“判断三角形”软件

测试报告

姓名：马晨夕

学号：**14126126**

Lab Project Introduction

【Lab Project Name】：Software Testing based on Visual Studio

【Lab Project Purposes】：

(1) Understand the basic procedures and concepts for software testing;

(2) Learn basic skills for using testing framework embedded in Visual Studio;

(3) Learn how to use white box and black box techniques for guiding the design of test cases;

(4) Learn how to do the coded UI test with Visual Studio.

【Lab Environment】：

(1) Hardware Environment：

(2) OS Environment：win7

(3) Programming Language For Test Code：c#

(4) Programming Language For Sample Code：c#

(5) Network Environment：

(6) Others：

Software: Visual studio 2012

【References】：

[1] Java. http://zh.wikipedia.org/wiki/Java

[2] JUnit. http://www.junit.org/

[3] Tom，Jerry。Introduction to Software Testing. Tsing Hua Press，2010.

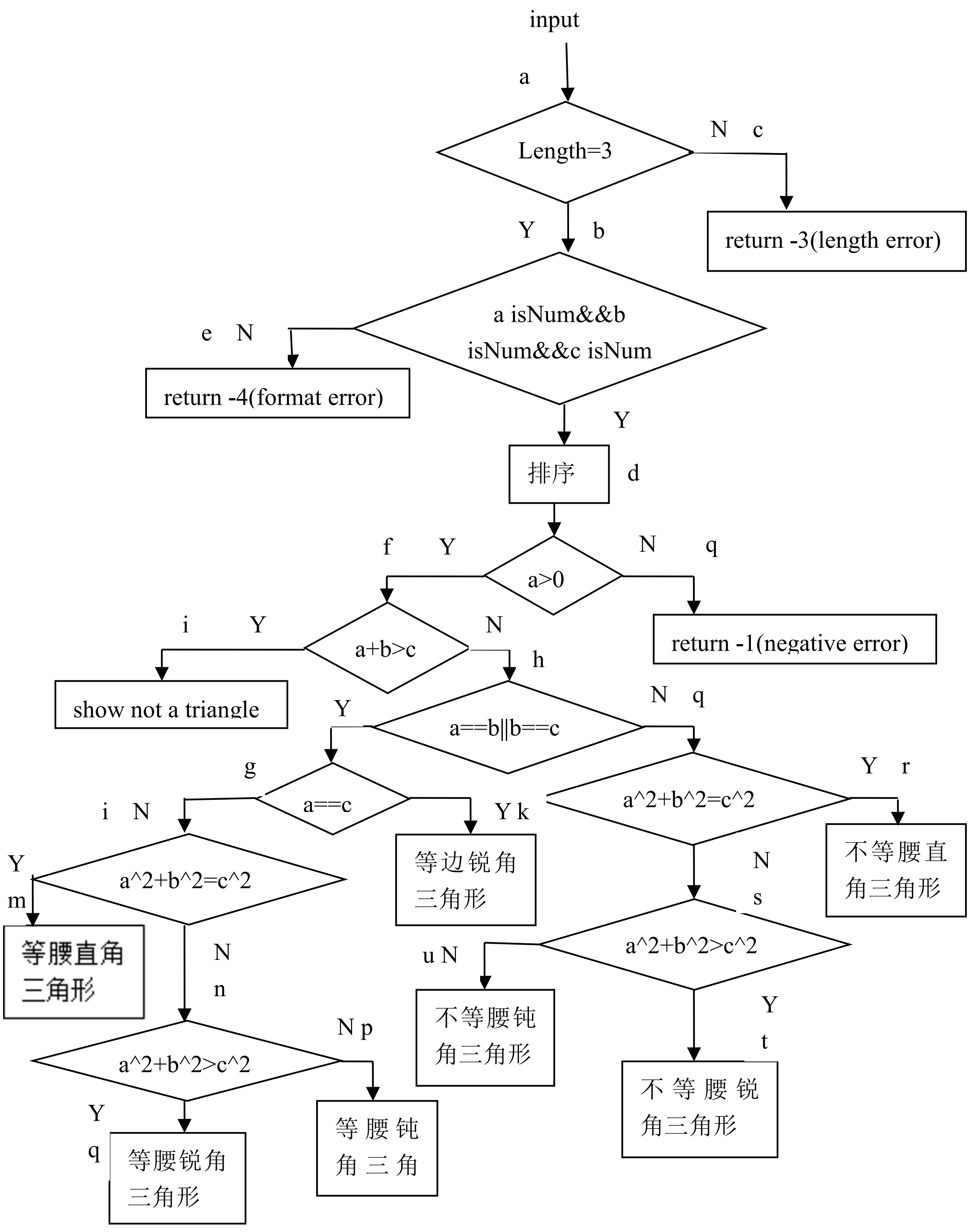
Lab Project (Main Part)

