Práctica de arreglos

Introducción

En esta guía, creará un programa para jugar “equis cero” contra la computadora. Para ello, utilizará todo lo que ha aprendido hasta este momento sobre el lenguaje C++.

Una vez finalizado, su juego se verá así:

```
Juego de equis-cero

    0   1   2
   -----------
0 |   |   |   |
   -----------
1 |   |   |   |
   -----------
2 |   |   |   |
   -----------

Jugador:
Fila: 1
Columna: 1

    0   1   2
   -----------
0 |   |   |   |
   -----------
1 |   | X |   |
   -----------
2 |   |   |   |
   -----------

Computadora:

    0   1   2
   -----------
0 | 0 |   |   |
   -----------
1 |   | X |   |
   -----------
2 |   |   |   |
   -----------
```

El jugador indicará a la computadora la casilla que desea marcar ingresando un número de fila y columna. La computadora responderá marcando una casilla al azar (no es una estrategia muy inteligente, por lo que probablemente usted ganará la mayoría de juegos, sin embargo, crear una inteligencia artificial para jugar equis-cero no está dentro del alcance de la materia).

Para simplificar aún más el escenario, el jugador humano siempre iniciará el juego. Es decir, le tocan las ‘X’s, mientras que la computadora siempre jugará con los ‘0’s.

El juego finaliza cuando uno de los jugadores haya logrado colocar su símbolo (‘X’ o ‘0’) en una fila, columna o diagonal, o cuando ya no haya casillas disponibles para marcar (empate).

Cuando el juego ha finalizado, se da al usuario la oportunidad de jugar de nuevo.

```
Jugador:
Fila: 0
Columna: 0

    0   1   2
   -----------
0 | X | 0 |   |
   -----------
1 |   | X |   |
   -----------
2 | 0 |   | X |
   -----------

Usted gana!!!
Desea jugar de nuevo (S|n)? _
```

Creación del programa

Abra Dev-C++ y cree un nuevo archivo de código fuente con el nombre **equis_cero.cpp** y agregue el siguiente código inicial:

```
1  #include <iostream>
2  #include <string>
3  #include <cstdlib>
4  #include <ctime>
5  #include <cctype>
6
7  using namespace std;
8
9  const int LADO = 3;
10
11 // Prototipos de funciones
12 void imprimirTablero(char tablero[][LADO]);
13 int marcarCasillaJugador(char tablero[][LADO]);
14 int leerNumero(string indicacion, string mensajeError, int min, int max);
15 int marcarCasillaComputadora(char tablero[][LADO]);
16 char ganadorFilas(char tablero[][LADO]);
17 char ganadorColumnas(char tablero[][LADO]);
18 char ganadorDiagonales(char tablero[][LADO]);
19 char determinarGanador(char tablero[][LADO]);
20 void jugar();
21
22 int main() {
23     char jugarDeNuevo;
24     srand(time(0)); // Inicializa la semilla de los numeros aleatorios
25     do {
26         jugar();
27         cout << "Desea jugar de nuevo (S|n)? ";
28         cin >> jugarDeNuevo;
29         cout << endl;
30     } while (toupper(jugarDeNuevo) == 'S');
31
32     return 0;
33 }
```

Como puede observar en las líneas 12-20 de la imagen anterior, escribirá un total de 7 funciones (líneas 13-19, sin incluir **main**) y 2 procedimientos (líneas 12 y 20).

Función main()

La función **main**() de las líneas 22-33 inicializa un generador de números aleatorios (línea 24, esto se hace una sola vez en todo el programa), presenta un juego de equis-cero, invocando a la función **jugar()** en la línea 26 y luego pregunta al usuario si quiere jugar otra vez. Este ciclo se repite mientras el usuario continúe respondiendo 'S' (línea 30).

Función jugar()

Aquí es donde sucede la mayor parte de la acción. A grandes rasgos, esta función:

1. Crea una matriz para representar un **tablero** de equis-cero (línea 36) y una variable para llevar el control de las casillas no marcadas (**casillasLibres**, en la línea 40).
   El valor inicial de todas las celdas de la matriz es el número 0 (como lo indica {}).
2. Imprime el tablero de equis-cero en la línea 42, invocando a la función **imprimirTablero()**, a la que le pasa de parámetro la variable **tablero**.

3. Mientras haya casillas libres o no exista un ganador (línea 43), se invoca a las funciones **marcarCasillaJugador()** o **marcarCasillaComputadora()**, de acuerdo a quien le toca jugar (lo que se determina con la variable booleana **turnoJugador**, en las líneas 44-52).
4. Una vez finaliza el bucle, imprime el ganador (líneas 53-62).

```
34
35  void jugar() {
36      char tablero[LADO][LADO] = {};
37      char ganador = 0;
38
39      cout << "Juego de equis-cero" << endl;
40      int casillasLibres = LADO * LADO;
41      bool turnoJugador = true;
42      imprimirTablero(tablero);
43
44      while (casillasLibres > 0 && ganador == 0) {
45          if (turnoJugador) {
46              casillasLibres -= marcarCasillaJugador(tablero);
47          } else {
48              casillasLibres -= marcarCasillaComputadora(tablero);
49          }
50          turnoJugador = !turnoJugador; // Turno del otro jugador
51          ganador = determinarGanador(tablero);
52      }
53      switch (ganador) {
54          case 'X':
55              cout << "Usted gana!!!" << endl;
56              break;
57          case '0':
58              cout << "La computadora gana :(" << endl;
59              break;
60          default:
61              cout << "Empate." << endl;
62      }
63  }
```

Procedimiento imprimirTablero()

Escriba cuidadosamente el código que se muestra a continuación para implementar el procedimiento **imprimirTablero()**, que recibe como parámetro el **tablero**.

