实验四 传统软件工程的软件测试

姓名：高树林

班级：2020185

学号：202018526

实验学时：2（必修）

一、目的与任务

目的：

1、了解软件开发测试的工具。理解软件测试策略和方法，其中包括：
   1）白盒测试
   2）黑盒测试
   3）静态测试
   4）动态测试
2、掌握测试技术

任务：可采用不同的软件测试方法，分析问题，设计测试用例，完成对综合问题的软件测试过程。

二、实验内容

1.软件开发测试工具学习

通过网络搜索，体会软件开发测试工具的应用，重点了解下面几种工具的概况：

1) visual studio

2) Dev-C++

3) VScode

2.软件测试

（1）白盒测试任务：

已知有一段代码

```
int a, b, c;
if ( a < 1 and b > 0 )
c = 5;
else if ( b < -3)
c = 4;
else
c = 3;
```

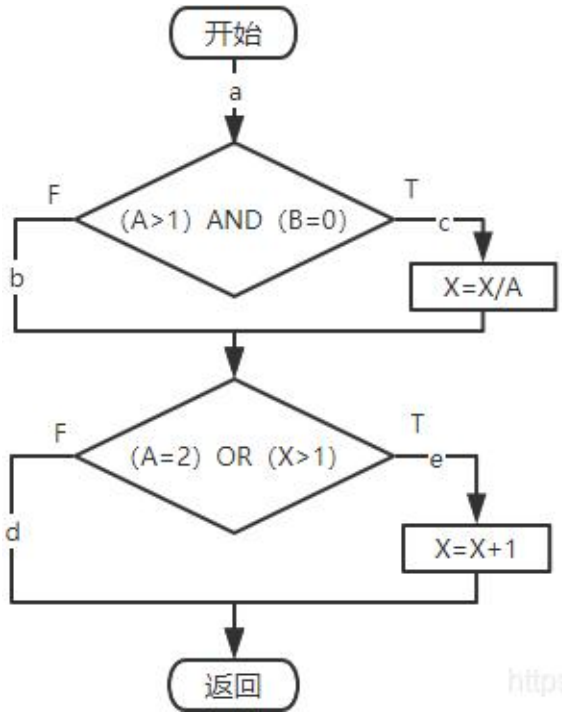
请画出这段代码的程序流程图，并分别采用判定覆盖、条件覆盖、判定条件覆盖、条件组合覆盖和路径覆盖的方法设计测试用例。

（2）黑盒测试任务（三角形问题）：

程序接受 3 个整数 a,b,c 作为输入，用作三角形的 3 条边，程序输出由这三条边确定的三角形类型：等边三角形、等腰三角形、非等边三角形、非三角形。请用分别用等价类划分法和边界值划分法设计测试此程序的测试用例（等价类划分法先画出该问题等价类表并编号，再设计测试用例；边界值划分法需先分析输入、输出域边界值再设计测试用例）。

（3）测试实例

请依据下图的详细设计流程图,设计测试用例找到程序中的所有缺陷。下图是程序员编写的程序，需要测试用例能被执行。



Int A,B;

Double X;

if (A>2 || B=2)

X=X/A;

If (A=3 && X>1)

X=X+1;

