Hands-on Activity 5.1	
Multidimensional Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic & Design	**Date Performed: 9/30/2025**
Section: CPE11S1	**Date Submitted: 9/30/2025**
Name(s) Ralph Angelov F. Braganza	**Instructor: Engr. Jimlord M. Quejado**

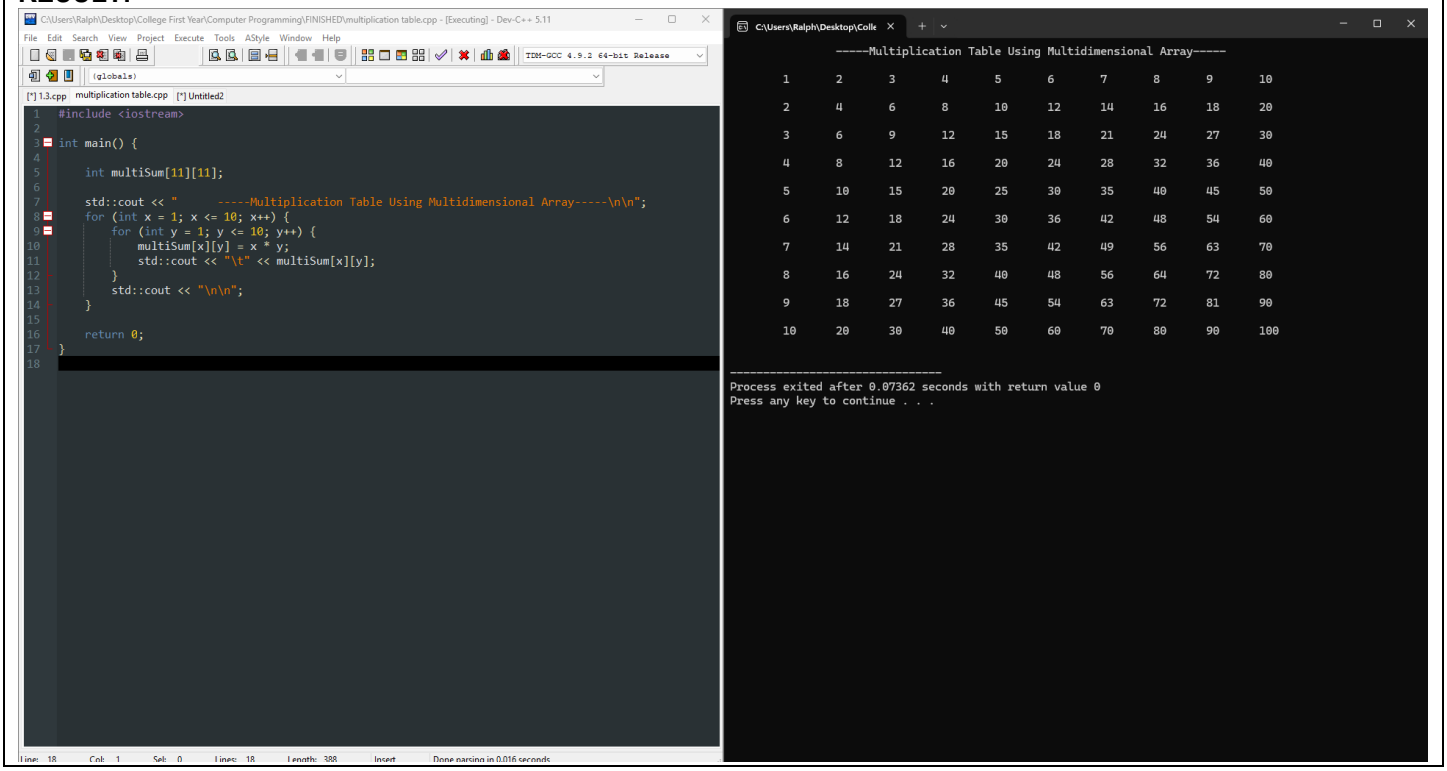
Output

1. Write a program that creates a multiplication table using multidimensional array.

CODE:

```
#include <iostream>

int main() {

    int multiSum[11][11];

            std::cout << "                -----Multiplication Table Using Multidimensional Array-----\n\n";
    for (int x = 1; x <= 10; x++) {
        for (int y = 1; y <= 10; y++) {
            multiSum[x][y] = x * y;
            std::cout << "\t" << multiSum[x][y];
        }
        std::cout << "\n\n";
    }

    return 0;
}
```

RESULT:



2. Write a program that creates a board with a tic-tac-toe moves.

CODE:

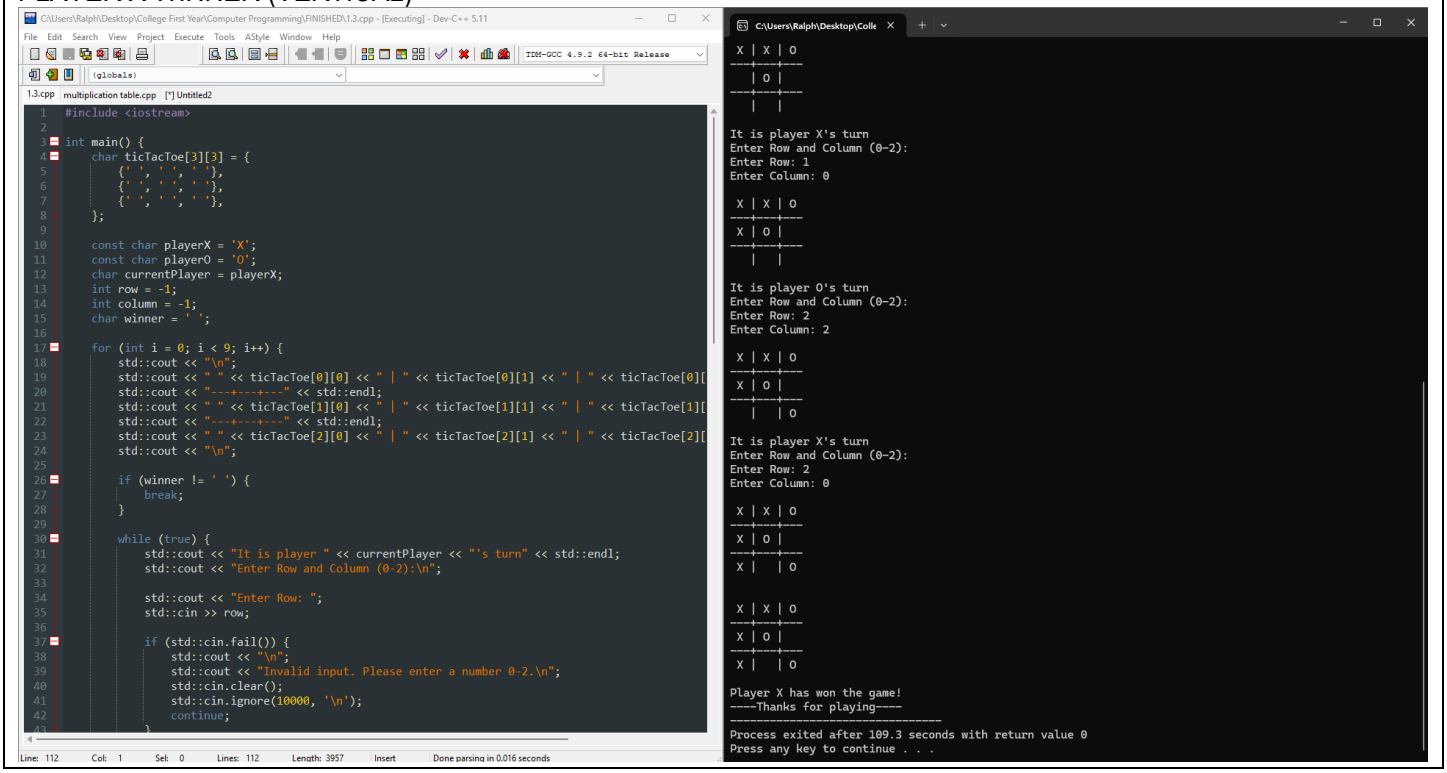
```
#include <iostream>

int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };

    const char playerX = 'X';
    const char playerO = 'O';
    char currentPlayer = playerX;
    int row = -1;
    int column = -1;
    char winner = ' ';

    for (int i = 0; i < 9; i++) {
        std::cout << "\n";
        std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][2] << std::endl;
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][2] << std::endl;
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][2] << std::endl;
        std::cout << "\n";

        if (winner != ' ') {
            break;
        }

        while (true) {
            std::cout << "It is player " << currentPlayer << "'s turn" << std::endl;
            std::cout << "Enter Row and Column (0-2):\n";

            std::cout << "Enter Row: ";
            std::cin >> row;

            if (std::cin.fail()) {
                std::cout << "\n";
                std::cout << "Invalid input. Please enter a number 0-2.\n";
                std::cin.clear();
                std::cin.ignore(10000, '\n');
                continue;
            }

            std::cout << "Enter Column: ";
            std::cin >> column;

            if (std::cin.fail()) {
```

```
            std::cout << "\n";
            std::cout << "Invalid input. Please enter a number 0-2.\n";
            std::cin.clear();
            std::cin.ignore(10000, '\n');
            continue;
        }

        if (row < 0 || row > 2 || column < 0 || column > 2) {
            std::cout << "\nThat is an invalid move! Please try again.\n";
        } else if (ticTacToe[row][column] != ' ') {
            std::cout << "\nTile already occupied! Please try again.\n";
        } else {
            break;
        }
    }

    ticTacToe[row][column] = currentPlayer;

    for (int r = 0; r < 3; r++) {
        if (ticTacToe[r][0] != ' ' && ticTacToe[r][0] == ticTacToe[r][1] && ticTacToe[r][1] == ticTacToe[r][2]) {
            winner = ticTacToe[r][0];
            break;
        }
    }

    for (int c = 0; c < 3; c++) {
        if (ticTacToe[0][c] != ' ' && ticTacToe[0][c] == ticTacToe[1][c] && ticTacToe[1][c] == ticTacToe[2][c]) {
            winner = ticTacToe[0][c];
            break;
        }
    }

    if (ticTacToe[0][0] != ' ' && ticTacToe[0][0] == ticTacToe[1][1] && ticTacToe[1][1] == ticTacToe[2][2]) {
        winner = ticTacToe[0][0];
    } else if (ticTacToe[0][2] != ' ' && ticTacToe[0][2] == ticTacToe[1][1] && ticTacToe[1][1] == ticTacToe[2][0]) {
        winner = ticTacToe[0][2];
    }

    if (currentPlayer == playerX) {
        currentPlayer = playerO;
    } else {
        currentPlayer = playerX;
    }
}

std::cout << "\n";
std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][2] << std::endl;
std::cout << "---+---+---" << std::endl;
std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][2] << std::endl;
std::cout << "---+---+---" << std::endl;
std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][2] << std::endl;
std::cout << "\n";
```

