PROGRAMACIÓN 2. Curso 2017-18. Grupo de tarde

3ª prueba voluntaria de evaluación

Esta es la tercera y última prueba voluntaria que se plantea en la asignatura ***Programación 2*** para la evaluación de los alumnos matriculados en el grupo de tarde. Tiene un valor de **10 puntos** y debe ser resuelta individualmente.

Primer problema [3.5 puntos]

Supongamos que se ejecuta el código mostrado a continuación partiendo de un estado inicial que satisfaga su precondición $(NUM\alpha \in [0, n-1].v[\alpha] > 0) > 1$. Se pide escribir las aserciones **`P1`**, **`P2`**, **`P3`**, **`P4`** y **`P5`** más fuertes que se satisfagan al alcanzar la ejecución del código los puntos en los que han sido escritas.

```
/*
 * Pre: (NUM alfa EN [0,n-1]. v[alfa] > 0) > 1
 * Post: uno > 0 AND otro > 0 AND
 *         (EX alfa EN [0,n-1].(EX beta EN [0,n-1].
 *                                   alfa != beta AND v[alfa] = uno AND v[beta] = otro) )
 */
void dosPositivos (const int v[], const int n, int& uno, int& otro) {
    int i = n - 1;
    while (v[i] <= 0) {
        // P1
        i = i - 1;
    }
    uno = v[i];
    // P2
    i = i - 1;
    while (v[i] <= 0) {
        // P3
        i = i - 1;
    }
    // P4
    otro = v[i];
    // P5
}
```

Segundo problema [6.5 puntos]

Demostrar formalmente que el diseño iterativo de la función **`elevar`**`(x,n)` presentado a continuación es correcto. Se sugiere apoyar el conjunto de pruebas que constituyen la demostración en el predicado invariante del bucle $INV : n \geq 0 \ \wedge \ i \ \geq \ 0 \ \wedge \ i \leq n \ \wedge \ x^n = r \times (x^i)^e \ \wedge \ pot = x^e$. Al calificar este problema se tendrá muy en cuenta la claridad con la que se presenten las pruebas que contituyen la demostración de la corrección del diseño, la legibilidad de dichas pruebas y la explicación de las justificaciones de aquellas pruebas que no sean inmediatas.

```
// Pre: n ≥ 0
// Post: elevar(x, n) = x^n
double elevar (const double x, const int n) {
    int i = n;
    int e = 1;
    double r = 1.0;
    double pot = x;
    // INV : n ≥ 0 ∧ i ≥ 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e
    while ( i! = 0 ) {
        if ( i % 2 ! = 0 ) {
            r = pot * r;
        }
        i = i / 2;
        e = 2 * e;
        pot = pot * pot;
    }
    return r;
}
```

Modo de entrega de la prueba

Cada alumno preparará un documento en papel, manuscrito o impreso, con la resolución de los dos problemas anteriores. El documento será entregado al profesor de la asignatura (al final de una de sus clases o en su despacho) con límite el **`lunes 28 de mayo de 2018 a las 17 horas`**. No se admitirán trabajos presentados de otro modo, ni entregas fuera de plazo, ni trabajos plagiados total o parcialmente.

La prueba corregida y calificada se pasará a recoger por el despacho del profesor Martínez (D1.08 del edificio Ada Byron) a partir del lunes 4 de junio en horario de mañana.

Una solución del primer problema

Los predicados pedidos **P1**, **P2**, **P3**, **P4** y **P5** se han anotado sobre el propio código de la función **`dosPositivos`**`(v,n,uno,otro)` para facilitar su comprensión. Estos son los predicados más fuertes que se satisfacen en los puntos del código en los que han sido escritos cuando la función se ejecuta desde un estado inicial que satisface su precondición.

```
/*
 * Pre:  (NUM alfa EN [0,n-1]. v[alfa] > 0) > 1
 * Post: uno > 0 AND otro > 0 AND
 *         (EX alfa EN [0,n-1].(EX beta EN [0,n-1].
 *                                  alfa != beta AND v[alfa] = uno AND v[beta] = otro) )
 */
void dosPositivos (const int v[], const int n, int& uno, int& otro) {
    int i = n - 1;
    while (v[i] <= 0) {
        // P1: (NUM alfa EN [0,n-1]. v[alfa] > 0) > 1 AND
        //       (PT alfa EN [i, n-1]. v[alfa] <= 0) AND i > 0 AND i <= n-1
        i = i - 1;
    }
    uno = v[i];
    // P2: (NUM alfa EN [0,n-1]. v[alfa] > 0) > 1 AND
    //       (PT alfa EN [i+1, n-1]. v[alfa] <= 0) AND
    //       uno = v[i] AND v[i] > 0 AND i > 0 AND <= n-1
    i = i - 1;
    while (v[i] <= 0) {
        // P3: (NUM alfa EN [0,n-1]. v[alfa] > 0) > 1 AND
        //       uno = v[I_PRIMERO] AND uno > 0 AND I_PRIMERO <= n-1 AND
        //       (PT alfa EN [I_PRIMERO+1,n-1]. v[alfa] <= 0) AND
        //       (PT alfa EN [i,I_PRIMERO-1]. v[alfa] <= 0) AND i >= 0 AND i < I_PRIMERO
        i = i - 1;
    }
    // P4: (NUM alfa EN [0,n-1]. v[alfa] > 0) > 1 AND
    //       uno = v[I_PRIMERO] AND uno > 0 AND I_PRIMERO <= n-1 AND
    //       (PT alfa EN [I_PRIMERO+1,n-1]. v[alfa] <= 0) AND
    //       (PT alfa EN [i+1,I_PRIMERO-1]. v[alfa] <= 0) AND
    //        v[i] > 0 AND i >= 0 AND i < I_PRIMERO
    otro = v[i];
    // P5: uno > 0 AND uno = v[I_PRIMERO] AND I_PRIMERO <= n-1 AND
    //       (PT alfa EN [I_PRIMERO+1,n-1]. v[alfa] <= 0)
    //       (PT alfa EN [i+1,I_PRIMERO-1]. v[alfa] <= 0) AND
    //        v[i] > 0 AND otro = v[i]AND i >= 0 AND i < I_PRIMERO
}
```

