Practica 6 CPLP

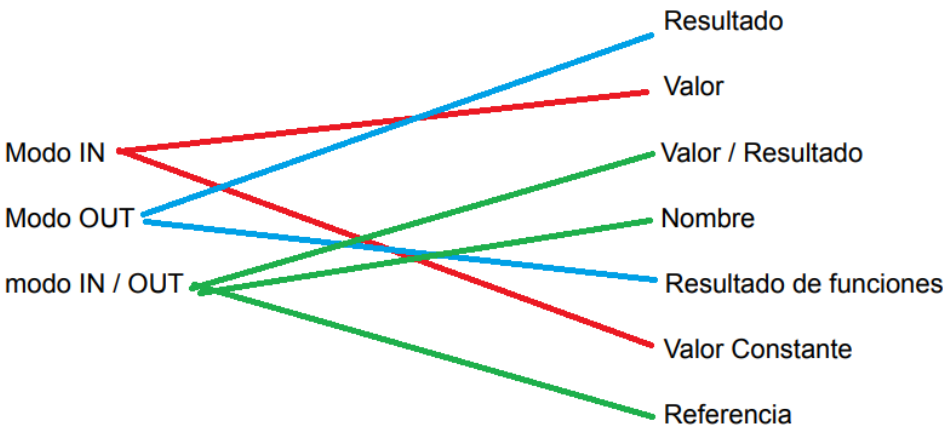
Objetivo: Descubrir las técnicas existentes para pasaje de parámetros entre unidades y sus
diferencias esenciales de acuerdo al lenguaje que lo implementa

Ejercicio 1:
a- Explique brevemente los siguientes conceptos
● Parámetro
● Parámetro real
● Parámetro formal
● Ligadura posicional
● Ligadura por palabra clave o nombre

a_

- Parametro: forma de compartir datos entre unidades.
- Parametro real: un valor u otra entidad que se pasa a un procedimiento o funcion, osea donde se invoca el metodo.
- Parametro formal: es una variable utilizada para recibir valores de entrada en una rutina, osea es la variable que aparece en la definicion de la funcion
- Ligadura posicional: Se corresponden con la posición que ocupan en la lista
- Ligadura por palabra clave o nombre: Se corresponden con el nombre por lo tanto
  pueden estar colocados indistintamente en la lista.

Ejercicio 2: Unir los siguientes puntos según corresponda y de una definición y un ejemplo
de cada par.



Pasaje de Parámetros

● Modo IN

Se usa para pasar datos **hacia** un procedimiento o función.

El valor llega, pero **no se modifica fuera** del procedimiento.

- **Valor:**

  Se pasa una copia del valor de una variable. No afecta a la variable original.

  Ejemplo:

```
pascal
CopiarEditar
procedure Mostrar(x: Integer);
begin
  writeln(x);
end;
```

  Llamada:

```
pascal
CopiarEditar
```

```
Mostrar(5);  // Se pasa una copia del 5
```

- **Valor constante:**

  Similar al valor, pero garantiza que dentro de la función **no se podrá modificar** (solo lectura).

  Ejemplo:

  ```
  pascal
  CopiarEditar
  procedure Mostrar(const x: Integer);
  begin
    writeln(x);  // No se puede cambiar x
  end;
  ```

  Llamada:

  ```
  pascal
  CopiarEditar
  Mostrar(8);
  ```

● Modo OUT

Se usa para **sacar** datos de un procedimiento o función.

El procedimiento escribe en una variable dada como parámetro.

- **Por resultado:**

  El procedimiento escribe directamente en el parámetro.

  Ejemplo:

  ```
  pascal
  CopiarEditar
  procedure Sumar(a, b: Integer; var resultado: Integer);
  begin
    resultado := a + b;
  ```

```
end;
```

Llamada:

```
pascal
CopiarEditar
var r: Integer;
Sumar(3, 4, r);  // r ahora vale 7
```

- **Por resultado de funciones:**

  Se retorna el resultado usando `return` (no se pasa una variable extra).

  Ejemplo:

```
pascal
CopiarEditar
function Sumar(a, b: Integer): Integer;
begin
  Sumar := a + b;
end;
```

  Llamada:

```
pascal
CopiarEditar
var resultado: Integer;
resultado := Sumar(3, 4);  // resultado = 7
```

● Modo IN–OUT

El dato se pasa a la función, puede ser leído y **también modificado** (la modificación se conserva al salir).

