Experiment #7

1.

//TestException.h

#ifndef TESTEXCEPTION\_H

#define TESTEXCEPTION\_H

class TestException : public runtime\_error {

public:

TestException()

:runtime\_error("This is a test\n") {}

};

#endif // !TESTEXCEPTION\_H

//////////////////////////////////////////////////

//main.cpp

#include<iostream>

#include<exception>

using namespace std;

#include"TestException.h"

int main() {

try {

try {

throw TestException();

}

catch (TestException& testexception) {

cout << testexception.what();

throw TestException();

//throw it again

}//first layer to catch the exception

catch (...) {

cout << "abnormal program termination (inner layer)\n";

}

}

catch (...) {

cout << "abnormal program termination\n";

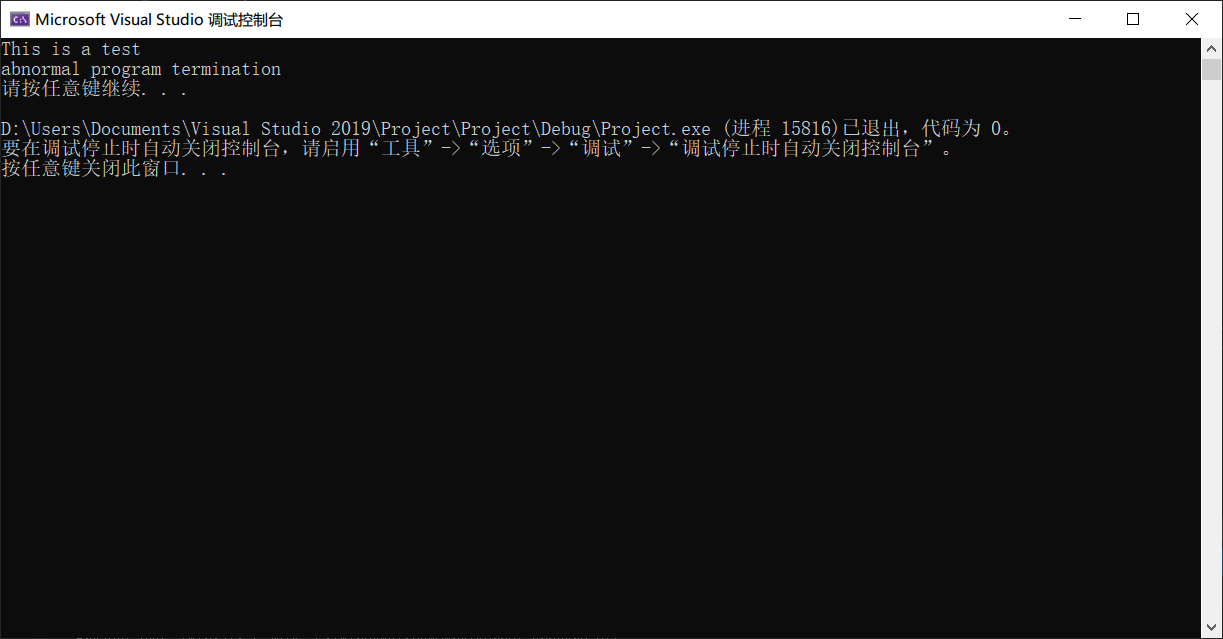
}//second layer to catch the exception throwed in the "catch(){}" above

system("pause");

return 0;

}

//In order to catch the second exception, we should add another "try - catch" construction outside it



2.

//Item.h

#ifndef ITEM\_H

#define ITEM\_H

class Item {

public:

Item(int v)

:value(v) {

cout << "Item " << value << " constructor called\n";

if (value == 3)

throw runtime\_error("An exception was thrown\n");

}

~Item() {

cout << "Item " << value << " destructor called\n";

}

private:

int value;

};

#endif // !ITEM\_H

//////////////////////////////////////////////////

//main.cpp

#include<iostream>

#include<exception>

using namespace std;

#include"Item.h"

int main() {

cout << "Constructing an object of class ItemGroup\n";

try {

Item i1(1), i2(2), i3(3);

}

catch(runtime\_error &ValueEqualTo3){

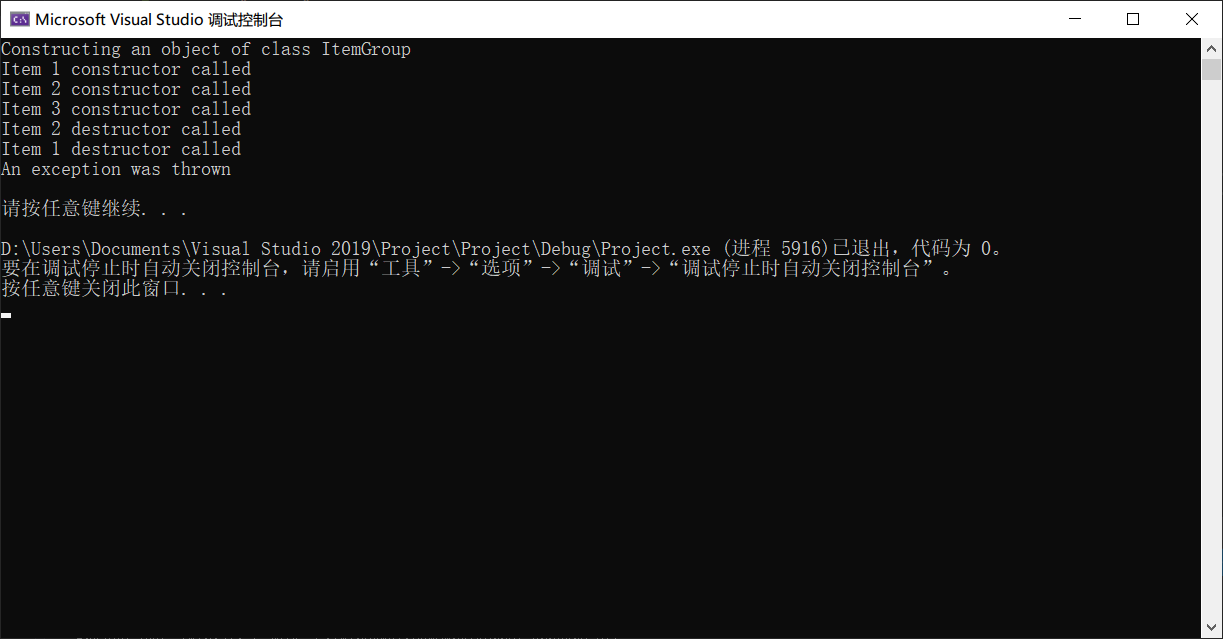
cout << ValueEqualTo3.what() << endl;

