1. Introducción: la abstracción y su uso en la resolución de problemas

- **Ejemplo:** dados cuatro números enteros positivos distintos entre sí, escribir una algoritmo que determine el máximo

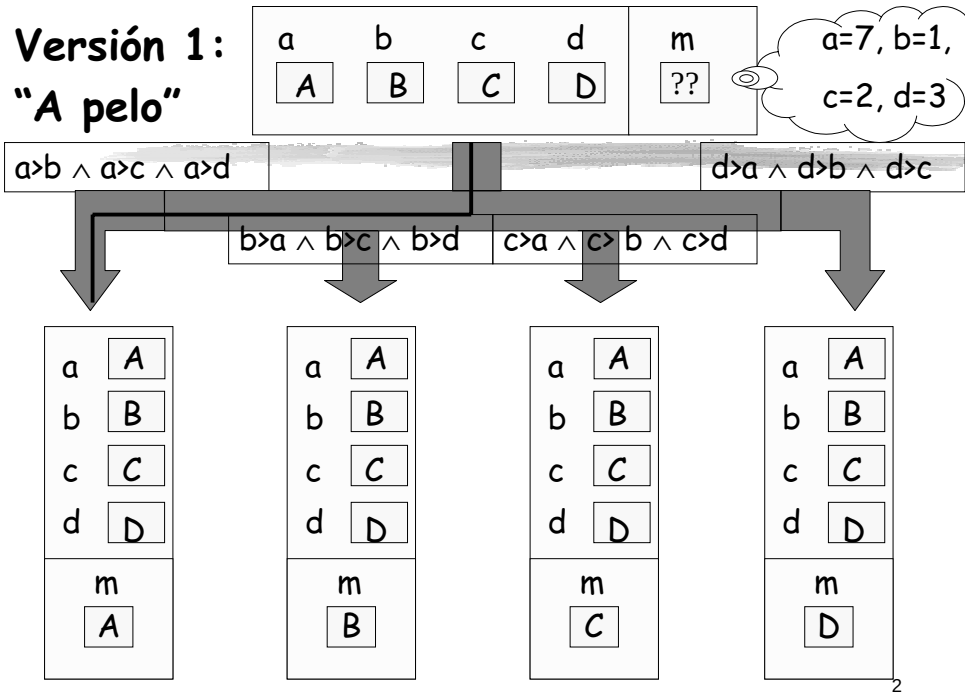
algoritmo mayor4(**DATOS** a,b,c,d: entero;
RESULTADOS m: entero**)**

P={$a \geq 0 \wedge b \geq 0 \wedge c \geq 0 \wedge d \geq 0 \wedge a \neq b \neq c \neq d$**}**

Q={$(m \geq a \wedge m \geq b \wedge m \geq c \wedge m \geq d) \wedge (m = a \vee m = b \vee m = c \vee m = d)$**}**

1

Versión 1: "A pelo"



2

algoritmo mayor4(**DATOS** a,b,c,d: entero;
RESULTADOS m: entero)

P={ a≥0 ∧b≥0 ∧c≥0 ∧d≥0 ∧a≠b ∧a≠c ∧a≠d ∧b≠c ∧b≠d ∧c≠d **}**

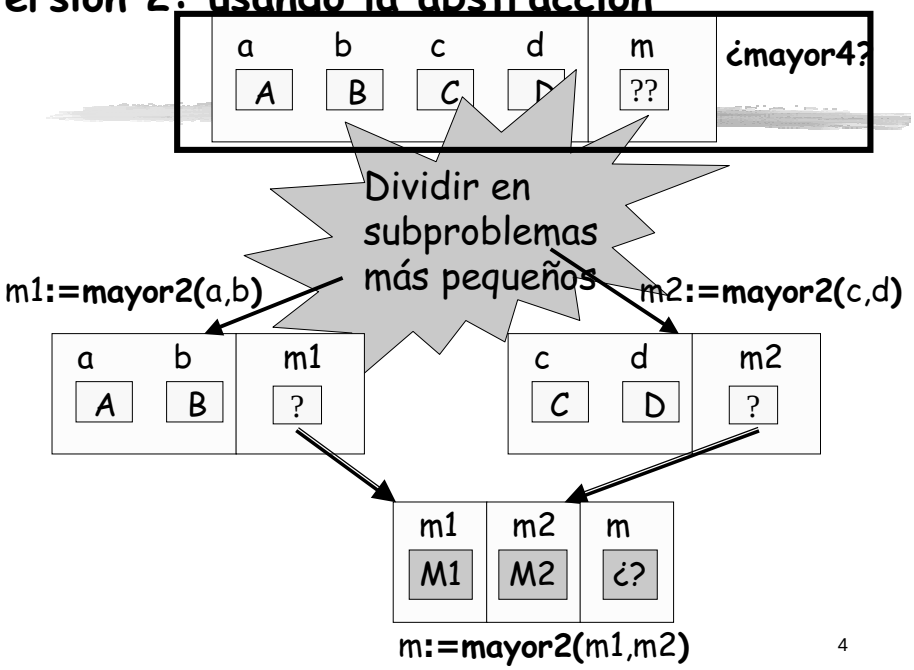
opción
a>b ∧ a>c ∧ a>d: m:=a;
b>a ∧ b≥c ∧ b>d: m:=b;
c>a ∧ c>b ∧ c>d: m:=c;
d>a ∧ d>b ∧ d>c: m:=d;
fopción

Q={ (m≥a ∧ m≥b ∧ m≥c ∧ m≥d)∧(m=a ∨ m=b ∨ m=c ∨ m=d) **}**

falgoritmo

3

Versión 2: usando la abstracción



4

Versión 2: usar la abstracción

Problema más pequeño: dados dos números enteros, escribir un algoritmo que determine el mayor

algoritmo mayor2(**DATOS** x,y: entero;
RESULTADOS mayor: entero**)**

P= {} ! **Ojo** i x e y, son dos enteros cualesquiera

Q= { $(x \geq y \wedge mayor=x) \vee (x<y \wedge mayor=y)$**}**

5

algoritmo mayor4(**DATOS** a,b,c,d: entero;
RESULTADOS m: entero**)**

P={$a \geq 0 \wedge b \geq 0 \wedge c \geq 0 \wedge d \geq 0 \wedge a \neq b \neq c \neq d$**}**