【Testing Plan Design】：

Testing Plan A：White Box Test

(1) Testing Technique：Complete Coverage Strategy

(2) Flowchart of the sample code to be tested：



(3) Test Case Design (Complete Coverage Strategy)

According to the definition of complete coverage strategy, we need to design test cases to satisfy both condition combination coverage and path coverage. Therefore, we firstly design test cases which satisfy condition combination coverage. The design details are shown as listed below:

There are 4 conditions in the sample code:

Condition 1: { length=3 }

Condition 2: { a isNum}

Condition 3: { b isNum}

Condition 4: { c isNum}

Condition 5: {a>0}

Condition 6: {a+b>c }

Condition 7: {a=b}

Condition 8: {b=c}

Condition 9: {a=c}

Condition 10: {a^2+b^2=c^2}

Condition 11: {a^2+b^2>c^2}

Condition 12: {a^2+b^2=c^2}

Condition 13: {a^2+b^2>c^2}

In the sample code, there are 13 conditions which can form 28 possible combinations.

All 28 possible combinations of 13 conditions are listed below:

① length=3 T1

② length≠3 F1

③ a isNUM,b isNum, c isNum T2 T3 T4

④ a isNum, b isNotNum, c isNum T2 F3 T4

⑤ a isNum, b isNum, c isNotNum T2 T3 F4

⑥ a isNum, b isNotNum, c isNotNum T2 F3 F4

⑦ a isNotNum, b isNum, c isNum   F2 T3 T4

⑧ a isNotNum, b isNotNum, c isNum F2 F3 T4

⑨ a isNotNum, b isNum, c isNotNum F2 T3 F4

⑩ a isNotNum, b isNotNum, c isNotNum F2 F3 F4

⑪ a>0 T5

⑫ a<=0 F5

⑬ a+b>c T6

⑭ a+b<=c F6

⑮ a=b, b=c T7 T8

⑯ a=b, b≠c T7 F8

⑰ a≠b, b=c F7T8

⑱ a≠b, b≠c F7F8

⑲ a=c T9

⑳ a≠c F9

㉑ a^2+b^2=c^2 T10

㉒ a^2+b^2≠c^2 F10

㉓ a^2+b^2>c^2 T11

㉔ a^2+b^2<=c^2 F11

㉕ a^2+b^2=c^2 T12

㉖ a^2+b^2≠c^2 F12

㉗ a^2+b^2>c^2 T13

㉘ a^2+b^2<=c^2 F14

Because the values of a, b, c have been sorted, so the they are in order in the test cases.

Therefore, according to the analysis above and the definition of condition combination coverage, 8 test cases are design as shown in Table 1.

