CS1102 – PROGRAMACIÓN ORIENTADA A OBJETOS 1
CICLO 2018-2

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Unidad 5: Arrays de arrays

http://bit.ly/2p3fgiD

Profesores:

Ernesto Cuadros-Vargas, PhD. *ecuadros@utec.edu.pe*

María Hilda Bermejo, M. Sc. *mbermejo@utec.edu.pe*

Telegram:

1. Configurar tu cuenta

2. Link: http://bit.ly/2OW5Ss9

SISAP:

Evento: SISAP 2018 – 11th International Conference on Similarity Search and Applications
Fechas: October 7-9 Lima, Perú
Resumen: http://www.sisap.org/2018/

The 11th International Conference on Similarity Search and Applications (SISAP) is an annual forum for researchers and application developers in the area of similarity data management. It aims at the technological problems shared by numerous application domains, such as data mining, information retrieval, multimedia, computer vision, pattern recognition, computational biology, geography, biometrics, machine learning, and many others that make use of similarity search as a necessary supporting service.

Inscripciones: https://eventos.spc.org.pe/spire2018/registration_sisap.html

SPIRE:

SPIRE 2018: 25th International Symposium on String Processing and Information Retrieval
Fechas: October 9-11 Lima, Perú
Resumen: https://eventos.spc.org.pe/spire2018/venue.html

SPIRE 2018 is the 25th edition of the annual Symposium on String Processing and Information Retrieval. SPIRE has its origins in the South American Workshop on String Processing, which was first held in Belo Horizonte, Brazil, in 1993. Since 1998 the focus of the workshop has also included information retrieval, due to its increasing relevance to and inter-relationship with string processing.
SPIRE 2018 will be held in UTEC Lima, Peru.
Inscripciones: https://eventos.spc.org.pe/spire2018/registration.html

Logro de la sesión:

Al finalizar la sesión, los alumnos desarrollan sus programas utilizando arrays de arrays.

Arrays de arrays (Matrices)

Lo que solía denominarse arrays multidimensionales son actualmente arrays de arrays.

Definiciones:

```
int ia[3][4];   // array of size 3, each element is an arrays of ints of size 4

int arr[10][20][30] ={0};
//  array of size 10; each element is a 20-element arrays whose element are arrays of 30 ints.
// initialize all elements to 0
```

Inicialización de los elementos de un array:

```
int ia[3][4]={          //three elements, each element is an array of size 4
    {0, 1, 2, 3},       //  initializers for the row indexex by 0
    {4, 5, 6, 7},       //  initializers for the row indexed by 1
    {8,9,10,11}     //  initializers for the row indexed by 2
  };
```

		0	1	2	3
	0	0	1	2	3
ia	1	4	5	6	7
	2	8	9	10	11

ia	0	1	2	3	4	5	6	7	8	9	10	11

```
int  ia[3][4]  = {  {  0  },  {  4  } , {  8 } };
// explicity initialize only element 0 in each row
```

```
int  ix[3][4] = { 0, 3, 6, 9 };
// explicity  initialize row 0; the remaining elements are value initialized.
```

The remaining elements are initialized to 0.

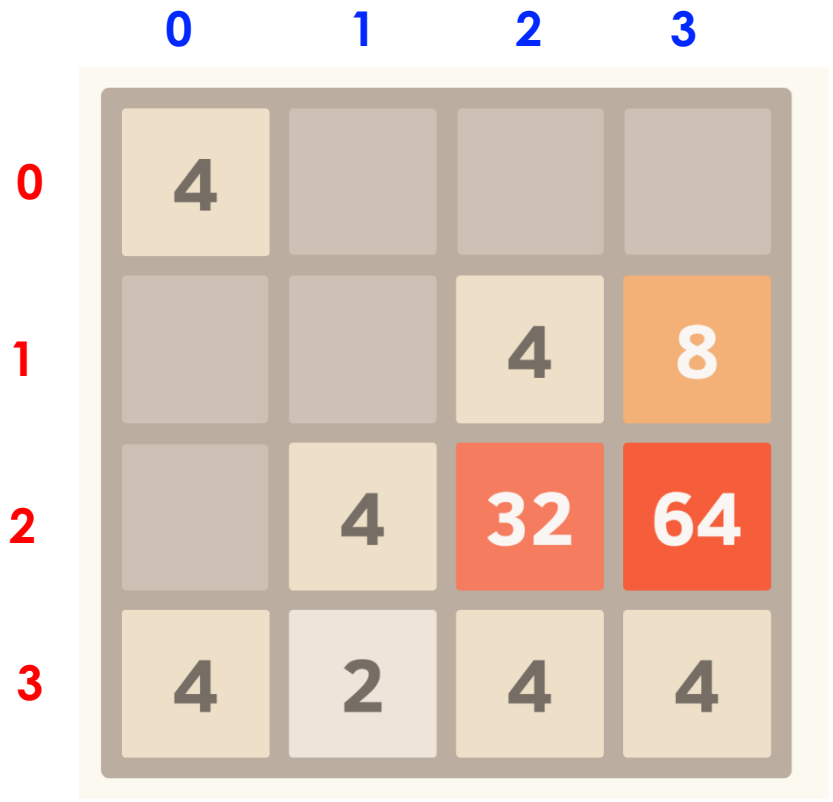
```
constexpr  size_t  rowCnt = 3,  colCnt = 4;

int ia[rowCnt][colCnt];    //  12 uninitialized elements


// for each row
for( sizet_t  i=0; i<rowCnt; i++)
    //--- for each column within row
   for( size_t j=0; j<colCnt; j++)
      ia[i][j] =  i*colCnt + j;
      // Assing the element's positional index as tis value
```

