1151044 - Programación Orientada a Objetos
Tarea 3 - 2026-I

Implementa un tipo de dato `matriz` que modele una matriz de enteros, con filas y columnas numeradas a partir de 0. La memoria interna de la matriz se debe gestionar con `new` y `delete`. Tu programa puede elevar una excepción de tipo `std::bad_alloc` si `new` no tiene suficiente memoria.

```
class matriz {
   //...
public:
   //...
};
```

- Constructor `matriz( );`.
  Crea una matriz vacía (sin filas ni columnas). Es un detalle de implementación si tu programa usa o no `new` en este constructor.

```
matriz a;
```

- Constructor `explicit matriz(int f, int c);`.
  Crea una matriz de f filas y c columnas con todos sus elementos en cero. Debe lanzar una excepción compatible con `std::exception` si el valor de f o el valor de c son menores que cero.

```
matriz a(5, 3);
```

- Constructor `matriz(const matriz&);`.
  Crea una matriz que es una copia independiente de otra.

```
matriz a(5, 3);
matriz b = a;
```

- Destructor `~matriz( );`.
  Libera la memoria manejada por la matriz.

```
int main( ) {
   matriz a(5, 3);
}  // el destructor libera la memoria usada por a
```

- Operador de asignación `void operator=(const matriz&);`.
  Libera el contenido previo de la matriz receptora y luego hace una copia independiente de la otra matriz.

```
matriz a(5, 3);
matriz b(4, 6);
b = a;
```

- Función miembro `int filas( ) const;`
  Devuelve la cantidad de filas de la matriz.

```
int f = a.filas( );
```

- Función miembro `int columnas( ) const;`
  Devuelve la cantidad de columnas de la matriz.

```
int c = a.columnas( );
```

- Función miembro `int& operator()(int i, int j);`
  Devuelve el elemento de la fila i y columna j.[1] Debe lanzar una excepción compatible con `std::exception` si los índices dados no son válidos en la matriz.

```
int v = a(1, 2);
```

[1] En C++ moderno existe el `operator[]` de más de un parámetro dentro de los corchetes (que habilitaría la notación `mat[1, 2]`). Sin embargo, querer implementar el `operator[]` de un solo índice (como para habilitar la notación `mat[1][2]`) es fastidioso en general; el caso de matrices de dos dimensiones no representa un gran problema, pero el caso de más dimensiones sí.

- Función miembro `const int& operator()(int i, int j) const;`
  Igual a la función anterior, pero para matrices `const`.

```
const matriz& r = a;
int v = r(1, 2);
```

El tipo de dato no debe exponer públicamente ninguna variable miembro interna y debe implementarse de modo de que sea imposible que un objeto de este tipo adquiera un estado inválido. El tipo de dato no debe permitir fugas de memoria. Sólo puedes usar los archivos de biblioteca `<stdexcept>` y `<exception>`. Tu código no debe declarar variables estáticas ni globales. Un programa que manipule a lo mucho cien matrices y que haga a lo mucho mil llamadas a función debe terminar en menos de un segundo. Puedes consultar una página de prueba en https://racc.mx/uam/trimestre-actual/2026-i/poo/tarea3.html. El código que envíes no debe contener `main` y no debe leer ni imprimir nada directamente.

Envía tu código fuente desde tu cuenta institucional al formulario en https://forms.gle/hKTWWHzwj873dGN38. Tu código será evaluado con varios casos de prueba y se espera que cumpla la semántica descrita.

Ejemplo de uso	**Ejemplo de salida**

`int main( ) {`
`  matriz a(5, 3);`
`  matriz b = a;`
`  a(0, 1) = 7;`
`  std::cout << a(0, 1) << "\n";`
`  std::cout << b(0, 1) << "\n";`
`}`
	`7`
	`0`