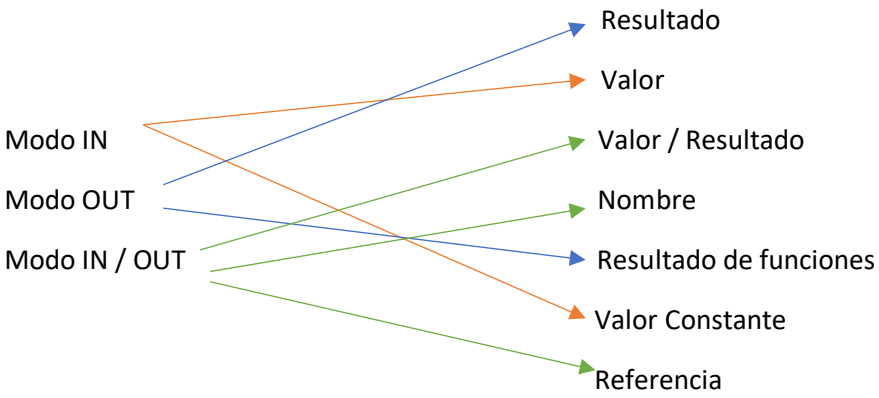
Práctica 6: Parámetros

Ejercicio 1:

a- Explique brevemente los siguientes conceptos

- Parámetro: Es una variable que se define en la declaración de una función o método y que sirve para recibir valores cuando la función es llamada. Representa los datos que la función necesita para trabajar.
- Parámetro real (argumento): Es el valor concreto que se pasa a la función cuando se la invoca. Es lo que se "entrega" al parámetro de la función en una llamada específica.
- Parámetro formal: Es el nombre de la variable que aparece en la definición de la función. Actúa como un marcador en el código de la función, y se vincula a los parámetros reales cuando la función se ejecuta.
- Ligadura posicional: Es un método para asociar parámetros reales con parámetros formales basado en el orden en que se pasan los argumentos. Es decir, el primer argumento se asigna al primer parámetro formal, el segundo al segundo, y así sucesivamente.
- Ligadura por palabra clave o nombre: Es un método en el que los parámetros reales se asignan a los parámetros formales explícitamente por su nombre, independientemente del orden. Por ejemplo, en una llamada como func(nombre="Juan", edad=25), los argumentos se vinculan usando las palabras clave "nombre" y "edad".

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



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

– Modo IN: El parámetro formal recibe el dato desde el parámetro real

- **Valor**: El parámetro formal recibe una copia del valor del parámetro real. Cualquier cambio dentro de la función no afecta al argumento original.
  - **Ejemplo**:

```c
void duplicar(int x) {
    x = x * 2; // Solo modifica la copia local
}
int main() {
    int a = 5;
    duplicar(a);
    printf("%d", a); // Imprime 5, no 10
}
```

- **Valor Constante**: Similar al modo "valor", pero el parámetro formal se trata como una constante dentro de la función, por lo que no puede modificarse.
  - **Ejemplo**:

```c
void mostrar(const int x) {
    // x = 10; // Error: no se puede modificar
    printf("%d", x);
}
int main() {
    int a = 5;
    mostrar(a); // Imprime 5
}
```

– Modo OUT: El parámetro formal envía el dato al parámetro real

- **Por resultado**: El parámetro formal actúa como una variable local dentro de la función, y al finalizar, su valor se copia al parámetro real.
  - **Ejemplo**:

```pascal
procedure calcularCuadrado(x: integer; var resultado: integer);
begin
    resultado := x * x;
end;
var
    a, b: integer;
begin
    a := 4;
    calcularCuadrado(a, b);
    writeln(b); // Imprime 16
end;
```

- **Por resultado de funciones**: La función devuelve un valor a través de su nombre o un mecanismo de retorno, no necesariamente mediante parámetros.
  - **Ejemplo**:

```c
int cuadrado(int x) {
    return x * x; // El resultado se devuelve directamente
}
int main() {
    int a = 4;
    int b = cuadrado(a);
    printf("%d", b); // Imprime 16
}
```

– Modo IN/OUT: El parámetro formal recibe el dato del parámetro real y el parámetro formal le envía el dato al parámetro real

- **Valor – Resultado**: El parámetro formal recibe una copia del valor inicial del argumento, y al finalizar la función, el valor final del parámetro formal se copia de vuelta al argumento.
  - **Ejemplo**:

```ada
procedure incrementar (X : in out Integer) is
begin
    X := X + 1;
end;
declare
    A : Integer := 5;
begin
    incrementar(A);
    Put_Line(Integer'Image(A)); -- Imprime 6
end;
```

