

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 дисципліни «Основи програмування»
(назва дисципліни)
на тему: «Гра у 15»

Студента 1 курсу, групи ПІ-15
Мельника Данила Євгенійовича

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 інформатики та програмної інженерії

Дисципліна Основи програмування

Напрямок "Програмна інженерія"

Курс 1 Група ІІІ-15

Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Мельника Данила Євгенійовича

(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
М. _____
(підпис)

ст.в.Головченко М.

(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 «Гра у 15» та алгоритмів її розв'язання, розроблений графічний інтерфейс для взаємодією користувача з ПЗ. Вивчено метод розробки програмного забезпечення з використанням парадигми програмування ООП.

ЗМІСТ

ВСТУП.....	6
1 ПОСТАНОВКА ЗАДАЧІ	7
2 ТЕОРЕТИЧНІ ВІДОМОСТІ	8
2.1 Правила гри «Гра у 15»	8
3 ОПИС АЛГОРИТМІВ	14
3.1 Загальний алгоритм	14
3.2 Алгоритм випадкової генерації поля	15
3.3 Алгоритм зміни розташування поля користувачем	15
3.4 Алгоритм створення масиву послідовності дій для розв’язання пазла	16
3.5 Алгоритм заповнення рядка або стовпця відповідними значеннями	16
3.6 Алгоритм підстановки клітинок 10 і 14	17
3.7 Алгоритм заповнення останніх трьох клітинок.....	17
3.8 Алгоритм переміщення клітинки у певну позицію.....	18
3.9 Алгоритм переміщення пробіла у певну позицію	18
3.10 Алгоритм знаходження найкоротшого шляху між вершинами у графі за допомогою пошуку вшир	18
4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	20
4.1 Діаграма класів програмного забезпечення	20
4.2 Опис методів частин програмного забезпечення	21
4.2.1 Стандартні методи.....	21
4.2.2 Користувацькі методи	24
5 ПЛАН ТЕСТУВАННЯ	44
5.1 План тестування.....	44
5.2 Приклади тестування	45
6 ІНСТРУКЦІЯ КОРИСТУВАЧА.....	56
6.1 Робота з програмою.....	56

6.2 Формат вхідних та вихідних даних.....	59
6.3 Системні вимоги	59
ВИСНОВКИ	61
ПЕРЕЛІК ПОСИЛАНЬ	62
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ.....	63
ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ	66

ВСТУП

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

З точки зору програмування та створення алгоритмів вона об'єднує найцікавіші характеристики: математичну базу, зовнішню простоту та невизначену складність при пошуку оптимального розв'язання.

У своїй роботі я намагався приділити увагу не глибокому 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 розв'язує людина, поступово заповнюючи окремі ділянки поля потрібними значеннями.

1 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 розв'язаним.

Програма має перевіряти дані на можливість розв'язку та за потреби корегувати їх, якщо матриця була згенерована, або запитувати нові дані, якщо введені користувачем були некоректними.

Під час гри має бути можливість увімкнення та вимкнення режиму автоматичної гри, під час якої програма самостійно розв'язує 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.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Правила гри «Гра у 15»

«Гра у 15» (або «П'ятнашки») 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 звільняючи свою. Метою гри є заповнення поля клітинками у відсортованому порядку (рисунок 2.1)[1].

