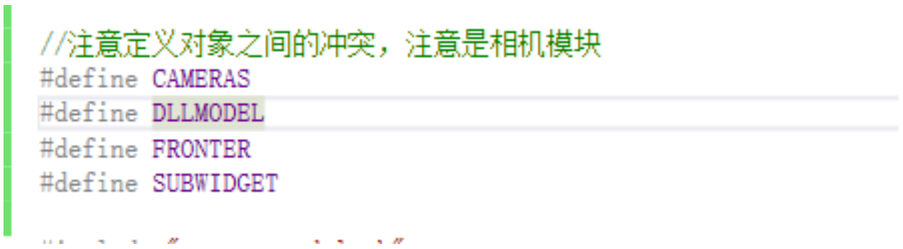
Introduction of the testing

Since all the designed modules use Qt, QTest is adopted as the main module of test module development. The detailed test is as follows:

Test Module Definition

Use macro defined module, which module needs to be defined just need to ensure that its own module can run, remove the main function, run QTest, because the test is a lib library form rather than the executable program, can also use the exe form.

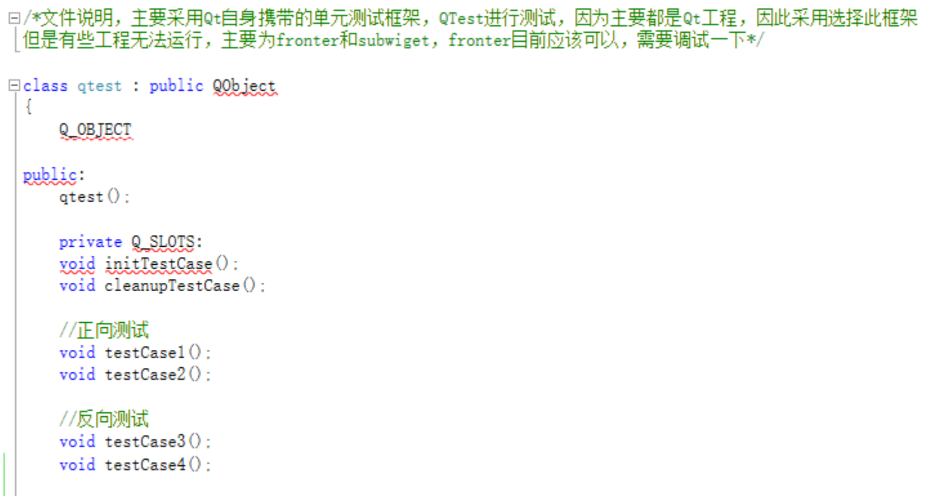
（1）Define module macro tests



```
//注意定义对象之间的冲突，注意是相机模块
#define CAMERAS
#define DLLMODEL
#define FRONTER
#define SUBWIDGET
```

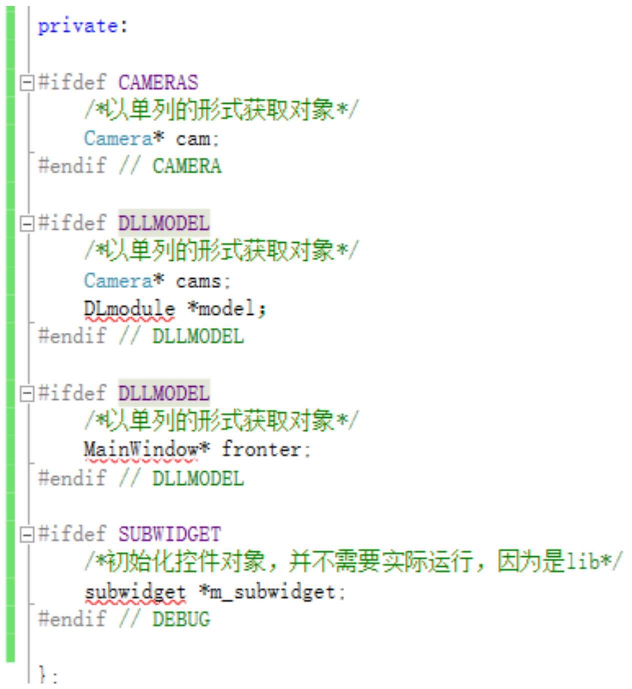
（2）Test the framework pattern with QTest