Ejemplo:

```
int  Dosmil48[ 4 ] [ 4 ];
```

filas

columnas



Orden de la matriz 4x4

Pasando un array multimensional a funciones:

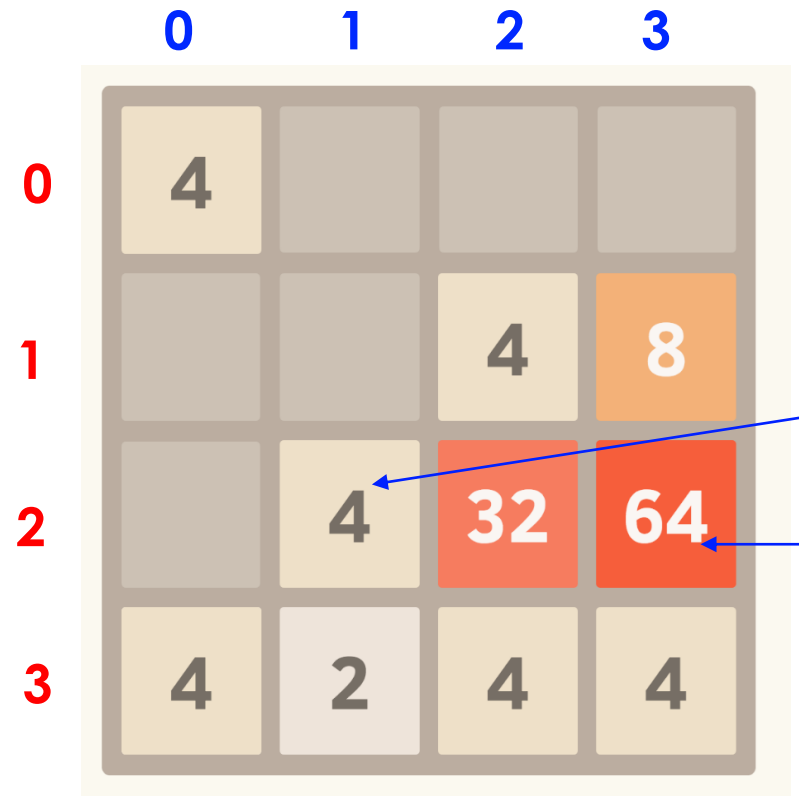
```
void  funcion1( int matrix[][10], int rowSize, int colSize);
```

El parámetro es un puntero a un array de 10 enteros.

El array nunca se pasa por valor, lo que envía es el puntero al primer elemento

Cómo se asigna un dato a un casillero de la matriz?