Test Case ID	a	b	c	length	Path	Combination Covered	Condition Covered
Case1	1	2		2	a c	②	F1
Case2	1	a	1	3	a b e	① ④	T1 T2 F3 T4
Case3	1	1	a	3	a b e	① ⑤	T1 T2 T3 F4
Case4	1	a	a	3	a b e	① ⑥	T1 T2 F3 F4
Case5	a	1	1	3	a b e	① ⑦	T1 F2 T3 T4
Case6	a	a	1	3	a b e	① ⑧	T1 F2 F3 T4
Case7	a	1	a	3	a b e	① ⑨	T1 F2 T3 F4
Case8	a	a	a	3	a b e	① ⑩	T1 F2 F3 F4
Case9	-1	1	1	3	a b d g	① ③ ⑫	T1 T2 T3 T4 F5
Case10	1	1	2	3	a b d f i	① ③ ⑪ ⑭	T1 F2 F3 F4 T5 F6
Case11	1	1	1	3	a b d f h g k	① ③ ⑪ ⑬ ⑮ ⑲	T1 T2 T3 T4 T5 T6 T7 T8 T9
Case12	1	1	2ˆ0.5	3	a b d f h g l m	① ③ ⑪ ⑬ ⑯ ⑳ ㉑	T1 T2 T3 T4 T5 T6 T7 T8 F9 T10
Case13	2	2	3	3	a b d f h g l n o	① ③ ⑪ ⑬ ⑯ ⑳ ㉒ ㉔	T1 T2 T3 T4 T5 T6 T7 F8 F9 F10 F11
Case14	1	2	2	3	a b d f h g l n p	① ③ ⑪ ⑬ ⑰ ⑳ ㉒ ㉓	T1 T2 T3 T4 T5 T6 T7 F8 F9 F10 T11

Case15	3	4	5	3	a b d f h q r	①③⑪⑬ ⑱㉕	T1 T2 T3 T4 T5 T6 F7 F8 T12
Case16	4	5	6	3	a b d f h q s t	①③⑪⑬ ⑱㉖㉗	T1 T2 T3 T4 T5 T6 F7 F8 F12 T13
Case17	4	5	7	3	a b d f h q s u	①③⑪⑬ ⑱㉖㉘	T1 T2 T3 T4 T5 T6 F7 F8 F12 F13

Table 1

Path Coverage:

Test case	a	b	c	Path covered
Case 1	1	2		ac
Case2	1	a	b	abe
Case 3	-1	1	1	abdg
Case 4	1	1	2	abdfi
Case 5	1	1	1	abdfhg
Case 6	1	1	2ˆ0.5	abdfhglm
Case 7	2	2	3	abdfhglno
Case 8	1	2	2	abdfhglnp
Case 9	3	4	5	abdfhqr
Case 10	4	5	6	abdfhgst
Case 11	4	5	7	abdfhgsu

Then, we found that all paths are covered by test cases in Table 1.

So the complement coverage is

Test Case ID	a	b	c	length	Path	Combination Covered	Condition Covered
Case1	1	2		2	a c	②	F1
Case2	1	a	1	3	a b e	①④	T1 T2 F3 T4
Case3	1	1	1.1.1	3	a b e	①⑤	T1 T2 T3 F4

	Case4	1	a	a	3	a b e	① ⑥	T1 T2 F3 F4
	Case5	a	1	1	3	a b e	① ⑦	T1 F2 T3 T4
	Case6	a	a	1	3	a b e	① ⑧	T1 F2 F3 T4
	Case7	a	1	a	3	a b e	① ⑨	T1 F2 T3 F4
	Case8	a	a	a	3	a b e	① ⑩	T1 F2 F3 F4
	Case9	-1	1	1	3	a b d g	① ③ ⑫	T1 T2 T3 T4 F5
	Case10	1	1	2	3	a b d f i	① ③ ⑪ ⑭	T1 F2 F3 F4 T5 F6
	Case11	1	1	1	3	a b d f h g k	① ③ ⑪ ⑬ ⑮ ⑲	T1 T2 T3 T4 T5 T6 T7 T8 T9
	Case12	1	1	2ˆ0.5	3	a b d f h g l m	① ③ ⑪ ⑬ ⑯ ⑳ ㉑	T1 T2 T3 T4 T5 T6 T7 T8 F9 T10
	Case13	2	2	3	3	a b d f h g l n o	① ③ ⑪ ⑬ ⑯ ⑳ ㉒ ㉔	T1 T2 T3 T4 T5 T6 T7 F8 F9 F10 F11
	Case14	1	2	2	3	a b d f h g l n p	① ③ ⑪ ⑬ ⑰ ⑳ ㉒ ㉓	T1 T2 T3 T4 T5 T6 T7 F8 F9 F10 T11
	Case15	3	4	5	3	a b d f h q r	① ③ ⑪ ⑬ ⑱ ㉕	T1 T2 T3 T4 T5 T6 F7 F8 T12
	Case16	4	5	6	3	a b d f h q s t	① ③ ⑪ ⑬ ⑱ ㉖ ㉗	T1 T2 T3 T4 T5 T6 F7 F8 F12 T13
	Case17	4	5	7	3	a b d f h q s u	① ③ ⑪ ⑬ ⑱ ㉖ ㉘	T1 T2 T3 T4 T5 T6 F7 F8 F12 F13

