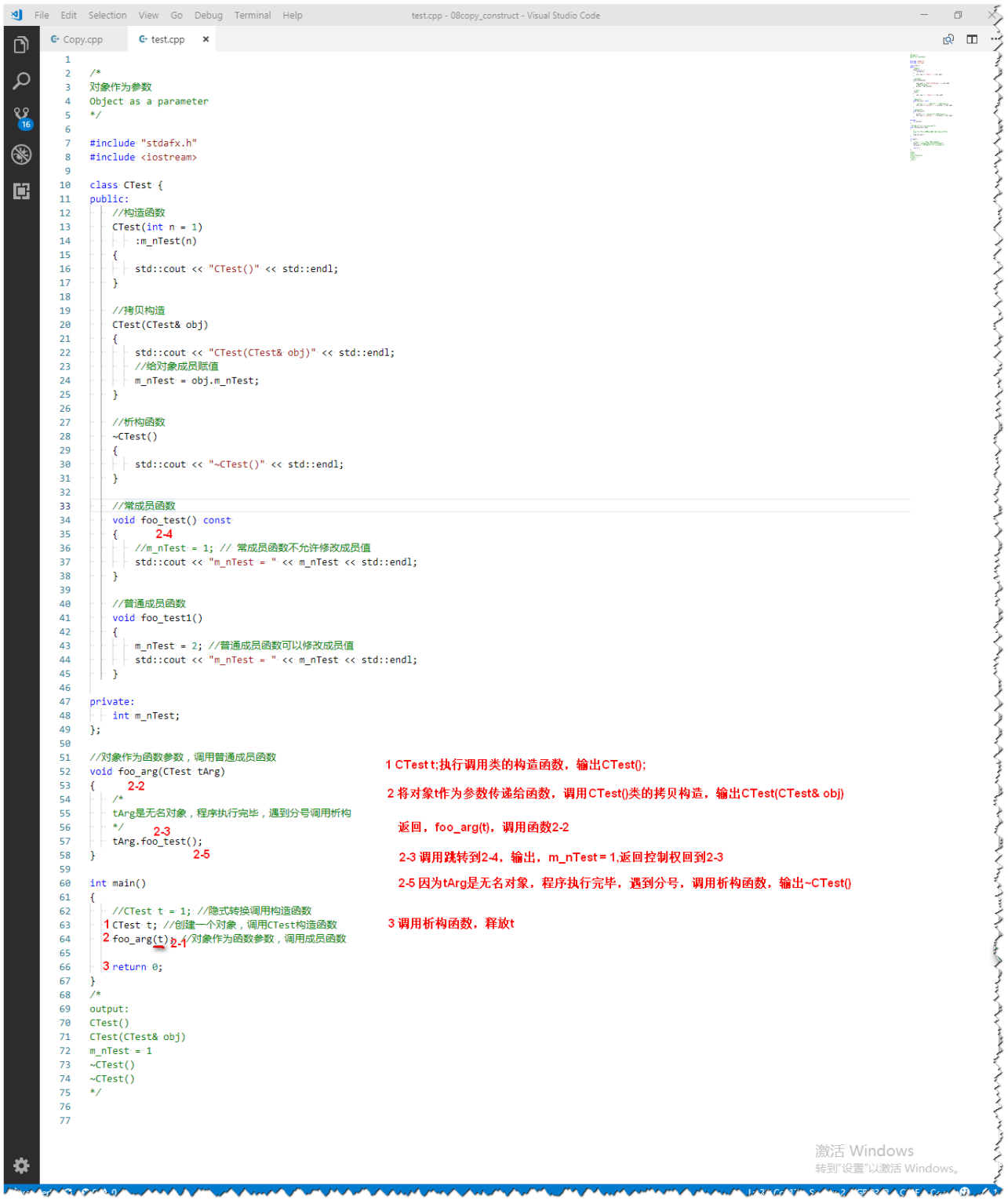
对象作为参数

1. 形参会拷贝构造产生: `void foo_arg(CTest tArg)`
2. 通常参数的传递会使用引用(const 引用): `void foo_argRef(const CTest& tRefArg)`

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

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

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;
};

//对象作为函数参数，调用普通成员函数
void foo_arg(CTest tArg)
{
    /*
    tArg是无名对象，程序执行完毕，遇到分号调用析构
    */
    tArg.foo_test();
}

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_arg(t); //对象作为函数参数，调用成员函数

    return 0;
}
/*
output:
CTest()
CTest(CTest& obj)
m_nTest = 1
~CTest()
~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>
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
{
2-4
//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;
};
//对象作为函数参数，调用普通成员函数
void foo_arg(CTest tArg)
{
2-2
/*
tArg是无名对象，程序执行完毕，遇到分号调用析构
*/
2-3
tArg.foo_test();
}
2-5
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_arg(t); //对象作为函数参数，调用成员函数
2-1
3 return 0;
}
/*
output:
CTest()
CTest(CTest& obj)
m_nTest = 1
~CTest()
~CTest()
*/
1 CTest t;执行调用类的构造函数，输出CTest();
2 将对象t作为参数传递给函数，调用CTest()类的拷贝构造，输出CTest(CTest& obj)
返回，foo_arg(t)，调用函数2-2
2-3 调用跳转到2-4，输出，m_nTest = 1,返回控制权回到2-3
2-5 因为tArg是无名对象，程序执行完毕，遇到分号，调用析构函数，输出~CTest()
3 调用析构函数，释放t
激活 Windows
转到"设置"以激活 Windows。