《软件测试》实训大纲

一、关于本实训大纲的说明：

1、**课程名称：**软件测试

2、**课程归属院系及教研室：**应用技术学院计算机应用技术教研室

3、**适用专业、层次：**软件工程应用本科

4、**学时、项目数量：**8 学时，8 个项目

5、**实训设备**

（1）主流 PC 机一套，要求安装 windows 操作系统、VC 开发工具和 OFFICE 工具；

（2）安装 QuickTest 和 WinRunner 自动测试工具。

6、**实训地点：**软件工程实训室

二、实训目的：

《软件测试》是软件工程专业的学科主干课，本实验课程配合《软件测试技术》理论课同步开设，其中包括验证型、设计型和综合型实验。本实验课程着眼于理论与应用的结合，注重培养学生软件测试的实际动手能力，增强软件工程项目的质量管理意识。通过实验教学，使学生掌握软件测试的方法和技术，并能运用软件测试工具进行自动化测试。

三、实训任务和基本要求：

本实验课程要求学生完成适当的上机实习，并写出相应的实验报告。验证和设计题单独完成，综合题任选一题。验证型题目使学生熟悉常用的软件测试工具。设计型题目使学生掌握软件测试的基本理论和基本方法，培养基本的应用能力。综合型题目在于提高学生分析问题、解决问题的能力，培养工程项目的测试能力和管理能力。

四、实训课时分配：

序号	实训项目	实训课时	实训场地	备注
1	测试用例的设计和单元测试工具	3	软件工程实训室	
2	功能性测试	3	软件工程实训室	
3	结构性测试	3	软件工程实训室	
4	单元测试	3	软件工程实训室	
5	QuickTest Professional 初级使用	3	软件工程实训室	
6	QuickTest Professional 高级使用	3	软件工程实训室	
7	WinRunner 的使用	4	软件工程实训室	
8	Web 系统测试	4	软件工程实训室	

五、考核办法（考核内容、比例、方法、评分标准）

指导教师从学生的态度、技能、效果三方面综合考核，具体成绩标准及比例见如下实训成绩评价标准表。

评价指标		成绩				
		优	良	中	差	得分
态度	出勤率	5	4	3	2	
	纪律	5	4	3	2	
	积极性	5	4	3	2	
技能	熟练程度	20	16	13	10	
	操作能力	15	12	10	7	
	分析解决问题能力	20	16	13	10	
效果	按时完成情况	10	8	7	5	

	完成质量	20	16	13	10	

《软件测试》实训指导书

一、课程名称：软件测试

二、课程归属院系及教研室：应用技术学院计算机应用技术教研室

三、适用专业：软件工程应用本科

四、实训学时：10 学时

五、实验设备

（1）主流 PC 机一套，要求安装 windows 操作系统、VC 开发工具和 OFFICE 工具；

（2）安装 QuickTest 和 WinRunner 自动测试工具。

六、实训成绩标准及比例

指导教师从学生的态度、技能、效果三方面综合考核，具体成绩标准及比例见如下实训成绩评价标准表。

评价指标		成绩				
		优	良	中	差	得分
态度	出勤率	5	4	3	2	
	纪律	5	4	3	2	
	积极性	5	4	3	2	
技能	熟练程度	20	16	13	10	
	操作能力	15	12	10	7	
	分析解决问题能力	20	16	13	10	
效果	按时完成情况	10	8	7	5	
	完成质量	20	16	13	10	

七、软件测试相关知识

1．软件测试

软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码的最终复审，是软件质量保证的关键步骤。 软件测试是为了发现错误而执行程序的过程。或者说，软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例（即输入数据及其预期的输出结果），并利用这些测试用例去运行程序，以发现程序错误的过程。

2．测试过程

为了保证测试的质量，将测试过程分成几个阶段，即：代码审查、单元测试、集成测试、系统测试和验收测试。

代码会审由一组人通过阅读、讨论和争议对程序进行静态分析的过程。

单元测试集中在检查软件设计的最小单位——模块上，通过测试发现实现该模块的实际功能与定义该模块的功能说明不符合的情况，以及编码的错误。

集成测试是将模块按照设计要求组装起来同时进行测试，主要目标是发现与接口有关的问题。

系统测试是测试整个系统，以证实它满足“需求规格说明书”所规定的功能、质量和性能等方面的特性。

验收测试的目的是向未来的用户表明系统能够像预定要求那样工作。与系统测试非常相似，主要区别是测试人员不同，验收测试由用户执行。

3．测试方法

软件测试的方法分为功能性测试和结构性测试。

功能测试是指在对程序进行功能抽象的基础上，将程序划分成功能单元，然后在数据

抽象的基础上，对每个功能单元生成测试数据进行测试。进行功能测试时，被测程序被当作打不开的黑盒，因而无法了解其内部构造，因此又称为黑盒测试。

结构性测试是知道产品内部工作过程，检测产品内部动作是否按照规格说明书的规定正常进行。它允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。此方法把测试对象看作一个透明的盒子，又叫白盒测试。

4．测试工具

软件测试的工作量很大（据统计，会用到 40% 的开发时间；一些可靠性要求非常高的软件，测试时间甚至占到总开发时间的 60% ），但测试却是在整个软件过程中极有可能应用计算机进行自动化的工作，原因是测试的许多操作是重复性的、非智力创造性的、需求细致注意力的工作。测试工具的应用已经成为了普遍的趋势。测试工具一般可分为白盒测试工具、黑盒测试工具、性能测试工具，另外还有用于测试管理（包括测试流程管理、缺陷跟踪管理、测试用例管理）的工具。

本实验课程中主要涉及到 QuickTest Professional 和 WinRunner 自动化测试工具。

5. 测试用例的编写

(1)．测试用例

软件测试的本质是针对要测试的内容确定一组测试用例。测试用例是为实施一次测试而向被测系统提供的输入数据、操作或各种环境设置。

测试用例应该包含基本的内容有输入和预期输出，输入实际有两种类型：前提（在测试用例执行前已经存在的环境）和由某种测试方法所标识的实际输入。预期输出也有两类：后果和实际输出。测试活动要建立必要的前提条件，提供测试用例输入、观测输出、然后将这些输出与预期输出进行比较，以确定该测试是否通过。开发良好的测试用例的其他信息（如表 1）主要支持测试管理，测试用例需求被开发、评审、使用、管理和保存。

表 1-1 测试用例

测试用例 ID			
目的			
前提			
输入			
预期输出			
后果			
执行历史			
日期		结果	执行人

有两种基本方法可以用来标识测试用例，即功能性测试和结构性测试。

功能性测试的基本观点是，任何程序都可以看作是将从定义域取值映射到输出值域的函数。这种观点常常在工程中使用，将系统看作是黑盒。采用功能性方法标识测试用例，所使用的唯一信息就是软件的规格说明。功能性测试用例具有两个显著的优点：（1）功能性测试与软件如何实现无关，所以如果实现发生变化，测试用例仍然有用；（2）测试用例开发可以与实现并行进行，因此可缩短总的开发时间。在缺点方面，功能性测试用例也常常带来两个问题：测试用例可能存在严重的冗余，此外可能还会有未测试的软件漏洞。功能性测试的主流方法主要有：边界值分析、健壮性分析、最坏情况分析、特殊值测试、输入等价类、输出等价类和基于决策树的测试。

结构性测试有时也叫白盒测试。结构性测试是知道软件产品内部工作过程，检测软件产品内部动作是否按照规格说明书的规定正常进行。结构性测试需要全面了解程序内部逻

辑结构、对所有逻辑路径进行测试。结构性测试是穷举路径测试，并力求提高测试覆盖率。结构性测试的主要方法有：逻辑覆盖测试、基路径测试、数据流测试等方法。

在实际应用中，为全面的测试软件产品，一般将结构性测试和功能性测试结合起来使用。

(2) 软件缺陷分类

有多种方法可以对缺陷分类：以出现相应错误的开发阶段来划分、以相应失效产生的后果来划分、以解决难度来划分、以不解决难度会产生的风险来划分等等。在日常的软件测试中，通常给出的缺陷是根据缺陷后果的严重程度来进行划分，如下表 2 所示。在实际应用中可以根据具体情况对严重程度来划分不同的等级。

(3) 测试用例的选择

选择测试用例是软件测试员最重要的一项任务，不正确的选择可能导致测试量过大或过小，甚至测试目标不对。

从工程实践的角度讲，测试用例有几条基本准则：

（1）测试用例的代表性：能够代表各种合理和不合理的、合法的和非法的、边界和越界的,以及　极限的输入数据、操作和环境设置等；

（2）测试结果的可判定性：即测试执行结果的正确性是可判定的或可评估的；

（3）测试结果的可再现性：即对同样的测试用例,系统的执行结果应当是相同的。

表 1-2 缺陷划分表

编号	缺陷等级	举例
1	轻微	词语拼写错误
2	中等	误导或重复信息
3	使人不悦	被截取的名称
4	影响使用	有些交易没有处理
5	严重	丢失交易
6	非常严重	不正确的交易处理
7	极为严重	经常出现非常严重的错误
8	无法忍受	数据库破坏
9	灾难性	系统停机
10	容易传染	扩展到其他系统的系统停机

实训一 测试用例的设计和单元测试工具

[实验目的]

1．掌握白盒测试、黑盒测试用例的设计。

2．熟悉使用 Junit 框架进行基于 java 语言的单元测试。

[实验要求]

1、使用白盒测试用例设计方法为下面的程序设计测试用例：

程序要求：10 个铅球中有一个假球（比其他铅球的重量要轻），用天平三次称出假球。

程序设计思路：第一次使用天平分别称 5 个球，判断轻的一边有假球；拿出轻的 5 个球，取出其中 4 个第二次称，两边分别放 2 个球：如果两边同重，则剩下的球为假球；若两边不同重，拿出轻的两个球称第三次，轻的为假球。

2、使用等价类划分法设计下面的测试用例：

输入三个整数作为边，分别满足一般三角形、等腰三角形和等边三角形。

[实验内容及步骤]

1、使用白盒测试用例设计方法为下面的程序设计测试用例：

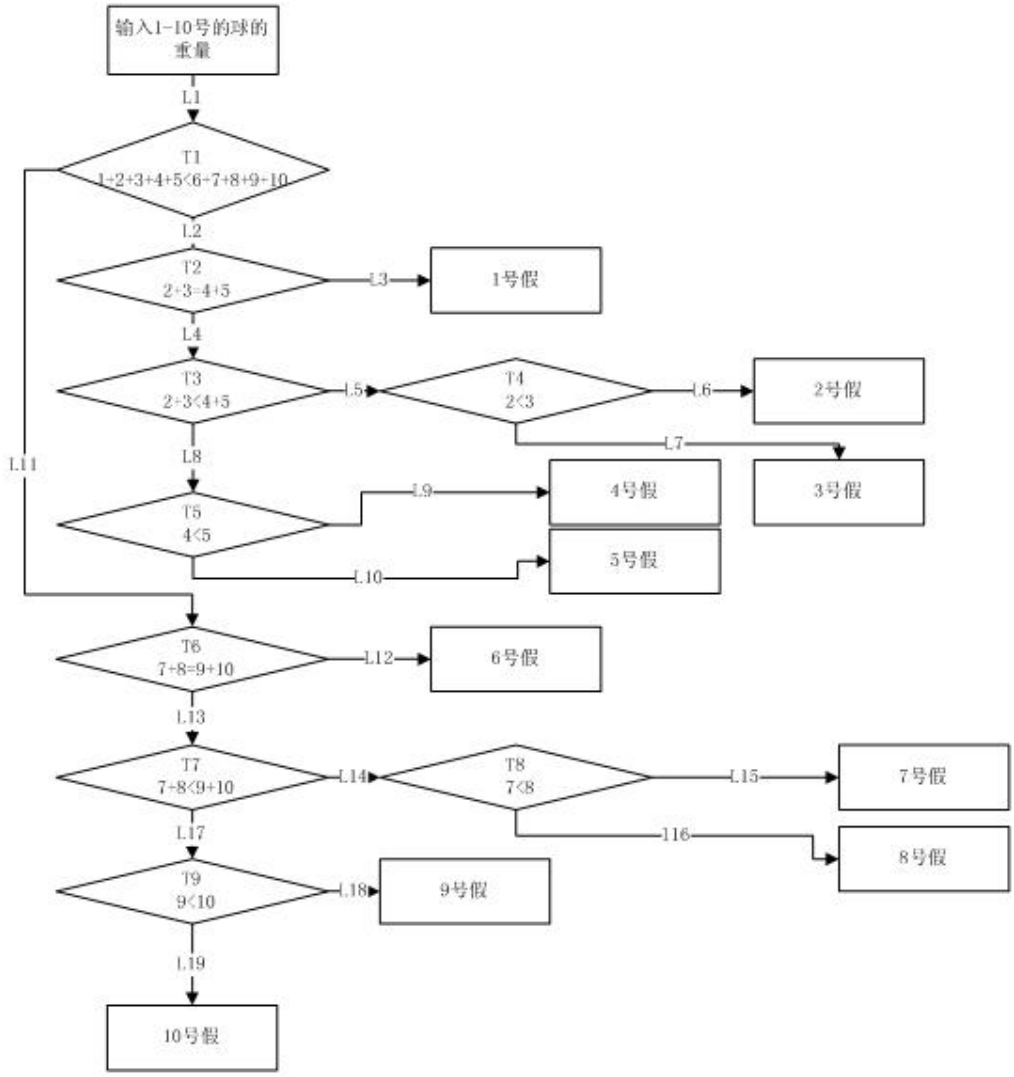
（1）源程序

```
package P1;

public class SearchBall {
    private static int x[]=new int[10];
    public SearchBall(){}
  public void setBWeight(int w[]){
        for(int i=0;i<w.length;i++){
            x[i]=w[i];
            }

    }
    public String BeginSearch(){
        if(x[0]+x[1]+x[2]+x[3]+x[4]<x[5]+x[6]+x[7]+x[8]+x[9])
        {
            if(x[1]+x[2]==x[3]+x[4])
            {
                System.out.println("1 号是假球");
                return "1 号是假球";
            }
        if(x[1]+x[2]<x[3]+x[4])
        {
            if (x[1]<x[2])
            {
                System.out.println("2 号是假球");
                return "2 号是假球";
            }
         else
            {
             System.out.println("3 号是假球");
             return "3 号是假球";
         }
            }
        else
        {
            if (x[3]<x[4])
            {
                System.out.println("4 号是假球");
                return "4 号是假球";
```

```
                }
            else
                {
                System.out.println("5 号是假球");
                return "5 号是假球";
                }
                }
            }
            else
            {
                if(x[6]+x[7]==x[8]+x[9])
                {
                    System.out.println("6 号是假球");
                    return "6 号是假球";
                }
            if(x[6]+x[7]<x[8]+x[9])
            {
                if (x[6]<x[7])
                {
                    System.out.println("7 号是假球");
                    return "7 号是假球";
                }
              else
                {
                 System.out.println("8 号是假球");
                 return "8 号是假球";
              }
                }
            else
            {
                if (x[8]<x[9])
                {
                    System.out.println("9 号是假球");
                    return "9 号是假球";
                }
              else
              {
                    System.out.println("10 号是假球");
                    return "10 号是假球";
              }
              }
            }

            }
public static void main (String args[]){
        SearchBall sb=new SearchBall();
      int a0[]={1,2,2,2,2,2,2,2,2,2};
      int a1[]={2,1,2,2,2,2,2,2,2,2};
      int a2[]={2,2,1,2,2,2,2,2,2,2};
      int a3[]={2,2,2,1,2,2,2,2,2,2};
      int a4[]={2,2,2,2,1,2,2,2,2,2};
      int a5[]={2,2,2,2,2,1,2,2,2,2};
      int a6[]={2,2,2,2,2,2,1,2,2,2};
      int a7[]={2,2,2,2,2,2,2,1,2,2};
      int a8[]={2,2,2,2,2,2,2,2,1,2};
      int a9[]={2,2,2,2,2,2,2,2,2,1};
      sb.setBWeight(a0);
      System.out.println(sb.BeginSearch());
```

```
        sb.setBWeight(a1);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a2);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a3);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a4);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a5);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a6);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a7);
        System.out.println(sb.BeginSearch());
           sb.setBWeight(a8);
        System.out.println(sb.BeginSearch());
        sb.setBWeight(a9);
        System.out.println(sb.BeginSearch());
        }
}
```