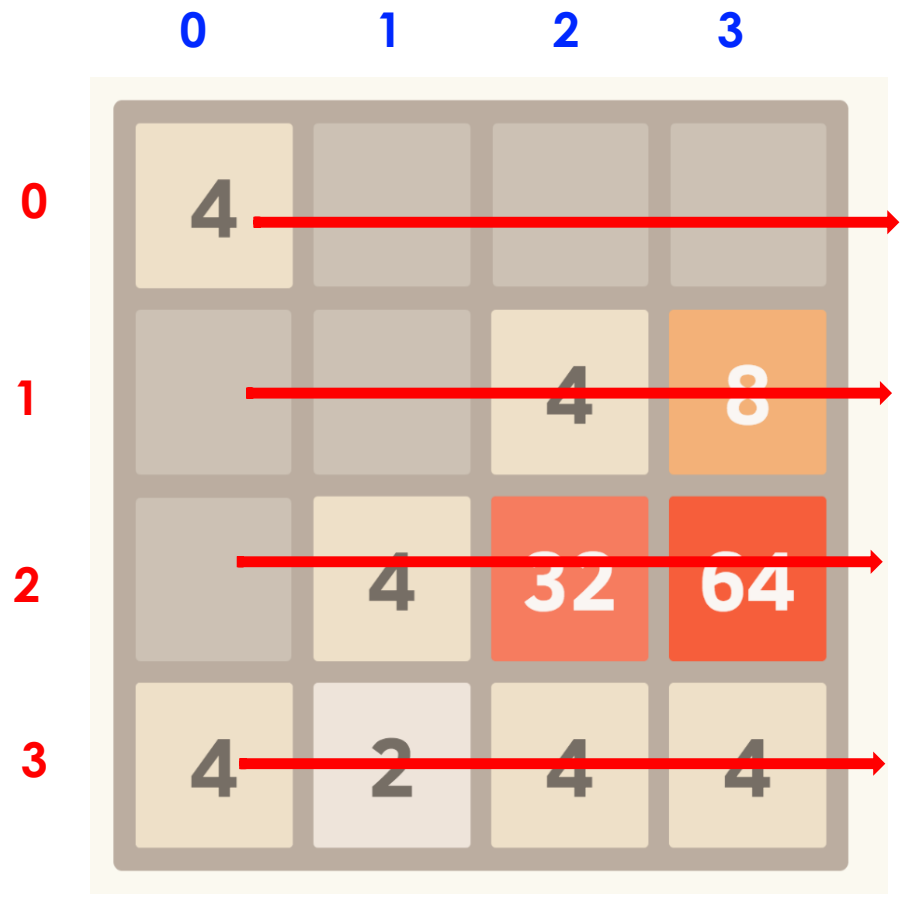
```
int  Dosmil48[ 4 ] [ 4 ];
```



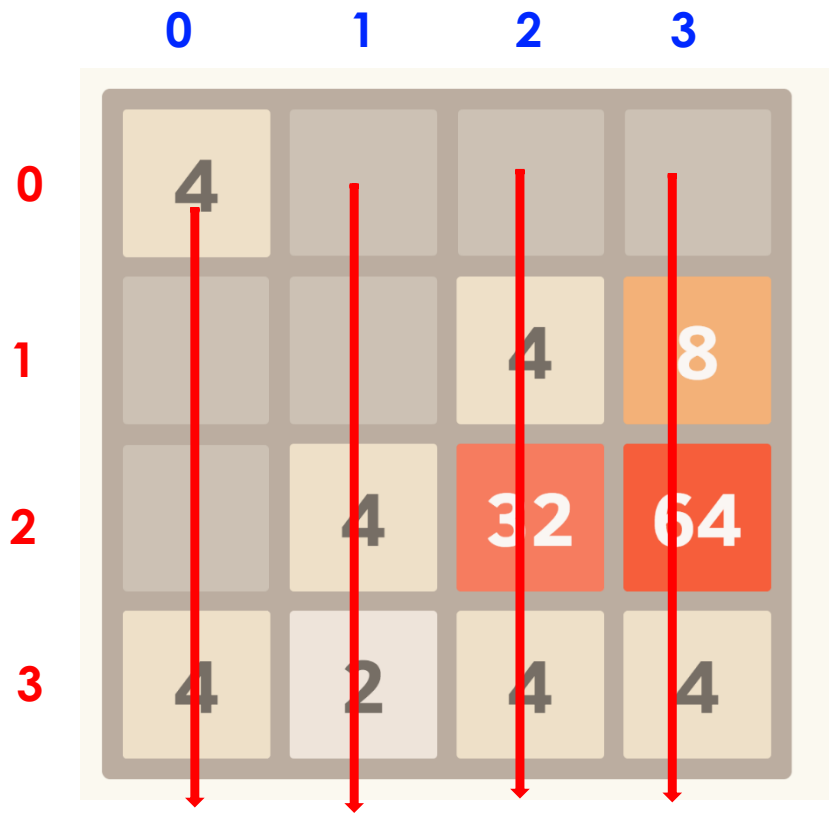

```
Dosmil48[2][1] = 4;
```

```
cout << Dosmil48[2][3];
```

