

# Taller de programación sobre matrices y tableros

## Laboratorio Algoritmos y Estructura de Datos I




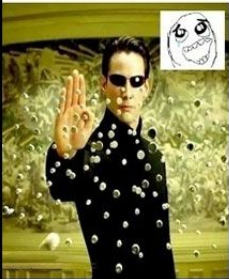

DEPARTAMENTO  
DE COMPUTACION

---

Facultad de Ciencias Exactas y Naturales - UBA

1er Cuatrimestre 2019

That's so 2019

| Teacher   |   |
|---|---|
|  |   |
| Today, we are going to learn about matrix   |   |
| Expectation   | Reality   |
|  | $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}$ $B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix}$ $D = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$  |

## ¿Qué es una matriz?

- ▶ Una matriz, en nuestro contexto, es simplemente un vector de dos dimensiones que tiene el mismo largo en cada uno de sus elementos.

## ¿Qué es una matriz?

- ▶ Una matriz, en nuestro contexto, es simplemente un vector de dos dimensiones que tiene el mismo largo en cada uno de sus elementos.
- ▶ Para declarar una matriz (de enteros) pueden hacer:

```
vector<vector<int> > m;
```

En vez de *int* pueden poner cualquier otro tipo (*string*, *char*, etc).

**Importante!** : Los '>' que cierran tienen que tener un espacio en el medio, porque sino el compilador piensa que queremos usar el operador >>.

## ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

## ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

- ▶ Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).

## ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

- ▶ Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).
- ▶ Tableros (de ajedrez, por ejemplo).

## ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

- ▶ Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).
- ▶ Tableros (de ajedrez, por ejemplo).
- ▶ Mapas (por ejemplo, en cada casillero guardamos la altura del territorio en esas coordenadas, o la cantidad de personas que viven en una determinada manzana).



# ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

- ▶ Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).
- ▶ Tableros (de ajedrez, por ejemplo).
- ▶ Mapas (por ejemplo, en cada casillero guardamos la altura del territorio en esas coordenadas, o la cantidad de personas que viven en una determinada manzana).
- ▶ Imágenes.

## ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

- ▶ Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).
- ▶ Tableros (de ajedrez, por ejemplo).
- ▶ Mapas (por ejemplo, en cada casillero guardamos la altura del territorio en esas coordenadas, o la cantidad de personas que viven en una determinada manzana).
- ▶ Imágenes.
- ▶ Series temporales.

## ¿Y qué hacemos con la matriz?

Muchas veces queremos utilizar matrices para representar estructuras como por ejemplo:

- ▶ Matrices de verdad (esas de Álgebra que tienen determinante y esas cosas).
- ▶ Tableros (de ajedrez, por ejemplo).
- ▶ Mapas (por ejemplo, en cada casillero guardamos la altura del territorio en esas coordenadas, o la cantidad de personas que viven en una determinada manzana).
- ▶ Imágenes.
- ▶ Series temporales.
- ▶ Muchísimos etcéteras.

## Operaciones (que vamos a necesitar) sobre matrices

- Declarar una matriz.

```
vector<vector<int> > m;
```

- Inicializar una matriz de  $m$  filas  $\times$   $n$  columnas con ceros.

```
vector<vector<int> > res(m,vector<int>(n));
```

- Inicializar una matriz de  $m$  filas  $\times$   $n$  columnas todas con el mismo valor ( $x$ ).

```
vector<vector<int> > res(m,vector<int>(n,x));
```

- Agregar una fila.

```
vector<vector<int> > m;  
vector<int> v = {1,2,3}  
m.push_back(v);
```

- Acceder a un elemento en la posición  $(i,j)$ .

```
m[i][j]
```

# Rotación de Matrices

Dada una matrix *mat* de  $n \times m$  y dos enteros *d* y *a* queremos devolver una matrix con las columnas *m* movidas *d* veces a la derecha y las filas movidas *a* veces hacia abajo.

# Rotación de Matrices

Resolvamos el siguiente problema:

```
proc rotar (in mat: seq<seq<ℤ>>, in d: ℤ, in a: ℤ, out res:
seq<seq<ℤ>>) {
  Pre { |mat| > 0 ∧ (∀ i : ℤ) (0 ≤ i < |mat| →L |mat[i]| = |mat[0]|) }
  Post { mismaForma(res, mat) ∧L (∀ i, j : ℤ) (0 ≤ i <
    |mat| ∧L 0 ≤ j < |mat[i]|) →L res[i][j] = mat[(i - a)
    mód |mat|][(j - d) mód |mat[i]|]) }
}

pred mismaForma (in m1, m2: seq<seq<ℤ>>) {
  |m1| = |m2| ∧L (∀ i : ℤ) 0 ≤ i < |m1| →L |m1[i]| = |m2[i]|
}
```

## Rotación de Matrices

```
vector<vector<int> > rotar(vector<vector<int> > mat,
                           int a, int d) {
    int n = mat.size();
    int m = mat[0].size();
    vector<vector<int> > res(n,vector<int>(m));
    int i = 0;
    while(i < n) {
        int j = 0;
        while(j < m) {
            res[i][j] = mat[(i-a)%n][(j-d)%m];
            j++;
        }
        i++;
    }
    return res;
}
```

# Matrices y más matrices

Durante la carrera verán más ejercicios de matrices hasta el cansancio en:

- ▶ Organización del Computador 2: Verán como aplicar filtro a imágenes (como los de Instagram) pero en lenguaje ASM.
- ▶ Algoritmos y Estructuras de Datos 3: Ejercicios sobre grafos, programación dinámica, etc.
- ▶ Métodos Numéricos: mejor conocida como “Matrices: la materia” (verán algoritmos sobre matrices como las de Álgebra).



## Taller de Matrices

El taller de hoy tiene un enunciado y un archivo comprimido. Dentro del archivo que se que se descarguen desde la página de la materia van a encontrar los siguientes archivos y carpetas:

- ▶ Directorio `lib`: Con el GTest comprimido que es preciso descomprimir.
- ▶ Directorio `tests`: Con 8 tests, uno por cada ejercicio del taller.
- ▶ `ejercicios.cpp`: Aquí es donde van a volcar sus implementaciones.
- ▶ `ejercicios.h`: *headers* de las funciones que tienen que implementar.
- ▶ `main.cpp`: Punto de entrada del programa.

Para trabajar, se debe crear el archivo `CMakeList.txt` que involucre todos los directorios y archivos del taller. Se puede aprovechar aquel `CMakeList.txt` del laboratorio 6. Desde el CLION se puede abrir el proyecto con "Open Project".