三、实验结果：

1.软件工具分析

2 软件总体设计

（1） 白盒测试任务

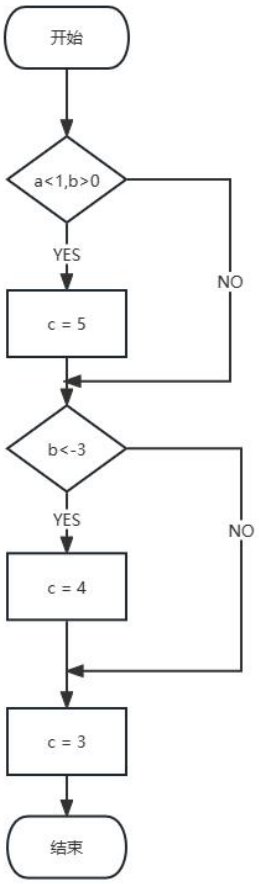


图 1 白盒测试的程序流程图

判定覆盖（设计足够多的测试用例，使程序中的每个判定都至少获得“真值”和“假值”）

假设：(a<1)&&（b>0)为真记为 T1,为假记为 F1；b<-3 为真记为 T2,为假记为 F2

根据判定覆盖的定义，应使两个判定都至少获得“真值”和“假值”。即（a<1）&&（b>0）和 b<-3 都至少获得“真值”和“假值”。

序号	状态	条件	a	b	c
1	F1T2	(a>=1),(b<-3)	1	-4	4
2	T1F2	(a<1),(b>0)	0	1	5

条件覆盖（每一个判定中每个逻辑条件的可能的值至少被满足一次）

假设：（a<1）为真记为 T1,为假记为 F1；（b>0）为真记为 T2，为假记为 F2；（b<-3）为真记为 T3，为假记为 F3。

序号	状态	条件	a	b	c
1	T1T2F3	(a<1),(b>0),(b>=-3)	0	2	5

2	F1F2T3	(a>=1),(b<-3),(b<0)	1	-4	4

判定条件覆盖（判定中每个条件所有可能至少出现一次，判定本身的判定结果业至少出现一次）

假设：（a<1）为真记为 T1,为假记为 F1；（b>0）为真记为 T2，为假记为 F2；（b<-3）为真记为 T3，为假记为 F3。

序号	状态	条件	a	b	c
1	T1T2F3	(a<1),(b>0),(b>=-3)	0	2	5
2	F1F2T3	(a>=1),(b<-3),(b<0)	1	-4	4

条件组合覆盖（每个判定中条件的各种可能组合都至少出现一次）

假设：（a<1）为真记为 T1,为假记为 F1；（b>0）为真记为 T2，为假记为 F2；（b<-3）为真记为 T3，为假记为 F3。

序号	状态	条件	a	b	c
1	T1T2F3	(a<1),(b>0),(b>=-3)	0	2	5
2	T1F2T3	(a<1),(b<0),(b<-3)	0	-4	3
3	F1T2T3	(a>=1),(b>0),(b<-3)	2	不存在	无
4	T1F2F3	(a<1),(b>=0),(b>=-3)	0	-2	3
5	F1T2F3	(a>=1) ,(b>0),(b>=-3)	2	2	3
6	F1F2T3	(a>=1),(b<-3),(b<0)	2	-4	4
7	F1F2F3	(a>=1) ,(b>=0),(b>=-3)	2	-2	3
8	T1T2T3	(a<1), (b>=0),(b>=-3)	0	不存在	无

路径覆盖（每个路径都可能被执行）

假设：（a<1）为真记为 T1,为假记为 F1；（b>0）为真记为 T2，为假记为 F2；（b<-3）为真记为 T3，为假记为 F3。

序号	状态	条件	a	b	c
1	T1T2F3	(a<1),(b>0),(b>=-3)	0	2	5
2	F1F2T3	(a>=1),(b<-3),(b<0)	1	-4	4

（2） 黑盒测试任务

1. 划分有效等价类和无效等价类

输入条件	有效等价类	无效等价类
是否能构成三角形的三条边	a>0 (1)	a<=0 (7)
	b>0 (2)	b<=0 (8)
	c>0 (3)	c<=0 (9)
	a+b>0 (4)	a+b<=c (10)
	b+c>a (5)	b+c<=a (11)
	c+a>b (6)	c+a<=b (12)
是否等腰三角形	a=b (13)	a!=b && b!=c && c!=a (16)
	b=c (14)	
	c=a (15)	
是否等边三角形	a=b && b=c && c=a (17)	a!=b (18)
		b!=c (19)
		c!=a (20)

2. 为有效等价类设计测试用例

测试用例（a ,b ,c）	预期输出	覆盖范围
3、4、5	一般三角形	（1）、（2）、（3）、（4）、（5）、（6）
3、3、4	等腰三角形	（1）、（2）、（3）、（4）、（5）、（6）、（13）
3、4、4		（1）、（2）、（3）、（4）、（5）、（6）、（14）
3、4、3		（1）、（2）、（3）、（4）、（5）、（6）、（15）
3、3、3	等边三角形	（1）、（2）、（3）、（4）、（5）、（6）、（17）

