

Práctico 3: Introducción a la Programación Imperativa

Algoritmos y Estructuras de Datos I

1^{er} cuatrimestre 2019

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
 $\llbracket \sigma_3 : \quad \quad \quad \rrbracket$

<p>j) $\text{Var } i : \text{Int};$ $\llbracket \sigma_0 : (i \mapsto 4) \rrbracket$ do $i < 0 \rightarrow$ $\llbracket \sigma_1^1 : \quad, \sigma_1^2 : \quad, \dots \rrbracket$ $i := i - 1$ $\llbracket \sigma_2^1 : \quad, \sigma_2^2 : \quad, \dots \rrbracket$ od $\llbracket \sigma_3 : \quad \rrbracket$</p>	<p>k) $\text{Var } i : \text{Int};$ $\llbracket \sigma_0 : (i \mapsto 0) \rrbracket$ do $i \leq 0 \rightarrow$ $\llbracket \sigma_1^1 : \quad, \sigma_1^2 : \quad, \dots \rrbracket \star$ $i := i - 1$ $\llbracket \sigma_2^1 : \quad, \sigma_2^2 : \quad, \dots \rrbracket \star'$ od $\llbracket \sigma_3 : \quad \rrbracket$</p>	<p>l) $\text{Var } r : \text{Int};$ $\llbracket \sigma_0 : (r \mapsto 3) \rrbracket$ do $r \neq 0 \rightarrow$ $\llbracket \quad \rrbracket$ if $r < 0 \rightarrow$ $\llbracket \quad \rrbracket$ $r := r + 1$ $\llbracket \quad \rrbracket$ $\llbracket r > 0 \rightarrow$ $\llbracket \quad \rrbracket$ $r := r - 1$ $\llbracket \quad \rrbracket$ fi $\llbracket \quad \rrbracket$ od $\llbracket \quad \rrbracket$</p>
---	--	---

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 \star y en \star' ? Si la respuesta es sí, escríbalos. Es más, si existen, ¿cuál es la ventaja con respecto a enumerar todos los estados? ¿La desventaja?

3. Anotar con predicados los programas del ejercicio anterior.

4. Decidir si los siguientes programas anotados son correctos (equivalentes a *True*) o incorrectos (equivalentes a *False*).

<p>a) $\text{Var } x : \text{Int};$ $\{x > 0\}$ $x := x * x$ $\{True\}$</p>	<p>b) $\text{Var } x : \text{Int};$ $\{x \neq 100\}$ $x := x * x$ $\{x \geq 0\}$</p>	<p>c) $\text{Var } x : \text{Int};$ $\{x > 0 \wedge x < 100\}$ $x := x * x$ $\{x \geq 0\}$</p>
<p>d) $\text{Var } x : \text{Int};$ $\{x > 0\}$ $x := x * x$ $\{x < 0\}$</p>	<p>e) $\text{Var } x : \text{Int};$ $\{x < 0\}$ $x := x * x$ $\{x < 0\}$</p>	<p>f) $\text{Var } x : \text{Int};$ $\{True\}$ $x := x * x$ $\{x \geq 0\}$</p>

5. Responda las siguientes preguntas en función del ejercicio anterior:

- a) ¿Es útil la anotación del programa (4a)? ¿Por qué?
- b) Descartando los programas incorrectos, ¿Cual programa tiene la precondition más débil? ¿Cual tiene la precondition más fuerte? Explíquelo con sus propias palabras.
- c) Con respecto a la última pregunta, ¿se puede definir alguna otra precondition tal que esta sea aún más débil?
- d) En general, ¿qué implica tener un “ $\{True\}$ ” al principio de un programa? ¿Al final? (Ver programas (4a) y (4f) como ejemplo de esto).

