Programación I

Temas

- ✓ Repaso
- ✓ Comunicación entre módulos
- ✓ Ejemplos

Repaso

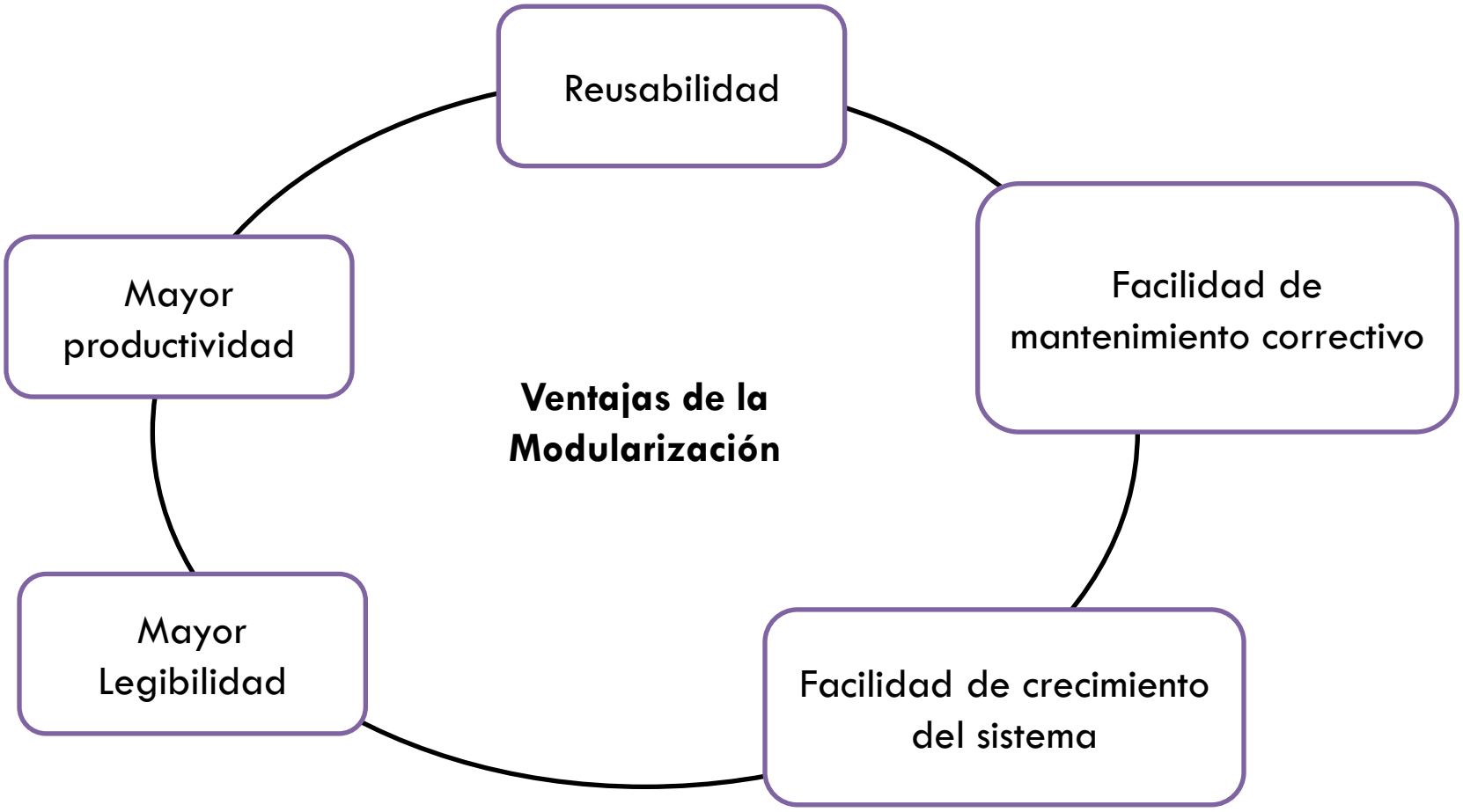
✓ ¿Qué es la modularización?

Modularizar es una estrategia que implica dividir un problema en partes funcionalmente independientes, que encapsulen operaciones y datos

✓ ¿A qué llamamos módulo?

