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?

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

$$\mathbf{x := x + 2}$$

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

Triplas de Hoare

$\{P\}$ **codigo** $\{Q\}$
¿Es la siguiente tripla válida?

$$\{x \geq 4\}$$
$$\mathbf{x := x + 2}$$
$$\{x \geq 5\}$$

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

Precondición más débil (WP)

Definición. La precondición más débil de un programa **S** respecto de una postcondición Q es el predicado P más débil posible tal que $\{P\}\mathbf{S}\{Q\}$.

Precondición más débil (WP)

Definición. La precondición más débil de un programa **S** respecto de una postcondición Q es el predicado P más débil posible tal que $\{P\}\mathbf{S}\{Q\}$. **Notación.** $wp(\mathbf{S}, Q)$.

Precondición más débil (WP)

Definición. La precondición más débil de un programa **S** respecto de una postcondición Q es el predicado P más débil posible tal que $\{P\}\mathbf{S}\{Q\}$. **Notación.** $wp(\mathbf{S}, Q)$.

Ejemplo S: $\mathbf{x := x + 2}$ y $Q : x \geq 5$

Precondición más débil (WP)

Definición. La precondición más débil de un programa **S** respecto de una postcondición Q es el predicado P más débil posible tal que $\{P\}\mathbf{S}\{Q\}$. **Notación.** $wp(\mathbf{S}, Q)$.

Ejemplo S: $\mathbf{x := x + 2}$ y $Q : x \geq 5$
$wp(\mathbf{S}, Q) = x \geq 3$

Lenguaje *SmallLang*

- Variables

Lenguaje *SmallLang*

- Variables
- Instrucciones
  1. Nada: Instrucción **skip** que no hace nada.

Lenguaje *SmallLang*

- Variables
- Instrucciones
  1. Nada: Instrucción **skip** que no hace nada.
  2. Asignación: Instrucción **x := E**.

Lenguaje *SmallLang*

- Variables
- Instrucciones
  1. Nada: Instrucción **skip** que no hace nada.
  2. Asignación: Instrucción **x := E**.
- Estructuras de control:

Lenguaje *SmallLang*

- Variables
- Instrucciones
  1. Nada: Instrucción **skip** que no hace nada.
  2. Asignación: Instrucción **x := E**.
- Estructuras de control:
  1. Secuencia: **S1; S2** es un programa, si **S1** y **S2** son dos programas.

Lenguaje *SmallLang*

- Variables
- Instrucciones
  1. Nada: Instrucción **skip** que no hace nada.
  2. Asignación: Instrucción **x := E**.
- Estructuras de control:
  1. Secuencia: **S1; S2** es un programa, si **S1** y **S2** son dos programas.
  2. Condicional: **if B then S1 else S2 endif** es un programa, si **B** es una expresión lógica y **S1** y **S2** son dos programas.

Lenguaje *SmallLang*

- Variables
- Instrucciones
  1. Nada: Instrucción **skip** que no hace nada.
  2. Asignación: Instrucción **x := E**.
- Estructuras de control:
  1. Secuencia: **S1; S2** es un programa, si **S1** y **S2** son dos programas.
  2. Condicional: **if B then S1 else S2 endif** es un programa, si **B** es una expresión lógica y **S1** y **S2** son dos programas.
  3. Ciclo: **while B do S endwhile** es un programa, si **B** es una expresión lógica y **S** es un programa.

Definiciones

- Dada una expresión E, llamamos $\text{def}(E)$ a las condiciones necesarias para que E esté definida.

Definiciones

- Dada una expresión E, llamamos $\mathrm{def}(E)$ a las condiciones necesarias para que E esté definida.
- Dado un predicado Q, el predicado Q^x_E se obtiene reemplazando en Q todas las apariciones libres de la variable x por E.

Definiciones

- Dada una expresión E, llamamos $\mathrm{def}(E)$ a las condiciones necesarias para que E esté definida.
- Dado un predicado Q, el predicado Q_E^x se obtiene reemplazando en Q todas las apariciones libres de la variable x por E.

