Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра СКС



**Звіт**

з лабораторної роботи №1

з дисципліни: “Автоматизоване проектування комп'ютерних систем”

завдання № 4, гра шашки

Виконав:

ст.гр. КІ-41з

Діхтярук Валерій Валерійович

Прийняв: викладач

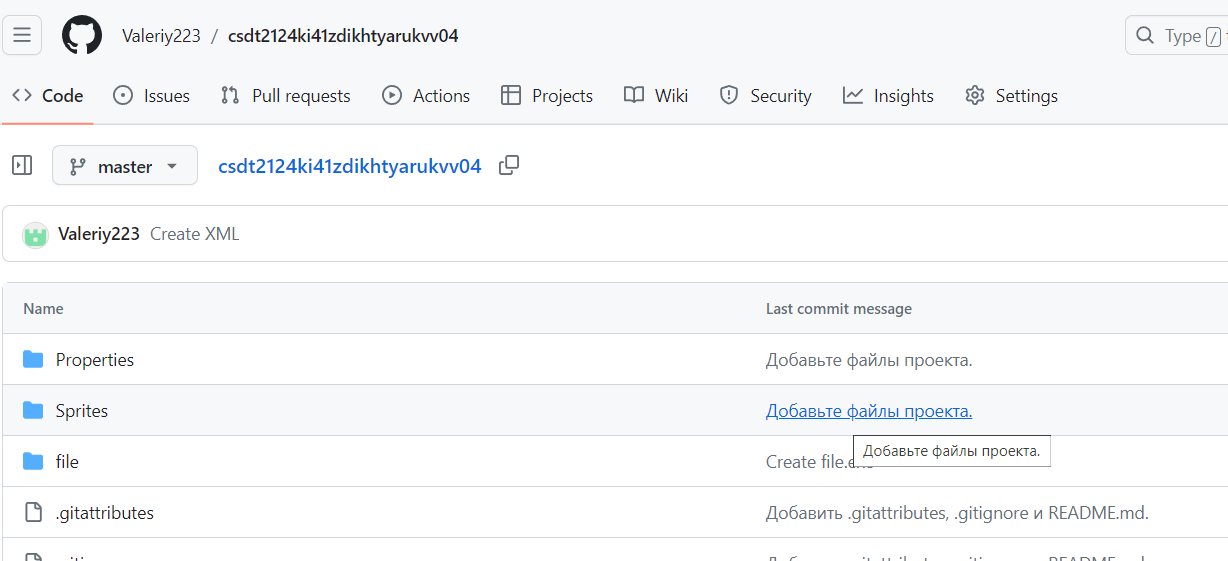
[Шпіцер](https://vns.lpnu.ua/user/view.php?id=54806&course=1) А.С.