Es un conjunto de instrucciones que cumplen una tarea específica bien definida, se comunican entre sí adecuadamente y cooperan para conseguir un objetivo común

Repaso


Reusabilidad
Facilidad de mantenimiento correctivo
Facilidad de crecimiento del sistema
Mayor Legibilidad
Mayor productividad
Ventajas de la Modularización

Alcance de una variable

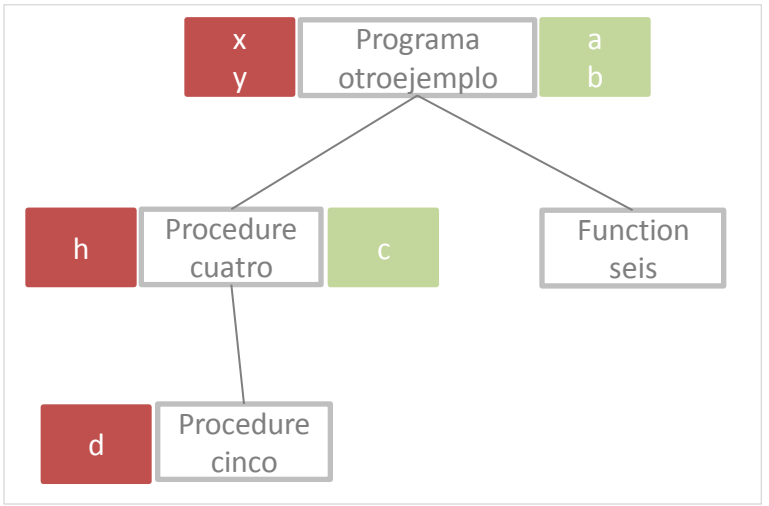
- ✓ Establece el contexto donde la variable es conocida o puede ser referenciada en el marco de un programa.
- ✓ La variable puede ser de alcance: GLOBAL O LOCAL

Alcance de los datos

```
program otroejemplo;
var a, b: integer;

  procedure cuatro (parámetros formales);
   var c: char;
    procedure cinco (parámetros formales);
       var d: real;
       begin
         b := a * 2 + 4; d:= 2,5;
         writeln(a, b, c, d);
       end;
   var h: integer;
    begin
      b := a * 2;
      cinco (parámetros actuales);
      c:= 'A';
      writeln(a, b, c, h);
      h:= seis (parámetros actuales);
    end;

 Function seis (parámetros formales): tipo;
  begin
    a:= a+b;
    seis:= (a+c)/100;
  end;

   Var x, y: integer;
   begin
     x:= 5; y:= 20 mod 10;
     cuatro (parámetros actuales);
     cinco (parámetros actuales);
     a:= seis (parámetros actuales);
     writeln(x, y);
     writeln(a, b);
   end.
```

Supongamos que identificamos:

- variables locales
- variables “globales”



Teniendo en cuenta el alcance de las variables y la visibilidad de los módulos, analizar:

¿Qué invocaciones son válidas?

