مقدمة عامة على لغة C

*** مميزات لغة C** : - تتميز لغة C بمجموعة من المزايا مثل :-

- **لغة عامه** : أى تصلح لعمل برامج قواعد البيانات والرسومات والحسابات ونظم التشغيل .
- **لغة تركيبية** Structured Language : البرنامج المكتوب بلغة C عبارة عن دالة رئيسية تنادى مجموعة من الدوال الأخرى . وكل داله مجموعه من الأوامر .
- **تتعامل على مستوى (البت) Bit manipulation** : - حيث تستطيع أن تقرأ وتكتب وتغير وتقوم بعمليات على مستوى الـ Bit . حيث أن Bit هى اصغر وحدة لقياس المعلومات داخل الكمبيوتر وهى جزء من ثمانية أجزاء تعادل فى مجموعها حرف واحد byte .
- **لغة متنقلة Portable** : أى يمكن للبرنامج المكتوب بلغة C أن يعمل مع أكثر من جهاز مثل IBM و Apple .
- **لغة سريعة** : - لأن أدوات اللغة تتعامل مع الآلة مما يختصر وقت التنفيذ .
- **لغة قياسية** : معظم مترجمات اللغة تتوافق مع اللغة القياسية C ANSI

The Basic Structure of c Program

قواعد بناء البرنامج

- البرنامج التالى يمثل أبسط تركيب لبرنامج مكتوب بلغة C

```
# include < stdio .h >
main ( )
{
printf ( " hello egypt " ) ;
}
```

ملاحظات هامة

- يبدأ البرنامج بالعبارة # include < .....h > وبين العلامتين اسم ملف التوجيه الخاص بالدوال المستخدمة فى البرنامج – يمكن كتابة أكثر من include .
- يتكون البرنامج من دالة رئيسية ( ) main وتبدأ بالقوس { وتنتهي بالقوس } .
- جميع كلمات ودوال اللغة تكتب بالحروف الصغيرة .
- تنتهي كل عبارة بفاصله منقوطة (;)
- يجوز كتابة أى ملاحظات أو تعليقات خاصة بالبرنامج بوصفها بين العلامين /* */ لأى عدد من السطور

∴ جسم الدالة يوضع ما بين الأقواس { }

أنواع البيانات Data Type

*** البيانات التى تتعامل معها إما أرقام أو حروف أو كلمات :-**

- والأرقام يمكن أن تكون صحيحه (أى ليس بها علامة عشرية) integer أو حقيقية (أى بها علامة عشرية) float .
- والحروف يمكن أن تكون حرف واحد أو أكثر من حرف

*** الجدول التالى يوضح هذه الأنواع وكذلك عدد البايت byte التى يشغلها :-**

نوع المتغير	طوله بالبايت	المدى المسموح
حرف (char)	1	حرف أو رمز واحد
صحيح قصير (int)	2	-٣٢٧٦٨ إلى ٣٢٧٦٨
صحيح طويل (long)	4	-٢٠١٤٧٠٤٨٣٠٦٤٨ إلى ٢٠١٤٧٠٤٨٣٠٦٤٨
حقيقى (float)	4	e- 38 إلى e + 38
حقيقى مضاعف (double)	8	e -308 إلى e+308

وفيما يلى المقصود بكل هذه الأنواع : -

- **متغير من نوع حرف** : أى متغير يصلح لتخزين حرف فقط .
- **متغير من نوع صحيح** : أى متغير يصلح لتخزين رقم صحيح (ليس به علامة عشرية) .
- **متغير من نوع صحيح ولكن طويل(Long)** : أى يستطيع أى يخزن رقم صحيح ضعف المتغير الصحيح العادى ويستعمل هذا النوع إذا كانت الأرقام التى تتعامل معها أكبر من المساحة المخصصة وإلا سنحصل على نتائج خاطئة بالرغم من إن البرنامج سليم
- **متغير حقيقى** : أى متغير يصلح لتخزين رقم حقيقى يقبل الكسور العشرية مثل 5 .33
- **متغير حقيقى مضاعف** : أى يستطيع أن يخزن رقم حقيقى ضعف المتغير الحقيقى العادى .

*** تسميته المتغير** : - يخضع اسم المتغير لشروط معينه :-

- يجب أن يبدأ المتغير بحرف ثم يكمل المتغير بعد ذلك حروف أو أرقام .
- يفرق المترجم بين الحروف الصغيرة والكبيرة فالمتغير HP يختلف عن المتغير hp فإذا استعملا فى البرنامج يعتبرهما البرنامج متغيرين
- يجب ألا يكون المتغير بإسم كلمه من الكلمات المحجوزة .

*** الأعلان عن المتغيرات : -**

- فى لغة الـ C لابد الإعلان عن المتغيرات Variables فى بداية البرنامج إما إذا كنت تستخدم مترجم لغة C ++ يتم الإعلان عن المتغيرات فى إى مكان بالبرنامج

```
int a ;
float ;
```

*** المؤثرات** operators

المؤثرات هى الرموز التى تربط بين المتغيرات والثوابت لإنشاء علامة أو معادلة تختلف أنواع المؤثرات باختلاف وظيفة كل مؤثر .

١- المؤثرات الحسابية arithmetic operators

+ علامات الجمع addition

- علامات الطرح Subtraction

* علامات الضرب multiplication

/ علامات القسمة division

وتستخدم مع المتغيرات والثوابت الرقمية

٢- مؤثرات المقارنة Relational operators:- وتستخدم لمقارنة قيمتين :

المؤثر	الرمز	مثال	النتيجة
أكبر من greater than	>	10>8	1
أصفر من less than	<	10<8	٠
يساوى equal to	==	١٠==٨	0
لا يساوى not equal to	= !	١٠!=٨	1
أقل من أو يساوى less than or equal to	<=	١٠ < = 8	0
أكبر من أو يساوى greater than or equal to	>=	10 > = 8	٠

٣- المؤثرات المنطقية Logical operator

المؤثر	الرمز	مثال	النتيجة
و AND	&&	10 > 8 && 9 > 7	1
أو OR	||	1 0 < 8 || 7 < 8	١
لا NOT	!	! (10 = = 8)	1

٤- مؤثرات التخصيص Assignment Operators

وهى مؤثرات تخزين قيمة فى متغير فمثلا إذا كانت قيمة ٦ = ٩

المؤثر	النتيجة	الطريقة الحديثة	التخصيص التقليدى
+ = addition assignment operators	11	A + = 5	A = a +5
Subtraition assignment opertors	1	A - = 5	A = a – 5
Multipication assibnment operators	30	A * = 5	A = a + 5
Division assignment operators	2	A / = 3	A = a / 3

٥- مؤثرات الزيادة والنقصان increment& Decrement

A = a + 1	A ++	7	مؤثر زيادة واحد
A = a – 1	A --	5	مؤثر نقصان واحد

٦- مؤثر باقى خارج القسمة %

يستخدم لمعرفة باقى القسمه (لتحديد هل الأرقام الموجودة فى المتغير زوجية أو فردية فمثلا إذا كانت قيمة a = 5 وكتب C = a % 2 يكون باقى الرقم 2 / 5 = 1

دوال الإدخال والإخراج

* دالة الطباعة على الشاشة printf() *

ملاحظات هامه : كل دالة مرتبطة بملف توجيه معين حيث يستدعى هذا الملف فى أول البرنامج بالعبارة # include فمثلا الدالة printf () معرفة بالملف stdio.h وتكتب العبارة # include < stdio.h > فى أول البرنامج حتى يتعرف المترجم على الدالة وهكذا مع باقى الدوال

- تستخدم دالة الطباعة printf ( ) لطباعة البيانات بجميع أنواعها (String , char , float , int) على الشاشة فقط .
- ونأخذ دالة الطباعة عدة صور وكذلك معاملات وأكواد تحدد شكل المخرجات

مثال ١

```
printf ( " welcome with compuscience " )
```

هنا يتم طباعة ما بين علامتى التنصيص " "

مثال ٢

```
printf ( " \n welcome \n with \n compuscience " ) ;
```

فى هذا المثال : الكود ١٠٠ معناه new line أى سطر جديد وعندما يجد المترجم ١٠٠ يترجمها إلى سطر جديد ويكون الناتج

```
welcome
with
compusciene
```

وفيما يلى الاكواد المستخدمة مع الدالة printf( )

الكود	الاستخدام	المثال

\n	الانتقال السطر الجديد new line	printf (" \n ")
\t	نقل المؤشر بعد ٨ مسافات (Tap)	print f (" \t ")
\b	إرجاع المؤشر مسافة خلفية Backspace	print f (" \b ")
\xdd	طباعة الحرف المناظر للكود المكتوب بالنظام السادس عشر hexadecimal	printf (" x41 ") : a والنتيجة
\ddd	طباعة الحرف المناظر للكود المكتوب بالنظام الثمانى octol (each d represents a digit)	printf (" \101") : a النتيجة
\"	طباعة علامة التنصيص double quate	printf (" \"")
\a	إخراج صوت الصافرة (يبب)	printf (" \a ")

*** أوجد ناتج تنفيذ البرنامج التالى :**

```
/ * program name 1 * /
# include < stdio.h >
main ( )
{
printf ( " \n this text display in new line " ) ;
printf ( " \n word1 \t lab1 \t tab2 " ) ;
printf ( " \n bell \a bell \ b " ) ;
printf ( " \n this line display quotations \ " " ) ;
printf ( " \n " ) ;
}
```

*** طباعة قيم المتغيرات على الشاشة :** لطباعة القيم الموجودة بالمتغيرات تستخدم أكواد معينة لتحدد نوع البيانات المراد طباعتها بالدالة print f ()

مثال

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

- فى هذا المثال عندما يقابل مترجم اللغة العلامة % ينظر إلى الحرف التالى لهذه العلامة . ويعتبر هذا الحرف توصيف لقيمة موجودة بعد العلامة وكل حرف يحدد تنوع معين من البيانات .

والجدول التالى يوضح أكواد طباعة أنواع البيانات :

الكود	الاستخدام	مثال

%d	توصيف لمتغير أو ثابت رقمى صحيح (Signed decimal integer) int	printf (" % d " , - 10)
%f	توصيف لمتغير أو ثابت رقمى حقيقى (floating point) float	printf (" % p " , 507)
%c	توصيف لمتغير أو ثابت (حرف واحد) char Single character	printf (" % c " , " a ")
%s	توصيف لعبارة حرفية حرف أو أكثر String	printf (" % s " , " is ")
%u	توصيف لمتغير أو ثابت رقمى صحيح بدون إشارة) (unsigned decimal integer)	printf (" % u " , " 1٠)
%x	توصيف لمتغير أو ثابت بالنظام السادس عشر hex	printf (" % x " , af)
%o	توصيف لمتغير أو ثابت بالنظام الثمانى Qctal	printf (" % o " , 67)

*** أوجد ناتج تنفيذ البرنامج التالى**

```
/ * program name 2 * /
# include < stdio.h >
main ( )
{
int a , b , c ;
float f ;
long t ;
char ch = 'y ' ;
char name [10] = " aly " ;
a = 5 ;
b = 10 ;
c = a + b ;
printf (" \n c = % d " , c ) ;
printf ( "\n f = % f " , f ) ;
printf (" \n name = % s " , name ) ;
printf ( " \n  ch = % c " , ch) ;
printf ( "\n t = % ld " , t ) ;
}
```

*** ملاحظات على الحل ***

- يشمل البرنامج السطر رقم ١ للتعليق أو الملاحظة
- فى السطر ٢ يشمل على الجملة # include < stdio.h > وتستخدم لتحميل ملف التوجيه stdio.h الذى يحتوى على تعريف الدالة printf ( )

- السطر رقم ٣ تبدأ الدالة الرئيسية ( ) main ثم السطر ٤ تبدأ الدالة الرئيسية بالقوس {
- فى السطر رقم 5 , 6 , 7 إعلان عن المتغيرات
- فى السطر رقم ٨ الإعلان عن المتغير ch من نوع حرف (char) وإعطائه القيمة y
- فى السطر رقم ٩ الإعلان عن المتغير name لتخزين عبارة حرفية وإعطائه القيمة الابتدائية كلمة Aly
- فى السطر رقم ١٠ ، ١١ ، ١٢ لإعطاء قيم للمتغيرات A , B وقيمة C
- ثم طباعة المتغيرات ثم تنتهى الدالة الرئيسية بالقوس } وبالتالى ينتهى البرنامج

ملاحظات : الصورة 3F % : يعنى طباعة ثلاث أرقام بعد العلامة العشرية فمثلا الرقم 534.6735 يظهر بالصورة 534.674

* دالة الإدخال العامة Scanf ( )*

هى دالة الإدخال الرئيسية التى تسمح بإدخال جميع أنواع البيانات وهى تأخذ نفس المعاملات التى تأخذها الدالة Print f ( )

مثال :

```
# include < stdio.h >
main ( )
{
int a , b , c ;
float r , s , t ;
char name [10];
printf ( " \n \n enter your name : " ) ;
scanf ( " % s " , name ) ;
printf ( " a = " ) ;
scanf ( " % d " , & a ) ;
printf ( “ b = " ) ;
scanf ( % d " , & b ) ;
printf ( " r = " ) ;
                                    scanf ( " % f " , &r ) ;
printf ( " s= " ) ;
scanf ( " % f " & s ) ;
printf ( " \n welcome % s " , name ) ;
printf ( " \n \n c = a + b = % d " , a + b ) ;
printf ( " \n \n t = r + s = % d " , r + s ) ;
}
```

