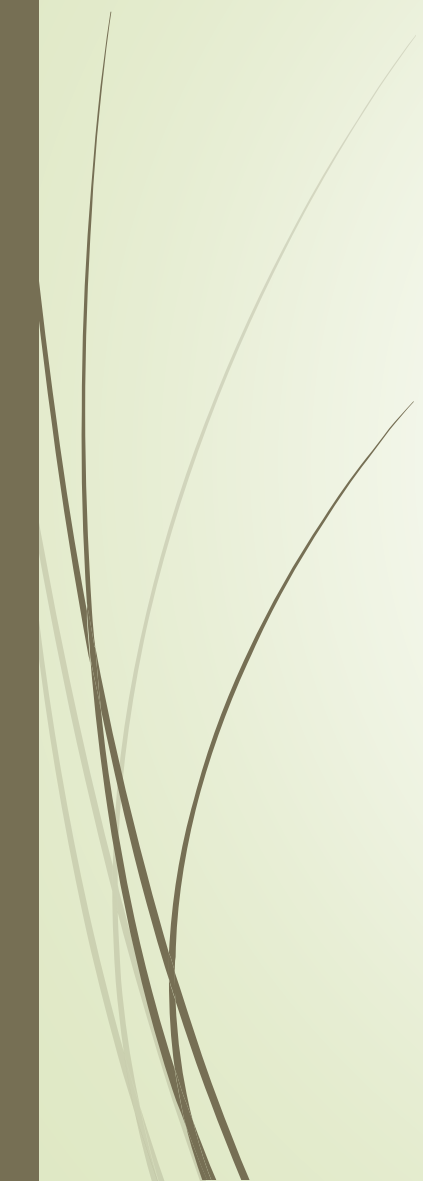


Arreglos

Catedrático: Ing. David Rajo



Estructura de Datos

- En programación, una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación.
- 



¿Qué es un arreglo? (array)

- Un arreglo (array) es una colección de datos del mismo tipo, que se almacenan en posiciones consecutivas de memoria y reciben un nombre común.
- Los Arreglos se utilizan para almacenar un conjunto de variables, que sean del mismo tipo de dato, y todas estas bajo un mismo nombre.

Clasificación de los arreglos

- Unidimensionales (vectores)
- Bidimensionales (tablas o matrices)
- Multidimensionales (tres o más Dimensiones)

Vector

Elemento 1
Elemento 2
Elemento 3
....
Elemento n

Tabla o matriz

Elemento 1,1	Elemento 1,2	...	Elemento 1,n
Elemento 2,1	Elemento 2,2	...	Elemento 2,n
Elemento 3,1	Elemento 3,2	...	Elemento 3,n
....
Elemento m,1	Elemento m,2	...	Elemento m,n

ARREGLOS UNIDIMENSIONALES

- Un Array(arreglo) o vector es una estructura que permite almacenar un conjunto de elementos o datos de un mismo tipo de datos.
- Un Arreglo se declara mediante el nombre del arreglo, el tamaño o número de elementos y el tipo de dato.
- Para acceder a cada elemento del arreglo se utiliza un índice que identifica la posición de dicho elemento. El primer elemento se identifica con el índice 0, el segundo elemento con el índice 1 y así sucesivamente.

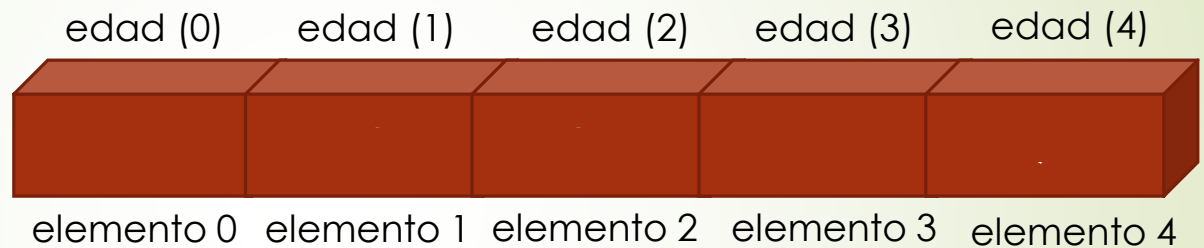
	0	1	2	3	4	5
Pares [6]	40	28	4	16	32	12

En c++: `int Pares[6];`

Arreglos unidimensionales

Edades
15
10
25
50
9

**Arreglo
edad**

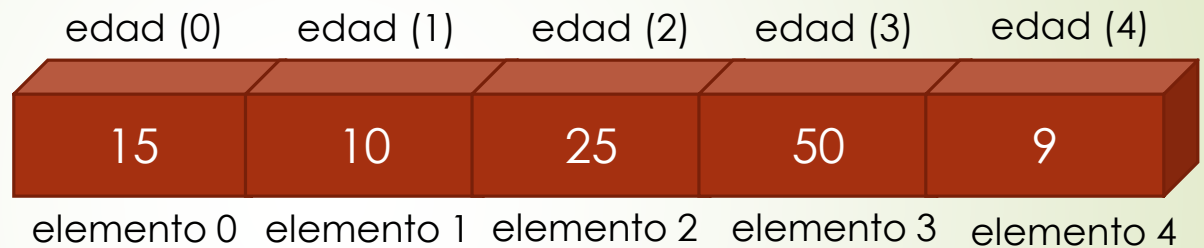


¿En qué posición del arreglo se encuentra la edad de 50 años?