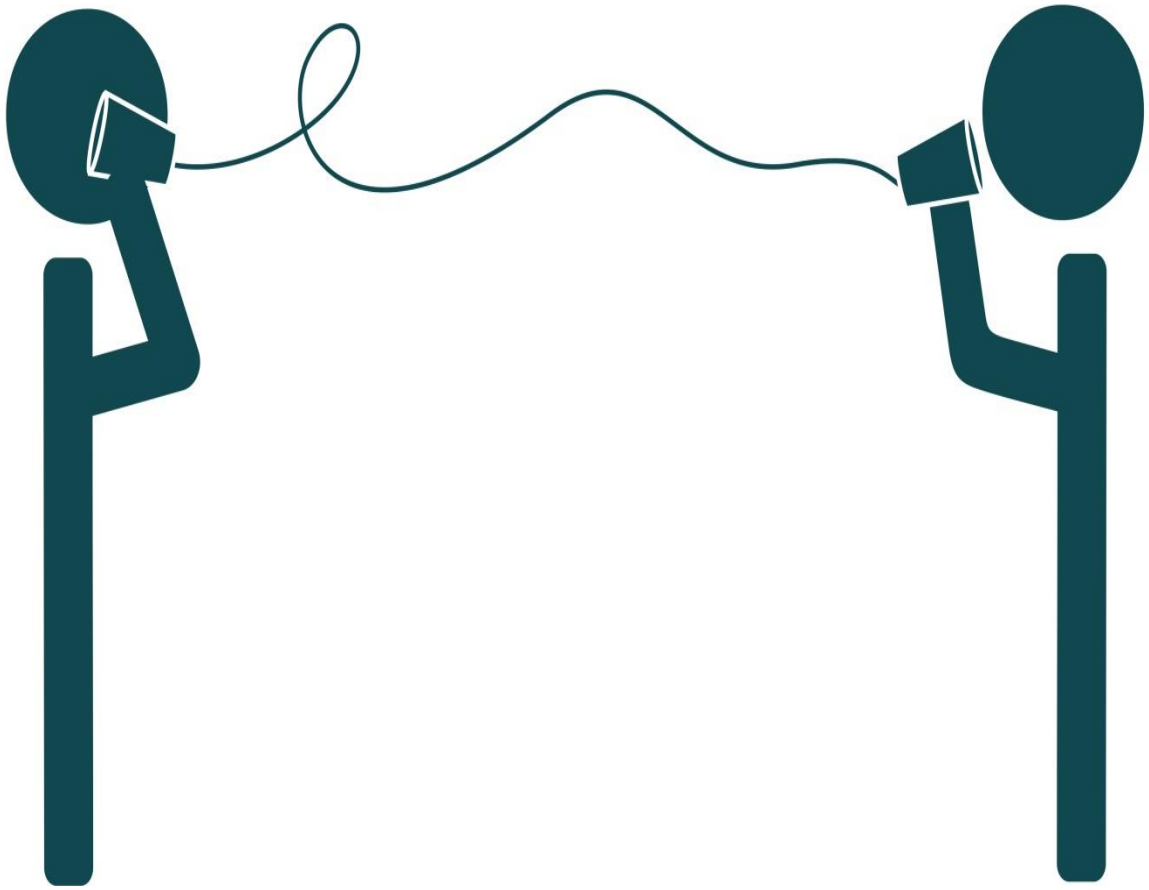
¿Es válidala utilización de las variables en el programa principal?

¿Y en el procedure cuatro?

¿Y en el procedure cinco?

¿Y en la function seis?

COMUNICACIÓN ENTRE MÓDULOS



Modularización - Comunicación

¿CÓMO SE COMUNICAN LOS DATOS ENTRE MÓDULOS Y PROGRAMA?

VARIABLES GLOBALES

NO ACONSEJABLE. PUEDE TRAER EFECTOS COLATERALES. PERJUDICA LA LEGIBILIDAD

PARÁMETROS

INDICAN DE MANERA EXPLÍCITA QUÉ DATOS UTILIZARÁ EL MÓDULO

Desventajas de la comunicación a través de variables globales

- Posibilidad de perder integridad de los datos, al modificar involuntariamente en un módulo alguna variable que luego deberá utilizar otro módulo. ¿Independencia? ¿Reusabilidad?

- Dificultad durante la etapa de la verificación

- Demasiadas variables en la sección de declaración

- Posibilidad de conflicto entre los nombres de las variables utilizadas por diferentes programadores

- Falta de especificación del tipo de comunicación entre los módulos

- Uso de memoria

No se recomienda el uso de variables globales

Se recomienda una solución que combina:

OCULTAMIENTO DE DATOS Y PARAMETROS

- El **ocultamiento de datos** significa que los datos exclusivos de un módulo o programa **NO** deben ser "visibles" o utilizables por los demás módulos.

- El **uso de parámetros** significa que los datos compartidos se deben especificar como parámetros que se trasmiten entre módulos.

Variables locales del módulo

Variables locales del programa

Los datos propios del módulo se declararan locales al módulo

Los datos propios del programa se declararan locales al programa

Los datos compartidos se declararán como parámetros.

No nos sirven para la comunicación!!

