实验二黑盒测试

**2.1** 实验类型

实验类型为验证型，4 个学时。

# **2.2** 实验目的

1. 能熟练应用黑盒测试技术进行测试用例设计；
2. 对测试用例进行优化设计；

**2.3** 实验环境

Windows 环境,Word 和 Visio 或者相关的办公软件, C/C++或 Java 编程环境。

# **2.4** 实验报告

程序 1 和程序 3 应用黑盒测试方法，设计测试用例（通过表格的方式），运行测试结果并分析，记录到实验报告中。

实验报告命名：实验二黑盒测试 +学号姓名.pdf。

程序 2 在实验任务书中填写即可。

# **2.5** 实验内容

1．程序一：三角形问题。

根据下面给出的规格说明，利用等价类划分的方法，导出足够的测试用例。输入三个整数 a、b、c 分别作为三角形的三条边，现通过程序判断由三条边构成的三角形的类型为等边三角形、等腰三角形、一般三角形以及构不成三角形。现在要求输入三个整数 a、b、c，必须满足以下条件：

条件 1 1≤a≤100 条件 4 a<b+c 条件 2 1≤b≤100 条件 5 b<a+c 条件 3 1≤c≤100 条件 6 c<a+b

2．程序二：能够求出三个在-10000 到 10000 间整数中的最大者。

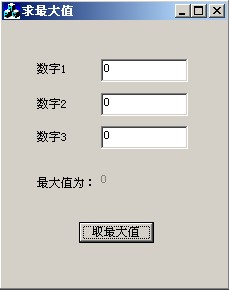


图 1 程序运行界面

3．题目三：日期问题。

用因果图法设计测试用例测试程序:该程序有三个输入变量 month、day、year （month 、 day 和 year 均为整数值，并且满足：1≤month≤12 和 1≤day≤31),分别作为输入日期的月份、日、年份，通过程序可以输出该输入日期在日历上隔一天的日期。例如，输入为 2004 年 11 月 29 日，则该程序的输出为 2004 年 12 月

1 日。

1. 分析各种输入情况，列出为输入变量 month 、 day 、 year 划分的有效等价类。
2. 分析程序的规格说明，并结合以上等价类划分的情况，给出问题规定的可能采取的操作。
3. 根据 (1) 和 (2) ，画出简化后的决策表，并导出测试用例。

4．题目四：针对不同月薪需要缴纳不同的个人所得税计算程序，结合边界值分析法和等价类划分方法，设计充分的测试用例。

表1 个税计算表

|  |  |
| --- | --- |
| 应纳税所得额**(**减去起征点 **2000** 元后的结果**)** | 税率**/%** |
| 不超过 500 元 | 5 |
| 超过 500 元至 2000 元 | 10 |
| 超过 2000 元至 5000 元 | 15 |
| 超过 5000 元至 20000 元 | 20 |
| 超过 20000 元至 40000 元 | 25 |
| 超过 40000 元至 60000 元 | 30 |
| 超过 60000 元至 80000 元 | 35 |
| 超过 80000 元至 100000 元 | 40 |
| 超过 100000 元 | 45 |

# **2.6** 实验步骤

1. 根据黑盒测试技术设计测试用例，主要考虑等价类划分和边界值分析测试技术；
2. 根据所学知识确定优化策略（原则：用最少的测试用例检测出更多的缺陷、软件测试的充分性与冗余性考虑），设计两套测试用例集； （3） 根据设计的两套测试用例集进行测试；

理论课上介绍了几种测试用例的设计方法。在实际的应用过程中，有时这些方法的边界并不是十分的清晰，例如一个测试用例可以算做是等价类划分法，也可以算做是边界值划分法。因此，我们在编写测试用例时不必拘泥于严格的区分每个用类的类型，而是要设计出能够实现测试目标的测试用例。

下面我们就结合理论课中介绍的设计黑盒测试用例的方法，设计测试用例。

## **2.6.1** 程序一：三角形问题

根据程序清单，给出等价类表和测试用例表。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 程序清单：

#include <iostream.h> void main(void)

