***Лабораторна робота №1***

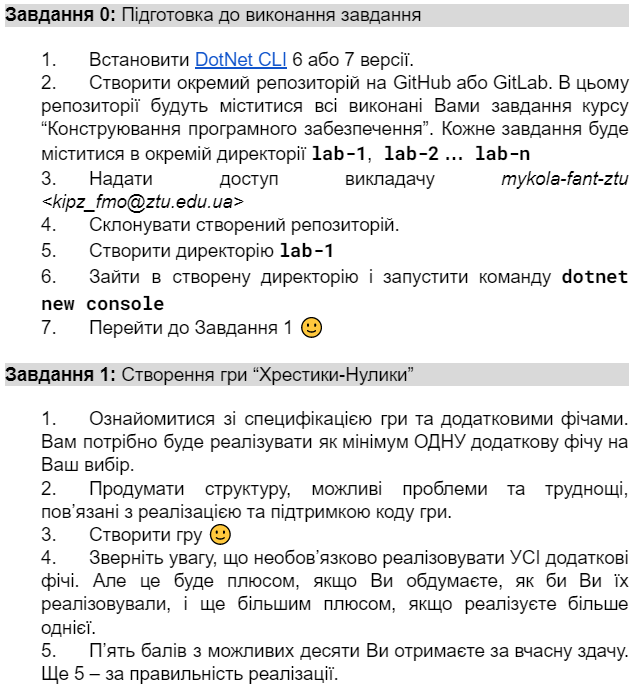
***Тема: Повторення ООП. Гра “Хрестики-Нулики”***

***Мета роботи***: згадати принципи ООП, написати цікаву консольну гру.

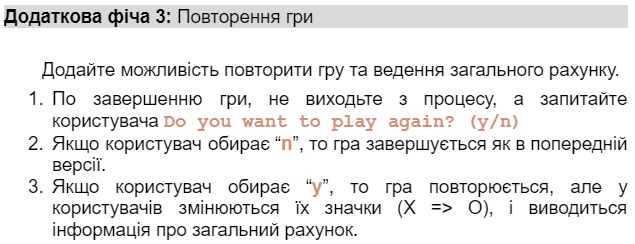
***Хід роботи:***

***Завдання на лабораторну роботу:***

1. ***Завдання:***



1. ***Додаткові фічі:***

******

1. ***Лістинг програми:***
   1. ***Бібліотека класів:***
      1. ***TicTacToe.cs:***

namespace TicTacToeLibrary

{

public class TicTacToe

{

protected static Player[] Players = new Player[2];

protected static string[,] GameField = new string[3, 3];

protected static XML PlayersFile = new XML("players.xml");

protected static void InitializatePlayers()

{

if (Players[0] is null && Players[1] is null)

{

Player[] SavedPlayers = PlayersFile.ReadPlayers();

if (SavedPlayers[0] is not null && SavedPlayers[1] is not null)

{

foreach (Player player in SavedPlayers)

Console.WriteLine($"Number: {player.GetNumber()}; Role: {player.GetRole()}");

Players[0] = SavedPlayers[0];

Players[1] = SavedPlayers[1];

}

else

{

Players[0] = new Player(1);

Players[1] = new Player(2, Players[0].GetRole());

}

}

}

protected static void InitializateGameFields()

{

int FieldNumber = 1;

for (int i = 0; i < GameField.GetLength(0); i++)

{

for (int j = 0; j < GameField.GetLength(1); j++)

{

GameField[i, j] = FieldNumber.ToString();

FieldNumber++;

}

}

}

public static void StartGame()

{

Console.Clear();

Console.WriteLine("Welcome to Tic Tac Toe!\n");

InitializatePlayers();

Menu();

}

protected static void Menu()

{

Console.Clear();

Console.WriteLine("\tMenu");

Console.WriteLine("1. Play");

Console.WriteLine("2. Rating");

Console.WriteLine("3. Exit");

int MenuOption;

do

{

Console.Write($"\nSelect menu option: ");

if (int.TryParse(Console.ReadLine(), out MenuOption) && (MenuOption >= 1 && MenuOption <= 3))

break;

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"There is no option with '{MenuOption}' in menu.");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("\nEnter valid option: ");

}

} while (!int.TryParse(Console.ReadLine(), out MenuOption) && (MenuOption >= 1 && MenuOption <= 3));

switch (MenuOption)

{

case 1:

{

Console.Clear();

Play();

break;

}

case 2:

{

Console.Clear();

ShowPlayersRating();

break;

}

case 3:

{

return;

}

}

}

protected static void Play()

{

string? PlayAgain;

do

{

Game();

Console.Write("Do you want to play again? (y/n) ");

PlayAgain = Console.ReadLine();

if (PlayAgain == null || (PlayAgain != "y" && PlayAgain != "n"))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Not 'y' or 'n'.");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("\nEnter correct symbol: ");

PlayAgain = Console.ReadLine();

}

if(PlayAgain == "y")

{

string TempRole = Players[0].GetRole();

Players[0].SetRole(Players[1].GetRole());

Players[1].SetRole(TempRole);

}

} while (PlayAgain == "y");

PlayersFile.WritePlayers(Players);

StartGame();

}

protected static void Game()

{

InitializateGameFields();

bool Playing = true;

uint PlayerNumber;

if (Players[0].GetRole() == "X")

PlayerNumber = 1;

else

PlayerNumber = 2;

Console.Clear();

while (Playing)

{

Console.WriteLine("Let's play Tic Tac Toe!\n");

Console.WriteLine($"Player {PlayerNumber}'s turn. Player's role '{Players[PlayerNumber - 1].GetRole()}'.\nSelect from 1 to 9 the game board.\n");

ShowGameField();

if (CheckWin(Players[0].GetRole()))

{

Console.WriteLine("\n\n\tCongrats!");

Console.WriteLine("First player won!");

Players[0].SetRating(Players[0].GetRole());

Players[1].SetRating("TotalGames");

break;

}

else if (CheckWin(Players[1].GetRole()))

{

Console.WriteLine("\n\n\tCongrats!");

Console.WriteLine("Second player won!");

Players[1].SetRating(Players[1].GetRole());

Players[0].SetRating("TotalGames");

break;

}

else if (IsNoMoves())

{

Console.WriteLine("\n\nDraw!");

Console.WriteLine("No players won!");

Players[0].SetRating("TotalGames");

Players[1].SetRating("TotalGames");

break;

}

int Move;

do

{

do

{

Console.Write($"\n\nEnter number: ");

if (int.TryParse(Console.ReadLine(), out Move) && (Move >= 1 && Move <= 9))

break;

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"There is no field with '{Move}' on the field.");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("\nEnter valid field: ");

