

```
1 /
    *****
    ***
2 * AUTHOR      : Andrew Gharios
3 * STUDENT ID  : 1449366
4 * AS #1      : Multi-Dimensional Array - Tic Tac Toe
5 * CLASS      : CS1B
6 * SECTION    : M-TH: 5-7:20p
7 * DUE DATE   : 6/28/21
8 *****
    */
9 #ifndef HEADER_H_
10 #define HEADER_H_
11 #include <iostream>
12 #include <iomanip>
13 #include <string>
14 #include <stdlib.h>
15 #include <time.h>
16 using namespace std;
17
18 const int NUM_COLS = 3;
19 const int NUM_ROWS = 3;
20 const int AR_SIZE = 9;
21
22 /
    *****
    **
23 * OutputInstruct
24 * This function outputs instructions to the users. There are no input
25 * or output parameters for this function as it only displays text to
26 * the screen.
27 *
28 * RETURNS: nothing
29 * Displays the instructions to the user
30 *****/
31 void OutputInstruct();
32
33 /
    *****
    **
34 * InitBoard
35 * This function initializes each spot in the board to a space ' '.
36 *
37 * RETURNS: Board initialized with all spaces
38 *****/
39 void InitBoard(char boardAr[][NUM_COLS]); // OUT - tic tac toe board
40
41 /
    *****
```

```

    **
42 * DisplayBoard
43 * This function outputs the tic tac toe board including the tokens
44 * played in the proper format (as described below).
45 *
46 1 2 3
47 * [1][1] | [1][2] | [1][3]
48 * | |
49 * 1 | |
50 * | |
51 * -----
52 * [2][1] | [2][2] | [2][3]
53 * | |
54 * 2 | |
55 * | |
56 * -----
57 * [3][1] | [3][2] | [3][3]
58 * | |
59 * 3 | |
60 * | |
61 *
62 * RETURNS: nothing
63 * à outputs the current state of the board
64 ASSIGNMENT #2 Multi-DimensionalArrays - TIC TAC TOE CS1B
65 4 of5
66 *****/
67 void DisplayBoard(const char boardAr[][NUM_COLS]); // IN - tic tac toe board
68
69 /
    *****/
    **
70 * GetPlayers
71 * This function prompts the user and gets the input for the players' names.
72 * playerX will always contain the name of the player that is using the X
73 * playerO will always contain the name of the player that is using the O
74 *
75 * RETURNS: the players names through the variables playerX and playerO.
76 *****/
77 void GetPlayers(string& playerX, // OUT - player X's name
78 string& playerO); // OUT - player O's name
79
80 // As this was written in class - you need to document
    this
81 void GetAndCheckInp(char boardAr[][NUM_COLS], char token, string playerX,
82 string playerO);
83 /

```

```

*****
**
84 * SwitchToken
85 * This function switches the active player.
86 * It takes in a parameter representing the current player's token
87 * as a character value (either an X or an O) and returns the opposite.
88 * For example, if this function receives an X it returns an O. If it
89 * receives an O it returns an X.
90 *
91 * RETURNS: the token opposite of the one in which it receives.
92 *****/
93 char SwitchToken(char token); // IN - current player's token ('X' or 'O')
94
95 /
*****
**
96 * CheckWin
97 * This function checks to see if either player has won. Once it is
98 * possible for a win condition to exist, this should run after each a
99 * player makes a play.
100 *
101 * RETURNS the character value of the player that won or a value that
102 * indicates a tie.
103 *****/
104 char CheckWin(const char boardAr[][NUM_COLS], // IN - tic tac toe board
105               char token); // IN - token of who's playing.
106
107 /
*****
**
108 * OutputWinner
109 * This function receives as input a character indicating which player won
110 * or if the game was a tie and outputs an appropriate message. This function
111 * does not return anything as it simply outputs the appropriate message to
112 * the screen.
113 *
114 * RETURNS: nothing
115 * à Displays the winner's name
116 *****/
117 void OutputWinner(char whoWon, // IN - represents the winner or a value
118                  // indicating a tied game.
