METODOLOGÍA DE LA PROGRAMACIÓN
EJERCICIOS DEL TEMA 4
DERIVACIÓN DE PROGRAMAS

Derivación formal de iteraciones:

1) Calcular la suma de los elementos de un vector (1ª versión)

Derivar formalmente un programa que, dado un vector A(1..n), calcula en la variable *s* la suma de los elementos de A(1..n). El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {s = $\sum_{k=1}^{n} A(k)$ }
{INV} ≡ {(1 ≤ i ≤ n + 1) ∧ s = $\sum_{k=1}^{i-1} A(k)$ } E = n + 1 – i

2) Calcular la suma de los elementos de un vector (2ª versión)

Derivar formalmente un programa que, dado un vector A(1..n), calcula en la variable *s* la suma de los elementos de A(1..n). El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {s = $\sum_{k=1}^{n} A(k)$ }
{INV} ≡ {(0 ≤ i ≤ n) ∧ s = $\sum_{k=1}^{i} A(k)$ } E = n – i

3) Calcular la suma de los elementos de un vector (3ª versión)

Derivar formalmente un programa que, dado un vector A(1..n), calcula en la variable *s* la suma de los elementos de A(1..n). El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{s = \sum_{k=1}^{n} A(k)\}$
$\{INV\} \equiv \{(1 \leq i \leq n) \wedge s = \sum_{k=1}^{i} A(k)\}$ $E = n - i$

4) Calcular la suma de los elementos de un vector (4ª versión)

Derivar formalmente un programa que, dado un vector A(1..n), calcula en la variable *s* la suma de los elementos de A(1..n). El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{s = \sum_{k=1}^{n} A(k)\}$
$\{INV\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^{n} A(k)\}$ $E = i$

5) Calcular la suma de los elementos de un vector (5ª versión)

Derivar formalmente un programa que, dado un vector A(1..n), calcula en la variable *s* la suma de los elementos de A(1..n). El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {s = $\sum_{k=1}^{n} A(k)$ }
{INV} ≡ {(1 ≤ i ≤ n) ∧ s = $\sum_{k=i}^{n} A(k)$ } E = i – 1

6) Devolver 0 en c si y solamente si A(1..n) contiene sólo ceros (1ª versión)

Derivar formalmente un programa que, dado un vector A(1..n) que sólo contiene números no negativos, devuelve un 0 en la variable entera *c* si A(1..n) contiene sólo ceros y devuelve algo distinto de 0 si en A(1..n) no todos son ceros. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1 ∧ noneg(A(1..n))} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {(c = 0) ↔ ∀k(1 ≤ k ≤ n → A(k) = 0)}
{INV} ≡ {noneg(A(1..n)) ∧ (1 ≤ i ≤ n + 1) ∧ c = $\sum_{k=1}^{i-1} A(k)$ } E = n + 1 – i noneg(A(1..n)) ≡ {∀k(1 ≤ k ≤ n → A(k) ≥ 0)}

7) Devolver 0 en c si y solamente si A(1..n) contiene sólo ceros (2ª versión)

Derivar formalmente un programa que, dado un vector A(1..n) que sólo contiene números no negativos, devuelve un 0 en la variable entera *c* si A(1..n) contiene sólo ceros y devuelve algo distinto de 0 si en A(1..n) no todos son ceros. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1 \wedge \text{noneg}(A(1..n))\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{(c = 0) \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = 0)\}$
$\{INV\} \equiv \{\text{noneg}(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^{i} A(k)\}$ $E = n - i$ $\text{noneg}(A(1..n)) \equiv \{\forall k(1 \leq k \leq n \rightarrow A(k) \geq 0)\}$

8) Devolver 0 en c si y solamente si A(1..n) contiene sólo ceros (3ª versión)

Derivar formalmente un programa que, dado un vector A(1..n) que sólo contiene números no negativos, devuelve un 0 en la variable entera *c* si A(1..n) contiene sólo ceros y devuelve algo distinto de 0 si en A(1..n) no todos son ceros. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1 \wedge \text{noneg}(A(1..n))\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{(c = 0) \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = 0)\}$
$\{INV\} \equiv \{\text{noneg}(A(1..n)) \wedge (1 \leq i \leq n) \wedge c = \sum_{k=1}^{i} A(k)\}$ $E = n - i$ $\text{noneg}(A(1..n)) \equiv \{\forall k(1 \leq k \leq n \rightarrow A(k) \geq 0)\}$

