

PROGRAMACIÓN

Unidad 5 - Parte 1: Tipos de datos derivados: arreglos. Arreglos unidimensionales. Arreglos numéricos.

Repasemos lo visto



Tipos de datos derivados



“Existe una cantidad conceptualmente infinita de tipos de datos derivados, formados a partir de los tipos de datos simples”.

Brian Kernighan y Dennis Ritchie

Estructuras de Datos Clasificación

Según el tipo de la información que contienen

Estructuras de datos
homogéneas

Estructuras de datos
heterogéneas

Estructuras de Datos Clasificación



Según la asignación de memoria

Estructuras de datos
estáticas

Estructuras de datos
dinámicas

En el lenguaje C se predefinen los siguientes tipos de datos derivados:



- ❑ Arreglos
- ❑ Apuntadores o punteros
- ❑ Estructuras
- ❑ Uniones

Arreglos



- “Un arreglo es una colección de variables ordenadas e indexadas, todas de idéntico tipo que se referencian usando un nombre común”.
- “El array (también conocido como arreglo, vector o matriz) permite trabajar con una gran cantidad de datos del mismo tipo bajo un mismo nombre o identificador.”

Características de los arreglos

- ❑ Es un tipo de dato homogéneo.
- ❑ Es una estructura indexada.
- ❑ Es un tipo de dato estático.
- ❑ Los arreglos usan un espacio de almacenamiento contiguo.
- ❑ El nombre del arreglo es una referencia a la dirección de la 1° componente

Un arreglo puede ser:



- ❑ De una dimensión (un índice)
- ❑ De varias dimensiones (varios índices)
- ❑ Las componentes del arreglo pueden ser de cualquier tipo: caracteres, enteros, reales, punteros , estructuras, arreglos.

Arreglos de una dimensión

“ Un vector o arreglo lineal es un tipo de dato arreglo con un índice, es decir, con una dimensión”.

- El acceso a las componentes de un arreglo se puede hacer en forma directa, a través del nombre del arreglo y la notación de subíndice que indica la posición .

A tener en cuenta

Un arreglo de una dimensión con los 5 primeros números pares positivos, tiene la forma:

pares	2	4	6	8	10
--------------	----------	----------	----------	----------	-----------

El acceso directo, vía nombre y subíndice :

pares[0] = 2 pares[1] = 4 pares[2] = 6

pares[3] = 8 pares[4] = 10.

¿Cómo se declara un arreglo de una dimensión?

Forma General: **tipo** **nombre**[**tamaño**];

nombre: cualquier identificador válido.

tipo: puede ser un tipo aritmético (*int*, *float*, *double*), *char*, puntero, estructura, etc.

tamaño : expresión constante entre corchetes que define cuantos elementos guardará el arreglo.

A tener en cuenta...



Un arreglo siempre se declara incluyendo entre los corchetes el máximo número de elementos, salvo que inicialice el arreglo al mismo tiempo.

Uso de constantes enteras

Definición de
una constante
TAMA

```
1  #include <stdio.h>
2  #define TAMA 20
3
4  int main()
5  {
6      int arreglo[TAMA];
7      ...
```

Declaración de
un arreglo



Importante!



- ❑ En C no se puede operar (comparar, sumar, restar, etc) con todo un vector o toda una matriz como una única entidad
- ❑ Hay que tratar sus elementos uno a uno por medio de bucles for o while

Operaciones

Los elementos de un vector se utilizan en las expresiones de C como cualquier otra variable.

Ejemplos:

$$a[5] = 0.8;$$

$$a[9] = 30. * a[5];$$

$$a[0] = 3. * a[9] - a[5] / a[9];$$

$$a[3] = (a[0] + a[9]) / a[3];$$

Del tamaño del arreglo



- ❑ La dimensión del arreglo debe ser establecida a priori.
- ❑ El tamaño o dimensión del arreglo debe ser una expresión constante.
- ❑ La declaración de un arreglo, es una reserva de memoria, por lo tanto, el tamaño no puede ser una variable ni una expresión variable

Recomendación

- ❑ Usar constantes definidas con `#define` para indicar el tamaño.
 - ▣ Facilita , ordena y organiza los datos.
 - ▣ Asegura que las referencias posteriores al arreglo (por ejemplo en un lazo de lectura) no podrán superar el tamaño especificado.
 - ▣ Superar el tamaño máximo declarado puede generar situaciones irregulares con resultados perjudiciales.

Ejemplos de Arreglos

```
1  #include <stdio.h>
2  #define TAMA 3
3
4  int main() {
5      |
6      int arre1[5];
7      int arre2[TAMA];
8      int arre3[] = {0,1,2,3,4};
9
10     return 0;
11 }
12
```

Un arreglo siempre se declara incluyendo entre los corchetes el máximo número de elementos, salvo que inicialice el arreglo al mismo tiempo.

La declaración que sigue..... ¿es correcta?

int arreglo[];

Acceso al contenido de un arreglo

Para acceder a uno de los elementos del arreglo en particular, basta con invocar el nombre del arreglo y especificar entre corchetes del elemento.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int arre3[] = {2,4,6,8,10};
6      printf("arre3[3]= %d", arre3[3]);
7      printf("arre3[3]= %d", arre3[3]);
8      return 0;
9  }
```

Por ejemplo, si se quiere acceder al cuarto elemento del arreglo `arre3`, se invocaría de la siguiente manera: `arre[3]`. Recuerde que el arreglo almacena desde la casilla 0. Por tanto, en un arreglo de 20 casillas, éstas están numeradas del 0 al 19.

¿Qué numero me mostrara por pantalla?

`arre3[3] = 8`

Arreglos

