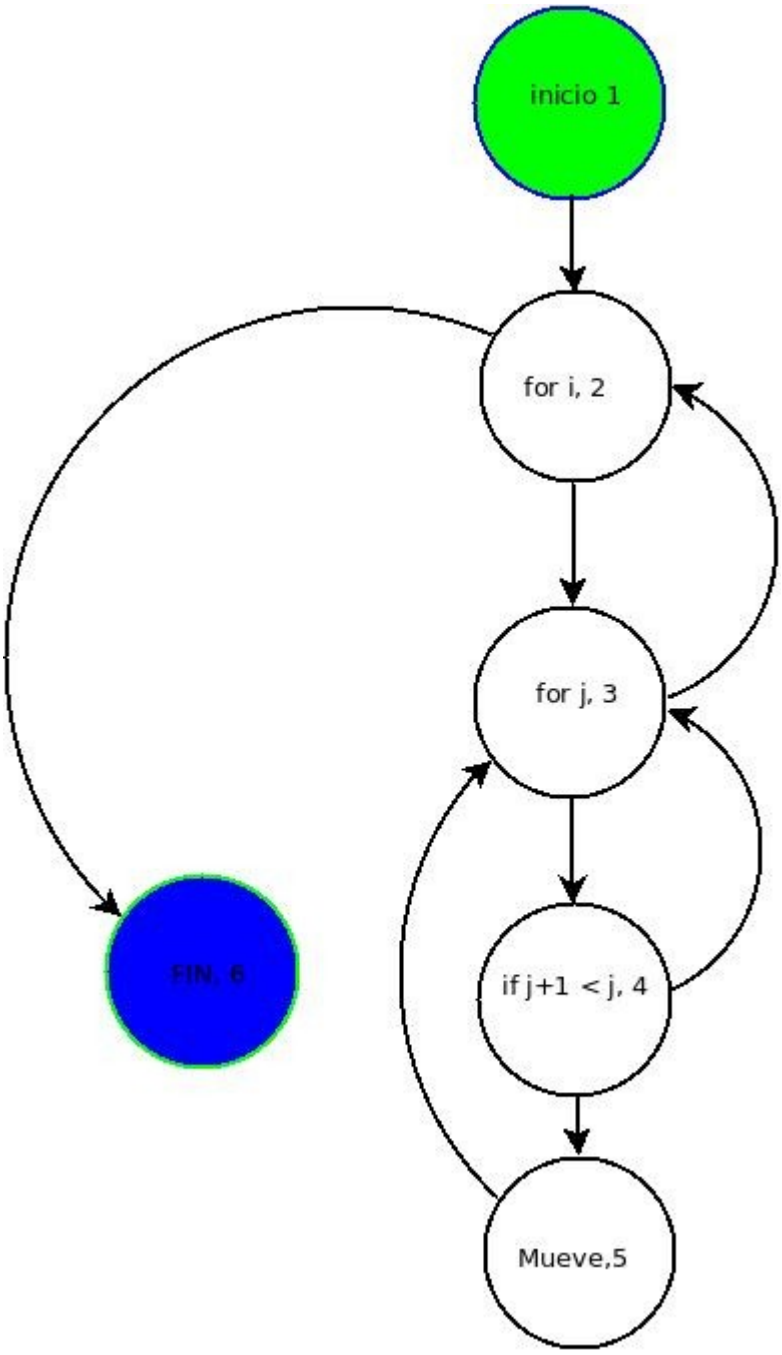
Javier González Rives

Tarea 3.1.Pruebas de Camino básico y Clases de equivalencia

Ejercicio 1. Dado el algoritmo de ordenación de la burbuja, realiza pruebas del camino básico.



Javier González Rives

Complejidad: El numero de nodos predicado es de 3, +1 nos da una complejidad ciclomática de 4, poco riesgo y fácil de testar con baja posibilidad de error.

Posibles caminos independientes:

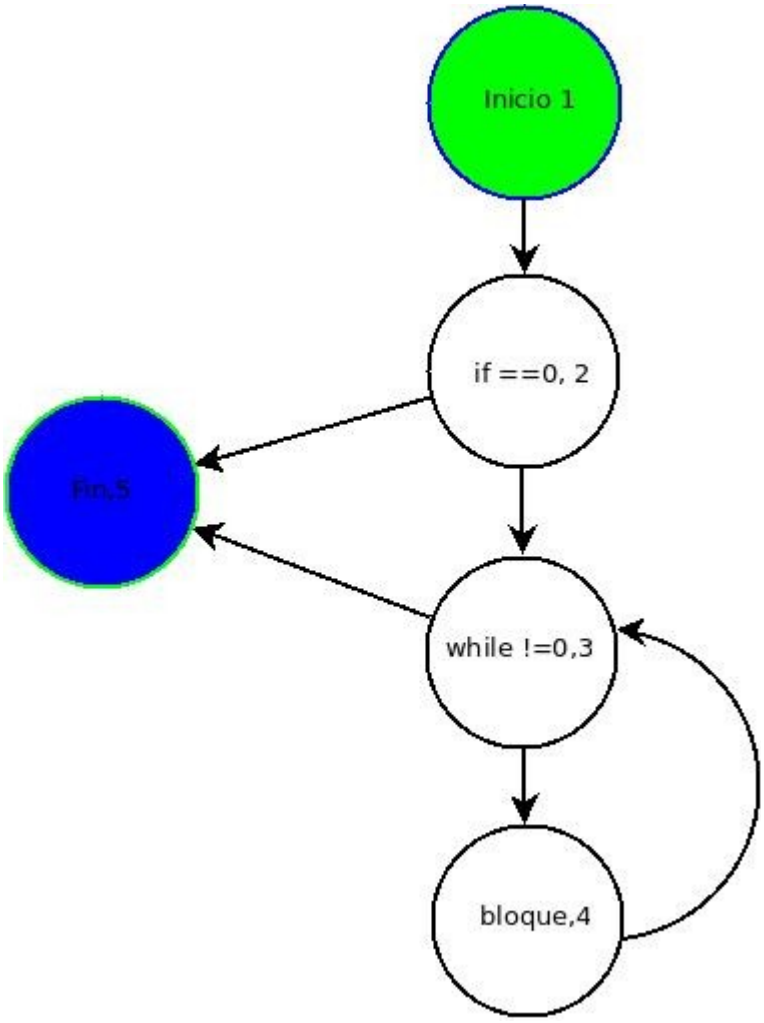
- 1,2,6.
- 1,2,3,2,6
- 1,2,3,4,3,2,6.
- 1,2,3,4,5,3,2,6.

Casos de prueba:

- Caso en que el array tenga una longitud 0, no entra en el primer bucle.
- Caso que el array no tenga 2 elementos, no entra en el segundo bucle.
- Caso que el array ya este ordenado, no se ejecuta la sentencia condicional.
- Caso que el array sea “correcto” y desordenado, se ejecutan todas las sentencias.

Javier González Rives

Ejercicio 2.1 Calcula las pruebas del camino básico del método mcd()



Complejidad: 2 nodos predicado + 1 = 3

Posibles caminos independientes:

- 1,2,5.
- 1,2,3,5.
- 1,2,3,4,3,5.

Casos de prueba:

- Denominador = 0.
- Denominador <> 0.

Javier González Rives

Ejercicio 2.2. Clases de equivalencia del constructor parametrizado.

La clase fracción tiene dos constructores parametrizados.

- Constructor que define como fracción un numero entero.
- Constructor que que tiene el numerador y el denominador por parámetro.

En el primer caso se acepta cualquier tipo de valor entero (int), en el segudo de igual modo con la diferencia que se pasan por parámetro dos valores enteros en lugar de uno.

En este caso todos los errores son de tipo lógico, dado que el posible error esta en no introducir un tipo de dato valido.

Logico 1ºconstructor	Introducir un valor entero v1	Introducir un valor no entero nv1 No introducir un valor nv2
Logico 2º constructor	Introducir un valor entero v1	Introducir un valor no entero nv1 No introducir un valor nv2
	Introducir un valor entero v2	Introducir un valor no entero nv3 No introducir un valor nv4

Para las pruebas habría que crear una clase principal para probar uno a uno los posibles errores, para no solapar los y no perderlos de vista al esconderse detrás de otro error o excepción, o un método mas eficiente seria la creación de test unitarios (Junit) para los métodos que nos permitan realizar todas las pruebas necesarias de forma rápida y con un listado de todo lo que ha fallado durante la ejecucion.

```
public  void FraccionTest() {
        int numerador = 5;
        int denominador = 2;
        //constructor solo un argumento
        Fraccion fraccion = new Fraccion(numerador);
        assertEquals("5" , fraccion.aString() );
        //constructor dos argumentos
        fraccion = new Fraccion(numerador,denominador);
        assertEquals("5/2", fraccion.aString());
        //intento elementos vacios
        //fraccion = new Fraccion((null);

    }
```

Hay que tener cuidado con las entradas desde teclado ya que pueden dar errores inesperados, introducir una cadena en lugar de un entero en este caso.