**ЛАБОРАТОРНА РОБОТА № 1**

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

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

**Хід роботи:**

1. Огляд гри

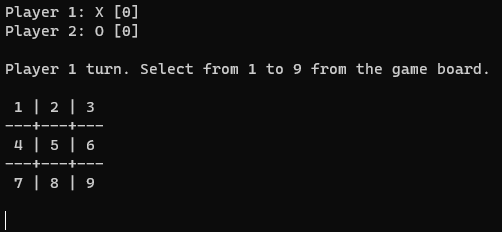


Рис. 1. Початок гри з пустою доскою

Після того як користувач вводить номер клітинки на те місце ставиться відповідний знак:

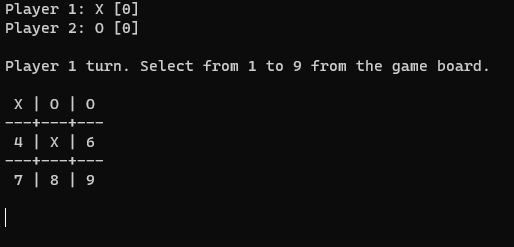


Рис. 2. Відображення доски з розставленими хрестиками та нуликами

У випадку якщо користувач введе неправильне значення або номер клітинки з’явиться помилка:

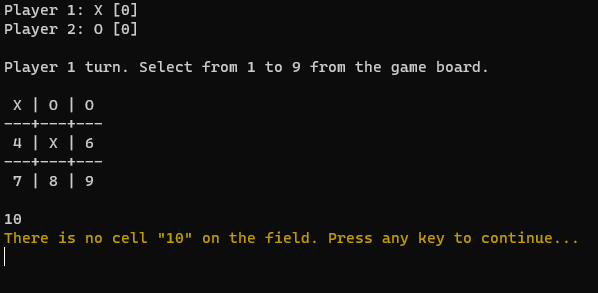


Рис. 3. Помилка про невірний номер клітинки

Якщо є переможець тоді буде виведено відповідне повідомлення та буде запропоновани зіграти ще один раунд:

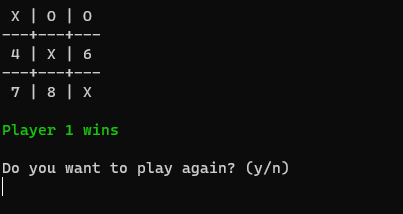


Рис. 4. Перемога

У випадку нічиєї також буде виведено повідомлення:

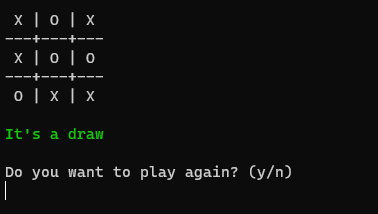


Рис. 5. Нічия

Після кожного раунду буде оновлюватись рахунок гравців, а також з кожним раундом міняється гравець що ходить перший:

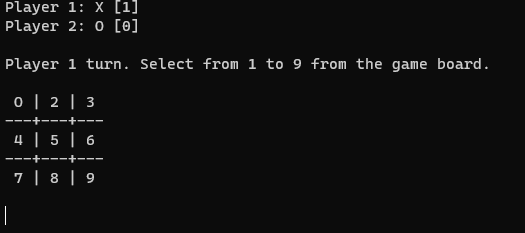


Рис. 6. Рахунок гравців

Ми можемо відмінити останній хід якщо введемо u:

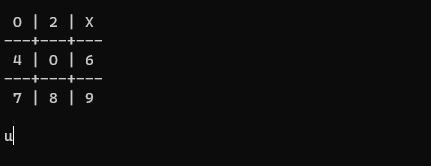


Рис. 7. Відміна останнього ходу

Якщо ще не було зроблено ні одного ходу то виведется помилка:

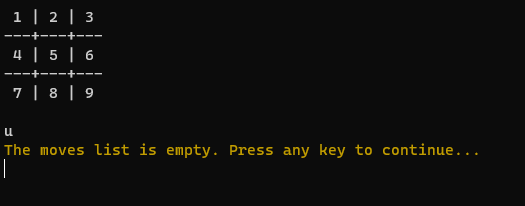


Рис. 8. Помилка при спробі відмінити останній хід

При спробі поставити знак на вж зайняту клітинку також буде виведена помилка:

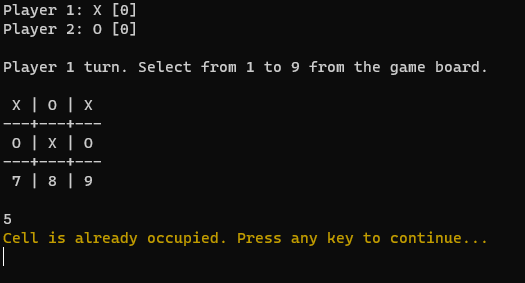


Рис. 9. Помилка при спробі поставити знак на зайняту клітинку

В кінці гри виводиться рахунок гравців:



Рис. 10. Рахунок гравців

1. Лістинг коду

Program.cs:

using lab\_1;

Game game = new Game();

game.Run();

Game.cs:

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Runtime.InteropServices;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace lab\_1

{

public class Game

{

private GameBoard \_gameBoard;

private Player player1;

private Player player2;

private char? lastWinnerChar;

private Player CurrentPlayer;

private Player FirstMovePlayer;

public Game()

