



**UNER**

Facultad de Ciencias  
de la **Administración**

*Algoritmos y Programación*

**Estructuras de  
Datos**



Universidad Nacional  
de **Entre Ríos**

**Año: 2018**

# Estructuras de Datos

---

## CONCEPTO:

Una **estructura de datos**, es un conjunto de datos elementales que tienen un mismo nombre colectivo, y que están agrupados y organizados de forma tal que resulte simple su procesamiento.

## CLASIFICACION:

### a) Por su complejidad

1. Arreglos
  - 1.1. Unidimensionales
    - 1.1.1. Lineales
    - 1.1.2. Enlazados
  - 1.2. Bidimensionales
  - 1.3. Multidimensionales
2. Archivos
3. Bases de Datos

### b) Por su organización

1. Estructuras estáticas
2. Estructuras dinámicas

# Estructuras de Datos

---

## Operaciones:

Cualquiera sea el tipo de estructura que consideremos, las operaciones que habitualmente realizaremos serán:

| OPERACIÓN     | DESCRIPCION DE LA OPERACIÓN  |
|---------------|--|
| Recorrido:    | Procesar cada uno de los datos que integran la estructura                |
| Búsqueda:     | Ubicar uno de los datos de la estructura                                 |
| Inserción:    | Agregar un nuevo dato a la estructura                                    |
| Borrado:      | Eliminar un dato de la estructura  |
| Ordenamiento: | Organizar los datos de la estructura de acuerdo con algún tipo de orden. |
| Mezcla:       | Combinar dos estructuras en una sola.                                    |

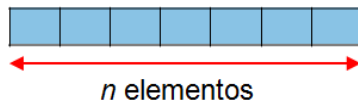
# Estructuras de Datos

## Arreglos.

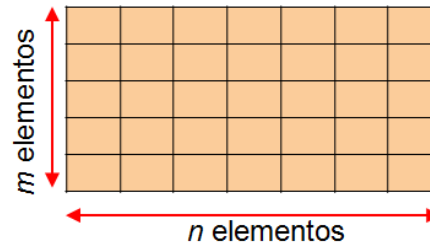
Un arreglo (array) es un objeto de datos conformado por un conjunto de datos ( llamados elementos o componentes ) homogéneos (es decir de un mismo tipo).

## Dimensiones:

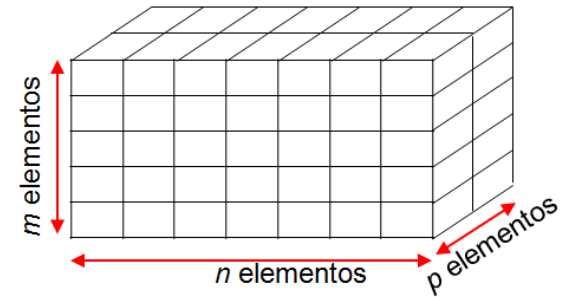
Una dimension



Dos dimensiones



Tres dimensiones



## Arreglos Unidimensionales

Sus elementos están dispuestos en una sola dimensión.

Para la referencia o acceso a un elemento se utiliza una única dirección.

## Arreglos unidimensionales lineales

Llamados también vectores o listas, son aquellos formados por un conjunto finito 'n' de elementos que forman una secuencia, almacenados en posiciones consecutivas de memoria.

# Estructuras de Datos

---

Arreglos unidimensionales lineales:

Representación Gráfica



# Estructuras de Datos

---

El número 'n' de elementos recibe el nombre de longitud o tamaño del vector.

A este vector se lo indica como una sucesión de elementos con notación subindicada utilizando el nombre colectivo.

Por ejemplo:

$$A_1, A_2, A_3, \dots, A_n$$

que representa el vector 'A' de longitud 'n', siendo  $A_1, A_2, \dots, A_n$  cada uno de sus elementos o componentes.


# Estructuras de Datos

---

Gráficamente se suele indicar a los vectores como 'cajas unidas' una a continuación de la otra.

Ejemplo: un vector **DATOS** con 8 elementos se representaría:

**DATOS**



|           |    |     |    |    |     |    |    |     |
|-----------|----|-----|----|----|-----|----|----|-----|
| Valores:  | 12 | 345 | 24 | 73 | 127 | 87 | 72 | 155 |
| Posición: | 1  | 2   | 3  | 4  | 5   | 6  | 7  | 8   |

Así, en el vector **DATOS**:

Valor del primer elemento: 12

valor del segundo elemento: 345

valor del tercer elemento: 24

# Estructuras de Datos

---

## Características principales y notación.

- a) Cada elemento del vector debe tener la misma longitud.
- b) La notación para hacer referencia a un elemento del vector es: Nombre colectivo y entre paréntesis la posición relativa que ocupa el elemento dentro de la lista o secuencia.

