

TEMA 4: Diseño iterativo y noción de secuencia

4.1 Tratamiento secuencial

- Definición de secuencia
- Definición de una máquina secuencial
- Primer modelo de acceso secuencial
- Análisis iterativo

4.2 Esquemas algorítmicos de recorrido

4.3 Esquemas algorítmicos de búsqueda

4.4 Combinación de esquemas de recorrido y búsqueda

4.5 Generalización de los modelos de acceso secuencial

4.1 Definición de secuencia

- **Secuencia:** Conjunto de valores de cualquier tipo. Es significativo el orden entre ellos:
Secu1=[32,64,85] Secu2=['h', 'o', 'l', 'a'] Secu3=[]
- **Longitud** de una secuencia (*long*) es el número de elementos que la componen:

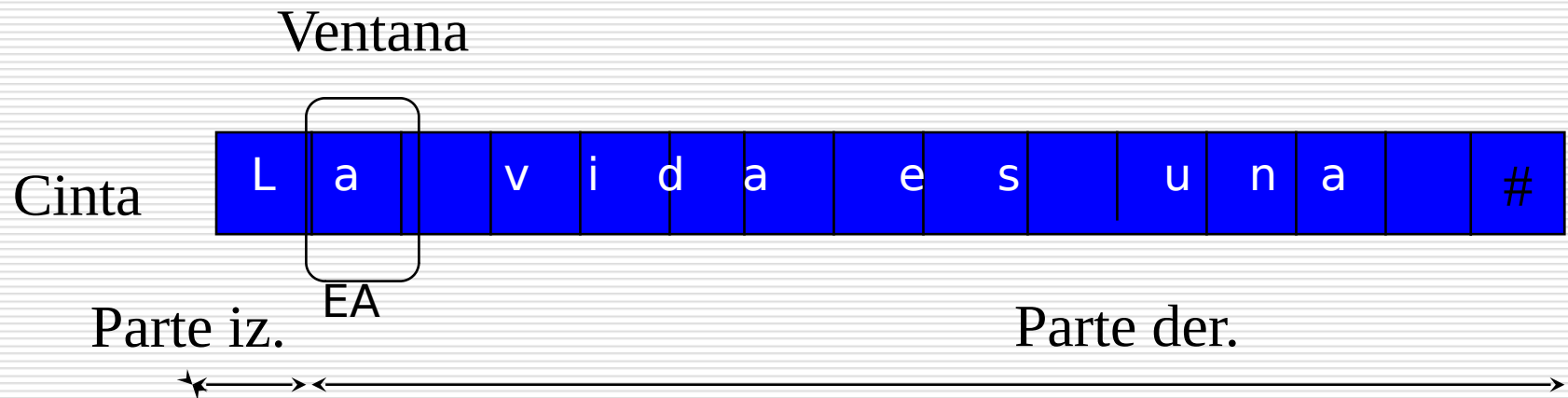
$$S = S_1S_2S_3\dots S_n \quad \text{long}(S)=n$$

La longitud de una secuencia no se conoce a priori

$$\text{long}(\text{Secu1})=3 \quad \text{long}(\text{Secu2})=4 \quad \text{long}(\text{Secu3})=0$$

- Existe una **relación de orden** $[32,64,85] \neq [85,64,32]$
-

4.1 Definición de una máquina secuencial



4.1 Primitivas para el primer modelo de acceso secuencial

Acceso Secuencial

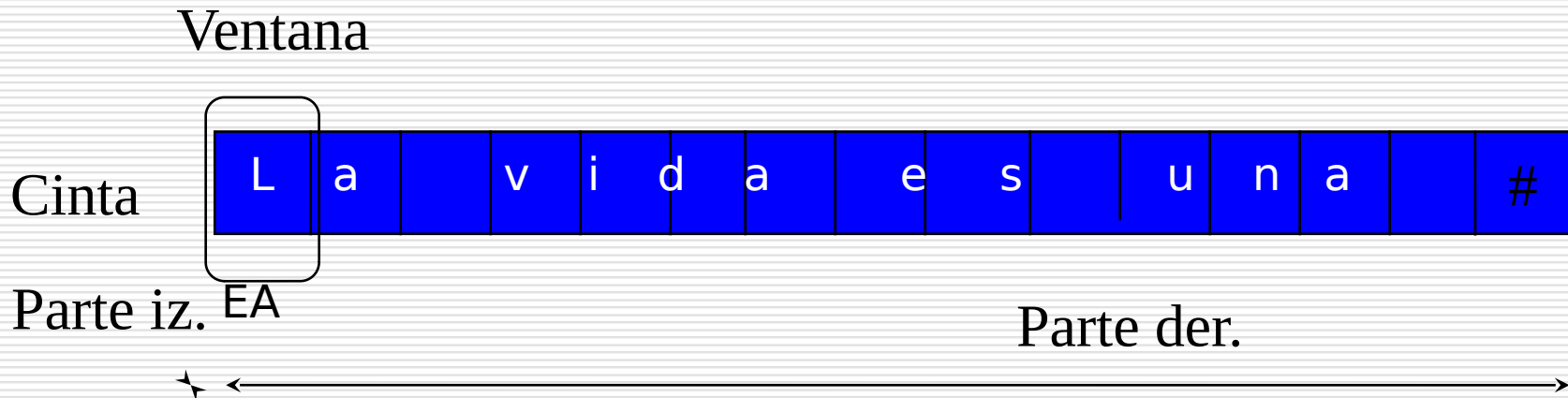
Comenzar (S)	Inicia una consulta sobre una secuencia S y su primer elemento es el elemento actual
Avanzar (S)	Avanza al siguiente elemento de S
EA (S)	Retorna el elemento actual de S

Creación Secuencial

Crear (S)	Crea S como una secuencia vacía sobre la que podemos añadir elementos por la derecha
Registrar (S, e)	Graba el elemento e como último elemento de una secuencia S
Marcar (S)	Graba la marca de fin en la secuencia S

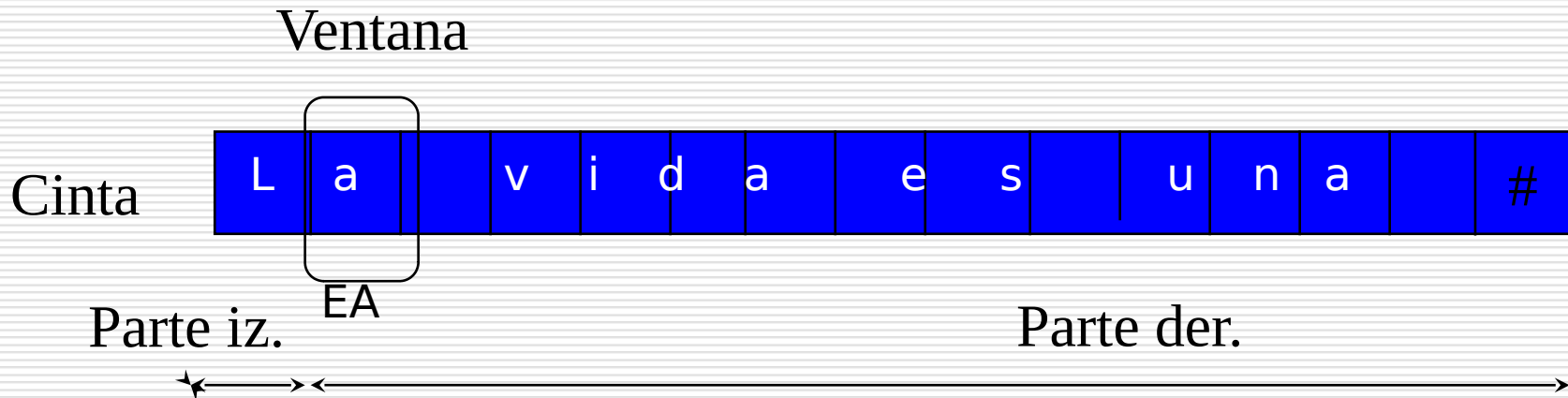
4.1 Primitivas para el primer modelo de acceso secuencial

Comenzar (S)



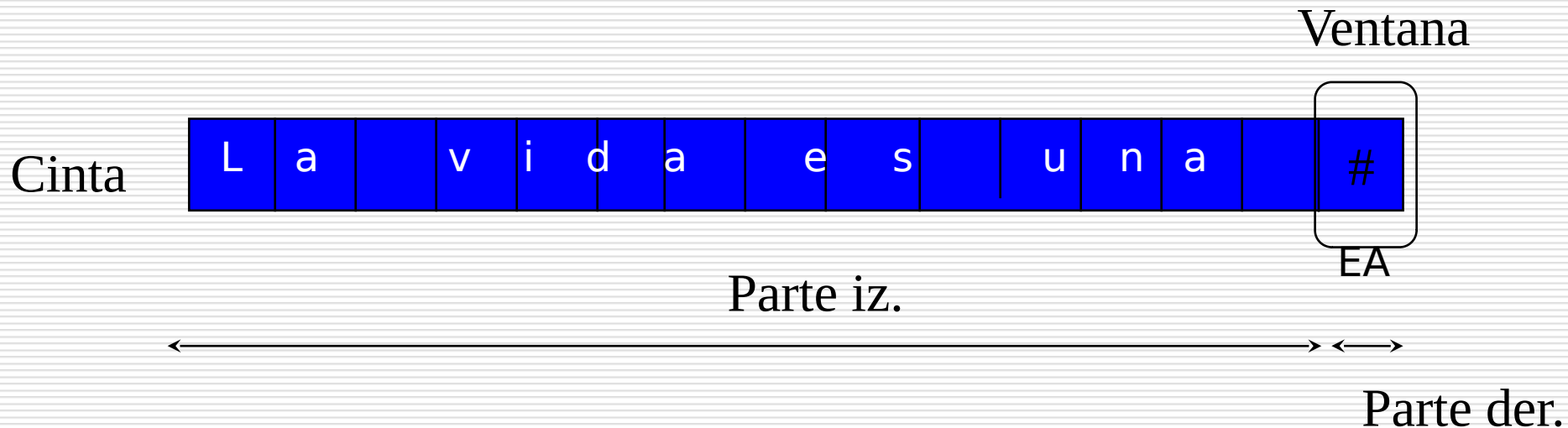
4.1 Primitivas para el primer modelo de acceso secuencial

