Práctico 3: Introducción a la Programación Imperativa

Algoritmos y Estructuras de Datos I
1er cuatrimestre 2023

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 $x : Int$;
$[\![\sigma_0 : (x \mapsto 1)]\!]$
$x := 5$
$[\![\sigma_1 : \quad\quad]\!]$

b) Var $x, y : Int$;
$[\![\sigma_0 : (x \mapsto 2, y \mapsto 5)]\!]$
$x := x + y;$
$[\![\sigma_1 : \quad\quad]\!]$
$y := y + y$
$[\![\sigma_2 : \quad\quad]\!]$

c) Var $x, y : Int$;
$[\![\sigma_0 : (x \mapsto 2, y \mapsto 5)]\!]$
$y := y + y;$
$[\![\sigma_1 : \quad\quad]\!]$
$x := x + y$
$[\![\sigma_2 : \quad\quad]\!]$

d) Var $x, y : Int$;
$[\![\sigma_0 : (x \mapsto 2, y \mapsto 5)]\!]$
$y, x := y + y, x + y$
$[\![\sigma_1 : \quad\quad]\!]$

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

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

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

h) Var $i : Int$;
$[\![\sigma_0 : (i \mapsto 4)]\!]$
do $i \neq 0 \rightarrow$
  $[\![\sigma_1^1 : \quad, \sigma_1^2 : \quad, \cdots]\!]$
  $i := i - 1$
  $[\![\sigma_2^1 : \quad, \sigma_2^2 : \quad, \cdots]\!]$
od
$[\![\sigma_3 : \quad\quad]\!]$

i) Var $i : Int$;
$[\![\sigma_0 : (i \mapsto 400)]\!]$
do $i \neq 0 \rightarrow$
  $[\![\sigma_1^1 : \quad, \sigma_1^2 : \quad, \cdots]\!]$
  $i := 0$
  $[\![\sigma_2^1 : \quad, \sigma_2^2 : \quad, \cdots]\!]$
od
$[\![\sigma_3 : \quad\quad]\!]$

j) Var $i : Int$;
$[\![\sigma_0 : (i \mapsto 4)]\!]$
do $i < 0 \to$
$[\![\sigma_1^1 : \quad , \sigma_1^2 : \quad , \cdots]\!]$
$i := i - 1$
$[\![\sigma_2^1 : \quad , \sigma_2^2 : \quad , \cdots]\!]$
od
$[\![\sigma_3 : \quad]\!]$

k) Var $i : Int$;
$[\![\sigma_0 : (i \mapsto 0)]\!]$
do $i \leq 0 \to$
$[\![\sigma_1^1 : \quad , \sigma_1^2 : \quad , \cdots]\!]$ ★
$i := i - 1$
$[\![\sigma_2^1 : \quad , \sigma_2^2 : \quad , \cdots]\!]$ ★′
od
$[\![\sigma_3 : \quad]\!]$

l) Var $r : Int$;
$[\![\sigma_0 : (r \mapsto 3)]\!]$
do $r \neq 0 \to$
$[\![\quad]\!]$
if $r < 0 \to$
$[\![\quad]\!]$
$r := r + 1$
$[\![\quad]\!]$
$\square$ $r > 0 \to$
$[\![\quad]\!]$
$r := r - 1$
$[\![\quad]\!]$
fi
$[\![\quad]\!]$
od
$[\![\quad]\!]$

2. Responda las siguientes preguntas respecto al ejercicio anterior:

   a) ¿Qué se puede concluir de los ítems 1*b*, 1*c* y 1*d*? ¿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 1*c* con sólo una asignación múltiple? Justifique.

   c) ¿Qué sucedería si en el ítem 1*g* se utilizaran las guardas "$x > y$" y "$x < y$"?

   d) Con respecto al programa del ítem 1*k*: ¿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. Anotar con predicados los programas del ejercicio anterior.

