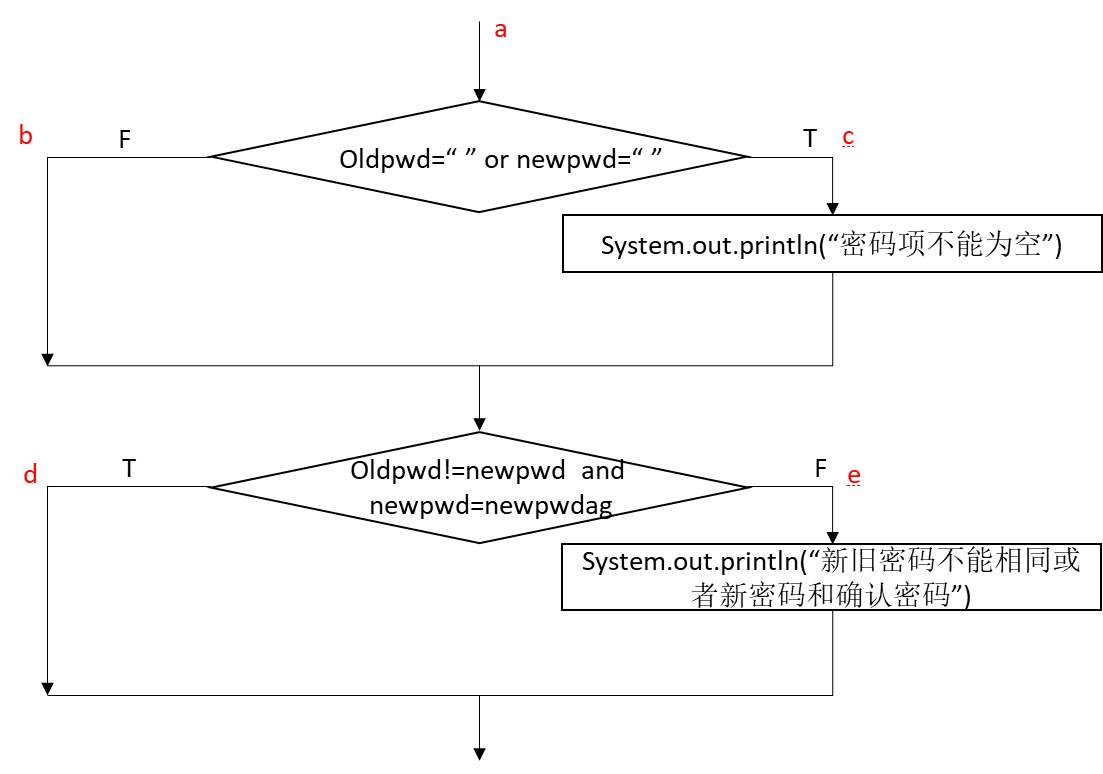
一、测试用例

1、逻辑覆盖

学生成绩管理系统的用户密码修改模块部分流程图如下，其中变量oldpwd为原密码，newpwd为新密码，newpwdag为确认密码。



a,b,c,d,e为路径标识，我们采用如下路径表示：

L1(a->c->e)

L2(a->b->d)

L3(a->b->e)

L4(a->c->d)

其中：







（1）语句覆盖

根据上述程序流程图，可以编码如下：

if Oldpwd=“ ” or newpwd=“ ”

then System.out.println(“密码项不能为空”)

end if

if Oldpwd=newpwd or newpwd!=newpwdag

then System.out.println(“新旧密码不能相同或者新密码和确认密码要相同”)

end if

在图例中，正好所有的可执行语句都在路径L1上，所以选择路径 L1设计测试用例，就可以覆盖所有的可执行语句

测试用例的设计格式如下  
【输入的(oldpwd, newpwd, newpwdag)，输出的(提示1,提示2)】

为图例设计满足语句覆盖的测试用例是:  
【(“”, “”, “”)，(“密码项不能为空”, “新旧密码不能相同或者新密码和确认密码不同”】　 覆盖 ace【L1】

