Conceptos y Paradigmas de Lenguajes de Programación
2024

Práctica Nro. 6
Parámetros

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

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

	Resultado
	Valor
Modo IN	Valor / Resultado
Modo OUT	Nombre
modo IN / OUT	Resultado de funciones
	Valor Constante
	Referencia

Ejercicio 3:

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

Tipo de pasaje de parámetros	Lenguaje
	ADA
	C
	Rubi
	JAVA
	Python

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

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

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

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
b) (3,5,6),(4,6,7), 2, 2
c) (3,5,6),(5,5,6), 0, -1

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.

Que sucede en cada caso?

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.
```

Ejercicio 8:

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

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.

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)

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.

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

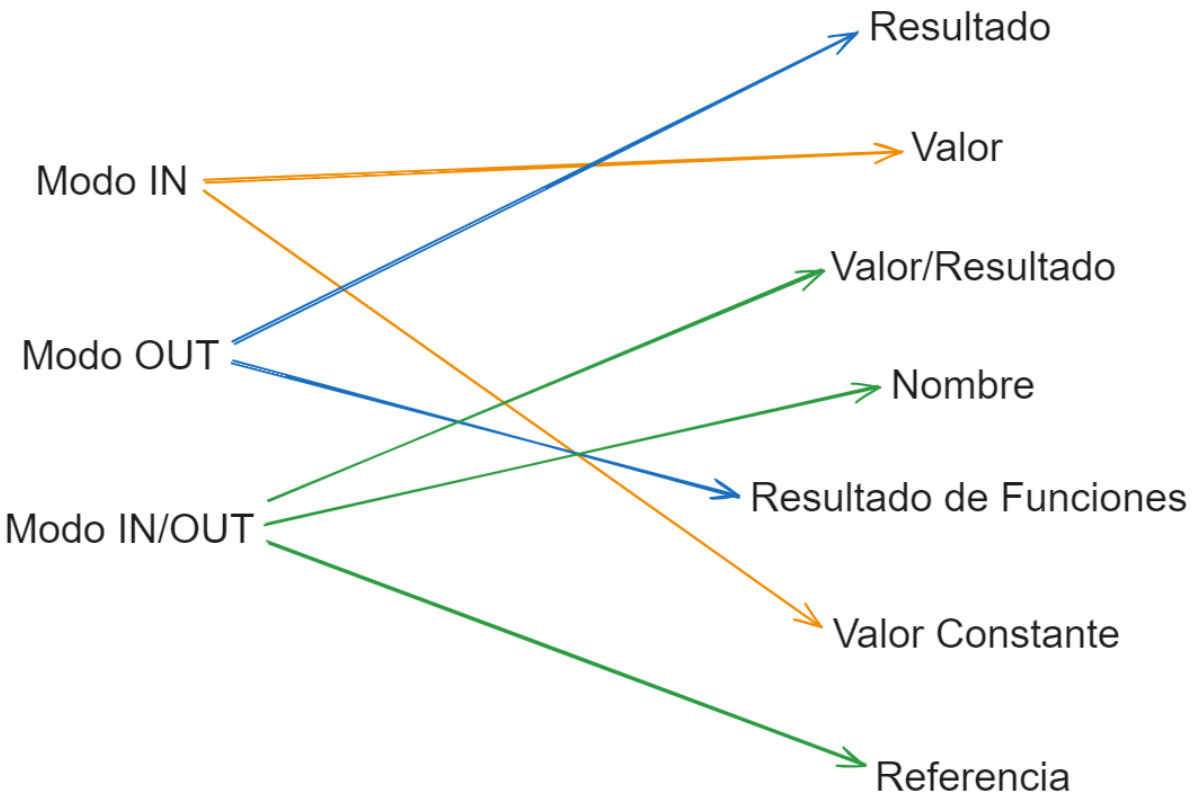
Ejercicio 1

Inciso a)

- **Parámetro:** Es una de las 2 formas de compartir datos entre Subrutinas. El Pasaje De Parámetros es mejor, ya que el uso intensivo de accesos al ambiente no local puede provocar alguna pérdida de control, y provocar que las variables terminen siendo visibles donde no es necesario y llevar a errores.

- **Parámetro Real:** Es un valor, entidad, expresión u otro, que puede ser local, no local o global, que se pasa a un procedimiento o función. Están colocados en la parte de la invocación de la rutina.
- **Parámetro Formal:** Es una variable utilizada para recibir valores de entrada en una rutina, subrutina etc. Están colocados en la parte de la declaración de la rutina. Son Variables locales a su entorno.
- **Ligadura Posicional:** Los parámetros reales se corresponden con la posición que ocupan en la lista de parámetros formales. Van en el mismo orden.
- **Ligadura por Palabra clave o Nombre:** Los parámetros reales se corresponden con el nombre por lo tanto pueden estar colocados en distinto orden en la lista de parámetros formales. La desventaja que tiene es que cuando invocas hay que conocer/recordar el nombre de los parámetros formales para poder hacer la asignación. Esto puede llevar a cometer errores.

Ejercicio 2



Ejercicio 3

Inciso a)

Tipos de Pasajes de Parámetros	*Lenguaje*

Modo IN: Pasaje por valor o por valor constante. **Modo OUT:** Pasaje por resultado. **Modo IN/OUT:** Pasaje por referencia.	**ADA**
Modo IN: Pasaje por valor. **Modo OUT:** No tiene un modo OUT explícito, pero se simula usando punteros (pasaje por referencia). **Modo IN/OUT:** Se usa el pasaje por referencia mediante punteros.	**C**
Modo IN: Pasaje por valor (aunque técnicamente es pasaje por referencia de objetos, pero para tipos primitivos se comporta como pasaje por valor). **Modo OUT:** No tiene un modo OUT explícito. **Modo IN/OUT:** Pasaje por referencia de objetos, pero los tipos primitivos se comportan como pasaje por valor.	**Ruby**
Modo IN: Pasaje por valor (para tipos primitivos) y pasaje por referencia (para objetos, aunque técnicamente es una referencia a un objeto). **Modo OUT:** No tiene un modo OUT explícito. **Modo IN/OUT:** No se permite directamente. Se puede simular mediante el uso de objetos.	**JAVA**
Modo IN: Pasaje por valor para tipos inmutables (int, float, str, etc.) y pasaje por referencia para tipos mutables (listas, diccionarios, etc.). **Modo OUT:** No tiene un modo OUT explícito. **Modo IN/OUT:** Pasaje por referencia para objetos mutables.	**Python**

Inciso b)

- Ada es más seguro que Pascal en el contexto del pasaje de parámetros debido a su especificación clara de modos de pasaje, chequeos de tipos estrictos, protección contra aliasing y la posibilidad de definir contratos de subprogramas, todo lo cual contribuye a un código más robusto y menos propenso a errores.

Inciso c)

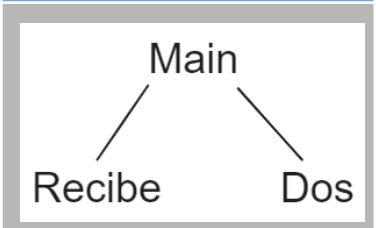
- **Tipos Escalares (Enteros, Flotantes, Booleanos, etc.)**
  - Para tipos escalares, el parámetro in out se pasa por referencia. Esto significa que el subprograma recibe una referencia al objeto original, y cualquier modificación al parámetro dentro del subprograma afecta directamente al objeto original.
- **Tipos Compuestos (Registros, Arreglos, etc.)**
