TIC TAC TOE

Final project

Game description

This is a project TIC TAC TOE game using c programming language.

User will have small menu with options:

1. **Start game**
    1. **PLAYER vs COMPUTER**
        - Then player will choose either X or O
        - Then player will choose the difficulty of the game and game will start
    2. **PLAYER 1 vs PLAYER 2**
        - Game will start after choosing this option
2. **Game history**
    - In this section all the previously games results will be displayed
3. **Quit game**
    - Exit(1);

PSUEDOCODE

Code is seperated into 15 functions:

Global variables:

```
char board[3][3]; //board
char PLAYER = 'X'
char COMPUTER = 'O'
const char PLAYER2 = 'O'
```

Menu part:

void startMenu();

- This will call using switch either gameMode()or gameHistory() functions depending on user choice, or will remove(“history.txt”) file exit the game.

void gameMode();

- Asks user if he wants to play against computer or another player.
- Then asks which player user wants to be.
- Then asks difficulty level.
- And if game is hard mode - call playWithComputer(1) .
- If game is easy mode – call playWithComputer(0).
- If playing against human – call playWithHuman();

void gameHistory();

- Will read data from history.txt file.

Game structure part:

void playWithComputer(int);

- This function takes integer as a parameter and if integer = 0 computerEasyMove() function will be used.
- If integer = 1 computerHardMove() function will be used.
- Game is in loop and wont and while winner is empty and checkspaces() function doesn’t returns 0.
- In loop if Player is assigned to ‘X’ first playerMove(PLAYER) will be called and then computer move’s function. Or if player is ‘O’ opposite will happen.
- Then after game ends winner will be displayed using printWinner(winner) function
- Then addHistory(winner, loser) function will be called to save the data
- user will be asked if he/she is going to continue the game or not.
- Then startMenu() will be called again.

void playWithHuman();

- This function will be in loop and first will be called playerMove(PLAYER).
- Then playerMove(PLAYER2).
- Then after game ends winner will be displayed using printWinner(winner) function.
- Then addHistory(winner, loser) function will be called to save the data
- user will be asked if he/she is going to continue the game or not.
- Then startMenu() will be called again.

void resetBoard();

- Will assign board positions to ‘ ’ using for loop.

void printBoard();

- Prints the board.

int checkSpaces();

- This function has variable freespaces and if space in board is taken it will be decremented in a loop and then the number of free spaces will be returned. Using this function we will determine if the game is still going or not.

void playerMove(char);

- This function will have one parameter which will determine which players move is it.
- Then player will enter row and column. And their sign will be placed at given position.

void computerEasyMove();

- This function will call lfsr32(seed) function which will generate two numbers from 1-3 and then the sign will be placed at random position

void computerHardMove();

-

char checkWinner();

- This function will loop through the rows and columns. Will check if the row/column contains all the same signs and will return the char of the sign
- Then will check if diagonals contains all the same signs and will return the char of the sign
- If no winner empty char will be returned

void printWinner(char);

- Takes winner char as a parameter and if winner is PLAYER it will be displayed
- If winner is PLAYER2 it will be displayed
- And if winner is COMPUTER it will be displayed

void addHistory(char*,char*);

- This function is taking two strings as a parameter and first is winner and second one is loser
- Then history.txt file is being created and opend where the information about the game is stored

unsigned long lfsr32(unsigned long seed);

- In this function it takes seed as a parameter and new bit is generated and shifting seed to the right and putting new bit at the front shifted by 31-bit then returning generated seed

Driver:

int main()

- Welcome message is displayed
- And startMenu() function is called

Estimated work time

Writing this project took me around 16-18 hours.

