UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

INTEGRANTES: JORDY CEVALLOS, ANA ULLOA, BRYAN MIGUITAMA

NRC: 20823

FECHA: 13/06/2025

TALLER 2 U2

```
40              printf ("%d ",mt[i][j]);
41          printf ("\n");
42      }
43  }
```

El ejemplo 3.11 es una ampliación del ejemplo 3.10 en el que se reserva espacio adicional para la matriz `mt`, que es donde se almacenará la traspuesta. Una vez introducida la matriz `matriz` se utilizan dos bucles `for` para asignar al elemento (j,i) de `mt` el valor almacenado en el elemento (i,j) de `matriz`. Finalmente se imprimen ambas matrices por pantalla.

Figura 3.2: Tablero del 4 en raya (ilustración por cortesía de François Haffn

3.7. Juegos con vectores y matrices

Los ejemplos anteriores habrán ayudado a consolidar vuestro manejo de bu vectores y matrices. Es el momento de buscar aplicaciones más lúdicas de los co mientos adquiridos.

Comenzaremos con una versión en C del juego de mesa conocido como 4 en que se ilustra en la figura 3.2.

Como vemos, el código del juego 3.2 es notablemente largo en comparación con todos los programas realizados hasta el momento. Tras la declaración de las variables que serán utilizadas en el programa, se incluyen unas líneas para inicializar todos los elementos de la matriz `tablero` con el carácter 'O'. Nótese que los símbolos empleados para representar las fichas de los jugadores se almacenan en el vector `simbolo` (las fichas del jugador 1 se representan con '+' y las del segundo con '*').

El grueso del código se encuentra entre las líneas 20 y 87, donde se utiliza un bucle `while` para realizar las siguientes tareas:

- Cálculo del jugador al que le toca (líneas 22 a 24): se acumula el número de jugadas realizadas hasta el momento en la variable `numjugadas` y se utiliza el resto de dividir su valor entre dos como base para saber si le toca al primer jugador (`turno`=1) o al segundo (`turno`=2).
- Impresión del tablero (líneas 26 a 38): se imprime el número de cada columna por pantalla y a continuación la matriz `tablero`.
- Elección de columna por parte del jugador al que le toca (líneas 41 a 60): dentro de un bucle `do/while` se pregunta al jugador por la columna elegida. Si la elección es correcta (columna dentro de los límites de la matriz (línea 47) y tiene espacio disponible (líneas 50 a 52), se introduce la ficha con el símbolo correspondiente y se hace `ok`=1, lo que permite la ruptura del bucle. Destacan en este trozo del código dos aspectos:
  - El bucle `for` de la línea 50 repite una instrucción vacía ya que lleva punto y coma al final. Esto se utiliza para empaquetar en una sola línea todo lo necesario para encontrar el primer elemento de la columna escogida en el que se puede introducir una ficha. Como no hay ninguna instrucción que repetir se pone directamente ';' tras `for`. Nótese que el bucle empieza por la última fila de la matriz y retrocede mientras que no se encuentre un hueco disponible e `i>=0`. A la salida, se comprueba el valor de `i` en el `if` para determinar cuál de las dos condiciones propició la ruptura del bucle, de modo que solo se inserta la ficha si se encontró un hueco disponible.
  - La condición de ruptura de `do/while` es `!ok`, que es equivalente a `ok==0`. En efecto, el operador negación '!' conmuta la interpretación lógica de lo que le sucede. Así, si `ok` es igual a 0, `!ok` será interpretado como algo diferente de 0, lo que desde el punto lógico se interpreta como verdadero. Si `ok` vale algo diferente de cero, `!ok` pasa a ser cero, algo que desde el punto de vista lógico se interpreta como falso. Por tanto, al hacer `ok`=1, la condición `!ok` rompe el bucle. Nótese que podría haberse utilizado también este recurso en la línea 20 con la condición del `while`.
- Búsqueda de 4 en raya (líneas 62 a 85): el programa comprueba si la ficha introducida lleva a la finalización del programa. Ello requiere recorrer la matriz y comprobar si la ficha de la posición (`i`,`j`) es igual a las que la suceden en horizontal (líneas 62 a 67), vertical (líneas 69 a 73) y diagonal (líneas 75 a 85). Cada uno de los bucles `for` empleados ajusta el rango de variación de las variables `i` y `j` para evitar que durante la comprobación se acceda a una casilla fuera de los límites definidos para la matriz. Si se encuentran 4 en raya, la variable `gameover` pasa a valer 1, lo que desencadena el fin del juego. Nótese que los `for` e `if` aquí utilizados afectan respectivamente a la instrucción que lo sucede, por lo que pueden omitirse las llaves, lo que aligera el número de líneas del código.

Desde la línea 89 hasta el final se indica el jugador vencedor y se imprime la disposición final del tablero.

En la figura 3.3 se muestra una captura de pantalla del aspecto del programa en ejecución.

```
Columnas a elegir:
0 1 2 3 4 5 6 7
O O O O O O O O
O O O O O O O O
O O O O O O O O
O O O O O O O O
O O O O O O O O
O O O O O O O O
O O O * O O O O
O * + + + * O O

Le toca a jugador 1. Elija columna:
```

Figura 3.3: Tablero del 4 en raya de nuestra versión en C.

- REQUISITOS FUNCIONALES

- El sistema deberá permitir iniciar una nueva partida.
- El sistema deberá mostrar un tablero vacío de 6 filas por 7 columnas.
- El sistema deberá alternar el turno entre dos jugadores.
- El sistema deberá permitir a los jugadores seleccionar una columna para colocar su ficha.
- El sistema deberá colocar la ficha en la posición más baja disponible de la columna seleccionada.
- El sistema deberá detectar automáticamente si un jugador ha logrado alinear 4 fichas consecutivas de forma vertical, horizontal o diagonal.
- El sistema deberá mostrar un mensaje de victoria cuando un jugador gane.
- El sistema deberá detectar si el tablero está lleno sin ganador y declarar un empate.
- El sistema deberá ofrecer la opción de reiniciar la partida luego de que termine.

- Pseint