

بِحَذْرِي بِحَذْرِي لِأَبُو الْفَطَّانِ

١٩
سلسلة تيسير علوم الحاسوب
مجموعة كتب برنامج الحاسوب

المجمع الأدبي لـ سخري



محمد عزبي



Biblioteca Alexandria

الغرفة التجارية المغربية

كمبيو ساينس



المرجع الأساسي لمستخدمي

C - Language

تأليف: عزب محمد عزب

من اجتهد: مجلدى محمد أبو العطا

الطبعة الأولى

١٤١٨ هـ - ١٩٩٦ م



كمبيوuter ساينتس

القاهرة / مصر

المركز الرئيسي : ٤٩ شارع الحجاز . أمام دار المناسبات - مصر الجديدة. القاهرة

٢٤٩١٢٩٥ - ٢٤٠٥٣٣٠ ت / فاكس :

حقوق الطبع محفوظة للناشر ، ولا يجوز نشر أي جزء من
هذا الكتاب أو إعادة طبعة أو تصويره أو احتزان مادته
العلمية بأية صورة دون موافقة كتابية من الناشر.

رقم الإيداع : ٩٦/١٠٠٦٤

977-5735-03-3 : I. S. B. N

الكتاب في سطور

يشتمل كتاب المرجع الأساسي لمستخدمي C على خمسة عشر فصلاً نوجزها فيما يلى الفصل الأول.. نظرة عامة تشمل مقدمة تاريخية عن لغة C ومميزاتها والخطوات اللازمة لكتابه وتنفيذ برنامج بلغة C والقواعد التي يجب اتباعها وأخيراً أنواع البيانات والمؤثرات.

الفصل الثاني... دوال الإدخال والإخراج والتي تسمح بقبول البيانات وإظهارها وتعديل ألوانها وطبيعتها والتعامل مع الطابعة.

الفصل الثالث... الدوارات وأنواعها والفرق بينها ومتى تستخدم واحدة دون الأخرى ، واستخدام الدوارات اللانهائية.

الفصل الرابع... التفريع المشروط وغير المشروط والتراكيب المختلفة لجملة IF.

الفصل الخامس... كيفية إنشاء الدوال والماקרו والفرق بينهما مع اعطاء أمثلة بسيطة.

الفصل السادس... المصفوفات ذات البعد وذات البعدين وكيفية الإعلان عنها والتعامل معها وطباعة عناصرها ، وإرسال المصفوفات إلى دالة كعامل ، وأخيراً أوامر المترجم.

الفصل السابع... توظيف المفاتيح والتحكم فى حركة المؤشر والتعريف بملف ANSI.SYS واستخداماته ومتطلباته والتحكم فى خصائص الحروف وكيفية تصميم قوائم الإختيارات.

الفصل الثامن... مؤشرات العناوين (Pointers) وضروره استخدامها وكيفية تمرير مصفوفات إلى الدوال باستعمال المؤشرات ، والعلاقة بين المؤشرات والعبارات الحرفية وأخيراً مصفوفة الحرفيات والمؤشرات.

الفصل التاسع... السجلات Structures يشرح معنى السجل وال الحاجه إلى استعماله وكيفية الشانه والتعامل مع عناصره ، واستعمال السجل كعنصر في سجل آخر والتعامل مع مصفوفة السجلات وكيفية الاعلان عن مؤشر من نوع Structure.

الفصل العاشر... الملفات ويتناول الطرق المختلفة للتعامل مع الملفات ودوال فتح الملف وغلقه وكيفية القراءة والكتابة حرف بحرف أو عبارة في المرة أو سجل في المرة أو مجموعة بيانات.

الفصل الحادى عشر... يشرح دوال مهمة للمبرمج تساعد في نسخ جزء من الشاشة إلى موضع آخر وحفظه واسترجاعه ، والتعامل مع الألوان والصوت بالإضافة إلى برنامج شامل يستخدم كمحرر للنصوص.

الفصل الثاني عشر... الرسم في لغة C ويشرح تهيئة الشاشة لحالة الرسم ، وكيفية رسم الخطوط بأنواعها مع تغيير الألوان ، ورسم القطع الناقص ومتعدد الخطوط وكيفية تلوين الرسم أو تظليله وكيفية الحصول على رسم بياني ذو بعدين أو ثلاثي الأبعاد وتحريرك الرسوم واستخدام الخطوط في عمل أشكال رسومية.

الفصل الثالث عشر... يشرح دوال التعامل مع الذاكرة مثل حجز مساحة من الذاكرة ، وتفرعها وإعادة تخصيصها وحجز مساحة متغيرة.

الفصل الرابع عشر... يشرح روئينات الذاكرة Rom Bios وكيفية استعمالها مع شرح لبرامج تحديد حجم الذاكرة والتعامل مع المؤشر.

الفصل الخامس عشر... عبارة عن مثال شامل يشتمل على معظم الموضوعات التي مرت بالفصل السابقة. وهذا البرنامج يمكن الاستفادة منه في إعداد برامج مماثلة تستفيد منها في حياتك العملية.

الملاحق... الملحق الأول. شفرة تبادل المعلومات ASCII ، الملحق الثاني التعامل مع محرر البرامج ، الملحق الثالث الكلمات المحجوزة في لغة C

المحتويات

١	الكتاب في سطور
ج	المحتويات
١	مقدمة

الفصل الأول

٥	نظرة عامة
٦	ميزات لغة C
٨	ثبتت قرص التمارين المرفق بالكتاب
٩	تشغيل البرنامج لأول مرة
١١	فتح ملف البرنامج
١٢	قواعد بناء البرنامج
١٢	التعامل مع البرنامج
١٣	خطوات التنفيذ
١٤	شرح البرنامج
١٥	أنواع البيانات
١٦	تسمية المتغير
١٦	الإعلان عن المتغيرات
١٧	المؤثرات Operators
١٧	مؤثرات المقارنة (Relational Operators)
١٨	المؤثرات المنطقية Logical Operators
١٩	مؤثر باقي خارج القسمة %

الفصل الثاني

٢١	دوال الإدخال والخروج
٢٢	دالة الطباعة على الشاشة () printf
٢٥	طباعة قيم المتغيرات على الشاشة
٢٨	ملاحظات
٢٨	دالة الإدخال العامة () scanf()
٣١	دوال إدخال حرف واحد() getchar(),getche(),getch()

٣١	الدالة <code>getchar()</code>
٣١	الدالة <code>getche()</code>
٣٢	الدالة <code>getch()</code>
٣٢	دالة طباعة حرف واحد <code>putchar()</code>
٣٤	دالة طباعة عبارة حرفية <code>puts()</code>
٣٥	دالة إدخال عبارة حرفية <code>gets()</code>
٣٧	دوال تحسين المدخلات والمخرجات
٣٧	دالة تغيير موضع المؤشر <code>gotoxy()</code>
٣٨	دالة تغيير لون الكتابة <code>(textcolor()</code>
٤٠	دالة تغيير لون الخلفية <code>textbackground</code>
٤١	دوال الإدخال والإخراج التي تستخدم الألوان
٤٣	دوال التعامل مع الطابعة
٤٤	تذكرة

الفصل الثالث

٤٧	الدوارات Loops
٤٨	الدورة <code>for</code>
٥١	تنفيذ أكثر من جملة مع <code>for</code>
٥٣	تغير معدل الزيادة
٥٤	تغير معدل الزيادة بالسابق
٥٥	الدوارات المتداخلة بإستخدام <code>for</code>
٥٧	الدورة اللاحقة بإستخدام <code>for</code>
٥٨	الدورة <code>while</code>
٥٩	ملاحظات
٦١	الفرق بين <code>for</code> و <code>while</code>
٦١	الدورة اللاحقة بإستخدام <code>while</code>
٦١	الدورة <code>do.....while</code>
٦٥	تذكرة

الفصل الرابع

٦٧	التفرع Branshing
٦٨	التفرع المشروط
٦٨	جملة الشرط <code>if</code>

٧١	جملة if الشرطية المتداخلة
٧٢	الجملة الشرطية if.....else
٧٥	الجملة الشرطية if.....else if
٧٨	التفرع switch.....case
٨٣	المؤثر الشرطي؟(conditional operator)
٨٤	التفرع غير المشروط
	الفصل الخامس
٨٥	الدوال والماקרו Functions & Macros
٨٦	المقصود بالدالة
٨٧	مثال لدالة بسيطة
٩٠	أنواع الدوال functions types
٩٢	استدعاء الدالة
٩٣	استدعاء الدالة بمتغيرات
٩٥	أمثلة مختلفة على أنواع الدوال
٩٥	مثال لدالة من نوع void
٩٦	مثال لدالة من نوع int
٩٨	مثال لدالة من نوع float
٩٩	معاملات الدالة الرئيسية() main
١٠٢	الماקרו Macros
١٠٢	ما المقصود بالماקרו؟
١٠٢	كيفية انشاء الماקרו
١٠٤	الفرق بين الماקרו وبين الدالة ومتى نستخدم الماקרו ومتى نستخدم الدالة
١٠٥	المشروع project
	الفصل السادس
١١١	المصفوفات Arrays
١١٢	معنى المصفوفات
١١٣	المصفوفة ذات البعد الواحد
١١٥	اعطاء قيم ابتدائية لعناصر المصفوفة
١١٦	المصفوفة الغير محددة العدد
١١٨	المصفوفة ذات البعدين
١٢٠	اعطاء قيم ابتدائية للمصفوفة ذات بعدين

١٢٣	مصفوفة العبارة الحرفية Arroy of string
١٢٥	ارسال مصفوفة للدالة كمعامل
١٢٧	دوال العبارات الحرفية string functions
١٢٨	اوامر المترجم preprocessors
الفصل السابع	
١٣٣	توظيف المفاتيح والتحكم في حركة المؤشر
١٣٤	تروظيف المفاتيح
١٣٥	تروظيسف مفاتيح الوظائف و مفاتيح التحكم
١٣٨	مثال تحديد المفتاح وكوده
١٤٢	استعمال جدول الاكراد الممتدة Extended code table
١٤٢	استخدام ملف Ansi.sys
١٤٣	متطلبات ملف Ansi.sys
١٤٣	التحكم في حركة المؤشر
١٤٦	الرسم باستخدام مفاتيح الاسهم
١٤٨	توجيه المؤشر إلى أي مكان على الشاشة
١٤٩	التحكم في خصائص الحروف
١٥١	برنامجه القائمة ذات الشريط المضاء
الفصل الثامن	
١٥٧	مؤشرات العنوانين Pointers
١٥٨	معنى المؤشر Pointer
١٥٨	الإعلان عن المؤشر Pointer
١٦٠	مزايا استخدام المؤشرات Pointers
١٦٠	اعادة اكثرب من قيمة من الدوال
١٦٤	المؤشرات والمصفوفات Pointers and Arrays
١٦٦	إرسال المصفوفة الى الدالة كمعامل
١٦٧	المؤشرات والعبارات الحرفية Pointers and Strings
١٦٩	مصفوفة المؤشرات Array of Pointer
الفصل التاسع	
١٧٣	السجلات Structures
١٧٤	معنى السجل (structure) وال الحاجة إلى إستعماله
١٧٤	استعمال السجل (structure)

١٧٦	كيفية ادخال بيانات الى عناصر السجل structure
١٧٧	وضع محتويات سجل في آخر.....
١٧٩	السجلات المتداخلة Nested Structures
١٨١	السجلات والدوال.....
١٨٣	مصفوفة السجلات Arrays of Structures
١٨٤	المؤشرات والسجلات Pointers and Structures
١٨٦	تغيير نوع البيانات Typecasting
١٨٧	اتحاد البيانات تحت اسم واحد Union
١٨٨	لماذا نستخدم Union
١٨٩	استعمال union كعنصر من عناصر structure

الفصل العاشر

١٩١	السلاسل
١٩٢	الكتابة حرف بحرف في ملف char by char
١٩٥	القراءة من ملف حرف بحرف char by char
١٩٧	المشاكل المتوقعة عند فتح ملف
١٩٨	الكتابة والقراءة في الملف عبارة حرفية كل مرة
١٩٩	القراءة من الملف عبارة بعبارة string by string
٢٠١	التعامل مع الطابعة و الملفات الثابتة
٢٠٢	كتابة وقراءة بيانات صحيحة وحقيقية وحرفية
٢٠٤	القراءة من ملف بإستخدام fscan()
٢٠٤	الكتابة والقراءة في الملف سجل سجل record by record
٢٠٦	القراءة من ملف سجل سجل
٢٠٨	الوصول المباشر لسجل معين في ملف Random Access
٢١٠	كتابة وقراءة مجموعات من البيانات Buffer by Buffer
٢١٣	رسائل الخطأ Error Messages
٢١٤	كتابة مجموعة من البيانات Buffer by Buffer
٢١٥	إرسال المخرجات الى الطابعة
٢١٦	متى تستخدم كل طريقة من الطرق السابقة
٢١٧	حالتي الكتابة و القراءة من ملف Text Mode & Binary Mode

الفصل الحادى عشر

٢٢١	دوال مهمة للسيرنج
-----------	-------------------------

دالة تحديد احداثيات نافذة على الشاشة ٢٢٢	
نسخ جزء من الشاشة إلى موقع آخر ٢٢٤	
حفظ جزء من الشاشة في متغير ٢٢٦	
استرجاع الجزء المحفوظ ٢٢٧	
تغيير درجة اضاءة الحرف ٢٢٩	
الفصل الثاني عشر	
الرسم في لغة C ٢٣٧	
تهيئة الشاشة لحالة الرسم ٢٣٨	Initialization Graphics Mode
تحديد حالة الرسم التلقائي ٢٤١	Auto Initialization Detect
رسم الخطوط وتغيير الألوان ٢٤٣	Lines & Colors
القطع الناقص والشكل المترعرع ٢٤٥	Ellipses & Polygons
رسم شكل غير منتظم (متعرج) ٢٤٧	
التلويين والتلليل والأشكال المختلفة للتلليل ٢٤٩	
الرسم البياني ذو البعدين وثلاثي الأبعاد ٢٥١	
رسم الشكل الدائري ٢٥٣	Pie
رسم خطوط بالنسبة لآخر نقطة (الرسم النسبي) ٢٥٥	
رسم النقطة على الشاشة ٢٥٧	Pixels
تحريك الرسومات ٢٥٨	Animations
استخدام خطوط الكتابة ٢٦٠	
الفصل الثالث عشر	
دوال التعامل مع الذاكرة ٢٦٥	Memory Allocation
ملاحظات على البرنامج ٢٦٨	
الدالة () ٢٨٦	malloc
الدالة () ٢٧١	realloc
الدالة () ٢٧٤	free
الدالة () ٢٧٤	calloc
الدالة () ٢٧٦	faralloc
الفصل الرابع عشر	
روتينات الذاكرة ٢٧٩	ROM BIOS
المسجلات العامة للمعالج ٢٨٠	registeis
المسجلات ٢٨١	

٢٨١	ما هي المسجلات العامة للمعالج intel
٢٨٢	مزايا استعمال الروتينات الموجودة بالذاكرة الثابتة
٢٨٢	كيفية استدعاء الروتينات الموجودة بالذاكرة ROM BIOS
٢٨٣	الدالة () int86()
٢٨٦	استعمال ملف العناوين dos.h
٢٨٧	تغير حجم المؤشر setting the cursor size
٢٨٩	احفاء المؤشر
٢٩٠	دوال لغة C التي تستعمل ROM BIOS مباشرةً
	الفصل الخامس عشر :
٢٩١	مثال تطبيقي شامل
٢٩٢	أولاً متابعة تنفيذ البرنامج ...
٢٩٤	نص البرنامج
٣١٩	الملاحق
٣٢٠	أولاً ملحق التعامل مع البيئة المتكاملة في لغة C
٣٢٤	ثانياً ملحق الكلمات الممحورة في لغة C
٣٢٥	ثالثاً ملحق بشفرة تبادل المعلومات ASCII

تقديم

ان الحمد لله ، نحمده ونستعينه ونستهديه ، ونصلى ونسلم على سيدنا
محمد صلى الله عليه وسلم وآلـه وصحبه أجمعين .

﴿سبحانك لا علم لنا الا ما علمنا ، انك أنت العليم الحكيم﴾ ... وبعد

بفضل من الله كان النجاح الذى حققته سلسلة تيسير علوم الحاسوب
وبقدر الجهد الذى بذل ، كان النجاح الذى تحقق . وكانت سعادتى غامرة
بالرضا الذى استقبل به القراء كتبنا . لقد كان التشجيع من الجميع دافعا
وحافراً كما كان النقد البناء واللاحظات القيمة من الزملاء ومن كل من
قرأ كتب سلسلتنا - والذى عبرت عنه رسائلهم إلينا - كنزاً نعتر به

وهذا الكتاب اضافة جديدة إلى سلسلة تيسير علوم الحاسوب التى نشرف
باصدارها . والكتاب يشتمل على معلومات وافية وغزيرة تحرر فى
المؤلف الدقة التامة ، وعلى كم هائل من البرامج الهدف منها تحقيق
أقصى فائدة ولتكون بمثابة نماذج يقتدى بها فى اعداد برامج مماثلة فى
حياتك العملية .

ان طموحاتنا كبيرة وآمالنا عظيمة وثقتنا بالله بغير حدود. ولذلك فاننا
نناشد كل صاحب قلم أو فكر من المهتمين والمتخصصين في مجال
الحاسب أن يضع يده في أيدينا لكي نخرج أفكاره ومؤلفاته إلى النور
ونساهم في سد العجز الهائل في هذا المجال .

مجلد محمد أبو العطا

مقدمة

اذا كنت قد تعودت على أن تعامل مع برنامج نوافذى تشعر معه أنك مقيد
الحركات، جامد الإبداع فسيتحول هذا الشعور إلى الأفضل من خلال دراسة أقوى لغات
البرمجة للمحترفين على الإطلاق.

لقد حاولنا في هذا الكتاب أن نزيل الحاجز النفسي بين عامة المستخدمين
ولغات البرمجة وخاصة لغة C، ولا نطلب منك إلا أن تعطينا الفرصة لإتمام ذلك
وتشاركنا عملية تحويل المستخدم إلى مبرمج يصبح غيره مستخدماً لما ينتجه.
فإن تكون منشئ الشيء ، لا من يستخدمه ... هذا ما نرتضيه لك
أن تكتشف أعني ملكاتك ... هذا ما سنساعدك عليه
أن تقف موقف الملك في ساحة العطاء... هذا ما نصبو إليه

م/عزب محمد عزب

الفصل الأول

نظرة عامة

نتناول في هذا الفصل نظرة عامة تشمل التعريف بلغة C وموقعها بين لغات البرمجة الأخرى ومميزاتها. وكيفية كتابة برنامج بلغة C لأول مرة وأنواع البيانات والمؤثرات التي تستخدمها.

النهاء هذا الفصل سترى على :

- ♦ مقدمة تاريخية عن لغة C.
- ♦ مميزات لغة C.
- ♦ فتح ملف البرنامج.
- ♦ خطوات كتابة وتنفيذ برنامج بلغة C.
- ♦ قواعد كتابة البرنامج.
- ♦ شرح مكونات أول برنامج.
- ♦ أنواع البيانات.
- ♦ أنواع المؤثرات (Operators).

في عام ١٩٦٠ صممت لغة Combined Programming CPL في جامعة لندن وقد استخدمت فيها بعض تعليمات لغة ALGOL-60 ، وفي عام ١٩٦٧ ابتكرت لغة BCPL من لغة CPL على يد مارتن ريتشاردز ، ثم قام ثومبسون بتطوير BCPL وسماها لغة B وكل هذه اللغات تعتبر من لغات المستوى الأدنى (Low Level Languages) أي القريب من لغة الآلة مثل لغة التجميع (Assembly).

وفي عام ١٩٧٢ وفي معامل شركة AT&T الأمريكية قام ريتشاردز بإنشاء لغة جديدة من لغة B أخذ منها أحسن تعليماتها وأضاف إليها أوامرًا جديدة وأنواعًا جديدة للبيانات وكثيراً من الدوال التي تفيد البرمجة وسميت هذه اللغة بلغة C. ومنذ ذلك التاريخ أخذت لغة C شهرة واسعة لأنها أصبحت تنتهي ل اللغات المستوى الأعلى (High Level Languages) مثل لغة البيسيك والباسكال من حيث سهولة الاستخدام من ناحية ومن ناحية أخرى تنتهي إلى لغات المستوى الأدنى من حيث قدرة اللغة على مخاطبة مكونات الجهاز (Hardware).

ومع نجاح وانتشار لغة C انتشرت لهجات كثيرة للمتحدثين بلغة C وكذلك يحدث معها ما حدث للغة بيسيك من وجود بيسيك خاص لكل نوع من أنواع الأجهزة سواء كانت أجهزة أي بي أم أو أبل أو كومودور أو سكلير لولا أن قام معهد القياسات الأمريكية بعملية توحيد لهذه اللهجات المختلفة فأصدر في عام ١٩٨٣ اللغة القياسية ANSI C ، والجدير بالذكر أن معظم مترجمات لغة C تتوافق مع ANSI C مثل Borland C, Microsoft C, Lattice C.

مميزات لغة C

نورد فيما يلى أهم ما يميز لغة C عن غيرها من لغات البرمجة وهي المميزات التي تدعوك لفضيلتها على غيرها من لغات البرمجة المعروفة وترغبك في تعلمها واستخدامها.

تتميز لغة C بمجموعة من المزايا مثل:

لغة عامة

أى تصلح لعمل برامج قواعد البيانات والرسومات والحسابات ونظم التشغيل

وغيرها

لغة تركيبية (*Structured Language*)

البرنامج المكتوب بلغة C عبارة عن دالة رئيسية تتضمن مجموعة من الدوال الأخرى وكل دالة مجموعة من الأوامر.

تعامل على مستوى "البت" (*Bit Manipulation*)

حيث تستطيع أن تقرأ وتكتب وتغير وتقوم بعمليات على مستوى الـ Bit وكما هو معروف فإن الـ Bit هي أصغر وحدة لقياس المعلومات داخل الكمبيوتر وهي جزء من ثمانية أجزاء تعادل في مجموعها حرف واحد (Byte).

وهذه الميزة جعلتها متخصصة في بعض مجالات التحكم الآلي والروبوت وبرامج الـ Utility وبرامج معالجة الصور وضغط الملفات واكتشاف الأعطال.

لغة متنقلة (*Portable*)

أى يمكن للبرنامج المكتوب بلغة C أن يعمل مع أكثر من جهاز مثل IBM أو الأجهزة المتوسطة والكبيرة مع بعض التعديلات الطفيفة.

لغة سريعة

لأن أدوات اللغة تتعامل مباشرة مع الآلة مما يختصر وقت التنفيذ.

لغة قياسية

معظم مترجمات اللغة تتوافق مع اللغة القياسية C ANSI.

لغة نظام التشغيل Unix

مما يدل على ثراء هذه اللغة وموتها أن نظام التشغيل المشهور UNIX مكتوب بها.

تثبيت قرص التمارين المرفق بالكتاب

تجد مع هذا الكتاب قرص مرفق باسم : المراجع الأساسية للغة C .. التمارين العملية. يشتمل القرص على التمارين العملية الموجودة بالكتاب والتي يعتمد عليها الشرح الوارد بالكتاب. ويفضل نسخ التمارين الموجودة على القرص إلى القرص الصلب قبل أن تشرع في دراسة هذا الكتاب.

فيما يلى خطوات تثبيت القرص المرفق مع الكتاب :

١. قم بعمل نسخة احتياطية من هذا القرص باستخدام أمر DISKCOPY ثم احفظ النسخة الأصلية من القرص في مكان آمن.
٢. ضع القرص المنسوخ في المشغل A أو B حسب ما هو متوفّر في جهازك
٣. أكتب من مشغل القرص الموجود به الاسطوانة الامر Install بالصورة التالية :

A:> INSTALL

سيقوم برنامج التثبيت(INSTALL) بنسخ محتويات القرص (جميع أمثلة الكتاب) الى القرص الصلب تحت الدليل C_BOOK ويقف المبحث داخل هذا الدليل

٤. أكتب الامر DIR

ستجد أدلة فرعية بأسماء فصول الكتاب وكل دليل فرعى يحتوى على تمارين الفصل فمثلا الملف CS_1-1.C معناه المثال الاول فى الفصل الاول حيث CS اختصارا لاسم الشركة Compu Science و 1-1 تعنى الشكل الأول من الفصل الأول.

تشغيل البرنامج لأول مرة

نفترض ان لديك جهاز مناسب للعمل مع لغة C كما أشرنا من قبل وأنك قمت بتشييت قرص التمارين على القرص الصلب (Hard Disk) وأيضا قمت بتشييت أحد البرامج المترجمة للغة C (C Compiler) ولتكن Turbo C أو Turbo C++ أو Turbo C# أو Turbo Pascal تشغيل البيئة المتكاملة IDE

بفرض أن الدليل الخاص ب Turbo C أو Turbo C++ هو tc تابع الخطوات

التالية :

١ . اكتب الأمر التالي ثم إضغط مفتاح الإدخال

cd \ tc

يظهر المحتوى بهذا الشكل

c:\tc>

٢ . اكتب الأمر التالي ثم إضغط مفتاح الإدخال

cd bin

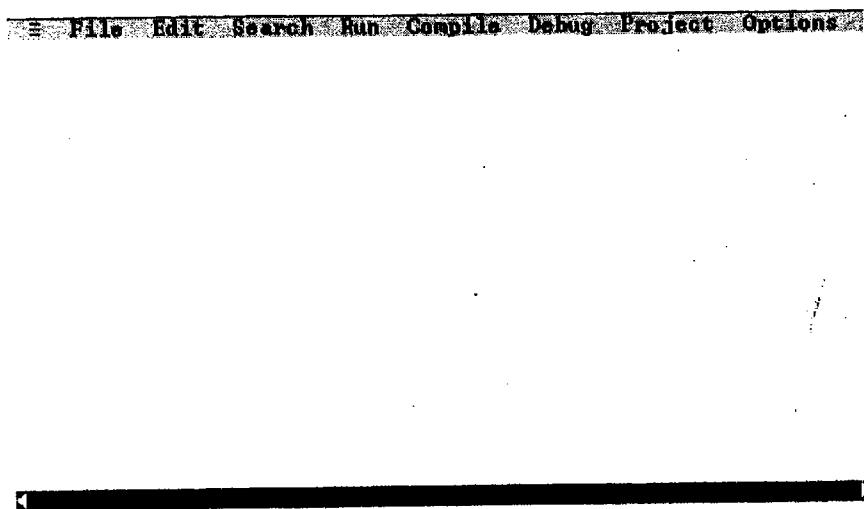
يظهر المحتوى بهذا الشكل

c:\tc\bin>

٣ . من محت DOS أكتب الأمر التالي ثم إضغط مفتاح الإدخال

c:\tc\bin>tc

تظهر الشاشة الخاصة بتحرير البرامج كما في شكل (١-١)



شكل (١-١) شاشة تحرير البرامج

تستخدم هذه الشاشة لكتابة البرامج وتعديلها وترجمتها وإكتشاف الأخطاء وأيضاً تنفيذها.

وتكون الشاشة من مجموعة من القوائم المنسدلة الخاصة بالملفات (File) وتحرير السطور (Edit) والبحث عن الكلمات وإستبدالها (Search) وترجمة البرامج (Compile) وتنفيذها (Run) والبحث عن الأخطاء وتصحيحها (Debug) وبعض الخيارات الأخرى (Options).

لتشييط شريط القوائم اضغط مفتاح Alt ولفتح قائمة اضغط مفتاح Alt مع أول حرف من إسم القائمة مثلاً لفتح قائمة File اضغط Alt+F.

راجع الملحق (ب) للحصول على تفصيلات عن كيفية التعامل مع برنامج

Turbo C.



للخروج من البرنامج ثانية إلى ممح DOS

٣. اضغط Alt+F تظهر قائمة File
٤. اختر Exit أو اضغط المفاتيح Alt+X معاً

فتح ملف البرنامج

إذا كانت لك خبرة بلغة بيسيك فإنك تعرف أنه بإمكانك أن تكتب برنامج وتنفذه في أول ساعة وستثبت لك أنك تستطيع أن تفعل ذلك مع لغة C أيضاً.

افتح بيئه تشغيل البرامج من جديد بالأمر TC بالصورة التالية :

c:\tc\bin>tc

افتح ملف البرنامج من دليل التمارين بإتباع الخطوات الآتية :

١. اضغط Alt+F تظهر قائمة أوامر التعامل مع الملفات
٢. اختر الأمر Open يظهر مربع حوار فتح الملفات (شكل ١-٢)
في خانة Name اكتب إسم الملف cs1_1 إسم الدليل كالتالي :



شكل ١-٢

اضغط Enter .

يظهر البرنامج كما في الشكل ١-٣

```
/* Program Name : CS1_3.C */
```

```
/* This program prints the message wellcome with CompuScience */
```

```
#include <stdio.h>
```

```
main()
```

```
{  
    printf ("Welcome With CompuScience "); /*output function */  
}
```

شكل رقم ١-٣ شكل البرنامج بعد كتابته

والأفضل أن تكتب البرنامج للتعرف على أوامر اللغة والشعور بالألفة معها.
قبل أن نشرح البرنامج والتعليمات التي يشتمل عليها سنشرح قواعد كتابة
البرنامج وخطوات تنفيذه بصفة عامة وعليك أن تطبقها على البرنامج الذي بين
أيدينا.

قواعد بناء البرنامج

- البرنامج cs1_1.c الموجود بشكل ١-٣ يمثل أبسط تركيب لبرنامج مكتوب بلغة C ويخصّص للقواعد الآتية :
- يبدأ البرنامج بالعبارة `#include <.....h>` وبين العلامتين إسم ملف التوجيه الخاص بالدوال المستخدمة في البرنامج (سيأتي ذكر ملف التوجيه الخاص بكل دالة عند شرحها)
 - يتكون البرنامج من دالة رئيسية main() وتبعد بالقوس { وتنتهي بالقوس }
 - جميع كلمات ودوال اللغة تكتب بالحروف الصغيرة
 - تنتهي كل عبارة بفواصلة منقطة (;) مع وجود استثناءات سوف تعرفها فيما بعد
 - يجوز كتابة أي ملاحظات أو تعليقات خاصة بالمبرمج بوضعها بين العلامتين /* لأى عدد من السطور.

التعامل مع البرنامج

فيما يلى نوضح كيفية التعامل مع البرنامج :

- لحفظ ملف البرنامج اضغط F2 أو إختر Save من قائمة File
- لترجمة وتنفيذ البرنامج اضغط على المفاتيح Ctrl+F9 أو ALT+R ثم على الاختيار الأول من القائمة نضغط مفتاح الادخال
- لرؤية نتيجة التنفيذ اضغط على المفاتيح Alt+F5 تظهر النتيجة كما يلى :

Welcome With CompuScience

- للرجوع الى شاشة الكتابة اضغط أى مفتاح

خطوات التنفيذ

لاحظ أن البرنامج الذى تم تنفيذه هو cs1_1.exe وليس cs1_1.c حيث تمت ترجمة البرنامج المصدر C. وتحويله الى ملف قابل للتنفيذ (executable) على الخطوات الآتية :

- ضم ملف (ملفات) التوجيه المعلن عنه بالعبارة #include <stdio.h> إلى الملف المصدر cs1_1.c
- ترجمة الكود بعد الضم إلى لغة الآلة ليعطى ملف له الامتداد .obj.
- ضم ملفات مكتبات الدوال إلى الملف الناتج وربطهم بواسطة برنامج Linker في ملف واحد وهو cs1_1.exe والملف s1_1.exe لا يحتاج لبيئة تشغيل خاصة بل مثل أى برنامج تحت محث DOS ولتشغيل البرنامج أخرج من برنامج Turbo C بالضغط على Alt+x واكتب إسم البرنامج بالصورة التالية :
c:\tc\bin>cs1_1

والآن نعود لشرح البرنامج الموجود بالشكل ١-٣

شرح البرنامج

- السطورة رقم ١ ، ٢ ، ٣ عبارة عن تعليقات (comments) لا تؤثر في تنفيذ البرنامج ويتم ذلك بوضع /* في بداية التعليقات ووضع */ في نهاية التعليقات مهما كان عدد سطور التعليقات.
- السطور رقم ٤ هو #include <stdio.h> فيه الكلمة #include بمعنى اشتمل وكلمة stdio.h هي ملف اسمه stdio وإمتداده h هذا الملف موجود مع حزمة برنامج لغة C وذلك في الدليل INCLUDE والسطر كلها معناه حمل (أضف) محتويات هذا الملف إلى البرنامج) الملف stdio.h لأن به تعاريفات الدوال التي سوف تستخدم في البرنامج ومن هنا نفهم أن كل مجموعة دوال من دوال لغة C لها ملف مثل الملف stdio.h يجب كتابته في أول البرنامج وسوف نتعرض لهذه النقطة بالتفصيل في موضع آخر.
- السطور رقم ٥ يحتوى على عبارة main() هذه العبارة هي الدالة الرئيسية للبرنامج حيث أن البرنامج سوف يتربّك من مجموعة دوال ولكن الدالة main() هي الدالة الرئيسية التي تناول باقي الدوال.
- السطور رقم ٦ هو بداية الدالة main() و تبدأ بالقوس { وتنتهي الدالة الرئيسية في السطور رقم ٨ بالقوس } (يعنى أنها نفتح القوس) في أول الدالة الرئيسية ونهاي الدالة الرئيسية بالقوس }
- ويشتمل السطور رقم ٧ على العبارة printf ("Welcome With CompuScience"); وفيها الدالة printf() وهي دالة الالخراج الرئيسية على الشاشة وسوف نناقشها بالتفصيل في الفصل الثاني وهنا تطبع عبارة على الشاشة.

- السطر رقم ٨ والأخير ويشتمل على القوس { } ومعناه نهاية الدالة الرئيسية .main()

أنواع البيانات

كما تعرف أن البيانات التي نتعامل معها إما أرقام أو حروف أو كلمات والارقام يمكن أن تكون صحيحة (أى ليس بها علامة عشرية) أو حقيقة أى بها علامة عشرية.

والحروف يمكن أن تكون حرف واحد أو أكثر من حرف وهكذا تختلف أنواع البيانات عن بعضها البعض و من الضروري معرفة أنواع البيانات ومعرفة كيفية الإعلان عنها وكذلك كيفية استعمالها.

والجدول الآتي يوضح هذه الانواع وكذلك عدد البايت (Byte) التي يشغلها كل نوع

نوع المتغير	المدى المسموح	طوله بالبايت
حرف (char)	ـ حرف أو رمز واحد	ـ ١
صحيح قصير (int)	ـ ٣٢٧٦٨ إلى ٣٢٧٦٨	ـ ٢
صحيح طويل (long)	ـ ٢٠١٤٧٠٤٨٣٠٦٤٨ إلى ٢٠١٤٧٠٤٨٣٠٦٤٨	ـ ٤
حقيقي (float)	ـ E+38 إلى E-38	ـ ٤
حقيقي مضاعف (double)	ـ E+308 إلى E-308	ـ ٨

ولووضح فيما يلى المقصود بكل هذه الانواع :

- متغير من نوع حرف أى متغير يصلح لتخزين حرف فقط
- متغير من نوع صحيح أى متغير يصلح لتخزين رقم صحيح (ليس به علامة عشرية مثل ٥٤٤ ، ٥٧ ، ٥) .

- متغير من نوع صحيح ولكن طويل (long) اي يستطيع ان يخزن رقم صحيح ضعف المتغير الصحيح العادي ونستعمل هذا النوع اذا كانت الارقام التي نتعامل معها أكبر من المساحة المخصصة للرقم الصحيح العادي والا سنحصل على نتائج خاطئة بالرغم من ان البرنامج سليم .
- متغير حقيقي اي متغير يصلح لتخزين رقم حقيقي (يقبل الكسور العشرية مثل (٣٣,٤ ٤٤,٥ ٣,٣)
- متغير حقيقي مضاعف اي يستطيع ان يخزن رقم حقيقي ضعف المتغير الحقيقي العادي

تسمية المتغير

- يخضع اسم المتغير لشروط معينة يجب أن تعرفها تجنبًا لأخطاء قد تقع فيها وفيما يلى نوضح هذه الشروط :
- يجب أن يبدأ المتغير بحرف ثم يكمل المتغير بعد ذلك حروف أو أرقام ويجب ألا يحتوى على علامة خاصة سوى الشرطة التحتية (_).
- من الممكن أن يشتمل إسم المتغير حتى ٣٢ حرفاً وما زاد عن ذلك لا يلتفت إليه مترجم اللغة .
- يفرق المترجم بين الحروف الصغيرة و الكبيرة فالمتغير St يختلف عن المتغير st فإذا استعملنا في البرنامج يعتبرهما البرنامج متغيرين
- يجب ألا يكون المتغير باسم كلمة من الكلمات المحجوزة في اللغة مثل .int,return

الإعلان عن المتغيرات

اذا كنت تستخدم مترجم لغة C/C++ فيمكن يتم الإعلان عن المتغيرات في أي مكان بالبرنامج ولكن بشرط ان تكون قبل العبارات التي تستخدم هذا المتغير أما اذا

كنت تستخدم مترجم لغة C فقط فيجب أن يكون الإعلان في أول البرنامج لتفادي الأخطاء.

مثال :

```
int a;  
float b;
```

المؤثرات Operators

المؤثرات هي الرموز التي تربط بين المتغيرات والثوابت لإنشاء علاقة ما أو معادلة تختلف أنواع المؤثرات ب اختلاف وظيفه كل مؤثر. وسأحد الأنواع الآتية

Arithmatic Operators

وهي علامات الجمع (+) والطرح (-) والقسمة (/) والضرب (*) وتستخدم مع المتغيرات والثوابت الرقمية

مؤثرات المقارنة (Relational Operators)

وتستخدم لمقارنة قيمتين لمعرفة هل هما متساويتين أو أحدهما أكبر أو أقل من الأخرى وهكذا . ويوضح الجدول التالي مؤثرات المقارنة والرموز التي تستخدم بدلا عنها.

المؤثر	الرمز	مثال	النتيجة
أكبر من	>	10>8	1
صغر من	<	10<8	0
يساوي	==	10 == 8	0
لا يساوي	!=	10 != 8	1
أقل من أو يساوي	<=	10 <= 8	0
أكبر من أو يساوي	>=	10 >= 8	1

المؤثرات المنطقية Logical Operators

وتوستخدم لتحديد شرط مركب مثل الشرط التالي if($a==b \&& c==d$)

ويعني إذا كانت قيمة المتغير A تساوى قيمة المتغير B وفي نفس الوقت قيمة المتغير C تساوى قيمة المتغير D.

ويوضح الجدول التالي هذه المؤثرات والرموز التي تستخدم بدلا منها :

المؤثر	الرمز	مثال	النتيجة
و AND	&&	$10>8 \&\& 9>7$	1
أو OR		$10<8 7<8$	1
لا NOT	!	$!(10 == 8)$	1

مؤثرات التخصيص Assignment Operators

وهي مؤثرات تخزين قيمة في متغير مثل $=, +=, -=, *=, /=$

وتوستخدم لتخزين قيمة في متغير بالإعتماد على القيمة الموجودة في نفس المتغير فمثلا إذا قمت ب تخزين القيمة 6 في المتغير a باستخدام الأمر $a=6$ وأردت مضاعفة القيمة المخزنة يجب أن تكتب الأمر $a=a*2$ بهذا تصبح قيمة a تساوى 12 ولزيادة قيمة المتغير 1 $a=a+1$ وهكذا وهذه الطريقة تستخدم في جميع لغات البرمجة وتميز لغة C بوجود طريقة بجانب الطريقة السابقة موضحة بالجدول الآتي بفرض أن $a=6$

التخصيص القليدي	الطريقة الحديثة	النتيجة
$a=a+5$	$a+=5$	11
$a=a-5$	$a-=5$	1
$a=a*5$	$a*=5$	30

الفصل الأول : نظرة عامة

النتيجة	الطريقة العديدة	التخصيص التقليدي
2	a/=3	a=a/3
7	a++	a=a+1
5	a--	a=a-1

هناك فرق بين المؤثر = والمؤثر == حيث أن المؤثر = يستخدم كما سبق في الحساب قيمة متغير أما المؤثر == يستخدم للمقارنة مثل if معناها اذا كان a يساوى b



مؤثر باقى خارج القسمة %

يستخدم لمعرفة باقى القسمة (و تستطيع أن تحدد هل الأرقام الموجودة في متغير ما زوجية أم فردية) فمثلا اذا كانت A = 5

$$C = A \% 2$$

فإن C ستكونقيمتها 1 وهو باقى قسمة الرقم 5 على 2

مؤثران الزيادة والنقصان Increment & Decrement من المعروف أن التعبر

A = A + 1 معناه زيادة قيمة المتغير A بمقدار واحد والتعبير A - 1 معناه إنقصاص

قيمة المتغير A بمقدار واحد ولكن توجد صورة آخرى مسموح بها لهتان العمليتان وهى

$$A = A + 1 \quad A = A - 1 \quad \text{وتقابيل} \quad A++ \quad \text{وتقابيل} \quad A--$$



الفصل الثاني

دوال الإدخال وال 출력

في هذا الفصل سوف نشرح دوال الإدخال والإخراج بأنواعها المختلفة والتي تتيح لك تطوير برامج أكثر فائدة وهي:

- ♦ دالة الطباعة على الشاشة `(printf)`
- ♦ دالة استقبال قيم من لوحة المفاتيح `(scanf)`
- ♦ دوال إدخال حرف واحد `(getchar(), getche())`
- ♦ دالة إظهار حرف واحد `(putchar()` و دالة `getch()`
- ♦ دالة طباعة و إدخال عبارة حرفية `(puts(), gets())`
- ♦ دالة مسح الشاشة `(clrscr())`
- ♦ دالة تغيير موضع المؤشر `(gotoxy())`
- ♦ دالة تغيير الألوان `(textcolor(), textbackground())`
- ♦ دالة الطباعة والإدخال باستخدام الألوان `(cprintf(), cscanf())`
- ♦ دوال التعامل مع الطابعة `(fprintf(), fputs(), fputc(), fwrite())`

شرحنا في الفصل الأول أن كل دالة مرتبطة بملف توجيه معين حيث يستدعي هذا الملف في أول البرنامج العبارة `#include` فمثلا الدالة `printf()` معرفة بالملف `stdio.h` وتكتب العبارة `#include <stdio.h>` في أول البرنامج حتى يعرف المترجم على الدالة وهكذا مع باقي الدوال. في هذا الفصل سنشير إلى ملف التوجيه المعرفة به كل دالة في بداية شرح الدالة.

دالة الطباعة على الشاشة (`printf()`)

ملف التوجيه : `stdio.h`

تستخدم الدالة `printf()` لطباعة البيانات بجميع أنواعها `int, char, string`) على الشاشة `float, ...`.

وتأخذ الدالة عدة صور وكذلك عواملات وأكسناد تحدد شكل المخرجات.

وسنوضح فيما يلي مثال لكل صورة مع الشرح

مثال :

```
printf ("Welcome with CompuScience")
```

وفي هذا المثال يتم طباعة ما بين علامتي التنصيص " " كما هو على الشاشة وبالتالي نحصل على النتيجة التالية :

Welcome with CompuScience

مثال

```
printf ("\n Welcome \nwith \n CompuScience ")
```

وفي هذا المثال : الكود `\n` معناه new line أي سطر جديد وعندما يوجد المترجم `\n` يترجمها إلى سطر جديد وبالتالي نحصل على النتيجة التالية :

Welcome
With
CompuScience

وهنالك أكواد أخرى تؤدي نتائج مختلفة فمثل الكود \t معناه tab أي مسافة جدولية خالية ويشمل الجدول رقم (٢-١) على الأكواد المستخدمة مع الدالة printf() والتي تؤدي أشكال خرج مختلفة

الكود	الاستخدام	مثال
\n	الانتقال لسطر جديد	printf("\n")
\t	نقل المؤشر بعد ٨ مسافات	printf("\t")
\a	إخراج صوت الصافرة (بيب)	printf("\a")
\b	إرجاع المؤشر مسافة خلفية	printf("\b")
\xdd	طباعة الحرف المناظر للكود المكتوب بالنظام السادس عشر . hexadecimal	printf("\41") النتيجة : A
\ddd	طباعة الحرف المناظر للكود المكتوب بالنظام الشماني octal	printf("\101") النتيجة : A
\ \	طباعة الشرطة المائلة	printf("\\")
\?	طباعة علامة الاستفهام	printf("\?")
\''	طباعة العلامة (')	printf("\'")
\\"	طباعة علامة التنصيص	printf("\\"")

الجدول رقم (٢-١) الأكواد المستخدمة مع دالة printf()

ويوضح البرنامج الموجود في شكل (٢-١) طريقة استخدام هذه الأكواد

```
0: /*Program Name cs2_1.c */  
1: #include <stdio.h>
```

```

2: main()
3: {
4:     printf("\n this text display in new line");
5:     printf("\n word1\t tab1 \t tab2 \t tab3.....");
6:     printf("\n Bell \a Bell \a Bell \b");
7:     printf("\n \" this line display Quotaions \"");
8:     printf("\n\xc9\xcd\xbb\n");
9:     printf("\xc8\xcd\xbc\n");
10: }

```

شكل ٢-١ برنامج استخدام أكواد دالة (printf)

وعن هذا البرنامج نوضح ما يلى :

بداية من هذا البرنامج سنضع رقم لكل سطر من سطور البرنامج للإرشاد
برقم السطر في الشرح فقط مع مراعاة أن هذه الأرقام لا تكتب في البرنامج.

- في السطر رقم ٤ الكود \n تعنى سطر جديد
- في السطر رقم ٥ الكود \t تعنى مسافة tab أي مسافة جدوله
- في السطر رقم ٦ \a تعنى نغمة الصافرة (بيب) وهكذا
- وعند تنفيذ هذا البرنامج ستحصل على النتيجة الموجودة بشكل ٢-٢

this text display in newline

word1 tab1 tab2 tab3

Bell Bell Bell

" this line display Quotaions "



شكل ٢-٢ نتائج مختلفة لدالة (printf)

طباعة قيم المتغيرات على الشاشة

طباعة القيم الموجودة بالمتغيرات نستخدم أكواد معينة لتحديد نوع البيانات المراد طباعتها بالدالة `printf()` ، انظر المثال التالي :

1: `printf("%d",a);`
2: `printf("%f",b);`

في هذا المثال عندما يقابل مترجم اللغة العلامة `%` ينظر إلى الحرف التالي لهذه العلامة ويعتبر هذا الحرف توصيف لقيمة موجودة بعد العلامة ، وكل حرف يحدد نوع معين من البيانات ففي هذا المثال نلاحظ :

int	أى رقم صحيح	%d
float	أى رقم حقيقي	%f

وعلى ذلك فإن السطر رقم 1 معناه اطبع القيمة الصحيحة الموجودة بالمتغير `a` و السطر رقم 2 معناه اطبع القيمة الحقيقة الموجودة بالمتغير `b` وهكذا

ويوضح جدول ٢-٢ أكواد طباعة أنواع البيانات الأخرى :

مثال	الاستخدام	الكود
<code>printf("%d",-10)</code>	توصيف لمتغير(أو ثابت) رقمي صحيح	%d
<code>printf("%f",5.7)</code>	توصيف لمتغير(أو ثابت) رقمي حقيقي	%f
<code>printf("%c",'a')</code>	توصيف لمتغير (أو ثابت) char (حرف واحد)	%c
<code>printf("%s","ab")</code>	توصيف لعبارة حرفية string (حرف أو أكثر)	%s

الكود	الاستخدام	مثال
%u	توصيف لمتغير (أو ثابت) رقمي صحيح بدون إشارة	printf("%u",10)
%x	توصيف لمتغير (أو ثابت) بالنظام السادس عشر hex	printf("%x",af)
%o	توصيف لمتغير (أو ثابت) بالنظام الشمسي octal	printf("%o",67)

جدول (٢-٢) أكواد طباعة البيانات

يشتمل البرنامج الموجود في شكل ٢-٣ على مثال جامع يوضح معظم الأكواد المستخدمة ، وعند تفريذ البرنامج ستحصل على النتيجة الموضحة بشكل ٢-٤

```

0: /*Program Name : cs2_3.c */
1: #include <stdio.h>
2: main()
3: {
4:     int A=5,B=10,C;
5:     float F=45.5;
6:     char ch='Y';
7:     char name[10]="Ahmed";
8:     long t=32142134;
9:     A=5;
10:    B=10;
11:    C=A+B;
12:    printf("\n the Int C=%d",C);
13:    printf("\n the float F=%f",F);
14:    printf("\n the char ch=%c",ch);
15:    printf("\n the string name=%s",name);

```

```
16:     printf("\n the long no t=%ld",t);  
17: }
```

شكل ٣-٢ مثال لاستخدام أكواد الطباعة

C = 15
F = 45.5
ch = Y
Name = Ahmed
t = 32142134

(شكل ٤ - ٢) نتيجة تنفيذ برنامج أ��واد الطباعة

وعن هذا البرنامج نوضح ما يلي :

- في هذا البرنامج يشتمل السطر رقم ١ على الجملة #include<stdio.h> وستستخدم لتحميل ملف التوجيه stdio.h (الاضافة سطور الملف stdio.h الى ملف البرنامج CS2_2.C) الذي يحتوى على تعريف الدالة printf() وفي السطر رقم ٢ الدالة الرئيسية main() وفي السطر رقم ٣ تبدأ الدالة الرئيسية int بالقوس { ثم في السطر رقم ٤ اعلان عن ثلاث متغيرات من نوع صحيح وفي السطر رقم ٥ اعلان عن المتغير F من نوع حقيقي (float) واعطائه قيمة الابتدائية 45.5 . في السطر رقم ٦ اعلان عن المتغير ch من نوع حرف (char) واعطائه القيمة الابتدائية حرف Z . في السطر رقم ٧ اعلان عن المتغير name لتخزين عبارة حرفية واعطائه القيمة الابتدائية كلمة Ahmed . السطر رقم ٨ اعلان عن المتغير t (صحيح طويل) واعطائه القيمة الابتدائية 32142134 ونلاحظ أن المتغير t أخذ النوع long لأن القيمة التي خزنت فيه كبيرة .

- السطورة رقم ٩ ، ١٠ ، ١١ لاعطاء قيم للمتغيرات A,B وقيمة C
- في السطر رقم ١٢ الكود %d, يعني اطبع قيمة المتغير الذي يلى العلامة .
- وحرف d تعنى أنه صحيح وبالتالي يتم طباعة القيمة الموجودة بالمتغير C
- في السطور رقم ١٣ ، ١٤ ، ١٥ ، ١٦ تطبع قيم المتغيرات F,ch,name,t و تنتهي الدالة الرئيسية بالقوس } وبالتالي ينتهي البرنامج

ملاحظات

يمكن أن تستخدم الأكواد %f و %d لتحديد عدد الأرقام التي تظهر على الشاشة فمثلاً الصورة ٣.٣f يعني طباعة ثلاث أرقام بعد العلامة العشرية فمثلاً الرقم ٥٣٤,٥٦٣٧٨ يظهر بالصورة ٥٣٤,٥٦٣٧٨

لاحظ أننا شرحنا هذا البرنامج بالتفصيل لأنه أول برنامج ولكن في الأمثلة التالية سوف نشرح المفاهيم أو الأوامر الجديدة في البرنامج فقط.

دالة الإدخال العامة (scanf)

هي دالة الإدخال الرئيسية التي تسمح بادخال جميع أنواع البيانات وهي تأخذ نفس المعاملات التي تأخذها الدالة printf() للتعامل مع البيانات و الموجودة بالجدول رقم (٢-٢).

والصورة العامة للدالة scanf() هي

```
int scanf (const char *format [, address, ...]);
```

والمثال الموجود بالشكل ٢-٥ يوضح استخدام الدالة scanf() حيث يقوم باستقبال مجموعة قيم مختلفة النوع وطباعتها على الشاشة

```
0: /* Program Name : cs2_5c */  
1: #include<stdio.h>  
2: main()
```

```
3:  {
4:      int A,B,C;
5:      float R,S,T;
6:      char name[10];
7:      printf("\n\n Enter your name:");
8:      scanf("%s",name);
9:      printf("A=");
10:     scanf("%d",&A);
11:     printf("B=");
12:     scanf("%d",&B);
13:     printf("R=");
14:     scanf("%f",&R);
15:     printf("S=");
16:     scanf("%f",&S);
17:     printf("\n WELCOME %s",name);
18:     printf("\n\n C=A+B=%d",A+B);
19:     printf("\n\n T=R+S=%f",R+S);
20: }
```

شكل ٢-٥ استخدام الدالة ()

وعن هذا البرنامج نوضح ما يلى:

- في السطور رقم ٤ ، ٥ ، ٦ اعلان عن المتغيرات A,B,C ,R,S,T ,NAME
- في السطر رقم ٧ تطبع الدالة printf() الرسالة Enter your name:
- في السطر رقم ٨ تستقبل الدالة () العباره الحرفية التي يدخلها المستخدم وتضعها في المتغير name.
- السطور من ٩ الى ١٦ تستخدم نفس فكرة السطرين ٧ و ٨ في استقبال قيم المتغيرات
- وتلاحظ في السطر رقم ١٠ أن الدالة () scanf() تستقبل قيمة صحيحة وتخزنها في المتغير A ولكن ماذا يعني المؤثر & ؟

A & A تعنى تخزين القيمة الصحيحة في المكان المخزن عنوانه في المتغير A بمعنى أن A يشير إلى عنوان المكان الذى تخزن فيه القيمة. وبالتالي العلامة & تجعل المتغير يشير إلى عنوان المكان

Enter your name : Ahmed

A = 5

B = 10

R = 20

S = 30

Welcome Ahmed

C = A+b = 15

T = R + S = 50

شكل ٢-٦ نتائج استخدام الدالة (scanf)

وفي هذا المستوى من الدراسة يكفى أن نعرف هذا القدر عن المؤثر & ولكن لنا مع هذا المؤثر & وقفه أخرى في درس المؤشرات. مع ملاحظة أن المؤشر & يوضع فقط مع البيانات الصحيحة والحقيقة (Int, float) والحرف ولا يوضع مع متغير العبارة الحرفية string. ولزيادة توضيح الدالة () Scanf.

انظر المثال التالي وتتابع شرحه

```
scanf("%d,%f,%c",&a,&b,&ch);
```

هذا السطر معناه تخزين الرقم الصحيح %d في أول متغير يلى الفاصلة وهو المتغير a ثم الرقم الحقيقي %f في المتغير الذى يليه ، وكذلك الحرف في المتغير ch مع ملاحظة أن الصورة %d,%f,%c تعنى أن المدخلات لابد ان تكون بنفس الصورة. أي بفاصيل ، بين كل قيمة مدخلة.

وعند تنفيذ البرنامج ستحصل على النتيجة الموجودة بشكل ٢-٦

دوال إدخال حرف واحد `getchar()`,`getche()`,`getch()`

بالرغم من أن الدالة `scanf()` تستقبل جميع أنواع البيانات إلا ان لغة C تشتمل على دوال اخرى تتعامل مع أنواع خاصة من البيانات كالحروف و العبارات الحرفية ونوضح فيما يلى أهم هذه الدوال.

