.إذا كان التعبير نظامي فاكتب الناتج وحـدد إلـى جانـبه هل هو real أو integer وإلا اكتب أن التعبير غير نظامي واشرح لماذا.

a)10 3+5*2 b)10 mod 3+5 mod 2
c)10 div 3+5 div 2 d)12.5+(2.5/(6.2/3.1))
e) -4-(-5+6) f)13 mod 5/3
g)(10 3 mod 2)/3

الحل:

a) 13.33 real
b) 2 integer
c) 5 integer
d) 13.76 real
e) -4 integer
f) 1.0 real
g)بالنسبة للعبارة الأخيرة لا يمكن حسابها لأن mod يطبق فقط على الأعداد الصحيحة.

تمارين عامة

١- أوجد ناتج كل من العبارات التالية:

1- Succ (Round (5/2) – Pred (3));

2- 3 * 1 div 2 mod 7/3 – Trunc (sin (1));

٢- بين أولوية العمليات بالنسبة للعلاقة التالية:

- a mod b +a div b * c

٣- بين أي من العبارات التالية صحيحة ولماذا:

1- k mod 7 = k div 5 – 1 ; k = 1

2- succ (' 0 ') = pred (' 3 ') ;

3- ord (' q ') < = ord (' z) ;

4- chr (ord (' 5 ') + 1) = ' 7 ' ;

5- ord (' z ') - ord (' a ') = 26

6- ord (pred (chr (24)) = 22

٤- اكتب الصيغة الرياضية التالية بلغة تربو باسكال:

$$\frac{\sin s + \cos x}{\frac{x+y}{4}} - (\ln x)$$

٥- اكتب التعبير البرمجي التالي بشكل رياضي:

+A – sqrt (sqr(B) – 8 + C)/(5 * a)

٦- احسب قيمة التعبير البرمجي التالي:

(sin (sqr (A) – 4) + 5 * abs (B)/ cos (3 + 4)

حيث

A = 2 و B = -3

٧- حدد القيم التي ستطبع عند تنفيذ المقطع التالي من برنامج.

A := (sin (sqr (3) –9) + 3 * abs (- 1)/cos (5 – 5);

B := A * (sqrt (3));

Writeln (A, B);

٨- أوجد ناتج تنفيذ المقطع التالي من برنامج

Readln (A, B);

C := sqr (sqrt (A) + sqrt(B));

Writeln (C);

حيث أن قيم المدخلات على الترتيب هي 0.25، 0.09

٩- اكتب ناتج تنفيذ المقطع التالي من برنامج:

Readln (A, B);

A := A + 2;

B := B * B;

A := A – B;

Writeln (A, B);

حيث أن المدخلات على الترتيب هي: 3، 2.

١٠- اكتب مقطع من برنامج يدخل عدداً حقيقياً ثم يحسب لغارتمه الطبيعي أي:

x: neal; Ln (x)

١١- أوجد ناتج مايلي:

a) 15 mod 4
b) 7 div + 2
c) 2 +2*7
d) 45 div 8*4+2
e) 17+(21 mod 6) *2
f) sqr(4*2+2)

الفصل الخامس

نمط المتحولات المنطقية Boolean

٥-١- مقدمة:

درسنا حتى الآن متحولات من النوع العددي فقط، ورأينا أنها تقسم إلى قسمين: صحيح وحقيقي. بالإضافة إلى أنواع المعطيات العددية تملك لغة تربو باشكال نوعاً مسبق التعريف لتخزين القيم المنطقية، تأخذ متحولات هذا النوع إحدى القيمتين: True أو False (صح أو خطأ).

إن نوع المعطيات أو المتحولات من نمط بولياني لا يملك سوى قيمتين وحيدتين هما True و False. إن القيم البوليانية لا تحتاج إلى فواصل علوية، ذلك لأن هذه القيم هي قيم منطقية مسبقة التعريف.

٥-٢- التصريح عن متحول من نمط منطقي Boolean:

يجري التصريح عن متحولات منطقية بكلمة النمط Boolean بالشكل التالي مثلاً:

Var t, T: Boolean

نذكر هنا أنه لا يمكن قراءة قيم لمتحولات بولبانية، ولكن يمكن إسناد قيم عبارات منطقية لها. مثلاً إذا كتبنا

T:= A < B

هذا يعني أن المتحول T وهو متحول بولياني يأخذ القيمة True إذا كانت قيمة المتحول العددي A أصغر من قيمة المتحول العددي B، وفي حال عدم تحقق A

أصغر من B فإن قيمة المتحول T تكون False.

إن النمط المنطقي Boolean في تربـــو باسكال هو نمط مرتب، فالقيمة False أصغر من القيمة True.

٥-٣- عمليات القراءة والكتابة والإسناد:

يجب التنويه هنا إلى أنه لا يمكن إدخال قيم بولينية بواسطة الإجراءين Read أو Readln، والحل هو إلحاق قيمـــة بولينية بمتحول بولياني، وذلك بأن نضع تعبيراً بولينياً في الطرف الأيمن من معامل الإســـناد حيث يعطي التعبير البولياني إحدى القيمتين المنطقيتين إما True أو False. حيث تتـــم القراءة فقط بعملية الإسناد. يجري الإسناد بالشكل:

مثلاً

```
b := True;
```

أو

```
b := False;
```

أما الكتابة فيسمح بكتابة المتحولات والتعابير من نمط Boolean مثال:

```
b : True;
Writeln (b);
```

هذا يؤدي إلى كتابة True على وحـــدة الخرج وهنا الشاشة هي وحـــدة الخرج مثلاً.

٥-٤- العمليات المعرفة على النمط المنطقي:

إن العمليات المعرفة على النمـــط المنطقي هـــي: AND، OR، XOR

، NOT وهي أدوات التركيب في التعبير المنطقي.

٥-٤-١- العملية AND:

ليكن A متحولاً من نمط Boolean عندها يمكن كتابة التعبير التالي:

```
A := (M > 0)  AND  (N < 0);
```

إن التعبير السابق هـــو تعبير منطقي، وإذا تحقق فإن A تأخذ القيمة True، وإذا لم يتحقق فإن A تأخذ القيمة False. وتتحقق A عندما تكون M > 0 و N < 0 في آن واحد.

نكتب الآن جدول الحقيقة للعملية AND الذي سوف يبين ناتج كل حالة من الحالات الممكنة:

قيمة A	قيمة B	قيمة A AND B
False	False	False
False	True	False
True	False	False
True	True	True

نذكر هنا أن الأقواس حـــول التعابير البوليانية ضرورية بســـبب كون العملـــية AND تملك أسبقية أعلى من أسبقية معاملات المقارنة الأخرى. فإذا لم نضع أقواساً فإن هذا سوف يؤدي إلى حدوث خطأ.

٥-٤-٢- العملية OR:

إن تطبيق العملية OR على حدين بوليانيين تكون النتيجة قيمة بوليانية، هذه القيمة هي False عندما تكون قيمـــة الحدين False وإلا تكون النتيجة هي True.

ليكن B متحولاً من نمط Boolean عندما يمكن كتابة التعبير التالي:

A := B OR (M > 0):

إن A تأخذ القيمة True إذا ما تحقق إحدى الحالتين إما B أو M > 0.

نكتـــب جدول الحقيقة للعملية OR، والتي تبين لنـــا ناتج كل حالة مـــن الحالات الممكنة:

قيمة A	قيمة B	قيمة A OR B
False	False	False
False	True	True
True	False	True
True	True	True

نذكر هنا أيضاً أن الأقواس حول التعابير البوليانية ضرورية بسبب كون العملـــية OR تملك أسبقية أعلى مـــن أسبقية معاملات المقارنة الأخرى. فإذا لم نضع أقواساً فإن هذا سوف يؤدي إلى حدوث خطأ.

إن الفرق بين جدول الحقيقة للعملية AND وجدول الحقيقة للعملية OR هو في الحالتين عندمـــا تكون قيمة كلا الحدين مختلفين فالعملية AND تعطينا القيمة False أما العملية OR فتعطينا القيمة True.

٥-٤-٣- العملية XOR:

إن العمليات السابقة هي عمليات معرفة في لغـــة باسكال القياسية أمـــا في لغة تربو باسكال فتوجد عملية إضافية وهي XOR التي تعطينا القيمة True فقـــط إذا كانت قيمة أحـــد الحدين True (أي تعطينا True عند اختلاف قيمـــة الحدين).

نكتـــب جدول الحقيقة للعملية XOR، والتي تبين لنـــا ناتج كل حالة من الحالات الممكنة:

قيمة A	قيمة B	قيمة A XOR B
False	False	False
False	True	True
True	False	True
True	True	False

إن هذه العملية تختلف عن العملية OR فقط في الحالة عندما تكون كلا القيمتين True حيث OR يعطينا True أما XOR فيعطينا القيمة False.

٥-٤-٤- العملية NOT:

العملية الأخيرة هي عملية NOT. تطبق هذه العملية على حد بولياني واحد ويعيد قيمة بولياني واحدة معاكسة لقيمة الحد. إذا كانت قيمة الحد هي True فإن العملية NOT ستعطينا False. للعملية NOT أسبقية أعلى من العملية AND، هذا يعني أن NOT ستنفذ أولاً العملية ثم العملية AND.

أمثلة:

لتكن لدينا العمليات التالية:

1) NOT (6 > 9)
2) NOT NOT True
3) NOT (3 > 4) AND (4 > 6)
4) NOT ((3 > 4) AND (4 > 6))
5) NOT True

في المثال (١) بما أن 6 ليست أكبر من 9 فإن هذه العبارة خاطئة أي False ونفي الـ False يعطي True إذا جواب المثال الأول هو True.

في المثال (٢) يكون نفي الصح هو خطأ ونفي الخطأ هو صح إذا جواب المثال الثاني يكون True.

في المثال (٣) يتم تنفيذ NOT (3 > 4) أولاً وذلك لأن لـ NOT أسبقية على AND فيكون NOT (3 > 4) هو True وجواب (4 > 6) هو False وبالتالي فإن True AND False هو False إذاً جواب المثال الثالث هو False.

في المثال (٤) تم وضع أقواس على العملية فيكون (False AND False) هو NOT (False) وهو True أي جواب المثال الرابع هو True.

في المثال (٥) نفي الصح هو خطأ أي جواب المثال الخامس هو False.

إن العبارات التالية هي عبارات خاطئة:

NOT ('v')

NOT (5)

NOT ('Seeyou')

حيث لا توجد عمليات مقارنة ولا توجد قيم لمتحول منطقي التي هي إما True أو False.

٥-٤-٥- أسبقية العمليات البولينية:

تمتلك كل العمليات البولينية أسبقية أعلى من أسبقية عمليات المقارنة وعملية الإسناد طبعاً.

إن الجدول التالي يرتب لنا العمليات التي تعرفنا عليها حتى الآن من حيث أسبقيتها:

نوع العملية	العملية
النفي المنطقي	NOT
عمليات الضرب والقسمة	* , / , DIV , MOD , AND
عمليات الجمع والطرح	+ , - , OR XOR
عمليات المقارنة	= , <> , > , < , >= , <, <=
عملية الاسناد	:=

٥-٥- قوانين الجبر البولي:

توجد قوانين كثيرة لتحديد العلاقة بين العمليات البولیانية، وهذه القوانين مهمة من أجل تركيب وتبسيط التعابير البولیانية.

ليكن A ، B ، C متحولات من نمط منطقي عندها نكتب القوانين التالية:

١- قوانين عامة:

1) A AND False = False
2) A AND True = A
3) A OR False = A
4) A OR True = True
5) A AND A = A
6) A OR A = A
7) A AND NOT A = False
8) A OR NOT A = True
9) NOT NOT A = A

٢- قوانين التبديل:

1) A AND B = B AND A
2) A OR B = B OR A

٣- قوانين التجميع:

1) (A AND B) AND C = A AND B AND C
= A AND (B AND C)

2) (A OR B) OR C = A OR B OR C
= A OR (B OR C)

٤- قوانين التوزيع:

1) (A AND B) OR (A AND C) =
A AND (B OR C)
2) (A OR B) AND (A OR C) =
A OR (B AND C)

٥- قوانين دومورغان:

1) NOT (A AND B) = NOT A OR NOT B
2) NOT (A OR B) = NOT A AND NOT B

مما تقدم نستطيع القول أن المتحولات مـن النمط المنطقي مع العمليات المنطقية تخضع لقواعد جبر بول كمـا في مجموعـات المفهوم الريـاضي: المجموعة الكليـة، المجموعة الخالية، العمليـات على المجموعـات: التقاطع، الاجتماع الإتمام والتي تقابل هنا في حالتنا صح، خطأ، و، أو، لا.

٥-٦- أمثلة عامة:

مثال١:

بسط التعبير المنطقي التالي إلى أبسط صورة حيث A، B متحولين من نمط Boolean.

(NOT A AND NOT B) OR (NOT A AND B) OR (A AND B)

الحل:

(NOT A AND NOT B) OR (NOT A AND B) OR (A AND B) =

= NOT A AND (NOT B OR B) OR (A AND B)

= NOT A AND True OR (A AND B)

= NOT A OR (A AND B)

= (NOT A OR A) AND (NOT A OR B)

= True AND (NOT A OR B)

= NOT A OR B

مثال٢:

اكتب برنامجاً يبين فيما إذا كان المتحول العـددي x ينتمي إلى المجال العددي [a, b] أم لا.

الحل:

إذا كـان العدد ينتمي إلى المجال فأجب بكلمة صح. وإذا كان العـدد لا ينتمي إلى المجال فأجب بكلمة خطأ.

- لغة الخوارزمية:

١- إقرأ الأعداد a, b, c

٢- إذا a =< x <= b نفذ:

اكتب "صح"

وإلا

اكتب "خطأ"

لحل هذا المثال لا بد من إدخال الأعداد الحقيقية a, b, c أولاً ثم اختبار شرط انتماء العدد x إلى المجال [a, b] من خلال تحقق الشرط a =< x >= b.

- لغة تربو باسكال:

```
Program BelongNumber;
Var
   x,a,b:real;
   y:boolean;
Begin
   writeln('Input x,a,b');
   readln(x,a,b);
   y:=(x<=b)and(x>=a);
   if y then
      writeln('Right')
   else
      writeln('Wrong');
readln
End.
```

في المثال يمكن استبدال الإجابـــة بكتابة

Writeln (y);

وسوف تكون الإجابة إما True أو False وذلك حسب المعطيات المدخلة.

مثال٣:

لتكن لدينا المتتاليتان:

$$X_{i+1} = Y_i/3$$

$$Y_{i+1} = X_i/3 - 6Y_i/9$$

و $\varepsilon > 0$ عدد حقيقي صغير معطى. والمطلوب كتابة برنامج لإيجاد أصغر عدد صحيح i يحقق العلاقتين $|X_i| < \varepsilon$ و $|Y_i| < \varepsilon$ حيث أن $X_0 = Y_0 = 3$.

الحل:

لا بد في البداية مـــن إدخال متحول بولياني وليكن Q نســـند لـــه القيمة False في البداية حيث يكون $\varepsilon > |X|$ و $\varepsilon > |Y|$ - Q أي يحقق الشرط ويجب حســـاب قيمة كل من المتتاليتين طالما أن الشرط Q قيمته False وعندما يصبح Q، True أي محققاً لأول مرة. اكتب قيمة المتحول الصحيح I الذي يعد عـــداداً لحدود المتتاليتين والذي عنده تتحقق Q أي تصبح Q True.

- لغة الخوارزمية:

١- إقرأ eps

٢- أسند 3 =: x و 3 =: y.

٣- أسند 0 =: I و False =: Q.

٤- ما دام NOT Q كرر

٤-١- X1 := Y/3

٤-٢- Y := X/3 - 6 Y/9

٤-٣- X := X1

٤-٤- I := I + 1

٤-٥- Q := (abs (x) < eps) AND (abs (y) < eps)

٥- اكتب I

- لغة تربو باسكال:

```
Program Epselon;
Var
   X,Y,X1,EPS:real;
        I:integer;
        Q:boolean;
Begin
```



```
   readln(EPS);
   X:=3;Y:=3;
   I:=0;
   Q:=false;
   while not Q do
    begin
      X1:=Y/3;
      Y:=X/3-6*Y/9;
      X:=X1;
      I:=I+1;
      Q:=(abs(X)<EPS)and(abs(Y)<EPS)
    end;
   writeln(I);
readln
End.
```

نتيجة تنفيذ البرنامج:

من أجل: eps := 0.3 ⇐ I = 8

eps := 0.9 ⇐ I = 3

eps := 0.1 ⇐ I = 13

eps := 0.004 ⇐ I = 28

مثال ٤:

اكتـــب برنامجاً يكتب جدول العمليـــات أو جدول الحقيقة لكل مـــن AND و OR و NOT و XOR ويتحقق من قانوني مورغان.

- لغة تربو باسكال:

```
PROGRAM SHOW_BOOLEAN;
(*SHOW BOOLEAN OPERATIONS*)
```

```
VAR T,F:BOOLEAN;
BEGIN
    T:=TRUE;F:=FALSE;
    WRITELN('AND TABLE:');
    WRITELN(F,' AND ',F,'=',F AND F);
    WRITELN(F,' AND ',T,'=',F AND T);
    WRITELN(T,' AND ',F,'=',T AND F);
    WRITELN(T,' AND ',T,'=',T AND T);
    WRITELN;
    WRITELN('OR TABLE:');
    WRITELN(F,' OR ',F,'=',F OR F);
    WRITELN(F,' OR ',T,'=',F OR T);
    WRITELN(T,' OR ',F,'=',T OR F);
    WRITELN(T,' OR ',T,'=',T OR T);
    WRITELN;
    WRITELN('NOT TABLE:');
    WRITELN(' NOT ',F,'=',NOT F);
    WRITELN(' NOT ',T,'=',NOT T);
    WRITELN;
    WRITELN('DE MORGAN LAW:');
    WRITELN('NOT(T AND F)  (NOT T)OR(NOT F)?',
              NOT(T AND F)=(NOT T)OR(NOT F));


  WRITELN('NOT(T OR F)=(NOT T)AND(NOT F)?',
              NOT(T OR F)=(NOT T)AND(NOT F));
  READLN
END.
```

نتيجة تنفيذ البرنامج:


```
AND TABLE:
FALSE AND FALSE=FALSE
FALSE AND TRUE=FALSE
TRUE AND FALSE=FALSE
TRUE AND TRUE=TRUE

OR TABLE:
FALSE OR FALSE=FALSE
FALSE OR TRUE=TRUE
TRUE OR FALSE=TRUE
TRUE OR TRUE=TRUE

XOR TABLE:
FALSE XOR FALSE=FALSE
FALSE XOR TRUE=TRUE
TRUE XOR FALSE=TRUE
TRUE XOR TRUE=FALSE

NOT TABLE:
 NOT FALSE=TRUE
 NOT TRUE=FALSE


DE MORGAN LAW:
NOT(T AND F)=(NOT T)OR(NOT F)?TRUE
NOT(T OR F)=(NOT T)AND(NOT F)?TRUE
```

24
24
25
25
28
28

تمارين عامة

١- اكتب برنامجاً يتحقق من قوانين التبديل والتجميع والتوزيع في حالتي AND و OR.

٢- اكتب برنامجاً لحساب قيمة التابع المعرف بالعلاقة:

$$Y=\begin{cases} X; & X<-3 \quad OR \quad X>2 \\ X^2-5X+5 & \text{خلاف ذلك} \end{cases}$$

٣- اكتب برنامجاً يظهر الأعداد التي تقبل القسمة على 7 و 9 في نفس الوقت والأعداد التي تقبل القسمة على 7 أو 9 وذلك للأعداد بين 1 و 90.

٤- لدينا المتحولات البولیانية التالية A,B,C,D حيث :

A=True , B=False , C=False , D=True

والمطلوب إيجاد القيمة البولیانية لكل من التعابير التالية:

a) (A AND B) OR (A AND D)
b) NOT C AND D AND A
c) NOT(A AND D) OR (C OR D)
d) (A OR B) AND NOT (C OR D)

٥- اكتب عبارة بولیانية تعرف متحول بولياني "مرشح" (Candidate) تكون قيمته True إذا كان score أكبر من أو يساوي 1000 والعمر أكبر من 18، المرشح في أي حالة أخرى يأخذ القيمة False .

٦- بسط العبارة المنطقية التالية إلى أبسط صورة.

A AND X OR B AND X OR C AND X OR
A AND Y OR B AND Y OR C AND Y OR
A AND C OR B AND C OR C

٧- لدينا متحول بولياني LeftPage مصرح عنه بالشكل التالي:

```
Var
    LeftPage:boolean;
    PageNumber:integer;
```

إن المتحول PageNumber هو عدد صحيح يدل على رقم الصفحة.اكتب الحالة التي تجعل من المتحول LeftPage يأخذ القيمة True إذا كان PageNumber عدد زوجي.

الفصل السادس

نمط المتحول المحرفي CHAR

٦-١- مقدمة:

تسمح لغة باسكال بتمثيل المعطيات غير الرقمية، حيث يوجد نوع متحولات مسبق التعريف لتمثيل الرموز. يدعى هذا النوع بالنوع المحرفي. يستخدم هذا النوع المحرفي قيماً على شكل رموز منفصلة، يتم إدخاله بواسطة لوحة المفاتيح. وبالتالية فإن هذا النوع يضم كل الحروف الأبجدية اللاتينية، والأرقام العربية ورموز العمليات والأقواس وغيرها من الرموز.

أي أنه يمكن اعتبار أي رمز، يتم إدخاله بواسطة لوحة المفاتيح رمزاً من النوع المحرفي. ويرمز له بالرمز Char.

٦-٢- شفرة أسكي:

يستخدم نوع المتحولات من نمط محرف في لغة باسكال من أجل تخزين رموز مستقلة على شكل شفرات ASCII. حيث يخصص للمتحولات من النمط Char بايت واحد في الذاكرة يقوم بتخزين 256 رمزاً بين (0) و (255) تتضمن مجموعة رموز أسكي ASCII القياسية 128 رمزاً ومجال رموزها بين (0) و (127) أما الرموز من 128 إلى 255 فتعرف على أنها رموز ASCII الموسعة Extended ASCII Character والتي قد تستخدم لأغراض خاصة في بعض البرامج.

إن ترميز 128 عنصراً يكفي له 7 bits، أما الخانة الثامنة فكانت

مخصصة لكشف الأخطاء. إذ تساعد على كشف الأخطاء في المعلومة الواصلة. ومع التطوير في وثوقيـــة الأجهزة الحاسوبيـــة، لم تعد هناك حاجة للكشف عن الأخطاء، فأصبحت بعض الحواسيب تستخدم هـــذه الخاصة في ترميز مجموعة أكبر من العناصر مما سمح بترميز عناصر أخرى في المجال 128 وحتى 255، تسمح بإظهار أشكال أخرى غير الأحرف المعروفة لدينا.

٦-٣- رموز أسكي:

كل رمز ضمن مجموعة رموز ASCII، يتطابق مع رقم ترتيب له في المجموعة إذا يملك الحرف A الترتيب 65، أمـــا الحرف a فيمـــلك الترتيب 97 ورموز ASCII للأحرف مرتبة بشكل تسلسلي هذا يعني أن ترتيب B هو 66، وترتيب C هو 67 وهكذا ... . نفس الأمر ينطبق على الأحرف الصغيرة.

بالإضافة إلى الأحرف الكبيرة والأحرف الصغيرة فإن مجموعة رموز أسكي ASCII تحتوي على الأرقـــام من (0) إلى (9) وبعض الرمـــوز الأخرى مثل: (# ، & ، $ ، ...). يجب الملاحظة هنـــا أن الأرقام هي كرمـــوز تختلف عن الأرقام كأعداد.

بالإضافة إلى الرمـــوز السابقـــة توجـــد مجموعـــة من الرموز تدعى بمجموعة رموز التحكم Control Character. وكل رمز من هذه الرموز يقوم بعمل محدد مثلاً رمز مفتاح الجدولة Tab هو رمز تحكم. إن جميع الرموز التي ترتيبها بين (0) و (31) هي رموز تحكم.

٦-٤- التصريح عن متحول من نمط محرفي:

يجري التصريح عـــن متحول مـــن نمط محرفي باستخدام كلمة Char مثلاً:


```
Var  P, Q : Char;
```

أي أن كل من المتحولين P و Q هو من نمط محرفي أي يمكن أن تأخذ قيماً هي أحد محارف لوحة المفاتيح.

٦-٥- عملية الإسناد:

عندما نصرح عن متحول أنه من نمط محرفي عندها يمكن أن نسند له قيم لهذه المتحولات بالشكل التالي:

مثلاً:

```
P := 'b' ;  Q := 'C' ;  p := 'I' ;
```

وهكذا، نضع الرمـــز المطلوب إســـناده إلى المتحول المحرفي مـــا بين فاصلتين علويتين وبعد تنفيذ عملية الإسناد يصبح قيمة المتحول المحرفي مساوية للرمز الظاهر بين الفاصلتين العلويتين.

إن الإسناد بالشكل ; A =: Q هو خاطئ والصحيح هو ; 'A' =: Q حيث في الحالة الأولى سيفهم التعبير على أنه إسناد قيمة متحول مسمى A إلى متحول مسمى Q. أما في الحالة الثانية فيعني أن المتحول Q أصبح قيمته هو المحرف A نفسه.

بعد تنفيذ عمليـــة إسناد إلى متحول مـــا، يمكن تنفيذ عملية كتابة هـــذا المتحول على وحدة الخرج ولتكن هنا الشاشة. يتم ذلك بإهمال الفواصل العلوية.

فمثلاً إذا كان :'C' =: Q فإن:

```
Writeln (Q);
```

سوف يؤدي إلى طباعة C فقط دون الفواصل العلوية.

٦-٦- عمليات المقارنة:

إن النمط المحرفي هـو نمط مرتب فعمليـات المقارنـة معـرفة لأي محرفيـن، فالترتيب معرف وذلك حسب جدول أسكي. إن النمط المحرفي هـو نمط متقطع.

وبالتالي فإنه يمكن تطبيق عمليات المقارنة في لغة باسكال القياسية على المـتحولات من النوع Char، وفي أي عمليـة مقارنـة تستخدم شفرات أسكي ASCII المقابلـة للمتحولات من النوع Char، مثلاً إن 'I' < 'L'. وهذا صحيح لأن مجموعة الأحرف الكبيـرة تقع بالترتيب قبـل مجموعة الأحرف الصغيرة وبالتالي فرقم ترتيبها أصغر من رقم ترتيب الأحرف الصغيرة.

إن 'A' < 'B' و 'C' < 'D' و 'Y' < 'Z' لهـا جميعاً قيمة True. علماً أن 'A' > 'B' و 'C' > 'D' و 'Y' > 'Z' لها جميعاً قيمة False.

٦-٧- جدول توصيف أسكي:

إن الترميز أسكي ASCII يعني الترميـز الأمريكي المعيـاري لتبادل المعلومـات (American Standard Code for Information Interchange) حيث تم استخدام 1 Byte = 8 Bits من أجل خزن قيمة بين 127 .. 0.

نورد فيما يلي وصف لجدول الترميز أسكي:

-1	0 .. 31	تسمى أحرف تحكم وهي أحرف غير قابلة للإظهار
-2	32	الحرف فراغ "Space"
-3	33 .. 47	أدوات تنقيط: / , . , - , , , + , * ,) , (, & , % , $, # , " , !
-4	48 .. 57	الأرقام (Digits): 9 ,..., 2 , 1 , 0
-5	58 .. 64	أدوات تنقيط: ? , > , = , < , ; , : , @
-6	65 .. 90	الأحرف الأبجدية الكبيرة: Z .. A

-7	96 .. 91	أدوات تنقيط: ` , _ , ^ ,] , \ , [
-8	122 .. 97	الأحرف الأبجدية الصغيرة: z ... a
-9	127 .. 123	أدوات تنقيط: } ,	, { , ~ , Del

٦-٨- التوابع المعرفة على النمط المحرفي:

٦-٨-١- تابع الترتيب أو رقم الترميز ORD: (Order)

يعطي هـذا التابع قيمة ترتيب المحرف ضمـن مجموعـة الأحـرف المستخدمة، هذه القيمة هي قيمة الترميز ASCII

بالشكل

ord: char → 0 .. 127

أي يطبق على المحارف ليكون الناتج ضمن المجموعة من (0) وحتى (127).

مثال:

```
Writeln (ord ('A'));
```

الجواب: 65.

٦-٨-٢- تابع التحويل من رقم لمحرف CHR: (Character)

إن التابع CHR هـو التابع العكسي للتابع ORD فهو يعطي المحـرف المقابل لعدد معطى من ضمن أرقام الترميز 127 .. 0.

بالشكل:

chr : 0 .. 127 → Char

مثال:

```
Writeln (chr (66));
```

الجواب: B.

42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G

72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_
96	`
97	a
98	b
99	c
100	d

42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G

72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_
96	`
97	a
98	b
99	c
100	d

101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	□

مثال٢:

اكتب برنامجاً يقوم بقراءة محارف وطباعتها ولا يتوقف البرنـامج إلا عند إدخال المحرف /*/ ويقوم بتغيير كل محرف مـن الشكل '?' أو الشكل '!' إلى نقطة '.'.

الحل:

نلاحـظ من نص المثال أن البرنامـج لا ينهي عملـه إلا عند إدخـال المحـرف '*' وأثناء عمله يراقب الرموز المدخلة ويقـوم بتغير أي محرف من الشكل '?' أو الشكل '!' إلى نقطة.

يمكن كتابة خوارزمية الحل بالشكل:

- لغة الخوارزمية:

١- إقرأ C

٢- ما دام "?" <> C كرر

- إذا '?' = C أو '!' = C نفذ:

اكتب '.'

- وإلا

اكتب C.

- لغة تربو باسكال:

```
PROGRAM POINT;
VAR
   C:CHAR;
BEGIN
    WRITELN('ENTER CHARACTER');
    READLN(C);
    WHILE C<> '*' DO
```

```
    IF (C='!') OR (C='?') THEN
      WRITE('.')
    ELSE
      WRITE(C);
READLN
END.
```


تمارين عامة

١- أذكر أي العبارات التالية صحيحة:

١- إذا كان C متحولاً محرفياً وكان '0' ≤ C ≤ '9' هذا يعني أن C هو رقم.

٢- إذا كـــان C متحولاً محرفياً وكان 'A' ≤ C ≤ 'Z' هذا يعني أن C هو حرف لاتيني كبير.

٣- 'a' < '9'

٤- ord ('0') = 0

٥- 'a' = 'A'

٦- ليكن c و d متحولين من نمط محرفي يكون c < d إذا وفقط إذا كان:

ord (c) < ord (d)

٧- succ ('0') = pred ('2')

٨- 'd' > 'a'

٩- ord ('q') < ord ('3')

١٠- k ord (chr(k)) من أجل أي عدد صحيح k

١١- c chr (ord (c)) من أجل أي محرف c

٢- احسب قيمة كل من العبارات التالية:

1- pred ('7')

2- ord ('5') – ord ('0')

3- ord (pred (chr (24)))

4- chr (ord ('5') + 1)

٣- حـــدد قيمـــة المتحول المحرفي d بعد إتمام العمليات في كل مـــن الحالتين التاليتين:

1- c := '+'; d := c;

2- c := '+'; d := 'c';

٤- اكتب برنامجاً يقوم بقراءة محرف، فإذا كان هـــذا المحرف '-' أو '+' يقـــوم بطباعة هذا المحرف ثلاث مـــرات، وإلا يقوم بطباعته مرتين ثم يكرر القراءة حتى يتم إدخال '.'.

الفصل السابع

أشكال أخرى لبنى التكرار في لغة تربو باسكال

٧-١- التعليمة التكرارية (الحلقة التكرارية) Repeat- Until:

في مرحلة ما من مراحل بناء برنامج ما قـــد نحتاج إلى تنفيذ حلقة مـــا عدداً غير محدد من المرات، وشرط إيقاف تنفيذ هـــذه الحلقة هـــو تحقق شرط معين سيتم فحصه قبل أو بعد كل تكرار لهذه الحلقة.

الشكل العام للتعليمة التكرارية هو بالشكل التالي:

```
Repeat
    < Statement >;
Until < Condition >;
```

إن < Statement > هي مجموعـــة الأوامر أو التعليمـــات البسيطة أو المركبـــة الواجب تنفيذهـــا ضمن الحلقـــة حتى يتحقـــق الشـــرط. حيـــث أن < Condition > هـــو شـــرط إنهاء الحلقة وهـــو تعبير بولياني. حيث لا يمكن الخروج من الحلقة إلا عند تحقق الشرط. أي عندما يصبح True.

٧-٢- خواص التعليمة التكرارية Repeat:

١- لا تحتاج كتلة التعليمة إلى تحديد البداية والنهاية بواسطة Begin و End. إذ أن بدايـــة التعليمة (الحلقة) محدد بكلمـــة Repeat ونهايتها محـــدد بكلمة

.Until

٢- تنفذ عبارات الحلقة Repeat مرة واحدة أولاً ومـــن ثم تفحص فيمـــا إذا كانـــت ستتوقف أو تتكرر مرة أخرى. حيث أن شكل التنفيذ يفيد أن الحلقـــة ستنفذ مرة واحدة على الأقل، ثم تأتي عملية الفحص.

٣- يستخدم في شرط الإنهاء متحول تحدد قيمته ناتج التعبير البولياني فيما إذا كان True أو False، وهـــذا المتحول يدعى متحول الحلقة. تتغير قيمة هذا المـــتحول حسب الحاجة، إن قيمة هـــذا المتحول تحدد متى تنتهي الحلقة. إذا بقيت قيمة هذا المتحول ثابتة ولم يتمكن من جعل قيمة الشرط True فإن تنفيذ الحلقة Repeat لن يتوقف أبداً.

٧-٣- أمثلة عامة:

مثال ١:

اكتب برنامجاً لحساب مجموع عددين صحيحين. يقوم البرنـــامج بسؤال المستثمر في كل مرّة عـــن الرغبة في متابعة العمل أم لا (أي إدخال عدديـــن آخرين أم لا).

الحل:

```
PROGRAM SUMNUM;
VAR
  N1,N2,C:INTEGER;
      A:CHAR;
BEGIN
 REPEAT
  WRITELN('INPUT N1,N2');
  READLN(N1,N2);
  C:=N1+N2;
  WRITELN('THE SUM IS=',C:10);
```



```
  WRITELN('DO YOU WANT TOCONTINUE(Y/N)?');
  READLN(A);
 UNTIL(A='N')OR(A='N');
END.
```

مثال ٢:

اكتب برنامجاً بلغة تربـــو باسكال يوجد (العدد) المقابل الروماني للعدد العشـــري المدخل. حيث يُطلب في البرنامج إدخال عدد بين الصفر والخمســـة آلاف لـــيوجد الرمز الروماني المقابل له ولا يخرج البرنامج إلا إذا أعطى عدداً خارج المجال السابق.

➢ مقدمة عن الأعداد الرومانية:

نرمـــز بـــ I ، V ، X ، L ، C ، D ، M لمجموعـــة مـــن الأعـــداد الرومانية حيث:

I := 1 – 4, V := 5 – 9

X := 10 – 49, L := 50 – 99

C := 100 – 499, D := 500 – 999

M := 1000 – 4999

حيث أن:

1) MXXIIII = 1024
2) MMMCXX = 3120
3) DCVIIII = 614
4) MMMMDCCCLXXVI = 4876
5) MMXXXXXVIIII = 2049

- البرنامج بلغة باسكال:

```
PROGRAM ROMAN_NUMBERS;
VAR
  I,N:INTEGER;
BEGIN
 REPEAT
  WRITELN('INPUT NUMBER >0 AND <5000');
  READLN(N);
  IF (N>0) AND (N<5000) THEN
   BEGIN
    WHILE N>=1000 DO
     BEGIN
      WRITE('M'); N:=N-1000
     END;
    IF N>=500 THEN
     BEGIN
      WRITE('D'); N:=N-500
     END;
    WHILE N>=100 DO
     BEGIN
      WRITE('C'); N:=N-100
     END;
    IF N>=50 THEN
     BEGIN
      WRITE('L'); N:=N-50
     END;
```



```
    WHILE N>=10 DO
     BEGIN
      WRITE('X'); N:=N-10
     END;
    IF N>=5 THEN
     BEGIN
      WRITE('V'); N:=N-5
     END;
    WHILE N>=1 DO
     BEGIN
      WRITE('I'); N:=N-1
     END;
    N:=N+1;
    WRITELN;
   END
  ELSE
    WRITELN
 UNTIL (N<=0) OR (N>5000)
END.
```

مثال ٣:

نفس المثال السابق ولكن عند إعطاء الشرط N = -1

الحل:

```
PROGRAM ROMAN_NUMBERS;
VAR
  I,N:INTEGER;
BEGIN
```

```
REPEAT
 WRITELN('INPUT NUMBER >0 AND <5000');
 READLN(N);


 WHILE N>=1000 DO
  BEGIN WRITE('M'); N:=N-1000 END;
 IF N>=500 THEN
  BEGIN WRITE('D'); N:=N-500  END;
 WHILE N>=100 DO
  BEGIN WRITE('C'): N:=N-100  END;
 IF N>=50 THEN
  BEGIN WRITE('L'): N:=N-50   END;
 WHILE N>=10 DO
  BEGIN WRITE('X'): N:=N-10   END;
 IF N>=5 THEN
  BEGIN WRITE('V'): N:=N-5    END;
 WHILE N>=1 DO
  BEGIN WRITE('I'); N:=N-1    END;
 WRITELN;
 UNTIL N=-1
END.
```

٧-٤- التعليمة التكرارية (الحلقة التكرارية) For- Do:

تستخدم حلقة For في لغة تربو باسكال عندما نعلم بشكل مسبق عدد مرات التكرار التي نريد فيها تكرار العمل. حيث تعتبر الحلقة For من أشهر أشكال الحلقات المستخدمة في لغات البرمجة المختلفة.