（2）分支覆盖

选择路径L1和L2:

【(“”, “”, “”)，(“密码项不能为空”, “新旧密码不能相同或者新密码和确认密码要相同”】覆盖 ace【L1】

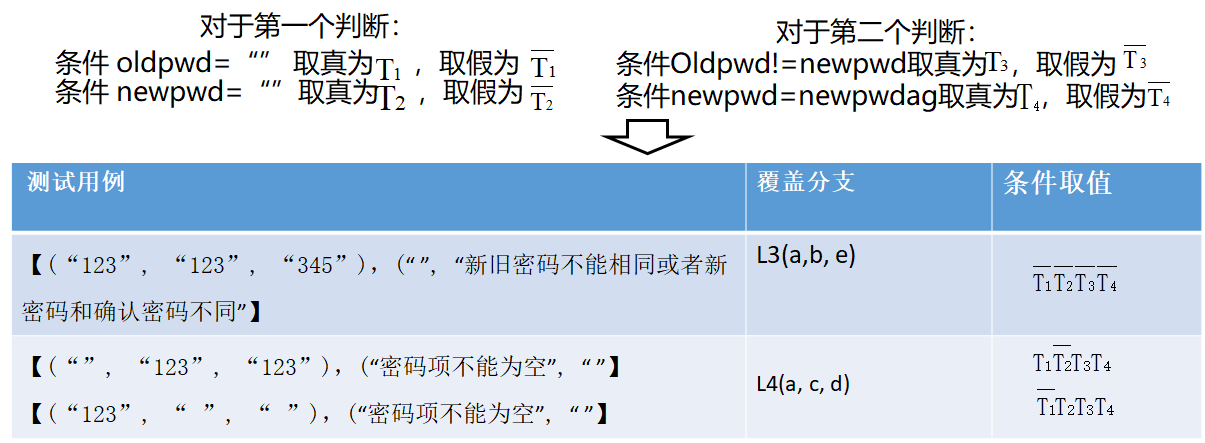
【(“123”, “234”, “234”)，(“ ”, “ ”】覆盖 abd【L2】

