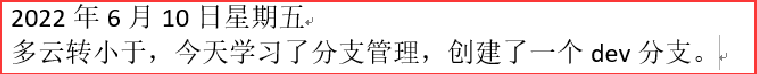
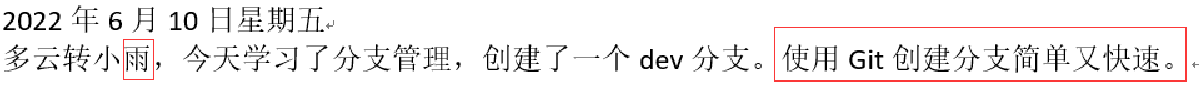
2023年3月23日星期四

天晴，使用git进行管理

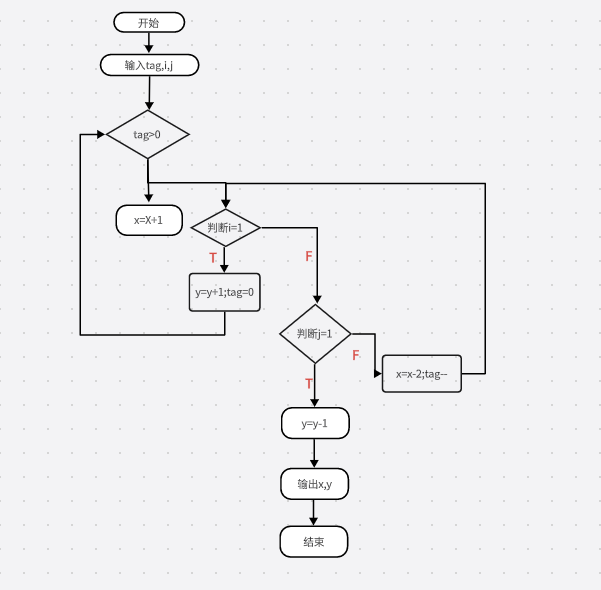




**一：运用路径分析的方法测试程序。 import java.util.Scanner; public class Example\_ { public static void main(String[] args) { int tag, i, j, x = , y = ; Scanner scanner = new Scanner(System.in); System.out.print("Enter an integer for tag: "); tag = scanner.nextInt（）; System.out.print("Enter an integer for i: "); i = scanner.nextInt（）; System.out.print("Enter an integer for j: "); j = scanner.nextInt（）; while (tag > ) { x += ; if (i == ) { y += ; tag = ; } else { if (j == ) y -= ; else x -= ; tag- -; } } System.out.println("x=" + x + ",y=" + y); } }**

**步骤：**

1. **阅读代码绘制流程图及控制流图;**



1. **用三种方式计算程序环形复杂度V(G);**
2. 控制流图法: V(G) = 4 + 1 - 9 = -4, 根据公式得到的环形复杂度是负数，不合理。
3. 回路法：V(G) = E - N + 2P = 12 - 9 + 2 × 2 = 7,根据该计算方法，该程序的环形复杂度为7。
4. 判定表法：环形复杂度V(G) = 条件判定数 + 循环判定数 + 跳转语句数量 + 1 = 6 + 1 + 3 + 1 = 11,根据判定表法，该程序的环形复杂度为11
5. **运用路径分析测试的方法分析源程序有多少独立路径，并设计测试用例表。**

6个独立路径:

1. 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-25
2. 1-2-3-4-5-6-7-8-9-10-11-12-13-14-18-19-25
3. 1-2-3-4-5-6-7-8-9-10-11-12-13-14-18-20-22-12-13-14-15-16-25
4. 1-2-3-4-5-6-7-8-9-10-11-12-13-14-18-20-21-22-12-13-14-15-16-25
5. 1-2-3-4-5-6-7-8-9-10-11-12-13-14-18-20-21-22-18-19-25
6. 1-2-3-4-5-6-7-8-9-10-11-12-13-14-18-20-22-18-19-25

根据以上6个独立路径，设计的测试用例表如下：

**二、运用路径分析的方法测试程序。**

int GetMaxDay(intyear,intmonth)

{

intmaxday=;

if(month>=&&month<=)

{

if( month == )

{

if(year%==)

{

if(year%==)

{

if(year%==)

maxday=;

else

maxday=;

}

else

maxday=;

}

else

maxday=;

}

else if(month==||month==||month==||month==)

maxday=;

else

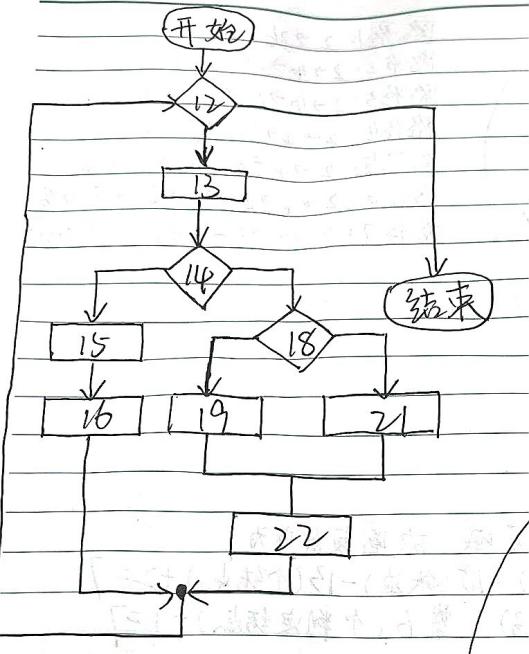
maxday=;

}

ruturn maxday;}

**说明：程序段中每行开头的数字（~）是对每条语句的编号。**

1. **画出程序的控制流图（用题中给出的语句编号表示）**

****

**（）请计算上述控制流程图的环形复杂度V(G)**

**控制流程图的圈复杂度V（G）为3+3+4+1＝11。**

**（）假设输入的取值范围是 < year < , 请使用基本路径测试法为变量 year 、 month 设计测试用例（写出 year 取值、 month 取值、 maxday 预期结果），使其满足基本路径覆盖要求。**

* 路径 1: start -> 2 -> 3 -> 4 -> 5a -> 12
* 路径 2: start -> 2 -> 3 -> 4 -> 5b -> 12
* 路径 3: start -> 2 -> 3 -> 4 -> 8a -> 9 -> 12
* 路径 4: start -> 2 -> 3 -> 4 -> 8b -> 10a -> 11 -> 12
* 路径 5: start -> 2 -> 3 -> 4 -> 8b -> 10b -> 12