

FUNDAMENTOS DE PROGRAMACIÓN ARREGLOS ESTÁTICOS



Facultad de Ingeniería
Universidad Nacional de Jujuy

DATOS SIMPLES Y COMPUESTOS

- Las variables de tipos de datos simples permiten almacenar un valor específico.

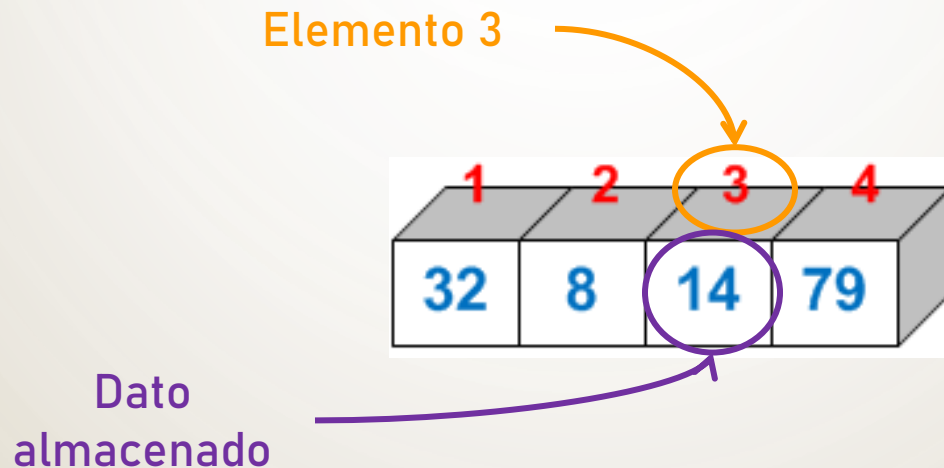


- Las variables de tipos de datos compuestos permiten almacenar un grupo de valores.



ARREGLOS. DEFINICIÓN

- Un arreglo es un **conjunto finito** de elementos del mismo tipo cuyo acceso se realiza a través de *índices*.
- Los **índices** permiten identificar en forma individual cada elemento del arreglo.



ÍNDICES

- Los índices permiten referenciar posiciones específicas de un arreglo.



- Los índices pueden ser constantes, variables o expresiones de tipo ordinal.
- El valor mínimo de un índice de arreglo se denomina límite inferior (**LI**), mientras que el máximo valor se llama límite superior (**LS**).

CLASIFICACIÓN

- Según la organización de sus elementos, los arreglos se clasifican en:

- Unidimensionales (vectores)

--- > **1 índice**

- Bidimensionales (matrices)

--- > **2 índices**

- Multi-dimensionales (n dimensiones)

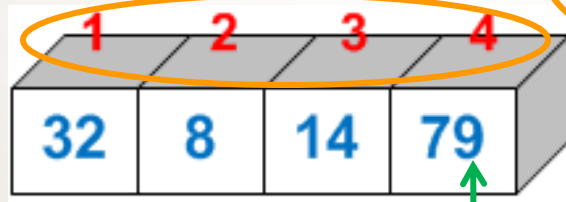
--- > **n índices**

- Los arreglos emplean tantos índices como dimensiones posean para referenciar sus elementos individuales.

VECTORES

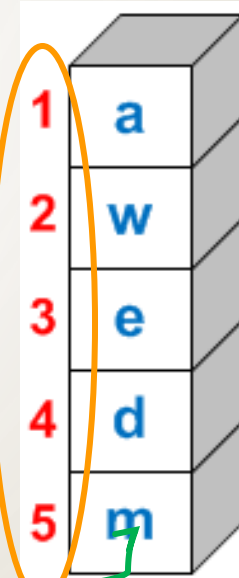
- El vector es el tipo más simple de arreglo ya que sus elementos se disponen en una única fila o columna y pueden accederse utilizando un solo índice.

Valor del índice
para cada
posición



Vector Fila

Datos del
vector



Vector Columna

¿CÓMO SE DECLARA?