```
1  #include <stdio.h>
2  #define TAMA 20
3
4  int main() {
5
6      int arre2[TAMA], j=0;
7
8      printf("Ingrese un 20 numeros enteros:");
9
10     for(int i=0; i<TAMA; i++)
11     {
12         printf("\n arre2[%d]:", i);
13         scanf("%d", &arre2[i]);
14     }
15
16     printf("El arreglo ingresado es: ");
17     while(j<TAMA)
18     {
19         printf("\n arre2[%d]=%d", j, arre2[j]);
20         j++;
21     }
22     return 0;
23 }
```

En C no se puede operar con todo un vector o toda una matriz como una única entidad.

Hay que tratar sus elementos uno a uno mediante bucles como for() y while().

Acceso al contenido de un arreglo

```
1  #include <stdio.h>
2  #define TAMA 20
3
4  int main() {
5
6      int arre2[TAMA], j=0;
7
8      printf("Ingresa un 20 numeros enteros:");
9
10     for(int i=0; i<5; i++)
11     {
12         printf("\n arre2[%d]:", i);
13         scanf("%d", &arre2[i]);
14     }
15
16     printf("El arreglo ingresado es: ");
17     while(j<TAMA)
18     {
19         printf("\n arre2[%d]=%d", j, arre2[j]);
20         j++;
21     }
22     return 0;
23 }
```

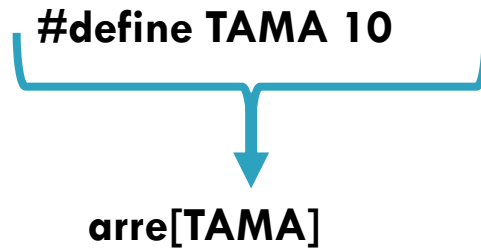
¿Qué error encuentra en el código?

El arreglo ingresado es:

```
arre2[0]=1
arre2[1]=2
arre2[2]=3
arre2[3]=4
arre2[4]=5
arre2[5]=0
arre2[6]=16
arre2[7]=0
arre2[8]=0
arre2[9]=0
arre2[10]=0
arre2[11]=0
arre2[12]=8
arre2[13]=0
arre2[14]=4199705
arre2[15]=0
arre2[16]=8
arre2[17]=0
arre2[18]=22
arre2[19]=0
```

Tamaño de un Arreglo

`#define TAMA 10`



`arre[TAMA]`

2	4	6	8	10	#\$'%"	12548	/&\$#"	985	!°#&%"
arre[0]	arre[1]	arre[2]	arre[3]	arre[4]	arre[5]	arre[6]	arre[7]	arre[8]	arre[9]

Tamaño de un Arreglo

A costa de la **no verificación de los límites del arreglo**, los arreglos en C tienen un código ejecutable rápido.

Se debe **ser responsable en la administración del espacio de memoria** cuando se trabaja con arreglos.

```
1  #include <stdio.h>
2  #define LIMITE 10
3  #define PROBLEMA 500
4
5  int main(void)
6  {   int indice, arreglo[LIMITE];
7
8      for(indice=0; indice < PROBLEMA; indice++)
9      {
10         arreglo[indice]=indice;
11     }
12
13     return 0;
14 }
```


Arreglo de más de una dimensión

Como se declara un arreglo de más de una dimensión?

Forma General: **tipo** **nombre**[tam1][tam2]...[tamN];

La diferencia principal es que se necesitarán múltiples índices para acceder a los elementos en las diferentes dimensiones del arreglo.

Ejemplo

Veamos un ejemplo de un arreglo de dos dimensiones



```
#include <stdio.h>
#define FILAS 5
#define COLUMNAS 4

int main()
{
    int matriz[FILAS][COLUMNAS];

    // Llena la matriz con algunos valores
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            matriz[i][j] = (i * FILAS) + j + 1;
        }
    }

    // Muestra la matriz
    printf("Matriz:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Ejemplo

Como sería el código si trabajamos con un arreglo de 3 dimensiones?

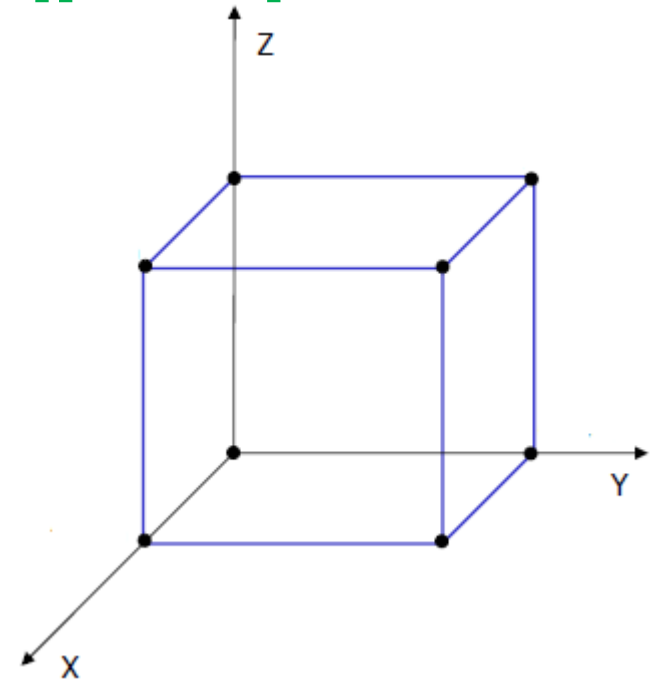
`arreglo[indice1][indice2][indice3]`

Debemos tener en cuenta que:

1° índice: sería la profundidad.

2° índice: la cantidad de filas.

3° índice: la cantidad de columnas.



Tarea: cargar y mostrar como queda un arreglo de 3 dimensiones

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

```
int main()
```

```
{
```

```
    printf("Hasta la próxima clase!!\n");
```

```
    return 0;
```

```
}
```