  - Para tipos compuestos, como los registros y los arreglos, el pasaje de parámetros in out también se realiza por referencia. Esto permite a la subprograma modificar directamente los componentes del objeto original.
- **Tipos Acceso (Punteros)**

- Para tipos acceso (punteros), el parámetro in out se refiere al puntero en sí, no al objeto apuntado. Esto significa que el subprograma puede cambiar el puntero para que apunte a un objeto diferente, además de poder modificar el contenido del objeto apuntado.

- **Tipos Protegidos**
  - Los tipos protegidos en Ada, que son utilizados para la concurrencia segura, también pueden ser pasados como in out. Estos tipos son pasados por referencia y cualquier acceso a ellos está sujeto a las reglas de exclusión mutua, asegurando la consistencia del acceso concurrente.

Ejercicio 4

Inciso a)

Árbol de Anidamiento Sintáctico	***Registro de Activación de Main***	***Registro de Activación de Recibe***	***Registro de Activación de Dos***
	Registro de Activación Main Punto de Retorno j m i Procedure Recibe Procedure Dos Valor de Retorno	Registro de Activación Recibe Punto de Retorno Link Estático Link Dinámico x y Valor de Retorno	Registro de Activación Dos Punto de Retorno Link Estático Link Dinámico m Valor de Retorno

Inciso b)

i) Modo IN/OUT - Referencia

Código	***Cadena Estática***	***Cadena Dinámica***
`` 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; ```	*1 Registro de Activación Main Punto de Retorno j = 3 -> 5 m = 2 -> 6 i = 1 -> 5 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5	*1 Registro de Activación Main Punto de Retorno j = 3 -> 8 m = 2 i = 1 -> 5 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 -> 9

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.	Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x ↑ (i - *1) y ↑ (j - *1) Valor de Retorno Imprime 5, 5, 5, 5, 6 5, 5, 5 5, 5, 6	Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x ↑ (i - *1) y ↑ (j - *1) Valor de Retorno Imprime 5, 8, 5, 8, 9 5, 8, 9 5, 8, 2

ii) Modo IN - Valor

Código	***Cadena Estática***	***Cadena Dinámica***
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;	*1 Registro de Activación Main Punto de Retorno j = 3 m = 2 -> 6 i = 1 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x = 1 -> 5 y = 3 -> 5 Valor de Retorno Imprime 5, 5, 1, 3, 6	*1 Registro de Activación Main Punto de Retorno j = 3 m = 2 -> 6 i = 1 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 -> 9 Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x = 1 -> 5 y = 3 -> 8 Valor de Retorno Imprime 5, 8, 1, 3, 9

begin m:= 2; i:=1; j:=3; Dos; write (i, j, m); end.	1, 3, 5 1, 3, 6	1, 3, 9 1, 3, 2

iii) Modo IN/OUT - Valor/Resultado

Código	***Cadena Estática***	***Cadena Dinámica***
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.	*1 Registro de Activación Main Punto de Retorno j = 3 -> 5 m = 2 -> 6 i = 1 -> 5 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x = 1 -> 5 y = 3 -> 5 Valor de Retorno Imprime 5, 5, 1, 3, 6 5, 5, 5 5, 5, 6	*1 Registro de Activación Main Punto de Retorno j = 3 -> 8 m = 2 i = 1 -> 5 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 -> 9 Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x = 1 -> 5 y = 3 -> 8 Valor de Retorno Imprime 5, 8, 1, 3, 9 5, 8, 9 5. 8, 2

iv) Modo IN/OUT - Nombre

Código	*Cadena Estática*	*Cadena Dinámica*
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.	*1 Registro de Activación Main Punto de Retorno j = 3 -> 5 m = 2 -> 6 i = 1 -> 5 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x ↑ (i - *1) y ↑ (j - *1) Valor de Retorno Imprime 5, 5, 5, 5, 6 5, 5, 5 5, 5, 6	*1 Registro de Activación Main Punto de Retorno j = 3 -> 8 m = 2 i = 1 -> 5 Procedure Recibe Procedure Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) m = 5 -> 9 Valor de Retorno *3 Registro de Activación Recibe Punto de Retorno Link Estático (*1) Link Dinámico (*2) x ↑ (i - *1) y ↑ (j - *1) Valor de Retorno Imprime 5, 8, 5, 8, 9 5, 8, 9 5, 8, 2

v) Modo OUT - Resultado

Código	*Cadena Estática*	*Cadena Dinámica*
Procedure Main; var j, m, i: integer; Procedure Recibe (x:integer; y:integer); begin	No es posible realizarlo. Se genera un error al querer ejecutar la sentencia “m:= m+1+y;” del Procedure Recibe ya que el parámetro “y” no fue inicializado.	No es posible realizarlo. Se genera un error al querer ejecutar la sentencia “m:= m+1+y;” del Procedure Recibe ya que el parámetro “y” no fue inicializado.

