ESTRUCTURAS DE DATOS

Matrices
En Lenguaje C++

Definición

Una matriz, o también llamado arrays de arrays, es una estructura bidimensional de almacenamiento.

Bidimensional significa que utiliza dos dimensiones, filas y columnas para almacenar datos.

Representación grafica (1)

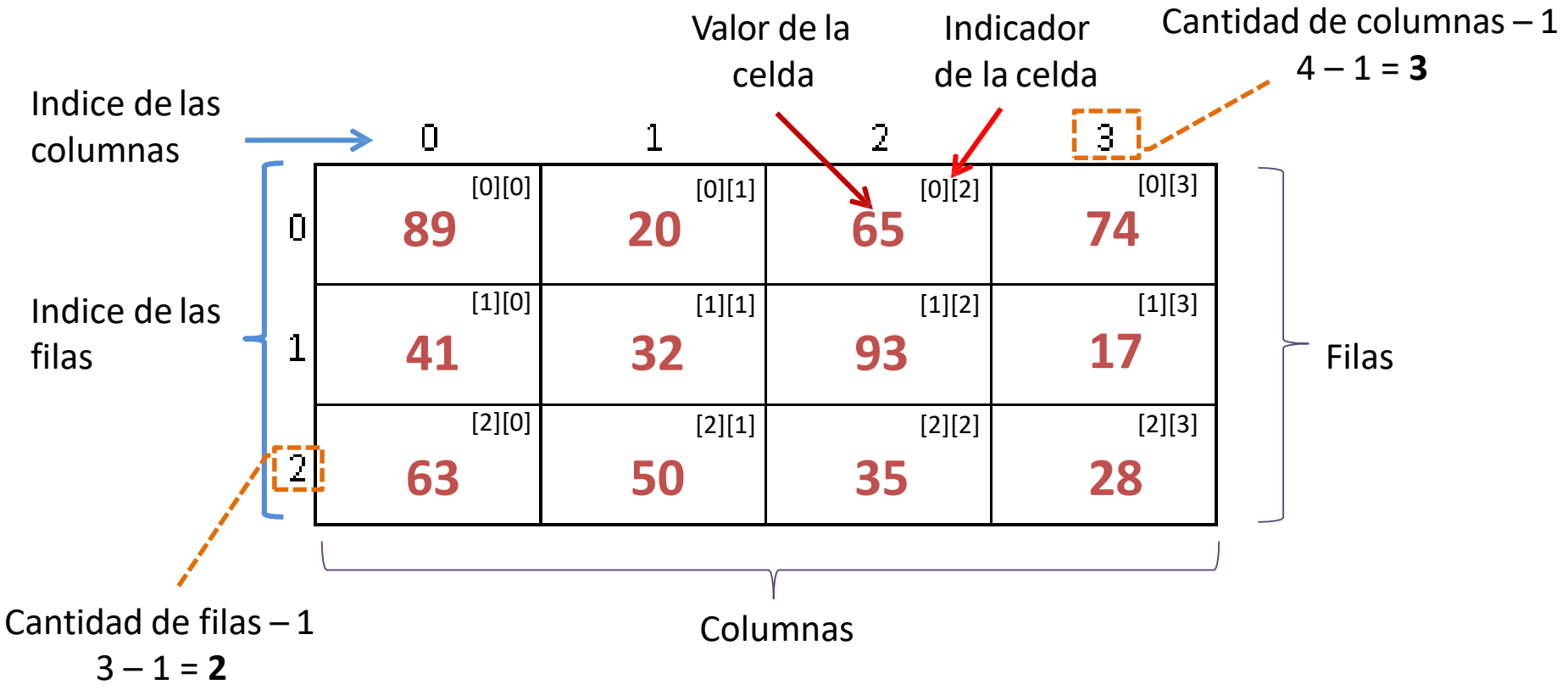
Una matriz se puede representar gráficamente como una serie de *cajones* o *celdas* organizados de forma horizontal (filas) y vertical (columna) en los cuales se almacena un dato.

Cada celda se identifica mediante dos índices. Uno para la fila y otro para la columna.

Tanto el índice de la fila como el de la columna comienzan en cero, hasta la cantidad de filas y columnas -1, respectivamente.

Representación grafica (2)

A continuación se representa una matriz de 3 x 4. Es decir, 3 filas x 4 columnas.