Axiomas

- **Axioma 1.** $wp(\mathbf{x := E}, Q) \equiv \mathrm{def}(E) \wedge_L Q^x_E$.

Axiomas

- **Axioma 1.** $wp(\mathbf{x} := \mathbf{E}, Q) \equiv \mathrm{def}(E) \wedge_L Q^x_E$.
- **Axioma 2.** $wp(\mathbf{skip}, Q) \equiv Q$.

Axiomas

- **Axioma 1.** $wp(\mathbf{x := E}, Q) \;\equiv\; \mathrm{def}(E) \wedge_L Q^x_E$.
- **Axioma 2.** $wp(\mathbf{skip}, Q) \;\equiv\; Q$.
- **Axioma 3.** $wp(\mathbf{S1;\ S2}, Q) \;\equiv\; wp(\mathbf{S1}, wp(\mathbf{S2}, Q))$.

Axiomas

- **Axioma 1.** $wp(\mathbf{x := E}, Q) \equiv \mathrm{def}(E) \wedge_L Q^x_E$.
- **Axioma 2.** $wp(\mathbf{skip}, Q) \equiv Q$.
- **Axioma 3.** $wp(\mathbf{S1;\ S2}, Q) \equiv wp(\mathbf{S1}, wp(\mathbf{S2}, Q))$.
- **Axioma 4.** Si $\mathbf{S} = \mathbf{if\ B\ then\ S1\ else\ S2\ endif}$, entonces

$$wp(\mathbf{S}, Q) \equiv def(B) \wedge_L \Big((B \wedge wp(\mathbf{S1}, Q)) \vee (\neg B \wedge wp(\mathbf{S2}, Q)) \Big)$$

Axiomas

- **Axioma 1.** $wp(\mathbf{x := E}, Q) \equiv \mathrm{def}(E) \land_L Q^x_E$.
- **Axioma 2.** $wp(\mathbf{skip}, Q) \equiv Q$.
- **Axioma 3.** $wp(\mathbf{S1;\ S2}, Q) \equiv wp(\mathbf{S1}, wp(\mathbf{S2}, Q))$.
- **Axioma 4.** Si $\mathbf{S} =$ **if B then S1 else S2 endif**, entonces

$$wp(\mathbf{S}, Q) \equiv def(B) \land_L \Big((B \land wp(\mathbf{S1}, Q)) \lor (\neg B \land wp(\mathbf{S2}, Q))\Big)$$

Ejercicio 1

```
proc transformarEnPar (inout n: ℤ) {
  Pre {n = N₀ ∧ ??}
  Post {esPar(n) ∧ n > N₀}
}
```

Ejercicio 1

```
proc transformarEnPar (inout n: ℤ) {
  Pre {n = N_0 ∧ ??}
  Post {esPar(n) ∧ n > N_0}
}
```

Programa 1

S1: `n := 2*n`

Programa 2

S2: `n := n + 1`

Ejercicio 1

```
proc transformarEnPar (inout n: ℤ) {
  Pre {n = N_0 ∧ ??}
  Post {esPar(n) ∧ n > N_0}
}
```

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

S1: `n := 2*n`

$\{Q : n \ mod \ 2 = 0 \land n > N_0\}$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

S2: `n := n + 1`

$\{Q : n \ mod \ 2 = 0 \land n > N_0\}$

Ejercicio 1

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

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv \{\text{def}(2 * n) \land_L (2 * n) \ mod \ 2 = 0 \land 2 * n > N_0\}$

S1: `n := 2*n`

$\{Q : n \ mod \ 2 = 0 \land n > N_0\}$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

S2: `n := n + 1`

$\{Q : n \ mod \ 2 = 0 \land n > N_0\}$

Ejercicio 1

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

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv\ \{\text{def}(2 * n)\ \wedge_L\ (2 * n)\ mod\ 2 = 0 \wedge 2 * n > N_0\}$

