**1. Introduction : TIC TAC TIO GAME**

Welcome to the Tic-Tac-Toe game program, where timeless strategy meets modern convenience! Tic-Tac-Toe is a classic game that has delighted players of all ages for generations. The rules are simple yet the gameplay offers endless possibilities for strategic thinking and tactical maneuvers.

In this digital adaptation of the beloved game, you'll find all the familiar elements that make Tic-Tac-Toe a favorite pastime. The 3x3 grid awaits your moves, with each player taking turns to place their symbol, 'X' or 'O', on an empty space. The objective is clear: be the first to get three of your symbols in a row, either horizontally, vertically, or diagonally.

But this program offers more than just the traditional gameplay. You have the option to challenge a friend in a thrilling head-to-head match, where every move counts and victory hinges on your ability to outwit your opponent. Alternatively, test your skills against a computer-controlled AI opponent, ranging from beginner-friendly to expert-level difficulty. Can you outsmart the AI and emerge victorious?

The user-friendly interface makes playing Tic-Tac-Toe a breeze. Simply click or tap to place your symbol, track the progress of the game with visual indicators, and enjoy smooth gameplay that enhances your overall gaming experience. Whether you're a seasoned Tic-Tac-Toe strategist or a newcomer looking to learn the ropes, this program is designed to provide hours of entertainment and mental stimulation.

**2. Requirement Specification for Project**

**2.1 Software**

1. Install Visual Studio Code:

Download and install Visual Studio Code from the official website: https://code.visualstudio.com/.

Follow the installation instructions for your operating system (Windows, macOS, Linux).

1. Install C/C++ Extension:

Open Visual Studio Code.

Go to the Extensions view by clicking on the Extensions icon in the Activity Bar on the side of the window or by pressing Ctrl + Shift+ X.

Search for "C/C++" in the Extensions Marketplace.

Install the "C/C++" extension provided by Microsoft.

1. Open Your Project Folder:

Create a new folder on your computer to contain your Tic-Tac-Toe game code.

Open this folder in Visual Studio Code. You can do this by clicking on "File" > "Open Folder" in the menu bar and selecting your project folder.

Create a New C File:

Inside your project folder in VS Code, create a new C file for your Tic-Tac-Toe game code. You can do this by right-clicking on the folder in the Explorer view, selecting "New File," and giving it a name with a .c extension (e.g., tictactoe.c).

1. Write Your C Code:

Open the newly created C file (tictactoe.c) in Visual Studio Code's editor.

Write or copy-paste your Tic-Tac-Toe game code into this file.

Configure Build Task:

1. Run your code
   * + - **System :**

Edition : Windows 11 Pro

Version : 23H2

Installed on : 1/‎13/‎2024

OS build : 22631.3374

Experience Windows Feature Experience Pack 1000.22688.1000.0

* 1. **HARDWARE:**

Device name : PCUSER

Processor :12th Gen Intel(R) Core(TM) i5-12400 2.50 GHz

6 cores (12 threads) Max turbo Frequency 4.4GHz

Installed RAM :16.0 GB (15.7 GB usable)(DDR4)

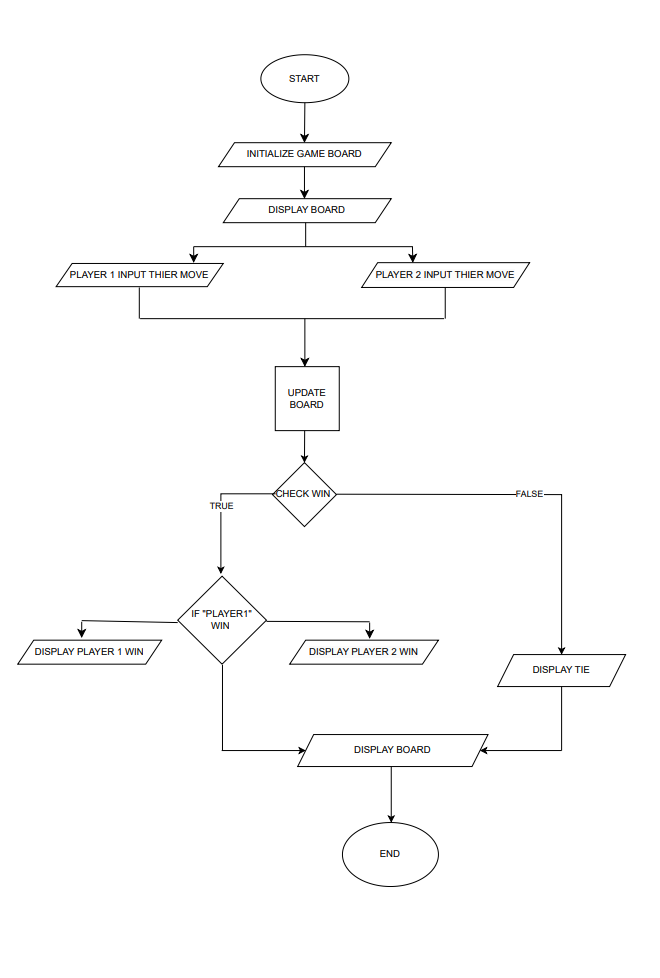
Storage :512 GB (SSD)

Device ID :31C13483-5953-447B-BF64-8D87D0E679AB

Product ID :00331-10000-00001-AA804

System type :64-bit operating system, x64-based processor

1. **FLOWCHART :**

****

**4. Algorithm :**

1.start.

2. Initialize the game board as an array board[] with 10 elements, labeled from 0 to 9 (ignore index 0).

-Set board[1] to board [9] as '1' to '9' representing the cells of the grid.

3.Set player as 1 (representing X) and status as -1 (indicating an ongoing game). initial game board using print Board() function.