مثلاً من أجل دليل Index يأخذ قيمة بين < Exp1 > والقيمة < Exp2 >

كرر < مجموعة التعليمات >.

أي أنه ما دامت قيمة العدد Index أصغر من القيمة العليا لمجال التكرار< Exp2 > نفذ < مجموعة التعليمات >. يزداد العداد آلياً خطوة واحدة.

نكتب الآن الشكل العام لتعليمة For في لغة تربو باسكال، وهي من الشكل التالي:

For Index := < Exp1 > to< Exp2 > do

< Statements >

نكتب الآن بعض الخواص التي تتميز بها تعليمة For عن غيرها من التعليمات التكرارية.

٧-٥- خواص التعليمة التكرارية For:

١- يُسمى متحول الحلقة في حلقة For بمتحول التحكم وهو الذي يحدد عدد مرات التكرار وهو يمثل عداداً للحلقة، هذا المتحول يمكن أن ينتمي إلى نوع من أنواع المعطيات المرتبة لكن أكثر أنواع المعطيات المستخدمة لمتحول الحلقة هو النوع الصحيح.

٢- يجب الملاحظة هنا أنه لسنا بحاجة إلى تعليمة لتغيير قيمة متحول الحلقة For لأن البرنامج وبشكل تلقائي يغير قيمة متحول الحلقة بعد كل تكرار للحلقة.

٣- إذا كانت القيمة الابتدائية لمتحول الحلقة أكثر من القيمة النهائية لهذا المتحول فإن هذه الحلقة لن تنفذ.

٤- إذا كانت القيمة الابتدائية لمتحول الحلقة تساوي القيمة النهائية لهذا المتحول فإن هذه الحلقة سوف تنفذ مرة واحدة فقط.

٥- عند كتابة الحلقة For لا يكتب فاصلة منقوطة (;) بعد المميز المحجوز

do لأن ذلك سيجعل الحلقة For تنفذ عبارة فارغة.

٦- يمكن أن يكون عداد الحلقة For تنازلياً بدلاً مـــن كونـــه تصاعدياً وذلك باستبدال كلمة to بـــ Down to في تعليمة For لتصبح بالشكل التالي:

For Index := < Exp1 > Down to< Exp2 > do

< Statements > ;

٧- إن حلقة For في حالة الحلقة العكسية تنقص متحول الحلقة بمقدار رمـــز بعـــد كل تنفيذ للحلقة. ويجب أن تكون القيمة التي تأتي بعد Downto أصغر من القيمة الابتدائية لمتحول الحلقة.

في حالة كون القيمة النهائية لمتحول الحلقة العكسية أكبر من القيمة الابتدائية فإن الحلقة لن تنفذ أبداً. وإذا ما تساوت قيمتا متحول الحلقة الابتدائية والنهائية فإن الحلقة ستنفذ مرة واحدة فقط.

٨- يمكن وضع حلقتي For واحـــدة خارجية وواحدة داخلية، تتحكم حلقة For الخارجية بحلقة For الداخلية، وبالتالي يكون لدينا حلقات For متداخلة ضمن بعضها البعض. إن الحلقة الداخلية تُنفذ بشـــكل كامل في كل مرة ينتهي فيها تنفيذ الحلقة الخارجية.

٧-٦- أمثلة عامة:

مثال ١:

اكتب برنامجاً لاختيار أصغر عـــدد من مجموعة أعداد مـــدخلة، مؤلفة من n عدداً حقيقياً.

الحل:

من أجل كتابة البرنامج يلزمنا في البداية معرفة كل مما يلي:

١- عدد الأعداد المدخلة وهو عدد صحيح.

٢- عداد الحلقة وهو من النوع الصحيح.

٣- ممثل العدد المدخل وهو من النوع الحقيقي.

٤- ممثل العدد الأصغر وهو من النوع الحقيقي.

وعلية نكتب خوارزمية الحل بالشكل التالي:

- لغة الخوارزمية:

١- إقرأ n.

٢- إقرأ x.

٣- x min := x

٤- من أجل I = 2 وحتى n كرر:

- إقرأ x

- إذا x < x min نفذ:

x min := x

- البرنامج بلغة تربو باسكال:

```
Program Xmin;
Var
   N,I:integer;
    X:real;
Begin
   readln(N);
   readln(X);
   Xmin:=X;
   for I:=1 to N do
      begin
         readln(X);
         If X<Xmin then
```

```
        Xmin:=X;
    end;
    Writeln('Xmin=',Xmin:10:2);
Readln
End.
```

مثال ٢:

اكتب برنامجاً لكتابة أرقام المحارف ASCII لمجال من المحارف.

- البرنامج بلغة تربو باسكال:

```
PROGRAM ASCII;
VAR
  C1,C2,C:CHAR;
BEGIN
   WRITELN('ENTER TWO CHARACTERS');
   READLN(C1,C2);
   FOR C:=C1 TO C2 DO
   WRITELN(C,'      ',ORD(C));
   READLN
END.
```

ندخل المعطيات دون فراغ مثلاً CZ.

نتيجة تنفيذ البرنامج وفق المعطيات المدخلة هي بالشكل التالي:

```
ENTER TWO CHARACTERS
CZ
C      67
D      68
```



```
E      69
F      70
G      71
H      72
I      73
J      74
K      75
L      76
M      77
N      78
O      79
P      80
Q      81
R      82
S      83
T      84
U      85
V      86
W      87
X      88
Y      89
Z      90
```

مثال ٣:

اكتب برنامجاً لإيجاد جدول الضرب لـ m , n عدداً.

-البرنامج بلغة تربو باسكال:

```
PROGRAM MULTTABLE;
VAR
```

```
  I,J,N,M,K:INTEGER;
BEGIN
   WRITE('INPUT M=');READLN(M);
   WRITE('INPUT N=');READLN(N);
   FOR I:=1 TO M DO
    BEGIN
     WRITELN;
     FOR J:=1 TO N DO
      BEGIN
       K:=I*J;
       WRITELN(I,'*',J,'=',K);
      END;
     READLN
    END;READLN
END.
```

نتيجة تنفيذ البرنامج وفق المعطيات المدخلة هي بالشكل التالي:

```
INPUT M=9
INPUT N=9

1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
```



```
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18

3*1=3
3*2=6
3*3=9
3*4=12
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27

4*1=4
4*2=8
4*3=12
4*4=16
4*5=20
4*6=24
4*7=28
4*8=32
4*9=36
```

```
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45

6*1=6
6*2=12
6*3=18
6*4=24
6*5=30
6*6=36
6*7=42
6*8=48
6*9=54

7*1=7
7*2=14
7*3=21
7*4=28
7*5=35
7*6=42
7*7=49
7*8=56
7*9=63
```



```
8*1=8
8*2=16
8*3=24
8*4=32
8*5=40
8*6=48
8*7=56
8*8=64
8*9=72

9*1=9
9*2=18
9*3=27
9*4=36
9*5=45
9*6=54
9*7=63
9*8=72
9*9=81
```

مثال ٤:

اكتب برنامجاً لطباعة الشكل التالي:

```
a
a b
a b c
...
a b c ... x y z
```

-البرنامج بلغة تربو باسكال:

```
Program AB;
Var
  M,N:char;
Begin
   for M:='a' to 'z' do
    begin
      for N:='a' to M do
         write(N);
      writeln;
    end;
readln
End.
```

نتيجة تنفيذ البرنامج هي بالشكل التالي:

```
a
ab
abc
abcd
abcdc
abcdef
abcdefg
abcdefgh
abcdefghi
abcdefghij
abcdefghijk
abcdefghijkl
abcdefghijklm
abcdefghijklmn
abcdefghijklmno
abcdefghijklmnop
```



```
abcdefghijklmnopq
abcdefghijklmnopqr
abcdefghijklmnopqrs
abcdefghijklmnopqrst
abcdefghijklmnopqrstu
abcdefghijklmnopqrstuv
abcdefghijklmnopqrstuvw
abcdefghijklmnopqrstuvwx
abcdefghijklmnopqrstuvwxy
abcdefghijklmnopqrstuvwxyz
```

وهو المطلوب.

مثال٥:

اكتب برنامجاً يقوم بتوليد أعداداً عشـــوائية، والمطلوب إحصاء عـــدد مـــرات تكرار عملية التوليد العشوائي لعدد صحيح يقبل القسمة على 13.

المطلوب مـــن البرنامج هو توليد عدد صحيح عشـــوائي وفحص قابلية قسمة العدد على 13 فإذا تحقق الشـــرط فسيتوقف تنفيذ الحلقة وإلا سيزداد عدد مرات التوليد لعدد عشوائي حتى نحصل على عدد يقبل القسمة على العدد 13 .

إن العـــبارة Repeat-Until تضمن تكرار عمـــلية التوليد حتى يتحقق الشـــرط ونخرج من الحلقة.والمطلوب طباعة عدد مرات التوليد ليتحقق الشرط.

فـــي البداية لا بد مـــن تعريف متتالية توليد قيم عشوائيةRandomize .وأيضـــا تابع توليد قيم عشوائية Random الذي يطبق على Maxint الذي هو تـــابع مسبق التعريف في تربو باسكال أكبر قيمة يمكن أن يأخذها هي 32767 وإذا تـــم تطبـــيق ذلك على MaxLongint فإن أكبر قيمة يكمن أن يأخذها هي 2.147.483.647 .

وهو المطلوب.

-البرنامج بلغة تربو باسكال:

```
PROGRAM RepeatDemo;
VAR N,I:INTEGER;
BEGIN
 RANDOMIZE;
 I:=0;
 REPEAT
   N:=RANDOM ( MAXINT);
   I:=I+1;
 UNTIL  N MOD 13 = 0;
 WRITELN(I:5);
READLN;
END.
```

مثال تعليمي:

اكتب برنامجاً لإخراج متتالية الرموز التالية:

a bb ccc dddd ... zzz...z

-البرنامج بلغة تربو باسكال:

```
Program AB;
Var
  M:char;
  Q,I:integer;
Begin
   for M:='a' to 'z' do
```



```
    begin
      Q:=ord(M)-ord('a')+1;
      for I:=1 to Q do
         write(M);
      write(' ');
    end;
readln
End.
```

ناتج تنفيذ البرنامج بالشكل التالي:

a bb ccc dddd eeeee fffffff ggggggg hhhhhhhh iiiiiiiii
jjjjjjjjjj kkkkkkkkkkk lllllllllll mmmmmmmmmmmmm
nnnnnnnnnnnnnn ooooooooooooooo pppppppppppppppp
qqqqqqqqqqqqqqqq rrrrrrrrrrrrrrrr ssssssssssssssssss
ttttttttttttttttt uuuuuuuuuuuuuuuuuuuu vvvvvvvvvvv
vvvvvvvvvvv wwwwwwwwwwwwwwwwwwwww
xxxxxxxxxxxxxxxxxxxxxxxx yyyyyyyyyyyyyyyyyyyyy
yyy zzzzzzzzzzzzzzzzzzzzzzzzzz

توجـد عدة طرق لحل هذا البرنامج غير الحل السابق نذكر منها إحدى عشر طريقة قام بكتابتها طلاب السنة الأولى رياضيات عام ٢٠٠٢ حيث كان هذا السؤال سؤال الدورة الثانية ٢٠٠٢ .
نذكر عددا من هذه الطرق تشـجيعا للطلاب من الدورات

الأخـــرى لكي يعمدوا إلى إيجاد طرقهم الخاصة . ننوه إلى أنه في البرمجة لا توجد طريـقة وحيدة و إنمـــا توجد عدة طرق لكتابة البرامـــج ، وهي دعوة لكي يعمل الطلاب إلى اســتخدام كل أوامر اللغة وابتكار طرق جديدة لحل البرامج.

طرق حل البرنامج :

-١

```
Program AB;
Var
  M:char;
  Q,I:integer;

Begin
   for M:='a' to 'z' do
    begin
      Q:=ord(M)-ord('a')+1;
      for I:=1 to Q do
         write(M);
      write(' ');
    end;
readln
End.
```

-٢

```
Program AB;
Var
```



```
  I:char;
  J:integer;
Begin
   for I:='a' to 'z' do
    begin
     for J:=1 to (ord(I)-96) do
         write(I);
     write(' ');
    end;
readln
End.
```

-٣

```
Program AB;
Var
  I,J:integer;

Begin
   for I:=1 to 26 do
    begin
     for J:=1 to I do
         write(chr(I+96));
     write(' ');
    end;
readln
End.
```

-٤

```
Program AB;
```

```
Var
  I,J:integer;
Begin
   for I:=1 to 26 do
    begin
     J:=1;
     while J<=I do
      begin
         write(chr(I+96));
         J:=J+1;
      end;
     write(' ');
   end;
readln
End.
```

-٥

```
Program AB;
Var
    X:char;
  I,Y:integer;
Begin
   for X:='a' to 'z' do
    begin
      Y:=ord(X);
      for I:=1 to Y-96 do
       write(X);
      write(' ');
```



```
   end;readln
End.
```

-٦

```
Program AB;
Var
  I,J:integer;
Begin
   for J:=97 to 122 do
    begin
      I:=97;
      while I<=J do
       begin
        write(chr(J));
        I:=I+1;
       end;

      write(' ');
    end;
readln
End.
```

-٧

```
Program AB;
Label 3;
Var
  I:char;
  J,Y:integer;
Begin
```

```
Program AB;
Var
  I,J,N:integer;
Begin
   N:=1;
   for I:=97 to 122 do
    begin
      for J:=1 to N do
        write(chr(I));
      write(' ');
      N:=N+1;
    end;
readln
End.
```

-١٠

```
Program AB;
Var
  I:char;
  N:integer;
Begin
   for I:='a' to 'z' do
    begin
      N:=ord(I);
      while N>=97 do
```



```
    for I:='a' to 'z' do
     begin
       J:=1;
      3:write(I);
       while J<(ord(I)-96) do
        begin
          J:=J+1;
          goto 3;
        end;
       write(' ');
    end;
readln
End.
```

-٨

```
Program AB;
Label 3;
Var
  N:char;
  I,J:integer;
Begin
   for N:='a' to 'z' do
    begin
      I:=ord(N);
      for J:=97 to I do
        write(N);
      write(' ');
    end;
readln
End.
```

-٩

```
      begin
        write(I);
        N:=N-1;
      end;
     write(' ');
   end;
readln
End.
```

-١١

```
Program AB;
Var
  A:char;
  I,J:integer;
Begin
   I:=1;
   A:='a';
   write(A);
   write(' ');
   repeat
    I:=I+1;
    A:=succ(A);
    for J:=1 to I do
      write(A);
    write(' ');
   until I=26;
readln
End.
```


تمارين عامة

١) اكتب برنامج لإدخال متتالية مـــن الرموز تنتهي بالرمز "$" ،احســـب عدد مرات تكرار الأحرف الكبيرة فيها.

٢) اكتب برنامجاً لإدخال متتالية مـــن الرموز تنتهي بالرمز / . / وحساب عدد مرات تكرار الحرف .I فيها.

٣) اكتب برنامجاً لطباعة عدد مـــا مدخل يتزايد 5 حتى المئة. باستخدام حلقـــة Repeat-Until.

٤) اكتب برنامجاً لحساب مجموع السلسلة التالية:

$$S=\sum_{i=1}^{50}\frac{1}{2_i}$$

باستخدام حلقة Repeat-Until.

٦) اكتب برنامجاً لحساب قيمة السلسلة التالية باستخدام حلقة For-do

$S=1^2+2^2+3^2+...+n^2$.

٧) اكتب برنامجاً لحساب قيمة التابع !n باستخدام حلقة For-do.

٨) اكتب برنامجاً لحساب قيمة السلسلة:

$$S=\sum_{i=1}^{20}\frac{1}{3_i}$$

باستخدام حلقة For-do.

٩) اكتب برنامجاً لطباعة عدد مـا مدخل بتزايد 5 حتى المئة باستخدام حلقـة For-do.

١٠) اكتب برنامجاً لحساب قيمة التابع !n باستخدام حلقة For-do حيث:

$n! = n\,(n-1)\,(n-2)\,\ldots\,3.2.1$

المطلوب حساب قيمة !n وبشكل تنازلي.

١١) اكتب برنامجاً لحسـاب مجموع الأعـداد من ١ وحتى n باستخدام حلقـة For-do.

١٢) اكتب برنامجاً لحساب مجموع n عدد معطى، باستخدام حلقة For-do.

١٣) اكتب برنامجاً لحساب المتوسط الحسابي لـ n عدد معطى.

١٤) اكتب برنامجاً لإخراج متتالية الرموز التالية:

abbcccdddd ... zzz...z

١٥) اكتب برنامجاً لإخراج متتالية الرموز التالية:

z yy xxx ... bb a

١٦) اكتب برنامجاً لإيجاد مجموع السلسلة التالية:

$$S := 1+\frac{1}{2!}+\frac{1}{3!}+\ldots+\frac{1}{n!}$$

١٧) ليكن A و B متحولين مـن نمـط بولياني. المطلوب اكتب جـدول الـ AND وجدول الـ OR باستخدام حلقة For.

الفصل الثامن

بنية التحكم الشرطية CASE
والنمط المعدود

٨-١- مقدمة:

تمتلك لغة تربـو باسكال نوعاً آخر من بنى التحكم، وهو نوع مهم جداً، هو عندما يكون ناتج الفحص قيمة عددية، أو عندما يكون له احتمالات متعددة.

أعطتنا لغة باسكال عبارة Case التي سهلت عملية الفحص تلك، أما لغة تربو باسكال فقد جعلت الأمر أسهل وذلك بإضافة ميزات لهذه العبارة.

تزودنا عبارة Case إمكانية الاختيار، ذلك اعتماداً على قيمـة المتحول الذي يجب أن ينتمي إلى أحد أنواع المعطيات المرتبة.

المقصود بأحد المعطيات المرتبة هو أحد أنواع الأنماط التالية:

Enumerated , Char , Boolean , Integer.

٨-٢- التعليمة الشرطية Case:

إن الشكل العام للتعليمة الشرطية Case هو:

```
Case A of
        A1 : < Statement1 > ;
        A2, A3 : < Statement2 > ;
        B1 .. B2 : < Statement3 > ;
        B3 .. B4, A4 : < Statement4 > ;
```

```
Else
        < Statement5 > ;
End;
```

إن القيم A1, A2, A3 هي قيم للمتحول A، وباختيار أحدهـا يتم تنفيذ التعليمة المقابلـة فقط. إن B1 .. B2 يعني المجال الواقع بيـن القيم B1 و B2. عند وقوع قيمة A في المجال السابق عندها يتم تنفيذ التعليمة أو مجموعة مـن التعليمات المقابلة فقط. وإذا ما أخذت A قيمة A4 أو قيمة في المجال B3 .. B4 عندها يتم تنفيذ التعليمة أو مجموعة التعليمات المقابلة. وفيما عدا ذلك يتـم تنفيذ مجموعة التعليمات بعد كلمة Else.

٨-٣- خواص التعليمة الشرطية:

١- إن تجمع التعليمـات المشـمول بالتعليمـة Case لا يُبتدأ بـ Begin لأن البداية هي بعد سـطر Case مباشرة، لكن لا بد من تحديد نهاية الكتلة بـ End;. أي أن Case لا تبتدئ بـ Begin وتنتهي بـ End;.

٢- تخضع كتلة التعليمات التي تنفذ في كل مدخل من مداخل Case إلى قواعد تعريف كتلة التعليمات في باسكال أي البداية والنهاية تحددان بكلمتي Begin و End; ويمكـن الاستغناء عنها فقط في حال كون الكتلة مؤلفة مـن تعليمة واحدة فقط.

٣- إن عبارة Else ضمن تعليمـة Case هي اختياريـة أي عنـد الحاجـة والضرورة، فهو سـطر اختياري في لغة تربـو باسكال، علماً أن في لغـة باسكال القياسية غير مسموح به. إن الفاصلة المنقوطة قبل Else في تعليمـة Case هي ضرورية.

٤- إن مداخل تنفيذ مجموعة مـن التعليمات يمكن أن لا يقتصر على قيمـة

واحدة بل قد يكون لأجل أكثر من قيمة مثل A2, A3 أو مجال من القيم مثل B1 .. B2.

٥- إن التحكم بعبارة Case يجري وفق قيمة المتحول (أو قيمة التعبير) الذي يأتي مباشرة بعد المميز المحجوز Case، وهـذه القيمة يمكن أن تكون إحدى القيم الموجودة ضمن لوائح القيم في عبارة Case، وفي حال وجود تعبير بعد المميز المحجوز Case يحسب هـذا التعبير أولاً ثم على ضوء ناتج الحساب هذا ستنفذ عبارة Case.

٦- يمكن أن تتألف لائحة القيم من قيمة واحدة أو أكثر، مفصولة عن بعضها بفاصلة (,)، ويمكن أن تكون مجالاً جزئياً من القيم محدداً بقيمته الابتدائية، ويأتي بعده نقطتان متتاليتان، ومن ثم قيمته النهائية، إن هذه الإمكانية متوفرة فقط في لغة تربو باسكال، ولم تكن موجودة في لغة باسكال القياسية.

٧- يمكن بناء عبارات Case متداخلة فيما بينها.

٨-٤- أمثلة عامة:

مثال ١:

اكتب برنامجاً لإدخال عددين صحيحين ورمز عملية حسابيـة ثم طباعة ناتج تنفيذ هذه العملية. مستوضحاً رغبة المستثمر في المتابعة من أجـل عددين آخرين ورمز عملية أخرى في نهاية كل تنفيذ.

الحل:

```
Program OperationNum;
Var
    N1,N2:real;
  operator:char;
       A:char;
```

```
Begin
  repeat
    writeln('Input two Numbers');
    readln(N1,N2);
    writeln('Input operation');
    readln(operator);
    case operator of
        '+':writeln(N1+N2:8:2);
        '-':writeln(N1-N2:8:2);
        '*':writeln(N1*N2:8:2);
        '/':writeln(N1/N2:8:2);
    end;
    writeln('do you want to continue(Y/N)?');
    readln(A);
  until(A='N')or(A='n');
End.
```

مثال ٢:

اكتب برنامجاً لطباعة اسم الشهر اعتماداً على رقمه.

الحل:

```
PROGRAM MONTHES;
VAR
  M:INTEGER;
BEGIN
  READLN(M);
```



```
  CASE M OF
    1:WRITELN('JANUARY');
    2:WRITELN('FEBRUARY');
    3:WRITELN('MARCH');
    4:WRITELN('APRIL');
    5:WRITELN('MAY');
    6:WRITELN('JUNE');
    7:WRITELN('JULY');
    8:WRITELN('AUGUST');
    9:WRITELN('SEPTEMBER');
    10:WRITELN('OCTOBER');
    11:WRITELN('NOVOMBER');
    12:WRITELN('DECEMBER');
  END;
READLN
END.
```

مثال ٣:

اكتب برنامجاً لطباعة نوع المحرف في أسكي المقابل لرقمه.

الحل:

```
PROGRAM CHRACTER_FROM_DIGIT;
VAR
  I:INTEGER;
BEGIN
  WRITELN('ENTER THE NUMBER');
  READLN(I);
```

```
  CASE I OF
    1..31:WRITELN('CONTROL CHARACTER');
      32:WRITELN('SPACE');
   48..57:WRITELN('DIGIT');
   65..90:WRITELN('CAPITAL LETTER');
  97..122:WRITELN('SMALL LETTER');
     ELSE
        WRITELN('MARK');
  END;
READLN
END.
```

مثال ٤:

اكتب برنامجاً يحسب ضريبة راتب مـــا ويطبع الراتب والضريبة وفق التوزيع التالي:

الضريبة Tax	الراتب Value
3 %	[1999]
4 %	[2000 - 2999]
5 %	[3000 - 5999]
6 %	≤6000

حيث يطلب البرنامج إدخال الراتب ليحسب الضريبة ولا يخرج البرنامج إلا عند إعطاء قيمة الراتب صفر.

الحل:


```
PROGRAM TAX_VALUE;
VAR
  VALUE,CATEGORY:INTEGER;
      RATE,TAX:REAL;
BEGIN
   WRITELN('INPUT THE VALUE');
   READLN(VALUE);
   WHILE VALUE>0 DO
   BEGIN
     CATEGORY:=VALUE DIV 1000;
     IF CATEGORY>6 THEN
        CATEGORY:=6;
     CASE CATEGORY OF
       0,1:RATE:=0.03;
         2:RATE:=0.04;
     3,4,5:RATE:=0.05;
         6:RATE:=0.06
     END;
     TAX:=VALUE*RATE;
     WRITELN('VALUE':10,'TAX':10);
     WRITELN(VALUE:10,TAX:10:2);
     READLN(VALUE)
   END;
END.
```

٨-٥- النمط المعدود Enumerated:

إن بعض بنى التحكم التي درسناها سابقاً تتعامل مـــع الأنماط المتقطعة. فمثلاً تعليمة For تأخذ عداداً مـــن نمط متقطع، ومتحول تعليمة Case هو نمط متقطع أيضاً.

نلاحـــظ في المثال الذي كتبنا فيه اسم الشهر اعتماداً على رقمه أننا قمنا بترميز الأشير بالأعداد مـــن ١ وحتى ١٢. ثم أظهرنا سلسلة المحارف المناسبة لرقم الشهر.

يمكن التعامل مع الأشهر كقيم دون الحاجة إلى ترميزها وذلك باستخدام الـــنمط المعـــدود (Enumerated Type) حيث نعرف نمطاً ونســـميه Months يتضمن جميع الأشهر.

يُعرف النمط بكلمـــة المفتاح Type، وتذكر اســـم النمط ثـــم مفرداته. والتصريح عن نمط جديد بالكلمة Type يجب أن يســـبق عبارات التصريح عن المتحولات في أول البرنامج.

```
Type
Months = (January, February, March, April, May, June, July,
August, September, October, November, December);
```

يتم التصريح عن متحول من نمط Months مثلاً I بالشكل:

```
Var I: Months
```

حيث يتم التعامل مع I كأي متحول آخر، إذ يمكن إسناد قيمة إليه مـــن بين القيم المذكورة في نمط تعريفه. مثلاً يمكن كتابة:

```
L := June
```

حيث يمكن فحص قيمته ضمن شرط التعليمة الشـــرطية أو التعليمـــة التكرارية. مثلاً:

```
Fa I := January To December do
```

أو مثلاً:


```
Case I of
    January : Writeth ('January');
    .....
    December : Writeth ('December');
End;
```

٨-٦- العمليات والتوابع المعرفة على النمط المعدود:

إن النمط المعدود هو نمط مرتب فعمليات المقارنـــة معرفة لأي قيمتين وذلك لوجود تقابل بين القيم المعرفة في النمط وبين مجموعة الأعداد الصحيحة الموجودة بين 0 و(عدد القيم – ١).

إن جميع التوابع التي عرفناها سابقاً معرفة هنـــا مثل الـــ Ord و Succ و Pred. إن تابع الترتيب يعطي ترتيب القيمة ضمن مجموعة القيم المعرفة في النمط.

مثلاً:

```
Writeth (Ord (May))
```

يعطي القيمة ٤.

٨-٧- أمثلة عامة:

مثال١:

اكتـــب برنامجاً لطباعة اسم الشهر وذلك بمعرفة رقمـــه مستخدماً النمط Type لتعريف الأشهر.

الحل:

```
PROGRAM MONTHES_type;
TYPE
 MONTHES=(JANUARY,FEBRUARY,MARCH,APRIL,MAY,
          JUNE,JULY,AUGUST,SEPTEMBER,OCTOBER,
                              NOVOMBER,DECEMBER);
VAR
     M:MONTHES;
     A:1..12;
BEGIN
  WRITELN('INPUT MONTH NUMBER');
  READLN(A);
 FOR M:=JANUARY TO DECEMBER DO
  IF ORD(M)+1=A THEN
  CASE M OF
    JANUARY  :WRITELN('THE MONTHE IS JANUARY');
   FEBRUARY :WRITELN('THE MONTHE IS FEBRUARY');
     MARCH   :WRITELN('THE MONTHE IS MARCH');
      APRIL   :WRITELN('THE MONTHE IS APRIL');
      MAY     :WRITELN('THE MONTHE IS MAY');
       JUNE   :WRITELN('THE MONTHE IS JUNE');
       JULY   :WRITELN('THE MONTHE IS JULY');
    AUGUST  :WRITELN('THE MONTHE IS AUGUST');
  SEPTEMBER:WRITELN('THE MONTHE IS SEPTEMBER');
   OCTOBER  :WRITELN('THE MONTHE IS OCTOBER');
```



```
   NOVOMBER :WRITELN('THE MONTHE IS NOVOMBER');
    DECEMBER :WRITELN('THE MONTHE IS DECEMBER');
 END;
READLN
END.
```

مثال٢:

اكتب برنامجا يبين أي يوم من أيام الإسبوع هو اليوم الأكثر ورودا في السنة علما أن السنة غير كبيسة(أي عدد أيامها ٣٦٥)ورأس السنة يبتدئ يوم الأربعاء.

```
Program MaxDay;
Type
     weekday=(sat,sun,mon,tues,wend,thur,fri);
     day=array[weekday] of integer;
Var
     MaxD,D:weekday;
     ValueD:day; K:integer;
Begin
     for D:=sat to fri do
       ValueD[D]:=0;
     for K:=1 to 365 do
         case K mod 7 of
                0:ValueD[wend]:=ValueD[wend]+1;
                1:ValueD[thur]:=ValueD[thur]+1;
                2:ValueD[fri]:=ValueD[fri]+1;
```

```
            3:ValueD[sat]:=ValueD[sat]+1;
            4:ValueD[sun]:=ValueD[sun]+1;
            5:ValueD[mon]:=ValueD[mon]+1;
            6:ValueD[tues]:  ValueD[tues]+1;
        end;
    K: =ValueD[sat];
    MaxD:=sat;
    for D:=sun to fri do
        if K < ValueD[D] then;
            begin
                K:=ValueD[d];
                MaxD:=D;
            end;
        case MaxD of
            sat:writeln('saturday');
            sun:writeln('sunday');
            mon:writeln('monday');
            tues:writeln('tuesday');
            wend:writeln('wendsday');
            thur:writeln('thurthday');
            fri:writeln('friday');
        end;
    readln
    End.
```


تمارين عامة

١- اكتب برنامجا يطبع أحد أيام الأسبوع وذلك عند اختيار الرقم الموافق.

٢- اكتب ناتج تنفيذ البرنامج التالي:

```
Program DigitCase;
var
  D:integer;T:boolean;
Begin
   readln(D);
   case D of
   case D of
      2,3,5,7:begin
            T:=true;D:=succ(D);
            end;
   0,1,4,6,8,9:T:=false;
         else T:=true;
    end;
   writeln(T,D);
readln
end.
```

وذلك عند إعطاء قيم الدخل في كل مرة بالشكل التالي:

١- 3 . ٢- 6 . ٣- 5 . ٤- 44 .

لكن عند العمل مـــع نمط معطيات مركبة مثل Array فإن الأمر سوف يختلف وذلك لأن أنواع المعطيات هذه تبنى من كتل، وهذه الكتل يمكن أن تكون بحد ذاتها أنواع معطيات أخرى.

٩-٢- المصفوفات:

تسمح الكثير من لغات البرمجة بتعريف متحولات من نمط مصفوفة. إن المصفوفة هي بنية معطيات مركبة مـــن تجميع عناصر من نفس النمط، يمكن الوصول إلى عنصر فيها أو تحديد كل عنصر منها بأدلة.

إن المصفوفة (النسق أو الصف) Array عبارة عن متحول يحتوي عدة قيم متجمعة فيما بينها وتنتمي إلى نـــوع معطيات واحـــد. إن هذه القيم تدعـــى بحجرات النسق (Cell). تملك الحجـــرات دليـــلاً (Index) مرتبطاً بها، وعـــدد حجرات (النسق أو الصف) المصفوفـــة (نذكر هنا أن المصفوفة أحاديـــة البعد يمكن أن تسمى نسقاً أو صفاً) يجب أن يكون محدداً وثابتاً في لغة باسكال.

إن دلـــيـــل المصفوفـــة يجب أن يأخذ قيماً ضمن حدود المصفوفة، أي ضمن حدود عدد حجرات المصفوفة.

٩-٣- التصريح عن متحول من نمط مصفوفة أحادية البعد:

إن المصفوفة أحادية البعد هي جدول من ســـطر واحد، أو عمود واحد، أو يمكن أن نسميها شعاعاً. يتم التصريح عن متحول من نمط مصفوفة أحاديـــة البعد بلغة باسكال بالشكل:

; <نمط> of <ثابت ٢> .. <ثابت ١> Array :<اسم المتحول>

حيث أن <ثابت ١> و <ثابت ٢> هـــا قيم مـــن نمط متقطع. أي يمكن أن يكون من أحد الأنماط Enumerated ، Boolean ، Char ، Integer.

أما <نمط> فهو أي نمط عرفناه حتى الآن.

إن التصريح عن متحول بهذا الشكل يعرف شعاعاً من النمط <نمط> وأدلته في المجال <ثابت ٢> .. <ثابت ١> وحجمه (عدد عناصره) يساوي طول المجال بين الثابتين.

أمثلة:

1) A: Array [1 .. 5] of Integer;

إن A هو شعاع من خمسة عناصر كل عنصر منها هو مـــن نمط صحيح مثلاً A [3] هو العنصر الثالث فيها.

2) AB: Array ['a' .. 'z'] of Integer;

إن AB هو شعاع من 26 عنصر نمطهـــا صحيح لكن الأدلـــة فيها هي أحرف حيث أن AB ['B'] هو العنصر الثاني فيها.

3) C: Array [-5 .. 5] of Real;

إن C هو شعاع من أحـــد عشر عنصراً نمطها Real. حيث أن C [-4] هـــو العنصر الثاني فيها.

مما تقدم نستطيع أن نستنتج بعض الملاحظات الهامة على المصفوفات.

٩-٤- ملاحظات على المصفوفات:

١- ينبغي التمييز دائماً بين الحجرة وبين القيمة المخزنة في هـــذه الحجرة. إذ أن دليل الحجرة هو جزء من اسم الحجرة. مثلاً A [5] يمثل موضعاً محدداً من الذاكرة يحتوي مثلاً على عـــدد صحيح، أي القيمة المخزنـــة في هـــذه الحجرة.

بكلام آخر إن A [5] تمثل موضعاً في الذاكرة يحتوي على قيمة مـــن النوع الصحيح.

٢- يتم تخزين محتويات حجرات المصفوفة وفق تتابع مـن حجرات الذاكرة، حيـث يتـم تخصيص الحجم الكافي مـن الذاكرة لكل حجرة مـن حجرات المصفوفة. وبشكل عام إن قيمة أدلـة المصفوفة لا تخزن في الذاكرة وإنمـا تخزن قيـم حجرات المصفوفة. أما قيم أدلـة المصفوفة فيستخدمها المترجم Compiler في حسابه لعناوين مواضع الذاكرة المخصصة لحجم المصفوفة.

٣- إن المصفوفة تتألف مـن حجرات تحتوي على معلومات تنتمي إلى نـوع معطيات معين. يُحدد حجـم المصفوفة بنوع المعطيات الأساسـي لها وبعد عناصر هذه المصفوفة وحجم المصفوفة يجب أن لا يتجاوز 64 كيلو بايت.

٤- يمكن استخدام أي نوع معطيات مرتب كدليل لحجر المصفوفة عـدا نوع المعطيات عـدد صحيح طويـل Longint. إذ تحدد القيمـة الابتدائية لدليل المصفوفـة، ثم تتبعها نقطتان متتاليتان، ومـن ثـم القيمـة النهائيـة لدليل المصفوفة.

٥- لا بد أولاً وقبل استخدام مصفوفة في برنامج من قراءة عناصرها أو إسناد قيم إليها. مثلاً يعبر عـن العنصـر رقم i في المصفوفـة السطر A بالشكل A [i].

يمكـن للدليل i أن يعطى بدلالة ثوابت ومتحولات أخرى على أن تكون قيمة هذه العبارات موافقة لنوع هذا الدليل ومجال قيمتـه، فمثلاً لا يمكن أن يكتب A [m/3] مهما كانت قيمة m لأن عملية القسمة ستعطي عدداً حقيقياً، يختلف عن نوع الدليل i الذي هو عدد صحيح.

٩-٥-أمثلة عامة:

٩-٥-١- مثال ١:

نفرض أنه لدينا 20 عدداً صحيحاً مخزنة في المتتالية:

X1, X2, .. , X20

ونريد طباعة هذه المتتالية بشكل عكسي.

الحل:

- البرنامج بلغة تربو باسكال:

```
Program Series;
Var
  X:array[1..20] of integer;
  I:integer;
Begin
   for I:=1 to 20 do
     read(X[I]);
   for I:=20 downto 1 do
     write(X[I]:4);
readln;
readln
End.
```

٩-٥-٢- مثال ٢:

اكتب برنامجا لبناء وكتابة متتالية A_N عناصرها معرفة بالشكل:

$$a_1 = 1 \; , \; a_{i+1} = a_i^2 + a_i/2 \; ; \; i = 2, 3, \ldots$$

والحد الأعظمي لعدد الحدود هو 30.

الحل:

عند حـل مسائل على المصفوفات والتي يختلف طولها من تنفيذ لآخر، يفضل أن يحجز مجال أعظمي لطول المصفوفـة وتحديـد الكميـة اللازمـة للاسـتخدام من هذه المصفوفة مـن داخل البرنـامج وبقراءة متحول ممثل لهذا

الطول.

