EJERCICIOS RESUELTOS EN CLASE DE TEORÍA

1. Realizar un programa que pida 10 números enteros y calcule la media de dichos valores. Realizar una función/procedimiento para rellenar los valores en un vector y otra para calcular la media.

2. (P1) Realiza un programa que pida 10 números enteros y los guarde en un vector. Luego pida otro número y diga si se encuentra en el vector (indicando en qué posición está) o si no se encuentra. Para verificarlo, muestra el contenido del vector por pantalla. Recuerda modelizar el programa.

3. Realiza un programa que sume dos vectores de un espacio vectorial real de dimensión n (R^n). Supón que el espacio nunca será mayor que 50. Para ello deberás pedir la dimensión del espacio antes de pedir los datos y comprobarlo. Modeliza el programa.

4. (P5) Realiza un programa que pida al usuario N números enteros y los inserte de forma ordenada en un vector. (N será un valor que pediremos al principio pero nunca deberá ser mayor de 100). El programa deberá permitir entonces la búsqueda de elementos dentro del vector mediante "búsqueda dicotómica". Para ello (y mientras el usuario no decida salir) se pedirá el número que se desea saber si existe en el vector y se devolverá, en ese caso, su posición o se informará de que no existe. Para verificarlo, muestra el contenido del vector por pantalla. Recuerda modelizar el programa.

EJERCICIOS PARA RESOLVER

5. (P2) **(alumnos.cpp)** Realizar un programa que lea las notas de los 10 alumnos de una clase para una asignatura, calcule la media, y determine cuántos alumnos superan y cuántos están por debajo de la media. La estructura del programa será la siguiente:

```
#include <iostream>
using namespace std;
#define ALUMNOS 10

void leer_notas ( float [ALUMNOS] );
float media ( float [ALUMNOS]);
void sup_inf_media ( float [ALUMNOS], float, int & , int & );
int main ( void );

void leer_notas ( float notas [ALUMNOS] ))
{
   ...
}
float media ( float notas[ALUMNOS] )
{
   ...
}
void sup_inf_media ( float notas[ALUMNOS], float med, int &sup_med, int &inf_med )
{
   ...
}
int main ( void )
{
   float notas[ALUMNOS], med;
   int sup_med, inf_med;

   leer_notas ( notas );
   med = media ( notas);
   sup_inf_media ( notas, med, sup_med, inf_med );

   cout << "La media es " << med << " Y hay " << sup_med;
   cout << " alumnos con nota Superior a la media y " << inf_med;
   cout << " alumnos con nota inferior" << endl;

   system("pause");
   return 0;
}
```

6. (Op1) **(carrera.cpp)** Realizar un programa que lea los tiempos en los que de 10 corredores han acabado una carrera. El programa debe determinar qué corredores tienen el primer, segundo y último puesto, así como cuál es el tiempo medio en que se ha corrido la carrera. La estructura del programa será la siguiente:

```
#include <iostream>
using namespace std;
#define CORREDORES 10

void leer_tiempos ( float [CORREDORES] );
float calcula_media ( float [CORREDORES]);
void calcula_ganadores( float [CORREDORES], int & , int & );
int calcula_perdedor( float [CORREDORES]);

void leer_tiempos ( float datos[CORREDORES] )
{
   ...
}
float calcula_media ( float datos[CORREDORES] )
{
   ...
}
void calcula_ganadores(float datos[CORREDORES],int &primer_puesto,int &segundo_puesto)
{
   ...
}
int calcula_perdedor( float datos [CORREDORES])
{

}
int main ( void )
{
   float tiempos[CORREDORES], med;
   int primer, segundo, ultimo;

   cout << "Dame los tiempos de los 10 corredores " << endl;
   leer_tiempos ( tiempos );

   med = calcula_media ( tiempos);
   calcula_ganadores ( tiempos, primer,segundo );
   ultimo = calcula_perdedor(tiempos);


   cout << "La media de tiempos es " << med << endl;
   cout << "El primer puesto es del corredor número" << primer<< endl;
   cout << "El segundo puesto es del corredor número" << segundo << endl;
   cout << "El ultimo puesto es del corredor número" << ultimo << endl;

   system("pause");
   return 0;
}
```

