Programación 2

Análisis de la corrección: escritura de aserciones en un algoritmo

Problemas 11

En los problemas que siguen se propone la escritura de los predicados "más fuertes" que satisfacen los datos de cada algoritmo en diversos puntos de su código, cuando ese código se ejecuta desde un estado inicial que satisface su precondición.

Con su resolución se pretende:

1. **Comprender mejor el diseño de cada algoritmo**
2. **Familiarizarnos con la escritura de predicados formales**
3. **Preparar el terreno para la demostración formal de la corrección de algoritmos**

Problema 1. Escribir los predicados más fuertes **P1**, **P2**, **P3** y **P4** que se satisfacen en los puntos del código donde figuran:

```
// n > 0 AND n <= #v
double r = v[0];
int i = 0;
// P1
while (i != n - 1) {
    i = i + 1;
    // P2
    if (v[i] > r) {
        r = v[i];
    }
    // P3
}
// P4
// r = (Máx α∈[0,n-1]. v[α])
```

```
// n > 0 ∧ n <= #v
double r = v[0];
int i = 0;
// P1: n > 0 ∧ n <= #v ∧ i = 0 ∧ r = v[0]
while (i != n - 1) {
    i = i + 1;
    // P2: n > 0 ∧ n <= #v ∧ i > 0 ∧ i ≤ n - 1 ∧
    //     r = (Máx α∈[0,i-1]. v[α])
    if (v[i] > r) {
        r = v[i];
    }
    // P3: n > 0 ∧ n <= #v ∧ i > 0 ∧ i ≤ n - 1 ∧
    //     r = (Máx α∈[0,i]. v[α])
}
// P4: n > 0 ∧ n <= #v ∧ i = n - 1
//     ∧ r = (Máx α∈[0,n-1]. v[α])
// r = (Máx α∈[0,n-1]. v[α])
```

Problema 2. Escribir los predicados más fuertes **P1**, **P2**, **P3** y **P4** que se satisfacen en los puntos del código donde figuran:

```
// n ≥ 0 ∧ n <= #v
int cuenta = 0;
int i = n - 1;
// P1
while (i >= 0) {
   // P2
   if (v[i] > 0.0) {  cuenta = cuenta + 1;  }
   // P3
   i = i - 1;
}
// P4
// cuenta = (Núm α∈[0,n-1]. v[α] > 0.0)
```

```
// n ≥ 0 ∧ n <= #v
 int cuenta = 0;
 int i = n - 1;
 // P1: n ≥ 0 ∧ n <= #v ∧ i = n - 1 ∧ cuenta = 0
 while (i >= 0) {
    // P2: n ≥ 0 ∧ n <= #v ∧ i ≥ 0 ∧ i ≤ n - 1 ∧
    //     cuenta = (Núm α∈[i+1,n-1]. v[α] > 0.0)
    if (v[i] > 0.0) {  cuenta = cuenta + 1;  }
    // P3: n ≥ 0 ∧ n <= #v ∧ i ≥ 0 ∧ i ≤ n - 1 ∧
    //     cuenta = (Núm α∈[i,n-1]. v[α] > 0.0)
    i = i - 1;
 }
 // P4: n > 0 ∧ n <= #v ∧ i = - 1 ∧
 //     cuenta = (Núm α∈[0,n-1]. v[α] > 0.0)
 // cuenta = (Núm α∈[0,n-1]. v[α] > 0.0)
```