Testing Plan B：Black Box Test

(1) Testing Technique：Equivalent Class Partitioning + Boundary Value Analysis

Boundary value analysis: the input condition is input number greater than 0, because the values of a, b, c have been sorted, so they are in order in the test cases.

The boundary values are -1, 0,1.

The length of input is 3, so the boundary of it is 2,3,4

(2) Flowchart of the sample code to be tested：(as same as in Plan A)

The requirement condition of a,b,c is

Condition 1: a is not null

Condition 2: b is not null

Condition 3:c is not null

Condition 4: a is number

Condition 5: b is number

Condition 6: c is number

Condition 7: a>0

Condition 8: b>0

Condition 9: c>0

Condition 10: a+b>c

The equivalent class:

Input condition	Valid equivalent class	ID	Invalid equivalent class	ID

Input a, b, c three numbers	**a is not null** **b is not null** **c is not null** **a is number** **b is number** **c is number** **a>0** **b>0** **c>0** **a+b>c**	**1**	**The length of a,b,c is not 3**	**10**
	Satisfy 1 **a=b or b=c**	**2**	**a, b, c are not all numbers**	**11**
	Satisfy 1, 2 **a=c**	**3**	**a, b ,c are not all in the boundary(0,500)**	**12**
	Satisfy 1,2 a!=c and a^2+b^2=c^2	**4**	**a+b<=c**	**13**
	Satisfy 1,2 a!=c and a^2+b^2>c^2	**5**		
	Satisfy 1,2 a!=c and a^2+b^2<c^2	**6**		
	Satisfy 1, a!=b and b!=c, a^2+b^2=c^2	**7**		
	Satisfy 1, a!=b and b!=c, a^2+b^2>c^2	**8**		

	Satisfy 1, a!=b and b!=c, a^2+b^2<c^2	**9**		

Test case ID	**input**				**Equivalent coverage**	**Expected result**
	a	**b**	**c**			
Case 1	**1**	**2**			**10**	**Length error**
Case 2	**1**	**2**	**3**	**4**	**10**	**Length error**
Case 3	**1**	**a**	**1**		**11**	**Format error**
Case 4	**a**	**a**	**a**		**11**	**Format error**
Case 5	**0**	**1**	**1**		**12**	**Not positive error**
Case 6	**1**	**1**	**2**		**13**	不是三角形
Case 7	**1**	**1**	**1**		**1,2,3**	等边锐角三角形
Case 8	**1**	**1**	**2^0.5**		**1,2,4**	等腰直角三角形
Case 9	**1**	**2**	**2**		**1,2,5**	等腰锐角三角形
Case 10	**2**	**2**	**3**		**1,2,6**	等腰钝角三角形
Case 11	**3**	**4**	**5**		**1,7**	不等腰直角三角形
Case 12	**4**	**5**	**6**		**1,8**	不等要锐角三角形
Case 13	**4**	**5**	**7**		**1,9**	不等腰钝角三角形
Case 14	**3**	**3**	**3.000001**		**1,2,5**	不等要锐角三角形
Case 15	**2.999999**	**3**	**5**		**1,8**	不等腰锐角三角形
Case 16	**3**	**4**	**5.0001**		**1,9**	不等腰钝角三角形
Case 17	**1**	**1**	**1.414**		**1,2,5**	等边锐角三角形

