Slide 1

TEMA 8: SUBPROGRAMAS: FUNCIONES

Índice

Slide 2

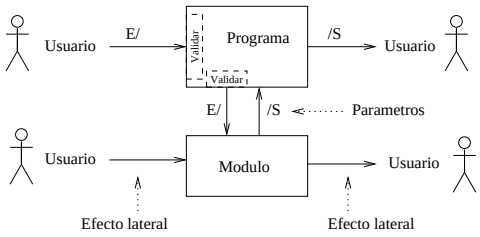
subalgoritmo (**módulo**)	**subprograma**
acción o función no elemental que se "contrata"	procedimiento o función definida por el programador

- problemas complejos: elevan el nivel del lenguaje
- economía de trabajo: reutilización
- economía de trabajo: distribución
- corrección: prueba individual
- mantenimiento: modificación localizada
- claridad de diseño: legibilidad

Slide 3

Subprogramas

- tienen un nombre
- se comunican vía *parámetros*. Otra comunicación: "*efecto lateral*"



- son responsables de *cumplir la especificación*:

  si los argumentos de entrada cumplen la precondición
  entonces los de salida cumplirán la postcondición

- tipo *función* (cálculo) o *procedimiento* (acción)

Slide 4

Subprogramas. Pistas indicadoras de calidad

- nombre de elección casi evidente. Sintagma nominal para funciones, verbo para procedimientos. Sin conjunciones.
- código breve ($\simeq$ una página máximo), pero no demasiado breve (mínimo $\simeq$ 4 líneas).
- número de parámetros moderado.
- tarea dependiente únicamente de los parámetros (eventualmente puede no haberlos)
- efectos laterales: los mínimos posibles
- en general: máxima *cohesión* (fuerzas que unen el interior del subprograma) y mínimo *acoplamiento* (dependencia entre subprogramas).

Slide 5

```
program DiasDeMes (input, output);
var m, a : integer;
(* definición de función bisiesto *)
...
BEGIN
   write ('Escriba mes año:'); readln (m, a);
   (* Se supone entrada correcta *)
   case m of
        1,3,5,7,8,10,12 : writeln ('31 días');
              4,6,9,11 : writeln ('30 días');
                     2 : if bisiesto (a) then
                            writeln ('29 días')
                         else
                            writeln ('28 días')
   end (* case *)
END.
```

Slide 6

```
function bisiesto (anyo: integer) : boolean;
(* Pre: anyo >= 1583 *)
(* bisiesto = true sii el anyo es bisiesto *)
begin
   if (anyo mod 4 = 0) and (anyo mod 100 <>0)
      or
      (anyo mod 400 = 0)
   then bisiesto := true
   else bisiesto := false
end;
```

Slide 7

Nombre	**Entradas**	**Valor devuelto**	**Otros efectos**	**Objetivo**
bisiesto	entero >= 1583	lógico	-	cierto sii año bisiesto
maximo	2 enteros	entero	-	máximo de los números
MCD	2 enteros positivos	entero positivo	-	Máximo Común divisor
potencia	real x, entero n no ambos nulos	real	-	x^n
PrimDiv	entero >1	entero	-	1^{er}divisor>1
DiaSig	enumerado	enumerado	-	día siguiente

Slide 8

FUNCIONES

Definición: (Una. Antes del código del algoritmo que la usará)

function *<id-función>* (*<lista-parámetros>*) **:** *<id-tipo>* **;**
<secciones de constantes, variables etc locales >
begin
<sentencias; asignación a <id-función> >
end ;

<lista-parámetros> ::= *<id-par>* { , *<id-par>* } : *<id-tipo>*
{ ; *<id-par>* { , *<id-par>* } : *<id-tipo>* }

Uso: expresión (varios usos)

<id-de-función> (*<lista de expresiones argumentos>*)

parámetros formales $\overset{\text{posición}}{\longleftrightarrow}$ parámetros actuales (argumentos)

Slide 9

FUNCIONES

- *Definición*
  - *Declaración*
    - **nombre**: identificador
    - **tipo**: de parámetros y resultado
  - *Asignación* de **valor**: algoritmo de cálculo
- *Uso*: en expresiones; significa el valor calculado con los argumentos

Entrada: los argumentos (en los parámetros)
Salida: el valor devuelto
Precondición: sobre los argumentos

Tipo del resultado : simple

Algoritmo de cálculo: usa los parámetros y los elementos locales

Obtención del resultado (valor devuelto): asignación al identificador de función:

$<$*id-función*$>$ **:=** $<$*expresión*$>$

Slide 10

Nombre	**Entradas**	**Valor devuelto**	**Otros efectos**	**Objetivo**
DiaEnum	entero en $[1,7]$	enumerado	-	enumerado asociado
maximo	2 enteros	entero	-	máximo de los números
unired	-	real	-	unidad de redondeo
leidoPos	-	entero >0	lee de teclado escribe en p.	entero >0 del usuario
leidoRan	p,q enteros	entero en $[p,q]$	lee de teclado escribe en p.	entero $\in [p,q]$ del usuario

