HW4

PB18111793

裴启智

1

某计算机语言中规定，“标识符是由字母开头，后跟字母或数字的任意组合构成。有效字符数为8个（有效标识符的字符数至少为8个），最大字符数为80个。不能是保留字”。请用等价类划分方法对标识符命名是否正确进行测试。要求给出等价类表，和具体的覆盖数据。

输入条件	有效等价类	无效等价类
标识符字符数	8~80个（1）	0个~7个（2），>80个（3）
标识符组成	字母（4），数字（5）	非字母数字字符（6）、保留字（7）
第一个字符	字母（8）	非字母（9）

以 C++ 语言为例，覆盖数据如下

覆盖数据	对应序号
`int abcd12345`	（1），（4），（5），（8）
`int abc`	（2）
`int abcd.....`（大于 80 个字符）	（3）
`int #a`	（6）
`int char`	（7）
`int 12a`	（9）

2

给出以下两个代码的**环路复杂度**，并给出所有的**独立路径**，同时对于每条独立路径给出**完整测试用例以及对应输出。**

要求：根据代码画出流图，求环路复杂度和独立路径。进行测试时，例如代码一，

- 如果有的独立路径不能单独进行测试，可以将这些路径作为另一个路径的一部分进行测试，或者适当修改某些变量的值进行测试
- 如果有的路径实际中并不存在，可以说明一下不进行测试
- 代码1

```
int i = 0;
int n = 4;
while (i < n-1) {
    j = i + 1;
    while(j < n) {
        if (A[i] < A[j]) {
            swap(A[i],A[j]);
        }
        j = j + 1
    }
    i = i + 1;
}
```

- 代码2

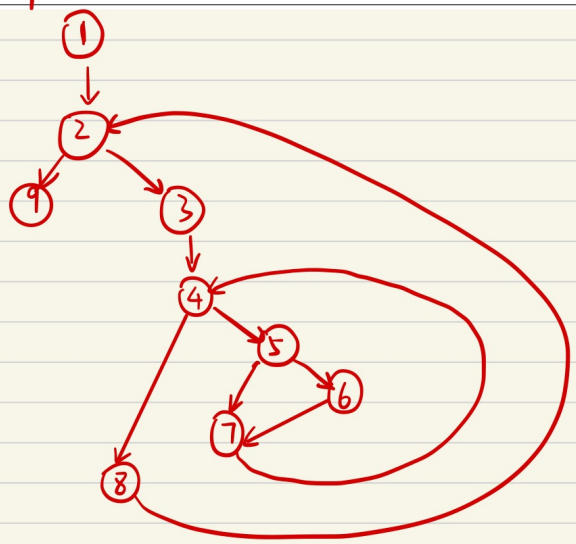
```
public int sum(int n, int upperbound) {
    int result, i;
    result = 0;
    i = 0;
    if (n < 0) {
        n = -n;
    }
    while(i<n && result <= upperbound) {
        i = i + 1;
        result = result + i;
    }
    if(result <= upperbound) {
        System.out.printin("The sum is " + result);
    }
    else {
        System.out.printin("The sum is too large!");
    }
    return result;
}
```

代码1

- 流图

代码一：

```
1 int i = 0;
2 int n = 4;
3 while (i < n-1) {
4     j = i + 1;
5     while(j < n) {
6         if (A[i] < A[j]) {
7             swap(A[i],A[j]);
8         }
9         j = j + 1
10    }
11    i = i + 1;
12 }
```



- 环路复杂度：V(G)= 判定结点数 + 1 = 3 + 1 = 4
- 独立路径
  - 1 2 9
  - 1 2 3 4 8 2 ...
  - 1 2 3 4 5 7 4 ...
  - 1 2 3 4 5 6 7 4 ...
- 测试用例
  - 1 2 9
    - 需要修改变量的值，使得 i >= n，可以修改 n 的初始值为 0，即

```
i = 5
n = 0
A 任意
```

- 1 2 3 4 8 2 ...
  - 当 3 执行结束后， 满足 i < n - 1 且 j = i + 1，那么 j < n 一定成立，不可能直接执行 8 。即该路径实际并不存在，故不作测试
- 1 2 3 4 5 7 4 ...
  - 保证 A 数组降序排列即可

```
i = 0
n = 4
A = {3, 2, 1, 0}
```

- 1 2 3 4 5 6 7 4 ...
  - 可以使 A 数组升序排列

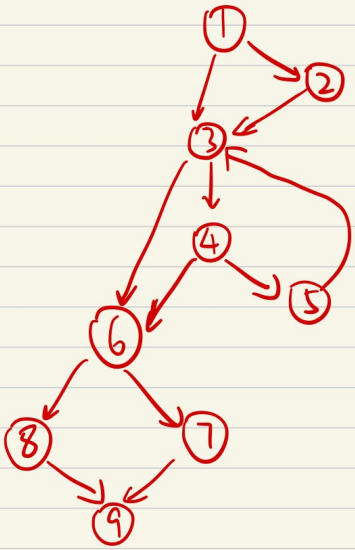
```
i = 0
n = 4
A = {0, 1, 2, 3}
```

代码2

- 流图

代码二：

```
1 public int sum(int n, int upperbound) {
2     int result, i;
3     result = 0;
4     i = 0;
5     if (n < 0) {
6         n = -n;
7     }
8     while(i<n && result <= upperbound) {
9         i = i + 1;
10        result = result + i;
11    }
12    if(result <= upperbound) {
13        System.out.printin("The sum is " + result);
14    }
15    else {
16        System.out.printin("The sum is too large!");
17    }
18    return result;
19}
```



- 环路复杂度：V(G)= 判定结点数 + 1 = 4 + 1 = 5
- 独立路径
  - 1 3 6 7 9
  - 1 3 6 8 9
  - 1 2 3 6 8 9
  - 1 3 4 6 8 9
  - 1 3 4 5 3 ...
- 测试用例
  - 1 3 6 7 9

```
n = 0
upperbound = 1
输出：The sum is 0
```

- 1 3 6 8 9

```
n = 0
upperbound = -1
输出：The sum is too large!
```

- 1 2 3 6 8 9

  需要修改 i 初始化的值

```
n = -1
i = 1
upperbound = - 1
输出：The sum is too large!
```

- 1 3 4 6 8 9

```
n = 1
upperbound = -1
输出：The sum is too large!
```

- 1 3 4 5 3 6 7 9

```
n = 1
upperbound = 1
输出：The sum is 1
```