الدالة `getchar()`

ملف التوجيه : `stdio.h`

تستخدم لإدخال حرف واحد ويظهر الحرف على الشاشة بعد الكتابة ولا تسمح بالانتقال الى الامر التالي الا اذا ضغط المستخدم مفتاح الادخال (`Enter`) و هذا يشبه ما يحدث مع أمر `Format` في نظام التشغيل DOS حيث يعطيك أمر `Format` الرسالة `Format Another (y/n)` وينتظر لتكتب حرف Y او حرف N ويظهر الحرف ويستظر حتى تضغط على مفتاح الادخال `Enter`.

مثال :

```
char a;  
a=getchar();  
printf("%c",a);
```

الدالة `getche()`

ملف التوجيه : `conio.h`

تستخدم لإدخال حرف واحد ويظهر هذا الحرف على الشاشة ولكنها تختلف عن الدالة `getchar()` في أنها لا تحتاج الى الضغط على مفتاح الادخال(`Enter`) للانتقال

للسطر التالي وتعمل هذه الدالة بطريقة مشابهة. لرسالة الخطأ التي تظهر في نظام التشغيل بهذا الشكل

(A)abort, (R)etry, (F)ail ?

حيث يتم التنفيذ بمجرد الضغط على أحد الحروف A,R,F بدون حاجة لضغط مفتاح الإدخال ويظهر الحرف على الشاشة

مثال :

```
char a;  
a=getche();  
printf("%c",a);
```

الدالة ()
getch

ملف التوجيه : *conio.h*

تستخدم لإدخال حرف واحد ولكن تختلف عن الدالتين السابقتين في أن هذا الحرف لا يظهر على الشاشة وكذلك في أنها لا تحتاج إلى الضغط على مفتاح الإدخال Enter للانتقال للسطر التالي وهذا ما يحدث في كثير من الأمثلة مثل حالة تنفيذ الامر dir/p في نظام التشغيل DOS حيث يتم عرض الفهرس صفحة بصفحة وينتظر الضغط على أي مفتاح للاستفمارا وعند الضغط لا يظهر الحرف المضغوط عليه ولكن ينفذ الامر فقط.

مثال :

```
char a;  
a=getch();  
printf("%c",a);
```

دالة طباعة حرف واحد (putchar)

ملف التحويلي : stdio.h

تستخدم لطباعة حرف واحد على الشاشة فمثلا الصورة: putchar('a'); تطبع على الشاشة الحرف a كما هو والمثال الموجود في (شكل ٢-٧) يوضح كيفية استخدام دوال إدخال حرف وكذلك دالة طباعة حرف حيث يستقبل حرف باستخدام كل دالة لتوضيح الفرق بينهم.

```
0: /* Program Name : cs2_7.c */
1-1: #include <stdio.h>
1-2: #include <conio.h>
2: main ()
3: {
4:     char ch1,ch2,ch3;
5:     printf("\n ch1=");
6:     ch1=getchar();
7:     printf("\n ch2=");
8:     ch2=getche();
9:     printf("\n ch3=");
10:    ch3=getch();
11:    putch(ch1);
12:    putch(ch2);
13:    putch(ch3);
14: }
```

شكل ٢-٧ استخدام دوال إدخال وطباعة حرف

وفي هذا البرنامج

- في السطر رقم ٥ الدالة () printf بينما يشتمل السطر رقم ٦ على الدالة () getchar وهي تستقبل حرف وتخزنه في المتغير ch1 وبالمثل نفهم السطور ٧ و ٨ والسطور ٩ و ١٠ مع ملاحظة الفرق في طبيعة كل دالة
 - ويشتمل السطر رقم ١٠ على الدالة () getch التي تسمح باستقبال حرف من لوحة المفاتيح إلا أن هذا الحرف لا يظهر على الشاشة.
 - وفي السطور رقم ١١ و ١٢ و ١٣ الدالة () puts وتستخدم لطباعة الحروف المخزنة بالمتغيرات ch3,ch2,ch1
- وعند تنفيذ البرنامج ستحصل على النتيجة التالية

ch1=a

ch2=b

ch3=

a

b

c

دالة طباعة عبارة حرفية (puts())

ملف الترجمة : stdio.h

تستخدم لطباعة عبارة حرفية string حيث تطبع بدون توصيف شكل

المحركات

مثال

```
char name[10] = "Ahmed";
puts(name);
puts("Mohamed");
```

وعند تنفيذ هذا المثال نحصل على النتيجة التالية

Ahmed

Mohmaed

الاعلان [10] معناه أن المتغير name من نوع حرفى ويصلح لتخزين كلمة أقصى عدد حروف لها هو ١٠ حروف



دالة إدخال عبارة حرفية () gets()

ملف التوجيه : stdio.h

وتستخدم الدالة gets() في إدخال عبارة حرفية string انظر المثال التالي:

```
char name [20];  
gets(name);
```

وفي هذا المثال تخزن الدالة gets() العبارة الحرفية في المتغير name

مثال

يوضح المثال الموجود بشكل ٢-٨ استخدام الدالتين () , puts()

```
0: /*Program Name : cs2_8.c*/  
1: #include <stdio.h>  
2: main ()  
3: {  
4:     char name[10];  
5:     printf("\n Enter your name:");  
6:     gets(name);  
7:     puts("welcome ");  
8:     puts(name);  
9: }
```

شكل ٢-٨ استخدام دالى () , puts()

وعند التنفيذ ستحصل على النتيجة التالية :

Enter your name: Hesham

welcome

Hesham

وعن هذا البرنامج نوضح ما يلى:

- في السطر رقم ٥ دالة الطباعة printf() تظهر الرسالة: Enter your name:
- في السطر رقم ٦ دالة إدخال العبارة الحرفية gets() تخزن العبارة المدخلة في المتغير name الذي تم الإعلان عنه في السطر الرابع.
- في السطر رقم ٧ الدالة puts() تطبع الرسالة welcome
- في السطر رقم ٨ الدالة puts() تطبع محتويات المتغير name الذي أدخله المستخدم

مما سبق تجد أن لغة C تحتوى على دوال كثيرة للإدخال والإخراج مثل :

printf(), scanf(), putch(), getchar(), getche(), getch(), puts(), gets()

والسؤال : لماذا تعددت ومتى نستخدم ايها؟

تعدد دوال الإدخال والاخراج ليعطى المبرمج المرونة والقدرة في كتابة البرنامج حيث لديه أكثر من طريقة للتعامل مع البيانات وما تعجز عنه دالة تقوم به دالة أخرى وهكذا فمثلا الدالة scanf() عند استخدامها لإدخال عبارة حرفية في متغير لا تسمح بوجود مسافات داخل نفس العبارة الحرفية حيث أنها تستخدم المسافة كفاصل بين المتغيرات المدخلة في حين أن الدالة gets() تستقبل البيانات الحرفية بشتى أنواعها بما فيها المسافة.

وأيضا كل دالة لها استخدام أمثل، فمثلاً إذا أردت إدخال أو طباعة أكثر من قيمة في سطر واحد وهذه القيم مختلفة النوع تكون أنساب دالة هي `(scanf(),printf())`، أما إذا أردت إدخال أو طباعة عبارة حرفية `string` فتكون أنساب دالة هي `(gets() و puts()` وإذا أردت إدخال أو طباعة حرف واحد تحدد طريقة الإدخال المطلوبة وتستعمل أحدي الدوال `(putch(),getchar(),getche(),getch())` و `(clrscr())`

دوال تحسين المدخلات والمخرجات

يحتاج المبرمج المحترف لإضافة بعض اللمسات الجمالية على شكل البيانات المدخلة والمخرجة لدفع الملل عن مستخدم البرنامج وسنشرح فيما يلى الدوال التي تساعدك على تحسين شكل المخرجات

دالة مسح الشاشة `(clrscr())`

ملف التوجيه: `conio.h`

تستخدم لمسح الشاشة ووضع المؤشر في أول عمود من الصف الأول على الشاشة وتستخدم بالشكل التالي
`clrscr()`

دالة تغيير موضع المؤشر `(gotoxy())`

ملف التوجيه: `conio.h`

تستخدم لوضع المؤشر في العمود `x` من الصف `y` وتأخذ الصورة التالية :
`gotoxy(x,y)`

مثال :

للاتصال بالمؤشر الى العمود رقم ٣٠ من الصف العاشر نستخدم الدالة بالصورة

التالية :

`gotoxy(30,10)`

وهكذا تساعد هذه الدالة مع الدالة `printf()` على الكتابة في اي مكان على الشاشة مما يساعد على تنسيق شكل المخرجات

دالة تغيير لون الكتابة (`textcolor()`)

ملف التوجيه : `conio.h`

تستخدم لتعديل لون الكتابة التي ستطبع بعد الدالة وتأخذ الصيغة :

`textcolor(color no)`

أو الصيغة :

`textcolor(color name)`

حيث يتم تحديد اللون إما برقم اللون أو ياسمه ولابد من كتابة إسم اللون بالحروف الكبيرة فقط فمثلا للطباعة باللون الأزرق اكتب الدالة بالصورة الآتية :

`textcolor(1)`

أو

`textcolor(BLUE)`

والجدول التالي ٣-٢ يوضح أكواد الألوان وأسمائها .

اسم اللون	رقم اللون	اللون
BLACK	٠	اسود
BLUE	١	أزرق
GREEN	٢	أخضر

الفصل الثاني: دوال الإدخال والإخراج

إسم اللون	رقم اللون	اللون
CYAN	٣	سماوي
RED	٤	أحمر
MAGENTA	٥	بنفسجي
BROWN	٦	بني
LIGHTGRAY	٧	رمادي فاتح
DARKGRAY	٨	رمادي غامق
LIGHTBLUE	٩	أزرق فاتح
LIGHTGREEN	١٠	أخضر فاتح
LIGHTCYAN	١١	سماوي فاتح
LIGHTRED	١٢	أحمر فاتح
LIGHTMAGENTA	١٣	بنفسجي فاتح
YELLOW	١٤	أصفر
WHITE	١٥	أبيض

جدول ٢-٣ أكواود الألوان وأسمائها

إذا أردت استعمال اسماء الالوان مباشرة فعليك أن تكتب السطر

#include <conio.h>

يشتمل شكل ٢-٩ على برنامج بسيط يطبع الألوان التي يمكن استعمالها

باستخدام الأرقام الدالة عليها

0: /* Program Name : cs2_9.c */

1-1: # include <stdio.h>

1-2: #include <conio.h>

```
2: int main(void)
3: {
4:     int i;
5:     for (i=0; i<15; i++)
6:     {
7:         textcolor(i);
8:         cprintf("    Foreground Color=%d\r\n",i);
9:     }
10:    getch();
11:    return 0;
12: }
```

شكل ٢-٩ برنامج طباعة الألوان

في السطر رقم ٧ استخدمنا الدالة (textcolor(i)) لتغيير لون الكتابة إعتماداً على قيمة التغير i الذي تتغير قيمته مع التكرار وقد إستخدمنا الدالة (cprintf) بدلاً من الدالة (printf) حيث أن الأخيرة لا تعمل مع الألوان وعند استعمال الدالة (cprintf) نستبدل العلامة \n بالعلامة \r\n

دالة تغيير لون الخلفية (textbackground())

ملف التوجيه : *conio.h*

وستستخدم لتغيير لون خلفية الكتابة التي ستطيع بعد تحديد لون الخلفية بها وتأخذ الصيغة التالية :

textbackground(color no);

أو الصيغة التالية :

textbackground(color name);

و معاملات هذه الدالة هي نفس معاملات الدالة السابقة ()textcolor مع ملاحظة أن الدالة ()textbackground لا تستخدم سوى الألوان من رقم ١ إلى رقم ٧ المذكورين في الجدول رقم ٢-٣

ويشتمل شكل ٢-١٠ على مثال يوضح استخدام الألوان مع الكتابة والخلفية

```
0: /*Program Name :cs2_10.c */
1-1: #include <stdio.h>
1-2: #include <conio.h>
2: int main(void)
3: {
4:     int i, j;
5:     clrscr();
6:     for (i=0; i<9; i++)
7:     {
8:         for (j=0; j<5; j++)
9:             cprintf("Allah ");
10:            cprintf("\r\n");
11:            textcolor(i+1);
12:            textbackground(i);
13:    }
14:    return 0;
15: }
```

شكل ٢-١٠ برنامج يوضح استخدام الألوان مع الكتابة والخلفية

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٩ الدالة printf("Allah "); تطبع الكلمة باللون الحالي
- في السطر رقم ١٠ الدالة printf("\r\n"); لطباعة سطر جديد
- في السطر رقم ١١ الدالة textcolor(i+1); تغير لون الكتابة باللون رقم i+1

٠ في السطر رقم ١٢ الدالة textbackground(i) تغير لون الخلفية باللون رقم i

دوال الإدخال والإخراج التي تستخدم الألوان

سبق أن أشرنا إلى أن دالة الطابعة printf و دالة الإدخال scanf وكذلك باقي الدوال التي تم شرحها قد صممت بحيث تعمل باللون الأبيض على الأسود ولا تتأثر بدوال تغيير الألوان فإن هناك مجموعة من الدوال المقابلة للدوال السابقة والتي صممت للتعامل بالألوان المحددة وكلها مسبوقة بالحرف C مثل cprintf(),cscanf(),cputs(),cgets(),.....

وكلها معرفة في الملف conio.h

إذاً إستخدمت الدالة textbackground() لتحديد لون الخلفية وأتبعتها بالدالة clrscr() سيتم مسح الشاشة باللون الذي تم تحديده ويصبح هذا اللون هو لون خلفية الشاشة كلها أما إذا لم تستخدم الدالة clrscr() فأن لون الخلفية سيخصص للكلمات التي تكتب فقط بينما يبقى لون الشاشة كما هو

نفذ البرنامج الموجود بشكل ٢-١١ ولاحظ الألوان التي ستحصل عليها

```
0: /*Program Name : cs2_11 */
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     textbackground(BLUE);
6:     clrscr();
7:     textcolor(RED);
8:     cprintf("\r\n THIS TEXT DISPLAYED WITH RED ON
          BLUE COLOR");
9:     getch();
10:    textbackground(GREEN);
```

```
11: clrscr();  
12: textcolor(YELLOW);  
13: printf("\r\n THIS TEXT DISPLAYED WITH YELLOW  
ON RED COLOR");  
14: }
```

شكل ٢-١١ برنامج لتغيير الألوان

وعن هذا البرنامج نوضح مايلي

- في السطر رقم ٥ تغير الدالة textbackground(BLUE); لون خلفية الكتابة الى اللون الازرق
- في السطر رقم ٦ تممسح الدالة clrscr() الشاشة (يصبح لونها ازرق)
- في السطر رقم ٧ الدالة textcolor(RED); تغير لون الكتابة الى اللون الاحمر
- وهكذا

لاحظ \r\n : تقابل \n في حالة الدالة printf()

دوال التعامل مع الطابعة

من الأمور المهمة للمبرمج المبتدئ والمتحترف على السواء إخراج البيانات على الطابعة والتعامل مع الطابعة.

ولغة C تمكنتنا من التعامل مع الطابعة من حيث إخراج البيانات وإختبار حالة الطابعة وهل هي تعمل أم مغلقة وهل الورق موجود بالطابعة أم لا وفي الفقرة التالية سوف نعرض لأبسط صورة للتعامل مع الطابعة وهي الطابعة فقط أما باقى الحالات فستتعرض لها فيما بعد

وستستخدم الدالة fprintf() للطابعة على الطابعة ولكن مع معرف الطابعة القياسي stdprn ويظهر كأول معامل في الدالة fprintf والمتغير stdprn يعني الطابعة ومعناه

طبع هذه الرسالة على الطابعة ومعنى ذلك أن هناك معرفات أخرى لها معنى مختلف سوف نتعرض لها لاحقا.

وتشبه الدالة `fprintf()` الدالة `printf()` حيث تستخدم نفس المعاملات التي تستخدمها إلا أن المخرجات تظهر على الطابعة. وكذلك باقي صور الدالة `printf()` مسموح بها مع `fprintf()`

جرب البرنامج التالي :

```
0: /* Program Name :cs2_12.c
1: #include <stdio.h>
2: main ()
3: {
4:     fprintf(stderr," These words go to the printer.....");
5: }
```

شكل ٢-١٢ برنامج للطباعة على الطابعة

ستحصل على الرسالة التالية :
These words go to the printer

تقدير

عندما تريـد	استـخدم الدـالة
طباعة حروف أو أرقام	<code>printf()</code>
طباعة عبارات حرفية فقط	<code>puts()</code>

استخدم الدالة	عندما تريده
scanf()	إدخال حروف أو أرقام
gets()	إدخال عبارات حرفية
getch()	إدخال حرف دون إظهاره على الشاشة
getche()	إدخال حرف مع إظهاره على الشاشة
clrscr()	مسح الشاشة
gotoxy()	تغيير موضع المؤشر
textcolor()	تغيير ألوان الكتابة
textbackground()	تغيير ألوان خلفية الكتابة
cprintf(), cputs	الطباعة بالألوان على الشاشة
cscanf(), cgets	إدخال البيانات بالألوان
fprintf()	الطباعة على الطابعة





الفصل الثالث

الدوارات LOOPS

في هذا الفصل سنشرح موضوعاً من الموضوعات المهمة وهو موضوع الدوايرات ونقصد به تكرار تنفيذ مجموعة من الأوامر أكثر من مرة. وذلك من خلال الموضوعات التالية

- ♦ الدوارة **for**
- ♦ تنفيذ أكثر من جملة مع الدوارة **for**
- ♦ الدوايرات المتداخلة بإستخدام **for**
- ♦ الدوارة اللانهائية بإستخدام **for**
- ♦ الدوارة **while**
- ♦ الفرق بين **For** و **while**
- ♦ الدوارة اللانهائية بإستخدام **while**
- ♦ الدوارة **do while**
- ♦ الفرق بين **do.....while** و **while**
- ♦ الدوارة اللانهائية بإستخدام **do.....while**

الدّوّارة for

تستخدم لـ تكرار تنفيذ عملية عدد محدد من المرات وتأخذ الصيغة العامة التالية :

**for(initial-value;condition;increment)
statements;**

حيث

Initial-Value هي القيمة الابتدائية

Condition هو شرط انهاء التكرار

increment هي قيمة الزيادة الدورية

و معناها ابدأ العد من القيمة الابتدائية (initial-Value) طالما أن الشرط (Condation) صحيح ومقدار الزيادة كل مرة هو increment.

مثال ١

البرنامج الموجود في شكل ٣-١ يستخدم الدّوّارة for لطباعة كلمة Allah عشر مرات كل كلمة في سطر مستقل

```
0: /* Program Name :cs3_1.c */  
1: #include <stdio.h>  
2: main()  
3: {  
4:     int i;  
5:     for(i=0;i<10;i++)  
6:         printf("\n Allah");  
7: }
```

شكل ٣-١ استخدام الدّوّارة for

الفصل الثالث : الدوالات LOOPS

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٤ اعلان عن متغير صحيح من نوع int
- في السطر رقم ٥ جملة for حيث يبدأ بقيمة للمتغير ١ تساوى صفر ثم يزيدتها كل مرة واحد حتى تصل القيمة الى ١٠ وفي كل مرة يطبع كلمة Allah في سطر مستقل
- في السطر رقم ٦ الجملة printf("\n allah"); يتم تنفيذها ١٠ مرات وكل مرة يتم زيادة قيمة المتغير ١ واختبار قيمته هل وصلت الى القيمة ١٠ أم لا فإذا وصلت القيمة الى ١٠ يتوقف التكرار

وعند تنفيذ هذا البرنامج ستحصل على النتيجة التالية :

Allah
Allah
Allah

Allah

مثال ٢

والبرنامج الموجود في شكل ٣-٢ يطبع الأرقام من صفر إلى ٩

```
0: /*Program Name : cs3_2.c */  
1: #include <stdio.h>  
2: main()  
3: {  
4:     int i  
5:     for(i=0;i<10;i++)  
6:         printf("\n i=%d",i);  
7: }
```

شكل ٣-٢ برنامج طباعة الأرقام من صفر إلى ٩

مثال ٣

والبرنامج الموجود في شكل ٣-٣ يطبع شفرة Ascii وهي الأكواود المقابلة للأحرف القابلة للطباعة وغير القابلة للطباعة

```

0: /* Program Name : cs3_3.c */
1: /*Ascii example*/
2: #include <stdio.h>
3: #include <conio.h>
4: main ()
5: {
6:     int i;
7:     for (i=0;i<256;i++)
8:         printf ("%d=%c\n",i,i);
9: }

```

شكل ٣-٣ برنامج طباعة شفرة Ascii

000	<null>	016 ▶	<dle>	032 sp	048 0	064 e	080 P	096 '	112 p
001 ☐	<soh>	017 ↗	<dc1>	033 ↘	049 1	065 A	081 Q	097 a	113 q
002 ☒	<stx>	018 ↗	<dc2>	034 "	050 2	066 B	082 R	098 b	114 r
003 ☓	<etx>	019 ↗	<dc3>	035 #	051 3	067 C	083 S	099 c	115 s
004 ☔	<etb>	020 ↗	<dc4>	036 \$	052 4	068 D	084 T	100 d	116 t
005 ☕	<eng>	021 ↗	<nak>	037 %	053 5	069 E	085 U	101 e	117 u
006 ☖	<ack>	022 ↗	<syn>	038 &	054 6	070 F	086 V	102 f	118 v
007 ☗	<bel>	023 ↗	<etb>	039 ,	055 ?	071 G	087 W	103 g	119 w
008 ☘	<bs>	024 ↗	<can>	040 <	056 8	072 H	088 X	104 h	120 x
009 ☙	<tab>	025 ↗		041 >	057 9	073 I	089 Y	105 i	121 y
010 ☚	<lf>	026 ↗	<eof>	042 *	058 :	074 J	090 Z	106 j	122 z
011 ☛	<ut>	027 ↗	<esc>	043 +	059 ;	075 K	091 [107 k	123 <
012 ☜	<np>	028 ↗	<fs>	044 -	060 <	076 L	092 \	108 l	124 \
013 ☝	<cr>	029 ↗	<gs>	045 =	061 =	077 M	093 ^	109 m	125 ^
014 ☞	<so>	030 ↗	<rs>	046 >	062 >	078 N	094 _	110 n	126 _
015 ☟	<si>	031 ↗	<us>	047 >	063 ?	079 o	095 -	111 o	127 △
128 ☑	143 8	158 R	172 %	186	200 u	214	228 E	242 z	
129 ☒	144 6	159 f	173 ↗	187]]	201]]	215]	229 o	243 s	
130 ☓	145 z	160 6	174 «	188 »	202]	216]	230 μ	244 J	
131 ☔	146 g	161 i	175 »	189 »	203]	217]	231 τ	245]	
132 ☕	147 6	162 6	176 ↗	190 ↗	204]	218]	232 x	246 +	
133 ☖	148 8	163 u	177 ↗	191 ↗	205 =	219]	233 0	247 N	
134 ☗	149 o	164 n	178 ↗	192 ↗	206 v	220]	234 ο	248 °	
135 ☘	150 o	165 N	179 ↗	193 ↗	207 v	221]	235 6	249 -	
136 ☙	151 u	166 a	180 ↗	194 ↗	208 u	222]	236 *	250 .	
137 ☚	152 u	167 o	181 ↗	195 ↗	209 τ	223]	237 *	251 ✓	
138 ☛	153 8	168 z	182	196 -	210	224 a	238 €	252 *	
139 ☜	154 u	169 r	183 n	197 +	211	225 o	239 n	253 *	
140 ☝	155 6	170 ↗	184 ↗	198 ↗	212 t	226 r	240 ≡	254 ▨	
141 ☟	156 F	171 %	185	199	213 F	227	241 ±	255 ▨	
142 ☟	157 W								

Ascii Code شفرة

الفصل الثالث : الدوالات LOOPS

وفي البرنامج الموجود في الشكل رقم ٣-٣ في السطر رقم ٧ تقوم الدوارة `for` بغير قيمة المتغير من القيمة ٠ الى القيمة ٢٥٥ وكل مرة تقوم الدالة `printf()` بطباعة قيمة المتغير `i` والعرف المقابل لهذا الكود وبالتالي نحصل على جدول شفرة Ascii

تنفيذ أكثر من جملة مع `for`

في جملة `for` السابقة كنا نطلب تكرار تنفيذ جملة واحدة عدد محدد من المرات لكن ما هو العمل اذا كان المطلوب تكرار تنفيذ أكثر من جملة لعدد محدد من المرات. بعبارة أخرى تكرار تنفيذ بلوك كامل داخل البرنامج أكثر من مرة هذا ماستحاول الاجابة عليه فيما يلى :

لو أردنا تنفيذ أكثر من جملة لعدد محدد من المرات يجب وضع القوس { في بداية البلوك المراد تكراره ووضع القوس } في نهاية البلوك.

مثال

اقرأ المثال الموجود بشكل ٤-٣ واكتبه وتحصنه جيداً وقبل أن تنفذه حاول الاجابة على السؤال التالي :

ماذا تعتقد أن يكون شكل المخرجات ؟

```
0:/*Program Name : cs3_4.c */  
1:#include <stdio.h>  
2:main()  
3:{  
4:    int i;  
5:    for(i=0;i<10;i++)  
6:        printf("\n wlecome ");  
7:    printf("with ");
```

```
8:     printf("CompuScience");
9: }
```

شكل ٤-٣ استخدام الدوارة for

من أول نظرة للبرنامج الموجود بالشكل ٤-٣ قد تظن أن المخرجات طباعة الكلمات الآتية ١٠ مرات

Welcome With CompuScience

ولكن لونفذنا البرنامج سيكون الناتج هو طباعة الكلمة Welcome ١٠ مرات ثم طباعة الكلمات With CompuScience مرة واحدة هل عرفت ما هو السبب ؟؟؟ السبب هو أن جملة for تأخذ أول أمر يليها فقط وتنفذه ثم بعد ذلك تنتقل إلى الأوامر التالية.

ولكن كيف يمكننا طباعة الثلاث جمل ١٠ مرات ؟

لابد من تحديد أول هذه الجمل بالقوس { وفي نهاية الجمل القوس } فيصبح البرنامج السابق كما في الشكل ٣-٥

```
/* Program Name :cs3_5.c */
/* program for print 10 once */
1: #include <stdio.h>
2: main()
3: {
4:     int i;
5:     for(i=0;i<10;i++)
6:     {
7:         printf("\n wlecome ");
8:         printf("with ");
9:         printf("CompuScience");
```

الفصل الثالث : الدوالات LOOPS

```
10:    }
11: }
```

شكل ٣-٥ برنامج تكرار تنفيذ مجموعة جمل

وبالتالي عند تنفيذ البرنامج الموجود بشكل ٣-٥ ستحصل على عبارة

Welcome With CompuScience

عدد ١٠ مرات وذلك نتيجة اضافة السطر رقم ٦ أى القوس { قبل الجمل

المراد تكرارها والسطر رقم ١٠ أى القوس { في نهاية الجمل المراد تكرارها

تغيير معدل الزيادة

في الأمثلة السابقة إستخدمنا تعبير الزيادة ++ و معناه زيادة قيمة المتغير

بمقدار واحد في كل مرة

ولكن في بعض الأحيان نحتاج أن يكون مقدار الزيادة أكثر أو أقل من واحد

حينئذ يلزم تغيير هذا التعبير + + إلى الصورة المطلوبة.

مثلا $i = i + 2$ أى زيادة قيمة i كل مرة بمقدار إثنين

$i = i + 0.5$ أى زيادة قيمة i كل مرة بمقدار 0.5 وهكذا

والبرنامج الموجود في شكل ٣-٦ يستعمل الدوارة for ولكن بمعدل زيادة مقداره ٢

في كل مرة

```
0:/* Program Name : cs3_6.c */
```

```
1:#include <stdio.h>
2:main()
3:{}
4:int i;
5:for(i=0;i<10;i+=2)
```

```
6:     printf("\n i=%d",i);
7: }
```

شكل ٣-٦ طباعة الأرقام بمعدل زيادة ٢

و عند تنفيذ البرنامج ستحصل على النتيجة التالية :

```
i=0
i=2
i=4
.....
i=8
```

تغبير معدل الزيادة بالسالب

في بعض الاحيان نحتاج أن يكون معدل التغبير بالسالب كما في بعض العمليات الحسابية مثلا ١٠ ، ٩ ، ٨ ... وهكذا

والبرنامج الموجود وبشكل ٣-٧ يقوم بطباعة الأرقام من ١٠ الى ١ وذلك باستعمال خطوة سالبة مع الدوارة for

```
0:/* Program Name : cs3_7.c */
1:#include <stdio.h>
2:main()
3:{ 
4:    int i;
5:    for(i=10;i>0;--i)
6:        printf("\n i=%d",i);
7: }
```

شكل ٣-٧ طباعة الأرقام بخطوة سالبة

و عند تنفيذ البرنامج نحصل على النتيجة التالية :

```
i=10
```

الفصل الثالث : الدورات LOOPS

i=9

i=8

...

i=1

وفي هذا البرنامج يشتمل السطر رقم ٥ على الجملة `for(i=10;i>0;i--)` و معناها أبدأ بوضع القيمة ١٠ في المتغير i ثم انقص هذه القيمة بمقدار واحد كل مرة نتيجة للتعبير `--i` والشرط أن i أكبر من القيمة صفر وبالتالي يتم العد التنازلي.

الدورات المتداخلة باستخدام for

الدورات المتداخلة عبارة عن دوارة كبيرة تشمل بداخلها على دوارة أو أكثر، بمعنى أن مجموعة التعليمات الموجودة بالدوارة الداخلية يتكرر تنفيذها طالما لم ينته العدد فإذا انتهى يتنقل التنفيذ إلى الدوارة الخارجية ويتكرر تنفيذ تعليمات الدوارة الخارجية حتى ينتهي العدد المحدد لها . وتشبه فكرة الدورات المتداخلة لكرة عمل عقارب الساعة فتجد عقرب الثوانى يدور ٦٠ دورة فيدور عقرب الدقائق بمقدار دقيقه وهكذا.

والبرنامج الموجود في الشكل رقم ٣-٨ يستخدم فكرة الدورات المتداخلة لطباعة الأرقام من صفر إلى ٩ باستخدام الدوارة الداخلية ثم يطبع الأرقام من صفر إلى ٥ باستخدام الدوارة الخارجية. بحيث نحصل في النهاية على طباعة رقم من الدوارة الخارجية مقابل طباعة عشرة أرقام من الدوارة الداخلية.

```
0:/* Program Name : cs3_8.c */
1:#include <stdio.h>
2:main()
3:{}
4:    int i;
5:    for(i=0;i<5;i++)
6:    {
```

```

7:     for(j=0;j<10;j++)
8:     {
9:         printf("\n i=%d",i);
10:        printf("\t j=%d",j);
11:    } /* first for loop */
12: } /* secound for loop */
13: */ /* main() function */

```

شكل ٣-٨ استخدام الدورات المتداخلة

ومن الأمثلة المفيدة لاستخدام الدورات المتداخلة استخدام الدوارة for لطباعة جدول الضرب من أول 1×1 إلى 12×12 حيث نستعمل الدوارة الخارجية لتغيير الرقم المضروب ونستعمل الدوارة الداخلية لتغيير الرقم المضروب فيه من 1 إلى 12 كما يتضح من البرنامج الموجود في الشكل رقم ٣-٩

```

0:/* Program Name :cs3_9.c */
1: #include <stdio.h>
2: main()
3: {
4:     int i,j;
5:     for(i=1;i<13;i++)
6:     {
7:         printf("\n");
8:         for(j=i;j<13;j++)
9:         {
10:             printf(" i\t j=%d",i*j);
11:         } /* first for */
12:     } /* secand for */
13: } /* main() function */

```

شكل ٣-٩ برنامج استخدام الدورات المتداخلة لطباعة جدول الضرب

أكتب البرنامج ونفذه وتأكد من النتيجة

ولتحسين البرنامج بحيث لا يطبع القيم المكررة مثل 2×1 و 2×2 لقوم بتعديل بداية الدوارة الثانية بحيث تبدأ من القيمة 1. فيصبح السطر رقم 7 كما يلى:

```
for (j = i;j < 13;j++)
```

الدوارة اللانهائية بإستخدام for

و معناها تكرار تنفيذ الجملة بدون شرط ولا يتوقف التنفيذ حتى يضغط المستخدم Ctrl+C او تأخذ الدوارة اللانهائية باستخدام for الصورة

```
for(;;)
```

والمثال التالي يستعمل الدوارة اللانهائية للاستمرار في طباعة كلمة Allah مالم يضغط المستخدم على Ctrl+C فإذا ضغط المستخدم على Ctrl+C تنتهي الدوارة ويترافق التنفيذ

```
#include <stdio.h>
main ()
{
    for(;;)
        printf("\t Allah");
}
```

لوكبرت الدوارة بالصورة for(;;) وأتبعتها بالعلامة ؛ وحاوت تنفيذ الملف النهائي (EXE). فان الجهاز يدخل في مسار لانهائي(hang) ولا تستطيع حل هذه المشكلة الا بغلق الجهاز وتشغيله مرة أخرى ويمكك استخدام هذه الفكرة مع برنامج كلمة السر فإذا أدخل المستخدم كلمة سر خطأ لا يستجيب الجهاز لاي أمر ولا بد من غلقه.



الدالة while

تستخدم الدالة while لتكرار تنفيذ جملة أو مجموعة جمل عدد من المرات غير معلوم العدد وانما يتوقف هذا العدد على شرط موجود بالدالة وتأخذ الدالة while الصورة التالية :

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

ويعندها ظالماً أن الشرط (condition) صحيح فنفذ الجملة (Statement)

وهي تقوم بتكرار الجملة أو مجموعة الجمل التابعة لها ظالماً كان شرط التكرار صحيح وعندما يصبح شرط التكرار غير صحيح يتوقف تنفيذ الدالة

والبرنامج الموجود في الشكل ٣-١٠ يستخدم الدالة while لطباعة كلمة Allah ١٠ مرات كل مرة في سطر مستقل وهذا الاستخدام للدالة while غير جيد ولكننا نكتب هذا البرنامج لوضوح أجزاء الدالة while

```
0: /* Program Name : cs3_10.c */
1:#include <stdio.h>
2: main ()
3: {
4:     int i=0;
5:     while(i<10)
6:     {
7:         printf("\n Aallah");
8:         i++;
9:     }
10: }
```

شكل ٣-١٠ استخدام while

وعن هذا البرنامج

- في السطر رقم ٤ إعلان عن متغير من نوع صحيح هو `a`
- السطر رقم ٥ معناه طالما أن قيمة `a` أقل من ١٠ نفذ الجمل التالية وبالفعل أول مرة ستكون قيمة لمتغير `a` تساوى صفر وبالتالي يتم تكرار الدوارة
- السطر رقم ٧ معناه نفذ الجملة `printf("\n Allah");` طالما أن الشرط صحيح وهو طباعة كلمة `Allah`
- السطر رقم ٨ زيادة قيمة المتغير `a` بمقدار ١ ثم يعود البرنامج الى اختبار شرط الدوارة `while` فإذا كانت قيمة `a` ماتزال أقل من ١٠ يتكرر تفيد الجمل التالية وهكذا حتى يصبح الشرط غير صحيح

ملاحظات

١. لاحظ أن كتبنا القوس { بعد `while` لأننا نرغب في تكرار التنفيذ على أكثر من جملة (بلوك) والقوس { في سطر مستقل لاغلاق الدوارة
٢. لتغيير قيمة الزيادة أو جعلها بالسالب غير التعبير `++a` كما نشاء كما تم مع `for`

الفرق بين `for` و `while`

الدوارة `for` دوارة عدديّة حيث تعتمد على العداد وينتهي التكرار فيها بانتهاء عدد مرات التكرار أما الدوارة `while` فدوارة شرطيّة أي تعتمد على الشرط الذي يلى الأمر `while` حيث تكرر الجمل التي تليها طالما كان الشرط صحيحًا وتنتهي الدوارة بكسر هذا الشرط. وبالتالي الاستخدام الأمثل للدوارة `for` هو تكرار عملية أكثر من مرة بشرط أن يكون عدد مرات التكرار معلوم والاستعمال الأمثل للدوارة `while` هو التكرار بناء على شرط معين

مثال

يقوم البرنامج الموجود بشكل ٣-١١ باستقبال حروف من لوحة المفاتيح ويستمر في ذلك حتى تضغط مفتاح Enter وبعد ضغط مفتاح Enter يتوقف قبول الحروف ويطبع البرنامج عدد الحروف التي أدخلتها.

```
1/* Program Name : cs3_11.c */
2: #include <stdio.h>
3: #include <conio.h>
4: main ()
5: {
6:     int count=0;
7:     printf ("\n ENTER CHARCTERS..\n");
8:     while (getche() !='r')
9:         count++;
10:    printf ("\n character count is %d ",count);
11: }
```

شكل ٣-١١ برنامج عد الحروف المدخلة

وعن البرنامج الموجود بالشكل ٣-١١ نوضح مايلي :

في السطر رقم ٨ تقوم الدالة getche() باستقبال حرف وتقوم الدوارة while باختبار هذا الحرف فإذا كان هذا الحرف هو مفتاح الادخال Enter يتوقف عمل الدوارة وإذا كان أي حرف آخر يقوم السطر رقم ٩ بزيادة قيمة المتغير count وهكذا حتى يضغط المستخدم على مفتاح الادخال وفي النهاية يطبع البرنامج عدد الحروف المدخلة.

الدواره اللانهائيه باستخدام while

سبق أن قلنا أن الدواره اللانهائيه مع for تأخذ الصورة (();) أما الدواره اللانهائيه مع while فتأخذ الصورة (1)while و هي أكثر استعمالاً من الدواره for والبرنامجه الموجود بشكل ٣-١٢ يطبع قيم متواتله ٠١٢٣ وهكذا وذلك باستعمال دواره لانهائيه حتى يضغط المستخدم على Ctrl+C.

```
/* Program Name : cs3_12.c */
#include <stdio.h>
main ()
{
    int i=0;
    while(1)
    {
        printf ("i=%d",i);
        i++;
    }
}
```

شكل ٣-١٢ الدواره اللانهائيه

الدواره do while

تستخدم الدواره while do لتكرار تنفيذ جملة أو مجموعة جمل أكثر من مرة بناء على شرط معين كما هو الحال مع الدواره while ولكن الفرق بينهما أن الدواره while تختبر الشرط أولاً فإذا كان صحيحاً تنفذ الجمل التالية لها ولا فلا ، أما الدواره do.....while فتنفذ الجمل التالية لها أولاً ثم تختبر الشرط. فإذا كان صحيحاً تعيد التنفيذ ولا توقف التكرار.

وتأخذ الدوارة do while الصيغة

```
do
{
    statement1;
}while (condition);
```

و معناها do أي نفذ الجمل التالية وهي Statement1 وما يليها طالما كان الشرط
Cndertion صحيح .

والبرنامج الموجود بشكل ١٣-٣ يستخدام الدوارة do ...while لطباعة رسالة
يطلب فيها من المستخدم ادخال كلمة سر ومقارنتها بالكلمة المخزنة فإذا كانت الكلمة
المدخلة صحيحة انتهى البرنامج والا يتم تكرار استقبال كلمة السر مرة اخرى .

```
1: /* Program name : cs3_13.c */
2: /* password...*/
3: #include <stdio.h>
4: #include <conio.h>
5: main ()
6: {
7:     char pass[10];
8:     do
9:     {
10:         printf ("\n Enter Password:");
11:         scanf ("%s",pass);
12:     } while (strcmp(pass, "azab") !=0);
13: }
```

شكل ١٣-٣ برنامج كلمة السر

الفصل الثالث : الدوالات LOOPS

- وعن هذا البرنامج نوضح ما يلى:
- في السطر رقم ٨ بداية الدوارة do والسطر رقم ١٠ يطبع الرسالة Enter على الشاشة Password:
 - والسطر رقم ١١ يستقبل كلمة ويخزنها في المتغير pass والسطر رقم ١٣ يقارن بين الكلمة التي أدخلت وكلمة azab فإذا كانتا متطابقتين ينتهي عمل do..while.
 - نفذ البرنامج وأدخل كلمة سر غير الواردة بالبرنامج وهي azab مرة واثنين قبل كتابة كلمة السر الصحيحة للاحظ تنفيذ البرنامج.

الدالة () strcmp تقوم بمقارنه متغيرين من نوع عبارة حرفيه string فإذا كان المتغيرين متطابقين كان الفرق بينهما صفر.



مثال

البرنامج الموجود بالشكل ٣-١٤ تعديل لبرنامج كلمة السر السابق بحيث لا تظهر كلمة السر التي يكتبها المستخدم وهي تكتب على الشاشة حتى لا يراها شخص آخر وتعتمد فكرته على تغيير الألوان.

اكتب البرنامج ونفذه ولاحظ ذلك.

```
1: /* Program Name : cs3_14.c */
2: /* password...*/
3: #include <stdio.h>
4: #include <conio.h>
5: main ()
6: {
7:     char ch;
8:     char pass[10];
```

```
9:     do
10:    {
11:        textColor(WHITE);
12:        textbackground(BLUE);
13:        cprintf ("\r\n Enter Password:");
14:        textbackground(WHITE);
15:        cscanf ("%s",pass);
16:    } while ((strcmp(pass,"azab")) !=0);
17: }
```

شكل ٤-٣ برنامج كلمة سر لا تظهر على الشاشة

وعن هذا البرنامج نوضح ما يلى :

- يقوم البرنامج في السطر رقم ١١ و ١٢ بتحديد لون الكتابة أبيض ولون الخلفية أزرق
- والسطر رقم ١٣ يطبع العبارة Enter Password بهذه الألوان.
- السطر رقم ١٤ يجعل لون الخلفية أبيض وبالتالي يتم الكتابة باللون الأبيض على خلفية بيضاء فلا تظهر كلمة السر وهذا هو المطلوب.
- ثم يعود البرنامج تغير الألوان وهكذا حتى يدخل المستخدم كلمة السر الصحيحة.

تذكرة

عندما تريد

إتبع الآتي

عمل دوارة تعتمد على عدد معلوم بدايته يستخدم الدوارة for
ونهايته

عمل دوارة تعتمد على شرط غير معلوم يستخدم الدوارة while

تنفيذ الدوارة أولا ثم الشرط يستخدم الدوارة do....while

تنفيذ أكثر من عبارة داخل البلوك ضع القوس { في بداية البلوك و القوس } في نهاية الدوارة





الفصل الرابع

التفريع Branching

البرامج التي شرحناها حتى الآن تفتقت بمفهوم تسلسلي (sequential) بمعنى أن البرنامج ينفذ أول تعليمية ثم التي تليها إلى أن ينتهي البرنامج ، وفي هذا الفصل سوف نتعرف على جمل الشرط التالية :

- ♦ جملة الشرط **if**
- ♦ جمل **if** الشرطية المتداخلة
- ♦ جملة الشرط **ifelse**
- ♦ جملة الشرط **if else if**
- ♦ التركيب **switch,case,break,default** & **continue**
- ♦ المؤثر الشرطي ?
- ♦ جملة التفرع غير المشروط **goto**

التفرير يعني تغير مسار البرنامج. والتفرير إما يكون مشروط كجملة if أو غير مشروط كجملة goto

التفرير المشروط

جملة الشرط if

تستخدم الكلمة if لتنفيذ جملة أو أكثر حسب شرط معين

وأبسط صورة لجملة if هي :

```
if (condition)
    statement;
```

ومعناها اذا تحقق الشرط (Condition) نفذ الجملة التالية أما اذا لم يتحقق الشرط فلا تنفذ هذه الجملة وانتقل إلى التي تليها.

أمثلة

يقوم البرنامج الموجود بالشكل ١-٤ باستقبال حرف ثم يختبر هذا الحرف باستعمال جملة if وعلى أساسها يطبع البرنامج النتيجة

```
0: /*Program Name : cs4_1.c */
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     char ch;
6:     clrscr();
7:     printf("\n type letter ....:");
8:     ch=getche();
9:     if (ch=='y')
```

```
10:     printf ("you typed letter y ");
11: }
```

شكل ٤-٤ استعمال الجملة if

وفي هذا البرنامج في السطر رقم ٨ تقوم الدالة () getch باستقبال حرف وتخزنه في المتغير ch وفي السطر رقم ٩ ، ١٠ يختبر هل هذا الحرف هو الحرف y أم لا ، فإذا كان الحرف المدخل هو y يطبع البرنامج رسالة you typed letter y ثم تنتهي والا ينتهي البرنامج بدون طباعة الرسالة

معنى هذا أن جملة if جمله اعتراضيه إذا تحقق شرطها يتم تنفيذ التعليمات المعتمدة على الشرط ثم باقى تعليمات البرنامج ، أما إذا لم يتحقق الشرط فلا تنفذ هذه التعليمات وينتقل التنفيذ إلى باقى جمل البرنامج .

يفرق مترجم لغة C بين الحروف الكبيرة والحروف الصغيرة ولحل هذه

 المشكلة هناك طريقتين

١. استخدام المؤثر المنطقي OR والذى يرمز له بالعلامة || كما يلى :

```
if(ch=='y'||ch=='Y')
```

٢. استخدام دالة التحويل إلى الحروف الكبيرة () مثل :

```
ch=toupper(ch)
```

أما البرنامج الموجود بالشكل رقم ٤-٤ فيسمح للمستخدم بكتابة مجموعة حروف وفي النهاية يطبع البرنامج عدد هذه الحروف وعدد الكلمات الموجودة في هذه الحروف . وذلك بإستعمال جملة الشرط if .

```
0: /*Program Name : cs4_2.c */
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     int charcnt=0;
```

```

6:     int wordcnt=0;
7:     printf ("\n type charcters: ");
8:     while ( (ch=getche()) !='\r')
9:     {
10:         charcnt++;
11:         if (ch==' ')
12:             wordcnt++;
13:     }
14:     printf ("In character count is %d",charcnt);
15:     printf ("\n word count is %d",wordcnt+1);
16: }
```

شكل ٢ - ٤ برنامج عد الحروف والكلمات

وعن هذا البرنامج نوضح مايلي:

- في السطر رقم ٧ يطبع البرنامج الرسالة type charcters
- يشتمل السطر رقم ٨ على الدوارة while وهي تختبر الحرف الذي استقبلته الدالة() getche() وطالما أن هذا الحرف ليس مفتاح Enter يستمر البرنامج في استقبال حرف جديد.
- في السطر رقم ١٠ تزداد قيمة المتغير charcnt بمقدار واحد كلما استقبل حرف جديد وبالتالي يغير المتغير charcnt عن عدد الحروف.
- السطر رقم ١١ يختبر ما إذا كان الحرف المدخل عبارة عن مسافة خالية أم لا فإذا كان الحرف المدخل مسافة خالية فهذا معناه أن المستخدم ضغط على مفتاح المسافات أى كتب كلمة جديدة وبالتالي يزداد عدد الكلمات بمقدار واحد ويظل هكذا حتى اذا ضغط المستخدم على مفتاح Enter ينهي البرنامج سطر ١٤ ، ١٥ يطبع عدد الحروف وعدد الكلمات التي أدخلها المستخدم و عند تنفيذ البرنامج ستحصل على النتيجة التالية :

type charcters : Allah the god of all world

charcter count is 27

word count is 6

اذا كان هناك أكثر من جملة نريد تنفيذها مع if لا بد من فتح قوس { قبل

مجموعة الجمل والقوس { في آخر الجمل كما يلى

```
if (condition)
{
    statement1;
    statement2;
}
```

جمل if الشرطية المتداخلة

يمكن أن تداخل جمل if فتأخذ الشكل التالي :

```
if (condition)
    if (condition)
        if (condition)
```

وهذا معناه اذا تحقق الشرط الاول انظر الى الشرط الثاني .. وهكذا

مثال

البرنامج الموجود بالشكل ٣-٤ تطبيق على تداخل جمل if

```
0: /* Program Name : cs4_3.c */
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     printf("type letters ....:");
6:     if (getch()=='n')
7:         if (getch()=='o')
8:             printf ("\n you typed no");
9: }
```

شكل ٣-٤ جمل if المتداخلة

وفي هذا البرنامج تستقبل الدالة `getch()` حرف وتقارنه مع الحرف `n` فإذا كانت المقارنة صحيحة تنفذ الجملة التالية التي هي شرط آخر فإذا تحقق هذا الشرط (وهو أن الحرف الثاني 0) تطبع الرسالة `you typed no`

 ماذا يحدث لو كان الحرف الأول أي حرف غير الحرف `n` نفذ البرنامج وأدخل حرفًا آخر. ولاحظ النتيجة وحاول تفسيرها.

الجملة الشرطية if else

تستخدم لتنفيذ أحد إختيارات وتأخذ الصورة التالية :

```
if (condition)
{
    statement1
}
else
{
    statement2
}
```

ومعناها إذا كان الشرط (Condition) صحيح نفذ الجملة الأولى statement1 والا نفذ الجملة الثانية2

وهذا يعني أن تركيب `if .. else` يستخدم لتحديد اختيار واحد من إختيارات ولا يمكن تنفيذ الاختيارات معاً كما يحدث مع جملة `if` وحدها .

والبرنامج الموجود بالشكل ٤-٤ يستعمل التركيب `if..else`

```
0: /*Program Name : cs4_4.c */
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     char ch;
6:     printf("\n\n type character :");
```

```
7:     if ((ch=getche()) =='y')
8:         printf ("In you typed letter y");
9:     else
10:        printf ("In you typed another letter");
11:    }
```

شكل ٤-٤ استخدام ..if ..else

وعن هذا البرنامج نوضح ما يلى :

في السطر رقم ٧ الدالة getche() تستقبل حرف وتخزنه في المتغير ch وتقارن
جملة if اذا كان هذا الحرف هو حرف y أم لا. فإذا كان الحرف يساوى
حرف y فهذا معناه أن الشرط تحقق وبالتالي تظهر الرسالة you

.typed letter y

• وسطر ٩ else معناه وإلا

• وسطر ١٠ يطبع العبارة you typed another letter

والبرنامج الموجود بالشكل رقم ٥-٤ يقوم باستعمال التركيب if ..else مع
دوارتين for ليعطي النتيجة الموجودة بالشكل رقم ٦-٤

```
0: /* Program Name : cs4_5.c */
1: #include <stdio.h>
2: #include <conio.h>
3: main()
4: {
5:     int r,c;
6:     clrscr();
7:     for(r=0;r<20;r++)
8:     {
9:         printf("\n");
10:        for(c=0;c<40;c++)
11:        {
```

```

12:         if(r==c||c==40-r)
13:             printf("*");
14:         else
15:             printf(".");
16:     } /* secound for */
17: } /* first for */
18: /* End of main() */

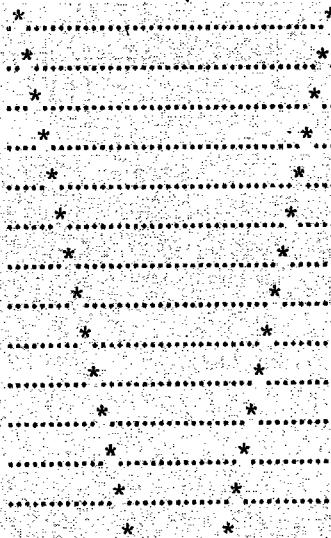
```

شكل ٤-٥ استعمال التركيب if...else مع دواراتين

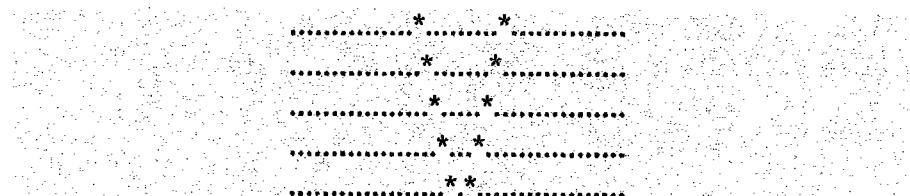
وعن هذا البرنامج نوضح ما يلى :

يستخدم هذا البرنامج فكرة استخدام الدوالات المتداخلة(Nested Loops) وهي عبارة عن دواري for لطباعة شبكة من النقط حيث تبدأ في السطر رقم ٧ الدوارة for يتتنفيذ ما بداخلها ٤ مرة ودوارة for أخرى في السطر رقم ١٠ تقوم بتنفيذ ما بداخلها ٤ مرة وهو طباعة النقطة . وبالتالي نتيجة سطر رقم ٧ وسطر رقم ١٠ هو رسم شبكة من النقط . عبارة عن ٤ نقط في ٢٠ سطر في السطر رقم ١٢ اذا تحقق هذا الشرط يوضع بدل النقطة الحرف * لحصول على شكل

٤-٦



الفصل الرابع : التفرع (Branching)



شكل ٤-٤ نتيجة تنفيذ برنامج cs4-5.c

الجملة الشرطية if else if

لتنفيذ خيار من مجموعة خيارات كمقارنة رقمين مثلا حيث يكون الرقم الأول أكبر من أو يساوى أو أقل من الرقم الثاني. يوجد طريقتين ، الطريقة الأولى باستخدام ثلاثة جمل if وفي كل جملة نضع أحد الشروط الثلاثة كالتالي :

```
i=5;  
if (i<5)  
    printf("i less than 5");  
if (i=5)  
    printf("i equal to 5");  
if (i>5)  
    printf("i greater than 5");
```

ويعيب تلك الطريقة أن البرنامج سيقوم باختبار شرط if الثلاثة حتى وإن كان الشرط قد تحقق في جملة if الثانية فهو لا بد أن يختبر جملة if الثالثة لأن كل جملة من جمل if مستقلة بنفسها ويجرى تنفيذها على حدة مما يستهلك وقتا في إختبار جمل شرطية لاداعي لإخبارها حيث نفذت إحداها بالفعل.

والطريقة الثانية تستخدم لخلافى ذلك العيب وفيها نستخدم الجملة الشرطية

if...else if وصيغتها كالتالي :

```
if (condition)
```

```
statement1;  
else if (condition )  
    statement2;  
else if (condition)  
    statement3;
```

في هذه الصيغة لا يتم إختبار جملة if الثانية إلا إذا كانت جملة if الأولى غير صحيحة.