Slide 11

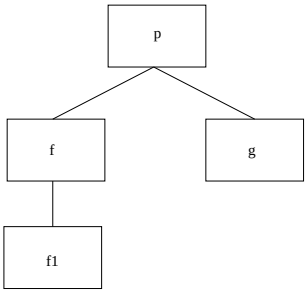
```
function unired : real;
var uni : real;
begin
  uni := 1.0;
  while 1.0 + uni > 1.0 do uni := uni/2;
  unired := uni
end;
```

```
function unired : real;
var uni : real;
begin
  uni := 1.0;
  while 1.0 + uni > 1.0 do
      begin uni := uni/2; write (.) end
  unired := uni
end;
```

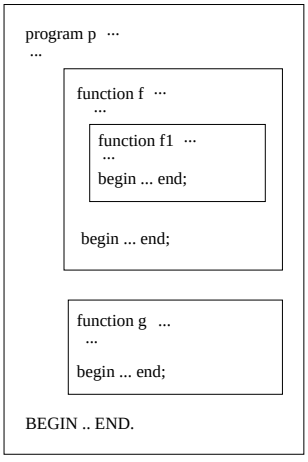
Slide 12

Anidamientos de funciones

Estructura de uso (diseño)



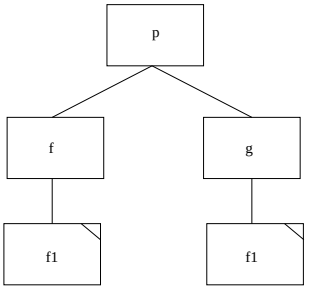
Estructura textual (codificación)



Slide 13

Anidamientos de funciones

Estructura de uso (diseño)



Estructura textual (codificación)

program p ···
···

function f1 ···
···
begin ... end;

function f ···
···
begin ... end;

function g ...
···
begin ... end;

BEGIN .. END.

Slide 14

```
program p (input, output);
   var a, b : real;  (* VARIABLES GLOBALES *)
   function f (x : real): real;
    var c : integer; (* VARIABLES LOCALES A f *)
   begin
       c := 7;
       a := a + c;
       f := 10*x
   end;
BEGIN
   a := 1; b := 2;
   writeln (a, b);     (* salida: 1 2 *)
   b := f(5);          (* multiplicar 5 por 10 *)
   writeln (a, b);     (* salida: ¡8! 50 *)
END.
```

Slide 15

Ámbito (de una declaración) parte de programa en la que la declaración de un objeto (constante, tipo, variable, función, procedimiento) tiene efecto

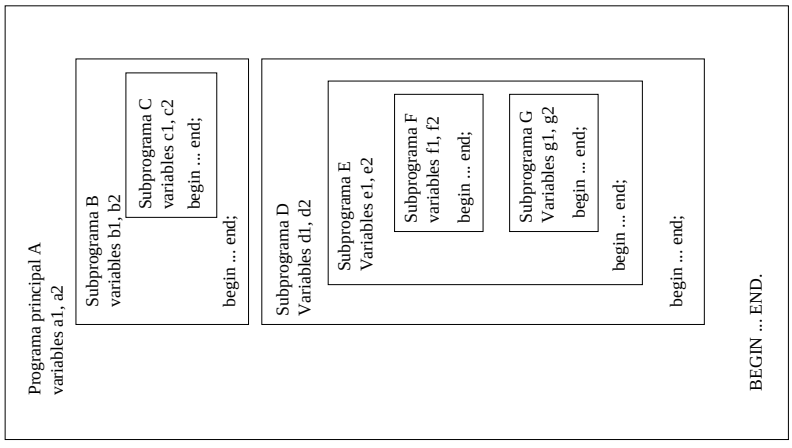
Área de visibilidad ("scope") (de un objeto): parte de un programa en la que se puede acceder al objeto mediante su nombre.

Un objeto es visible en algunas regiones del ámbito de su declaración.

