Universidad Nacional de Río Cuarto

Análisis y Diseño de Sistemas

Segundo parcial - Parte uno

Jeremias Parladorio
42.357.468

Juan Ignacio Alanis
41.815.640

17 de Mayo, 2020

Ejercicios

1. **Prueba de caja blanca**: Dado el siguiente método que cuenta las vocales de un arreglo:

```
1  static int cuentaVocales(char arreglo[]) {
2
3      int cont, i;
4
5      i = 0;
6
7      while (i < arreglo.length - 1) {
8
9          cont = i;
10
11         if (arreglo[i] == 'a' || arreglo[i] == 'e' || arreglo[i] == 'i
               ' || arreglo[i] == 'o' || arreglo[i] == 'u') {
12
13             i++;
14         }
15
16         cont++;
17     }
18
19     return cont;
20 }
```

   a) Apliquen la técnica de prueba estructural caja blanca utilizando el criterio de cobertura de caminos.

   b) En caso de que corresponda realicen lo necesario para poder aplicar el criterio solicitado (grafo de flujo de control, complejidad ciclomática, caminos independientes).

   c) Definan el conjunto T mínimo de casos de prueba.

   d) Describan el proceso seguido para realizar el testing indicando si se encontraron errores o no con cada caso de prueba, y cómo se procedió cuando se encontraron errores.

   e) En caso de encontrar fallas, propongan los cambios apropiados para reparar el o los errores detectados en el código a partir de la prueba realizada.

2. **Prueba de caja negra**: Dada una función que calcula el volumen de un cilindro

```
1 public double volumenCilindro(double alto, double radio)
```

   cuya fórmula de cálculo es

$$V = \pi r^2 h$$

   a) Apliquen la técnica de prueba funcional caja negra utilizando el criterio de de valor límite robusto.

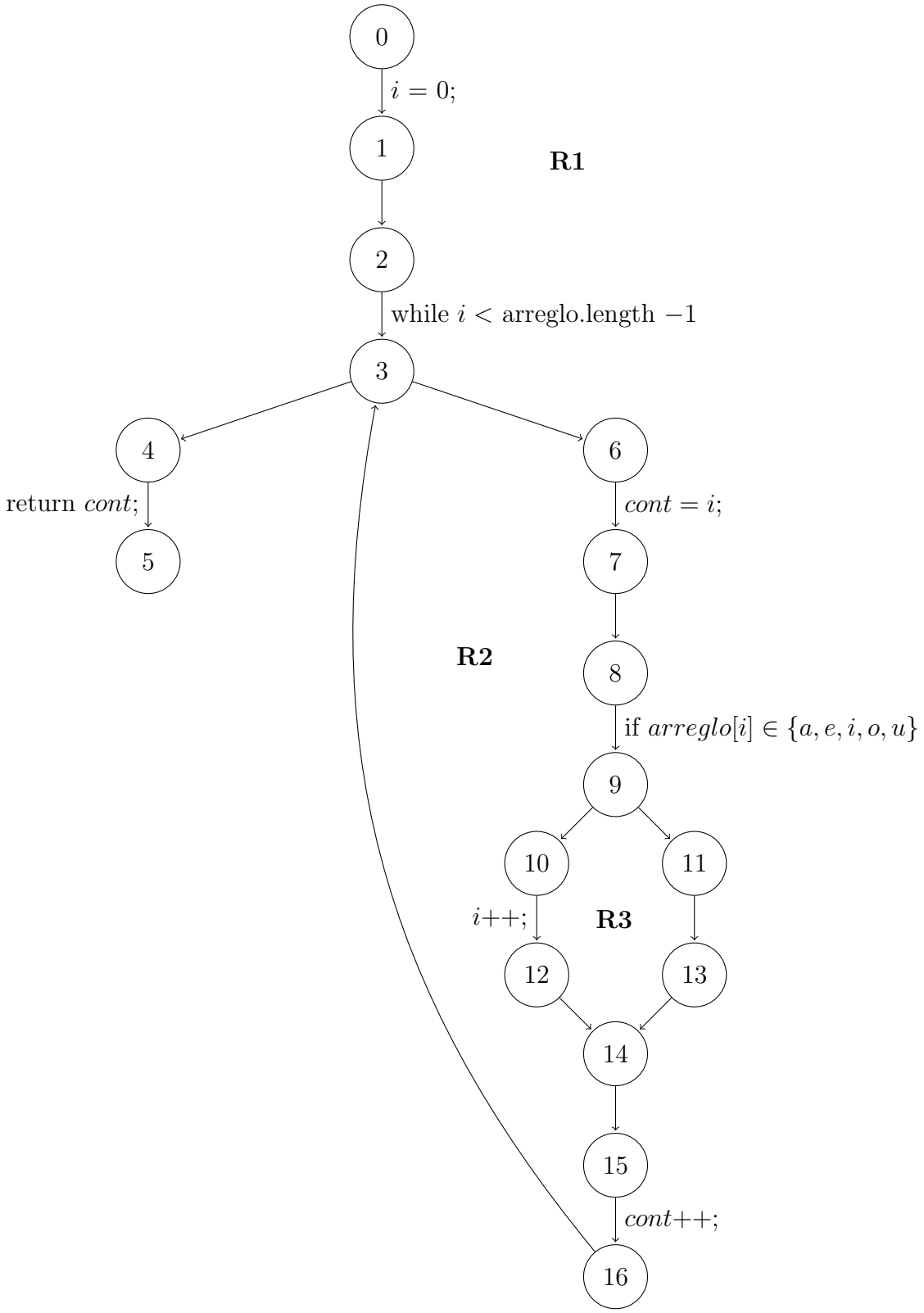
   b) Realicen lo necesario según lo que plantea el criterio solicitado para poder definir el conjunto de casos de prueba.

   c) Definan el conjunto T de casos de prueba.