Set up the forward test and two reverse tests. The forward test mainly tests whether the call is correct, and the reverse test mainly calls the processing after the exception.



```
/*文件说明，主要采用Qt自身携带的单元测试框架，QTest进行测试，因为主要都是Qt工程，因此采用选择此框架
但是有些工程无法运行，主要为fronter和subwiget，fronter目前应该可以，需要调试一下*/

class qtest : public QObject
{
    Q_OBJECT

public:
    qtest();

    private Q_SLOTS:
    void initTestCase();
    void cleanupTestCase();

    //正向测试
    void testCase1();
    void testCase2();

    //反向测试
    void testCase3();
    void testCase4();
```

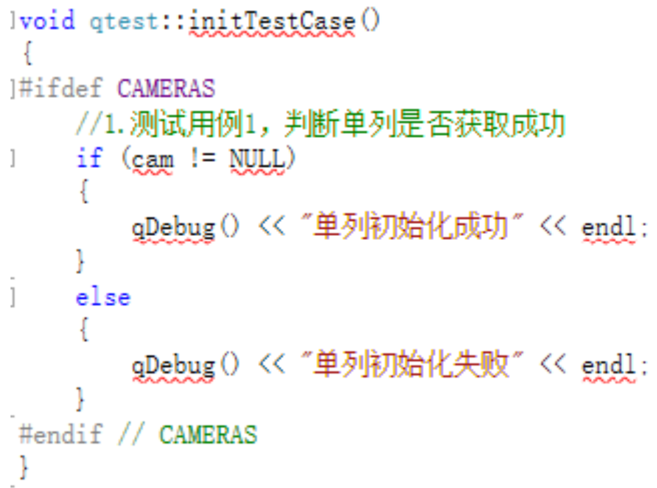
（3）Apply classes for each module



```
private:

#ifdef CAMERAS
    /*以单列的形式获取对象*/
    Camera* cam;
#endif // CAMERA

#ifdef DLLMODEL
    /*以单列的形式获取对象*/
    Camera* cams;
    DLmodule *model;
#endif // DLLMODEL

#ifdef DLLMODEL
    /*以单列的形式获取对象*/
    MainWindow* fronter;
#endif // DLLMODEL

#ifdef SUBWIDGET
    /*初始化控件对象，并不需要实际运行，因为是lib*/
    subwidget *m_subwidget;
#endif // DEBUG

};
```

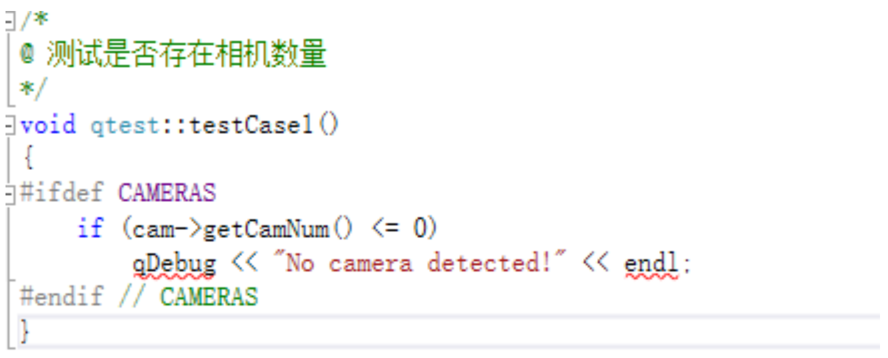
Camera module test

（1） The main function provided by the camera module is to open the camera and call it in the form of instance, while the test is mainly to test whether instance is successfully applied.



