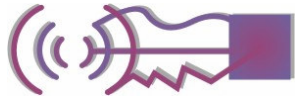


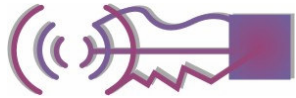
# Parámetros

- Formas de compartir datos entre diferentes unidades:
  - A través del acceso al ambiente no local
  - A través del uso de **parámetros**



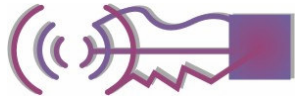
# Parámetros

- A través del acceso al ambiente no local
  - **Ambiente común explícito**
    - Ejemplos:
      - COMMON de FORTRAN
      - Con uso de paquetes de ADA
      - Con variables externas de PL/1
  - **Ambiente no local implícito**
    - Utilizando regla de alcance dinámico
    - Utilizando regla de alcance estático



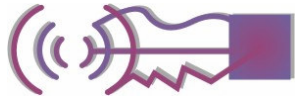
# Parámetros

```
Procedure Main;  
  var x,z,n: integer;  
  Procedure A1()  
    var m: integer;  
    Begin  
      x:= x+m+1;      z:=z+1+n;  
    end;  
  Procedure A2()  
    var x, z: integer;  
    Procedure A3();  
      var z, n: integer;  
      begin  
        n:=3;      z:= x + z; A1();  
      end;  
    begin  
      x:= 1;      z:= x +n; A3();  
    end;  
  begin  
    x:=2; z:=1;      n:=4;      A2();  
  end.
```



# Parámetros

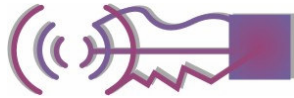
- Pasaje de Parámetros
  - mas flexible, diferentes datos en cada llamada.
  - legibilidad y modificabilidad.
  - qué es exactamente lo que se comparte



# Parámetros

## – Lista de parámetros:

- Conjunto de datos que se van a compartir
  - Parámetros **reales**
    - » Parámetros que se codifican en la invocación del subprograma.
  - Parámetros **formales**
    - » Parámetros declarados en la especificación del subprograma
    - » Contiene los nombres y los tipos de los datos compartidos



# Parámetros

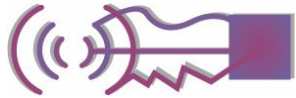
## – Evaluación de los parámetros reales y ligadura con los parámetros formales

- **Evaluación:**

- en el momento de la invocación primero se evalúa los parámetros reales, y luego se hace la ligadura antes de transferir el control a la unidad llamada.

- **Ligadura:**

- **Posicional:** posición que ocupan en la lista
- **Palabra clave o nombre:** Se corresponden con el nombre



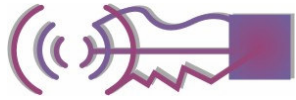
# Parámetros

## – Clases de parámetros: Datos y Subprograma

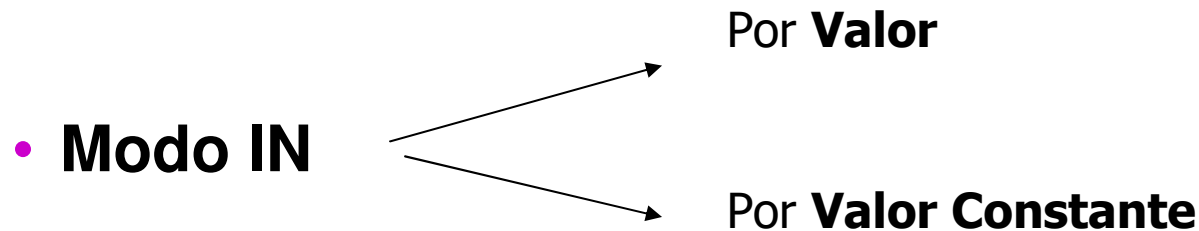
- **Parámetros datos**

Desde el punto de vista semántico los parámetros formales pueden ser:

- **Modo IN:** parámetro formal recibe el dato desde el parámetro real
- **Modo out:** parámetro formal envía el dato al parámetro real
- **Modo IN/OUT:** parámetro formal recibe el dato del parámetro real y el parámetro formal envía el dato al parámetro real



# Parámetros



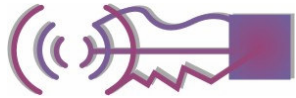
## Por Valor:

- El valor del parámetro real se usa para inicializar el correspondiente parámetro real al invocar la unidad.
- Se transfiere el dato real.
- En este caso el parámetro formal actúa como una variable local de la unidad llamada.