- البرنامج بلغة تربو باسكال:

```
Program BildSeries;
Var
  A:array[1..30] of real;
  N,I:integer;
Begin
   readln(N);
   A[1]:=1;
   for I:=1 to N do
     A[I+1]:=A[I]*A[I]+A[I]/2;
   for I:=1 to N do
     writeln(A[I]:10:1);
readln
End.
```

٩-٥-٣- مثال ٣:

اكتب برنـــامجاً يسمح بحساب أصوات ٤ مـــن المرشحين. حيث تصل نـــتائج الاقتراع على شكل دخل عددي، كل عدد على ســـطر يمثل رقم المرشح المنتخب ينتهي الدخل بعدد مميز وليكن الصفر.

الحل:

- البرنامج بلغة تربو باسكال:

```
PROGRAM ELECTION;
VAR
```



```
  E:INTEGER;
  C:ARRAY[1..4] OF INTEGER;
BEGIN
   FOR E:=1 TO 4 DO
     C[E]:=0;
   READLN(E);
   WHILE (E<>0) DO
     BEGIN
       IF (1<=E) AND (E<=4) THEN
         C[E]:=C[E]+1;
       READLN(E);
     END;
   FOR E:=1 TO 4 DO
     WRITELN('C',E,'=',C[E]);
READLN
END.
```

٩-٥-٤- مثال ٤:

اكتب برنامجاً لحساب قيمة المقدار

$x_1y_1+x_3y_3+\dots+x_9y_9 / x_2y_2+x_4y_4+\dots+x_{10}y_{10}$

في حالتي الإدخال التاليتين:

1) $x_1, x_2, \dots, x_{10}, y_1, y_2, \dots, y_{10}$

2) $x_1, y_1, x_2, y_2, \dots, x_{10}, y_{10}$

الحل:

- البرنامج بلغة تربو باسكال:

```
Program Series;
```

```
Var
  X,Y:array[1..10] of real;
  I:integer;
  S1,S2:real;
Begin
   for I:=1 to 10 do
     read(X[I]);
   for I:=1 to 10 do
     read(Y[I]);
   S1:=0;S2:=0;
   for I:=1 to 5 do
    begin
      S1:=S1+X[2*I-1]*Y[2*I-1];
      S2:=S2+X[2*I]*Y[2*I];
    end;
   writeln(S1/S2:10:2);
readln;
readln
End.
```

- في الحالة الثانية تتم القراءة بالشكل:

For I := 1 To N do Read (x [I] , y [I]);

وهو المطلوب.

٩-٦- التصريح عن مصفوفة ثنائية البعد:

نحتاج في الكثير من المسائل العلمية لتشكيل قائمة، أو جدول بأكثر من بعد واحد. ففي مثال المصفوفات نحتاج إلى تعين دليل السطر ودليل العمود أي مصفوفة ببعدين ولكي نحصل على قيمة عنصر في المصفوفة علينا تعيين

قيمتين للدليلين دليل السطر ودليل العمود. يتم التصريح عن متحول من نمط مصفوفة ثنائي البعد بالشكل التالي:

< اسم المتحول > : Array [<A1> .. <A2> , <B1> ..

<B2>] of < نمط >;

حيث أن التصريح هو لمصفوفة مؤلفة من <A2> .. <A1> سطراً ومن <B1> <B2> .. عموداً وكل عنصر من عناصرها هو من النمط المحدد < نمط >.

مثال:

Q : Array [1 .. 6, 1 .. 6] of Integer

إن Q هي مصفوفة مربعة حجمها 6 × 6 وكل عنصر من عناصرها هو عدد صحيح.

٩-٧- أمثلة عامة:

٩-٧-١- مثال ١:

مثلث الكرخي أو مثلث باسكال (أكتشف المثلث في عصر أقدم من عصر الرياضي الفرنسي باسكال الذي سميت لغة البرمجة Pascal نسبة إليه، واكتشف في مخطوطات أقدم من عصر الرياضي باسكال، هذه المخطوطات تعود للعالم الرياضي العربي محمد الكرخي).

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
```

يستخدم هذا المثلث في الكثير من العمليات الحسابية، على سبيل المثال يمكن أخذ أمثال كثير الحدود "(x + 1) منه. فهي عبارة عن مجموع لحدود قوى x من 0 وحتى n وأمثالها هي السطر n من مثلث باسكال.

مثلاً:

$$(1+x)^2 = 1+2*x+1*x^2$$
$$(1+x)^3 = 1+3+x+3*x^2+1*x^3$$

يمكن إنشاء المثلث بوضع أصفار في المصفوفة كلها مـــا عـــدا العمود الأول حيث توضع قيمة تساوي 1 ثم نحسب العنـــاصر الأخرى وفـــق العلاقة التالية:

قيمة العنصر = مجموع قيم عنصرين من السطر السابق (عنصر العمود نفسه + عنصر العمود السابق).

إذا رمزنا على المصفوفة بالمتحول P. عندها يكون العنصر في السطر i والعمود j هو P_{ij} أي:

$$P_{i,j} = P_{i-1,j} + P_{i-1,j-1}$$

وذلك من أجل كل العناصر ما عدا السطر 1 والعمود 1.

- البرنامج بلغة تربو باسكال:

```
PROGRAM PASCAL_TRIANGLE;
VAR
  P:ARRAY[1..5,1..5] OF INTEGER;
  I,J:INTEGER;
BEGIN
   FOR I:=1 TO 5 DO
     FOR J:=1 TO 5 DO
        P[I,J]:=0;
   FOR I:=1 TO 5 DO
     P[I,1]:=1;
   FOR I:=2 TO 5 DO
     FOR J:=2 TO 5 DO
```



```
        P[I,J]:=P[I-1,J]+P[I-1,J-1];
   FOR I:=1 TO 5 DO
    BEGIN
     FOR J:=1 TO 5 DO
       WRITE(P[I,J]:4);
     WRITELN;
    END;
READLN
END.
```

مما تقدم نستطيع أن نكتب أن البرنـــامج بهذا الشكل يقوم بالحساب على كل المصفوفة ونحن نعلم أن المصفوفـــة مثلثية أي أن العناصر الواقعـــة فـــوق القطر الرئيسي معدومة أي مساوية للصفر لهذا لا حاجة إلى كتابتها. لذلك يمكن إيقاف حلقة الحساب والكتابة عند عناصر القطر وهي العناصر التي يكون فيها دليل السطر مساوياً لدليل العمود. مثلاً العنصر p [i,i] هو عنصر السطر رقم i والعمود رقم i وهو عنصر القطر. حيث يمكن استبدال كل i بـــ 5 في سطري البرنامج للحصول على المطلوب.تصبح حلقة الطباعة بالشكل التالي:

```
FOR I:=1 TO 5 DO
 BEGIN
  FOR J:=1 TO I DO
    WRITE(P[I,J]:4);
  WRITELN;
 END;
```

٩-٧-٢- مثال ٢:

بفرض A مصفوفة مؤلفة مـــن M سطر و N عمود عناصر المصفوفة

أعداد حقيقية. اكتب برنامجاً لطباعة عناصر هذه المصفوفة بعد طرح العمـــود الأخير من جميع الأعمدة الأخرى، علماً بأن البعد الأعظمي لأدلة المصفوفة هو 10 لكل منهما. (المطلوب طباعة فقط).

الحل:

```
Program ChangeMatrix;
Type
    matrix=array[1..10,1..10] of real;

Var
    a:matrix;
    i,j,n,m:integer;
Begin
    writeln('input n,m:n<10,m<10');
    readln(n,m);
    for i:=1 to n do
      for j:=1 to m-1 do
        begin
          a[i,j]:=a[i,j]-a[i,m];
          write(a[i,j]:8:1);
           writeln;
        end;
readln
End.
```

ملاحظة:

يمكن تعميم المثال بأن يُطلب قراءة عناصر المصفوفة الدخل. ثم تجري عملية الطرح للحصول على المصفوفة الجديـــدة ثم طباعة كل مـــن المصفوفة

الأصلية والمصفوفة الناتجة. كما يلي:

- تعميم: بفرض A هي المصفوفـــة الأصلية و B المصفوفة الناتجة بعد عملية الطرح.

الحل:

```
Program ChangeMatrix;
Type
    matrix=array[1..10,1..10] of real;
Var
    a,b:matrix;
    i,j,n,m:integer;
Begin
    writeln('input n,m:n<10,m<10');
    readln(n,m);
    for i:=1 to n do
     for j:=1 to m do
      begin
        write('input a[',i,',',j,']=');
        readln(a[i,j]);
      end;
     writeln('Matrix A');writeln;
for i:=1 to n do
 begin
  for j:=1 to m do
    write(a[i,j]:8:1);
  writeln;
 end;
for i:=1 to n do
  for j:=1 to m-1 do
     a[i,j]:=a[i,j]-a[i,m];
```

```
      writeln('Matrix B');writeln;
    for i:=1 to n do
      begin
        for j:=1 to m do
          write(b[i,j]:8:1);
        writeln;
      end;
readln
End.
```

نتيجة تنفيذ البرنامج وفق قيم معطاة هي بالشكل التالي:

```
input n,m:n<10,m<10
input a[1,1]=2
input a[1,2]=3
input a[1,3]=4
input a[1,4]=1
input a[2,1]=3
input a[2,2]=6
input a[2,3]=2
input a[2,4]=1
input a[3,1]=5
input a[3,2]=6
input a[3,3]=7
input a[3,4]=2
input a[4,1]=1
input a[4,2]=8
input a[4,3]=4
input a[4,4]=2
Matrix A
```



```
   2.0    3.0    4.0    1.0
   3.0    6.0    2.0    1.0
   5.0    6.0    7.0    2.0
   1.0    8.0    4.0    2.0
Matrix B

   1.0    2.0    3.0    0.0
   2.0    5.0    1.0    0.0
   3.0    4.0    5.0    0.0
  -1.0    6.0    2.0    0.0
```

وهو المطلوب.

٩-٨- تعريف الثوابت:

يمكن الملاحظة في مثال مثلث باسكال أنه إذا استبدلنا حجم المصفوفة 5 بـ N. حيث N عدد صحيح موجب فإنه يمكن التصريح عن أسماء الثوابت في بداية البرنامج في مثال مثلث باسكال يمكن التصريح بالشكل:

```
Const Max N = 5;
```

مثلاً. وعليه يصبح المثل بالشكل التالي:

```
PROGRAM PASCAL_TRIANGLE;
CONST N=5;
VAR
  P:ARRAY[1..N,1..N] OF INTEGER;
  I,J:INTEGER;
BEGIN
    FOR I:=1 TO N DO
      FOR J:=1 TO N DO
```

```
        P[I,J]:=0;
    FOR I:=1 TO N DO
      P[I,I]:=1;
    FOR I:=2 TO N DO
      FOR J:=2 TO I DO
        P[I,J]:=P[I-1,J]+P[I-1,J-1];
    FOR I:=1 TO N DO
     BEGIN
      FOR J:=1 TO I DO
       WRITE(P[I,J]:4):
      WRITELN;
     END;
READLN
END.
```

مما تقدم نكتب ما يلي:

تسمح لغة باسكال أن نعين أسماء لمثل هـذه القيم. حيث يمكن أن نشير إلى هذه القيم بواسطة أسمائها. يتم التصريح عـن ثابت في قسم التصريح عن الثوابت بالشكل التالي:

Const < اسم الثابت > = < قيمة الثابت >

يجب الملاحظة هنا أنه في قسم التصريح عـن الثوابت يمكن التصريح عن أكثر من ثابت حيث يبدأ السطر الأول من تعريف الثوابت بالكلمة المحجوزة Const، والأسطر التالية لها تعرف الثوابت التي نريد، مثلاً:

```
Const A = 10;
      N = 5; C = 17.5;
```

في تعريف الثوابت نستخدم إشارة المسـاواة (=) فقط بدلاً من عملية الإسناد (:=) حيث عند ترجمـة البرنـامج، يقوم المترجـم باستبدال القيـم

المخصصة للثوابت (أي قيمة الثابت) عند كل ظهور لأسماء هذه الثوابت.

إن استخدام عملية تعريف الثوابت تمكننا مـن تغيير قيمة هـذا الثابت بسهولة عند أي عملية تعديل مستقبلية على البرنامج.

إن تعريف الثوابت يجب أن يأتي قبل التصريح عـن المتحولات ضمن البرنامج في لغة تربو باسكال القياسية، أما في لغة تربو باسكال فإن الترتيب بين تعريف الثوابت والتصريح عن المتحولات ليس مهماً.

إذا كان Q ثابتاً ما وكان ;'$' = Const Q فإن قيمة هذا الثابت لا يمكن أن تتغـير في أي عملية إسناد داخل البرنامج مثلاً: ; .. := هو خاطئ واعتماداً على مـا تقدم يمكن في مثالنا من أجـل مثلث باسكال من المرتبة 5 يمكن تغير قيمة N مثلاً 10 لحساب مثلث باسكال من المرتبة 10 فقط بوضع:

```
Const N = 10;
```

بدلاً من:

```
Const N = 5;
```

ننوه هنا أنه لا يمكن استخدام N هنـا كمتحول إذ يجب أن تكون ثـابت وذلك لأن تحديد حجم المصفوفة يكون بثوابت ولا يكون بمتحولات.

٩-٩- المصفوفة ذات الحجوم المختلفة:

لا يمكن للمصفوفة أن تكون ذات حجم متغير فالتصريح يجب أن يكون بثوابـت تحدد الحجم الأعظمي الذي يجب على البرنامج أن يحجزه في الذاكرة. وعليه فلا بد من التصريح عن الحجم الأعظمي ونحدد بعد ذلك الحجم الذي نريد باسـتخدام متحول مـا، وذلك لتحديد الحجم الجزئي الذي نريـد. في مثال مثل باسكال يمكن تعريف Max N كثابت يحدد القيمة العظمى التي سيأخذها متحول N كحجم لمثلث باسكال. وليكن مثلاً Max N = 15. حيث يمكن الكتابة بالشكل

التالي:

```
Program PascalTriangleConst;
Const
        Max N = 15;
Var
         P: Array [1 .. Max N, 1 .. Max N] of Integer;
        N , I , J: Integer;
```

وفـــي بداية البرنامج نطلب قراءة N وهـــو الحجم الجزئي الذي نريـــد الحساب من أجله.

وعليه نكتب البرنامج بالشكل:

```
PROGRAM PASCAL_TRIANGLE;
{PASCAL TRIANGLE USING CONSTANTS(CONST)
                        AS MATRIX LIMITS}
CONST MAXN=10;
VAR
  P:ARRAY[1..MAXN,1..MAXN] OF INTEGER;
  I,J:INTEGER;
  N :INTEGER;
BEGIN
    WRITELN('INPUT N: N<=',MAXN);
    READLN(N);
    FOR I:=1 TO N DO
     FOR J:=1 TO N DO
```



```
      P[I,J]:=0;
    FOR I:=1 TO N DO
     P[1,1]:=1;
    FOR I:=2 TO N DO
     FOR J:=2 TO I DO
      P[I,J]:=P[I-1,J]+P[I-1,J-1];
    FOR I:=1 TO N DO
     BEGIN
      FOR J:=1 TO I DO
         WRITE(P[I,J]:4);
      WRITELN;
     END;
READLN
END.
```

٩-١٠- ضرب مصفوفة بشعاع:

ليكن المطلوب ضرب المصفوفـــة A بشعاع X ليعطي شعاعـــاً B عدد عناصـــره يساوي عدد أســـطر A. ولإجراء عملية الضرب هذه يجب أن تكون عملية الجداء ممكنة. ولا يمكن أن تكون عمليـــة الجداء ممكنة إلا إذا كان هناك توافق بالنسبة لعملية الضرب.

إذا كان كلاً مـــن A و X متوافقين بالنسبة للضرب عندها يمكن إجراء عملية الضرب فإذا كان المطلوب هو إجراء الضرب بالشكل التالي:

$$A \cdot X = B$$

فإن عدد أعمدة المصفوفة A يجب أن يساوي عدد أسطر الشعاع B.

إن المصفوفة الناتجة يجب أن يكون عـــدد أسطرها هـــو عدد أسطر A وعدد أعمدتها هو عدد أعمدة X. وعليه إذا كانت A ذات ســـعة (N, M) و X

ذات سعة M بالشكل التالي:

$$A_{(N,M)} \cdot X_{(M)} = B_{(N)}$$

فإن المصفوفة الناتجة يجب أن تكون ذات سعة N.

أما قيمة عنصر في سطر في الشعاع B فإنه يساوي إلى مجموع جداءات عناصر نفس السطر في A في عناصر المصفوفة X.

$$[\text{——}] \cdot [\ |\ ] = [\text{—}]$$
$$A \quad X \quad B$$

والمطلوب:

١- قراءة المعطيات:

وهي حجم المصفوفة A ثم قراءة عناصر المصفوفة A ثم قراءة حجم المصفوفة العمود X ومن ثم قراءة عناصر المصفوفة العمود X.

٢- حساب الجداء:

أي حساب A * X ووضع النتيجة في B.

٣- كتابة المصفوفات الثلاث.

- البرنامج بلغة تربو باسكال:

```
PROGRAM MATRIX_MULTIPLICATIONS;
{MAKE MATRIX MULTIPLICATIONS:A*X=B}
CONST
   MAXN=10;
   MAXM=10;
VAR
  A:ARRAY[1..MAXN,1..MAXM] OF REAL;
  X:ARRAY[1..MAXM] OF REAL;
  B:ARRAY[1..MAXN] OF REAL;
  M,N:INTEGER;
```



```
  I,J:INTEGER;
BEGIN
  WRITELN('MATRIX BY VECTOR
                    MULTIPLICATION A*X=B');
  WRITELN('==========================');
  WRITELN('INPUT N,M<=',MAXN,',',MAXM);
  READLN(N,M);
  WRITELN('INPUT MATRIX ', N ,'x' ,M);
  FOR I:=1 TO N DO
   BEGIN
    WRITELN('INPUT A[',I,'] ', M ,' ELEMENTS');
    FOR J:=1 TO M DO
     READ(A[I,J]);
    READLN;
   END;
  WRITELN('INPUT X');
  FOR J:=1 TO M DO

BEGIN
    WRITELN('X[',J,']');
    READLN(X[J]);
  END;
  FOR I:=1 TO N DO
   BEGIN
    B[I]:=0;
    FOR J:=1 TO M DO
     B[I]:=B[I]+A[I,J]*X[J];
    END;
  WRITELN('A');
  FOR I:=1 TO N DO
```

```
  BEGIN
    FOR J:=1 TO M DO
      WRITE(A[I,J]:8:3);
    WRITELN;
  END;
  WRITELN('X');
  FOR J:=1 TO M DO
    WRITELN(X[J]:8:3);
  WRITELN('B');
  FOR I:=1 TO N DO
    WRITELN(B[I]:8:3);
READLN
END.
```

إن نتيجة تنفيذ البرنامج وفق قيم معطاة هي بالشكل التالي:

```
MATRIX BY VECTOR MULTIPLICATION A*X=B

INPUT N,M<=10,10
3 3
INPUT MATRIX 3x3
INPUT A[1] 3 ELEMENTS
1 2 3
INPUT A[2] 3 ELEMENTS
4 5 6
INPUT A[3] 3 ELEMENTS
7 8 9
INPUT X
```



```
X[1]
1
X[2]
2
X[3]
3

A
   1.000   2.000   3.000
   4.000   5.000   6.000
   7.000   8.000   9.000


X
   1.000
   2.000
   3.000

B
  14.000
  32.000
  50.000
```

- تعديل المثال السابق: (تعديل التمرين ليصبح البرنامج يسأل المستثمر في كل مرة عن إجراء عملية جداء جديدة أولاً).

يمكن تعديل البرنامج السابق بإدخـــال بعض المصطلحات الجديدة. مثلاً الكلمة المحجوزة Label الذي يمثل تعريف للتصريح عـــن قيمـــة ينتقل إليهـــا البرنامج عند الضرورة كأن نكتب مثلاً ;3 Label حيث يتم كتابة :3 في الجزء

المطلوب مـــن البرنامج. وعليه يضاف تعريف الواحدات وتعريف وحدة تنظيف الشاشة بالشكل: Uses crt; وفي نص البرنـــامج بعد الكلمـــة المحجوزة Begin نكتب ClrScr في مثالنـــا سوف نجعل البرنامج يسأل إذا أردنا حساب آخر أو الانتهاء من البرنامج دون حساب كما في الشكل التالي:

```
PROGRAM MATRIX_MULTIPLICATIONS;
{MAKE MATRIX MULTIPLICATIONS:A*X=B}
USES CRT;
LABEL START;
CONST
   MAXN=10;
   MAXM=10;
Type
  MATRIX=ARRAY[1..MAXN,1..MAXM] OF REAL;
  VECTOR=ARRAY[1..MAXM] OF REAL;
  VECTOR1=ARRAY[1..MAXN] OF REAL;


VAR
  A:MATRIX;B:VECTOR1;X:VECTOR;
  M,N,I,J:INTEGER;
  D:CHAR;
BEGIN
 START:CLRSCR;
 WRITELN('MATRIX BY VECTOR MULTIPLICATION
                                        A*X=B');
 WRITELN('==============================');
  WRITELN('INPUT N,M<=',MAXN,',',MAXM);
  READLN(N,M);
```



```
 WRITELN('INPUT MATRIX ', N ,'x' ,M);
 FOR I:=1 TO N DO
  BEGIN
   WRITELN('INPUT A[',I,'] ', M ,' ELEMENTS');
   FOR J:=1 TO M DO
     READ(A[I,J]);
   READLN;
  END;
 WRITELN('INPUT X');
 FOR J:=1 TO M DO
  BEGIN
    WRITELN('X[',J,']');
    READLN(X[J]);
  END;
 FOR I:=1 TO N DO
  BEGIN
   B[I]:=0;
   FOR J:=1 TO M DO
     B[I]:=B[I]+A[I,J]*X[J];
   END;
 WRITELN('A');
 FOR I:=1 TO N DO
  BEGIN
   FOR J:=1 TO M DO
     WRITE(A[I,J]:8:3);
   WRITELN;
  END;
 WRITELN('X');
 FOR J:=1 TO M DO
  WRITELN(X[J]:8:3);
```

```
  WRITELN('B');
  FOR I:=1 TO N DO
    WRITELN(B[I]:8:3);
  WRITE('IF YOU WANT TO CALCULATE AGAIN
                PRESS(Y) THEN PRESS ENTER');
  READLN(D);
  IF(D='Y')OR(D='y')THEN GOTO START;
  WRITELN('PRESS ENTER TO STOP');
READLN
END.
```

٩-١١- خوارزمية الترتيب الفقـــاعي (خوارزميـــة الفرز بالتعويم):

Bubble Sort

ليكن لدينا شعاع يحوي عناصر من نمط قابل للترتيب، أي علاقة أصغر أو وأكبر معرفة على عناصره. والمطلوب ترتيب أو فـــرز هذه العناصر حسب ترتيب معين تصاعدي أو تنازلي. توجد عـــدة خوارزميـــات للترتيب Sorting سنأخذ خوارزمية الفرز بالتعويم Bubble Sort وليكن الفرز تصاعديـــاً. تعتمد خوارزمية الفرز بالتعويم أو خوارزمية الترتيب الفقاعي على ما يلي:

١- عبور كامل المصفوفة السطر لكل عنصر من عناصرها عدداً من المرات أقل بمرة واحدة من عناصر المصفوفة السطر.

٢- المقارنة أثناء العبور بين محتويـــات حجرتين متجاورتيـــن مثلاً في أول عبور سيتم المقارنة بين الحجرتين (1) و (2) ومن ثم محتوى الحجرتين (2) و (3) وهكذا ... إلى نهاية حجرات السطر.

٣- أثناء عملية المقارنـــة تتم عملية التبديل بين محتويـــات الحجرات إذا دعا الأمر إلى ذلك أي إذا كانا مختلفين في الترتيب وإلا حافظـــاً على أماكنهمـــا وذلك من أجل جعل الحجرة ذات الرقم الأكبر تحتوي العنصر الأكبر.

٤- بعد انتهاء عملية مقارنة المحتويات لجميع الحجرات فإن آخر حجرة سوف تحتوي العنصر المرتب الأخير، أي سيؤدي ذلك إلى تعويـــم العنصر الأكبر في المصفوفة السطر.

٥- وهكذا بتكرار العملية مرة أخرى ستجعل الحجرة قبل الأخيرة تحتوي على العنصر المرتب المناسب، أي في كل مرة يطفو أو يعوم العنصر الأكبر بين العناصر المتبقية وهكذا بالتالي نحصل في النهاية على عناصـــر المصفوفـــة السطر مرتبة تصاعدياً كما هو المطلوب.

٩-١١-١- مثال على خوارزمية الفرز بالتعويـــم :

لتكن A هي مصفوفة السطر، عدد عناصـــر المصفوفة N، نحتاج إلى متحول J له نفس نمط عناصر المصفوفة السطر.

- الخوارزمية:

١- من أجل I = 1 حتى N-1 كرر:

١-١- عوم العنصر

١-٢- من أجل J = 1 حتى N-1 كرر:

- بدل المواقع

- إذا $A_J > A_{J+1}$ نفذ:

$L := A_J ; A_J := A_{J+1} ; A_{J+1} = L ;$

- مثال عددي:

لتكن لدينا مصفوفة السطر التالية:

A = [4 9 2 3 1 8 7]

والمطلوب ترتيب عناصر المصفوفة بشكل تصاعدي وذلك وفـــق خوارزميـــة الفرز بالتعويـــم .

الحل:

الحالـــة الابتدائية هي: 7 8 1 3 2 9 4 يتم مقارنة عنصرين متجاورين والتبديل بينهما وذلك إذا لم يوافقا الترتيب وذلك كما يلي:

- في المرة الأولى:

4 9 2 3 1 8 7
4 2 9 3 1 8 7
4 2 3 9 1 8 7
4 2 3 1 9 8 7
4 2 3 1 8 9 7
4 2 3 1 8 7 9

تم هنا تعويـــم العنصر الأكبر نطبق نفس العمل أي تعويـــم العنصر الأكبر بين العناصر المتبقية.

- في المرة الثانية:

4 2 3 1 8 7 9
2 4 3 1 8 7 9
2 3 4 1 8 7 9
2 3 1 4 8 7 9
2 3 1 4 7 8 9
2 3 1 4 7 8 9

- في المرة الثالثة:

2 3 1 4 7 8 9
2 1 3 4 7 8 9

- في المرة الرابعة:

1 2 3 4 7 8 9

تجـــدر الإشارة هنـــا إلى التذكير إلى أن هـــذه الخوارزميـــة ليست من الخوارزميات المثلى فيما يخص الفرز لأنها ستضطر لعدد من عمليات المقارنة بين العناصر من مرتبة N^2 تقريباً (N-1 مقارنة لتعويم العنصر 1).

إن عـــدد المقارنات ثابتاً ولو كانت المجموعة مرتبـــة جزئياً أو كليـــاً. نلاحـــظ في هذا المثال أنه يمكن الحصول على المجموعة مرتبـــة بشكل مبكر وإن لـــم نكن بحاجة إلى هـــذا العدد من التكرارات (المقارنـــات) التي فرضتها الخوارزمية.

- البرنامج بلغة تربو باسكال:

```
PROGRAM BUBLE_SORT;
{sort numbers using Buble Sort Algorithem}
CONST MAXN=100;
VAR
  A:ARRAY[1..MAXN] OF INTEGER;
  L:INTEGER;
  I,J,N:INTEGER;


BEGIN
  WRITELN('INPUT N: N<=',MAXN);
  READLN(N);
  WRITELN('INPUT ', N,' DATA TO SORT');
  FOR I:=1 TO N DO
   READLN(A[I]);
  FOR I:=1 TO N-1 DO
```

```
  FOR J:=1 TO N-1 DO
   IF (A[J]>A[J+1]) THEN
    BEGIN
     L:=A[J];
     A[J]:=A[J+1];
     A[J+1]:=L;
    END;
  WRITELN('SORTED ELEMENTS');
  FOR I:=1 TO N DO
   WRITELN(A[I]);
READLN
END.
```

٩-١٢- نمط المعطيات سلسلة رمزية:

تمتلك لغة تربو باسكال نوع معطيات غير موجود في لغة باسكال القياسية وهو نوع معطيات سلسلة رمزية String. إن متحولات هذا النوع تُستخدم لتخزين تتابع من الرموز أي لتخزين سلاسل رمزية أو أسماء. يمكن التصريح عن هذا النمط من السلاسل الرمزية بدون تحديد لطول هذه السلاسل أو بتحديد طول أعظمي لها.

مثلاً إذا كان:

```
Var Q : String;
```

فهذا يعني أن المتحول Q هو من نمط سلسلة وبدون تحديد لطول هذا المتحول يعني في لغة تربو باسكال أن هذا المتحول لا يمكن أن يكون أكثر من 255 رمزاً.

وإذا كان:

```
Var I : String [20];
```


فهذا يعني أن المتحول I لا يمكن أن يكون بطول أكثر من 20 محرفاً أي الحد الأعظمي لمحارفه هو 20 ويمكن أن يكون أقل.

وعليه يمكن أن نكتب مثلاً:

```
Var A : String;
    B : String [5];
    C : String [15];
```

هذا يعني أن المتحول A بطول أعظمي 255 رمزاً. والمتحول B بطول أعظمي 5 رموز. والمتحول C بطول أعظمي 15 رمزاً.

من أجل إسناد قيمة للمتحول من النمط String يجب أن نضع سلسلة الرموز بين فواصل علوية. كما في المثال التالي:

```
A := ' This String is Less Than 255 Char ' ;
B := ' Five ' ;
C := ' My Name is Ahmad ' ;
```

في C عدد المحارف هو 14 وهو أقل من 15 وهو مقبول.

في B تم إسناد 4 محارف وهو مقبول أيضاً.

عند إسناد سلسلة رمزية إلى متحول يجب أن يكون طول السلسلة أقل من الحجم المخصص لتخزين هذا المتحول أما إذا كانت أطول من الحجم المخصص عندها سيتم حذف أو تجاهل القيم أو المحارف التي تقع خارج الحجم المخصص.

يتم تطبيق عملية الإسناد وعمليات المقارنة على نمط المعطيات سلسلة String كما طبقناها سابقاً على نمط المعطيات Char. طبعاً نستخدم شفرات ASCII في عمليات المقارنة هذه. وتنجز هذه المقارنة رمزاً رمزاً. فمثلاً عملية المقارنة التالية تعطينا القيمة True.

' Hisham' > ' Hishac '

والسبب أن المحرف m ضمن السلسلة الرمزيـــة Hisham يملك شفرة ASCII مقابلة أكبر من شفرة المحرف C ضمن السلسلة الرمزيـــة Hishac. أما ما قبل هذا المحرف فإن السلسلتين متطابقتين.

يمكن التنويه هنا إلى أنه يمكن جمع سلسلتين لإنتاج سلسلة أطول، مثلاً يمكـــن كـــتابـــة This is my Program بنتيجـــة جمع السلسلتين This is و my Program حيث أن الفراغ هو محرف هنا ويتم إلحاقها بالمتحول Q مثلاً:

Q := ' This is ' + ' my Program ' ;

– إن عمليات القراءة والكتابة متاحة في النمط String حيث يمكن كتابة ما يلي:

إذا كان

Var Q : String ;

فإن

Readln (Q) ;

قراءة الدخل كسلسلة رمزية متاحة.

وإن

Writeln (Q) ;

كتابة محتوى Q كسلسلة رمزيـــة أيضاً متاحة. وتصح أيضـــاً Read و Write وذلك عند تحديد طول محدد للسلسلة الرمزية.

٩-١٣- مصفوفات السلاسل الرمزيـــة أحاديـــة البعد في لغة تربـــو باسكال:

تتميز مصفوفات السلاسل الرمزية أحادية البعد

Array [...] of string

بعمليات خاصة.

ليكن مثلا:

Q : Array [1 .. 20] of String

١- الإسناد: يمكن الإسناد مباشرة.

يمكـــن الكـــتابة مثلاً: ' ALI ' =: [1] Q إذ يجب أن تتضمن السلسلـــة الرمزية المحددة بين فواصل علوية عدداً من المحارف مساوياً أو أقل من الحجم الأعظمـــي المحدد للسلسلة. في هذا المثال لم يتم التحديد وعليه فإن الحد الأول في المصفوفة Q أخذ القيمة ' ALI '.

٢- عملية القراءة والكتابة:

إن القراءة والكتابة متاحـــة مثلاً : Readln Q [1] أو ; Readln Q [5] وهكذا ... وأيضاً ; Writeln Q[2] أو ; Writeln Q[6] وهكذا.

٣- عملية المقارنة:

إن عمليات المقارنة متاحة للمقارنـــة بين قيمتين مركبتين مـــن مركبات المصفوفة السطر. أي عمليات < , <= , > , =>.

فمثلاً إذا كان ' ALI ' =: [1] Q و ' SAM ' =: [2] Q فإنه يمكن أن نكتب:

```
If Q [1] > Q[2] Then
    Writeln ( ' Q1 >= Q2 ' )
Else Writeln ( ' Q1 <= Q2 ' ) ;
```

مثال:

نفس المثال السابق مثال الفرز بالتعويـــم ولكن مـــن أجل ترتيب أسماء حيث أن البرنامج سيسمح لنا بتعريف العمليات على مصفوفات السلاسل الرمزية

أحادية البعد.

الحل:

```
PROGRAM BUBLE_SORT_STRING;
{sort numbers using Buble Sort Algorithem}
CONST MAXN=100;
VAR
  A:ARRAY[1..MAXN] OF STRING;
  L:STRING;
  I,J,N:INTEGER;
BEGIN
 WRITELN('INPUT N: N<=',MAXN);
 READLN(N);
 WRITELN('INPUT ', N,' DATA TO SORT');
 FOR I:=1 TO N DO
  READLN(A[I]);
 FOR I:=1 TO N-1 DO
  FOR J:=1 TO N-1 DO
   IF (A[J]>A[J+1]) THEN
    BEGIN
     L:=A[J]; A[J]:=A[J+1];A[J+1]:=L;
    END;
 WRITELN('SORTED ELEMENTS');
 FOR I:=1 TO N DO
  WRITELN(A[I]);
READLN
END.
```

إن نتيجة تنفيذ البرنامج وفق قيم معطاة هي بالشكل التالي:


```
INPUT N: N<=100
10
INPUT 10 DATA TO SORT
RAYD
SAMER
YASIR
AHMAD
LAELA
BELAL
ALE
MOHAMED
MOONA
NUR

SORTED ELEMENTS

AHMAD
ALE
BELAL
LAELA
MOHAMED
MOONA
NUR
RAYD
SAMER
YASIR
```

مثال:

اكتب برنامجا لقراءة مصفوفة وطباعتها ثم تبديل عناصر الأعمدة الفردية بالقيمـــة 8 وطباعـــة المصفوفة الناتجة.

البرنامج بلغة باسكال:

```
Program ChangeMatrix;
Type
    matrix=array[1..20,1..20] of real;
Var
    a:matrix;
    i,j,n,m:integer;
Begin
    writeln('input n,m:n<20,m<20');
    readln(n,m);
    for i:=1 to n do
     for j:=1 to m do
      begin
       write('input a[',i,',',j,']=');
       readln(a[i,j]);
      end;
    writeln;
    writeln('Matrix before Change');writeln;
    for i:=1 to n do
      begin
        for j:=1 to m do
          write(a[i,j]:5:1);
        writeln;
      end;
    writeln;
```



```
    writeln('Matrix after Change');writeln;
    for i:=0 to n do
     for j:=1 to m do
        a[i,2*j]:=8;
    for i:=1 to n do
      begin
        for j:=1 to m do
          write(a[i,j]:5:1);
        writeln;
      end;
readln
End.
```

تمارين عامة

١- لتكـــن لدينا متتالية رمـــوز عددها زوجي لا يتجاوز 100 رمز والمطلوب: تحديد فيما إذا كانت هذه المتتالية قابلة للطي (يقال عن متتالية أنها قابلة للطي إذا كان نصفها الأول سرداً عكسياً لنصفها الثاني)، أي أنها من الشكل:

$$S1 , S2 , \dots Sn , Sn , Sn-1 , \dots , S1$$

٢- اكتـــب بـــرنامجاً يقـــرأ عناصر المتتالية $a_1 , a_2 , \dots , a_n$ والمطلوب كتابة برنامج يحسب قيمة المقدار:

$$a_1 a_n + a_2 a_{n-2} + a_3 a_{n-3} + \dots + a_i a_{n-i}$$

٣- اكتب برنامجاً لإيجاد المتوسط الحسابي لعناصر مصفوفة سطر أو عمود.

٤- اكتب برنامجاً لإيجاد مجموع مصفوفتي السطر أو العمود.

٥- اكتب برنامجاً لإيجاد جداء مصفوفتي السطر أو العمود.

٦- اكتب برنامجاً لإيجاد أصغر عنصر في مصفوفة.

٧- اكتب برنامجاً يبحث عـن سطر وعمود لهما المجموع نفسه في مصفوفـــة مربعة.

٨- اكتب برنامجاً لحساب منقول مصفوفة مربعة من البعد $N \times N$.

٩- اكتب برنامجا لإنشاء وكتابة مصفوفة من الشكل التالي:

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 9 \end{bmatrix}$$

١٠- اكتب برنامجا لقراءة مصفوفة ثم تبديل عناصر أول خمســـة أســـطرفيها بالقيمـــــة 8 وطباعـــة المصفوفة الناتجة.

١١- اكتـــب بـــرنامجا يقوم بالتبديل بين الأسطر الفردية والأسطر الزوجية في مصفوفة مدخلة .

١٢- ليكن:

