9.1 مقدمة

كلما تقدم المبرمج في مجال البرمجة وجد الحاجة أكثر لتنظيم برنامجه و تنسيقه، بحيث تسهل عملية إعداده وتعديله واستيعابه، فالتطبيق الواحد قد ينقسم إلى مجموعة مهام، وقد تتكرر المهمة الواحدة في اكثر من جزء من التطبيق .
من هنا جاءت الحاجة إلى مفهوم (البرنامج الجزئي)في البرمجة بصورة عامة ويطلق عليها في لغة سي (الدوال) ، فالدالة هي فرع من البرنامج العام يقوم بمهمة معينة كلما استدعي الأمر ذلك.

وبهذه الطريقة يقتصد المبرمج الكثير من التكرار في جمل البرنامج، ويصبح برنامجه أكثر قابلية للقراءة والمتابعة والتعديل .

9.2 الدالة من النوع الفارغ void

لنظر إلى الشكل (9.2.1) حيث يحتوي على برنامج رئيسي ( )main متكون من جملتين فقط هما :

```
void  welcome( ) ;
welcome( )  ;
```

الجملة الأولى هي عبارة عن إعلان declaration عن أن الدالة welcome من النوع الفارغ void ، أي أن هذه الدالة ترجع لنا قيمة فارغة void (أي لاشئ) . أما الجملة الثانية فهي عبارة عن استدعاء call للدالة welcome حيث يتم هذا الاستدعاء على الصورة

```
welcome ( ) ;
```

ومعنى الاستدعاء هو أن يتحول المصرّف سي إلى هذه الدالة ، وينفذ ما بها من أوامر ، ثم يرجع إلى البرنامج المستدعى وهو في هذه الحالة الدالة main.

لننظر الآن إلى الدالة welcome نفسها حيث نجد

```
void  welcome ( )
   {
    printf ( " \n  welcome to the function lesson " ) ;
   }
```

تتكون هذه الدالة من الاسم والنوع (بدون فاصلة منقوطة)

```
void   welcome ( )
```

يلي ذلك جسم الدالة function body ، وهو عبارة عن جمل الدالة محصورة بين القوسين { } ، وفى هذا المثال لا توجد إلا جملة واحدة هي جملة طباعة النضيد :

```
" welcome to the function lesson "
```

أن استدعاء الدالة welcome سينتج عنه طباعة هذا النضيد .

ملاحظة :

الأقواس ( ) الموضوعة بعد اسم الدالة ضرورية رغم أنها فارغة ولا تحتوى على شئ. سوف ندرس بعد قليل كيف نضع بين هذه الأقواس بارامترات (متغيرات) كوسيلة لتمرير بعض القيم بين الدالة المستدعية والدالة المستدعاة .

```
main()
{
      void welcome();
      welcome();
}

void welcome()
{
 printf("\n Welcome to the function lesson.");
}
```

الشكل(9.2.1) برنامج يحتوي على دالة

9.3 المتغير العام global variable

المتغير العام هو المتغير المشترك بين جميع الدوال ، أي أنه متغير يتم تعريفه خارج الدوال في بداية البرنامج . فمثلا المتغير الحرفي name في الشكل التالي يعتبر متغيراً عاماً .

```
char name [20] ;
main( )
  {
    .....
  }
void getname( )
  {
    .....
  }
```

في هذه الحالة يمكن استخدام المتغير name في الدالة main أو الدالة getname على حد سواء ، ويكون له نفس المدلول في الدالتين أو أي دالة أخرى إن وجدت بنفس الملف .

مثال (9.3.1) : اكتب برنامجا يتكون من 3 دوال (إلى جانب الدالة main) وهي :

1 . الدالة welcome لطباعة النضيد :

" Welcome to the function lesson "

2. الدالة getname لقراءة اسم لمستخدم بعد سؤاله على النحو :

" What is your name ?"

3. الدالة thanks لطباعة النضيد :

" Thank you Mr ..... for using the function lesson "

حيث يطبع اسم المستخدم في الفراغ المبين .

بين الشكل (9.3.1) البرنامج المطلوب ، حيث نلاحظ أن النضيد name هو متغير عام global تم تعريفه قبل الدالة main ، وبذلك يمكننا استخدامه في أي دالة من الدوال الثلاث ، وبالتحديد تم استخدامه في الدالتين getname والدالة thanks .

أما الدالة الرئيسية main فتحتوى على تعريف الدوال الثلاث ، وهى هنا من النوع الفارغ void ، ثم استدعاء الدالة getname لقراءة الاسم ، ثم استدعاء الدالة thanks لشكر المستخدم مع طباعة اسمه .

```
char name[20];
main()
{
        void welcome();
        void getname();
        void thanks();
        welcome();
        getname();
        thanks();
```

```
}
void welcome()
{
    printf("\n Welcome to the function lesson.");
}

void getname()
{
    printf("What is your name? ");
    gets(name);
}

void thanks()
{
    printf("\n Thank you Mr. %s for using the function
lesson.",name);
}
```

الشكل (9.3.1) برنامج يتكون من دالة رئيسية و 3 دوال أخرى

9.4 المتغير المحلى local variable

رأينا أن المتغير العام يتم إعلانه خارج الدوال ، فماذا يحدث إذا تم تعريفه داخل دالة ما؟

إذا تم إعلان متغير داخل إحدى الدوال فإنه يعتبر متغيراً محلياً خاصاً بتلك الدالة ، ولا علاقة له بالمتغير المعلن في دالة أخرى حتى لو كان يحمل نفس الاسم .

