



第4章 分支控制

——程序测试



哈尔滨工业大学

苏小红

sxh@hit.edu.cn

本节要讨论的主要问题

- 程序测试的目的是什么？
- 常用的程序测试方法有哪些？



```

#include<stdio.h>
int main()
{
    int score, mark;
    printf("Please input  score:");
    scanf("%d", &score);
    mark = score / 10;
    switch (mark)
    {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:    printf("garde:E\n");
                   break;
        case 6:    printf("garde:D\n");
                   break;
        case 7:    printf("garde:C\n");
                   break;
        case 8:    printf("garde:B\n");
                   break;
        case 9:
        case 10:   printf("garde:A\n");
                   break;
        default:   printf("Input error!\n");
    }
    return 0;
}

```

■ 程序测试（Software Testing）

- * 给定一组输入，通过运行被测程序，检查程序输出是否与预期结果一致

输入数据	预期结果	实际输出结果
0, 15, 55	E	E
	D	D
	C	C
85	B	B
95, 100	A	A
-10, 110	Input error!	Input error!
-5	Input error!	E
105	Input error!	A

测试用例

```

#include<stdio.h>
int main()
{
    int score, mark;
    printf("Please input  score:");
    scanf("%d", &score);
    mark = score / 10;
    switch (mark)
    {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:    printf("garde:E\n");
                   break;
        case 6:    printf("garde:D\n");
                   break;
        case 7:    printf("garde:C\n");
                   break;
        case 8:    printf("garde:B\n");
                   break;
        case 9:
        case 10:   printf("garde:A\n");
                   break;
        default:   printf("Input error!\n");
    }
    return 0;
}

```

■ 测试用例的选取方法

- * 尽量覆盖所有分支（路径）
- * 应考虑到合法的输入和不合法的输入以及各种边界条件

输入数据	预期结果	实际输出结果
0, 15, 55	E	E
65	D	D
75	C	C
85	B	B
95	A	A
10, 110	Input error!	Input error!
-5	Input error!	E
105	Input error!	A

抽样检查?

程序测试的目的

- 测试只能证明程序有错，不能证明程序无错

——E.W.Dijkstra

- 测试的目的

- * 通过运行测试用例，找出程序中尽可能多的Bug
- * 成功的测试在于发现迄今为止尚未发现的Bug

- 测试人员的主要任务

- * 站在使用者角度，通过不断使用（包括非常规使用），尽可能多地找Bug
- * 测试的过程就像黑客的攻击过程



程序测试方法的分类

白盒测试（结构测试）

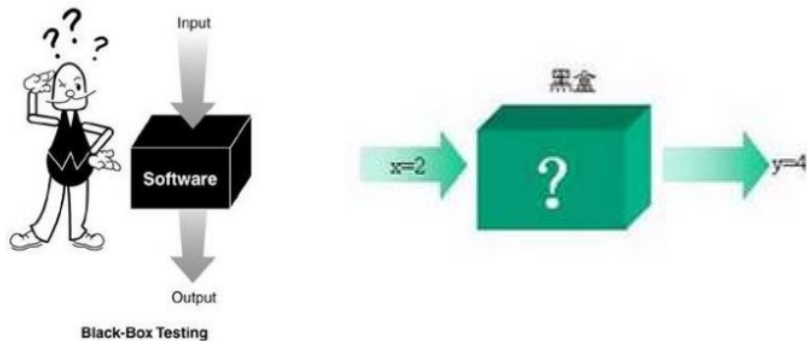
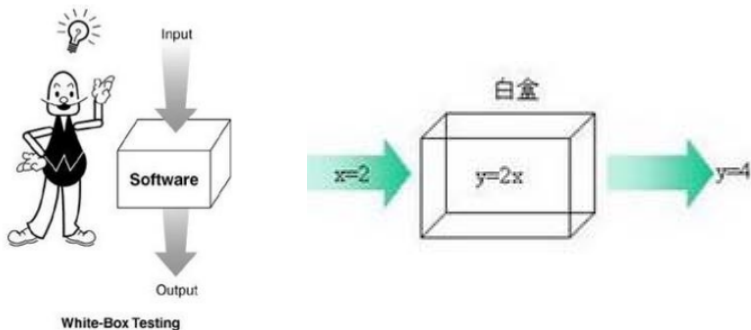
完全了解程序内部的逻辑结构和处理过程，按照程序内部的逻辑测试程序，检验程序中的每条逻辑路径是否都能按预定要求正确工作

