# MODULARIZACIÓN

## EXPLICACIÓN PRÁCTICA 3

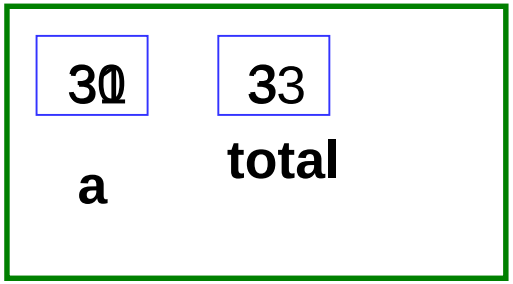
CADP 2018

# MODULARIZACIÓN

## Analizar el siguiente código

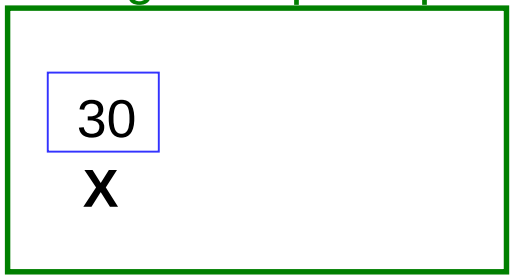
```
Program paramValor;
Procedure uno (a: integer);
  var
    total: integer;
  Begin
    total:= 3;
    total:= total + a;
    a:= a + 1;
    writeln ('El valor de total es: ',total);
    writeln ('El valor de a es: ', a);
  end;
var
  x: integer;
begin
  x:= 30;
  uno(x);
  writeln ('El valor de x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

Uno




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

Programa principal

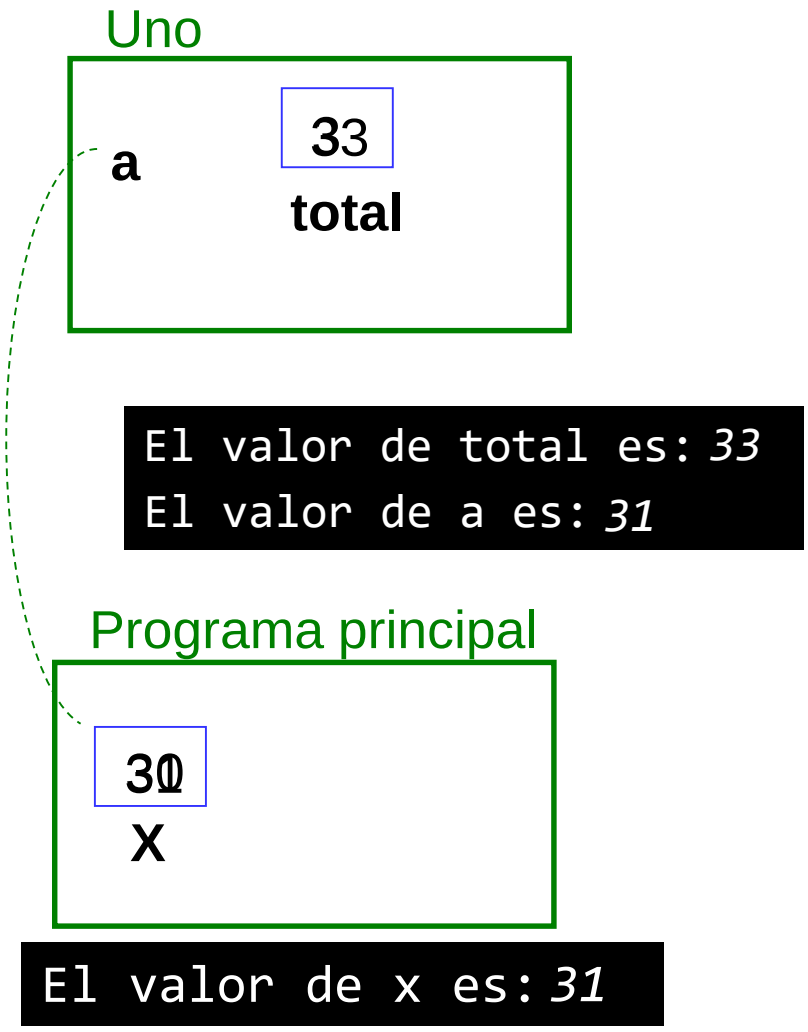



```
El valor de x es:30
```

# MODULARIZACIÓN

## Analizar el siguiente código

```
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.
```



# Ejercicio

**a)** Realice un **procedimiento** que **reciba** como parámetro **un número** entero y **retorne** la cantidad de dígitos impares y la cantidad de dígitos pares que posee el número recibido.

**b)** Utilizando el procedimiento definido en **a)** realice un programa que lea 20 números enteros e informe la cantidad de números que tienen más dígitos pares que impares.

¿Qué datos a comunicar entre el módulo y su llamador?

```
Procedure descomponer (num: integer; var cantP, cantI: integer);
```

# Solución

```
Procedure descomponer (num: integer; var cantP,cantI: integer);
Var
  dig: integer;
Begin
  cantP:= 0;
  cantI:= 0;
  while (num <> 0) do begin
    dig:= num mod 10;
    if ((dig mod 2) = 0) then
      cantP:= cantP + 1
    else
      cantI:= cantI + 1;
    num:= num div 10;
  end;
end;
```

**Variables locales del módulo:**

- Sólo conocidas por el módulo.
- Se crean cuando se invoca al módulo y "*desaparecen*" al llegar al "end" del módulo.

# Solución

```
Program ejercicio;
Procedure descomponer (num: integer; var cantP,cantI: integer);
   {aquí va el cuerpo del procedure descomponer  … }
var
  i, num, pares, impares: integer;
  cant: integer;
begin
  cant:= 0;
  for i:= 1 to 20 do begin
    readln(num);
    descomponer(num, pares, impares);
    if (pares > impares) then
      cant:= cant + 1;
  end;
  writeln('La cantidad de nros que tienen mas dig pares que imp es:', cant);
end.
```

**Variables del programa ppal:**

- Accesibles sólo por el prog ppal.
- Se crean al empezar su ejecución,

desaparecen al llegar al "end" del programa.