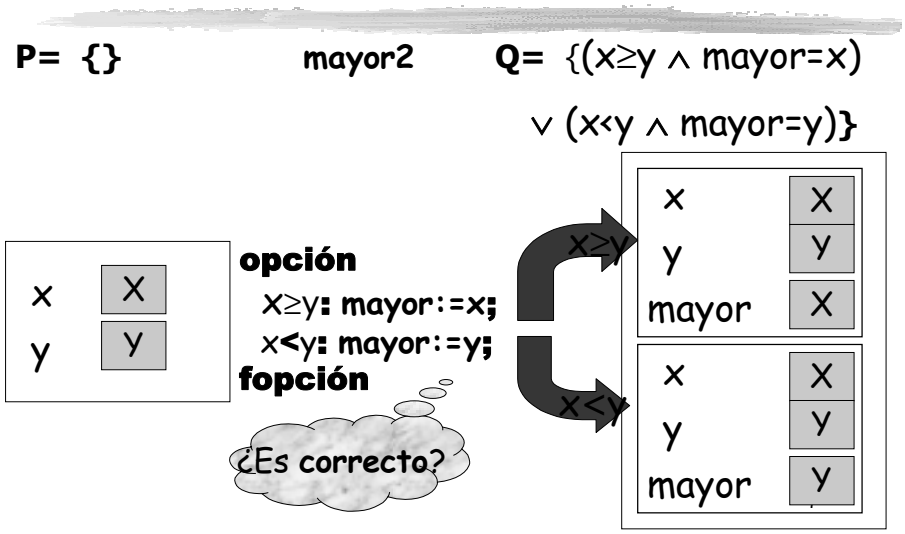
var m1,m2: entero **fvar**
m1:= mayor2(a,b);
m2:=mayor2(c,d);
m:=mayor2(m1,m2);

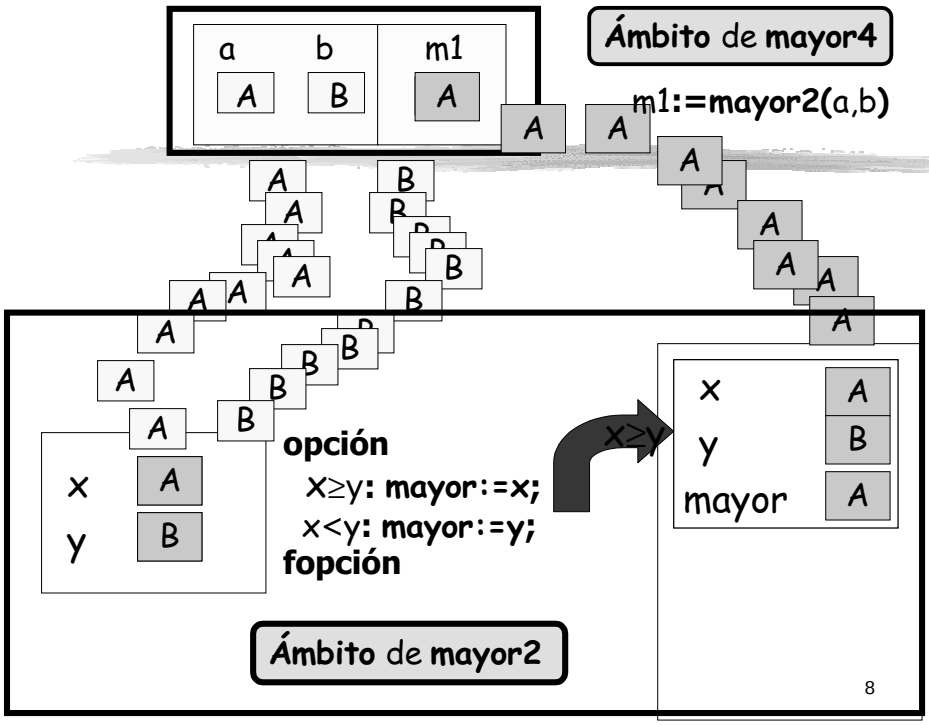
Q={$(m \geq a \wedge m \geq b \wedge m \geq c \wedge m \geq d) \wedge (m=a \vee m=b \vee m=c \vee m=d)$**}**

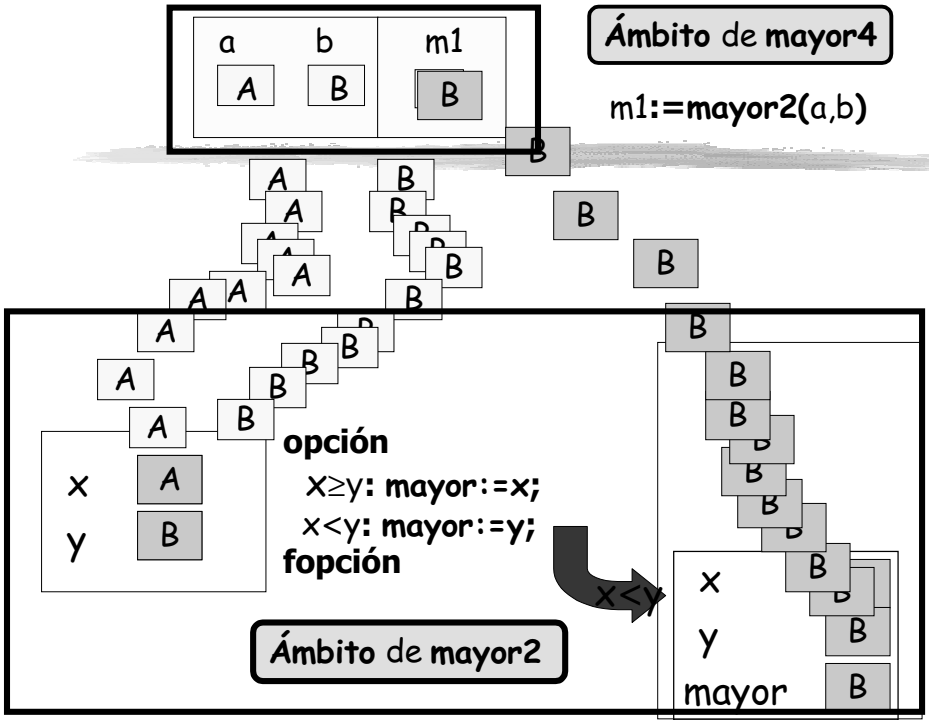
falgoritmo

6

Versión 2: usar la abstracción (*máximo "reconvertido" en mayor2*)







2. Parametrización de un algoritmo: **funciones y procedimientos**

- **¿Qué es un parámetro?**

¿Son iguales los parámetros **DATO** que los **RESULTADO**?

