📝 Practica 4

Siguiente

Anterior

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:
- 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?

```
1.Procedure Practica4();
2.var
3.  a,i:integer
4.  p:puntero
5.Begin
6.  a:=0;
7.  new(p);
8.  p:= ^i
9.  for i:=1 to 9 do
10.     a:=a+i;
11. end; //Este end esta mal
12. ... //Hagamos como que no existe
13. p:= ^a;
14. ...
15. dispose(p);
16.end;
```

Identificador	L-VALOR	R-VALOR	ALCANCE	T.VIDA
Practica4	-	-	1-16	1-16
a	automatica	basura	4-16	1-16
i	automatica	basura	4-16	1-16
p	automatica	basura	5-16	1-16
p^	dinamica	basura	5-16	7-15

Ejercicio 2

a) Indique cuales son las diferentes formas de inicializar una variable en el momento de la declaración de la misma.

Una variable puede inicializarse vacía o con un valor acorde al tipo de la variable. Internamente, si no le asigno un valor en memoria se guardará vacía hasta que la variable tome un valor

b) 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.

Lenguaje	Sintaxis de inicialización de variables
Java	Tipo variable = valor;
	Tipo variable;
	variable = valor;
	Tipo[] variable = new Tipo[tamaño];
C	Tipo variable = valor;
	Tipo variable;
	variable = valor;
	Tipo variable[tamaño];
Python	variable = valor
	variable = None
Ruby	variable = valor
	variable = nil

En los lenguajes Java y C, se pueden inicializar variables declarando y asignando un valor en la misma línea o declarando primero la variable y luego asignando un valor. Además, en Java es posible inicializar arreglos mediante la creación de una nueva instancia y especificando su tamaño. En Python y Ruby, las variables se pueden inicializar asignándoles un valor o dejándolas en estado nulo. En general, los cuatro lenguajes permiten la inicialización de variables de manera similar, aunque con algunas diferencias sintácticas. Python y Ruby tienen una sintaxis más simple, mientras que Java y C ofrecen más opciones y control en la inicialización de arreglos.

Ejercicio 3

- Explique los siguientes conceptos asociados al atributo l-valor de una:
- 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.

El atributo l-valor se refiere a la capacidad de una variable para ser utilizada como una referencia a la ubicación de memoria donde se almacena su valor. Los diferentes tipos de variables pueden tener diferentes atributos l-valor según la forma en que se declaran y utilizan en el programa.

Ejemplo	Definicion
`void myFunction() {` `  static int x = 0; // variable estática` `  x++; // incrementar valor de x` `  printf("Valor de x: %d\n", x);` `}`	Variable estática. Se declara con la palabra clave "static" y se almacena en una ubicación fija en la memoria durante toda la vida del programa.
`procedure myProcedure is` `  a : Integer := 10; -- variable automática` `begin` `  null;` `end myProcedure;`	Variable automática o semiestática. Se declara dentro de una función o un bloque y se almacena en la pila durante la ejecución de la función o el bloque.
`my_list = [1, 2, 3] # variable dinámica` `my_list.append(4)` `print(my_list) # [1, 2, 3, 4]`	Variable dinámica. Se crea y se destruye dinámicamente durante la ejecución del programa, y su ubicación en la memoria se determina en tiempo de ejecución.

``c void myFunction() { int n; printf("Ingrese tamaño de la matriz: "); scanf("%d", &n); int arr[n]; // variable semidinámica for (int i = 0; i < n; i++) { arr[i] = i; printf("%d ", arr[i]); } } ```	Variable semidinámica. Se declara como una matriz en tiempo de compilación pero se inicializa y cambia de tamaño en tiempo de ejecución.

Ejercicio 4

a) ¿A qué se denomina variable local y a qué se denomina variable global?

- `local` es aquella que tiene alcance dentro del contexto donde fue declarada
- `global` es una variable que se asigna en el programa principal y su alcance es de todo el programa.

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

- No, una variable local no puede ser estática respecto a su l-valor, ya que el término "estático" se refiere a una propiedad, no al l-valor.
- `static` significa que su valor se mantiene entre llamadas a una función, pero esto no afecta su l-valor.

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

No, una variable global no siempre es estática. La propiedad de ser estática o no depende de cómo se declare la variable

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

La principal diferencia entre una variable estática y una constante es que una variable estática se refiere a una ubicación de memoria que retiene su valor entre llamadas a una función o durante la ejecución del programa, mientras que una constante es un valor que no cambia durante la ejecución del programa. Una variable estática se inicializa solo una vez y se mantiene su valor durante toda la vida útil del programa. Su l-valor puede cambiar, pero su valor se mantiene constante. Por otro lado, una constante es una variable cuyo valor no cambia durante la ejecución del programa, y su l-valor tampoco.

Ejercicio 5

a) En Ada hay dos tipos de constantes, las numéricas y las comunes. Indique a que se debe dicha clasificación.

