

# PROGRAMACIÓN

## Unidad 5: Tipos de datos derivados - Arreglos

Lic. Mariela A. Velázquez

# Estructuras de Datos Clasificación

**Según el tipo de la información que contienen**

Estructuras de datos  
homogéneas

Estructuras de datos  
heterogéneas

**Según la asignación de memoria**

Estructuras de datos  
estáticas

Estructuras de datos  
dinámicas

# 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 o vector) 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<sup>o</sup> 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.

# A tener en cuenta

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

<b>pares</b>	<b>2</b>	<b>4</b>	<b>6</b>	<b>8</b>	<b>10</b>
--------------	----------	----------	----------	----------	-----------

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**];

- **tipo**: puede ser un tipo aritmético (*int, float, double*), *char*, puntero, estructura, etc.
- *nombre*: cualquier identificador válido.
- **tamaño** : expresión constante entre corchetes que define cuantos elementos guardará el 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



# 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, una reserva de memoria, por lo tanto, el tamaño no puede ser una variable ni una expresión variable

# Arreglos

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

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()`.

# Ejemplos de Arreglos

```
1
2  #include <stdio.h>
3  #define TAMA 3
4
5  int main() {
6
7      int arre1[5];
8      int arre2[TAMA];
9      int arre3[] = {0,1,2,3,4};
10
11      return 0;
12  }
13
```

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
2  #include <stdio.h>
3  #define TAMA 3
4
5  int main()
6  {
7      int arre3[] = {2,4,6,8,10};
8      printf( "arre3[3] = %d", arre3[3]);
9      return 0;
10 }
11
```



¿Qué numero me  
mostrara por  
pantalla?

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.

→ **arre3[3] = 8**

# Acceso al contenido de un arreglo

```
1
2  #include <stdio.h>
3  #define TAMA 20
4
5  int main()
6  {
7
8      int arre2[TAMA];
9
10     printf("Ingrese un 5 numeros enteros:");
11
12     for(int i=0; i<5; i++)
13     {
14         printf("\n arre2[%d]:", i);
15         scanf("%d", &arre2[i]);
16     }
17
18     printf("El arreglo ingresado es: ");
19     for(int j = 0; j<TAMA; j++)
20     {
21         printf("\n arre2[%d]=%d", j, arre2[j]);
22     }
23     return 0;
24 }
25
```

- ¿Qué error encuentra en el código?

```
Ingrese un 5 numeros enteros:
arre2[0]:1

arre2[1]:2


arre2[2]:3

arre2[3]:4

arre2[4]:5
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]=67
arre2[19]=0
```

# Arreglo

**#define TAMA 10**



**arre[TAMA]**

<b>2</b>	<b>4</b>	<b>6</b>	<b>8</b>	<b>10</b>	<b>#\$"%</b>	<b>1254</b> <b>8</b>	<b>/&amp;\$#</b> <b>"</b>	<b>985</b>	<b>!°#&amp;</b> <b>%</b>
<b>arre[0]</b>	<b>arre[1]</b>	<b>arre[2]</b>	<b>arre[3]</b>	<b>arre[4]</b>	<b>arre[5]</b>	<b>arre[6]</b>	<b>arre[7]</b>	<b>arre[8]</b>	<b>arre[9]</b>

# 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
2  #include <stdio.h>
3  #define LIMITE 10
4  #define PROBLEMA 500
5
6  int main()
7  {   int indice, lio[LIMITE];
8
9      for(indice=0; indice < PROBLEMA; indice++)
10     {
11         lio[indice]=indice;
12     }
13
14     return 0;
15 }
16
```

# Inicialización de Arreglos

Hay 3 formas de inicializar un arreglo:

- Por omisión.
- Por inicialización explícita.
- En tiempo de ejecución.



