TIC TAC TOE

COURSE CODE : INTE 11223

COURSE TITLE : Programming Concepts

SUBMISSION DATE : 24.7.2025

GROUP MEMBERS : DIKKUMBURA D.K.C.P (IM/2023/007)

AFHAM M.A.M (IM/2023/005)

BANDARA M.G.S.S (IM/2023/126)

SIRIWARDHANA K.A.S.K (IM/2023/125)

Abstract

This report outlines the development of Tic Tac Toe, a console-based two-player and player-vs-computer strategy game written in C++. The game allows players to select between two modes, Player vs Player **(PvP)** and Player vs Computer **(PvC)**. Designed with an interactive and engaging user interface using the console, the game emphasizes logic design, input validation, and turn-based mechanics.

Key features include smart AI for the computer, real-time input handling, replay and exit options, and visual board representation using ASCII characters. The project enhanced our understanding of control structures, functions, input validation, and turn-based game mechanics in C++.

Table of Contents

Introduction

The Tic Tac Toe game was developed to strengthen our understanding of C++ programming through the design and implementation of a complete mini-game. Inspired by the classic pencil-and-paper version, this project introduces two game modes, basic AI logic, and polished user interaction all within a console window.

This project demonstrates our practical knowledge of arrays, conditional logic, loops, functions, and basic AI decision-making using C++.

Libraries Used

- <iostream>

  For standard input/output
- <iomanip> –

  For formatted output
- <thread> & <chrono>

  To add timed delays and smooth animations
- <cstdlib>

  For generating random numbers and exit control
- <limits>

  To handle user input validation

Key Functions

- printBoard()

  Displays the current game board.
- showBoard()

  Shows reference numbers for each board cell.
- checkWin()

  Checks if a player has won.
- getBestMove()

  AI logic to block or win.
- playPvp()

  Main loop for PvP mode.
- playPvc()

  Main loop for Player vs Computer mode.
- main()

  Game menu and navigation.

User Guide

Launching the Game

- Run the compiled .exe file.
- The main menu will appear with 3 options:
    1. Player vs Player
    2. Player vs Computer
    3. Exit

Game Modes

1. Player vs Player (PvP)

- Two users enter their names.
- Takes turns choosing positions (1–9).
- Game ends with win/draw and offers replay or exit.

2. Player vs Computer (PvC)

- You play as ‘X’, computer is ‘O’.
- Computer uses logic to block or win.
- Clear feedback and AI move display included.

Controls

- Input: Numbers 1–9 (cell positions).
- Text-based prompts guide the player through input.
- Errors are handled and user is asked to re-enter.

Future Enhancements

1. Add Graphics
   Use colors or create a version with buttons and mouse clicks (GUI).
2. Improve Computer Player
   Make the computer play smarter so it doesn't lose easily.
3. Online Multiplayer
   Let two people play from different computers using the internet.
4. Add Scoreboard
   Show player scores or wins after each game.
5. Mobile Version
   Create a version that works on phones or tablets.

Challenges and Solutions

During the development of our Tic Tac Toe game, we faced several challenges. These helped us learn and improve our programming skills.

1. Input Validation

At first, the program would crash when users entered letters or left the input empty instead of typing numbers. We had to add proper checks to make sure the player enters only numbers between 1 and 9, and nothing else.

2. Same Spot Selection

Players sometimes chose a cell that was already taken. We had to write extra conditions to detect this and show a message asking the player to try again.

3. Player Name Handling

When we asked players to enter their names, the input sometimes didn't work correctly (especially if the name had spaces). We solved this by using getline() and cin >> ws to allow full names properly.

4. AI Logic in PvC Mode

Creating a computer player that could block or win was tricky. At first, the computer made random moves, which was too easy. So, we wrote logic to check if the computer could win in the next move or stop the player from winning.

5. Replay and Menu Navigation

After the game ends, we wanted to let the player play again or go back to the main menu. Managing the flow of choices without repeating or exiting the game by mistake required good planning using loops and flags.

