**江苏科技大学**

**实 验 报 告**

课 程： 软件工程与软件安全

学 院： 计算机学院

姓 名： 陈四贵

学 号： 182210710119

班 级： 1822107101

指导老师： 李红梅

目 录

[实验一·白盒测试 1](#_Toc73367538)

[一、实验目的 1](#_Toc73367539)

[二、实验内容 1](#_Toc73367540)

[三、实验原理、方法和手段 1](#_Toc73367541)

[1、测试程序 1](#_Toc73367542)

[2、实验要求 1](#_Toc73367543)

[四、实验环境 1](#_Toc73367544)

[五、测试用例、结果及分析 1](#_Toc73367545)

[1、语句覆盖测试 1](#_Toc73367546)

[2、条件覆盖测试 2](#_Toc73367547)

[六、总结与反思 3](#_Toc73367548)

[七、程序源代码 3](#_Toc73367549)

[实验二·黑盒测试 5](#_Toc73367550)

[一、实验目的 5](#_Toc73367551)

[二、实验内容 5](#_Toc73367552)

[三、实验原理、方法和手段 5](#_Toc73367553)

[1、测试程序 5](#_Toc73367554)

[2、实验要求 5](#_Toc73367555)

[四、实验环境 5](#_Toc73367556)

[五、测试用例、结果及分析 5](#_Toc73367557)

[1、等价划分法测试 5](#_Toc73367558)

[2、边界值分析法测试 8](#_Toc73367559)

[六、总结与反思 9](#_Toc73367560)

[七、程序源代码 9](#_Toc73367561)

# 实验一·白盒测试

## 一、实验目的

巩固白盒测试知识，对于给定的待测程序，能熟练应用基本控制流覆盖方法设计测试用例；能够执行白盒测试过程，并撰写白盒测试分析报告。

## 二、实验内容

白盒测试又称为结构测试、逻辑驱动测试或基于程序的测试，一般用来分析程序的内部结构。测试规划基于产品内部结构进行测试，检查内部操作是否按规定执行，软件各个部分功能是否得到充分使用，则这种测试方法称为白盒测试方法。

白盒测试将被测程序看做一个打开的盒子，测试者能够看到被测源程序，可以分析被测程序的内部结构，此时测试的焦点集中在根据其内部结构设计测试用例。白盒测试要求是对某些程序的结构特性做到一定程度的覆盖，或者说这种测试是“基于覆盖率的测试”。

