

陆初

脑子不太好用的普通人。

友元黑魔法——在类内定义友元函数

起因是我写了这么一段代码：

```
template <typename type>
class LIFO_stack : public my_stack<type> {
    template <typename ano_type>
    friend ostream& operator<<(ostream& os,
LIFO_stack<ano_type>& s);
}
template <typename type>
ostream& operator<<(ostream& os, LIFO_stack<type>& s) {
    auto it = --s.vec.end();
    for (; it != s.vec.begin(); --it) {
        os << *it << ' ';
    }
    os << *it << endl;
    return os;
}
```

然后跑去问jp同学是不是这种友元函数一定要在template类里面再声明一个template才行，别的方法可不可以。

```

using namespace std;

template <typename type>
class my_stack {
public:
    void push(type v);

    void pop();

    friend ostream& operator<< (ostream &os, my_stack<type> &s) {
        return os;
    }
};

int main() {
    auto tmp = my_stack<int>();
    cout << tmp;
    return 0;
}

```

然后jp同学扔了这么一段代码给我，说：可以。

我当时就蒙了：？？？？等等这都写在类里面了！！「<<」操作符不是不能作为成员函数吗！

jp同学说：你看仔细点，这是友元，不算成员函数。



还有这种操作？？这是什么友元黑魔法？

然后我又试着自己写了一段代码尝试一下：

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  class B {
7  public:
8      B() : b(1) {}
9      int b;
10 };
11
12 class A {
13     friend B& operator+=(B& b1, B& b2) {
14         b1.b += b2.b;
15         return b1;
16     }
17 };
18
19 int main() {
20     B b1, b2;
21     b1 += b2;
22     cout << b1.b;
23 }
```

%

错误列表

整个解决方案

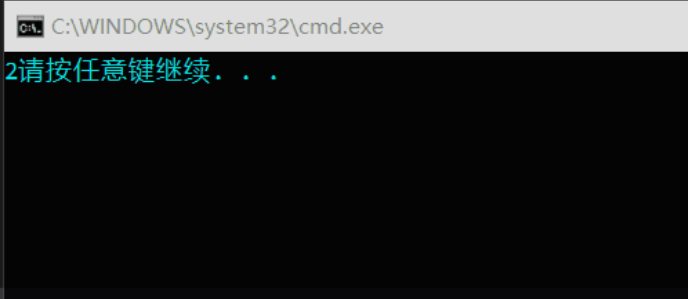
错误 2 警告 0 消息 0 生成 + IntelliSense

代码	说明
E0349	没有与这些操作数匹配的 "+=" 运算符

报错了【。

这不科学啊！明明这个「+=」操作符也是友元函数啊！凭什么这个就不能通过编译！

```
Learning (全局范围)
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  class B {
7  public:
8      friend B& operator+=(B& b1, B& b2) {
9          b1.b += b2.b;
10         return b1;
11     }
12     friend ostream& operator<<(ostream& os, const B& b) {
13         os << b.b;
14         return os;
15     }
16     B() : b(1) {}
17     int b;
18 };
19
20 int main() {
21     B b1, b2;
22     b1 += b2;
23     cout << b1;
24 }
```



然后再经过jp同学的提示，把这个函数放在所操作的类里面作为友元并定义，结果居然！通！过！了！



凭什么哦，同样是友元为什么待遇不一样

后来经过jp同学和我（其实主要还是jp同学 唔唔唔jp大佬太强了 扑通）的查找，终于在这么个答案里面找到了原因：

[来自微软的官方文档](#)

内联友元定义

可以在类声明中定义友元函数。 这些函数是内联函数，类似于成员内联函数，其行为就像它们在所有类成员显示后但在类范围关闭前（类声明的结尾）被定义时的行为一样。

类声明中定义的友元函数不被认为在封闭类的范围内；它们在文件范围内。

[难得遇到的有质量的百度知道回答](#)

hello()的确被定义为全局函数，但是外面调用hello()的时候找不到函数定义，实际上根本就不知道这个函数被定义过，因为在作用域内没有声明。在外面加一行设声明即可。

参见C++标准7.3.1.2节第三段：

The name of the friend is not found by unqualified lookup (3.4.1) or by qualified lookup (3.4.3) until a matching declaration is provided in that namespace scope (either before or after the class definition granting friendship). If a friend function is called, its name may be found by the name lookup that considers functions from namespaces and classes associated with the types of the function arguments (3.4.2).

C++标准3.4.2节 (Argument-dependent name lookup) 第四段：

Any namespace-scope friend functions or friend function templates declared in associated classes are visible within their respective namespaces even if they are not visible during an ordinary lookup (11.3).

也就是说，除非友元函数带了class A的参数，可以通过Argument-dependent name lookup (ADL) 被找到，否则必须在作用域里有声明才能在作用域中可见。



这么复杂的吗

原因总算是找到了，在类内被定义的友元函数实际上的作用域是整个文件，所以实际上不仅是这个类，这个文件内都可以访问这个函数。

那为什么写在所操作的类内就可以调用，除此之外就不能调用呢？

其实是因为编译器找不到入口找 (look up) 这个函数。

如果写在所操作的类内，编译器就可以通过unqualified lookup中的Argument-dependent name lookup (ADL) 找到这个函数，除此之外就找不到，所以就无法调用。

那我们无论如何都想调用怎么办？

实际上，很简单，这属于有定义（definition）没声明（declaration）的情况，我们只需要在文件域声明一次这个函数就行了。

也就是说，我们掌握了一个友元黑魔法，以后想要定义操作自身类成员的友元函数的时候，直接定义在类内就行了，不需要在类外定义。特别是「<<」操作符采用这种方法就特别方便。

例如我一开始写的代码可以改成这样：

```
template <typename type>
class LIFO_stack : public my_stack<type> {
    friend ostream& operator<<(ostream& os, const LIFO_stack&
s) {
        auto it = --s.vec.end();
        for (; it != s.vec.begin(); --it) {
            os << *it << ' ';
        }
        os << *it << endl;
        return os;
    }
}
```

可以达到同样的效果。

如果要更深入的话，就要讨论到qualified name lookup和unqualified name lookup了。

详细的定义可以看这里：

https://en.cppreference.com/w/cpp/language/qualified_lookup

https://en.cppreference.com/w/cpp/language/unqualified_lookup

这一段我也还没有研究透，还是等研究透了再更新吧。



作者：陆初

脑子不太好用的普通人。顺带一提性格也有点古怪。在老妈子和厌世肥宅中来回切换。

[查看陆初的所有文章](#)



陆初 / 2018-07-20 / C++ /

《友元黑魔法——在类内定义友元函数》有一个想法



tono

2018-07-20 21:29

是神仙斗法呜呜呜

此站点使用Akismet来减少垃圾评论。 [了解我们如何处理您的评论数据。](#)

陆初 / 自豪地采用WordPress