Práctico 3: Introducción a la Programación Imperativa

Algoritmos y Estructuras de Datos I
2do cuatrimestre 2024

Introducción

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. 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.*

En relación a los problemas de ***laboratorio*** *se pretende introducir:*

- *el concepto de estado y de programas como transformadores de estado;*
- *el modelo computacional imperativo, y sus diferencias con el modelo funcional;*
- *la implementación en lenguaje "C" de programas imperativos vistos en el teórico-práctico.*

Lenguaje "C"

A lo largo del práctico 3 y 4 se utilizará el lenguaje C y algunas herramientas como `GDB: The GNU Project Debugger`*, para ayudar a la comprensión del concepto de estado y del paradigma imperativo.*

En el caso del lenguaje "C", para poder ejecutar un programa, lo vamos a tener que "compilar", y de esa manera generamos un archivo binario que podrá ser ejecutado en la computadora.

Cómo compilar en C: *Para compilar un archivo* `.c` *escribir en la terminal:*

```
$> gcc -Wall -Wextra -std=c99 miarchivo.c -o miprograma
```

Para ejecutar escribir:

```
$> ./miprograma
```

Para compilar para `gdb`*, agregar el flag* `-g` *al momento de compilar* `.c` *escribir en la terminal:*

```
$> gcc -Wall -Wextra -std=c99 -g miarchivo.c -o miprograma
```

Laboratorio 1 **Entrada/Salida** Hacé un programa en C, que solicite el ingreso de los valores de las variables `x,y,z`, e imprima el resultado de las siguiente expresiones. Completá los resultados de la tablas para los dos estados dados.

Expresión	(x↦7, y↦3, z↦5)	(x↦1, y↦10, z↦8)
`x + y + 1`		
`z * z + y * 45 - 15 * x`		
`y - 2 == (x * 3 + 1) % 5`		
`y / 2 * x`		
`y < x * z`		

¿En la última expresión, que tipo tiene el resultado en lenguaje "C"?

Laboratorio 2 **Debugging** Utilizá **GDB o printf** como ayuda y encontrá valores para las variables que forman el estado:

$$(\texttt{x} \mapsto \square,\ \texttt{y} \mapsto \square,\ \texttt{z} \mapsto \square,\ \texttt{b} \mapsto \square,\ \texttt{w} \mapsto \square\ )$$

de manera que las siguientes expresiones tengan el valor indicado:

Expresión	Valor
`x % 4 == 0`	True
`x + y == 0 && y - x == (-1) * z`	True
`not b && w`	False

Podes cambiar el programa hecho en el ejercicio anterior, agregando las nuevas expresiones booleanas.

Asignación

La asignación se usa, como su nombre lo indica, para asignar valores a variables. La sintaxis es $x := E$ donde x es una lista de variables separadas por comas y E es una lista de expresiones de la misma longitud, también separadas por comas, tal que la n-ésima expresión es del mismo tipo que la n-ésima variable de x. El efecto de la sentencia $x := E$ es reemplazar el valor de x por el de E.

Ejemplos de asignaciones son:

$x := x + 1$

$x, y := x - 2, x + y$

$y, p := y/2, True$

1. En cada uno de los siguientes programas, anotá los estados por los que pasa el programa (i.e. que valores van tomando las variables) a medida que éste se ejecuta.

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

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

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

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

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.

Laboratorio 3 Traducí al lenguaje C los programas 1*a*, 1*b* y 1*c* del práctico. Esos programas están escritos en un pseudocódigo de la materia y la traducción a C no siempre es directa.

El estado σ_0 debe solicitarse al usuario utilizando el comando `scanf()`. Luego, ejecute cada programa 3 veces con diferentes valores de las variables solicitadas y escriba los valores del estado final resultante en la siguiente tabla:

programa	usuario ingresa un σ_0	produce una salida σ
1*a* ejecución 1		
1*a* ejecución 2		
1*a* ejecución 3		
1*b* ejecución 1		
1*b* ejecución 2		
1*b* ejecución 3		
1*c* ejecución 1		
1*c* ejecución 2		
1*c* ejecución 3		

Laboratorio 4 Observar el ejercicio 1*d*. ¿Es posible implementarlo en C?