NO

1.-Es una abstracción

2.-Un **DATO** o un **RESULTADO** en la definición de un problema

3.-Existen diferentes **tipos de parámetros**

2. Parametrización de un algoritmo: **funciones y procedimientos**

- **¿Cómo y dónde se indican las diferencias?**

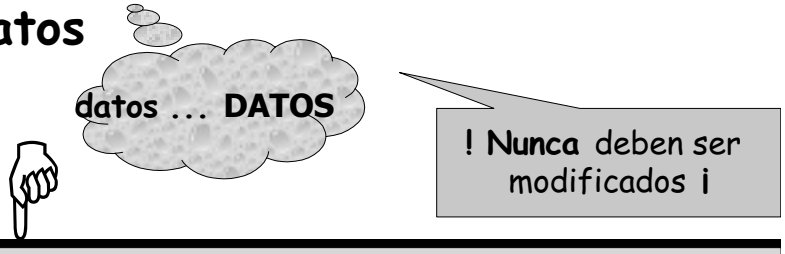
  1.- En la definición de las informaciones relevantes del problema

  2.- En el diseño del algoritmo

11

2.1. Tipos de parámetros

1 ❑ Parámetros que son una abstracción de los **valores** de los **datos**



PARÁMETROS DE ENTRADA

2.1. Tipos de parámetros

2 ❑ Parámetros que son una abstracción de **nombres** de los **valores** de los **datos**

! DEBEN ser modificados **¡** (a partir de un cierto valor inicial)

datos ... DATOS y RESULTADOS

PARÁMETROS DE ENTRADA/SALIDA

2.1. Tipos de parámetros

2_bis ❑ Parámetros que son una abstracción de los **valores** de los **resultados**

! DEBEN ser inicializados y modificados **¡**

resultados ... RESULTADOS

PARÁMETROS DE SALIDA

2.2. Diseño y notación de algoritmos parametrizados

- Etapa 1: **Parametrización**

A partir de la **definición se establecen los tipos de parámetros** que aparecen

FUNCIÓN: **0** de entrada/salida **Y** **1** de salida

PROCEDIMIENTO: ≥ **1** de entrada/salida **Y/Ó** ≥ **2** de salida

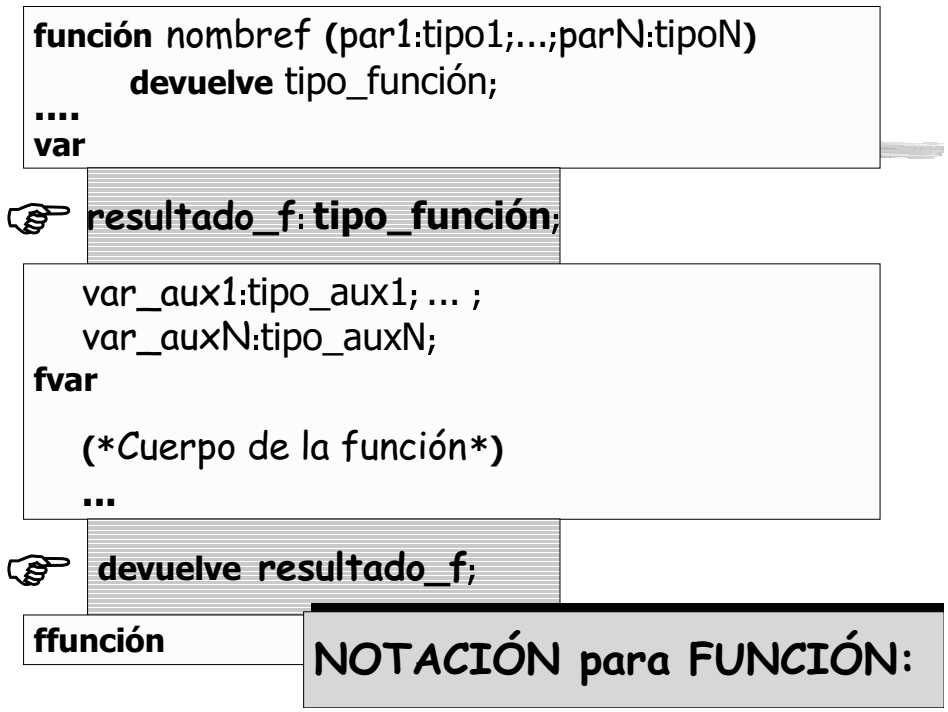
2.2. Diseño y notación de algoritmos parametrizados