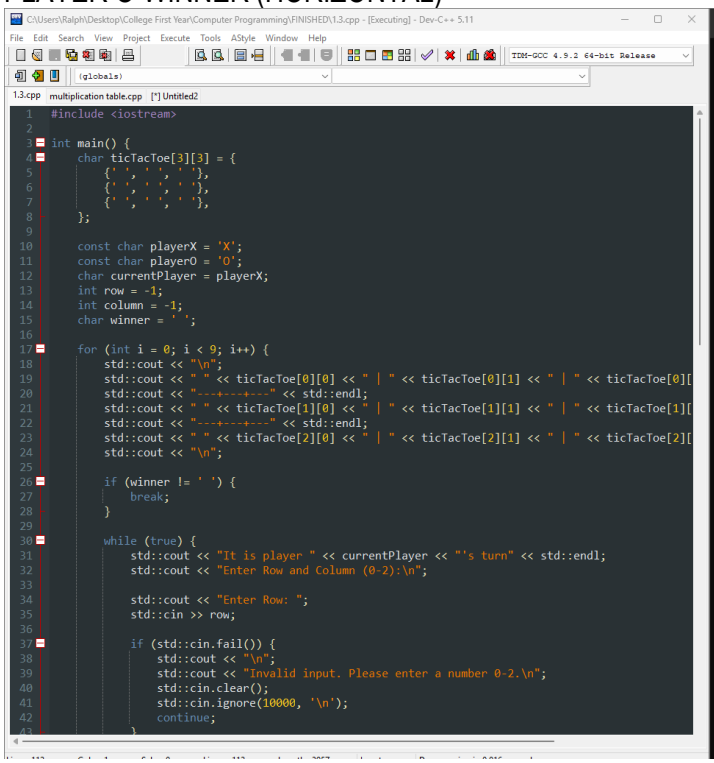
```
    if (winner != ' ') {
        std::cout << "Player " << winner << " has won the game!\n";
        std::cout << "----Thanks for playing----";
    } else {
        std::cout << "          Its a Tie!\n";
        std::cout << "----Thanks for playing----" << std::endl;
    }

    return 0;
}
```

RESULT:

PLAYER X WINNER (VERTICAL)



