МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з Основ програмування

(назва дисципліни)

на тему: гра «Хрестики-нулики»

Студента 1 курсу, групи ІП-14

Прокопенко Олексій Анатолійовича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник Головченко Максим Миколайович

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ- 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-14 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Прокопенка Олексія Анатолійовича

(прізвище, ім’я, по батькові)

1. Тема роботи Гра «Хрестики-нулики»

2. Строк здачі студентом закінченої роботи 12.06.2022

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

6. Дата видачі завдання 10.02.2022

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 10.02.2022 |  |
| 2. | Підготовка ТЗ | 02.05.2022 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 03.05.2022 |  |
| 4. | Розробка сценарію роботи програми | 05.05.2022 |  |
| 6. | Узгодження сценарію роботи програми з керівником | 06.05.2022 |  |
| 5. | Розробка (вибір) алгоритму рішення задачі | 08.05.2022 |  |
| 6. | Узгодження алгоритму з керівником | 08.05.2022 |  |
| 7. | Узгодження з керівником інтерфейсу користувача | 09.05.2022 |  |
| 8. | Розробка програмного забезпечення | 10.05.2022 |  |
| 9. | Налагодження розрахункової частини програми | 15.05.2022 |  |
| 10. | Розробка та налагодження інтерфейсної частини програми | 16.05.2022 |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу | 18.05.2022 |  |
| 12. | Тестування програми | 20.05.2022 |  |
| 13. | Підготовка пояснювальної записки | 03.06.2022 |  |
| 14. | Здача курсової роботи на перевірку | 12.06.2022 |  |
| 15. | Захист курсової роботи | 18.06.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент

(підпис)

Керівник Головченко Максим Миколайович

(підпис) (прізвище, ім’я, по батькові)

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

**АНОТАЦІЯ**

Пояснювальна записка до курсової: 80 сторінок, 10 рисунків, 22 таблиці, 2 посилання.

Об’єкт дослідження: гра «Хрестики-нулики».

Мета роботи: дослідження методів прорахунку можливого ходу противника, розробка програмного забезпечення для прорахунку можливих ходів, оцінювання кожного ходу, і як наслідку наближення до побудови переможної лінії, або блокування можливого виграшу противника.

Опановано розробку програмного забезпечення з використанням ООП. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація гри «Хрестики-нулики»

ЗМІСТ

[ВСТУП 5](#_Toc105680656)

[1 ПОСТАНОВКА ЗАДАЧІ 6](#_Toc105680657)

[2 ТЕОРИТИЧНІ ВІДОМОСТІ 7](#_Toc105680658)

[3 ОПИС АЛГОРИТМІВ 9](#_Toc105680659)

[3.1 Загальний алгоритм 9](#_Toc105680660)

[3.2 Алгоритм створення ігрового поля 9](#_Toc105680661)

[3.3 Алгоритм роботи під час гри з таймером 9](#_Toc105680662)

[3.4 Алгоритм роботи програми після натискання кнопки початку гри 10](#_Toc105680663)

[3.5 Алгоритм роботи програми під час натискання на кнопку ігрового поля (ходу) 11](#_Toc105680664)

[3.6 Алгоритм роботи бота 12](#_Toc105680665)

[4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15](#_Toc105680666)

[4.1 Діаграма класів програмного забезпечення 15](#_Toc105680667)

[4.2 Опис методів частин програмного забезпечення 15](#_Toc105680668)

[4.2.1 Стандартні методи 15](#_Toc105680669)

[4.2.2 Користувацькі методи 17](#_Toc105680670)

[5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 26](#_Toc105680671)

[5.1 План тестування 26](#_Toc105680672)

[5.2 Приклади тестування 27](#_Toc105680673)

[6 ІНСТРУКЦІЯ КОРИСТУВАЧА 38](#_Toc105680674)

[6.1 Робота з програмою 38](#_Toc105680675)

[6.2 Системні вимоги 41](#_Toc105680676)

[ВИСНОВОК 42](#_Toc105680677)

[ПЕРЕЛІК ПОСИЛАНЬ 43](#_Toc105680678)

[ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ 44](#_Toc105680679)

[ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ 47](#_Toc105680680)

[MenuForm.cs 47](#_Toc105680681)

[GameForm.cs 48](#_Toc105680682)

[Presenter.cs 55](#_Toc105680683)

[Game.cs 56](#_Toc105680684)

[EnemyBot.cs 63](#_Toc105680685)

[Player.cs 75](#_Toc105680686)

[IPlayer.cs 76](#_Toc105680687)

[IController.cs 76](#_Toc105680688)

[Grid.cs 77](#_Toc105680689)

[Program.cs 78](#_Toc105680690)

# ВСТУП

Дана робота присвячена розробці гри «Хрестики-нулики» з використанням об’єктно-орієнтованого програмування. Задача програмного забезпечення полягає в графічному відображенні процесу гри, забезпеченню коректності дій гравців відповідно до правил гри, можливості вибирати в якості супротивника комп’ютер.

# 1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, для гри в хрестики-нулики на полі 20х20 в таких режимах:

1. Два гравця на одному девайсі
2. Два гравця на одному девайсі з таймером
3. Один гравець проти комп’ютера
4. Один гравець проти комп’ютера з таймером

Вхідними даними для програми є режим гри (один з вищеперерахованих).

Програмне забезпечення повинно обробляти вибір режиму гри, кліки на клітинки поля та виводити відповідну інформацію. Також повинно обробляти кліки на кнопки початку та виходу з гри.

Вихідними даними є результат кліку на клітинку поля (на ній з’являється хрестик або нулик). Також у випадку перемоги одного з них виводиться відповідне повідомлення.

# 2 ТЕОРИТИЧНІ ВІДОМОСТІ

Для реалізації можливості вибору режиму гри було створено окрему форму з двома кнопками, які відповідають за кількість гравців, та чекбоксом для вибору режиму гри з таймером. Для створення ігрового поля було використано двовимірний масив з власного класу Grid який унаслідується від класу Button і додатково містить свої координати на полі. Також було використано обробник подій який реагував на натискання кнопки ігрового поля і за допомогою класу Presenter викликав відповідний метод класу Game який обробляв клік. Для гри створюється об’єкт типу IPlayer у який присвоюється об’єкт класу Player або EnemyBot (залежно від режиму гри)

Детальніше про алгоритм для кожного режиму гри:

* Гра з іншим гравцем на одному девайсі:

Було використано булеву змінну для чередування ходу. Після кожного ходу викликався метод IsWin який перевіряв чи виграв гравець і повертав булеве значення.

* Гра з іншим гравцем на одному девайсі з таймером:

Після кожного ходу не викликається метод IsWin натомість після закінчення часу викликався метод підрахунку балів який рахує кількість повних виграшних ліній (лінія з 5 в ряд дає одне очко) і визначає переможця.

* Гра проти комп’ютера:

Створюється екземпляр класу EnemyBot в якого викликається метод Move одразу після кліку на клітинку поля. Метод Move знаходить можливі ходи (сусідні клітинки з тими на яких є хрестик або нулик), потім оцінює ці ходи за допомогою методу GetGrade. Цей метод перевіряє чи співпадає лінія клітинок з шаблоном (елементом списку \_patternsStrForX та \_patternsStrForO) і присвоює їй оцінку відповідно до збігу. І в клітинку з найбільшою оцінкою здійснюється хід.

* Гра проти комп’ютера з таймером:

Після кожного ходу не викликається метод IsWin натомість після закінчення часу викликався метод підрахунку балів який рахує кількість повних виграшних ліній (лінія з 5 в ряд дає одне очко) і визначає переможця.

# 3 ОПИС АЛГОРИТМІВ

## 3.1 Загальний алгоритм

1. ПОЧАТОК
2. Зчитати режим гри, приховати форму вибору режиму, створити ігрову форму
3. Створити ігрове поле та кнопки початку гри, та виходу з неї (3.2)
4. ЯКЩО, режим гри з таймером. ТО створити таймер а поле для виведення часу на екран (3.3)
5. Якщо натиснуто кнопку початку гри (3.4)
6. Якщо натиснуто кнопку ігрового поля (3.5)
7. КІНЕЦЬ

## 3.2 Алгоритм створення ігрового поля

Таблиця 3.2 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| \_startButton | Кнопка початку гри |
| \_exitButton | Кнопка виходу з гри |
| \_mapPanel | Панель на якій розміщується ігрове поле |
| gameMap | Ігрове поле (двовимірний масив об’єктів класу Grid) |

1. ПОЧАТОК
2. Створити кнопку початку та виходу з гри
3. Створити панель для ігрового поля
4. ЦИКЛ від 0 до 20:
   1. ЦИКЛ від 0 до 20:
      1. присвоїти кожному gameMap[i, j] новий екземпляр класу Grid та додати обробник подій OnClickGrid.
5. КІНЕЦЬ

## 3.3 Алгоритм роботи під час гри з таймером

Таблиця 3.3 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| \_countSecond | Кількість секунд відведених на гру |
| countX | Кількість переможних ліній у хрестиків |
| \_count0 | Кількість переможних ліній у нуликів |

1. ПОЧАТОК
2. Задати час для роботи таймера (\_countSecond = 30)
3. Викликати метод TimerTick:
   1. Вивести поточний час на екран
   2. ЯКЩО \_countSecond = 0, ТО викликати подію OnStopTimer:
      1. Викликати метод StopTimer класу Presenter:
         1. Викликати метод StopTimer класу GameForm:
            1. Зупинити таймер
         2. Викликати метод CountPoints класу Game:
            1. ПОКИ IsWin в який передається хрестик повертає true:

countX++

* + - * 1. ПОКИ IsWin в який передається нулик повертає true:

count0++

* + - * 1. Порівняти змінні countX та count0, залежно від результату вивести інформацію про переможця.
  1. Відняти 1 від змінної \_countSecond

1. КІНЕЦЬ

## 3.4 Алгоритм роботи програми після натискання кнопки початку гри

Таблиця 3.4 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| \_timer | Змінна для гри з таймером |
| \_timeLabel | Поле для виведення часу що залишився |