Source code

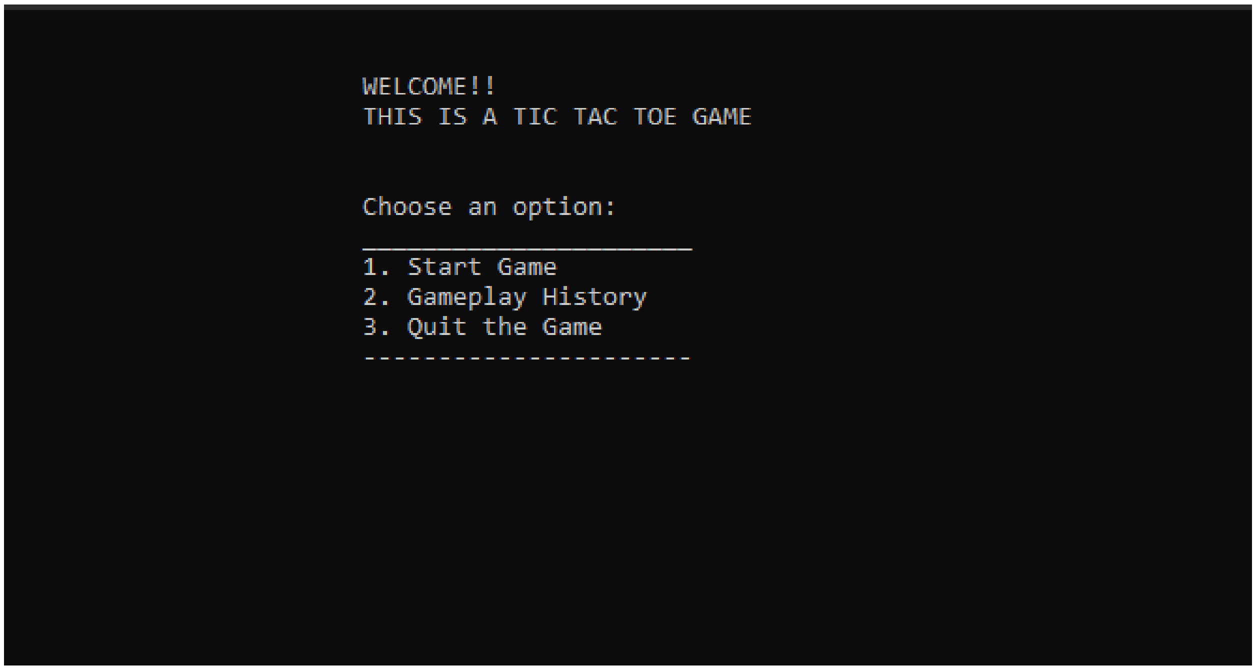
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h> //for getch()

char board[3][3]; //board
char PLAYER = 'X';
char COMPUTER = 'O';
const char PLAYER2 = 'O';

//menu part
void startMenu();
void gameMode();
void gameHistory();

void playWithComputer(int);
void playWithHuman();
//game structure
void resetBoard();
void printBoard();
int checkSpaces();
void playerMove(char);
void computerEasyMove();
void computerHardMove();
char checkWinner();
void printWinner(char);
void addHistory(char*,char*);
unsigned long lfsr32(unsigned long seed);

//driver
int main(){
  printf("\n\n\t\t\tWELCOME!!\n");
  printf("\t\t\tTHIS IS A TIC TAC TOE GAME\n\n");
  startMenu();
  return 0;
}
```

```
//menu part
void startMenu(){
  char option;
  printf("\n\t\t\tChoose an option:\n");
  printf("\t\t\t____________________\n");
  printf("\t\t\t1. Start Game\n");
  printf("\t\t\t2. Gameplay History\n");
  printf("\t\t\t3. Quit the Game\n");
  printf("\t\t\t----------------------\n");
  while(1){
    printf("\t\t\t");
    option = getch();
    switch (option)
    {
      case '1':
        system("cls");
        gameMode();
        break;
      case '2':
        system("cls");
        gameHistory();
        break;
      case '3':
        remove("history.txt");
        exit(1);
      default:
        printf("Invalid option!! CHOOSE AGAIN\n");
        continue;
    }
  }
}

void gameMode(){
  printf("\n\n\t\t\t1. PLAYER vs COMPUTER\n");
  printf("\t\t\t2. PLAYER 1 vs PLAYER 2\n");
  printf("\n\t\t\t");
  char option;
  option = getch();
  if(option == '1'){
    //asking what user wants to be
    system("cls");
    char sign;
    printf("\n\n\t\t\tChoose what you want to play\n");
    printf("\n\t\t\t\t1. X\n");
    printf("\t\t\t\t2. O\n");
    while(1){
      printf("\n\t\t\t");
      sign = getch();
      if(sign == '2'){
        PLAYER = 'O';
        COMPUTER = 'X';
        break;
      }
      else if(sign == '1'){
        PLAYER = 'X';
        COMPUTER = 'O';
        break;
      }
      else {
        printf("\t\t\tInvalid Option\n");
        continue;
      }
    }
    //then ask difficulty level
    system("cls");
    char lvl;
    printf("\n\n\t\t\tChoose difficulty level:\n");
    printf("\n\t\t\t1. Beginner\n");
    printf("\t\t\t2. Proffesional\n");
    do
    {
      printf("\t\t\t\t");
      lvl = getch();
      if(lvl == '1') {
        playWithComputer(0);
      }
      else if(lvl == '2'){
        playWithComputer(1);
      }
      else{
        printf("\t\t\tInvalid Option\n");
      }
    } while (lvl != '1' || lvl != '2');
  }
  else if(option == '2'){
    playWithHuman();
  }
  else{
    printf("\t\t\tInvalid option\n");
  }
}

void gameHistory(){
  FILE *filePtr = fopen("history.txt", "r");
  int counter = 0;
  if (filePtr == NULL){
        printf("Error opening the file history.txt\n");
        return;
  }
  printf("\n\t\t\t\tGAME HISTORY\n");
  printf("\t\t\t---------------------------\n\n");
  char text[50];
  while(fgets(text, 50, filePtr)){
    printf("\t\t\tN%d. %s", ++counter,text);
  }
  fclose(filePtr);
  getch();
  system("cls");
```

