

**METODOLOGÍA DE LA PROGRAMACIÓN**  
**TEMA 3**  
**VERIFICACIÓN DE PROGRAMAS**

3.1. Introducción a la verificación de programas .....	2
3.1.1. Objetivo .....	2
3.1.2. Ejemplo de programa correcto .....	2
3.1.3. Ejemplo de programa incorrecto .....	2
3.1.4. Corrección parcial, terminación y corrección total .....	2
3.2. Cálculo de Hoare .....	4
3.2.1. Procedimiento a seguir .....	4
3.2.2. Axioma de la Asignación (AA) y Regla de la Consecuencia (RCN).....	4
3.2.2.1. Axioma de la Asignación (AA) .....	4
3.2.2.2. Ejemplo 1.....	5
3.2.2.3. Ejemplo 2.....	6
3.2.2.4. Ejemplo 3.....	7
3.2.2.5. Regla de la consecuencia (RCN) .....	7
3.2.2.6. Ejemplo 3 (continuación) .....	8
3.2.2.7. Ejemplo 4.....	9
3.2.2.8. Ejemplo 5.....	10
3.2.2.9. Ejemplo 6.....	12
3.2.2.10. Ejemplo 7.....	13
3.2.2.11. Ejemplo 8.....	14
3.2.3. Regla de la Composición (RCP).....	16
3.2.3.1. Ejemplo 9.....	16
3.2.4. Regla del While (RWH) .....	20
3.2.4.1. Ejemplo 10.....	21
3.2.4.2. Ejemplo 11.....	26
3.2.4.3. Ejemplo 12.....	27
3.2.4.4. Ejemplo 13.....	32
3.2.4.5. Ejemplo 14.....	38
3.2.4.6. Ejemplo 15.....	45
3.2.4.7. Ejemplo 16.....	50
3.2.4.8. Ejemplo 17.....	51
3.2.4.9. Ejemplo 18.....	53
3.2.4.10. Ejemplo 19.....	56
3.2.4.11. Ejemplo 20.....	58
3.2.4.12. Ejemplo 21.....	61

## **3.1. Introducción a la verificación de programas**

### **3.1.1. Objetivo**

La verificación de programas consiste en, dado un programa P y una especificación pre-post (precondición  $\{\phi\}$  y poscondición  $\{\psi\}$ ), decidir si el programa es correcto con respecto a la especificación pre-post. Es decir, la verificación responde a la siguiente pregunta: Si el programa P parte de un estado que cumple  $\{\phi\}$ , ¿termina en un estado que cumple  $\{\psi\}$ ?

### **3.1.2. Ejemplo de programa correcto**

$$\{\phi\} \equiv \{x = a \wedge y = b\}$$

```

aux := x;
x := y;
y := aux;

```

$$\{\psi\} \equiv \{x = b \wedge y = a\}$$

En este ejemplo, mediante las fórmulas  $\{\phi\}$  y  $\{\psi\}$  se especifica que si al principio x vale a e y vale b, después de ejecutar las tres asignaciones x valdrá b e y valdrá a.

En este tema se explicará cómo demostrar que ese programa es correcto.

### **3.1.3. Ejemplo de programa incorrecto**

$$\{\phi\} \equiv \{x = a \wedge y = b\}$$

```

y := x;
x := y;

```

$$\{\psi\} \equiv \{x = b \wedge y = a\}$$

También en este ejemplo, mediante las fórmulas  $\{\phi\}$  y  $\{\psi\}$  se especifica que si al principio x vale a e y vale b, después de ejecutar las tres asignaciones x valdrá b e y valdrá a.

En este tema se explicará también cómo demostrar que ese programa no es correcto.

### **3.1.4. Corrección parcial, terminación y corrección total**

Decir que el programa P es **parcialmente correcto** con respecto a la especificación pre-post dada mediante  $\{\phi\}$  y  $\{\psi\}$  significa que si el programa P comienza a ejecutarse en un estado que cumple lo expresado por  $\{\phi\}$ , entonces si el programa termina, terminará en un estado que cumpla lo expresado por la fórmula  $\{\psi\}$ .

Cuando se demuestra que el programa  $P$  es parcialmente correcto con respecto a  $\{\phi\}$  y  $\{\psi\}$ , no se garantiza que  $P$  termine.

Por ejemplo, si tenemos un programa que tiene la forma:

$\{\phi\} \equiv \{x = a \wedge y = b\}$ <u>while</u> $y \neq 0$ <u>loop</u> $x := x + 1;$ $y := y - 1;$ <u>end loop</u> ; $\{\psi\} \equiv \{x = a + b \wedge y = 0\}$	<b>Es parcialmente correcto pero no es totalmente correcto porque no se puede garantizar que siempre termine</b>
--	--

Se puede demostrar que es parcialmente correcto, es decir, que si al principio se cumple  $\{x = a \wedge y = b\}$ , ejecutamos el while y éste termina en algún momento, entonces se cumplirá  $\{x = a + b \wedge y = 0\}$ . Pero no podemos garantizar que ese while termine, porque si por ejemplo el valor de  $y$  es negativo al principio, es decir, si  $b$  es negativo, entonces ese while no terminará nunca.

Decir que el programa  $P$  es **totalmente correcto** con respecto a la especificación pre-post dada mediante  $\{\phi\}$  y  $\{\psi\}$  significa que si el programa  $P$  comienza a ejecutarse en un estado que cumple lo expresado por  $\{\phi\}$ , entonces el programa terminará y además terminará en un estado que cumpla lo expresado por la fórmula  $\{\psi\}$ . Cuando se demuestra que el programa  $P$  es totalmente correcto con respecto a  $\{\phi\}$  y  $\{\psi\}$ , se garantiza que  $P$  terminará.

$\{\phi\} \equiv \{x = a \wedge y = b \wedge b \geq 0\}$ <u>while</u> $y \neq 0$ <u>loop</u> $x := x + 1;$ $y := y - 1;$ <u>end loop</u> ; $\{\psi\} \equiv \{x = a + b \wedge y = 0\}$	<b>Es totalmente correcto porque siempre termina y además siempre termina cumpliendo la poscondición</b>
--	--

Se puede demostrar que es totalmente correcto, es decir, que si al principio se cumple  $\{x = a \wedge y = b \wedge b \geq 0\}$ , el programa terminará y además terminará en un estado en el que se cumple  $\{x = a + b \wedge y = 0\}$ .

Para probar que un programa  $\{\phi\} P \{\psi\}$  es totalmente correcto hay que demostrar que es parcialmente correcto y además que termina.

En el caso de la asignación y las instrucciones condicionales la corrección parcial y la corrección total coinciden. En el caso de las instrucciones iterativas además de probar la corrección parcial habrá que probar también la terminación.

Los programas que contienen instrucciones iterativas (while-loop, loop-exit-when, ...) pueden terminar o no terminar. Los programas que no contienen instrucciones iterativas no tienen el problema de la terminación, siempre terminan.

Para probar que un programa  $\{\phi\} P \{\psi\}$  es totalmente correcto hay que demostrar que es parcialmente correcto y además que termina

Corrección total = terminación + corrección parcial

## **3.2. Cálculo de Hoare**

El cálculo de Hoare consiste en un conjunto de axiomas y reglas a utilizar para probar la corrección del programa.

En los siguientes apartados se mostrará cómo utilizar este método para las instrucciones más habituales en los lenguajes de programación:

- Asignación.
- Composición.
- Estructuras iterativas (while).

### **3.2.1. Procedimiento a seguir**

Cuando se nos pida probar la corrección total de un programa  $P$  con respecto a una especificación pre-post dada mediante las fórmulas  $\{\phi\}$  y  $\{\psi\}$ , habrá que intentar probar la corrección total de dicho programa utilizando el cálculo de Hoare y luego:

- Si el programa es totalmente correcto, habrá que dar la demostración formal.
- Si el programa es parcialmente correcto pero no termina siempre, habrá que dar la demostración formal de la corrección parcial y habrá que dar también un ejemplo concreto que muestre que a veces el programa no termina.
- Si el programa no es parcialmente correcto, habrá que dar un ejemplo concreto en el que partiendo de un estado que cumpla la precondition  $\{\phi\}$ , tras ejecutar el programa se llegue a un estado que no cumpla la postcondición  $\{\psi\}$ .

### **3.2.2. Axioma de la Asignación (AA) y Regla de la Consecuencia (RCN)**

#### **3.2.2.1. Axioma de la Asignación (AA)**

Para empezar veremos cómo se verifica la corrección de una asignación utilizando el cálculo de Hoare.

$\{\phi\}$ $x := t;$ $\{\psi\}$
---------------------------------------

Una asignación  $x := t;$  es correcta con respecto a la precondition  $\{\phi\}$  y la poscondición  $\{\psi\}$  si se cumple  $\{\phi\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$ .

Dicho de otra manera, lo siguiente es correcto:

<b>Axioma de la asignación (AA)</b>	$\{\text{def}(t) \wedge \psi_x^t\}$ $x := t;$ $\{\psi\}$
---	--

$\text{def}(t)$  indicará en qué casos está definida la expresión  $t$  y la fórmula  $\psi_x^t$  es la fórmula  $\psi$  pero sustituyen todas las apariciones de  $x$  por  $t$ .

El axioma de la asignación viene a decir que si queremos que al final  $x$  cumpla la postcondición  $\psi$  y si previamente a  $x$  le vamos a asignar el valor de la expresión  $t$ , para que la poscondición se cumpla tendrá que ocurrir que la expresión  $t$  sea calculable y que al principio  $t$  cumpla  $\psi_x^t$ , es decir, que al principio  $t$  cumpla lo que queremos que  $x$  cumpla al final.

### 3.2.2.2. Ejemplo 1

¿Es correcto el siguiente programa?

$\{\phi\} \equiv \{1 \leq i \leq n \wedge A(i) > 0\}$ $x := A(i);$ $\{\psi\} \equiv \{x > 0\}$
--

Tenemos que hacer dos cosas:

- Calcular la fórmula  $\{\text{def}(t) \wedge \psi_x^t\}$ , a la que llamaremos  $\{\phi_1\}$  y donde  $t$  es  $A(i)$ .
- Comprobar si se cumple que  $\{\phi\}$  y  $\{\text{def}(t) \wedge \psi_x^t\}$  son iguales.

AA  $\swarrow$

$\{\phi\} \equiv \{1 \leq i \leq n \wedge A(i) > 0\}$ $\{\phi_1\} \equiv \{\text{def}(A(i)) \wedge \psi_x^{A(i)}\} \equiv \{1 \leq i \leq n \wedge A(i) > 0\}$ $x := A(i);$ $\{\psi\} \equiv \{x > 0\}$
--

En este caso  $\text{def}(A(i))$  es  $1 \leq i \leq n$  porque para poder calcular  $A(i)$  u obtener el valor de  $A(i)$  tiene que ocurrir que el índice  $i$  esté en el rango del vector  $A(1..n)$ , es decir, tiene que ocurrir que  $i$  tenga un valor comprendido entre 1 y  $n$ .

Como  $\phi$  y  $\phi_1$  son iguales, el programa es correcto con respecto a  $\phi$  y  $\psi$ .

Se puede decir que partimos de abajo, de  $\{\psi\}$ , y vamos hacia arriba calculando  $\{\phi_1\}$ , utilizando para ello  $\{\psi\}$  y la asignación  $x := A(i)$ , y por último comprobamos si  $\phi$  y  $\phi_1$  son iguales.

El axioma de la asignación viene a decir que si queremos que al final  $x$  sea mayor que 0 y si previamente a  $x$  le vamos a asignar  $A(i)$ , para que la postcondición se cumpla tendrá que ocurrir que  $A(i)$  sea calculable y que al principio  $A(i)$  sea mayor que 0. De esa manera tras asignar  $A(i)$  a  $x$  conseguiremos que  $x$  sea mayor que 0.

Como el programa es correcto hay que dar la demostración formal.

- **Demostración formal:**

$$1. \{ \varphi \} x := A(i); \{ \psi \} \text{ (AA)}$$

Mediante esa demostración formal estamos indicando que la asignación es correcta por el Axioma de la Asignación.

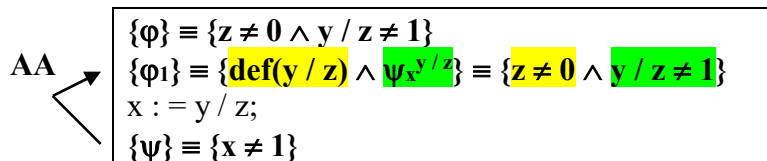
### 3.2.2.3. Ejemplo 2

¿Es correcto el siguiente programa?

$$\begin{aligned} \{ \varphi \} &\equiv \{ z \neq 0 \wedge y / z \neq 1 \} \\ x &:= y / z; \\ \{ \psi \} &\equiv \{ x \neq 1 \} \end{aligned}$$

Tenemos que hacer dos cosas:

- Calcular la fórmula  $\{ \text{def}(t) \wedge \psi_x^t \}$  a la que llamaremos  $\{ \varphi_1 \}$  y donde t es y / z.
- Comprobar si se cumple que  $\{ \varphi \}$  y  $\{ \text{def}(t) \wedge \psi_x^t \}$  son iguales.

AA 

$$\begin{aligned} \{ \varphi \} &\equiv \{ z \neq 0 \wedge y / z \neq 1 \} \\ \{ \varphi_1 \} &\equiv \{ \text{def}(y / z) \wedge \psi_x^{y / z} \} \equiv \{ z \neq 0 \wedge y / z \neq 1 \} \\ x &:= y / z; \\ \{ \psi \} &\equiv \{ x \neq 1 \} \end{aligned}$$

La expresión  $\text{def}(y / z)$  es  $z \neq 0$  porque para poder calcular  $y / z$  tiene que ocurrir que el valor z no sea cero.

Como  $\varphi$  y  $\varphi_1$  son iguales, el programa es correcto con respecto a  $\varphi$  y  $\psi$ .

Se puede decir que partimos de abajo, de  $\{ \psi \}$ , y vamos hacia arriba calculando  $\{ \varphi_1 \}$ , utilizando para ello  $\{ \psi \}$  y la asignación  $x := y / z$ , y por último comprobamos si  $\varphi$  y  $\varphi_1$  son iguales.

El axioma de la asignación viene a decir que si queremos que al final x sea distinto de 1 y si previamente a x le vamos a asignar  $y / z$ , para que la poscondición se cumpla tendrá que ocurrir que  $y / z$  sea calculable y que al principio  $y / z$  sea distinto de 1. De esa manera tras asignar  $y / z$  a x conseguiremos que x sea distinto de 1.

Como el programa es correcto hay que dar la demostración formal.

- **Demostración formal:**

$$1. \{ \varphi \} x := y / z; \{ \psi \} \text{ (AA)}$$

Mediante esa demostración formal estamos indicando que la asignación es correcta por el Axioma de la Asignación.

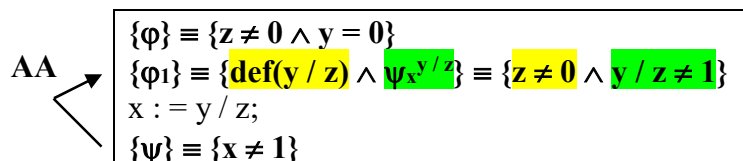
### 3.2.2.4. Ejemplo 3

¿Es correcto el siguiente programa?

$$\begin{aligned} \{\varphi\} &\equiv \{z \neq 0 \wedge y = 0\} \\ x &:= y / z; \\ \{\psi\} &\equiv \{x \neq 1\} \end{aligned}$$

Tenemos que hacer dos cosas:

- Calcular la fórmula  $\{\text{def}(t) \wedge \psi_x^t\}$  a la que llamaremos  $\{\varphi_1\}$  y donde  $t$  es  $y / z$ .
- Comprobar si se cumple que  $\{\varphi\}$  y  $\{\text{def}(t) \wedge \psi_x^t\}$  son iguales.

AA 

$$\begin{aligned} \{\varphi\} &\equiv \{z \neq 0 \wedge y = 0\} \\ \{\varphi_1\} &\equiv \{\text{def}(y / z) \wedge \psi_x^{y / z}\} \equiv \{z \neq 0 \wedge y / z \neq 1\} \\ x &:= y / z; \\ \{\psi\} &\equiv \{x \neq 1\} \end{aligned}$$

La expresión  $\text{def}(y / z)$  es  $z \neq 0$  porque para poder calcular  $y / z$  tiene que ocurrir que el valor  $z$  no sea cero.

En este caso  $\varphi$  y  $\varphi_1$  no son iguales, pero intuitivamente se ve que el programa es correcto con respecto a  $\varphi$  y  $\psi$ . Esto quiere decir que el Axioma de la Asignación no es suficiente para decidir si una asignación es o no es correcta. Nos hace falta la Regla de la Consecuencia.

### 3.2.2.5. Regla de la consecuencia (RCN)

Regla de la Consecuencia (RCN)

$\varphi \rightarrow \varphi_1, \{\varphi_1\} P \{\psi\}$
$\{\varphi\} P \{\psi\}$

Esta regla viene a decir que si la fórmula  $\varphi$  implica a la fórmula  $\varphi_1$  y  $\{\varphi_1\} P \{\psi\}$  es correcto, entonces  $\{\varphi\} P \{\psi\}$  es correcto.

Dicho de otra forma:

Si se cumple  $\varphi \rightarrow \varphi_1$  y  $\begin{array}{c} \{\varphi_1\} \\ P \\ \{\psi\} \end{array}$  es correcto, entonces  $\begin{array}{c} \{\varphi\} \\ P \\ \{\psi\} \end{array}$  es correcto

Ahora que tenemos la regla de la consecuencia y que hemos visto que el axioma de la asignación por sí solo no sirve, los pasos que hay que dar para decidir si una asignación es correcta con respecto a una precondición  $\phi$  y una poscondición  $\psi$  son los siguientes:

- Calcular  $\{\text{def}(t) \wedge \psi_x^t\}$ , al que llamaremos  $\phi_1$ .
- Comprobar que se cumple  $\phi \rightarrow \phi_1$ .

### 3.2.2.6. Ejemplo 3 (continuación)

$$\{\phi\} \equiv \{z \neq 0 \wedge y = 0\}$$

$$\{\phi_1\} \equiv \{z \neq 0 \wedge y / z \neq 1\}$$

$$\text{¿} \phi \rightarrow \phi_1 \text{?}$$

$$\underbrace{z \neq 0}_{\alpha} \wedge \underbrace{y = 0}_{\beta} \rightarrow \underbrace{z \neq 0}_{\text{sí por } \alpha} \wedge \underbrace{y / z \neq 1}_{\text{sí por } \alpha \text{ y } \beta} \text{?}$$

Es decir, si  $z \neq 0 \wedge y = 0$  es cierto, entonces  $z \neq 0 \wedge y / z \neq 1$  es cierto?

La respuesta es que sí y por tanto el programa del ejemplo 3 es correcto. Mediante letras griegas se indica qué partes de la izquierda hacen que la parte derecha sea cierta. En este caso  $y / z \neq 1$  es cierto porque  $z \neq 0$  y porque  $y = 0$ .

- **Demostración formal:**

1.  $\phi \rightarrow \phi_1$
2.  $\{\phi_1\} x := y / z; \{\psi\}$  (AA)
3.  $\{\phi\} x := y / z; \{\psi\}$  (RCN 1, 2)

Mediante esa demostración formal estamos indicando que la asignación es correcta por la regla de la consecuencia (3) ya que previamente hemos probado que  $\phi$  implica  $\phi_1$  (punto 1) y que la asignación es correcta con respecto a  $\phi_1$  y  $\psi$  (punto 2).