6. Debugging and Testing

Sometimes the game didn’t switch turns correctly or didn't detect a win properly. We had to carefully test each part, especially the win-checking conditions, to make sure all 8 possible winning combinations worked.

Conclusion

The **Tic Tac Toe** game project served as an ideal platform to explore core C++ concepts in a fun and practical way. It deepened our understanding of game loops, conditionals, arrays, user input handling, and AI basics. Through this project, we also developed skills in problem-solving, debugging, and presenting interactive content via console UI.

Working collaboratively helped us learn software design as a team and highlighted the importance of testing and iterative development.

Reference

- GeeksforGeeks. (n.d.). Tic Tac Toe Game in C++
- Microsoft Learn Docs: C++ chrono and thread library
- Stack Overflow Discussions on getline() and stoi() input parsing

Appendix

```
#include <iostream>
#include <iomanip>
#include <thread>     // For sleep
#include <chrono>     // For milliseconds
#include <cstdlib>
#include <limits>
#include <windows.h>

using namespace std;

#include <windows.h>
#include <iostream>
using namespace std;

const int SCREEN_WIDTH = 80;   // Adjust based on your console
const int SCREEN_HEIGHT = 25;  // Adjust based on your console

void CursorLocation(int x, int y) {
   COORD coord;
   coord.X = x;
   coord.Y = y;
   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),
coord);
}
```

```
void drawFullScreenBorder() {
    char borderChar = 219;

    // Draw top and bottom borders
    for (int x = 0; x < SCREEN_WIDTH; x++) {
        CursorLocation(x, 0); cout << borderChar;
        CursorLocation(x, SCREEN_HEIGHT - 1); cout << borderChar;
    }

    // Draw left and right borders
    for (int y = 1; y < SCREEN_HEIGHT - 1; y++) {
        CursorLocation(0, y); cout << borderChar;
        CursorLocation(SCREEN_WIDTH - 1, y); cout << borderChar;
    }
}



void printBoard(char spaces[]){

    cout << "\t\t\t+----------------------+\n";
    cout << "\t\t\t|       TIC TAC TOE     |\n";
    cout << "\t\t\t+----------------------+\n";
    cout << "\t\t\t|       |       |       |\n";
    cout << "\t\t\t|   " << spaces[0] << "   |   " << spaces[1] << "   |   " <<
spaces[2] << "    |\n";
```

```
    cout << "\t\t\t|_______|_______|_______|\n";
    cout << "\t\t\t|       |       |       |\n";
    cout << "\t\t\t|   " << spaces[3] << "   |   " << spaces[4] << "   |   " << spaces[5] << "   |\n";
    cout << "\t\t\t|_______|_______|_______|\n";
    cout << "\t\t\t|       |       |       |\n";
    cout << "\t\t\t|   " << spaces[6] << "   |   " << spaces[7] << "   |   " << spaces[8] << "   |\n";
    cout << "\t\t\t|_______|_______|_______|\n";
}

void showBoard(){

    cout << "\t\t\t+-----------------------+\n";
    cout << "\t\t\t|       TIC TAC TOE     |\n";
    cout << "\t\t\t+-----------------------+\n";
    cout << "\t\t\t|       |       |       |\n";
    cout << "\t\t\t|   1   |   2   |   3   |\n";
    cout << "\t\t\t|_______|_______|_______|\n";
    cout << "\t\t\t|       |       |       |\n";
    cout << "\t\t\t|   4   |   5   |   6   |\n";
    cout << "\t\t\t|_______|_______|_______|\n";
    cout << "\t\t\t|       |       |       |\n";
    cout << "\t\t\t|   7   |   8   |   9   |\n";
    cout << "\t\t\t|_______|_______|_______|\n";
```

```
}


bool checkWin(char spaces[],char currentPlayer){
   return (
      (spaces[0]==currentPlayer && spaces[1]==currentPlayer &&
spaces[2]==currentPlayer) ||
      (spaces[3]==currentPlayer && spaces[4]==currentPlayer &&
spaces[5]==currentPlayer) ||
      (spaces[6]==currentPlayer && spaces[7]==currentPlayer &&
spaces[8]==currentPlayer) ||
      (spaces[0]==currentPlayer && spaces[3]==currentPlayer &&
spaces[6]==currentPlayer) ||
      (spaces[1]==currentPlayer && spaces[4]==currentPlayer &&
spaces[7]==currentPlayer) ||
      (spaces[2]==currentPlayer && spaces[5]==currentPlayer &&
spaces[8]==currentPlayer) ||
      (spaces[0]==currentPlayer && spaces[4]==currentPlayer &&
spaces[8]==currentPlayer) ||
      (spaces[2]==currentPlayer && spaces[4]==currentPlayer &&
spaces[6]==currentPlayer)
   );
}

int getBestMove(char spaces[],char ai, char player){

   //check if AI can Win
```

```
for (int i=0; i<9;i++){
    if (spaces[i]==' '){
        spaces[i]= ai;
        if (checkWin(spaces,ai)){
            spaces[i]=' ';
            return i;
        }
        spaces[i]=' ';
    }
}
//check if AI needs to block  players win

for (int i=0;i<9;i++){
    if (spaces[i]==' '){
        spaces[i]=player;
        if(checkWin(spaces, player)){
            spaces[i]=' ';
            return i;
        }
        spaces[i]=' ';
    }
}
//choose random available space
for(int i=0;i<9;i++){
    if (spaces[i]==' ') return i;
}
```

```
    return -1;  //should not happen
}

void playPvp(){
        char spaces[9]= {' ',' ',' ',' ',' ',' ',' ',' ',' '};
        char currentPlayer = 'X';
        int moveCount = 0;
        bool gameOver = false;
        string playerXname, playerOname;
        while(true){
            cout<<"\n";
            system("CLS");
            drawFullScreenBorder();      // Draw full screen border
            CursorLocation(5, 2);        // Print inside the border

            cout<<"\t\t\t";
            for(int i=0;i<3;i++){
                this_thread::sleep_for(chrono::milliseconds(400));
                 cout<<".."<<flush;
            }
            this_thread::sleep_for(chrono::milliseconds(400));
            drawFullScreenBorder();      // Draw full screen border
            CursorLocation(5, 2);        // Print inside the border
            cout<<" PvP Mode ....."<<flush;

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

