МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з дисципліна «Основи програмування»

(назва дисципліни)

на тему: «Гра у 15»

Студента 1 курсу, групи ІП-15

Мельника Данила Эвгенійовича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник ст.в. Головченко М. М.

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  | К.т.н.доц. Муха. І.П |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  | Ст.в. Головченко М.М. |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ- 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-15 Семестр 2

**ЗАВДАННЯ**

на курсову роботу студента

Мельник Данило Євгенійович

(прізвище, ім’я, по батькові)

1. Тема роботи «Гра у 15»

2. Строк здачі студентом закінченої роботи 12 червня 2022 року

3.

4.

5.

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 | 18.02.2022 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 05.04.2022 |  |
| 4. | Розробка сценарію роботи програми | 05.04.2022 |  |
| 5. | Узгодження сценарію роботи програми з керівником | 07.04.2022 |  |
| 6. | Розробка (вибір) алгоритму рішення задачі | 07.04.2022 |  |
| 7. | Узгодження алгоритму з керівником | 12.04.2022 |  |
| 8. | Узгодження з керівником інтерфейсу користувача | 12.04.2022 |  |
| 9. | Розробка програмного забезпечення | 10.04.2022 |  |
| 10. | Налагодження розрахункової частини програми | 10.04.2022 |  |
| 11. | Розробка та налагодження інтерфейсної частини програми | 01.05.2022 |  |
| 12. | Узгодження з керівником набору тестів для контрольного прикладу | 10.05.2022 |  |
| 13. | Тестування програми | 19.05.2022 |  |
| 14. | Підготовка пояснювальної записки | 25.05.2022 |  |
| 15. | Здача курсової роботи на перевірку | 12.06.2022 |  |
| 16. | Захист курсової роботи | 19.06.2022 |  |

Студент

(підпис)

Керівник ст.в.Головченко М. М.

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

“12” червня 2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 64 сторінок, 19 рисунків, 22 таблиць, 1 посилання.

Об’єкт дослідження: головоломка «Гра у 15»

Мета роботи: дослідження розв’язання головоломки «Гра у 15»

Виконана програмна реалізація гри та «Гра у 15» та алгоритмів її розвʼязяання, розроблений графічний інтерфейс для взаємодією користувача з ПЗ.

зміст

[Вступ 7](#__RefHeading___Toc7248_755719302)

[1 Постановка задачі 8](#__RefHeading___Toc7250_755719302)

[2 Теоретичні відомості 9](#__RefHeading___Toc7252_755719302)

[2.1 Правила гри «Гра у 15» 9](#__RefHeading___Toc7254_755719302)

[3 Опис алгоритмів 14](#__RefHeading___Toc20322_755719302)

[1.1. Загальний алгоритм 14](#__RefHeading___Toc20324_755719302)

[1.2.Алгоритм випадкової генерації поля 15](#__RefHeading___Toc20326_755719302)

[1.3.Алгоритм зміни розташування поля користувачем 15](#__RefHeading___Toc20328_755719302)

[1.4.Алгоритм розвʼязання пазла 16](#__RefHeading___Toc20330_755719302)

[1.5.Алгоритм заповнення рядка або стовпця відповідними значеннями 16](#__RefHeading___Toc20332_755719302)

[1.6.Алгоритм заповнення позицій 10 та 14 17](#__RefHeading___Toc20334_755719302)

[1.7.Алгоритм заповнення останніх трьох клітинок 17](#__RefHeading___Toc20336_755719302)

[1.8.Алгоритм переміщення клітинки у певну позицію 17](#__RefHeading___Toc20338_755719302)

[1.9.Алгоритм переміщення пробіла у певну позицію 18](#__RefHeading___Toc20340_755719302)

[1.10.Алгоритм знаходження найкоротшого шляху у графі 18](#__RefHeading___Toc20342_755719302)

[4 Опис програмного забезпечення 20](#__RefHeading___Toc7272_755719302)

[4.1 Діаграма класів програмного забезпечення 20](#__RefHeading___Toc7274_755719302)

[4.2 Опис методів частин програмного забезпечення 21](#__RefHeading___Toc7276_755719302)

[4.2.1 Стандартні методи 22](#__RefHeading___Toc7278_755719302)

[4.2.2 Користувацькі методи 24](#__RefHeading___Toc7280_755719302)

[5 План тестування 44](#__RefHeading___Toc7282_755719302)

[5.1 План тестування 44](#__RefHeading___Toc7284_755719302)

[5.2 Приклади тестування 45](#__RefHeading___Toc7286_755719302)

[6 Інструкція користувача 56](#__RefHeading___Toc7288_755719302)

[6.1 Робота з програмою 56](#__RefHeading___Toc7290_755719302)

[6.2 Формат вхідних та вихідних даних 60](#__RefHeading___Toc7292_755719302)

[6.3 Системні вимоги 60](#__RefHeading___Toc7294_755719302)

[Висновки 62](#__RefHeading___Toc7296_755719302)

[Перелік посилань 63](#__RefHeading___Toc7298_755719302)

[ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ 64](#__RefHeading___Toc7300_755719302)

[Додаток Б Тексти програмного коду 67](#__RefHeading___Toc7302_755719302)

Вступ

Завданням моєї курсової є програмна реалізація гри «Гра у 15» та реалізація алгоритму розвʼязування цієї гри. Вона має прості правила та чітке представлення.

З точки зору програмування та створення алгоритмів вона обʼєднує найцікавіші характеристики: математичну базу, зовнішню простоту та невизначену складність при пошуку оптимального розвʼязання

У своїй роботі я намагався приділити увагу не глибшому зануренню у нескінченний шлях оптимізації цієї задачі, а швидше зручності використання програми та надаванню фундаменту користувачеві, який можливо щойно познайомився з цією грою. Для цього я реалізував алгоритм подібний до того, як пазл розвʼязує людина, поступово заповнюючи окремі ділянки поля потрібними значеннями.

# Постановка задачі

Розробити програмне забезпечення, що буде реалізовувати гру «Гра у 15», режим автоматичної гри для неї та виконання наступного кроку.

Вхідними даними для даної роботи є згенероване випадковим чином або задане користувачем поле для гри, що може бути розвʼязаним.

Програма має перевіряти дані на можливість розвʼязку та за потреби корегувати їх, якщо матриця була згенерована, або запитувати нові дані, якщо введені користувачем були некороктними.

Під час гри має бути можливість увімкнення та вимкнення режиму автоматичної гри, під час якої програма самостійно розвʼязує пазл, а також можливість виконання наступного кроку при натисканні відповідної кнопки.

Після складання пазла програма має надавати можливість зберегти дані про цю гру у файл.

Вихідними даними програми є повідомлення про перемогу у випадку успішного складання пазлу та дані, що зберігаються у файл після цього.

# Теоретичні відомості

## Правила гри «Гра у 15»

«Гра у 15» (або «Пʼятнашки») являє собою поле 4 на 4 клітинки. 15 із них заповнені натуральними числами від 1 до 15, а 16-та залишається порожньою. Гравець може переміщати клітинку поруч із порожньою на її місце, тим самим звільняючі свою. Метою гри є заповнення поля клітинками у відсортованому порядку (рисунок 2.1)[1].

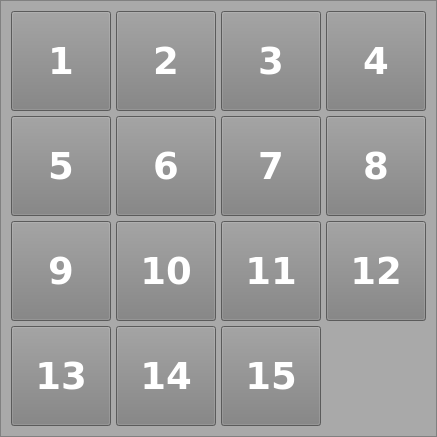


Рисунок 2.1 – Поле для гри у відсортованому стані

2.2 Розвʼязуваність ігрового поля

Поле не завжди можна привести до фінального стану для довільного початкового розташування. Для того, щоб пазл міг бути складений, ми маємо визначити інваріант, як парність суми кількості інверсій послідовності, що відповідає порядку розташування заповнених клітинок та відстанню від пробілу до останнього рядка. Якщо даний інваріант є дійсним, то поле може бути розвʼязаним[1].

Доведення:

Якщо пробіл переміщується горизонально, сама послідовність зповнених клітинок і кількість інверсій в ній залишаються сталими. Пробіл при цьому залишається у тому ж самому рядку, а отже інваріант не змінюється

При вертикальному переміщенні пробіла число, представлене клітинкою, яка змінила своє розташування, зміститься на чотири позиції у послідовності, що розглядається. Це змінить її відносний порядок із трьома іншими числами, які змінили своє розташування у послідовності, що призведе до зміни кількості інверсій на непарне число. Оскільки відстань від останнього рядка поля до пробілу зміниться на один, інваріант залишиться сталим.

Оскільки даний інваріант відповідає фінальному розташуванню, то він буде виконуватися і для всіх початкових станів поля, які можуть бути розвʼязанимим.

2.3 Алгоритм розвʼязання пазла

Поле може бути розвʼязаним почерговим заповненням його рядків та стовпців, починаючи з верхнього рядка (рис. 2.2).