يبين الشكل (9.4.1) أحد الأخطاء الشائعة في البرمجة ، وهو الخلط بين متغير محلي لدالة ما وبين متغير محلي لدالة أخرى. فمثلا لو عرفنا المتغير k في الدالة الرئيسية main على الصورة :

```
main( )
 { int k = 5 ;
   void fun ( ) ;
        ........
 }
```

فإن k في هذه الحالة هو متغير محلى خاص بالدالة الرئيسية ، وإذا حاولنا طباعة قيمة k في دالة أخرى مثل :

```
void  fun ( )
 {  int  k ;
    printf ( "\n %d " , k ) ;
 }
```

فسيعطي المصرف سي إنذارا بأن k غير معروفة القيمة في الدالة ( ) fun رغم أنها قد تم تحديدها في الدالة الرئيسة .

```
main()
{
    int k=5;
    void fun();
}
void fun()
{
    int k;
    printf("\n %d",k);
}
```

الشكل (9.4.1) مثال لخطأ شائع في استخدام المتغيرات المحلية

أي أن المتغير المحلى يؤدى عملاً خاصاً بالدالة الوارد بها ولا علاقة له بالدوال الأخرى.

مثال (9.4.2) : ماذا يطبع البرنامج المبين بالشكل 9.4.2؟

لدينا في هذا البرنامج متغير عام هو y ، ومتغير محلى في الدالة main هو x . أما الدالة f1 فيوجد بها متغيران محليان هما x , y . يجب أن نلاحظ هنا أن تغيير قيمة x في الدالة f1 لم يؤثر على قيمة كل منهما ، لأن x المتغير المحلى في الدالة f1 لا علاقة له بالمتغير x في الدالة main . أما المتغير y فهو متغير عام لا يؤثر عليه متغير بنفس الاسم في دالة أخرى .
وبالتالي فإن ناتج تنفيذ البرنامج (9.4.1) هو

1.300000 2.500000

أي أن الاستدعاء ( ) f1 لم يؤثر على قيم x و y في الدالة main .
وبصفة عامة يجب استخدام المتغيرات العامة بحذر لأنها قد تتضارب مع متغيرات محلية بنفس الاسم في أحد دوال البرنامج .

```
float y=1.3;
main()
{       float x=2.5;
        void f1();
        f1();
        printf("\n %f %f ",x,y);
}
void f1()
{
        float x=3.4, y=5.6;
}
```

الشكل(9.4.2) المتغير المحلى

9.5 تمرير القيم إلى الدالة

نأتي الآن إلى السؤال : لماذا نضع القوسين ( ) أمام اسم الدالة ؟
والجواب : أنه عادة ما يكون للدالة بارا مترات (متغيرات) تعتمد عليها في طريقة عملها . فمثلاً الدالة الرياضية :

$$f(x) = x^2 + 3x - 2$$

تعتمد في قيمتها على المتغير x . فإذا أعطينا x القيمة 4 على سبيل المثال فإن الدالة تأخذ القيمة :

$$f(4) = 16 + 12 - 2 = 26$$

في لغة سي ، يجب تحديد نوع بارا مترات الدالة وكذلك الدالة نفسها . فمثلا التحديد

```
float  f(float x)  ;
```

يعني أن الدالة f من النوع العائم float ، وأنها ذات بارا متر واحد من النوع العائم أيضاً .

لاحظ أن الدالة يمكن أن يكون لها أكثر من متغير واحد ، ولكنها تقوم بحساب قيمة واحدة في اسمها ، وتقوم بترجيعها عن طريق الأمر return إلى الدالة المستدعية ، كما في المثال التالي :

مثال (9.5.1) : ماذا يطبع البرنامج التالي ؟

عند تنفيذ هذا البرنامج سيطبع مربعات الأعداد من 1 إلى 10 على النحو التالي :

1	1
2	4
3	9
4	16
...	...
10	100

```
main()
{
        int x,y;
        int square(int  x);
        for(x=1; x<=10; x++)
        {
                y=square(x);
                printf("\n %d  %d ",x,y);
        }
}
int square(int k)
{       int z;
        z=k*k;
        return(z);
}
```

الشكل (9.5.1) دالة ذات متغير واحد

من الأخطاء الشائعة التي تحدث عند تمرير قيمة بارا متر هو عدم توافق الأنواع ، كأن يستخدم الاستدعاء

y = square(2.5) ;

بينما الدالة square معرّفة على أنها ذات متغير من النوع الصحيح . ويحدث هذا الخطأ بصورة خاصة عند وجود العديد من البارامترات .

لاحظ أيضاً عدم ضرورة تطابق الأسماء . فمثلاً استخدامنا في الدالة square المتغير k ، بينما يقابله المتغير x في الاستدعاء . وما يحدث هنا هو أن قيمة المتغير x تتحول إلى المتغير k في الدالة square ، ولهذا يسمى الاستدعاء

هنا الاستدعاء بالقيمة call - by - value ، وفيه تنتقل القيمة من الدالة المستدعية calling function إلى الدالة المستدعاة called function عبر بارا متر الدالة .

مثال (9.5.2) : اكتب دالة تقوم باستقبال قيمتين من النوع float وترجع أكبرهما ، واستخدمها لحساب أكبر قيمة من بين 10 قيم موجبة .

في كتابة هذا البرنامج نستخدم الخوارزمية التالية :

1 . ابدأ بأكبر قيمة y تساوى الصفر (لأن الأعداد المدخلة لا تقل عن الصفر) .

2 . اقرأ قيمة x .

3 . استخدم دالة max لحساب القيمة الأكبر من بين y و x وعيّنها للمتغير y

4 . الرجوع إلى الخطوة (2)عشر مرات .

دعنا نستخدم في هذا البرنامج الدالة max بالصورة :

```
float   max ( float , float )
```

حيث لم نحدد أسماء المتغيرات لأن وضع الأسماء عند إعلان الدالة غير ضروري ولكن الضروري هو تحديد نوعها .

