河北师范大学软件学院
Software College of Hebei Normal University

C++

第十二讲 继承与派生（二）

C++备课组 丁盟

自我介绍

丁盟

QQ：2622885094

上一讲教学目标

- 理解C++中继承的概念
- 掌握C++中如何使用继承
- 理解C++中的三种继承方式

本讲教学目标

- 掌握C++中单重继承及多重继承中构造函数的调用
- 掌握C++中单重继承及多重继承中析构函数的调用

1 派生类构造与析构

- 单重继承
- 多重继承

单重继承的派生类构造与析构函数

构造函数,析构函数的定义和执行顺序

- 派生类里继承自基类的成员只能通过基类构造函数完成初始化，因此派生类构造函数的格式是特殊的，调用顺序也有规则。
- 我们首先看：单重继承时，派生类构造函数和析构函数执行顺序。

单重继承的派生类构造与析构函数

```
class A {
public:
    A() { cout << "A构造"
               << endl; }
    ~A() { cout << "A析构"
                << endl; }
};

class B : public A {
public:
    B() { cout << "B构造"
               << endl; }
    ~B() { cout << "B析构"
                << endl; }
};
```

```
class C : public B {
public:
    C() { cout << "C构造"
               << endl; }
    ~C() { cout << "C析构"
                << endl; }
};

int main(void)
{
    C *p = new C;

    return 0;
}
```

A构造
B构造
C构造

单重继承的派生类构造与析构函数

```
class Other {
public:
    Other() { cout
        << "Other构造\n"; }
    ~Other() { cout
        << "Other析构\n"; }
};

class Base {
public:
    Base() { cout
        << "Base构造\n"; }
    ~Base() { cout
        << "Base析构\n"; }
};
```

```
class Derive : public Base{
public:
    Derive() { cout
        << "Derived构造\n"; }
    ~Derive() { cout
        << "Derived析构\n"; }
private:
    Other m_Other;
};

int main(void) {
    Derive op;

    return 0;
}
```

```
Base构造
Other构造
Derived构造
Derived析构
Other析构
Base析构
```

单重继承的派生类构造与析构函数

- 派生类构造函数调用顺序如下：
  - 调用基类的构造函数
  - 派生类对象成员所属类的构造函数（有的话）
  - 最后调用派生类的构造函数
- 派生类析构函数调用顺序如下：
  - 调用派生类的析构函数
  - 派生类对象成员所属类的析构函数函数（有的话）
  - 调用基类的析构函数

递归

单重继承的派生类构造与析构函数

[思考]

- 如果将上例中子对象的定义放入基类对象中结果如何？

单重继承的派生类构造与析构函数

```
class Other {
public:
    Other() { cout
        << "Other构造\n"; }
    ~Other()  { cout
        << "Other析构\n"; }
};
class Base {
public:
    Base()  { cout
        << "Base构造\n"; }
    ~Base() { cout
        << "Base析构\n"; }
    Other m_Other;
};
```

```
class Derive : public Base {
public:
    Derive() { cout
        << "Derived构造\n"; }
    ~Derive() { cout
        << "Derived析构\n"; }
};

int main(void)
{
    Derive op;

    return 0;
}
```

```
Other构造
Base构造
Derived构造
Derived析构
Base析构
Other析构
```

单重继承的派生类构造与析构函数

❖ 单重继承时派生类构造函数的定义

```
<派生类名>(总形式参数表) : <直接基类名>(<参数表>),
    [子对象1(参数表1), ……], [派生类数据成员初始化]
{     [<派生类自身数据成员的初始化>]     }
```

❖ 说明：

- ➢在多层次继承中，每个派生类只需要负责向直接基类的构造函数提供参数。
- ➢如果通过派生类构造函数调用基类默认构造函数，则可以不给出显示调用形式，系统自动调用默认构造函数。

单重继承的派生类构造与析构函数

- 如果派生类或基类中有子对象，则子对象也必须使用初始化列表初始化（如果子对象没有默认构造函数）。
- 派生类中的常数据成员和引用成员的初始化必须放在初始化列表中。
- 派生类中一般数据成员可以在初始化列表或构造函数中初始化。

