FUNCTIONS

Prof.J.Rexy

Computer Science Department

FUNCTION

- A function is a group of statements that together perform a task. Every C++ program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.

SYNTAX

```
return_type function_name( parameter list )
{
 body of the function
 }
```

FUNCTION

- **Return Type**: A function may return a value. The **return_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword **void**.
- **Function Name:** This is the actual name of the function. The function name and the parameter list together constitute the function signature.
- **Parameters:** A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
- **Function Body:** The function body contains a collection of statements that define what the function does.

EXAMPLE 1

```
// function returning the max between two numbers
 int max(int num1, int num2)
 {
int result;
 if (num1 > num2)
result = num1;
Else
 result = num2;
 return result;
}
```

EXAMPLE 2

```
#include < iostream>
int sum (int x, int y);
 //declaring function
 int main()
 {
int a = 10;
 int b = 20;
int c = sum (a, b);
cout << c;
}
 int sum (int x, int y)
{
return (X + y);
 }
```

INLINE FUNCTION

Introduction

- Inline functions can be implemented using the keyword ***inline***.
- The inline is a request to the compiler.
- The function always can not be inline by the compiler
- The complicated functions will not be inline.
- Just like a macro, but different from macro.

- C++ proposes a new feature called inline functions.
- By using this we can eliminate the cost of calls to small functions.
- An inline function is a function that in expanded in line when it is invoked.
- Inline function must be defined before they are called.
- Inline keyword sends a request, not a command to the compiler.

SYNTAX

```
Inline function-header
{
        function body
}
```

Workings of Inline

- The inline function is just a code replacement instead of the function call.
- There is a need of stack storage and other special mechanism for function call and return.
- The stack storage is used to store the return value and return address for function call and return process.
- And while passing the parameter the stack storage needed to store the parameter values, and from the stack area the values moved to data area.

EXAMPLE

```
#include<iostream.h>
#include<conio.h>

class line
{
  public:
 inline float mul(float x,float y)
            {
                  return(x*y);
            }
  inline floatcube(float x)
            {
                  return(x*x*x);
            }
};
```

```
void main()
{
            line obj;
            float val1,val2;
            clrscr();
            cout<<"Enter two values:";
            cin>>val1>>val2;
            cout<<"\nAns"<<obj.mul(val1,val2);
          cout<<"\n\nCube   :"<<obj.cube(val1)
<<"\t"<<obj.cube(val2);
            getch();
}
```

OUTPUT

Enter two values: 5 7

Multiplication Value is: 35

Cube Value is: 25 and 343

Function Overloading

- If any class have multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class.

Ways to overload a function

- By changing number of Arguments.
- By having different types of argument.

Number of Arguments different

```
int sum (int x, int y)
 {
 cout << x+y;
 }
int sum(int x, int y, int z)
{
cout << x+y+z;
 }
```

Calling

- int main()
- {
- sum (10,20); // sum() with 2 parameter will be called
- sum(10,20,30); //sum() with 3 parameter will be called }

Different Datatype of Arguments

```
int sum(int x,int y)
{
 cout<< x+y;
}
double sum(double x,double y)
 {
 cout << x+y;
 }
 int main()
 {
 sum (10,20);
sum(10.5,20.5);
}
```