- **Referencia**: El parámetro formal es un alias del parámetro real (se pasa la dirección de memoria). Cualquier cambio en el parámetro formal afecta directamente al argumento.
  - **Ejemplo**:

```c
void incrementar(int &x) {
    x = x + 1;
}
int main() {
    int a = 5;
    incrementar(a);
    printf("%d", a); // Imprime 6
}
```

- **Nombre**: El parámetro formal actúa como una referencia simbólica al argumento, evaluándose cada vez que se usa dentro de la función (como una especie de "llamada por necesidad"). Es común en lenguajes como ALGOL.
  - **Ejemplo (simulado)**:

```pseudo
procedure swap(a, b: name integer);
begin
    temp := a;
    a := b;
    b := temp;
end;
x := 1;
y := 2;
swap(x, y);
// x ahora es 2, y ahora es 1
```

En este caso, "a" y "b" no son copias ni referencias directas, sino que se resuelven dinámicamente cada vez que se accede a ellos.

Ejercicio 3:

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

Tipo de pasaje de parámetros	**Lenguaje**
Por valor (IN), por resultado (OUT), por valor-resultado (IN OUT), por referencia (implícito en algunos casos)	ADA
Por valor, por referencia (simulada con punteros)	C
Por referencia a objetos (único modo, comportamiento depende de mutabilidad)	Ruby
Por valor (único modo, con copias de referencias para objetos)	JAVA
Por valor para tipos inmutables (int, float, str, etc.) y pasaje por referencia para tipos mutables (listas, diccionarios, etc.).	Python

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

Ada es más seguro que Pascal en el pasaje de parámetros porque ofrece una semántica explícita y estricta (in, out, in out), verificada en tiempo de compilación, que minimiza errores. Pascal, en cambio, depende más de la responsabilidad del programador y no proporciona las mismas garantías, lo que lo hace más propenso a errores relacionados con el manejo de parámetros.

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

1. Tipos escalares (Integer, Float, Boolean, etc.)

Mecanismo: Por copia (valor-resultado).

Funcionamiento:

- El valor inicial del argumento se copia al parámetro formal al entrar al procedimiento.
- Dentro del procedimiento, se trabaja con esa copia.
- Al salir, el valor final del parámetro se copia de vuelta al argumento original.

2. Tipos compuestos pequeños (Registros o estructuras pequeñas sin restricciones especiales)

Mecanismo: Por copia (valor-resultado), generalmente.

Funcionamiento:

- Similar a los tipos escalares: el registro completo se copia al entrar y se copia de vuelta al salir.
- Se asegura consistencia y evita problemas de aliasing.

3. Tipos compuestos grandes o complejos (Arreglos, Registros grandes, Tipos con restricciones)

Mecanismo: Por referencia (implícito), aunque la semántica sigue siendo como valor-resultado desde la perspectiva del programador.

Funcionamiento:

- En lugar de copiar todo el objeto (lo cual sería ineficiente para estructuras grandes como arreglos), el compilador pasa una referencia al objeto original.
- Las modificaciones se realizan directamente sobre el objeto en memoria.

4. Tipos con restricciones (Constrained Types)

Mecanismo: Por copia o por referencia, dependiendo del tamaño y la implementación.

Funcionamiento:

- Si el tipo tiene restricciones (como un subtipo con un rango específico), Ada garantiza que las modificaciones respeten esas restricciones.
- Para tipos pequeños, se usa copia; para tipos grandes, se usa referencia.

5. Tipos protegidos o tareas (Protected Types, Tasks)

Mecanismo: Por referencia.

Funcionamiento:

- Los tipos protegidos o tareas son complejos y no se copian. Se pasan por referencia para mantener la integridad del objeto y permitir la sincronización.

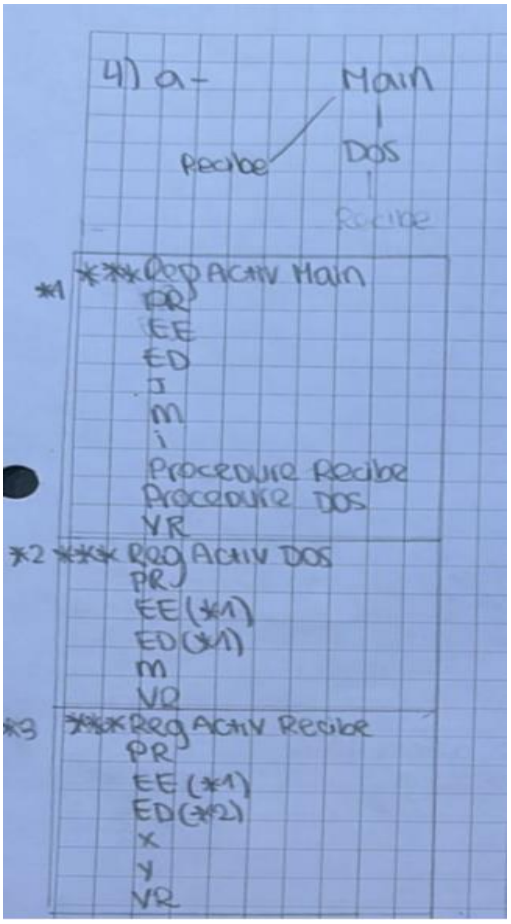
Ejercicio 4: Sea el siguiente programa escrito en Pascal-like

```
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;
```

