Subtopics -

- **Functions**
- **Function Declaration**
- **Function Arguments**
- **Return Statements and values**

FUNCTIONS

- Functions are building blocks of the programs.
- They make the programs more modular and easy to read and manage.
- All C++ programs must contain the function main( ).
- The execution of the program starts from the function main( ).

Function is divided into three sections

- Function Declaration
- Function Call
- Function Definition

Form of Function

Syntax: Function Declaration

```
return_type    function_name(parameter list) ;
```

Syntax : Function Call

```
int main()
 {
    Function_name(parameter list);
 }
```

Syntax: Function Definition

```
 {
      _ _ _ _ _ _
       _ _  _ _ _ _
 }
```

Function Declaration

- A function declaration is made by declaring the return type of the function, name of the function and the data types of the parameters of the function.
- Always terminated by semicolon.
- The general form of function declaration :-

  return_type function_name(parameter list);

- The return_type specifies the type of the data the function returns.
- The parameter list could be empty .
- The parameter list should contain both data type and name of the variable.
- For example,

int factorial(int n, float j)

Function Arguments

- Arguments contain the actual value which is to be passed to the function when it is called.
- The sequence of the arguments in the call of the function should be same as the sequence of the parameters in the parameter list of the declaration of the function.
- When a function call is made arguments replace the parameters of the function.

The Return Statement and Return values

- A return statement is used to exit from the function where it is.
- It returns a value to the calling code.
- The general form of the return statement is:-

  return expression;

Example

```
#include<iostream.h>
#include<conio.h>
int factorial(int n);
 int main ()
{
    int n1,fact;
    cout <<"Enter the number whose factorial has to be calculated"
   <<  endl;
    cin >> n1;
            fact=factorial(n1);
    cout << "The factorial of " << n1 << "  is : " << fact << endl;
           getch();
return(0);
```

```
 }
int factorial(int n)
{
    int i=0,fact=1;
        if(n<=1)
        {
          return(1);
        }
         else
         {
           for(i=1;i<=n;i++)
           {
               fact=fact*i;
           }
               return(fact);
            }

}
```

Subtopics-

- **Parameters Pass/Call by Value**
- **Parameters Pass/Call by Reference**
- **Return by Reference**

Pass/Call by value

- Copies of the arguments are created .
- The parameters are mapped to the copies of the arguments created.
- The changes made to the parameter do not affect the arguments.

Example

```
 #include<iostream.h>
#include<conio.h>
int add(int n);

int main()
{
        int number,result;
        number=5;
cout << " The initial value of number : " << number <<
   endl;
        result=add(number);
```

```
cout << " The final value of number : " << number << endl;
        cout << " The result is : " << result << endl;
        getch();
        return(0);
}

int add(int number)
{
        number=number+100;
        return(number);
}
```

Pass/Call by reference

- Pass by reference is the second way of passing parameters to the function.
- The address of the argument is copied into the parameter.
- The changes made to the parameter affect the arguments.

Example-

```
#include<iostream.h>
#include<conio.h>
void swap(int &a,int &b)
{
    int t=a;
    a=b;
    b=t;
    }
int main()
    {
        int m=1,n=2;
```

```
cout<<"Value of m before swaping\t"<<m<<endl;
cout<<"Value of n before swaping\t"<<n<<endl;
        swap(m,n);
      cout<<"Value of m after swaping\t"<<m<<endl;
      cout<<"Value of n after swaping\t"<<n<<endl;
        getch();
      }
```

Return by reference

- A function can also return a reference.
- Example:

```
#include<iostream.h>
#include<conio.h>
int &max(int &x,int &y)
{
    if(x>y)
    return x;
    else
    return y;
```

```
}
int main()
{
   int m=1,n=2;
   max(m,n)=4;
   cout<<"Value of  m"<<m<<endl;
    cout<<"value of n"<<n<<endl;
    getch();
    return 0;
}
```

Subtopics-

- Inline functions
- Default arguments
- Function overloading

Inline Functions

- An inline function is a function that expanded in line when it is invoked.
- That is the compiler replaces the function call with the corresponding function code .
- **Syntax:**

```
inline function-header
{
    Function body
}
```

Example:

```
#include <iostream.h>
#include<conio.h>
int multiply(int);
int main( )
{
int x;
cout<< "\n Enter the Input Value: ";
cin>>x;
```

```
cout<<"\n The Output is: " << multiply(x);
getch();
}
inline int multiply(int x1)
{
return 5*x1;
}
```

Default Arguments

- Default values are specified when the function is declared.
- Compier looks at the prototype to see how many arguments function uses.
- Default arguments are useful in situations where some arguments always have the same value.

Example

```
#include<iostream.h>
#include<conio.h>
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 argumennts
   amount = value(5000.00,5);    //default for 3rd argument
```

```
cout<<"\n Final value = "<<amount<<"\n\n";
   printline('=');        //use default value for 2nd argument
   return 0;
   }
   float value(float p, int n, float r)
   {
       int year =1;
        float sum = p;
       while(year <= n)

{
            sum = sum*(1+r);
            year = year+1;
        }
```

```
getch();
        return(sum);
           }
   void printline(char ch, int len)
   {
       for(int i=1;i<=len;i++)
        printf("%ch",ch);
        printf("\n");
        }
```

Function Overloading

- A function is overloaded when same name is given to different function.
- The two functions with the same name will differ at least in one of the following.
  a) The number of parameters
  b) The data type of parameters
  c) The order of appearance

Example

```
#include <iostream.h>
#include<conio.h>
class arithmetic {
public:
    void calc(int num1)
{
cout<<"Square of a given number: " <<num1*num1 <<endl;
}
 void calc(int num1, int num2 )
{
```

```
cout<<"Product of two whole numbers: "
  <<num1*num2 <<endl;
}
};
int main() //begin of main function
{
    arithmetic a;
    a.calc(4);
    a.calc(6,7);
    getch();
}
```