```
    cout<< "\t\t   Enter name for Player X : ";
    cin>>ws;
    getline(cin,playerXname);

if (!playerXname.empty()){
    break;
}else{
    cout<<"\t\t\t Name Cannot be Empty. Please Enter Again. ";
}
}


cout<<endl;

while(true){
    cout<<"\t\t   Enter name for Player O : ";
    cin>>ws;
    getline(cin,playerOname);

    if (!playerOname.empty()){
    break;
}else{
    cout<<"\t\t\t Name Cannot be Empty. Please Enter Again. ";
}
}
```

```
    for (int i=0;i<9;i++){
       spaces[i]=' ';

    }
       moveCount = 0;
       gameOver = false;
       currentPlayer = 'X';
```

```
cout<<endl;
system("CLS");
drawFullScreenBorder();      // Draw full screen border
CursorLocation(5, 2);        // Print inside the border
cout<<"\t\t\t  "<<playerXname<<" vs "<<playerOname;
for(int i=0;i<5;i++){
   this_thread::sleep_for(chrono::milliseconds(400));
cout<<"."<<flush;
}
cout<<"\n\n";
drawFullScreenBorder();      // Draw full screen border
CursorLocation(5, 2);        // Print inside the border
cout<<"\t\t This shows How the Game Board \n";
cout<<"\t\t Loading ";
```