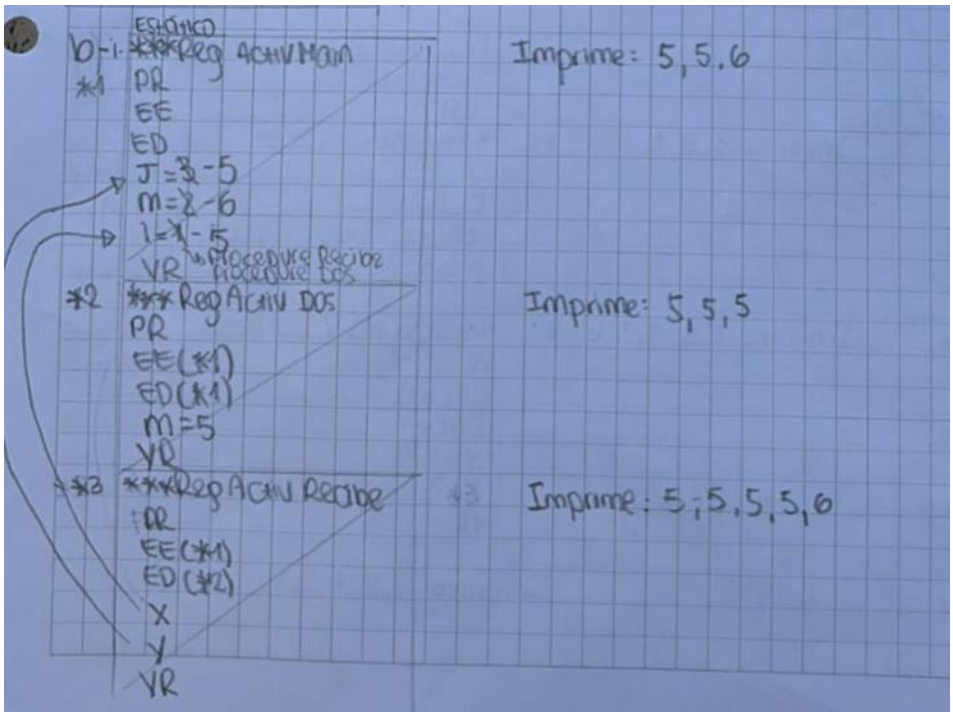
```
Procedure Dos;
        var m:integer;
        begin
                m:= 5;
                Recibe(i, j);
                write (i, j, m);
        end;
begin
        m:= 2;
        i:=1; j:=3;
        Dos;
        write (i, j, m);
end.
```

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.


ESTÁTICO
0-1 Reg ACTIVACIÓN
Imprime: 5, 5, 6
*1
PR
EE
ED
J=3-5
M=2-6
I=1-5
VR
Procedure Recibe Recibe DOS
*2
Reg ACTIV DOS
Imprime: 5, 5, 5
PR
EE(*1)
ED(*1)
M=5
VR
*3
Reg ACTIV Recibe
Imprime: 5; 5, 5, 5, 6
PR
EE(*1)
ED(*2)
X
Y
VR

Dinámica

*1 *** Reg Activ Main
PR
EE
ED
J = 3 - 8
m = 2
i = 1 - 5
Procedure Recibe
Procedure DOS
VR

Imprime 5, 8, 2

*2 *** Reg Activ DOS
PR
EE (*1)
ED (*1)
m = 5 - 9
VR

Imprime: 5, 8, 9

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*2)
X
Y
VR

Imprime 5, 8, 5, 8, 9

ii- Estática

*1 *** Reg Activ Main
PR
EE
ED
J = 3
m = 2 - 6
i = 1
Procedure Recibe
Procedure DOS
VR

Imprime: 1, 3, 6

*2 *** Reg Activ DOS
PR
EE (*1)
ED (*1)
m = 5
VR

Imprime: 1, 3, 5

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*2)
x = 1 - 5
y = 3 - 5
VR

Imprime 5, 5, 1, 3, 6

Dinámica

*1 *** Reg Activ Main
PR
EE
ED
J = 3
m = 2
i = 1
Procedure Recibe
Procedure DOS
VR

