Parámetros

Mecanismo de transmisión de datos para una rutina. Permite que esta opere con valores específicos definidos al momento de la invocación

---

- **Parámetro Real →** Es el argumento pasado como parámetro.
- **Parámetro Formal →** Son variables locales a la rutina invocada. Sirven para intercambiar información entre la rutina que hace la llamada y la que la recibe

Vinculación

Pasaje de datos como parámetros

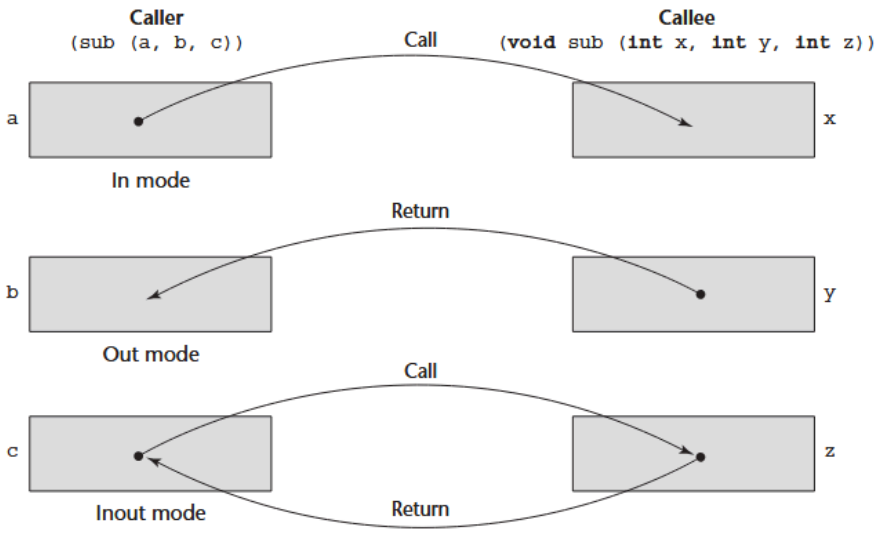
- **Modo IN →** La conexion es al inicio, se copia y se corta la vinculación
  - **Por Valor →** Tipo de comunicación entre un parámetro real y uno formal, donde el parámetro formal actúa como una variable local a la rutina y recibe una copia del valor del parámetro real
  - **Por Valor Constante →** Similar a la comunicación por Valor, con la diferencia de que la rutina no puede modificar el valor recibido, y que no es necesario hacer una copia del parámetro real (depende del lenguaje)
- **Modo OUT →** La conexión se da al final de la ejecución de la rutina
  - **Por Resultado →** Tipo de comunicación donde el valor del parámetro formal se copia al parámetro real tras finalizar la rutina
  - **Por Resultado de Funciones →** El resultado de la función se reemplaza por la invocación en la expresión que contiene el llamado. No utiliza la comunicación entre parámetros, al menos no de la misma forma que los otros
- **Modo IN/OUT →** La conexión es al inicio y al final
  - **Por Valor-Resultado →** Tipo de comunicación donde la rutina recibe un valor y devuelve un resultado, “por medio de la misma variable”. El parámetro real le da valor al parámetro formal, y una vez que se termina

de ejecutar la rutina, el parámetro formal le devuelve un valor al parámetro real (por copia)

- **Por Referencia →** Tipo de comunicación donde el parámetro formal recibe la dirección del parámetro real (el L-Valor de la variable en la unidad llamante). La conexión permanece desde el inicio hasta el final
- **Por Nombre →** Tipo de comunicación donde se reemplaza cada vez que aparece un parámetro formal por el parámetro real, por lo que siempre se tiene que reevaluar.
  - Como se reemplaza la expresión tal cual es, si se llegara a pasar como argumento el elemento de un arreglo o una lista, o cualquier expresión que hiciera referencia a otra variable, y esa otra variable llegara a cambiar de valor en el transcurso de la ejecución de la rutina invocada, entonces el argumento cambiaría de índice en el momento.
  - Cuando se implementa, por lo general también se incluye un mecanismo llamado Thunk, que básicamente le provee un entorno al argumento para evaluarse.