```
Const N=20;
Var
    A,B,C:array[1..20,1..20] of real;
    X,Y:array[1..20] of real;
```

والمطلوب حساب:

١- C=A+B .

٢- Y=A.X .

٣- C=A.B .

٤- B=BT (منقول B).

١٣- اكتب برنامجا لإنشاء وكتابة مصفوفة من الشكل التالي:

$$\begin{bmatrix} 1 & 2 & 3 & \ldots\ldots & 10 \\ 0 & 1 & 2 & \ldots\ldots & 9 \\ 0 & 0 & 1 & \ldots\ldots & 8 \\ \ldots & \ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & 0 & \ldots\ldots & 1 \end{bmatrix}$$

١٤- اكتب برنامجا لإنشاء وكتابة مصفوفة من الشكل التالي:

$$\begin{bmatrix} 1 & 2 & 3 & \ldots\ldots & 10 \\ 0 & 2 & 3 & \ldots\ldots & 10 \\ 0 & 0 & 3 & \ldots\ldots & 10 \\ \ldots & \ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & 0 & \ldots\ldots & 10 \end{bmatrix}$$

١٥- اكتب برنامجا لإنشاء وكتابة مصفوفة من الشكل التالي:

$$\begin{bmatrix} 1 & 2 & 3 & \ldots\ldots & 10 \\ 11 & 12 & 13 & \ldots\ldots & 20 \\ 21 & 22 & 23 & \ldots\ldots & 30 \\ \ldots & \ldots & \ldots & \ldots\ldots & \ldots \\ 91 & 92 & 93 & \ldots\ldots & 100 \end{bmatrix}$$

١٦- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Matrix;
Var
```



```
    a:array[1..10,1..10] of integer;
    i,j:integer;

Begin
    for i:=1 to 10 do
      a[1,i]:=0;
    for i:=2 to 10 do
      begin
        a[i]:=a[1];
        a[i,i]:=i-1;
      end;
    for i:=1 to 10 do
     begin
       for j:=1 to 10 do
         write(a[i,j]:3);
       writeln
     end;
readln
End.
```

١٧- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Matrix;
Var
    a:array[1..10,1..10] of integer;
    i,j,k:integer;
Begin
    k:=0;
    for i:=1 to 10 do
      begin
```

```
        for j:=1 to 10 do
            a[i,j]:=k+j;
        k:=k+10;
      end;
    for i:=1 to 10 do
     begin
       for j:=1 to 10 do
          write(a[i,j]:3);
       writeln
     end;
readln
End.
```

١٨- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Matrix;
Var
    a:array[1..10,1..10] of integer;
    i,j:integer;
Begin
    for i:=1 to 10 do
        for j:=1 to 10 do
            a[i,j]:=j;
    for i:=1 to 10 do
     begin
       for j:=1 to 10 do
         if j>=i then
            write(a[i,j]:3)
         else
            write('0':3);
```



```
       writeln
     end;
readln
End.
```

١٩- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Matrix;
Var
    a:array[1..10,1..10] of integer;
    i,j:integer;
Begin
    for j:=1 to 10 do
        a[1,j]:=j;
    for i:=2 to 10 do
        for j:=1 to 10 do
            a[i,j]:=a[i-1,j]-1;
    for i:=1 to 10 do
     begin
       for j:=1 to 10 do
         if j>=i then
            write(a[i,j]:3)
         else
            write('0':3);
       writeln
     end;
readln
End.
```

٢٠- اكتب برنامجا لحساب مجموع عناصر القطر الرئيسي في مصفوفة مربعة.

٢١- اكتب برنامجا لحساب مجموع عناصر القطر الثانوي في مصفوفة مربعة.

٢٢- اكتب برنامجا لحساب مجموع العناصر الواقعة فوق القطر الرئيسي في مصفوفة مربعة.

٢٣- اكتب برنامجا لحساب مجموع العناصر الواقعة تحت القطر الرئيسي في مصفوفة مربعة.

٢٤- اكتب برنامجا لحساب منقول مصفوفة.

الفصل العاشر

البرامج الجزئية Subprograms

١٠-١- مقدمة:

تحتوي معظم لغات البرمجة على بنى تسمح بكتابة البرامج الجزئية. حيث يتم تقسيم البرامج إلى مجموعة من الأجزاء المنفصلة. يقوم كل جزء منها بإنجاز قسم من العمل المطلوب من البرنامج الكلي. إن فكرة التجزئة للسيطرة على مسألة ما (Divide And Conguer) هي أحد مبادئ التفكير العلمي الأساسية التي نادى بها العالم الرياضي ديكارت Descartes.

تكمن أهمية استخدام البرامج الجزئية هو في تجنب تكرار نفس التعليمات للقيام بأعمال متماثلة. في أغلبية لغات البرمجة يمكن التمييز بين تجنب تكرار نفس التعليمات للقيام بأعمال متماثلة.

في أغلبية لغات البرمجة يمكن التمييز بين نوعين من البرامج الجزئية:

١- التابع Function.

٢- الإجرائية Procedure.

إن التابع هو برنامج جزئي يقوم بحساب لقيمة معينة هي قيمة التابع. أما الإجرائية فهي برنامج جزئي يمكن أن يكون له عدة معطيات دخل وخرج تسمى معاملات أو وسطاء (Parameters) البرنامج الجزئي، كما يمكن ألا يكون هناك أي وسطاء.

١٠-٢- البرنامج الجزئي التابع Function:

صممت التوابع لحساب قيمة مـــا. حيث يصدر التابع على حساب قيمـــة خرج معينة. يمكن أن تُستخدم مباشرة في أي تعبير له نفس نمط هذه القيمة.

مثلاً التابع Cos (x) له وسيط واحد مـــن نمط حقيقي ويعيد قيمـــة في المجال [-1,1] أي أن خرج التابع أو قيمة التابع هي أيضاً مـــن نمط حقيقي. لذا يمكن استخدام التابع Cos في أي تعبير حسابي مثلاً يمكن كتابة:

```
Y := Cos (x) + 5.25
```

أو

```
Y := Sqr (Cos (x))
```

وبشكل عام تكون التوابع في لغة باسكال هي روتينات معينة تعيد دائماً قيمة محددة.

١٠-٢-١- تعريف التابع في لغة تربو باسكال:

يتم التصريح عـــن البرنامج الجزئي التابع في البرنامج وذلك في الجزء الذي يلي قسم التصريحات عن المتحولات أي بعد ;... Var إن وجدت.

يبدأ رأس التابع بالمميز المحجوز Function إن الشكل العـــام لترويسة التابع هي من الشكل:

```
Function < Function_Name > (<P1>: <Type1>; ...
... ; <Pn> : <Typen>) : < Value_Type >;
```

حيث:

< Function_Name >: هو اسم التابع الذي يعرفه المبرمج.

<P1>, ... , <Pn>: متحولات من الأنماط <Type1>, ... , <Typen>.

< Value_Type >: هو نمط قيمة التابع.

يجدر الملاحظة هنا أنه يمكن تعريف التوابع دون وسطاء بالشكل:

```
Function < Function_Name > : < Value_Type >;
```

أي في هذه الحالة يعرف التابع من اسم التابع فقط ومن نمط قيمة التابع.

يعيد التابع أي قيمة مـــن أي نوع معطيات يحدد بعد النقطتان في رأس التابع.

إن استدعاء التابع يعيد قيمة هـــذا التابع. لذا يجب أن تعطى هذه القيمة إلى التابع من خلال عبارة إسناد في أي مكان ضمن جسم البرنامج حيث تجعل التابع مساوياً لقيمة مـــا أو مساوياً إلى قيمة أحـــد المتحولات أو مساوياً لنتائج تعبير معين بشرط أن تكون هذه القيمة هي من نفس نوع المعطيات المخصص للتابع.

ونذكر في النهاية دومـــاً أن رأس التابع يجب أن يبدأ بالكلمة المحجوزة Function وينتهي بنوع معطيـــات محدد يأتي بعد نقطتين. أمـــا جسم البرنامج فيجـــب أن يحتوي على عبارة إلحاق تحدد القيمة التي يجب أن تعـــاد على أنها قيمة هذا التابع.

مثال:

اكتب برنامجاً لحساب قيمة التعبير التالي:

$$Y = \frac{N!}{M! - N!}$$

يمكنـــنا كتابة البرنامج بالطريقة التقليدية، ويمكن استخدام تعريف التابع لإيجاد حل لهذا المثال.

- البرنامج بلغة تربو باسكال:

```
PROGRAM FACT_N_M;
VAR
```

```
  N,M :INTEGER;
  Y   :REAL;
FUNCTION FACT(N:INTEGER):INTEGER;
VAR
  I,F:INTEGER;
BEGIN
    F:=1;
    FOR I:=1 TO N DO
       F:=F*I;
       FACT:=F;
END;
BEGIN
    WRITELN('GIVE N,M:(N>M)');
    READLN(N,M);
    Y:=FACT(N)/(FACT(M)*FACT(N-M));
    WRITELN('C(',N,',',M,')=',Y);
READLN
END.
```

١٠-٢-٢- ملاحظات عامة حول تعريف التوابع:

١- إن التصريح عن التابع يتم بالكلمة المفتاحية Function يليها اسم التابع المصرح عنه.

٢- يعرف نمط قيمة التابع في آخر ترويسة التابع.

٣- يمكن للتابع أن يعرف دون وسطاء.

٤- التصريح عن برامج التابع يكون في الجزء الذي يلي التصريح عن المتحولات كما في الشكل:


```
Program ... ;
Const ... ;
Type ... ;
Function ... ;
   ...
   Begin
   ...
   End;
Var ... ;
Begin
...
End.
```

٥- يخضع جسم التابع لنفس قواعد جسم البرنامج الرئيسي أي قسم التصريح ثم بداية ونهاية وبينهما جزء التعليمات.

٦- لا بد أن تتم عملية إسناد لاسم التابع حتى تعرف القيمة التي يعيدها التابع كما في المثال السابق ;Fact := f

وعليه فإن جسم التابع يجب أن يحتوي على الأقل عبارة إسناد واحدة على الأقل تحتوي في الطرف الأيسر على اسم التابع.

٧- إن التابع هو عبارة عن طريقة لتوليد قيمة محدودة هي قيمة التابع. إن استدعاء التابع في لغة باسكال هو عبارة عن قيمة. لذا يمكن أن يتم استدعاء التابع وفي أي مكان يمكن أن توجد فيه قيمة هذا التابع.

٨- عند تنفيذ تعليمات البرنامج، وعندما يصل التنفيذ إلى تعليمة استدعاء التابع عندها يتوقف تنفيذ البرنامج بشكل مؤقت ويُنتقل التنفيذ إلى التابع حيث يقوم

التابع بتنفيذ تعليماته ويستمر التنفيذ حتى تُحسب قيمة معينة وتُسند للتابع وبعد الإسناد تعاد هذه القيمة إلى أوامر البرنامج ويستأنف البرنامج تنفيذ تعليماته وفق قيمة جديدة حصل عليها نتيجة تنفيذ التابع.

٩- تمتلك باسكال عدداً كبيراً من التوابع مسبقة التعريف يمكن استخدامها مباشرة ضمن برامجنا وأضافت تربو باسكال كميات أخرى من التوابع مسبقة التعريف.

من التوابع مسبقة التعريف في لغة باسكال القياسية نذكر ما يلي:

ABS, ARCTAN, CHR, COS, EXP, LN, ODD, PRED. ROUND, SIN, SQR, SQRT, SUSS, TRUNC.

ومن التوابع مسبقة التعريف التي أضافها تربو باسكال نذكر:

CONCAT, COPY, DEC, DELETE, FRAC, INC, INSERT, INT, LENGTH, POS, RANDON, SIZEOF, UPCASE.

١٠- إن المتحولات المعرفة داخل البرنامج الجزئي التابع هي متحولات محلية خاصة بالبرنامج الجزئي التابع، لذا لا ترى خارج هذا التابع.

١١- إن المعاملات في البرنامج الجزئي التابع هي على نوعين معاملات دخل فقط ومعاملات دخل وخرج في آن معاً.

إن معاملات الدخل والخرج هي المعاملات التي يمكن أن تتغير نتيجة تنفيذ البرنامج الجزئي التابع، وحتى تصبح المعاملات هي معاملات دخل وخرج في نفس الوقت يكفي أن نكتب كلمة Var قبل المتحول.

على سبيل المثال ليكن n متحول خرج ونريد إيجاد !n عندها التصريح عن التابع يصبح بالشكل:

Function Fact (Var n: Integer);

إن المثالين التاليين يوضحان طريقة عمل المعاملات في كل من

الحالتين السابقتين.

مثال:

```
PROGRAM FACT_N_M;
VAR
  L :INTEGER;
FUNCTION FACT(N:INTEGER):INTEGER;
VAR
  I,F:INTEGER;
BEGIN
    F:=1;
    FOR I:=1 TO N DO
       F:=F*I;
    FACT:=F;
    N:=50;
END;
BEGIN
    L:=7;
    WRITELN(L,'.....',FACT(L));
    WRITELN(L);
READLN
END.
```

إن هذا المثال يقوم بحساب عاملي عدد ثم يطبع قيمة L المدخلة 7 وبعد تنفيذ Fact يطبع L مرة أخرى التي ستكون 7 أيضاً وذلك لأن N هو متحول دخل فقط وليس خرج، أي أن قيمة N لا تخرج من البرنامج بالقيمة N = 50 كما في المثال .

مثال:

```
PROGRAM FACT_N_M;
VAR
  L :INTEGER;
FUNCTION FACT(VAR N:INTEGER):INTEGER;
VAR
  I,F:INTEGER;
BEGIN
   F:=1;
   FOR I:=1 TO N DO
      F:=F*I;
   FACT:=F;
   N:=50;
END;
BEGIN
   L:=7;
   WRITELN(L,'.....',FACT(L));
   WRITELN(L);
READLN
END.
```

في هذا المثال وبعد طباعة L وحساب قيمة عاملي عـدد تكون القيمة المخرجة بعـد تطبيق Fact هي 50 وذلك لكون n متحول خرج في هذا المثال.

١٢- إذا كان المتحول المعرف في التابع هو خرج ولم نكتب قيمة Var قبله فإن هـذا سوف يؤدي إلى إبقاء قيمة هذا المتحول مساوياً إلى قيمته الأصلية دون تغيير وذلك عند الخروج من التابع.

١٣- إن اسـتخدام تعريف التابع في التعريف يمكن البرنامج العام من استخدام هذا التابع في نص البرنامج العام كأي تابع جاهـز أو مسبق التعريف. إذ أن كل عمليـة استخدام لهذا التابع سوف تؤدي إلى حساب قيمتها مـرة واحدة.

١٠-٣- البرنامج الجزئي (الإجرائية): Procedure

تتكون الإجرائية من مجموعة من العبارات من أجل إنجاز عمل معين. إن الإجرائية وبشكل عام يمكن أن تكون برنـامج مصغر يملك قسم تصريـح خاص به. صممت الإجرائيات لإنجاز مهام معينة، مثل مسح الشاشة، أو ترتيب مجموعة القيم، الخ ...

١٠-٣-١- تعريف الإجرائية Procedure:

إن الإجرائية هي برنامج جزئي كما هـو التابع. لكن تختلف عنه بعدم إعادته لقيمة معينـة بذاتها هي قيمة التابـع. لذا فإن التصريح عن الإجرائية لا يتضمن نمط القيمة المعادة في آخر الترويسة كمـا رأينا في حالة التابع. تعرف الإجرائية بتحديـد المتحولات المستخدمـة في هـذه الإجرائية. ووضع سلسلة التعليمات التي بتنفيذها تنجز الإجرائية العمل الموكل إليها. إن تعريف الإجرائية يجب أن يتم أو يصح عنه في قسم التصريحات الخاص بالبرنامج.

إن الشكل العام لترويسة الإجرائية هو كما يلي:

```
Procedure <Procedure_name> (<P1>: <Type1>; ...,
                            ; <Pn> : <Typen>);
```

حيث:

<Procedure_name>: هو اسم الإجرائية التي يجب أن تميز الإجرائية

عن غيرها من الإجرائيات.

<Pn>,... , <P1>: وسطاء الإجرائية وهي من أنماط مختلفة.

يمكن أن نعـرف الإجرائيـة دون وسـطاء وذلك بالكلمـة المحجوزة Procedure والاسم فقط كما يلي:

Procedure < Procedure_Name>;

بعد تعريـف الإجرائيـة أصبح بالإمكان اسـتخدامهـا في البرامـج، وباستدعائه يتم إنجاز العمل المطلوب منـه. أي يجب تعريـف الإجرائية قبـل استدعائها ضمن البرنامج. وعند استدعائها تنفذ مجموعة التعليمات المحتواة في هذه الإجرائية. بعد الانتهاء مـن تنفيذ آخر تعليمة من تعليمـات الإجرائية يعاد التحكم إلى البرنامج الأصلي من نقطة استدعائه للإجرائية.

١٠-٣-٢- ملاحظات حول تعريف الإجرائية:

١- إن البنية العامة للإجرائية تشبه بنية البرنامج الرئيسي فتتألف مـن الأجزاء التالية:

١- رأس الإجرائية A Procedure Head.

٢- قسم التصريح A Declaration Section.

٣- جسم الإجرائية A Procedure Body.

٢- إن رأس الإجراء يبتدئ بالكلمة المحجوزة أو المميز المحجوز Procedure متبوعاً باسـم الإجراء. يمكن أن يتبع الاسم لائحة من الوسطاء ويجب أن ينتهي الإجراء بفاصلة منقوطة لتفصل عن باقي أجزاء الإجرائية.

٣- تملك كل إجرائية قسم تصريح خاص بها يتم التصريح فيها عن الثوابت أو المتحولات أو نعرف قيمة إجراءات وتوابع أخرى.

٤- يحتوي جسم الإجرائية على مجموعة من الأوامر الضرورية ليقوم الإجراء

بالعمل المطلوب منه ويبدأ جسم الإجراء بالمميز المحجوز Begin وينتهي بالمميز المحجوز .End متبوعاً بفاصلة منقوطة.

٥- يمكن أن تسبق الوسطـاء في الإجرائية بكلمة Var للتعريف بأنهـا وسطاء دخل وخرج معـاً. حيث يمكن التصريح عن عدة وسطاء ضمن الإجرائية الواحدة. ويمكن أن تكون هذه الوسطاء من نوع واحد أو من أنواع مختلفة.

١٠-٤- أمثلة:

مثال ١:

في مثال حساب مثلث باسكال من المرتبة N يمكن تقسيم العمل في هذا البرنامج إلى جزأين:

١- حسـاب المثلث مـن المرتبة N وإسناد الناتج إلى مصفوفـة P بواسطـة الإجرائية:

Procedure Pascal T (N: Integer; Var P: Matrix);

إن N هنا هو معامل دخل في حين P هو معامل دخل وخرج.

٢- كتابة مثلث باسكال من أجل مرتبة N من مصفوفة P ويتم بالإجرائية:

Procedure Writet (N: Integer; P: Matrix);

يجب الملاحظة هنـا أن كل من N و P هي معاملات دخل فقـط لأن المطلوب من هذه الإجرائية هو كتابة هذه العناصر لا تغيير قيمتها.

وعليه فإن أوامر البرنامج الرئيسي هي:

١- قراءة مرتبة المثلث.

٢- استدعاء الإجرائيين المذكورين باستخدام الوسطاء المناسبة.

إن البرنامج يكون بالشكل التالي:

```
PROGRAM PASCAL_TRIANGLE;
{COMPUTE PASCAL TRIANGLE WITH
SUBPROGRAM}
CONST
   MAXN=10;
TYPE
  MATRIX=ARRAY[1..MAXN,1..MAXN] OF
                                    INTEGER;
VAR
  P:MATRIX;
  N :INTEGER;
PROCEDURE PASCALT(N:INTEGER;VAR
P:MATRIX);
{COMPUTE PASCAL TRIANGLE OF ORDER N}
VAR
  I,J:INTEGER;
BEGIN
   FOR I:=1 TO N DO
      FOR J:=1 TO N DO
         P[I,J]:=0;
   FOR I:=1 TO N DO
      P[I,1]:=1;
   FOR I:=2 TO N DO
   FOR J:=2 TO I DO
      P[I,J]:=P[I-1,J]+P[J-1,J-1];
END;
PROCEDURE WRITET(N:INTEGER;P:MATRIX);
VAR
  I,J:INTEGER;
BEGIN
```



```
   FOR I:=1 TO N DO
    BEGIN
       FOR J:=1 TO I DO
          WRITE(P[I,J]:4);
       WRITELN;
    END;
END;
BEGIN
   WRITELN('INPUT N: N<=',MAXN);
   READLN(N);
   PASCALT(N,P);
   WRITET(N,P);
READLN
END.
```

مثال ٢:

ضرب مصفوفة بشعاع.

نعرف أنماط المتحولات المركبة اللازمة لمسألتنا:

```
Const
      Max N=10;
      Max N=10;
Type
      Matrix NM = Array [1.. Max N, 1.. Max N] of Reel;
      Vector = Array [1.. Max N] of Reel;
```

نقسم العمل في هذا البرنامج إلى الأقسام التالية:

١- قراءة المصفوفة.

٢- قراءة الشعاع.

٣- ضرب مصفوفة بشعاع.

٤- كتابة المصفوفة.

٥- كتابة الشعاع.

لدينا في هذا المثال الإجرائيات التالية:

١) إجرائية قراءة المصفوفة:

Procedure ReadMat (Var N,M: Integer; Var A: Matrix NM);

وهي تعني قراءة مصفوفة N,M عناصر هذه المصفوفة تخزن في المعامل A. حيث A هو معامل خرج .

٢) إجرائية قراءة شعاع:

Procedure ReadVect (M: Integer; Var X: Matrix);

أي قراءة شعاع X حجمه محدد دخل بالقيمة M والمعامل X هو معامل خرج.

٣) إجرائية الضرب:

Procedure MultMatVect (N,M: Integer; A: Matrix N,M;
X:Vector; Nar B: Vector):

وتعني ضرب مصفوفة A بشعاع X وعلماً أنه يجب أن تكون المصفوفات متوافقة بالنسبة لعملية الضرب هنا حجم A هو N,M وجسم الشعاع Xهو M. نتيجة الضرب سوف تخزن بالشعاع B كمعامل خرج.

٤) إجرائية طباعة المصفوفة:

Procedure WriteMat (N,M: Integer; A: Matrix NM);

كتابة مصفوفة A حجمها محدد كدخل بالقيم N,M.

٥) إجرائية طباعة شعاع:

Procedure WriteVect (M: Integer; Var X: Vector);

كتابة شعاع X حجمه محدد كدخل بالقيمة M.

اعتماداً على ما تقدم فإن البرنامج الرئيسي سوف يقوم باستدعاء كل تلك الإجرائيات.

- البرنامج بلغة تربو باسكال:

```
PROGRAM MATRIX_MULT_BY_PROCEDURES;
{MAKE MATRIX MULTIPLICATIONS:A*X=B USING
                              SUBPROGRAMS}
CONST
   MAXN=10;
   MAXM=10;
TYPE
 MATRIXNM=ARRAY[1..MAXN,1..MAXM] OF REAL;
 VECTOR=ARRAY[1..MAXM] OF REAL;
VAR
  A:MATRIXNM;
  X:VECTOR;
  B:VECTOR;
  N,M:INTEGER;
PROCEDURE READMAT(VAR N,M:INTEGER;
                          VAR A:MATRIXNM);
VAR
  I,J:INTEGER;
BEGIN
   WRITELN('INPUT N.M<=',MAXN,',',MAXM);
   READLN(N.M);
   WRITELN('INPUT MATRIX ', N ,'x' ,M);
   FOR I:=1 TO N DO
    BEGIN
```

```
END;
PROCEDURE
WRITEMAT(N,M:INTEGER;A:MATRIXNM);
VAR
  I,J:INTEGER;
BEGIN
    FOR I:=1 TO N DO
    BEGIN
       FOR J:=1 TO M DO
         WRITE(A[I,J]:8:3);   WRITELN;
    END;
END;
PROCEDURE WRITEVECT(M:INTEGER;
                                VAR X:VECTOR):
VAR
  J:INTEGER;
BEGIN
    FOR J:=1 TO M DO
    WRITELN(X[J]:8:3);
END;
BEGIN
  WRITELN('MATRIX BY VECTOR
                      MULTIPLICATION A*X=B');
  WRITELN('==============================');
  READMAT(N,M,A);  READVECT(M,X);
  MULTMATVECT(N,M,A,X,B); WRITELN('A');
  WRITEMAT(N,M,A);  WRITELN('X');
  WRITEVECT(M,X);  WRITELN('B');
  WRITEVECT(N,B);READLN
END.
```



```
      WRITELN('INPUT A[',I,'] ', M ,' ELEMENTS');
      FOR J:=1 TO M DO
        READ(A[I,J]);
      READLN;
    END;
END;
PROCEDURE READVECT(M:INTEGER;
                                VAR X:VECTOR);
VAR
  J:INTEGER;
BEGIN
    WRITELN('INPUT X');
    FOR J:=1 TO M DO
    BEGIN
       WRITELN('VECTOR[',J,']');
       READLN(X[J]);
    END;
END;
PROCEDURE
MULTMATVECT(N,M:INTEGER;A:MATRIXNM;
                      X:VECTOR;VAR B:VECTOR);
VAR
  I,J:INTEGER;
BEGIN
    FOR I:=1 TO N DO
    BEGIN
       B[I]:=0;
       FOR J:=1 TO M DO
         B[I]:=B[I]+A[I,J]*X[J];
    END;
```

١٠-٥- الفارق بين الإجرائية والتابع:

إن الفارق بين الإجرائية Procedure والتابع Function هــو أن التابع يركــز على إعادة قيمــة من نمط بسيط نسميها قيمة التابع، وهذا ما يمكننا من استخدام التابع في أي تعبير نمطه هو نمط قيمــة التابع. باستثناء هذا الفارق لا يوجد أي اختلاف في البنية بين التابع والإجرائية.

كقاعدة عامة يُنصح باستخدام التابع عندما لا يوجد لدينا إلا معامل خرج وحيد ومن نمط بسيط. أما إذا كان هناك أكثر من معامل خرج للبرنامج الجزئي فإننا نفضل استخدام الإجرائية وتكون قيمة التابع هي أحد معاملات الخرج.

مثال:

```
Function Fa (Var A: Char) : Integer;
```

يمكن كتابته كإجرائية:

```
Procedure Fa (Var A :Char; Var Fat: Integer);
```

حيث يتم جعل المعامل Fat يحل محل الإسناد إلى اسم التابــع (إن القيمة المعادة من التابع تتم بعملية إسناد إلى اسم التابع في جسم التابع).

١٠-٦- البرامج الجزئية العودية: Recusive Subprogram

في الخوارزمية العودية، يعاد حــل المسألة العامة إلى حــل المسألة في الحالة البسيطة.

في الرياضيات العلاقة المعرفــة في الحالة n كتعميم مــن التعريف في الحالة n-1 نسميها علاقة تراجع.

$$P_n = I\, P_{n-1}$$

لدينا لحساب عاملي عدد وليكن n العلاقة:

$$n! = n \times (n-1)!$$

هنا لا بد من معرفة الحالة الابتدائية وهي 1 = !0.

فــي مثال عاملي عدد، إن Fact (N) هــو التابع الذي يحسب !N لذلك يكفــي استدعاء Fact (N-1) ضمن هــذا التابع كي يُكتب التابــع Fact بشكل عمودي.

إن البرنامج الجزئي يكتب بالشكل التالي:

```
Function  Fact (N: Integer) : Integer;
  Var F: Integer;
  Begin
     If N=0 Then
          F := 1
     Else
          F := N*Fact (N-1);
     Fact := F;
End;
```

إن آلية التنفيذ تتم بالشكل التالي مثلاً من أجل N = 4 نجد:

```
4*Fact (3)
     3*Fact (2)
          2*Fact (1)
               1*Fact(0)
```

عندما يصل التراجع إلى Fact(0) يتوقف وتعوض القيمة بـــ 1 لأن:

```
If N=0 Then F := 1
```

ويتم التعويض بالسلسلة السابقة.

مثال:

اكتب برنامجاً لحساب $Q = \frac{n!}{m!(n-m)}$ باستخدام البرامج الجزئية لحساب عاملي عدد وذلك وفق علاقة تراجع (البرامج الجزئية العودية).

الحل:

- البرنامج بلغة تربو باسكال:

```
PROGRAM FACTORY;
VAR
  N,M:INTEGER;
  Q:REAL;
FUNCTION FACT(N:REAL):REAL;
VAR
   F:REAL;
BEGIN
  IF (N=0) THEN
    F:=1
 ELSE
   F:=N*FACT(N-1);
 FACT:  F;
END;
BEGIN
  WRITELN('INPUTE TWO NUMBERS N AND M ; M<N ');
  READLN(N,M);
  WRITE('THE FACTORIAL OF ', N,'  IS=');
  WRITELN(FACT(N):8:3);
  WRITELN('--------------------');
  WRITE('THE FACTORIAL OF ', M,'  IS=');
  WRITELN(FACT(M):8:3);
```



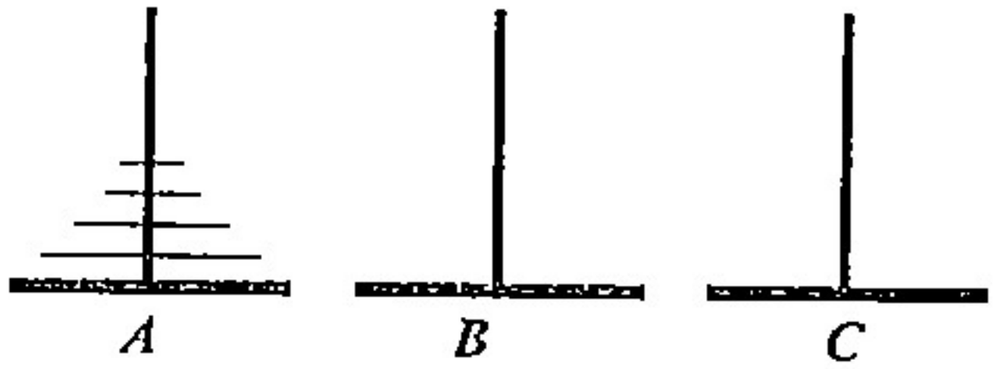
```
  WRITELN('--------------------');
  WRITE('SUM THE FACTORIAL OF TWO NUMBERS ',
                                  N,'  AND ', M,' IS=');
  WRITELN(FACT(N)+FACT(M):8:3);
  WRITELN('--------------------');
  WRITELN('DEVIDE THE FACTORIAL OF TWO NUMBERS
                                  ', N,'  AND ', M,' IS ');
  WRITE('FACT(N)/FACT(M)=');
  WRITELN(FACT(N)/FACT(M):8:3);
  WRITELN('--------------------');
  WRITE('FACT(N)/2=');
  WRITELN(FACT(N)/2:8:3);
  WRITELN('--------------------');
  WRITE('FACT(N*2)=');
  WRITELN(FACT(N*2):8:3);
  Q:=FACT(N)/(FACT(M)*FACT(N-M));
  WRITELN('--------------------');
  WRITE('C(',N,',',M,')=');
  WRITELN('FACT(',N,')/(FACT(',M,')*FACT(',N-M,')) =',
                                                  Q:5:2);
READLN
END.
```

١٠-٧-مسألة أبراج هانوي:The Towers Of Hanoi Problem

إن مسألة أبراج هانوي هي مسألة تقليدية عن البرامج الجزئية العودية.تنسب مسألته إلى المعابد الآسيوية في هانوي hanoi ،حيث يعمل رهبان المعابد على طرح مسألة علمية تقوم على نقل مجموعة من الأقراص(وليكن n قرصا)من قاعدة أولى ،إلى قاعدة أخرى،باستخدام قاعدة مساعدة أو

وسيطة.يمكن شرح المسألة بالشكل التالي:

يوجد ثلاث أعمدة (A,B,C) وثلاثة أقراص مختلفة الأبعاد متوضعة في العمـود (A) بحيث أن القرص الأكبر دائما تحت القرص الأصغر، والمطلوب تحريك الأقراص الثلاثة الى العمود (C) باسـتخدام العمود (B) كمساعد، يمكن تحريك فقط القمة في أي عمود الى عمود آخر، والقرص الأكبر ألا يتوضع أبدا فوق القرص الأصغر منه.أي أنه في أي وضع انتقالي يجب أن تكون الأقراص تشكل أبراجا هرمية على القواعد.

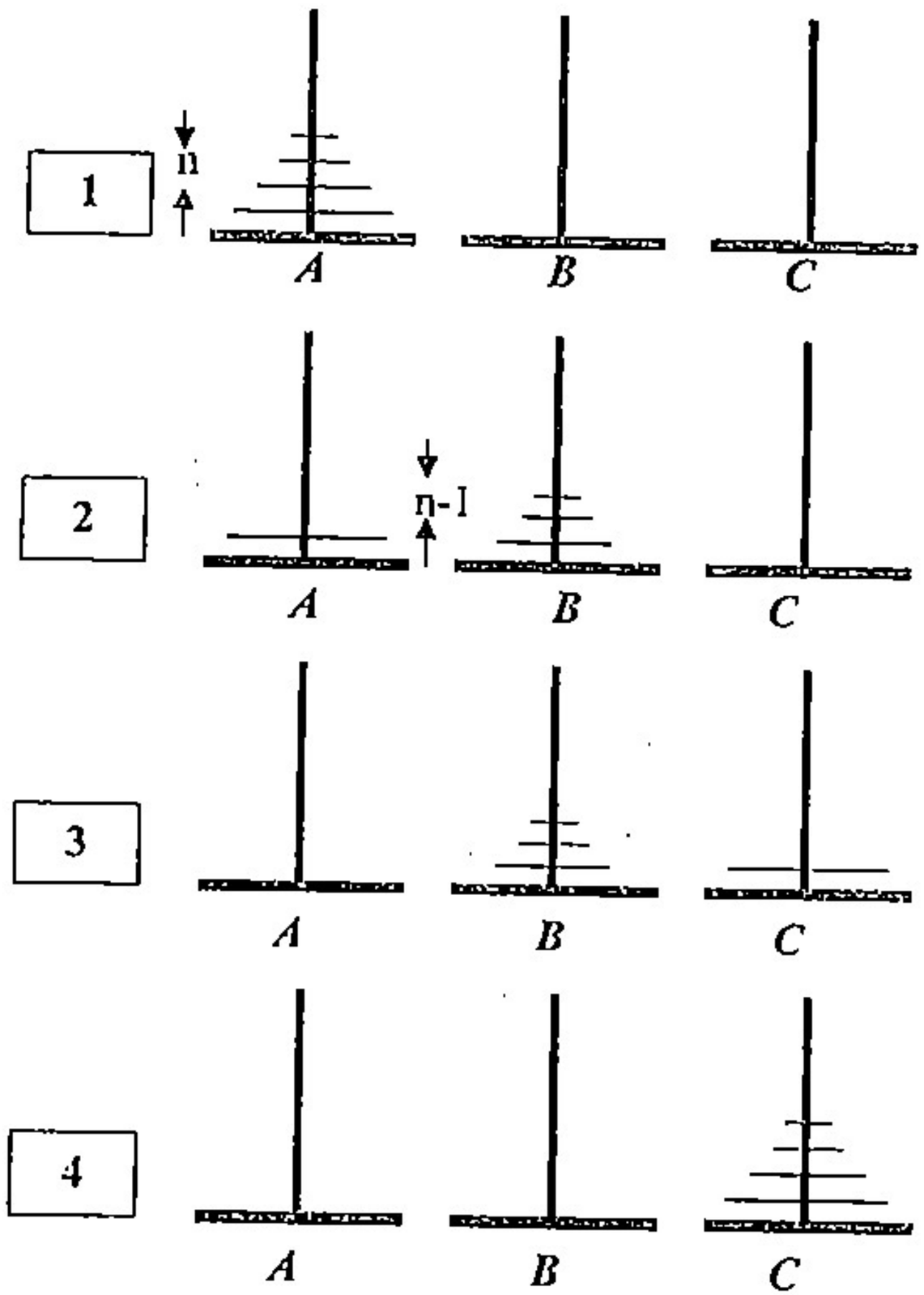


إن الحـل بديهي إذا كان عدد الأقراص قليلا إي من أجل N=1 أو من أجل N=2 ويكون الحل سهلا من أجل N=3 ،أما من أجل N أكبر من ذلك فأن الحـل يصـبح أكـثر تعقيدا،إن الحل باستخدام الحاسب ليس صعبا،وعليه فإن خوارزمية الحل تعطى كما يلي.

طريقة الحل (من أجل N قرص):

١.اذا كانت (N=1) عندئذ تحريك القرص من (A) الى (C) ثم التوقف.

٢.تحريك (N-1) قرص في القمـة من (A) الى (B) باسـتخدام (C) كعمـود مسـاعد. ٣.تحريك القرص المتبقي من (A) الى (C) . ٤.تحريك الـ (N-1) قـرص في القمـة من (B) الى (C) باستخدام (A) كعمود مساعد. إن الجدول التالي يشرح طريقة الحل:



يمـكن التعبير عن خوارزمية الحل إذا اعتبرنا الإجرائية HANQI(N,A,B,C) ذات المعاملات الأربعة N,A,B,C تقوم بنقل N قرصا مـن المعامل الثاني A إلى المعامـل الأخير C باستخدام المعامل الثالث B كعامل مسـاعد أو كوسيط. وبذلك يمـكن التعبير عن كل المراحل السابقة باسـتدعاء العملية HANOI من أجل N-1 ،سيتم العبير عن الحل بكتابة الأمر بنقل قرص من قاعدة إلى أخرى.

خوارزمية الحل:

إذا N=1 نفذ:

اكتب:"انقل قرص من"،A، "إلى"،C

وإلا

HANOI(N-1,A,C,B)

اكتب:"انقل قرص من"،A، "إلى"،C

HANOI(N-1,B,A,C)

مـن أجل كتابة البرنامج ننقل الخوارزمية إلى إجرائية بلغة تربو باسكال وعند التطبيق نعوض الأرقام 1,2,3 من أجل تمثيل الوسطاء A,B,C على التتالي.

أن الـبرنامـج التالي يسـأل عن عدد الأقراص المـطلوب نقلها ويحصي عدد الخطوات المطلوبه لإنجاز عملية النقل هذه.

البرنامج بلغة تربو باسكال:

```
PROGRAM HANOI_TOWER;
USES CRT;
VAR
    A,B,C,N,F:LONGINT;
```



```
PROCEDURE HANOI(N,A,B,C:INTEGER);
BEGIN
 IF(N=1)THEN
  BEGIN
    WRITELN('MOVE FROM ',A,' TO ',C);
    F:=F+1;
  END
 ELSE
  BEGIN
    HANOI(N-1,A,C,B);
    WRITELN('MOVE FROM ',A,' TO ',C);
    F:=F+1;
    HANOI(N-1,B,A,C);
  END;
END;

BEGIN
 CLRSCR;
 WRITELN('INPUT N THE NUMBER OF DISKS');
 READLN(N);
 A:=1;B:=2;C:=3;
 HANOI(N,A,B,C);
 WRITELN('NUMBER OF MOVES = ',F);
READLN;
END.
```

- من أجل N=2 نتيجة تنفيذ البرنامج تكون بالشكل التالي:

```
INPUT N THE NUMBER OF DISKS
2
```

```
MOVE FROM 1 TO 2
MOVE FROM 1 TO 3
MOVE FROM 2 TO 3
NUMBER OF MOVES = 3
```

- من أجل N=3 نتيجة تنفيذ البرنامج تكون بالشكل التالي:

```
INPUT N THE NUMBER OF DISKS
3
MOVE FROM 1 TO 3
MOVE FROM 1 TO 2
MOVE FROM 3 TO 2
MOVE FROM 1 TO 3
MOVE FROM 2 TO 1
MOVE FROM 2 TO 3
MOVE FROM 1 TO 3
NUMBER OF MOVES = 7
```

- من أجل N=4 نتيجة تنفيذ البرنامج تكون بالشكل التالي:

```
INPUT N THE NUMBER OF DISKS
4
MOVE FROM 1 TO 2
MOVE FROM 1 TO 3
MOVE FROM 2 TO 3
MOVE FROM 1 TO 2
MOVE FROM 3 TO 1
MOVE FROM 3 TO 2
MOVE FROM 1 TO 2
MOVE FROM 1 TO 3
```



```
MOVE FROM 2 TO 3
MOVE FROM 2 TO 1
MOVE FROM 3 TO 1
MOVE FROM 2 TO 3
MOVE FROM 1 TO 2
MOVE FROM 1 TO 3
MOVE FROM 2 TO 3
NUMBER OF MOVES = 15
```

وهكذا من أجل N=5 نحتاج إلى 31 نقلة ، وأيضا من أجل N=6 نحتاج إلى 63 نقلة .

وهو المطلوب.

-يمكن كتابة البرنامج بالشكل التالي:

برنامج تربو باسكال لحل خوارزمية هانوي من أجل N قرص:

```
Program  Towers_ Of_ Hanoi;
Var
   Numdisk:1..Maxint;
Procedure Move ( N,Start,End,Aux:Integer );
 Begin
   If N=1 Then
    Writeln ( 'Move Disk ',N,'From ',Start,'To',End);
  Else
   Begin
      Move(N-1,Start,Aux,End);
      Writeln('Move Disk',N,'From',Start,'To',End);
      Move (N-1,Aux,End,Start);
   End;
End;
```

```
Begin   {Main Program}
  Writeln('Enter Number Of Disks :');
  Readln(Numdisk);
  Move (Numdisk,1,3,2);
End. {Main Program}
```

عمل البرنامج من أجل (ثلاثة أقراص) يتم بالشكل التالي:

```
Enter Number Of Disks:  3
Move (3,1,3,2)

Procedure Move (N=3,Start=1,End=3,Aux=2)
Begin
If  N=1 false
Else Begin
Move(N-1=2,Start=1,Aux=2,End=3)

Procedure Move(N=2,Start=1,End=2,Aux=3)
Begin
If N=1 False
Else Begin
Move(N-1=1,Start=1,Aux=3,End=2)

Procedure Move(N=1,Start=1,End=3,Aux=2)
Begin
If N=1 Then
   Writeln(Move Disk /1/ from /1/to/3/)
End;
Writeln(Move Disk /2/ From/1/ To /2/)
Move (N-1=1,Aux=3,End=2,Start=1)
```



```
Procedure Move(N=1,Start=3,End=2,Aux=1)
Begin
If n=1 Then
   Writeln(Move Disk /1/ From /3/ To /2/)
End;
Writeln(Move Disk /3/ From/1/ To/3/)
Move (N-1=2,Aux=2,End=3,Start=1)

Procedure Move(N=2,Start=2,End=3,Aux=1)
If  N=1 False
Else  Begin
Move (N-1=1,Start=2,Aux=1,End=3)

Procedure Move (N=1,Start=2,End=1,Aux=3)
If  N=1 Then
  Writeln(Move Disk /1/From/2/To/1/)
End;
Writeln(Move Disk /2/From/2/To/3/)
Move(N-1=1,Aux=1,End=3,Start=2)

Procedure Move (N=1,Start=1,End=3,Aux=2)
If N=1 Then
   Writeln(Move Disk/1/From/1/ To /3/)
End;
End;
End.
```

نلاحظ في هذا المسألة أن برنامج حل هذه الخوارزمية قد قام بعملية الحل من خلال سبعة خطوات وذلك من أجل ثلاثة أقراص، وكلما ازداد عدد الأقراص كلما ازدادت عددُ خطوات الحل، وهذه الخطوات السبعة هي:

١.نقل القرص /١/من العمود الأول الى العمود الثالث.
٢.نقل القرص/٢/من العمود الأول الى العمود الثاني.
٣.نقل القرص/١/من العمود الثالث الى العمود الثاني.
٤.نقل القرص/٣/من العمود الأول الى العمود الثالث.
٥.نقل القرص/١/من العمود الثاني الى العمود الأول.
٦.نقل القرص/٢/ من العمود الثاني الى العمود الثالث.
٧.نقل القرص/١/من العمود الأول الى العمود الثالث.

وهو المطلوب.

١٠-٨- الإجرائيات والتوابع مسبقة التعريف:

تمتلك لغـــة باسكال كمية كبيرة من التوابع والإجرائيات مسبقة التعريف والتي يمكن استخدامها في برامجنا. وأضافت لغة تربو باسكال كميات أخرى من هذه التوابع والإجرائيات مسبقة التعريف.

سوف نتحدث في هذه الفقرة عن بعض هذه التوابع والإجرائيات مسبقة التعريف.

١٠-٨-١- التوابع والإجرائيات مسبقة التعريف في لغة باسكال القياسية:

إن التوابع والإجرائيات مسبقة التعريف في لغـــة باسكال القياسيـــة هي أيضاً معرفة في لغة تربو باسكال، لكن توجد بعض الاختلافات في طريقة عمل كلٍ منها.

مثلاً بعض هذه التوابع والإجرائيات تأخذ أي متحول صحيح، وهذا يعني في لغة تربو باسكال أننا نستطيع تمرير أحد أنواع المعطيات العددية التالية:

Longint, Word, Integer, Shortint, Byte

أما في لغة باسكال القياسية فإن هذا المتحول الصحيح سوف يكون فقط من النوع Integer وذلك لأن الأنواع الصحيحة الأخرى لم تكن معرفة في لغـــة باسكال القياسية.

نشرح فيما يلي بعض هذه التوابع والإجرائيات:

١- التابع ABS:

إن التابع ABS يعرف بالشكل:

```
Function ABS (I : Shortint) : Shortint;
Function ABS (I : Integer) : Integer;
Function ABS (I : Longint) : Integer;
Function ABS (I : Real) : Real;
```

إن التوابع السابقة تعيد القيمة المطلقة Absolute Value لمتحولاتها التي يجـــب أن تملك قيمة صحيحة موجبـــة أو سالبة أو حقيقية، وتعيد قيمة من نفس نوع المعطيات التي تنتمي إليها المتحولات. إن القيمة المطلقة لعدد هـــو الشكل الموجب لهذا العدد.

إن العبارتان التاليتان تعطيان نفس النتائج:

```
Q1 := ABS (-42.0);
Q2 := ABS (42.0);
```

إن الـــبرنامج يشـــرح التابع ABS حيث يقوم بإيجاد القيمـــة المـــطلقة لمجموعـــة مـــن الأعداد،ولا يخرج البرنامج حتى يعطى عددا أكبر من 1000

يستخدم إجرائية من أجل قراءة العدد.وفي متن البرنامج الرئيسي يتم إيجاد القيمة المطلقة.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM AbsDemo;
VAR
    RVal, Result : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
BEGIN
  WRITE ( Message, ' ');
  READLN ( Value);
END; (* GetReal *)
BEGIN  (* Main Program *)
 REPEAT
  GetReal ( 'Value (> 1000 to stop)? ', RVal);
  Result := ABS ( RVal);
  WRITELN ( 'ABS ( ', RVal : 8:3, ') = ', Result : 8 :
3);
 UNTIL RVal > 1000.0;
READLN;
END.
```

وهو المطلوب.

٢- التابع ARCTAN:

إن التابع ARCTAN يعرف بالشكل:

Function ARCTAN (V : Real) : Real;

يعيد التابع ARCTAN قيمة مقنية عبارة عن قيمـــة الزاوية التي ظلها

هو المتحول ضمن قوسين والذي يجب أن يكون قيمـــة حقيقية، ويعيد هذا التابع قيمة حقيقية أيضاً.

إن البرنامج التالي يشرح لنا عمل التابع ARCTAN حيث يقوم بإيجاد قيمة الزاوية التي ظلها معطى عبر المتحول المرر ضمن قوسين لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أكبر من 20 .يستخدم إجرائية من أجل قراءة العدد المرر.وفي متن البرنامج الرئيسي يتم إيجاد قيمة الزاوية.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM ArctanDemo;
VAR
  RVal, Result : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
 BEGIN
  WRITE ( Message, ' ');
  READLN ( Value);
 END; (* GetReal *)
BEGIN  (* Main Program *)
 REPEAT
  GetReal ( 'Value (> 20.0 to stop)? ', RVal);
  Result := ArcTan ( RVal);
  WRITELN ( 'ArcTan ( ', RVal : 8:3, ') = ', Result : 8
: 3);
 UNTIL RVal > 20.0;
 READLN;
END.
```

وهو المطلوب.

٣- التابع CHR:

يعرف التابع CHR بالشكل التالي:

Function CHR (OV : Byte) : Char;

يعيد التابع CHR الرمـز الذي حددت قيمة ترتيبه ضمـن القوسين أي يعطي شفرة ASCII لهذا الرمز.

إن البرنامج التالي يكتب عدة رمـوز على سطر واحـد باستخدام التابع CHR:

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM CHRDemo;
CONST
    CarriageReturn = 13;
    BackSpace = 8;
VAR
   Result : INTEGER;
BEGIN
  Result := 100;
  WRITE ( Result : 30);
  (* send a backspace, to move back one column, then
   write a new digit at the end of the number--- which is
                where the cursor is after the Backspace.*)
  WRITE ( CHR ( BackSpace), 1);
  (* send a carriage return but no linefeed,
     so cursor moves to start of same line. *)
  WRITE ( CHR ( CarriageReturn));
  WRITE ( Result : 20);
  WRITE ( CHR ( BackSpace), 2);
  WRITE ( CHR ( CarriageReturn));
```



```
  WRITE ( Result : 10);
  WRITE ( CHR ( BackSpace), 3);
  WRITE ( CHR ( CarriageReturn));
  READLN;
END.
```

وهو المطلوب.

إن البرنامج السابق أنجز مجموعة من الأوامر التالية:

١- كتب قيمة المتحول Result وهي هنـا 100 وفق عرض كبير للمجال ولكن دون أن ينقل المؤشر إلى بداية سطر جديد. وعرض المجال هذا أخذ بالتناقص في كل مرة كتبت فيها قيمة المتحول Result ضمن البرنامج.

٢- كتب رمز الحذف الخلفي Backspace والذي شفرة ASCII المقابلة له هي 8 وهذا الرمز ينقل المشيرة Cursor عموداً واحـداً إلى الخلف فوق آخر رقم من أرقام العدد 100.

٣- كتب رقماً جديداً عوضاً عن الرقم الأخير في العدد 100 الذي حذفه.

٤- كتب رمز الإرجاع Carriage return والذي شفرة أسكي المقابلة له هي 13 ، هـذا الرمـز يرجع المشـيرة إلى أول عمود ضمن السـطر المحدد، أي السطر الذي عليه موجودة المشيرة. وحتى ننتقل إلى سـطر جديد نحن بحاجة إلى رمز التغذية السطرية Linefeed، الذي ينقل المشيرة إلى سطر جديد.

٤- التابع COS:

يعرف التابع COS بالشكل:

Function COS (A : Real) : Real;

يعيد هذا التابع تجيب الزاوية المحددة بالمتحول الحقيقي الموجود ضمن قوسين إن قيمة التابـع محدودة ضمن المجال [1+ ,1-]. إن قيمة المتحول يجب

أن تكون مقدرة بالراديان وليس بالدرجات، تعلم أن 90 تكافئ $\frac{\pi}{2}$ و 360 تكافئ 2π يعني هذا أن 1 راديان هو حوالي 57 درجة.

إن البرنامج التالي يفحص عمل التابع COS حيث إن البرنامج التالي يقوم بإيجاد قيمة تجيب الزاوية المعطاة عبر المتحول المرر ضمن قوسين وذلك لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أكبر من 200 .يستخدم إجرائية من أجل قراءة العدد الرسالة والعدد المرر.وفي متن البرنامج الرئيسي يتم إيجاد قيمة تجيب الزاوية علما أن الزاوية يجب أن تكون معطاة بالراديان.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM CosDemo;
VAR
  Cosine, Angle : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
BEGIN
   WRITE ( Message, ' '); READLN ( Value);
END; (* GetReal *)
BEGIN
 REPEAT
   GetReal ( 'Angle (in radians; > 200.0 to stop)?',
Angle);
   Cosine := COS ( Angle);
   WRITELN ( 'Cosine of ', Angle : 6 : 4,
         ' radians = ', Cosine : 6 : 4);
 UNTIL Angle > 200.0; READLN;
END.
```

وهو المطلوب.

٥- التابع EXP:

يعرف التابع EXP بالشكل:

```
Function EXP (R : Real) : Real;
```

يعيد هذا التابع ناتج رفع العدد النبري e إلى قيمة المتحول ضمن قوسين لهذا التابع حيث e = 2.7181628.

ومن أجل توضيح عمل هذا التابع نكتب البرنامج التالي الذي يحسب التابع EXP من أجل القيم التي ندخلها.

إن البرنامج التالي يقوم بإيجاد قوة عدد مرفوع للأساس e عبر المتحول المرر ضمن قوسين وذلك لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أكبر من 15 .يستخدم إجرائية من أجل قراءة الرسالة والعدد المرر.وفي متن البرنامج الرئيسي يتم إيجاد قيمة القوة.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM ExpDemo;
VAR
  Val, Result : REAL;
PROCEDURE GetReal ( Message : STRING;
              VAR Value : REAL);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)

BEGIN
 REPEAT
   GetReal ( 'Value? (> 15 to stop) ', Val);
```

```
   Result := EXP ( Val);
   WRITELN ( 'EXP (', Val : 10 : 3, ') = ', Result : 10 :
                                                    3);
 UNTIL Val > 15.0;
 READLN;
END.
```

وهو المطلوب.

٦- التابع LN:

يعرف التابع LN بالشكل:

Function LN (V : Real) : Real;

يعيد هـذا التابع قيمة اللوغاريتم الطبيعي (أي الذي أساسه e) للمتحول الحقيقي ضمن قوسين قيمة هذا المتحول يجب أن تكون أكبر تماماً مـن (0.0). إن اللوغاريتم الطبيعي لعدد هو قوة العدد النيبري e والذي يعطينا بعد رفع العدد e إليه العدد المطلوب حساب اللوغاريتم الطبيعي له.

إن $e2 \cong 7.389056$ حيث $LN(7.389056) \cong 2$ أي يجب رفـع e للقوة 2 حتى نحصل على 7.389056. إن البرنامج التالي يشرح عمـل التابعين LN و EXP.

إن البرنامج التالي يقوم بإيجاد قوة عدد مرفوع للأساس e عبر المتحول المرر ضمن قوسين وأيضا لغاريتم العدد الناتج من العملية السابقة وذلك لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أصغر من 0 .يستخدم إجرائية من أجل قراءة الرسالة والعدد المرر.وفي متن البرنامج الرئيسي يتم إيجاد قيمة القوة،ثم لغاريتم العدد الناتج.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:


```
PROGRAM ExpAndLnDemo;
VAR
  Val, Result, LResult : REAL;
PROCEDURE GetReal ( Message : STRING;
              VAR Value : REAL);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)

BEGIN
 REPEAT
   GetReal ( 'Value? (< 0 to quit)', Val);
   Result := Exp ( Val);
   WRITELN ( 'EXP (', Val : 10 : 3, ') = ', Result : 10 :
                                                    3);
   LResult := LN ( Result);
   WRITELN ( 'LN  (', Result : 10 : 3, ') = ',
                                        LResult:10:3);
 UNTIL Val < 0.0;
 READLN;
END.
```

وهو المطلوب.

٧- التابع ODD:

يعرف التابع ODD بالشكل:

Function ODD (V : Longint) : Boolean;

يعيد التابع ODD القيمة True إذا كان العدد ضمن قوسين فردياً، وإلا فإنه يعيد القيمة False (أي إذا كان العدد زوجياً). إن المتحول ضمن قوسين يجب أن يكون عدداً صحيحاً.

إن البرنامج التالي يوضح عمل التابع ODD، إذ يولد البرنامج قيمة عشوائية صحيحة، فإذا كانت هذه القيمة فردية فسيكتب نقطة، أما إذا كانت هذه القيمة زوجية فسيكتب فراغاً، ويكرر هذه العملية 1840 مرة.

يمكن تنفيذ البرنامج عدة مرات مع تغير مجال القيم العشوائية، بوضع مجال أكبر من 5000 في المتحول MaxTrials وملاحظة النتائج.

يمكن الملاحظة أن نسبة الأعداد الفردية تقترب من 50 بالمئة والسبب في ذلك أن القيم الحقيقية العشوائية المتولدة سوف تكون وفق أحد احتمالين لا ثالث لهما وهي إما فردية أو زوجية، إن احتمال كون القيمة المتولدة فردية هو 50 بالمئة.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM OddDemo;
CONST
   MaxTrials = 1840;
VAR
  Count, OddCount, MaxVal, Result : INTEGER;

PROCEDURE GetInteger ( Message : STRING;
                       VAR Value : INTEGER);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetInteger *)
```



```
BEGIN
  OddCount := 0;

  RANDOMIZE;
  GetInteger ( 'Largest value to generate? (< 0 for
                                    MAXINT)',MaxVal);
  IF MaxVal < 0 THEN
       MaxVal := MAXINT;
  FOR Count := 1 TO MaxTrials DO
  BEGIN
   Result := RANDOM ( MaxVal);
   (* Note that function can be used wherever
                              a BOOLEAN is valid.*)
   IF ODD ( Result) THEN
   BEGIN
    WRITE ( '.');
    (* Increment the odd value counter by 1 *)
    INC ( OddCount);
   END
   ELSE
    WRITE ( ' ');
 END;
(*determine and display the proportion of odd values*)
 WRITE ( OddCount, ' odd values out of ', MaxTrials);
 WRITELN ( '; (', (OddCount / MaxTrials) * 100 :5:2,
                                                '%)');
 READLN;
END.
```

وهو المطلوب.

٨- التابع ORD:

إن التابع ORD يأخذ الشكل التالي:

```
Function ORD (V : Byte) : Longint;
Function ORD (V : Shortint) : Longint;
Function ORD (V : Integer) : Longint;
Function ORD (V : Word) : Longint;
Function ORD (V: Longint) : Longint;
Function ORD (V: Char) : Longint;
Function ORD (V: Boolean) : Longint;
```

يعيد هذا التابع قيمة ترتيب المتحول الموجود ضمن قوسين والذي ينتمي إلى أحد المعطيات الترتيبية التي هنا هي: Shortint, Integer, Word, Longint, Char, Boolean. أي أنه يمكن أن ينتمي إلى أحد أنواع المعطيات التالية:

١- أحد أنواع المعطيات الصحيحة.

٢- نوع المعطيات محرف.

٣- نوع المعطيات بولياني.

٤- نوع المعطيات التعدادي.

من أجل نوع المعطيات محرف CHAR تكون القيمة المعادة بواسطة هذا التابع هي قيمة شفرة أسكي المقابلة لقيمة المتحول الوسيطي. أما إذا كان المتحول الوسيطي من أحد أنواع الأعداد الصحيحة فإن القيمة التي يعيدها هذا التابع هي نفس قيمة المتحول الوسيطي. وإذا كان المتحول الوسيطي بولياناً فإن القيمة التي يعيدها إما 0 إذا كانت قيمة المتحول False أو 1 إذا كانت قيمة المتحول True. أما إذا كان المتحول من النوع التعدادي فأن قيمة هذا التابع

ستساوي إلى موقع قيمة هذا المتحول ضمن اللائحة التعدادية مطروحاً منه القيمة ١ علماً أن قيمة ترتيب أول عنصر في اللائحة التعدادية هو 0 وقيمة ترتيب العنصر الثاني في اللائحة هو ١.

إن البرنامج التالي يشرح استخدام التابع ORD.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM OrdTest;
TYPE
   MyType = ( vala, valb, valc);
VAR
   I1 : INTEGER;
   C1 : CHAR;
   B1 : BOOLEAN;
   M1 : MyType;
BEGIN
  I1 := -1;  C1 := 'a';  B1 := FALSE;
  (* for enumerated types, first element has ordinal
value 0, second element has ordinal value 1, etc. *)
  M1 := valc;
  WRITELN ( 'ORD ( I1) = ', ORD ( I1));
  (* returns -1 *)
  WRITELN ( 'ORD ( C1) = ', ORD ( C1));
(* returns 97 *)
  WRITELN ( 'ORD ( B1) = ', ORD ( B1));
(* returns 0 *)
  WRITELN ( 'ORD ( M1) = ', ORD ( M1));
(* returns 2 *)
  READLN;
END.
```

وهو المطلوب.

٩- التابع PRED:

يعرف التابع PRED بالشكل:

```
Function PRED (V : Byte) : Byte;
Function PRED (V : Shortint) : Shortint;
Function PRED (V : Integer) : Integer;
Function PRED (V : Word) : Word;
Function PRED (V: Longint) : Longint;
Function PRED (V: Char) : Char;
Function PRED (V: Boolean): Boolean;
```

يعيد هـذا التابع القيمة السابقة لقيمة المتحول ضمن قوسين الذي ينتمي إلى أحد الأنواع الترتيبية التي ذكرناها. ناتج هذا التابع سيكون غير معـرف إذا استدعينا هـذا التابع من أجل أصغر قيمة في هذه الأنواع المرتبة. أي إذا أردنا من التابع أن يجلب القيمة السابقة لأول قيمة مـن النوع التعدادي فإن ناتج هذا التابع سيكون غير معرف.

إن البرنامج التالي يشرح كيفية استخدام تابع القيمـة السابقـة PRED بالشكل:

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM PredDemo;
TYPE
   MyType = ( vala, valb, valc);
VAR
   I1 : INTEGER;
   C1 : CHAR;
   B1 : BOOLEAN;
```



```
   M1 : MyType;
BEGIN
 I1 := -32768;
 C1 := 'q';
 B1 := FALSE;
 (* for user defined types, first element has
       ordinal value 0, second element has
                  ordinal value 1, etc. *)
 M1 := valc;
 WRITELN ( 'PRED ( I1) = ', PRED ( I1));
 WRITELN ( 'PRED ( C1) = ', PRED ( C1));
 WRITELN ( 'PRED ( B1) = ', PRED ( B1));
 (* You can't write the value of an enumerated type;
  one way to determine whether the PRED did anything
   is to write the new ordinal value.Since ORD ( valc)
                  = 2, the new value should be 1.*)
 WRITELN ( 'PRED ( M1) = ', ORD ( PRED ( M1)));
 READLN;
END.
```

إن نتيجة تنفيذ البرنامج سوف تكون بالشكل التالي:

```
PRED ( I1) = 32767
PRED ( C1) = p
PRED ( B1) = TRUE
PRED ( M1) = 1
```

وهو المطلوب.

١٠- الإجرائيتان READ و READLN:

تعرف الإجرائيتان READ و READLN بالشكل:

```
Procedure READ;
```

Procedure READLN;

تستخـــدم الإجرائيتان والتي عرفنـــاهـــا في برامجنـــا سابقـــاً READ و READLN لقراءة المعلومات من ملف أو من جهاز الدخل القياسي أي لوحة المفاتيح.

١١- التابع ROUND:

يعرف التابع ROUND بالشكل:

Function ROUND (V: Real) : Longint;

يُحـــول هـــذا التابع قيمة حقيقية إلى قيمة مـــن النوع الصحيح الطويل Longint وذلك بواسطة تدوير القسم الكسري وقد تحدثنا عـــن هذا التابع سابقاً بالتفصيل.يمكن فحص هذا التابع بتنفيذ البرنامج.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM RoundDemo;
VAR
  RVal, Result : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)

BEGIN  (* Main Program *)
 REPEAT
   GetReal ( 'Value (> 500 to stop)? ', RVal);
   Result := ROUND ( RVal);
```



```
   WRITELN ( 'ROUND ( ', RVal : 8:3, ') = ', Result : 8
                                              : 3);
 UNTIL RVal > 500.0;
 READLN;
END.
```

وهو المطلوب.

١٢- تابع SIN:

يعرف التابع SIN بالشكل:

Function SIN (Radians: Real) : Real;

يعيد هذا التابع جيب الزاوية المحددة بالمتحول ضمن قوسين الذي يجب أن يكون متحولاً حقيقياً ويجب أن تكون قيمة الزاويـــة مقاسه بالراديان وليست بالدرجات. إن قيم هذا التابع أو ناتجه محصور ضمـــن المجال [1.0+ ,1.0-].

إن البرنامج التالي يوضح عمل التابع SIN من أجل قيم مختلفة لزاوية بين 0 و 2π-

إن الـــبرنامج التالي يقوم بإيجاد قيمة جيب الزاوية المعطاة عبر المتحول المرر ضمن قوسين وذلك لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أكبر من 200 .يستخدم إجرائية من أجل قراءة العدد المرسالة والعدد المرر وفي متـــن الـــبرنامج الرئيسي يتم إيجاد قيمة جيب الزاوية علما أن الزاوية يجب أن تكون معطاة بالراديان.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM SinDemo;
VAR
  Sine, Angle : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
```

```
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)
BEGIN
 REPEAT
   GetReal ( 'Angle (in radians; > 200 to stop)?',
Angle);
   Sine := SIN ( Angle);
   WRITELN ( 'Sine of ', Angle : 6 : 4,
         ' radians = ', Sine : 6 : 4);
 UNTIL Angle > 200;
 READLN;
END.
```

وهو المطلوب.

١٣- التابع SQR:

يعرف التابع SQR بالشكل:

```
Function SQR (V : Shortint) : Shortint;
Function SQR (V : Integer) : Integer;
Function SQR (V: Longint) : Longint;
Function SQR (V: Real) : Real;
```

يعيد هذا التابع ناتج تربيع قيمة المتحول الوسيطي، أي يعيد الناتج V*V الذي سيكون من نفس نوع المعطيات الذي ينتمي إليه المتحول ضمن قوسين. أي إما نوع صحيح أو حقيقي.

إن البرنامج التالي يوضح لنا عمل التابع SQR.

إن البرنامج التالي يقوم بإيجاد مربع عدد عبر المتحول المرر ضمن قوسين وذلك لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أصغر من 500. يستخدم إجرائية من أجل قراءة الرسالة والعدد المُرر.وفي متن البرنامج الرئيسي يتم إيجاد قيمة مربع العدد.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM SqrDemo;
VAR
  LVal, Result : LONGINT;
PROCEDURE GetLongInt ( Message : STRING;
                VAR Value : LONGINT);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetLongInt *)
BEGIN  (* Main Program *)
 REPEAT
   GetLongInt ( 'Value (< -500 to stop)? ', LVal);
   Result := SQR ( LVal);
   WRITELN ( 'SQR ( ', LVal:10, ') = ', Result : 15);
 UNTIL LVal < -500.0;
 READLN;
END.
```

وهو المطلوب.

- يستخدم البرنامج التالي التابع SQR لتوضيح العلاقة بين تابع الجيب Sine وتابع التجيب Cosine.

البرنامج يعطي قيمة جيب زاوية وقيمة تجيب الزاوية ثم قيمة مجموع مربع الجيب والتجيب،من أجل توضيح العلاقة بين الجيب والتجيب.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM SqrAndTrigDemo;
VAR
   Sine, Cosine, Angle : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)
BEGIN
 WRITELN ( ' SIN(x) ', ' COS(X) ',
       'SQR(Sine) + SQR(Cosine)');
 WRITELN;
 REPEAT
  GetReal ( 'Angle (> 100 to stop)?', Angle);
  Sine := SIN ( Angle);
  Cosine := COS ( Angle);
  WRITELN ( Sine : 10 : 4, Cosine : 10 : 4,
        SQR ( Sine) + SQR ( Cosine) : 17 : 4);
 UNTIL Angle > 100;
 READLN;
END.
```

وهو المطلوب.

١٤- التابع SQRT:

يعرف التابع SQRT بالشكل:

```
Function SQRT (V : Shortint) : Real;
Function SQRT (V : Integer) : Real;
```



```
Function SQRT (V: Longint) : Real;
Function SQRT (V: Real) : Real;
```

يعيد هذا التابع الجذر التربيعي لقيمة المتحول ضمن قوسين والذي يجب ان تكون قيمته غير سالبة. لأن هذا المتحول يجب أن يكون من أي نوع صحيح أو من النوع الحقيقي ولكن ناتج هذا التابع دائماً من النوع الحقيقي.

إن البرنامج التالي يوضح عمل التابع SQRT:

إن البرنامج التالي يقوم بإيجاد جذر عدد عبر المتحول المرر ضمن قوسين وذلك لمجموعة من الأعداد،ولا يخرج البرنامج حتى يعطى عددا أصغر من 0 .يستخدم إجرائية من أجل قراءة الرسالة والعدد المرر،وفي متن البرنامج الرئيسي يتم إيجاد قيمة جذر العدد.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM SqrtDemo;
VAR
  RVal, Result : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)
BEGIN  (* Main Program *)
 GetReal ( 'Value (< 0 to stop)? ', RVal);
 WHILE RVal > 0 DO
 BEGIN
  Result := SQRT ( RVal);
  WRITELN ( 'SQRT ( ', RVal : 8:3, ') = ', Result : 8 :
3);
```

```
  GetReal ( 'Value (< 0 to stop)? ', RVal);
 END; (* WHILE RVal > 0 *)
END.
```

وهو المطلوب.

١٥- التابع SUCC:

يعرف التابع SUCC بالشكل:

```
Function SUCC (V : Byte) : Byte;
Function SUCC (V : Shortint) : Shortint;
Function SUCC (V : Integer) : Integer;
Function SUCC (V : Word) : Word;
Function SUCC (V: Longint) : Longint;
Function SUCC (V: Char) : Char;
Function SUCC (V: Boolean): Boolean;
```

يعيد هـذا التابع القيمة التالية لقيمة المتحول ضمن قوسين والذي ينتمي إلى أحد أنواع المعطيات المرتبة المعروفة لدينـا سابقاً. إن ناتج هـذا التابع لن يكون معرفـاً إذا طلبنا منه أن يحصر لنا القيمة التالية لآخر قيمة مـن قيم نوع المعطيات الذي ينتمي إليه هذا المتحول. إن البرنامج التالي يشرح لنا تابع القيمة اللاحقة المسمى التابع SUCC.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM SuccDemo;
TYPE
  MyType = ( vala, valb, valc);
VAR
  I1 : INTEGER;
  C1 : CHAR;
```



```
  B1 : BOOLEAN;
  M1 : MyType;
BEGIN
 I1 := 32767;
 C1 := 'q';
 B1 := TRUE;
 (* for user defined types, first element has ordinal
   value 0,second element has ordinal value 1, etc.*)
 M1 := valc;
 WRITELN ( 'SUCC ( I1) = ', SUCC ( I1));
 WRITELN ( 'SUCC ( C1) = ', SUCC ( C1));
 WRITELN ( 'SUCC ( B1) = ', SUCC ( B1));
 (* You can't write the value of an enumerated type;
 one way to determine whether the SUCC did anything
is to write the new ordinal value. Since ORD ( valc) =
2, the new value should be 3, although there is no
          enumerated value with this ordinal value.*)
 WRITELN ( 'SUCC ( M1) = ', ORD ( SUCC ( M1)));
 READLN;
END.
```

إن نتيجة تنفيذ البرنامج سوف تكون بالشكل التالي:

```
SUCC ( I1) = -32768
SUCC ( C1) = r
SUCC ( B1) = TRUE
SUCC ( M1) = 3
```

وهو المطلوب.

١٦- التابع TRUNC:

يعرف التابع TRUNC بالشكل:

Function TRUNC (V: Real) : Longint;

يحول هـــذا التابع القيمة الحقيقية إلى قيمة صحيحة طويلة وذلك بحذف الجزء الكسري من هذه القيمة. إن ناتج هذا التابع هو من النوع الصحيح الطويل Longint في لغة تربـــو باسكال، وهو مـــن النوع الصحيح Integer في لغـــة باسكال القياسية. وقد شرحنا سابقاً عن عمل هذا التابع.

إن التابعان TRUNC و ROUND يعملان على تحويل قيمة حقيقية إلى قيمة صحيحة. وللمقارنة بين عمـــل كل مـــن التابعين TRUNC و ROUND نكتب البرنامج التالي:

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM TruncDemo;
VAR
  RVal, Result : REAL;
PROCEDURE GetReal ( Message : STRING:
                              VAR Value : REAL);
BEGIN
   WRITE ( Message, ' '): READLN ( Value):
END: (* GetReal *)
BEGIN  (* Main Program *)
 REPEAT
   GetReal ( 'Value (> 500 to stop)? ', RVal):
   Result := TRUNC ( RVal);
   WRITELN ( 'TRUNC ( ', RVal : 8:3, ') = ', Result : 8
                                          : 3);
   Result := ROUND ( RVal);
   WRITELN ( 'ROUND ( ', RVal : 8:3, ') = ', Result : 8
                                          : 3);
 UNTIL RVal > 500.0;  READLN;
END.
```


وهو المطلوب.

١٧- الإجرائيتان WRITE و WRITELN:

تعرف الإجرائيتان WRITE و WRITELN بالشكل:

Procedure WRITE;

Procedure WRITELN;

وتستخدم للكتابة على الشاشة أو على ملف. وتحدثنا عنهما سابقـــاً بشكل مفصل.

١٠-٨-٢- الإجرائيات والتوابع مسبقة التعريف في لغة تربو باسكال:

الإجرائيات والتوابع المعرفة في هـــذه الفقرة هي برامج جزئية مسبقـــة التعريف في لغة تربـــو باسكال فقط وليست معرفة في لغة باسكال القياسية.

إن أغلـــب هـــذه التوابع والإجرائيات هي مـــن أجل معالجـــة السلاسل المحرفية.

١- التابع CONCAT:

يعرف هذا التابع بالشكل:

Function CONCAT (S1, ..., Sn: String): String;

ينشئ هـــذا التابع سلسلة محرفية طويلة بعـــد عملية ضم لكل متحولات السلاسل الممرة إليـــه ضمن قوسين، ويعيد بعدهـــا سلسلة محرفية طويلة. فعند تمرير سلسلتين محرفيتين هما S1 و S2 إلى هذا التابع فإن ناتج تنفيذ هذا التابع هو عبارة عن سلسلة محرفية هي عبارة عن مجموع أو اتحاد هاتين السلسلتين، إذ توضع السلسلة S1 أولاً وتليها السلسلة S2. يمكن لهذا التابع أن يأخذ سلسلتين أو أكثر كمتحولات وسيطية لضمها وجعلها سلسلة محرفية واحدة.

```
  RVal, Result : REAL;

PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);

BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)
BEGIN  (* Main Program *)
 REPEAT
   GetReal ( 'Value (> 500 to stop)? ', RVal);
   Result := FRAC ( RVal);
   WRITELN ( 'FRAC ( ', RVal : 8:3, ') = ', Result : 8 :
                                            3);

 UNTIL RVal > 500.0;
 READLN;
END.
```

وهو المطلوب.

٣- التابع INT:

يعرف التابع INT بالشكل:

Function INT (V :Real) : Real;

يعيد هذا التابع القسم الصحيح من العدد الحقيقي V الممر إليه ضمن قوسين، ولكن يعيده عدداً حقيقياً. فمثلاً إذا كانت قيمة المتحول MR هي 24.678 فإنه سوف تسند العبارة التالية القيمة 24.0 بالمتحول RV.

