**Program: 1**

class A

{

public:

virtual void show()

{

cout<<"in A"<<endl;

}

};

class B:public A

{

public:

void show()

{

cout<<"in B"<<endl;

}

};

int main()

{

A \*a=new B;

**a->A::show();//call A method (it show how call itself method)**

a->B::show()//error

}

Output:in A

**Program:2(virtual Base class)**

**Virtual base class is used to avoid multiple copy of base class in the derived class.**

class A

{

int a;

};

class B:virtual public A

{

};

class C:virtual public A

{

};

class D:public B,public C

{

};

int main()

{

cout<<sizeof(A)<<endl;//4

cout<<sizeof(B)<<endl;//16

cout<<sizeof(C)<<endl;//16

cout<<sizeof(D)<<endl;//24

**Without virtual keyword it print:4 4 4 8**

}

**Program:3-a(without virtual keyword OR without virtual base class)**

using namespace std;

class A

{

int a[10];

};

class B: public A

{

};

class C: public A

{

};

class D:public B,public C

{

};

int main()

{

cout<<sizeof(A)<<endl;//40

cout<<sizeof(B)<<endl;//40

cout<<sizeof(C)<<endl;//40

cout<<sizeof(D)<<endl;//80

**using virtual keyword it print: 40 48 48 56**

}

Program:3-b(when use virtual keyword)

class A

{

int a[10];

};

class B:virtual public A

{

};

class C:virtual public A

{

};

class D:public B,public C

{

};

int main()

{

cout<<sizeof(A)<<endl;//40

cout<<sizeof(B)<<endl;//48

cout<<sizeof(C)<<endl;//48

cout<<sizeof(D)<<endl;//56

}

Program: 3-d:

class Test1

{

int x;

public:

void show() { }

};

class Test2

{

int x;

public:

virtual void show() { }

};

int main(void)

{

cout<<sizeof(Test1)<<endl; //4

cout<<sizeof(Test2)<<endl; //16

return 0;

}

Output:

4

8

 The extra pointer vptr adds to the size of objects, that is why we get 8 as size of Test2.

**Program:3-b**

**class A**

{

public:

void print()

{

cout<<"in A"<<endl;

}

};

**class B: public A**

{

public:

void print()

{

cout<<"in B"<<endl;

}

};

class C: public B

{

};

int main()

{

C c;

c.print();//in B

}

**Program:4-a**

class A

{

int a;

};

int main()

{

cout<<sizeof(A);//4

}

**Program:4-b**

class A

{

};

int main()

{

cout<<sizeof(A);//1

}

**Program: 4-c**

class A

{

static int i;

};

**int A::i=10; OR int A::I;**

int main()

{

cout<<sizeof(A);//1

}

**Program: 4-d**

class A

{

static int i;

int j;

};

int A::i=10; OR int A::I;

int main()

{

cout<<sizeof(A);//4

}

**Program:5-a**

class A

{

public:

virtual void show();

};

class B:public A

{

void show();

};

int main()

{

}

**Above program will compile and run ,no need to define virtual function in base class;**

**But when we create object of any class then it necessary to define a virtual function in base class even though it is not use in derived class:**

Program:5-b

class A

{

public:

virtual void show();

};

class B:public A

{

void show();

};

int main()

{ B b;

}

**Above program show error bz, virtual is the runtime concept,so when we create object of any class then then it necessary to define a virtual function in base class even though it is not use in derived class:**

**Program:6**

class A

{

public:

virtual void show()

{

cout<<"in A"<<endl;

}

};

class B:public A

{

void show()

{

cout<<"in B"<<endl;

}

};

int main()

{

A \*a=new B;

a->show();//in B

A &a1=\*a;//**reference work as original variable here**

a1.show();//in B

}