

# Practica 4 - Variables

## TIPS que tengo para ayudarme a resolver esta practica.

Las variables estaticas, tienen R-Value 0, su tiempo de vida arranca en 1 y va hasta el final del programa. Ademas, el tiempo de vida se debe poner entre <>.

Por ejemplo si el programa termina en la linea 30, y la variable estatica tiene el nombre de numeros entonces, el Tiempo de vida de numeros seria <1...30>

Ademas, creo que las variables que sean globales y locales al programa, arrancan con R-Value=0

Mientras que las que se encuentran en procedimientos/funciones, su R-Value es indefinido.

El alcance de las funciones es desde donde se declaran hasta el final del codigo (CREO)

## Algunos apuntes (por lo que entendi)

### L-Value:

Es la direccion de memoria ligada a la variable durante la ejecucion

En el L-Value la alocaion de memoria puede ser:

- Estatico: Se hace en compilacion (cuando se carga el programa en memoria y perdura hasta el final de la ejecucion)
- Dinamico: Se hace en tiempo de ejecucion
  - Automatica: Cuando aparece una declaracion de una variable en la ejecucion
  - Explicita: Requerida por el programador con alguna sentencia de creacion

### R-Value:

Valor asignado en memoria (segun el lenguaje).

**Las constantes son estaticas en cuanto a su R-Valor**

### Alcance:

Rango de instrucciones en el que es conocido el nombre de la variable.

- Alcance de una funcion: Desde que se define hasta que termina.

## ATENCION!!!

**En la practica en el alcance de variables se toma la linea siguiente a su declaracion.**

## Tiempo de vida:

Intervalo desde que una variable existe en memoria hasta que se desaloca de memoria.  
Desde que se inicia el bloque que lo contiene, hasta que finaliza el bloque que lo contiene.

## Ejercicio 1: Indique y defina cuales son sus atributos:

### Inciso A)

Identificador	L - Valor	R - Valor	Alcance	Tiempo de Vida
a	Automatica	indefinido? o 0 ?	5 ... 16	1 ... 16

### Inciso B)

DUDA: Cuando trabajamos con punteros, hay que poner el tiempo de vida en ambas memorias?

Y que pasa con su R Valor? por que antes de que se haga el new, no se cual seria...  
Cuando se le hace un new, recién ahí el puntero tiene nil?

Identificador	L - Valor	R - Valor	Alcance	Tiempo de Vida
p	estatico	indefinido	4 ... 16	1 ... 16
^ P		nill	8..15	7 ... 15

## Ejercicio 2

### Inciso A)

Cuales son las formas de inicializar una variable en el momento de su declaracion?

- Inicializacion directa:

```
int x = 10; // En Java
```

```
x = 10 ## En Python
```

- Inicializacion con valor por defecto (implicito):

```
static int x; // En C: valor por defecto 0
```

- Inicializacion con resultado de una funcion:

```
int x = obtenerValor(); // JAVA / C
```

- Inicializacion multiple (en una sola linea)

```
int a,b,c,d,e,f,g ;
```

- Inicializacion con expresion matematica

```
int x = 10 - 7 ; // En Java, X = 3
```

## Inciso B)

Característica	Java	C	Python	Ruby
Tipado	Estático	Estático	Dinámico	Dinámico
Declaración + inicialización	int x=10;	int x = 10;	x = 10	x = 10
Valor por defecto	Sí (en campos)	Sí (global) No (local)	Sí (tipo dinámico)	Sí ( nil )
Inicialización múltiple	No	No	Sí: a, b = 1, 2	Sí: a, b = 1, 2
Con función o expresión	Sí	Sí	Sí	Sí

## Ejercicio 3: Conceptos asociados al L - Valor de una variable

El **L-valor** (location value) representa **la dirección de memoria** donde vive una variable. Dependiendo del tipo de almacenamiento, ese L-valor puede ser fijo o cambiar durante la ejecución.