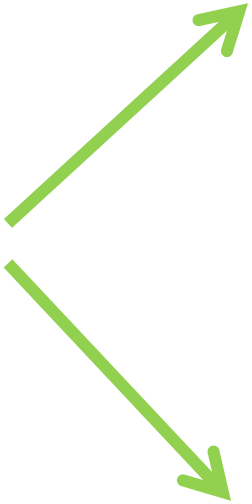
El parámetro se define como una variable que representa un dato compartido entre módulos o entre un módulo y el programa principal.

Por lo tanto, el dato compartido se especificará como un parámetro que se trasmite entre los módulos.

¿Cómo se implementan los parámetros en Pascal?

Modularización - Comunicación

La comunicación mediante parámetros puede ser **en Pascal**



- Parámetros por valor
- Parámetros por referencia

Parámetros por valor

Comunicación – Parámetros por valor

- Un dato de entrada por valor es llamado parámetro **IN** y significa que el módulo recibe (sobre una variable local) un valor proveniente de otro módulo (o del programa principal).

- Con él se pueden realizar operaciones y/o cálculos, pero no producirá ningún cambio ni tampoco tendrá incidencia fuera del módulo.

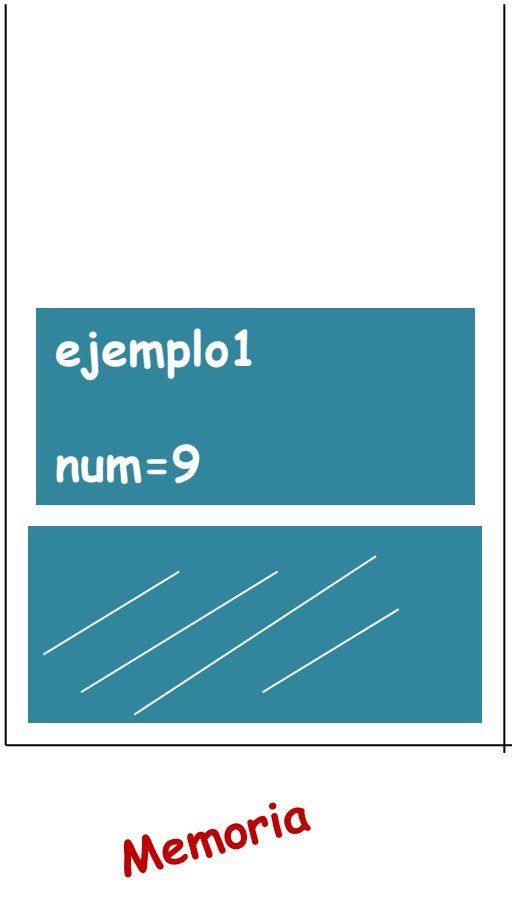
Parámetro por valor

Un parámetro por valor es un dato de entrada que significa que el módulo recibe una copia de un valor proveniente de otro módulo o del programa principal.

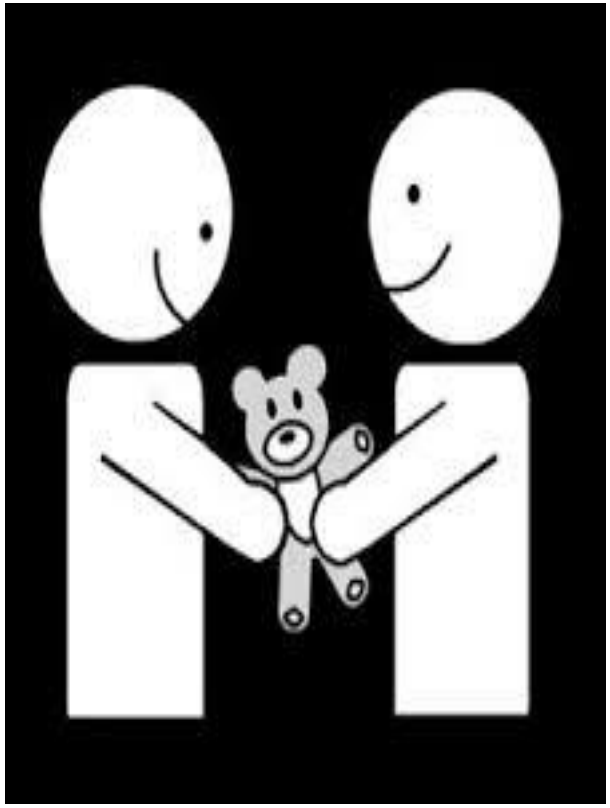
Con este dato el módulo puede realizar operaciones y/o cálculos, pero fuera del módulo ese dato NO reflejará cambios.

```
Program ejemplo1;

Procedure uno (x:integer);
  Begin
    x:= x+1;
    write (x);
  End;

var num: integer;

Begin
  num:=9;
  uno (num);
  write (num);
End.
```

¿Qué valores imprime?