（2）测试用例设计



（3）测试程序设计

测试用例	轻球编号	所经路径	覆盖条件
Case1	1	L1，L2,L3	T1,T2
Case2	2	L1,L2,L4,L5,L6	T1,T2,T3,T4
Case3	3	L1,L2,L4,L5,L7	T1,T2,T3,T4
Case4	4	L1,L2,L4,L8,L9	T1,T2,T3,T5
Case5	5	L1,L2,L4,L8,L10	T1,T2,T3,T5
Case6	6	L1,L11,L12	T1,T6
Case7	7	L1,L11,L13,L14,L15	T1,T6,T7,T8
Case8	8	L1,L11,L13,L14,L16	T1,T6,T7,T8
Case9	9	L1,L11,L13,L17,L18	T1,T6,T7,T9
Case10	10	L1,L11,L13,L17,L19	T1,T6,T7,T9

```
package P1;
import junit.framework.TestCase;
public class SearchBallTest extends TestCase {
    int a0[]={1,2,2,2,2,2,2,2,2,2};
    int a1[]={2,1,2,2,2,2,2,2,2,2};
    int a2[]={2,2,1,2,2,2,2,2,2,2};
    int a3[]={2,2,2,1,2,2,2,2,2,2};
    int a4[]={2,2,2,2,1,2,2,2,2,2};
    int a5[]={2,2,2,2,2,1,2,2,2,2};
    int a6[]={2,2,2,2,2,2,1,2,2,2};
    int a7[]={2,2,2,2,2,2,2,1,2,2};
    int a8[]={2,2,2,2,2,2,2,2,1,2};
    int a9[]={2,2,2,2,2,2,2,2,2,1};

     public void testBeginSearch() {
          SearchBall SB=new SearchBall();
            SB.setBWeight(a0);
            assertEquals("1 号是假球",SB.BeginSearch());
            SB.setBWeight(a1);
            assertEquals("2 号是假球",SB.BeginSearch());
            SB.setBWeight(a2);
            assertEquals("3 号是假球",SB.BeginSearch());
            SB.setBWeight(a3);
            assertEquals("4 号是假球",SB.BeginSearch());
            SB.setBWeight(a4);
            assertEquals("5 号是假球",SB.BeginSearch());
            SB.setBWeight(a5);
            assertEquals("6 号是假球",SB.BeginSearch());
            SB.setBWeight(a6);
            assertEquals("7 号是假球",SB.BeginSearch());
            SB.setBWeight(a7);
            assertEquals("8 号是假球",SB.BeginSearch());
            SB.setBWeight(a8);
            assertEquals("9 号是假球",SB.BeginSearch());
            SB.setBWeight(a9);
            assertEquals("10 号是假球",SB.BeginSearch());
     }
     public static void main(String args[]) {
          junit.textui.TestRunner.run(SearchBallTest.class);
          }

}
```

2、使用等价类划分法设计下面的测试用例：

（1）源程序

```
package p1;
import java.io.*;
public class JTriangle {
        private int b,c,a;
        private static int x1,x2,x3;
        private static String s1,s2,s3;
    public void setA(int a){
            this.a=a; }
    public void setB(int b){
            this.b=b;}
    public void setC(int c){
            this.c=c; }
    public boolean IsTriangle(){
           if(a+b>c&&a+c>b&&b+c>a&&a!=0&&b!=0&&c!=0)
                return true;
           else
                return false;
       }

      public JTriangle(){

       }
      public JTriangle(int _a,int _b,int _c){

           setA(_a);
           setB(_b);
           setC(_c);
      }
       public String JudgeTriangle()
       {
            if(IsTriangle())
       {
                    if(a==b&&a==c)
                       {
                            return "等边三角形";
                       }
                       else if(a==b||b==c||a==c){
                            return "等腰三角形";
                       }
                       else
                       {
                            return "一般三角形";
                       }
       }
            else
                 return "不能组成三角形";
 }
       public boolean ISNumble(String s)
       {
            if(s.length()==0){
            return false;
       }
       else{
            char ch[]=s.toCharArray();

            for(int i=0;i<ch.length;i++)
            {
```

```
                if(ch[i]>='0'&&ch[i]<='9')
                    continue;
                else
                    return false;

            }

            return true;
        }
        }
    public void SetTriangle(){
            try
            {
            BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
            System.out.println("请输入三角形的三边：");
            System.out.println("a= ");
            s1=in.readLine();

            while(!ISNumble(s1)){
                System.out.println("你输入的不是个数字");
                System.out.println("a= ");
                s1=in.readLine();
            }
        System.out.println("b= ");
            s2=in.readLine();
            while(!ISNumble(s2)){
                System.out.println("你输入的不是个数字");
                System.out.println("b= ");
                s2=in.readLine();
            }
        System.out.println("c= ");
            s3=in.readLine();
            while(!ISNumble(s3)){
                System.out.println("你输入的不是个数字");
                System.out.println("C= ");
                s3=in.readLine();
            }
            }
           catch(IOException e)
           {}
           x1=Integer.parseInt(s1);
           x2=Integer.parseInt(s2);
           x3=Integer.parseInt(s3);


           setA(x1);
           setB(x2);
           setC(x3);
    }
public static void main(String args[])
        {       JTriangle tr=new JTriangle();
        tr.SetTriangle();
        System.out.print(tr.JudgeTriangle()) ;
        }

}
```

（2）测试用例设计

输入条件	有效等价类	无效等价类
是否三角形的三条边	(A>0)，(1) (B>0)，(2) (C>0)，(3) (A+B>C)，(4) (B+C>A)，(5) (A+C>B)，(6)	(A≤0)，(7) (B≤0)，(8) (C≤0)，(9) (A+B≤C)，(10) (B+C≤A)，(11) (A+C≤B)，(12)
是否等腰三角形	(A=B)，(13) (B=C)，(14) (C=A)，(15)	(A≠B) and (B≠C) and (C≠A) (16)
是否等边三角形	(A=B) and (B=C) and (C=A) (17)	(A≠B)，(18) (B≠C)，(19) (C≠A)，(20)

用例	【A，B，C】	覆盖等价类	输出
1	【3，4，5】	(1)，(2)，(3)，(4)，(5)，(6)	一般三角形
2	【0，1，2】	(7)	不能构成三角形
3	【1，0，2】	(8)	
4	【1，2，0】	(9)	
5	【1，2，3】	(10)	
6	【1，3，2】	(11)	
7	【3，1，2】	(12)	
8	【3，3，4】	(1)，(2)，(3)，(4)，(5)，(6)，(13)	等腰三角形
9	【3，4，4】	(1)，(2)，(3)，(4)，(5)，(6)，(14)	
10	【3，4，3】	(1)，(2)，(3)，(4)，(5)，(6)，(15)	
11	【3，4，5】	(1)，(2)，(3)，(4)，(5)，(6)，(16)	非等腰三角形
12	【3，3，3】	(1)，(2)，(3)，(4)，(5)，(6)，(17)	是等边三角形

（3）测试程序设计

```
package p1;
import junit.framework.TestCase;
public class JTriangleTest extends TestCase {
        JTriangle tr;

      public void testJudgeTriangle() {
```

```
        tr=new JTriangle(0,1,2);
            assertEquals("不能组成三角形",tr.JudgeTriangle());
        tr=new JTriangle(1,0,2);
            assertEquals("不能组成三角形",tr.JudgeTriangle());
        tr=new JTriangle(1,2,0);
            assertEquals("不能组成三角形",tr.JudgeTriangle());

            tr=new JTriangle(1,2,3);
            assertEquals("不能组成三角形",tr.JudgeTriangle());
            tr=new JTriangle(1,3,2);
            assertEquals("不能组成三角形",tr.JudgeTriangle());
            tr=new JTriangle(3,1,2);
            assertEquals("不能组成三角形",tr.JudgeTriangle());

            tr=new JTriangle(3,4,5);
             assertEquals("一般三角形",tr.JudgeTriangle());

             tr=new JTriangle(3,3,4);
                 assertEquals("等腰三角形",tr.JudgeTriangle());
                 tr=new JTriangle(3,4,3);
                     assertEquals("等腰三角形",tr.JudgeTriangle());
                     tr=new JTriangle(4,3,3);
                         assertEquals("等腰三角形",tr.JudgeTriangle());

        tr=new JTriangle(3,3,3);
            assertEquals("等边三角形",tr.JudgeTriangle());

    }

}
```

实训二　功能性测试

[实训目的]

（1）能熟练应用功能性测试技术进行测试用例设计；

（2）对测试用例进行优化设计；。

[实训原理]

功能测试是指在对程序进行功能抽象的基础上，将程序划分成功能单元，然后在数据抽象的基础上，对每个功能单元生成测试数据进行测试。进行功能测试时，被测程序被当作打不开的黑盒，因而无法了解其内部构造，因此又称为黑盒测试。

一、等价类测试

等价类测试方法是把所有可能的输入数据，即程序的输入域划分成若干部分，然后从每一部分中选取少数有代表性的数据作为测试用例。使用等价类划分方法设计测试用例要经历划分等价类（列出等价类表）和选取测试用例两步。

等价类的划分有两种不同的情况： ① 有效等价类：是指对于程序的规格说明来说，是合理的，有意义的输入数据构成的集合。 ② 无效等价类：是指对于程序的规格说明来说，是不合理的，无意义的输入数据构成的集合。在设计测试用例时，要同时考虑有效等价类和无效等价类的设计。

用等价类划分法设计测试用例步骤：

(1) 形成等价类表，每一等价类规定一个唯一的编号；

(2) 设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类，重复这一步骤，直到所有有效等价类均被测试用例所覆盖；

(3) 设计一个新测试用例，使其只覆盖一个无效等价类，重复这一步骤直到所有无效等价类均被覆盖。

二、边界值测试

1、边界值分析

边界值分析是考虑边界条件而选取测试用例的一种功能测试方法。边界值分析关注输入空间的边界，以标识测试用例，因为错误更可能出现在输入变量的极值附近。

边界值分析的基本思想是：使用在最小值、略高于最小值、正常值、略低于最大值和最大值处取输入变量值。

2、健壮性测试

健壮性是指在异常情况下，软件还能正常运行的能力。健壮性考虑的主要部分是预期输出，而不是输入。

健壮性测试是边界值分析的一种简单扩展。除了变量的 5 个边界分析取值还要考虑略超过最大值（max）和略小于最小值（min）时的情况。

3、最坏情况测试

最坏情况测试将意味着更大工作量，n 变量函数的最坏情况测试会产生 5 的 n 次方个测试用例，而边界值分析只产生 4n+1 个测试用例。

三、基于决策表的测试

决策表适合描述不同条件集合下采取行动的若干组合的情况。使用决策表标识测试用例，则把条件解释为输入，行动解释为输出。有时条件最终引用输入的等价类，行为引用被测试软件的主要功能处理部分，规则解释为测试用例。

对于有限条目决策表，如果有 n 个条件，则必须有 2 条规则。如果不关心条目实际地表明条件是不

相关的，则没有不关心条目的规则统计为 1 条规则，规则中每出现一个不关心条目，该规则数乘一次 2。

[实训内容]