Tipo de Variable	Definicion	Lenguaje	Ejemplo
Estatica	Su L-Valor es fijo en memoria durante toda la ejecucion del	Ejemplo en C	static int contador = 0;  Contador : Integer := 0;

Tipo de Variable	Definicion	Lenguaje	Ejemplo
	programa. Se aloja en memoria estatica	Ejemplo en Ada	
Automatica o semiAutomatica	Su L-Valor es fijo,. Mientras que la funcion/procedimiento esta activo.  Se libera al salir del bloque	Ejemplo en C	void ejemplo() { int x = 5; }
Dinamica	Su L-Valor es asignado en tiempo de ejecucion. Se asigna por medio de una operacion como new Vive en el heap (puede cambiar durante la ejecucion)	Ejemplo en C  Ejemplo en Ada	int p = (int)malloc(sizeof(int)); *p = 10;  type Int_Ptr is access Integer;  P : Int_Ptr := new Integer'(10);
Semidinamica	Su L-Valor se asigna dinamicamente solo cuando se activa el bloque. Pero el tamaño o la estructura pueden variar.	Ejemplo en C	void ejemplo(int n) { int arreglo[n]; }

## Investigación: Tipos de variables respecto a su L-valor en C y Ada

Lenguaje	Variable Estática	Variable Automática	Variable Dinámica	Variable Semidinámica
<b>C</b>	<code>static int x;</code>	<code>int x;</code> dentro de función	<code>malloc()</code>	<code>VLA ( int x[n]; )</code>
<b>Ada</b>	Variables globales	Declaradas en bloques	<code>new</code> con <code>access</code>	Arrays con tamaño dinámico

## Ejercicio 4

# A que se denomina variable local y a que se denomina variable global?

## Variable local:

Es una variable declarada dentro de una función, método o bloque de código. Solo se puede acceder a ella dentro del mismo ámbito en el que fue definida. Se crea cuando se entra al bloque y se destruye al salir.

## Variable global:

Es una variable declarada fuera de cualquier función o clase (según el lenguaje). Puede ser accedida por cualquier parte del programa.

## b. ¿Una variable local puede ser estática respecto de su l-valor? En caso afirmativo, dé un ejemplo.

Sí, una variable local puede ser **estática respecto de su l-valor**, lo que significa que **mantiene su valor entre distintas ejecuciones de la función**, pero **su ámbito sigue siendo local**.

## c. ¿Una variable global siempre es estática? Justifique la respuesta.

Sí, una variable global **es estática en cuanto a su duración**, es decir, **existe durante toda la ejecución del programa** (tiene almacenamiento estático).  
**Pero no necesariamente tiene el modificador `static` aplicado.**

La palabra “estática” en este contexto se refiere a su **tiempo de vida**, no a su **visibilidad o alcance**.

## d. Indique qué diferencia hay entre una variable estática respecto de su l-valor y una constante.

Característica	Variable estática (respecto a l-valor)	Constante
Valor	Puede cambiar durante la ejecución	No puede cambiar
Duración	Es persistente (almacenamiento estático)	También puede ser persistente
Modificable	Sí (puede reasignarse)	No (valor fijo)
Ejemplo (C)	<code>static int x = 5;</code>	<code>const int y = 10;</code>

## Conclusión:

- Una variable **estática** mantiene su valor entre llamadas o durante toda la ejecución.
- Una **constante** no puede ser modificada, aunque puede tener almacenamiento estático.

## Ejercicio 5:

A) En Ada hay dos tipos de constantes: las numéricas y las comunes. ¿A qué se debe dicha clasificación?

Esta clasificación se debe al **momento de ligadura** (binding time) del valor de la constante.

### ► Constantes numéricas:

- Son constantes literales como `2`, `3.5`, etc.
- **No tienen un tipo ligado inicialmente.**
- El tipo se determina **cuando se usa** (ligadura diferida).
- Se comportan más como valores universales que se ajustan al tipo según el contexto.

### Constantes comunes:

- Son declaradas explícitamente con `constant`, como en:


```
H : constant Float := 3.5;
```

- Tienen un **tipo asignado explícitamente o inferido** y una **ligadura temprana**.
- Su valor se fija **en tiempo de compilación**.


b) En base a lo anterior, determine el momento de ligadura de las constantes del siguiente código:

```
H: constant Float := 3.5;  
I: constant := 2;  
K: constant Float := H * I;
```



### ► H: constant Float := 3.5;

- **Tipo:** `Float` (declarado explícitamente)
- **Valor:** `3.5`, una constante **numérica** pero con tipo ya ligado.
-  **\*\*Ligadura temprana** (en tiempo de compilación).

