

## Activity No. 5.1

### Multidimensional Arrays

Course Code: CPE 007

Program: Computer Engineering

Course Title: Programming Logic and Design

Date Performed: 9/25/2025

Section: CPE11S1

Date Submitted: 9/30/2025

Name(s): Cenar, Marqui Joshua

Instructor: Engr. Jimlord M. Quejado

#### 6. Output

##### 1. CODE:

```
[*] cnar.cpp  cenarr.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5      const int size = 10;
6      int table [size][size];
7
8      for (int r = 0; r < size; r++){
9          for (int c = 0; c < size; c++){
10             cout << (r+1) * (c+1) << "\t";
11
12             if (c == size - 1){
13                 cout << endl;
14             }
15         }
16     }
17
18     return 0;
19 }
20 }
```

##### OUTPUT:

```
C:\Users\TIPQC\Downloads\ci X + v
1   2   3   4   5   6   7   8   9   10
2   4   6   8  10  12  14  16  18  20
3   6   9  12  15  18  21  24  27  30
4   8  12  16  20  24  28  32  36  40
5  10  15  20  25  30  35  40  45  50
6  12  18  24  30  36  42  48  54  60
7  14  21  28  35  42  49  56  63  70
8  16  24  32  40  48  56  64  72  80
9  18  27  36  45  54  63  72  81  90
10 20  30  40  50  60  70  80  90 100
```

```
-----
Process exited after 0.05113 seconds with return value 0
Press any key to continue . . . |
```

2.

CODE:

```
1  #include <iostream>
2  using namespace std;
3
4  char board[3][3] = {
5      {' ', ' ', ' '},
6      {' ', ' ', ' '},
7      {' ', ' ', ' '}
8  };
9
10
11 void printBoard() {
12     cout << endl;
13     for (int i = 0; i < 3; i++) {
14         for (int j = 0; j < 3; j++) {
15             cout << board[i][j];
16             if (j < 2) cout << " | ";
17         }
18         cout << endl;
19         if (i < 2) cout << "---+---+---" << endl;
20     }
21     cout << endl;
22 }
23
24
25 bool winner(char player) {
26
27     for (int i = 0; i < 3; i++) {
28         if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) ||
29             (board[0][i] == player && board[1][i] == player && board[2][i] == player)) {
30             return true;
```

```

30         return true;
31     }
32 }
33
34 if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) ||
35     (board[0][2] == player && board[1][1] == player && board[2][0] == player)) {
36     return true;
37 }
38 return false;
39 }
40
41
42 int main() {
43     int row, col;
44     char player = 'X';
45     int moves = 0;
46
47     cout << "Welcome to Tic-Tac-Toe!" << endl;
48     printBoard();
49
50     while (true) {
51         cout << "Player " << player << ", enter row (1-3) and column (1-3): ";
52         cin >> row >> col;
53
54
55         row--; col--;
56
57
58         if (row < 0 || row > 2 || col < 0 || col > 2 || board[row][col] != ' ') {
59             cout << "Invalid move" << endl;

```

```

59         cout << "Invalid move" << endl;
60         continue;
61     }
62
63
64     board[row][col] = player;
65     moves++;
66     printBoard();
67
68
69     if (winner(player)) {
70         cout << "Player " << player << " is the winner!" << endl;
71         break;
72     }
73
74
75     if (moves == 9) {
76         cout << "It's a draw" << endl;
77         break;
78     }
79
80     player = (player == 'X') ? 'O' : 'X';
81 }
82
83 return 0;
84 }

```

## OUTPUT:

Welcome to Tic-Tac-Toe!

```

| |
+---+
| |
+---+
| |

```

Player X, enter row (1-3) and column (1-3): 1 1

```

X | |
+---+
| |
+---+
| |

```

Player O, enter row (1-3) and column (1-3): 2 1

```

X | |
+---+
O | |
+---+
| |

```

Player X, enter row (1-3) and column (1-3): 2 2

```

X | |
+---+
O | X |
+---+
| |

```

Player O, enter row (1-3) and column (1-3): 3 1

```

X | |
+---+
O | X |
+---+
O | |

```

Player X, enter row (1-3) and column (1-3): 3 3

```

X | |
+---+
O | X |
+---+
O | | X

```

Player X is the winner!