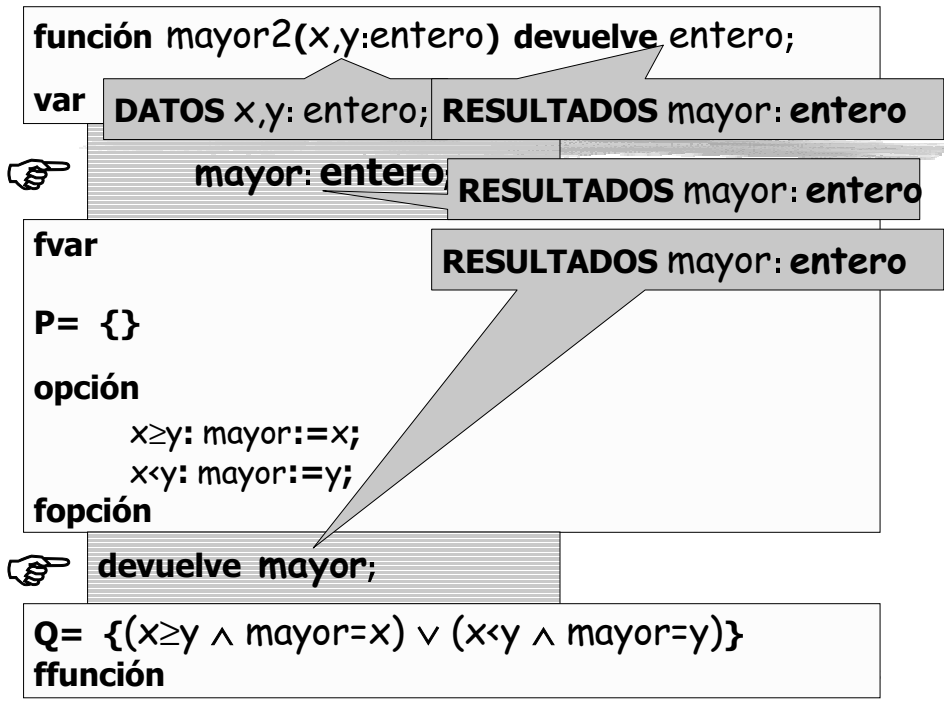
- Etapa 2: **Diseño (y su notación)**

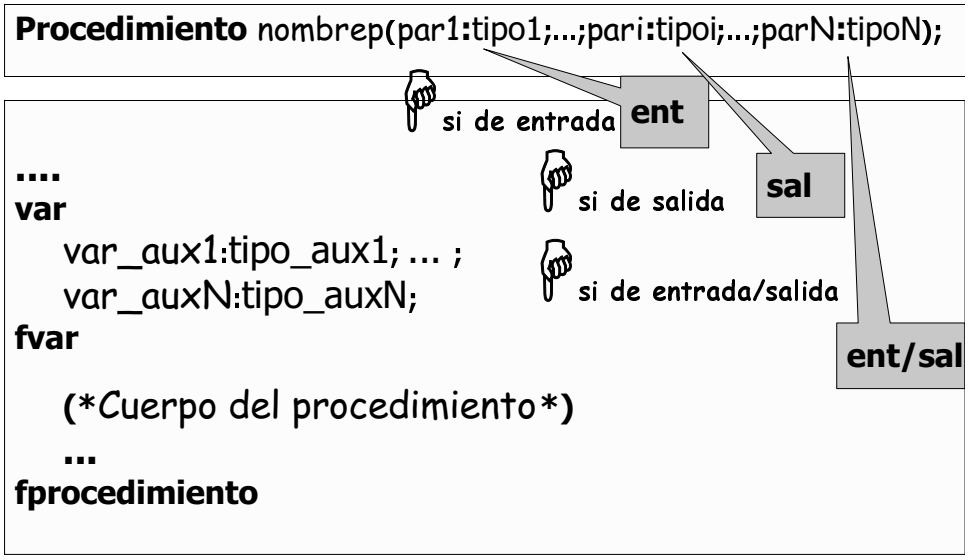
NOTACIÓN para FUNCIÓN:

NOTACIÓN para PROCEDIMIENTO:

16

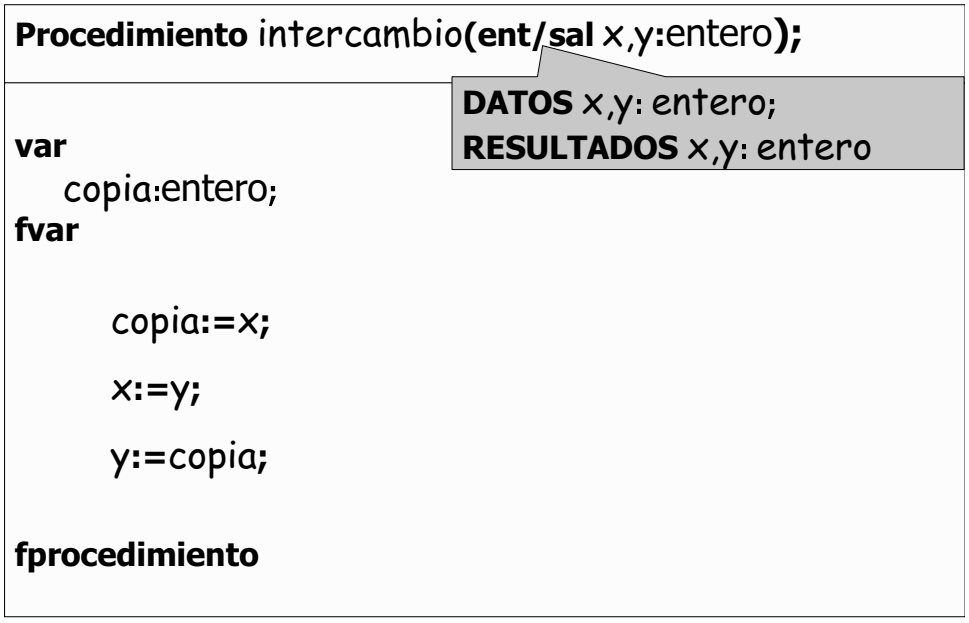

función nombref (par1:tipo1;...;parN:tipoN)
devuelve tipo_función;
....
var
resultado_f: tipo_función;
var_aux1:tipo_aux1; ... ;
var_auxN:tipo_auxN;
fvar
(*Cuerpo de la función*)
...
devuelve resultado_f;
ffunción
NOTACIÓN para FUNCIÓN:


función mayor2(x,y:entero) devuelve entero;
var
DATOS x,y: entero;
RESULTADOS mayor: entero
mayor: entero;
RESULTADOS mayor: entero
fvar
RESULTADOS mayor: entero
P= {}
opción
x≥y: mayor:=x;
x<y: mayor:=y;
fopción
devuelve mayor;
Q= {(x≥y ∧ mayor=x) ∨ (x<y ∧ mayor=y)}
ffunción



NOTACIÓN para PROCEDIMIENTO:

19



20

```
algoritmo tiendas(DATOS l,n1,n2,n3: entero; RESULTADOS ncomb: entero)
 P={l=n1+n2+n3 ∧ n1≥0 ∧ n2≥0 ∧ n3≥0 ∧ n1≥n2≥n3 }
var nc:entero; fvar
```

```
algoritmo fact(DATOS n : entero; RESULTADOS f : entero)
P={n≥0}
var i: entero fvar
        f := 1;
        para i := 1 hasta n hacer f := f * i ; fpara

Q={f=n!}
falgoritmo
algoritmo combina(DATOS x,y: entero; RESULTADOS res: entero)
P={x≥0 ∧ y≥0}
var i, j, k: entero fvar
        i := fact(x);  j := fact(x-y);  k := fact(y);  res := i div (j * k);
Q={res=C(x,y) = x! div(y! *(x - y)!)}
falgoritmo
```

```
combina(l,n1,  ncomb);
combina(l-n1,n2,nc) ; ncomb:=ncomb*nc;
combina(l-n1-n2,n3,nc) ; ncomb:=ncomb*nc;
 Q={ncomb=C(l,n1)*C(l-n1,n2)*C(l-n1-n2,n3)}
falgoritmo
```



```
algoritmo tiendas(DATOS l,n1,n2,n3: entero; RESULTADOS ncomb: entero)
 P={l=n1+n2+n3 ∧ n1≥0 ∧ n2≥0 ∧ n3≥0 ∧ n1≥n2≥n3 }
var nc:entero; fvar
```

```
función fact (n : entero) devuelve entero ;
P={n≥0}
var i : entero ; f : entero ; fvar
        f := 1;
        para i := 1 hasta n hacer f := f * i ;fpara
        devuelve f
Q={f=n!}
ffunción
procedimiento combina (ent x ,y : entero; sal res : entero)
P={x≥0 ∧ y≥0}
var i, j, k: entero fvar
        i := fact(x);  j := fact(x-y);  k := fact(y);  res := i div (j * k);
Q={res=C(x,y) = x! div(y! *(x - y)!)}
fprocedimiento
```

```
combina(l,n1,  ncomb);
combina(l-n1,n2,nc) ; ncomb:=ncomb*nc;
combina(l-n1-n2,n3,nc) ; ncomb:=ncomb*nc;
 Q={ncomb=C(l,n1)*C(l-n1,n2)*C(l-n1-n2,n3)}
falgoritmo
```

```
PROGRAM tiendas (INPUT, OUTPUT);
  VAR l,n1,n2,n3: INTEGER ; ncomb :INTEGER; nc: INTEGER ;
        FUNCTION fact (n : INTEGER): INTEGER;
        VAR i, f : INTEGER;
        BEGIN
                (* P={n≥0} *)
                f :=1;    FOR i :=2 TO n DO f := f * i ;  fact := f
                (* Q={f=n!} *)
        END; { fact }
        PROCEDURE combina (x,y : INTEGER; VAR res: INTEGER);
        VAR i, j, k: INTEGER ;
        BEGIN
                (* P={x≥0 ∧ y≥0} *)
                i := fact(x); j := fact(x-y);  k := fact(y); res := i div (j * k);
                (* Q={res=C(x,y) = x! div(y! *(x - y)!)} *)
        END; { combina }
  BEGIN
        (* Introducción de los datos y comprobación de la precondición *)
        .................
        (* P={l=n1+n2+n3 y n1>=n2>=n3>=0} *)
        combina(l,n1,  ncomb);
        combina(l-n1,n2,nc) ; ncomb:=ncomb*nc;
        combina(l-n1-n2,n3,nc) ; ncomb:=ncomb*nc;
        (* Visualización de los resultados *) .....................
        (* Q={ncomb=C(l,n1)*C(l-n1,n2)*C(l-n1-n2,n3)} *)
  END. { tiendas }
```

23

función nombref
(par$_1$:tipo$_1$; … ;**par$_n$**:tipo$_n$)
devuelve tipo_función;
….

var
resultado_f:tipo_función;
var_aux$_1$:tipo_aux$_1$**;** … **;**
var_aux$_n$:tipo_aux$_n$**;**
fvar

P
(*Cuerpo **función***)
…
Q
devuelve resultado_f;
ffunción

function nombref
(par$_1$:tipo$_1$; … ;**par$_n$**:tipo$_n$):
tipo_función;
const, type, otras funciones y procedimientos
var
resultado_f: tipo_función;
var_aux$_1$:tipo_aux$_1$**;** … **;**
var_aux$_n$:tipo_aux$_n$**;**

Begin
P
(*Cuerpo de la función*)
…
Q
nombref := resultado_f;
End ;

Procedimiento nombre_p (par$_1$:tipo$_1$;...;par$_i$:tipo$_i$;...;par$_n$:tipo$_n$);
....

var
var_aux$_1$:tipo_aux$_1$; ...;
var_aux$_n$:tipo_aux$_n$;
fvar

P
(*Cuerpo del procedimiento***)**
...
Q

fprocedimiento

Procedure nombref_p (par$_1$:**tipo$_1$;**...;par$_i$:**tipo$_i$;**...;par$_n$:**tipo$_n$);**

const, type, otras funciones y procedimientos
var
var_aux$_1$:tipo_aux$_1$; ... ;
var_aux$_n$:tipo_aux$_n$;
Begin
P
(*Cuerpo del procedimiento***)**
...
Q
End ;

2.3. Parámetros formales y reales

SITUACIONES DE UN ALGORITMO PARAMETRIZADO:

② **Invocación** ó **nominación**

☞ **desde el algoritmo complejo que lo usa (programa principal)**

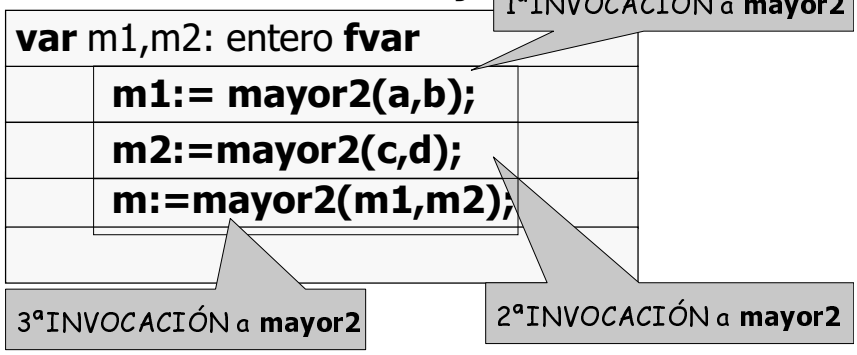
① **diseño** del algoritmo

☞ **para poder ejecutarlo tras su invocación (declaración)**

26

algoritmo mayor4(**DATOS** a,b,c,d: entero;
RESULTADOS m: entero)