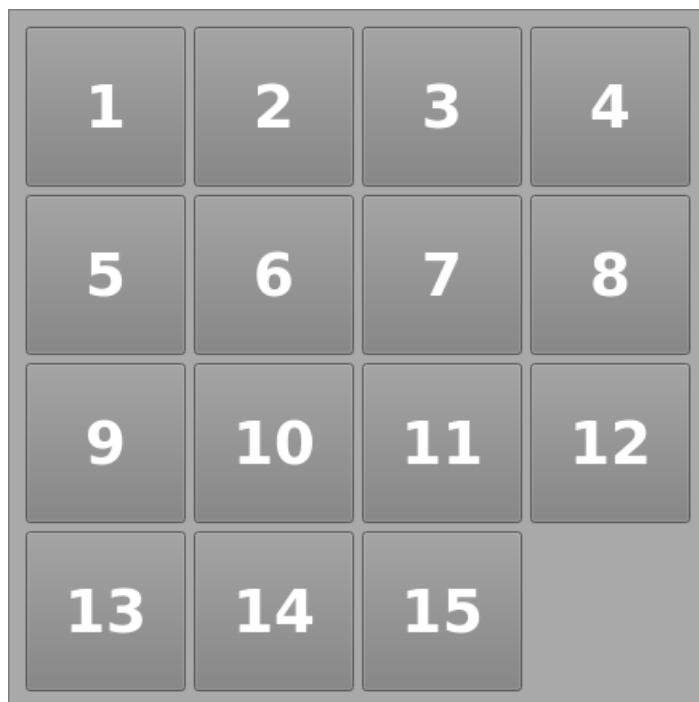


Рисунок 2.1 – Поле для гри у відсортованому стані

2.2 Розв'язуваність 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 розв'язаним[1].

Доведення:

Якщо пробіл переміщується горизонтально, сама послідовність заповнених 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 розв’язаними.

2.3 Алгоритм розв’язання пазла

Поле може бути розв’язаним почерговим заповненням його рядків та стовпців, починаючи з верхнього рядка (рисунок 2.2).

1	2	3	4	1	2	3	4	1	2	3	4
6		15	12	5		14	15	5	6	7	8
7	13	9	14	9	6	8	12	9		14	11
10	8	11	5	13	7	10	11	13	10	12	15
				1	2	3	4	1	2	3	4
				5	6	7	8	5	6	7	8
				9	10		11	9	10	11	12
				13	14	15	12	13	14	15	

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 поставити на своє фінальне місце, необхідно перемістити її у позицію поруч із нею (рисунок 2.3).

1	2	3	14
12		5	4
10	11	13	8
7	9	15	6

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

Алгоритм встановлення останнього елемента представлено на рисунку 2.4.

1	2	3	14	1	3	14		1	3		4
12		5	4	12	2	5	4	12	2	14	5
10	11	13	8	10	11	13	8	10	11	13	8
7	9	15	6	7	9	15	6	7	9	15	6

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 15. Може виникнути ситуація, коли це буде проблемно (рисунок 2.5).