```
for(int i=0;i<5;i++){
    this_thread::sleep_for(chrono::milliseconds(400));
cout<<"."<<flush;
}

cout<<"."<<flush;
cout<<"\n\n\n";
showBoard();
this_thread::sleep_for(chrono::seconds(3));
    while(!gameOver){
       system("CLS");
       drawFullScreenBorder();      // Draw full screen border
       CursorLocation(5, 2);        // Print inside the border
       cout<<'\n'<<'\n';
       cout<<"\t\t\t  "<<playerXname<<" vs "<<playerOname;
       cout<<"\n\n\n";
       printBoard(spaces);

//int move;
   cout<<endl;
   if(currentPlayer=='X'){
       cout<<"\n";
       cout<<"\t\t"<<playerXname<<" (X) Enter your move (1-9) :";
   }
   else{
       cout<<"\n";
```

```
            cout<<"\t\t"<<playerOname<<" (O) Enter your move (1-9) :";
        }


        string input;
        getline(cin, input);

        if (input.empty()) {
        cout << "\n\t\t Input cannot be empty! Please enter a number (1-9).\n";
        this_thread::sleep_for(chrono::seconds(2));
        continue;
    }

        bool validNumber = true;
        for (char c : input) {
        if (!isdigit(c)) {
        validNumber = false;
        break;
    }
}

        if (!validNumber) {
        cout << "\n\t\t Invalid input! Please enter digits only (1-9).\n";
        this_thread::sleep_for(chrono::seconds(2));
        continue;
```

```
}

        int move = stoi(input) - 1;

        if (move < 0 || move > 8) {
        cout<<"\n";
        cout << "\n\t\t Move out of range! Enter a number between 1 and 9.\n";
        this_thread::sleep_for(chrono::seconds(2));
        continue;

        } else if(spaces[move]!=' '){
            cout<<"\n";
            cout<<"\t\t That Spot is taken. TRY AGAIN!";
            this_thread::sleep_for(chrono::seconds(2));  // Optional, to let user see
the message
            continue;

}


      if (spaces[move]==' '){
        spaces[move]=currentPlayer;
        moveCount++;
```

```
 // here, check for win


if(checkWin(spaces,currentPlayer)){
 system("CLS");
 drawFullScreenBorder();      // Draw full screen border
 CursorLocation(5, 2);        // Print inside the border
 printBoard(spaces);
if (currentPlayer=='X'){
     cout<<"\n\n\n";
     cout<<"\t\t\t"<<playerXname<<"(X) WINS !";
}
else{
   cout<<"\n\n\n";
   cout<<"\t\t\t"<<playerOname<<"(O) WINS !";
}
//cout<<"\t\t\t Player "<< currentPlayer<<" WINS ! \n";
 gameOver= true;
 break;
}


 if (moveCount==9){
 system("CLS");
 drawFullScreenBorder();      // Draw full screen border
 CursorLocation(5, 2);        // Print inside the border
 printBoard(spaces);
 cout<<"\n";
```

```
            cout<<"\t\t\t It's a draw! "<<'\n';
            gameOver=true;
        }
         //switch Player

        currentPlayer= (currentPlayer=='X')? 'O':'X';

        }

    }

}

void playPvc(){
            char spaces[9]= {' ',' ',' ',' ',' ',' ',' ',' ',' '};
            char currentPlayer = 'X';
            int moveCount = 0;
            bool gameOver = false;
            char aiPlayer = 'O';
            char humanPlayer = 'X';

        cout<<endl;
        system("CLS");
        cout<<"\t\t\t";
              drawFullScreenBorder();      // Draw full screen border
```

```
    CursorLocation(5, 2);        // Print inside the border

    for(int i=0;i<3;i++){
    this_thread::sleep_for(chrono::milliseconds(400));
    cout<<".."<<flush;
  }
  this_thread::sleep_for(chrono::milliseconds(400));
  drawFullScreenBorder();      // Draw full screen border
  CursorLocation(5, 2);        // Print inside the border

  cout<<" PvC Mode ....."<<flush;
  cout<<"\n\n";

cout<<"\t\t This shows How the Game Board \n";
cout<<"\t\t Loading ";
for(int i=0;i<5;i++){
  this_thread::sleep_for(chrono::milliseconds(400));
  cout<<"."<<flush;
}