测试内容包括语句覆盖测试、分支覆盖测试、条件覆盖测试、分支/条件覆盖测试、条件组合覆盖测试和基本路径测试。

## 三、实验原理、方法和手段

### 1、测试程序

三角形问题可以更详细地描述为：输入3个整数a、b和c分别作为三角形的三条边，要求a、b和c必须满足以下条件：

Con 1．1≤a≤100

Con 2．1≤b≤100

Con 3．1≤c≤100

Con 4．a<b+c

Con 5．b<a+c

Con 6．c<a+b

### 2、实验要求

至少采用两种语句覆盖技术设计测试用例，包括测试数据及预期结果。

## 四、实验环境

windows 10系统、visual studio 2017

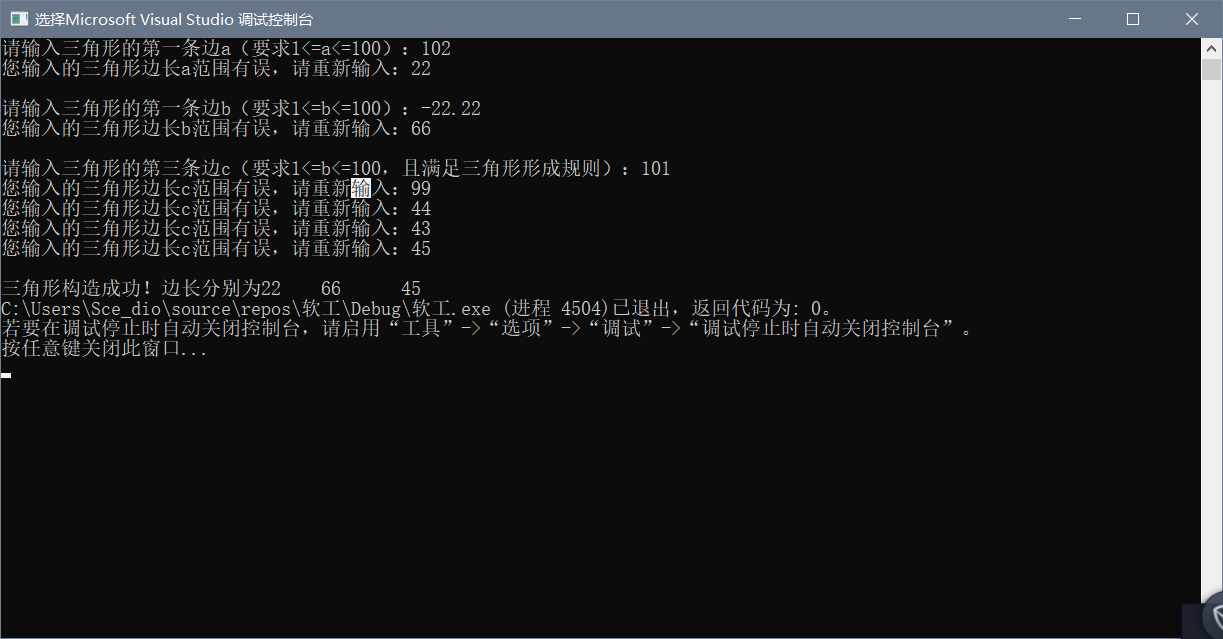
## 五、测试用例、结果及分析

### 1、语句覆盖测试

①测试特点：语句覆盖是最起码的结构覆盖要求，其要求设计足够多的测试用例，使得程序中每条语句至少被执行一次。

②测试用例：a=102,a=22,b=-22.22,b=66,c=101,c=99,c=44,c=43,c=45

③测试结果：



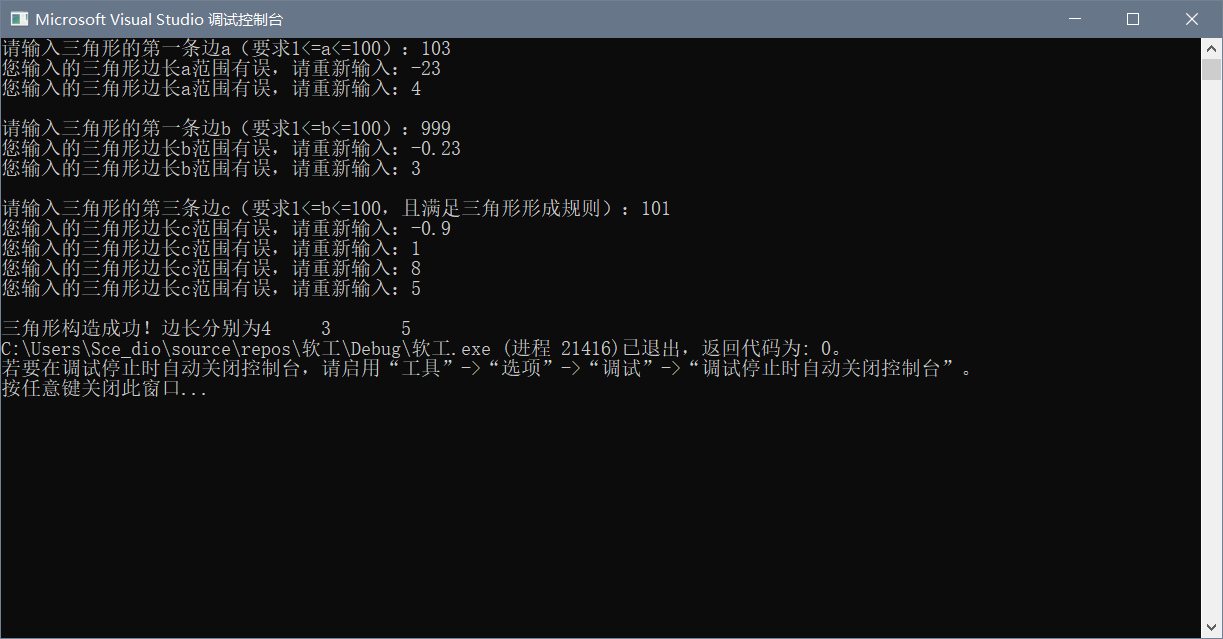
④结果分析：该测试结果符合软件工程白盒测试中语句覆盖测试的思想，测试三条边构成三角形。这三条边需满足6个条件，对于a，我们输入了不符范围的数，其成功返回了错误信息；对于b同理，只是输入的数是负数；对于c，第一次输入不符范围，第二次、第三次则是不满足三角形构成条件（两边和大于第三边）。经此测试案例，程序语句基本执行，达到了我们的测试目的。

