

# PROGRAMACIÓN

## Unidad 5: Tipos de datos derivados - Arreglos

Lic. Mariela A. Velázquez

# 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 código C estándar se predefinen los siguientes tipos de datos derivados:

- Arreglos
- **Funciones**
- 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<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.



# 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:

<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 trabaja con el tipo arreglo en diseño?

Ejemplo: Realizar un algoritmo que cuenta la cantidad de cada dígito que hay en una frase.

Algoritmo contadorDigito

ENTRADA: frase: arreglo de caracteres. MF= '\0'

SALIDA: cont\_digito: entero >0

Vble aux: i: entero >=0

A0. Inicializar

A1. LEER(frase)

A2. Mientras (frase<sub>i</sub> <> MF)

    Si (frase<sub>i</sub> es dígito) // en c usamos la función **isdigit()**

        cont\_digito ← cont\_digito +1

    fin\_si

    i ← i+1

fin\_mientras

A3. ESCRIBIR(cont\_digito)

A4. PARAR

# ¿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.

# 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.

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

***int arreglo[];***

# Uso de constantes enteras

```
#include <stdio.h>  
#define TAMA 20
```

```
int main()  
{  
    int arreglo[TAMA];  
    ...  
}
```

# 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

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

Ejemplos:

$a[5] = 0.8;$

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

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

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



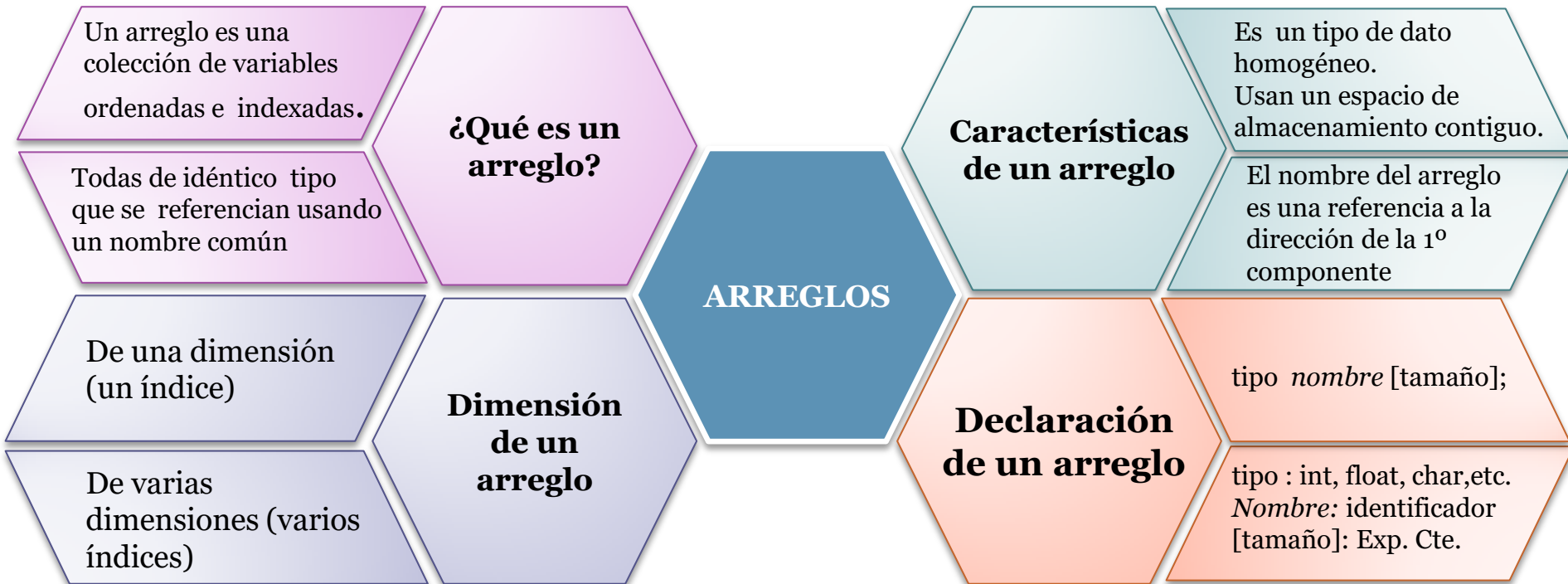
# Del tamaño del arreglo

- 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, 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.