所有取真和取假分支均可经历一次，满足分支覆盖要求

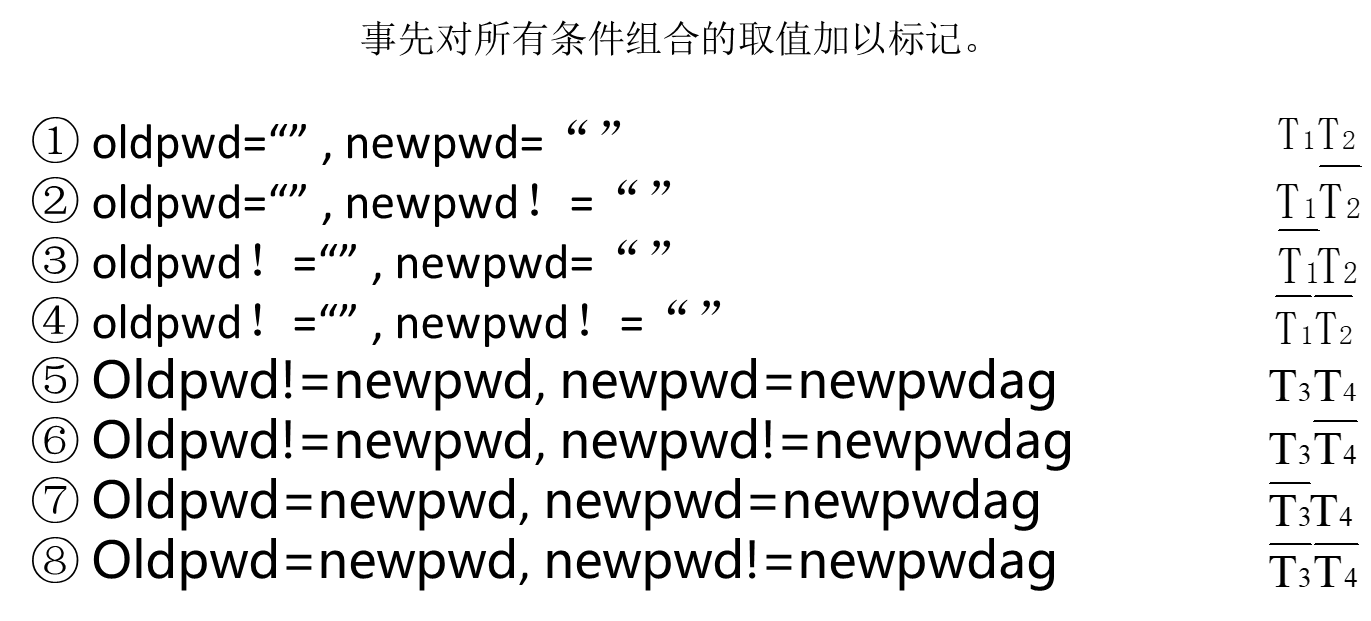
思考，如何选择路径L3和L4呢，是否也可以满足要求。

（3）判断覆盖

事先对所有条件的取值加以标记。



（4）条件组合覆盖

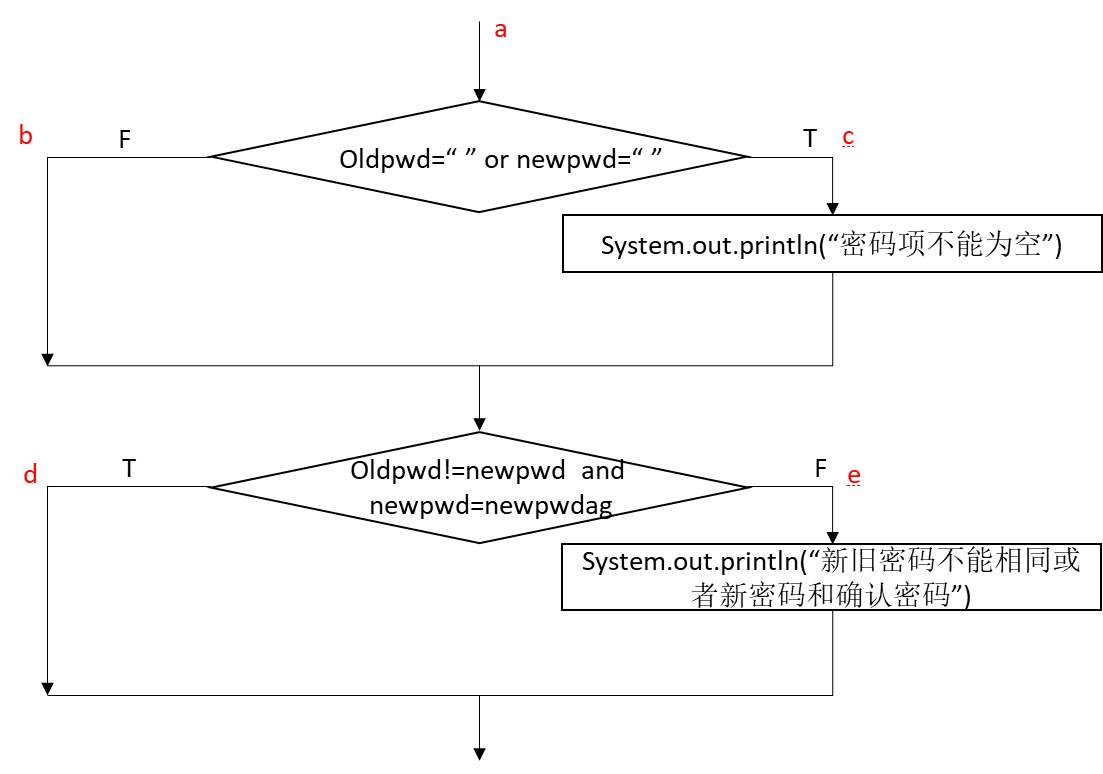




2、控制流图覆盖测试