ملاحظات على الحل

- يتم الإعلان عن المتغيرات a , b , c , r , s , t , name
- تطبع الدالة printf() الرسالة enter your name
- تستقبل الدالة ( ) scan f العبارة الحرفية التى يدخلها المستخدم ونصـفها فـى المتغيـر name ٠٠٠٠ كذلك المتغيرات الأخرى
- تستقبل الدالة ( ) scan f فى سطر (&a , " % d “) ( ) scan f قيمة صحيحه وتخزنها فى المتغير a

 # ماذا يعنى المؤشر & #

&a : تعنى تخزين القيمة الصحيحة فى المكان المخزن عنوانه فى المتغير a بمعنى أن a يشير إلى عنوان المكان الذى تخزن فيه القيمة حيث العلامة & تجعل المتغير يشير إلى عنوان المكان

*** الناتج :-**

```
enter your name : ahmed
a = 5
b = 1٠
r = 2٠
s = 3٠
```

```
welcome ahmed
c = a + b = 15
t = r + s = 50
```

تمرين : برنامج لإيجاد مساحة الدائرة

```
# include < stdio .h >
/ * program to calculate erea of circle * /
main  ( )                                   / * function heading */
{
float redius, area ;                        /* variable declarations */
printf ( " redius = ? " ) ;                 /* output statement  */
scanf ( " % f " , redius ) ;                /* input statement */
area = 3.14159 * redius * redius ;          /* assignment statement */
printf  ( " area = % f " , area ) ;         /* output statement */
}
```

*** دوال إدخال حرف *** هناك دوال أخرى تتعامل مع أنواع خاصة من البيانات كـالحروف والعبـارات الحرفية وهى :

getchar ( ) , getche ( ) , getch ( )

١ – الدالة getchar ( ) : - (ملف توجيه stdio .h)

تستخدم لإدخال حرف واحد ويظهر الحرف على الشاشة بعد الكتابة ولا تسمح بالانتقال إلى الأمر التـالى إلا إذا ضغط المستخدم مفتاح الإدخال enter

مثال

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

٢- الدالة getche ( ) **:** - (ملف التوجيه conio .h)

تستخدم لإدخال حرف واحد ويظهـر هذا الحرف على الشاشة ولكنهـا تختـلف عـن الدالـة getchar ( ) فى أنها لا تحتاج إلى الضغط على مفتاح الإدخال enter للانتقال للسطر التالى وتعمل هذه الدالة بطريقة مشابهة .

مثال

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

٣ – الدالة getch ( ) : (ملف التوجيه conio.h)

تستخدم لإدخال حرف واحد ولكن تختلف عن الدالتين السابقتين فى أن هذا الحرف لا يظهر على الشاشة وكذلك فى أنها لا تحتاج إلى الضغط على مفتاح الإدخال enter للانتقال إلى السطر التالى

مثال

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

*** دالة طباعة حرف واحد putchar ( ) : (** ملف توجيه stdio.h)

- تستخدم لطباعة حرف واحد على الشاشة
- فمثلا putchar (' a ') تطبع على الشاشة الحرف a كما هو

*** دالة إدخال عبارة حرفية gets ( ) *** ملف التوجيه stdio.h

- تستخدم الدالة gets ( ) فى إدخال عبارة حرفية string

مثال

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

فى هذا المثال تخزن الدالة gets ( ) العبارة الحرفية فى المتغير Name

*** دالة طباعة عبارة حرفية puts ( ) *** ملف التوجيه stdio.h

- تستخدم لطباعة عبارة حرفية string حيث تطبع بدون توصيف شكل المخرجات

مثال :

```
char name[10] = " ahmed "
puts (name) ;
puts ( "mohammed " ) ;
```

عند تنفيذ البرنامج نحصل على النتيجة

```
ahmed
mohammed
```

ملاحظة : الإعلان char name[10] معناه أن المتغير name من نوع حرفى ويصلح لتخزين كلمة أقصى عدد حروف لها هو ١٠ حروف

*** بعض الدوال المكتبة**

الدالة	النوع	الغرض	ملف include
abs (i)	صحيح	تعيد قيمة i المطلقة	stdlib.h
log (d)	مزدوج	تعيد لوغاريتم طبيعيا لـ d	math.h
pow (d1 , d2)	مزدوج	تعيد d1 مرفوعة إلى القوة d1	math.h
labs (e)	صحيح طويل	تعيد قيمة مطلقة لـ l	math.h
sin (d)	مزدوج	تعيد جيب d	math.h

sqrt (d)	مزدوج	تعيد الجزر التربيعى لـــ d	math.h
strcpy (s1,s2)	حرف	تنسخ السلسلة s2 فى السلسلة s1	string.h

أوامر التحكم

يتم تنفيذ السطور بين أقواس الدالة الرئيسية main بترتيب كتابتها . وعند وجود سطور متشابهة يتم العمل بأسلوب أوامر التحكم كالتالى : -

التكرار (الدورات) التفريغ

أولا التكرار Loop

١- الدوارة for : تستخدم لتكرار تنفيذ عملية عدد محدد من المرات

الصورة العامة : For (initial – value ; condition ; increment) ; statement ;

حيث :

initial – value : هى القيمة الابتدائية

condition : هو شرط إنهاء التكرار

increment : هى قيمة الزيادة الدورية

مثال : برنامج يطبع الأرقام من صفر إلى ٩

```
# include < stdio.h >
main ( )
{
int i ;
for ( i = 0 ; i < 10 ; i ++ )
printf ( " \ n i = % d " , i ) ;
}
```

*** تنفيذ أكثر من جملة مع For** : لتنفيذ أكثر من جملة لعدد محدد من المرات يجب وضع القوس { فى بداية البلوك المراد تكراره ووضع القوس { فى نهاية البلوك .

مثال : حساب متوسط قائمة من الأعداد

```
# include < stdio.h >
/ * calculate the average of n numbers * /
main ( )
{
int n , count ;
float x , average , sum = 0 ;
/ * initialize and read in a value for n * /
```

```
printf ( " how many numbers ? " ) ;
scanf ( " % d " , & n ) ;
/ * read in the numbers * /
for ( count = 1 ; count < = n ; count ++ )
{
printf ( " x = " ) ;
scanf ( " % f " ; & x ) ;
sum + = x ;
}
/ * calculate the average and write out the answer * /
average = sum / n ;
printf ( " \ n the average is % f \n " , average ) ;
}
```

*** الدوارات المتداخلة باستخدام For**

- الدوارات المتداخلة عبارة عن دوارة كبيرة تشتمل بداخلها على دوارة أو أكثر
- بمعنى أن مجموعة التعليمات بالـــ loop الداخلى يتم تكرار تنفيذها طالما لم ينته العداد فإذا أنتهـــى ينتقل التنفيذ إلى الدوارة الخارجية ويتم تكرار التعليمات فى الـــ Loop الخارجى حتى ينتهى .

مثال : طباعة جدول الضرب من أول ١ × ١ إلى ١٢ × ١٢

```
# include < stdio.h >
main ( )
{
int i , j  ;
        for ( i = 1 ; i < = 12 ; i ++ )
        {
        printf ( " in ") ;
                for ( j = i ; j < = 12 ; j ++ )
                {
                printf ( " i * j = % d " , i * j ) ;
                }
        }
}
```

- **الدوارة اللانهائية باستخدام For** : معناها تكرار تنفيذ الجملة بدون شرط ولا يتوقف التنفيذ حتى يضغط المستخدم CTRL + C وتأخذ الدوارة اللانهائية الصورة (;;) For

٢ – الدوارة while: تستخدم الدوارة while لتكرار تنفيذ جملة أو مجموعة جمل عدد من المـــرات غير معلوم العدد ويتوقف هذا على شرط موجود بالدوارة

الصورة العامة :

While (condition)
Statement ;
	While (condition)
	{
	statement 1 ;
	statement 2 ;
	}

- يتم تكرار مجموعة الأوامر بين الأقواس { } عدد من المرات يتوقف على الشرط بين الأقواس ( )
- عندما يصبح الشرط . False يتوقف تنفيذ هذه الأوامر ويستمر أداء باقى البرنامج .
- التكرار يكون صفرا حينما يكون الشرط False من البداية

تمرين : تعديل برنامج حساب متوسط قائمة من الأعداد باستخدام While

```
# include < stdio.h >
main ( )
{
int n , count = 1  ;
float x , average , sum = 0 ;
printf ( " how many number ? " ) ;
scanf ( " % d , & n ) ;
while ( count < = n ) ;
{
printf ( " x = ") ;
scanf ( " % f " , & x ) ;
sum + = x ;
count ++;
}
average = sum / n ;
printf ( " \ n the average is % f " ; average ) ;
}
```

*** ملاحظات على For و While ***

- الـــ For دوارة عددية حيث تعتمد على العداد وينتهى التكرار فيها بانتهاء عدد مرات التكرار
- أما الدوارة while فدوارة شرطية أى تعتمد على الشرط الذى يلى الأمر while حيث تتكرار الجمل التى تليها طالما كان الشرط صحيحا

٣- الدوارة Do ... while : تستخدم do while لتكرار تنفيذ جملة أو مجموعة جمل أكثر من مرة بناء على شرط معين . أى يتم تنفيذ الجمل التالية لـــ Do ثم تختبر الشرط فإذا كان صحيحا تعيد التنفيذ وإلا توقف التكرار

إذن يتم التنفيذ مرة واحد على الأقل ثم يتم اختبار الشرط المنطقى ( ) فإذا كان True ينفذ مرة أخرى وإذا كان False يتوقف التكرار ويستمر تنفيذ البرنامج الأصلى

تمرين : إيجاد مجموع الأرقام من ٠ إلى ٩

```
# include < stdio.h >
main ( )
{
int count = 0  ;
int total = 0  ;
do
{
total + = count ;
```

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

ثانياً : التفريـغ : لتنفيذ سطر أو عدة سطور طبقا لشرط معين .أى تفريغ بعنى تغير مسار البرنـــامج . والتفريغ إما يكون مشروط أو غير مشروط

١) التفريغ المشروط

*** جملة الشرط** IF : تستخدم كلمة IF لتنفيذ جملة أو أكثر حسب شرط معين (اختبار منطقى)

الصورة العامة

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

معناه إذا تحقق الشرط (condition) نفذ الجملة التالية أما إذا لم يتحقق الشرط فلا تنفذ هـــذه الجملـــة وانتقل إلى التى تليها

ملاحظة إذا كان هناك أكثر من جملة تريد تنفيذها مع if لابد من فتح قوس { قبل مجموعة الجمل والقوس { فى أخر الجمل كما يلى :

```
if ( condition )
{
statement 1 ;
statement 2 ;
}
```

*** جملة if الشرطية المتداخلة :** يمكن أن تتداخل جمل if فتأخذ الشكل التالى :

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

وهذا معناه إذا تحقق الشرط الأول انظر إلى الشرط الثانى .... وهكذا

*** الجملة الشرطية if…else :** تستخدم لتنفيذ أحد اختيارين وتأخذ الصورة التالية :

```
if ( condition )
  {
   statement 1
  }
else
  {
  statement 2
  }
```

ومعناها إذا كان الشرط (condition) صحيح true نفذ الجملة الأولى (statement 1) وإلا نفذ الجملة الثانية (statement 2) أى الجملة الشرطية if.... else تستخدم لتحديد اختبار واحد من اختيارين ولا يمكن تنفيذ الاختيارين معا .

*** الجملة الشرطية if .... else if :** لتنفيذ خيار من مجموعة خيارات كمقارنة رقمين

الطريقة الأولى : باستخدام ثلاث جمل IF وفى كل جملة نضع أحد الشروط الثلاثة كما يلى : -

```
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 " ) ;
```

وهذه طريقة تستهلك وقتا فى اختبار جمل الشرط

الطريقة الثانية

```
if ( condition )
   statement 1 ;
else if ( condition )
   statement 2 ;
else if  ( condition )
  statement 3 ;
```

وتكون صيغة المثال السابق كآلاتى :

```
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 " ) ;
```

برنلمج : اشرح وظيفة البرنامج مع إيجاد الناتج إذا كانت القيم ١٣و٧على التوالى :-

```
# include < stdio.h >
main ( )
{
float num1 , num2 ;
char op ;
while (1)
 {
 printf ( " type number , operator , number \ n " ) ;
```

```
 scanf ( " % f % c % f ) , & num 1 , & op , & num 2  ) ;
 if ( op = = ' + ' )
    printf ( " = % f " , num1 + num2 ;
 else if ( op = = ' – ' )
    printf ( " = % f " , num1 – num2 ;
else if ( op = = ' * ' )
   printf ( " % f " , num1 * num2 ;
else if ( op = = ' / ' )
  printf (" = % f " , num1 / num2 ;
  printf ( " \ n \ n " ) ;
 }
}
```

٢ - التفريع Switch …. case

تتسبب عبارة Switch فى اختبار مجموعة من عبارات معينة من عديد من المجموعات المتاحة للاستخدام . ويعتمد الاختبار على القيمة الحالية لتعيير موجود داخل عبارة Switch

تمرين : تعديل برنامج الآلة الحاسبة السابعة باستخدام Switch ….. case

```
#include < stdio.h>
main ( )                                   /* void main(void)  */
{
float num1,num2;
char ch,op;
do
{
printf( "\in type num1 op num2: " ) ;
scanf( "%f %c %f ", &num1,&op,&num2);
switch (op)
{
case "+";
printf ( "sum = %f ", num1+num2);
break;
case "-";
printf ( "sub = %f ", num1-num2);
break;
```

```
case “*”;
printf ( "mul = %f ", num1*num2);
break;
case “/”;
printf ( "div = %f ", num1/num2);
break;
default:
printf\n unknowen operator .”);
}
printf(“\n Again (y/n): “);
}
while (( ch=getch())== ‘y’ ):
}
```