El camino que se ha seguido para demostrar el programa ha sido el siguiente:

- Calcular  $\{\text{def}(t) \wedge \psi_x^t\}$  a partir de la asignación  $x := y / z$ ; y la poscondición  $\{\psi\}$ .
- Comprobar si  $\{\phi\}$  y  $\{\text{def}(t) \wedge \psi_x^t\}$  son iguales.
- Como no son iguales, comprobar si  $\phi \rightarrow (\text{def}(t) \wedge \psi_x^t)$ .
- Como la implicación ha resultado ser cierta, se ha dado la demostración formal.



### 3.2.2.7. Ejemplo 4

¿Es correcto el siguiente programa?

$$\begin{aligned} \{\varphi\} &\equiv \{1 \leq i \leq n \wedge \text{posit}(A(1..n))\} \\ x &:= A(i); \\ \{\psi\} &\equiv \{x > 0\} \end{aligned}$$

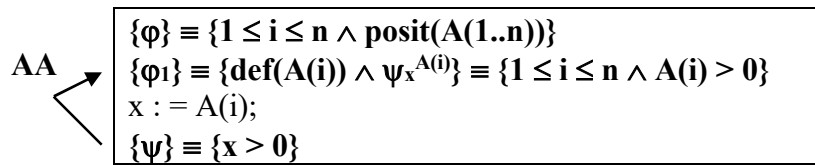
Donde el predicado  $\text{posit}(A(1..n))$  se define como sigue:

$$\text{posit}(A(1..n)) \equiv \forall k (1 \leq k \leq n \rightarrow A(k) > 0)$$

Por tanto  $\text{posit}(A(1..n))$  indica que todos los elementos de la tabla  $A(1..n)$  son positivos ( $> 0$ ).

Tenemos que hacer dos cosas:

- Calcular  $\{\text{def}(t) \wedge \psi_x^t\}$  donde  $t$  es  $A(i)$
- Comprobar si se cumple que  $\{\varphi\}$  y  $\{\text{def}(t) \wedge \psi_x^t\}$  son iguales y en caso de que no lo sean comprobar  $\varphi \rightarrow (\text{def}(t) \wedge \psi_x^t)$ .

AA 

$$\begin{aligned} \{\varphi\} &\equiv \{1 \leq i \leq n \wedge \text{posit}(A(1..n))\} \\ \{\varphi_1\} &\equiv \{\text{def}(A(i)) \wedge \psi_x^{A(i)}\} \equiv \{1 \leq i \leq n \wedge A(i) > 0\} \\ x &:= A(i); \\ \{\psi\} &\equiv \{x > 0\} \end{aligned}$$

En este caso  $\text{def}(A(i))$  es  $1 \leq i \leq n$  porque para poder calcular  $A(i)$  u obtener el valor de  $A(i)$  tiene que ocurrir que el índice  $i$  esté en el rango del vector  $A(1..n)$ , es decir, tiene que ocurrir que  $i$  tenga un valor comprendido entre 1 y  $n$ .

Como  $\varphi$  y  $\varphi_1$  no son iguales, hay que comprobar si se cumple la implicación  $\varphi \rightarrow \varphi_1$ .

¿ $\varphi \rightarrow \varphi_1$ ?

$$\underbrace{1 \leq i \leq n}_{\alpha} \wedge \underbrace{\text{posit}(A(1..n))}_{\beta} \rightarrow \underbrace{1 \leq i \leq n}_{\text{por } \alpha} \wedge \underbrace{A(i) > 0}_{\text{por } \alpha \text{ y } \beta}$$

Es decir, si  $1 \leq i \leq n \wedge \text{posit}(A(1..n))$  es cierto, ¿ $1 \leq i \leq n \wedge A(i) > 0$  es cierto?

En este caso la respuesta es que sí y mediante letras griegas se indica qué partes de la izquierda hacen que las distintas partes de la derecha sean ciertas. Así, por ejemplo,  $A(i) > 0$  es cierto porque estamos considerando que  $\text{posit}(A(1..n))$  es cierto.

Como la implicación se cumple el programa del ejemplo 4 es correcto.

• **Demostración formal:**

1.  $\varphi \rightarrow \varphi_1$
2.  $\{\varphi_1\} x := A(i); \{\psi\}$  (AA)
3.  $\{\varphi\} x := A(i); \{\psi\}$  (RCN 1, 2)

Mediante esa demostración formal estamos indicando que la asignación es correcta por la regla de la consecuencia (punto 3) ya que previamente hemos comprobado que  $\varphi$  implica  $\varphi_1$  (punto 1) y que la asignación es correcta con respecto a  $\varphi_1$  y  $\psi$  (punto 2).

El camino que se ha seguido para demostrar el programa ha sido el siguiente:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  a partir de la asignación  $x := A(i)$ ; y la poscondición  $\{\psi\}$ .
- Comprobar si  $\{\varphi\}$  y  $\{\varphi_1\}$  son iguales.
- Como no son iguales, comprobar si  $\varphi \rightarrow \varphi_1$ .
- Como la implicación ha resultado ser cierta, se ha dado la demostración formal.

### 3.2.2.8. Ejemplo 5

¿Es correcto el siguiente programa?

$$\begin{aligned} \{\varphi\} &\equiv \{1 \leq i < n\} \\ i &:= i + 1; \\ \{\psi\} &\equiv \{1 \leq i \leq n\} \end{aligned}$$

Tenemos que hacer dos cosas:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  donde  $t$  es  $i + 1$  y en vez de  $x$  tenemos  $i$ .
- Comprobar si se cumple que  $\varphi$  y  $\varphi_1$  son iguales y en caso de que no lo sean comprobar  $\varphi \rightarrow \varphi_1$ .

AA  $\rightarrow$  
$$\begin{aligned} \{\varphi\} &\equiv \{1 \leq i \leq n - 1\} \\ \{\varphi_1\} &\equiv \{\text{def}(i + 1) \wedge \psi_i^{i+1}\} \equiv \{\text{true} \wedge 1 \leq i + 1 \leq n\} \equiv_{\text{simplificación}} \{0 \leq i \leq n - 1\} \\ i &:= i + 1; \\ \{\psi\} &\equiv \{1 \leq i \leq n\} \end{aligned}$$

En este caso  $\text{def}(i + 1)$  es *true* porque  $i + 1$  se puede calcular siempre, no se tiene que cumplir ninguna condición para que se pueda calcular  $i + 1$ .

Una vez calculado  $\{\text{def}(i + 1) \wedge \psi_i^{i+1}\}$  se ha procedido a simplificar la fórmula para que sea más entendible. Para ello se ha tenido en cuenta que para cualquier fórmula  $\delta$  se cumple que  $\delta \equiv \text{true} \wedge \delta$ . Por otra parte la expresión  $1 \leq i + 1 \leq n$  se entiende mejor si la expresamos en función de  $i$  en vez de expresarla en función de  $i + 1$ . Para conseguir expresar  $1 \leq i + 1 \leq n$  en función de  $i$ , hay que restar 1 a sus tres componentes  $1 - 1 \leq i + 1 - 1 \leq n - 1$  y tras realizar las restas queda  $0 \leq i \leq n - 1$ .

Como  $\varphi$  y  $\varphi_1$  no son iguales, hay que comprobar si se cumple la implicación  $\varphi \rightarrow \varphi_1$ .

$$\begin{array}{ccc} \text{¿} \varphi \rightarrow \varphi_1? & & \\ \text{¿} \underbrace{(1 \leq i \leq n-1)}_{\alpha} \rightarrow \underbrace{(0 \leq i \leq n-1)}_{\text{sí por } \alpha} ? & & \end{array}$$

Es decir, si  $1 \leq i \leq n-1$  es cierto ¿es cierto  $0 \leq i \leq n-1$ ?

La respuesta es que sí y mediante letras griegas se indica qué partes de la izquierda hacen que las distintas partes de la derecha sean ciertas. Por ejemplo,  $0 \leq i \leq n-1$  es cierto porque estamos considerando que  $1 \leq i \leq n-1$  es cierto.

Como la implicación se cumple el programa del ejemplo 5 es correcto.

- **Demostración formal:**

1.  $\varphi \rightarrow \varphi_1$
2.  $\{\varphi_1\} i := i + 1; \{\psi\}$  (AA)
3.  $\{\varphi\} i := i + 1; \{\psi\}$  (RCN 1, 2)

Mediante esa demostración formal estamos indicando que la asignación es correcta por la regla de la consecuencia (punto 3) ya que previamente hemos probado que  $\varphi$  implica  $\varphi_1$  (punto 1) y que la asignación es correcta con respecto a  $\varphi_1$  y  $\psi$  (punto 2).

El camino que se ha seguido para demostrar el programa ha sido el siguiente:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  a partir de la asignación  $i := i + 1$ ; y la poscondición  $\{\psi\}$ .
- Comprobar si  $\{\varphi\}$  y  $\{\varphi_1\}$  son iguales.
- Como no son iguales, comprobar si  $\varphi \rightarrow \varphi_1$ .
- Como la implicación ha resultado ser cierta, se ha dado la demostración formal.


### 3.2.2.9. Ejemplo 6

¿Es correcto el siguiente programa?

$$\begin{array}{l} \{\varphi\} \equiv \{z = 1\} \\ y := 0; \\ \{\psi\} \equiv \{z = x^y \wedge y \geq 0\} \end{array}$$

Tenemos que hacer dos cosas:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  donde  $t$  es 0
- Comprobar si se cumple que  $\varphi$  y  $\varphi_1$  son iguales y en caso de que no lo sean comprobar si se cumple  $\varphi \rightarrow \varphi_1$ .

AA 

$$\begin{array}{l} \{\varphi\} \equiv \{z = 1\} \\ \{\varphi_1\} \equiv \{\text{def}(0) \wedge \psi_y^0\} \equiv \{\text{true} \wedge z = x^0 \wedge 0 \geq 0\} \\ y := 0; \\ \{\psi\} \equiv \{z = x^y \wedge y \geq 0\} \end{array}$$

En este caso  $\text{def}(0)$  es *true* porque 0 se puede calcular siempre, no se tiene que cumplir ninguna condición para que se pueda calcular 0.

Como  $\varphi$  y  $\varphi_1$  no son iguales, hay que comprobar si se cumple la implicación  $\varphi \rightarrow \varphi_1$ .

Primero vamos a simplificar  $\varphi_1$ .

$$\{\varphi_1\} \equiv \{\text{true} \wedge z = x^0 \wedge 0 \geq 0\} \equiv_{\text{simplificación}} \{z = x^0\}$$

Para llevar acabo la simplificación por una parte se ha tenido en cuenta que para cualquier fórmula  $\delta$  se cumple que  $\delta \equiv \text{true} \wedge \delta$ . Por otra parte  $0 \geq 0$  es siempre cierto (o *true*) y se puede quitar ya que  $\delta \equiv \delta \wedge \text{true}$  para cualquier fórmula  $\delta$ . Por último es importante darse cuenta de que  $x^0$  no puede ser simplificado a 1 ya que existe un caso,  $0^0$ , que no es 1, sino una indeterminación que genera un error en la ejecución de programas.

Ahora vamos a comprobar si se cumple la implicación:

¿ $\varphi \rightarrow \varphi_1$ ?

¿ $z = 1 \rightarrow z = x^0$ ?

Es decir, si  $z = 1$  es cierto, ¿ $z = x^0$  es cierto?

Siempre que  $z$  sea 1 ¿ $x$  elevado a 0 será igual a  $x$ ? La respuesta es que no. Hay un caso en el que eso no se cumple y es cuando  $x$  es 0 ya que  $0^0$  es una indeterminación.

Como la implicación no se cumple el programa no es correcto. Cuando un programa no es correcto en vez de dar la demostración formal hay que dar un contraejemplo.

- **Contraejemplo:**

Cuando  $x$  es 0, se cumple la precondition pero al ejecutar la asignación no se cumple la poscondición.

### 3.2.2.10. Ejemplo 7

¿Es correcto el siguiente programa?

$$\begin{array}{l} \{\varphi\} \equiv \{x \neq 0 \wedge z = 1\} \\ y := 0; \\ \{\psi\} \equiv \{z = x^y \wedge y \geq 0\} \end{array}$$

Tenemos que hacer dos cosas:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  donde  $t$  es 0
- Comprobar si se cumple que  $\varphi$  y  $\varphi_1$  son iguales y en caso de que no lo sean comprobar si se cumple  $\varphi \rightarrow \varphi_1$ .

AA  $\swarrow$

$$\begin{array}{l} \{\varphi\} \equiv \{x \neq 0 \wedge z = 1\} \\ \{\varphi_1\} \equiv \{\text{def}(0) \wedge \psi_y^0\} \equiv \{\text{true} \wedge z = x^0 \wedge 0 \geq 0\} \\ y := 0; \\ \{\psi\} \equiv \{z = x^y \wedge y \geq 0\} \end{array}$$

En este caso  $\text{def}(0)$  es *true* porque 0 se puede calcular siempre, no se tiene que cumplir ninguna condición para que se pueda calcular 0.

Como  $\varphi$  y  $\varphi_1$  no son iguales, hay que comprobar si se cumple la implicación  $\varphi \rightarrow \varphi_1$ .

Primero vamos a simplificar  $\varphi_1$ .

$$\{\varphi_1\} \equiv \{\text{true} \wedge z = x^0 \wedge 0 \geq 0\} \equiv_{\text{simplificación}} \{z = x^0\}$$

Para llevar acabo la simplificación por una parte se ha tenido en cuenta que para cualquier fórmula  $\delta$  se cumple que  $\delta \equiv \text{true} \wedge \delta$ . Por otra parte  $0 \geq 0$  es siempre cierto (o *true*) y se puede quitar ya que  $\delta \equiv \delta \wedge \text{true}$  para cualquier fórmula  $\delta$ . Por último es importante darse cuenta de que  $x^0$  no puede ser simplificado a 1 ya que existe un caso,  $0^0$ , que no es 1, sino una indeterminación que genera un error en la ejecución de programas.

¿ $\varphi \rightarrow \varphi_1$ ?

$$\underbrace{x \neq 0}_{\alpha} \wedge \underbrace{z = 1}_{\beta} \rightarrow \underbrace{z = x^0}_{\text{sí por } \alpha, \beta}$$

Es decir, si  $x$  no es 0 y  $z$  es 1, ¿ $z = x^0$  es cierto?

Sabiendo que se cumple  $x \neq 0 \wedge z = 1$ , también es cierto  $z = x^0$  ya que si  $x$  es distinto de 0,  $x^0$  será 1. Por tanto la respuesta es que sí y mediante letras griegas se indica qué partes de la izquierda hacen que  $z = x^0$  sea cierta.

Como la implicación se cumple el programa del ejemplo 7 es correcto y hay que dar la demostración formal.

- **Demostración formal:**

1.  $\phi \rightarrow \phi_1$
2.  $\{\phi_1\} y := 0; \{\psi\}$  (AA)
3.  $\{\phi\} y := 0; \{\psi\}$  (RCN 1, 2)

Mediante esa demostración formal estamos indicando que la asignación es correcta por la regla de la consecuencia (3) ya que previamente hemos probado que  $\phi$  implica  $\phi_1$  (1) y que la asignación es correcta con respecto a  $\phi_1$  y  $\psi$  (2).

El camino que se ha seguido para demostrar el programa ha sido el siguiente:

- Calcular  $\{\phi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  a partir de la asignación  $y := 0$ ; y la poscondición  $\{\psi\}$ .
- Comprobar si  $\{\phi\}$  y  $\{\phi_1\}$  son iguales.
- Como no son iguales, comprobar si  $\phi \rightarrow \phi_1$ .
- Como la implicación ha resultado ser cierta, se ha dado la demostración formal.

### 3.2.2.11. Ejemplo 8

¿Es correcto el siguiente programa?

$\{\phi\} \equiv \{b \wedge \text{posit}(A(1..i)) \wedge 1 \leq i \leq n - 1 \wedge A(i + 1) < 0\}$   
 $b := \text{false};$   
 $\{\psi\} \equiv \{b \leftrightarrow \text{Posit}(A(1..i + 1))\}$

donde  $\text{posit}(A(1..r)) \equiv \forall k (1 \leq k \leq r \rightarrow A(k) > 0)$

Tenemos que hacer dos cosas:

- Calcular  $\{\phi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  donde  $t$  es la constante booleana *false*
- Comprobar si se cumple que  $\phi$  y  $\phi_1$  son iguales y en caso de que no lo sean comprobar si se cumple  $\phi \rightarrow \phi_1$ .

AA  $\swarrow$

$\{\phi\} \equiv \{b \wedge \text{posit}(A(1..i)) \wedge 1 \leq i \leq n - 1 \wedge A(i + 1) < 0\}$   
 $\{\phi_1\} \equiv \{\text{def}(\text{false}) \wedge \psi_b^{\text{false}}\} \equiv \{\text{true} \wedge (\text{false} \leftrightarrow \text{posit}(A(1..i + 1)))\}$   
 $b := \text{false};$   
 $\{\psi\} \equiv \{b \leftrightarrow \text{posit}(A(1..i + 1))\}$

En este caso  $\text{def}(\text{false})$  es *true* porque *false* es una constante y se puede calcular siempre, no se tiene que cumplir ninguna condición para que se pueda calcular *false*.

Como  $\phi$  y  $\phi_1$  no son iguales, hay que comprobar si se cumple la implicación  $\phi \rightarrow \phi_1$ . Primero vamos a simplificar  $\phi_1$ .

$$\{\phi_1\} \equiv \{\text{true} \wedge (\text{false} \leftrightarrow \text{posit}(A(1..i + 1)))\} \equiv_{\text{simplificación}} \{\neg \text{posit}(A(1..i + 1))\}$$

Para llevar acabo la simplificación por una parte se ha tenido en cuenta que para cualquier fórmula  $\delta$  se cumple que  $\delta \equiv \text{true} \wedge \delta$ . Por otra parte para cualquier fórmula  $\delta$  se cumple que  $\neg\delta \equiv \text{false} \leftrightarrow \delta$ .

$$\begin{array}{c} \text{¿}\varphi \rightarrow \varphi_1? \\ \text{¿}b \wedge \text{posit}(A(1..i)) \wedge \underbrace{1 \leq i \leq n-1}_{\alpha} \wedge \underbrace{A(i+1) < 0}_{\beta} \rightarrow \underbrace{\neg\text{posit}(A(1..i+1))}_{\text{sí por } \alpha \text{ y } \beta} \end{array}$$

Es decir, si  $b \wedge \text{posit}(A(1..i)) \wedge 1 \leq i \leq n-1 \wedge A(i+1) < 0$  es cierto, ¿se cumple  $\neg\text{posit}(A(1..i+1))$ ?

Por  $\alpha$  sabemos que  $i+1$  está en el rango de  $A(1..n)$ . Por  $\beta$  sabemos que en la posición  $i+1$  del vector  $A(1..n)$  tenemos un valor negativo. Por tanto,  $\text{posit}(A(1..i+1))$  es false porque no todos los elementos de  $A(1..i+1)$  son positivos. Como consecuencia de ello tenemos que  $\neg\text{posit}(A(1..i+1))$  es cierto:

Mediante letras griegas se indica qué partes de la izquierda hacen que la parte derecha sea cierta.

Como la implicación se cumple el programa del ejemplo 8 es correcto y hay que dar la demostración formal.

- **Demostración formal:**

1.  $\varphi \rightarrow \varphi_1$
2.  $\{\varphi_1\} b := \text{false}; \{\psi\}$  (AA)
3.  $\{\varphi\} b := \text{false}; \{\psi\}$  (RCN 1, 2)

Mediante esa demostración formal estamos indicando que la asignación es correcta por la regla de la consecuencia (3) ya que previamente hemos probado que  $\varphi$  implica  $\varphi_1$  (1) y que la asignación es correcta con respecto a  $\varphi_1$  y  $\psi$  (2).