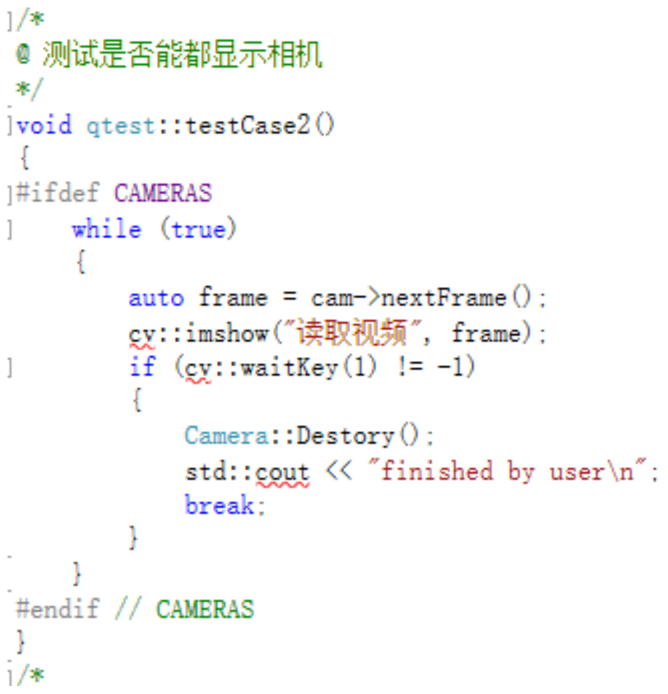
```
void qtest::initTestCase()
{
#ifdef CAMERAS
    //1.测试用例1，判断单列是否获取成功
    if (cam != NULL)
    {
        qDebug() << "单列初始化成功" << endl;
    }
    else
    {
        qDebug() << "单列初始化失败" << endl;
    }
#endif // CAMERAS
}
```

（2） Test whether exists a camera



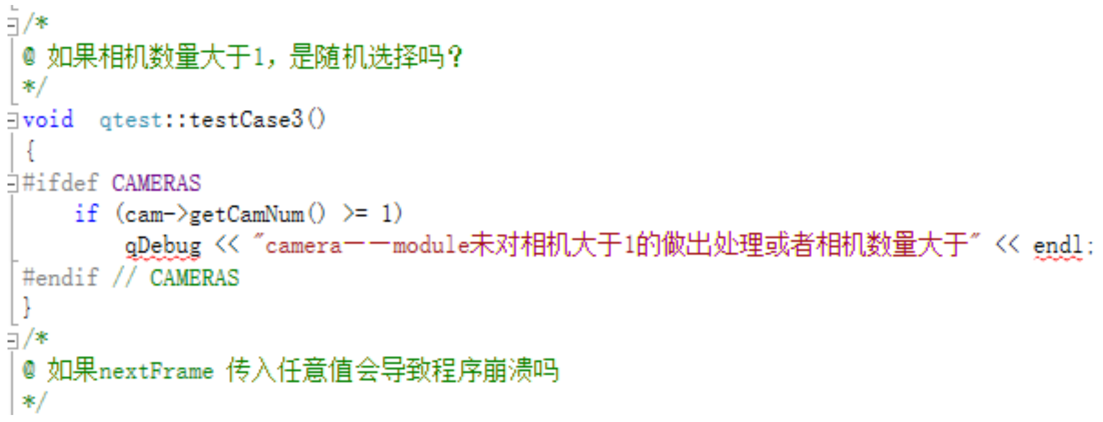
```
/*
@ 测试是否存在相机数量
*/
void qtest::testCase1()
{
#ifdef CAMERAS
    if (cam->getCamNum() <= 0)
        qDebug << "No camera detected!" << endl;
#endif // CAMERAS
}
```