Avanzar (S)

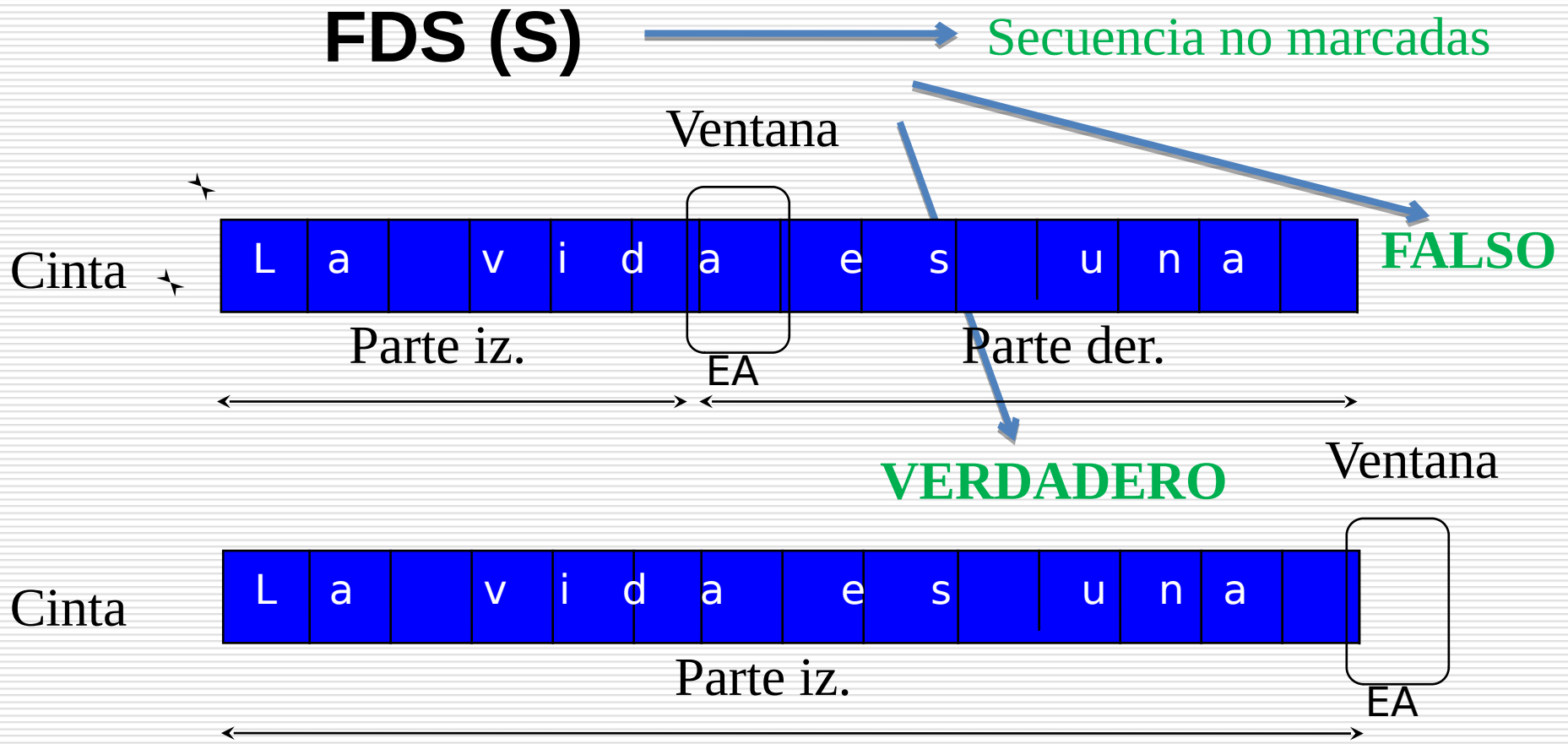


4.1 Primitivas para el primer modelo de acceso secuencial

$EA(S) = \text{MarcaFin}$ \longrightarrow Secuencia marcadas

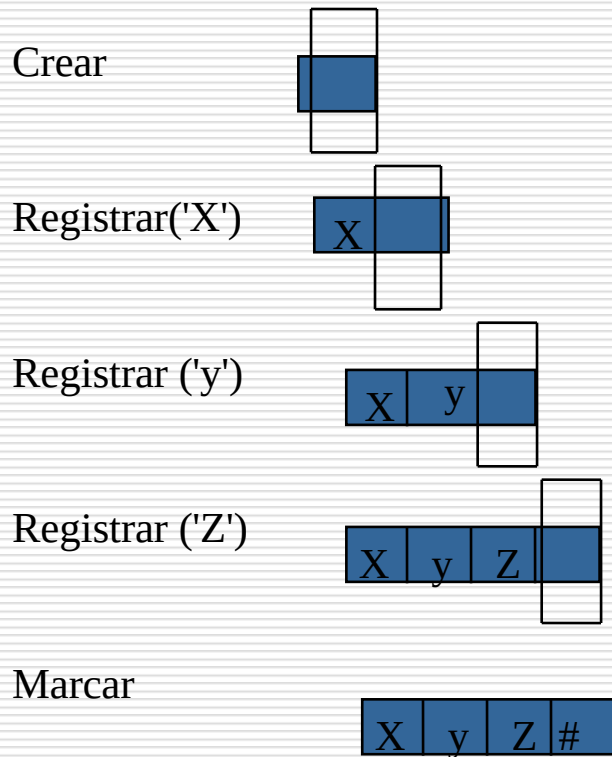


4.1 Primitivas para el primer modelo de acceso secuencial

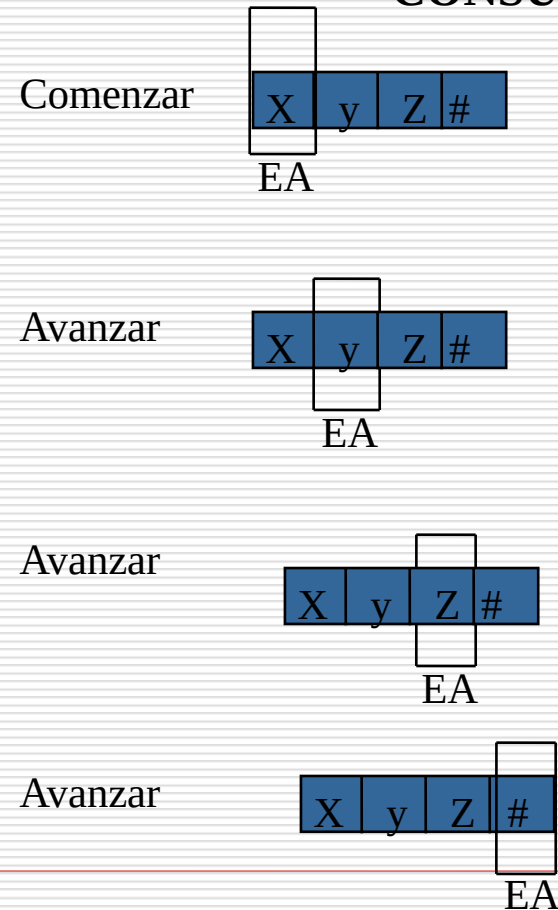


4.1 Primitivas para el primer modelo de acceso secuencial

CREACIÓN



CONSULTA

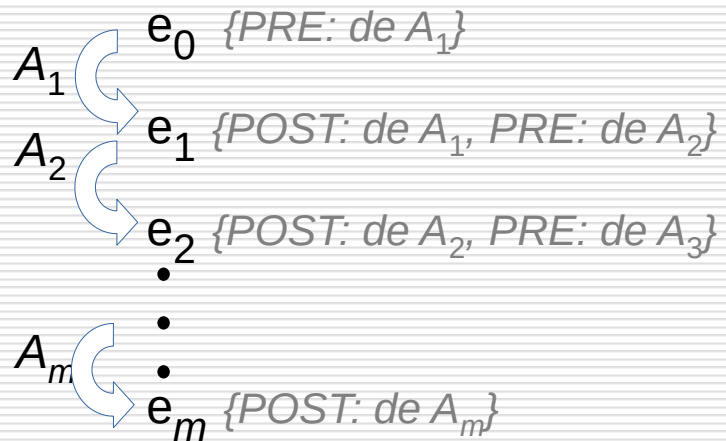


4.1 Primitivas para el primer modelo de acceso secuencial: operaciones permitidas en cada estado

	ninguno	creación	marcada	consulta
Comenzar	×	×	consulta	consulta
Avanzar	×	×	×	consulta
EA	×	×	×	consulta
Crear	creación	creación	creación	creación
Registrar	×	creación	×	×
Marcar	×	marcada	×	×

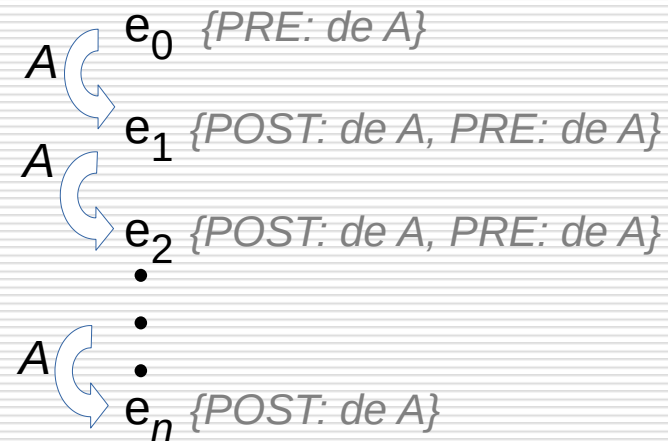
4.1 Análisis iterativo

Comp. secuencial:



- El número m de estados intermedios es conocido y fijado en el diseño
- Las acciones que llevan de un estado al siguiente son distintas

Comp. iterativa:



- El número n de estados intermedios es variable y, en general, desconocido a priori
- La acción que lleva de un estado al siguiente siempre es la misma

4.1 Análisis iterativo

- Diseñaremos algoritmos iterativos modelando la iteración como el tratamiento de una secuencia. El número de estados intermedios dependerá de la longitud de la secuencia, que en general será desconocida a priori.

