### 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.

Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload the function.

#### Ways to overload a function

1. By changing number of Arguments.
2. By having different types of argument.

#### Number of Arguments different

In this type of function overloading we define two functions with same names but different number of parameters of the same type. For example, in the below mentioned program we have made two sum() functions to return sum of two and three integers.

int sum (int x, int y)

{

cout << x+y;

}

int sum(int x, int y, int z)

{

cout << x+y+z;

}

Here sum() function is overloaded, to have two and three arguments. Which sum() function will be called, depends on the number of arguments.

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

In this type of overloading we define two or more functions with same name and same number of parameters, but the type of parameter is different. For example in this program, we have two sum() function, first one gets two integer arguments and second one gets two double 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);

}

### Default Arguments

When we mention a default value for a parameter while declaring the function, it is said to be as default argument. In this case, even if we make a call to the function without passing any value for that parameter, the function will take the default value specified.

sum(int x,int y=0)

{

cout << x+y;

}

Here we have provided a default value for y, during function definition.

int main()

{

sum(10);

sum(10,0);

sum(10,10);

}

Output : 10 10 20 First two function calls will produce the exact same value.

for the third function call, y will take 10 as value and output will become 20.

By setting default argument, we are also overloading the function. Default arguments also allow you to use the same function in different situations just like function overloading.

#### Rules for using Default Arguments

1. Only the last argument must be given default value. You cannot have a default argument followed by non-default argument.

sum (int x,int y);

sum (int x,int y=0);

sum (int x=0,int y); // This is Incorrect

1. If you default an argument, then you will have to default all the subsequent arguments after that.

sum (int x,int y=0);

sum (int x,int y=0,int z); // This is incorrect

sum (int x,int y=10,int z=10); // Correct

1. You can give any value a default value to argument, compatible with its datatype.

#### Placeholder Arguments

When arguments in a function are declared without any identifier they are called placeholder arguments.

void sum (int,int);

Such arguments can also be used with default arguments.

void sum (int, int=0);