

Hoja de ejercicios del Tema 6

1. Utilizando las estructuras de datos definidas en el ejercicio 7 de la hoja de ejercicios del Tema 5 (empleados de una empresa), implementa los siguientes subprogramas:
 - a) Añadir un nuevo empleado a la plantilla en la posición i de la lista (con $0 \leq i < N$).
 - b) Eliminar el empleado en la posición i de la lista (con $0 \leq i < N$).
2. Un archivo de texto contiene información acerca de los productos que se venden en un almacén. Lo único que se sabe acerca del número de productos es que no puede superar un cierto valor `MaxProductos`. De cada producto se guarda información sobre su código identificador (entero positivo), su precio (real) y el número de unidades existentes (entero positivo). El formato en el que se guarda la información dentro del archivo es el siguiente:

```
id1 precio1 unidades1
id2 precio2 unidades2
...
idN precioN unidadesN
-1
```

- a) Declara un tipo `tProducto` que represente la información de un producto y un tipo `tLista` que mantenga la información de todos los productos.
- b) Escribe un subprograma que lea los datos del archivo de texto que almacena la información, los guarde en la lista y luego los muestre en la pantalla.
- c) Escribe un subprograma que encuentre el producto con máximo valor en el almacén, considerando que el valor del producto i es $\text{precio}_i * \text{unidades}_i$.
- d) Escribe un subprograma que dado un identificador de producto a eliminar del almacén, lo busque en la lista y lo elimine actualizando la lista como corresponda.

3. Define un tipo `tVector` que permita representar secuencias de N enteros e implementa, además de subprogramas para leer y mostrar un vector:
- a) Un subprograma que, dado un vector, mueva sus componentes un lugar a la derecha. El último componente se moverá al 1^{er} lugar.
 - b) Un subprograma que, dado un vector, calcule y devuelva la suma de los elementos que se encuentran en las posiciones pares del vector.
 - c) Un subprograma que, dado un vector, encuentre y devuelva la componente de mayor valor.
 - d) Un subprograma que, dados dos vectores, devuelva un valor que indique si son iguales.
 - e) Un subprograma que obtenga si alguno de los valores almacenados en el vector es igual a la suma del resto de los valores del mismo.
 - f) Un subprograma que obtenga si alguno de los valores almacenados en el vector está repetido.

4. Dado un vector de N caracteres $v1$, en el que no hay elementos repetidos, y otro vector de M caracteres $v2$, donde $N \leq M$, se quiere comprobar si todos los elementos del vector $v1$ están también en el vector $v2$. Por ejemplo, si:

$v1 =$ 'a' 'h' 'i' 'm'
 $v2 =$ 'h' 'a' 'x' 'x' 'm' 'i'

El resultado sería cierto, ya que todos los elementos de $v1$ están en $v2$.

5. Implementa un programa que permita realizar operaciones sobre matrices de $N \times N$. El programa debe permitir al usuario la selección de alguna de las siguientes operaciones:
- a) Sumar 2 matrices.
 - b) Restar 2 matrices.
 - c) Multiplicar 2 matrices.
 - d) Trasponer una matriz.
 - e) Mostrar una matriz señalando cuáles son los puntos de silla (los *puntos de silla* de una matriz son aquellos elementos de la misma que cumplen ser el mínimo de su fila y el máximo de su columna).

Habrán también dos subprogramas para leer del teclado o mostrar en la pantalla una matriz.

6. Construye las declaraciones de tipos necesarias para almacenar las respuestas (acierto, fallo) a un cuestionario de 20 preguntas. A partir de este tipo de datos define otro que almacene los datos de un examen con las respuestas al cuestionario de 30 estudiantes. Por otro lado, construye una estructura de datos para almacenar el número de veces que cada pregunta ha sido acertada y otra estructura con las calificaciones obtenidas que almacene el número de suspensos, aprobados, notables y sobresalientes (declara un enumerado).

El programa cargará las respuestas de los 30 estudiantes a las 20 preguntas desde un archivo `examen.txt` que contendrá 30 líneas, cada una con 20 enteros separados por un espacio. Cada entero será un 1 si la pregunta se ha acertado y un 0 en caso contrario.

Además se implementarán los siguientes subprogramas:

- a) Construye un subprograma que evalúe un cuestionario y devuelva la nota obtenida (entre 0 y 10).
 - b) Construye un subprograma que, a partir de los datos de un examen, rellene la estructura con el número de veces que se ha acertado cada pregunta.
 - c) Construye un subprograma que, a partir de los datos de un examen, rellene la estructura de las calificaciones.
7. Una pantalla de texto del PC tiene 25 filas, de 80 columnas cada una de ellas. En cada posición de pantalla se guarda el carácter representado en dicha posición, el color con que se escribe el carácter, el color de fondo donde se visualiza el carácter y si el carácter parpadea o no. Diseña las estructuras de datos necesarias considerando que los colores posibles son: negro, azul, verde, amarillo y rojo. Construye un subprograma que, dada una fila, devuelva el color de fondo más utilizado en esa fila.
8. Utilizando las estructuras de datos definidas en el ejercicio anterior, escribe un subprograma que determine en qué fila y columna se encuentra el primer carácter parpadeante.
9. Escribe un programa que lea dos cadenas del teclado y determine si una es un anagrama de la otra, es decir, si una cadena es una permutación de los caracteres de la otra. Por ejemplo, “acre” es un anagrama de “cera” y de “arce”. Ten en cuenta que puede haber letras repetidas en la cadena (“carro”, “llave”).