### ► I: constant := 2;

- No tiene tipo explícito → el compilador infiere el tipo.
- Como `2` es una constante **numérica**, su tipo se liga **cuando se use**.
-  **Ligadura diferida** → el tipo se determina cuando se evalúa su contexto de uso.

## ► **K: constant Float := H \* I;**

- H es de tipo `Float`.
- I es una constante numérica que, al ser usada en `H * I`, será convertida a `Float`.
- Por lo tanto, **I se liga a tipo Float en ese momento.**
- K también es declarado explícitamente como `Float`, con ligadura temprana.
-  **K:** ligadura temprana (compilación)
-  **I:** ligadura diferida → en este caso, se liga a tipo Float **al usarse en la expresión** `H * I`.

## Ejercicio 6:

Tipo de variable	Visibilidad	Duración (alocación de memoria)
<b>Global</b> ( <code>int x=1</code> )	Visible desde cualquier función	Desde el inicio del programa hasta que termina
<b>Local</b>	Solo dentro de su función	Se crea cada vez que se llama la función y se destruye al salir
<b>Local estática</b> ( <code>static int x</code> )	Solo visible dentro de la función	Se <b>crea una sola vez</b> (alocación estática), y <b>conserva su valor entre llamadas</b>

Con respecto a su alocacion en memoria, sí, tienen el mismo comportamiento, una variable estática local se aloca desde el inicio del programa, como una global, pero su visibilidad está restringida a su función.

## ¿Qué pasa con la aloación de memoria?

Variable	¿Cuándo se aloca?	¿Dónde se guarda en memoria?
Global	En tiempo de carga (inicio del programa)	En la sección de datos globales (data segment)
Static local	También en tiempo de carga	Igual que una global, en sección estática
Local normal	En cada llamada (tiempo de ejecución)	En la <b>pila (stack)</b> del programa

Ambas variables se alocan de forma **estática** (una vez en la vida del programa), pero **su alcance o visibilidad es diferente.**

## Ejercicio 7 Indique para los identificadores si son globales o locales.

```
Clase Persona{
public long id // local a la instancia
public string nombreApellido // local a la instancia
public Domicilio domicilio // local a la instancia
private string dni; // local a la instancia
public string fechaNac; // local a la instancia
public static int cantTotalPersonas; // global de la clase
}

// en los metodos no se pueden declarar variables con el modificador de acceso
public
// asumiendo que eso no es posible, dejo los resultados:
public int getEdad(){
    public int edad=0; // local al metodo
    public string fN = this.getFechaNac(); // local al metodo
    return edad;
}

Clase Domicilio {
public long id; // local a la instancia
public static int nro // global a la clase
public string calle // local a la instancia
public Localidad loc; // local a la instancia
}
```

### Ejercicio 8:



```

1- Program Uno;
2- type tpuntero= ^integer;
3- var mipuntero: tpuntero;
4- var i:integer;
5- var h:integer;
6- Begin
7-     i:=3;
8-     mipuntero:=nil;
9-     new(mipuntero);
10-    mipuntero^:=i;
11-    h:= mipuntero^+i;
12-    dispose(mipuntero);
13-    write(h);
14-    i:= h- mipuntero;
15- End.

```

a) rango de instrucciones del tiempo de vida de las variables i , h , mipuntero

Variable	Tiempo de Vida
i	1 .... 15
h	1 ... 15
mipuntero	1 ... 15
^ mipuntero	9... 12

b) rango de instrucciones que representa el alcance de las variables i , h , mipuntero

Variable	Alcance
i	5 ... 15
h	6. ... 15
mipuntero	6 ...15
^ mipuntero	10...12?

c) indique si el programa anterior presenta un error al intentar escribir el valor de h. Justifique  
 No presenta ningun problema, porque antes de eliminar el puntero, se le asigna el valor al que apuntaba el puntero, por lo tanto en la variable h, vamos a tener el valor apuntado por el puntero mipuntero

d) indique si el programa anterior presenta un error al intentar asignar a i la resta de h con mipuntero. Justifique.