Imprime: 1, 3, 2

*2 *** Reg Activ DOS
PR
EE (*1)
ED (*1)
m = 1 - 9
VR

Imprime: 1, 3, 9

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*1)
x = 1 - 5
y = 3 - 8
VR

Imprime 5, 8, 1, 3, 9

iii- Estático

*1 *** Reg Activ Main
PR
EE
ED
J = 3 - 5
m = 2 - 6
i = 1 - 5
Procedure Recibe
Procedure Dos
VR

Imprime: 5, 5, 6

*2 *** Reg Activ Dos
PR
EE (*1)
ED (*1)
m = 5
VR

Imprime: 5, 5, 5

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*2)
X = 1 - 5
Y = 3 - 5
VR

Imprime: 5, 5, 1, 6, 3

Dinámica

*1 *** Reg Activ Main
PR
EE
ED
J = 3 - 8
m = 2
i = 1 - 5
Procedure Recibe
Procedure Dos
VR

Imprime: 5, 8, 2

*2 *** Reg. Activ Dos
PR
EE (*1)
ED (*1)
m = 5 - 9
VR

Imprime: 5, 8, 9

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*2)
X = 1 - 5
Y = 3 - 8
VR

Imprime: 5, 8, 1, 3, 9

iv- Estático

*1 *** Reg Activ Main
PR
EE
ED
J = 3 - 5
m = 1 - 6
i = 1 - 5
Procedure Recibe
Procedure Dos
VR

Imprime: 5, 5, 6

*2 *** Reg Activ Dos
PR
EE (*1)
ED (*1)
m = 5
VR

Imprime: 5, 5, 5

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*2)
X ↑ (i) * 1
Y ↑ (J) * 1
VR

Imprime: 5, 5, 5, 5, 6

Dinámica

*1 *** Reg Activ Main
PR
EE
ED
J = 3 - 8
m = 2
i = 1 - 5
Procedure Recibe
Procedure Dos
VR

Imprime: 5, 8, 2

*2 *** Reg Activ Dos
PR
EE (*1)
ED (*1)
m = 5 - 9
VR

Imprime: 5, 8, 9

*3 *** Reg Activ Recibe
PR
EE (*1)
ED (*2)
X ↑ (i) * 1
Y ↑ (J) * 1
VR

Imprime: 5, 8, 5, 8, 9

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

En el caso v-Resultado ocurre el mismo error tanto por cadena estática que por cadena dinámica. La sentencia m:= m+1+y es la que genera error, ya que "y" es pasado como parámetro únicamente por resultado, y dicho parámetro no está inicializado.

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

No dan el mismo resultado, en el inciso b, se detalla las pilas de activación de cada una.

Ejercicio 5: Suponiendo que se está ejecutando un programa con el siguiente registro de activación en memoria y se llama al procedimiento rutina(iter,vec,a). Determine el tipo de parámetro que se deben utilizar en el llamado para que los resultados sean los siguientes:

a) (4,6,7), (4,6,7), 2, 2

Modo IN/OUT por Referencia en los 3 parámetros.

b) (3,5,6), (4,6,7), 2, 2

Modo IN por Valor para el parámetro vector, y Modo IN/OUT por Referencia para los parámetros restantes (iteración y vit).

c) (3,5,6), (5,5,6), 0, -1

Modo IN por Valor para todos los parámetros. Se puede considerar también Modo IN/OUT por Valor/Resultado en los mismos.

PR
LD
LE
Iter: true
Vec:[3,5,6]
a: -1
Rutina()
VR

```
.....
procedura rutina(tipoParam iteracion,tipoParam vector,tipoParam vit):

        while iteracion begin
                vit = a+1
                vector[vit] = vector[vit]+1
                iteracion = (vector[vit] mod 2)==0
        end
        print vec
        print vector
        print vit
        print a
.....

rutina(iter,vec,a)
```

Ejercicio 6: Indique con un ejemplo el comportamiento del parámetro por nombre (en el parámetro formal) para los siguientes casos de parámetros reales:

- Un valor entero.
- Una constante.
- Un elemento de un arreglo.
- Una expresión.

¿Qué sucede en cada caso?

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 suscripto entre las distintas referencias.
- Si es una expresión se evalúa cada vez.