El camino que se ha seguido para demostrar el programa ha sido el siguiente:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(t) \wedge \psi_x^t\}$  a partir de la asignación  $b := \text{false};$  y la poscondición  $\{\psi\}$ .
- Comprobar si  $\{\varphi\}$  y  $\{\varphi_1\}$  son iguales.
- Como no son iguales, comprobar si  $\varphi \rightarrow \varphi_1$ .
- Como la implicación ha resultado ser cierta, se ha dado la demostración formal.

### 3.2.3. Regla de la Composición (RCP)

Hasta ahora hemos visto cómo verificar programas que tienen únicamente una instrucción, y concretamente una asignación. A continuación vamos a analizar cómo se verifican programas con más de una instrucción. Los primeros ejemplos tendrán sólo asignaciones.

Regla de la composición (RCP)	
Composición de dos subprogramas o instrucciones	
$\{\varphi\} P_1$	$\{\varphi_1\} P_2 \{\psi\}$
$\{\varphi\} P_1 P_2 \{\psi\}$	
Esta regla viene a decir que si $\{\varphi\} P_1 \{\varphi_1\}$ es correcto y $\{\varphi_1\} P_2 \{\psi\}$ es correcto, entonces $\{\varphi\} P_1 P_2 \{\psi\}$ es correcto.	

Dicho de otra forma:

Si	$\{\varphi\}$ $P_1$ $\{\varphi_1\}$	es correcto y	$\{\varphi_1\}$ $P_2$ $\{\psi\}$	es correcto, entonces	$\{\varphi\}$ $P_1$ $P_2$ $\{\psi\}$	es correcto
----	---	---------------	--	-----------------------	---	-------------

Es importante darse cuenta de que la postcondición del primer programa ha de ser la precondición del segundo. En este caso  $\{\varphi_1\}$ .

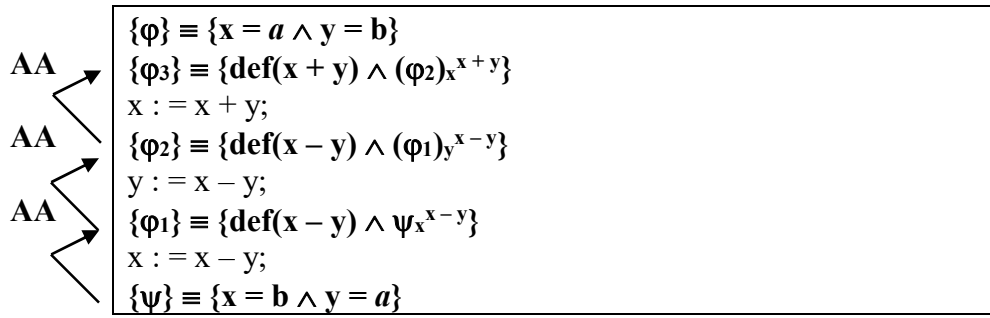
#### 3.2.3.1. Ejemplo 9

¿Es correcto el siguiente programa?

$P_1$	$\{\varphi\} \equiv \{x = a \wedge y = b\}$
$P_2$	$x := x + y;$
$P_3$	$y := x - y;$
	$x := x - y;$
	$\{\psi\} \equiv \{x = b \wedge y = a\}$

Se calcularán tres aserciones intermedias utilizando el axioma de la asignación.





Tenemos que dar los siguientes pasos:

- Calcular  $\{\varphi_1\} \equiv \{\text{def}(x - y) \wedge \psi^{x-y}\}$
- Calcular  $\{\varphi_2\} \equiv \{\text{def}(x - y) \wedge (\varphi_1)^{x-y}\}$
- Calcular  $\{\varphi_3\} \equiv \{\text{def}(x + y) \wedge (\varphi_2)^{x+y}\}$
- Comprobar si se cumple que  $\varphi$  y  $\varphi_3$  son iguales y en caso de que no lo sean comprobar si se cumple  $\varphi \rightarrow \varphi_3$ . Si se cumple la implicación, el programa será correcto y si no, no será correcto.
- Si el programa es correcto habrá que dar la prueba de la corrección y si no es correcto, habrá que dar un ejemplo que muestre que el programa no es correcto.

Es importante darse cuenta de que los cálculos se hacen partiendo desde abajo, desde la postcondición y se va subiendo hacia arriba en el programa.

A continuación se darán las tres aserciones. Al calcular las aserciones se simplificarán las apariciones de true, false y repeticiones de fórmulas:

- $\{\varphi_1\} \equiv \{\text{def}(x - y) \wedge \psi^{x-y}\} \equiv \{\text{true} \wedge x - y = b \wedge y = a\} \equiv_{\text{simplificación}} \{x - y = b \wedge y = a\}$   
 $\{\varphi_1\} \equiv \{x - y = b \wedge y = a\}$
- $\{\varphi_2\} \equiv \{\text{def}(x - y) \wedge (\varphi_1)^{x-y}\} \equiv \{\text{true} \wedge x - (x - y) = b \wedge x - y = a\} \equiv_{\text{simplificación}} \{y = b \wedge x - y = a\}$   
 $\{\varphi_2\} \equiv \{y = b \wedge x - y = a\}$
- $\{\varphi_3\} \equiv \{\text{def}(x + y) \wedge (\varphi_2)^{x+y}\} \equiv \{\text{true} \wedge y = b \wedge x + y - y = a\} \equiv_{\text{simplificación}} \{y = b \wedge x = a\}$   
 $\{\varphi_3\} \equiv \{y = b \wedge x = a\}$

Las expresiones  $\text{def}(x - y)$  y  $\text{def}(x + y)$  son *true* porque la suma y la resta siempre se pueden calcular para los números enteros, no es necesario que se cumpla ninguna condición.

Una vez que hemos llegado arriba, es decir, una vez que hemos calculado  $\varphi_3$ , como  $\varphi$  y  $\varphi_3$  no son iguales tenemos que comprobar si  $\varphi$  implica  $\varphi_3$ :

$$\begin{array}{ccccccc} & & \varphi \rightarrow \varphi_3? & & & & \\ & \underbrace{x = a}_{\alpha} \wedge & \underbrace{y = b}_{\beta} \rightarrow & \underbrace{y = b}_{\text{por } \beta} \wedge & \underbrace{x = a}_{\text{por } \alpha} & & \\ & \alpha & & \beta & & & \end{array}$$

Es decir, si  $x = a \wedge y = b$  es cierto, ¿se cumple  $y = b \wedge x = a$ ? La respuesta es que sí porque se tiene lo mismo aunque en distinto orden y para cualquier par de fórmulas  $\delta$  y  $\gamma$  se cumple  $\delta \wedge \gamma \equiv \gamma \wedge \delta$ .

Mediante las letras griegas  $\alpha$  y  $\beta$  se indica qué partes de la izquierda hacen que las distintas partes de la derecha sean ciertas.

Como la implicación se cumple el programa del ejemplo 9 es correcto y hay que dar la demostración formal.

- **Demostración formal:**

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} 1. \varphi \rightarrow \varphi_3 \\ 2. \{\varphi_3\} x := x + y; \{\varphi_2\} \text{ (AA)} \\ 3. \{\varphi\} x := x + y; \{\varphi_2\} \text{ (RCN 1, 2)} \\ 4. \{\varphi_2\} y := x - y; \{\varphi_1\} \text{ (AA)} \end{array} \right. \\ 5. \{\varphi\} \\ \quad x := x + y; \\ \quad y := x - y; \\ \quad \{\varphi_1\} \text{ (RCP 3, 4)} \\ 6. \{\varphi_1\} x := x - y; \{\psi\} \text{ (AA)} \\ 7. \{\varphi\} \\ \quad x := x + y; \\ \quad y := x - y; \\ \quad x := x - y; \\ \quad \{\psi\} \text{ (RCP 5, 6)} \end{array} \right.$$

Mediante esa demostración formal estamos indicando que el programa es correcto. Para ello nos basamos básicamente en implicaciones y asignaciones que luego se juntan mediante la regla de la consecuencia y la regla de la composición. En el último paso se tiene el programa entero, y en los pasos previos se tienen trozos del programa y se indica por qué son correctos esos trozos. En el punto 7 se indica que el programa entero es correcto por la regla de la composición y porque los programas de los puntos 5 y 6 son correctos. El programa del punto 6 es correcto por el axioma de la asignación. El programa del punto 5 es correcto por la regla de la composición y porque los programas de los puntos 3 y 4 son correctos. El programa del punto 4 es correcto por el axioma de la asignación. El programa del punto 3 es correcto por la regla de la consecuencia y porque tenemos la implicación del punto 1 y el programa correcto del punto 2. el programa del punto 2 es correcto por el axioma de la asignación.

El camino que se ha seguido para demostrar el programa ha sido el siguiente:

- Calcular  $\varphi_1$ ,  $\varphi_2$  y  $\varphi_3$  partiendo desde la poscondición  $\{\psi\}$  y teniendo en cuenta las asignaciones.
- Comprobar si  $\{\varphi\}$  y  $\{\varphi_3\}$  son iguales.
- Como no son iguales, comprobar si  $\varphi \rightarrow \varphi_3$ .
- Como la implicación  $\varphi \rightarrow \varphi_3$  se cumple, el programa es correcto y se ha dado la demostración formal.

### 3.2.4. Regla del While (RWH)

A continuación vamos a estudiar cómo se verifica la corrección de la instrucción **While**.

Un programa while tendrá la siguiente forma:

$\begin{array}{l} \{\varphi\} \\ \text{while } \{\text{INV}\} \text{ B loop} \\ \quad \text{Instrucciones} \\ \text{end loop} \\ \{\psi\} \end{array}$	<p>donde</p> <p><math>\varphi</math> es la precondition, <math>\psi</math> es la poscondition,          INV es el invariante,          B es la condición del while e          Inst son las instrucciones de dentro del while</p>
--	--

Un programa while de la forma

$\begin{array}{l} \{\varphi\} \\ \text{while } \{\text{INV}\} \text{ B loop} \\ \quad \text{Instrucciones} \\ \text{end loop} \\ \{\psi\} \end{array}$
--

es **parcialmente correcto** si:

- I. Se cumple  $\varphi \rightarrow \text{INV}$
- II. Se cumple  $\text{INV} \rightarrow \text{def}(B)$
- III. El programa

$\begin{array}{l} \{\text{INV} \wedge B\} \\ \text{Instrucciones} \\ \{\text{INV}\} \end{array}$
--

es correcto

- IV. Se cumple  $(\text{INV} \wedge \neg B) \rightarrow \psi$

Para que el programa while sea **totalmente correcto**, además ha de verificar los siguientes dos puntos.

- V. Se cumple  $(\text{INV} \wedge B) \rightarrow E > 0$
- VI. El programa

$\begin{array}{l} \{\text{INV} \wedge B \wedge E = v\} \\ \text{Instrucciones} \\ \{E < v\} \end{array}$
--

es correcto,

siendo  $v$  es una variable que no aparece en el programa.

Con los primeros cuatro puntos se prueba que en caso de que el while termine, terminará cumpliendo la poscondición y por tanto será correcto, pero no se prueba que el while termine.

Con los puntos V y VI se prueba que el while terminará.

Por tanto si se consigue probar los seis puntos, se habrá probado que el while siempre termina y que termina cumpliendo la poscondición. En ese caso diríamos que el while es totalmente correcto.

Pero es posible que a veces sólo se puedan probar los primeros cuatro puntos y que alguno de los dos últimos (V ó VI) no se cumpla. En ese caso diríamos que el while es parcialmente correcto.

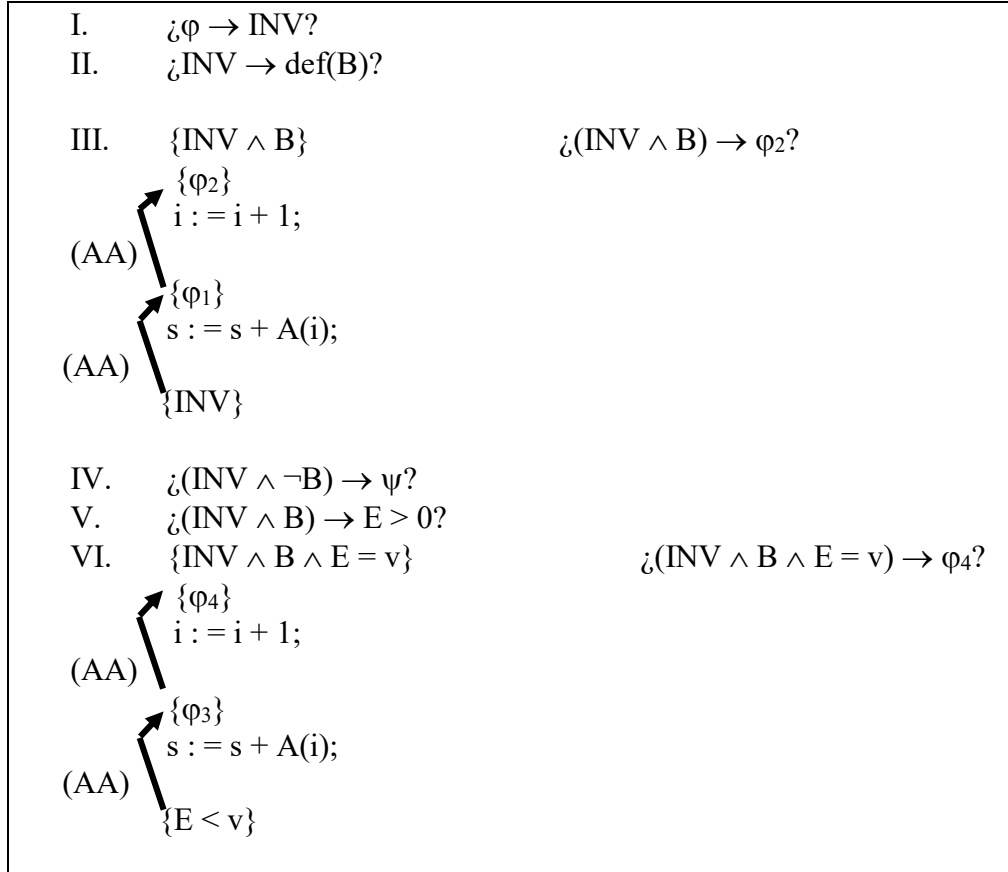
### 3.2.4.1. Ejemplo 10

¿Es correcto el siguiente programa?

$\{\varphi\} \equiv \{i = 0 \wedge s = 0 \wedge n \geq 1\}$ <b><u>while</u></b> {INV} $i \neq n$ <b><u>loop</u></b> $i := i + 1;$ $s := s + A(i);$ <b><u>end loop</u></b> ; $\{\psi\} \equiv \{s = \sum_{k=1}^n A(k)\}$
$\{INV\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)\}$  $E = n - i$

#### **Esquema:**

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones



I.  $\zeta \varphi \rightarrow \text{INV}?$

$$\zeta i = 0 \wedge s = 0 \wedge n \geq 0 \rightarrow (0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)?$$

Es decir, suponiendo que  $i = 0 \wedge s = 0 \wedge n \geq 0$  es cierto, hay mirar si  $(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)$  es cierto.

$$\underbrace{i = 0}_{\alpha} \wedge \underbrace{s = 0}_{\beta} \wedge \underbrace{n \geq 1}_{\gamma} \rightarrow (0 \leq i \leq n) \wedge s = \underbrace{\sum_{k=1}^i A(k)}_{\text{por } \alpha \text{ y } \beta}$$

Sí se cumple, ya que cada componente de la segunda parte de la implicación se justifica mediante las componentes de la primera parte tal como se indica mediante las letras griegas  $\alpha$ ,  $\beta$  y  $\gamma$ .

II.  $\zeta \text{INV} \rightarrow \text{def}(\text{B})?$

$\zeta \text{INV} \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.

## III.

$$\begin{aligned}
\bullet \quad \{\varphi_1\} &\equiv \{\text{def}(s + A(i)) \wedge (\text{INV})_s^{s+A(i)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge (0 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k)\} \equiv \text{simplificación} \\
&\equiv \{(1 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k)\}
\end{aligned}$$

Se puede quitar  $(0 \leq i \leq n)$  y dejar sólo  $(1 \leq i \leq n)$  porque si se cumple  $(1 \leq i \leq n)$  también se cumple  $(0 \leq i \leq n)$ . Pero no se podría quitar  $(1 \leq i \leq n)$  y dejar  $(0 \leq i \leq n)$  porque el que se cumpla  $(0 \leq i \leq n)$  no quiere decir que se cumpla  $(1 \leq i \leq n)$ .

$$\begin{aligned}
\bullet \quad \{\varphi_2\} &\equiv \{\text{def}(i + 1) \wedge (\varphi_1)_{i+1}^{i+1}\} \equiv \\
&\equiv \{\text{true} \wedge (1 \leq i + 1 \leq n) \wedge s + A(i + 1) = \sum_{k=1}^{i+1} A(k)\} \equiv \text{simplificación} \\
&\equiv \{(0 \leq i \leq n - 1) \wedge s + A(i + 1) = \sum_{k=1}^{i+1} A(k)\}
\end{aligned}$$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $(1 \leq i + 1 \leq n)$ , lo mejor es ponerlo utilizando  $i$  en vez de  $i + 1$ . Para ello hay que restar 1 a los tres elementos:  $(1 - 1 \leq i + 1 - 1 \leq n - 1)$ . Tras realizar las operaciones queda  $(0 \leq i \leq n - 1)$ .

$$\bullet \quad \text{¿}(\text{INV} \wedge B) \rightarrow \varphi_2\text{?}$$

$$\begin{array}{ccc}
(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k) \wedge i \neq n & & \\
\begin{array}{ccc} \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} \\ \alpha & \beta & \delta \end{array} & & \\
\downarrow ? & & \\
(0 \leq i \leq n - 1) \wedge s + A(i + 1) = \sum_{k=1}^{i+1} A(k) & & \\
\begin{array}{ccc} \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} & \\ \text{por } \alpha \text{ y } \delta & \text{por } \alpha, \beta \text{ y } \delta & \end{array} & & 
\end{array}$$

En la segunda parte de abajo, por  $\alpha$  y  $\delta$  sabemos que con  $i + 1$  no nos salimos del rango de  $A(1..n)$  y luego por  $\beta$  se termina de deducir " $s + A(i + 1) =$

$$\sum_{k=1}^{i+1} A(k) "$$

IV. ¿ $(INV \wedge \neg B) \rightarrow \psi$ ?

$$\begin{array}{c}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge \underbrace{s = \sum_{k=1}^i A(k)}_{\beta} \wedge \underbrace{i = n}_{\delta} \\
 \downarrow ? \\
 \underbrace{\{s = \sum_{k=1}^n A(k)\}}_{\text{por } \beta \text{ y } \delta}
 \end{array}$$

En este caso al ser  $i = n$ , sustituyendo  $i$  por  $n$  en  $\beta$  tenemos lo mismo que en  $\psi$ .

V. ¿ $(INV \wedge B) \rightarrow E > 0$ ?

$$\begin{array}{c}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge \underbrace{s = \sum_{k=1}^i A(k)}_{\beta} \wedge \underbrace{i \neq n}_{\beta} \\
 \downarrow ? \\
 \underbrace{n - i > 0}_{\text{por } \alpha \text{ y } \beta}
 \end{array}$$

Por  $\alpha$  sabemos que  $n \geq i$  y por tanto  $n - i \geq 0$  ya que si se cumple  $n \geq i$ , también se cumple  $n - i \geq i - i$  y eso es lo mismo que  $n - i \geq 0$ . Además por  $\beta$  sabemos que  $i \neq n$ . Por tanto  $n - i > 0$ .

VI.