Use los puntos de las líneas 67 y 74 para identificar la cantidad de espacios entre caracteres para que el tablero se imprima correctamente.

```
64
65  void imprimirTablero(char tablero[][LADO]) {
66      // Cuente los espacios usando los puntos
67      // Guia: ...............   ...............
68      cout << "\n     0   1   2\n    -----------" << endl;
69      for (int fila = 0; fila < LADO; fila++) {
70          cout << " " << fila << " |";
71          for (int columna = 0; columna < LADO; columna++) {
72              cout << " " << tablero[fila][columna] << " |";
73          }
74          // Guía: ................
75          cout << "\n    -----------" << endl;
76      }
77      cout << endl;
78  }
79
```

Función leerNumero()

Esta función se encarga de leer un número ingresado por el usuario, con ciertas validaciones. Primero, en la línea 85, la función verifica que el valor ingresado efectivamente sea un número y que se encuentre en el rango [min, max), en caso contrario, descarta el valor ingresado por el usuario (líneas 86-87), muestra un mensaje de error (línea 88) y le da otra oportunidad al usuario para ingresar un valor (línea 89). Finalmente, cuando el valor ingresado pasa las validaciones, se devuelve al programa principal (línea 92).

```
79
80  int leerNumero(string indicacion, string mensajeError, int min, int max) {
81      int numero;
82      cout << indicacion << ": ";
83
84      // Valida que la entrada sea un entero
85      while (!(cin >> numero) || (numero < min || numero >= max)) {
86          cin.clear(); // limpia la entrada estandar
87          cin.ignore(10000, '\n'); // descarta la entrada hasta encontrar un salto de linea
88          cout << mensajeError << endl; // muestra un mensaje de eror
89          cout << indicacion << ": ";  // le da otra oportunidad al usuario
90      }
91
92      return numero;
93  }
94
```

Esta función es utilizada por la función **marcarCasillaJugador**(), cuyo código se explica en la página siguiente.

Función marcarCasillaJugador()

Esta función le pide al usuario que ingrese el número de **fila** y de **columna** de la celda a marcar con una letra 'X'. Para validar que el número ingresado esté entre 0 y 2, se usa la función **leerNumero()** que escribió antes (líneas 99-100). Además, si la casilla está ocupada (lo que ocurre cuando el valor guardado en esa casilla es el número 0, véase la línea 101), se da otra oportunidad al usuario para ingresar valores de **fila** y **columna** (líneas 101-104). Cuando el usuario ingresa una casilla libre (una vez finaliza el bucle), esta se marca con una 'X' (línea 106) y se imprime el tablero en pantalla para que se pueda apreciar el resultado (línea 107).

La última línea de la función (línea 108) devuelve el número de casillas marcadas al programa principal (siempre será el número 1).

```
95  int marcarCasillaJugador(char tablero[][LADO]) {
96      int fila, columna;
97      cout << "Jugador:" << endl;
98      string mensaje = "Ingrese un valor entre 0 y " + to_string(LADO);
99      fila = leerNumero("Fila", mensaje, 0, LADO);
100     columna = leerNumero("Columna", mensaje, 0, LADO);
101     while (tablero[fila][columna] != 0) {
102         cout << "Casilla ocupada. Intente de nuevo" << endl;
103         fila = leerNumero("Fila", mensaje, 0, LADO);
104         columna = leerNumero("Columna", mensaje, 0, LADO);
105     }
106     tablero[fila][columna] = 'X';
107     imprimirTablero(tablero);
108     return 1;
109 }
110
```

Función marcarCasillaComputadora()

Esta función opera de forma muy similar a la función anterior, excepto que los valores de las variable **fila** y **columna** se generan al azar (líneas 113-114) y no se muestran mensajes de error en pantalla. Una vez se encuentra una casilla libre (líneas 115-118) esta se marca con un '0' (línea 119) y se imprime el tablero en pantalla (líneas 120-121).

