

# Práctico 3: Introducción a la Programación Imperativa

## Algoritmos y Estructuras de Datos I

2<sup>do</sup> cuatrimestre 2016

Esta guía tiene como objetivo afianzar los conceptos elementales de la programación imperativa. Por un lado, se pretende reforzar la noción de programa como transformador de estados de manera intuitiva. Por otro lado, priorizando la interpretación de los programas como transformadores de predicados, se busca consolidar la noción de **anotación de programa**, y en particular de **precondición** y **postcondición**. Además se presenta las noción de corrección de programas anotados.

1. En cada uno de los siguientes programas, anote los valores que las variables toman a medida que se ejecutan.

a) **Var**  $x : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto 1) \rrbracket$   
 $x := 5$   
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$

b) **Var**  $x, y : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto 2, y \mapsto 5) \rrbracket$   
 $x := x + y$ ;  
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$   
 $y := y + y$   
 $\llbracket \sigma_2 : \quad \quad \quad \rrbracket$

c) **Var**  $x, y : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto 2, y \mapsto 5) \rrbracket$   
 $y := y + y$ ;  
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$   
 $x := x + y$   
 $\llbracket \sigma_2 : \quad \quad \quad \rrbracket$

d) **Var**  $x, y : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto 2, y \mapsto 5) \rrbracket$   
 $y, x := y + y, x + y$   
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$

e) **Var**  $x, y : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto 3, y \mapsto 1) \rrbracket$   
**if**  $x \geq y \rightarrow$   
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$   
 $x := 0$   
 $\llbracket \sigma_2 : \quad \quad \quad \rrbracket$   
 $\square$   $x \leq y \rightarrow$   
 $\llbracket \sigma'_1 : \quad \quad \quad \rrbracket$   
 $x := 2$   
 $\llbracket \sigma'_2 : \quad \quad \quad \rrbracket$   
**fi**  
 $\llbracket \sigma_3 : \quad \quad \quad \rrbracket$

f) **Var**  $x, y : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto -100, y \mapsto 1) \rrbracket$   
**if**  $x \geq y \rightarrow$   
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$   
 $x := 0$   
 $\llbracket \sigma_2 : \quad \quad \quad \rrbracket$   
 $\square$   $x \leq y \rightarrow$   
 $\llbracket \sigma'_1 : \quad \quad \quad \rrbracket$   
 $x := 2$   
 $\llbracket \sigma'_2 : \quad \quad \quad \rrbracket$   
**fi**  
 $\llbracket \sigma'_3 : \quad \quad \quad \rrbracket$

g) **Var**  $x, y : Int$ ;  
 $\llbracket \sigma_0 : (x \mapsto 1, y \mapsto 1) \rrbracket$   
**if**  $x \geq y \rightarrow$   
 $\llbracket \sigma_1 : \quad \quad \quad \rrbracket$   
 $x := 0$   
 $\llbracket \sigma_2 : \quad \quad \quad \rrbracket$   
 $\square$   $x \leq y \rightarrow$   
 $\llbracket \sigma'_1 : \quad \quad \quad \rrbracket$   
 $x := 2$   
 $\llbracket \sigma'_2 : \quad \quad \quad \rrbracket$   
**fi**  
 $\llbracket \sigma_3 : \quad \quad \quad , \sigma'_3 : \quad \quad \quad \rrbracket$

h) **Var**  $i : Int$ ;  
 $\llbracket \sigma_0 : (i \mapsto 4) \rrbracket$   
**do**  $i \neq 0 \rightarrow$   
 $\llbracket \sigma_1^1 : \quad , \sigma_1^2 : \quad , \dots \rrbracket$   
 $i := i - 1$   
 $\llbracket \sigma_2^1 : \quad , \sigma_2^2 : \quad , \dots \rrbracket$   
**od**  
 $\llbracket \sigma_3 : \quad \quad \quad \rrbracket$

i) **Var**  $i : Int$ ;  
 $\llbracket \sigma_0 : (i \mapsto 400) \rrbracket$   
**do**  $i \neq 0 \rightarrow$   
 $\llbracket \sigma_1^1 : \quad , \sigma_1^2 : \quad , \dots \rrbracket$   
 $i := 0$   
 $\llbracket \sigma_2^1 : \quad , \sigma_2^2 : \quad , \dots \rrbracket$   
**od**  
 $\llbracket \sigma_3 : \quad \quad \quad \rrbracket$

```

j) Var i : Int;
   [[σ0 : (i ↦ 4)]]
   do i < 0 →
     [[σ11 : , σ12 : , ...]]
     i := i - 1
     [[σ21 : , σ22 : , ...]]
   od
   [[σ3 : ]]

k) Var i : Int;
   [[σ0 : (i ↦ 0)]]
   do i ≤ 0 →
     [[σ11 : , σ12 : , ...]] ★
     i := i - 1
     [[σ21 : , σ22 : , ...]] ★'
   od
   [[σ3 : ]]

l) Var r : Int;
   [[σ0 : (r ↦ 3)]]
   do r ≠ 0 →
     [ ]
     if r < 0 →
       [ ]
       r := r + 1
       [ ]
     [] r > 0 →
       [ ]
       r := r - 1
       [ ]
     fi
   od
   [ ]