{

int a,b,c;

cout<<"请输入 3 个整数："<<endl;

cin>>a>>b>>c; if(!(a+b<=c||a+c<=b|| b+c<=a)&&(a>=1&&a<=100)&&(b>=1&&b<=100)&&(c>=1&&c<=100))

{

if (a==b && a==c) cout<<"等边三角形"<<endl; else if (a==b || a==c || b==c) cout<<"等腰三角形"<<endl;

else cout<<"一般三角形"<<endl;

}

else

{

if(a<1||a>100) cout<<"a="<<a<<"不满足条件 1,不能构成三角形"<<endl; else if(b<1||b>100) cout<<"b="<<b<<"不满足条件 2,不能构成三角形"<<endl; else if(c<1||c>100) cout<<"c="<<c<<"不满足条件 3,不能构成三角形"<<endl; else if(a>=b+c) cout<<"不满足 a<b+c,不能构成三角形"<<endl; else if(b>=a+c) cout<<"不满足 b<a+c,不能构成三角形"<<endl; else if(c>=b+a) cout<<"不满足 c<a+b,不能构成三角形"<<endl;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

表2 等价类表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case** | 有效 | 编号 | 无效 | 编号 |
| 条件 1 | 1≤a≤100 | 1 | a<1 | 7 |
| a>100 | 8 |
| 条件 2 | 1≤b≤100 | 2 | b<1 | 9 |
| b>100 | 10 |
| 条件 3 | 1≤c≤100 | 3 | c<1 | 11 |
| c>100 | 12 |
| 条件 4 | a≤b+c | 4 | a≥b+c | 13 |
| 条件 5 | b≤a+c | 5 | b≥a+c | 14 |
| 条件 6 | c≤a+b | 6 | c≥a+b | 15 |

表3 测试用例表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Testcase** | **a** | **b** | **c** | 预期输出 | 输出结果 | 测试是否成功 | 失败原因 |
| 1 | 1 | 1 | 1 | 等边三角形 | 等边三角形 | √ |  |
| 2 | 2 | 3 | 2 | 等腰三角形 | 等腰三角形 | √ |  |
| 3 | 3 | 4 | 5 | 一般三角形 | 一般三角形 | √ |  |
| 4 | -1 | 2 | 2 | a= -1 不满足条件 1，不能够成三角形 | a= -1 不满足条件 1，不能够成三角形 | √ |  |
| 5 | 101 | 2 | 2 | a= 101 不满足条件 1，不能够成三角形 | a= 101 不满足条件 1，不能够成三角形 | √ |  |
| 6 | 2 | -1 | 2 | b= -1 不满足条件 2，不能够成三角形 | b= -1 不满足条件 2，不能够成三角形 | √ |  |
| 7 | 2 | 101 | 2 | b= 101 不满足条件 2，不能够成三角形 | b= 101 不满足条件 2，不能够成三角形 | √ |  |
| 8 | 2 | 2 | -1 | c= -1 不满足条件 3，不能够成三角形 | c= -1 不满足条件 3，不能够成三角形 | √ |  |
| 9 | 2 | 2 | 101 | c= 101 不满足条件 3，不能够成三角形 | c= 101 不满足条件 3，不能够成三角形 | √ |  |
| 10 | 3 | 1 | 2 | 不满足 a<b+c，不能构成三角形 | 不满足 a<b+c，不能构成三角形 | √ |  |
| 11 | 1 | 3 | 2 | 不满足 b<a+c，不能构成三角形 | 不满足 b<a+c，不能构成三角形 | √ |  |
| 12 | 1 | 2 | 3 | 不满足 c<a+b，不能构成三角形 | 不满足 c<a+b，不能构成三角形 | √ |  |