```
 startMenu();
}

void playWithComputer(int n){
   char choice;
   do{//looping
     char winner = ' ';
     resetBoard();
     while(winner == ' ' && checkSpaces()!=0 ){
       system("cls");
       printBoard();
       if(PLAYER == 'X')
         playerMove(PLAYER);
       else{
         if(n == 0)
           computerEasyMove();
         if(n == 1)
           computerHardMove();
       }
       winner = checkWinner();
       if(winner != ' ' || checkSpaces() == 0){
         break;
       }
       system("cls");
       printBoard();
       if(COMPUTER == 'O'){
         if(n == 0)
           computerEasyMove();
         if(n == 1)
           computerHardMove();
       }
       else
         playerMove(PLAYER);
       winner = checkWinner();
       if(winner != ' ' || checkSpaces() == 0){
         break;
       }
     }
     system("cls");
     printBoard();
     printWinner(winner);
     if(winner == PLAYER) addHistory("Player","Computer");
     else addHistory("Computer","Player");
     printf("\n\t\t\tWould you like to continue the game?\n");
     printf("\n\t\t\t(Y - Play Again // N - Back to Menu)\n");
     printf("\n\t\t\t\t\t");
     choice = getch();
     system("cls");
   }while(choice =='Y' || choice == 'y');
   //calling the startMenu again
   startMenu();
}

void playWithHuman(){
 char choice;
 PLAYER = 'X';
 do{//looping
   char winner = ' ';
   resetBoard();
   while(winner == ' ' && checkSpaces()!=0 ){
     system("cls");
     printBoard();
     playerMove(PLAYER);
     winner = checkWinner();
     if(winner != ' ' || checkSpaces() == 0){
       break;
     }
     system("cls");
     printBoard();
     playerMove(PLAYER2);
     winner = checkWinner();
     if(winner != ' ' || checkSpaces() == 0){
       break;
     }
     system("cls");
   }
   system("cls");
   printBoard();
   printWinner(winner);
   if(winner == PLAYER) addHistory("Player 1","Player 2");
   else addHistory("Player 2","Player 1");
   printf("\n\t\t\tWould you like to continue the game?\n");
   printf("\n\t\t\t(Y - Play Again // N - Back to Menu)\n");
   printf("\n\t\t\t\t\t");
   choice = getch();
   system("cls");
 }while(choice =='Y' || choice == 'y');
 //calling the startMenu again
 startMenu();

}

//structure part
void resetBoard(){
 int i,j;
 for(i = 0; i < 3; i++){
   for(j = 0; j < 3; j++){
     board[i][j]= ' ';
   }
 }
}

void printBoard(){
 printf("\n\n\t\t\t\t     |     |     \n");
 printf("\t\t\t\t  %c  |  %c  |  %c  \n", board[0][0], board[0][1], board[0][2]);
 printf("\t\t\t\t-----|-----|-----\n");
 printf("\t\t\t\t  %c  |  %c  |  %c \n", board[1][0], board[1][1], board[1][2]);
 printf("\t\t\t\t-----|-----|-----\n");
 printf("\t\t\t\t  %c  |  %c  |  %c \n", board[2][0], board[2][1], board[2][2]);
 printf("\t\t\t\t     |     |    \n");
 printf("\n");
}

int checkSpaces(){
 int freeSpaces = 9;
 int i,j;
 for(i = 0; i < 3; i++){
   for(j = 0; j < 3; j++){
     if(board[i][j] != ' '){
       freeSpaces--;
     }
   }
 }
 return freeSpaces;
}

void playerMove(char player){
 int x,y;
 printf("\n\t\t\t\tPLAYER %c MOVE:\n\n",player);
 do
 {
   printf("\t\t\t\tEnter row #(1-3): ");
   scanf("%d", &x);
   x--;
   printf("\t\t\t\tEnter column #(1-3): ");
   scanf("%d", &y);
   y--;
   if(board[x][y]!=' '){
     printf("\t\t\t\tInvalid Move!!\n");
   }
   else{
     board[x][y] = player;
     break;
   }
 } while (board[x][y]!= ' ');