# Por omisión

```
1
2  #include <stdio.h>
3  #define TAMA 5
4  int arre2[TAMA];
5
6  int main()
7  {
8      int arre1[TAMA];
9
10     printf("El arreglo declarado de manera global es: ");
11     for(int j = 0; j<TAMA; j++)
12     {
13         printf("\n arre2[%d]=%d", j, arre2[j]);
14     }
15
16     printf("\nEl arreglo declarado local a main(): ");
17     for(int j = 0; j<TAMA; j++)
18     {
19         printf("\n arre1[%d]=%d", j, arre1[j]);
20     }
21     return 0;
22 }
```

Un arreglo declarado en forma global, se inicializa por omisión (por default), en ceros binarios, a menos que se indique lo contrario.

```
El arreglo declarado de manera global es:
arre2[0]=0
arre2[1]=0
arre2[2]=0
arre2[3]=0
arre2[4]=0
El arreglo declarado local a main():
arre1[0]=8
arre1[1]=0
arre1[2]=67
arre1[3]=0
arre1[4]=1862784
```

# Por inicialización explícita

```
1
2  #include <stdio.h>
3  #define TAMA 3
4
5  int main()
6  {
7      float arre1[] = {78.6, 98.9, 45.7 };
8      int arre3[] = {2,4,6};
9
10     printf("El arreglo 3 es: ");
11     for(int j = 0; j<TAMA; j++)
12     {
13         printf("\n arre3[%d]=%d", j, arre3[j]);
14     }
15
16     printf("\nEl arreglo 1: ");
17     for(int j = 0; j<TAMA; j++)
18     {
19         printf("\n arre1[%d]=%.1f", j, arre1[j]);
20     }
21     return 0;
22 }
```

En la declaración, se asignan valores, según la siguiente norma: los valores a ser asignados a los elementos del arreglo deben estar encerrados entre llaves y separados por comas.

```
El arreglo 3 es:
arre3[0]=2
arre3[1]=4
arre3[2]=6
El arreglo 1:
arre1[0]=78.6
arre1[1]=98.9
arre1[2]=45.7
```

# En tiempo de Ejecución

```
1
2  #include <stdio.h>
3  #define TAMA 5
4
5  int main()
6  {
7
8      float arre1[TAMA];
9
10     printf("Ingresar 5 numeros reales: ");
11     for(int j = 0; j<TAMA; j++)
12     {
13         printf("\n arre1[%d]", j);
14         scanf("%f", &arre1[j]);
15     }
16
17     printf("\nEl arreglo 1: ");
18     for(int j = 0; j<TAMA; j++)
19     {
20         printf("\n arre1[%d]=%.2f", j, arre1[j]);
21     }
22
23     return 0;
24 }
25
```

El caso mas común.

Las componentes del arreglo tomarán valores que son leídos desde algún dispositivo de entrada ó sino valores generados por el mismo programa.

```
1
2  #include <stdio.h>
3  #define TAMA 5
4
5  int main()
6  {
7      float arre1[TAMA];
8
9      printf("Ingresar 5 numeros reales: ");
10     for(int j = 0; j<TAMA; j++)
11     {
12         arre1[j] = j * 2;
13     }
14
15     printf("\nEl arreglo 1: ");
16     for(int j = 0; j<TAMA; j++)
17     {
18         printf("\n arre1[%d]=%d", j, arre1[j]);
19     }
20
21     return 0;
22 }
```

# Cadenas -Arreglos de caracteres

- Este es el tipo de arreglo mas popular en código C.
- La ultima celda del vector de caracteres se reserva para el carácter que marca el fin de la cadena, que es el carácter nulo: “\0”.

# Inicialización explícita de una cadena

Los valores a asignar deben estar encerrados entre llaves y separados por comas (lista).

Formato de lectura  
“ %s”, indicado para  
leer/escribir arreglos  
de caracteres.

```
1
2  #include <stdio.h>
3  #define MAX 45
4
5  int main(void)
6  {
7      char apellido[MAX] = {'P','e','r','e','z','\0'};
8      char nombre[MAX] = "Marcelo";
9      char unt[12] = {'U','n','i','v','e','r','s','i','d','a','d'};
10
11     /* Mostramos por pantalla el contenido de los arreglos */
12
13     printf("%s \n", apellido);
14     printf("%s \n", nombre);
15     printf("%s \n", unt);
16
17     return 0;
18 }
```

# Funciones para la entrada -salida de cadenas interactiva

Prototipos de las función para la entrada:

**gets(arre):** almacena datos ingresados desde stdin a la cadena denominada arre. Un carácter ingresado \n de nueva línea se convierte en un cero de terminación (\0)

**puts(arre):** encamina la cadena arre hacia stdout. Un cero de terminación (\0) al final de la cadena se convierte en un carácter de nueva línea (\n).

```
1
2  #include <stdio.h>
3  #define MAX 45
4
5  int main(void)
6  {
7      char apellido[MAX];
8
9      /*Almacenamos datos en el arreglo */
10     gets(apellido);
11     /* Mostramos los datos cargados en el arreglo */
12     puts(apellido);
13
14     return 0;
15 }
```

