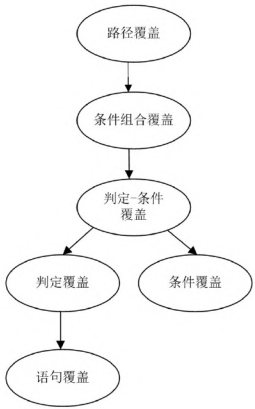
测试覆盖指标（二）

回顾

- 基于程序图的测试覆盖指标
  - 节点覆盖
  - 边覆盖
- 路径覆盖逻辑覆盖指标
  - 语句覆盖（等于节点覆盖）
  - 判断覆盖（等于边覆盖）
  - 条件覆盖
  - 条件-判定覆盖
  - 条件组合覆盖
  - 路径覆盖



目录

■Miller的覆盖指标

■4种循环

Miller的覆盖指标

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有“统计重要性”的部分
C_∞	所有可能的执行路径

- 当一组测试用例满足DD路径覆盖指标时，可以发现程序中大约85%的缺陷。

Miller的覆盖指标

- 由于覆盖指标C_0最初针对的是完整语句，即if-then或do-until等都作为一个完整语句来考虑。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 如果将指标C_0中的语句对应到语句片段，则C_0等同于C_1。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 这里的覆盖指标C_1等同于前面所述的语句覆盖和节点覆盖。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_{1p}指的是覆盖所有判定的每种分支，这与前面的判定覆盖和边覆盖相同。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_2是在DD路径覆盖C_1基础上增加了循环覆盖。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_d是在覆盖指标C_1基础上增加了DD路径的所有依赖关系。
  - 在一些程序中，DD路径的执行存在依赖关系，单纯的DD路径覆盖是无法发现这类情况的；
  - 一些复杂的深层次缺陷无法被发现。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有“统计重要性”的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_{mcc}指的是考虑每个判断中可能取值的各种情况，这与前面的条件组合覆盖相同。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_{ik}指的是包含最多k次循环的所有程序路径，通常k = 2
  - 该指标考虑了循环的情况，但是对于循环问题进行了简化。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有“统计重要性”的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_{stat}指的是重点考虑“重要的”路径。
  - 难点：如何确定路径是否重要？

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有“统计重要性”的部分
C_∞	所有可能的执行路径

Miller的覆盖指标

- 覆盖指标C_∞指的是覆盖程序所有可能的执行路径，与前面路径覆盖指标相同。

指标	覆盖描述
C_0	所有语句
C_1	所有DD路径
C_{1p}	所有判断的每种分支
C_2	C_1覆盖+循环覆盖
C_d	C_1覆盖+DD路径的所有依赖对偶
C_{MCC}	多条件覆盖
C_{ik}	包含最多k次循环的所有程序路径（通常k=2）
C_{stat}	路径具有"统计重要性"的部分
C_∞	所有可能的执行路径

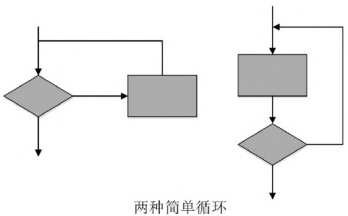
目录

■Miller的覆盖指标

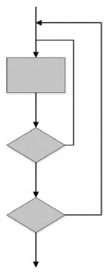
■4种循环

循环的类型

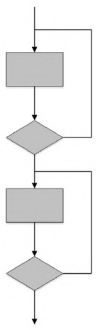
- 4种类型：简单循环、串接循环、嵌套循环和复杂循环



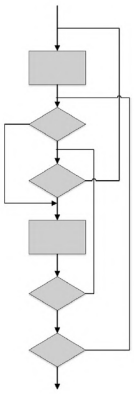
两种简单循环



嵌套循环



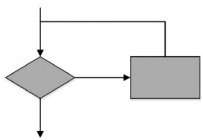
串接循环



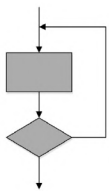
复杂循环

简单循环的测试

- 采用改进的边界值来设计测试用例
  - 不进入循环
  - 只通过一次循环
  - 两次通过循环
  - m次通过循环，且m < n
  - n-1 次通过循环
  - n次通过循环