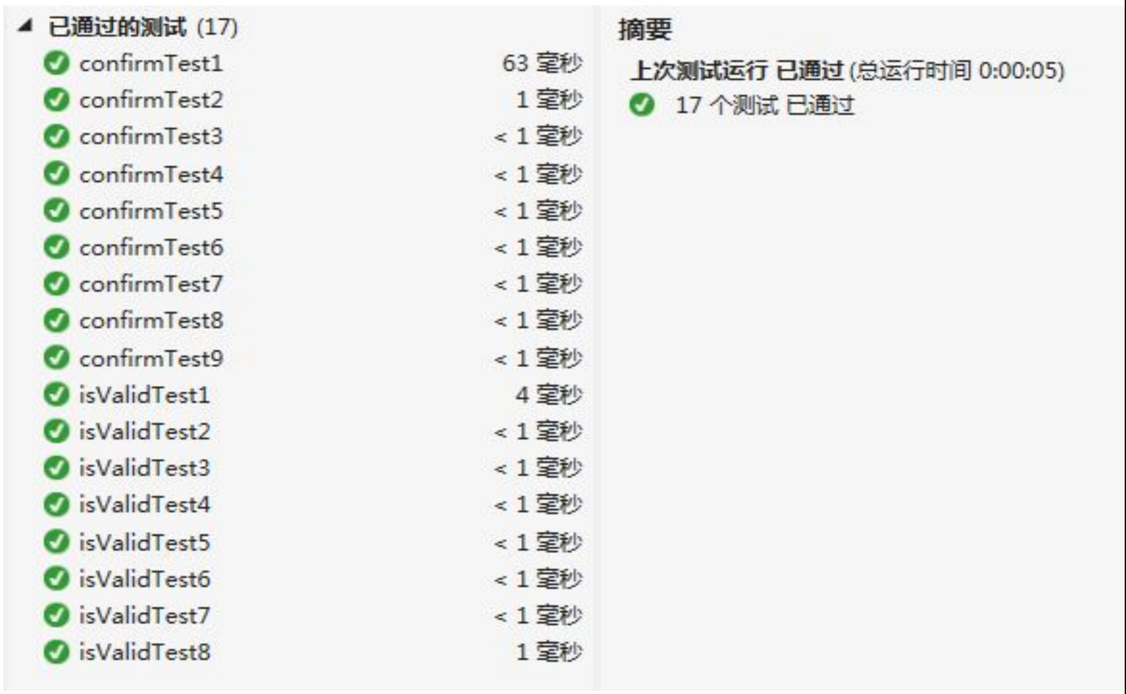
Testing Plan C: Coded UI Test

Test case	Texbox1	Label2
Case 1		A,B,C 都不可以为空
Case 2	1,2,3	不是三角形
Case 3	1,1,1	等边锐角三角形
Case 4	1,2,2	等腰锐角三角形
Case 5	2,2,3	等腰钝角三角形
Case 6	3,4,5	不等腰直角三角形
Case 7	4,5,6	不等腰锐角三角形
Case 8	4,5,7	不等腰钝角三角形

【Testing Results For Testing Plan A】：

The testing results for all test cases in Plan A are shown as following:

Test Case ID	a	b	c	Expected output	Actual output	Test result
Case1	1	2		−3	-3	Pass
Case2	1	a	1	−4	-4	Pass
Case3	1	1	1. 1. 1	−4	-4	Pass
Case4	1	a	a	−4	-4	Pass
Case5	a	1	1	−4	−4	Pass
Case6	a	a	1	−4	−4	Pass
Case7	a	1	a	−4	−4	Pass
Case8	a	a	a	−4	−4	Pass
Case9	−1	1	1	−1	−1	Pass
Case10	1	1	2	−2	−2	Pass
Case11	1	1	1	1	1	Pass
Case12	1	1	2ˆ0. 5	2	2	Pass
Case13	2	2	3	4	4	Pass
Case14	1	2	2	3	3	Pass
Case15	3	4	5	5	5	Pass
Case16	4	5	6	6	6	Pass
Case17	4	5	7	7	7	Pass