(3) Test whether the module runs well



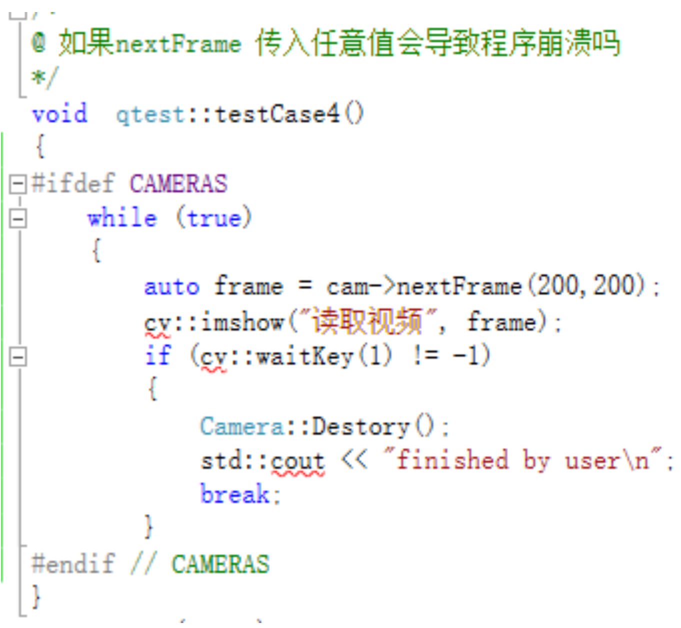
```
/*
@ 测试是否能都显示相机
*/
void qtest::testCase2()
{
#ifdef CAMERAS
    while (true)
    {
        auto frame = cam->nextFrame();
        cv::imshow("读取视频", frame);
        if (cv::waitKey(1) != -1)
        {
            Camera::Destory();
            std::cout << "finished by user\n";
            break;
        }
    }
#endif // CAMERAS
}
/*
```

(4) Test the camera if larger than one



```
/*
@ 如果相机数量大于1，是随机选择吗？
*/
void  qtest::testCase3()
{
#ifdef CAMERAS
    if (cam->getCamNum() >= 1)
        qDebug << "camera——module未对相机大于1的做出处理或者相机数量大于" << endl;
#endif // CAMERAS
}
/*
@ 如果nextFrame 传入任意值会导致程序崩溃吗
*/
```

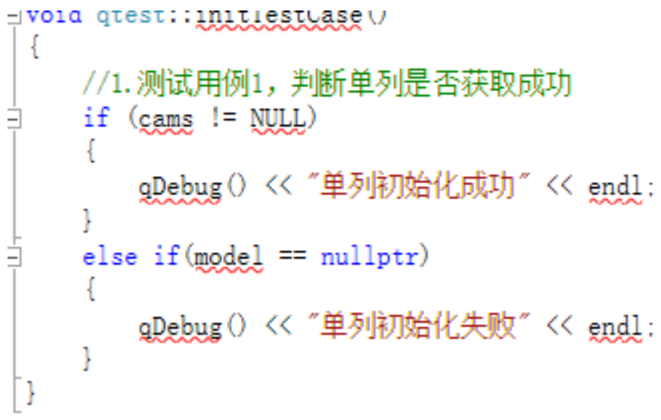
(5) Can the frame size passed in be any value



```
@ 如果nextFrame 传入任意值会导致程序崩溃吗
*/
void  qtest::testCase4()
{
#ifdef CAMERAS
    while (true)
    {
        auto frame = cam->nextFrame(200,200);
        cv::imshow("读取视频", frame);
        if (cv::waitKey(1) != -1)
        {
            Camera::Destory();
            std::cout << "finished by user\n";
            break;
        }
#endif // CAMERAS
}
```

