Práctica 4

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

```
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;
12....
13.p:= ^a;
14....
15.dispose(p);
16.end;
```

Atributos de la variable a.

Nombre: a

Alcance: Estático, de la línea 3 hasta línea 16

Tipo: Predefinido Integer, ligadura estática.

L-Valor: Alocacion dinámica automática.

R-Valor: Dinámico 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

b) Compare los atributos de la variable del punto a) con los atributos de la variable de la línea 4. ¿Qué dato contiene esta variable?

Atributos de la variable p.

Nombre: p

Alcance: Estático, de la línea 4 hasta línea 16

Tipo: Tipos definidos por el usuario Puntero, ligadura estática.

L-Valor: Alocación dinámica explícita.

R-Valor: Dinámico l-valor de i (línea 8), l-valor de a (línea 13).

Ejercicio 2:

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

- Ignorar el problema: la inicializo con lo que haya en memoria
- Estrategia de inicialización:
    - Inicialización por defecto: Enteros se inicializan en 0, los caracteres en blanco, etc.
    - Inicialización en la declaración: C int i =0, j= 1

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.

Java	- Declaración e inicialización en la misma línea: int num = 10; - Declaración e inicialización separadas: int num; num = 10; - Inicialización por defecto: int num; (inicializada a 0 automáticamente)
C	- Declaración e inicialización en la misma línea: int num = 10; - Declaración e inicialización separadas: int num; num = 10; - No hay inicialización por defecto, el valor inicial es indeterminado
Python	- Declaración e inicialización en la misma línea: num = 10 - No es necesario declarar el tipo de variable antes de su inicialización
Ruby	- Declaración e inicialización en la misma línea: num = 10 - No es necesario declarar el tipo de variable antes de su inicialización

En resumen, Java y C tienen formas similares de inicialización de variables, mientras que Python y Ruby tienen una sintaxis más flexible y sencilla. En Python y Ruby no es necesario declarar el tipo de variable antes de su inicialización, lo que puede hacer que el código sea más fácil de escribir y entender. Por otro lado, Java y C no permiten la asignación de variables sin declarar su tipo previamente. Además, Java tiene la opción de inicialización por defecto, que es útil en ciertos casos, como cuando se desea inicializar un array.

Ejercicio 3: Explique (de al menos un ejemplo de cada uno) los siguientes conceptos asociados al atributo l-valor de una:

El atributo l-valor de una variable hace referencia a su capacidad de ser referenciada o manipulada mediante una expresión de asignación. En otras palabras, una variable con l-valor puede ser asignada a otra variable o valor.

a. Variable estática.

Una variable estática tiene un ámbito de visibilidad limitado a la función o archivo en el que se declara y su valor se mantiene constante durante toda la ejecución del programa. Es decir, es una variable que existe y se inicializa en tiempo de compilación y su valor persiste en memoria durante toda la ejecución del programa.

Ejemplo en C:

```c
#include <stdio.h>

void static_example() {
  static int num = 0;
  printf("El valor de num es: %d\n", num);
  num++;
}

int main() {
  for (int i = 0; i < 5; i++) {
    static_example();
  }
  return 0;
}
```

En este ejemplo, la variable num es una variable estática dentro de la función static_example().

La salida del programa sería:

```
El valor de num es: 0
El valor de num es: 1
El valor de num es: 2
El valor de num es: 3
El valor de num es: 4
```

b. Variable automática o semiestática.

Una variable automática o semiestática se declara dentro de un bloque y su ámbito de visibilidad se limita a ese bloque. Su valor se inicializa al entrar al bloque y se destruye al salir de él. Estas variables también se llaman locales, ya que solo son visibles dentro de la función en la que se declaran.

Ejemplo en Python:

```python
def automatic_example():
  num = 0
  print("El valor de num es: ", num)
  num += 1

for i in range(5):
  automatic_example()
```

En este ejemplo, la variable num es una variable automática dentro de la función automatic_example().

La salida del programa sería:

```
El valor de num es: 0
El valor de num es: 0
El valor de num es: 0
El valor de num es: 0
El valor de num es: 0
```

c. Variable dinámica.

Una variable dinámica es una variable que se crea y se destruye en tiempo de ejecución. En la mayoría de los lenguajes de programación, estas variables se crean utilizando funciones o métodos específicos como malloc() o new. El valor de una variable dinámica puede cambiar a lo largo de la ejecución del programa.

Ejemplo en Python:

```python
def dynamic_example():
  num = input("Ingrese un número: ")
  print("El valor ingresado es:", num)

for i in range(5):
  dynamic_example()
```

En este ejemplo, la variable num es una variable dinámica que se inicializa mediante la entrada de usuario en cada llamada a la función dynamic_example().

La salida del programa dependerá de lo que ingrese el usuario.

d. Variable semidinámica.

Una variable semidinámica es una variable que se inicializa en tiempo de ejecución pero que no cambia de tamaño después de ser creada.

```python
def semidynamic_example():
    num = [1, 2, 3]
    print("El valor de num es:", num)
    num[1] = 5
    print("El valor modificado de num es:", num)

semidynamic_example()
```

En este ejemplo, la variable num es una variable semidinámica, ya que se crea en tiempo de ejecución como una lista con tres elementos y su tamaño no cambia después de ser creada. Sin embargo, su valor puede ser modificado, como se ve en la línea num[1] = 5.