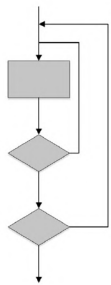
while-do循环



do-until循环

嵌套循环

- 如果一个循环体内包含其它循环，则产生的程序路径数量会呈几何增加，所需要的测试数量也急剧增加。
- 为使测试工作更具效率，采用“由内向外”逐层测试的思想进行测试用例设计。



嵌套循环

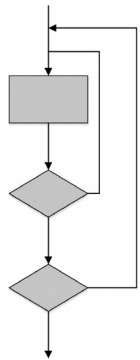
嵌套循环

步骤1：首先采用简单循环测试策略来测试最内层循环，同时其它层次的循环变量为最小值（即尽可能将除被测层次循环外的所有循环不进行多次循环处理）。

步骤2：由内向外构造下一层循环，所选择的这层采用简单循环处理方式；这层所包含的内层循环采用“典型”值；这层之外的循环设置为最小值。

步骤3：重复步骤2，直到所有层次的循环都测试完成。

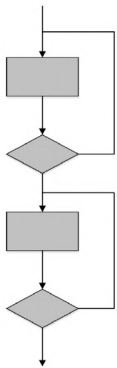
步骤4：对全部各层循环同时取最小值或同时取最大值进行测试。



嵌套循环

串接循环

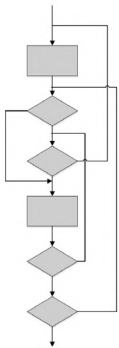
- 串接循环是指程序中两个连续的循环。
- 如果两个循环彼此独立，则使用简单循环测试策略分别测试这两个循环；
- 如果两个循环并不独立，即：第一个循环会影响第二个循环指数取值，则采用嵌套循环测试策略较为合适。



串接循环

串接循环

- 复杂循环也叫不规则循环，是指循环体内有节点的入度或出度大于1，甚至两个或多个循环相互嵌套。
- 这种情况属于不良风格的程序，应该重新设计程序。



总结

- Miller的覆盖指标
- 4种循环

案例分析

目录

案例：保险金程序

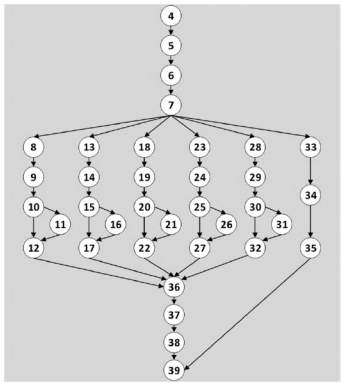
某保险公司在给人们投健康险的时候，主要考虑2个因素：年龄（Age）和过去一年的就医次数（Num）。保险费 = 基础价×年龄因子 - 优惠。其中：年龄因子是投保人年龄的函数；优惠是指投保人过去一年就医次数不高于一个与年龄有关的门限值时，保险公司给予的减免优惠。保险公司认可的投保人年龄范围是2～80岁。如果投保人过去一年就医次数超过8次，则保险公司拒绝投保。假设基础价为1000元。

年龄范围	年龄因子	就医次数门限	优惠
[2, 12]	1.5	4	100
(12, 25]	0.8	2	150
(25, 45]	1	4	200
(45, 65]	1.2	4	150
(65, 80]	2	6	100