Львів – 2023

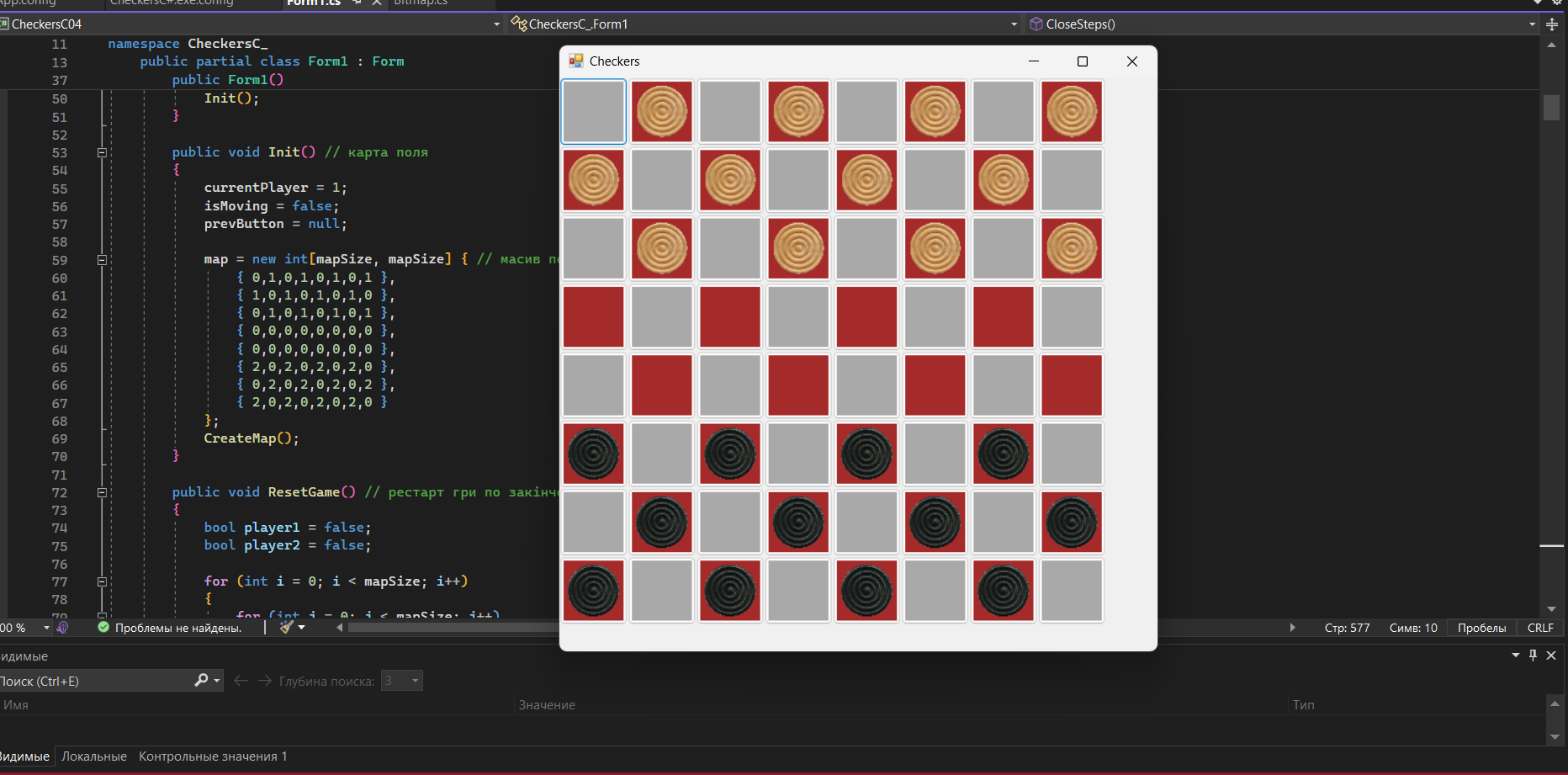
**Мета роботи:** спроектувати гру згідно з завданням, створити репозиторій та викласти проект в доступ.

***Task 1. Initiate GIT repository:***

1. Create a github repo with the namе csdt2124ki41zdikhtyarukvv04
2. Створений GIT TAGcheckers С#\_1\_01\_ ww20230103
3. Створений репозиторій на гітхаб



1. Розроблений код взятий на основі кода шахмат



1. Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CheckersC\_

{

public partial class Form1 : Form

{

const int mapSize = 8; // розмір карти

const int cellSize = 65; // розмір клітки

int currentPlayer; // змінна, яка зберігає поточного гравця

List<Button> simpleSteps = new List<Button>();

int countEatSteps = 0; // кількість можливих вірних ходів

Button prevButton; // запис минулої нажатої кніпки

Button pressedButton;

bool isContinue = false; // контролює можливість наступного ходу однією шашкою

bool isMoving; // відслідкування шашки в процесі ходьби

int[,] map = new int[mapSize, mapSize];

Button[,] buttons = new Button[mapSize, mapSize];

Image whiteFigure; // модель білої шашки

Image blackFigure; // модель чорної шашки

public Form1()

{

InitializeComponent();

whiteFigure = new Bitmap(new Bitmap

(@"D:\Documents\Visual Studio 2022\CheckersC#\Sprites\w.png"),

new Size(cellSize - 15, cellSize - 15)); // путь до спрайту білої шашки

blackFigure = new Bitmap(new Bitmap

(@"D:\Documents\Visual Studio 2022\CheckersC#\Sprites\b.png"),

new Size(cellSize - 15, cellSize - 15)); // путь до спрайту білої шашки

this.Text = "Checkers";