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

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

   b) Var $x : Int$;
   $\{x \neq 100\}$
   $x := x * x$
   $\{x \geq 0\}$

   c) Var $x : Int$;
   $\{x > 0 \wedge x < 100\}$
   $x := x * x$
   $\{x \geq 0\}$

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

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

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

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

   a) ¿Es útil la anotación del programa (4*a*)?¿Por qué?

   b) Descartando los programas incorrectos, ¿Cual programa tiene la precondición más débil? ¿Cual tiene la precondición más fuerte? Explíquelo con sus propias palabras.

   c) Con respecto a la última pregunta, ¿se puede definir alguna otra precondición tal que esta sea aún más débil?

   d) En general, ¿qué implica tener un "$\{True\}$" al principio de un programa? ¿Al final? (Ver programas (4*a*) y (4*f*) como ejemplo de esto).

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

a) Var $x, y : Num$;
$\{\quad\}$
$x := x + y$
$\{x = 6 \wedge y = 5\}$

b) Var $x : Num$;
$\{\quad\}$
$x := 8$
$\{x = 8\}$

c) Var $x : Num$;
$\{\quad\}$
$x := 8$
$\{x = 7\}$

d) Var $x, y : Num$;
$\{\quad\}$
$x, y := y, x$
$\{x = B \wedge y = A\}$

e) Var $x, y, a : Num$;
$\{\quad\}$
$a, x := x, y;$
$\{\quad\}$
$y := a$
$\{x = B \wedge y = A\}$

f) Var $x, y : Num$;
$\{\quad\}$
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 $x > y \rightarrow$ **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) $\{y = Y \wedge N \geq 0\}$
$x, n := 1, 0;$
do $n \neq N \rightarrow$
$x, n := x * y, n + 1$
od
$\{x = Y^N\}$

8. Calcular las expresiones **E** y **F** de modo que los siguientes programas sean correctos:

a) Var $x, y : Nat$;
$\{True\}$
$x, y := x + 1, \mathbf{E}$
$\{y = x + 1\}$

b) Var $a, q, c, w : Num$;
$\{q = a * c \wedge w = c^2\}$
$a, q := a + c, \mathbf{E}$
$\{q = a * c\}$

c) Const $A, B : Nat$;
Var $q, r : Nat$;
$\{A = q * B + r\}$
$q := \mathbf{E};\ r := r - B$
$\{A = q * B + r\}$

d) Const $N : Num$;
Var $x, y, p, q : Num$;
$\{x * y + p * q = N\}$
$x := x - p;$
$q := \mathbf{F}$
$\{x * y + p * q = N\}$

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. Para cada programa, anote los valores que las variables van tomando a medida que éste se ejecuta (es decir, describa los estados) utilizando la tabla que se le brinda. Explique qué hace cada programa.

a) Const $A : array[0, 4)\ of Int$;
Var $i, s : Int$;
$[\![\sigma_0 : (i \mapsto -3, s \mapsto 5, A \mapsto [2, 10, 10, -1])]\!]$
$i, s := 0, 0;$ (★)
$[\![\sigma'_0]\!]$
do $i < 4 \rightarrow$
$[\![\sigma^0_1, \cdots, \sigma^3_1]\!]$
$s, i := s + A.i, i + 1$
$[\![\sigma^0_2, \cdots, \sigma^3_2]\!]$
od
$[\![\sigma_3]\!]$

Estado	i	s
σ_0		
σ'_0		
σ^0_1		
σ^0_2		
σ^1_1		
σ^1_2		
σ^2_1		
σ^2_2		
σ^3_1		
σ^3_2		
σ_3		

b) Const $A : array[0, 4)\ of Int$;
Var $i, c : Int$;
$[\![\sigma_0 : (i \mapsto 3, c \mapsto 12, A \mapsto [12, -9, 10, -1])]\!]$
$i, c := 0, 0;$
$[\![\sigma'_0]\!]$
do $i < 4 \rightarrow$
$[\![\sigma^0_1, \cdots, \sigma^3_1]\!]$
if $A.i > 0 \rightarrow$
$c := c + 1$
▯ $A.i \leq 0 \rightarrow$
skip
fi;
$[\![\sigma^0_2, \cdots, \sigma^3_2]\!]$
$i := i + 1$ (★)
$[\![\sigma^0_3, \cdots, \sigma^3_3]\!]$
od
$[\![\sigma_4]\!]$