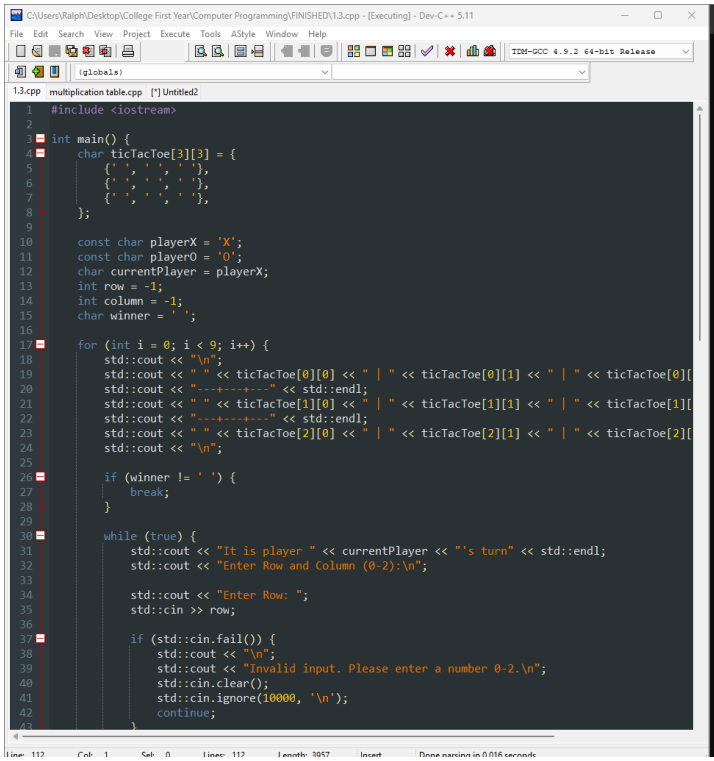
PLAYER O WINNER (HORIZONTAL)



```
#include <iostream>

int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };

    const char playerX = 'X';
    const char playerO = 'O';
    char currentPlayer = playerX;
    int row = -1;
    int column = -1;
    char winner = ' ';

    for (int i = 0; i < 9; i++) {
        std::cout << "\n";
        std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][
        std::cout << "\n";

        if (winner != ' ') {
            break;
        }

        while (true) {
            std::cout << "It is player " << currentPlayer << "'s turn" << std::endl;
            std::cout << "Enter Row and Column (0-2):\n";

            std::cout << "Enter Row: ";
            std::cin >> row;

            if (std::cin.fail()) {
                std::cout << "\n";
                std::cout << "Invalid input. Please enter a number 0-2.\n";
                std::cin.clear();
                std::cin.ignore(10000, '\n');
                continue;
            }
```

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

It is player O's turn
Enter Row and Column (0-2):
Enter Row: 0
Enter Column: 1

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

It is player X's turn
Enter Row and Column (0-2):
Enter Row: 1
Enter Column: 1

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

It is player O's turn
Enter Row and Column (0-2):
Enter Row: 0
Enter Column: 2

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


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

Player O has won the game!
----Thanks for playing----
--------------------------------
Process exited after 103.5 seconds with return value 0
Press any key to continue . . .
```

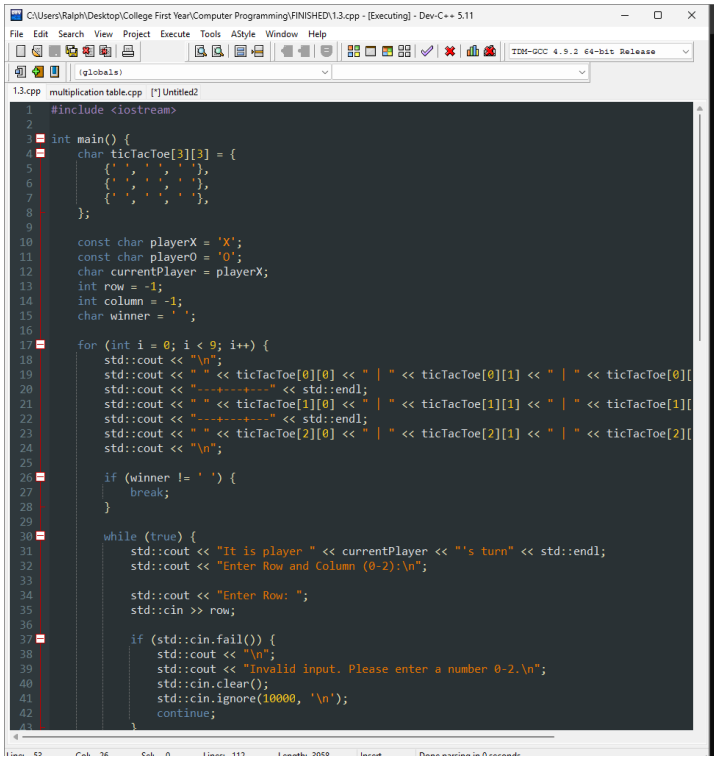
DIAGONAL WINNER



