# Práctica 4

Corrección de Algoritmos

Introducción a la Computación

$1^{er}$ cuatrimestre 2019

**Ejercicio 1.** Demostrar que cada uno de los siguientes algoritmos es correcto respecto de su especificación.

a) Encabezado: $identidad : x \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0\}$
Poscondición: $\{RV = x_0\}$
Algoritmo:
$RV \leftarrow x$

b) Encabezado: $duplicar : x \in \mathbb{Z} \to \emptyset$.
Precondición: $\{x = x_0\}$
Poscondición: $\{x = x_0 \times 2\}$
Algoritmo:
$x \leftarrow x \times 2$

c) Encabezado: $suma : x \in \mathbb{Z} \times y \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0 \wedge y = y_0\}$
Poscondición: $\{RV = x_0 + y_0\}$
Algoritmo:
$RV \leftarrow x$
$RV \leftarrow RV + y$

d) Encabezado: $positivo : x \in \mathbb{Z} \to \mathbb{B}$
Precondición: $\{x = x_0\}$
Poscondición: $\{(x_0 > 0 \wedge RV = true) \vee (x_0 \leq 0 \wedge RV = false)\}$
Algoritmo:

```
if (x > 0) {
   RV ← true
} else {
   RV ← false
}
```

e) Igual al punto d), pero cambiando la poscondición por la siguiente:
Poscondición: $\{(x_0 > 0 \Rightarrow RV = true) \wedge (x_0 \leq 0 \Rightarrow RV = false)\}$

**Ejercicio 2.** Para cada una de las siguientes especificaciones, dar un algoritmo y demostrar su corrección.

a) Encabezado: $dos : x \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{true\}$
Poscondición: $\{RV = 2\}$

b) Encabezado: $swap : x \in \mathbb{Z} \times y \in \mathbb{Z} \to \emptyset$.
Precondición: $\{x = x_0 \wedge y = y_0\}$
Poscondición: $\{x = y_0 \wedge y = x_0\}$
(En este caso dar dos algoritmos: uno usando una variable auxiliar, y otro sin variables auxiliares.)

c) Encabezado: $bisiesto : x \in \mathbb{Z} \to \mathbb{B}$
Precondición: $\{x = x_0\}$
Poscondición: $\big\{RV = \big((4|x_0 \wedge \neg 100|x_0) \vee 400|x_0\big)\big\}$ donde $a|b \equiv (\exists k)(b = a \times k)$

d) Encabezado: $f : x \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0\}$
Poscondición: $\big\{RV = 1 \Leftrightarrow (\exists k)(x_0 = 2 \times k)\big\}$

**Ejercicio 3.** Demostrar que cada uno de los siguientes algoritmos es correcto respecto de su especificación.

a) Encabezado: $sumaUnoN : x \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0 \wedge x_0 > 0\}$
Poscondición: $\{RV = \Sigma_{1 \leq j \leq x_0}\ j\}$
Variable auxiliar: $i \in \mathbb{Z}$
Algoritmo:

```
RV ← 0
i ← 1
while (i ≤ x) {
    RV ← RV + i
    i ← i + 1
}
```

Invariante sugerido: $\big\{1 \leq i \leq x_0 + 1 \wedge RV = \Sigma_{1 \leq j < i}\ j \wedge x = x_0\big\}$

b) Encabezado: $sumaTodos : A \in \mathbb{Z}[] \to \mathbb{Z}$
Precondición: $\{A = A_0\}$
Poscondición: $\big\{RV = \Sigma_{0 \leq i < |A_0|} A_0[i]\big\}$
Variable auxiliar: $i \in \mathbb{Z}$
Algoritmo:

```
RV ← 0
i ← 0
while (i < |A|) {
    RV ← RV + A[i]
    i ← i + 1
}
```

Invariante sugerido: $\big\{0 \leq i \leq |A_0| \wedge RV = \Sigma_{0 \leq j < i}\ A_0[j] \wedge A = A_0\big\}$

c) Encabezado: $shiftRight : A \in \mathbb{Z}[] \to \emptyset$
Precondición: $\{A = A_0 \wedge |A| > 0\}$
Poscondición: $\big\{|A| = |A_0| \wedge A[0] = A_0[|A| - 1] \wedge (\forall i)(1 \leq i < |A| \Rightarrow A[i] = A_0[i - 1])\big\}$
Variables auxiliares: $i, previousValue, tmp \in \mathbb{Z}$
Algoritmo:

$previousValue \leftarrow A[|A| - 1]$
$i \leftarrow 0$
while ($i < |A|$) {
    $tmp \leftarrow A[i]$
    $A[i] \leftarrow previousValue$
    $previousValue \leftarrow tmp$
    $i \leftarrow i + 1$
}

d) Encabezado: $maximo : A \in \mathbb{Z}[] \to \mathbb{Z}$
Precondición: $\{A = A_0 \wedge |A| > 0\}$
Poscondición: $\big\{(\exists i)\big(0 \leq i < |A_0| \wedge RV = A_0[i] \wedge \big(\forall j)\big(0 \leq j < |A_0| \Rightarrow A_0[j] \leq A_0[i]\big)\big)\big\}$
Variable auxiliar: $i \in \mathbb{Z}$
Algoritmo:

$RV \leftarrow A[0]$
$i \leftarrow 1$
while ($i < |A|$) {
    if ($A[i] > RV$) {
        $RV \leftarrow A[i]$
    }
    $i \leftarrow i + 1$
}

**Ejercicio 4.** Para cada una de las siguientes especificaciones, dar un algoritmo y demostrar su corrección.

a) Usar sólo '+' como operación entre enteros.
Encabezado: $producto : x \in \mathbb{Z} \times y \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0 \wedge x > 0 \wedge y = y_0 \wedge y > 0\}$
Poscondición: $\{RV = x_0 \times y_0\}$

b) Encabezado: $raizCP : x \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0 \wedge x \geq 0\}$
Poscondición: $\{RV = \lfloor\sqrt{x_0}\rfloor\}$
donde $\sqrt{x}$ es la raíz cuadrada positiva de $x$, y $\lfloor x \rfloor$ es la parte entera de $x$.

c) Encabezado: $sumaATodos : A \in \mathbb{Z}[] \times n \in \mathbb{Z} \to \emptyset$.
Precondición: $\{A = A_0 \wedge n = n_0\}$
Poscondición: $\{|A| = |A_0| \wedge (\forall i)(0 \leq i < |A| \Rightarrow A[i] = A_0[i] + n_0)\}$

d) Encabezado: $dobleSuma : x \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{x = x_0\}$
Poscondición: $\{RV = \Sigma_{1 \leq i \leq x_0} \Sigma_{1 \leq j \leq i}\ j\}$

e) Encabezado: $permuta : A \in \mathbb{Z}[] \to \emptyset$
Precondición: $\{A = A_0\}$
Poscondición: $\big\{|A| = |A_0| \wedge \big(\forall i\big)\big(0 \leq i < |A| \Rightarrow$
$(\#j)(0 \leq j < |A| \wedge A_0[i] = A_0[j]) = (\#k)(0 \leq k < |A| \wedge A_0[i] = A[k])\big)\big\}$

**Ejercicio 5.** Ordenar los elementos de una lista es un problema que puede especificarse de forma sencilla, pero cuyas soluciones no son triviales. A continuación presentamos un algoritmo llamado *insertion sort* que resuelve el problema. Demostrar que es correcto respecto de su especificación.

Encabezado: $sort : A \in \mathbb{Z}[] \rightarrow \emptyset$

Precondición: $\{A = A_0 \wedge |A| > 0\}$

Poscondición: $\big\{|A| = |A_0| \wedge \big(\forall i\big)\big(0 \leq i < |A| - 1 \Rightarrow A[i] \leq A[i+1]\big) \wedge$
$\big(\forall j\big)\big(0 \leq j < |A| \Rightarrow$
$(\#k)(0 \leq k < |A| \wedge A_0[j] = A_0[k]) = (\#l)(0 \leq l < |A| \wedge A_0[j] = A[l])\big)\big\}$

Variables auxiliares: $i, j, newValue \in \mathbb{Z}$

Algoritmo:

```
i ← 1
while (i < |A|) {
    newValue ← A[i]
    j ← i
    while (j > 0 ∧ A[j − 1] > newValue) {
        A[j] ← A[j − 1]
        j ← j − 1
    }
    A[j] ← newValue
    i ← i + 1
}
```