C++ virtual function

* A C++ virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword. It is used to tell the compiler to perform dynamic linkage or late binding on the function.
* There is a necessity to use the single pointer to refer to all the objects of the different classes. So, we create the pointer to the base class that refers to all the derived objects. But, when base class pointer contains the address of the derived class object, always executes the base class function.
* This issue can only be resolved by using the 'virtual' function. A 'virtual' is a keyword preceding the normal declaration of a function. When the function is made virtual,
* C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer.

Rules of Virtual Function -

1. Virtual functions must be members of some class.
2. Virtual functions cannot be static members.
3. They are accessed through object pointers.
4. They can be a friend of another class.
5. A virtual function must be defined in the base class, even though it is not used.
6. The prototypes of a virtual function of the base class and all the derived classes must be identical. If the two functions with the same name but different prototypes, C++ will consider them as the overloaded functions.
7. We cannot have a virtual constructor, but we can have a virtual destructor
8. Consider the situation when we don't use the virtual keyword.

**#include<iostream.h>**

**class A {**

**int x=5;**

**public:**

**void display() {**

**cout<< "Value of x is : " << x<<std::endl;**

**}**

**};**

**class B: public A {**

**int y = 10;**

**public:**

**void display() {**

**cout<< "Value of y is : " <<y<< std::endl;**

**}**

**};**

**void main() {**

**A \*a;**

**B b;**

**a = &b;**

**a->display();**

**}**

Output – Value of x is: 5

C++ virtual function Example -

**#include<iostream.h>**

**class A**

**{**

**public:**

**virtual void display()**

**{**

**cout << "Base class is invoked"<<endl;**

**}**

**};**

**class B:public A**

**{**

**public:**

**void display()**

**{**

**cout << "Derived Class is invoked"<<endl;**

**}**

**};**

**int main()**

**{**

**A\* a; //pointer of base class**

**B b; //object of derived class**

**a = &b;**

**a->display(); //Late Binding occurs**

**}**

**Output – Derived Class is invoked**