

Algoritmos y

Estructuras

De

Datos

Estructuras de Datos

ARREGLOS MULTIDIMENSIONALES

ESTRUCTURAS DE DATOS

- **Simples o básicos:** caracteres, reales, flotantes.
- **Estructurados:** colección de valores relacionados. Se caracterizan por el tipo de dato de sus elementos, la forma de almacenamiento y la forma de acceso.

- **Estructuras estáticas:** Su tamaño en memoria se mantiene inalterable durante la ejecución del programa, y ocupan posiciones fijas.

ARREGLOS - CADENAS - ESTRUCTURAS

- **Estructuras dinámicas:** Su tamaño varía durante el programa y no ocupan posiciones fijas.

LISTAS - PILAS - COLAS - ARBOLES - GRAFOS

ESTRUCTURAS DE DATOS: Clasificaciones

- Según donde se almacenan
 - Internas** (en memoria principal)
 - Externas** (en memoria auxiliar)
- Según tipos de datos de sus componentes
 - Homogéneas** (todas del mismo tipo)
 - No homogéneas** (pueden ser de distinto tipo)
- Según la implementación
 - Provistas por los lenguajes** (básicas)
 - Abstractas** (TDA - Tipo de dato abstracto que puede implementarse de diferentes formas)
- Según la forma de almacenamiento
 - Estáticas** (ocupan posiciones fijas y su tamaño nunca varía durante todo el módulo)
 - Dinámicas** (su tamaño varía durante el módulo y sus posiciones también)

Arreglos Multidimensionales

- Son arreglos que permiten varios índices.
- A los arreglos **bidimensionales** se los llama comúnmente matrices, tablas o arreglos de arreglos.
- A los de **3 dimensiones**, se los denomina cubos
- Y al resto, en general, se los denomina multidimensionales.

Declaración de Arreglos Bidimensionales:

- Formato de declaración:

tipo_dato *nombre-del-arreglo* [nroDeFilas] [nroDeColumnas];

Ejemplo: Definir un arreglo bidimensional de 2 x 3 enteros:

```
int Matriz [2] [3];
```

Arreglos Multidimensionales

Declaración utilizando tipo estructurado:

- Formato de declaración:

```
typedef tipo_dato nombre-del-arreglo [dim1] [dim2];
```

Ejemplo: Definir un arreglo bidimensional de 2 x 3 enteros:

```
typedef int tMatriz [2] [3];
```

Creación de variables:

- Formato de declaración:

```
tipo_dato nombre-variable;
```

Ejemplo: **tMatriz m1;**

Arreglos Multidimensionales

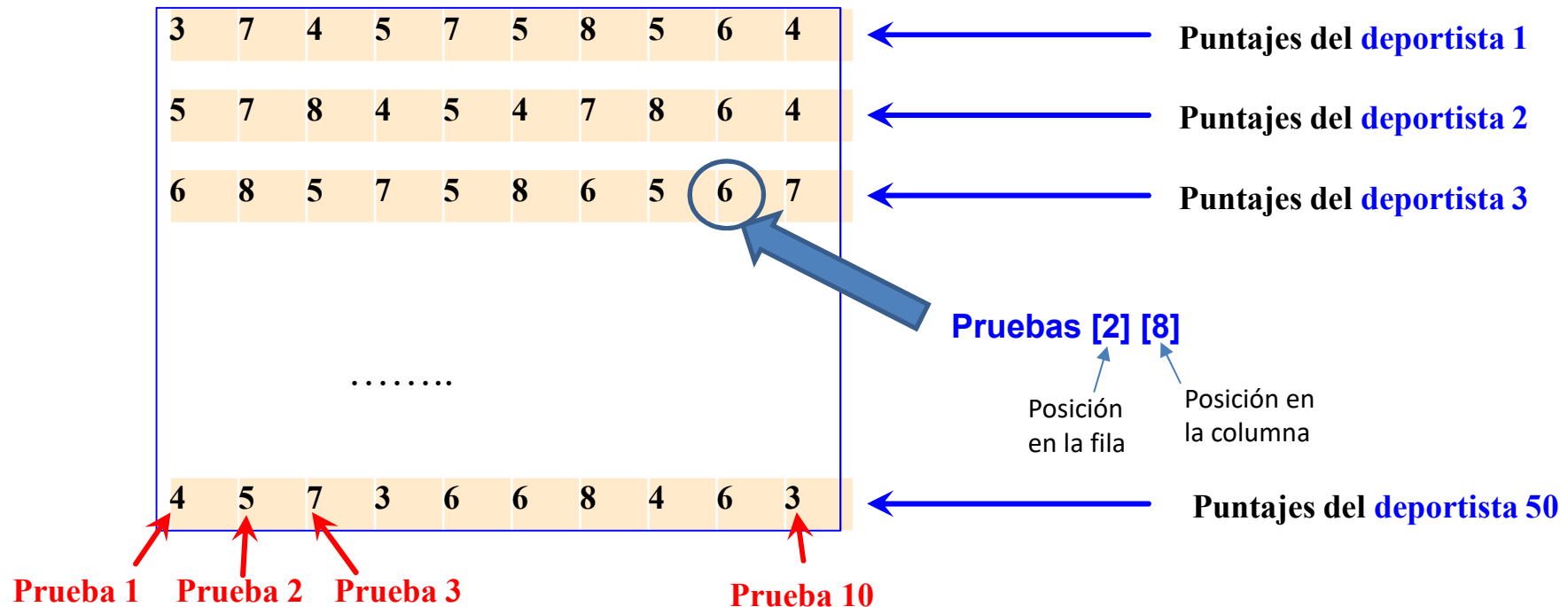
EJEMPLO 1:

- Si se deben almacenar los puntajes de un deportista en 10 pruebas:

`int Pruebas [10] ;` ← **Arreglo**

- Si se deben almacenar los puntajes de 50 deportistas en 10 pruebas:

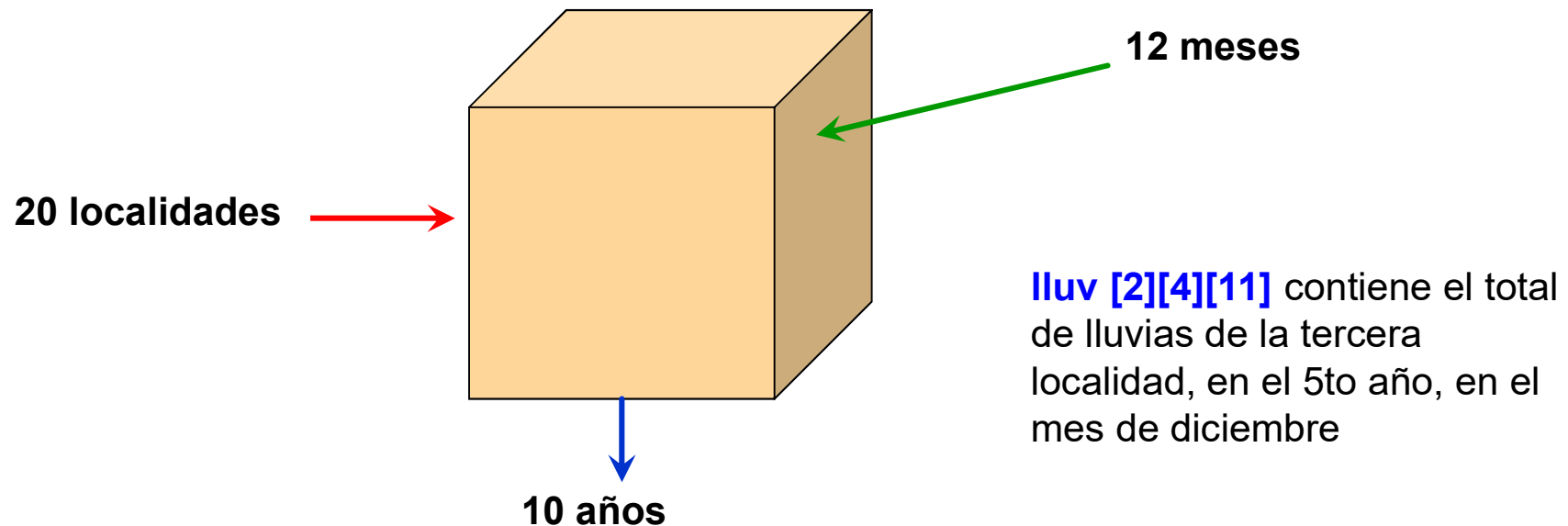
`int Pruebas [50] [10];` ← **Matriz**



Arreglos Multidimensionales

- Las dimensiones pueden ser múltiples.
- Si se desea almacenar los totales mensuales de lluvias mensuales en 20 localidades, durante cada uno de los meses de una década:

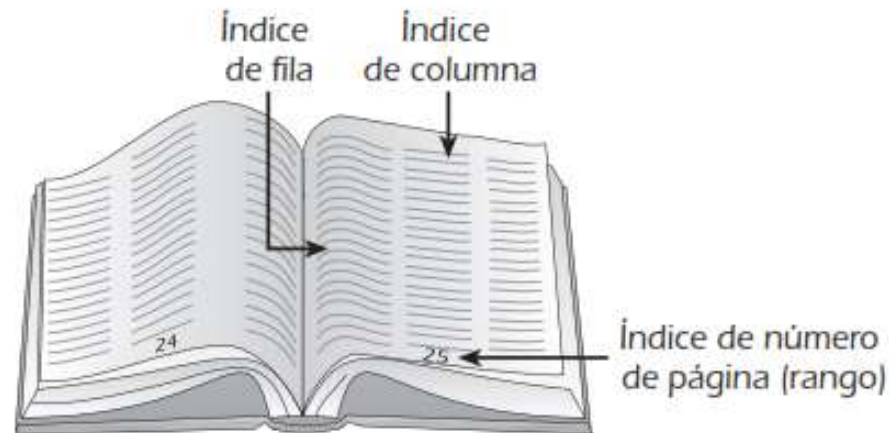
```
float lluv [20] [10] [12];
```



C++ proporciona la posibilidad de almacenar varias dimensiones, aunque raramente los datos del mundo real requieren más de dos o tres dimensiones.

Arreglos Multidimensionales

- Un arreglo tridimensional puede verse como un libro de tablas de datos.
- El 1er índice puede considerarse como la ubicación de la fila deseada en una tabla, el 2do índice como la columna deseada y el 3er índice, como el número de página de la tabla seleccionada.



Del mismo modo, un **arreglo tetradimensional** puede representarse como un estante de libros, donde la 4ta dimensión se usa para declarar un libro deseado en el estante, y un **arreglo pentadimensional** puede verse como una biblioteca llena de libros en el que la 5ta dimensión se refiere al estante seleccionado en la biblioteca.

Arreglos Multidimensionales

Inicialización:

- **Por medio de asignaciones:**

```
m1[0][0] = 0; m1[0][1] = 0; m1[0][2] = 0;
```

```
m1[1][0] = 0; m1[1][1] = 0; m1[1][2] = 0;
```

- **Enumerando sus elementos en orden de sus dimensiones:**

```
tMatriz m1 = { {0,0,0}, {0,0,0} };
```

```
int m1[2][3] = { {0,0,0}, {0,0,0} };
```

```
int m1[ ][3] = { {0,0,0}, {0,0,0} };
```

Las llaves separan filas individuales

En este caso se omite la longitud de la primera componente (sólo de la primera, no de todas).

Arreglos Multidimensionales

Se pueden definir arreglos con componentes de otros arreglos previamente definidos.

- Ejemplos:

```
typedef char tAlumno[15]; // 15 caracteres
```

```
typedef tAlumno tClase[85]; // 85 componentes del tipo tAlumno
```

```
typedef int tDatos[3]; // 3 enteros
```

```
typedef tDatos tDatosSemanales[5]; // 5 componentes del tipo tDatos
```

```
tDatosSemanales dat1 = { {9,13,10}, {10,4,8}, {8,19,7}, {16,21,5},  
                        {18,20,19} };
```

Se pueden definir tablas como *constantes*, de la misma manera que se hace con los tipos elementales.

- Ejemplos:

```
const int tab2[3][2] = { {0, 2}, {4, 4}, {10, 10} };
```

```
const int tab4[ ][2] = { {0, 2}, {4, 4}, {10, 10} };
```

Pasaje de arreglos bidimensionales como parámetros

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <iomanip>
using namespace std;
```

```
const int DIM1=10;
const int DIM2=10;
```

```
void cargarMatriz(int m[][DIM2], int cf, int cc);
void mostrarMatriz(int m[][DIM2], int cf, int cc);
```

```
int main(){
    int matriz[DIM1][DIM2];
    int cf=4, cc=3;
    cargarMatriz(matriz, cf, cc);
    mostrarMatriz(matriz, cf, cc);
    return 0;
}
```