1. ПОЧАТОК
2. Зчитати натискання на кнопку
   1. При натисканні на кнопку початку гри (викликається метод OnStartButtonClick):
      1. Викликається метод EnabledButtons у який передається true:
         1. За допомогою двох вкладених ітераційних циклів змінюється властивість кнопки ігрового поля (Enabled) на true (Далі: увімкнення/вимкнення кнопок поля).
      2. ЯКЩО гра з таймером, ТО змінити властивість таймера Enabled на true.
      3. Викликати метод Reset:
         1. За допомогою двох вкладених ітераційних циклів змінюється властивість кнопки ігрового поля (Text на «», та BackColor на Color.White). Також змінній \_countSecond присвоюється значення 30;
      4. ЯКЩО гра з комп’ютером, ТО поставити хрестик в середині поля вимкнути кнопки, викликати метод EnableButtonsAfterMove і передати в нього клітинку яка знаходить в середині поля (Далі: обмежити можливі ходи):
         1. За допомогою двох вкладених ітераційних циклів здійснити обхід ігрового поля і для кожного елемента ЯКЩО цей елемент знаходить на відстані <5 від переданої кнопки, ТО змінюється властивість кнопки (Enabled) на true. Змінити переданої кнопки (Enabled) на true.
3. КІНЕЦЬ

## 3.5 Алгоритм роботи програми під час натискання на кнопку ігрового поля (ходу)

1. ПОЧАТОК
2. Зчитати натискання на кнопку ігрового поля
   1. При натисканні на кнопку ігрового поля викликається метод OnClickGrid:
      1. Обмежити ігрове поле. Викликати метод GridClick класу Game:
         1. ЯКЩО гра з іншим гравцем на одному девайсі, ТО:
            1. ЯКЩО черга ходити першого гравця (\_queue=true), ТО:

Поставити на клітинці символ відповідного гравця і змінити фон кнопки на зелений та викликати метод Move:

ЯКЩО гра не з таймером, ТО:

Викликати метод IsWin і передати в нього символ відповідного гравця:

За допомогою ітераційного циклу пройти по всіх кнопках поля, ЯКЩО знайдено 5 в ряд, ТО повернути true, ІНАКШЕ повернути false

ЯКЩО IsWin повернув true, то вивести повідомлення про перемогу.

Передати хід наступному гравцю (\_queue = false)

* + - 1. ІНАКШЕ, поставити нулик в натиснутій клітинці, перевірити чи виграв гравець (пункт 2.1.1.1.1.1.1.1)
      2. ЯКЩО IsWin повернув true, то вивести відповідне повідомлення.
      3. Викликати метод Move у класу EnemyBot (3.6)
      4. Перевірити чи виграв гравець

1. КІНЕЦЬ

## 3.6 Алгоритм роботи бота

Таблиця 3.6 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| possibleMoves | Можливі ходи |
| grades | Оцінки можливих ходів (бота) |
| \_patternsStrForX | Шаблони комбінацій для хрестиків |
| \_patternsStrForO | Шаблони комбінацій для нуликів |
| \_patternsKey | Оцінки шаблонів комбінацій |
| move | Текст для ходу («Х») |

Продовження таблиці 3.6

|  |  |
| --- | --- |
| Змінна | Призначення |
| \_gameMap | Ігрове поле |
| enemyGrades | Оцінки можливих ходів противника (гравця) |

1. ПОЧАТОК
2. Викликати метод FillPatterns:
   1. Заповнити \_patternsKey
   2. Заповнити \_patternsStrForX
   3. Для кожного елемента списку \_patternsStrForX:
      1. Замінити «Х» на «0»
   4. Для кожної кнопки ігрового поля:
      1. Викликати метод IsPossibleMove:
         1. Перевірити чи по сусідству є кнопка з текстом. ЯКЩО так повернути true, ІНАКШЕ повернути false.
      2. ЯКЩО IsPossibleMove повертає true, ТО додати його до списку можливих ходів
   5. Для кожного можливого ходу викликати метод GetGrade, передавши в нього цей хід
      1. Для кожного елемента списку \_patternsStrForX:
         1. Викликати методи перевірки збігу:
            1. ЯКЩО довжина елемента + поточна координата ходу ((move.indexX + pattern.Length – 1) більша довжини мапи, ТО повернути false
            2. Для кожного символу елемента списку \_patternsStrForX:

ЯКЩО наступна клітинка в напрямку перевірки пуста (\_gameMap[move.indexY - i, move.indexX + i].Text = ""), І символ шаблону не є пробілом (pattern[i] != ' '), АБО клітинка в напрямку перевірки не дорівнює символу шаблону (pattern[i].ToString() != \_gameMap[move.indexY - i, move.indexX + i].Text), І клітинка не пуста (\_gameMap[move.indexY - i, move.indexX + i].Text != ""), ТО повернути false

* + - * 1. Повернути true
      1. ЯКЩО один з методів перевірки збіги повертає true, ТО додати відповідну оцінку в список (grades.Add(\_patternsKey[i])), ІНАКШЕ, додати в список оцінок 0
    1. Для кожного елемента списку \_patternsStrForO повторити від пункту 2.5.1.1 до 2.5.1.1.3
    2. Знайти максимальну оцінку серед grades і enemyGrades і повернути її
  1. Знайти максимальну оцінку в списку grades і здійснити хід в цю клітинку

1. КІНЕЦЬ

# 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 4.1 Діаграма класів програмного забезпечення

Діаграма класів розробленого програмного забезпечення наведена на рисунку 4.1.

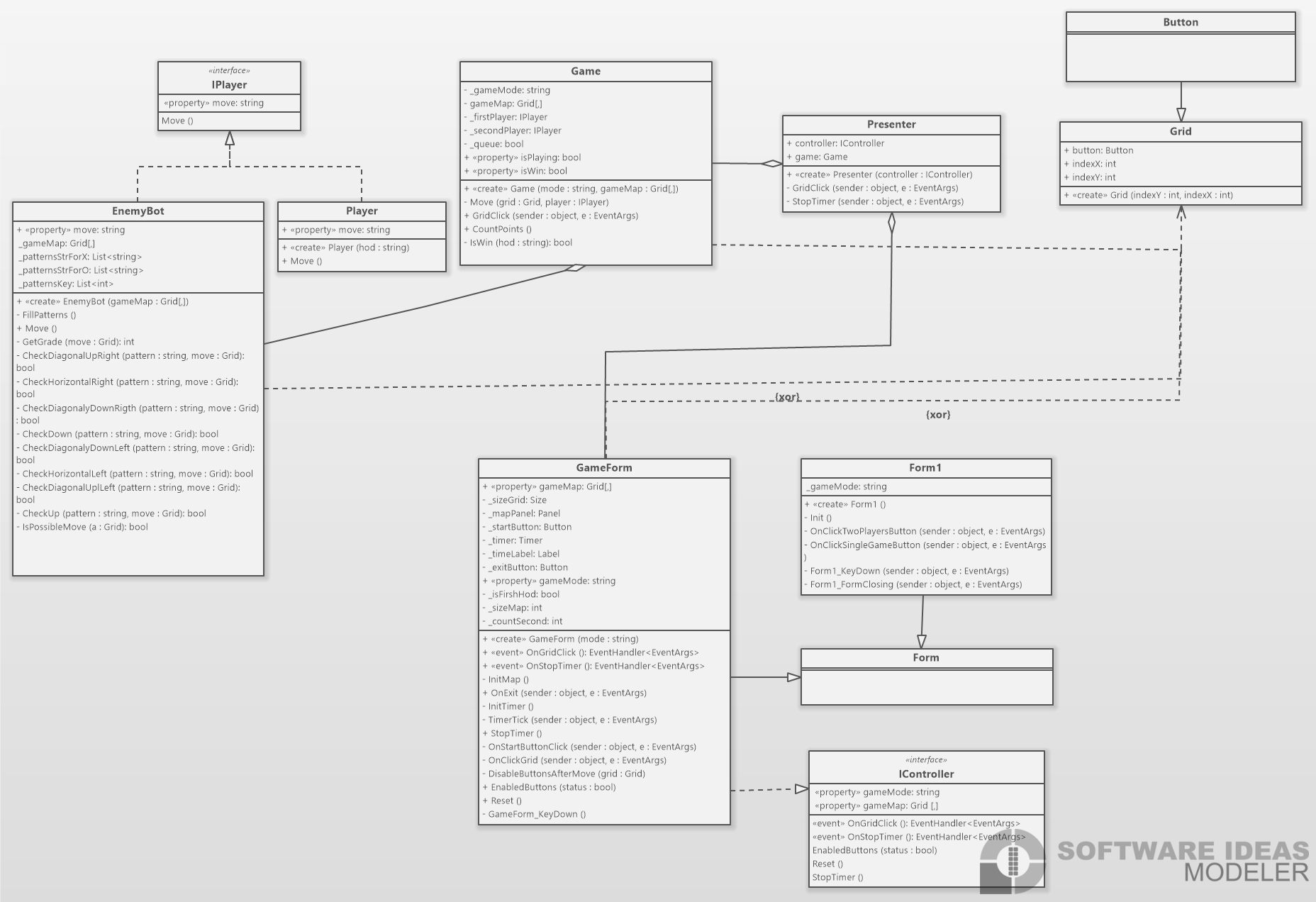
****

Рисунок 4.1 – Діаграма класів

## 4.2 Опис методів частин програмного забезпечення

### 4.2.1 Стандартні методи

У таблиці 4.1 наведено стандартні методи, використані при розробці програмного забезпечення.