1	2	3	4
5	6	7	8
9	14	12	11
13	10		15

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

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

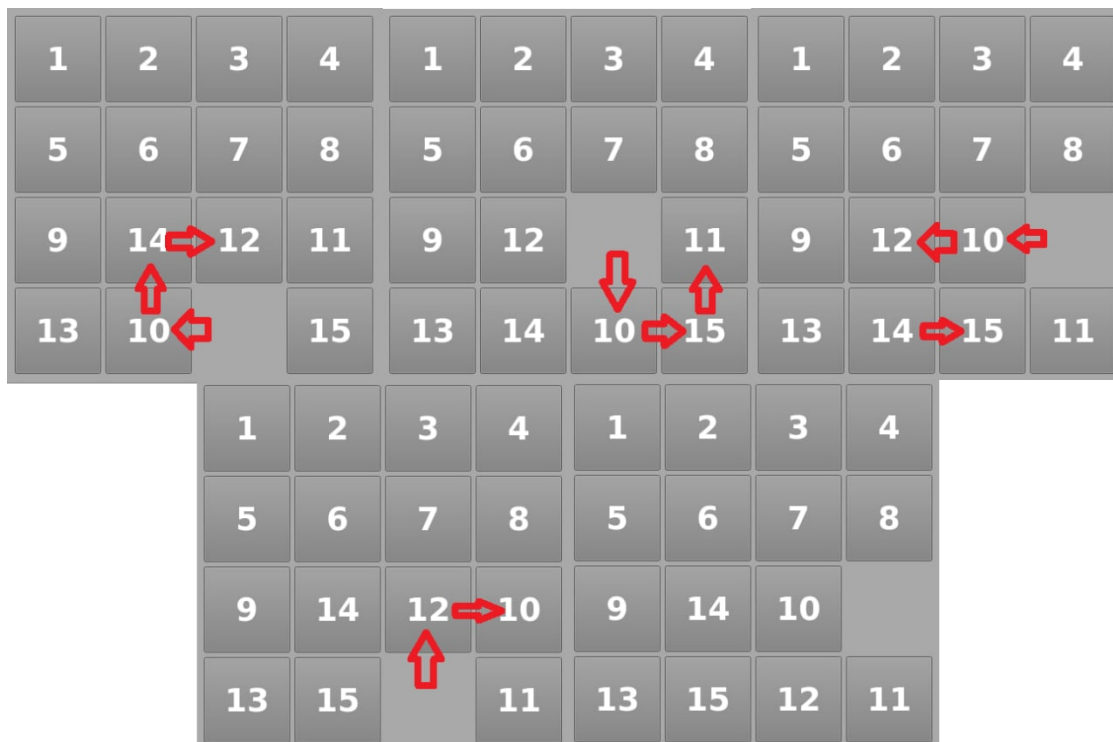


Рисунок 2.5 – Переміщення 10 на позицію 11

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

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

3 ОПИС АЛГОРИТМІВ

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

Таблиця 3.1 – Основні змінні та їхні призначення

Змінна	Призначення
seq	послідовність дій для розв'язку задачі
reord_cell	Зберігає клітинку, яку користувач може поміняти місцями при створенні початкового поля вручну
along	Напрямок уздовж
across	Напрямок 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

3.1 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:

6.1. ЯКЩО користувач наниснув на клітинку поруч із порожньою ТО перемістити її на порожнє місце:

6.1.1 Видалити перший елемент seq.

6.1.2 ЯКЩО переміщення не відповідає йому ТО створити новий масив seq.

6.2. ЯКЩО користувач натиснув на кнопку наступного кроку ТО виконати наступне переміщення розв'язку.

6.3. ЯКЩО користувач натиснув на кнопку автоматичного розв'язування, ТО:

6.3.1 ДОПОКИ seq не порожній ТА користувач не натискає кнопку зупинки виконувати наступне переміщення розв'язку.

7. КІНЕЦЬ

3.2 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

3.3 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 розв'язати:

2.1 ЯКЩО користувач натиснув на клітинку, ТО:

2.1.1 ЯКЩО reord_cell зберігає клітинку ТО поміняти її місцями з попередньою та очистити reord_cell.

2.1.2 ІНАКШЕ зберегти цю клітинку в reord_cell.

3. КІНЕЦЬ

3.4 Алгоритм створення масиву послідовності дій для розв'язання 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 п'ять позицій, додаючи усі переміщення пробілу у seq.
5. КІНЕЦЬ

3.5 Алгоритм заповнення рядка або стовпця відповідними значеннями

1. ПОЧАТОК
2. ЯКЩО заповнюється рядок 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:
 - 5.1. ЦИКЛ перебору всіх клітинок рядка, починаючи із напрямка протилежного along, окрім останньої:
 - 5.1.1. Перемістити клітинку на своє місце.
 - 5.1.2. Видалити відповідну вершину із графу .
 - 5.2. Перемістити останню клітинку у відповідну їй позицію, зміщену у напрямку across.
 - 5.3. Перемістити пробіл на позицію, зміщену на across від другої з кінця у лінії, що заповнюється, не переміщуючи при цьому останню клітинку.
 - 5.4. Виконати послідовність переміщень для пробіла: -across, along, across, -along, -across, -along, across.
 - 5.5. Видалити із графу вершину, що відповідає останній клітинці у лінії.