- **CUESTIÓN:**

- Cualesquiera estados intermedios e_k y e_{k+1} son, al mismo tiempo, precondition y postcondición de la misma acción elemental A .
 - Si e_k y e_{k+1} son pre y postcondición de la misma acción, ¿significa que son el mismo estado? ¿¿El proceso iterativo no avanza??

- **RESPUESTA:**

Los estados intermedios **no** son el mismo, pero todos comparten una propiedad común que les hace ser pre y postcondición a la vez de A . Esta propiedad es la que caracteriza a todo diseño iterativo, y se llama:

INVARIANTE

4.1 Análisis iterativo

Veámoslo a través de un ejemplo: Determinar la longitud de un texto representado como una secuencia de caracteres.

Léxico

conta : entero ≥ 0 ;

S : secuencia de caracteres;

Algoritmo

Comenzar (S); conta $\leftarrow 0$; // Acciones de inicialización

MIENTRAS EA (S) \neq MarcaFin **HACER**

 conta \leftarrow conta + 1; Avanzar (S) // Cuerpo

FIN_MIENTRAS;

 Escribir (conta) // Finalización

Fin.

Ejemplo: Determinar la longitud de un texto dado

e s t o e s u n e j e m #

Para determinar la longitud del texto, ¿cuál es el tratamiento elemental que debemos aplicar a cada elemento? ¿Qué es lo que se cumple en el estado inicial, en los intermedios y en el estado final?

e_0 : EA = primero (S)

conta = 0 y EA = 'e'

e_1 : **T** aplicado sobre S_1 y EA = S_2

conta = 1 y EA = 's'

...

e_k : **T** aplicado sobre S_1, S_2, \dots, S_k y \Rightarrow
EA = S_{k+1}

...

conta = k y EA = 'e' k = 13

...

e_n : **T** aplicado sobre S y
EA = MarcaFin

...

conta = 15 y EA = MarcaFin

**INVARIANTE: cont = número de
elementos de la parte izquierda de S**

4.1 Análisis iterativo

- En un esquema algorítmico iterativo hay que completar los fragmentos de código dependientes del problema: **inicialización** y **tratamiento elemental** en el cuerpo del bucle
- Cada variable tendrá una inicialización y un tratamiento elemental en el cuerpo del bucle

En el ejemplo: Determinar la longitud de un texto dado

Léxico

conta : entero ≥ 0 ;

S : secuencia de caracteres;

Algoritmo

Comenzar (S); $\text{conta} \leftarrow 0$; // Acciones de inicialización

MIENTRAS EA (S) \neq MarcaFin **HACER**

$\text{conta} \leftarrow \text{conta} + 1$; Avanzar (S) // Cuerpo

FIN_MIENTRAS;

Escribir (conta) // Finalización

Fin.

4.1 Análisis iterativo

¿Cómo obtenemos la inicialización?

- Para cada variable v debemos responder a la pregunta. ¿Cuál es la solución para una secuencia vacía?
 - $v \leftarrow$ valor inicial

¿Cómo obtenemos el tratamiento elemental en el cuerpo del bucle?

- Suponemos que conocemos la solución para subsecuencia formada por la parte izquierda de S (es decir, los elementos que ya hemos recorrido y tratado). **Este valor es conocido en ese punto de la ejecución y está en la variable v -> Hipótesis de inducción**
 - ¿Qué tratamiento se debe realizar para tener ahora la solución de la subsecuencia formada la parte izquierda de S y el elemento actual?
 - $v_i \leftarrow g_i(v_i, EA)$
-

Problema: cálculo del valor medio del conjunto de notas de una asignatura, las notas se introducen como una secuencia de reales, con marca de fin.

➤ **Especificación**

PRE { S es una secuencia de reales }

POST { $i = \text{Long}(S)$ y $\text{suma} = \sum S_i$, y $\text{numElem} = \text{Long}(S)$ }

- **Diseño.** Realizar un *razonamiento inductivo* para ver cómo inicializar y actualizar las variables, de modo que se mantenga el invariante cuando se avance un elemento.

Inicialización

$\text{suma} \leftarrow 0$
 $\text{numElem} \leftarrow 0$

Inicialización
(casos base)

Tratamiento elemental

$\text{suma} \leftarrow \text{suma} + \text{EA}(S)$
 $\text{numElem} \leftarrow \text{numElem} + 1$

Cuerpo
(relaciones de inducción)

Problema: cálculo del valor medio del conjunto de notas de una asignatura, las notas se introducen como una secuencia de reales, con marca de fin.

LÉXICO

S : Secuencia de Real;

suma : Real; // lleva cuenta de la suma total de las notas

numElem : Entero; // lleva cuenta del número de notas

ALGORITMO

Comenzar (S); suma \leftarrow 0; numElem \leftarrow 0;

MIENTRAS EA (S) \neq MarcaFin **HACER**

 suma \leftarrow suma + EA (S); numElem \leftarrow numElem + 1;

 Avanzar (S)

FIN_MIENTRAS;

SI numElem > 0 **ENTONCES**

 Escribir ("Valor medio de las notas: ", suma/numElem)

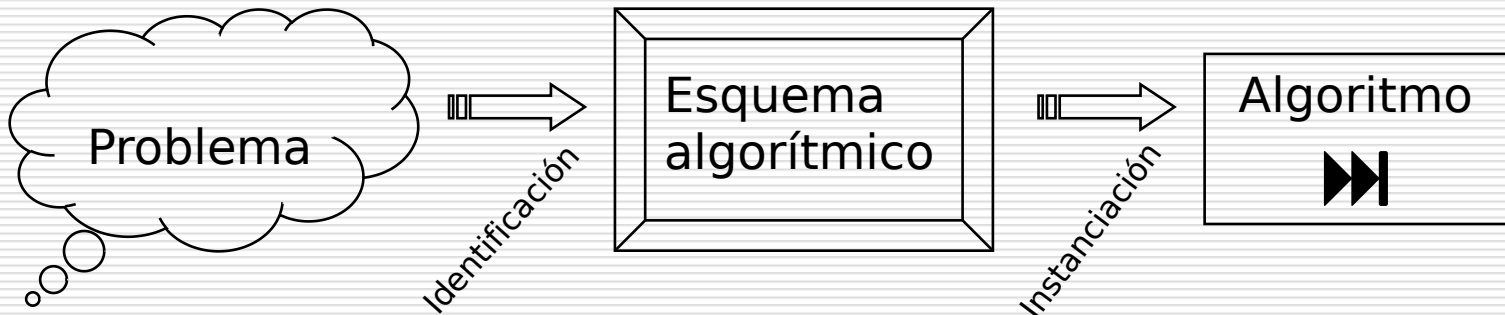
SI_NO Escribir ("Secuencia vacía")

FIN_SI

FIN.

4.2 Esquemas algorítmicos de recorrido

- **Esquema algorítmico:** plantilla de algoritmo aplicable no a un problema, sino a una *clase* de problemas.



- **Recorrido, o enumeración secuencial:** Aplicación de un mismo tratamiento a todos los elementos de la colección. En cada caso debemos estudiar si podemos:
 - a) Tratar la secuencia vacía como el resto de secuencias (H1)
 - b) Tratar el primer elemento como al resto de elementos (H2)
-

4.2 Esquemas algorítmicos de recorrido

- Estudiaremos tres esquemas algorítmicos de recorrido para secuencias marcadas:
 - **Esquema 1:** Tratamiento integrado de la secuencia vacía y del primer elemento en el caso general ($H1$ y $H2$)
 - **Esquema 2:** Tratamiento especial de la secuencia vacía y tratamiento integrado del primer elemento en el caso general ($\neg H1$ y $H2$)
 - **Esquema 3:** Tratamiento especial de la secuencia vacía y del primer elemento ($\neg H1$ y $\neg H2$)
 - Ilustraremos estos esquemas algoritmos empleando el problema del cálculo de la media de una secuencia de valores reales positivos.
-

4.2 Esquemas algorítmicos de recorrido

Esquema 1: H1 y H2

ESQUEMA ALGORÍTMICO:

Comenzar;

*{ Inicialización del tratamiento }**

MIENTRAS EA \neq MarcaFin **HACER**

*{ Tratamiento de EA }**

Avanzar

FIN_MIENTRAS;

*{ Terminación del tratamiento }**

* : Instanciar un esquema supone identificar y particularizar estas acciones al caso concreto

Léxico

S : secuencia de Real;

sum, media : Real; n : Entero;

Algoritmo

Comenzar (S);

sum \leftarrow 0.0; n \leftarrow 0;

MIENTRAS EA (S) \neq MarcaFin **HACER**

sum \leftarrow sum + EA (S); n \leftarrow n + 1;

Avanzar (S)

FIN_MIENTRAS;

SEGÚN n

n = 0 : Escribir ("Secuencia vacía");

n > 0 : media \leftarrow sum / n;

Escribir (media);

FIN_SEGÚN

FIN.

