C++ 多态性：C++11：override 与 final（学习笔记：第8章 09）

品颜完月
华东交通大学 电气工程硕士
已关注

C++11：override 与 final[1]

override

- 多态行为的基础：基类声明虚函数，继承类声明一个函数覆盖该虚函数
- 覆盖要求：函数签名（signature）完全一致
- 函数签名包括：*函数名 参数列表 const*

下列程序就仅仅因为疏忽漏写了const，导致多态行为没有如期进行：

源代码：

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

class Base
{
public:
        virtual void f1(int)const;
        virtual ~Base() {
        };
};
void Base::f1(int) const
{
        cout << "Base f1" << endl;
        return;
}
class Derived:public Base
{
public:
        void f1(int);
        ~Derived() {
        };
};
void Derived::f1(int)
{
        cout << "derived f1" << endl;
}
int main() {
        Base *b;
        b = new Base;
        b->f1(1);
        b = new Derived;
        b->f1(1);
        return 0;
}
```

程序运行结果：


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

class Base {
public:
  virtual void f1(int) const;
  virtual ~Base() {
};
};

void Base::f1(int) const {
  cout << "Base f1" << endl;
  return;
}
```

```
class Derived: public Base {
public:
  void f1(int);
  ~Derived() {
};
};

void Derived::f1(int) {
  cout << "derived f1" << endl;
}
```

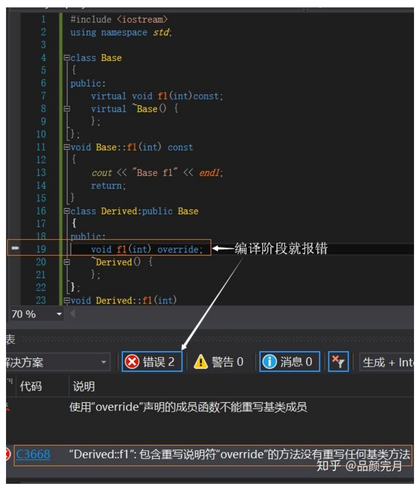
```
int main() {
  Base *b;
  b = new Base;
  b->f1(1);
  b = new Derived;
  b->f1(1);
  return 0;
}
```

运行结果
Base f1
Base f1

显式函数覆盖

- C++11 引入显式函数覆盖，在编译期而非运行期捕获此类错误。
- 在虚函数显式重载中运用，编译器会检查基类是否存在一虚拟函数，与派生类中带有声明override的虚拟函数，有相同的函数签名（signature）；若不存在，则会回报错误。

修改上例，发现添加override，会在编译阶段报错，方便查找错误地方。



在编译阶段发现错误后，对程序进行修改：

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

class Base
{
public:
        virtual void f1(int)const;
        virtual ~Base() {
        };
};
void Base::f1(int) const
{
        cout << "Base f1" << endl;
        return;
}
class Derived:public Base
{
public:
        void f1(int) const override;//省略了virtual
        ~Derived() {
        };
};
void Derived::f1(int) const
{
        cout << "derived f1" << endl;
}
int main() {
        Base *b;
        b = new Base;
        b->f1(1);
        b = new Derived;
        b->f1(1);
        return 0;
}
```

程序运行结果：

final

- C++11提供的final，用来避免类被继承，或是基类的函数被改写
- 例：

```
struct Base1 final { };
struct Derived1 : Base1 { }; // 编译错误：Base1为final，不允许被继承
struct Base2 { virtual void f() final; };
struct Derived2 : Base2 { void f(); // 编译错误：Base2::f 为final，不允许被覆盖 };
```

> *注意：在Visual Stuidio和C++11标准中，可以将override与final用作变量名和函数名，它们不是语言本身的关键字，但不建议这样使用。*

参考

1. ^ http://www.xuetangx.com/courses/course-v1:TsinghuaX+00740043_2x_2015_T2+sp/courseware/93f0d3a029d84059a84d02745a1e2bfd/5bf2454cc05d46f0b3314c65c6db5f89/

发布于 2020-02-02 03:06

C++ C++11 C++ 入门

文章被以下专栏收录

C++学习笔记
学习C++过程中的笔记，遇到的相关问题及解决办法

还没有评论

写下你的评论...