}

} while (!int.TryParse(Console.ReadLine(), out Move) && (Move >= 1 && Move <= 9));

} while (!TickGameField(Move, Players[PlayerNumber - 1].GetRole()));

switch (PlayerNumber)

{

case 1: PlayerNumber = 2; break;

case 2: PlayerNumber = 1; break;

}

Console.Clear();

}

}

protected static bool TickGameField(int Move, string Role)

{

bool[] IsTicked = new bool[9];

int NumericMove;

switch (Move)

{

// 1-st line

case 1:

{

if(int.TryParse(GameField[0, 0], out NumericMove))

GameField[0, 0] = Role;

else IsTicked[0] = true;

break;

}

case 2:

{

if (int.TryParse(GameField[0, 1], out NumericMove))

GameField[0, 1] = Role;

else IsTicked[1] = true;

break;

}

case 3:

{

if (int.TryParse(GameField[0, 2], out NumericMove))

GameField[0, 2] = Role;

else IsTicked[2] = true;

break;

}

// 2-nd line

case 4:

{

if (int.TryParse(GameField[1, 0], out NumericMove))

GameField[1, 0] = Role;

else IsTicked[3] = true;

break;

}

case 5:

{

if (int.TryParse(GameField[1, 1], out NumericMove))

GameField[1, 1] = Role;

else IsTicked[4] = true;

break;

}

case 6:

{

if (int.TryParse(GameField[1, 2], out NumericMove))

GameField[1, 2] = Role;

else IsTicked[5] = true;

break;

}

// 3-rd line

case 7:

{

if (int.TryParse(GameField[2, 0], out NumericMove))

GameField[2, 0] = Role;

else IsTicked[6] = true;

break;

}

case 8:

{

if (int.TryParse(GameField[2, 1], out NumericMove))

GameField[2, 1] = Role;

else IsTicked[7] = true;

break;

}

case 9:

{

if (int.TryParse(GameField[2, 2], out NumericMove))

GameField[2, 2] = Role;

else IsTicked[8] = true;

break;

}

}

for (int i = 0; i < IsTicked.Length; i++)

if (IsTicked[i])

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write($"{i + 1} field is already ticked! Choose another one.");

Console.ForegroundColor = ConsoleColor.White;

return false;

}

return true;

}

protected static bool IsNoMoves()

{

bool IsNoMoves = true;

for (int i = 0; i < GameField.GetLength(0); i++)

for (int j = 0; j < GameField.GetLength(1); j++)

{

int Numeric;

if (int.TryParse(GameField[i, j], out Numeric))

{

IsNoMoves = false;

break;

}

}

return IsNoMoves;

}

protected static void ShowGameField()

{

for (int i = 0; i < GameField.GetLength(0); i++)

{

for (int j = 0; j < GameField.GetLength(1); j++)

{

Console.Write($" {GameField[i, j]} ");

if(j != GameField.GetLength(1) - 1)

Console.Write("|");

}

if(i != GameField.GetLength(0) - 1)

Console.WriteLine("\n---+---+---");

}

}

protected static void ShowPlayersRating()

{

Console.WriteLine("\tRating");

for (int i = 0; i < Players.Length; i++)

{

double WinRate = 0;

if (Players[i].GetRating("TotalGames") != 0)

WinRate = Math.Round((Players[i].GetRating("X") + Players[i].GetRating("O")) / Players[i].GetRating("TotalGames") \* 100, 2);

Console.WriteLine($"\n\tPlayer {i + 1}");

Console.WriteLine($"X wins: {(int)Players[i].GetRating("X")}");

Console.WriteLine($"O wins: {(int)Players[i].GetRating("O")}");

Console.WriteLine($"Total games: {(int)Players[i].GetRating("TotalGames")}");

Console.WriteLine($"Win rate: {WinRate}%");

}

Console.Write("\nPress any KEY to go back to menu...");

Console.ReadKey();

StartGame();

}

protected static bool CheckWin(string PlayerRole)

{

// Horizontal

if (GameField[0, 0] == PlayerRole && GameField[0, 1] == PlayerRole && GameField[0, 2] == PlayerRole)

return true;

if (GameField[1, 0] == PlayerRole && GameField[1, 1] == PlayerRole && GameField[1, 2] == PlayerRole)

return true;

if (GameField[2, 0] == PlayerRole && GameField[2, 1] == PlayerRole && GameField[2, 2] == PlayerRole)

return true;

// Diagonal

if (GameField[0, 0] == PlayerRole && GameField[1, 1] == PlayerRole && GameField[2, 2] == PlayerRole)

return true;

if (GameField[0, 2] == PlayerRole && GameField[1, 1] == PlayerRole && GameField[2, 0] == PlayerRole)

return true;

// Coloumns

if (GameField[0, 0] == PlayerRole && GameField[1, 0] == PlayerRole && GameField[2, 0] == PlayerRole)

return true;

if (GameField[0, 1] == PlayerRole && GameField[1, 1] == PlayerRole && GameField[2, 1] == PlayerRole)

return true;

if (GameField[0, 2] == PlayerRole && GameField[1, 2] == PlayerRole && GameField[2, 2] == PlayerRole)

return true;

return false;

}

}

}

* + 1. ***Player.cs:***

namespace TicTacToeLibrary

{

public class Player

{

public struct Core

{

public int X;

public int O;

public int TotalGames;

public Core()

{

X = 0;

O = 0;

TotalGames = 0;

}

public Core(int x, int o, int totalGames)

{

X = x;

O = o;

TotalGames = totalGames;

}

}

protected int Number;

protected string Role;

protected Core Rating;

public Player(int number, string checkRole = "")

{

Number = number;

Role = EnterPlayerRole(number, checkRole);

Rating = new Core();

}

public Player(int number, string role, int x, int o, int totalGames)

{

Number = number;

Role = role;

Rating = new Core(x, o, totalGames);

}

public static string EnterPlayerRole(int PlayerNumber, string CheckRole = "")

{

string Role;

switch (CheckRole)

{

case "X": return "O";

case "O": return "X";

default: Console.WriteLine("Existing game roles: X, O."); break;

}

do

{

Console.Write($"Player {PlayerNumber} role: ");

Role = Console.ReadLine();

if (CheckRole != Role && (Role == "X" || Role == "O"))

break;

else

{

Console.Write("Error! Role is entered wrong. Repeat: ");

Role = Console.ReadLine();

}

} while (CheckRole == Role && (Role == "X" || Role == "O"));

return Role;

}

public void SetRating(string RatingKey)

{

switch (RatingKey)

{

case "X":

{

Rating.X++;

Rating.TotalGames++;

break;

}

case "O":

{

Rating.O += 1;

Rating.TotalGames++;

break;

}

case "TotalGames":

{

Rating.TotalGames++;

break;