主要用于测试的早期和重要的路径

黑盒测试（功能测试）

不考虑程序内部的逻辑结构和处理过程，把系统看成一个黑盒子，只根据需求规格说明书的要求，设计测试用例，检查程序的功能是否符合它的功能说明

主要用于测试的后期和重要的功能



```
#include<stdio.h>
int main()
{
    int score, mark;
    printf("Please input  score:");
    scanf("%d", &score);
    mark = score / 10;
    switch (mark)
    {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:    printf("garde:E\n");
                  break;
        case 6:    printf("garde:D\n");
                  break;
        case 7:    printf("garde:C\n");
                  break;
        case 8:    printf("garde:B\n");
                  break;
        case 9:
        case 10:   printf("garde:A\n");
                  break;
        default:   printf("Input error!\n");
    }
    return 0;
}
```



输入数据	预期结果	实际输出结果
0, 15, 55	E	E
65	D	D
75	C	C
85	B	B
95, 100	A	A
-10, 110	Input error!	Input error!
-5	Input error!	E
105	Input error!	A



实例：判断三角形的类型

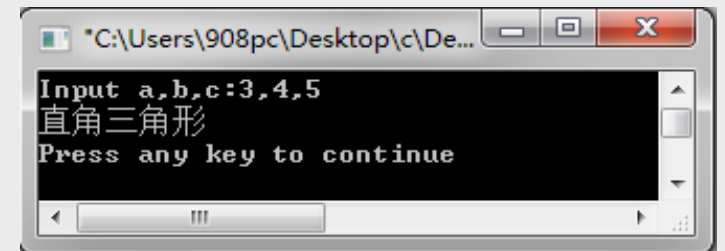
```
#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c;
    printf("Input the three edge length:");
    scanf("%f, %f, %f", &a, &b, &c);
    if (a+b>c && b+c>a && a+c>b)
    {
        if (a==b || b==c || c==a)
            printf("等腰三角形\n");
        else if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
            printf("直角三角形\n");
        else
            printf("一般三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
    return 0;
}
```

4个测试用例都通过
能说明程序正确吗？



直角3,4,5
等腰4,4,5
一般3,4,6
非三角形3,4,9

黑盒测试




```

#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c;
    printf("Input the three edge length:");
    scanf("%f, %f, %f", &a, &b, &c);
    if (a+b>c && b+c>a && a+c>b)
    {
        if (a==b || b==c || c==a)
            printf("等腰三角形\n");
        else if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
            printf("直角三角形\n");
        else
            printf("一般三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
    return 0;
}

```

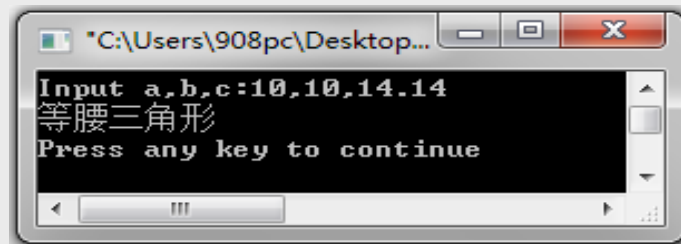
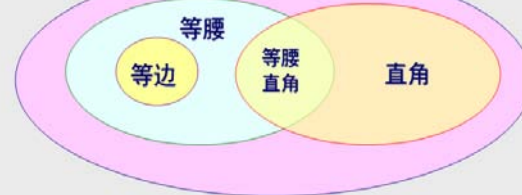
实例：判断三角形的类型

4个测试用例都通过
能说明程序正确吗？



直角3,4,5
 等腰4,4,5
 一般3,4,6
 非三角形3,4,9
 等腰直角三角形
 10, 10, ?

不是三角形

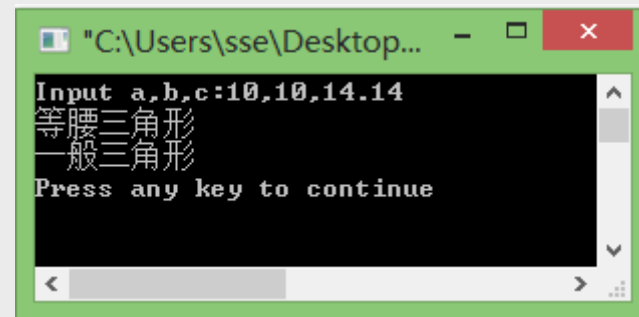


实例：判断三角形的类型

如何比较
实数相等？

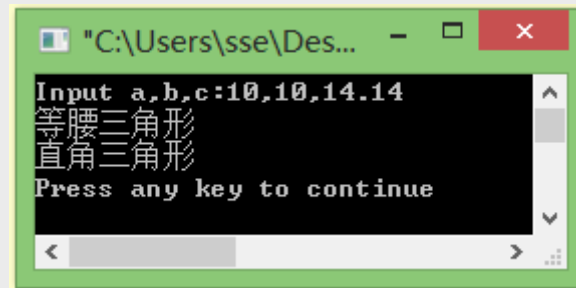


```
#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c;
    printf("Input the three edge length:");
    scanf("%f, %f, %f", &a, &b, &c);
    if (a+b>c && b+c>a && a+c>b)
    {
        if (a==b || b==c || c==a)
            printf("等腰三角形\n");
        if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
            printf("直角三角形\n");
        else
            printf("一般三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
    return 0;
}
```



