**Arreglo**

Un arreglo es un conjunto o colección finita de datos de un mismo tipo. Los elementos de un arreglo pueden ser accedidos por medio de un subíndice **i**.

Podemos considerar a un arreglo desde el punto de vista matemático como un vector, y a un arreglo bidimensional una matriz.

**Declaración de un arreglo**

Un arreglo se define indicando el tipo de arreglo, es decir, el tipo de datos de todos los elementos del arreglo, luego se le da un nombre al arreglo y finalmente se le da un tamaño.

**<tipo>** nombreArreglo**[**Tamaño**]**

Por ejemplo tengo un arreglo de números enteros:

**int** arreglo[4];

En el caso anterior el tipo del arreglo es entero (**int**).  
Se le da una dimensión al arreglo que va entre los caracteres '[' y ']' , en el caso anterior la dimensión es 4, esto quiere decir que en la memoria se reservaron 4 posiciones para almacenar 4 valores enteros.

**Inicializar un Arreglo**

Existen varias maneras de inicializar un arreglo, una manera muy sencilla es poner entre llaves (**{ }**),  los elementos del arreglo separados por comas.

arreglo = {51, 60, 70, 95};

También podemos utilizar la estructura de control **for** para inicializar los elementos del arreglo como se ve en el Ejemplo 7.1

**Acceso a los elementos de un Arreglo**

Puedo acceder a un elemento por medio de un **subíndice**, por ejemplo si yo quiero acceder al primer elemento tendré que hacerlo de esta manera  
    int nro = arreglo[0];  
En la variable nro se almacenara el valor de 51, para acceder al segundo valor:  
    nro = arreglo[1];  
En la variable nro se almacenará el valor de 60, y así sucesivamente con los demás elementos.

|  |  |
| --- | --- |
| arreglo[0] | 51 |
| arreglo[1] | 60 |
| arreglo[2] | 70 |
| arreglo[3] | 95 |

Si nos damos cuenta tener un arreglo es mucho más ventajoso que tener definidas 4 variables.

*Desventajas*. En ocasiones, no podemos predecir con precisión el tamaño que un arreglo tendrá por lo que podemos definir un arreglo muy grande, lo que nos lleva a desperdiciar memoria, por ejemplo si defino un arreglo de 100 elemento (arreglo[100]), y sólo utilizo 5 posiciones de memoria, las demás 95 estarán desperdiciadas, la solución a este problema la veremos más adelante en lo que se denomina punteros.

**Ejemplo 7.1**

Se desea ingresar las notas finales de 10 alumnos de la materia de Introducción a la programación, para luego emitir un reporte del promedio de todas las notas.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #include <stdio.h>  main()  {  const int TAM = 5;  int i;  float promedio=0;  int arreglo[TAM];  for(i=0; i < TAM ; i++)  {  printf("Ingrese la nota del estudiante # %d: ",i+1);  scanf("%d", &arreglo[i]);  }  printf("\nLas calificaciones son: ");  for(i=0; i < TAM ; i++)  {  printf("%d\t", arreglo[i]);  promedio = promedio + arreglo[i];  }  promedio = promedio / TAM ;  printf("\nEl promedio de las notas es: %.1f",promedio);  getchar();  getchar();  } | | | | | | | | |
|  | | | | | | | | |
|  |  | |  | |  | |  | |
| 0 | | 1 | | 2 | | 3 | | 4 |
|  | | | | | | | | |

No hubiera sido muy práctico manejar 10 variables diferentes para guardar las notas de los 10 alumnos, utilizando un arreglo se nos simplifican mucho las cosas.

N**ota:** Se puede declarar un arreglo dándole la dimensión o tamaño por un valor constante

**const** **int** TAM=10;  
**int** arreglo[TAM];

**Ejercicio:**

Pedir al usuario 5 números enteros y guardarlos en un vector. Al final, imprimir el arreglo e indicar cuál es el número mayor de ese arreglo y en qué posición se encuentra.

* **Referencias:**

<http://www.omijal.org/pagina_c/matrix.html>

<http://c.conclase.net/curso/?cap=010>