Lecture 1

Functions in C++

1.1 Introduction

A function groups a number of program statements into a unit and gives it a name. This unit can then be invoked from other parts of the program.

Functions play an important role in C program development. Dividing a program into functions is one of the major principles of top-down structured programming. Another advantage of using functions is that it is possible to reduce the size of a program by calling and using them at different places in the program.

OR

A function:- is a function that performs a set of operations, whether arithmetic, logical, and input and output. The program consists of one or group of Functions.

1.2 Benefits of Function

1- Helps in the abbreviation of writing software code.
2- Avoid repetition of the code when you need to write it again.
3- Provides the required memory space for the program.
4- Shorten the time in the implementation of the program and the implementation of the program as soon as possible.
5- Easy to review the code and correct the code and amend.

1.3 Declaring Functions

The general form for the C++ style of declaring functions is:

```
return type   function_ name (typed parameter list);
```

Example:-

float volume (int x , float y , float Z) ;

Or

float volume (int , float , float) ;

Remember the following rules when declaring C++ functions:

1- The return type of the C++ function appears before the function name.

2- It the parameter list is empty we used the key word "void" to explicitly state that there are no parameters.

3- You must declare each parameter explicitly even that this parameter has the same type.

4- The body of a C++ function is enclosed in braces ({}) there is no semicolon after the closing brace.

5- C++ supports passing arguments either, by value or by reference.

Example:-

```
Void swap (int &a, int &b )          // uses reference
{
int   t=a ;
      a=b;
      b=t;
}
```

Now, if m and n are two integer variables, then the function-call

swap (m,n); // will exchange the values of M and n using their aliases a & b .

The passing arguments by reference, the function is working with its own arguments, it is actually working on the original data. Call passes arguments by value. The called function creates a new set of variables and copies the values of arguments into them; the function does not have access to the actual variables in the calling program, and can only work on the copies of values.

6- C++ supports local constants data type and variable although these data items can appear in nested block statements. C++ does not support nested function.

7- The return keyword returns the functions value.

8- If the functions return type is void, you do not have to use the return keyword.

Note:- C++ dictates that you either declare or define a function before you use it. Declaring a function commonly called “prototype”.

Example:-Write program define a function to find and print value of square any number.

```
  // prototype the function square
#include <iostream>
using namespace std;
 double sqr (double);
 int main ( )
 {
    cout<<”5^2=”<<sqr(5)<<endl;
    return 0;
  }
   double sqr (double a)
    {  return  a*a;}
```

Component	Definition and Purpose	Syntax
Declaration (prototype)	Referred to as the function signature **Specifies function name, argument types, and return value.** Alerts compiler (and programmer) that a Function is coming up later .terminated with a semicolon.(**before main ()**)	return_type fun_name(type arguments);
Calling the function	The function name, followed by parentheses. The syntax of the call is very similar to that of the declaration, except that the return type is not used. **The call** is terminated by a semicolon. **Causes the function to be executed.**	Void main( ) { fun_name(type arguments); }
The Function Definition	The definition contains the actual code for the function. The definition consists of a line called the declarator, followed by the function body.	return_type fun_name (type arguments) { Function statements; return ; }

1.4 Eliminating the Declaration الطريقة الثانية بدون خطوة الاعلان

The second approach to inserting a function into a program is to eliminate the function declaration and place the function definition (the function itself) in the listing before the first call to the function.

Define the function below تعريف الدالة في الاسفل	**Define the function above تعريف الدالة في الاعلى**

Example:-
#include <iostream>
using namespace std;
int main ( )
{
cout<<"5^2="<<**sqr(5)**<<endl;
return 0; }
double sqr (double a)
{ return a*a;}
	Example:-
	#include <iostream>
	using namespace std;
	double sqr (double a)
	{ return a*a;}
	int main ( )
	{ cout<<"5^2="<<**sqr(5)**<<endl;
	return 0;
	}

1.5 Local and Global Variables

All variables define inside the block of a function are called **local variables (**defined inside main() or inside other functions). These variables belong to the function exclusively. That is no other function has access to it.

When variables declared outside any function, it is called **global variables (**A global variable is visible to all the functions). This type of variables can be accessed by all the functions in the program as well as the main function.

1.6 Default Arguments

C++ allows use to call a function without specifying all its arguments. In Such cases, the function assigns default value to the parameter which does not have a matching argument in the function call. Default values are specified when the function is declared.

Example:-

Prototype (i.e. function declaration) with default value.

Example:-

float amount (float principal, int period, float rate =1.5);

Example:-

//Default Arguments //

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main( )
{
float amount;
```

```
float value (float p , int n, float  r = 0.15 );          // prototype
void printline (char ch = '*', int len = 40 );            // prototype
printline ( );                                            // uses default values for
arguments
amount = value (5000.00,5);
cout<<``\n Final Value = ``<<amount <<``\n\n``;
printline (`=` ) ;
return 0;
}
```

1.7 Passing Arguments to Functions

1- Passing by Value

When parameters are passed by value, a copy of the parameters value is taken from the calling function and passed to the called function. The original variables inside the calling function, regardless of changes made by the function to it are parameters will not change. All the pervious examples used this method.

Example:-Write program define a function to compute power of any integer number.