Таблиця 4.1 – Стандартні методи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | List<T> | Add | Додати елемент до списку | Об’єкт (T) | - |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 2 | String | Replace | Замінити підрядок на інший | Рядок який потрібно замінити (string)  Рядок на який потрібно замінити (string) | - |
| 3 | Form | InitializeComponent | Створити форму | - | - |
| 4 | Application | Exit | Закриття програми | - | - |
| 5 | Form | Show | Вивести форму на екран | - | - |
| 6 | Form | Hide | Приховує форму | - | - |
| 7 | List<T> | ForEach | Перебір всіх елементів списку | Лямбда-вираз | - |
| 8 | List<T> | Max | Знаходить максимальний елемент | - | Максимальний елемент |

Продовження таблиці 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 9 | Array | GetLength | Повертає довжину масиву | Число яке вказує на те довжину (int) | Довжина масиву (int) |
| 10 | String | IndexOf | Знаходить позицію підрядка в рядку | Рядок (string) | Його позиція в рядку (int) |

### 4.2.2 Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, створені при розробці програмного забезпечення.

Таблиця 4.2 – користувацькі методи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | GameForm | InitMap | Створити ігрове поле і кнопки початку на виходу з гри | - | - |
| 2 | GameForm | InitTimer | Створити таймер | - | - |
| 3 | GameForm | OnExit | Вийти з гри | Кнопка (object), подія (EventArgs) | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 4 | GameForm | TimerTick | Виводить час який залишився до кінця гри | Таймер  (object), подія (EventArgs) | - |
| 5 | GameForm | StopTimer | Зупинка таймера | - | - |
| 6 | GameForm | OnStartButtonClick | Почати гру | Кнопка (object), подія (EventArgs) | - |
| 7 | GameForm | OnClickGrid | Зробити хід | Кнопка (object), подія (EventArgs) | - |
| 977777777 | GameForm | DisableButtonsAfterMove | Обмежити можливі варіанти ходу на 5 клітинок | Поточний хід (Grid) | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 10 | GameForm | EnabledButtons | Увімкнути / вимкнути кнопки ігрового поля | Увімкнути / вимкнути (bool) | - |
| 11 | GameForm | Reset | Очистити ігрове поле | - | - |
| 12 | GameForm | GameForm\_KeyDown | Кнопка зміни режиму гри | Кнопка (object), подія (KeyEventArgs) | - |
| 13 | GameForm | GameForm | Конструктор | Режим гри (string) | - |
| 14 | Presenter | Presenter | Конструктор | Інтерфейс для взаємодії з формою (IController) | - |
| 15 | Presenter | GridClick | Обробка натискання на кнопку ігрового поля | Кнопка (object), подія (EventArgs) | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 16 | Presenter | StopTimer | Зупинка таймера | Таймер  (object), подія (EventArgs) | - |
| 17 | EnemyBot | EnemyBot | Конструктор | Ігрове поле (Grid[,]) | - |
| 18 | EnemyBot | FillPatterns | Заповнити шаблони комбінацій | - | - |
| 19 | EnemyBot | Move | Зробити хід | - | - |
| 20 | EnemyBot | GetGrade | Оцінити можливий хід | Клітинка яку потрібно оцінити (Grid) | Оцінка (int) |
| 21 | EnemyBot | CheckDiagonalUpRight | Перевірити на співпадіння паттерну по діагоналі вгору вправо | Паттерн (string)  Клітинка яку потрібно оцінити (Grid) | Співпадає / не співпадає (bool) |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 22 | EnemyBot | CheckHorizontalRight | Перевірити на співпадіння паттерну по горизонталі вправо | Аналогічно для методу №21 | Аналогічно для методу №21 |
| 23 | EnemyBot | CheckDiagonalyDownRigth | Перевірити на співпадіння паттерну по діагоналі вниз вліво | Аналогічно для методу №21 | Аналогічно для методу №21 |
| 24 | EnemyBot | CheckDown | Перевірити на співпадіння паттерну по вертикалі вниз | Аналогічно для методу №21 | Аналогічно для методу №21 |
| 25 | EnemyBot | CheckDiagonalyDownLeft | Перевірити на співпадіння паттерну по діагоналі вниз вліво | Аналогічно для методу №21 | Аналогічно для методу №21 |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 26 | EnemyBot | CheckHorizontalLeft | Перевірити на співпадіння паттерну горизонтально вліво | Аналогічно для методу №21 | Аналогічно для методу №21 |
| 27 | EnemyBot | CheckDiagonalUplLeft | Перевірити на співпадіння паттерну по діагоналі вгору вліво | Аналогічно для методу №21 | Аналогічно для методу №21 |
| 28 | EnemyBot | CheckUp | Перевірити на співпадіння паттерну по вертикалі вгору | Аналогічно для методу №21 | Аналогічно для методу №21 |
| 29 | EnemyBot | IsPossibleMove | Перевірити чи є сусідня клітинка з хрестиком або нуликом | Клітинка яку потрібно оцінити (Grid) | Є по сусідству хрестик або нулик / немає (bool) |
| 30 | MenuForm | MenuForm | Конструктор | - | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 31 | MenuForm | Init | Додати обробники подій на кнопки | - | - |
| 32 | MenuForm | OnClickTwoPlayersButton | Опрацьовувати кліки на кнопку, вибирати режим гри | Кнопка (object), подія (EventArgs) | - |
| 33 | MenuForm | OnClickSingleGameButton | Опрацьовувати кліки на кнопку, вибирати режим гри | Кнопка (object), подія (EventArgs) | - |
| 34 | MenuForm | Form1\_FormClosing | Закриття гри | Кнопка (object), подія (KeyEventArgs) | - |
| 35 | Game | Game | Конструктор | Режим гри (string), ігрове поле (Grid[,]) | - |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 36 | Game | Move | Метод для ходу одного з гравців | Клітинка в яку походив гравець (Grid), гравець (IPlayer) | - |
| 37 | Game | GridClick | Метод для обробки натискання на кнопку ігрового поля | Кнопка (object), подія (EventArgs) | - |
| 38 | Game | CountPoints | Підрахунок кількості ліній з 5 хрестиків / нуликів і визначення переможця | - | - |
| 40 | Game | IsWin | Визначення чи переміг гравець після свого ходу | Хрестик / нулик (string) | Переміг / не переміг (bool) |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва методу | Призначення методу | Опис вхідних параметрів | Опис вихідних параметрів |
| 41 | Player | Player | Конструктор | Визначає чи гравець грає хрестиком / нуликом (string) | - |
| 42 | Player | Move | Потрібний для реалізації інтерфейсу | - | - |

# 5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 5.1 План тестування

Складемо план тестування програмного забезпечення, за допомогою якого протестуємо весь основний функціонал та реакцію на виключні ситуації

1. Тестування коректності вибору визначення режиму.
   1. Тестування при виборі режиму гри з таймером вдвох.
   2. Тестування при виборі режиму гри без таймеру вдвох.
2. Тестування коректності роботи кнопок.
   1. Тестування натискання на кнопку початку гри.
   2. Тестування натискання на кнопку виходу з гри.
   3. Тестування на натискання кнопки ігрового поля в режимі гри з двома гравцями.
   4. Тестування коректності обмеження можливих ходів.
3. Тестування зміни режиму гри.
   1. Тестування зміни режиму гри з двох гравців на одному девайсі на гру проти бота з таймером.
4. Тестування на можливість програми забезпечувати гру відповідно до правил у режиму гри з двома гравцями.
   1. Тестування на коректність натискання кнопок ігрового поля після запуску нової партії.
   2. Тестування на коректність визначення переможця.
   3. Тестування на коректність визначення переможця при грі з таймером
5. Тестування на відповідність дій бота правилам гри
   1. Тестування на коректність визначення можливих ходів.
   2. Тестування на коректність побудови переможної лінії.
   3. Тестування на коректність блокування можливої перемоги супротивника.
   4. Тестування на коректність визначення пріоритету ходу.

## 5.2 Приклади тестування

Проведемо тестування програмного забезпечення згідно з розробленим планом, фіксуючи мету, початковий стан програми, вхідні дані, схему проведення, очікуваний результат і стан програми після проведення випробувань кожного тесту в окрему таблицю (таблиці 5.1-5.14).

Таблиця 5.1 - Тестування при виборі режиму гри з таймером вдвох

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність вибору режму гри |
| Початковий стан програми | Відкрите вікно вибору режиму гри |
| Вхідні дані | Режим гри |
| Схема проведення тесту | Натискання на чекбокс «гра з таймером», вибір режиму гри вдвох, потім натиснути кнопку початку гри |
| Очікуваний результат | Вибиреться режим гри з таймером вдвох |
| Стан програми після проведення випробувань | Відкрита форма ігрового поля, виведений час на екран, кнопки поля активні |

Таблиця 5.2 - Тестування при виборі режиму гри без таймеру вдвох.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність вибору режиму гри |
| Початковий стан програми | Відкрите вікно вибору режиму гри |
| Вхідні дані | Режим гри |
| Схема проведення тесту | Натиснути на кнопку вибору режиму гри вдвох, потім натиснути кнопку початку гри |
| Очікуваний результат | Вибиреться режим гри без таймеру вдвох |

Продовження таблиці 5.2

|  |  |
| --- | --- |
| Стан програми після проведення випробувань | Відкрита форма ігрового поля, кнопки поля активні |

Таблиця 5.3 - Тестування натискання на кнопку початку гри.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи кнопки початку гри |
| Початковий стан програми | Відкрита форма ігрового поля, кнопки поля неактивні |
| Вхідні дані | Натискання на кнопку початку гри |
| Схема проведення тесту | Натиснути на кнопку початку гри |
| Очікуваний результат | Кнопки ігрового поля стануть активними |
| Стан програми після проведення випробувань | Відкрита форма ігрового поля, кнопки поля активні |

Таблиця 5.4 - Тестування натискання на кнопку виходу з гри.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи кнопки виходу з гри |
| Початковий стан програми | Відкрита форма ігрового поля, кнопки поля неактивні |
| Вхідні дані | Натискання на кнопку виходу з гри |
| Схема проведення тесту | Натиснути на кнопку виходу з гри |
| Очікуваний результат | Програма припинить свою діяльність, форма зникне з екрану |
| Стан програми після проведення випробувань | Програма закрита, форма зникла з екрану |

Таблиця 5.5 - Тестування на натискання кнопки ігрового поля в режимі гри з двома гравцями.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність обробки програми натискань на клітинки ігрового поля |
| Початковий стан програми | Відкрита форма ігрового поля, кнопки поля активні, вибраний режим гри з двома гравцями |
| Вхідні дані | Натискання на кнопку поля |
| Схема проведення тесту | Натиснути на будь-які кнопки ігрового поля 3 рази |
| Очікуваний результат | Після першого кліку на кнопці повинен з’явитися хрестик, а її фон повинен стати зелений, також кнопка повинна стати неактивною, після другого натискання на кнопці повинен з’явитися нулик, також кнопка повинна стати неактивною, після третього – те що після першого |
| Стан програми після проведення випробувань | На полі дві неактивні кнопки з хрестиком і зеленим фоном і одна неактивна з нуликом, також обмежилась область можливих ходів |

Таблиця 5.6 ­- Тестування коректності обмеження можливих ходів

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність обмежування можливих ігрових ходів |
| Початковий стан програми | Відкрита форма ігрового поля, кнопки поля активні, вибраний режим гри з двома гравцями |

Продовження таблиці 5.6

|  |  |
| --- | --- |
| Вхідні дані | Координати кнопки (7, 7) |
| Схема проведення тесту | Натиснути на кнопку з відповідними координатами |
| Очікуваний результат | Клітинки які знаходяться на відстані більше 5 від заданої точки стануть неактивними |
| Стан програми після проведення випробувань | Клітинка на яку натиснули стала неактивною, там з’явився хрестик, її фон став зелений. Неактивними стали всі кнопки крім кнопок з координатами від (2, 2) до (12,12) наглядно не видно на рисунку 5.6.1 |