◢ 已通过的测试 (17)

```
✓ confirmTest1        63 毫秒
✓ confirmTest2         1 毫秒
✓ confirmTest3       < 1 毫秒
✓ confirmTest4       < 1 毫秒
✓ confirmTest5       < 1 毫秒
✓ confirmTest6       < 1 毫秒
✓ confirmTest7       < 1 毫秒
✓ confirmTest8       < 1 毫秒
✓ confirmTest9       < 1 毫秒
✓ isValidTest1         4 毫秒
✓ isValidTest2       < 1 毫秒
✓ isValidTest3       < 1 毫秒
✓ isValidTest4       < 1 毫秒
✓ isValidTest5       < 1 毫秒
✓ isValidTest6       < 1 毫秒
✓ isValidTest7       < 1 毫秒
✓ isValidTest8         1 毫秒
```

摘要

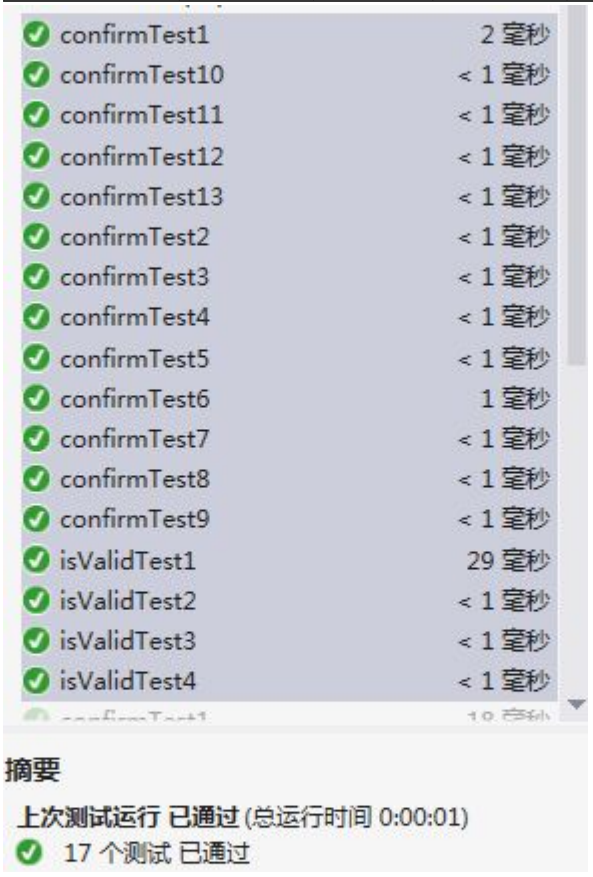
上次测试运行 已通过 (总运行时间 0:00:05)

✓ 17 个测试 已通过

【Testing Results For Testing Plan B】：

Test case ID	input				Expected output	Actual output	Test result
	a	b	c				
Case 1	1	2			-3	-3	pass
Case 2	1	2	3	4	-3	-3	pass
Case 3	1	a	1		-4	-4	pass
Case 4	a	a	a		-4	-4	pass
Case 5	0	1	1		-1	-1	pass
Case 6	1	1	2		-2	-2	pass
Case 7	1	1	1		1	1	pass
Case 8	1	1	2^0.5		2	2	pass
Case 9	1	2	2		3	3	pass
Case 10	2	2	3		4	4	pass
Case 11	3	4	5		5	5	pass
Case 12	4	5	6		6	6	pass
Case 13	4	5	7		7	7	pass
Case 14	3	3	3.000001		3	3	pass

