**Function Overloading vs Function Overriding in C++**

**Function overloading (achieved at compile time)**

It provides multiple definitions of the function by changing signature i.e changing number of parameters, change datatype of parameters, return type doesn’t play any role.

* It can be done in base as well as derived class.
* **Example:**
* void area(int a);
* void area(int a, int b);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | // CPP 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;  }    // Method 1  void test(int var)  {      cout << "Integer number: " << var << endl;  }    // Method 2  void test(float var)  {      cout << "Float number: "<< var << endl;  }    // Method 3  void test(int var1, float var2)  {      cout << "Integer number: " << var1;      cout << " and float number:" << var2;  } | // CPP program to illustrate  // Function Overriding  #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:      // Overriding method - new working of      // base class's display method      void Display()      {          cout << "\nThis is Display() method"                 " of DerivedClass";      }  };    // Driver code  int main()  {      DerivedClass dr;      BaseClass &bs = dr;      bs.Display();      dr.Show();  } | | Integer number: 5  Float number: 5.5  Integer number: 5 and float number: 5.5 | This is Display() method of DerivedClass  This is Show() method of BaseClass | |

**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";};
* }

|  |
| --- |
|  |

**Function Overloading VS Function Overriding**

1. **Inheritance:** Overriding of functions occurs when one class is inherited from another class. Overloading can occur without inheritance.
2. **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.
3. **Scope of functions:** Overridden functions are in different scopes; whereas overloaded functions are in same scope.
4. **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.