表 3 给出了一组基于等价类划分的测试用例，请你根据预期输出，构造测试的数据，输入的数据应用有效效等价类和无效等价类的知识选择，注意，该类型是数据仅仅是数值型的，不足以检查程序中存在的问题。因此，请你根据学习的黑盒测试方法知识，应用等价类、边界值等方法构造一组新的测试用例。如果能够找到 bug，请分析 bug 产生的原因，并给出修改意见。

|  |  |  |
| --- | --- | --- |
| 输入条件 | 有效等价类 | 无效等价类 |
| 是否构成三角形 | 1<=a<=100 (1) | a<=0 (7) |
| 1<=b<=100 (2) | b<=0 (8) |
| 1<=c<=100 (3) | c<=0 (9) |
| c<a+b (4) | a>100 (21) |
| b<a+c (5) | b>100 (22) |
| a<b+c (6) | c>100 (23) |
| a+b<=c (10) |
| a+c<=b (11) |
| b+c<=a (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) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试用例表 | |  |  |  |  |  |
| 序号 | [a,b,c] | 覆盖等价类 | 预期输出 | 输出结果 | 测试是否通过 | 未通过原因 |
| 1 | [3,4,5] | (1)(2)(3)(4)(5)(6) | 一般三角形 | 一般三角形 | √ |  |
| 2 | [0,1,2] | 7 | 不满足条件 1，不能够成三角形 | 不满足条件 1，不能够成三角形 | √ |  |
| 3 | [1,0,2] | 8 | 不满足条件 2，不能够成三角形 | 不满足条件 2，不能够成三角形 | √ |  |
| 4 | [1,2,0] | 9 | 不满足条件 3，不能够成三角形 | 不满足条件 3，不能够成三角形 | √ |  |
| 5 | [1,2,3] | 10 | 不满足 c<a+b，不能构成三角形 | 不满足 c<a+b，不能构成三角形 | √ |  |
| 6 | [1,3,2] | 11 | 不满足 b<a+c，不能构成三角形 | 不满足 b<a+c，不能构成三角形 | √ |  |
| 7 | [3,1,2] | 12 | 不满足 a<b+c，不能构成三角形 | 不满足 a<b+c，不能构成三角形 | √ |  |
| 8 | [3,3,2] | (1)(2)(3)(4)(5)(6)(13) | 等腰三角形 | 等腰三角形 | √ |  |
| 9 | [2,3,3] | (1)(2)(3)(4)(5)(6)(14) | 等腰三角形 | 等腰三角形 | √ |  |
| 10 | [3,2,3] | (1)(2)(3)(4)(5)(6)(15) | 等腰三角形 | 等腰三角形 | √ |  |
| 11 | [3,4,5] | (1)(2)(3)(4)(5)(6)(16) | 一般三角形 | 一般三角形 | √ |  |
| 12 | [5,5,5] | (1)(2)(3)(4)(5)(6)(17) | 等边三角形 | 等边三角形 | √ |  |
| 13 | [4,5,5] | (1)(2)(3)(4)(5)(6)(14)(18) | 等腰三角形 | 等腰三角形 | √ |  |
| 14 | [5,4,5] | (1)(2)(3)(4)(5)(6)(15)(19) | 等腰三角形 | 等腰三角形 | √ |  |
| 15 | [5,5,4] | (1)(2)(3)(4)(5)(6)(14)(20) | 等腰三角形 | 等腰三角形 | √ |  |
| 16 | [101,1,2] | 21 | 不满足条件 1，不能够成三角形 | 不满足条件 1，不能够成三角形 | √ |  |
| 17 | [1,101,2] | 22 | 不满足条件 2，不能够成三角形 | 不满足条件 2，不能够成三角形 | √ |  |
| 18 | [1,2,101] | 23 | 不满足条件 3，不能够成三角形 | 不满足条件 3，不能够成三角形 | √ |  |

## **2.6.2** 求三个数中最大数问题

现在以图 1 中的小程序设计黑盒测试用例。所有测试用例的测试数据请同学们根据“应产生的行为”自行构造。

**1** 单个文本框的测试用例设计一、数值等价类对每个文本框而言，输入值的限制是在-10000 到 10000 之间，因此，我们可以划分一个有效等价类和二个无效等价类：

-10000 0 10000

* <-10000
* -10000 到 10000
* >+10000

根据这三个等价类，设计如下的测试用例：

表 4 数值等价类用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例序号 | 测试用例 | 应产生行为 | 预期输出 | 输出结果 | 结果 | 失败原因 |
| MAX001 | 50，50，50 | 程序必须能接受输入并运行正常 | 50 | 50 | √ |  |
| MAX002 | -50，-50，-50 | 程序必须能接受输入并运行正常 | -50 | -50 | √ |  |
| MAX003 | 0，0，0 | 程序必须能接受输入并运行正常 | 0 | 0 | √ |  |
| MAX004 | -10001 | 程序必须能判断输入的数越界并能告知用户 | 提示错误 | 请输入-10000至10000之间的整数 | √ |  |
| MAX005 | 10001 | 程序必须能判断输入的数越界并能告知用户 | 提示错误 | 请输入-10000至10000之间的整数 | √ |  |