P={a≥0 ∧ b≥0 ∧ c≥0 ∧ d≥0 ∧ a≠b≠c≠d**}**



Q={(m≥a ∧ m≥b ∧ m≥c ∧ m≥d) ∧ (m=a ∨ m=b ∨ m=c ∨ m=d)**}**

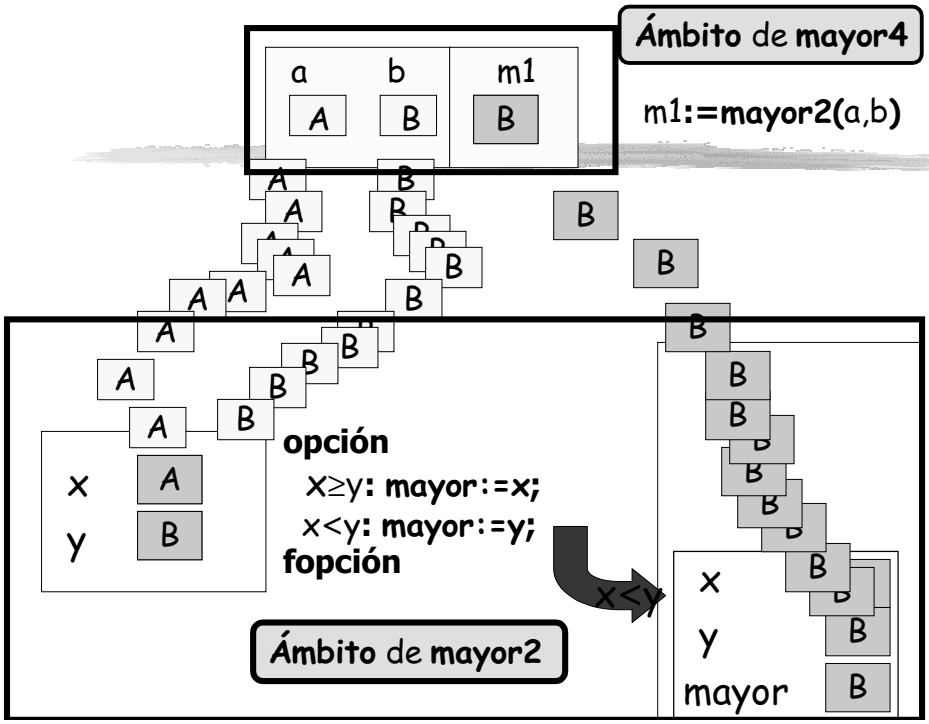
falgoritmo

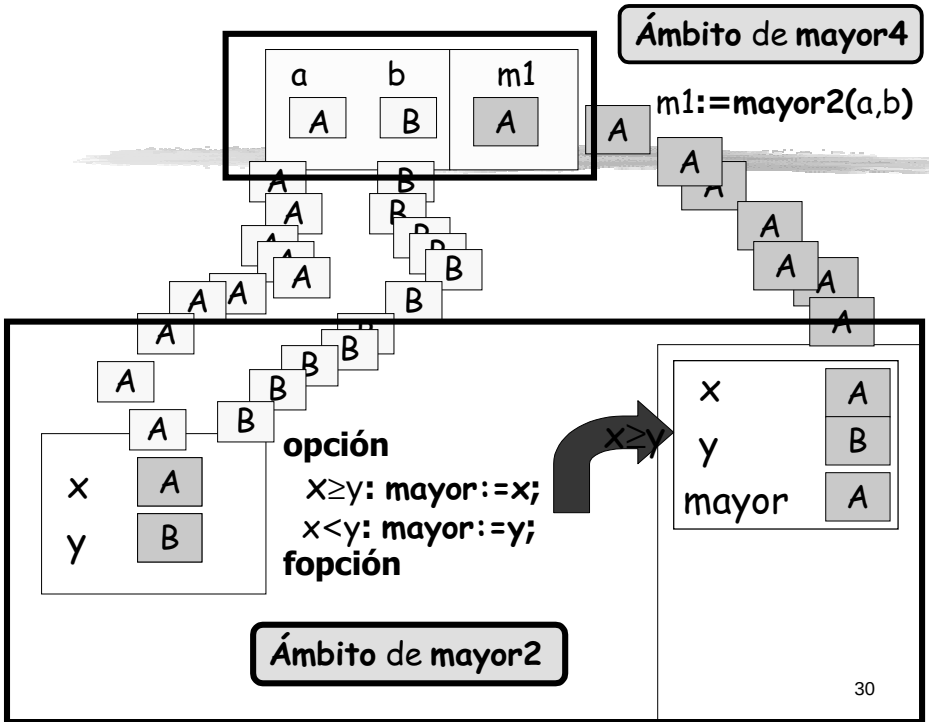
27

algoritmo mayor4(**DATOS** a,b,c,d: entero;
RESULTADOS m: entero)

falgoritmo

28


Ámbito de mayor4
a
b
m1
A
B
B
m1:=mayor2(a,b)
opción
x≥y: mayor:=x;
x<y: mayor:=y;
fopción
x
y
A
B
x<y
Ámbito de mayor2
x
y
mayor
B


Ámbito de mayor4
a
b
m1
A
B
A
m1:=mayor2(a,b)
opción
x≥y: mayor:=x;
x<y: mayor:=y;
fopción
x
y
A
B
x≥y
Ámbito de mayor2
x
y
mayor
A
B
A
30

2.3. Parámetros formales y reales

SITUACIONES DE UN ALGORITMO PARAMETRIZADO ⟺ **SITUACIONES DE SUS PARÁMETROS**

② **Invocación** ó **nominación**

Parámetros reales ("actual");

i.e. a,b:entero

① **Diseño**

Parámetros formales;

i.e. x,y:entero

31

2.3. Parámetros formales y reales

SITUACIONES DE UN ALGORITMO PARAMETRIZADO ⟺ **SITUACIONES DE SUS PARÁMETROS**

② **Invocación** ó **nominación**

Tipo_p_reales
Orden_p_reales
N°_p_reales

=

Tipo_ p_formales
Orden _p_formales
N°_p_formales

① **Diseño**

2.3. Parámetros formales y reales

NOTACIÓN PARA LA NOMINACIÓN

DE FUNCIÓN:

x := nombref(lista_parámetros_efectivos);

DE PROCEDIMIENTO:

nombrep(lista_parámetros_efectivos);

33

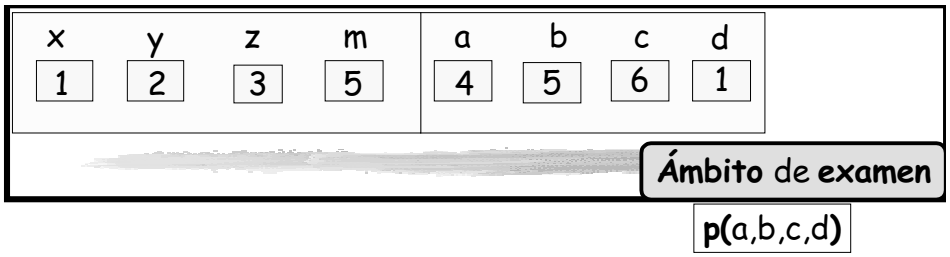
Ejemplo: dado el siguiente procedimiento