119                  string playerX, // OUT - player X's name
120                  string playerO); // OUT - player O's name
121
122 /
*****
**
123 * PrintHeaderFile
124 * This function will output the header information

```

```
125  *
126  ****
    /
127 void PrintHeaderFile(ostream& output,    // IN - output datatype.
128     string asName,        // IN - assignment name
129     int asNum,            // IN - assignment number
130     string studentName,   // IN - student's name
131     string classInfo,     // IN - class that is being taken
132     char asType,          // IN - assignment type
133     long long studentID); // IN - student ID
134
135 #endif /* HEADER_H_ */
136
```

```

1  /
    *****
    ***
2  * AUTHOR      : Andrew Gharrios
3  * STUDENT ID  : 1449366
4  * AS #2       : Multi-Dimensional Array - Tic Tac Toe
5  * CLASS       : CS1B
6  * SECTION     : M-TH: 5-7:20p
7  * DUE DATE    : 6/29/21
8  * *****
    */
9  #include "Header.h"
10
11 /
    *****
    ***
12 * Multi-Dimensional Array - Tic Tac Toe
13 * -----
    -
14 * This program will interact with the user through a menu and allow them to
15 * play tic tac toe. The user has the option to set players name, play in two
16 * players and then in single player vs the computer.
17 * -----
    -
18 * INPUT:
19 * playerX : name of player using token X.
20 * playerO : name of player using token O.
21 * menuPick : option pick of menu.
22 * *****
    */
23 int main()
24 {
25     /
        *****
        ***
26     * CONSTANTS
27     *
        -----
        -
28     * OUTPUT - USED FOR CLASS HEADING
29     *
        -----
        -
30     * PROGRAMMER : Programmer's Name
31     * CLASS      : Student's Course
32     * SECTION    : Class Days and Times
33     * LAB_NUM    : Lab Number (specific to this lab)
34     * LAB_NAME   : Title of the Lab
35     * *****

```

```
    /
36
37     const string AS_NAME = "Functions and Arrays";
38     const int AS_NUM = 1;
39     const string STUDENT_NAME = "Andrew Gharrios";
40     const string CLASS_INFO = "M-Th 5-7:20p";
41     const char AS_TYPE = 'A';
42     const long long STUDENT_ID = 1449366;
43
44     char boardAr[NUM_ROWS][NUM_COLS]; // CALC - Playing board for tictactoe.
45     string playerX; // IN & OUT - Player using token X.
46     string playerO; // IN & OUT - Player using token O.
47     char menuPick; // IN & CALC - Player's menu pick.
48     char token; // CALC - Token being played.
49     char win; // CALC & OUT - Match winner.
50     int plays; // CALC - How many rounds user played.
51     int compCol; // CALC - Computer's column value.
52     int compRow; // CALC - Computer's row value.
53     bool winner; // CALC - Condition to check if someone won.