**Por ejemplo:** Sea el vector DATOS que contiene 8 elementos, la expresión:

**DATOS(5)**

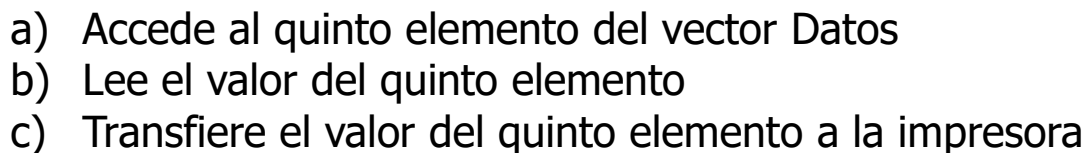
hace referencia el elemento ubicado en la quinta posición del vector DATOS.

y: **Imprimir: DATOS(5)**

es la acción que permite imprimir el contenido del elemento ubicado en la quinta posición del vector datos.



## Ejemplo.



# Estructuras de Datos

---

## **Formas de acceder:**

### **a.- Directa:**

A través del subíndice (Selector dinámico), se puede acceder a cualquier elemento del array.

### **b.- Secuencial:**

En este caso se accede a todos los elementos del array, recorriendo la estructura de datos.

Se utiliza un esquema repetitivo.

El recorrido puede efectuarse en dos direcciones:

- Desde el primer elemento hasta el último
- Desde el último elemento hasta el primero.

# Estructuras de Datos

---

## Selector Dinámico (Subíndices)

La notación para hacer referencia a un elemento de un vector es consignar su nombre colectivo y entre paréntesis la posición relativa que ocupa el elemento dentro de la lista o secuencia.

Por lo tanto, lo consignado entre paréntesis, que llamaremos **subíndice**, indica la posición del elemento accedido.

Se pueden utilizar como subíndices:

**a.- Una constante**

**b.- Una variable**

En este caso se accede al elemento ubicado en la posición que coincide con el valor de la variable.

**c.- Una expresión aritmética**

En este caso se accede al elemento ubicado en la posición que coincide con el resultado de la expresión aritmética.

# Estructuras de Datos

---

## Especificación.

### TIPOS DE DATOS:

a) Proporcionados por el diseño del lenguaje:

- Enteros
- Reales
- String
- Booleanos

b) **Generados por el usuario:** Son los que pueden ser creados por el usuario a partir de los datos proporcionados por el diseño del lenguaje:

### **tipo estructurados: Arrays**

En general se definen en una parte especial del programa **"Type"** que se codifica antes de la definición de variables.

# Estructuras de Datos

## Tipo booleano (lógico):

Puede representar valores de **lógica** binaria ( solamente 2 valores).

Estos valores normalmente representan falso o verdadero.

### Especificación:

```
Var  
  (nombre variable) : Boolean;
```

### Asignación: Sintaxis:

```
(nombre variable) := { True  
                      False
```

### Program Ejemplo:

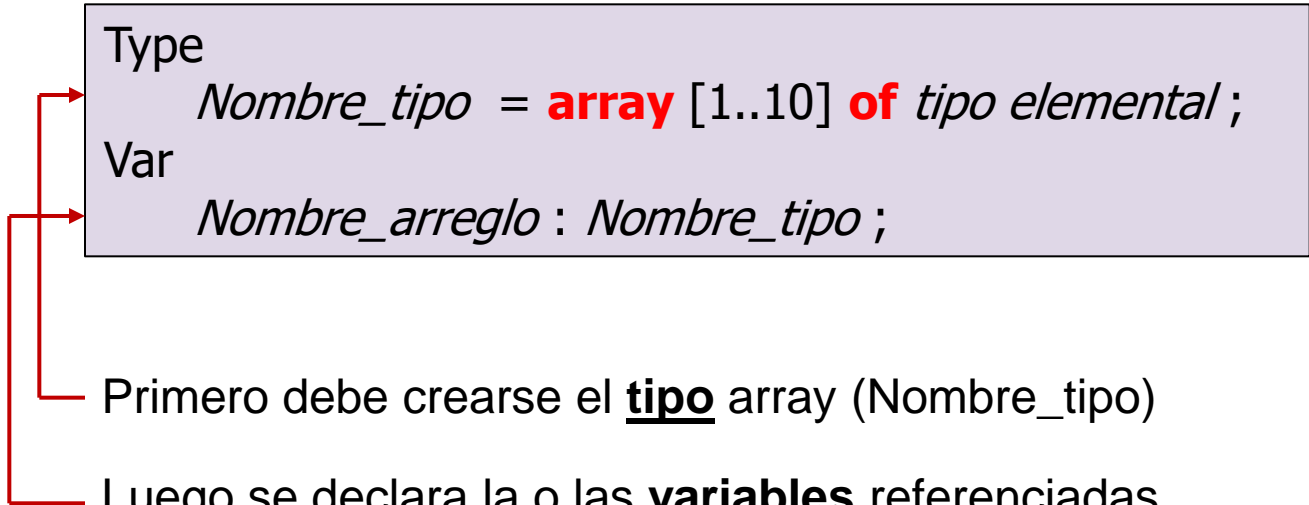
```
Var  
  OPCION : Boolean;  
  .....  
Inicio  
  OPCION := False  
  .....  
  Si OPCION = True  
    .....  
  FinSi  
  Mientras OPCION = False  
    .....  
  FinMientras  
  .....  
FIN
```

# Estructuras de Datos

---

## ARRAYS (Arreglos) : Especificación

- Para declarar una variable tipo array, se utiliza la siguiente sintaxis:



```
Type
    Nombre_tipo = array [1..10] of tipo elemental ;
Var
    Nombre_arreglo : Nombre_tipo ;
```

Primero debe crearse el **tipo** array (Nombre\_tipo)

Luego se declara la o las **variables** referenciadas al tipo array (nombre\_arreglo).