وفي هذه الحالة تصبح صيغة المثال السابق كالتالي :

```
i=5;  
if (i<5)  
    printf("i less than 5");  
else if (i==5)  
    printf("i equal to 5");  
else if (i>5)  
    printf("i greater than 5");
```

وهذا معناه تحديد اختيار من عدة اختيارات

والبرنامج الموجود في الشكل ٧-٤ يستعمل التركيب if ..else if ..else في عمل مثال للة حاسبة بسيطة حيث يقوم البرنامج بطباعة رسالة يطلب فيها رقمين وبينهما علامة حسابية وعند ادخال الرقمين والعلامة الحسابية يطبع البرنامج النتيجة ثم يسأل هل تزيد الاستمرار ويستمر هكذا طالما تجib بالحرف y نفذ البرنامج ثمتابع معنا الشرح

```
0: /* Program Name : cs4_7.c */  
1: #include <stdio.h>  
2: #include <conio.h>  
3: main ()  
4: {  
5:     float num1,num2;  
6:     char op,ch;  
7:     do
```

```
8:     {
9:         printf ("\n Type num1,op,num2:");
10:        scanf ("%f %c %f",&num1,&op,&num2);
11:        if (op=='+')
12:            printf ("sum=%f",num1+num2);
13:        else if (op=='-')
14:            printf ("sub=%f",num1-num2);
15:        else if (op=='/')
16:            printf ("div=%f",num1/num2);
17:        printf ("\n again(y/n)");
18:    }while ((ch=getch()) == 'y');
19: }
```

شكل ٧-٤ برنامج يستخدم التركيب if...else if لعمل آلة حاسبة بسيطة

وعن هذا البرنامج نوضح ما يلى :

- السطر رقم ٩ يطبع العبارة Type num1,op,num2: باستخدام الدالة printf()
- في السطر رقم ١٠ تستقبل الدالة scanf() ثلاث متغيرات هم رقم ، مؤثر (علامة حسابية) ، ورقم.
- في السطر رقم ١١ جملة if تقول اذا كان المؤثر (العلامة الحسابية) المدخل هي علامة الجمع + نفذ السطر رقم ١٢ وهو جمع الرقمين وطباعة النتيجة. وهكذا باقي السطور حتى السطر رقم ١٦ وهي الحالات الأربع للمؤثر.
- في السطر رقم ١٨ الدالة getch() تستقبل حرف وتحتبر جملة while هذا الحرف فإذا كان حرف y يعود التنفيذ لبداية الدوارة من أول السطر رقم ٩ والا انتهى تنفيذ البرنامج. وبالتالي يماثل هذا البرنامج آلة حاسبة بسيطة لعمليات جمع وطرح وقسمة وضرب ويتم الخروج من البرنامج بإدخال أي حرف غير الحرف y

وعند تفريغ البرنامج نحصل على النتيجة التالية :

```
Type num1,op,num2:45+66
sum=111.000000
again(y/n)
Type num1,op,num2:88+22
sum=110.000000
again(y/n)
Type num1,op,num2:95-8
sub=87.000000
again(y/n)n
```

التفرريعswitch

لاحظت في المثال السابق أن استخدام جملة if في حالة تعدد الإختيارات لأكثر من إختيارات يمثل عبئاً على المبرمج في تبع خطوات البرنامج ويسبب بطلاً نسبياً في تفريغ البرنامج لهذا إستخدمنا الجملة الشرطية if...else if...else if... وهي طريقة أسهل كما سنرى وتستخدم كبديل لجملة if ... else if ... case switch ... case

بالصيغة التالية

```
ch=getch();
switch (ch)
{
    case '1':
        statement1;
        statement2;
        ....
        break;
    case '2':
        statement1;
        statement2;
```

الفصل الرابع : التفرع (Branching)

```
....  
break;  
default:  
    statement1;  
    statement2;  
...  
}
```

في هذا التركيب تختبر جملة switch المتغير OP ثم تفرض له مجموعة حالات هذه الحالات تحدد بكلمة case ، ففي الحالة الأولى (case "a") إذا كانت قيمة المتغير OP هي a يتم تنفيذ الجمل التالية وال موجودة في هذه الحالة ثم الخروج من تركيب .. switch ياستخدام كلمة break ، بالمثل الحالة الثانية وهكذا. أما إذا لم تتحقق حالة من الحالات يتجه التنفيذ إلى كلمة default وهي بمعنى إذا لم يتحقق أي حالة من الحالات السابقة نفذ مايلي:

ونصح باستعمال هذا الأسلوب في حالة اختيار حالة من مجموعة حالات .
والبرنامج الموجود بالشكل ٤-٨ تعديل لبرنامج الآلة الحاسبة السابق والموجود بالشكل ٤-٧ ولكن باستخدام التركيب switch ... case

```
0/* Program Name : os4_8.c */  
1: #include <stdio.h>  
2: #include <conio.h>  
3: main ()  
4: {  
5:     float num1,num2;  
6:     char ch,op;  
7:     do  
8:     {  
9:         clrscr();  
10:        printf ("\n Type num1 op num2:");  
11:        scanf ("%f %c %f",&num1,&op,&num2);
```

```

12:     switch (op)
13:     {
14:         case '+':
15:             printf ("sum=%f",num1+num2);
16:             break;
17:         case '-':
18:             printf ("sub=%f",num1-num2);
19:             break;
20:         case '*':
21:             printf ("mul=%f",num1*num2);
22:             break;
23:         case '/':
24:             printf ("div=%f",num1/num2);
25:             break;
26:         default:
27:             printf ("\n unknown operator..");
28:     }
29:     printf ("\n again (y/n):");
30: }while ((ch=getch()) =='y');
31: }
```

شكل ٨-٤ استخدام التركيب switch...case

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ١١ الدالة `scanf()` تستقبل ثلاث متغيرات رقمين ، ومؤشر (علامة حسابية) ، وفي السطر ١٢ الجملة `Switch` تختبر المتغير `op` وفى السطر رقم ١٤ الكلمة `case` تحدد حالة ما اذا كان المؤشر (العلامة الحسابية) هو علامة الجمع + فإذا كان المؤشر هو علامة الجمع + ينفذ السطر رقم ١٥ وهو جمع الرقمين وطباعة النتيجة ثم الخروج من الاختيارات عن طريق `break` وهكذا السطور حتى السطر رقم ٢٨

- في السطر رقم ٣٠ دالة getch() تستقبل حرف وتحتبره جملة while اذا كان y يعود التنفيذ مرة أخرى من أول السطر رقم ٩ والا انتهي تنفيذ البرنامج ومن هذا البرنامج نجد أن التركيب switch ... case أكثر وضوحاً من التركيب if..else if

وعند تنفيذ البرنامج نحصل على النتيجة التالية

```
Type num1 op num2:55+55
sum=110
again (y/n):y
Type num1 op num2:100-50
sub=50
again (y/n):n
```

مثال

من التطبيقات المشهورة لاستخدام التفريع switch().....case هو استخدامه في قوائم الاختيارات (menu) كما في الشكل التالي :

MAIN MENU

- 1-INTRODUCTION TO C-LANG.
- 2-STRUCTURE OF C-PROG.
- 3-LOOPS
- 4-DECISIONS.

ENTER SELECTION:

ويحتوى شكل ٩-٤ على البرنامج المطلوب لإظهار هذه القائمة والتعامل معها

```
0: /* Program Name : cs4_9.c */
1: #include <stdio.h>
1: #include <conio.h>
2: main()
3: {
```

```
4:     int se;
5:     do {
6:         clrscr();
7:         printf("\n      MAIN MENU");
8:         printf("\n      1-INTRODUCTION TO C-LANG.");
9:         printf("\n      2-STRUCTUTE OF C-PROG.");
10:        printf("\n      3-LOOPS");
11:        printf("\n      4-DECISIONS.");
12:        printf("\n      ENTER SELECTION:");
13:        scanf("%d",&se);
14:        switch(se)
15:        {
16:            case 1:
17:                printf("\n 1-INTRODUCTION TO C-LANG.");
18:                break;
19:            case 2:
20:                printf("\n 2-STRUCTUTE OF C-PROG.");
21:                break;
22:            case 3:
23:                printf("\n 3-LOOPS");
24:                break;
25:            case 4:
26:                printf("\n 4-DECISIONS");
27:                break;
28:            default:
29:                printf("\n\n unknown selection");
30:                break;
31:        }
32:    printf("\n\n again(y/n)==>");
33: }while(getch()=='y');
```

شكل ٩-٤ برنامج يستعمل الترکيب switch...case

وعن هذا البرنامج نوضح مايلي:

- السطور من ٨ الى ١٣ تطبع القائمة الرئيسية
- السطر رقم ١٤ يستقبل رقم صحيح ويخزنها في المتغير se
- في السطر رقم ١٥ يختبر المتغير se ويحدد قيمته
- السطر ١٧ يختبر قيمة المتغير se فإذا كانت الرقم ١
- السطر ١٨ يطبع الرسالة التالية 1-INTRODUCTION TO C-LANG
- وهكذا حتى السطر رقم ٢٩ وهو آخر سطر في تركيب Switch ... Case
- السطر ٣٠ يطبع الرسالة > again(y/n)==> والسطر ٣١ يستقبل حرف ويختبره فإذا كان حرف y تعيين الدوارة while تنفيذ البرنامج مرة أخرى وهكذا

المؤثر الشرطي ؟ (Conditional operator)

هذا المؤثر يقوم مقام جملة if

ويأخذ الشكل الآتي :

`var=(condition) ? num1:num2;`

و معناه

`if condition is true then var = num1
if condition is false then var = num2`

أى اذا كان الشرط صحيح فإن `var=num1` واذا كان الشرط غير صحيح فإن

`var=num2`

وهو بديل لتركيب if الذي يأخذ الشكل الآتي :

```
if (condition)  
    var=num1;  
else  
    var=num2;
```

التفرير غير المشروط

التفرير غير المشروط معناه الانتقال إلى مكان محدد داخل البرنامج بدون شرط وتقوم جملة goto بهذا الغرض وتأخذ الشكل العام التالي :

Goto Ib :

حيث Ib متغير من نوع char يشير إلى المكان المطلوب الانتقال إليه ولا ننسى باستخدام هذه الجملة لأنها تستخدم بكثرة مع اللغات الغير تركيبية مثل لغة البيسك أما في حالة اللغة التركيبة (لغة C) فيفضل استخدام الدوال لتغيير مسار تنفيذ البرنامج.

مثال :

البرنامج الموجود بالشكل ١٠-٤ يستخدم جملة goto للتفرير غير المشروط داخل البرنامج وفيه نلاحظ أن البرنامج سيطبع كلمة Allah باستمرار لأن التفرير غير مشروط أى لا يوجد شرط لانهاء تكرار التنفيذ فإذا رغبت في انهاء البرنامج فيجب الضغط على Ctrl+C

```
0: /*Program Name : cs4_10.c */
1: #include <stdio.h>
2: main ()
3: {
4:     char RE;
5:     RE:
6:     printf("it allah");
7:     goto RE;
8: }
```

شكل ١٠-٤ التفرير الغير مشروط باستخدام أمر goto



الفصل الخامس

إنشاء الدوال والماקרו

Functions & Macros

شرحنا في الفصل دوال الادخال والاخراج وهي دوال موجودة داخل اللغة بامكانك انشاء دوال لتسهيل عملك وتبسيط برمجك لتقوم بوظائف محددة على غرار تلك الموجودة داخل اللغة.

في هذا الفصل ستعرف الموضوعات التالية :

- ♦ ما المقصود بالدالة
- ♦ مثال لدالة بسيطة
- ♦ أنواع الدوال
- ♦ معاملات الدوال
- ♦ الماكروز MACROS والفرق بينها وبين الدوال
- ♦ كيفية ربط أكثر من ملف لانشاء ما يسمى بالمشروع
PROJECT

المقصود بالدالة

الدوال التي استخدمناها في الفصول السابقة مثل (printf() أو scanf() دوال مبنية في لغة C وهي دوال عامة يستطيع أي مبرمج استخدامها. من مزايا لغة C المرونة في الاستخدام ولذلك يمكنك إنشاء دوال مثل الدوال القياسية الموجودة في صلب اللغة لتؤدي وظائف مختلفة أو مشابهة والدالة عبارة عن برنامج صغير (أو مجموعة تعليمات تؤدي غرض معين) يخصص لهذا البرنامج اسم ويتم استدعائه داخل الدالة الرئيسية main () بهذا الاسم.

ويتحقق استخدام الدوال مزايا عديدة منها :

1. عدم تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحدة ثم يتم استدعائهما أكثر من مره عند الحاجة إليها
2. بإستخدام الدوال يصبح البرنامج أكثر وضوحا حيث يأخذ البرنامج الشكل التركيبي فيصبح بالشكل الآتي :

```
#include<filename.h>
functions declarations;
main ()
{
    function1_calling();
    function2_calling();
    .....
    .....
}

function1_defination()
{
}
```

```
function2_definition()
```

```
{
```

```
}
```

وبهذا يصبح البرنامج كما ترى اسهل للفهم حيث يتكون من الدالة الرئيسية ومن داخلها يتم استدعاء مجموعه من الدوال وبالتالي يكفي أن تفهم عمل كل دالة لفهم البرنامج كله.

٣. يمكن للمبرمج المتمرس انشاء مكتبه دوال خاصة توفر عليه إعادة كتابة البرامج في كل مرة يحتاج اليها

مثال لدالة بسيطة

يوضح البرنامج الموجود بالشكل رقم (١-٥) كيف يتم الاعلان عن الدالة واستبداعها وكيف تظهر تعليماتها داخل البرنامج.

```
0: /*CS5_1.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: void line2 (void)
4: main ()
5: {
6:     clrscr();
7:     line2();
8:     printf ("***** Allah the god of all world *****\n");
9:     line2(); } /* END OF main() function*/
10: void line2 (void)
11: {
12:     int j;
13:     for (j=0;j<=40;j++)
14:         printf ("%*");
```

```
15:         printf ("\n");
16:     }
```

شكل ٩ - مثال لدالة بسيطة

في البرنامج الموجود بالشكل رقم (١-٥) أنشأنا دالة بالإسم line2() وقد

ظهرت كلمة line2() في ثلاث مواضع :

يسمى بالإعلان عن الدالة function declaration يكون ذلك قبل الدالة الرئيسية main() كما في السطر رقم ٣ ونلاحظ الفاصلة المنقوطة في نهاية هذا الجزء لأنه اعلان، ويكون بالصورة ; void line2(); وكلمة void هي نوع الدالة التي سوف نتكلم عنها فيما بعد بالتفصيل

الموضوع الأول

داخل الدالة الرئيسية : ويظهر في أي مكان داخل الدالة الرئيسية ويسمى function calling أي استدعاء الدالة ويكون بالشكل ; line2(); كما في السطر رقم ٧ وسطر رقم ٩ وفيه يتم كتابة إسم الدالة فقط بدون نوع. وإذا كان لها معاملات نكتب المعاملات

الموضوع الثاني

يكتب بعد إنتهاء الدالة الرئيسية main(). وهذا الجزء يسمى تعريف الدالة Function definition وفيه يتم كتابة محتويات الدالة وتبدأ في البرنامج الذي بين أيدينا من السطر رقم ١٠ باسم الدالة ثم بالقوس { وكأنها برنامج صغير وبعد هذا القوس نبدأ كتابة تعليمات الدالة والدالة هنا عبارة دوارة for تقوم مع الدالة printf() بطباعة العلامة * ٤ مرة وعند استدعاء هذه الدالة يتم تنفيذ هذه السطور أي طباعة العلامة * ٤ مرة.

الموضوع الثالث

ويوضح ذلك من نتيجة التنفيذ التالية :

***** Allah the god of all world *****

ماذا تخيل أن يكون شكل البرنامج السابق بدون دوال
يكون البرنامج كما في الشكل رقم (٥-٢)

```
#include <stdio.h>
#include <conio.h>
main ()
{
    clrscr();
    for (j=0;j<=4;j++)
        printf ("***");
    printf ("\n");
    printf ("***** Allah the god of all world *****");
    for (j=0;j<=4;j++)
        printf ("***");
    printf ("\n");
}
```

شكل رقم ٥-٢ البرنامج بدون الدالة line 2()

في البرنامج الموجود في الشكل رقم ٥-٢ يبدو الفرق بين حجم هذا البرنامج والبرنامج الموجود بالشكل رقم ٥-١ بسيطاً وذلك لأن حجم الدالة صغير إلا أن الفرق بين حالة استخدام الدوال من عدمه يتضح أكثر عند استخدام الدوال الكبيرة. وهكذا نرى أن استخدام الدوال يسهل قراءة البرنامج وفهمه ومراجعته

أنواع الدوال Functions Types

في شرحنا للبرنامج الموجود بالشكل رقم ١-٥ عرفت أن الإعلان عن الدالة

يتم بالصورة التالية :

void line2();

وعرفنا أن كلمة void هي أحد أنواع الدوال وهناك أنواع أخرى من الدوال نوضحها

فيما يلي :

- دوال تعيد قيمة صحيحة int function
- دوال تعيد قيمة حقيقة float function
- دوال تعيد عبارة حرفية string function
- دوال تعيد حرف واحد char function
- دوال لا تعيد أى قيمة void function
- دوال تعيد قيمة من نوع structure وتسمى struct function

ولكن ماذا يعني أن نوع الدالة int أو float أو الخ
لتوضيح ذلك انظر البرنامج الموجود بالشكل رقم (٥-٣) والشرح الوارد بعده.

```
0: /*CS5_3.C*/
1: #include <stdio.h>
2: int sum(int a,int b);
3: main ()
4: {
5:     int z,x=10,y=40;
6:     z=sum(x,y);
7:     printf("\n\n z=%d",z);
8: }
9: int sum(int a,int b)
10: {
```

```

11: int s;
12: s=a+b;
13: return s;
14: }
```

شكل ٥-٣ إنشاء دالة داخل البرنامج

و عن البرنامج الموجود في الشكل رقم (٥-٣) نوضح ما يلى:

- في السطر رقم ٢ تم الإعلان عن دالة بالاسم sum() و سبقت بالكلمة int وهي نوع الدالة و تقابل كلمة void في البرنامج السابق و نلاحظ في الإعلان عن الدالة وجود متغيرين بين الأقواس وهما معلمات الدالة .
- في السطر رقم ٦ يتم استدعاء الدالة وبين أقواسها المتغيران x,y ويستخدمان كمعلمات للدالة ولابد من كتابة معلمات الدالة وذلك لأننا أعلنا عنها بهذه الصورة
- تشمل السطور من رقم ٩ إلى ١٤ على جمل الدالة نفسها و نوضحها فيما يلى:
في السطر رقم ٩ نعرض عن المتغير a بالقيمة الموجودة في المتغير x (وهي ٤٠) وبالمثل نعرض عن المتغير b بالقيمة الموجودة في المتغير y (وهي ٤٠) وفي السطر رقم ١٢ نجمع محتويات كلا من المتغير a والمتغير b ونضع النتيجة في متغير جديد هو s .
- وفي السطر رقم ١٣ نطلب إعادة محتويات المتغير s إلى مكان استدعاء الدالة بإستخدام الكلمة return وهو السطر رقم ٦ . وبهذا تفهم أن جملة z=s الموجودة بالسطر رقم ٦ تعادل الجملة z=sum(x,y)
- ولكن حتى الان لم يتضح معنى أن الدالة من نوع int . معنى نوع الدالة يتضح من القاعدة التي تقول أن نوع الدالة يتوقف على القيمة المرجعة من الدالة . فإذا كانت القيمة المرجعة int (صحيحة) كان نوع الدالة int (صحيحاً) وإذا كانت القيمة المرجعة float (حقيقية) كان نوع الدالة float أما الدالة التي لا تعيد قيمة وبعبارة

أخرى لا تشتمل على جملة `return` فتكون من نوع `void` ومن أمثلتها الدوال الموجودة في لغة C فمثلا دالة `clrscr()` تستخدم لمسح الشاشة فقط وبالتالي لا تعيد قيمة. ولذلك فهي تنتمي إلى النوع `void`.

استدعاء الدالة

يتم استدعاء الدوال اما بمعاملات او بدون معاملات ، و تكون بدون معاملات كما في الدالة `line2()` في البرنامج الموجود بالشكل رقم ٥-١ وبدون معاملات معناه عدم كتابة قيم بين أقواس الدالة ، ويرجع ذلك الى كيفية الاعلان عن الدالة وكيفية تصميمها فالدالة `line2()` تم الاعلان عنها بأنها بدون معاملات ولذلك عند كتابة سطور الدالة لم تستقبل الدالة معاملات من الدالة الرئيسية ويتم إستعمال المعاملات كما في الدالة `sum()` الموجودة في البرنامج الموجود بالشكل رقم ٣-٣ فقد تم الاعلان عنها على أنها تستقبل معاملات

ويوضح البرنامج الموجود بالشكل رقم (٥-٤) فكرة الاستدعاء بمعاملات و البرنامج عبارة عن تعديل الدالة `line2()` الموجودة في البرنامج ٥-١ مع تغير اسمها الى `line3()` و اعطائها معامل من نوع `int`.

```
0: /*CS5_4.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: void line3 (int no);
4: main ()
5: {
8:     clrscr();
9:     line3(30);
10:    printf ("* Allah the god of all world *\n");
11:    line3(70);
12: } /* END OF main() function*/
```

```
13: void line3 (int no)
14: {
15:     int j,no;
16:     for (j=0;j<=no;j++)
17:         printf ("*");
18:     printf ("\n");
19: }
```

شكل ٤-٥ استدعاء الدالة بمعاملات

وفي هذا البرنامج تم الإعلان عن الدالة line3() في السطر رقم ٣ ونجد بين القوسين (int no) ومعناه أن للدالة معامل واحد من نوع صحيح وهو no ويقوم كلا من السطر رقم ٩ والسطر رقم ١١ بإستدعاء الدالة ;() line3 ولكن كل مره يتم إرسال قيمه مختلفه لالمعامل ففي المرة الأولى في السطر ٩ تكون القيمه هي ٣٠ وفي المرة الثانية في السطر رقم ١١ تكون القيمة هي ٧٠ وبمعنى أنتا عوضنا عن المتغير no أول مره بالقيمة ٣٠ وبالتالي تطبع الدالة العلامة * ٣٠ مرة وثاني مره. بالقيمة ٧٠ وبالتالي تطبع الدالة العلامة * ٧٠ مرة وهذا ما تلاحظه من نتيجة التنفيذ التالية :

* Allah the god of all world *

استدعاء الدالة بمتغيرات

في البرنامج السابق تم استدعاء الدالة line3() بمعاملات من نوع قيم ثابتة موجودة بالبرنامج نفسه. بالإضافة الي ذلك يمكن أن تكون هذه المعاملات متغيرات تستقبل قيمها من المستخدم أو من داخل البرنامج ، وهذا مفيد عندما نريد استقبال قيمة هذه المتغيرات من المستخدم بدلاً من تحديدها داخل البرنامج. لأنها تتغير حسب الحالة

وبالتالي تعطى مرونة في التعامل مع البرنامج ويتحقق ذلك من البرنامج الموجود بالشكل

٥-٥

```
0: /*CS5_5.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: void line4 (int no);
4: main ()
5: {
6:     int k;
7:     clrscr();
8:     do {
9:         printf ("\***** Enter Length of line(-ve to exit) :");
10:        scanf ("%d",&k);
11:        line4(k);
12:    } while(k>0);
13: }
14: void line4 (int no)
15: {
16:     int j;
17:     for (j=1;j<=no;j++)
18:         printf ("*");
19:     printf ("\n");
20: }
```

شكل ٥-٥ إستدعاء الدالة بمتغيرات

في هذا البرنامج تم الإعلان عن دالة بالاسم (line4)

في السطر رقم ١٠ يتم استقبال قيمة عبارة عن رقم صحيح وتخزينه في المتغير K وفي السطر رقم ١١ يتم استدعاء الدالة بالقيمة المخزنة في المتغير K وهذه القيمة المخزنة

في المتغير k تغير حسب البيانات التي يدخلها المستخدم ويوضح من ذلك نتيجة التنفيذ التالية :

***** Enter Length of line(-ve to exit) :30

***** Enter Length of line(-ve to exit) :60

***** Enter Length of line(-ve to exit) :-5

أمثلة مختلفة على أنواع الدوال

فيما يلى سووضح أمثلة على أنواع الدوال من خلال ٣ برمج

مثال للدالة من نوع Void

البرناموج الموجود بالشكل رقم ٥-٦ يقوم بانشاء دالة بالاسم (twobEEP)

تقوم هذه الدالة بعزف نغمة بتردد متغير وهذه الدالة من نوع void ولانا لانحتاج من

هذه الدالة أن ترجع لنا أى قيمة فليس بها جملة return.

```
0: /*CS5_6.C*/
0: #include <dos.h>
1: #include <stdio.h>
2: #include <conio.h>
3: void twobEEP(void);
4: main ()
5: {
6:     twobEEP();
7:     printf ("\n WELCOME WITH C-PROGRAMMING :");
8:     twobEEP();
```

```

9:      }
10: void twobeep(void)
11: {
12:     int j;
13:     for (j=0;j<10000;j++)
14:     {
15:         sound(j*10);
16:         delay(5);
17:         nosound();
18:     }
19: }

```

شكل ٦-٥ برنامج دالة من النوع void

تشتمل السطور من ١٠ الى ١٩ على الدالة twobeep() وتستخدم داللين من دوال لغة C الاولى sound() وتستخدم لاخراج نفمة بتردد يتوقف على الرقم المكتوب بين القوسين. والثانية delay() لعمل فاصل زمني بين النفمة والتي تليها وتتوقف مدة هذا الفاصل الزمني على الرقم المكتوب بين القوسين.

أكتب البرنامج ونفذة وتابع النتيجة وهي سماع صوت من سماعة جهاز الكمبيوتر

مثال لدالة من نوع int

البرنامج الموجود بالشكل رقم ٦-٧ مثال لبرنامج يستعمل دالة من نوع int وفي هذه الدالة يتم ارسال قيمة صحيحة لها وتقوم الدالة بحساب مربع القيمة واعادة حاصل الضرب الى الدالة الرئيسية عند الاستدعاء وبالمثل دالة تحسب مكعب قيمة وتعيدها عند الاستدعاء وبالتالي تكون القيم المرتجلة في الداللين من نوع صحيح وهو نوع الداللين.

```

0: /*CS5_7.C*/
1: #include <stdio.h>

```

التحول النسبي Macro Functions والماקרו

```
2: int sqr(int a);
3: int qup(int q);
4: main ()
5: {
6:     int s,qu,no=10;
7:     s=sqr(no);
8:     qu=qup(no);
9:     printf("\n square of no=%d",s);
10:    printf("\n qupic of no=%d",qu);
11: }

12: int sqr(int a)
13: {
14:     int v1;
15:     v1=a*a;
16:     return v1;
17: }

18: int qup(int q)
19: {
20:     int v2;
21:     v2=q*q*q;
22:     return v2;
23: }
```

شكل رقم ٥-٧ برنامج لدالة من النوع int

نتيجة التفيد

square of no =100

qupic of no =1000

مثال لدالة من نوع float

في الشكل رقم (٥-٨) مثال لبرنامج يحتوى على دالة من نوع float تقوم بجمع رقمين واعادة النتيجة الى الدالة الرئيسية عند الاستدعاء والقيمة المرجعة قيمة حقيقية (float) وبالتالي يكون نوع الدالة float

```
0: /*CS5_8.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: float add(float x ,float y);
4: main ()
5: {
6:     float no1,no2;
7:     printf ("\n Enter no1,no2:");
8:     scanf ("%f,%f",&no1,&no2);
9:     printf ("\n addation of Numbers is %f",add(no1, no2));
10: }
11:
12: float add (float x,float y)
13: {
14:     float yt;
15:     yt=x+y;
16:     return yt;
17: }
```

شكل ٤-٨ برنامج لدالة من النوع float

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

Ente no1,no2: 3.2,4.3

addation of squares is 6.5

معاملات الدالة الرئيسية main()

حتى الآن تلاحظ أننا نكتب الدالة الرئيسية main() بدون معاملات. وبالتالي نحصل على الصورة القابلة للتنفيذ من البرنامج وتنفذ بدون معاملات. بعبارة أخرى يكفي ان تكتب اسم البرنامج تحت ممح نظام التشغيل DOS لاستدعائه للتنفيذ. وللوضيح ذلك أكتب البرنامج الموجود بالشكل ٥-٩ وقم بتنفيذه ثم اخرج الى بيئه نظام التشغيل DOS ثم اكتب أمر dir نجد أن هذا الملف له ثلاث نسخ هي cs5-9.obj cs5-9.c cs5-9.c. النسخة القابلة للتنفيذ هي cs5-9.exe ولذلك يكفي أن تكتب اسم البرنامج لتنفيذ البرنامج الا هناك حالات تتطلب ادخال معاملات لوجه البرنامج لغرض معين.

فمثلا الأمر format يتطلب ذكر اسم المشغل المطلوب تشكيلاه بعد كتابة

الامر وهذا اسم المشغل يعتبر معامل للبرنامج :

```
/*CS5_9.C*/  
#include <stdio.h>  
main ()  
{  
    printf("In Welcome Mr.");  
}
```

شكل ٥-٩ برنامج يستخدم دالة الطباعة

فكيف لنا أن ننفذ البرنامج السابق cs5_9 بنفس الاسلوب بحيث نرسل له معامل ويقوم البرنامج باستقبال هذا المعامل واستعماله؟

نفرض أن المطلوب هو الحصول على النتيجة التالية :

Welcome Mr : Mohamed

في هذه الحالة يجب أن نرسل للبرنامج معامل هو كلمة Mohamed بالإضافة إلى اسم البرنامج ويشتمل البرنامج الموجود في الشكل رقم (٥-١٠) على البرنامج اللازم لهذا الغرض

```

0: /*CS5_10.C*/
1: #include<stdio.h>
2: main (int argc,char *argv[ ])
3: {
4:     if(argc!=2)
5:     {
6:         printf("\n\n Error in Arguments");
7:         exit(1);
8:     }
9:     printf("\n Welcome Mr.%s",argv[1]);
10: }
```

شكل ٥-١ برنامج لاستقبال معامل من خارجه عند التنفيذ

وعند النظر إلى هذا البرنامج لأول مرة تلاحظ اختلاف شكل الدالة (main) عما تعودت عليه في المرات السابقة ، حيث يظهر داخل أقواس الدالة الرئيسية معاملين : الاول هو int argc ، والثاني هو [char *argv[]] ، والاول هو متغير من نوع صحيح عادي ويمكن أن يأخذ أي اسم ولكننا اختربنا هذا الاسم لأنه اختصار الكلمة argument count أي عدد المعاملات وقد صممت الدالة الرئيسية (main) بحيث تضع في المعامل الأول عدد المتغيرات التي يكتبها المستخدم فمثلاً : لو كتبت الصيغة

copy a:file1.ext b:

تجد أن هذا السطر به ثلاث متغيرات الأول اسم البرنامج copy ، والثاني هو المعامل الأول للبرنامج وهو إسم الملف a:file.ext ، والثالث هو b ، إذن عدد المتغيرات التي كتبها المستخدم ثلاثة ويخزن هذا الرقم في المتغير argc وفائدته تحديد

عدد المتغيرات التي سيقبلها البرنامج ساعة التنفيذ وبهذه الطريقة نستطيع جعل البرنامج يؤخذ أكثر من وظيفة وكل وظيفة تتوقف على عدد المعاملات التي يكتبها المستخدم.
و بهذه الرقمن تقوم الدالة الرئيسية بحجز مصفوفة من العبارات الحرفية (أى مصفوفة عناصرها كلمات Strings).

عدد هذه العناصر هو argc واسم هذه المصفوفة هو المعامل الثاني للدالة main() وهو argv ويتم تخزين المتغيرات التي يدخلها المستخدم في هذه المصفوفة.

ماذا لو أردت كتابة برنامج وأردت تحديد اسم البرنامج بحيث لا يسمح لأحد بتنفيذ هذا البرنامج باسم آخر بالرغم من أن أي برنامج تشفى لا يستطيع تغيير اسمه واستعماله كما شاء؟

ستجد الحل في القرص المصاحب للكتاب تحت اسم solv1e.c

والآن نتابع شرح البرنامج

في السطر رقم 4 جملة if تقول اذا كان المتغير argc لا يساوى 2 أى اذا كان عدد المعاملات التي أدخلها المستخدم لا يساوى 2 أى اذا لم يكتب المستخدم الصورة المطلوبة لتنفيذ البرنامج وهي

CS5_10 Mohamed

تظهر العبارة التالية

Error in Arguments

وينتهي عمل البرنامج بالدالة exit(1)

وفي السطر رقم 9 اذا أدخل المستخدم الصورة الصحيحة تظهر الرسالة التالية

Welcome Mr. Mohamed

والسؤال من أين أتى البرنامج بكلمة Mohamed ؟

الإجابة عندما يكتب المستخدم الصورة

CS5_10 Mohamed

يوضع اسم البرنامج CS5_10 في العنصر الأول من المصفوفة أي [0] Argr

وكلمة Mohamed في العنصر الثاني من المصفوفة أي [1] Argr

الماكرو (MACROS)

ما المقصود بالماкро ؟

هو مجموعة تعليمات تؤدي غرض معين ويشبه إلى حد كبير الدالة ، ويتم إنشاؤه مرة واحدة وبعد ذلك يمكنك استدعائه كلما احتجت إليه.

وأقبل أن نسأل ما الفرق بينه وبين الدالة تعالى بما نرى أولاً كيفية إنشائه واستعماله ثم نناقش بعد ذلك الفرق ثم نوضح متى نستخدم الماكرو ومتى نستخدم الدالة.

كيفية إنشاء الماكرو

يتم ذلك باستعمال الكلمة #define وهذه الكلمة تسمى directive أو preprocessor ومعناها توجيه.

ولإنشاء الماكرو تستخدم الصورة التالية :

#define macro line

وهي عبارة عن تعريف طرف بطرف مثل #define A 5 ومعناها عرف المتغير

A بالقيمة 5

والبرنامج الموجود بالشكل رقم (٥-١١) يوضح لنا كيفية الاعلان عن الماكرو وكيفية استعماله

```
0: /*CS5_11.C*/
1: #define sum(a,b) a+b
2: #define mul(x,y) x*y
3: #include <stdio.h>
4: main ()
5: {
6:     int v1=5,v2=10;
7:     printf("\n\n sum(v1,v2)=%d",sum(v1,v2));
8:     printf("\n\n mul(v1,v2)=%d",mul(v1,v2));
9: }
```

الشكل ٥-١١ الاعلان عن الماكرو واستعماله

وعند تنفيذ هذا البرنامج ستحصل على النتيجة التالية :

sum(v1,v2)=15
mul(v1,v2)=50

وعن هذا البرنامج نوضح ما يلى :

في السطر رقم ١ استخدمنا كلمة define لتعريف ماكرو بالاسم sum ووظيفته استبدال المتغيرين a,b بالصورة a+b وبالمثل في السطر رقم ٢ يستبدل المتغيرين x,y بنتيجة الضرب y*x و معناها كلما قابل مترجم اللغة الطرف الأول للماكرو يستبدل بالطرف الثاني.

ونلاحظ في هذا المثال أن المتغيرين a,b يمكن استبدالهما بأى متغيرين أو قيمتين داخل البرنامج واسم الماكرو هو الذى يحدد العملية التى يقوم بها الماكرو هل هي عملية جمع أم ضرب بناءً على المعادلة الموجودة في الطرف الایمن من الماكرو

في السطر رقم ٧ استخدمنا الماكرو sum لجمع متغيرين هما v1,v2 وبالمثل
السطر رقم ٨.

ما الفرق بين الماكرو وبين الدالة ومتى نستخدم الماكرو ومتى نستخدم
الدالة؟

من العروف أن التعامل مع البرنامج يمر بثلاث مراحل :

١. كتابة البرنامج وهذا ما نسميه source code ويخصص لملف المصدر
الامتداد C.
٢. ترجمة البرنامج للغة يفهمها الحاسوب ويسمى compilation ويخصص للملف
الامتداد obj.
٣. ربط ملف object بمكتبات اللغة ليصبح قابل للتنفيذ وتسمى هذه العملية
ويخصص للملف linking امتداد exe.

ويظهر الفرق بين الماكرو وبين الدالة في هذه المراحل كما يلى
في مرحلة الكتابة ليس هناك فرق بين الماكرو والدالة. في مرحلة الترجمة
(compilation). يتم تحويل تعليمات الدالة الى لغة الالة (object) وتنتظر مرحلة الربط
ولاتنفذ الدالة الا في مرحلة الربط. أما في حالة الماكرو يتم استبدال الماكرو بنتيجة
تنفيذ الماكرو وبالرجوع الى البرنامج الموجود بالشكل رقم ٥-١١ يتم استبدال
الماكرو الموجود بالسطر رقم ٧ بنتيجة التنفيذ مباشرة أي يتم وضع القيمة ١٥ التي هي
نتيجة تنفيذ الماكرو مكان sum(v1,v2) وبالتالي عندما تأتي مرحلة التنفيذ يجد البرنامج
نتيجة تنفيذ الماكرو وهي ١٥ وبالمثل يمكن أن تفهم الماكرو الموجود في السطر

رقم ٨

وهكذا نرى أن للماكرو مزايا منها :

أنه بسيط في إنشاء بسيط في الاستعمال ويعطي في النهاية ملف تنفيذى أصغر من المقابل له باستعمال الدوال فإذا كانت العملية المطلوبة عمل دالة لها بسيطة ويمكن كتابتها في سطر واحد نستعمل الماקרו أما إذا كانت تحتاج أكثر من سطر نستخدم الدالة.

المشروع Project

إذا كانت لك خبرة بقواعد البيانات DBASE أو CLIPPER ، فأنك تعلم أن إنشاء تطبيق متكامل يتم عن طريق عمل ملف رئيسي (برنامج رئيسي) وبرامج فرعية . ويتم استدعاء البرامج الفرعية من البرنامج الرئيسي وذلك من داخل البرنامج الرئيسي بالأمر do أى تقسيم البرنامج إلى برنامج رئيسي وبرامج فرعية .

والسؤال كيف يتم ذلك في لغة C ؟

يتم ذلك بأكثر من طريقة

إذا كان البرنامج الكلى صغيرا ولا يتطلب أكثر من ملف يكفى استعمال الدوال كما مر بما في البرامج السابقة أى نشئ الدالة الرئيسية main() وبداخلها يتم استدعاء الدوال الأخرى .

إذا كان البرنامج كبير بحيث يتطلب أكثر من ملف فهناك طريقتين لتنظيم ذلك الطريقة الأولى هي طريقة تقليدية اجتهادية لاستعمال كثيرة وهي أن نكتب البرنامج الرئيسي وبه الدالة الرئيسية في ملف منفرد ونكتب البرامج الفرعية في صورة دوال ونضع هذه الدوال في ملف منفصل . ويتم استدعاء هذه الدوال من داخل الدالة الرئيسية ولكن يتم ربط الملف الثاني الذي به الدوال بالملف الأول نكتب اسم الملف الثاني مع

كلمة `#include` بالصورة `<file2.c>` في بداية الملف الأول ولتوسيع ذلك نسوق المثال التالي :

في هذا المثال ستتجدد ملفين الأول اسمه `calc.c` وهو موجود بالشكل رقم ٥-١٢ والملف الثاني اسمه `tools.c` موجود بالشكل رقم ٥-١٣

```
0: /*Program Name CALC.C*/
1: #include "tools.c"
2: #include <stdio.h>
3: int sum(int a,int b);
4: int mul(int x,int y);
5: main ()
6: {
7:     int no1=5,no2=10;
8:     printf("\n\n no1+no2=%d",sum(no1,no2));
9:     printf("\n\n no1*no2=%d",mul(no1,no2));
10: }
```

شكل رقم ٥-١٢ ملف البرنامج الرئيسي `calc.c`

وتلاحظ في بداية البرنامج الأول الموجود في الشكل رقم ٥-١٢ التوجيه `#include "tools.c"` ومعناه اربط الملف `tools.c` مع هذا الملف في مرحلة الترجمة وبالتالي تصبح الدوال الموجودة بالملف `tools.c` كأنها بالملف `calc.c`

```
0: /*Program Name TOOLS.C*/
1: int sum (int a,int b)
2: {
3:     int yt;
4:     yt=a+b;
5:     return yt;
6: }
```

```
7: int mul (int x,int y)
8: {
9:     int k;
10:    k=x*y;
11:    return k;
21: }
```

شكل رقم ٥-١٣ ملف الدوال tools.c

الطريقة الثانية وهى الطريقة المتبعة في البرامج الكبيرة وهي انشاء مشروع (project) هذا المشروع عبارة عن برنامج رئيسى يشتمل على دالة رئيسية () main() وبرامج فرعية تشتمل فقط على دوال ولا تحتوى على دالة main() ويتم ذلك من خلال بيئة كتابة برامج لغة C ولتطبيق ذلك على الملفين السابقين CALC.C,TOOLS.C.

اتبع الخطوات الآتية

١. افتح ملف البرنامج calc.c (راجع شكل ٥-١٢)
 ٢. احذف السطر رقم ١ من البرنامج ثم احفظ الملف
 ٣. اضغط مفتاحي alt+p تظهر قائمة project
 ٤. من قائمة project اختر new تظهر نافذة صغيرة تكتب فيها اسم المشروع
 ٥. اكتب اسم لهذا المشروع وليكن main ثم اضغط مفتاح الادخال
 ٦. اضغط مفتاح insert تظهر قائمة بها اسماء الملفات الموجودة عندك
 ٧. اختر الملف tools.c ثم الملف calc.c
 ٨. اضغط مفتاح Esc بعد الانتهاء من اختيار ملف المشروع
 ٩. اضغط مفتاحي Ctrl+F9 لتنفيذ المشروع
- وبهذا تحصل على ملف تنفيذى باسم المشروع وهو main

مثال

نختتم هذا الفصل ببرنامج متكامل يجمع معظم المفاهيم التي شرحتها حتى الآن (انظر شكل رقم ١٤-٥) وهذا البرنامج يقوم بطباعة قائمة اختيارات على الشاشة عبارة عن عمليات الجمع والطرح والقسمة والضرب ويطلب من المستخدم تحديد اختيار وعندما يقوم المستخدم بتحديد اختيار يقوم البرنامج بتنفيذ هذا الاختيار.

اكتب البرنامج أو فتحه من القرص المصاحف للكتاب ثم نفذه وشاهد نتيجة التنفيذ.

```
/*CS5_14.C*/
#include <stdio.h>
#include <conio.h>
void add(void);
void sub(void);
void mul(void);
void div(void);
main ()
{
    char ch='0';
    while (ch!='5')
    {
        /* draw menu */
        printf ("\n***** main menu *****\n");
        printf ("***** (1)addation....");
        printf ("***** (2)sub.....");
        printf ("***** (3)mul.....");
        printf ("***** (4)Div.....");
        printf ("***** (5)exit.....");
        printf ("\n\n Enter selection :");
        ch=getch();
```

الفصل الخامس : الدوال و الماكرو Functions Macro

```
switch (ch)
{
    case '1':
        add();
        break;
    case '2':
        sub();
        break;
    case '3':
        mul();
        break;
    case '4':
        div();
        break;
    case '5':
        ch='5';
        break;
    default:
        printf ("\n unknown operator");
}
}

void add()
{
    float no1,no2;
    char op;
    clrscr();
    printf ("\n enter no1,op,no2");
    scanf ("%f%c%f",&no,&op,&no2);
    printf ("\n sum= %f",no1+no2);
}
```

```

void sub()
{
    float no1,no2;
    char op;
    clrscr();
    printf ("\n enter no1,op,no2");
    scanf ("%f%c%f",&no1,&op,&no2);
    printf ("\n sub= %f",no1-no2);
}

void mul()
{
    float no1,no2;
    char op;
    clrscr();
    printf ("\n enter no1,op,no2");
    scanf ("%f%c%f",&no1,&op,&no2);
    printf ("\n mul= %f",no1*no2);
}

void div()
{
    float no1,no2;
    char op;
    clrscr();
    printf ("\n enter no1,op,no2");
    scanf ("%f%c%f",&no1,&op,&no2);
    printf ("\n div= %f",no1/no2);
}

```

شكل رقم ٥-١٤ برنامج قائمة اختيارات يستخدم الدوال



الفصل السادس

المصفوفات ARRAYS

- ♦ في هذا الفصل نتناول الموضوعات التالية
- ♦ معنى المصفوفات وأنواعها
- ♦ كيفية التعامل مع المصفوفة ذات البعد الواحد
- ♦ المصفوفة ذات بعدين
- ♦ ارسال مصفوفة إلى دالة كمعامل
- ♦ المصفوفات وسلسلة الحروف (الكلمات) & **String**
- ♦ دوال العبارات الحرفية **string functions**
- ♦ أوامر المترجم **preprocessors**

معنى المصفوفات

تقسم البيانات إلى بيانات حرفية (char) وبيانات رقمية (int) وبيانات حقيقية (float) وتسمى هذه الأنواع (int,float,char) بالأنواع الرئيسية للبيانات ، حيث لا يمكن تجزئتها أقل من ذلك.

ولكن هناك أنواع أخرى من البيانات تسمى بالأنواع المشقة (data types) من هذه الأنواع المصفوفات Arrays. تعرف المصفوفة بأنها مجموعة من العناصر تتسمى إلى نوع واحد. وينحصر لها اسم واحد وتنقسم المصفوفات إلى مصفوفات ذات بعد واحد ومصفوفات ذات بعدين

والمصفوفة ذات البعد الواحد مثل :

$$A = [3 \ 4 \ 5 \ 7 \ 9]$$

وتسمى مصفوفة ذات بعد واحد لأنها تكون من صف واحد أو عمود واحد، وفيها حرف A هو اسم المصفوفة ، والارقام هي عناصر المصفوفة ويتم الاشارة إلى كل عنصر برقم العنصر أى بترتيبه داخل المصفوفة على أن يبدأ العدد بالرقم صفر كما يلى العنصر $A[0]$ يساوى 3 و العنصر $A[1]$ يساوى 4 و العنصر $A[2]$ يساوى 5 وهكذا ..

والمصفوفة ذات البعدين تأخذ الشكل التالي :

$$C = \begin{bmatrix} 5 & 4 & 2 \\ 7 & 1 & 7 \\ 2 & 9 & 5 \end{bmatrix}$$

وتسمى هذه المصفوفة 3×3 أي 3 صفوف و 3 أعمدة ويسم الاشارة الى عناصر المصفوفة برقم الصف ورقم العمود الذي يقع عندهما العنصر كما يلى.

العنصر	C [0] [0]	يساوى ٥
العنصر	C [0] [1]	يساوى ٤
العنصر	C [0] [2]	يساوى ٢
العنصر	C [2] [1]	يساوى ٩
وهكذا ..		

والخلاصة أن المصفوفة هي مجموعة من العناصر سواء ذات بعد واحد أو بعدين بشرط أن تكون جميع العناصر من نوع واحد وفيما يلى سنوضح كيفية الاعلان عن المصفوفة وكيفية التعامل مع عناصرها

المصفوفة ذات البعد الواحد

البرنامج الموجود بالشكل رقم (٦-١) يوضح التعامل مع المصفوفة ذات البعد الواحد وفيه يتم الاعلان عن المصفوفة واستقبال عناصر المصفوفة من المستخدم واضافة قيمة صحيحة الى كل عنصر من عناصر المصفوفة ثم طباعة عناصر المصفوفة كما يتضح ذلك من نتيجة التنفيذ

```

0: /*Program Name CS6_1.C*/
1: #include<stdio.h>
2: main ()
3: {
4:     int A[10];
5:     int i;
6:     for (i=0;i<10;i++)
7:     {
8:         printf ("\n A[%d]=",i);
9:         scanf("%d",&A[i]);

```

```
10:     A[i]= A[i]+5;
11: }
12: for (i=0; i<10; i++)
13: printf (" \n A[%d]=%d",i,A[i]);
14: /* main () function -----*/
```

شكل رقم (٦-١) التعامل مع مصفوفة ذات بعد واحد

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

A[0]=5

A[1]=7

A[2]=56

.....

.....

A[0]=10

A[1]=12

A[2]=61

.....

.....

وعن هذا البرنامج نوضح مايلي :

• في السطر رقم ٤ اعلان عن مصفوفة عدد عناصرها ١٠ عناصر ونوع هذه

العناصر int واسم هذه المصفوفة هو الحرف A وطريقة الاعلان عن المصفوفة

بسيئة كما لو كنت تعلن عن متغير واحد ولكنك تضيف عدد عناصر المصفوفة

إلى هذا المتغير .

• في السطر رقم ٦ إستخدمنا دوارة for لاستقبال عناصر المصفوفة ولا حظ

استخدام دوارة for مع المصفوفة فإذا لم تستعمل دوارة for وكان عدد عناصر

المصفوفة ١٠ عناصر فلابد من كتابة السطرين التاليين عشرة مرات

لاستقبال عناصر المصفوفة

```
printf ("\n A[0] =");  
scanf ("%d", & A[0]);
```

وهكذا - عشرة مرات - وهذا غير معقول لذلك نستخدم دوارة for كما يلى :

```
for (i=0;i<10;i++)  
{  
    printf ("\n A[%d] =", i);  
    scanf ("%d", & A[i]);  
}
```

لابد من استعمال الدوارة for مع المصفوفات

- في السطر رقم 8 دالة printf تطبع العبارة A[0]
- في السطر رقم 9 دالة scanf تستقبل الرقم الذي يدخله المستخدم وتخزنه في عنصر المصفوفة A[i].

ونظرا لأن المتغير A يبدأ بالقيمة صفر فان القيمة المدخلة تخزن في العنصر الأول من المصفوفة والمشار اليه بالصورة A [0] ثم تزداد قيمة A وبالتالي تتوالى عناصر المصفوفة و تمتليء بالترتيب

يشار لأول عنصر في المصفوفة بالرقم صفر هكذا A[0]

في السطر رقم 12 دوارة for أخرى لطباعة عناصر المصفوفة بعد اضافة الرقم 5 الى كل عنصر

اعطاء قيمة ابتدائية لعناصر المصفوفة

من الممكن الاعلان عن المتغير واعطائه قيمة ابتدائية بالشكل ا لاتي

```
int A=5;
```

وهذا اعلان عن متغير صحيح وفي نفس الوقت اعطائه قيمة ابتدائية

وبنفس الأسلوب يمكن الإعلان عن المصفوفة واعطائها قيم ابتدائية كما يلى
int A[3] = {5,7,9} ;

char name [10] = { 'c', 'b' , 't' , 'r', -----} ;

وهذا معناه اعطاء قيم ابتدائية لعناصر المصفوفة وهو الأفضل كلما استطعت ذلك حتى لا يقوم البرنامج بتخزين قيم عشوائية من الذاكرة في عناصر المصفوفة وحتى لاتطبع قيم ليس لها معنى

المصفوفة الغير محددة العدد

المقصود بها هو عدم تحديد عدد العناصر في حالة الإعلان وتأخذ الصورة الآتية

int A[]={3,4,5} ;

أو

char name []= " abdef ";

وتحديد عدد عناصر هذه المصفوفة في هذه الحالة يتم من خلال المترجم عن طريق عدد العناصر في الطرف اليمين وحجز مصفوفة بهذا العدد

فمثلا ; int A[]={3,4,5} : معناه أن المصفوفة [] A عدد عناصرها ٣ عناصر وهكذا وهذا لا يصلح الا اذا كنت ستعطي عناصر المصفوفة قيم ابتدائية ولكن لا يصح أن تعلن عن مصفوفة غير محددة العدد ثم تستعملها في استقبال قيم من المستخدم فمثلا لا يصح أن تقول [] a ثم تستقبل عناصر المصفوفة a من المستخدم.

مثال

يقوم البرنامج الموجود بالشكل (٦-٢) باستقبال الاسم ثم يطبعه بحيث تكون الحروف معكوسه كما يظهر ذلك من نتيجة التنفيذ.

```
0: /*Program Name CS6_2.C*/
1: #include <stdio.h>
2: main ()
```

```

3: {
4:     int i, c=0;
5:     char name [20];
6:     clrscr ();
7:     printf (" \n Enter your name :");
8:     while (name [c]= getche () ) !=' \r'
9:         c++;
10:    printf(" \n your name in reverse is :");
11:    for (i=c ;i>= 0; i--)
12:        printf ("%c", name [i]);
13 }

```

شكل رقم ٦ استقبال اسم وعكس حروف

وعن هذا البرنامج نوضح مايلي :

في السطر رقم ٥ اعلان عن مصفوفة حروف علذ عناصرها ٢٠ كحد أقصى
ويشتمل السطر رقم ٨ على الدالة getche() التي تستقبل حرف وتتخزنه في عنصر
المصفوفة name[c] و ٥ لها قيمة ابتدائية صفر اي تخزنه في أول عنصر وتحتبر جملة
هذا الحرف فإذا لم يكن هو مفتاح الادخال تستمر في تنفيذ الجمل التالية لها

السطر رقم ٩ يزيد قيمة العدد c بمقدار واحد بالصورة ++c++ ويظل البرنامج يستقبل
حرف ويزيد قيمة العدد c بمقدار واحد كلما كتب المستخدم حرفًا وطالما لم يضغط
المستخدم على مفتاح الادخال Enter

لاحظ أن الدالة getche() لاحتياج الى ضغط مفتاح Enter لذلك يستمر
الاستقبال في سطر واحد كعبارة حرفية.

- في السطر رقم ١١ الدالة for تبدأ من آخر عنصر تم ادخاله وهو آخر قيمة
للمتغير c وتنتهي عند أول عنصر ورقمها صفر.

- في السطر رقم ١٢ يتم طباعة عناصر المصفوفة بالعكس وذلك لأننا من آخر عنصر ادخل حتى أول عنصر وعند تنفيذ البرنامج تحصل على النتيجة التالية :

Enter your name:SAMY
your name in reverse is:YMAS

المصفوفة ذات البعدين

هي المصفوفة التي ترتب عناصرها في شكل صافوف واعمدة ويتم الإعلان عنها بالشكل التالي :

int A [5] [10];

وهذا معناه أن المصفوفة A. مصفوفة ذات بعدين ، ٥ صفوف و ١٠ اعمدة و يتم الاشارة الى العنصر برقم الصف و رقم العمود

ويجب الانتباه الى أنه عندما تستخدم مصفوفة لأبد من استعمال دوارة for ويوضح ذلك من المثال الذي ذكرناه في المصفوفة ذات البعد الواحد وأما في حالة المصفوفة ذات البعدين فلابد من استعمال ما يسمى بالدوارات المتداخلة nested loops.

وهذا مثلاً من خلال البرنامج الموجود بالشكل رقم (٦-٣) حيث يقوم باستقبال مجموعة قيم ويختزنه في مصفوفة ذات بعدين ثم يقوم بطباعة هذه القيم في شكل مصفوفة كما في الشكل رقم (٦-٤)

```
0: /*Program Name CS6_3.C*/  
1: #include <stdio.h>  
2: main()  
3: {  
4:     int r,c;  
5:     int A[3][4];
```

```
6:     for (r=0;r<3; r++)
7:     { printf ("\n");
8:     for (c=0;c<4;c++)
9:     {
10:        printf ("\t A[%d] [%d]=", r,c);
11:        scanf (" %d",& A[r][c] );
12:    }
13:    }
14:    for (r=0; r<3;r++)
15:    {
16:        printf ("\n");
17:        for (c=0; c<4; c++ )
18:            printf ("\t %d ", A[r][c] );
19:    }
20: }
```

شكل ٦ التعامل مع مصفوفة ذات بعدين

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج في السطر رقم ٥ بالاعلان عن مصفوفة ذات بعدين وذلك بالشكل `int A[3][4]` وهي مصفوفة تكون من ٣ صفوف و ٤ أعمدة من نوع صحيح.
- في السطر رقم ٦ الدوارة `for` للتكرار ٣ مرات والسطر رقم ٨ للتكرار ٤ مرات وبالتالي كل قيمة للمتغير `r` تقابلها ٤ قيم لمتغير `C` وهو ما تم دراسته في الفصل الخاص بالدوارات.
- في السطر رقم ١١ الدالة `scanf()` تستقبل العنصر رقم `[r][c]` وهو يتدرج من `[0][0]` ، `[1][0]` ، `[2][0]` حتى يستقبل جميع عناصر المصفوفة.
- ويشتمل كلا من السطر رقم ١٤ والسطر رقم ١٧ على دوارة `for` وهي تستخدم نفس الفكرة السابقة والسطر رقم ١٨ لطباعة عناصر المصفوفة.