Si, ocurre un error, al momento de hacer la resta, se quiere restar la dirección de memoria al valor que apunta mipuntero, pero ese valor es nil, ya que antes se hizo un dispose y se eliminó de memoria dinámica el contenido apuntado por el puntero.

e) Determine si existe otra entidad que necesite ligar los atributos de alcance y tiempo de vida para justificar las respuestas anteriores. En ese caso indique cuál es la entidad y especifique su tiempo de vida y alcance.

Si, la otra entidad que existe es el programa principal, el cual es una entidad que necesita ligar los atributos de alcance y tiempo de vida para justificar las respuestas anteriores. En el caso de Pascal, el programa principal tiene alcance global y su tiempo de vida es desde su declaración hasta el final del programa (línea 6 – línea 15)

f) Especifique el tipo de variable de acuerdo a la ligadura con el l-valor de las variables que encontró en el ejercicio.

i y h tienen son de tipo integer, tienen ligadura estática.

mipuntero es de tipo puntero, tiene ligadura estática, pero ligadura dinámica para la dirección que apunta

## Ejercicio 9: Elija un lenguaje y escriba un ejemplo:

Vamos con Java

a. En el cual el tiempo de vida de un identificador sea mayor que su alcance

```
1  public class EjemploA {
2      public List<Runnable> crearTareas() {
3          List<Runnable> tareas = new ArrayList<>();
4
5          for (int i = 0; i < 3; i++) {
6              int finalI = i;
7              tareas.add(() -> System.out.println("Tarea " + finalI));
8          }
9
10         return tareas;
11     }
```

Tiempo de vida:

- Desde que se ejecuta el bloque que lo contiene
- Desde línea 2 hasta línea 11

Alcance:

- Solamente dentro del for
- Desde línea 6 hasta línea 8 se toma la siguiente línea en la que se declara

b. En el cual el tiempo de vida de un identificador sea menor que su alcance

```

Usuario a = new Usuario();    // (1)
a.nombre = "Juan";           // (2)
a = new Usuario("Pepito");    // (3)

```

El alcance de la variable a, iría desde la línea 1 hasta la línea 3.

Mientras que el tiempo de vida es (una línea después de su declaración), sería la línea 2 únicamente, porque en la línea 3 se está creando otro usuario.

c. En el cual el tiempo de vida de un identificador sea igual que su alcance

```

1  program TiempoVidaIgualAlcance;
2  var
3  x: Integer;
4  begin
5  x := 10;
6  WriteLn(x);
7  end.

```

### Ejercicio 10:

Sí, en los tres lenguajes se puede asegurar que la variable c tiene alcance y tiempo de vida limitados al procedimiento en el que se encuentra definida, ya que no hay definiciones de procedimientos internos que puedan afectar su alcance o tiempo de vida.

### Ejercicio 11:

a) Responda Verdadero o Falso para cada opción.

**¿El tipo de dato de una variable es?**

- I) Un string de caracteres que se usa para referenciar a la variable y operaciones que se pueden realizar sobre ella. [Falso]
- II) Conjunto de valores que puede tomar y un rango de instrucciones en el que se conoce el nombre. [Falso]
- III) Conjunto de valores que puede tomar y lugar de memoria asociado con la variable. [Falso]
- IV) Conjunto de valores que puede tomar y conjunto de operaciones que se pueden realizar sobre esos valores. [Verdadero]

b) Escriba la definición correcta de tipo de dato de una variable

Un tipo de dato es una abstracción que define:

- Un conjunto de valores que una variable puede tomar

- El conjunto de operaciones validas que se pueden realizar sobre estos valores.

## Ejercicio 12

Ident.	Tipo	r-valor	Alcan ce	T.V.
a (línea 4)	automática	basura	5-14	1-14

DUDA: Aca el problema es si el tiempo de vida arranca en 1 o en 2 ?