Las declaraciones de tipo array no crea ninguna variable específica de tipo array, sino que proporciona información del array como un todo.

# Estructuras de Datos

---

## Especificación de arreglos.

- **Ejemplo:**

Definir:

- a) un tipo de arreglo de nombre “Vector” con 50 elemento de tipo entero.
- b) una variable de nombre “Datos” de tipo “Vector”

**Type**

Vector = array [1..50] of integer;

**Var**

Datos : Vector;

# Estructuras de Datos

## Formato Para – FinPara

$$\text{Para ( variable-1 )} = \left\{ \begin{array}{c} \text{Variable-2} \\ \text{Constante-1} \end{array} \right\} ; \left\{ \begin{array}{c} \text{Variable-3} \\ \text{Constante-2} \end{array} \right\} ; \left\{ \begin{array}{c} \text{Variable-4} \\ \text{Constante-3} \end{array} \right\}$$

**Acción/es**

FinPara

### Donde:

**variable-1** : es la variable que cambia de valor (incremento o decremento)

**Variable-2 o constante-1** : es el valor de inicialización de variable-1

**variable-3 o constante-2** : es el valor final de variable-1 que permite terminar el esquema.

**variable-4 o constante-3** : es el valor de incremento o decremento de cada iteración.

De esta manera el tercer valor puede adquirir signo positivo o negativo.



# Estructuras de Datos

---

## Casos Tipos:

1.- Generar un vector 'DATOS' de 100 elementos enteros con valores Cero.

```
Type  
  Vector = array [1..100] of integer;  
Var  
  Datos : Vector;  
  i : integer;
```

```
INICIO  
PARA i = 1, 100, 1  
    DATOS(i) := 0  
FINPARA  
FIN
```

2.- Listar por impresora los valores de los elementos de un vector 'TOTALES' que tiene 80 elementos

```
Type  
  Vector = array [1..80] of real;  
Var  
  Totales : Vector;  
  i : integer;
```

```
INICIO  
PARA i = 1, 80, 1  
    Imprimir TOTALES(i)  
FINPARA  
FIN
```

# Estructura de Datos

---

Ingresa un dato por teclado, buscarlo en el vector 'DATO' que tiene 200 elementos enteros y mostrar la posición que ocupa.

## **Program Vector1**

Type

Vector = array [1..200] of integer;

Var

Dato : Vector;

i , VALOR : integer;

## **INICIO**

Ingresa VALOR

Para i = 1, 200, 1

Si DATO(i) = VALOR

Mostrar i

i := 200

FINSI

FinPara

## **FIN**

# Estructuras de Datos

En el caso anterior, contemplar la posibilidad de que el valor ingresado NO se encuentre en el vector, en tal caso Mostrar '**VALOR NO ENCONTRADO**'.

## Program Vector1

Type

Vector = array [1..200] of integer;

Var

Dato : Vector;

i , VALOR : integer;

Ban : boolean;

## INICIO

Ingresar VALOR

Ban := False

Para i = 1, 200, 1

Si DATO(i) = VALOR

Mostrar i

i := 200

Ban := True

FINSI

FinPara

SI Ban = False

Mostrar 'Valor No encontrado'

FINSI

FIN

# Estructuras de Datos

Caso en que se contemple varias consultas y salida del loop sin modificar la variable de loop.

## Program Arreglo

**type**

**TIPVEC = array** (1..200) of integer

**Var**

DATO : TIPVEC;

i, VALOR : Integer;

BAN : Boolean;

**BEGIN**

Ingresa VALOR

**Mientras** VALOR <> 0

BAN := False

i := 0

**Mientras** i < 200 and BAN = False

i := i + 1

Si DATO(i) = VALOR

Mostrar i

BAN := True

FinSi

**FinMientras**

Si BAN = False

Mostrar "Valor no Encontrado"

FinSi

Ingresa VALOR

**FinMientras**

**END**

# Estructuras de Datos

---

## Ejercicios Elementales

1. Generar un vector de 10 elementos que contenga valor 0 en los elementos 1 a 9 y un valor 1 en el décimo elemento.
2. Generar un vector de 20 elementos que contenga valor 0 en sus elementos pares y valor 1 en los impares.
3. Generar un vector de 100 elementos que contenga los números naturales impares comenzando por el número 1.
4. Diagramar el proceso mediante el cual se obtenga la sumatoria de los elementos de un vector de "N" dimensiones e imprimir su resultado.
5. Se tiene grabado en memoria un vector de 100 elementos que contiene diversos números naturales impares, imprimir todos los elementos cuyos valores estén comprendidos entre el 21 y el 77 junto con la posición que ocupa.

# Estructuras de Datos

---

**FIN**