Recorrido fila por fila:

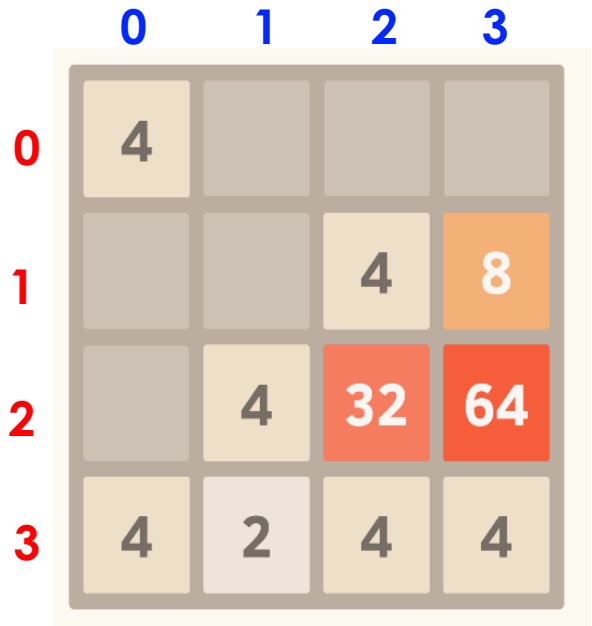


Recorrido columna por columna:



Por ejemplo cuando se quiere leer datos desde el teclado y almacenarlos en la matriz.

```
int  Dosmil48[ 4 ] [ 4 ];
```



Se recorre la matriz fila por fila

```
for(int f=0; f<4; f++)
{
  for(int c=0; c<4; c++)
  { cout << "Dosmil48[" << f << "][" << c << "]=";
    cin >> Dosmil48[f][c];
  }
}
```

Ejemplo: 1

Desarrolle un programa que permita generar aleatoriamente números enteros (entre 0 y 99), los almacene en un array de arrays, cuyas características se muestra en la figura:

M

	0	1	2	3	4	5	6	7
0	22	45	67	12	34	21	3	56
1	21	45	65	76	77	89	99	88
2	5	65	34	52	21	33	45	23
3	21	56	78	80	32	45	22	27
4	45	67	56	23	45	4	43	87
5	55	23	23	4	12	3	23	65
6	78	45	45	12	34	23	22	43
7	12	23	28	33	65	12	12	44

continúa

Luego realice lo siguiente:

1. **Imprima el array de arrays.**
2. **Hallar el dato mas pequeño del array de arrays.**
3. **Imprimir la suma de los elementos de la diagonal**
4. **Hallar la suma de los elementos que estan por encima de la diagonal.**

```
 65   29   75   29   68   98   70
  8   38   63   13   18   71   42
 94   69   46   38    4   27   57
 97   93   59   45   97    9   79
 23   72   79   42   36   72   53
 14   65   27   60   92   93    7
 94   28   99   61   37   79   24

El menor elemento almacenado en el array es 4
La suma de la diagonal es igual a 347
Suma de los elementos por encima de la diagonal : 1019
```

Main.cpp

```
1  #include <iostream>
2  #include <ctime>      //Se incluye <ctime> para poder usar la función srand()
3  #include "Arrays.h"
4
5  using namespace std;
6
7
8  int main()
9  {
10   int M[nFILAS][nCOLUMNAS];
11
12   srand(time(nullptr));
13   cout<<"\n";
14   LlenarArray(M,nFILAS, nCOLUMNAS);
15   ImprimirArray(M,nFILAS, nCOLUMNAS);
16   cout<<"\n";
17   cout<<"El menor elemento almacenado en el array es " << ElMenor(M,nFILAS,nCOLUMNAS);
18   cout<<"\n";
19   cout<<"La suma de la diagonal es igual a "<<SumadeDiagonal(M,nFILAS,nCOLUMNAS);
20   cout<<"\n";
21   cout<<"Suma de los elementos por encima de la diagonal : " <<SumaPorEncimadelaDiagonal(M,nFILAS,nCOLUMNAS);
22   cout<<"\n";
23   return 0;
24 }
```