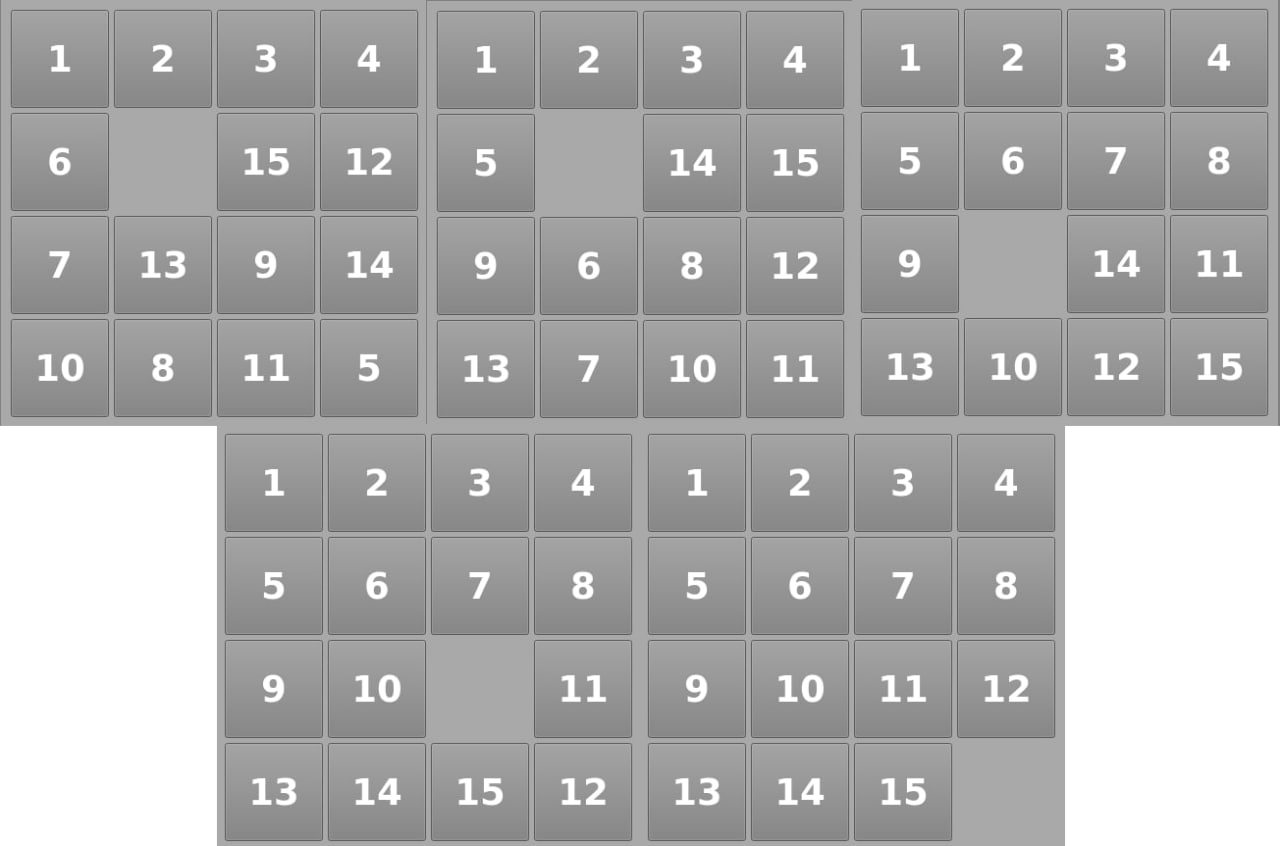


Рисунок 2.2 – Послідовність заповнення рядків і стовпців поля

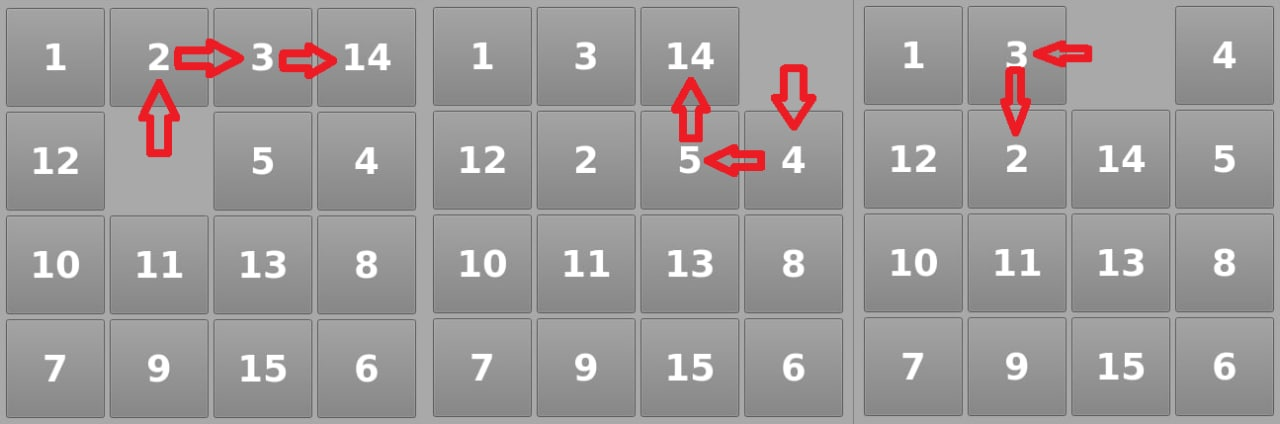
Для заповнення двох верхніх рядків на першого стовпця можна виконати наступну послідовність дій: перемістити по черзі усі клітинки рядка чи стовпця (починаючи зліва чи зверху) окрім останньої на свої місця у кінцевому розташуванні, не змінюючи при цьому раніше виставлені клітинки.

Наступним кроком, якщо останню клітинку не можна одразу поставини на своє фінальне місце, необхідно перемістити її у позицію поруч із нею (рис 2.3).



Рисунок 2.3 – Розташування поля перед переміщенням останнього елемента першого рядка на своє місце

Алгоритм встановлення останнього елементу зображено на рисунку 2.4.

Рисунок 2.4 – Переміщення останнього елемента рядка на своє місце

Даний спосіб можна використати для заповнення першого рядка, першого стовпця (необхідно віддзеркалити переміщення по головній діагоналі) та другого рядка, оскільки він не порушить розташування раніше розміщених клітинок.

Для заповнення другого стовпця, а саме клітинок 10 та 14 можна спочатку поставити 14 на кінцеве місце 10, а далі спробувати поставити 10 на місце 151 Може виникнути ситуація, коли це буде проблемно (рис. 2.5).



Рисунок 2.5 – розташування 10, яке потребує додаткових переміщень для заповнення другого стовпця

Для того, що поставити клітинку 10 на позицію, що відповідає кінцевому розташуванню 11 (положення клітинки 12 на рис. 2.5), потрібно виконати переміщення, показані на рисунку 2.6.

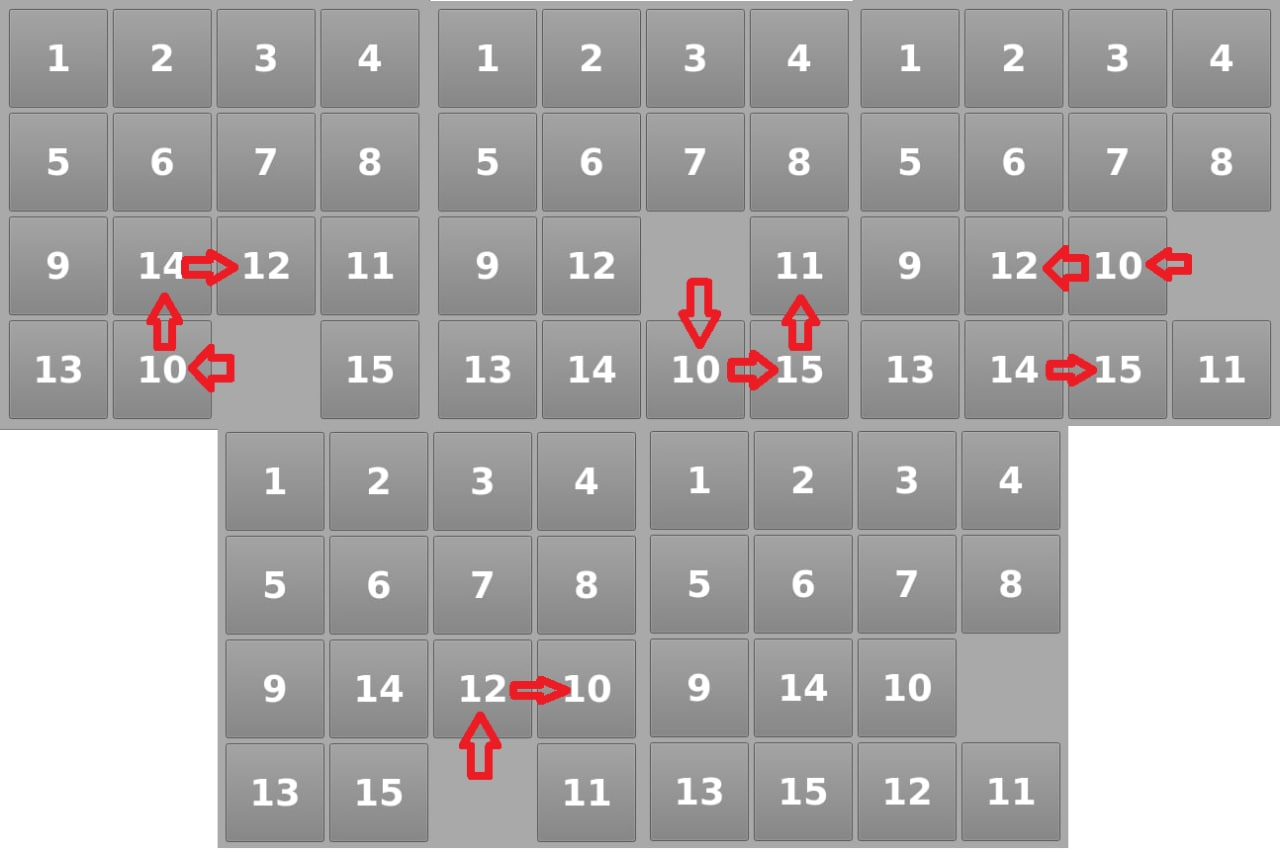


Рисунок 2.5 – Переміщення 10 на позицію 11

Після того, як 10 опинилася на позиції 11, можна легко заповнити другий стовпець, якщо не переміщувати 10, допоки 14 не буде на своєму місці.

Далі заповнення поля зведеться до підстановки трьої клітинок у квадраті 2 на 2 на свої місця. Для цього потрібно спершу поставити на своє місце 11, а вже потім 12 і 15 (рис. 2.6).



Рисунок 2.6 – Приклад підстановки на свої місця останніх двох клітинок

Після того, як 11, 12 і 15 опиняться на своїх місцях, ми отримаємо стан поля, зображений на рисунку 2.1, який відповідає виграшному.

# Опис алгоритмів

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначенням

|  |  |
| --- | --- |
| Змінна | Призначення |
| seq | послідовність дій для розвʼязку задачі |
| reord\_cell | Зберігає клітинку, яку користувач може поміняти місцями при створенні початкового поля вручну |
| along | Напрям уздовж |
| across | Напрям перпендикулярно |
| top | Позиція, що відповідає передостанньому рядку на третьому з кінця стовпцю |
| bot | Позиція, що відповідає останньому рядку на третьому з кінця стовпцю |
| p | Масив попередніх вершин у шляху |
| v | Поточна вершина |

## Загальний алгоритм

1. ПОЧАТОК
2. Зчитати спосіб задавання початкового стану поля.
3. ЯКЩО генерація випадковим чином ТО викликати метод випадкової генерації
4. ЯКЩО створення вручну ТО викликати метод зміни порядку користувачем
5. Створити масив seq
6. ДОПОКИ seq не порожній
   1. ЯКЩО користувач наниснув на клітинку поруч із порожньою ТО перемістити її на порожнє місце
      1. видалити перший елемент seq
      2. ЯКЩО переміщення не відповідає йому ТО створити новий масив seq
   2. ЯКЩО користувач натиснув на кнопку наступного кроку ТО виконати наступне переміщення розвʼязку
   3. ЯКЩО користувач натиснув на кнопку автоматичного розвʼязування
      1. ДОПОКИ seq не порожній ТА користувач не натискає кнопку зупинки виконувати наступне переміщення розвʼязку
7. КІНЕЦЬ

## Алгоритм випадкової генерації поля

1. ПОЧАТОК
2. Створити випадково згенерований масив
3. ЯКЩО парність рядка пробіла та парність кількості інверсій решти клітинок збігаються ТО змінити місцями пробіл та клітинку поруч із сусідньою у тому ж рядку
4. КІНЕЦЬ

## Алгоритм зміни розташування поля користувачем

1. ПОЧАТОК
2. ДОПОКИ користувач не натисне кнопку почати гру ТА поле буде відмінне від виграшного ТА його можна буде розвʼязати
   1. ЯКЩО користувач натиснув на клітинку
      1. ЯКЩО reord\_cell зберігає клітинку ТО поміняти її місцями з попередньою та очистити reord\_cell
      2. ІНАКШЕ зберегти цю клітинку в reord\_cell
3. КІНЕЦЬ

## Алгоритм створення масиву послідовності дій для розвʼязання пазла

1. ПОЧАТОК
2. Створити граф, в якому вершини відповідають позиціям у полі, а ребра поєднують ті із них, що мають спільні сторони
3. Послідовно заповнити відповідними значеннями верхній рядок, лівий стовпець та другий рядок згори, додаючи переміщення пробілу у seq
4. Заповнити відповідними значеннями останні пʼять позицій, додаючи усі переміщення пробілу у seq
5. КІНЕЦЬ

## Алгоритм заповнення рядка або стовпця відповідними значеннями

1. ПОЧАТОК
2. ЯКЩО заповнюється рядоок ТО присвоїти along напрям вправо, присвоїти across напрям вниз
3. ЯКЩО заповнюється стовпець ТО присвоїти along напрям вниз, присвоїти along напрям вправо
4. ЯКЩО лінія заповнена ТО видалити із графу вершини, що відповідають позиціям лінії та завершити виконання функції
5. ІНАКШЕ
   1. ЦИКЛ перебору всіх клітинок рядка, починаючи із напрямка протилежного along, окрім останньої
      1. Перемістити клітинку на своє місце
      2. Видалити відповідну вершину із графу
   2. Перемістити останню клітинку у відповідну їй позицію, зміщену у напрямку across
   3. Перемістити пробіл на позицію, зміщену на across від другої з кінця у лінії, що заповнюється, не переміщуючи при цьому останню клітинку
   4. Виконати послідовність переміщень для пробіла: -across, along, across, -along, -across, -along, across
   5. Видалити із графу вершину, що відповідає останній клітинці у лінії
6. КІНЕЦЬ

## Алгоритм заповнення двох лівих позицій із прямокутника 2 на 3, що залишився після виконання функцій, що заповнюють рядки зверху та стовпці зліва