9) Devolver 0 en c si y solamente si A(1..n) contiene al menos un 0 (1ª versión)

Derivar formalmente un programa que, dado un vector A(1..n) de enteros, devuelve un 0 en la variable entera *c* si A(1..n) contiene por lo menos un cero y devuelve algo distinto de 0 si en A(1..n) no hay ningún cero. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {(c = 0) ↔ ∃k(1 ≤ k ≤ n ∧ A(k) = 0)}
{INV} ≡ {(1 ≤ i ≤ n + 1) ∧ c = $\prod_{k=1}^{i-1} A(k)$ } E = n + 1 – i

10) Devolver 0 en c si y solamente si A(1..n) contiene al menos un 0 (2ª versión)

Derivar formalmente un programa que, dado un vector A(1..n) de enteros, devuelve un 0 en la variable entera *c* si A(1..n) contiene por lo menos un cero y devuelve algo distinto de 0 si en A(1..n) no hay ningún cero. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {(c = 0) ↔ ∃k(1 ≤ k ≤ n ∧ A(k) = 0)}
{INV} ≡ {(0 ≤ i ≤ n) ∧ c = $\prod_{k=1}^{i} A(k)$ } E = n – i

11) Devolver 0 en c si y solamente si A(1..n) contiene al menos un 0 (3ª versión)

Derivar formalmente un programa que, dado un vector A(1..n) de enteros, devuelve un 0 en la variable entera *c* si A(1..n) contiene por lo menos un cero y devuelve algo distinto de 0 si en A(1..n) no hay ningún cero. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {(c = 0) ↔ ∃k(1 ≤ k ≤ n ∧ A(k) = 0)}
{INV} ≡ {(1 ≤ i ≤ n) ∧ c = $\prod_{k=1}^{i} A(k)$ } E = n – i

12) Decidir si dos vectores son iguales (1ª versión)

Derivar formalmente un programa que, dados dos vectores A(1..n) y B(1..n), decide en la variable booleana *c* si A(1..n) y B(1..n) son iguales. El programa derivado deberá ser eficiente en el sentido de que al recorrer los vectores, si en un momento se detecta que la respuesta va a ser negativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

{φ} ≡ {n ≥ 1} ¿Inicializaciones? {INV} **while** {INV} {E} B **loop**   ¿Instrucciones? **end loop;** {ψ} ≡ {c ↔ ∀k(1 ≤ k ≤ n → A(k) = B(k))}
{INV} ≡ {(1 ≤ i ≤ n + 1) ∧ (c ↔ ∀k(1 ≤ k ≤ i – 1 → A(k) = B(k)))} E = n + 1 – i

13) Decidir si dos vectores son iguales (2ª versión)

Derivar formalmente un programa que, dados dos vectores A(1..n) y B(1..n), decide en la variable booleana *c* si A(1..n) y B(1..n) son iguales. El programa derivado deberá ser eficiente en el sentido de que al recorrer los vectores, si en un momento se detecta que la respuesta va a ser negativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = B(k))\}$
$\{INV\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))\}$ $E = n - i$

14) Decidir si dos vectores son iguales (3ª versión)

Derivar formalmente un programa que, dados dos vectores A(1..n) y B(1..n), decide en la variable booleana *c* si A(1..n) y B(1..n) son iguales. El programa derivado deberá ser eficiente en el sentido de que al recorrer los vectores, si en un momento se detecta que la respuesta va a ser negativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = B(k))\}$
$\{INV\} \equiv \{(1 \leq i \leq n) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))\}$ $E = n - i$

15) Decidir si dos vectores son iguales (4ª versión)

Derivar formalmente un programa que, dados dos vectores A(1..n) y B(1..n), decide en la variable booleana *c* si A(1..n) y B(1..n) son iguales. El programa derivado deberá ser eficiente en el sentido de que al recorrer los vectores, si en un momento se detecta que la respuesta va a ser negativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = B(k))\}$
$\{INV\} \equiv \{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \forall k(i \leq k \leq n \rightarrow A(k) = B(k)))\}$ $E = i - 1$

16) Decidir si dos vectores son iguales (5ª versión)