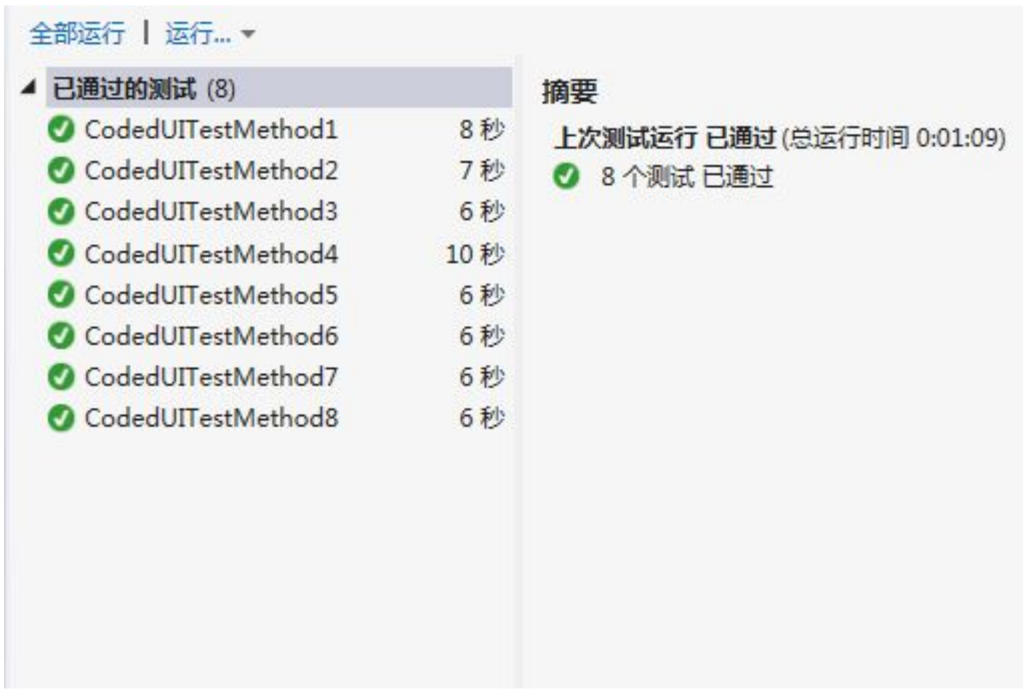
Case 15	**2.999999**	**3**	**5**		**7**	**7**	**pass**
Case 16	**3**	**4**	**5.0001**		**7**	**7**	**pass**
Case 17	**1**	**1**	**1.414**		**3**	**3**	**pass**



```
confirmTest1      2 毫秒
confirmTest10   < 1 毫秒
confirmTest11   < 1 毫秒
confirmTest12   < 1 毫秒
confirmTest13   < 1 毫秒
confirmTest2    < 1 毫秒
confirmTest3    < 1 毫秒
confirmTest4    < 1 毫秒
confirmTest5    < 1 毫秒
confirmTest6      1 毫秒
confirmTest7    < 1 毫秒
confirmTest8    < 1 毫秒
confirmTest9    < 1 毫秒
isValidTest1     29 毫秒
isValidTest2    < 1 毫秒
isValidTest3    < 1 毫秒
isValidTest4    < 1 毫秒

摘要
上次测试运行 已通过 (总运行时间 0:00:01)
17 个测试 已通过
```

【Testing Results For Testing Plan C】:



```
已通过的测试 (8)
CodedUITestMethod1     8 秒
CodedUITestMethod2     7 秒
CodedUITestMethod3     6 秒
CodedUITestMethod4    10 秒
CodedUITestMethod5     6 秒
CodedUITestMethod6     6 秒
CodedUITestMethod7     6 秒
CodedUITestMethod8     6 秒

摘要
上次测试运行 已通过 (总运行时间 0:01:09)
8 个测试 已通过
```

:

【Conclusion】:

From this subject, I have learnt that the basic procedures and concepts for software testing, how to do the white box and black box test and how to do the coded UI test with visual studio 2012.

What's more, I have also obtain the basic skills for testing framework embedded in visual studio, how to design the test cases. I find that software testing is also a significant technique for us to code the high - quality program. We should study it carefully and eliminate the errors in our code.

Finally, I have learnt some knowledge about C# as well and been able to find the errors and correct the errors by myself. I have enhanced the ability to study by myself and the interest in coding, too.