# Funciones y Arreglos

```
1
2  #include <stdio.h>
3  #define TAMA 10
4
5  void Inicializar(int arre[TAMA] , int n);
6  void CargarArreglo(int arre[TAMA], int n);
7
8  int main()
9  {
10     int arre[TAMA];
11     int n;
12
13     printf("Ingrese la cantidad de Elementos (<10):");
14     scanf("%d", &n);
15
16     Inicializar(arre, n);
17     CargarArreglo(arre, n);
18
19     for(int i=0; i< n; i++)
20     {
21         printf("\n arre[%d] = %d", i, arre[i]);
22     }
23
24     return 0;
25 }
```

```
28
29 void Inicializar(int arre[TAMA] , int n)
30 {   int i;
31     for(i=0; i<n; i++)
32     {   arre[i]=0;   }
33 }
34
35
36 void CargarArreglo(int arre[TAMA], int n)
37 {
38     int i;
39     for(i=0; i<n; i++)
40     {   arre[i]= i * 3; }
41 }
42
```

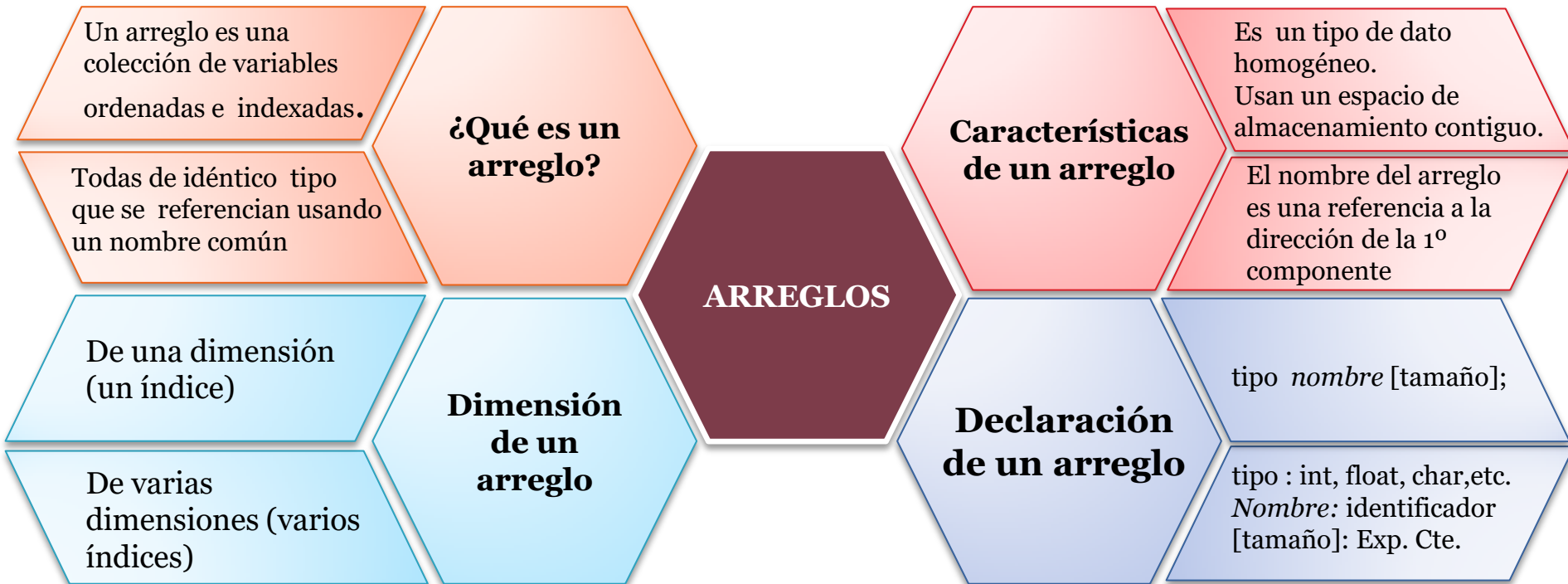
# A modo de resumen

Los arreglos, como una unidad, no admiten:

- Asignación directa entre ellos
- Operaciones aritméticas directas
- Comparaciones directas
- Devolución como valor de devolución de una función



# Temas vistos hasta acá



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

```
int main()
```

```
{
```

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

```
    return 0;
```

```
}
```