```
#include <iostream>

int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };

    const char playerX = 'X';
    const char playerO = 'O';
    char currentPlayer = playerX;
    int row = -1;
    int column = -1;
    char winner = ' ';

    for (int i = 0; i < 9; i++) {
        std::cout << "\n";
        std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][
        std::cout << "\n";

        if (winner != ' ') {
            break;
        }

        while (true) {
            std::cout << "It is player " << currentPlayer << "'s turn" << std::endl;
            std::cout << "Enter Row and Column (0-2):\n";

            std::cout << "Enter Row: ";
            std::cin >> row;

            if (std::cin.fail()) {
                std::cout << "\n";
                std::cout << "Invalid input. Please enter a number 0-2.\n";
                std::cin.clear();
                std::cin.ignore(10000, '\n');
                continue;
            }
```

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

It is player X's turn
Enter Row and Column (0-2):
Enter Row: 1
Enter Column: 1

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

It is player O's turn
Enter Row and Column (0-2):
Enter Row: 0
Enter Column: 2

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

It is player X's turn
Enter Row and Column (0-2):
Enter Row: 2
Enter Column: 2

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


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

Player X has won the game!
----Thanks for playing----
--------------------------------
Process exited after 48.59 seconds with return value 0
Press any key to continue . . .
```

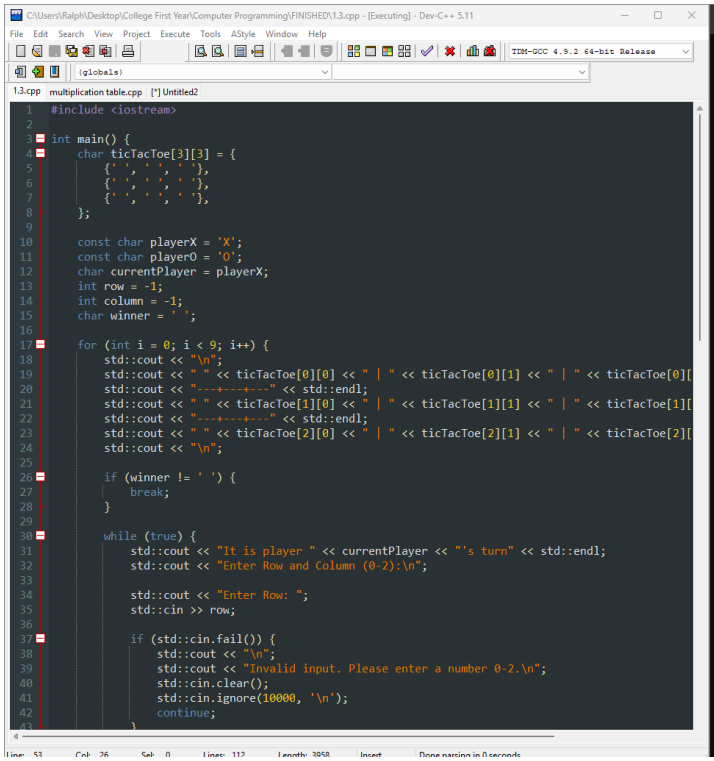
TIED GAME



```
#include <iostream>

int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };

    const char playerX = 'X';
    const char playerO = 'O';
    char currentPlayer = playerX;
    int row = -1;
    int column = -1;
    char winner = ' ';

    for (int i = 0; i < 9; i++) {
        std::cout << "\n";
        std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][
        std::cout << "\n";

        if (winner != ' ') {
            break;
        }

        while (true) {
            std::cout << "It is player " << currentPlayer << "'s turn" << std::endl;
            std::cout << "Enter Row and Column (0-2):\n";

            std::cout << "Enter Row: ";
            std::cin >> row;

            if (std::cin.fail()) {
                std::cout << "\n";
                std::cout << "Invalid input. Please enter a number 0-2.\n";
                std::cin.clear();
                std::cin.ignore(10000, '\n');
                continue;
            }
```

```
It is player O's turn
Enter Row and Column (0-2):
Enter Row: 0
Enter Column: 1

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

It is player X's turn
Enter Row and Column (0-2):
Enter Row: 1
Enter Column: 0

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

It is player O's turn
Enter Row and Column (0-2):
Enter Row: 1
Enter Column: 2

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

It is player X's turn
Enter Row and Column (0-2):
Enter Row: 2
Enter Column: 1

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

       Its a Tie!
----Thanks for playing----

--------------------------------
Process exited after 80.28 seconds with return value 0
Press any key to continue . . .
```