RV := INT (MR);

يمكن تنفيذ البرنامج التالي ومقارنة خرج التابعين FRAC و INT.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM IntDemo;
VAR
  RVal, Result : REAL;
PROCEDURE GetReal ( Message : STRING;
                    VAR Value : REAL);

BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetReal *)
BEGIN  (* Main Program *)
 REPEAT
   GetReal ( 'Value (> 500 to stop)? ', RVal);
   Result := INT ( RVal);
   WRITELN ( 'INT  ( ', RVal : 8:3, ') = ', Result : 8 : 3);
   Result := FRAC ( RVal);
   WRITELN ( 'FRAC ( ', RVal : 8:3, ') = ', Result : 8 : 3);
 UNTIL RVal > 500.0;
 READLN;
END.
```

وهو المطلوب.

٤- التابع LENGTH:

يعرف التابع LENGTH بالشكل:

Function LENGTH (S :String) : Integer;

يعيد هذا التابع طول السلسة المحرفية S. فإذا كانت قيمة المتحول MS هي "This is" عندئذ سوف تُلحق القيمة 7 بالمتحول الصحيح RV.

RV := LENGTH (MS);

إن البرنامج التالي يشرح عمل التابع LENGTH:

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM LengthDemo;
CONST
   EmptyString = '';
VAR
   SVal : STRING;
   Result : INTEGER;
PROCEDURE GetString ( Message : STRING;
                      VAR Value : STRING);
BEGIN
   WRITE ( Message, ' ');
   READLN ( Value);
END; (* GetString *)
BEGIN  (* Main Program *)
  GetString ( 'String (ENTER to stop)? ', SVal);
  WHILE SVal <> EmptyString DO
  BEGIN
    Result := LENGTH ( SVal);
    WRITELN ( SVal, ' is ', Result:2, ' characters long');
    GetString ( 'String (ENTER to stop)? ', SVal);
  END; (* WHILE SVal <> EmptyString *)
END.
```

وهو المطلوب.

٥- التابع PI:

يعرف التابع PI بالشكل:

```
Function PI : Real;
```


يعيد هذا التابع القيمة π والتي تستوي تقريباً 3.14156265 وهذا التابع مفيد عند استخدام التوابع المثلثية Cos و Sin.

٦- التابع RANDOM:

يعرف التابع RANDOM بالشكل:

```
Function RANDOM: Real;
Function RANDOM (Max Val : Word) : Word;
```

يعيد هـذا التابع قيمة شبه عشوائية. يمكن استدعاء هـذا التابـع دون محولات وسيطية، عندئذ يعيد هـذا التابع قيمة حقيقية شبه عشوائية بين (0.0) و (1.0) أما إذا استدعينا التابع السابق مـع متحول من النوع Word فإن التابع سوف يولد قيمة عشوائية بين (0) وأكبر قيمـة مخصصة له. أي القيمة الممررة له ضمن المتحول الوسيطي بين قوسين.

٧- الإجرائية RANDOMIZE:

تعرف بالشكل:

```
Procedure RANDOMIZE;
```

تهيئ هذه الإجرائية مولد الأعداد العشوائية مع بذور قيم عشوائية، وهذه البذور تخزن ضمن المتحول المعرف بشكل مسبق Randseed.

يجـب استدعاء هـذه الإجرائية في بداية أي برنـامج يستخدم الأعداد العشوائية بواسطة التابع RANDOM. أمـا إذا لـم نستدع هذه الإجرائية فإن تسلسل الأعداد العشوائية المتولدة سيعتمد على القيمـة المخزنة ضمن المتحول Randseed (يـبقى نفسه). أما إذا أردنا توليد أعداداً لها نفس التسلسل العشوائي فـي كل تنفيذ للبرنامج علينا تهيئة المتحول Randseed بنفس القيمة عوضاً عن استدعائنا للإجرائية RANDOMIZE.

٨- التابع SIZEOF:

يعرف بالشكل:

Function SIZEOF (<Type or Variable Name>): Word;

يعيد هذا التابع كمية الذاكرة المخصصة لمتحول أو لنوع معطيات. وهذا الناتج يمثل عدد البايتات المخصصة للمتحول.

إن البرنامج التالي يوضح كميـــة الذاكرة المخصصة لكل نوع من أنواع المعطيات التي درسناها حتى الآن:

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
PROGRAM SizeDemo;
VAR
  I1 : INTEGER;
  By1 : BYTE;
  S1 : SHORTINT;
  L1 : LONGINT;
  W1 : WORD;
  C1 : CHAR;
  Bo1 : BOOLEAN;
  Stg : STRING;   R1 : REAL;
BEGIN
 WRITELN ( 'INTEGER : ', SIZEOF ( INTEGER) : 3,
       ' bytes; ', '  I1 : ', SIZEOF ( I1) : 3, ' bytes');
 WRITELN ( 'BYTE   : ', SIZEOF ( BYTE) : 3,
     ' bytes; ', '  By1: ', SIZEOF ( By1) : 3, ' bytes');
 WRITELN ( 'SHORTINT: ', SIZEOF ( SHORTINT) : 3,
         ' bytes; ','  S1 : ', SIZEOF ( S1) : 3, ' bytes');
 WRITELN ( 'LONGINT : ', SIZEOF ( LONGINT) : 3,
         ' bytes; ','  L1 : ', SIZEOF ( L1) : 3, ' bytes');
```



```
 WRITELN ( 'WORD   : ', SIZEOF ( WORD) : 3,
       ' bytes; ', '  W1 : ', SIZEOF ( W1) : 3, ' bytes');
 WRITELN ( 'CHAR   : ', SIZEOF ( CHAR) : 3,
       ' bytes; ', '  C1 : ', SIZEOF ( C1) : 3, ' bytes');
 WRITELN ( 'BOOLEAN : ', SIZEOF ( BOOLEAN) : 3,
       ' bytes; ', '  Bo1: ', SIZEOF ( Bo1) : 3, ' bytes');
 WRITELN ( 'STRING  : ', SIZEOF ( STRING) : 3,
       ' bytes; ', '  Stg: ', SIZEOF ( Stg) : 3, ' bytes');
 WRITELN ( 'REAL    : ', SIZEOF ( REAL) : 3,
       ' bytes; ', '  R1 : ', SIZEOF ( R1) : 3, ' bytes');
 READLN;
END.
```

إن البرنامج السابق يعطينا نفس النتائج فيمـــا إذا استدعينا التابع SIZEOF من أجل نوع معطيات أو متحول ينتمي لهذا النوع.

لنوع المعطيـــات ســـلسلة محرفيـــة String، 256 بايتاً في حين أن الطول الأعظمي للسلسلة المحرفيـــة لا يتجاوز 255 رمزاً. إن البرنامج السابق يعطينا الخرج التالي:

إن نتيجة تنفيذ البرنامج سوف تكون بالشكل التالي:

```
INTEGER :   2 bytes;    I1 :   2 bytes
BYTE    :   1 bytes;    By1:   1 bytes
SHORTINT:   1 bytes;    S1 :   1 bytes
LONGINT :   4 bytes;    L1 :   4 bytes
WORD    :   2 bytes;    W1 :   2 bytes
CHAR    :   1 bytes;    C1 :   1 bytes
BOOLEAN :   1 bytes;    Bo1:   1 bytes
STRING  :   256 bytes;  Stg:   256 bytes
REAL    :   6 bytes;    R1 :   6 bytes
```

وهو المطلوب.

```
    if N=0 then power:=1
   else
      if N<0 then
         power:=1/power(X,abs(N))
      else
         power:=X*power(X,N-1)
end;
Begin
   writeln('input X,N');
   readln(Y,M);
   power(Y,M);
   writeln(power(Y,M):8:2);
 readln
End.
```

٣- بأستخدام الخوارزمية التعاودية اكتب تابعا لحساب قيمة

$C(m,n)$

حيث

$0 \le m \le n$

ولدينا

$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$; $0 < m < n$

علما أن

$C_n^0 = C_n^n = 1$

إن التابع الذي يقوم بحساب هذه الخوارزمية له الشكل التالي:

```
function C(m,n:integer):integer;
begin
      If (m=0) or (m=n) then
```



```
         C:=1
   else
         C:=C(m,n-1)+C(m-1,n-1)
end;
```

٤- احسب قيمة التكامل:

$$\int_0^{\pi} \sin e^{10x}\, dx$$

إن قيمة التكامل تعطى بشكل تقريبي من أجل N=20 وفق الصيغه الدورية التالية:

$$\int_a^b f(x)\,dx \approx h\left[f(a)/2 + \sum_{i=1}^{n-1} f(a+ih) + f(b)/2\right]$$

حيث:

$$h = (b-a)/n$$

البرنامج بلغة تربو باسكال :

```
Program Integrals;
function f(X:real):real;
begin
   f:=sin(exp(10*X))
end;
function int(A,B:real;N:integer):real;
var
   H,S:real;
   I:integer;
begin
   H:=(B-A)/N;
   S:=(f(a)+f(b))/2;
   for I:=1 to N-1 do
        S:= S+f(A+I*H);
```

```
   int:=H*S
end;
Begin
  writeln(int(0,3.1415927,20):15:2);
readln
End.
```

إن نتيجة تنفيذ البرنامج تكون: −0.30

٥- لتكن لدينا C و D أعداد حقيقية حيث C<D والمطلوب حساب قيمة التكامل

$$\int_c^d \sin^2(15x)dx$$

إن قـيمة التكامل تعطى بشكل تقريبي من أجل N=100 وفق الصيغه الدورية التالية:

$$\int_a^b f(x)dx \approx h\left[f(a)/2 + \sum_{i=1}^{n-1} f(a+ih) + f(b)/2\right]$$

حيث:

$$h = (b-a)/n$$

البرنامج بلغة تربو باسكال :

```
Program Integrals;
Var
   C,D:real;
function f(X:real):real;
begin
   f:=sqr(sin(15*X))
end;
function int(A,B:real;N:integer):real;
var
```



```
   H,S:real; I:integer;
begin
   H:=(B-A)/N;
   S:=(f(a)+f(b))/2;
   for I:=1 to N-1 do
       S:=S+f(A+I*H);
   int:=H*S
end;
Begin
  read(C,D);
  writeln(int(C,D,100):15:2);
readln;readln
End.
```

إن نتيجة تنفيذ البرنامج تكون بالشكل التالي وذلك وفق القيم المعطاة:

من أجل	1 2	تكون النتيجة	0.49
ومن أجل	2 3	تكون النتيجة	0.48
ومن أجل	4 5	تكون النتيجة	0.52

٦- أوجد الأخطاء في البرنامج التالي:

```
Program Error;

const
       a='1234';
type
      string4=array[1..4] of integer;
Var
      b:string4;
```

```
Procedure p(s1:string4;var s:string4);
begin
     writeln(s1,s2)
end;
Begin
     b:=c;
     p(b,b);
     c:='3456';
     p(c,c);
readln
End.
```

الأخطاء هي:

التصريح عن نمط من نوع char .

يجب التصريح عن متحول c .

في تعريف الإجراء يجب التصريح عن s2 بدلا من s .

في بداية البرنامج يجب يجب كتابة b:=a; .

وهو المطلوب.

٧- نقـول عن شـــعاع إنه قابل للطي إذا كان نصفه الأول هو ترتيب عكســـي لنصفه الثاني.

والمطلوب كتابة برنامج يقوم بقراءة النصف الأول مـــن الشعاع،ومن ثم يطبع عناصر الشعاع.

البرنامج بلغة تربو باسكال :

```
Program VectorBuld;
uses crt;
label start;
```



```
const
    maxn=40;
type
    vector=array[1..maxn] of real;
Var
   i,n:integer;
   a:vector;
Begin
start:clrscr;
    writeln('inpyt n :n mod 2<>0');
    readln(n);
    if n mod 2 <>0 then
       begin
         writeln('Error');
         writeln('press enter to countinue');
         readln;
         goto start;
       end
    else
       for i:=1 to n div 2 do
        begin
         writeln('input a[',i,'] ');
         readln(a[i]);
        end;
    for i:=1 to n div 2 do
       a[n+1-i]:=a[i];

    for i:=1 to n  do
       write(a[i]:5:2);
readln
```

End.

إن نتيجة تنفيذ البرنامج تكون بالشكل التالي وذلك وفق القيم المعطاة:

```
inpyt n :n mod 2<>0
6
input  a[1]
1
input  a[2]
3
input  a[3]
55
      1.00     3.00     55.00     55.00     3.00     1.00
```

وهو المطلوب.

٨- لتكن لدينا الأعداد الحقيقية التالية:

$$a_0, a_1 \ldots\ldots, a_{30}, b_0, b_1 \ldots\ldots b_{30}, c_0, c_1 \ldots\ldots c_{30}$$

والمطلوب كتابة برنامج باستخدام البرامج الجزئية لحساب قيمة المقدار التالي:

$$\frac{(a_0x^{30} + a_1x^{29} + \ldots\ldots + a_{30})^2 - (b_0x^{30} + b_1y^{29} + \ldots\ldots + b_{30})}{c_0(x+z)^{30} + c_1(x+z)^{29} + \ldots\ldots + c_{30}}$$

البرنامج بلغة تربو باسكال:

```
Program Polynomial;
const N=30;
Type vector=array[0..N] of real;
Var
   A,B,C:vector;
   X,Y,Z,D:real;
procedure inp(var V:vector);
var
```



```
   I:integer;
begin
   writeln('input vector 30 values');
   for I:=0 to N do
        read(V[I]);
end;
function value(var V:vector;T:real):real;
var
   S:real;
   I:integer;
begin
   S:=V[0];
   for I:=1 to N do
   S:=S*T+V[I];
   value:=S
end;
Begin
  inp(A);inp(B);inp(C);
  read(X,Y,Z);
  D:=(sqr(value(A,X))-value(B,Y))/value(C,X+Z):
  writeln(D:15:2);
readln;
End.
```

٩- ليكن x عددا حقيقيا والمطلوب،اكتب برنامجا لحساب المقدار

$$sh(x)tg(x+1) - tg^2(2 + sh(x+1))$$

حيث أن: sh يحسب بالشكل:

$$sh := \frac{e^z - 1/e^z}{z}$$

البرنامج بلغة تربو باسكال:

```
    end;
```

٤- ليكن لدينا التابع المعرف في تربو باسكال بالشكل التالي:

```
function f(n:integer):integer;
    begin
        if n>100 then
            f:=n-10
        else
            f:=f(f(n+11)
    end;
```

يحســـب المقادير f(106) و f(99) و f(85) والمطلـــوب معرفة القيم التي يقوم بتعريفها هذا التابع.

٥- أوجد ناتج تنفيذ البرنامج التالي:

```
Program SideEffect;
Var
   A,B:integer;


function f(X:integer):integer;
begin
   f:=X;A:=0
end;
function g(var X:integer):integer;
begin
   g:=X;X:=0
end;
begin
```



```
   A:=1;write(A+f(A));
   A:=1;write(f(A)+A);
   B:=2;writeln(g(b)=g(b));
readln
End.
```

٦- أوجد ناتج تنفيذ البرنامج التالي:

```
Program Print;
Var
   a,b,c,d:integer;

Procedure p(var b:integer;c:integer);
var
   d:integer;
begin
   a:=9;b:=6;c:=7;d:=8;
   writeln(a,b,c,d)
end;


Begin
   a:=1;b:=2;c:=3;d:=4;
   p(a,b);
   writeln(a,b,c,d);
readln
End.
```

٧- أوجد ناتج تنفيذ البرنامج التالي:

```
Program Print;
type
   string5=array[1..5] of char;
Var
   i:integer;
   t:string5;
Procedure p(var s:string5);
begin
   i:=1;
   while s[i]<'9' do
       begin
         s[i]:=succ(s[i]);
         i:=i+1
       end;
end;
Begin
   i:=1;t:='12945';
   p(t);
   writeln(i,t[i]:5);
readln
End.
```

٨- اكتب برنامج لحساب قيمة التابع التالي:

$$f(n)=\begin{cases} n-10 & ; n>100 \\ 91 & ; n\le 100; \end{cases}$$

٩- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Print;
```



```
Var
   x,y:char;

Procedure p(x:integer);
const
    y=true;
begin
    writeln(x,' ',y)
end;

Procedure q;
var
    x:char;
begin
    x:=succ(y);
    y:='*';
    writeln(x,' ',y)
end;

Begin
    x:='a';
    y:='5';
    p(8);
    writeln(x,' ',y);
    q;
    writeln(x,' ',y);
readln
End.
```

١٠- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Error;
type
   string3=array[1..3] of char;
Var
   x:string3;

Procedure q(c:char;var d:char);
begin
    d:=succ(c)
end;

Begin
   x:='295';
   q(x[1],x[3]);
   writeln(x);
readln
End.
```

١١- اكتب ناتج تنفيذ البرنامج التالي:

```
Program Print;
type
   vect=array[1..2] of real;
Var
```



```
   a:vect;
   i:integer;
Procedure r(var k:integer;var x:real);
begin
   k:=2;
   x:=0
end;
Begin
   a[1]:=1;
   a[2]:=2;
   i:=1;
   r(i,a[i]);
   writeln(a[i]:8:2,a[2]:8:2);
readln
End.
```

١٢- ليكن لدينا

```
Const
        n=15;
type
        matrix=array[1..n,1..n] of real;
        vector=array[1..n] of real;

var
        a,b,c:matrix;
        x,y,z,u:vector;
```

والمطلوب حساب قيمة التابع u حيث u يعطى بالشكل التالي:

U=ax+by-cz+bx

١٣- أوجد الخطأ في كل من التوابع التالية:

```
function h(x:integer):integer;
begin
   h(x):=(sqr(x)+x)/2
end;

function g(k:integer):0..maxint;
var
   i,s:0..maxint;
begin
   s:=0;
   for i:=1 to k do
       s:=s+sqr(i)
end;

function f(a:'a'..'z'):integer;
begin
   f:=ord(a)-ord('p');
   if f<0 then f:=-f
end;
End.
```

١٤- ليكن لدينا البرنامج التالي:

```
Program Natural;
Type
   natur=1..maxint;
```



```
Var
   n,m:natur;
function hod(a,b:natur):natur;
begin
   while a<>b do
      if a>b then
         a:=a-b
      else
         b:=b-a;
   hod:=a;
end;
begin
  writeln('input n,m');
  readln(n,m);
  writeln(hod(n,m),n,m);
readln
End.
```

والمطلوب:

١- كتابة ناتج البرنامج عندما m=6 n = 8 .

٢-اشـــرح لماذا عند حســـاب hod(n,m) لا تتغير قيمة المتحولات m. n مع العلم أنه في جسم التابع تتغير قيم المتحولات.

الفصل الحادي عشر

نمط المعطيات:مجموعه SET

١١-١- مقدمه:

تعرف المجموعة رياضيا بأنها عبارة عن عدد مـن القيم أو العناصر غير المرتبة و غير المكررة ولا توجد بالضرورة علاقة ترتيب معرفة على عناصرها. تمثل المجموعة رياضيا بواسطة سرد عناصرها دون تكرار.

إن عمليات الانتمـاء والتقاطع والاجتماع والفرق معرفة على المجموعة Set ،وتسمح لغة البرمجة تربو باسكال بتعريف بنية معطيات: مجموعة مـن العناصر تنتمي إلى نمط متقطع، حيث يعرف النمط على أنه مجوعة Set of.

١١-٢- التصريح عن مجموعة Set:

إن التصريح عن متحول من نمط مجموعة يأخذ الشكل التالي:

< اسم المتحول > : set of < نمط متقطع > ;

إن النمط المتقطع < Discrete Type > هو أحد الأنماط التالية:

(Enumerated,Integer,Char,Boolean) .

أمثلة:

```
Q: set of 1..100 ;
Larg: set of 'a'..'z' ;
Ali: set of char ;
Digits: set of o..9 ;
```

أكثر المترجمات تكتفي كحد أعظم بـ 256 عنصرا في المجموعة وبرجع السبب في ذلك إلى تحقيق فعالية أكبر في طريقة التخزين وتنفيذ العمليات.

١١-٣- العمليات على المجموعات:

١١-٣-١- الاسناد والتعبير عن الثوابت:

على سبيل المثال إذا كان لدينا المتحول Q من نمط مجموعة، حيث:

```
Q: set of 1..200 ;
```

يمكن أن يأخذ Q الإسناد التالي :

```
Q:=[ 2 , 4 , 8..20 ,31 ] ;
```

إن الإسناد بهذا الشكل يجعل المجموعة Q حاوية للعناصر 2 و 4 و 31 و إضافة إلى العناصر الواقعة في المجال 8 و 20 .

تكتب المجموعة الخالية بالشكل التالي [ ] بفراغ أو بدون فراغ.

أمثلة:

-١

```
chs: set of char;
chs:=['a','b','c','ww'..'z'];
```

إن chs تحوي الأحرف c,b,a إضافة إلى الأحرف من w إلى z .

-٢

```
ln: set of 1..30 ;
ln:=[10];
```

هذه المجموعة تحوي عنصرا وحيدا هو الرقم 10.

-٣

```
i,j:integer;
ln:=[ i , j , i+j ];
```

هذه المجموعة تحوي ثلاثة عناصر قيمة i وقيمة j وقيمة i+j .

١١-٣-٢- العمليات الجبرية على المجموعات:

لتكن A وB مجموعتين عناصرهما من نمط واحد وليكن l متحولا من نفس النمط. يعبر عن عمليات المجموعات كما يلي:

اسم العملية	العلاقة الرياضية	العملية في تربو باسكال
التقاطع	$A \cap B$	$A * B$
الإجتماع	$A \cup B$	$A + B$
الفرق	A / B	$A - B$
احتواء أو مساواة	$A \supseteq B$	$A \geq B$
احتواء تام	$A \supset B$	$A > B$
مساواة	$A = B$	$A = B$
عدم المساواة	$A \neq B$	$A <> B$
الإنتماء	$l \in A$	$l \text{ in } A$

١١-٤- تطبيقات على المجموعات:

١١-٤-١- اختبار الانتماء لمجموعة:

يمكن استخدام النمط الثابت لمجموعة كاستخدامنا للشكل الثابت لأي

نمط، أي دون تصريح.

كمثال على اســتخدام المجموعات بشكلها الثابت، يمكن وضع خيارات المســتخدم في برنامج تخاطبي،بطريقة إدخال أحد الأحرف لاختيار العملية التي يريد تنفيذها.

البرنامج بلغة تربو باسكال:

```
PROGRAM SET_CHAR;
VAR
  CHOICE:CHAR;
BEGIN
 REPEAT
   WRITELN('(A)dd');
   WRITELN('(D)elete');
   WRITELN('(C)hange');
   WRITELN('(Q)uit');
   WRITELN('INPUT OPERATION,A,D,C,Q');
   READLN(CHOICE);
 UNTIL(CHOICE IN ['A','a','D','d','C','c','Q','q']);
END.
```

١١-٤-٢- اختبار تقاطع مجموعتين:

المطلوب كتابة برنامج بلغة تربو باســكال لمعرفة أرقام الطلاب الذين يتبعون دراسة مادتين محددتين.

هناك مجموعة من الطلاب تتبع مادة البرمجة PRO ومجموعة تتبع مـــــادة اللغة الإنكليزية ENG ، و توجد لدينا قوائم بأرقام الطلاب فــي كلا المادتين(إن الأرقام مخزنة في جدولين ARRAYS).

خوارزمية الحل:

نفرض أن أرقام الطلاب محصورة ضمن المجال 1..N حيث أن N لا تتجاوز 255. ليكن APRO و AENG الجدولين اللذين يحويان أرقام الطلاب للمادتيــن PRO و ENG على الترتيب و NPRO و NENG هو عدد الطلاب في كل منهما.

سنستخدم في الحل المجموعتين SPRO و SENG اللتين تحويان أرقام الطلاب للمادتين PRO وENG على الترتيب.

ستكون مراحل الحل كالتالي:

١- إسناد أرقام الطلاب من الجدول ARRAY إلى مجموعات SET .

٢- حساب التقاطع: S=SENG*SPRO .

٣- كتابة مجموعة التقاطع S .

نشرح الخطوة ٣ كتابة مجموعة التقاطع S كما يلي:

I:=1

مادام(S ليست خالية) و (I<=N) كرر ما يلي:

إذا كانت I تنتمي لـ S نفذ ما يلي:

١- اكتب I . ٢- اطرح I من S : S:=S-[I] .

I:=I+1

```
PROGRAM SET_ARRAY;
(* PERBORMS THE INTERSECTION OF TOW
                    SETS STORED IN ARRAYS *)
VAR
  APRO,AENG:ARRAY[1..150] OF INTEGER;
  SPRO,SENG,S:SET OF 1..150;
  NPRO,NENG,I:INTEGER;
BEGIN
  {THE NUMBER OF STUDENTS IN THE ARRAYS}
```

```
  MIX=SET OF COLOR;
VAR
  A,B,C:MIX;
  X:COLOR;
BEGIN
  A:=[RED,GREEN,BLUE];
  B:=[GREEN,BLUE,YELLOW];
  C:=A+B;
  FOR X:= RED TO VIOLET DO
    IF X IN C THEN
      CASE X OF
        RED:WRITELN('RED');
        GREEN:WRITELN('GREEN');
        BLUE:WRITELN('BLUE');
        YELLOW:WRITELN('YELLOW');
        CYAN:WRITELN('CYAN');
        VIOLET:WRITELN('VIOLET');
      END;
    READLN
END.
```

١١-٤-٣- الحصول على الأعداد الأولية:

يمكن استخراج الأعداد الأولية ضمن مجال معين دون استخدام عمليات الضرب أو القسمة وذلك بإتباع خوارزمية خاصة تسمى خوارزمية المنخل أو SIEVE .

فكرة الخوارزمية هي أن نضع الأعداد من 2 إلى نهاية المجال المطلوب في مجموعة سنسميها مجموعة المنخل SIEVE ،ثم نبدأ بانتزاع الأعداد الأولية منها ووضعها في مجموعة PRIMES ، كلما أخذنا عددا ننزع كل مضاعفاته

من المنخل،أما كيفية تحديد العدد الأولي فتجري بأخذ أول عنصر بين العناصر المتبقية في مجموعة المنخل.

مثال: الأعداد الأولية في المجال 2..10

المنخل	العنصر الأول	مضاعفاته	الأعداد الأولية
2,3,4,5,6,7,8,9,10	2	2,4,6,8,10	2
3,5,7,9	3	3,9	2,3
5,7	5	5	2,3,5
7	7	7	2,3,5,7

```
PROGRAM SIEVE_PRIMES;
(*GENERATED PRIMES USING SETS,WITHOUT
                MULTPLICATIONS OR DIVISION*)

CONST
   N=255;


VAR
  SIEVE,PRIMES:SET OF 1..N;
  NEXT,I:INTEGER;
BEGIN
  {INITIALISE}
  SIEVE:=[2..N];
  PRIMES:=[];
  NEXT:=2;
```

```
REPEAT  {FIND NEXT PRIME}
  WHILE (NOT(NEXT IN SIEVE)) DO
     NEXT:=SUCC(NEXT);
  PRIMES:=PRIMES+[NEXT];
  I:=NEXT;
  WHILE I<=N DO {ELIMINATE OF MULTIPLE
                                    OF"NEXT"}
     BEGIN
      SIEVE:=SIEVE-[I];
      I:=I+NEXT;
     END;
UNTIL SIEVE=[];
{WRITING PRIMES}
WRITELN('PRIMES BETWEEN 0..',N);
FOR I:=1 TO N DO
  BEGIN
   IF I IN PRIMES THEN
     WRITE(I:4)
   ELSE
     WRITE(' ---');
   IF I MOD 15=0 THEN
     WRITELN;
  END;
  READLN
END.
```

نتيجة تنفيذ البرنامج هي بالشكل التالي:

```
PRIMES BETWEEN 0..255

---  2  3 ---  5 ---  7 --- --- ---  11 ---  13 --- ---
```



```
---  17 ---  19 --- --- ---  23 --- --- --- --- ---  29 ---
 31 --- --- --- --- ---  37 --- --- ---  41 ---  43 --- ---
--- 47 --- --- --- --- ---  53 --- --- --- --- ---  59 ---
 61 --- --- --- --- ---  67 --- --- ---  71 ---  73 --- ---
--- --- ---  79 --- --- ---  83 --- --- --- --- ---  89 ---
--- --- --- --- --- ---  97 --- --- --- 101 --- 103 --- ---
--- 107 --- 109 --- --- --- 113 --- --- --- --- --- --- ---
--- --- --- --- --- --- 127 --- --- --- 131 --- --- --- ---
--- 137 --- 139 --- --- --- --- --- --- --- --- --- 149 ---
151 --- --- --- --- --- 157 --- --- --- --- --- 163 --- ---
--- 167 --- --- --- --- --- 173 --- --- --- --- --- 179 ---
181 --- --- --- --- --- --- --- --- --- 191 --- 193 --- ---
--- 197 --- 199 --- --- --- --- --- --- --- --- --- --- ---
211 --- --- --- --- --- --- --- --- --- --- --- 223 --- ---
--- 227 --- 229 --- --- --- 233 --- --- --- --- --- 239 ---
241 --- --- --- --- --- --- --- --- --- 251 --- --- --- ---
```

يكفي كتابة حلقة الطباعة في البرنامج بالشكل التالي:

```
FOR I:=1 TO N DO
  IF I IN PRIMES THEN
    WRITE(I:4);
```

ويكون ناتج التنفيذ بالشكل التالي:

```
PRIMES BETWEEN 0..255

  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47
53 59 61 67 71 73 79 83 89 97 101 103 107 109
```

113 127 131 137 139 149 151 157 163 167 173 179 181
191 193 197 199 211 223 227 229 233 239 241 251

وهو المطلوب.

أمثلة عامة:

مثال١:

اكتـــب برنامجا للحصول على مجموعة مربعات الأعداد الصحيحة التي لا تزيد على 100 ، ومجموعة أسس العدد 2 التي لا تزيد أيضا على العدد 100 ، ثـم اطبع المجموعة الحاصلة من تقاطعهما.(الحل يتم باستخدام بنية المعطيات مجموعة set) .

البرنامج بلغة تربو باسكال:

```
Program int_set;
Type
   intset=set of 0..100;
Var
   q,sqrset,exp2set:intset;
   X,I:integer;


Begin
   sqrset:=[];
   I:=1;
   X:=sqr(I);
   while X<=100 do
```



```
   begin
      sqrset:=sqrset+[X];
      I:=I+1;
      X:=sqr(I);
   end;
X:=1;
exp2set:=[];
while X<100 do
   begin
      exp2set:=exp2set+[X];
      X:=X*2
   end;
q:=sqrset*exp2set;
for I:=0 to 100 do
   if I in q then
     writeln(I);
readln
End.
```

ناتج التنفيذ هو 64 16 4 1 .

وهو المطلوب.

مثال٢:

اكتـــب برنامجا لإدخال نص لا يزيد عن 255 محرفا ، يقوم الـبرنامج بحســاب وطباعة عدد الحروف فـــي النص وعدد الأرقام فيه وعدد العلامـــات و الرموز الخاصـــة الأخـــرى ، و كذلك يقوم البرنامـــج بالحصول

على الحروف الصوتية والحروف الساكنة والأرقام المستخدمة و يطبعها.

البرنامج بلغة تربو باسكال:

```
Program TextDemo;

Type
   charset=set of char;

Var
   digits,letters,others:charset;
   vowels,consonants,allset:charset;
   di,let,ot,i:integer;
   tx:string;
   ch:char;

Begin
   writeln('Enter text:');
   readln(tx);
   di:=0;let:=0;ot:=0;
   digits:=['0'..'9'];
   letters:=['A'..'Z']+['a'..'z'];
   others:=[' '..'z']-(digits+letters);
   vowels:=['a','e','i','o','u','A','E','I','O','U'];
   consonants:=letters-vowels;
   allset:=[];

   for i:=1 to length(tx) do
   begin
    if tx[i] in letters then let:=let+1
    else
```



```
    if tx[i] in digits then di:=di+1
    else
    ot:=ot+1;
    allset:=allset+[tx[i]]
   end;

  vowels:=vowels*allset;
  consonants:=consonants*allset;
  digits:=digits*allset;
  writeln('the Text is:');writeln(tx);
  writeln('has:',let:3,' letters');
  writeln(' ',di:6,' digits');
  writeln('end',ot:3,' other symbols');writeln;
  writeln('The vowels in text is:');
  for ch:=' ' to 'z' do
    if ch in vowels then write(ch);writeln;
  writeln('The consonants in text is:');
  for ch:=' ' to 'z' do
    if ch in consonants then write(ch);writeln;
  writeln('The digits in text is:');
  for ch:=' ' to 'z' do
    if ch in digits then write(ch);writeln;
readln
End.
```

وهو المطلوب.

مثال٣:

اكتب برنامجا لإدخال نص لا يزيد عن 255 محرفا ، يقوم البرنامج بحساب وطباعة عدد الحروف في النص وعدد الأرقام فيه وعدد العلامات و

الـــرموز الخاصـــــة الأخـــــرى ، و عـــدد الحـــروف الصـــوتية وغـــير الصوتية(الســـاكنة)، باســـتخدام إجرائية خاصة اسمها Countchr و كذلك يقوم البرنامـــج بالحصول على الحـــروف الصوتيـــة و الحروف الســـاكنة و الأرقام المستخدمة باستخدام إجرائية اسمها Dispset مع طباعتها.

البرنامج بلغة تربو باسكال:

```
Program TextDemo;
type
   charset=set of char;
Var
   digits,letters,others:charset;
   vowels,consonants,allset:charset;
   di,let,ot,i,vo,co:integer;
   tx:string;
   ch:char;

procedure countchr(line:string;transset:charset;
                                        var count:integer);
var i:integer;
begin
    count:=0;
    for i:=1 to length(line) do
     if line[i] in transset then count:=count+1;
end;



procedure dispset(transset:charset);
var ch:char;
```



```
begin
    for ch:=' ' to 'z' do
     if ch in transset then write(ch);
     writeln;
end;

Begin
   writeln('Enter text:');
   readln(tx);

   digits:=['0'..'9'];
   letters:=['A'..'Z']+['a'..'z'];
   others:=[' '..'z']-(digits+letters);
   vowels:=['a','e','i','o','u','A','E','I','O','U'];
   consonants:=letters-vowels;

allset:=[];

for i:=1 to length(tx) do
     allset:=allset+[tx[i]];

vowels:=vowels*allset;
consonants:=consonants*allset;
digits:=digits*allset;



countchr(tx,digits,di);
```

```
countchr(tx,letters,let);
countchr(tx,others,ot);
countchr(tx,vowels,vo);
countchr(tx,consonants,co);

writeln('the Text is:');
writeln(tx);
writeln('has',let:3,' letters');
writeln(' ',di:5,' digits');
writeln(' ',ot:5,' other symbols');
writeln(' ',vo:5,' vowels');
writeln('and',co:3,' consonants');
writeln;
writeln('The vowels in text is:');
dispset(vowels);
writeln;
writeln('The consonants in text is:');
dispset(consonants);
writeln;
writeln('The digits in text is:');
dispset(digits);
readln
End.
```

وهو المطلوب.

مثال٤:

توجد في شركة N وظيفة شاغرة وتقدم لشـغل هذه الوظيفة M شخصا **حيث أن** M>N،وبفرض أن هؤلاء الأشخاص متساوين في المؤهلات والخبرات حيث M<=N . والمطلوب كتابة برنامـج لانتقاء N شخصا منهم على أن يتاح للجميع فرص متماثلة.

الحل:

لحل هذه المسـألة يرقم الأشـخاص بالأعداد من 1 وحتى M و ترقم الوظائف بالأعداد من 1 وحتى N .

ولـنفرض أن المجموعة work سـتمثل مجموعة أرقام الأشـخاص المختارين لشـغل الوظائف، من مجموعة الأشخاص الكلية person، وعلى أن تتم طريقة الاختيار بشكل عشوائي وتعطى الفرص للجميع.

إن المتحول X هو متحول عشـوائي حقيقي، يأخذ القيـم بيـن الصفر والواحد.

البرنامج بلغة تربو باسكال:

```
Program Job;
Type
   workset=set of 1..200;
Var
   S,work:workset;
   I,K,N,M:integer;
   X,L:real;
   T:boolean;
Begin
   writeln('Enter The number of Job:');
   readln(N);
   writeln('Enter The number of Person:');
```

تمارين عامة محلولة

تمرين١:

اكتب برنامجا لحساب قيمة التابع:

$$y=\begin{cases}|x| & ; x<0\\ e^{x} & ; 0\le x\le 1\\ \dfrac{1}{(2x+1)} & ; x>1\end{cases}$$

مستخدما تعريف Label ودون استخدام else .

البرنامج بلغة تربو باسكال:

```
Program Culculate_Function;
Label
   5,6,7;
Var
    i:integer;
  x,y:real;
Begin
   writeln('Input x Value');
   readln(x);
   if x<0 then goto 5;
 5:y:=abs(x);goto 7;
   if x<=1 then goto 6;
 6:y:=exp(x);goto 7;
   y:=1/(2*x+1);
 7:writeln('x=',x:8:1,' y=',y:8:1);
readln
End.
```



```
readln(M);
work:=[];
K:=1;
while K<=N do
 begin
  X:=random;
  L:=1/M;T:=true;I:=1;
  while T do
   begin
    if X<=I*L then
     begin
      S:=[I];T:=false
     end
    else
     I:=I+1;
   end;
  if S*work=[] then
     begin
      K:=K+1;
      work:=work+S
     end;
 end;
for I:=1 to M do
  if I in work then
    write(I,' ');
readln
End.
```

وهو المطلوب.

تمرين ٢ :

اكتب برنامجا لحساب معدل درجة الحرارة،ودرجة الحرارة العظمى ودرجة الحرارة الدنيا خلال 24 ساعة،حيث توجد قراءة لكل ساعة من ساعات اليوم،البرنامج يقوم بالحساب من أجل كل يوم من أيام السنة. إن البرنامج التالي يعطي قيمة اليوم ودرجة الحرارة العظمى ودرجة الحرارة الدنيا ومعدل درجة الحرارة في ذلك اليوم من أيام السنة.

البرنامج بلغة تربو باسكال:

```
Program Temperature_State;
{This program to calculates daily temperature,
                                    statistics for a year}
Var
  temp,  {an hourly temperature reading}
  high,  {highest temperature seen so far}
  low,   {lowest temperature seen so far}
  sum,   {running sum of temperatures}
  hour,  {loop control variable for hours in a day}
  day,   {loop control variable for days in year}
  average:integer;{average temperature}
Begin   {temp}
  {initialize loop control variable}
  day:=1;
   while day <= 365 do
    begin
     {initialize high to impossibly low value}
     high:=-maxint;
     {initialize low to impossibly high value}
```



```
     low:=maxint;
     {initialize loop control variable}
     hour:=1;
     {initialize running sum}
     sum:=0;
     writeln('Input Temperatures');
     while hour <= 24 do
     begin
        {get temp}
        readln(temp);
        {add temp to sum}
        sum:=sum+temp;
        {check for new low}
        if temp < low then
          low:=temp;
        {check for new high}
        if temp > high then
          high:= temp;
        {increment loop control variable}
        hour:=hour+1;
      end;
     average:=sum div 24;
     {print}
     writeln('Temperature data for 2003');
     writeln;writeln;
     writeln('day    high    low    average':40); writeln;
     writeln(day:10,high:10,low:10,average:10);
     day:=day+1;
    end;readln
End.
```

تمرين٣:

اكتـب مقطع من برنامج يقرأ عددا ويحصي عدد الأعداد المدخلة مادام العدد المدخل أقل من 15 وغير سالب .

الحل:

```
NEG:=false;
Count:=1;
While(Count<=15) AND NOT NEG do
    Begin
        read(data):
        If data < 0 then
           NEG:  true
        else
           Count:=Count+1;
    End;
```

تمرين٤:

اكتب برنامجا لإيجاد العوامل الأولية لعدد ما معطى.

البرنامج بلغة تربو باسكال :

```
Program  Prime_parameters:
Var
   i,n:integer;
begin
    writeln('input n');
    readln(n);
    i:=2;
    while n<>1 do
       begin
          if n mod i=0 then
            begin
```



```
              writeln(i);
              n:=n div i;
            end
          else
            i:=i+1;
       end;
readln
End.
```

تمرين٥:

اكتب برنامجا لإيجاد قواسم عدد ما معطى.

البرنامج بلغة تربو باسكال :

```
PROGRAM DIVESOR;
VAR
    COUNT,NUMBER:INTEGER;
BEGIN
    COUNT:=1;
    WRITE('INPUT NUMBER=');
    READLN(NUMBER);
    WHILE COUNT<=NUMBER DO
     BEGIN
       IF NUMBER MOD COUNT=0 THEN
          WRITELN('COUNT=',COUNT);
       COUNT:=COUNT+1;
     END;
READLN
END.
```

يمكن تعميم البرنامج بحيث يسأل البرنامج عن امكانية تكرار العمل من أجل حساب قواسم عدد آخر أو عدم التكرار وإنهاء البرنامج.

البرنامج يصبح بالشكل التالي:

```
PROGRAM DIVESOR;
LABEL 2;
VAR
  COUNT,NUMBER:INTEGER;
  W:CHAR;
BEGIN
 2: COUNT:=1;
   WRITE('INPUT NUMBER=');READLN(NUMBER);
   WHILE COUNT<=NUMBER DO
   BEGIN
    IF NUMBER MOD COUNT=0 THEN
      WRITELN('COUNT=',COUNT);
    COUNT:=COUNT+1;
   END;
   WRITELN('DO YOU WHANT TO FIND DIVESOR
                                    ANOTHER NUMBER');
   WRITELN('PRECE (Y/N)');
   READLN(W);
   IF W='Y' THEN GOTO 2
   ELSE
END.
```

تمرين ٦:

اكتب ناتج تنفيذ البرنامج التالي:

```
PROGRAM TRUNC_DIV_MOD;
BEGIN
```



```
 WRITELN(TRUNC(SIN(1)):4);
 WRITELN(3 DIV 2*4 :4);
 WRITELN(3*7 DIV 2 MOD 7/3 -TRUNC(SIN(1)):4);
 READLN
END.
```

تمرين٧:

اكتب برنامجا لحساب عاملي عدد باستخدام التعليمة التكرارية :WHILE- DO

البرنامج بلغة تربو باسكال :

```
PROGRAM FACTORIAL;
VAR
    I,N:INTEGER;
      F:REAL;
BEGIN
    WRITELN('INPUT N');
    READLN(N);
    IF N=0 THEN
     BEGIN
       F:=1;
       WRITELN('THE FACTORIAL OF',N,'IS=',F);
     END
     ELSE
       BEGIN
         F:=1;I:=1;
         WHILE I<=N DO
          BEGIN
            F:=F*I;I:=I+1;
          END;
         WRITELN('THE FACTORIAL OF',N:2,
```

```
                                    ' IS=',F:8:3);
        END;READLN
END.
```

تمرين٨:

اكتب برنامجا لحساب عدد مرفوع إلى قوة .يطلب البرنامج إدخال العدد ليحسب قوته، ولا يتوقف البرنامج إلى إذا كان العدد المدخل يساوي الصفر.