لاحظ أيضاً أن الدالة max تقوم بترجيع return قيمة واحدة وهى إما x أو y بناء على أيهما أكبر وذلك باستخدام جملة if على الصورة :

```
    if ( x > y )
        return (x)  ;
    else
        return (y)  ;
```

```
main()
{   float x, y=0;
    int i;
    float max(float, float);
    for(i=1; i<=10; i++)
    {
        printf("\n enter value%d-->",i);
        scanf("\n %f",&x);
        y=max(x,y);
    }
    printf("\n The maximum value is %f",y);
}
float max( float x, float y )
{
    if( x>y )
        return(x);
    else
        return(y);
}
```

الشكل (9.5.2) دالة ذات متغيرين

9.6 استخدم الماكرو Macro

إذا كانت الدالة بسيطة التركيب (مثل الدالة max في البرنـــامج 9.5.1) يمكننـا تعريفها بما يعرف بالماكرو macro وذلك باستخدام التوجيه

define

فمثلاً إذا عرّفنا الدالة f على الصورة التالية :

```
# define    f(x)    2* x+1
```

يمكننا استخدام الدالة f(x) بدلا من 2 *1 + x على النحو التالي :

```
# define    f(x)    2* x+1
main( )
 { float  x = 2.5 , y ;
   y = f (x) ;
   printf ( "\n  x = %f , y = %f " , x , y ) ;
 }
```

عند تنفيذ هذا البرنامج نحصل على النتائج

x = 2.500000 y = 6.000000

مثال (9.6.1) : ما هو ناتج تنفيذ البرنامج التالي؟ :

```
#define F(X)   5*X*X+1
main()
{  int i;
   float x;
   printf("\n\n");
   for(i=1;i<=10;i++)
   {
     x= i*0.1;
     printf("\n %5.2f %5.2f ",x, F(x) );
   }
}
```

الناتج هو :

0.10	1.05
0.20	1.20
0.30	1.45
0.40	1.80
0.50	2.25
0.60	2.80
0.70	3.45
0.80	4.20
0.90	5.05
1.00	6.00

هناك استفادة أخرى من الماكرو (إلى جانب استخدامه بما يشبه الدالة) وهي اختصار بعض الجمل التي كثيراً ما ترد في لغة سي . فمثلاً يمكن استخدام الدالة :

```
readf (x)   ;
```

بدلا من :

```
scanf ( " %f " , &x ) ;
```

إذا قمنا بتعريف الماكرو التالي :

```
# define    readf (x)    scanf ( " %f " , &x ) ;
```

وإذا لم يكف سطر واحد لتعريف الماكرو يمكنك استخدام الرمز \ (أي backslash) للاستمرارية ،كما في التوجيه التالي

```
# define    PR (x)   \
            printf ( "\n %f " , x ) ;
```

الذي يجعل الجملة :

```
PR (x)   ;
```

لها نفس مفعول الجملة :

```
printf ( " \n %f " , x )  ;
```

9.7 المصفوفة كمتغير لدالة

هل يجوز أن نبعث بمصفوفة كاملة إلى دالة ما ؟ أم أنه لا بد من أن نرسل لعناصر واحداً تلو الآخر لهذه الدالة ؟

كلا الأمرين جائز في لغة سي (وفى معظم اللغات الأخرى) . فمثلاً إذا أعلنا لدالة max على الصورة :

```
float  max ( int n  ,  float x[ ] ) ;
```

فمن الواضح هنا أن المتغير x ليس متغيراً عادياً بل هو مصفوفة . وعند استدعاء هذه الدالة نستخدم اسم المصفوفة فقط (بدون أقواس) على النحو التالي مثلاً :

```
y = max ( 9 , x ) ;
```

حيث العدد 9 هو قيمة n .

مثال (9.7.1) : اكتب الدالة max من النوع float التي توجد أكبر عنصر لمصفوفة x التي تتكون من n + 1 عنصر من النوع float . واستخدم هذه الدالة لإيجاد وطباعة أكبر عنصر لمصفوفة تتكون من 10 عناصر .

ليكن الشكل (9.7.1) البرنامج المطلوب ، وفى هذا البرنامج يجب أن نلاحظ الآتي :

1 . يطلب البرنامج إدخال 10 قيم على النحو التالي (كمثال):

```
enter    x[0]   →    34
enter    x[1]   →    52
enter    x[2]   →    60
....     ....   ....
enter    x[9]   →    45
```

2 . بعد تكوين المصفوفة ، نم استدعاء الدالة max على النحو :

```
y = max ( n , m ) ;
```

حيث وضعنا اسم المصفوفة بدون الأقواس [ ] في المكان المناظر في تعريف الدالة . ولكن وضع الأقواس أمام اسم المصفوفة ضروري في تعريف الدالة حتى نبين أن هذا المتغير هو رمز لمصفوفة وليس متغيراً عاديا .

3 . المتغير i والمتغير y في الدالة main متغيران محليان ، وليس لهما علاقة بالمتغيرين y و i في الدالة max ، ولذلك يجب تعريفهما في الدالتين .

```
main()
{
    float x[10], y;        int i, n=9;
    float max (int n, float x[]);
    for(i=0; i<n; i++)
```

```
        {       printf("\n enter x[%d]-->",i); scanf("%f",&x[i]);
        }
        y=max(n,x);
        printf("\n maximum = %f",y);
}
float max( int n, float x[])
{   int i;      float y=x[0];
        for(i=1; i<n; i++)  if( x[i] > y ) y=x[i];
        return(y);
}
```

الشكل (9.7.1)

9.8 تمرير قيم من الدالة

عرفنا كيف نحصل على قيمة واحدة من الدالة ، ولكن ماذا لو نريد أن نحصل منها على أكثر من قيمة ؟

مثلاً نريد من الدالة أن تحسب لنا متوسط درجات مجموعة من الطلبة وعدد الذين تحصلوا على درجة أكبر من المتوسط .