Alternativa

La sintaxis de la alternativa es

$$\begin{array}{l}\textbf{if } B_0 \to S_0\\ \square\ B_1 \to S_1\\ \quad \ldots\\ \square\ B_n \to S_n\\ \textbf{fi}\end{array}$$

donde para todo $0 \leq i \leq n$, B_i es una expresión booleana y S_i es una instrucción. Las expresiones B_i suelen llamarse guardas o protecciones (en inglés, guards) y $B_i \to S_i$ se denominan comandos resguardados o protegidos (en inglés, guarded commands).

La alternativa funciona de la siguiente manera: todas las guardas son evaluadas; si ninguna es $True$, se produce un abort; en caso contrario, se elige (de alguna manera) una guarda B_i que sea $True$ y se ejecuta el correspondiente S_i . Ya hemos dicho que **abort***no es una sentencia ejecutable; por lo tanto, un* **if***con ninguna guarda verdadera será considerado un error.*

3. En cada uno de los siguientes programas, anotá los estados por los que pasa el programa (i.e. que valores van tomando las variables) a medida que éste se ejecuta.

a) Var $x, y : Int$;
$[\![\sigma_0 : (x \mapsto 3, y \mapsto 1)]\!]$
if $x \geq y \to$
$[\![\sigma_1 :\ ]\!]$
$x := 0$
$[\![\sigma_2 :\ ]\!]$
$\square\ x \leq y \to$
$[\![\sigma_1' :\ ]\!]$
$x := 2$
$[\![\sigma_2' :\ ]\!]$
fi
$[\![\sigma_3 :\ ]\!]$

b) Var $x, y : Int$;
$[\![\sigma_0 : (x \mapsto -100, y \mapsto 1)]\!]$
if $x \geq y \to$
$[\![\sigma_1 :\ ]\!]$
$x := 0$
$[\![\sigma_2 :\ ]\!]$
$\square\ x \leq y \to$
$[\![\sigma_1' :\ ]\!]$
$x := 2$
$[\![\sigma_2' :\ ]\!]$
fi
$[\![\ \sigma_3' :\ ]\!]$

c) Var $x, y : Int$;
$[\![\sigma_0 : (x \mapsto 1, y \mapsto 1)]\!]$
if $x \geq y \to$
$[\![\sigma_1 :\ ]\!]$
$x := 0$
$[\![\sigma_2 :\ ]\!]$
$\square\ x \leq y \to$
$[\![\sigma_1' :\ ]\!]$
$x := 2$
$[\![\sigma_2' :\ ]\!]$
fi
$[\![\sigma_3 :\ ,\ \sigma_3' :\ ]\!]$

4. Responda las siguientes preguntas respecto al ejercicio anterior:

   a) ¿Qué sucedería si en el ítem 3*c* se utilizaran las guardas "$x > y$" y "$x < y$"?

Laboratorio 5 Traducí al lenguaje C los programas 3*a*, 3*b* y 3*c*.
El estado σ_0 debe solicitarse al usuario, agregando las instrucciones necesarias para que el usuario pueda ingresar los valores de las variables de entrada.

Laboratorio 6 Traducí a lenguaje C los programas que siguen a continuación, agregando las instrucciones necesarias para que el usuario pueda ingresar los valores de las variables de entrada. Luego, completá los estados:

$[\![\sigma_0 : (\mathtt{x} \mapsto \boxed{5}, \mathtt{y} \mapsto \boxed{4}, \mathtt{z} \mapsto \boxed{8}, \mathtt{m} \mapsto \boxed{0})\ ]\!]$

Var $x, y, z, m : Int$;
if $(x < y) \to m := x$
$\square\ (x \geq y) \to m := y$
fi

$[\![\sigma_0 : (\mathtt{x} \mapsto \square, \mathtt{y} \mapsto \square, \mathtt{z} \mapsto \square, \mathtt{m} \mapsto \square)\ ]\!]$

if $(m < z) \to$**skip**
$\square\ (m \geq z) \to m := z$
fi

$[\![\sigma_0 : (\mathtt{x} \mapsto \square, \mathtt{y} \mapsto \square, \mathtt{z} \mapsto \square, \mathtt{m} \mapsto \square)\ ]\!]$

Volvé a ejecutar nuevamente con otros estados iniciales. ¿Qué hace este programa? ¿Cuál es el valor final de la variable `m`?.

