Teoría

Pseudocódigo: Estructura de datos

Resolución de Problemas y Algoritmos
Fundamentos de la Programación

Ingeniería en Computación (TU y TFA)
Ingeniería en Minas (TU)
Profesorado en Ciencias de la Computación (TU y TFA)

Teoría

Pseudocódigo: Estructura de datos

- ✓ **Estructura de datos: Arreglo lineal.**
  - Declaración/definir.
  - Operaciones con arreglos: Asignación y recuperación de valores.
- ✓ **Estructuras de control: secuencial, condicional y de repetición (PARA).**
- ✓ **Pseudocódigo con PSeInt.**

Representación de datos de entrada y salida.

Los principales **tipos de datos primitivos o simples** son:

- Numéricos: Enteros y Reales.
- Caracteres.
- Lógicos o Booleanos.

Entero	Real	Lógico	Caracter
			
Números completos sin decimales.	Números con punto decimal.	Valores booleanos de verdad.	Un solo símbolo de texto entre comillas simples.
128 -5	3.14 -0.5	VERDADERO FALSO	'A' '?'

Tipos de datos primitivos

ENTERO: consiste de un conjunto finito de los números enteros positivos y negativos. Ejemplos: 3, -3, 345, -56.

REAL: consiste de un conjunto finito de valores de los números reales. Números con **punto decimal**, positivos y negativos. Ejemplos: 2.3, -4.6.

LÓGICO: también llamado tipo **booleano**, es el conjunto de los valores de verdad: VERDADERO y FALSO.

CARACTER: es el conjunto finito y "ordenado" de caracteres que el procesador puede reconocer. Ejemplos: 'A', '3', '+'.

las letras mayúsculas del abecedario.
las letras minúsculas del abecedario.
los caracteres numéricos del 0...9.
el caracter de espacio blanco, caracteres especiales tales como: *, +, -, _, /, (,), $, ^, %, $, < ,>, ", .

¿Dónde se guardan los datos de entrada y salida?

Una **variable** es un espacio, en la memoria de la computadora, reservado para guardar un dato.

Se llama "variable" porque el dato que guardamos adentro puede cambiar a lo largo del tiempo, pero el espacio siempre es el mismo.

Para que esta variable funcione bien, necesita dos cosas:

- un **Nombre**: Para poder encontrar el dato rápidamente sin buscar en otros lados.
- un **Tipo**: Para saber qué "clase" de dato entra ahí (por ejemplo, si el espacio es para guardar un número o para guardar una palabra).

Toda **variable** debe definirse indicando el **nombre** y el **tipo** de valores que la misma puede tomar.

Definir <nombre de variable> [,<nombre de variable>]* **como** <tipo>

Ejemplos:

Definir NUMERO **como** Entero

Definir numero **como** Real

Definir Peso, Suma **como** Real

```
//Identificador de variable

No puede tener espacios;
No puede empezar por un numero;
No puede ser una palabra
reservada;
Distingue mayusculas
```

DIFERENTES TIPOS DE DATOS Y SUS REPRESENTACIONES

TIPOS DE DATOS

DATOS NUMÉRICOS

- **Enteros** (e.g., 5, -12)
- **Flotantes** (e.g., 3.14, -0.01)

DATOS DE TEXTO (ALFANUMÉRICOS)

- **Cadenas** (e.g., 'Hola', 'Ecuación')
- **Caracteres** (e.g., 'a', '@')

DATOS LÓGICOS (BOOLEANOS)

- **Verdadero / Falso** (e.g., Sí/No)
- **Estados binarios**

DATOS ESTRUCTURADOS (COMPLEJOS)

- **Listas/Arreglos** (e.g., [1,2,3])
- **Registros/Objetos** (e.g., (Nombre: 'Ana', Edad: 25))

REPRESENTACIONES DE DATOS

REPRESENTACIÓN NUMÉRICA

REPRESENTACIÓN DE TEXTO

REPRESENTACIÓN LÓGICA

REPRESENTACIÓN ESTRUCTURADA

Tipos de datos.

Tipos de datos

- Primitivos o simples
  - Numéricos
    - Enteros
    - Reales
  - Caracteres
  - Lógicos
- Estructurados
  - Internos
    - Estáticos → Arreglos
    - Dinámicos → Listas, Pilas, Colas, Árboles
  - Externos → Archivos

Estructuras de datos.

Una **estructura de datos** es una forma particular de organizar datos en una computadora para que pueda ser utilizado de manera eficiente.

Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.

Las estructuras de datos son un medio para manejar grandes cantidades de datos de manera eficiente

Una **estructura de datos** es un conjunto de datos reunidos bajo un **único nombre** colectivo.

Se tiene que guardar las notas de 15 estudiantes de la materia para luego calcular su promedio.

¿cómo lo resolverías en PseInt con lo que sabemos hasta ahora?

Crear 15 variables diferentes (`nota1`, `nota2`, ...,`nota15`) y en el programa tendría que:

Definir nota1, nota2, nota3, nota4, nota5, nota6, nota7, nota8, nota9, nota10, nota11, nota12, nota13, nota14, nota15 **Como Real**

¿Y si ahora quiero sacar el promedio de todos los estudiantes que están cursando ahora?

Un cliente tiene 100 sucursales. El cliente necesita guardar la venta mensual de cada una para luego calcular e informar el promedio.