Ejercicio 7: Realice la pila de ejecución del siguiente programa: **a)** siguiendo la cadena estática **b)** siguiendo la cadena dinámica

```
Procedure Uno;
y, z: integer;
r1:array[1..6] of integer;
r2:array[1..5] of integer;
        Procedure Dos( nombre x, t:integer;
var
                io:integer; valor-resultado
        y:integer);
                Procedure Dos( nombre
        t1:integer);
                        Procedure Tres;
                        begin
                        y:= y + 1;
                        z:= z + 1;
                        end;
                begin
                t1:= t1 + 1;
                t:= t + 1;
                Tres;
                t1:= t1 + 2;
                t:= t + 2;
                end;
        begin
        x:= x + 1;
        t:= t + 1;
        io:= io + 1;
        x:= x + 2;
        if z =2 then Dos ( t );
        end;
begin
for y:= 1 to 6 do r1(y):= 2;
for y:= 1 to 5 do r2(y):= 1;
z:= 2;
y:= 1;
Dos( r1( y + r2( y )), r2( z ), y, z);
for y:= 1 to 6 do write (r1(y));
for y:= 1 to 5 do write (r2(y));
end.
```

7) a) *1

```
***Reg Activ uno
PR
EE
ED
y = 1.6 - 1.5 - x - 2 - 1.6 - 1.5
z = x - 3
r1(1) = 2
r1(2) = 3 - 3
r1(3) = 2
r1(4) = 8 - 4
r1(5) = 2
r1(6) = 2
r2(1) = 1
r2(2) = x - 2 - 3 - 4
r2(3) = x - 3 - 5
r2(4) = 1
r2(5) = 1
Procedure DOS
VR
```

*2

```
***Reg Activ Dos
PR
EE(*1)
ED(*1)
x ↑ ((r1[y + r2[y]]) *1
t ↑ (r2[z]) *1
i0
y = x - 3
Procedure DOS
VR
```

*3

```
***Reg Activ Dos
PR
EE(*2)
ED(*2)
t1 ↑ (t) *2
Procedurtres
VR
```

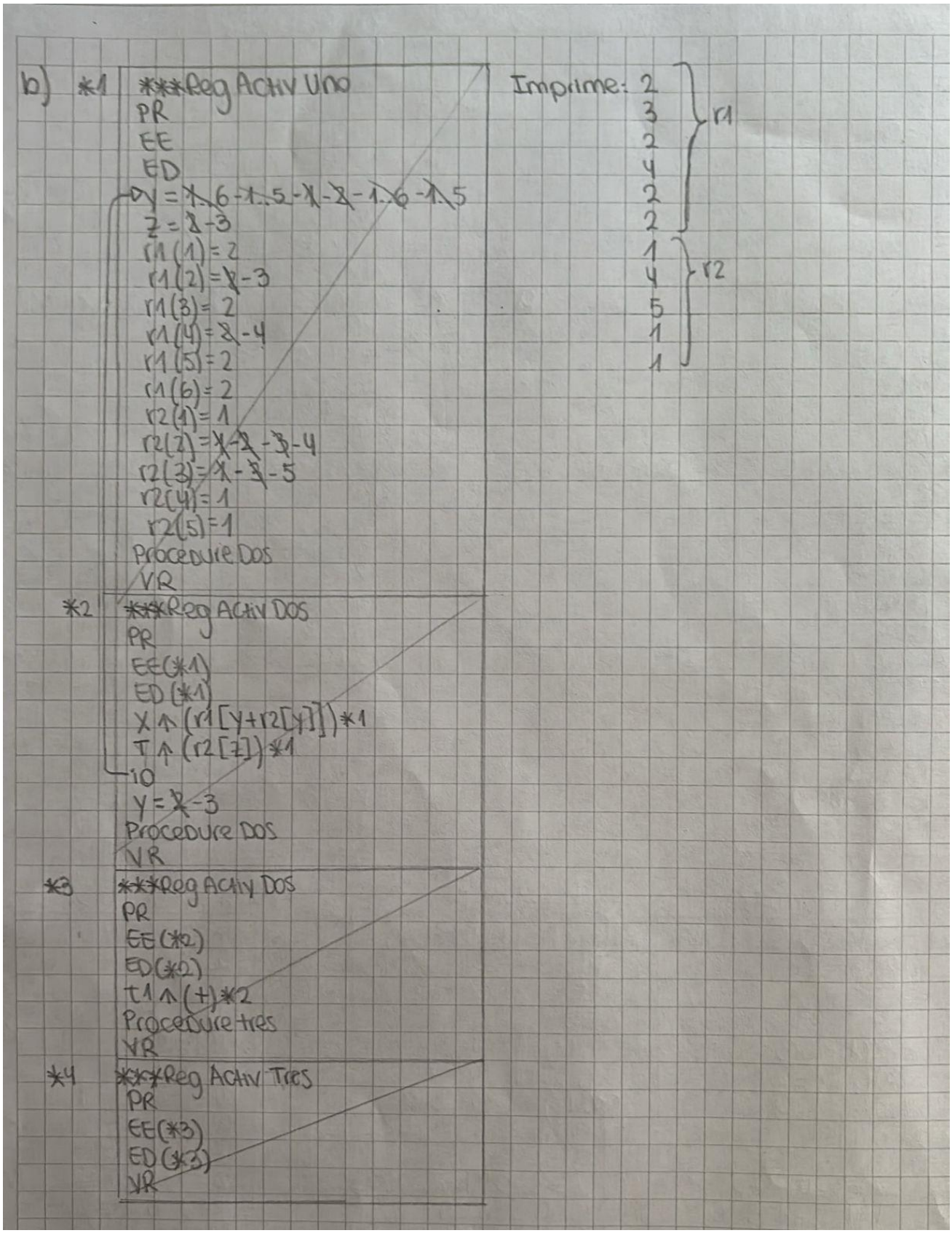
*4

```
***Reg Activ Tres
PR
EE(*3)
ED(*3)
VR
```

