UD04.Estructuras de almacenamiento. Ejercicios.

DAM1-Programación 2024-25

Otras fuentes de ejercicios:

- Aceptaelreto.com - Categoría Arrays
- Aceptaelreto.com - Categoría Arrays multidimensionales
- Aceptaelreto.com - Categoría Cadenas

Sobre los ejercicios.

1. Organiza los ficheros de código fuente en los paquetes que se indican en cada apartado.
2. Nombra los ficheros y las clases con el nombre/código del ejercicio en negrita.
3. Utiliza un *comentario de documentación* para indicar tu nombre y apellidos como autor del código.
4. Opcional. Entrega el paquete comprimido en el Aula Virtual al terminar la UD.

Ejercicios Arrays y Matrices

Paquete: `arraysejercicios`

EjArrays01. Calcular la media de una serie de números que se leen por teclado.

EjArrays02. Leer 10 números enteros por teclado y guardarlos en un array. Calcula y muestra la media de los números que estén en las posiciones pares del array.

EjArrays03. Leer por teclado la nota de los alumnos de una clase y calcular la nota media del grupo. Mostrar los alumnos con notas superiores a la media.

EjArrays04. Guardar en un array los 20 primeros números pares.

EjArrays05. Contar el número de elementos positivos, negativos y ceros en un array de 10 enteros.

EjArrays06. Leer 10 enteros y mostrar la media de los valores negativos y la de los positivos.

EjArrays07. Leer N alturas y calcular la altura media. Calcular cuántas hay superiores a la media y cuántas inferiores.

EjArrays08. Leer el nombre y sueldo de 20 empleados y mostrar el nombre y sueldo del empleado que más gana.

EjArrays09. Llenar un array con números aleatorios.

EjArrays10. Escribe un programa en Java que lea una matriz de enteros de 3 filas y 3 columnas del usuario y luego calcule la suma de todos los elementos de la matriz.

EjArrays11. Escribe un programa en Java que implemente un **juego de tres en raya.** El programa debe permitir a dos jugadores jugar al juego turnándose para colocar fichas en un tablero de 3x3. El juego termina cuando un jugador consigue tres fichas en línea (horizontal, vertical o diagonal)

Sugerencias:

a. Tablero como una matriz 3x3 char, 3 símbolos distintos.
b. 2 jugadores
c. iniciarTablero()
d. mostrarTablero()
e. leerMovimiento()
f. boolean comprobarVictoria()

EjArrays12_MatrizIdentidad. Crea una función que determine si una matriz dada es una matriz identidad.

EjArrays12b_MatrizIdentidad Escribe un programa en Java que reciba un número entero n y genere una matriz identidad de tamaño n x n.

EjArrays13. Una matriz diagonal es una matriz cuadrada cuyos elementos fuera de la diagonal principal son todos cero.
Ejemplo de una matriz de tamaño 3×3:

$$\begin{bmatrix} 6 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Crea una función que determine si una matriz dada es una matriz diagonal. La función devolverá true si la matriz de números enteros que se le pasa como parámetro es una matriz diagonal, y false en cualquier otro caso. Utiliza el siguiente prototipo para implementar la función:

```
boolean esDiagonal(int[][] t);
```

Pruebas que deberá pasar la función:

```
@Test
public void pruebaMatrizDiagonal() {
    int[][] matriz = {{1, 0, 0}, {0, 2, 0}, {0, 0, 3}};
    assertTrue(esDiagonal(matriz));
}

@Test
public void pruebaMatrizNoDiagonal() {
    int[][] matriz = {{1, 0, 0}, {0, 2, 0}, {0, 1, 3}};
    assertFalse(esDiagonal(matriz));
}

@Test
public void pruebaMatrizNoCuadrada() {
    int[][] matriz = {{1, 0, 0}, {0, 2, 0}};
    assertFalse(esDiagonal(matriz));
}

@Test
public void pruebaMatrizNoCuadrada2() {
    int[][] matriz = {{1, 0}, {0, 2}, {0, 0}};
    assertFalse(esDiagonal(matriz));
}

@Test
public void pruebaMatrizVacia() {
    int[][] matriz = {};
    assertFalse(esDiagonal(matriz));
}

@Test
public void pruebaMatrizNull() {
    int[][] matriz = null;
    assertFalse(esDiagonal(matriz));
}
```

EjArrays14. Crea una función que calcule la traspuesta de una matriz.

EjArrays15. Crea una función que calcule la suma de dos matrices.

EjArrays16. Crea una función que calcule el producto de dos matrices.

```
public static int[][] multiplicar(int t1[][], int t2[][])
```

EjArrays17. Crea una función que determine si una matriz es simétrica.

EjArrays18. Crea una función que determine si una matriz es ortogonal.

CuadradosDiabolicos. Cuadrados diabólicos y esotéricos - ¡Acepta el reto!

- Implementa y prueba la siguiente función:

```
esCuadradoEsoterico(int[][] t);
```

BuscarFilaMayorSuma. Escribe un método en Java que reciba una matriz de enteros y determine y devuelva en un array la fila que tiene la mayor suma de sus elementos.

Casos especiales:

- matriz de entrada == null
  - devuelve null
- matriz de entrada vacía
  - devuelve array vacío []
- matriz de una única fila
  - devuelve esa fila
- empate de dos o más filas en la suma de sus elementos
  - devuelve la primera de las filas empatadas

Haz un programa de ejemplo que muestre una matriz de entrada y la fila resultante.