### 2、条件覆盖测试

①测试特点：条件覆盖要求设计足够多的测试用例，使得判定中的每个条件获得各种可能的结果，即每个条件至少有一次为真值，有一次为假值。

②测试用例：a=103,a=-23,a=4,b=999,b=-0.23,b=3,c=101,c=-0.9,c=1,c=8,c=5

③测试结果：



④结果分析：本次测试结合了白盒测试中条件覆盖的思想，基本遍历了全部的判定条件，也即基本实现了所有判定条件下三角形边长的取值。可以看到，a、b、c都出现过上下越界的情况，除此之外，c还多出了大于两边之和的情况。至此，我们可以判定，条件覆盖测试成功。

## 六、总结与反思

白盒测试又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。白盒测试是一种测试用例设计方法，盒子指的是被测试的软件。在对软件进行白盒测试时，我们应当对程序内部的结构有着比较清晰的了解，然后按照程序本身的结构去测试，得到我们应得的结果。可以说，白盒测试就是检测系统能否按照自身编写时所预想的那样进行。它只能代表系统可以正常运行，而不能代表系统可以经受得住用户的考验。白盒测试并不能代替黑盒测试。

## 七、程序源代码

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main() {
5. //此为三角形问题程序，以让用户输入符合要求的三角形三条边
6. **double** a, b, c;//定义三角形的三条边
8. cout << "请输入三角形的第一条边a（要求1<=a<=100）：";//提示用户
9. **do** {//直到用户输入正确的边长a，才跳出循环
10. cin >> a;
11. **if** (a < 1 || a > 100) cout << "您输入的三角形边长a范围有误，请重新输入：";
12. } **while** (a < 1 || a > 100);
14. cout << "请输入三角形的第二条边b（要求1<=b<=100）：";//提示用户
15. **do** {//直到用户输入正确的边长b，才跳出循环
16. cin >> b;
17. **if** (b < 1 || b > 100) cout << "您输入的三角形边长b范围有误，请重新输入：";
18. } **while** (b < 1 || b > 100);
20. cout << "请输入三角形的第三条边c（要求1<=b<=100，且满足三角形形成规则）：";//提示用户
21. **do** {//直到用户输入符合条件的边长c，才跳出循环
22. cin >> c;
23. **if** ((c < 1) || (c > 100) || (a >= b + c) || (b >= a + c) || (c >= a + b)) cout << "您输入的三角形边长c范围有误，请重新输入：";
24. } **while** ((c < 1) || (c > 100) || (a >= b + c) || (b >= a + c) || (c >= a + b));
25. }

# 实验二·黑盒测试

## 一、实验目的

巩固黑盒测试知识，黑盒测试能够发现下述错误，功能不正确或遗漏了的功能；界面错误；数据结构错误或外部数据库访问错误；性能错误；初始化和终止错误。能够熟练的运用所学知识测试程序。