Parámetros por referencia



Comunicación – Parámetros por referencia

La comunicación por referencia significa que **el módulo recibe la dirección de memoria** de una variable conocida en el punto de invocación.

Dentro del módulo se puede operar con el valor original contenido en esa dirección de memoria, y las modificaciones que se produzcan se reflejan en los demás módulos que conocen la variable.

En el encabezado del módulo se distinguen por tener la palabra clave VAR.

Parámetro por referencia

El módulo que recibe este parámetro puede operar con la información que se encuentra en la dirección de memoria compartida y las modificaciones que se produzcan se reflejarán en los demás módulos que conocen esa dirección de memoria compartida.

```
Program ejemplo2;

Procedure dos (var x:integer);
  Begin
    x:= x+1;
    write (x);
  End;

var num: integer;
Begin
  num:=9;
  dos (num);
  write (num);
End.
```

¿Qué valores imprime?

```
Program ejemplo1;

Procedure uno (x:integer);
  Begin
    x:= x+1;
    write (x); {1}
  End;

var num: integer;

Begin
  num:=9;
  uno (num);
  write (num);   {2}
End.
```

En {1} se muestra 10 como valor de x

En {2} se muestra 9 como valor de x

```
Program ejemplo2;

Procedure dos (var x:integer);
  Begin
    x:= x+1;
    write (x);    {3}
  End;

var num: integer;

Begin
  num:=9;
  dos (num);
  write (num);    {4}
End.
```

En {3} se muestra 10 como valor de x

En {4} se muestra 10 como valor de x

¿Cómo se explican los valores que se imprimen?

Consideraciones...

Un **parámetro por valor** debe ser tratado como una variable local del módulo.

La utilización de este tipo de parámetros puede significar una utilización importante de memoria.

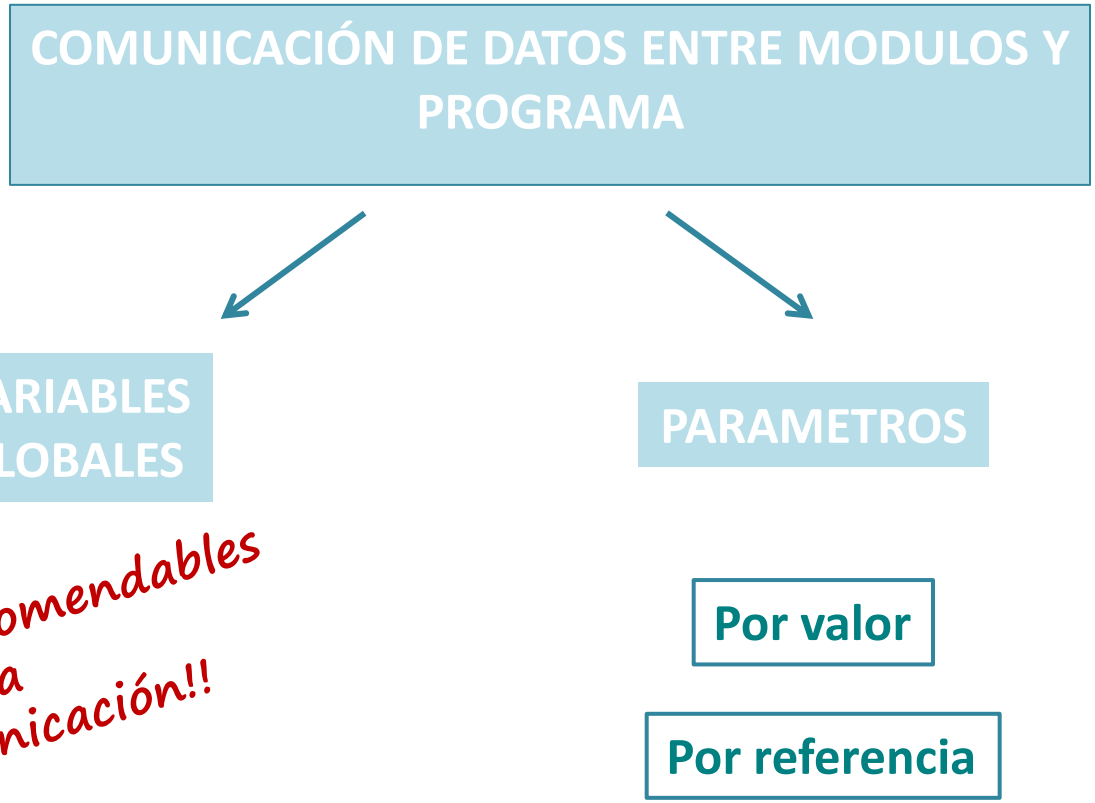
Los **parámetros por referencia** en cambio operan directamente sobre la dirección de la variable original, en el contexto del módulo que llama.