$\equiv\ \{\text{def}(n) \wedge_L\ True \wedge n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Ejercicio 1

```
proc transformarEnPar (inout n: ℤ) {
  Pre {n = N_0 ∧ ??}
  Post {esPar(n) ∧ n > N_0}
}
```

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv \{\text{def}(2*n) \land_L (2*n)\ mod\ 2 = 0 \land 2*n > N_0\}$

$\equiv \{\text{def}(n) \land_L True \land n > N_0/2\}$

$\equiv \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \land n > N_0\}$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \land n > N_0\}$

Ejercicio 1

```
proc transformarEnPar (inout n: ℤ) {
   Pre {n = N_0 ∧ ??}
   Post {esPar(n) ∧ n > N_0}
}
```

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv \{\text{def}(2*n) \wedge_L (2*n) \ mod \ 2 = 0 \wedge 2*n > N_0\}$

$\equiv \{\text{def}(n) \wedge_L \ True \wedge n > N_0/2\}$

$\equiv \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n \ mod \ 2 = 0 \wedge n > N_0\}$

Necesitamos que $P \rightarrow n > N_0$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

S2: `n := n + 1`

$\{Q : n \ mod \ 2 = 0 \wedge n > N_0\}$

Ejercicio 1

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

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv\ \{\text{def}(2*n)\ \land_L\ (2*n)\ mod\ 2 = 0 \land 2*n > N_0\}$

$\equiv\ \{\text{def}(n) \land_L\ True \land n > N_0/2\}$

$\equiv\ \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \land n > N_0\}$

Necesitamos que $P \rightarrow n > N_0$

$P : n = N_0 \land n > 0$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \land n > N_0\}$

Ejercicio 1

```
proc transformarEnPar (inout n: ℤ) {
   Pre {n = N_0 ∧ ??}
   Post {esPar(n) ∧ n > N_0}
}
```

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv \{\text{def}(2*n) \wedge_L (2*n) \ mod \ 2 = 0 \wedge 2*n > N_0\}$

$\equiv \{\text{def}(n) \wedge_L \ True \wedge n > N_0/2\}$

$\equiv \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n \ mod \ 2 = 0 \wedge n > N_0\}$

Necesitamos que $P \rightarrow n > N_0$

$P : n = N_0 \wedge n > 0$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

$\equiv \{\text{def}(n+1) \wedge_L (n+1) \ mod \ 2 = 0 \wedge (n+1) > N_0\}$

S2: `n := n + 1`

$\{Q : n \ mod \ 2 = 0 \wedge n > N_0\}$

Ejercicio 1

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

Programa 1

$\{wp($`n := 2*n`$, Q)\}$

$\equiv \{\text{def}(2*n)\ \wedge_L\ (2*n)\ mod\ 2 = 0 \wedge 2*n > N_0\}$

$\equiv \{\text{def}(n) \wedge_L\ True \wedge n > N_0/2\}$

$\equiv \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Necesitamos que $P \rightarrow n > N_0$

$P : n = N_0 \wedge n > 0$

Programa 2

$\{wp($`n := n + 1`$, Q)\}$

$\equiv \{\text{def}(n+1)\ \wedge_L\ (n+1)\ mod\ 2 = 0 \wedge (n+1) > N_0\} \equiv$

$\{\text{def}(n) \wedge_L\ n\ mod\ 2 = 1 \wedge n \geq N_0/\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Ejercicio 1

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

Programa 1

$\{wp($`n := 2*n`$, Q)\}$

$\equiv \{\text{def}(2 * n) \land_L (2 * n)\ mod\ 2 = 0 \land 2 * n > N_0\}$

$\equiv \{\text{def}(n) \land_L\ True \land n > N_0/2\}$

$\equiv \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \land n > N_0\}$

Necesitamos que $P \rightarrow n > N_0$

$P : n = N_0 \land n > 0$

Programa 2

$\{wp($`n := n + 1`$, Q)\}$

$\equiv \{\text{def}(n+1) \land_L (n+1)\ mod\ 2 = 0 \land (n+1) > N_0\} \equiv$

$\{\text{def}(n) \land_L\ n\ mod\ 2 = 1 \land n \geq N_0/\}$

$\equiv \{n\ mod\ 2 = 1 \land n \geq N_0\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \land n > N_0\}$