Init();

}

public void Init() // карта поля

{

currentPlayer = 1;

isMoving = false;

prevButton = null;

map = new int[mapSize, mapSize] { // масив поля розташування шашок згідно правил гри

{ 0,1,0,1,0,1,0,1 },

{ 1,0,1,0,1,0,1,0 },

{ 0,1,0,1,0,1,0,1 },

{ 0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0 },

{ 2,0,2,0,2,0,2,0 },

{ 0,2,0,2,0,2,0,2 },

{ 2,0,2,0,2,0,2,0 }

};

CreateMap();

}

public void ResetGame() // рестарт гри по закінченню шашок

{

bool player1 = false;

bool player2 = false;

for (int i = 0; i < mapSize; i++)

{

for (int j = 0; j < mapSize; j++)

{

if (map[i, j] == 1)

player1 = true;

if (map[i, j] == 2)

player2 = true;

}

}

if (!player1 || !player2)

{

this.Controls.Clear();

Init();

}

}

public void CreateMap()

{

this.Width = (mapSize + 1) \* cellSize; // висота

this.Height = (mapSize + 1) \* cellSize; // ширина

for (int i = 0; i < mapSize; i++) // поле для гри в шахи

{

for (int j = 0; j < mapSize; j++)

{

Button button = new Button();

button.Location = new Point(j \* cellSize, i \* cellSize);

// зміна позиції кнопки згідно з індексами карти

button.Size = new Size(cellSize, cellSize); // розмір кніпки

button.Click += new EventHandler(OnFigurePress); // обробник подій для кожної кніпки

if (map[i, j] == 1) // перевірка вмісту клітинки 1 або 2 і відповідно встановлення спрайту

button.Image = whiteFigure;

else if (map[i, j] == 2) button.Image = blackFigure;

button.BackColor = GetPrevButtonColor(button);

button.ForeColor = Color.Blue; // підсвітка дамки

buttons[i, j] = button;

this.Controls.Add(button);

}

}

}

public void SwitchPlayer()

{

currentPlayer = currentPlayer == 1 ? 2 : 1; // обробка зміни гравця

ResetGame();

}

public Color GetPrevButtonColor(Button prevButton)

{

if ((prevButton.Location.Y / cellSize % 2) != 0)

{

if ((prevButton.Location.X / cellSize % 2) == 0)

{

return Color.Brown;

}

}

if ((prevButton.Location.Y / cellSize) % 2 == 0)

{

if ((prevButton.Location.X / cellSize) % 2 != 0)

{

return Color.Brown;

}

}

return Color.DarkGray; // підсвітка заднього фону

}

public void OnFigurePress(object sender, EventArgs e)

{

if (prevButton != null)

prevButton.BackColor = GetPrevButtonColor(prevButton);

pressedButton = sender as Button;

if (map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] != 0 && map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] == currentPlayer)

{

CloseSteps();

pressedButton.BackColor = Color.Green; // підсвітка натиснутої кніпки

DeactivateAllButtons();

pressedButton.Enabled = true;

countEatSteps = 0;

if (pressedButton.Text == "D")

ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize, false);

else ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize);

if (isMoving)

{

CloseSteps();

pressedButton.BackColor = GetPrevButtonColor(pressedButton);

ShowPossibleSteps();

isMoving = false;

}

else

isMoving = true;

}

else

{

if (isMoving)

{

isContinue = false;

if (Math.Abs(pressedButton.Location.X / cellSize - prevButton.Location.X / cellSize) > 1)

{

isContinue = true;

DeleteEaten(pressedButton, prevButton);

}

int temp = map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize];

// заміна кніпок на карті

map[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] = map[prevButton.Location.Y / cellSize, prevButton.Location.X / cellSize];

map[prevButton.Location.Y / cellSize, prevButton.Location.X / cellSize] = temp;

pressedButton.Image = prevButton.Image;

prevButton.Image = null;

pressedButton.Text = prevButton.Text;

prevButton.Text = "";