Repetición

Para la repetición tenemos una sentencia que permite la ejecución repetida de una acción (ciclo). La sintaxis es

$$\begin{array}{ll}\textbf{do} & B_0 \to S_0 \\ [] & B_1 \to S_1 \\ & \dots \\ [] & B_n \to S_n \\ \textbf{od} & \end{array}$$

donde para todo $0 \leq i \leq n$, B_i es una expresión booleana llamada guarda y S_i es una instrucción.

El ciclo funciona de la siguiente manera: mientras haya guardas equivalentes a $True$, se elige (de alguna manera) una de ellas y se ejecuta la instrucción correspondiente; luego vuelven a evaluarse las guardas. Esto se repite hasta que ninguna guarda sea $True$. Si desde un principio ninguna guarda es $True$, el ciclo equivale a un **skip**.

5. En cada uno de los siguientes programas, anotá los estados por los que pasa el programa (i.e. que valores van tomando las variables) a medida que éste se ejecuta.

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

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

c) 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
$[\![\sigma_3 : \quad]\!]$

d) 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
$[\![\sigma_3 : \quad]\!]$

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

6. Responda las siguientes preguntas respecto al ejercicio anterior:

   a) Con respecto al programa del ítem 5*d*: ¿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?

7. *a)* Escriba un programa que calcule $N!$, a donde $N \geq 0$ es una constante de tipo Int.

   b) Escriba otro programa que calcule $N!$ efectuando las multiplicaciones en un orden diferente.

   c) Para $N = 5$, ejecute manualmente ambos programas y escriba las tablas de estados.

Laboratorio 7 Traducí al lenguaje C los programas 5*a*, 5*b* y 5*c*. El estado σ_0 debe solicitarse al usuario, agregando las instrucciones necesarias para que el usuario pueda ingresar los valores de las variables de entrada.

Laboratorio 8 Traducí a lenguaje C los programas que siguen a continuación, agregando las instrucciones necesarias para que el usuario pueda ingresar los valores. Luego, completá los estados, donde el estado a completar es el resultado de realizar 1, 2, 3 o 4 iteraciones del ciclo. Una iteración es la ejecución completa del cuerpo del ciclo.

a) .

$[\![\sigma_0\colon\ (\mathtt{x}\mapsto\boxed{13}, \mathtt{y}\mapsto\boxed{3}, \mathtt{i}\mapsto\boxed{0})\ ]\!]$

$i := 0$
do $(x \geq y) \to$
  $x := x - y$
  $i := i + 1$
  $[|\sigma_1^0, \sigma_1^1, \sigma_1^2, \sigma_1^3|]$
od

$[|\sigma_1^0\colon (\mathtt{x}\mapsto\square, \mathtt{y}\mapsto\square, \mathtt{i}\mapsto\square)$, **luego de iter. 1**, $\sigma_1^1\colon (\mathtt{x}\mapsto\square, \mathtt{y}\mapsto\square, \mathtt{i}\mapsto\square)$, **luego de iter. 2**,
$\sigma_1^2\colon (\mathtt{x}\mapsto\square, \mathtt{y}\mapsto\square, \mathtt{i}\mapsto\square)$, **luego de iter. 3** $\sigma_1^3\colon (\mathtt{x}\mapsto\square, \mathtt{y}\mapsto\square, \mathtt{i}\mapsto\square)$ **luego de iter. 4** $|]$

b) .