```
PROGRAM EXPONENT;
VAR
    X,Y,Z:REAL;
BEGIN
   WRITELN('THIS PROGRAM TO GET
                                    THE RESULT OF X^Y');
   WRITELN('INPUT X'); READLN(X);
   WHILE X>0 DO
     BEGIN
       WRITELN('INPUT Y');READLN(Y);
       Z:=EXP(Y*LN(X));
       WRITELN('THE RESULT OF X^Y IS =',Z:8:2);
       READLN(X);
     END;
END.
```

تمرين٩:

اكتب برنامجا لإيجاد الأعداد الأولية بين العدد 2 العدد 100 .

توجد أكثر من طريقة نذكر هنا طريقتين:

-الطريقة الأولى :

```
Program PrimeNumbers;
Var
  I,S,N:integer;
```



```
begin
   N:=1;
   while N<=100 do
    begin
      I:=1;S:=0;
      while I<=N do
        begin
         if N mod I=0 then
           begin
            S:=S+I;
            I:=I+1;
           end;
         else
           I:=I+1;
        end;
       if N=S-1 then
        writeln(N:3);
       N:=N+1;
    end;
readln
End.
```

-الطريقة الثانية:

```
Program PrimeNumbers;
Var
  I,S,N:integer;
begin
   for N:=2 to 100 do
    begin
     S:=0;
     for I:=2 to N do
```

```
      if N mod I=0 then
        S:=S+I;
    if N=S then
      writeln(N:3);
  end;
readln
End.
```

نتيجة تنفيذ البرنامج تكون بالشـــكل التالي:

```
 2
 3
 5
 7
 11
 13
 17
 19
 23
 29
 31
 37
 41
 43
 47
 53
 59
 61
 67
 71
```



```
73
79
83
89
97
```

تمرين ١٠:

اكتب برنامجا لإيجاد العوامل الأولية لعدد ما معطى.

البرنامج بلغة تربو باسكال :

```
Program Prime_parameters;
Var
  i,n:integer;
begin
  writeln('input n');
  readln(n);
  i:=2;
  while n<>1 do
    begin
      if n mod i=0 then
        begin
          writeln(i);
          n:=n div i;
        end
      else
        i:=i+1;
    end;
readln
End.
```

تمرين ١١:

اكتب برنامجا لترتيب عناصر مصفوفه وفق عناصر أسطرها بشكل تصاعدي وذلك وفق العناصر الأولى ، يطلب طباعة المصفوفة قبل الترتيب وبعد الترتيب.

```
PROGRAM SORTING_MATRIX;
TYPE
      VECTOR=ARRAY[1..30]OF REAL;
      MATRIX=ARRAY[1..30] OF VECTOR;
VAR
     A:MATRIX;X:VECTOR;
     I,J,K,M,N:INTEGER;
BEGIN
     WRITELN('INPUT N:N<=30');READLN(N);
     WRITELN('INPUT M:M<=30');READLN(M);
     WRITELN('INPUT THE MATRIX:');
FOR I:=1 TO N DO BEGIN
  FOR J:=1 TO M DO BEGIN
    WRITE('A[',I,',',J,']=');READLN(A[I,J]);
  END;
END;
WRITELN('MATRIX BEFOR
          SORTING');WRITELN;
FOR I:=1 TO N DO
 BEGIN
  FOR J:=1 TO M DO
    WRITE(A[I,J]:8:1);
  WRITELN
 END;
FOR K:=N DOWNTO 2 DO
```



```
     BEGIN
       J:=1;
       FOR I:=2 TO K DO
         IF A[I,1]>A[J,1] THEN
           J:=I;
       X:=A[K];A[K]:=A[J];A[J]:=X;
     END;
     WRITELN;
     WRITELN('MATRIX AFTER
                SORTING');WRITELN;
     FOR I:=1 TO N DO
      BEGIN
       FOR J:=1 TO M DO
         WRITE(A[I,J]:8:1);
       WRITELN
      END;
READLN
END.
```

إن نتيجة تنفيذ البرنامج وفق المعطيات المدخلة هي بالشكل التالي:

```
INPUT N:N<=30
4
INPUT M:M<=30
5
INPUT THE MATRIX:
A[1,1]=1
A[1,2]=2
A[1,3]=1
A[1,4]=3
A[1,5]=1
A[2,1]=7
```

A[2,2]=8
A[2,3]=9
A[2,4]=4
A[2,5]=3
A[3,1]=3
A[3,2]=2
A[3,3]=4
A[3,4]=1
A[3,5]=2
A[4,1]=1
A[4,2]=2
A[4,3]=1
A[4,4]=2
A[4,5]=1
MATRIX BEFOR SORTING

1.0	2.0	1.0	3.0	1.0
7.0	8.0	9.0	4.0	3.0
3.0	2.0	4.0	1.0	2.0
1.0	2.0	1.0	2.0	1.0

MATRIX AFTER SORTING

1.0	2.0	1.0	2.0	1.0
1.0	2.0	1.0	3.0	1.0
3.0	2.0	4.0	1.0	2.0
7.0	8.0	9.0	4.0	3.0

تمرين١٢:

اكتـــب برنامجا لقراءة مصفوفة وطباعتها ثم تبديل عناصر الأســـطر الفردية بالقيمة 8 وطباعة المصفوفة الناتجة.

البرنامج بلغة باسكال:

```
Program ChangeMatrix;
Type
    matrix=array[1..20,1..20] of real;
Var
    a:matrix;
    i,j,n,m:integer;
Begin
    writeln('input n,m:n<20,m<20');
    readln(n,m);
    for i:=1 to n do
     for j:=1 to m do
      begin
       write('input a[',i,',',j,']=');
       readln(a[i,j]);
      end;
writeln;writeln('Matrix before Change');writeln;
for i:=1 to n do
   begin
      for j:=1 to m do
        write(a[i,j]:5:1);
      writeln;
   end;
writeln;writeln('Matrix after Change');writeln;
for i:=0 to n do
 for j:=1 to m do
  a[2*i+1,j]:=8;
for i:=1 to n do
   begin
      for j:=1 to m do
```

```
            write(a[i,j]:5:1);
          writeln;
        end;
readln
End.
```

إن نتيجة تنفيذ البرنامج وذلك حسب القيم المعطاة تكون بالشكل التالي:

```
input n,m:n<20,m<20
5 6
input a[1,1]=11
input a[1,2]=12
input a[1,3]=13
input a[1,4]=14
input a[1,5]=15
input a[1,6]=2
input a[2,1]=1
input a[2,2]=3
input a[2,3]=4
input a[2,4]=5
input a[2,5]=22
input a[2,6]=33
input a[3,1]=44
input a[3,2]=55
input a[3,3]=66
input a[3,4]=4
input a[3,5]=5
input a[3,6]=5
input a[4,1]=5
input a[4,2]=5
input a[4,3]=66
input a[4,4]=55
```



```
input a[4,5]=4
input a[4,6]=4
input a[5,1]=3
input a[5,2]=3
input a[5,3]=11
input a[5,4]=11
input a[5,5]=67
input a[5,6]=43

Matrix before Change

 11.0 12.0 13.0 14.0 15.0  2.0
  1.0  3.0  4.0  5.0 22.0 33.0
 44.0 55.0 66.0  4.0  5.0  5.0
  5.0  5.0 66.0 55.0  4.0  4.0
  3.0  3.0 11.0 11.0 67.0 43.0

Matrix after Change

 8.0  8.0  8.0  8.0  8.0  8.0
 1.0  3.0  4.0  5.0 22.0 33.0
 8.0  8.0  8.0  8.0  8.0  8.0
 5.0  5.0 66.0 55.0  4.0  4.0
 8.0  8.0  8.0  8.0  8.0  8.0
```

وهو المطلوب.

تمرين١٣:

يعرف العدد الكامل بأنه العدد الذي تساوي قيمته مجموع قيم قواسمه ما عدا العدد نفسه.والمطلوب كتابة برنامج بلغة تربو باسكال لإيجاد الأعداد الكامله بين العدد 2 والعدد 500 .

يجب التنويه هنا إلى أنه توجد أكثر من طريقة لحل هذا التمرين سوف نكتب بعض هذه الطرق.

- الطريقة الأولى:

```
Program CompNumb;
Uses crt;
Var X:array[1..500] of longint;
   I,J,L:integer;
Begin
 clrscr;
 for I:=2 to 500 do
  X[I]:=0;
 for I:=2 to 500 do
  for J:=1 to I-1 do
   begin
    L:=I mod J;
    if L=0 then X[I]:=x[I]+j;
   end;
 for I:=2 to 500 do
  if I=X[I] then
    writeln(I,' ','is a completed number');
 readln
End.
```

-نتيجة تنفيذ البرنامج تكون بالشكل التالي:


```
6 is a completed number
28 is a completed number
496 is n completed number
```

- الطريقة الثانية:

```
Program Complet_Number;
Var
 i,j,s,n:integer;
Begin
 n:=1;
 while n<=500 do
  begin
   i:=1;s:=0;
   while i<=n do
    begin
     if n mod i=0 then
      begin
       s:=s+i;
       i:=i+1;
      end
     else
       i:=i+1;
    end;
   if n=s-n then
     writeln('n=',n:8,' it"s a complete number');
   n:=n+1;
  end;
readln
End.
```

-نتيجة تنفيذ البرنامج تكون بالشكل التالي:

```
n=     6 it"s a complete number
n=    28 it"s a complete number
n=   496 it"s a complete number
```

- الطريقة الثالثة:

```
Program CompletNum;
Var i,j,s:integer;
Begin
  for i:=2 to 500 do
  begin
    s:=0;
    for j:=1 to i-1 do
      if (i mod j=0) then
        s:=s+j;
    if s=i then
      writeln(i,' is a complete number');
  end;
readln
End.
```

- الطريقة الرابعة:

يتم استخدام تعريف التابع في حل التمرين.

البرنامج بلغة تربو باسكال يكون بالشكل التالي:

```
Program Numbers;
Var
   i:integer;
```



```
Function sundiv(n:integer):integer;
var s,m:integer;
begin
 s:=0;
 for m:=1 to n-1 do
  if n mod m=0 then
    s:=s+m;
 sundiv:=s;
end;
Begin
  for i:=1 to 500 do
   if sundiv(i)=i then
    writeln(i:8);
readln
End.
```

مما تقدم نلاحظ وجود أربع طرق لحل هذا التمرين ، رغبنا بكتابة هذه الطـــرق لكي يطلع الطالب على كل الإمكانات المتوفرة في لغة التربو باسكال ومن أجل مقارنة الطرق مع بعضها البعض وهو المطلوب.

تمرين ١٤:

يقال عن عددين n و m بأنهما عددان صديقان إذا كان مجموع جميع قواسم العدد m يساوي n ومجوع جميع قواسم العدد n يساوي العدد m . والمطلوب:

اكتـــب بـــرنامجا بلغة تربو باسكال لإيجاد الأعداد الصديقة بين العدد 1 والعدد 3000 .

نـــنوه هنا إلى وجود أكثر من طريقة لكتابة البرنامج سوف نذكر بعضا من هذه الطرق.

-الطريقة الأولى:

```
Program Friends;
Uses crt;
Var
   X:array[1..3000] of longint;
   N,M,L:integer;
   A,B:boolean;
Begin
 clrscr;
 for N:=1 to 3000 do
  X[N]:=0;
 X[1]:=1;
 for N:=2 to 3000 do
  for M:=1 to N-1 do
   begin
    L:=N mod M;
    if L=0 then
      X[N]:=x[N]+M;
   end;
 for N:=1 to 3000-1 do
  for M:=N+1 to 3000 do
   begin
    if N=X[M] then A:=true else A:=false;
```



```
    if M=X[N] then B:=true else B:=false;
    if (A and B) then
      writeln(N,',',M,' Friends');
   end;
 readln
End.
```

يمـــكن تعديل البرنامـــج السابق وذلك بتعديل بعض الأوامر فيه ، كما في الشكل التالي:

```
Program Friends;
Uses crt;
Var X:array[1..3000] of longint;
   N,M,L:integer;
   A,B:boolean;
Begin
 clrscr;
 for N:=1 to 3000 do
  X[N]:=0;
 X[1]:=1;
 for N:=2 to 3000 do
  for M:=1 to N-1 do
   begin
    L:=N mod M;
    if L=0 then
      X[N]:=x[N]+M;
   end;
 for N:=1 to 3000-1 do
  for M:=N+1 to 3000 do
```

```
  begin
   if N=X[M] then A:=true else A:=false;
   if M=X[N] then B:=true else B:=false;
   if (A and B) then
     writeln(N,',',M,' Friends');
  end;
 readln
End.
```

-نتيجة تنفيذ البرنامج تكون بالشكل التالي:

```
220 , 284 friends
1210 , 1184 friends
2924 , 2620 friends
```

-الطريقة الثانية:

```
Program Friends;
Var k,i,j,f,s:integer;
Begin
 for i:=2 to 3000 do
 begin
   f:=0;s:=0;
   for j:=1 to i-1 do
     if (i mod j=0) then
       f:=f+j;
   for k:=1 to f-1 do
     if (f mod k=0) then
       s:=s+k;
   if (s=i) and (f<3000) and (f<>i) then
     writeln(f,' and ',i,' are friends');
```



```
  end;
readln
End.
```

وهو المطلوب.

تمرين ١٥:

ليكن لدينا التابع

$$Y = e = 1 + \frac{X}{1!} + \frac{X^2}{2!} + \frac{X^3}{3!} + \ldots\ldots\ldots + \frac{X^N}{N!}$$

والمطلوب كتابة برنامج لحساب قيمة التابع باستخدام تعريف التابع وذلك من أجل:

١- X=1 , N=3
٢- X=2 , N=3
٣- X=3 , N=3

البرنامج بلغة تربو باسكال:

```
PROGRAM EXP;
VAR
  X,Y,P:REAL;
  N:INTEGER;
FUNCTION FUNC(X:REAL;N:INTEGER):REAL;
VAR
  A,Y:REAL;
  I:INTEGER;
BEGIN
```

```
    Y:=1;A:=1;
    FOR I:=1 TO N DO
     BEGIN
      A:=A*X/I;
      Y:=Y+A;
     END;
    FUNC:=Y
END;
BEGIN
 WRITELN('THE FUNCTN FUNC(1,3) IS='
                                    ,FUNC(1,3):8:3);
 WRITELN;
 WRITELN('THE FUNCTN FUNC(2,3) IS='
                                    ,FUNC(2,3):8:3);
 WRITELN;
 WRITELN('THE FUNCTN FUNC(2,3) IS='
                                    ,FUNC(3,3):8:3);
 WRITELN;
 P:=FUNC(1,3)+FUNC(2,3)+FUNC(3,3);
 WRITELN('P = FUNC(1,3)+FUNC(2,3)+FUNC(3,3) =
                                    ',P:8:3);
READLN
END.
```

ناتج تنفيذ البرنامج هو بالشكل التالي:

```
THE FUNCTN FUNC(1,3) IS=  2.667

THE FUNCTN FUNC(2,3) IS=  6.333
```



```
THE FUNCTN FUNC(2,3) IS=  13.000

P = FUNC(1,3)+FUNC(2,3)+FUNC(3,3) =   22.000
```

تمرين ١٦:

اكتب برنامجا لحساب حجم صندوق،وطباعة خط بطول 45 ،من الشكل التالي: **.....******* ولا ينتهي البرنامج إلا إذا كانت أحد المدخلات تساوي الصفر. مستخدما في ذلك تعريف البرامج الجزئية.

البرنامج بلغة تربو باسكال:

```
PROGRAM PRINTVOL_;
VAR
     L,W,H:REAL;
PROCEDURE
PRINTVOL(LENGTH,WIDTH,HEIGHT:REAL);
VAR
    V:REAL;
BEGIN
   V:=LENGTH*WIDTH*HEIGHT;
   WRITELN('THE VALUE IS = ',V:8:3);
END;
PROCEDURE PRINTLINE2;
CONST
     LENGTH=45;
```

```
VAR
      I:INTEGER;
BEGIN
   FOR I:=1 TO LENGTH DO
      WRITE('*');
   WRITELN
END;

BEGIN
 REPEAT
  WRITELN('INPUT LENGTH AND WIDTH AND HEIGHT
                                          OF BOX');
  READLN(L,W,H);
  PRINTVOL(L,W,H);
  PRINTLINE2;
 UNTIL (L=0) OR (W=0) OR (H=0);
READLN
END.
```

تمرين ١٧:

اكتب برنامج بلغة تربو باسكال ينجز ما يلي:

١- ينشئ تابعا لإيجاد أكبر عددين مدخلين .

٢- ينشئ تابعا لإيجاد عاملي عدد .

٣- يكتب إجرائية لرسم خط بطول 35 من الشكل:---------- .

٤- يحسب قيمة السلسلة $X = 1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+.....+\frac{1}{10!}$.

٥- يطبع قيمة السلسلة .

٦- يطبع الخط .

٧- يأخذ قيمة أكبر عددين مدخلين ويطبعها .

٨- يربع الناتج في الطلب السابق ويطبعه .

وهو المطلوب .

البرنامج بلغة تربو باسكال:

```
PROGRAM FACTORIAL_AND_MAX;
VAR
  X,Y,Q,Z,W,A,B:REAL;
  J:INTEGER;
FUNCTION MAX(X,Y:REAL):REAL;
BEGIN
  IF X>Y THEN MAX:=X ELSE MAX:=Y
END;
FUNCTION FACTORIAL(N:INTEGER):REAL;
VAR
  I,F:INTEGER;
BEGIN
   F:=1;
   FOR I:=1 TO N DO
      F:=F*I;
   FACTORIAL:=F
END;
PROCEDURE PRINTLINE;
```

```
CONST
   LENGTH=35;
VAR
  I:INTEGER;
BEGIN
   FOR I:=1 TO LENGTH DO
     WRITE('-');
   WRITELN
END;
BEGIN
   X:=1;
   FOR J:=1 TO 10 DO
    BEGIN
     Y:=1/FACTORIAL(J);
     X:=X+Y;
    END;
   WRITE('THE VALUE OF X=');
   WRITELN(X:8:2);
   PRINTLINE;
   WRITELN('INPUT TOW NUMBERS');
   READLN(A,B);
   Z:=MAX(A,B);
   WRITELN('THE MAX OF',A:5:2,' AND ',B:5:2,'
                                    IS = ',Z:8:2);
   Q:=Z*Z;
   WRITELN('THE SQR OF Z  IS = ',Q:8:2);
READLN
END.
```

تمرين ١٨:

اكتب برنامجا بلغة تربو باسكال يقوم بمايلي :

١- قراءة المحارف الممثلة لنص لاتيني علما بأن النص ينتهي بحرف(.) .

٢- استعراض النص واستبدال كل حرف صغير في بداية أية كلمة بالحرف الكبير(فقط في بداية الكلمات جميعها) واستبدال كل علامة استفهام موجودة في النص بعلامة تعجب .

٣- حساب طول النص ما عدا(.) .

-البرنامج بلغة تربو باسكال:

```
Program ReadCode;
Var
 a:array[0..300] of char;
 i,j,x:integer;
Begin
   i:=1;a[0]:= ' ';
   while a[i]<>'.' do
     begin
      i:=i+1;
      read(a[i]);
     end;
   for j:=1 to i do
    if a[j-1]=' ' then
     begin
      x:=ord(a[j]);
      if (x>96) and (x<123) then
       begin
```

```
          x:=x-32;a[j]:=chr(x);
        end;
      end
    else
      if (a[j]='?') then
        a[j]:='!';
  for j:=1 to i do
    write(a[j]);
  writeln;
  writeln('The Length of The Text is=',i-1);
readln;readln
End.
```

تمرين ١٩:

اكتب برنامجا بلغة تربو باسكال يقوم بمايلي :

١- قراءة المحارف الممثلة لنص لاتيني علما بأن النص ينتهي بحرف($) .

٢- اظهار عدد كل من الأحرف الكبيره والصغيره وعلامات التنقيط الموجودة في النص بالترتيب وطول النص .

-البرنامج بلغة تربو باسكال:

```
Program ReadCode;
Var
 w:char; l,s,t,n:integer;
Begin
 writeln('When You Want to End The Text Press $');
 l:=0;s:=0;t:=0;n:=0;
 read(w);
 while w<>'$' do
  begin
```



```
   if (ord(w)>=65) and (ord(w)<=90) then  l:=l+1;
   if (ord(w)>=97) and (ord(w)<=122) then s:=s+1;
   if (ord(w)=44) or (ord(w)=46) or (ord(w)=58)
         or (ord(w)=59) then  t:=t+1;
   n:=n+1;
   read(W);
  end;
 writeln('Large Letters  ',l);
 writeln('Small Letters=',s);
 writeln('Points=',t);
 writeln('All Letters=',n);
readln;readln
End.
```

تمرين ٢٠:

اكتب برنامجا بلغة تربو باسكال يقوم بمايلي :

١- يقرأ عناصر المصفوفه A ذات السعه (n,2n) أي A=[n,2n] .

٢- يكتب هذه المصفوفه على شكل مصفوفتين ، أي يجزئ المصفوفه A إلى مصفوفتين بالشكل: A=[B | C] ، حيث B=[n,n] و C=[n,n] .

٣- يجمع عناصر المصفوفه B مع بعضها البعض ويجمع عناصر المصفوفه C مع بعضها البعض .

٤- يحسب القيمة المطلقة للفرق بين القيمتين السابقتين ويطبعه.

-البرنامج بلغة تربو باسكال:

```
PROGRAM ABSLUT_SUM_NUMBERS;

TYPE
  MATRIX=ARRAY[1..5,1..10] OF REAL;

VAR
  N,I,J  :INTEGER;
  S1,S2,Q:REAL;
  A,B,C  :MATRIX;

BEGIN
   WRITELN('INPUT N ');
   READLN(N);
   WRITELN('INPUT MATRIX ELEMENTS');
   FOR I:=1 TO N DO
    FOR J:=1 TO 2*N DO
     begin
       write('A[',I,',',J,']=');
       READ(A[I,J]);
     end;
  FOR I:=1 TO N DO
   FOR J:=1 TO N DO
   BEGIN
      B[I,J]:=A[I,J];
      C[I,J]:=A[I,J+N];
   END;
```



```
    S1:=0;S2:=0;
    FOR I:=1 TO N DO
     FOR J:=1 TO N DO
      BEGIN
        S1:=S1+B[I,J];
        S2:=S2+C[I,J];
      END;
    Q:=ABS(S2-S1);
    WRITELN('ABS(S2-S1) =',Q:8:2);
READLN;
READLN
END.
```

تمرين ٢١:

اكتـــب بـــرنامج بلغة تربو باسكال يقرأ مجموعة من الأعداد ، ثم يطبع الأعـــداد التـــي هـــي أكبر من 10 ، والأعداد التي هي أقل من 10 ، والأعداد الزوجية ، ويسأل البرنامج فيما إذا أردنا إدخال مجموعة اخرى من الأعداد أم لا اي يكرر عملية الإدخال حتى نطلب منه عدم الإدخال .

-البرنامج بلغة تربو باسكال:

```
PROGRAM NUMBER;
LABEL 2;
VAR
  N,I:INTEGER;
  X:ARRAY[1..100] OF INTEGER;
```

```
  W:CHAR;
BEGIN
 2: WRITE('HOU MANY NOMBERS DO YOU WHANT
                                    TO READ=');
 READLN(N);
 FOR I:=1 TO N DO
  BEGIN
     WRITE('X[',I,']=');READLN(X[I]);
  END;
 WRITE('NUMBERS>10= ');
 FOR I:=1 TO N DO
   IF X[I]>10 THEN
     WRITE(X[I],' ');
 WRITELN;WRITELN('----------------');
 WRITE('NUMBERS<0= ');
  FOR I:=1 TO N DO
    IF X[I]<0 THEN
      WRITE(X[I],' ');
 WRITELN;WRITELN('----------------');
 WRITE('EVEN NUMBERS= ');
 FOR I:=1 TO N DO
  IF X[I] MOD 2=0 THEN
    WRITE(X[I],' ');
 WRITELN;
 WRITE('DO YOU WHANT TO INPUT ANOTHER
                                NUMBERS (Y/N):');
 READLN(W);
 IF W='Y' THEN GOTO 2
 ELSE
readln
```



```
END.
```

تمرين ٢٢:

اكتـب بـرنامجا بلغة تربو باسكال لإيجاد أكبر عدد وأصغر عدد من مجموعة أعداد معطاة باستخدام الإجرائيات .

-البرنامج بلغة تربو باسكال:

```
PROGRAM MAXIM_MINM;

CONST
   M=200;

TYPE
   VECTOR=ARRAY[1..M] OF REAL;

VAR
   X:VECTOR;
   MAX,MIN:REAL;
   LONG,Q:INTEGER;

PROCEDURE INPUTVEC(VAR A:VECTOR;
                                  L:INTEGER);
VAR
  I:INTEGER;
BEGIN
 WRITELN('INPUT THE VALUES');
 FOR I:=1 TO L DO
  BEGIN
```

```
    WRITE('A[',I,']=');READLN(A[I]);
  END;
END;

PROCEDURE MAXMIN(A:VECTOR;N:INTEGER;
                          VAR LARG,SMAL:REAL);
VAR
  I:INTEGER;
BEGIN
   IF A[1]<A[2] THEN
    BEGIN
     SMAL:=A[1];LARG:=A[2];
    END
   ELSE
    BEGIN
     SMAL:=A[2];LARG:=A[1];
    END;
   FOR I:=3 TO N DO
    IF A[I]<SMAL THEN
       SMAL:  A[I]
    ELSE
    IF A[I]>LARG THEN
       LARG:=A[I];
END;

BEGIN
  REPEAT
    Q:=1;
    WRITE('INPUT LONG=');READLN(LONG);
    INPUTVEC(X,LONG);
```



```
    MAXMIN(X,LONG,MAX,MIN);
    WRITELN('MAX=',MAX:8:2,'  MIN=',MIN:8:2);
    WRITE('INPUT Q=');READLN(Q);
  UNTIL Q=0;
END.
```

إن الـــبرنامج السابق يستخدم إجرائية لإدخال قيم الأعداد حيث يتعامل معهـــا كصف من الأعداد معرفا إياها كشعاع . إن المتحول المصفوفي A هو مـــتحول دخل خرج حيث تؤخذ قيمـــه في عملية حســـاب الأصغر والأكبر في الإجرائـــية الثانية ، أن العدد الأصغر و العدد الأكبر هي متحولات دخل خرج أيضا ، وذلك من أجل طباعتها على الشاشة . لا ينهي البرنامج عمله إلا عندما يتم إدخال القيمة 0 ،عندها ينتهي البرنامج .

تمرين ٢٣:

اكتب برنامجا بلغة تربو باسكال لإيجاد جداء مصفوفتين A و B بشكل عام ،أي يجب دراسة التوافقيه بالنسبة لعملية الضرب .

-البرنامج بلغة تربو باسكال:

```
PROGRAM MATMULT;
VAR
  A,B,C:ARRAY[1..30,1..30] OF INTEGER;
```

```
  FOR I:=1 TO R2 DO BEGIN
    FOR K:=1 TO C2 DO BEGIN
      WRITE('B[',I,',',K,']=');
      READLN(B[I,K]);
    END;
  END;
  IF C1=R2 THEN
   BEGIN
    FOR I:=1 TO R1 DO BEGIN
      FOR J:=1 TO C2 DO BEGIN
        S:=0;
        FOR K:=1 TO R2 DO BEGIN
          S:=S+A[I,K]*B[K,J];
        END;
        C[I,J]:=S;
      END;
    END;
    WRITELN;
    WRITELN('OUTPUT');WRITELN;
    FOR I:=1 TO R1 DO BEGIN
      FOR J:=1 TO C2 DO BEGIN
       WRITELN('C[',I,',',J,']=',C[I,J]);
      END;
    END;
    WRITELN;
  END;
  IF R1=C2 THEN
   BEGIN
    FOR I:=1 TO R2 DO BEGIN
      FOR J:=1 TO C1 DO BEGIN
```



```
  R1,C1,R2,C2:BYTE;
  I,J,K,S:INTEGER;

BEGIN
  WRITE('INPUT THE NUMBER OF ROWS OF FIRST
                                        MATRIX:');
  READLN(R1);
  WRITE('INPUT THE NUMBER OF COLUMNS OF FIRST
                                        MATRIX:');
  READLN(C1);
  WRITE('INPUT THE NUMBER OF ROWS OF SECOND
                                        MATRIX:');
  READLN(R2);
  WRITE('INPUT THE NUMBER OF COLUMNS OF
                                SECOND MATRIX:');
  READLN(C2);
  IF (C1<>R2) AND (R1<>C2) THEN
       WRITELN('IMPOSSIBELE')


ELSE
   BEGIN
    WRITELN('INPUT THE FIRST MATRIX:');
    FOR I:=1 TO R1 DO BEGIN
      FOR K:=1 TO C1 DO BEGIN
       WRITE('A[',I,',',K,']=');
       READLN(A[I,K]);
      END;
    END;
    WRITELN('INPUT THE SECOND MATRIX:');
```

```
      S:=0;
      FOR K:=1 TO R1 DO BEGIN
       S:=S+B[I,K]*A[K,J];
      END;
      C[I,J]:=S;
    END;
  END;
  WRITELN;
  WRITELN('OUTPUT');WRITELN;
  FOR I:=1 TO R2 DO BEGIN
    FOR J:=1 TO C1 DO BEGIN
     WRITELN('C[',I,',',J,']=',C[I,J]);
    END;
  END;
  WRITELN;
 END;
END;
READLN
END.
```

تمرين ٢٤:

اكتب برنامجا بلغة تربو باسكال لإيجاد جداء مصفوفتين A و B متوافقتين بالنسبة لعملية الضرب .أي أن عدد أعمدة المصفوفه الأولى يجب أن يســـاوي عدد أسطر المصفوفه الثانية . ويكون الناتج مصفوفه جديدة عدد

اسطرها يساوي عدد أسطر المصفوفه الأولى وعدد أعمدتها يساوي عدد أعمدة المصفوفه الثانية .

نوضح ذلك بالشكل التالي:

$$A\,[I,K]_{(M,P)} \cdot B\,[K,J]_{(P,N)} = C\,[I,J]_{(M,N)}$$

-البرنامج بلغة تربو باسكال:

```
PROGRAM MATMULT;
VAR
  A,B,C:ARRAY[1..30,1..30] OF INTEGER;
  R1,C1,R2,C2:BYTE;
  I,J,K,S:INTEGER;
BEGIN
 WRITE('INPUT THE NUMBER OF ROWS OF FIRST
                                    MATRIX:');
 READLN(R1);
 WRITE('INPUT THE NUMBER OF COLUMNS OF FIRST
                                    MATRIX:');
 READLN(C1);
 WRITE('INPUT THE NUMBER OF ROWS OF SECOND
                                    MATRIX:');
 READLN(R2);
 WRITE('INPUT THE NUMBER OF COLUMNS OF
                                SECOND MATRIX:');
 READLN(C2);
 IF (C1<>R2) AND (R1<>C2) THEN
     WRITELN('IMPOSSIBELE')
```

```
ELSE
 BEGIN
  WRITELN('INPUT THE FIRST MATRIX:');
  FOR I:=1 TO R1 DO BEGIN

    FOR K:=1 TO C1 DO BEGIN
     WRITE('A[',I,',',K,']='); READLN(A[I,K]);
    END;
  END;
  WRITELN('INPUT THE SECOND MATRIX:');
  FOR I:=1 TO R2 DO BEGIN
    FOR K:=1 TO C2 DO BEGIN
     WRITE('B[',I,',',K,']=');READLN(B[I,K]);
    END;
  END;
 IF C1=R2 THEN
 BEGIN
  FOR I:=1 TO R1 DO BEGIN
    FOR J:=1 TO C2 DO BEGIN
     S:=0;
     FOR K:=1 TO R2 DO BEGIN
      S:=S+A[I,K]*B[K,J];
     END;
     C[I,J]:=S;
    END;
  END;
  WRITELN;
  WRITELN('OUTPUT');WRITELN;
  FOR I:=1 TO R1 DO BEGIN
    FOR J:=1 TO C2 DO BEGIN
```