وعند تفريذ هذا البرنامج نحصل على النتيجة التالية :

A[0] [0]=55

A[0] [1]=66

A[0] [2]=77

A[0] [3]=88

A[1] [0]=99

A[1] [1]=44

A[1] [2]=33

A[1] [3]=22

.....

.....

55 66 77 88

99 44 33 22

11 22 43 54

شكل (٤-٦) نتيجة استقبال وطباعة مصفوفة ذات البعدين

اعطاء قيم ابتدائية للمصفوفة ذات بعدين :

كما يمكن اعطاء قيم ابتدائية للمصفوفة ذات البعد الواحد يمكن كذلك اعطاء

قيم ابتدائية للمصفوفة ذات البعدين ويكون بالشكل الاتي :

```
int A[ 3 ][ 4 ] = {  
    { 4,5,7,8},  
    {3,2,4,5},  
    {7,9,8,1} };
```

وفي هذا الشكل يأخذ العنصر رقم 0,0 القيمة 4، والعنصر رقم 0,1 القيمة 5 وهكذا

مثال : (برنامج ورقة رسم بياني)

البرنامج الموجود بالشكل رقم (٦-٥) يطبع مصفوفة من النقط (٠) على

الشاشة ثم يستقبل من المستخدم قيمة الصاف والعمود ثم يوقع هذه النقطه على الشاشه

في شكل الحرف * . فإذا أدخل المستخدم قيمة سالبة ينتهي البرنامج وهذا البرنامج يسائل ورقة رسم بياني كما يظهر ذلك في نتيجة التنفيذ الموجودة بالشكل رقم (٦-٦).

```
0: /*Program Name CS6_5.C*/
1: #define rows 10
2: #define cols 70
3: main ()
4: {
5:     int x,y;
6:     char matrix[rows][cols];
7:     for (y=0;y<rows;y++)
8:         for (x=0;x<cols;x++)
9:             matrix[y][x]='.';
10:
11:    do
12:    {
13:        clrscr();
14:        for (y=0;y<rows;y++)
15:        {
16:            printf ("\n");
17:            for (x=0;x<cols;x++)
18:                printf("%c",matrix [y][x] );
19:            printf ("\n");
20:        }
21:        printf ("\nEnter cordantes in form x,y -v to exit :");
22:        scanf ("%d,%d",&y,&x);
23:        matrix [y][x] ='*';
24:    }while(y>0);
25:
26: }
```

شكل رقم (٦-٥) برنامج ورقة رسم بياني

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج في السطر رقم ٦ بالاعلان عن مصفوفة ذات بعدين ولكن من نوع حرف char لأننا سوف نملأ عناصرها بحروف
 - في السطر رقم ٧ و ٨ دوراتين for لأننا نستعمل مصفوفة ذات بعدين
 - السطر رقم ٩ يملأ عناصر المصفوفة بالنقطة .
 - في السطر رقم ١١ تبدأ دوارة do..while clrsr() ثم مسح الشاشة بـ ()
 - السطور رقم ١٤ و ١٧ و ١٨ لطباعة المصفوفة السابقة التي سبق ملء عناصرها بالنقطة. وبالتالي سيتم طباعة شبكة من النقط عبارة عن ١٠ صفوف و ٧ عمود.
 - في السطر رقم ٢٣ يتم استقبال رقمين من المستخدم هما الاحداثيات المطلوب توقعها. وفي السطر رقم ٢٤ يتم وضع العلامة * في العنصر الذي يقابل الاحداثيات التي أدخلها المستخدم فمثلاً اذا أدخل المستخدم الرقمين ٥،٧ فيوضع في العنصر رقم [٧][٥] العلامة * ثم تعيين جملة while التشغيل الى السطر رقم ١٣ حيث تممسح الشاشة وتعيين رسم الشبكة بالوضع الجديد وهو أن العنصر المقابل لاختيار المستخدم به العلامة * فيرسمها وبالتالي توقع على الرسم وهكذا حتى نحصل على نتيجة التنفيذ كما في الشكل رقم ٦-٦
-*
-*
-*
-*
-*
-*
-*

Enter cordantes in form x,y -v to exit:

شكل رقم ٦-٦ ورقة رسم بياني

مصفوفة العبارة الحرفية Arroy of string

تستخدم الكلمة سلسلة حروف مقابل الكلمة string وقد ذكرنا من قبل أن طريقة الاعلان عن سلسلة الحروف هي string name[10]; ومعناها مصفوفة من الحروف ولكن ما معنى الاعلان التالي:

char name[5][10]

معناه مصفوفة حرفية عدد صفوفها ٥ وعدد اعمدتها ١٠ أي ٥ صفوف كل صف عبارة عن الكلمة لا يتعدى عدد حروفها عن ١٠ حروف وعلى ذلك يمكن أن تأخذ هذه المصفوفة قيمه بالشكل التالي

```
char name [5] [10]= {  
    {"string1"},  
    {"string2"},  
    {"string3"},  
    {"string4"},  
    {"string5"}  
};
```

في هذه المصفوفة يمكن التعامل مع حرف معين كما يمكن التعامل مع صف كامل على انه سلسلة حروف String ولذلك تسمى هذه المصفوفة array of string أو مصفوفة حرفية.

والبرنامج الموجود في الشكل رقم ٦-٧ يشتمل على مصفوفة عبارة عن مجموعة كلمات وفيه يطلب البرنامج من المستخدم ادخال كلمة سر فإذا كانت هذه

الكلمة أحدى الكلمات الموجودة في مصفوفة الكلمات سمح له البرنامج بتكميل العمل
والا استمر البرنامج في دوارة do..while حتى يدخل المستخدم احدى الكلمات
الموجودة بالمصفوفة

ويظهر ذلك في نتيجة التنفيذ الموجودة في الشكل رقم ٦-٨

```
0: /*Program Name CS6_7.C*/
1: #include <stdio.h>
2: main ()
3: {
4:     int i; char name2 [10]; int l=0;
5:     char name [5] [10] = {
6:         {" ahmed "},
7:         {"mohamed"}, ,
8:         {"samy "}, ,
9:         {" hamdy "}, ,
10:        {"nabil "}, ,
11:        }; ;
12:    do {
13:        clrscr ();
14:        printf ("\n Enter your name :");
15:        scanf ("%s", name2);
16:        for ( i=0; i<5 ; i++)
17:        {
18:            if ( strcmp (name2, name [i] == 0 )
19:                t=1 ;
20:            }
21:        }while (t== 0);
22: }
```

شكل رقم ٦-٧ مصفوفة العبارات الحرفية

وعن هذا البرنامج نوضح ما يلى :

يبدأ هذا البرنامج في السطر رقم ٥ بالاعلان عن مصفوفة حرفية وعطائها قيمة ابتدائية عبارة عن مجموعة كلمات وذلك بالاسم name وتنسخ المصفوفة لعدد ٥ اسماء كل اسم لا يتعدي ١٠ حروف . وفي السطر رقم ١٢ يبدأ التكرار بـ do ثم مسح الشاشة ثم رسالة لاستقبال الاسم . في السطر رقم ١٥ الدالة scanf() تستقبل سلسلة حروف وتخزنها في المتغير name2 ويشتمل السطر رقم ١٦ على الدوارة for ووظيفتها تكرار المقارنة ٥ مرات ويستخدم السطر رقم ١٨ الدالة strcmp() للمقارنة بين الاسم الذي ادخله المستخدم والاسماء المخزنة في المصفوفة فإذا كانا متساوين أي الفرق بينهما صفر اصبحت قيمة المتغير ٤ تساوى القيمة ١ وبالتالي لن يتحقق شرط التكرار الموجود في while أما اذا كانوا غير متساوين فان while تعيد التنفيذ الى السطر رقم ١٣ وبالتالي يتم مسح الشاشة وقبول كلمة اخرى وهكذا حتى يدخل المستخدم كلمة من الكلمات الموجودة في المصفوفة كما يظهر من تنفيذ البرنامج التالية :

Enter your name :aly

Enter your name :samah

Enter your name :hany

Enter your name :samy

شكل رقم ٨ - نتائج التنفيذ

ارسال مصفوفة للدالة كمعامل

من الامور المهمة استعمال المصفوفة كمعامل من معاملات الدالة ويتم توضيح ذلك في البرنامج الموجود في الشكل رقم ٦-٩ .

وفي هذا البرنامج يتم الاعلان عن مصفوفة اسماء وكذلك الاعلان عن دالة أحد معاملاتها مصفوفة ويتم استدعاء الدالة مع ارسال مصفوفة حرفية الى الدالة وتقوم الدالة بطباعة عناصر المصفوفة كما يتضح في نتيجة التنفيذ

```

0: /*Program Name CS6_9.C*/
1: #include <stdio.h>
2: void display(char array[5][10],int no);
3: main()
4: {
5:     char name [5] [10] = {
6:         {" ahmed "},
7:         {"mohamed"}, ,
8:         {"samy "},
9:         {" hamdy "},
10:        {"nabil "},
11:    };
12:    display(name,5);
13: }

14: void display(char array[5][10],int no)
15: {
16:     int i;
17:     for (i=0;i<no;i++)
18:     {
19:         printf("\n %s",array[i]);
20:     }
21: }

```

شكل رقم (٦-٩) ارسال المصفوفة الى الدالة

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج في السطر رقم ٥ بالاعلان مصفوفة حرفية واعطائها قيم ابتدائية هي مجموعة اسماء
- يستدعي السطر رقم ١٢ الدالة () display ويرسل لها اسم المصفوفة وعدد العناصر وفي السطر رقم ١٤ يبدأ انشاء الدالة () display()

- يقوم السطر رقم ١٩ بطباعة عناصر المصفوفة وذلك بتغيير رقم العنصر بالمتغير حتى نحصل على نتيجة التنفيذ الموجودة في شكل ٦-١٠ :

ahmed

mohamed

samy

hamdy

nabil

شكل ٦-١٠ طباعة عناصر المصفوفة

دوال العبارات الحرفية string functions

هناك مجموعة كبيرة من الدوال التي تعامل مع العبارات الحرفية وتستخدم في كثير من الأحيان مثل المقارنة بين الكلمات و معرفة عدد حروف عبارة حرفية وتحويل كلمة من الحروف الصغيرة (small) الى الحروف الكبيرة و نسخ سلسلة حروف في سلسة حروف أخرى أو إضافة سلسلة الى أخرى وهكذا

و معظم هذه الدوال موجودة في الملف string.h ولعلك تذكر أن هذا الملف وجميع الملفات المشابهة له تأتي مع برنامج المترجم الخاص بلغة C وتوضع في الفهرس .include المسمى

واليك أسماء بعض هذه الدوال والغرض من استخدامها ووظيفتها كلا منها :

(اما اذا أردت معرفة جميع الدوال فعليك بفتح الملف 4 string.h من اي محرر سطور)

الدالة	وظيفتها
strcat()	أضافة سلسلة حرفية (كلمة) الى نهاية سلسلة حرفية أخرى (كلمة أخرى)
strchr()	ايجاد ترتيب موقع حرف معين داخل كلمة
strcmp()	مقارنة كلمتين (أو متغيرين من نوع حرفى)

الدالة	وظيفتها
strcpy()	نسخ محتويات متغير حرفى فى متغير حرفى آخر
strlen()	ايجاد عدد حروف سلسلة حرفية
strlwr()	تحويل الكلمة من الحروف الكبيرة الى الحروف الصغيرة
strcmp()	مقارنة حروف الكلمة مع حروف الكلمة أخرى
strncpy()	نسخ حروف متغير حرفى فى متغير حرفى آخر
strrev()	عكس حروف متغير حرفى
strsett()	جعل حروف متغير كلها من حرف واحد
strupr()	تحويل الكلمة من الحروف الصغيرة الى الحروف الكبيرة

وهناك مجموعة دوال اخرى تستخدم للتحويل من متغير رقمى الى حرفى
والعكس وفيما يلى أهم هذه الدوال

الدالة	وظيفتها
atof	تحول متغير من نوع حرفى الى متغير من نوع رقم حقيقي
atoi	تحول متغير من نوع حرفى الى متغير من نوع رقم صحيح حتى يصلح التعامل معه كرقم
atol	تحول متغير من نوع حرفى الى متغير من نوع رقم صحيح طويل
_atold	تحول متغير من نوع حرفى الى متغير من نوع رقم حقيقي مضاعف
_itoa	تحول متغير من نوع صحيح الى متغير من نوع حرفى
_ltoa	تحول متغير من نوع رقم طويل الى متغير نوع حرفى
strtod	تحول متغير من نوع حرفى الى متغير من نوع رقم حقيقي مضاعف

أوامر المترجم preprocessors

أوامر المترجم هي أوامر تنفذ في حالة ترجمة البرنامج فقط وبالتالي هي أوامر توجه إلى المترجم. وعندما يقابل المترجم أحد هذه الأوامر يقوم بتنفيذ ما يدل عليه هذا الأمر ويطلق على هذه الأوامر preprocessors أو directive.

وكلمة directive معناها توجيه أي أمر إلى المترجم أما preprocessor فمعناه قبل التنفيذ والكلمتان تحققان المعنى لكل أمر يوجه إلى المترجم يحقق غرض معين وينفذ في حالة الترجمة ومن هذه الأوامر مایلی

#define	#error	#include
#elif	#if	#line
#else	#ifdef	#pragma
#endif	#ifndef	#undef

وقد درسنا معا الأمر #define في درس الماكروز وعرفنا أنه أمر يوجه إلى المترجم فمثلا #define a 5 يقول للمترجم ضع مكان كل حرف a القيمة 5 وهكذا ودرسنا الجملة #include<stdio.h> فمثلا #include معناها عند ترجمة هذا البرنامج افتح الملف stdio.h وضمه إلى الملف الحالى

أما #if و #else و #endif فهى تقوم بنفس العمل الذى تقوم الجملة if والجملة else داخل البرنامج ولكن مع المترجم ، فمثلا #if ... #endif تقول للمترجم اذا كان الشرط صحيح قم بعمل كذا و #else ... #endif تقول والا قم بعمل كذا و #endif تنهى الجملة الشرطية.

والأمر #ifdef معناها اذا لم تعرف كذا نفذ الجملة التالية ، والأمر #ifndef معناه اذا لم يتم تعريف ... نفذ الجمل التالية وهكذا ... وهناك ثوابت معرفة للجهاز ايضا تؤدى معنى معينا منها مایلی :

____LINE____, ____DATE____, ____TIME____, ____TIMESTAMP____, ____FILE____

فمثلاً :

تعيد رقم السطر في البرنامج الذي تكتب عنده هذا الشافت
____LINE____
وستستخدم لتحديد رقم سطر الخطأ.

تعيد التاريخ الحالي

____DATE____

تعيد الوقت الحالي

____TIME____

تعيد اسم الملف الحالي الذي يجري ترجمته

____FILE____

والبرنامج الموجود في الشكل رقم ٦-١١ عبارة عن برنامج يقوم باستعمال بعض هذه
الأوامر والثوابت ليعطي النتيجة الموضحة بالشكل رقم ٦-١٢ :

```
0: /*Program Name CS6_11.C*/
1: #ifdef _WINDOWS
2: #define STRING "DOING A WINDOWS PROGRAM!"
3: #else
4: #define STRING "NOT DOING A WINDOWS PROGRAM"
5: #endif
6: int main(void)
7: {
8:     printf( "\n\n" );
9:     printf( STRING );
10:    #ifdef _MSC_VER
11:        printf( "\n\nUsing a Microsoft compiler!" );
12:        printf( "\n Your Compiler version is %s", _MSC_VER );
13:    #endif
14:    #ifdef __TURBOC__
15:        printf( "\n\nUsing the Turbo C compiler!" );
16:        printf( "\n Your compiler version is %x", __TURBOC__ );
17:    #endif
18:    #ifdef __BORLANDC__
19:        printf( "\n\nUsing a Borland compiler!" );
20:    #endif
```

```
21:     return(0);  
22: }
```

شكل رقم (٦-١١) استعمال الأوامر والثوابت المعرفة للمترجم

وعن هذا البرنامج نوضح مايلي :

يشتمل البرنامج على بعض التوجيهات وبعض الثوابت ففي السطر رقم ١ جملة
#if تختبر هل البرنامج الحالى هو برنامج نوافذ أم لا وذلك باختبار الشابت
_WINDOWS وضع النتيجة في المتغير STRING ويطلب السطر رقم ٩ طباعة
نتيجة المقارنة

وبالمثل السطر رقم ١٠ و ١٤ و ١٨ لاختبار اصدار مترجم لغة C المستخدم
وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

NOT DOING A WINDOWS PROGRAM
Using the Turbo C compiler!
Your compiler version is 295

شكل رقم ٦-١٢

والبرنامج الموجود بالشكل رقم ٦-١٣ يقوم باستعمال ثوابت الوقت والتاريخ
ورقم السطر كما يظهر من نتيجة التنفيذ :

```
/*Program Name CS6_13.C*/  
#include <string.h>  
int main(void)  
{  
    printf("\n\nCurrently at line %d", __LINE__ );  
    printf("\n\nThe value of __DATE__ is: ");  
    printf(__DATE__);  
    printf("\n\nThe value of __TIME__ is: ");
```

```
    printf(__TIME__);
    printf("\n\nThe value of __LINE__ is: %d", __LINE__);
    printf("\n\nThe value of __FILE__ is: ");
    printf(__FILE__);
    return;
}
```

شكل رقم ٦-١٣

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Currently at line 14
The value of __DATE__ is: May 02 1995
The value of __TIME__ is: 22:56:18
The value of __LINE__ is: 22
The value of __FILE__ is: C:\CBOOK\CHP06\CS6_13.C
```



الفصل السابع

توظيف المفاتيح

والتحكم في حركة المؤشر

في هذا الفصل نتناول الموضوعات التالية:

استخدام جدول الأكواد الممتدة EXTENDED CODE TABLE

- ♦ التعريف بملف ANSI.SYS واستخداماته ومتطلباته.
- ♦ التحكم في حركة المؤشر
- ♦ وضع المؤشر في أي مكان على الشاشة
- ♦ التحكم في خصائص الحروف
- ♦ تصميم قوائم الاختيارات HIGH LIGHT MENU

توضيف المفاتيح

تلخص فكرة توظيف أي مفتاح في استقبال حرف من لوحة المفاتيح واختبار هذا الحرف لتقرير هل هو الحرف المطلوب أم لا ، وبالتالي اتخاذ قرار بناء على نتيجة المقارنة ويكون الاختبار بطريقتين اما بمقارنة الحرف بالحرف أو بمقارنة كود الحرف المستقبل بكود الحرف المطلوب توظيفه . ونوضح فيما يلى مثال لكل حالة المثال الموجود بالشكل ٧-١ يختبر هل الحرف الذي يكتبه المستخدم هو حرف y فإذا كانت نتيجة المقارنة صحيحة يطبع الرسالة you typed y

```
main()
{
    ch=getch();
    if(ch=='y')
        printf("\n you typed y ");
}
```

شكل رقم ٧-١ برنامج لاختبار الحرف المدخل

المثال الموجود بالشكل ٧-٢ يختبر هل كود الحرف الذي يكتبه المستخدم يقابل حرف a أم لا ؟ فإذا كانت نتيجة المقارنة صحيحة يطبع الرسالة a.you typed a.

```
0: /*CS7_2.C*/
1: main()
2: {
3:     int ch;
4:     ch=getch();
5:     if(ch==58)
```

الفصل السابع : توظيف المفاتيح والتحكم في حركة المؤشر

```
6: printf("\n you typed letter a ");
7: }
```

شكل رقم ٧-٢ برنامج لاختبار كود الحرف المدخل

وفي هذا المثال نلاحظ أن :

السطر رقم ٤ يشتمل على الدالة () getch وهذه الدالة تستقبل حرف وتخزنه في المتغير ch والسطر رقم ٦ يقارن اذا كان كود الحرف المستقبل هو الكود المطلوب أم لا فإذا كان هو يطبع الرسالة

هذا بالنسبة للمفاتيح العادية أي جميع الحروف التي لها كود موجود جدول الأكواود المعروفة ASCII TABLE والموجود بالملحق رقم (أ)

ولكن ماذا عن المفاتيح التي ليس لها كود في هذا الجدول مثل :

F1,F2,ALT,CTR,.....

هذا ما سترى في البند التالي :

توظيف مفاتيم الوظائف ومتانيم التحكم

تشتمل لوحة المفاتيح بالإضافة إلى المفاتيح العادية التي تستخدم في الكتابة على مفاتيح أخرى يتم توظيفها غالباً وتختلف وظيفتها من برنامج لآخر ومن أمثلتها مفاتيح الوظائف F1 إلى F12 أو مفاتيح التحكم مثل المفتاح Ctrl أو مفتاح Alt.

عند الضغط على مفتاح من المفاتيح العادية مثل A,B,C ترسل لوحة المفاتيح للجهاز بait واحدة ويتم اختبار هذه البايت وبالتالي توظيف المفتاح

ولكن عند الضغط على أحد مفاتيح الوظائف أو التحكم مثل المفتاح F1 ترسل لوحة المفاتيح ٢ بايت للجهاز

البایت الأولى لابد ان تكون صفر (0) والبایت الثانية هي الكود الحقيقي للمفتاح. البایت الأولى هي التي تقول أن هذا المفتاح مفتاح ممتد أى من مفاتيح الوظائف وبالتالي ليس له ASCII CODE.

وبالتالي جميع المفاتيح الممتد مثل F1,F2,...ALT,CTR تشتراك في أن البایت الاولى لها صفر (0) والبایت الثانية هي التي تميز كل مفتاح عن الآخر وهنالك جدول آخر لمثل هذه المفاتيح يسمى جدول المفاتيح الممتدة .EXTENDED CODES TABLE

ويوضح الجدول رقم ١-٧ الأكواد المخصصة للمفاتيح التي تأخذ ضغطة واحدة مثل F1,F2,F3,.....

ويوضح الجدول رقم ٢-٧ أكواد المفاتيح التي تأخذ ضغطتين أو أكثر مثل ALT+C,CTR+A,

المفتاح	البایت الثاني	البایت الاولى
F1	0	59
F2	0	60
F3	0	61
F4	0	62
F5	0	63
F6	0	64
F7	0	65
F8	0	66
F9	0	67
F10	0	68

الفصل السابع : توظيف المفاتيح والتحكم في حركة المؤشر

المفتاح	البait الاولى	البait الثاني
Home	71	0
Up arrow	72	0
PgUp	73	0
Left arrow	75	0
Right Arrow	77	0
End	79	0
Down	80	0
Pg Dn	81	0
Ins	82	0
Del	83	0

جدول رقم ٧-١ أكواد المفاتيح المفردة

المفتاح	البait الاولى	البait الثاني
Shift Tab	15	0
Alt Q,W,E,R,T,Y,U,I,O,P	16 to 25	0
Alt A,S,D,F,G,H,J,K,L	30 to 38	0
Alt Z,X,C,V,B,N,M	44 to 50	0
Shift F1 to F10	84 to 93	0
Ctrl F1 to F10	94 to 103	0
Alt F1 to F10	104 to 113	0
Ctrl PrtSc	114	0
Ctrl Left arrow	115	0
Ctrl right arrow	116	0
Ctrl End	117	0
PgDn	118	0

المفتاح	البايت الاولى	البايت الثانية	
	119	0	Ctrl Home
	120 to 131	0	Alt 1,2,3,4,5,6,7,8,9,0,
	132		Ctrl PgUp

جدول رقم ٧-٢ أكواد المفاتيح المشتركة

مثال تحديد المفتاح وكوده

البرنامج الموجود بالشكل رقم ٧-٣ يختبر المفتاح الذى يضغطه المستخدم فإذا كان مفتاح عادى وكوده موجود بجدول الأكواد العادى يطبع رسالة تدل على ذلك بالإضافة إلى الكود الخاص به، وإذا كان المفتاح من مفاتيح الوظائف يطبع رسالة تدل على ذلك بالإضافة إلى كود المفتاح كما يظهر ذلك في نتيجة التنفيذ في الشكل رقم ٧-٤.

```

0: /*CS7-3.C*/
1: #include <stdio.h>
2: main()
3: {
4:     int key1,key2;
5:     while((key1=getch())!="\r")
6:     {
7:         if(key1==0)
8:         {
9:             key2=getch();
10:            printf("\n This Is Extended Code=%d",key2);
11:        }
12:    else

```

```
13:         printf("\n This Normal Code =%d",key1);
14:     }
15: }
```

شكل رقم ٣-٧ تحديد المفتاح وكوده

Extended Code = 59

Extended Code = 60

Extended Code = 61

Extended Code = 62

Normal Code = 121

Normal Code = 106

Normal Code = 106

Normal Code = 104

شكل رقم ٤-٧ نتائج التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٤ اعلان عن متغيرين من نوع صحيح
- و السطر رقم ٥ يستقبل حرف ويحزرنه في المتغير key1 ويستمر في استقبال الحروف ما لم يضغط المستخدم المفتاح Enter و السطر رقم ٧ يقارن اذا كان هذا الكود هو صفر(٠) أي مفتاح ممتد مثل F1,F2 يرسل ٢ بايت الى المخزن المؤقت (buffer) ويشتمل السطر رقم ٩ على الدالة () getch التي تأخذ من الـ buffer بايت واحدة وتخبرها اذا كانت صفر تقوم بطباعة رسالة وكود المفتاح كما في السطر رقم ١٠ فإذا لم تكن هذه البايت صفر فهذا معناه أن الحرف حرف عادي وبالتالي يطبع رسالة تدل على ذلك بالإضافة الى كود المفتاح العادي. وذلك في السطر رقم ١٣

ولمعرفة كيفية توظيف المفاتيح الممتدة لتنفيذ دالة معينة تابع البرنامج الموجود بالشكل رقم (٧-٥) وهو برنامج ينتظر حتى يضغط المستخدم على مفتاح فإذا ضغط المستخدم مفتاح الوظيفة F1 يطبع البرنامج رسالة بذلك ويظهر ذلك في نتيجة التنفيذ في شكل رقم ٧-٦

```
0: /*CS7-5.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     char key,key2;
6:     while ((key=getch()) !='\r')
7:     if (key ==0)
8:     {
9:         key2=getch();
10:        switch (key2)
11:        {
12:            case 59:
13:                printf ("\n function 1");
14:                break;
15:            case 60:
16:                printf ("\n function 2");
17:                break;
18:            default:
19:                printf ("\n other extended code ");
20:        }
21:    }
22:    else
23:        printf ("\n normal code %3d",key);
24: }
```

شكل رقم ٧-٥ توظيف مفاتيح الوظائف

```
function 1
function 2
other extended code
function 2
other extended code
normal code 102
normal code 116
normal code 55
normal code 121
other extended code
```

شكل رقم ٧-٦ نتيجة التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- السطر رقم ٦ يستقبل حرف . ويوضعه في المتغير key باستخدام الدالة ()getch
- في السطر رقم ٧ تختبر جملة if كود الحرف المدخل لتحديد المفتاح من نوع ممتد (مفاتيح الوظائف) أم لا
- ثم في السطر رقم ٩ الدالة ()getch تستقبل الكود الثاني وتخزنه في المتغير key2 والذي يمثل الكود الحقيقي للمفتاح الممتد
- وفي السطر رقم ١٢ case تختبر هل هذا الكود هو 59 أى كود المفتاح F1 إذا كان كذلك اطبع الجملة 1 function 1 وهكذا السطر رقم ١٥
- أما السطر رقم ٢٢ فيقول اذا لم يكن الكود ممتد أى لم يكن الكود الاول ،نفذ السطر رقم ٢٣ الذي يطبع عبارة أنه مفتاح عادي

استعمال جدول الأكواد الممتدة EXTENDED CODE TABLE

من الجدول رقم ١-٧ والجدول رقم ٢-٧ والبرنامج الموجود بالشكل رقم ٥-٧ يتضح أن فكرة توظيف أي مفتاح ممتد هي معرفة كود هذا المفتاح من الجدول الخاص به واختباره فمثلاً لتوظيف المفتاح F1 ليؤدي وظيفة help نكتب دالة بالاسم `help()` بالشكل الآتي :

case 59:

help();

break;

على أن يتم كتابة محتويات الدالة `help()` في البرنامج

استخدام ملف ANSI.SYS

ملف ANSI.SYS هو ملف من انتاج المعهد الامريكي لتوحيد القياسات ويأتي هذا الملف مع ملفات نظام التشغيل DOS وهو يوفر مجموعة من الوظائف يمكن استغلالها مع معظم اللغات وكذلك لغة C

والوظائف التي يوفرها ملف ANSI.SYS يمكن تحقيقها بثلاث طرق :

- دوال لغة C مباشرة
- ملف ANSI.SYS
- استخدام ROM BIOS

وسوف نشرح فيما بعد تفصيل كل طريقة ونبداً الأن بملف ANSI.SYS والوظائف التي يوفرها وكيفية استعمالها.

متطلبات ملف ANSI.SYS

لكي تستخدم ملف ANSI.SYS يجب تهيئة الجهاز بملف CONFIG.SYS وذلك بكتابة السطر التالي في الملف

DEVICE=ANSI.SYS

بشرط أن يكون ملف ANSI.SYS موجود على الفهرس الرئيسي للقرص الذي تبدأ فيه تشغيل الجهاز أو ان يشتمل أمر path على اسم الدليل الذي يوجد تحته هذا الملف.

ومن داخل البرنامج يتم استدعاء ملف ANSY.SYS لنادبة وظيفة معينة بالشكل الآتي :

```
printf("\x1B[code]");
```

تأمل الكود [\x1B] هذا الكود ثابت ومعناه أنك تتحدى ملف ansi.sys أما الوظيفة المطلوبة فيكتب كودها مكان كلمة code وسيتضح ذلك من الأمثلة القادمة.

التحكم في حركة المؤشر

المقصود بالتحكم في حركة المؤشر (Cursor) استخدام المؤشر للتحرك لأعلى أو لأسفل أو يميناً أو يساراً أو صفحة لأعلى pageup أو صفحة لأسفل pagedown أو بداية السطر home أو نهاية السطر end وذلك باستخدام أكواد ملف ansi.sys من خلال الدالة printf() التي تأخذ الشكل العام التالي :

```
printf("\x1B[movement code]");
```

حيث movement code هو كود يحدد اتجاه الحركة المطلوبة فمثلاً :

في السطر السابق تجد الكود الثابت لاستخدام ansi.sys وهو \x1B بالاضافة للحرف B هو كود التحرك لأسفل بمقدار صف واحد (بالنظام السادس عشر)

وبالتالي كل ما يفعله هذا الكود هو تحريك المؤشر لأسفل بمقدار صف واحد من موضعه الحالى.

والبرنامج الموجود بالشكل رقم (٧-٧) يوضح كيفية استخدام أكواد التحكم لتحريك المؤشر .

```
0: /*CS7_7.C*/
1: #include <sdtio.h>
2: #include <conio.h>
3: main ()
4: {
    clrscr();
    while (getche() != '.')
        printf ("\x1B[B");
}
```

شكل رقم ٧-٧ استخدام أكواد Ansi.sys في تحريك المؤشر

وعن هذا البرنامج نوضح ما يلى :

يشتمل هذا البرنامج على الدالة printf() التي تقوم بتحريك المؤشر لأسفل بمقدار صف واحد كلما أدخل المستخدم حرفا من لوحة المفاتيح ويوضح الجدول رقم ٣-٧ بقية الأكواد الخاصة بتحرك المؤشر cursor والتي تحتاج إليها في تحريك المؤشر في اتجاهات أخرى

و عند تفريذ هذا البرنامج ستحصل على النتيجة التالية :

الفصل السابع : توظيف المفاتيح والتحكم في حركة المؤشر

a
b
c
d
e
f
g
h
i

شكل رقم ٧-٨ نتيجة التنفيذ

الكود	التأثير
"[2J"	مسح الشاشة
"[K"	مسح الحروف حتى نهاية السطر
"[A"	حركة المؤشر لاعلى
"[B"	حركة المؤشر لأسفل
"[C"	حركة المؤشر لليمين
"[D"	حركة المؤشر للشمال
"[%d;%df"	وضع المؤشر في الصف والعمود المحددين
"[s"	حفظ الموضع الحالى للمؤشر
"[u"	استرجاع الموضع السابق للمؤشر
"[%dA"	حركة المؤشر لاعلى عدد صافوف A
"[%dB"	حركة المؤشر لأسفل عدد من الاعمدة B
"[%dC"	حركة المؤشر لليمين عدد من الاعمدة C
"[%dD"	حركة المؤشر للشمال عدد من الاعمدة D

جدول ٣-٧ الأكواد المختلفة لحركة المؤشر

الرسم باستخدام مفاتيح الأسهم

البرنامج الموجود بالشكل رقم ٧-٩ برنامج شامل يوضح كيفية توظيف مفاتيح الأسهم واستخدام ملف ansi.sys في تحريك المؤشر والحصول على الرسم الحر

```
0: /*CS7_9.C*/
1: #define C_LEFT "\x1B[D"
2: #define C_RIGHT "\x1B[C"
3: #define C_UPUP "\x1B[A"
4: #define C_DOWN "\x1B[B"
5: #define L_ARRO 75
6: #define R_ARRO 77
7: #define U_ARRO 72
8: #define D_ARRO 80
9: #define ACCROSS 205
10: #define UPDOWN 186
11: void so();
12: main ()
13: {
14:     int key,key2;
15:     clrscr();
16:     while ((key=getch()) == 0)
17:     {
18:         key2=getch();
19:         switch (key2)
20:         {
21:             case L_ARRO:
22:                 printf(C_LEFT); printf("%c",ACCROSS);
23:                 so();
24:                 break;
25:             case R_ARRO:
```

الفصل السابع : توظيف المفاتيح والتحكم في حركة المؤشر

```
26:         printf (C_RIGHT); printf ("%c",ACCROSS);
27:         so();
28:         break;
29:     case U_ARRO:
30:         printf (C_UPUP); printf ("%c",UPDOWN);
31:         so();
32:         break;
33:     case D_ARRO:
34:         printf (C_DOWN); printf ("%c",UPDOWN);
35:         so();
36:         break;
37:     }
38:     printf (C_LEFT);
39:   }
40: }
```



```
41: void so()
42: {
43:   int i;
44:   for (i=0;i<350;i++)
45:   {
46:     sound(i*4);
47:     delay(4);
48:     nosound();
49:   }
50: }
```

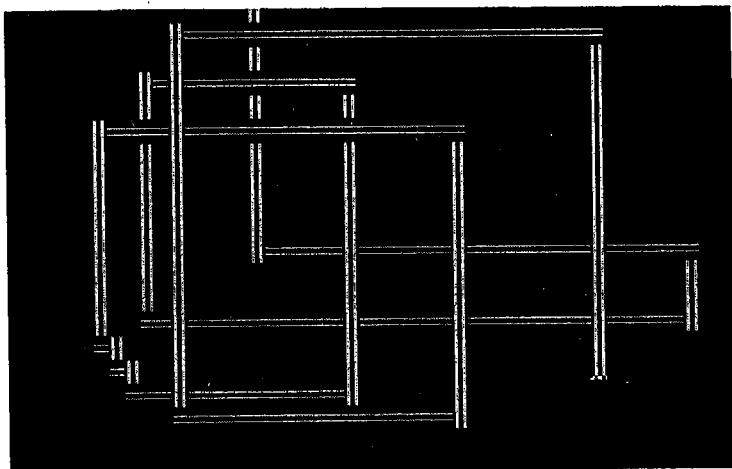
شكل رقم ٧-٩ برنامج الرسم الحر

وعن هذا البرنامج نوضح ما يلى :

تقوم فكرة هذا البرنامج على استقبال كود المفتاح وتحديد نوع المفتاح الذى تم الضغط عليه هل هو مفتاح الحركة لاعلى أو لأسفل أو اليمين أو الشمال وبعد

تحديد اتجاه الحركة يتم تحريك المؤشر الى الاتجاه المطلوب مع طباعة الحرف المستعمل في الرسم وهكذا يستمر البرنامج بالسماح للك باستعمال مفاتيح الحركة في الرسم حتى يتم الضغط على مفتاح الادخال.

وبتنفيذ هذا البرنامج والتحرك بمفاتيح الاسهم تحصل على شكل مشابه لشكل ٧-١٠.



شكل رقم ٧-١٠ نتائج برنامج الرسم الحر

توجيه المؤشر الى أي مكان على الشاشة

باستخدام ملف ansi.sys يمكنك وضع المؤشر في أي مكان على الشاشة

ويستخدم الصورة التالية :

```
printf("\x1B[R;Cf")
```

حيث R هي رقم الصف ، C هي رقم العمود ، و تكتب f كما هي لأنها تابعة للكود والبرنامج الموجود بالشكل رقم ٧-١١ يوضح ذلك حيث يقوم البرنامج باستقبال قيمتين صحيحتين ثم يقوم بتوجيه المؤشر الى هذا المكان على الشاشة مع طباعة العلامة * عند هذا المكان ليظهر موقعه على الشاشة.

الفصل السابع : توظيف المفاتيح والتحكم في حركة المؤشر

```
0: /*CS7_11.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     int x,y;
6:     while (1)
7:     {
8:         printf ("\n Enter Row,col:");
9:         scanf ("%d,%d",&x,&y);
10:        printf ("\x1B[%d;%df",x,y);
11:        printf ("%d,%d",x,y);
12:    }
13: }
```

شكل رقم ٧-١١ برنامج توجيه المؤشر على الشاشة

التحكم في خصائص الحروف

من الاستخدامات المتأحة لملف Ansi.sys تغيير خصائص الكتابة والحرروف التي تظهر على الشاشة فمثلا يمكن عكس ألوان الكتابة بدلا من أن تكون باللون الأبيض على اللون الأسود يمكن عكسها لظهور باللون الأسود على الأبيض وكذلك يمكن الكتابة بحروف مائلة أو مسطرة أو ثقيلة وهكذا.

ولتحقيق ذلك نسيعمل الصورة التالية :

```
printf( "\x1B[NOm")
```

حيث يأخذ المتغير NO رقم يحدد الخاصية المطلوبة للحروف وهذا الرقم يتم

اختياره من الجدول الآتي :

الرقم	تأثيره	مثال
0	Normal	أى الكتابة بخط عادي
1	Bold	أى الكتابة بخط ثقيل
4	Under Line	أى تسطير الكتابة
5	Blink	أى تجعل الكتابة توomez
7	Reverse	أى عكس الكتابة
8	Invisible	أى إخفاء الكتابة

والبرنامج الموجود بالشكل رقم ٧-١٢ يستخدم الاكواذ الموجودة بالجدول السابق لطباعة مجموعة كلمات بأشكال مختلفة ويظهر ذلك في نتيجة التنفيذ الموجودة بالشكل رقم ٧-١٣

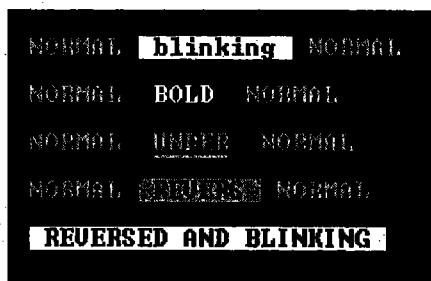
```

0: /*CS7_12.C*/
1: #define NORMAL "\x1B[0m"
2: #define BOLD " \x1B[1m"
3: #define UNDER "\x1B[4m"
4: #define BLINK "\x1B[5m"
5: #define REVERES "\x1B[7m"
6: main()
7: {
8: printf("NORMAL%s blinkng%sNORMAL\n\n",BLINK,NORMAL);
9: printf("NORMAL%sBOLD%sNORMAL\n\n",BOLD,NORMAL);
10: printf("NORMAL%sUNDER%sNORMAL\n\n",UNDER,NORMAL);
11: printf("NORMAL%sREVERS%s NORMAL \n\n",REVERES,NORMAL);
12: printf ("%s%s REVERSED AND BLINKING \n\n",REVERES,BLINK);
13: }

```

الشكل رقم ٧-١٢ تغيير خصائص الحروف

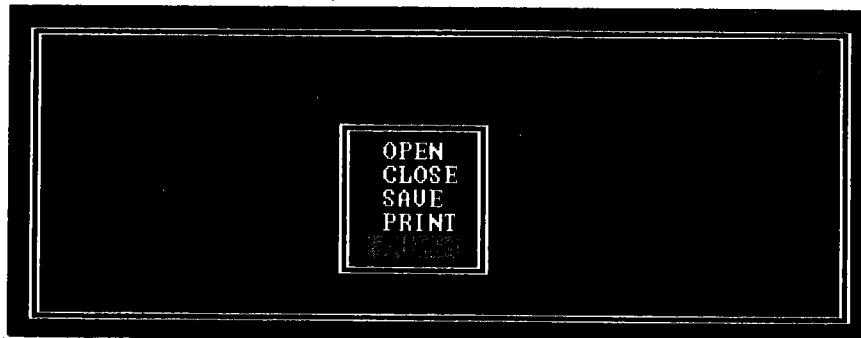
و عند تنفيذ هذا البرنامج نحصل على النتيجة التالية :



الشكل رقم ٧-١٣ طباعة الخطوط باشكال مختلفة

HIGH LIGHT MENU

البرنامج الموجود بالشكل رقم ٧-١٥ يعتبر من التطبيقات الجذابة و هو عمل قائمة انتيارات ولكن بدلا من كتابة أرقام لاختيارات القائمة و كتابة رقم الاختيار يظهر شريط مضاء فوق أول اختيار ويقوم المستخدم بتحريك الشريط المضاء باستخدام مفاتيح الاسهم ثم الضغط على مفتاح الادخال عند الاختيار المطلوب (انظر شكل ٧-١٤).



الشكل رقم ٧-١٤ استخدام الشريط المضاء في قوائم الأنتيارات

```
0: /* Program Name CS7_15.C*/
1: #define TRUE 1
2: #define NUM 5
3: #define CLEAR "\x1B[2J"
4: #define ERASE "\x1B[K"
5: #define NORMAL "\x1B[0m"
6: #define REVERSE "\x1B[7m"
7: #define HOME "\x1B[10;5f"
8: #define BOTTOM "\x1B[20;1f"
9: #define U_ARRO 72
10: #define D_ARRO 80
11: #define INSERT 82
12: #include <stdio.h>
13: #include <conio.h>
14: #include <stdlib.h>
15: void fancy_box(int x1,int y1,int x2,int y2)
16: {
17:     int i;
18:     gotoxy(x1,y1);putch(201);
19:     for(i=x1+1;i<x2;i++) putch(205);
20:     putch(187);
21:     for(i=y1+1;i<y2;i++)
22:     {
23:         gotoxy(x1,i);putch(186);
24:         gotoxy(x2,i);putch(186);
25:     }
26:     gotoxy(x1,y2);putch(200);
27:     for (i=x1+1;i<x2;i++) putch(205);
28:     putch(188);
29: }
30: void action(int);
31: void display (char *arr[],int size,int pos);
32: char getcode(void);
```

```
33: void main (void)
34: {
35:     static char *items[NUM]={
36:         " OPEN ",
37:         " CLOSE ",
38:         " SAVE ",
39:         " PRINT ",
40:         " QUIT ";
41:
42:     int curpos=0;
43:     textbackground(BLUE);
44:     textcolor(WHITE);
45:     clrscr();
46:     fancy_box(1,1,80,24);
47:     fancy_box(8,8,60,20);
48:     fancy_box(28,12,37,18);
49:     while (TRUE)
50:     {
51:         display(items,NUM,curpos);
52:         switch (getcode())
53:         {
54:             case U_ARRO:
55:                 if(curpos>0) --curpos;
56:                 else
57:                     curpos+=4;
58:                 break;
59:             case D_ARRO:
60:                 if(curpos<NUM-1) ++curpos;
61:                 else
62:                     curpos -=4;
63:                 break;
64:             case 'r':
65:                 action(curpos);break;
66:         }
}
```

```
67:     }
68: }
69: void display (char *arr[],int size,int pos)
70: {
71:     int j;
72:     printf (HOME);
73:     for (j=0;j<size;j++)
74:     {
75:         if(j==pos)
77:         {
78:             textcolor(RED);
79:             textbackground(GREEN);
80:         }
81:         gotoxy(15,10+j);
82:         cprintf ("%s\r\n",*(arr+j));
83:         {
84:             textcolor(WHITE);
85:             textbackground(BLUE);
86:         }
87:     }
88:     printf (BOTTOM);
89: }

90: char getcode(void)
91: {
92:     int key;
93:     if ((key=getch()) ==0)
94:         return (getch());
95:     else if (key=='r')
96:         return(key);
97:     else
98:         return (0);
99: }
```

الفصل السابع : توظيف المفاتيح والتحكم في حركة المؤشر

```
100: void action(int pos)
101: {
102:     printf (ERASE);
103:     switch (pos)
104:     {
105:         case 0:
106:             printf ("openla");break;
107:         case 1:
108:             printf ("closela");break;
109:         case 2:
110:             printf ("savela");break;
111:         case 3:
112:             printf ("printla");break;
113:         case 4:
114:             exit (0);
115:     }
116: }
```

الشكل رقم ٧-١٥ برنامج القائمة ذات الشريط المضاء

وعن هذا البرنامج نوضح ما يلى :

يقسم البرنامج الموجود في الشكل رقم ٧-١٤ إلى الأجزاء الآتية :

- من السطر رقم ٣ إلى السطر رقم ١١ استخدامنا كلمة define في عمل مجموعة تعریفات يتم استخدامها في البرنامج.
- من السطر رقم ١٥ إلى ٢٨ أنشأنا دالة لرسم مستطيل
- من السطر رقم ٣٥ إلى ٤٠ مصفوفة عبارات حرفية تحتوى على إختيارات الشاشة الرئيسية.
- في السطر رقم ٤٩ الدوارة while1(TRUE) وهي الدوارة الائتمانية
- في السطر رقم ٥١ استدعاء الدالة () display التي تقوم باظهار قائمة

الاختيارات الموجودة في المصفوفة items وتم الشاء هذه الدالة ابتداء من السطر رقم ٦٩.

وأنشأنا في هذا البرنامج مجموعة دوال لتحقق الغرض وهي :

الدالة () display تقوم بعرض اختيارات القائمة وتحديد مكان الشريط المضاء.
الدالة () getcode() تقوم باستقبال مفتاح من المستخدم واعادة كود هذا المفتاح
إلى الدالة الرئيسية.

الدالة () action والتي تقوم بتنفيذ المستخدم.

ويأخذ تنفيذ البرنامج التسلسل الآتي :

- في السطر رقم ٥ الدالة () display تقوم بعرض قائمة الاختيارات على الشاشة.
- في السطر رقم ٥٢ الدالة () getcode() تقوم باستقبال مفتاح من المستخدم وتقوم الدوارة while باختبار هذا المفتاح
- في السطر رقم ٤ أول حالة من الحالات المحتملة للمفتاح وهي حالة ضغط المستخدم على مفتاح السهم العلوي وفي هذه الحالة يتم انقص قيمة المتغير curpos بمقدار واحد اذا كانت قيمة المتغير أكبر من صفر والاأخذ المتغير القيمة ٤ حتى يذهب الشريط المضاء الى آخر القائمة ثم تنتهي هذه الحالة ويتم الرجوع الى الدوارة while التي تقوم باستدعاء الدالة () display مرة ولكن بقيمة مختلفة للمتغير curpos تعبر عن حركة الاسهم وهكذا باقى الحالات.



الفصل الثامن

مؤشرات العنوانين

POINTERS

- ♦ يتناول هذا الفصل الموضوعات التالية
- ♦ معنى المؤشر **pointer**
- ♦ مزايا استخدام المؤشر **pointer**
- ♦ إعادة أكثر من قيمة من الذاكرة
- ♦ المؤشرات والمصفوفات
- ♦ تمرير مصفوفات إلى الذاكرة باستعمال المؤشرات
- ♦ الحرفيات والمؤشرات **pointers and string**
- ♦ مصفوفة الحرفيات والمؤشرات

معنى المؤشر Pointer

المؤشر (pointer) هو نوع من أنواع البيانات ويعرف بأنه متغير يحتفظ (يخزن) بعنوان مكان في الذاكرة.

من المعلوم أن كل مكان في الذاكرة له عنوان والجهاز يتعامل مع هذا المكان بالعنوان المحدد له ونحن بطريقة غير مباشرة نتعامل مع هذا العنوان، فمثلاً هذا الإعلان معناه احجز مكان في الذاكرة (RAM) حجمه ٢ بايت (حجم int) واجعل اسمه a ووضع فيه القيمة ٥

وبالتالي كلما تعاملنا مع المتغير a فنحن نتعامل مع القيمة المخزنة فيه وليس العنوان المخصص لهذه القيمة . هذا عن الإعلان العادي ، فماذا عن الإعلان عن المؤشر (Pointer) هذا ما سنوضحه في البند التالي

الإعلان عن المؤشر pointer

يتم الإعلان عن مؤشر إلى أي متغير من أنواع البيانات بنفس الطريقة التي نعلن بها عن البيانات العادية وهي تحديد نوع البيانات ثم اسم المتغير ولكن الفرق بين الإعلان عن المتغير والإعلان عن المؤشر أن اسم المتغير يجب أن يسبق بالعلامة * ليدل على أنه مؤشر ، أي أن العلامة * تجعل المتغير مؤشر. فمثلاً للإعلان عن مؤشر من نوع صحيح نكتب الصورة التالية

int * p;

وكما ترى ليس هناك جديد غير أن اسم المتغير سبق بالعلامة *

وماذا يعني هذا الإعلان ؟

يعني أن المتغير *p* أصبح مؤشر إلى مساحة في الذاكرة مقدارها ٢ بait مع الإحتفاظ
بعنوان هذا المكان في المتغير *p*

هل لاحظت الكلمة عنوان هذا ما يهمنا، وكلما أردنا أن نتعامل مع هذه القيمة
تعاملنا عن طريق العنوان أى بدلاً من أن نتعامل نحن مع القيمة ونترك الجهاز يتعامل مع
العنوان بهذا الأسلوب نستطيع أن نتعامل مباشرة مع عنوان المكان مما يعطينا القدرة على
عمليات كثيرة منها التعامل مع مخارج الجهاز مثل مخرج آلة الطباعة حيث أن لمخرج
الطباعة عنوان فنستطيعأخذ هذا العنوان وتخزينه في متغير ثم التعامل مع هذا المتغير كما
شاء وكذلك الكتابة في ذاكرة العرض مباشرة وهكذا.

يمكن للمؤشر أن يشير إلى أي نوع من أنواع البيانات حسب الإعلان



شرحنا كيف يتم الإعلان عن مؤشر يشير إلى قيمة صحيحة فكيف يكون
الإعلان عن مؤشر يشير إلى قيمة حقيقة (pointer to float)
يكون ذلك بالصورة التالية

`float *k;`

ومعناه أحجز مكان في الذاكرة مقداره ٤ بait وخزن عنوان هذا المكان في
المتغير *k* الذي يحفظ بهذا العنوان

اذن طريقة الإعلان عن مؤشر إلى أي نوع من أنواع البيانات هي نفس الطريقة
المستخدمة للإعلان عن المتغيرات غير أنها نسبق المتغير بالعلامة * وهذا يعني أنه مؤشر
إلى هذا النوع

مزايا استخدام المؤشرات pointer

يتحقق استعمال المؤشرات فوائد كثيرة منها

- إعادة أكثر من قيمة من الدوال.
- التعامل مع المصفوفات والحرفيات وتمريرها الى الدوال بشكل أفضل
- إنشاء أنواع أكثر قوة من البيانات
- التعامل مع الجهاز ومكوناته وعناوين مداخل وخارج الجهاز

إعادة أكثر من قيمة من الدوال

من الفوائد المشهورة للمؤشرات استخدامها في إعادة أكثر من قيمة من الدالة.

فما معنى ذلك ؟

في الفصل الخامس شرحنا الدوال والتعامل معها وكيفية إعادة قيمة من الدالة لاحظنا في الأمثلة التي استخدمناها أننا استخدمنا كلمة return مرة واحدة مع كل دالة وهذا معناه عدم امكانية إعادة أكثر من قيمة من الدالة.

فلو فرضنا أن لدينا مجموعة عمليات وأردنا إنشاء دالة لهذه العمليات وانشأنا الدالة وتم حساب نتائج العمليات ووضع هذه النتائج في متغيرات وأردنا إعادة هذه القيم إلى الدالة الرئيسية ، هنا تظهر المشكلة . وهي أننا لا نستطيع استعمال أكثر من كلمة return وكلمة return لا تعيد إلا قيمة واحدة أما في حالة استخدام المؤشرات فيمكننا إعادة أكثر من قيمة.

ولتوضيح ذلك سنكتب برنامج البرنامج الأول بدون استعمال المؤشرات وفيه ستظهر هذه المشكلة ، والبرنامج الثاني باستخدام المؤشرات ومنه سترى كيف يمكن حل هذه المشكلة

يشتمل الشكل رقم ٨-١ على برنامج يقوم بإنشاء دالة تأخذ معاملين من نوع صحيح ثم تقوم الدالة بإضافة القيمة ٥ إلى كل معامل

```
0: /* Program Name CS8_1.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: void get2(int xx, int yy);
4: main ()
5: {
6:     int x=4,y=7;
7:     get2(x,y);
8:     printf("first no:is %d secand no:%d",x,y);
9: }
10: void get2(int xx,int yy)
11: {
12:     xx+=5;
13:     yy+=5;
14: }
```

الشكل رقم ٨-١ إنشاء دالة داخل البرنامج

وعن هذا البرنامج نوضح ما يلى :

يبدأ البرنامج الموجود في الشكل رقم ٨-١ في السطر رقم ٣ بالإعلان عن الدالة () get2 التي تأخذ معاملان من نوع صحيح وفي السطر رقم ٧ يتم استدعاء الدالة مع ارسال قيمتين صحيحتين لها تأخذهما الدالة الى السطر رقم ١٠ وتضعهما في المتغيرين xx,yy .

وفي السطرين رقم ١٢ و ١٣ يتم اضافة القيمة ٥ الى كل من المتغيرين ثم تقوم الدالة الرئيسية في السطر رقم ٨ بطباعة قيم المتغيرين.

والسؤال هنا ما هي القيم التي يطبعها البرنامج؟ هل يطبع البرنامج القيم بعد اضافة القيمة ٥ الى كل متغير كما في سطور الدالة أما يطبعها كما هي؟ قبل أن تتابع شرح البرنامج حاول الوقوف والتفكير في ذلك.

قد تظن للوهلة الأولى أن النتيجة هي ٩ و ١٢ وذلك بعد اضافة القيمة ٥ الى القيمتين ٤ ، ٧ في حين أنك لو دققت النظر ستجد أن الدالة من النوع void كما في الاعلان في السطر رقم ٣ وبالتالي الدالة لا تعيد قيم وهذا ماتم حيث قامت الدالة باستقبال القيم واضافة القيمة ٥ الى كل عنصر ، ولكن لم تعيد الدالة النتيجة وبالتالي تظل قيم المتغيرات كما هي ٤ و ٧ وحتى لو أعلنا أن نوع الدالة int فلن تستطيع الدالة إعادة أكثر من قيمة. اذا كيف نعيد قيم المتغيرين في البرنامج السابق بعد اضافة القيمة ٥ اليهما هذا ما نراه من خلال البرنامج الموجود في الشكل رقم ٨-٢ .