Arreglos unidimensionales

Edades
15
10
25
50
9

**Arreglo
edad**



¿En qué posición del arreglo se encuentra la edad de 50 años?

ARREGLOS UNIDIMENSIONALES

- Para acceder a cada elemento del arreglo y almacenar el respectivo valor, se hace de la siguiente manera:

```
int Pares[6];  
Pares[0]=40;  
Pares[1]=28;  
Pares[2]=4;  
Pares[3]=16;  
Pares[4]=32;  
Pares[5]=12
```

Para obtener un valor del arreglo:

```
int Par;  
Par = Pares[3]
```

En la variable Par quedará almacenado el numero 16.

	0	1	2	3	4	5
PARES [6]	40	28	4	16	32	12

ARREGLOS UNIDIMENSIONALES

- Para leer el arreglo por teclado:

```
for(int i=0;i<6;i++)  
{  
    cout << "Introduzca un número par " << endl;  
    cin >> Pares[i];  
}
```

	0	1	2	3	4	5
PARES (5)	40	28	4	16	32	12

ARREGLOS UNIDIMENSIONALES

- Para recorrer todo el arreglo e imprimirlo:

```
int Pares(6);
```

```
Pares[0]=40;
```

```
Pares[1]=28;
```

```
Pares[2]=4;
```

```
Pares[3]=16;
```

```
Pares[4]=32;
```

```
Pares[5]=12
```

Imprimiendo el arreglo:

```
for(int i=0;i<6;i++)  
{  
    cout << Pares[i]<<endl;  
}
```

PARES [5]

0	1	2	3	4	5
40	28	4	16	32	12

¿Cómo declarar un arreglo?

- En c++ para declarar un arreglo se emplea la sintaxis:

Tipo identificador [tamaño] = {lista de inicialización} ;

- Tipo se refiere al tipo de dato que contendrá el arreglo
- Identificador es el nombre que se le dará al arreglo.
- Tamaño es opcional e indica el número de elementos que tendrá el arreglo. Si un arreglo se declara sin tamaño se deberá emplear la lista de inicialización en la declaración del arreglo.
- Lista de inicialización del arreglo. Valores iniciales que tendrá el arreglo declarado.



Ejemplos

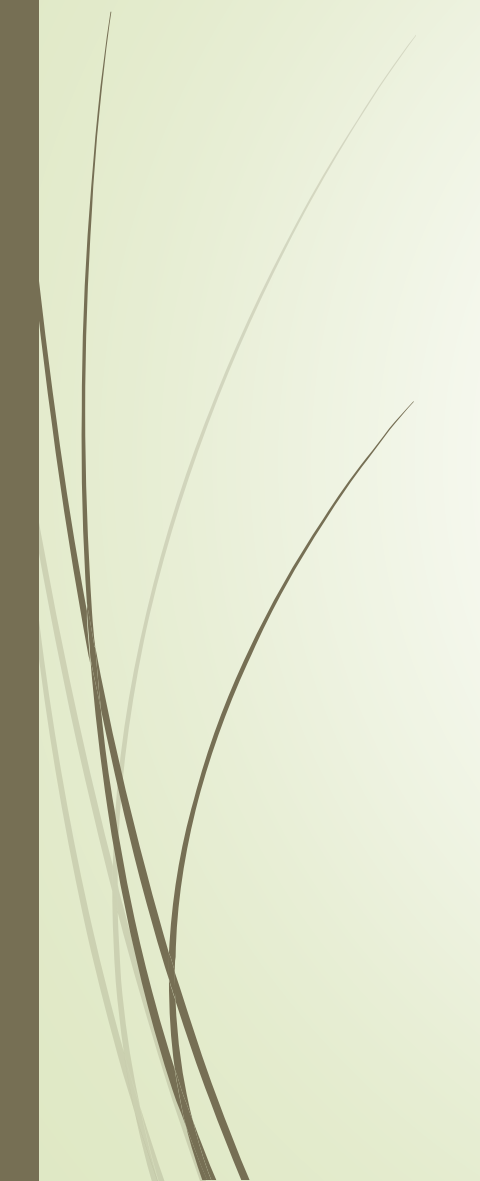

- `int num[5];`
- `long numB[5] = {1,2,3,4,5};`
- `char textoA[3]={'a','b','c'};`

Acceder a los Datos de un arreglo

- Para acceder a los datos de un arreglo o vector, debemos hacerlo mediante el **índice** o **número del elemento** del mismo. Por ejemplo:

```
int notas[5];           //Declaración del arreglo
notas[0]=9;             //Asignación de un valor en posición 0
notas[1]=8;
notas[2]=7;
cin >> notas[3];        //Asignación por usuario en posición 3
notas[4]=10;
cout << notas[5] << endl; //Mostrar en pantalla posición 5
cout << notas[4] << endl;
```