保险金程序源码

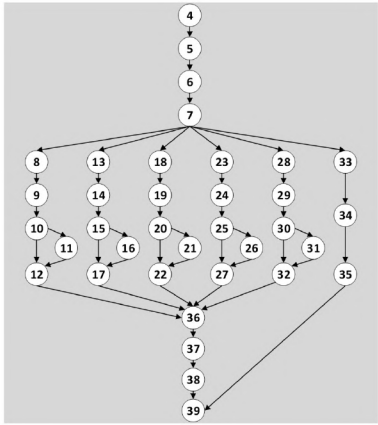
```
1.    Program Insurance                    //保险金程序
2.    Int age, num, insurance, discount;
      //定义四个变量：年龄，就医次数，保险费，优惠
3.    Float ageFactor;//年龄因子
4.    Int baseInsurance = 1000;            //保险金基数
5.    Input(age , num);                              //输入参数：年龄，就医次数
6.    insurance = discount = 0;
7.    Select Case age
8.    Case 1: 2 <= age <= 12
9.                ageFactor = 1.5;
10.               If(0 <= num <= 4)
11.                       Then discount = 100;
12.               EndIf
13.   Case 2: 12 < age <= 25
14.               ageFactor = 0.8;
15.               If(0 <= num <= 2)
16.                       Then discount = 150;
17.               EndIf
18.   Case 3: 25 < age <= 45
19.               ageFactor = 1;
20.               If(0 <= num <= 4)
21.                       Then discount = 200;
22.               EndIf
23.   Case 4: 45 < age <= 65
24.               ageFactor = 1.2;
25.               If(0 <= num <= 4)
26.                       Then discount = 150;
27.               EndIf
28.   Case 5: 65 < age <= 80
29.               ageFactor = 2;
30.               If(0 <= num <= 6)
31.                       Then discount = 100;
32.               EndIf
33.   Case 6: Else
34.               Output("投保人年纪超出范围");
35.               Return;        //              程序返回
36.   EndSelect
37.   insurance = baseInsurance * ageFactor – discount;
38.   Output(insurance)
39.   End Insurance                //程序结束
```

程序图



可行路径

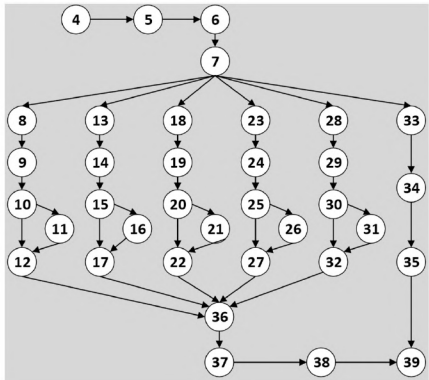
编号	可行路径
P1	4-5-6-7-8-9-10-12-36-37-38-39
P2	4-5-6-7-8-9-10-11-12-36-37-38-39
P3	4-5-6-7-13-14-15-17-36-37-38-39
P4	4-5-6-7-13-14-15-16-17-36-37-38-39
P5	4-5-6-7-18-19-20-22-36-37-38-39
P6	4-5-6-7-18-19-20-21-22-36-37-38-39
P7	4-5-6-7-23-24-25-27-36-37-38-39
P8	4-5-6-7-23-24-25-26-27-36-37-38-39
P9	4-5-6-7-28-29-30-32-36-37-38-39
P10	4-5-6-7-28-29-30-31-32-36-37-38-39
P11	4-5-6-7-33-34-35-39



语句覆盖

- 只有6条路径上面有语句，6个测试用例

编号	输入		通过路径
	age	num	
1	5	1	P2
2	20	1	P4
3	40	1	P6
4	60	1	P8
5	70	1	P10
6	90	1	P11



判定覆盖

保险金程序总共包含6个判定，分别为：

（1）语句（7）Select Case age，共有6条分支

S1：2 <= age <= 12，S2：12 < age <= 25，S3：25 < age <= 45，S4：45 < age <= 65，S5：65 < age <= 80，S6：Else