```

0: /*Program Name CS8_2.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: void rets(int *xx,int *yy);
4: main ()
5: {
6:     int *x=5,*y=10;
7:     rets(&x,&y);
8:     printf ("first no:is %d second is %d",x,y);
9: }
10: void rets (int *xx,int *yy)
11: {
12:     *xx+=5;
13:     *yy+=10;
14: }
```

شكل رقم ٨-٢ استخدام المؤشرات مع الدوال

وعند تفريذ البرنامج تحصل على النتيجة التالية

first no:is10 secand is 20

وعن هذا البرنامج نوضح ما يلى :

في السطر رقم ٣ اعلان عن دالة لها معاملان وهذا المعاملان من نوع مؤشر الى قيمة صحيحة ، ومعنى أن المعاملات من نوع مؤشرات أثنا في حالة استدعاء الدالة لن نرسل الى الدالة قيم ولكن نرسل الى الدالة عناوين هذه القيم ، وهذا ما تم في السطر رقم ٧ حيث تم ارسال عناوين المتغيران u, x وذلك بالصورة $\&x, \&y$ فوجود العلامة $\&$ مع المتغير يجعل المتغير يشير الى عنوان المكان وليس القيمة المخزنة في المتغير ففي هذا السطر يتم ارسال عناوين المتغيرين u, x الى الدالة $rets()$ التي تقوم باستقبال العناوين والتعويض بهما في المتغيرين yy, xx ، وتقوم الدالة بزيادة القيم الموجودة في هذه العناوين وبالتالي تكون نتيجة هذا البرنامج هي طباعة القيم بعد زيادة القيمة المخزنة في المتغير. وبهذا الاسلوب كأننا اعادنا قيمتين من الدالة. هل لاحظت فكرة ارسال عنوان الى دالة اذا ما فائدة هذه الفكرة ؟

الفائدة هي امكانية انشاء دالة تقوم بعمليات كثيرة وتخرج أكثر من ناتج أما بدون استعمال المؤشرات فلا تستطيع ان تعيد هذه القيم الى الدالة الرئيسية لأنك لا تستطيع استعمال أكثر من جملة `return`.

ولكن يمكن كما في هذا المثال أن نستدعي الدالة بحيث نرسل لها عناوين أي عدد من المتغيرات والدالة بدورها تجري العمليات المطلوبة ثم تضع الناتج في هذه العناوين ونستخدمها بعدها من داخل الدالة الرئيسية.

نحضر للإعلان عن مؤشر تضع العلامة `*` قبل المتغير وللتعامل مع عنوان المكان مع الدالة يسبق المتغير بالعلامة `&` فمثلاً تكتب `p*` للإعلان عن مؤشر حسب النوع

ونكتب `&p` لإرسال عنوان المكان

المؤشرات والمصفوفات Pointers and Arrays

للمؤشرات دور مع المصفوفات حيث تتعامل مع عناوين عناصر المصفوفات ، وهذا بالطبع أسرع من الطريقة المعتادة. وقبل أن نوضح كيف تتعامل المؤشرات مع المصفوفات نورد مثلاً يذكرنا بالتعامل مع المصفوفات بدون استعمال المؤشرات. والبرنامج الموجود بالشكل رقم ٨-٣ يقوم بالإعلان عن مصفوفة واعطائها قيم ابتدائية ثم يقوم بطباعة هذه القيم على الشاشة.

```
0: /*Program Name CS8_3.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     int nums[ ]={92,81,70,69,58};
6:     int dex;
7:     for (dex=0;dex<5;dex++)
8:         printf ("\t%d",nums[dex]);
9: }
```

شكل رقم ٨-٣ التعامل مع المصفوفة بدون مؤشرات

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

92 81 70 69 58

اما في حالة استخدام المؤشرات مع المصفوفات فيتم التعامل مع عناصر المصفوفة عن طريق اسم المصفوفة ، حيث يعتبر اسم المصفوفة هو عنوان أول عنصر فيها فمثلاً المصفوفة [10] يعتبر اسم المصفوفة وهو a هو عنوان أول عنصر. فإذا طبعنا قيمة المتغير a نحصل على عنوان أول عنصر في المصفوفة ولطباعة قيمة أول عنصر نسبق اسم المصفوفة بالعلامة * وبال التالي الصورة a* تعبّر عن قيمة أول عنصر واذا

أضفنا ١ الى عنوان المصفوفة ستحصل على عنوان ثانى عنصر فمثلاً $(a+1)^*$ تشير الى قيمة ثانى عنصر وهكذا . والبرنامج الموجود بالشكل رقم (٤-٨) يوضح ذلك

```

0: /*Program Name CS8_4.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     int nums[ ]={92,81,70,69,58};
6:     int dex;
7:     for (dex=0;dex<5;dex++)
8:         printf ("%d",*(nums+dex));
9: }
```

الشكل رقم ٤-٨ استخدام المؤشرات مع المصفوفات

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٥ اسم المصفوفة هو `nums` وهو فى نفس الوقت عنوان أول عنصر كما اشرنا بمعنى لو كتبت السطر `printf("%d",nums);` سوف يطبع لك عنوان أول عنصر تم حجزه لهذه المصفوفة حيث أنه في حالة الاعلان عن مصفوفة يتم حجز أماكن بعد العناصر ولكن يشار الى أول عنصر فقط باسم المصفوفة وتميز نهاية المصفوفة العلامة '٠' فإذا وضعنا العلامة * بجانب `nums` بالشكل `*nums`؟ فإننا نشير الى القيمة الموجودة بهذا العنوان وبالتالي الصورة `printf ("%d",*nums);` تطبع قيمة أول عنصر.
- والصورة `(*%d",nums+1)` تطبع عنوان العنصر الثاني
- والصورة `((%d",*(nums+1))` تطبع قيمة العنصر الثاني
- والصورة `printf ("%d",nums+2);` تطبع عنوان العنصر الثالث
- والصورة `printf ("%d",*(nums+2));` تطبع قيمة العنصر الثالث وهكذا

وهذا ما يتم في البرنامج السابق حيث يتم زيادة العنوان بمقدار واحد بزيادة قيمة المتغير dex في السطر رقم 8
وبالتالي طباعة عناصر المصفوفة وبهذه الطريقة تعامل المؤشرات مع المصفوفات
وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

92 81 70 69 58

رسال المصفوفة إلى الدالة كمعامل

شرحنا في درس الدوال أن معامل الدالة الذي نرسله لها عند الاستدعاء يمكن أن يكون متغير صحيح أو متغير حقيقي ويمكن أيضاً أن يكون عنوان مكان فكيف تكون المصفوفة معامل للدالة؟

يتم ذلك بارسال اسم المصفوفة الذي هو عنوانها وعدد العناصر إلى الدالة وبالتالي تأخذ الدالة الاسم (العنوان) وعدد العناصر وتعامل مع هذه العناصر حسب العمليات الموجودة بالدالة. والبرنامج الموجود بالشكل رقم ٨-٥ يوضح كيفية استعمال المصفوفة كمعامل للدالة

```
0: /*Program Name CS8_5.C*/
1: #define SIZE 5
2: void addcon(int *ptr,int num,int con);
3: main ()
4: {
5:     int array[SIZE]={3,5,7,9,11};
6:     int konst=10;
7:     int j;
8:     addcon(array,SIZE,konst);
9:     for (j=0;j<SIZE;j++)
10:        printf ("%d\n",*(array+j));
11: }
12: void addcon(int *ptr,int num,int con);
```

```
13: {  
14:     int k;  
15:     for (k=0;k<num;k++)  
16:         *ptr=*(ptr++)+con;  
17: }
```

شكل رقم ٨-٥ استعمال المصفوفة كمعامل للدالة

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٢ اعلان عن دالة لها ثلاث معاملات الأول مؤشر (pointer) والثاني والثالث رقم صحيح
- في السطر رقم ٥ اعلان عن مصفوفة واعطائها قيم ابتدائية
- في السطر رقم ٨ استدعاء للدالة () addcon() وتأخذ ثلاث معاملات الاول اسم المصفوفة (عنوان المصفوفة) والثاني size وهو عدد عناصر المصفوفة والثالث رقم صحيح مقداره ١٠ ليتم جمعه على كل عنصر من عناصر المصفوفة
- في السطر رقم ١٢ تبدأ الدالة وتستقبل المعاملات كما أشرنا اليها
- وفي السطر رقم ١٦ تعامل الدالة مع عناوين العناصر فتجمع على كل قيمة داخل كل عنوان القيمة الثابتة المرسلة ١٠
- وفي الدالة الرئيسية في السطر رقم ١٠ يتم طباعة عناصر المصفوفة بعد استدعاء الدالة.

وعند تفريذ هذا البرنامج نحصل على النتيجة التالية :

13 15 17 19 21

المؤشرات والعبارات الحرفية Pointers and strings

كما أن للمؤشرات (Pointers) دور مع المصفوفات فلها أيضا دوراً مع عبارات الحرفية. والحقيقة أن للمؤشرات دور مع جميع عناصر اللغة فلها دور مع الملفات (FILES) ولها دور مع دوال التعامل مع الذاكرة MEMORY ALLOCATION() وبالتالي لابد من فهم موضوع المؤشرات فهماً جيداً. تستخدم

المؤشرات مع العبارات الحرفية لزيادة سرعة تفيد البرنامج . وطريقة الاعلان المعتادة عن العبارات الحرفية بدون استخدام المؤشرات تأخذ الشكل التالي

```
char name[10];
```

وهو عبارة عن سلسلة من الحروف عدد ها ١٠ ولكن مع المؤشرات هناك طريقة أخرى للإعلان عن العبارة الحرفية مثل "SAMY" char *name="SAMY" هذا الإعلان يعني أن المتغير name مؤشر يشير إلى عبارة حرفية ولكن ما طول هذه العبارة الحرفية ؟ يحدد ذلك من عدد حروف القيمة المعطاة , وهذه الحالة عدد الحروف هو أربعة حروف وهو عدد حروف كلمة SAMY.

والبرنامج الموجود بالشكل رقم ٨-٦ يوضح كيفية استعمال المؤشرات في الإعلان عن عبارة حرفية

```
0: /*Program Name CS8_6.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     char *saluate="GREETING MR.";
6:     char name[7];
7:     puts("Enter your name:");
8:     gets(name);
9:     puts(saluate);
10:    puts(name);}
```

الشكل رقم ٨-٦ استعمال المؤشرات مع العبارة الحرفية

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٤ يتم الإعلان عن مؤشر من نوع حرفى واعطائه قيمة ابتدائية هي الجملة GREETING MR وهذا شكل آخر للإعلان عن العبارة الحرفية

- باستخدام المؤشرات وفى السطر رقم ٧ استقبال كلمة بالدالة gets() وتخزينها بالمتغير name
- وفى السطر رقم ٨ طباعة محتويات المتغير saluate وهو الجملة .GREETING MR
 - في السطر رقم ٩ طباعة محتويات المتغير name وهو SAMY بعد استقباله وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

Enter your name:

SAMY

GREETING MR.

SAMY

مصفوفة المؤشرات Array of Pointer

كما يمكن استخدام المؤشرات في الاعلان عن عبارة حرفية ، يمكن أيضا استخدام المؤشرات في الاعلان عن مصفوفة حرفيات والبرنامج الموجود بالشكل رقم ٨ يقوم بالاعلان عن مصفوفة الحروف واعطائها مجموعة اسماء كقيم ابتدائية ثم يقوم البرنامج باستقبال اسم من المستخدم ويقارن هذا الاسم في المصفوفة بمجموعة الاسماء المخزنة في المصفوفة فإذا وجد هذا الاسم في المصفوفة يطبع البرنامج رسالة بذلك وإذا لم يجده يطبع رسالة اخرى.(وهو شكل اخر من البرنامج الموجود في فصل المصفوفات)

```
0: /* Program Name CS8_7.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: #define MAX 5
4: main ()
5: {
6:     int d;
7:     int enter=0;
8:     char name[40];
```

```

9: static char *list[MAX]=
10:   { "azab",
11:     "hamdy",
12:     "samy",
13:     "nabil",
14:     "mona" };
15: clrscr();
16: while(enter !=1) {
17:   printf ("\nEnter your name:");
18:   gets(name);
19:   for(d=0;d<MAX;d++)
20:     if(strcmp(list[d],name)==0)
21:       enter=1;
22:     if (enter==1)
23:       printf ("\\nyour name is found..");
24:     else
25:       printf("\\nsorry your name not found.");
26:   }
27: }
```

الشكل رقم ٨-٧

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٩ اعلان عن مصفوفة عناصرها من نوع حرفيات (كلمات) مع اعطائها قيم ابتدائية هي مجموعة اسماء
- وفي السطر رقم ١٦ الدوارة while للاستمرار حتى شرط $!=1$ enter
- وفي السطر رقم ١٨ الدالة gets() تستقبل مجموعة حروف عبارة عن اسم من المستخدم
- في السطر رقم ١٩ الدوارة for بعد عناصر المصفوفة لتكرار عملية المقارنة بين الاسم المستقبل والاسماء المحذنة في المصفوفة

- في السطر رقم ٢٠ مقارنة بين كل عنصر من عناصر المصفوفة والاسم الذي أدخله المستخدم فإذا كان الاسم موجوداً ضمن الأسماء الموجودة في المصفوفة تظهر الرسالة.. your name is found.. وينتهي البرنامج والا ظهرت الرسالة sorry your name not found. وهكذا

في البرنامج السابق في السطر رقم 20 يمكن استبدال [d] list بالصورة *



وهذا البرنامج يصلح ليصبح برنامج كلمة سر ولكن به مجموعة من الأسماء اذا كان المستخدم أحد هذه الأسماء سمح له البرنامج بالدخول في العمل وأكمل له خطوات التشغيل والا لم يسمح له بالدخول في البرنامج

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter your name:ahmed  
sorry your name not found.  
Enter your name:mohamed  
sorry your name not found.  
Enter your name:samy  
your name is found..
```





الفصل التاسع

السجلات

STRUCTURES

في هذا الفصل نشرح نوع من أنواع البيانات المهمة وهو السجل (structure) وهو مفيد بالدرجة الأولى في برامج قواعد البيانات.

يقابل كلمة **Record** في برمج قواعد البيانات كلمة **Structuer** ويتناول هذا الفصل الموضوعات التالية :

- ◆ معنى السجل وال الحاجة الى استعماله
- ◆ انشاء السجل والاعلان متغير من السجل
- ◆ التعامل مع عناصر السجل (الحقول)
- ◆ استعمال السجل كعنصر في سجل آخر
- ◆ **Array Of Structure**
- ◆ المؤشرات والسجلات
- ◆ **Type Casting**
- ◆ اتحاد العناصر تحت اسم واحد **union**
- ◆ الفرق بين **union** و **structure**

معنى السجل (STRUCTURE) وال الحاجة الى استعماله

من أهم التطبيقات في عالم البرامج تطبيقات قواعد البيانات فمثلاً قاعدة بيانات موظفين تمثل بيانات الموظفين في شكل سجلات كل سجل يتكون من مجموعة حقول ولو أن لك خبرة بأحد برامج قواعد البيانات مثل dbase فستعرف أن الملف ينقسم إلى سجلات (records) والسجل ينقسم إلى حقول (fields) ودائماً نحتاج للتعامل مع السجل كوحدة وكذلك مع الحقول كوحدة. وتستخدم لغة C كلمة Structure بنفس المفهوم الذي تستخدمه لغات البرمجة الأخرى لكلمة Record .

استعمال السجل (STRUCTURE)

هناك خطوات تبع للتعامل مع السجل وهي إنشاء السجل (تركيب السجل) وتحديد الحقول المطلوبة ثم الإعلان عن متغير من نوع هذا السجل ثم التعامل مع حقول هذا السجل. والبرنامج الموجود بالشكل ٩-١ يشتمل على هذه الخطوات

```
0: /*Program Name CS9_1.C */
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     struct data
6:     {
7:         int num;
8:         char stat;
9:     };
10:
11:     struct data stud;
12:     stud.num=5;
13:     stud.stat='t';
```

```
14:     printf ("\n stud.num=%d,stud.stat=%c",stud.num,stud.stat);  
15: }
```

الشكل ٩-١ برنامج إنشاء السجل و استخدامه

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٥ يبدأ إنشاء السجل وذلك باستعمال كلمة struct واعطاء هذا السجل اسم وهو data وكلمة data ممكن أن تكون أي كلمة.
- في السطر رقم ٦ تبدأ مكونات هذا السجل بالقوس { وفي السطرين رقم ٧ ورقم ٨ اعلان عن حقول السجل وهي عبار عن متغير من نوع صحيح ومتغير من نوع حرف وينتهي السجل في السطر رقم ٩ بالقوس }
- في السطر رقم ١١ يتم الإعلان عن متغير من نوع السجل وهو المتغير stud وبالتالي أخذ المتغير stud نفس التركيب فأصبح له عنصر اسمه num من نوع صحيح وكذلك عنصر من نوع حرف وهو stat.
- في السطرين رقم ١٢ ورقم ١٣ تم اعطاء قيم لحقول السجل ولكن الملاحظ أنه للتعامل مع حقل في سجل يتم كتابة اسم الحقل مسبوقا باسم السجل التابع له وبينهما نقطة بالصورة stud.num
- في السطر رقم ١٤ يتم طباعة قيم حقول السجل وبينهما الأسلوب تم كتابة اسم الحقل مسبوقا باسم السجل وبينهما النقطة للإشارة أن هذا الحقل تابع لهذا السجل.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
stud.num=5,stud.stat=t
```

يمكن الإعلان عن أكثر من متغير من نوع السجل كما يحدث مع أنواع البيانات الأخرى فمثلا انظر الإعلان التالي

```
struct data stud1,stud2,stud3
```

و معناه أن المتغيرات stud1,stud2,stud3 من نوع data وبالتالي لها نفس العناصر.



و كذلك يمكن الإعلان عن متغيرات من نوع structure بطريقة أخرى نوضحها

فيما يلى :

```
struct data
```

```
{  
    int num;  
    char ch;  
}stud1,stud2;
```

ويعطى هذا الإعلان نفس النتيجة السابقة حيث أصبح stud1,stud2 لهما نفس تركيب السجل.

كيفية إدخال بيانات إلى عناصر السجل structure

يمكن معاملة عناصر السجل structure معاملة المتغيرات العاديّة حيث يمكن اعطائها قيم كما سبق ويمكن استقبال قيم بدوال الاستقبال من المستخدم ووضع هذه القيم في عناصر السجل والبرنامج الموجود في الشكل رقم ٩-٢ يقوم بإنشاء سجل واستقبال قيم عناصره من المستخدم ثم طباعة هذه القيم على الشاشة

```
0: /* Program Name CS9_2.C*/  
1: #include <stdio.h>  
2: main ()  
3: {  
4:     struct data  
5:     {  
6:         int no;  
7:         char name[10];  
8:     };  
9:     struct data stud;  
10:    printf("\n\n stud.no=");  
11:    scanf("%d",&stud.no);  
12:    printf("\n\n stud.name=");  
13:    scanf("%s",stud.name); /*Notes name is string*/
```

الفصل التاسع : STRUCTURES

```
14: clrscr();  
15: printf("\n\n stud.no=%d",stud.no);  
16: printf("\n\n stud.name=%s",stud.name);  
17: }
```

شكل رقم ٩-٩ إنشاء السجل و استقبال عناصره ثم طباعتها

وعن هذا البرنامج نوضح ما يلى

- من السطر رقم ٤ الى السطر رقم ٨ تم انشاء السجل
 - في السطر رقم ٩ تم الاعلان عن متغير من نوع السجل
 - في السطر رقم ١١ استخدمنا دالة الاستقبال scanf() لاستقبال عناصر السجل
- وهنا يتضح الفرق بين التعامل مع متغير عادي ومتغير عنصر في سجل وهو أننا نسب عنصر السجل الى السجل التابع له وذلك عن طريق النقطة. بالصورة stud.no و معناها الاشارة الى العنصر no التابع للسجل المسمى stud.
- و يتم التعامل مع عناصر السجل كما يتم التعامل مع المتغيرات العاديّة.

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية:

```
stud.no=1  
stud.name=mohamed  
stud.no=1  
stud.name=mohamed
```

وضع محتويات سجل في آخر

درسنا من قبل امكانية مساواة متغيرين من نوع واحد وذلك لوضع قيمة المتغير الأول في المتغير الثاني وذلك بالصورة التالية :

```
int a,b;  
b=5;  
a=b;
```

وهذا معناه وضع القيمة المخزنة في المتغير b وهي القيمة ٥ في المتغير a ويمكن تحقيق ذلك مع السجلات بحيث يمكن مساواة متغير من نوع سجل مع آخر وبالتالي يتم مساواة قيم جميع العناصر بين السجلين بشرط أن يكون السجلين من نفس النوع، والبرنامج الموجود بالشكل ٩-٣ يوضح ذلك

```
0: /* Program Name CS9_3.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main ()
4: {
5:     struct data
6:     {
7:         int num;
8:         char na;
9:     };
10:
11:    struct data stud1,stud2;
12:    stud1.num=5;
13:    stud1.na='t';
14:    stud2=stud1;
15: printf("\n stud1.num=%d,stud1.na=%s",stud1.num,stud1.na);
16: printf("\n stud2.num=%d,stud2.na=%s",stud2.num,stud2.na);
17 }
```

شكل ٩-٣ وضع محتويات سجل في سجل آخر

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٥ تم إنشاء سجل جديد
- في السطر رقم ١١ تم تعريف متغيرين من نوع السجل هما stud1,stud2
- في السطر رقم ١٤ تم مساواة المتغيرين stud1 و stud2 وبالتالي وضع نسخة من القيم الموجودة في السجل stud1 في السجل الثاني stud2.

وعن تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
stud1.num=5,stud1.name=t  
stud2.num=5,stud2.name=t
```

السجلات المتداخلة Nested Structures

شرحنا أن السجل هو مجموعة من العناصر أياً كان نوع هذه العناصر وبالتالي يمكن أن تكون العناصر أو بعضها سجلات وهذا ما يسمى بالسجلات المتداخلة والبرنامج الموجود بالشكل رقم ٤-٩ يوضح كيف يكون السجل عصراً في سجل آخر وكيفية التعامل مع عناصر السجلات في هذه الحالة.

```
0: /* Program Name CS9_4.C*/  
1: #include <stdio.h>  
2: main ()  
3: {  
4:     struct person  
5:     {  
6:         int no;  
7:         char name[10];  
8:     };  
9:     struct group  
10:    {  
11:        struct person peno1;  
12:        struct person peno2;  
13:        int code;  
14:    };  
15:    struct group group1;  
16:    printf("\n\n group1.peno1.no=");  
17:    scanf("%d",&group1.peno1.no);  
18:    printf("\n group1.peno1.name=");  
19:    scanf("%s",group1.peno1.name);  
20:    printf("\n group1.code=");  
21:    scanf("%d",&roup1.code);
```

```

22:     group1.peno2=group1.peno1;
23:     clrscr();
24:     printf("\n\n the data of groups:\n\t");
25:     printf("\n group1.peno1.no=%d",group1.peno1.no);
26:     printf("\n group1.peno1.name=%s",group1.peno1.name);
27:     printf("\n group1.code=%d",group1.code);
28:     printf("\n group1.peno2.no=%d",group1.peno2.no);
29:     printf("\n group1.peno2.name=%s",group1.peno1.name);
30: }

```

شكل رقم ٤-٩ السجلات المتداخلة

وعن هذا البرنامج نوضح ما يلى :

في هذا البرنامج تم الإعلان عن سجل يمثل بيانات اشخاص ثم تم الإعلان عن سجل آخر والعنصر الثاني في هذا السجل هو سجل من نوع السجل الأول وأخذ الاسم peno1 ثم تم التعامل مع عناصر السجلات كما في سطور البرنامج كما يلى

- من السطر رقم ٤ إلى السطر رقم ٨ تم إنشاء السجل الأول وهو person وعناصره هي no و name ومن السطر رقم ٩ إلى السطر رقم ١٤ تم إنشاء السجل الثاني هو group وعناصره هي code ويمثل كود المجموعة والسطر رقم ١٥ إعلان عن متغير من نوع السجل الثاني
- من السطر رقم ١٦ إلى السطر رقم ٢٢ يتم استقبال العناصر ومن السطر رقم ٢٤ إلى السطر رقم ٢٩ يتم طباعة عناصر السجل مع ملاحظة أننا ننسب كل عنصر إلى السجل التابع له

و عند تفريذ هذا البرنامج نحصل على النتيجة التالية :

```

group1.peno1.no=1
group1.peno1.name=hamdy
group1.code=01
the data of group:

```

```
group1.peno1.no=1
group1.peno1.name=hamdy
group1.code=01
group1.peno2.no=1
group1.peno2.name=hamdy
group1.code=01
```

السجلات والدوال

وهنا نناقش

- استعمال السجل كمعامل للدالة
- اعادة السجل من الدالة

درستنا في فصل الدوال أن الدالة يمكن أن تستعمل معاملات من أي نوع من البيانات
وبالتالي يمكن أن يكون هذا المعامل من نوع سجل

يعنى انه يمكن أن نرسل للدالة سجل كمعامل والدالة بدورها تقوم بأى عمليات على هذا السجل وبالمثل يمكن للدالة أن تقوم ببعض العمليات على السجل ثم تعيد سجل الى الدالة الرئيسية ويكون نوع الدالة من نوع هذا السجل والبرنامج الموحد بالشكل ٩-٥ يقوم بالاعلان عن دالتين الاولى تقوم باستقبال عناصر سجل من المستخدم وعند استدعائهما تعيد هذا السجل الى الدالة الرئيسية والدالة الثانية تأخذ هذا السجل كمعامل وتقوم بطباعة بياناته على الشاشة

```
0: /*Program Name CS9_5.C*/
1: struct personal
2: {
3:     char name[30];
4:     int numb;
5: };
6: struct personal addname(void);
7: void display(struct personal custm);
7: main ()
```

```
8: {
9:     struct personal custmer1;
10:    struct personal custmer2;
12:    custmer1=addname();
13:    custmer2=addname();
14:    display(custmer1);
15:    display(custmer2);
16: }
17: /* addname() */
18: struct personal addname()
19: {
20:     char numstr[81];
21:     struct personal custmer;
22:     printf ("\n new custmer \n Enter name:");
23:     gets(custmer.name);
24:     printf ("\n Enter custmer no:");
25:     gets(numstr);
26:     custmer.numb=atoi(numstr); /*convert string to
                                integer*/
27:     return custmer;
28: }
29: /* list() */
30: void display(struct personal custm)
32: {
33:     printf("\n Custmers:\n");
34:     printf ("Name:%s\n",custm.name);
35:     printf ("Number:%d \n",custm.numb);
36: }
```

شكل ٩-٥ السجلات والدوال

وعن هذا البرنامج نوضح مايلي :

في السطر رقم ٦ تم الإعلان عن الدوال () display() و () addname(). والدالة struct personal addname() من نوع struct personal فمادا يعني أنها من هذا النوع ، نعرف أن نوع الدالة يعتمد على نوع القيمة التي تعدها الدالة ولو نظرت الى كلمة return في الدالة struct في السطر رقم ٢٧ تجد أنها تعيد structure من نوع addname()

struct personal لذلك لابد أن يكون نوعها من نفس نوع القيمة التي تعدها وهو structure. والدالة display لا تعيد قيم بل تقوم بطباعة قيم عناصر السجل personal فقط لذلك كان نوعها void

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
new custmer
Enter name:samy
Enter custmer no:1
new custmer
Enter name:hamdy
Enter custmer no:2
Custmer:
Name:samy
Number:1
Custmer:
Name:hamdy
Number:2
```

مصفوفة السجلات Arrags Of Structures

درستنا في درس المصفوفات أن المصفوفة هي مجموعة من العناصر من نفس النوع هذا النوع يمكن أن يكون أي نوع من البيانات وبالتالي يمكن أن نعلن عن

مصفوفة عناصرها من نوع سجلات والبرنامج الموجود بالشكل ٩-٦ يقوم بإنشاء سجل ثم الإعلان مصفوفة من هذا السجل ثم استقبال قيم عناصر هذه المصفوفة ثم طباعتها

```
* Prgram Name CS9_6.C*
#include <stdio.h>
main ()
{
    int i;
    struct personal
    {
        char name[30];
        int numb;
    };
    struct personal custmer[5];
    for(i=0;i<5;i++)
    {
        printf("\n custmer.no[%d]=",i);
        scanf("%d",& custmer[i].numb);
        printf("\n custmer.nname[%d]=",i);
        scanf("%s", custmer[i].name);
    }
    for(i=0;i<5;i++)
    {
        printf("\n %d lt %s", custmer[i].numb, custmer[i].name);
    }
}
```

شكل ٩-٦ إستخدام مصفوفة السجلات

المؤشرات والسجلات

ذكرنا أن المؤشرات (pointers) تدخل في كل عناصر اللغة وبالتالي لها دور مع السجلات (structures) وهنا نناقش كيفية الإعلان عن مؤشر من نوع structure .(structures) وكيفية استعمال هذا المؤشر مع السجلات (pointer to structure)

الفرق بين اعلان مؤشر الى سجل واعلان متغير من نوع سجل هو أن المؤشر يسبق بالعلامة * ونوضح ذلك من خلال البرنامج الموجود بالشكل رقم ٩-٧

```
0: /* Program Name CS9_7.C */  
1: void main (void)  
2: {  
3:     struct xx  
4:     {  
5:         int num1;  
6:         char ch;  
7:     };  
8:     struct xx xx1;  
9:     struct xx *ptr;  
10:  
11:    ptr=&xx1;  
12:    ptr->num1=303;  
13:    ptr->ch='q';  
14:    printf ("ptr->num1=%d\n",ptr->num1);  
15:    printf ("ptr->ch=%c",ptr->ch);  
16: }
```

شكل رقم ٩-٧ استخدام المؤشرات مع السجلات

وعن هذا البرنامج نوضح ما يلى

في السطر رقم ٣ بدأ البرنامج بإنشاء سجل ، وفي السطر رقم ٨ تم الإعلان عن متغير من نوع هذا السجل ، وفي السطر رقم ٩ تم الإعلان عن مؤشر إلى هذا السجل وفي السطر رقم ١١ تم تخزين عنوان السجل xx1 في المتغير ptr وبعد ذلك يتم التعامل مع السجل عن طريق التعامل مع المؤشر إلى هذا السجل كما في السطر رقم ١٢ والسطر رقم ١٣ ونلاحظ أن التعامل مع عنوان السجل وعنابر السجل هو نفس التعامل مع متغير السجل وعنابرها غير أنها تستبدل الصورة xx.no بالصورة ptr->no حيث أن ptr عبارة عن مؤشر إلى السجل

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية

ptr->num1=303

ptr->ch=q

تغبيـر نوع البيانات TYPECASTING

من العمليات المفيدة في لغة C وجود أكثر من طريقة لتحويل بيانات من نوع

إلى نوع آخر وتشتم بطرقين:

الطريقة الأولى : استخدام مجموعة دوال مثل الدالة (atoi) بمعنى ascii to int وهي تقوم بتحويل الأرقام التي في صورة حروفيات إلى صورة أرقام للتعامل معها كأرقام والدالة (atof) بمعنى ascii To float والتي ذكرناها في فصل المصفوفات.

الطريقة الثانية : هي ما يسمى Typecasting وهي كتابة النوع المطلوب التحويل إليه قبل المتغير المطلوب تغيير نوعه. والبرنامج الموجود في الشكل رقم ٩-٨ يوضح فكرة تغيير النوع باستخدام هذه الطريقة

```
0: /* Prgram Name CS9_8.C*/
1: #include<stdio.h>
2: main ()
3: {
4:     int a=66,b;
5:     char ch='t',g;
6:     b=(int) ch;
7:     g=(char) a;
8:     printf("b=%d",b);
9:     printf("\n g=%c",g);
10: }
```

شكل رقم ٩-٨ برنامج تغيير نوع البيانات

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية:

STRUCTURES : الفصل التاسع

b=116

g=B

وعن هذا البرنامج : نوضح ما يلى :

يبدأ البرنامج في السطر رقم ٤ بالاعلان عن متغيرين من نوع صحيح هما b و السطر رقم ٥ اعلان عن متغيرين من نوع حرف وهما ch,g وفي السطر رقم ٦ كتابة الكلمة int قبل المتغير ch يؤدى الى تحويل الحرف الذي خزن في المتغير ch الى كوده وبالمثل السطر رقم ٧ وهذا ما يسمى typecasting وبالتالي نحصل على النتيجة السابقة.

اتحاد البيانات تحت اسم واحد UNION

من أنواع البيانات المشتقة الموجودة في لغة C نوع يسمى union وهو يشبه بدرجة كبيرة السجل (STRUCTURE) مع وجود فارق بينهما وقبل معرفة هذا الفارق نوضح شكل UNION من خلال البرنامج الموجود بالشكل رقم ٩-٩

```
0: /* Prgram Name CS9_9.C*/
1: void main (void)
2: {
3:     union intflo
4:     {
5:         int intnum;
6:         float fltnum;
7:     }unex;
9:     printf ("sizeof (union intflo)=%d\n",sizeof(union intflo);
10:    unex.intnum=734;
11:    printf ("unex.intnum=%d\n",unex.intnum);
12:    unex.fltnum=54.4;
13:    printf ("unex.fltnum=%,.2f\n",unex.fltnum);
14: }
```

الشكل رقم ٩-٩ إستخدام union داخلي البرنامج

وعن هذا البرنامج نوضح ما يلى :

في البرنامج الموجود بالشكل رقم ٩-٩ في السطر رقم ٣ تم انشاء union ولا يختلف انشاءه عن انشاء structure ويتم التعامل معه كما يتم التعامل مع structure كما يظهر ذلك من سطور البرنامج ولكن الفرق بين union و structure هو أن union يحجز مساحة في الذاكرة مقدارها مجموع عناصره بينما الـ structure يحجز مساحة مقدارها مساحة أكبر عنصر فلو تأملت البرنامج السابق تجد أنه من السطر رقم ٣ إلى السطر رقم ٨ تم تعريف union وتركيه عبارة عن متغير صحيح و متغير حقيقي، ومن المفترض أن تكون المساحة الكلية لهذا الـ union هي مجموع مساحة المتغير الصحيح وال حقيقي. أي ٦ بايت هذا ما يحدث مع السجل ولكن مع union يتم حجز مساحة أكبر عنصر وهو في هذه الحالة المتغير الحقيقي أي ٤ بايت

و عند تنفيذ هذا البرنامج نحصل على النتيجة التالية

```
sizeof(union inflo)=4  
unex.intnum=734  
unex.fltnum=54.4
```

لماذا نستخدم UNION

يمكّنا استعمال الـ union من استقبال أكثر من نوع من البيانات على متغير واحد ويظهر ذلك في الشكل رقم ٩-١٠

```
union NO  
{  
    double dno;  
    float fno;  
    long lno;  
    int no;
```

```
char cno;  
};
```

الشكل ٩-١٠ استخدام union لاستقبال أكثر من نوع من البيانات

في حالة استعمال union يقوم union بحجز مساحة واحدة تستعمل لجمع عناصره وبالتالي كلما تم ادخال قيمة عنصر تلغى القيمة القديمة لأنها مساحة واحدة

في هذا المثال تم انشاء union عناصره double,float,long,int,char وكما اتفقنا أن union يحجز في الذاكرة RAM مساحة مقدارها مساحة أكبر عنصر فقط وفي نفس الوقت يمكن استقبال جميع العناصر ويتم استقبالها على نفس المساحة وبالتالي تمكينا هذه الفكرة من استقبال عدة أنواع من البيانات وكأنه متغير واحد

استعمال structure كعنصر من عناصر union

من أهم التطبيقات المشهورة لاستخدام union انشاء union عناصره وسوف نشرح هذا المفهوم بالتفصيل في الفصل (ROM BIOS) ويسمى structures .union of structures

ويوضح ذلك في البرنامج الموجود بالشكل رقم ٩-١١

```
/* Program Name CS9_11.C */  
void main ()  
{  
    struct twoInts  
    {  
        int intnum1;  
        int intnum2;  
    };
```

```
union intflo
{
    struct twoints setx;
    float fltnum;
} unex;
printf ("sizeof (union intflo)=%d\n",sizeof(union intflo));
unex.setx.intnum1=723;
unex.setx.intnum2=-455;
printf ("unex.setx.intnum1=%d\n",unex.setx.intnum1);
printf ("unex.setx.intnum2=%d\n",unex.setx.intnum2);
unex.fltnum=875.45; /* we print structure element first */
printf("unex.fltnum=%f\n",unex.fltnum);
}
```

شكل ١١-٩ إستعمال union كعنصر من عناصر structure



الفصل العاشر

الملفات

باستعمال الملفات يمكن حفظ البيانات بصفة دائمة،
ولغة C تتميز بأن بها طرق كثيرة للتعامل مع الملفات تناولتها
خلال الموضوعات التالية :

- ◆ الطرق المختلفة للتعامل مع الملفات
- ◆ الكتابة والقراءة حرف بحرف
- ◆ دوال فتح الملف وغلقده
- ◆ كتابة وقراءة عبارة حرفية كل مرة
- ◆ التعامل مع الطابعة والملفات الفاتحة
- ◆ كتابة وقراءة بيانات صحيحة وحقيقة وحرفية
- ◆ قراءة وكتابة سجل في كل مرة
- ◆ قراءة وكتابة مجموعات من البيانات

ربما يكون التعامل مع الملفات في لغة C مختلفاً شيء ما عن التعامل مع الملفات في لغات قواعد البيانات مثل dBASE وغيرها ، ولكن رغم سهولة التعامل مع الملفات في اللغات الأخرى مثل dbase فهي محدودة بعمليات معينة تحصر المبرمج في إطار ملفات قواعد البيانات ، أما لغة C فتتميز بأن بها طرق كثيرة للتعامل مع الملفات تمكنك من التعامل مع أنواع مختلفة من الملفات. وفيما يلى الطرق المختلفة للتعامل مع الملفات

الكتابة حرف بحرف في ملف (char by char)

من الطرق المتاحة في لغة C الكتابة في ملف حرف بحرف بمعنى إستقبال حرف من المستخدم وتخزينه مباشرة في ملف ، وقبل أن نبدأ في شرح قواعد التعامل مع الملف لنتابع البرنامج الموجود في الشكل رقم ١٠-١ الذي يسمح للمستخدم بكتابة حرف بحرف مع تخزين هذا الحرف في ملف ويستمر البرنامج في قبول الحرف حتى يضغط المستخدم على مفتاح الإدخال

```
0: /*Program Name CS10_1.C*/
1: #include <stdio.h>
2: main ()
3: {
4:     FILE *fptr;
5:     char ch;
6:     fptr=fopen("textfile.txt","w");
7:     printf("\n write to the file:\n");
8:     while ((ch=getche()) !='\r')
9:     {
10:         putc(ch,fptr);
11:     }
12: }
```

شكل ١٠-١ الكتابة في الملف حرف بحرف

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٤ اعلان عن متغير من نوع مؤشر الى ملف ويستعمل هذا المؤشر للإشارة الى الملف الذي يتم فتحه مع ملاحظة أن كلمة FILE تكتب بالحروف الكبيرة لأنها معروفة بلغة C (الملف Stdio.h) ، وهى تجعل المتغير ptr من نوع مؤشر الى FILE أي يشير الى ملف.
- وفي السطر رقم ٦ يتم فتح ملف بالاسم textfile والامتداد txt للكتابة فيه وذلك باستعمال الدالة fopen() وقبل أن نكمل شرح سطور البرنامج نشرح الدالة fopen والصورة العامة لدالة fopen هي.

`fptr=fopen("filename.ext","mode");`

- حيث المعامل filename هو اسم الملف المراد فتحه
- والمعامل ext هو امتداد الملف الذي تريد فتحه وفي حالة انشاء ملف جديد فإن تحديد هذا الامتداد اختياره أمر اختياري يرجع إليك ولذلك عند عمل برنامج متكامل يمكنك أن تحدد الامتداد الذي يروق لك
- والمعامل mode هو الحالة التي تريد فتح الملف من أجلها هل تريد فتح الملف لكتابته فقط ، أم للقراءة فقط ، أم للإضافة فقط ، أم للقراءة والكتابة ، يتم تحديد ذلك بكتابية الحرف الدال على الحالة المطلوبة ويوضح الجدول التالي الحرف التي تستخدم للحالات المختلفة :-

الحرف	متى يستخدم
-------	------------

a في حالة فتح ملف للإضافة وبالتالي تبدأ الكتابة من نهاية الملف
وما يكتبه المستخدم يضاف لمحتويات الملف الأصلي. واذا لم يكن الملف موجود يتم انشاءه والكتابة فيه

التعريف	مدى استخدامه
مثلاً الحالات السابقة ولكن تسمح هذه الحالة بالقراءة أيضاً	"a+"
فيتمكن لك استخدام أوامر الكتابة وأوامر القراءة مع نفس الملف المقترن بهذه الحالة	
فتح الملف للقراءة فقط وإذا لم يكن الملف المطلوب فتحه موجوداً على القرص أو إذا كان القرص غير صالح تعيد الدالة صفر أي لا تستطيع فتح الملف المذكور.	"r"
تسمح بالقراءة والكتابة أي استعمال دوال القراءة والكتابة مع الملف ولكن الكتابة لا تعنى الإضافة فهى تكتب لأول مرة أو تكتب فوق البيانات الموجودة	"w+"
فتح ملف جديد للكتابية فقط فإذا كان الملف المطلوب فتحه موجوداً على القرص تكتب البيانات الجديدة فوقه ويتم الغاء محتوياته	"w"
نفس الحالات السابقة ولكن تسمح بالقراءة والكتابة	"w+"

نعود إلى شرح البرنامج

- في السطر رقم 7 الدالة (getche) تستقبل حرف وتخزنـه في المتغير ch والدوارة while تجعل الاستقبال يستمر ما لم يضغط المستخدم على مفتاح الإدخال Enter.
- وفي السطر رقم 9 الدالة (putc) تكتبـ الحـرف المـوجـود فيـ المتـغـير ch فيـ المـلـفـ الذـى يـشـيرـ إلـيـهـ المتـغـيرـ fptrـ أـىـ فـيـ المـلـفـ textfile.txtـ وهـكـذاـ تـظـلـ الدـالـةـ (getche)ـ تـسـتـقـبـلـ حـرـفـ وـفـيـ نـفـسـ الـوقـتـ تـقـومـ الدـوارـةـ whileـ باـخـبارـ هذاـ الحـرـفـ وـتـقـومـ الدـالـةـ (putc)ـ بـكـتـابـةـ هـذـاـ الحـرـفـ فـيـ المـلـفـ،ـ فـاـذـاـ ضـغـطـ المستـخدمـ عـلـىـ مـفـتـاحـ الـادـخـالـ enterـ يـتـهـىـ عـلـىـ اـعـلـامـ الدـوارـةـ whileـ وـتـقـومـ الدـالـةـ (fclose)ـ بـأـغـلـاقـ المـلـفـ.

وعند تفريز هذا البرنامج نحصل على النتيجة التالية

write to the file :

allah the god of all world

القراءة من ملف حرف بحرف (char by char)

في البرنامج السابق والموجود بالشكل رقم ١٠-١ شرحنا كيف تم فتح ملف للكتابية فيه وكيف نستقبل حروف من المستخدم ونكتبها في هذا الملف ، والبرنامج الموجود بالشكل رقم ١٠-٢ يوضح كيفية فتح ملف للقراءة بطريقة حرف بحرف ثم طباعته على الشاشة

```
0: /*Program Name CS10_2.C*/
1: #include <stdio.h>
2: #include <stdlib.h>
3: main (void)
4: {
5:     FILE *fptr;
6:     char ch;
7:     clrscr();
8:     fptr=fopen("textfile.txt","r");
9:     printf("\n contents of the file:\n");
10:    while ((ch=getc(fptr)) !=EOF)
11:    {
12:        printf ("%c",ch);
13:    }
14:    fclose (fptr);
15:    getch();
16:    return 0;
17: }
```

الشكل رقم ١٠-٢ قراءة ملف حرف بحرف

و عن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٥ اعلان عن مؤشر الى ملف لاستعماله في الاشارة الى ملف حيث تم جميع العمليات مع الملف عن طريق هذا المؤشر
- وفي سطر رقم ٨ دالة فتح الملف وتلاحظ أن حالة الفتح حدثت هنا بالحرف ٢ أي فتح الملف للقراءة فقط.
- وفي سطر رقم ١٠ الدالة `getc(fp)` تقرأ حرف من الملف المشار اليه بالمؤشر `fp` وتضع هذا الحرف في المتغير `ch` وفي نفس الوقت تقوم الدوارة `while` بمقارنة هذا الحرف بالقيمة `EOF` وهو ثابت معرف بمعنى `END OF FILE` أي نهاية الملف وتستمر الدالة `(getc())` في قراءة حرف من الملف ما لم يكن هذا الحرف هو القيمة `EOF` أي ما لم تصل الى نهاية الملف كلها.
- وتقوم الدالة `(printf())` بطباعة هذا الحرف على الشاشة ويستمر ذلك حتى يتم عرض محتويات الملف.

و عند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

contents of the file:

allah the god of all world

المشاكل المتوقعة عند فتح ملف

- هناك الكثير من المشاكل المتوقعة عند فتح ملف منها ما يلى
- القرص المطلوب التعامل معه سواء لكتابه أو للقراءة غير صالح للاستعمال وبالتالي لا تستطيع الدالة `(fopen)` فتح الملف
 - الملف غير موجود على الاسطوانة الحالية وبالتالي على المبرمج الثاکد من أن عملية فتح الملف تمت بدون مشاكل ليتجنب ظهور رسالة خطأ من نظام التشغيل DOS وإنتهاء البرنامج بشكل غير مقبول

البرنامج الموجود بالشكل رقم ٣ - ١٠ يقوم بالتأكد أولاً من فتح الملف فإذا لم يتم فتحه يعطي البرنامج رسالة خطأ وينتهي عمل البرنامج ، وإذا تم فتح الملف يكمل البرنامج باقي الأوامر . فيفتح الملف ويعرض محتوياته

```
0: /* Program Name CS10_3.c */
1: #include <stdio.h>
2: #include <stdlib.h>
3: int main (void)
4: {
5:     FILE *fptr;
6:     char ch;
7:     clrscr();
8:     if((fptr=fopen("textfile.txt","r"))== NULL)
9:     {
10:         printf ("\n can't open the file ");
11:         exit(1);
12:     }
13:     while ((ch=getc(fptr)) !=EOF)
14:     {
15:         printf ("%c",ch);
16:     }
17:     fclose (fptr);
18:     getch();
19:     return 0;
20: }
```

الشكل رقم ٣ - ١٠ إختبار مشاكل فتح الملف

وعن هذا البرنامج نوضح ما يلى :

يبدأ البرنامج في السطر رقم ٨ باختبار الدالة () fopen فاذا فتحت الملف بدون مشاكل تجعل المؤشر يشير الى هذا الملف أما اذا لم تستطع فتح الملف لأى سبب تعيّد الدالة القيمة NULL (0) وبالتالي في السطر رقم ٨ نختبر اذا كان الناتج هو

NULL و معناه لم يتم فتح الملف وتظهر رسالة خطأ ثم ينتهي البرنامج بالدالة (1) exit
و اذا تم فتح الملف بنجاح تنفذ باقى اوامر البرنامج.

الكتابة والقراءة في الملف عبارة حرفية كل مرة

فيما يلى نشرح كيفية كتابة البيانات أو قراءة البيانات باستخدام العبارات الحرفية
و قبل أن نتابع البرنامج الذي يتحقق ذلك نعرض أولاً الدوال المستعملة في كتابة وقراءة
عبارة حرفية مع الملف

الدالة fputs()	وتستخدم لكتابة عبارة حرفية في ملف
الدالة fgets()	وتستخدم لقراءة عبارة حرفية من ملف
الدالة العامة fopen()	الدالة العامة لفتح الملف

والبرنامج الموجود بالشكل رقم ٤ - ١٠ يسمح للمستخدم بكتابة كلمات ويقوم
بتخزينها في ملف ويظل يستقبل الكلمات ويضعها في ملف حتى يضفط المستخدم مفتاح
الادخال بدون كتابة أي كلمة.

```
0: /* Program Name CS10_4.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main()
4: {
5:     FILE *fptr;
6:     char string[81];
7:     fptr=fopen("textfile2.txt","w");
8:     while (strlen(gets(string))>0)
9:     {
10:         fputs(string,fptr);
11:         fputs("\n",fptr);
12:     }
```

13: **fclose (fptr);**

14: }

الشكل رقم ٤ -١ كتابة عبارة حرفية في ملف

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٧ تم فتح الملف بالطريقة المعتادة لحالة الكتابة
- وفي السطر رقم ٨ الدالة () gets تستقبل من المستخدم عبارة حرفية
والدالة while تختبر هذه العبارة طالما أن طولها لا يساوى صفر أى لم يضغط
المستخدم مفتاح الادخال Enter بدون كتابة عبارة ينفذ سطر رقم ١٠
- في السطر رقم ١٠ يقوم البرنامج بكتابه العبارة في الملف الذي يشير اليه
المؤشر fptr ثم يعود البرنامج الى الدالة While في السطر ٨ وهكذا
الحالة W للكتابة فقط وبالتالي اذا كان الملف موجود من قبل يتم الكتابة فوق
بياناته وت فقد البيانات القديمة.



وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

write string to the file:

**the method of write to the file string by string it more quick than
char/char**

(string by string)

كما يمكن الكتابة في الملف عبارة حرفية كذلك يمكن القراءة من
الملف عبارة حرفية بعبارة حرفية (String by string) وهو أسرع من الكتابة والقراءة
حرف بحرف ولكن كل طريقة تستخدم حسب غرض البرنامج وتستخدم الدالة (Fgets)
للقراءة من الملف بطريقة عبارة حرفية في كل مرة والبرنامج الموجود بالشكل رقم
٤ -٥ يقوم بفتح ملف والقراءة منه عبارة بعبارة وطباعة محتويات الملف على الشاشة

```

0: /* Program Name CS10_5.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: int main(int argc,char *argv[])
4: {
5:     FILE *fptr;
6:     char string[81];
7:     if(argc !=2)
8:     {
9:         printf ("\n format c>type2 filename");
10:        exit(1);
11:    }
12:    if( (fptr=fopen(argv[1],"r") ) == NULL)
13:    {
14:        printf ("\n can't open this file");
15:        exit(1);
16:    }
17:    while (fgets(string,80,fptr) !=NULL)
18:        printf ("%s",string);
19:    fclose (fptr);
20:    return 0;
21: }
```

الشكل رقم ١٠-٥ قراءة ملف عبارة بعبارة

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ١٧ استخدمنا الدالة (fgets) للقراءة من الملف وكما تلاحظ أن لهذه الدالة ثلاثة معلمات . الاول هو المتغير string الذى نقرأ فيه البيانات والثانى القيمة ٨٠ ويمثل عدد الحروف المراد قرائتها كل مرة ، والثالث المؤشر الذى يشير الى الملف الذى نقرأ منه .
- السطر رقم ١٨ يطبع هذه البيانات ويستمر البرنامج حتى تنتهي محتويات الملف .

التعامل مع الطابعة والملفات الثابتة

من الموضوعات المهمة اخراج النتائج أو محتويات الملف على الطابعة ويتم ذلك باستخدام دوال الكتابة في الملف بحيث نكتب متغير ثابت معرف للطابعة وهو المتغير stdprn ومعناه standard printer مكان اسم الملف المراد الكتابة فيه. والبرنامج الموجود بالشكل رقم ٦-١٠ يوضح كيفية اخراج محتويات ملف على الطابعة

```
0: /* Program Name CS10_6.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: int main(int argc,char *argv[])
4: {
5:     FILE *fptr;
6:     char string[81];
7:     if(argc !=2)
8:     {
9:         printf ("\n form is c:\>type2 filename");
10:        exit(1);
11:    }
12:    if( (fptr=fopen("argv[1]","r") ) == NULL)
13:    {
14:        printf ("\n can't open this file");
15:        exit(1);
16:    }
17:    while (fgets(string,80,fptr) !=NULL)
18:    {
19:        fputs(string,stdprn);
20:        putc('r',stdprn);
21:    }
22:    fclose (fptr);
```

```
23:     return 0;
24: }
```

الشكل رقم ١٠-٦ أخراج محتويات الملف على الطابعة

وعن هذا البرنامج نوضح ما يلى:

في السطر رقم ١٩ استخدمنا المتغير الذي يشير إلى الطابعة وهو stdprn والدالة fputs(string, stdprn) معناها اكتب عبارة حرفية (String) على الطابعة المشار إليها بالمتغير stdprn

وهكذا يظل البرنامج يقرأ محتويات الملف ويرسلها إلى الطابعة حتى يصل البرنامج إلى نهاية الملف

كتابة وقراءة بيانات صحيحة وحقيقية وحرفية

شرحنا فيما سبق كيفية القراءة والكتابة في الملف باستخدام نوعين من البيانات وهما متغير من نوع حرف ومتغير من نوع عبارة حرفية وهنا نشرح كتابة وقراءة أنواع أخرى من أنواع البيانات مثل القيم الصحيحة والقيم الحقيقة ،والدوال المستخدمة لذلك هي :

الدالة fprintf() و تستخدم لكتابة بيانات مختلفة النوع في ملف
والدالة fscanf() و تستخدم لقراءة بيانات مختلفة النوع من الملف

و هذه الدوال تقوم بنفس عمل الدوال printf(),scanf() ولكن مع الملفات
والبرنامج الموجود بالشكل رقم ١٠-٧ يقوم باستقبال بيانات مختلفة النوع ثم باستخدام الدالة fprintf() يتم كتابة هذه البيانات في الملف.

