PROGRAMACIÓN I

Temas

- ✔ Repaso
- ✔ Comunicación entre módulos
- ✔ Ejemplos

¿REPASAMOS?

Modularización - Repaso

✔ ¿Qué es la modularización?

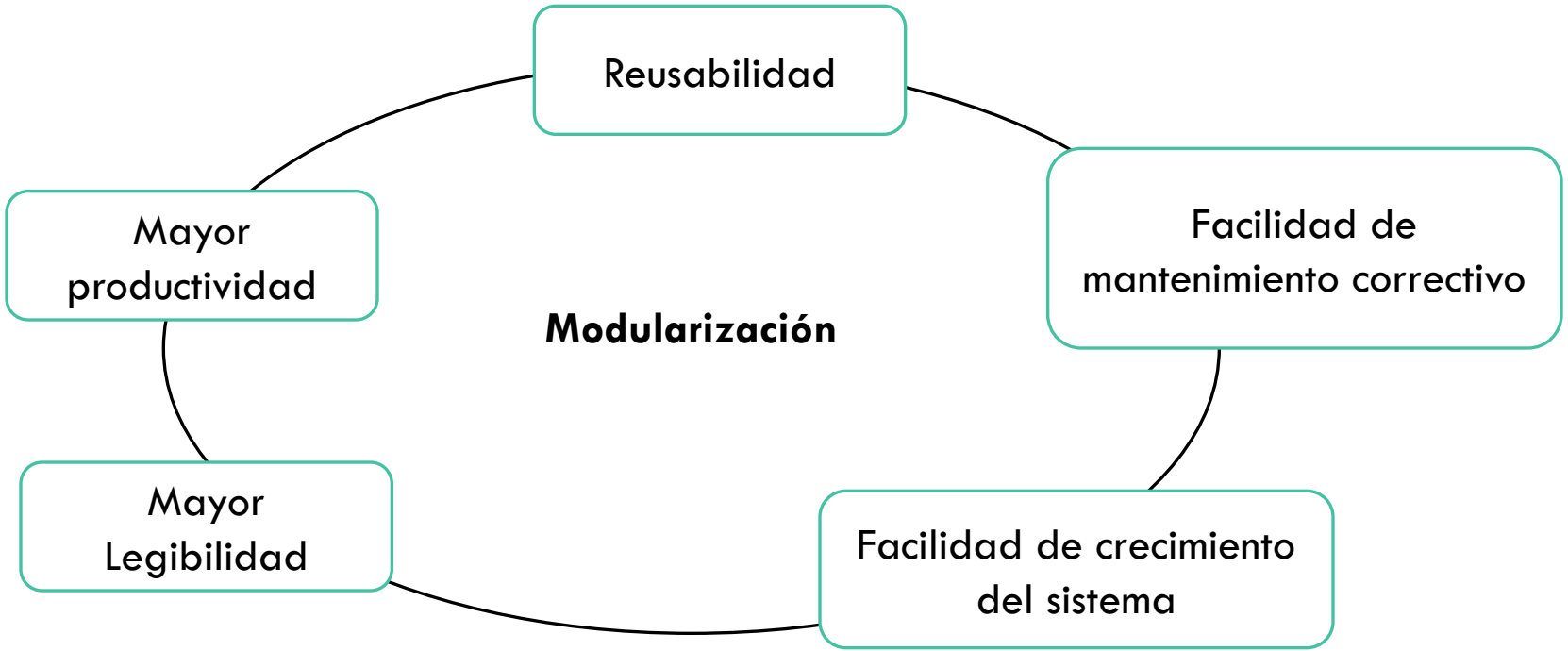
Modularizar es una estrategia que implica dividir un problema en partes funcionalmente independientes, que encapsulen operaciones y datos alta cohesión y bajo acomplamiento

✔ ¿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

✔ Ventajas de la modularización

Modularización - Repaso



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

TIEMPO DE VIDA DE UNA VARIABLE: DESDE EL MOMENTO EN QUE SE RESERVA SU MEMORIA HASTA QUE SE LIBERA

EJERCICIOS DE REPASO SOBRE ALCANCE

Modularización – ALCANCE DE UNA VARIABLE

```
Program ejemplo;
var  cantidad: integer;

Procedure  Asignar;
   var precio: real;
   Begin
        cantidad:= 25;
        precio:= 62.50;
        writeln(cantidad, precio);
   End;

var precio: real;

Begin
    cantidad:= 28;
    precio:= 100.50;
    writeln(cantidad, precio);
    Asignar;
    writeln(cantidad, precio);
End.
```

