

Demostraciones de corrección

Algoritmos y Estructuras de Datos I

Triplas de Hoare

$\{P\}$ **codigo** $\{Q\}$

Triplas de Hoare

$\{P\}$ **codigo** $\{Q\}$

¿Es la siguiente tripla válida?

$\{x \geq 4\}$

$x := x + 2$

$\{x \geq 5\}$

Triplas de Hoare

$\{P\}$ **codigo** $\{Q\}$

¿Es la siguiente tripla válida?

$$\{x \geq 4\}$$

$x := x + 2$

$$\{x \geq 5\}$$

¿Es $\{x \geq 4\}$ la precondition más débil para el programa $x := x + 2$ y la postcondición $\{x \geq 5\}$?

Ejercicio 1

Indicar como se debería completar la precondition para poder demostrar corrección en cada caso:

```
proc transformarEnPar (inout n:  $\mathbb{Z}$ ) {  
  Pre { $n = N_0 \wedge ??$ }  
  Post { $esPar(n) \wedge n > N_0$ }  
}
```

Ejercicio 1

Indicar como se debería completar la precondition para poder demostrar corrección en cada caso:

```
proc transformarEnPar (inout n:  $\mathbb{Z}$ ) {  
  Pre { $n = N_0 \wedge ??$ }  
  Post { $esPar(n) \wedge n > N_0$ }  
}
```

Programa 1

S1: $n := 2 * n$

Programa 2

S2: $n := n + 1$

Ejercicio 2

Demostrar que el programa dado es correcto respecto a la siguiente especificación:

```
proc restaYpromedio (inout a:  $\mathbb{Z}$ , inout b:  $\mathbb{Z}$ ) {  
  Pre  $\{a = A_0 \wedge b = B_0\}$   
  Post  $\{a = A_0 - B_0 \wedge b = (A_0 + B_0)/2\}$   
}
```

Ejercicio 2

Demostrar que el programa dado es correcto respecto a la siguiente especificación:

```
proc restaYpromedio (inout a:  $\mathbb{Z}$ , inout b:  $\mathbb{Z}$ ) {  
  Pre  $\{a = A_0 \wedge b = B_0\}$   
  Post  $\{a = A_0 - B_0 \wedge b = (A_0 + B_0)/2\}$   
}
```

S1: aux := a

S2: a := a-b

S3: b := (aux + b) / 2

Ejercicio 3

Demostrar que el programa dado es correcto respecto a la siguiente especificación:

```
proc swap (inout a:  $\mathbb{Z}$ , inout b:  $\mathbb{Z}$ ) {  
  Pre  $\{a = A_0 \wedge b = B_0 \wedge a \neq 0 \wedge b \neq 0\}$   
  Post  $\{a = B_0 \wedge b = A_0\}$   
}
```

Ejercicio 3

Demostrar que el programa dado es correcto respecto a la siguiente especificación:

```
proc swap (inout a:  $\mathbb{Z}$ , inout b:  $\mathbb{Z}$ ) {  
  Pre { $a = A_0 \wedge b = B_0 \wedge a \neq 0 \wedge b \neq 0$ }  
  Post { $a = B_0 \wedge b = A_0$ }  
}
```

S1: $a := a*b$

S2: $b := a/b$

S3: $a := a/b$

Ejercicio 4

Completar la especificación de diferenciaPositiva de forma tal que se pueda demostrar que ambos programas son correctos.

```
proc diferenciaPositiva (in a:  $\mathbb{Z}$ , in b:  $\mathbb{Z}$ , out res:  $\mathbb{Z}$ ) {  
  Pre {??}  
  Post {res = |a - b|}  
}
```

Ejercicio 4

Completar la especificación de diferenciaPositiva de forma tal que se pueda demostrar que ambos programas son correctos.

```
proc diferenciaPositiva (in a:  $\mathbb{Z}$ , in b:  $\mathbb{Z}$ , out res:  $\mathbb{Z}$ ) {  
  Pre {??}  
  Post {res = |a - b|}  
}
```

► Programa 1

S1: res := a-b

► Programa 2

S2: if (a > b) then res := a - b else res := b - a
endif

Ejercicio 5

Completar la especificación de `sumarTodos` de forma tal que se pueda demostrar que el siguiente programa es correcto.

```
proc sumarTodos (in s: seq<ℤ>, in n: ℤ, inout suma: ℤ) {  
  Pre {...}  
  Post {suma =  $\sum_{i=0}^{|s|-1} s[i]$ }  
}
```