- 10.** Escribe un programa que lea del teclado una frase y a continuación visualice las palabras de la frase en columna, seguida cada una del número de letras que la componen.
- 11.** Para que los cálculos resultasen más fáciles, los romanos solían utilizar una notación aditiva pura. Un número romano en esta notación es una cadena de dígitos romanos, comenzando por el dígito del valor más alto y terminando con el de valor más pequeño, y su valor se obtenía simplemente como la concatenación de sus dígitos (en esta notación, por ejemplo, el número arábigo 19 se representa como XVIII, X -10- seguido de VIII -9-). (Nota: en lugar de usar las expresiones sustractivas, como IX, se usan siempre las aditivas, VIII.)

Escribe un subprograma `romanoAEntero()` que lea una cadena de caracteres introducida por el teclado y devuelva:

- Un valor que indique si la cadena leída se puede interpretar como un número romano válido en notación aditiva pura, y
- el número arábigo equivalente calculado en caso de ser válido el número romano.

Así, por ejemplo, si se introduce por teclado MDCCCCLXXXVIII el subprograma devolverá un valor que indique que el número romano es válido y el número arábigo 1989. En cambio, si se introduce por teclado aX, el subprograma devolverá un valor que indique que no es un número romano válido.

NOTA: La cadena no se podrá interpretar como número romano si contiene algún carácter que no sea un dígito romano. Recuerda que los dígitos romanos y sus valores correspondientes son: I = 1; V = 5; X = 10; L = 50; C = 100; D = 500; M = 1000.

- 12.** Como parte de un programa de gestión contable, se necesita una estructura de datos capaz de almacenar los gastos en distintos conceptos producidos cada mes por cada uno de los departamentos. Define tal estructura de datos teniendo en cuenta que los departamentos y los tipos de gastos contemplados son los siguientes:
- ✓ Departamentos: Marketing, Contabilidad, Recursos_Humanos, Distribución, Ingeniería, Investigación.
 - ✓ Conceptos: Salarios, Suministros, Mobiliario, Equipamiento, Otros.

Define subprogramas que, partiendo de una matriz de gastos completa, realicen las siguientes tareas:

- a) Escribir en la pantalla la suma de gastos de cada uno de los departamentos.
- b) Escribir en la pantalla la suma de gastos por cada concepto.
- c) Calcular y devolver el total de gastos de un año.
- d) Construir y devolver un array con los gastos generados cada mes.
- e) Encontrar el valor más alto de la tabla de gastos y devolver el departamento responsable, el tipo de gasto y el mes en que se produjo.

13. Queremos programar el juego de las 3 en raya:

X	O	
	X	X
	O	O

- a) Define una estructura de datos para representar el tablero de juego.
- b) Implementa un subprograma para inicializar el tablero con blancos.
- c) Implementa un subprograma para dibujar el tablero junto con el contenido de cada posición ('X', 'O', ' ').
- d) Implementa un subprograma que devuelva `true` o `false`, dependiendo de si todo el tablero está ocupado o existen casillas libres.
- e) Implementa la función Booleana `TresEnRaya(tableroJuego, jugador)` que devuelva `true` en caso de que un jugador tenga la jugada de tres en raya y `false` en caso contrario.

14. En un archivo de texto se encuentran grabados los resultados correspondientes a unas elecciones autonómicas. Cada línea de dicho archivo almacena los datos de una comunidad (máximo 15 comunidades). Éstos son:

- Número de la comunidad autónoma.
- Número de partidos políticos que han obtenido escaños en el parlamento regional correspondiente (máx. 10 partidos).
- Por cada uno de los partidos:
 - ❖ Nombre (una palabra)
 - ❖ Número de escaños obtenidos

Como los nombres de los partidos no pueden tener espacios intermedios nos basta un espacio para separar los distintos campos:

1 4 Azul 3 Rojo 4 Verde 1 Negro 1
 2 3 Rojo 1 Azul 3 Rosa 1
 3 5 Negro 2 Azul 3 Rojo 3 Verde 1 Amarillo 1
 4 3 Verde 1 Azul 2 Rojo 1
 5 4 Rojo 3 Verde 2 Azul 3 Rosa 1
 -1



Escribe un programa que muestre los nombres de los partidos vencedores en cada una de las comunidades autónomas, así como el porcentaje de escaños obtenidos. En caso de empate se escribirán los nombres de los partidos empatados.

15. Escribe un programa que tras leer de archivo una matriz de orden $N \times M$ de valores enteros, indique si el máximo y el mínimo de los valores de la matriz coinciden en la misma fila o en la misma columna. Si hubiera varios máximos o mínimos se tomarían las posiciones de la primera aparición.

