Arrays o Arreglos

¿Qué es un array?

Un **array** es una **estructura de datos** que almacena una **colección de elementos del mismo tipo**, en **posiciones contiguas de memoria**.

- Todos los elementos son del mismo tipo (ej: int, float, char, etc.).
- Cada elemento se accede mediante un **índice** (empezando en 0).
- El array tiene un **tamaño fijo**, definido al momento de su declaración.

```
int numeros[5]; // array de 5 enteros
```

¿Cómo se almacenan los arrays en memoria?

- Los elementos del array se almacenan en **espacios consecutivos** de memoria RAM.
- Cada celda ocupa un número fijo de bytes (depende del tipo de dato).
- Se puede acceder a cada elemento mediante el **nombre del array + índice**.

Ejemplo array **int numeros[4] = {10, 20, 30, 40};**

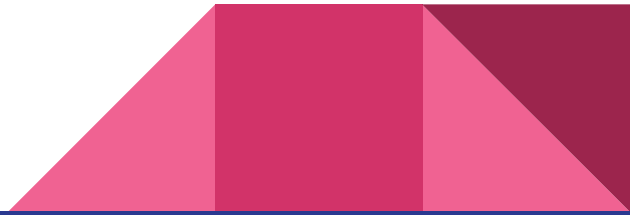
¿Cómo se almacenan los arrays en memoria?

Dirección de memoria	valor	Índice
0x100	10	numeros[0]
0x104	20	numeros[1]
0x108	30	numeros[2]
0x10C	40	numeros[3]

Si un int ocupa 4 bytes, cada celda está separada por 4 bytes.

Declaración acceso e inicialización

```
int edades[5];                    // declara un array de 5 enteros
edades[0] = 18;                   // asigna 18 al primer elemento
int x = edades[2];                // lee el valor del tercer elemento
int a[5] = {1, 2, 3, 4, 5};       // todos los elementos
int b[5] = {1, 2};                // el resto se llena con ceros
int c[ ] = {10, 20, 30};          // el tamaño se deduce automáticamente
```



¿Por qué los arrays son de tamaño fijo?

En C, los arrays no crecen ni se reducen dinámicamente por sí solos.El compilador necesita saber cuánto espacio reservar en memoria.
El tamaño es necesario para calcular la posición de cada elemento (base *primer elemento del array* + i * tamaño_del_tipo).

C no permite arrays con elementos de distintos tipos. El tipo define cómo se interpreta la memoria.

```
int enteros[4];       // solo enteros

char letras[10];      // solo caracteres

float decimales[3]; // solo floats
```

Cómo recorrerlos…

```
for (int i = 0; i < 5; i++) {
        printf("%d\n", edades[i]);
}
```

Recorre uno a uno los elementos de un array llamado edades y los imprime junto a un salto de línea.

Arrays multidimensionales (matrices)

C permite arrays de varias dimensiones, típicamente 2D (pueden ser más).

```
int matriz[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
};
```

En el primer corchete ponemos la cantidad de filas y en el segundo la cantidad de columnas

visualmente:
Fila 0: [1 2 3]

Fila 1: [4 5 6]

Cómo lo recorremos...

```
for (int i = 0; i < 2; i++) {                  //el for de afuera se mete en la fila...
        for (int j = 0; j < 3; j++) {          //el de adentro se mete en la columna...
        printf("%d ", matriz[i][j]);           //i es la fila j la columna...
        }
        printf("\n");                          //este salto de línea es clave
}
```

esto va a impimir nuestra matrix : 1 2 3

4 5 6

Arrays como argumentos de funciones

Cuando se pasa un array a una función, **se pasa la dirección (puntero) del primer elemento**. No se copia todo el array.

```
void imprimir(int arr[], int n) {       **fuera del main declaro mi función
                for (int i = 0; i < n; i++) {
                printf("%d\n", arr[i]);
                }
}
```

Llamada dentro del main:

```
int numeros[4] = {1, 2, 3, 4};
imprimir(numeros, 4);
```