```
10  2  6
10  8  10
3   1  8
1   1  5
```

```
void cargarMatriz(int m[][DIM2], int cf, int cc){
    //carga la matriz por filas
    int f, c;
    srand(time(NULL));
    for (f=0; f<cf; f++)
        for (c=0; c<cc; c++)
            m[f][c] = rand()%10+1;
}
```

Si se inserta un número dentro de los corchetes el compilador lo ignorará

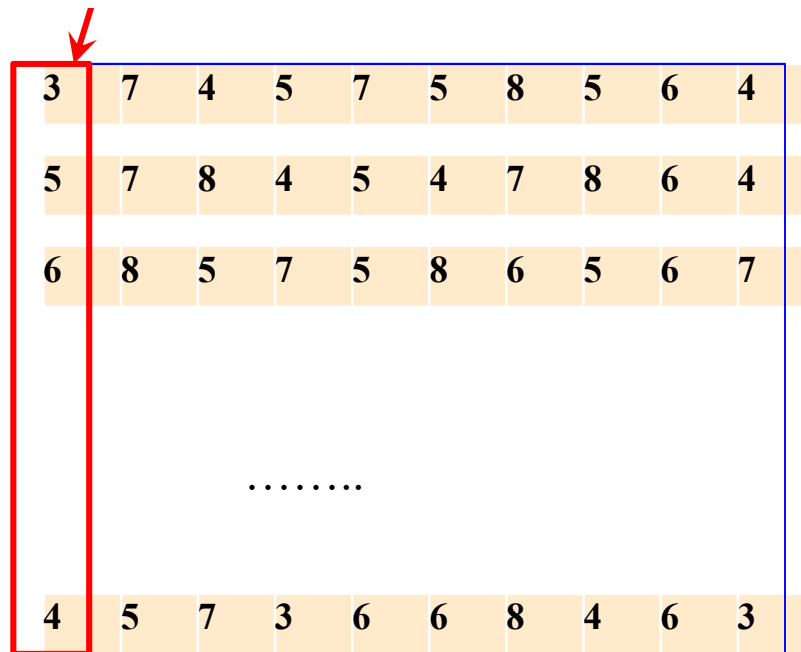
```
void mostrarMatriz(int m[][DIM2], int cf, int cc){
    int f, c;
    cout << endl;
    for (f=0; f<cf; f++){
        for (c=0; c<cc; c++)
            cout << setw(2) << m[f][c] << " ";
        cout << endl;
    }
}
```

Arreglos Bidimensionales

EJEMPLO 1:

1. Informar cuál fue el puntaje promedio de la primera prueba.

Puntajes en la prueba 1



3	7	4	5	7	5	8	5	6	4
5	7	8	4	5	4	7	8	6	4
6	8	5	7	5	8	6	5	6	7
.....									
4	5	7	3	6	6	8	4	6	3

```
int matriz[DIM1][DIM2];
```

```
int suma=0, max=0, puntos=0, depomax;
for (int f=0; f<DIM1; f++)
.....
    suma += matriz[f][0];
cout << endl << "Promedio de 1era prueba: " << float(suma)/DIM1;
```

Arreglos Bidimensionales

EJEMPLO 1:

2. Informar cuál fue el mayor valor obtenido en alguna prueba de las últimas 3.

Puntajes de las 3 últimas pruebas

3	7	4	5	7	5	8	5	6	4
5	7	8	4	5	4	7	8	6	4
6	8	5	7	5	8	6	5	6	7
.....									
4	5	7	3	6	6	8	4	6	3

```
int matriz[DIM1][DIM2];
```

```
for (int f=0; f<DIM1; f++)  
    for (int c=DIM2-3; c<DIM2; c++)  
        if (matriz[f][c] > max) max = matriz[f][c];  
cout << endl << "El mayor puntaje obtenido en una prueba ";  
cout << "de las ultimas 3 es: " << max;
```

Arreglos Bidimensionales

EJEMPLO 1.

El puntaje final de cada Deportista se calcula como el promedio de los puntajes de todas las pruebas.

3. Informar cuál fue el deportista con mayor puntaje final.

