

**SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING (SMME)**

**NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SECTOR H-12, ISLAMABAD**

**PROGRAM: BACHELOR OF AEROSPACE ENGINEERING**

**SEMESTER: 1ST**

**SESSION: FALL 2023**

**SECTION: AE-01**

**COURSE TITLE: FUNDAMENTALS OF PROGRAMMING (FOP)**

**COURSE CODE: CS-109**

***FOP LAB PROJECT – TIC TAC TOE GAME***

***END SEMESTER PROJECT***

**NAME: ATEEQ-UR-REHMAN, MUHAMMAD AWAIS ALI, SYED SHAH UMER**

**CMS IDs: 465502, 461465 , 463322**

**DATE: 29TH DEC. 2023**

**INTRODUCTION:**

Tic tac toe is a simple and fun game for two players, X and O, who take turns marking the spaces in a 3x3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. If all nine spaces are filled and neither player has three in a row, the game is a tie. Tic tac toe is also known as noughts and crosses or X-O game in some parts of the world. It is a classic game that can be played with paper and pencil, board and pieces, or online and on mobile devices. It is a zero-sum game, which means that if both players play optimally, the game will always end in a tie.

Tic tac toe can be a good project for learning and practicing computer programming languages, such as C++. By building a tic tac toe game using C++, various concepts and skills can be learnt and applied accordingly, such as:

* Arrays and matrices: A two-dimensional array of characters can be used to represent the 3x3 grid of the game board and store the symbols of the players (X or O) in the array elements.
* Functions and parameters: The code can be divided into modular and reusable functions that perform specific tasks, such as displaying the board, getting the player input, updating the board, and checking the game status. Parameters can also be passed to your functions to communicate data between them.
* Loops and conditions: Loops can be used to iterate over the array elements and conditions to check the validity of the input and the outcome of the game (win, draw, or continue).
* Input and output: The standard input and output streams (cin and cout) can be used to interact with the user and display the game information on the console screen.

Moreover, building a tic tac toe game using C++ can help improve your logical thinking and problem-solving skills, as well as creativity and fun. The game can also be extended by adding more features, such as a graphical user interface, a computer opponent, or a scoring system.

**EXPLANATION OF THE CODE:**

In the creation of the game (code given below), the already learnt topics of arrays and loops and sequence and logic have been utilized to make an efficient and optimized game. This is done through taking a move from the player, and the computer makes a move in accordance with the current setup of the game, checking for winning and blocking moves. The games has implied the use of modular approach i.e. the game is broken into multiple functions, each with its special function that are called and ran together to complete the gameplay.

Below is a brief description of the modules used in the game.

1. Screen\_update: The console is cleared after every move to display an updated screen, enhancing the gameplay experience

This module is run until there is a winner or the game is drawed

1. Player\_move: This module allows the player i.e. the user to input the move (a number from 1 to 9)
2. Computer\_move: This module makes the computer to make a move based on the updated screen (after the player’s move), with priorities starting from a winning move to a blocking move to capturing the center place and making a random move as the last priority
3. Player\_wins: This module checks for all possible conditions that the player could win in
4. Computer\_wins: Likewise, this module checks for all possible conditions that the computer could win in
5. is\_Draw: This module checks if the game is drawed or not

The Player’s move symbol is initialized as ‘X’ while the Computer’s move symbol is initialized as ‘O’.

**CODE:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

char game\_char[9] = { '1', '2', '3', '4', '5', '6', '7', '8', '9' }; //The game board is initialised as an array of base elements as numbers

int num; //The input to be taken from the user/player for the move

char Player = 'X'; //The player symbol is initialised as 'X'

char Computer = 'O'; //The player symbol is initialised as 'O'

//All functions used in the game

bool is\_Draw();

void Screen\_update();

void Player\_move();

bool Player\_wins();

bool Computer\_wins();

void Computer\_move();

void Screen\_update() //Updates the screen after every move

{

system("CLS");

cout << endl;

cout << endl;

cout << "\t\t\t\tTic Tac Toe Game - AE-01 \n" << endl << endl;

cout << "\t\t\t\tAteeq ur Rehman - 465502 \n" << endl;

cout << "\t\t\t\tMuhammad Awais Ali - 461465 \n" << endl;

cout << "\t\t\t\tSyed Shah Umer - 463322 \n" << endl << endl;

cout << "\t\t\t\tPlayer v/s Computer \n" << endl << endl;

cout << "\t\t\t\tPlayer: X\n";

cout << "\t\t\t\tComputer: O\n";

cout << endl << endl;

cout << "\t\t\t\t" << game\_char[0] << " | " << game\_char[1] << " | " << game\_char[2] << "\n";

cout << "\t\t\t\t-------------" << endl;

cout << "\t\t\t\t" << game\_char[3] << " | " << game\_char[4] << " | " << game\_char[5] << "\n";

cout << "\t\t\t\t-------------" << endl;

cout << "\t\t\t\t" << game\_char[6] << " | " << game\_char[7] << " | " << game\_char[8] << "\n\n";