- **Valor–resultado:**

El parámetro se pasa **por referencia**, permitiendo **entrada y salida** de datos.

Ejemplo:

```
pascal
CopiarEditar
procedure Incrementar(var x: Integer);
begin
  x := x + 1;
end;
```

Llamada:

```
pascal
CopiarEditar
var num: Integer;
num := 5;
Incrementar(num);  // num ahora vale 6
```

- **Referencia:**

  Se pasa una **referencia** (dirección de memoria) al dato real, no una copia.

  Ejemplo:

```
pascal
CopiarEditar
procedure CambiarValor(var x: Integer);
begin
  x := 10;
end;
```

- **Nombre:**

  Se pasa el **nombre** de la variable, permitiendo que se pueda utilizar directamente como alias.

Ejemplo:

```
pascal
CopiarEditar
procedure Intercambiar(var a, b: Integer);
var temp: Integer;
begin
  temp := a;
  a := b;
  b := temp;
end;
```

Llamada:

```
pascal
CopiarEditar
var x, y: Integer;
x := 5; y := 10;
Intercambiar(x, y);  // x=10, y=5
```

Ejercicio 3:
a- Complete el siguiente cuadro según lo correspondiente a cada lenguaje:

Tipo de pasaje de parametros	Lenguajes
in, out, in-out	ADA
por valor, por referencia (utiliza punteros)	C
referencias a objetos por valor de referencia	Rubi
por valor	JAVA
pasaje por asignacion(pasaje por objeto, se pasa una referencia al objeto, pero la referencia misma es por valor)	Python

b- Ada es más seguro que Pascal, respecto al pasaje de parámetros en las funciones.
Explique por qué.

Es mas seguro porque ADA obliga a indicar si el parametro es:

- Solo de entrada (in), y solo sera de lectura dentro del metodo
- Solo de salida (out)
- De entrada y salida (in-out)

En cambio en pascal:

- Distingue por valor, lo que no se aclara al momento de la declaracion
- Por referencia, indicando la palabra “var” previo a la declaracion

c- Explique cómo maneja Ada los tipos de parámetros in-out de acuerdo al tipo de dato

Los parámetros in out de tipos simples, registros, arreglos, tipos de datos protegidos o tipos limitados suelen ser tratados como pasaje por referencia.

Si el parámetro in out es un puntero (access), el puntero mismo puede ser modificado.
O sea:No solo el contenido apuntado puede cambiar, también podés redirigir el puntero a otro objeto.

Ejercicio 4: Sea el siguiente programa escrito en Pascal-like

```
Procedure Main;
var j, m, i: integer;

Procedure Recibe (x:integer; y:integer);
        begin
                m:= m + 1 + y;
                x:=i + x + j;
                y:=m - 1;
                write (x, y, i, j, m);
        end;
```

```
Procedure Dos;
        var m:integer;
        begin
                m:= 5;
                Recibe(i, j);
                write (i, j, m);
        end;
begin
        m:= 2;
        i:=1; j:=3;
        Dos;
        write (i, j, m);
end.
```

a- Arme el árbol de anidamiento sintáctico y el registro de activación de cada una de las
unidades.

b- Decir qué imprime el programa suponiendo que para todas las variables que

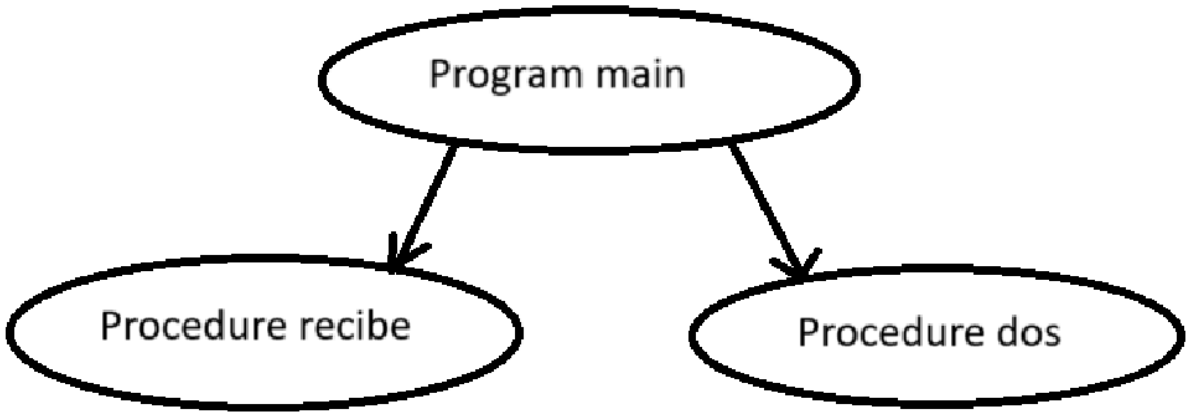
se pasan el
pasaje de parámetros es por: (Deberá hacer la pila estática y dinámica para cada caso)

