Course Name: **A Level (2nd Sem)**
Topic: **Function Overriding**
Subject: **Data Struture using C++**
Date: **27-03-2020**

Function Overriding

Function overriding is a feature that allows us to have a same function in child class which is already present in the parent class. A child class inherits the data members and member functions of parent class, but when you want to override functionality in the child class then you can use function overriding. It is like creating a new version of an old function, in the child class.

Function Overriding Example

To override a function you must have the same signature in child class. By signature I mean the data type and sequence of parameters. Here we don't have any parameter in the parent function so we didn't use any parameter in the child function.

```
#include <iostream>
using namespace std;
class BaseClass {
public:
  void disp(){
    cout<<"Function of Parent Class";
  }
};
class DerivedClass: public BaseClass{
public:
  void disp() {
    cout<<"Function of Child Class";
  }
};
int main() {
  DerivedClass obj = DerivedClass();
  obj.disp();
  return 0;
}
```

Output:

```
Function of Child Class
```

Note: In function overriding, the function in parent class is called the overridden function and function in child class is called overriding function.

Example 1: Program to illustrate Function Overloading

```
#include <iostream>
using namespace std;

// overloaded functions
void test(int);
void test(float);
void test(int, float);

int main()
{
   int a = 5;
   float b = 5.5;
```

```
    // Overloaded functions
    // with different type and
    // number of parameters
    test(a);
    test(b);
    test(a, b);

    return 0;
}

void test(int var)
{
    cout << "Integer number: " << var << endl;
}

void test(float var)
{
    cout << "Float number: "<< var << endl;
}

void test(int var1, float var2)
{
    cout << "Integer number: " << var1;
    cout << " and float number:" << var2;
}
```

<u>Output:</u>

```
                    Integer number: 5
                    Float number: 5.5
                    Integer number: 5 and float number: 5.5
```

Function Overriding (achieved at run time)

It is the redefinition of base class function in its derived class with same signature i.e return type and parameters.

It can only be done in derived class.

Example:

```
Class a
{
public:
      virtual void display(){ cout << "hello"; }
}
Class b:public a
{
public:
       void display(){ cout << "bye";};
}
```

<u>Program to illustrate Function Overriding</u>

```
#include<iostream>
using namespace std;

class BaseClass
{
```

```
public:
   virtual void Display()
   {
      cout << "\nThis is Display() method"
            " of BaseClass";
   }
   void Show()
   {
      cout << "\nThis is Show() method "
           "of BaseClass";
   }
};

class DerivedClass : public BaseClass
{
public:
   void Display()
   {
      cout << "\nThis is Display() method"
           " of DerivedClass";
   }
};
int main()
{
   DerivedClass dr;
   BaseClass &bs = dr;
   bs.Display();
   dr.Show();
}
```

Output:

```
This is Display() method of DerivedClass
This is Show() method of BaseClass
```

Function Overloading VS Function Overriding

Inheritance: Overriding of functions occurs when one class is inherited from another class. Overloading can occur without inheritance.

Function Signature: Overloaded functions must differ in function signature ie either number of parameters or type of parameters should differ. In overriding, function signatures must be same.

Scope of functions: Overridden functions are in different scopes; whereas overloaded functions are in same scope.

Behavior of functions: Overriding is needed when derived class function has to do some added or different job than the base class function. Overloading is used to have same name functions which behave differently depending upon parameters passed to them.