1．题目一：电话号码问题

某城市电话号码由三部分组成。它们的名称和内容分别是：

（1）地区码：空白或三位数字；

（2）前缀：非'0'或'1'的三位数字；

（3）后缀：4 位数字。

假定被测程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的电话号码。根据该程序的规格说明，作等价类的划分，并设计测试方案。

2．题目二：三角形问题

根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例。

“一个程序读入三个整数。把此三个数值看成是一个三角形的三个边。这个程序要打印出信息，说明这个三角形是三边不等的、是等腰的、还是等边的。”

3．题目三：日期问题

用决策表测试法测试以下程序: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（即列出所有的动作桩）。

(3) 根据 (1) 和 (2) ，画出简化后的决策表。

4．题目四：找零钱最佳组合

假设商店货品价格(R)皆不大于 100 元（且为整数），若顾客付款在 100 元内 (P) ， 求找给顾客最少货币个（张）数？（货币面值 50 元 10 元，5 元，1 元四 种 ）

[实训步骤]

（1） 根据功能性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；

[实训要求]

（1） 根据题目要求编写测试用例（参照表 1 进行用例设计）；

（2） 实验结果要求给出两套测试用例集测试效果比较；

（3） 撰写实验报告；

[实训思考]

（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；

实训三　结构性测试

[实训目的]

（1）掌握结构性测试技术，并能应用结构性测试技术设计测试用例；

（2）对测试用例进行优化设计；

[实训原理]

结构性测试是知道产品内部工作过程，检测产品内部动作是否按照规格说明书的规定正常进行。结构性测试允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。

一、逻辑覆盖

结构性测试力求提高测试覆盖率。逻辑覆盖是对一系列测试过程的总称，它是在使用白盒测试法时，选用测试用例执行程序逻辑路径的方法。

逻辑覆盖按覆盖程度由低到高大致分为以下几类：

（1） 语句覆盖：设计若干测试用例，使程序中每一可执行语句至少执行一次；

（2） 判断覆盖：设计用例，使程序中的每个逻辑判断的取真取假分支至少经历一次；

（3） 条件覆盖：设计用例，使判断中的每个条件的可能取值至少满足一次；

（4） 判断/条件覆盖：设计用例，使得判断中的每个条件的所有可能结果至少出现一次，而且判断本身所有可能结果也至少出现一次；

（5） 条件组合覆盖。设计用例，使得每个判断表达式中条件的各种可能组合都至少出现一次；显然，满足⑤的测试用例也一定是满足②、③、④的测试用例。

（6） 路径覆盖。设计足够的测试用例，使程序的每条可能路径都至少执行一次。

如果把路径覆盖和条件组合覆盖结合起来，可以设计出检错能力更强的测试数据用例。

二、基本路径测试

如果把覆盖的路径数压缩到一定限度内，例如，程序中的循环体只执行零次和一次，就成为基本路径测试。它是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例的方法。

设计出的测试用例要保证在测试中，程序的每一个可执行语句至少要执行一次。

① 程序的控制流图

控制流图是描述程序控制流的一种图示方法。基本控制构造的图形符号如图所示。符号○称为控制流图的一个结点，一组顺序处理框可以映射为一个单一的结点。控制流图中的箭头称为边，它表示了控制流的方向，在选择或多分支结构中分支的汇聚处，即使没有执行语句也应该有一个汇聚结点。边和结点圈定的区域叫做区域，当对区域计数时，图形外的区域也应记为一个区域。

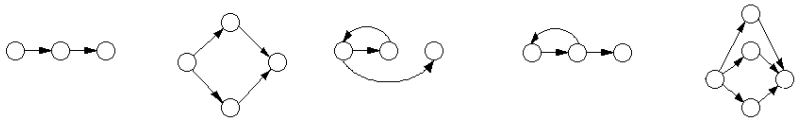


图 3-1 控制流图的各种图形符号

如果判定中的条件表达式是复合条件时，即条件表达式是由一个或多个逻辑运算符（OR，AND，NAND，NOR）连接的逻辑表达式，则需要改复合条件的判定为一系列只有单个条件的嵌套的判定。例如对应图 3-2. (a) 的复合条件的判定，应该画成如图 4-2. (b) 所示的控制流图。 条件语句 if a OR b 中

条件 a 和条件 b 各有一个只有单个条件的判定结点。

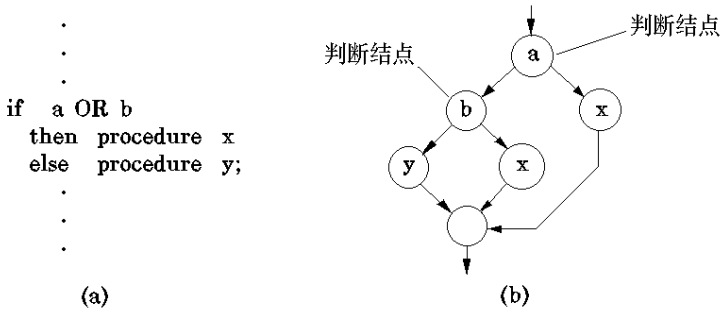


图 3-2　复合逻辑下的控制流图

② 计算程序环路复杂性

进行程序的基本路径测试时，程序的环路复杂性给出了程序基本路径集合中的独立路径条数，这是确保程序中每个可执行语句至少执行一次所必需的测试用例数目的上界。

所谓独立路径，是指包括一组以前没有处理的语句或条件的一条路径。如在图 3-3(b)所示的控制流图中，一组独立的路径是：

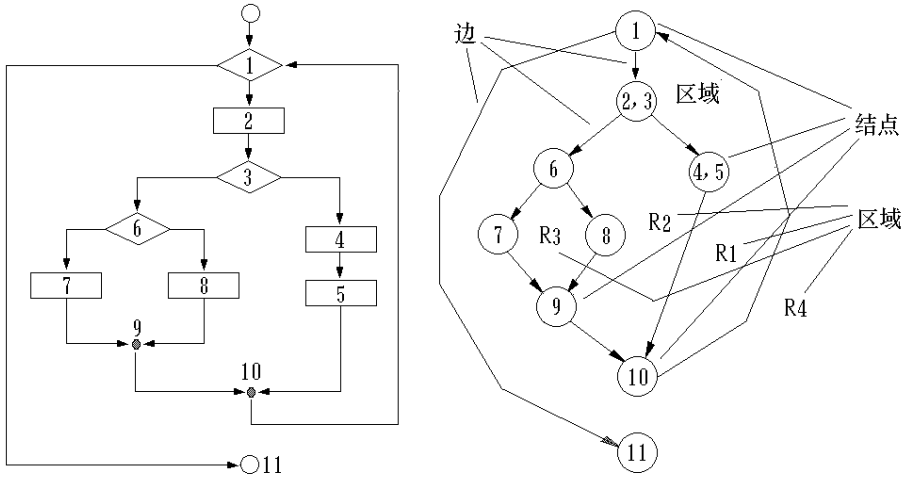
path1：1 - 11

path2：1 - 2 - 3 - 4 - 5 - 10 - 1 - 11

path3：1 - 2 - 3 - 6 - 8 - 9 - 10 - 1 - 11

path4：1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11

路径 path1，path2，path3，path4 组成了图 3-3 (b) 所示控制流图的一个基本路径集。只要设计出的测试用例能够确保这些基本路径的执行，就可以使得程序中的每个可执行语句至少执行一次，每个条件的取真分支和取假分支也能得到测试。基本路径集不是唯一的，对于给定的控制流图，可以得到不同的基本路径集。



(a) 程序流程图　　(b) 控制流图

图 3-3 程序流程图与对应的控制流图

通常环路复杂性可用以下三种方法求得。

§ 将环路复杂性定义为控制流图中的区域数。

§ 设 E 为控制流图的边数，N 为图的结点数，则定义环路复杂性为 V(G)＝E－N＋2。

§ 若设 P 为控制流图中的判定结点数，则有 V(G)＝P＋1。

因为图 3.3 (b)所示控制流图有 4 个区域。其环路复杂性为 4。 它是构成基本路径集的独立路径数的上界。可以据此得到应该设计的测试用例的数目。

③ 导出测试用例

利用逻辑覆盖方法生成测试用例，确保基本路径集中每条路径的执行。

[实训内容]

1．题目一：使用逻辑覆盖测试方法测试以下程序段

```
  void DoWork (int x,int y,int z)
 {
1     int k=0,   j=0;
2     if ( (x>3)&&(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）是对每条语句的编号。

（1）画出程序的控制流图（用题中给出的语句编号表示）。

（2）分别以语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、组合覆盖和路径覆盖方法设计测试用例，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）。

2．题目二：三角形问题

在三角形计算中，要求输入三角型的三个边长：A、B 和 C。当三边不可能构成三角形时提示错误，可构成三角形时计算三角形周长。若是等腰三角形打印“等腰三角形”，若是等边三角形，则提示“等边三角形”。画出程序流程图、控制流程图、计算圈复杂度 V(g)，找出基本测试路径。

3．题目三：计算生日是星期几

已知公元 1 年 1 月 1 日是星期一。编写一个程序，只要输入年月日，就能回答那天是星期几。应用逻辑覆盖方法和基路径测试方法为上面的问题设计测试用例。

4．题目四：选择排序

下面是选择排序的程序，其中 datalist 是数据表，它有两个数据成员：一是元素类型为 Element 的数组 V，另一个是数组大小 n。算法中用到两个操作，一是取某数组元素 V[i]的关键码操作 getKey ( )，一是交换两数组元素内容的操作 Swap( )：

```
void SelectSort ( datalist & list ) {
            /对表 list.V[0]到 list.V[n-1]进行排序,  n 是表当前长度。
  for ( int i = 0; i < list.n-1; i++ ) {
  int k = i;        //在 list.V[i].key 到 list.V[n-1].key 中找具有最小关键码的对象
  for ( int j = i+1;   j < list.n;   j++)
    if ( list.V[j].getKey ( ) < list.V[k].getKey ( ) ) k = j;//当前具最小关键码的对象
```