4.2 Esquemas algorítmicos de recorrido

Esquema 2: $\neg H1$ y $H2$

ESQUEMA ALGORÍTMICO:

Comenzar;

SEGÚN EA

EA = MarcaFin :

{ Tratamiento sec. vacía }

EA \neq MarcaFin :

{ Inic. del tratamiento }

REPETIR

{ Tratamiento de EA }

Avanzar

HASTA EA = MarcaFin;

{ Terminación del tratto. }

FIN_SEGÚN;

Léxico

S : secuencia de Real;

sum, media : Real; n : Entero;

Algoritmo

Comenzar (S);

SEGÚN EA (S)

EA (S) = MarcaFin :

→ Escribir ("Secuencia vacía");

EA (S) \neq MarcaFin :

→ sum \leftarrow 0.0; n \leftarrow 0;

REPETIR

→ sum \leftarrow sum + EA (S); n \leftarrow n + 1;

Avanzar (S)

HASTA EA (S) = MarcaFin;

→ media \leftarrow sum / n;

→ Escribir (media)

FIN_SEGÚN

FIN.

4.2 Esquemas algorítmicos de recorrido

Esquema 3: $\neg H1$ y $\neg H2$

ESQUEMA ALGORÍTMICO:

Comenzar;

SEGÚN EA

EA = MarcaFin :

{ *Tratamiento sec. vacía* }

EA \neq MarcaFin :

{ *Tratamiento 1^{er} elemto.* }

ITERAR

Avanzar

DETENER EA = MarcaFin;

{ *Tratamiento de EA* }

FIN_ITERAR;

{ *Terminación del tratto.* }

FIN_SEGÚN;

Léxico // el mismo de los ej. anteriores

Algoritmo

Comenzar (S);

SEGÚN EA (S)

EA (S) = MarcaFin :

→ Escribir ("Secuencia vacía");

EA (S) \neq MarcaFin :

→ sum \leftarrow EA (S); n \leftarrow 1;

ITERAR

Avanzar (S)

DETENER EA (S) = MarcaFin;

→ sum \leftarrow sum + EA (S); n \leftarrow n + 1

FIN_ITERAR;

→ media \leftarrow sum / n; Escribir (media)

FIN_SEGÚN

FIN.

Los casos base de la inducción ahora son: long = 1 y suma = EA(S)

Consiste en la solución para una secuencia de un elemento

4.2 Esquemas algorítmicos de recorrido

Problema: cálculo de la meseta mayor

Dada una secuencia de enteros > 0 , llamamos *meseta* a una sucesión de valores idénticos. Queremos diseñar un algoritmo que calcule la longitud de la meseta mayor.

Ejemplo: $S = [35, 18, 18, 18, 5, 62, 35, 35, 11, 11, 11, 11, 9, 9]$ la longitud de la meseta mayor sería 4, que sería la longitud de la meseta de valores 11.

ANÁLISIS SUPONIENDO H2:

- $\text{Post} = \{ p = \text{LongMesetaMayor}(S) \}$
 - $\text{Inv} = \{ p = \text{LongMesetaMayor}(P_{iz}) \text{ y } u = \text{último}(P_{iz}) \text{ y } m = \text{LongUltMeseta}(P_{iz}) \}$
 - $p \leftarrow 0$ *// inicialización*
 - SI $p < m$ ENTONCES $p \leftarrow m$ *// tratamiento elemental*
 - $m \leftarrow 0$ *// inicialización*
 - SI $\text{EA}(S) \neq u$ ENTONCES $m \leftarrow 1$ *// tratamiento elemental*
 - SINO $m \leftarrow m + 1$
 - $u \leftarrow ??$;no está definido para la secuencia vacía! *// inicialización*
 - $u \leftarrow \text{EA}(S)$ *// tratamiento elemental*
-

4.2 Esquemas algorítmicos de recorrido

Problema: cálculo de la meseta mayor

ANÁLISIS SUPONIENDO $\neg H2$:

- $Post = \{ p = LongMesetaMayor(S) \}$
- $Inv = \{ p = LongMesetaMayor(P_{iz}) \text{ y } u = \text{último}(P_{iz}) \text{ y } m = LongUltMeseta(P_{iz}) \}$
- $p \leftarrow 0$ *// inicialización*
- SI $p < m$ ENTONCES $p \leftarrow m$ *// tratamiento elemental*
- $m \leftarrow 0$ *// inicialización*
- SI $EA(S) \neq u$ ENTONCES $m \leftarrow 1$ *// tratamiento elemental*
- SINO $m \leftarrow m + 1$
- $u \leftarrow EA(S)$;no está definido para la secuencia vacía! *// inicialización*
- $u \leftarrow EA(S)$ *// tratamiento elemental*

Puesto que la secuencia es de enteros positivos, podremos aplicar cualquiera de los dos esquemas, si en el primero tomamos la precaución de tomar como último un valor ≤ 0 (truco *sucio*)

4.2 Esquemas algorítmicos de recorrido

Problema: cálculo de la meseta mayor

Solución con H2 (Esquema 1):

- Inicialización ($P_{iz} = []$):
 $p \leftarrow 0$; $m \leftarrow 0$;
 $u \leftarrow 0$; // por ejemplo
- Cuerpo:
SI EA (S) = u **ENTONCES**
 $m \leftarrow m + 1$
SI_NO
 $m \leftarrow 1$
FIN_SI;
SI $m > p$ **ENTONCES**
 $p \leftarrow m$
FIN_SI;
 $u \leftarrow \text{EA}(S)$;
- Terminación:
Escribir (p);

Léxico

S : secuencia de Entero;
 p, m, u : Entero;

Algoritmo

Comenzar (S);
 $p \leftarrow 0$; $m \leftarrow 0$; $u \leftarrow 0$;
MIENTRAS EA (S) \neq MarcaFin **HACER**
 SI EA (S) = u **ENTONCES**
 $m \leftarrow m + 1$
 SI_NO
 $m \leftarrow 1$
 FIN_SI;
 SI $m > p$ **ENTONCES**
 $p \leftarrow m$
 FIN_SI;
 $u \leftarrow \text{EA}(S)$;
 Avanzar (S)
FIN_MIENTRAS;
Escribir (p)
FIN.

4.2 Esquemas algorítmicos de recorrido

Problema: cálculo de la meseta mayor

Solución con $\neg H2$ (Esquema 3):

- Inicialización ($P_{iz} = [e]$):
 $p \leftarrow 1$; $m \leftarrow 1$;
 $u \leftarrow EA(S)$; // $u \leftarrow e$
- Cuerpo:
SI $EA(S) = u$ **ENTONCES**
 $m \leftarrow m + 1$
SI_NO
 $m \leftarrow 1$
FIN_SI;
SI $m > p$ **ENTONCES**
 $p \leftarrow m$
FIN_SI;
 $u \leftarrow EA(S)$;
- Terminación:
Escribir (p);

Léxico // el mismo del algoritmo anterior

Algoritmo

```
Comenzar (S);
SEGÚN EA (S)
    EA (S) = MarcaFin :
        Escribir ("Secuencia vacía");
    EA (S)  $\neq$  MarcaFin :
         $p \leftarrow 1$ ;  $m \leftarrow 1$ ;  $u \leftarrow EA(S)$ ;
        ITERAR
            Avanzar (S)
        DETENER EA (S) = MarcaFin;
            SI EA (S) = u ENTONCES  $m \leftarrow m + 1$ 
            SI_NO  $m \leftarrow 1$ 
            FIN_SI;
            SI  $m > p$  ENTONCES  $p \leftarrow m$  FIN_SI;
             $u \leftarrow EA(S)$ 
        FIN_ITERAR;
        Escribir (p)
    FIN_SEGÚN
FIN.
```

4.2 Esquemas algorítmicos de recorrido

Problema: obtención del máximo

Postcondición:

$$m = \max (S)$$

Invariante:

$$m = \max (P_{iz}(S))$$

max para la secuencia vacía = ??

// no definido: $\neg H2$

- Inicialización:
 $m \leftarrow EA(S);$
- Cuerpo:
SI $EA(S) > m$ **ENTONCES**
 $m \leftarrow EA(S)$
FIN_SI;
- Terminación:
Escribir (m);

Léxico

S : secuencia de Entero;

m : entero; // el anterior

Algoritmo

Comenzar (S);

SEGÚN EA (S)

EA (S) = MarcaFin :

 Escribir ("Secuencia vacía");

EA (S) \neq MarcaFin :

$m \leftarrow EA(S);$

ITERAR

 Avanzar (S)

DETENER EA (S) = MarcaFin;

SI $EA(S) > m$ **ENTONCES**

$m \leftarrow EA(S)$

FIN_SI

FIN_ITERAR;

 Escribir (m)

FIN_SEGÚN

FIN.

4.2 Esquemas algorítmicos de recorrido

Problema: copia de una secuencia

- A partir de una secuencia S1 deseamos crear otra secuencia S2 idéntica a la anterior. Utilizaremos un esquema de recorrido de S1 y creación simultánea de S2:

Léxico

S1, S2 : secuencia de Carácter;

Algoritmo

Comenzar (S1);

Crear (S2);

// Inv : { Piz (S1) = Piz (S2) }

MIENTRAS EA (S1) \neq MarcaFin **HACER**

 Registrar (S2, EA (S1));

 Avanzar (S1)

FIN_MIENTRAS;

 Marcar (S2)

FIN.

