非内联成员函数只会诞生一份函数实例，也就是说多个类对象公用一份函数代码，那么这块函数代码如何区分到底是哪个类对象调用的自己呢？

this指针，指向被调用的成员函数所属的对象。啥意思呢?

Person是一个类，func()是非静态成员函数

Person p1;

P1.func();

func()是被调用的成员函数，是由p1这个类对象调用的，因此，this指针便指向了Person p1这个类对象。即 \*this = p1;

当调用p1调用func（）时，其实编译器在func()中隐含地添加了参数func(Person \*this)， 意思就是将p1这个类对象当参数传了进去，因此，成员函数func()就能够区分出来是p1调用了它。

this指针是隐含在类对象内的一种指针， 当类对象被创建后，它的非静态成员函数都隐含有指针this，指向的是这个类对象的地址。This指针是隐含的，不是类对象的一部分，不占类对象的空间。

那为什么静态成员函数没有this指针呢？这还得从静态成员函数的性质说起，静态成员函数是整个类共享的，并不属于某一个类对象，因此不需要区分类对象。而this指针的作用是为了区分每个类对象，提供类对象的地址，从这里可以看出，静态成员函数是不需要this指针的。

那么，什么时候用this指针呢？

1. 当传入的参数和类对象的参数命名冲突时，我们就可以用this->来区分两种同名参数。
2. 当要返回类对象或类的参数时，用this->来返回。

空指针访问成员函数

注意：若用空指针访问成员函数时，参数中隐含着this指针，此时的this指针为空指针NULL，，当该成员函数中没有用到this指针时，是可以正常地输出的，如下面代码中的”text

函数，若成员函数由用到this指针，则会报错，因为是空指针，如下面代码中的”num”成员函数。

代码如下：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

using namespace std;

class Person

{

public:

void text()

{

cout << “text” << endl;  
}

void num()

{

cout << num << endl;  
}

int num;  
};

void test()

{

Person \*p = NULL;  
}

Int main()

{  
test();

Return 0;

}