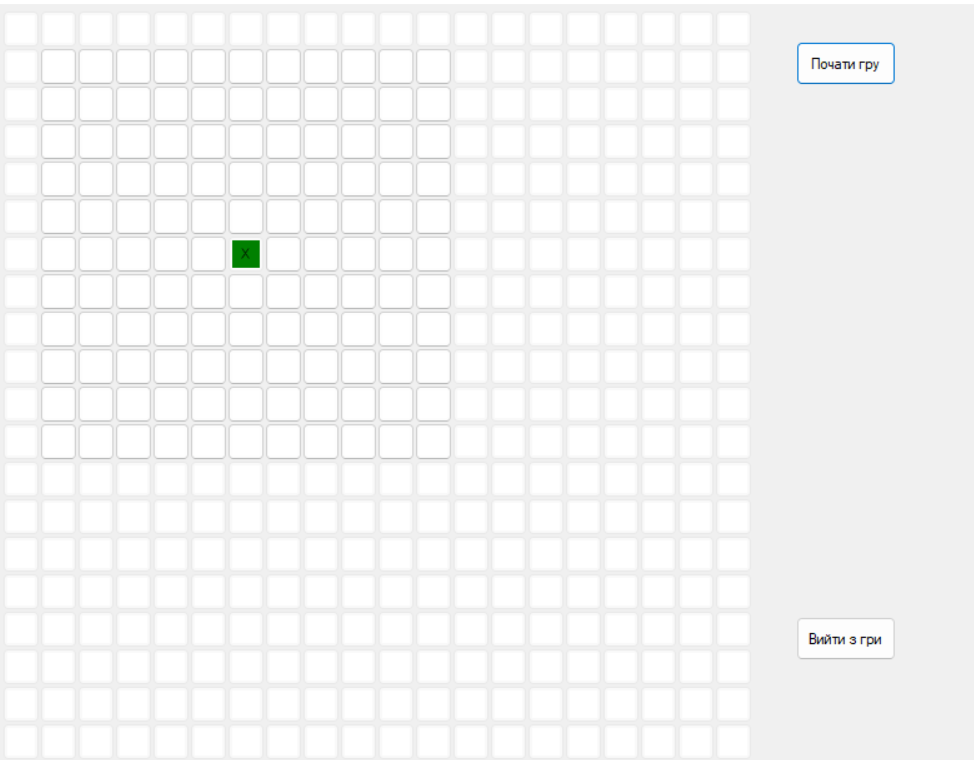


Рисунок 5.6.1

Таблиця 5.7 - Тестування зміни режиму гри з двох гравців на одному девайсі на гру проти бота з таймером

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність зміни режимів гри без перезапуску програми |
| Початковий стан програми | Відкрите вікно з ігровим полем, кнопки неактивні |
| Вхідні дані | Режим гри |
| Схема проведення тесту | Натиснути кнопку escape, у формі яка відкрилась натиснути чекбокс «гра з таймером», та вибрати режим гри проти комп’ютера |
| Очікуваний результат | Зміниться режим гри |
| Стан програми після проведення випробувань | Відкрита форма з ігровим полем в режимі гри проти комп’ютера, виведений час на екран, в центрі поля стоїть хрестик |

Таблиця 5.8 - Тестування на коректність натискання кнопок ігрового поля після запуску нової партії.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність натискання кнопок. Перевірити правильність черги після початку нової гри |
| Початковий стан програми | Відкрите вікно з ігровим полем в режимі гри вдвох, кнопки неактивні |
| Вхідні дані | - |
| Схема проведення тесту | Натиснути кілька будь-яких кнопок ігрового поля, після цього натиснути кнопку початку гри, і натиснути на будь-яку кнопку ігрового поля |

Продовження таблиці 5.8

|  |  |
| --- | --- |
| Очікуваний результат | Текст на кнопках буде змінюватися по черзі, після натискання кнопки початку гри поле очиститься і при натисканні кнопки ігрового поля та ній з’явиться хрестик |
| Стан програми після проведення випробувань | Відкрита форма з ігровим полем, на якому стоїть хрестик, також кнопки в радіусі більше ніж 5 неактивні |

Таблиця 5.9 - Тестування на коректність визначення переможця.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність визначення переможця |
| Початковий стан програми | Відкрита форма з ігровим полем в режимі два гравця без таймеру, побудована лінія з 4 хрестиків в ряд |
| Вхідні дані | - |
| Схема проведення тесту | Натиснути кнопку яка побудує 5 хрестиків в ряд |
| Очікуваний результат | Виведеться повідомлення про перемогу, клітинки в яких була переможна лінія очистяться від хрестиків. |
| Стан програми після проведення випробувань | Виведене повідомлення про перемогу хрестиків, при його закритті всі кнопки стали неактивними, на місті хрестиків які утворили переможну лінію пусті клітинки з зеленим фоном. |

Таблиця 5.10 - Тестування на коректність визначення переможця при грі з таймером

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність визначення переможця у грі з таймером |
| Початковий стан програми | Відкрита форма ігрового поля в режиму два гравці з таймером |
| Вхідні дані | Час – 30 секунд |
| Схема проведення тесту | Побудувати виграшну лінію хрестиків за відведений час |
| Очікуваний результат | Програма тільки після закінчення часу виведе повідомлення про перемогу хрестиків |
| Стан програми після проведення випробувань | Виведене повідомлення про перемогу хрестиків, при його закритті всі кнопки стали неактивними, на місті хрестиків які утворили переможну лінію пусті клітинки з зеленим фоном, час на таймері: 00:00 |

Таблиця 5.11 - Тестування на коректність визначення можливих ходів

|  |  |
| --- | --- |
| Мета тесту | Визначити чи коректно бот визначає можливі ходи |
| Початковий стан програми | Відкрита форма ігрового поля в режиму гравець проти комп’ютера. В центрі поля стоїть хрестик |
| Вхідні дані | Хід противника |
| Схема проведення тесту | Додати зафарбовування можливих ходів в червоний колір |

Продовження таблиці 5.11

|  |  |
| --- | --- |
| Очікуваний результат | Клітинки по сусідству з клітинками з хрестиком або нуликом стають червоними, в одній з них з’явиться хрестик |
| Стан програми після проведення випробувань | Стан програми можна побачити на рисунку 5.11.1 |

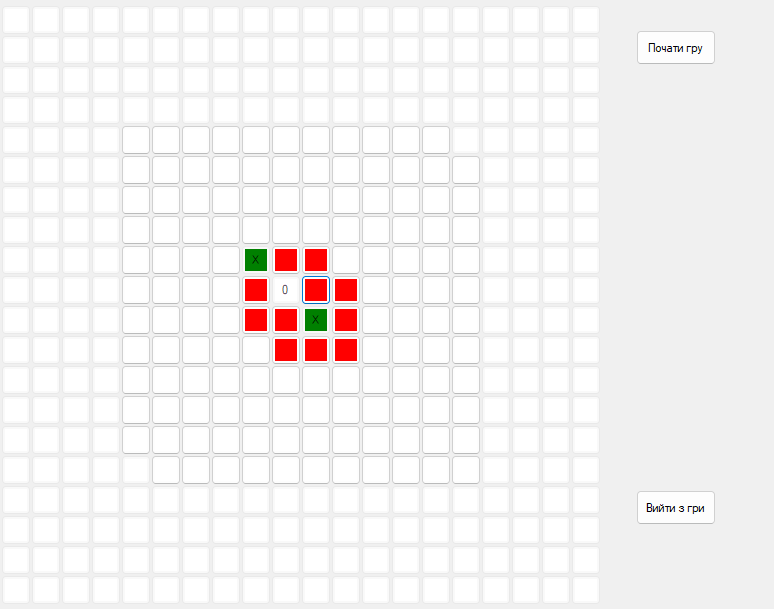


Рисунок 5.11.1

Таблиця 5.12 - Тестування на коректність побудови переможної лінії

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність побудови переможної лінії |
| Початковий стан програми | Відкрита форма з ігровим полем, в режимі гри з ботом |
| Вхідні дані | - |

Продовження таблиці 5.12

|  |  |
| --- | --- |
| Схема проведення тесту | Не перешкоджати боту будувати переможну лінію і не намагатися перемогти його |
| Очікуваний результат | Бот побудує переможну лінію за 5 ходів і переможе |
| Стан програми після проведення випробувань | Стан програми можна побачити на рисунку 5.12.1 |

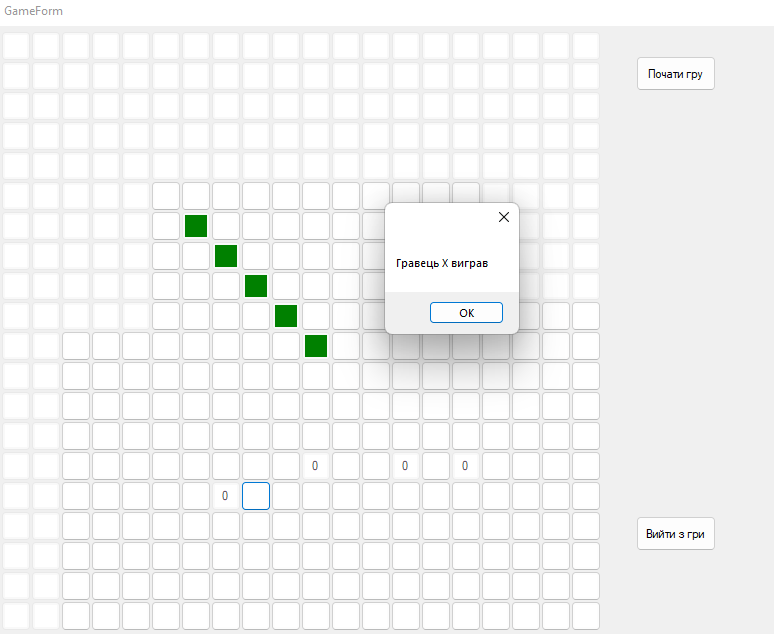


Рисунок 5.12.1

Таблиця 5.13 - Тестування на коректність блокування можливої перемоги супротивника.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність блокування можливої перемоги супротивника |

Продовження таблиці 5.13

|  |  |
| --- | --- |
| Початковий стан програми | Відкрита форма з ігровим полем, в режимі гри з ботом |
| Вхідні дані | - |
| Схема проведення тесту | Побудувати лінію з 4 нуликів в ряд і перевірити чи бот заблокує можливість перемогти |
| Очікуваний результат | Бот поставить на місце можливого п’ятого переможного нулика хрестик |
| Стан програми після проведення випробувань | Стан програми можна побачити на рисунку 5.13.1 |