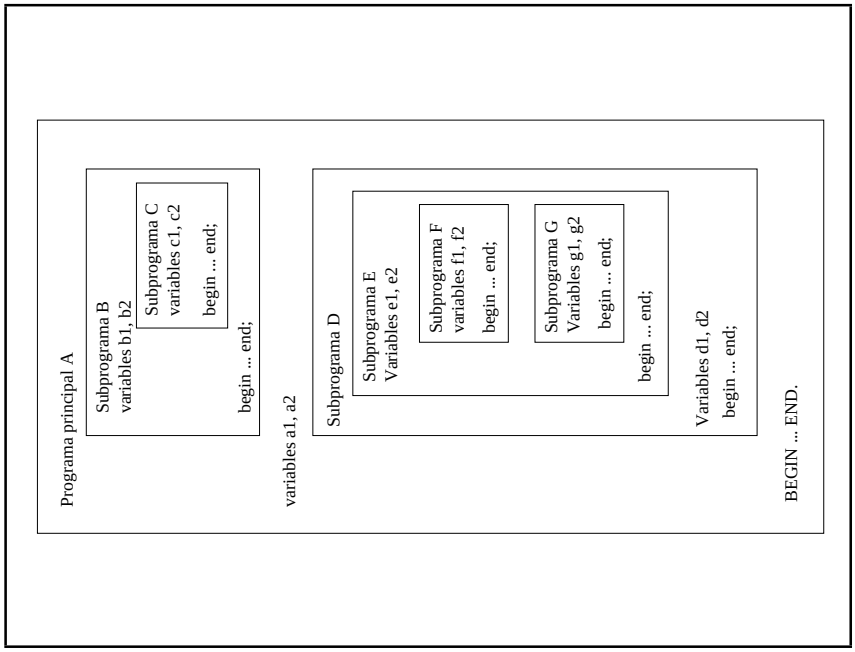
Vida de un objeto: tiempo en el que una entidad se mantiene en memoria

Principio fundamental del ámbito El ámbito de un objeto es el programa o subprograma que lo declara