}

void computerEasyMove(){
 int x;
 int y;
 static unsigned long firstSeed = 0x5AA5F100;
 static unsigned long secondSeed = 0x5AA5F700;

 if(checkSpaces() > 0){
   do{
     firstSeed = lfsr32(firstSeed);
     secondSeed = lfsr32(secondSeed);
     x = firstSeed % 3;
     y = secondSeed % 3;
   }while(board[x][y]!=' ');
   board[x][y] = COMPUTER;
 }
 else{
   printWinner(' ');
 }
}

void computerHardMove(){
 if(checkSpaces()> 0){
   int i;
   for(i = 0; i < 3; i++){
     //rows check for computer
     if(board[i][0] == COMPUTER && board[i][1] == COMPUTER){
       if(board[i][2] == ' '){
         board[i][2] = COMPUTER;
         return;
       }
       continue;
     }
     if(board[i][1] == COMPUTER && board[i][2] == COMPUTER){
       if(board[i][0] == ' '){
         board[i][0] = COMPUTER;
         return;
       }
       continue;

     }
     if(board[i][0] == COMPUTER && board[i][2] == COMPUTER){
       if(board[i][1] == ' '){
         board[i][1] = COMPUTER;
         return;
       }
       continue;

     }
     //columns check for computer
     if(board[0][i] == COMPUTER && board[1][i] == COMPUTER){
       if(board[2][i] == ' '){
         board[2][i] = COMPUTER;
         return;
       }
       continue;

     }
     if(board[1][i] == COMPUTER && board[2][i] == COMPUTER){
       if(board[0][i] == ' '){
         board[0][i] = COMPUTER;
         return;
       }
       continue;

     }
     if(board[0][i] == COMPUTER && board[2][i] == COMPUTER){
       if(board[1][i] == ' '){
         board[1][i] = COMPUTER;
         return;
       }
       continue;
     }
     //rows check for player
     if(board[i][0] == PLAYER && board[i][1] == PLAYER){
       if(board[i][2] == ' '){
         board[i][2] = COMPUTER;
         return;
       }
       continue;
     }
     if(board[i][1] == PLAYER && board[i][2] == PLAYER){
       if(board[i][0] == ' '){
         board[i][0] = COMPUTER;
         return;
       }
       continue;
     }
     if(board[i][0] == PLAYER && board[i][2] == PLAYER){
       if(board[i][1] == ' '){
         board[i][1] = COMPUTER;
         return;
       }
       continue;
     }
     //columns check for player
     if(board[0][i] == PLAYER && board[1][i] == PLAYER){
       if(board[2][i] == ' '){
         board[2][i] = COMPUTER;
         return;
       }
       continue;
     }
     if(board[1][i] == PLAYER && board[2][i] == PLAYER){
       if(board[0][i] == ' '){
         board[0][i] = COMPUTER;
         return;
       }
       continue;
     }
     if(board[0][i] == PLAYER && board[2][i] == PLAYER){
       if(board[1][i] == ' '){
         board[1][i] = COMPUTER;
         return;
       }
       continue;
     }
   }
   //first diag diagonal check for computer
   if(board[0][0] == COMPUTER && board[1][1] == COMPUTER){
     if(board[2][2] == ' '){
       board[2][2] = COMPUTER;
       return;
     }
     else if(board[2][2] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   if(board[1][1] == COMPUTER && board[2][2] == COMPUTER){
     if(board[0][0] == ' '){
       board[0][0] = COMPUTER;
       return;
     }
     else if(board[0][0] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   if(board[0][0] == COMPUTER && board[2][2] == COMPUTER){
     if(board[1][1] == ' '){
       board[1][1] = COMPUTER;
       return;
     }
     else if(board[1][1] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   //second diagonal check for player
   if(board[0][2] == COMPUTER && board[1][1] == COMPUTER){
     if(board[2][0] == ' '){
       board[2][0] = COMPUTER;
       return;
     }
   }
   if(board[1][1] == COMPUTER && board[2][0] == COMPUTER){
     if(board[0][2] == ' '){
       board[0][2] = COMPUTER;
       return;
     }
   }
   if(board[0][2] == COMPUTER && board[2][0] == COMPUTER){
     if(board[1][1] == ' '){
       board[1][1] = COMPUTER;
       return;
     }
     else if(board[1][1] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   //first diagonal check for Computer
   if(board[0][0] == PLAYER && board[1][1] == PLAYER){
     if(board[2][2] == ' '){
       board[2][2] = COMPUTER;
       return;
     }
     else if(board[2][2] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   if(board[1][1] == PLAYER && board[2][2] == PLAYER){
     if(board[0][0] == ' '){
       board[0][0] = COMPUTER;
       return;
     }
     else if(board[0][0] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   if(board[0][0] == PLAYER && board[2][2] == PLAYER){
     if(board[1][1] == ' '){
       board[1][1] = COMPUTER;
       return;
     }
     else if(board[1][1] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   //second diag check for player
   if(board[0][2] == PLAYER && board[1][1] == PLAYER){
     if(board[2][0] == ' '){
       board[2][0] = COMPUTER;
       return;
     }
   }
   if(board[1][1] == PLAYER && board[2][0] == PLAYER){
     if(board[0][2] == ' '){
       board[0][2] = COMPUTER;
       return;
     }
   }
   if(board[0][2] == PLAYER && board[2][0] == PLAYER){
     if(board[1][1] == ' '){
       board[1][1] = COMPUTER;
       return;
     }
     else if(board[1][1] == COMPUTER){
       computerEasyMove();
       return;
     }
   }
   //side middle positions
   if(board[0][1] == PLAYER|| board[1][2] == PLAYER || board [2][1] == PLAYER || board[1][0] || board[1][1]){ //maybe leave
 board[1][1] there
     computerEasyMove();
     return;
   }
   //corner positions
   if(board[0][0] == PLAYER || board[0][2] == PLAYER || board[2][0] == PLAYER || board[2][2] == PLAYER ){
     if(board[1][1] == ' '){
       board[1][1] = COMPUTER;
       return;
     }
     if(board[0][0] == ' '){
       board[0][0] = COMPUTER;
       return;
     }
     if(board[0][2] == ' '){
       board[0][2] = COMPUTER;
       return;
     }
     if(board[2][0] == ' '){
       board[2][0] = COMPUTER;
       return;
     }
     if(board[2][2] == ' '){
       board[2][2] = COMPUTER;
       return;
     }
   }
 }
 else{
   printWinner(' ');
 }
 }