```
    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.
```

Incisos c) y d)

- Respondidos en **Inciso b).**

Ejercicio 5

Inciso a)

- Para conseguir esos resultados se debe usar Modo IN/OUT por Referencia en los 3 parámetros.

Inciso b)

- Para conseguir esos resultados se debe usar Modo IN/OUT por Referencia para los parámetros *iteración y vit* y Modo IN por Valor para el parámetro *vector*.

Inciso c)

- Para conseguir esos resultados se puede usar Modo IN por Valor o Modo IN/OUT por Valor/Resultado en cada uno de los parámetros cuales fuera.

Ejercicio 6

- **Un valor entero:** Un único valor se comporta exactamente igual que el pasaje por referencia.
- **Una constante:** Si es una constante es equivalente a por valor.
- **Un elemento de un arreglo:** Si es un elemento de un arreglo puede cambiar el suscripto entre las distintas referencias.
- **Una expresión:** Sí es una expresión se evalúa cada vez.

Ejercicio 7

Código	***Cadena Estática***	***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;	*1 Registro de Activación Uno Punto de Retorno y = 1...6 -> 1...5 -> 1 -> 2 -> 1...6 -> 1...5 z = 2 -> 3 r1[1] = 2 r1[2] = 2 -> 3 r1[3] = 2 r1[4] = 2 -> 4 r1[5] = 2 r1[6] = 2 r2[1] = 1 r2[2] = 1 -> 2 -> 3 -> 4 r2[3] = 1 -> 3 -> 5 r2[4] = 1 r2[5] = 1 Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) x ↑ (r1[y + r2[y]] - *1) t ↑ (r2[z] - *1) io ↑ (y - *1) y = 2 -> 3 Dos Valor de Retorno *3 Registro de Activación Dos Punto de Retorno Link Estático (*2) Link Dinámico (*2) t1 ↑ (t - *2) Valor de Retorno	*1 Registro de Activación Uno Punto de Retorno y = 1...6 -> 1...5 -> 1 -> 2 -> 1...6 -> 1...5 z = 2 -> 3 r1[1] = 2 r1[2] = 2 -> 3 r1[3] = 2 r1[4] = 2 -> 4 r1[5] = 2 r1[6] = 2 r2[1] = 1 r2[2] = 1 -> 2 -> 3 -> 4 r2[3] = 1 -> 3 -> 5 r2[4] = 1 r2[5] = 1 Dos Valor de Retorno *2 Registro de Activación Dos Punto de Retorno Link Estático (*1) Link Dinámico (*1) x ↑ (r1[y + r2[y]] - *1) t ↑ (r2[z] - *1) io ↑ (y - *1) y = 2 -> 3 Dos Valor de Retorno *3 Registro de Activación Dos Punto de Retorno Link Estático (*2) Link Dinámico (*2) t1 ↑ (t - *2) Valor de Retorno

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.	+4 Registro de Activación Tres Punto de Retorno Link Estático (*3) Link Dinámico (*3) Valor de Retorno Imprime 2 3 2 2 4 2 1 4 5 1 1	+4 Registro de Activación Tres Punto de Retorno Link Estático (*3) Link Dinámico (*3) Valor de Retorno Imprime 2 3 2 2 4 2 1 4 5 1 1

Ejercicio 8

Inciso a)

- **Ligadura shallow o superficial**
  - El ambiente Referencia, es el del subprograma que tiene declarado el parámetro formal del subprograma.
- **Ligadura deep o profunda**
  - El Ambiente es del subprograma dónde está declarado el subprograma usado como parámetro real. Se utiliza en los lenguajes con alcance estático y estructura de bloque.

Inciso b)

Ligadura Shallow

Código	***Cadena Estática y Dinámica***
Program A Var x:integer; Var y: char; Procedure B;	*1 Registro de Activación A Punto de Retorno x = 0 y = a B C D

