EJERCICIOS de Arrays (Vectores y Matrices)

1. ***(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:

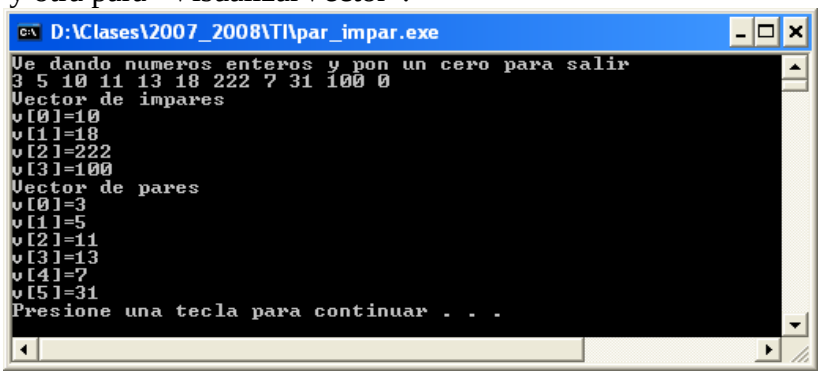
```
#include <iostream>
using namespace std;
#define CORREDORES 10

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;
}
```

2. ***(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.

3. **(*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
4 [  -2.00 -56.20  +3.30  +2.00  +1.00]  -> -51.90
```

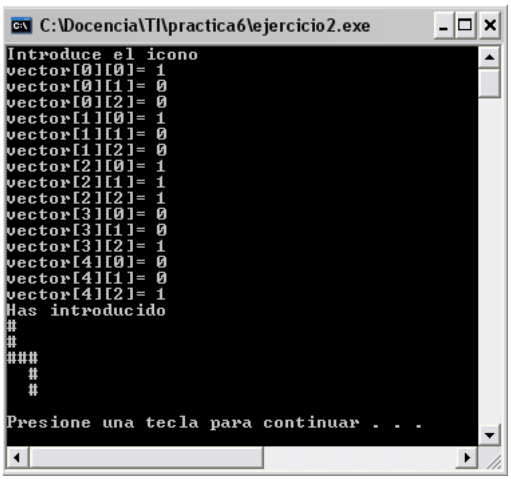
4. ***(par_impar.cpp)*** Realiza un programa que vaya pidiendo números enteros mientras que no se introduzca el cero y rellene dos vectores, uno con los números pares, y otro con los números impares. Al final, se debe mostrar por pantalla tanto el vector de números pares como el de impares, indicando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



5. ***(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];
```



6. **(*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)

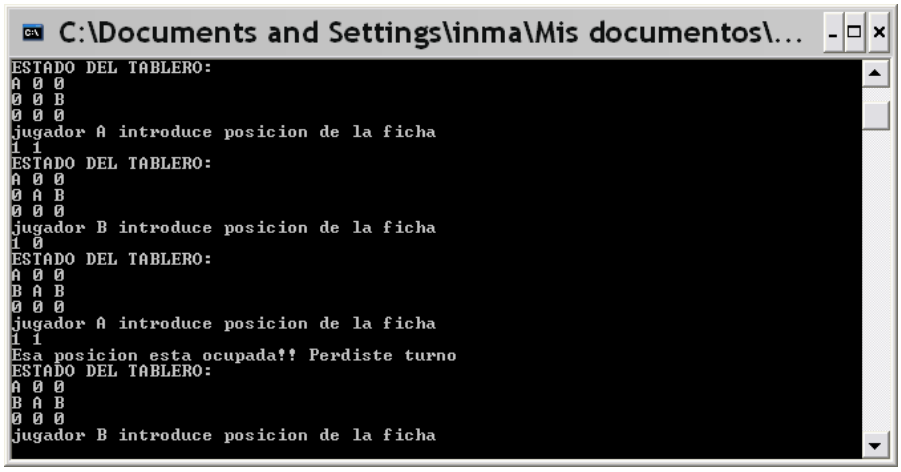
   Si m=p y n=q entonces:

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

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

Opcionales:

7. **(*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.

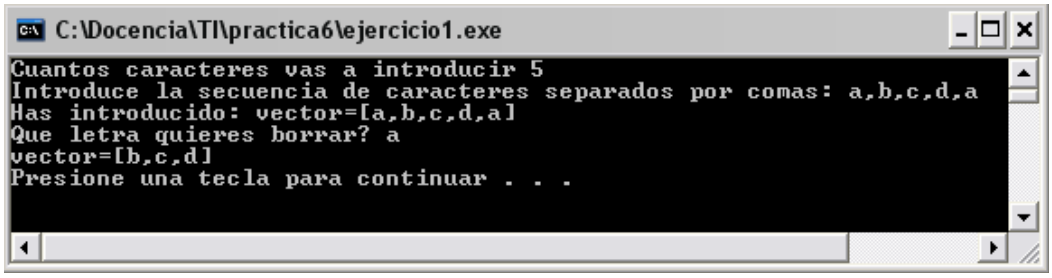


8. **(*tresenraya2.cpp*)** Completar el programa del tres en raya para que cada vez que se inserte una ficha se compruebe si en el tablero hay un tres en raya hecho por alguno de los dos jugadores.

9. **(*caracteres.cpp*)** Hacer un programa que:
   a) Lea una secuencia de como máximo 1000 caracteres y los almacene en un vector.
   b) Muestre por pantalla el vector leído.
   c) Permita eliminar todas las ocurrencias de un carácter en el vector.
   d) 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.



10. **(*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.