第13章 继承构造函数

《现代C++语言核心特性解析》 谢丙堃

继承关系中构造函数的困局

- 不得不重复的构造函数

```cpp
class Base {
public:
    Base() : x_(0), y_(0.) {};
    Base(int x, double y) : x_(x), y_(y) {}
    Base(int x) : x_(x), y_(0.) {}
    Base(double y) : x_(0), y_(y) {}
    void SomeFunc() {}
private:
    int x_;
    double y_;
};
```

```cpp
class Derived : public Base {
public:
    Derived() {};
    Derived(int x, double y) : Base(x, y) {}
    Derived(int x) : Base(x) {}
    Derived(double y) : Base(y) {}
    void SomeFunc() {}
};
```

使用继承构造函数

- 基本语法

```
class Base {
public:
    Base() : x_(0), y_(0.) {};
    Base(int x, double y) : x_(x), y_(y) {}
    Base(int x) : x_(x), y_(0.) {}
    Base(double y) : x_(0), y_(y) {}
private:
    int x_;
    double y_;
};

class Derived : public Base {
public:
    using Base::Base;
};
```

继承构造函数的规则

1. 派生类是隐式继承基类的构造函数，所以只有在程序中使用了这些构造函数，编译器才会为派生类生成继承构造函数的代码
2. 派生类不会继承基类的默认构造函数和拷贝构造函数
3. 继承构造函数不会影响派生类默认构造函数的隐式声明

继承构造函数的规则

4. 在派生类中声明签名相同的构造函数会禁止继承相应的构造函数

```
class Base {
public:
    Base() : x_(0), y_(0.) {};
    Base(int x, double y) : x_(x), y_(y) {}
    Base(int x) : x_(x), y_(0.) { std::cout << "Base(int x)" << std::endl; }
    Base(double y) : x_(0), y_(y) { std::cout << "Base(double y)" << std::endl; }
private:
    int x_;
    double y_;
};

class Derived : public Base {
public:
    using Base::Base;
    Derived(int x) { std::cout << "Derived(int x)" << std::endl; }
};
```

继承构造函数的规则

5. 派生类继承多个签名相同的构造函数会导致编译失败

```
class Base1 {
public:
    Base1(int) { std::cout << "Base1(int x)" << std::endl; };
};
class Base2 {
public:
    Base2(int) { std::cout << "Base2(int x)" << std::endl; };
};

class Derived : public Base1, Base2 {
public:
    using Base1::Base1;
    using Base2::Base2;
};
```

继承构造函数的规则

6. 继承构造函数的基类构造函数不能为私有

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

int main() {
    Derived d(5.5);
    Derived d1(5);
}
```

感谢聆听
欢迎关注