# Práctica Nro. 4 Variables

**Objetivo:** Conocer el manejo de identificadores en memoria y como lo definen e implementan los diferentes lenguajes.

**Ejercicio 1: a)** Tome una de las variables de la línea 3 del siguiente código e indique y defina cuales son sus atributos:

nombre: “a”  
alcance: 4-16  
tipo: integer  
tiempo de vida:1-16  
r-valor: indefinido  
l-valor: automatico

# Procedure Practica4();

1. **var**
2. a,i:integer
3. p:puntero

# Begin

1. a:=0;
2. new(p);
3. p:= ^i
4. for i:=1 to 9 do **10.**a:=a+i;  
   **11.**end;
5. ...
6. p:= ^a;
7. ...
8. dispose(p); **16.end**;

**b)** Compare los atributos de la variable del punto a) con los atributos de la variable de la línea 4. Que dato contiene esta variable?,

nombre: “p”  
alcance: p: 5-16 / p^:4-16   
tipo: puntero  
tiempo de vida: p: 1-16 / p^: 7-15  
r-valor: p:nil / p^: indefinido  
l-valor: p: automatico / p^: dinamico

# Ejercicio 2:

1. Indique cuales son las diferentes formas de inicializar una variable en el momento de la declaración de la misma.
   * Inicializacion en la declaracion: Las variables pueden inicicializarse en el momento de declararse con un valor. Ej: int num=5;
   * Inicializacion por defecto: Confiar en la inicializacion realizada por el lenguaje de programacion de acuerdo al tipo. Int en 0 o char en “” o punteros en nil
   * Ignorar el problema: La variable toma el valor que este alojado en memoria (la cadena de bits determinada. Puede llevar a errores y requiere chequeos adicionales.
2. Analice en los lenguajes: Java, C, Phyton y Ruby las diferentes formas de inicialización de variables que poseen. Realice un cuadro comparativo de esta característica.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inicializacion | Java | Python | C | Ruby |
| Al declarar | Int i=0; | i=0 | Int i=0; | i=0 |
| Por defecto | Int i; | No es posible | Variables globales y estaticas tienen un valor por defecto, las locales y resultan impredecibles sino se les asigna un valor | No es posible |
| Ignorar el problema | No es posible | No es posible | Se puede porque trabaja con direcciones de memoria | No es posible |

**Ejercicio 3:** Explique los siguientes conceptos asociados al atributo l-valor de una:

1. Variable estática.
2. Variable automática o semiestática.
3. Variable dinámica.
4. Variable semidinámica.

De al menos un ejemplo de cada uno.

Investigue sobre que tipos de variables respecto de su l-valor hay en los lenguajes C y Ada.

L-Valor: “Área de memoria ligada a la variable”, y su **tiempo de vida (lifetime)** es el **período de tiempo en que esa ligadura está activa**.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Definicion | Ejemplo C | Ejemplo Ada |
| Estatica | Su tiempo de vida abarca todo el programa se asigna al comienzo y se libera al final. Conserva su valor entre llamadas si es local | Static int x = 0; | No corresponde |
| automatica | Su tiempo de vida esta limitado al bloque donde se declara. Se reserva al comenzar y se destruye al salir | Int x = 10; | auto\_var integer :x= 10 |
| Dinamica | La memoria se reserva y libera **manualmente** durante la ejecución, usando instrucciones del lenguaje. **Tiempo de vida**: Desde la asignación dinámica (ej. new, malloc) hasta la liberación (ej. delete, free). | int \*p =malloc(sizeof(int));  \*p = 5;  free(p); | type Int\_Ptr is access Integer;  P : Int\_Ptr := new Integer'(10); |
| Semidinamica | La memoria se reserva durante la ejecución, **pero el identificador está ligado estáticamente**. Se usa, por ejemplo, para arreglos cuyo tamaño se conoce en tiempo de ejecución. Similar a automatica en TV pero con alocación dinámica interna | No corresponde | procedure Ejemplo is  type Vector is array (Positive range <>) of Integer;  V : Vector(1 .. N); -- N es dinámico, pero V tiene alcance estático |

# Ejercicio 4:

1. ¿A qué se denomina variable local y a qué se denomina variable global?

* **Variable local**:  
  Es una variable **declarada dentro de un bloque**, procedimiento o función.  
  → Su **alcance** está limitado a ese bloque y, por lo general, su **tiempo de vida (L-value)** coincide con la ejecución de ese bloque.
* **Variable global**:  
  Es una variable **declarada fuera de todas las funciones o procedimientos**, y su **alcance** se extiende a todo el programa o unidad de compilación.  
  → Su **L-value** suele ser estático, ya que su lugar en memoria se reserva al iniciar el programa y permanece hasta el final.