## 二、实验内容

测试规划是基于产品的功能，目的是检查程序各个功能是否能够实现，并检查其中的功能错误，这种策试方法称为黑盒测试方法。

黑盒测试又称为功能测试、数据驱动测试和基于规格说明的测试。它是一种从用户观点出发的测试，一般被用来确认软件功能的正确性和可操作性。

黑盒测试的基本观点是：任何程序都可以看作是从输入定义域映射到输出值域的函数过程，被测程序被认为是一个打不开的黑盒子，黑盒中的内容完全不知道，只明确要做到什么。

黑盒测试主要根据规格说明书设计测试用例，并不涉及程序内部构造和内部特性，只依靠被测程序输入和输出之间的关系或程序的功能设计测试用例。

黑盒测试技术包括：等价划分法、边界值分析法、错误推测法、因果图法等。

## 三、实验原理、方法和手段

### 1、测试程序

实现求两整数的加、减、乘、除运算结果，要求两整数的范围都是[0,100]。从键盘输入数m，数n，判断他们的范围，若不在[0,100]范围内，则输出提示语，要求重新输入，并且在做除法运算时，当除数为0时，输出提示语，说明除数不能为0。

### 2、实验要求

要求至少采用两种黑盒测试技术，设计测试用例，包括测试数据及预期结果。

## 四、实验环境

windows 10系统、visual studio 2017

## 五、测试用例、结果及分析

### 1、等价划分法测试

①测试思想：将程序的输入域划分为若干个区域（等价类），并在每个等价类中选择一个具有代表性的元素生成测试用例。该方法是常用的黑盒（Blackbox Testing）测试用例（Testcase）设计方法。

②测试分析：根据程序规格说明，我们将输入数据划分为等价有效类和无效等价类两类，然后给出相应案例，并按相应例子进行测试分析。

|  |  |
| --- | --- |
| 对操作符而言 | |
| 等价有效类 | 无效等价类 |
| +、-、\*、/ | 除+、-、\*、/四大运算符外的一切输入，如=、<=、数字等 |

|  |  |
| --- | --- |
| 对左操作数m而言 | |
| 等价有效类 | 无效等价类 |
| 范围在[0,100]的整数 | 范围不在[0,100]的整数 |
| 浮点数 |
| 字符 |
| … |

|  |  |  |
| --- | --- | --- |
| 对右操作数n而言 | | |
| 情况 | 等价有效类 | 无效等价类 |
| 操作符不为/时 | 范围在[0,100]的整数 | 范围不在[0,100]的整数 |
| 浮点数 |
| 字符 |
| … |
| 操作符为/时 | 范围在(0,100]的整数 | 范围不在(0,100]的整数 |
| 浮点数 |
| 字符 |
| … |