```
int matriz[DIM1][DIM2];
```

Puntaje final deportista 1

```
float promedio;  
max = 0;  
for (int f=0; f<cf; f++){  
    puntos = 0;  
    for (int c=0; c<cc; c++){  
        puntos += matriz[f][c];  
    }  
    promedio = float (puntos)/cc;  
    if (promedio > max){  
        max = promedio;  
        depomax = f+1;  
    }  
}
```

3	7	4	5	7	5	8	5	6	4
5	7	8	4	5	4	7	8	6	4
6	8	5	7	5	8	6	5	6	7
.....									
4	5	7	3	6	6	8	4	6	3

```
cout << endl << "El deportista con mayor puntaje final es el " << depomax;
```

Calcular la transpuesta de una matriz

EJEMPLO 2.

```
const int TF=10;
void cargarmatriz(int m[][TF], int tam);
void mostrarmatriz(int m[][TF], int tam);
void transponermatriz(int m[][TF], int tam);

int main() {
    int matriz[TF][TF];
    int i, j, N;
    cout << "Ingreso tamaño (hasta 10): ";
    cin >> N;
    cargarmatriz(matriz, N);
    cout << endl << "Matriz:" << endl;
    mostrarmatriz(matriz, N);
    transponermatriz(matriz, N);
    cout << endl << "Matriz transpuesta:" << endl;
    for (i=0; i<N; i++) {
        cout << endl;
        for (j=0; j<N; j++)
            cout << setw(2) << matriz[i][j] << "--";
    }
    return 0;
}
```

```
void transponermatriz(int m[][TF], int tam) {
    int i, j, aux;

    for (i=0; i<tam; i++)
        for (j=0; j<i; j++) {
            aux = m[i][j];
            m[i][j] = m[j][i];
            m[j][i] = aux;
        }
    return;
}
```

```
Ingreso tamaño (hasta 10): 4
Matriz:
22  93  92  41
51  16  87  19
13  42  99  89
53  98  91  23

Matriz transpuesta
22--51--13--53--
93--16--42--98--
92--87--99--91--
41--19--89--23--

<< El programa ha finalizado: c
```


Obtener diferentes datos de una matriz

EJEMPLO 3.

```
const int TF=5;
void mostrar_datos(int mat[][TF], int tl);
void mostrar(int mat[][TF], int tl);
int Mayor_de_matriz(int mat[][TF], int tl);
int Mayor_de_diagonal_princ(int mat[][TF], int tl);
int Mayor_de_diagonal_sec(int mat[][TF], int tl);
int Mayor_de_una_fila(int mat[], int tl);
int Mayor_de_una_col(int mat[][TF], int tl, int c);
int Columna_con_mayor_suma(int mat[][TF], int tl);
float Suma_columna_promedio(int mat[][TF], int tl);
```

Obtener diferentes datos de una matriz

EJEMPLO 3.

```
int main() {
    int matriz[TF][TF] = {{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}},
        matriz1[TF][TF] = {{0,1,2,3,4},{1,0,2,3,4},{2,2,0,3,4},{3,3,3,0,4},{4,4,4,4,0}},
        matriz2[TF][TF] = {{0,0,0,0,0},{0,0,0,0,0},{0,0,1,0,0},{0,0,0,0,0},{0,0,0,0,0}},
        matriz3[TF][TF] = {{0,0,0,0,3},{0,0,0,2,2},{0,0,1,0,0},{0,0,0,0,0},{0,0,0,1,1}},
        matriz4[TF][TF] = {{1,0,0,0,0},{0,1,0,0,0},{0,0,1,0,0},{0,0,0,2,0},{0,0,0,0,3}},
        matriz5[TF][TF] = {{1,1,1,1,1},{2,0,0,0,2},{3,0,1,0,3},{4,0,0,0,4},{5,5,5,5,5}};

    mostrar_datos(matriz, TF);
    mostrar_datos(matriz1, TF);
    mostrar_datos(matriz2, TF);
    mostrar_datos(matriz3, TF);
    mostrar_datos(matriz4, TF);
    mostrar_datos(matriz5, TF);

    return 0;
}
```

La matriz:

1	1	1	1	1
2	0	0	0	2
3	0	1	0	3
4	0	0	0	4
5	5	5	5	5

El mayor elemento de la matriz es: 5

El mayor elemento de la diagonal principal es: 5

El mayor elemento de la diagonal secundaria es: 5

El mayor elemento de la columna 2 es: 5

El mayor elemento de la fila 3 es: 4

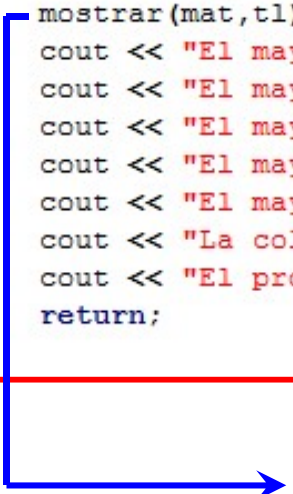
La columna de mayor suma es: 0

El promedio de las sumas de columnas es: 6.8

Obtener diferentes datos de una matriz

EJEMPLO 3.