INVALID MOVE AND OCCUPIED TILE



```
#include <iostream>

int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };

    const char playerX = 'X';
    const char playerO = 'O';
    char currentPlayer = playerX;
    int row = -1;
    int column = -1;
    char winner = ' ';

    for (int i = 0; i < 9; i++) {
        std::cout << "\n";
        std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][
        std::cout << "\n";

        if (winner != ' ') {
            break;
        }

        while (true) {
            std::cout << "It is player " << currentPlayer << "'s turn" << std::endl;
            std::cout << "Enter Row and Column (0-2):\n";

            std::cout << "Enter Row: ";
            std::cin >> row;

            if (std::cin.fail()) {
                std::cout << "\n";
                std::cout << "Invalid input. Please enter a number 0-2.\n";
                std::cin.clear();
                std::cin.ignore(10000, '\n');
                continue;
            }
```

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

It is player X's turn
Enter Row and Column (0-2):
Enter Row: d

Invalid input. Please enter a number 0-2.
It is player X's turn
Enter Row and Column (0-2):
Enter Row: 2
Enter Column: fff

Invalid input. Please enter a number 0-2.
It is player X's turn
Enter Row and Column (0-2):
Enter Row: 44
Enter Column: 44

That is an invalid move! Please try again.
It is player X's turn
Enter Row and Column (0-2):
Enter Row: 0
Enter Column: 0

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

It is player O's turn
Enter Row and Column (0-2):
Enter Row: 0
Enter Column: 0

Tile already occupied! Please try again.
It is player O's turn
Enter Row and Column (0-2):
Enter Row:
```

Supplementary Activity

1. Write a program that creates a multiplication table using multidimensional array.

CODE:

```
#include <iostream>

int main() {
```

```
    int multiSum[11][11];

            std::cout << "                -----Multiplication Table Using Multidimensional Array-----\n\n";
    for (int x = 1; x <= 10; x++) {
        for (int y = 1; y <= 10; y++) {
            multiSum[x][y] = x * y;
            std::cout << "\t" << multiSum[x][y];
        }
        std::cout << "\n\n";
    }

    return 0;
}
```

ANALYSIS:

First we must put our header <iostream> for the basic function of inputs/outputs (cin,cout) and after that we have our two-dimensional array named multiSum with 11 rows and 11 columns, this will store the results of the multiplication table. Next, it prints the title of the activity that we are doing and here is where our for-loops are located, the outer for-loop uses a variable x that will go from 1 to 10 (this will represent the first number in the multiplication table). The inner for-loop is where we use variable y that also goes from 1 to 10 and it will represent the second number in the multiplication table. Inside our inner for-loop is where we will get the product of the x and y being calculated and stored in our multiSum array at the position of [x][y]. Example if we have 2 as our x and 5 as our y the value of that will be 10 being stored at multiSum array (multiSum[2][5]). After that the code will start printing the calculated products followed by "\t" for spacing between the numbers to make it neater. When the inner for-loop is done with its cycle for a single value of x, the code prints a newline character which is the "\n\n", which moves it to the next line and adding an extra blank line to be able to make the rows for the multiplication table. The outer for-loop then increments x and the process repeats until both loops have completed, resulting to our 10x10 multiplication table.

2. Write a program that creates a board with a tic-tac-toe moves.

CODE:

```
#include <iostream>

int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };

    const char playerX = 'X';
    const char playerO = 'O';
    char currentPlayer = playerX;
    int row = -1;
    int column = -1;
    char winner = ' ';

    for (int i = 0; i < 9; i++) {
        std::cout << "\n";
        std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][2] << std::endl;
        std::cout << "---+---+---" << std::endl;
        std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][2] << std::endl;
        std::cout << "---+---+---" << std::endl;
```

```
    std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][2] << std::endl;
    std::cout << "\n";

    if (winner != ' ') {
        break;
    }

    while (true) {
        std::cout << "It is player " << currentPlayer << "'s turn" << std::endl;
        std::cout << "Enter Row and Column (0-2):\n";

        std::cout << "Enter Row: ";
        std::cin >> row;

        if (std::cin.fail()) {
            std::cout << "\n";
            std::cout << "Invalid input. Please enter a number 0-2.\n";
            std::cin.clear();
            std::cin.ignore(10000, '\n');
            continue;
        }

        std::cout << "Enter Column: ";
        std::cin >> column;

        if (std::cin.fail()) {
            std::cout << "\n";
            std::cout << "Invalid input. Please enter a number 0-2.\n";
            std::cin.clear();
            std::cin.ignore(10000, '\n');
            continue;
        }