1 派生类构造与析构

- 单重继承
- 多重继承

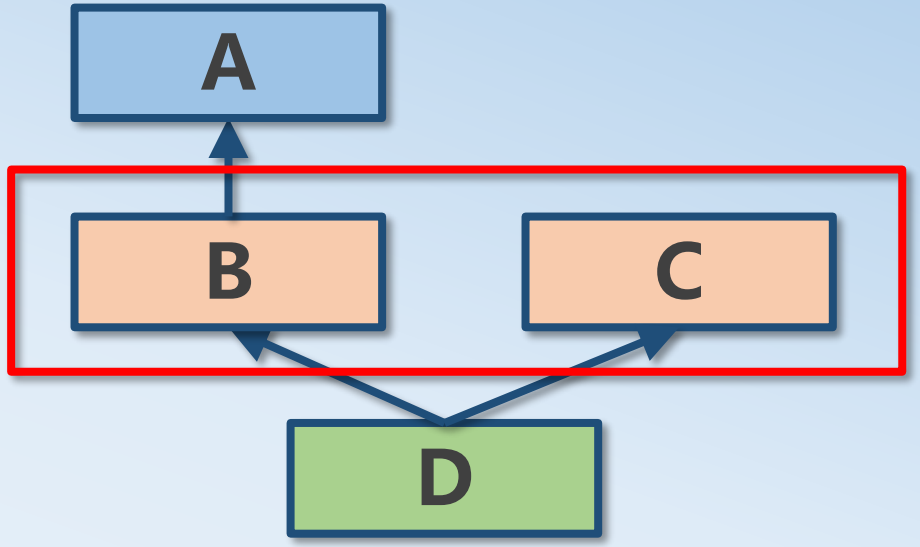
多重继承的派生类构造与析构函数

❖ 多重继承时派生类构造函数的定义

```
<派生类名>(总形式参数表)：<直接基类1>(<参数表1>),
                          <直接基类2>(<参数表2>),
                          <子对象1>(<参数表n>),
                          <子对象2>(<参数表n>),
                          ……
                          [,<其他初始化项>]
{
   [<派生类自身数据成员的初始化>]
}
```

多重继承的派生类构造与析构函数

- 说明
  - 只负责平行直接基类的初始化（如果直接基类没有默认构造函数）




```
D(int x=1, int y=2, int z=3, int m=4)
    : B(x,y), C(z)
{
    m_iD = m;
    cout << "Constructor D" << endl;
}
```

多重继承的派生类构造与析构函数

- 多重继承时，派生类构造函数执行顺序：
  - 按继承时直接基类声明顺序调用直接基类的构造函数
  - 如果派生类有子对象时，按子对象声明顺序调用派生类对象成员所属类的构造函数
  - 最后调用派生类构造函数
- **析构函数执行顺序和构造函数相反。**

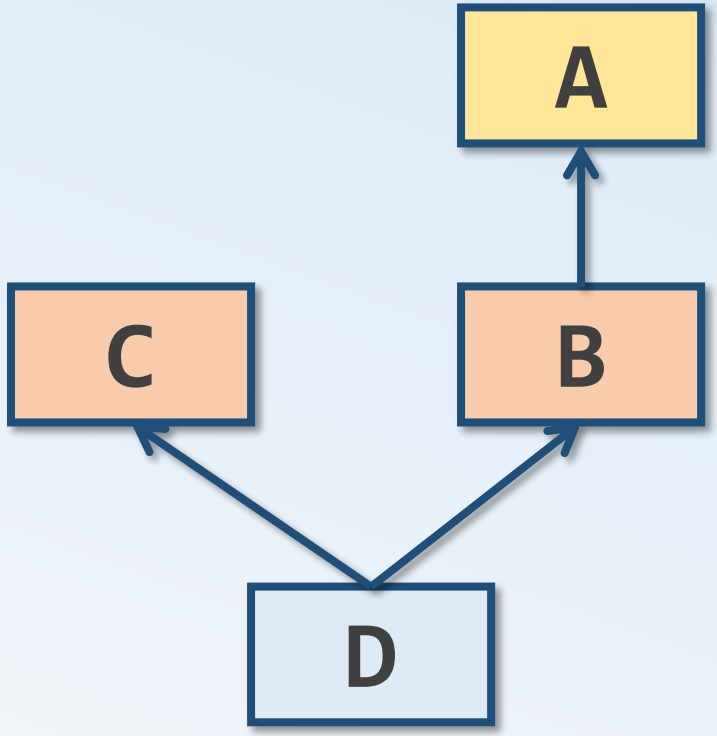
多重继承的派生类构造与析构函数

- 分析D对象实例化构造函数和析构函数的执行顺序

```
class A {
public:
    A() {
        cout << "A构造\n"; }
    ~A() {
        cout << "A析构\n"; }
};
class B : public A {
public:
    B() {
        cout << "B构造\n"; }
    ~B() {
        cout << "B析构\n"; }

};
```

