Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI-1201 Programación II - 2012b
Profesor Jeisson Hidalgo-Céspedes
08-Noviembre-2012. Duración 2 horas

Examen 02 (reposición)

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 `Matriz` (o preferiblemente en inglés `Matrix`) para representar matrices *dinámicas* de dos dimensiones de valores reales. Se pretende que su clase sea dinámica con el fin de poder leer matrices a partir de archivos sin conocer inicialmente su tamaño. Debe implementar al menos los siguientes miembros.

1. [10%] **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++. Debe además diferenciar entre la capacidad de la matriz de la cantidad de valores realmente almacenados en ella.
2. [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.
3. [10%] **`Matrix(n,m)`**. Crea una matriz de `n` filas por `m` columnas de valores reales inicializados en 0. Se dice que esta matriz tendrá tanto tamaño como capacidad `nxm`. Si alguno de los valores `n` ó `m` son cero, crea una matriz nula.
4. [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. [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.
6. [5%] **Métodos `rows()` y `cols()`**. Retornan la cantidad de filas y columnas actualmente almacenados en la matriz.
7. [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
8. [10%] **Operador +**. Permite sumar dos objetos matriz *del mismo tamaño*. Si son de distinto tamaño se producirá la matriz nula. La suma de dos matrices $A_{nxm} + B_{nxm}$ es una matriz C_{nxm} 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:

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

9. [20%] **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. Note que el archivo no indica el tamaño de la matriz, sino que su programa lee valores del archivo e incrementa el tamaño de las columnas dinámicamente, hasta encontrar un cambio de línea. A partir de este momento su matriz sigue creciendo el tamaño de filas dinámicamente. Asuma que la matriz en el archivo es válida.
10. [10%] **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.