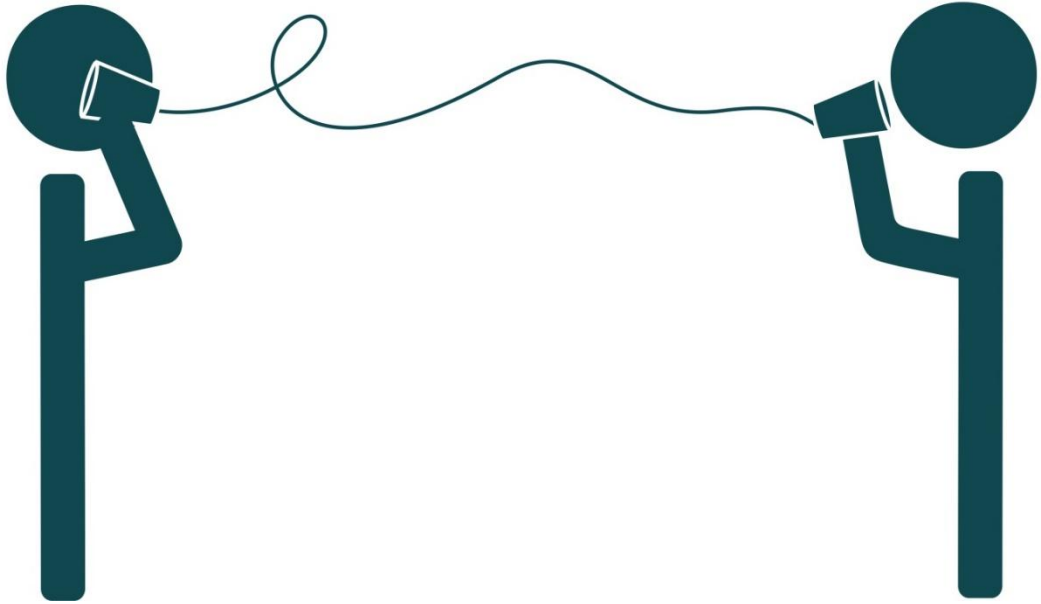
¿Qué imprime?

Modularización – ALCANCE DE UNA VARIABLE

```
Program ejemplo;
var  y: integer;

Function  Duplicar: integer;
    var x: integer;
    Begin
          x:= y * 2;
          Duplicar:= x;
    End;

var z: integer;

Begin
     y:= 30;
     z:= Duplicar();
     writeln(z);
     y:= 5;
     writeln (Duplicar());
End.
```

¿Qué imprime?

COMUNICACIÓN ENTRE MÓDULOS



Modularización - Comunicación

¿CÓMO SE COMUNICAN LOS DATOS ENTRE MÓDULOS Y PROGRAMA?

VARIABLES GLOBALES

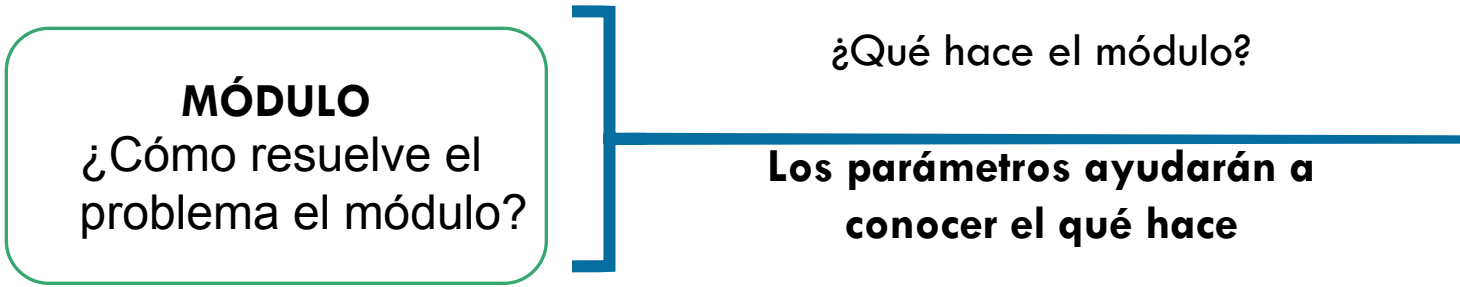
NO ACONSEJABLE. PUEDE TRAER EFECTOS COLATERALES. PERJUDICA LA LEGIBILIDAD

PARAMETROS

INDICAN DE MANERA EXPLÍCITA QUÉ DATOS UTILIZARÁ EL MÓDULO

Modularización - Comunicación

Teóricamente, cada módulo lo podemos pensar como una caja negra con una tarea bien definida (QUÉ) que puede ser implementada internamente de muchos modos (CÓMO).