}

}

}

public void SetRole(string NewRole)

{

if (NewRole != "X" && NewRole != "O")

return;

Role = NewRole;

}

public int GetNumber() => Number;

public string GetRole() => Role;

public double GetRating(string RatingKey = "")

{

switch (RatingKey)

{

case "X": return Rating.X;

case "O": return Rating.O;

case "TotalGames": return Rating.TotalGames;

}

return 0;

}

}

}

}

* 1. ***Консольний додаток:***
     1. ***CLI.cs:***

using TicTacToeLibrary;

namespace TicTacToeCLI

{

public class CLI

{

private static void Main(string[] args)

{

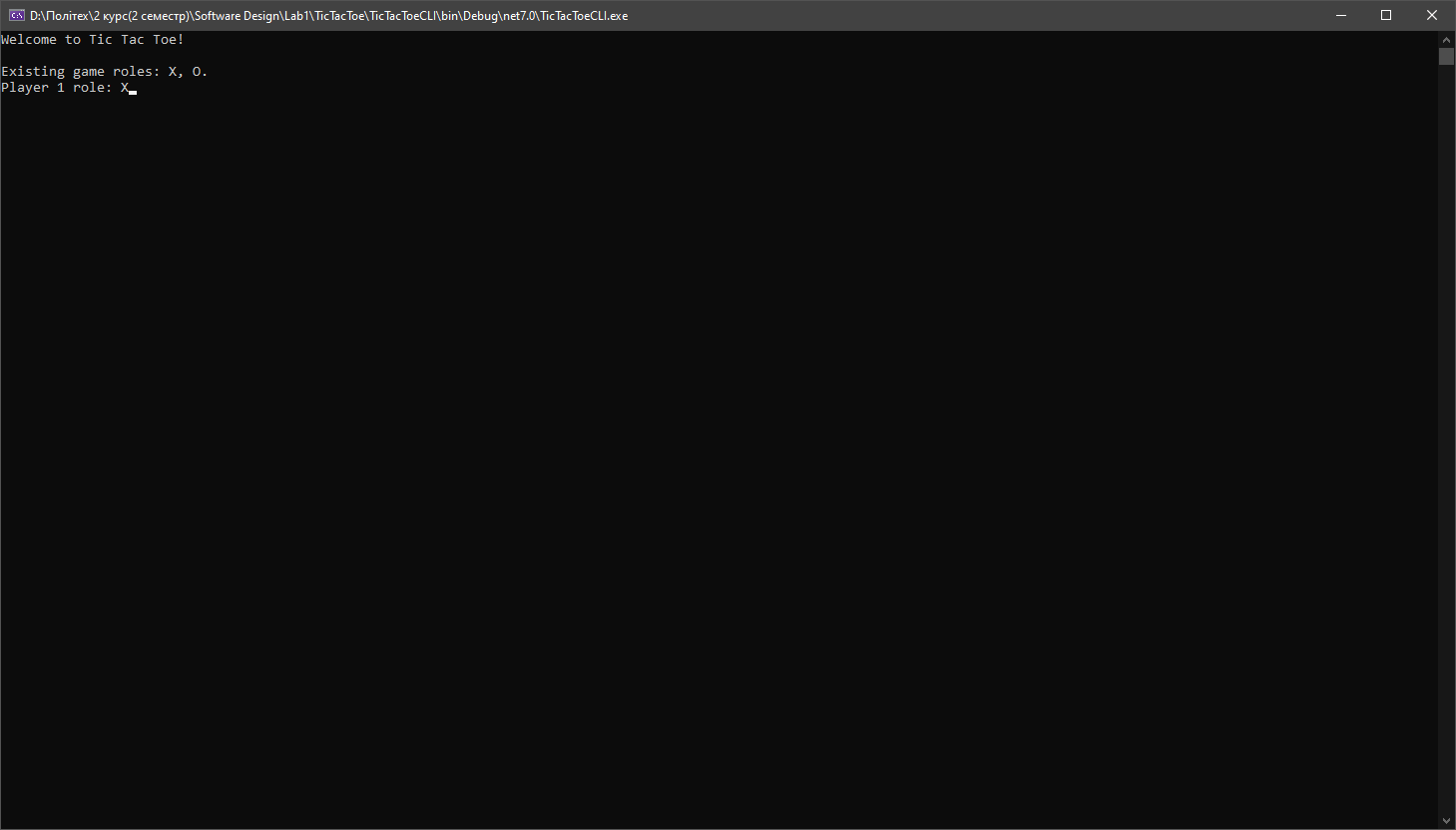
TicTacToe.StartGame();