4. Display the initial game board using print Board() function.

5. While status is -1 (indicating the game is ongoing):

a. Determine the current player's marker:

If player is 1, set mark as 'X'; otherwise, set mark as '0'.

b. Prompt the current player to enter a number (input) for their move.

i. If input is less than 1 or greater than 9, display "Invalid input" and ask for input again.

ii. If the cell corresponding to input is already marked ('X' or '0'), display "Cell already occupied" and ask for input again.

c. Update the game board by marking the cell with the current player's (mark) based on the input.

d. Display the updated game board using printBoard() function.

e. Check for a winner:

Call the checkWin() function to determine the result:

i. If a player has won, set status as 1 and break the loop.

ii. If the game is a draw (no winner and board is full), set status as 0 and break the loop.

f. Switch players:

If player is 1, set player as 2;

otherwise, set player as 1.

6. If status is 1 (indicating a winner):

Display "Player X is the Winner" or "Player O is the Winner" based on the current player.

7. Else if status is 0 (indicating a draw):

Display "Match is draw".

8. End.

**5. Main code :**

#include <stdio.h>

#include <conio.h>

void printBoard();

int checkWin();

void system();

char board[]={'0','1','2','3','4','5','6','7','8','9'};

void main(){

int player=1,input,status=-1;

printBoard();

while (status==-1)

{

player=(player%2==0) ? 2 : 1;

char mark=(player==1) ? 'X' :'O';

printf("Please enter Number For Player %d\n",player);

scanf("%d",&input);

if(input<1 || input>9){

printf("invalid input");

}

board[input]=mark;

printBoard();

int result=checkWin();

if(result==1){

printf("Player %d is the Winner",player);

return;

}else if(result==0){

printf("Match is draw");

return;

}

player++;

}

}

void printBoard(){

system("cls");

printf("\n\n");

printf("=== TIC TAC TOE ===\n\n");

printf(" | | \n");

printf(" %c | %c | %c \n",board[1],board[2],board[3]);

printf("||\_\n");

printf(" | | \n");

printf(" %c | %c | %c \n",board[4],board[5],board[6]);

printf("||\_\n");

printf(" | | \n");

printf(" %c | %c | %c \n",board[7],board[8],board[9]);

printf(" | | \n");

printf("\n\n");

}

int checkWin(){

if(board[1]==board[2] && board[2]==board[3]){

return 1;

}

if(board[1]==board[4] && board[4]==board[7]){

return 1;

}

if(board[7]==board[8] && board[8]==board[9]){

return 1;

}

if(board[3]==board[6] && board[6]==board[9]){

return 1;

}

if(board[1]==board[5] && board[5]==board[9]){

return 1;

}

if(board[3]==board[5] && board[5]==board[7]){

return 1;

}

if(board[2]==board[5] && board[5]==board[8]){

return 1;

}

if(board[4]==board[5] && board[5]==board[6]){

return 1;

}

int count=0;

for (int i = 1; i <=9; i++)

{

if(board[i]=='X' || board[i]=='O'){

count++;

}

}

if(count==9){

return 0;

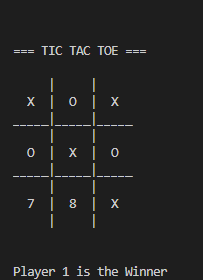
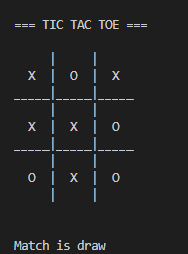
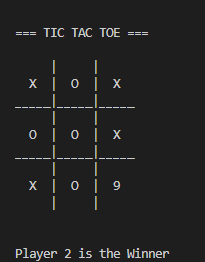
}

    return -1;

}

**6. OUTPUT :**





In this output “Player 1” In this output “Player 2” In this output match is



is winner is winner “draw”

**7. CONCLUSION :**

1. Game Initialization:

The program starts by initializing the game board, which is represented by an array with numeric placeholders for each cell (1 to 9).

1. Player Interaction:

Two players take turns to play the game, with Player 1 using 'X' marks and Player 2 using 'O' marks.

The program prompts each player to enter a number (1-9) corresponding to the cell where they want to place their mark.

Input validation ensures that players enter valid numbers within the allowed range and that the chosen cell is not already occupied.

1. Game Logic:

After each move, the program checks for win conditions and a draw.

Win conditions are checked by comparing the marks in rows, columns, and diagonals to see if there are three identical marks in a sequence.

If a win condition is met, the program declares the respective player as the winner and ends the game.

If all cells are filled without a winner, the game is declared a draw.

1. User Interface:

The program provides a simple command-line interface for players to interact with the game.

The game board is displayed after each move, showing the current state of the game with marks placed by players.

1. Conclusion Messages:

When the game ends (either due to a win or a draw), the program displays a conclusion message indicating the outcome.

If a player wins, the program prints "Player X wins!" or "Player O wins!" depending on which player achieved the win condition.

If the game ends in a draw, the program prints "It's a draw!" to signify that neither player won.

1. Termination:

After printing the conclusion message, the program terminates, ending the game session.

1. Overall Functionality:

The C code for the Tic-Tac-Toe game demonstrates fundamental programming concepts such as arrays, loops, conditional statements, input/output handling, and basic game logic.

It provides a playable implementation of the Tic-Tac-Toe game with two human players taking turns to play until a win or a draw occurs.

While the code lacks graphical user interface (GUI) elements and advanced features like AI opponents or networking capabilities, it serves as a foundation for understanding and building more complex games or applications in C.

In conclusion, the provided C code successfully implements a functional Tic-Tac-Toe game, showcasing core programming principles and offering an interactive gameplay experience within a command-line environment