Estado	i	$A.i$	c
σ_0			
σ'_0			
σ^0_1			
σ^0_2			
σ^0_3			
σ^1_1			
σ^1_2			
σ^1_3			
σ^2_1			
σ^2_2			
σ^2_3			
σ^3_1			
σ^3_2			
σ^3_3			
σ_4			

11. Responda las siguientes preguntas:

a) En el ejercicio (10*a*) ¿Qué pasa si la linea de código (★) es eliminada? Esa asignación tiene una función específica, por lo cual recibe un nombre particular ¿Cuál es ese nombre?

b) ¿Cómo modificaria el programa del ejercicio (10*a*) para que calcule el promedio de los valores en el array A?

c) En el ejercicio (10*a*) ¿Qué pasa con el valor $A.i$ en el estado σ_0?¿Por que esto no es un problema?

d) En el ejercicio (10*b*) ¿Qué pasa si la linea de código (★) es movida al principio del loop/ciclo/do?

e) ¿Cómo modificaria el programa del ejercicio (10*b*) para que cuente los valores negativos que hay en el array A?

f) ¿Cómo modificaría el programa del ejercicio (10*b*) para que solo tenga en cuenta las posiciones pares del array A? ¿Y para que tenga en cuenta las posiciones impares?

12. Decida si las siguientes anotaciones son correctas. En caso de que no lo sean, corríjalas.

Ayuda: Notar que el primer programa es exactamente el primer progama que aparece en el ejercicio 10. En cambio el segundo está levemente modificado con respecto al que aparece en el mismo ejercicio.

a) Const $A : array[0,4)\ of Int$;
Var $i, s : Int$;
$\{i = -3 \wedge s = 5\}$
$i, s := 0, 0;$
$\{i = -3 \wedge s = 5\}$
do $i < 4 \rightarrow$
$\quad \{0 \leq i < 4 \wedge s = \langle \sum j \ : \ 0 \leq j < i \ : \ A.j \rangle\}$
$\quad s, i := s + A.i, i + 1$
$\quad \{0 \leq i < 4 \wedge s = \langle \sum j \ : \ 0 \leq j < i \ : \ A.j \rangle\}$
od
$\{i = 3 \wedge s = \langle \sum j \ : \ 0 \leq j < i \ : \ A.j \rangle\}$

b) Const $A : array[0,4)\ of Int$;
Var $i, s : Int$;
$\{i = 3 \wedge c = 12\}$
$i, c := 0, 0;$
$\{i = 0 \wedge c = 0\}$
do $i < 4 \rightarrow$
$\quad \{0 \leq i < 4 \wedge c = 0\}$
$\quad$ **if** $A.i > 0 \rightarrow$
$\quad\quad c, i := c + 1, i + 1$
$\quad$ ▯ $A.i \leq 0 \rightarrow$
$\quad\quad i := i + 1$
$\quad$ **fi**
$\quad \{0 \leq i < 4 \wedge c = \langle \mathrm{N}\ j \ : \ 0 \leq j < i \ : \ A.j > 0 \rangle\}$
od
$\{i = 4 \wedge c = \langle \mathrm{N}\ j \ : \ 0 \leq j < i \ : \ A.j > 0 \rangle\}$

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

a) ¿Por qué en las anotaciones no es necesarion mencionar los valores del array A?

b) Compare las anotaciones finales de cada programa con la explicación informal que realizó en el Ejercicio 10. ¿Las mismas se relacionan?

14. 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 $N : Int$;
Var $s, i : Int$;
$\quad a : array[0, N)\ of Int$;
$\{N \geq 0\}$
$i, s := 0, 0;$
do $i \neq N \rightarrow$
$\quad s := s + a.i$
od
$\{s = \langle \sum k \ : 0 \leq k < N : \ a.k \rangle\}$
donde el arreglo a no cambia.

b) Const $N : Int$;
Var $s, i : Int$;
$\quad a : array[0, N)\ of Int$;
$\{N \geq 0\}$
$i, s := 0, 0;$
do $i \neq N \rightarrow$
$\quad i := i + 1;$
$\quad s := s + a.i$
od
$\{s = \langle \sum k \ : 0 \leq k < N : \ a.k \rangle\}$
donde el arreglo a no cambia.

c) Const $N : Int$;
Var $s, i : Int$;
$\quad a : array[0, N)\ of Int$;
$\{N \geq 0\}$
$i, s := -1, 0;$
do $i \neq N \rightarrow$
$\quad i := i + 1;$
$\quad s := s + a.i$
od
$\{s = \langle \sum k \ : 0 \leq k < N : \ a.k \rangle\}$
donde el arreglo a no cambia.