}

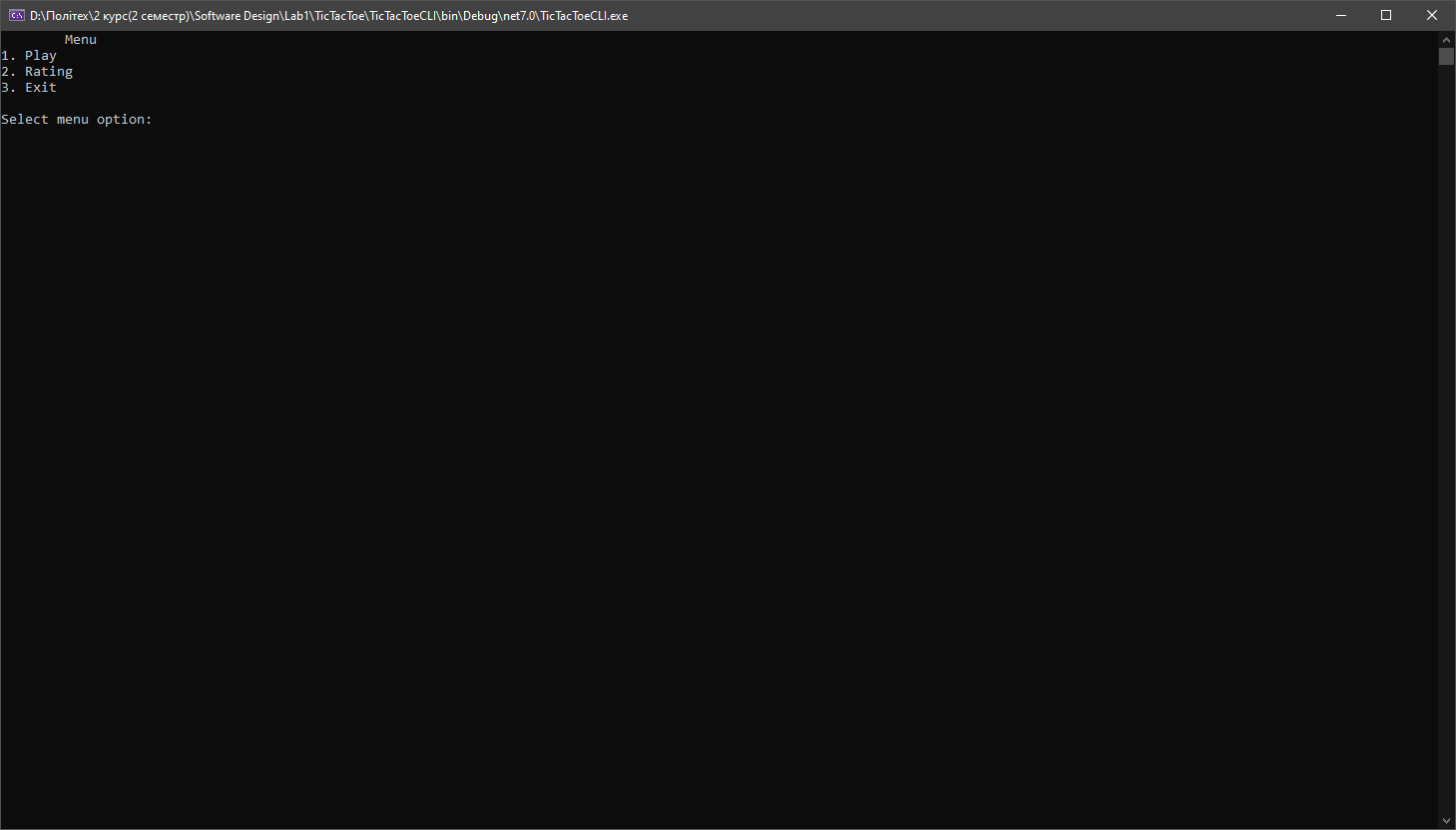
}

}

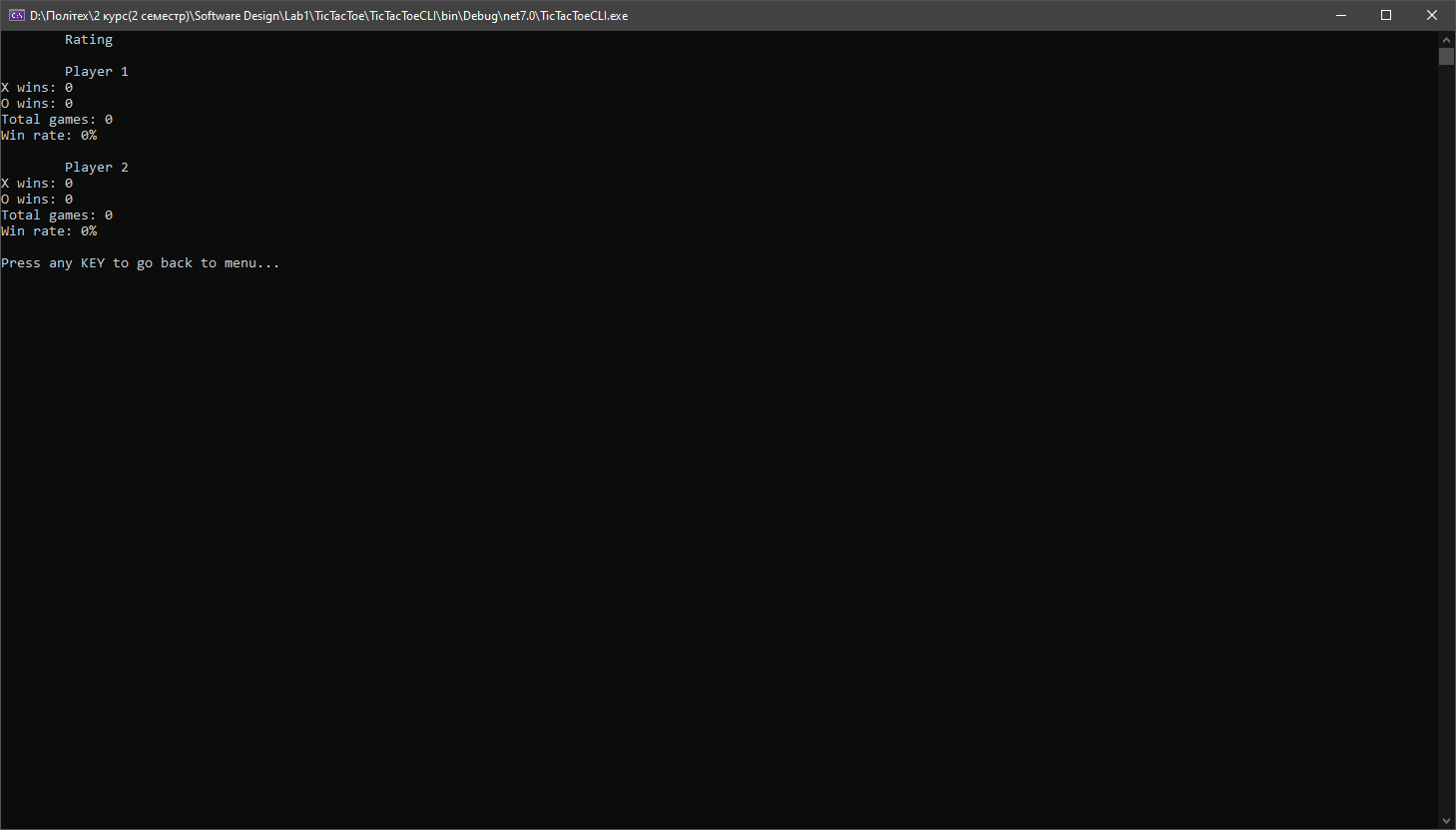
1. ***Результати програми:***
   1. ***Вибір ролей гравцями:***