- $\{\varphi_3\} \equiv \{\text{def}(s + A(i)) \wedge (E < v)_{s+A(i)}\} \equiv$   
 $\equiv \{(1 \leq i \leq n) \wedge n - i < v\} \equiv$
- $\{\varphi_4\} \equiv \{\text{def}(i + 1) \wedge (\varphi_3)_{i+1}^{i+1}\} \equiv$   
 $\equiv \{\text{true} \wedge (1 \leq i + 1 \leq n) \wedge n - (i + 1) < v\} \equiv \text{simplificación}$   
 $\equiv \{(0 \leq i \leq n - 1) \wedge n - i - 1 < v\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $(1 \leq i + 1 \leq n)$ , lo mejor es ponerlo utilizando  $i$  en vez de  $i + 1$ . Para ello hay que restar el valor 1 a los tres elementos:  $(1 - 1 \leq i + 1 - 1 \leq n - 1)$ . Tras realizar las operaciones queda  $(0 \leq i \leq n - 1)$ .



- ¿ $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$ ?

$$\begin{array}{c}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge s = \sum_{k=1}^i A(k) \wedge \underbrace{i \neq n}_{\beta} \wedge \underbrace{n - i = v}_{\delta} \\
 \downarrow ? \\
 \underbrace{(0 \leq i \leq n - 1)}_{\text{por } \alpha \text{ y } \beta} \wedge \underbrace{n - i - 1 < v}_{\text{por } \delta}
 \end{array}$$

En la primera parte de abajo, por  $\alpha$  y  $\beta$  sabemos que se cumplirá  $0 \leq i \leq n - 1$  en la segunda parte de abajo, por  $\delta$  se cumplirá  $n - i - 1 < v$ .

- **Demostración formal de la corrección total:**

I	1. $\varphi \rightarrow INV$
II	2. $INV \rightarrow \text{def}(B)$
III	3. $(INV \wedge B) \rightarrow \varphi_2$ 4. $\{\varphi_2\} i := i + 1; \{\varphi_1\} \text{ (AA)}$ 5. $\{INV \wedge B\} i := i + 1; \{\varphi_1\} \text{ (RCN 3, 4)}$ 6. $\{\varphi_1\} s := s + A(i); \{INV\} \text{ (AA)}$ 7. $\{INV \wedge B\}$ $i := i + 1;$ $s := s + A(i);$ $\{INV\} \text{ (RCP 5, 6)}$
IV	8. $(INV \wedge \neg B) \rightarrow \psi$
V	9. $(INV \wedge B) \rightarrow E > 0$
VI	10. $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$ 11. $\{\varphi_4\} i := i + 1; \{\varphi_3\} \text{ (AA)}$ 12. $\{INV \wedge B \wedge E = v\} i := i + 1; \{\varphi_3\} \text{ (RCN 10, 11)}$ 13. $\{\varphi_3\} s := s + A(i); \{E < v\} \text{ (AA)}$ 14. $\{INV \wedge B \wedge E = v\}$ $i := i + 1;$ $s := s + A(i);$ $\{E < v\} \text{ (RCP 12, 13)}$ 15. $\{\varphi\}$ <b><u>while</u></b> $\{INV\} i \neq n$ <b><u>loop</u></b> $i := i + 1;$ $s := s + A(i);$ <b><u>end loop</u></b> ; $\{\psi\} \text{ (RWH 1, 2, 7, 8, 9, 14)}$

### 3.2.4.2. Ejemplo 11

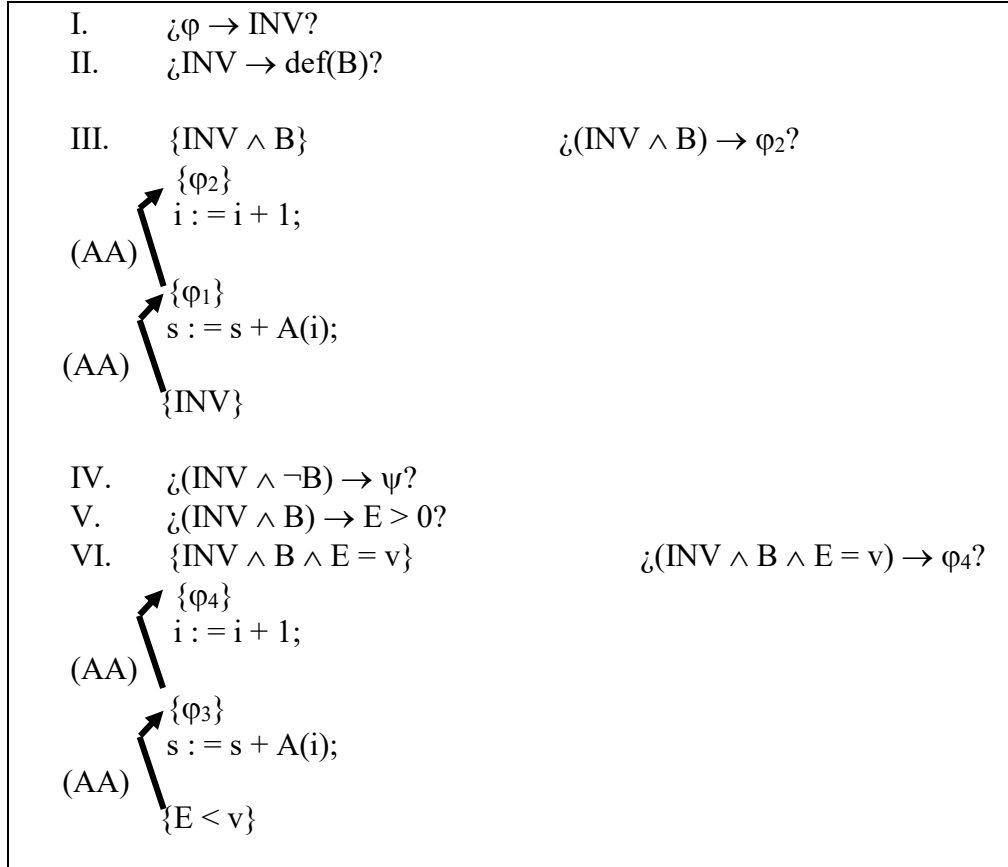
¿Es correcto el siguiente programa?

$\{\varphi\} \equiv \{i = 0 \wedge s = 0 \wedge n \geq 1\}$ <b><u>while</u></b> $\{INV\}$ $i \neq n$ <b><u>loop</u></b> $i := i + 1;$ $s := s + A(i);$ <b><u>end loop</u></b> ; $\{\psi\} \equiv \{s = \sum_{k=1}^n A(k)\}$
$\{INV\} \equiv \{(1 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)\}$ $E = n - i$

En verde se indica la única diferencia con respecto al ejemplo anterior.

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones



I. ¿ $\phi \rightarrow \text{INV}$ ?

$$\begin{array}{c}
 i = 0 \wedge s = 0 \wedge n \geq 1 \\
 \underbrace{\quad\quad\quad}_{\alpha} \quad \underbrace{\quad\quad\quad}_{\beta} \\
 \downarrow? \\
 (1 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k) \\
 \text{por } \alpha \text{ y } \beta
 \end{array}$$

En este caso  $\phi$  no implica el invariante ya que es posible que  $\phi$  sea cierto y que INV no lo sea. De hecho  $\phi$  dice que  $i = 0$  e INV dice que  $1 \leq i \leq n$ , y eso no es posible. Por tanto este programa no es correcto y no hay que proseguir con la prueba.

### 3.2.4.3. Ejemplo 12

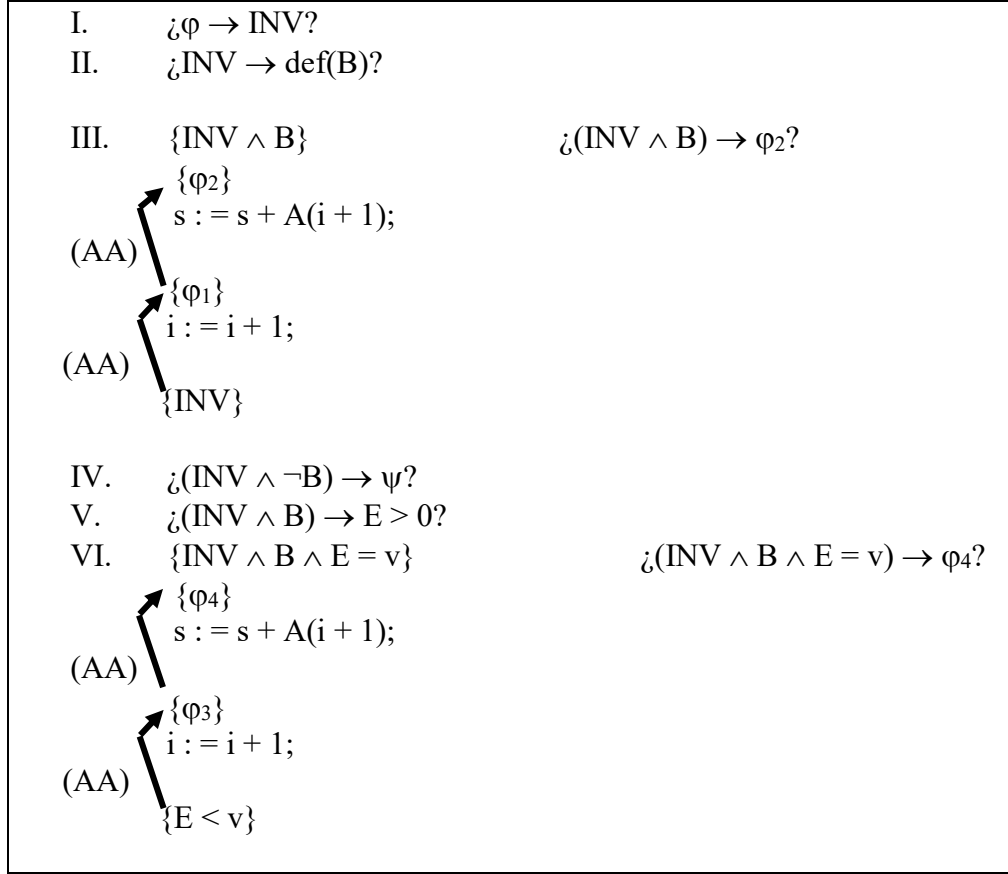
Este programa es muy parecido al del ejemplo 10 pero las instrucciones que van dentro del while están en orden inverso.

¿Es correcto el siguiente programa?

$\{\phi\} \equiv \{i = 0 \wedge s = 0 \wedge n \geq 1\}$ <b><u>while</u></b> $\{\text{INV}\}$ $i \neq n$ <b><u>loop</u></b> $s := s + A(i + 1);$ $i := i + 1;$ <b><u>end loop</u></b> ; $\{\psi\} \equiv \{s = \sum_{k=1}^n A(k)\}$
$\{\text{INV}\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)\}$  $E = n - i$

**Esquema:**

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones



I.  $\zeta \varphi \rightarrow \text{INV}?$

$$\zeta i = 0 \wedge s = 0 \wedge n \geq 0 \rightarrow (0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)?$$

$  \underbrace{i = 0}_{\alpha} \wedge \underbrace{s = 0}_{\beta} \wedge \underbrace{n \geq 1}_{\gamma} \rightarrow (0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)  $ <p style="text-align: center;"> <math>\underbrace{\hspace{1.5cm}}_{\text{por } \alpha \text{ y } \gamma} \quad \underbrace{\hspace{1.5cm}}_{\text{por } \alpha \text{ y } \beta}</math> </p>
---

Sí se cumple, ya que cada componente de la segunda parte de la implicación se justifica mediante las componentes de la primera parte tal como se indica mediante las letras griegas  $\alpha$ ,  $\beta$  y  $\gamma$ .

II.  $\zeta \text{INV} \rightarrow \text{def}(\text{B})?$

$\zeta \text{INV} \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.

## III.

$$\begin{aligned}
\bullet \quad \{\varphi_1\} &\equiv \{\text{def}(i+1) \wedge (\text{INV})_i^{i+1}\} \equiv \\
&\equiv \{\text{true} \wedge (0 \leq i+1 \leq n) \wedge s = \sum_{k=1}^{i+1} A(k)\} \equiv \text{simplificación} \\
&\equiv \{(-1 \leq i \leq n-1) \wedge s = \sum_{k=1}^{i+1} A(k)\}
\end{aligned}$$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $(0 \leq i+1 \leq n)$ , lo mejor es ponerlo utilizando  $i$  en vez de  $i+1$ . Para ello hay que restar 1 a los tres elementos:  $(0-1 \leq i+1-1 \leq n-1)$ . Tras realizar las operaciones queda  $(-1 \leq i \leq n-1)$ .

$$\begin{aligned}
\bullet \quad \{\varphi_2\} &\equiv \{\text{def}(s + A(i+1)) \wedge (\varphi_1)_{s+A(i)}^{s+A(i+1)}\} \equiv \\
&\equiv \{(1 \leq i+1 \leq n) \wedge (-1 \leq i \leq n-1) \wedge s + A(i+1) = \sum_{k=1}^{i+1} A(k)\} \equiv \\
\text{simplificación} &\equiv \{(0 \leq i \leq n-1) \wedge s + A(i+1) = \sum_{k=1}^{i+1} A(k)\}
\end{aligned}$$

Primero  $(1 \leq i+1 \leq n)$  se puede expresar como  $(0 \leq i \leq n-1)$ . A continuación se puede quitar  $(-1 \leq i \leq n-1)$  y dejar sólo  $(0 \leq i \leq n-1)$  porque si se cumple  $(0 \leq i \leq n-1)$  también se cumple  $(-1 \leq i \leq n-1)$ . Pero no se podría quitar  $(0 \leq i \leq n-1)$  y dejar  $(-1 \leq i \leq n-1)$  porque el que se cumpla  $(-1 \leq i \leq n-1)$  no quiere decir que se cumpla  $(0 \leq i \leq n-1)$ .

$$\bullet \quad \zeta(\text{INV} \wedge B) \rightarrow \varphi_2?$$

$$\begin{array}{ccc}
(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k) \wedge i \neq n & & \\
\begin{array}{ccc} \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} \\ \alpha & \beta & \delta \end{array} & & \\
\downarrow? & & \\
(0 \leq i \leq n-1) \wedge s + A(i+1) = \sum_{k=1}^{i+1} A(k) & & \\
\begin{array}{ccc} \underbrace{\hspace{1.5cm}} & \underbrace{\hspace{1.5cm}} & \\ \text{por } \alpha \text{ y } \delta & \text{por } \alpha, \beta \text{ y } \delta & \end{array} & & 
\end{array}$$

En la segunda parte de abajo, por  $\alpha$  y  $\delta$  sabemos que con  $i+1$  no nos salimos del rango de  $A(1..n)$  y luego por  $\beta$  se termina de deducir " $s + A(i+1) = \sum_{k=1}^{i+1} A(k)$ "

IV.  $\downarrow(\text{INV} \wedge \neg B) \rightarrow \psi?$

$$\begin{array}{c}
 (\underbrace{0 \leq i \leq n}_{\alpha}) \wedge \underbrace{s = \sum_{k=1}^i A(k)}_{\beta} \wedge \underbrace{i = n}_{\delta} \\
 \downarrow? \\
 \underbrace{\{s = \sum_{k=1}^n A(k)\}}_{\text{por } \beta \text{ y } \delta}
 \end{array}$$

En este caso al ser  $i = n$ , sustituyendo  $i$  por  $n$  en  $\beta$  tenemos lo mismo que en  $\psi$ .

V.  $\downarrow(\text{INV} \wedge B) \rightarrow E > 0?$

$$\begin{array}{c}
 (\underbrace{0 \leq i \leq n}_{\alpha}) \wedge \underbrace{s = \sum_{k=1}^i A(k)}_{\beta} \wedge \underbrace{i \neq n}_{\beta} \\
 \downarrow? \\
 \underbrace{n - i > 0}_{\text{por } \alpha \text{ y } \beta}
 \end{array}$$

Por  $\alpha$  sabemos que  $n \geq i$  y por tanto  $n - i \geq 0$  ya que si se cumple  $n \geq i$ , también se cumple  $n - i \geq i - i$  y eso es lo mismo que  $n - i \geq 0$ . Además por  $\beta$  sabemos que  $i \neq n$ . Por tanto  $n - i > 0$ .

VI.

$$\begin{aligned}
 \bullet \quad \{\varphi_3\} &\equiv \{\text{def}(i+1) \wedge (E < v)_i^{i+1}\} \equiv \\
 &\equiv \{\text{true} \wedge n - (i+1) < v\} \equiv \text{simplificación} \\
 &\equiv \{n - i - 1 < v\}
 \end{aligned}$$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

$$\begin{aligned}
 \bullet \quad \{\varphi_4\} &\equiv \{\text{def}(s + A(i+1)) \wedge (\varphi_3)_s^{s+A(i)}\} \equiv \\
 &\equiv \{(1 \leq i+1 \leq n) \wedge n - i - 1 < v\} \equiv \text{simplificación} \\
 &\equiv \{(0 \leq i \leq n-1) \wedge n - i - 1 < v\}
 \end{aligned}$$

En cuanto a la fórmula  $(1 \leq i+1 \leq n)$ , lo mejor es ponerlo utilizando  $i$  en vez de  $i+1$ . Para ello hay que restar el valor 1 a los tres elementos:  $(1-1 \leq i+1-1 \leq n-1)$ . Tras realizar las operaciones queda  $(0 \leq i \leq n-1)$ .

- ¿ $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$ ?

$$\begin{array}{c}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge s = \sum_{k=1}^i A(k) \wedge \underbrace{i \neq n}_{\beta} \wedge \underbrace{n - i = v}_{\delta} \\
 \downarrow ? \\
 \underbrace{(0 \leq i \leq n - 1)}_{\text{por } \alpha \text{ y } \beta} \wedge \underbrace{n - i - 1 < v}_{\text{por } \delta}
 \end{array}$$

En la primera parte de abajo, por  $\alpha$  y  $\beta$  sabemos que se cumplirá  $0 \leq i \leq n - 1$ .  
 En la segunda parte de abajo, por  $\delta$  se cumplirá  $n - i - 1 < v$ .

- **Demostración formal de la corrección total:**

I	1. $\varphi \rightarrow INV$
II	2. $INV \rightarrow \text{def}(B)$
III	3. $(INV \wedge B) \rightarrow \varphi_2$ 4. $\{\varphi_2\} s := s + A(i + 1); \{\varphi_1\}$ (AA) 5. $\{INV \wedge B\} s := s + A(i + 1); \{\varphi_1\}$ (RCN 3, 4) 6. $\{\varphi_1\} i := i + 1; \{INV\}$ (AA) 7. $\{INV \wedge B\}$ $s := s + A(i + 1);$ $i := i + 1;$ $\{INV\}$ (RCP 5, 6)
IV	8. $(INV \wedge \neg B) \rightarrow \psi$
V	9. $(INV \wedge B) \rightarrow E > 0$
VI	10. $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$ 11. $\{\varphi_4\} s := s + A(i + 1); \{\varphi_3\}$ (AA) 12. $\{INV \wedge B \wedge E = v\} s := s + A(i + 1); \{\varphi_3\}$ (RCN 10, 11) 13. $\{\varphi_3\} i := i + 1; \{E < v\}$ (AA) 14. $\{INV \wedge B \wedge E = v\}$ $s := s + A(i + 1);$ $i := i + 1;$ $\{E < v\}$ (RCP 12, 13)
	15. $\{\varphi\}$ <b><u>while</u></b> $\{INV\} i \neq n$ <b><u>loop</u></b> $s := s + A(i + 1);$ $i := i + 1;$ <b><u>end loop</u></b> ; $\{\psi\}$ (RWH 1, 2, 7, 8, 9, 14)

### 3.2.4.4. Ejemplo 13

Este programa es muy parecido al del ejemplo 10 y al del ejemplo 12 pero las instrucciones que van dentro del while cambian ligeramente y la precondition y el invariante también.

¿Es correcto el siguiente programa?

