

Examen 02

En cada ejercicio se evaluará la eficiencia del código, el uso de identificadores significativos, la indentación, escritura correcta de llaves {} y el uso adecuado de la palabra reservada `const`. Se dispone de dos horas para entregar la prueba y debe realizarse en forma estrictamente individual.

Escriba en C++ una clase `Matrix` (o preferiblemente en inglés `Matrix`) para representar matrices de dos dimensiones de valores reales. Debe implementar al menos los siguientes miembros.

- [5%] **Miembros de datos.** Debe almacenar los valores reales de la matriz en memoria dinámica. Tenga en cuenta que la expresión `new double[n][m]` no es válida en C++.
- [5%] **Constructor por defecto.** Crea una matriz no válida de tamaño 0x0, también llamada matriz nula. Útil para indicar resultados de operaciones no válidas. Debe imprimirse como "(null)", sin las comillas.
- [5%] `Matrix(n,m)`. Crea una matriz de n filas por m columnas de valores reales inicializados en 0. Si alguno de los valores n ó m son cero, crea una matriz nula.
- [15%] **Sobrecargas fundamentales para manejo de memoria dinámica.** Dado que la clase `Matrix` utiliza memoria dinámica, debe evitar fugas de memoria o accesos inválidos a toda costa.
- [5%] **Operador de conversión a booleano.** Si una matriz se usa en un contexto booleano, debe evaluarse como `false` si es la matriz nula, `true` en cualquier otro caso. **Operador !** se evalúa como `true` si la matriz es nula, `false` en cualquier otro caso.
- [5%] **Métodos `rows()` y `cols()`.** Retornan la cantidad de filas y columnas en la matriz.
- [10%] **Operador `(i,j)`.** Sobrecarga del operador paréntesis para acceder al valor (i,j) de la matriz. Debe tener sus dos variantes: acceso en modo sólo lectura y en modo escritura. Asume que los índices i y j son válidos. Es decir, si se utilizan valores fuera de rango hará que el programa se caiga. El mismo comportamiento si se invoca este operador en una matriz nula.
- [10%] **Operador `+` y Operador `-`.** Permite sumar o restar dos objetos matriz *del mismo tamaño*. Si son de distinto tamaño se producirá la matriz nula. La suma de dos matrices $A_{n \times m} + B_{n \times m}$ es una matriz $C_{n \times m}$ resultado de sumar cada entrada respectiva de ambas matrices. De forma análoga la resta de dos matrices es la resta de sus respectivas entradas. Es decir:

$$A_{n \times m} + B_{n \times m} = C_{n \times m} \implies c_{ij} = a_{ij} + b_{ij}$$

$$A_{n \times m} - B_{n \times m} = C_{n \times m} \implies c_{ij} = a_{ij} - b_{ij}$$

- .
- [10%] **Producto por un escalar.** Con el operador `*`. El producto de una matriz $A_{n \times m}$ con un real r es una matriz $C_{n \times m}$ resultado de multiplicar r por cada entrada de A . Debe implementar esta operación en forma conmutativa. Matemáticamente:

$$r * A_{n \times m} = C_{n \times m} \implies c_{ij} = r * a_{ij}$$

- .
- [15%] **Producto de dos matrices.** También con el operador `*`. El producto de una matriz $A_{n \times m}$ con una matriz $B_{m \times p}$ es una matriz $C_{n \times p}$, denotado por:

$$A_{n \times m} \times B_{m \times p} = C_{n \times p} \implies c_{ij} = \sum_{k=1}^m a_{ik} b_{kj}$$

Note las restricciones de tamaño que deben cumplir las matrices para poderse multiplicar, de lo contrario, se debe retornar la matriz nula.

- [5%] **Operador `+=` y operador `-=`.** Permiten sumar y restar una matriz con otra y almacenar el resultado en la matriz que aparece en el lado izquierdo del operador. El **operador `*=`** permite multiplicar la matriz por un escalar o por otra matriz y almacenar el resultado en la matriz que aparece en el lado izquierdo del operador. Puede implementar estos métodos en forma ineficiente.

12. [10%] **operator >>**: permite leer una matriz de un archivo. No debe imprimir nada en la salida estándar, sólo leer las entradas una tras otra del archivo. **Operador <<**: imprime la matriz a un archivo separando las entradas un tabulador y las filas por cambios de línea. Para efectos de este examen, el operador << **no** debe tener acceso directo a los miembros de la clase.

Su clase Matrix debe hacer funcionar el siguiente main():

```
#include "Matrix.h"

int main()
{
    size_t n = 0, m = 0;

    cout << "Matrix 1 (nxm): ";
    cin >> n; cin.ignore(); cin >> m;
    Matrix m1(n, m);
    cin >> m1;

    cout << "Matrix 2 (nxm): ";
    cin >> n; cin.ignore(); cin >> m;
    Matrix m2(n, m);
    cin >> m2;

    cout << "m1 + m2 =" << endl << m1 + m2 << endl;
    cout << "m1 - m2 =" << endl << m1 - m2 << endl;
    cout << ".5 * m2 =" << endl << .5 * m2 << endl;
    cout << "m1 * m2 =" << endl << m1 * m2 << endl;

    return 0;
}
```

Ejemplos de ejecución:

```
Matrix 1 (nxm): 2x3
1 -2 3
-3 0 4

Matrix 2 (nxm): 2x3
3 0 -4
-1 1 2

m1 + m2 =
4 -2 -1
-4 1 6

m1 - m2 =
-2 -2 7
-2 -1 2

.5 * m2 =
1.5 0 -2
-0.5 0.5 1

m1 * m2 =
(null)
```

```
Matrix 1 (nxm): 2x3
1 0 2
-1 -2 1

Matrix 2 (nxm): 3x4
2 1 3 1
-2 -1 0 -1
1 0 -1 -3

m1 + m2 =
(null)

m1 - m2 =
(null)

.5 * m2 =
1 0.5 1.5 0.5
-1 -0.5 0 -0.5
0.5 0 -0.5 -1.5

m1 * m2 =
4 1 1 -5
3 1 -4 -2
```