Ejercicio 1

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

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv \ \{\text{def}(2 * n) \ \wedge_L \ (2 * n) \ mod \ 2 = 0 \wedge 2 * n > N_0\}$

$\equiv \ \{\text{def}(n) \wedge_L \ True \wedge n > N_0/2\}$

$\equiv \ \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n \ mod \ 2 = 0 \wedge n > N_0\}$

Necesitamos que $P \to n > N_0$

$P : n = N_0 \wedge n > 0$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

$\equiv \ \{\text{def}(n+1) \ \wedge_L \ (n+1) \ mod \ 2 = 0 \wedge (n+1) > N_0\} \ \equiv$

$\{\text{def}(n) \wedge_L \ n \ mod \ 2 = 1 \wedge n \geq N_0/\}$

$\equiv \ \{n \ mod \ 2 = 1 \wedge n \geq N_0\}$

S2: `n := n + 1`

$\{Q : n \ mod \ 2 = 0 \wedge n > N_0\}$

Necesitamos que

$P \to (n \ mod \ 2 = 1 \wedge n \geq N_0)$

Ejercicio 1

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

Programa 1

$\{wp(\texttt{n := 2*n}, Q)\}$

$\equiv\ \{\mathrm{def}(2 * n)\ \wedge_L\ (2 * n)\ mod\ 2 = 0 \wedge 2 * n > N_0\}$

$\equiv\ \{\mathrm{def}(n) \wedge_L\ True \wedge n > N_0/2\}$

$\equiv\ \{n > N_0/2\}$

S1: `n := 2*n`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Necesitamos que $P \rightarrow n > N_0$

$P : n = N_0 \wedge n > 0$

Programa 2

$\{wp(\texttt{n := n + 1}, Q)\}$

$\equiv\ \{\mathrm{def}(n + 1)\ \wedge_L\ (n + 1)\ mod\ 2 = 0 \wedge (n + 1) > N_0\}\ \equiv$

$\{\mathrm{def}(n) \wedge_L\ n\ mod\ 2 = 1 \wedge n \geq N_0/\}$

$\equiv\ \{n\ mod\ 2 = 1 \wedge n \geq N_0\}$

S2: `n := n + 1`

$\{Q : n\ mod\ 2 = 0 \wedge n > N_0\}$

Necesitamos que

$P \rightarrow (n\ mod\ 2 = 1 \wedge n \geq N_0)$

$P : n = N_0 \wedge n\ mod\ 2 = 1$

Ejercicio 2

```
proc restaYpromedio (inout a: ℤ, inout b: ℤ) {
  Pre {a = A_0 ∧ b = B_0}
  Post {a = A_0 − B_0 ∧ b = (A_0 + B_0)/2}
}
```

Ejercicio 2

proc restaYpromedio (inout a: $\mathbb{Z}$, inout b: $\mathbb{Z}$) {

  Pre $\{a = A_0 \land b = B_0\}$

  Post $\{a = A_0 - B_0 \land b = (A_0 + B_0)/2\}$

}

S1: `aux := a`

S2: `a := a-b`

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

Ejercicio 3

```
proc swap (inout a: ℤ, inout b: ℤ) {
   Pre {a = A0 ∧ b = B0 ∧ a ≠ 0 ∧ b ≠ 0}
   Post {a = B0 ∧ b = A0}
}
```

Ejercicio 3

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

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

Ejercicio 4

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

$\texttt{proc sumarTodos}$ (in s: $seq\langle\mathbb{Z}\rangle$, in n: $\mathbb{Z}$, inout suma: $\mathbb{Z}$) $\{$

$\texttt{Post}$ $\{suma = \sum_{i_0}^{|s|-1} s[i]\}$

$\}$

Ejercicio 5

proc sumarTodos (in s: $seq\langle\mathbb{Z}\rangle$, in n: $\mathbb{Z}$, inout suma: $\mathbb{Z}$) {