$\{\varphi\} \equiv \{i = 1 \wedge n \geq 1\}$ $s := 0;$ <b>while</b> {INV} $i \neq n + 1$ <b>loop</b> $s := s + A(i);$ $i := i + 1;$ <b>end loop;</b> $\{\psi\} \equiv \{s = \sum_{k=1}^n A(k)\}$
$\{INV\} \equiv \{(1 \leq i \leq n + 1) \wedge s = \sum_{k=1}^{i-1} A(k)\}$ $E = n + 1 - i$

#### Esquema:

- Como aparte del while hay una asignación previa al while, hay que distinguir dos subprogramas:
- 

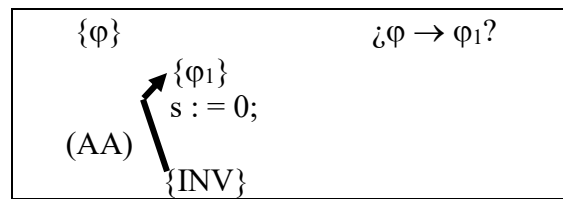
$\{\varphi\}$ $s := 0;$ $\{INV\}$
---

y

$\{INV\}$ <b>while</b> {INV} $i \neq n + 1$ <b>loop</b> $s := s + A(i);$ $i := i + 1;$ <b>end loop;</b> $\{\psi\}$
---



- Primero se ha de verificar el primer subprograma

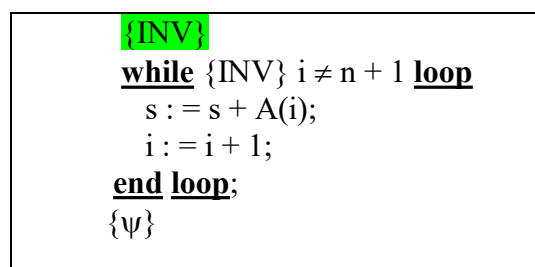


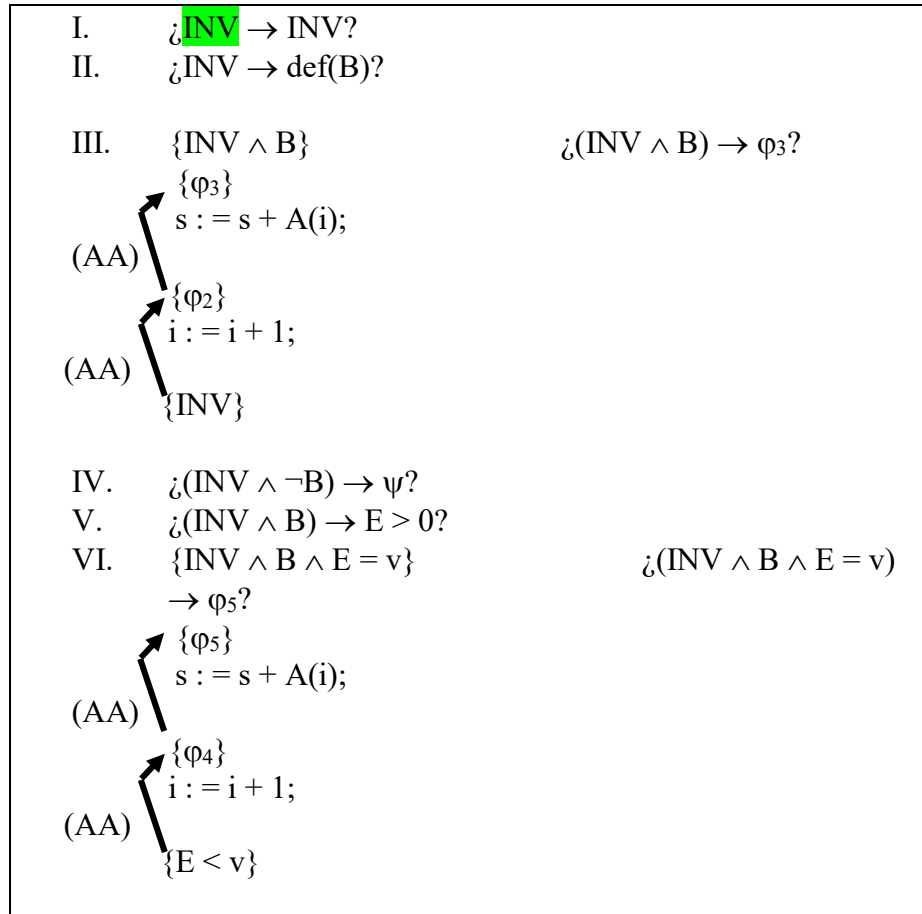
- $$\begin{aligned} \{ \varphi_1 \} &\equiv \{ \text{def}(0) \wedge (\text{INV})_s^0 \} \equiv \\ &\equiv \{ \text{true} \wedge (1 \leq i \leq n+1) \wedge 0 = \sum_{k=1}^i A(k) \} \equiv \text{simplificación} \\ &\equiv \{ (1 \leq i \leq n+1) \wedge 0 = \sum_{k=1}^i A(k) \} \end{aligned}$$
- $$\varphi \rightarrow \varphi_1?$$

$$\begin{array}{c} \{ i = 1 \wedge n \geq 1 \} \\ \underbrace{\quad \quad}_{\alpha} \quad \underbrace{\quad \quad}_{\beta} \\ \downarrow ? \\ (1 \leq i \leq n+1) \wedge 0 = \sum_{k=1}^{i-1} A(k) \\ \underbrace{\quad \quad}_{\text{por } \alpha \text{ y } \beta} \quad \underbrace{\quad \quad}_{\text{por } \alpha} \end{array}$$

En la segunda parte de abajo, por  $\alpha$  sabemos que  $i$  vale 1 y por tanto el sumatorio es 0 porque  $k$  va de 1 a 0 y siempre que el límite inferior sea mayor que el superior el sumatorio vale 0.

- La aplicación de la Regla del While (RWH) al segundo subprograma supondrá realizar los siguientes cálculos y comprobaciones





I.  $\{INV\} \rightarrow INV?$

Sí se cumple porque en ambos lados de la implicación tenemos lo mismo

II.  $\{INV\} \rightarrow \text{def}(B)?$

$\{INV\} \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.

III.

$$\begin{aligned}
 \bullet \quad \{\varphi_2\} &\equiv \{\text{def}(i+1) \wedge (INV)_i^{i+1}\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq i+1 \leq n+1) \wedge s = \sum_{k=1}^{i+1-1} A(k)\} \equiv \text{simplificación} \\
 &\equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)\}
 \end{aligned}$$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $(1 \leq i+1 \leq n)$ , lo mejor es ponerlo utilizando  $i$  en vez de  $i+1$ . Para ello hay que restar 1 a los tres elementos:  $(1-1 \leq i+1-1 \leq n-1)$ . Tras realizar las operaciones queda  $(0 \leq i \leq n-1)$ .

$$\begin{aligned}
\bullet \quad \{\varphi_3\} &\equiv \{\text{def}(s + A(i)) \wedge (\varphi_2)^{s + A(i)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge (0 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k)\} \equiv \text{simplificación} \\
&\equiv \{(1 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k)\}
\end{aligned}$$

se puede quitar  $(0 \leq i \leq n)$  y dejar sólo  $(1 \leq i \leq n)$  porque si se cumple  $(0 \leq i \leq n)$  también se cumple  $(1 \leq i \leq n)$ . Pero no se podría quitar  $(1 \leq i \leq n)$  y dejar  $(0 \leq i \leq n)$  porque el que se cumpla  $(0 \leq i \leq n)$  no quiere decir que se cumpla  $(1 \leq i \leq n)$ .

$$\bullet \quad \downarrow(\text{INV} \wedge B) \rightarrow \varphi_3?$$

$$\begin{array}{ccc}
\underbrace{(1 \leq i \leq n + 1)}_{\alpha} & \underbrace{\wedge s = \sum_{k=1}^{i-1} A(k)}_{\beta} & \underbrace{\wedge i \neq n + 1}_{\delta} \\
& \downarrow? & \\
\underbrace{(1 \leq i \leq n)}_{\text{por } \alpha \text{ y } \delta} & \underbrace{\wedge s + A(i) = \sum_{k=1}^i A(k)}_{\text{por } \alpha, \beta \text{ y } \delta} & 
\end{array}$$

En la segunda parte de abajo, por  $\alpha$  y  $\delta$  sabemos que con  $i$  no nos salimos del rango de  $A(1..n)$  y luego por  $\beta$  se termina de deducir " $s + A(i) = \sum_{k=1}^i A(k)$ "

$$\text{IV.} \quad \downarrow(\text{INV} \wedge \neg B) \rightarrow \psi?$$

$$\begin{array}{ccc}
\underbrace{(1 \leq i \leq n + 1)}_{\alpha} & \underbrace{\wedge s = \sum_{k=1}^{i-1} A(k)}_{\beta} & \underbrace{\wedge i = n + 1}_{\delta} \\
& \downarrow? & \\
& \underbrace{\{s = \sum_{k=1}^n A(k)\}}_{\text{por } \beta \text{ y } \delta} & 
\end{array}$$

En este caso al ser  $i = n + 1$ , sustituyendo  $i$  por  $n + 1$  en  $\beta$  tenemos lo mismo que en  $\psi$ .

V.  $\dot{?}(\text{INV} \wedge B) \rightarrow E > 0?$

$$\begin{array}{c}
 \underbrace{(1 \leq i \leq n+1)}_{\alpha} \wedge s = \sum_{k=1}^{i-1} A(k) \wedge \underbrace{i \neq n+1}_{\beta} \\
 \downarrow? \\
 \underbrace{n+1-i > 0} \\
 \text{por } \alpha \text{ y } \beta
 \end{array}$$

Por  $\alpha$  sabemos que  $n+1 \geq i$  y por tanto  $n+1-i \geq 0$  ya que si se cumple  $n+1 \geq i$ , también se cumple  $n+1-i \geq i-i$  y eso es lo mismo que  $n+1-i \geq 0$ . Además por  $\beta$  sabemos que  $i \neq n+1$ . Por tanto  $n+1-i > 0$ .

VI.

- $\{\varphi_4\} \equiv \{\text{def}(i+1) \wedge (E < v)_i^{i+1}\} \equiv$   
 $\equiv \{\text{true} \wedge n+1-(i+1) < v\} \equiv \text{simplificación}$   
 $\equiv \{n-i < v\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

- $\{\varphi_5\} \equiv \{\text{def}(s+A(i)) \wedge (\varphi_4)_s^{s+A(i)}\} \equiv$   
 $\equiv \{(1 \leq i \leq n) \wedge n-i < v\}$
- $\dot{?}(\text{INV} \wedge B \wedge E = v) \rightarrow \varphi_5?$

$$\begin{array}{c}
 \underbrace{(1 \leq i \leq n+1)}_{\alpha} \wedge s = \sum_{k=1}^{i-1} A(k) \wedge \underbrace{i \neq n+1}_{\beta} \wedge \underbrace{n+1-i = v}_{\delta} \\
 \downarrow? \\
 \underbrace{(1 \leq i \leq n)}_{\text{por } \alpha \text{ y } \beta} \wedge \underbrace{n-i < v}_{\text{por } \delta}
 \end{array}$$

En la primera parte de abajo, por  $\alpha$  y  $\beta$  sabemos que se cumplirá  $1 \leq i \leq n$  y en la segunda parte de abajo, de  $\delta$  se deduce que  $n-i = v-1$ , restando 1 en ambos lados de la igualdad y por tanto se cumplirá  $n-i < v$ .

• **Demostración formal de la corrección total:**

	1. $\varphi \rightarrow \varphi_1$
	2. $\{\varphi_1\} s := 0; \{INV\}$ (AA)
	3. $\{\varphi\} s := 0; \{INV\}$ (RCN 1, 2)
I	4. $INV \rightarrow INV$
II	5. $INV \rightarrow \text{def}(B)$
III	6. $(INV \wedge B) \rightarrow \varphi_3$
	7. $\{\varphi_3\} s := s + A(i); \{\varphi_2\}$ (AA)
	8. $\{INV \wedge B\} s := s + A(i); \{\varphi_2\}$ (RCN 6, 7)
	9. $\{\varphi_2\} i := i + 1; \{INV\}$ (AA)
	10. $\{INV \wedge B\}$ $s := s + A(i);$ $i := i + 1;$ $\{INV\}$ (RCP 8, 9)
IV	11. $(INV \wedge \neg B) \rightarrow \psi$
V	12. $(INV \wedge B) \rightarrow E > 0$
VI	13. $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$
	14. $\{\varphi_4\} s := s + A(i); \{\varphi_3\}$ (AA)
	15. $\{INV \wedge B \wedge E = v\} s := s + A(i); \{\varphi_3\}$ (RCN 13, 14)
	16. $\{\varphi_3\} i := i + 1; \{E < v\}$ (AA)
	17. $\{INV \wedge B \wedge E = v\}$ $s := s + A(i);$ $i := i + 1;$ $\{E < v\}$ (RCP 15, 16)
	18. $\{INV\}$ <u>while</u> $\{INV\} i \neq n + 1$ <u>loop</u> $s := s + A(i);$ $i := i + 1;$ <u>end loop;</u> $\{\psi\}$ (RWH 4, 5, 10, 11, 12, 17)
	19. $\{\varphi\}$ $s := 0;$ <u>while</u> $\{INV\} i \neq n + 1$ <u>loop</u> $s := s + A(i);$ $i := i + 1;$ <u>end loop;</u> $\{\psi\}$ (RCP 3, 18)

### 3.2.4.5. Ejemplo 14

**Verificar**, utilizando el Cálculo de Hoare, la corrección total del siguiente programa que según la especificación pre-post debería decidir en la variable booleana *ord* si los elementos del vector *A(1..n)* están en orden creciente (es decir, si cada elemento es menor o igual que el siguiente):

$\{\varphi\} \equiv \{n \geq 1 \wedge i = 1\}$ $\text{ord} := \text{true};$ <b><u>while</u></b> $\{\text{INV}\} \ i \neq n$ <b><u>and</u></b> $\text{ord}$ <b><u>loop</u></b> $\text{ord} := (A(i) \leq A(i + 1));$ $i := i + 1;$ <b><u>end loop</u></b> ; $\{\psi\} \equiv \{\text{ord} \leftrightarrow \text{creciente}(A(1..n))\}$
$\{\text{INV}\} \equiv \{(1 \leq i \leq n) \wedge (\text{ord} \leftrightarrow \text{creciente}(A(1..i)))\}$  $E = n - i$  $\text{creciente}(C(1..p)) \equiv \{\forall k(2 \leq k \leq p \rightarrow C(k-1) \leq C(k))\}$

#### Esquema:

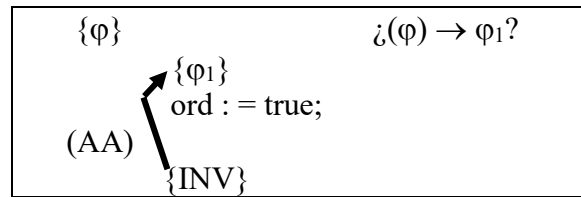
- Como aparte del while hay una asignación previa del while, hay que distinguir dos subprogramas:

$\{\varphi\}$ $\text{ord} := \text{true};$ $\{\text{INV}\}$
---

y

$\{\text{INV}\}$ <b><u>while</u></b> $\{\text{INV}\} \ i \neq n + 1$ <b><u>loop</u></b> $\text{ord} := (A(i) \leq A(i + 1));$ $i := i + 1;$ <b><u>end loop</u></b> ; $\{\psi\}$
--

- Primero se ha de verificar el primer subprograma



- $\{\varphi_1\} \equiv \{\text{def}(\text{true}) \wedge (\text{INV})_{\text{ord}^{\text{true}}}\} \equiv$   
 $\equiv \{\text{true} \wedge (1 \leq i \leq n) \wedge (\text{true} \leftrightarrow \text{creciente}(A(1..i)))\} \equiv \text{simplificación}$   
 $\equiv \{(1 \leq i \leq n) \wedge \text{creciente}(A(1..i))\}$

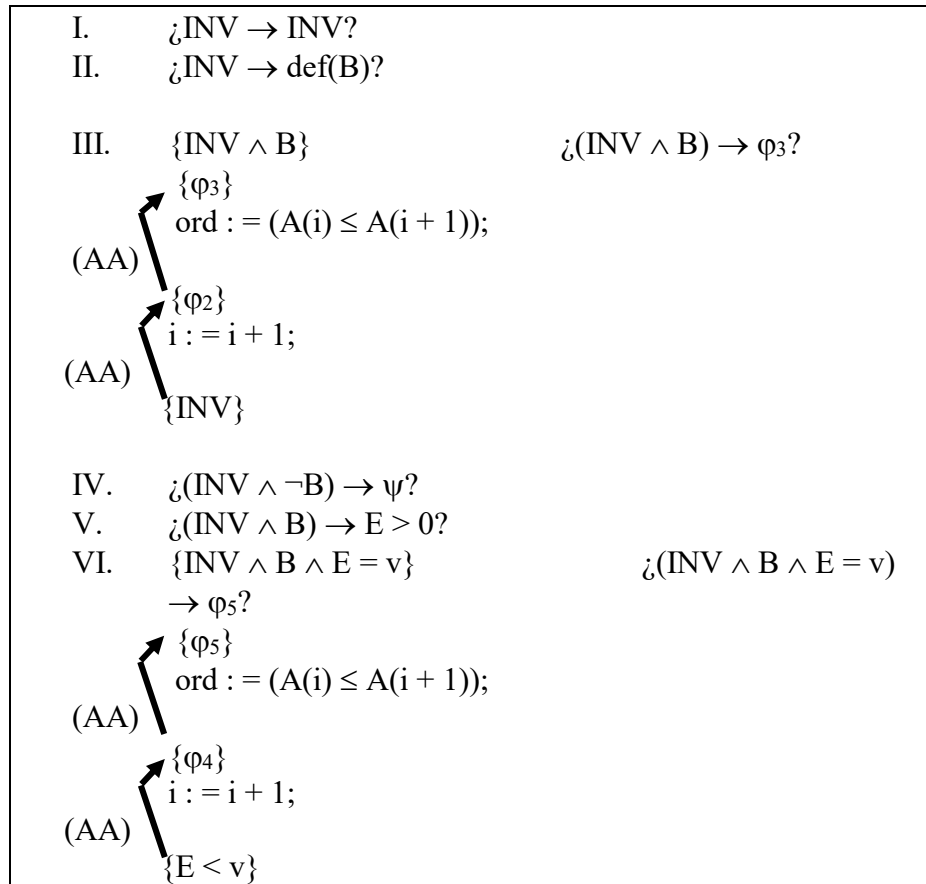
La simplificación está basada en que para cualquier fórmula  $\delta$  se cumple por una parte que  $\text{true} \wedge \delta \equiv \delta$  y por otra parte que  $\text{true} \leftrightarrow \delta \equiv \delta$ .

- ¿ $\varphi \rightarrow \varphi_1$ ?

$$\begin{array}{c}
 \{n \geq 1 \wedge i = 1\} \\
 \underbrace{\quad}_{\alpha} \quad \underbrace{\quad}_{\beta} \\
 \downarrow? \\
 (1 \leq i \leq n) \wedge \text{creciente}(A(1..i)) \\
 \underbrace{\quad}_{\text{por } \alpha \text{ y } \beta} \quad \underbrace{\quad}_{\text{por } \beta}
 \end{array}$$

Cuando  $i$  vale 1, en  $\text{creciente}(A(1..i))$  tenemos un intervalo vacío y se cumple la fórmula.

- La aplicación de la Regla del While (RWH) al **segundo subprograma** supondrá realizar los siguientes cálculos y comprobaciones
- Luego aplicamos la Regla del While (RWH) considerando que la precondition del while es  $\{INV\}$  y que la poscondición del while es  $\{\psi\}$ .