        if (row < 0 || row > 2 || column < 0 || column > 2) {
            std::cout << "\nThat is an invalid move! Please try again.\n";
        } else if (ticTacToe[row][column] != ' ') {
            std::cout << "\nTile already occupied! Please try again.\n";
        } else {
            break;
        }
    }

    ticTacToe[row][column] = currentPlayer;

    for (int r = 0; r < 3; r++) {
        if (ticTacToe[r][0] != ' ' && ticTacToe[r][0] == ticTacToe[r][1] && ticTacToe[r][1] == ticTacToe[r][2]) {
            winner = ticTacToe[r][0];
            break;
        }
    }

    for (int c = 0; c < 3; c++) {
```

```
            if (ticTacToe[0][c] != ' ' && ticTacToe[0][c] == ticTacToe[1][c] && ticTacToe[1][c] == ticTacToe[2][c]) {
                winner = ticTacToe[0][c];
                break;
            }
        }

        if (ticTacToe[0][0] != ' ' && ticTacToe[0][0] == ticTacToe[1][1] && ticTacToe[1][1] == ticTacToe[2][2]) {
            winner = ticTacToe[0][0];
        } else if (ticTacToe[0][2] != ' ' && ticTacToe[0][2] == ticTacToe[1][1] && ticTacToe[1][1] == ticTacToe[2][0]) {
            winner = ticTacToe[0][2];
        }

        if (currentPlayer == playerX) {
            currentPlayer = playerO;
        } else {
            currentPlayer = playerX;
        }
    }

    std::cout << "\n";
    std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][2] << std::endl;
    std::cout << "---+---+---" << std::endl;
    std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][2] << std::endl;
    std::cout << "---+---+---" << std::endl;
    std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][2] << std::endl;
    std::cout << "\n";

    if (winner != ' ') {
        std::cout << "Player " << winner << " has won the game!\n";
        std::cout << "----Thanks for playing----";
    } else {
        std::cout << "          Its a Tie!\n";
        std::cout << "----Thanks for playing----" << std::endl;
    }

    return 0;
}
```

ANALYSIS:

Again, we have to put our header <iostream> for the basic function of inputs/outputs (cin,cout) and this time we have an multidimensional array declared as a char named ticTacToe that represents a 3x3 gameboard to input our X's and O's with each element initialized to a space ' '.

```
int main() {
    char ticTacToe[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '},
    };
```

Now, here will be assigning our variables where our to constant character variables is playerX and playerO these will be

our two players who will be playing the game then currentPlayer variable is initialized to playerX so 'X' goes first every time the game starts when the code is ran. Then here we have our input variables row and column so the player can place their X's and O wherever they want and they're initialized to -1 which is an invalid index, to ensure they don't accidentally match a valid position before the user's input. And of course, the winner variable is initialized to a space to indicate that no one has won yet in the game.

This is also our gameboard where I just put designs for the outline of the gameboard, spaces to make it look nice and printed the array spaces as well. winner !=1 ' ' will immediately stop the game if it finds a winner already.

```
const char playerX = 'X';
const char playerO = 'O';
char currentPlayer = playerX;
int row = -1;
int column = -1;
char winner = ' ';

for (int i = 0; i < 9; i++) {
    std::cout << "\n";
    std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][2] << std::endl;
    std::cout << "---+---+---" << std::endl;
    std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][2] << std::endl;
    std::cout << "---+---+---" << std::endl;
    std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][2] << std::endl;
    std::cout << "\n";

    if (winner != ' ') {
        break;
    }
```

The next line of code is where it will get and validate the user input this is where we use a while-loop so it will keep asking the user to input a valid move till the valid move is entered. This will display who's turn it is, and it will ask for the row and column for it to be displayed and move to the next player and the rest of the line of code is the conditions for it to be able to move on.

```
if (std::cin.fail()) {
    std::cout << "\n";
    std::cout << "Invalid input. Please enter a number 0-2.\n";
    std::cin.clear();
    std::cin.ignore(10000, '\n');
    continue;
}
```

This block of code validates the user's input. std::cin.fail() becomes true if the user enters something that is not a number. If this happens, an error message is printed, std::cin.clear() resets the error flags on std::cin, and std::cin.ignore(...) discards the invalid input from the buffer, preventing an infinite loop (where it prints a bunch of times and never stops). Then the continue statement skips to the next iteration of the while loop.

So here in the first statement it will check whether the numbers that the players entered are outside the range of 0-2 and if they did so, it will print out "invalid move" and they have to try again. The else if checks if the tile is already occupied or it isn't a space, it will also print out an error message and this will keep on going until it eventually breaks (meaning the player finally inputted a valid value of 0-2). This line ticTacToe[row][column] = currentPlayer places the current player's symbol ('X' or 'O') into the selected square on the board, marking their move.