Recursos:

- Arrays como parámetros. Referencias
- Recorrer una matriz fila a fila
- Acumuladores
- Calcular máximo

Ejercicios Paraninfo

EP0511. Realiza la función: `int[] buscarTodos(int t[], int clave)`, que crea y devuelve una tabla con todos los índices de los elementos donde se encuentra la clave de búsqueda. En el caso de que clave no se encuentre en la tabla `t`, la función devolverá una tabla vacía.

EP0512_Desordenar. Escribe la función `static void desordenar(int t[])`, que cambia de forma aleatoria los elementos contenidos en la tabla `t`. Si la tabla estuviera ordenada, dejaría de estarlo.

Recursos:
- Recorrer todo un array para modificar
- Generar números aleatorios
- Intercambiar valores de dos variables

Para probar en un programa principal:
- Ordenar un array
- Mostrar un array

EP0513. Modifica la Actividad de aplicación **EP0512** para que la función no modifique la tabla que se pasa como parámetro y, en su lugar, cree y devuelva una copia de la tabla donde se han desordenado los valores de los elementos.

EP0514_Sueldos. El ayuntamiento de tu localidad te ha encargado una aplicación que ayude a realizar encuestas estadísticas para conocer el nivel adquisitivo de los habitantes del municipio. Para ello, tendrás que preguntar el sueldo a cada persona encuestada. A priori, no conoces el número de encuestados. Para finalizar la entrada de datos, introduce un sueldo con valor -1.

Una vez terminada la entrada de datos, muestra la siguiente información:
- Todos los sueldos introducidos ordenados de forma decreciente.
- El sueldo máximo y mínimo.
- La media de los sueldos.
- Cuántos sueldos hay por encima de la media.
- Cuántos sueldos hay por debajo de la media.

Recursos:
- Bucle controlado por condición (lectura adelantada?)
- Añadir elemento al final de un array
- Ordenar un array
- Mostrar un array
- Recorrer todo un array para lectura (for-each)
- Sumadores y Contadores

EP0515. Debes desarrollar una aplicación que ayude a gestionar las notas de un centro educativo. Los alumnos se organizan en grupos compuestos por 5 personas. Leer las notas (números enteros) del primer, segundo y tercer trimestre de un grupo. Debes mostrar al final la nota media del grupo en cada trimestre y la media del alumno que se encuentra en una posición dada (que el usuario introduce por teclado).

EP0516. El mapa de un juego de rol puede implementarse como una matriz donde las filas y las columnas representan lugares (lugar 0, lugar 1, lugar 2, etc.) que estarán conectados. Si desde el lugar X podemos ir hacia el lugar Y, entonces la matriz en la posición [x][y] valdrá cierto; en caso contrario, valdrá falso. Escribe una función que, dada una matriz que representa el mapa y dos

lugares, indique si es posible viajar desde el primer lugar al segundo (directamente o pasando por lugares intermedios).

La función anterior se podría implementar según los algoritmos de
- Búsqueda en anchura, usando una estructura de tipo pila.
- Búsqueda en profundidad, usando una estructura de tipo cola.

El segundo algoritmo se puede implementar también usando recursividad. En cualquiera de las implementaciones será necesario recordar los lugares ya visitados para evitar posibles bucles infinitos..

EP0517. Implementa la función: `int[] suma(int t[], int numElementos)`, que crea y devuelve una tabla con las sumas de los `numElementos` elementos consecutivos de `t`. Veamos un ejemplo, sea t= [10, 1, 5, 8, 9, 2]. Si los elementos de `t` se agrupan de 3 en 3, se harán las sumas:

10 + 1 + 5. Igual a 16.
1 + 5 + 8. Igual a 14.
5 + 8 + 9. Igual a 22.
8 + 9 + 2. Igual a 19.

Por lo tanto, la función devolverá una tabla con los resultados: [16, 14, 22, 19].

EP0518. Escribe un programa que solicite los elementos de una matriz de tamaño 4 x 4. La aplicación debe decidir si la matriz introducida corresponde a una matriz mágica, que es aquella donde la suma de los elementos de cualquier fila o de cualquier columna valen lo mismo.

EP0519. Diseña una aplicación para gestionar la llegada a meta de los participantes de una carrera. Cada uno de ellos dispone de un dorsal (un número entero) que los identifica. En la aplicación se introduce el número de dorsal de cada corredor cuando éste llega a la meta. Para indicar que la carrera ha finalizado (han llegado todos los corredores a la meta), se introduce como dorsal el número -1.

A continuación, la aplicación solicita información extra de los corredores. En primer lugar, se introducen los dorsales de todos los corredores menores de edad; para premiarlos por su esfuerzo se les avanza un puesto en el ranking general de la carrera, es decir, es como si hubieran adelantado al corredor que llevaban delante. En segundo lugar, se introducen los dorsales de los corredores que han dado positivo en el test antidopaje, lo que provoca su expulsión inmediata. Para finalizar, se introducen los dorsales de los corredores que no han pagado su inscripción en la carrera, lo que provoca que se releguen a los últimos puestos del ranking general. La aplicación debe mostrar los dorsales de los corredores que han conseguido las medallas de oro, plata y bronce.

EP0520. La **fusión de dos tablas ordenadas** consiste en copiar todos sus elementos (de ambas tablas) en una tercera que deberá seguir ordenada. Podemos realizar una fusión ineficiente copiando los elementos de ambas tablas (sin tener en cuenta el orden) en la tabla final y ordenar esta. Existe una manera óptima de realizar la fusión en la que se elige en cada momento el primer elemento no copiado de alguna de las tablas y se añade a la tabla final, que seguirá ordenada sin necesidad de ordenación alguna. Busca información sobre el algoritmo de fusión e impleméntalo en Java.

```
public static int[] fusionOrdenada(int[] t1, int[] t2);
```