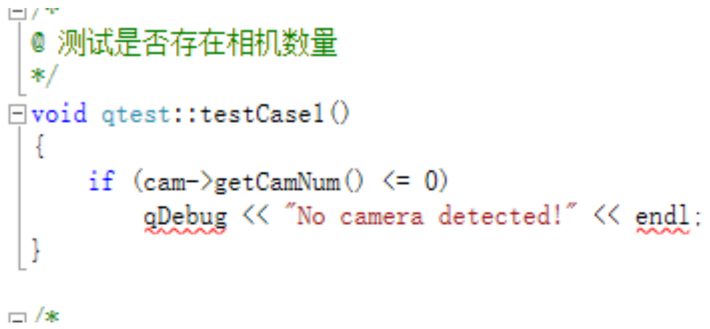
DL_module test

(1) Module initialization test



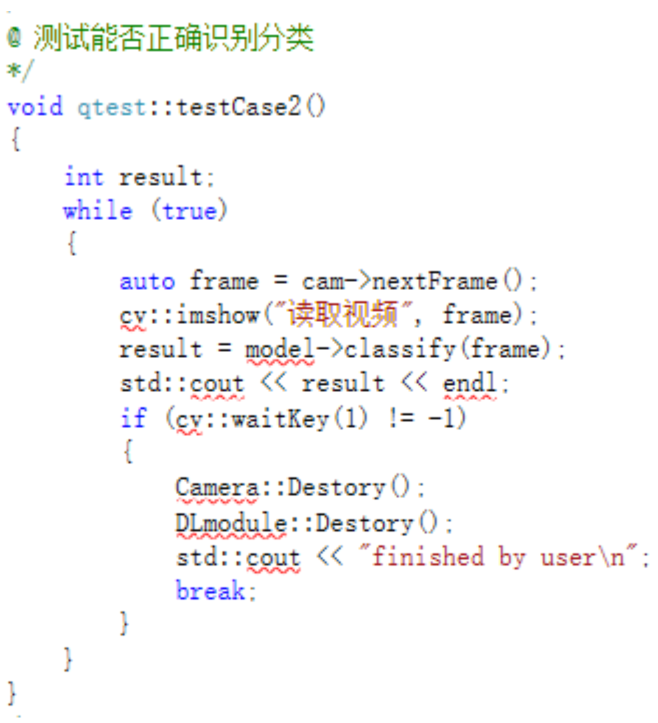
```
void qtest::initTestCase()
{
    //1.测试用例1，判断单列是否获取成功
    if (cams != NULL)
    {
        qDebug() << "单列初始化成功" << endl;
    }
    else if(model == nullptr)
    {
        qDebug() << "单列初始化失败" << endl;
    }
}
```

(2) Test whether there is a camera



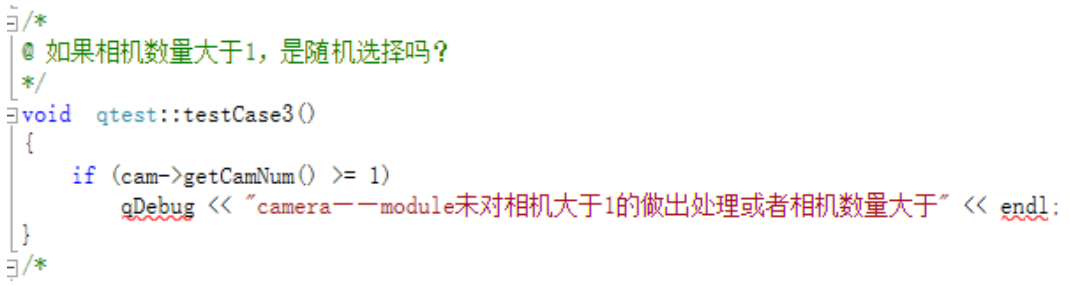
```
@ 测试是否存在相机数量
*/
void qtest::testCase1()
{
    if (cam->getCamNum() <= 0)
        qDebug << "No camera detected!" << endl;
}

/*
```

