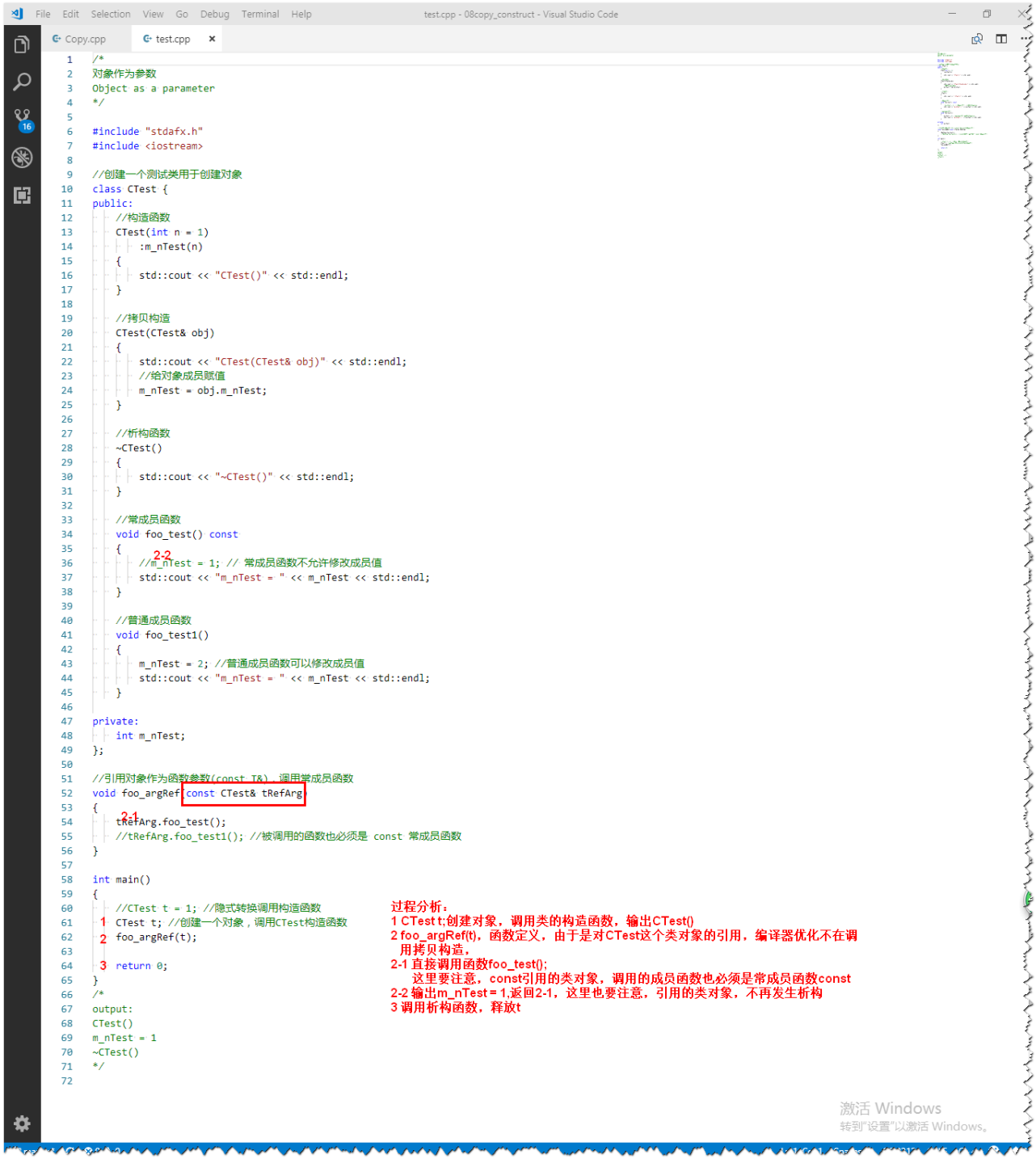
```cpp
/*
对象作为参数
Object as a parameter
*/

#include "stdafx.h"
#include <iostream>

//创建一个测试类用于创建对象
class CTest {
public:
    //构造函数
    CTest(int n = 1)
        :m_nTest(n)
    {
        std::cout << "CTest()" << std::endl;
    }

    //拷贝构造
    CTest(CTest& obj)
    {
        std::cout << "CTest(CTest& obj)" << std::endl;
        //给对象成员赋值
        m_nTest = obj.m_nTest;
    }

    //析构函数
    ~CTest()
    {
        std::cout << "~CTest()" << std::endl;
    }

    //常成员函数
    void foo_test() const
    {
        //m_nTest = 1; // 常成员函数不允许修改成员值
        std::cout << "m_nTest = " << m_nTest << std::endl;
    }

    //普通成员函数
```

```
    void foo_test1()
    {
        m_nTest = 2; //普通成员函数可以修改成员值
        std::cout << "m_nTest = " << m_nTest << std::endl;
    }

private:
    int m_nTest;
};

//引用对象作为函数参数(const T&)，调用常成员函数
void foo_argRef(const CTest& tRefArg)
{
    tRefArg.foo_test();
    //tRefArg.foo_test1(); //被调用的函数也必须是 const 常成员函数
}

int main()
{
    //CTest t = 1; //隐式转换调用构造函数
    CTest t; //创建一个对象，调用CTest构造函数
    foo_argRef(t);

    return 0;
}
/*
output:
CTest()
m_nTest = 1
~CTest()
*/

```


File Edit Selection View Go Debug Terminal Help
test.cpp - 08copy_construct - Visual Studio Code
Copy.cpp
test.cpp
/*
对象作为参数
Object as a parameter
*/
#include "stdafx.h"
#include <iostream>
//创建一个测试类用于创建对象
class CTest {
public:
//构造函数
CTest(int n = 1)
:m_nTest(n)
{
std::cout << "CTest()" << std::endl;
}
//拷贝构造
CTest(CTest& obj)
{
std::cout << "CTest(CTest& obj)" << std::endl;
//给对象成员赋值
m_nTest = obj.m_nTest;
}
//析构函数
~CTest()
{
std::cout << "~CTest()" << std::endl;
}
//常成员函数
void foo_test() const
{
2-2
//m_nTest = 1; // 常成员函数不允许修改成员值
std::cout << "m_nTest = " << m_nTest << std::endl;
}
//普通成员函数
void foo_test1()
{
m_nTest = 2; //普通成员函数可以修改成员值
std::cout << "m_nTest = " << m_nTest << std::endl;
}
private:
int m_nTest;
};
//引用对象作为函数参数(const T&)，调用常成员函数
void foo_argRef(const CTest& tRefArg)
{
2-1
tRefArg.foo_test();
//tRefArg.foo_test1(); //被调用的函数也必须是 const 常成员函数
}
int main()
{
//CTest t = 1; //隐式转换调用构造函数
1 CTest t; //创建一个对象，调用CTest构造函数
2 foo_argRef(t);
3 return 0;
}
/*
output:
CTest()
m_nTest = 1
~CTest()
*/
过程分析：
1 CTest t;创建对象，调用类的构造函数，输出CTest()
2 foo_argRef(t)，函数定义，由于是对CTest这个类对象的引用，编译器优化不在调用拷贝构造，
2-1 直接调用函数foo_test();
这里要注意，const引用的类对象，调用的成员函数也必须是常成员函数const
2-2 输出m_nTest = 1,返回2-1，这里也要注意，引用的类对象，不再发生析构
3 调用析构函数，释放t
激活 Windows
转到"设置"以激活 Windows。