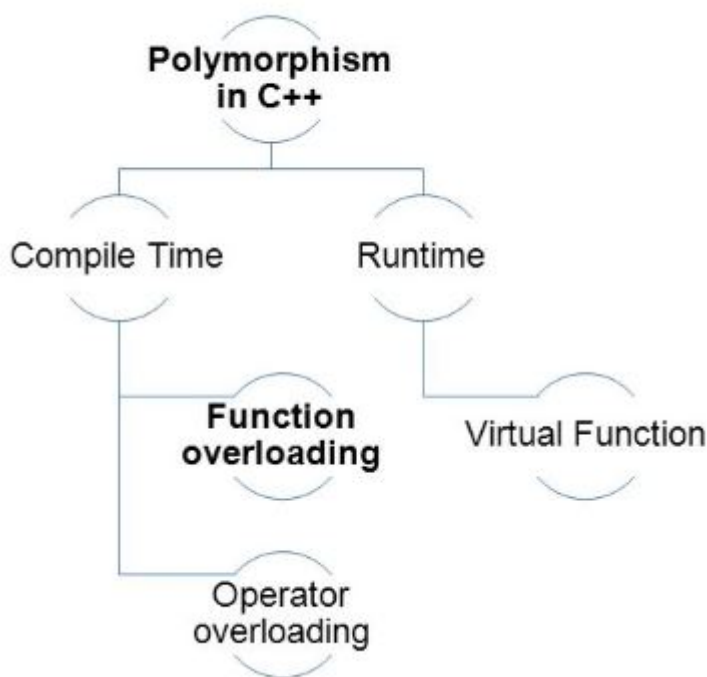


Course Code: CL-217	Course : Object Oriented Programming Lab
Instructor(s)	Muhammad Fahim, Muhammad Irfan Ayub, Mubashara Fayyaz, Aqsa Zahid, Nida Munawar

C++ is an Object Oriented Programming Language in nature and it boasts off various features. In this session we would be discussing how to implement Function Overloading And Function Overriding in C++.

Polymorphism on C++:

- Function Overloading
- Function Overriding



Function Overloading

Functions having the same name but different parameters is allowed in C++ and is called **Function Overloading**. It is also called compile-time Polymorphism.

For example:

```

1 sum( int a, float b)
2 sum(int a, int b)
3 sum(int a, int b, int c)
  
```

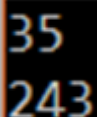
Here, there are three functions with the same name but the only thing that differentiates them is that the parameters are different on each. So, depending on the parameters passed, a function is called.

If the return types of the functions are different then it is considered invalid.

Sample Code For Function Over Loading

```
1  include <iostream>
2  using namespace std;
3  class Addition {
4  public:
5  int add(int n1,int n2) {
6  return n1+n2;
7  }
8  int add(int n1,int n2,int n3) {
9  return n1+n2;
10 }
11 };
12 int main(void) {
13 Addition a;
14 cout<<a.add(20, 15)<<endl;
15 cout<<a.add(81, 162,21);
16 return 0;
17 }
```

Output



```
35
243
```

Explanation

In the program above, we have two functions in the addition class. Both named add. One has 2 parameters and the other has 3 parameters.

In the main function, we create an object of class addition called a. We call the add functions with 2 and 3 parameters respectively and the functions add are called and they perform addition.

This is how the function overloading takes place.

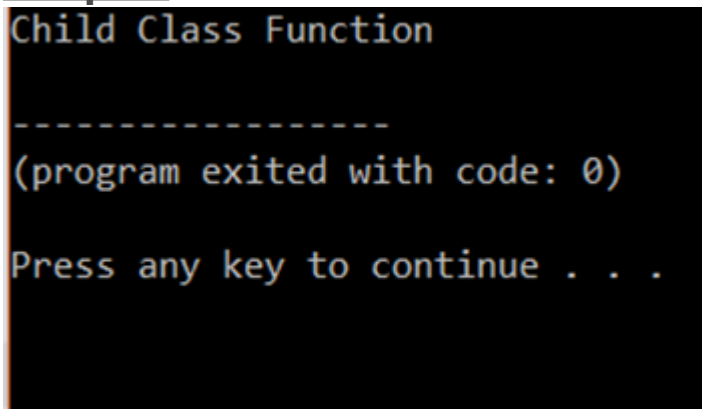
Function Overriding

When a derived class has a function with the same name as a function of the base class, it is called *Function Overriding*. Both functions must have the same parameters in both classes.

