МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра інженерії програмного забезпечення



Лабораторна робота №6

з дисципліни «Супроводження програмного забезпечення»

На тему: «Дослідження метода реінженерії програмного забезпечення»

Виконав студент:

групи ПІ-421Б

Іванюк Н. О.

Перевірив викладач:

Корнієнко С. П.

Київ 2022

**Мета роботи**: дослідити метод реінженерії програмного забезпечення.

Реінженерія застосовується для зміни ділових процесів, зниження кількості зайвих видів діяльності в них і підвищення ефективності окремих ділових процесів за рахунок впровадження нових програм або модифікації існуючих. Якщо бізнес-процес залежить від успадкованої системи, то зміни в ній повинні плануватися. Головна відмінність між реінженерією і новою розробкою системи полягає в тому, що опис системної специфікації починається не з «нуля», а з розгляду можливостей старої успадкованої системи.

**Завдання**

1. Вибрати програмний продукт та згідно з вказаним викладачем варіантом виконати відповідний процес інженерії.
2. Зробити скріншоти програми до реінженерії та після. Вказати зміни. Пояснити.
3. В звіт поставити код програми.
4. Зробити відповідні висновки.

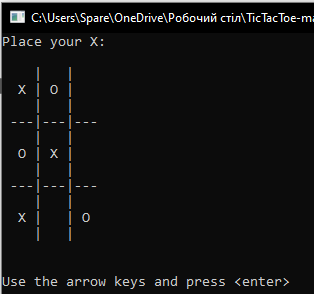
**Варіант №5**

Зміна даних, з якими працює програма.

**Виконання**

*Tic Tac Toe* або хрестики нулики – це гра з паперу та олівцем для двох гравців, які по черзі позначають пробіли в сітці три на три з X або O. Гравець, якому вдалося розмістити три свої позначки в горизонтальному, вертикальному або діагональному ряду, є переможцем. Цевирішена гра, з вимушеною нічиєю, яка передбачаєнайкращу груобох гравців.

**Програма до реінженерії**



using System;

using System.Collections.Generic;

using System.Linq;

namespace TicTacToe

{

class Program

{

static void Main(string[] args)

{

var stillPlaying = true;

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("-----------------------");

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

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

Console.ResetColor();

while (stillPlaying)

{

Console.WriteLine("What would you like to do:");

Console.WriteLine("1. Start a new game");

Console.WriteLine("2. Quit\n");

Console.Write("Type a number and hit <enter>: ");

var choice = GetUserInput("[12]");

switch(choice)

{

case "1":

PlayGame();

Console.Clear();

break;

case "2":

stillPlaying = false;

break;

}

}

}

private static string GetUserInput(string validPattern = null)

{

var input = Console.ReadLine();

input = input.Trim();

if (validPattern != null && !System.Text.RegularExpressions.Regex.IsMatch(input, validPattern))

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("\"" + input + "\" is not valid.\n");

Console.ResetColor();

return null;

}

return input;

}

private static void PlayGame()

{

string numRowsChoice = “3”;

var boardSize = (int)Math.Pow(int.Parse(numRowsChoice), 2);

var board = new string[boardSize];

var turn = "X";

while (true)

{

Console.Clear();

var winner = WhoWins(board);

if (winner != null)

{

AnnounceResult(winner[0] + " WINS!!!", board);

break;

}

if (IsBoardFull(board))

{

AnnounceResult("It's a tie!!!", board);

break;

}

Console.WriteLine("Place your " + turn + ":");

DrawBoard(board);

Console.WriteLine("\nUse the arrow keys and press <enter>");

var xoLoc = GetXOLocation(board);

board[xoLoc] = turn;

turn = turn == "X" ? "O" : "X";

}

}

private static void AnnounceResult(string message, string[] board)

{

Console.WriteLine();

DrawBoard(board);

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine(message);

Console.ResetColor();

Console.Write("\nPress any key to continue...");

Console.CursorVisible = false;

Console.ReadKey();

Console.CursorVisible = true;

}

private static int GetXOLocation(string[] board)

