# 模板函数在类中的实现

## 一、现象

普通的模板函数是可以定义在类中的，例如：

#include "stdafx.h"

#include <iostream>

#include <stdio.h>

using namespace std;

class Printor

{

public:

template<class T>

void print(T val)

{

cout << val << endl;

}

};

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

{

Printor pt;

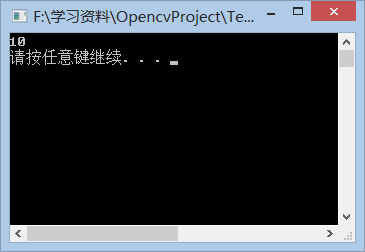
double t = 10.0;

pt.print(t);

system("pause");

return 0;

}



但是，如果把Printor类的定义分别写到两个文件中，即头文件和源文件如下：

Printor.h文件：

#pragma once

#include <iostream>

using namespace std;

class Printor

{

public:

public:

template<class T>

void print(T val);

};

Printor.cpp文件：

#include "stdafx.h"

#include "Printor.h"

template<class T>

void print(T val)

{

cout << val << endl;

}

Main文件：

#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include "Printor.h"

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

{

Printor pt;

double t = 10.0;

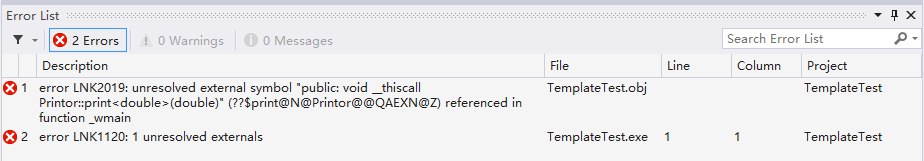
pt.print(t);

system("pause");

return 0;

}

类的定义没有任何更改，但是在编译的时候会出现下面的错误：



## 二、原因

模板的具体定义要在模板参数确定了之后才能实例化。而在模板实例化的过程中（比如在main函数中，但只包含模板声明函数）。包含实例化的.cpp文件编译成.o文件，留了类函数的入口地址等待填充。

一般，包含普通函数的.cpp文件编译成.o文件时。函数是确定的，能编译成二进制代码。然后就有函数入口地址可以让链接程序填入调用了这个函数的.o文件中，然后函数调用就成功了。

但是，对于模板类的实现。编译成.o文件时，仍然没有实例化。就是说模板类的实现函数不知道具体的模板函数是什么，不能实例化成一个真正的类型（比如Vector<int> a ; //参数实例化成int，vector<int>为一个类型( 和内置类型double一样了 ))。但是没有实例化前，函数是不确定的。就是还没有编译成二进制文件。所以没有函数的入口地址提供。由于没有入口地址，链接程序在帮main函数找实例化成具体类型的模板找实现函数时找不到，就提示链接错误了。

## 三、解决办法

将模板函数的实现放在头文件中，而不是源文件中，对于开始那段程序，可以这样改写：

Printor.h:

#pragma once

#include <iostream>

using namespace std;

class Printor

{

public:

public:

template<class T>

void print(T val);

};

template<class T>

void Printor::print(T val)

{

cout << val << endl;

}

Printor.cpp：

#include "stdafx.h"

#include "Printor.h"

//其它一般类成员函数

Main文件：

#include <stdio.h>

#include <stdlib.h>

#include "Printor.h"

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

{

Printor pt;

double t = 10.0;

pt.print(t);

system("pause");

return 0;

}