Declaración de una matriz (1)

- Sintaxis en C y C++ #1:

Esta sintaxis es utilizada cuando sabemos cuantos elementos tendrá la matriz

```
tipo_dato nom_matriz[can_f][can_c];
```

Declaración de la matriz | **Asignación de memoria**

Una sola instrucción

Por ejemplo:

```
int enteros[2][2];
```
Ej: **matriz_int_tamano_fijo.cpp**

```
long largos[3][2];
```
Ej: **matriz_long_tamano_fijo.cpp**

```
double reales[2][3];
```
Ej: **matriz_double_tamano_fijo.cpp**

```
string cadenas[2][2];
```
Ej: **matriz_string_tamano_fijo.cpp**

```
char letras[2][2];
```
Ej: **matriz_char_tamano_fijo.cpp**

```
bool logicos[2][3];
```
Ej: **matriz_bool_tamano_fijo.cpp**

Declaración de una matriz (2)

Esta sintaxis es utilizada cuando **no** sabemos cuantos elementos tendrá la matriz

- Sintaxis en C y C++ #2:

```
tipo_dato **nom_matriz;
```

Declaración de la matriz

```
nom_matriz = new tipo_dato *[cant_f];
for(int f = 0; f < cant_f; f++) {
  nom_matriz[f] = new tipo_dato[cant_c];
}
```

Asignación de memoria

Dos instrucciones:

1ra: declarar la matriz
2da: asignar memoria

Esta sintaxis permite declarar la matriz de un tamaño que se desconoce, lo cual da flexibilidad, permitiendo por ejemplo que el usuario ingrese la cantidad de filas y la cantidad de columnas.

Declaración de una matriz (3)

Por ejemplo:

```
int **enteros;
long **largos;
double **reales;
string **cadenas;
char **letras;
boolean **logicos;
```

Ej: **matriz_int_dinamica.cpp**

Ej: **matriz_long_dinamica.cpp**

Ej: **matriz_double_dinamica.cpp**

Ej: **matriz_string_dinamica.cpp**

Ej: **matriz_char_dinamica.cpp**

Ej: **matriz_bool_dinamica.cpp**

```
cout<<"Ingrese la cantidad de filas: ";
cin>>can_filas;
cout<<"Ingrese la cantidad de columnas: ";
cin>>can_columnas;
```

Esta rutina solo muestra la creación de la matriz entera de forma dinámica.
Ejemplo completo:
matriz_int_dinamica.cpp

```
enteros = new int* [cant_est];//cantidad de filas
   for (int i=0; i < cant_est; i++)
         enteros[i] = new int[can_columnas]; //cantidad de columnas
```

Asignación de memoria

Asignación de valores en una matriz

Para almacenar datos en una matriz, se debe *llamar* a la matriz e indicar en que posición o *celda* se desea guardar datos.

Por ejemplo; tomando la matriz para almacenar valores de tipo long, la asignación seria:

```
largos[0][0] = 20180001; //cod estudiantil
largos[0][1] = 210; //cod plan
```

gráficamente la matriz se vería así:

Asignación de valores en una matriz

largos[40][2] =

	0	1
0	20180001	210
1	0	0
2	0	0
....	0	0
39	0	0

Recorrido de una matriz (1)

Para recorrer una matriz, ya bien sea para leerla, asignarla o imprimirla, es necesario utilizar dos ciclos anidados. Por lo general se utiliza el bucle for.

El ciclo *externo* controla la fila que se recorre en la matriz. Y el ciclo *interno* controla la columna.

El recorrido es igual a como se escribe en una hoja cuadriculada. Es decir de derecha a izquierda, de arriba hacia abajo, y un digito (valor) en cada cuadrito (*celda*)

Recorrido de una matriz (2)

Por ejemplo: asignar una matriz de [2][3] con los primeros números pares desde el 2.

```
int pares[2][3]; //usando la sintaxis # 1
int n = 2;

for(int f = 0; f < 2; f++){
   for(int c = 0; c < 3; c++){
      pares[f][c] = n;
      n+=2;
   }
}
```

Recorrido de una matriz (3)

Por ejemplo: imprimir por pantalla la matriz anterior.

```
for(int f = 0; f < 2; f++){
  for(int c = 0; c < 3; c++){
     cout<<pares[f][c]<<“  ”;
   }
   cout<<“\n”;
}
```

Salida

2	4	6
8	10	12