Práctica 4 - CPLP

1) a) <nombre, alcance, tipo, L-value, r-value> => <a, local, integer, estática, 0>

b) línea 4 p => <p, local, puntero, dinámica, in> el dato que contiene esta variable es una dirección de memoria.

2) Hay varias formas de inicializar una variable en el momento de la inicialización

- Asignar directa: se asigna un valor a la variable en la misma línea de declaración,
  ej: int edad: 25;
- Asignar x defecto: se declara una variable sin asignar un valor entonces se inicializa con un valor por defecto que depende del tipo de dato.
  ej: int edad,
- Inicializar a través de una expresión: la variable se inicializa usando una expresión que puede incluir operaciones aritméticas, llamadas a f×, etc.

  int edad = 2023 - 1998.
- Inicializar por referencia: se inicializa una variable con la dirección de memoria de otra variable.

  ej: int *b = &a;
- Inicializar mediante un constructor: se usa un constructor para crear y asignar un valor inicial a una variable

b) Inicializax en Java, C, Python, Ruby

Lenguaje	Inicializax x valor	Inicializa x referencia	Inicializax x constructor
Java	Sí	Sí	Sí
C	Sí	Sí c/punteros	No
Python	Sí	Sí c/punteros	No
Ruby	Sí	Sí	Sí p/objetos.

3) a) variable estática: variable declarada dentro de una fx pero que se mantiene en memoria durante toda la ejecux del programa. En términos de l-value, una variable estática tiene un alcance global en la fx donde se declara.

b) variable automática/semiestática: variable declarada dentro de una fx y se elimina automáticamente al salir de la fx. En términos de l-value, una variable automática es la direx donde se encuentra almacenada en la pila de ejecux.

c) variable dinámica: variable reservada en tiempo de ejecux. En términos de l-value, una var. dinámica tiene un alcance global pero su direx puede cambiar durante la ejecux del programa. El l-value de una var dinámica es la direx de memoria donde se encuentra almacenado pero puede cambiar a medida que el programa se ejecuta.

d) variable semidinámica: variable reservada en tiempo de ejecux compilax o de ejecux, usando fx especiales pero que se elimina automáticamente al salir de la fx. La var. semidinámica tiene un alcance local dentro de donde se declara.

C clasifica sus variables en 2 tipos respecto a su l-value

1. Variables l-value: variable que tiene una ubicación de memoria asociada a ellas y por lo tanto, puede ser asignadas y modificadas. Ej: globales, locales y miembros de estructuras

- Expresiones r-value: expresiones q' no tienen ubicax de memoria asociada a ellas y/x lo tanto, no pueden ser asignadas. Ej: literales, constantes o rsdo de exp. aritmética

ADA

- Variables de objetos: variables q' tienen una ubicación de memoria asociada a ellas y pueden ser asignadas y modificadas. Como arreglos o registros

- Parámetros de entrada: variables que se pasan como argumentos a una subrutina y no se pueden modificar

- Parámetros de salida: variables pasadas como argumentos a una subrutina y se pueden modificar dentro de la subrutina.

- Parámetros de E/S: variables que se pasan como argumentos a una subrutina y se pueden modificar y leer dentro de la subrutina.

Ejercicio 4:

a) Variable local: definida dentro de una fx o bloque y solo puede accederse/usarse en ese ámbito.
Variable global: se define fuera de una fx o bloque e igual puede ser usada

b) Sí

c) Una variable global no siempre es estática. porque puede venir de otro archivo/módulo ∴ su l-value se establece en binding-time.

d) Una variable estática respecto de su l-value es una variable cuyo espacio de almacenamiento es reservado durante la compilar y su valor persiste a lo largo de la vida del programa. X otro lado, una constante es un valor que no cambia durante la ejecuc del programa y se define en tiempo de compilar.

Ejercicio 5:

a) ADA clasifica las constates en constantes numéricas (p/ cálculos numéricos, binding en compilar) y comunes. (char, strings.etc, binding en ejecuc.)

b) I : constant Float := 3.5; constantes numéricas, en compilar
I : constant := 4;
K : constant Float := 'A'; const. común en ejecuc.