（1）路径覆盖

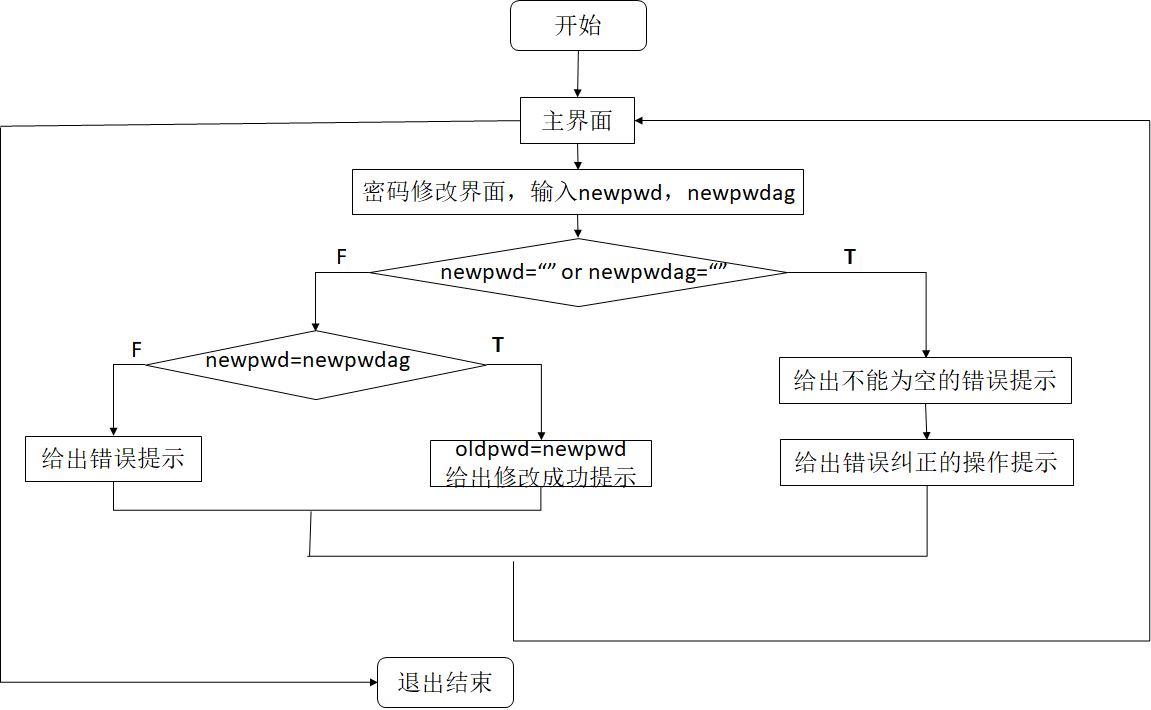
仍以前面的案例为例，如下图所示，所有可能的路径包括L1、L2、L3、L4



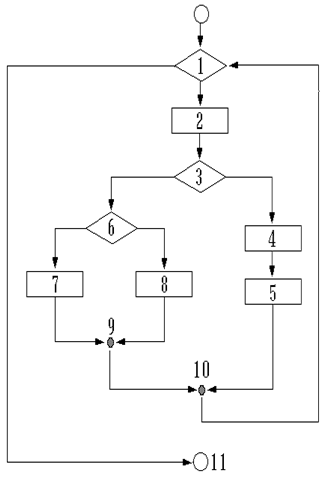
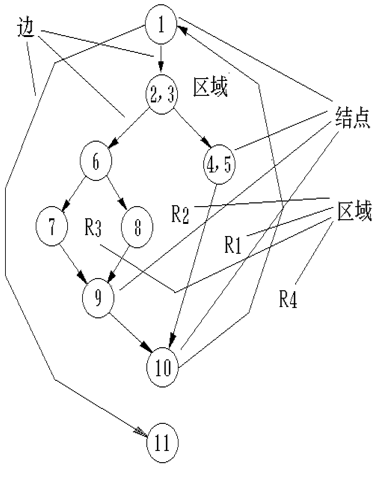
因此：分别设计4个测试用例，通过L1-L4即可。

