



Modularización con parámetros

Explicación Práctica 2

Programa con módulos

Estructura

```
Program NombrePrograma;  
Type  
    {declaraciones de tipos de datos}
```

```
{Declaración de módulos: procedimientos y funciones}
```

```
var  
    {variables a usar en el programa principal}  
begin  
    {Acciones del prog. principal}  
end.
```

Procedimientos

```
Program NombrePrograma;  
Type  
    {declaraciones de tipos de datos}  
  
Procedure nombre (lista de parámetros);  
Var  
    {Variables locales al procedimiento}  
Begin  
    {Cuerpo del procedimiento}  
end;  
  
    {variables a usar en el programa principal}  
begin  
    {Acciones del prog. principal}  
    nombre(param actuales)  
end.
```

Parámetros formales:

Para cada parámetro:

- Tipo de pasaje:
por valor o referencia (VAR)
- Nombre del parámetro
- Tipo de dato

Invocación al procedure:

Utilizando el **nombre** del proc. con la lista de **parámetros actuales**

La **asociación** entre parámetros formales y actuales es por **posición**.

Parámetros por valor

Concepto:

- USO: El módulo necesita **recibir** información para realizar su tarea.
- El **llamador** no ve modificaciones en el parámetro actual.

Mecanismo:

- Se realiza una copia del valor del parámetro actual en otra posición de memoria correspondiente al parámetro formal. Esta copia se destruye cuando se termina de ejecutar el módulo.
- **Dentro del módulo se puede modificar el valor del parámetro, pero el cambio no será visto por el llamador.**

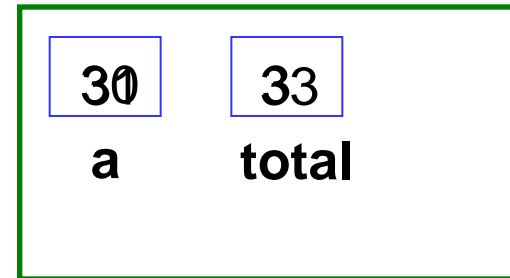
Parámetros por valor ¿Qué imprime?

```
Program paramValor;  
  Procedure uno (a: integer);  
  var  
    total: integer;  
  Begin  
    total:= 3;  
    total:= total + a;  
    a:= a + 1;  
    writeln ('El valor de total es: ',total);  
    writeln ('El valor de a es: ', a);  
  end;  
var  
  x: integer;  
begin  
  x:= 30;  
  uno(x);  
  writeln ('El valor de x es: ', x);  
  readln;  
end.
```

Parámetros por valor ¿Qué imprime?

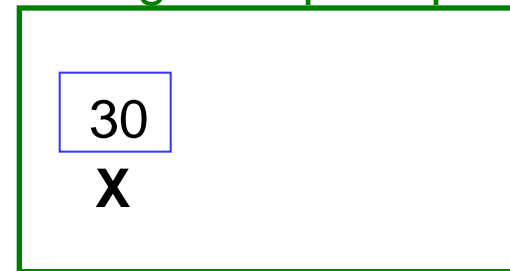
```
Program paramValor;  
→ Procedure uno (a: integer);  
  var  
→   total: integer;  
  Begin  
→   total:= 3;  
→   total:= total + a;  
→   a:= a + 1;  
→   writeln ('El valor de total es: ',total);  
→   writeln ('El valor de a es: ', a);  
→ end;  
→ var  
  x: integer;  
  begin  
→   x:= 30;  
→   uno(x);  
→   writeln ('El valor de x es: ', x);  
→   readln;  
  end.
```

Uno



El valor de total es:33
El valor de a es: 31

Programa principal



El valor de x es:30

Parámetros por referencia

Concepto:

- USO: El módulo necesita **opcionalmente recibir** un dato, **procesarlo** y **devolverlo** modificado al llamador.
- *Cuando se modifica el parámetro formal, la modificación es vista por el módulo llamador.*

Mecanismo:

- El parámetro formal recibe la **dirección de memoria** donde se encuentra la variable que se pasó como parámetro actual.
- El parámetro actual y formal “comparten” el mismo espacio de memoria.