Array.h

```
1   //
2   // Created by Maria Hilda Bermejo on 9/17/18.
3   //
4
5   #ifndef EJEMPLO1_RECORRIDOS_ARRAYS_H
6   #define EJEMPLO1_RECORRIDOS_ARRAYS_H
7
8   #include <iostream>
9   #include <cstddef>  //-- para usar size_t
10
11  using namespace std;
12
13  constexpr  size_t  nFILAS = 7,  nCOLUMNAS = 7;
14
15  void LlenarArray(int M[][nCOLUMNAS],size_t filas, size_t columnas);
16  void ImprimirArray(int M[][nCOLUMNAS],size_t filas, size_t columnas);
17  int ElMenor(int M[][nCOLUMNAS], size_t filas, size_t columnas);
18  int SumadeDiagonal(int M[][nCOLUMNAS], size_t filas, size_t columnas);
19  int SumaPorEncimadelaDiagonal(int M[][nCOLUMNAS], size_t filas, size_t columnas);
20
21  #endif //EJEMPLO1_RECORRIDOS_ARRAYS_H
22
```

Array.cpp

```cpp
//
// Created by Maria Hilda Bermejo on 9/17/18.
//

#include <cstdlib>
#include <iomanip>
#include "Arrays.h"


void LlenarArray(int M[][nCOLUMNAS],size_t filas, size_t columnas)
{//-------------------------------------------------------
  for(size_t f=0; f<filas; f++)
    for(size_t c=0; c<columnas; c++)
      M[f][c] = rand()%100;
}

void ImprimirArray(int M[][nCOLUMNAS],size_t filas, size_t columnas)
{//-------------------------------------------------------------
  for(size_t f=0; f<filas; f++)
  {
    for (size_t c = 0; c < columnas; c++)
      cout << setw(5) << M[f][c];
    cout<<"\n";
  }
}

```

Continúa …. Main.cpp

```
27  int ElMenor(int M[][nCOLUMNAS], size_t filas, size_t columnas)
28  {//-------------------------------------------------------
29    int Menor;
30
31    Menor=M[0][0];
32    for(size_t f=0; f<filas; f++)
33      for (size_t c = 0; c < columnas; c++)
34        if(M[f][c]<Menor)
35          Menor =M[f][c] ;
36    return Menor;
37  }
38
39
40  int SumadeDiagonal(int M[][nCOLUMNAS], size_t filas, size_t columnas)
41  {//-------------------------------------------------------
42    int Suma=0;
43
44    for(size_t f=0; f<filas; f++)
45      Suma+=M[f][f];
46    return Suma;
47  }
48
```

Continúa …. Main.cpp

```
48
49  int SumaPorEncimadelaDiagonal(int M[][nCOLUMNAS], size_t filas, size_t columnas)
50  {//-----------------------------------------------------------------------------
51    int Suma=0;
52
53    for(size_t f=0; f<filas-1; f++)
54      for (size_t c = f+1; c < columnas; c++)
55        Suma+=M[f][c];
56    return Suma;
57  }
```

M

	0	1	2	3	4	5	6	7
0	22	45	67	12	34	21	3	56
1	21	45	65	76	77	89	99	88
2	5	65	34	52	21	33	45	23
3	21	56	78	80	32	45	22	27
4	45	67	56	23	45	4	43	87
5	55	23	23	4	12	3	23	65
6	78	45	45	12	34	23	22	43
7	12	23	28	33	65	12	12	44

Ejemplo: 2

Desarrolle un programa que permita multiplicar dos matrices cuyo orden es de 5 por 5.

Los valores para la matriz 1 y la matriz 2, se generarán aleatoriamente con números entre el 0 y el 99.

El algoritmo para multiplicar matrices:

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1(3)+0(2)+2(1) & 1(1)+0(1)+2(0) \\ -1(3)+3(2)+1(1) & -1(1)+3(1)+1(0) \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

continúa

```
Primera Matriz

     14     65     63     21     60
     23     50     17     63     34
     57     17     91     31     14
     94     50     36     36     13
     12      9     85     91      7

Segunda Matriz

     96     88     75     36     16
     25     11     81     67      6
     76     58     49     33     24
     27     40     31     34     50
     98      0     39     50     24

Tercera Matriz

  14204   6441  12393  10652   4616
   9783   6080   9887   8581   5042
  15022  11721  11618   7948   5084
  15256  12350  14487   9796   4780
  10980   9725   8888   7284   7004
```

