

**本科实验报告**

课程名称： 软件测试技术

实验项目： 软件测试技术实验

实验地点： C1实验室

专业班级： 软件1711 学号： 2017006040

学生姓名： 李亚飞

指导教师： 宋晓涛

2019年 10 月17日

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | **实验一 白盒测试方法** | | |
| **实验地点** | C1实验室 | **实验时间** | 2019.10.17 |
| **一、实验目的和要求** | | | |
| 1. 熟练掌握白盒测试方法中的逻辑覆盖和路径测试覆盖方法。   (2)通过实验掌握逻辑覆盖测试的测试用例设计，掌握程序流图的绘制。  (3)运用所学理论，完成实验研究的基本训练过程 | | | |
| **二、实验内容和原理** | | | |
| 实验内容：测试以下程序段  void DoWork(int x,int y,int z)  {  1 int k=0,j=0;  2 if((x>0)&&(z<10))  3 {  4 k=x\*y-1;  5 j=sqrt(k);  6 }  7 if((x==4)||(y>5))  8 j=x\*y+10;  9 j=j%3;  10 }  说明：程序段中每行开头的数字（1-10）是对每句语句的编号。 | | | |
| **三、主要仪器设备** | 笔记本电脑 | | |
| **四、操作方法与实验步骤** | | | |
| 1. 画出程序的控制流图（用题中给出的语句编号表示）      1. 分别以语句覆盖、判定覆盖、条件覆盖和基本路径覆盖法设计测试用例，并写出每个测试用例的执行路径（用题中给出的语句编号表示）  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 用例编号 | 输入 | 期望结果 | 覆盖标准 | 覆盖路径 | 实际结果 | | 1 | X=1,z=10,y=1 | 2 | 语句覆盖 | 12345678910 | 2 | | 2 | X=1,z=10,y=1 | 2 | 判定覆盖 | 12345678910 | 2 | | 3 | X=0,z=5,y=0 | 0 | 判定覆盖 | 127910 | 0 | | 4 | X=4,z=10,y=10 | 2 | 条件覆盖 | 1278910 | 2 | | 5 | X=0,z=5,y=0 | 0 | 条件覆盖 | 127910 | 0 | | 6 | X=0,z=5,y=0 | 0 | 基本路径覆盖 | 127910 | 0 | | 7 | X=1,z=1,y=1 | 0 | 基本路径覆盖 | 12345678910 | 0 | | 8 | X=4,z=10,y=10 | 2 | 基本路径覆盖 | 1278910 | 2 |  1. 编写完整的C程序（含输入和输出），使用你所设计的测试用例运行上述程序段。完整填写相应的测试用例表 | | | |
| **五、实验数据记录和处理** | | | |
| #include <iostream>  #include "stdio.h"  #include "math.h"  #include "conio.h"  int test(int x,int y,int z) {  int k = 0, j = 0;  if ((x>0)&&(z<10)) {  k = x\*y-1;  j = sqrt(k);  }  if((x==4)||(y>5))  j = x\*y+10;  j = j % 3;  printf("%d ", k);  return j;  }  int main(){  int x, y, z;  int n;  printf("请依次输入 x y z");  scanf("%d %d %d", &x, &y, &z);  n = test(x, y, z);  printf("%d",n);  getch();  return 0;  } | | | |
| **六、实验结果和分析** | | | |
| . | | | |
| **七、讨论、心得** | | | |
| 这次试验，让我明白了一些基本的覆盖，首先语句覆盖、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和基本路径覆盖这几种白盒测试方法。  其次对于这几种覆盖功能有更深入的了解，希望以后能熟练运用。  最后，明白了白盒测试在软件测试中占据的重要的作用。一定要细心，做测试的时候由于路径复杂很容易出现错误；还要把问题考虑全面，尽量做到所有可能的覆盖。 | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **实验名称** | | **实验二 黑盒测试方法** | | | | | |
| **实验地点** | | C1实验室 | | | **实验时间** | 2019.10.24 | |
| **一、实验目的和要求** | | | | | | | |
| ⑴ 熟练掌握黑盒测试方法中的等价类测试方法和边界值测试方法。  ⑵ 通过实验掌握如何应用黑盒测试方法设计测试用例。  ⑶ 运用所学理论，完成实验研究的基本训练过程。 | | | | | | | |
| **二、实验内容和原理** | | | | | | | |