Figure 9.1

The three semantics models of parameter passing when physical moves are used



Tres formas de pasaje de parámetros donde se copian los valores nada mas

Ejemplos de comunicación entre parámetros

- *Por valor y por referencia*

```
program Hello;
{Esta rutina utiliza tanto el Modo IN/OUT por Referencia (variable x)}
{como el Modo IN por Valor (variable local c)}
procedure test(var x: integer; c: integer);
   begin
      x:= x + 5;
      c:= c + 5;
      writeln(x);
      writeln(c);
   end;
var
   a, b, c: integer;
begin
  a:= 1;
  b:= 10;
  c:= 30;
  test(a, b);
  writeln(a);
  writeln(b);
  {la variable global c nunca fue modificada}
  writeln(c);
end.
```

- *Por resultado de funciones*

```
program Test;
{Esta función utiliza el pasaje por resultado de funciones}
function potenciaCuadrada(num: integer):integer;
   begin
         potenciaCuadrada:= num * num;
   end;
var
   a: integer;
begin
```

```
    a:= potenciaCuadrada(2);
end.
```

- *Por referencia y por nombre*

```
program Test2;
type
    vector = array [1..7] of integer;
{Si fuera POR REFERENCIA, tras ejecutarse el programa se vería asi y devolvería
procedure proc(num: integer)
    begin
        num:= num + 1;
        b:= b + 3;
        num:= num + 2;
    end;
var
    a: vector;
  b: integer;
begin
    for i:=1 to 7 do begin
        a[i]:= i;
    end;
    b:= 2;
    proc(a[b])
    {Para el final de la ejecución, el valor de las variables quedan así}
    {a[1]:= 1 // a[2]:= 5 // a[3]:= 3 // a[4]:= 4 }
    {a[5]:= 5 // a[6]:= 6 // a[7]:= 7 // b:= 5 }
end.
{}
{Si fuera POR NOMBRE, tras ejecutarse el programa se vería asi y devolvería tan
procedure proc(num: integer)
    begin
        {Se reemplaza num por a[b], que se evalúa con el valor de b}
        {en ese momento y queda a[2]}
        a[2]:= a[2] + 1;
```

```
            b:= b + 3;
            {Como a la variable b se le sumó 3, y por falta de variable local}
            {y por cadena estática se agarró la variable del main, cuando se}
            {evalúe a[b] va a quedar como a[5]}
            a[5]:= a[5] + 2;
      end;
var
      a: vector;
   b: integer;
begin
      for i:=1 to 7 do begin
            a[i]:= i;
      end;
      b:= 2;
      proc(a[b])
      {Para el final de la ejecución, el valor de las variables quedan así}
      {a[1]:= 1 // a[2]:= 3 // a[3]:= 3 // a[4]:= 4 }
      {a[5]:= 7 // a[6]:= 6 // a[7]:= 7 // b:= 5 }
end.
```

Pasaje de subrutinas como parámetros

Al pasar un subprograma como parámetro, surje la complicación de determinar qué entorno utilizar cuando el subprograma recibido tiene que acceder a variables no locales, que se complica aún mas cuando hay entornos anidados o variables con el mismo nombre en los entornos

- ***En general**: si hay una referencia a una variable que no pertenece a él y no es local entonces surge el problema a dónde va a buscarla porque este es un procedimiento que se mandó como parámetro*

Ligadura Deep

Mira la unidad donde está declarado el subprograma que contiene a la función como parámetro formal (subprograma que lo invoca)

- Se asemeja al alcance estático

Ligadura Shallow

Mira la unidad donde está el subprograma parámetro

- Se asemeja al alcance dinámico

Ligadura Ad-hoc

Mira la unidad donde se invoca a la rutina con el subprograma pasado como parámetro