由于有些错误会被其他错误所掩盖，因而我们应尽量一次只覆盖一个无效等价类。

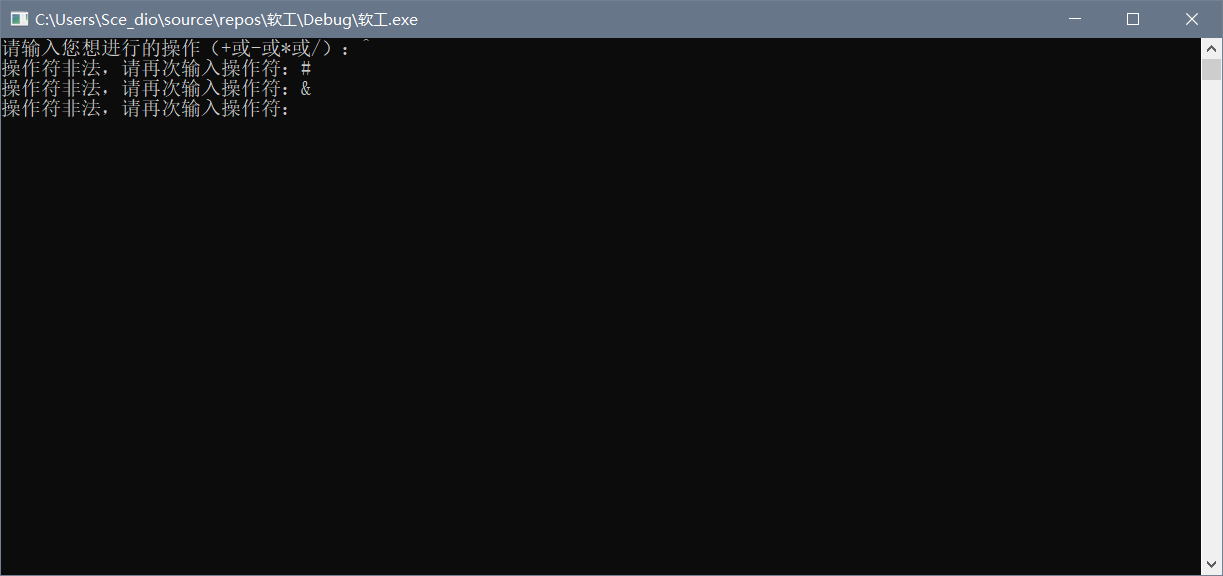
③设计用例：

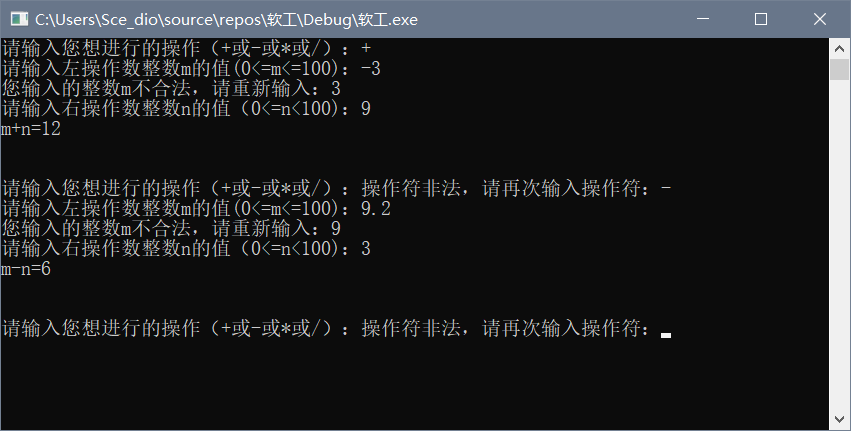
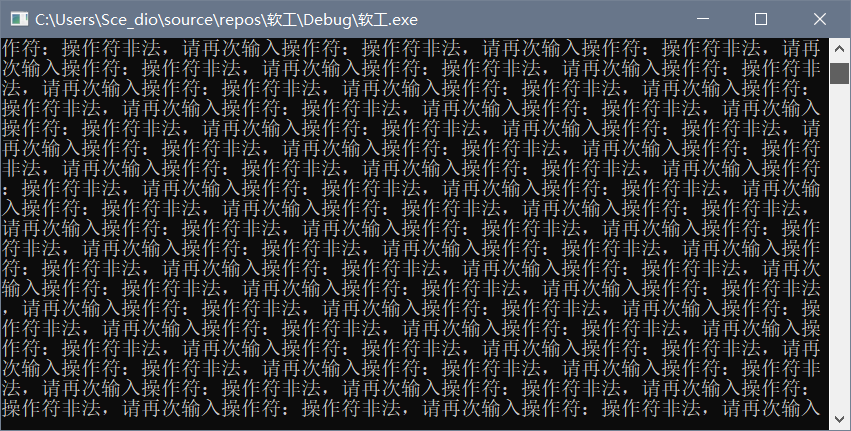
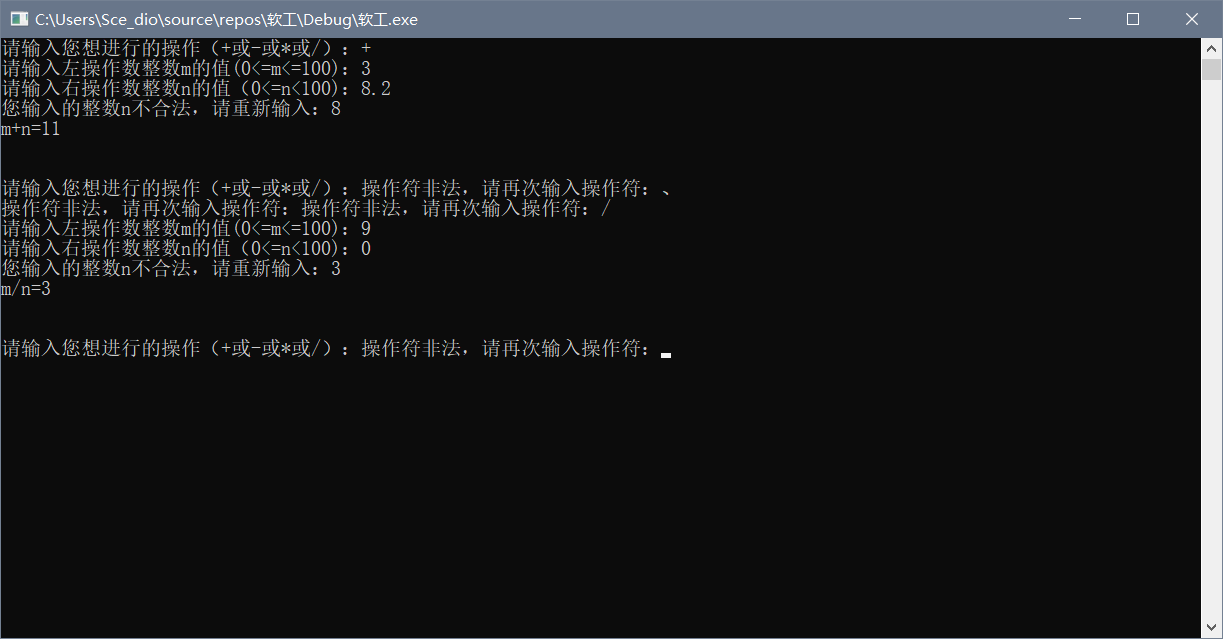
主要针对操作符op的无效输入：9^24、9#24、9&24