6. КІНЕЦЬ

3.6 Алгоритм підстановки клітинок 10 і 14

1. ПОЧАТОК

2. ЯКЩО ці дві клітинки на на своїх місцях, ТО:

- 2.1. Присвоїти змінній bot значення позиції нижньої із них.
- 2.2. Присвоїти змінній top значення позиції верхньої із них.
- 2.3. Перемістити клітинку, що відповідає bot у позицію top.
- 2.4. Перемістити пробіл на позицію справа від позиції bot.
- 2.5. 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.
- 2.6. Перемістити клітинку, що відповідає top у позицію справа від top.
- 2.7. Перемістити клітинку, що відповідає позиції bot, у відповідну їй позицію, не переміщуючи клітинку, що відповідає top.
- 2.8. Перемістити пробіл вправо, поставивши тим самим у top відповідне значення.

3. Видалити з графу вершини, що відповідають top та bot.

4. КІНЕЦЬ

3.7 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

3.8 Алгоритм переміщення клітинки у певну позицію

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

3.9 Алгоритм переміщення пробіла у певну позицію

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

3.10 Алгоритм знаходження найкоротшого шляху між вершинами у графі за допомогою пошуку вшир

1. ПОЧАТОК
2. Створити чергу.
3. Створити 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:
 - 6.1. ЦИКЛ перебору сусідніх вершин зі списку суміжності для v :

6.1.1. ЯКЩО сусідня вершина не перевірена ТО додати її у кінець черги та вказати вершину v як попередню їй.

6.2. Замінити v на вершину, вилучену з початку черги.

7. Створити масив результату.

8. ДОПОКИ v не є початковою вершиною:

8.1. Додати v у кінець масиву результату v .

8.2. Замінити вершину v на попередню для неї.

9. Змінити порядок у масиві результату на зворотній.

10. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Діаграма класів програмного забезпечення

Діаграма класів розробленого програмного забезпечення наведена на рисунку 4.1.