```
0: /* Program Name CS10_7.C*/
1: #include <stdio.h>
2: #include <conio.h>
```

```
3: main ()  
4: {  
5:     FILE *fptr;  
6:     float height;  
7:     int code;  
8:     char name[40];  
9:     fptr=fopen("txtfile3.txt","a");  
10:    do  
11:    {  
12:        printf ("\n Type name: code number:,and height:");  
13:        scanf ("%s",name);  
14:        scanf("%d",&code);  
15:        scanf("%f",&height);  
16:        fprintf(fptr,"%s%d%f",name,code,height);  
17:        printf ("In again(y/n)==>");  
18:    }  
19:    while (getche()=='y');  
20:    fclose(fptr);  
21: }
```

الشكل رقم ١٠-٧ كتابة بيانات مختلفة النوع في ملف

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

Type name: code number:,and height:samy 1 20

again(y/n)==>y

Type name: code number:,and height:hamdy 2 21

again(y/n)==>y

Type name: code number:,and height:nabil 3 23

again(y/n)==>n

وعن هذا البرنامج نوضح ما يلى :

• يقوم البرنامج الموجود في شكل ١٠-٧ في السطور رقم ١٣ و ١٤ و ١٥

باستقبال بيانات هي الاسم والكود والوزن

- في السطر رقم ١٦ يتم كتابة البيانات في الملف الذي يتم فتحه وباستخدام الدالة (fprintf).
- السطر رقم ١٧ يطبع رسالة (Y/N) Again.
- في السطر رقم ١٩ الدوارة While تنتظر استقبال حرف وتحتبره فإذا كان الحرف ٢ تعيد التنفيذ الى السطر رقم ١٠ وتعيد استقبال بيانات ثم كتابتها في الملف وإذا لم يكن حرف ٢ ينتهي البرنامج.

القراءة من ملف باستخدام (fscanf)

تقوم الدالة fscanf بنفس عمل الدالة scanf الا انها تستقبل البيانات من ملف وتكتب بالصورة (ptr,data) حيث أن المتغير ptr هو الملف المفتوح والمطلوب القراءة منه ، والمتغير أو المتغيرات التي توضع فيها البيانات المقرأة.

مثال

الكتابة والقراءة في الملف سجل بسجل (record by record)

تعبر الكتابة والقراءة في الملف سجل بسجل أنساب طريقة لتطبيقات قواعد البيانات حيث تتشكل سجل (structure) بالمواصفات المطلوبة ويتم استعماله سواء للكتابة أو القراءة حسب الرغبة ، والدوال المستعملة لذلك وهي :

الدالة fopen() التي تستخدم لفتح ملف بصفة عامة

الدالة fwrite() و تستخدم لكتابة سجل في ملف

الدالة fread() و تستخدم لقراءة سجل من ملف

وتأخذ الدالة (fwrite) الصورة التالية :

```
fwrite(&st,sizeof(st),1,ptr);
```

و فيها المتغير `&st` هو عنوان السجل (structure) المراد كتابته
والماקרו `(st)` يعيد حجم السجل المطلوب كتابته بالبايت
والرقم `1` هو عدد السجلات المراد كتابتها كل مرة
والمؤشر `ptr` هو مؤشر الى الملف المطلوب الكتابة فيه
والبرنامج الموجود بالشكل رقم `٨-١٠` يقوم باستعمال الدالة `fwrite` لكتابة سجل
بسجل في ملف كما يظهر ذلك من نتيجة التنفيذ

```
0: /* Program Name CS10_8.C*/
1: #include <stdio.h>
2: int main (void)
3: {
4:     struct
5:     {
6:         char name[40];
7:         int agnumb;
8:         double height;
9:     } emb;
10:    char numstr[81];
11:    FILE *fptr;
12:
13:    if( (fptr=fopen("agent.rec","wb")) == NULL)
14:    {
15:        printf ("\n can't open this file ");
16:        exit(1);
17:    }
18:    do {
19:        clrscr();
20:        printf ("\n Enter name:");
21:        scanf ("%s",emb.name);
22:        printf ("\n Enter number:");
23:        scanf ("%d",&emb.agnumb);
```

```
24:     printf ("\n Enter height:");
25:     scanf ("%f",&emb.height);
26:     fwrite (&emb,sizeof(emb),1,fptr);
27:     printf ("\n again(y/n)==>");
28: } while (getch() =='y');
29: fclose(fptr);
30: return 0;
31: }
```

الشكل رقم ٨ - إدخال الكتابة سجل سجل

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter name:hamdy
Enter number:321
Enter height:21
again(y/n)==>n
```

وعن هذا البرنامج نوضح ما يلى :

فى السطور من رقم ٤ الى السطر رقم ٩ إنشاء سجل و اعلان متغير من نوع
هذا السجل هو emb وفى السطر رقم ١٣ يتم فتح ملف للكتابة فيه فى حالة
binary ومن السطر رقم ١٨ حتى السطر رقم ٢٥ يتم استقبال بيانات السجل من
المستخدم وفى السطر رقم ٢٦ يتم كتابة هذا السجل فى الملف باستخدام الدالة
() fwrite بالمعاملات التى أشرنا إليها. وهذا البرنامج يصلح ليكون برنامج بسيط

القراءة من ملف سجل بسجل

تستخدم الدالة () fread للقراءة من الملف سجل بسجل وتأخذ نفس معاملات
الدالة () fwrite كما يلى :

```
fread(&st,sizeof(st),1,ptr)
```

Filing الفصل العاشر : الملفات

ويوضح ذلك من البرنامج الموجود في الشكل رقم ١٠-٩ الذي يستخدم الدالة `fread()` في قراءة سجلات من ملف ثم عرضها على الشاشة

```
0: /* Program Name CS10_9.C*/
1: #include <stdio.h>
2: void main (void)
3: {
4:     FILE *fptr;
5:     struct
6:     {
7:         char name [40];
8:         int numb;
9:         double height;
10:    } emb;
11: }
12: if ( (fptr=fopen("agent.rec","rb") )==NULL)
13: {
14:     printf ("\n can't open this file ");
15:     exit(1);
16: }
17: while (fread(&emb,sizeof(emb),1,fptr) ==1)
18: {
19:     printf ("\n Name: %s",emb.name);
20:     printf ("\n Number: %d",emb.numb);
21:     printf ("\n height: %d",emb.height);
22: }
23: fclose (fptr);
24: return 0;
25: }
```

شكل رقم ١٠-٩ قراءة ملف سجل بسجل

ونوضح فيما يلي الجديد في البرنامج الموجود بالشكل رقم ١٠-٩

في السطر رقم ١٧ تقرأ الدالة (fread) سجل من الملف بالمعاملات التي أشرنا إليها ولكن الدوارة while تقول طالما أن نتيجة الدالة (fread) تعيد قيمة صحيحة أقرأ من الملف وهي نفس القاعدة السابقة حيث يظل البرنامج يقرأ من الملف طالما توجد فيه سجلات والسطور ١٩ ، ٢٠ ، ٢١ تطبع عناصر السجل على الشاشة.

الوصول المباشر لسجل معين في ملف random access

في الأمثلة السابقة كانت طرق القراءة سواء حرف بحرف أو عبارة حرفية أو بيانات مختلفة الأنواع أو سجل تم بطريقة متتالية أي قراءة سجل ثم الذي يليه وهكذا ، حيث لا يمكن الوصول إلى سجل إلا بعد قراءة السجلات التي قبله أما إذا أردت الوصول مباشرة إلى سجل معين فيجب أن تستخدم الدالة (fseek) التي تقوم بتوجيه مؤشر الملف إلى المكان المطلوب في الملف مباشرة وتأخذ الصورة العامة التالية

```
int fseek(FILE *stream, long offset, int whence);
```

وفيها مؤشر الملف الذي نتعامل معه ، والمتغير whence يحدد مكان بداية الحركة ويأخذ أحد الأرقام الثلاثة ، أو ١ أو ٢ والقيمة ٠ تعني ابدأ البحث من أول الملف والقيمة ١ تعني ابدأ من المكان الحالي لمؤشر الملف والقيمة ٢ تعني أن الحركة تبدأ من نهاية الملف ، والمتغير offset يتم توضيحه في شرح البرنامج التالي والموجود بالشكل رقم ١٠ - ١٠ حيث يسأل المستخدم عن رقم السجل المراد اظهار بياناته ويقوم البرنامج باظهار بيانات هذا السجل مباشرة.

```
0: /* Program Name CS12_10.C*/
1: #include <stdio.h>
2: int main (void)
3: {
4:     struct
5:     {
6:         char name[40];
7:         int num;
8:         double height;
```

Filling الفصل العاشر : الملفات

```
9:     }emb;
10:    FILE *fptr;
11:    int recno;
12:    long int offset;
13:    if ( (fptr=fopen("agent.rec","r") ) ==NULL)
14:    {
15:        printf ("can't open this file ");
16:        exit(1);
17:    }
18:    printf("\n ENTER RECORD NO:");
19:    scanf ("%d",&recno);
20:    offset=(recno-1)*sizeof(emb);
21:    if(fseek(fptr,offset,0) !=0)
22:    {
23:        printf ("\n can't move pointer there");
24:        exit(1);
25:    }
26:    fread(&emb,sizeof(emb),1,fptr);
27:    printf ("\n NAME:%s\n",emb.name);
28:    printf ("\n Number:%d\n",emb.num);
29:    printf ("\n Height:%f\n",emb.height);
30:    fclose (fptr);
31:    return 0;
32: }
```

شكل رقم ١٠-١٠ الوصول الى سجل مباشرة

وعن هذا البرنامج نوضح ما يلى :

السطر رقم ١٨ يسأل المستخدم عن رقم السجل وفي السطر رقم ١٩ تستقبل الدالة (`scanf()`) رقم صحيح وتخزنه في المتغير `recno` وفي السطر رقم ٢٠ يتم حساب عدد البايت المطلوب الذهاب إليها بالمعادلة التالية :

`offset=(recno-1)*sizeof(emb)`

و معناها حجم structure الذي يتم معرفته باستخدام الماكرو sizeof() مصريباً في رقم السجل مطروح منه القيمة 1 حيث يقف مؤشر الملف في بداية السجل المطلوب ثم باستخدام الدالة (fseek) نوجه مؤشر الملف إلى السجل المطلوب ونستخدم الدالة (fread) في القراءة فتقرأ السجل المطلوب ثم تتم طباعته على الشاشة

كتابة وقراءة مجموعات من البيانات (Buffer By Buffer)

كلمة Buffer عبارة عن مساحات متغيرة داخل الذاكرة تستخدم مؤقتاً كمحطات انتقالية لمعالجة المدخلات والمخرجات. كما سبق أن أشرنا في هذا الفصل أن قراءة الملفات يمكن أن تتم حرف أو عبارة عبارة أو سجلاً سجلاً، يمكن أيضاً أن تتم على شكل مجموعات وهذه هي الطريقة التي يستخدمها نظام التشغيل في القراءة والكتابة وتستخدم لهذا الغرض الدوال التالية :

الدالة () read تقوم بالقراءة من الملف بمقدار عدد من البايت(الحرف) يحدده

المبرمج والصورة العامة لهذه الدالة كما يلى :

read(int handle, void far *buf, unsigned len, unsigned *nread);

حيث :

عبارة عن رقم يشير إلى الملف المراد القراءة منه المتغير handle

متغير عبارة عن مصفوفة الحروف التي تمثل المخزن المؤقت المتغير buf

والذي تخزن به البيانات

يمثل عدد الحروف المطلوب قراءتها كل مرة المتغير nread

بينما تستخدم الدالة () write لكتابة مجموعة من البيانات في

الملف وتأخذ نفس معاملات الدالة () read.

وتستخدم الدالة () open لفتح الملف وتأخذ الصورة العامة التالية :

int open(const char *path, [, unsigned mode]);

حيث :

path يمثل اسم الملف والمسار المراد فتحه

mode هي حالة فتح الملف

و فيما يلى نوضح المعاملات التي تستخدم مع كل حالة من حالات فتح الملف

المعامل	استخدامه
O_APPEND	فتح الملف بفرض إضافة بيانات اليه للاضافة وبالعالي تضع مؤشر الملف في نهاية الملف
O_BINARY	فتح الملف بحالة binary mode
O_CREAT	فتح الملف للكتابة فقط وبالتالي اذا كان الملف موجود من قبل تكتب البيانات الجديدة على البيانات القديمة
O_RDONLY	تفتح الملف للقراءة فقط
O_RDWR	تفتح الملف للقراءة والكتابة
O_TEXT	text mode في
	يمكن الجمع بين حالتين منطقتين باستخدام المؤشر بالصورة التالية
O_RDONLY O_BINARY	كما في المثال التالي
	و معناها افتح الملف للقراءة فقط وبحالة binary mode

والبرنامج الموجود بالشكل رقم ١٠ - ١١ يوضح استعمال هذه الدالة في قراءة بيانات ملف

```

0: /* Program Name CS10_11.C*/
1: #include <fcntl.h>
2: #define BUFSIZE 512
3: char buff[BUFSIZE];

```

```

4: int main(int argc,char *argv[])
5: {
6:     int inhandle,bytes,j;
7:     if (argc != 2)
8:     {
9:         printf ("\n format c:>serch filename");exit(1);
10:    }
11:    if( (inhandle=open(argv[1],O_RDONLY | O_BINARY)) <0)
12:    {
13:        printf ("\n can't open this file ");
14:        exit(1);
15:    }
16:
17:    while ( (bytes=read(inhandle,buff,BUFFSIZE)) >0)
18:    for (j=0;j<bytes;j++)
19:        putch(buff[j]);
20:    close(inhandle);
21:    return 0;
22: }

```

شكل رقم ١٠-١١ قراءة دفعه من الحروف

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ١١ تستخدم الدالة () open لفتح الملف inhandle
- وفي السطر رقم ١٧ تقرأ الدالة read من الملف المشار اليه بالرقم inhandle ثم تضع الحروف المقرؤة في مصفوفة الحروف buff وتقرأ عدد من البايت مقداره BUFFSIZE . وتعيد العدد الحقيقي في المتغير bytes
- في السطر رقم ١٨ دوارة for تبدأ من صفر الى العدد bytes وهو عدد الحروف الذي أدخل.
- في السطر رقم ١٩ الدالة () putch() تطبع عناصر المصفوفة وهي محتويات الملف على الشاشة.

رسائل الخطأ Error Messages

في حالة التعامل مع الملفات بطريقة نظام التشغيل وهي buffer by buffer يمكن لنا تحديد هل تم فتح الملف أم لا وذلك باستخدام الدالة open() ويوضح ذلك من البرنامج الموجود في الشكل رقم ١٠-١٢

```
0: /* Program Name CS10_12.C*/
1: #include <fcntl.h>
2: #define BUFFSIZE 512
3: char buff[BUFFSIZE];
4: int main(int argc,char *argv[])
5: {
6:     int inhandle,bytes,j;
7:     if (argc != 2)
8:     {
9:         printf ("In format c:>serch filename");exit(1);
10:    }
11:    if ((inhandle=open(argv[1],O_RDONLY | O_BINARY)) <0)
12:    {
13:        perror ("In can't open this file ");
14:        exit(1);
15:    }
16:    while ((bytes=read(inhandle,buff,BUFFSIZE)) >0)
17:    for (j=0;j<bytes;j++)
18:        putch(buff[j]);
19:    close(inhandle);
20:    return 0;
21: }
```

شكل رقم ١٠-١٢ تحديد رسائل الخطأ

وفي البرنامج الموجود في الشكل رقم ١٠-١٢ تم وضع شرط مع الدالة open() في السطر رقم ١١ بحيث اذا أعادت الدالة قيمة أقل من صفر فهذا يدل على أن الدالة لم تسطيع فتح الملف وبالتالي يقوم البرنامج بطباعة رسالة خطأ وذلك في السطر رقم ١٣ وينتهي البرنامج

كتابة مجموعات من البيانات buferr by buffer

المقصود بذلك هو استقبال كمية كبيرة من البيانات ثم تسجيلها مرة واحدة في الملف وهذا الاسلوب مفيد مع برامج كتابة الرسائل حيث يسمح للمستخدم بالكتابة وفي النهاية يخزن البيانات في الملف والبرنامج الموجود بالشكل رقم ١٠-١٣ يوضح هذه الطريقة

```
0: /* Program Name CS10_13.C*/
1: #include <fcntl.h>
2: #include <stat.h>
3: #define BUFFSIZE 4096
4: char buff[BUFFSIZE];
5: int main(int argc,char *argv[])
6: {
7:     int inhandle,outhandle,bytes,j;
8:     if (argc != 3)
9:     {
10:         printf ("\n format c:\>copy2 file1 file2");exit(1);
11:     }
12:     if( (inhandle=open(argv[1],O_RDWR | O_BINARY)) <0)
13:     {
14:         printf ("\n can't open INPUT file ");
15:         exit(1);
16:     }
17:    if((outhandle=open(argv[2],O_CREAT|O_BINARY|S_WRITE))<0)
18:    {
19:        printf ("\n can't open OUTPUT file ");




```

```
20:         exit(1);
21:     }
22:     while ( (bytes=read(inhandle,buff,BUFFSIZE)) >0)
23:     {
24:         write (outhandle,buff,bytes);
25:     }
26:     close(inhandle);
27:     close(outhandle);
28:     return 0;
29: }
```

الشكل رقم ١٣ - ١٠ كتابة الحروف دفعه دفعه

وعن هذا البرنامج نوضح ما يلى :

- يقوم البرنامج بفتح ملفين ، في السطر رقم ١٠ يتم فتح الملف الأول للقراءة وفي السطر رقم ١٧ يتم فتح الملف الثاني للكتابة بحالة binary mode أيضاً binary mode
- في السطر رقم ٢٢ يتم القراءة من الملف الأول بالدالة read() وفي السطر رقم ٤ يتم الكتابة في الملف الثاني ويستمر البرنامج في القراءة من الملف الأول والكتابة في الملف الثاني حتى ينتهي الملف الاول ، وبالتالي يتم انشاء ملف جديد يحتوى على نفس محتويات الملف الاول وهى نفس فكراة أمر Copy الذي يستخدمه نظام التشغيل DOS.

إرسال المخرجات إلى الطابعة

ارسال المخرجات إلى الطابعة باستخدام الدالة () write ، فبدلاً من الكتابة في ملف نكتب الرقم الخاص بالطابعة وهو الرقم ٤ وهذا الرقم محجوز للطابعة الموصولة بالجهاز ويوضح ذلك من البرنامج الموجود بالشكل رقم ١٤ - ١٠

```
0: /* Program Name CS10_14.C*/
1: #include <fcntl.h>
2: #define BUFFSIZE 512
```

```
3: char buff[BUFFSIZE];
4: int main(int argc,char *argv[])
5: {
6:     int inhandle,bytes,j;
7:     if (argc != 2)
8:     {
9:         printf ("\n format c:\>serch filename");exit(1);
10:    }
11:    if( (inhandle=open(argv[1],O_RDONLY | O_BINARY)) <0)
12:    {
13:        printf ("\n can't open this file ");
14:        exit(1);
15:    }
16:    while ( (bytes=read(inhandle,buff,BUFFSIZE)) >0)
17:        write (4,buff,bytes);
18:    close(inhandle);
19:    return 0;
20: }
```

شكل رقم ١٤

والجديد في البرنامج أن الدالة write الموجودة بالسطر رقم ١٧ تكتب في الملف رقم ٤ وهذا الرقم محجوز لآلة الطباعة وبالتالي تخرج النتائج على الطابعة
متى تستخدم كل طريقة من الطرق السابقة

وفي نهاية هذا الفصل نسأل متى تستخدم كل طريقة من الطرق السابقة للتعامل مع الملفات وللرد على هذا التساؤل نقول :

* اذا أردت أن تكتب برنامج معالج كلمات word processor يفضل استعمال طريقة Beffer By Buffer . وهي الكتابة والقراءة بمقدار المخزن المؤقت

* اذا أردت عمل dbase system تستخدم طريقة القراءة والكتابة باستخدام السجلات
.structure By Strudure

* اذا أردت عمل برنامج يتعامل مع الحروف مثل عد حروف ملف مثلاً أو قياس سرعة
كاتب نستخدم طريقة القراءة والكتابة حرف بحرف

حالتي الكتابة والقراءة من ملف text mode,binary mode

يمكن تسجيل البيانات في الملفات في صورتين
الصورة الاولى تسمى text mode وهي تسجيل جميع البيانات في شكل
حروف حتى الارقام كل رقم يأخذ بait وهي طريقة النظام dos
، ويتم تمثيل علامة السطر الجديد (CR/LF) carriage

1A وتمثيل نهاية الملف (EOF) بالكود
الصورة الثانية تسمى binary mode وهي تسجيل جميع البيانات في شكلها
ال الطبيعي الارقام تسجل كأرقام والاحروف كحروف وهي طريقة
نظام التشغيل unix وفيها يتم تمثيل علامة السطر الجديد حرف
.linefeed LF واحد

من هذه المقارنة تجد أن الملف اذا كتب في حالة text وقراء في حالة
binary فأن عدد الحروف سيختلف، ويفضل استعمال حالة binary اذا كان البرنامج
يعامل مع الارقام.

ويتم تحديد حالة فتح ملف بطريقة binary mode أو text mode باستخدام
الدالة fopen() في حالة binary mode نضع حرف b بجانب الحرف الدال على
الفتح كما يلى :

`fopen("filename","rb");`

اما في حالة text mode فنكتب الدالة كما سبق في البرامج بدون حرف b

بالصورة التالية :

```
fopen("filename","r");
```

وهي الحالة الافتراضية

والبرنامج الموجود في الشكل رقم ١٠-١٥ يوضح الفرق بين الحالتين وذلك بفتح ملف في binary mode من الملفات الموجودة على القرص والمكتوبة في text mode وطبع عدد حروفه ومن نظام التشغيل DOS وبالامر dir نقارن عدد الحروف من البرنامج ونظام التشغيل لنرى الفرق.

```
0: /* Program Name CS10_15.C*/
1: #include <stdio.h>
2: #include <conio.h>
3: main(int argc ,char *argv[])
4: {
5:     FILE *fptr;
6:     char string[81];
7:     int count=0;
8:     if (argc !=2)
9:     {
10:         printf("\n format: c>charcnt filename");
11:         exit(1);
12:     }
13:     if((fptr=fopen(argv[1],"rb"))==NULL);
14:     {
15:         printf ("can't open this file");
16:         exit(1);
17:     }
18:     while (getc(fptr) !=EOF)
19:     {
20:         count++;
21:     }
22: }
```

الشكل رقم ١٠-١٥

وعن هذا البرنامج نوضح ما يلى :

البرنامج الموجود بالشكل رقم ١٥-١٠ هونفس برنامج عد الحروف السابق
والموارد بالشكل رقم ٤-١٠ والفرق بينهما أن هذا البرنامج يقوم بفتح الملف في
binary mode وذلك في السطر رقم ١٣ وبالتالي يختلف عدد الحروف.



الفصل الحادى عشر

دوال مهمة للمبرمج

فى هذا الفصل تتناول مجموعه من الدول التى تساعده فى تحسين
أداء البرنامج حيث تتناول الدول التالية:-

دالة تحديد احداثيات نافذة على الشاشة ◆

نسخ جزء من الشاشة إلى موضع آخر ◆

حفظ جزء من الشاشة في متغير ◆

استرجاع الجزء المحفوظ من متغير ◆

تغيير درجة اضاءة الحرف ◆

تخزين مكان المؤشر في متغير (*Wherex()*,*wherey()*) ◆

التعامل مع الالوان والصوت ◆

برنامج محرر نصوص ◆

دالة تحديد احداثيات نافذة على الشاشة

من الوظائف المهمة تحديد جزء معين على الشاشة بحيث يمكن اجراء عمليات معينة على هذا الجزء فقط مثل تغيير الوان الكتابة أو الخلفية لهذا الجزء. ويتم ذلك بتحديد احداثيات هذا الجزء باستخدام الدالة window تأخذ الدالة window الشكل التالي:

window(L,T,R,B)

حيث أن المتغيرات L تعنى LEFT وهو الطرف الايسر للنافذة، T تعنى TOP وهو الطرف العلوي للنافذة ، R : ANY أي الطرف اليمين ، B : ANY أي أسفل النافذة ANY الاحداثيات الأربع للجزء المحدد وهي احداثى نقطة الركن الشمالي العلوي واليمين السفلي.

والبرنامج الموجود بالشكل رقم ١١-١ يقوم بتحديد احداثيات نافذة ثم استعمال الدوارة for لنكرار طباعة كلمة Allah داخل هذه النافذة فتعطي النتيجة الموجودة في الشكل رقم ١١-٢

```
0: /*Program Name CS11_1.C*/
1: #include <conio.h>
2: #define LEFT 10
3: #define TOP 8
4: #define RIGHT 52
5: #define BOT 21
7: void main (void)
8: {
9:     int j;
10:    window(LEFT, TOP, RIGHT, BOT);
11:    textcolor(RED);
12:    textbackground(GREEN);
13:    for (j=0;j<200;j++)
14:    {
```

```
15:         cputs(" ALLAH ");
16:         delay(10);
17:     }
18:     gotoxy(15,8);
19:     cputs(" THE GOD ");
20:
21:     getch();
22: }
```

شكل رقم ١١-١ استخدام الدالة window لتحديد احداثيات نافذة

ومن تنفيذ هذا البرنامج نحصل على النتيجة التالية :

H ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
AH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
H ALLAH ALLA THE GOD LAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
LAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH
AH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH ALLAH

شکل رقم ۱۱-۲

وعن هذا البرنامج نوضح ما يلى :

- يُشتمل السطر رقم ١٠ على الدالة Window وبها المعاملات LEFT, TOP, RIGHT, BOT التي تم تعریفها في أول البرنامج ومعناها تحديد نافذة بهذه الاحداثيات.

وفي السطرين رقم ١١ و ١٢ تقوم الدالیین textcolor() و textbackground() بتغيير لون الكتابة ولون الخلفية.

في السطر رقم ١٣ الدوارة for للتكرار ٢٠٠ مرة.

- في السطر رقم ١٥ الدالة `cputs()` التي تطبع كلمة ALLAH وهي نفس عمل الدالة `puts()` ولكنها تكتب داخل النافذة التي تم تحديدها بالالوان والقيم المحددة مسبقاً.

نسخ جزء من الشاشة إلى موضع آخر

يمكن نسخ أي مساحة من الشاشة إلى مكان آخر على الشاشة وذلك باستخدام الدالة `movetext()`. وتظهر بالصورة الآتية :

`movetext (LEFT, TOP, RIGHT, BOT, DESLEFT, DESTOP);`
 حيث المتغيرات LEFT , TOP, RIGHT, BOT هي احداثيات المساحة المراد نسخها والمتغيران DESLEFT, DESTOP هما احداثيات النقطة المراد النسخ اليها ، والبرنامح الموجود بالشكل رقم ١١-٣ يقوم بالكتابة على الشاشة ثم يقوم باستعمال الدالة `(movetext())` لنسخ مساحة إلى مكان اخر من الشاشة كما يتضح من نتيجة التنفيذ الموجودة في الشكل رقم ١١-٤

```

0: /*Program Name CS11_3.C*/
1: #include <conio.h>
2: #define LEFT 26
3: #define TOP 7
4: #define RIGHT 65
5: #define BOT 20
6: #define DESLEFT 1
7: #define DESTOP 1
8: #define NUMCOLORS 16
9: #define HEIGHT (BOT-TOP+1)
10: void fancy_box(int x1,int y1,int x2,int y2)
11: {
12:     int i;
13:     gotoxy(x1,y1);putch(201);
14:     for(i=x1+1;i<x2;i++) putch(205);

```

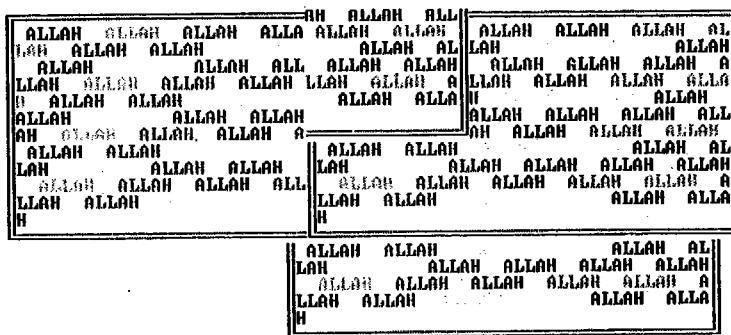
```
15:     putch(187);
16:     for(i=y1+1;i<y2;i++) {
17:         gotoxy(x1,i);putch(186);
18:         gotoxy(x2,i);putch(186);
19:     }
20:     gotoxy(x1,y2);putch(200);
21:     for (i=x1+1;i<x2;i++) putch(205);
22:     putch(188);
23: }

24: void main (void)
25: {
26:     int j;
27:     clrscr();
28:     fancy_box(LEFT, TOP, RIGHT, BOT);
29:     window(LEFT+1, TOP+1, RIGHT-1, BOT-1);
30:     /*textcolor(RED);*/
31:     textbackground(GREEN);
32:     for (j=0;j<98;j++)
33:     {
34:         textcolor(j % NUMCOLORS);
35:         cputs(" ALLAH ");
36:         /*delay(10); */
37:     }
38:     /*delay(200); */
39:     movetext(LEFT, TOP, RIGHT, BOT, DESLEFT, DESTOP);
40:     getch();
41:     movetext(LEFT, TOP, RIGHT, BOT, DESLEFT+27, DESTOP);
42:     getch();
43: }
44: }
```

شكل رقم ١١-٣

وعن هذا البرنامج نوضح ما يلى :

- من السطر رقم ١٠ حتى السطر رقم ٢٣ يتم الشاء دالة لرسم مستطيل مزدوج الخط
- وفي السطر رقم ٢٩ يتم استدعاء هذه الدالة ، وفي السطر رقم ٣٠ تحدد الدالة window() إحداثيات نافذة وباستخدام دالة fancy_box() نرسم مستطيل حول هذه النافذة
- وفي السطر رقم ٤٠ تقوم الدالة movetext() بنسخ المساحة المحددة بالاحداثيات LEFT, TOP, RIGHT, BOT والمعرفة في أول البرنامج الى النقطة التي احداثياتها DESLEFT, DESTOP والمعرفة أيضا في أول البرنامج ثم تكرر نفس العملية في احداثيات أخرى في السطر رقم ٤٢



شكل رقم ١١-٤

حفظ جزء من الشاشة في متغير

يمكن حفظ أي مساحة من الشاشة بمحوياتها في متغير لاسترجاعها عند الحاجة اليها ويتم ذلك باستخدام الدالة gettext() والتي تأخذ الصورة العامة التالية :

```
gettext(LEFT, TOP, RIGHT, BOT, buff);
```

حيث المتغيرات LEFT, TOP, RIGHT, BOT هي احداثيات المساحة المراد حفظها . والمتغير buff متغير من نوع مصفوفة نستعمله في تخزين المساحة المطلوب حفظها

استرجاع الجزء المحفوظ

بعد حفظ أي جزء من الشاشة في متغير باستخدام الدالة () gettext يمكن استرجاع هذا الجزء بالدالة puttext() والتي تأخذ الصورة التالية :

puttext(LEFT, TOP, RIGHT, BOT, buff);

كما تلاحظ أن الدالة puttext تأخذ نفس المعاملات التي تأخذها الدالة () gettext والبرنامج الموجود بالشكل رقم ١١-٥ يقوم باستخدام كلام من الدالة puttext() والدالة gettext() في حفظ جزء من الشاشة ثم مسح الشاشة ثم استرجاع هذا الجزء

```
0: /*Program Name CS11_5.C*/
1: #include <conio.h>
2: #define LEFT 1
3: #define TOP 1
4: #define RIGHT 80
5: #define BOT 25
6: #define DESLEFT 1
7: #define DESTOP 1
8: #define NUMCOLORS 16
9: #define HEIGHT (BOT-TOP+1)
10: int buff[80][25];
11: void fancy_box(int x1,int y1,int x2,int y2)
12: {
13:     int i;
14:     gotoxy(x1,y1);putch(201);
15:     for(i=x1+1;i<x2;i++) putch(205);
16:     putch(187);
17:     for(i=y1+1;i<y2;i++)
```

```
18:     { gotoxy(x1,i);putch(186);
19:         gotoxy(x2,i);putch(186);
20:     }
21:     gotoxy(x1,y2);putch(200);
22:     for (i=x1+1;i<x2;i++) putch(205);
23:     putch(188);
24: }
25: void main (void)
26: {
27:     int x,y,j;
28:     fancy_box(LEFT, TOP, RIGHT, BOT);
29:     window(LEFT+1, TOP+1, RIGHT-1, BOT-1);
30:     textcolor(RED);
31:     textbackground(GREEN);
32:     for (j=0;j<200;j++)
33:     {
34:         textcolor(j % NUMCOLORS);
35:         cputs(" ALLAH ");
36:     }
37:     gettext(LEFT, TOP, RIGHT, BOT,buff);
38:     x=wherex();
39:     y=wherey();
40:     clrscr();
41:     getch();
42:     clrscr();
43:     puts("restore screen ....");
44:     getch();
45:     puttext(LEFT, TOP, RIGHT, BOT,buff);
46:     getch();
47:     clrscr();
48:     puts("restore screen ....");
49:     getch();
50:     puttext(LEFT, TOP, RIGHT, BOT,buff);
51:     getch();
52:     clrscr();
53:     puts("restore screen ....");
54:     getch();
55:     puttext(LEFT, TOP, RIGHT, BOT,buff);
56:     gotoxy(x,y-1);
```

```
57: getch();  
58: }
```

شكل رقم ١١-٥ استخدام دوال حفظ واسترجاع الشاشة

وعن هذا البرنامج نوضح ما يلى :

- فى السطر رقم ٣٧ تقوم الدالة (gettext()) بحفظ المساحة المحددة بالاحاديث LEFT, TOP, RIGHT, BOT فى المتغير buff وجميع هذه المتغيرات تم تعريفها في أول البرنامج
- وفي السطر رقم ٤ يتم مسح الشاشة باستخدام الدالة clrscr وفي السطر رقم ٦ تجعل الدالة (getch()) البرنامج يتضرر الضغط على أي مفتاح
- في السطر رقم ٥ يتم استعادة المساحة المخزنة في المتغير buff مرة أخرى باستخدام الدالة (puttext()) وهكذا باقى البرنامج

تغيير درجة اضاءة الحرف

المقصود بتغيير درجة اضاءة الحرف جعلها مضيئة أو خافتة أو عادية ويستخدم لهذا الغرض الدوال التالية :

الدالة	وظيفتها
highvideo()	تجعل الحرف مضيء
lowvideo()	تجعل الحرف خافت
normvideo()	تعيد اضاءة الحرف الى الحالة المعتادة

كما توجد مجموعة من الدوال المفيدة التي توفر خدمات مختلفة معرفة كلها

في الملف conio.h

لذكر منها :

الدالة	وظيفتها
delline()	تحذف سطر من سطور النص
insline()	تضيف سطر بين سطور النص
textattr()	تغير طبيعة الحروف من حيث الالوان وطبيعة الظهور
textmode()	تغير حالة كتابة النصوص الى حالة اخرى

والبرنامج الموجود في الشكل رقم ١١-٦ مثال مصغر لبرنامج محرر سطور (Editor) حيث يسمح لك بكتابة سطور داخل نافذة محددة الاحداثيات كما يسمح بعض العمليات المشهورة مثل مسح سطر أو اضافة سطر أو الرجوع الى العملية السابقة وذلك باستخدام الدوال التي ذكرناها

```

0: /* Program Name CS11_6.C*/
1: #include <conio.h>
2: #define LEFT 10
3: #define TOP 8
4: #define RIGHT 50
5: #define BOT 21
6: #define WIDTH (RIGHT-LEFT+1)
7: #define TRUE 1
8: #define ESC 27
9: #define HEIGHT (BOT-TOP+1)
10: #define L_ARRO 75
11: #define R_ARRO 77
12: #define U_ARRO 72
13: #define D_ARRO 80
14: #define INS 82
15: #define DEL 83
16: #define ALT_H 35
17: #define ALT_C 46
18: #define ALT_U 22
19: #define ALT_L 38
20: int buff[WIDTH][HEIGHT];

```

```
21: void fancy_box(int x1,int y1,int x2,int y2)
22: {
23:     int i;
24:     gotoxy(x1,y1);putch(201);
25:     for(i=x1+1;i<x2;i++) putch(205);
26:     putch(187);
27:     for(i=y1+1;i<y2;i++) {
28:         gotoxy(x1,i);putch(186);
29:         gotoxy(x2,i);putch(186);
30:     }
31:     gotoxy(x1,y2);putch(200);
32:     for (i=x1+1;i<x2;i++) putch(205);
33:     putch(188);
34: }
35: void main (void)
36: {
37:     int x,y,j;
38:     char key;
39:     x=1;y=1;
40:     clrscr();
41:     fancy_box(LEFT, TOP, RIGHT, BOT);
42:     printf("\n\n  INS: Inseret line, DEL: delete line ESC:EXIT");
43:     printf ("\n  ALT_u undo,ALT_H high intensity,ALT_C color
44:             ENG.AZAB ");
45:     window(LEFT+1, TOP+1, RIGHT-1, BOT-1);
46:     textbackground(GREEN);
47:     while ( (key=getch()) !=ESC)
48:     {
49:         if (key==0)
50:         {
51:             switch(getch())
52:             {
53:                 case L_ARROW:
```

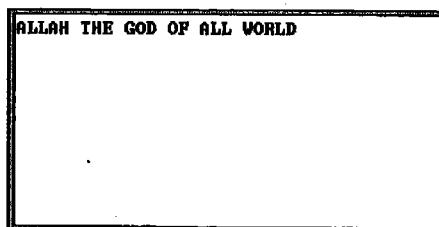
if (x>1)

```
54:           gotoxy(--x,y);
55:           break;
56:       case R_ARRO:
57:           if (x<WIDTH)
58:               gotoxy(++x,y);
59:           break;
60:       case U_ARRO:
61:           if (y>1)
62:               gotoxy(x,y--);
63:           break;
64:       case D_ARRO:
65:           if(y<HEIGHT)
66:               gotoxy(x,++y);
67:           break;
68:       case INS:
69:           gettext(LEFT+1,TOP+1,RIGHT-1,BOT-1,buff);
70:           insline();
71:           break;
72:       case DEL:
73:           gettext(LEFT+1,TOP+1,RIGHT-1,BOT-1,buff);
74:           deline();
75:           break;
76:       case ALT_H:
77:           highvideo();
78:           break;
79:       case ALT_L:
80:           lowvideo();
81:           break;
82:       case ALT_C:
83:           textcolor(getch()-'0');
84:           break;
85:       case ALT_U:
86:           puttext(LEFT+1,TOP+1,RIGHT-1,BOT-1,buff);
87:           break;
```

```
88:         case 'l':  
89:             printf ("\n");  
90:             break;  
91:         } /* end of switch */  
92:  
93:     } /* end if */  
94: else  
95: {  
96:     putch(key);  
97:     x=wherex();  
98:     y=wherey();  
99: }  
100: } /*end of while */  
101: } /* end of main () */
```

شكل رقم ١١-٦ محرر سطور بسيط

و عند تنفيذ : البرنامج نحصل على النتيجة التالية



INS: insert line, DEL: delete line ESC:EXIT
ALT_U undo, ALT_H high intensity, ALT_C color ENG.AZAB

شكل ١١-٧

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ٤ الدالة fancybox() ترسم مستطيل يحدد المساحة التي يكتب فيها المستخدم.

- السطر رقم ٤٣ و ٤٤ لطباعة التعليمات الخاصة بمحرر السطور
- وفي السطر رقم ٤ الدالة ()window تقوم بتحديد مساحة الكتابة
- ثم في السطر رقم ٤٦ تبدأ الدالة ()getche في استقبال حرف
في السطر رقم ٨ جملة if تختبر هل الحرف المدخل ممتد (مفتاح وظائف)
(راجع الفصل السابع) أم مفتاح عادي اذا كان مفتاح من مفاتيح الوظائف تستقبل
الدالة الرقم الثاني المميز له وفي السطر رقم ٥ جملة switch تختبر الكود
الثاني المميز للحرف الممتد وفيها الحالة الأولى والثانية لتوظيف مفاتحي الاسهم
الايمن واليسار والحالة الثالثة والرابعة لتوظيف مفاتحي الاسهم العلوي والسفلي
- السطر رقم ٦٨ توظيف المفتاح ins باستدعاء الدالة ()insline
- السطر رقم ٧٢ توظيف المفتاح del باستدعاء الدالة ()delline
- السطر رقم ٧٦ توظيف مفاتحي alt_h لجعل الكتابة مضيئة باستخدام الدالة
()highvideo ، وبالمثل السطر ٧٩ يجعل الكتابة خافتة باستخدام الدالة
()lowvideo
- في السطر رقم ٨٢ الدالة تستقبل ()getch رقم من المستخدم. هذا الرقم يدل
على اللون المطلوب ثم تضع رقم اللون كمعامل للدالة ()textcolor بعد طرح
كود ، حتى يطابق الرقم المطلوب.
- يقوم السطر رقم ٨٥ بالوظيفة undo ومعناها الرجوع عن آخر عملية تمت ،
ويتم ذلك عن طريق حفظ الشاشة بعد كل عملية بالدالة ()gettext
تتفذ جميع حالات التحرير (تحريك السهم أو الحذف أو النسخ) اذا كان
المفتاح الذي ضغط عليه المستخدم أحد مفاتيح الوظائف أما اذا كان المفتاح

مفتاحاً عادياً فيظهر الحرف على الشاشة ويعاد تفريز الدوارة ما لم يضفط
المستخدم مفتاح ESC.





الباب الثاني

الرسم في لغة C

الرسم من الموضوعات الجذابة في معظم اللغات ولمعظم هواء البرمجة، ولغة C لها نصيب كبير من دوال الرسم بل ان شئت قل أكبر نصيب بين اللغات ، فهى تختوى على جميع دوال أدوات الرسم، وفي هذا الفصل نناقش هذه الدوال ونتعرف على الموضوعات التالية :

- ♦ تهيئة الشاشة لحالة الرسم
- ♦ تحديد حالة الرسم المناسبة للقائمة
- ♦ تحديد الاخطاء الناتجة في حالة الرسم
- ♦ رسم الخطوط بألوانها مع تغير الألوان
- ♦ رسم قطع الناقص ومتعدد الخطوط
- ♦ التلوين (التظليل) والأشكال المختلفة للتظليل
- ♦ الرسم البياني ذو البعدين والأبعاد الثلاثية (3-DIM , 2-DIM)
- ♦ تحريك الرسوم
- ♦ استخدام خطوط الكتابة

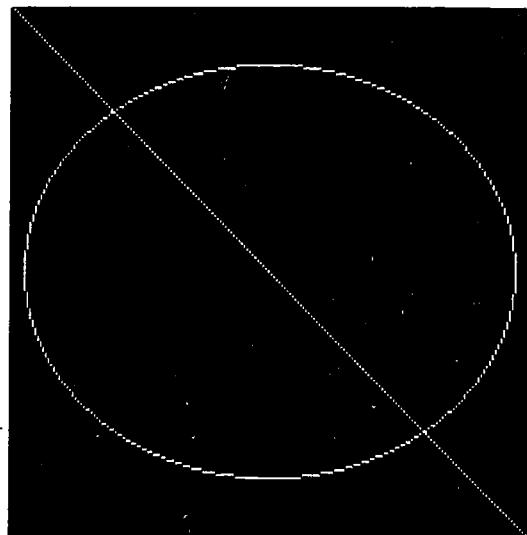
تستخدم لغة C مجموعة كبيرة من دوال الرسم ولكن قبل أن نبدأ في شرح واستعمال هذه الدوال نوضح كيفية تهيئة الجهاز لحالة الرسم حتى يقبل تنفيذ دوال الرسم

تهيئة الشاشة لحالة الرسم Initialization Graphics Mode

شرحنا في الفصول السابقة كثير من الدوال مثل دالة طباعة البيانات على الشاشة وكذلك دوال تغير لون الكتابة ولون الخلفية ، وجميع هذه الدوال تعمل في حالة النصوص (text mode) ولكن لاستعمال دوال الرسم لابد من تحويل الشاشة من حالة النصوص الى حالة الرسم (graphics mode) ولتوسيع ذلك نتابع البرنامج الموجود بالشكل رقم ١٢-١ الذي يقوم بتحويل الشاشة الى حالة الرسم ثم يرسم خط ودائرة ويوضح ذلك من الشكل ١٢-٢

```
0: /*Program Name CS12_1.C*/
1: #include <graphics.h>
2: void main (void)
3: {
4:     int driver,mode;
5:     int x1=0,y1=0;
6:     int x2=199,y2=199;
7:     int xc=100,yc=100;
8:     int radius=40;
9:     driver=CGA;
10:    mode=CGAC0;
11:    initgraph(&driver,&mode,"");
12:    line(x1,y1,x2,y2);
13:    circle(xc,yc,radius);
14:    getch();
15:    closegraph();
16: }
```

شكل ١٢-١ برنامج تحويل الشاشة إلى حالة الرسم ورسم خط ودائرة



شكل ١٢-٢ رسم خط ودائرة

و قبل شرح هذا البرنامج نوضح ما يلى :

من المعروف أن للشاشات كارت يركب داخل الجهاز وتوصى به الشاشة وهو ما يسمى بـ **بكارت الشاشة (Display Card)** وتوجد أنواع كثيرة من هذه الكروت من أشهرها :
كارت شاشة من نوع CGA (Color Graphics Adaptor)
كارت شاشة من نوع EGA (Enhanced Graphics Adaptor)
كارت شاشة من نوع VGA (Video Array Adaptor)

و في حالة الرسم (في لغة C) يحتاج كل نوع من هذه الكروت لملف يسمى **driver** ليهيء الشاشة للرسم وهذا الملف يأتي مع مترجم لغة C داخل الفهرس المسمى **bgi** بالاسماء **CGA.BGI**, **EGAVGA.BGI**, **HERC.BGI**, **EGAVGA**. خاص من هذه الملفات خاص بنوع شاشة (وكارت الشاشة) فمثلاً الملف **EGAVGA**

بالشاشات EGA والشاشات VGA . ويجب أن تكتب اسم الملف الموافق لкар特 الجهاز الذي نستعمله.

والأأن نعود لشرح البرنامج

- من السطر رقم ٤ إلى السطر رقم ٨ اعلان عن مجموعة متغيرات.
- في السطر رقم ٩ نضع في المتغير الصحيح driver رقم ، هذا الرقم معرف في المتغير CGA وهو يشير الى اسم الملف المطلوب لتهيئة الشاشة والمتوافق مع الشاشة المستعملة ، والمتغير CGAC0 يحدد حالة رسم وجميع الحالات معرفة في الملف graphics.h بمجموعة متغيرات.
- وفي السطر رقم ١١ الدالة initgraph() تقوم بتهيئة الشاشة لحالة الرسم ومعاملاتها كما يلى:
 - الاول عنوان المتغير driver الذي يشير الى نوع الكارت المركب.
 - الثاني عنوان المتغير mode الذي يحدد الحالة المطلوبة.
 - المعامل الثالث للدالة هو المسار الموجود فيه ملف التهيئة (CGA.BGI أو EGAVGA.BGI) حسب نوع الشاشة.
- والسطور من ٤ الى ١١ تعتبر جزء ثابت للتحويل الى حالة الرسم.
- وفي السطر رقم ١٢ نستخدم دالة line() لرسم خط له نقطتان ، إحداثيات النقطة الأولى هما $x1,y1$ وإحداثي النقطة الثانية هما $x2,y2$
- وفي السطر رقم ١٣ الدالة circle() لرسم دائرة حيث xc,yc مرکز الدائرة والمتغير radius نصف قطر الدائرة وجميع هذه المتغيرات معرفة في أول البرنامج.
- وفي السطر رقم ١٥ الدالة closegraph() تقوم بإغلاق حالة الرسم والرجوع الى حالة كتابة النصوص العادي ولو لم نستخدم هذه الدالة ستتجدد أن أي كلام

يكتب على الشاشة يكتب و كأنه رسم ويصبح المؤشر غير مرئي وذلك لأنك مازلت في حالة الرسم.

تحديد حالة الرسم تلقائياً Auto Initialization Detect

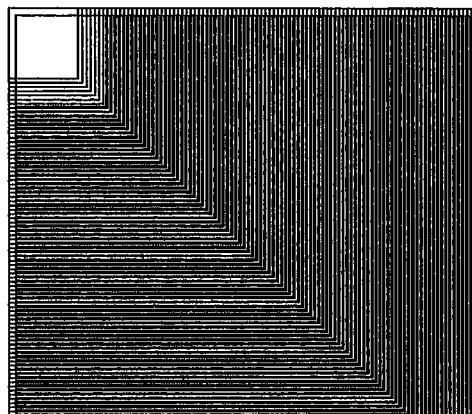
في البرنامج السابق رأينا أنه لاستعمال دوال الرسم لابد من تحديد نوع كارت الشاشة وبالتالي نوع الملف المستخدم لتهيئة الشاشة للرسم وكذلك تحديد الحالة (mode) التي تريدها من حالات الرسم وهذا الأمر مجهد ولله عبوب من أهمها أن البرنامج المصمم ليعمل مع كارت شاشة معين لا يعمل مع كارت شاشة آخر فمثلا البرامج المكتوبة لتعمل مع شاشة من نوع CGA لا تعمل مع شاشة من نوع EGA أو VGA والعكس صحيح. وهذا أمر متعب حيث كانت البرامج القديمة تسأل المستخدم أولا عن نوع الشاشة ثم تقوم بتهيئة الجهاز ليعمل مع الكارت المناسب. ولكن من الأفضل ترك مهمة تحديد نوع كارت الشاشة وتحديد حالة التشغيل للبرنامج. وهذا ما نراه في الشكل رقم ١٢-٣ الذي يتولى تحديد كارت الشاشة والحالة المناسبة لها ثم يقوم باستعمال دالة رسم مستطيل ودالة الصوت لرسم مجموعة مستويات كما تظهر النتيجة كما في الشكل رقم ١٢-٤

```
0: /*Program Name CS12-3C*/
1: #include<graphics.h>
2: void main(void)
3: {
4:     int i;
5:     int driver =DETECT;
6:     int mode;
7:     int maxx,maxy;
8:     int left,top,right,bot;
9:     initgraph(&driver,&mode,"");
10:    maxx=getmaxx();
11:    maxy=getmaxy();
12:    left=top=0;
```

```
14:     right=maxx;
15:     bot=maxy;
16:     for(i=0;i<400;i+=3)
17:     {
18:         rectangle(left,top,50+i,50+i);
19:         rectangle(left+5,top+5,70+i,70+i);
20:         sound(i*40);
21:         delay(10);
22:         nosound();
24:     }
25:     getch();
26:     closegraph();
27: }
```

شكل ١٢-٣ تحديد كارت الشاشة والحالة المناسبة لها

وعند تفريذ هذا البرنامج نحصل على النتيجة التالية.



شكل ١٢-٤ رسم مجموعة مستطيلات

وعن هذا البرنامج نوضح ما يلى:-

يتم تحديد نوع كارت الشاشة مباشرة وذلك في السطر رقم ٦ من البرنامج باستعمال الجملة

`driver=DETECT`

ومعنى كلمة DETECT أي أكتشف. وهي تحدد نوع الشاشة وتضع الرقم الدال على ذلك في المتغير `driver` وكذلك تحدد أفضل حالة رسم لهذه الشاشة وتضع الرقم `mode` الدال عليها في المتغير

في السطر رقم ١٠ يتم استخدام القيمتين في الدالة `initgraph()`

وبهذا الاسلوب فإن عملية التحويل الى حالة الرسم أصبحت سهلة جدا ولست مطالباً بمعرفة نوع الكارت المركب ونوع الشاشة.

وفي السطر رقم ١١ الدالة `getmaxx()` تحدد أقصى قيمة افقية ممكن الرسم فيها ، وهي القيمة العظمى للإحداثى الأفقي للشاشة وتخزنها في المتغير `maxx`

وفي السطر رقم ١٢ الدالة `getmaxy()` تحدد أقصى قيمة رأسية ممكن الرسم فيها وهي القيمة العظمى للإحداثى الرأسى للشاشة وتخزنها في المتغير `maxy` وفي السطر رقم ١٨ استخدمنا دالة `rectangle` لرسم مستطيل. وتأخذ دالة رسم المستطيل أربعة معاملات هي `Left, Top , Right , Bot` حيث أن المتغيرات `LEFT, TOP` هما إحداثى النقطة الأولى والمتغيران `RIGHT, BOT` هما إحداثى النقطة الثانية وهي متغيرات تم الإعلان عنها في أول البرنامج.

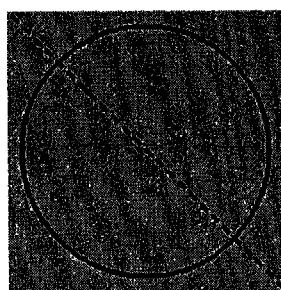
رسم الخطوط و تغيير الألوان Lines and Colors

يمكن تغيير نوع خط الرسم وسمكه وذلك قبل استعمال دوال الرسم باستخدام الدالة `setlinestyle()` التي تقوم بتحديد نوع وسمك الخط ويوضح ذلك من البرنامج الموجود بالشكل رقم ١٢-٥ الذي يقوم برسم خط دائرة بأشكال مختلفة ويظهر ذلك من نتيجة التنفيذ الموجودة بالشكل رقم ١٢-٦

```
0: /*Program Name CS12_5.C*/
1: #include <graphics.h>
2: #define IGNORED 0
3: void main (void)
4: {
5:     int driver,mode;
6:     int x1=0,y1=0;
7:     int x2=199,y2=199;
8:     int xc=100,yc=100;
9:     int radius=90;
10:    driver=DETECT;
11:    initgraph(&driver,&mode," ");
12:    setcolor(4);setbkcolor(7);cleardevice();
13:    setlinestyle(DASHED_LINE,IGNORED,THICK_WIDTH);
14:    setcolor(BLUE);
15:    line (x1,y1,x2,y2);
16:    circle(xc,yc,radius);
17:    getch();
18:    closegraph();
```

شكل رقم ١٢-٥ برنامج رسم خط دوائرة

و عند تنفيذ البر ناجٍ نحصل على النتيجة التالية.



شكل رقم ٦-١ رسم خط دائرة

الجديد في البرنامج الموجود بالشكل رقم ١٢-٥ أنه في السطر رقم ١٢ تقوم الدالة `setlinestyle()` بتحديد مواصفات الخط حسب معاملات الدالة كما يلى :

المعامل الأول : وهو المتغير `DASHED_LINE` يحدد شكل الخط هل هو متواصل أم متقطع وهكذا، و هناك مجموعة أشكال للخطوط يمكن تحديدها أحدها (تسمى `line style`) بالمتغير الدال على الشكل أو بالرقم المقابل له وهذه الاشكال هي

- 0 SOLID_LINE
- 1 DOTTED_LINE
- 2 CENTER_LINE
- 3 DASHED_LINE
- 4 USERBIT_LINE

والمعامل الثاني : هو المتغير `IGNORED` وقيمة صفر ومعناه تحديد نوع خط من الأنواع المعرفة باللغة.

والمعامل الثالث:: يمكن أن يأخذ احدى قيمتين هما :

- 1 NORM_WIDTH
- 3 THICK WIDTH

حيث ١ هو سماك الخط العادي أما ٣ فهو ثلث أضعاف الخط العادي وقد استخدمنا في هذا المثال `THICK-WIDTH`.