لنطلق على هذه الدالة اسم ave . لاحظ أولا أن إعلان هذه الدالة يتم على النحو التالي :

```
float  ave ( float g[ ] , int  n , int  *kp )
```

حيث

g[ ] مصفوفة الدرجات التي نمررها للدالة .

n عدد الطلبة الذي نمرره أيضاً للدالة.
kp مؤشرلعدد الطلبة الذين تحصلواعلى أكبر من المتوسط .

ملاحظة:
البارامتر الذي يرجع لنا قيمة من الدالة يجب أن يكون من النوع المؤشر pointer .

لاحظ ثانياً أن استدعاء الدالة ave يكون على النحو التالي :

```
a = ave ( g , n , &k) ;
```

حيث k هو عدد الطلبة الذين تحصلوا على درجة أكبر من المتوسط ، أي أن التمرير يكون بعنوان k وليس k نفسه .

ولهذا يسمى هذا النوع من التمرير (الاستدعاء بالعنوان call-by-address) أو (الاستدعاء بالمرجع call-by-reference) تمييزاً له عن الاستدعاء بالقيمة call-by-value . أي أن الاستدعاء بالعنوان يؤثر على القيمة الموضوعة في ذلك العنوان ، أما الاستدعاء بالقيمة فلا يؤثر على قيمة المتغير الذي تمرر قيمته (وليس عنوان) إلى الدالة.

يبين الشكل (9.8.1) مثالاً لبرنامج الاستدعاء بالعنوان وفيه نلاحظ ما يلي :

1. يتكون البرنامج من الدالتين main و ave ، حيث تتم في الدالة الأولى قراءة (n) عدد الطلبة و المصفوفة g لدرجاتهم ، وفى الدالة الثانية يتم حساب متوسطهم وفى نفس الوقت حساب عدد الطلبة k الذين تحصلوا على درجات فوق المتوسط .

2. تم استخدام المؤشر kp للمتغير k في الدالة ave لأننا نريد تمرير قيمة k من الدالة ave إلى الدالة المستدعية main .

```
main()
{
        float g[12], a ;
        int n ,i, k;
        float ave( float g[], int n , int *kp);
        printf("\n How many students? ");
        scanf("%d",&n);
        for(i=0; i<n; i++)
        {       printf("\n enter grade[%d]-->",i);
                scanf("%f",&g[i]);
        }
        a=ave(g,n,&k);
        printf("\n average=%f",a);
        printf("\n number of students above average=%d",k);
}
float ave(float g[],int n , int *kp)
{
        float sum=0, a;
```

```
    int i;
    *kp=0;
    for(i=0; i<n ; i++)
            sum += g[i];
    a=sum/n;
    for(i=0; i<n ; i++)
            if( g[i]>a) (*kp)++;
    return(a);
}
```

الشكل (9.8.1) برنامج الاستدعاء بالعنوان

3 . آخر جملة في الدالة ave هي جملة return (a) . لاحظ عدم جواز وضع هذه الجملة قبل حساب kp* ، لأن أي جملة ترد بعد جملة return سيتم إهمالها ولا تحسب .

والآن قد يتساءل الدارس : هل تمرير مصفوفة إلى دالة يعتبر استدعاء بالقيمة أم بالعنوان ؟

والسبب وراء هذا التساؤل هو أننا لاحظنا من قبل أن إعلان مصفوفة [ ] x يعنى أن x هو عنوان العنصر [0] x وبالتالي فإن الاستدعاء

fun (x) ;

هو استدعاء بالعنوان . وإذا تم تغيير المصفوفة المناظرة داخل الدالة fun فسينتج تغيير في المصفوفة نفسها .

مثال (9.8.2) : ماذا يطبع البرنامج المبين بالشكل (9.8.2) ؟

إن ما يطبعه هذا البرنامج هو القيم

m [0] = 55 m [1] = 66 m [2] = 88

وليس القيم التي عينت للمصفوفة m في الدالة main . معنى ذلك أن تغيير المصفوفة في الدالة fun أدى إلى تغييرها في الدالة main . وهذا إثبات للملاحظة التي أشرنا إليها سابقاً .

```
main()
{       int m[3]={ 12 , 23,  44} ;
        void fun( int m[] );
        fun(m);
        printf("\n m[0]=%d  m[1]=%d  m[2]=%d ",
m[0],m[1],m[2]);
}

void fun( int m[] )
{
        m[0]=55;
        m[1]=66;
        m[2]=88;
}
```

الشكل (9.8.2) برنامج استخدام المصفوفة في الاستدعاء بالعنوان

9.9 استدعاء الدالة لنفسها recursion

هل يجوز أن تستدعي الدالة نفسها ؟

نعم يجوز ذلك في لغة سي ، ويسمى هذا النوع من الاستدعاء بالتتابع recursion . وهو أسلوب في البرمجة قد يجد فيه الدارس شيئا من صعوبة الاستيعاب في البداية ، ولكن بشيء من التركيز قد يجد فيه وسيلة ممتعة للاستفادة من الحاسوب في تكرار عمل معين .

لنأخذ مثلاً الدالة :

```
void   p(int i )
 {  printf ( " \n %d " , i ) ;
  }
```

هذه الدالة تقوم بطباعة العدد الصحيح i .