1. ПОЧАТОК
2. ЯКЩО ці дві клітинки на на своїх місцях
   1. Присвоїти змінній bot значення позиції нижньої із них
   2. Присвоїти змінній top значення позиції верхньої із них
   3. Перемістити клітинку, що відповідає bot у позицію top
   4. Перемістити пробіл на позицію справа від позиції bot
   5. ЯКЩО у позиції bot знаходиться клітика, що відповідає top ТО виконати наступну послідовність переміщень для пробіла: вліво, вгору, вправо, вниз, вправо, вгору, вліво, вліво, вниз
   6. Перемістити клітинку, що відповідає top у позицію справа від top
   7. Перемістити клітинку, що відповідає позиції bot, у відповідну їй позицію, не переміщуючи клітинку, що відповідає top
   8. Перемістити пробіл вправо, поставивиши тим самим у top відповідне значення
3. Видалити з графу вершини, що відповідають top та bot
4. КІНЕЦЬ

## Алгоритм заповнення останніх трьох клітинок

1. ПОЧАТОК
2. Заповнити позицію зліва та зверху від правого нижнього кута поля відповідною клітинкою
3. Заповнити позицію зверху від правого нижнього кута поля відповідною клітинкою
4. Заповнити позицію зліва від правого нижнього кута поля відповідною клітинкою
5. КІНЕЦЬ

## Алгоритм переміщення клітинки у певну позицію

1. ПОЧАТОК
2. Знайти найоротший шлях від поточної клітинки до кінцевої позиції, використовуючи пошук вшир у графі
3. ЦИКЛ перебору усіх переходів у цьому шляху
   1. Виключити вершину, що відповідає поточній позиції клітинки, що переміщується, із графу
   2. Перемістити пробіл у поточну позицію у шляху
   3. Повернути останню вилучену вершину у граф
   4. Перемістити пробіл на місце клітинки, що переміщується
4. КІНЕЦЬ

## Алгоритм переміщення пробіла у певну позицію

1. ПОЧАТОК
2. Знайти найоротший шлях від пробіла до кінцевої позиції, використовуючи пошук вшир у графі
3. ЦИКЛ перебору усіх переходів у цьому шляху
   1. перемістити пробіл у поточну позицію шляху
4. КІНЕЦЬ

## Алгоритм знаходження найкоротшого шляху між вершинами у графі за доппомогою пошуку вшир

1. ПОЧАТОК
2. Створити чергу
3. Cтворити масив p попередніх вершин у шляху для кожної вершини та позначити їх як непройдені
4. Відмітити початкову вершину, як таку, що не має попередньої
5. Ініціалізувати поточну вершину v початковою вершиною
6. ДОПОКИ v не є кінцевою вершиною
   1. ЦИКЛ перебору сусідніх вершин зі списку суміжності для v
      1. ЯКЩО сусідня вершина не перевірена ТО додати її у кінець черги та вказати вершину v як попередню їй
   2. Замінити v на вершину, вилучену з початку черги
7. Створити масив результату
8. ДОПОКИ v не є початковою вершиною
   1. Додати в кінець масиву результату v
   2. Замінити вершину v на попередню для неї
9. Змінити порядок у масиві результату на зворотній
10. КІНЕЦЬ

## Т

1. ПОЧАТОК
2. Заповнити позицію зліва та зверху від правого нижнього кута поля відповідною клітинкою
3. Заповнити позицію зверху від правого нижнього кута поля відповідною клітинкою
4. Заповнити позицію зліва від правого нижнього кута поля відповідною клітинкою

КІНЕЦЬ

## Т

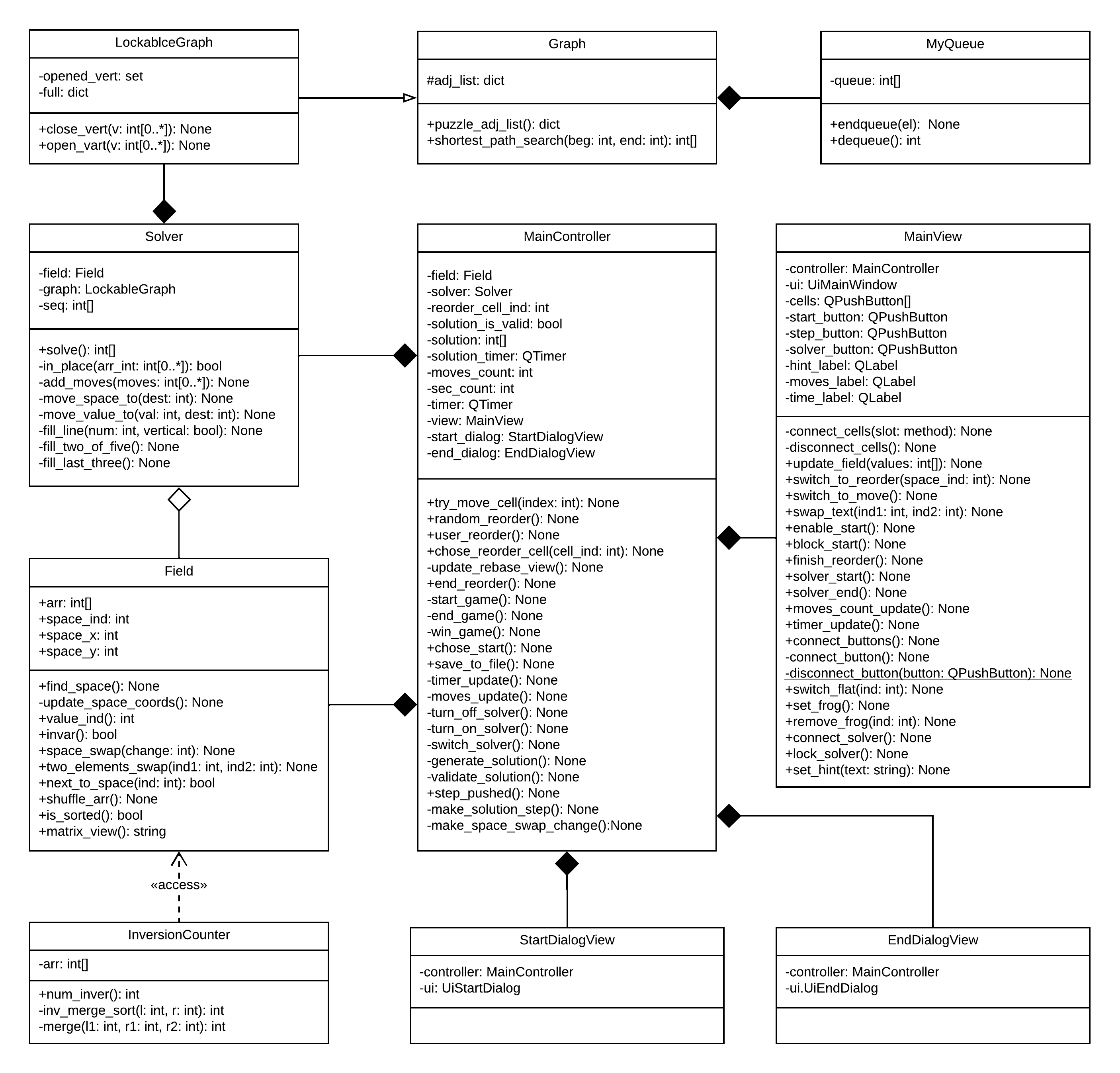
1. ПОЧАТОК
2. Заповнити позицію зліва та зверху від правого нижнього кута поля відповідною клітинкою
3. Заповнити позицію зверху від правого нижнього кута поля відповідною клітинкою
4. Заповнити позицію зліва від правого нижнього кута поля відповідною клітинкою

КІНЕЦЬ

# Опис програмного забезпечення

## Діаграма класів програмного забезпечення

Діаграма класів розробленого програмного забезпечення наведена на рисунку 4.1.

* + - * 1. Діаграма класів

У своїй програмі я реалізував шаблон проектування Модель-Представлення-Предʼявлення. Роль Моделі виконує клас Field, Представлення – клас MainView, Предʼявника – клас MainController.

MainView – клас головного вікна, який звʼязується з класом MainController при введенні даних та виводить їх при виклику відповідних методів із нього. Він є представленням обʼєкту класу Field, хоча і не має прямого доступу до нього.

StartDialogView та EndDialogView – класи діалового вікна початку та кінця гри відповідно. Вони слугують для передачі даних про вибір задавання поля та збереження даних гри у файл класу MainController.

Клас Field відповідає за зберігання, та доступ до даних, які представляють поточний стан поля. MainController за потреби запитує доступ до цих даних та, в залежності від результату їх обробки, маніпулює їх представленням у MainView. Field використвує клас InversionCounter для перевірки поля на можливість бути розвʼязаним. Він використовує три методи які, обробляючи вміст масиву поля, повертають інформацію про кількість інверсій його непорожніх клітинок.

Клас Solver відповідає за реалізацію алгоритму пошуку розвʼязання ігрового поля, у певному його стані. Під час обробки даних, що зберігає Field, він маніпулює копією обʼєкту цього класу та повертає дані, з допомогою яких MainController може виконати розвʼязання гри, маніпулюючи Моделлю та Представленням.

Під час виконання алгоритму Solver використовує обʼєкт класу LockableGraph, який наслідується від класу Graph. Graph зберігає список суміжності, що представляє усі вершини графу та сусідні ребра для кожної із них, а такоє включає в себе метод пошуку найкоротшого шляху між двома вершинами, який використовує пошук вшир. Lockable graph надає можливість видаляти вершини з графу та повертати їх. Це реалізовано за рахунок параметрів, що зберігають початковий масив вершин графу, масиву вершин, доступних на даний момент та методів видалення та повернення вершин.

Для виконання пошуку вшир Graph використовує чергу, представлену обʼєктом класу Queue, що надає можливість додавати елемент у кінець та вилучати з початку масиву.

## Опис методів частин програмного забезпечення

### Стандартні методи

У таблиці 4.1 наведено стандартні методи, які були використані для реалізації поставленої задачі.

Стандартніметоди

| № п/п | Назва заголовного файлу | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| --- | --- | --- | --- | --- | --- | --- |
| 1 |  | UiMainWindow | setupUi | Створення віджетів головного вікна та визначення їх початкових характеристик |  |  |
| 2 |  | QAbstractButton | setText | Встановлення тексту у віджет | Текст, що буде встановлений |  |
|  |  | QLabel | setText | Встановлення тексту у віджет label | Текст, що буде встановлений |  |
| 3 |  | QAbstractButton | text | Отримення тексту кнопки |  | Текст кнопки |
| 4 |  | QPushButton | setFlat | Визначення параметра Flat кнопки | Стан Flat |  |
| 5 |  | QPushButton | isFlat | Отримання значення Flat кнопки |  | Булеве значення, що відповідає параметру Flat |

Продовження таблиці 4.1

| № п/п | Назва заголовного файлу | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| --- | --- | --- | --- | --- | --- | --- |
| 6 |  | QWidget | setStyleSheet | Визначення параметра StyleSheet віджета | Опис параметра StyleSheet |  |
| 7 |  | QTimer | start | Запуск таймера із вказаним інтервалом | Кількість мілісекунд між виконанням підключеного методу |  |
| 8 |  | QTimer | isActive | Визначення чи таймер запущено | Булеве значення, що відповідає стану таймера |  |
| 9 |  | IO | write | Запис рядка у файл | Рядок, який буде записан у файл |  |
| 10 | time |  | strftime | Конвертування даних про локальний час у рядок | Рядок для форматування, дані про час | Відформатований рядок |
| 11 | time |  | localtime | Отримання локального часу | Час у секундах | Структура з даними локального часу |

Продовження таблиці 4.1

| № п/п | Назва заголовного файлу | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| --- | --- | --- | --- | --- | --- | --- |
| 12 | time |  | time | Отримання Unix Time |  | Unix Tyme у секундах |
| 13 | copy |  | deepcopy | Створює копію обʼєкта | Обʼєкт | Копія вхідного обʼєкту |
| 14 | random |  | shuffle | Змінює порядок обʼєктів у масиві на випадковий | Масив |  |
| 15 |  | list | index | Повертає індекс елемента у списку | Елемент | Індекс у списку |
| 16 |  | list | append | Додає елемент у кінець списку | Елемент |  |
| 17 |  | list | pop | Видаляє елемент зі списку за індексом та повертає його | Індекс елемента | Елемент |
| 18 |  | QObject | disconnect | відключає сигнал від функції, повʼязаної із обʼєктом |  |  |
| 19 |  | pyqtBoundSignal | connect | підключає сигнал до функції | Функція, що підключається |  |

### Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, які були застосовані для реалізації задачі.