ملاحظة : من التطبيقات المشهودة لاستخدام التفريع case ….. switch هو استخدامه فـــى قـــوائم الاختيارات ( ) menu

تستخدم عبارة break فى انهاء دورات او الخروج من switch

٣) التفريغ غير المشروط goto

معناه الانتقال الى مكان محدد داخل البرنامج بدون شرك

```
#include <stdio.h>
main( )
{
char ss;
ss:
      printf(“\t ALLAH”);
      goto ss;
}
```

ملاحظات : - لا ننصح باستخدام جملة goto ويفضل استخدام الدوال لتغيير مسار تنفيذ البرنامج

- للخروج من البرنامج السابق أضغط CTRL+C

** دوال تحسين المدخلات والمخرجات **

(١) داله مسح الشاشة clrscr( ) : ملف التوجيه conio.h

- تستخدم لمسح الشاشة ووضع المؤشر فى أول عمود من الصف الأول على الشاشة
- الشكل العام clrscr( )

(٢) داله تغيير ووضع المؤشر gotoxy( ) : ملف التوجيه conio

- تستخدم لوضع الموشر فى العمود x من الصف y
- الشكل العام gotoxy( )

مثال : الانتقال بالمؤشر الى العمود 30 من الصف العاشر أكتب gotoxy(30,10)

(٣) داله تغيير لون الكتابة ( )textcolor : ملف التوجيه conio.h

- تستخدم لتغيير لون الكتابة التى ستطبع بعد الداله
- الصورة العامه

textcolor(color no)

أو

textcolor(color name)

حيث يتم تحديد اللون إما برقم اللون أو باسمه . ولا بد من كتابة اسم اللون بالحرف الكبيرة فقط

الجدول التالى لوضع اكواد الألوان وأسماها

اللو ن	رقم اللو ن	اسم اللو ن
اسود	0	BLACK
ازرق	١	BLUE
أخضر	٢	GRCEN
سماوي	٣	CYAN
أحمر	٤	RED
بنفسجي	٥	MAGENTA
بني	٦	BROWN
رمادي فاتح	7	LIGHTGRAY
رمادي غامق	٨	DARKGRAY
ازرق فاتح	٩	LIGHBLUE
أخضر فاتح	١٠	LIGHTGREEN
سماني فاتح	١١	LIGHTRED
أحمر فاتح	١٢	LIGHTRED

بنفسجي فاتح	١٣	LIGHTMAGENTA
أصفر	١٤	YELLOW
أبيض	١٥	WHITG

مسائل :

١- اكتب برنامج لإيجاد جملة مبلغ ما اودع فى بنك لمدة ٢٠ سنة بفائدة ٦% لمبلغ ١٠٠ $ والمعادلة كما يلى :

f=p(1+i)n

حيث: n عدد السنوا ت p المبلغ I معدل الفائدة

*** * البرنامج * ***

```
#include <stdio.h>
#include <math.h>
main()
{
float p,r,n,i,f;
/* read input data */
printf( “please enter a value for the princepal (p) : “ );
scanf(“%f” k &p);
printf(“please enter a value for the interest rate( r) : “ );
scanf(“%f”, &r);
printf(“please enter a value for the number or year ( n) : “ );
scanf(“%f”, &n);
/*calculate i , then f */
i=r/100
f=p*pow((1+i),n);
/* write output */
```

```
printf( "\n the final value (f) is : %2f \n",f);
}
```

٢- اكتب برنامج لإيجاد الجذور الحقيقية لمعادلة من الدرجة الثانية بمعلومية $ax^2 + bx + c = 0$ وباستخدام الصيغة

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

** البرنامج **

```
# include <stdio.h>
# include < math.h>
/* real roots of a quadratic equation */
main( )
{
float a,b,c,d,x1,x2;
/* read input data */
printf("a= " );
scanf("%f",&a);
printf("b= " );
scanf("%f",&b);
printf("c= " );
scanf("%f",&c);
/* carry out the calculations */
d=sqrt(b*b – 4*a*c);
```

```
x1=(-b+d)/(2*a);
x2=(-b-d)/(2*a);
/* write output */
printf("x1= ",%e      x2=%e ",x1,x2);
}
```

ملاحظة :

%e : تستخدم للعد الحقيقى بالصور الأسية

٣- اكتب برنامج لتقويم كثيرة الحدود

$$y = (\frac{x-1}{x}) + \frac{1}{2}(\frac{x-1}{x})^2 + \frac{1}{3}(\frac{x-1}{x})^3 + \frac{1}{4}(\frac{x-1}{x})^4 + \frac{1}{5}(\frac{x-1}{x})^5$$

***تمهيد: - لتبسيط البرنامج يتم وضع المتغير** $u = (\frac{x-1}{x})$

$$y = + \frac{u^2}{2} + \frac{u^3}{3} + \frac{u^4}{4} + \frac{u^5}{5}$$

**** البرنامج ****

```
# include<stdio.h>
# include<math.h>
main( )
{
float u,x,y;
/* read input data */
printf (" x = " );
scanf(" %f ",&x );
/* carry out the calculations */
u= (x –1)/x;
y=u+pow(u,2.)/2+pow(u,3.)/3+poe(u,4.)/4+pow(u,5.)/5;
```

```
printf("x= ",%f       y ="%f ", x,y );
}
```

٤- برنامج لإدخال كلمة سر

```
# include<stdio.h>
# include<conio.h>
main( )
{
char pass[10];
do
      {
            printf("\n enter password: " );
            scanf("%s",pass);
      }
while(strcmp(pass,"dahe")!=0);
}
```

ملاحظات:

- هنا كلمة السر سوف تظهر أثناء الكتابة
- الدالة ( )strcmp : تقوم بمقارنه متغيرين من نوع عبارة حرفية string فإذا كـــان المتغيـــرين متطابقين كان الفرق بينهما صفر

تعديل لبرنامج كلمة السر:-
(عدم ظهور كلمة السر التى يكتبها المستخدم على الشاشة)

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

```
main( )
{
chat ch;
char pass[10];
do
        {
        textcolor(WHITE);
        textbackground(BLUE);
        cprintf("\n enter password: " );
        textbackgrounf(WHITE);
        cscanf("%s",pass);
        }
while(strcmp(pass,"dahe")!=0);
}
```

اكتب برنامج:
لطباعة عبارة حرفية تم ادخالها مع بيان عدد حروفها وعدد الكلمات

```
/* count characters and word in a phrase عبارة typed in */
# include <stdio.h>
main( )
{
int charcnt=0;
int wordcnt=0;
char ch;
printf(" type in a phrase : \n ");
/* read characters and quit loop on [ return ] */
while((ch=getche())!='\r');
{
charcnt ++                  /* count character */
if(ch==' ' );               /* space ? */
wordcnt++;
}
printf("\n character count is %d ",charcnt );
printf("\n word count is %d " , wordcnt );
}
```

تذكـــر

عندما تريد	استخدام الدالة
طباعة حروف او أرقام	printf( )

puts( )	طباعة عبارات حرفية فقط
scanf( )	ادخال حروف او أرقام
gets( )	ادخال عبارات حرفية
getch()	ادخال حرف دون إظهاره على الشاشة
getche( )	ادخال حرف مه اظهاره 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	ادخال البيانات بالألوان

الـــــدوال FUNCTION

الدوال فى لغةالـ C نوعين :-

(١) دوال اللغة Built in Function وهى الدوال القياسية مثل دالة ( )printf أو دالة ( )scanf وهى دوال عامة يستطيع اى مبرمج استخدامها

(٢) دوال المستخدم المبتكرة User Function:-

- **وهى الدوال التى من وضع المبرمج**
- **والهدف منها : انه عند تكرار مجموعة من سطور الأوامر اكثر من مرة فى مواضع مختلفة فـــإن أوامر التكرار لن تكون ذات منفعة . ولذلك يتم كتابة هذه السطور منفصلة عن البرنامج الأساسى**

مزايا استخدام الدوال

١ - عدم تكرار التعليمات داخل البرنامج : حيث يتم إنشاء الدالة مرو واحدة ثم يتم استدعائها أكثر من مرة عند الحاجة إليها

٢ - باستخدام الدوال يصبح البرنامج أكثر وضوحاً

الصور العامة:

```
# include < filename>
function declarations;
main( )
{
```

```
    function1
    function2
}

function1
{
------
------
}

function2
{
------
------
}
```

أذ ن تتكون الدالة من :-

(١) الإعلان عن الدالة (٢) جسم الدالة

مثال : دالة بدو ن دليل

```
#include <stdio.h>
#include<conio.h>
void line2(void);
main( )
{
clrscr( )
line2( )
printf(" ** Allah the god of all world ** \n ");
line2( )
/* end of main( ) function */
}
void line2(void)
{
int j;
for(j=0;j<=40;j++);
printf( " * ");
printf("\n ");
}
```

فى البرنامج السابق أنشأنا دالة بالاسم ( )line2 وقد ظهرت كلمة ( )line2 فى ثلاث مواضع :-

الموضع الأول : يسمى الأعلان عن الدالة**function declaration** يكون ذلك قبل الدالة الرئيسية ( )main كما فى السطر رقـــــم ٣ ونلاحظ الفاصلة المنقوطة فى نهاية الجــزء لأنه أعلان.

الموضع الثانى : داخل الدالة الرئيسية ( )main ويظهر فى أى مكان داخل الدالة الرئيسية ويسمى **function coling** أى استدعاء الدالة ويكون بالشكل ( )line2 كما فـــى الســطر ٧و٩ وفيه يتم كتابة اسم الدالة فقط بدون نــوع وإذا كــان لهــا معــاملات نكتــب المعاملات.

الموضع الثالث : يكتب بعد انتهاء الدالة الرئيسية ( )main. وهذا الجزء يســمى تعريــف الدالــة **function definition** وفيه يتم كتابة محتويات الدالة . وتبدأ فى البرنــامج مــن السطر رقم ١١ باسم الدالة ثم بالقوس } وكانها برنامج ونبدأ كتابة تعليمات الدالــة بعد القوس ثم ننتهى بالقوس {

** **أنواع الدوال** ** Function Type

int function	**دوال تعيد قيمة صحيحة**
float function	**دوال تعيد قيمة حقيقية**
string function	**دوال تعيد عبارة حرفية**
void function	**دوال لا تعيد اى قيمة**
struct function	**دوال تعيد قيمة من نوع structure**

تمرين :

```
# include <stdion.h>
int sum(int a, int b )                    الاعلان عن الدالة
main( )
{
int z , x = 10 , y = 40;
z = sum(x,y);
printf("\n\n z = %d " , z );
}
/*    الدالة */
int sum(int a , int b )
{
```

```
int s;
s = a + b ;
return s;
}
```

** **ملاحظات على البرنامج** **

- فى السطر رقم ٢ تم الاعلان عن دالة بالاسم ( )sum وسبقت بالكلمة int وهى نوع الدوال وتقابل كلمة void مع ملاحظة وجود متغيرين بين الأقواس وهما معاملات الدالة.
- فى السطر رقم ٦ يتم استدعاء الدالة وبين أقواسها المتغيرات x ,y ويستخدمان كمعاملات للدالة (لابد من كتابة معاملات الدالة لأننا أعلنا عنها بهذه الصورة)
- تشمل السطور من ٩ الى ١٤ على جمل الدالة نفسها :-
  - السطر رقم ٩ نعوض عن المتغير a بالقيمة الموجودة فى المتغير x وهى القيمة ١٠. كذلك نعوض عن المتغير b بالقيمة الموجودة فى المتغير y وهى ٤٠.
  - السطر رقم١٢ نجمع محتويات كلا من المتغير a والمتغير b ونضع النتيجة فى متغير جديد هو s
  - السطر رقم ١٣ نطلب اعادة محتويات المتغير s الى مكان استدعاء الدالة باستخدام كلمة return
  - نفهم ان جملة z = sum(x,y) الموجودة بالسطر رقم ٦ تعادل الجملة z = s .

ملاحظة هامة: معنى الدالة يتضح من القاعدة التى تقول أن نوع الدالة يتوقف على القيمة المرتجعة من الدالة.

فإذا كانت القيمة المرتجعة int كان نوع الدالة int

وإذا كانت القيمة المرتجعة float كان نوع الدالة float

أما الدالة التى لا تعيد قيمة(الدالة لا تشتمل على جملة return) فتكون من نوعvoid

** **<u>استدعاء الدالة</u>** **

- يتم استدعاء الدوال اما بمعاملات او بدون معاملات
- تكون الدالة بدون معاملات مثل الدالة void line2(void) اى عدم كتابة قيم بين أقواس الدالة

* برنامج يوضح كيفية استدعاء الدالة بمعاملات *

```
# include < stdio.h>
# include <conio.h>
void line3(int no)
main( )
{
clrscr( )
line3(30);
printf(" ** Allah the god of all world ** \n ");
line3(70);
}
```

```
/*  الدالة  */
void line3(int no)
{
int j , no;
for(j = 0 ; j <= no ; j++ )
printf(“ * “ );
printf(“\n”);
}
```

** ملاحظة : الدالة هنا لها معامل واحد من نوع صحيح وهو no. وفى كل مرة يتم ارسال قيمة مختلفة للمعامل وذلك عند استدعاء الدالة.