54
55     srand(time(NULL));
56
57     PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO,
58         AS_TYPE, STUDENT_ID);
59
60     OutputInstruct();
61
62     cout << "a. Exit" << endl;
63     cout << "b. Set Player Names" << endl;
64     cout << "c. Play in Two Player Mode" << endl;
65     cout << "d. Play in One Player Mode" << endl;
66
67     cout << endl << "Enter option: ";
68     cin.get(menuPick);
69     cin.ignore(10000, '\n');
70
71     while (menuPick != 'a')
72     {
73         switch (menuPick)
74         {
75             case 'b' :
76                 GetPlayers(playerX, playerO);
77                 break;
78
79             case 'c' :
```

```
80      InitBoard(boardAr);
81      token = 'X';
82      plays = 0;
83      winner = false;
84
85      while (!winner && plays < 8)
86      {
87          DisplayBoard(boardAr);
88          GetAndCheckInp(boardAr, token, playerX, playerO);
89          plays++;
90
91          if (plays > 4)
92          {
93              win = CheckWin(boardAr, token);
94
95              if (win == 'X' || win == 'O')
96              {
97                  winner = true;
98              }
99          }
100
101          token = SwitchToken(token);
102      }
103
104      if (!winner)
105      {
106          win = 'T';
107      }
108
109      OutputWinner(win, playerX, playerO);
110      break;
111
112      case 'd' :
113          InitBoard(boardAr);
114          token = 'X';
115          plays = 0;
116          winner = false;
117
118          while (!winner && plays < 8)
119          {
120              DisplayBoard(boardAr);
121              if (token == 'X')
122              {
123                  GetAndCheckInp(boardAr, token, playerX, playerO);
124              }
125              else
126              {
127                  do
128                  {
```

```
129         compCol = rand() % 3;
130         compRow = rand() % 3;
131
132         if (!isspace(boardAr[compRow][compCol]))
133         {
134             compCol = rand() % 3;
135             compRow = rand() % 3;
136         }
137
138     } while (!isspace(boardAr[compRow][compCol]));
139
140     boardAr[compRow][compCol] = token;
141
142 }
143
144 plays++;
145
146 if (plays > 4)
147 {
148     win = CheckWin(boardAr, token);
149
150     if (win == 'X' || win == 'O')
151     {
152         winner = true;
153     }
154 }
155
156 token = SwitchToken(token);
157 }
158
159 if (!winner)
160 {
161     win = 'T';
162 }
163
164 OutputWinner(win, playerX, playerO);
165 break;
166
167 }
168
169 cout << "a.  Exit" << endl;
170 cout << "b.  Set Player Names" << endl;
171 cout << "c.  Play in Two Player Mode" << endl;
172 cout << "d.  Play in One Player Mode" << endl;
173
174 cout << endl << "Enter option: ";
175 cin.get(menuPick);
176 cin.ignore(10000, '\n');
177
```



---

```
178     }  
179  
180  
181  
182     return 0;  
183  
184 }
```

```
1  #include "Header.h"
2
3  void OutputInstruct()
4  {
5      cout << "Welcome to Tic Tac Toe!\n" << endl;
6      cout << "Select each player's name and token(X or O).";
7      cout << "\n Then select if you want to have 2 players or play with the      ↗
           computer.";
8      cout << "\nEach round you will be prompted to select a location to input      ↗
           your token.";
9      cout << "\nThe winner will be declared at the end. Good luck and enjoy!!!" ↗
           << endl;
10 }
```

```
1  #include "Header.h"
2
3  /*****
4  * InitBoard
5  * This function initializes each spot in the board to a space ' '.
6  *
7  * INPUTS:
8  *   boardAr - Gameboard.
9  *
10 * No outputs.
11 *****/
12 void InitBoard(char boardAr[][NUM_COLS]) // OUT - tic tac toe board
13 {
14     int row;
15     int col;
16
17     for (row = 0; row < NUM_COLS; row++)
18     {
19         for (col = 0; col < NUM_COLS; col++)
20         {
21             boardAr[row][col] = ' ';
22         }
23     }
24 }
```

```
1  #include "Header.h"
2
3  /*****
4  * The following function is provided for you... please desk check it and ensure
5  * that you thoroughly understand it. MODIFY it as stated below!
6  *
7  * 1 - Be sure to document the following in detail!
8  * (demonstrate that you understand this code segment).
9  * 2 - Modify the variable names to something more appropriate.
10 * 3 - Use appropriate constants if necessary.
