المحاضرة الثالثة

الدوال (التوابع) Functions:

الدالة هي جزء من البرنامج تقوم بأداء مهمة معينة وتعيد نتيجة ما، ويمكن أن تُمرر لها بارمترات من البرنامج المستدعي.

الهدف الأساسي من استخدام الدوال هو منع تكرار كتابة التعليمات البرمجية لأكثر من مرة، مما يؤدي إلى تقليل أسطر البرنامج، وبالتالي سهولة تتبع الأخطاء في حال حدوثها في البرنامج،
هذا بالإضافة إلى أن الدوال تجعل البرنامج منظم بشكل أفضل.

يمكن استخدام الدالة بعد التصريح عنها، عن طريق استدعاء هذه الدالة في البرنامج أو في دالة أخرى، وذلك بذكر اسم الدالة وتمرير قيم مناسبة للبارمترات الخاصة بهذه الدالة.
عند استدعاء دالة ما، ينتقل سير البرنامج إلى تلك الدالة ويتم تنفيذ التعليمات الموجودة ضمنها، ثم تتم العودة إلى البرنامج الذي قام بالاستدعاء.

التصريح عن الدالة (التابع): يتم التصريح عن التابع أو الدالة بالشكل التالي:

```
Type function_name(type parameter1, type parameter2 ,…..)
{
    التصريح عن المتغيرات المحلية الخاصة بالدالة

  ……..
   تعليمات الدالة

  ……..
  return …….
}
```

حيث:

Type : نوع القيمة التي ستعيدها الدالة (int , float ,....)، اذا كانت الدالة تعيد قيم
Function_name : اسم الدالة.
Type parameter1 : نوع واسم الوسيط الاول من وسطاء الدالة.
Type parameter2: نوع واسم الوسيط الثاني من وسطاء الدالة.
.....

ملاحظات:

١. يجب أن يتطابق نوع القيمة التي ستتم إعادتها في تعليمة return مع النوع المحدد في القيمة المعادة type.
٢. في حال كانت الدالة لا تعيد قيمة عندها تكون القيمة المعادة type هي **void**، وعندها لا نستخدم تعليمة return .
٣. يجب أن يتطابق عدد ونوع البارمترات المستخدمة عند التصريح عن الدالة مع عدد ونوع البارمترات التي سيتم تمريرها إلى هذه الدالة عند استدعائها.
٤. يتم التصريح عن الدوال عادة قبل الدالة الرئيسية Main.

المتغيرات المحلية (local) والمتغيرات العامة (global):

المتغيرات المحلية: هي المتغيرات التي يتم الإعلان عنها داخل الدالة, وتسمى بهذا الاسم لأنها تكون غير معروفة خارج الدالة المعرفة ضمنها, ومن الجدير بالذكر أن المتغير المحلي تنشأ له قيمة وكيان عند ابتداء تنفيذ الدالة التي ينتمي إليها فقط, وتختفي عند الانتهاء من تنفيذ تلك الدالة.

المتغيرات العامة: هي التي تكون قيمتها معروفة من أول البرنامج إلى آخره, ويمكن استعماله في أي جزء منه, وتحتفظ بقيمتها أثناء تنفيذ البرنامج, يتم التصريح عن هذه المتغيرات خارج حدود الدوال, ومن المعتاد عملياً التصريح عن هذه المتغيرات في بداية البرنامج.

برنامج 1: اكتب تابع يقوم بطباعة العبارة " Hello , in C++ " على الشاشة :

```
#include <iostream.h>
void  func1 ( )
{
```

```
   Cout<< Hello , in C++ " ;

}

main()
{
  func1( ) ;

}
```

نلاحظ من المثال السابق أن الأقواس ضرورية عند التصريح عن الدالة وكذلك عند استدعائها حتى وان لم تكن هذه الدالة تستخدم أية بارمترات.

برنامج 2: اكتب دالة تقوم بجمع عددين وإعادة النتيجة .

```
#include <iostream.h>
int   add  ( int  x , int  y )
{
   int   z ;

   z = x +  y ;

return  z ;

}

main ( )
{
 int  num1 , num2  , result ;

cout << "Enter first number: " ;

cin >> num1 ;
```

```
cout << "Enter second number: " ;

cin >> num2 ;

result = add (num1 , num2);

cout<< " Result is : " << result ;

}
```

برنامج 3: اكتب دالة تقوم بجمع الأعداد من 1 وحتى عدد معين يتم تحديده من قبل المستخدم، وإعادة ناتج الجمع لتتم طباعته في البرنامج الرئيسي .

```
#include <iostream.h>
int   func_add  ( int  a  )
{
   int   s = 0 ;

   for ( int i = 1 ; i < a ; i++ )

   s = s + i ;

   return  s ;

}

main ( )
{
 int  num , result ;

cout << " Enter the number: " ;

cin >> num ;
```

```
result = func_add ( num );

cout<< " Result is : " << result ;

}
```

برنامج 4: اكتب دالة تقوم بحساب مربع عدد ما ، وطباعة النتيجة في البرنامج الرئيسي.

```
#include <iostream.h>
int   func_Squre  ( int  z  )
{
   return  z * z ;

}

main ( )
{
 int  x ;

cout << " Enter the number: " ;

cin >> x ;

cout<< " Result is : " << func_Squre  ( x );

}
```

برنامج : اكتب برنامج للتعامل مع المصفوفات أحادية البعد يحوي الدوال التالية:

١. دالة لإدخال عناصر المصفوفة.
٢. دالة لطباعة عناصر المصفوفة.
٣. دالة لجمع عناصر المصفوفة.
٤. دالة لإيجاد اكبر عنصر في المصفوفة.
٥. دالة لترتيب عناصر المصفوفة تصاعدياً.

```
#include<iostream.h>
void read ( int x [ ] , int y )
{
   for(  int i = 0 ; i <  y ; i++ )
    {
        cout << " x[ " << i << " ]= " ;
        cin >>  x[ i ] ;
    }
}
void print( int x [ ] , int y )
{
   cout  << " [ ";
   for(  int i = 0 ; i <  y ; i++ )
        {
          cout  <<  x[ i ] << " \t ";
        }
   cout  << " ] " ;
}
```

```
int add ( int x [ ] , int  y)
{
   int sum=0 ;
   for(  int i = 0 ; i <  y ; i++ )
                sum= sum +x [ i ] ;
    return sum ;
}
int findmax( int  x[ ] , int  y)
{
        int g = x[ 0 ] ;
        for( int  i = 0 ; i < y ; i++)
         {
                If ( x[ i ] > g )
                     g = x [ I ] ;
         }
        return g ;
}
void sort ( int  x [ ] , int  y )
{
   int  p ;
   for ( int i = 0 ; i < y ; i++)
   for( int j = i ; j < y ; j++)
        {
          if ( x[ j ] < x[ i ] )
```

```
                {
                   p = x[ j ];
                   x[ j ] = x[ i ] ;
                    x[ i ] = p ;
                }
}
cout<< " After Sort "<< " \n " << " x = [ " ;
for( i = 0 ; i < y ; i++ )
cout << x[ i ]<<" \t " ;
cout<< " ] " ;
}
main()
{
 const  int  n = 5 ;
 int x[ n ] , s[ n ] ;
 cout <<" Please, Enter Array elements";
 read( x, n) ;
 cout << " \n " <<  " Array Elements are: "
 print( x , n ) ;
 cout<< " \n ";
 cout<< " Sum of Array Elements = " << add(x , n) << " \n " ;
 cout<< " Max Number in Array = " << max( x , n ) << " \n " ;
 sort( x , n ) ;
 }
```