** **استدعاء الدالة بمتغيرا ت** **

- ممكن استدعاء الدالة بمعاملات من نوع قيم ثابتة موجودة بالبرنامج نفسه
- وأيضا يمكن ان تكون هذه المعاملات متغيرات تستقبل قيمها من المستخدم او من داخل البرنامج وهذا يفيد فى حالة تغير واختلاف المتغيرات فى كل مرة (إعطاء مرونة فى التعامل مع البرنامج)

* برنامج لتحديد الكمية الأكبر من ثلاث كميات صحيحة

```
# include <stdio.h>
/* determine the largest of three integer quantities */
main( )
{
int a, b, c, d;
/* read the integer quantities */
printf(“\n a = “);
scanf( % d “ , &a );
printf(“\n b = “);
scanf( % d “ , &b );
printf(“\n c = “);
scanf( % d “ , &c );
/* calculate and display the maximum value */
d = maximum( a, b );
printf(“\n \n maximum = % d , maximum(c ,d ));
}
/* determine the larger of two integer quatities */
maximum(x ,y )
```

```
int x ,y;
{
int z;
z = (x >= y ) ? x | y;
return(z);
}
```

ملاحظة هامة : من ضمن أوامر التحكم (علامة الاستفهام الشرطية |?) حيث :-

- (x > = y) عبارةtest اختبار
- ؟ عبارة عن سؤال if
- | عبارة عن Else

بمعنى إذا كان الاختبار (x >= y) صحيحا يأخذ القيمة x وإذا كانت القيمة غير صحيحة يأخذ القيمة y

** باستخدا م ا لدولا أكتبرنامج
لتحويل الساعات إلى دقائق مع إيجاد الفرق

```
# include<stdio.h>
/* calculates difference between two times */
main( )
{
int mins1  , mins2;
printf(“ type first time ( from 3:22 ): “ );
mins1=getmins();
printf(“ type second(later)  time “ );
mins2=getmins();
printf(“ Difference is %d minutes.” , mins2-mins1);
}
/* getmins function */
/* gets time in hours : minutes formal */
/* return time in minutes */
getmins( )
{
int hours, minuts;
scanf(“ %d : %d “, &hours , &minutes);
return(hours * 60 + minutes);
}
```

* الآتى برنامج يستعمل دالة من نوع int تتكون من دالتين :-

- دالة لحساب مربع القيمة وإعادة حاصل الضرب الى الدالة الرئيسية

- دالة تحسب نكعب قيمة وتعيدها عند الاستدعاء

```
# include < stdio.h>
int sqr(int a):
int qup(int q):
main( )
{
int s , qu , no =10;
s = aqr(no);
qu = qup(no);
printf("\n squer of no = %d " , s ):
printf("\n qupic of no = %d " , qu ):
}
int sqr(int a)
{
int v1;
v1= a*a;
return v1;
}
int qup(int q )
{
int v2;
v2=q*q*q;
return v2;
}
```

نتيجة التنفيذ :

```
squar of on = 100
qupic of on = 1000
```

الآتي برنامج يستعمل دالة من نوع float تقوم بجمع رقمين وإعادة النتيجة الى الدالة الرئيسية عند الاستدعاء . والقيمة المرتجعة قيمة حقيقية float وبالتالى يكون نوع الدالة float

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

float add( float x , float y )

main( )
{
 float no1, no2;
printf("\n enter no1 , no2 : );
scanf(" %f ,%f " , &no` ,&no2 ):
```

```
printf("\n addition of number is  %f", add(no1,no2) );
}

float add(float x , float y );
{
float yt;
yt= x + y ;
return yt;
}
```

عند التنفيذ نحصل على :-

```
Enter no1 ,no2 : 3,2 , 4,3
Addition of squares is  6.5
```

** الآتي برنامج لإيجاد العدد فردى أم زوجى وكذلك موضحا العـدد سـالب أم موجب

```
# include < stdio.h>
/* tests use of external variables */

int keynumb;

main( )
/* external variables */
{
printf(" type keynumb : " );
scanf("%d " , &keynumb );
addeven( );
negative( );
}

/* checks if keynumb is odd or even */
oddeven( )
{
if (keynumb % 2 )
printf(" keynumb is odd. \n");
else
printf( " keynumb is even. \n");
}
```

```
/* checks if keynumb is nagative */
ngative( )
{
if (keynumb < 0 )
printf(“ keynumb is negative . \n”);
else
printf( “ keynumb is positive . \n”);
}
```

الماكـــرو MACROS

هو مجموعة من التعليمات تؤدى غرض معين ويشبه إلى حد كبير الدالـــة function ويـــتم إنشـــاء الماكرو مرة واحدة وبعد ذلك يمكن استدعائه كلما احتجت إيه (أى يتم تعريف الثوابـــت او عمليـــات محدده فى بداية البرنامج وتكون لها صفه العمومية للاستخدام داخل الدالة الرئيسية والدوال الفرعية)

إذن : الماكرو من ضمن بناء برنامج لغة الـــــــ C

كيفية إنشاء الماكرو :-

- يتم ذلك باستعمال الكلمة #define
- وهذه الكلمة تسمى directive او preprocessor ومعناها التوجيه

* الصورة العامة

```
# define macro line
```

مثلا

```
#define  a  5
```

وهى عبارة عن تعريف طرف بطرف ومعناها عرف المتغير a بالقيمة 5

تمرين يوضح كيفية الإعلان عن الماكرو وكيفية استعماله

```
# include<stdio.h>
# define sum(a,b) a+b
# define mul(x,y) x*y

main ( )
{
```

```
int v1=5 , v2 = 10;
printf("\n\n sum(v1,v2) = % d",sum(v1,v2);
printf("\n\n mul(v1,v2) = % d",mul(v1,v2);
}
```

ملاحظات على البرنامج :-

- فى السطر رقم 2 استخدمنا كلمة **define** لتعريف ماكرو بالاسم **sum** ووظيفته استبدال المتغيرين**a,b** بالصور **a+b**
- كذلك فى السطر رقم 3 يستبدل المتغيرين **x,y** بنتيجه الضرب **x*y**
- ومعناها كلما قبل المترجم اللغة الطرف الأول للماكرو يستبدله بالطرف الثانى

ملاحظ هامة : المتغيرين **a,b** يمكن استبدالهما بأى متغيرين أو قيمتين داخل البرنامج. واسم الماكرو هو الذى يحدد العملية التى يقوم بها الماكرو (هل هى عملية جمع أم ضرب أم ..... بناء على المعادلة الموجودة فى الطرف الأيمن من الماكرو)

* **الفرق بين الماكرو وبين الدالة** :- اى برنامج يمر بثلاث مراحل :-

المرحلة الأولى : كتابة وهذا يسمى **source code** ويخصص لملف المصدر الامتداد **C.**

المرحلة الثانية : ترجمة البرنامج للغة يفهمها الحاسب وتسمى هذه المرحلة **compilation** ويخصص للملف الامتداد **obj.**

المرحلة الثالثة : ربط الملف الـ **object** بمكتبات اللغة ليصبح قابل للتنفيذ وتسمى هذه العملية **linking** ويخصص لهذا الملف الإمتداد **exe.**

ومن خلال هذه المراحل تستطيع ان تستخرج الفرق بين الماكرو وبين الدالة كما يلى :-

١ - فى مرحلة الكتابة ايس هناك فرق بين الماكرو وبين الدالة.

٢ - فى مرحلة الترجمة **compilation** يتم تحويل تعليمات الدالة الى لغة الآلة **object** وتنتظر وحدة الربط **linking** ولا تنفذ الدالة إلا فى مرحلة الربط

٣ - اما فى حالة الماكرو يتم استبدال الماكرو بنتيجة تنفيذ الماكرو

(فى التمرين السابق يتم استبدال الماكرو الموجود فى السطر رقم 7 بنتيجة التنفيذ مباشرة . اى يتم وضع القيمة 15 وهى نتيجة تنفيذ الماكرو مكان **sum(v1,v2)** وبالتالى عندما تاتى مرحلة التنفيذ يجد البرنامج نتيجة تنفيذ الماكرو جاهز وهى 15)

\ **مزايا الماكرو** :-

١ - بسيط فى الإنشاء

٢ - بسيط فى الاستعمال ويعطى فى النهاية ملف تنفيذى أصغر

٣ - إذا كانت العملية المطلوبة بسيطة ويمكن كتابتها فى سطر واحد نستعمل الماكرو.

تمرين :

```
# include<stdio.h>
# define  pi 3.14159
main ( )
{
float area(float);
float radius;
printf("enter radius sphers : ");
scanf("%f", &radius):
printf("area of sphere is 2f",area(radius));
}
/* returns ares of sphere */
float area(rad)
float rad;
{
 return(4*pi*rad*rad);
}
```

المصفوفات ARRAYS

مقدمة *

تنقسم البيانات الى بيانات حرفية char وبيانات رقمية int وبيانات حقيقة float وتسمى هذه الأنواع (int , float , char) بالأنواع الرئيسية للبيانات حيث لا يمكن تجزئتها أقل من ذلك.

وهناك انواع أخرى من البينات تسمى بالأنواع المشتقة (Type dived data) من هذه النواع المصفوفات arrays.

المصفوفة **

المصفوفة array هى مجموعة من البيانات التى تتشابه فى النوع ولها اسم مشترك. تتشابه فى النوع بمعنى أن تكون البينات التى تخزنها فى المصفوفة كلها اعداد صحيحة int او اعداد ذات علامة عشرية float . فيمكن ان نضع المصفوفة من اى نوع من انواع البيانات.

***الإعلان عن المصفوفة** * للإعلان عن إحدى المصفوفات هناك ثلاثة أشياء يجب ان تحددها.

(١) اسم المصفوفة : وهو اسم تختاره مثلما تختار اسم أى متغير

(٢) عدد العناصر داخلها

(٣) نوع البيانات المستخدم فيها

مثلاً : نفرض أنك تريد تخزين مرتبات 120 موظفاً يعملون فى شركتك وكانت مرتبات هؤلاء الموظفين تحتوى على كسور عشرية. فى هذه الحالة شيكون نوع البيانات float ويتم الاعلان عن المصـــفوفة كما يلى : float salary[120]

تنقسم المصفوفة إلى المصفوفة ذات بعد واحد ومصفوفات ذات البعدين.

** المصفوفة ذات البعد الواحد :-

مثلا : A[2 5 9 12 15] وتسمى مصفوفة ذات بعد واحد لأنها تتكون من صف واحد أو عمود واحد وفيها حرف A هو اسم المصفوفة والأرقام هى عناصر المصفوفة ويتم الإشارة إلى كل عنصر بـــرقم العنصر اى بترتيبه داخل المصفوفة

** الإشارة إلى عناصر المصفوفة **

- يتم ذلك بان تذكر اسم المصفوفة ثم رقم المصفوفة الذى تريد التعامل معه بين القوسين [ ]
- اى عندما ترغب فى ذكر رقم العنصر داخل المصفوفة يجب أن تبدأ العد داخل المصفوفة من صفر وليس من 1
- مثلاً : إذا كنت تريد وضع الرقم أو القيمة 75 فى العنصر 3 من المصفوفة المســماه student نكتب التالى : student[3] = 75
- وإذا كنت تريد ان تنقل القيمة الموجودة فى العنصر 38 مــن المصـــفوفة salary إلـــى أحـــد المتغيرات نكتب التالى : a = salary[37]
- كذلك يمكن ان تمرر القيمة الموجودة فى أحد عناصر المصفوفة الى اجراءات مكتبة التشغيل مثل

printf(" The higest mark is %d ", student[45]);

** البرنامج التالى يعرض مثالاً عن كيفية الإعلان عن إحدى المصفوفات واستخدامها :-

```
# include <stdio.h>
# define max 10
void main(void)
{
float salary[max];
float average;
int count;
average = 0.0;
for(coint=0; count<max ; count++)
```

```
{
printf(" please enter salary for employee %d", count+1);
scanf("%f", &salary[count]);
}
printf("\n\n salary average is : %f", average/max);
}
```

ملاحظات على البرنامج :-

- هذا البرنامج يطلب من المستخدم إدخال مرتبات 10 موظفين ثم يقوم بحساب وطباعة متوسط هذه الرواتب.
- فى بداية البرنامج نعلن عن المصفوفة float salary[max] هذه المصفوفة تستخدم لتخزين 10 اعداد ذات كسور عشرية وبالتالى فإن الإعلان السابق يساوى float salary[10]
- بعد ذلك نعلن عن المتغير average وهو تستخدم لتخزين مجموع الرواتب حتى نتمكن من استخراج متوسطها.
- لهذا نستخدم جملة التكرار for لادخال 10 موزفين لذلك نستخدم الإجراء scanf() لنطلب من المستخدم إدخال الرواتب scanf("%f ", &salary[count])
- وكلما ادخل المستخدم أحد الرواتب أضفنا قيمته الى المتغير average كما يلى

average += salary[count]

الملاحظة الهامة : هى كيف استخدمنا المتغير count كعداد لتكرار for وفى نفس الوقت استخدمناه للإشارة للعناصر المختلفة داخل المصفوفة. Scanf("%f", salary[couny]) ففى بداية التكرار سيكون المتغير count يساوى الصفر

**** تهيئة المصفوفة عند الإعلان عنها ****

- من الممكن ان تقوم بتهيئة المصفوفة عند الإعلان عنها إذا كنت تعرف مسبقا المحتويات التى ستضعها فيها int marks[5] = { 12 4 7 9 11} هنا نعلن عن مصفوفة مكونه من خمسه عناصر عددية صحيحة int

*** المصفوفة الغير محددة العدد ****

- إذا كنت ستهيئ مصفوفة عند الاعلان عنها يمكن ان تترك للمترجم مهمة حساب عدد العناصر فيها.
- مثلاً : int marks[ ] = { 5 9 4 10 7 }
- هنا أعلنا عن المصفوفة ولكن لم نحدد عدد العناصر فيها لأن المترجم سيقوم بِعَد القيم المذكورة بين القوسين { } ويحدد حجم المصفوفة تلقائيا .

** برنامج **