{

int numRows = (int)Math.Sqrt(board.Length);

int curRow = 0, curCol = 0;

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

{

if (board[i] == null)

{

curRow = i / numRows;

curCol = i % numRows;

break;

}

}

while (true)

{

Console.SetCursorPosition(curCol \* 4 + 2, curRow \* 4 + 3);

var keyInfo = Console.ReadKey();

Console.SetCursorPosition(curCol \* 4 + 2, curRow \* 4 + 3);

Console.Write(board[curRow \* numRows + curCol] ?? " ");

switch (keyInfo.Key)

{

case ConsoleKey.LeftArrow:

if (curCol > 0)

curCol--;

break;

case ConsoleKey.RightArrow:

if (curCol + 1 < numRows)

curCol++;

break;

case ConsoleKey.UpArrow:

if (curRow > 0)

curRow--;

break;

case ConsoleKey.DownArrow:

if (curRow + 1 < numRows)

curRow++;

break;

case ConsoleKey.Spacebar:

case ConsoleKey.Enter:

if (board[curRow \* numRows + curCol] == null)

return curRow \* numRows + curCol;

break;

}

}

}

private static void DrawBoard(string[] board)

{

var numRows = (int)Math.Sqrt(board.Length);

Console.WriteLine();

for (int row = 0; row < numRows; row++)

{

if (row != 0)

Console.WriteLine(" " + string.Join("|", Enumerable.Repeat("---", numRows)));

Console.Write(" " + string.Join("|", Enumerable.Repeat(" ", numRows)) + "\n ");

for (int col = 0; col < numRows; col++)

{

if (col != 0)

Console.Write("|");

var space = board[row \* numRows + col] ?? " ";

if (space.Length > 1)

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.Write(" " + space[0] + " ");

Console.ResetColor();

}

Console.WriteLine("\n " + string.Join("|", Enumerable.Repeat(" ", numRows)));

}

Console.WriteLine();

}

private static bool IsBoardFull(IEnumerable<string> board)

{

return board.All(space => space != null);

}

private static string WhoWins(string[] board)

{

var numRows = (int)Math.Sqrt(board.Length);

// Check rows

for (int row = 0; row < numRows; row++)

{

if (board[row \* numRows] != null)

{

bool hasTicTacToe = true;

for (int col = 1; col < numRows && hasTicTacToe; col++)

{

if (board[row \* numRows + col] != board[row \* numRows])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int col = 0; col < numRows; col++)

board[row \* numRows + col] += "!";

return board[row \* numRows];

}

}

}

// Check columns

for (int col = 0; col < numRows; col++)

{

if (board[col] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + col] != board[col])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + col] += "!";

return board[col];

}

}

}

// Check top left -> bottom right diagonal

if (board[0] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + row] != board[0])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + row] += "!";

return board[0];

}

}

// Check top right -> bottom left diagonal

if (board[numRows - 1] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + (numRows - 1 - row)] != board[numRows - 1])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + (numRows - 1 - row)] += "!";

return board[numRows - 1];

}

}

return null;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

namespace TicTacToe

{

class Program

{

static void Main(string[] args)

{

var stillPlaying = true;

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("-----------------------");

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

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

Console.ResetColor();

while (stillPlaying)

{

Console.WriteLine("What would you like to do:");

Console.WriteLine("1. Start a new game");

Console.WriteLine("2. Quit\n");

Console.Write("Type a number and hit <enter>: ");

var choice = GetUserInput("[12]");

switch(choice)

{

case "1":

PlayGame();

Console.Clear();

break;

case "2":

stillPlaying = false;

break;

}

}

}

private static string GetUserInput(string validPattern = null)

{

var input = Console.ReadLine();

input = input.Trim();

if (validPattern != null && !System.Text.RegularExpressions.Regex.IsMatch(input, validPattern))

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("\"" + input + "\" is not valid.\n");

Console.ResetColor();

return null;

}

return input;

}

private static void PlayGame()