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

15. Dado el siguiente programa

Const $N : Int$;
Var $s, i : Int$;
$a : array[0, N)\ of Int$;
$\{N \geq 0\}$
$i, s := 0, 0$;
do $i \neq N \rightarrow$
$s := s + a.i$
$i := i + 1$;
od
$\{s = \langle \sum k\ : 0 \leq k < N :\ a.k \rangle\}$
(donde el arreglo a no cambia)

decir si los siguientes predicados cumplen las tres primeras propiedades de un invariante (que se cumple antes, que se cumple durante y que se cumple al terminar el ciclo):

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

16. Utilizando el transformador de predicados *wp* demostrar la siguiente equivalencia:

$\{P\}$		$P \Rightarrow P \wedge (B_0 \vee B_1)$
if $B_0 \rightarrow S_0$		$\wedge$
$\square\ B_1 \rightarrow S_1$	$\equiv$	$\{P \wedge B_0\}$
fi		S_0
$\{Q\}$		$\{Q\}$
		$\wedge$
		$\{P \wedge B_1\}$
		S_1
		$\{Q\}$

Nota: Observar que esto permite verificar un **if** anotando el programa como sigue[1]:

$\{P\}$
$\{P \wedge (B_0 \vee B_1)\}$
if $B_0 \rightarrow$
$\{P \wedge B_0\}$
S_0
$\{Q\}$
$\square\ B_1 \rightarrow$
$\{P \wedge B_1\}$
S_1
$\{Q\}$
fi
$\{Q\}$

17. Demostrar que si el programa

$\{P\}$		$\{P\}$
if $B_0 \rightarrow S_0$	es correcto, entonces también lo es	**if** $B_0 \rightarrow S_0$
$\square\ B_1 \rightarrow S_1$		$\square\ \neg B_0 \rightarrow S_1$
fi		**fi**
$\{Q\}$		$\{Q\}$

¿Qué utilidad tiene esta propiedad cuando se programa en lenguaje C?

[1] Esto figura en el digesto.

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) $exp.x.y \;=\; (\; y = 0 \to 1$
$\Box\; y \neq 0 \to x * exp.x.(y - 1)$
$)$

(ii) $exp.x.y \;=\; (\; y = 0 \to 1$
$\Box\; y \neq 0 \to (\; y \bmod 2 = 0 \to exp.(x * x).(y \div 2)$
$\Box\; y \bmod 2 = 1 \to x * exp.x.(y - 1)$
$)$
$)$

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

$$\begin{array}{l}\{P\}\\ \textbf{if } B \to S\\ \textbf{fi}\\ \{Q\}\end{array} \qquad \text{es correcto, entonces también lo es} \qquad \begin{array}{c}\{P\}\\ \text{S}\\ \{Q\}\end{array}$$

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 $B_0 \to S$
    $\square\ B_1 \to S$
    fi

    ii) S

    c) *i*)
    if $B_0 \to S; S_0; T$
    $\square\ B_1 \to S; S_1; T$
    fi

    ii)
    if $B_0 \to S; S_0$
    $\square\ B_1 \to S; S_1$
    fi;
    T

    iii)
    S;
    if $B_0 \to S_0; T$
    $\square\ B_1 \to S_1; T$
    fi

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

    i)
    $\{P\}$
    do $x \neq 0 \to x := x - 1$
    od
    $\{x = 0\}$

    ii)
    $\{P\}$
    do $x \neq 0 \to x := x - 2$
    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, ¿como 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) $x := x \simeq \textbf{skip}$
    2) S; **skip** $\simeq S$ y simétricamente **skip**; $S \simeq S$
    3) S; **abort** $\simeq$ **abort** y simétricamente **abort**; $S \simeq$ **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)?