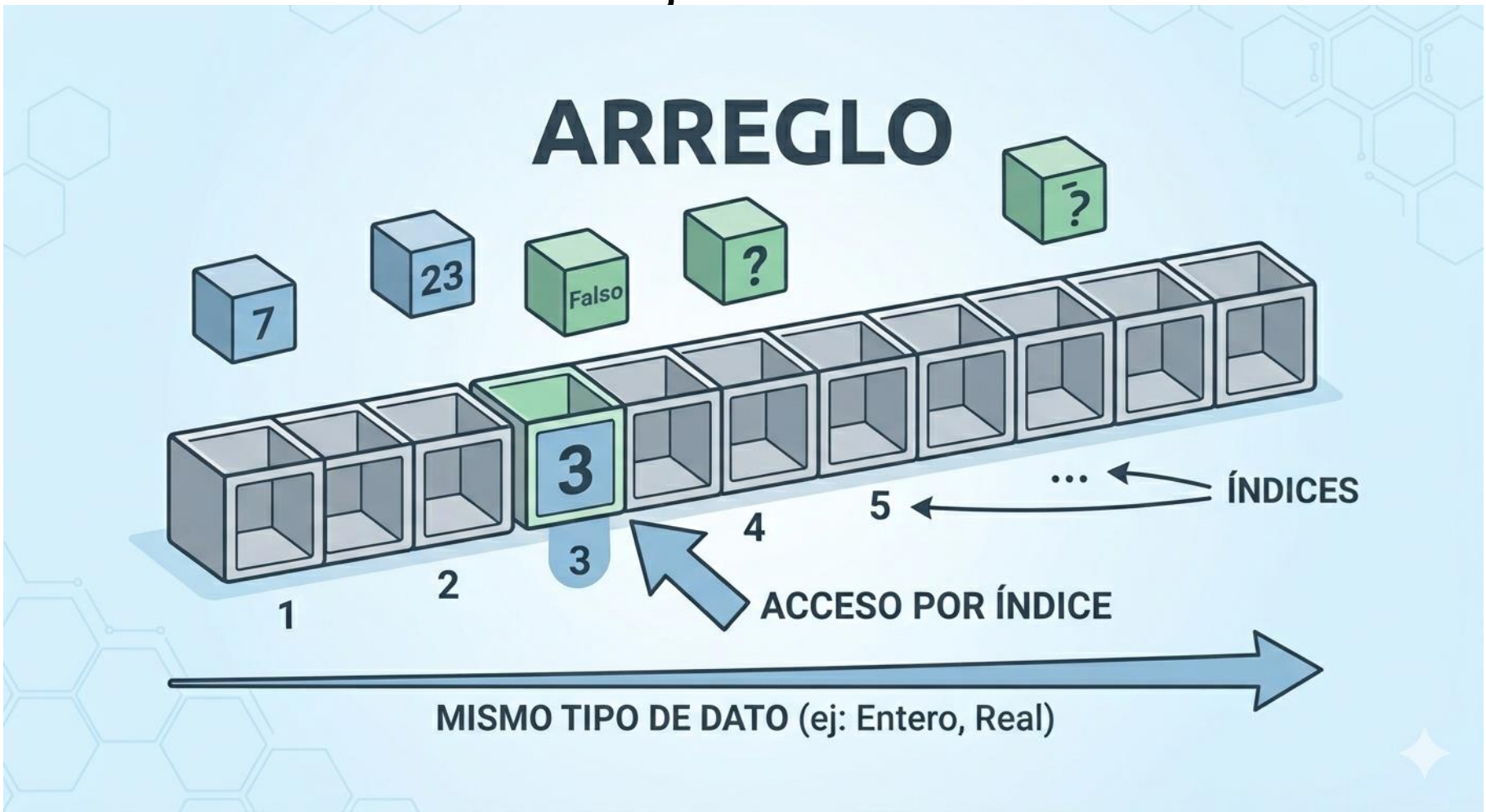
La mala idea: Crear 100 variables distintas.

```
Definir VENT1, VENT2, VENT3, VENT4,
VENT5, VENT6, VENT7... VENT100 como real
Leer VENT1
Leer VENT2
Leer VENT3
Leer VENT4...
Definir VENT1, VENT2, VENT3, VENT4,
VENT5, VENT6, VENT7... VENT100 como real
Leer VENT1
Leer VENT2
Leer VENT3
Leer VENT4...
Leer VENT5
Leer VENT6
Leer VENT7
Leer VENT8
Leer VENT9
Leer VENT10...
```

Llamar a cada dato VENTA1, VENTA2, VENTA3...
es lento, aburrido y fácil de equivocarse.

Un **arreglo** es la solución a este problema.

Es una estructura de datos que permite almacenar una **colección de elementos** bajo un **mismo nombre**.

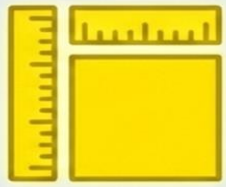


Un arreglo agrupa muchos casilleros bajo un solo nombre

Las 3 reglas de oro de los arreglos.

Mismo Tipo

Todos los casilleros guardan el mismo tipo de dato (solo números o solo caracter o solo lógicos).



Tamaño Fijo

Decides cuántos casilleros necesitas desde el principio. No cambia.

Acceso Directo

Llegas rápido a cualquier dato usando su número de posición.

Arreglo lineal.

Un **arreglo** es una zona de almacenamiento contigua en la memoria que contiene elementos del **mismo tipo de dato,** a los que se accede mediante un **índice numérico**.

Características Principales

Homogéneos: Todos los elementos dentro del arreglo deben ser del mismo tipo (ej: todos números enteros, o todas letras).

Tamaño Fijo: Debes definir cuántos elementos tendrá el arreglo antes de usarlo. Este tamaño no cambia durante la ejecución del programa.