 char checkWinner(){
 int i;
 for(i = 0; i < 3; i++){ //column checks
   if(board[i][0] == board[i][1] && board[i][0] == board[i][2]){
     return board[i][0];
   }
 }
 for(i = 0; i < 3; i++){ //row checks
   if(board[0][i] == board[1][i] && board[0][i] == board[2][i]){
     return board[0][i];
   }
 }
 if(board[0][0] == board[1][1] && board[0][0] == board[2][2]){
   return board[0][0];
 }
 if(board[0][2] == board[1][1] && board[0][2] == board[2][0]){
   return board[0][2];
 }
 return ' ';
 }

 void printWinner(char winner){
 if(winner == PLAYER){
   printf("\t\t\t************************************\n");
   printf("\t\t\t\tPLAYER %c WINS!!\n", PLAYER);
   printf("\t\t\t************************************\n");
 }
 else if(winner == PLAYER2){
   printf("\t\t\t************************************\n");
   printf("\t\t\t\tPLAYER O WINS!!\n");
   printf("\t\t\t************************************\n");
 }
 else if(winner == COMPUTER){
   printf("\t\t\t************************************\n");
   printf("\t\t\tCOMPUTER WINS!! BETTER LUCK NEXT TIME!\n");
   printf("\t\t\t************************************\n");
 }
 else {
   printf("\t\t\t************************************\n");
   printf("\t\t\t\tIT'S A TIE :|\n");
   printf("\t\t\t************************************\n");
 }
 }

 void addHistory(char winner[], char loser[]){
 FILE* filePtr = fopen("history.txt", "a");
 if (filePtr == NULL){
   printf("Error opening the file history.txt\n");
   return;
 }
 if(checkWinner() == ' '){
   fprintf(filePtr,"%s and %s had a tie!\n", winner, loser);
 }
 fprintf(filePtr,"%s won against %s\n", winner, loser);
 fclose(filePtr);
 }

