FUNDAMENTOS DE PROGRAMACIÓN

Inicio Clase 09

Profesor: Carlos Díaz

Clase 09: Arreglos bidimensionales

- Array bidimensional
- Inicializar array bidimensional
- Arrays como argumentos de funciones

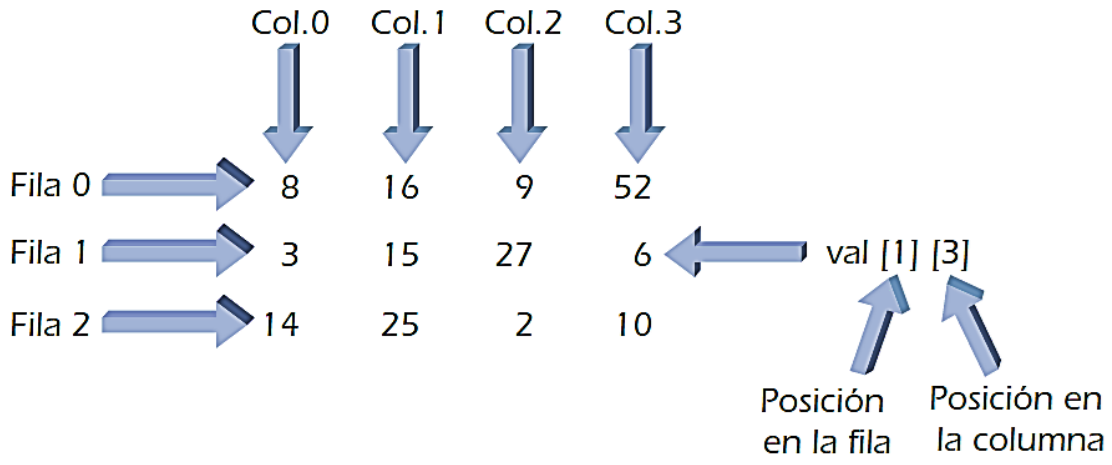
Array bidimensional

Un **arreglo bidimensional**, el cual a veces se denomina tabla, consiste de filas y columnas de elementos. Por ejemplo, el arreglo de números

8	16	9	52
3	15	27	6
14	25	2	10

se llama arreglo bidimensional de números enteros. Este arreglo consiste de tres filas y cuatro columnas. Para reservar almacenamiento para este arreglo, deben incluirse el número de filas y el número de columnas en la declaración del arreglo. Llamando al arreglo `val`, la especificación correcta para este arreglo bidimensional es

```
int val[3][4];
```



Inicializar un array bidimensional

```
int val[3][4] = { {8,16,9,52},
                  {3,15,27,6},
                  {14,25,2,10} };
```

```
int val[3][4] = {8,16,9,52,
                 3,15,27,6,
                 14,25,2,10};
```

```
int val[3][4] = {8,16,9,52,3,15,27,6,14,25,2,10};
```

Ejercicio 1

Determine la salida producida por el siguiente programa:

```
#include <iostream>
using namespace std;

int main()
{
  int i, j, val[3][4] = {8,16,9,52,3,15,27,6,14,25,2,10};

  for (i = 0; i < 3; ++i)
    for (j = 0; j < 4; ++j)
      cout << "   " << val[i][j];

  return 0;
}
```

Ejercicio 2

a. Escriba un programa en C++ que sume los valores de todos los elementos en el arreglo val usados en el ejercicio 1 y despliegue el total.

b. Modifique el programa escrito para el ejercicio 2a para desplegar el total de cada fila por separado.

Ejercicio 3

Escriba un programa en C++ que sume elementos equivalentes de los arreglos bidimensionales llamados `primero` y `segundo`. Ambos arreglos deberán tener dos filas y tres columnas. Por ejemplo, el elemento `[1][2]` del arreglo resultante deberá ser la suma de `primero[1][2]` y `segundo[1][2]`. Los arreglos `primero` y `segundo` deberán inicializarse como sigue:

PRIMERO			SEGUNDO		
16	18	23	24	52	77
54	91	11	16	19	59

Ejercicio 4

a. Escriba un programa en C++ que encuentre y despliegue el valor máximo en un arreglo bidimensional de números enteros. El arreglo deberá ser declarado como un arreglo de números enteros de 4 por 5 e inicializarse con los datos 16, 22, 99, 4, 18, −258, 4, 101, 5, 98, 105, 6, 15, 2, 45, 33, 88, 72, 16, 3

b. Modifique el programa escrito en el ejercicio 4a de modo que también despliegue los números de los subíndices de fila y columna del valor máximo.