Таблиця 4.2 – Користувацькі методи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва заголовного файлу | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 |  | MainController | try\_move\_cell | Переміщення клітинки за можливості та перевірка чи не призвело воно до перемоги | Індекс клітинки, що переміщується |  |
| 2 |  | MainController | random\_reorder | Початок нової гри з випадково згенерованим полем |  |  |
| 3 |  | MainController | user\_reorder | Перехід до режиму перестановки клітинок |  |  |
| 4 |  | MainController | chose\_reorder\_cell | Зміна розташування клітинок у режимі перестановки клітинок | Індекс однієї із клітинки, що переставляються |  |

Продовження таблиці 4.2

| № п/п | Назва заголовного файлу | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| --- | --- | --- | --- | --- | --- | --- |
| 5 |  | MainController | update\_rebase\_view | Оновлення інтерфейсу при користувацькому задаванн |  |  |
| 6 |  | MainController | end\_reorder | Вихід із режиму перестановок |  |  |
| 7 |  | MainController | start\_game | Початок гри |  |  |
| 8 |  | MainController | end\_game | Завершення гри |  |  |
| 9 |  | MainController | win\_game | Сповіщення користувача про перемогу. Відкривання вікна, запиту на збереження у файл |  |  |
| 10 |  | MainController | chose\_start | Передчасне завершення гри та відкривання вікна вибору режиму задавання поля |  |  |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 11 |  | MainController | save\_to\_file | Збереження інформації про гру у файл |  |  |
| 12 |  | MainController | time\_update | Оновлення таймера |  |  |
| 13 |  | MainController | moves\_update | Оновлення лічильника кількості кроків |  |  |
| 14 |  | MainController | turn\_off\_solver | Вихід із режиму автоматичного розвʼязування |  |  |
| 15 |  | MainController | turn\_on\_solver | Перехід у режим автоматичного розвʼязування |  |  |
| 16 |  | MainController | switch\_solver | Переключення режиму автоматичного розвʼязування |  |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 17 |  | MainController | generate\_solution | Отримання розвʼязку поточного стану поля |  |  |
| 18 |  | MainController | validate\_solution | Оновлення розвʼязок, якщо він неактуальний |  |  |
| 19 |  | MainController | step\_pushed | Виконання наступний крок розвʼязку |  |  |
| 20 |  | MainController | make\_solution\_step | Оновлення розвʼязання та виконання наступного кроку розвʼязку при  натисканні на кнопку |  |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 21 |  | MainController | make\_space\_swap | Виконання переміщення пробілу у певному напрямку | Ціле число, що відповідає напрямку переміщення |  |
| 22 |  | MainView | connect\_cells | Підключення усіх клітинок поля до певного методу | Метод, який буде підключено |  |
| 23 |  | MainView | disconnect\_cells | Відключення усіх клітинок поля від підключеного методу |  |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 24 |  | MainView | update\_field | Оновлює відображення поля у вікні | Цілочисельний масив: оновлений  масив ігрового поля |  |
| 25 |  | MainView | switch\_to\_reorder | Зміна методу, підключеного до клітинок, для переходу в режим перестановок | Індекс пробіла |  |
| 26 |  | MainView | switch\_to\_move | Зміна методу, підключеного до клітинок, для переходу в режим гри |  |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 27 |  | MainView | swap\_text | Обмін текстом між двома клітинками | Індекс першої клітинки; Індекс другої клітинки |  |
| 28 |  | MainView | enable\_start | Підключення до кнопки початку гри методу завершення режиму перестановки |  |  |
| 29 |  | MainView | block\_start | Відключення від кнопки початку гри поточного методу |  |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 30 |  | MainView | finish\_reorder | Переведення інтерфейсу в ігровий режим |  |  |
| 31 |  | MainView | solver\_start | Зміна тексту кнопки автоматичного розвʼязання на «Зупинити» |  |  |
| 32 |  | MainView | solver\_end | Зміна тексту кнопки автоматичного розвʼязування на «Автоматичне розвʼязування» |  |  |
| 33 |  | MainView | moves\_count\_update | Оновлення відображення ліч | Нове значення лічильника |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 34 |  | MainView | timer\_update | Оновлення відображення таймера | Нове значення кількості секунд, що минули |  |
| 35 |  | MainView | connect\_buttons | Підключення усіх кнопок інтерфейсу (окрім кнопок ігрового поля) |  |  |
| 36 |  | MainView | connect\_button | Підключення кнопки до сигналу з відключенням поточного, якщо він присутній | кнопка та метод, що підключається |  |
| 37 |  | MainView | disconnect\_button | Підключення кнопки від сигналу, якщо він підключений | Кнопка |  |

Продовження таблиці 4.2

| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| --- | --- | --- | --- | --- | --- | --- |
| 38 |  | MainView | switch\_flat | Переключення стану Flat кнопки за індексом | Індекс кнопки |  |
| 39 |  | MainView | set\_frog | Зміна тексту порожньої клітинки на «🐸», щоб повідомити користувача про те, що пазл зібрано |  |  |
| 40 |  | MainView | remove\_frog | Встановлення порожнього рядка у якості тексту кнопки за вказаним індесом | Індекс клітинки |  |
| 41 |  | MainView | connect\_solver | Підключення кнопок наступного крокута автоматичного розвʼязування |  |  |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 42 |  | MainView | lock\_solver | Відключення кнопок наступного кроку та автоматичного розвʼязування |  |  |
| 43 |  | MainView | set\_hint | Заміна тексту напису-підказки на вказаний | Новий текст підказки |  |
| 44 |  | Field | find\_space | Знаходження індексу пробілу |  |  |
| 45 |  | Field | update\_space\_coords | Оновлення даних про стовпець та рядок пробілу |  |  |
| 46 |  | Field | value\_ind | Знаходження індексу клітинки із вказаним значенням | Значення клітинки | Ідекс клітинки |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 47 |  | Field | invar | Перевірка, чи може поле бути розвʼязаним |  | Булеве значення, що відповідає коректності заданаго поля |
| 48 |  | Field | space\_swap | Переміщення пробілу у вказаному напрямку, вираженим числом | Зміна індекса пробіла для одновимірного масиву |  |
| 49 |  | Field | two\_elements\_swap | Зміна місцями двох клітинок та оновлення даних про розташування пробілу | Індекс першої клітинки; Індекс другої клітинки |  |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 50 |  | Field | next\_to\_space | Перевірка, чи знаходиться клітинка за вказаним індексом поруч із пробілом | Індекс клітинки | відповідає розташуванню поруч із пробілом |
| 51 |  | Field | shuffle\_arr | Зміна розташування клітинок поля на випадкову відмінну від початкової та відсортованої |  |  |
| 52 |  | Field | is\_sorted | Перевірка, чи відсортоване поле |  | Булеве значення, що відповідає відсортованості поля |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 53 |  | Field | matrix\_view | Створення рядка, що представляє поле у вигляді матриці |  | Рядок: масив поля представлений у вигляді матриці |
| 54 |  | Solver | solve | Генерує масив-розвʼязання |  | Послідовність переміщень пробілу, представлених у вигляді цилих чисел |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 55 |  | Solver | in\_place | Перевіряє, чи знаходяться на вказаних позиціях клітинки, що відповідають фінальному розташуванню | Цілочисельний масив: індекси позицій поля | Булеве значення: відповідає розташуванню відповідних значень в усіх переданих позиціях |
| 56 |  | Solver | add\_moves | Додавання переміщень, представлених цілими числами у масив | Переміщення |  |
| 57 |  | Solver | move\_space\_to | Переміщення пробілу в указану позицію | Індекс позиції-цілі |  |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 58 |  | Solver | move\_value\_to | Переміщення клітинки із вказаним значенням у вказану позицію | Значення клітинки; Кінцева позиція |  |
| 59 |  | Solver | fill\_line | Заповнення лінії під вказаним номером на із вказаною орієнтацією відповідними значення | Номер рядка/стовпця; Булеве значення відповідає вертикальній орієнтації |  |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 60 |  | Solver | fill\_two\_of\_five | Заповнення відповідними значеннямидвох лівих клітинок прямокутника 2 на 3 в правому нижньому кутку поля |  |  |
| 61 |  | Solver | fill\_last\_three | Заповнення відповідними значеннями квадрата 2 на 2 у правому нижньому кутку поля |  |  |
| 62 |  | Graph | puzzle\_adj\_list | Повернення списку суміжності для графа, що відповідає клітинкам ігрового поля та їх спільним сторонам |  | Словник: список суміжності |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 63 |  | Graph | shortest\_path\_search | Повернення послідовність вершин між початковою та кінцевою, що відповідає одному із найкоротших шляхів між ними |  | Цілочисельний масив: найкоротший шлях між вершинами |
| 64 |  | LockableGraph | close\_vert | Видалення вершини із графу |  |  |
| 65 |  | LockableGraph | open\_vert | Повернення у граф, раніше видаленої вершини |  |  |
| 66 |  | MyQueue | enqueue | Додавання елемента в кінець черги | Новий елемент |  |
| 67 |  | MyQueue | dequeue | Видалення елемента із початку черги |  | Видалений елемент |
| 68 |  | InversionCounter | num\_inver | Підрахунок кількості інверсій у масиві |  | Клькість інверсій |

Продовження таблиці 4.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | **Назва заголовного файлу** | **Назва классу** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** |
| 69 |  | InversionCounter | inv\_merge\_sort | Рекурсивний обрахунок кількості інверсій, використовуючи сортування злиттям | Ліва межа підмасиву; Права межа підмасиву | Кількість інверсій для вказаного підмасиву |
| 70 |  | InversionCounter | merge | Злиття масивівпри сортуванні з поверненням кількості перестановок | ліва межа лівого підмасиву; Права межа лівого підмасиву; Права межа правого підмасиву | Кількістьперестановок |

# План тестування

## План тестування

1. Пересування клітинок
   1. Тестування роботи програми при натисканні на порожню клітинку
   2. Тестування роботи програми при натисканні на клітинку сусідню із порожньою
   3. Тестування роботи програми при натисканні на клітинку віддалену від порожньої
   4. Тестування роботи програми при натисканні на клітинку, яке призводить до відсортованого стану поля, під час гри
2. Натискання на кнопку «Почати нову гру»
   1. Тестування роботи програми при натисканні на кнопку під час гри, коли автоматичне розвʼязання вимкнене
   2. Тестування роботи програми при натисканні на кнопку під час гри, коли автоматичне розвʼязання увімкнене
   3. Тестування роботи програми при натисканні на кнопку поза грою
   4. Тестування роботи програми під час користувацького задавання поля, коли поле може бути розвʼязаним
   5. Тестування роботи програми під час користувацького задавання поля, коли поле не може бути розвʼязаним
3. Натискання на кнопки «Наступний крок» та «Автоматичне розвʼязування»
   1. Тестування роботи програми при натисканні на кнопки поза грою
   2. Тестування роботи програми при натисканні на кнопки під час гри, коли автоматичне розвʼязування вимкнене
   3. Тестування роботи програми при натисканні на кнопки під час гри, коли автоматичне розвʼязування увімкнене
4. Робота діалогових вікон
   1. Тестування роботи програми при натисканні на кнопку випадкової генерації поля
   2. Тестування роботи програми при натисканні на кнопку генерації поля вручну
   3. Тестування роботи програми при натисканні на кнопку підтвердження збереження даних у файл
   4. Тестування роботи програми при натисканні на кнопку відмови від збереження даних у файл
5. Режим користувацького задавання поля
   1. Тестування першого натиску
   2. Тестування другого натиску

## Приклади тестування

Результати тестувань наведено у таблицях 5.1 – 5.22

Тестування роботи програми при натисканні на порожню клітинку

|  |  |
| --- | --- |
| Мета тесту | Перевірка реакції програми на натискання на порожню клітинку |
| Початковий стан програми | Стан поля на рисунку 5.1 |
| Вхідні дані | Інформація про натискання на порожню клітинку |
| Схема проведення тесту | Натискання на порожню клітинку |
| Очікуваний результат | Розташування поля не змінилося |
| Стан програми після проведення випробувань | Збігається з очікуваним |



Рисунок 5.1 – Початковий стан поля при тестуванні, що наведене у таблиці 5.1

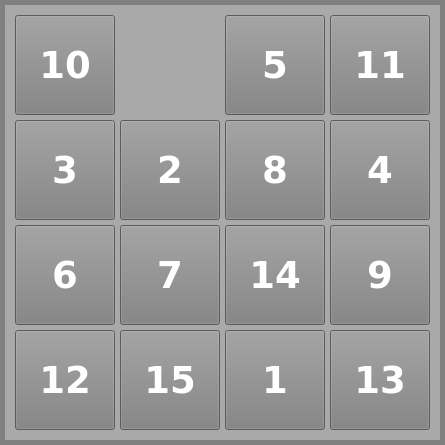
Тестування роботи програми при натисканні на клітинку сусідню із порожньою