(3) Classification recognition test



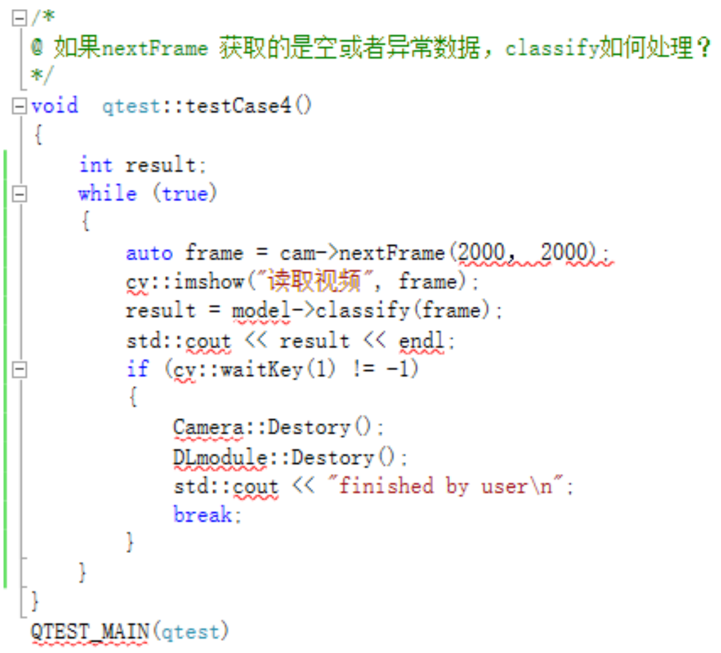
```
@ 测试能否正确识别分类
*/
void qtest::testCase2()
{
    int result;
    while (true)
    {
        auto frame = cam->nextFrame();
        cv::imshow("读取视频", frame);
        result = model->classify(frame);
        std::cout << result << endl;
        if (cv::waitKey(1) != -1)
        {
            Camera::Destory();
            DLmodule::Destory();
            std::cout << "finished by user\n";
            break;
        }
    }
}
```

(4) What if the number of test cameras is greater than one



```
/*
@ 如果相机数量大于1，是随机选择吗？
*/
void  qtest::testCase3()
{
    if (cam->getCamNum() >= 1)
        qDebug << "camera——module未对相机大于1的做出处理或者相机数量大于" << endl;
}
/*
```

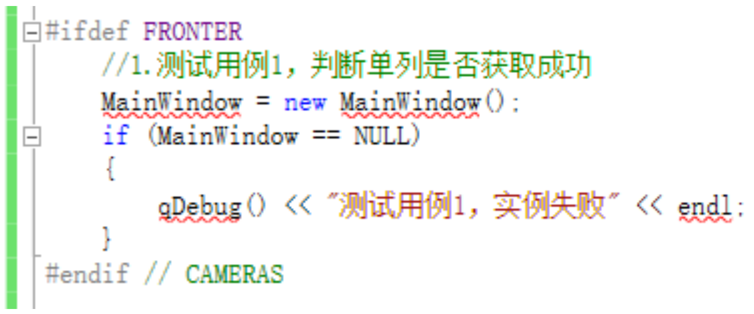
(5) Abnormal test of width and height



```
/*
@ 如果nextFrame 获取的是空或者异常数据，classify如何处理？
*/
void  qtest::testCase4()
{
    int result;
    while (true)
    {
        auto frame = cam->nextFrame(2000,  2000);
        cv::imshow("读取视频", frame);
        result = model->classify(frame);
        std::cout << result << endl;
        if (cv::waitKey(1) != -1)
        {
            Camera::Destory();
            DLmodule::Destory();
            std::cout << "finished by user\n";
            break;
        }
    }
}
QTEST_MAIN(qtest)
```

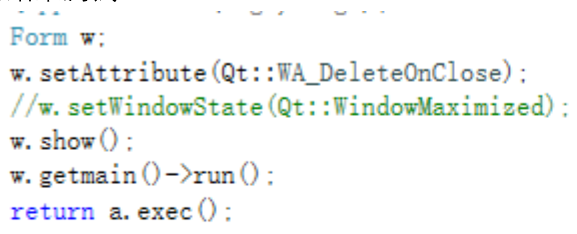
Fronter test

(1) Module initialization test



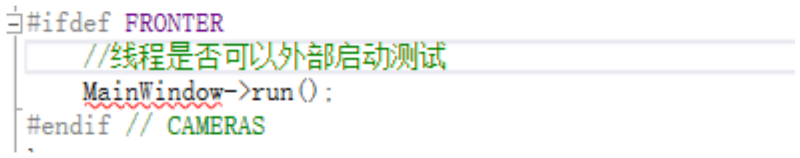
```
#ifdef FRONTER
    //1.测试用例1，判断单列是否获取成功
    MainWindow = new MainWindow();
    if (MainWindow == NULL)
    {
        qDebug() << "测试用例1，实例失败" << endl;
    }
#endif // CAMERAS
```

