Arrays multidimensionales

Arrays bidimensionales

Las matrices, también conocidas como arrays bidimensionales o tablas bidimensionales, no son otra cosa sino un array con dos dimensiones. Por lo que los conceptos de acceso, inicialización, etc. son similares a las de un array ordinario.

La declaración de una matriz tiene la siguiente forma:

```
tipo_de_dato nombre_variable [dimension1][dimension2];
```

Donde *dimension1* y *dimension2* indican respectivamente, el número de filas y de columnas de la matriz.
Otro hecho importante es que las matrices en C almacenan "por filas". Es decir, que los elementos de cada fila se sitúan en memoria en forma contigua. Entonces en la matriz que figura en la Tabla1, el primer elemento almacenado en la memoria es el (0, 0), el segundo el (0, 1), el tercero (0, 2)... (0, M-1), después (1, 0) y así sucesivamente hasta el último elemento, osea (N-1, M-1).

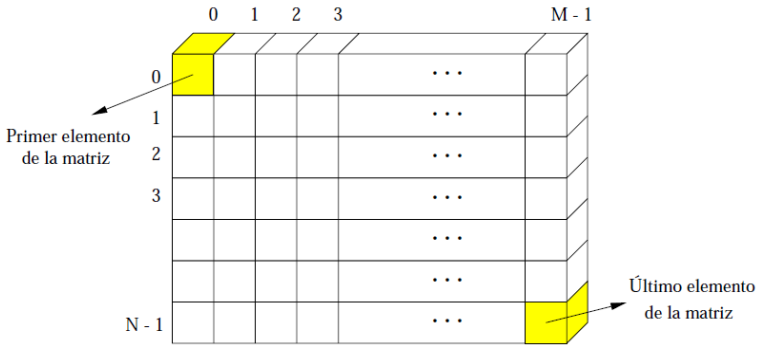

Figura 1 – Representación gráfica de una matriz NxM

Consulta

El acceso al contenido de una "celda" de la matriz se realiza mediante el nombre de ésta y sus índices (uno para cada dimensión) entre corchetes. El primer índice representa la fila y el segundo la columna en que se encuentra dicho elemento. El índice de las filas toma un valor entre 0 y el número de filas menos uno y en el caso de las columnas el valor es entre 0 y el número de las columnas menos uno. Es responsabilidad del programador garantizar este hecho.

```
nombre_matriz[índice_filas][índice_columnas];
```

Asignación

Para asignarle valores a una matriz se puede realizar de las siguientes formas:

El próximo ejemplo declara una matriz de 3x4 y le asigna los valores en la misma declaración. Ésta práctica sólo puede hacerse en el momento de la declaración.

```
int matriz[3][4] = {1, 2, 3, 4, 10, 20, 30, 40, 100, 200, 300, 400};
```

El siguiente ejemplo declara una matriz de 3x4 y le asigna los valores en la misma declaración, para mejor entendimiento del código se separan las filas entre llaves {}. Ésta práctica sólo puede hacerse en el momento de la declaración.

```
int matriz[3][4] = { { 1, 2, 3, 4 } , { 10, 20, 30, 40} , { 100, 200, 300, 400 } };
```

Otra forma de para asignar valores es a través del nombre de la matriz y sus dimensiones entre corchetes.

```
int matriz[3][4];
matriz[0][0] = 0;
matriz[2][3] = 100;
```

Ejercicio de ejemplo:

Se dispone de una matriz de 5x12 para almacenar los sueldos de las faltas de los cinco empleados de una empresa
durante el año 2009.
Se propone generar un array con la cantidad de faltas en total por mes de toda la empresa.

```
#include<iostream>
using namespace std;
const int _EMPLEADOS = 5;

void cargarFaltasRandom(int (*)[12]);
void listarFaltas(int [_EMPLEADOS][12], char [12][15]);
void buscarMaximo(int (*)[12], int *, int*);

int main(void){
   int mfaltasxMes[_EMPLEADOS][12], empleadoMax, mesMax;
   char mMeses[12][15] =
{"Enero","Febrero","Marzo","Abril","Mayo","Junio","Julio","Agosto","Septiembre","Octubre","No
viembre","Diciembre"};
   cargarFaltasRandom(mfaltasxMes);
   listarFaltas(mfaltasxMes, mMeses);
   system("pause");
   buscarMaximo(mfaltasxMes, &empleadoMax, &mesMax);
   cout << "El empleado " << (empleadoMax + 1) << " faltó " << mfaltasxMes[empleadoMax]
[mesMax] << " veces en el mes de " << mMeses[mesMax] << endl;
   system("pause");
}

void cargarFaltasRandom(int (*M)[12]){
    int i,j;
    for(i=0; i<_EMPLEADOS; i++){
      for(j=0; j<12; j++){
        M[i][j] = rand() % 30;
      }
    }
}
void listarFaltas(int M[_EMPLEADOS][12], char meses[12][15]){
   int i,j;
   for(i=0; i<12; i++){
```

```
    cout << meses[i] << endl << endl;
    for(j=0; j<_EMPLEADOS; j++){
     cout << "Empleado " << (j+1) << ": " << M[j][i] << endl;
    }
    system("pause");
    system("cls");
   }
}
void buscarMaximo(int (*M)[12], int *F, int *C){
 int i,j,max=0;
 for(i=0; i<_EMPLEADOS; i++){
  for(j=0; j<12; j++){
   if(M[i][j] > max)
   {
    max = M[i][j];
    *F = i;
    *C = j;
   }
  }
 }
}
```