11 *****/
12 void DisplayBoard(const char boardAr[][3])
13 {
14     int i;
15     int j;
16     cout << setw(10) << "1" << setw(8) << "2" << setw(9) << "3\n";
17     for (i = 0; i < 3; i++)
18     {
19         cout << setw(7) << "[" << i + 1 << "]"[1] | " << "[" << i + 1;
20         cout << "]"[2] | " << "[" << i + 1 << "]"[3] << endl;
21         cout << setw(14) << "|" << setw(9) << "|" << endl;
22         for (j = 0; j < 3; j++)
23         {
24             switch (j)
25             {
26                 case 0: cout << i + 1 << setw(9) << boardAr[i][j];
27                         cout << setw(4) << "|";
28                         break;
29                 case 1: cout << setw(4) << boardAr[i][j];
30                         cout << setw(5) << "|";
31                         break;
32                 case 2: cout << setw(4) << boardAr[i][j] << endl;
33                         break;
34                 default: cout << "ERROR!\n\n";
35             }
36         }
37         cout << setw(14) << "|" << setw(10) << "|\n";
38         if (i != 2)
39         {
40             cout << setw(32) << "-----\n";
41         }
42     }
43     cout << endl << endl;
44 }
```

```
1  #include "Header.h"
2
3  /*****
4  * GetPlayers
5  * This function prompts the user and gets the input for the players' names.
6  * playerX will always contain the name of the player that is using the X token.
7  * player0 will always contain the name of the player that is using the 0 token.
8  *
9  * INPUTS:
10 *   playerX - Player with token X.
11 *   player0 - Player with token 0.
12 *
13 * No outputs.
14 *****/
15 void GetPlayers(string& playerX, // OUT - player X's name
16                string& player0) // OUT - player 0's name
17 {
18     cout << "Enter the name of player using X: ";
19     getline(cin, playerX);
20
21     cout << "Enter the name of player using 0: ";
22     getline(cin, player0);
23 }
```

```
1  #include "Header.h"
2
3  void GetAndCheckInp(char boardAr[][NUM_COLS],
4                      char token,
5                      string playerX,
6                      string playerO)
7  {
8      int row;
9      int col;
10     bool valid;
11
12     valid = false;
13
14     do
15     {
16         if (token == 'X')
17         {
18             cout << playerX;
19         }
20         else
21         {
22             cout << playerO;
23         }
24
25         cout << "\n's turn! What is your play?: ";
26         cin >> row >> col;
27         row--;
28         col--;
29         if (row > NUM_ROWS - 1 || row < 0)
30         {
31             cout << "Invalid row - Please try again!\n";
32         }
33         else if (col > NUM_COLS - 1 || col < 0)
34         {
35             cout << "Invalid column - Please try again!\n";
36         }
37         else if (!isspace(boardAr[row][col])) // > if( boardAr[row][col] != ' '
38             ' ')
39         {
40             cout << "That spot is taken already - try again!\n";
41         }
42         else
43         {
44             valid = true;
45         }
46     } while (!valid);
47
48     boardAr[row][col] = token;
49     cin.ignore(10000, '\n');
```

49

50 }

```
1  #include "Header.h"
2
3  /*****
4  * SwitchToken
5  * This function switches the active player.
6  * It takes in a parameter representing the current player's token
7  * as a character value (either an X or an O) and returns the opposite.
8  * For example, if this function receives an X it returns an O. If it
9  * receives an O it returns an X.
10 *
11 * INPUTS:
12 * token - Current player's token.
13 *
14 * OUTPUTS:
15 * token - Opposite player's token.
16 *****/
17 char SwitchToken(char token) // IN - current player's token ('X' or 'O')
18 {
19     if (token == 'X')
20     {
21         token = 'O';
22     }
23     else if (token == 'O')
24     {
25         token = 'X';
26     }
27
28     return token;
29 }
```



```
1  #include "Header.h"
2
3  /*****
4  * CheckWin
5  * This function checks to see if either player has run. Once it is
6  * possible for a win condition to exist, this should run after each a
7  * player makes a play.
8  *
9  * INPUTS:
10 * boardAr - Array for the gameboard.
11 * token   - player's turn.
12 *
13 * OUTPUTS:
14 * whoWon   - character representing game winner or tie.
15 *****/
16 char CheckWin(const char boardAr[][NUM_COLS], // IN - tic tac toe board
17               char token)                     // IN - token of who's playing.
