



Code Example Demonstrating Virtual Base Class in C++ | C++ Tutorials for Beginners #45



[Overview](#) [Q&A](#) [Downloads](#) [Announcements](#)

Code Example Demonstrating Virtual Base Class in C++ | C++ Tutorials for Beginners #45

In this tutorial, we will discuss demonstrating of virtual base class in C++

Virtual Base Class in C++

The virtual base class is a concept used in multiple inheritances to prevent ambiguity between multiple instances. For example: suppose we created a class “Student” and two classes “Test” and “Sports”, are being derived from class “Student”. But once we create a class “Result” which is being derived from class “Test” and “Sports” as shown in figure 1.

Virtual base class
↑

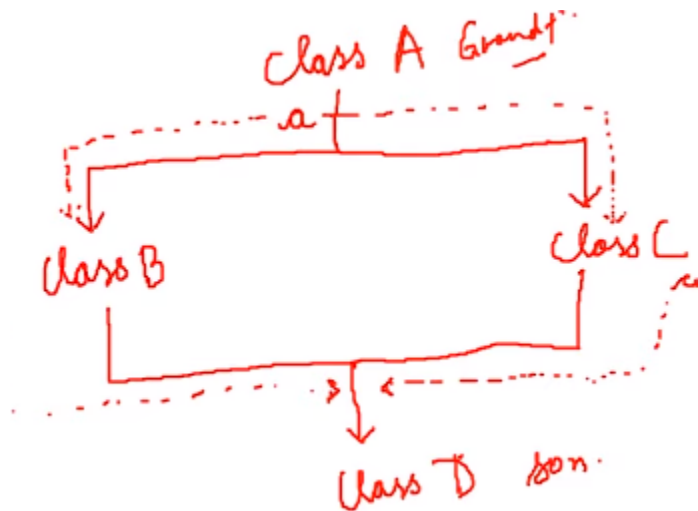


Figure 1: Virtual Base Class Example Diagram

As shown in figure 1,

1. Class "Student" is a parent class of two classes "Test" and "Sports"
2. And both "Test" and "Sports" classes are the parent of class "Result"

The main thing to note here is that the data members and member functions of class "Student" will be inherited twice in class "Result" because class "Test" and "Sports" are the parent classes of class "Result" and they both are being derived from class "Student".

So when the class "Result" will try to access the data member or member function of class "Student" it will cause ambiguity for the compiler and the compiler will throw an error. To solve this ambiguity we will make class "Student" as a virtual base class. To make a virtual base class "virtual" keyword is used.

When one class is made virtual then only one copy of its data member and member function is passed to the classes inheriting it. So in our example when we will make class "Student" a virtual class then only one copy of data member and member function will be passed to the classes "Test" and "Sports" which will be shared between all classes. This will help to solve the ambiguity.

An example program of the following diagram is shown in a code snippet below,

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

class Student{
    protected:
        int roll_no;
    public:
        void set_number(int a){
            roll_no = a;
        }
        void print_number(void){
            cout<<"Your roll no is "<< roll_no<<endl;
        }
};

class Test : public Student{
    protected:
        float maths, physics;
    public:
        void set_marks(float m1, float m2){
            maths = m1;
            physics = m2;
        }

        void print_marks(void){
            cout << "You result is here: "<<endl
```

```
        << "Maths: "<< maths<<endl
        << "Physics: "<< physics<<endl;
    }

};

class Sports: public Student{
protected:
    float score;
public:
    void set_score(float sc){
        score = sc;
    }

    void print_score(void){
        cout<<"Your PT score is "<<score<<endl;
    }

};

class Result : public Test, public Sports{
private:
    float total;
public:
    void display(void){
        total = maths + physics + score;
        print_number();
    }
};
```

```
        print_marks();  
        print_score();  
        cout<< "Your total score is: "<<total<<endl;  
    }  
};
```

Code Snippet 1: Virtual Base Class Example Program

As shown in a code snippet 1,

1. We have created a “Student” class that consists of protected data member “roll_no” and member functions “set_number” and “print_number”. The function “set_number” will assign the value to the protected data member “roll_no” and the function “print_number” will print the value of data member “roll_no”.
2. We have created a “Test” class that is inheriting the virtual base class “Student”. The “Test” consists of protected data members “maths” and “physics” and member functions “set_marks” and “print_marks”. The function “set_number” will assign the values to the protected data members “maths” and “physics” and the function “print_marks” will print the value of data members “maths” and “physics”.
3. We have created a “Sports” class that is inheriting the virtual base class “Student”. The “Sports” consists of protected data member “score” and member functions “set_score” and “print_score”. The function “set_score” will assign the values to the protected data members “score” and “physics” and the function “print_score” will print the value of data members “score”.
4. We have created a “Result” class which is inheriting base classes “Test” and “Sports”. The “Result” consists of protected data member “total” and member functions “display”. The function “display” will first add the values of data members “math”, “physics”, and “score” and assign the value to the protected data members “total” and second the “display” function will call the functions “print_number”, “print_marks”, and “print_score” and also print the value of the data member “total”.

The code of the main function is shown below,

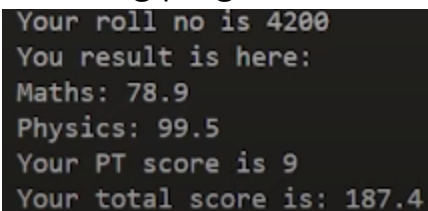
```
int main(){
    Result harry;
    harry.set_number(4200);
    harry.set_marks(78.9, 99.5);
    harry.set_score(9);
    harry.display();
    return 0;
}
```

Code Snippet 2: Main Program

As shown in code snippet 2,

1. Object “harry” is created of the “Result” data type.
2. The function “set_number” is called by the object “harry” and the value “4200” is passed.
3. The function “set_marks” is called by the object “harry” and the values “48.9” and “99.5” are passed.
4. The function “set_score” is called by the object “harry” and the value “9” is passed.
5. The function “display” is called by the object “harry”.

The main thing to note here is that there will be no ambiguity because we have made the “Student” class as a virtual base class but if we remove the “virtual” keyword then the compiler will throw an error. The output of the following program is shown below.



```
Your roll no is 4200
You result is here:
Maths: 78.9
Physics: 99.5
Your PT score is 9
Your total score is: 187.4
```

Figure 2: Program Output

[Previous](#)

[Next](#)



CodeWithHarry

Copyright © 2022 CodeWithHarry.com