I.  $\zeta INV \rightarrow INV?$  Sí por ser iguales.

II.  $\zeta INV \rightarrow \text{def}(B)?$   
 $\zeta INV \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.



## III.

- $$\begin{aligned}\{\varphi_2\} &\equiv \{\text{def}(i+1) \wedge (\text{INV})_i^{i+1}\} \equiv \\ &\equiv \{(1 \leq i+1 \leq n) \wedge (\text{ord} \leftrightarrow \text{creciente}(A(1..i+1)))\}\end{aligned}$$
- $$\begin{aligned}\{\varphi_3\} &\equiv \{\text{def}(A(i) \leq A(i+1)) \wedge (\varphi_2)_{\text{ord}}^{A(i) \leq A(i+1)}\} \equiv \\ &\equiv \{(1 \leq i \leq n) \wedge (1 \leq i+1 \leq n) \wedge (1 \leq i+1 \leq n) \wedge \\ &\quad (A(i) \leq A(i+1) \leftrightarrow \text{creciente}(A(1..i+1)))\} \equiv \text{simplificación} \\ &\equiv \{(1 \leq i \leq n) \wedge (1 \leq i+1 \leq n) \wedge \\ &\quad (A(i) \leq A(i+1) \leftrightarrow \text{creciente}(A(1..i+1)))\} \equiv \text{simplificación} \\ &\equiv \{(1 \leq i \leq n) \wedge (0 \leq i \leq n-1) \wedge \\ &\quad (A(i) \leq A(i+1) \leftrightarrow \text{creciente}(A(1..i+1)))\} \equiv \text{simplificación} \\ &\equiv \{(1 \leq i \leq n-1) \wedge (A(i) \leq A(i+1) \leftrightarrow \text{creciente}(A(1..i+1)))\}\end{aligned}$$

En la primera simplificación se ha eliminado uno de los intervalos repetidos. En la segunda simplificación se ha pasado de tener  $(1 \leq i+1 \leq n)$  a tener  $(0 \leq i \leq n-1)$  restando 1 a los tres componentes. En la tercera simplificación, al tener dos intervalos distintos para  $i$ , hay que quedarse con el mayor de los límites inferiores (el mayor entre 1 y 0) y el menor de entre los límites superiores (el menor entre  $n$  y  $n-1$ ). Por ello el nuevo intervalo es  $(1 \leq i \leq n-1)$ .

- ¿ $(\text{INV} \wedge B) \rightarrow \varphi_3$ ?

$$\begin{array}{c} \underbrace{\{(1 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{ord} \leftrightarrow \text{creciente}(A(1..i)))}_{\beta} \wedge \underbrace{i \neq n}_{\gamma} \wedge \underbrace{\text{ord}}_{\delta} \\ \downarrow ? \\ \underbrace{(1 \leq i \leq n-1)}_{\text{por } \alpha \text{ y } \gamma} \wedge \underbrace{(A(i) \leq A(i+1) \leftrightarrow \text{creciente}(A(1..i+1)))}_{\text{por } \beta \text{ y } \delta} \end{array}$$

Por  $\alpha$  y  $\gamma$  deducimos que  $1 \leq i \leq n-1$ . En la doble implicación de  $\varphi_3$  se pregunta a ver si el hecho de que se cumpla  $\text{creciente}(A(1..i+1))$  depende de que se cumpla  $A(i) \leq A(i+1)$ . Y la respuesta es que sí porque por  $\delta$  sabemos que  $\text{ord}$  vale *true* y como conseqüencia de ello y por  $\beta$  sabemos que cumple  $\text{creciente}(A(1..i))$ . Por tanto  $A(i) \leq A(i+1)$  es la única comprobación que falta por hacer para ver si se cumple  $\text{creciente}(A(1..i+1))$ .

IV. ¿ $(INV \wedge \neg B) \rightarrow \psi$ ?

$$\underbrace{\{(1 \leq i \leq n) \wedge (\text{ord} \leftrightarrow \text{creciente}(A(1..i))) \wedge (i = n \vee \neg \text{ord})\}}_{\alpha} \quad \underbrace{\hspace{10em}}_{\beta} \quad \underbrace{\hspace{2em}}_{\gamma} \quad \underbrace{\hspace{2em}}_{\delta}$$

$\downarrow ?$   
 $\text{ord} \leftrightarrow \text{creciente}(A(1..n))?$

Como tenemos una disyunción hay que tener en cuenta las tres posibilidades de que  $(i = n \vee \neg \text{ord})$  sea True:

$i = n$	$\neg \text{ord}$
True	True
True	False
False	True

- ✓ En los dos primeros casos al ser  $i = n$ , ocurre que  $\beta$  y  $\psi$  son iguales y por tanto se cumple la implicación.
- ✓ En el tercer caso tenemos  $i \neq n$  y  $\text{ord} = \text{False}$ . De  $\alpha$  deducimos que  $i < n$  y de  $\beta$  deducimos que  $\neg \text{creciente}(A(1..i))$ . Es decir,  $\text{creciente}(A(1..i))$  es False. Al ser  $\text{creciente}(A(1..i))$  False e  $i < n$ , también  $\text{creciente}(A(1..n))$  es False. Por tanto en  $\psi$  nos queda  $\text{False} \leftrightarrow \text{False}$  y ello supone que  $\psi$  es True.

Esto todo quiere decir que la implicación se cumple.

V. ¿ $(INV \wedge B) \rightarrow E > 0$ ?

$$\begin{array}{c}
 \underbrace{\{(1 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{ord} \leftrightarrow \text{creciente}(A(1..i)))}_{\beta} \wedge \underbrace{i \neq n}_{\gamma} \wedge \underbrace{\text{ord}}_{\delta} \\
 \downarrow? \\
 \underbrace{n - i > 0}_{\text{por } \alpha \text{ y } \gamma}
 \end{array}$$

VI.

- $\{\varphi_4\} \equiv \{\text{def}(i + 1) \wedge (E < v)_{i+1}\} \equiv$   
 $\equiv \{\text{true} \wedge n - (i + 1) < v\} \equiv \{n - i - 1 < v\}$
- $\{\varphi_5\} \equiv \{\text{def}(A(i) \leq A(i + 1)) \wedge (\varphi_4)_{\text{ord}}^{A(i) \leq A(i + 1)}\} \equiv$   
 $\equiv \{(1 \leq i \leq n) \wedge (1 \leq i + 1 \leq n) \wedge (n - i - 1 < v)\} \equiv$   
 $\equiv \{(1 \leq i \leq n - 1) \wedge (n - i - 1 < v)\}$
- ¿ $(INV \wedge B \wedge E = v) \rightarrow \varphi_5$ ?

$$\begin{array}{c}
 \underbrace{\{(1 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{ord} \leftrightarrow \text{creciente}(A(1..i)))}_{\beta} \wedge \underbrace{i \neq n}_{\gamma} \wedge \underbrace{\text{ord}}_{\delta} \wedge \underbrace{(n - i = v)}_{\lambda} \\
 \downarrow? \\
 \underbrace{(1 \leq i \leq n - 1)}_{\text{por } \alpha \text{ y } \gamma} \wedge \underbrace{(n - i - 1 < v)}_{\text{por } \lambda}
 \end{array}$$

• **Demostración formal:**

	1. $\varphi \rightarrow \varphi_1$
	2. $\{\varphi_1\} \text{ ord} := \text{true}; \{\text{INV}\} \text{ (AA)}$
	3. $\{\varphi\} \text{ ord} := \text{true}; \{\text{INV}\} \text{ (RCN 1, 2)}$
I	4. $\text{INV} \rightarrow \text{INV}$
II	5. $\text{INV} \rightarrow \text{def(B)}$
III	6. $(\text{INV} \wedge \text{B}) \rightarrow \varphi_3$
	7. $\{\varphi_3\} \text{ ord} := (\text{A(i)} \leq \text{A(i + 1)}); \{\varphi_2\} \text{ (AA)}$
	8. $\{\text{INV} \wedge \text{B}\} \text{ ord} := (\text{A(i)} \leq \text{A(i + 1)}); \{\varphi_2\} \text{ (RCN 6, 7)}$
	9. $\{\varphi_2\} \text{ i} := \text{i + 1}; \{\text{INV}\} \text{ (AA)}$
	10. $\{\text{INV} \wedge \text{B}\}$ $\text{ord} := (\text{A(i)} \leq \text{A(i + 1)});$ $\text{i} := \text{i + 1};$ $\{\text{INV}\} \text{ (RCP 8, 9)}$
IV	11. $(\text{INV} \wedge \neg \text{B}) \rightarrow \psi$
V	12. $(\text{INV} \wedge \text{B}) \rightarrow \text{E} > 0$
VI	13. $(\text{INV} \wedge \text{B} \wedge \text{E} = \text{v}) \rightarrow \varphi_5$
	14. $\{\varphi_5\} \text{ ord} := (\text{A(i)} \leq \text{A(i + 1)}); \{\varphi_4\} \text{ (AA)}$
	15. $\{\text{INV} \wedge \text{B} \wedge \text{E} = \text{v}\} \text{ ord} := (\text{A(i)} \leq \text{A(i + 1)}); \{\varphi_4\} \text{ (RCN 13, 14)}$
	16. $\{\varphi_4\} \text{ i} := \text{i + 1}; \{\text{E} < \text{v}\} \text{ (AA)}$
	17. $\{\text{INV} \wedge \text{B} \wedge \text{E} = \text{v}\}$ $\text{ord} := (\text{A(i)} \leq \text{A(i + 1)});$ $\text{i} := \text{i + 1};$ $\{\text{E} < \text{v}\} \text{ (RCP 15, 16)}$
	18. $\{\text{INV}\}$ <u>while</u> $\{\text{INV}\} \text{ i} \neq \text{n}$ <u>and</u> <u>ord loop</u> $\text{ord} := (\text{A(i)} \leq \text{A(i + 1)});$ $\text{i} := \text{i + 1};$ <u>end loop;</u> $\{\psi\} \text{ (RWH 4, 5, 10, 11, 12, 17)}$
	19. $\{\varphi\}$ $\text{ord} := \text{true};$ <u>while</u> $\{\text{INV}\} \text{ i} \neq \text{n}$ <u>and</u> <u>ord loop</u> $\text{ord} := (\text{A(i)} \leq \text{A(i + 1)});$ $\text{i} := \text{i + 1};$ <u>end loop;</u> $\{\psi\} \text{ (RCP 3, 18)}$

### 3.2.4.6. Ejemplo 15

**Verificar**, utilizando el Cálculo de Hoare, la corrección total del siguiente programa que según la especificación pre-post debería decidir en la variable booleana *res* si en cada posición del vector  $R(1..n)$  se tiene el resto de dividir el correspondiente elemento de  $A(1..n)$  por 2

$\{\varphi\} \equiv \{n \geq 1 \wedge res\}$ $i := 0;$ <b>while</b> $\{INV\}$ $i \neq n$ <b>and</b> $res$ <b>loop</b> $res := (A(i + 1) \bmod 2 = R(i + 1));$ $i := i + 1;$ <b>end loop;</b> $\{\psi\} \equiv \{res \leftrightarrow concuerda(A(1..n), R(1..n))\}$
$\{INV\} \equiv \{(0 \leq i \leq n) \wedge (res \leftrightarrow concuerda(A(1..i), R(1..i)))\}$  $E = n - i$  $concuerda(D(1..p), F(1..p)) \equiv \{\forall k(1 \leq k \leq p \rightarrow D(k) \bmod 2 = F(k))\}$

#### Esquema:

- Como aparte del while hay una asignación previa del while, hay que distinguir dos subprogramas:

$\{\varphi\}$ $i := 0;$ <span style="background-color: #00FF00; padding: 2px;"><math>\{INV\}</math></span>
--

y

<span style="background-color: #00FF00; padding: 2px;"><math>\{INV\}</math></span> <b>while</b> $\{INV\}$ $i \neq n$ <b>and</b> $res$ <b>loop</b> $res := (A(i + 1) \bmod 2 = R(i + 1));$ $i := i + 1;$ <b>end loop;</b> $\{\psi\}$
--

- Primero se ha de verificar el primer subprograma

$\{\varphi\}$  <div style="display: flex; align-items: center;"> <div style="margin-right: 10px;">(AA)</div> <div style="text-align: center;"> <math>\begin{array}{c} \nearrow \{\varphi_1\} \\ i := 0; \\ \searrow \{INV\} \end{array}</math> </div> </div>	$\zeta(\varphi) \rightarrow \varphi_1?$
--	---

- $\{\varphi_1\} \equiv \{\text{def}(0) \wedge (\text{INV})_i^{0_1}\} \equiv$   
 $\equiv \{\text{true} \wedge (0 \leq 0 \leq n) \wedge (\text{res} \leftrightarrow \text{concuerda}(\text{A}(1..0), \text{R}(1..0)))\} \equiv \text{simplificación}$   
 $\equiv \{(0 \leq n) \wedge (\text{res} \leftrightarrow \text{concuerda}(\text{A}(1..0), \text{R}(1..0)))\} \equiv \text{simplificación}$   
 $\equiv \{(0 \leq n) \wedge (\text{res} \leftrightarrow \text{true})\} \equiv \text{simplificación}$   
 $\equiv \{(0 \leq n) \wedge \text{res}\}$

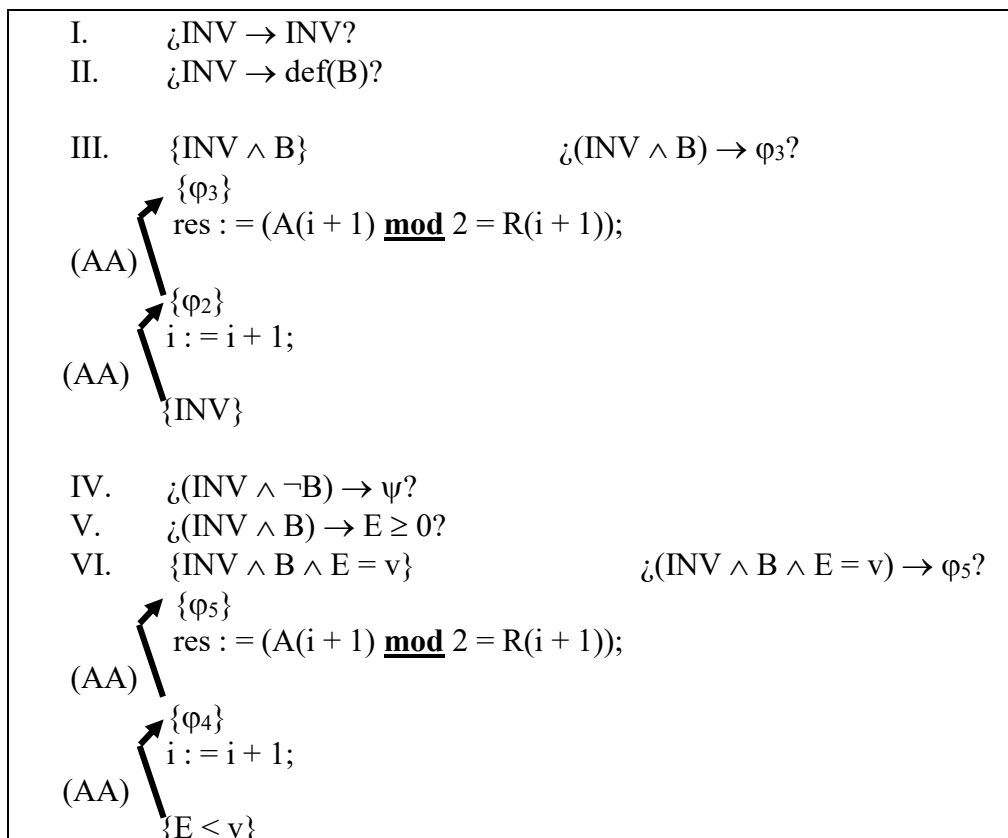
La segunda simplificación se realiza porque en  $\text{concuerda}(\text{A}(1..0), \text{R}(1..0))$  el intervalo es vacío y por tanto es true.

En el proceso de simplificación se ha tenido en cuenta que para cualquier fórmula  $\delta$  se cumple por una parte que  $\text{true} \wedge \delta \equiv \delta$  y por otra parte que  $\text{true} \leftrightarrow \delta \equiv \delta$ .

- $\zeta\varphi \rightarrow \varphi_1?$

$$\begin{array}{c}
 \underbrace{\{n \geq 1\}}_{\alpha} \wedge \underbrace{\{\text{res}\}}_{\beta} \\
 \downarrow ? \\
 \underbrace{(0 \leq n)}_{\text{por } \alpha} \wedge \underbrace{\{\text{res}\}}_{\text{por } \beta}
 \end{array}$$

- La aplicación de la Regla del While (RWH) al **segundo subprograma** supondrá realizar los siguientes cálculos y comprobaciones considerando que la precondition del while es  $\{\text{INV}\}$  y que la poscondición del while es  $\{\psi\}$ .



I.  $\dot{I}INV \rightarrow INV$ ? Sí por ser iguales.

II.  $\dot{I}INV \rightarrow \text{def}(B)$ ?

$\dot{I}INV \rightarrow \text{true}$ ? Sí, porque la segunda parte de la implicación es true.

III.

- $\{\varphi_2\} \equiv \{\text{def}(i+1) \wedge (INV)_i^{i+1}\} \equiv$   
 $\equiv \{(0 \leq i+1 \leq n) \wedge (\text{res} \leftrightarrow \text{concuerda}(A(1..i+1), R(1..i+1)))\}$
- $\{\varphi_3\} \equiv \{\text{def}(A(i+1) \bmod 2 = R(i+1)) \wedge (\varphi_2)_{\text{res}}^{A(i+1) \bmod 2 = R(i+1)}\} \equiv$   
 $\equiv \{(1 \leq i+1 \leq n) \wedge (1 \leq i+1 \leq n) \wedge (0 \leq i+1 \leq n) \wedge$   
 $(A(i+1) \bmod 2 = R(i+1) \leftrightarrow \text{concuerda}(A(1..i+1), R(1..i+1)))\} \equiv$   
 $\equiv \{(0 \leq i \leq n-1) \wedge$   
 $(A(i+1) \bmod 2 = R(i+1) \leftrightarrow \text{concuerda}(A(1..i+1), R(1..i+1)))\}$
- $\dot{I}(INV \wedge B) \rightarrow \varphi_3$ ?

$$\begin{array}{c}
 \underbrace{\{(0 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{res} \leftrightarrow \text{concuerda}(A(1..i), R(1..i)))}_{\beta} \wedge \underbrace{i \neq n}_{\gamma} \wedge \underbrace{\text{res}}_{\delta} \\
 \downarrow ? \\
 \underbrace{(0 \leq i \leq n-1)}_{\text{por } \alpha \text{ y } \gamma} \wedge \underbrace{(A(i+1) \bmod 2 = R(i+1) \leftrightarrow \text{concuerda}(A(1..i+1), R(1..i+1)))}_{\text{por } \beta \text{ y } \delta}
 \end{array}$$

Por  $\beta$  y  $\delta$  deducimos que es cierto **concuerda**(A(1..i), R(1..i)). En la doble implicación de  $\varphi_3$  se pregunta a ver si el hecho de que se cumpla **concuerda**(A(1..i+1), R(1..i+1)) depende de que se cumpla  $A(i+1) \bmod 2 = R(i+1)$ . Y la respuesta es que sí porque sabemos que se cumple **concuerda**(A(1..i), R(1..i)) y por tanto  $A(i+1) \bmod 2 = R(i+1)$  es la única comprobación que falta por hacer para ver si se cumple **concuerda**(A(1..i+1), R(1..i+1)).

