C++ 继承与派生

什么是继承与派生

继承：子类拥有父类的成员
派生：用已有的类派生出新类

Base（父类） → Derived（子类）

父类写 通用特性
子类写 特殊特性

继承的作用--不用重复写代码

```
class Student { int id; string name; };
class Teacher { int id; string name; };
```

↓

```
class Person { int id; string name; };
class Student : public Person {};
class Teacher : public Person {};
```

最基本的继承语法

```
class Base {
public:
    int a;
};

class Derived : public Base {
public:
    int b;
};
```

使用方法 ↘

```
Derived d;
d.a = 10;    // 来自 Base
d.b = 20;    // 自己的
```

访问权限!

成员访问权限

关键字	类内	派生类	类外
public	✔	✔	✔
protected	✔	✔	✘
private	✔	✘	✘

继承方式

```
class Derived : public Base {};
class Derived : protected Base {};
class Derived : private Base {};
```

访问变化规则

父类成员	public继承	protected继承	private继承
public	public	protected	private
protected	protected	protected	private
private	不可访问	不可访问	不可访问

构造与析构顺序

先构造父类 → 再构造子类

先析构子类 → 再析构父类

```cpp
#include <iostream>
using namespace std;

class Base {
public:
    Base() {
        cout << "Base constructor" << endl;
    }
    ~Base() {
        cout << "Base destructor" << endl;
    }
};

class Derived : public Base {
public:
    Derived() {
        cout << "Derived constructor" << endl;
    }
    ~Derived() {
        cout << "Derived destructor" << endl;
    }
};

int main() {
    Derived d;
    return 0;
}
```

```
Base constructor
Derived constructor
Derived destructor
Base destructor
```

父类构造函数的调用--初始化列表

父类没有默认构造函数 → 子类必须显式调用

```
class Base {
public:
    int x;
    Base(int x) : x(x) {}
};

class Derived : public Base {
public:
    Derived(int x) : Base(x) {}
};
```

同名成员

子类定义同名成员，会隐藏父类成员

```
class Base {
public:
    int value = 10;
};

class Derived : public Base {
public:
    int value = 20;
};

int main() {
    Derived d;
    cout << d.value << endl;        // 20
    cout << d.Base::value << endl;  // 10
}
```

继承中的 this 指针

this 指向 当前对象

```
this->member;          // 优先找子类
Base::member;          // 明确访问父类
```

虚函数+重写+继承与派生

```
#include<iostream>
using namespace std;
class A{
    int x;
public:
    A(int x=0){this->x=x;}
    virtual void f(){cout<<x<<endl;}
};
class B:public A{
    int y;
public:
    B(int x, int y=1):A(x){this->y=y;}
    void f(){cout<<y<<endl;}
};
int main(void) {
    A a1(10), *pa;
    B b1(20, 30);
    a1.f();
    pa=&a1;
    pa->f();
    pa=&b1;
    pa->f();
    pa->f();
    return 0;
}
// 写出程序运行结果：
```

- 10
- 10
- 30
- 30

1.基类中必须将目标函数声明为虚函数（A 类的f()加了virtual）；
2.子类必须公有继承基类（B 类public A），并重写（覆盖）虚函数（B 类重写了f()）；
3.必须通过基类的指针或引用调用虚函数（用 A 类指针pa调用f()）。

子类构造函数必须通过初始化列表调用基类构造函数，这是继承中初始化基类成员的唯一方式

虚基类构造函数执行的核心规则

1. 虚基类的构造函数优先于非虚基类执行；

2. 多个虚基类的执行顺序：按继承声明的 “从左到右” 顺序，且同一虚基类只执行一次构造函数；

3. 非虚基类的执行顺序：按派生类继承列表的从左到右；

4. 派生类自身构造函数最后执行。

```cpp
#include<iostream>
using namespace std;

// 基类1
class base1{
public:
    base1() {
        cout << "class base1" << endl;
    }
};

// 基类2
class base2{
public:
    base2() {
        cout << "class base2" << endl;
    }
};

// 派生类level1：公有继承base2，虚继承base1
class level1:public base2, virtual public base1{
public:
    level1() {
        cout << "class level1" << endl;
    }
};

// 派生类level2：公有继承base2，虚继承base1
class level2:public base2, virtual public base1{
public:
    level2() {
        cout << "class level2" << endl;
    }
};

// 最顶层派生类toplevel：公有继承level1，虚继承level2
class toplevel:public level1, virtual public level2{
public:
    toplevel() {
        cout << "class toplevel" << endl;
    }
};

int main() {
    toplevel obj; // 创建toplevel对象，触发所有基类+自身构造函数
    return 0;
}
```

- class base1
- class base2
- class level2
- class base2
- class level1
- class toplevel