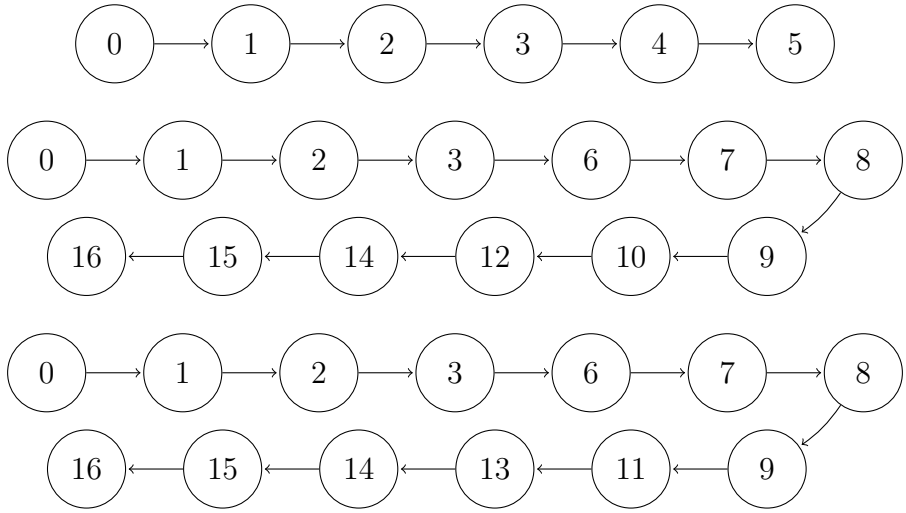
1. Prueba de caja blanca

1.1. Prueba de cobertura de camino

Para el algoritmo dado en el primer ejercicio, podemos formular el siguiente diagrama de flujo G:



Viendo las regiones del diagrama **R1**, **R2** y **R3** podemos concluir que la complejidad ciclomática es $V(G) = 3$, lo cual nos da también 3 caminos independientes:



Conjunto mínimo de casos de prueba

Dado $T = \{\langle e\rangle, \langle b, a\rangle, \langle a, e\rangle\}$, donde cada elemento de T es una instancia de *arreglo*, podemos observar lo siguiente para cada uno de sus elementos:

- $T[0] = \langle e\rangle$: No entra al ciclo ya que la condición del while para este caso sería $0 < 1 - 1$, lo cual es falso. Retorna *cont*. Recorre el camino independiente 1.

Resultado esperado	Resultado obtenido
1	*cont* (sin inicializar)

- $T[1] = \langle b, a\rangle$: Con esta instancia de *arreglo* se entra al ciclo pero nunca se llega a los nodos 10 y 12 (por falsedad del if), por lo que i es siempre 0 y queda en un ciclo infinito. Recorre el camino independiente 2.

Resultado esperado	Resultado obtenido
1	ciclo infinito

- $T[2] = \langle a, e\rangle$: Esta instancia de arreglo entra al while, cuenta la vocal a y luego sale. Nunca evalúa el último elemento del arrelgo. Recorre el camino independiente 3.

Resultado esperado	Resultado obtenido
2	1

En conclusión, incluso con el conjunto mínimo de caso de prueba, el programa no funciona correctamente, ya que falla en todas sus instancias. Por lo tanto, proponemos la siguiente solución:

```
1  static int cuentaVocales(char[] arreglo) {
2
3      int cont = 0;
4
5      int i = 0;
6
7      while (i < arreglo.length) {
8
9          if (arreglo[i] == 'a' || arreglo[i] == 'e' || arreglo[i] == 'i' ||
               arreglo[i] == 'o' || arreglo[i] == 'u') {
10
11             cont++;
12         }
13
14         i++;
15     }
16
17     return cont;
18 }
```

2. Prueba de caja negra

2.1. Prueba de robustez

Sea $h \in [0, 100]$ y $r \in [0, 100]$. Se establecen los siguientes valores límites:

1. $min - 1 = -1$
2. $min = 0$
3. $min + 1 = 1$
4. $nom = 50$
5. $max - 1 = 99$
6. $max = 100$
7. $max + 1 = 101$

Caso	h	r	Salida esperada
1	50	0	0
2	50	1	157.079
3	50	50	392699.081
4	50	99	1539537.479
5	50	100	1570796.326
6	50	-1	Error[1]
7	50	101	1602369.332[2]
8	0	50	0
9	1	50	7853.981
10	50	50	392699.081
11	99	50	777544.181
12	100	50	785398.163
13	-1	50	Error[3]
14	101	50	793252.145[4]

[1,3] Ni el radio ([1]) ni la altura ([3]) de un cilindro pueden ser negativos.
[2,4] Si bien se sale del rango establecido para h y para r, en este caso el programa no debería fallar sino hasta que el resultado (o alguno de los parámetros) estén por encima del rango de representación establecido por el lenguaje de programación utilizado.