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість переміщувати клітинки поля |
| Початковий стан програми | Стан поля на рисунку 5.2 |
| Вхідні дані | Інформація про натискання на клітинку 5 |
| Схема проведення тесту | Натискання на клітинку 5 |
| Очікуваний результат | Клітинка 5 обмінялася місцями з порожньою |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Рисунок 5.2 – Початковий стан поля при тестуванні, що наведене у таблиці 5.2

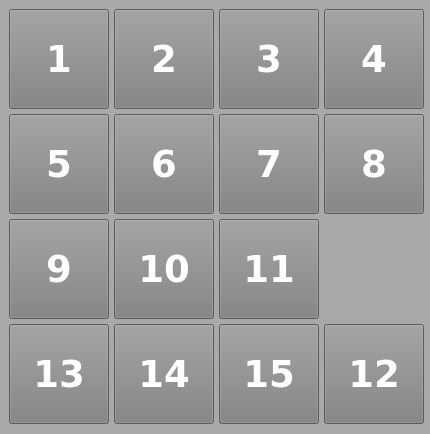
Тестування роботи програми при натисканні на клітинку віддалену від порожньої

|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу програми при спробі перемістити клітинку, яка не може бути переміщена |
| Початковий стан програми | Стан поля на рисунку 5.3 |
| Вхідні дані | Інформація про натискання на клітинку 14 |
| Схема проведення тесту | Натискання на клітинку 14 |
| Очікуваний результат | Розташування поля не змінилося |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Рисунок 5.3 – Початковий стан поля при тестуванні, що наведене у таблиці 5.3

Тестування роботи програми при натисканні на клітинку, яке призводить до відсортованого стану поля

|  |  |
| --- | --- |
| Мета тесту | Перевірка сповіщення гравця про перемогу |
| Початковий стан програми | Стан поля на рисунку 5.4  Кнопки автоматичної гри на наступного кроку активні, працює таймер |
| Вхідні дані | Інформація про натискання на клітинку 12 |
| Схема проведення тесту | Натискання на клітинку 12 |
| Очікуваний результат | Клітинка 12 обмінялася місцями з порожньою. Зʼявилося вікно, що пропонує зберегти результат у файл. Кнопки автоматичної гри та наступного кроку відключилися та змінили колір. Таймер зупинився |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Рисунок 5.4 – Початковий стан поля при тестуванні, що наведене в табл 5.4

Тестування роботи програми при натисканні на кнопку нової гри під час гри, коли автоматичне розвʼязання вимкнене

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості перезапуску поточної гри з вимкненим режимом автоматичної гри |
| Початковий стан програми | Працює таймер, кнопки автоматичної гри та наступного кроку активні |
| Вхідні дані | Інформація про натискання на кнопку нової гри |
| Схема проведення тесту | Натискання на кнопку нової гри |
| Очікуваний результат | Таймер зупиняється та обнуляється, обнуляється лічильник зроблених переміщень. Зникає можливість взаємодіяти з головним вікном. Стають неактивними на змінюють колір кнопки автоматичної гри та наступного кроку. Відкривається вікно вибору способа задавання поля |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопку під час гри, коли автоматичне розвʼязання увімкнене

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості перезапуску поточної гри з увімкненим режимом автоматичної гри |
| Початковий стан програми | Працює таймер, кнопки автоматичної гри та наступного кроку активні, відбувається автоматичне розвʼязування |
| Вхідні дані | Інформація про натискання на кнопку нової гри |
| Схема проведення тесту | Натискання на кнопку нової гри |
| Очікуваний результат | Зупиняється автоматичне розвʼязання, таймер зупиняється та обнуляється, обнуляється лічильник зроблених переміщень. Зникає можливість взаємодіяти з головним вікном. Стають неактивними на змінюють колір кнопки автоматичної гри та наступного кроку. Відкривається вікно вибору способу задавання поля |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопку нової гри поза грою

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості запуску нової гри |
| Початковий стан програми | Не працює таймер, кнопки автоматичної гри та наступного кроку не активні |
| Вхідні дані | Інформація про натискання на кнопку нової гри |
| Схема проведення тесту | Натискання на кнопку нової гри |
| Очікуваний результат | Не працює таймер, кнопки автоматичної гри та наступного кроку не активні, головне вікно стає неактивним, відкривається вікно вибору способа задавання поля |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні кнопки нової гри під час користувацького задавання поля, коли поле може бути розвʼязаним

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості початку гри після користувацького задавання поля |
| Початковий стан програми | Не працює таймер, кнопки автоматичної гри та наступного кроку не активні, кнопка початку нової гри активна |
| Вхідні дані | Інформація про натискання на кнопку початку нової гри |
| Схема проведення тесту | Натискання на кнопку початку нової гри |
| Очікуваний результат | Працює таймер, кнопки автоматичної гри та наступного кроку активні |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні кнопки нової гри під час користувацького задавання поля, коли поле не може бути розвʼязаним

|  |  |
| --- | --- |
| Мета тесту | Перевірка відсутності можливості початку гри, якщо поле не може бути розвʼязаним |
| Початковий стан програми | Не працює таймер, кнопки автоматичної гри та наступного кроку не активні, кнопка початку нової гри активна |
| Вхідні дані | Інформація про натискання на кнопку нової гри |
| Схема проведення тесту | Натискання на кнопку нової гри |
| Очікуваний результат | Нічого не змінилося |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопки наступного кроку і автоматичного розвʼязання поза грою

|  |  |
| --- | --- |
| Мета тесту | Перевірка відсутності можливості використання кнопок наступного кроку та автоматичної гри поза грою |
| Початковий стан програми | Не працює таймер, кнопки автоматичної гри та наступного кроку не активні |
| Вхідні дані | інформація про натискання на кнопку наступного кроку; інформація про натискання на кнопку автоматичної гри |
| Схема проведення тесту | Натискання на кнопку наступного кроку; натискання на кнопку автоматичної гри |
| Очікуваний результат | Нічого не відбувається |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопки наступного кроку і автоматичного розвʼязання під час гри, коли автоматичне розвʼязування вимкнене

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості зробити крок автоматично та ввімкнути автоматичне розвʼязування |
| Початковий стан програми | Працює таймер, кнопки автоматичної гри та наступного кроку активні, автоматичне розвʼязування не відбувається |
| Вхідні дані | інформація про натискання на кнопку наступного кроку; інформація про натискання на кнопку автоматичної гри |
| Схема проведення тесту | Натискання на кнопку наступного кроку; натискання на кнопку автоматичної гри |
| Очікуваний результат | Виконується переміщення клітинки; Починається переміщення клітинок з певною періодичністю, яке згодом призводить поле до відсортованого стану |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопки наступного кроку і автоматичного розвʼязання під час гри, коли автоматичне розвʼязування увімкнене

|  |  |
| --- | --- |
| Мета тесту | Перевірка виконання переміщення при натисканні на кнопкою наступного кроку під час автоматичного розвʼязування; Перевірка можливості зупинити автоматичне розвʼязування |
| Початковий стан програми | Працює таймер, кнопки наступного кроку та автоматичної гри активні, виконується автоматичне розвʼязування |
| Вхідні дані | інформація про натискання на кнопку наступного кроку; інформація про натискання на кнопку автоматичної гри |
| Схема проведення тесту | Натискання на кнопку наступного кроку; натискання на кнопку автоматичної гри |
| Очікуваний результат | Виконалося додаткове переміщення, через певний час поле стає відсортованим; Автоматичне розвʼязування зупиняється |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопку випадкової генерації поля

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості згенерувати поле випадковим чином |
| Початковий стан програми | Відкрите діалогове вікно вибору способа задавання поля, таймер не працює, кнопки наступного ходу та автоматичної гри не активні |
| Вхідні дані | інформація про натискання на кнопку генерації поля випадковим чином |
| Схема проведення тесту | Натискання на кнопку генерації поля випадковим чином |
| Очікуваний результат | Діалогове вікно закрилося. Розташування клітинок у полі змінилося. Таймер почав працювати, кнопки наступного кроку та автоматичної гри стали активними |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопку генерації поля вручну

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості переходу в режим ручної генерації |
| Початковий стан програми | Відкрите діалогове вікно вибору способа задавання поля |
| Вхідні дані | інформація про натискання на кнопку генерації поля вручну |
| Схема проведення тесту | Натискання на кнопку генерації поля випадковим чином |
| Очікуваний результат | Діалогове вікно закрилося. Напис-підказка вказує як міняти клітинки місцями |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопку підтвердження збереження даних у файл

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості зберігати дані гри у файл |
| Початковий стан програми | Відкрите діалогове вікно запиту підтвердження збереження у файл |
| Вхідні дані | інформація про натискання на кнопку підтвердження |
| Схема проведення тесту | Натискання на кнопку підтвердження |
| Очікуваний результат | Діалогове вікно закрилося. У кінець файлу додалися дані попередньої гри |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на кнопку відмови збереження даних у файл

|  |  |
| --- | --- |
| Мета тесту | Перевірка роботи програми при відмові зберегти дані у файл |
| Початковий стан програми | Відкрите діалогове вікно запиту підтвердження збереження у файл |
| Вхідні дані | інформація про натискання на кнопку відмови |
| Схема проведення тесту | Натискання на кнопку відмови |
| Очікуваний результат | Діалогове вікно закрилося. Файл не змінився |
| Стан програми після проведення випробувань | Збігається з очікуваним |

Тестування роботи програми при натисканні на клітинку в режимі користувацького задавання поля, до вибору першої клітинки із пари

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості обрати першу клітинку для зміни порядку |
| Початковий стан програми | Таймер не працює, напис-підказка вказує, як міняти клітинки місцями, клітинки відрізняються лише текстом |
| Вхідні дані | інформація про натискання на клітинку |
| Схема проведення тесту | Натискання на клітинку |
| Очікуваний результат | Клітинка змінила свій вигляд |
| Стан програми після проведення випробувань | Збігається з очікуваним |

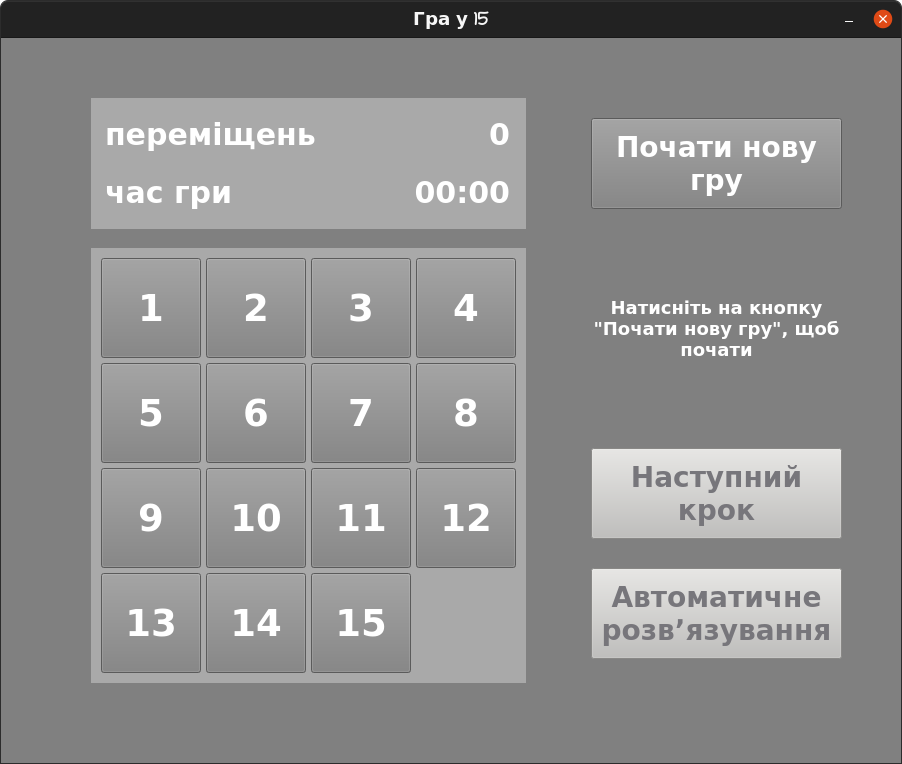
Тестування роботи програми при натисканні на клітинку в режимі користувацького задавання поля, після вибору першої клітинки із пари

|  |  |
| --- | --- |
| Мета тесту | Перевірка можливості змінити дві клітинки місцями |
| Початковий стан програми | Таймер не працює, напис-підказка вказує, як міняти клітинки місцями, одна клітинка відрізняється від решти |
| Вхідні дані | інформація про натискання на клітинку |
| Схема проведення тесту | Натискання на клітинку |
| Очікуваний результат | Клітинки помінялися місцями |
| Стан програми після проведення випробувань | Збігається з очікуваним |

# Інструкція користувача

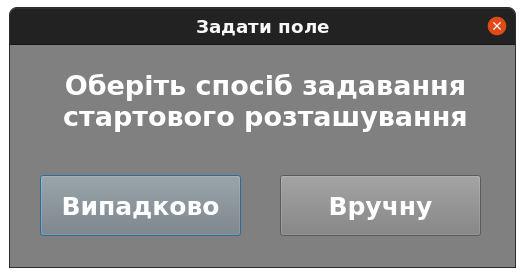
## Робота з програмою

Після запуску виконуваного файлу з розширенням \*.exe відкривається головне вікно програми (Рисунок 6.1).