```
# include <stdio.h>
/* calculate the average of a numbers , the compute deviation of each number
about the average */
main( )
{
int n , count ;
fliat avg, d , sum=0 ;
/* read in a value for n */
printf("\n how many numbers will be average? ";
scanf("%d", &n);
printf("\n");
/* read in the numbers and calculate ther sum */
for(count = 0 ; count < n count++ )
{
printf("i = %d    x= ", count+1);
scanf("%f", &list[count]);
sum+= list[count];
}
/* calcuate and write out the average */
avg = sum/n;
printf("\n the average is %5.2f\n\n ", avg);
/* calculate and write out the deviations about the average */
for(count = 0 ; count < n ; count++)
{
d=list[count] – avg ;
```

```
printf("i = %d   x%5.2f   d=%5.2f \n" , count+1, list[count] , d);
}
}
```

ناتج هذا البرنامج :-

```
The average is 4.18
i = 1 x = 03.00     d = -1.18
i = 2 x = -2.00     d = -6.18
i = 3 x = 12.00     d = 7.82
i = 4 x = 4.40      d = 0.22
i = 5 x = 3.50      d = -0.68
```

*** * برنامج إعادة ترتيب قائمة من الأعداد * ***

```
# include<stdio.h>
# define size 100
/* reorder a one-dimensinal, integer array from smallest to largest */
main( )
{
int i , n , x[size];
void reorder(int n , int x[ ]);
/* read in the a value for n */
printf("\n how many number will be entered ? " );
scanf("%d", &n );
printf("\n");
/* read in the list of number */
for(i = 0 ; i<n ; i++)
{
printf("i = %d    x = ", i+1);
scanf("%d", &x[ i ]);
}
/* reprder all array elements */
reorder(n , x )
/* display the reordered list of numbers */
print("\n\reordered list of number :\n\n ");
for(i=0;i<n; i++)
printf("i=%d    x=%d\n", i+1 , x[ i ] );
}
int i, item ,temp;
```

```
for(item = 0;item<n-1 ; item++ )
if(x[ i ] < x[item])
{
/* interchange twwo elements */
temp = x[item];
x[item] = x[ i ];
x[ i ] = temp ;
}
return;
}
```

* المصفوفة ذات البعدين :*

- هى المصفوفة التى ترتب عناصرها فى شكل صفوف وأعمدة
- ويتم الإعلان عنها بالشكل التالى int a[5] [10] ومعناه أن المصفوفة ش مصفوفة ذات بعدين 5 صفوف و 10 أعمده ويتم الاشارة الى العنصر برقم الصف ورقم العمود

ملاحظة : عند استخدام مصفوفة ذات البعدين لابد من استعمال دوارة for

* **إعطاء قيمة ابتدائية للمصفوفة**: يمكن إعطاء قيمة ابتدائية للمصفوفة ذات البعدين كمايلى

```
int a[3][4] =
        {
        {4,5,7,8 },
        {3,2,4,5},
        {7,8,9,6}
        };
```

ملاحظة : فى هذا الشكل يأخذ العنصر رقم 0,0 القيمة 4 والعنصر رقم 0,1 القيمة 5

- تتحد القيم الأربع فى أول زوج أقواس { } داخلية لعناصر المنظومة فى الصف الأول وتتحدد القيم الموجودة فى زوج الأقواس { } التالى لعناصر المنظومة فى الصف الثانى٠٠٠ وهكذا

شكل آخر للمصفوفة ذا ة البعدين :

```
int values[3][4] = {1,2,3,4,5,6,7,8,9,,10,11,12};
```

فى هذا الشكل يأخذ العنصر رقم 0,0 القيمة 1 أى values[0][0] ويــأخذ العنصر 0,1 القيمة 2 أى values[0][1]

مثال: **على استخدام المصفوفات متعددة الأعمدة**

```
# include <stdio.h>
# define stud 3
```

```
# define marks 10
void main (void)
{
int student[stud][marks];
int c1 ,c1 ;
for(c1 = 0 ; c1 < stud ; c1++)
  for ( c2 = 0 ; c2 < marks ; c2++)
 {
 printf ( " enter marks: %2d for student %2d : " , c2+1 , c1+1 );
 scanf (" %d ", &student[c1][c2]);
 }
for(c1 = 0 ; c1 < stud ; c1++ )
for(c2= 0 ; c2 < marks ; c2++)
printf(" student [%2d] , marks[%2d] = %d\n ", c1+1, c2+1 , student[c1][c2]);
}
```

ملاحظات : فى بداية البرنامج نعلن عن المصفوفة student ذات البعدين

```
int student[stud][marks];
```

- وبعد ذلك نستخدم تكرارين for داخل بعضهم . نستخدم هذين التكرارين لطلب الدرجات من المستخدم. التكرار الأول يستمر بعد الطلاب والتكرار الثانى بعد الدرجات.
- كذلك نستخدم تكرارين لنطبع الدرجات التى أدخلها المستخدم.

*** برنامج : جمع جدولين من العداد اى نحسب مجموع العناصر المتناظرة فى الجدولين أى نحسب**

```
c[i][j] =  a[i][j] + b[i][j]
```

ثم نستخرج الجدو ل الجديد محتوياً على المجموع

```
# include <stdio.h>
# define mrow 20
# define mcol 30
/* caluculates the sum of the elements in two tables of integers */
void main (void)
{
int mrow , mcol;
/* array definitions */
int a[mrow][mcol] , b[mcol][mcol] , c[mcol][mcol];
/* function protyotypes */
void readinput(int a[ ][mcol],int nrow , int ncol);
void computesum ( int a[    ][mcol], int b[   ][mcol] , int c[   ][mcol],  int nrow ,
        int ncol);
```

```
void writeoutput ( int c[  ][mcol], int nrow , int ncol );

printf ( " How many rows ?" );
scanf (" %d " , nrow);
printf ( " How many columns  ?" );
scanf (" %d " , ncol);

printf ( " \n\n first table : \n" );
readinput( a, nrow , ncol );

printf ( " \n\n second table : \n" );
readinput( b, nrow , ncol );

computsum(a, b, c, nrow, ncol);
printf ( " \n\n sums of the elements : \n \n" );
writeoutput(c , nrow , ncol);
}

void readinput( int a[  ][mcol] , int m , int n )

/* read in a table of integers */
{
int row , col ;
for(row = 0 ; row < m ; row ++ )
{
printf\n enter data for row no. %2d \n " , row+1 );
for( col = 0 ; col < n ; col++)
scanf(" %d ", &a[row[col]):
}
return;
}

void computsum(int a[ ][mcol] , int b[  ][mcol] , int c[  ][mcol] , int m ,int n )
/* add the elements of two integer tables */
{
int row, col ;
for(row = 0 ; row < m ; row++)
for(col = 0 ; col < n ; col++)
c[row][col] = a[row][col] + b[row][col];
return;
}

void writeoutput(int a[ ] [mcol] , int m , int n )
/* write out a table of integers */
{
```

```
int roe , col ;
for (row = 0 ; row <m ; row ++)
{
for(col = 0 ; col < n ; col ++)
printf("%4d ", a[row][col]):
printf("\n";
}
return;
}
```

ملاحظات :

- a ,b ,c مصفوفة ثنائية الأبعاد ولكل منها نفس عدد الصفوف ونفس عدد العمدة
- row متغير صحيح يحدد العدد الفعلى للصفوف فى كل جدول.
- ncol متغير صحيح يحدد العدد الفعلى للأعمدة فى كل جدول
- row عداد صحيح يحدد رقم الصف
- col عداد صحيح رقم العمود

* أفرض ا ن البرنامج استخدم فى جمع الجدولين الاتى :-

الجدول الأول			
1	2	3	4
5	6	7	8
9	10	11	12

الجدول الثانى			
10	11	12	13
14	15	16	17
18	19	20	21

والاتى تنفيذ البرنامج :-

```
how many rows?  3
how many columns? 4
first table :
enter data for row no. 1
1       2       3       4
enter data for row no. 2
5       6       7       8
enter data for rpw no. 3
9       10      11      12

second tables :
enter data for row no. 1
10      11      12      13
enter data for row no. 2
14      15      16      17
enter data for row no. 3
```

18 19 20 21

suns if the elements :
11 13 15 17
19 21 23 25
16 19 31 33

* سلاسل الحروف * (مصفوفة العبارة الحرفية array of string)

تستخدم كلمة سلسلة حروف مقابل لكلمة string

* الإعلان عن سلاسل الحروف *

- **عندما ترغب فى استخدام مصفوفة حروف فأنك تعلن عنها كما تعلن عن بقية المصفوفات الأخرى**

** **وللأعلان عن نوع البيانات char نكتب التالى :** **char name[80]**

- **هنا نعلن عن المصفوفة name التى تتسع لـ ٧٩ حرف (لأن لغة c تفترض ان المصفوفة تنتهى بحرف الصفر (null)**

** **شكل آخر للأعلان** **char name[ ] = "ali ahmed"**

- **هنا سوف يقوم المترجم بعـــد الحروف وتحديد حجم المصفوفة وتهيئتها بالحروف المذكورة**

** استخدام مصفوفة الحروف :-

- **بعد أن تنتهى من الإعلان عن مصفوفة الحروف . يمكنك البدء فى استخدامها**
- **كذلك يمكن ان تمرر اسم المصفوفة الى أحد اجراءات مكتبة التشغيل مثـل الاجـراء printf أو gets أو غيرها. ويمكنك أيضا الكتابة الى أى حرف داخلها أو قراءة الحرف فيها ووضـعها فـى متغيرات أخرى**

** والبرنامج التالى يوضح استخدام مصفوفات الحروف :-

```
# include<stdio.h?
```

```
void main(void)
{
char namr[80]
printf("place enter your name : ");
gets(name);
printf("welcome , %s ", name);
}
```

* **ملاحظات** :

- فى هذا البرنامج يعلن عن مصفوفة الحروف name ثم نستخدم الاجراء gets ليقرأ اسم المستخدم من لوحة المفاتيح ثم يخزن الحروف التى أدخلها المستخدم فى المصفوفة
- إذا رغبت أن تتعامل مع كل حرف من حروف المصفوفة على حدة اكتب التالى name[0] = 'a'
- إذا اردت ان تنقل القيمة الموجودة فى أحد الحروف فى المصفوفة الى أحد المتغيرات الأخرى نكتب ss = name[15];

** مصفوفة الحروف تنتهى بالرقم صفر **

يجب ان تلاحظ ان مصفوفة الحروف يجب ان تنتهى بالرقم صفر (null)

* برنامج *

```
# include <stdio.h>
void main(void)
{
char strin[ ] = "hello";
printf("character array = %s , its length = %d \n, strin , sizeof(string));
}
```

* **ملاحظات** :

- فى هذا البرنامج نعلن عن مصفوفة باسم string
- نطلب من المترجم ان يهيئها بان يضع فيها كلمة hello كما يلى char string[ ] = "hello"
- بعد ذلك نستخدم الاجراء printf ليطبع محتويات هذه المصفوفة
- استخدام كلمة sizeof لنحصل على حجم هذه المصفوفة ونطبعه يكون الناتج

character array = hello its length = 6

- لاحظ ان حجم المصفوفة هو 6 حروف فى حين اننا وضعنا فيها 5 فقط وهى الحروف hello
- السبب : هو أن مصفوفة الحروف تنتهى دائماص بالرقم صفر الذى يستخدم كدلالة على نهايتها كما فى الشكل :

1	2	3	4	5	6
h	e	l	l	o	\0

** لكى نتأكد من هذه النقطة انظر البرنامج التالى :-

```
# include<stdio.h>
void main(void)
{
int count;
char string[  ] = " hello "
for(count = 0 ; count < sizeof(string); count ++)
printf(" string[%d] = =%d %c \n ", count, string[count], string[count]);
}
```

- فى هذا البرنامج نعلن عن مصفوفة الحروف string ثم نستخدم تكرار for لنطبع محتويات كل حرف فيها
- لاحظ شرط استمرار التكرار. Count < sizeof(string)
- سيؤدى هذا الشرط الى استمرار التكرار بعدد الحروف الموجود فى المصفوفة string وفى كل مرة يعمل فيها التكرار نطبع الحرف الموجود فى المصفوفة كحرف وكرقم فى جدول آسكى كما فى الشكل التالى :-

String[0] = 104 h	String[1] = 101 e
String[2] = 108 l	String[3] = 108 l
String[4] = 111 o	String[5] = 0

لاحظ كيف يعرف البرنامج الحرف السادس فى المصفوفة على أنه = صفراً

** بعض الاجراءا ت التى تتعامل مع مصفوفات الحرو ف **

هناك العديد من اجراءات مكتبة التشغيل التى تسهل التعامل مع مصفوفات الحروف

(١) معرف عدد الحروف فى المصفوفة :-

- إذا اردت ان تعرف عدد الحروف الموجودة فى أى مصفوفة حروف استخدم الاجراء strlen يعود بعدد الحروف الفعلية الموجودة فى المصفوفة بدون عد اصفر الموجود فى نهايتها
- مثلا len = strlen("count this string ")

(٢) نسخ مصفوفة حروف الى اخرى :-

- لا تسمح لك لغة الـــــ c بنقل محتويات مصفوفة حروف الى اخرى مباشرة .
- هناك الاجراء strcpy يستخدم كما يلى : strcpy (string1 , string2) ســيقوم الاجــراء بنسخ محتويات المصفوفة string2 الى المصفوفة string1
- ويمكن ان نستخدم الاجراء strcpy لنسخ مجموعة احرف الى مصفوفة اخرى .
- مثلاً strcpy(string , " hello my string "); هنا سيقوم المترجم بوضع الجملة المذكورة فى المصفوفة string
- كذلك يجب أن نتأكد أن المصفوفة التى ستنقل إليها الحروف تتسع لكل الحروف الموجــودة فــى المصفوفة الأخرى لأن الاجراء strcpy لا يتأكد من ذلك.

(٣) دمج محتويات مصفوفتى حروف :-

- لعمل ذلك استخدم الاجراء strcat كما يلى: strcat(string1 , string2) سيضيف الاجراء srcat محتويات المصفوفة string2 الى محتويات المصفوفة string1
- هنا يجب أن نتأكد أن المصفوفة string1 تتسع لكل الحروف الموجودة فى المصفوفة string2 بالإضافة الى الحروف التى كانت أصلا موجودة