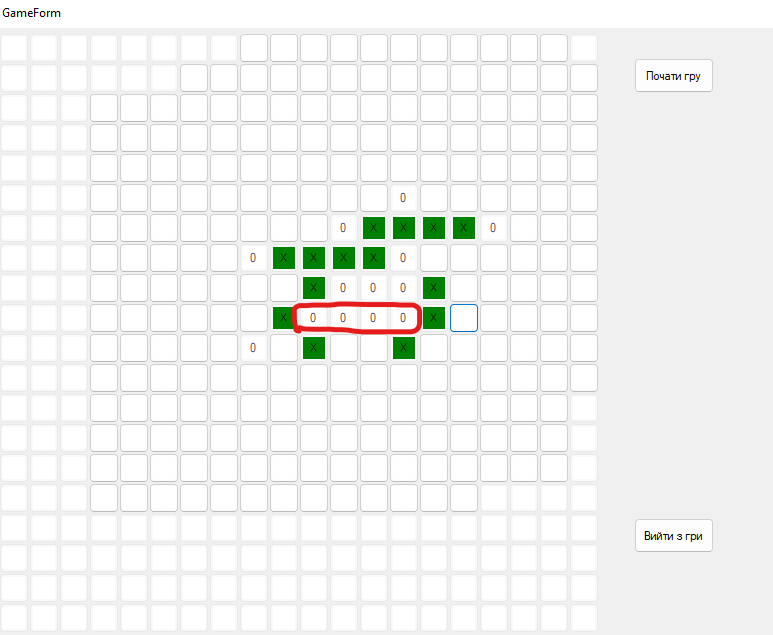


Рисунок 5.13.1

Таблиця 5.14 - Тестування на коректність визначення пріоритету ходу.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність визначення пріоритету ходу |
| Початковий стан програми | Відкрита форма з ігровим полем, в режимі гри з ботом |
| Вхідні дані | - |
| Схема проведення тесту | Почати будувати переможну лінію не заважаючи ботові |
| Очікуваний результат | Так як бот ходить перший він першим і побудує переможну лінію, і не буде блокувати нашу |
| Стан програми після проведення випробувань | Відкрите вікно з повідомленням що хрестики перемогли, на формі лінія з 5 в ряд пустих клітинок з зеленим фоном, і лінія з 4 нуликів |

# 6 ІНСТРУКЦІЯ КОРИСТУВАЧА

## 6.1 Робота з програмою

Після запуску файлу з розширенням \*.exe відкривається вікно з вибором режиму гри (рисунок 6.1).

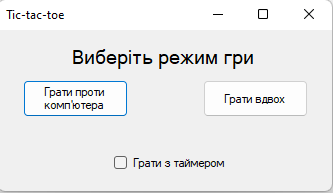


Рисунок 6.1 – Привітальне вікно

Для вибору режиму гри потрібно натиснути на одну з цих кнопок, для гри з таймером потрібно вибрати пункт «Грати з таймером». Після вибору режиму гри у вас відкривається ігрове поле (рисунок 6.2)

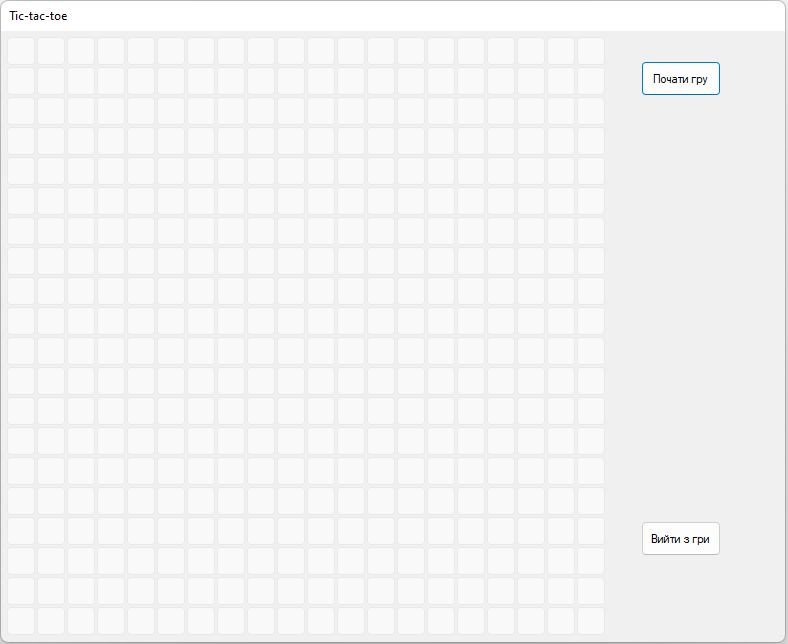


Рисунок 6.2 – Ігрове поле

Для початку гри натисніть кнопку «Почати гру» або Enter на клавіатурі, після цього ви можете натискати на кнопки ігрового поля. (рисунок 6.3) Якщо ви вибрали режим гри з таймером, то у вас відкриється поле як на рисунку 6.4, а якщо з комп’ютером, то як на рисунку 6.5

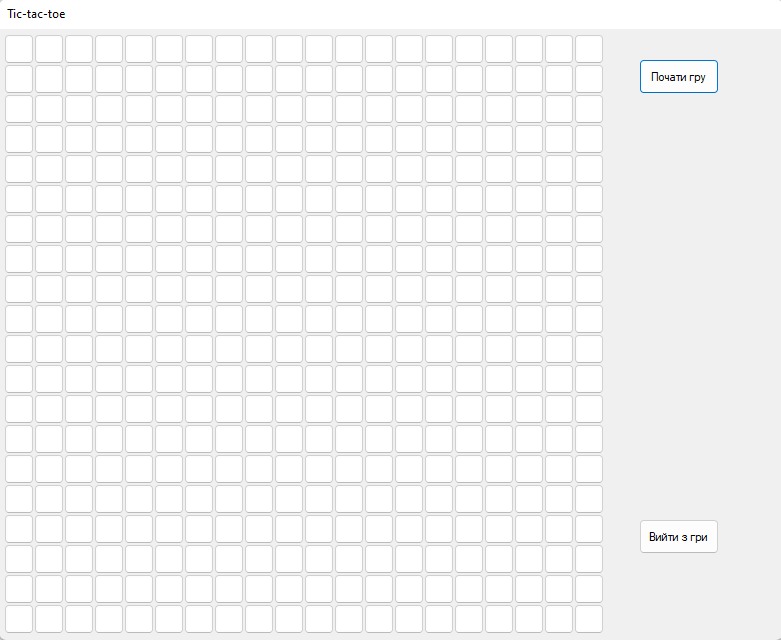


Рисунок 6.3 – Ігрове поле з початою грою

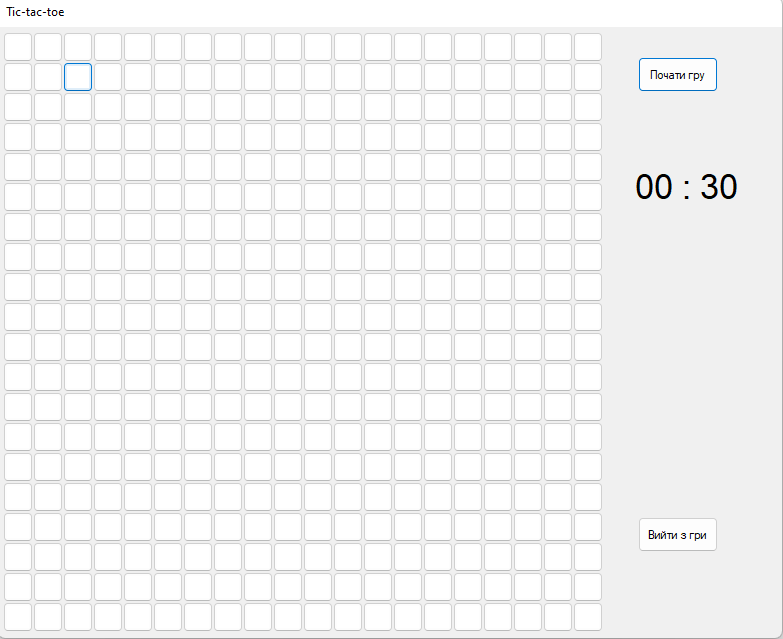


Рисунок 6.4 – Ігрове поле в режимі гри з таймером

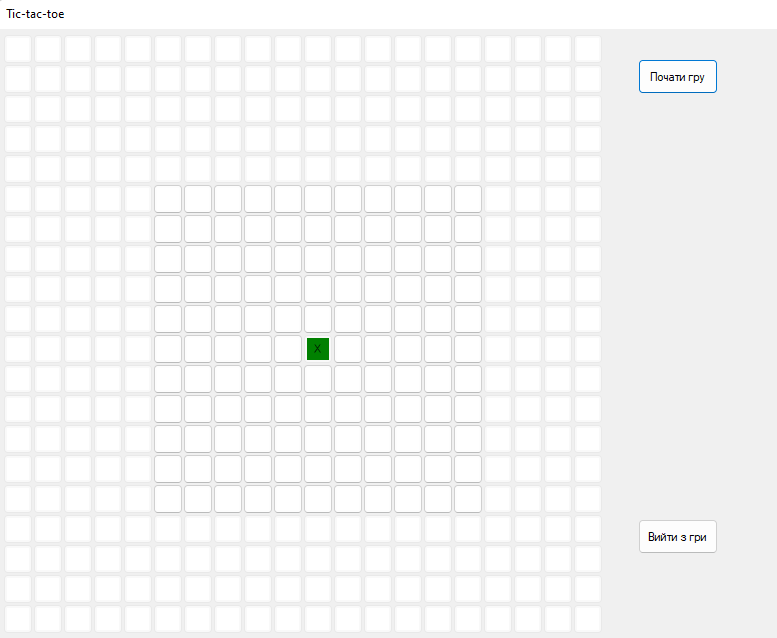


Рисунок 6.5 – Ігрове поле в режимі гри проти комп’ютера

Після побудови 5-ти ліній в ряд виведеться повідомлення про перемогу одного з гравців (рисунок 6.6) Після виведення повідомлення ви можете закрити його і почати гру заново натиснувши кнопку «Почати гру» або Enter на клавіатурі