cout<<"\n\n\n";
showBoard();
this_thread::sleep_for(chrono::seconds(3));

  while (!gameOver)
```

```
        {
            system("CLS");
            drawFullScreenBorder();      // Draw full screen border
            CursorLocation(5, 2);        // Print inside the border

            cout<<'\n'<<'\n';
            printBoard(spaces);

        int move;

        if (currentPlayer==humanPlayer){
            //Human Turn
            cout<<"\n";
            cout<<"\t\t\t Your Move (1-9): ";
            cin>>move;

            move-=1;
            if (cin.fail()) {
            cin.clear(); // Clear the error flag
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Discard invalid
input
            cout << "\n\t\t\t Invalid input! Please enter a number (1-9).\n";
            this_thread::sleep_for(chrono::seconds(2));
            continue; // Go back and ask again
        }
            else if(
```

```
        move>9 || move<0 ){
        cout<<"\n";
        cout<<"\t\t\t Move Not Valid.Please Enter a number (1-9)  \n";
        this_thread::sleep_for(chrono::seconds(2));
        continue;
    }
    else if(spaces[move]!= ' '){
        cout<<"\n";
        cout<<"\t\t\t That spot is Taken, TRY AGAIN ! "<<endl;
        this_thread::sleep_for(chrono::seconds(2));
        continue;
    }

}
else{
    // Smart AI turn with "Thinking..."
    cout<<"\n";
    cout << "\t\t\t Computer is thinking";
    this_thread::sleep_for(chrono::milliseconds(400));
    cout << "." << flush;
    this_thread::sleep_for(chrono::milliseconds(400));
    cout << "." << flush;
    this_thread::sleep_for(chrono::milliseconds(400));
    cout << "." << flush;
    this_thread::sleep_for(chrono::milliseconds(300));
    cout << endl;
```

```
    move = getBestMove(spaces, aiPlayer, humanPlayer);
    system("CLS");
    cout<<"\n";
    cout << "\t\t\t Computer chooses " << (move + 1) << endl;

}

    spaces[move]=currentPlayer;
    moveCount++;

    if (checkWin(spaces,currentPlayer)){
        system("CLS");
        drawFullScreenBorder();      // Draw full screen border
        CursorLocation(5, 2);        // Print inside the border
        cout<<"\n";

        printBoard(spaces);
        if(currentPlayer==humanPlayer){
            cout<<"\n\n";
            cout<<"\t\t\t You Win! \n";
        }

        else {
        cout<<"\n\n";
        cout<<"\t\t\t Computer Wins! \n";
```

```
            }

            gameOver = true;
            break;
        }
        if(moveCount==9){
            system("CLS");
            drawFullScreenBorder();      // Draw full screen border
            CursorLocation(5, 2);        // Print inside the border
            printBoard(spaces);
            cout<<endl;
            cout<<"\t\t\t It's a Draw! \n";
            gameOver=true;
        }
        if (currentPlayer=='X'){
            currentPlayer='O';

        }else {
            currentPlayer='X';
        }
    }
}

int main(){

    char spaces[9]= {' ',' ',' ',' ',' ',' ',' ',' ',' '};
```

```
int menuChoice;
main_menu:
  while(true){

  system("CLS");
  drawFullScreenBorder();      // Draw full screen border
  CursorLocation(5, 2);        // Print inside the border

  int mode;
  cout<<"\n\n\n";
  cout << "\t\t ================================\n";
  cout << "\t\t       TIC TAC TOE GAME      \n";
  cout << "\t\t ================================\n";
  cout<<"\n\n\n";
  cout<<"\t\t   Welcome to Tic Tac Toe ! \n\n";
  cout<<"\t\t   Choose Game Mode \n\n";
  cout<<"\t\t   1. Player vs Player (PvP)\n";
  cout<<"\t\t   2. Player vs Computer (Random) \n";
  cout<<"\t\t   3. Exit Game \n";
  cout<<"\t\t   Enter Your Choice : ";
  cin>>mode;



if (cin.fail() || mode < 1 || mode > 3) {
  cin.clear();                  // Clear error state
```