| 1. 用你熟悉的语言编写一个判断三角形问题的程序。   要求：读入代表三角形边长的三个整数，判断它们能否组成三角形。如果能 够，则输出三角形是等边、等腰或一般三角形的识别信息；如何不能构成三角形， 则输出相应提示信息。  2．使用等价类方法和边界值方法设计测试用例。 | | | | | | | |
| **三、主要仪器设备** | | | 笔记本电脑 | | | | |
| **四、操作方法与实验步骤** | | | | | | | |
| ⑴ 先用等价类和边界值方法设计测试用例，然后用白盒法进行检验与补充。 ⑵ 判断三角形问题的程序流程图和程序流图如图 1 和图 2 所示。用你熟悉 的语言编写源程序。  **package** com.exp; **import** java.util.Scanner; **public class** Demo {  **public static void** main(String[] args) {  **int** x1=0,x2=0,x3=0;  Scanner scanner = **new** Scanner(System.***in***);  Triangle triangle = **null**;  System.***out***.println(**"请输入三个数字："**);  **try** {  x1 = Integer.*parseInt*(scanner.nextLine());  x2 = Integer.*parseInt*(scanner.nextLine());  x3 = Integer.*parseInt*(scanner.nextLine());  triangle = **new** Triangle(x1,x2,x3);  }**catch** (NumberFormatException e){  System.***out***.println(**"请输入数字！"**);  }  **if** (**null**!=triangle){  **if** (*isTriangle*(triangle)){  **if** (*isIsocelesTriangle*(triangle)){  **if** (*isRegulatTriangle*(triangle)){  System.***out***.println(**"是等边三角形"**);  }**else** {  System.***out***.println(**"是等腰三角形"**);  }  }**else** {  System.***out***.println(**"是普通三角形"**);  }  }**else** {  System.***out***.println(**"不是三角形"**);  }  }  }   **private static boolean** isTriangle(Triangle triangle) {  **if** (  (triangle.getX1() + triangle.getX2() > triangle.getX3())  && ( triangle.getX1() + triangle.getX3() > triangle.getX2() )  && ( triangle.getX2() + triangle.getX3() ) > triangle.getX1()  )  **return true**;  **return false**;  }   **private static boolean** isIsocelesTriangle(Triangle triangle){  **if** (triangle.getX1() == triangle.getX2()){  **return true**;  }**else if** (triangle.getX1() == triangle.getX3()){  **return true**;  }**else if** (triangle.getX2() == triangle.getX3()){  **return true**;  }  **return false**;  }   **private static boolean** isRegulatTriangle(Triangle triangle){  **if** (triangle.getX3() == triangle.getX2() && triangle.getX2() == triangle.getX1()){  **return true**;  }  **return false**;  } } **class** Triangle{  **private int x1**;  **private int x2**;  **private int x3**;  **public** Triangle() {  }   **public** Triangle(**int** x1, **int** x2, **int** x3) {  **this**.**x1** = x1;  **this**.**x2** = x2;  **this**.**x3** = x3;  }  **public int** getX1() {  **return x1**;  }  **public void** setX1(**int** x1) {  **this**.**x1** = x1;  }  **public int** getX2() {  **return x2**;  }  **public void** setX2(**int** x2) {  **this**.