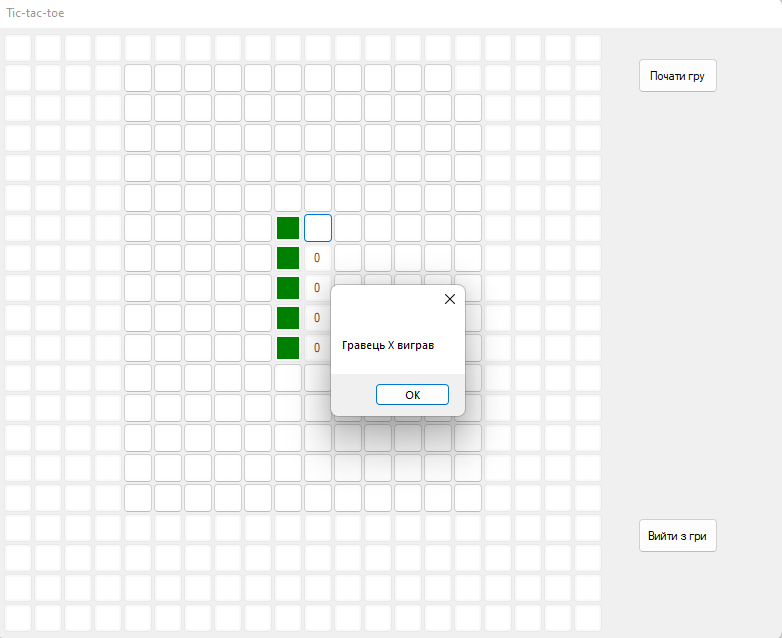


Рисунок 6.6 – Повідомлення про перемогу одного з гравців

Для зміни режиму гри натисніть кнопку Escape після цього у вас відкриється вікно вибору режиму гри (рисунок 6.1). Для виходу з гри натисніть кнопку «Вийти з гри»

## 6.2 Системні вимоги

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows 10/Windows 11 (з останніми обновленнями) | Windows 11  (з останніми обновленнями) |
| Процесор | Intel® Pentium® ІІІ  1.0 GHz або  AMD Athlon™ 1.0 GHz | Intel® Pentium® D або AMD Athlon™ 64 X2 |
| Оперативна пам'ять | 2 GB RAM | 8 GB RAM |
| Відеоадаптер | Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Дисплей | 800х600 | 1024х768 або краще |
|  | Мінімальні | Рекомендовані |
| Прилади введення | Клавіатура, комп'ютерна миша | |
| Додаткове програмне забезпечення | Microsoft .NET Framework 4.8 | |

# ВИСНОВОК

Отже, ми дослідили методи та створили алгоритм для гри в хрестики-нулики проти реального гравця, створили програмне забезпечення на його основі з використанням об’єктно-орієнтованого програмування та протестували його.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Хрестики-нулики URL:

<https://uk.wikipedia.org/wiki/%D0%A5%D1%80%D0%B5%D1%81%D1%82%D0%B8%D0%BA%D0%B8-%D0%BD%D1%83%D0%BB%D0%B8%D0%BA%D0%B8>

2. Гомоку URL:

<https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BC%D0%BE%D0%BA%D1%83>

# ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

Затвердив

Керівник Головченко М. М.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

Виконавець:

Студент Прокопенко О.А.\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: *Гра «Хрестики-Нулики»*

з дисципліни:

«Основи програмування»

Київ 2022

* 1. *Мета*: Метою курсової роботи є розробка

програмного забезпечення на мові C# з використанням

WinForms для гри в хрестики-нулики.

* 1. *Дата початку роботи*: «20» березня 2022 р.
  2. *Дата закінчення роботи*: «9» червня 2022 р.
  3. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:

* Можливість грати в хрестики-нулики на полі 20х20 клітинок
* Можливість робити хід
* Грати вдвох на одному комп’ютері
* Грати проти комп’ютера
* Можливість грати з таймером
* Можливість починати гру заново на чистому полі
* Можливість обмежувати варіанти для можливого ходу
* Можливість обирати режим гри
* Можливість перевірки відповідності дій гравця правилам гри
* Можливість визначення переможця, та виведення відповідного інформаційного повідомлення

1. Нефункціональні вимоги:

* Можливість запускати програму на комп’ютері з Windows 10 і вище.
* Запуск програми з .NET Framework версії 4.8.
* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до\_\_.\_\_.202\_ р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до \_\_.\_\_.202\_р.)
3. Розробка програмного забезпечення (до \_\_.\_\_.202\_р.)
4. Тестування розробленої програми (до \_\_.\_\_.202\_р.)
5. Розробка пояснювальної записки (до \_\_.\_\_.202\_ р.).
6. Захист курсової роботи (до \_\_.\_\_.202\_ р.).
   1. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

# ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

## MenuForm.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 cursachMVC

{

public partial class MenuForm : Form

{

string \_gameMode;

private GameForm \_gameForm;

public MenuForm()

{

InitializeComponent();

Init();

}

private void Init ()

{

singleGameButton.Click += new EventHandler(OnClickSingleGameButton);//подія при натисканні на днопку "грати проти комп'ютера"

twoPlayersButton.Click += new EventHandler(OnClickTwoPlayersButton);//подія при натисканні на днопку "грати вдвох"

}

private void OnClickTwoPlayersButton(object sender, EventArgs e) //метод при натисканні на днопку "грати вдвох"

{

\_gameMode = "two ";

if (timerCheckBox.Checked)

\_gameMode += "timer";

this.Hide();

\_gameForm = new GameForm(\_gameMode);

Presenter presenter = new Presenter(\_gameForm);

\_gameForm.Show();

}

private void OnClickSingleGameButton(object sender, EventArgs e)//метод при натисканні на днопку "грати проти комп'ютера"

{

\_gameMode = "single ";

if (timerCheckBox.Checked)

\_gameMode += "timer";

this.Hide();

\_gameForm = new GameForm(\_gameMode);

Presenter presenter = new Presenter(\_gameForm);

\_gameForm.Show();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)//подія при закриванні форми

{

Application.Exit();

}

}

}

## GameForm.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 cursachMVC

{

public partial class GameForm : Form, IController //форма де відбувається гра

{

public GameForm(string mode)

{

InitializeComponent();

this.ControlBox = false;

this.Text = "Tic-tac-toe";

StartPosition = FormStartPosition.CenterScreen;

FormBorderStyle = FormBorderStyle.FixedDialog;

gameMode = mode;

InitMap();

if (gameMode == "two timer" || gameMode == "single timer") //гра вдвох на одному компі з таймером

{

InitTimer();

}

}

public Grid[,] gameMap { get; private set; } //ігрове поле

private Size \_sizeGrid; //розмір клітинки поля

private Panel \_mapPanel; //панель на якій розміщується поле

private Button \_startButton; //кнопка для запуску гри

private Timer \_timer; //таймер

private Label \_timeLabel; //поле для виведення часу

private Button \_exitButton; //кнопка виходу з програми

//ігровий режим (людина проти комп'ютера, людина проти комп'ютера з таймером,

//людина проти людиин, людина проти людиин з таймером)

public string gameMode { get; set; }

public bool isFirstMove { get; set; } //змінна для визначення чи був здійснений перший хід

private int \_sizeMap; //розмір ігрового поля

private int \_countSecond; //кількість секунд таймера

public event EventHandler<EventArgs> OnGridClick; //подія на натискання на игрове поле

public event EventHandler<EventArgs> OnStopTimer; //подія зупинки таймера

private void InitMap () //створення ігрового поля і інтерфейсу

{

\_sizeGrid = new Size(30, 30);

\_sizeMap = 20;

\_mapPanel = new Panel ();

gameMap = new Grid[\_sizeMap, \_sizeMap];

\_startButton = new Button();

this.Width = (\_sizeGrid.Width + 10) \* \_sizeMap;

this.Height = (\_sizeGrid.Height) \* \_sizeMap+50;

\_mapPanel.Location = new Point(0, 0);

\_mapPanel.Size = new Size(\_sizeGrid.Width\*\_sizeMap+10, this.Height);

\_mapPanel.Parent = this;

\_startButton.Text = "Почати гру";

\_startButton.Size = new Size(80, 35);

\_startButton.Location = new Point(\_mapPanel.Width+30, 30);

\_startButton.Parent = this;

\_startButton.Click += new EventHandler(OnStartButtonClick);

\_exitButton = new Button();

\_exitButton.Text = "Вийти з гри";

\_exitButton.Size = new Size(80, 35);

\_exitButton.Location = new Point(\_mapPanel.Width + 30, this.Height-160);

\_exitButton.Parent = this;

\_exitButton.Click += new EventHandler(OnExit);

for (int i = 0; i < \_sizeMap; i++)

{

for (int j = 0; j < \_sizeMap; j++)

{

gameMap[i, j] = new Grid(i,j);

gameMap[i, j].Size = \_sizeGrid;

gameMap[i, j].Location = new Point(\_sizeGrid.Width \* j + 5, \_sizeGrid.Width \* i + 5);

gameMap[i, j].Parent = \_mapPanel;

gameMap[i, j].Click += new EventHandler(OnClickGrid);

gameMap[i, j].Enabled = false;

}

}

}

public void OnExit(object sender, EventArgs e) //закриття програми

{

Application.Exit();

}

private void InitTimer () //створення таймера

{

\_timer = new Timer();

\_timeLabel = new Label();

\_timeLabel.Size = new Size(this.Width - \_mapPanel.Width - 30, 60);

\_timeLabel.Location = new Point (\_mapPanel.Width + 20, 140);

\_timeLabel.Parent = this;

\_timeLabel.Font = new Font(FontFamily.GenericSansSerif, 25, FontStyle.Regular);

\_timer.Tick += new EventHandler(TimerTick);

\_timer.Interval = 1000;

\_countSecond = 30;

}

private void TimerTick(object sender, EventArgs e) //виведення часу таймера

{

\_timeLabel.Text = $"00 : {\_countSecond}";

if (\_countSecond == 0)

{

OnStopTimer(sender, e);

EnabledButtons(false);

}

\_countSecond--;

}

public void StopTimer () //зупинка таймера

{

\_timer.Enabled = false;

}

private void OnStartButtonClick(object sender, EventArgs e) //початок гри

{

isFirstMove = true;

Button grid = (Button)sender;

EnabledButtons(true);

if (gameMode.IndexOf("timer") !=-1)

\_timer.Enabled = true;

Reset();

if (gameMode.IndexOf("single") != -1)

{

gameMap[\_sizeMap / 2, \_sizeMap / 2].Text = "X";

gameMap[\_sizeMap / 2, \_sizeMap / 2].BackColor = Color.Green;