{

string numRowsChoice = “3”;

var boardSize = (int)Math.Pow(int.Parse(numRowsChoice), 2);

var board = new string[boardSize];

var turn = "X";

while (true)

{

Console.Clear();

var winner = WhoWins(board);

if (winner != null)

{

AnnounceResult(winner[0] + " WINS!!!", board);

break;

}

if (IsBoardFull(board))

{

AnnounceResult("It's a tie!!!", board);

break;

}

Console.WriteLine("Place your " + turn + ":");

DrawBoard(board);

Console.WriteLine("\nUse the arrow keys and press <enter>");

var xoLoc = GetXOLocation(board);

board[xoLoc] = turn;

turn = turn == "X" ? "O" : "X";

}

}

private static void AnnounceResult(string message, string[] board)

{

Console.WriteLine();

DrawBoard(board);

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine(message);

Console.ResetColor();

Console.Write("\nPress any key to continue...");

Console.CursorVisible = false;

Console.ReadKey();

Console.CursorVisible = true;

}

private static int GetXOLocation(string[] board)

{

int numRows = (int)Math.Sqrt(board.Length);

int curRow = 0, curCol = 0;

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

{

if (board[i] == null)

{

curRow = i / numRows;

curCol = i % numRows;

break;

}

}

while (true)

{

Console.SetCursorPosition(curCol \* 4 + 2, curRow \* 4 + 3);

var keyInfo = Console.ReadKey();

Console.SetCursorPosition(curCol \* 4 + 2, curRow \* 4 + 3);

Console.Write(board[curRow \* numRows + curCol] ?? " ");

switch (keyInfo.Key)

{

case ConsoleKey.LeftArrow:

if (curCol > 0)

curCol--;

break;

case ConsoleKey.RightArrow:

if (curCol + 1 < numRows)

curCol++;

break;

case ConsoleKey.UpArrow:

if (curRow > 0)

curRow--;

break;

case ConsoleKey.DownArrow:

if (curRow + 1 < numRows)

curRow++;

break;

case ConsoleKey.Spacebar:

case ConsoleKey.Enter:

if (board[curRow \* numRows + curCol] == null)

return curRow \* numRows + curCol;

break;

}

}

}

private static void DrawBoard(string[] board)

{

var numRows = (int)Math.Sqrt(board.Length);

Console.WriteLine();

for (int row = 0; row < numRows; row++)

{

if (row != 0)

Console.WriteLine(" " + string.Join("|", Enumerable.Repeat("---", numRows)));

Console.Write(" " + string.Join("|", Enumerable.Repeat(" ", numRows)) + "\n ");

for (int col = 0; col < numRows; col++)

{

if (col != 0)

Console.Write("|");

var space = board[row \* numRows + col] ?? " ";

if (space.Length > 1)

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.Write(" " + space[0] + " ");

Console.ResetColor();

}

Console.WriteLine("\n " + string.Join("|", Enumerable.Repeat(" ", numRows)));

}

Console.WriteLine();

}

private static bool IsBoardFull(IEnumerable<string> board)

{

return board.All(space => space != null);

}

private static string WhoWins(string[] board)

{

var numRows = (int)Math.Sqrt(board.Length);

// Check rows

for (int row = 0; row < numRows; row++)

{

if (board[row \* numRows] != null)

{

bool hasTicTacToe = true;

for (int col = 1; col < numRows && hasTicTacToe; col++)

{

if (board[row \* numRows + col] != board[row \* numRows])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int col = 0; col < numRows; col++)

board[row \* numRows + col] += "!";

return board[row \* numRows];

}

}

}

// Check columns

for (int col = 0; col < numRows; col++)

{

if (board[col] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + col] != board[col])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + col] += "!";

return board[col];

}

}

}

// Check top left -> bottom right diagonal

if (board[0] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + row] != board[0])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + row] += "!";

return board[0];

}

}

// Check top right -> bottom left diagonal

if (board[numRows - 1] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + (numRows - 1 - row)] != board[numRows - 1])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + (numRows - 1 - row)] += "!";

return board[numRows - 1];

