

1 Comunicación entre módulos

Variables globales

Parámetros

Ejercitación

Etapas de la
resolución
de un
problema

Problema del
Mundo Real

Análisis

Diseño

*Solución
Modularizada*

Módulos

Datos propios

Datos compartidos

Implementación

Verificación

Alcance de los datos

```
Program dos;  
Var  
  a, b: integer;
```

Variables
globales

```
procedure calculo;  
  var x: integer;  
  Begin  
    x:= 9; a:= 100;  
    write (x);  
  End;
```

Variable
Local del módulo

```
procedure mostrar;  
  var y: char;  
  Begin  
    y:= 'P';  
    a:= a+10;  
    write (a);  
    write (b);  
  End;
```

Variable
Local del módulo

```
Var  
  h: char;  
Begin  
  a:= 80;  
  b:= a * 2;  
  h:= 'A';  
  calculo;  
  mostrar;  
  write (a);  
End.
```

Variable Local del
programa

Datos propios



Variables
locales
(módulo y
programa)

Datos
compartidos



¿Variables
globales?

**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

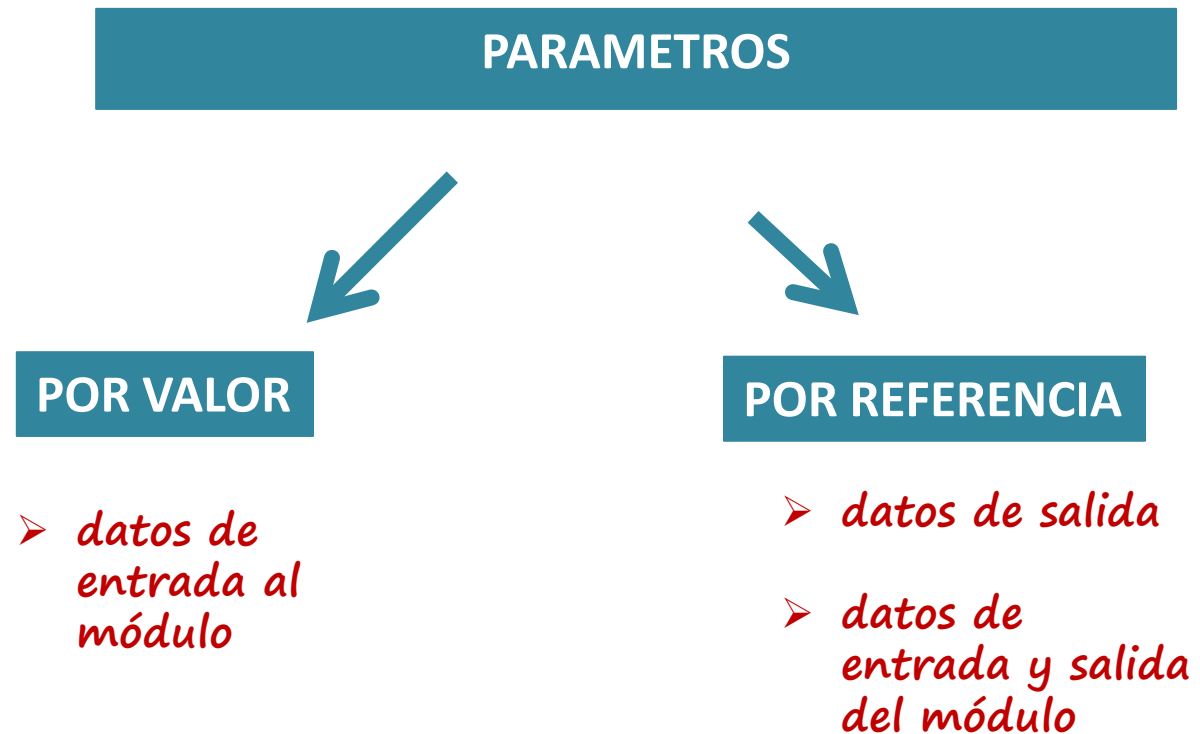
Se recomienda evitar el uso de variables globales

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 transmite entre los módulos.

Del curso de Ingreso se sabe que el dato compartido puede ser un dato de entrada al módulo, un dato de salida del módulo o bien un dato de entrada y salida del módulo .

¿Cómo se implementan los parámetros en Pascal?



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?

Un parámetro por referencia es un dato de salida o entrada/salida que contiene la dirección de memoria donde se encuentra la información compartida con otro módulo o programa que lo invoca.

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.