i- Referencia. ii- Valor iii-Valor Resultado iv- Nombre v-Resultado.

c- ¿Existió algún caso que no pudo realizarlo porque saltó algún tipo de error? Diga cuál y
por qué.

d- ¿Dará el mismo resultado si se trata de un lenguaje que sigue la cadena dinámica?
Justifique la respuesta realizando las pilas de activación

a_



PR
LE
LD
Variables y parámetros
Procedimientos
VR

b_

i- Por referencia y cadena estática

Procedure Main *1	
PR: LE: LD: Variables y parámetros: j: ~~3~~ 5 m: ~~2~~ 6 i: ~~1~~ 5 Procedimientos y funciones: Recibe Dos VR:	write (i, j, m) i → 5 j → 5 M → 6
~~Procedure Dos *2~~	
PR: LE: *1 LD: *1 Variables y parámetros: m: 5 Procedimientos y funciones: VR:	write (i, j, m) i → 5 j → 5 m → 5
~~Procedure Recibe *3~~	
PR: LE: *1 LD: *2 Variables y parámetros: x: ~~1~~ 5 y: ~~3~~ 5 Procedimientos y funciones: VR:	write(x, y, i, j, m): x → 5 y → 5 i → 5 j → 5 m → 6

ii- Por valor y cadena estática

Procedure Main *1	
PR: LE: LD: Variables y parámetros:	write (i, j, m) i → j → M →

j: ~~3~~ 5 m: ~~2~~ 6 i: ~~1~~ 5 Procedimientos y funciones: Recibe Dos VR:	
~~Procedure Dos *2~~	
PR: LE: *1 LD: *1 Variables y parámetros: m: 5 Procedimientos y funciones: VR:	write (i, j, m) i → j → m →
~~Procedure Recibe *3~~	
PR: LE: *1 LD: *2 Variables y parámetros: x: ~~1~~ 5 y: ~~3~~ 5 Procedimientos y funciones: VR:	write(x, y, i, j, m): x → 5 y → 5 i → 5 j → 5 m → 6

iii- Por valor resultado y cadena estática

Procedure Main *1	
PR: LE: LD: Variables y parámetros: j: ~~3~~ 5 m: ~~2~~, m = m + 1 + y = 6	write (i, j, m) i → 5 j → 5 M → 6

i: ~~1~~ 5 Procedimientos y funciones: Recibe Dos VR:	
~~Procedure Dos *2~~	
PR LE *1 LD *1 Variables y parametros m: 5 VR	write (i, j, m) i → 5 j → 5 M → 5
~~Procedure recibe *3~~	
PR LE *1 LD *2 Variables y parametros x: ~~1~~, x= i+x+j = 5 y: ~~3~~, y=m-1= 5 VR	write(x,y,i,j,m) = 5, 5, 1, 3, 6

iv- por nombre y cadena estática

Procedure Main *1	
PR: LE: LD: Variables y parámetros: j: 3 m: ~~2~~, m = m+1+y= 6 i: 1 Procedimientos y funciones: Recibe Dos VR:	write (i, j, m) i → 1 j → 3 M → 6
~~Procedure Dos *2~~	

PR: LE: *1 LD: *1 Variables y parámetros: m: 5 VR	write(i,j,m) = 1, 3, 5
~~Procedure Recibe *3~~	
PR: LE: *1 LD: *2 Variables y parámetros: x: ~~1~~, x=i+x+j= 5 y: ~~3,~~ y=m-1= 5 VR:	write(x,y,i,jm) = 5, 5, 1, 3, 6

v- resultado