7. (P3) **(repetidos.cpp)** Diseña un programa que pida el valor de 10 números enteros distintos y los almacene en un vector. Si se da el caso y se trata de introducir un número repetido, el programa advertirá al usuario tan pronto sea posible, y solicitará nuevamente el número hasta que sea diferente de todos los anteriores. A continuación, el programa mostrará los 10 números por pantalla.

8. (Op3) **(capicua.cpp)** Realizar un programa que lea una cantidad determinada (que se pedirá al principio del programa) de números enteros de una sola cifra, almacene dichos números en un vector y compruebe si el número formado por cada uno de los elementos del vector es capicúa o no. Tomar como tamaño máximo del vector, 100 elementos. Hacer una función “LeerVector” y otra, “EsCapicua”.

9. (puntos.cpp) Realizar un programa que pida las coordenadas de dos puntos en el espacio (puntos tridimensionales) y te muestre un menú con las siguientes opciones:

- Mostrar los puntos por pantalla
- Calcular la distancia entre los dos puntos
- Calcular el vector dirección de la recta que pasa por los dos puntos.

Habrá que hacer una función/procedimiento para leer el punto tridimensional, otra para visualizarlo, otra para calcular la distancia y otra para calcular el vector dirección de la recta. Y todas serán llamadas desde el programa principal.

10. (vectores.cpp) Realizar un programa que tenga una función para leer vectores tridimensionales, otra para mostrarlos por pantalla, otra para sumar, otra para multiplicar escalarmente los vectores y otra para multiplicarlos vectorialmente. Realizar la función principal que haga uso de estas funciones (mediante un menú).

Nota: Dados dos vectores {(a1, a2, a3) y (b1 ,b2, b3)}, su producto escalar y su producto vectorial se calculan mediante:

Producto escalar = Σ ai *bi y

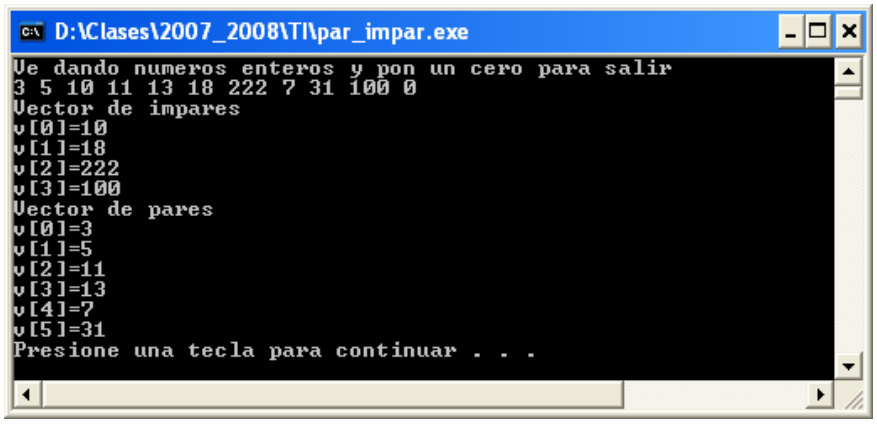
Producto vectorial de dos vectores u(x,y,z) y v(x,y,z), obteniendo: w(x,y,z)

w[x] = u[y] * v[z] - u[z] * v[y];
w[y] = u[z] * v[x] - u[x] * v[z];
w[z] = u[x] * v[y] - u[y] * v[x];