（2）基本路径覆盖测试

学生管理系统中，用户账号修改的程序流程图如下：



按照基本路径法，画出程序控制示意图和程序控制流图，如下所示：

计算程序环路复杂性：

V(G)=e(11)-n(9)+2=4

确定基本路径级：  
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

基本路径集：path1，path2，path3，path4

3、等价类划分

（1）原则1：

学生信息的出生年月的字段部分约束：“有效月份为数字1-12”

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 学生出生月份 | 1 ≤出生月份≤12 (1) | 出生月份＜1 (2)；出生月份＞12 (3) |

（2）原则2：

学生信息的学号字段字段部分约束：“学号必须为13位数字串”

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 学生的学号 | 13位数字串(1) | 非13位数字串(2) |

（3）原则3：

学生信息的在读状态字段部分约束：“为布尔量”

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 学生的在读状态字段 | 是(1) | 否(2) |

（4）原则4：

学生信息的性别字段部分约束：“性别为{“男”，“女”}，且在系统中需要进行分别计数”。

教师信息中

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 学生的性别字段 | “男”(1)，”女”(2) | 不在{“男”，“女”}的其他值集合(3) |

（5）原则5：

学生信息的学院字段部分约束：“必须为字符型”（感觉这个在测试中没有定义清楚，应该是在某一集合类中取值。比如，输入一个“123”，那么作为输入串是字符串型符合条件，但内容却是整数不符合要求）

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 学生的学院字段数据类型 | 字符型(1) | 整数型(2),浮点型(3)布尔型(4) |

（6）综合题目：

在学生管理系统添加教师信息的部分约束：“账号由字母开头，后跟字母或数字的任意组合构成，账号不能为空，账号字符数不超过8个，性别为{“男”，“女”}，且在系统中需要进行分别计数，教师在岗状态为布尔量。”

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 第一个字符 | 字母 (1) | 非字母 (2) |
| 标识符组成 | 字母、数字的任意组合 (2)； | 非字母数字字符 (3)；保留字 (4) |
| 账号长度 | 0<账号长度≤8 (6) | 0个 (7)；>8个 (8) |
| 性别 | “男”(9)，”女”(9) | 不在{“男”，“女”}的其他值集合(10) |
| 教师在岗状态 | 是(11) | 否(12) |

4、边界值分析

（1）单侧边界为闭区间

可以在学生成绩管理系统中查询优秀成绩信息，查询标准为成绩字段至少85分，采用针对身高条件采用边界值分析法设计测试用例。

|  |  |  |
| --- | --- | --- |
| **测试条件** | **输入值（** | **预期结果** |
| **查询优秀成绩信息** | 85 | 符合要求 |
| 86 | 符合要求 |
| 84 | 错误提示 |

（2）单侧边界为开区间

可以在学生成绩管理系统中查询不及格成绩信息，查询标准为成绩字段低于60分，采用针对身高条件采用边界值分析法设计测试用例。

|  |  |  |
| --- | --- | --- |
| **测试条件** | **输入值** | **预期结果** |
| **身高** | 59 | 符合要求 |
| 60 | 错误提示 |