Imprime:

2
3
2
4
2
2 } r1

1
4
5
1
1 } r2



Ejercicio 8:

a) Indique las diferencias entre los pasajes de subprogramas como parámetros deep y shallow.

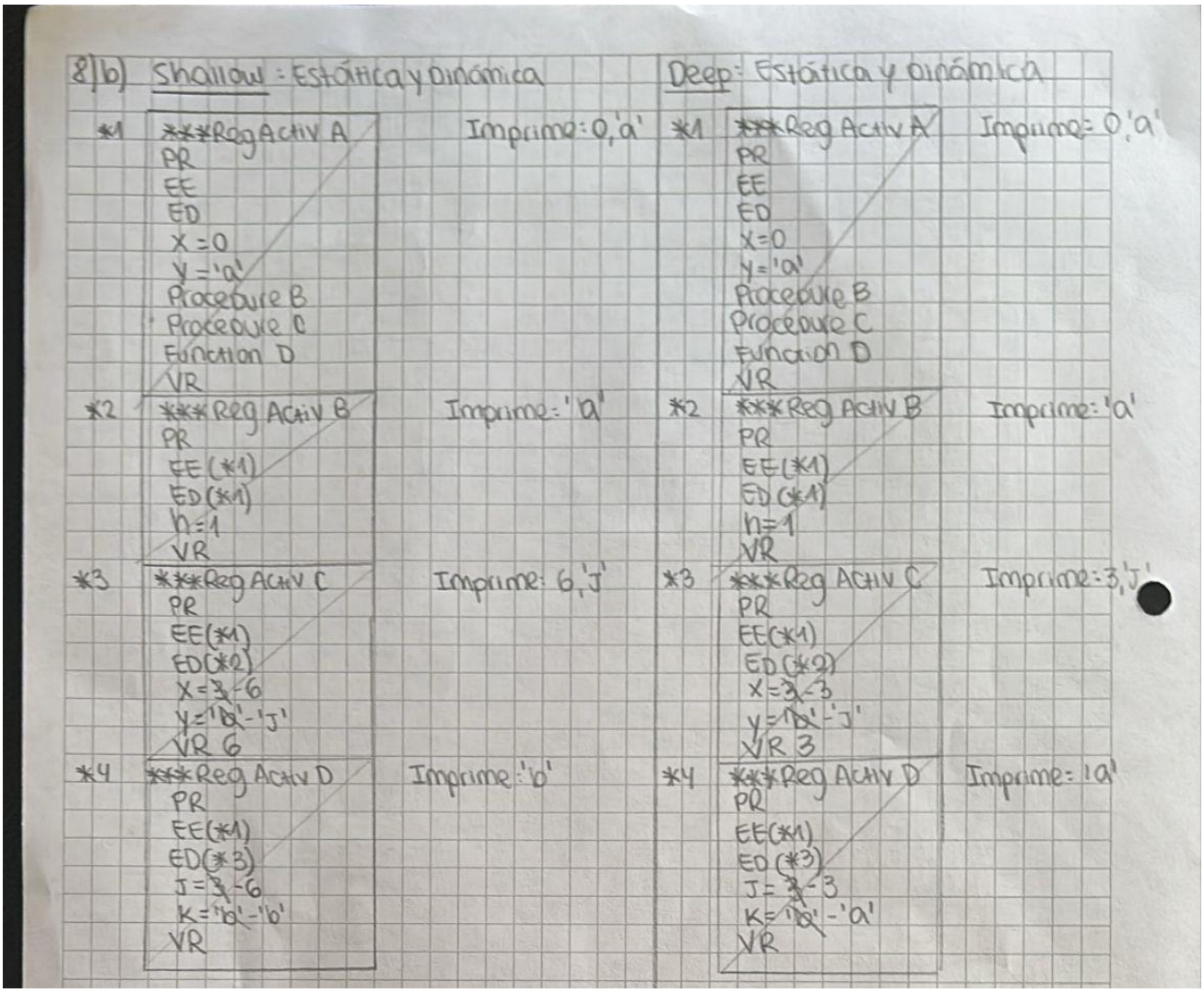
Ligadura shallow o superficial: El ambiente de referencia, es el del subprograma que tiene declarado el parámetro formal del subprograma.