Ejemplo



```
int cantnotas=5;  
int notas[cantnotas];
```

```
//lectura de las 5 notas del arreglo  
for(int i=0;i<cantnotas;i++)  
{  
    cout << "Introduzca una Nota" << endl;  
    cin >> notas[i];  
}
```

```
//Impresión en pantalla de las 5 notas del arreglo  
for(int i=0;i<cantnotas;i++)  
{  
    cout << "Nota en posición " << i;  
    cout << " Es: " << notas[i]<<endl;  
}
```



Ejercicio

- Elaborar un programa que permita leer 10 números e imprimirlos en orden inverso a como fueron leídos.
- 

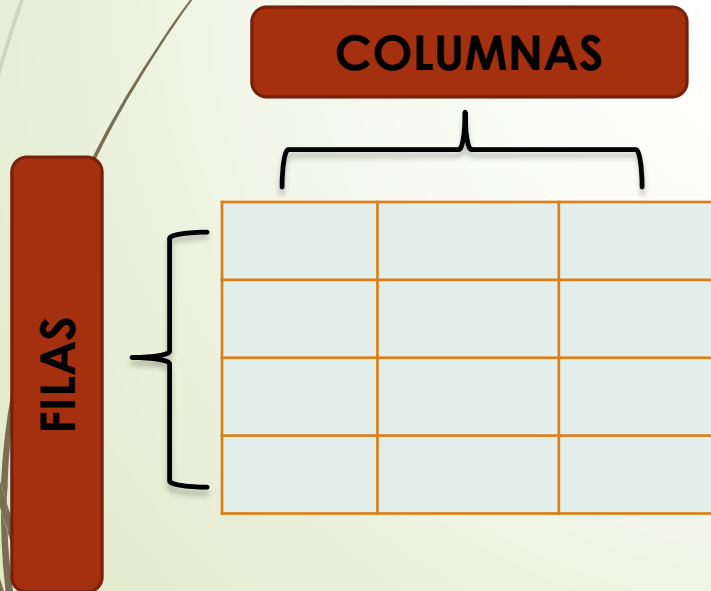
Inicialización de arreglos

- `int temp[5] = {98, 45, 34, 87, 78};`
- `char códigos[6] = {'m', 'u', 'e', 's', 't', 'r', 'a'};`
- `double pendientes[7] = {11.95, 6.34, 2.345, 3.2, 2.0};`

ARREGLOS MULTIDIMENSIONALES

- Son aquellos que constan de 2 o mas dimensiones. Los arreglos de 2 dimensiones también se conocen con el nombre de matriz, ya que forman una tabla compuesta por filas (Horizontales) y columnas (verticales). Los arreglos de 3 dimensiones forman un cubo.

```
int Pares[3][2];
```



	0	1	2
0			
1			
2			
3			

ARREGLOS MULTIDIMENSIONALES

- Para almacenar datos en cada elemento de la matriz:

Declaramos Matriz:

```
double Tabla[4][2];
```

Almacenamos valores:

```
Tabla[0][0]=0.0;
```

```
Tabla[0][1]=0.1;
```

```
Tabla[1][0]=1.0;
```

```
Tabla[1][1]=1.1;
```

```
Tabla[2][0]=2.0;
```

```
Tabla[2][1]=2.1;
```

```
Tabla[3][0]=3.0;
```

```
Tabla[3][1]=3.1;
```

	0	1
0		
1		
2		
3		

0.0	0.1
1.0	1.1
2.0	2.1
3.0	3.1

ARREGLOS MULTIDIMENSIONALES

- Para llenar matriz desde teclado:

Declaramos Matriz:

```
double tabla [4][2];
```

Almacenamos valores:

```
for(int fila=0;fila<4;fila++)  
{  
    for(int columna=0;columna<2;columna++)  
    {  
        cout << "Introduzca un numero" << endl;  
        cin >> tabla[fila][columna];  
    }  
}
```

	0	1
0		
1		
2		
3		

0.0	0.1
1.0	1.1
2.0	2.1
3.0	3.1

ARREGLOS MULTIDIMENSIONALES

➤ Imprimir matriz:

Declaramos Matriz:

```
double tabla [4][2];
```

Mostramos valores:

```
for(int fila=0;fila<4;fila++)  
{  
    for(int columna=0;columna<2;columna++)  
    {  
        cout << Tabla[fila][columna] << "\\t";  
    }  
    cout << endl;  
}
```

	0	1
0		
1		
2		
3		

0.0	0.1
1.0	1.1
2.0	2.1
3.0	3.1



Ejemplo

- Crear una matriz de 10 x 10 que permita capturar los datos y luego imprimirlos.
- 



Ejercicio

- Crear un programa que permita capturar una matriz de 10x10 y luego imprimir la suma de todos sus elementos.
- 