SwitchButtonToCheat(pressedButton);

countEatSteps = 0;

isMoving = false;

CloseSteps();

DeactivateAllButtons();

if (pressedButton.Text == "D")

ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize, false);

else ShowSteps(pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize);

if (countEatSteps == 0 || !isContinue)

{

CloseSteps();

SwitchPlayer();

ShowPossibleSteps();

isContinue = false;

}

else if (isContinue)

{

pressedButton.BackColor = Color.Green;

pressedButton.Enabled = true;

isMoving = true;

}

}

}

prevButton = pressedButton;

}

public void ShowPossibleSteps() // показує можливі ходи шашки

{

bool isOneStep = true; // перевіряє дамка або шашка

bool isEatStep = false; // перевіряє доступність хода захвату

DeactivateAllButtons();

for (int i = 0; i < mapSize; i++)

{

for (int j = 0; j < mapSize; j++)

{

if (map[i, j] == currentPlayer)

{

if (buttons[i, j].Text == "D")

isOneStep = false;

else isOneStep = true;

if (IsButtonHasEatStep(i, j, isOneStep, new int[2] { 0, 0 }))

{

isEatStep = true;

buttons[i, j].Enabled = true;

}

}

}

}

if (!isEatStep)

ActivateAllButtons();

}

public void SwitchButtonToCheat(Button button) // перевірка позиції при якій шашка стає дамкою

{

if (map[button.Location.Y / cellSize, button.Location.X / cellSize] == 1 && button.Location.Y / cellSize == mapSize - 1)

{

button.Text = "D";

}

if (map[button.Location.Y / cellSize, button.Location.X / cellSize] == 2 && button.Location.Y / cellSize == 0)

{

button.Text = "D";

}

}

public void DeleteEaten(Button endButton, Button startButton) // функція видаляє захоплену шашку

{

int count = Math.Abs(endButton.Location.Y / cellSize - startButton.Location.Y / cellSize);

int startIndexX = endButton.Location.Y / cellSize - startButton.Location.Y / cellSize;

int startIndexY = endButton.Location.X / cellSize - startButton.Location.X / cellSize;

startIndexX = startIndexX < 0 ? -1 : 1;

startIndexY = startIndexY < 0 ? -1 : 1;

int currCount = 0;

int i = startButton.Location.Y / cellSize + startIndexX;

int j = startButton.Location.X / cellSize + startIndexY;

while (currCount < count - 1)

{

map[i, j] = 0;

buttons[i, j].Image = null;

buttons[i, j].Text = "";

i += startIndexX;

j += startIndexY;

currCount++;

}

}

public void ShowSteps(int iCurrFigure, int jCurrFigure, bool isOnestep = true)

{

simpleSteps.Clear();

ShowDiagonal(iCurrFigure, jCurrFigure, isOnestep);

if (countEatSteps > 0) //залишає лише ходи для захвату

CloseSimpleSteps(simpleSteps);

}

public void ShowDiagonal(int IcurrFigure, int JcurrFigure, bool isOneStep = false) // алгоритм рахунку ходів

{

int j = JcurrFigure + 1;

for (int i = IcurrFigure - 1; i >= 0; i--)

{

if (currentPlayer == 1 && isOneStep && !isContinue) break;

if (IsInsideBorders(i, j))

{

if (!DeterminePath(i, j))

break;

}

if (j < 7)

j++;

else break;

if (isOneStep)

break;

}

j = JcurrFigure - 1;

for (int i = IcurrFigure - 1; i >= 0; i--)

{

if (currentPlayer == 1 && isOneStep && !isContinue) break;

if (IsInsideBorders(i, j))

{

if (!DeterminePath(i, j))

break;

}

if (j > 0)

j--;

else break;

if (isOneStep)

break;

}

j = JcurrFigure - 1;

for (int i = IcurrFigure + 1; i < 8; i++)

{

if (currentPlayer == 2 && isOneStep && !isContinue) break;

if (IsInsideBorders(i, j))

{

if (!DeterminePath(i, j))

break;