}

void Player\_move() //Takes the input for the move from the player

{

cout << "\t\t\t\tPlayer - Enter number: ";

cin >> num;

switch (num)

{

case 1:

game\_char[0] = Player;

break;

case 2:

game\_char[1] = Player;

break;

case 3:

game\_char[2] = Player;

break;

case 4:

game\_char[3] = Player;

break;

case 5:

game\_char[4] = Player;

break;

case 6:

game\_char[5] = Player;

break;

case 7:

game\_char[6] = Player;

break;

case 8:

game\_char[7] = Player;

break;

case 9:

game\_char[8] = Player;

break;

default:

cout << "\t\t\t\t\nPlease enter a number from 1 to 9!\n\n" << endl;

Player\_move();

break;

}

}

void Computer\_move() //Makes the computer to marks its symbol 'O' based on conditions

{

srand(time(0)); //Initiates seed of random fuction

for (int i = 0; i < 9; ++i) //Checks if there is a winning move

{

if (game\_char[i] != 'X' && game\_char[i] != 'O')

{

char originalValue = game\_char[i];

game\_char[i] = Computer;

if (Computer\_wins())

{

return;

}

game\_char[i] = originalValue;

}

}

for (int i = 0; i < 9; ++i) //Checks if there is a blocking move

{

if (game\_char[i] != 'X' && game\_char[i] != 'O')

{

char originalValue = game\_char[i];

game\_char[i] = Player;

if (Player\_wins())

{

game\_char[i] = Computer;

return;

}

game\_char[i] = originalValue;

}

}

// Fork: Create an opportunity for the computer to win in the next move

// Block Fork: Block the user from creating a fork

// Center: Take the center if available

if (game\_char[4] != 'X' && game\_char[4] != 'O')

{

game\_char[4] = Computer;

return;

}

for (;;)

{

int i = rand() % 10;

while (i == 9)

{

i = rand() % 10;

}

if (game\_char[i] != 'X' && game\_char[i] != 'O')

{

game\_char[i] = Computer;

break;

}

}

}

bool is\_Draw() //Checks if the game is drawed

{

for (int i = 0; i <= 8; i++)

{

if (game\_char[i] != 'X' && game\_char[i] != 'O')

{

//If there is an empty space, the game is not a draw

return false;

}

else

{

continue;

}

}

//If all spaces are filled and no one has won

return true;

}

bool Player\_wins() //Checks if the player has won the game

{

if (game\_char[0] == 'X' && game\_char[1] == 'X' && game\_char[2] == 'X')

return true;

else if (game\_char[3] == 'X' && game\_char[4] == 'X' && game\_char[5] == 'X')

return true;

else if (game\_char[6] == 'X' && game\_char[7] == 'X' && game\_char[8] == 'X')

return true;

else if (game\_char[0] == 'X' && game\_char[3] == 'X' && game\_char[6] == 'X')

return true;

else if (game\_char[1] == 'X' && game\_char[4] == 'X' && game\_char[7] == 'X')

return true;

else if (game\_char[2] == 'X' && game\_char[5] == 'X' && game\_char[8] == 'X')

return true;

else if (game\_char[0] == 'X' && game\_char[4] == 'X' && game\_char[8] == 'X')

return true;

else if (game\_char[2] == 'X' && game\_char[4] == 'X' && game\_char[6] == 'X')

return true;

else

return false;