Sample Code For Function Overriding

```
1
2 #include <iostream>
3 using namespace std;
4 class BaseClass {
5 public:
6 void disp(){
7 cout<<"Parent Class Function";
8 }
9 };
10 class DerivedClass: public BaseClass{
11 public:
12 void disp() {
13 cout<<"Child Class Function";
14 }
15 };
16 int main() {
17 \DerivedClass obj = DerivedClass();
18 obj.disp();
19 return 0;
20 }
```

Output:



```
Child Class Function
-----
(program exited with code: 0)

Press any key to continue . . .
```

Explanation:

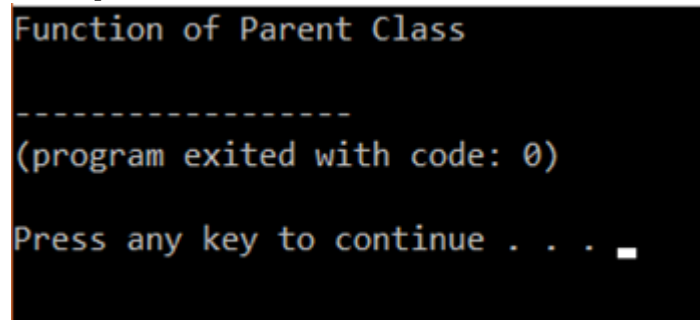
In the program above, we show basic function, with the same name in derived and base class. Here the object is created of the derived class so when we call display only the child class object is displayed.

Order to Perform Overriding

Consider the code:

```
1
2 #include <iostream>
3 using namespace std;
4 class BaseClass {
5 public:
6 void disp(){
7 cout<<"Function of Parent Class";
8 }
9 };
10 class DerivedClass: public BaseClass{
11 public:
12 void disp() {
13 cout<<"Function of Child Class";
14 }
15 };
16 int main() {
17 BaseClass obj = DerivedClass();
18 obj.disp();
19 return 0;
20 }
```

Output:



```
Function of Parent Class
-----
(program exited with code: 0)
Press any key to continue . . .
```

Explanation:

In the program above, we show basic function, with same name in derived and base class. Here, the only difference from the previous program is that. We create the object of the child class. The child class object is given the reference of base class. This can also be done by using another method,

Syntax

```
1 Parent_class_name::function()
```

In the above example, we use it as,

```
1 BaseClass::disp()
```

This is another way of overriding.

Function Overriding using Base Class Pointer & Virtual Function:

```
#include<iostream>
using namespace std;
class Base
{
public:
virtual void show() // virtual function
{
cout << "Base class";
}
};
class Derived:public Base
{
public:
void show()
{
cout << "Derived Class";
}
};

int main()
{
Base* b;    //Base class pointer
Derived d;  //Derived class object
b = &d; // passing derived class address into base class pointer
b->show();  //Late Binding Occurs
}
```

Output

Total Objects: 1
Derived Class

Program Explanation:

In this program, we made the base class's show function as virtual. This enables us to use the base class's object to call the functions of derived class which are overridden and virtual. Thus in the main function you can see that we created a

base class pointer object and made it to point derived class object by passing derived class objects address. Then using the (->) operator we called the derived class objects show function which prints the output as “Derived Class”. This is known as Dynamic Polymorphism or Late binding.

Function Overloading VS Function Overriding

Function Overloading	Function Overriding
The scope is the same	The scope is different
Signatures must differ (ex: parameter)	Signatures must be same
Number of overloading functions possible	Only one overriding function possible
May occur without inheritance	It mainly occurs due to inheritance

Benefits of Function Overloading and Overriding:

- The one main advantage of these overriding and overloading is time-saving.
- Save memory space.
- The readability of the code is increased.
- Here, for function overloading concept, we can use different same function names for different operations eliminating the use of different function names.
- Flexibility and maintainability of code become easier.
- In the case of overriding, the child class can have functions of parent class and can even have its own implementation of that function.
- The objects can be referenced and the functions of both the parent class and child class can be accessed by the child class.

Exercises:

1. Create a Message class with a constructor that takes a single string with a default value. Create a private member string, and in the constructor simply assign the argument string to your internal string. Create two overloaded member functions called print(): one that takes no arguments and simply prints the message stored in the object, and one that takes a string argument, which it prints in addition to the internal message.
2. Write a program to find the area of a rectangle by passing length and breadth as arguments after creating member function in both derived and base class with the same name by technique of function overriding.
3.
Create a base class area of shape having two parameters (a,b) which can be overloaded in derived class in rectangle and square. Create a class to print the area of a square and a rectangle. The class has two methods with the same name but different number of parameters. The method for printing area of rectangle has two parameters which are length and breadth respectively while the other method for printing area of square has one parameter which is side of square.
4. A boy has his money deposited \$1000, \$1500 and \$2000 in banks- Bank A, Bank B and Bank C respectively. We have to print the money deposited by bank.
Create a class 'Bank' with a method 'getBalance' which returns 0. Make its three subclasses named 'BankA', 'BankB' and 'BankC' with a method with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the method 'getBalance' by the object of each of the three banks.