Post $\{suma = \sum_{i_0}^{|s|-1} s[i]\}$
}

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

Ejercicio 5

proc sumarTodos (in s: $seq\langle\mathbb{Z}\rangle$, in n: $\mathbb{Z}$, inout suma: $\mathbb{Z}$) {
    Pre $\{suma = suma_0 \wedge n = |s| \wedge suma = \sum_{i=0}^{|s|-2} s[i]\}$
    Post $\{suma = \sum_{i_0}^{|s|-1} s[i]\}$
}

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

Teorema del Invariante (corrección)

- **Teorema.** Si def(B) y existe un predicado I tal que
  1. $P \Rightarrow I$,
  2. $\{I \wedge B\}\ S\ \{I\}$,
  3. $I \wedge \neg B \Rightarrow Q$,

  ... y el ciclo termina, entonces
  $\{P\}$ **while B do S endwhile** $\{Q\}$.

¿Cómo sabemos si el ciclo termina?

Teorema de Terminación

Sea $\mathbb{V}$ el producto cartesiano de los dominios de las variables del programa y sea I un invariante del ciclo **while B do S endwhile**. Si existe una función $fv : \mathbb{V} \to \mathbb{Z}$ tal que

1. $(\forall v_0 : \mathbb{Z})(\ \{I \land B \land fv = v_0\}\ \mathbf{S}\ \{fv < v_0\}),$
2. $I \land fv \leq 0 \Rightarrow \neg B,$

... entonces la ejecución del ciclo **while B do S endwhile** siempre termina.

Ejercicio 6

```
proc productoria (in a: seq⟨ℤ⟩, out prod) {
  Pre {length(a) mod 2 = 0}
  Post {prod = ∏_{i=0}^{length(a)−1} s[i]}
}
```

Ejercicio 6

$\texttt{proc productoria}$ (in a: $seq\langle\mathbb{Z}\rangle$, out prod) $\{$
$\quad\texttt{Pre}$ $\{length(a)\ mod\ 2 = 0\}$
$\quad\texttt{Post}$ $\{prod = \prod_{i=0}^{length(a)-1} s[i]\}$
$\}$

Pc: $i = 0 \land prod = 1$

Ejercicio 6

```
proc productoria (in a: seq⟨ℤ⟩, out prod) {
  Pre {length(a) mod 2 = 0}
  Post {prod = ∏_{i=0}^{length(a)-1} s[i]}
}
```

Pc: $i = 0 \land prod = 1$

I: $0 \leq i \leq a.length \land i \ mod \ 2 = 0 \land_L prod = \prod_{j=0}^{i-1} s[j]$

Ejercicio 6

proc productoria (in a: $seq\langle\mathbb{Z}\rangle$, out prod) {
  Pre $\{length(a)\ mod\ 2 = 0\}$
  Post $\{prod = \prod_{i=0}^{length(a)-1} s[i]\}$
}

Pc: $i = 0 \wedge prod = 1$
I: $0 \leq i \leq a.length \wedge i\ mod\ 2 = 0 \wedge_L prod = \prod_{j=0}^{i-1} s[j]$

```
i := 0;
prod := 1;

while( i < a.length-1 ) do
  prod := prod * a[i] * a[i+1];
  i:=i+2
endwhile
```

Ejercicio 6

- P_c:
- Q_c:
- I:
- fv :

```
i := 0;
prod := 1;

while( i < a.length-1 ) do
  prod := prod * a[i] * a[i+1];
  i:=i+2
endwhile
```

Ejercicio 6

- P_c: $i = 0 \wedge prod = 1$
- Q_c:
- I:
- fv :

```
i := 0;
prod := 1;

while( i < a.length-1 ) do
  prod := prod * a[i] * a[i+1];
  i:=i+2
endwhile
```