（2）语句（10）0 <= num <= 4，有2条分支

T2：True，F2：False

（3）语句（15）0 <= num <= 2，有2条分支

T3：True，F3：False

（4）语句（20）0 <= num <= 4，有2条分支

T4：True，F4：False

（5）语句（25）0 <= num <= 4，有2条分支

T5：True，F5：False

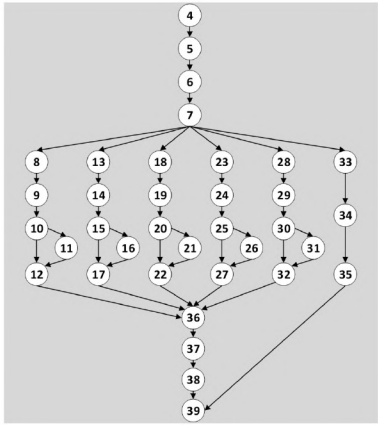
（6）语句（30）0 <= num <= 6，有2条分支

T6：True，F6：False

判定覆盖

- 11个测试用例

编号	输入		覆盖分支	通过路径
	age	num		
1	5	7	S1、F2	P1
2	5	1	S1、T2	P2
3	20	7	S2、F2	P3
4	20	1	S2、T2	P4
5	40	7	S3、F2	P5
6	40	1	S3、T2	P6
7	60	7	S4、F2	P7
8	60	1	S4、T2	P8
9	70	7	S5、F2	P9
10	70	1	S5、T2	P10
11	90	1	S6	P11



条件覆盖

- 由于每个判断只有一个条件，因此，这组测试用例满足条件覆盖。

编号	输入		覆盖分支	通过路径
	age	num		
1	5	7	S1、F2	P1
2	5	1	S1、T2	P2
3	20	7	S2、F2	P3
4	20	1	S2、T2	P4
5	40	7	S3、F2	P5
6	40	1	S3、T2	P6
7	60	7	S4、F2	P7
8	60	1	S4、T2	P8
9	70	7	S5、F2	P9
10	70	1	S5、T2	P10
11	90	1	S6	P11

- 本课程中，我们认为类似于“0 <= num <= 4”的表达为一个条件。

判定-条件覆盖

- 判定-条件覆盖是判定覆盖和条件覆盖的综合，这组测试用例满足判断-条件覆盖。

编号	输入		覆盖分支	通过路径
	age	num		
1	5	7	S1、F2	P1
2	5	1	S1、T2	P2
3	20	7	S2、F2	P3
4	20	1	S2、T2	P4
5	40	7	S3、F2	P5
6	40	1	S3、T2	P6
7	60	7	S4、F2	P7
8	60	1	S4、T2	P8
9	70	7	S5、F2	P9
10	70	1	S5、T2	P10
11	90	1	S6	P11

条件组合覆盖

- 由于每个判断只有一个条件，不存在条件组合问题；因此，这组测试用例满足条件组合覆盖。

编号	输入		覆盖分支	通过路径
	age	num		
1	5	7	S1、F2	P1
2	5	1	S1、T2	P2
3	20	7	S2、F2	P3
4	20	1	S2、T2	P4
5	40	7	S3、F2	P5
6	40	1	S3、T2	P6
7	60	7	S4、F2	P7
8	60	1	S4、T2	P8
9	70	7	S5、F2	P9
10	70	1	S5、T2	P10
11	90	1	S6	P11

路径覆盖

- 由于保险金程序中不存在循环，共存在11条路径，这组测试用例满足路径覆盖。

编号	输入		覆盖分支	通过路径
	age	num		
1	5	7	S1、F2	P1
2	5	1	S1、T2	P2
3	20	7	S2、F2	P3
4	20	1	S2、T2	P4
5	40	7	S3、F2	P5
6	40	1	S3、T2	P6
7	60	7	S4、F2	P7
8	60	1	S4、T2	P8
9	70	7	S5、F2	P9
10	70	1	S5、T2	P10
11	90	1	S6	P11

总结

- 针对保险金程序，采用逻辑覆盖指标，设计测试用例。