```

2. Responda las siguientes preguntas respecto al ejercicio anterior:

- a) ¿Qué se puede concluir de los ítems 1b, 1c y 1d? ¿Qué tienen en común? ¿Cuáles son las diferencias en la ejecución de cada uno de ellos?
- b) ¿Se puede escribir un programa equivalente al del ítem 1c con sólo una asignación múltiple? Justifique.
- c) ¿Qué sucedería si en el ítem 1g se utilizaran las guardas “ $x > y$ ” y “ $x < y$ ”?
- d) Con respecto al programa del ítem 1k: ¿Existen predicados que caractericen exactamente todos los valores que puede tomar la variable  $i$  cuando el mismo se encuentra en ★ y en ★'? Si la respuesta es sí, escríbalos. Es más, si existen, ¿cuál es la ventaja con respecto a enumerar todos los estados? ¿La desventaja?

3. En cada uno de los siguientes programa, anotá los valores de las expresiones que se mencionan en la tabla respectiva, de acuerdo a cómo cambian los estados a medida que se ejecutan. Describí qué hace cada programa, en los términos más generales que encuentres.

```

a) Const A : array[0, 4] of Int;
   Var i, s : Int;
   [[σ0 : (i ↦ -3, s ↦ 5, A ↦ [2, 10, 10, -1])]]
   i, s := 0, 0; (★)
   [[σ0']]
   do i < 4 →
     [[σ10, ..., σ13]]
     s, i := s + A.i, i + 1
     [[σ20, ..., σ23]]
   od
   [[σ3]]

```

| Estado       | $i$ | $s$ |
|--------------|-----|-----|
| $\sigma_0$   |     |     |
| $\sigma'_0$  |     |     |
| $\sigma_1^0$ |     |     |
| $\sigma_2^0$ |     |     |
| $\sigma_1^1$ |     |     |
| $\sigma_2^1$ |     |     |
| $\sigma_1^2$ |     |     |
| $\sigma_2^2$ |     |     |
| $\sigma_1^3$ |     |     |
| $\sigma_2^3$ |     |     |
| $\sigma_3$   |     |     |

```

b) Const A : array[0, 4] of Int;
   Var i, c : Int;
   [[σ0 : (i ↦ 3, c ↦ 12, A ↦ [12, -9, 10, -1])]]
   i, c := 0, 0;
   [[σ'0]]
   do i < 4 →
     [[σ10, ..., σ13]]
     if A.i > 0 →
       c := c + 1
     □ A.i ≤ 0 →
       skip
     fi
     [[σ20, ..., σ23]]
     i := i + 1 (★)
     [[σ30, ..., σ33]]
   od
   [[σ4]]

```

| Estado                      | i | A.i | c |
|-----------------------------|---|-----|---|
| σ <sub>0</sub>              |   |     |   |
| σ' <sub>0</sub>             |   |     |   |
| σ <sub>1</sub> <sup>0</sup> |   |     |   |
| σ <sub>2</sub> <sup>0</sup> |   |     |   |
| σ <sub>3</sub> <sup>0</sup> |   |     |   |
| σ <sub>1</sub> <sup>1</sup> |   |     |   |
| σ <sub>2</sub> <sup>1</sup> |   |     |   |
| σ <sub>3</sub> <sup>1</sup> |   |     |   |
| σ <sub>1</sub> <sup>2</sup> |   |     |   |
| σ <sub>2</sub> <sup>2</sup> |   |     |   |
| σ <sub>3</sub> <sup>2</sup> |   |     |   |
| σ <sub>1</sub> <sup>3</sup> |   |     |   |
| σ <sub>2</sub> <sup>3</sup> |   |     |   |
| σ <sub>3</sub> <sup>3</sup> |   |     |   |
| σ <sub>4</sub>              |   |     |   |

4. Responda las siguientes preguntas sobre los programas del ejercicio anterior:

- En el ítem 3a ¿Qué pasa si la línea de código (★) es eliminada? Esa asignación tiene una función específica, por lo cual recibe un nombre particular ¿Cuál es ese nombre?
- ¿Cómo modificaría el programa del ítem 3a para que calcule el promedio de los valores en el arreglo A?
- En el ítem 3b ¿Qué pasa si la línea de código (★) es movida al principio del ciclo?
- ¿Cómo modificaría el programa del ítem 3b para que cuente los valores negativos que hay en el arreglo A?
- ¿Cómo modificaría el programa del ítem 3b para que solo tenga en cuenta las posiciones pares del arreglo A? ¿Y para que tenga en cuenta las posiciones impares?