* + - * 1. Головне вікно програми

Ви можете переміщувати сусідні із пробілом клітинки, проте гра ще не почалася. Кнопки покрокового розвʼязання та автоматичної гри не доступні. Натисніть на кнопку «Почати нову гру».

Відкриється вікно способу задавання стартового поля (Рисунок 6.2).

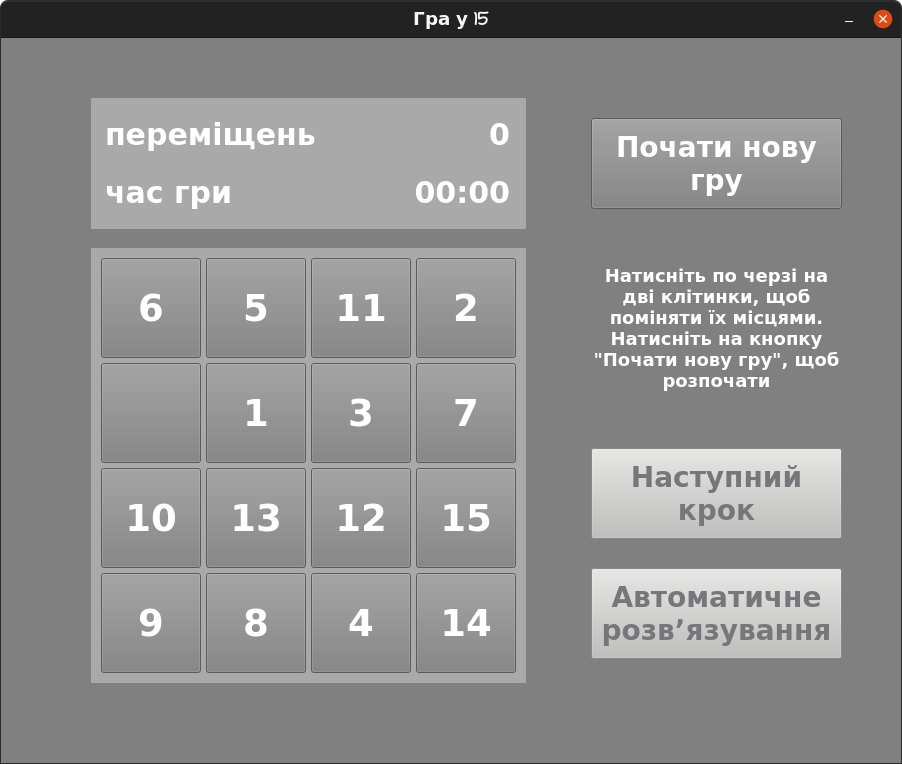


* + - * 1. Вікно вибору способу задаванняння стартового поля

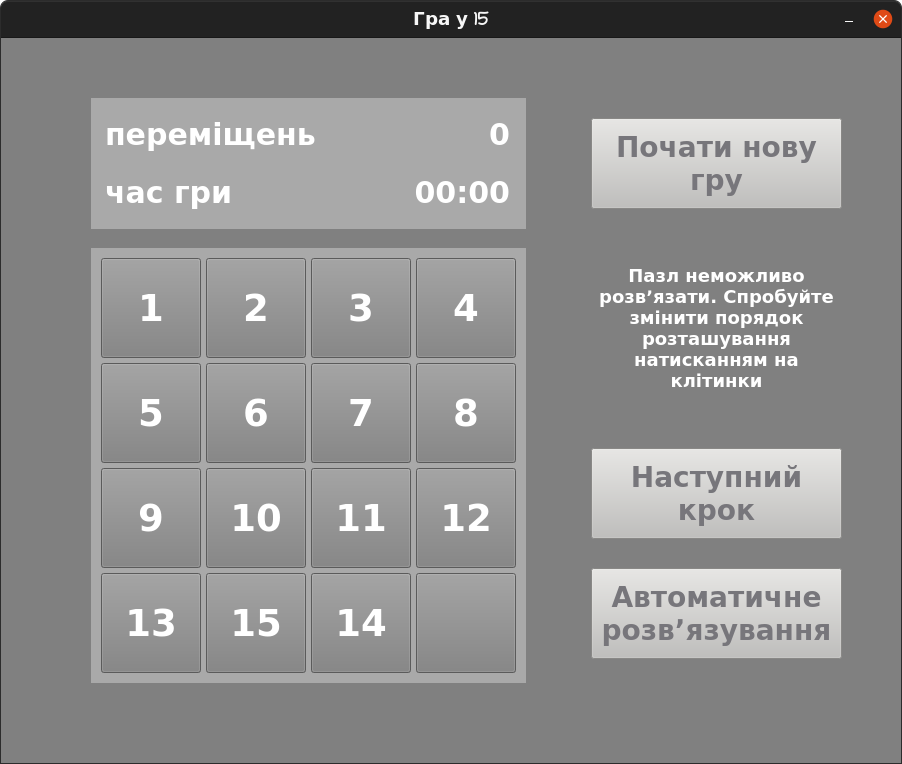
У випадку натискання першої кнопки поле згенерується випадковим чином і воно матиме розвʼязок. Одразу після цього почнеться гра та запуститься таймер (Рисунок 6.3).

* + - * 1. Головне вікно під час гри

У разі вибору задавання поля вручну ви отримеєте можливість попарно міняти місцями розташування клітинок, натискаючи на них по черзі для задавання бажаного стартового розташування поля (Рисунок 6.4).

* + - * 1. Головне вікно під час користувацького задавання поля

Якщо в процесі зміни розташування клітинок поле втратить можливість бути розвʼязаним або буде у повністю відстортованому стані, кнопка початку гри перестане бути активною (рисунок 6.5).

* + - * 1. Головне вікно під час користувацького задавання поля у випадку відсутності розвʼязку

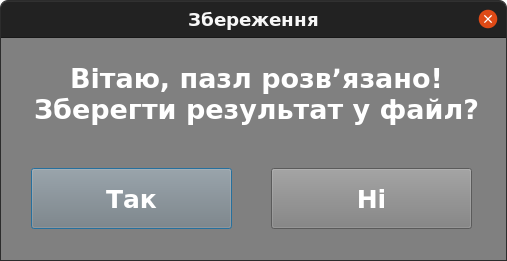
Якщо під час гри у вас виникли труднощі, ви можете натиснути кнопку наступного кроку. При її натисненні гра, використовуючи вбудований алгоритм зробить одне переміщення.

Якщо ви тільки знайомитеся з грою, ви можете використати функцію автоматичного розвʼязування. Після натискання відповідної кнопки пазл почне розвʼязуватися самостійно (Рисунок 6.6).

* + - * 1. Автоматичне розвʼязання

У будь-який момент ви можете зупинити автоматичне розвʼязування, натиснувши на кнопку «Зупинити».

Після того, як пазл буде зібрано, зʼявиться діалогове вікно, яке дасть можливість зберегти дані про витрачений час, кількість кроків та початкове розташування поля у файл (Рисунок 6.7).

* + - * 1. Діалогове вікно збереження у файл

Якщо ви бажаєте почати нову гру, не завершуючи попередню, ви можете натиснути кнопку «Почати нову гру». Одразу після натискання поточна гра зупиниться.

## Формат вхідних та вихідних даних

Результатом виконання програми є доданий у файл текст, що складається з даних про час завершення гри, кількість зроблених переміщень, витрачений час та початковий стан поля.

## Системні вимоги

## 

Системні вимоги до програмного забезпечення наведені в таблиці 6.9.

Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows 7/ Windows 8/Windows 10 (з останніми обновленнями) | Windows 10  (з останніми обновленнями) |
| Процесор | Intel® Core® i5-4690K  3.50 GHz або AMD FX-8300 3.3 GHz | Intel® Core® i5-10400F  2.90 GHz або AMD Ryzen 5 1600 3.2 GHz |
| Оперативна пам'ять | 1 GB | 2 GB |
| Відеоадаптер | NVIDIA GeForce 940MX  (або сумісний аналог) | |
| Дисплей | 1600х1024 | 1920х1080 або краще |
| Прилади введення | комп’ютерна миша |  |
| Додаткове програмне забезпечення |  | |

Висновки

Мною було реалізовано програмне забезпечення, що моделює гру «Гра у 15». Гра створена

У розділі 2 описано правила гри та алгоритм, що використовується для розвʼязання задачі.

У пункті 3 викладено покрокове виконання основних алгоритмів, що використовуються для пошуку розвʼязку головоломки.

У пункті 4 представлено діаграму класів програмного забезпечення, опис користувацьких та стандартних методів.

У пункті 5 представлено детальний план тестування, який охоплює усі можливі варіанти розвитку подій. Програмне забезпечення вдало пройшло усі тестові випадки.

У пункті 6 представлена інструкція користувача, яка пояснює усі способи взаємодії із ПЗ.

Перелік посилань

1. 15 puzzle *en.wikipedia.org*

URL: <https://en.wikipedia.org/wiki/15_puzzle>(дата звернення 16.04.2022)

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко. М. М.

«12» квітня 2022 р.

Виконавець:

Студент Мельник Данило Євгенійович

«13» червня 2022р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Гра у 15»

з дисципліни:

«Основи програмування»

Київ 2022

* 1. *Мета*: Метою курсової роботи є розробка додатку, який генерує головоломку «Гра у 15»
  2. *Дата початку роботи*: «12» квітня2022 р.
  3. *Дата закінчення роботи*: «12» червня 2022 р.
  4. *Вимоги до програмного забезпечення*.

Функціональні вимоги:

* Можливість задавання початкового розташування клітинок вручну;
* Можливість генерації дошки випадковим чином;
* Можливість перевірки, чи має початкове розташування клітинок розвʼязок;
* Можливість переміщувати на порожнє поле сусідню із ним клітинку;
* Можливість перевірки, чи відповідає поточне розташування клітинок на дошці виграшному;
* Можливість фіксувати кількість зроблених за гру переміщень;
* Можливість фіксувати витрачений на гру час;
* Можливість зберігати початкове розташування клітинок, кількість переміщень та витрачений час у файл;
* Можливість переходу у режим автоматичної гри, що використовує самостійно створений алгоритм;
* Нефункціональні вимоги:
* Можливість запускати програмне забезпечення на операційній системі Ubuntu 21.10 і вище
* 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.

*5. Стадії та етапи розробки*:

* Об'єктно-орієнтований аналіз предметної області задачі (до 05.05.2022 р.)
* Об'єктно-орієнтоване проектування архітектури програмної системи (до 12.05.2022р.)
* Розробка програмного забезпечення (до 25.05.2022р.)
* Тестування розробленої програми (до 01.06.202\_р.)
* Розробка пояснювальної записки (до 12.06.2022 р.).
* Захист курсової роботи (до 19.06.2022 р.).