```
    if ( k != i ) Swap ( list.V[i], list.V[k] );       //交换
          }
      }
```

（1） 试计算此程序段的 McCabe 复杂性；
（2） 用基本路径覆盖法给出测试路径；
（3） 为各测试路径设计测试用例。

[实训步骤]

（1） 根据结构性测试技术设计测试用例，主要考虑逻辑覆盖测试（语句覆盖、判断覆盖、条件覆盖、判断/条件覆盖、条件组合覆盖、路径覆盖）和基路径测试技术；
（2） 根据所学知识确定优化策略（原则：用最少的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、参照表 2 所示的缺陷等级给出缺陷列表；
（4） 计算测试用例的分支覆盖率、条件覆盖率和语句覆盖率等测试管理指标；

[实训要求]

（1） 根据题目要求编写测试用例（参照表 1 进行用例设计）；
（2） 实验结果要求给出两套测试用例集测试效果比较；计算测试用例的分支覆盖率、条件覆盖率和语句覆盖率等测试管理指标；
（3） 撰写实验报告；

[实训思考]

（1） 使用公式 e-n+p/e-n+2p 确定的 McCabe 基路径与实际分析的是否完全一致？
（2） DD-路径和 MM-路径的区别与联系。

实训四　单元测试

[实训目的]

（1）　掌握单元测试技术，并按单元测试的要求设计测试用例。

（2）　能熟练应用功能性测试技术进行测试用例设计；

（3）　能熟练应用结构性测试技术进行测试用例设计；

（4）　对测试用例进行优化设计；

（5）　熟悉测试管理中的量化指标。

[实训原理]

一、单元测试的内容

（1）模块接口测试 ：对通过被测模块的数据流进行测试。为此，对模块接口，包括参数表、调用子模块的参数、全程数据、文件输入 / 输出操作都必须检查。

（2）局部数据结构测试 ：设计测试用例检查数据类型说明、初始化、缺省值等方面的问题，还要查清全程数据对模块的影响。

（3）路径测试 ：选择适当的测试用例，对模块中重要的执行路径进行测试。对基本执行路径和循环进行测试可以发现大量的路径错误。

（4）错误处理测试 ：检查模块的错误处理功能是否包含有错误或缺陷。例如，是否拒绝不合理的输入；出错的描述是否难以理解、是否对错误定位有误、是否出错原因报告有误、是否对错误条件的处理不正确；在对错误处理之前错误条件是否已经引起系统的干预等。

（5）边界测试 ：要特别注意数据流、控制流中刚好等于、大于或小于确定的比较值时出错的可能性。对这些地方要仔细地选择测试用例，认真加以测试。

此外，如果对模块运行时间有要求的话，还要专门进行关键路径测试，以确定最坏情况下和平均意义下影响模块运行时间的因素。这类信息对进行性能评价是十分有用的。

二、　单元测试的步骤

通常单元测试在编码阶段进行。在源程序代码编制完成，经过评审和验证，确认没有语法错误之后，就开始进行单元测试的测试用例设计。利用设计文档，设计可以验证程序功能、找出程序错误的多个测试用例。对于每一组输入，应有预期的正确结果。

模块并不是一个独立的程序，在考虑测试模块时，同时要考虑它和外界的联系，用一些辅助模块去模拟与被测模块相联系的其它模块。这些辅助模块分为两种：

（1）驱动模块：相当于被测模块的主程序。它接收测试数据，把这些数据传送给被测模块，最后输出实测结果。

（2）桩模块：用以代替被测模块调用的子模块。桩模块可以做少量的数据操作，不需要把子模块所有功能都带进来，但不允许什么事情也不做。

被测模块、与它相关的驱动模块及桩模块共同构成了一个“测试环境”，如图 4-1 所示。

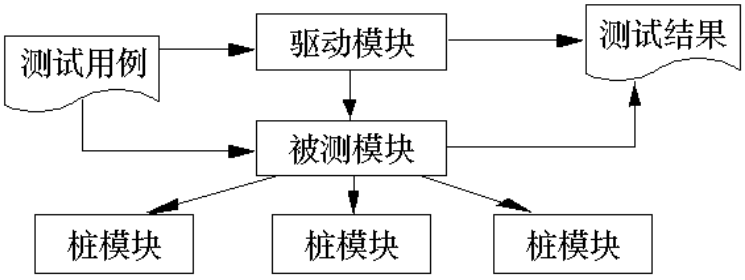


图 4-1 单元测试环境

[实训内容]

题目一：　针对三角形问题进行单元测试

三角形问题：接受三个正整数 a、b、c 作为输入，用做三角形的边。三边必须满足条件： 0<a、b、c≤1000；程序的输出是由这三条边确定的三角形类型：等边三角形、等腰三角形、不等边三角形或非三角形。该程序的实现已经用 c 语言实现，请参考附录 1（略）。

[实训步骤]

（1） 根据功能测试技术设计测试用例，主要考虑边界测试、等价类（弱一般、强一般、弱健壮性、强健壮性）和基于决策表等技术；

（2） 根据结构性测试技术设计测试用例，主要考虑路径测试、数据流等测试技术；

（3） 根据所学知识确定优化策略（原则：用最少的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；

（4） 根据设计的两套测试用例集进行测试、参照表 2 所示的缺陷等级给出缺陷列表；

（5） 计算测试用例的分支覆盖率、条件覆盖率和语句覆盖率等测试管理指标；

[实训要求]

（1） 以实验报告的形式撰写单元测试的测试用例。

（2） 实验结果要求给出两套测试用例集测试效果比较（根据实验步骤中的 4、5 两步进行撰写）；

[实训思考]

（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；.

为了高效地进行软件测试，目前还有哪些测试技术可以使用？

实训五　QuickTest Professional 初级使用

[实训目的]

了解 QuickTest 测试工具的操作界面，了解 QuickTest 测试工具的测试模式和过程，并能使用 QuickTest 测试工具录制测试脚本、执行并分析测试脚本。

[实训原理]

Mercury QuickTest Professional 是一款先进的自动化测试解决方案，用于创建功能和回归测试。它自动捕获、验证和重放用户的交互行为。使用 QuickTest Professional 关键字视图、自动文档（Auto-documentation）和活动屏幕（Active Screen），无需一行代码，就可以创建和修改测试脚本，同时满足了技术型和非技术型用户的需求，让各个公司有能力部署更高质量的应用。

QuickTest 主要应用在回归测试中。QuickTest 针对的是 GUI 应用程序，包括传统的 Windows 应用程序，以及现在越来越流行的 Web 应用。

一、QuickTest 窗口

在开始录制测试脚本之前，先熟悉 QuickTest 的窗口。QuickTest 的主窗口如图 5-1 所示。

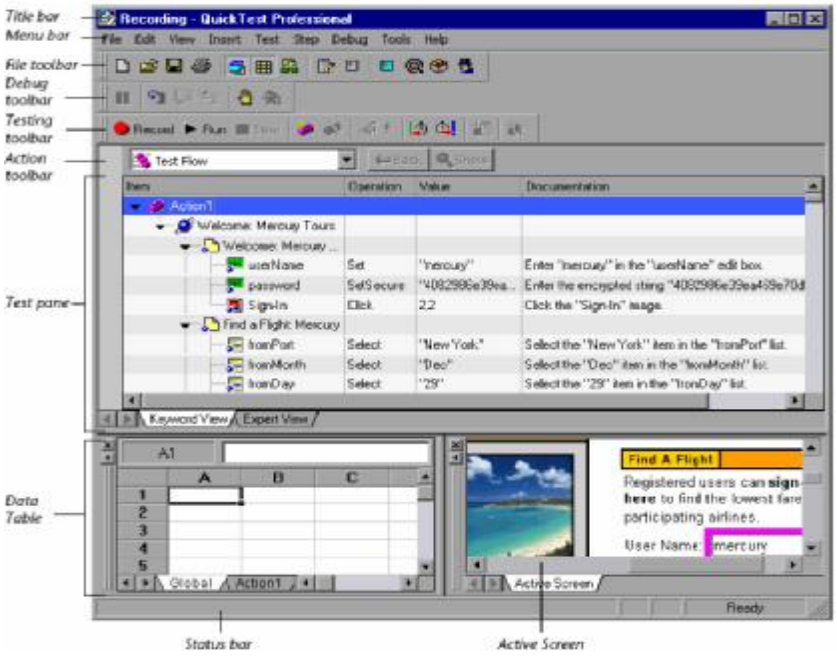


图 5-1 QuickTest 的主窗口

QuickTest 的主窗口包含下列的组件：

- Title bar：显示目前测试脚本的名称。
- Menu bar：显示 QuickTest 的菜单。
- File toolbar：管理测试脚本常用的工具列。File toolbar 如图 5-2 所示。

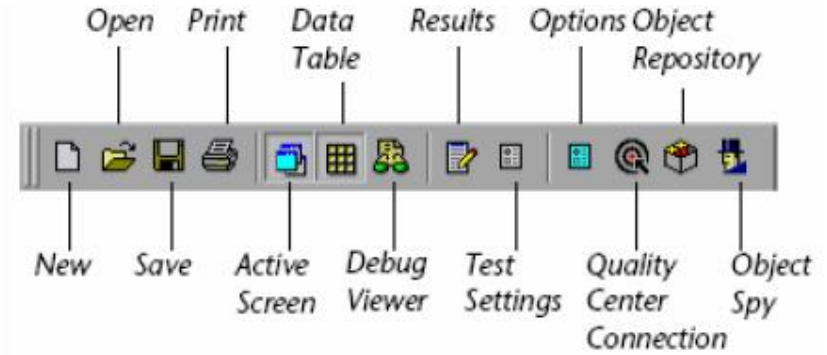


图 5-2File toolbar

- Test toolbar：录制测试脚本常用的工具列。Test toolbar 如图 5-3 所示。

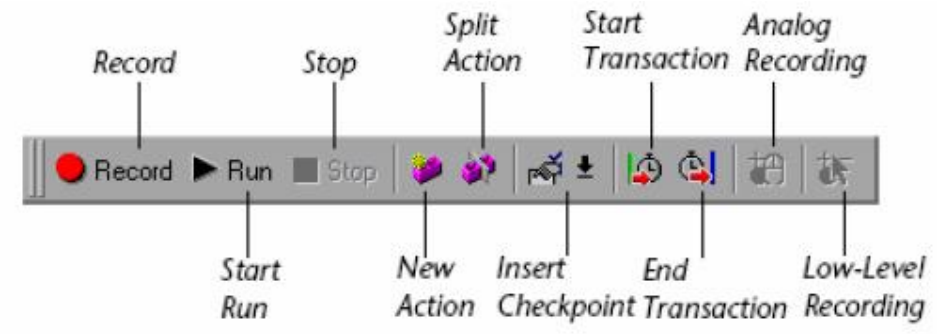


图 5-3 Test toolbar

- Debug toolbar：对测试脚本除错常用的工具列。Debug toolbar 如图 5-4 所示。

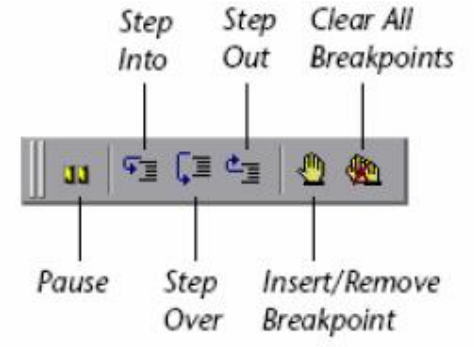


图 5-4 Debug toolbar

- Action toolbar：包含常用的功能按钮，以及一个显示测试动作（action）的下拉式清单（list），方便你检视整个测试脚本中的测试动作。
- Test pane：包含 Keyword View 以及 Expert View。
- Active Screen pane：包含 Active Screen。
- Data Table：当你对测试脚本做参数化时存放参数数据的地方，除了一个 Global 的数据表外，每一个 action 也会有各自的资料表。
- Debug Viewer pane：协助你对测试脚本除错（debug）。Debug Viewer pane 包含 WatchExpressions、Variables 以及 Command。
- Status bar：显示测试脚本的状态。

二、QuickTest 测试开发流程

QuickTest 的测试流程包含七大阶段：

1. 录制测试脚本前的准备

在测试前需要先确认应用程序以及 QuickTest 是符合测试需求。

确认你已经知道如何对应用程序进行测试，例如要测那些功能、操作步骤、输入的数据、预期的结果等。

同时应该检查一下 QuickTest 的设定，如 Test Settings（【Test】>【Settings】）以及 Options 对话窗口（【Tools】>【Options】），以确保 QuickTest 会适切的录制并储存信息。例如，你应该确认一下 QuickTest 的 Object Repository 是以什么模式储存信息的。

2. 录制测试脚本

当浏览网站或是操作应用程序时，QuickTest 会在 Keyword View 中以表格的方式显示录制的操作步骤。每一个操作步骤都是使用者在录制时的操作，如在网页上点选一个超级链接（link），或是按下窗口上的按钮。

3. 加强测试脚本

- 在测试脚本中加入检查点，可以检查网页超级链接、对象属性或是字符串，以验证应用程序的功能是否正确。
- 将录制的固定值（hard code）参数以取代，使用多组的数据测试应用程序。
- 使用逻辑（logic）或是条件（conditional）判断式，可以进行更复杂的测试。

4. 调试脚本（对测试脚本除错）

在修改过测试脚本之后，需要调试测试脚本，检查脚本是否存在错误，以确保测试脚本能正常且流畅的执行。

5. 在应用程序或网站上执行测试脚本

在对应用程序或网站的回归测试中，通过 QuickTest 回放对应用程序或网站的操作，检验软件正确性，实现测试的自动化进行。

6. 分析测试结果

查看 QuickTest 记录的运行结果，分析测试执行的结果，记录问题，找出应用程序的问题所在。

7. 回报问题（defect）

如果安装了 Quality Center（TestDirector），则你可以将发现的问题回报到 QualityCenter（TestDirector）的数据库中。Quality Center（TestDirector）是 Mercury 的测试管理工具。