******

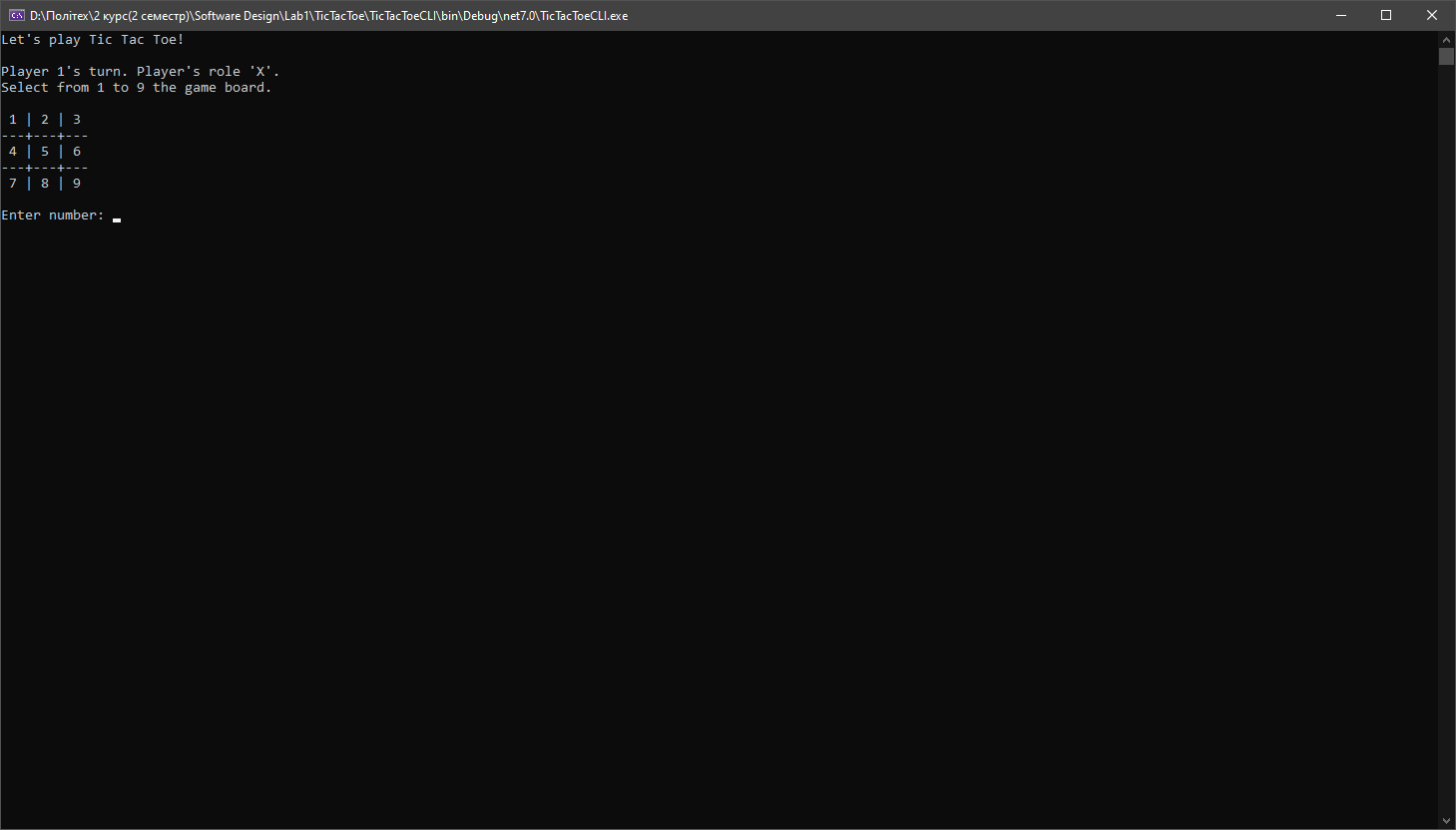
* 1. ***Головне меню гри:***

******

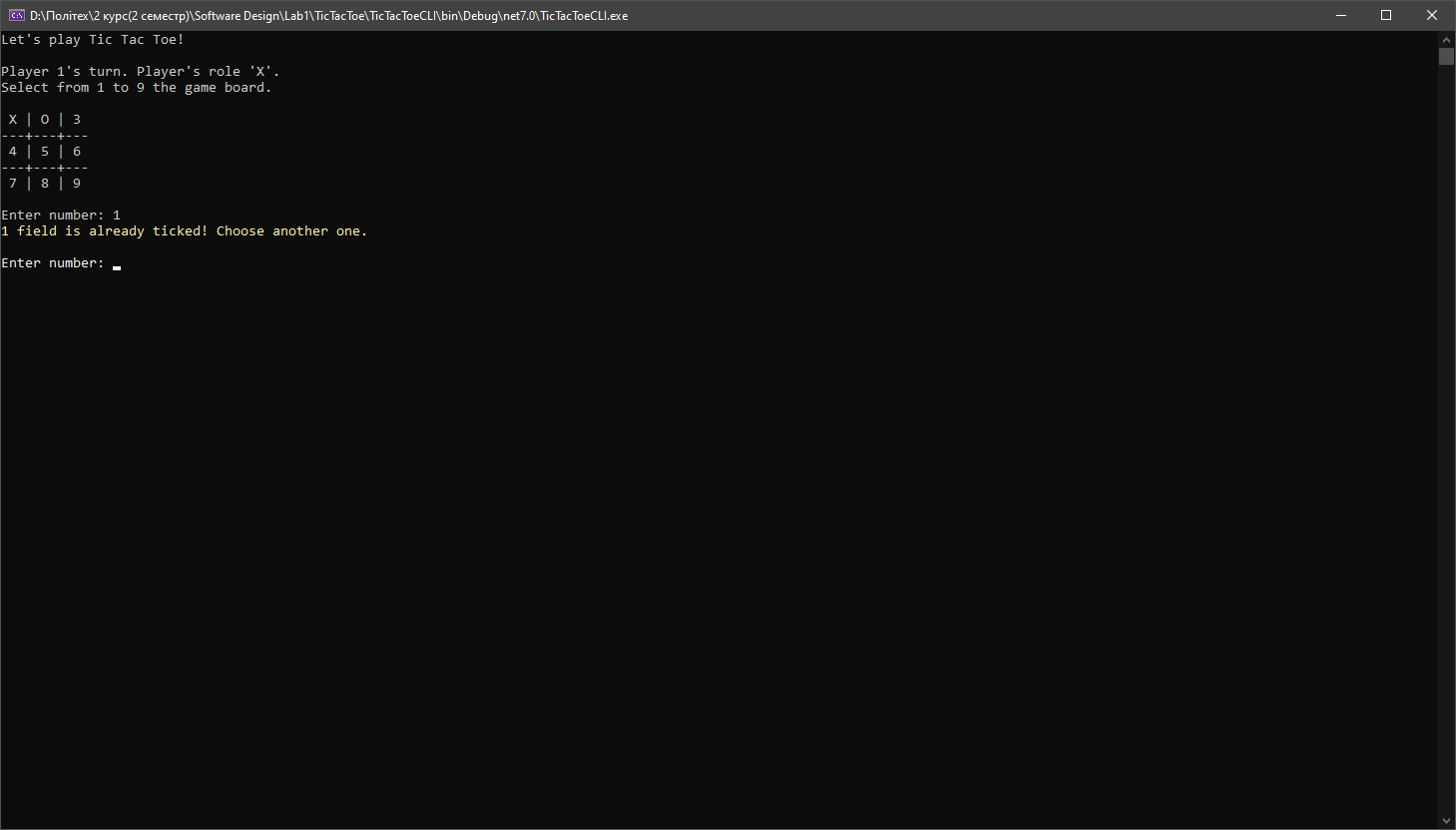
* 1. ***Меню рейтинга до початку гри:***