IV.  $\dot{I}(INV \wedge \neg B) \rightarrow \psi$ ?

$$\begin{array}{c}
 \underbrace{\{(0 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{res} \leftrightarrow \text{concuerda}(A(1..i), R(1..i)))}_{\beta} \wedge \underbrace{(i = n \vee \neg \text{res})}_{\gamma} \\
 \downarrow ? \\
 \text{res} \leftrightarrow \text{concuerda}(A(1..n), R(1..n))
 \end{array}$$

Como tenemos una disyunción hay que tener en cuenta las tres posibilidades de que  $(i = n \vee \neg \text{res})$  sea True:

	$i = n$	$\neg \text{res}$
{	True	True
	True	False
}	False	True

- En los dos primeros casos al ser  $i = n$ , ocurre que  $\beta$  y  $\psi$  son iguales y por tanto se cumple la implicación.
- En el tercer caso tenemos  $i \neq n$  y  $\text{res} = \text{False}$ . De  $i \neq n$  y  $a$  deducimos que  $i < n$  y de  $\text{res} = \text{False}$  y  $b$  deducimos que  $\neg \text{concuerta}(A(1..i), R(1..i))$ . Es decir,  $\text{concuerta}(A(1..i), R(1..i))$  es False. Al ser  $\text{concuerta}(A(1..i), R(1..i))$  False e  $i < n$ , también  $\text{concuerta}(A(1..n), R(1..n))$  es False. Por tanto en  $\psi$  nos queda  $\text{False} \leftrightarrow \text{False}$  y de ahí tenemos que  $\psi$  es True.

Esto todo quiere decir que la implicación se cumple.

V.  $\downarrow(\text{INV} \wedge B) \rightarrow E > 0?$

$$\begin{array}{c}
 \underbrace{\{(0 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{res} \leftrightarrow \text{concuerta}(A(1..i), R(1..i)))}_{\beta} \wedge \underbrace{i \neq n}_{\gamma} \wedge \underbrace{\text{res}}_{\delta} \\
 \downarrow? \\
 n - i > 0 \\
 \text{por } \alpha \text{ y } \gamma
 \end{array}$$

VI.

- $\{\varphi_4\} \equiv \{\text{def}(i+1) \wedge (E < v)^{i+1}\} \equiv$   
 $\equiv \{\text{true} \wedge n - (i+1) < v\} \equiv \{n - i - 1 < v\}$
- $\{\varphi_5\} \equiv \{\text{def}(A(i+1) \bmod 2 = R(i+1)) \wedge (\varphi_4)_{\text{res}}^{A(i+1) \bmod 2 = R(i+1)}\} \equiv$   
 $\equiv \{(1 \leq i+1 \leq n) \wedge (1 \leq i+1 \leq n) \wedge (n - i - 1 < v)\} \equiv$   
 $\equiv \{(0 \leq i \leq n-1) \wedge (n - i - 1 < v)\}$
- $\downarrow(\text{INV} \wedge B \wedge E = v) \rightarrow \varphi_5?$

$$\begin{array}{c}
 \underbrace{\{(0 \leq i \leq n)\}}_{\alpha} \wedge \underbrace{(\text{res} \leftrightarrow \text{concuerta}(A(1..i), R(1..i)))}_{\beta} \wedge \underbrace{i \neq n}_{\gamma} \wedge \underbrace{\text{res}}_{\delta} \wedge \underbrace{(n - i = v)}_{\lambda} \\
 \downarrow? \\
 (0 \leq i \leq n-1) \wedge (n - i - 1 < v) \\
 \underbrace{(0 \leq i \leq n-1)}_{\text{por } \alpha \text{ y } \gamma} \wedge \underbrace{(n - i - 1 < v)}_{\text{por } \lambda}
 \end{array}$$



• **Demostración formal:**

	1. $\varphi \rightarrow \varphi_1$
	2. $\{\varphi_1\} i := 0; \{INV\}$ (AA)
	3. $\{\varphi\} i := 0; \{INV\}$ (RCN 1, 2)
I	4. $INV \rightarrow INV$
II	5. $INV \rightarrow \text{def}(B)$
III	6. $(INV \wedge B) \rightarrow \varphi_3$
	7. $\{\varphi_3\} \text{res} := (A(i+1) \bmod 2 = R(i+1)); \{\varphi_2\}$ (AA)
	8. $\{INV \wedge B\} \text{res} := (A(i+1) \bmod 2 = R(i+1)); \{\varphi_2\}$ (RCN 6, 7)
	9. $\{\varphi_2\} i := i + 1; \{INV\}$ (AA)
	10. $\{INV \wedge B\}$ $\text{res} := (A(i+1) \bmod 2 = R(i+1));$ $i := i + 1;$ $\{INV\}$ (RCP 8, 9)
IV	11. $(INV \wedge \neg B) \rightarrow \psi$
V	12. $(INV \wedge B) \rightarrow E > 0$
VI	13. $(INV \wedge B \wedge E = v) \rightarrow \varphi_5$
	14. $\{\varphi_5\} \text{res} := (A(i+1) \bmod 2 = R(i+1)); \{\varphi_4\}$ (AA)
	15. $\{INV \wedge B \wedge E = v\} \text{res} := (A(i+1) \bmod 2 = R(i+1)); \{\varphi_4\}$ (RCN 13, 14)
	16. $\{\varphi_4\} i := i + 1; \{E < v\}$ (AA)
	17. $\{INV \wedge B \wedge E = v\}$ $\text{res} := (A(i+1) \bmod 2 = R(i+1));$ $i := i + 1;$ $\{E < v\}$ (RCP 15, 16)
	18. $\{INV\}$ <b>while</b> $\{INV\} i \neq n$ <b>and</b> $\text{res}$ <b>loop</b> $\text{res} := (A(i+1) \bmod 2 = R(i+1));$ $i := i + 1;$ <b>end loop;</b> $\{\psi\}$ (RWH 4, 5, 10, 11, 12, 17)
	19. $\{\varphi\}$ $i := 0;$ <b>while</b> $\{INV\} i \neq n$ <b>and</b> $\text{res}$ <b>loop</b> $\text{res} := (A(i+1) \bmod 2 = R(i+1));$ $i := i + 1;$ <b>end loop;</b> $\{\psi\}$ (RCP 3, 18)

### 3.2.4.7. Ejemplo 16

¿Es correcto el siguiente programa?

$\{\phi\} \equiv \{\text{true}\}$ <b>while</b> $\{\text{INV}\} \ i \neq 0$ <b>loop</b> $i := i - 2;$ <b>end loop</b> ; $\{\psi\} \equiv \{i = 0\}$
$\{\text{INV}\} \equiv \{(i \geq 0) \wedge \text{par}(i)\}$  $E = i$

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones

I.	$\zeta \phi \rightarrow \text{INV}?$	
II.	$\zeta \text{INV} \rightarrow \text{def}(B)?$	
III.	$\{\text{INV} \wedge B\}$	$\zeta(\text{INV} \wedge B) \rightarrow \phi_1?$
(AA)	$\begin{array}{l} \nearrow \{\phi_1\} \\ i := i - 2; \\ \searrow \{\text{INV}\} \end{array}$	
IV.	$\zeta(\text{INV} \wedge \neg B) \rightarrow \psi?$	
V.	$\zeta(\text{INV} \wedge B) \rightarrow E > 0?$	
VI.	$\{\text{INV} \wedge B \wedge E = v\}$	$\zeta(\text{INV} \wedge B \wedge E = v) \rightarrow \phi_2?$
(AA)	$\begin{array}{l} \nearrow \{\phi_2\} \\ i := i - 2; \\ \searrow \{E < v\} \end{array}$	

- I.  $\zeta \phi \rightarrow \text{INV}?$   
 $\zeta \text{true} \rightarrow (i \geq 0) \wedge \text{par}(i)?$

Esa implicación no se cumple. Un contraejemplo sería cuando  $i$  vale 3. Se cumple la primera parte de la implicación, pero la segunda es false. Nos queda  $\text{true} \rightarrow \text{false}$  y eso es false.

Por tanto **el programa no es parcialmente correcto.**

### 3.2.4.8. Ejemplo 17

¿Es correcto el siguiente programa?

Este programa es muy parecido al del ejemplo 16 pero cambia el invariante.

$\{\varphi\} \equiv \{\text{true}\}$ <b><u>while</u></b> $\{\text{INV}\} \ i \neq 0$ <b><u>loop</u></b> $\quad i := i - 2;$ <b><u>end loop</u></b> ; $\{\psi\} \equiv \{i = 0\}$
$\{\text{INV}\} \equiv \{\text{true}\}$  $E = i$

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones

I.	$\zeta \varphi \rightarrow \text{INV}?$	
II.	$\zeta \text{INV} \rightarrow \text{def}(B)?$	
III.	$\{\text{INV} \wedge B\}$	$\zeta(\text{INV} \wedge B) \rightarrow \varphi_1?$
(AA)	$\begin{array}{l} \nearrow \{\varphi_1\} \\ i := i - 2; \\ \searrow \{\text{INV}\} \end{array}$	
IV.	$\zeta(\text{INV} \wedge \neg B) \rightarrow \psi?$	
V.	$\zeta(\text{INV} \wedge B) \rightarrow E > 0?$	
VI.	$\{\text{INV} \wedge B \wedge E = v\}$	$\zeta(\text{INV} \wedge B \wedge E = v) \rightarrow \varphi_2?$
(AA)	$\begin{array}{l} \nearrow \{\varphi_2\} \\ i := i - 2; \\ \searrow \{E < v\} \end{array}$	

- I.  $\zeta \varphi \rightarrow \text{INV}?$   
 $\zeta \text{true} \rightarrow \text{true}?$  Sí
- II.  $\zeta \text{INV} \rightarrow \text{def}(B)?$   
 $\zeta \text{true} \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.

**III.**

$$\begin{aligned} \bullet \quad \{\varphi_1\} &\equiv \{\text{def}(i-2) \wedge (\text{INV})_i^{i-2}\} \equiv \\ &\equiv \{\text{true} \wedge \text{true}\} \equiv \text{simplificación} \\ &\equiv \{\text{true}\} \end{aligned}$$

$$\bullet \quad \dot{\vdash}(\text{INV} \wedge B) \rightarrow \varphi_1?$$

$\dot{\vdash}(\text{true} \wedge i \neq 0) \rightarrow \text{true}$ ? Sí, porque la segunda parte de la implicación es true

$$\text{IV.} \quad \dot{\vdash}(\text{INV} \wedge \neg B) \rightarrow \psi?$$

$$\dot{\vdash}(\text{true} \wedge \neg(i \neq 0)) \rightarrow i = 0? \text{ Sí.}$$

$$\text{V.} \quad \dot{\vdash}(\text{INV} \wedge B) \rightarrow E > 0?$$

$$\dot{\vdash}(\text{true} \wedge (i \neq 0)) \rightarrow i > 0? \text{ No.}$$

Ya no hace falta mirar el punto VI. El programa es parcialmente correcto pero no totalmente correcto.

• **Demostración formal de la corrección parcial:**

I	1. $\varphi \rightarrow \text{INV}$
II	2. $\text{INV} \rightarrow \text{def}(B)$
III	3. $(\text{INV} \wedge B) \rightarrow \varphi_1$ 4. $\{\varphi_1\} i := i - 2; \{\text{INV}\} \text{ (AA)}$ 5. $\{\text{INV} \wedge B\}$ $i := i - 2;$ $\{\text{INV}\} \text{ (RCP 5, 6)}$
IV	6. $(\text{INV} \wedge \neg B) \rightarrow \psi$  7. $\{\varphi\}$ <b><u>while</u></b> $\{\text{INV}\} i \neq n$ <b><u>loop</u></b> $i := i - 2;$ <b><u>end loop</u></b> ; $\{\psi\} \text{ (RWH 1, 2, 5, 6)}$

Con esta prueba sólo se demuestra la corrección parcial.

### 3.2.4.9. Ejemplo 18

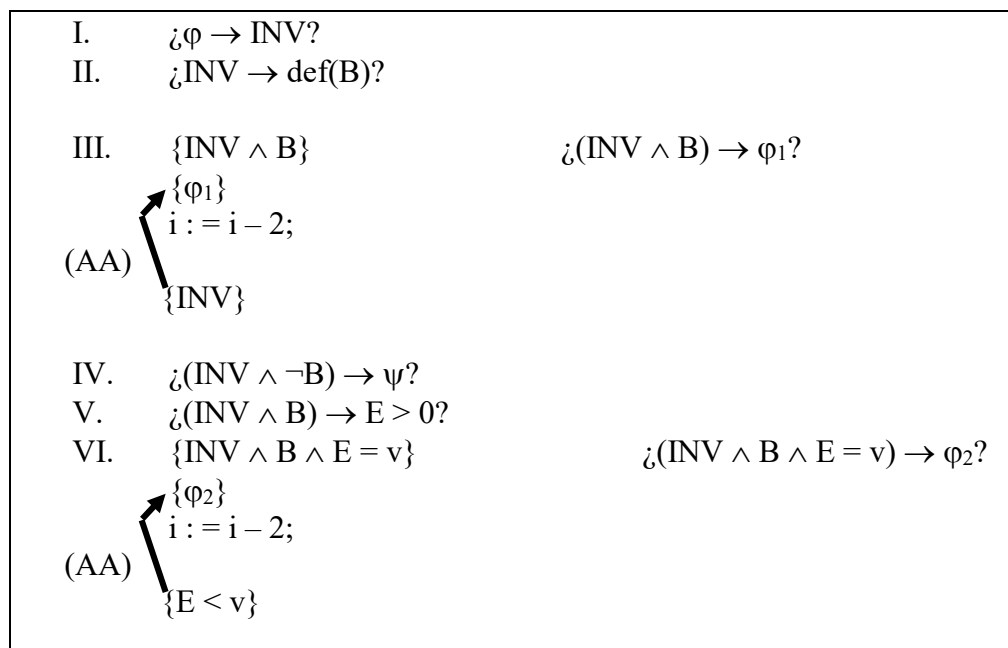
¿Es correcto el siguiente programa?

Este programa es muy parecido a los ejemplos anteriores. La diferencia está en la precondition y en el invariante.

$\{\varphi\} \equiv \{i \geq 0 \wedge \text{par}(i)\}$ <b>while</b> $\{\text{INV}\}$ $i \neq 0$ <b>loop</b> $i := i - 2;$ <b>end loop</b> ; $\{\psi\} \equiv \{i = 0\}$
$\{\text{INV}\} \equiv \{i \geq 0 \wedge \text{par}(i)\}$  $E = i$

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones



- I.  $\zeta \varphi \rightarrow \text{INV}?$   
 $\zeta i \geq 0 \wedge \text{par}(i) \rightarrow i \geq 0 \wedge \text{par}(i)?$  Sí, porque los dos lados son iguales
- II.  $\zeta \text{INV} \rightarrow \text{def}(B)?$   
 $\zeta i \geq 0 \wedge \text{par}(i) \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.

**III.**

- $\{\varphi_1\} \equiv \{\text{def}(i-2) \wedge (\text{INV})_i^{i-2}\} \equiv$   
 $\equiv \{\text{true} \wedge i-2 \geq 0 \wedge \text{par}(i-2)\} \equiv \text{simplificación}$   
 $\equiv \{i-2 \geq 0 \wedge \text{par}(i-2)\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

- $\dot{?}(\text{INV} \wedge B) \rightarrow \varphi_1?$

$$\begin{array}{c}
 \{i \geq 0 \wedge \text{par}(i) \wedge i \neq 0\} \\
 \underbrace{\quad \quad \quad}_{\alpha} \quad \underbrace{\quad \quad}_{\beta} \quad \underbrace{\quad \quad}_{\delta} \\
 \quad \quad \quad \downarrow ? \\
 \{i-2 \geq 0 \wedge \text{par}(i-2)\} \\
 \underbrace{\quad \quad}_{\text{por } \alpha, \beta \text{ y } \delta} \quad \underbrace{\quad \quad}_{\text{por } \beta}
 \end{array}$$

**IV.**  $\dot{?}(\text{INV} \wedge \neg B) \rightarrow \psi?$ 

$$\dot{?}(i \geq 0 \wedge \text{par}(i) \wedge \underbrace{\neg(i \neq 0)}_{\alpha}) \rightarrow i = 0? \text{ Sí por } \alpha$$

**V.**  $\dot{?}(\text{INV} \wedge B) \rightarrow E > 0?$ 

$$\dot{?}(\underbrace{i \geq 0}_{\alpha} \wedge \text{par}(i) \wedge \underbrace{i \neq 0}_{\beta}) \rightarrow i > 0? \text{ Sí por } \alpha \text{ y } \beta.$$

**VI.**

- $\{\varphi_2\} \equiv \{\text{def}(i-2) \wedge (E < v)_i^{i-2}\} \equiv$   
 $\equiv \{\text{true} \wedge (i-2) < v\} \equiv \text{simplificación}$   
 $\equiv \{(i-2) < v\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

- $\dot{?}(\text{INV} \wedge B \wedge E = v) \rightarrow \varphi_2?$

$$\dot{?}(i \geq 0 \wedge \text{par}(i) \wedge \underbrace{i \neq 0}_{\alpha} \wedge \underbrace{i = v}_{\alpha}) \rightarrow \underbrace{(i-2) < v}_{\text{por } \alpha}? \text{ Sí.}$$

• **Demostración formal de la corrección total:**

I	1.	$\varphi \rightarrow \text{INV}$
II	2.	$\text{INV} \rightarrow \text{def}(\text{B})$
III	3.	$(\text{INV} \wedge \text{B}) \rightarrow \varphi_1$
	4.	$\{\varphi_1\} i := i - 2; \{\text{INV}\} \text{ (AA)}$
	5.	$\{\text{INV} \wedge \text{B}\}$ $i := i - 2;$ $\{\text{INV}\} \text{ (RCN 3, 4)}$
IV	6.	$(\text{INV} \wedge \neg \text{B}) \rightarrow \psi$
V	7.	$(\text{INV} \wedge \text{B}) \rightarrow E > 0$
VI	8.	$(\text{INV} \wedge \text{B} \wedge E = v) \rightarrow \varphi_2$
	9.	$\{\varphi_2\} i := i - 2; \{E < v\} \text{ (AA)}$
	10.	$\{\text{INV} \wedge \text{B} \wedge E = v\}$ $i := i - 2;$ $\{E < v\} \text{ (RCN 8, 9)}$
	11.	$\{\varphi\}$ <b><u>while</u></b> $\{\text{INV}\} i \neq n$ <b><u>loop</u></b> $i := i - 2;$ <b><u>end loop</u></b> ; $\{\psi\} \text{ (RWH 1, 2, 5, 6, 7, 10)}$

### 3.2.4.10. Ejemplo 19

¿Es correcto el siguiente programa?

Este programa es muy parecido al de los ejemplos anteriores. La diferencia está en la precondition y en el invariante.

$\{\phi\} \equiv \{i \geq 0 \wedge \text{par}(i)\}$ <b>while</b> $\{\text{INV}\} i \neq 0$ <b>loop</b> $i := i - 2;$ <b>end loop;</b> $\{\psi\} \equiv \{i = 0\}$
$\{\text{INV}\} \equiv \{\text{true}\}$  $E = i$

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones

I.	$\dot{?} \phi \rightarrow \text{INV}?$	
II.	$\dot{?} \text{INV} \rightarrow \text{def}(B)?$	
III.	$\{\text{INV} \wedge B\}$	$\dot{?} (\text{INV} \wedge B) \rightarrow \phi_1?$
	$\begin{array}{c} \uparrow \{\phi_1\} \\ i := i - 2; \\ \downarrow \{\text{INV}\} \end{array}$	
(AA)		
IV.	$\dot{?} (\text{INV} \wedge \neg B) \rightarrow \psi?$	
V.	$\dot{?} (\text{INV} \wedge B) \rightarrow E > 0?$	
VI.	$\{\text{INV} \wedge B \wedge E = v\}$	$\dot{?} (\text{INV} \wedge B \wedge E = v) \rightarrow \phi_2?$
	$\begin{array}{c} \uparrow \{\phi_2\} \\ i := i - 2; \\ \downarrow \{E < v\} \end{array}$	
(AA)		

- I.  $\dot{?} \phi \rightarrow \text{INV}?$   
 $\dot{?} i \geq 0 \wedge \text{par}(i) \rightarrow \text{true}?$  Sí porque la segunda parte es true.
- II.  $\dot{?} \text{INV} \rightarrow \text{def}(B)?$   
 $\dot{?} \text{true} \rightarrow \text{true}?$  Sí, porque la segunda parte de la implicación es true.