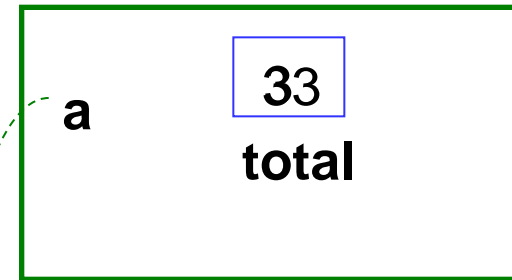
Parámetro por referencia ¿Qué imprime?

```
Program paramReferencia;  
  Procedure uno (var a: integer);  
  var  
    total: integer;  
  Begin  
    total:= 3;  
    total:= total + a;  
    a:= a + 1;  
    writeln ('El valor de total es: ',total);  
    writeln ('El valor de a es: ', a);  
  end;  
var  
  x: integer;  
begin  
  x:= 30;  
  uno(x);  
  writeln ('El valor de x es: ', x);  
  readln;  
end.
```


Parámetro por referencia ¿Qué imprime?

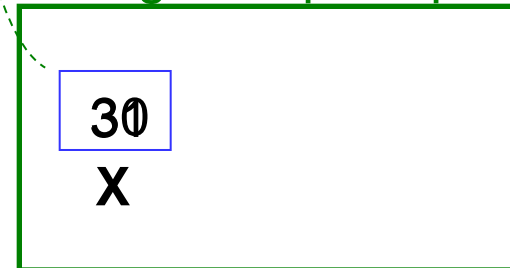
```
Program paramReferencia;  
→ Procedure uno (var a: integer);  
  var  
→   total: integer;  
  Begin  
→   total:= 3;  
→   total:= total + a;  
→   a:= a + 1;  
→   writeln ('El valor de total es: ',total);  
→   writeln ('El valor de a es: ', a);  
→ end;  
  var  
→   x: integer;  
  begin  
→   x:= 30;  
→   uno(x);  
→   writeln ('El valor de x es: ', x);  
→   readln;  
  end.
```

Uno



El valor de total es: 31
El valor de a es: 31

Programa principal



El valor de ³¹x es:

Funciones

Program NombrePrograma;

Type

{declaraciones de tipos de datos}

function nombre (*lista de parámetros*) : tipo;

var

{declaración de variables locales}

begin

{0, 1 o más sentencias}

nombre:= VALOR;

end;

{variables a usar en el programa principal}

begin

{Acciones del prog. principal}

end.

Parámetros formales:
Solo parámetros **por valor**
Para cada parámetro:
• Nombre del parámetro
• Tipo de dato

- Las funciones retornan un dato.

La invocación a una function puede hacerse de distintas formas...

Ejercicio 1

Realizar un programa que lea una secuencia de números enteros positivos hasta que se lee el valor 0 y para cada uno de ellos informe su factorial.

Analizando el problema...

1. ¿Conozco la cantidad de números a procesar?



NO. Se deben procesar nros hasta que el usuario ingrese un 0.

USO UN WHILE

2. ¿Qué necesitamos hacer con cada valor leído?



1. Calcular el factorial
2. Imprimir resultado

3. Para calcular el factorial de un número...
¿Sé cuántas veces debo multiplicar?



Se multiplican los números desde 1 hasta el número en cuestión.

USO UN FOR

4. ¿Qué puedo modularizar?



El cálculo del factorial, ya que es una tarea lógica.

5. ¿Qué uso? ¿Función o procedimiento?



Función, ya que recibe un nro. y retorna un nro. entero.

Program numeros;

Function factorial (n:integer): integer;

Var

I : integer;

fac: integer;

Begin

fac:= 1;

For i:= 2 to n do

fac:=fac * i;

factorial:= fac;

end;

Var

num, c: integer;

Begin

write('Ingrese un numero para calcular su factorial: ');

readln(num);

while (num > 0) do begin

c:= factorial(num);

writeln ('El factorial de ',num, 'es: ', c);

write('Ingrese un numero para calcular su factorial: ');

readln(num);

end;

end.