主要针对左操作数m的无效输入：-3+9、9.2-3、c\*3

主要针对右操作数n的无效输入： 3+8.2、9/0

④测试结果：





⑤结果分析：我们运用黑盒测试中的等价划分法测试方法，依次对非法的输入数据进行了遍历测试。经过测试，我们发现，虽然该程序的功能是对[0,100]的两个整数进行+、-、\*、/操作，但是该程序对错误输入仍然具有一定的抵抗力。从结果来看，该程序能够正确识别操作符和浮点数、越界数的错误，并做出相应处理。另外，该程序虽然能识别输入的字符，但也只是步入循环提示错误，不能做出正确的相应处理机制。此外，该程序接收到上一次错误时，有时会错误提示多次，由此观之，程序流程、使用语句可能存在一点问题。

### 2、边界值分析法测试

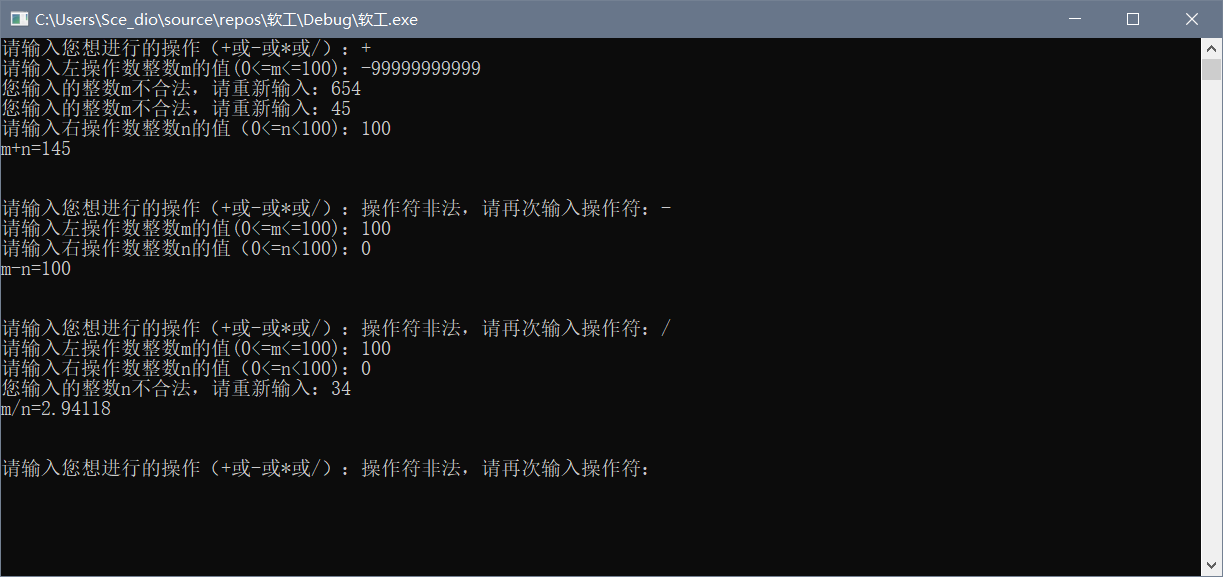
①测试思想：与等价划分法类似，但分析所使用的例子更加极端，基本是处于边界值的数据。

