Virtual Base Class in C++

```
#include <iostream>
using namespace std;
class A {
public:
   void say()
   {
      cout << "Hello world"<<endl;
   }
};
class B : public virtual A {
};
class C : public virtual A {
};
class D : public B, public C {
};
```

As shown in figure 1,

1. Class “A” is a parent class of two classes “B” and “C”
2. And both “B” and “C” classes are the parent of class “D”

The main thing to note here is that the data members and member functions of class “A” will be inherited twice in class “D” because class “B” and “C” are the parent classes of class “D” and they both are being derived from class “A”.
So when the class “D” will try to access the data member or member function of class “A” it will cause ambiguity for the compiler and the compiler will throw an error. To solve this ambiguity we will make class “A” 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 “A” a virtual class then only one copy of the data member and member function will be passed to the classes “B” and “C” which will be shared between all classes. This will help to solve the ambiguity.

Pointers to Derived Classes in C++

```
#include<iostream>
using namespace std;
class BaseClass{
   public:
      int var_base;
      void display(){
         cout<<"Dispalying Base class variable var_base "<<var_base<<endl;
      }
};

class DerivedClass : public BaseClass{
   public:
         int var_derived;
         void display(){
            cout<<"Dispalying Base class variable var_base "<<var_base<<endl;
            cout<<"Dispalying Derived class variable var_derived "<<var_derived<<endl;
         }
};
int main(){
   BaseClass * base_class_pointer;
```

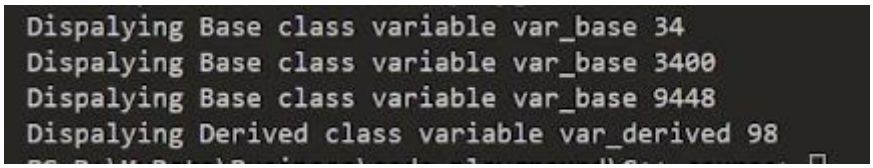
```
    BaseClass obj_base;
    DerivedClass obj_derived;
    base_class_pointer = &obj_derived; // Pointing base class pointer to derived class

    base_class_pointer->var_base = 34;
    // base_class_pointer->var_derived= 134; // Will throw an error
    base_class_pointer->display();

    base_class_pointer->var_base = 3400;
    base_class_pointer->display();

    DerivedClass * derived_class_pointer;
    derived_class_pointer = &obj_derived;
    derived_class_pointer->var_base = 9448;
    derived_class_pointer->var_derived = 98;
    derived_class_pointer->display();

    return 0;
}
```



Virtual Functions in C++

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

class BaseClass{
    public:
```

```
    int var_base=1;
    virtual void display(){
        cout<<"1 Dispalying Base class variable var_base "<<var_base<<endl;
    }
};

class DerivedClass : public BaseClass{
  public:
      int var_derived=2;
      void display(){
          cout<<"2 Dispalying Base class variable var_base "<<var_base<<endl;
          cout<<"2 Dispalying Derived class variable var_derived "<<var_derived<<endl;
      }
};
int main(){
  BaseClass * base_class_pointer;
  BaseClass obj_base;
  DerivedClass obj_derived;

  base_class_pointer = &obj_derived;
  base_class_pointer->display();
  return 0;
}
```

```
2 Dispalying Base class variable var_base 1
2 Dispalying Derived class variable var_derived 2
```

Virtual Functions Example in C++

```
class CWH{
  protected:
    string title;
    float rating;
  public:
```

```
        CWH(string s, float r){
            title =  s;
            rating = r;
        }
        virtual void display(){}
};
class CWHVideo: public CWH
{
    float videoLength;
    public:
        CWHVideo(string s, float r, float vl): CWH(s, r){
            videoLength = vl;
        }
        void display(){
            cout<<"This is an amazing video with title "<<title<<endl;
            cout<<"Ratings: "<<rating<<" out of 5 stars"<<endl;
            cout<<"Length of this video is: "<<videoLength<<" minutes"<<endl;
        }
};
class CWHText: public CWH
{
    int words;
    public:
        CWHText(string s, float r, int wc): CWH(s, r){
            words = wc;
        }
     void display(){
      cout<<"This is an amazing text tutorial with title "<<title<<endl;
      cout<<"Ratings of this text tutorial: "<<rating<<" out of 5 stars"<<endl;
      cout<<"No of words in this text tutorial is: "<<words<<" words"<<endl;
        }
```

```
};
int main(){
    string title;
    float rating, vlen;
    int words;

    // for Code With Harry Video
    title = "Django tutorial";
    vlen = 4.56;
    rating = 4.89;
    CWHVideo djVideo(title, rating, vlen);

    // for Code With Harry Text
    title = "Django tutorial Text";
    words = 433;
    rating = 4.19;
    CWHText djText(title, rating, words);

    CWH* tuts[2];
    tuts[0] = &djVideo;
    tuts[1] = &djText;

    tuts[0]->display();
    tuts[1]->display();

    return 0;
}
```

```
This is an amazing video with title Django tutorial
Ratings: 4.89 out of 5 stars
Length of this video is: 4.56 minutes
This is an amazing text tutorial with title Django tutorial Text
Ratings of this text tutorial: 4.19 out of 5 stars
No of words in this text tutorial is: 433 words
```

Pure Virtual Functions in C++

Abstract Base Class in C++

```
class CWH{
   protected:
      string title;
      float rating;
   public:
      CWH(string s, float r){
         title =  s;
         rating = r;
      }
      virtual void display()=0;
};
class CWHVideo: public CWH
{
   float videoLength;
   public:
      CWHVideo(string s, float r, float vl): CWH(s, r){
         videoLength = vl;
      }
      void display(){
         cout<<"This is an amazing video with title "<<title<<endl;
         cout<<"Ratings: "<<rating<<" out of 5 stars"<<endl;
         cout<<"Length of this video is: "<<videoLength<<" minutes"<<endl;
      }
};
class CWHText: public CWH
{
   int words;
   public:
```

```
    CWHText(string s, float r, int wc): CWH(s, r){
        words = wc;
    }
    void display(){
     cout<<"This is an amazing text tutorial with title "<<title<<endl;
     cout<<"Ratings of this text tutorial: "<<rating<<" out of 5 stars"<<endl;
     cout<<"No of words in this text tutorial is: "<<words<<" words"<<endl;
      }
};
int main(){
   string title;
   float rating, vlen;
   int words;

   // for Code With Harry Video
   title = "Django tutorial";
   vlen = 4.56;
   rating = 4.89;
   CWHVideo djVideo(title, rating, vlen);

   // for Code With Harry Text
   title = "Django tutorial Text";
   words = 433;
   rating = 4.19;
   CWHText djText(title, rating, words);

   CWH* tuts[2];
   tuts[0] = &djVideo;
   tuts[1] = &djText;

   tuts[0]->display();
```

```
    tuts[1]->display();


    return 0;
}
```

```
tut58.cpp:14:22: note:  'virtual void CWH::display()'
        virtual void display()=0; // do-nothing function --> pure virtual function
                     ^~~~~~~
```

```
This is an amazing video with title Django tutorial
Ratings: 4.89 out of 5 stars
Length of this video is: 4.56 minutes
This is an amazing text tutorial with title Django tutorial Text
Ratings of this text tutorial: 4.19 out of 5 stars
No of words in this text tutorial is: 433 words
```