6. Anotar con predicados los siguientes programas utilizando el transformador de predicados wp.

- a) $\text{Var } x, y : \text{Num};$
 $\{ \quad \}$
 $x := x + y$
 $\{x = 6 \wedge y = 5\}$
- b) $\text{Var } x : \text{Num};$
 $\{ \quad \}$
 $x := 8$
 $\{x = 8\}$
- c) $\text{Var } x : \text{Num};$
 $\{ \quad \}$
 $x := 8$
 $\{x = 7\}$
- d) $\text{Var } x, y : \text{Num};$
 $\{ \quad \}$
 $x, y := y, x$
 $\{x = B \wedge y = A\}$
- e) $\text{Var } x, y, a : \text{Num};$
 $\{ \quad \}$
 $a, x := x, y;$
 $\{ \quad \}$
 $y := a$
 $\{x = B \wedge y = A\}$
- f) $\text{Var } x, y : \text{Num};$
 $\{ \quad \}$
if $x \geq y \rightarrow$
 $\{ \quad \}$
 $x := 0$
 $\{ \quad \}$
 $\square x \leq y \rightarrow$
 $\{ \quad \}$
 $x := 2$
 $\{ \quad \}$
fi
 $\{(x = 0 \vee x = 2) \wedge y = 1\}$

7. Demostrar que los siguientes programas anotados son correctos. En todos los casos las variables x, y son de tipo Int , y a, b de tipo Bool .

- a) $\{True\}$
if $x \geq 1 \rightarrow x := x + 1$
 $\square x \leq 1 \rightarrow x := x - 1$
fi
 $\{x \neq 1\}$
- b) $\{x \neq y\}$
if $x > y \rightarrow \text{skip}$
 $\square x < y \rightarrow x, y := y, x$
fi
 $\{x > y\}$
- c) $\{True\}$
 $x, y := y * y, x * x;$
if $x \geq y \rightarrow x := x - y$
 $\square x \leq y \rightarrow y := y - x$
fi
 $\{x \geq 0 \wedge y \geq 0\}$
- d) $\{True\}$
if $\neg a \vee b \rightarrow a := \neg a$
 $\square a \vee \neg b \rightarrow b := \neg b$
fi
 $\{a \vee b\}$
- e) $\{N \geq 0\}$
 $x := 0;$
do $x \neq N \rightarrow x := x + 1$
od
 $\{x = N\}$
- f) $\{True\}$
 $r := N;$
do $r \neq 0 \rightarrow$
if $r < 0 \rightarrow r := r + 1$
 $\square r > 0 \rightarrow r := r - 1$
fi
od
 $\{r = 0\}$

8. Calcular las expresiones \mathbf{E} y \mathbf{F} de modo que los siguientes programas sean correctos:

- a) $\text{Var } x, y : \text{Nat};$
 $\{True\}$
 $x, y := x + 1, \mathbf{E}$
 $\{y = x + 1\}$
- b) $\text{Var } a, q, c, w : \text{Num};$
 $\{q = a * c \wedge w = c^2\}$
 $a, q := a + c, \mathbf{E}$
 $\{q = a * c\}$
- c) $\text{Const } A, B : \text{Nat};$
 $\text{Var } q, r : \text{Nat};$
 $\{A = q * B + r\}$
 $q := \mathbf{E}; r := r - B$
 $\{A = q * B + r\}$
- d) $\text{Const } N : \text{Num};$
 $\text{Var } x, y, p, q : \text{Num};$
 $\{x * y + p * q = N\}$
 $x := x - p;$
 $q := \mathbf{F}$
 $\{x * y + p * q = N\}$

9. Especifique los siguientes problemas con ternas de Hoare y luego derive un programa que los verifique.

- a) Calcular el mínimo de dos valores.
b) Calcular el valor absoluto de un número.

10. Utilizando el transformador de predicados w_p demostrar la siguiente equivalencia:


```

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; (★)
   [[σ'0]]
   do i < 4 →
     [[σ10, ..., σ13]]
     s, i := s + A.i, i + 1
     [[σ20, ..., σ23]]
   od
   [[σ3]]

```

Estado	Var. <i>i</i>	Var. <i>s</i>
σ ₀		
σ' ₀		
σ ₁ ⁰		
σ ₂ ⁰		
σ ₁ ¹		
σ ₂ ¹		
σ ₁ ²		
σ ₂ ²		
σ ₁ ³		
σ ₂ ³		
σ 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	Var. <i>i</i>	Val.(or) <i>A.i</i>	Var. <i>c</i>
σ ₀			
σ' ₀			
σ ₁ ⁰			
σ ₂ ⁰			
σ ₃ ⁰			
σ ₁ ¹			
σ ₂ ¹			
σ ₃ ¹			
σ ₁ ²			
σ ₂ ²			
σ ₃ ²			
σ ₁ ³			
σ ₂ ³			
σ ₃ ³			
σ ₄			