11. (P4) **(sumafila.cpp)** Diseña un programa que lea los elementos de una matriz de 4 × 5 reales y genere un vector de tamaño 4 en el que cada elemento contenga el sumatorio de los elementos de cada fila. El programa debe mostrar la matriz original y el vector en este formato (evidentemente, los valores deben ser los que correspondan a lo introducido por el usuario):

```
        0      1      2      3      4       Suma
0 [ +27.33 +22.22 +10.00  +0.00 -22.22]  -> +37.33
1 [  +5.00  +0.00  -1.50  +2.50 +10.00]  -> +16.00
2 [  +3.45  +2.33  -4.56 +12.56 +12.01]  -> +25.79
3 [  +1.02  +2.22 +12.70 +34.00 +12.00]  -> +61.94
```

12. (par_impar.cpp) Realiza un programa que vaya pidiendo números enteros mientras que no se introduzca el cero y rellene dos vectores, uno con los números pares, y otro con los números impares. Al final, se debe mostrar por pantalla tanto el vector de números pares como el de impares, indicando la posición del vector y el valor que ha sido almacenado. Hacer una función/procedimiento para "CargarVectores" y otra para "VisualizarVector".



13. (VectorCaracteres.cpp) Hacer un programa que:

b) Lea una secuencia de como máximo 1000 caracteres y los almacene en un vector.

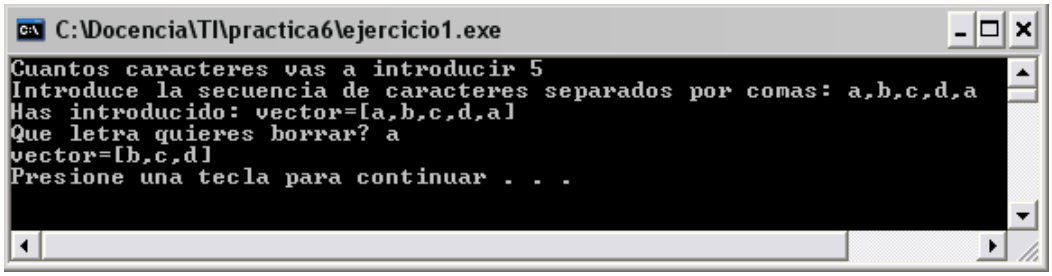
c) Muestre por pantalla el vector leído.

d) Permita eliminar todas las ocurrencias de un carácter en el vector.

e) Muestre por pantalla el vector modificado.

Para ello se deberán realizar 5 funciones:

- Función para leer vectores que tomará como parámetro el número de caracteres a leer y devolverá por referencia el vector de caracteres.
- Función para mostrar el vector que recibirá como parámetros el vector a mostrar y su dimensión.
- Función para modificar un elemento del vector que recibirá como parámetros el vector a modificar, la posición a modificar y el nuevo carácter.
- Función para buscar una letra en el vector y que devolverá la posición en la que se encuentra la letra o -1 si no aparece.
- Función para borrar una letra de un vector que empleará las funciones anteriores para eliminar la primera ocurrencia de una letra en el vector.



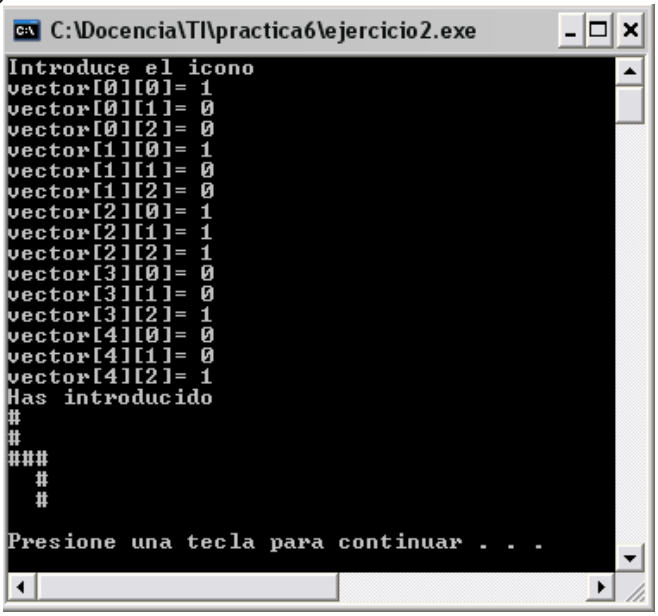
14. (Op2) (icono.cpp) Hacer un programa que permita leer iconos en blanco y negro. Los iconos se almacenarán como matrices (arrays de dos dimensiones). El programa deberá permitir introducir un icono y mostrarlo, representando los 0 por espacios en blanco y los 1 por el carácter '#'.