EnabledButtons(false);

DisableButtonsAfterMove(new Grid(\_sizeMap/2, \_sizeMap / 2));

isFirstMove = false;

}

}

private void OnClickGrid(object sender, EventArgs e) //дія при натисканні на клітинку ігрового поля

{

Grid grid = (Grid)sender;

if (isFirstMove)

{

EnabledButtons(false);

OnGridClick(sender, e);

isFirstMove = false;

DisableButtonsAfterMove(grid);

}

else

{

DisableButtonsAfterMove(grid);

OnGridClick(sender, e);

}

}

private void DisableButtonsAfterMove(Grid grid) //обмеження доступних клітинок для ходу

{

for (int i = 0; i < \_sizeMap; i++)

{

for (int j = 0; j < \_sizeMap; j++)

{

if (i >= grid.indexY - 5 && i <= grid.indexY + 5 && j >= grid.indexX - 5 && j <= grid.indexX + 5 && gameMap[i, j].Text == "")

gameMap[i, j].Enabled = true;

}

}

grid.Enabled = false;

}

public void EnabledButtons(bool status) //вимкнення або увімкнення всіх кнопок

{

for (int i = 0; i < \_sizeMap; i++)

{

for (int j = 0; j < \_sizeMap; j++)

{

gameMap[i, j].Enabled = status;

}

}

}

public void Reset () //очищення ігрового поля для початку гри заново

{

for (int i = 0; i < \_sizeMap; i++)

{

for (int j = 0; j < \_sizeMap; j++)

{

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

gameMap[i, j].BackColor = Color.White;

}

}

\_countSecond = 30;

}

//подія при натисканні кнопки Escape для вибору режиму гри

//приховує цю форму і показує MenuForm

private void GameForm\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyValue == (char)Keys.Escape)

{

MenuForm form = new MenuForm();

form.Show();

this.Hide();

}

if (e.KeyValue == (char)Keys.Enter)

{

OnStartButtonClick (\_startButton, new EventArgs());

}

}

}

}

## Presenter.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace cursachMVC

{

internal class Presenter //клас для передавання подій з форми в гру, та передавання даних для виведення на форму з гри

{

public IController controller; //змінна яка забезпечує доступ до подій і змінних форми

public Game game;

public Presenter(IController controller)

{

this.controller = controller;

game = new Game(controller.gameMode, controller.gameMap);

controller.OnGridClick += new EventHandler<EventArgs>(GridClick);

controller.OnStopTimer += new EventHandler<EventArgs>(StopTimer);

}

//в цьому методі відбувається обробка результату натискання на кнопку ігрового поля та передачі в форму

//для виведення на екран

private void GridClick(object sender, EventArgs e)

{

Grid grid = (Grid)sender;

if (controller.isFirstMove == true)

game.queue = true;

game.GridClick(sender, e);

if (game.isWin)

{

controller.EnabledButtons(false);

game.isWin = false;

}

}

//у цьому методі обробляється зупинка таймера

private void StopTimer (object sender, EventArgs e)

{

controller.StopTimer();

game.CountPoints(); //метод підрахунку кількості балів

}

}

}

## Game.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace cursachMVC

{

internal class Game

{

private string \_gameMode;

private Grid[,] gameMap;

private IPlayer \_firstPlayer;

private IPlayer \_secondPlayer;

public bool queue { get; set; } //змінна яка служить для чередування ходів

//public bool isPlaying { get; set; }

public bool isWin { get; set; }

public Game(string mode, Grid[,] gameMap)

{

\_gameMode = mode;

this.gameMap = gameMap;

queue = true;

\_secondPlayer = new Player("X");

//створення гравців залежно від режиму гри

if (mode == "two ")

{

\_firstPlayer = new Player("X");

\_secondPlayer.move = "0";

}

else if (mode == "single ")

{

\_firstPlayer = new EnemyBot(gameMap);

\_secondPlayer.move = "0";

}

else if (mode == "two timer")

{

\_firstPlayer = new Player("X");

\_secondPlayer.move = "0";

}

else if (mode == "single timer")

{

\_firstPlayer = new EnemyBot(gameMap);

\_secondPlayer.move = "0";

}

}

//метод в якому визначається черга ходу, переможець

private void Move(Grid grid, IPlayer player)

{

grid.Text = player.move;

if (\_gameMode.IndexOf('r') == -1) //якщо гра не на таймер після кожного ходу перевіряється чи виграв гравець який походив

{

if (IsWin(player.move))

{

isWin = true;

queue = true;

MessageBox.Show($"Гравець {player.move} виграв");

return;

}

}

}

public void GridClick(object sender, EventArgs e)

{

Grid grid = (Grid)sender;

if (\_gameMode.IndexOf("two") != -1) //гра з іншим гравцем

{

if (queue) //хід першого або другого гравця

{

grid.BackColor = Color.Green;

queue = false;

Move(grid, \_firstPlayer);

}

else

{

queue = true;

Move(grid, \_secondPlayer);

}

}

else //гра з ботом

{

grid.Text = \_secondPlayer.move;

if (\_gameMode.IndexOf('r') == -1)

{

if (IsWin(\_secondPlayer.move))

{

isWin = true;

MessageBox.Show($"Гравець {\_secondPlayer.move} виграв");

}

}

if (!isWin)

\_firstPlayer.Move(); //ход бота

if (\_gameMode.IndexOf('r') == -1)

{

if (IsWin(\_firstPlayer.move))

{

isWin = true;

MessageBox.Show($"Гравець {\_firstPlayer.move} виграв");

}

}

}

}

public void CountPoints () //метод який викликається при грі з таймером для підрахунку кількості ліній 5 в ряд

{

int countX=0, count0 = 0;

while (IsWin("X") == true)

{

countX++;

}

while (IsWin("0") == true)

{

count0++;

}

if (countX > count0)

{

MessageBox.Show("Гравець X виграв");

}

else if (countX == count0)

{

MessageBox.Show("Нічия");

}

else

{

MessageBox.Show("Гравець 0 виграв");

}

}

private bool IsWin(string hod) //перевірити яи коректно визначає переможця

{

for (int i = 0; i < 20; i++) //перевірка кожної клітинки на те чи веред від неї ще 4 клітинки з одним символом в ряд

{

for (int j = 0; j < 20; j++)

{

if (i + 4 < 20)

{

if (gameMap[i, j].Text == hod && gameMap[i + 1, j].Text == hod && gameMap[i + 2, j].Text == hod && gameMap[i + 3, j].Text == hod && gameMap[i + 4, j].Text == hod)

{

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

gameMap[i + 1, j].Text = "";

gameMap[i + 2, j].Text = "";

gameMap[i + 3, j].Text = "";

gameMap[i + 4, j].Text = "";

return true;

}

}

if (j + 4 < 20)

{

if (gameMap[i, j].Text == hod && gameMap[i, j + 1].Text == hod && gameMap[i, j + 2].Text == hod && gameMap[i, j + 3].Text == hod && gameMap[i, j + 4].Text == hod)

{

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

gameMap[i, j + 1].Text = "";

gameMap[i, j + 2].Text = "";

gameMap[i, j + 3].Text = "";

gameMap[i, j + 4].Text = "";

return true;

}

}

if (j + 4 < 20 && i + 4 < 20)

{

if (gameMap[i, j].Text == hod && gameMap[i + 1, j + 1].Text == hod && gameMap[i + 2, j + 2].Text == hod && gameMap[i + 3, j + 3].Text == hod && gameMap[i + 4, j + 4].Text == hod)

{

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

gameMap[i + 1, j + 1].Text = "";

gameMap[i + 2, j + 2].Text = "";

gameMap[i + 3, j + 3].Text = "";

gameMap[i + 4, j + 4].Text = "";

return true;

}

}

if (i + 4 < 20 && j - 4 > 0)

{

if (gameMap[i, j].Text == hod && gameMap[i + 1, j - 1].Text == hod && gameMap[i + 2, j - 2].Text == hod && gameMap[i + 3, j - 3].Text == hod && gameMap[i + 4, j - 4].Text == hod)

{

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

gameMap[i + 1, j - 1].Text = "";

gameMap[i + 2, j - 2].Text = "";

gameMap[i + 3, j - 3].Text = "";

gameMap[i + 4, j - 4].Text = "";

return true;

}

}

}

}

return false;

}

}

}

## EnemyBot.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace cursachMVC

{

internal class EnemyBot : IPlayer

{

public string move { get; set; }

Grid[,] \_gameMap;

List<string> \_patternsStrForX;

List<string> \_patternsStrForO;

List<int> \_patternsKey;

public EnemyBot(Grid[,] gameMap)

{

move = "X";

this.\_gameMap = gameMap;

\_patternsStrForX = new List<string>();

\_patternsStrForO = new List<string>();

\_patternsKey = new List<int>();

FillPatterns();

}

private void FillPatterns() //заповнення шаблонів

{

\_patternsKey.Add(100000);

\_patternsKey.Add(7000);

\_patternsKey.Add(4000);

\_patternsKey.Add(4000);

\_patternsKey.Add(2000);

\_patternsKey.Add(2000);

\_patternsKey.Add(2000);

\_patternsKey.Add(2000);

\_patternsKey.Add(2000);

\_patternsKey.Add(2000);

\_patternsKey.Add(3000);

\_patternsKey.Add(1500);

\_patternsKey.Add(1500);

\_patternsKey.Add(800);

\_patternsKey.Add(800);

\_patternsKey.Add(800);

\_patternsKey.Add(800);

\_patternsKey.Add(200);

////////////////////////////////

\_patternsStrForX.Add("XXXXX");

\_patternsStrForX.Add(" XXXX ");

\_patternsStrForX.Add("XXXX ");

\_patternsStrForX.Add(" XXXX");

\_patternsStrForX.Add(" X XXX");

\_patternsStrForX.Add(" XX XX");

\_patternsStrForX.Add(" XXX X");

\_patternsStrForX.Add("XXX X ");

\_patternsStrForX.Add("XX XX ");

\_patternsStrForX.Add("X XXX ");

\_patternsStrForX.Add(" XXX ");

\_patternsStrForX.Add(" XXX");

\_patternsStrForX.Add("XXX ");

\_patternsStrForX.Add(" XX X");

\_patternsStrForX.Add("XX X ");

\_patternsStrForX.Add(" X XX");

\_patternsStrForX.Add("X XX ");

\_patternsStrForX.Add(" XX ");

////////////////////////////

foreach (var a in \_patternsStrForX)