# Temas vistos hasta acá



# Ejemplos de Arreglos

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 3

int main(void) {

    int arre1[5];
    int arre2[TAMA];
    int arre3[] = {0,1,2,3,4};
    ...
    return 0;
}
```

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.

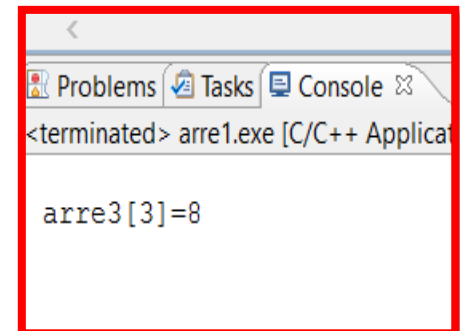
```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 3

int main(void) {

    int arre3[] = {2,4,6,8,10};
    printf( "arre3[3] = %d", arre3[3]);
    return 0;
}
```

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?



# Arreglos

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 20

int main(void) {

    int arre2[TAMA];

    printf("Ingrese un 20 numeros enteros:");

    for(int i=0; i<TAMA; i++)
    {
        printf("\n arre2[%d]:", i);
        scanf("%d", &arre2[i]);
    }

    printf("El arreglo ingresado es: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre2[%d]=%d", j, arre2[j]);
    }
    return 0;
}
```

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

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 20

int main(void) {

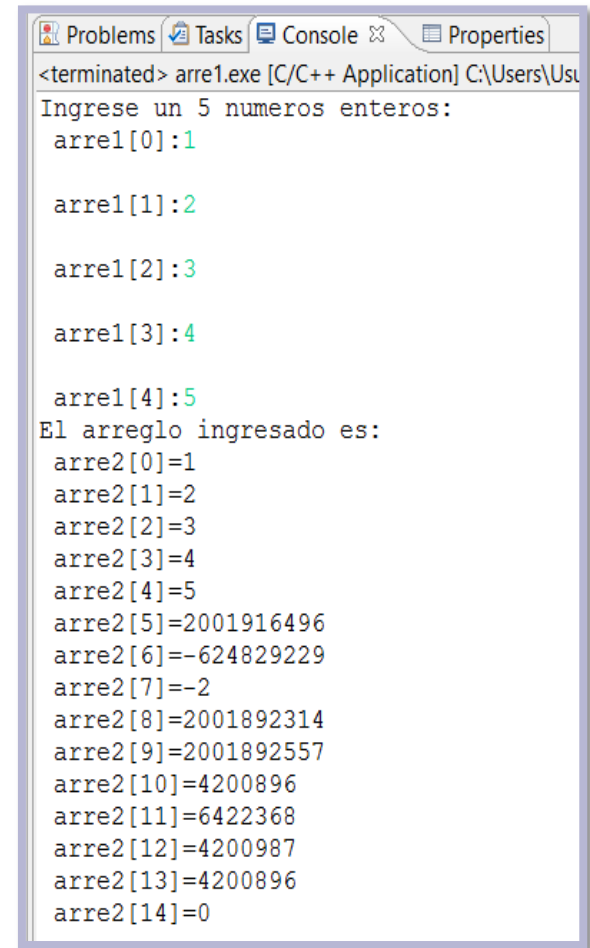
    int arre2[TAMA];

    printf("Ingrese un 5 numeros enteros:");

    for(int i=0; i<5; i++)
    {
        printf("\n arre2[%d]:", i);
        scanf("%d", &arre2[i]);
    }

    printf("El arreglo ingresado es: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre2[%d]=%d", j, arre2[j]);
    }
    return 0;
}
```

- ¿Qué error encuentra en el código?



```
Problems Tasks Console Properties
<terminated> arre1.exe [C/C++ Application] C:\Users\Usu
Ingrese un 5 numeros enteros:
arre1[0]:1

arre1[1]:2


arre1[2]:3

arre1[3]:4

arre1[4]:5
El arreglo ingresado es:
arre2[0]=1
arre2[1]=2
arre2[2]=3
arre2[3]=4
arre2[4]=5
arre2[5]=2001916496
arre2[6]=-624829229
arre2[7]=-2
arre2[8]=2001892314
arre2[9]=2001892557
arre2[10]=4200896
arre2[11]=6422368
arre2[12]=4200987
arre2[13]=4200896
arre2[14]=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.

```
#define LIMITE 10  
#define PROBLEMA 500  
  
int main(void)  
{      int ind, lio[LIMITE];  
  
        for(ind=0; ind < PROBLEMA; ind++)  
  
            lio[ind]=ind;  
        return 0;  
}
```

# Inicialización de Arreglos

Hay 3 formas de inicializar un arreglo:

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

# Por omisión

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 5
int arre2[TAMA];