$[\![\sigma_0\colon\ (\mathtt{x}\mapsto\boxed{5}, \mathtt{i}\mapsto\boxed{0}, \mathtt{res}\mapsto\boxed{F})\ ]\!]$
$i := 2$
$res := True$
do$(i < x \wedge res) \to$
  $res := res \wedge (\mathtt{mod}(x, i) \neq 0)$
  $i := i + 1$
  $[|\sigma_1^0, \sigma_1^1, \sigma_1^2|]$
od

$[|\sigma_1^0\colon (\mathtt{x}\mapsto\square, \mathtt{i}\mapsto\square, \mathtt{res}\mapsto\square)$, **luego de iter. 1**, $\sigma_1^1\colon (\mathtt{x}\mapsto\square, \mathtt{i}\mapsto\square, \mathtt{res}\mapsto\square)$, **luego de iter. 2**,
$\sigma_1^2\colon (\mathtt{x}\mapsto\square, \mathtt{i}\mapsto\square, \mathtt{res}\mapsto\square)$, **luego de iter. 3** $|]$

c) Ejecutá los programas con otros estados iniciales para deducir qué hace cada uno.

Arreglos

Un arreglo o vector (en inglés, array) es una función definida sobre un segmento de los naturales. En general, un arreglo se declara de la siguiente manera:

$$a : array[p, q) of [A]$$

donde $p \leq q$ y A indica el tipo de los elementos del arreglo. Por ejemplo:
$a : array[0, 4) of Int$ declara un arreglo de 4 elementos de tipo entero.
$a : array[5, 7) of Bool$ declara un arreglo de 2 elementos de tipo Bool.

- *La cantidad de elementos de un arreglo es $q - p$. Como permitimos $q = p$, el arreglo puede no contener elementos. En este caso, se dirá que es un arreglo vacío.*
- *No es necesario que el primer índice del arreglo sea 0.*
- *Un elemento de un arreglo a se referenciará por $a.n$, donde n sólo tiene sentido en el rango $p \leq n < q$.*
- *Diremos que un valor v está en el arreglo si algún elemento del arreglo es igual a v.*
- *Es posible usar variables cuantificadas en relación a arreglos.*
  Por ejemplo: si $a : array[p, q) of Int$, entonces $\langle \text{Max } i : p \leq i < q : a.i \rangle$ indica el máximo elemento del arreglo.

8. En cada uno de los siguientes programas, anote los valores de las expresiones que se mencionan en la tabla respectiva, de acuerdo a cómo cambian los estados a medida que se ejecutan. Describa qué hace cada programa, en los términos más generales que encuentre.

   a) Const $A : array[0, 4)\ of Int$;
   Var $i, s : Int$;
   $[\![\sigma_0 : (i \mapsto -3, s \mapsto 5, A \mapsto \boxed{2}\boxed{10}\boxed{10}\boxed{-1})]\!]$
   $i, s := 0, 0$;
   $[\![\sigma'_0]\!]$
   do $i < 4 \to$
   $[\![\sigma^0_1, \cdots, \sigma^3_1]\!]$
   $s, i := s + A.i, i + 1$
   $[\![\sigma^0_2, \cdots, \sigma^3_2]\!]$
   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 $i, c : Int$;
   $[\![\sigma_0 : (i \mapsto 3, c \mapsto 12, A \mapsto \boxed{12}\boxed{-9}\boxed{10}\boxed{-1})]\!]$
   $i, c := 0, 0$;
   $[\![\sigma'_0]\!]$
   do $i < 4 \to$
   $[\![\sigma^0_1, \cdots, \sigma^3_1]\!]$
   if $A.i > 0 \to$
   $c := c + 1$
   $\square$ $A.i \leq 0 \to$
   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			

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

   a) ¿Cómo modificaría el programa del ítem 8*a* para que calcule el promedio de los valores en el arreglo A?

   b) ¿Cómo modificaría el programa del ítem 8*b* para que solo tenga en cuenta las posiciones pares del arreglo A?

10. *a*) Escriba un programa que sume, por un lado, los valores positivos y, por otro, los valores múltiplos de 3 de un arreglo A de tamaño N.

    b) Para $N = 5, A = \boxed{12}\boxed{-9}\boxed{10}\boxed{0}\boxed{-1}$, ejecute manualmente el programa y escriba la tabla de estados.

11. *a*) Escriba un programa que, dados dos arreglos A y B de tamaño N y M respectivamente, calcule cuántas veces coinciden dos elementos entre ellos.

    b) Para $A = \boxed{8}\boxed{0}\boxed{8}$, $B = \boxed{0}\boxed{8}$, ejecute manualmente el programa y escriba la tabla de estados.