1. 数据类型等价类

1、由于在文本框中只能输入整数，因此我们可以确定：有效等价类：数字

无效等价类：字母、小数点、控制字符、功能键根据上述分析，确定如下测试用例

表 5 数据类型等价类测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例序号 | 测试用例 | 应产生行为 | 预期输出 | 输出结果 | 测试 结果 | 失败原因 |
| MAX006 | 输入 9 ，0，0 | 程序必须能接受输入并运行正常 | 9 | 9 | √ |  |
| MAX007 | 输入-数 ，0，0 | 运行不正常 | 提示应输入一个整数 | 请输入一个整数 | √ |  |
| MAX008 | 输入+数 ，0，0 | 接收输入，运行不正常 | 提示未输入整数 | 请输入一个整数 | √ |  |
| MAX009 | 输入TAB键，0，0 | 不能接收输入，出现错误提示 | 提示未输入整数 | 请输入一个整数 | √ |  |
| MAX010 | 输入空格键，0，0 | 不能接收输入，出现错误提示 | 提示未输入整数 | 请输入一个整数 | √ |  |
| MAX011 | 输入A，0，0 | 运行不正常 | 提示未输入整数 | 请输入一个整数 | √ |  |
| MAX012 | 输入a，0，0 | 运行不正常 | 提示未输入整数 | 请输入一个整数 | √ |  |
| MAX013 | 输入0.9，0，0 | 运行不正常 | 提示未输入整数 | 请输入一个整数 | × | 将0.9看作了0 |
| MAX014 | 输入-0.9，0，0 | 运行不正常 | 提示未输入整数 | 请输入一个整数 | × | 将-0.9看作了0 |
| MAX015 | 输入，键，0，0 | 运行不正常 | 提示未输入整数 | 请输入一个整数 | √ |  |
| MAX016 |  |  |  |  |  |  |
| MAX017 |  |  |  |  |  |  |
| MAX018 |  |  |  |  |  |  |
| MAX019 |  |  |  |  |  |  |
| MAX020 |  |  |  |  |  |  |
| MAX020 |  |  |  |  |  |  |

1. 其他

除了上面列出的测试用例以外，为了使程序的性能更稳定、良好，我们还要设计如下的测试用例：

表 6 测试用例表三

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例序号 | 测试用例 | 应产生行为 | 预期输出 | 输出结果 | 测试结果 | 失败原因 |
| MAX021 | 在输入一个数字后，等待很长时间后再输入下一个数字 9，0，0 | 测试的超时控制能否正常工作 | 9 | 9 | √ |  |
| MAX022 | 程序不关闭，睡眠之后再测试9，0，0 | 能正常工作 | 9 | 9 | √ |  |
| MAX023 | 输入1 00，0，0 | 不能正常工作 | 100 | 1 | × | 1 00为1 |
| MAX024 | 输入0，1 99，0 | 不能正常工作 | 1 | 1 | √ |  |
| MAX025 |  |  |  |  |  |  |
| MAX026 |  |  |  |  |  |  |
| MAX027 |  |  |  |  |  |  |
| MAX028 |  |  |  |  |  |  |
| MAX029 |  |  |  |  |  |  |

1. 边界值

要测试的程序有两个边界值，-10000 和+10000，同时，按照经验，对于 0 和位数升级的数值（例如，从 99 到 100，从 999 到 1000 等）也要做一个边界值来进行测试。因此，我们可以设计出如下的测试用例：