El archivo de entrada está formado por N líneas; en cada línea hay M enteros.

16. Se quiere desarrollar una función que dadas dos matrices cuadradas de enteros, M de 10×10 y P de 3×3 , compruebe si entre todas las submatrices de 3×3 que se pueden formar en la matriz M , desplazándose por filas o columnas, existe al menos una que coincida con la matriz P . En ese caso, se indicarán la fila y la columna de la matriz M en la que empieza la submatriz que coincide. Si hay varias, bastará con indicar la primera.

17. El pirata Pata de Palo desea conseguir el Gran Tesoro sin morir ahogado. La *Isla*, que consiste en un tablero cuadrado de $N \times N$, rodeado de agua y con dos puentes en dos esquinas, se puede representar así:

A	G	U	A	
G				A
U				U
A				G
	A	G	U	A

- a) Escribe un procedimiento `inicializarTablero()` que coloque sobre el tablero al pirata **Pata de Palo** y el **Gran Tesoro** en casillas distintas,

determinadas de una forma aleatoria. El **agua** se ha de colocar en los bordes y los **puentes** en las esquinas indicadas.

- b) Escribe un subprograma `direccionPirata()` que nos dé la dirección en la que avanzará el pirata (Norte, Sur, Este u Oeste). Un número aleatorio entre 1 y 4 decidirá qué dirección toma el pirata.
- c) Escribe un procedimiento `dibujarTablero()` que dibuje el tablero con el formato de la figura.
- d) Escribe un programa que permita al pirata **Pata de Palo** intentar buscar el **Gran Tesoro** en menos de 50 movimientos. El pirata recorre los cuadros de uno en uno según la dirección. Al final ha de indicarse si el pirata ha encontrado el tesoro o si se ha ahogado (ha caído al agua).

- 18.** En una panadería se elaboran diariamente los siguientes tipos de productos: colines (30), viena (45), barra (85), baguette (90), hogaza de $\frac{1}{2}$ kg (100), hogaza de 1 kg (150) y hogaza de 3 kg (250). Los fines de semana se elaboran tortas de anises (130) y tortas de chicharrones (180). Los precios de cada producto son los indicados entre paréntesis. El panadero desea mantener actualizada la información de una semana de la cantidad diaria de productos elaborados, de productos vendidos en horario de mañana y de productos vendidos en horario de tarde. Considera que el horario de mañana es de 8 a 13 h. y el de tarde es de 16 a 19 h. Realiza las declaraciones de tipos apropiadas para representar la citada información y escribe un subprograma que muestre por pantalla el dinero obtenido por la venta de todos los productos el último día de la semana.
- 19.** Define una estructura de datos para representar las temperaturas de todos los días del año, por días de semana y por semanas. Considera 52 semanas al año y 7 días por semana. Escribe un subprograma que devuelva la menor temperatura de todo un año, junto con el día de la semana y la semana del año.
- 20.** Construye un programa que lea una cadena de caracteres del teclado y construya una nueva cadena con los plurales de las palabras. Suponemos que las cadenas solo contienen nombres y los nombres están separados por un espacio en blanco. Para formar el plural solamente se seguirán las siguientes reglas:
- Los nombres que terminan en vocal añaden **s**
 - Los nombres que terminan en consonante añaden **es**

- 21.** El juego de las 4 en línea consta de un tablero formado por siete columnas y seis filas. En una partida participan dos jugadores, uno con fichas blancas y otro rojas. Inicialmente todas las posiciones del tablero están libres. Cada jugador coloca alternativamente una ficha en una columna. La ficha colocada cae por su propio peso hasta el fondo de la columna correspondiente (primera fila de la columna libre); por ejemplo, en la figura si el jugador Rojo coloca una ficha en la columna 2, la ficha se coloca en la fila 3. La partida la gana el jugador que coloque en primer lugar cuatro de sus fichas en línea horizontal, vertical o en diagonal. La partida queda en tablas si ninguno de los jugadores es capaz de alinear cuatro fichas después de llenar el tablero.

6	L	L	L	L	L	L	L
5	L	L	L	L	L	L	L
4	L	L	L	■	L	L	L
3	L	L	■	□	■	L	L
2	L	■	□	□	□	■	L
1	L	■	■	□	□	□	■
	1	2	3	4	5	6	7

- Codifica las estructuras de datos necesarias para jugar una partida.
- Escribe un subprograma `inicializarJuego()` que quite todas las fichas del tablero y prepare el tablero de juego.
- Escribe un subprograma `colocarFicha()` que dado un jugador (blanco o rojo) y una columna (1 a 7), coloque la ficha en la posición correspondiente.
- Escribe un subprograma `presentarTablero()` que visualice en la pantalla el estado del tablero (como en la figura, excepto la rejilla).
- Escribe un subprograma `comprobarGanador()` que, dado un jugador (blanco o rojo) y una determinada casilla (1 a 6, 1 a 7), determine si hay cuatro fichas del mismo jugador alineadas en horizontal.

