Práctica 6: Arrays

1. Hacer un algoritmo que lea del teclado 20 números enteros y los almacene en un vector. Utilizando este vector, visualizar y sumar los elementos que ocupan las posiciones pares.

2. Algoritmo para visualizar, contar y sumar los elementos pares que ocupan las posiciones impares de la lista anterior. Visualizar las posiciones que ocupan dichos elementos en la lista.

3. Visualizar los elementos, así como la posición que ocupan, que sean al mismo tiempo múltiplos de 2 y múltiplos de 3. Calcular su suma.

4. Generar y visualizar una lista con la tabla de multiplicar de un número introducido por teclado. Imprimir el elemento que ocupa la posición 3.

5. Crear e imprimir una lista unidimensional de 50 elementos con números aleatorios entre 1 y 100 de tal forma que no se repita ninguno.

6. Crear un procedimiento para insertar números en un vector, por posición. Hacer un algoritmo que utilizando dicho procedimiento nos permita insertar un número en una lista numérica de 10 elementos.

7. Hacer un procedimiento para borrar un número de una lista. Si no lo encuentra deberá dar un mensaje de error. Hacer un algoritmo para borrar un número introducido desde teclado en la lista numérica anteriormente creada.

8. Diseñar un algoritmo que lee una matriz de tamaño 6×8 de números enteros (fila a fila), la almacena en un array bidimensional llamado ***A***. A continuación almacena los resultados de las sumas de los elementos de cada fila en un vector llamado ***B***, y las sumas de los elementos de cada columna en un vector llamado ***C***. Finalmente imprime los tres arrays.

9. Hacer un programa que:
   - Cree y cargue un array unidimensional de 12 elementos de tipo entero.
   - Recorra ese array metiendo en un segundo array todos aquellos elementos que sean pares y mayores que 25.
   - Muestre en pantalla el contenido de los dos arrays unidimensionales.

10. Hacer un programa que lea una secuencia de 15 números enteros, almacenándolos en un array, y a continuación los visualice en orden inverso al de entrada.

11. Hacer un programa de menú con las siguientes opciones:
    i. Suma de dos polinomios de grado N y M.
    ii. Resta de dos polinomios de grado N y M.
    iii. Multiplicación de dos polinomios de grado N y M.

12. Hacer una función para trasponer matrices de N x M elementos, otra para leer matrices desde teclado y otra para visualizar matrices. Hacer un programa que utilizando dichas funciones nos visualice la matriz traspuesta de una introducida desde teclado.

13. Utilizar una matriz de 4 x 10 elementos para representar una baraja con cuatro palos que contenga cada uno las cartas del 1 al 7 y la Sota, el Caballo y el Rey. Utilizando dicha matriz implementar el juego de las 7 y media con dos jugadores (usuario y ordenador como banca).

14. Crear la clase ***Carta*** que representa a una carta de la baraja española. Crear la clase ***Baraja*** que inicialmente contenga a todas las cartas en un montón. Esta clase tendrá un método *void Baraja( )* que permite barajar el montón de cartas. Otro método, *Carta getCarta( )* nos proporciona la carta que está situada en la parte superior del montón retirándola del mismo. Con esta clase realizar un programa para jugar a las siete y media con el ordenador.

15. Crear las clases ***Bombo***, para simular un bombo de n bolas, y ***Carton***, para simular los cartones que se compran en el juego del Bingo.

    La clase *Bombo* debe mantener información sobre si una bola ha salido ya o no, además del número de bolas que tiene inicialmente el bombo, y proporcionar una interfaz que proporcione al menos:

    - Constructor: debe dar los valores más convenientes a las variables de instancia.
    - extraeBola( ): sacará una bola aleatoriamente del bombo, habrá que tener en cuenta que no se puede sacar una bola que ya haya salido anteriormente.

    La clase *Carton* debe mantener información sobre los 15 números que contiene cada cartón, así como del número de bolas que tiene inicialmente el bombo con el que juegan, y del número de aciertos que se tiene en cada momento. Deberá proporcionar una interfaz con al menos:

    - Constructor: inicializará las variables de instancia y generará un cartón con 15 números diferentes de forma aleatoria (teniendo en cuenta el tamaño del bombo).
    - existeNumero(n): comprobará si en el cartón existe el número n.
    - compruebaNumero(n): si existe en el cartón el número n incrementará el número de aciertos del cartón.
    - quedanNumeros( ): devuelve *true* si todavía quedan números sin acertar en el cartón y *false* si el cartón está completo.

    Utilizando las dos clases anteriores hacer un programa ***MainBingo*** para jugar al Bingo con un bombo de 100 números (entre 1 y 100) con el ordenador: tanto el ordenador como el jugador dispondrá de un único cartón, ganará el primero que cante bingo.