4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

Dada una secuencia de enteros S , obtener:

- El valor máximo (ma), y también:
 - La posición donde éste aparece por primera vez (pma)
 - La posición donde éste aparece por última vez (uma)
 - El número de veces que éste aparece (nma)
- El valor mínimo (mi), y también:
 - La posición donde éste aparece por primera vez (pmi)
 - La posición donde éste aparece por última vez (umi)
 - El número de veces que éste aparece (nmi)

Ejemplo: Si $S = [34, 1, 90, 39, 1, 15, 90, 25, 1, 10]$

Los resultados buscados son:

$$ma = 90, pma = 3, uma = 7, nma = 2$$

$$mi = 1, pmi = 2, umi = 9, nmi = 3$$

4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

Necesitamos aplicar un razonamiento inductivo:

$ma \leftarrow EA(S)$

SI $EA(S) > ma$ ENTONCES $ma \leftarrow EA(S)$

$mi \leftarrow EA(S)$

SI $EA(S) < mi$ ENTONCES $mi \leftarrow EA(S)$

4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

$\text{pos} \leftarrow 1$

$\text{pos} \leftarrow \text{pos} + 1$

$\text{nma} \leftarrow 1$

SI $\text{EA}(S) > \text{max}$ ENTONCES $\text{nma} \leftarrow 1$

SI $\text{EA}(S) = \text{max}$ ENTONCES $\text{nma} \leftarrow \text{nma} + 1$

$\text{nmi} \leftarrow 1$

SI $\text{EA}(S) < \text{min}$ ENTONCES $\text{nmi} \leftarrow 1$

SI $\text{EA}(S) = \text{min}$ ENTONCES $\text{nmi} \leftarrow \text{nmi} + 1$

4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

$pma \leftarrow 1$

SI $EA(S) > \max$ ENTONCES $pma \leftarrow \text{long}$

$uma \leftarrow 1$

SI $EA(S) \geq \max$ ENTONCES $uma \leftarrow \text{long}$

$pmi \leftarrow 1$

SI $EA(S) < \min$ ENTONCES $pmi \leftarrow \text{long}$

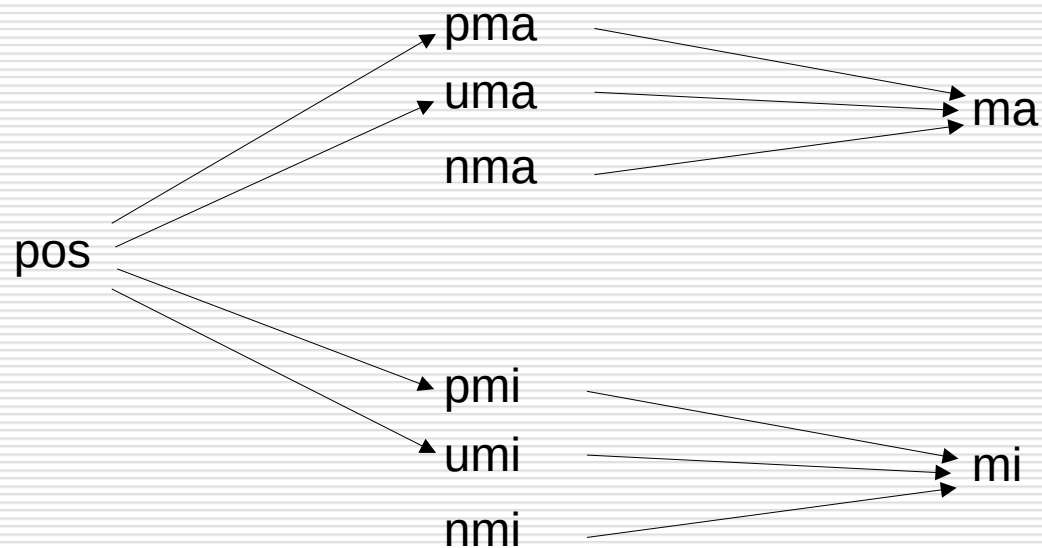
$umi \leftarrow 1$

SI $EA(S) \leq \min$ ENTONCES $umi \leftarrow \text{long}$

4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

Las relaciones de dependencia de las variables del problema son:



4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

- A partir de lo anterior obtenemos la inicialización y, después de realizar una ordenación topológica adecuada, también los tratamientos del cuerpo:

- Inicialización:

$\text{pos} \leftarrow 1; \text{ma} \leftarrow \text{EA}(S); \text{pma} \leftarrow 1; \text{uma} \leftarrow 1; \text{nma} \leftarrow 1;$
 $\text{mi} \leftarrow \text{EA}(S); \text{pmi} \leftarrow 1; \text{umi} \leftarrow 1; \text{nmi} \leftarrow 1;$

- Cuerpo:

$\text{pos} \leftarrow \text{pos} + 1;$

SEGÚN $\text{EA}(S)$, ma

$\text{EA}(S) = \text{ma} :$ $\text{uma} \leftarrow \text{pos}; \text{nma} \leftarrow \text{nma} + 1;$

$\text{EA}(S) > \text{ma} :$ $\text{uma} \leftarrow \text{pos}; \text{pma} \leftarrow \text{pos}; \text{ma} \leftarrow \text{EA}(S); \text{nma} \leftarrow 1;$

$\text{EA}(S) < \text{ma} :$; // acción vacía

FIN_SEGÚN;

SEGÚN $\text{EA}(S)$, mi

$\text{EA}(S) = \text{mi} :$ $\text{umi} \leftarrow \text{pos}; \text{nmi} \leftarrow \text{nmi} + 1;$

$\text{EA}(S) < \text{mi} :$ $\text{umi} \leftarrow \text{pos}; \text{pmi} \leftarrow \text{pos}; \text{mi} \leftarrow \text{EA}(S); \text{nmi} \leftarrow 1;$

$\text{EA}(S) > \text{mi} :$; // acción vacía

FIN_SEGÚN;

4.2 Esquemas algorítmicos de recorrido

Problema: máximos y mínimos

Léxico

S : secuencia de Entero;
ma, nma, pma, uma : Entero;
mi, nmi, pmi, umi, pos : Entero;

Algoritmo

Comenzar (S);

SEGÚN EA (S)

EA (S) = MarcaFin :

Escribir ("Secuencia vacía");

EA (S) \neq MarcaFin :

ma \leftarrow EA (S); mi \leftarrow EA (S);

nma \leftarrow 1; pma \leftarrow 1; uma \leftarrow 1;

nmi \leftarrow 1; pmi \leftarrow 1; umi \leftarrow 1;

pos \leftarrow 1;

ITERAR

Avanzar (S);

DETENER EA (S) = MarcaFin;

pos \leftarrow pos + 1;

// continúa...

// ... continuación

SEGÚN EA (S), ma

EA (S) = ma : uma \leftarrow pos;

nma \leftarrow nma + 1;

EA (S) > ma : uma \leftarrow pos;

nma \leftarrow 1;

pma \leftarrow pos; ma \leftarrow EA (S);

EA (S) < ma : ;

FIN_SEGÚN;

SEGÚN EA (S), mi

EA (S) = mi : umi \leftarrow pos;

nmi \leftarrow nmi + 1;

EA (S) < mi : umi \leftarrow pos; nmi \leftarrow 1;

pmi \leftarrow pos; mi \leftarrow EA (S);

EA (S) > mi : ;

FIN_SEGÚN

FIN_ITERAR;

Escribir (ma, nma, pma, uma);

Escribir (mi, nmi, pmi, umi)

FIN_SEGÚN

FIN.

4.2 Esquemas algorítmicos de recorrido

Problema: intercalación de secuencias

- **Enunciado:** Dadas dos secuencias de enteros S1 y S2 ordenadas decrecientemente obtener otra secuencia R que contenga los elementos de S1 y S2 y conserve la misma relación de orden.
 - Ejemplo: Si las secuencias de entrada son:
S1 = [25, 11, 11, 3, 1, 1]
S2 = [38, 25, 14, 11, 9, 9, 7]
 - La secuencia resultado debería ser:
R = [38, 25, 25, 14, 11, 11, 11, 9, 9, 7, 3, 1, 1]
 - Postcondición: Post : { R = Intercalar (S1, S2) }
 - Invariante: Inv : { $P_{iz}(R) = \text{Intercalar}(P_{iz}(S1), P_{iz}(S2))$ }
-

4.2 Esquemas algorítmicos de recorrido

Problema: intercalación de secuencias

- Necesitamos una definición inductiva de la función *Intercalar*. Los casos triviales son:
 $\text{Intercalar}([], []) = []$
 $\text{Intercalar}(T \cdot e, []) = T \cdot e \quad \text{Intercalar}([], T \cdot e) = T \cdot e$
 - Análisis 1: (punto de vista de lo ya tratado)
 $\text{Intercalar}(T1 \cdot e1, T2 \cdot e2) =$
 - $\text{Intercalar}(T1, T2) \& [e1, e2] \quad \text{si } e1 = e2$
 - $\text{Intercalar}(T1, T2 \cdot e2) \& [e1] \quad \text{si } e1 < e2$
 - $\text{Intercalar}(T1 \cdot e1, T2) \& [e2] \quad \text{si } e1 > e2$
 - Análisis 2: (punto de vista de lo pendiente de recorrer)
 $\text{Intercalar}(e1 \circ T1, e2 \circ T2) =$
 - $[e1, e2] \& \text{Intercalar}(T1, T2) \quad \text{si } e1 = e2$
 - $[e2] \& \text{Intercalar}(e1 \circ T1, T2) \quad \text{si } e1 < e2$
 - $[e1] \& \text{Intercalar}(T1, e2 \circ T2) \quad \text{si } e1 > e2$
-