5. Escriba un programa que calcule  $N!$ , a donde  $N$  es una constante de tipo  $Nat$ .

6. Escriba dos programas distintos que calculen  $\langle \sum i : 1 \leq i \leq N : i \rangle$  (a donde  $N$  es una constante de tipo  $Nat$ ). Uno debe usar un ciclo, y el otro debe basarse en la propiedad matemática

$$\langle \sum i : 1 \leq i \leq N : i \rangle = N * (N + 1) / 2.$$

Compárelos en términos de *eficiencia* y de *facilidad* para su comprensión.

7. Escriba un programa que cuente, por un lado, la cantidad de valores positivos y, por otro, la cantidad de valores negativos que ocurren en un arreglo A de tamaño N.

8. Escriba un programa que sume, por un lado, los valores positivos y, por otro, la elementos negativos de un arreglo A de tamaño N.

9. Decida cuáles de las siguientes anotaciones son correctas y cuáles incorrectas:

a) Var x : Int;  
 {x > 0}  
 x := x \* x  
 {True}

b) Var x : Int;  
 {x ≠ 100}  
 x := x \* x  
 {x ≥ 0}

c) Var x : Int;  
 {x > 0 ∧ x < 100}  
 x := x \* x  
 {x ≥ 0}

d) Var x : Int;  
 {x > 0}  
 x := x \* x  
 {x < 0}

e) Var x : Int;  
 {x < 0}  
 x := x \* x  
 {x < 0}

f) Var x : Int;  
 {True}  
 x := x \* x  
 {x ≥ 0}

10. Responda las siguientes preguntas sobre el ejercicio anterior:

- a) ¿Cuál es la utilidad de las anotaciones del programa 9a?
- b) De entre las anotaciones correctas ¿cuál tiene la precondition más fuerte? ¿cuál tiene la precondition más débil? ¿se te ocurre alguna otra precondition aún más débil?
- c) En general, ¿qué implica tener “{True}” como precondition? ¿y como postcondition?

11. Decida si las siguientes anotaciones son correctas. En caso de que no lo sean, corrijalas. (Notar que los programas son los del ejercicio 3, pero el segundo está levemente modificado.)

a) **Const**  $A : \text{array}[0, 4) \text{ of } \text{Int};$

**Var**  $i, s : \text{Int};$

$\{i = -3 \wedge s = 5\}$

$i, s := 0, 0;$

$\{i = -3 \wedge s = 5\}$

**do**  $i < 4 \rightarrow$

$\{0 \leq i < 4 \wedge s = \langle \sum j : 0 \leq j < i : A.j \rangle\}$

$s, i := s + A.i, i + 1$

$\{0 \leq i < 4 \wedge s = \langle \sum j : 0 \leq j < i : A.j \rangle\}$

**od**

$\{i = 3 \wedge s = \langle \sum j : 0 \leq j < i : A.j \rangle\}$

b) **Const**  $A : \text{array}[0, 4) \text{ of } \text{Int};$

**Var**  $i, s : \text{Int};$

$\{i = 3 \wedge c = 12\}$

$i, c := 0, 0;$

$\{i = 0 \wedge c = 0\}$

**do**  $i < 4 \rightarrow$

$\{0 \leq i < 4 \wedge c = 0\}$

**if**  $A.i > 0 \rightarrow$

$c, i := c + 1, i + 1$

$\square$   $A.i \leq 0 \rightarrow$

$i := i + 1$

**fi**

$\{0 \leq i < 4 \wedge c = \langle \text{N } j : 0 \leq j < i : A.j > 0 \rangle\}$

**od**

$\{i = 4 \wedge c = \langle \text{N } j : 0 \leq j < i : A.j > 0 \rangle\}$

12. Responda las siguientes preguntas sobre el ejercicio anterior:

- a) ¿Por qué en las anotaciones no es necesario mencionar los valores del arreglo  $A$ ?
- b) Compare las anotaciones finales de cada programa con la explicación informal que realizó en el Ejercicio 3. ¿Las mismas se relacionan?