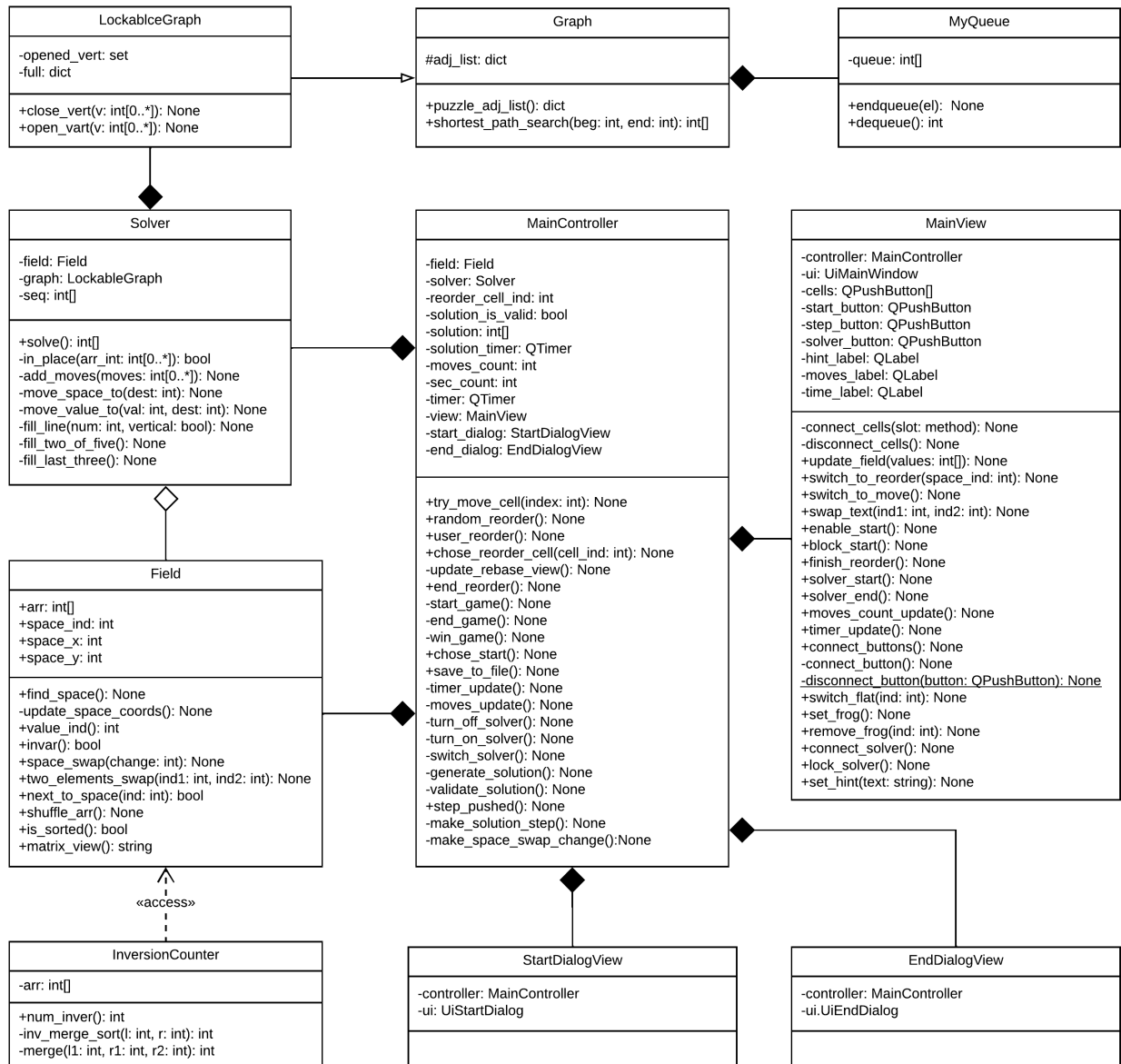


Рисунок 4.1 – Діаграма класів

У своїй програмі я реалізував шаблон проектування Модель-Представлення-Пред'явлення. Роль Моделі виконує клас Field, Представлення – клас MainView, Пред'явника – клас MainController.

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

StartDialogView та EndDialogView – класи діалогового 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 використовує клас InversionCounter для перевірки поля на можливість бути розв’язаним. Він використовує три методи які, обробляючи вміст масиву поля, повертають інформацію про кількість інверсій його непорожніх клітинок.

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

Під час виконання алгоритму Solver використовує об’єкт класу LockableGraph, який наслідується від класу Graph. Graph зберігає список суміжності, що представляє усі вершини графу та сусідні ребра для кожної із них, а також 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 об’єктом класу Queue, що надає можливість додавати елемент у кінець та вилучати з початку масиву.

4.2 Опис методів частин програмного забезпечення

4.2.1 Стандартні методи

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

Таблиця 4.1 – Стандартні методи

№ п/ п	Назва заголовно го файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параметр ів	Опис вихідних параметрі в
1		UiMainWi ndow	setupUi	Створення віджетів головного вікна та визначення їх початкових характеристик		
2		QAbstract Button	setText	Встановлення тексту у віджет	Текст, що буде встановле ний	
		QLabel	setText	Встановлення тексту у віджет label	Текст, що буде встановле ний	
3		QAbstract Button	text	Отримання тексту кнопки		Текст кнопки
4		QPushButt on	setFlat	Визначення параметра Flat кнопки	Стан Flat	
5		QPushButt on	isFlat	Отримання значення Flat кнопки		Булеве значення, що відповідає параметру Flat

Продовження таблиці 4.1

№ п/ п	Назва заголовно го файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параметр ів	Опис вихідних параметрі в
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 записаний 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 заголовно го файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параметр ів	Опис вихідних параметрі в
12	time		time	Отримання Unix Time		Unix Time у секундах
13	copy		deepcopy	Створює копію об'єкта	Об'єкт	Копія вхідного об'єкту
14	random		shuffle	Змінює порядок об'єктів 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, пов'язаної із об'єктом		
19		pyqtBound Signal	connect	підключає сигнал до функції	Функція, що підключає ться	