{

\_patternsStrForO.Add(a.Replace('X', '0'));

}

}

//(якщо вага ходу противника(людини) бульша за вагу ходу бота,

//то бот робить хід для блокування можливої виграшної комбінації,

//якщо навпаки то бот робить хід для своєї перемоги)

public void Move() //метод для здійснення ходу

{

List<Grid> possibleMoves = new List<Grid>();

List<int> grades = new List<int>();

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

{

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

{

if (IsPossibleMove(\_gameMap[i, j]))

{

possibleMoves.Add(\_gameMap[i, j]);

//gameMap[i, j].BackColor = Color.Red;

}

}

}

possibleMoves.ForEach(move => //оцінювання кожного можливого ходу

{

grades.Add(GetGrade(move));

});

int maxGrade = 0;

int index = 0;

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

{

if (maxGrade<grades[i])

{

maxGrade = grades[i];

index = i;

}

}

possibleMoves[index].Text = "X";

possibleMoves[index].Enabled = false;

possibleMoves[index].BackColor = Color.Green;

}

private int GetGrade(Grid move) //метод для оцінювання кожного можливого ходу

{

List<int> grades = new List<int>(); //оцінки по одному паретну по всіх напрямках

List<int> enemyGrades = new List<int>(); //оцінки моїх ходів по одному паттерну та всіх напрямках

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

{

if (CheckDiagonalUpRight(\_patternsStrForX[i], move) || CheckHorizontalRight(\_patternsStrForX[i], move) || CheckDiagonalyDownRigth(\_patternsStrForX[i], move) ||

CheckDown(\_patternsStrForX[i], move) || CheckDiagonalyDownLeft(\_patternsStrForX[i], move) || CheckHorizontalLeft(\_patternsStrForX[i], move) ||

CheckDiagonalUplLeft(\_patternsStrForX[i], move) || CheckUp(\_patternsStrForX[i], move))

{

grades.Add(\_patternsKey[i]);

}

else

{

grades.Add(0);

}

if (CheckDiagonalUpRight(\_patternsStrForO[i], move) || CheckHorizontalRight(\_patternsStrForO[i], move) || CheckDiagonalyDownRigth(\_patternsStrForO[i], move) ||

CheckDown(\_patternsStrForO[i], move) || CheckDiagonalyDownLeft(\_patternsStrForO[i], move) || CheckHorizontalLeft(\_patternsStrForO[i], move) ||

CheckDiagonalUplLeft(\_patternsStrForO[i], move) || CheckUp(\_patternsStrForO[i], move))

{

enemyGrades.Add(\_patternsKey[i]);

}

else

{

enemyGrades.Add(0);

}

}

if (grades.Max()>enemyGrades.Max()) //знаходження максимальної ваги ходу (знайдене співпадіння з паттерном який має найбільшу вагу)

return grades.Max();

return enemyGrades.Max();

}

///////////Методи для перевірки співпадіння з паттерном у всих напрямках від можливого ходу///////////

////

private bool CheckDiagonalUpRight(string pattern, Grid move)

{

if (move.indexX + pattern.Length - 1 >= \_gameMap.GetLength(0) || move.indexY - (pattern.Length - 1) >= \_gameMap.GetLength(1) || move.indexY - (pattern.Length - 1) < 0)

return false;

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

{

if ((\_gameMap[move.indexY-i, move.indexX+i].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY - i, move.indexX + i].Text != "" && pattern[i].ToString() != \_gameMap[move.indexY - i, move.indexX + i].Text)

{

return false;

}

}

return true;

}

private bool CheckHorizontalRight(string pattern, Grid move)

{

if (move.indexX + pattern.Length - 1 >= \_gameMap.GetLength(1))

return false;

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

{

if ((\_gameMap[move.indexY, move.indexX + i].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY, move.indexX + i].Text != pattern[i].ToString() && \_gameMap[move.indexY, move.indexX + i].Text != "")

{

return false;

}

}

return true;

}

private bool CheckDiagonalyDownRigth(string pattern, Grid move)

{

if (move.indexX + pattern.Length - 1 >= \_gameMap.GetLength(0) || move.indexY + (pattern.Length - 1) >= \_gameMap.GetLength(1))

return false;

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

{

if ((\_gameMap[move.indexY + i, move.indexX + i].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY + i, move.indexX + i].Text != pattern[i].ToString() && \_gameMap[move.indexY + i, move.indexX + i].Text != "")

{

return false;

}

}

return true;

}

private bool CheckDown(string pattern, Grid move)

{

if (move.indexY + (pattern.Length - 1) >= \_gameMap.GetLength(1))

return false;

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

{

if ((\_gameMap[move.indexY + i, move.indexX].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY + i, move.indexX].Text != pattern[i].ToString() && \_gameMap[move.indexY + i, move.indexX].Text != "")

{

return false;

}

}

return true;

}

private bool CheckDiagonalyDownLeft(string pattern, Grid move)

{

if (move.indexX - (pattern.Length - 1) >= \_gameMap.GetLength(0) || move.indexX- (pattern.Length - 1) < 0 || move.indexY + (pattern.Length - 1) >= \_gameMap.GetLength(1))

return false;

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

{

if ((\_gameMap[move.indexY + i, move.indexX - i].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY + i, move.indexX - i].Text != pattern[i].ToString() && \_gameMap[move.indexY + i, move.indexX - i].Text!="")

{

return false;

}

}

return true;

}

private bool CheckHorizontalLeft (string pattern, Grid move)

{

if (move.indexX - (pattern.Length - 1) >= \_gameMap.GetLength(0) || move.indexX - pattern.Length - 1 <0)

return false;

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

{

if ((\_gameMap[move.indexY, move.indexX - i].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY, move.indexX - i].Text != pattern[i].ToString() && \_gameMap[move.indexY, move.indexX - i].Text != "")

{

return false;

}

}

return true;

}

private bool CheckDiagonalUplLeft(string pattern, Grid move)

{

if (move.indexX - (pattern.Length - 1) >= \_gameMap.GetLength(0) || move.indexY - (pattern.Length - 1) >= \_gameMap.GetLength(1) || move.indexX - pattern.Length - 1<0 || move.indexY - (pattern.Length - 1)<0)

return false;

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

{

if ((\_gameMap[move.indexY - i, move.indexX - i].Text == "" && pattern[i] != ' ') ||

\_gameMap[move.indexY - i, move.indexX - i].Text != pattern[i].ToString() && \_gameMap[move.indexY - i, move.indexX - i].Text != "")

{

return false;

}

}

return true;

}

private bool CheckUp(string pattern, Grid move)

{

if (move.indexY - (pattern.Length - 1) >= \_gameMap.GetLength(1) || move.indexY - (pattern.Length - 1)<0)

return false;

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

{

if ((\_gameMap[move.indexY - i, move.indexX].Text == "" && pattern[i] != ' ') || //////ex

\_gameMap[move.indexY - i, move.indexX].Text != pattern[i].ToString() && \_gameMap[move.indexY - i, move.indexX].Text != "")

{

return false;

}

}

return true;

}

////

///////////Методи для перевірки співпадіння з паттерном у всих напрямках від можливого ходу///////////

private bool IsPossibleMove (Grid a) //метод для оцінювання можливості супротивника/бота походити в цю клітинку

{

int j = a.indexX;

int i = a.indexY;

int hard = 1;

if (i == 0 && j != 0 && j < 19)

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i, j + hard].Text != "" || \_gameMap[i + hard, j].Text != "" ||

\_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i, j - hard].Text != "" ||

\_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i + hard, j - hard].Text != ""))

{

return true;

}

}

else if (i != 0 && j == 0 && i < 19)

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i, j + hard].Text != "" || \_gameMap[i + hard, j].Text != "" ||

\_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i - hard, j + hard].Text != "" ||

\_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i - hard, j + hard].Text != "" ||

\_gameMap[i - hard, j].Text != ""))

{

return true;

}

}

else if (i == 0 && j == 0)

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i, j + hard].Text != "" || \_gameMap[i + hard, j].Text != "" ||

\_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i + hard, j + hard].Text != ""))

{

return true;

}

}

else if (i == 19 && j < 19)

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i, j + hard].Text != "" ||

\_gameMap[i - hard, j + hard].Text != "" || \_gameMap[i - hard, j + hard].Text != "" ||

\_gameMap[i - hard, j].Text != ""))

{

return true;

}

}

else if (j == 19 && i < 19)

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i + hard, j].Text != "" ||

\_gameMap[i, j - hard].Text != "" || \_gameMap[i + hard, j - hard].Text != ""))

{

return true;

}

}

else if (i == 19 && j == 19)

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i, j - hard].Text != "" || \_gameMap[i - hard, j - hard].Text != "" ||

\_gameMap[i - hard, j].Text != ""))

{

return true;

}

}

else

{

if (\_gameMap[i, j].Text == "" && (\_gameMap[i, j + hard].Text != "" || \_gameMap[i + hard, j].Text != "" ||

\_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i, j - hard].Text != "" || \_gameMap[i - hard, j - hard].Text != "" ||

\_gameMap[i - hard, j + hard].Text != "" || \_gameMap[i + hard, j + hard].Text != "" || \_gameMap[i - hard, j + hard].Text != "" ||

\_gameMap[i + hard, j - hard].Text != "" || \_gameMap[i - hard, j].Text != ""))

{

return true;

}

}

return false;

}

}

}

## Player.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace cursachMVC

{

internal class Player : IPlayer //гравець (людина)

{

public string move { get; set; }

public Player(string hod)

{

this.move = hod;

}

public void Move()

{

}

}

}

## IPlayer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace cursachMVC

{

internal interface IPlayer //інтерфейс гравця

{

void Move(); //метод для ходу

string move { get; set; } //поле яке визначає за кого буде грати гравець

}

}

## IController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace cursachMVC

{

internal interface IController //інтерфейс для передавання і виведення даних з гри на форму

{

event EventHandler<EventArgs> OnGridClick;

event EventHandler<EventArgs> OnStopTimer;

string gameMode { get; }

bool isFirstMove { get; set; }

Grid [,] gameMap { get; }

void EnabledButtons(bool status);

void Reset();

void StopTimer();

}

}

## Grid.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace cursachMVC

{

public class Grid : Button

{

public Button button;

public int indexX;

public int indexY;

public Grid(int indexY, int indexX)

{

button = new Button();

button.Size = new Size(25, 25);

this.indexX = indexX;

this.indexY = indexY;

}

}

}

## Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace cursachMVC

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MenuForm());

}

}

}