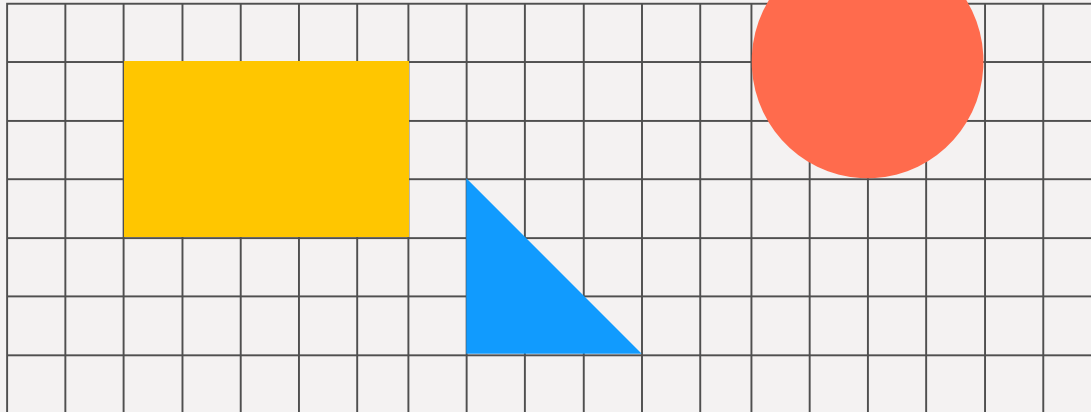


# Unidad 4

## Arreglos



# Arreglos Unidimensionales (Vectores)



Un arreglo (o array) es una colección de variables del mismo tipo almacenadas en posiciones contiguas de memoria.

Sirve para guardar muchos datos similares bajo un mismo nombre.

¿Cómo se declara un arreglo?

```
tipo nombre[tipo nombre[tamaño];
```

# ¿Cómo se asignan valores?

## Manualmente:

```
int numeros[3];  
numeros[0] = 10;  
numeros[1] = 20;  
numeros[2] = 30;
```

## Directamente:

```
int numeros[3] = {10,  
20, 30};
```

```
#include <stdio.h>
```

```
int main() {
```

```
    int numeros[5] = {10, 20, 30, 40, 50};
```

```
    // Recorrido: mostrar cada número
```

```
    for (int i = 0; i < 5; i++) {
```


```
        printf("Elemento %d: %d\n", i, numeros[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

## Arreglos Bidimensionales (Matrices)



Son como una tabla  
con filas y columnas.

```
#include <stdio.h>
```

```
int main() {
```

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

```
    // Recorrer la matriz
```

```
    for (int fila = 0; fila < 2; fila++) {  
        for (int col = 0; col < 3; col++) {  
            printf("%d ", matriz[fila][col]);  
        }  
        printf("\n");  
    }
```

```
    return 0;
```

```
}
```

## Operaciones con Arreglo: Recorrido



Usar un for para mostrar  
o procesar cada  
elemento.



A code editor window with a dark background and a white border. At the top left, there are three colored circles (red, yellow, green) representing window controls. The code is written in a monospaced font with syntax highlighting: keywords are blue, identifiers and literals are green, and string literals are red. The code defines an array of five integers and iterates through them, printing each element's index and value.

```
#include <stdio.h>
```

```
int main() {
```

```
    int numeros[5] = {10, 20, 30, 40, 50};
```

```
    // Recorrido: mostrar cada número
```

```
    for (int i = 0; i < 5; i++) {
```


```
        printf("Elemento %d: %d\n", i, numeros[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

## Operaciones con Arreglo: Búsqueda



usar un for para mostrar  
o procesar cada  
elemento.

```
#include <stdio.h>
```

```
int main() {  
    int numeros[5] = {3, 7, 9, 1, 4};  
    int buscado = 9;  
    int encontrado = 0;  
    for (int i = 0; i < 5; i++) {  
        if (numeros[i] == buscado) {  
            printf("Encontrado en la posición %d\n", i);  
            encontrado = 1;  
            break;  
        }  
    }  
    if (!encontrado) {  
        printf("No encontrado.\n");  
    }  
    return 0;  
}
```

# Operaciones con Arreglo: Inserción



En C los arreglos tienen tamaño fijo, así que podemos reemplazar valores, pero no agregar nuevos dinámicamente sin usar estructuras más avanzadas.

```
int numeros[] = {10, 20, 30, 40, 50};
int size = sizeof(numeros) / sizeof(numeros[0]);
// Imprime el arreglo original
printf("Arreglo original:\n");
for (int i = 0; i < size; i++) {
    printf("numeros[%d] = %d\n", i, numeros[i]);
}
// Recorre el arreglo y reemplaza el valor en la posición 2
for (int i = 0; i < size; i++) {
    if (i == 2) {
        numeros[i] = 100; // Reemplaza el valor en la posición 2
    }
}
// Imprime el arreglo modificado
printf("\nArreglo modificado:\n");
for (int i = 0; i < size; i++) {
    printf("numeros[%d] = %d\n", i, numeros[i]);
}
return 0;
```

`int size = sizeof(original) / sizeof(original[0]);`

**sizeof(original)** devuelve el tamaño total en bytes ocupado por el arreglo.

**sizeof(original[0])** devuelve el tamaño en bytes del primer elemento del arreglo (un entero en este caso).

Al dividir el tamaño total del arreglo entre el tamaño de un elemento, se obtiene el número de elementos que contiene el arreglo.