ماذا لو وضعنا لها استدعاء لنفسها كالآتي :

```
  void   p (int    i)
{    printf ( " \n %d " , i ) ;
     p (i) ;
}
```

لو نفذنا هذه الدالة مثلا بالاستدعاء

```
p (5) ;
```

فإنها تقوم بطباعة العدد 5 مالا نهاية من المرات!!

ومعنى ذلك أن استدعاء الدالة لنفسها جائز في لغة سي ، ولكن يجب أخذ الحذر عند التعامل مع الدوال التتابعية recursive (أي التي تستدعي نفسها) ، فقد تدخل في حلقة لانهائية لا خروج منها .

مثال (9.9.1)

يمكن استخدام الدالة p في طباعة الأعداد من i إلى 1 تنازليا كالآتي :

```
void  p (int  i) ;
 {  if ( i = = 0 )  return   ;
    else
       printf ( " \n %d " , i ) ;
       p ( i - 1)  ;
 }
```

الآن يمكننا استدعاء هذه الدالة بالأمر :

```
p (20);
```

حيث تتعين قيمة 20 للبارامتر i في البداية ، ثم تتناقص قيمة i في كل مرة بفعل الاستدعاء :

```
p (i - 1)  ;
```

إلى أن يتحقق الشرط

```
i = = 1
```

عندها يتوقف استدعاء الدالة ويتم الرجوع إلى الدالة المستدعية . ويبين الشكل (9.9.1) البرنامج متكاملاً . لاحظ هنا أن وظيفة الدالة main لم تتعد الإعلان عن الدالة واستدعاءها .

```
main()
{
      void p(int i);
      p(20);
}


void p(int i)
{
      if(i==0)return;
      else
      printf("\n %d",i);
      p(i-1);
}
```

الشكل (9.9.1) دالة تتابعية recursive

مثال (9.9.2) : اكتب برنامجاً لحساب مضروب n حيث

$$n! = n(n - 1)(n - 2) \ldots\ldots\ldots (3)(2)(1)$$

وذلك باستخدام الأسلوب التتابعي recursive .

إذا أطلقنا اسم fac(n) على الدالة مضروب n ، نلاحظ العلاقة التالية :

$$fac(n) = n * fac(n - 1)$$

وذلك لأن :

$$n! = n * (n-1)!$$

وبالتالي يمكننا استخدام هذه العلاقة التتابعية كالآتي :

```
int  fac(int  n)
{  if (n = = 1)  return (1)  ;
   else
return (n * fac(n - 1) ) ;
  }
```

فإذا فرضنا أن n = 4 مثلاً ، سيكون أول ترجيع return للقيمة

$$4 * fac(3)$$

وهنا استدعاء للدالة fac بقيمة (n = 3) لنحصل على :

$$4 * 3 * fac(2)$$

وهنا أيضاً استدعاء للدالة fac بقيمة n = 2 لنحصل على :

$$4 * 3 * 2 * fac(1)$$

الآن قيمة n = 1 مما يسبب توقف الاستدعاء وترجيع قيمة 1 لهـا ، لنحـصل أخيراً على مضروب 4.

```
main()
{
    int fac(int n);
    printf("\n factorial(5)=%d",fac(5));
}
int fac(int n)
{
      if(n==1)return(1);
      else
      return(n*fac(n-1));
}
```

الشكل (9.9.2) برنامج حساب المضروب تتابعا

يبين الشكل (9.9.2) برنامجا لحسـاب !5 ، وعند تنفيذ هذا البرنـامج نحـصل على :

factorial(5) = 120

ملاحظة : يمكننا حساب الدالة الأسية

$$f(x,y) = x^n$$

من العلاقة التتابعية

$$f(x,y) = x * x^{n-1}$$
$$= x * f(x , n-1)$$

بشرط أن يكون n عددا صحيحا موجبا . انظر التمارين .

9.10 الدوال الجاهزة

تتميز لغة سي بتوفير العديد من الدوال الجاهزة التي يمكن أن يستفيد منها المبرمج في العديد من المجالات . للتعرف على هذه الدوال وطريقة عملها يمكن لمستخدم توربو سي Turbo C أن يستعمل المفتاح F1 (halp) للحصول على قائمة بالدوال الجاهزة ، وذلك باختيار موضوع header files ملفات العناوين ، حيث يجد قائمة بالدوال التي تخص كل ملف من هذه الملفات . فمثلا، إذا أردنا الحصول على قائمة بالدوال الرياضية نتحول (بعد الضغط على F1) إلى ملفات العناوين ، ثم نختار الملف math.h لنحصل على قائمة بالدوال والثوابت الرياضية المعرفة في هذا الملف. ويبين الشكل (9.10.1) جدولاً لهذه الدوال ووظيفة كل منها ، مع بيان النطاق (أي مجال متغيراتها) ، والمدى (أي مجال الدالة نفسها) .

الدالة	الوظيفة	النطاق	المدى
abs	القيمة المطلقة لعدد صحيح	int	int
Acos	معكوس جيب التمام	double	[0 , π]
Asin	معكوس الجيب	double	(-π /2 , π/2)

Atan	معكوس الظل	double	$(-\pi/2 , \pi/2)$
Atan2	معكوس ظل (x,y)	(double,double)	$(-\pi , \pi)$
Atof	التحويل من رمز إلى عدد عائم	char	foalt
ceil	التقريب لأقرب أعلى عدد صحيح	double	double
cos	جيب التمام	double	double
cosh	جيب التمام الزائد	double	double
exp	الدالة الأسية e^x	double	double
fabs	القيمة المطلقة لعدد مضاعف أو عائم	double	double
floor	التقريب لأقرب عدد صحيح أو صفر	double	double
fmod	باقي قسمة عددين مضاعفين	double	(double,double)
labs	القيمة المطلقة لعدد طويل	long	long
log	اللوغاريثم الطبيعي	double	double
log10	اللوغاريثم العشري	double	double
pow(x,y)	حساب x أس y	double	(double,double)

pow(2)	حساب 10 أس x	double	double
sin	جيب الزاوية	double	double
sinh	الجيب الزائد	double	double
sqrt	الجذر التربيعي	double	double
tan	ظل الزاوية	double	double

