# 关于函数类对象

在C++中有四种可调用对象，分别是函数、函数指针、lambda表达式、重载了函数调用运算符的类（函数对象）。前两者较为常见，lambda表达式可参考《C++ Primer》第五版389页至353页。这里举例说明函数对象的用法。

以下代码用来说明函数类与普通类的区别，程序中定义了两个类，一个PrintString类和一个ClassA类，两者的区别主要在于，PrintString类中重载了调用运算符，即括号，而ClassA中没有。如果类定义了调用运算符，则该类的对象称作函数对象，因为可以调用这种对象，所以这些对象的行为像函数一样，这可以在下面的代码中得到验证。

#include "stdafx.h"

#include <vector>

#include <algorithm>

#include <iostream>

#include <string>

#include <stdlib.h>

using namespace std;

class PrintString{

public:

PrintString(char c = 's'):

sep(c){

cout << "构造函数执行..." << endl;

}

const PrintString &operator=(PrintString printor)

{

cout << "执行重载的=运算符函数" << endl;

if (this == &printor)

{

return \*this;

}

else

{

sep = printor.sep;

}

}

void operator()(const string &s)

{

cout << s << sep;

}

public:

char sep;

};

void PrintFunc(string &str)

{

cout << str <<"输出函数";

}

class ClassA{

ClassA()

{

cout << "类A的构造函数" << endl;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

string str("Hello World");

vector<string> strVector;

strVector.push\_back("Hello");

strVector.push\_back("China!");

PrintString psa(' ');

psa("Hello Russia!");

//ClassA(); //这里会出错，普通类不可以以这种方式定义一个临时的类对象

PrintString ps('a');

cout <<ps.sep<< endl;

ps=PrintString(); //正确，这里会生成一个临时的类对象

cout << ps.sep << endl;

for\_each(strVector.begin(), strVector.end(), PrintFunc);

for\_each(strVector.begin(), strVector.end(), PrintString());

cout << endl;

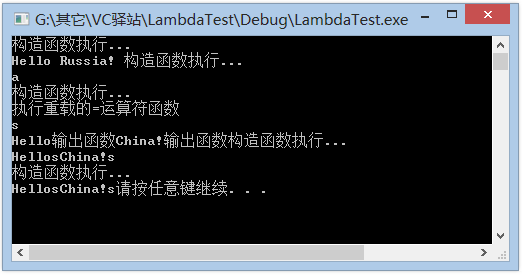
PrintString printor;

for\_each(strVector.begin(), strVector.end(), printor);

system("pause");

return 0;

}结果如下：



上述代码说明了几个问题：

1、函数对象可以直接调用，其行为像函数一样，同时由于它还有内部成员，因此可以保存状态。

PrintString psa(' ');

psa("Hello Russia!");

定义了一个psa对象，并将内部成员sep赋值为空格，然而后直接将字符串传递给psa，程序正确输出了字符串，即执行了重载的调用运算符。

2、函数类+()，生成一个临时函数类对象，但普通类不可以。

PrintString ps('a');

cout <<ps.sep<< endl;

ps=PrintString(); //正确，这里会生成一个临时的类对象

cout << ps.sep << endl;

以上代码首先定义了一个函数对象ps，并将字符a传递给内部sep，打印结果说明赋值正确，执行ps=PrintString()后，再次打印ps.sep，其值改变，说明PrintString()隐性的生成了一个临时函数对象。但对于普通类ClassA，如果直接调用ClassA()则会出错。

3、由于上述事实，以下代码都是正确的：

for\_each(strVector.begin(), strVector.end(), PrintFunc);

for\_each(strVector.begin(), strVector.end(), PrintString());

PrintString printor;

for\_each(strVector.begin(), strVector.end(), printor);

MSDN中对for\_each算法的原型定义如下：

template<class InputIterator, class Function>

Function for\_each(

InputIterator \_First,

InputIterator \_Last,

Function \_Func

);

其中第三个参数是一个函数对象（function object），示例该代码中传入的参数是PrintFunc、PrintString()、printor都是正确的。