main. cpp

```cpp
#include <iostream>
#include <ctime>
#include "Matrices.h"

using namespace std;

int main()
{
  Tipo M1[nFILAS][nCOLUMNAS], M2[nFILAS][nCOLUMNAS];
  Tipo M3[nFILAS][nCOLUMNAS];
  size_t nfilasM1, ncolM1, nfilasM2, ncolM2, nfilasM3, ncolM3;

  //-- si se modifica estos valores y las constantes que definen
  //-- el tamaño de la matriz, se podria cambiar el orden de las
  //-- de las matrices manualmente con pocos cambios en el programa,
  //-- sin embargo para un mejor trabajo, este programa se resolverá
  //-- de diferente manera cuando se usen punteros y asignación dinámica de memoria
  nfilasM1 = nFILAS;
  ncolM1   = nCOLUMNAS;
  nfilasM2 = nFILAS;
  ncolM2   = nCOLUMNAS;

  srand(time(nullptr));
  GenerarMatriz(M1,nfilasM1, ncolM1);
  GenerarMatriz(M2,nfilasM2, ncolM2);
  cout<<"\n";
```

continúa

main. cpp

```
26      cout<<"\n";
27      cout <<"Primera Matriz \n\n";
28      ImprimirMatriz(M1,nfilasM1, ncolM1);
29      cout<<"\n";
30      cout <<"Segunda Matriz \n\n";
31      ImprimirMatriz(M2,nfilasM2, ncolM2);
32      if( ncolM1!=nfilasM2)
33          cout <<"Imposible multiplicar las matrices ";
34      else
35      {
36          MultiplicarMatrices(M1,nfilasM1, ncolM1, M2,nfilasM2, ncolM2,
37                              M3,nfilasM3, ncolM3);
38       cout << "\n";
39       cout << "Tercera Matriz\n";
40       ImprimirMatriz(M3,nfilasM3, ncolM3);
41      }
42      return 0;
43  }
```

Matrices. h

```
1   //
2   // Created by Maria Hilda Bermejo on 9/17/18.
3   //
4
5   #ifndef EJEMPLO_2_MULTIPLIACION_DE_MATRICES_MATRICES_H
6   #define EJEMPLO_2_MULTIPLIACION_DE_MATRICES_MATRICES_H
7
8   #include <iostream>
9   #include <cstddef>
10  #include <stdlib.h>
11  #include <iomanip>
12
13  typedef int Tipo;
14
15  constexpr size_t  nFILAS=5, nCOLUMNAS=5;
16
17  void GenerarMatriz(Tipo M[][nCOLUMNAS], size_t nfilas,  size_t ncol);
18  void ImprimirMatriz(Tipo M[][nCOLUMNAS], size_t nfilas,  size_t ncol);
19  void MultiplicarMatrices(Tipo M1[][nCOLUMNAS], size_t nfilasM1, size_t ncolM1,
20                           Tipo M2[][nCOLUMNAS], size_t nfilasM2, size_t ncolM2,
21                           Tipo M3[][nCOLUMNAS], size_t &nfilasM3, size_t &ncolM3);
22
23  #endif //EJEMPLO_2_MULTIPLIACION_DE_MATRICES_MATRICES_H
24
```

Matrices. cpp

```cpp
//
// Created by Maria Hilda Bermejo on 9/17/18.
//
#include "Matrices.h"
using namespace std;


void GenerarMatriz(Tipo M[][nCOLUMNAS], size_t nfilas,  size_t ncol)
{//-------------------------------------------------------------
  for(size_t contFilas=0; contFilas<nfilas; contFilas++)
    for(size_t contColumnas=0; contColumnas<ncol; contColumnas++)
      M[contFilas][contColumnas] = rand()%100;
}

void ImprimirMatriz(Tipo M[][nCOLUMNAS], size_t nfilas,  size_t ncol)
{//-------------------------------------------------------------
  for(size_t contFilas=0; contFilas<nfilas; contFilas++)
  {
    for (size_t contColumnas = 0; contColumnas < ncol; contColumnas++)
      cout << setw(7) << M[contFilas][contColumnas];
    cout <<"\n";
  }

}

