Matrices C (Arrays)

Diego Iván Oliveros Acosta

Unidimencionales:

- Las matrices se utilizan para almacenar múltiples valores en una sola variable, en lugar de declarar variables separadas para cada valor.
- Para crear una matriz, defina el tipo de datos (como int) y especifique el nombre de la matriz seguido de corchetes [] .
- Para insertarle valores, utilice una lista separada por comas, dentro de llaves, de esta forma hemos creado una variable que contiene una matriz de cuatro números enteros:

```
int myNumbers[] = {25, 50, 75, 100};
```

Acceder a los elementos de una matriz

- Para acceder a un elemento de una matriz, consulte su número de índice .
- Los índices de matriz comienzan con 0 : [0] es el primer elemento. [1] es el segundo elemento, etc.
- Esta declaración accede al valor del primer elemento [0] en myNumbers:

```c
#include <stdio.h>

int main() {
  int myNumbers[] = {25, 50, 75, 100};
  printf("%d", myNumbers[0]);

  return 0;
}
```

Cambiar un elemento de matriz

- Para cambiar el valor de un elemento específico, consulte el número de índice:

```c
#include <stdio.h>

int main() {
  int myNumbers[] = {25, 50, 75, 100};
  int i;

  for (i = 0; i < 4; i++) {
    printf("%d\n", myNumbers[i]);
  }

  return 0;
}
```

Establecer el tamaño de la matriz

```c
#include <stdio.h>
/*Otra forma común de crear matrices es especificar el tamaño de la matriz
y agregar elementos más tarde:*/
int main() {
  // Declare an array of four integers:
  int myNumbers[4];

  // Add elements to it
  myNumbers[0] = 25;
  myNumbers[1] = 50;
  myNumbers[2] = 75;
  myNumbers[3] = 100;
  printf("%d\n", myNumbers[0]);
  return 0;
}
```

Ejercicios:

1. Crea una matriz de tipo `int` llamada `myNumbers` con los valores `25, 50, 75, 100`.
2. Imprime el valor del segundo elemento de la myNumbersmatriz.
3. Cambie el valor del primer elemento a 90.
4. Recorre los elementos de la matriz utilizando un bucle for .

Obtener el tamaño o la longitud de la matriz

- Para obtener el tamaño de una matriz, puede utilizar el operador sizeof, para el siguiente ejemplo, analice, ¿Por qué el resultado se muestra 20 en lugar de 5, cuando la matriz contiene 5 elementos?

```c
#include <stdio.h>

int main() {
  int myNumbers[] = {10, 25, 50, 75, 100};
  printf("%lu", sizeof(myNumbers));

  return 0;
}
```

Obtener el tamaño o la longitud de la matriz

- ¿Qué hacer cuando solo desea saber cuántos elementos tiene una matriz?

```c
#include <stdio.h>

int main() {
  int myNumbers[] = {10, 25, 50, 75, 100};
  int length = sizeof(myNumbers) / sizeof(myNumbers[0]);

  printf("%d", length);
  return 0;
}
```

Puede utilizar la fórmula (que divide el tamaño de la matriz por el tamaño de un elemento de la matriz):

Creando mejores bucles

- Escribimos el tamaño de la matriz en la condición de bucle (i < 4).
- Esto no es ideal, ya que solo funcionará para matrices de un tamaño específico.

- Sin embargo, al utilizar la fórmula sizeof del ejemplo anterior, ahora podemos crear bucles que funcionen para matrices de cualquier tamaño, lo que es más sostenible.

```c
#include <stdio.h>

int main() {
  int myNumbers[] = {25, 50, 75, 100};
  int i;

  for (i = 0; i < 4; i++) {
    printf("%d\n", myNumbers[i]);
  }

  return 0;
}
```

Todo junto:

```c
#include <stdio.h>

int main() {
  int myNumbers[] = {25, 50, 75, 100};
  int length = sizeof(myNumbers) / sizeof(myNumbers[0]);
  int i;

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

  return 0;
}
```

Ejemplo 1:

```c
#include <stdio.h>
// Calcule el promedio de diferentes edades
int main() {
  int ages[] = {20, 22, 18, 35, 48, 26, 87, 70};  // An array storing different ages
  float avg, sum = 0;
  int i;
  int length = sizeof(ages) / sizeof(ages[0]);   // Get the length of the array

  for (i = 0; i < length; i++) {     // Loop through the elements of
    sum += ages[i];                  //the array and accumulate the sum
  }

  avg = sum / length;   // Calculate the average by dividing the sum by the length
  printf("The average age is: %.2f", avg);  // Print the average
  return 0;
}
```