13. Responda las siguientes preguntas:

- En el ejercicio (12a) ¿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 ejercicio (12a) para que calcule el promedio de los valores en el array *A*?
- En el ejercicio (12a) ¿Qué pasa con el valor *A.i* en el estado σ₀? ¿Por que esto no es un problema?
- En el ejercicio (12b) ¿Qué pasa si la línea de código (★) es movida al principio del loop/ciclo/do?
- ¿Cómo modificaría el programa del ejercicio (12b) para que cuente los valores negativos que hay en el array *A*?
- ¿Cómo modificaría el programa del ejercicio (12b) para que solo tenga en cuenta las posiciones pares del array *A*? ¿Y para que tenga en cuenta las posiciones impares?

14. Decida si las siguientes anotaciones son correctas. En caso de que no lo sean, corríjalas. (Notar que el primer programa es exactamente el primer programa que aparece en el ejercicio 12. En cambio el segundo está levemente modificado con respecto al que aparece en el mismo ejercicio.)

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

15. Responda las siguientes preguntas en función del ejercicio anterior:

- a) ¿Por qué en las anotaciones no es necesario mencionar los valores del array A ?
- b) Compare las anotaciones finales de cada programa con la explicación informal que realizó en el Ejercicio 12. ¿Las mismas se relacionan?

16. Explicar en lenguaje natural que piden las especificaciones de los siguientes programas y analizar que hace realmente cada uno. En base a estas observaciones determinar si son correctos.

- a) **Const** $N : \text{Num};$
Var $s, i : \text{Num};$
 $a : \text{array}[0, N) \text{ of } \text{Num};$
 $\{N \geq 0\}$
 $i, s := 0, 0$
do $i \neq N \rightarrow$
 $s := s + a.i$
od
 $\{s = \langle \sum k : 0 \leq k < N : a.k \rangle\}$
donde el arreglo a no cambia.
- b) **Const** $N : \text{Num};$
Var $s, i : \text{Num};$
 $a : \text{array}[0, N) \text{ of } \text{Num};$
 $\{N \geq 0\}$
 $i, s := 0, 0$
do $i \neq N \rightarrow$
 $i := i + 1$
 $s := s + a.i$
od
 $\{s = \langle \sum k : 0 \leq k < N : a.k \rangle\}$
donde el arreglo a no cambia.
- c) **Const** $N : \text{Num};$
Var $s, i : \text{Num};$
 $a : \text{array}[0, N) \text{ of } \text{Num};$
 $\{N \geq 0\}$
 $i, s := -1, 0$
do $i \neq N \rightarrow$
 $i := i + 1$
 $s := s + a.i$
od
 $\{s = \langle \sum k : 0 \leq k < N : a.k \rangle\}$
donde el arreglo a no cambia.
- d) **Const** $N : \text{Num};$
Var $i : \text{Num}; r : \text{Bool};$
 $a : \text{array}[0, N) \text{ of } \text{Num};$
 $\{N \geq 0\}$
 $i, r := 0, \text{False}$
do $i \neq N \wedge \neg r \rightarrow$
if $a.i = e \rightarrow r := \text{True}$
 $\square a.i \neq e \rightarrow \text{skip}$
fi
 $i := i + 1$
od
 $\{\langle \exists k : 0 \leq k < N : a.k = e \rangle \Rightarrow a.i = e\}$
donde el arreglo a no cambia.