 unsigned long lfsr32(unsigned long seed){
   if(seed == 0)//if seed is equal 0 it will return the next 32-bit value
       return seed + 1;
   unsigned long new_bit;//creating new bit
   for(int i = 0; i< 32; i++){//calculating new bit
   //feedback polinomial: x^32 + x^30 + x^26 + x^25 + 1
   //              /tap 32/     /tap 30/     /tap 26/     /tap 25/
       new_bit = ( (seed >> 0) ^ (seed >> 2) ^ (seed >> 6) ^ (seed >> 7) ) & 1;
       //shifting seed to the right and putting new bit at the front shifted by 31-bit
       seed = (seed >> 1) | (new_bit << 31);
   }
   return seed;//returning seed
 }
```

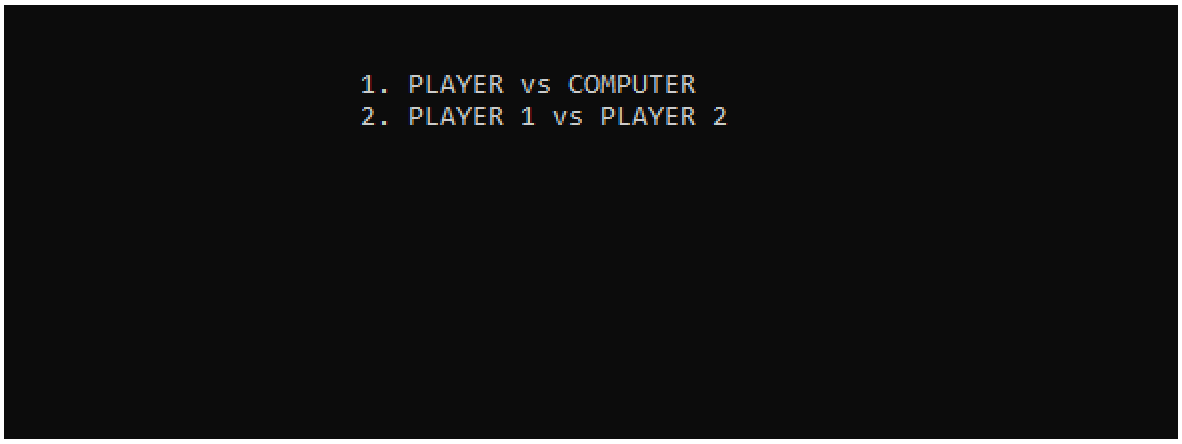
Output screenshots

Main menu:



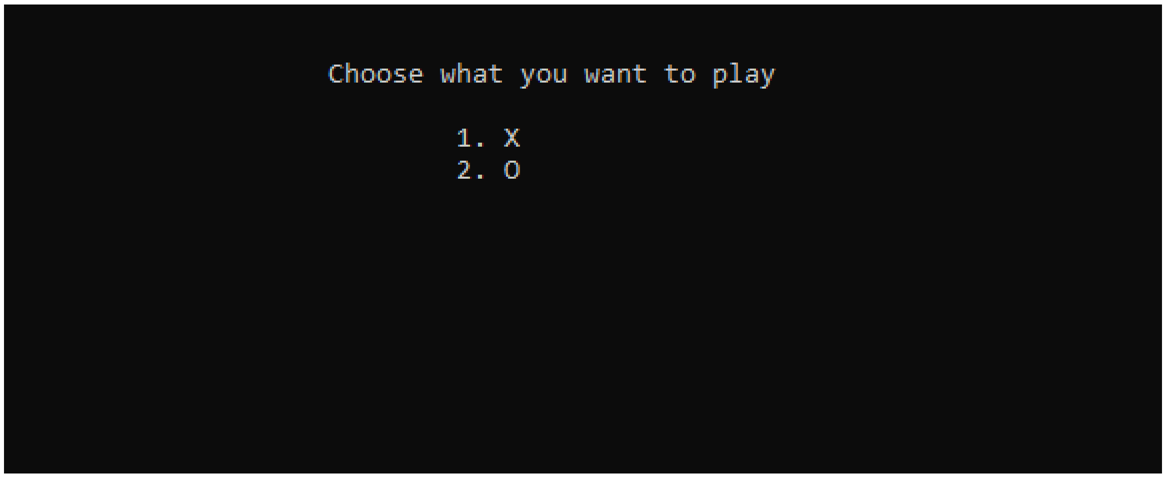
```
WELCOME!!
THIS IS A TIC TAC TOE GAME