Modularización - Comunicación

La comunicación externa de un módulo con el resto del sistema es aconsejable que se produzca a través de datos de entrada y datos de salida a través de parámetros.

Modularización - Comunicación

MÓDULO

Tiene un Encabezado

No tienen comunicación mediante parámetros en este caso

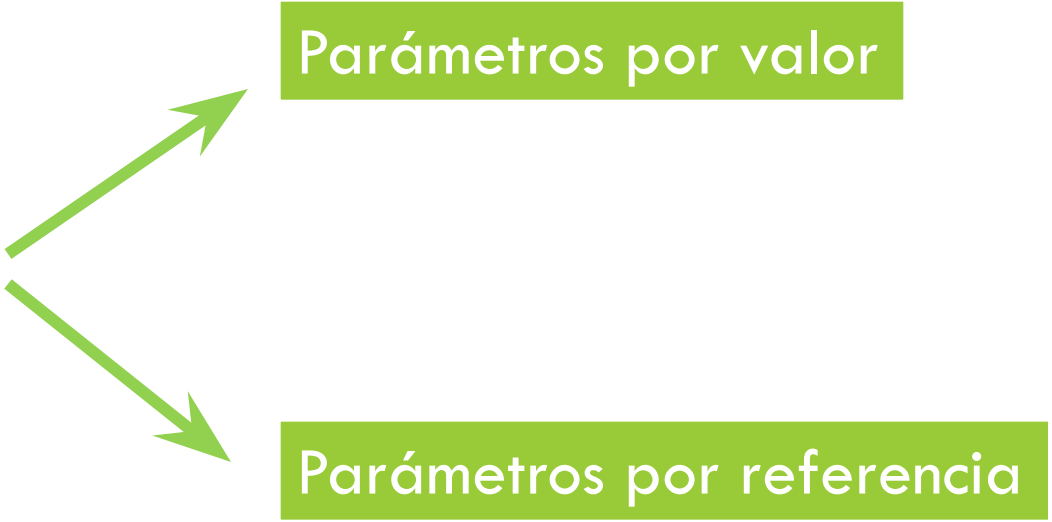
- **PROCEDURE Nombre;**
- **FUNCTION Nombre:Tipo;**

Comunicación Por parámetros

- **PROCEDURE Nombre (dato:char);**
- **FUNCTION Nombre(dato:char):integer;**

Modularización - Comunicación

La comunicación puede ser **en Pascal**



Parámetros por valor

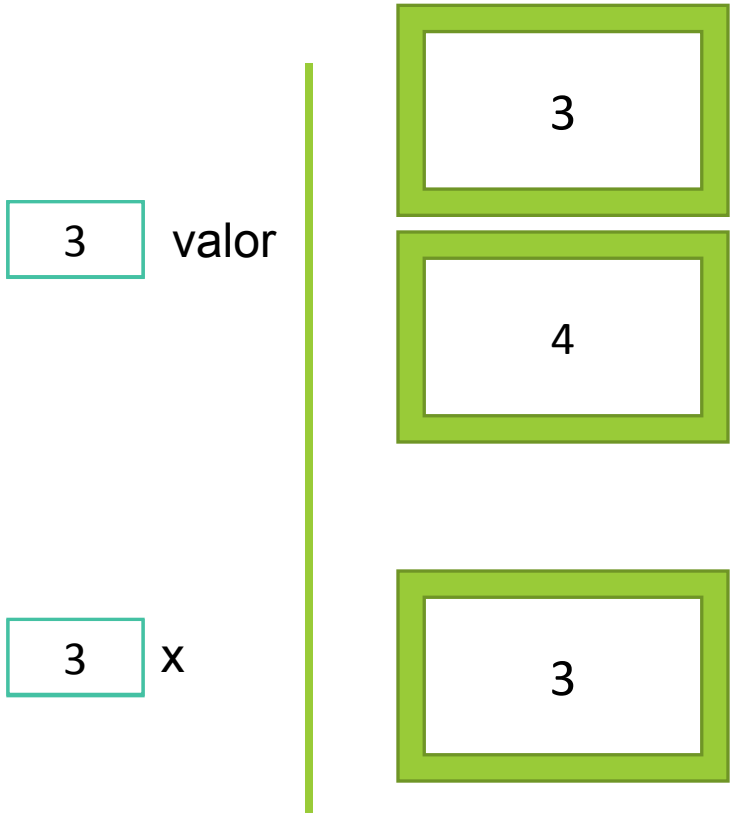
Comunicación – Parámetros por valor

- 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, el módulo puede realizar operaciones y/o cálculos, pero no producirá ningún cambio ni tampoco tendrá incidencia fuera del módulo.

