C++ Array of Objects

An object of class represents a single record in memory, if we want more than one record of class type, we have to create an array of class or object. As we know, an array is a collection of similar type, therefore an array can be a collection of class type.

#include <iostream>

using namespace std;

class student{

public:

int id;

string name;

void getdata(){

cout<<"enter id";

cin>>id;

cout<<"enter name";

cin>>name;

}

void display(){

cout<<id<<" "<<name<<endl;

}

};

int main()

{

int i;

student st[3];

for(i=0;i<3;i++){

cout<<"enter details"<<endl;

st[i].getdata();

}

for(i=0;i<3;i++){

st[i].display();

}

return 0 ;

}

# **C++ Ambiguity in multiple inheritance**

In multiple inheritance, there may be possibility that a class may inherit member functions with same name from two or more base classes and the derived class may not have functions with same name as those of its base classes. If the object of the derived class need to access one of the same named member function of the base classes then it result in ambiguity as it is not clear to the compiler which base’s class member function should be invoked. The ambiguity simply means the state when the compiler confused.

#include <iostream>

using namespace std;

class A{

public:

void show(){

cout<<"class A";

}

};

class B{

public:

void show(){

cout<<"class B";

}

};

class C:public A,public B{

};

int main()

{

C obj;

obj.show();// compiler error

obj.A::show();

obj.B::show();

return 0 ;

}

The members are ambiguous without scope resolution operator when the member function show () is accessed by the derived class object, naturally, the compiler can’t distinguish between member function of the class A and B.

### Order of Constructor in inheritance:

Base class constructors are always called in the derived class constructors. Whenever you create derived class object, first the base class default constructor is executed and then the derived class's constructor finishes execution.

using namespace std;

class A{

public:

int id;

string name;

A(int id){

this->id=id;

}

void disp(){

cout<<id;

}

};

class B: public A{

public:

B(int id):A(id){

}

};

int main()

{

B b=B(101);

b.disp();

return 0 ;

}