- Las constantes numéricas son aquellas que representan valores numéricos, como enteros, reales, complejos, etc. Estas constantes se definen utilizando una notación numérica estándar y la ligadura se produce durante la compilación; esto significa que los valores de las constantes numéricas se conocen antes de que el programa se ejecute y se incorporan directamente en el código objeto generado por el compilador.

b) 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; es una constante numérica con el valor 3.5 del tipo float. La ligadura se realiza en la compilación.
- `I` constant:= 2; es una constante numérica con el valor 2, sin tipo especificado así que se asume que es integer. La ligadura se realiza en la compilación.
- `K` constant float:= H*I; se define como una expresión que utiliza las constantes H e I. Dado que tanto H como I son constantes numéricas, su valor se evalúa durante la compilación. Por lo tanto, la constante K también se evalúa durante la compilación, en el primer momento de ligadura.

Ejercicio 6

Sea el siguiente archivo con funciones de C:

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;

```
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;
  }
}
```

Despues lo hago

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 //Global
  public string nombreApellido //Global
  public Domicilio domicilio //Global
  private string dni; //Local
  public string fechaNac; //Global
  public static int cantTotalPersonas; //Global
  //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
}
```

Los identificadores se pueden clasificar de la siguiente manera:

Globales (variables estáticas de clase, aquellos que se declaran a nivel de la clase y pueden ser accedidos sin necesidad de crear una instancia de la clase):

- cantTotalPersonas (estático) en la clase Persona
- nro (estático) en la clase Domicilio

Variables de instancia (no estáticas, son variables de instancia que se declaran dentro de la clase y solo pueden ser accedidas a través de una instancia de la clase):

- id en las clases Persona y Domicilio
- nombreApellido en la clase Persona
- domicilio en la clase Persona
- dni en la clase Persona
- fechaNac en la clase Persona
- calle en la clase Domicilio
- loc en la clase Domicilio

Locales (son aquellos que se declaran dentro de un método y solo pueden ser accedidos dentro del mismo):

- edad en el método getEdad() de la clase Persona
- fN en el método getEdad() de la clase Persona

La diferencia entre variables globales, variables de instancia y variables locales en Java se basa en su alcance y su duración. Las variables globales son visibles en todo el programa y tienen una duración de toda la vida del mismo, mientras que las variables de instancia son específicas de cada objeto y duran tanto como el objeto exista. Por último, las variables locales solo son visibles en el método o bloque de código en el que se declaran y tienen una duración limitada a la ejecución de ese método o bloque.

Ejercicio 8

Sea el siguiente ejercicio escrito en Pascal

- a) Indique el rango de instrucciones que representa el tiempo de vida de las variables i, h y mipuntero.
- b) Indique el rango de instrucciones que representa el alcance de las variables i, h y mipuntero.
- c) Indique si el programa anterior presenta un error al intentar escribir el valor de h. Justifique.
- d) Indique si el programa anterior presenta un error al intentar asignar a i la resta de h con mipuntero. Justifique
- 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.
- f) Especifique el tipo de variable de acuerdo a la ligadura con el l-valor de las variables que encontró en el ejercicio.

```
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-  mipunterno^:=i;
11-  h:= mipuntero^+i;
12-  dispose(mipuntero);
13-  write(h);
14-  i:= h- mipuntero;
15- End.
```

Identificador	L-VALOR	ALCANCE	T.VIDA
minipuntero	automatico	4-15	1-15
minipuntero^	dinamico	4-15	9-12
i	automatico	5-15	1-15
h	automatico	6-15	9-12

Ejercicio 9

Elija un lenguaje y escriba un ejemplo:

- a) En el cual el tiempo de vida de un identificador sea mayor que su alcance
- b) En el cual el tiempo de vida de un identificador sea menor que su alcance
- c) En el cual el tiempo de vida de un identificador sea igual que su alcance

```
1. static int aux;
2. int v2;
3. static int fun2( )
4. { extern int v1;
5.    aux=aux+1;
6. }
7. int fun3( )
8. { int aux;
9.    aux=aux+1;
10. }
```

Identificador	Lvalor	Rvalor	Alcance	T. vida
aux	estática	0	1-8	<1-10>
v2	automática	0	3-10	1-10
fun2			4-10	3-6
v1	automática	basura	5-6	3-6
fun3			8-10	7-10
aux	automática	basura	9-10	7-10

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?

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

```
01.with text_io; use text_io;
02.
03.procedure Main is
04.   type vector is array (integer range <>) of integer;
05.   a, n, p: integer;
06.   v1: vector(1..100);
07.   c1: constant integer := 10;
08.
09.   procedure Uno is
10.      type puntero is access integer;
11.      v2: vector(0..n);
12.      c2: character;
13.      p, q: puntero;
14.   begin
15.      n := 4;
16.      v2(n) := v2(1) + v1(5);
17.      p := new puntero;
18.      q := p;
19.      -- ...
20.      free p;
21.      -- ...
22.      free q;
23.      -- ...
24.   end Uno;
25.
26.begin
27.   n := 5;
28.   -- ...
29.   Uno;
30.   a := n + 2;
31.   -- ...
32.end Main;
```

Identificador	Lvalor	Rvalor	Alcance	T. vida
a	automatica	basura	6-32	3-32
n	automática	basura	6-32	3-32
p	automatica	basura	6-13	3-32
v1	automática	basura	7-32	3-32
c1	constante	basura	8-10	7-10
v1				
c2				
p				
q				
p^				
q^				

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 memori

---

```
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	Lvalor	Rvalor	Alcance	T. vida
v1				
a*				
fun2				
v1				
y				
v2				
var3				
v2				
v1				
y				
var1				
aux				
v2				
fun2				
v1				
fun3				
aux				