Ejercicio 5

Escriba un programa en C++ para seleccionar los valores en un arreglo de cuatro por cinco de números enteros positivos en orden creciente y almacenar los valores seleccionados en el arreglo unidimensional llamado **ordenar**. Use la instrucción de datos dada en el ejercicio 4a para inicializar el arreglo bidimensional.

Ejercicio 6

a. Un ingeniero ha construido un arreglo bidimensional de números reales que tiene 3 filas y 5 columnas. Este arreglo contiene en la actualidad los voltajes de prueba de un amplificador. Escriba un programa en C++ que introduzca de manera interactiva 15 valores del arreglo y luego determine el número total de voltajes en los rangos menor que 60, mayor que o igual a 60 y menor que 70, mayor que o igual a 70 y menor que 80, mayor que o igual a 80 y menor que 90, y mayor que o igual a 90.

b. Introducir 15 voltajes cada vez que se ejecuta el programa escrito para el ejercicio 6a es engorroso. Por consiguiente, ¿qué método es apropiado para inicializar el arreglo durante la fase de prueba?

c. ¿Cómo podría modificarse el programa que escribió para el ejercicio 6a para incluir el caso de que no haya ningún voltaje presente? Es decir, ¿qué voltaje podría usarse para indicar un voltaje inválido y cómo tendría que modificarse su programa para excluir el conteo de dicho voltaje?

Arrays como argumentos de funciones

Los elementos del arreglo individuales son transmitidos a una función llamada de la misma manera que las variables escalares individuales; tan sólo se incluyen como variables subindexadas cuando se hace la llamada a la función. Por ejemplo, la llamada a la función

```
hallarMin(voltios[2], voltios[6]);
```

transmite los valores de los elementos `voltios[2]` y `voltios[6]` a la función `hallarMin()`.

Transmitir un arreglo completo de valores a una función es en muchos aspectos una operación más fácil que transmitir elementos individuales. La función llamada recibe acceso al arreglo real, en lugar de a una copia de los valores en el arreglo. Por ejemplo, si `voltios` es un arreglo, la llamada a la función `hallarMax(voltios);` hace que el arreglo `voltios` completo esté disponible para la función `hallarMax()`.

Ejercicio 7

Calcula el máximo valor en un vector.

```
#include <iostream>
using namespace std;

const int MAXELS = 5;
int hallarMax(int [MAXELS]); // prototipo de la función

int main()
{
  int nums[MAXELS] = {2, 18, 1, 27, 16};

  cout<<"El valor máximo es "<<hallarMax(nums)<<endl;

  return 0;
}

// halla el valor máximo
int hallarMax(int vals[MAXELS])
{
  int i, max = vals[0];
  for (i = 1; i < MAXELS; i++)
    if (max < vals[i])
        max = vals[i];

  return max;
}
```

Ejercicio 8

Despliega los elementos de un array.

```
#include <iostream>
#include <iomanip>
using namespace std;

const int FILAS = 3;
const int COLS = 4;
void desplegar(int [FILAS][COLS]); // prototipo de la función

int main()
{
  int val[FILAS][COLS] = {8,16,9,52,
                          3,15,27,6,
                          14,25,2,10};

  desplegar(val);

  return 0;
}

void desplegar(int nums[FILAS][COLS])
{
  int num_fila, num_col;
  for (num_fila = 0; num_fila < FILAS; num_fila++)
  {
    for (num_col = 0; num_col < COLS; num_col++)
      cout << setw(4) << nums[num_fila][num_col];
    cout << endl;
  }

  return;
}
```

Ejercicio 9

Escriba un programa que declare tres arreglos unidimensionales llamados `voltios`, `corriente` y `resistencia`. Cada arreglo deberá declararse en `main()` y deberá ser capaz de contener diez números de precisión doble. Los números que deberán almacenarse en `corriente` son 10.62, 14.89, 13.21, 16.55, 18.62, 9.47, 6.58, 18.32, 12.15, 3.98. Los números que deberán almacenarse en `resistencia` son 4, 8.5, 6, 7.35, 9, 15.3, 3, 5.4, 2.9, 4.8. Su programa deberá transmitir estos tres arreglos a una función llamada `calc_voltios()`, la cual deberá calcular los elementos en el arreglo `voltios` como el producto de los elementos correspondientes en los arreglos `corriente` y `resistencia` (por ejemplo, `voltios[1] = corriente[1] * resistencia[1]`). Después que `calc_voltios()` ha puesto valores en el arreglo `voltios`, los valores en el arreglo deberán desplegarse desde adentro de `main()`.

FUNDAMENTOS DE PROGRAMACIÓN

Fin Clase 09

Profesor: Carlos Díaz