4.2.2 Користувацькі методи

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

Таблиця 4.2 – Користувацькі методи

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
1		MainCont roller	try_move _cell	Переміщення клітинки за можливості та перевірка чи не призвело воно до перемоги	Індекс клітинк и, що переміщ ується	
2		MainCont roller	random_r eorder	Початок нової гри з випадково згенерованим полем		
3		MainCont roller	user_reor der	Перехід до режиму перестановки клітинок		
4		MainCont roller	chosed_reo rder_cell	Зміна розташування клітинок у режимі перестановки клітинок	Індекс однієї із клітинк и, що переста вляютьс я	

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
5		MainCont roller	update_re base_vie w	Оновлення інтерфейсу при користувацькому задаванню		
6		MainCont roller	end_reord er	Вихід із режиму перестановок		
7		MainCont roller	start_gam e	Початок гри		
8		MainCont roller	end_game	Завершення гри		
9		MainCont roller	win_gam e	Сповіщення користувача про перемогу. Відкривання вікна, запиту на збереження у файл		
10		MainCont roller	chose_sta rt	Передчасне завершення гри та відкривання вікна вибору режиму задавання поля		

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
11		MainCont roller	save_to_f ile	Збереження інформації про гру у файл		
12		MainCont roller	time_upd ate	Оновлення таймера		
13		MainCont roller	moves_up date	Оновлення лічильника кількості кроків		
14		MainCont roller	turn_off_ solver	Вихід із режиму автоматичного розв'язування		
15		MainCont roller	turn_on_s olver	Перехід у режим автоматичного розв'язування		
16		MainCont roller	switch_so lver	Переключення режиму автоматичного розв'язування		

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
17		MainCont roller	generate_ solution	Отримання розв'язку поточного стану поля		
18		MainCont roller	validate_s olution	Оновлення розв'язок, якщо він неактуальний		
19		MainCont roller	step_push ed	Виконання наступний крок розв'язку		
20		MainCont roller	make_sol ution_ste p	Оновлення розв'язання та виконання наступного кроку розв'язку при натисканні на кнопку		

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
21		MainCont roller	make_spa ce_swap	Виконання переміщення пробілу у певному напрямку	Ціле число, що відповід ає напрямок у переміщ ення	
22		MainVie w	connect_c ells	Підключення усіх клітинок поля до певного методу	Метод, який буде підключ ено	
23		MainVie w	disconnec t_cells	Відключення усіх клітинок поля від підключеного методу		

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
24		MainView	update_filed	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 заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
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 заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
30		MainView	finish_reorder	Переведення інтерфейсу в ігровий режим		
31		MainView	solver_start	Зміна тексту кнопки автоматичного розв'язання на «Зупинити»		
32		MainView	solver_end	Зміна тексту кнопки автоматичного розв'язування на «Автоматичне розв'язування»		
33		MainView	moves_counter_update	Оновлення відображення ліч	Нове значення лічильника	

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
34		MainView	timer_update	Оновлення відображення таймера	Нове значенн я кількост і 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 підключ ається	
37		MainView	disconnect_button	Підключення кнопки від сигналу, якщо він підключений	Кнопка	

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
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 індексом	Індекс клітинки	
41		MainView	connect_solver	Підключення кнопок наступного кроку та автоматичного розв'язування		

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
42		MainView	lock_solver	Відключення кнопок наступного кроку та автоматичного розв'язування		
43		MainView	set_hint	Заміна тексту напису-підказки на вказаний	Новий текст підказки	
44		Field	find_space	Знаходження індексу пробілу		
45		Field	update_spaces_coords	Оновлення даних про стовпець та рядок пробілу		
46		Field	value_index	Знаходження індексу клітинки із вказаним значенням	Значення клітинки	Індекс клітинки