الشكل (9.10.1) الدوال الرياضية الجاهزة

ملاحظات :

1 . الدوال المثلثية tan(x) و cos(x) و sin(x)

تستخدم التقدير الدائري radians وليس الدرجات degrees . فمثلاً لحساب sin(x) ، حيث x زاوية مقاسة بالدرجات ، يجب إجراء عملية التحويل

$$\sin (x * 180/\pi)$$

حيث π مقدار ثابت يساوى تقريباً 3.15196....

2 . الدالة acos(x) هي دالة معكوس جيب التمام ، وهى تعطى قيمة y من النوع المضاعف ، double في الفترة

$$0 \leq y \leq \pi$$

3 . الدالة asin(x) هي معكوس الجيب ، وهى أيضاً من النوع المضاعف ، ولكنها تعطى قيمة y في الفترة

$$-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$$

وهذا المدى ينطبق أيضاً على الدالة atan .

4 . هناك نوعان من دالة معكوس الظل : الدالة atan و الدالة atan2 ، والفرق بينهما يكمن في عدد المتغيرات لكل منهما . فبينما تتطلب الدالة atan قيمة واحدة من النوع double ، تتطلب الدالة atan2 قيمتين (x,y) من هذا النوع. ونستخدم الدالة atan2 عندما نهتم بموقع الزاوية ، فمثلاً إذا كانت x سالبة وأيضا y سالبة فمعنى ذلك أن الزاوية تـــقع في الربع الثالث (أي أكبر من π وأصغر من 3π/2) .

5 . الدالة pow(x,y) تمكننا من حساب x^y ، ولكـــن يجب أخذ الحذر من عملية مثل $(-2)^{0.5}$ لأننا بذلك نقوم بأخذ جذور عدد سالب وهى عملية ممنوعة .

6 . الدالة sqrt(x) تحسب الجذر التربيعى لأي عدد مضـــاعف x حيث $0 \leq x$.

7 . عند حساب القيمة لعدد x نستخدم الدالة المناسبة للنوع. فالنوع int يتطلب الدالة abs ، والنوع العائم أو المضاعف يتطلب fbas ، أما النوع الطويل long فيتطلب الدالة labs .

و هناك العديد من الدوال المهمة الأخرى إلى جانب الدوال الرياضية . والجدول (9.10.2) يبين بعض هذه الدوال المعرفة في الملف (stdio.h)

الدالة	الوظيفة
strtod	تحويل نضيد إلى عدد مضاعف
fcvt	تحويل عدد عائم إلى نضيد
atof	تحويل نضيد إلى عدد عائم
atoi	تحويل نضيد إلى عدد صحيح
atol	تحويل نضيد إلى عدد طويل
itao	تحويل عدد صحيح إلى نضيد
ltao	تحويل عدد طويل إلى نضيد
rand	تكوين رقم عشوائي

الجدول (9.10.2) بعض دوال الملف (stdlib.h)

مثال (9.10.1) : اكتب برنامجا لحل المعادلة من الدرجة الثانية

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

باستخدام القانون :

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

لاحظ إمكانية وجود جذور مركبة complex وذلك عندما :

$$b^2 - 4ac < 0$$

في هذه الحالة يتكون الجدر المركب من جزء حقيقي وجزء تخيلي كالآتي :

الجزء الحقيقي للجذر الأول $x_1 = \frac{-b}{2a}$

الجزء التخيلي للجذر الأول $y_1 = \frac{\sqrt{4ac - b^2}}{2a}$

الجزء الحقيقي للجذر الثاني $x_1 = x_2$

الجزء التخيلي للجذر الثاني $y_1 = y_2$

ويبين الشكل (9.10.1) البرنامج المطلوب.

```
/*  Solution of the quadratic equation:

        ax^2 +bx + c =0

*/
#include <math.h>
main()
{
        double a,b,c,*x1p,*x2p,*y1p,*y2p,  x1, x2, y1, y2, d;
        char string[30];
        void roots(double a, double b,  double c,
                        double *x1p, double *x2p,
                        double *y1p ,double *y2p,
                        char string[] ) ;
        x1p=&x1;    x2p=&x2;
        y1p=&y1;    y2p=&y2;
        printf("\n enter a-->");      scanf("%lf",&a);
        printf("\n enter b-->");      scanf("%lf",&b);
        printf("\n enter c-->");      scanf("%lf",&c);
        roots(a,b,c,&x1,&y1,&x2,&y2, string);
        printf("\n Solution of quadratic equation. \n");
        puts(string);
        printf("\n x1=%f \t y1=%f \n x2=%f \t
                y2=%f",*x1p,*y1p,*x2p,*y2p);
}
void roots(double a, double b, double c,
                double  *x1p, double *y1p, double *x2p,
                double *y2p,
                char string[])
{
```

```
double d;
d= pow(b,2)-4*a*c;
printf("\n det=%f",d);
if(d==0)
{
        *x1p=-b/(2*a);
        *x2p=*x1p;
        *y1p=0;
        *y2p=0;
        strcpy(string," There are 2 equal real roots:");
        return;
}
if(d>0)
{       *x1p=(-b+sqrt(d))/(2*a);
        *x2p=(-b-sqrt(d))/(2*a);
        *y1p=0;                 *y2p=0;
        strcpy(string,"There are two real roots:");
        return;
}

if(d<0)
{
        *x1p=-b/(2*a);                  *x2p=*x1p;
        *y1p=sqrt(-d)/(2*a);            *y2p=-*y1p;
        strcpy(string,"There are 2 complex roots:");
        return;
    }
}
```