```
void mostrar_datos(int mat[][TF], int tl){  
    int fila=3, col=2;  
    mostrar(mat,tl);  
    cout << "El mayor elemento de la matriz es: " << Mayor_de_matriz(mat,tl) << endl;  
    cout << "El mayor elemento de la diagonal principal es: " << Mayor_de_diagonal_princ(mat,tl) << endl;  
    cout << "El mayor elemento de la diagonal secundaria es: " << Mayor_de_diagonal_sec(mat,tl) << endl;  
    cout << "El mayor elemento de la columna " << col << " es: " << Mayor_de_una_col(mat,tl,col) << endl;  
    cout << "El mayor elemento de la fila " << fila << " es: " << Mayor_de_una_fila(mat[fila],tl) << endl;  
    cout << "La columna de mayor suma es: " << Columna_con_mayor_suma(mat,tl) << endl;  
    cout << "El promedio de las sumas de columnas es: " << Suma_columna_promedio(mat,tl) << endl;  
    return;  
}
```



```
void mostrar(int mat[][TF], int tl){  
    int i, j;  
    cout << " -----" << endl;  
    cout << endl << "La matriz: " << endl << endl;  
    for (i=0; i<tl; i++){  
        for (j=0; j<tl; j++){  
            cout << mat[i][j] << " ";  
            cout << endl << endl;  
        }  
    }  
    return;  
}
```

Ejemplo 3

```
int Mayor_de_matriz(int mat[][TF], int tl){
    int i, j, max = mat[0][0];
    for (i=0; i<tl; i++)
        for (j=0; j<tl; j++)
            if (mat[i][j] > max) max = mat[i][j];
    return max;
}
```

Devuelve el mayor elemento de la matriz

```
int Mayor_de_diagonal_princ(int mat[][TF], int tl){
    int i, max = mat[0][0];
    for (i=1; i<tl; i++)
        if (mat[i][i] > max) max = mat[i][i];
    return max;
}
```

Devuelve el mayor elemento de la diagonal principal

Ejemplo 3

```
int Mayor_de_diagonal_sec(int mat[][TF], int tl){
    int i, max = mat[0][tl-1];
    for (i=1; i<tl; i++)
        if (mat[i][tl-1-i] > max) max = mat[i][tl-1-i];
    return max;
}
```

Devuelve el mayor elemento de la diagonal secundaria

```
int Mayor_de_una_col(int mat[][TF], int tl, int c){
    int i, max = mat[0][c];
    for (i=1; i<tl; i++)
        if (mat[i][c] > max) max = mat[i][c];
    return max;
}
```

Devuelve el mayor elemento de una columna determinada

```
cout << "El mayor elemento de la columna " << col << " es: " <<
Mayor_de_una_col(mat,tl,col);
```

Ejemplo 3

```
int Mayor_de_una_fila(int mat[], int tl) {  
    int i, max = mat[0];  
    for (i=1; i<tl; i++)  
        if (mat[i] > max) max = mat[i];  
    return max;  
}
```

Devuelve el mayor elemento de una fila determinada

```
cout << "El mayor elemento de la fila " << fila << " es: " <<  
Mayor_de_una_fila (mat[fila], tl);
```

Se le pasa una fila (un vector) como parámetro



Ejemplo 3

```
int Columna_con_mayor_suma(int mat[][TF], int tl){
    int i, j, max, colmax, totc=0;
    for (i=0; i<tl; i++)
        totc += mat[i][0];
    max = totc;
    colmax = 0;
    for (j=1; j<tl; j++){
        totc = 0;
        for (i=0; i<tl; i++)
            totc += mat[i][j];
        if (totc > max){
            max= totc;
            colmax = j;
        }
    }
    return colmax;
}
```

Devuelve el índice de la columna que tiene mayor suma de sus elementos

Ejemplo 3

```
float Suma_columna_promedio(int mat[][TF], int tl){
    int i, j, totc=0, sum=0;
    for (j=1; j<tl; j++){
        totc = 0;
        for (i=0; i<tl; i++)
            totc += mat[i][j];
        sum += totc;
    }
    return (float) sum/tl;
}
```

Devuelve el promedio de las sumas de columnas

Distintas características de una matriz

EJEMPLO 4.