实例：判断三角形的类型

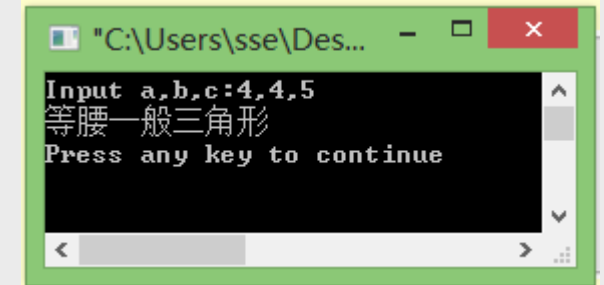
```
#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c;
    printf("Input the three edge length:");
    scanf("%f, %f, %f", &a, &b, &c);
    if (a+b>c && b+c>a && a+c>b)
    {
        if (fabs(a-b)<=0.1 || fabs(b-c)<=0.1 || fabs(c-a)<=0.1)
            printf("等腰三角形\n");
        if (fabs(a*a+b*b-c*c)<=0.1 || fabs(a*a+c*c-b*b)<=0.1 ||
            fabs(b*b+c*c-a*a)<=0.1)
            printf("直角三角形\n");
        else
            printf("一般三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
    return 0;
}
```



实例：判断三角形的类型

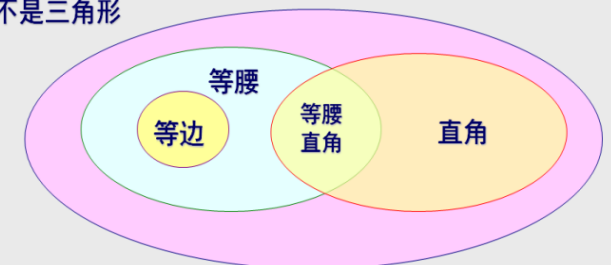
```
#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c;
    .....
    if (a+b>c && b+c>a && a+c>b)
    {
        if (fabs(a-b)<=0.1 || fabs(b-c)<=0.1 || fabs(c-a)<=0.1)
            printf("等腰");
        if (fabs(a*a+b*b-c*c)<=0.1 || fabs(a*a+c*c-b*b)<=0.1 ||
            fabs(b*b+c*c-a*a)<=0.1)
            printf("直角");
        else
            printf("一般");
        printf("三角形\n");
    }
    else
    {
        printf("不是三角形\n");
    }
    return 0;
}
```

回归测试



等腰直角10,10,14.14
直角三角形3,4,5
等腰三角形4,4,5
一般三角形3,4,6
非三角形3,4,9

不是三角形



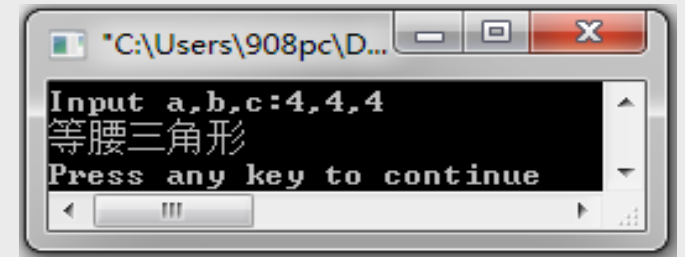
实例：判断三角形的类型

```
int main()
{
    float a, b, c;
    int flag = 1;    //先假设是一般三角形

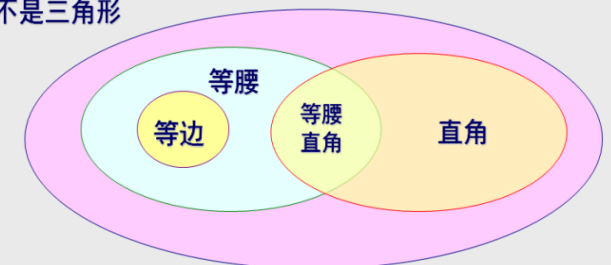
    if (.....
        if (a+b>c && b+c>a && a+c>b)
        {
            if (fabs(a-b)<=0.1 || fabs(b-c)<=0.1 || fabs(c-a)<=0.1)
            {
                printf("等腰");
                flag = 0;
            }
            if (fabs(a*a+b*b-c*c)<=0.1 || fabs(a*a+c*c-b*b)<=0.1 ||
                fabs(b*b+c*c-a*a)<=0.1)
            {
                printf("直角");
                flag = 0;
            }
            if (flag)
            {
                printf("一般");
            }
            printf("三角形\n");
        }
        else
        {
            printf("不是三角形\n");
        }
        return 0;
}
```

if (flag != 0)