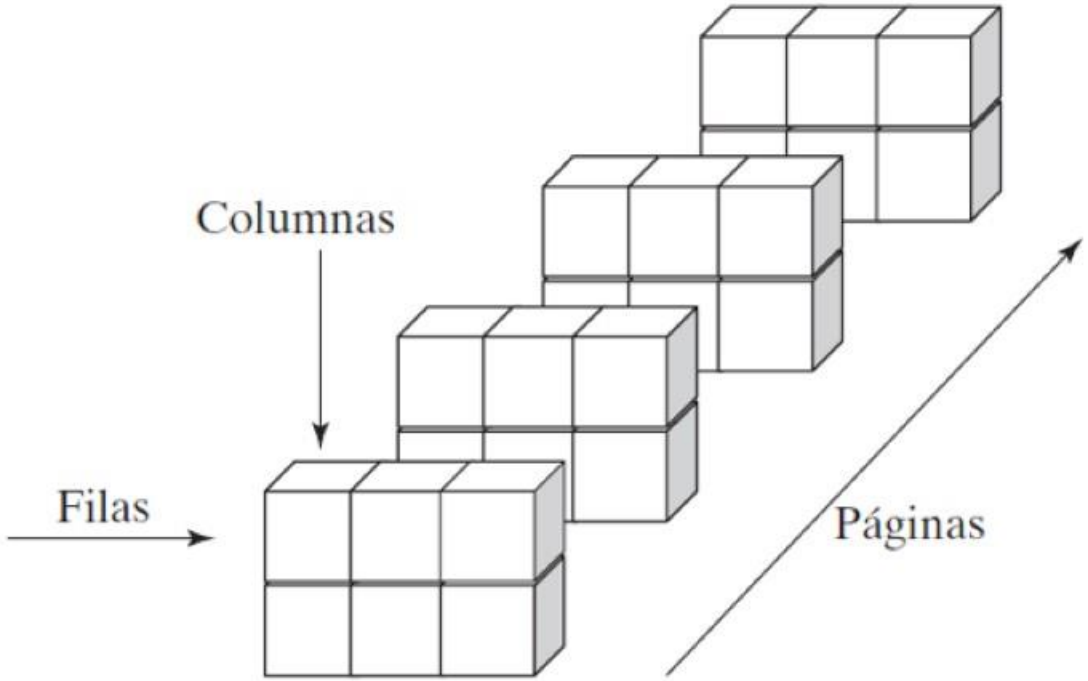
Ejemplo 2:

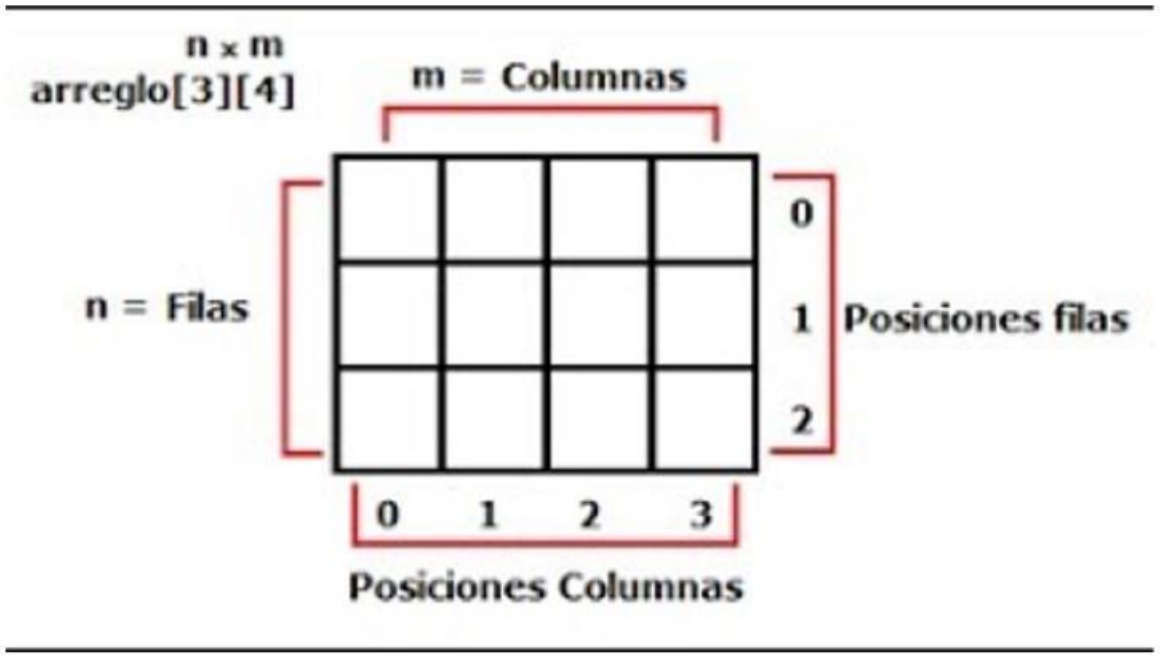
```c
#include <stdio.h>
//Encuentra la edad más baja entre diferentes edades:

int main() {
  int ages[] = {20, 22, 18, 35, 48, 26, 87, 70};   // An array storing different ages
  int i;
  int length = sizeof(ages) / sizeof(ages[0]);   // Get the length of the array
  int lowestAge = ages[0];   // Create a 'lowest age' variable and assign the first array element of ages to it
  for (i = 0; i < length; i++) {   // Loop through the elements of the ages array to find the lowest age
    if (lowestAge > ages[i]) {   // Check if the current age is smaller than current the 'lowest age'
      lowestAge = ages[i];   // If the smaller age is found, update 'lowest age' with that element
    }
  }
  printf("The lowest age in the array is: %d", lowestAge);   // Output the value of the lowest age
  return 0;
}
```

Matrices multidimensionales en C

Diego Iván Oliveros Acosta





Arreglos multidimensionales

Matrices bidimensionales (2D).

Diego Iván Oliveros Acosta @scalapp.co

Matrices multidimensionales

- En la primera parte, aprendiste sobre las matrices , también conocidas como **matrices unidimensionales**, son geniales y las usarás mucho mientras programabas en C. Sin embargo, si quieres almacenar datos en formato tabular, como una tabla con filas y columnas, necesitas familiarizarte con **las matrices multidimensionales** .

Una matriz multidimensional es básicamente una matriz de matrices.

- Las matrices pueden tener cualquier número de dimensiones. Presentaremos las más comunes: **matrices bidimensionales (2D).**

Matrices bidimensionales

- Una matriz 2D también se conoce como matriz (una tabla de filas y columnas).
- Para crear una matriz 2D de números enteros, observe el siguiente ejemplo:
- `int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };`

- La primera dimensión representa el número de filas **[2]** , mientras que la segunda dimensión representa el número de columnas **[3]** . Los valores se colocan en orden de fila y se pueden visualizar como muestra la figura.

	Columna 0	**Columna 1**	**Columna 2**
Fila 0	**1**	**4**	**2**
Fila 1	**3**	**6**	**8**

Acceda a los elementos de una matriz 2D

- Para acceder a un elemento de una matriz bidimensional, debe especificar el número de índice tanto de la fila como de la columna.
- Esta declaración accede al valor del elemento en la **primera fila (0)** y **la tercera columna (2)** de la **matriz** .
- **Recuerde que:** los índices de matriz comienzan con 0: [0] es el primer elemento. [1] es el segundo elemento, etc.

Acceda a los elementos de una matriz 2D

```c
#include <stdio.h>

int main() {
   int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
   printf("%d", matrix[0][2]);

   return 0;
}
```

Cambiar elementos en una matriz 2D

```c
#include <stdio.h>
/*Para cambiar el valor de un elemento, consulte el número de índice
del elemento en cada una de las dimensiones: El siguiente ejemplo
cambiará
el valor del elemento en la primera fila (0) y la primera columna (0) :*/

int main() {
  int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
  matrix[0][0] = 9;
  printf("%d", matrix[0][0]);  // Now outputs 9 instead of 1

  return 0;
}
```

Recorrer una matriz 2D con un bucle.

```c
#include <stdio.h>
/*Para recorrer una matriz multidimensional, necesita un bucle  para cada una de las
dimensiones de la matriz. El siguiente ejemplo genera todos los elementos de la matriz :*/


int main() {
  int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
  int i, j;
  for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) { printf("%d\n", matrix[i][j]);}
  }
  return 0;
}
```