}

system("pause");

return 0;

}



3.

//TestException.h

#ifndef TESTEXCEPTION\_H

#define TESTEXCEPTION\_H

class TestException : public runtime\_error {

public:

TestException()

:runtime\_error("Exception") {}

private:

};

#endif // !TESTEXCEPTION\_H

//////////////////////////////////////////////////

//main.cpp

#include<iostream>

#include<exception>

using namespace std;

#include"TestException.h"

void g() {

try {

throw TestException();

}

catch (...) {

cout << "Exception caught in function g(). Rethrowing...\n";

throw TestException();

}

}

int main() {

try {

g();

}

catch (TestException& exception) {

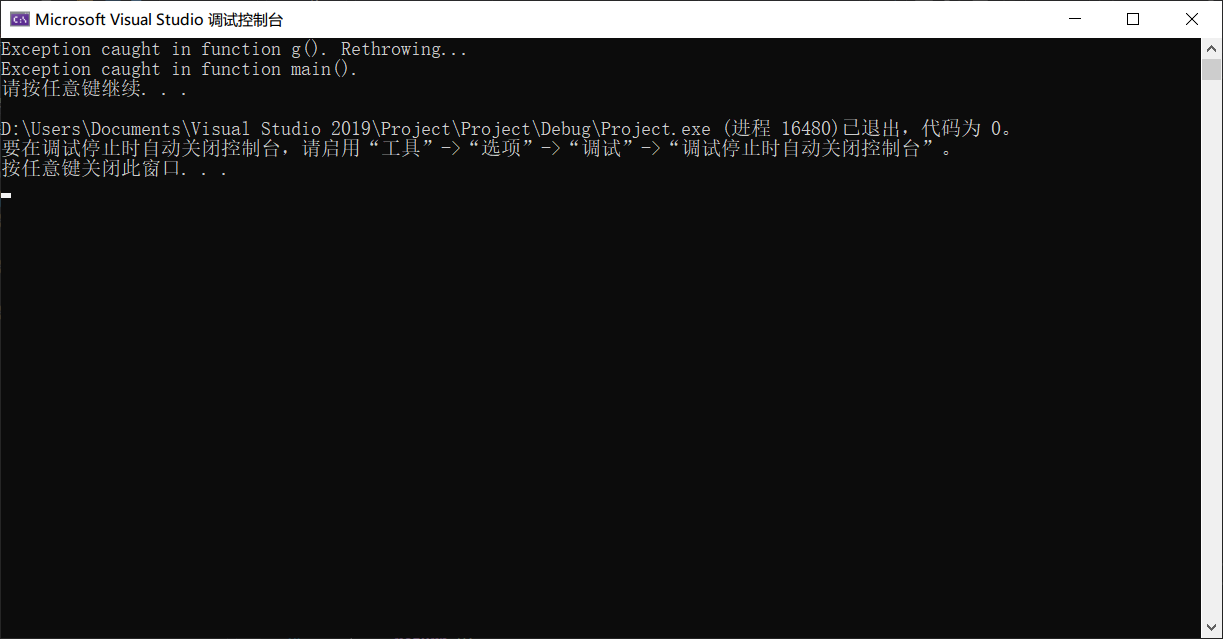
cout << "Exception caught in function main().\n";

}

system("pause");

return 0;

}



4.

//TestException.h

#ifndef TESTEXCEPTION\_H

#define TESTEXCEPTION\_H

class TestException : public runtime\_error {

public:

TestException()

:runtime\_error("TestException") {}

private:

};

#endif // !TESTEXCEPTION\_H

//////////////////////////////////////////////////

//main.cpp

#include<iostream>

#include<exception>

using namespace std;

#include"TestException.h"

void f() {

throw TestException();//Throw

//Will not be executed:

cout << "In f(): Throwed TestException\n";

}

void g() {

f();//Call f()

//Will not be executed:

try {

f();

}

catch (TestException& testexception) {

cout << "In g(): Caught " << testexception.what() << endl;

}

}

void h() {

g();//Call g()

//Will not be executed:

try {

g();

}

catch (TestException& testexception) {

cout << "In h(): Caught " << testexception.what() << endl;

}

}

int main() {

try {

h();//Call h(), and meet the exxception

}

catch (TestException& testexception) {

cout << "In main: Caught " << testexception.what() << endl;//Catch

}

system("pause");

return 0;

}