وفي السطر رقم ١٣ الدالة `(setcolor())` لتغيير اللون في حالة الرسم وتشتمل على الرقم الدال على اللون أو اسم اللون

وفي السطرين رقم ١٤ و ١٥ نستخدم دوال رسم الخط والدائرة

القطم الناقص والشكل المتعرج

نستخدم الدالة `(ellipse)` لرسم قطع ناقص وهو الشكل البيضاوى وتأخذ الصورة التالية:

`ellipse(xe,ye,stangle,endangle,xrad,yrad);`

وفيها المتغيرات

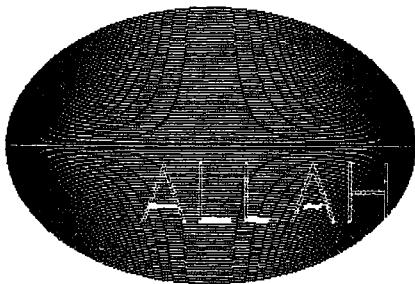
xe,ye هما مركز القطع الناقص والمتغيران stangle,endangle هما زاوية البداية
والنهاية

والمتغيران xrad,yrad هما نصف قطر (الافقى) فى اتجاه x ونصف قطر الرأسى
والبرنامح الموجود بالشكل رقم ١٢-٧ يستخدم الدالة ellipse() مع الدوارة
ليعطى نتيجة التنفيذ الموجودة بالشكل رقم ١٢-٨

```
0: /*Program Name CS12_7.C*/
1: #include <graphics.h>
2: void main (void)
3: {
4:     int driver,mode;
5:     int xe=250,ye=100;
6:     int xrad=150,yrad;
7:     int stangle=0,endangle=360;
8:     driver=DETECT;
9:     initgraph(&driver,&mode," ");
10:    for (yrad=0;yrad<100;yrad+=2)
11:        ellipse(xe,ye,stangle,endangle,xrad,yrad);
12:    moveto(200,90);
13:    setcolor(BLACK);
14:    settextstyle(3,0,7);
15:    outtext("ALLAH");
16:    getch();
17:    closegraph();
18: }
```

شكل رقم ١٢-٨

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :



شكل رقم ١٢-٩

وعن هذا البرنامج نوضح ما يلى:
في السطر رقم ١٠ يتم إستعمال الدوارة `for` لتغير قيمة المتغير `yrad` من صفر إلى
١٠٠.

في السطر رقم ١١ يتم استخدام الدالة () `ellipse` مع تغير قيمة المتغير `yrad` وبالتالي
يتم رسم أشكال مختلفة من القطع الناقص مشتركة في المركز و مختلفة في نصف القطر
كما في الشكل ١٢-٩.

رسم شكل غير منتظم (متعرج)

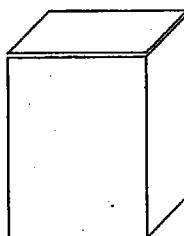
الشكل المتعرج هو أي شكل يتكون من مجموعة خطوط تعطى شكل غير منتظم وتقوم
الدالة () `drawpoly` برسم الشكل المتعرج وتأخذ الصورة العامة الآتية
`drawpoly(pno,rightpara);` والمعامل الأول للدالة هو المتغير `pno` ويمثل عدد
النقاط المطلوب التوصيل بينها. والمعامل الثاني عبارة عن مصفوفة من الأرقام تمثل
النقط المراد تمثيلها ويستخدم البرنامج الموجود بالشكل رقم ١٢-١٠ الدالة
`drawpoly()` لرسم شكل متعرج كما يتضح من نتيجه التنفيذ الموجودة بالشكل رقم

١٢-١١

```
0: /*Program Name CS12_10.C*/
1: #include <graphics.h>
2: #define LEFT 50
3: #define TOP 50
4: #define RIGHT 150
5: #define BOT 180
6: int rightpara[]={150,50,180,20,180,135,150,180};
7: int toppara[]={50,47,150,47,180,17,95,17,50,47};
8: void main (void)
9: {
10:     int driver,mode;
11:     driver=DETECT;
12:     initgraph(&driver,&mode,"");
13:     rectangle(LEFT,TOP,RIGHT,BOT);
14:     drawpoly(4,rightpara);
15:     drawpoly(5,toppara);
16:     getch();
17:     closegraph();
18: }
```

الشكل رقم ١٢-١٠ برنامج رسم شكل متعرج

وعند تنفيذ البرنامج نحصل على النتيجة التالية:



الشكل رقم ١٢-١١ رسم خط متعرج

و عن هذا البرنامج نوضح ما يلى:

في السطر رقم ٦ اعلان عن مصفوفة قيم صحيحة و تخزين هذه القيم بها ، وهذه القيم هي نقط متعدد الخطوط المطلوب التوصيل بينها (نقط الشكل المعرج اليمين كما في الشكل ١١-١٢) وبالمثل السطر رقم ٧ نقط متعدد الخطوط الأفقي والسطر رقم ١٣ يرسم مستطيل وهو الوجه الأمامي للشكل والسطر رقم ٤ و ١٥ يرسم متعدد الخطوط الجانبي والعلوي حتى تحصل الشكل رقم ١٢-١١

التلوين والتظليل والأشكال المختلفة للتظليل

يمكن تظليل أي مساحة مغلقة بأشكال مختلفة من أشكال التظليل وذلك بتحديد مواصفات شكل التظليل باستخدام الدالة `setfillstyle()` التي تأخذ الصورة التالية
`setfillstyle(PATTERNS,color)`
حيث المتغير `PATTERNS` هو شكل التظليل الذي يتم تحديده بالمتغير الدال عليه.
و فيما يلى قائمة المتغيرات المستخدمة في تحديد شكل التظليل.

<code>EMPTY_FILL</code>	<code>uses background color</code>
<code>SOLID_FILL</code>	<code>uses solid fill color</code>
<code>LINE_FILL</code>	<code>-- fill</code>
<code>LTSLASH_FILL</code>	<code>/// fill</code>
<code>SLASH_FILL</code>	<code>/// fill thick lines</code>
<code>BKSLASH_FILL</code>	<code>/// fill thick lines</code>
<code>LTBKSLASH_FILL</code>	<code>/// fill</code>
<code>HATCH_FILL</code>	<code>light hatch</code>
<code>XHATCH_FILL</code>	<code>heavy cross hatch</code>
<code>INTERLEAVE_FILL</code>	<code>interleaving line</code>
<code>WIDE_DOT_FILL</code>	<code>widely spaced dots</code>
<code>CLOSE_DOT_FILL</code>	<code>closely spaced dots</code>
<code>USER_FILL</code>	<code>user-defined fill</code>

و المعامل الثاني للدالة هو لون التظليل (وتوجد دالة خاصة برسم متعدد الخطوط `drawpoly()` و تظليله في نفس الوقت هي الدالة `fillpoly()` وهي تقوم نفس عمل الدالة

ولكن تظلل وتلوّن الشكل بالموصفات المحددة بالدالة (`setfillstyle()`) ، والبرنامج الموجود بالشكل رقم ١٢-١٢ يرسم شكل متعرج مظلل الاوجه كما يتضح من نتيجة التفريغ الموجودة بالشكل رقم ١٢-١٣

```
0: /*Program Name CS12_12.C*/
1: #include <graphics.h>
2: int rect[]={50,50 ,150,50, 150,180, 50,180, 50,50};
3: int sidepara[]={150,50, 180,20, 180,135,150,180};
4: int toppara[]={50,47, 150,47, 180,17, 95,17, 50,47};
5: void main (void)
6: {
7:     int driver,mode;
8:     driver=DETECT;
9:     initgraph(&driver,&mode," ");
10:    setfillstyle(11,GREEN);
11:    fillpoly(5,rect);
12:    setfillstyle(1,RED);
13:    fillpoly(4,sidepara);
14:    fillpoly(5,toppara);
15:    getch();
16:    closegraph();
17: }
```

شكل رقم ١٢-١٢ رسم شكل متعرج مظلل



شكل رقم ١٢-١٣ شكل متعرج مظلل

في البرنامج الموجود بالشكل رقم ١٢-١٢ في السطر رقم ١٠ استخدمنا الدالة `setfillstyle()` لتحديد شكل ولون التظليل و في السطر رقم ١١ استخدمنا الدالة `fillpoly()` لرسم شكل متعدد خطوط مظلل.

الرسم البياني ذو البعدين وثلاثي الأبعاد [2-DIM ,3-DIM]

من الدوال المفيدة في الرسم البياني الدالتين `bar()`,`bar3d()` حيث تقوم الدالة `bar()` برسم مستطيل وتأخذ الصورة `bar(l,t,r,b)` ومعاملاتها `l,t,r,b` هي كما في دالة رسم المستطيل والفرق بينها وبين دالة رسم المستطيل أن الدالة `bar()` ترسم مستطيل مظلل بشكل التظليل الذي تم تحديده مسبقا ، والدالة `bar3d()` ترسم مستطيل ثلاثي الأبعاد وهو يمثل الرسم في ٣ اتجهات وتأخذ الصورة `bar3d(l,t,r,b,z,f)`

وفيها المتغير `z` هو البعد الثالث والمتغير `f` يأخذ أحد قيمتين أما صفر أو ١ والقيمة صفر تعنى عدم رسم المستطيل العلوى للمكعب بينما القيمة ١ تعنى رسمه ، والبرنامج الموجود بالشكل رقم ١٢-١٤ يستخدم الدالة `bar3d()` في رسم مستطيل ثلاثي الأبعاد كما يظهر من الشكل رقم ١٢-١٥

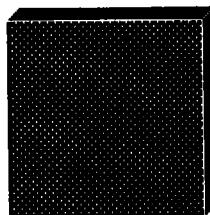
```
/*Program Name CS12_14.C*/
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{
    /* request auto detection */
    int gdriver = DETECT, gmode, errorcode;
    int midx, midy, i;
    /* initialize graphics, local variables */
    initgraph(&gdriver, &gmode, "");
```

```

/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk) /* an error occurred */
{
    printf("Graphics error: %s\n", grapherrmsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1); /* terminate with error code */
}
midx = getmaxx() / 2;
midy = getmaxy() / 2;
/* loop through the fill patterns */
for (i=EMPTY_FILL; i<USER_FILL; i++)
{
    /* set the fill style */
    setfillstyle(i, getmaxcolor());
    /* draw the 3-d bar */
    bar3d(midx-50, midy-50, midx+50, midy+50, 10, 1);
    getch();
}
/* clean up */
closegraph();
return 0;
}

```

شكل رقم ٤-١٢ برنامج رسم مستطيل ثلاثي الأبعاد



شكل رقم ٤-١٣ مستطيل ثلاثي الأبعاد

رسم الشكل الدائري Pie

يستخدم الشكل الدائري لتمثيل مجموعة قيم في شكل رسم بياني ويتم رسم الشكل الدائري باستخدام الدالة () pieslice التي تأخذ الصورة التالية

`pieslice(X,Y,startangle,endangle,RADIUS);`

حيث :

المتغيران X,Y هما احداثي نقطه مركز الشكل الدائري ، والمتغير startangle هو زاوية بداية رسم الشكل الدائري

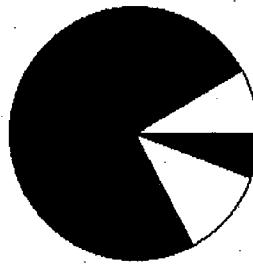
والمتغير endangle هو زاوية النهاية والمتغير RADIUS هو نصف القطر والبرنامج الموجود بالشكل رقم ١٦-١٢ يستخدم الدالة () pieslice() في رسم شكل دائري يمثل عدة قيم ويظهر ذلك في نتيجة التنفيذ الموجودة بالشكل رقم ١٧-١٢

```
0: /*Program Name CS12_16.C*/
1: #include <graphics.h>
2: #define N 6
3: #define RADIUS 90
4: #define X 100
5: #define Y 100
6: int data[N]={11,19,44,32,15,7,};
7: void main(void)
8: {
9:     float datasum,startangle,endangle,relangle;
10:    int driver,mode,j;
11:    driver=DETECT;
12:    Initgraph(&driver,&mode," ");
13:    for (j=0,datasum=0;j<N;j++)
14:        datasum +=data[j];
15:    endangle=0;
16:    for (j=0;j<N;j++)
17:    {
```

```

18:         startangle=endangle;
19:         relangle=360*(data[j]/datasum);
20:         endangle=startangle+relangle;
21:         setfillstyle(SOLID_FILL, j % 4 );
22:         pieslice(X,Y,startangle,endangle,RADIUS);
23:     }
24:     getch();
25:     closegraph();
26: }
```

شكل رقم ١٢-١٦ برنامج رسم شكل دائري



شكل رقم ١٢-١٧ شكل دائري

وعن هذا البرنامج نوضح ما يلى :

- في السطر رقم ١٣ يتم استعمال دورة for للتعامل مع القيم المخزنة بالمصفوفة [N].
- في السطر رقم ١٦ يتم استعمال دورة for لتعديل مقدار زاوية البداية وزاوية النهاية للشكل الدائري.
- في السطرين ١٩ ، ٢٠ يتم حساب زاويتي البداية والنهاية للشكل الدائري كل مرة.
- في السطر رقم ٢٢ يتم رسم الشكل الدائري بإستعمال الدالة () pieslice وذلك بالمعاملات التي تم تحديدها.

رسم خطوط بالنسبة لآخر نقطة (الرسم النسبي)

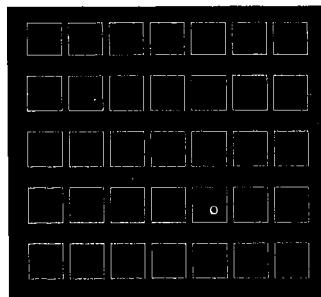
شرحنا في أول الفصل الدالة `(line)` التي ترسم خط وفيها لا بد من تحديد نقطى الخط ولكن في بعض الاحيان نحتاج لرسم خط بمعلومية نقطة واحدة والنقطة الثانية هي النقطة الحالية أو النقطة التي يمكن تحديدها بالدالة `moveto(x,y)` ، ويتم تحقيق ذلك باستخدام الدالة `linerel(x,y)` التي ترسم خط من المكان الحالى للمؤشر الى النقطة المحددة بـ `x,y` وهذا يفيد في الرسومات البيانية، والبرنامح الموجود بالشكل رقم ١٢-١٨ يستخدم الدالة `linerel(x,y)` لرسم مجموعه مستطيلات كما يظهر ذلك في الشكل رقم ١٢-١٩

```
/*Program Name cs12_18.c */
#include <graphics.h>
#define MAX 200
#define GRID 30
#define SIDE 25
void square(int side);
void main (void)
{
int driver,mode;
int x,y;
driver=DETECT;
Initgraph(&driver,&mode," ");
for (y=0;y<MAX;y+=GRID+10)
    for(x=0;x<MAX;x+=GRID)
    {
        moveto(x,y);
        square(SIDE);
    }
getch();
closegraph();
}
```

```
void square(int side)
```

```
{  
    linerel(side,0);  
    linerel(0,side);  
    linerel(-side,0);  
    linerel(0,-side);  
}
```

شكل رقم ١٢-١٨ برنامج رسم مجموعة مستطيلات



شكل رقم ١٢-١٩ مجموعة مستطيلات

وعن هذا البرنامج نوضح ما يلى :

في البرنامج الموجود بالشكل رقم ١٢-١٨ في السطر رقم ١٥ نستخدم الدالة moveto() لوضع المؤشر عند النقطة x,y (التي تتغير باستخدام دوارة for) وفي السطر رقم ١٦ نستدعي الدالة square() التي ترسم مربع باستخدام الدالة linerel() أو في السطر رقم ٢١ تبدأ الدالة () square ياستقبال رقم ثابت هو المتغير SIDE وقيمه ٣٠ و تستخدeme الدالة () linerel لرسم مربع طول ضلعه ٣٠ وبتغير موضع المؤشر بالدالة () moveto() تحصل على شكل رقم ١٢-١٩ وهو شبكة من المربعات.

رسم النقطة على الشاشة (Pixels)

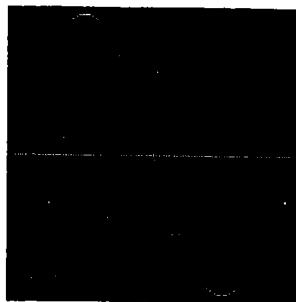
للنقطة (pixel) استخدامات كثيرة منها رسم أي معايرة بال نقط المتقاربة، وبضم رسم نقطة باستخدام الدالة putpixel(x,y,color); حيث المتغيران x,y هما احداثي النقطة والمتغير color هو لون النقطة المطلوب رسمها ، ومن التطبيقات التي تستخدم النقطة رسم المعادلات الرياضية

والبرنامج الموجود بالشكل رقم ١٢-٢٠ يستخدم الدالة putpixel() في رسم الدالة الجيبية (sine wave) كما يتضح من نتيجة التنفيذ الموجودة بالشكل رقم ١٢-٢١

```
/* Program Name cs12_20.c */
#include <dos.h>
#include <graphics.h>
#include <math.h>
void main (void)
{
    int driver,mode;
    double angle,sineofa;
    int x,y;
    driver=DETECT;
    initgraph(&driver,&mode," ");
    line (0,100,200,100);
    for (x=0;x<200;x++)
    {
        angle=( (double)x/200 *(2*3.14159265));
        sineofa=sin(angle);
        y=100-100*sineofa;
        putpixel(x,y,WHITE);
        sound(50*x);
        delay(3);
    }
    nosound();
    getch();
```

```
closegraph();  
}
```

شكل رقم ١٢-٢٠ برنامج رسم النقط



شكل رقم ١٢-٢١ رسم النقط

تحريك الرسومات *Animations*

من العمليات المشهورة في الرسم تحريك الرسومات ويتم ذلك بأكثر من طريقة من هذه الطرق تخزين الشكل المراد تحريكه في متغير مع مسح الشكل من الشاشة ثم رسم الشكل في موضع آخر ثم مسح الشكل وهكذا فيظهر وكأنه يتحرك ويتم ذلك بإستخدام الداللين (`getimage()` `putimage()`) والدالة `getimage()` تأخذ الصورة العامة التالية :

`getimage(x1,y1,x2,y2,clip)`

حيث المتغيرات $x2, y2, x1, y1$ هي احداثيات المساحة المطلوب تخزينها.

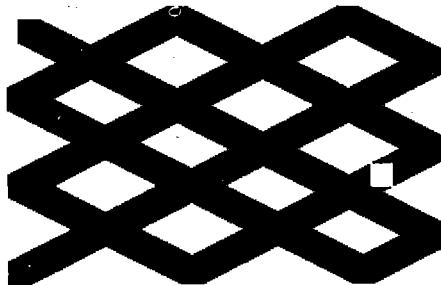
والمتغير Clip يأخذ القيمة صفر أو القيمة 1 ، و القيمة 1 معناها مسح الشكل بعد تخزينه ، والقيمة صفر معناها عدم مسح الشكل بعد تخزينه.

والبرنامج الموجود بالشكل رقم ١٢-٢٢ يقوم برسم دائرة وتحريكها على الشاشة كما يتضح من الشكل رقم ١٢-٢٣

```
/* Program Name cs12_22.c */
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include<graphics.h>
#define left 0
#define top 0
#define right 319
#define bottom 199
#define radius 8
void main (void)
{
    int driver,mode;
    int x,y,dx,dy,oldx,oldy;
    void *ballbuff;
    unsigned size;
    driver=DETECT;
    initgraph (&driver,&mode," ");
    rectangle(left,top,right,bottom);
    x=y=radius+10;
    setcolor(WHITE);
    setfillstyle(1,WHITE);
    circle(x,y,radius);
    floodfill(x,y,WHITE);
    size=imagesize(x-radius,y-radius,x+radius,y+radius);
    ballbuff=(void *)malloc(size);
    getimage(x-radius,y-radius,x+radius,y+radius,ballbuff);
    dx=2;
    dy=1;
    while ( !kbhit() )
    {
        putimage(x-radius,y-radius,ballbuff,COPY_PUT);
        oldx=x,oldy=y;
        x+=dx,y+=dy;
```

```
sound(10*x);
delay(10);
if (x<=left+radius+2 || x>=right-radius-2)
    dx=-dx;
if (y<=top+radius+1 || y>=bottom-radius-1)
    dy=-dy;
putimage(olidx-radius,oldy-radius,ballbuff,XOR_PUT);
}
nosound();
closegraph();
}
```

شكل رقم ١٢-٢٢ برنامج رسم دائرة وتحريكها



شكل رقم ١٢-٢٣ تحريك الدائرة

استخدام خطوط الكتابة

يمكن في حالة الرسم كتابة الكلمات بفونتات وأحجام مختلفة مما يعطي مخرجات جذابة - وذلك بتحديد مواصفات الكتابة مثل اسم الفونت والحجم واتجاه الكتابة ويتم ذلك باستخدام الدالة (settextstyle()) ثم تحديد مكان الكتابة على الشاشة باستخدام الدالة (moveto()) ثم طباعة الكلمات المراد طباعتها باستخدام الدالة .outtext()

ويمكن الكتابة في موضع معين مباشرة باستعمال الدالة (`outtextxy()`) التي تطبع الكلمات عند نقطة محددة ((X,Y)) والدالة (`settextstyle()`) تأخذ الصورة التالية:-
`settextstyle(FONT,DIRECTION,FONT_SIZE);`

حيث :

المتغير `FONT`: هو الأسم أو الرقم الدال على الفونت المطلوب الكتابة به ويتم اختياره من الجدول التالي وذلك بكتابة المتغير أو الرقم الدال عليه

0	<code>DEFAULT_FONT</code>
1	<code>TRIPLEX_FONT</code>
2	<code>SMALL_FONT</code>
3	<code>SANS_SERIF_FONT</code>
4	<code>GOTHIC_FONT</code>

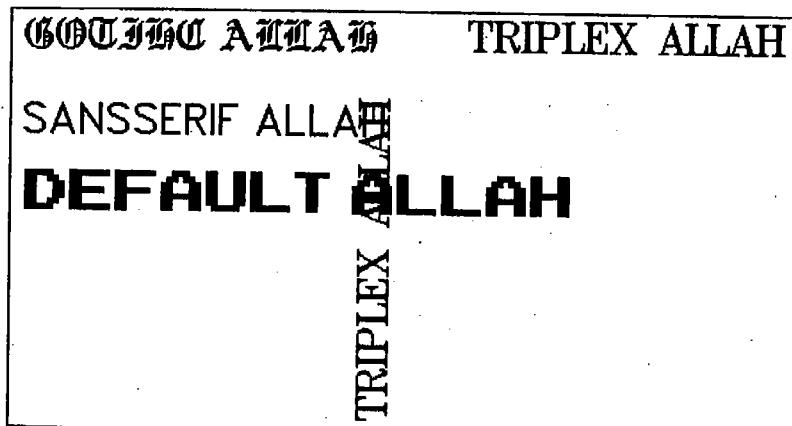
0	<code>HORIZ_DIR left to right</code>	افقى
1	<code>VERT_DIR bottom to top</code>	راسى

المتغير `FONT_SIZE` : يحدد حجم الفونت وبأخذ أرقام تعبر عن حجم الفونت . يقوم البرنامج الموجود بالشكل رقم ١٢-٢٤ بكتابة مجموعة كلمات بفونتات مختلفة وأحجام مختلفة ويوضح ذلك من نتيجة التنفيذ الموجودة بشكل رقم ١٢-٢٥

```
0: /* program name cs12_24.c */
1: #include <graphics.h>
2: #define FONT_SIZE 4
3: void main (void)
4: {
5:     int driver,mode;
6:     driver=DETECT;
7:     initgraph(&driver,&mode," ");
8:     settextstyle(4,0,FONT_SIZE);
```

```
9:     outtext("GOTIHC ALLAH ");
10:    settextstyle(1,0,FONT_SIZE);
11:    outtext("TRIPLEX ALLAH ");
12:    moveto(0,60);
13:    settextstyle(3,0,FONT_SIZE);
14:    outtext("SANSERIF ALLAH ");
15:    moveto(0,120);
16:    settextstyle(0,0,FONT_SIZE);
17:    outtext("DEFAULT ALLAH ");
18:    moveto(250,0);
19:    settextstyle(1,1,FONT_SIZE);
20:    outtext("TRIPLEX ALLAH ");
21:    getch();
22:    closegraph();
23: }
```

شكل رقم ٤-٢٤ برنامج استخدام خطوط مختلفة للكتابة



شكل رقم ٤-٢٥ الكتابة بخطوط مختلفة

وعن هذا البرنامج نوضح ما يلى :

في السطر رقم ٨ تم تحديد مواصفات الكتابة بالدالة (setTextStyle) وفيها المعامل الأول هو الرقم ٤ الذي يحدد نوع الفونت المطلوب الكتابة به ويؤخذ من الجدول السابق. وفي هذا المثال اخترنا الفونت GOTHIC_FONT ، والمعامل الثاني يحدد اتجاه الكتابة وفي هذا البرنامج اخترنا الاتجاه الأفقي والمعامل الثالث هو حجم الفونت المطلوب الكتابة به ، وفي السطر رقم ٩ تم طباعة الكلمات على الشاشة باستخدام الدالة (outtext) وهكذا باقي البرنامج .

مثال:

لا يعتمد الرسم الجيد في البرمجة على استخدام دوال الرسم المعروفة فقط بل يعتمد أكثر على الافكار الجيدة لاستغلال هذه الدوال ولتوسيع هذه الفكرة نتابع البرنامج التالي فلا نجد به دوال جديدة ولكن الجديد فيه هو فكرة اظهار خطوط الكتابة على شكل رسم حيث يقوم باستعمال دوال الكتابة بfonتات مع دوارة for لطباعة كلمة على الشاشة هذه الكلمة تطبع ٢٠ طبقة فوق بعضها بالوان مختلفة مما يعطي شكل جذاب

كما في الشكل ١٢-٢٦

```
0: /* program name cs12_26.c */
1: #include <graphics.h>
2: #define FONT_SIZE 6
3: void main (void)
4: {
5:     int driver,mode,i;
6:     driver=DETECT;
7:     initgraph(&driver,&mode,"");
8:     setTextStyle(1,0,FONT_SIZE);
9:     for (i=0;i<20;i++)
10:    {
11:        setcolor (i);
12:        outtextxy(50+i,(getmaxx()/2)+i, "ENG AZAB .. ");
13:        delay(200);
```

```
14:     }
15: getch();
16: closegraph();
17: }
```

الشكل رقم ١٢-٢٦ برنامج طباعة الابناء بألوان مختلفة



شكل رقم ١٢-٢٧ الكتابة بطبقات متعددة





الفصل الثالث عشر

دوال التساعل مع الذاكرة

Memory Allocation

في هذا الفصل نتناول موضوع التعامل مع الذاكرة مثل حجز أماكن في الذاكرة حسب الطلب وإلغاء حجز أماكن المتغيرات بعد الانتهاء من استعمالها وذلك من خلال مجموعة

دوال هي :

- ♦ **malloc ()** الدالة
- ♦ **free ()** الدالة
- ♦ **realloc ()** الدالة
- ♦ **calloc ()** الدالة
- ♦ **faralloc ()** الدالة

شرحنا في فصل المصفوفات كيفية الاعلان عن مصفوفة. فمثلاً مصفوفة عدد عناصرها ١٠٠ عنصر تتطلب من البرنامج حجز ١٠٠ مكان في الذاكرة وعادة لا نستعمل جميع عناصر المصفوفة وبالتالي تبقى بعض عناصر المصفوفة غير مستعملة وهذا استعمال سيء للذاكرة.

وأيضاً بعد حجز متغيرات والانتهاء من استعمالها تظل الاماكن محجوزة في الذاكرة وهذا يشغل حيز من الذاكرة لذلك يفضل مسحها ، وقبل أن نشرح دوال التعامل مع الذاكرة نراجع كيفية التعامل مع المصفوفة بالطريقة المعتادة

البرنامج الموجود في الشكل رقم ١٣-١ يقوم بالاعلان عن مصفوفة حرفيات عدد عناصرها ١٠٠ عنصر في حين أنك قد لا تستخدم كل هذه العناصر ولأننا لا نعرف العدد المراد حجزه تقوم بمحجز عدد كبير ولكن ١٠٠ عنصر يستخدم منهم البرنامج ما يشاء والباقي يعتبر مساحة غير مستغلة من الذاكرة

```

0: /*Program Name CS13_1.C*/
1: #include <stdio.h>
2: #define MAX 100
3: void main(void)
4: {
5:     char stud_name[MAX][35];
6:     long x;
7:     printf("\nEnter student names, a blank line will end\n");
8:     for (x = 0; x < MAX; x++)
9:     {
10:         printf("Enter student %5d: ", x+1);
11:         gets(stud_name[x]);
12:         if( stud_name[x][0] == '\0' )
13:             x = MAX;
14:     }
15:     printf("\n\nYou entered :\n");
16:     for (x = 0; stud_name[x][0] != '\0' && x < MAX; x++)

```

الفصل الثالث عشر : دوال التعامل مع الذاكرة

```
17: {  
18:     printf("InStudent %5d:", x+1);  
19:     printf("%s", stud_name[x]);  
20: }  
21: }
```

الشكل رقم ١٣-١ برنامج استخدام مصفوفة محددة العناصر

وعند تنفيذ هذا البرنامج نحصل على النتيجة التالية :

```
Enter student names, a blank line will end  
Enter student 00001: samy mohamed azab  
Enter student 00002: hamdy mohamed azab  
Enter student 00003: nabil mohamed azab  
Enter student 00004: azab mohamed azab  
Enter student 00005:
```

```
You entered :  
Student 00001: samy mohamed azab  
Student 00002: hamdy mohamed azab  
Student 00003: nabil mohamed azab  
Student 00004: azab mohamed azab
```

الشكل رقم ١٣-٢ نتائج التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- يقوم السطر رقم ٥ بالاعلان عن مصفوفة من نوع حرفى عدد عناصرها ١٠٠ لاتزيد عدد حروف كل منها عن ٣٥ حرفاً
- وفي السطر رقم ٨ الدوارة for للتكرار ١٠٠ مرة.
- وفي السطر رقم ١١ الدالة gets() تستقبل متغير حرفى يمثل اسم الطالب وتخزنه فى عنصر المصفوفة المقابل.

- وفي السطر رقم ١٢ شرط أذا كانت القيمة المدخلة صفر أى أن المستخدم ضغط على المفتاح Enter بدون ادخال قيمة يأخذ المتغير x القيمة max وبالتالي ينتهي التكرار ويتوقف ادخال البيانات
- وفي السطر رقم ١٦ تبدأ دوارة for لطباعة البيانات التي أدخلها المستخدم

ملاحظات على البرنامج

في السطر رقم ٥ تم الاعلان عن مصفوفة عدد عناصرها ١٠٠ عنصر في حين أن العناصر المستخدمة من المؤكد أنها لا تصل إلى ١٠٠ عنصر. أذا باقي العناصر الغير مستخدمة تعتبر محجوزة في الذاكرة بدون استعمال وهذا استعمال سوء للذاكرة لذلك تقوم دوال التعامل مع الذاكرة بتحديد عدد الأماكن المطلوبة فقط وهذا ما نراه في البرنامج الموجود بالشكل رقم ١٣-٣ الذي يسأل المستخدم عن عدد الطلاب المراد ادخال بياناتهم ثم يقوم بحجز أماكن بهذا العدد فقط وقبل أن نتابع البرنامج نشرح الدالة المستخدمة في حجز أماكن حسب الطلب.

المدالة malloc()

تقوم الدالة بحجز عدد من الأماكن في الذاكرة وتأخذ الصورة التالية :

```
void ptr = *malloc(size)
```

و معناها احجز عدد من الأماكن مقداره size وضع عنوان هذه الأماكن في

المؤشر ptr

والدالة تعيد مؤشر الى هذه الأماكن التي تم حجزها. فإذا لم تتمكن الدالة من حجز أماكن بسبب ما (أن الذاكرة غير كافية مثلاً) تعيد القيمة NULL (0)، ولذلك من حجز الأماكن نختبر قيمة المؤشر. ويوضح ذلك من البرنامج الموجود بالشكل رقم

١٣-٣

```
0: /*Program Name CS13_3.C*/
1: #include <stdio.h>
```

```
2: #include <stdlib.h>
3: void main(void)
4: {
5:     long nbr_students = 0;
6:     long ctr;
7:     char *stud_name;
8:     char trash[80]; /* to clear keyboard buffer */
9:     while( nbr_students < 1 || nbr_students > 2000000000 )
10:    {
11:        printf("\nHow many students will be entered? ==> ");
12:        scanf("%d", &nbr_students);
13:        gets(trash); /* clear out keyboard buffer */
14:        stud_name = (char *) malloc( 35*nbr_students );
15:        if( stud_name == NULL ) /* verify malloc() was successful */
16:        {
17:            printf( "\nError in line %3.3d: Could not allocate
               memory.", __LINE__ ); exit(1);
18:        }
19:        for( ctr = 0; ctr < nbr_students; ctr++ )
20:        {
21:            printf("\nEnter student %5d: ", ctr+1);
22:            gets(stud_name+(ctr*35));
23:        }
24:        printf("\n\nYou entered the following:\n");
25:        for ( ctr = 0; ctr < nbr_students; ctr++ )
26:        {
27:            printf("\nStudent %5d: ", ctr+1);
28:            printf(" %s", stud_name+(ctr*35));
29:        }
30:    }
```

شكل ١٣-٣ استخدام الدالة malloc لتحديد عدد العناصر المطلوبة

وعند تنفيذ البرنامج نحصل على النتيجة التالية :

How many students will be entered? ==> 5

Enter student 00001: samy

Enter student 00002: hamdy

Enter student 00003: nabil

Enter student 00004: mohamed

Enter student 00005: azab

You entered the following:

Student 00001: samy

Student 00002: hamdy

Student 00003: nabil

Student 00004: mohamed

Student 00005: azab

شكل ١٠-٤ نتيجة التنفيذ

وعن هذا البرنامج نوضح ما يلى :

- يقوم السطر رقم ١٥ بطباعة رسالة الى المستخدم تسأله عن عدد الطلاب.
- السطر رقم ١١ يستقبل البرنامج رقم صحيح وهو عدد الطلاب الذى يريد المستخدم أن يدخلها.
- وفي السطر رقم ١٤ الدالة `(malloc)` تحيز عدد من الاماكن مقدارها `nbr_students` العدد الذى أدخله المستخدم مضروبا في ٣٥ وهو عدد حروف كل عنصر لأن الدالة `(malloc)` تحيز بالبait وبالتالي تحيز ١٤ عنصر طول كل عنصر ٣٥ حرف وتجعل المؤشر `stud_name` يشير الى هذه الاماكن

والجزء `(char *)` يسمى Type Casting (راجع الفصل التاسع)

- وفي السطر رقم ١٥ جملة `if` تختبر قيمة المؤشر الى الحرفى `stud_name` فإذا كان يساوى `null` فمعنى هذا ان الدالة `(malloc)` لم تتمكن من حجز

- اماكن لأن الذاكرة غير كافية وبالتالي يطبع السطر رقم ١٧ رسالة تفيد بذلك وفيه الدالة `(exit)` التي تنهي البرنامج
- في السطر رقم ١٩ نستعمل دوارة `for` بعد المرات التي حددتها المستخدم لاستقبال أسماء الطلبة
- وفي السطر رقم ٢٢ الدالة `gets()` تستقبل هذه العناصر التي تمثل أسماء طلبة ونلاحظ هنا أن الزيادة تتم على العنوان فمثلاً المتغير `stud_name+(ctr*35)` من أول قيمة المتغير `ctr` تساوى صفر وبالتالي المعادلة تعطى `stud_name` وهو عنوان أول عنصر، وثانية قيمة للمتغير `ctr` هي ١ وبالتالي يزداد العنوان بمقدار ٣٥ بايت وهكذا...
- وفي السطر رقم ٢٥ تبدأ دوارة `for` أخرى لطباعة البيانات التي أدخلت والملاحظ هنا أن الدالة `(malloc)` ساعدتنا على حجز الاماكن المطلوبة فقط وبالتالي ليس هناك هناك أماكن محجوزة وغير مستعملة.

الدالة `realloc()`

تقوم الدالة بتحديث عدد الاماكن التي تم حجزها وبالتالي لانحتاج لتحديث عده الاماكن من البداية وتأخذ الصورة التالية :

`ptr=realloc(size)`

حيث أن المتغير `size` هو الحجم الجديد الذي نريد حجزه

نفرض أنك تريد إدخال مجموعة من القيم ولكن لا تعرف عدد هذه القيم ولا تريد تحديده ، بل تريدين أن تحجز مساحة عنصر واحد وكلما أدخلت عنصر حجز البرنامج له مكان في الذاكرة . لتحقيق ذلك نستعمل الدالة `realloc()` بحيث نحجز مساحة عنصر واحد ثم نزيد المساحة كلما أراد المستخدم أن يدخل بيانات جديدة

وهذا نراه من في البرامج الموجود في الشكل رقم ٥ - ١٣ وهذا البرنامج لا يسأل المستخدم على عدد الطالب كما في البرنامج الموجود بالشكل رقم ١٣ - ٣ ولكن يسأل مباشرة عن اسم الطالب ويظل يستقبل بيانات الطلبة حتى يضغط المستخدم مفتاح الادخال بدون بيانات

```
0: /*Program Name CS13_5.C*/
1: #include <stdio.h>
2: #include <stdlib.h>
3: #define NAME 35
4: void main(void)
5: {
6:     long student_cont = 0;
7:     long ctr;
8:     char *stud_name=NULL;
9:     while((stud_name =
realloc(stud_name, (NAME * (student_cont+1))))!=NULL )
10:    {
11:        printf("\nEnter student %5.#d:", student_count+1);
12:        gets(stud_name+( student_cont * NAME));
13:        if( stud_name[student_cont * NAME] == NULL )
14:        {
15:            break;
16:        }
17:        else
18:        {
19:            student_cont++;
20:        }
21:    }
22:    printf("\n\nYou entered the following:\n");
23:    for ( ctr = 0; ctr < student_cont; ctr++)
24:    {
25:        printf("\nStudent %5. d:", ctr+1);
```

```
26:     printf("%s", stud_name+(ctr*NAME));
27: }
28: free(stud_name);
29: }
```

شكل ١٣-٥ استخدام الدالة realloc()

عند تنفيذ : البرنامج نحصل على النتيجة التالية :

```
Enter student 00001: samy mohamed azab
Enter student 00002: hamdy mohamed azab
Enter student 00003: nabil mohamed azab
Enter student 00004: azab mohamed azab
Enter student 00005:
You entered the following:
Student 00001: samy mohamed azab
Student 00002: hamdy mohamed azab
Student 00003: nabil mohamed azab
Student 00004: azab mohamed azab
```

شكل ١٣-٦ نتيجة تنفيذ برنامج cs3_5.c

وعن هذا البرنامج توضح ما يلى :

- في السطر رقم ٩ يقوم البرنامج بمحجز مكان لعنصر واحد حيث قامت الدالة أول مرة بمحجز مساحة في الذاكرة مقدارها NAME أي ٣٥ بايت
- وفي السطر رقم ١٢ تستقبل الدالة gets() اسم طالب وتخزنه في المكان الذي يشير إليه المؤشر stud_name +(student_count*NAME) وفيه أول قيمة للمتغير student_count تساوى صفر وبالتالي تخزين أول اسم في أول عنوان ، ثم يزداد هذا العنوان كلما أدخل المستخدم اسم آخر وذلك نتيجة

زيادة قيمة المتغير student_cont وبالتالي تزداد المساحة المحجوزة ، من هذا المثال نجد أن الدالة realloc() ساعدتنا على حجز أماكن حسب الطلب فقط وبدون تحديد عدد الاماكن مسبقا مما يحقق أفضل استخدام للذاكرة.

الدالة (free)

تقوم هذه الدالة باخلاء الذاكرة من القيم التي تم استعمالها والتي لم يعد هناك حاجة الي وجودها في الذاكرة وتأخذ الدالة الصورة التالية

free (ptr)

و معناها أجعل المؤشر ptr حر بحيث لا يشير إلى أماكن بالذاكرة وبالتالي تصبح هذه الاماكن جاهزة للاستعمال من قبل أي برنامج آخر راجع البرنامج الموجود بالشكل رقم ١٣-٥ والبرنامج الموجود بالشكل رقم ١٣-٧ للتعرف على كيفية استخدام الدالة (free) في الغاء حجز المتغيرات التي انتهي العمل بها

الدالة (calloc)

تقوم الدالة بحجز أماكن في الذاكرة بعدد معين وتأخذ الصورة التالية :
ptr=calloc(no,size)

والدالة تقوم بنفس عمل الدالة malloc ولكن تختلف في طريقة الاستعمال حيث تقوم بحجز عدد من الاماكن هذا العدد هو no وحجم كل مكان هو size.

والبرنامج الموجود بالشكل رقم ١٣-٧ يوضح استعمال الدالة (calloc)

```
0: /*Program Name CS13_7.C*/
1: #include <stdio.h>
2: #include <stdlib.h>
3: void main( void )
4: {
5:     int no_grades = 0;
6:     int total = 0;
```

```
7:     int ctr;
8:     int * stud_grades ;
9:     char trash[80];           /* to clear keyboard buffer */
10:    while( nof_grades < 1||nof_grades >= 10000 )
11:    {
12:        printf("\nHow many grades will be entered? ==> ");
13:        scanf("%d", &nof_grades);
14:        gets(trash);          /* clear out keyboard buffer */
15:    }
16:    stud_grades = (int *) calloc( nof_grades , sizeof(int));
17:    if(stud_grades == NULL )
18:    {
19:        printf( "\nError in line %3.3d : Could not allocate memory"
19:                , __LINE__);
20:        exit(1);
21:    }
22:    for( ctr = 0; ctr < nof_grades ; ctr++ )
23:    {
24:        printf("\nEnter grade %4.4d: ", ctr+1);
25:        scanf("%d", stud_grades +ctr);
26:    }
27:    printf("\n\nYou entered the following:\n");
28:    for ( ctr = 0; ctr < nof_grades ; ctr++ )
29:    {
30:        printf("\nGrade %4.4d:", ctr+1);
31:        printf(" %d", *( stud_grades +ctr));
32:        total += *( stud_grades +ctr);
33:    }
34:    printf("\n\nThe average grade is: %d\n\n", (total/nbr_grades));
35:    free( stud_grades );
36: }
```

شكل ١٣-٧ استخدام الدالة (calloc)

وعند تفید : البرنامج نحصل على النتيجة التالية

How many grades will be entered? ==> 3

Enter grade 0001: 1000

Enter grade 0002: 789

Enter grade 0003: 5687

You entered the following:

Grade 0001: 1000

Grade 0002: 789

Grade 0003: 5687

The average grade is: 2492

شكل ١٣-٨ نتائج تنفيذ برنامج cs13-7.c

وعن هذا البرنامج نوضح ما يلى :

فى السطر رقم ١٦ يتم استخدام الدالة `calloc()` فى حجز مساحة بعدد العناصر المطلوبة وهى كما ترى تشبه بذرجة كبيرة الدالة `malloc()` لا أن قاعدة الحجز تختلف حيث أن الدالة `malloc()` تحجز مساحة مقدارها عدد البايت المكتوب بين القوسين `.malloc(50)`.

ولكن الدالة `calloc()` تأخذ الصورة `calloc(5,10)` و معناها أحجز ٥ أماكن مساحة المكان الواحد ١٠ بايت وباقى البرنامج كما فى البرنامج الموجود بالشكل رقم ١٣-٣

الدالة `faralloc()`

تقوم الدالة بحجز أماكن فى الذاكرة وتختلف هذه الدالة فى أن الاماكن التى تتحجز لها يمكن أن تتعدي ٦٤ كيلو بايت وتأخذ الصورة التالية :

`ptr= faralloc(size)`

ويقوم البرنامج الموجود فى الشكل رقم ١٠-٩ باستعمال الدالة `(faralloc()`

فى حجز أماكن فى الذاكرة تتعدى مساحتها ٦٤ كيلو بايت

```
0: /*Program Name CS13_9.C*/
1: #include <alloc.h>
```

```
2: #include <stdio.h>
3: #include <stdlib.h>
4: int do_b_p( long );
5: void main(void)
6: {
7:     int rv;
8:     unsigned long nbr_pages = 0;
9:     unsigned long page = 0;
10:    printf("\n\nEnter number of pages to do ==> ");
11:    scanf( "%d", &nbr_pages );
12:    for( page = 1; page <= nbr_pages; page++)
13:    {
14:        rv = do_b_p( page );
15:        if (rv ==100)
16:        {
17:            printf("\nAllocation error, exiting..."), exit(1);
18:        }
19:    }
20:    }
21:    printf( "\n\nDid all!\n" );
22: }
23: Int do_b_p( long page_nbr )
24: {
25:     char far *book_page; /* pointer to assign allocation to */
26:     b_page = (char *) farmalloc( 1000 );
27:     If( b_page == NULL )
28:     {
29:         printf( "\nError in line %3.3d: Could not allocate
memory.", __LINE__ );
30:         return(100);
31:     }
32:     else
33:     {
34:         printf( "\n Allocation for book page %ld is ready to
use...",page_nbr);
```

```
35:     }
36:     return(0);
37: }
```

شكل ١٣-٩ استخدام الدالة faralloc

وعندتنفيذ البرنامج الموجود بالشكل (١٣-٩) يعرض النتيجة التالية :

```
Allocation for book page 9991 is ready to use...
Allocation for book page 9992 is ready to use...
Allocation for book page 9993 is ready to use...
Allocation for book page 9994 is ready to use...
Allocation for book page 9995 is ready to use...
Allocation for book page 9996 is ready to use...
Allocation for book page 9997 is ready to use...
Allocation for book page 9998 is ready to use...
Allocation for book page 9999 is ready to use...
Allocation for book page 10000 is ready to use...
Did all the pages!
```

وعن هذا البرنامج نوضح ما يلى :

في البرنامج الموجود في الشكل رقم ١٠-٩ في السطر ٢٦ الدالة faralloc() تستخدلم لحجز ١٠٠ مكان كل مرة يتم فيها استدعاء الدالة do_b_p وهو نفس عمل الدالة malloc الا أن الدالة faralloc تتعدي في حجزها 64kb للتأكد من ذلك استبدل الدالة faralloc في هذا البرنامج بالدالة malloc ونفذ البرنامج وشاهد النتيجة تجد البرنامج باستعمال الدالة malloc يحجز عدد من الاماكن أقل.





الفصل الرابع عشر

روتينات الذاكرة

ROM BIOS

في هذا الفصل ستعرف على :

- ◆ المسجلات (registers) العامة للمعالج
- ◆ كيفية استعمال الروتينات.
- ◆ برنامج لمعرفة حجم الذاكرة الأساسية.
- ◆ استعمال ملف العناوين .dos.h
- ◆ برنامج للتعامل مع المؤشر.

تشتمل الذاكرة الثابتة (ROM) على مجموعة من التعليمات الضرورية لتشغيل الجهاز والتعامل مع مكوناته ، وعادةً تقوم الشركات الصانعة بوضع هذه التعليمات داخل الذاكرة الثابتة ، وتبقى هذه التعليمات ثابتة بحيث يمكن التعامل معها في أي وقت وتسمى (Basic Input Output System) BIOS كل مجموعة من التعليمات تقوم بوظيفة محددة أو روتين (routine) وهذه الروتينات هي المسؤولة عن عمليات الدخول والخرج وتعامل لغات البرمجة مع الروتينات (أو البرامج الصغيرة) الموجودة بالذاكرة عندما تحتاج إليها دون تدخل من المبرمج، وبذلك توفر عليه وقتاً وجهداً كبيرين كما يحدث في عمليات الدخول والخرج ولكن إستعمال هذه الروتينات مباشرة يمكنك من عمل الكثير مما لا تستطيع أن تتحققه بدوال وتعليمات اللغة مباشرة. يحقق إستعمال الروتينات مزايا أخرى كثيرة وستعرفها في هذا الفصل وقبل أن نشرح خطوات استدعاء الروتينات الموجودة في الذاكرة ROM سنشرح المقصود بالمسجلات العامة للمعالج.

المسجلات (registers) العامة للمعالج

يعتبر المعالج أهم جزء في جهاز الكمبيوتر ، وتنتج هذه المعالجات أكثر من شركة ، وكل شركة تضع معالجاتها في عائلة وتعطيها اسم فمثلاً هناك شركة "موتوريلا" التي تنتج معالجات (processors) وتعطيها الاسم "موتوريلا" وبالمثل توجد شركة "انتل" وتنتج معالجات باسم انتل وأجهزة IBM تقوم على معالجات شركة انتل وقد بدأت شركة انتل منتجاتها ابتداء من المعالج رقم ٨٠٨٨ وكان الجهاز الذي يبني على هذا المعالج يسمى XT ثم انتجت الشركة المعالج ٨٠٢٨٦ ثم ٨٠٣٨٦ ثم المعالج ٨٠٤٨٦ وأخيراً المعالج ٨٠٥٨٦ وعلى هذه المعالجات قامت أجهزة IBM والاجهزه المتواافقه معها

الفصل الرابع عشر : استعمال الروتينات الموجودة بالذاكرة ROM BIOS

وجميع المعالجات تتفق في التركيب العام ، وتشتمل المعالجات في داخلها على أجزاء دقيقة تسمى المسجلات (registers) فما هي المسجلات ، وما هي الحاجة لمعرفة هذه المسجلات ، وما هي المسجلات العامة للمعالج ؟

المسجلات : هي وحدات ذاكرة من نوع خاص تشبه عناصر الذاكرة وتستخدم لتخزين البيانات ، وهي التي يتم فيها إجراء العمليات الحسابية وإرسال المعلومات إلى الذاكرة أو استقبال البيانات منها.

معرفة هذه المسجلات ضرورية لأن استدعاء روتينات الذاكرة ROM يتم من خلالها ، وتوضع فيها معاملات الروتينات وكذلك نتائج العمليات

ما هي المسجلات العامة للمعالج ؟ intel

يقسم المعالج إلى مجموعات من المسجلات ، من هذه المجموعات مجموعة تسمى المسجلات العامة وهي التي تهمنا هنا من وجهة نظر البرمجة وتأخذ الأسماء AX,BX,CX,DX

وهذه المسجلات كما أشرنا هي وحدات ذاكرة من نوع خاص سعة التخزين الطبيعية لها 16 بت ومن مزايا أو طبيعة هذه المسجلات إمكانية التعامل معها بطريقتين :

الطريقة الأولى

معاملة كل مسجل على أنه عنصر واحد سعة تخزينه 2 بait . AX,BX,CX,DX

الطريقة الثانية

معاملة كل مسجل على أنه عنصرين سعة كل عنصر 1 بait وينظر ذلك من الجدول التالي :

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

وكما ترى في الجدول السابق أن المسجل AX يقسم إلى AH يسمى (HIGH) و AL (LOW) وهكذا BX, CX, DX قبل أن نشرح كيفية استعمال الروتينات نذكر بعض مزاياها.

مزايا استعمال الروتينات الموجودة بالذاكرة الثابتة

- إنشاء برامج باستعمال ROM BIOS ROUTINES يعطى برنامج صغير الحجم
- يعطى برنامج سريع التنفيذ
- باستعمال ROM BIOS ROUTINES تستطيع تنفيذ كثير من العمليات التي لا يمكن تنفيذها باستعمال تعليمات اللغة مباشرة.

كيفية استدعاء الروتينات الموجودة بالذاكرة (ROM BIOS)

تأخذ الروتينات الموجودة في الذاكرة شكل برنامج كل برنامج يحتوى على مجموعة دوال تؤدى عمل متقارب كلها تدور حول خدمات معينة ، وكل برنامج يسمى interrupt وكل interrupt مخصص لموضوع معين ، فمثلاً تجد interrupt مسئول عن خدمات الشاشة (video interrupt) وتختص جميع دواله بهذا الموضوع فنجد به دوال تغيير الألوان ودوال التعامل مع المؤشر وهكذا ومثال آخر هو interrupt (disk interrupt) ويهتوى على دوال لجميع الوظائف المتعلقة بالاسطوانة مثل كيفية تحديد حالة القرص سليمة أم لا وكيفية الكتابة في أي مكان عليها وكثير من الخدمات التي لا يمكن أن تؤدى إلا بهذه الروتينات.

