INSTITUTO TECNOLÓGICO BELTRÁN

Centro de Tecnología e Innovación

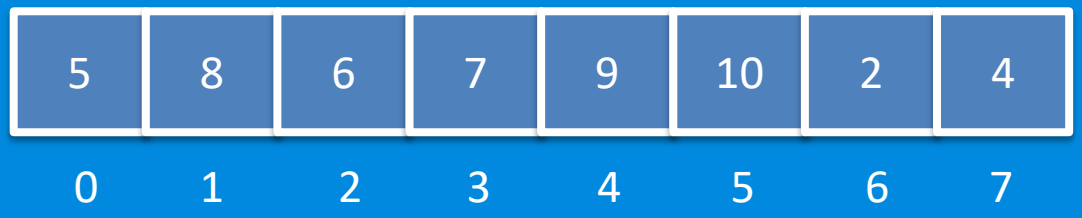
ARRAY (ARREGLO) UNIDIMENSIONAL: VECTORES

Es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales. Están formados por un conjunto de elementos de un mismo tipo de datos que se almacenan bajo un mismo nombre, y se diferencian por la posición que tiene cada elemento dentro del arreglo de datos. Al declarar un arreglo, se debe inicializar sus elementos antes de utilizarlos. Para declarar un arreglo tiene que indicar su tipo, un nombre único y la cantidad de elementos que va a contener.

ARRAY (ARREGLO) UNIDIMENSIONAL: VECTORES

Es decir, que estará conformado por un conjunto finito y ordenado (Cada elemento del arreglo puede ser identificado) de elementos homogéneos (Son del mismo tipo de dato).

El tipo más simple de arreglo es el unidimensional o vector (matriz de una dimensión). Por ejemplo, podríamos definir un vector para almacenar las notas de un alumno con un tamaño de ocho elementos:



Los subíndices del 0 al 7 nos indican la posición del elemento. El primer elemento es el 5, el segundo es el 8, el tercero es el 6 y así sucesivamente hasta llegar a la posición 7 cuyo valor es 4.

ESTRUCTURA Y USO

La sintaxis para la creación de este tipo de dato es la siguiente:

```
tipoDeElementosDelArray  nombreDelArray  [numeroElementos];
```

Por ejemplo:

```
int notas [8];
```

Esto declara que se crea un vector de enteros que contendrá 8 valores de tipo int. El número en la declaración es 8, pero el elemento notas[8] no existirá.

¿POR QUÉ?

Porque en C, al igual que en otros lenguajes de programación, la numeración de elementos empieza en cero y no en uno. De esta manera al indicar un 8, los índices de elementos en el array serán del 0 al 7. Es decir, si indicamos 8 el array tendrá 8 elementos (índices 0 a 7).

EJEMPLOS DE DECLARACIONES

```
int notas[8]; /* almacena ocho notas */
char nombre[21]; /* almacena nombres de largo menor o igual a 20 */
int multiplos[n]; /* donde n tiene un valor, declara un arreglo de tamaño n*/
```

Los índices pueden ser cualquier expresión entera. Si un programa utiliza una expresión como subíndice esta se evalúa para determinar el índice. Por ejemplo si a=5 y b=10, el enunciado *arreglo2[a+b] = 100*, asigna el valor 100 al elemento del arreglo número 15.

Los arreglos pueden ser declarados para que contengan distintos tipos de datos. Por ejemplo un arreglo del tipo char puede ser utilizado para almacenar una cadena de caracteres.

VISUALIZACIÓN DE DATOS

```
#include <stdio.h>
#include <stdlib.h>
int main() {
        int notas [8];

        notas[0] = 5;

        printf ("Nota 1: %d \n", notas[0]);
        printf ("Nota 2: %d \n", notas[1]);
        printf ("Nota 3: %d \n", notas[2]);
        printf ("Nota 4: %d \n", notas[3]);
        printf ("Nota 5: %d \n", notas[4]);
        printf ("Nota 6: %d \n", notas[5]);
        printf ("Nota 7: %d \n", notas[6]);
        printf ("Nota 8: %d \n", notas[7]);

        return 0;
}
```

RESULTADO

```
Nota 1: 5
Nota 2: 0
Nota 3: 46
Nota 4: 0
Nota 5: 0
Nota 6: 0
Nota 7: 1
Nota 8: 0
```

VISUALIZACIÓN DE DATOS (INICIALIZACIÓN)

¿Por qué ocurre esto? Realmente no siempre ocurrirá esto, podríamos decir que el resultado puede cambiar dependiendo del compilador. Lo que es cierto que es notas[0] vale 5 porque así lo hemos declarado. Sin embargo, es posible que nos aparezcan valores aparentemente aleatorios para el resto de elementos del array porque no los hemos inicializado. El compilador al no tener definido un valor específico para la inicialización les ha asignado unos valores aparentemente aleatorios. Para evitar esta circunstancia tendremos que inicializar todos los elementos de un array antes de utilizarlo. Si bien lo podemos realizar con una asignación manual por cada índice, también podemos realizarlo por medio de un bucle:

```
for (i = 0; i< 8; i++) {
     notas[i] = 0;
 }
```

Si no se inicializa explícitamente el array no se puede estar seguro del valor que contienen los elementos del mismo.

VISUALIZACIÓN DE DATOS (INICIALIZACIÓN)

Los elementos de un arreglo pueden ser inicializados en la declaración del arreglo haciendo seguir a la declaración un signo de igual y una lista entre llaves de valores separados por comas:

```
int n[10]={32, 27, 64, 18, 95, 24, 90, 70, 8, 3};
```

Si en la declaración hay menos inicializadores que el tamaño del array, los elementos son inicializados a cero. Puedo entonces inicializar todo un array en 0 con la declaración:

```
int n[10]={0};
```

Declarar más inicializadores que el tamaño del arreglo es un error de sintaxis. Si en una declaración con una lista inicializadora se omite el tamaño del arreglo el numero de elementos del arreglo será el número de elementos incluidos en la lista inicializadora:

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

No se puede asignar un arreglo en otro, se tiene que copiar posición a posición.

OPERACIONES FRECUENTES

RECORRIDO DE UN ARREGLO DE TAMAÑO N

El índice puede ir del primero al último elemento:

```
for (i=0; i < n; i++) {
    //proceso
}
```

El índice puede ir del último al primer elemento:

```
for (i=n-1; i >=0; i--) {
    //proceso
}
```

CÓMO PASAR ARREGLOS A FUNCIONES

```
void pCargarArreglo (int a[], int n) {
        int i;
        for (i=0;i < n;i++) {
                printf("Ingrese elemento :");
                scanf (" %d", &a[i]);
                printf("\n");
        }
}
```

Invocamos al procedimiento anterior pasando como argumento el nombre del arreglo y su tamaño, por ejemplo: *pCargarArreglo(array2, TAMANIO).*

Estudiemos la función: Recibe como parámetros un arreglo (parámetro a) y un entero (parámetro n) que representa el tamaño del arreglo.

En el lenguaje C se pasa de forma automática los arreglos a las funciones por referencia (cualquier modificación que realice en la función al array tendrá efecto en el array que se pasa como parámetro).

INSTITUTO TECNOLÓGICO BELTRÁN

Centro de Tecnología e Innovación