17. Decidir y justificar si los siguientes predicados son invariantes para el programa 16(b):

- a) $\{1 \leq i\}$
- b) $\{0 \leq i\}$
- c) $\{0 \leq i \leq N\}$
- d) $\{s = \langle \sum k : 0 \leq k < N : a.k \rangle\}$
- e) $\{0 \leq s \leq \langle \sum k : 0 \leq k < N : a.k \rangle\}$

Ejercicios extra

18. Derivar los siguientes programas con bucles utilizando los invariantes dados:

- a) Desarrollar un algoritmo para calcular el máximo común divisor entre dos enteros positivos, especificado como:

Const $X, Y : Int$;

Var $x, y : Int$;

$\{X > 0 \wedge Y > 0 \wedge x = X \wedge y = Y\}$

S

$\{x = mcd.X.Y\}$

Utilizando como invariante $\{I : x > 0 \wedge y > 0 \wedge mcd.x.y = mcd.X.Y\}$

Para derivar S serán de utilidad las propiedades del mcd:

(1) $mcd.x.x = x$

(2) $mcd.x.y = mcd.y.x$

(3) $x > y \Rightarrow mcd.x.y = mcd.(x - y).y$

$y > x \Rightarrow mcd.x.y = mcd.x.(y - x)$

- b) Derivar dos programas que calculen $r = X^Y$ a partir de cada una de las siguientes definiciones funcionales de la función exponencial especificada como $exp.x.y = x^y$:

(i) $exp.x.y = (y = 0 \rightarrow 1$
 $\square y \neq 0 \rightarrow x * exp.x.(y - 1)$
 $)$

(ii) $exp.x.y = (y = 0 \rightarrow 1$
 $\square y \neq 0 \rightarrow (y \bmod 2 = 0 \rightarrow exp.(x * x).(y \div 2)$
 $\square y \bmod 2 = 1 \rightarrow x * exp.x.(y - 1)$
 $)$

Diseñar los dos programas a partir de:

Precondición R: $\{x = X \wedge y = Y \wedge x \geq 0 \wedge y \geq 0\}$

Postcondición Q: $\{r = X^Y\}$

Invariante I: $\{y \geq 0 \wedge r * x^y = X^Y\}$

Para cada programa usar una de las definiciones. Tener en cuenta las mismas a la hora de decidir la manera de achicar la cota.

19. Demostrar que si el programa

$$\begin{array}{l} \{P\} \\ \mathbf{if} B \rightarrow S \\ \mathbf{fi} \\ \{Q\} \end{array} \quad \text{es correcto, entonces también lo es} \quad \begin{array}{l} \{P\} \\ S \\ \{Q\} \end{array}$$

20. Calcular una precondición P de modo que sean correctos los siguientes programas anotados. Agregar además las anotaciones intermedias en caso que haya sentencias compuestas con “;”. Suponer que las variables x, y, z, q, r son de tipo *Int*, las variables i, j de tipo *Nat* y las variables a, b de tipo *Bool*:

a) $\{P\} x := 8 \{x = 8\}$

b) $\{P\} x := 8 \{x \neq 8\}$

c) $\{P\} x := 9 \{x = 7\}$

d) $\{P\} x := x + 1; y := y - 2 \{x + y = 0\}$

e) $\{P\} x := x + 1; y := y - 1 \{x * y = 0\}$

f) $\{P\} x := x + 1; y := y - 1 \{x + y + 10 = 0\}$

g) $\{P\} z := z * y; x := x - 1 \{z * y^x = C\}$

h) $\{P\} x, y, z := 1, d, c \{x * x^y = c^d\}$

i) $\{P\} i, j := i + i, j; j := j + i \{i = j\}$

- j) $\{P\} x := (x - y) * (x + y) \{x + y^2 = 0\}$
k) $\{P\} q, r := q + 1, r - y \{q * y + r = x\}$
l) $\{P\} a := a \equiv b; b := a \equiv b; a := a \equiv b \{(a \equiv B) \wedge (b \equiv A)\}$

21. a) Usando las propiedades del transformador de predicados *weakest precondition* que se encuentran en los puntos 1 y 9 del “Digesto para la programación imperativa”, demostrar las siguientes propiedades:

- 1) $\{P\} S \{Q\} \wedge [P_0 \Rightarrow P] \Rightarrow \{P_0\} S \{Q\}$
2) $\{P\} S \{Q\} \wedge [Q \Rightarrow Q_0] \Rightarrow \{P\} S \{Q_0\}$
3) $\{P\} S \{Q\} \wedge \{P\} S \{R\} \equiv \{P\} S \{Q \wedge R\}$
4) $\{P\} S \{Q\} \wedge \{R\} S \{Q\} \equiv \{P \vee R\} S \{Q\}$

b) Desde un punto de vista práctico, ¿qué aportan las propiedades anteriores a la hora de verificar la corrección de un programa?