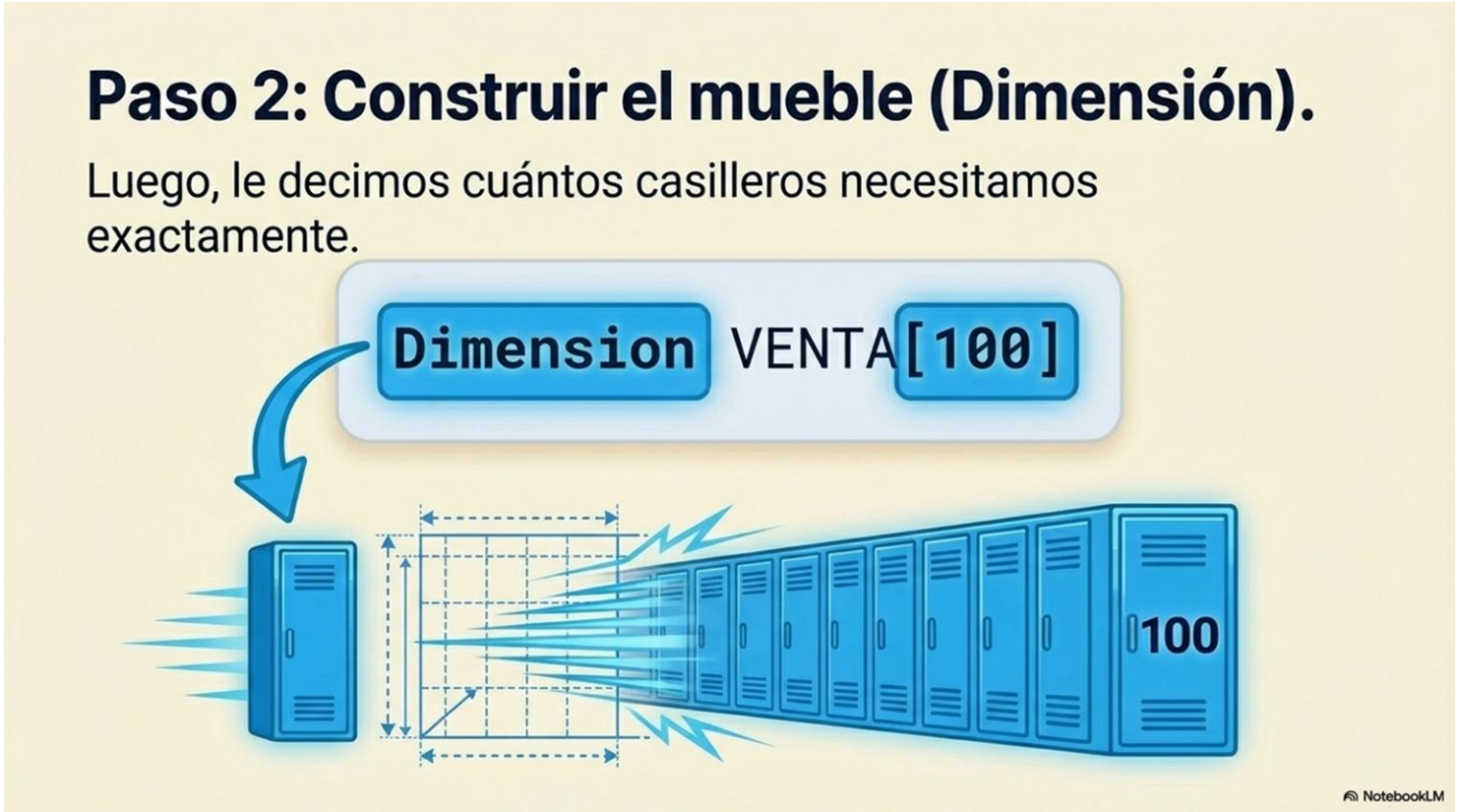
Índice: Es un número entero que indica la posición de un elemento dentro del arreglo. En muchos lenguajes (y por defecto en PSeInt) el índice comienza en 1.

En primer lugar, se define el **nombre del arreglo** con el **tipo de dato** asociado a todos los elementos del arreglo. Por ejemplo:

Definir V Como Entero

Luego se establece la dimensión (cantidad de elementos) de la estructura.

Dimension V [7]



Declaración en Dos Pasos

Paso 1: Definir Tipo

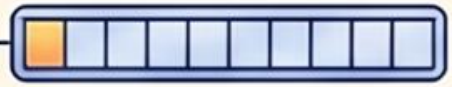
```
Definir A Como Real;
```

Número Real
Establece qué tipo de valores guardará.

	Tipo de Dato	
	Definir A Como Real	Establece qué tipo de valores guardará.

Paso 2: Dimensionar

```
Dimension A[10];
```

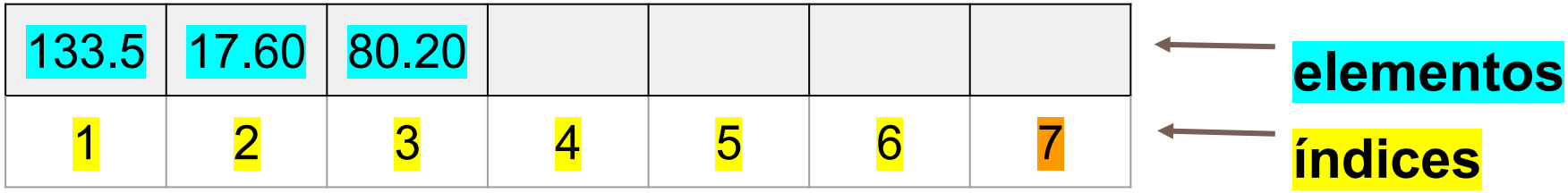


Reserva 10 espacios de memoria indexados.

Tamaño	**Dimension A[10]**	Reserva 10 espacios de memoria indexados.
Índice	**A[1]**	En esta configuración, el primer elemento inicia en 1.