表 7 边界值测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例序号 | 测试用例 | 应产生行为 | 预期输出 | 输出结果 | 测试结果 | 失败原因 |
| MAX030 | 输入-10000 ，0，0 | 程序必须能接受输入并运行正常 | 0 | 0 | √ |  |
| MAX031 | 输入-10001 ，0，0 | 程序必须能检查用户的输入是否合理并给出提示 | 提示错误信息 | 请输入一个-10000至10000之间的整数 | √ |  |
| MAX032 | 输入-9999，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX033 | 输入-999，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX034 | 输入-99，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX035 | 输入-9，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX036 | 输入-10，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX037 | 输入-100，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX038 | 输入-1000，0，0 | 正常输出结果 | 0 | 0 | √ |  |
| MAX038 | 输入9，0，0 | 正常输出结果 | 9 | 9 | √ |  |
| MAX038 | 输入10，0，0 | 正常输出结果 | 10 | 10 | √ |  |
| MAX039 | 输入99，0，0 | 正常输出结果 | 99 | 99 | √ |  |
| MAX040 | 输入999，0，0 | 正常输出结果 | 999 | 999 | √ |  |
| MAX041 | 输入1000，0，0 | 正常输出结果 | 1000 | 1000 | √ |  |
| MAX042 | 输入9999，0，0 | 正常输出结果 | 9999 | 9999 | √ |  |
| MAX043 | 输入100，0， | 正常输出结果 | 100 | 100 | √ |  |

**2** 程序功能的测试用例设计一、等价类

两个数值的大小有三种情况，大于、等于或小于，现在我们要对三个数进行比较，因此可以划分出如下的等价类：

表 8 等价类划分表

|  |  |  |
| --- | --- | --- |
| A>B | B>C |  |
| B=C |  |
| B<C | A>C |
| A<C |
| A=B | B>C |  |
| B=C |  |
| B<C |  |
| A<B | B>C |  |
| B=C |  |
| B<C |  |

根据上面的等价类划分，我们可以设计出如下的测试用例：

表 9 程序功能等价类用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例序号 | 测试用例 | 应产生行为 | 输出结果 | 测试结果 | 失败原因 |
| MAX042 | 输入 3 2 1 | 显示最大数是 3 | 3 | √ |  |
| MAX043 | 输入 3 2 2 | 显示最大数是 3 | 3 | √ |  |
| MAX044 | 输入4，2，3 | 显示最大数是 4 | 4 | √ |  |
| MAX045 | 输入3，2，4 | 显示最大数是 4 | 4 | √ |  |
| MAX046 | 输入3，3，2 | 显示最大数是 3 | 3 | √ |  |
| MAX047 | 输入3，3，3 | 显示最大数是 3 | 3 | √ |  |
| MAX048 | 输入3，3，4 | 显示最大数是 4 | 4 | √ |  |
| MAX049 | 输入3，2，1， | 显示最大数是 3 | 3 | √ |  |
| MAX050 | 输入3，2，2 | 显示最大数是 3 | 3 | √ |  |
| MAX051 | 输入3，2，3 | 显示最大数是 3 | 3 | √ |  |

二、其它

我们还可以测试其它的一些测试用例，如下表所示：表 10 其它测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例序号 | 测试用例 | 应产生行为 | 输出结果 | 结果 | 失败原因 |
| MAX052 | 使用 Tab 键 | 光标可在文本框间顺序移动 | 光标在文本框间顺序移动 | √ |  |
| MAX053 | 使用上下键 | 使用上下键可使光标移动 | 不能移动 | × | 上下键不能使光标移动 |
| MAX054 | 使用左右键 | 使用左右键不能使光标在文本框间移动 | 使用左右键不能使光标在文本框间移动 | √ |  |