（3）单侧边界多条件

学生成绩管理系统中，课程成绩构成最多由三部分组成，必须包含期末成绩，假定期末成绩为A，其他两部分成绩B，C，其有效成绩部分约束为：A>50,B≥0,C≥0，1, B+C<50。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例 | A | B | C | 预期输出 |
| 1 | 50 | 30 | 20 | 错误提示 |
| 2 | 51 | 30 | 20 | 符合要求 |
| 3 | 60 | 1 | 39 | 符合要求 |
| 4 | 60 | 0 | 40 | 符合要求 |
| 5 | 60 | -1 | 41 | 错误提示 |
| 6 | 60 | 39 | 1 | 符合要求 |
| 7 | 60 | 40 | 0 | 符合要求 |
| 8 | 60 | 41 | -1 | 错误提示 |
| 9 | 50 | 20 | 30 | 错误提示 |
| 10 | 51 | 20 | 29 | 符合要求 |

（4）区间范围-闭区间

学生成绩管理系统的成绩输入中，输入分数范围为[85,100]，则成绩等级为优秀，采用边界值分析法设计测试用例。

上点：为85和100

内点：即85到100之间的任何一点，如：93。

离点：为84和101。

测试用例为{84，85，93，100，101}

（5）区间范围-开区间

学生成绩管理系统的成绩输入中，输入分数范围为（0,60），则成绩等级为不及格，采用边界值分析法设计测试用例。

上点：为0和60

内点：即0到60之间的任何一点，如：33。

离点：为1和59。

测试用例为{0，1，33，59，60}

（6）区间范围-半开半闭区间

学生成绩管理系统的成绩输入中，如输入分数范围为[60,75），则成绩等级为及格，采用边界值分析法设计测试用例。

上点：为60和75

内点：即60到75之间的任何一点，如：67。

离点：为59和74。

测试用例为{59，60，67，74，75}

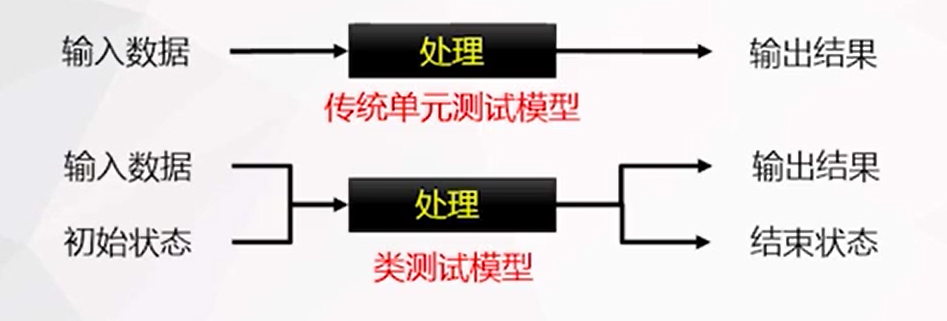
5、状态转换图

采用原有案例，将QQ登陆改为，软件启动。

二、面向对象测试

根据面向对象程序的特点，可以理解为：

面向对象的单元测试：类测试，考察封装在类中的方法和类的状态行为



面向对象的集成测试：类簇测试

类簇是指相互有影响，联系比较紧密的类，通过继承和派生构建。是一个相对独立的实体，在整体上是可执行和测试的。并且实现了一个内聚的责任集合，但不提供被测试程序的全部功能，相当于一个子系统。

类簇测试主要根据系统中相关类的层次关系，检查类之间的相互作用的正确性，即检查各相关类之间的消息连接的合法性、子类的继承性与父类的一致性、动态绑定执行的正确性、类簇协同完成系统功能的正确性等。

面向对象软件没有传统的层次结构，传统软件的自上而下、自下而上的集成方法并不太适用，面向对象软件的集成策略通常有一下两种：

（1）基于类间协作关系的横向测试，由系统的一个输入事件作为激励，对其触发的一组类进行测试

（2）基于类间继承关系的纵向测试，首先测试不使用或很少使用其他类的独立类来构建初始系统，其次测试下一层继承独立类的类，称为依赖类。这个依赖类的测试序列一直执行到构建完整个系统

面向对象的系统测试，由于系统测试不考虑内部结构和中间结果，因此与传统的测试方法差异不大，可以沿用传统的测试方法。