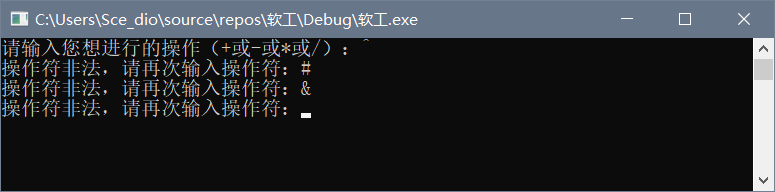
②设计用例：

主要针对操作符op的无效输入：0^100、0#100、0&100

主要针对左操作数m的无效输入：-99999999999+100、100-0、a\*3

主要针对右操作数n的无效输入： 0+100.0、100/0

③测试结果：



④结果分析：我们运用了黑盒测试中的边界值分析法测试方法，依次对非法输入数据的极端情况进行了遍历。经过测试，我们发现，结果与等价划分法一般无二。这是因为该程序结构较简单，输入的数据，行就是行，不行就是不行，不会造成太多混淆。但是，在大型项目开发中，我们应当极其注意极端情况的测试。

## 六、总结与反思

黑盒测试是软件测试中的重要一环，相比于按照程序内部结构进行测试的白盒测试，黑盒测试要更加贴切于大众，更加贴切于实际情况。因为我们开发时很难完全预见到用户所有的输入，又或者是网络情况而导致服务器载荷的变化出现的问题，因此，在系统上线前，对其做黑盒测试是很有必要的。

## 七、程序源代码

1. #include <iostream>
2. **using** **namespace** std;
4. **bool** checkLegle(**char** op,**int** arg) {//判断除法运算时除数是否合法
5. **if** (op == '/'&&arg == 0) **return** **false**;
6. **return** **true**;
7. }
9. **bool** checkLegle(**char** op) {//判断操作符是否合法
10. **if** (op != '+'&&op != '-'&&op != '\*'&&op != '/') **return** **false**;
11. **return** **true**;
12. }
14. **int** main() {
15. **double** m, n,num;
16. **char** op;
17. **do** {
18. cout << "请输入您想进行的操作（+或-或\*或/）：";
19. op = getchar();
20. **while** (!checkLegle(op)) {
21. cout << "操作符非法，请再次输入操作符：";
22. cin >> op;
23. }
24. cout << "请输入左操作数整数m的值(0<=m<=100)：";
25. cin >> m;
26. **while** (m < 0 || m > 100 || m-((**int**)m)!=0) {
27. cout << "您输入的整数m不合法，请重新输入：";
28. cin >> m;
29. }
30. cout << "请输入右操作数整数n的值（0<=n<100)：";
31. cin >> n;
32. **while** ((n < 0 || n>100) || (!checkLegle(op, n))|| n - ((**int**)n) != 0) {//超出范围或除数为0
33. cout << "您输入的整数n不合法，请重新输入：";
34. cin >> n;
35. }
36. **switch** (op) {
37. **case** '+':num = m + n; **break**;
38. **case**'-':num = m - n; **break**;
39. **case**'\*':num = m \* n; **break**;
40. **case** '/':num = m / n; **break**;
41. }
42. cout << "m" << op << "n=" << num;
43. cout << "\n\n\n";
44. } **while** (1 > 0);
45. }