```
    if (row < 0 || row > 2 || column < 0 || column > 2) {
        std::cout << "\nThat is an invalid move! Please try again.\n";
    } else if (ticTacToe[row][column] != ' ') {
        std::cout << "\nTile already occupied! Please try again.\n";
    } else {
        break;
    }
}

ticTacToe[row][column] = currentPlayer;
```

The next line of code will check if there are any winners, this loop checks all three rows for a win. It checks if the three characters in a row are the same and not an empty space. If a win is found, winner is set to the winning player's symbol, and the loop breaks. Same goes for the next loop where it will check if any of the columns are the same and not an empty space meaning there is a winner. Then the next line of code, if and else if block checks the two diagonal lines for a winner.

```
for (int r = 0; r < 3; r++) {
    if (ticTacToe[r][0] != ' ' && ticTacToe[r][0] == ticTacToe[r][1] && ticTacToe[r][1] == ticTacToe[r][2]) {
        winner = ticTacToe[r][0];
        break;
    }
}

for (int c = 0; c < 3; c++) {
    if (ticTacToe[0][c] != ' ' && ticTacToe[0][c] == ticTacToe[1][c] && ticTacToe[1][c] == ticTacToe[2][c]) {
        winner = ticTacToe[0][c];
        break;
    }
}

if (ticTacToe[0][0] != ' ' && ticTacToe[0][0] == ticTacToe[1][1] && ticTacToe[1][1] == ticTacToe[2][2]) {
    winner = ticTacToe[0][0];
} else if (ticTacToe[0][2] != ' ' && ticTacToe[0][2] == ticTacToe[1][1] && ticTacToe[1][1] == ticTacToe[2][0]) {
    winner = ticTacToe[0][2];
}
```

That was all checking but if no one has won yet this if-else statement switches the current player. If the currentPlayer was 'X', it is changed to 'O', and if it was 'O', it is changed to 'X', preparing for the next turn.

```
if (currentPlayer == playerX) {
    currentPlayer = playerO;
} else {
    currentPlayer = playerX;
}
```

If the winner != ‘ ‘ is true and the conditions are met then we have a winner and it will print out the player that has won (‘X’ or ‘O’) and print out "----Thanks for playing----". However, if the both players used all of their 9 moves and none of the conditions were met for winning, it will print out "It’s a Tie!" and tell both players "----Thanks for playing----".

```
std::cout << "\n";
std::cout << " " << ticTacToe[0][0] << " | " << ticTacToe[0][1] << " | " << ticTacToe[0][2] << std::endl;
std::cout << "---+---+---" << std::endl;
std::cout << " " << ticTacToe[1][0] << " | " << ticTacToe[1][1] << " | " << ticTacToe[1][2] << std::endl;
std::cout << "---+---+---" << std::endl;
std::cout << " " << ticTacToe[2][0] << " | " << ticTacToe[2][1] << " | " << ticTacToe[2][2] << std::endl;
std::cout << "\n";

if (winner != ' ') {
    std::cout << "Player " << winner << " has won the game!\n";
    std::cout << "----Thanks for playing----";
} else {
    std::cout << "  Its a Tie!\n";
    std::cout << "----Thanks for playing----" << std::endl;
}
```

Conclusion

This activity has taken me a long time to do, but it has taught me to be more mindful about curly braces and to remember to name my variables correctly, so I won't have a hard time in the long run of my code. I had so many errors, and it took me so long to figure out what to do, but I managed to pull through. I learned how to make both of the players have turns, I’ve learned how to make conditions if the tiles were occupied or if there was an invalid move, and I’ve also done a little bit

of designing in this activity. I didn't do well because, and I'm going to be honest, I used a lot of YouTube for this, but I'm proud of myself for having the ability to do this activity and put my own things into this code. In this activity I've also implemented my learnings about operators, which are the OR (||), Comparison (==), and AND (&&) operators, and I've gotten a better understanding of how multidimensional arrays work becauseI really had to think outside of the box to be able to put X's and O's into the box/tiles, and I will be honest, it was a headache, but it is something I have to go through so I don't suffer in the future the more I move on to coding. I'd say I did an okay job in this activity; it isn't the greatest work, but it's a start for me to become a better C++ coder so I can build that foundation for myself to be that better person in the future. Overall, I just hope if I ever go through a wall, I'll be able to overcome it and keep trying even if I fall multiples times.