Function Overloading in C++

More than one function with same name, with different signature in a class or in a same scope is called function overloading. Signature of function includes -

- Number of arguments
- Type of arguments
- Sequence of arguments

When you call an overloaded function, the compiler determines the most appropriate definition to use by comparing the signature of calling statement with the signature specified in the definitions.

Example of function overloading -

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

class CalculateArea
{
        public:
                void Area(int r)                        //Overloaded Function 1
                {
                        cout<<"\n\tArea of Circle is : "<<3.14*r*r;
                }

                void Area(int l,int b)                  //Overloaded Function 2
                {
                        cout<<"\n\tArea of Rectangle is : "<<l*b;
                }

                void Area(float l,int b)                //Overloaded Function 3
                {
                        cout<<"\n\tArea of Rectangle is : "<<l*b;
                }

                void Area(int l,float b)                //Overloaded Function 4
                {
                        cout<<"\n\tArea of Rectangle is : "<<l*b;
                }
};
```

```
void main()
{
        CalculateArea C;

         C.Area(5);                 //Statement 1
         C.Area(5,3);               //Statement 2
         C.Area(7,2.1f);            //Statement 3
         C.Area(4.7f,2);            //Statement 4
}
```

In the above example, we have four member functions named Area. Statement 1 will invoke the function 1 b'coz the signature of function 1 is similar to the statement 1. Similarly Statement 3 will invoke function 4 b'coz statement 3 is passing two arguments, 1st is of integer type and 2nd is of float type. Function 4 is the only function who is receiving integer and float respectively.