4.2 Esquemas algorítmicos de recorrido

Problema: intercalación de secuencias

- Para los casos triviales definiremos una acción parametrizada que copie el resto de una secuencia en otra:

CopiarResto : **acción (dato-resultado** S, R : secuencia de Entero);

Pre : { (Estado (S) = consulta) Y (Estado (R) = creación) Y

(P_{dr} (S) = T·MarcaFin) Y (EA (S) ≠ MarcaFin) Y (R = U) }

Post : { (EA (S) = MarcaFin) Y (R = U & T) }

Algoritmo

REPETIR

Registrar (R, EA (S));

Avanzar (S)

HASTA EA (S) = MarcaFin

FIN;

- En los casos no triviales haremos un análisis de casos y registraremos en cada caso el mayor elemento.
-

4.2 Esquemas algorítmicos de recorrido

Problema: intercalación de secuencias

Léxico

S1, S2, R : secuencia de Entero;
CopiarResto : acción // definida antes

Algoritmo

Comenzar (S1); Comenzar (S2); Crear (R);

MIENTRAS (EA (S1) \neq MarcaFin) **Y**
 (EA (S2) \neq MarcaFin) **HACER**

SEGÚN EA (S1), EA (S2)

 EA (S1) = EA (S2) :

 Registrar (R, EA (S1));

 Registrar (R, EA (S2));

 Avanzar (S1); Avanzar (S2);

 EA (S1) > EA (S2) :

 Registrar (R, EA (S1));

 Avanzar (S1);

 EA (S1) < EA (S2) :

 Registrar (R, EA (S2));

 Avanzar (S2);

FIN_SEGÚN

FIN_MIENTRAS;

// continúa...

// ... continuación

SEGÚN EA (S1), EA (S2)

 (EA (S1) \neq MarcaFin) **Y**

 (EA (S2) = MarcaFin) :

 CopiarResto (S1, R);

 (EA (S1) = MarcaFin) **Y**

 (EA (S2) \neq MarcaFin) :

 CopiarResto (S2, R);

EN_OTRO_CASO : ;

FIN_SEGÚN;

 Marcar (R)

FIN.

4.3 Esquemas algorítmicos de búsqueda

- **Problema de la búsqueda:** Encontrar el primer elemento de la secuencia que cumpla una cierta propiedad *Pro*

- Postcondición:

$$\text{Post} = \{ (\text{Pro}(\text{EA}) = \text{Verdadero} \bullet \text{EA} = \text{MarcaFin}) \\ \mathbf{Y} (\forall e \in P_{iz} : \text{Pro}(e) = \text{Falso}) \}$$

- Invariante:

$$\text{Inv} = \{ \forall e \in P_{iz} : \text{Pro}(e) = \text{Falso} \}$$

Inv = Ningún elemento de la parte izquierda de S cumple la propiedad Pro

4.3 Esquemas algorítmicos de búsqueda

- **Problema de la búsqueda:** Encontrar el primer elemento de la secuencia que cumpla una cierta propiedad *Pro*
 - Condición de continuación de la búsqueda:
(EA \neq MarcaFin) **Y NO** Pro (EA)
 - CC = Continuará la iteración si no hemos llegado al final de la secuencia y tampoco hemos encontrado un elemento que cumpla la propiedad Pro
 - La propiedad Pro puede involucrar no solo al elemento actual 'e' sino también propiedades de la Piz. En ese caso, dentro de la iteración habrá que hacer los tratamientos necesarios para mantener esa parte del invariante.
-

4.3 Esquemas algorítmicos de búsqueda

ESQUEMA ALGORÍTMICO:

Comenzar; { *Establecimiento inicial del invariante* }*

MIENTRAS (EA \neq MarcaFin) **Y NO** Pro (EA) **HACER**

{ *Tratamientos para mantener el invariante* }*

Avanzar

FIN_MIENTRAS;

SEGÚN EA

EA = MarcaFin : { *Tratamiento elemento no encontrado* }

EA \neq MarcaFin : { *Tratamiento EA encontrado* }

FIN_SEGÚN;

**(En el caso en que sea necesario)*

- Si la marca de fin no sólo no es significativa sino que es erróneo acceder a ella para comprobar la propiedad, puede ser necesario:

MIENTRAS (EA \neq MarcaFin) **Y DESPUÉS NO** Pro (EA) **HACER**

{ *Tratamientos para mantener el invariante* }*

Avanzar

FIN_MIENTRAS;

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

- Existencia de una 'A' en un texto (sec. de caracteres):

Léxico

S : secuencia de Carácter;

Algoritmo

Comenzar (S);

MIENTRAS (EA (S) \neq MarcaFin) **Y** (EA (S) \neq 'A') **HACER**

Avanzar (S)

FIN_MIENTRAS;

SEGÚN EA (S)

EA (S) = MarcaFin : Escribir ("No hay ninguna A");

EA (S) \neq MarcaFin : Escribir ("Hay al menos una A");

FIN_SEGÚN

FIN.

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

- **Enunciado:** Comprobar si en un texto (secuencia de caracteres) existen al menos x ocurrencias de un carácter c dado:
 - Propiedad buscada:
$$\text{Pro}(e) = \{ S = P_{iz} \& [e] \& P_{dr} \ \mathbf{Y} \ (e = c) \ \mathbf{Y} \ \text{NroOcur}(c, P_{iz}) = x - 1 \}$$
 - Invariante:
$$\text{Inv} = \{ \text{noc} = \text{NroOcur}(c, P_{iz}) \ \mathbf{Y} \ \text{noc} < x \}$$
 - Análisis inductivo:
$$\text{NroOcur} \leftarrow 0$$

$$\text{Si } e = c \text{ ENTONCES } \text{NroOcur} \leftarrow \text{NroOcur} + 1$$
-

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

Léxico

S : secuencia de Carácter;

x : Entero > 0; noc : Entero; c : Carácter;

Algoritmo

Comenzar (S);

Leer (x, c); $\text{noc} \leftarrow 0$;

Establecimiento inicial del invariante

no Pro(EA)

MIENTRAS (EA (S) \neq MarcaFin) **Y** ((EA (S) \neq c) **O** (noc \neq x - 1)) **HACER**

SI EA (S) = c **ENTONCES** noc \leftarrow noc + 1 **FIN_SI**;

Avanzar (S)

*Tratamientos para
mantener el invariante*

FIN_MIENTRAS;

SEGÚN EA (S)

EA (S) = MarcaFin : Escribir (c, " no aparece ", x, " veces");

EA (S) \neq MarcaFin : Escribir (c, " aparece al menos ", x, " veces");

FIN_SEGÚN

FIN.

4.3 Esquemas algorítmicos de búsqueda

Búsqueda con garantía de éxito:

- Si se sabe que al menos un elemento cumple la propiedad, se puede omitir la comparación con la marca de fin:

ESQUEMA ALGORÍTMICO:

Comenzar;

MIENTRAS NO Pro (EA) **HACER**

Avanzar

FIN_MIENTRAS;


SEGÚN EA

EA = MarcaFin : { No encontrado }

EA \neq MarcaFin : { EA encontrado }

FIN_SEGÚN;

*(estamos suponiendo que podemos
comparar la marca de fin con el blanco,
y que el resultado de la comparación
sería Falso)*



Problema: ¿Un texto es blanco?

Léxico

S : Secuencia de Carácter;

Algoritmo

Comenzar (S);

MIENTRAS EA (S) = ' ' **HACER**

Avanzar (S)

FIN_MIENTRAS;

SEGÚN EA (S)

EA (S) = MarcaFin :

Escribir ("Texto blanco");

EA (S) \neq MarcaFin :

Escribir ("Texto no blanco");

FIN_SEGÚN

FIN.

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

- **Enunciado:** Calcular la longitud de la primera palabra de un texto:
 - Análisis:
 - Doble búsqueda:
 - Búsqueda del comienzo de la primera palabra
 - Búsqueda del final de la primera palabra y conteo de su número de caracteres
 - La primera de las búsquedas será con garantía de éxito, como la del ejemplo anterior.
 - Para la segunda búsqueda introduciremos un contador.
-

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

Léxico

S : secuencia de Carácter;

lpp : Entero;

Algoritmo

Comenzar (S);

1) { **MIENTRAS** EA (S) = ' ' **HACER**
 Avanzar (S)
FIN_MIENTRAS;
SEGÚN EA (S)

 EA (S) = MarcaFin :

 Escribir ("Texto blanco");

 EA (S) ≠ MarcaFin :

 lpp ← 0;

2) { **MIENTRAS** (EA (S) ≠ MarcaFin) **Y** (EA (S) ≠ ' ') **HACER**
 lpp ← lpp + 1;
 Avanzar (S)
FIN_MIENTRAS;
 Escribir (lpp)

FIN_SEGÚN

FIN.

Cuestión: ¿Se podría sustituir 2) por...

REPETIR

lpp ← lpp + 1;

Avanzar (S)

HASTA (EA (S) = MarcaFin) **O** (EA (S) = ' ');

⇒ En caso afirmativo, ¿qué iteración parece la más adecuada en este caso? ¿Por qué? ¿Existe otra aún mejor que la mejor de estas dos?

