

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4
5 using namespace std;
6
7 /*****
8     Constants
9 *****/
10 const int ROWS = 3;
11 const int COLS = 3;
12 const int PLAYER_ONE = 1;
13 const int PLAYER_TWO = 2;
14 const int BLANK = 0;
15
16 /*****
17     Function Prototypes
18 *****/
19 void displayMenu ();
20 void initializeBoard (int board[][COLS]);
21 void drawBoard (int board[][COLS]);
22 char getBoardChar (int board[][COLS], int row, int col);
23 void getUserSelection (int board[][COLS], int &row, int &col);
24 bool determineWinner (int board[][COLS], int &winner);
25 bool checkTiedGame (int board[][COLS]);
26 void displayEndingMessage (int winner);
27
28 /*****
29     Function:    main
30
31     Description: Displays the menu, then depending on the user's selection,
32                 either encodes or decodes a message.
33
34     Parameters:  None
35
36     Returned:   Exit Status
37 *****/
38 int main ()
39 {
40     int gameBoard[ROWS][COLS];
41     int userRow, userCol;
42     bool bGameOver = false;
43     int winner = 0;
44     int currentPlayer;
45
46     initializeBoard (gameBoard);
47
48     currentPlayer = PLAYER_ONE;
49
50     while (false == bGameOver)
51     {
52         displayMenu ();
```

```
53     drawBoard (gameBoard);
54     cout << endl << "Player " << currentPlayer;
55     getUserSelection (gameBoard, userRow, userCol);
56     gameBoard[userRow][userCol] = currentPlayer;
57     system ("pause");
58
59     displayMenu ();
60     drawBoard (gameBoard);
61     bGameOver = determineWinner (gameBoard, winner);
62     if (false == bGameOver)
63     {
64         bGameOver = checkTiedGame (gameBoard);
65     }
66
67     if (PLAYER_ONE == currentPlayer)
68     {
69         currentPlayer = PLAYER_TWO;
70     }
71     else
72     {
73         currentPlayer = PLAYER_ONE;
74     }
75 }
76
77 displayEndingMessage (winner);
78
79 return EXIT_SUCCESS;
80 }
81
82 /*****
83 Function:     displayTitle
84
85 Description:  Displays the title and then calls the function drawBoard to
86               display the board
87
88 Parameters:   None
89
90 Returned:     None
91 *****/
92 void displayMenu ()
93 {
94     system ("CLS");
95     cout << "*****" << endl
96          << "   Tic-Tac-Toe   " << endl
97          << "*****" << endl << endl;
98 }
99
100 /*****
101 Function:     initializeBoard
102
103 Description:  Initialize all of the elements in the board to blanks
104
```

```

105 Parameters: board - 2D array that represents the board
106
107 Returned: None
108 *****/
109 void initializeBoard (int board[][COLS])
110 {
111 }
112
113 *****/
114 Function: drawBoard
115
116 Description: Displays the board to the screen. This function must call
117 the function displayBoardChar on each element in the array
118
119 Parameters: board - 2D array that represents the board
120
121 Returned: None
122 *****/
123 void drawBoard (int board[][COLS])
124 {
125 }
126
127 *****/
128 Function: getBoardChar
129
130 Description: Returns a single board character:
131 If value is 0 - returns a blank
132 If value is 1 - returns an X
133 If value is 2 - returns an O
134
135 Parameters: board - 2D array that represents the board
136 row - the row index of the element
137 col - the col index of the element
138
139 Returned: the character representing the player number
140 *****/
141 char getBoardChar (int board[][COLS], int row, int col)
142 {
143 return ' ';
144 }
145
146 *****/
147 Function: getUserSelection
148
149 Description: Asks the user to enter in their selection and makes sure that
150 the user selection is valid and that the element in the array
151 is blank
152
153 Parameters: board - 2D array that represents the board
154 row - the row index of the element the user selected
155 col - the col index of the element the user selected
156

```

```
157     Returned:     none
158     *****/
159 void getUserSelection (int board[][COLS], int &row, int &col)
160 {
161 }
162
163 /*****
164     Function:     determineWinner
165
166     Description:  Determines the winner of the game. A win can happen diagonally,
167                  vertically, or horizontally.
168
169     Parameters:   board - 2D array that represents the board
170                  winner - The winning player. Either PLAYER_ONE or PLAYER_TWO
171
172     Returned:     true if a winner is found, false otherwise
173     *****/
174 bool determineWinner (int board[][COLS], int &winner)
175 {
176     return true; // REPLACE THIS
177 }
178
179 /*****
180     Function:     checkTiedGame
181
182     Description:  Determines if all of the elements in the board have been filled.
183                  If they have been filled, then the game is a tie.
184
185     Parameters:   board - 2D array that represents the board
186
187     Returned:     true if there are no empty slots on the board, false otherwise
188     *****/
189 bool checkTiedGame (int board[][COLS])
190 {
191     return true; // REPLACE THIS
192 }
193
194 /*****
195     Function:     displayEndingMessage
196
197     Description:  Display a message depending on the outcome of the game:
198                  "Congratulations player one! You have won."
199                  "Congratulations player two! You have won."
200                  "Cat's Game. The game was a tie."
201
202     Parameters:   winner - The winning player
203
204     Returned:     none
205     *****/
206 void displayEndingMessage (int winner)
207 {
208 }
```