Parámetro por referencia

```
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?


Variables globales vs Parámetros


Variables globales


- 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


Parámetros

 Los módulos no pueden afectar involuntariamente los datos de otros módulos. ¡Independencia! ¡Reusabilidad!

 Se reduce significativamente la cantidad de variables globales

 Permite distinguir el tipo de comunicación

 Los parámetros por referencia no utilizan memoria local

 El uso de parámetros por valor puede significar una importante utilización de memoria local

COMUNICACIÓN DE DATOS ENTRE MODULOS Y PROGRAMA

VARIABLES
GLOBALES

PARAMETROS

*No recomendables
para la
comunicación!!*

Por valor

Por referencia

```

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.

```

Al llamar a un módulo se deben tener cuenta:

Relación al módulo (parámetros actuales)

Relación al módulo (parámetros formales)

Parámetros actuales y formales



Se relacionan 1 a 1

Deben coincidir en cantidad y tipo de dato

- Restricciones propias del lenguaje de implementación

Ejercitación

Program Ejemplo4;

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

*Analicemos ¿cómo se
modifican las variables?
¿Qué se imprime?*

var x, y, z: integer;

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

*Observar que la variable local x no está
inicializada*

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, b);
```

```
    end;
```

```
  var c: integer;
```

```
  begin
```

```
    b := a * 2;
```

```
    calculo(a,b);
```

```
    c:= 0;
```

```
    writeln(a, b, c);
```

```
  end;
```

```
Var a, b,c: integer;
```

```
begin
```

```
  a:= 5;
```

```
  b:= 20 div 10;
```

```
  writeln(a, b, c);
```

```
  cuenta(c, a);
```

```
  writeln(a, b, c);
```

```
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, b, c);  
        a := a + 5;  
        c := a + b;  
        b := b * 5;  
        write (a, b, c);  
    end;
```

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

*¿Qué imprime
en cada
sentencia
write?*

¿Qué imprime en cada sentencia write para la siguiente secuencia?
10 15 3 22 -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);

c:=0;

a:=0;

read (num);

while (num <> 24) do begin

calcular (num,c, a);

read (num);

end;

writeln ('a: ',a, 'c: ', c);

end.

Ejercitación

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.

Recordemos.... ¿Qué tipos de módulos ofrece Pascal?

PROCEDIMIENTOS
(PROCEDURE)

FUNCIONES
(FUNCTION)

Tienen características comunes, pero ciertas particularidades determinan cual es el mas adecuado para implementar un módulo particular

- ¿El módulo devuelve datos?
 - ¿Cuántos datos devuelve?
 - ¿De qué tipo son los datos que devuelve?
 - ¿Qué tipo de acciones ejecuta el módulo?

¿PROCEDURE?

¿FUNCTION?

¿Cuáles son los aspectos que los diferencian?

- Encabezamiento del módulo
- Invocación
- Lugar donde retorna el flujo de control una vez ejecutado el módulo

FUNCTION

Conjunto de instrucciones que realiza una tarea específica y como resultado retorna un único valor de tipo simple.

¿Cómo se define el módulo?

```
Function nombre (lista de parametros ): tipo;
```

```
Type .....
```

```
Var .....
```

```
begin
```

```
.....
```

```
nombre := ...;
```

```
end;
```

— **Asignación
(obligatoria)**

Encabezamiento

**Declaración de tipos
(opcional)**

**Declaración de
variables (opcional)**

**Sección de
instrucciones**

FUNCTION

Conjunto de instrucciones que realiza una tarea específica y como resultado retorna un único valor de tipo simple.

¿Cómo se invoca el módulo?

```
Program otro;  
  
Function cubo (x:integer): integer;  
Begin  
    cubo:= x*x*x;  
End;
```

```
Var a: integer;
```

```
begin  
    read (a);  
    write (cubo (a));  
End.
```

```
Program otro;
```

```
Function cubo (x:integer): integer;  
Begin  
    cubo:= x*x*x;  
End;
```

```
Var a, c: integer;
```

```
begin  
    read (a);  
    c := cubo (a);  
End.
```

¿Qué ocurre
con el flujo de
control del
programa?

Luego de ejecutado el módulo, el flujo de control retorna a la misma instrucción de invocación del módulo

FUNCTION

Conjunto de instrucciones que realiza una tarea específica y como resultado retorna un único valor de tipo simple.

¿Cómo se invoca el módulo?

```
Program otro;
```

```
Function cubo (x:integer): integer;  
Begin  
    cubo:= x*x*x;  
End;
```

```
Var a: integer;
```

```
begin  
    read (a);  
    If (cubo (a) > 100) then ...;  
End.
```

```
Program otro;
```

```
Function cubo (x:integer): integer;  
Begin  
    cubo:= x*x*x;  
End;
```

```
Var a: integer;
```

```
begin  
    read (a);  
    while (cubo (a) < 50) do  
        ...  
End.
```

¿Qué ocurre
con el flujo de
control del
programa?

Luego de ejecutado el módulo, el flujo de control retorna a la misma instrucción de invocación del módulo

Program Ejemplo8;

*¿Qué
imprime el
programa?*

```
Function EsPar (num:integer): boolean;  
  var resto:integer;  
  begin  
    resto := num mod 2;  
    if resto = 0 then EsPar := true  
      else EsPar:= False;  
  end;
```

```
var x: integer;  
begin  
  read (x);  
  if EsPar(x)=true then writeln ('El número ', x, 'es par')  
    else writeln ('El número ', x, 'no es par')  
end.
```