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

- **Enunciado:** Determinar si un texto contiene las letras 'A', 'E' e 'I' en este orden:
- **Análisis:**
 - Tres búsquedas distintas pero de idéntica naturaleza.
 - Para realizarlas definiremos una acción parametrizada:

```
BuscarLetra : acción (dato c : Carácter; dato-resultado S : secuencia de Carácter)
// Las secuencias siempre las pasaremos como dato-resultado pues, aunque
// no modifiquemos su valor, si avanzamos modificamos su estado interno
Pre : { Estado (S) = consulta }
Post : { (EA (S) = c) O (EA (S) = MarcaFin) }
Algoritmo
    MIENTRAS (EA (S) ≠ MarcaFin) Y (EA (S) ≠ c) HACER
        Avanzar (S)
    FIN_MIENTRAS
FIN;
```

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

Léxico

S : Secuencia de Carácter;

BuscarLetra : **acción** // definida antes

Algoritmo

Comenzar (S);

BuscarLetra ('A', S);

SEGÚN EA (S)

EA (S) = MarcaFin : Escribir ("No");

EA (S) \neq MarcaFin :

BuscarLetra ('E', S);

SEGÚN EA (S)

EA (S) = MarcaFin : Escribir ("No");

EA (S) \neq MarcaFin :

BuscarLetra ('I', S);

SEGÚN EA (S)

EA (S) = MarcaFin : Escribir ("No");

EA (S) \neq MarcaFin : Escribir ("Sí");

FIN_SEGÚN

FIN_SEGÚN

FIN_SEGÚN

FIN.

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

➤ Cuestiones:

- ¿Sería igualmente correcto el algoritmo del modo siguiente?:

```
Comenzar (S);  
BuscarLetra ('A', S);  
BuscarLetra ('E', S);  
BuscarLetra ('I', S);  
SEGÚN EA (S)  
    EA (S) = MarcaFin : Escribir ("No");  
    EA (S) ≠ MarcaFin : Escribir ("Sí");  
FIN_SEGÚN
```

- ¿Nos serviría el algoritmo original y/o este último si en lugar de buscar las letras 'A', 'E' e 'I' buscásemos las letras 'A', 'A' y 'E'?
 - ¿Podríamos adaptarlo modificando el algoritmo principal?
 - ¿Y modificando la acción? ¿Habría que establecer diferentes pre/postcondiciones para ella en ese caso?
-

4.3 Esquemas algorítmicos de búsqueda

Ejemplos:

- **Enunciado:** Dadas dos secuencias S1 y S2 deseamos determinar si son iguales, es decir, tienen los mismos elementos y en igual orden. Búsqueda de su primera diferencia recorriendo ambas simultáneamente:

Léxico

S1, S2 : secuencia de Carácter;

Algoritmo

Comenzar (S1); Comenzar (S2);

// Inv : { Piz (S1) = Piz (S2) }

MIENTRAS (EA (S1) \neq MarcaFin) **Y** (EA (S2) \neq MarcaFin) **Y**
(EA (S1) = EA (S2)) **HACER**

Avanzar (S1); Avanzar (S2)

FIN_MIENTRAS;

SEGÚN EA (S1), EA (S2)

(EA (S1) = MarcaFin) **Y** (EA (S2) = MarcaFin) : Escribir ("Son iguales");

EN_OTRO_CASO : Escribir ("Son diferentes");

FIN_SEGÚN

FIN.

4.4 Combinación de esquemas de recorrido y búsqueda

- En la práctica los esquemas de recorrido y búsqueda aparecen frecuentemente combinados en un mismo problema.
 - Para resolver estos problemas es preciso:
 - Identificar la naturaleza de cada subproblema o parte del problema.
 - Establecer la forma en que éstos se combinan: un subproblema anidado dentro de otro (como parte del tratamiento de su elemento actual) o compuesto de forma secuencial (uno a continuación del otro).
 - Una vez identificados los subproblemas y la forma de combinar éstos aplicaremos el esquema algorítmico adecuado en cada caso.
-

4.4 Combinación de esquemas de recorrido y búsqueda. Ejemplos:

- **Enunciado:** Dadas dos secuencias de enteros S1 y S2 ordenadas decrecientemente y sin elementos repetidos, determinar qué elementos de S1 están también en S2 y cuáles no.
 - **Análisis:**
 - Todos los elementos de S1 deben ser tratados, pues hay que comprobar si están o no en S2: Recorrido de S1
 - Para cada elemento de S1 debemos comprobar si está o no en S2: Búsqueda en S2 **anidada** en el recorrido de S1.
 - Puesto que ambas secuencias están ordenadas, cada búsqueda en S2 puede continuar a partir del punto donde quedó la búsqueda anterior sin necesidad de recomenzar.
-

4.4 Combinación de esquemas de recorrido y búsqueda. Ejemplos:

Léxico

S1, S2 : secuencia de Entero;

Algoritmo

Comenzar (S1); Comenzar (S2);

MIENTRAS EA (S1) \neq MarcaFin **HACER**

2) { **MIENTRAS** (EA (S2) \neq MarcaFin) **Y** (EA (S2) > EA (S1)) **HACER**
 Avanzar (S2)
 FIN_MIENTRAS;

1) { **SI** (EA (S2) \neq MarcaFin) **Y** (EA (S2) = EA (S1)) **ENTONCES**
 Escribir (EA (S1), " está también en S2")

SI_NO

Escribir (EA (S1), " no está en S2")

FIN_SI;

Avanzar (S1)

FIN_MIENTRAS

FIN.

1) Recorrido de S1 (H1 y H2)
2) Búsqueda en S2
(anidada en el anterior)

Cuestión: Realice una traza manual para los siguientes casos:

a) S1 = [30, 24, 12, 6, 5] S2 = [60, 58, 30, 15, 12, 3, 2]

b) S1 = [60, 58, 30, 15, 12, 3, 2] S2 = [30, 24, 12, 6, 5]

4.4 Combinación de esquemas de recorrido y búsqueda. Ejemplos:

- **Enunciado:** Dadas dos secuencias de enteros S1 y S2 determinar si en S2 existe algún elemento igual a la suma de todos los elementos de S1, si hay alguno.
 - Análisis:
 - Distinguiremos el caso de S1 vacía del resto ($\neg H1$), pues la suma de una secuencia vacía *no* está definida.
 - Debemos sumar los elementos de S1: Recorrido de S1.
 - Una vez obtenida la suma de los elementos de S1, debemos determinar si ese valor es igual a alguno de los elementos de S2: Búsqueda en S2 **compuesta secuencialmente** con el anterior recorrido.
-

4.4 Combinación de esquemas de recorrido y búsqueda. Ejemplos:

Léxico

S1, S2 : secuencia de Entero; suma : Entero;

Algoritmo

Comenzar (S1);

SEGÚN EA (S1)

EA (S1) = MarcaFin : Escribir ("Secuencia S1 vacía");

EA (S1) \neq MarcaFin :

suma \leftarrow EA (S1);

ITERAR

Avanzar (S1)

DETENER EA (S1) = MarcaFin;

suma \leftarrow suma + EA (S1)

FIN_ITERAR;

Comenzar (S2);

MIENTRAS (EA (S2) \neq MarcaFin) **Y** (EA (S2) \neq suma) **HACER**

Avanzar (S2)

FIN_MIENTRAS;

SI EA (S2) = MarcaFin **ENTONCES** Escribir ("No")

SI_NO Escribir ("Sí")

FIN_SI

FIN_SEGÚN

FIN.

1)

2)

1) Recorrido de S1

($\neg H1$ y $\neg H2$)

2) Búsqueda en S2

(en secuencia con el anterior)

4.5 Generalización de los modelos de acceso secuencial

- En general, el acceso secuencial a una colección de elementos se distingue por dos características:
 - Disponibilidad inmediata del primer elemento tras la operación de inicio de acceso (DP) o necesidad de una primera operación de lectura o avance (\neg DP).
 - Detección del final cuando se sobrepasa el último elemento significativo (FS) o cuando se está justo en éste (\neg FS).
- Estas características resultan en cuatro combinaciones posibles, que son los **cuatro modelos** de acceso:

	DP	\neg DP
FS	primer modelo	cuarto modelo
\neg FS	tercer modelo	segundo modelo

4.5 Generalización de los modelos de acceso secuencial: Primer modelo

- El modelo de secuencias marcadas visto hasta ahora sigue el **primer modelo** (DP y FS), ya que:
 - a) Disponemos del primer elemento (EA) tras *Comenzar* (DP).
 - b) Detectamos el final cuando llegamos a la *MarcaFin*, que está más allá del último elemento significativo (FS).
 - En la mayoría de los casos, en el primer modelo no existe realmente una *MarcaFin* que se registre en la secuencia, sino que se dispone de una función booleana que indica el final de la secuencia (FDS) cuando se sobrepasa el último elemento significativo y ya no es posible acceder al EA.
 - En ese caso, tendremos que cuidar las comparaciones que hagamos en las iteraciones, sobre todo en las búsquedas, pues el orden de evaluación de éstas será importante para la corrección del algoritmo (operadores lógicos **YDESPUÉS** y **ODESPUÉS**).
-

4.5 Generalización de los modelos de acceso secuencial: Primer modelo

- Esquemas algorítmicos generalizados del primer modelo:

ESQUEMA 1 (H1 y H2):

Comenzar;

{ Inicialización del tratamiento }

MIENTRAS NO FDS HACER

{ Tratamiento de EA }

Avanzar

FIN_MIENTRAS;

{ Terminación del tratamiento }

ESQUEMA 2 (\neg H1 y H2):

Comenzar;

SEGÚN FDS

FDS :

{ Tratamiento sec. vacía }

NO FDS :

{ Inic. del tratamiento }

REPETIR

{ Tratamiento de EA }

Avanzar

HASTA FDS;

{ Terminación del tratto. }

FIN_SEGÚN;

4.5 Generalización de los modelos de acceso secuencial: Primer modelo

- Esquemas algorítmicos generalizados del primer modelo:

ESQUEMA 3 ($\neg H1$ y $\neg H2$):

Comenzar;

SEGÚN FDS

FDS :

{ Tratamiento sec. vacía }

NO FDS :

{ Tratamiento 1^{er} elemto. }

ITERAR

Avanzar

DETENER FDS;

{ Tratamiento de EA }

FIN_ITERAR;

{ Terminación del tratto. }

FIN_SEGÚN;

ESQUEMA DE BÚSQUEDA:

Comenzar; // Inicialización del invariante*

MIENTRAS NO FDS

YDESPUÉS NO Pro (EA) **HACER**

// TTos. para mantener el invariante*

Avanzar

FIN_MIENTRAS;

SEGÚN FDS

FDS : *{ Tratto. elemento no encontrado }*

NO FDS : *{ Tratamiento EA encontrado }*

FIN_SEGÚN;

En el caso de la búsqueda es necesario usar **YDESPUÉS** para no acceder a EA cuando éste ya no está disponible.