```
const int TF=5;
void caracterizar (int mat[][TF], int tl);
void mostrar (int mat[][TF], int tl);
bool Es_nula (int mat[][TF], int tl);
bool Es_simetrica (int mat[][TF], int tl);
bool Es_diagonal_nula (int mat[][TF], int tl);
bool Es_triangular_superior (int mat[][TF], int tl);
bool Es_solo_diagonal (int mat[][TF], int tl);
bool Es_marco (int mat[][TF], int tl);
bool Tiene_fila_nula (int mat[][TF], int tl);
bool Tiene_columna_nula (int mat[][TF], int tl);
```

```
void caracterizar (int mat[][TF], int tl) {
    mostrar(mat,tl);
    if (Es_nula (mat,tl)) cout << "La matriz es nula " << endl;
    if (Es_simetrica (mat,tl)) cout << "La matriz es simetrica " << endl;
    if (Es_diagonal_nula (mat,tl)) cout << "La matriz tiene diagonal nula " << endl;
    if (Es_triangular_superior (mat,tl)) cout << "La matriz es triangular superior " << endl;
    if (Es_solo_diagonal (mat,tl)) cout << "La matriz es solo diagonal " << endl;
    if (Es_marco (mat,tl)) cout << "La matriz es marco " << endl;
    return;
}
```

Distintas características de una matriz

EJEMPLO 4.

```
int main() {
    int m[TF][TF] = {{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}},
        m1[TF][TF] = {{0,1,2,3,4},{1,0,2,3,4},{2,2,0,3,4},{3,3,3,0,4},{4,4,4,4,0}},
        m2[TF][TF] = {{0,0,0,0,0},{0,0,0,0,0},{0,0,1,0,0},{0,0,0,0,0},{0,0,0,0,0}},
        m3[TF][TF] = {{0,0,0,0,3},{0,0,0,2,2},{0,0,1,0,0},{0,0,0,0,0},{0,0,0,1,1}},
        m4[TF][TF] = {{1,0,0,0,0},{0,1,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}},
        m5[TF][TF] = {{1,1,1,1,1},{2,0,0,0,2},{3,0,0,0,3},{4,0,0,0,4},{5,5,5,5,5}};

    caracterizar (m, TF);
    caracterizar (m1, TF);
    caracterizar (m2, TF);
    caracterizar (m3, TF);
    caracterizar (m4, TF);
    caracterizar (m5, TF);

    return 0;
}
```

Ejemplo 4

```
bool Es_nula(int mat[][TF], int tl) {  
    int i=0, j; bool b=true;  
    while((i<tl) && b) {  
        j=0;  
        while ((j<tl) && b) {  
            if (mat[i][j]!=0) b=false;  
            j++;  
        }  
        i++;  
    }  
    return b;  
}
```

Determinar si una matriz es nula: tiene todos sus elementos iguales a cero

La matriz :

```
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0  
0 0 0 0 0
```

La matriz es nula.

Ejemplo 4

```
bool Es_diagonal_nula (int mat[][TF], int tl) {  
    int i=0; bool b=true;  
    while ((i<tl) && b) {  
        if(mat[i][i]!=0) b=false;  
        i++;  
    }  
    return b;  
}
```

Determinar si una matriz tiene la diagonal principal nula: tiene todos los elementos de la diagonal principal iguales a cero

La matriz:

```
0 1 2 3 4  
1 0 2 3 4  
2 2 0 3 4  
3 3 3 0 4  
4 4 4 4 0
```

La matriz es simetrica
La matriz tiene diagonal nula

Ejemplo 4

```
void mostrar (int mat[][TF], int tl) {  
    int i,j;  
    cout << "-----" << endl;  
    cout << endl << "La matriz: " << endl << endl;  
    for (i=0; i<tl; i++) {  
        for(j=0; j<tl; j++)  
            cout << mat[i][j] << " ";  
        cout << endl << endl;  
    }  
}
```

Mostrar una matriz (por filas)

Ejemplo 4