الشكل (9.10.1) برنامج حل معادلة الدرجة الثانية

ملاحظات عن البرنامج (9.10.1)

1. الدالة roots هي التي تقوم بعملية إيجاد الجذرين ، حيث تستقبل هذه الدالة المعاملات a,b,c من النوع double وترجع الآتي :

الجزء الحقيقي للجدر الأول = *x1p

الجزء الحقيقي للجذر الثاني = *x2p

الجزء التخيلي للجذر الأول = *y1p

الجزء التخيلي للجذر الثاني = *y2p

حيث نلاحظ استخدام المؤشرات لهذه المتغيرات الأربعة لأنها تمرر قيماً من الدالة roots إلى الدالة المستدعية .

2. استخدام التوجيه

```
# include < math.h >
```

وذلك نظراً لاستخدام الدالة الرياضية ( )sqrt لإيجاد الجذر التربيعي للمحدد ، وكذلك الدالة pow(b,2) لحساب b^2.

3. أضفنا string إلى بارا مترات الدالة ( )roots ، وهو عبارة عن نضيد يكافئ الآتي :

أ . في حـــالة وجـــود جذرين حـــقيقيـــين متساويين

(أي عندما $d = b^2 - 4ac = 0$) فإن :

string = " There are 2 equal real roots "

ب . فـــي حـــالة وجـــود جذرين حقـــيقـــيين غـــير متساويين (أي عندما $d > 0$) فإن :

string = " There are 2 real roots "

جـــ . في حالة وجود جذرين مركبين (أي عندما $d > 0$) فإن :

string = " There are 2 complex roots "

لاحظ أن النضيد string هو عبارة عن مصفوفة من الرموز ، وأن :

string = & string[0]

وبالتالي فإن string يعتبر مؤشراً ويمكن استخدامه في الاستدعاء بالعنوان .
كما لاحظنا سابقا لا يجوز في لغة سي استخدام المؤثر " = " لتعيـــين نـــضيد ، ولكن (بدلاً من ذلك) نستخدم الدالة strcpy كما في البرنامج .

مثال (9.10.2) : اكتب برنامجا يقوم بقراءة عدد من النوع الصحيح ولكن يفترض أولاً أنه نضيد (مصفوفة من الرموز) ، ثم بعد التأكد من خلوه من أي رمز (غير الأرقام من 0 إلى 9) يقوم بتحويله إلى عدد صحيح .

يبين الشكل (9.10.2) البرنامج المطلوب في هذا المثال ، حيث نجد المتغيرات والدوال التالية :

itemp : نضيد مؤقت لقراءة العدد المدخل .

inum : عدد صحيح يكافئ itemp بعد عملية التحويل .

check : دالة لاختبار نضيد والتأكد من خلوه من أي رمز عدا الأرقام من 0 إلى 9 . تقوم بترجيع 1 بعد التأكد من هذه العملية وترجيع 0 في الحالة الأخرى .

C : متغير صحيح توضع فيه القيمة المرجعة من الدوال check .

لاحظ أن عملية التحويل من نضيد إلى عدد صحيح

```
inum = atoi(itemp) ;
```

لا تتم إلا إذا كانت (c = = 1) ، أي بعد التأكد من خلو itemp من الرموز الأخرى غير الأرقام .

لكي تختبر هذا البرنامج ، أدخل أرقاماً سليمة ، لتحصل على الرسالة :

you have entered the number .....

أو أدخل أرقاما تحتوى على رمز غـير رقمية لتحـصل على الرسالة :

data entery error

```
#include <ctype.h>
main()
{
        char  itemp[5];
        int inum, c;
        printf("\n enter a number-->");
        gets(itemp);
        c=check(itemp);
        if(c==1)
        {       inum=atoi(itemp);
                printf("\n you have entered the number
%d",inum);
        }
        else
                printf("\n data entry error");
}
int check(char str[] )
{
        int i;
        for(i=0;  str[i] != '\0' ; ++i)
                if( ! isdigit(str[i])  ) return(0);
        return(1);
}
```

الشكل (9.10.2) برنامج يستخدم الدالة atoi

9.11 تمارين

1 . اكتب الدالة ( )greet التي تطبع النضيد :

" Good morning. How are you ? "

واستخدمها في الدالة ( )main .

2 . ما معنى المصطلحات التالية؟ :

global variable متغير عام
local variable متغير محلي
function دالة

3 . اكتب الدالة getage من النوع float التي تقوم بسؤال المستخدم على النحو :

How old are you ?

ثم تقوم بقراءة عمره .

استدع هذه الدالة في الدالة الرئيسية ( )main وذلك لغرض طباعة كلمة (ok إذا كان العمر أكبر من 18 وإلا فتتم طباعة العبارة (too young) .

ملاحظة : استخدم متغيرا عاما age في هذا البرنامج ، واستخدم دالة لطباعة المخرجات .

4 . ماذا يطبع البرنامج التالي؟ :

```
main( )
  { int k = 5
    void f( int k ) ;
    f(k) ;
    printf( " \n %d " , k ) ;
  }
void f( int k )
  { k = 6 ;
    return ;
  }
```

5

5 . ماذا يطبع البرنامج التالي؟ :

```
float x = 8.5
main( )
  { void fun( void ) ;
    fun( ) ;
    printf( " \n %f " , x ) ;
  }
void fun( void )
  { x = 9.5 ;
```

9.5

```
    return ;
}
```

6. اكتب الدالة

```
float  tax( float  income )
```

التي تقوم باستقبال قيمة الدخل income وحساب الضريبة tax على النحو التالي :

الضريبة = 15 % من الدخل إذا قل الدخل عن 500 .

