**实验十一 白盒测试用例设计（解析）**

# 题目



# 程序伪代码

 MAXSALERS = 0;

 ERRCODE = 0;

 if(ESIZE <= 0 || DSIZE <= 0)

 ERRCODE = 1;

 else

 {

 for(i = 1; i < DSIZE ; i++) //选出各部门中最大的销售额值

 if(SALES[i] > MAXSALES)

 MAXSALES = SALES[i];

 for(j = 1; j < DSIZE; j++) //循环找出销售额最大的部门

 if(SALES[j] == MAXSALES)

 {

 FOUND = false;

 for(int k = 1; k < ESIZE ; k++) //检查销售额最大部门的所有人员

 if(EMPTAB.DEPT(k) == DEPTTAB.DEPT(j))

 {

 FOUND = true;

 if(SALARY(k) >= 15000.00 or JOB(k) = 'M') //满足条件的人员加工资

 SALARY(k) = SALARY(k) + 100.00;

 else

 SALARY(k) = SALARY(k) + 200.00;

 }

 if(FOUND == false)

 ERRCODE == 2;

 }

 }

**注意：** 1）参数表中EMPTAB 为职员表，DEPTTAB为部门表；

 2 ) ESIZE为职员表长度，DSIZE为部门表长度；

 3）ERRCODE为出错码

# 程序流程图



# 解析

**步骤一：分析流程图，挑选判定**

对程序中判定进行分析（即粉红色图形if语句）

**判定如下：**

if(ESIZE <= 0 || DSIZE <= 0)

if(SALES[i] > MAXSALES)

if(SALES[j] == MAXSALES)

if(EMPTAB.DEPT(k) == DEPTTAB.DEPT(j))

if(SALARY(k) >= 15000.00 or JOB(k) = 'M')

if(FOUND == false)

**步骤二：分析判定真假值**

采用“判定覆盖”标准，使得上述6个判定都取到“真”“假”两种结果，则分析如下：

|  |  |  |
| --- | --- | --- |
| **判定** | **结果为“真”** | **结果为“假”** |
| if(ESIZE<=0||DSIZE<=0) | ESIZE或DSIZE为0 | ESIZE和DSIZE都大于0 |
| If(SALES[i]>MAXSALES) | 总会出现 | 部门表中，将销售量较小的某个部门放在后面 |
| If(SALES[j]==MAXSALES) | 总会出现 | 部门表中，各部门的销售量不全相等 |
| If(EMPTAB.DEPT(k)== DEPTTAB.DEPT(j)) | 职员表中，有职员在销售量最大的部分中工作 | 职员表中，有个职员不在销售量最大的部门工作 |
| If(SALARY(k)>=15000.00 or JOB(k)=’M’) | 销售量最大的部门中，有个职员的工资大于15000，或者是经理 | 销售量最大的部门中，有个职员不是经理，而且工资小于15000 |
| if（FOUND=false） | 销售量最大的部门中没有职员 | 销售量最大的部门中有职员 |

注意：对于流程图较复杂的情况，可以直接分析各个判定，让每个判定分别取一次“真”“假”值即可，即设计尽可能少的用例覆盖上表中列举的情况即可，不一定非要分析流程图。

**步骤三：设计“判定覆盖”的测试用例**

针对以上列举的判定分析，设计相应的用例如下：

**注意：**其中职务一栏，E表示一般职员，M表示经理

首先思考：输入？输出？

尽量少的用例覆盖！



**步骤四：分析用例结果**

虽然这两个用例满足“判定覆盖”标准，但是它们不能发现程序中许多其他错误，如：ERRCODE=0、职员是经理、且部门表为“空”情况。

**步骤五：分析判定中条件真假值**

****

**步骤六：设计“条件覆盖”的测试用例**



**步骤七：分析用例结果**

 虽然上图中用例设计满足了“条件覆盖”，但是它们不能将程序中每一条语句都执行，也会遗漏程序中许多其他的错误，如：ERRCODE=0,如果语句” (ESIZE<=0)||(DSIZE<=0)”写成”(ESIZE<=0) AND (DSIZE<=0)”,则该错误不能发现。