}

}

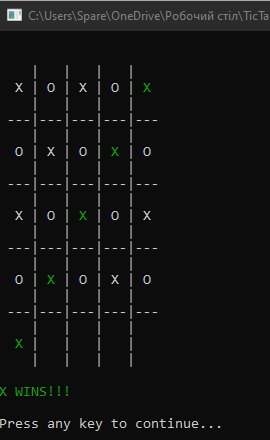
return null;

}

}

}

**Програма після реінженерії**

****

using System;

using System.Collections.Generic;

using System.Linq;

namespace TicTacToe

{

class Program

{

static void Main(string[] args)

{

var stillPlaying = true;

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("-----------------------");

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

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

Console.ResetColor();

while (stillPlaying)

{

Console.WriteLine("What would you like to do:");

Console.WriteLine("1. Start a new game");

Console.WriteLine("2. Quit\n");

Console.Write("Type a number and hit <enter>: ");

var choice = GetUserInput("[12]");

switch(choice)

{

case "1":

PlayGame();

Console.Clear();

break;

case "2":

stillPlaying = false;

break;

}

}

}

private static string GetUserInput(string validPattern = null)

{

var input = Console.ReadLine();

input = input.Trim();

if (validPattern != null && !System.Text.RegularExpressions.Regex.IsMatch(input, validPattern))

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("\"" + input + "\" is not valid.\n");

Console.ResetColor();

return null;

}

return input;

}

private static void PlayGame()

{

string numRowsChoice = null;

while (numRowsChoice == null)

{

Console.Write("How many rows do you want to have? (3, 4, or 5) ");

numRowsChoice = GetUserInput("[345]");

}

var boardSize = (int)Math.Pow(int.Parse(numRowsChoice), 2);

var board = new string[boardSize];

var turn = "X";

while (true)

{

Console.Clear();

var winner = WhoWins(board);

if (winner != null)

{

AnnounceResult(winner[0] + " WINS!!!", board);

break;

}

if (IsBoardFull(board))

{

AnnounceResult("It's a tie!!!", board);

break;

}

Console.WriteLine("Place your " + turn + ":");

DrawBoard(board);

Console.WriteLine("\nUse the arrow keys and press <enter>");

var xoLoc = GetXOLocation(board);

board[xoLoc] = turn;

turn = turn == "X" ? "O" : "X";

}

}

private static void AnnounceResult(string message, string[] board)

{

Console.WriteLine();

DrawBoard(board);

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine(message);

Console.ResetColor();

Console.Write("\nPress any key to continue...");

Console.CursorVisible = false;

Console.ReadKey();

Console.CursorVisible = true;

}

private static int GetXOLocation(string[] board)

{

int numRows = (int)Math.Sqrt(board.Length);

int curRow = 0, curCol = 0;

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

{

if (board[i] == null)

{

curRow = i / numRows;

curCol = i % numRows;

break;

}

}

while (true)

{

Console.SetCursorPosition(curCol \* 4 + 2, curRow \* 4 + 3);

var keyInfo = Console.ReadKey();

Console.SetCursorPosition(curCol \* 4 + 2, curRow \* 4 + 3);

Console.Write(board[curRow \* numRows + curCol] ?? " ");

switch (keyInfo.Key)

{

case ConsoleKey.LeftArrow:

if (curCol > 0)

curCol--;

break;

case ConsoleKey.RightArrow:

if (curCol + 1 < numRows)

curCol++;

break;

case ConsoleKey.UpArrow:

if (curRow > 0)

curRow--;

break;

case ConsoleKey.DownArrow:

if (curRow + 1 < numRows)

curRow++;

break;

case ConsoleKey.Spacebar:

case ConsoleKey.Enter:

if (board[curRow \* numRows + curCol] == null)

return curRow \* numRows + curCol;

break;

}

}

}

private static void DrawBoard(string[] board)

{

var numRows = (int)Math.Sqrt(board.Length);

Console.WriteLine();

for (int row = 0; row < numRows; row++)

{

if (row != 0)

Console.WriteLine(" " + string.Join("|", Enumerable.Repeat("---", numRows)));