Laboratorio 9 Traducí al lenguaje C los programas 8*b* y 8*a*. El estado σ_0 debe solicitarse al usuario, agregando las instrucciones necesarias para que el usuario pueda ingresar los valores de las variables de entrada.

Ternas de Hoare y Weakest Precondition

12. 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 $x : Int$;
    $\{True\}$
    $x := x * x$
    $\{x \geq 0\}$

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

    a) ¿Es útil la anotación del programa (12*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?

14. Para cada uno de los siguientes programas, calcule la precondición más débil y las anotaciones intermedias. Para ello utilice 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
    $\{(x = 0 \vee x = 2) \wedge y = 1\}$

15. Demuestre que las siguientes ternas de Hoare son correctas. En todos los casos las variables x, y son de tipo Int, y a, b de tipo $Bool$.

a) $\{True\}$
if $x \geq 1 \to x := x + 1$
▯ $x \leq 1 \to x := x - 1$
fi
$\{x \neq 1\}$

b) $\{x \neq y\}$
if $x > y \to$ **skip**
▯ $x < y \to x, y := y, x$
fi
$\{x > y\}$

c) $\{True\}$
$x, y := y * y, x * x;$
if $x \geq y \to x := x - y$
▯ $x \leq y \to y := y - x$
fi
$\{x \geq 0 \wedge y \geq 0\}$

d) $\{True\}$
if $\neg a \vee b \to a := \neg a$
▯ $a \vee \neg b \to b := \neg b$
fi
$\{a \vee b\}$

e) $\{N \geq 0\}$
$x := 0;$
do $x \neq N \to x := x + 1$
od
$\{x = N\}$

f) $\{True\}$
$r := N;$
do $r \neq 0 \to$
if $r < 0 \to r := r + 1$
▯ $r > 0 \to r := r - 1$
fi
od
$\{r = 0\}$

16. Analice los siguientes programas anotados. En cada caso, describa en lenguaje natural la postcondición, y decida si el programa efectivamente valida las anotaciones. No es necesario hacer las demostraciones.

a) Const $N : Int,\ A : array[0, N)\ ofNum;$
Var $s : Num,\ i : Int;$
$\{N \geq 0\}$
$i, s := 0, 0$ **;**
do $i \neq N \to$
$s := s + A.i$
od
$\{s = \langle \sum k\ : 0 \leq k < N :\ A.k \rangle\}$

b) Const $N : Int,\ A : array[0, N)\ ofNum;$
Var $s : Num,\ i : Int;$
$\{N \geq 0\}$
$i, s := 0, 0$ **;**
do $i \neq N \to$
$i := i + 1$ **;**
$s := s + A.i$
od
$\{s = \langle \sum k\ : 0 \leq k < N :\ A.k \rangle\}$

c) Const $N : Int,\ A : array[0, N)\ ofNum;$
Var $s : Num,\ i : Int;$
$\{N \geq 0\}$
$i, s := -1, 0$ **;**
do $i \neq N \to$
$i := i + 1$ **;**
$s := s + A.i$
od
$\{s = \langle \sum k\ : 0 \leq k < N :\ A.k \rangle\}$

d) Const $E : Num,\ N : Int,\ A : array[0, N)\ ofNum;$
Var $i : Int,\ r : Bool;$
$\{N \geq 0\}$
$i, r := 0, False$ **;**
do $i \neq N \wedge \neg r \to$
if $A.i = E \to r := True$
▯ $A.i \neq E \to$ **skip**
fi ;
$i := i + 1$
od
$\{\langle \exists k\ : 0 \leq k < N :\ A.k = E \rangle \Rightarrow A.i = E\}$

Ejercicios extra

17. Calcule las precondiciones más débiles (*weakest preconditions*) P de modo que sean correctos los siguientes programas anotados. Agregue además las anotaciones intermedias en caso que haya sentencias compuestas con ";". Asuma 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) $\{P\}\ a := a \equiv b;\ b := a \equiv b;\ a := a \equiv b\ \{(a \equiv B) \wedge (b \equiv A)\}$