```
# include < stdio.h>
void main(void)
{
char string1[20] = "learning c ";
char string2[   ] = " is easy ";
strcat(string1 , string2 );
printf(string1);
}
```

الناتج : learning c is easy

(٤) مقارنة محتويات مصفوفتين :

- من الممكن ايضا ان تقارن محتويات مصفوفتين من الحروف
- لعمل ذلك استخدم الاجراء strcmp وله الصيغة التالية:

```
result =  strcmp(string1 , string2 );
```

- سيقارن الاجراء strcmp محتويات المصفوفتين ويضع ناتج المقارنة فى المتغير result
- إذا كانت المصفوفتان متطابقتين (اى يحتويان على نفس مجموعة الحروف) يكون المتغير result فى هذه الحالة يساوى صفراً
- وإذا كانت المصفوفتان مختلفتين فإن المتغير result يحتوى على نتيجة بخلاف الصفر

* البرنامج التالى يعطى مثالاً على استخدام الاجراء strcmp :-

```
# include < stdio.h>
# include  <string.h>
void main(void)
{
char string1[80];
char string2[80];
int result;
printf(" enter the first string ");
gets(string1);
printf(" enter the second string : ");
gets(string2);
result = strcmp(string1,string2);
if(result == 0 )
printf("the two string are identical : );
else
printf("the two string are different : );
```

```
}
```

ملاحظات :

- فى هذا البرنامج سيطلب من المستخدم غدخال عبارتين ثم يستخدم الاجـراء strcmp ليقلـرن بينهما ويعرض نتيجة المقارنة
- ويمكن ايضا ان تستخدم الاجراء strcmp ليقارن بين محتويات مصفوفة حـروف ومجموعـة حروف مثل result = strcmp(string1,string2);

*** برنامج عام ***

لاستخدام الاجراءا ت المختلفة التى تتعامل معها مصفوفة الحرو ف

```
# include < stdio.h>
# include < string.h>
void main(void)
{
char string1[80];
char string2[80];
int result , choice;
printf(" enter the first string : " );
gets(strig1);
printf("enter the second string : " );
gets(strung2);
printf("\n enter the function you want: \n "
      "1- strcpy \n "
      "2- strcat \n "
      "3- strcmp \n");
printf("enter you choice : );
scanf("%d " , &choice);
switch(choice)
{
case 1:
      printf(\n copy second string to first \n");
      strcpy(string1 , string2 );
      printf(string1);
break;
case 2:
      printf(\n merging the two string \n");
      strcat(string1 , string2 );
```

```
        printf(string1);
 break;
 case 3:
        printf("\n comparing the two strings \n ");
        result = strcmp(string1,string2);
        if(result == 0 )
        printf( " the two strimg are identical ");
        else
        printf( " the two strimg are different ");
break;
default :
printf("plase enter a number 1-3 " );
}
}
```

** تمرير مصفوفة الى الاجراءا ت **

- من الاستخدامات المفيدة للمصفوفات سواء المصفوفات العادية أو مصفوفات الحروف هو ان تتمكن من تمريرها الى بعض الاجراءات التى تكتبها
- عندما ترغب فى تمرير إحدى المصفوفات الى اجراء يجب ان تعلن عن الاجراء بحيث تخبر المترجم انه سيتلقى مصفوفة وليس متغيراً عادياً مثل ; void fn(int num[ ]) هنا تخبر المترجم ان الاجراء fn سيتلقى مصفوفة من نوع int بعد ذلك عندما ترغب فى تمرير المصفوفة اى الاجراء أذكر اسم المفوفة فقط كما يلى ;fn(num)

* البرنامج التالى :-

يعطى مثالاً على كيفية تمير مصفوفة حرو ف الى احد الاجراءا ت :-

```
# include < stdio.h>
# include <ctype.h>
# include <string.h>
void  changetoupper(char string[ ] );

void main(void)
{
char string[80];
do
  {
   printf(" enter a string to change to upper " case \n enter \ "quit \ "to exit");
  gets(string);
  changetoupper(string);
  }
   while(strcmp(string , "quit"));
}
```

```
void changetoupper(char string[ ] )
{
unsigned int count;
for(counr = 0 ; count < strlen(string); count++)
printf("%c" , toupper(string[count]));
printf("\n\n");
}
```

ملاحظات :

- هذا البرنامج يطلب من المستخدم ان يدخل مجموعة حروف ثم يحولها الى حروف كبيرة
- عندما يبدا البرنامج يدخل فى تكرار **do….while** ويستمر فى طلب مجموعات الحروف حتى يدخل المستخدم كلمة **quit** وعندها يخرج من التكرار
- فى كل مرة يدخل المستخدم مجموعة حروف تستدعى الاجراء **changetoupper** ونمرر له مصفوفة الحروف التى أدخلها المستخدم ليطبعها بالحرف الكبيرة.
- هنا تستدعى الإجراء **toupper** أولاً ثم نمرر الحرف الذى يعود به الى الإجراء **printf** ليطبعه **printf("%c" , toupper(string[count]));**

** **بعض دوال العبارات الحرفية** string functions

الدالة	وظيفتها
strcat( )	اضافة سلسلة حرفية (كلمة) الى نهاية سلسلة حرفية اخرى (كلمة اخرى)
strchr( )	ايجاد ترتيب موضع حرف معين داخل كلمة
strcmp( )	مقارنة كلمتين (او متغيرين من نوع حرفى)
strcpy( )	نسخ محتويات متغير حرفى فى متغير حرفى آخر
strlen( )	ايجاد عدد حروف سلسة حرفية
strupr( )	تحويل كلمة من الحروف الصفير الى الحروف الكبيرة

** **بعض دوال للتحويل من متغير رقمى الى حرفى والعكس** :-

الدالة	وظيفتها
atof	تحويل متغير من نوع حرفى الى متغير من نوع رقم حقيقى
Atoi	تحويل متغير من نوع حرفى الى متغير من نوع رقم صحيح
Atol	تحويل متغير من نوع حرفى الى متغير من نوع رقم صحيح طويل
Strtod	تحويل متغير من نوع حرفى الى متغير من نوع رقم حقيق مضاعق
_itoa	تحويل متغير من نوع صحيح الى متغير من نوع حرف
_ltoa	تحويل متغير من نوع رقم طويل الى متغير حرفى

التركيب (المنشات) Structure

معنـاه : التركيب **structure** معناه ان تضع مجموعة من البيانات التى تختلف فى النـــوع معـــاً. بحيث يمكن التعامل معها كوحدة واحدة أو يمكن التعامل مع العناصر المفردة داخلها.

- ومن اهم التطبيقات فى عالم البرامج (تطبيقات قواعد البيانات). فمثلا لكتابة برنـــامج تســـجيل بيانات الموظفين فى الشركة التى تعمل فى هذه الحالة انت تحتاج الى تخزين :

- اسم الموظف وعنوان (سلسلة حروف من نوع **char**)
- سن الموظف (عدد صحيح **int**)
- راتب الموظف (عدد ذو علامة عشرية **float**)

فى هذه الحالة نقوم بعمل تركيب **structure** يضم هذه العناصر سوياً كوحدة واحدة. وبعد ذلك سوف نتمكن من التعامل مع هذه البيانات المختلفة كوحدة واحدة او نتعامل مع كل عنصر فيها على حدة

* الاعلان عن التركيب :-

- عندما ترغب فى الاعلان عن تركيب جديد أتبع الصيغة التالية :-

```
struct اسم التركيب
{
        ----------;
        ----------;        مجموعة البيانات التى يتكون منها التركيب
        ----------;
};
```

وكلمة **struct** من الكلمات الاساسية فى لغة **C**

- إذا رغبت فى عمل تركيب ليحمل بيانات الموظفين :-

```
struct employee
{
    char name[40];
```

```
    char address[40];
    int age;
    float salary;
};
```

- التركيب employee يعتبر (وصفه) نعطيها للمترجم حتى يتعرف على نوع التركيب وحجمه
- عندما ترغب فى استخدام التركيب فعلاً يجب ان تعلن عن متغير من نفس النوع مثل:

```
struct employee  emp1;
```

- أصبح لدينا المتغير emp1 وسيقوم المترجم بحجز الذاكرة له حسب (الوصفة) السابقة وبالتالى سيكون حجم emp1 يساوى 86 byte ويتكون من اربعة عناصر هى :-

* الاسم (40 byte) * العنوان (40 byte) * العمر (2 byte) * المرتب (4 byte)

* استخدام التركيب :-

- للتعامل مع اى عنصر داخل التركيب أذكر اسم التركيب ثم حرف النقطة ثم اسم العنصر داخل التركيب مثلاً : emp1.age
- وإذا أردت ان تضع الرقم 30 فى المتغير age الموجود فى التركيب نكتب التالى :

```
emp1.age=30;
```

- كذلك إذا أردت ان تنقل القيمة الموجودة فى العنصر الى متغير آخر اكتب : a.emp1.salary;
- كذلك يمكن ان تمرر عناصر التركيب إلى إجراءات مكتبة التشغيل كما نفعل مع المتغيرات الأخرى مثل : gets(emp1.name); هنا نستدعى الإجراء gets ليقرأ سلسلة حروف من لوحة المفاتيح ويضعها فى المتغير name الموجود داخل التركيب emp1

* البرنامج التالى يوضح كيفية الإعلان عن التركيبات واستخدامها :-

```
# include <stdio.h>
void main(void)
{
struct employee
      {
            char name[40];
            char address[40];
            int age;
            float salary;
      }

struct employee emp1;
printf(" enter name :");
gets(emp1.name);
printf("enter address: ");
gets(emp1.address);
printf(" enter age : ");
scanf("%d", &emp1.age);
```

```
pruntf("enter salary : " );
scanf("%d",&emp1.salary);
printf("\n\n you entered: \n ";
printf(" employee name :\t%s\n ", emp1.name);
printf(" employee address :\t%s\n ", emp1.address);
printf(" employee age :\t%d\n ", emp1.age);
printf(" employee salary :\t%f\n ", emp1.salary);
}
```

* مصفوفات من التركيبات *

- لكى نعلن عن مصفوفة من التركيبات يجب أن تعلن عن التركيب أولاً ثم نعلن عن المصفوفة
- مثلاً لو أردنا تخزين 100 موظف

```
struct employee
{
      char namr[40];
      char address[40];
      int age;
      float salary;
};
struct employee emp[100]
```

- هنا اعلانا عن التركيب employee ثم اعلنا عن المصفوفة emp التى تتسع لتخزين معلومات عن 100 موظف
- بعد ذلك عند التعامل مع احد التركيبات فى المصفوفة فأنك تذكر رقمه داخل المصفوفة أولاً.
- فمثلاً إذا أردنا ان تدخل بيانات الموظف الخامس فى المصفوفة نكتب emp[4].age=35;
- كذلك emp[count].salary = 123.5 هنا نشير الى رقم التركيب داخل المصفوفة باستخدام المتغير count

* البرنامج الآتي يعطى مثالاً على كيفية إنشاء مصفوفة من التركيبات والتعامل مع العناصر المختلفة فيها :-

```
#inclide<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 10          // عدد الكتب
void main(void)
```

```
{
    struct booktag
    {
        char title[40];
        char author[40];
        int pages;
        int year;
        char edition;
    };
struct bootag book[max];      //مصفوفة الكتب
int c ,c1 , result;
char temp[10];
// أبدأ فى استقبال المعلومات من المستخدم
for(c = 0 ; c < max ; c++)
{
printf(" enter data for record no[%d]\n "m c+1);
printf(" or enter \ " quit \ " to end \n\n";
printf"enter title : " );
gets(boo[c].title);
/*  إذا كتب المستخدم كلمة (خروج) فإننا نخرج من التكرار
result = strcmp (book[c].title,"quit");
if (result = = 0 )
break;
printf("enter author : ");
gets(book[c].author);
printf("enter number of pages : " );
gets(temp);
book[c].pages = atoi(temp);
printf("enter year of publication : " );
gets(temp);
book[c].year = atoi(temp);
printf("enter edition : " );
gets(temp);
book[c].edition = atoi(temp);
}
// اطبع المعلومات
printf("record no. %d\n:, c1++);
printf("book title : \t\%\n ", book[c1].title);
printf(" book author : \t%d\n", book[c1].author);
printf("no of pages :\t%d \n ", book[c1].pages);
printf("date of pub. \t%d\n",book[c1].year);
printf("edition : \t%d\n\n ",book[c1].edition);
}
```

** ملاحظات على البرنامج **

- هذا البرنامج ينشئ مصفوفة باسم book تستخدم لتخزين المعلومات عن الكتب فى مكتبة ما
- البرنامج يتوقف ليسأل المستخدم عن المعلومات الخاصة بكل كتاب ثم تخزن هذه المعلومات فى المصفوفة ثم طباعة هذه المعلومات
- فى بداية البرنامج نعلن عن التركيب booktag الذى سيستخدم لحمل بيانات الكتب
- هذا التركيب يتكون من خمسة عناصر هى :-

  * العنوان * المؤلف * عدد الصفحات * سنة النشر * الطبعة

- بعد ذلك نعلن عن مصفوفة مكونة من 10 تركيبات struct booktag boo[max]
- بعد ذلك نستخدم التكرار for لإدخال المعلومات الخاصة بالكتب من المستخدم ويستمر ذلك حتى يدخل المستخدم معلومات خاصة بـ 10 كتب او يدخل كلمة quit للخروج بدلاً من عنوان الكتاب
- لذلك عندما نطلب من المستخدم غدخال عنوان الكتب نقارن ما ادخله بكلمة quit فغن ادخل هذه الكلمة فإننا نستخدم كلمة break لنخرج من التكرار