Cada casillero tiene un número de posición. En PSeInt, siempre empezamos a contar desde el 1. Usamos corchetes [ ] para elegir el casillero.

Arreglos lineales.



¿Cuál podría ser el nombre de esta variable arreglo?

Dimensión= 7

1. La **dimensión o tamaño del arreglo** es la cantidad de celdas.
2. Los **elementos del arreglo** son los valores de cada celda.
3. Si se quiere recuperar o almacenar un elemento de una celda, se necesita la posición del arreglo a través de los **índices**.

IMPORTANTE!

Definir el **tipo de dato** que es el **mismo** para todos los elementos del arreglo.

Arreglos en PSeInt.

Entonces, la definición de arreglos en PSeInt se realiza en **2 partes**:

Temp							
	-3	17	-10	26			
	1	2	3	4	5	6	7

En primer lugar, se define el **nombre del arreglo** con el **tipo de dato** asociado a todos los elementos del arreglo. Por ejemplo:

Definir Temp **Como** Entero

Luego se establece la dimensión (cantidad de elementos) de la estructura.

Dimension Temp[7]

El perfil de la UNSL que se utiliza en la materia establece que el índice comienza en 1.

Operaciones con arreglos

Al igual que sucede con cualquier variable simple o primitiva, es posible:

ALMACENAR UN VALOR EN UN ELEMENTO DE UN ARREGLO Y

RECUPERAR SU VALOR.

```
M ← 54
car ← 'k'
flag ← VERDADERO
Leer NUM
Leer Nota
```

```
Escribir NUM
Escribir Nota
```

```
Prom ← ( M * 8 ) / 4
```

Para guardar información, escribes el nombre del arreglo, la posición exacta y el valor.


```
VENTA[100] <- 1230
```


Para ver la información o usarla en un cálculo matemático, solo llamas al nombre y su posición.

Recuperación

```
Escribir VENTA[3]
```

Uso en Cálculo

```
J <- 13 * J + VENTA[1]
```

Operaciones con arreglos: Asignación

Al igual que sucede con cualquier variable simple o primitiva, es posible **asignar UN valor a UN elemento de un arreglo y recuperar su valor**.

Por ejemplo, sea **Temp** declarado como un arreglo de 7 elementos enteros, entonces la operación de asignación:

$$Temp[3] \leftarrow -15$$

Indica que al tercer elemento de **Temp** se le asigna el valor -15, gráficamente:

Temp

		-15				
1	2	3	4	5	6	7

Al igual que sucede con cualquier variable simple o primitiva, es posible con la sentencia **Leer** almacenar **UN valor en UN elemento** de un arreglo.

Por ejemplo, sea **Temp** declarado como un arreglo de 7 elementos enteros, entonces la sintaxis es:

Leer Temp
[3]

Indica que al tercer elemento de **Temp** se le almacena el valor ingresado por teclado con la sentencia Leer.

¿Qué valor debería ser para el día de hoy?

Temp

1	2	3	4	5	6	7

Operaciones con arreglos: Recuperación de valores

			-2											
1	2	3	4	5	6	7								K

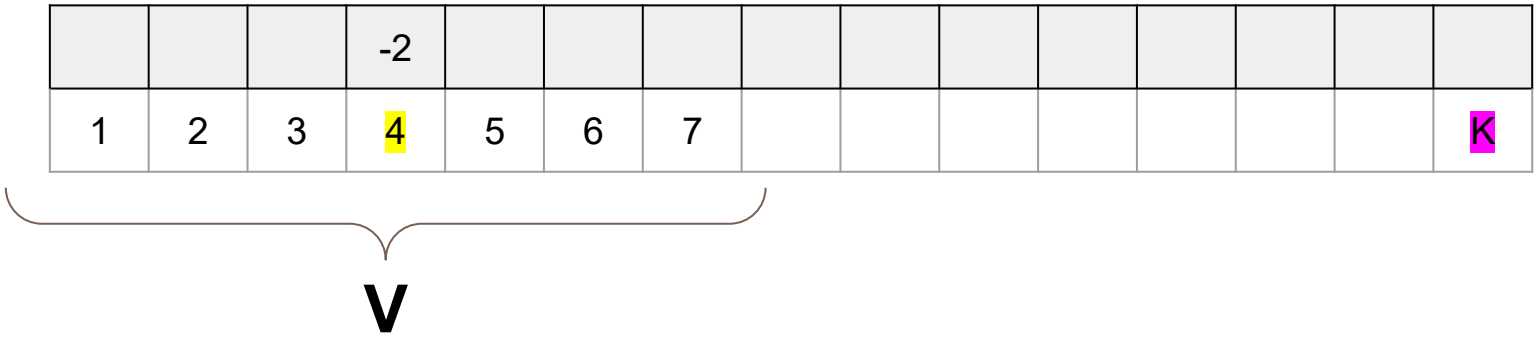
V

Escribir **V** [4]

Para resolver:

1. Recuperar el valor del cuarto elemento del arreglo **V.**
2. Muestra en la pantalla el valor recuperado.

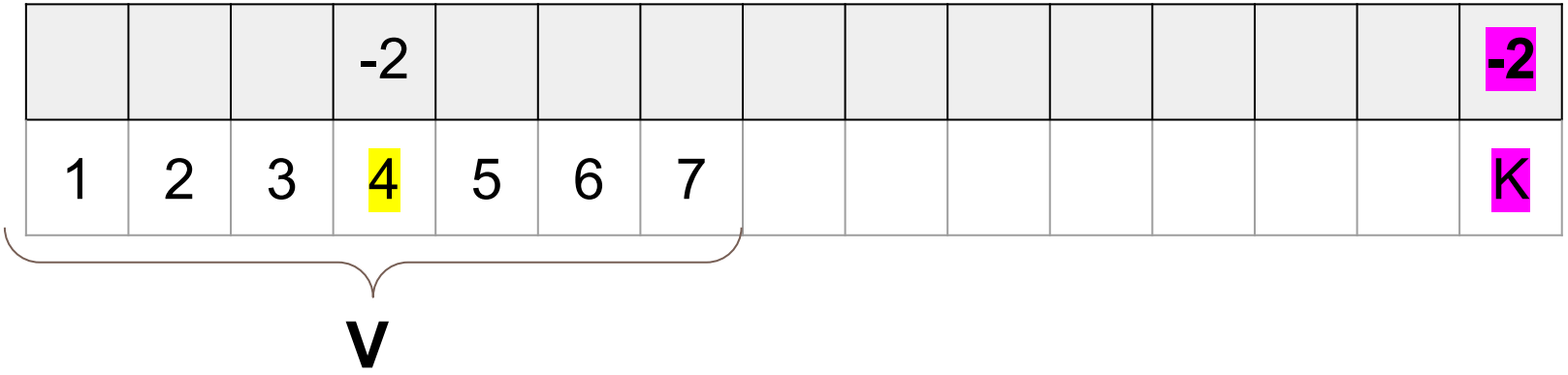
Operaciones con arreglos: Recuperación de valores



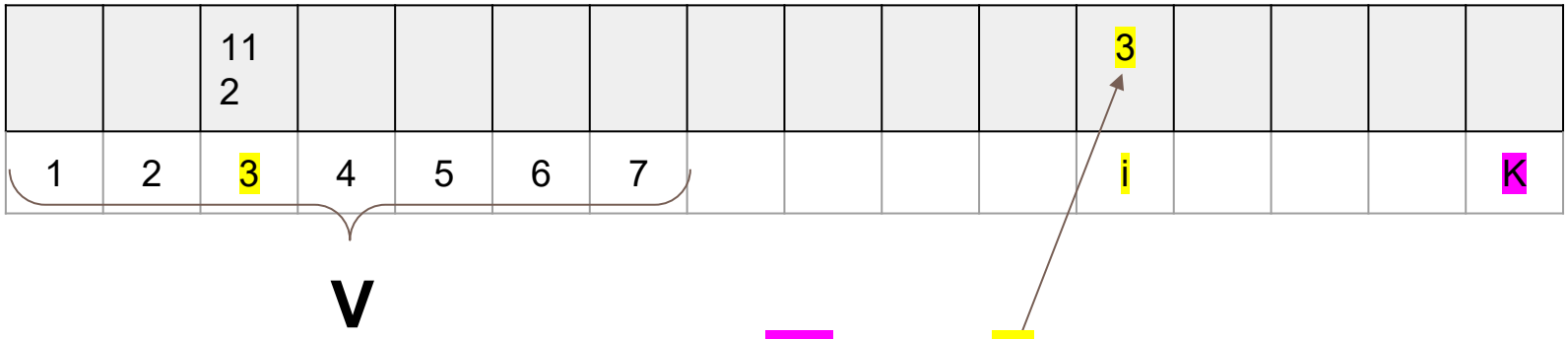
K ← V [4]

Para resolver:

1. Recuperar el valor del cuarto elemento del arreglo **V.**
2. Asignar a la variable K el valor recuperado.

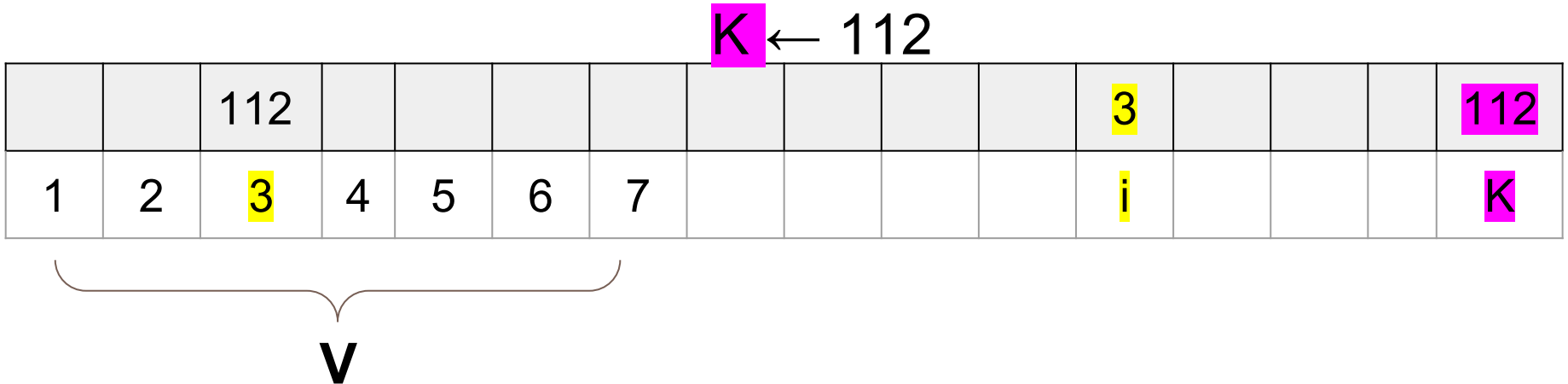


Operaciones con arreglos: Recuperación de valores

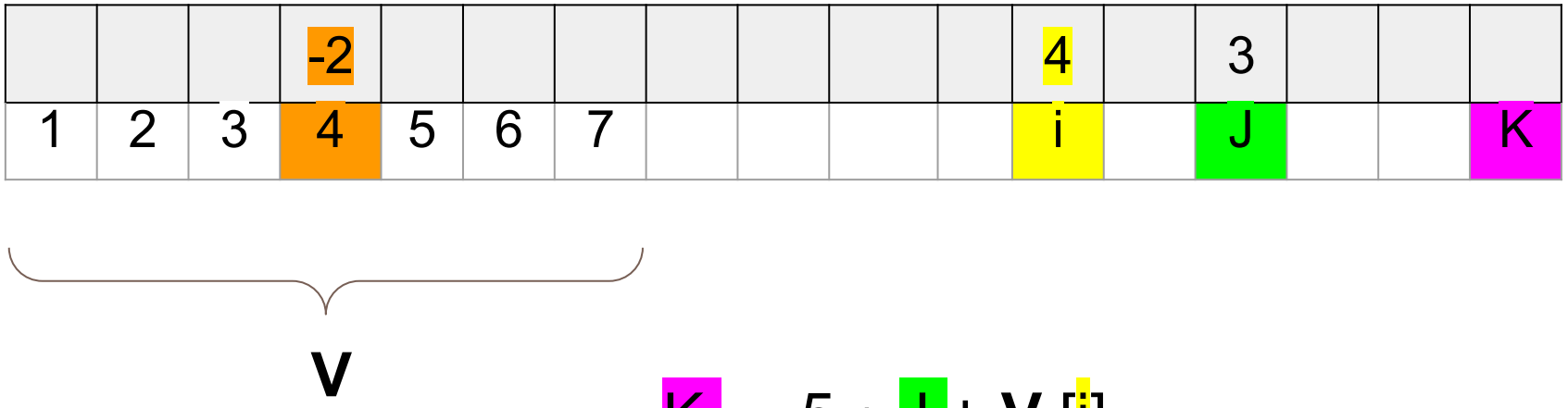


Para resolver:

1. Recuperar el valor del i-ésimo elemento del arreglo **V**, es decir el valor del tercer elemento.
2. Asignar a la variable K el valor recuperado.



Operaciones con arreglos: Recuperación de valores



$$K \leftarrow 5 * J + V[i]$$

Para resolver:

1. Recuperar el valor de la variable J

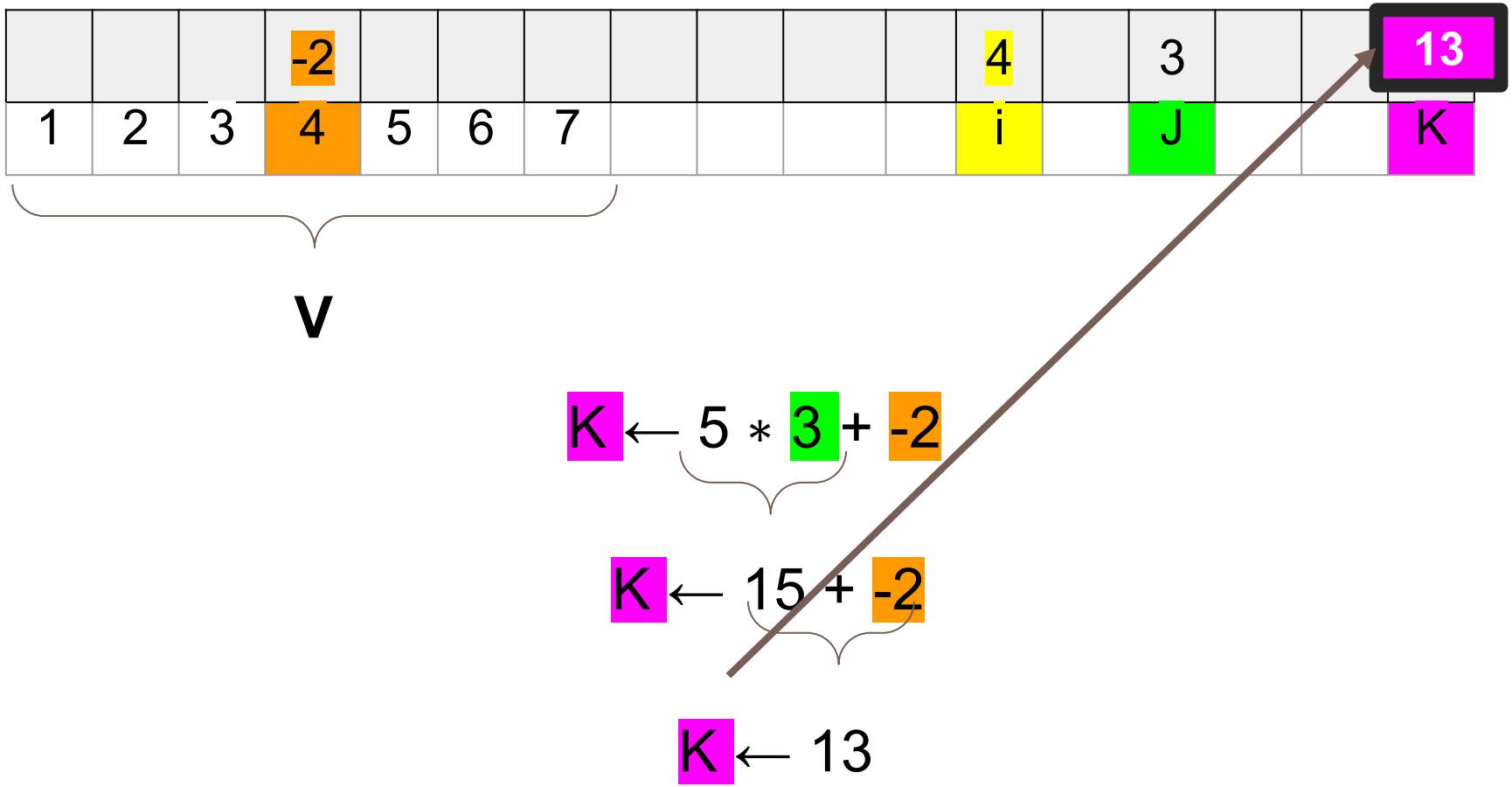
$$K \leftarrow 5 * 3 + V[i]$$

2. Recuperar el valor de la variable i

$$K \leftarrow 5 * 3 + V[4]$$

3. Recuperar el valor de V[4]

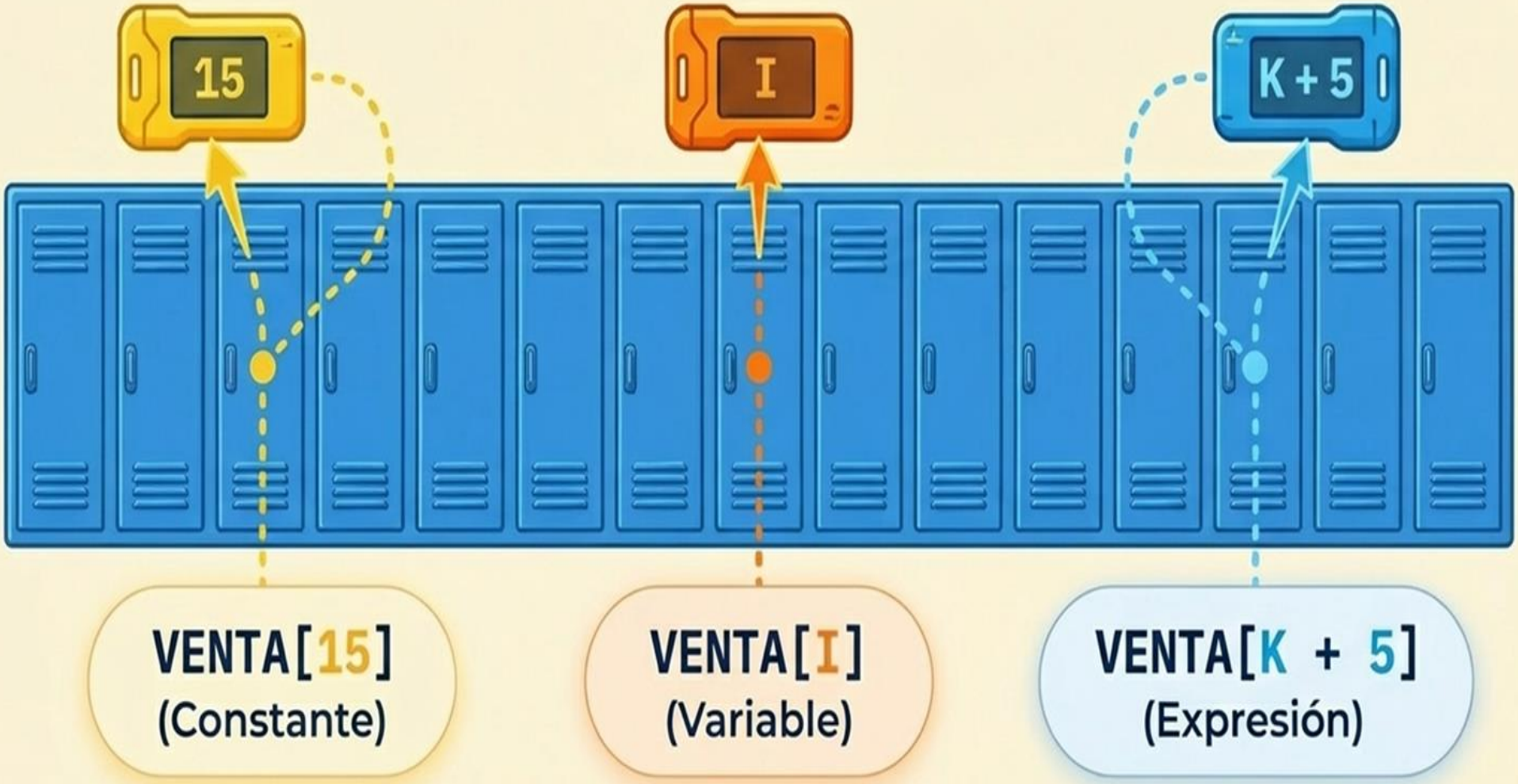
$$K \leftarrow 5 * 3 + -2$$



IMPORTANTE!

Los valores del arreglo **V,** y de las variables **i** y **J** se conservan igual, no se modifican, ni se sobreescriben en estas sentencias.

El número entre los corchetes no tiene que ser fijo. ¡Puede ser una variable o una fórmula! Esto nos permite crear magia.



Tu Hoja de Ruta Visual.

Crear y dar Tipo:

```
Definir VENTA Como Real
```

Dar Tamaño:

```
Dimension VENTA[100]
```

Guardar Dato:

```
VENTA[1] <- 10
Leer VENTA[1]
```

Mostrar Dato:

```
Escribir VENTA[1]
```

Iteración simple

Mientras <condicion> **Hacer**

<secuencia de acciones>

FinMientras

```
cont<-1
Mientras cont <= 4 Hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
    si nro MOD 2 = 0 Entonces
        Escribir "El numero es par"
    FinSi
    cont<-cont+1
FinMientras
```

Para trabajar con arreglos, puede resultar útil disponer de una estructura de control repetitiva que permita que una secuencia de acciones se ejecute un **número fijo** de veces, conocido de antemano.

