Práctica 7

Datos generales:

Nombre de la Práctica	Arreglos multidimensionales
Nombre de la carrera	Ingeniería de Software
Nombre de la materia	Laboratorio de Estructuras de Datos
Número y nombre de Unidad(es) temática(s)	1. Introducción a las estructuras de datos y estructuras fundamentales.
Docente que imparte la materia	Aldonso Becerra Sánchez
Fecha de entrega para los alumnos	28-marzo-2025
Fecha de entrega con extensión y penalización	29-marzo-2025
Fecha de elaboración	27-marzo-2025

Objetivo de la Práctica	Comprender el uso de los arreglos multidimensionales numéricos para la definición de una base API funcional para su futura aplicación en entornos reales.
Tiempo aproximado de realización	4 horas
Introducción	La facilidad que los arreglos multidimensionales tienen para permitir guardar datos en forma de columnas/renglones, los hace pertinentes para la resolución de muchos problemas dónde se requiere esta situación. El único detalle con esta cuestión es que es poco flexible el número de elementos que podemos manipular, ya que se requiere conocer a priori la cantidad de elementos a guardar.

Referencias que debe consultar el alumno (si se requieren):

Referencia 1:

1.Cairo, Osvaldo; Guardati, Silvia. Estructura de Datos, Tercera Edición. McGraw-Hill, México, Tercera Edición, 2006.

Referencia 2:

2.Mark Allen Weiss. Estructura de datos en Java. Ed. Addison Wesley.

Referencia 3:

3. Joyanes Aguilar, Luis. Fundamentos de Programación. Algoritmos y Estructuras de Datos. Tercera Edición, 2003. McGraw – Hill.

Actividades que debe realizar el alumno:

Actividad inicial:

Lea primero toda la práctica. No inicie a programar sin leer todo cuidadosamente primero. Recuerde que debe generar el reporte en formato IDC.

Actividad 1:

Primero genere la **Introducción**.

Actividad 2:

Defina un TDA llamado "Arreglo2DNumerico" que herede de "Arreglo2D", de modo que este tipo de matriz pueda realizar lo siguiente:

1. Valide que todos los valores pasados a cada método de esta clase sean estrictamente heredados de "Number", tal como se hizo en ArregloNumerico. Para esto deberá sobre-escribir e invocar los métodos de la super clase de tal manera que se cumpla este cometido también en esos métodos. Guarde los valores como double de manera predeterminada.
2. Constructor: Arreglo2DNumerico (int filas, int columnas). Que llame a inicializar con valores en cero de manera predeterminada.
3. Constructor: Arreglo2DNumerico (int filas, int columnas, Object valor). Que llame a inicializar con valores indicados por contenido.
4. Multiplicar un escalar por la matriz: boolean porEscalar(Number escalar). Debe validar las dimensiones y el procedimiento de acuerdo a la teoría de matrices.
5. Multiplicar un escalar para cada elemento de la matriz: boolean porEscalares(ArregloNumerico escalares). Multiplica el elemento de una

posición del arreglo de números (cada posición contiene un escalar) por un solo elemento colocado en la posición correspondiente en la matriz, recorriéndola en orden natural (renglones, columnas). Quiere decir que el escalar de la posición 0 del arreglo se multiplica por el elemento 0,0 de la matriz, el escalar de la posición 1 del arreglo se multiplica por el elemento 0,1 de la matriz, etc.

6. Sumar un escalar a la matriz: boolean sumarEscalar(Number escalar). Debe validar las dimensiones y el procedimiento de acuerdo a la teoría de matrices.
7. Sumar un escalar para cada elemento de la matriz: boolean sumarEscalares(ArregloNumerico escalares). Suma un elemento contenido en una posición del arreglo de números (cada posición contiene un escalar) más un solo elemento colocado en la posición correspondiente en la matriz, recorriéndola en orden natural (renglones, columnas). Quiere decir que el escalar de la posición 0 del arreglo se suma al elemento 0,0 de la matriz, el escalar de la posición 1 del arreglo se suma al elemento 0,1 de la matriz, etc.
8. Multiplicar por otra matriz: boolean multiplicar(Arreglo2DNumerico matriz2). Debe validar las dimensiones y el procedimiento de acuerdo a la teoría de matrices.
9. Sumar con otra matriz: boolean sumar(Arreglo2DNumerico matriz2). Debe validar las dimensiones y el procedimiento de acuerdo a la teoría de matrices.
10. Elevar a una potencia la matriz (elemento por elemento): boolean aplicarPotencia(Number escalar). Aplicar la potencia elemento por elemento.
11. Aplicar el logaritmo (elemento por elemento) a la matriz: boolean aplicarLogaritmo(TipoLogaritmo tipoLogaritmo). Valide que si existe un valor en cero en la matriz, no se puede aplicar el logaritmo. El proceso es elemento por elemento. Defina un enumerado llamado TipoLogaritmo, que indica que es NATURAL(base e), BASE10 o BASE2. Este tipo será pasado como argumento al método.
12. Generar una matriz diagonal con el valor proporcionado: boolean matrizDiagonal(Number contenido).
13. Determinar si la matriz es una matriz triangular superior: boolean esDiagonalSuperior().
14. Determinar si la matriz es una matriz triangular inferior: boolean esDiagonalInferior().
15. Determinar la potencia de una matriz. boolean potencia(int exponente). Aplicar esta operación dado por $A^2=AA$, $A^3=AAA$, en general:

$$A^k = \underbrace{A\,A\,\ldots\,A}_{k\ veces}\quad,\quad k \in \mathbb{N}$$

16. Redimensionar (sumando) una matriz por columnas, a la mitad (si la matriz no tiene un número par de columnas, la del centro debe pasar intacta, solo debe acumular las demás), de tal forma que boolean doblarRenglones():

4	6	4	3
1	2	2	4
5	3	2	1
0	8	7	3

Quedaría así:

10	7
3	6
8	3
8	10

17. Redimensionar (sumando) una matriz por renglones, a la mitad (si la matriz no tiene un número par de renglones, el del centro debe pasar intacto, solo debe acumular los demás), de tal forma que boolean doblarColumnas():

4	6	4	3
1	2	2	4
5	3	2	1
0	8	7	3

Quedaría así:

5	8	6	7
5	11	9	4

Haga el programa (actividad 2, la cual es el **Desarrollo** del programa, junto con la captura de pantalla del programa funcionando).

Actividad 3:

Pruebe el funcionamiento del programa de la actividad 2 con todo y sus capturas de pantalla.

Actividad 4:

Realice la sección de **Código agregado** (diagrama de clases UML).

Actividad 5:

Realice la sección de **Pre-evaluación** (use los lineamientos establecidos).

Actividad 6:

Finalmente haga las **Conclusiones**.

Actividad 5:

Enviar en http://ingsoftware.reduaz.mx/moodle

Archivo anexo que se requiere para esta tarea (opcional):

Dudas o comentarios: a7donso@gmail.com