- En general, se puede aplicar la siguiente sintaxis:

PROGRAMA ejemplo_vectores

CONSTANTES

MAX_ELEMENTOS=10

TIPOS

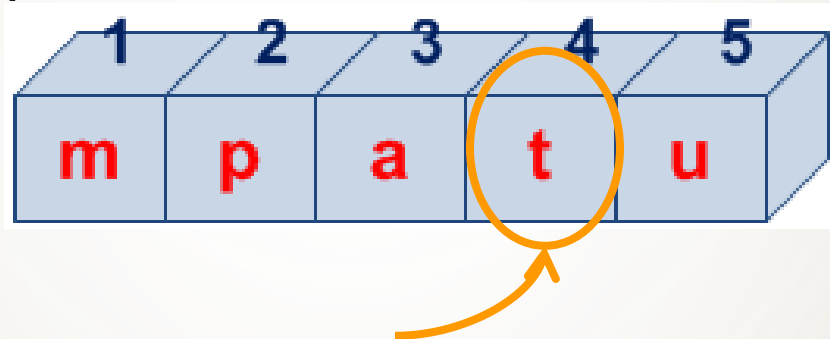
tipo_vector=**ARREGLO**[1..**MAX_ELEMENTOS**]de tipo_dato

VARIABLES

```
nombre_variable const int MAX_ELEMENTOS=10;  
typedef tipo_dato tipo_vector[MAX_ELEMENTOS];  
  
main()  
{ tipo_vector nombre_variable;  
}
```

¿CÓMO SE ACCEDE?

- Para acceder a una posición específica de un vector debe indicarse el nombre del vector y el índice del elemento requerido.



letras [4]

- *letras*: nombre del vector
- 4: posición del vector

MATRICES

- Un arreglo bidimensional (matriz o tabla) es un conjunto de elementos, todos del mismo tipo, que se organizan en filas y columnas.
- Requiere de 2 índices para identificar cada posición.
- El primer índice especifica la fila y el segundo la columna.

¿CÓMO SE DECLARA?

- En general, se puede aplicar la siguiente sintaxis:

PROGRAMA ejemplo_matrices

CONSTANTES

FILAS=12

COLS=14

TIPOS

tipo_matriz=**ARREGLO**[1..**FILAS**,1..**COLS**]de tipo_dato

VARIABLES

```
nombre_variable = tipo_matriz;

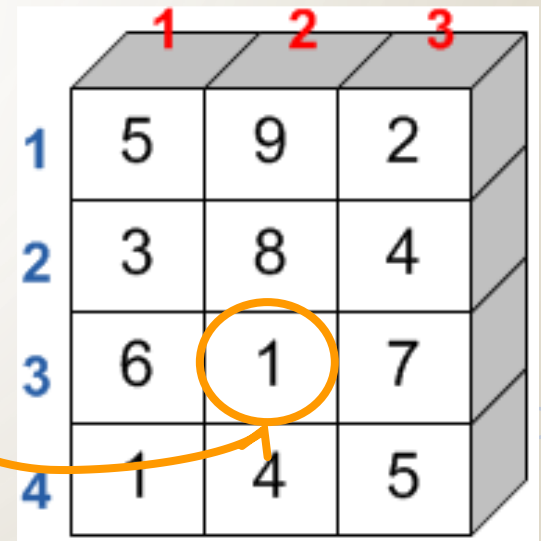
const int FILAS=12, COLS=14;
typedef tipo_dato tipo_matriz[FILAS][COLS];

main()
{ tipo_matriz nombre_variable;
}
```

¿CÓMO SE ACCEDE?

- Para acceder a una posición específica de una matriz debe indicarse el nombre de la matriz y los índices de fila y columna del elemento requerido.
 - *números*: nombre de la matriz
 - 3: fila de la matriz
 - 2: columna de la matriz

números [3, 2]



	1	2	3
1	5	9	2
2	3	8	4
3	6	1	7
4	1	4	5

MULTIDIMENSIONALES

- Un arreglo se dice multidimensional si posee 3, 4 o n dimensiones.
- Para acceder a los elementos de un arreglo multidimensional es necesario especificar tantos índices como dimensiones tenga el arreglo.

¿CÓMO SE DECLARA?

- En general, se puede aplicar la siguiente sintaxis:

PROGRAMA ejemplo_multidimensional

CONSTANTES

D1=12

D2=4

...