```
    cin.ignore(1000, '\n');          // Ignore leftover input
    cout << "\n\t\t Invalid input! Please enter 1, 2, or 3.\n";
    this_thread::sleep_for(chrono::seconds(2));
    continue; // Ask again
}

    else if (mode==1){
        bool playAgain=true;
        while (playAgain){
            playPvp();
            this_thread::sleep_for(chrono::seconds(3));
           // printBoard(spaces);
             system("CLS");

            while(true){
            int menuChoice;
            cout<<"\n\n";
            drawFullScreenBorder();      // Draw full screen border
            CursorLocation(5, 2);        // Print inside the border

            cout<<"\t\t\t Game OVER!! \n\n";
            cout<<"\t\t\t Do you want to \n\n";
            cout<<"\t\t\t 1.Play the game again \n";
            cout<<"\t\t\t 2.Go to Main Menu \n";
            cout<<"\t\t\t 3.Exit Game \n\n";
            cout<<"\t\t\t Enter your choice : ";
```

```
cin >> menuChoice;
system("CLS");

 if (menuChoice == 1) break;
   else if (menuChoice == 2) {playAgain=false;
   goto main_menu;
   break;}
   else if (menuChoice==3)
 {
   cout<<"\t\t Exiting Game.... GOOD BYE! ";
   exit(0);
 }
  else{
   if (cin.fail() || menuChoice < 1 || menuChoice > 3) {
   cin.clear();
   cin.ignore(1000, '\n');
   drawFullScreenBorder();      // Draw full screen border
   CursorLocation(5, 2);        // Print inside the border
   cout << "\n\t\t Invalid input! Please enter 1, 2, or 3.\n";
   this_thread::sleep_for(chrono::seconds(2));
   system("CLS");
   continue;
  }
}
```

```
    }
    }
}

else if (mode==2){
    bool playAgain=true;
    while (playAgain){
        playPvc();
        //printBoard(spaces);
        this_thread::sleep_for(chrono::seconds(3));
        system("CLS");
        while(true){

        int menuChoice;

        drawFullScreenBorder();      // Draw full screen border
        CursorLocation(5, 2);        // Print inside the border
        cout<<"\t\t\t Game OVER!! \n\n";
        cout<<"\t\t\t Do you want to \n\n";
        cout<<"\t\t\t 1.Play the game again \n";
        cout<<"\t\t\t 2.Go to Main Menu \n";
        cout<<"\t\t\t 3.Exit Game \n\n";
        cout<<"\t\t\t Enter your choice : ";
```

```
cin >> menuChoice;
system("CLS");

if (menuChoice == 1) break;
   else if (menuChoice == 2) {
   playAgain=false;
   goto main_menu;
   break;
}

else if (menuChoice==3)
 {
   cout<<"\t\t Exiting Game.... GOOD BYE! ";
   exit(0);
 }
  else{
   if (cin.fail() || menuChoice < 1 || menuChoice > 3) {
   cin.clear();
   cin.ignore(1000, '\n');
   drawFullScreenBorder();      // Draw full screen border
   CursorLocation(5, 2);        // Print inside the border
   cout << "\n\t\t Invalid input! Please enter 1, 2, or 3.\n";
   this_thread::sleep_for(chrono::seconds(2));
   system("CLS");
   continue;
  }
```

```
            }
        }
    }
}
        else{
            if(mode==3){
                cout<<"\n\n ";
                cout<<"\t\t Exiting Game.... GOOD BYE! ";
                break;
            }
        }

        char spaces[9]= {' ',' ',' ',' ',' ',' ',' ',' ',' '};

         cout<<"\n\n";
        printBoard(spaces);
        this_thread::sleep_for(chrono::seconds(5));
        system("CLS");




       return 0;
```

```
}
}
```