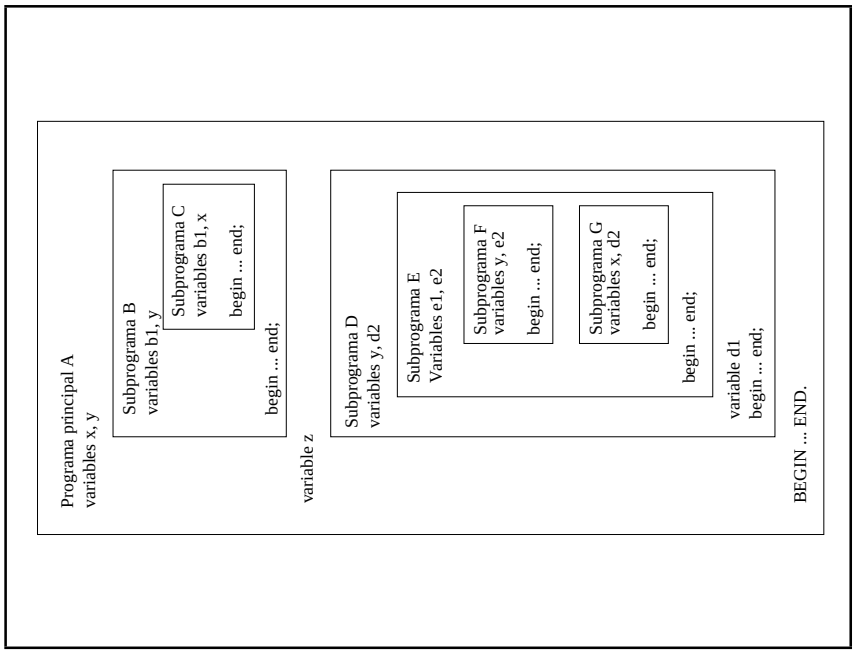
Slide 21



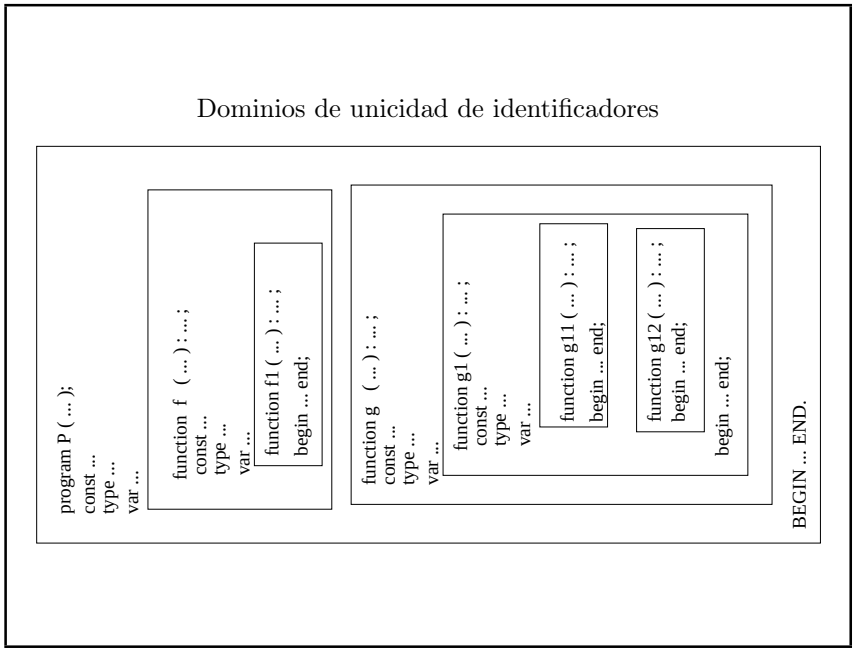
Slide 22



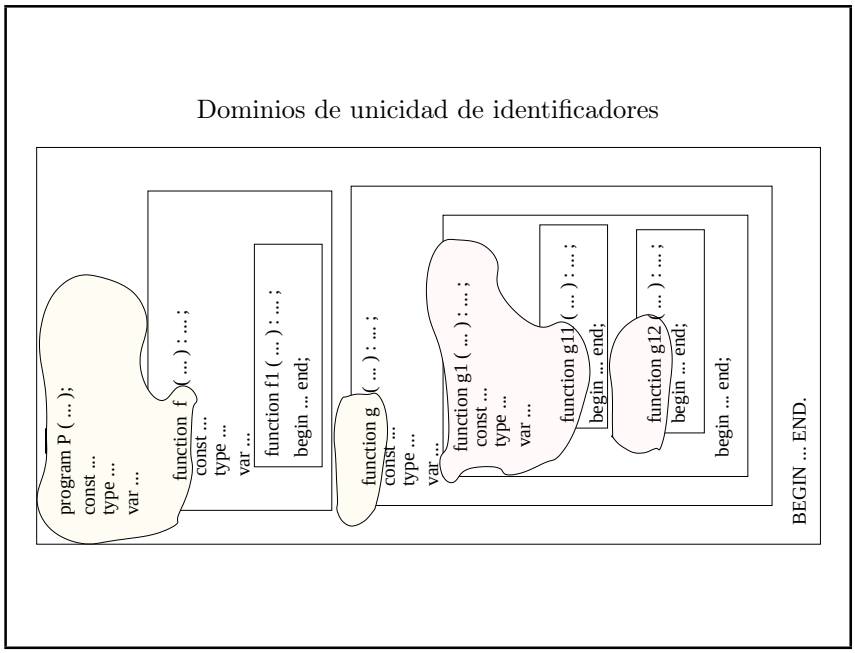
Slide 23



Slide 24



Slide 25



Slide 26

```
type tDiaSemana = (lun, mar, mie, jue, vie, sab, dom);
...
function diaEnum (n: integer):tDiaSemana;
(* Pre: 1<=n<=7 *)
(* Devuelve el día (enumerado) correspondiente *)
var i : integer;
    dia : tDiaSemana;
begin
    dia := lunes;
    for i := 2 to n do
        dia := succ(dia);
    diaEnum := dia
end;
```

Slide 27

```
type tDiaSemana = (lun, mar, mie, jue, vie, sab, dom);
...
function diaEnum (n: integer):tDiaSemana;
(* Pre: 1<=n<=7 *)
(* Devuelve el día (enumerado) correspondiente *)
var dia : tDiaSemana;
begin
    (*1*) dia := lunes;
    (*2*) while n>1 do
            begin
              (*3*) dia := succ(dia);
              (*4*) n := n-1 (*5*)
          end;
    (*6*) diaEnum := dia (*7*)
end;
```

Slide 28

Regla de ámbito 1 *Un objeto declarado dentro de un subprograma no es accesible fuera de él.*

Regla de ámbito 2 *Se puede acceder a un objeto global en el programa principal y en cualquier subprograma que no tenga un objeto local con el mismo nombre (que el global).*

Regla de ámbito 3 *Se puede acceder a un objeto declarado en un subprograma desde cualquier otro definido dentro de éste, siempre que no se haya declarado un objeto con el mismo nombre en el subprograma interior.*

Regla de ámbito 4 *Dados dos subprogramas A y B definidos en el mismo nivel de un programa o subprograma, A puede utilizar a B si A está definido después de B.*

Regla de ámbito 5 *El ámbito de una declaración de un objeto comprende desde el punto de declaración hasta el final del programa o subprograma donde se realiza la declaración.*

Slide 29

Regla de ámbito 6 *Resumen: un identificador en un subprograma se refiere al objeto que primero aparezca con tal identificador de la siguiente lista*

1. *las variables y objetos locales al subprograma, los parámetros y el identificador del subprograma*
2. *las variables y otros objetos locales del subprograma de nivel exterior inmediato declaradas en líneas previas así como los parámetros e identificador de dicho suprograma*
3. *las variables y otros objetos locales del subprograma de siguiente nivel exterior inmediato declaradas en líneas previas así como los parámetros e identificador de dicho suprograma*
4. *. . . (sucesivamente hasta)*

n. las variables y objetos globales declarados en líneas previas, así como los parámetros y el identificador del programa