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

Si, seria local en su alcance pero se mantendria su valor entre llamados (estatico en l-valor). Por ejemplo en C:  
void contador() {

static int x = 0;

x++;

printf("%d\n", x);

}

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

**Sí, en la mayoría de los lenguajes tradicionales como C.** Esto se debe a que su **memoria se reserva antes de que empiece la ejecución** del programa, y **permanece** hasta que el programa termina.

"Una ligadura es estática si se establece antes de la ejecución y no se puede cambiar."  
Las variables globales están ligadas a una dirección fija durante toda la ejecución del programa.

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

Ambas pueden tener **ligadura estática de L-value**, pero lo que **distingue** a una constante es que su **r-value es inmutable**. En cambio, una variable estática puede ser modificada durante la ejecución.

# Ejercicio 5:

1. En Ada hay dos tipos de constantes, las numéricas y las comunes. Indique a que se debe dicha clasificación**.**En Ada, podemos definir constantes y en el momento de su declaración inconrporarles un tipo numerico, se lo conoce como ligadura temprana porque en ese punto ya tienen un tipo de dato asociado y se lo asigna el compilador. Por otro lado, tambien se puede declarar una constante con un valor numerico pero sin un tipo explicito. El compilador de Ada no le otroga un tipo, es en tiempo de ejecucion que recibe uno de acuerdo al contexto que se utilice y lo fija por el resto del programa, seria una ligadura tardia.
2. En base a lo respondido en el punto **a)**, 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 en su declaracion (linea 1), es un Float

I al utilizarla en la declaracion de K se asocia a un Float, antes era indefinido

K al momento de su declaracion (linea 3), es un Float

**Ejercicio 6:** Sea el siguiente archivo con funciones de C:

# Archivo.c

{ int x=1; **(1)**

**int func1**();{

int i;

for (i:=0; i < 4; i++) x=x+1;

}

**int func2**();{

int i, j;

/\*sentencias que contienen declaraciones y sentencias que no contienen declaraciones\*/

......

for (i:=0; i < 3; i++) j=func1 + 1;

}

}

Analice si llegaría a tener el mismo comportamiento en cuanto a alocación de memoria, sacar la declaración **(1)** y colocar dentro de **func1()** la declaración **static int x =1**;

En la version original, x es una variable global accesible desde caulquier punto del programa porque su scope o alcance va desde la linea 1 hasta el fin del programa y su l-valor es estatico.   
Si se redefine como una variable estatica local a la funcion func1 su l-valor se mantiene pero su alcance se ve limitado a la funcion. Solo func1 puede acceder a x. Sin embargo, por la ejecucion del programa su resultado final sigue siendo el mismo.

**Ejercicio 7:** Sea el siguiente segmento de código escrito en Java, indique para los identificadores si son globales o locales.

|  |  |
| --- | --- |
| **Clase Persona** {  public long id  public string nombreApellido public Domicilio domicilio private string dni;  public string fechaNac;  public static int cantTotalPersonas;  //Se tienen los getter y setter de cada una de las variables  //Este método calcula la edad de la persona a partir de la fecha de nacimiento | **public int getEdad()**{  public int edad=0;  public string fN = this.getFechaNac();  ...  ...  return edad;  }  }  **Clase Domicilio** {  public long id; public static int nro public string calle public Localidad loc;  //Se tienen los getter y setter de cada una de las variables  } |

**Globales :**

* public static int nro
* public static int cantTotalPersonas

Locales:

Todas las demas .\_.

**Ejercicio 8:** Sea el siguiente ejercicio escrito en Pascal

# Program Uno;

1. type tpuntero= ^integer;
2. var mipuntero: tpuntero;
3. var i:integer;
4. var h:integer;

# Begin

**7-** i:=3;

1. mipuntero:=nil;
2. new(mipuntero);
3. mipunterno^:=i;
4. h:= mipuntero^+i;
5. dispose(mipuntero);
6. write(h);
7. i:= h- mipuntero;
8. **End**.
   1. Indique el rango de instrucciones que representa el tiempo de vida de las variables i, h y mipuntero.  
      I=1-15

h= 1-15

mipuntero=1-15

mipuntero^ = 9-12

* 1. Indique el rango de instrucciones que representa el alcance de las variables i, h y mipuntero.

I=5-15

h= 6-15

mipuntero=4-15

mipuntero^ = 4-12

* 1. Indique si el programa anterior presenta un error al intentar escribir el valor de h. Justifique

No, el programa no presenta ningun error. El tipo de dato en todos los casos es integer y guarda en h el valor de sumar i + el valor guardado en la direccion que apunta mipuntero antes de realizar el dispose.