3. 为无效等价类设计测试用例

测试用例（a ,b ,c）	预期输出	覆盖范围
0、1、2	不构成三角形	（7）
1、0、2		（8）
1、2、0		（9）
1、2、3		（10）
3、2、1		（11）
2、3、1		（12）
3、4、5	非等腰三角形	（16）
3、4、4	非等边三角形	（18）
3、4、3		（19）
3、3、4		（20）

用边界值测试方法设计测试用例

1. 分析各变量取值

边界值分析的基本思想是使用输入变量的最小值、略高于最小值、正常值、略 低于最大值和最大值设计测试用例。因此 a, b, c 的边界取值是: 1, 2，100,199，200

2. 测试用例数

有 n 个变量的程序，其边界值分析会产生 4n+1 个测试用例。这里有 3 个变 量，因此会产生 13 个测试用例。

3. 设计测试用例

用边界值分析法设计测试用例就是使一个变量取边界值(分别取最小值、略 高于最小值、正常值、略低于最大值和最大值)，其余变量取正常值，然后对每 个变量重复进行。本例用边界值分析法设计的测试用例如下

测试用例	输入数据			期望输出
	a	b	c	
1	100	100	1	等腰三角形
2	100	100	2	等腰三角形
3	100	100	100	等边三角形
4	100	100	199	等腰三角形
5	100	100	200	非三角形
6	100	1	100	等腰三角形
7	100	2	100	等腰三角形
8	100	199	100	等腰三角形
9	100	200	100	非三角形
10	1	100	100	等腰三角形
11	2	100	100	等腰三角形
12	199	100	100	等腰三角形
13	200	100	100	非三角形

（3） 测试实例

假设将 A>1 为真时记为 T1，则当 A<=1 时记为 F1;将 B=0 真记为 T2，B≠0 为假记为 F2;将 A=2 为真记为 T3，A≠2 记为 F3;将 X>1 为真记为 T4，X<=1 为 F4。则上述 8 中情况排列共会出现 16 种结果，列出表格如下所示：

序号	标志	A	B	X	程序框图结果	程序结果	是否正确
1	T1T2T3T4	2	0	2	2	2	正确
2	T1T2T3F4	2	0	1	1.5	1.0	错误
3	T1T2F3T4	3	0	2	2/3	2/3	正确
4	T1F2T3T4	2	1	2	3	2	错误
5	F1T2T3T4	无	0	2	\	\	\
6	F1F2T3T4	无	1	2	\	\	\
7	F1T2F3T4	-1	0	2	3	2	错误
8	F1T2T3F4	无	0	1	\	\	\
9	T1F2F3T4	3	1	2	3	2/3	错误
10	T1F2T3F4	2	1	1	2	1	错误
11	T1T2F3F4	3	0	1	1/3	1/3	正确
12	F1F2F3T4	1	1	2	3	2	错误
13	F1F2T3F4	无	1	1	\	\	\
14	F1T2F3F4	1	0	1	1	1	正确
15	T1F2F3F4	3	1	1	1	1/3	错误
16	F1F2F3F4	1	1	1	1	1	正确

上述结果是利用代码自动运行实现的，代码如下：

```
A, B = map(int, input().split(" "))
X = float(input())
def chengxu(A, B, X):
    if A > 2 or B == 2:
        X = X / A
    if A == 3 and X > 1:
        X = X + 1
    return X
def tu(A, B, X):
    if A > 1 and B == 0:
        X = X / A
    if A == 2 or X > 1:
        X += 1
    return X
print("图结果：", tu(A, B, X), end=" ")
print("代码结果：", chengxu(A, B, X))
```

根据以上结果分析，该程序的正确率仅为 31.25%，准确率显著不足，无法满足大多数样例的要求，因此无法通过测试。该结果表明该程序在处理样例时存在严重的不合格问题，需要进一步改进和优化，以提高准确性并满足实际需求。