DN=7

TIPOS

tipo_mult=ARREGLO[1..D1,1..D2,...,1..DN] de
tipo_dato

VARIABLES

nombre_variabl

```
const int D1=12, D2=4, ..., DN=7;
typedef tipo_dato tipo_multi[D1][D2]...[DN];

main()
{ tipo_multi nombre_variable;
}
```

¿CÓMO SE ACCEDE?

- Para acceder a una posición específica de un arreglo multidimensional debe indicarse el nombre del arreglo y los índices correspondientes a cada dimensión.

- *alfabeto*: nombre del arreglo
- 4: primera dimensión
- 3: segunda dimensión
- 1: tercera dimensión

`alfabeto[4,3,1]`

	1	2	3
1	A	Z	U
2	M	V	O
3	E	T	W
4	Q	X	Y

RANGO DE UN ARREGLO (1)

- El *rango de un arreglo* indica el número de elementos o posiciones que posee un arreglo.
- El rango de un arreglo puede calcularse de la siguiente forma:
 - Rango Vector = $LS - LI + 1$
 - Rango Matriz = $(LS_1 - LI_1 + 1) \times (LS_2 - LI_2 + 1)$
 - Rango n-dimensional
 $= (LS_1 - LI_1 + 1) \times (LS_2 - LI_2 + 1) \times (LS_3 - LI_3 + 1) \times \dots \times (LS_n - LI_n + 1)$

RANGO DE UN ARREGLO (1)

- Rango Vector

6	10	17	22	23	33	40	43
-4	-3	-2	-1	0	1	2	3

$$LS-LI+1 = 3 - (-4) + 1 = 8 \text{ ELEMENTOS}$$

- Rango Matriz

	0	1	2	3
0	q	p	d	e
1	i	l	t	l
2	n	m	e	d
3	l	h	s	m
4	g	k	k	j
5	s	v	w	n

$$\begin{aligned} & (LS_F - LI_F + 1) * (LS_C - LI_C + 1) \\ &= (5 - 0 + 1) * (3 - 0 + 1) \\ &= 24 \text{ ELEMENTOS} \end{aligned}$$

OPERACIONES

- Asignación
- Lectura/escritura
- Recorrido
- Actualización
 - agregar, insertar y borrar
- Búsqueda
 - secuencial y binaria
- Intercalación
- Ordenación
 - burbuja selección inserción shaker sort rápido y

PROGRAMA DE EJEMPLO DE VECTOR

```
1  /*Enunciado del problema: defina un vector de 5 elementos numérico entero.
2  Calcule el cuadrado de los elementos de posiciones pares.
3  Calcule el cubo de los elementos de las posiciones impares.
4  Muestre los datos del vector original y el generado a partir de los cálculos realizados.*/
5
6  #include<bits/stdc++.h>
7  using namespace std;
8  const int MAX=5;
9  typedef float tvector[MAX]; //definición de un tipo de dato vector llamado tvector de 5 elementos de tipo entero
10 main ()
11 {
12     tvector datos, v_res; //definición de la variable datos de tipo vector y el tipo vector es tvector
13     int i;
14     for(i=0; i < MAX; i++) //recorrido que permite el ingreso de los elementos al vector
15     {
16         cout << "Ingrese un nro: ";
17         cin >> datos[i]; //operacion de escritura en cada elemento del vector
18     }
19     for(i=0; i < MAX; i++) //recorrido que permite realizar los cálculos solicitados
20     {
21
22         if(i%2==0)
23             v_res[i]=pow(datos[i],2); //datos[i]*datos[i];
24         else
25             v_res[i]=pow(datos[i],3); //datos[i]*datos[i]*datos[i];
26     }
27     cout<<"Vector original "<<endl;
28     for(i=0; i < MAX; i++) //recorrido que permite mostrar los elementos ingresados
29         cout << datos[i] << endl;
30     cout<<"Vector resultado "<<endl;
31     for(i=0; i < MAX; i++) //recorrido que permite mostrar los elementos ingresados
32         cout << v_res[i] << endl;
33     system("pause");
34 }
35
```

BIBLIOGRAFÍA

- Sznajdleder, Pablo Augusto. Algoritmos a fondo. Alfaomega. 2012.
- López Román, Leobardo. Programación estructurada y orientada a objetos. Alfaomega. 2011.
- De Giusti *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta, 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Joyanes Aguilar, Luis. Programación en Turbo Pascal. Mc Graw Hill. 1990.