**virtual function**

A virtual function is a member function that is declared within a base class and redefined by a derived class. To create virtual function, precede the function’s declaration in the base class with the keyword virtual. When a class containing virtual function is inherited, the derived class redefines the virtual function to suit its own needs.

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 base class pointer, it can be achieved by defining the function as virtual in base class, this is how virtual functions support **runtime polymorphism**.

**#include**<iostream>

**using** **namespace** std;

**class** aa

{

**public**:

pure virtaul **void** **test**();

**virtual** **void** **disp**(**void**)

{

cout<<"\nclass a disp";

}

**void** **test**(**void**)

{

cout<<"\ntest called";

}

};

**class** bb:**public** aa

{

**void** **test**()

{

cout<<"test";

}

};

**int** **main**()

{

aa \*a=**new** aa();

a->disp();

cout<<"\naddress of class a :"<<a;

bb obj;

cout<<"\naddress of class a :"<<&obj;

a=&obj;

cout<<"\naddress of class a :"<<a;

a->disp();

**return** 0;

}

Q:- If you don’t defined the body of derived of virtual function it will call the base class function.

**virtual table**

It’s also known as virtual table / virtual pointer / **virtual method table**, **virtual function table**, **virtual call table**, [dispatch table](http://en.wikipedia.org/wiki/Dispatch_table), or **vtable**

Whenever a class defines a [virtual function](http://en.wikipedia.org/wiki/Virtual_function) (or method), most compilers add a hidden member variable to the class which points to a so-called virtual method table (VMT or Vtable).

A virtual pointer is a pointer that will point to virtual table. Its automatically added by class if you define any function as virtual.

Virtual Table : it’s a array of pointer that’s will points to the function which is most derived.

This VMT is basically an array of pointers to (virtual) functions. At runtime these pointers will be set to point to the right function, because at compile time, it is not yet known if the base function is to be called or a derived one implemented by a class that inherits from the base class it will help to achieved the dynamic polymorices in C++

Q:- for which class virtual table is created ??

Answer :- A separate virtual table is created for any class which has

* Atleast one virtual function
* Or has a base class at any level of hierarchy which has at least one virtual function.

Example :-One

**class** data

{

//virtual table is not created because not virtual

Function.

};

**class** aa:data

{

**public**:

**virtual** **void** **show**() //virtual table is created

{

cout<<"\n class a";

}

};

**class** cc:**public** aa

{

**public**:

**void** **show**() //virtual table is created because base class has one virtual

// function but the virtual table will point to show() of cc

Class its most driven for the virtual table of cc class.\

{

cout<<"\n class b";

}

};

**class** dd:cc

{

//virtual table is created because base class has one virtual

// function but the virtual table will point to show() of cc

// Class its most driven for the virtual table of cc class.\

};

Example : - two



**Abstract class**

If you can’t create the object of the class .then aIl will become abstract class. It can be archives by defining a pure virtual function within the class**.**

**Pure virtual function:-**

1.>If you added =0 ; in the function then that function will become pure virtual function and class will become abstract class.

2.> if you don’t provide the body to the pure virtual within the derived class that it will also become abstract class.

3.> you can’t create the object of the class but you can create a pointer of the class.

**#include**<iostream>

**using** **namespace** std;

**class** aa

{

**public**:

**virtual** **void** **show**()=0;

};

**class** bb:aa

{

};

**class** cc:bb

{

**public**:

**void** **show**()

{

cout<<"\n show() called :";

}

};

**int** **main**()

{

aa \*obj; // you can create the pointer pure virtual function class

bb obj1; // error because you have not defined the pure virtual function body

cc obj2;

**return** 0;

}

4>if you try to access the pure virtual function via base class pointer I will show error

“ virtual void aa::show()' is inaccessible “

Example givien as following :-

**#include**<iostream>

**using** **namespace** std;

**class** aa

{

**public**:

**virtual** **void** **show**()=0;

};

**class** bb:aa

{

};

**class** cc:bb

{

**public**:

**void** **show**()

{

cout<<"\n show() called :";

}

};

**int** **main**()

{

bb \*obj;

obj->show();

**return** 0;

}