**III.**

- $\{\varphi_1\} \equiv \{\text{def}(i-2) \wedge (\text{INV})_i^{i-2}\} \equiv$   
 $\equiv \{\text{true} \wedge \text{true}\} \equiv \text{simplificación}$   
 $\equiv \{\text{true}\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

- $\zeta(\text{INV} \wedge B) \rightarrow \varphi_1?$   
 $\zeta(\text{true} \wedge i \neq 0) \rightarrow \text{true?}$  Sí, porque la segunda parte de la implicación es true

**IV.**  $\zeta(\text{INV} \wedge \neg B) \rightarrow \psi?$ 

$\zeta(\text{true} \wedge \underbrace{\neg(i \neq 0)}_{\alpha}) \rightarrow \underbrace{i = 0}_{\text{por } \alpha}?$  Sí.

**V.**  $\zeta(\text{INV} \wedge B) \rightarrow E > 0?$ 

$\zeta(\text{true} \wedge (i \neq 0)) \rightarrow i > 0?$  No. El que  $i$  sea distinto de 0 no quiere decir que sea mayor que 0, ya que podría ser menor que 0.

Ya no hace falta mirar el punto VI. El programa es parcialmente correcto pero no totalmente correcto.

- Demostración formal de la corrección parcial:**

I	1.	$\varphi \rightarrow \text{INV}$
II	2.	$\text{INV} \rightarrow \text{def}(B)$
III	3.	$(\text{INV} \wedge B) \rightarrow \varphi_1$
	4.	$\{\varphi_1\} i := i - 2; \{\text{INV}\} \text{ (AA)}$
	5.	$\{\text{INV} \wedge B\}$ $i := i - 2;$ $\{\text{INV}\} \text{ (RCN 3, 4)}$
IV	6.	$(\text{INV} \wedge \neg B) \rightarrow \psi$

Esta prueba sólo demuestra la corrección parcial.

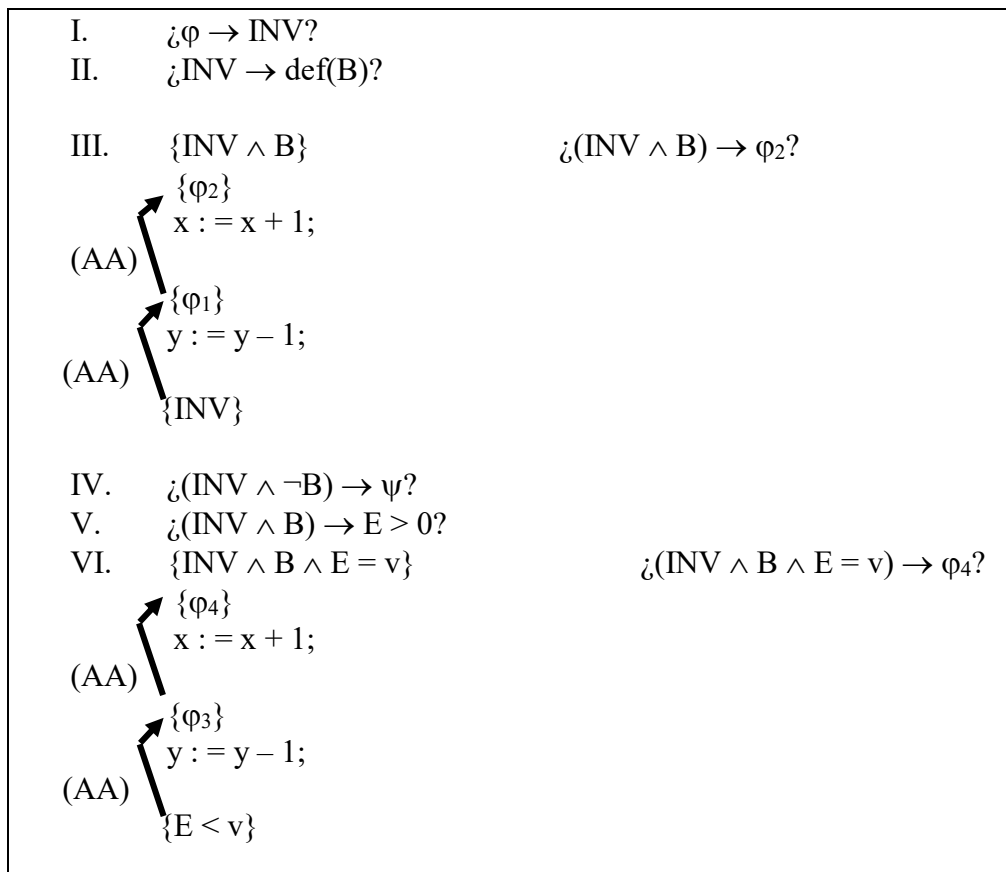
### 3.2.4.11. Ejemplo 20

¿Es correcto el siguiente programa?

$\{\varphi\} \equiv \{x = a \wedge y = b\}$ <b>while</b> $\{INV\}$ $y \neq 0$ <b>loop</b> $x := x + 1;$ $y := y - 1;$ <b>end loop;</b> $\{\psi\} \equiv \{x = a + b\}$
$\{INV\} \equiv \{x + y = a + b\}$ $E = y$

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones



I.  $\dot{?}\varphi \rightarrow \text{INV}?$ 

$$\begin{array}{c}
 \underbrace{\{x = a \wedge y = b\}} \\
 \alpha \qquad \qquad \beta \\
 \downarrow ? \\
 \underbrace{\{x + y = a + b\}} \\
 \text{por } \alpha \text{ y } \beta
 \end{array}$$

Es decir, suponiendo que  $x = a \wedge y = b$  es cierto, hay que mirar si  $x + y = a + b$  es cierto.

II.  $\dot{?}\text{INV} \rightarrow \text{def}(B)?$ 

$\dot{?}\text{INV} \rightarrow \text{true}$ ? Sí, porque la segunda parte de la implicación es true.

## III.

- $\{\varphi_1\} \equiv \{\text{def}(y - 1) \wedge (\text{INV})_y^{y-1}\} \equiv$   
 $\equiv \{\text{true} \wedge x + y - 1 = a + b\} \equiv$  simplificación  
 $\equiv \{x + y - 1 = a + b\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

- $\{\varphi_2\} \equiv \{\text{def}(x + 1) \wedge (\varphi_1)_x^{x+1}\} \equiv$   
 $\equiv \{\text{true} \wedge x + 1 + y - 1 = a + b\} \equiv$  simplificación  
 $\equiv \{x + y = a + b\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $x + 1 + y - 1 = a + b$ , como en la parte izquierda se suma 1 y se resta 1, se quitarían esa suma y esa resta quedando  $x + y = a + b$ .

- $\dot{?}(\text{INV} \wedge B) \rightarrow \varphi_2?$

$$\begin{array}{c}
 \underbrace{x + y = a + b} \wedge \underbrace{y \neq 0} \\
 \alpha \qquad \qquad \beta \\
 \downarrow ? \\
 \underbrace{x + y = a + b} \\
 \text{por } \alpha \text{ y } \delta
 \end{array}$$

IV.  $\dot{?}(\text{INV} \wedge \neg B) \rightarrow \psi?$

$$\begin{array}{c} \underbrace{x + y = a + b}_{\alpha} \wedge \underbrace{y = 0}_{\beta} \\ \downarrow ? \\ \underbrace{\{x = a + b\}}_{\text{por } \alpha \text{ y } \beta} \end{array}$$

En este caso al ser  $y = 0$ , sustituyendo  $y$  por  $0$  en  $\beta$  tenemos lo mismo que en  $\psi$ .

V.  $\dot{?}(\text{INV} \wedge B) \rightarrow E > 0?$

$$\begin{array}{c} \underbrace{x + y = a + b}_{\alpha} \wedge \underbrace{y \neq 0}_{\beta} \\ \downarrow ? \\ y > 0 \end{array}$$

No. Y por tanto no se puede probar la corrección total, sólo la parcial.

Ya no hace falta mirar el punto VI.

• **Demostración formal de la corrección parcial:**

I	1. $\varphi \rightarrow \text{INV}$
II	2. $\text{INV} \rightarrow \text{def}(B)$
III	3. $(\text{INV} \wedge B) \rightarrow \varphi_2$ 4. $\{\varphi_2\} x := x + 1; \{\varphi_1\} \text{ (AA)}$ 5. $\{\text{INV} \wedge B\} x := x + 1; \{\varphi_1\} \text{ (RCN 3, 4)}$ 6. $\{\varphi_1\} y := y - 1; \{\text{INV}\} \text{ (AA)}$ 7. $\{\text{INV} \wedge B\}$ $x := x + 1;$ $y := y - 1;$ $\{\text{INV}\} \text{ (RCP 5, 6)}$
IV	8. $(\text{INV} \wedge \neg B) \rightarrow \psi$  9. $\{\varphi\}$ <b><u>while</u></b> $\{\text{INV}\} i \neq n$ <b><u>loop</u></b> $x := x + 1;$ $y := y - 1;$ <b><u>end loop</u></b> ; $\{\psi\} \text{ (RWH 1, 2, 7, 8)}$

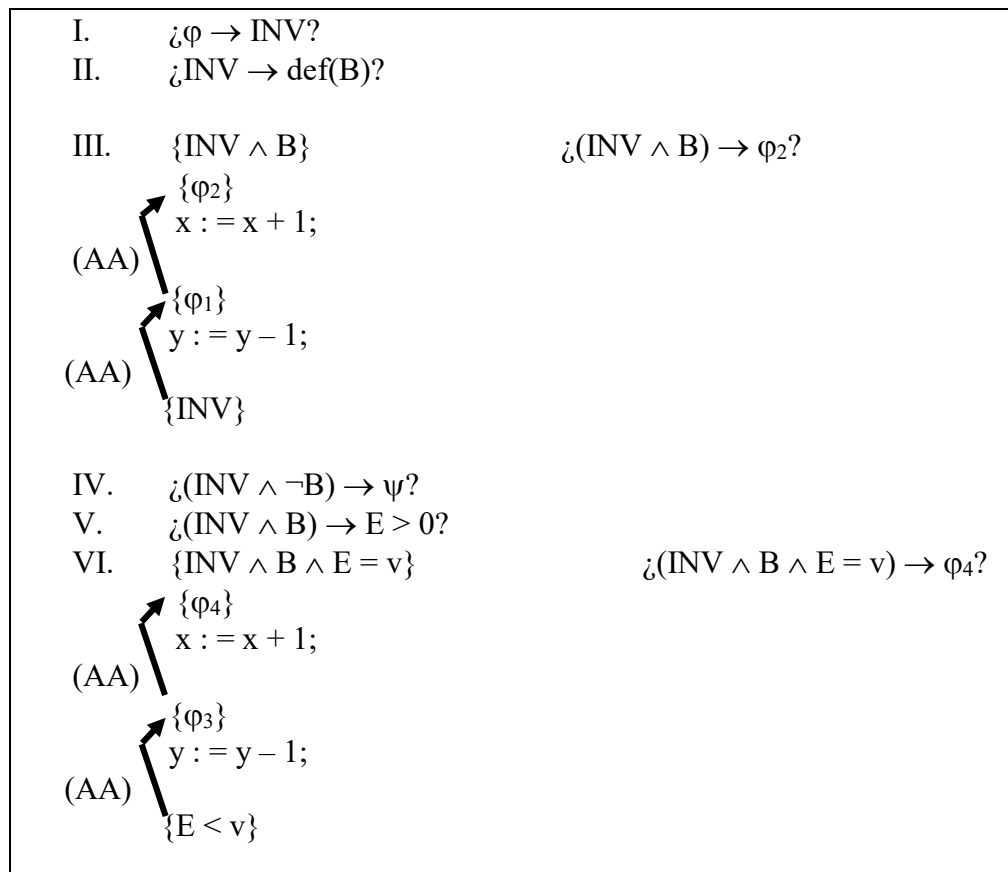
### 3.2.4.12. Ejemplo 21

¿Es correcto el siguiente programa?

$\{\varphi\} \equiv \{x = a \wedge y = b \geq 0\}$ <b>while</b> $\{INV\}$ $y \neq 0$ <b>loop</b> $x := x + 1;$ $y := y - 1;$ <b>end loop;</b> $\{\psi\} \equiv \{x = a + b\}$
$\{INV\} \equiv \{(0 \leq y \leq b) \wedge x + y = a + b\}$ $E = y$

#### Esquema:

- La aplicación de la Regla del While (RWH) supondrá realizar los siguientes cálculos y comprobaciones



**I.**  $\phi \rightarrow \text{INV}?$

$$\begin{array}{c}
 \underbrace{\{x = a\}}_{\alpha} \wedge \underbrace{\{y = b \geq 0\}}_{\beta} \\
 \downarrow ? \\
 \underbrace{\{(0 \leq y \leq b)\}}_{\text{por } \beta} \wedge \underbrace{\{x + y = a + b\}}_{\text{por } \alpha \text{ y } \beta}
 \end{array}$$

Es decir, suponiendo que  $x = a \wedge y = b \geq 0$  es cierto, hay mirar si  $(0 \leq y \leq b) \wedge x + y = a + b$  es cierto.

Sí se cumple, ya que cada componente de la segunda parte de la implicación se justifica mediante las componentes de la primera parte tal como se indica mediante las letras griegas  $\alpha$  y  $\beta$ .

**II.**  $\text{INV} \rightarrow \text{def}(B)?$

$\text{INV} \rightarrow \text{true}$ ? Sí, porque la segunda parte de la implicación es  $\text{true}$ .

**III.**

- $\{\phi_1\} \equiv \{\text{def}(y-1) \wedge (\text{INV})_y^{y-1}\} \equiv$   
 $\equiv \{\text{true} \wedge (0 \leq y-1 \leq b) \wedge x + y - 1 = a + b\} \equiv \text{simplificación}$   
 $\equiv \{(1 \leq y \leq b+1) \wedge x + y - 1 = a + b\}$

Se puede quitar  $\text{true}$  porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $(0 \leq y-1 \leq b)$ , lo mejor es ponerlo utilizando  $y$  en vez de  $y-1$ . Para ello hay que sumar 1 a los tres elementos:  $(0+1 \leq y-1+1 \leq b+1)$ . Tras realizar las operaciones queda  $(1 \leq y \leq b+1)$ .

- $\{\phi_2\} \equiv \{\text{def}(x+1) \wedge (\phi_1)_x^{x+1}\} \equiv$   
 $\equiv \{\text{true} \wedge (1 \leq y \leq b+1) \wedge x+1+y-1 = a+b\} \equiv \text{simplificación}$   
 $\equiv \{(1 \leq y \leq b+1) \wedge x+y = a+b\}$

Se puede quitar  $\text{true}$  porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

En cuanto a la fórmula  $x+1+y-1 = a+b$ , como en la parte izquierda se suma 1 y se resta 1, se quitarían esa suma y esa resta quedando  $x+y = a+b$ .

- $\dot{I}(\text{INV} \wedge B) \rightarrow \varphi_2?$

$$\begin{array}{c}
 \underbrace{(0 \leq y \leq b)}_{\alpha} \wedge \underbrace{x + y = a + b}_{\beta} \wedge \underbrace{y \neq 0}_{\delta} \\
 \downarrow? \\
 \underbrace{(1 \leq y \leq b + 1)}_{\text{por } \alpha \text{ y } \delta} \wedge \underbrace{x + y = a + b}_{\text{por } \beta}
 \end{array}$$

- IV.  $\dot{I}(\text{INV} \wedge \neg B) \rightarrow \psi?$

$$\begin{array}{c}
 \underbrace{(0 \leq y \leq b)}_{\alpha} \wedge \underbrace{x + y = a + b}_{\beta} \wedge \underbrace{y = 0}_{\delta} \\
 \downarrow? \\
 \underbrace{\{x = a + b\}}_{\text{por } \beta \text{ y } \delta}
 \end{array}$$

En este caso al ser  $y = 0$ , sustituyendo  $y$  por  $0$  en  $\beta$  tenemos lo mismo que en  $\psi$ .

- V.  $\dot{I}(\text{INV} \wedge B) \rightarrow E > 0?$

$$\begin{array}{c}
 \underbrace{(0 \leq y \leq b)}_{\alpha} \wedge \underbrace{x + y = a + b \wedge y \neq 0}_{\beta} \\
 \downarrow? \\
 \underbrace{y > 0}_{\text{por } \alpha \text{ y } \beta}
 \end{array}$$

Por  $\alpha$  sabemos que  $y \geq 0$  y por  $\beta$  sabemos que  $y \neq 0$ . Por tanto  $y > 0$ .

VI.

- $\{\varphi_3\} \equiv \{\text{def}(y - 1) \wedge (E < v)_{y^{y-1}}\} \equiv$   
 $\equiv \{\text{true} \wedge y - 1 < v\} \equiv \text{simplificación}$   
 $\equiv \{y - 1 < v\}$
- $\{\varphi_4\} \equiv \{\text{def}(x + 1) \wedge (\varphi_3)_{x^{x+1}}\} \equiv$   
 $\equiv \{\text{true} \wedge y - 1 < v\} \equiv \text{simplificación}$   
 $\equiv \{y - 1 < v\}$

Se puede quitar *true* porque para cualquier fórmula  $\delta$  se cumple  $\text{true} \wedge \delta \equiv \delta$ .

- ¿ $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$ ?

$$(0 \leq y \leq b) \wedge x + y = a + b \wedge y \neq 0 \wedge \underbrace{y = v}_{\alpha}$$

$$\begin{array}{c} \downarrow? \\ \underbrace{y - 1 < v} \\ \text{por } \alpha \end{array}$$

Si  $y$  es igual a  $v$ ,  $y - 1$  será  $v - 1$  y, por tanto, menor que  $v$ .

- **Demostración formal de la corrección total:**

I	1. $\varphi \rightarrow INV$
II	2. $INV \rightarrow \text{def}(B)$
III	3. $(INV \wedge B) \rightarrow \varphi_2$ 4. $\{\varphi_2\} x := x + 1; \{\varphi_1\}$ (AA) 5. $\{INV \wedge B\} x := x + 1; \{\varphi_1\}$ (RCN 3, 4) 6. $\{\varphi_1\} y := y - 1; \{INV\}$ (AA) 7. $\{INV \wedge B\}$ $x := x + 1;$ $y := y - 1;$ $\{INV\}$ (RCP 5, 6)
IV	8. $(INV \wedge \neg B) \rightarrow \psi$
V	9. $(INV \wedge B) \rightarrow E > 0$
VI	10. $(INV \wedge B \wedge E = v) \rightarrow \varphi_4$ 11. $\{\varphi_4\} x := x + 1; \{\varphi_3\}$ (AA) 12. $\{INV \wedge B \wedge E = v\} x := x + 1; \{\varphi_3\}$ (RCN 10, 11) 13. $\{\varphi_3\} y := y - 1; \{E < v\}$ (AA) 14. $\{INV \wedge B \wedge E = v\}$ $x := x + 1;$ $y := y - 1;$ $\{E < v\}$ (RCP 12, 13)
	15. $\{\varphi\}$ <b>while</b> $\{INV\}$ $i \neq n$ <b>loop</b> $x := x + 1;$ $y := y - 1;$ <b>end loop;</b> $\{\psi\}$ (RWH 1, 2, 7, 8, 9, 14)