* si los hay

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

- La mayoría de los accesos secuenciales sigue los modelos primero o segundo.
 - Estudiaremos los esquemas algorítmicos también del segundo modelo.
 - El tercer modelo nos resultará útil para hacer búsquedas en tablas, aunque no lo estudiaremos formalmente.
 - Como ya dijimos, el **segundo modelo** se distingue por:
 - El primer elemento no está disponible tras el inicio del acceso: es preciso una operación de avance (\neg DP).
 - El final del recorrido se detecta no cuando se sobrepasa el último elemento significativo, sino cuando se está sobre éste (\neg FS). Por tanto, cuando detectemos el final el EA será un elemento que habrá que tratar \Rightarrow variación en el orden de los tratamientos de los esquemas.
-

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

- El conjunto de estados ahora será:
{ ninguno, creación, iniciada, consulta }
- **Operaciones primitivas** del segundo modelo:

Acceso Secuencial

Iniciar (S)	Inicia la consulta sobre la secuencia S. Su primer elemento no está disponible (EA (S) no es significativo)
Avanzar (S)	Avanza al siguiente elemento de S
EA (S)	Retorna el elemento actual de S
EsVacía (S)	Función booleana. Retorna Verdadero si S es la secuencia vacía
EsÚltimo (S)	Función booleana. Retorna Verdadero si ya no es posible realizar más operaciones Avanzar .

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

Creación Secuencial

Crear (S) Crea S como una secuencia vacía sobre la que podemos añadir elementos por la derecha

Registrar (S, e) Graba el elemento e como último elemento de una secuencia S

	ninguno	creación	iniciada	consulta
Iniciar	×	iniciada	iniciada	iniciada
Avanzar	×	×	consulta	consulta
EA	×	×	×	consulta
EsVacía	ninguno	creación	iniciada	consulta
EsÚltimo	×	×	iniciada	consulta
Crear	creación	creación	creación	creación
Registrar	×	creación	×	×

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

- Esquemas algorítmicos del segundo modelo:

ESQUEMA 1 (H1 y H2):

Iniciar;

{ Inicialización del tratamiento }

MIENTRAS NO EsÚltimo **HACER**

Avanzar

{ Tratamiento de EA }

FIN_MIENTRAS;

{ Terminación del tratamiento }

ESQUEMA 2 (\neg H1 y H2):

Iniciar;

SEGÚN EsVacía

EsVacía :

{ Tratamiento sec. vacía }

NO EsVacía :

{ Inic. del tratamiento }

REPETIR

Avanzar

{ Tratamiento de EA }

HASTA EsÚltimo;

{ Terminación del tratto. }

FIN_SEGÚN;

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

- Esquemas algorítmicos del segundo modelo:

ESQUEMA 3 ($\neg H1$ y $\neg H2$):

Iniciar;

SEGÚN EsVacía

EsVacía :

{ Tratamiento sec. vacía }

NO EsVacía :

Avanzar;

{ Tratamiento 1^{er} elemto. }

MIENTRAS NO EsÚltimo **HACER**

Avanzar

{ Tratamiento de EA }

FIN_MIENTRAS;

{ Terminación del tratto. }

FIN_SEGÚN;

ESQUEMA DE BÚSQUEDA:

Iniciar;

SEGÚN EsVacía

EsVacía : *{ Tratamiento: no encontrado }*

NO EsVacía : // Inicialización invariante*

REPETIR

Avanzar // y TTos. mantener invar.*

HASTA EsÚltimo **O** Pro (EA);

SEGÚN EA

Pro (EA) :

{ Tratamiento EA encontrado }

NO Pro (EA) :

{ Tratamiento: no encontrado }

FIN_SEGÚN

FIN_SEGÚN;

* si los hay

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

- Ejemplo: media aritmética con el segundo modelo:

Solución 1 (H1 y H2):

Iniciar (S);

suma \leftarrow 0.0; n \leftarrow 0;

MIENTRAS NO EsÚltimo (S) **HACER**

 Avanzar (S);

 suma \leftarrow suma + EA (S); n \leftarrow n + 1

FIN_MIENTRAS;

SEGÚN n

 n = 0 : Escribir ("Secuencia vacía");

 n > 0 : media \leftarrow suma / n;

 Escribir (media);

FIN_SEGÚN

Solución 2 (\neg H1 y H2):

Iniciar (S);

SEGÚN EsVacía (S)

 EsVacía (S) :

 Escribir ("Secuencia vacía");

NO EsVacía (S) :

 suma \leftarrow 0.0; n \leftarrow 0;

REPETIR

 Avanzar (S);

 suma \leftarrow suma + EA (S);

 n \leftarrow n + 1

HASTA EsÚltimo (S);

 media \leftarrow suma / n;

 Escribir (media)

FIN_SEGÚN;

4.5 Generalización de los modelos de acceso secuencial: Segundo modelo

Solución 3 ($\neg H1$ y $\neg H2$):

Iniciar (S);

SEGÚN EsVacía (S)

EsVacía (S) :

Escribir ("Secuencia vacía");

NO EsVacía (S) :

Avanzar (S);

suma \leftarrow EA (S); n \leftarrow 1;

MIENTRAS NO EsÚltimo (S) **HACER**

Avanzar (S);

suma \leftarrow suma + EA (S);

n \leftarrow n + 1

FIN_MIENTRAS;

media \leftarrow suma / n;

Escribir (media)

FIN_SEGÚN;

Ejemplo de búsqueda:

¿Un texto es blanco?

Iniciar (S);

SEGÚN EsVacía (S)

EsVacía (S) :

Escribir ("Texto vacío");

NO EsVacía (S) :

REPETIR

Avanzar (S);

HASTA EsÚltimo (S) **O** EA (S) \neq ' ';

SI EA (S) \neq ' ' **ENTONCES**

Escribir ("Texto no blanco")

SI_NO

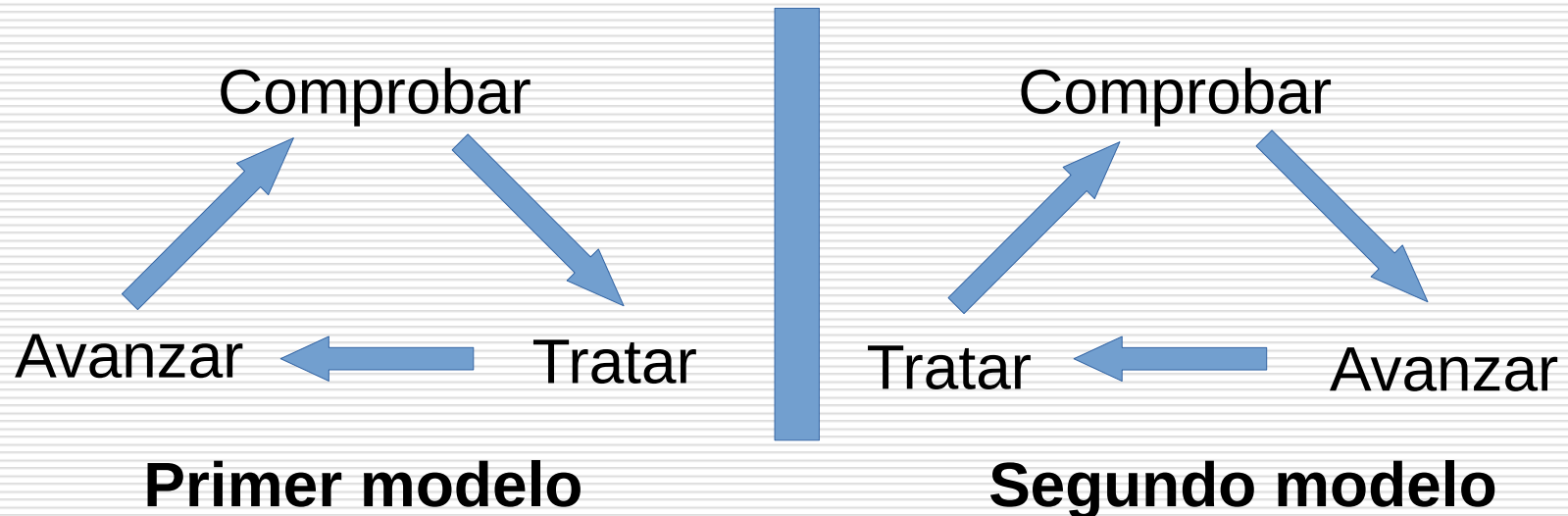
Escribir ("Texto blanco")

FIN_SI

FIN_SEGÚN;

4.5 Generalización de los modelos de acceso secuencial: ciclos de iteración

En general, los dos modelos de acceso secuencial se van a diferenciar por el orden en el que hacemos los tres tratamientos elementales del ciclo de la iteración: **comprobación** del fin, **tratamiento** del EA y **avance**:



Esto se debe a las diferencias entre ambos modelos en el inicio (disponibilidad o no del primer elemento tras el inicio) y en la detección del final (al sobrepasar el último elemento significativo o al estar sobre él).