*Desventaja:* consume tiempo y almacenamiento

*Ventaja:* protege los datos de la unidad llamadora





# Parámetros

## Ejemplo:

**Procedure A;**

Var x: integer;

**Procedure B( y:integer);**

**Begin**

y := y + 1;

Write (x,y);

**End;**

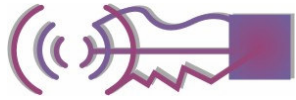
**Begin**

X := 9;

B(x);

Write (x);

**End;**



# Parámetros

## Por valor constante:

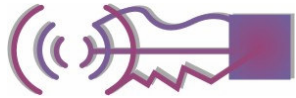
- No indica si se realiza o no la copia, lo que establece es que la implementación **debe verificar** que el **parámetro real no sea modificado**.

*Desventaja:* requiere realizar mas trabajo para implementar los controles.

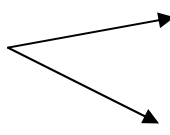
*Ventaja:* protege los datos de la unidad llamadora

## Ejemplo en C/C++

```
void ActualizarMax( const int x, const int y )  
{if ( x > y ) Max= x ;  
    else Max= y ;}
```

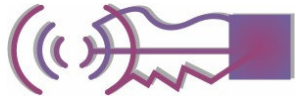


# Parámetros

- **Modo OUT** 
  - Por **Resultado**
  - Resultado de funciones**

## **Por Resultado:**

- El valor del parámetro formal se copia al parámetro real al terminar de ejecutarse la unidad llamada.
- El parámetro formal es una variable local, sin valor inicial.
- *Desventaja:*
  - Consume tiempo y espacio.
  - Se debe tener en cuenta el momento en que se evalúa el parámetro real
- *Ventaja:* protege los datos de la unidad llamadora



# Parámetros

## Por resultado de funciones:

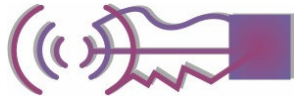
- El resultado de una función puede devolverse con el return o en el nombre de la función

Ej: En C

```
int f1(int m);  
{....  
  return(m)  
}
```

En Pascal

```
Function F1(m:integer):integer;  
  begin  
    F1:=m + 5;  
  end;
```

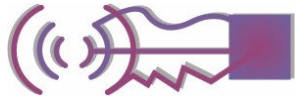


# Parámetros

- **Modo IN/OUT**
  - Por **Valor-Resultado**
  - Por **Referencia**
  - Por **Nombre**

## Por Valor/Resultado:

- Copia a la entrada y a la salida de la activación de la unidad llamadora.
- El parámetro formal es una variable local que se recibe una copia a la entrada del contenido del parámetro real y a la salida el parámetro real recibe una copia de lo que tiene el parámetro formal.
- Cada referencia al parámetro formal es una referencia local.
- Tiene las desventajas y las ventajas de ambos.



# Parámetros

**Ejemplo:**

**Procedure A ();**

var m:integer;

**Procedure B (valor-resultado j:integer);**

**begin**

j:=j+5;      j:= j+ m;

write (j,m);

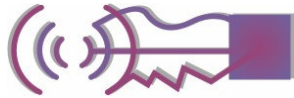
**end;**

**begin**

m:=4; B(m);

write (m);

**end;**



# Parámetros

## **Por Referencia:**

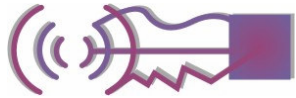
- Se transfiere la dirección del parámetro real al parámetro formal.
- El parámetro formal será una variable local a la unidad llamadora que contiene la dirección en el ambiente no local.
- Cada referencia al parámetro formal será a un ambiente no local. El parámetro real es compartido por la unidad llamada.

### *Desventajas:*

- El acceso al dato es mas lento por la indirección
- Se pueden modificar el parámetro real inadvertidamente
- Se pueden generar alias

### *Ventaja:*

- Eficiente en tiempo y espacio.



# Parámetros

## Ejemplo en Pascal:

**Procedure A ();**

var m:integer;

**Procedure B (var j:integer);**

**begin**

j:=j+5;      j:= j+ m;

write (j,m);

**end;**

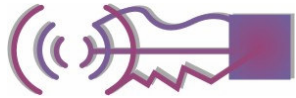
**begin**

m:=4; B(m);

write (m);

**end;**





# Parámetros

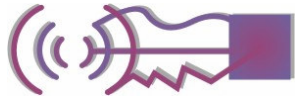
## Por Nombre:

- El parámetro formal es sustituido textualmente por el parámetro real.

**la ligadura de valor se difiere hasta el momento en que se lo utiliza.**

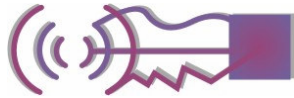
.