(2) 运行结果测试



```
Form w;
w.setAttribute(Qt::WA_DeleteOnClose);
//w.setWindowState(Qt::WindowMaximized);
w.show();
w.getmain()->run();
return a.exec();
```

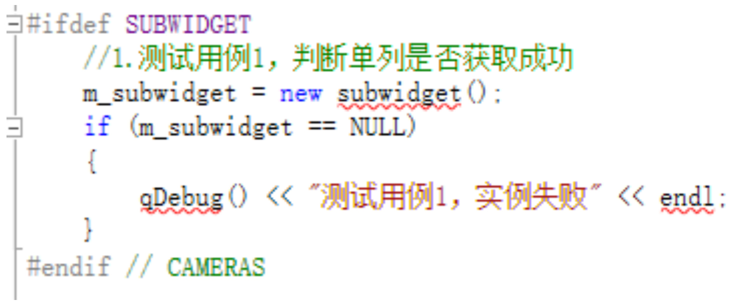
(3) 是否可以通过外部线程启动测试



```
#ifdef FRONTER
    //线程是否可以外部启动测试
    MainWindow->run();
#endif // CAMERAS
```

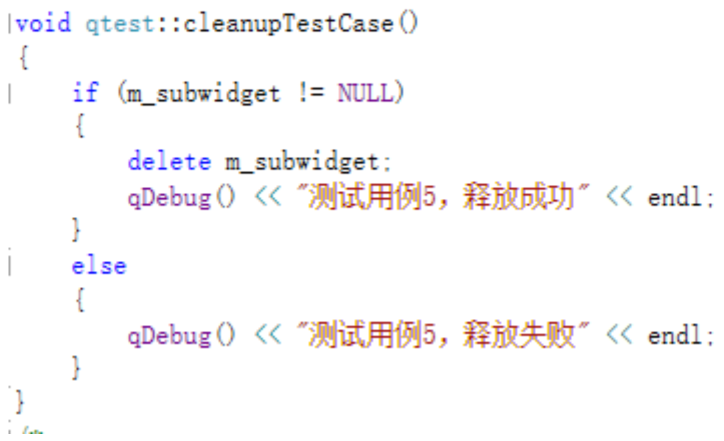
Subwidget test

(1) Module initialization test

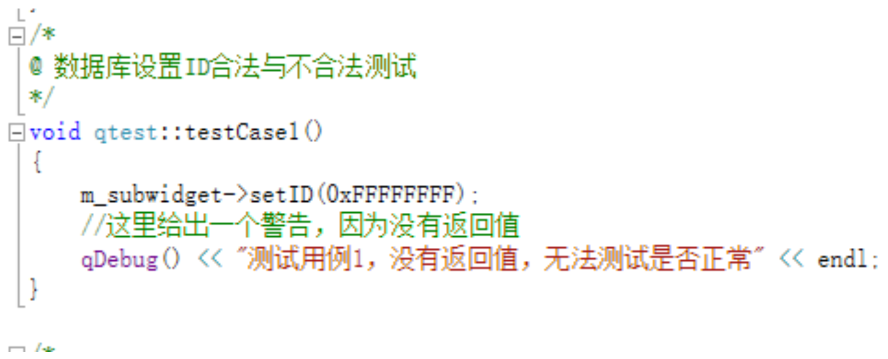


```
#ifdef SUBWIDGET
    //1.测试用例1，判断单列是否获取成功
    m_subwidget = new subwidget();
    if (m_subwidget == NULL)
    {
        qDebug() << "测试用例1，实例失败" << endl;
    }
#endif // CAMERAS
```