Problema 3. Escribir los predicados más fuertes **P1**, **P2**, **P3** y **P4** que se satisfacen en los puntos del código donde figuran:

```
// n ≥ 0 ∧ n <= #v
bool esta = false;
int i = 0;
// P1
while (!esta && i != n) {
   // P2
   if (v[i] == dato) {  esta = true;  }
   else { i = i + 1; }
   // P3
}
// P4
if (!esta) { i = -107; }
// ((∃α∈[0,n-1]. v[α] = dato) → v[i] = dato) ∧
// ((∀α∈[0,n-1]. v[α] ≠ dato) → i < 0)
```

```
// n ≥ 0 ∧ n <= #v
bool esta = false;
int i = 0;
// P1: n ≥ 0 ∧ n <= #v ∧ i = 0 ∧ ¬esta
while (!esta && i != n) {
   // P2: n ≥ 0 ∧ n <= #v ∧ i ≥ 0 ∧ i < n ∧ ¬esta ∧
   //     (∀α∈[0,i-1]. v[α] ≠ dato)
   if (v[i] == dato) {  esta = true;  }
   else { i = i + 1; }
   // P3: n ≥ 0 ∧ n <= #v ∧ i ≥ 0 ∧ (¬esta → i ≤ n) ∧
   //     (esta ↔ v[i] = dato ∧ i < n) ∧
   //     (∀α∈[0,i-1]. v[α] ≠ dato)
}
// P4: n ≥ 0 ∧ n <= #v ∧ i ≥ 0 ∧ (¬esta → i = n) ∧
//     ∧ (esta ↔ v[i] = dato ∧ i < n)
//     ∧ (∀α∈[0,i-1]. v[α] ≠ dato)
if (!esta) { i = -107; }
// ((∃α∈[0,n-1]. v[α] = dato) → v[i] = dato) ∧
// ((∀α∈[0,n-1]. v[α] ≠ dato) → i < 0)
```

Problema 4. Escribir los predicados más fuertes **P1**, **P2** , **P3** , **P4** y **P5** que se satisfacen en los puntos del código donde figuran:

```
// Pre: n ≥ 0
// Post: resultado = (∏α∈[1,n]. α)
void factorial (const int n, int& resultado) {
   if (n == 0) {  // P1
      resultado = 1;
      // P2
   }
   else { // P3
      factorial(n-1, resultado);
      // P4
      resultado = n * resultado;
      // P5
   }
}
```

```
// Pre: n ≥ 0
// Post: resultado = (∏α∈[1,n]. α)
void factorial (const int n, int& resultado) {
   if (n == 0) {
      // P1: n = 0
      resultado = 1;
      // P2: n = 0 ∧ resultado = 1
   }
   else {
      // P3: n > 0
      factorial(n-1, resultado);
      // P4: n > 0 ∧ resultado = (∏α∈[1,n-1]. α)
      resultado = n * resultado;
      // P5: n > 0 ∧ resultado = (∏α∈[1,n]. α)
   }
}
```

Problema 5. Escribir los predicados más fuertes **P1**, **P2** , **P3** , **P4** y **P5** que se satisfacen en los puntos del código donde figuran:

```
// Pre: n ≥ 0 ∧ n <= #v ∧ v = Vo
// Post: (∀α∈[0,n-1]. (Vo[α] ≥ 0.0 → v[α] = Vo[α]) ∧
//                    (Vo[α] ≤ 0.0 → v[α] = -Vo[α]) )
void abs (double v[], const int n) {
   if (n > 0) {  // P1
      if (v[n-1] < 0.0) {  // P2
          v[n-1] = - v[n-1];
          // P3
      }
      // P4
      abs(v, n - 1);
      // P5
   }
}
```

```
// Pre: n ≥ 0 ∧ n <= #v ∧ v = Vo
// Post: (∀α∈[0,n-1]. (Vo[α] ≥ 0.0 → v[α] = Vo[α]) ∧
//                    (Vo[α] ≤ 0.0 → v[α] = -Vo[α]) )
void abs (double v[], const int n) {
   if (n > 0) {
      // P1: n > 0 ∧ n <= #v ∧ (∀α∈[0,n-1]. v[α] = Vo[α])
      if (v[n-1] < 0.0) {
          // P2: n > 0 ∧ n <= #v ∧ v[n-1] < 0.0
          //     ∧ (∀α∈[0,n-1]. v[α] = Vo[α])
          v[n-1] = - v[n-1];
          // P3: n > 0 ∧ n <= #v ∧ Vo[n-1] < 0.0 ∧
          //     v[n-1] = -Vo[n-1] ∧ (∀α∈[0,n-2]. v[α] = Vo[α])
      }
      // P4: n > 0 ∧ n <= #v ∧ (∀α∈[0,n-2]. v[α] = Vo[α])
      //     ∧ (Vo[n-1] ≥ 0.0 →  v[n-1] = Vo[n-1])
      //     ∧ (Vo[n-1] ≤ 0.0 →  v[n-1] = -Vo[n-1])
      abs(v, n - 1);
      // P5:
   }
}
```

```
// Pre: n ≥ 0 ∧ n <= #v ∧ v = Vo
// Post: (∀α∈[0,n-1]. (Vo[α] ≥ 0.0 → v[α] = Vo[α]) ∧
//                    (Vo[α] ≤ 0.0 → v[α] = -Vo[α]) )
void abs (double v[], const int n) {
   if (n > 0) {
      // P1: n > 0 ∧ n <= #v ∧ (∀α∈[0,n-1]. v[α] = Vo[α])
      if (v[n-1] < 0.0) {
          // P2: n > 0 ∧ n <= #v ∧ v[n-1] < 0.0 ∧ …
          v[n-1] = - v[n-1];
          // P3: n > 0 ∧ n <= #v ∧ Vo[n-1] < 0.0 ∧ …
      }
      // P4: n > 0 ∧ n <= #v ∧ (∀α∈[0,n-2]. v[α] = Vo[α])
      //     ∧ (Vo[n-1] ≥ 0.0 →  v[n-1] = Vo[n-1])
      //     ∧ (Vo[n-1] ≤ 0.0 →  v[n-1] = -Vo[n-1])
      abs(v, n - 1);
      // P5: n > 0 ∧ n <= #v ∧
      //     (∀α∈[0,n-1]. (Vo[α] ≥ 0.0 → v[α] = Vo[α]) ∧
      //                  (Vo[α] ≤ 0.0 → v[α] = -Vo[α]) )
   }
}
```