Derivar formalmente un programa que, dados dos vectores A(1..n) y B(1..n), decide en la variable booleana *c* si A(1..n) y B(1..n) son iguales. El programa derivado deberá ser eficiente en el sentido de que al recorrer los vectores, si en un momento se detecta que la respuesta va a ser negativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = B(k))\}$
$\{INV\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \forall k(i + 1 \leq k \leq n \rightarrow A(k) = B(k)))\}$ $E = i$

17) Decidir si x está en A(1..n) (1ª versión)

Derivar formalmente un programa que, dados un número x y un vector A(1..n), decide en la variable booleana *c* si x aparece en A(1..n). El programa derivado deberá ser eficiente en el sentido de que al recorrer el vector, si en un momento se detecta que la respuesta va a ser afirmativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = x)\}$
$\{INV\} \equiv \{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i - 1 \wedge A(k) = x))\}$ $E = n + 1 - i$

18) Decidir si x está en A(1..n) (2ª versión)

Derivar formalmente un programa que, dados un número x y un vector A(1..n), decide en la variable booleana *c* si x aparece en A(1..n). El programa derivado deberá ser eficiente en el sentido de que al recorrer el vector, si en un momento se detecta que la respuesta va a ser afirmativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = x)\}$
$\{INV\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))\}$ $E = n - i$

19) Decidir si x está en A(1..n) (3ª versión)

Derivar formalmente un programa que, dados un número x y un vector A(1..n), decide en la variable booleana *c* si x aparece en A(1..n). El programa derivado deberá ser eficiente en el sentido de que al recorrer el vector, si en un momento se detecta que la respuesta va a ser afirmativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = x)\}$
$\{INV\} \equiv \{(1 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))\}$ $E = n - i$

20) Decidir si x está en A(1..n) (4ª versión)

Derivar formalmente un programa que, dados un número x y un vector A(1..n), decide en la variable booleana *c* si x aparece en A(1..n). El programa derivado deberá ser eficiente en el sentido de que al recorrer el vector, si en un momento se detecta que la respuesta va a ser afirmativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$ ¿Inicializaciones? {INV} **while** {INV} {E} ¿B? **loop**   ¿Instrucciones? **end loop;** $\{\psi\} \equiv \{c \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = x)\}$
$\{INV\} \equiv \{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \exists k(i \leq k \leq n \wedge A(k) = x))\}$ $E = i - 1$

21) Decidir si x está en A(1..n) (5ª versión)

Derivar formalmente un programa que, dados un número x y un vector A(1..n), decide en la variable booleana *c* si x aparece en A(1..n). El programa derivado deberá ser eficiente en el sentido de que al recorrer el vector, si en un momento se detecta que la respuesta va a ser afirmativa el while deberá terminar sin analizar las posiciones restantes. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$
¿Inicializaciones?
{INV}
while {INV} {E} ¿B? **loop**
  ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{c \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = x)\}$

$\{INV\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(i + 1 \leq k \leq n \wedge A(k) = x))\}$
$E = i$

22) Decidir en p si todos los elementos de A(1..n) son pares (Junio 2008)

Derivar, utilizando la regla del while y el axioma de la asignación del Cálculo de Hoare, un programa que decide en la variable booleana p si todos los elementos de A(1..n) son pares. El programa ha de ser eficiente en el sentido de que si se detecta algún elemento que no es par, el while termine en ese momento ya que se conoce la respuesta final. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$
¿Inicializaciones?
while {INV} ¿B? **loop**
  ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{p \leftrightarrow \forall k(1 \leq k \leq n \rightarrow par(A(k)))\}$

$\{INV\} \equiv \{(0 \leq i \leq n) \wedge p \leftrightarrow \forall k(1 \leq k \leq i \rightarrow par\ (A(k)))\}$

$E = n - i$

$par(x) \equiv \{x \textbf{ mod } 2 = 0\}$

23) Decidir en hp si A(1..n) contiene por lo menos un elemento par (Septiembre 2008)

Derivar, utilizando la regla del while y el axioma de la asignación del Cálculo de Hoare, un programa que decide en la variable booleana hp si hay algún elemento par en A(1..n). El programa ha de ser eficiente en el sentido de que si se encuentra un elemento par el while termine en ese momento ya que se conoce la respuesta final. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1\}$
¿Inicializaciones?
while {INV} ¿B? **loop**
    ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{hp \leftrightarrow \exists k(1 \leq k \leq n \wedge par(A(k)))\}$

$\{INV\} \equiv \{(1 \leq i \leq n + 1) \wedge hp \leftrightarrow \exists k(i \leq k \leq n \wedge par(A(k)))\}$