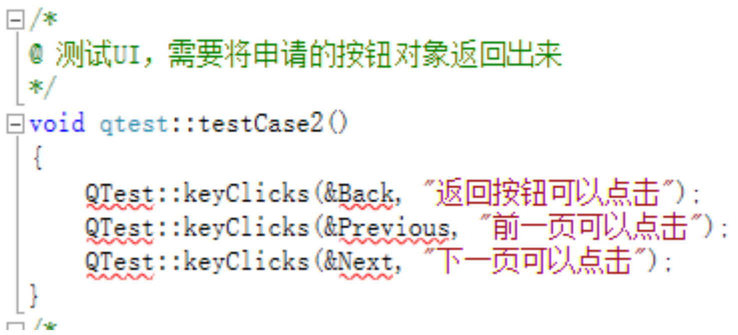
(2) Whether the release was successful


```
void qtest::cleanupTestCase()
{
    if (m_subwidget != NULL)
    {
        delete m_subwidget;
        qDebug() << "测试用例5，释放成功" << endl;
    }
    else
    {
        qDebug() << "测试用例5，释放失败" << endl;
    }
}
```

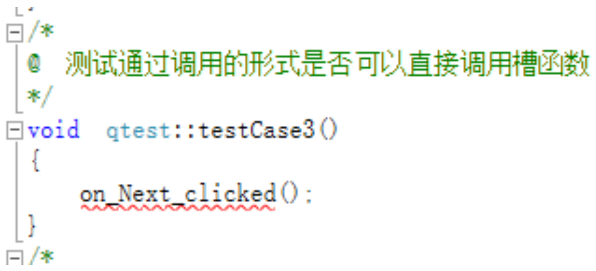
(3)　Whether the setID is correct


```
/*
@ 数据库设置ID合法与不合法测试
*/
void qtest::testCase1()
{
    m_subwidget->setID(0xFFFFFFFF);
    //这里给出一个警告，因为没有返回值
    qDebug() << "测试用例1，没有返回值，无法测试是否正常" << endl;
}
```

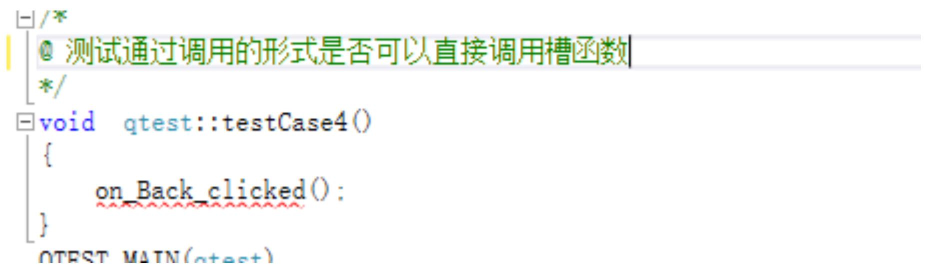
(4)　The button test


```
/*
@ 测试UI，需要将申请的按钮对象返回出来
*/
void qtest::testCase2()
{
    QTest::keyClicks(&Back, "返回按钮可以点击");
    QTest::keyClicks(&Previous, "前一页可以点击");
    QTest::keyClicks(&Next, "下一页可以点击");
}
```

(5)　Test whether private functions can be externally called


```
/*
@  测试通过调用的形式是否可以直接调用槽函数
*/
void  qtest::testCase3()
{
    on_Next_clicked();
}
```

(6)　Test whether private functions can be externally called



```
/*
@ 测试通过调用的形式是否可以直接调用槽函数
*/
void  qtest::testCase4()
{
    on_Back_clicked();
}
QTEST_MAIN(qtest)
```

Test Results

```
********* Start testing of Qtest *********
Config: Using QtTest library 5.10.1, Qt 5.10.1 (i386-little_endian-
ilp32 shared (dynamic) debug build; by GCC 5.3.0)
PASS : Qtest::initTestCase()
PASS : Qtest::testCase1()
PASS: Qtest::testCase2()
PASS : Qtest::testCase3()
PASS: Qtest::testCase4()
Totals: 4 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
********* Finished testing of Qtest *********
```