Ligadura deep o profunda: El ambiente es el del subprograma dónde está declarado el subprograma usado como parámetro real. Se utiliza en los lenguajes con alcance estático y estructura de bloque.

b) Realice la pila estática y dinámica tanto con el pasaje de parámetros deep y shallow para el siguiente código.

```
Program A
  Var x:integer;
  Var y: char;
  Procedure B;
     Var h:integer;
     Begin
        h:=1+x;
        Write (y);
        C(D);
        Write (y);
     End;
  Procedure C (Subrutina S);
     Var x:integer;
     Var y: char;
     Begin
        x:=3;
        y:= "b";
        x:=S(x,y)
        y:= "j";
        Write (x,y);
     End;
```

```
  Function D (j:integer, k:char);
     Begin
        j:=j+x;
        k:=y;
        Write (k);
        Return j;
     End;
BEGIN
  x:=0;
  y:="a";
  B();
  Write (x,y);
END.
```



Ejercicio 9: Sea el siguiente código escrito en Pascal like

```
Procedure main
        a: array(1..5) of integer;
        x: integer;
        i;integer;
        Procedure Uno (tipo_pasaje
        m:integer)
                Begin
                x:=0;
                x:=x+1;
                m:=m+x + a(3);
                x:=x*2;
                a(3):=a(3) - 1;
                m:=m+1;
                End;
```

```
Begin
For i:=1 to 5 a(i):=1;
x:=3;
Uno(a(x));
For i:=1 to 5 write (a(i));
End.
```

a- Plantee diferencias, relacionada con la forma de implementación de cada uno y los resultados sobre este ejemplo considerando los siguientes tipos de pasajes parámetros nombre, referencia y valor resultado.

Nombre

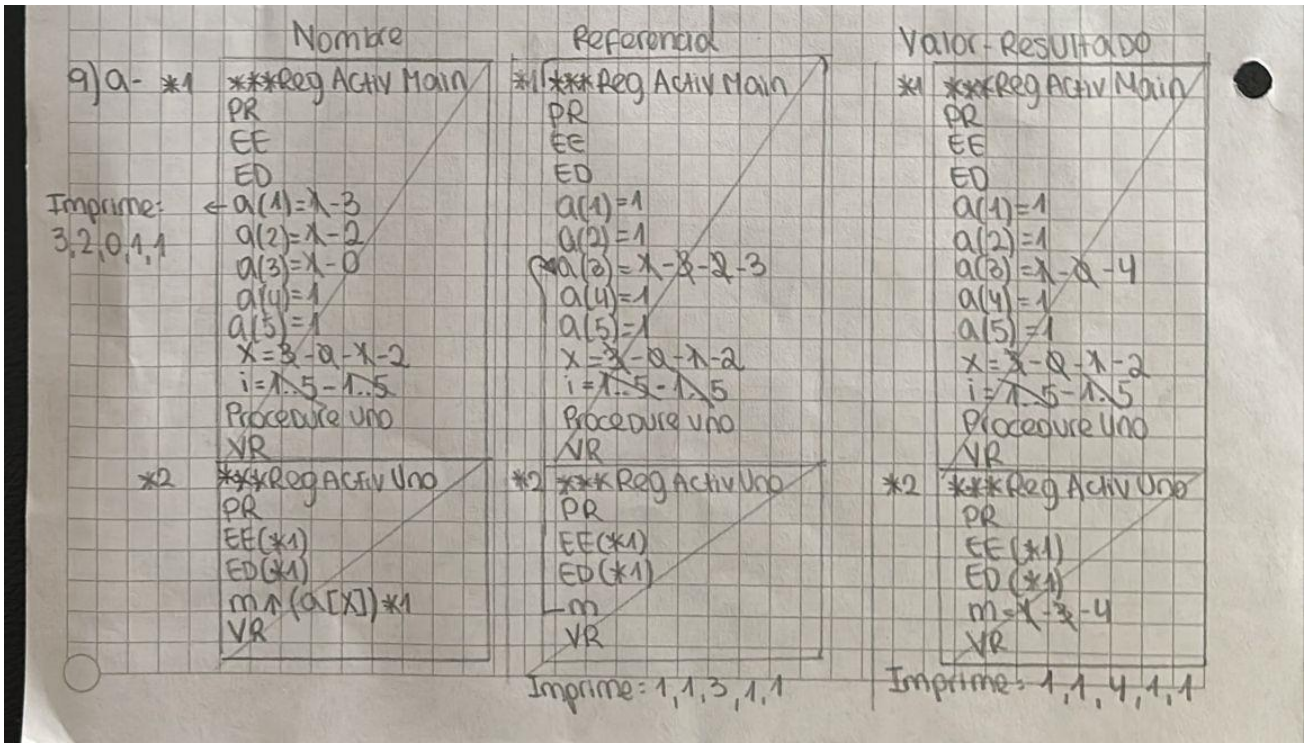
- El parámetro formal es sustituido textualmente por una expresión del parámetro real más un puntero al entorno del parámetro real. (Se maneja una estructura aparte que resuelve esto).
- Se establece la ligadura entre parámetro formal y parámetro real en el momento de la invocación, pero la "ligadura de valor" se difiere hasta el momento en que se lo utiliza (la dirección se resuelve en ejecución).
- Distinto a por referencia. Es decir, no apunta a una dirección fija, puede ir cambiando (pero el nombre tiene que ser el mismo).