La salida del programa sería:

```
El valor de num es: [1, 2, 3]
El valor modificado de num es: [1, 5, 3]
```

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

En el lenguaje de programación C, se pueden clasificar las variables según su l-valor de la siguiente manera:

a. Variable estática: se define usando la palabra clave "static" y se mantiene en memoria durante toda la ejecución del programa. Estas variables se inicializan automáticamente a cero si no se les asigna un valor explícitamente.

b. Variable automática o semiestática: se definen dentro de una función y se eliminan de la memoria cuando la función termina. Estas variables no se inicializan automáticamente, por lo que su valor es indeterminado hasta que se les asigna uno.

c. Variable dinámica: se asignan en tiempo de ejecución usando funciones como malloc() y calloc(). Estas variables se mantienen en la memoria hasta que se libera explícitamente usando la función free().

d. Variable semidinámica: se crean usando un tamaño fijo en tiempo de compilación y su valor puede cambiar en tiempo de ejecución. Por ejemplo, un arreglo puede ser una variable semidinámica.

En el lenguaje de programación Ada, las variables se clasifican según su l-valor de la siguiente manera:

a. Variable estática: se definen usando la palabra clave "constant" o "static" y se mantiene en memoria durante toda la ejecución del programa.

b. Variable automática o semiestática: se definen dentro de un bloque y se eliminan de la memoria cuando el bloque termina.

c. Variable dinámica: se asignan en tiempo de ejecución usando la palabra clave "new". Estas variables se mantienen en la memoria hasta que se libera explícitamente usando la palabra clave "delete".

d. Variable semidinámica: no hay una clasificación explícita para las variables semidinámicas en Ada, pero se pueden crear utilizando arreglos de tamaño fijo. El tamaño del arreglo es fijo en tiempo de compilación, pero el valor de sus elementos puede cambiar en tiempo de ejecución.

Ejercicio 4:

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

Una variable local es una variable declarada dentro de una función o bloque de código y su ámbito de alcance se limita a esa función o bloque de código. Por otro lado, una variable global es una variable declarada fuera de cualquier función o bloque de código y su ámbito de alcance se extiende a todo el 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 a su l-valor. En algunos lenguajes de programación, como C y C++, se pueden definir variables locales como estáticas para mantener su valor entre llamadas sucesivas a una función.

Un ejemplo en C podría ser:

```c
#include <stdio.h>

void ejemploFuncion() {
  static int variableLocal = 0;
  variableLocal++;
  printf("El valor de la variable estática es %d\n", variableLocal);
}

int main() {
  ejemploFuncion(); // Imprime "El valor de la variable estática es 1"
  ejemploFuncion(); // Imprime "El valor de la variable estática es 2"
  ejemploFuncion(); // Imprime "El valor de la variable estática es 3"
  return 0;
}
```

En este ejemplo, la variable variableLocal se declara como static dentro de la función ejemploFuncion(). Esto significa que su valor se mantiene entre llamadas sucesivas a la función. En cada llamada a la función, el valor de variableLocal se incrementa en uno y se imprime en pantalla. Como la variable es estática, su valor se mantiene entre llamadas y no se pierde cuando la función termina su ejecución.

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

No necesariamente. La estática es una propiedad que puede tener una variable independientemente de su alcance. En algunos lenguajes de programación, como C, las variables globales son, por defecto, estáticas en términos de su l-valor, lo que significa que su valor se mantiene durante toda la vida útil del programa y no puede ser modificado por otras funciones o bloques de código. Sin embargo, en otros lenguajes de programación como Java y Python, las variables globales no son estáticas en términos de su l-valor, ya que pueden ser modificadas en tiempo de ejecución y su valor puede ser accedido y modificado por cualquier función o bloque de código en el programa.

En resumen, la naturaleza de las variables globales respecto a su l-valor no depende del hecho de que sean globales o no, sino que depende del lenguaje de programación utilizado y de cómo se manejan las variables en ese lenguaje. Por lo tanto, es importante conocer las características del lenguaje de programación que se está utilizando para comprender el comportamiento de las variables globales en términos de su l-valor.

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

La diferencia principal entre una variable estática respecto a su l-valor y una constante es que el valor de una constante no puede ser modificado una vez que se ha definido, mientras que el valor de una variable estática puede ser modificado durante la ejecución del programa.

Una constante se define en el código del programa y su valor es fijo durante toda la ejecución. Es decir, el valor de una constante se define una sola vez y no se puede modificar en tiempo de ejecución. Las constantes se usan para representar valores fijos como pi, el número de días en una semana, entre otros.

Por otro lado, una variable estática se define en el código del programa y su valor se mantiene en memoria durante toda la ejecución del programa, pero el valor de la variable se puede modificar en cualquier momento. Las variables estáticas se utilizan para mantener un valor a lo largo de la vida útil del programa, como contar el número de veces que se llama a una función.

En resumen, la diferencia fundamental entre una variable estática y una constante es que el valor de una constante no puede ser modificado después de ser definido, mientras que el valor de una variable estática puede ser modificado en tiempo de ejecución.

Ejercicio 5:

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

La clasificación de las constantes en Ada en numéricas y comunes se debe a que las constantes numéricas son aquellas que se refieren a valores numéricos, mientras que las constantes comunes son aquellas que pueden contener caracteres y símbolos, como cadenas de texto.

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;

I: constant:= 2;

K: constant float:= H*I;

El momento de ligadura de las constantes en Ada es durante la compilación del programa, es decir, en tiempo de compilación. En el código proporcionado, el momento de ligadura de la constante "H" es cuando se le asigna el valor 3.5, el momento de ligadura de la constante "I" es cuando se le asigna el valor 2, y el momento de ligadura de la constante "K" es cuando se calcula su valor mediante la multiplicación de "H" e "I". Por lo tanto, el momento de ligadura de "K" es después del momento de ligadura de "H" e "I" durante el proceso de compilación. Como todas las constantes se definen en tiempo de compilación, su valor se conoce antes de que se ejecute el programa.