Se deberá emplear una función/procedimiento que implemente cada una de estas tareas ("LeerMatriz", "VisualizarMatriz" y "VerIcono").

Las matrices que usaremos serán de la forma:

```
#define HORIZ 5
#define VERTI 3

bool icono[HORIZ][VERTI];
```



15. (Op4) **(SumaMatrices.cpp)** Realizar un programa que pida dos matrices y las sume (siempre que sea posible) o avise de que no se pueden sumar. Hacer el diseño descendente del problema (habrá seguro una función/procedimiento para leer la matriz y otra para el cálculo de la suma). Lo primero que se hará es pedir la dimensión de la matriz y luego se leerán sus datos.

Nota:

La suma de dos matrices A y B es otra matriz C teniendo en cuenta que, para poder sumar matrices, ambas deben tener la misma dimensión, es decir:

A←m x n (A es una matriz de m filas y n columnas)

B←p x q (B es una matriz de p filas y q columnas)

Si m=p y n=q entonces:

C←m x n (C será la matriz suma que tendrá m filas y n columnas) y se obtendrá como:

$C_{i,j} = A_{i,j} + B_{i,j}$

16. **(MultiplicaMatrices.cpp)** Realizar un programa que pida dos matrices (utilizar para ello una función de lectura de matices) y las multiplique (comprobando que esta operación se puede realizar).

Nota:

Producto de las matrices A y B es C

A←m x p (A es una matriz de m filas y p columnas)

B←p x n (B es una matriz de p filas y n columnas)

C←m x n (C será una matriz que tendrá m filas y n columnas)

Donde $C_{i,j} = \sum_{k=1}^{p} A_{i,k} * B_{k,j}$

El programa principal deberá de tener la siguiente estructura:

```
#include <iostream>

/* Prototipos de funciones */
...

/* Desarrollo de las funciones necesarias */
...

/* Funcion principal */


int main ( void )
{
   int mat1[TAM][TAM], mat2[TAM][TAM], resultado[TAM][TAM];
….
/* si se puede realizar la operación entonces… */

   llenar_matriz ( mat1 , m , p);
   llenar_matriz ( mat2 , p , n);
   multiplicar ( mat1, mat2, resultado , m , p);
   mostrar_matriz ( resultado );
…
/* si no se puede avisar de ello */
   return 0;
}
```

17. (traspuesta.cpp) Calcular la matriz transpuesta de una matriz dada. Una matriz transpuesta de otra dada es la que resulta de cambiar los valores de las filas de la matriz original por el de las columnas.

Definirse un prototipo de la forma:

```
#define MAX_F 100
#define MAX_C 100
void transpuesta (double matriz[MAX_F][MAX_C], int f, int c);
```

La función "transpuesta" recibe la matriz original introducida por teclado y el número de filas y columnas (parámetros f y c respectivamente) y tiene que calcular su transpuesta.

El programa principal deberá mostrar por pantalla, en formato de matriz (filas x columnas) la matriz original y su transpuesta.