Process finished with exit code 0

X is the winner.

```
Welcome to Tic-Tac-Toe!

| | 
--+---+--
| | 
--+---+--
| | 

Player X, enter row (1-3) and column (1-3): 1 2

| X | 
--+---+--
| | 
--+---+--
| | 

Player 0, enter row (1-3) and column (1-3): 1 1

0 | X | 
--+---+--
| | 
--+---+--
| | 
```

```
Player X, enter row (1-3) and column (1-3): 2 1

0 | X | 
--+---+--
X |  | 
--+---+--
| | 

Player 0, enter row (1-3) and column (1-3): 2 2

0 | X | 
--+---+--
X | 0 | 
--+---+--
| | 

Player X, enter row (1-3) and column (1-3): 3 1

0 | X | 
--+---+--
X | 0 | 
--+---+--
X |  | 
```

```
Player 0, enter row (1-3) and column (1-3): 3 3

0 | X | 
--+---+--
X | 0 | 
--+---+--
X |  | 0

Player 0 is the winner!

Process finished with exit code 0
```

O is the winner.

```
Welcome to Tic-Tac-Toe!
```

```
| |  
--+---+--  
| |  
--+---+--  
| |
```

```
Player X, enter row (1-3) and column (1-3): 1 1
```

```
X | |  
--+---+--  
| |  
--+---+--  
| |
```

```
Player O, enter row (1-3) and column (1-3): 1 2
```

```
X | O |  
--+---+--  
| |  
--+---+--  
| |
```

```
Player X, enter row (1-3) and column (1-3): 1 3
```

```
X | O | X  
--+---+--  
| |  
--+---+--  
| |
```

```
Player O, enter row (1-3) and column (1-3): 2 2
```

```
X | O | X  
--+---+--  
| O |  
--+---+--  
| |
```

```
Player X, enter row (1-3) and column (1-3): 2 1
```

```
X | O | X  
--+---+--  
X | O |  
--+---+--  
| |
```

```
Player O, enter row (1-3) and column (1-3): 3 1
```

```
X | O | X  
--+---+--  
X | O |  
--+---+--  
O | |
```

```
Player X, enter row (1-3) and column (1-3): 3 2
```

```
X | O | X  
--+---+--  
X | O |  
--+---+--  
O | X |
```

```
Player O, enter row (1-3) and column (1-3): 3 3
```

```
X | O | X  
--+---+--  
X | O |  
--+---+--  
O | X | O
```

```
Player X, enter row (1-3) and column (1-3): 2 3
```

```
X | O | X  
--+---+--  
X | O | X  
--+---+--  
O | X | O
```

```
It's a draw
```

```
Process finished with exit code 0
```

**Draw**

## 7. Supplementary Activity

### Analysis for no. 1:

Line 5-6, I initialized a constant size of 10, this means setting the size of the table to 10. Then declared a two dimensional array called "table", this has 10 rows and 10 columns that is enough to store the multiplication table.

On Line 8-10 I created two for loops that goes through every row (represented by "r") and every column (represented by "c") then "(r+1)" and "(c+1)" calculates the multiplication results. I used "\t" so the values line up neatly per row and column.

Line 12-13 is an IF statement to check if the loop is at the last column.

### **Analysis for no. 2:**

Line 4-8 I made a 3x3 array called "board" for the tic-tac-toe grid.

Line 11-22 I used a void function and called it "printBoard" and made two FOR loops to display the tic-tac-toe board.

Line 25-39 I declared a bool function called "winner" to check if X or O has three in a row to be declared as the winner, The first FOR loop and IF statements are used to check all rows and all columns, the second IF statement is used to check the diagonal parts of the board. Then when a player wins, it returns.

Line 42-81 where the main program is coded. I initialized "row" and "col" to store where the players want to place their move, It will always start with 'X', and "int moves" counts how many moves are played by each player (X or O). Then prints out a message saying "Welcome to Tic-Tac-Toe!". The WHILE statement is used to keep the game going until a player is declared the winner or results in a draw. An IF statement is used to check if every move the user's inputs is valid. Another IF statement is used to check for each move if a player has 3 in a row. Another IF statement to check if all 9 places are filled and the game ends in a draw. The "player = (player == 'X') ? 'O' : 'X';" is used to switch turns of the players and goes back and forth from X to O.

## **8. Conclusion**

In this activity I learned how to create a multiplication table and a working tic-tac-toe program by using loops and arrays. I think there are more areas I need to improve to have a better program than this one. I could have made a shorter and more efficient code than the one I made but I lack the knowledge to do so. By accomplishing this activity I realized that I need to practice more on using FOR loops and arrays.