**x2** = x2;  }  **public int** getX3() {  **return x3**;  }  **public void** setX3(**int** x3) {  **this**.**x3** = x3;  } }  ⑶ 使用等价类方法设计测试用例，并填写完成表 2 和表 3。  ⑷ 输入设计好的测试用例，执行源程序，记录输出结果。    图一 程序流程图  绘图1  图二 程序流图  绘图2  2. 等价类划分表   |  |  |  | | --- | --- | --- | | 输入条件 | 有效等价类 | 无效等价类 | | 是否构成三角形 | 1.三条边均大于零  2.任意两边之和大于第三边 | 3.至少一对两边和小于第三边  4.至少有一边小于等于0 | | 是否等腰三角形 | 5.有且只有两边相等 | 6.任意两边长均不相等 | | 是否等边三角形 | 7.三边都相等 | 8.任意两边不相等 |   表 3. 测试用例表   |  |  |  |  | | --- | --- | --- | --- | | 用例编号 | 测试数据  (A, B, C) | 等价类覆盖情况 | 输出 | | 1 | X=1,y=1,z=1 | 1、2、3 |  | | 2 | X=2,y=2,z=1 | 1、2、5、6 |  | | 3 | X=3,y=3,z=3 | 1、2、7 |  | | 4 | X=1,y=1,z=-1 | 3、4 |  | | 5 | X=1,y=1,z=3 | 3 |  | | | | | | | | |
| **五、实验数据记录和处理** | | | | | | | |
| ID:1  用例名称：一边为0  测试项：正确判断是否三角形  环境要求：OS X 和java  参考文档：软件规格说明书  优先级：高  层次：1  依赖的测试用例：无实际结果  步骤：   1. 依次输入ABC，值为0,6,8 2. 观察结果   期望结果：  输出不是三角形。  ID:2  用例名称：两边为0  测试项：正确判断是否三角形  环境要求：windows7和java  参考文档：软件规格说明书  优先级：高  层次：1  依赖的测试用例：无实际结果  步骤：   1. 依次输入ABC，值为0,0,8 2. 观察结果   期望结果：  输出不是三角形。 | | | | | | | |
| **六、实验结果与分析** | | | | | | | |
| 代码的编写也不是很困难，但是在编写过程中要注意细节问题，比如逻辑控制等，以免程序自身错误导致测试无效。 但通过这个例子可以更好地掌握黑盒测试。它是把程序看作一个不能打开的黑盒子，不考虑程序的内部结构和内部特性，而是考查数据的输入、条件限制和数据输出来完成测试。 | | | | | | | |
| **七、讨论、心得** | | | | | | | |
| 试验过程中有许多困难需要克服，也有需要自己注意的细节。其他方面感觉自己还有许多不到位的地方，希望得到大佬教育。  做黑河测试要以用户的角度，去设计去输入数据与输出数据，需要把被测程序当作一个黑盒子去处理。 | | | | | | | |
| **实验名称** | **实验三 QTP 自动化测试** | | | | | | | |
| **实验地点** | C1实验室 | | | **实验时间** | | | 2019.10.30 | |
| **一、实验目的和要求** | | | | | | | | |
| ⑴ 实验要求  自带笔记本，安装 QTP 软件(V9.2)，独立完成实验内容，可以互相讨论。  ⑵ 实验目的  了解 QTP 软件，熟悉 QTP 自动化测试原理；掌握 QTP 软件的脚本录制；掌握 QTP 检查点的插入，能够利用 QTP 软件进行功能测试(界面测试)和回归测试。 | | | | | | | | |
| **二、实验内容和原理** | | | | | | | | |
| ⑴ 实验原理  Quick Test Professional 简称 QTP，是 Mercury Interactive 公司开发一款自动化测试软件，该公司于 2006 年 11 月被 HP 公司收购。QTP 自动化测试软件主要侧重于功能测试(界面测试)和回归测试。  ⑵ 实验内容  本实验以 QTP9.2 版本为例，首先对 QTP 自带实例“Flight Resevationg”(飞机订票系统)进行脚本录制，然后插入检查点进行测试，最后分析和讨论实验结果。 | | | | | | | | |
| **三、主要仪器设备** | | | | （1）笔记本电脑  （2）QTP9.2 | | | | |
| **四、操作方法与实验步骤** | | | | | | | | |