```
bool Es_simetrica (int mat [][][TF], int tl) {  
    int i=0, j; bool b=true;  
    while((i<tl) && b) {  
        j=0;  
        while ((j<i-1) && b) {  
            if (mat[i][j]!=mat[j][i]) b=false;  
            j++;  
        }  
        i++;  
    }  
    return b;  
}
```

Determinar si una matriz es simétrica: todo par de elementos simétricos ((i,j) y (j,i)) son iguales

La matriz:

```
0 1 2 3 4  
1 0 2 3 4  
2 2 0 3 4  
3 3 3 0 4  
4 4 4 4 0
```

La matriz es simetrica

La matriz tiene diagonal nula

Ejemplo 4

```
bool Es_triangular_superior (int mat[][TF], int tl) {
    int i=0, j; bool b=true;
    while((i<tl) && b) {
        j=0;
        while((j<=i-1) && b) {
            if (mat[i][j]!=0) b=false;
            j++;
        }
        i++;
    }
    return b;
}
```

Determinar si una matriz es triangular superior: tiene todos los elementos que están por debajo de la diagonal principal iguales a cero

La matriz:

1 0 0 0 0

0 1 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

La matriz es triangular superior

Ejemplo 4

```
bool Es_solo_diagonal (int mat[][TF], int tl) {  
    int i=1;  
    bool b=true;  
    if (b) {  
        i=0;  
        while ((i<tl) && b) {  
            if (mat[i][i]==0) b=false;  
            i++;  
        }  
    }  
    return b;  
}
```

La matriz:

1 0 0 0 0

0 1 0 0 0

0 0 1 0 0

0 0 0 2 0

0 0 0 0 3

La matriz es simetrica

La matriz es triangular superior

La matriz es solo diagonal

Determinar si una matriz es Solo Diagonal: todos los elementos de la diagonal principal son no nulos y el resto es todo nulo.

Ejemplo 4

```
bool Es_marco (int mat[][TF], int tl) {
    int i, j; bool b=true;
    j=0;
    while ((j<tl) && b) {
        if (mat[0][j]==0 || mat[tl-1][j]==0) b=false;
        j++;
    }

    if (b) {
        j=1;
        while ((j<tl-1) && b) {
            if (mat[j][0]==0 || mat[j][tl-1]==0) b=false;
            j++;
        }
    }

    if (b) {
        i=1;
        while ((i<tl-1) && b) {
            j=1;
            while ((j<tl-1) && b) {
                if (mat[i][j]!=0) b=false;
                j++;
            }
            i++;
        }
    }

    return b;
}
```

Determinar si una matriz es marco:

tiene todos los elementos del borde no nulos y los del interior nulos

La matriz :

1	1	1	1	1
2	0	0	0	2
3	0	0	0	3
4	0	0	0	4
5	5	5	5	5

La matriz es marco

Ejemplo 4

```
bool Tiene_fil_suma_cero (int mat[][TF], int tl) {  
    int i=0, j, sum;  
    bool b=false;  
    while (i<tl && !b) {  
        sum=0;  
        for (j=0;j<tl;j++) sum += mat[i][j];  
        if (sum==0) b=true;  
        i++;  
    }  
    return b;  
}
```

Determinar si una matriz tiene al menos una fila de suma cero

```
int Cant_fil_suma_cero (int mat[][TF], int tl) {  
    int i=0, j, cant=0, sum;  
    for(i=0;i<tl;i++) {  
        sum=0;  
        for (j=0;j<tl;j++) sum+=mat[i][j];  
        if (sum==0) cant++;  
    }  
    return cant;  
}
```

Determinar cuantas filas de suma cero tiene una matriz

Revisamos la diferencias

```
void mostrar_datos (int mat[][TF], int tl) {  
    mostrar(mat, tl);  
    if(Tiene_fil_suma_cero(mat, tl))  
        cout << "Tiene al menos una fila de suma cero" << endl;  
    cout << "La cantidad de filas de suma cero es: "  
        << Cant_fil_suma_cero(mat, tl) << endl;  
    return;  
}
```

La matriz :

```
0  1  2  3  4  
1  0  2  3  4  
2  2  0  3  4  
3  3  3  0  4  
4  4  4  4  0
```

La cantidad de filas de suma cero es: 0

La matriz :

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

Tiene al menos una fila de suma cero
La cantidad de filas de suma cero es: 4

Para profundizar en el tema:

Libro:

Resolución de problemas con C++ - Savitch

Capítulo 10

(Pág. 545)

Capítulo 6

Libro: Benjumea-Roldan

Universidad de Málaga