وكل interrupt له رقم بالنظام السادس عشر (HEX) فمثلاً video interrupt رقم ۱۰ بالنظام السادس عشر (يمكنك الرجوع إلى الجداول الخاصة بهذه الروتينات الموجودة في أحد الكتب المتخصصة مثل كتاب

الفصل الرابع عشر : استعمال الروتينات الموجودة بالذاكرة ROM BIOS

الدالة () int86

هي الدالة المسئولة عن استدعاء الروتينات الموجودة في ROM والمحروفة اختصار العباره intel والرقم 86 يعني عائلة انتل وهي 80486, 80386, 80286, 8088

int86(INTno,&inregs,&outregs)

حيث أن :

المتغير INTno هو رقم interrupt المطلوب استعماله
 والمتغير &inregs هي عناوين المسجلات التي توضع فيها المدخلات
 والمتغير &outregs هي عناوين المسجلات التي توضع بها المخرجات
 ولتوضيح كيفية استعمال هذه الدالة في استدعاء الروتينات نفترض روتين كالموجود بالشكل التالي :

Rom Bios	اسم الروتين
تحديد حجم الذاكرة	الغرض منه
12 Hex	الرقم الخاص به
لا يوجد	المدخلات
توضع في المسجل AX	المخرجات

والبرنامج الموجود في الشكل رقم (١٤-١) يستخدم الروتين السابق في طباعة حجم الذاكرة الرئيسية.

```

0: /*Program Name CS14_1.C */
1: #define MEM 0x12
2: void main (void)
3: {
```

```
1: struct WORDREGS
2: {
3:     unsigned int ax;
4:     unsigned int bx;
5:     unsigned int cx;
6:     unsigned int dx;
7:     unsigned int si;
8:     unsigned int di;
9:     unsigned int flags;
10:
11: };
12:
13: struct BYTEREGS
14: {
15:     unsigned char al,ah;
16:     unsigned char bl,bh;
17:     unsigned char cl,ch;
18:     unsigned char dl,dh;
19:
20: };
21: union REGS
22: {
23:     struct WORDREGS x;
24:     struct BYTEREGS h;
25: };
26: union REGS regs;
27: int size;
28: int86(MEM,&regs,&regs);
29: size=regs.x.ax;
30: printf ("Memory size is %d Kilo Bytes",size);
31:
32: }
```

الشكل رقم ١٤-١ برنامج طباعة حجم الذاكرة

ونتيجة تنفيذ هذا البرنامج كالتالي:

Memory size is 640 Kilo Bytes

الفصل الرابع عشر : استعمال الروتينات الموجودة بالذاكرة ROM BIOS

وعن هذا البرنامج نوضح ما يلى :

- يبدأ البرنامج الموجود في الشكل رقم ١٤-١ في السطر رقم ٤ بانشاء سجل بالاسم WORDREGS وعناصره هي المسجلات العامة ولكن بصورتها الأولى وهي كل مسجل سعته 2 Byte ولذلك كان نوع المتغيرات int حيث حجم الـ int هو 2 Byte ، وكذلك unsigned حتى لا يكون له اشارة وبالتالي حجم المتغير 2 Byte .
 - وفي السطر رقم ٤ أنشأنا سجل آخر باسم BYTEREGS وعناصره هي المسجلات بالصورة الثانية وهي أن كل مسجل عبارة عن قسمين فمثلا AX يتكون من AH, AL وهكذا
 - وفي السطر رقم ٢١ تم تعريف UNION عناصره عبارة عن سجلان الأول هو من النوع WORDREGS والثاني من النوع BYTEREGS ونلاحظ هنا أن الـ UNION حق الصورة المطلوبة حيث أن المسجلات في الحالتين هما شيء واحد ولكن يمكن معاملتها بطريقتين وهذه هي طبيعة الـ UNION حيث يحجز مساحة واحدة
 - وفي السطر رقم ٢٦ أعلنا عن متغير من نوع هذا الـ UNION وأسمه regs
 - وفي السطر رقم ٢٩ تم استدعاء الدالة (int86) وتم ارسال المعاملات اليها وهي رقم Interrupt وهو كلمة MEM المعرفة في أول البرنامج بالرقم 0x12 ، ونتيجة الاستدعاء هي وضع حجم الذاكرة في المسجل AX.
 - وفي السطر رقم ٣٠ نأخذ حجم الذاكرة الموجود بـ AX ونضعه في المتغير size ثم نطبع هذه القيمة في السطر رقم ٣١

ويمكن تلخيص خطوات التعامل مع أي روتين من روتينات الذاكرة (ROM BIOS) فيما يلى :

١. تحديد رقم Interrupt
٢. تحديد رقم الدالة أن وجد

٣. تحديد المدخلات والمسجلات التي يوضع فيها المدخلات
٤. تحديد المخرجات والمسجلات التي يوضع فيها المخرجات
٥. استدعاء الدالة INT86()

استعمال ملف العناوين dos.h

في المثال السابق أنشأنا UNION بالاسم REGS يحتوى على سجلان (2) STRUCTURES) يمثلان المسجلات العامة للمعالج بالصورتين. ويتم إعلان هذا الـ Union في كل برنامج يستعمل روتينات الذاكرة ، ونظرا لأن هذا الجزء ثابت فقد تم إنشاء هذا UNION في الملف DOS.H وليس عليك إلا أن تكتب السطر #include<dos.h> في أول البرنامج وبالتالي يوفر عليك إنشاء هذا الـ Union ويصبح البرنامج الموجود في الشكل رقم (١٤-١) بسيط كما في الشكل رقم (٢-٤) هو ويعطى نفس النتيجة

```
0: /*Program Name CS11_2.C*/
1: #include <dos.h>
2: #define MEM 0x12
3: void main (void)
4: {
6:     union REGS regs;
7:     int size;
8:     clrscr();
9:     int86(MEM,&regs,&regs);
10:    size=regs.x.ax;
11:    printf ("Memory size is %d Kbytes",size);
12:    getch();
13: }
```

الشكل رقم ١٤-٢ استعمال الملف dos.h

تغییر حجم المؤشر Seting the cursor size

الروتين التالي يقوم بتغيير حجم المؤشر حيث ينقسم المؤشر(cursor) الى ١٣ خط إلّى يمكن لنا تحديد خط بداية المؤشر وخط النهاية وبالتالي تحديد حجم المؤشر ويتم تحديد الحجم بوضع رقم خط البداية في المسجل CH ورقم خط النهاية في المسجل CL ثم إستدعاء الروتين فيتغیير حجم المؤشر.

Cursor Size	إسم الروتين
تغییر حجم المؤشر	الغرض منه
0x10 hex	رقم الروتين
المسجل CH يأخذ بداية المؤشر ، المسجل CL يأخذ نهاية	المدخلات
المؤشر	المحركات
لا يوجد (تغير شكل المؤشر)	رقم الدالة 01 ويوضع في المسجل Ah

رقم الدالة 01 ويوضع في المسجل Ah
والبرنامح الموجود بالشكل رقم ١٤-٣ يقوم باستعمال الروتين السابق في تغيير حجم المؤشر.

```
0: /*Program Name CS14_3.C */  
1: #include <dos.h>  
2: #define VIDEO 0x10  
3: #define CURSIZE 1  
4: void main (int argc,char *argv[])  
6: union REGS regs;  
7: int start,end;  
8: if (argc!=3)  
9: {
```

```

10:     printf ("\n Example usage: C>setcurs 12 13");
11:     exit();
12: }
13: start=atoi(argv[1]);
14: end=atoi(argv[2]);
15: regs.h.ch=(char)start;
16: regs.h.cl=(char)end;
17: regs.h.ah=CURSIZE;
18: int86(VIDEO,&regs,&regs);
19: }

```

الشكل رقم ١٤-٣ تغير حجم المؤشر

وعن هذا البرنامج نوضح ما يلى :

قم بترجمة البرنامج وتنفيذه باستخدام الأمر التالي من ممحث DOS :

CS14_3 2 6

حيث أن المتغير argc يحتوى على ٣ معادلات والمعاملات تخزن في

المصفوفة argv بالترتيب التالي :

argv[0],argv[1],argv[2]

وعلى ذلك فإن القيمة ٢ التي استقبلها البرنامج عند استدعائه تخزن في العنصر

.argv[1] والقيمة ٦ تخزن في العنصر [2]

• وفي السطر رقم ١٣ الدالة () atoi تحول المعامل الاول (القيمة ٢) الى رقم

صحيح لأنها يستقبل على أنه STRING ويخزن في المتغير start وبالمثل

.end المتغير.

• في السطر رقم ١٧ ، نضع في المسجل ah قيمة المتغير CURSIZE المعرفة

في أول البرنامج بالقيمة ١ وهو رقم الدالة الخاصة بتغيير حجم المؤشر.

• وفي السطر رقم ١٨ يتم استدعاء الدالة (int86()) بالمعاملات التي تم ضبطها تكون النتيجة هي تغير حجم المؤشر بالقيم المحددة كمعاملات للبرنامج .
وتنلاحظ في هذا الأمر أن استدعاء البرنامج للتنفيذ يتم بمعاملين هما ٢ ، ٦ وبالرجوع إلى البرنامج الموجود في شكل ٤-٣ تجد أن الدالة main تحتوى على متغيرين هما Argv , argc

اخفاء المؤشر

من الاستخدامات المتاحة للروتين السابق الذى استعمل لتغيير حجم المؤشر اخفاء المؤشر وذلك بوضع القيمة 0x20 في المسجل ch واستدعاء نفس الروتين .
والبرنامج الموجود بالشكل رقم (٤-٤) . ويقوم باخفاء المؤشر

أكتب هذا البرنامج ونفذه وشاهد النتيجة :

```
0: /*Program Name CS14_4.C*/
1: #include <dos.h>
2: #define VIDEO 0x10
3: #define CURSIZE 1
4: #define STOPBIT 0x20
5: void main (int argc,char *argv[])
6: {
7:     union REGS regs;
8:     regs.h.ch=STOPBIT;
9:     regs.h.ah=CURSIZE;
10:    int86(VIDEO,&regs,&regs);
11: }
```

الشكل رقم ٤-٤ اخفاء المؤشر

دوال لغة C التي تستعمل ROM BIOS مباشرة

بالإضافة لامكانية استعمال الروتينات الموجودة في ROM توجد مجموعة من الدوال قامت على هذه الروتينات بحيث توفر علينا كتابة هذه الروتينات من البداية منها الدوال التالية :

الدالة	الغرض منها
_bios_disk	تقوم بخدمات كلا من القرص الصلب (HARD DISK) والمرن
_bios_equiplist	تقوم بجمعية الاختبارات المطلوبة على مكونات الجهاز.
_bios_keybrd	تحقق جميع خدمات لوحة المفاتيح
_bios_memsize	تعطينا معلومات عن الذاكرة المتاحة
_bios_printer	تحقق جميع خدمات الة الطباعة
_bios_serialcom	تحقق خدمات منفذ التوالي الخاص بالجهاز (SERIAL PORT)
_bios_timeofday	التعامل مع ساعة الجهاز

ولا يتسع المجال هنا لتفصيلها ولكن يمكن لك أن ترجع إلى مرجع الدوال الخاص بلغة C (LIB REFERENCE) لمعرفتها واستعمالها وقت الحاجة أوصيتك : هناك مجموعة من البرامج النص الكامل لها موجود على الاسطوانة المصاحبة للكتاب لك أن تفتح هذه البرامج وتنفذها مباشرة.



مثال تطبيقى

متكملى

يحتوى هذا الفصل على برنامج متكملى يشتمل على
معظم الموضوعات التى ناقشناها فى الفصول السابقة .

وهذا البرنامج عبارة عن برنامج قواعد بيانات بسيط
يحتوى على معظم العمليات المطلوبة حيث يمكننا البرنامج من
إضافة أو حذف بيانات أو التعديل فى البيانات وكذلك اظهار
البيانات على الشاشة او طباعتها على الة الطباعة .

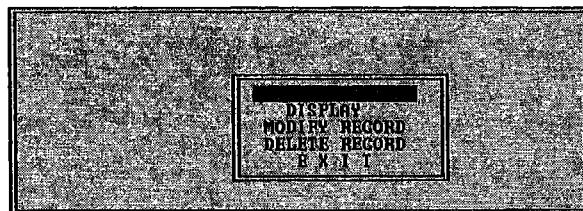
ويشتمل هذا الفصل على:

- ♦ متابعة مراحل تنفيذ البرنامج
- ♦ عرض نص البرنامج
- ♦ ملخص سريع عن أجزاء البرنامج

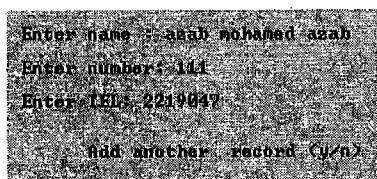
أولاً: متابعة تنفيذ البرنامج

اكتبه البرنامج كما هو مكتوب في نص البرنامج أو إفتح ملف البرنامج من القرص المصاحب للكتاب وتتابع معنا مراحل التنفيذ.

عند تنفيذ البرنامج تظهر أول شاشة وهي الشاشة الرئيسية للبرنامج كما في الشكل ١٥-١ وتحتوي على إختيارات البرنامج الرئيسية ويستخدم مفاتيح الاسم نحرك الشريط المضاء ونحدد الاختيار المطلوب تنفيذه ولنبدأ بالاختيار الاول ثم الضغط على مفتاح الادخال نحصل على شاشة ادخال البيانات كما في الشكل ١٥-٢



شكل رقم ١٥-١



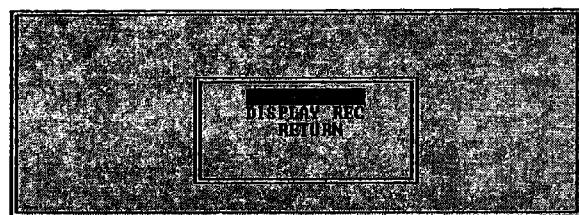
شكل رقم ١٥-٢

ادخل بيانات كما في الشكل وفي النهاية تحصل على رسالة

Add another record (y/n)

أدخل الحرف n وبالتالي تعود إلى الشاشة الرئيسية

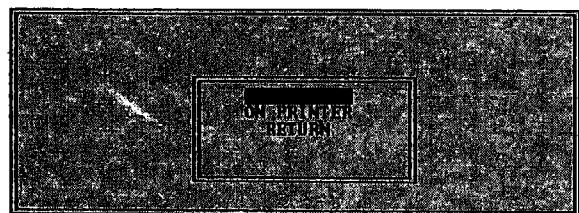
تحرك بالاسهم إلى الاختيار الثاني وهو display ثم اضغط مفتاح الادخال تحصل على الشكل ١٥-٣ وبه إختيارات هما display record و display all



شكل ١٥-٣

حرك الشريط المضاء إلى الاختيار الأول ثم اضغط على مفتاح الادخال تحصل على شاشة بها إختيارات هما on screen و on printer كما بالشكل ١٥-٤ اختر الاول ثم اضغط على مفتاح الادخال تحصل على شاشة التقارير على الشاشة كما بالشكل ٥

١٥



شكل ١٥-٤

بعد متابعة عرض البيانات على الشاشة اضغط اي مفتاح ترجع الى شاشة العرض (display) مرة اخرى ، اختر الاختيار الثاني وهو display rec ثم اضغط مفتاح الادخال تحصل على شاشة عرض بيانات سجل كما في الشكل ١٥-٦ اكتب الاسم المراد عرض بياناته ثم اضغط مفتاح الادخال تظهر بيانات السجل المطلوب كما

١٥-٧ في الشكل

NAME	NUMBER1	NUMBER2
..... azab mohamed a.....	111.....	2219047.....

١٥-٥ شكل



١٥-٦ شكل

```
*****
Name:azab mohamed azab
Number:111
TEL:2219047
***again<y/n>> *****
```

١٥-٧ شكل

وبنفس الاسلوب يمكن لك متابعة تنفيذ البرنامج بتنفيذ الاختبارات الباقيه ومشاهدة النتائج.

نعر البرنامج

فيما يلي النص الكامل للبرنام

```
#define TRUE 1
#define NUM 5
#define NUM2 3
#define CLEAR "\x1B[2J"
#define ERASE "\x1B[K"
```

```
#define NORMAL  "\x1B[0m"
#define REVERSE "\x1B[7m"
#define HOME   "\x1B[10;5f"
#define BOTTOM  "\x1B[20;1f"
#define U_ARRO  72
#define D_ARRO  80
#define INSERT  82
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <dos.h>
*****functions declarations part (1) *****
void dispp();
void displayallp(void);
void action2(int pos2);
void action3(int pos2);
void ex();
void add(void);
void action4 (int pos);
void displayall(void);
void displayrec(void);
void modify(void);
void del(void);
void disp();
void call();
void action(int);
void display (char *arr[],int size,int pos);
char getcode(void);

*****
char ag;
```

```
char namec[40];
char ch,y1;
***** THE RECORD STRUCTURE *****/
struct
{
    char name[40];
    char agnumb[10];
    char tel[20];
} agent;
*****
char numstr[81];
FILE *fptr,*fptr2;
***** BOX FUCTIONS *****/
void set_window1()
{
    window (2,2,79,9);
}
void set_window2()
{
    window(2,12,79,23);
}
void fancy_box(int x1,int y1,int x2,int y2)
{
    int i;
    gotoxy(x1,y1);putch(201);
    for(i=x1+1;i<x2;i++) putch(205);
    putch(187);
    for(i=y1+1;i<y2;i++) {
        gotoxy(x1,i);putch(186);
        gotoxy(x2,i);putch(186);
    }
    gotoxy(x1,y2);putch(200);
    for (i=x1+1;i<x2;i++) putch(205);
    putch(188);
```

```
}

void box_it(int x1,int y1,int x2,int y2)
{
    int i;
    gotoxy(x1,y1);putch(218);
    for (i=x1+1;i<x2;i++) putch (196);
    putch(191);
    for (i=y1+1;i<y2;i++) {
        gotoxy(x1,i);putch(179);
        gotoxy(x2,i);putch(179);
    }
    gotoxy(x1,y2);putch(192);
    for (i=x1+1;i<x2;i++) putch(196);
    putch(217);
}

***** MENUS STRUCTURES *****

static char *item[NUM2]=
{
    "DISPLAY ALL",
    "DISPLAY REC",
    " RETURN ";
};

static char *itemp[NUM2]=
{
    "ON SCREEN",
    "ON PRINTER",
    " RETURN ";
};

static char *iteme[NUM2]=
{
    "E N D",
    "DOS SHELL",
    " RETURN ";
};

static char *items[NUM]=
{
    "ADD RECORD",
    " DISPLAY",
    "MODIFY RECORD",
    "DELETE RECORD",
}
```

```
    EXIT "");  
***** MAIN PROGRAM main() function *****  
void main (void)  
{  
***** COLORS AND BOXES*****  
    textbackground(BLUE);  
    textcolor(WHITE);  
    clrscr();  
    fancy_box(1,1,80,24);  
    fancy_box(8,8,60,20);  
    fancy_box(28,12,47,18);  
    clrscr();  
    call();  
}  
void call()  
{  
int curpos,s;  
curpos=0;  
  
while (TRUE)  
{  
    textbackground(BLUE);  
    textcolor(WHITE);  
    clrscr();  
  
    clrscr();  
    fancy_box(1,1,80,24);  
    fancy_box(8,8,60,20);  
    fancy_box(28,12,47,18);  
***** DISPLAY MAIN MENU BY USING display()*****  
    display(items,NUM,curpos);  
}
```

```
switch (getcode())
{
    case U_ARRO:
        if(curpos>0)
        {
            -- curpos;
            sound(800);
            delay(20);
            nosound();
        }
        else
            curpos+=(NUM-1);
        break;
    case D_ARRO:
        if(curpos<NUM-1)
        {
            ++curpos;
            sound(900);
            delay(20);
            nosound();
        }
        else
            curpos-=(NUM-1);
        break;
    case 'r':
        for (s=0;s<10;s++)
        {
            sound(s*100);
            delay(50);
            nosound();
        }
        action(curpos);
        break;
}
```

```
        }
    }
}

//***** END OF main() FUNCTION *****

//***** display function() *****
void display (char *arr[],int size,int pos)
{
    int j;
    printf (HOME);
    for (j=0;j<size;j++)
    {
        if(j==pos)
        {
            textColor(2);
            textbackground(4);
        }
        gotoxy(30,13+j);
        cprintf ("%s\n",*(arr+j));
        textColor(WHITE);
        textbackground(BLUE);
    }
    printf (BOTTOM);
}
//***** getcode(void) function *****
char getcode(void)
{
    int key;
    if ((key=getch())==0)
        return (getch());
    else if (key=='r')
        return(key);
    else
        return (0);
}
```

```
        }
***** action(int pos) function *****
void action(int pos)
{
    printf (ERASE);
    switch (pos)
    {
        case 0:
            add();
            break;
        case 1:
            disp();
            break;
        case 2:
            modify();
            break;
        case 3:
            del();
            break;
        case 4:
            ex();
            break;
        default:
            printf("\n unkowen");
    }
}

***** disp() function *****
void disp()
{
    int curpos2=0;
    clrscr();
    while (TRUE)
```

```
textbackground(BLUE);
textcolor(WHITE);
clrscr();
fancy_box(1,1,80,24);
fancy_box(8,8,60,20);
fancy_box(25,12,45,18);
display(item,NUM2,curpos2);
switch (getcode())
{
    case U_ARRO:
        if(curpos2>0) -- curpos2;
        else
            curpos2+=(NUM2-1);
        break;
    case D_ARRO:
        if(curpos2<NUM2-1) ++curpos2;
        else
            curpos2 -=(NUM2-1);
        break;
    case 'l':
        action2(curpos2);
        break;
}
}
}

***** action2(int pos) *****

void action2(int pos)
{
    int s;
    printf(ERASE);
    switch (pos)
    {
        case 0:
```

```
        dispp();
        break;
case 1:
        displayrec();
        break;
case 2:
        for (s=0;s<10;s++)
        {
            sound(s*100);
            delay(50);
            nosound();
        }
        call();
        break;

    }

}

***** dispp() function *****
void dispp()
{
    int curpos2=0;
    clrscr();
    while (TRUE)
    {
        textbackground(BLUE);
        textcolor(WHITE);
        clrscr();
        fancy_box(1,1,80,24);
        fancy_box(8,8,60,20);
        fancy_box(25,12,45,18);
        display(itemp,NUM2,curpos2);
        switch (getcode())
        {
            case U_ARRO:
```

```
if(curpos2>0) -- curpos2;
else
    curpos2+=(NUM2-1);
break;
case D_ARRO:
    if(curpos2<NUM2-1) ++curpos2;
else
    curpos2-=(NUM2-1);
break;
case 'r':
    action4(curpos2);
break;
}
}
}

***** action4 (int pos) function *****
void action4 (int pos)
{
int s;
printf (ERASE);
switch (pos)
{
case 0:
    displayall();
    break;
case 1:
    displayallp();
    break;
case 2:
    for (s=0;s<10;s++)
    {
        sound(s*100);
        delay(50);
        nosound();
    }
}
```

```
        }
        call();
        break;

    }

}

***** ex() function *****
void ex()
{
    int curpos2=0;
    clrscr();

    while (TRUE)
    {
        textbackground(BLUE);
        textcolor(WHITE);
        clrscr();
        fancy_box(1,1,80,24);
        fancy_box(8,8,60,20);
        fancy_box(25,12,45,18);
        display(iteme,NUM2,curpos2);
        switch (getcode())
        {
            case U_ARRO:
                if(curpos2>0) --curpos2;
                else
                    curpos2+=(NUM2-1);
                break;
            case D_ARRO:
                if(curpos2<NUM2-1) ++curpos2;
                else
                    curpos2-=(NUM2-1);
                break;
        }
    }
}
```

```
case 'r':
    action3(curpos2);
    break;
}
}
}

***** action3(int pos) function *****
void action3(int pos)
{
    int s;
    printf (ERASE);
    switch (pos)
    {
        clrscr();
        case 0:
            clrscr();
            cputs("\n\n\n\n\n\n\n");
            cputs(".... thank's and goodby ENG.AZAB....");
            for (s=5;s<10;s++)
            {
                cputs("\n\n\n\n\n\n");
                cputs("..... thank's and goodby ENG.AZAB....");
                sound(s*50);
                delay(450);
                nosound();
                sound(s*150);
                delay(350);
                nosound();

                sound(s*120);
                delay(450);
                nosound();

                sound(s*120);
```

```
delay(450);
nosound();
sound(s*120);
delay(450);
nosound();

}

exit(0);
break;
case 1:
clrscr();
system("command.com");
printf ("\n\n\n TYPE EXIT TO RETURN TO MAIN
PROG..");
break;
case 2:
for (s=0;s<10;s++)
{
sound(s*100);
delay(50);
nosound();
}
call();
break;

}
}

***** add(void) function *****/
void add(void)
{
int s;
clrscr();

if( (fptr=fopen("agents.rec","ab"))==NULL)
```

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <dos.h>
#include <time.h>
#include <graph.h>

main()
{
    clrscr();
    cputs("Enter name : ");
    gets(agent.name);
    cputs(" Enter number: ");
    gets(agent.agnumb);
    agent.agnumb=atoi(numstr); */
    cputs ("r\n Enter tel: ");
    gets(agent.height);
    agent.height=atof(numstr); */
    fwrite(&agent,sizeof(agent),1,fptr);
    cputs ("r\n      Add an other record (y/n)");
    }
    while(toupper(getche())=="Y");
    fclose(fptr);
    for (s=0;s<10;s++)
    {
        {
            sound(s*100);
            delay(50);
            nosound();
        }
        clrscr();
    }

/************* displayrec(void) function *****/
void displayrec(void)
{
    int s;
    clrscr();
    if( (fptr=fopen("agents.rec","rb"))==NULL)
    { printf ("can't open file agents.rec ");exit(1);}
}
```

```
clrscr();
cprintf ("\n Enter name:");
gets(namec);
while ((fread(&agent,sizeof(agent),1,fptr)==1))
{
    if (strcmp(namec,agent.name)==0)
    {
        textattr(80);
        clrscr();
        cputs ("\r\n*****");
        cputs ("\r\n\r\n\r\n");
        cputs ("      Name:");
        cputs(agent.name);
        cputs ("\r\n\r\n");
        cputs("      Number:");
        cprintf ("%s",agent.agnumb);
        cputs ("\r\n\r\n");
        cputs("      tel:");
        cprintf ("%s",agent.height);
        cputs ("\r\n      ***again(y/n)> *****");
    }
}
fclose(fptr);
y1=getch();
for (s=0;s<10;s++)
{
    sound(s*100);
    delay(50);
    nosound();
}
clrscr();
}

***** displayallp(void) function *****
void displayallp(void)
```

```

{
    int s;
    int n=5;
    clrscr();
    if( (fptr=fopen("agents.rec","rb"))==NULL)
        { printf ("can't open file agents.rec ");exit(1);}
    fprintf (stdprn,"_____\n");
    fprintf (stdprn,"|t|t NAME      NUMBER1      NUMBER2 \n");
    fprintf (stdprn,"_____\n");
    while ((fread(&agent,sizeof(agent),1,fptr)==1))
    {
        fprintf (stdprn,"..... %s.....",agent.name);
        fprintf (stdprn,".....%s.....",agent.agnumb);
        fprintf (stdprn,".....%s.....\n",agent.height);
        fprintf (stdprn,"|t _____\n");
        sound(200*n);
        delay(200);
        n++;
        if (n==14)
        {
            nosound();
            getch();
            clrscr();
            n=5;
        }
        fprintf (stdprn,"_____\n");
        fprintf (stdprn,"|t|t NAME      NUMBER1      NUMBER2 \n");
        fprintf (stdprn,"_____\n");
    }
    fclose(fptr);
    getch();
    nosound();
    for (s=0;s<10;s++)
    {
}

```

```
        sound(s*100);
        delay(50);
        nosound();
    }
    clrscr();
}
***** displayall(void) *****
void displayall(void)
{
    int s;
    int n=5;
    clrscr();
    if( (fptr=fopen("agents.rec","rb"))==NULL)
    { printf ("can't open file agents.rec ");exit(1);}
    textattr(30);
    clrscr();
    printf ("_____\n");
    printf ("|tit NAME      NUMBER1      NUMBER2 |\n");
    printf ("|_____|\n");
    while ((fread(&agent,sizeof(agent),1,fptr)==1))
    {
        gotoxy(5,n);
        cprintf ("..... %s.....",agent.name);
        gotoxy(25,n);
        cprintf (".....%s.....",agent.agnumb);
        gotoxy(35,n);
        cprintf (".....%s.....\n",agent.height);
        printf ("|t _____|\n");
        sound(200*n);
        delay(200);
        n++;
        if (n==14)
        {
            nosound();
        }
    }
}
```

```

getch();
clrscr();
n=5;
printf("_____\n");
printf("Itlt NAME      NUMBER1      NUMBER2 \n");
printf("_____\n");
}
}

fclose(fptr);
getch();
nosound();
for (s=0;s<10;s++)
{
    sound(s*100);
    delay(50);
    nosound();
}
clrscr();
}

***** modify(void) function *****

void modify(void)
{
    int s;
    clrscr();
    if ((fptr=fopen("agents.rec","rb"))==NULL)
        { printf ("can't open file agents.rec ");exit(1);}
    if ((fptr2=fopen("agents2.rec","wb"))==NULL)
        { printf ("can't open file agents2.rec ");exit(1);}
    clrscr();
    cprintf ("\n Enter name:");
    gets(namec);
    while ((fread(&agent,sizeof(agent),1,fptr)==1))
    {
        if (strcmp(namec,agent.name)==0)

```

```
{  
    textattr(80);  
    clrscr();  
    cputs ("\r\n      ***** OLD DATA *****");  
    cputs("\r\n\r\n\r\n");  
    cputs ("      Name:");  
    cputs(agent.name);  
    cputs ("\r\n\r\n");  
    cputs("      Number:");  
    cprintf ("%s",agent.agnumb);  
    cputs ("\r\n\r\n");  
    cputs("      tel:");  
    cprintf ("%s",agent.height);  
    cputs ("\r\n      ***** PRESS ANY KEY *****");  
    getch();  
    clrscr();  
    cputs("\r\n===== N E W D A T A =====\r\n");  
    cputs (" Enter name2 : ");  
    gets(agent.name);  
    cputs ("\r\n Enter number2: ");  
    gets(agent.agnumb);  
    agent.agnumb=atoi(numstr);/*  
    cputs ("\r\n Enter tel2: ");  
    gets(agent.height);  
    agent.height=atof(numstr); */  
}  
  
fwrite(&agent,sizeof(agent),1,fptr2);  
  
fclose(fptr);  
fclose(fptr2);  
system("del agents.rec");  
system("ren agents2.rec agents.rec")  
getch();  
for (s=0;s<10;s++)
```

```
        {
            sound(s*100);
            delay(50);
            nosound();
        }
    clrscr();
}

***** del(void) function *****
void del(void)
{
    int s;
    textattr(80);
    clrscr();
    if( (fptr=fopen("agents.rec","rb"))==NULL)
    { printf ("can't open file agents.rec ");exit(1);}
    if( (fptr2=fopen("agents2.rec","wb"))==NULL)
    { printf ("can't open file agents2.rec ");exit(1);}
    clrscr();
    printf ("\n Enter name:");
    gets(namec);
    do
    {
        while ((fread(&agent,sizeof(agent),1,fptr)==1))
        {
            if (strcmp(namec,agent.name)==0)
            {
                textattr(80);
                clrscr();
                cputs("\r\n\r\n");
                cputs ("      Name:");
                cputs(agent.name);
                cputs ("\r\n\r\n");
                cputs("      Number:");
                cprintf ("%s",agent.agnumb);
            }
        }
    } while (strcmp(namec,agent.name));
}
```

```
cputs ("\r\n\r\n");
cputs("      tel:");
cprintf ("%s",agent.height);
cputs ("\r\n***** PRESS ANY KEY *****");
cputs ("\n ARE YOU SURE (Y/N)=====>");
ch=getche();
}
}
fwrite(&agent,sizeof(agent),1,fptr2);
}
while (toupper(ch)!="Y");
fclose(fptr);
fclose(fptr2);
system("del agents.rec");
system("ren agents2.rec agents.rec");
getch();
for (s=0;s<10;s++)
{
    sound(s*100);
    delay(50);
    nosound();
}
clrscr();
}
*****
/*****
```

شكل ١٥-٨ نص البرنامج

الشرم

لقد قسمنا البرنامج إلى الأجزاء التالية

***** functions declarations- part 1 ****
وهو خاص بالإعلان عن الدوال التي سوف تستخدم في البرنامج.

*****THE RECORD STRUCTURE- part2*****

وهو تركيب سجل البيانات الذي يستعمل في تسجيل البيانات في ملف قواعد البيانات

*****BOX FUCTIONS- part3*****

ويحتوى على مجموعة دوال تقوم برسم مستطيل (اطار) يتم استدعائها مع الشاشات

*****MENUS STRUCTURES-part4*****

تركيب السجلات التي تحتوى على محتويات شاشات البرنامج مثل الشاشة الرئيسية

شاشة الاظهار

***** MAIN PROGRAM main() function-part5*****

بداية الدالة الرئيسية وبها الاجزاء التالية

- **COLORS AND BOXES** و فيها تحدد الالوان و ترسم اطار الشاشة الرئيسية
- **DISPLAY MAIN MENU BY USING display()** و فيها استدعاء دالة الاظهار () display() كما في الفصل السابع والتي تقوم باظهار الشاشة الرئيسية
- **display()** طبورة دالة الاظهار القائمة الرئيسية كما في الفصل السابع
- **getcode(void)** الدالة تقوم باستقبال اختيار المستخدم و ارساله الى الدالة الرئيسية التي تقوم بدورها بتحديد هذا الاختيار و تستدعي الدالة المناسبة لهذا الاختيار
- **action(int pos)** الدالة التي تستدعيها الدالة الرئيسية لتنفيذ اختيار المستخدم حيث تقوم هذه الدالة باستعمال رقم الاختيار الذي تحدده الدالة الرئيسية باستدعاء الدالة المقابلة للاختيار
- **disp()** الدالة و تقوم هذه الدالة باظهار شاشة الاظهار رقم (١)
- **action2(int pos)** الدالة و تقوم باستدعاء الدالة المناسبة لاختيارات شاشة

الاظهار

- الدالة disp() هي دالة اظهار البيانات على الة الطباعة أم على الشاشة
- الدالة add(void) وهي دالة اضافة بيانات
- الدالة displayrec(void) هي دالة اظهار بيانات سجل
- الدالة displayallp(void) وهي دالة اظهار جميع البيانات على الة الطباعة
- الدالة displayall(void) وهي دالة اظهار جميع البيانات على الشاشة
- الدالة modify(void) function دالة التعديل في بيانات سجل
- الدالة del(void) هي دالة حذف بيانات سجل

بهذا التطبيق تكون قد انتهينا معك من فصول كتاب المرجع الاساسي للغة C والذى نرجو أن يؤدى الشمرة المرجوة منه وأن يعود على قارئية بالفع ،على أن نلتقي قريبا مع برمجة التوافذ و VISUAL C++ و

(وآخر دعوانا أن الحمد لله رب العالمين)



الملاحق

الملحق الأول التعامل مع البيئة المتكاملة لكتابه ببرنامج C

الملحق الثاني الكلمات الممحوزة في لغة C

الملحق الثالث شفرة تبادل المعلومات ASCII

الملحق الأول

التعامل مع محرر كتابة البرامج

يتناول هذا الملحق الموضوعات التالية:

- كيفية تشغيل بيئة C والخروج منها

- الأوامر المستخدمة أثناء تحرير البرنامج

تشغيل بيئة C والخروج منها

للدخول الى حزمة برنامج C اكتب الأمر الآتي:

c:\>tc

وذلك بفرض أن حزمة البرامج الخاصة بمترجم لغة C موجودة تحت الدليل tc

من محث DOS اكتب الأمر DIR ، ستظهر لك الأدلة التالية:

bin
include
lib
bgi
examples

- الفهرس bin يحتوى على الملفات التنفيذية لغة مثل ملف التشغيل tc.exe
- الفهرس include يحتوى على جميع الملفات التي لها الامتداد h والتي نسميتها header files والتي يوجد بها تعريف جميع دوال لغة C
- الدليل lib يحتوى على ملفات مكتبة دوال لغة C والتي يتم ربطها مع أي برنامج حسب الدوال المستخدمة

- الدليل **bgi** يحتوى على الملفات الخاصة بتهيئة الجهاز للرسم وكذلك ملفات الفونتات المعروفة باللغة
 - الدليل **examples** يحتوى على مجموعة من الأمثلة المتنوعة التي يمكن فتحها وتنفيذها ولكلى تتمكن من تشغيل بيئه كتابه البرنامج بسهولة ، وكذلك ترجمة وتنفيذ البرنامج بيسر أضف هذا السطر في الملف autoexec.bat


```
path =c:\tcl\bin;c:\tcl\lib;c:\tcl\include;
```

 للخروج من البيئة المتكاملة لكتابه البرنامج اما أن تضغط مفتاح F10 ثم تختار أمر Quit أو تضغط المفاتيح Alt+x معا.
 ولمشاهدة نتيجة البرنامج اضغط مفاتحي ALT+F5
- ### الأوامر المستخدمة أثناء تحرير البرنامج

أولاًً أوامر تحريك المؤشر

الأمر أو المفتاح	الوظيفة
Ctrl-S or Left arrow	التحرك لليسار حرف واحد
Ctrl-D or Right arrow	التحرك لليمين حرف واحد
Ctrl-A	التحرك لليسار كلمة واحدة
Ctrl-F	التحرك لليمين كلمة واحدة
Ctrl-E or Up arrow	التحرك سطر لأعلى
Ctrl-X or Down arrow	التحرك سطر لأسفل
Ctrl-W	طى الشاشة لأعلى
Ctrl-Z	طى الشاشة لأسفل

الوظيفة	الأمر أو المفتاح
صفحة لأعلى	Ctrl-R or PgUp
صفحة لأسفل	Ctrl-C or PgDn

ثانياً: أوامر إضافة وحذف حرف أو كلمة أو سطر

الوظيفة	الأمر أو المفتاح
الكتابة في وضع الأدراج	Ctrl-V or Ins
ادراج سطر	Ctrl-N
حذف سطر	Ctrl-Y
حذف إلى نهاية السطر	Ctrl-QY
حذف حرف لليسار	Ctrl-H or Backspace
حذف حرف	Ctrl-G or Del
حذف كلمة لليمين	Ctrl-T

ثالثاً: أوامر التعامل مع عادة سطور

الوظيفة	الأمر أو المفتاح
بداية تعليم مجموعة سطور	Ctrl-K B
نهاية تعليم مجموعة سطور	Ctrl-K K
تعلم كلمة واحدة	Ctrl-K T
نسخ مجموعة سطور تم تعليمها	Ctrl-K C
نقل مجموعة سطور تم تعليمها	Ctrl-K V
حذف مجموعة سطور تم تعليمها	Ctrl-K Y
قراءة ملف أو مجموعة سطور من القرص	Ctrl-K R

الملاحق

الوظيفة	الأمر أو المفتاح
كتابة مجموعة سطور الى الفرنس	Ctrl-K W
ازالة تعليم مجموعة سطور	Ctrl-K H
طباعة مجموعة سطور معلمة	Ctrl-K P
ضبط هامش مجموعة سطور معلمة	Ctrl-K I
الغاء ضبط هامش مجموعة سطور معلمة	Ctrl-K U

رابعاً أوامر البحث والاستبدال

الوظيفة	الأمر أو المفتاح
بحث	Ctrl-Q F
بحث واستبدال	Ctrl-Q A
تكرار آخر بحث	Ctrl-L
الغاء العملية	Esc

خامسياً: أوامر متعددة

الوظيفة	الأمر أو المفتاح
فتح سطر القوائم	F10
حفظ	Ctrl-K S or F2
ملف جديد	F3
لاغلاق النافذة النشطة	Alt-F3
جدولة	Ctrl-I or Tab
محاذاة تلقائية	Ctrl-O I
استرجاع سطر	Ctrl-Q L

الملحق الثاني

الكلمات المحجوزة في لغة C

C Language Keywords

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

الملحق الثالث

شفرة تبادل المعلومات ASCII

ASCII Value	Character
الشفرة	الحرف
000	(null)
001	^
002	●
003	▼
004	◆
005	♣
006	♦
007	(beep)
008	█
009	(tab)
010	(line feed)
011	(home)
012	(form feed)
013	(carriage return)
014	♪
015	✿
016	▶
017	◀
018	↑
019	↓
020	π
021	⌚
022	▬
023	▬
024	▬
025	▬
026	▬
027	▬
028	(cursor right)
029	(cursor left)
030	(cursor up)
031	(cursor down)

ASCII Value الشفرة	Character الحرف	ASCII Value الشفرة	Character الحرف
032	(space)	069	E
033	!	070	F
034	"	071	G
035	#	072	H
036	\$	073	I
037	%	074	J
038	&	075	K
039	,	076	L
040	(077	M
041)	078	N
042	*	079	O
043	+	080	P
044	.	081	Q
045	.	082	R
046	.	083	S
047	/	084	T
048	0	085	U
049	1	086	V
050	2	087	W
051	3	088	X
052	4	089	Y
053	5	090	Z
054	6	091	[
055	7	092	\
056	8	093]
057	9	094	^
058	:	095	-
059	:	096	,
060	<	097	a
061	=	098	b
062	>	099	c
063	?	100	d
064	@	101	e
065	A	102	f
066	B	103	g
067	C	104	h
068	D	105	i

الملاحق

ASCII Value	Character	الحرف	ASCII Value	Character	الحرف
106	j	ج	143	À	أ
107	k	ك	144	È	إ
108	l	ل	145	ë	ئ
109	m	م	146	Ã	آ
110	n	ن	147	ô	و
111	o	و	148	ö	ؤ
112	p	پ	149	û	ۈ
113	q	ق	150	ñ	ڻ
114	r	ر	151	ü	ۈ
115	s	س	152	ÿ	ڙ
116	t	ت	153	ò	و
117	u	ۇ	154	ú	ۈ
118	v	ۋ	155	¢	ڦ
119	w	ۋ	156	£	ڦ
120	x	خ	157	¥	ڦ
121	y	ي	158	Þ	ڦ
122	z	ڙ	159	ƒ	ڦ
123	ı	ى	160	á	ا
124	ı	ى	161	í	ي
125	ı	ى	162	ó	ؤ
126	~	ـ	163	ú	ۈ
127	□	□	164	ñ	ڻ
128	Ç	ڇ	165	ñ	ڻ
129	ü	ۈ	166	đ	ڏ
130	é	ئ	167	ö	ؤ
131	à	ا	168	ç	ڇ
132	ä	ا	169	ł	ڻ
133	à	ا	170	ł	ڻ
134	ä	ا	171	½	ڻ
135	¢	ڦ	172	¼	ڻ
136	ë	ئ	173	‘	‘
137	ë	ئ	174	’	’
138	ë	ئ	175	”	”
139	í	ي	176	”	”
140	í	ي	177	‰	‰
141	í	ي	178	‰	‰
142	A	ا	179	—	—

ASCII Value الشفرة	Character الحرف	ASCII Value الشفرة	Character الحرف
180	ـ	218	■
181	=	219	▬
182	ـ	220	▬
183	ـ	221	▬
184	=	222	▬
185	ـ	223	▬
186	==	224	ػ
187	ـ	225	ػ
188	ـ	226	ػ
189	ـ	227	ػ
190	ـ	228	ػ
191	ـ	229	ػ
192	ـ	230	ػ
193	ـ	231	ػ
194	ـ	232	ػ
195	ـ	233	ػ
196	ـ	234	ػ
197	ـ	235	ػ
198	ـ	236	ػ
199	ـ	237	ػ
200	ـ	238	ػ
201	ـ	239	ػ
202	ـ	240	ػ
203	ـ	241	ػ
204	ـ	242	ػ
205	ـ	243	ػ
206	ـ	244	ػ
207	ـ	245	ػ
208	ـ	246	ػ
209	ـ	247	ػ
210	ـ	248	ػ
211	ـ	249	ػ
212	ـ	250	ػ
213	ـ	251	✓
214	ـ	252	ػ
215	ـ	253	ػ
216	ـ	254	ػ
217	ـ	255	(blank 'FF')

المرجع الأساسي لنظام التشغيل

MS-DOS 5 / MS-DOS6

يقع هذا الكتاب في جزئين ، ويشرح جميع اصدارات نظام التشغيل MS-DOS ابتداء من الاصدار 1.1 الى الاصدار 6 ويركز بصفة أساسية على الاصدارات بين الخامس والسادس .

يشتمل الجزء الأول على الأبواب الرئيسية التالية :

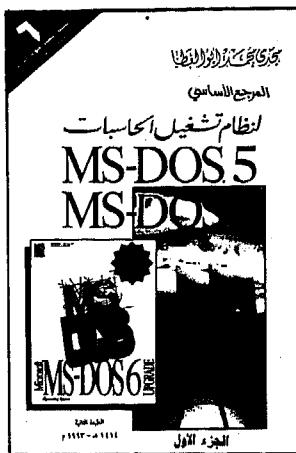
- ◆ تعرف على الحاسوب الشخصي
- ◆ دروس عملية للمبتدئين
- ◆ التعامل مع منسق النصوص Editor
- ◆ استخدام "دوس شيل"
- ◆ مرجع شامل للأوامر
- ◆ التعامل مع الملفات والأدلة والأقواس
- ◆ مفاهيم متقدمة تشمل :

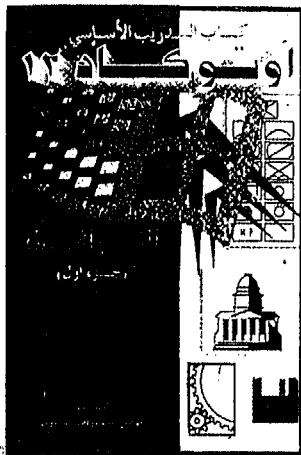
* الملفات التجميعية * وصل الأوامر * استخدام برمج ادارة ملحقات الحاسوب

* استخدام مفكرة الأوامر والمحترلات * اعادة التوجيه

ويشتمل الجزء الثاني على الأبواب الرئيسية التالية :

- ◆ حماية البيانات من الفيروسات باستخدام برنامج MS Anti-Virus
- ◆ النسخ الاحتياطي للملفات ومقارنتها واسترجاعها باستخدام برنامج MS Backup
- ◆ اعادة الملفات الملغية باستخدام برنامج MS Undelete
- ◆ مضاعفة حجم القرص المغناطيسي باستخدام برنامج DoubleSpace
- ◆ زيادة سرعة الحاسوب باستخدام برنامج SMARTDRV وبرنامج MS Defrag
- ◆ توفير حجم الذاكرة وتحقيق أقصى استفادة منها باستخدام برنامج MemMaker
- ◆ شرح الأوامر الجديدة في DOS 6





كتاب التدريب الأساسي

أوتوكاد ١٢

يشرح كتاب التدريب الأساسي أوتوكاد ١٢ الإصدار الأخير من برنامج AutoCAD الشهير الذي يعمل تحت بيئة التشغيل الرسمية «ويندوز»، ويحتوى على شرح وافٍ لمجموع الأوامر الالزامية لإلتحاز أي نوع من أنواع الرسومات فى مستوى واحد بين الحروف الأفقى والرأسي، ويستخدم الكتاب مفهوم خطورة.. خطورة، ولذلك يمكن أن تعتبره دليل تعليمي يصلح لمن يرغبون فى التعلم الذانى ومراكز التدريب المتخصصة. والكتاب مفيد لطلاب العلم فى الكليات الهندسية الذين يدرسون علم التصميم بمساعدة الحاسوب الآلى. والمهندسين الذين يعملون فى مجال الرسم الهندسى. والفنانين والرسامين والمصممين الذين يهمهم تنسيق الخطوط والدوائر والأشكال الهندسية.

المراجع الأساسي لقواعد البيانات

dBASE III PLUS

يشرح هذا الكتاب كيفية استخدام قاعدة البيانات dBASE III PLUS سواء من ناحية الأوامر واستخدام شاشات المساعدة ، أو من ناحية البرمجة والكتاب صريح بأسلوب تعليمي منظم يصلح للتدريس في الجامعات والمعاهد العملية ، وسهل للاستفادة منه العاملون في مجال الحاسوب والمتخصصون على حد سواء .

يقع الكتاب في جزئين . يشرح الجزء الأول كيفية بناء قواعد البيانات واعداد شاشات الادخال ، وترتيب الملفات واستعراض محتوياتها ، والاستفسار عنها ، واستخراج التقارير والملخصات .

ويشرح الجزء الثاني أساسيات البرمجة عموما ، وكيفية البرمجة بقاعدة البيانات . ولذلك فقد جاء شاملا كل ماتحتويه المادة ، وكل مايحتاج إليه العاملون في هذا المجال .



تعلم مايكروسوفت اكسل Excel 4.0 for Windows في يوم واحد



يشرح هذا الكتاب أساسيات التعامل مع برنامج الجداول الالكترونية Microsoft Excel 4.0 والكتاب يخاطب المتخصصين والذين لا يجدون الوقت الكافي لقراءة مراجع مطولة مثل كتابنا المراجع الأساسي لمستخدمي اكسل

وتتخصص فكرته في تقديم المادة بسهولة وسرعة تتناسب مع الهدف من استخدام Windows ، وهو السهولة والسرعة . والكتاب يبدأ من انشاء صفحة البيانات الالكترونية (Worksheet) وادخال بياناتها وتعديل محتوياتها ، وطباعتها باختيارات متعددة ، الى ادخال تحسينات عليها تساعد في اظهارها بشكل جيد مثل تغيير ابعاد الكتبة واستخدام البرواير والألوان والأساق المناسبة . وينتهي بالتعامل مع أكثر من صفحة بيانات وتبادل المعلومات بينها وانشاء علاقات دائمة بين المستندات . ثم يشرح كيفية تمثيل البيانات بالرسم البياني وكيفية حفظ الرسم وطباعته وحفظه واسترجاعه وادخال تحسينات عليه .

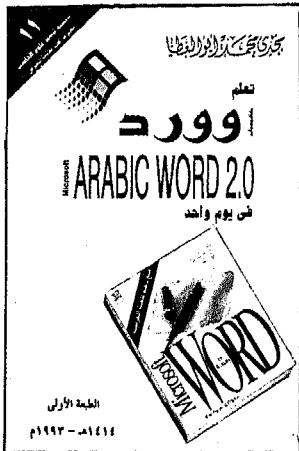
المرجع الأساسي لنظام التشغيل **Microsoft Windows 3.1** باعتماد اللغة العربية



يشرح هذا الكتاب بيئة التشغيل الرسومية الأكثر راحة وأسرع تفاعلًا والمعروفة باسم Microsoft Windows أو نوافذ مايكروسوفت ، ويشتمل على قواعد عامة للتعامل مع البرامج التي تأتي ضمن حزمة النوافذ ، أو المصممة للعمل تحت نظام Windows . و الكتاب عبارة عن دروس عملية تهتم سياسة "خطوة .. خطوة" في التعليم والتعلم لتساعدك في أن تعلم نفسك :

- التعامل مع نوافذ البرامج بفتحها وغلقها وتحجيمها وترصيصها وتكتييسها وترتيب أيقوناتها من سطح المكتب
- تجميع البرامج والمستندات في نوافذ جماعية واستخدام مدير البرامج لإنشاء النوافذ الجماعية ، وإعادة ترتيبها ، وتغيير اسمائها ، واللغتها ، وتصنيفها بأيقونات للبرامج .
- تبادل المعلومات بين البرامج المصممة للعمل تحت نظام Windows أو بينها وبين البرامج المصممة للعمل تحت نظام DOS .
- التعامل مع الملفات والأدلة باستخدام مدير الملفات بدلاً من استخدام الأوامر الصعبة من محوت DOS لفتحها ، أو إنشاؤها ، أو لاظهار محتوياتها ، أو لنقلها أو نسخها أو البحث عنها ، أو حذفها أو تغيير اسمائها .
- استخدام لوحة التحكم لتهيئة وضبط النظام ليوافق استخداماتك واحتاجاتك الخاصة عن طريق التوصيل إلى شبكات الطباعة ، وإعداد ملائدة الاتصالات ، وضبط الوقت والتاريخ ، وتركيب الخطوط ، واختيار الألوان المناسبة لتجميل منظر "المكتب" .
- استخدام مدير الطباعة لمعايير طوابير الطباعة وتغيير أولوياتها ، وجر و القاء المستندات للطباعة .
- البرامج المكتبية التي تأتي ضمن حزمة Windows والتي يطلق عليها Accessories وتشمل: الكتاب العربي ، الطرفية ، المفكرة العربية ، التقويم ، الفرشاة ، والساعة ، والحساب ،
- التعامل مع البرامج المصممة للعمل تحت DOS والتحكم فيها بتشغيلها داخل نوافذ أو على شاشة كاملة، وتحجيمها ، والانتقال من برنامج مفتوح لأخر ، وتبادل المعلومات بينها ، وكيفية إنشاء وتعديل ملف Windows PIF ليتم تشغيلها تلقائياً بنظام

لذلك فإن هذا الكتاب يعتبر بحق أول كتاب يشرح نظام نوافذ مايكروسوفت المدعم للغة العربية شرحاً وافياً ، بل هو الكتاب الوحيد الذي يشرح تعيين برامج النوافذ والبرامج المكتبية التي تأتي معها



تعلم مايكروسوفت وورد Arabic Word for Windows في يوم واحد

يشرح هذا الكتاب أساسيات التعامل مع برنامج Microsoft Arabic Word . والكتاب يخاطب المبتدئين والذين لا يجدون الوقت الكافي لقراءة مراجع مطولة مثل كتابنا المرجع الأساسي المستخدمي وورد . وتلخص فكرته في تقديم المادة بسهولة وسرعة تتناسب مع الهدف من استخدام Windows ، وهو السهولة والسرعة . والكتاب يبدأ من إنشاء المستند وتعديل محتوياته ، وطباعته باختيارات متعددة ، إلى إدخال تحسينات عليه تساعد في اظهاره بشكل جيد مثل تغيير أبسط الكتابة واستخدام البرواير والألوان والأنساق المناسبة . وينتهي بشرح مفاهيم متقدمة مثل استخدام الأنماط وإنشاء الجداول والدمج البريدي

تعرف على الحاسوب الشخصي

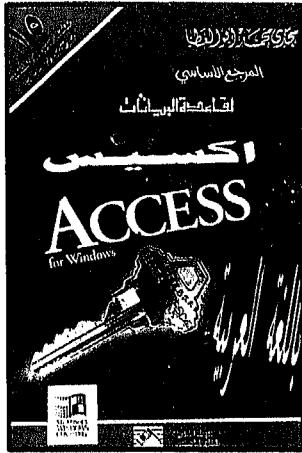
يشرح هذا الكتاب كل ما يهم القارئ عن تكنولوجيا الحاسوب الشخصي . وبرامجها وأجهزتها ويحتوى على مقدمة لنظام تشغيل الحاسوب بصنفه عامه ونظام التشغيل MS-DOS 5.

يصفه خاصة ، ومقدمة لنظام التشغيل Microsoft Windows 3.11 المدعم للغة العربية . ويشتمل على دروس عملية لمن يستخدمون الحاسوب لأول مرة . ويشرح لمن يخططون لشراء حاسب شخصي أو لتطوير حاسباتهم كيفية ترشيد قرار شراء الحاسب وملحقاته . وللمهتمين بتعريف الحاسوب كيف يختارون تعریب حاسباتهم . وأخيراً مقدمة عن تشبيك الحاسوبات .

- لذلك فإننا نعتبر أن دراسة هذا الكتاب ليست ضرورية للمبتدئين فقط ، ولكن أيضاً لذئات كثيرة تشمل :
- * رجال الأعمال والمديرون المهتمون بمكتنة أعمالهم ، والذين يخططون لشراء حاسوبات شخصية
 - * طلاب المدارس والمعلمون في جميع مراحل التعليم الذين يهجون سياسة "خطوة .. خطوة " في التعليم
 - * والتعلم
 - * الأشخاص الذين يستخدمون الحاسب إلا أن معلوماتهم عن الأجهزة والبرامج غير كافية



المراجع الأساسي لقواعد البيانات Access



يصلح كتاب المرجع الأساسي لقواعد البيانات Access 2 لكل من المدربين والمتدربين والمعاهد المتخصصة ، لأنه يعتمد سياسة خطوة خطوة في التعليم والتعلم ، من خلال تمارين عملية مسجلة على فرق مغناطيسي مرافق مع الكتاب ويعاد بجانبها يتكون كل تمرين من خطوات مسلسلة تشتمل على الإجراءات المطلوبة للوصول إلى المدلف ، وفي حالة الضرورة تظهر الشاشات التي توضح نتيجة الإجراء المتخذ داخل التمرين ، والمدلف من ذلك يحب الوقوع في أي خطأ أثناء تنفيذ الخطوات التالية.

يخاطب هذا الكتاب المبتدئين ومن يستخدمون برنامج Access 2 ، وإذا كانت لك خبرة سابقة بالتعامل مع البرنامج ، فإن الكتاب سيسعى يدك على مفاهيم متقدمة وعلى مواضع قوة البرنامج التي تبحث عنها والتي يجعلك تفضل البرنامج على غيره من برامج قواعد البيانات.

يدأ الكتاب بإعطاء خلفية ضرورية يجب أن تفهمها جيدا قبل أن تبدأ استخدام "آكسس" ثم يشرح الوظائف الأساسية لنظام إدارة قواعد البيانات والتي يحتاجها معظم الناس ومنها :

- إنشاء قاعدة البيانات وتعديلها.
- إظهار البيانات والتحكم فيها.
- إنشاء ملف الاستعلام واستخدامه.
- تصميم التقارير وطباعتها.
- تصميم النماذج واستخدامها.
- ربط الملفات.
- التعامل مع برامج أخرى.
- استخدام الماكروز.
- استخدام آكسس داخل شبكة اتصالات.