| 1．安装 QTP9.2，启动该软件。  先解压文件，解压后先查看“安装重要说明”文件，根据该文件说明进行安装， 安装成功后需要重新启动电脑， 根据软件提示操作即可。安装完成后启动“QuickTest Professional”，进入软件主界面。  2．对 Flight Resevationg(飞机订票)系统进行脚本录制。  ⑴ 启动 QTP 软件快捷方式，进入 QTP 工作界面。  ⑵ 点击工具栏上的“Record”图标进行脚本录制，如图 3 所示：  1  图 3  ⑶ 在 弹 出 的 “Record and Run Settings” 对 话 框 中 切 换 至“Windows  Applications”标签，如图 4 所示：  2  图4  ⑷ 在弹出的对话框中，选择“Application details：”中右边的“+”标签，如图 5所示  3  图 5  ⑸ 在弹出的对话框中按照默认选项选择，然后点击“ok”按钮，如图 6 所示：  4  图 6  ⑹ 在弹出的“Login”界面中输入用户名和密码，如图 7 所示。注意：用户名至少是 4 个字符，密码是“mercury”。  5  图 7  ⑺ 在弹出的“Flight Reservation”窗口中选择订票日期、出发地和目的地，然后选择“Flight”图标选择航班，如图 8 所示。注意：日期格式“MM—DD--YY”，并且必须是当前日期之后。  6  图 8  ⑻ 在弹出的“Flight Table”表中选择一趟航班，点击“ok”按钮返回。  ⑼ 在“Name”中填写订票人姓名，选择订票数目“Ticktets”和仓位“Class”，然后点击“Insert Order”按钮，生成订单，如图 9 所示。  7  图 9  ⑽ 选择工具栏“File”菜单中的“Exit”子菜单或者点击关闭按钮，关闭“Flight  Reservation”窗口。点击 QTP 主界面工具栏中的“Stop”按钮，停止脚本录制。至此，订票系统录制完毕，如图 10 所示：  8  图 10  3．保存录制脚本并对脚本进行回放  首先点击工具栏上的保存图标，保存刚才录制的脚本，然后点击“Run” 按钮，根据默认选择，对录制的脚本进行回放并生成测试结果表，如图 11 和  图 12 所示。  9  图 11  10  图 12  在测试结果表中，Result 结果为：Done，表明系统运行完成，测试结果是否满足预期？需要加入“检查点”进行验证。  4．插入“检查点”验证订单是否生成  ⑴ 在“keyword view”视图中单击“Insert Order”这一行，同时 QTP 主界面右下角“Active Screen”中会显示这一步操作的界面，如图 13 所示。  未命名4  图 13  ⑵ 在“Action Screen”视图中“Order No”标签对应的文本框中点鼠标右键，在弹出的快捷菜单中选择“Insert Standard Checkpoint”，如图 14 所示。  未命名  图 14  ⑶ 在弹出的对话框中单击“ok”按钮打开“Checkpoint properties”对话框，在该对话框中只选择“text”属性这一行，然后选择“insert statement：”中的“After currentstep”选项，单击“Configure value”标签中“Constant”类型最右边的图IMG_256标，如图 15所示：  图 15  ⑷ 在弹出的对话框中先选中“Regular expression”标签，然后在 Value 中输入值：“[0-9]+”，（“[0-9]+”是一个正则表达式，该表达式的含义是判断是否为数字），最后点击“ok”按钮，如图 16 所示：  14  图 16  ⑸ 返回“Checkpoint properties”对话框，单击“ok”按钮，至此，检查点插入成功，  ⑹ 保存脚本，重新运行脚本，生成如图 17 所示的测试结果表。测试结果表明测试通过，订单生成成功。  5．在图 15 中的“constant”一栏中输入常数值，重新运行脚本，分析实验结果  15  图 17  6. 重新录制脚本。重复上述过程“2. 对 Flight Resevationg(飞机订票)系统进行脚本录制”中的⑴-⑼，接着录制以下步骤⑽和⑾。  ⑽ 选择工具栏“File”菜单中的“Fax Order”子菜单，打开“Fax Order No.xx”将机票订单传真出去，在“Fax Order No.xx”窗口中填写传真号码以及代理人签名“Agent signature”，其中 No.xx 表示生成的传真订单号，具体号码以实际为准， 然后点击“Send”，如图 18 所示：  b  图 18  ⑾ 发送成功后系统返回“Flight Reservation”，并显示传真发送状态， 如图 19  所示。  c  图 19  ⑿ 关闭“Flight Reservation”窗口，停止录制。7．回放以上脚本，回放结果失败，分析原因。  8．按以下步骤修改。  ⒀ 切换到“Keyword View”视图，鼠标右击“Fax Order No.xx”，在弹出快捷菜单中选择“Object Properties”，如图 20 所示。  未命名4  图 20  ⒁ 在弹出的“Object Properties”窗口中点击 text 的 Value 值中的图标“<#>”，如图 21 所示。  IMG_256  图 21  ⒂ 在弹出的对话框中，选中“Regular expression”，把 constant 中的值改为如图22 所示的内容。  IMG_256  图 22  ⒃ 关闭窗口，重新回放脚本，查看脚本是否回放成功。  9．参照以上实验，再添加一个验证“Login”界面中“Agent Name”的检查点  提示：利用正则表达式对 Agent Name 进行校验，即：对不少于 4 个字符的  Agent Name 再加以限定，分别设计一个能够通过校验的正则表达式和一个不能  通过校验的正则表达式。 | | | | | | | | |
| **五、实验结果与分析** | | | | | | | | |
|  | | | | | | | | |
| **七、讨论、心得** | | | | | | | | |
|  | | | | | | | | |