Ejercicio 6

- P_c: $i = 0 \wedge prod = 1$
- Q_c: $i = n \wedge_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I:
- fv :

```
i := 0;
prod := 1;

while( i < a.length-1 ) do
  prod := prod * a[i] * a[i+1];
  i:=i+2
endwhile
```

Ejercicio 6

- P_c: $i = 0 \wedge prod = 1$
- Q_c: $i = n \wedge_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq n \wedge i \ mod \ 2 = 0 \wedge_L prod = \prod_{j=0}^{i-1} s[j]$
- fv :

```
i := 0;
prod := 1;

while( i < a.length-1 ) do
  prod := prod * a[i] * a[i+1];
  i:=i+2
endwhile
```

Ejercicio 6

- P_c: $i = 0 \land prod = 1$
- Q_c: $i = n \land_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq n \land i \ mod \ 2 = 0 \land_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

```
i := 0;
prod := 1;

while( i < a.length-1 ) do
  prod := prod * a[i] * a[i+1];
  i:=i+2
endwhile
```

Ejercicio 6

- P_c: $i = 0 \wedge prod = 1$
- Q_c: $i = length(a) \wedge_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq length(a) \wedge i \ mod\ 2 = 0 \wedge_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

Ejercicio 6

- P_c: $i = 0 \land prod = 1$
- Q_c: $i = length(a) \land_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq length(a) \land i \ mod \ 2 = 0 \land_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

- $P_c \Rightarrow I$

Ejercicio 6

- P_c: $i = 0 \land prod = 1$
- Q_c: $i = length(a) \land_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq length(a) \land i \ mod \ 2 = 0 \land_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

- $P_c \Rightarrow I$
- $(I \land \neg B) \Rightarrow Q_c$

Ejercicio 6

- P_c: $i = 0 \wedge prod = 1$
- Q_c: $i = length(a) \wedge_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq length(a) \wedge i \ mod \ 2 = 0 \wedge_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

- $P_c \Rightarrow I$
- $(I \wedge \neg B) \Rightarrow Q_c$
- $(I \wedge fv \leq 0) \Rightarrow \neg B$

Ejercicio 6

- P_c: $i = 0 \wedge prod = 1$
- Q_c: $i = length(a) \wedge_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq length(a) \wedge i \ mod \ 2 = 0 \wedge_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

- $P_c \Rightarrow I$
- $(I \wedge \neg B) \Rightarrow Q_c$
- $(I \wedge fv \leq 0) \Rightarrow \neg B$
- $\{I \wedge B\}\ S\ \{I\}$

Ejercicio 6

- P_c: $i = 0 \land prod = 1$
- Q_c: $i = length(a) \land_L prod = \prod_{j=0}^{length(a)-1} s[j]$
- I: $0 \leq i \leq length(a) \land i \; mod \; 2 = 0 \land_L prod = \prod_{j=0}^{i-1} s[j]$
- fv : $length(a) - i$

- $P_c \Rightarrow I$
- $(I \land \neg B) \Rightarrow Q_c$
- $(I \land fv \leq 0) \Rightarrow \neg B$
- $\{I \land B\}\ S\ \{I\}$
- $(\forall v_0 : \mathbb{Z})(\{I \land B \land fv = v_0\}S\{fv < v_0\})$

Ejercicio 7

```
proc sumarUno (inout a: seq⟨ℤ⟩)  {
  Pre {length(a) mod 2 = 0 ∧ a = A_0}
  Post {length(a) = length(A_0) ∧ (∀k : ℤ)(0 ≤ k < length(a) ⇒_L
          a[k] = A_0[k] + 1)}
}
```

$$\texttt{proc sumarUno (inout a: } seq\langle\mathbb{Z}\rangle) \ \{$$
$$\texttt{Pre } \{length(a) \ mod \ 2 = 0 \wedge a = A_0\}$$
$$\texttt{Post } \{length(a) = length(A_0) \wedge (\forall k : \mathbb{Z})(0 \leq k < length(a) \Rightarrow_L a[k] = A_0[k] + 1)\}$$
$$\}$$

