Subject Name: Object Oriented Programming Using C++

Subject Code: BCA-301 N

Subject Topic: Introduction to Function Overloading

Abhishek Dwivedi

Assistant Professor

Department of Computer Application

UIET, CSJM University, Kanpur

Function Overloading in C++

- Function overloading is a feature in C++ where two or more functions can have the same name but different parameters.

- Function overloading can be considered as an example of polymorphism feature in C++.

Different ways to Overload a Function

- By changing number of Arguments.

- By having different types of argument.

Function Overloading: Different Number of Arguments

- In this type of function overloading we define many functions with same names but different number of parameters of the same type.

```
// first definition
 int sum (int x, int y)
{ cout << x+y; }
 // second overloaded definition
 int sum(int x, int y, int z)
 { cout << x+y+z; }
int main()
{
     // sum() with 2 parameter will be called
        sum (10, 20);
    //sum() with 3 parameter will be called
        sum(10, 20, 30);
}
```

Function Overloading: Different Datatype of Arguments

- In this type of overloading we define many functions with same name and same number of parameters, but the type of parameter is different.

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

Constructor Overloading in C++

- In C++, We can have more than one constructor in a class with same name, as long as each has a different list of arguments. This concept is known as Constructor Overloading and is quite similar to function overloading.

1. Overloaded constructors essentially have the same name (name of the class) and different number of arguments.
2. A constructor is called depending upon the number and type of arguments passed.
3. While creating the object, arguments must be passed to let compiler know, which constructor needs to be called.

Example

```
Class ConstructorOverload
{
    public:
            int area;
             ConstructorOverload()           // Constructor with no parameters
            {
             area = 0;
            }
            ConstructorOverload(int a, int b) // Constructor with two parameters
            {
             area = a * b;
            }

            void displayArea()
            {
            cout<< area<< endl;
            }
};
```

```
void main()
{
    // Constructor Overloading
    // with two different constructors
    // of class name
     ConstructorOverload  obj1;
     ConstructorOverload  obj2( 10, 20);

    obj1.displayArea();
    obj2.displayArea();
    getch();
}
```

Functions that can not be Overloaded

- When function signatures are same, only the return type is different, then we cannot overload the function.

```
int my_func()
{    return 5; }
char my_func()
 {    return 'd'; }
```

- Member function declarations with the same name and the name parameter-type-list cannot be overloaded if any of them is a static member function declaration.

```
class My_Class
{    static void fun(int x)
{       //Something    }
void fun(int x)
 {       //something    } };
```

- Parameter declarations that differ only in a pointer * versus an array [] are equivalent. That is, the array declaration is adjusted to become a pointer declaration. Only the second and subsequent array dimensions are significant in parameter types.

```
int fun(int *ptr);
int fun(int ptr[]);          // redeclaration of fun(int *ptr)
```

- Parameter declarations that differ only in the presence or absence of const and/or volatile are equivalent.

```
int my_func(int x)
 {   //Do something }
int my_func(const int x)
 {   //do something }
```

References:

- www.studytonight.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- “Object oriented programming in C++”, Robert Lafore
- “Object oriented programming with C++”, E.Balagurusamy