Ya que de la línea 2 a la línea 6 (inclusive), son todas variables que le pertenecen al bloque main, y el bloque main arranca en el instante 2, no en el 1.

```

1. with text_io; use text_io;
2. Procedure Main is;
3. type vector is array(integer range <>);
4. a, n, p:integer;
5. v1:vector(1..100);
6. c1: constant integer:=10;
7. Procedure Uno is;
8. type puntero is access integer;
9. v2:vector(0..n);
10. c1, c2: character;
11. p,q: puntero;
12. begin
13.   n:=4;
14.   v2(n):= v2(1) + v1(5);
15.   p:= new puntero;
16.   q:= p;
17.   .....
18.   free p;
19.   .....
20.   free q;
21.   .....
22. end;
23.begin
24.   n:=5;
25.   .....
26.   Uno;
27.   a:= n + 2;
28.   .....
29. end

```

Identificador	Tipo	R-Valor	Alcance	T.V
a (línea 4)	automatica	basura	5 .. 14	1 .. 29
Main (línea 2)			3..14	2.. 29

Identificador	Tipo	R-Valor	Alcance	T.V
n (línea 4)	automatica	basura	5..14	2..29
p (línea 4)	automatica	basura	5 .. 11 23 .. 29	2.. 29
v1 (línea 5)	semidinamica	basura	6. . 14	2.. 29
c1 (línea 6)	automatica	10	7.. 10 23 .. 29	2... 29
Uno() (línea 7)			8.. 29	7..22
v2 (línea 9)	semidinamica	basura	10..22	7.. 22
c1 (línea 10)	automatica	basura	11..22	7.22
c2 (línea10)	automatica	basuar	11.22	7..22
p (línea11)	automatica	nil	12..22	7..22
q (línea11)	automatica	nil	12..22	7.22
^ p (línea 15)	dinamica	basura	12..22 ?	15..18?
^ q (línea 16)	dinamica	baqsura	12..22?	16..20?

## Ejercicio 13:

**Justifique la respuesta. El nombre de una variable puede condicionar:**

- a) **Su tiempo de vida:** El nombre de una variable no condiciona su tiempo de vida. El tiempo de vida de una variable no depende del nombre que tenga.
- b) **Su alcance:** El nombre una variable puede condicionar su alcance de, ya que el alcance de una variable es el rango de instrucciones en el que se conoce el nombre.
- c) **Su r-valor:** El nombre de una variable no condiciona su r-valor, ya que el r-valor se determina por la asignación de un valor a la variable (al nombre). Lo que si permite el nombre es poder acceder, mediante ese nombre, a la variable.
- d) **Su tipo:** El nombre de una variable no condiciona directamente su tipo, ya que el tipo se define explícitamente en la declaración de la variable. Sin embargo, es importante elegir un nombre descriptivo para la variable que refleje el tipo de dato que almacena, lo que puede hacer que sea más fácil para los programadores entender y trabajar con el código que utiliza la variable.

## Ejercicio 14:

## ARCHIVO1.C

```
1.    int v1;
2.    int *a;
3.    int fun2 ()
4.    { int v1, y;
5.        for(y=0; y<8; y++)
6.        { extern int v2;
7.            ...}
8.    }
9.    main()
10.   {static int var3;
11.       extern int v2;
12.       int v1, y;
13.       for(y=0; y<10; y++)
14.       { char var1='C';
15.           a=&v1;}
16.   }
```

## ARCHIVO2.C

```
17.   static int aux;
18.   int v2;
19.   static int fun2( )
20.   { extern int v1;
21.       aux=aux+1;
22.       ...
23.   }
24.   int fun3( )
25.   { int aux;
26.       aux=aux+1;
27.       ...
28.   }
```

Identificador	Tipo	R-Valor	Alcance	T.V
v1 (linea 1)	automatica	0	2..4 9..12 21..23	1..28
a (linea 2)	automatica	null	3..16	1..28
*a (linea 2)	dinamica	indef	3..16	15..16
int fun2() (linea3)			4..16	3..8
v1 (linea 4)	automatica	indef	5..8	3..8
y (linea 4)	automatica	indef	5..8	3..8

Identificador	Tipo	R-Valor	Alcance	T.V
main (linea 9)			10..16	9..16
static var3 (linea10)	estatica	0	11..16	<1..28>
v1 (linea 12)	automatica	indef	13..16	9..16
y (linea 12)	automatica	indef	13..16	9..16
var1 (linea14)	automatica	'C'	15..16	13..15
static aux (linea 17)	estatica	0	18..28	<1..28>
v2 (linea 18)	automatica	0	7 12..16 19..28	1..89
fun2 (linea 19)			20..28	19..23
fun3 (linea 24)			25..28	24..28
aux (linea 25)	automatica	indef	26..28	24..28