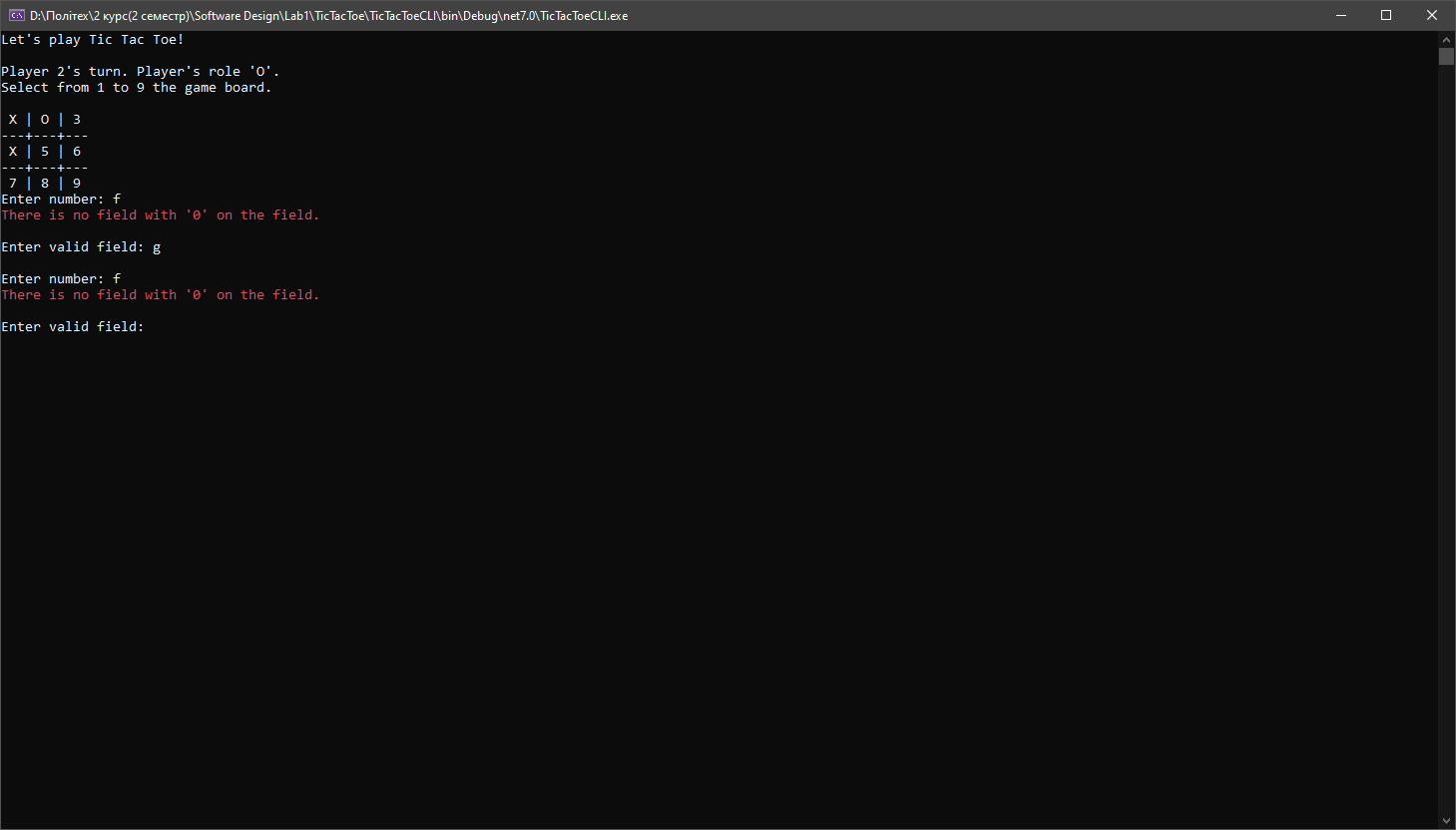
******

* 1. ***Сама гра:***

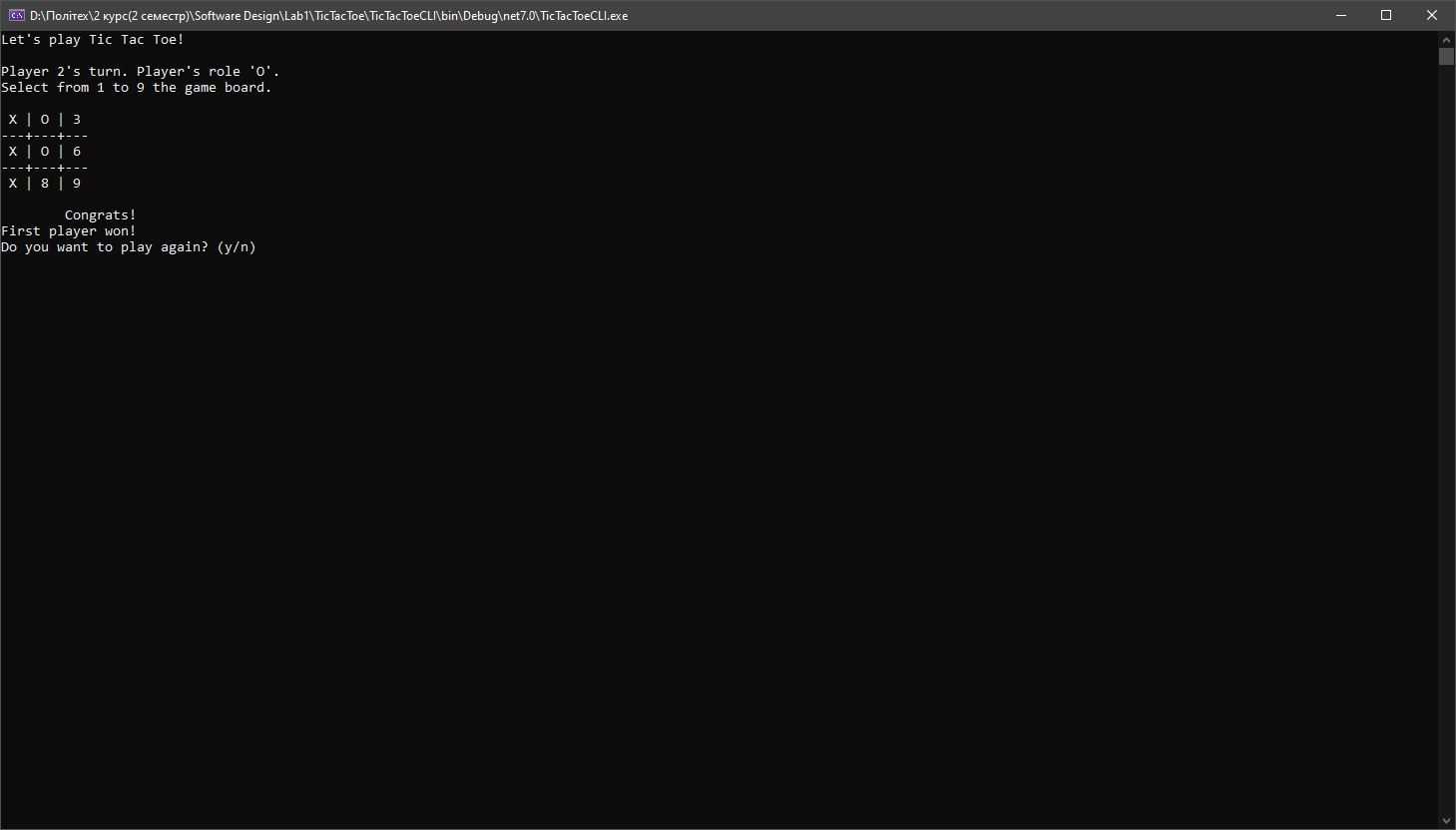
******

* 1. ***Валідація даних:***

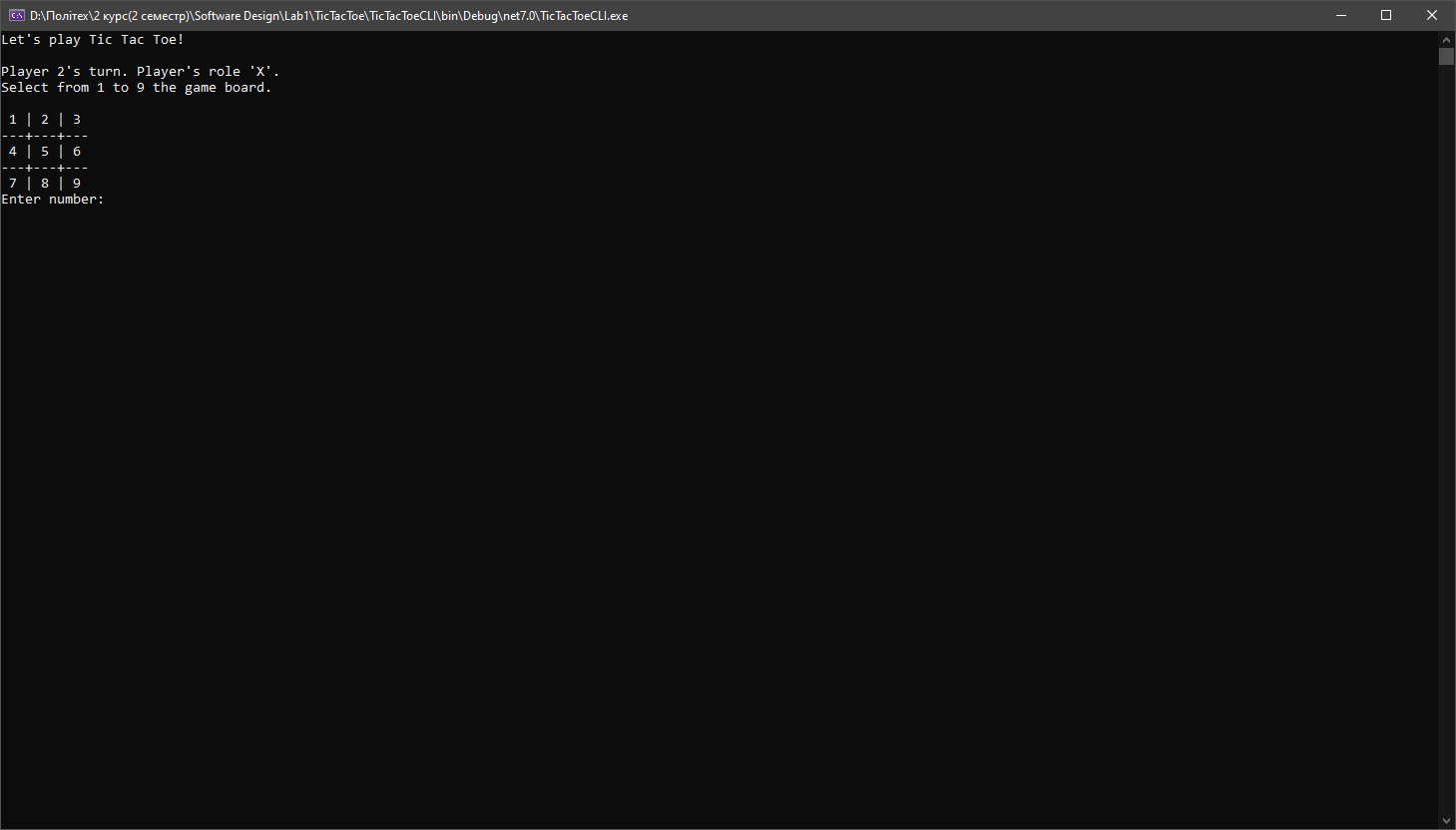