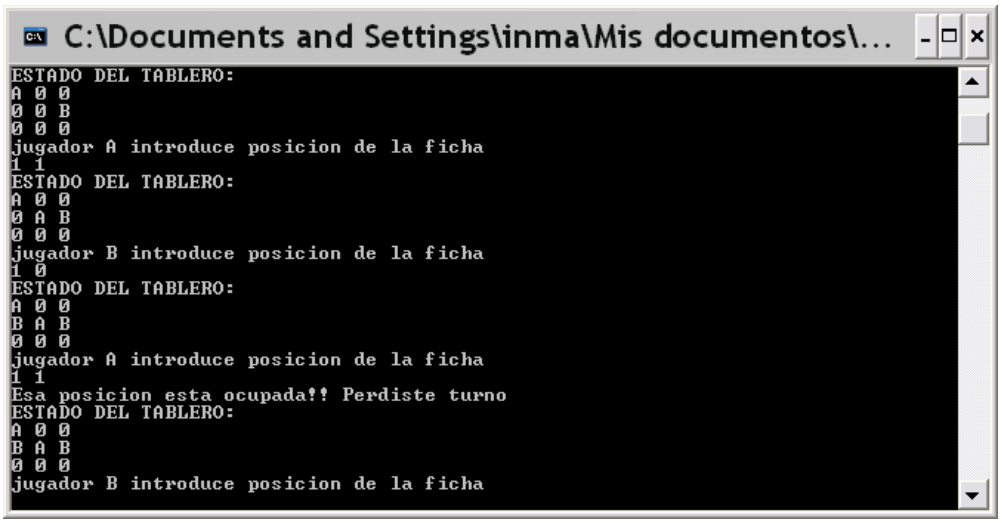
Nota: Hacer también una función para leer matrices y para mostrarlas.

18. (tresenraya.cpp) Queremos un programa que nos permita jugar de forma sencilla al tres en raya a dos jugadores (A y B). El programa permitirá almacenar el estado del tablero (0 si nadie ha insertado ficha en una posición 1 para el jugador A y 2 para el jugador B). Realiza las funciones/procedimientos que:

a. Muestren el estado actual del tablero.

b. Pidan la posición de la ficha a introducir alternativamente a cada uno de los jugadores.

c. Comprueben si el tablero está lleno. Para simplificar esta versión no comprobará si se ha hecho el tres en raya o no.



19. (tresenraya2.cpp) Completar el programa del tres en raya para que cada vez que se inserte una ficha se compruebe si en el tablero hay un tres en raya hecho por alguno de los dos jugadores.

20. (palindroma.cpp) Realizar un programa en C/C++ que pida una frase y nos indique si es o no palíndroma (se lee igual de izquierda a derecha que de derecha a izquierda). Por ejemplo:

Entrada → dabale arroz a la zorra el abad

Salida → cierto

21. (aleatorios.cpp) Implementa un programa que declare un vector de 100 números enteros. El programa deberá entonces mostrar el siguiente menú:

1- Rellenar el vector con números aleatorios
2- Buscar un número con búsqueda lineal
3- Mostrar el vector
4- Salir

Los números aleatorios se deberán generar entre 0 y 1.000. Cada número se deberá insertar ordenado en el vector y podrá haber números repetidos. Para generar números aleatorios se utilizará la función `rand()` de la librería `stdlib.h`, que genera un número aleatorio entre 0 y RAND_MAX. Para generar números entre 0 y el valor que queramos se puede utilizar la siguiente expresión:

`num_aleatorio = rand() * max/RAND_MAX` que genera números aleatorios entre 0 y max – 1.

Recuerda poner en el programa principal "la inicialización de la semilla":

```
srand(time(NULL))
```

22. (adivina.cpp) Escribir un programa por el que el primer jugador introduce una palabra, se reordena aleatoriamente y el segundo jugador debe averiguar cuál era la palabra introducida. No se deben tener en cuenta las mayúsculas y minúsculas, por lo tanto, se deberá realizar una función para convertir todas las cadenas leídas a minúsculas.