Además, de la sentencia de iteración simple **Mientras**.

```
cont<-1
Mientras cont <= 4 Hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
    si nro MOD 2 = 0 Entonces
        Escribir "El numero es par"
    FinSi
    cont<-cont+1
FinMientras
```

En la estructura de control Iteración simple tenemos:

Iteración simple

V ← Vinicial

Mientras V <= Vfinal **hacer**

<secuencia de acciones>

V ← V + P

FinMientras

Teniendo en cuenta las partes de la estructura de control Iteración simple:

V ← Vinicial

Mientras V <= Vfinal **hacer**

<secuencia de acciones>

V ← V + P

FinMientras

Iteración simple

El formato de esta nueva estructura de control es el siguiente:

Para V ← Vinicial **Hasta** Vfinal **Con Paso** P **Hacer**

<secuencia de acciones>

FinPara

Entonces este código escrito con la sentencia **Mientras**:

```
cont<-1
Mientras cont <= 4 Hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
    si nro MOD 2 = 0 Entonces
        Escribir "El numero es par"
    FinSi
    cont<-cont+1
FinMientras
```

Se puede escribir con la sentencia **Para** como:

```
Para cont<-1 hasta 4 con paso 1 hacer
    Escribir "Ingrese un número cualquiera "
    Leer nro
    si nro MOD 2 = 0 Entonces
        Escribir "El numero es par"
    FinSi
FinPara
```

Partes de la nueva estructura de control es el siguiente:

Para V ← Vinicial **Hasta** Vfinal **Con Paso** P **Hacer**
    <secuencia de acciones>
FinPara

Donde:

- V es una variable de **tipo entero** llamada **variable de control** de la repetición.

- Vinicial, Vfinal y P pueden ser variables o constantes de tipo entero o expresiones aritméticas.

El formato de esta nueva estructura de control es el siguiente:

Para V ← Vinicial **Hasta** Vfinal **Con Paso** P **Hacer**
    <secuencia de acciones>
FinPara

Donde:

- Vinicial recibe el nombre de valor inicial;
- Vfinal recibe el nombre de valor final y

- **P** es el paso, o sea en cuanto se incrementa (o decrementa) la variable V para llegar desde Vinicial al Vfinal y debe ser distinto de cero (pero puede ser positivo o negativo).

Lo mejor de los arreglos es combinarlos con un ciclo 'Para'.
La variable I avanza de casillero en casillero automáticamente.

```
Para I ← 1 Hasta 100 Con Paso 1 Hacer
    Escribir "Ingrese venta de la sucursal ",I
    Leer VENTA[I]
FinPara
```

Para V ← Vinicial **Hasta** Vfinal **Con Paso** P **Hacer**
<secuencia de acciones>
FinPara

Altura

1.23	1.67	1.78	1.6	1.5						
1	2	3	4	5	6	7	8	9	10	11

```
1  // Almacenar los datos en el arreglo Alturas
2
3  Proceso leerAlturas
4      Definir Altura como Real
5      Dimension Altura[11]
6      Definir i,N como Entero
7
8      N<-5
9
10     i<-1
11     Mientras i <= N Hacer
12         Escribir "Ingrese la altura ",i,":"
13         Leer Altura[i]
14     i<-i+1
15     FinMientras
16
17 FinProceso
```

```
// Almacenar los datos en el arreglo Altu

Proceso leerAlturas
    Definir Altura como Real
    Dimension Altura[11]
    Definir i,N como Entero

    N<-5

    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir "Ingrese el dato ",i,":"
        Leer Altura[i]
    FinPara


FinProceso
```

¿Cómo almacenar los valores de elementos de un arreglo?

Altura

1.23	1.67	1.78	1.6	1.5						
1	2	3	4	5	6	7	8	9	10	11

```
// Almacenar los datos en el arreglo Alturas

Proceso leerAlturas
    Definir Altura como Real
    Dimension Altura[11]
    Definir i,N como Entero

    N<-5

    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir "Ingrese la altura ",i,":"
        Leer Altura[i]
    FinPara

FinProceso
```

Almacenando valores de **N** elementos del arreglo, o sea de 5 elementos del arreglo.

Almacena valor de un elemento del arreglo

¿Cómo muestro en pantalla los elementos de un arreglo?

Altura

1.23	1.67	1.78	1.6	1.5						
1	2	3	4	5	6	7	8	9	10	11

```
// Almacenar los datos en el arreglo Alturas

Proceso leerAlturas
    Definir Altura como Real
    Dimension Altura[11]
    Definir i,N como Entero

    N<-5

    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir "Ingrese la altura ",i,":"
        Leer Altura[i]
    FinPara

    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir Altura[i]
    FinPara

FinProceso
```

Almacena la altura en **N** elementos

¿Cómo muestro en pantalla los elementos de un arreglo?

Altura

1.23	1.67	1.78	1.6	1.5						
1	2	3	4	5	6	7	8	9	10	11

```
// Almacenar los datos en el arreglo Alturas

Proceso leerAlturas
    Definir Altura como Real
    Dimension Altura[11]
    Definir i,N como Entero

    N<-5

    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir "Ingrese la altura ",i,":"
        Leer Altura[i]
    FinPara

    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir Altura[i]
    FinPara

FinProceso
```

Muestra valores de **N** elementos del arreglo, o sea de 5 elementos del arreglo.

Muestra el valor de un elemento del arreglo

Teoría

Pseudocódigo: Estructura de datos

- ✓ **Estructura de datos: Arreglo lineal.**
  - Declaración.
  - Operaciones con arreglos: Asignación y recuperación de valores.
- ✓ **Estructuras de control: secuencial, condicional y de repetición (PARA).**
- ✓ **Pseudocódigo con PSeInt.**

¡Ya podemos comenzar con el Práctico!