Problema 6. Escribir los predicados más fuertes **P1**, **P2**, **P3**, **P4** y **P5** que se satisfacen en los puntos del código donde figuran:

```
/*
 * Pre: v = Vo ∧ n >= 1 ∧ n <= #v
 * Post: v[0] = Vo[n-1] ∧ (∀α∈[1,n-1]. v[α] = Vo[α-1])
 */
template <typename T>
void rotar (T v[], const int n) {
    T ultimo = v[n-1]; // P1
    int i = n - 1;     // P2
    while (i != 0) {
        // P3
        v[i] = v[i-1];  i = i - 1;
        // P4
    }
    // P5
    v[0] = ultimo;
}
```

```
// Pre: v = Vo ∧ n >= 1 ∧ n <= #v
// Post: v[0] = Vo[n-1] ∧ (∀α∈[1,n-1]. v[α] = Vo[α-1])
template <typename T>
void rotar (T v[], const int n) {
    T ultimo = v[n-1];
    // P1: n > 0 ∧ n <= #v ∧ ultimo = Vo[n-1]
    //     ∧ (∀α∈[0,n-1]. v[α] = Vo[α])
    int i = n - 1;
    // P2 : n > 0 ∧ n <= #v ∧ i = n - 1 ∧ ultimo = Vo[n-1] ∧
    //      (∀α∈[0,n-1]. v[α] = Vo[α])
    while (i != 0) {
        // P3: n > 0 ∧ n <= #v ∧ i > 0 ∧ i ≤ n - 1 ∧
        //     ultimo = Vo[n-1] ∧(∀α∈[0,i]. v[α] = Vo[α]) ∧
        //     (∀α∈[i+1,n-1]. v[α] = Vo[α-1])
        v[i] = v[i-1];  i = i - 1;
        // P4: …
    }
    // P5: …
    v[0] = ultimo;
}
```

```
 // Pre: v = Vo ∧ n >= 1 ∧ n <= #v
 // Post: v[0] = Vo[n-1] ∧ (∀α∈[1,n-1]. v[α] = Vo[α-1])
template <typename T>
void rotar (T v[], const int n) {
     …
    while (i != 0) {
        // P3: n > 0 ∧ n <= #v ∧ i > 0 ∧ i ≤ n - 1
        //     ∧ ultimo = Vo[n-1] ∧ (∀α∈[0,i]. v[α] = Vo[α])
        //     ∧ (∀α∈[i+1,n-1]. v[α] = Vo[α-1])
        v[i] = v[i-1];  i = i - 1;
        // P4: n > 0 ∧ n <= #v ∧ i ≥ 0 ∧ i < n - 1
        //     ∧ ultimo = Vo[n-1] ∧ ∧ (∀α∈[0,i]. v[α] = Vo[α])
        //     ∧ (∀α∈[i+1,n-1]. v[α] = Vo[α-1])
    }
    // P5: n > 0 ∧ n <= #v ∧ i = 0 ∧ ultimo = Vo[n-1]
    //     ∧ v[0] = Vo[0] ∧ (∀α∈[1,n-1]. v[α] = Vo[α-1])
    v[0] = ultimo;
}
```