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
47		Field	invar	Перевірка, чи може поле бути розв'язаним		Булеве значення, що відповіда є коректно сті заданого поля
48		Field	space_sw ap	Переміщення пробілу у вказаному напрямку, вираженим числом	Зміна індекса пробіла для однови мірного масиву	
49		Field	two_elem ents_swa p	Зміна місцями двох клітинок та оновлення даних про розташування пробілу	Індекс першої клітинк и; Індекс другої клітинк и	

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
50		Field	next_to_s pace	Перевірка, чи знаходиться клітинка за вказаним індексом поруч із пробілом	Індекс клітинк и	відповіда є розташув анню поруч із пробілом
51		Field	shuffle_ar r	Зміна розташування клітинок поля на випадкову відмінну від початкової та відсортованої		
52		Field	is_sorted	Перевірка, чи відсортоване поле		Булеве значення, що відповіда є відсортов аності поля

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
53		Field	matrix_vie w	Створення рядка, що представляє поле у вигляді матриці		Рядок: масив поля представ лений у вигляді матриці
54		Solver	solve	Генерує масив- розв'язання		Послідов ність переміще нь пробілу, представ лених у вигляді цилих чисел

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
55		Solver	in_place	Перевіряє, чи знаходяться на вказаних позиціях клітинки, що відповідають фінальному розташуванню	Цілочис ельний масив: індекси позицій поля	Булеве значення: відповіда є розташув анню відповідн их значень в усіх передани х позиціях
56		Solver	add_moves	Додавання переміщень, представлених цілими числами у масив	Перемі щення	
57		Solver	move_space_to	Переміщення пробілу в указану позицію	Індекс позиції- цілі	

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
58		Solver	move_val ue_to	Переміщення клітинки із вказаним значенням у вказану позицію	Значенн я клітинк и; Кінцева позиція	
59		Solver	fill_line	Заповнення лінії під вказаним номером на із вказаною орієнтацією відповідними значення	Номер рядка/ст овпця; Булеве значенн я відповід ає вертика льний орієнта ції	

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
60		Solver	fill_two_ of_five	Заповнення відповідними значеннями двох лівих клітинок прямокутника 2 на 3 в правому нижньому кутку поля		
61		Solver	fill_last_t hree	Заповнення відповідними значеннями квадрата 2 на 2 у правому нижньому кутку поля		
62		Graph	puzzle_ad j_list	Повернення списку суміжності для графа, що відповідає клітинкам ігрового поля та їх спільним сторонам		Словник: список суміжнос ті

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
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		Lockable Graph	close_vert	Видалення вершини із графу		
65		Lockable Graph	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		Inversion Counter	num_inverter	Підрахунок кількості інверсій у масиві		Кількість інверсій

Продовження таблиці 4.2

№ п/п	Назва заголов ного файлу	Назва класу	Назва функції	Призначення функції	Опис вхідних параме трів	Опис вихідних парамет рів
69		Inversion Counter	inv_merg e_sort	Рекурсивний обрахунок кількості інверсій, використовуючи сортування злиттям	Ліва межа підмаси ву; Права межа підмаси ву	Кількість інверсій для вказаног о підмасив у
70		Inversion Counter	merge	Злиття масивів при сортуванні з поверненням кількості перестановок	ліва межа лівого підмаси ву; Права межа лівого підмаси ву; Права межа правого підмаси ву	Кількість перестан овок

5 ПЛАН ТЕСТУВАННЯ

5.1 План тестування

а) 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 відсортowanego стану поля, під час гри

б) Натискання на кнопку «Почати нову гру»

- 1) Тестування роботи програми при натисканні на кнопку під час гри, коли автоматичне розв'язання вимкнене
- 2) Тестування роботи програми при натисканні на кнопку під час гри, коли автоматичне розв'язання увімкнене
- 3) Тестування роботи програми при натисканні на кнопку поза грою
- 4) Тестування роботи програми під час користувацького задавання поля, коли поле може бути розв'язаним
- 5) Тестування роботи програми під час користувацького задавання поля, коли поле не може бути розв'язаним