Ejercicio 5

Completar la especificación de `sumarTodos` de forma tal que se pueda demostrar que el siguiente programa es correcto.

```
proc sumarTodos (in s: seq<ℤ>, in n: ℤ, inout suma: ℤ) {  
  Pre {...}  
  Post {suma =  $\sum_{i=0}^{|s|-1} s[i]$ }  
}
```

S: `suma := suma + s[n-1]`

Ejercicio 6

Completar la especificación de `todoPositivos` de forma tal que se pueda demostrar que el siguiente programa es correcto.

```
proc todoPositivos (inout s: seq⟨ℤ⟩) {  
  Post { |s| = |S0| ∧L (∀ i : ℤ) 0 ≤ i < |s| →L s[i] > 0 }  
}
```

Ejercicio 6

Completar la especificación de `todoPositivos` de forma tal que se pueda demostrar que el siguiente programa es correcto.

```
proc todoPositivos (inout s: seq⟨ℤ⟩) {  
  Post { |s| = |S0| ∧L (∀ i : ℤ) 0 ≤ i < |s| →L s[i] > 0 }  
}
```

S: `s[0] := -1 * s[0]`

Ejercicio 7

Especificación

```
proc sumar (in s: seq<ℤ>, out  
result: ℤ) {  
  Pre {true}  
  Post {result =  $\sum_{j=0}^{|s|-1} s[j]$ }  
}
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L result = \sum_{j=0}^{i-1} s[j]$$

a) Escribir la precondition y la postcondición del ciclo.

Implementación en SmallLang

```
result := 0;  
i := 0;  
while (i < s.size()) do  
  result := result + s[i];  
  i := i + 1;  
endwhile
```

Ejercicio 7

Especificación

```
proc sumar (in s: seq<ℤ>, out  
result: ℤ) {  
  Pre {true}  
  Post {result =  $\sum_{j=0}^{|s|-1} s[j]$ }  
}
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L result = \sum_{j=0}^{i-1} s[j]$$

- b) ¿Qué punto falla en la demostración de corrección si el primer término del invariante se reemplaza por “ $0 \leq i < |s|$ ”?

Implementación en SmallLang

```
result := 0;  
i := 0;  
while (i < s.size()) do  
  result := result + s[i];  
  i := i + 1;  
endwhile
```

Ejercicio 7

Especificación

```
proc sumar (in s: seq<ℤ>, out  
result: ℤ) {  
  Pre {true}  
  Post {result =  $\sum_{j=0}^{|s|-1} s[j]$ }  
}
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L result = \sum_{j=0}^{i-1} s[j]$$

- c) ¿Qué punto falla en la demostración de corrección si el límite superior de la sumatoria (que es “ $i - 1$ ”) se reemplaza por “ i ”?

Implementación en SmallLang

```
result := 0;  
i := 0;  
while (i < s.size()) do  
  result := result + s[i];  
  i := i + 1;  
endwhile
```

Ejercicio 7

Especificación

```
proc sumar (in s: seq⟨ℤ⟩, out  
result: ℤ) {  
  Pre {true}  
  Post {result =  $\sum_{j=0}^{|s|-1} s[j]$ }  
}
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L result = \sum_{j=0}^{i-1} s[j]$$

Implementación en SmallLang

```
result := 0;  
i := 0;  
while (i < s.size()) do  
  result := result + s[i];  
  i := i + 1;  
endwhile
```

- d) ¿Qué punto falla en la demostración de corrección si se invierte el orden de las dos instrucciones del cuerpo del ciclo?

Ejercicio 7

Especificación

```
proc sumar (in s: seq<ℤ>, out  
result: ℤ) {  
  Pre {true}  
  Post {result =  $\sum_{j=0}^{|s|-1} s[j]$ }  
}
```

Invariante de Ciclo

$$I \equiv 0 \leq i \leq |s| \wedge_L result = \sum_{j=0}^{i-1} s[j]$$

- e) Demostrar formalmente la corrección parcial del ciclo, usando el teorema del invariante.

Implementación en SmallLang

```
result := 0;  
i := 0;  
while (i < s.size()) do  
  result := result + s[i];  
  i := i + 1;  
endwhile
```