22. Dado $n > 0$, desarrollar un programa que devuelva en la variable k la mayor potencia de 2 menor o igual que n , utilizando el invariante dado.

Precondición R : $\{n > 0\}$

Postcondición Q : $\{0 < k \leq n \wedge n < 2 * k \wedge \langle \exists j : 0 \leq j : k = 2^j \rangle\}$

Invariante I : $\{0 < k \leq n \wedge \langle \exists j : 0 \leq j : k = 2^j \rangle\}$

23. Sean S, S_0, S_1, T programas cualesquiera, B_0, B_1 guardas cualesquiera, E, F expresiones cualesquiera. En cada caso, ¿son equivalentes los programas i, ii e iii ? En caso afirmativo demostralo, en caso negativo dá un contraejemplo (instanciando los programas y las guardas).

- | | | | |
|-------|---|------|---|
| a) i) | $x := E;$
$y := F;$ | ii) | $y := F;$
$x := E;$ |
| b) i) | if $B_0 \rightarrow S$
$\square B_1 \rightarrow S$
fi | ii) | S |
| c) i) | if $B_0 \rightarrow S; S_0; T$
$\square B_1 \rightarrow S; S_1; T$
fi | ii) | if $B_0 \rightarrow S; S_0$
$\square B_1 \rightarrow S; S_1$
fi;
T |
| | | iii) | $S;$
if $B_0 \rightarrow S_0; T$
$\square B_1 \rightarrow S_1; T$
fi |

24. Considerar los siguientes programas anotados, donde la variable x es de tipo *Int*:

- | | | | |
|----|--|-----|--|
| i) | $\{P\}$
do $x \neq 0 \rightarrow x := x - 1$
od
$\{x = 0\}$ | ii) | $\{P\}$
do $x \neq 0 \rightarrow x := x - 2$
od
$\{x = 0\}$ |
|----|--|-----|--|

a) Determinar en cada caso una precondición P , la más débil que encuentre, de manera que se satisfaga la corrección de las anotaciones.

b) El predicado P encontrado, ¿es la precondición más débil?

25. Considerar los siguientes programas que intercambian los valores de dos variables x e y de tipo *Int*:

- | | | |
|----------------|------------------------------------|--|
| $x, y := y, x$ | $z := x;$
$x := y;$
$y := z$ | $x := x - y;$
$y := x + y;$
$x := y - x$ |
|----------------|------------------------------------|--|

a) Especificar la pre y postcondición, y *verificá* los tres programas.

b) Decimos que dos programas S y T son *equivalentes*, denotado por $S \simeq T$, si y solo si $wp.S.Q \equiv wp.T.Q$ para cualquier predicado Q . Demostrar que el primer programa es equivalente al tercero, valiéndote de las siguientes propiedades de sustitución sintáctica en predicados:

- $(Q(x := E))(x := F) \equiv Q(x := E(x := F))$
donde x es una (lista de) variable(s), y E y F son (listas de) expresiones bien definidas.

- $Q(x := E) \equiv Q(x, y := E, y)$
donde x e y son variables distintas.

c) ¿Es el segundo programa equivalente a los demás? En caso negativo, ¿cómo podemos relajar la definición de equivalencia, de modo que sea satisfecha por este programa respecto a cualquiera de los otros dos?

26. a) Demostrar las siguientes equivalencias entre programas:

- 1) $x := x \simeq \mathbf{skip}$
- 2) $S; \mathbf{skip} \simeq S$ y simétricamente $\mathbf{skip}; S \simeq S$
- 3) $S; \mathbf{abort} \simeq \mathbf{abort}$ y simétricamente $\mathbf{abort}; S \simeq \mathbf{abort}$
- 4) $(S; T); U \simeq S; (T; U)$

b) Pensando a $;$ como un operador binario, y haciendo analogía con las propiedades del cálculo proposicional, ¿qué nombres podríamos darle a las propiedades (2), (3) y (4)?