T02CF01.cpp

Arrays multidimensionales

Este tipo de "tablas" se caracterizan por tener tres o más dimensiones. Al igual que los vectores y matrices, todos los elementos que allí se almacenan comparten el mismo tipo de datos.

La declaración de un array multidimensional es la siguiente:

```
tipos_de_dato nombre_variable[dimension1]...[dimensionN];
```

Ejemplos:

```
int m3[12][31][5];
float m4[2][5][10][5];
```

Para acceder a un elemento en particular será necesario tantos índices como dimensiones tenga la "tabla".

Ejemplo:

```
cout << m3[0][10][3];
float num = m4[0][2][8][2];
```

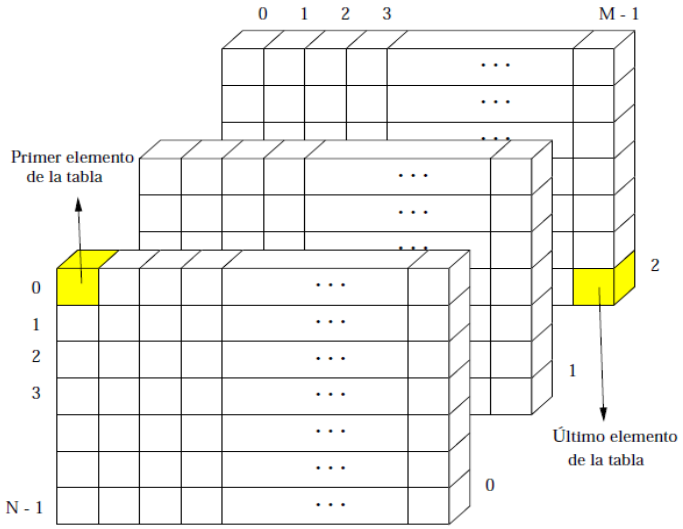

Figura 2 – Representación de una matriz de NxMxZ

Ejemplo:

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

const int X = 3, Y = 2, Z = 4;
const int MAX_NUM = 10;

void mostrarMatriz3d(int (*)[Y][Z]);
void cargarMatriz3d(int [X][Y][Z]);

int main(void){
   int Mat[X][Y][Z] = {0};
   cargarMatriz3d(Mat);
   mostrarMatriz3d(Mat);
}

void mostrarMatriz3d(int (*m)[Y][Z]){
   for(int i=0; i<X; i++){
      cout << "Matriz " << (i+1) << ": ";
      for(int j=0; j<Y; j++){
         cout << endl;
         for(int k=0; k<Z; k++){
            cout << m[i][j][k] << " ";
         }
      }
      cout << endl << endl ;
   }
}
void cargarMatriz3d(int m[X][Y][Z]){
   int i,j,k;
   for(i=0; i<X; i++){
      for(j=0; j<Y; j++){
```

```
        for(k=0; k<Z; k++){
            m[i][j][k] = rand() % MAX_NUM;
        }
      }
   }
}
```

T02CF03.cpp

Salida:

```
Matriz 1:
1 7 4
0 9 4
8 8 2

Matriz 2:
4 5 5
1 7 1
1 5 2

Matriz 3:
7 6 1
4 2 3
2 2 1

Matriz 4:
6 8 5
7 6 1
8 9 2
```

En el ejemplo anterior, se utilizan dos funciones *cargarMatriz3d* y *mostrarMatriz3d* para cargar una matriz de tres dimensiones con números aleatorios y mostrarla por pantalla respectivamente. Se declaran tres constantes de tipo de dato int para poder definir los subíndices de las dimensiones y para definir el tope en el rango de valores que puede variar el número al azar en *cargarMatriz3d*.
El ejercicio no presenta ninguna solución a ningún problema, simplemente tiene como objetivo mostrar el uso de una matriz de tres dimensiones.

Bibliografía

- Peña Basurto Marco, Cela Espín José, Introducción a la programación en C, UPC, Septiembre de 2000, ISBN: 84-8301-429-7.