******

******

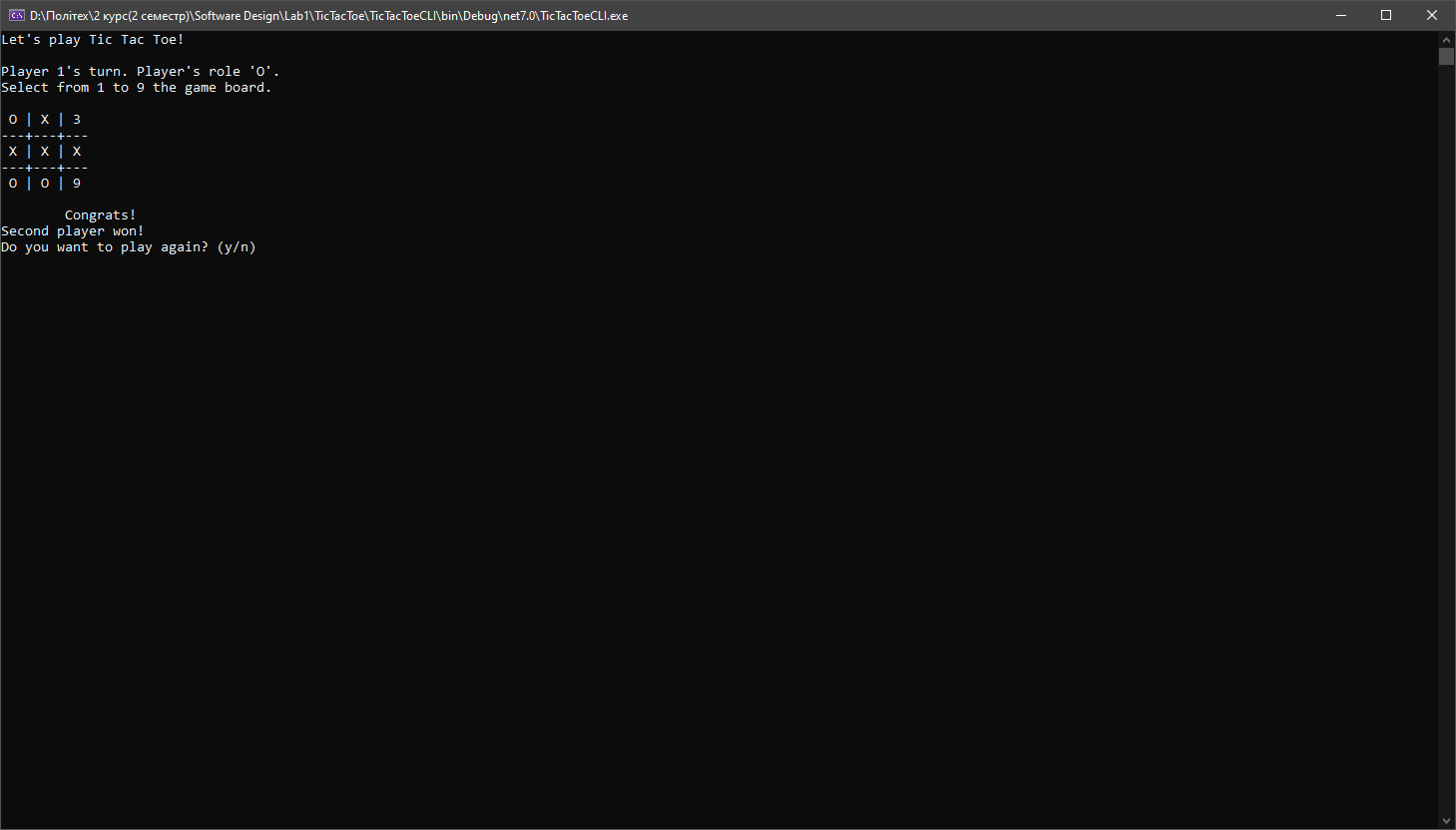
* 1. ***Перемога:***

******

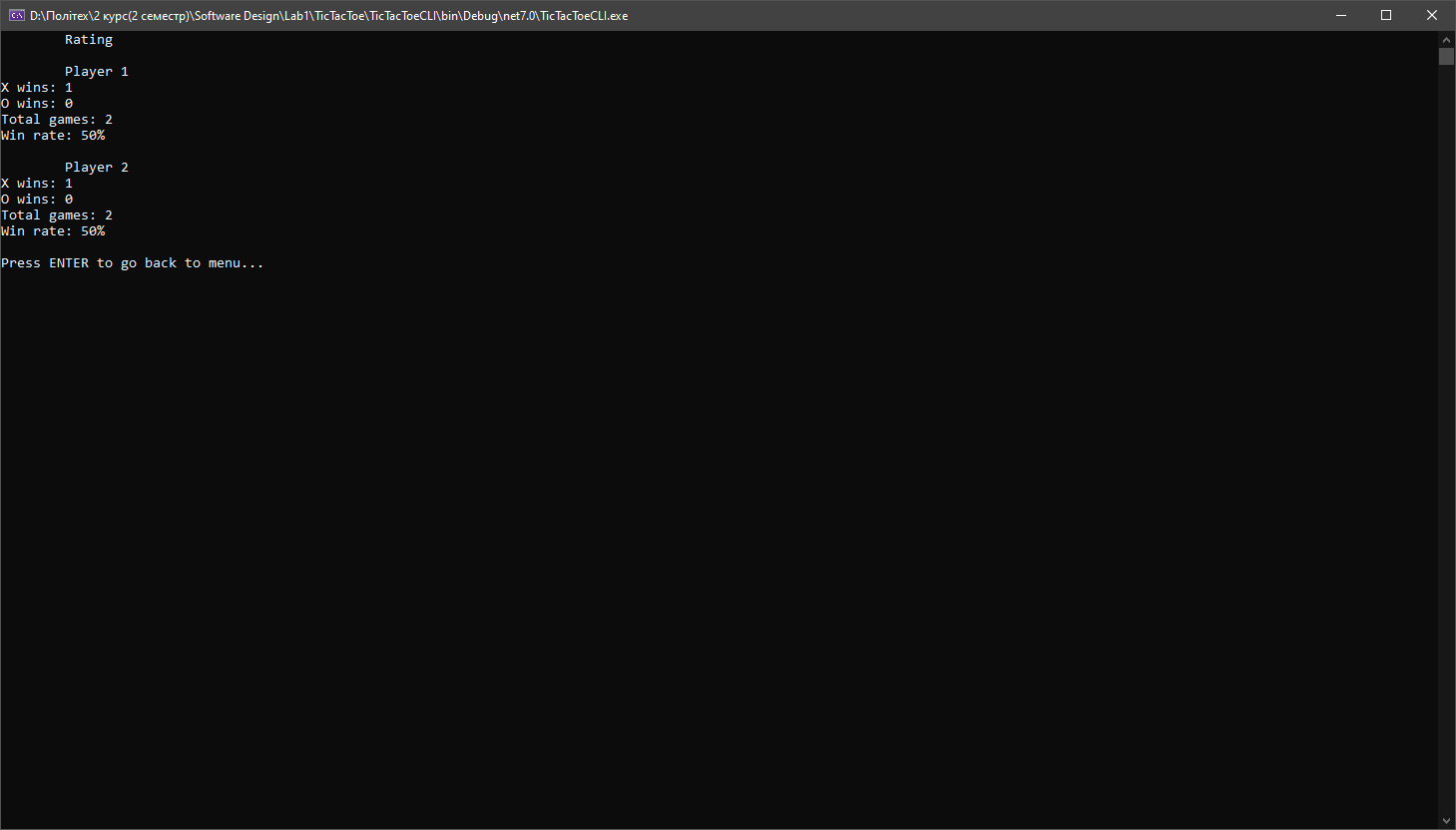
* 1. ***Повторення гри(тепер 2 гравець грає за хрестики):***

******

* 1. ***Перемога 2 гравця:***

******

* 1. ***Рейтинг гравців:***

******

***Висновок:*** під час виконання лабораторної роботи було отримано навички з використання принципів ООП в програмуванні для написання консольної гри.