Solución

Al final de una función se debe asignar un valor de retorno.
- Fac debe ser del mismo tipo que el tipo de retorno de la función (integer)

Program numeros;

Function factorial (n:integer): integer;

Var

i : integer;

fac: integer;

Begin

fac:= 1;

For i:= 2 to n do

fac:=fac * i;

factorial:= fac;

end;

Var

num, c: integer;

Begin

write('Ingrese un numero para calcular su factorial: ');

readln(num);

while (num > 0) do begin

c:= factorial(num);

writeln ('El factorial de ',num, 'es: ', c);

write('Ingrese un numero para calcular su factorial: ');

readln(num);

end;

end.

Solución

¿ Se puede eliminar la variable local **fac** y utilizar para el cálculo directamente **factorial** ?

Funciones

Invocación

- 1) **Asignando el valor a una variable** (*a la derecha de una asignación*)
- 2) **Dentro de un write** (*en caso de que el valor retornado se pueda imprimir*)
- 3) **Dentro de una expresión de una condición.**

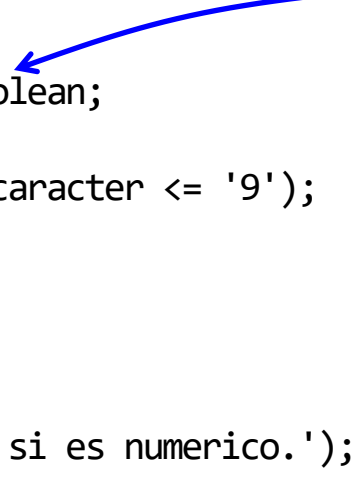
Invocación a una función

Dentro de una expresión en condición

Ejemplo

Hacer un programa que lea un carácter e indique si el mismo es un carácter numérico o no.

```
program ejemplo1;  
  function verificar (character: char) : boolean;  
  begin  
    verificar := (character >= '0') and (character <= '9');  
  end;  
var  
  carac: char;  
begin  
  writeln('Introduce un caracter para ver si es numerico.');
```



A blue arrow originates from the word 'boolean' in the function signature 'function verificar (character: char) : boolean;'. It curves around the right side of the code block and points to the 'verificar(carac)' expression within the 'if' statement condition 'if (verificar (carac))then'. This illustrates that the function returns a boolean value used for conditional logic.

```
    readln(carac);  
    if ( verificar (carac) )then  
      writeln('El carácter introducido es numérico.')
```

The diagram shows a blue arrow starting from the word 'boolean' in the function signature and pointing to the 'verificar(carac)' expression inside the 'if' statement's condition. This indicates that the function returns a boolean value that is used to determine whether to execute the code block following the 'then' keyword.

```
    else  
      writeln('El carácter introducido no es numérico.');
```

The diagram shows a blue arrow starting from the word 'boolean' in the function signature and pointing to the 'verificar(carac)' expression inside the 'if' statement's condition. This indicates that the function returns a boolean value that is used to determine whether to execute the code block following the 'then' keyword.

```
  end.
```

*Retorna un valor
booleano que se
usa en un if*

Invocación a una función

A la derecha de una asignación

Ejemplo

Realice un programa que lea dos números e imprima el cuadrado de ambos.

```
program ejemplo2;  
  
    function cuadrado(a:real):real;  
    begin  
        cuadrado := a*a;  
    end;  
  
var  
    x, y, cuad1, cuad2: real;  
begin  
    writeln('Introduzca dos numeros reales: ');  
    write(' primer numero: '); readln(x);  
    write(' segundo numero: '); readln(y);  
    cuad1:= cuadrado (x);  
    cuad2:= cuadrado (y);  
    writeln(cuad1,cuad2);  
end.
```

la función retorna un valor de tipo real. Dicho valor es asignado a una variable del mismo tipo (cuad1, cuad2)

Invocación a una función

Dentro de un write (si el valor retornado se puede imprimir)

Ejemplo

Realice un programa que lea dos números e imprima el cuadrado de ambos.

```
program ejemplo3;
    function cuadrado(a:real):real;
    begin
        cuadrado := a*a;
    end;
var
    x, y: real;
begin
    writeln('Introduzca dos numeros reales: ');
    write(' primer numero: '); readln(x);
    write(' segundo numero: '); readln(y);
    writeln(cuadrado (x), cuadrado (y));
end.
```

Se muestra directamente el resultado de la función dentro de un write.