

CAPÍTULO 6	2
ARREGLOS	2
1) VECTORES (ARRAY).	2
a) Declaración de un vector.	2
b) Inicialización de un vector.	2
2) UTILIZACIÓN DE ARRAYS COMO PARÁMETROS.	4
BIBLIOGRAFÍA.	4

# CAPÍTULO 6

## ARREGLOS

### 1) Vectores (Array).

Un array es un conjunto ordenado de datos del mismo tipo. Los datos se llaman elementos o términos del array y su posición dentro del arreglo se enumera consecutivamente iniciando desde cero 0, 1, 2, 3... etc. [1]

El tipo de elementos almacenados en un array puede ser cualquiera de los tipos de dato válido de C.

Los arrays son una conveniente manera de agrupar muchas variables bajo un mismo nombre, estos arrays pueden ser unidimensionales como una lista o vector, bidimensionales como una tabla o matriz, tridimensionales como Cubo Rugby o un edificio de apartamentos; de hecho, pueden tener cualquier dimensión arbitraria. [2]

Es siguiente ejemplo muestra un arreglo unidimensional de 6 elementos de nombre "a":

a	25.1	34.5	5.25	7.45	6.09	7.54
	0	1	2	3	4	5

← Índices del arreglo

En el array del ejemplo el primer elemento del arreglo "a", **a[0]** contiene **25.1**, el segundo elemento **a[1]** contiene **34.5** y así sucesivamente.

También se puede observar que los índices de un array siempre tienen como límite inferior 0 y como límite superior el tamaño (número de elementos) del array menos 1.

#### a) Declaración de un vector.

Al igual que con cualquier tipo de variable los arreglos deben ser declarados antes de ser utilizados, la declaración es igual a la de cualquier variable, excepto que se debe indicar el tamaño o longitud del array. [1]

La sintaxis para declarar un array de una dimensión es:

```
tipo nombreArray [numeroDeElementos];
```

Así, la declaración de un array de diez variables enteras se escribe:

```
int numeros[10];
```

Esta declaración hace que el compilador reserve espacio de memoria suficiente para contener diez valores enteros. [1]

#### b) Inicialización de un vector.

Los arrays deben ser inicializados antes de ser utilizados, tal como sucede con cualquier variable. Para asignar valores a cada elemento del array, se puede escribir:

```
numeros[0] = 10;  
numeros[1] = 20;  
numeros[2] = 10;  
...
```

Otro método utilizado es inicializar el array completo en una sola sentencia. En este caso el tamaño del array se puede determinar automáticamente por el número de elementos en la inicialización.

```
int numeros [6] = {10, 20, 30, 40, 50, 60};
int numeros [ ] = {3, 4, 5};    /*Declara un arreglo de tres elementos*/
```

---

Nota: En C se puede declarar un arreglo con los corchetes vacíos [ ], sólo cuando se inicializan valores en la misma sentencia.

---

Para asignar valores a arrays de mayor tamaño, se utiliza la lazos **for** o **while/do-while**. Por ejemplo, para inicializar todos los valores de un array a cero se puede usar la siguiente sentencia:

```
for(i=0; i<6; i++){
    numeros[i] = 0;
}
```

como el valor del subíndice *i* varía de 0 a 5, cada elemento del array se establece en cero.

---

Nota: C no valida que los subíndices del arreglo estén dentro del rango declarado. Se debe tener cuidado con esta acción, debido a que se sobrescribirían datos o código. [1]

---

Ejemplo:

El siguiente programa lee desde el terminal ocho números, a continuación visualiza cada uno de los elementos del arreglo y la suma total de ellos.

```
#include<stdio.h>
#define NUM 8

int main (){
    int nums[NUM];
    int i, total=0;
    /*Leo los numeros*/
    for(i=0; i<NUM; i++){
        printf("Por favor, introduzca el numero: ");
        scanf("%d",&nums[i]);
    }
    /*Imprimo los numeros*/
    printf("\nLista de numeros: ");
    for(i=0; i<NUM; i++){
        printf("%d ", nums[i]);
        total += nums[i];
    }

    printf("\nLa suma de los numeros es: %d", total);
    return 0;
}
```

## 2) Utilización de arrays como parámetros.

En C cuando se pasa un array como parámetro, en realidad se pasa una referencia a la dirección de memoria donde inicia el array. Esto significa que cuando se llama a una función que utiliza un array como parámetro, se debe tener cuidado al modificar el array en la función llamada. Como ejemplo crearemos el prototipo de una función que suma los elementos de un arreglo, llamada "sumaDeDatos".

```
int sumaDeDatos(int datos[],int n);
```

En el prototipo se pasan dos argumentos `int datos[ ]` para la referencia al vector e `int n` para el número de elementos del arreglo.

Ejemplo:

El siguiente programa, lee desde consola 5 números usando la función leer y los guarda en un arreglo, luego imprime los datos usando la función imprimir.

```
#include<stdio.h>
#define NUM 5

void leer (int arreglo[],int n);
void imprimir (int arreglo[],int n);

int main (){
    int nums[NUM], i, total=0;
    leer(nums,NUM);
    imprimir(nums,NUM);
    return 0;
}

void leer (int arreglo[],int n){
    int i;
    for(i=0; i<n; i++){
        printf("Por favor, introduzca un numero: ");
        scanf("%d", &arreglo[i]);
    }
}

void imprimir (int arreglo[],int n){
    int i;
    printf("\nLista de numeros: ");
    for(i=0; i<n; i++){
        printf("%d ", arreglo[i]);
    }
}
```

## Bibliografía.

- [1] Luis Joyanes Aguilar and Ignacio Zahonero Martínez, *Programación en C, Metodología, algoritmos y estructuras de datos*, Segunda Edición. Mc Graw Hill.
- [2] Mark Burgess, *The GNU C Programming Tutorial*, Ed. 4.1. Faculty of Engineering, Oslo College.