}

if (j > 0)

j--;

else break;

if (isOneStep)

break;

}

j = JcurrFigure + 1;

for (int i = IcurrFigure + 1; i < 8; i++)

{

if (currentPlayer == 2 && isOneStep && !isContinue) break;

if (IsInsideBorders(i, j))

{

if (!DeterminePath(i, j))

break;

}

if (j < 7)

j++;

else break;

if (isOneStep)

break;

}

}

public bool DeterminePath(int ti, int tj)

{

if (map[ti, tj] == 0 && !isContinue)

{

buttons[ti, tj].BackColor = Color.Yellow;

buttons[ti, tj].Enabled = true;

simpleSteps.Add(buttons[ti, tj]); // глобальний масив для запису початкових шагів

}

else

{

if (map[ti, tj] != currentPlayer)

{

if (pressedButton.Text == "D") // перевірка пешки на дамку

ShowProceduralEat(ti, tj, false);

else ShowProceduralEat(ti, tj);

}

return false;

}

return true;

}

public void CloseSimpleSteps(List<Button> simpleSteps)

{

if (simpleSteps.Count > 0)

{

for (int i = 0; i < simpleSteps.Count; i++)

{

simpleSteps[i].BackColor = GetPrevButtonColor(simpleSteps[i]);

simpleSteps[i].Enabled = false;

}

}

}

public void ShowProceduralEat(int i, int j, bool isOneStep = true)

// побудова наступного кроку після захвату шашки

{

int dirX = i - pressedButton.Location.Y / cellSize; // координати зробленого ходу осі х

int dirY = j - pressedButton.Location.X / cellSize; // координати зробленого ходу осі у

dirX = dirX < 0 ? -1 : 1;

dirY = dirY < 0 ? -1 : 1;

int il = i;

int jl = j;

bool isEmpty = true; // можливість наступного ходу

while (IsInsideBorders(il, jl))

{

if (map[il, jl] != 0 && map[il, jl] != currentPlayer)

{

isEmpty = false;

break;

}

il += dirX;

jl += dirY;

if (isOneStep)

break;

}

if (isEmpty)

return;

List<Button> toClose = new List<Button>(); // створення листа з кніпками, які потрібно вимкнути

bool closeSimple = false;

int ik = il + dirX;

int jk = jl + dirY;

while (IsInsideBorders(ik, jk))

{

if (map[ik, jk] == 0) //перевірка можливості здійснення ходу

{

if (IsButtonHasEatStep(ik, jk, isOneStep, new int[2] { dirX, dirY }))

//перевірка можливості наступного ходу

{

closeSimple = true; // закрити неможливі шаги

}

else

{

toClose.Add(buttons[ik, jk]);

}

buttons[ik, jk].BackColor = Color.Yellow; // зміна кольору на жовтий

buttons[ik, jk].Enabled = true;

countEatSteps++; // збільшуємо кількість шагів на один

}

else break;

if (isOneStep)

break;

jk += dirY;

ik += dirX;

}

if (closeSimple && toClose.Count > 0)

{

CloseSimpleSteps(toClose);

}

}

public bool IsButtonHasEatStep(int IcurrFigure, int JcurrFigure, bool isOneStep, int[] dir)

// перевірка ходу на вірність

{

bool eatStep = false;

int j = JcurrFigure + 1;

for (int i = IcurrFigure - 1; i >= 0; i--) // перевірка ходу в гору

{

if (currentPlayer == 1 && isOneStep && !isContinue) break; // для першого гравця

// обмеження першого ходу шашки в гору, якщо не дамка, то тільки в низ

if (dir[0] == 1 && dir[1] == -1 && !isOneStep) break; // перевірка ходу вниз і в ліво

if (IsInsideBorders(i, j)) // перевірка на можливість захоплення шашки

{

if (map[i, j] != 0 && map[i, j] != currentPlayer)

{

eatStep = true;

if (!IsInsideBorders(i - 1, j + 1))

eatStep = false;

else if (map[i - 1, j + 1] != 0) // перевірка можливості порожньої клітинки за шашкою

eatStep = false;

else return eatStep;