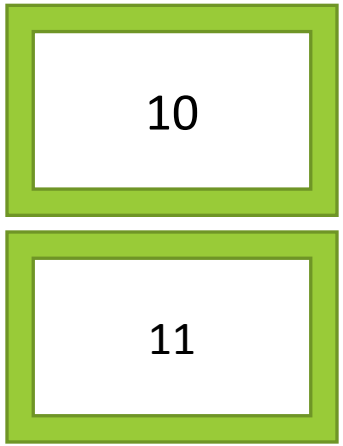
Comunicación – Parámetros por valor

```
Program valor;
  Procedure uno (valor:integer);
     Begin
        write (valor);
        valor:= valor+1;
        write (valor);
     End;
var x: integer;
Begin
  x:=3;
  uno (x);
  write (x);
End.
```

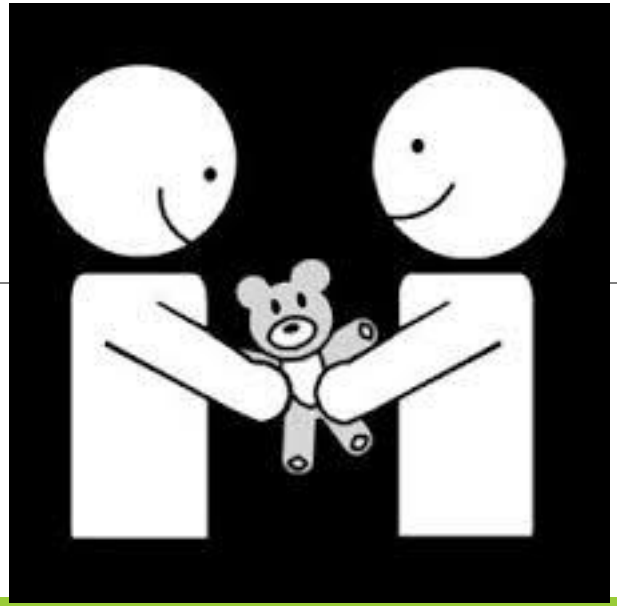


Comunicación – Parámetros por valor

```
Program valor;
  Procedure uno (valor:integer);
     Begin
        write (valor);
        valor:= valor+1;
        write (valor);
     End;

Begin

   uno (10);

End.
```



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.

Comunicación – Parámetros por referencia

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.

Comunicación – Parámetros por referencia

```
Program referencia;
  Procedure uno (VAR ref:integer);
     Begin
        write (ref);
        ref:= ref+1;
        write (ref);
     End;
var x: integer;
Begin
  x:=3;
  uno (x);
  write (x);
End.
```

Parámetros – Conceptos importantes

El número y tipo de los parámetros o argumentos utilizados en la invocación a una Función o un Procedimiento deben coincidir con el número y tipo de parámetros del encabezamiento del módulo.

Parámetros – Conceptos importantes

```
Program uno;
Var
  x:integer;
  c: char;
procedure ejemplo (var a:integer; c:char);
Begin
   c:='p';
  …
end;
```

Parámetros formales

```
begin
  x:= 98;
  c:='t';
  ejemplo (x,c);
end;
```

Parámetros actuales o reales

Parámetros – Conceptos importantes

Un **parámetro por valor** debiera ser tratado como una variable de la cuál el Procedimiento o Función hace una copia y la utiliza localmente.

Algunos lenguajes permiten la modificación local de un parámetro por valor, pero toda modificación realizada queda en el módulo en el cual el parámetro es utilizado.

Parámetros – Conceptos importantes

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

- Esto significa que no requiere memoria local.

Ejemplos

Parámetros – Ejemplos

```
Program uno;
      procedure ejemplo (var a:integer; j:char);
      begin
        …
      end;
var
  x:integer;
  c: char;
begin
   x:=25;
   ejemplo (x,'p');
end.
```

¿Es válida esta invocación?

Sí

Parámetros – Ejemplos

```
Program uno;
    procedure ejemplo (var a:integer; j:char);
    begin
      …
    end;
var
  x:integer;
  c: char;
begin {comienza el programa}
  c:='t';
  ejemplo (15, c);
end.
```

¿Es válida esta invocación?

No, ¿por qué?

Parámetros – Ejemplos

```
Program tres;
   Procedure saber  (b:  integer; Var a:integer);
      begin
            b:= b + 10 ;
            a: = a + 10 ;
           write (a, b);
      end;
var a, b: integer;
begin {comienza el programa}
   a := 5;
   b: = 3;
   saber ( a, b );
   write ( a, b);
end.
```

¿Qué imprime?