18. Demuestre que si la terna de Hoare (a) es correcta, entonces la terna (b) también lo es:

a) $\{P\}$ **if** $B_0 \to S_0$ ▯ $B_1 \to S_1$ **fi** $\{Q\}$

b) $\{P\}$ **if** $B_0 \to S_0$ ▯ $\neg B_0 \to S_1$ **fi** $\{Q\}$

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

19. *Swap:* Considere los siguientes programas que intercambian los valores de dos variables x e y de tipo Int:

$x, y := y, x$

$z := x;$ $x := y;$ $y := z$

$x := x - y;$ $y := x + y;$ $x := y - x$

Especifique el *swap* (con pre y postcondición), y verifique que los programas satisfacen la especificación.

20. 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** $B_0 \to S$ □ $B_1 \to S$ **fi** — *ii*) S

c) *i*) **if** $B_0 \to S; S_0; T$ □ $B_1 \to S; S_1; T$ **fi** — *ii*) **if** $B_0 \to S; S_0$ □ $B_1 \to S; S_1$ **fi**; T — *iii*) $S;$ **if** $B_0 \to S_0; T$ □ $B_1 \to S_1; T$ **fi**

Laboratorio 10 (Funciones en C) Escribí los siguientes programas:

1. `entradas.c` que lee una variable de tipo `int` y la imprime por pantalla. En esta ocasión el programa debe utilizar dos funciones a definir (además de la función `main`):

   - una función que le pide un entero al usuario y lo devuelve, con prototipo:

     ```
     int pedir_entero(void);
     ```

     Recordar avisarle al usuario mediante un mensaje que se espera que ingrese un valor.

   - que toma un entero como parámetro y lo imprime:

     ```
     void imprimir_entero(int x);
     ```

   - Pensá cómo agregar un parámetro más llamado `name` de tipo `char` a las funciones anteriores y usarlo en el mensaje que se muestra al usuario. Entonces por ejemplo para la función `pedir_entero` si se ejecuta:

     ```
     n = pedir_entero('n');
     ```

     debería indicarse en el mensaje que se está pidiendo un valor para almacenar en la variable de nombre `n`. Notar que no hay forma de impedir que alguien use "mal" la función:

     ```
     x = pedir_entero('n');
     ```

     en ese caso el mensaje indicará que el valor se almacena en `n` aunque en realidad se guarda en una variable llamada `x`.

     Escribí de nuevo ambas funciones con este parámetro extra (debe estar como primer parámetro de las funciones).

2. `entradas_bool.c` que lee una variable de tipo `bool` y la imprime por pantalla. El programa debe utilizar dos funciones a definir (además de la función `main`):

   - una función que le pide un booleano al usuario y lo devuelve, con prototipo:

     ```
     bool pedir_booleano(void);
     ```

   - que toma un booleano como parámetro e imprime un mensaje `"verdadero"` o `"falso"` según sea su valor de verdad:

     ```
     void imprimir_booleano(bool x);
     ```

   - Agregar a las funciones un parámetro llamado `name` de tipo `char` con la misma funcionalidad descripta en el tercer ítem del apartado 1

3. Escribí el programa del ejercicio 6, pero utilizando las funciones del ejercicio anterior. ¿Qué ventajas encontras en esta nueva versión?. ¿Podrías escribir alguna otra función en ese ejercicio, cual?. ¿En qué otros ejercicios de ese Proyecto lo podrías utilizar?. Reescribí los programas donde puedas utilizarlas.

4. En un archivo `saludos.c` implementar las siguientes funciones (además de la `main`):

   - una que imprime el string `"hola"`, que no toma ni devuelve parámetros, con prototipo:

     ```
     void imprimir_hola(void);
     ```

   - similar a la anterior con la el string `"chau"`:

     ```
     void imprimir_chau(void);
     ```

   ese programa tiene que imprimir dos veces "hola" seguido de dos veces "chau", llamando a las dos últimas funciones desde el `main`.

   ayuda: Se debe entender cómo corre el flujo de ejecución de este programa leyendo su código fuente.