*6. 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

(Вид носія даних)

*Електронний носій*

*студента групи ІП-15 І курсу*

*Мельника Д.Є.*

*121212312*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення вирішення та генерації головоломки «Гра у 15»*

(Обсяг програми (документа), арк., Кб)

*31 арк, 23 Kb*

**Файл MainController.py**

from Field import Field

from Solver import Solver

from Constants import \*

from PyQt5.QtCore import QTimer

from PyQt5.QtWidgets import QApplication

from copy import deepcopy

from time import time, localtime, strftime

from sys import argv

from MainView import MainView

from DialogView import StartDialogView, EndDialogView

class MainController:

def \_\_init\_\_(self):

self.\_\_field = Field()

self.\_\_solver = None

self.\_\_initial\_field = None

self.\_\_reorder\_cell\_ind = None

self.\_\_solution\_is\_valid = False

self.\_\_solution = []

self.\_\_solution\_timer = QTimer()

self.\_\_solution\_timer.timeout.connect(self.\_\_make\_solution\_step)

self.\_\_moves\_count = 0

self.\_\_sec\_count = 0

self.\_\_timer = QTimer()

self.\_\_timer.timeout.connect(self.\_\_time\_update)

app = QApplication(argv)

self.\_\_view = MainView(self)

self.\_\_start\_dialog = StartDialogView(self)

self.\_\_end\_dialog = EndDialogView(self)

self.\_\_view.show()

app.exec()

def try\_move\_cell(self, index):

if self.\_\_field.next\_to\_space(index):

change = index - self.\_\_field.space\_ind

self.\_\_make\_space\_swap(change)

if self.\_\_timer.isActive():

if self.\_\_solution\_is\_valid and change != self.\_\_solution.pop(0):

self.\_\_solution\_is\_valid = False

if self.\_\_field.is\_sorted():

self.\_\_end\_game()

self.\_\_win\_game()

def random\_reorder(self):

self.\_\_view.switch\_flat(self.\_\_field.space\_ind)

self.\_\_field.shuffle\_arr()

if not self.\_\_field.invar():

new\_x = (self.\_\_field.space\_x + 2) % FIELD\_SIDE

new\_y = self.\_\_field.space\_y

swap\_ind = new\_x + FIELD\_SIDE \* new\_y

self.\_\_field.two\_elements\_swap(swap\_ind, self.\_\_field.space\_ind)

self.\_\_view.update\_field(self.\_\_field.arr)

self.\_\_view.switch\_flat(self.\_\_field.space\_ind)

self.\_\_start\_game()

def user\_reorder(self):

self.\_\_view.switch\_to\_reorder(self.\_\_field.space\_ind)

self.\_\_update\_rebase\_view()

def chose\_reorder\_cell(self, cell\_ind):

if self.\_\_reorder\_cell\_ind is not None:

self.\_\_field.two\_elements\_swap(cell\_ind, self.\_\_reorder\_cell\_ind)

self.\_\_view.swap\_text(cell\_ind, self.\_\_reorder\_cell\_ind)

self.\_\_view.switch\_flat(self.\_\_reorder\_cell\_ind)

self.\_\_reorder\_cell\_ind = None

self.\_\_update\_rebase\_view()

else:

self.\_\_view.switch\_flat(cell\_ind)

self.\_\_reorder\_cell\_ind = cell\_ind

def \_\_update\_rebase\_view(self):

if self.\_\_field.is\_sorted():

self.\_\_view.block\_start()

self.\_\_view.set\_hint(SORTED\_HINT\_TEXT)

elif self.\_\_field.invar():

self.\_\_view.enable\_start()

self.\_\_view.set\_hint(REBASE\_HINT\_TEXT)

else:

self.\_\_view.block\_start()

self.\_\_view.set\_hint(UNSOLVABLE\_HINT\_TEXT)

def end\_reorder(self):

self.\_\_view.switch\_flat(self.\_\_field.space\_ind)

self.\_\_view.finish\_reorder()

self.\_\_start\_game()

def \_\_start\_game(self):

self.\_\_view.connect\_solver()

self.\_\_view.set\_hint(INGAME\_HINT)

self.\_\_initial\_field = self.\_\_field.matrix\_view()

self.\_\_timer.start(SEC\_TO\_MS)

def \_\_end\_game(self):

self.\_\_solution\_is\_valid = False

self.\_\_turn\_off\_solver()

self.\_\_view.set\_hint(START\_GAME\_HINT)

self.\_\_timer.stop()

self.\_\_view.lock\_solver()

def \_\_win\_game(self):

self.\_\_view.set\_frog()

self.\_\_end\_dialog.exec()

def chose\_start(self):

if self.\_\_timer.isActive():

self.\_\_end\_game()

else:

self.\_\_view.remove\_frog(self.\_\_field.space\_ind)

self.\_\_moves\_count = -1

self.\_\_moves\_update()

self.\_\_sec\_count = -1

self.\_\_time\_update()

self.\_\_start\_dialog.exec()

def save\_to\_file(self):

with open("results.txt", "a") as f:

f.write(f"Час закінчення гри: {strftime('%d.%m.%Y %H:%M:%S', localtime(time()))}\n")

f.write(f"Кількість переміщень: {self.\_\_moves\_count}\n")

f.write(f"Витрачений час: {self.\_\_sec\_count} сек.\n")

f.write(f"Початковий стан поля: \n{self.\_\_initial\_field}\n")

def \_\_time\_update(self):

self.\_\_sec\_count += 1

self.\_\_view.timer\_update(self.\_\_sec\_count)

def \_\_moves\_update(self):

self.\_\_moves\_count += 1

self.\_\_view.moves\_count\_update(self.\_\_moves\_count)

def \_\_turn\_off\_solver(self):

self.\_\_solution\_timer.stop()

self.\_\_view.solver\_end()

if self.\_\_timer.isActive():

self.\_\_view.set\_hint(INGAME\_HINT)

else:

self.\_\_view.set\_hint(START\_GAME\_HINT)

def \_\_turn\_on\_solver(self):

self.\_\_validate\_solution()

self.\_\_solution\_timer.start(SOLVE\_INTERVAL)

self.\_\_view.solver\_start()

self.\_\_view.set\_hint(SOLVER\_HINT)

def switch\_solver(self):

if self.\_\_solution\_timer.isActive():

self.\_\_turn\_off\_solver()

else:

self.\_\_turn\_on\_solver()

def \_\_generate\_solution(self):

self.\_\_solver = Solver(deepcopy(self.\_\_field))

self.\_\_solution = self.\_\_solver.solve()

def \_\_validate\_solution(self):

if not self.\_\_solution\_is\_valid:

self.\_\_generate\_solution()

self.\_\_solution\_is\_valid = True

def step\_pushed(self):

self.\_\_validate\_solution()

self.\_\_make\_solution\_step()

def \_\_make\_solution\_step(self):

self.\_\_make\_space\_swap(self.\_\_solution.pop(0))

if not self.\_\_solution:

self.\_\_end\_game()

self.\_\_win\_game()

def \_\_make\_space\_swap(self, change):

self.\_\_field.space\_swap(change)

space = self.\_\_field.space\_ind

self.\_\_view.swap\_text(space, space - change)

self.\_\_view.switch\_flat(space)

self.\_\_view.switch\_flat(space - change)

self.\_\_moves\_update()

**Файл MainView.py**

from UiMainWindow import UiMainWindow

from PyQt5.QtWidgets import QMainWindow

from functools import partial

from Constants import \*

class MainView(QMainWindow):

def \_\_init\_\_(self, controller):

super(MainView, self).\_\_init\_\_()

self.\_\_controller = controller

self.\_\_ui = UiMainWindow()

self.\_\_ui.setupUi(self)

self.\_\_cells = [self.\_\_ui.pushButton,

self.\_\_ui.pushButton\_2,

self.\_\_ui.pushButton\_3,

self.\_\_ui.pushButton\_4,

self.\_\_ui.pushButton\_5,

self.\_\_ui.pushButton\_6,

self.\_\_ui.pushButton\_7,

self.\_\_ui.pushButton\_8,

self.\_\_ui.pushButton\_9,

self.\_\_ui.pushButton\_10,

self.\_\_ui.pushButton\_11,

self.\_\_ui.pushButton\_12,

self.\_\_ui.pushButton\_13,

self.\_\_ui.pushButton\_14,

self.\_\_ui.pushButton\_15,

self.\_\_ui.pushButton\_16,

]

self.\_\_start\_button = self.\_\_ui.pushButton\_17

self.\_\_step\_button = self.\_\_ui.pushButton\_18

self.\_\_solver\_button = self.\_\_ui.pushButton\_19

self.\_\_hint\_label = self.\_\_ui.label\_5

self.\_\_moves\_label = self.\_\_ui.label\_2

self.\_\_time\_label = self.\_\_ui.label\_4

self.set\_hint(START\_GAME\_HINT)

self.lock\_solver()

self.\_\_connect\_button(self.\_\_ui.pushButton\_17, self.\_\_controller.chose\_start)

self.\_\_connect\_cells(self.\_\_controller.try\_move\_cell)

def \_\_connect\_cells(self, slot):

for cell in self.\_\_cells:

cell.pressed.connect(partial(slot, self.\_\_cells.index(cell)))

def \_\_disconnect\_cells(self):

for cell in self.\_\_cells:

cell.disconnect()

def update\_field(self, values):

for button, value in zip(self.\_\_cells, values):

button.setText(str(value).replace(str(FIELD\_SIZE), ""))

def switch\_to\_reorder(self, space\_ind):

self.\_\_disconnect\_cells()

self.\_\_connect\_cells(self.\_\_controller.chose\_reorder\_cell)

self.\_\_cells[space\_ind].setFlat(False)

def switch\_to\_move(self):

self.\_\_disconnect\_cells()

self.\_\_connect\_cells(self.\_\_controller.try\_move\_cell)

def swap\_text(self, ind1, ind2):

cell1 = self.\_\_cells[ind1]

cell2 = self.\_\_cells[ind2]

saved = cell1.text()

cell1.setText(cell2.text())

cell2.setText(saved)

def enable\_start(self):

self.\_\_connect\_button(self.\_\_start\_button, self.\_\_controller.end\_reorder)

def block\_start(self):

self.\_\_disconnect\_button(self.\_\_start\_button)

def finish\_reorder(self):

self.\_\_connect\_button(self.\_\_start\_button, self.\_\_controller.chose\_start)

self.switch\_to\_move()

def solver\_start(self):

self.\_\_solver\_button.setText(OFF\_SOLVER\_BUTTON)

def solver\_end(self):

self.\_\_solver\_button.setText(ON\_SOLVER\_BUTTON)

def moves\_count\_update(self, moves\_count):

self.\_\_moves\_label.setText(str(moves\_count))

def timer\_update(self, sec\_count):

self.\_\_time\_label.setText(f"{sec\_count // MIN\_TO\_SEC:02}:{sec\_count % MIN\_TO\_SEC:02}")

def connect\_buttons(self):

self.\_\_connect\_button(self.\_\_start\_button, self.\_\_controller.chose\_start)

self.\_\_connect\_button(self.\_\_step\_button, self.\_\_controller.make\_solution\_step)

self.\_\_connect\_button(self.\_\_solver\_button, self.\_\_controller.switch\_solver)

def \_\_connect\_button(self, button, slot):

self.\_\_disconnect\_button(button)

button.clicked.connect(slot)

button.setStyleSheet("")

@staticmethod

def \_\_disconnect\_button(button):

try:

button.disconnect() # повертає помилку, якщо жодний сигнал не підключено

button.setStyleSheet(BLOCKED\_STYLESHEET)

except TypeError:

pass

def switch\_flat(self, ind):

cell = self.\_\_cells[ind]

cell.setFlat(not cell.isFlat())

def set\_frog(self):

self.\_\_cells[-1].setText(FROG)

def remove\_frog(self, ind):

self.\_\_cells[ind].setText("")

def connect\_solver(self):

self.\_\_connect\_button(self.\_\_step\_button, self.\_\_controller.step\_pushed)

self.\_\_connect\_button(self.\_\_solver\_button, self.\_\_controller.switch\_solver)

def lock\_solver(self):

self.\_\_disconnect\_button(self.\_\_step\_button)

self.\_\_disconnect\_button(self.\_\_solver\_button)

def set\_hint(self, text):

self.\_\_hint\_label.setText(text)

