Consideraciones segun el lenguaje

Consideraciones a tener en cuenta:

Tipos de variables y como trabajan:

Tipo	Qué es	Vida útil (duración)	Ejemplos y comportamiento
Estática	Se reserva una sola vez en memoria	Desde que el programa inicia hasta que termina	**C**: `static int x;` dentro de funciones o global **Java**: variables de clase (con `static`)
Automática	Se crean al entrar en un bloque (función, procedimiento) y se destruyen al salir	Mientras dure el bloque/función	**C/Pascal**: variables locales **Ada**: variables locales sin `new`
Dinámica	Se reserva memoria manualmente con `new` o similar	Mientras el programador no la libere	**C**: `malloc`, `free` **Pascal**: `new(p)` y `dispose(p)` **Java**: `new Clase()` (aunque el garbage collector lo maneja)
Semidinámica	Tipo especial donde una parte es dinámica (por ej. límites de un arreglo), pero no se usa `new` directamente	Al entrar al bloque, pero su forma depende de una constante o parámetro	**Ada**: arreglos definidos por parámetros de entrada o constantes (`(1..L)`)

En C:

Consideraciones sobre variables globales y estaticas:

- Las variables globales o estaticas (numericas) no inicializadas tienen R-Valor 0
- Las variables locales su R-Valor puede ser:
  - Si no esta inicializada: basura

```
int a; // basura
```

- Si esta inicializada/definida: valor de inicializacion/ definicion

```
int i=4; // R-Valor = 4
```

Teniendo en cuenta que son variables numericas:

- Las variables globales se inicializan con R-Valor 0
  - Tienen tiempo de vida desde el momento 1 hasta que termina todo el codigo
- Las variables estaticas se inicializan con R-Valor 0
  - Se alocan al momento de alocar el codigo. Es decir, su tiempo de vida seria desde el instante 1 hasta el ultimo instante del codigo.
    - TIP IMPORTANTE: EL TIEMPO DE VIDA DE UNA VARIABLE ESTATICA DEBE ESTAR ENTRE <>, EJ: <1..N>

Conclusion: Las variables globales y variables estaticas trabajan de la misma manera en cuanto a su tiempo de vida:

- Su tiempo de vida iria entre 1 hasta que se termina el codigo.
  con respecto a su R-Valor:
- Si no se inicializa al momento de la declaracion: su R-Valor es 0
- Si se inicializa al momento de la declaracion: su R-Valor seria el valor inicialiazdo.

Variables locales:

- Las variables locales son automaticas y tienen R-Value basura.
- Las funciones (funciones estaticas y funciones no estaticas) se comportan de la misma forma ? (Consultar)
  - Su alcance : Es desde una linea siguiente a su declaracion, hasta que finaliza el bloque que lo contiene
  - Su tiempo de vida: es desde que se declara la funcion, hasta que termina su declaracion

Punteros:

Son estaticos. Su R-Value es Null.

Resumen de C:

Lenguaje	Tipo de Variable	L-Valor	R-Valor	Notas a considerar
C	Global numérica	automática	Toma valor por defecto (0)	Se inicializa si es global o estática, no si es local
	Global puntero	automática	Null por defecto	Igual que arriba, punteros globales se inicializan
	Global estática numérica	estático	Toma valor por defecto (0)	
	Local estática numérica	automática	Toma valor por defecto	Solo en variables estáticas
	Local (tipo: `int i=4`)	automática	Definido (4)	
	Puntero local sin inicializar	automática	Indefinido	Peligroso usar sin inicializar
	Local tipo: int v1	automatica	0	

En ADA

En ADA tenemos tipos automatico o semi-dinamico

Todo es automatico, excepto algunas declaraciones de los vectores.

Tipo automatico (Se conoce el valor en compilacion)

```
type Vector is array (1 .. 10) of Integer;  -- tamaño fijo
 V : Vector := (others => 0);  -- inicializado con ceros
```

Tipo semi-dinamico (Se conoce el valor en ejecucion)

```
N : constant Integer := Integer'Value(Argument(1));  -- valor en ejecución
type Vector is array (Positive range <>) of Integer;
V : Vector(1 .. N);  -- semidinámico: tamaño solo se conoce en ejecución
```

Resumen de ADA:

Lenguaje	Tipo de Variable	L-Valor	R-Valor	Notas a considerar
ADA	Constante numérica (`c : constant integer := 10`)	automática	10 (definido)	Es constante común
	Constante sin `:=` (`c : constant := 10`)	automática	Definido al compilar	
	Arreglo tipo `(1..10)`	automática	Indefinido	
	Arreglo `(L..C)` definido con constante	semi-dinámica	Indefinido	
	Arreglo de tipo constante	automática	Indefinido	
	Puntero	automática	Null	ADA parece siempre inicializar en Null los punteros

Pascal:

Las variables globales son sonsideradas de tipo Estaticas. Toman R-Valor 0 si no estan inicializadas por defecto.

Variables locales son automaticas:

- integer,real,string,char: su R-Value es indefinido si no se inicializan por defecto.
- de tipo puntero: Su R-Value es Nill

El tiempo de vida de lo que apunte el puntero por ejemplo p^, deberia ir desde que se hace un new, hasta que se hace un dispose. Y el alcance es una linea siguiente a la que se declara, hasta fin del bloque que contiene la variable de tipo puntero.

Lenguaje	Tipo de Variable	L-Valor	R-Valor	Notas a considerar
Pascal	Variable global (`var x: integer`)	estático	Toma valor por defecto (0)	Variables globales sí se inicializan por defecto
	Variable local (`var x: integer`)	automática	Indefinido	Si no se inicializa, el valor es basura
	Puntero (`p: ^integer`)	automática	Indefinido o Nil	No siempre se inicializa, depende del compilador

Lenguaje	Tipo de Variable	L-Valor	R-Valor	Notas a considerar
				(FreePascal lo hace a veces)
	Contenido de puntero (`p^`)	dinámico	Indefinido si no se usó `new(p)`	