Esto significa que no requiere memoria local.

Variables globales vs Parámetros

Variables globales	Parámetros
▪ Posibilidad de perder integridad de los datos, al modificar involuntariamente en un módulo alguna variable que luego deberá utilizar otro módulo ▪ Posibilidad de conflicto entre los nombres de las variables utilizadas por diferentes programadores ▪ Dificultad durante la etapa de la verificación	✓ Los módulos no pueden afectar involuntariamente los datos de otros módulos. ¡Independencia! ¡Reusabilidad!
▪ Demasiadas variables en la sección de declaración	✓ Se reduce significativamente la cantidad de variables globales
▪ Falta de especificación del tipo de comunicación entre los módulos	✓ Permite distinguir el tipo de comunicación
▪ Uso de memoria	✓ Los parámetros por referencia no utilizan memoria local ✗ El uso de parámetros por valor puede significar una importante utilización de memoria local

En resumen


COMUNICACIÓN DE DATOS ENTRE MODULOS Y PROGRAMA
VARIABLES GLOBALES
PARAMETROS
No recomendables para la comunicación!!
Por valor
Por referencia

Consideraciones generales

Cuando se invoca a un módulo se deben tener cuenta las restricciones propias del lenguaje de implementación. En Pascal:

- La invocación al módulo (parámetros actuales)
- La interface del módulo (parámetros formales)
- Los parámetros actuales y formales

Deben coincidir en cantidad y tipo de dato

Se relacionan 1 a 1

```
Program ejemplo3;
Procedure Calcular (x, y: integer;
        var suma, prod: integer);
begin
  suma:= x + y;
  prod:= x * y;
end;

Var  val1, val2, s, p: integer;
Begin
  Calcular (6, 17, s, p);

  val1:= 10; val2:= 5;
  Calcular (val1, val2, s, p);

  Calcular (23, val2, 14, p);
End.
```

Ejercitación

```
Program Ejemplo4;
  procedure Imprimir (var a:integer; b: integer; c: integer);
  begin
    writeln ('a= ', a, ' b= ', b, ' c= ', c);
    writeln;
    a := 10;
    c := a + b;
    b := b * 5;
    writeln ('a= ', a, ' b= ', b, ' c= ', c);
    writeln;
  end;

var x, y, z: integer;

begin
  y := 6;
  z := 5;
  writeln ('x= ', x, ' y= ', y,' z= ', z);
  writeln;
  imprimir (x, y, z);
  writeln ('x= ', x, ' y= ', y,' z= ', z);
  readln;
end.
```

Ejercitación

```
program ejemplo5;
 procedure cuenta(var b: integer; var a: integer);
  procedure calculo (var b: integer; a: integer);
  begin
     b := a * 2 + 4;
     writeln('a= ', a, ' b= ', b);
     writeln;
  end;
 var c: integer;
 begin
  b := a * 2;
  calculo(a,b);
  c:= 0;
  writeln('a= ', a, ' b= ', b, ' c= ', c);
  writeln;
 end;
Var a, b,c: integer;
begin
 a:= 5;
 b:= 20 div 10;
 writeln('a= ', a, ' b= ', b, ' c= ', c);
 writeln;
 cuenta(c, a);
 writeln('a= ', a, ' b= ', b, ' c= ', c);
 writeln;
 readln;
end.
```

Observar la declaración de la variable local c

¿Qué imprime en cada sentencia write?