[实训内容]

1．题目一：测试 MercuryTours 网站

使用 QuickTest 对 MercuryTours 网站进行功能测试。要求录制预订机票的完整过程，然后执行测试脚本并分析结果。

2．题目二：测试 163 网站

使用 QuickTest 对 MercuryTours 网站进行功能测试。要求录制打开 163 免费邮箱阅读邮件和发邮件的过程。然后执行测试脚本并分析结果。

[实训步骤]

一、录制脚本准备

当你浏览网站或使用应用程序时，QuickTest 会纪录你的操作步骤，并产生测试脚本。 当你停止录制测试脚本后，会看到 QuickTest 在 Keyword View 中以表格的方式显示测试脚本的操作步骤（steps）。

在测试前需要先确认你的应用程序以及 QuickTest 是符合你的测试需求的。

在开始时请先确认以下事项：

假如你是使用 Internet Explorer 浏览器，请你先取消「自动完成」的功能（取消「自动完成」的设

定：

1. 开启 Internet Explorer 浏览器，点选【工具】>【因特网选项】>【内容】。
2. 点选【个人信息】中的【自动完成】按钮，开启【自动完成设定】对话窗口。
3. 在【使用「自动完成」】取消【窗体上的使用者名称和密码】选项。)。

关闭所有的浏览器窗口。

二、录制脚本

录制一个测试脚本。在 Mercury Tours 范例网站上预订一张从纽约（New York）到旧金山（San Francisco）的机票。

1. 执行 QuickTest 并开启一个全新的测试脚本

要开启 QuickTest，请点选【开始】>【程序集】>【QuickTest Professional】>【QuickTest Professional】。在【Add-in Manager】勾选【Web Add-in】，并取消其它的 add-ins。然后点选【OK】按钮关闭【Add-in Manager】窗口，进入 QuickTest Professional 主窗口。

假如出现【Welcome】窗口，点选【Blank Test】。或者，点选【File】>【New】，或是按下工具列上的【New】按钮。QuickTest Professional 会开启全新的测试脚本档案。假如 QuickTest Professional 已经开启，检查【Help】>【About Quick Test Professional】看目前加载了哪些 add-ins 。

2. 开始录制测试脚本

点选【Test】>【Record】或是点选工具列上的【Record】按钮。会开启【Record and Run Settings】对话窗口。在【Web】页签，勾选【Open the following browser when a record or run session begins】。 从【Type】下拉列表选择使用的浏览器，并且在【Address】输入 http://newtours.mercuryinteractive.com。

请确认【Do not record and run on browsers that are already open】与【Close the browser when the test is closed】这二个选项都已经勾选了，如图 5-5 所示。

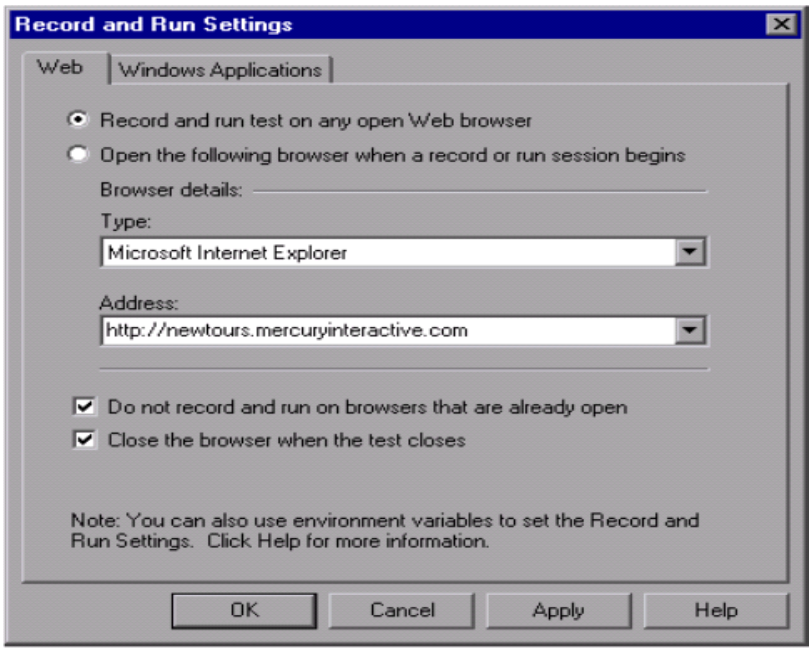


图 5-5 Web 选项

在【Windows Applications】页签，勾选【Record and run on these app（opened on session start）】，而且不要选取任何的应用程序。 此设定可以避免录制到其它应用程序（如 Outlook）的操作。如图 5-6 所示。

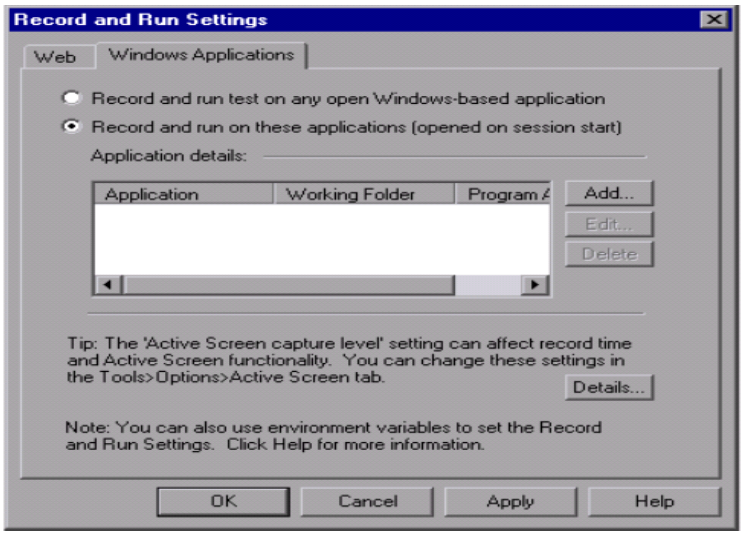


图 5-6 Windows Application 选项

点选【OK】。 QuickTest 会开启浏览器浏览 Mercury Tours 网站，并且开始录制测试脚本。

登入 Mercury Tours Web site 网站，在【User Name】与【Password】输入你当初注册的账号与密码。点选【Sign-In】，开启【Flight Finder】网页。 输入订票数据， 选择飞机航班 。点选【CONTINUE】按钮，开启【Book a Flight】页面， 输入必要字段 ，在【Billing Address】勾选【Ticketless Travel】。 按下网页下方的【SECURE PURCHASE】按钮，开启【Flight Confirmation】网页。 检视订票数据，并点选【BACK TO HOME】回到 Mercury Tours 网站的首页。

3. 停止录制

在 QuickTest，点选工具列上的【Stop】按钮，停止录制。现已经完成了预定「纽约-旧金山」机票的动作，QuickTest 已经录制了从按下【Record】按钮后，到按下【Stop】按钮之间所有的操作。

4. 储存测试脚本

选取【File】>【Save】或是点选工具列上的【Save】按钮，开启【Save】对话窗口。建立一个【Tutorial】目录，将测试脚本命名为【Recording】。 勾选【Save Active Screen files】。

按下【存盘】按钮，测试脚本名称（Recording）会出现在 QuickTest 窗口的标题列。

三、分析 Keyword View 中的测试脚本

录制测试脚本时，QuickTest 会将每一个操作录制下来，并在 Keyword View 类似 Excel 工作表的方式显示所录制的测试步骤。可以点选【View】>【Expend All】检视测试脚本的每一个步骤，如图 5-7 所示。

在 Keyword View 的中每个字段都有其意义：

【Item】：以阶层式的图标显示这个操作步骤所作用到的组件（测试对象（test object）、工具对象（utility object）、函数呼叫（function call）或脚本（statement））。

【Operation】：要在这个作用到的组件上执行的动作，如点选（Click）、选取（Select）。

【Value】：执行动作的参数（argument），例如当鼠标点选一张图片时是用左键还是右键。

【Assignment】：使用到的变量。

【Comment】：你在测试脚本中加入的批注。

【Documentation】：自动产生用来描述此操作步骤的英文说明。

脚本中的每一个步骤在 Keyword View 中都会以一列来显示，其中包含用来表示此组件类别的图标以及此步骤的详细数据。

[实训要求]

（1） 撰写实验报告，主要填写本人测试步骤和自己的实验体会。

（2） 提交录制的测试脚本。

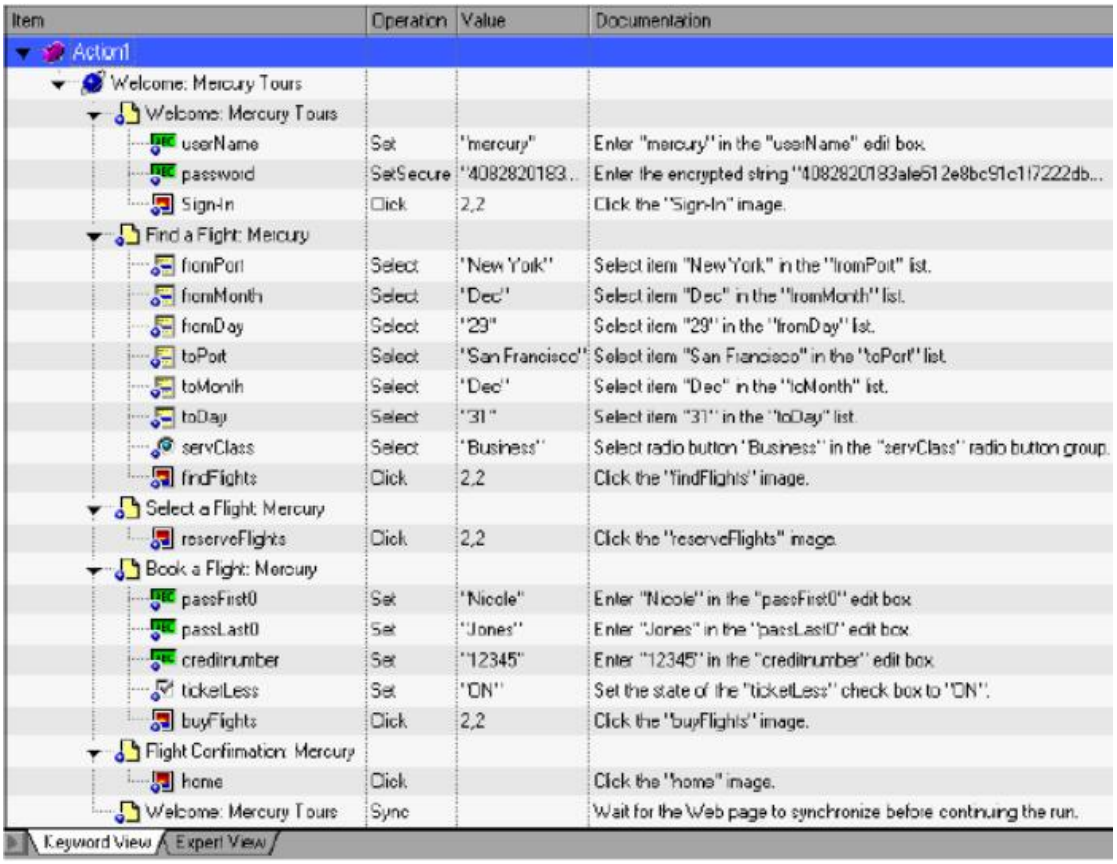


图 5-7　Keyword View 视图

实训六　QuickTest Professional 高级使用

[实训目的]

熟练使用 QuickTest Professional 进行自动化测试。掌握 QuickTest Professional 测试流程。

[实训原理]

一、创建检查点

1. 检查点类型

QuickTest Professional 提供的检查点如表 5-1 所示。

表 5-1 检查点类型

检查点类型	描述	用法示例
标准检查点	检查对象的属性值。	检查是否选中某单选按钮。
图像检查点	检查图像的属性值。	检查图像源文件是否正确。
表检查点	检查表中的信息。	检查表单元格中的值是否正确。
页面检查点	检查网页的特性。	检查加载网页所需的时间，或者检查网页是否包含中断链接。
文本/文本区域检查点	检查文本字符串是否显示在网页或应用程序窗口中的适当位置。	检查预期的文本字符串是否显示在网页或对话框上的预期位置。
位图检查点	将网页或应用程序的某个区域捕获为位图后对其进行检查。	检查网页或网页的任何部分是否能按预期显示。
数据库检查点	检查应用程序或网站所访问的数据库内容	检查数据库查询中的值是否正确。
可访问性检查点	对网站区域进行识别，以检查是否符合 508 部分。	检查网页上的图像是否包含 ALT 属性（该属性是 W3C Web 内容可访问性规则所要求的）。
XML 检查点	检查 XML 文档的数据内容。	**注意：**XML 文件检查点用于检查特定的 XML 文件；XML 应用程序检查点用于检查网页中的 XML 文档。

大多数检查点都可以在录制过程中或在录制之后添加到测试中。

2. 检查对象

本部分将在“Book a Flight”页中添加标准检查点。该检查点将验证包含乘客名字的框中的值。

执行 QuickTest 并开启「Recording」测试脚本。将测试另存为“Checkpoint”。

找到要向其添加标准检查点的页面。添加检查点，以便在乘客的名字输入到“First Name”编辑框后，对该框中的属性值进行检查。在关键字视图的“项”列中，单击 (+) 展开“Action1”>“Welcome:Mercury Tours”>“Book a Flight: Mercury”。

创建标准检查点。在 Active Screen 中，右键单击“First Name”框，然后选择“插入标准检查点”。将

打开“对象选择—检查点属性”对话框，如图 6-1 所示。

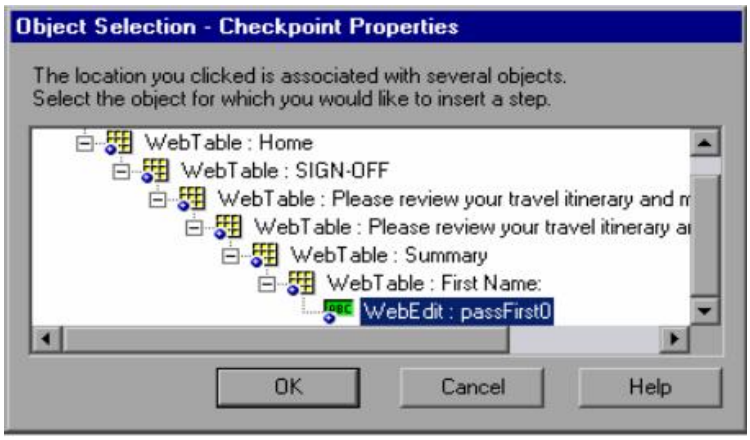


图 6-1 对象选择—检查点属性

确认已突出显示“WebEdit: passFirst0”，然后单击“确定”。 将打开“检查点属性”对话框，如图 6-2 所示。

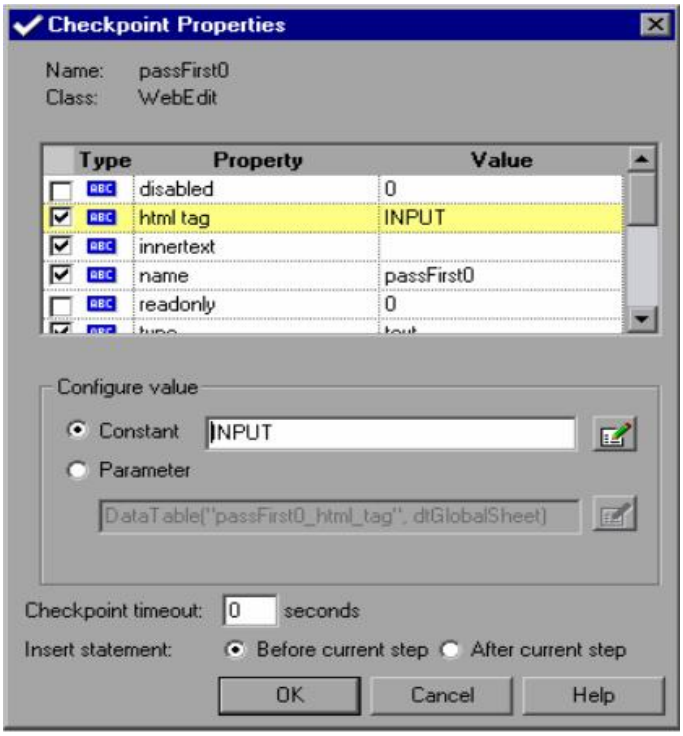


图 6-2 检查点属性

该对话窗口会显示对象的属性:【name】是这个对象的名称 【Class】是这个对象的类别。WebEdit 表示这个对象是个 edit box。 在【Type】字段中的【ABC】图标表示这个属性的值是个常数。

对每个对象，QuickTest 会使用预设的属性作为检查的属性。接受预设的设定值，然后点选【OK】。QuickTest 会在你选取的步骤之前建立一个标准的检查点。

3. 检查页面

网页检查点会检查网页的链结（link）以及图片的数量是否与当初录制时的数量一样。