```
result =  strcmp(book[c].title , “quit”);
if(result = = 0
break;
```

- فهنا نستدعى الإجراء strcmp ليقارن العنوان بكلمة quit فإذا كانا متطابقين فإن الإجراء strcmp سيعود بالقيمة صفر ويضعها فى المتغير result وبذلك نخرج من التكرار fot

** تمرير التركيب إلى الإجراءا ت**

من الممكن أن نمرر التركيبات الى الإجراءات. والبرنامج التالى يوضح ذلك :-

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct studenttag
{
      char name[40];
      int age;
      int grade;
}
void print(struct stdenttag student);
void main(void)
{
struct studenttag student;
char temp[10];
printf(“enter student namr : “);
gets(student.name)
printf(“enter student age : “ );
gets(temp);
student.age=atoi(temp);
```

```
printf("enter student grade : " );
gets(temp);
student.grade = atoi(temp);
printf("\n\n you entered: \n\n");
// استدعاء الاجراء الذى يطبع محتويات الكتب
printf(student);
}
// الاجراء الذى يطبع محتويات الكتب
void print(struct studenttag student)
{
printf("student name : \t%s\n ", student.name);
printf("student age : \t%d \n ", student.age);
printf("student grade :\t %d\n ", student.grade);
}
```

*** ملاحظات ***

- فى هذا البرنامج نكلب من المستخدم بعض المعلومات عن أحد الطلبة ثم نضعها فى تركيب الى الاجراء **print** ليطبع محتوياته
- فى بداية البرنامج نعلن عن التركيب **studenttag** الذى سيستخدم لتخزين المعلومات .
- وهو يتكون من ثلاثة عناصر :- * اسم الطالب * وعمره * ومرحلته
- لاحظ كيف أعلنا عن الاجراء الفرعى **print** الذى سيطبع محتويات التركيب

```
void print(struct studenttag student)
```

- كذلك كيف نستدعى الإجراء **print(student);**

** تلاحظ انك عندما تمرر تركيباً إلى أحد الاجراءات فإن هذا الإجراء يتلقى (نسخة) من هذا التركيب فقط ولا يتلقى التركيب الأصلي. لذلك إذا غيرت الإجراء فى محتويات التركيب فإنه سيغير فى محتويات النسخة ولن تؤثر ذلك فى التركيب الأصلى.

**** برنامج ****

```
// تمرير التركيبات الى الاجراءات والعودة بتركيب من الاجراء
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

struct sudenttag
  {
      char name[40];
      int age;
      int grade;
  };
```

```
void print(struct studenttag student);

struct studenttag detinfo(void)

void main(void)

{
struct studenttag student;
student = getinfo( );
print(student);
}
```

```
// الاجراء الذى يطبع محتويات التركيب
void print(sturct studenttag student)
{
printf("\n\n you entedred : \n\n ");
printf("student name: \t%s\n",student.name);
printf("sudent age: \t%d\n",student.age);
printf("student grade : \t%d\n",student.grade);
}
```

```
// الاجراء الذى يطلب المعلومات من المستخدم ويضعها فى التركيب
struct studenttag getinfo(void)
{
sturct studenttag student;
char temp[10];
printf("enter student name : " );
gets(student.name);
printf("enter student age : " );
gets(temp);
student.age = atoi)temp);
printf("enter student grade : " );
gets(temp);
student.grade = atoi(temp);
return(student);
}
```

ملاحظات :

- **هذا البرنامج يشبه البرنامج السابق ولكن يزيد عليه أننا وضعنا الاجراء getinfo هــذا الاجــراء يطلب المعلومات من المستخدم ويعود بها الى الاجراءmain**

- لاحظ كيف اعلنا عنه فى بداية البرنامج

```
struct studenttag detinfo(void)
```

- هذا الإعلان معناه أن الإجراء getinfo لا يستقبل اى بيانات وذلك بسبب كلمة void وانه يعود بقيمة إلى من يستدعيه. وهذه القيمة هى تركيب من نوع studenttag
- لاحظ كيف نستدعى getinfo من الاجراء main وهو student = getinfo وهذا الاجراء يطلب المعلومات من المستخدم وعندما ينتهى return(student);
- بهذا الشكل سيعود الاجراء getinfo بالمعلومات التى حصل عليها من المستخدم الى الاجراء main الذى يستقبل ويضعها فى التركيب student وسوف يقوم الاجراء main بعد ذلك بتمرير نفس هذا الترتيب الى الاجراء print ليطبع محتوياته

** تركيب داخل تركيب **

- من الممكن ان تضع تركيباً داخل تركيب آخر . فمثلا إذا كنت تريد وضع تركيب لكل قسم من الأقسام فى الشركة . من الممكن أن تضع التركيب employee فى داخل تركيب القسم لتخزن فيه المعلومات الخاصة بالموظف المسئول عن القسم
- فى هذه الحالة يجب أن تعلن عن التركيب employee أولاً

```
struct employee
{
char name[40];
char address[40];
int age;
float salary;
}
```

- وبعد ذلك نعلن عن التركيب الأخر ونضعه فيه مثل :-

```
struct dept
{
int deno;
int product;
sturct employee emp;
}
```

- لاحظ كيف وضعنا التركيب emp فى داخل التركيب dept
- بعد ذلك يمكن ان نعلن عن التركيب dept كما سبق مثل ;struct dept dept1
- فى حالة التعامل مع عنصر داخل تركيب فرعى اكتب : dept1.emp.age = 25

حيث dept1 اسم التركيب الرئيسى ثم نقطه

emp اسم التركيب الفرعى ثم نقطه

age اسم العنصر داخل التركيب الفرعى

** برنامج **

```
#include<stdio.h>
#include<stdlib.h>
struct employee
{
        char name[40];
        char address[40];
        int age;
        float salary;
};

struct department



{
int depnum;
char product[40];
struct employee emp;
};

void main(void)
{
struct deparment dept;
char temp[10];
printf("enter dept no : " );
gets(temp);
dept.deptnum = atoi(temp);
printf("enter dep. Product : " );
gets(dept.product);
prints("enter employee name : " );
gets(dept.emp.name);
printf(enter employee address" );
gets(dept.emp.address);
printf("enter employee age : " );
gets(temp);
dept.emp.age = atoi(temp);
printf("enter employee salary : " );
gets(temp);
dept.emp.salary = atof(temp)
printf("\n\n you entered \n\n " );
printf("dept no : \t\t%d\n ",dept.deptnum);
printf("dept product : \t\t%s\n ",dept.product);
printf("employee address : \t\t%d\n", dept.emp.address);
```

```
printf("employee age : \t\t%d\n",dept.emp.age);
printf("employee salary: \t\t%f",dept.emp.salary);
}
```

ملاحظات :-

- فى بداية البرنامج نعلن عن التركيب employee أولا
- ثم نعلن عن التركيب department ونضع داخله التركيب employee
- لاحظ كيف نتعامل مع عناصر التركيب الفرعى فعندما نطلب من المستخدم إدخال اسم الموظف ، فإننا نستخدم الإجراء gets ونمرر له مصفوفه الحروف name الموجودة فى التركيب الفرعى emp كمايلى : gets(dept.emp.name);
- بعد ان ينتهى المستخدم من ادخال جميع المعلومات نستخدم الاجراء printf لطبع المعلومات على الشاشة.

** تنويعات فى الإعلان عن التركيبات **

* فى جميع الأمثلة السابقة ذكرنا خطوتين للإعلان عن التركيب :-

- **الأولى** : ان نعلن عن التركيب نفسه مثل:-

```
struct enployee
{
char name[40];
char address[40];
int age;
float salary;
};
```

- **الثانية** : ان نعلن عن متغير عن هذا التركيب مثل :-

```
struct employee emp;
```

* **ملاحظة** :- يمكن دمج الخطوتين السابقتين معاً . فتعلن عن بنيه التركيب وعن متغير فيه فى خطوة واحدة مثل :-

```
struct employee
{
char name[40];
char address[40];
int age;
float salary;
} emp;
```

- لاحظ كيف تم وضع كلمة emp بين القوس { وبين الفاصلة المنقوطة وسيكون لها نفس تأثير العبارة struct employee emp;
- كذلك يمكن استخدام نفس الطريقة السابقة عن عدة تركيبات من نفس النوع

```
struct employee
{
```

```
char name[40];
char address[40];
int age;
float salary;
} emp1 , emp2 , emp3;
```

- يمكن الاعلان عن تركيب واحد أو عدة تركيبات مع الاستغناء عن اسم التركيب

```
struct
{
char name[40];
char address[40];
int age;
float salary;
} emp;
```

- كذلك يمكن ان تضع المعلومات فى التركيب عند الاعلان عنه مباشرة مثل :-

```
struct
{
char name[40];
char address[40];
int age;
float salary;
} emp = {
            "ali ahmed"
            "26 st. cairo "
            39
            135.6
        };
```

المؤشرات pointer

***معنى المؤشر pointer : -**

- **هو نوع من انواع البيانات . ويعرف بانه متغير يحتفظ (يخزن به) بعنوان مكان فى الذاكرة . أى هو(متغير يُستخدم ليحمل عنوان متغير آخر فى الذاكرة)**
- **اى أن المؤشر يشير إلى مكان فى الذاكرة.**
- **من المعلوم أن كل مكان فى الذاكرة له عنوان والجهاز يتعامل مع هذا المكان بالعنوان المحدد لـــه ونحن بطريقه غير مباشرة نتعامل مع هذا العنوان .**
- **فمثلا فى هذا الاعلان ; 5 = int d معناه احجز مكان فى الذاكرة ram حجمه 2 byte واجعل اسمه d وضع فيه القيمة 5**
- **عندما ترغب فى استخدام المؤشرات هناك خطوتان :-**
  - **الأولى : أن تعلن عن هذا المؤشر وتخبر المترجم عن اسمه وعن نوعيه البيانات التى تشـــير إليها.**
  - **الثانية : تهيئ هذا المؤشر بأن تجعله يشير إلى احد المتغيرات الفعلية**

**** الإعلان عن المؤشر ****

- **عندما ترغب فى استخدام المؤشر يجب ان تعلن عنه اولاً كما نفعل مع بقية المتغيرات (أي يـــتم الإعلان عن مؤشر إلى اى متغير من انواع البيانات بنفس الطريقة التى تعلن بها عـــن البيانـــات العادية)**
- **وعندما ترغب فى الإعلان عن مؤشر يجب ان نحدد شيئيين :-**

  الأول : هو اسم المؤشر ويتم اختيارة كما نختار المتغيرات العادية

  الثانية : هو نوعية البيانات التى سيشير إليها

```
int * p;
```

- هــذا الإعلان يخبر المترجــم أن المتغير p أصبح مؤشراص الى مساحة فى الذاكرة مقــدارها 2 byte مع الاحتفاظ بعنوان هذا المكان فى المتغير p

float * degree;

- هذا الاعلان يخبر المترجم أننا نريد استخدام المؤشر degress وان هذا المؤشر سيشير إلى بيانات من نوع float

- وإذا كنت ستستخدم المؤشر ليشير إلى تركيب يجب أن تعلن عن التركيب أولاً . مثل :

```
struct student
{
char name[40];
int mark;
}
```

ثم نعلن عن المؤشر بعد ذلك . مثل :

```
struct student *st;
```

**** تهيئة المؤشر ****

- بعد أن نعلن عن المؤشر يتم تهيئته اى نجعله يشير إلى عنوان فى الذاكرة فعلاً ويتم ذلك باستخدام معامل العنوان & هذا المعامل يستخدم للحصول على عنوان أى متغير فى الذاكرة الجهاز
- هذا المعامل من المعاملات الأحادية unary وعندما نضعه قبل احد المتغيرات فإنه يعنى عنــوان هذا المتغير فى الذاكرة مثلاً: scanf("%d",&var);
- مثال على تهيئة المتغير وجعله يشير إلى عنوان فى الذاكرة

```
int num;
int *ptr;
num = 5;
ptr = &num;
```

- فى هذه العبارات نعلن عن المتغير العددى num ثم نعلن عن المتغيــر ptr الــذى سنســتخدمه كمؤشر ليشير إلى عنوان المتغير num فى الذاكرة
- بعد ذلك نهيئ المتغير num ونضع فيه الرقم 5
- أما العبارة ptr = &num فاننا نستخدم المعامل &لنحصل على عنوان المتغيــر num فــى الذاكرة ونضع هذا العنوان فى المؤشر ptr
- بهذا الشكل يصبح المؤشر ptr حاملاً لعنوان المتغير num أى انه (يشير) إلى هذا المتغير.

*** * استخدام المؤشر * * هناك طريقتان لاستخدامه :-**

الطريقة الأولى : هى ان تستخدم العنوان الموجود فى المؤشر .