Ejercicio 6: tiene el mismo comportamiento ya que 'static int x=1' la memoria se asigna en el 1er llamado a la fx y se mantiene, igual que 'int x=1' como variable global. Entonces con respecto a la alocación en memoria tienen igual comportamiento

Ejercicio 7:

- Globales: son visibles en toda la clase y tienen una duración de toda la vida de la app.
- Locales: solo visibles en el método y duran la ejecuc de ese método.
- Instancia: específicas de c/ objeto y dura lo que dure ese objeto.

Globales: public static int cantTotalPersonas, public static int nro.

Locales: public int edad, public string fN

Instancia: long id, string NombreApellido, Domicilio domicilio, string dni, string fechaNac, string calle, Localidad loc

Ejercicio 8: dispose = libera memoria asignada dinámica

a) Variable i = línea 4 → línea 15 / Variable h = línea 5 → línea 15 / Variable miPuntero = línea 3 →

b) Variable i = línea 4 → línea 15 / Variable h = línea 5 → 15 / Variable miPuntero = línea 3 → 15

c) Sí, habría un error al escribir h en línea 13 porque en línea 12 el valor de miPuntero fue liberado en memoria

d) Si, habia error porque el valor de miPuntero es nil porque se efectuó un dispose(miPuntero) error en tiempo de ejecux

e) El otro atributo a ligar es el programa ppal.

f) i y h tipo entero, mipuntero tipo tpuntero, tpuntero puntero a un entero.

Ejercicio 9: ~~C~~ Pascal

Identificador c/ tiempo de vida mayor que su alcance

```
var
    p: ^integer;
begin
    new(p);
    p^ := 10;
end.
```

→ finaliza pero no se libera el espacio

Identificador c/ tiempo de vida menor a su alcance.
no hay

Identificador c/ tiempo de vida igual que su alcance

```
program ej3;
var
    x := 10;
begin
    i := 1 + x;
end.
```

→ var global que dura todo el programa.

Ejercicio 10.
Si, en todo procedimiento el cual no tenga procedimientos internos, el alcance y vida es igual

Ejercicio 11: El tipo de dato de una variable es IV) el conjunto de valores que puede tomar y conjunto de operaciones que se pueden realizar sobre esas valores.

El tipo de dato de una variable indica los conjuntos de valores y los operaciones validas. Antes de que una variable pueda ser referenciada debe ligarse con tipo p/ prohibir de operaciones no permitidas y chequeo de tipo para verificar el uso correcto de variables

automático. Ada deduce tipo de atributo.

(1)

Ejercicio 12:

```
with text_io; use text_io;
Procedure main is;
type vector is array (integer range <>);
    a,n,p: integer;
    v1: vector(1..100);
    c1: constant integer:=10;

  Procedure Uno is;
type puntero is access integer
  v2: vector (0..n);
  c1,c2: character;
  p,q: puntero;
  begin
      n:=4;
      v2(n):= v2(1)+v1(5);
      p:= new puntero;
      q:= p;
      free p;
      free q;
end;
begin
  n:=5;

  Uno;
  a:= n+2;
end.
```

Ident	Tipo	r-valor	Alcance	T. vida
a	autovariable	///	5-14	1-14
n	"	///	5-14	1-14
p	"	///	5-14	1-14
v1		0	6-14	1-14
c1		10	7-14	1-14
v2		0.	3-7.6	3-7.6.
c1, c2		posic 0 de ASCII	4-7.6	3-7.6
p, q		null	5-7.6	5-7.6

Ejercicio 13

b) Su alcance.

El nombre de una variable se refiere al identificador que se utiliza p/ hacer referencia a ello en el programa. Este identificador define el alcance de la variable, es decir, el ámbito en el cual la variable puede ser utilizada. El alcance de una variable está determinado por el bloque donde se declara.
Entonces, el nombre condiciona el alcance pero no su t.v, r-valor o tipo.

Ejercicio 14: hacer

Ejercicio 15:

const: declara variables de solo lectura y siempre debe acompañarse por un valor

var: declara variable local o global en una fx y tiene un alcance más amplio. Las var pueden ser reasignadas y su valor persistente en todo el ámbito de la fx donde se declaran. Si se declaran sin inicializar, su valor se establece como undefined.

let: declara variable local en un bloque de código y su alcance se limita al bloque donde se declara. Las variables declaradas con ~~var~~ let pueden ser reasignadas dentro del bloque donde se declaran. Si se declaran sin ser inicializados, se establece undefined. Si se modifica en otro lado será igual.

ausencia de modificador: si no tiene const, var o let, es una variable global.