# **2.6.3**

（1）month的有效等价类：M1{month=1，3，5，7，8，10}

M2 {month=2} M3 {month=4，6，9，11} M4 {month=12}

day的有效等价类： D1 {1<=day<=26}

D2 {day=27} D3 {day=28} D4 {day=29}

D5 {day=30} D6 {day=31}

Year的有效等价类：

Y1 {year是闰年} Y2 {year不是闰年}

(2)根据各种有效的输入情况，可能的动作有：

A1 {day+2} A2 {day=2}

A3 {day=1} A4 {month+1}

A5 {month=1} A6{year+1}

A7 {不可能}

1. 决策表





1. 测试用例表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 测试用例序号 | month | day | year | 预期输出 | 输出结果 | 测试是否成功 | 未成功原因 |
| 1 | 3 | 29 | 2012 | 2012\3\31 | 2012\3\31 | √ |  |
| 2 | 3 | 30 | 2012 | 2012\4\1 | 2012\4\1 | √ |  |
| 3 | 3 | 31 | 2012 | 2012\4\2 | 2012\4\2 | √ |  |
| 4 | 2 | 26 | 2010 | 2010\2\28 | 2010\2\28 | √ |  |
| 5 | 2 | 27 | 2008 | 2008\2\29 | 2008\2\29 | √ |  |
| 6 | 2 | 27 | 2010 | 2010\3\1 | 2010\3\1 | √ |  |
| 7 | 2 | 28 | 2008 | 2008\3\1 | 2008\3\1 | √ |  |
| 8 | 2 | 28 | 2010 | 2010\3\2 | 2010\3\2 | √ |  |
| 9 | 2 | 29 | 2008 | 2008\3\2 | 2008\3\2 | √ |  |
| 10 | 2 | 29 | 2010 | 提示出错 | 提示出错 | √ |  |
| 11 | 2 | 30 | 2010 | 提示出错 | 提示出错 | √ |  |
| 12 | 6 | 24 | 2012 | 2012\6\26 | 2012\6\26 | √ |  |
| 13 | 6 | 29 | 2012 | 2012\7\1 | 2012\7\1 | √ |  |
| 14 | 6 | 30 | 2012 | 2012\7\2 | 2012\7\2 | √ |  |
| 15 | 6 | 31 | 2012 | 提示不可能 | 提示不可能 | √ |  |
| 16 | 12 | 29 | 2011 | 2011\12\31 | 2011\12\31 | √ |  |
| 17 | 12 | 30 | 2011 | 2012\1\1 | 2012\1\1 | √ |  |
| 18 | 12 | 31 | 2011 | 2012\1\2 | 2012\1\2 | √ |  |

# **2.7** 实验要求

1. 运行题目一中的测试用例，并应用所学的黑盒测试方法构造一套新的测试用例，并运行，如果发现 bug，需要说明 bug 产生的原因，并给出修改意见。
2. 运行题目二中的测试用例，任意选择两套具有代表性的测试用例，如果发现 bug，需要说明 bug 产生的原因，并给出修改意见。
3. 根据题目 3 或题目 4 的需求编写测试用例，然后编写代码，运行测试用例，给出测试结果。
4. 实验结果要求给一个题目中两套不同的测试用例集的测试效果比较，并能对测试用例中发现的 bug，给出原因分析和修改意见。
5. 撰写实验报告。

# **2.8** 实验思考题

1. 在实际的测试中，如何设计测试用例才能达到用最少的测试用例检测出最多的缺陷？
2. 在进行用例设计时，如何考虑软件测试用例的充分性和减少软件测试用例的冗余性？
3. 创建二叉树，遍历二叉树。在二叉树做任何运算之前，二叉树本身必须存在。因此，首先必须创建二叉树，实际上就是建立二叉树的存储结构。建立二叉树的存储结构就是建立二叉链表。下面介绍一种按完全二叉树的层次顺序，依次输入结点的信息建成立二叉链表的算法。该算法的基本思想是：首先对一般的二叉树添加若干个虚结点，使其成为完全二叉树，然后依次输入结点信息。若输入的结不是虚结点，则建立一个新结点；若是第一个令其为根结点；否则将新结点插入到双亲结点上。如此重复下去，直到输入信息“@”为止。

为了使新结点能正确链接到其双亲结点上，可设置一个指针数组作为队列，保存已输入的结点的地址。因为按层自左至右输入结点的，所以首先输入结点的孩子先进队列，因此利用队列的队头指针 front 指向当前结点的双亲结点，利用队尾指针 rear 指向当前结点。若 rear 为偶数，则说明当前结点应作为左孩子链接到 front 所指向的结点上；否则，当前结点应作为右孩子链接到 front 所指向的结点上。若当前结点为虚结点则不需要链接。之后，使队头指针 front 指向下一个双亲结点。具体实现算法：tree.c。

根据要求，在白盒测试方法的基础上，应用黑盒测试方法，选择测试输入，测试 tree.c，找出 bug。