Ejercitación

```
Program Ejemplo6;
var a: integer;

   procedure Imprimir (var b: integer; c: integer);
   begin
      writeln('a= ', a, ' b= ', b, ' c= ', c);
      writeln;
      a := a + 5;
      c := a + b;
      b := b * 5;
      writeln('a= ', a, ' b= ', b, ' c= ', c);
      writeln;
   end;

var b, c: integer;
begin
   a := 20;
   b := 6;
   writeln('a= ', a, ' b= ', b, ' c= ', c);
   writeln;
   imprimir (b, a);
   writeln('a= ', a, ' b= ', b, ' c= ', c);
   writeln;
   readln;
end.
```

¿Qué imprime en cada sentencia write?

Ejercitación

¿Qué imprime en cada sentencia write para la siguiente secuencia?
10 15 3 -5 24

```
Program ejemplo7;

procedure calcular ( n: integer; var cant, aux: integer);
Begin
   cant := cant + 1;
   if (n mod 2 = 0) then aux := aux+1;
end;

Var
   a,c, num: integer;

begin
  writeln ('a= ',a, ' c= ', c);
  writeln;
  c:=0;
  a:=0;
  write ('Ingrese un numero (para finalizar 24): ');
  readln (num);
  while (num <> 24) do begin
    calcular (num,c, a);
    write ('Ingrese un numero (para finalizar 24): ');
    readln (num);
  end;
  writeln;
  writeln ('a= ',a, ' c= ', c);
  readln;
end.
```

Ejercitación

```
Program queimprime;
  Var
     c, n, num1: integer;

 procedure valores (var cant:integer; var aux:integer; num:integer);
   var num1: integer;
   Begin
      num:= 10;
      aux:= num + 15;
      cant:= 0;
      writeln ('num= ', num, ' num1= ', num1, ' aux= ', aux, ' cant= ', cant);
      writeln;
   end;
var
   a: integer;
begin
   num1:= 50;
   writeln ('a= ', a, ' c= ', c, ' n= ', n, ' num1= ', num1);
   writeln;
   valores (c, a, n);
   writeln ('a= ', a, ' c= ', c, ' n= ', n, ' num1= ', num1);
   readln;
end.
```

¿Qué imprime en cada sentencia write?

Ejercitación

Escribir un programa que:

a. Lea un número entero e invoque a un módulo que reciba dicho número y devuelva la cantidad de dígitos pares que contenga. El programa luego debe imprimir el resultado.
b. Invoque a un módulo que lea una secuencia de caracteres y devuelva la cantidad de letras minúsculas leídas, la cantidad de mayúsculas leídas, la cantidad de dígitos y la cantidad del resto de los caracteres.
c. Invoque a un módulo que lea 10 números enteros y retorne el número más grande y el más chico.
d. Lea un número entero e invoque a un módulo que reciba dicho número y devuelva verdadero si contiene exactamente 3 dígitos 5, o falso en caso contrario. El programa luego debe informar si cumple o no.

Modularización

- Al **modularizar** es muy importante **obtener independencia funcional de cada módulo**. Esto disminuye el acoplamiento con el resto del sistema y por lo tanto reduce el impacto de las fallas y modificaciones.

- La **comunicación** entre módulos **debe** acotarse a intercambio de datos por parámetros. Siempre que se pueda deben utilizarse parámetros por valor.

- Las **funciones** pueden pensarse como operadores definidos por el usuario, que **reciben** variables (parámetros por valor) y **producen** un resultado único.

- Los **procedimientos** son subprogramas que interactúan en el espacio de datos del módulo. Puede devolver resultados a través de parámetros por referencia.

- En todos los casos **deben** evitarse las variables globales.