Parámetros – Ejemplos

```
Program  cuatro;
    Function incremento (x: integer):  integer ;
        begin
          x  :=  x  +  1;
          incremento := x;
       end;
var
   a, b: Integer ;
begin  {comienza el programa}
      a := 10;
      b :=  incremento (a);
      writeln ( a, ´+1=´, b );
end.
```

¿Qué imprime?

Parámetros – Ejemplos

```
Program  cinco;
   procedure sumar  ( var  a:  integer; b:integer);
          begin
               a  :=  a  +  b;
           end;

var  a, b: Integer;
begin  {comienza el programa}
       a := 15; b:=5;
       sumar (a, b);
       writeln (a);
       writeln (b);
end.
```

¿Qué imprime?

Parámetros – Ejemplos

```
Program  seis;
   procedure sum_mul  ( var  x:  integer; b:integer);
          begin
               b:= b + 1;
               x  :=  b * 2;
           end;

var  a, can: Integer;
begin  {comienza el programa}
       can:=5;
       sum_mul (a, can);
       writeln (a);
       writeln (b);
end.
```

¿Qué imprime?

Para resolver en grupos

Parámetros – Ejemplos

```
Program  g1;
    procedure grupo1 (var  a, b :integer; var c:integer );
        begin
           b:= b + 1;
           a  :=  b * 2;
           c:= a+ b + c;
           write (a, b, c);
       end;
var  a,b,c : Integer ;
begin  {comienza el programa}
      a:= 5; b:=3; c:=4;
     grupo1 (b, a, c);
     write (a, b, c);
end.
```

Parámetros – Ejemplos

```
Program  g2;
   Procedure  grupo2 (var a:integer; var b:integer; c:integer);
   begin
        b := 11;
        a:= b MOD  5;
        c:= (c + a) DIV 3;
        write (a, b, c)
   end;
var  a, b, c : integer;
begin       {del programa principal}
        a := 10;  b:= 3; c:=5;
        grupo2 (c, a, b);
        write (a, b, c);
 end.
```

Parámetros – Ejemplos

```
Program g3;
    procedure grupo3  ( var  a: integer; b :integer; c:integer );
    begin
        b:= b + 1;
        a:=  b * 2;
        c:= a+ b + c;
        write (a, b, c);
    end;
var  a, b, c : Integer ;
begin  {comienza el programa}
     a:= 7;  b:=3;  c:=4;
    grupo3 (b, c, a);
    write (a, b, c);
end.
```

Parámetros – Ejemplos

```
Program  g4;
    procedure grupo4  (var  a: integer; var b, c: integer);
     begin
            b:= c + 1;
            a  :=  b * 2;
            c:= a+ b + c;
            write (a, b, c);
     end;
var  a,b,c : Integer ;
begin  {comienza el programa}
      a:= 2; b:=3;  c:=4;
     grupo4 (c, a, b);
     write (a, b, c);
end.
```

Parámetros – Ejemplos

```
Program  g5;
   procedure grupo5  (a :integer; var b, c: integer);
      begin
            b:= c + 1;
            a  :=  b * 2;
            c:= a+ b + c;
            write (a, b, c);
      end;
Var  a,b,c : Integer ;
begin  {comienza el programa}
       a:= 1; b:=3; c:=4;
      grupo5( c, a, b); writeln (a, b, c);
      grupo5 (a, b, c); writeln (a, b, c);
end.
```

Parámetros – Ejemplos

```
Program  g6;
    Procedure  grupo6 (var a:integer; var b;integer; c:integer);
    begin
        b := 11;
        a:= b MOD  5;
        c:= (c + a) DIV 3;
        write (a, b, c)
    end;
Var  a, b, c : Integer;
begin       {del programa principal}
        a := 10;  b:= 6; c:=5;
        grupo6 (c, a, b); write (a, b, c);
        grupo6 (b, a, c); write (a, b, c);
 end.
```

Modularización - Parámetros

- 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.

Para pensar

Si los módulos no utilizaran parámetros para comunicación, vistos desde afuera, no sabríamos qué datos está manipulando, debiéramos ver el módulo, línea a línea para ver los datos globales que está usando.

Mayor legibilidad

Para pensar

Problemas de usar variables globales

- No se especifica la comunicación deseada entre los módulos.
- Conflictos de nombres de identificadores utilizados por diferentes programadores.
- Posibilidad de perder integridad de los datos, al modificar involuntariamente en un módulo datos de alguna variable que luego deberá utilizar otro módulo.