}

}

if (j < 7)

j++; // рух в право

else break;

if (isOneStep)

break;

}

j = JcurrFigure - 1;

for (int i = IcurrFigure - 1; i >= 0; i--) // перевірка ходу в низ

{

if (currentPlayer == 1 && isOneStep && !isContinue) break; // для першого гравця

// обмеження першого ходу шашки в гору, якщо не дамка, то тільки в низ

if (dir[0] == 1 && dir[1] == 1 && !isOneStep) break; // перевірка ходу вниз і в право

if (IsInsideBorders(i, j)) // перевірка на можливість захоплення шашки

{

if (map[i, j] != 0 && map[i, j] != currentPlayer)

{

eatStep = true;

if (!IsInsideBorders(i - 1, j - 1))

eatStep = false;

else if (map[i - 1, j - 1] != 0) // перевірка можливості порожньої клітинки за шашкою

eatStep = false;

else return eatStep;

}

}

if (j > 0)

j--; // рух в ліво

else break;

if (isOneStep)

break;

}

j = JcurrFigure - 1;

for (int i = IcurrFigure + 1; i < 8; i++) //перевірка ходу

{

if (currentPlayer == 2 && isOneStep && !isContinue) break; // для другого гравця

// обмеження першого ходу шашки в низ, якщо не дамка, то тільки в гору

if (dir[0] == -1 && dir[1] == 1 && !isOneStep) break; // перевірка ходу вгору і в право

if (IsInsideBorders(i, j)) // перевірка на можливість захоплення шашки

{

if (map[i, j] != 0 && map[i, j] != currentPlayer)

{

eatStep = true;

if (!IsInsideBorders(i + 1, j - 1))

eatStep = false;

else if (map[i + 1, j - 1] != 0) // перевірка можливості порожньої клітинки за шашкою

eatStep = false;

else return eatStep;

}

}

if (j > 0)

j--; // рух в ліво

else break;

if (isOneStep)

break;

}

j = JcurrFigure + 1;

for (int i = IcurrFigure + 1; i < 8; i++) //перевірка ходу

{

if (currentPlayer == 2 && isOneStep && !isContinue) break; //для другого гравця

// обмеження першого ходу шашки в низ, якщо не дамка, то тільки в гору

if (dir[0] == -1 && dir[1] == -1 && !isOneStep) break; // перевірка ходу в гору і в ліво

if (IsInsideBorders(i, j)) // перевірка на можливість захоплення шашки

{

if (map[i, j] != 0 && map[i, j] != currentPlayer)

{

eatStep = true;

if (!IsInsideBorders(i + 1, j + 1))

eatStep = false;

else if (map[i + 1, j + 1] != 0) // перевірка можливості порожньої клітинки за шашкою

eatStep = false;

else return eatStep;

}

}

if (j < 7)

j++; // рух в право

else break;

if (isOneStep)

break;

}

return eatStep;

}

public void CloseSteps() //закриття всіх ходів для поточної шашки

{

for (int i = 0; i < mapSize; i++)

{

for (int j = 0; j < mapSize; j++)

{

buttons[i, j].BackColor = GetPrevButtonColor(buttons[i, j]); //присвоєння попереднього кольору

}

}

}

public bool IsInsideBorders(int ti, int tj) // перевіряє чи знаходяться індекси в границях масива

{

if (ti >= mapSize || tj >= mapSize || ti < 0 || tj < 0)

{

return false;

}

return true;

}

public void ActivateAllButtons() // увімкнення всіх кніпок

{

for (int i = 0; i < mapSize; i++)

{

for (int j = 0; j < mapSize; j++)

{

buttons[i, j].Enabled = true;

}

}

}

public void DeactivateAllButtons() // вимкнення всіх кніпок

{

for (int i = 0; i < mapSize; i++)

{

for (int j = 0; j < mapSize; j++)

{

buttons[i, j].Enabled = false;

}

}

}

}

}

**Висновок.** У процесі виконання даної лабораторної роботи я отримав практичні навики розробки в середовищі Microsoft Visual Studio Community 2022 та створив гру шашки. Засвоїв репозиторій GitHab.