Choose an option:
______________________
1. Start Game
2. Gameplay History
3. Quit the Game
----------------------
```

Start game:



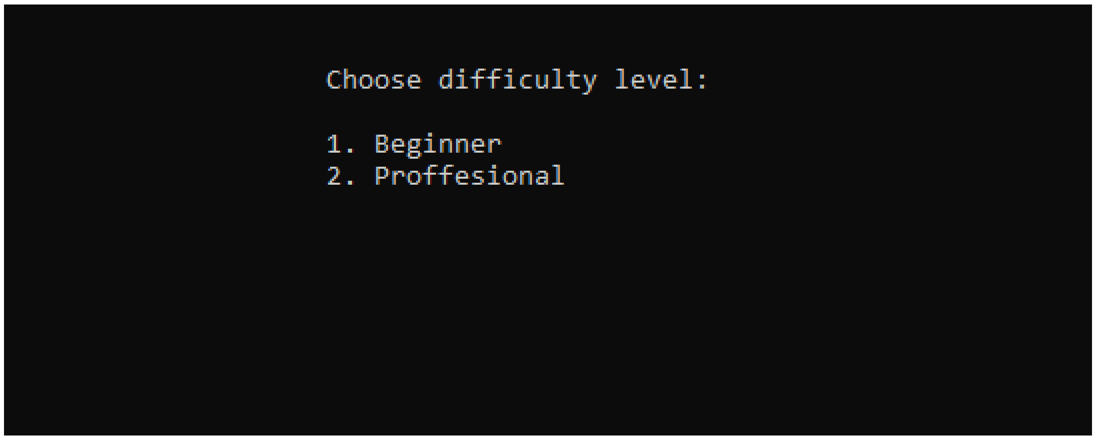
```
1. PLAYER vs COMPUTER
2. PLAYER 1 vs PLAYER 2
```

Player vs computer:



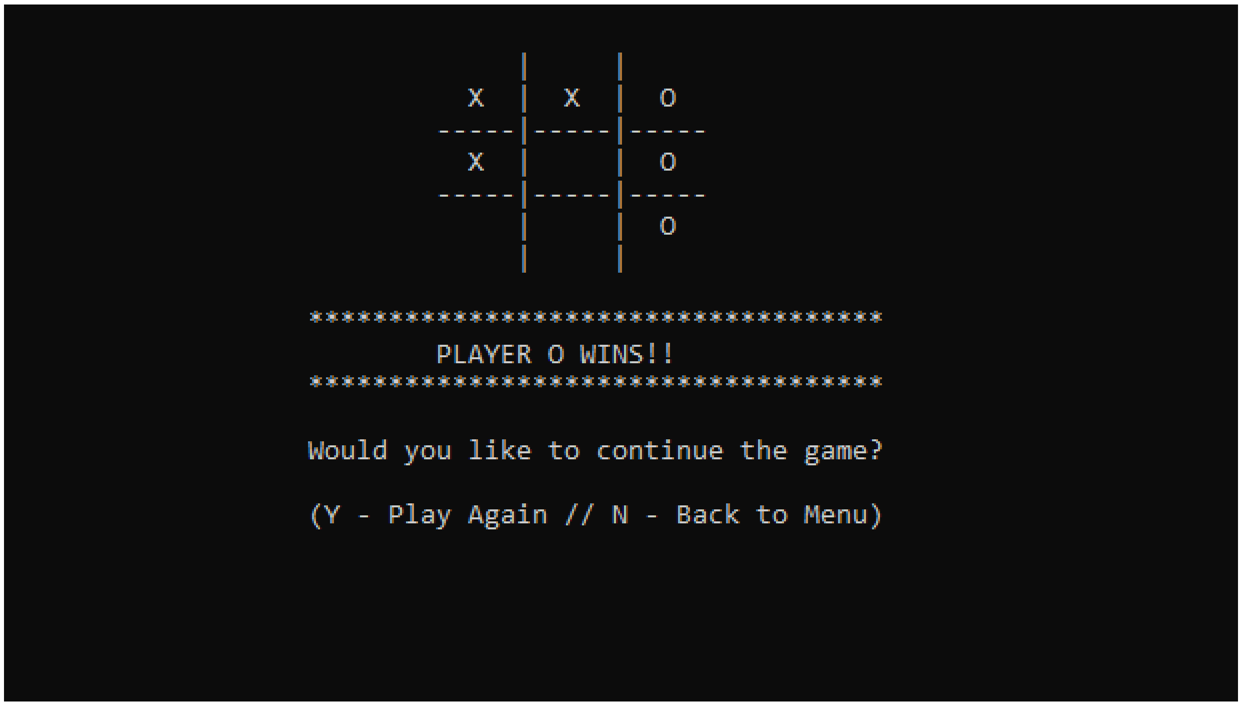
```
                    Choose what you want to play