```
#include <iostream>
using namespace std;
long power (int base,int exp)
{
int PW=1;
  for (int i=0;i<exp; i++)
  PW*=base;
  return PW;
  }
```

```
int main()
{
 cout<<"power of 2^8 =   "<<power(2,8)<<endl;
 int X,Y;
 cout <<"read value of X and Y :\t";
 cin>>X>>Y;
 cout <<"the power X^Y ="<<power(X,Y);
 cout <<"\n power of X^5=  "<<power(X,5);
return 0;
}
```

Example:-Write program define function to compute the power from the equation. $Y=\frac{X^2}{2}-\frac{X^4}{4}+\frac{X^6}{6}$ - ……..+ $\frac{X^n}{n}$

```
#include <iostream>
using namespace std;
long power (int base , int exp)
{
int PW=1;
   for(int i=0;i<exp;i++)
   PW*=base;
   return PW;
   }
int main()
{
 int X, n , r =1; double Y;
 cin>>X;
 cin>>n;
 for (int  i=2;i<= n ;i+=2)
 {
```

```
 Y+= (power(X,i)/i)*r;
 cout<<"\n the result value Y=\t"<<Y;
 r*=-1;
return 0;
 } }
```

Example:-Write program to find the value of Z, where Z=$\frac{X!+y!}{(X+y)!}$, define a function to compute factorial.

```
  #include<iostream>
#include <conio.h>
using namespace std;
int fact (int );
int main()
{
int X,Y;double Z;
cout <<"enter value of X,Y"<<endl;
cin>>X>>Y;
 Z=(float)fact(X)*fact(Y)/fact(X+Y);
cout<<"\n the value of Z=\t"<<Z;
getch();
}
int fact (int X1)
{
int f=1;
for(int i=1;i<=X1;i++)
f*=i;
return f;
}
```

2- Passing by Reference:

When parameters are passed by reference their addresses are copied to the Corresponding arguments in the called function, instead of copying their values. Thus pointers are usually used in function arguments list to receive passed references. This method is more efficient and provides higher execution speed than the call by value method, but call by value is more direct and easy to use.

Example:-Write program to exchange values of two numbers.

```
#include <iostream>
using namespace std;
void swap(int *a,int *b);
int main()
{
int num1=5,num2=20;
cout<<"num1 before swapping is:"<<num1<<endl;
cout<<"num2 before swapping is:"<<num2<<endl;
swap(& num1,& num2);
 cout<<"num1 after swapping is:"<<num1<<endl;
cout<<"num2 after swapping is:"<<num2<<endl;
}
 void swap(int *a,int *b)
 {
 int temp;
 temp=*a;
 *a=*b;
 *b=temp;
return 0;
 }
```

Example:-Suppose you have pairs of numbers in your program and you want to be sure that the smaller one always precedes the larger one. To do this you call a function, order(), which checks two numbers passed to it by reference and swaps the originals if the first is larger than the second.

```
// orders two arguments passed by reference
#include <iostream>
using namespace std;
void order(int& numb1, int& numb2)        //orders two numbers
{
if(numb1 > numb2)                         //if 1st larger than 2nd,
{
int  temp = numb1;                        //swap them
numb1 = numb2;
numb2 = temp;
}
}
int main()
{
int n1=99, n2=11; //this pair not ordered
int n3=22, n4=88; //this pair ordered
order(n1, n2); //order each pair of numbers
order(n3, n4);
cout << "n1=" << n1 << endl; //print out all numbers
cout << "n2=" << n2 << endl;
cout << "n3=" << n3 << endl;
cout << "n4=" << n4 << endl;
return 0;
}
```

The output reveals that the first pair has been swapped while the second pair hasn't. Here it is:

n1=11

n2=99

n3=22

n4=88

Example:-Write program define three functions first to read arrays A(7) ,B(7), second to find Z=A+B, third to print A,B,Z.

```
#include <iostream>
#include <conio.h>
using namespace std;
int const n=7;
void sum (int X1[n] ,int X2[n],int *X3);
void read  (int *y);
void print (int  *y);
int main()
{
int A[n],B[n],Z[n];
cout <<"\n read matrix A";
read (A);
cout <<"\n read matrix B";
read (B);
cout <<"\n z=a+b  "<<endl;
sum(A,B,Z);
cout <<"\n print A:"<<endl;
print (A);
cout <<"\n print B:"<<endl;
print (B);
```

```
cout <<"\n print Z:"<<endl;
print (Z);
getch();
return 0;
}
void read (int *X1)
{
 for (int i=0;i<n;i++)
 cin>>X1[i];
 }
 void sum (int X1[n],int X2[n],int *X3)
 {
  for (int i=0;i<n;i++)
  X3 [i]=X1[i]+X2[i];
  }
  void print (int *X)
  {
  for (int i=0;i<n;i++)   cout <<"  "<<X[i];  }
```

Example:-Write program using functions first one to read an array (3*4), second function to print an array.

```
#include <iostream>
#include <conio.h>
using namespace std;
  int const n=3,m=4;
void read (int Xn[n][m])
{
 for (int i=0;i<n;i++)
 for (int j=0;j<m;j++)
```

```
 cin>> Xn[i][j];
 }
void print (int X[n][m])
  {
  for (int i=0;i<n;i++)
  {
  for (int j=0;j<m;j++)
  cout <<"  "<<X[i][j];
  cout<<endl;
  }
  }
int main()
{
int X[n][m];
cout <<"\n read matrix X";
read (X);
cout <<"\n print A:"<<endl;
print (X);
getch();
return 0;
}
```

Homework:-

Q1:-Write program define 4 functions to compute area of square, area of rectangle, area of triangle and area of circle.

Q2:-Write program define 2 functions to compute circumference to circle and triangle.

Q3:-Write program define 4 functions to read and print two arrays of size (10) and third function to merge in new array, fourth function find maximum and minimum number in new array.

Q4:-Write program define four functions first to read arrays X(5) Y(6),second to sort X in acending order,third to sort Y in decending oreder.