```

Matrices. cpp

```
25
26  void MultiplicarMatrices(Tipo M1[][nCOLUMNAS], size_t nfilasM1, size_t ncolM1,
27                           Tipo M2[][nCOLUMNAS], size_t nfilasM2, size_t ncolM2,
28                           Tipo M3[][nCOLUMNAS], size_t &nfilasM3, size_t &ncolM3)
29  {//-------------------------------------------------------------------------
30    //  M3 = M1 * M2
31    for (int f = 0; f < nfilasM1; ++f)
32    {
33      for (int c = 0; c < ncolM2; ++c)
34      {
35        M3[f][c] = 0;
36        for (int k = 0;k < ncolM1; k++){
37          M3[f][c] += M1[f][k]*M2[k][c];
38        }
39      }
40    }
41    nfilasM3=nfilasM1;
42    ncolM3 = ncolM2;
43  }
```

Ejemplo: 3

El siguiente programa muesta el comportamiento de los alumnos del curso de Programación Orientada a Objetos 1, cuando ingresan a clases en el auditorio de UTEC.

Para efectos de la simulación, los asientos del auditorio se han representado en dos matrices cuadradas de 15 x 15.
Luego que los alumnos se sientan, con la finalidad de facilitar el acceso a los asientos disponibles, los alumnos se desplazan y ocupan los asientos de la parte central de cada fila.
Finalmente, se ha realizado el conteo del número de asientos disponibles en cada fila y en cada matriz.
Tal y como se puede apreciar en la siguiente diapositiva

```
1  1  1  0  0  1  1  0  0  0  1  1  0  1  0
1  0  1  0  0  1  0  1  1  0  0  1  1  1  0
1  1  0  1  0  0  1  1  1  0  0  1  0  1  0
1  0  1  1  1  0  1  0  0  0  1  0  1  0  0
1  0  0  1  0  0  0  0  1  0  1  1  0  0  0
1  1  0  1  1  1  1  1  1  0  0  1  1  1  1
1  0  1  1  1  0  1  1  0  0  1  1  1  1  0
0  0  0  0  1  1  0  0  0  0  1  0  0  0  0
0  1  0  1  0  0  0  0  0  1  1  1  0  1  1
1  1  0  0  0  1  1  1  0  0  0  1  0  0  0
1  0  1  1  1  0  0  0  0  0  1  1  0  1  0
1  0  0  1  1  0  0  0  0  1  1  1  1  1  1
0  1  1  0  0  1  1  0  1  1  0  1  0  1  1
0  1  1  1  0  0  1  0  0  0  0  1  0  0  1
1  1  0  0  1  0  0  1  1  0  0  1  1  1  1
```

```
1  0  0  1  1  1  0  0  0  1  1  1  1  0  0
0  0  1  0  0  1  0  0  1  0  0  1  0  1  1
1  0  0  0  1  1  0  1  1  1  1  0  1  1  1
0  0  1  0  1  1  0  1  1  0  0  0  1  1  1
1  1  0  0  0  0  0  0  0  0  1  1  1  0  0  0
0  0  0  0  0  1  0  0  1  1  1  1  0  0  0
0  1  0  1  0  1  0  0  0  1  0  0  0  1  1
0  0  1  1  0  0  0  0  0  0  0  0  0  1  0
1  1  0  0  0  1  0  1  0  1  1  1  1  0  1
0  0  1  0  1  1  1  1  1  1  0  1  0  0  0
1  0  0  0  0  0  1  0  1  1  1  0  0  1  0
0  1  1  1  0  1  0  1  1  0  0  1  0  0  0
1  0  1  0  0  0  1  0  1  0  1  1  0  1  0
1  0  1  1  1  1  1  1  1  0  1  1  0  1  0
1  0  0  0  1  1  1  1  0  1  1  1  0  0  1
```

```
0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  [7]
0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  [7]
0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  [7]
0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  [8]
0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  [10]
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  [3]
0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  [5]
0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  [12]
0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  [8]
0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  [9]
0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  [8]
0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  [6]
0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  [6]
0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  [9]
0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  [6]
```

```
1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  [7]
1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  [7]
1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  [7]
1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  [8]
1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  [10]
1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  [3]
1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  [5]
1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  [12]
1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  [8]
1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  [9]
1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  [8]
1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  [6]
1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  [6]
1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  [9]
1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  [6]
```

main.cpp

```cpp
#include <iostream>
#include <ctime>
#include "Matrices.h"

using namespace std;

int main()
{
  srand(time(nullptr));
  // Es para una matrix cuadrada
  valoresDeMatriz matrizA[s][s];
  valoresDeMatriz matrizB[s][s];
  //------------------------
  inicializarMatriz(matrizA);
  inicializarMatriz(matrizB);
  //---- imprime la primera fila tanto de la MatrizA, como de la Matriz B
  cout <<"\n\n";
  imprimirMatriz(matrizA, matrizB);
  //------mueve los valores hacia un extremo
  organizarMatriz(matrizA, true); //--- mueve hacia la derecha
  organizarMatriz(matrizB, false); //-- mueve hacia la izquierda
  //--- Imprime las matrices despues de la reorganización
  imprimirMatriz(matrizA, matrizB, true);
  return 0;
}
```