* 1. Indique si el programa anterior presenta un error al intentar asignar a i la resta de h con mipuntero.Justifique  
     En este caso si, intenta asignar el valor de restar h, que es integer, con el de un puntero en nil, lo cual no es una operación soportada
  2. 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.

Podriamos considerar el programa principal como “otra entidad”, con alcance global y tiempo de vida 1-15

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

I=automatica, integer

h= automatica, integer

mipuntero= automatica, tpuntero

mipuntero^ = dinamica, integer

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

1. En el cual el tiempo de vida de un identificador sea mayor que su alcance
2. En el cual el tiempo de vida de un identificador sea menor que su alcance
3. En el cual el tiempo de vida de un identificador sea igual que su alcance

**Ejercicio 10:** Si tengo la siguiente declaración al comienzo de un procedimiento: int c; **en C**

var c:integer; **en Pascal**

c: integer; **en ADA**

Y ese procedimiento NO contiene definiciones de procedimientos internos. ¿Puedo asegurar que el alcance y el tiempo de vida de la variable “c” es siempre todo el procedimiento en donde se encuentra definida?. Analícelo y justifique la respuesta, para todos los casos.

**Ejercicio 11: a)** Responda Verdadero o Falso para cada opción. El tipo de dato de una variable es?

1. Un string de caracteres que se usa para referenciar a la variable y operaciones que se pueden realizar sobre ella.
2. Conjunto de valores que puede tomar y un rango de instrucciones en el que se conoce el nombre.
3. Conjunto de valores que puede tomar y lugar de memoria asociado con la variable.
4. Conjunto de valores que puede tomar y conjunto de operaciones que se pueden realizar sobre esos valores.

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

**Ejercicio 12:** Sea el siguiente programa en ADA, completar el cuadro siguiente indicando para cada variable de que tipo es en cuanto al momento de ligadura de su l-valor, su r-valor al momento de alocación en memoria y para todos los identificadores cuál es su alcance y cual es su el tiempo de vida.

Indicar para cada variable su r-valor al momento de alocación en memoria

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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**     * 1. n:=4;       2. v2(n):= v2(1) + v1(5);       3. p:= new puntero;       4. q:= p;       5. .......       6. free p;       7. ......       8. free q;       9. ......   7.6. **end**;   1. **begin** 2. n:=5; 3. ..... 4. Uno; 5. a:= n + 2; 6. ..... 7. **end** |  | | | | | |
|  | **Ident.** | **Tipo** | **r-valor** | **Alcan**  **ce** | **T.V.** |
| a (linea  4) | automática | basura | 5-14 | 1-14 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | | | | | |

# Aclaración:

**Ident.**= Identificador / **Tipo** es el tipo de la variable respecto del l-value

**T.V.** = Tiempo de Vida / **r-valor** debe ser tomado al momento de la alocación en memoria. El alcance de los identificadores debe indicarse desde la línea siguiente a su declaración.

**Ejercicio 13:** El nombre de una variable puede condicionar:

1. Su tiempo de vida.
2. Su alcance.
3. Su r-valor.
4. Su tipo. Justifique la respuesta

**Ejercicio 14:** Sean los siguientes archivos en C, los cuales se compilan juntos

Indicar para cada variable de que tipo es en cuanto al momento de ligadura de su l-valor. Indicar para cada identificador cuál es su alcance y cual es su el tiempo de vida.

Indicar para cada variable su r-valor al momento de alocación en memoria

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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. }   1. **main**() 2. {static int var3; 3. extern int v2; 4. int v1, y; 5. for(y=0; y<10; y++) 6. { char var1='C'; 7. a=&v1;} 8. }   **ARCHIVO2.C**   1. static int aux; 2. int v2; 3. **static int fun2**( ) 4. { extern int v1; 5. aux=aux+1; 6. …   23. }   1. **int fun3**( ) 2. { int aux; 3. aux=aux+1; 4. …   28. } |  | | | | | |
|  | **Ident.** | **Tipo** | **r-valor** | **Alcance** | **T.V.** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | | | | | |

# Aclaración:

**Ident.**= Identificador

**T.V.** = Tiempo de Vida

**r-valor** debe ser tomado al momento de la alocación en memoria

El *alcance de los identificadores* debe indicarse desde la línea siguiente a su declaración.

**Ejercicio 15:** Para javascript investigue la diferencia semántica para declarar una variable utilizando los modificadores const, var, let y la ausencia de cualquiera de estos. Compárelo con un lenguaje de su preferencia.