在 Keyword View 中，展开(+)【Action1】>【"Welcome: Mercury Tours"】。 选取 Keyword View 中的【"Book a Flight: Mercury"】网页。在【Active Screen】会显示这个网页的画面。

在【Active Screen】上任意地方按下鼠标右键，选取【Insert Standard Checkpoint】，会开启【Object Selection – Checkpoint Properties】对话窗口，如图 6-3 所示。由于你点选的位置不同，对话窗口显示被选取的对象可能会不一样。

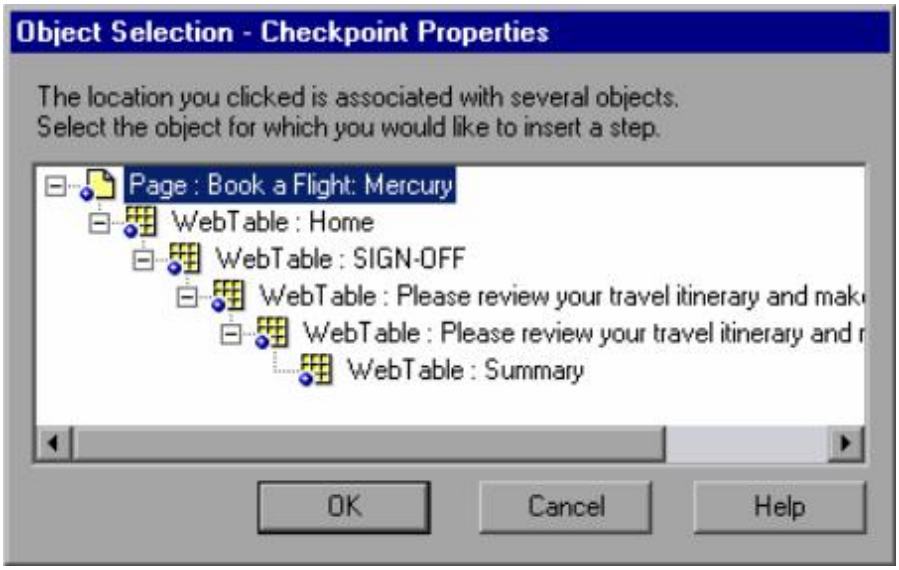


图 6-3 对象选择-检查点属性

点选【Page: Book a Flight: Mercury】（在最上层）然后点选【OK】。接着开启【Page Checkpoint Properties】对话窗口。当执行测试时，QuickTest 会检查网页的连结与图片的数量，以及加载的时间。QuickTest 也检查每个 link 的 URL 以及每个图片的原始文件是否存在。

接受默认值，点选【OK】。QuickTest 会在 Book a Flight: Mercury 网页下方加上一个网页检查点。

点选【File】>【Save】或是点选工具列上的【Save】按钮保存测试脚本。

4. 检查文本

在此建立一个文本检查点，检查在【"Flight Confirmation"】网页中是否出现【"New York"】。

在 Keyword View 中，展开(+)【Action1】>【"Welcome: Mercury Tours"】。选取 Keyword View 中的【"Flight Confirmation: Mercury"】网页。在【Active Screen】会显示网页的画面。

在【Active Screen】中，选取在「Departing,」下方的「New York」。对选取的文字按下鼠标右键，点选【Insert Text Checkpoint】开启【Text Checkpoint Properties】对话窗口，如图 4-12 所示。

当【Checked Text】出现在下拉式清单中时，在【Constant】字段会显示刚刚选取的文字。也就是 QuickTest 在执行测试脚本时所要检查的文字。 点选【OK】关闭对话窗口。Quick Test 会在测试脚本上加上一个文字检查点，这个文字检查点会出现在【"Flight Confirmation: Mercury"】网页下方。

点选【File】>【Save】或是点选工具列上的【Save】按钮保存测试脚本。

5. 检查表格

建立一个表格检查点，检查【Book a Flight: Mercury】网页上出国航班的价钱。

在 Keyword View 中，展开(+)【Action1】>【"Welcome: Mercury Tours"】。选取 Keyword View 中的【"Book a Flight: Mercury"】网页。在【Active Screen】会显示网页的画面。

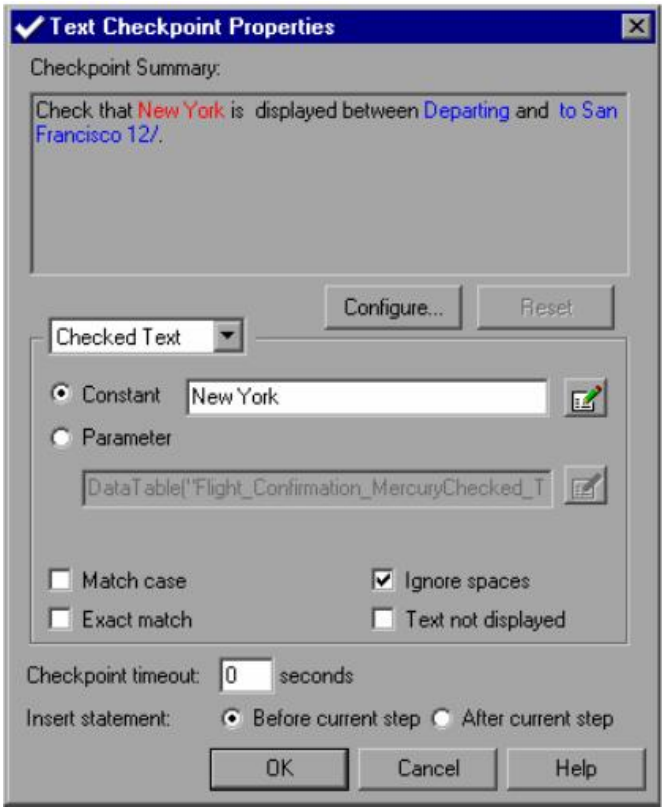


图 6-4 文本检查点属性

在【Active Screen】中，在第一个航班的价钱上（New York to San Francisco）—270—点选鼠标右键，然后选取【Insert Standard Checkpoint】。会开启【Object Selection – Checkpoint Properties】对话窗口。

一开始被选取的是 WebElement: 270，这时要点选上一层的 WebTable 对象，选取 Web Table : New York to San Fransisco。

注意，这时会看到在【Active Screen】上，选取的表格也会被框起来。点选【OK】。

接着开启【Table Checkpoint Properties】对话窗口，显示整个表格的内容。

预设是每个字段都会被勾选，表示所有字段都会做检查。你可以对某个字段点二下，取消检查字段，或是选取整个栏或列，执行选取或取消的动作。

在每个字段的标题列点二下，取消勾选的图示，然后在第 3 行、第 3 列点二下，会在 270 左边出现勾选的图示，则执行时 QuickTest 会对此字段值做检查。

点选【OK】关闭对话窗口。QuickTest 会在测试脚本中，【"Book a Flight: Mercury"】页面下加上一个表格检查点。

点选【File】>【Save】或是点选工具列上的【Save】按钮保存测试脚本。

6. 使用检查点运行并分析测试

使用检查点查看测试，运行测试，并分析检查点结果。

1) 展开并查看测试。

选择“视图”>“全部展开”或使用数字键盘上的“*”快捷键。所显示的关键字视图与图 6-5 相似。

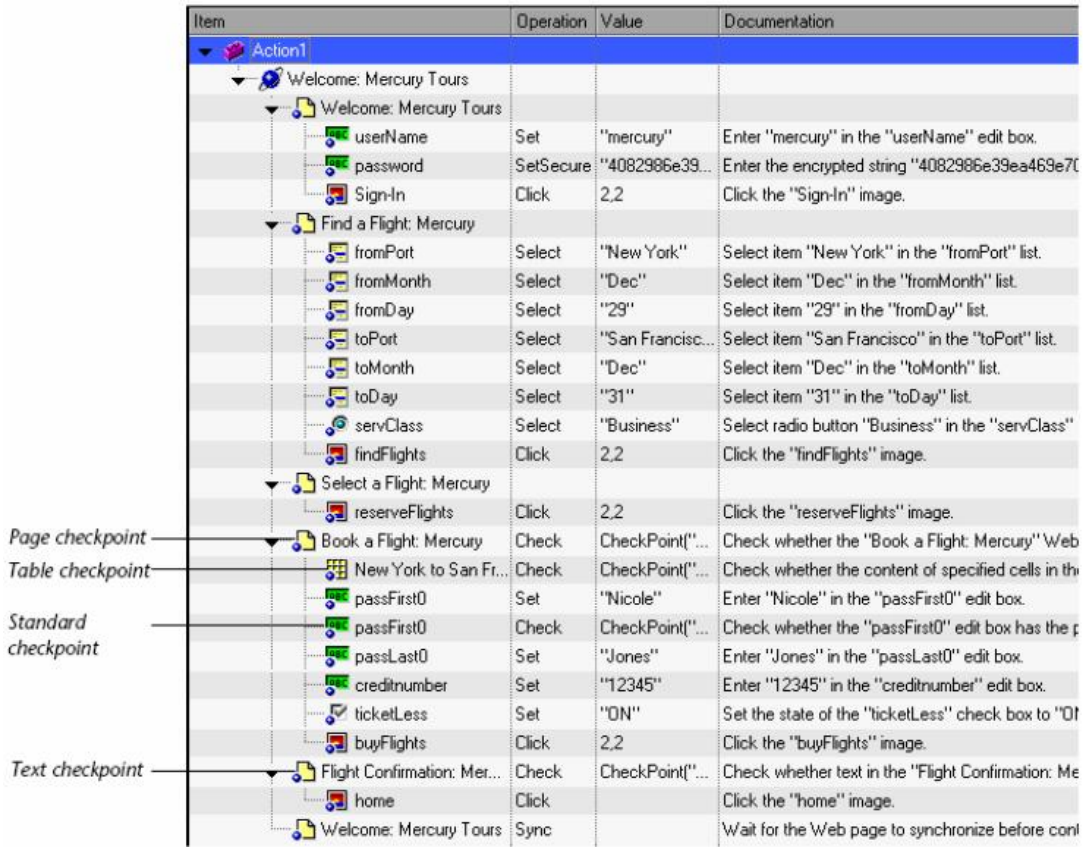


图 6-5 关键字视图

2) 开始运行测试。

单击“运行”或选择“测试”>“运行”。将打开“运行”对话框。请确保已选定“新建运行结果文件夹”。接受默认的结果文件夹名。单击“确定”。当完成测试运行时，将打开“测试结果”窗口。

3) 查看测试结果。

当 QuickTest 完成运行测试时，将打开“测试结果”窗口。测试结果应该为“通过”，表示所有检查点已通过测试。如果一个或多个检查点失败，则测试结果将为“失败”。

4) 查看页面检查点的结果。

在结果树中，单击 (+) 展开“Checkpoint Iteration 1 (Row 1)”> “Action1 Summary”> “Welcome: Mercury Tours”>“Book a Flight: Mercury”。 突出显示“检查点`Book a Flight: Mercury'”。

在“详细信息”窗格中列出了已检查的项目，可以查看页面检查点的详细信息。

检查点通过，因为已检查的实际属性值与预期值相匹配。

5) 查看表检查点的结果。

在结果树中的“Book a Flight:”页上，单击 (+) 展开“New York to San Francisco”。突出显示“检查点`New York to San Francisco'”。

在“详细信息”窗格中，可以查看表检查点的详细信息。还可以查看表单元格的值（已检查的单元格值显示为黑色；未检查的单元格值显示为灰色）。

6) 查看标准检查点的结果。

在结果树中的“Book a Flight: Mercury”页上，单击 (+) 展开“passFirst0”。

突出显示“检查点 passFirst0”。在“详细信息”窗格中列出了已检查的属性及其值，可以查看标准检查点的详细信息。

7) 查看文本检查点的结果。

在结果树中，单击 (+) 展开“Checkpoint Iteration1 (Row 1)”> “Action1 Summary”>“Welcome: Mercury Tours”>“Flight Confirmation: Mercury”。突出显示“检查点`New York'”。

在“详细信息”窗格中，可以查看文本检查点的详细信息。

8) 关闭“测试结果”窗口。

二、参数化测试

当您测试应用程序时，可能希望检查该应用程序用多组数据来执行相同操作的方式。例如，假设您希望检查网站响应十组单独数据的方式。您可以录制十项单独的测试，每项测试都使用自己的一组数据。或者，您也可以创建数据表参数，以便将测试运行十次，而每次运行都使用不同的一组数据。

1．定义数据表参数

在之前录制的测试脚本预订了从纽约到旧金山的机票，在测试脚本中，纽约是个常数值，也就是说，每次执行测试脚本预定机票时，出发地点都是纽约。在此将会学习如何将测试脚本中的出发地点参数化，如此一来，执行测试脚本时就会以不同的出发地点去预订机票了。

1) 执行 QuickTest 并开启「Checkpoint」测试脚本

2) 将测试脚本另存成名为「Parameter」测试脚本

3) 确认【Active Screen】是开启的

4) 选取要参数化的文字

在 Keyword View 中，展开(+)【Action1】>【"Welcome: Mercury Tour"】>【"Find a Flight: Mercury"】。在 Keyword View 中点选"fromPort"右边的【Value】字段，然后再点选参数化，如图 6-6 所示 。

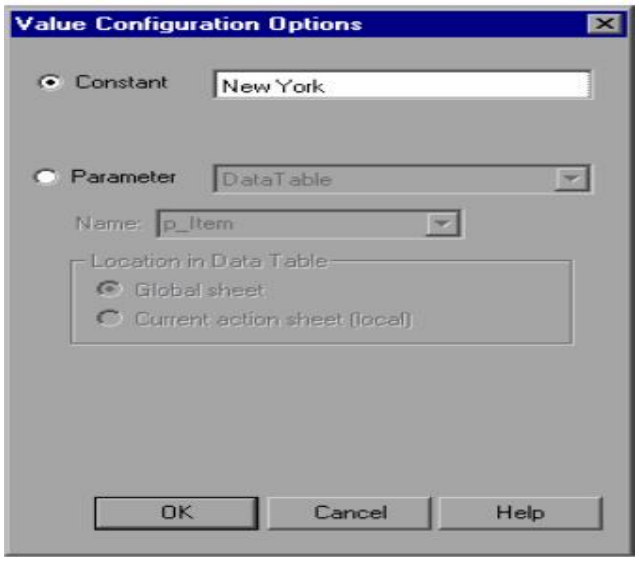


图 6-6 参数值配置选项

点选  图标，会开启【Value Configuration Options】对话窗口。

5) 设定要参数化的属性

点选 Parameter。如此一来，就可以使用参数值来取代纽约（New York）这个常数值。请选择【Data Table】这个选项，这个选项表示此参数的值会从 QuickTest 的 Data Table 中取得。而且【Name】字段会出现 p_Item，请将其修改成 departure。 如图 4-15 所示。

点选【OK】关闭窗口。QuickTest 会在 Data Table 中新增 departure 参数字段，并且插入一行 New York 的值。则 New York 会成为测试脚本执行时所用的第一个值。

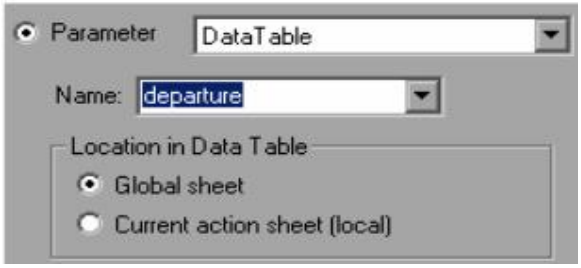


图 6-7 参数值设置

2．在数据表中输入参数

QuickTest 会在 Data Table 中显示参数值。可以在 Data Table 中加入更多笔出发地点的资料，让 QuickTest 可以使用这些资料执行测试脚本。

1) 在 departure 字段输入更多数据

2) 储存测试脚本

请特别注意一下在 Keyword View 中的变化。在参数化之前，此测试步骤是显示【"fromPort" Select " New York"】。现在，这个测试步骤变成了【fromPort Select Data Table("departure", DTGlobalSheet)】。而且当您点选【Value】字段时，【Value】字段会变成 <departure>，表示此测试步骤已经被参数化了，而且其值是从 Data Table 中的 departure 字段中取得。

3．修改受参数化影响的步骤