```
  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.
```

```
  Valor de Retorno

*2  Registro de Activación B
  Punto de Retorno
  Link Estático (*1)
  Link Dinámico (*1)
  h = 1
  Valor de Retorno

*3  Registro de Activación C
  Punto de Retorno
  Link Estático (*1)
  Link Dinámico (*2)
  x = 3 -> 6
  y = b -> j
  Valor de Retorno 6

*4  Registro de Activación D
  Punto de Retorno
  Link Estático (*1)
  Link Dinámico (*3)
  j = 6
  k = b
  Valor de Retorno

Imprime
  a
  b
  6, j
  a
  0, a
```

Ligadura Deep

Código	***Cadena Estática y Dinámica***
`Program A` `  Var x:integer;`	*1 Registro de Activación A Punto de Retorno x = 0

```
  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.
```

```
  y = a
  B
  C
  D
  Valor de Retorno

*2  Registro de Activación B
  Punto de Retorno
  Link Estático (*1)
  Link Dinámico (*1)
  h = 1
  Valor de Retorno

*3  Registro de Activación C
  Punto de Retorno
  Link Estático (*1)
  Link Dinámico (*2)
  x = 3 -> 3
  y = b -> j
  Valor de Retorno 3

*4  Registro de Activación D
  Punto de Retorno
  Link Estático (*1)
  Link Dinámico (*3)
  j = 3
  k = a
  Valor de Retorno

Imprime
  a
  a
  3, j
  a
  0, a
```

Ejercicio 9

Inciso a)

- **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 lo copio en la variable. Cada referencia al parámetro formal es una referencia local.
- **Resultados de Impresión:**
    - **Nombre:** 3, 2, 0, 1, 1
    - **Referencia:** 1, 1, 3, 1, 1
    - **Valor/Resultado:** 1, 1, 4, 1, 1

Inciso b)

- El único que cambia en su impresión es el de por **Nombre** y termina imprimiendo 1, 1, 3, 1, 1.

Ejercicio 10

- Perdón pero no voy a codear esto.