$E = i - 1$

$par(x) \equiv \{x \textbf{ mod } 2 = 0\}$

24) Devolver 0 en c si A(1..n) no contiene ningún 1 y no devolver 0 en caso contrario (Junio 2009)

Derivar, utilizando la regla del while y el axioma de la asignación del **Cálculo de Hoare**, un programa que dado un vector A(1..n) que sólo contiene ceros y unos, devuelve un 0 en la variable entera c si en A(1..n) el número de posiciones que contienen el valor 1 es 0. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1 \wedge bits(A(1..n))\}$
¿Inicializaciones?
{INV}
while {INV} ¿B? **loop**
    ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{(c = 0) \leftrightarrow (Nk(1 \leq k \leq n \wedge A(k) = 1) = 0)\}$

$\{INV\} \equiv \{bits(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^{i} A(k)\}$

$E = n - i$

$bits(A(1..n)) \equiv \{\forall k(1 \leq k \leq n \rightarrow (A(k) = 0 \vee A(k) = 1))\}$

25) Devolver el valor n en c si A(1..n) no contiene ningún 0 y no devolver n en caso contrario (Septiembre 2009)

Derivar, utilizando la regla del while y el axioma de la asignación del **Cálculo de Hoare**, un programa que dado un vector A(1..n) que sólo contiene ceros y unos, devuelve el valor n en la variable entera c si en A(1..n) el número de posiciones que contienen el valor 0 es 0. El programa derivado ha de ser correcto con respecto a φ, ψ, INV y E.

$\{\varphi\} \equiv \{n \geq 1 \wedge \text{bits}(A(1..n))\}$
¿Inicializaciones?
{INV}
while {INV} ¿B? **loop**
    ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{(c = n) \leftrightarrow (Nk(1 \leq k \leq n \wedge A(k) = 0) = 0)\}$

$\{\text{INV}\} \equiv \{\text{bits}(A(1..n)) \wedge (1 \leq i \leq n) \wedge c = \sum_{k=1}^{i} A(k)\}$
$E = n - i$

$\text{bits}(A(1..n)) \equiv \{\forall k(1 \leq k \leq n \rightarrow (A(k) = 0 \vee A(k) = 1))\}$

26) Decidir en z si A(1..n) y H(1..n) coinciden en alguna posición (Junio 2010)

Derivar, utilizando la regla del while y el axioma de la asignación del **Cálculo de Hoare**, un programa que, dados los vectores de enteros A(1..n) y H(1..n), decide en la variable booleana z si para alguna posición comprendida entre 1 y n los vectores A(1..n) y H(1..n) tienen el mismo valor. El programa obtenido ha de ser eficiente en el sentido de que si en algún momento se detecta que la respuesta va a ser afirmativa, el programa ha de parar sin analizar las posiciones restantes.

$\{\varphi\} \equiv \{n \geq 1\}$
¿Inicializaciones?
{INV}
while {INV} ¿B? **loop**
    ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{z \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = H(k))\}$

$\{\text{INV}\} \equiv \{(0 \leq i \leq n) \wedge z \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = H(k))\}$
$E = n - i$

27) Decidir en w si alguna posición de C(1..n) coincide con la suma de los elementos de esa misma posición de A(1..n) y B(1..n) (Septiembre 2010)

Derivar, utilizando la regla del while y el axioma de la asignación del **Cálculo de Hoare**, un programa que, dados los vectores de enteros A(1..n), B(1..n) y C(1..n), decide en la variable booleana w si el elemento de alguna posición del vector C(1..n) es igual a la suma de los elementos de esa misma posición en los vectores A(1..n) y B(1..n). El programa obtenido ha de ser eficiente en el sentido de que si en algún momento se detecta que la respuesta va a ser afirmativa, el programa ha de parar sin analizar las posiciones restantes.

$\{\varphi\} \equiv \{n \geq 1\}$
¿Inicializaciones?
{INV}
while {INV} ¿B? **loop**
    ¿Instrucciones?
end loop;
$\{\psi\} \equiv \{w \leftrightarrow \exists k(1 \leq k \leq n \wedge C(k) = A(k) + B(k))\}$

$\{INV\} \equiv \{(1 \leq i \leq n) \wedge w \leftrightarrow \exists k(1 \leq k \leq i \wedge C(k) = A(k) + B(k))\}$
$E = n - i$