Ejercicio 7

proc sumarUno (inout a: $seq\langle\mathbb{Z}\rangle$) {

Pre $\{length(a)\ mod\ 2 = 0 \wedge a = A_0\}$

Post $\{length(a) = length(A_0) \wedge (\forall k : \mathbb{Z})(0 \leq k < length(a) \Rightarrow_L a[k] = A_0[k] + 1)\}$

}

```
i := 0;
j := a.length-1;

while( j > i ) do
  a[i] = a[i]+1;
  a[j] = a[j]+1;
  i := i+1;
  j := j-1
endwhile
```

Ejercicio 7

- P_c:

Ejercicio 7

- P_c: $i = 0 \wedge j = length(a) - 1 \wedge a = A_0$

Ejercicio 7

- P_c: $i = 0 \wedge j = length(a) - 1 \wedge a = A_0$
- Q_c:

Ejercicio 7

- P_c: $i = 0 \wedge j = length(a) - 1 \wedge a = A_0$
- Q_c: $j = i - 1 \wedge i + j = n - 1 \wedge$
  $(\forall k : \mathbb{Z})(enRango(k, a) \Rightarrow_L a[k] = A_0[k] + 1)$

Ejercicio 7

- P_c: $i = 0 \wedge j = length(a) - 1 \wedge a = A_0$
- Q_c: $j = i - 1 \wedge i + j = n - 1 \wedge$
  $(\forall k : \mathbb{Z})(enRango(k, a) \Rightarrow_L a[k] = A_0[k] + 1)$
- I:

Ejercicio 7

- P_c: $i = 0 \land j = length(a) - 1 \land a = A_0$
- Q_c: $j = i - 1 \land i + j = n - 1 \land$
  $(\forall k : \mathbb{Z})(enRango(k, a) \Rightarrow_L a[k] = A_0[k] + 1)$
- I: $0 \leq i \leq length(a)/2 \land$
  $length(a)/2 - 1 \leq j < length(a) \land$
  $j + i = length(a) - 1 \land$
  $(\forall k : \mathbb{Z})(i \leq k \leq j \Rightarrow_L a[k] = A_0[k]) \land$
  $(\forall k : \mathbb{Z})((enRango(k, a) \land (k < i \lor k > j)) \Rightarrow_L a[k] =$
  $A_0[k] + 1)$

Ejercicio 7

- P_c: $i = 0 \land j = length(a) - 1 \land a = A_0$
- Q_c: $j = i - 1 \land i + j = n - 1 \land$
  $(\forall k : \mathbb{Z})(enRango(k, a) \Rightarrow_L a[k] = A_0[k] + 1)$
- I: $0 \leq i \leq length(a)/2 \land$
  $length(a)/2 - 1 \leq j < length(a) \land$
  $j + i = length(a) - 1 \land$
  $(\forall k : \mathbb{Z})(i \leq k \leq j \Rightarrow_L a[k] = A_0[k]) \land$
  $(\forall k : \mathbb{Z})((enRango(k, a) \land (k < i \lor k > j)) \Rightarrow_L a[k] =$
  $A_0[k] + 1)$
- fv :

Ejercicio 7

- P_c: $i = 0 \wedge j = length(a) - 1 \wedge a = A_0$
- Q_c: $j = i - 1 \wedge i + j = n - 1 \wedge$
  $(\forall k : \mathbb{Z})(enRango(k, a) \Rightarrow_L a[k] = A_0[k] + 1)$
- I: $0 \leq i \leq length(a)/2 \wedge$
  $length(a)/2 - 1 \leq j < length(a) \wedge$
  $j + i = length(a) - 1 \wedge$
  $(\forall k : \mathbb{Z})(i \leq k \leq j \Rightarrow_L a[k] = A_0[k]) \wedge$
  $(\forall k : \mathbb{Z})((enRango(k, a) \wedge (k < i \vee k > j)) \Rightarrow_L a[k] =$
  $A_0[k] + 1)$
- $fv : j - i$