**Файл Solver.py**

from Graph import LockableGraph

from Constants import \*

class Solver:

def \_\_init\_\_(self, field):

self.\_\_field = field

self.\_\_graph = LockableGraph(FIELD\_SIZE)

self.\_\_seq = []

def solve(self):

for i in range(FIELD\_SIDE - 3):

self.\_\_fill\_line(i, vertical=False)

self.\_\_fill\_line(i, vertical=True)

self.\_\_fill\_line(FIELD\_SIDE - 3, vertical=False)

self.\_\_fill\_two\_of\_five()

self.\_\_fill\_last\_three()

return self.\_\_seq

def \_\_in\_place(self, \*arr\_ind):

for ind in arr\_ind:

if self.\_\_field.arr[ind] != ind + 1:

return False

return True

def \_\_add\_moves(self, \*moves):

for move in moves:

self.\_\_seq.append(move)

self.\_\_field.space\_swap(move)

def \_\_move\_space\_to(self, dest):

prev = self.\_\_field.space\_ind

path = self.\_\_graph.shortest\_path\_search(prev, dest) # n^2

for i in path:

self.\_\_add\_moves(i - prev)

prev = i

def \_\_move\_value\_to(self, val, dest):

prev = self.\_\_field.value\_ind(val)

path = self.\_\_graph.shortest\_path\_search(prev, dest)

for i in path: # n^3

self.\_\_graph.close\_vert(prev)

self.\_\_move\_space\_to(i) # n^2

self.\_\_graph.open\_vert(prev)

self.\_\_add\_moves(prev - i)

prev = i

def \_\_fill\_line(self, num, vertical):

if vertical:

along, across = DOWN, RIGHT # fill column

beg, end = (num + 1) \* FIELD\_SIDE + num, FIELD\_SIDE\*(FIELD\_SIDE-1) + num

else:

along, across = RIGHT, DOWN # fill row

beg, end = num \* (FIELD\_SIDE + 1), FIELD\_SIDE\*(num + 1)-1

if self.\_\_in\_place(\*(range(beg, end + 1, along))):

self.\_\_graph.close\_vert(\*(range(beg, end + 1, along)))

return

for i in range(beg, end, along):

self.\_\_move\_value\_to(i + 1, i)

self.\_\_graph.close\_vert(i)

self.\_\_move\_space\_to(end + across)

if not self.\_\_in\_place(end):

self.\_\_move\_value\_to(end + 1, end + across)

self.\_\_graph.close\_vert(end + across)

self.\_\_move\_space\_to(end + 2 \* -along + across)

path = [-across, along, along, across, -along, -across, -along, across]

self.\_\_add\_moves(\*path)

self.\_\_graph.open\_vert(end + across)

self.\_\_graph.close\_vert(end)

def \_\_fill\_two\_of\_five(self):

start = FIELD\_SIZE + UP + 3 \* LEFT

if not self.\_\_in\_place(start, start + DOWN):

self.\_\_move\_value\_to(start + DOWN + 1, start)

self.\_\_graph.close\_vert(start)

self.\_\_move\_space\_to(start + DOWN + RIGHT)

if self.\_\_field.arr[start + DOWN] == start + 1:

path = [LEFT, UP, RIGHT, DOWN, RIGHT, UP, LEFT, LEFT, DOWN, RIGHT]

self.\_\_add\_moves(\*path)

self.\_\_move\_value\_to(start + 1, start + RIGHT)

self.\_\_graph.close\_vert(start + RIGHT)

self.\_\_graph.open\_vert(start)

self.\_\_move\_value\_to(start + DOWN + 1, start + DOWN)

self.\_\_graph.close\_vert(start + DOWN)

self.\_\_add\_moves(RIGHT)

self.\_\_graph.open\_vert(start + RIGHT)

self.\_\_graph.close\_vert(start)

else:

self.\_\_graph.close\_vert(start)

self.\_\_graph.close\_vert(start + DOWN)

def \_\_fill\_last\_three(self):

for ind in FIELD\_SIZE + UP + LEFT - 1, FIELD\_SIZE + UP - 1, FIELD\_SIZE + LEFT - 1:

self.\_\_move\_value\_to(ind + 1, ind)

self.\_\_graph.close\_vert(ind)

**Файл DialogView.py**

from UiStartDialog import UiStartDialog

from UiEndDialog import UiEndDialog

from PyQt5.QtWidgets import QDialog

class StartDialogView(QDialog):

def \_\_init\_\_(self, controller):

super(StartDialogView, self).\_\_init\_\_()

self.\_\_controller = controller

self.\_\_ui = UiStartDialog()

self.\_\_ui.setupUi(self)

self.\_\_ui.pushButton.clicked.connect(self.close)

self.\_\_ui.pushButton\_2.clicked.connect(self.close)

self.\_\_ui.pushButton.clicked.connect(self.\_\_controller.random\_reorder)

self.\_\_ui.pushButton\_2.clicked.connect(self.\_\_controller.user\_reorder)

class EndDialogView(QDialog):

def \_\_init\_\_(self, controller):

super(EndDialogView, self).\_\_init\_\_()

self.\_\_controller = controller

self.\_\_ui = UiEndDialog()

self.\_\_ui.setupUi(self)

self.\_\_ui.pushButton.clicked.connect(self.close)

self.\_\_ui.pushButton\_2.clicked.connect(self.close)

self.\_\_ui.pushButton.clicked.connect(self.\_\_controller.save\_to\_file)

**Файл Field.py**

from InversionCounter import InversionCounter as InvCount

from random import shuffle

from Constants import \*

class Field:

def \_\_init\_\_(self):

self.arr = []

for i in range(1, FIELD\_SIZE + 1):

self.arr.append(i)

self.space\_ind = FIELD\_SIZE - 1

self.space\_x = FIELD\_SIDE - 1

self.space\_y = FIELD\_SIDE - 1

def find\_space(self):

self.space\_ind = self.value\_ind(FIELD\_SIZE)

self.\_\_update\_space\_coords()

def \_\_update\_space\_coords(self):

self.space\_x, self.space\_y = \

self.space\_ind % FIELD\_SIDE, self.space\_ind // FIELD\_SIDE

def value\_ind(self, val):

return self.arr.index(val)

def invar(self):

check\_arr = self.arr.copy()

check\_arr.remove(FIELD\_SIZE)

inv = InvCount(check\_arr)

inversions\_par = inv.num\_inver() % 2

space\_y\_par = (FIELD\_SIDE - self.space\_y - 1) % 2

return space\_y\_par == inversions\_par

def space\_swap(self, change):

if change in DIRECTIONS:

self.two\_elements\_swap(self.space\_ind, self.space\_ind + change)

self.\_\_update\_space\_coords()

def two\_elements\_swap(self, ind1, ind2):

self.arr[ind1], self.arr[ind2] = self.arr[ind2], self.arr[ind1]

self.find\_space()

def next\_to\_space(self, ind):

return self.space\_ind - ind in DIRECTIONS

def shuffle\_arr(self):

prev = self.arr.copy()

while self.is\_sorted() or self.arr == prev:

shuffle(self.arr)

self.find\_space()

def is\_sorted(self):

return self.arr == list(range(1, FIELD\_SIZE + 1))

def matrix\_view(self):

res = ""

for i in range(FIELD\_SIDE):

for j in range(FIELD\_SIDE):

res += f"{str(self.arr[i \* FIELD\_SIDE + j]).replace(str(FIELD\_SIZE), ' '):>2} "

res += "\n"

return res

**Файл Graph.py**

from MyQueue import MyQueue

from copy import deepcopy

from Constants import \*

class Graph:

def \_\_init\_\_(self, adj\_list):

self.\_adj\_list = adj\_list

@staticmethod

def puzzle\_adj\_list(): # n = n0 ^ 2

\_adj\_list = {i: [] for i in range(FIELD\_SIZE)}

for i in range(FIELD\_SIZE - FIELD\_SIDE):

\_adj\_list[i].append(i + FIELD\_SIDE)

\_adj\_list[i + FIELD\_SIDE].append(i)

ind = 0

for i in range(FIELD\_SIDE):

for j in range(FIELD\_SIDE - 1):

\_adj\_list[ind].append(ind + 1)

\_adj\_list[ind + 1].append(ind)

ind += 1

ind += 1

return \_adj\_list

def shortest\_path\_search(self, beg, end):

queue = MyQueue()

p = [-1] \* FIELD\_SIZE

p[beg] = None

v = beg

while v != end: # n

for u in self.\_adj\_list[v]: # n0^2

if p[u] == -1:

p[u] = v

queue.enqueue(u)

v = queue.dequeue()

res = []

while v != beg:

res.append(v)

v = p[v]

return list(reversed(res))

class LockableGraph(Graph):

def \_\_init\_\_(self, size):

self.\_\_opened\_vert = set(range(size))

self.\_\_full = self.puzzle\_adj\_list()

Graph.\_\_init\_\_(self, deepcopy(self.\_\_full))

def close\_vert(self, \*v):

for i in v:

if i not in self.\_\_opened\_vert:

continue

self.\_\_opened\_vert.remove(i)

for j in self.\_adj\_list[i]: # c

self.\_adj\_list[j].remove(i) # n

self.\_adj\_list[i] = []

def open\_vert(self, \*v):

for i in v:

if i in self.\_\_opened\_vert:

continue

self.\_\_opened\_vert.add(i)

for j in self.\_\_full[i]: # c

if j in self.\_\_opened\_vert:

self.\_adj\_list[j].append(i) # c

self.\_adj\_list[i].append(j) # c

**Файл MyQueue.py**

class MyQueue:

def \_\_init\_\_(self):

self.\_\_queue = []

def enqueue(self, el):

self.\_\_queue.append(el) # 1

def dequeue(self):

return self.\_\_queue.pop(0) # n

**Файл InversionCounter.py**

class InversionCounter:

def \_\_init\_\_(self, arr):

self.\_\_arr = arr

def num\_inver(self):

return self.\_\_inv\_merge\_sort(0, len(self.\_\_arr) - 1)

def \_\_inv\_merge\_sort(self, l, r):

if l == r:

return 0

m = r + l >> 1

sw1 = self.\_\_inv\_merge\_sort(l, m)

sw2 = self.\_\_inv\_merge\_sort(m + 1, r)

swap\_count = self.\_\_merge(l, m, r)

return sw1 + sw2 + swap\_count

def \_\_merge(self, l1, r1, r2):

i1 = l1

i2 = r1 + 1

new\_arr = []

swap\_count = 0

while i1 <= r1 and i2 <= r2:

if self.\_\_arr[i1] <= self.\_\_arr[i2]:

new\_arr.append(self.\_\_arr[i1])

i1 += 1

else:

new\_arr.append(self.\_\_arr[i2])

i2 += 1

swap\_count += r1 - i1 + 1

self.\_\_arr[l1:r2 + 1] = new\_arr + self.\_\_arr[i1:r1 + 1] + self.\_\_arr[i2:r2 + 1]

return swap\_count

**Файл Constants.py**

FIELD\_SIDE = 4

FIELD\_SIZE = FIELD\_SIDE \* FIELD\_SIDE

SOLVE\_INTERVAL = 300

MIN\_TO\_SEC = 60

SEC\_TO\_MS = 1000

RIGHT = +1

LEFT = -1

UP = -FIELD\_SIDE

DOWN = +FIELD\_SIDE

DIRECTIONS = [RIGHT, LEFT, UP, DOWN]

FROG = "🐸"

ON\_SOLVER\_BUTTON = "Автоматичне\nрозвʼязування"

OFF\_SOLVER\_BUTTON = "Зупинити"

BLOCKED\_STYLESHEET = "color: rgb(119, 118, 123); background-color: rgb(192, 191, 188);"

START\_GAME\_HINT = "Натисніть на кнопку \"Почати нову гру\", щоб почати"

REBASE\_HINT\_TEXT = "Натисніть по черзі на дві клітинки, щоб поміняти їх місцями. " \

"Натисніть на кнопку \"Почати нову гру\", щоб розпочати"

UNSOLVABLE\_HINT\_TEXT = "Пазл неможливо розвʼязати. Спробуйте " \

"змінити порядок розташування натисканням " \

"на клітинки"

SORTED\_HINT\_TEXT = "Здається, що пазл вже складено, Змініть розташування " \

"натисканням на клітинки, які хочете поміняти місцями"

INGAME\_HINT = "Переміщуйте клітинки, допоки поле не буде у відсортованому стані " \

"з пробілом у правому нижньому кутку"

SOLVER\_HINT = "Відбувається автоматичне розвʼязування, натисніть на кнопку " \

"\"Зупинити\", щоб вимкнути його"

**Файл main.py**

from MainController import MainController

def main():

MainController()

if \_\_name\_\_ == '\_\_main\_\_':

main()