Console.Write(" " + string.Join("|", Enumerable.Repeat(" ", numRows)) + "\n ");

for (int col = 0; col < numRows; col++)

{

if (col != 0)

Console.Write("|");

var space = board[row \* numRows + col] ?? " ";

if (space.Length > 1)

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.Write(" " + space[0] + " ");

Console.ResetColor();

}

Console.WriteLine("\n " + string.Join("|", Enumerable.Repeat(" ", numRows)));

}

Console.WriteLine();

}

private static bool IsBoardFull(IEnumerable<string> board)

{

return board.All(space => space != null);

}

private static string WhoWins(string[] board)

{

var numRows = (int)Math.Sqrt(board.Length);

// Check rows

for (int row = 0; row < numRows; row++)

{

if (board[row \* numRows] != null)

{

bool hasTicTacToe = true;

for (int col = 1; col < numRows && hasTicTacToe; col++)

{

if (board[row \* numRows + col] != board[row \* numRows])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int col = 0; col < numRows; col++)

board[row \* numRows + col] += "!";

return board[row \* numRows];

}

}

}

// Check columns

for (int col = 0; col < numRows; col++)

{

if (board[col] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + col] != board[col])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + col] += "!";

return board[col];

}

}

}

// Check top left -> bottom right diagonal

if (board[0] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + row] != board[0])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + row] += "!";

return board[0];

}

}

// Check top right -> bottom left diagonal

if (board[numRows - 1] != null)

{

bool hasTicTacToe = true;

for (int row = 1; row < numRows && hasTicTacToe; row++)

{

if (board[row \* numRows + (numRows - 1 - row)] != board[numRows - 1])

hasTicTacToe = false;

}

if (hasTicTacToe)

{

// Put an indicator in the board to know which ones are part of the tic tac toe

for (int row = 0; row < numRows; row++)

board[row \* numRows + (numRows - 1 - row)] += "!";

return board[numRows - 1];

}

}

return null;

}

}

}

До зміни були приведені деякілька строк коду які відповідали за розмірність таблиці, а точніше в першій версії кожного разу створувалась таблиця 3x3 а в наступній можна було орати розмірність між 3x3, 4x4 та 5x5.

До зміни були приведений наступний метод

До зміни

private static void PlayGame()

{

string numRowsChoice = “3”;

var boardSize = (int)Math.Pow(int.Parse(numRowsChoice), 2);

var board = new string[boardSize];

var turn = "X";

while (true)

{

Console.Clear();

var winner = WhoWins(board);

if (winner != null)

{

AnnounceResult(winner[0] + " WINS!!!", board);

break;

}

if (IsBoardFull(board))

{

AnnounceResult("It's a tie!!!", board);

break;

}

Console.WriteLine("Place your " + turn + ":");

DrawBoard(board);

Console.WriteLine("\nUse the arrow keys and press <enter>");

var xoLoc = GetXOLocation(board);

board[xoLoc] = turn;

turn = turn == "X" ? "O" : "X";

}

}

Після зміни

private static void PlayGame()

{

string numRowsChoice = null;

while (numRowsChoice == null)

{

Console.Write("How many rows do you want to have? (3, 4, or 5) ");

numRowsChoice = GetUserInput("[345]");

}

var boardSize = (int)Math.Pow(int.Parse(numRowsChoice), 2);

var board = new string[boardSize];

var turn = "X";

while (true)

{

Console.Clear();

var winner = WhoWins(board);

if (winner != null)

{

AnnounceResult(winner[0] + " WINS!!!", board);

break;

}

if (IsBoardFull(board))

{

AnnounceResult("It's a tie!!!", board);

break;

}

Console.WriteLine("Place your " + turn + ":");

DrawBoard(board);

Console.WriteLine("\nUse the arrow keys and press <enter>");

var xoLoc = GetXOLocation(board);

board[xoLoc] = turn;

turn = turn == "X" ? "O" : "X";

}

}

**Висновки:** дослідив метод реінженерії програмного забезпечення. На практиці змінив данні, з якими працює програма