Referencia

- El parámetro formal será una variable local que contiene la dirección al parámetro real de la unidad llamadora que estará entonces en un ambiente no local.
- Cualquier cambio que se realice en el parámetro formal dentro del cuerpo del subprograma quedará registrado en el parámetro real.

Valor/Resultado

- El parámetro formal es una variable local que recibe una copia (a la entrada) del contenido del parámetro real y el parámetro real (a la salida) recibe una copia de lo que tiene el parámetro formal.
- Básicamente lo que hace la rutina es copiar en la variable.
- Cada referencia al parámetro formal es una referencia local.



b- ¿Qué sucede si en Uno se agrega la siguiente declaración: x: integer? Indique el resultado para cada uno de los tipos de pasajes de parámetros (nombre, referencia y valor resultado)

Sólo cambia la impresión de los valores cuando es pasado el parámetro por nombre, quedando de esta manera: 1, 1, 3, 1, 1.

Ejercicio 10: Sea el siguiente un programa escrito en Pascal:

```
Program Uno;
    var x:integer;
    Function Dos:integer;
        begin
        x:= x + 1;
        return (x);
        end;
```

```
    Procedure Tres (pasaje x:integer);
        begin
        x:= x + 5;
        x:= Dos + 10;
        end;
begin
x:= 8; Tres(x);
write (x);
end.
```

a- Explique cómo simularía en Pascal el pasaje por valor-resultado y hágalo sobre este ejemplo.

Nota: No se pueden agregar más variables, ni cambiar el nombre de las que están.

```pascal
program Uno;
var x: integer;

function Dos(x: integer): integer;
begin
    x := x + 1;
    Dos := x;
end;

procedure Tres(x: integer);  (* Cambiamos a pasaje por valor *)
begin
    x := x + 5;
    x := Dos(x) + 10;  (* Pasamos x a Dos *)
    write(x);  (* Para simular el efecto, escribimos aquí *)
end;

begin
    x := 8;
    Tres(x);
    (* No podemos asignar el valor de vuelta sin una variable temporal *)
end.
```

b- Transcriba este ejemplo en Ada de manera tal que el resultado de la ejecución sea diferente si el pasaje de parámetros es por referencia y luego por valor – resultado

Referencia:

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Uno is
   X : Integer := 8;

   function Dos return Integer is
   begin
      X := X + 1;  -- Modifica directamente la variable global X
      return X;
   end Dos;

   procedure Tres (X : in out Integer) is
   begin
      X := X + 5;
      X := Dos + 10;
   end Tres;

begin
   Tres(X);
   Put_Line(Integer'Image(X));
end Uno;
```

Valor - resultado:

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Uno is
   X : Integer := 8;

   function Dos (Val : Integer) return Integer is
      Temp : Integer := Val;
   begin
      Temp := Temp + 1;  -- Trabaja con una copia local, no con la variable global
      return Temp;
   end Dos;

   procedure Tres (X : in out Integer) is
      Local_X : Integer := X;  -- Copia local para simular valor-resultado
   begin
      Local_X := Local_X + 5;
      Local_X := Dos(Local_X) + 10;  -- Pasa Local_X a Dos
      X := Local_X;  -- Asigna el valor final de vuelta a X
   end Tres;

begin
   Tres(X);
   Put_Line(Integer'Image(X));
end Uno;
```