FUNCTIONS

A function is a group of statements used to perform a certain operation. It is called (تستدعى) from some point of the main program or another function. There are several advantages of using functions:

- Functions allow for breaking down the program into discrete units.
- Programs that use functions are easier to design, program, debug and maintain.

Functions are of two types:

1. Functions return a value to the main program end with ***return*** statement.
2. Functions do not return a value defined by the word ***void***.

1. FUNCTIONS RETURN A VALUE:

The function definition is illustrated below:

```
type  function_name  (type  parameter1, type  parameter2, …… )
{
        statements ;
        return value;
}
```

In the above definition the first word is the ***type*** of the function, it is the type of data it returns. The second item is the ***name*** of the function. ***(type parameter1, type parameter2, ……)*** are called arguments of the function. For example:

```
int sum(int x, int y)
{
     int sum=x+y;
     return sum;
}
```

ACCESSING A FUNCTION (The Call of the Function) استدعاء الدالة

The main program calls the function by declaring a variable followed by the function name and its arguments, as follows:

type variable_name=function_name(arguments)

For example:

```
int s1=sum(a, b) ;
```

Ex: Write a function finds the sum of two numbers, the main program calls the function to find the sum of four numbers.

```
#include<iostream.h>
float sum(int x, int y)
{
     float z=x+y;
     return z;
}
main( )
{
     float a,b,c,d;
     cin>>a>>b>>c>>d;
     float s1=sum(a,b);
     float s2=sum(c,d);   float s=sum(sum(a,b),sum(c,d));
     float s=sum(s1,s2);
     cout<<"sum= "<<s;
     return 0;
}
```

Ex: Write a program includes a function receives a character and returns its type (number, lowercase alphabet, uppercase alphabet, or symbol)

```
#include <iostream.h>
int chkletter( char c)
{
    if(c >= 'A' && c <= 'Z')
         return (1);
   else if(c >= 'a' && c <= 'z')
         return (2);
   else if (c>='0' && c<='9')
        return (0);
   else return(3);
}
main()
{
      int type;
      char ch;
      cout << "Enter the character:"<<endl;
      cin >> ch;
      type = chkletter( ch );
      switch(type)
      {
         case 0: cout << "It is a number"; break;
         case 1: cout << "It is an uppercase alphabet"; break;
         case 2: cout << "It is a lowercase alphabet"; break;
         default: cout<<" It is a symbol";
      }
      return 0;
}
```

<u>Ex:</u> Write a program includes two functions, the first is named power receives two numbers and return the power and the second is named octal receives an octal number and returns the decimal value by using the first function.

```
#include<iostream.h>
int power(int x, int y)
{
       int p=1;
       if (x==0)
       return p;
       else
       {
            for(int i=1; i<=x; i++)
                 p*=y;
            return p;
       }
  }
  int octal(int z)
  {
       int m=0, sum=0;
       while(z!=0)
```

```
          {
               int y=z%10;
               int k=power(m, 8);
               sum+=k*y;
               z/=10;
               ++m;
          }
          return sum;
     }
     main( )
     {
          int A;
          cin>>A;
          int n=octal(A);
          cout<<n;
     }
```

<u>Ex:</u> Write a function that finds the factorial of an integer, the main program calls the function to compute "y" from the following formula.

y=k!*m!/(k-m)!

```
#include<iostream.h>
long fact (int n)
{
     long f=1;
     int i;
     for(i=1; i<=n; i++)
          f*=i;
     return f;
}
main( )
{
     long y;
     int k, m;
     cin>>k>>m;
```

```
          if (m>k){
            long f1=fact(k);
            long f2=fact(m);
            long f3=fact(k-m);    y=fact(k)*fact(m)/fact(k-m);
            y=f1*f2/f3;
            cout<<"y="<<y;}
          else cout<<"Error";
          return 0;
     }
```

2. VOID FUNCTIONS

A function does not return a value to the main program and is known as a *procedure* or a *subroutine*. In C++, such a function is identified simply by placing the word ***void*** before the name of the function as shown below:

void function_name (arguments)

For example :

```
void AA(int x, int y)
{
     int sum;
     sum=x+y;
     cout<<sum;
}
```

This type of functions is called by its name without using any variable as shown:

AA(a, b) ;

Ex: Write a function that find and print the square of an integer, the main program calls this function to find the squares of 0-10.

```
#include<iostream.h>
void square(int x)
{
     int z;
     z=;
     cout<<x<<"\t"<< x*x <<endl;
}
main( )
{
    cout<<"x\t"<<"x*x"<<endl;
    cout<<"--\t"<<"----"<<endl;
    int i;
    for(i=0; i<=10; i++)
        square(i);
    return 0;
}
```

The following function does not receive arguments form the main program and does not return value to the main program. The definition is as shown below:

```
void AA(void)
{
    cout<<"this is a C++ program";
}
```

The call of this function is as shown below:

```
AA( );
```

H.W.

(1) Write a program to find the sum of the following series:

S=(1) + (1+2) + (1+2+3) + (1+2+3+4) +…………+ (1+2+3+……….+n)

(2) Write a program include two functions, the first is named fact receives one integer and returns its factorial, and the second is named pow receives two integers and returns the power of them. The main program uses the two functions to find the value of Z from the following series.

$$Z = 1 + \frac{x}{2!} + \frac{x^2}{4!} + \frac{x^3}{6!} + \ldots\ldots\ldots + \frac{x^n}{n!}$$