- Si el dato a compartir es:
  - » Un único valor se comporta exactamente igual que el pasaje por referencia.
  - » Si es una constante es equivalente a por valor.
  - » Si es un elemento de un arreglo puede cambiar el subíndice entre las distintas referencias
  - » Si es una expresión se evalúa cada vez



# Parámetros

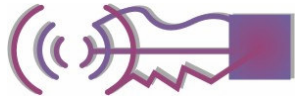
```
Procedure A ();  
  var m:integer;  
  var: arre[1..10] of integer;  
  Procedure B (nombre j:integer);  
    begin  
      j:=j+5;  
      m:=m+1;  
      j:= j+ m;   write (j,m);  
    end;  
  begin  
    m:=4;  
    B(arre(m));  
    write (m);  
  end;
```



# Parámetros

## Por Nombre (continuación):

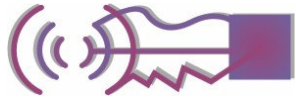
- Para implementarlo se utilizan los thunks. Cada aparición del parámetro formal se reemplaza en el cuerpo de la unidad llamado por un invocación a un thunk que en el momento de la ejecución activara al procedimiento que evaluará el parámetro real en el ambiente apropiado.
- Es un método mas flexible pero debe **evaluarse cada vez que se lo usa**.
- Es difícil de implementar y genera soluciones confusas para el lector y el escritor.



# Parámetros

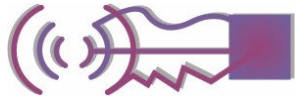
## — Pasaje de parámetros en funciones

- Las funciones no deberían producir efectos laterales.
- Los parámetros formales deberían ser siempre modo IN.
- Ortogonalidad. Los resultados deberían poder ser de cualquier tipo.



# Parámetros

- **Pasaje de parámetros en distintos lenguajes:**
  - **Fortran:**
    - Por valor resultado , por referencia
  - **Algol 60:**
    - Por nombre (por defecto)
    - Por valor (opcional)
  - **Algol 68, C:**
    - Por valor, (si se necesita por referencia se usan punteros).
    - C, permite pasaje por valor constante, agregándole const
  - **Modula II y Pascal:**
    - Por valor (por defecto)
    - Por referencia (opcional: var)
  - **C++:**
    - Igual que C más pasaje por referencia



# Parámetros

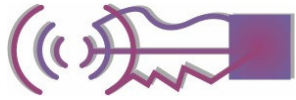
## — Pasaje de parámetros en distintos lenguajes (continuación):

- **Java:**

- El único mecanismo contemplado es el paso por copia de valor.
- Java no contemplan los punteros al estilo de C/C++, por lo tanto no es posible

- **ADA:**

- Por copia **IN** (por defecto)
- Por resultado **OUT**
- **IN OUT.**
  - » Para los tipos **primitivos** indica por **valor-resultado**,
  - » Para los tipos **no primitivos**, datos compuestos (arreglos, registro) se hace por **referencia**



# Parámetros

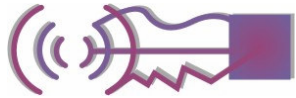
## – Subprogramas como parámetro:

- En algunas situaciones es conveniente poder manejar como parámetros los nombres de los subprogramas.

**Ada** no contempla los subprogramas como valores. Utiliza unidades genéricas.

**Pascal** permite que una referencia a un procedimiento sea pasada a un subprograma

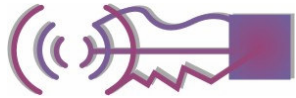
**C** permite pasaje de funciones como parámetros.



# Parámetros

- **Ambiente de referencia para las referencias no locales dentro del cuerpo del subprograma pasado como parámetro.**
  - cuál es el ambiente de referencia no local correcto para un subprograma que se ha invocado a través de un parámetro.
  - opciones:
    - Ligadura **shallow o superficial**: El ambiente de referencia, es el del subprograma que **tiene** el parámetro formal subprograma.
    - Ligadura **deep o profunda**: El ambiente es el del subprograma donde **esta declarado** el subprograma usado como parámetro real.
    - El ambiente del subprograma donde se encuentra el **llamado** a la unidad que tiene un parámetro subprograma.





# Parámetros

## Rutina A

declaracion m

### Rutina B

declaracion m

### Rutina D

.....

End D

### Rutina Z

.....

End Z

C(D)

End B

### Rutina C( rutina X)

declaracion m

x()

End C

...

B()

end A

Ambiente para el caso de  
"Profundo".

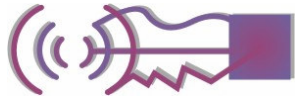
**Procedimiento B**

El ambiente del  
subprograma donde se  
encuentra **el llamado**.

**Procedimiento B**

Ambiente para el caso de  
"Superficial".

**Procedimiento C**



# Parámetros

- Los parámetros subrutinas se comportan muy diferente en lenguajes con reglas de alcance **estático** que **dinámico**.
- La ligadura **shallow** o superficial no es apropiada para lenguajes con estructuras de bloques
- Para lenguajes de alcance estático se utiliza la ligadura **deep**.