}

bool Computer\_wins() //Checks if the computer has won the game

{

if (game\_char[0] == 'O' && game\_char[1] == 'O' && game\_char[2] == 'O')

return true;

else if (game\_char[3] == 'O' && game\_char[4] == 'O' && game\_char[5] == 'O')

return true;

else if (game\_char[6] == 'O' && game\_char[7] == 'O' && game\_char[8] == 'O')

return true;

else if (game\_char[0] == 'O' && game\_char[3] == 'O' && game\_char[6] == 'O')

return true;

else if (game\_char[1] == 'O' && game\_char[4] == 'O' && game\_char[7] == 'O')

return true;

else if (game\_char[2] == 'O' && game\_char[5] == 'O' && game\_char[8] == 'O')

return true;

else if (game\_char[0] == 'O' && game\_char[4] == 'O' && game\_char[8] == 'O')

return true;

else if (game\_char[2] == 'O' && game\_char[4] == 'O' && game\_char[6] == 'O')

return true;

else

return false;

}

int main() //Main calling function of all other functions

{

do

{

Screen\_update(); //Initializes the game board before and after every turn

Player\_move(); //Takes input from the player

Screen\_update(); //Updates the screen after the move

if (Player\_wins()) //Checks whether the player has won

{

cout << "\t\t\t\tThe Player Wins!" << endl;

break;

}

if (is\_Draw()) //Checks whether the game is drawed

{

cout << "\t\t\t\tThe game is a draw!" << endl;

break;

}

Computer\_move(); //The computer makes amove

Screen\_update(); //Updates the screen after the move

if (is\_Draw()) //Checks if the game is drawed

{

cout << "\t\t\t\tThe game is a draw!" << endl;

break;

}

if (Computer\_wins()) //Checks whether the computer has won

{

cout << "\t\t\t\tThe Computer Wins!" << endl;

break;

}

cout << "\n\n\n";

} while (true);

return 0;

}

**OUTPUT:**

****

**CONCLUSIONS:**

Making a game such as tic tac toe in C++ is a good way to practice and improve programming skills, especially in areas such as logic, data structures, algorithms, and object-oriented design. Undoubtedly, there are many benefits that one can derive from this project, some of which are:

* You can learn how to use arrays, vectors, strings, and other data types to store and manipulate the game state and the user input.
* You can learn how to use loops, conditional statements, switch cases, and functions to implement the game logic and the user interface.
* You can learn how to use random number generation, recursion, and artificial intelligence techniques to create a computer opponent that can play against the user or itself.
* You can learn how to use classes, inheritance, polymorphism, and abstract data types to organize your code and make it more reusable and extensible.
* You can learn how to use pointers, references, dynamic memory allocation, and exception handling to manage memory and handle errors.
* You can learn how to use templates, generic programming, and operator overloading to make your code more flexible and expressive.
* You can learn how to use standard library containers, algorithms, and iterators to simplify and optimize your code.
* You can learn how to use input/output streams, files, and serialization to save and load the game state and the user preferences.
* You can learn how to use graphics, sound, and networking libraries to enhance the game’s appearance, interactivity, and multiplayer functionality.

Additionally, to make an efficient and optimized game of tic tac toe in C++, the following steps can be implemented:

* Use of appropriate data structures and algorithms to store and access the game state and the user input. For example, you can use a two-dimensional array, a vector of vectors, or a string to represent the game board, and use a simple mapping function to convert the user input to the corresponding index.
* Use of modular and well-structured code to implement the game logic and the user interface. For example, you can use separate functions to check the game over conditions, validate the user input, update the game board, display the game board, and handle the game loop. You can also use constants, enums, and macros to define the game symbols, the board size, and the game modes.
* Use of object-oriented design principles (OOP) to organize your code and make it more reusable and extensible. For example, you can use a class to encapsulate the game state and the game logic, and use inheritance and polymorphism to create different types of players, such as human, computer, or network players. You can also use abstract data types and interfaces to define the common behaviours and attributes of the players, and use virtual functions and dynamic binding to implement them in different ways.
* Use of advanced C++ features to make your code more flexible and expressive. For example, you can use templates and generic programming to make your code work with different data types and parameters, and use operator overloading to define custom operations for your classes and objects. You can also use smart pointers, containers, and algorithms to manage memory and perform common tasks more easily and efficiently.
* Use of external libraries and frameworks to enhance the game’s appearance, interactivity, and multiplayer functionality. For example, you can use graphics libraries such as SDL, SFML, or OpenGL to create a graphical user interface, sound libraries such as SDL\_mixer, SFML\_audio, or OpenAL to add sound effects and music, and networking libraries such as SDL\_net, SFML\_network, or Boost.Asio to enable online multiplayer mode.

All in all, this project serves as a hands-on learning experience, allowing one to excel their programming skills and have a deeper insight into game development, which is a field on its own, as well as explore the applications of artificial intelligence in practical contexts.