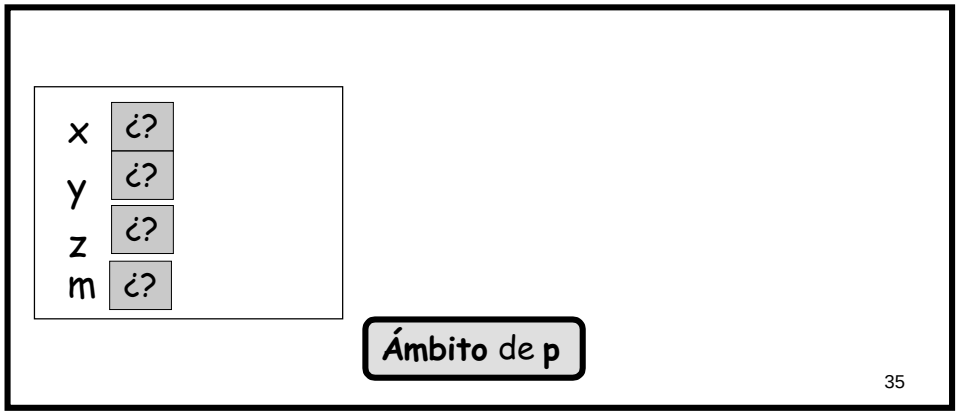
procedimiento p(**ent** x,y:entero; **sal** z,m:entero);
    y :=10* y;
    z:= x+ y ;
    m:=y;
fprocedimiento

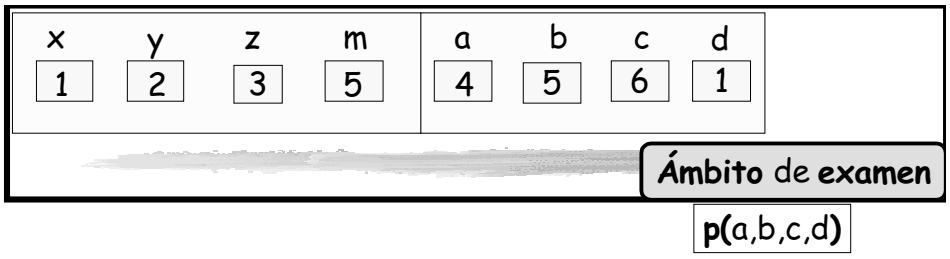
Se pide, dado el estado R_0, completar el estado R_1 obtenido tras la llamada a *p* en el siguiente algoritmo

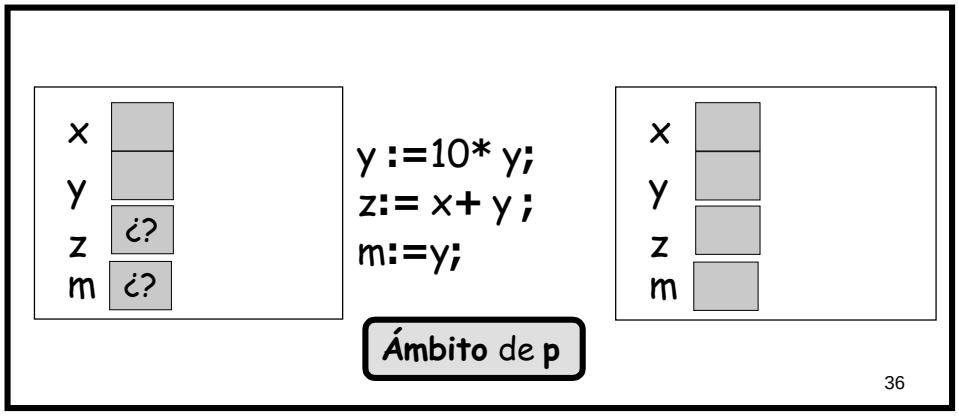
algoritmo examen(**DATOS** a,b,c,d,x,y,z,m: entero;
RESULTADOS a,b,c,d,x,y,z,m: entero)
$\mathbf{R_0}$= {x=1 ∧ y=2 ∧ z=3 ∧ m=5 ∧ a=4 ∧ b=5 ∧ c=6 ∧ d=1}
    p(a,b,c,d);
$\mathbf{R_1}$={x=? ∧ y=? ∧ z=? ∧ m=? ∧ a=? ∧ b=? ∧ c=? ∧ d=?}
falgoritmo

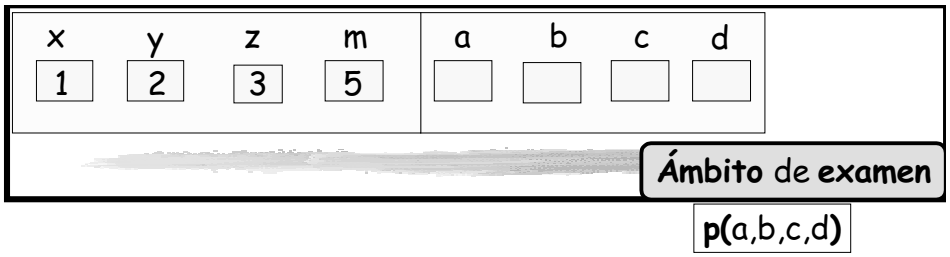
34

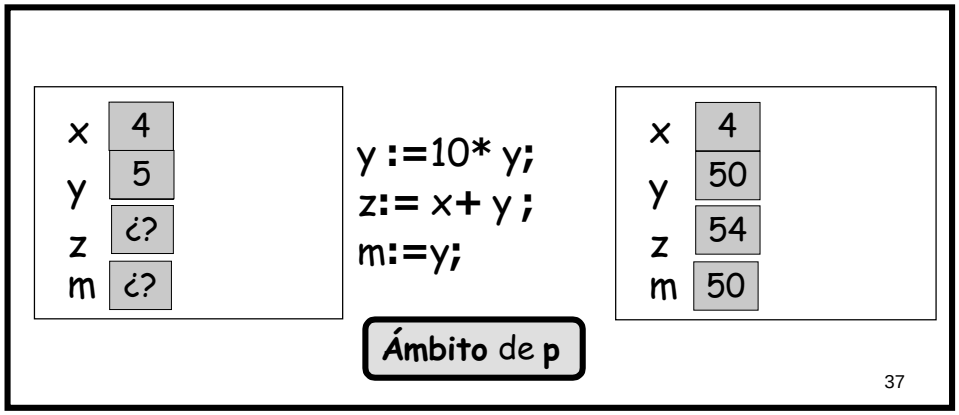

x y z m
1 2 3 5
a b c d
4 5 6 1
Ámbito de examen
p(a,b,c,d)