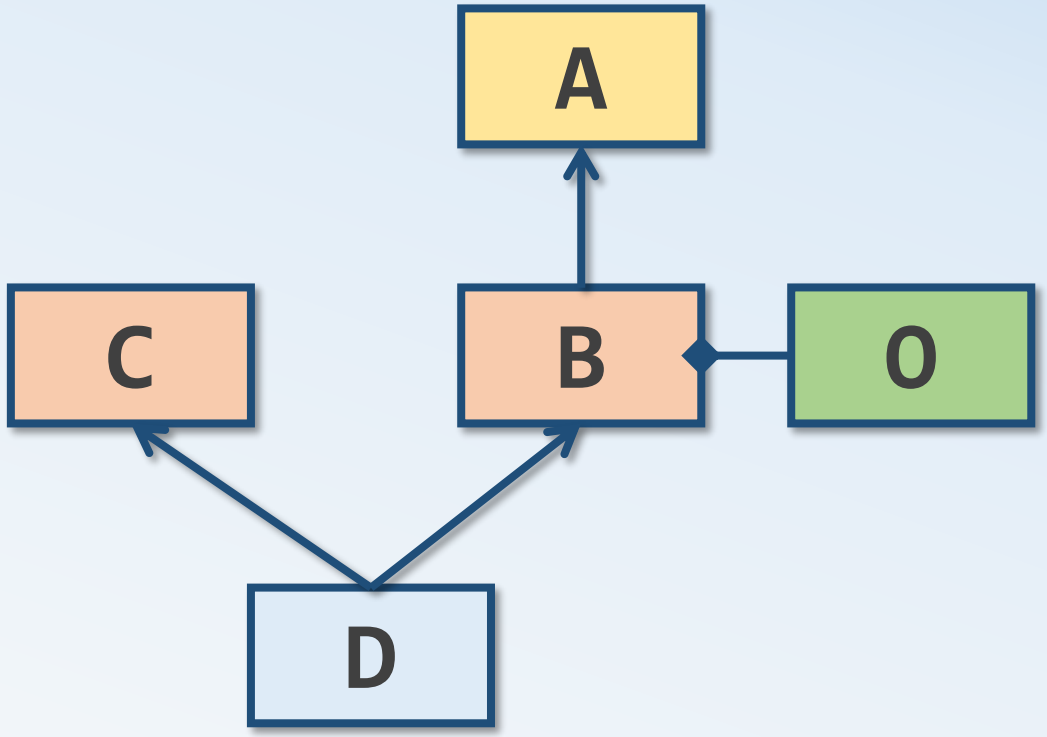
```
class C {
public:
    C()  {
        cout << "C构造\n"; }
    ~C() {
        cout << "C析构\n"; }
};

class D : public C, public B {
public:
    D(){
        cout << "D构造\n"; }
    ~D() {
        cout << "D析构\n"; }
};
```

多重继承的派生类构造与析构函数




```
Constructor C
Constructor A
Constructor B
Constructor D
Destructor D
Destructor B
Destructor A
Destructor C
```

多重继承的派生类构造与析构函数

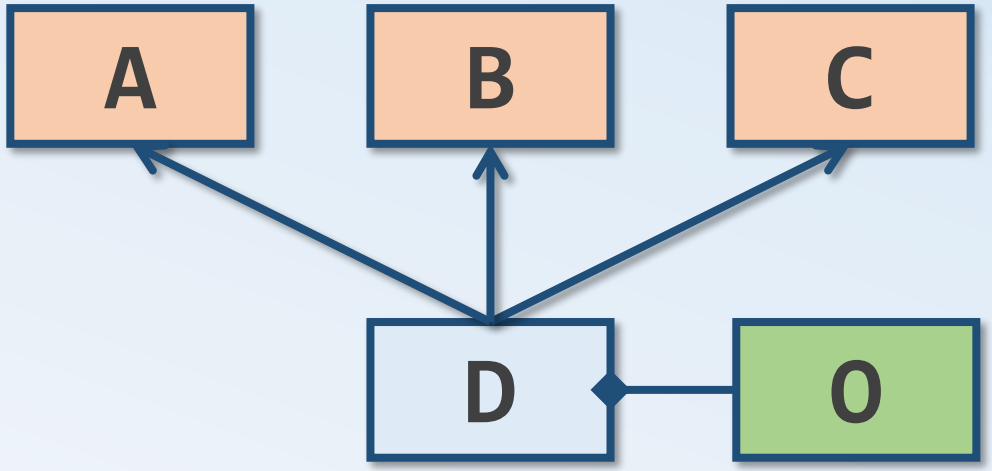



```
Constructor C
Constructor A
Constructor O
Constructor B
Constructor D
Destructor D
Destructor B
Destructor O
Destructor A
Destructor C
```

多重继承的派生类构造与析构函数

- 实例分析

  分析下面实例化D的对象时类的构造和析构的顺序



多重继承的派生类构造与析构函数

- 总结：
  - 多重继承下派生类构造函数必须同时负责其所有直接基类构造函数的调用
  - 不能负责调用间接基类构造函数
  - 如果调用基类默认构造函数，则可以省略
  - 无论单重继承还是多重继承他们构造函数的调用都体现递归的过程

本讲教学目标

- 掌握C++中单重继承及多重继承中构造函数的调用
- 掌握C++中单重继承及多重继承中析构函数的调用


THANKS