Esta función devuelve el número de casillas marcadas al programa principal (línea 122).

```
111 int marcarCasillaComputadora(char tablero[][LADO]) {
112     int fila, columna;
113     fila = rand() % LADO;
114     columna = rand() % LADO;
115     while (tablero[fila][columna] != 0) {
116         fila = rand() % LADO;
117         columna = rand() % LADO;
118     }
119     tablero[fila][columna] = '0';
120     cout << "Computadora:" << endl;
121     imprimirTablero(tablero);
122     return 1;
123 }
```

Determinar el ganador

Después de marcar una casilla, el programa revisará el tablero completo. Si hay tres no vacías con el mismo carácter (una 'X' o un '0') en una fila, columna o diagonal, se declara un ganador. El proceso es el siguiente:

a. Verificar si hay un ganador en las filas (línea 129).
b. Si aún no hay un ganador, verificar las columnas (líneas 132-133).
c. Si aún no hay un ganador, verificar las diagonales (líneas 136-137).

Si tras realizar estas tres verificaciones no hay un ganador, el ganador es 0.

Note que, para C++, el número **0** (sin apóstrofes) es diferente del carácter **'0'** (entre apóstrofes). El número 0 representa la ausencia de un valor (código ASCII 0), mientras que el carácter '0' representa un símbolo numérico (código ASCII 48). Ver tabla de códigos ASCII en:

https://ascii.cl/es/

El código de la función **determinarGanador** se reproduce a continuación.

```
124
125 char determinarGanador(char tablero[][LADO]) {
126     char ganador;
127
128     // Ganador por filas?
129     ganador = ganadorFilas(tablero);
130
131     // Ganador por columnas?
132     if (ganador == 0)
133         ganador = ganadorColumnas(tablero);
134
135     // Ganador por diagonales?
136     if (ganador == 0)
137         ganador = ganadorDiagonales(tablero);
138
139     return ganador;
140 }
141
```

Esta función se auxiliar de otras tres funciones, correspondientes a las tres posibilidades mencionadas anteriormente:

a. ganadorFilas()
b. ganadorColumnas()
c. ganadorDiagonales()

El valor devuelto por estas funciones siempre será 'X', '0' (el carácter cero) o 0 (el número cero, para indicar que no hay un ganador).

Función ganadorFilas()

Para saber si hay un ganador a nivel de filas, el programa examina el primer carácter de cada fila (línea 145) y luego se cuenta cuántas casillas tienen el mismo valor (líneas 146-158) para las celdas no vacías (líneas 147-153). Si este número coincide con el número máximo de celdas en la fila, la función devuelve un ganador (líneas 154-156). El proceso se repite hasta examinar todas las celdas y, de no existir un ganador a nivel de filas, este se considera nulo (línea 159).

```
142 char ganadorFilas(char tablero[][LADO]) {
143     for (int fila = 0; fila < LADO; fila++) {
144         // Primer caracter de la fila
145         char caracter = tablero[fila][0];
146         int veces = 0;
147         if (caracter != 0) {
148             // Contar cuantas veces aparece el caracter en la fila
149             for (int columna = 0; columna < LADO; columna++) {
150                 if (tablero[fila][columna] == caracter) {
151                     veces++;
152                 }
153             }
154             if (veces == LADO) {
155                 return caracter;
156             }
157         }
158     }
159     return 0;
160 }
161
```

Función ganadorColumnas()

La lógica para saber si hay un ganador a nivel de columnas es similar, solo que se examina el primer carácter de cada columna.

```
162 char ganadorColumnas(char tablero[][LADO]) {
163     for (int columna = 0; columna < LADO; columna++) {
164         // Primer caracter de la columna
165         char caracter = tablero[0][columna];
166         int veces = 0;
167         if (caracter != 0) {
168             // Contar cuantas veces aparece el caracter en la columna
169             for (int fila = 0; fila < LADO; fila++) {
170                 if (tablero[fila][columna] == caracter) {
171                     veces++;
172                 }
173             }
174             if (veces == LADO) {
175                 return caracter;
176             }
177         }
178     }
179     return 0;
180 }
181
```

Finalmente, se examinan individualmente las dos diagonales.

```
182 char ganadorDiagonales(char tablero[][LADO]) {
183     char caracter;
184     int veces;
185 
186     // Diagonal de izquierda a derecha (\)
187     caracter = tablero[0][0];
188     veces = 0;
189     if (caracter != 0) {
190         for (int i = 0; i < LADO; i++)
191             if (tablero[i][i] == caracter)
192                 veces++;
193         if (veces == LADO)
194             return caracter;
195     }
196 
197     // Diagonal de derecha a izquierda (/)
198     caracter = tablero[0][LADO - 1];
199     veces = 0;
200     if (caracter != 0) {
201         for (int i = 0; i < LADO; i++)
202             if (tablero[i][LADO - 1 - i] == caracter)
203                 veces++;
204         if (veces == LADO)
205             return caracter;
206     }
207 
208     return 0;
209 }
```

Con esta función finaliza su programa de equis-cero. Ejecute su programa y juegue varias veces contra la computadora para verificar que funciona correctamente, incluyendo las validaciones de los números de fila y columna y que se prevenga que el usuario trate de llenar una celda previamente marcada.

ENTREGABLES:
Suba a la U-Virtual el archivo de código fuente **equis_cero.cpp**. Coloque su nombre y su carnet como un comentario (una línea que inicia con //) al principio del archivo. Agregar capturas de pantalla de la ejecución, por favor no comprimir los archivos.