Una solución del segundo problema

Las pruebas que constituyen la demostración de la corrección del diseño de la función **`elevar`**`(x,n)` se han anotado sobre el propio código de la función, a excepción de la prueba de la terminación del bucle, que se presenta más adelante.

```
// Pre: n ≥ 0
// Post: elevar(x, n) = x^n
double elevar (const double x, const int n) {
    // n ≥ 0
    // ⇒  [1]
    // n ≥ 0 ∧ n ≥ 0 ∧ n ≤ n ∧ x^n = 1 × (x^n)^1 ∧ x = x^1
    int i = n;
    int e = 1;
    double r = 1.0;
    double pot = x;
    // INV : n ≥ 0 ∧ i ≥ 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e
    while ( i! = 0 ) {
        // n ≥ 0 ∧ i > 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e
        if ( i % 2 != 0 ) {
            // n ≥ 0 ∧ i > 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e ∧ i % 2 ≠ 0
            // ⇒  [2]
            // n ≥ 0 ∧ i/2 ≥ 0 ∧ i/2 ≤ n ∧ x^n = pot × r × (x^(i/2))^2e ∧ pot^2 = x^2e
            r = pot * r;
        }
        else {
            // n ≥ 0 ∧ i > 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e ∧ i % 2 = 0
            // ⇒  [3]
            // n ≥ 0 ∧ i/2 ≥ 0 ∧ i/2 ≤ n ∧ x^n = r × (x^(i/2))^2e ∧ pot^2 = x^2e
        }
        // n ≥ 0 ∧ i/2 ≥ 0 ∧ i/2 ≤ n ∧ x^n = r × (x^(i/2))^2e ∧ pot^2 = x^2e
        i = i / 2;
        e = 2 * e;
        pot = pot * pot;
        // n ≥ 0 ∧ i ≥ 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e
    }
    // n ≥ 0 ∧ i = 0 ∧ x^n = r × (x^i)^e ∧ pot = x^e
    // ⇒  [4]
    // r = x^n
    return r;
}
```

Para completar la demostración de la corrección del diseño de la función **`elevar`**`(x,n)` se han anotado sobre el propio código de la función la prueba de la terminación del bucle, construida a partir de la función de cota $f_{cota}(i) = i$.

```
// Pre: n ≥ 0
// Post: elevar(x, n) = x^n
double elevar (const double x, const int n) {
    // n ≥ 0
    int i = n;
    int e = 1;
    double r = 1.0;
    double pot = x;
    // INV : n ≥ 0 ∧ i ≥ 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e
    while ( i! = 0 ) {
        // Función de cota: f_cota(i) = i
        // i = A ∧ f_cota(inicio iteración) = i = A
        if ( i % 2 ! = 0 ) {
            r = pot * r;
        }
        i = i / 2;
        e = 2 * e;
        pot = pot * pot;
        // INV : n ≥ 0 ∧ i ≥ 0 ∧ i ≤ n ∧ x^n = r × (x^i)^e ∧ pot = x^e

        // i = A/2 ∧ f_cota(fin iteración) = i = A/2

        // La iteración termina ya que se satisfacen estas dos condiciones:
        // 1. La función de cota decrece en cada iteración:
        //          Si tenemos en cuenta que INV ∧ i! = 0 ⇒ A > 0 entonces A > A/2
        //          con lo que queda probado que f_cota(inicio iteración) > f_cota(fin iteración)
        // 2. La función de cota está acotada inferiormente en el conjunto de los enteros, ℤ,
        //    por el valor 0:
        //          INV ⇒ i ≥ 0  y i ≥ 0 ⇒ f_cota(i) = i ≥ 0
    }
    return r;
}
```

Algunos comentarios y justificaciones sobre las pruebas presentadas anteriormente que constituyen la demostración de la corrección del diseño de la función **`elevar`**`(x,n)`:

- La satisfacción de la relación [1] es inmediata y prueba la corrección del código que precede al bucle.
- La satisfacción de la relación [2] es consecuencia de que el valor de i es impar y, por lo tanto, se satisface que $(i/2) \times 2e = (i-1) \times e$ y que $(x^i)^e = pot \times (x^{i/2})^{2e} = pot \times x^{(i-1)\times e} = x^e \times x^{(i-1)\times e} = x^{i\times e}$ ya que $pot = x^e$ y prueba la corrección del código a iterar cuando se satisface la condición $i \,\%\, 2 \neq 0$.
- La satisfacción de la relación [3] es consecuencia de que el valor de i es par y, por lo tanto, se satisface que $(i/2) \times 2e = i \times e$ y que $(x^i)^e = (x^{i/2})^{2e} = x^{i\times e}$ y prueba la corrección del código a iterar cuando no se satisface la condición $i \,\%\, 2 \neq 0$.
- La satisfacción de la relación [4] es consecuencia de que el valor de $i = 0$ y, por lo tanto, $x^n = r \times (x^i)^e \equiv x^n = r \times (x^0)^e \equiv x^n = r \times 1^e \equiv x^n = r$. De esta forma queda probada la corrección del código que sigue al bucle.
- La terminación del bucle se ha probado haciendo uso de la función de cota $f_{cota}(i) = i$ que toma valores decrecientes en el conjunto de los enteros $\mathbb{Z}$ y sus valores están acotados inferiormente por el valor 0.