151down vote

A virtual constructor is not possible but virtual destructor is possible. Let us experiment....

#include <iostream>

using namespace std;

class Base

{

public:

Base(){

cout << "Base Constructor Called\n";

}

~Base(){

cout << "Base Destructor called\n";

}

};

class Derived1: public Base

{

public:

Derived1(){

cout << "Derived constructor called\n";

}

~Derived1(){

cout << "Derived destructor called\n";

}

};

int main()

{

Base \*b = new Derived1();

delete b;

}

The above code output the following:

Base Constructor Called

Derived constructor called

Base Destructor called

The construction of derived object follow the construction rule but when we delete the "b" pointer(base pointer) we have found that only the base destructor is call.But this must not be happened. To do the appropriate thing we have to make the base destructor virtual. Now let see what happen in the following:

#include <iostream>

using namespace std;

class Base

{

public:

Base(){

cout << "Base Constructor Called\n";

}

virtual ~Base(){

cout << "Base Destructor called\n";

}

};

class Derived1: public Base

{

public:

Derived1(){

cout << "Derived constructor called\n";

}

~Derived1(){

cout << "Derived destructor called\n";

}

};

int main()

{

Base \*b = new Derived1();

delete b;

}

The output changed as following:

Base Constructor Called

Derived constructor called

Derived destructor called

Base Destructor called

So the destruction of base pointer(which take an allocation on derived object!) follow the destruction rule i.e first the derived then the base. On the other hand for constructor there are nothing like virtual constructor.