参数化测试中的某一步骤后，在更改参数化的步骤的值时其他测试对象可能会受到影响。如果发生这种情况，您必须修改这些对象的预期值以匹配从参数化步骤中生成的值。在本部分中，将修改文本检查点，以便在运行测试时，QuickTest 检查与当前出发城市相匹配的文本。

1) 定位要修改的文本检查点。

在关键字视图中，单击 (+) 展开“Welcome:Mercury Tours”。

右键单击“Flight Confirmation:Mercury”，然后选择“检查点属性”。将打开“文本检查点属性”对话框，如图 6-8 所示。

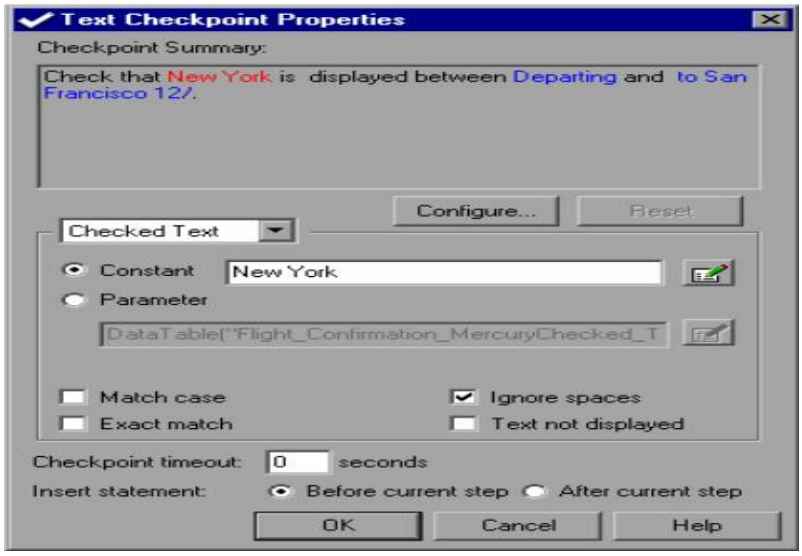


图 6-8 文本检查点属性

2) 参数化文本检查点。

在“已检查的文本”区域，“New York”显示在“常量”框中。“New York”是每一次循环时检查点的预期值。

选择“参数”，然后单击“参数选项”按钮 。将打开“参数选项”对话框，如图 6-9 所示。

5-9 参数选项

在“名称”框中，选择“departure”。这将指示检查点使用数据表中的 departure 参数值作为预期结果。单击“确定”关闭“参数选项”对话框，然后再次单击“确定”关闭“文本检查点属性”对话框。现已将该检查点参数化。

3) 保存测试。

4. 运行并分析使用参数的测试步骤

执行修改完成后的「Parameter」测试脚本，QuickTest 会使用 Data Table 中 departure 字段的值，执行三次测试脚本。

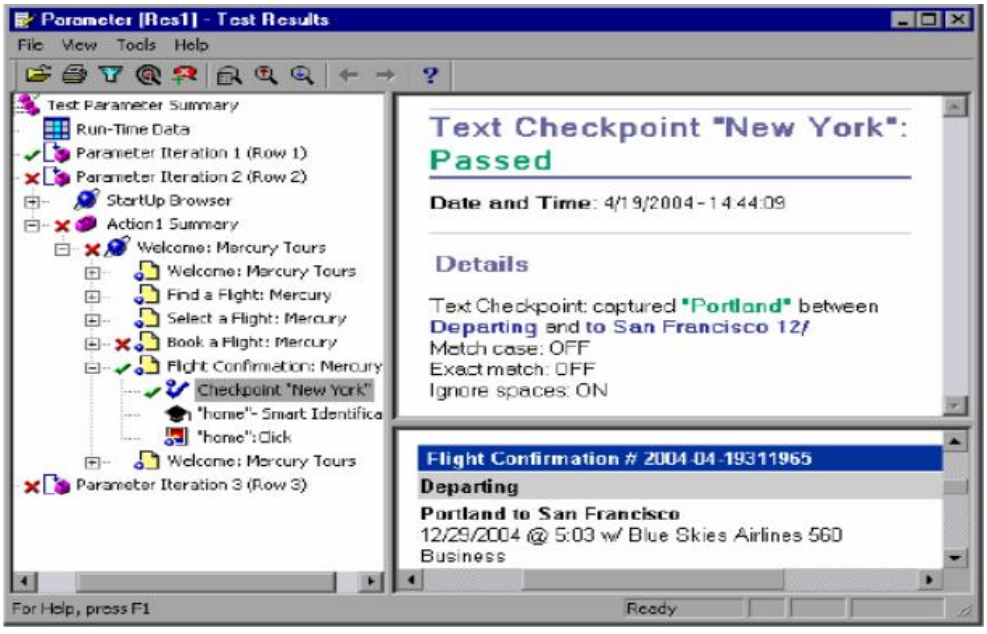


图 6-10 测试结果

1) 执行「Parameter」测试脚本

2) 检查测试结果

即使文本检查点在三次循环中都通过，“测试结果”窗口也会显示测试的第二次和第三次循环失败。有关循环失败原因的详细信息，请参阅以下内容。

循环 2： 在 results tree 中，展开(+)【Parameter Iteration 2】>【Action1 Summary】>【Welcome Mercury Tours】>【Flight Confirmation: Mercury】。选取【Checkpoint:"New York"】。如图 4-18 所示。

在检查点的【Details】窗格中，显示 Portland 为预期结果同时也是实际值，所以此文字检查点为通过。你也可以看到在下方的【Application】窗格中，显示机票的出发地点也是 Portland。

循环 3（Iteration 3）： 在 results tree 中，展开(+)【Parameter Iteration 3】>【Action Summary】>【Welcome Mercury Tours】>【Flight Confirmation: Mercury】。 选取【Checkpoint:"New York"】。

在检查点的【Details】窗格中，显示 Seattle 为预期结果同时也是实际值，所以此文字检查点为通过。你也可以看到在下方的【Application】窗格中，显示机票的出发地点也是 Seattle。

每次执行时，此文字检查点的结果如表 5-1 所示。

Iteration	Expected Text	Actual Text	Result
#1	New York	New York	Checkpoint Passed
#2	Portland	Portland	Checkpoint Passed
#3	Seattle	Seattle	Checkpoint Passed

表 5-1 文本检查点测试结果

注意：虽然每次执行时，文字检查点的结果是通过的，但是第二次与第三次的执行结果仍然为失败。这是因为出发地点改变，造成在表格检查点中的机票价钱改变，导致表格检查点失败。在以后将会修正表格检查点，让 QuickTest 自动更新表格检查点的预期结果，就可以检查正确的票价了。

3) 关闭测试结果窗口

[实训设备]

主流 PC 机一套，要求安装 windows 操作系统、Quick Test Professional8.2、OFFICE 工具；

[实训内容]

1．题目一：创建检查点

在前一个实验（QuickTestProfessional 初级使用）中录制的脚本中创建检查点，包括检查对象、检查页面、检查文本、检查表格。然后执行测试脚本，并分析测试脚本。

2．题目二：参数化测试

使用前一个实验（QuickTestProfessional 初级使用）录制的脚本，进行参数化测试。

[实训步骤]

在上一个实验中，创建并运行了测试，以检查在应用程序上或网站上执行的一系列步骤是否可以顺利执行。本实验要求对上一个实验做增强测试。通过在测试中插入检查点可以搜索页面、对象或文本字符串中的特定值，有助于确定应用程序或网站是否正常运行。通过扩大测试范围（用参数替换固定值），

可以检查应用程序如何使用多组数据来执行相同的操作。 本实验需测试步骤如下：

1. 检查对象：选取一个对象，创建标准检查点，检查该对象的属性值；
2. 检查文本：选取文本，创建文本检查点；
3. 检查表格：选取一个表格，检查表中的信息；
4. 参数化测试：选取可参数化的文本或对象，进行参数化测试。

[实训要求]

1. 撰写实验报告，主要填写本人上机测试步骤和测试内容；
2. 提交录制的测试脚本。

实训七　WinRunner 的使用

[实训目的]

了解 WinRunner 的安装过程，并进行安装实验。了解 WinRunner 测试模式和测试过程，并能够使用 WinRunner 进行简单的测试工作。

[实训原理]

WinRunner 是一种企业级的用于检验应用程序是否如期运行的功能性测试工具。通过自动捕获，检测，和重复用户交互的操作，WinRunner 能够辨认缺陷并且确保那些跨越多个应用程序和数据库的业务流程在初次发布就能避免出现故障，并且保持长期可靠运行。

WinRunner 的测试过程分六个步骤： 创建 GUI map、创建测试、调试测试、执行测试、查看测试结果、报告发现的错误。

一、 创建 GUI map

使用 RapidTest Script wizard(快速测试脚本巫师)回顾软件用户界面，并系统地把每个 GUI 对象的描述添加到 GUI map 中。也可以在录制测试的时候，通过点击对象把对单个对象的描述添加到 GUI map 中。

二、 创建测试

可以通过录制、编程或两者同用的方式创建测试脚本。录制测试时，在你需要检查软件反应的地方插入检查点（Checkpoint）。插入检查点来检查 GUI 对象，位图(Bitmap)和数据库。在这个过程中，WR 捕捉数据，并作为期望结果（被测软件的期望反应）储存下来。

三、 调试测试

可以先在调试模式（Debug mode）下运行脚本。也可以设置中断点(Breakpoint)，监测变量，控制 WR 识别和隔离错误。调试结果被保存在 Debug folder，一旦调试结束就可以删除。

四、 执行测试

在检验模式(Verify mode)下测试被测软件。WR 在脚本运行中遇到检查点后，就把当前数据和前期捕捉的期望值进行比较。如果发现有不符合，就记录下来作为实测结果。

五、 查看测试结果

测试是成功还是失败由你来认定。每次测试结束，WR 会把结果显示在报告中。报告会详述测试执行过程中发生的所有主要事件，如检查点、错误信息、系统信息或用户信息。

如果在检查点有不符合被发现，可以在 Test Results（测试结果）窗口查看预期结果和实测结果。如果是位图不符合，也可以查看用于显示预期值和实测结果之间差异的位图。

六、 报告发现的错误

如果由于测试中发现错误而造成测试运行失败，可以直接从 Test Results 窗口报告有关错误的信息。这些信息通过 EMAIL 发送给测试经理（QA Manager），用来跟踪这个错误直到被修复。

[实训内容]

1．题目一：测试 MercuryTours 网站

使用 WinRunner 对 Flight Reservation 范例程序进行功能测试。要求录制预订机票的完整过程，然后执行测试脚本并分析结果。

2．题目二：测试 Windows 应用程序

选择一个 Windows 应用程序。要求录制 此应用程序操作过程，然后执行测试脚本并分析结果。

3．题目三：测试 163 网站

使用 QuickTest 对 MercuryTours 网站进行功能测试。要求录制打开 163 免费邮箱阅读邮件和发邮件

的过程。然后执行测试脚本并分析结果。

[实训步骤]

一、WinRunner 使用概述

1. 启动 WinRunner

点击开始>程序>WinRunner>WinRunner 启动 WR。WR 的 Record/Run Engine(记录/执行引擎)的图标出现在 Windows 的任务条上。这个引擎设立和维护 WR 和被测软件之间的连接。 第一次启动 WR 会看到欢迎窗口，你可以选择新建测试、打开已有测试或快速预览 WR。如果不希望下次启动看到这个窗口，可以把 Show on startup 前面的勾去掉。

2. WinRunner 主窗口

WinRunner 主窗口入图 7-1 所示。主窗口包括以下部分：

a). WinRunner title bar 标题栏

b). Menu bar 菜单栏

c). Standard toolbar 标准工具栏，包含运行测试时常用的命令

d). User toolbar 用户工具栏，包含创建测试时常用的命令

e). Status bar 状态栏

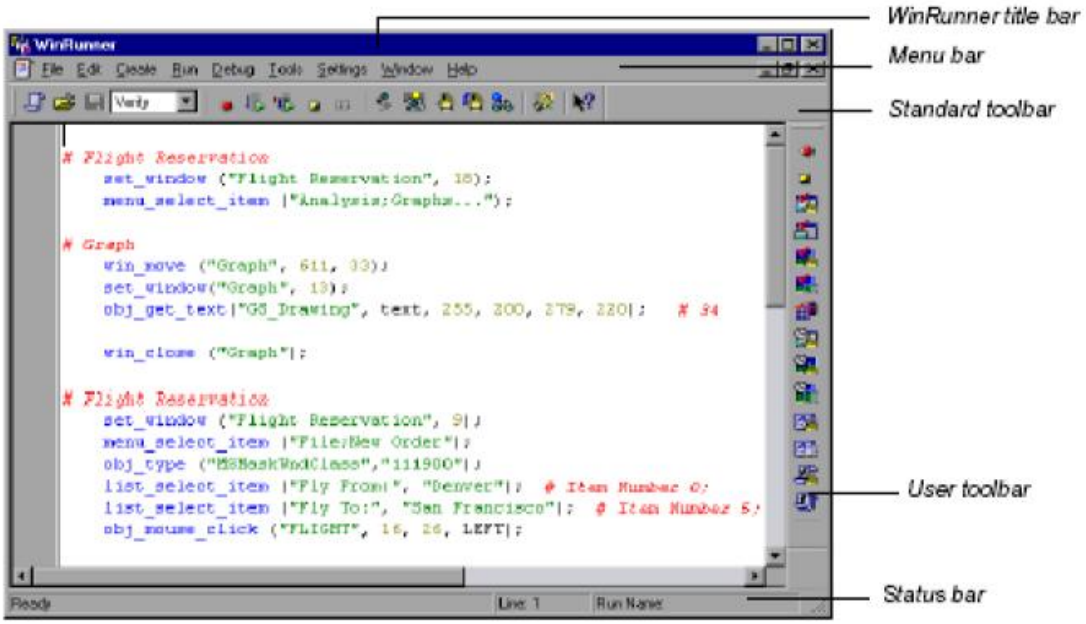


图 7-1 WinRunner 主窗口

3. 测试窗口

测试窗口如图 7-2 所示，在测试窗口创建和执行测试。窗口包含以下部分：

a). Test Window title bar 测试窗口标题栏，显示当前打开的测试名称

b). Test script 测试脚本，通过录制或编写代码方式生成

c). Execution arrow 执行箭头，指明当前正在执行的那一行脚本，如果想要移动这个标志到某一行，只需要在该行左侧空白处点击鼠标左键

d). Insertion point 插入点，支出你可以插入或编辑文本的地方

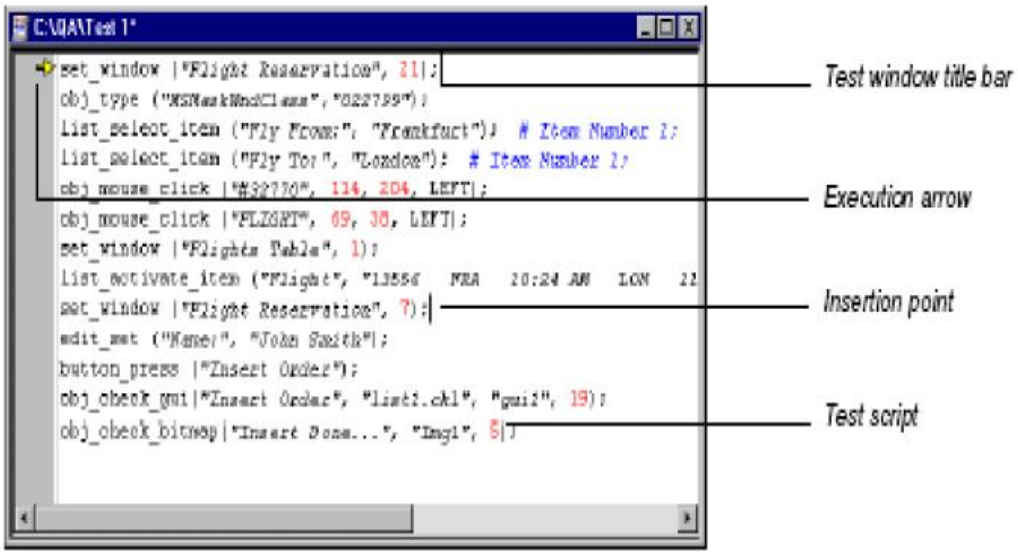


图 7-2 测试窗口

4．加载 WinRunner 插件