x ¿?
y ¿?
z ¿?
m ¿?
Ámbito de p
35


x y z m
1 2 3 5
a b c d
4 5 6 1
Ámbito de examen
p(a,b,c,d)


x
y
z ¿?
m ¿?
y :=10* y;
z:= x+ y ;
m:=y;
x
y
z
m
Ámbito de p
36





Ejemplo: dada la definición de la función ***factorial***

función factorial**(**x**:**entero**)** **devuelve** entero**;**
var fact**:**entero **fvar**
P= {x≥0**}**
....
devuelve fact**;**
Q= {fact= $\Pi_{i=1..x}$i**}**
ffunción

diséñese un algoritmo parametrizado ***nuevo_factorial*** que satisfaga la siguiente definición:

algoritmo nuevo_factorial(**DATOS** n**:**entero**;RESULTADOS**

supfact**:**entero**)**

P= { }

Q= {(n≥0 → supfact= $\Pi_{i=1..n}$i) ∧
(n<0 → ((par(n) → supfact= $\Pi_{i=1..|n|}$i) ∧
(impar(n) → supfact= -$\Pi_{i=1..|n|}$i))**}**

38

Versión 1:

función nuevo_factorial**(**n:entero**) devuelve** entero;
var supfact:entero **fvar**
P= { }

devuelve supfact;
Q= {$(n \geq 0 \rightarrow supfact=\Pi_{i=1..n} i) \wedge$
$(n<0 \rightarrow ((par(n) \rightarrow supfact=\Pi_{i=1..|n|} i) \wedge$
$(impar(n) \rightarrow supfact=-\Pi_{i=1..|n|} i))$**}**
ffunción

39

Versión 2:

función nuevo_factorial**(**n:entero**) devuelve** entero;
var supfact:entero **fvar**
P= { }

devuelve supfact;
Q= {$(n \geq 0 \rightarrow supfact=\Pi_{i=1..n} i) \wedge$
$(n<0 \rightarrow ((par(n) \rightarrow supfact=\Pi_{i=1..|n|} i) \wedge$
$(impar(n) \rightarrow supfact=-\Pi_{i=1..|n|} i))$**}**
ffunción

40

2.4. Criterios de selección de parámetros

• Detectar las informaciones **verdaderamente relevantes** del problema

• Distinguir bien entre **datos**, **datos-resultados** y **resultados**

* Cuanto **más** se parametriza más se independiza el algoritmo del contexto y, en consecuencia,

más reutilizable resulta

* Cuanto **menos** se parametriza con mayor precisión se adapta el algoritmo al contexto en el que se introduce y, en consecuencia,

menos reutilizable resulta

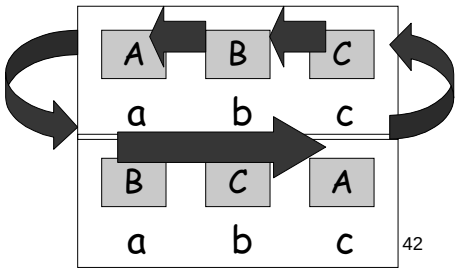
Ejemplos propuestos

Ejemplo: escribir un algoritmo que desplace circularmente los valores enteros asociados a tres variables a, b y c

algoritmo desplaza(**DATOS**
RESULTADOS **)**

P=

Q=



42

Ejemplos propuestos

Ejemplo: se dispone del siguiente algoritmo

algoritmo multpar(**DATOS** a,b**:**entero**;RESULTADOS**c**:**entero**)**

P= {par(a)**}**

Q= {c=a*b**}**

que se parametriza como **procedimiento**

Se pide diseñar un nuevo procedimiento **mult** para multiplicar dos números enteros cualesquiera

*a partir de **multpar**

sin hacer uso de la operación producto "" propia del tipo entero

43

1.- Definición de mult

algoritmo mult(DATOS RESULTADOS)

P=

Q=

2.- Parametrización y diseño de mult

44

Práctica 3, parte 2

El servicio de informática de una empresa se encuentra en la primera fase de desarrollo de una aplicación concerniente a la gestión de su personal. La información que se necesita de cada uno de los empleados es la siguiente:

- Número del D.N.I
- Nombre
- Dirección
- Fecha de ingreso en la empresa
- Sueldo, distinguiendo entre el salario base, complemento de destino y complemento específico

Con la finalidad de llevar a buen fin el proyecto, su responsable nos ha pedido que desarrollemos y probemos un programa en Pascal que, adecuándolo posteriormente, formará parte de la aplicación, y que debe abarcar los siguientes aspectos:

45

*Definición de las estructuras de datos necesarias para modelización del **concepto** empleado

*Implementación de una serie de subprogramas. A saber .....

*Implementar el programa principal, a través del cual se puedan ejecutar y probar cada uno de los subprogramas planteados

Hay que tener en cuenta que cada uno de los subprogramas enunciados posteriormente serán utilizados en la aplicación, por lo tanto deben de diseñarse persiguiendo la máxima independencia con respecto al programa principal

Ejemplos de empleado

	Empleado1	**Empleado2**	**Empleado3**
D.N.I	25444232	31212778	78654459
Nombre	José García	Javier Galán	Jorge Pérez
Fecha ingreso	2/3/1995	12/6/1995	3/3/1995
Sueldo:	150000	110000	180000
	80000	75000	50000
	20000	15000	18000

46

*Implementación de un subprograma que dadas dos fechas permita decidir si la primera es anterior o igual a la segunda (asumiendo que las dos fechas son correctas)

Por ejemplo, si la fecha a partir de la cual se debe de realizar el cálculo es 31/4/1995, y el empleado es Javier Galán, contratado el 12/6/1995 entonces el resultado que se espera obtener es que José ha sido contratado con posterioridad; i.e. la fecha dada SÍ es anterior (o lo misma) a la de contratación del empleado

*Implementación de un subprograma que dada una fecha indique si es **válida**

Por ejemplo, si la fecha a partir de la cual se debe de realizar el cálculo es 32/4/1995 la fecha NO es válida; si la fecha dada es 30/4/1995 SÍ es válida

47