Matrices.h

```
1
2   #ifndef KORGANIZARMATRIZ_MATRICES_H
3   #define KORGANIZARMATRIZ_MATRICES_H
4
5   #include <iostream>
6   #include <stdlib.h>
7   #include <iomanip>
8
9   constexpr int s = 15;
10  typedef int valoresDeMatriz;
11
12  void inicializarMatriz(valoresDeMatriz matriz[][s]);
13  void organizarMatriz(valoresDeMatriz matriz[][s], bool flag);
14  void imprimirMatriz(valoresDeMatriz matrizA[][s], valoresDeMatriz matrizB[][s], bool s = false);
15
16
17  #endif //KORGANIZARMATRIZ_MATRICES_H
18
```

Matrices.cpp

```cpp
#include "Matrices.h"

using namespace std;

void inicializarMatriz(valoresDeMatriz matriz[][s])
{//----------------------------------------
  // 1 significa que hay alguien sentado
  // 0 significa que es una asiento vacio

  for (int filas = 0; filas < s; ++filas){
    for (int columnas = 0; columnas < s; ++columnas){
      matriz[filas][columnas] = rand() % 2;
    }
  }
}

```

Matrices.cpp

```
16
17  void imprimirMatriz(valoresDeMatriz matrizA[][s], valoresDeMatriz matrizB[][s], bool flag)
18  {//-------------------------------------------------------------------------------
19    int count;
20    for (int filas = 0; filas < s; ++filas){
21      count = 0;
22      for (int columnas = 0; columnas < s; ++columnas){
23        cout << setw(3) << matrizA[filas][columnas] << " ";
24        count += !matrizA[filas][columnas];
25      }
26      if(flag) cout << setw(3) <<"["<<count<<"]"; //-- imprime cantidad de asientos disponibles por fila
27
28      count = 0;
29      cout << "\t\t";
30      for (int columnas = 0; columnas < s; ++columnas){
31        cout << setw(3)<< matrizB[filas][columnas] << " ";
32        count += !matrizA[filas][columnas];
33      }
34      if(flag)  cout << setw(3)<<"["<<count<<"]";//-- imprime cantidad de asientos disponibles por fila
35
36      cout << "\n";
37    }
38    cout << "\n";
39  }
```

Matrices.cpp

```
40
41
42  void organizarMatriz(valoresDeMatriz matriz[][s], bool flag)
43  {//-------------------------------------------------------
44    //--- si flag es true, mueve los unos a la derecha
45    //--- si flag es false, mueve los unos a la izquierda
46    for (int filas = 0; filas < s; ++filas){
47      for (int columnas_inicio = 0, columnas_fin = s-1; columnas_inicio < columnas_fin; ){
48        if(matriz[filas][columnas_inicio] == flag && matriz[filas][columnas_fin] == !flag){
49          valoresDeMatriz temporal = matriz[filas][columnas_inicio];
50          matriz[filas][columnas_inicio++] = matriz[filas][columnas_fin];
51          matriz[filas][columnas_fin--] = temporal;
52        }
53        if(matriz[filas][columnas_inicio] == !flag)
54          columnas_inicio++;
55
56        if(matriz[filas][columnas_fin] == flag)
57          columnas_fin--;
58      }
59    }
60  }
```

El Código de los programas lo pueden ubicar en este link:

http://bit.ly/2MKbvb8

SÁB 22, 14:00 - 18:00. A705

coding dojo

@acmutec

> Aprende, refuerza, desafíate.

> Desarrolla tu **pensamiento computacional**.

Inscripción:
https://goo.gl/oZZxfE

CS1102 – PROGRAMACIÓN ORIENTADA A OBJETOS 1
CICLO 2018-2

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Unidad 5: Arrays de arrays

http://bit.ly/2p3fgiD

Profesores:

Ernesto Cuadros-Vargas, PhD. *ecuadros@utec.edu.pe*

María Hilda Bermejo, M. Sc. *mbermejo@utec.edu.pe*

http://bit.ly/2xhGark