C++ VIRTUAL FUNCTION

Virtual function is a function in base class which is overrided in the derived class , and which tells the compiler to perform late binding on this function.

Virtual keyword is used to make a member function of the base class virtual,

VIRTUAL FUNCTION

A virtual function is a member function that is declared as virtual within a base class and redefined by a deriver class.

To create virtual function, precede the base version of function's declare with the keyword virtual.

The method name and type signature should be same for both base and derived version of function.

HOW VIRTUAL FUNCTION WORKS?

Base class pointer can point to derived class object.

In this case, using base class pointer if we call some function which is in both classes, then base class function is invoked.

But if we want to invoke derived class function using function base class pointer, it can be achieved by defining the function as virtual in base class, this is how virtual function supports runtime polymorphism.

USING VIRTUAL KEYWORD

Using virtual keyword with base class version of show function ; late binding takes place and derived of the function will be called , because base pointer pointes an derived type of object.

We know that in runtime polymorphism the call to a function is resolved at runtime depending upon the type of object.

SYNTAX

Virtual return _ type function _ name ( )

{

.....
.....
.....

}

Ex: virtual void print( )

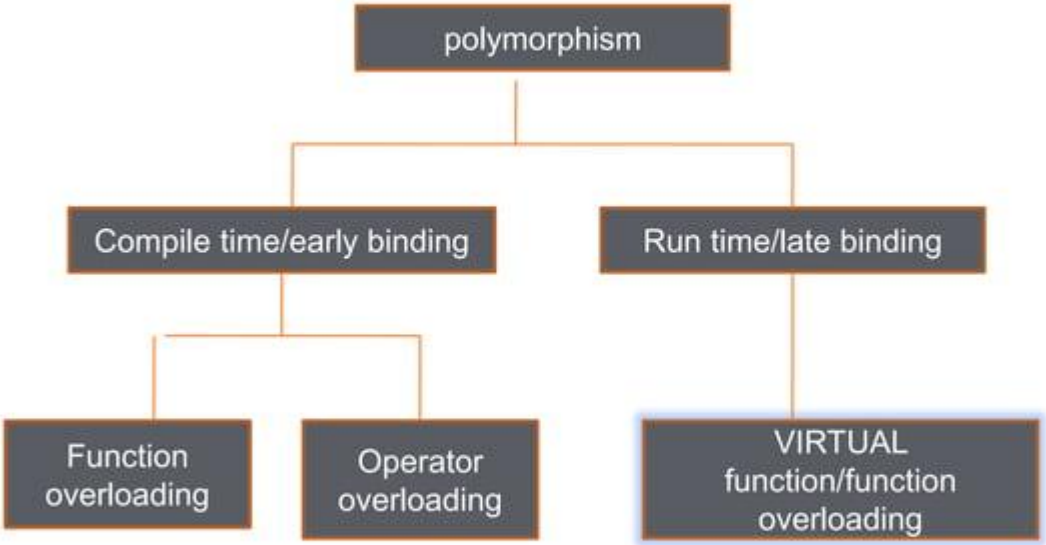
GENDRAL FORMAT

```
Class class  _ name
{

Public:

    virtual return _ type function _ name (arguments)
     {
      ....
     .....
     }
};
Class A
{
```

Virtual function belongs to the branch of runtime polymorphism in c++



EXAMPLE

```
#include<iostream.h>
Class base
{
Public:
Virtual void print ( )
{
Cout<<“print base class”;
}
Void show ( )
{
Cout<<“show base class”;
} };
Class derived : public base
{
Public:
Void print ( )
{
Cout<<“print derived class”;
}
Void show ( )
```

```
{
cout<<"show derived class";
}};
int main ( )
{
base*bptr;
bptr =d;
derived d;
bptr ⟶ print( );
bptr ⟶ show( );
}
```

OUTPUT
Print derived class
Show base class