等腰直角10,10,14.14
直角三角形3,4,5
等腰三角形4,4,5
一般三角形3,4,6
非三角形3,4,9

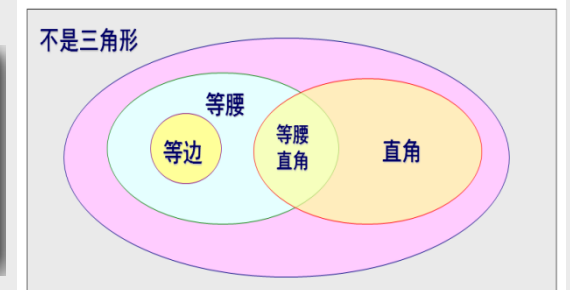
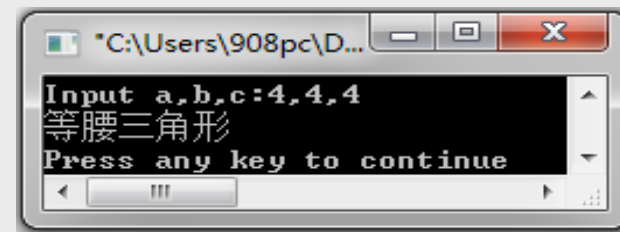


不是三角形



实例：判断三角形的类型

```
int main()  
{  
    .....  
    if (a+b>c && b+c>a && a+c>b)  
    {  
        if (fabs(a-b)<=0.1 || fabs(b-c)<=0.1 || fabs(c-a)<=0.1)  
        {  
            printf("等腰");  
            flag = 0;  
        }  
        else if (fabs(a-b)<=0.1&&fabs(b-c)<=0.1&&fabs(c-a)<=0.1)  
        {  
            printf("等边");  
            flag = 0;  
        }  
        if (fabs(a*a+b*b-c*c)<=0.1 || fabs(a*a+c*c-b*b)<=0.1 ||  
            fabs(b*b+c*c-a*a)<=0.1)  
        {  
            printf("直角");  
            flag = 0;  
        }  
        if (flag) printf("一般");  
        printf("三角形\n");  
    }  
    else  
        printf("不是三角形\n");  
    return 0;  
}
```



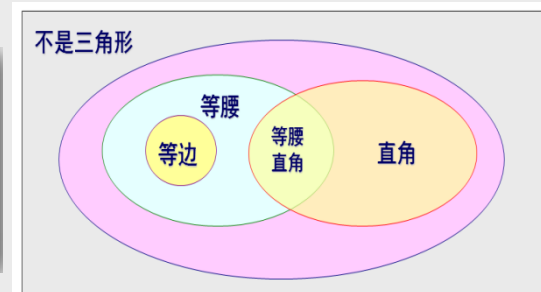
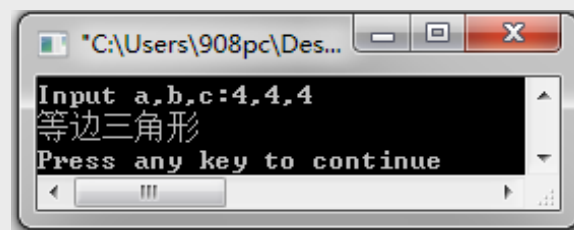
```

int main()
{
    if (.....
        if (a+b>c && b+c>a && a+c>b)
        {
            if (fabs(a-b)<=0.1&&fabs(b-c)<=0.1&&fabs(c-a)<=0.1)
            {
                printf("等边");
                flag = 0;
            }
            else if (fabs(a-b)<=0.1 || fabs(b-c)<=0.1 || fabs(c-a)<=0.1)
            {
                printf("等腰");
                flag = 0;
            }
            if (fabs(a*a+b*b-c*c)<=0.1 || fabs(a*a+c*c-b*b)<=0.1 ||
                fabs(b*b+c*c-a*a)<=0.1)
            {
                printf("直角");
                flag = 0;
            }
            if (flag) printf("一般");
            printf("三角形\n");
        }
    else
        printf("不是三角形\n");
    return 0;
}

```

实例：判断三角形的类型

有包含关系的，通常先测试范围小的



讨论

- 请举例说明，如何进行白盒测试，或者黑盒测试？是否需要
进行边界测试？如何设计测试用例进行边界测试？