{

\_gameBoard = new GameBoard();

player1 = new Player("Player 1", 'X', 0);

player2 = new Player("Player 2", 'O', 0);

CurrentPlayer = player1;

FirstMovePlayer = player1;

}

private void SwitchCurrentPlayer()

{

CurrentPlayer = CurrentPlayer == player1 ? player2 : player1;

}

private void SwitchFirstMovePlayer()

{

FirstMovePlayer = FirstMovePlayer == player1 ? player2 : player1;

CurrentPlayer = FirstMovePlayer;

}

public void Run()

{

while (true)

{

PrintGame();

if (CheckRoundEnd())

{

if (PrintAskMessage("Do you want to play again? (y/n)") == "y")

{

continue;

}

EndGame();

break;

}

try

{

Input();

}

catch (Exception e)

{

PrintYellowMessage(e.Message + ". Press any key to continue...");

}

}

}

private void Input()

{

string input = Console.ReadLine();

if (input == "u")

{

UndoMove();

}

else

{

MakeMove(int.Parse(input));

}

}

private bool CheckRoundEnd()

{

if (IsWin())

{

Player winner = lastWinnerChar == player1.PlayerChar ? player1 : player2;

winner.Score++;

PrintGreenMessage($"{winner.Name} wins");

SwitchFirstMovePlayer();

\_gameBoard = new GameBoard();

return true;

}

else if (isDraw())

{

PrintGreenMessage("It's a draw");

SwitchFirstMovePlayer();

return true;

}

return false;

}

private bool isDraw()

{

for (int i = 0; i < \_gameBoard.Board.GetLength(0); i++)

{

for (int j = 0; j < \_gameBoard.Board.GetLength(1); j++)

{

if (Char.IsDigit(\_gameBoard.Board[i, j]))

return false;

}

}

return true;

}

private bool IsWin()

{

char[,] board = \_gameBoard.Board;

int i = 1;

if (board[0, 0] == board[1, 1] && board[1, 1] == board[2, 2])

{

lastWinnerChar = board[0, 0];

return true;

}

if (board[0, 2] == board[1, 1] && board[1, 1] == board[2, 0])

{

lastWinnerChar = board[0, 2];

return true;

}

for (i = 0; i < board.GetLength(0); i++)

{

if (board[i, 0] == board[i, 1] && board[i, 1] == board[i, 2])

{

lastWinnerChar = board[i, 0];

return true;

}

}

for (i = 0; i < board.GetLength(0); i++)

{

if (board[0, i] == board[1, i] && board[1, i] == board[2, i])

{

lastWinnerChar = board[i, 0];

return true;

}

}

return false;

}

private void UndoMove()

{

\_gameBoard.UndoMove();

SwitchCurrentPlayer();

}

private void MakeMove(int position)

{

if (position <= 9 && position >= 1)

{

\_gameBoard.SetCharAt(position, CurrentPlayer.PlayerChar);

SwitchCurrentPlayer();

}

else

{

throw new Exception($"There is no cell \"{position}\" on the field");

}

}

private void PrintGame()

{

Console.Clear();

PrintScore();

Console.WriteLine($"\n{CurrentPlayer.Name} turn. Select from 1 to 9 from the game board.\n");

\_gameBoard.PrintBoard();

Console.WriteLine("\n");

}

public void PrintScore()

{

Console.WriteLine($"{player1.Name}: {player1.PlayerChar} [{player1.Score}]");

Console.WriteLine($"{player2.Name}: {player2.PlayerChar} [{player2.Score}]");

}

public void PrintGreenMessage(string message)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine(message);

Console.ForegroundColor = ConsoleColor.Gray;

Console.ReadLine();

}

public void PrintYellowMessage(string message)

{

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine(message);

Console.ForegroundColor = ConsoleColor.Gray;

Console.ReadLine();

}

public string PrintAskMessage(string message)

{

Console.WriteLine(message);

return Console.ReadLine();

}

public void EndGame()

{

Console.Clear();

Console.WriteLine("Game over");

PrintScore();

}

}

}

Player.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_1

{

public class Player

{

public string Name { get; set; }

public char PlayerChar;

public int Score;

public Player(string name, char playerChar, int score)

{

Name = name;

PlayerChar = playerChar;

Score = score;

}

}

}

GameBoard.cs:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Common;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace lab\_1

{

public class GameBoard

{

public char[,] Board { get; private set; }

private Stack<int> movesStack;

public GameBoard()

{

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

movesStack = new Stack<int>();

}

public void SetCharAt(int position, char c)

{

movesStack.Push(position);

position--;

int row = position / 3;

int column = position % 3;

if (Char.IsDigit(Board[row, column]))

{

Board[row, column] = c;

}

else

{

throw new Exception("Cell is already occupied");

}

}

public void PrintBoard()

{

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

{

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

{

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

if (j < 2)

{

Console.Write('|');

}

}

if (i < 2)

{

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

}

}

}

public void UndoMove()

{

if(movesStack.Count == 0)

{

throw new Exception("The moves list is empty");

}

int movePosition = movesStack.Pop();

movePosition--;

int row = movePosition / 3;

int column = movePosition % 3;

Board[row, column] = (++movePosition).ToString()[0];

}

}

}

***Висновки:*** в ході виконання практичної роботи було розроблено гру «Хрестики-нолики» з усім базовим функціоналом, а також реалізовано 2 додаткові фічі: відміна ходу та рахунок гравців.