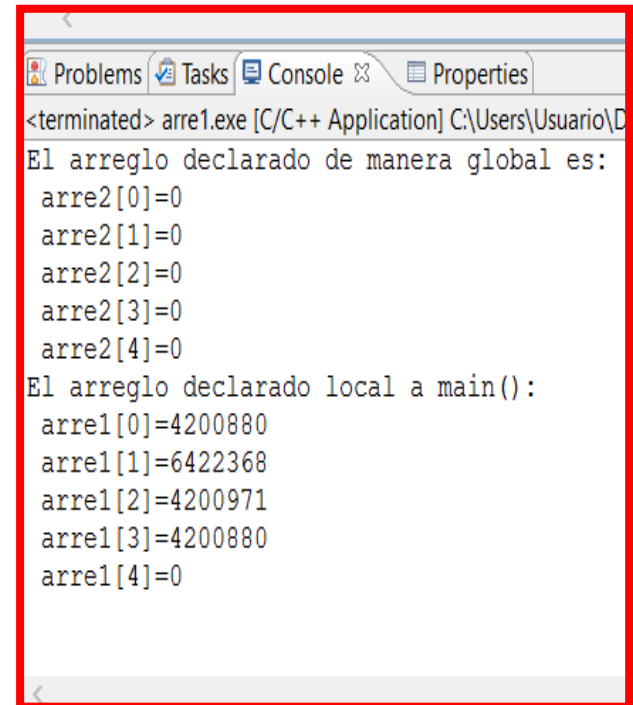
int main(void)
{
    int arre1[TAMA];

    printf("El arreglo declarado de manera global es: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre2[%d]=%d", j, arre2[j]);
    }

    printf("\nEl arreglo declarado local a main(): ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre1[%d]=%d", j, arre1[j]);
    }

    return 0;
}
```

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



```
<terminated> arre1.exe [C/C++ Application] C:\Users\Usuario\D
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]=4200880
arre1[1]=6422368
arre1[2]=4200971
arre1[3]=4200880
arre1[4]=0
```

# Por inicialización explícita

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 3

int main(void) {
    float arre1[] = {78.6, 98.9, 45.7, 34.5, 56.7};
    int arre3[] = {2,4,6};

    setvbuf(stdout, NULL, _IONBF, 0);

    printf("El arreglo 3 es: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre3[%d]=%d", j, arre3[j]);
    }

    printf("\nEl arreglo 1: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre1[%d]=%f", j, arre1[j]);
    }
    return 0;
}
```



**CUIDADO CON ESTA INICIALIZACIÓN!!!. ¿Cuál es el problema?**

```
Problems Tasks Console
<terminated> arre1.exe [C/C++ Applicat
El arreglo 3 es:
arre3[0]=2
arre3[1]=4
arre3[2]=6
El arreglo 1:
arre1[0]=78.599998
arre1[1]=98.900002
arre1[2]=45.700001
```

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.**

# En tiempo de Ejecución

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 5

int main(void) {

    float arre1[TAMA];

    setvbuf(stdout, NULL, _IONBF, 0);

    printf("Ingresar 5 numeros reales: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre1[%d]", j);
        scanf("%f", &arre1[j]);
    }

    printf("\nEl arreglo 1: ");
    for(int j = 0; j<TAMA; j++)
    {
        printf("\n arre1[%d]=%.2f", j, arre1[j]);
    }

    return 0;
}
```

- 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.

```
...
printf("Ingresar 5 numeros reales: ");
for(int j = 0; j<TAMA; j++)
{
    arre1[j] = j * 2;
}

printf("\nEl arreglo 1: ");
for(int j = 0; j<TAMA; j++)
{
    printf("\n arre1[%d]=%d", j,
arre1[j]);
}

return 0;
}
```

# 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).

```
#define MAX 45
```

```
...
```

```
int main(void)
```

```
{
```

```
    char apellido[MAX] = {'P','e','r','e','z'\0'};
```

```
    char nombre[MAX] = "Marcelo"; // agrega al final el  
correspondiente cero de terminación al final de la cadena
```

```
    ...
```

```
}
```

# También vale la inicialización:

```
Char unt[12]= {'U','n','i','v','e','r','s','i','d','a','d'};
```

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



# 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`).

# Funciones y Arreglos

```
#include <stdio.h>
#include <stdlib.h>
#define TAMA 10

void Inicializar(int arre[TAMA] , int n);
void CargarArreglo(int arre[TAMA], int n);

int main(void)
{
    int arre[TAMA];
    int n;

    printf("Ingrese la cantidad de Elementos (<10):");
    scanf("%d", &n);

    Inicializar(arre, n);
    CargarArreglo(arre, n);

    for(int i=0; i< n; i++)
    {
        printf("\n arre[%d] = %d", i, arre[i]);
    }

    return 0;
}
```

```
void CargarArreglo(int arre[TAMA], int n)
{
    int i;
    for(i=0; i<n; i++)
    { arre[i]= i * 3; }
}
```

```
Void Inicializar(int arre[TAMA] , int n)
{
    int i;
    For(i=0; i<n; i++)
    {
        arre[i]=0;
    }
}
```

# 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

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

```
int main()
```

```
{
```

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

```
    return 0;
```

```
}
```