= 20 % من الدخل إذا كان الدخل 500 فما فوق .

استخدم هذه الدالة لحساب الضريبة على الدخل لعدد 10 موظفين .

7. إذا كان قبول الطالب في قسم الحاسب الآلي يعتمد على متوسطه العام ودرجته في المقرر CS111 ، بحيث لا يقل المتوسط العام عن 65 ، ولا تقل درجته في المقرر CS111 عن 50 ، اكتب دالة ترجع قيمة 1 إذا توفر الشرط ، وقيمة 0 إذا لم يتوفر أحدهما أو كلاهما .

استخدم هذه الدالة في إعداد قائمة بالطلبة المقبولين من بين مجموعة المتقدمين .

8. إذا كانت أعمال الفصل تحسب من درجات 3 امتحانات ، بحيث تساوى درجة أعمال الفصل مجموع أكبر درجتين من الدرجات الثلاث ، اكتب دالة

تقوم بهذا العمل ، واستخدمها لحساب درجات أعمال الفصل لعدد من الطلبة .

9 . أحسب

$$f(x) = x^2 + 2x + 3$$

أ . باستخدام الماكرو.

ب . باستخدام الدالة .

وذلك لجميع قيم x من 0 إلى 2 بزيادة ثابتة مقدارها 0.1 .

10 . استخدم الماكرو لتعريف عنوان متغير في الذاكرة على النحو ADDRESS(x) كبديل للمؤشر المتعارف عليه &x ، وكذلك لتعريف AND بدلا من المؤشر && ، و OR بدلاً من || .

واستخدم هذه التعريفات في دالة تقوم بقراءة العمر ، وتطبع كلمة " OK " إذا كان العمر أقل من 65 وأكبر من 18 ، وتطبع كلمة " ERROR " إذا كان العمر أقل من 18 وأكبر من 65 .

11 . اكتب الدالة sum التي تستقبل مصفوفة من النوع float وعدد عناصرها n ، وتقوم بإيجاد مجموع كل العناصر .

استخدم هذه الدالة لحساب مجموع عناصر مصفوفة x عدد عناصرها 10 ، وكذلك حساب مجموع مربعات هذه العناصر .

12 . استخدم الاستدعاء بالعنوان في كتابة دالة calc تقوم بقراءة قيمة المشتريات ، وحساب الربح (15% من رأس المال) وقيمة المبيعات اللازمة لتحقيق هذا الربح . استخدم هذه الدالة لإعداد قائمة أسعار الشراء والبيع بها 10 أصناف من البضاعة .

13 . اكتب الدالة range التي تستقبل المصفوفة grades من النوع العائم float ، وتقوم بحساب أعلى قيمة max و أدنى قيمة min والفرق بينهما .

استخدم هذه الدالة في حساب أعلى وأدنى درجة ، والفرق بينهما لعدد 20 طالباً .

14 . اكتب الدالة star التي تقوم بطباعة الرمز * وذلك بعدد n من المرات

أ . استخدم الأسلوب العادي (التكراري) .

ب . استخدم الأسلوب التتابعي recursive .

15 . اكتب الدالة p(x,n) التي تقوم بحساب x^n بالأسلوب التتابعي recursive النوع العائم float ، و n من النوع الصحيح n . استخدم هذه الدالة في حساب $(3.5)^3$.

16 . اكتب الدالة reverse التي تقوم باستقبال string نضيد ، وتقوم بطباعته معكوساً وذلك باستخدام الأسلوب التتابعي . مثـــال : النـــضيد " book " تطبعه الدالة على النحو " koob " .

17 . اكتب الدالة الجاهزة التي تقوم بحساب الآتي :

أ . القيمة المطلقة للعدد (5-)

ب . القيمة المطلقة للعدد (6.4 -)

جـ . التحويل من 6.4 إلى 7

د . التحويل من 6.7 إلى 6

هـ . اللوغاريتم العشري للعدد 26.7

و . اللوغاريتم الطبيعي للعدد 34.5

ز . جيب الزاوية 30° درجة

ح . $e^{2.1}$

ط . $(1.3)^{3.4}$

ي . الجدر التربيعي للعدد (7.8)

ك . تحويل نضيد إلى عدد عائم .

18 . اكتب دالة لإجراء اللعبة التالية :

يكوّن الحاسوب رقما عشوائيا بين الصفر والمائة ، ويطلب من اللاعب أن يعرف ما هو هذا الرقم ، فإذا كانت إجابته أقل من الرقم المطلوب ، تظهر كلمة (higher) على الشاشة ، وإذا كانت إجابته أكبر من المطلوب ، تظهر كلمة (lower) وهكذا حتى يحصل اللاعب على الإجابة الصحيحة .

19 . استخدم الدالة الجاهزة atof لتحويل نضيد إلى عدد عائم ، وذلك بعد التأكد من خلو العدد من أي رمز غير الأرقام والفاصلة العشرية .

20 . اكتب الدالة sort التي تقوم باستقبال المصفوفة array من النوع float ، وعدد عناصرها n ويتم ترجيعها مرتبة تنازليا .
اختبر هذه الدالة في برنامج كامل .

21 . اكتب الدالة sortnames التي تقوم باستقبال المصفوفة names من النوع النضيد (أي مجموعة من الأسماء عددها n) ، وتقوم الدالة بترتيب هذه المصفوفة تصاعديا . اختبر هذه الدالة بمجموعة من الأسماء .