- **فى هذه الحالة نستخدم اسم المؤشر بدون إضافات.**
- **فمثلا إذا أردت ان نمرر العنوان إلى إجراء آخر نكتب : addten(ptr);**
- **هنا نستدعى الإجراء addten ونمرر له المؤشر ptr اى عنوان المتغير num فى الذاكرة**

الطريقة الثانية : هى ان تستخدمه لتعديل قيمة المتغير

- **فى هذه الحالة نضع حرف (*) قبله مثلاً *ptr = 10 ;**
- **فى هذه الحالة نعدّل قيمة المتغير num إلى 10 ولكن بطريقة غير مباشرة**
- **كذلك يمكن ان نستخدم المؤشر لقراءة محتويات العنوان الذى يشير إليه المؤشر نكتب a=*ptr;**

*** تمرين :**

```
#include<stdio.h>
void main(void)
{
int u = 3;
int v;
int *pu;          /* pointer to an integer */
int  *pv          /* pointer to an integer */

pu = &u;          /* assing address of u to pu */
v = pu;           /* assign values of u to v */
pv =&v;           /* assing address of v to pv */

printf("\n u=%d   &u=%x   pu=%x   *pu=%d" , u , &u , pu , *pu);
printf("\n v=%d   &v=%x   pv=%x   *pv=%d" , v , &v , pv , *pv);
}
```

ملاحظات :

- **لاحظ ان pu مشير إلى u ,pv مشير إلى v. وعلى هذا يمثل عنوان u ويمثل pv عنوان v**
- **فى السطر الأول u تمثل القيمة 3 ويتحدد عنوان u تلقائيا بواسطة المترجم بأنه مثلا FF8E (فى النظام السادس عشر)**
- **وتتحدد هذه القيمة للمشير pu . لذا يمثل pu العنوان (السادس عشر) FF8E**
- **كذلك القيمة التى يشير إليها pu(اى القيمة المخزنة فى خلية فى الذاكرة التى عنوانها FF8E) تعنى 3**
- **وبالمثل فى السكر الثانى يبين أن v تمثل القيمة 3 حيث اننا حددنا القيمة pv* الى v**

*** * تمرين آخر :**

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

```
main( )
{
int u1 ,u2;
int v =3;
int pv                / pv pointer to v */
u1 = 2*(v + 5 );    /* ordinary expression */
pv = &v ;
u2 = 2*(*pv + 5 );
printf("\n u1 = %d      us2 = %d ",u1 , u2 );
}
```

ملاحظات :

- يشمل هذا البرنامج تعبيرين صحيحين اولهما (2*(v + 5 وهو تعبير حسابى معتاد . اما الثانى (2*(*pv + 5 يحتوى على استخدام مشير ويتكافئ التعبيران حيث أن كلا من *pv , v يمثل نفس القيمة الصحيحة
- ناتج البرنامج :

v = 3	u1 = 15	u2 = 16	pv = 3
u1=2*(v + 5)		u2 = 2*(*pv + 5)	
u1=2*(3+5)		u2=2*(3+5)	
u1= 16		u2=16	

** تمرير المؤشرات إلى الاجراءات الفرعية **

- من المعلوم أن لغة C تمرر المتغيرات إلى الإجراءات بالقيمة وليس بالاشارة.
- اى عندما تمرر متغيراً لأحد الإجراءات فإن هذا الإجراء يتلقى نسخة من المتغير وليس المتغير نفسه (أى يتلقى " قيمة " المتغير وليس المتغير نفسه) كما فى الشكل التالى :-

```
void main(void)
{
int num = 5;
change(num);
……;
}

void change( int num )
{
num = 7 ;
……;
}
```

5

هذا التعديل لا يؤثر فى المتغير الأصلى

ملاحظات :

- فى الإجراء main نعلن عن المتغير num ونضع فيه العدد 5 ثم نستدعى الإجـراء change ونمرر له هذا المتغير.
- فى هذه الحالة يقوم المترجم بتمرير القيمة الموجودة فى المتغير num وهى العدد 5 إلى الاجراء change ولن يتمكن الإجراء change من رؤية المتغير num الموجود فى main
- فرغم ان المتغير num موجود فى الإجراء main وفى الإجـراء change إلا ان كـل واحـد منهما يختلف عن الاخر، فكل واحد منهما متغير محلى local خاص بالإجراء الذى يوجد فيه.
- ملاحظة : (تمرير المتغيرات بالقيمة وليس بالإشارة مسالة مفيدة لنها تحمى المتغيرات المحليـة لكل إجراء من اى تعديل غير مقصود)

* إذا أردت أن تتيح للإجراء الفرعى أن يعدل المتغير الذى تمرره له . أتبع الاتى :-

الخطوة الأولى : -

- هى أن تعدل الاعلان عن الإجراء فى بداية البرنامج ورأس الاجراء نفسه بحيث تخبر المترجم ان سيتلقى مؤشراً وليس متغيراً عادياً.
- فمثلاً الاعلان عن الاجراء التالى : void change (int * ptr);
- هنا نخبر المترجم ان الإجراء change سيتلقى مؤشراً وليس متغيراً عادياً

الخطوة الثانية :

- هى ان تمرر للإجراء عنواناً فى الذاكرة مثل : change(ptr) على اعتبار ptr هو مؤشر لأحـد المتغيرات
- فى هذه الحالة سيكون لدى الاجراء الفرعى عنوان المتغير ويتمكن بالتالى من تعديل قيمته.

البرنامج التالى يوضح ذلك :

```
#include<stdio.d>
void change( int *ptr );

void main(void)
{
int num = 5;
int *ptr;

ptr = &num ;
printf( “berfore change , num =%d \n “, num);
change(ptr);
printf(“after change, num = %d \n “, num );
}

void change( int * ptr )
```

```
{
   *ptr = 7;
}
```

ملاحظات

- فى بداية البرنامج نعلن عن الاجراء void change(int *ptr);
- هذا الاعلان يخبر المترجم ان الاجراء change يستقبل مؤشراً وليس متغيراً
- والسبب فى ذلك هو الحرف (*) الذى يسبق اسم المؤشر كذلك نفس الصيغة فى رأس الاجراء .
- وفى الاجراء main نعلن عن المتغير num ونضع فيه العدد 5 ثم نعلن عن المتغير ptr وننقل إليه عنوان المتغير num فى الذاكرة مثل : ptr = &num
- بهذا الشكل يصبح المتغير ptr حاملاً لعنوان المتغير فى الذاكرة
- بعد ذلك نستخدم الاجراء peintf لنطبع قيمة المتغير num قبل استدعاء الإجراء change ثم نستدعى الاجراء change ونمرر له المتغير ptr
- الاجراء change يحتوى على ;7 = ptr* هذه العبارة تستخدم المؤشر ptr لتضع العدد 7 فى العنوان الذى يشير إليه.
- ولما كان المتغير ptr يشير إلى num فإن هذه العبارة تغير محتويات المتغير num الى العدد
- ويكون ناتج البرنامج :-

```
before change , num = 5
after change , num = 7
```

**** العودة بأكثر من قيمة من الإجراء الفرعى ****

- ملاحظة : سبق إن ذكرنا أن استخدام كلمة return للعودة بقيمة ما من الاجراءات الفرعية ولكن كلمة return تعود بقيمة واحدة فقط
- الحل : هو ان تعلن عن متغيرات محلية بعدد المعلومات التى تريد ان يعود بها الاجراء ثم تمرر مؤشرات لهذه المتغيرات للاجراء ليضع المعلومات فيها
- البرنامج التالى يوضح ذلك :-

```
#include<stdio.h>
void fn( int *var1 , int *var2 , int *var3);
void main(void)
{
int var1 ,var2, var3;
fn(&var1 , &var2 , &var3)
printf("var1 = %d , var2 = %d , var3 = %d", var1 , var2 , var3);
}

void fn( int *p1 , int *p2 , int *p3)
{
*p1=10;
```

```
*p2=20;
*p3=30;
}
```

ملاحظات :

- فى بداية البرنامج نعلن عن ثلاثة متغيرات محلية var1,var2,var3 ولا نضع فيها اى قيمة
- بعد ذلك نستدعى الاجراء fn ونمرر له عناوين هذه المتغيرات الثلاثة
- الاجراء fn يستقبل هذه العناوين ويضعها فى مؤشرات باسم p1,p2,p3
- بعد ذلك يقوم الاجراء fn بتعيين هذه للعناوين التى تشير إليها هذه المؤشرات
- وعندما نعود الى الاجراء main نستخدم الاجراء printf لنطبع القيمة الموجودة فى المتغيرات var1,var2,var3
- نلاحظ انها نفس القيمة التى وضعها الاجراء fn فى العناوين التى مررناها له ويكون الناتج :-

  var1 = 10, var2 = 20 , var3 = 30

  بهذا الشكل يمكنك استقبال اى عدد من البيانات من الاجراءات الفرعية

**** برنامج عام ** (تجديد سجلات العملاء)**

- اعد نظاماً بسيطاً لإعداد فواتير عملاء فى هذا النظام من خلال تخزين سجلات العملاء داخل منظومة من الهياكل، ويخزن كل سجل كهيكل منفرد (اى كعنصر منظومة) محتوياً على اسم العميل ، واسم الشارع الذى يسكن فيه ، واسم المدينة ، ورقم الحساب ، وحالة الحساب (جارٍ أو متاخر السداد overdue ، أو مماطل delinquent) ، والموازنة السابقة ، والمبلغ المدفوع حالياً ، والموازنة الجديدة ، وتاريخ الدفع.
- الطريقة : هى إدخال سجل كل عميل ، وتجديده بمجرد إدخاله ليعكس المبالغ المدفوعة حاليا وبعد ذلك تعرض كل السجلات المجددة مع الحالة الحالية لكل حساب والتى تعتمد على قيمة المبلغ المدفوع بالنسبة للموازنة السابقة للعميل.

*** توضيح :-**

١- إذا كان المبلغ المدفوع – حالياً – أكبر من صفر – لكنه أقل من 10% من الموازنة القائمة السابقة ، حيث يكون الحساب متأخر السداد overdue

٢- إذا كانت هناك موزانة قائمة ، وكان المبلغ المدفوع حالياً صفراً ، فإن الحساب يكون مماطلاً delinquent وإلا كان الحساب جارياً

*** ويكون البرنامج كما يلى :-**

١- حدد عدد حسابات العملاء (أى عدد الهياكل) المراد تشغيلها .

٢- لكل حساب ... تقرأ العناصر التالية :-

1- name	2-street	3- city
4- account number	5- previous balance	6- current balance

7- payment date

٣ - مع قراءة سجل كل عميل ، يتم تجديده طبقاً لما ياتى :-

- يقارن المبلغ المدفوع حالياً مع الموزانة السابقة، وذلك لتحديد الحالة المناسبة للحساب
- تحسب موزانة الحساب الجديد بطرح المبلغ المدفوع – حالياً – من الموزانة السابقة

٤ - بعد ادخال كل سجلات العملاء وتشغيلها ، نكتب المعلومات التالية كمخرجات لكل عميل:

1- name	2- account number	3- street
4- city	5- old balance	6- current balance
7- new balance	8- account status	

* فيما يلى محتويات البرنامج :-

```
#include<stdio.h>
struct date
        {
        int month;
        int day;
        int year;
        };
struct account
        {
        char name[50];
        char street[45];
        char city[60];
        int acc_no;
        int acct_type;      /* c = current , o = overdue , d = delinquent */
        float oldbalance;
        float newbalance;
        float paymeny;
        struct date lastpayment;
        }

main(  )
{
int i ,n ;

void readinput(int i );
void writeoutput(int i (;
printf("customer billing system \n\n");
printf("how many customer are there ? " );
scanf("%d",&n);
for(i=0,i<n,i++)
{
```

```
 readinput(i);
 if(customer[i].payment > 0 )
        customer[i].acct_type =
        (customer[i].oldbalance < 0.1*customer[i].oldbalabce) ? 'o' | 'c';
else
        customer[i].acct_type =
        (customer[i].oldbalance > 0 ) ? 'd' | 'c' ;
};
for (i = 0 ; i < n ; i ++)
writeoutput(i);
}
void readinput(int i )
{
print("\n customer no. %d\n", i+1);
printf(" name : " );
sccanf(" %[   ] ",customer[ i ].name);
printf(" street : " ) ;
scanf("% [    ] ",customer[ i ] .street);
printf(" city : " );
scanf(" %[   ] ",customer[ i ] .city );
printf(" account number : " );
scanf(%d " , &customer[ i ] . oldbalance);
printf( current payment : ");
scanf("%f", &customer[ i ] .payment);
printf(" payment date (mm/dd/yyyy) : ");
scanf("%d / %d / %d ", &customer[ i ] .lastpayment.month,
                       &customer[ i ] .lastpayment.day
                       &customer[ i ] .lastpayment.year
return;
}
void writeoutput(int i)
{
printf(" name : %s", customer[i].name);
printf(" account number : %d\n", customer[i].acct_no);
printf("street : %s\n , customer[i].street);
printf("city : %s\n\n" , customer[i].city);
printf("old balance " %7.2f", customer[i].oldbalance);
printf(" current payment : %7.2f" , customer[i].payment);
printf(" new balance : %7.2f\n\ ",customer[i].newblance);
printf(" account status : ");
switch ( customer[i].acct_type)
{
case 'c'
        printf(" current\n\n");
        break;
```

```
case ‘o’
        printf(“ overdue\n\n”);
        break;
case ‘d’
        printf(“ delinqunt\n\n”);
        break;
default;
        printf(“ error \n\n”)
}
return;
}
```