                            1. X
                            2. O
```

After choosing one of them:



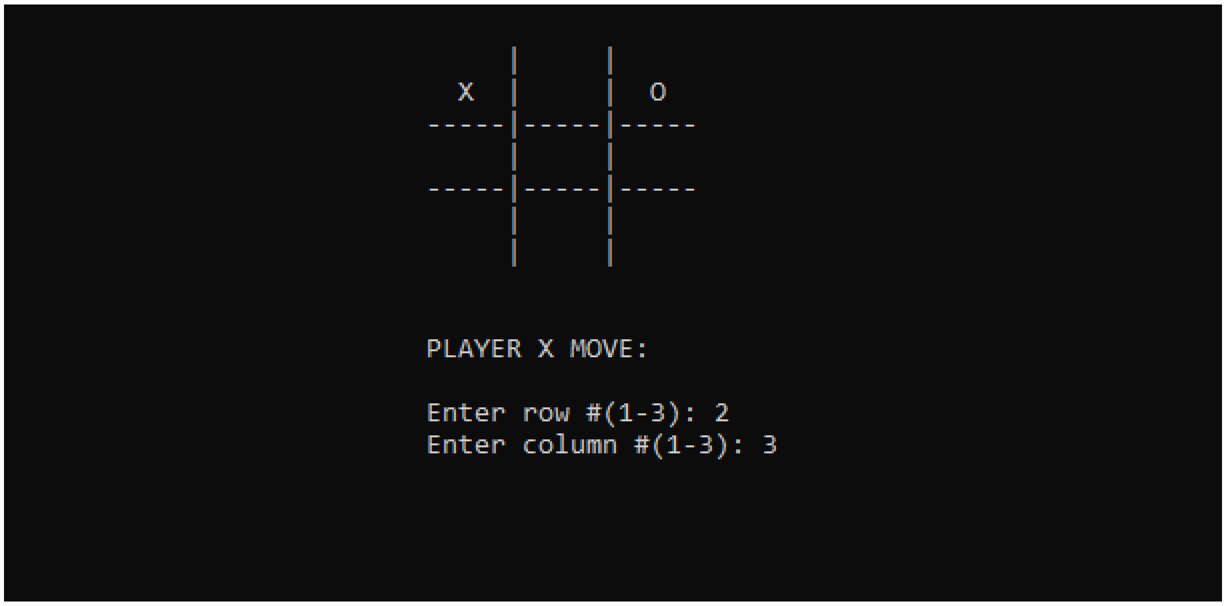
```
                    Choose difficulty level:

                    1. Beginner
                    2. Proffesional
```

After choosing any level game will start:



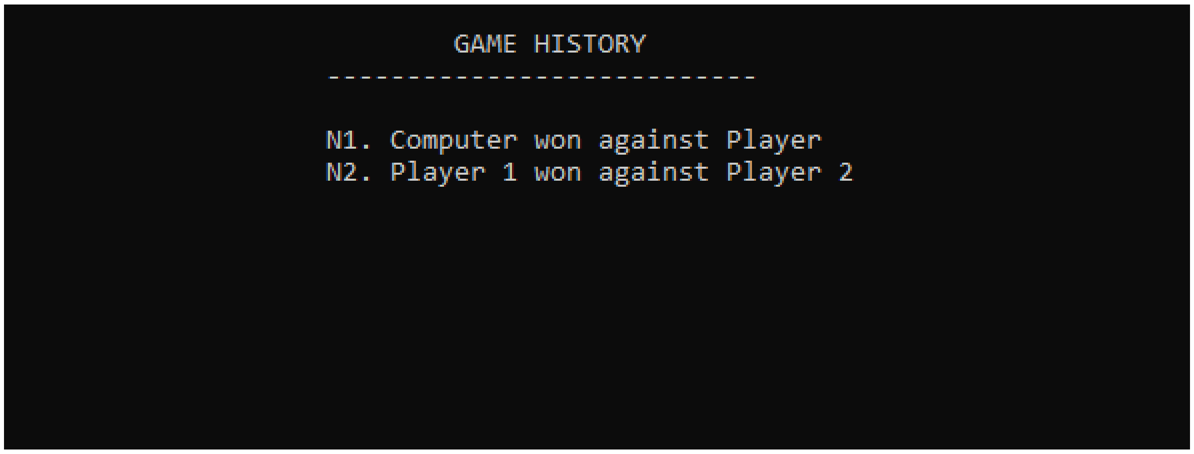
```
      |     |
   X  |  X  |  O
 -----|-----|-----
   X  |     |  O
 -----|-----|-----
      |     |  O
      |     |

************************************
        PLAYER O WINS!!
************************************

Would you like to continue the game?

(Y - Play Again // N - Back to Menu)
```

If you choose player 1 vs player 2 directly board will be displayed:



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


PLAYER X MOVE:

Enter row #(1-3): 2
Enter column #(1-3): 3
```

After returning to the main menu choose game history option:



```
          GAME HISTORY
--------------------------

N1. Computer won against Player
N2. Player 1 won against Player 2
```

After choosing the 3rd option game will exit.

Video demonstration

https://drive.google.com/file/d/1Mmusq_JuHk7bvGlS2tNn16OkvaEnLave/view?usp=sharing

Platforms and tools

Code is written on windows 10 OS. IDE used is Visual Studio Code. On the presentation CodeBlock will be used for execution of the code.

Resources

Resource 1 – [**Bro Code**] [**C Tic Tac Toe game**]

link: [https://www.youtube.com/watch?v=_889aB2D1KI]

This video helped me to create base structure.

Resource 2 – [https://www.includehelp.com/c-programming-questions/how-to-clear-output-screen-in-c.aspx] helped me to find the function (system("cls")) which is used to clear output screen.