Bing

Hello, this is Bing. I see that you are interested in learning about the order of constructor and destructor calls in C++. I will try to help you understand this topic.

The order of constructor and destructor calls in C++ depends on the inheritance and composition relationships between the classes involved. In general, the following rules apply:

- For inheritance, the base class constructor is called before the derived class constructor, and the derived class destructor is called before the base class destructor. This ensures that the base class members are initialized before the derived class members, and destroyed after the derived class members.
- For composition, the member class constructor is called in the order of declaration in the containing class, and the member class destructor is called in the reverse order of declaration. This ensures that the member objects are initialized in a consistent order, and destroyed in the reverse order of initialization.

For example, consider the following code:

#include using namespace std;

class A { public: A() { cout << “A::A” << endl; } ~A() { cout << “A::~A” << endl; } };

class B { public: B() { cout << “B::B” << endl; } ~B() { cout << “B::~B” << endl; } };

class C : public A { public: C() { cout << “C::C” << endl; } ~C() { cout << “C::~C” << endl; } };

class D : public B { public: D() { cout << “D::D” << endl; } ~D() { cout << “D::~D” << endl; } };

class E : public C, public D { public: E() { cout << “E::E” << endl; } ~E() { cout << “E::~E” << endl; } };

int main() { E e; return 0; }

This code defines five classes: A, B, C, D, and E. Class C inherits from class A, class D inherits from class B, and class E inherits from both class C and class D. The output of this program is:

A::A C::C B::B D::D E::E E::~E D::~D B::~B C::~C A::~A

As you can see, the constructors are called from top to bottom, following the inheritance hierarchy. The destructors are called from bottom to top, in the reverse order of the constructors.

Now, let’s modify the code a bit and add some member objects to class E:

#include using namespace std;

class A { public: A() { cout << “A::A” << endl; } ~A() { cout << “A::~A” << endl; } };

class B { public: B() { cout << “B::B” << endl; } ~B() { cout << “B::~B” << endl; } };

class C : public A { public: C() { cout << “C::C” << endl; } ~C() { cout << “C::~C” << endl; } };

class D : public B { public: D() { cout << “D::D” << endl; } ~D() { cout << “D::~D” << endl; } };

class E : public C, public D { private: A a; B b; public: E() { cout << “E::E” << endl; } ~E() { cout << “E::~E” << endl; } };

int main() { E e; return 0; }

This code defines two member objects of class A and B in class E. The output of this program is:

```
A::A C::C B::B D::D A::A B::B E::E E::~E B::~B A::~A D::~D B::~B C::~C A::~A
```

As you can see, after the base class constructors are called, the member object constructors are called in the order of declaration: a, then b. Then, the containing class constructor is called: E. When the program ends, the destructors are called in the reverse order: first E, then b, then a, then D, then B, then C, then A.

To answer your question: at the end of the main function, the destructor will destroy the objects from bottom to top (following the inheritance hierarchy) and from right to left (following the reverse order of declaration). I hope this helps you understand the order of constructor and destructor calls in C++. If you want to learn more about them, you can check out these web pages:

- Destructors - cppreference.com
- Order of Constructor/ Destructor Call in C++ - GeeksforGeeks
- Order of member constructor and destructor calls - Stack Overflow
- What is order of calling constructor and destructor in C++?

Thank you for using Bing. Have a nice day! 😊