```
     WRITELN('C[',I,',',J,']=',C[I,J]);
    END;
  END;
  WRITELN;
 END;
IF R1=C2 THEN
 BEGIN
  FOR I:=1 TO R2 DO BEGIN
    FOR J:=1 TO C1 DO BEGIN
     S:=0;
     FOR K:=1 TO R1 DO BEGIN
      S:=S+B[I,K]*A[K,J];
     END;
     C[I,J]:=S;
    END;
   END;
   WRITELN;
   WRITELN('OUTPUT');WRITELN;
   FOR I:=1 TO R2 DO BEGIN
     FOR J:=1 TO C1 DO BEGIN
      WRITELN('C[',I,',',J,']=',C[I,J]);
     END;
   END;
   WRITELN;
  END;
 END;READLN
END.
```

إن ناتج تنفيذ البرنامج وفق القيم المدخله هو بالشكل التالي:

INPUT THE NUMBER OF ROWS OF FIRST MATRIX:2
INPUT THE NUMBER OF COLUMNS OF FIRST
MATRIX=
THE NUMBER OF ROWS OF SECOND MATRIX=:3

INPUT THE NUMBER OF COLUMNS OF SECOND
MATRIX:2

INPUT THE FIRST MATRIX:

A[1,1]=1
A[1,2]=2
A[1,3]=3
A[2,1]=4
A[2,2]=5
A[2,3]=6

INPUT THE SECOND MATRIX:

B[1,1]=7
B[1,2]=8
B[2,1]=9
B[2,2]=10
B[3,1]=11
B[3,2]=12

OUTPUT

C[1,1]=58

C[1,2]=64
C[2,1]=139
C[2,2]=154

تمرين ٢٥:

اكتب برنامجا بلغة تربو باسكال لإيجاد مجموع مصفوفتين A و B متسويتي السعة .

- البرنامج بلغة تربو باسكال:

```
PROGRAM SUMMATRIX;
VAR
 A,B,C:ARRAY[1..30,1..30] OF INTEGER;
 I,J,S,M,N:INTEGER;
BEGIN
 WRITELN('INPUT THE NUMBER OF ROWS OF FIRST
MATRIX=:');
 WRITE('THE NUMBER OF ROWS OF SECOND
MATRIX=:');
 READLN(N);
 WRITELN('INPUT THE NUMBER OF COLUMNS OF
FIRST MATRIX=');
 WRITE('INPUT THE NUMBER OF COLUMNS OF
SECOND MATRIX:');
 READLN(M);
 WRITELN('INPUT THE FIRST MATRIX:');
 FOR I:=1 TO N DO BEGIN
    FOR J:=1 TO M DO BEGIN
     WRITE('A[',I,',',J,']=');READLN(A[I,J]);
```

```
    END;
  END;
  WRITELN('INPUT THE SECOND MATRIX:');
  FOR I:=1 TO N DO BEGIN
   FOR J:=1 TO M DO BEGIN
    WRITE('B[',I,',',J,']=');READLN(B[I,J]);
   END;
  END;
  FOR I:=1 TO N DO BEGIN
     FOR J:=1 TO M DO BEGIN
        S:=0;
        S:=S+A[I,J]+B[I,J];
        C[I,J]:=S;
     END;
  END;
  WRITELN;
  WRITELN('OUTPUT');WRITELN;
  FOR I:=1 TO N DO BEGIN
     FOR J:=1 TO M DO BEGIN
       WRITELN('C[',I,',',J,']=',C[I,J]);
     END;
  END;
  WRITELN;
READLN
END.
```


تمرين ٢٦:

اكتب برنامجا بلغة تربو باسكال لقراءة قيمة الدخل الوطني لعدد من الدول وهي سوريا ، لبنان ، الأردن ، مصر ، تونس ، المغرب . وإيجاد أعلى قيمة دخل بين هذه الدول .

البرنامج بلغة تربو باسكال:

```
Program Income;
Type
   vec=array[1..8] of string;
   vect=array[1..8] of real;
Var
   A:vec;B:vect;I:integer;
   Max1:string;Max2:real;
Begin
  writeln('Input The Incomes of The Countries:');
  A[1]:='Syria';a[2]:='Lebanon';
  a[3]:='Jordan';a[4]:='Egypt';
  a[5]:='Libya';a[6]:='Tunisia';
  a[7]:='Maroc';
  for I:=1 to 7 do
   begin
    write('Input The Incom of ',A[I],' : ');
    readln(B[I]);
   end;
  Max2:=B[1];Max1:=A[1];
  for I:=2 to 7 do
   if B[I]>Max2 then
```

```
    begin
     Max2:=B[I];Max1:=A[I];
    end;
  writeln;
  writeln(Max1,' ',Max2);
readln;
End.
```

تمرين ٢٦:

إن مجموع مربعات الأعداد الصحيحه من 1 إلى n يعطى بالعلاقة:

$$1+2^2+3^2+4^2+.....+n^2 = n\ (n+1)\ (2n+1)\ /\ 6$$

والمطلوب كتابة تابع منطقي يأخذ كوسيط شكلي القيمة n ويعطي نتيجة تدل على صحة العلاقة أو عدم صحتها.

- البرنامج بلغة تربو باسكال:

```
Program RightOrWrong;
Var
  n:integer;
Function f(n:integer):boolean;
var
  i:longint;
  r,s:real;
begin
  s:=0;
  for i:=1 to n do
      s:=s+sqr(i);
```



```
r:=(n*(n+1)*(2*n+1))/6;
  writeln(s:8:2,' ',r:8:2);
  if s=r then f:=true;
end;
Begin
   write('n=');readln(n);
   writeln(f(n));
readln;
End.
```

تمرين ٢٧:

يقال عن مصفوفة مربعة بأنها ذات قطر مسيطر إذا كانت القيمة المطلقة لكل عنصر موجود على القطر الرئيسي أكبر من مجموع القيم المطلقة للعناصر الموجودة معه في الصف نفسه أي أنه مهما تكن i يكون

$$|C_{ii}| > |C_{i1}|+|C_{i2}|+\cdots+|C_{i,i-1}|+|C_{i,i+1}|+\cdots+|C_{in}|$$

والمطلوب كتابة برنامج يقرأ مصفوفة مربعة ويتحقق من كونها ذات قطر مسيطرأم لا وذلك بكتابة تابع منطقي يختبر ما إذا كانت المصفوفة المعطاة ذات قطر مسيطر أم لا(البرنامج يطبع القطر المسيطر فقط).

- البرنامج بلغة تربو باسكال:

```
PROGRAM MATRIX;
VAR
  A:ARRAY[1..10,1..10] OF INTEGER;
  V:ARRAY[1..10] OF BOOLEAN;
  W:BOOLEAN;
  N,I,J,S:INTEGER;
```

```
BEGIN
  WRITELN('INPUT N<=10 ');
  READLN(N);
  FOR I:=1 TO N DO
   FOR J:=1 TO N DO
    begin
     write('A[',I,',',J,']=');
     READ(A[I,J]);
    end;
  FOR I:=1 TO N DO
   BEGIN
    S:=0;
    FOR J:=1 TO N DO
      IF I<>J THEN
         S:=S+ABS(A[I,J]);
    IF ABS(A[I,I])>S THEN
          V[I]:=TRUE
     ELSE    V[I]:=FALSE;
   END;
   W:=TRUE;
   FOR I:=1 TO N DO
    IF V[I]=FALSE THEN
           W:=FALSE;
   IF W=FALSE THEN
       WRITELN('WRONG')
   ELSE
     FOR I:=1 TO N DO
      WRITELN(A[I,I]:5);
READLN;READLN
```



```
END.
```

إن ناتج تنفيذ البرنامج وفق القيم المدخله هو بالشكل التالي:

```
INPUT N<=10
4
A[1,1]=11
A[1,2]=2
A[1,3]=3
A[1,4]=4
A[2,1]=5
A[2,2]=55
A[2,3]=6
A[2,4]=7
A[3,1]=8
A[3,2]=9
A[3,3]=99
A[3,4]=10
A[4,1]=11
A[4,2]=12
A[4,3]=13
A[4,4]=44

  55
  99
  44
```

مما تقدم نجد أن المصفوفة المعطاة ذات قطر مسيطر ، وهو المطلوب .

تمرين ٢٨:

يقـــوم برنامج بتخزين رقم لا على التعيين X في ذاكرته بين العدد 0 والعدد 100 ، ثم يقوم البرنامج بسؤال المستخدم عن الرقم ،إما أن يكون الرقم المخزن أكبر من الرقم المدخل أو أصغر منه وهكذا يكرر البرنامج السؤال حتى يحـــزر المستخدم الرقم فيطبع البرنامج عدد المحاولات والرقم . اكتب برنامجا بلغة تربو باسكال لتنفيذ ذلك .

- البرنامج بلغة تربو باسكال:

```
Program Game;
Uses crt;
Label 1,2;
Var x,i,n:integer;
   z:char;
Begin
 2:clrscr;
  randomize;
  x:=random(100);
  writeln('enter a number between 1_100');
  i:=0;
  1:write('Your number is=');readln(n);
  i:=i+1;
  if n>x then begin
     writeln('Your number is large');
```



```
    goto 1; end;
  if n<x then begin
     writeln('Your number is small');
     goto 1;end
  else
    begin
      writeln('You win');
      writeln('the number is=',n);
      writeln('Your tries is=',i);
    end;
  write('Do you want to play again(y/n)?');readln(z);
  if (z='y') or (z='Y') then goto 2;
  writeln('press any key to stop');
  readln
End.
```

تمرين ٢٩:

اكتب برنامج بلغة تربو باسكال لحساب حجم مخروط ، حيث أن حجم المخروط يتراوح مابين 10 و 20 Inches .

إن حجم المخروط يعطى بالعلاقة التالية: V=H.(22/7).(R.R)/3

- البرنامج بلغة تربو باسكال هو بالشكل التالي:

```
PROGRAM VOLUMES;
{THIS PROGRAM TO CALCULATE THE VOLUME OF A
CONE.WITHIN THE RANGE OF 10 TO 20 CUBIC INCHES}
CONST
     PI=3.1415;MINRANGE=10;MAXRANGE=20;
VAR
```

```
      H,R,V:REAL;
      VRANGE:BOOLEAN;
BEGIN
   WRITE('HEIGHT=');READLN(H);
   WRITE('RADIUS=');READLN(R);
   V:=H*PI*SQR(R)/3;
   IF (V>=MINRANGE) AND (V<=MAXRANGE) THEN
         VRANGE:=TRUE
   ELSE
         VRANGE:=FALSE;
   IF VRANGE THEN
    WRITELN('VOLUME=',V:8:2,' IS IN REQUIRED
                                        RANGEE')
   ELSE
    WRITELN('VOLUME=',V:8:2,' IS DISREGARD
                                        VOLUME');
READLN
END.
```

تمرين ٣٠:

اكتب برنامجا بلغة تربو باسكال لإيجاد القاسم المشترك الأعظم لأربعة أعداد وذلك بواسطة تابع function يحسب القاسم من أجل عددين ويستدعي البرنامج التابع من أجل ايجاد القاسم لأربعة أعداد.

- البرنامج بلغة تربو باسكال هو بالشكل التالي:

```
PROGRAM DENOMINATOR;

VAR
```



```
  NUM1,NUM2,NUM3,NUM4:INTEGER;
  ANS1,ANS2,FINAL_ANS:INTEGER;



FUNCTION GCD(L,M:INTEGER):INTEGER;
VAR
     S,R:INTEGER;
BEGIN
   IF L<M THEN BEGIN
     S:=L;L:=M;M:=S;
   END;
   REPEAT
      R:=L MOD M;
      IF R<>0 THEN BEGIN
            L:=M;M:=R;
      END;
   UNTIL R=0;
   GCD:=M;
END;


BEGIN
   WRITE('NUM1=');READLN(NUM1);
   WRITE('NUM2=');READLN(NUM2);
   ANS1:=GCD(NUM1,NUM2);
   WRITE('NUM3=');READLN(NUM3);
   WRITE('NUM4=');READLN(NUM4);
   ANS2:=GCD(NUM3,NUM4);
   FINAL_ANS:=GCD(ANS1,ANS2);
```

```
   WRITE('THE GCD OF ',NUM1,' ',NUM2,' ',
                                NUM3,' ',NUM4,' IS=');
   WRITELN(FINAL_ANS);
READLN
END.
```

تمرين ٣٢:

يسير قطار على خط سكة حديدية مارا بالمدن التالية:

دمشق ، حمص ، حماه ، حلب ، الرقه ، دير الزور . نريد بناء شـــعاع معرف علـــى هـــذه المدن بحيث تكون قيمة كل مركبه هي الســـاعة التي يمر فيها هذا القطار بهذه المدينه(من 0 إلى 23) . والمطلوب :

١- تعريف النمط المعدود اللازم .

٢- تعريف الشعاع الممثل لمواعيد مرور القطار في المدن المذكوره أعلاه.

٣- كتابة برنامج بلغة باسكال يتعامل مع البنى المعرفة في 1 و 2 ويقوم بما يلي :

أ- ملئ المواعيد حسب القيم المدخلة من المستثمر .

ب- يطلب مـــن المستثمر إدخال المدينة التي هو فيها والساعة التي يرغـــب بالسفر فيها من هذه المدينة ويعطي رسالة تبين كم عليه أن ينتظر أو ربمـــا عليه الإنتظار حتى الغد لأن مـــوعد مرور القطار بالمدينة يسبق الساعة التي يريد أن يسافر فيها .

البرنامج بلغة تربو باسكال هو بالشكل التالي:


```
Program Train;
Type
  D=(Damas,Homs,Hama,Aleppo,Raka,Deralzor);
  vec=array[D] of 0..23;

Var
  X:string;
  H,R:integer;
  C:vec;
Begin
  writeln('Input The Times of The Train Arraiving to The
                                                Cities:');
  write('Damas=');readln(C[Damas]);
  write('Homs=');readln(C[Homs]);
  write('Hama=');readln(C[Hama]);
  write('Aleppo=');readln(C[Aleppo]);
  write('Raka=');readln(C[Raka]);
  write('Deralzor=');readln(C[Deralzor]);
  write('Enter Your City ');readln(X);
  write('Enter your Time[0-23]');readln(R);
  if X='Damas' then H:=C[Damas]-R;

  if X='Homs' then H:=C[Homs]-R;
  if X='Hama' then H:=C[Hama]-R;
  if X='Aleppo' then H:=C[Aleppo]-R;
  if X='Raka' then H:=C[Raka]-R;
  if X='Deralzor' then H:=C[Deralzor]-R;
  if H<0 then writeln('Tomorrow')
  else
```

```
    write('staying ',H,' Hour');
readln;
End.
```

تمرين ٣٢:

تعرف متتالية Fibonacci بالعلاقة التالية:

من أجل

$$n >= 2 \Rightarrow f_n = f_{n-1} + f_{n-2}$$
$$f_0 = 0 \ , \ f_1 = 1$$

والمطلوب :

١- كـتابة تابع عودي لحساب قيمة f_n من أجل قيمة ما n تعطى من قبل المستثمر .

٢- كتابة تابع غير عودي لحساب قيمة f_n ولا يعتمد على وجود بنية شعاع لتخزين قيم f_n .

البرنامج بلغة تربو باسكال هو بالشكل التالي:

-١

```
Program Fibonacci1;
Var n:integer;
Function f(n:integer):integer;
begin
  if n=0 then f:=0
```



```
  else if n=1 then f:=1
  else f:=f(n-1)+f(n-2);
end;
Begin
  write('n=');readln(n);
  writeln('fibonacci(',n,')=',f(n));
readln
End.
```

-٢

```
Program Fibonacci2;
Uses crt;
Var
   fib,b:integer;
   i,j,m,n:integer;
Begin
   clrscr;
   write('Input n=');readln(n);
   i:=0;j:=1;m:=2;
   for m:=2 to n do
    begin
     fib:=i+j; i:=j; j:=fib;
    end;
   if n=0 then fib:=0;
   if n=1 then fib:=1;
   writeln('Fibonacci(',n,')=',fib);readln
End.
```

تمرين ٣٣:

يعرف تابع Ackermann بالعلاقات التالية:

$$if \quad n>0 \Rightarrow A(0,n)=n+1$$
$$if \quad m>0 \Rightarrow A(m,0)=A(m-1,1)$$
$$if \quad m,n>0 \Rightarrow A(m,n)=A(m-1,A(m,n-1))$$

والمطلوب:

اكتب تابع عودي يسمح بحسـاب قيمة التابع Ackermann في حالة وسيطين m و n .

البرنامج بلغة تربو باسكال هو بالشكل التالي:

```
Program Ackermann;
Var m,n:integer;
Function A(m,n:integer):integer;
begin
  if (m=0) and (n=0) then A:=n+1;
  if (m>0) and (n=0) then A:=A(m-1,1);
  if (m>0) and (n>0) then A:=A(m-1,A(m,n-1));
end;
Begin
  writeln('A(m,n)');
  write('Input m=');readln(m);
  write('Input n=');readln(n);
  A(m,n);
  writeln('A(',m,',',n,')=',A(m,n));
readln
End.
```


تمرين ٣٤:

اكتب برنامجا لاختبار قابلية قسـمة عدد على 7 ، و ذلك دون استخدام الشرط إذا كان العدد قابل القسمه على 7 (أي دون استخدام if n mod 7<> 0 then) ، و المطلوب كتابة خوارزميه التحقق من أن العدد قابل القسمه على 7 .

الحل:

بشكل عام نعلم أن العدد القابل القسمه على 7 هو العدد الذي يكون باقي قسمته على 7 هو 0 . ولكن لن نستخدم هذه الطريقه ، وسوف نستخدم طريقه آخرى، نكتب الخوارزمية التالية:

الخوارزمية:

١- اقرأ عددا x .

٢- كرر حتى يتحقق t<=9 .

٢-١- y:=x mod 10 . ايجاد باقي قسمة العدد على 10 ووضعه في y .

٢-٢- z:=x div 10 .ايجاد ناتج قسمة العدد على 10 ووضعه في z .

٢-٣- t:=|z-(2*y)| ايجاد القيمه المطلقة للفرق بين z و ضعف y و وضعه في t .

٢-٤- اسناد t إلى x .

٣- اذا t=7 أو t=0 عندئذ نفذ :

اكتب العدد هو مضاعف للعدد 7

وإلا اكتب العدد ليس مضاعف .

البرنامج بلغة تربو باسكال بالشكل التالي:

```
Program Divider;
Var
  x,y,z,t:longint;
Begin
  writeln('Input number');
  readln(x);
  repeat
    y:=x mod 10;
    z:=x div 10;
    t:=abs(z-(2*y));
    x:=t;
  until (t<=9);
  if (t=7) or (t=0) then
   writeln('the number is double of 7')
  else
   writeln('no double');
readln
End.
```


ملاحظه:

يمكن اختبار قابلية قسمة عدد على 7 وفق القاعدة التالية:

سوف نرمز للعدد وبشكل عام بالرمز Q حيث :

$$Q = M\ L\ K\ J\ I\ H\ F\ D\ C\ B\ A$$

يكون العدد Q قابل للقسمه على 7 إذا كان المجموع التالي قابلا للقسمه على 7.

$$\begin{aligned} Q = {} & A + B + C + D + E + F + G + H + I + J + K + L + M + \\ & A.0 + B.2 + C.1 + D.5 + E.3 + F.4 + G.7 + H.2 + I.1 + \\ & J.5 + K.3 + L.4 + M.7 \end{aligned}$$

إن القاعدة المذكوره أعلاه تحسب لنا قابلية قسمة عدد على 7 وذلك للأعداد التي لا يتجاوز عدد أرقامها 11 رقما ،لكن في مثالنا الحد الأعلى للأعداد الصحيحه الممكـن تمثيلها أو المعرفه في Longint هو 2×10^9 . بكلام آخر المطلوب إدخال عدد بين العدد 1 و المليار.

-البرنامج بلغة تربو باسكال بالشكل التالي:

```
Program Divideby7;

label start;

Var
  A,B,C,D,E,F,G,H,I,J,Q:integer;
  N,N1,N2,N3,N4,N5,N6,N7,N8,N9,N10:longint;
```

```
  R:real;
  Z,W:char;

Begin
    A:=0;B:=0;C:=0;D:=0;E:=0;F:=0;G:=0;H:=0;I:=0;J:=0;
start:writeln('input number>0 and<1*10^9');
    readln(R);



   if R<2000000000 then
    begin
     N:=trunc(R);
     N1:=N div 10; A:=N mod 10;
     N2:=N1 div 10; B:=N1 mod 10;
     N3:=N2 div 10; C:=N2 mod 10;
     N4:=N3 div 10; D:=N3 mod 10;
     N5:=N4 div 10; E:=N4 mod 10;
     N6:=N5 div 10; F:=N5 mod 10;
     N7:=N6 div 10; G:=N6 mod 10;
     N8:=N7 div 10; H:=N7 mod 10;
     N9:=N8 div 10; I:=N8 mod 10;
     N10:=N9 div 10; J:=N9 mod 10;
   Q:=A+B+C+D+E+F+G+H+I+J+A*0+B*2+C*1+
     D*5+E*3+F*4+G*7+H*2+I*1+J*5;
  if Q mod 7=0 then
   begin
     writeln('this number is abel to divide by 7');
     writeln('do you whant to get the division anser(Y/N)');
     readln(W);
```



```
     if (W='Y') or (W='y') then writeln('the result
                                        is=N/7=',N/7)
    end
  else writeln('sorry your number is not abel to divide by
                                                    7');
   end
  else writeln('this number is over 2*10^9 you ought to
                          input a number less than 2*10^9');
  writeln('do you whant another count(Y/N)');
  readln(Z);
  if (Z='Y') or (Z='y') then goto start
  else begin
     writeln('thank you for choosing our program');
     writeln('by by press enter to exit');
     end;
     readln
  End.
```

وهو المطلوب .

إذا كان المطلوب إيجاد الأعداد القابله القسمه على 7 وذلك للأعداد بين العدد 1 وعدد ما مدخل فإن البرنامج يكون بالشكل التالي:

```
Program Divideby7;

Var
   A,B,C,D,E,F,G,H,I,J,Q:integer;
   N,N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,R:longint;
```

N=1617 N=1624 N=1631 N=1638 N=1645 N=1652 N=1659
N=1666 N=1673 N=1680 N=1687 N=1694 N=1701 N=1708
N=1715 N=1722 N=1729 N=1736 N=1743 N=1750 N=1757
N=1764 N=1771 N=1778 N=1785 N=1792 N=1799 N=1806
N=1813 N=1820 N=1827 N=1834 N=1841 N=1848 N=1855
N=1862 N=1869 N=1876 N=1883 N=1890 N=1897 N=1904
N=1911 N=1918 N=1925 N=1932 N=1939 N=1946 N=1953
N=1960 N=1967 N=1974 N=1981 N=1988 N=1995
by by press enter to exit

وهو المطلوب .

تمرين ٣٥:

لتمثيل أي كثير حدود في الحاسب نحتاج إلى معرفة درجته n ،حيث n عدد صحيح ، وقيم الأمثال $a_n, \ldots\ldots, a_1, a_0$ هي أعداد حقيقية .

والمطلوب:

١- كتابة تابع يحسب القوة n لعدد ما x ، حيث x عدد حقيقي ، و n عدد صحيح .

٢- نريد حساب قيمة كثير الحدود عند النقطة x_0 باستخدام طريقة horner . والمطلوب كتابة تابع يأخذ الوسطاء التالية: كثير الحدود وفق التمثيل الذي اخترته له، والنقطة x_0 ،ويعيد قيمة كثير الحدود عند النقطة x_0 باستخدام العلاقات التالية:

$$y_n = a_n$$
$$y_{n-1} = a_{n-1} + xy_n$$
.....
.....
$$y_n = a_0 + xy_1 = p(x)$$

٣- لنأخذ حالا كثيرات الحدود من الدرجة الثانية. ونريد حل المعادلة p(x)=0 . اكتب تابعا لحساب مميز المعادلة .

٤- اكتب برنامجا يقرأ كثير الحدود p(x) و نقطة x_0 و يستدعي التابع المكتوب في 2 لحساب قيمة كثير الحدود ، وإظهار الناتج بمقارنته مع القيمة الناتجة من حساب قيمة كثير الحدود باستخدام التابع المكتوب في 1 . و إذا كان التابع من الدرجة الثانية يطلب حل المعادلة p(x) باستخدام التابع المكتوب في 3 و إظهار جذر أو جذري المعادله .

-البرنامج بلغة تربو باسكال بالشكل التالي:

```
Program Horner;
Type
  vector=array[0..200] of real;
Var
  x,s,x1,x2:real;
  n,i:integer;
  a:vector;

function f(x:real;n:integer):real;
begin
   f:=exp(n*ln(x));
```

```
end;

function p(a:vector;x:real):real;
var
  y:vector;
  i:integer;
begin
    y[n]:=a[n];
    for i:=n downto 2 do
       y[i-1]:=a[i-1]+x*y[i];
    y[0]:=a[0]+x*y[1];
    p:=y[0];
end;

function d(a:vector;n:integer):real;
begin
    d:=sqr(a[1])-4*a[0]*a[2];
end;

Begin
    write('n=');readln(n);
    for i:=0 to n do
     begin
      write('a[',i,']=');readln(a[i]);
      end;
    write('x0=');readln(x);
    write(p(a,x):8:2);
    s:=0;
    for i:=0 to n do
     s:=s+a[i]*f(x,i);
```



```
  writeln(' ',s:8:2);
  if (n=2) and (d(a,n)>0) then
   begin
    x1:=(-a[1]-sqrt(d(a,n)))/2*a[0];
    x2:=(-a[1]+sqrt(d(a,n)))/2*a[0];
    writeln('The solution: ','x1=',x1:8:2,' ','x2=',x2:8:2);
   end;
  if (n=2) and (d(a,n)=0) then
   begin
    x1:=-a[1]/2*a[0];
    writeln('The solution ','x1=',x1:8:2);
   end;
  if (n=2) and (d(a,n)<0) then
     writeln('Imposible');
readln
End.
```

وهو المطلوب .

تمرين ٣٦:

اكتب برنامجا بلغة تربو باسكال لحساب ما يلي:

١- حجم اسطوانه .

٢- السطح الخارجي لإسطوانه .

٣- السطح الكلي للإسطوانه .

علما أن :

الحجم = الإرتفاع × مربع نصف القطر × ٢٢/٧

السطح الخارجي = الإرتفاع × القطر × ٢٢/٧

السطح الكلي = القطر × ٢٢/٧ × (ارتفاع + نصف القطر)

البرنامج بلغة تربو باسكال هو بالشكل التالي:

```
PROGRAM VOL_GPAPHIC;
USES CRT;
LABEL 4,5;
CONST PI=3.1415;
VAR
  A:BYTE;
  H,R,V,M,O:REAL;
  W:CHAR;
BEGIN
 4: CLRSCR;
 5: WRITELN('CHOOSE BETWEEN NUMBERS 1,2,3');
   READLN(A);
   CASE A OF
 CASE A OF
  1:BEGIN
    WRITE('INPUT H=');READLN(H);
    WRITE('INPUT R=');READLN(R);
    V:=H*SQR(R)*PI;
    WRITELN('V=',V:20:3);
   END;
  2:BEGIN
    WRITE('INPUT H=');READLN(H);
    WRITE('INPUT R=');READLN(R);
    M:=H*2*R*PI;
    WRITELN('M=',M:20:3);
   END;
  3:BEGIN
```



```
    WRITE('INPUT H=');READLN(H);
    WRITE('INPUT R=');READLN(R);
    O:=2*R*PI*(R+H);
    WRITELN('O=',O:20:3);
   END;
  ELSE
   WRITELN;
   WRITELN('FALSE INPUT');
   WRITELN;
   GOTO 5;
END;
WRITELN('DO YOU WHANT ANOTHER COUNT');
READLN(W);
IF W='Y' THEN GOTO 4
ELSE
WRITELN('PRECE ENTER TO EXIT');
READLN
END.
```

وهو المطلوب.

تمرين ٣٧:

اكتب برنامجا لإدخال أسماء الطلاب و درجاتهم ثم يطبع أرقامهم وأسماءهم ودرجاتهم دفعة واحدة. بفرض أن الحد الأعلى للإدخال ١٠٠ طالب.

البرنامج بلغة تربو باسكال:

```
PROGRAM STUDENTS;
CONST MAXN=100;
VAR
```

```
  N,I:INTEGER;
  STNM:ARRAY[1..MAXN] OF STRING:
  M:ARRAY[1..MAXN] OF INTEGER;
BEGIN
 WRITELN('INPUT N ;N<=100');
 READLN(N);
 FOR I:=1 TO N DO
 BEGIN
  WRITELN('INPUT STUDENT NAME NUMBER '.I.'
                                        AND HIS MARK');
  READLN(STNM[I].M[I]);
 END;
 WRITELN('STUDENT NUMBER',' STUDENT NAME  ',
                                       'STUDENT MARK');
 WRITELN('-----------------------------------------');
  FOR I:=1 TO N DO
  BEGIN
   WRITELN(I:8,STNM[I]:16,M[I]:12);WRITELN:
  END;
READLN
END.
```

تمرين٣٨:

اكتـــب برنامجا لقراءة مجموعة من الأعداد، ثم يطبع كل الأعداد التي هـــي أكبر من 10 والأعداد التي هي أقل من 10 والأعداد الزوجية ويســـأل الـــبرنامج فيما إذا أردنا إدخال مجموعة أخرى من الأعداد ويكرر العمل السابق حتى تطلب من البرنامج الخروج، تفصل النتائج بخط من الشكل(-----).

البرنامج بلغة تربو باسكال هو بالشكل التالي:


```
PROGRAM NUMBER;

LABEL 2;

VAR
  N,I:INTEGER;
  X:ARRAY[1..100] OF INTEGER;
  W:CHAR;

BEGIN
 2: WRITE('HOU MANY NOMBERS DO YOU WHANT TO
                                        READ=');
 READLN(N);
 FOR I:=1 TO N DO
  BEGIN
    WRITE('X[',I,']=');READLN(X[I]);
 END;
WRITE('NUMBERS>10= ');
FOR I:=1 TO N DO
 IF X[I]>10 THEN
   WRITE(X[I],' ');
WRITELN;WRITELN('----------------');
WRITE('NUMBERS<0= ');
 FOR I:=1 TO N DO
  IF X[I]<0 THEN
    WRITE(X[I],' ');
WRITELN;WRITELN('----------------');
WRITE('EVEN NUMBERS= ');
FOR I:=1 TO N DO
 IF X[I] MOD 2=0 THEN
```

```
    WRITE(X[I],' ');
  WRITELN;
  WRITE('DO YOU WHANT TO INPUT ANOTHER
                                        NUMBERS (Y/N):');
  READLN(W);
  IF W='Y' THEN GOTO 2
  ELSE
READLN
END.
```

تمرين ٣٩:

اكتب برنامجـا لإيجاد أكبر و أصغر عدد مـن مجموعة أعداد معطاة باستخدام الإجرائيات.

البرنامج بلغة تربو باسكال هو بالشكل التالي:

```
PROGRAM MAXIM_MINM;

CONST M=200;

TYPE VECTOR=ARRAY[1..M] OF REAL;

VAR
  X:VECTOR;
  MAX,MIN:REAL;
  LONG,Q:INTEGER;

PROCEDURE INPUTVEC(VAR A:VECTOR;L:INTEGER);
VAR
```



```
  I:INTEGER;
BEGIN
  WRITELN('INPUT THE VALUES');
  FOR I:=1 TO L DO BEGIN
    WRITE('A[',I,']=');READLN(A[I]);
  END;
END;

PROCEDURE MAXMIN(A:VECTOR;N:INTEGER;
                          VAR LARG,SMAL:REAL);
VAR
  HSMAL,HLARG:REAL;
  I:INTEGER;
BEGIN
  IF A[1]<A[2] THEN BEGIN
    SMAL:=A[1];LARG:=A[2];
  END
  ELSE BEGIN
    SMAL:=A[2];LARG:=A[1];
  END;
  HSMAL:=SMAL;
  HLARG:=LARG;
  FOR I:=3 TO N DO
  IF A[I]<HSMAL THEN
     HSMAL:=A[I]
  ELSE
    IF A[I]>HLARG THEN
      HLARG:=A[I];
    SMAL:=HSMAL;
    LARG:=HLARG;
```

```
END;

BEGIN
 REPEAT
  Q:=1;
  WRITE('INPUT LONG=');READLN(LONG);
  INPUTVEC(X,LONG);
  MAXMIN(X,LONG,MAX,MIN);
  WRITELN('MAX=',MAX:8:2,'   MIN='.MIN:8:2);
  WRITE('INPUT Q=');READLN(Q):
 UNTIL Q=0;
END.
```

تمرين ٤٠ :

اكتب برنامجا لإيجاد القاسم المشترك الأعظم لأربعة أعداد وذلك بواسطة تابع، يحسب القاسم من أجل عددين ويستدعي التابع لإيجاد القاسم لأربعة أعداد.

```
PROGRAM DENOMINATOR;
VAR
  NUM1,NUM2,NUM3,NUM4:INTEGER;
  ANS1,ANS2,FINAL_ANS:INTEGER;

FUNCTION GCD(L,M:INTEGER):INTEGER;
VAR
  S,R:INTEGER;
BEGIN
   IF L<M THEN BEGIN
```



```
     S:=L;L:=M;M:=S;
   END;
   REPEAT
      R:=L MOD M;
      IF R<>0 THEN BEGIN
            L:=M;M:=R;
      END;
   UNTIL R=0;
   GCD:=M;
END;

BEGIN
   WRITE('NUM1=');READLN(NUM1):
   WRITE('NUM2=');READLN(NUM2);
   ANS1:=GCD(NUM1,NUM2):
   WRITE('NUM3=');READLN(NUM3):
   WRITE('NUM4=');READLN(NUM4);
   ANS2:=GCD(NUM3,NUM4);
   FINAL_ANS:=GCD(ANS1,ANS2):
   WRITE('THE GCD OF ',NUM1,' '.NUM2,' ',NUM3,'
                                        ',NUM4.' IS='):
   WRITELN(FINAL_ANS);
READLN
END.
```

تمرين ٤١ :

اكتب برنامجا يقوم بقلب الأحرف مثل المرآة، وإذا كان الحرف صغيرا يحوله إلى حرف كبير.(مثلا Ahmad تصبح DAMHA).

البرنامج بلغة تربو باسكال يصبح بالشكل التالي:

```
Program Reflection;
Uses crt;
Const
   Maxn=254;
Type
  text=array[0..Maxn]of char;
Var
  X,Y:text;
  N,I:integer;
Begin
  clrscr;
  write('Input The Number of Characters:');
  readln(N);
  for I:=1 to N do
  begin
   write('X[',I,']=');
   readln(X[I]);
  end;
writeln;
for I:=1 to N do
  write(X[I]);
writeln;
for I:=1 to N do
 begin
  if (ord(X[I])>=97)and(ord(X[I])<=122)then

   Y[I]:=chr(ord(X[I])-32)
  else
   Y[I]:=X[I];
 end;
```



```
  for I:= N downto 1 do
   begin
    write(Y[I]);
   end;
readln;
End.
```

تمرين ٤١ :

لتشفير نص نقوم بما يلي: نفرض أن جميع الأحرف الأبجدية موضوعة على محيط دائرة وإننا نقوم باستبدال كل حرف بالحرف الذي يبعد عنه بمقدار k ، وبفرض k=5 ،أما بالنسبة للأرقام فيجب أن يكون مجموع الرقمين يساوي 9 ، أمـــا علامات التنقيط فتبقى كما هي. حيث أن A تتحول إلى F ، و B تتحول G ،وهكذا.

والمطلوب :

١- كتابة إجرائية للتشفير تأخذ K وسيطا والشـــعاع المحتوي على النص، وتعيد شعاعا آخر يحتوي على النص المشفر.

٢- كتابة إجرائية لفك التشفير تقوم بعكس العملية السابقة .

٣- كـــتابة برنامج يقرأ نص ويشفره، ويقرأ نص يفترض أنه مشفر ويفك تشفيره .

البرنامج بلغة تربو باسكال:

```
Program CodingAndDecoding;
Uses crt;
Label 3;
```

```
Type
  vec=array[0..200] of char;
Var
  Y:vec;
  A,B:char;
  X,I,J:integer;

Procedure code(var Y:vec;var I:integer;J,X:integer);
Label 2;
type
  vect=array[0..200] of char;
var
  A:vect;
begin
  I:=0;J:=0;X:=0;
  clrscr;
  writeln('* To End The Text Press(S)*');
2:read(A[I]);
  if (A[I]<>'$') then
   begin
    X:=ord(A[I]);
    if ((X>=65)and(X<=90))or((X>=97)and(X<=122)) then
       begin
        X:=X+5; Y[I]:=chr(X);
       end
     else
      if ((X>=48) and (X<=57)) then
       begin
        X:=105-X;Y[I]:=chr(X);
       end
```



```
     else
        Y[I]:=A[I];
     I:=I+1;
     goto 2;
    end;
  clrscr;
  for J:=0 to I do
    write(Y[J]);
readln;
end;

Procedure decode(Y:vec;I:integer;J,X:integer);
label 1;
type
   vect=array[0..200] of char;
var
  A:vect;
begin
  I:=0;J:=0;X:=0;
  clrscr;
  writeln('* To End The Text Press($)*');
1:read(A[I]);
  if (A[I]<>'$') then
   begin
    X:=ord(A[I]);
    if ((X>=70)and(X<=95))or((X>=102)and(X<=127)) then
       begin
        X:=X-5;Y[I]:=chr(X);
       end
     else
```


مدونة ملحوظة