18 {
19     bool rowWin;
20     bool colWin;
21     bool diaWin;
22     char whoWon;
23
24     rowWin = (boardAr[0][0] == boardAr[0][1] && boardAr[0][1] == boardAr[0][2] &
25               && boardAr[0][1] != ' ') ||
26             (boardAr[1][0] == boardAr[1][1] && boardAr[1][1] == boardAr[1][2] &
27               && boardAr[1][1] != ' ') ||
28             (boardAr[2][0] == boardAr[2][1] && boardAr[2][1] == boardAr[2][2] &
29               && boardAr[2][1] != ' ');
30
31     colWin = (boardAr[0][0] == boardAr[1][0] && boardAr[1][0] == boardAr[2][0] &
32               && boardAr[1][0] != ' ') ||
33             (boardAr[0][1] == boardAr[1][1] && boardAr[1][1] == boardAr[2][1] &
34               && boardAr[1][1] != ' ') ||
35             (boardAr[0][2] == boardAr[1][2] && boardAr[1][2] == boardAr[2][2] &
36               && boardAr[1][2] != ' ');
37
38     diaWin = (boardAr[0][0] == boardAr[1][1] && boardAr[1][1] == boardAr[2][2] &
39               && boardAr[1][1] != ' ') ||
40             (boardAr[2][0] == boardAr[1][1] && boardAr[1][1] == boardAr[0][2] &
41               && boardAr[1][1] != ' ');
42
43     if (rowWin || colWin || diaWin)
44     {
45         whoWon = token;
46     }
47 }
```

---

```
42     else
43     {
44         whoWon = 'N';
45     }
46
47     return whoWon;
48 }
```

```
1  #include "Header.h"
2
3  void OutputWinner(char whoWon, // IN - represents the winner or value indicating tie.
4                      string playerX, // OUT - player X's name
5                      string playerO) // OUT - player O's name
6  {
7      if (whoWon == 'X')
8      {
9          cout << endl;
10         cout << playerX << " is the winner of the game.";
11         cout << endl;
12     }
13     else if (whoWon == 'O')
14     {
15         cout << endl;
16         cout << playerO << " is the winner of the game.";
17         cout << endl;
18     }
19     else
20     {
21         cout << endl;
22         cout << "No winners! It was a tie.";
23         cout << endl;
24     }
25 }
```

```
1  #include "Header.h"
2
3  /*****
4   * PrintHeaderFile
5   *   This function will output the header information
6   *
7   * PRE-CONDITIONS
8   *   The following parameters need to have a defined value prior to calling
9   *   the function
10  *       asName: The name of the assignment given in the course
11  *       asNum: The number of the assignment given in the course
12  *       studentName: The name of the student writing the code
13  *       classInfo: The course name, date, and time of the class
14  *       asType: Will either output as a lab or an assignment
15  *       studentID: The Identification Number of the student
16  *****/
17 void PrintHeaderFile(ostream& output,    // IN - output datatype.
18                     string asName,      // IN - assignment name
19                     int asNum,          // IN - assignment number
20                     string studentName, // IN - student's name
21                     string classInfo,   // IN - class that is being taken
22                     char asType,        // IN - assignment type
23                     long long studentID) // IN - student ID
24 {
25     output << left;
26     output << "*****\n";
27     output << "*   PROGRAMMED BY : " << studentName << endl;
28     output << "*   " << setw(14) << "STUDENT ID " << ": " << studentID << endl;
29     output << "*   " << setw(14) << "CLASS " << ": " << classInfo << endl;
30     output << "*   ";
31
32     // PROCESSING - This will adjust setws and format appropriately based
33     //               on if this is a lab 'L' or assignment
34
35     if (toupper(asType) == 'L')
36     {
37         output << "LAB #" << setw(9);
38     }
39     else
40     {
41         output << "ASSIGNMENT #" << setw(2);
42     }
43     output << asNum << ": " << asName << endl;
44     output << "*****";
45     output << right << endl;
46
47     return;
```

48 }