在 WinRunner 启动时，可以选择支持 ActiveX control、PowerBuilder、VisualBasic 或 WebTest 的插件。其他插件需要单独向 MI 公司购买，建议不要同时载入所有的插件，不必要的插件可能会对录制或执行脚步造成问题。 把 Show on startup 前面的勾去掉，这个 Add－In Manager 的窗口就不会在 WR 启动的时候出现。你也可以在进入 WR 后在 Settings>General Options>Enviroment 里面设置是否在开始时显示这个窗口以及等待时间等。

二、录制脚本

接下来你会以 Context Sensitive 模式录制一段测试脚本，此测试脚本的操作流程为在 FlightReservation 开启一笔订单。

1. 开启 WinRunner 并加载 GUI Map File 执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。

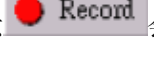
检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI MapEditor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 Flight Reservation 并登入执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

调整 WinRunner 与 Flight Reservation 的窗口大小与位置，让这二个窗口内容都可以清楚的倍看见。

3. 开始以 Context Sensitive 模式录制测试脚本

在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

按钮，从现在开始 WinRunner 会录制所有鼠标的点选以及键盘的输入。请注意 

会变成 

，蓝色的 Rec 会出现在按钮下方，表示现在已经进入 Context Sensitive 录制模式了。在 WinRunner 下方的状态列同样也会有变化，表示现在已经在录制测试脚本了。

4. 开启 3 号订单在 Flight Reservation 中点选【File】->【Open Order】，在 Open Order 窗口中点

选【OrderNo.】并且输入 3 后按下【OK】。

5. 停止录制
6. 储存测试脚本

三、执行脚本

当你完成上面的练习之后，你已经准备好执行测试脚本并分析测试结果了。WinRunner 提供三种执行测试脚本的模式：Verify、Debug、Update。

Verify：当你真正执行测试以检查应用软件的功能，并且要储存测试结果。

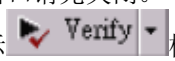
Debug：当你想检查测试脚本执行是否流畅，没有错误时。

Update：当你要更新检查点的预期值时。

执行：

1. 确认 WinRunner 与 Flight Reservation 的主窗口都已经开启
2. 开启 loeeson4 测试脚本

   先点选【File】->【Open】开启 lesson4 测试脚本。
3. 检查 Flight Reservation 在主窗口

   如果有其它对话窗口请先关闭。
4. 确认工具列上显示 

模式
5. 点选 Run From Top

   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则Run Test 窗口将会开启，按下【OK】开始执行测试。
6. 输入 Test Run Name

   输入Test Run Name，WinRunner 会将测试脚本执行的结果储存在Test Run Name 的目录下，如res1。而此测试结果将会储存在测试脚本目录下。

   请注意窗口下方【Display test results at end of run】，若勾选此选项，则当测试脚本执行完毕后，WinRunner 会自动开启测试执行结果的窗口。请勾选此选项。
7. 执行

   按下【OK】后 WinRunner 会开始执行测试脚本。

   请注意观察 WinRunner 如何执行测试脚本。
8. 检视执行结果

   当测试执行完毕后，WinRunner 会开启Test Results 窗口，显示测试执行的结果。

[实训要求]

1. 撰写实验报告，主要填写本人上机测试步骤和测试内容。
2. 提交录制的测试脚本。

实训七　Web 系统测试

[实训目的]

应用 Web 测试工具对 Web 系统进行功能和性能测试；

[实训原理]

对 Web 系统测试需要从功能、性能、可用性、安全性等多方面进行测试。

一、 功能测试

对 Web 系统进行功能测试包括以下几个方面：

1. 链接测试

链接是 Web 应用系统的一个主要特征，它是在页面之间切换和指导用户去一些不知道地址的页面的主要手段。链接测试可分为三个方面。首先，测试所有链接是否按指示的那样确实链接到了该链接的页面；其次，测试所链接的页面是否存在；最后，保证 Web 应用系统上没有孤立的页面，所谓孤立页面是指没有链接指向该页面。

2. 表单测试

当用户给 Web 应用系统管理员提交信息时，就需要使用表单操作，例如用户注册、登陆、信息提交等。在这种情况下，我们必须测试提交操作的完整性，以校验提交给服务器的信息的正确性。例如：用户填写的出生日期与职业是否恰当，填写的所属省份与所在城市是否匹配等。如果使用了默认值，还要检验默认值的正确性。如果表单只能接受指定的某些值，则也要进行测试。例如：只能接受某些字符，测试时可以跳过这些字符，看系统是否会报错。

3. Cookies 测试

Cookies 通常用来存储用户信息和用户在应用系统的操作，当一个用户使用 Cookies 访问了某一个应用系统时， Web 服务器将发送关于用户的信息，把该信息以 Cookies 的形式存储在客户端计算机上，这可用来创建动态和自定义页面或者存储登陆等信息。

如果 Web 应用系统使用了 Cookies ，就必须检查 Cookies 是否能正常工作。测试的内容可包括 Cookies 是否起作用，是否按预定的时间进行保存，刷新对 Cookies 有什么影响等。

4. 数据库测试

在 Web 应用技术中，数据库起着重要的作用，数据库为 Web 应用系统的管理、运行、查询和实现用户对数据存储的请求等提供空间。

在使用了数据库的 Web 应用系统中，一般情况下，可能发生两种错误，分别是数据一致性错误和输出错误。数据一致性错误主要是由于用户提交的表单信息不正确而造成的，而输出错误主要是由于网络速度或程序设计问题等引起的，针对这两种情况，可分别进行测试。

二、性能测试

对 Web 系统进行性能测试主要包括以下几个方面：

1. 连接速度测试

用户连接到 Web 应用系统的速度根据上网方式的变化而变化，他们或许是电话拨号，或是宽带上网。当下载一个程序时，用户可以等较长的时间，但如果仅仅访问一个页面就不会这样。如果 Web 系统响应时间太长（例如超过 5 秒钟），用户就会因没有耐心等待而离开。 另外，有些页面有超时的限

制，如果响应速度太慢，用户可能还没来得及浏览内容，就需要重新登陆了。而且，连接速度太慢，还可能引起数据丢失，使用户得不到真实的页面。

2. 负载测试

负载测试是为了测量 Web 系统在某一负载级别上的性能，以保证 Web 系统在需求范围内能正常工作。负载级别可以是某个时刻同时访问 Web 系统的用户数量，也可以是在线数据处理的数量。例如： Web 应用系统能允许多少个用户同时在线？如果超过了这个数量，会出现什么现象？ Web 应用系统能否处理大量用户对同一个页面的请求？

3. 压力测试

进行压力测试是指实际破坏一个 Web 应用系统，测试系统的反映。压力测试是测试系统的限制和故障恢复能力，也就是测试 Web 应用系统会不会崩溃，在什么情况下会崩溃。黑客常常提供错误的数据负载，直到 Web 应用系统崩溃，接着当系统重新启动时获得存取权。

压力测试的区域包括表单、登陆和其他信息传输页面等。

三、 可用性测试

1. 导航测试

导航描述了用户在一个页面内操作的方式。通过考虑下列问题，可以决定一个 Web 应用系统是否易于导航：导航是否直观？ Web 系统的主要部分是否可通过主页存取？ Web 系统是否需要站点地图、搜索引擎或其他的导航帮助？

2. 图形测试

一个 Web 应用系统的图形可以包括图片、动画、边框、颜色、字体、背景、按钮等。图形测试的内容有：

（1） 确保图形有明确的用途，图片或动画不要胡乱地堆在一起，以免浪费传输时间。
（2） 验证所有页面字体的风格是否一致。
（3） 背景颜色应该与字体颜色和前景颜色相搭配。
（4） 图片的大小和质量也是一个很重要的因素，一般采用 JPG 或 GIF 压缩。

3. 内容测试

内容测试用来检验 Web 应用系统提供信息的正确性、准确性和相关性。

4. 整体界面测试

整体界面是指整个 Web 应用系统的页面结构设计，是给用户的一个整体感。对整体界面的测试过程，其实是一个对最终用户进行调查的过程。一般 Web 应用系统采取在主页上做一个调查问卷的形式，来得到最终用户的反馈信息。

[实训内容]

1．题目一：测试网站功能

选择一个网站，对其进行功能测试。要求首先编写测试用例，然后用 QuickTest Professional 自动化测试工具对该网站进行测试。

2．题目二：测试网站性能

选择一个网站，对其进行性能测试。要求使用 WebLoad 进行压力测试。（WebLoad 使用方法请参阅文档：“用 webload 进行 web application 性能测试.doc”）

[实训步骤]

1．题目一实验步骤

（1） 拟定测试计划；

（2） 撰写测试方案；
（3） 设计测试用例；
（4） 录制测试脚本；
（5） 根据录制的脚本和测试用例创建页面检查点、对象检查点、文本检查点或者表格检查点；根据测试脚本和测试用例，进行参数化测试；
（6） 分析测试结果。

2．题目二实验步骤
（1） 计划一个压力会话；
（2） 创建测试议程；
（3） 创建压力模板；
（4） 运行压力模板；
（5） 输入测试报告并分析测试结果。

[实训要求]

1．撰写实验报告；
2．撰写 Web 系统的测试计划，测试方案；
3．撰写 Web 系统测试的测试用例；
4．撰写缺陷报告；
5．提交测试脚本。