в) Натискання на кнопки «Наступний крок» та «Автоматичне розв'язування»

- 1) Тестування роботи програми при натисканні на кнопки поза грою
- 2) Тестування роботи програми при натисканні на кнопки під час гри, коли автоматичне розв'язування вимкнене
- 3) Тестування роботи програми при натисканні на кнопки під час гри, коли автоматичне розв'язування увімкнене

г) 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
- д) Режим користувацького задавання поля
- 1) Тестування першого натиску
 - 2) Тестування другого натиску

5.2 Приклади тестування

Результати тестувань наведено у таблицях 5.1 – 5.22.

Таблиця 5.1 – Тестування роботи програми при натисканні на порожню клітинку

Мета тесту	Перевірка реакції програми на натискання на порожню клітинку
Початковий стан програми	Стан поля зображений 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



10	5		11
3	2	8	4
6	7	14	9
12	15	1	13

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

Таблиця 5.2 – Тестування роботи програми при натисканні на клітинку сусідню із порожньою

Мета тесту	Перевірити можливість переміщувати клітинки поля
Початковий стан програми	Стан поля зображений 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 очікуванням

10	5		11
3	2	8	4
6	7	14	9
12	15	1	13

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

Таблиця 5.3 – 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 зображений 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 очікуванням

10		5	11
3	2	8	4
6	7	14	9
12	15	1	13

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

Таблиця 5.4 – 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 зображений 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. З'явилося 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

1	2	3	4
5	6	7	8
9	10	11	
13	14	15	12

Рисунок 5.4 – Початковий стан поля при тестуванні, що наведене у таблиці 5.4

Таблиця 5.5 – Тестування роботи програми при натисканні на кнопку нової гри під час гри, коли автоматичне розв’язання 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 та змінюють колір кнопки автоматичної гри та наступного кроку. Відкривається вікно вибору способу задавання поля
Стан програми після проведення випробувань	Збігається з очікуваним

Таблиця 5.6 – Тестування роботи програми при натисканні на кнопку під час гри, коли автоматичне розв’язання 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 розв’язування
Вхідні дані	Інформація про натискання на кнопку нової гри
Схема проведення тесту	Натискання на кнопку нової гри
Очікуваний результат	Зупиняється автоматичне розв’язання, таймер зупиняється та обнуляється, обнуляється лічильник зроблених переміщень. Зникає можливість взаємодіяти з головним вікном. Стають неактивними та змінюють колір кнопки автоматичної гри та наступного кроку. Відкривається вікно вибору способу задавання поля
Стан програми після проведення випробувань	Збігається з очікуваним

Таблиця 5.7 – 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

Продовження таблиці 5.7

Очікуваний результат	Не працює таймер, кнопки автоматичної гри та наступного кроку не активні, головне вікно стає неактивним, відкривається вікно вибору способу задавання поля
Стан програми після проведення випробувань	Збігається з очікуванням

Таблиця 5.8 – Тестування роботи програми при натисканні кнопки нової гри під час користувацького задавання поля, коли поле може бути розв’язаним

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 очікуванням

Таблиця 5.9 – Тестування роботи програми при натисканні кнопки нової гри під час користувацького задавання поля, коли поле не може бути розв’язаним

Мета тесту	Перевірка відсутності можливості початку гри, якщо поле не може бути розв’язаним
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

Продовження таблиці 5.9

Очікуваний результат	Нічого не змінилося
Стан програми після проведення випробувань	Збігається з очікуванням

Таблиця 5.10 – Тестування роботи програми при натисканні на кнопки наступного кроку і автоматичного розв’язання 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 очікуванням

Таблиця 5.11 – Тестування роботи програми при натисканні на кнопки наступного кроку і автоматичного розв’язання під час гри, коли автоматичне розв’язування вимкнене

Мета тесту	Перевірка можливості зробити крок автоматично та ввімкнути автоматичне розв’язування
Початковий стан програми	Працює таймер, кнопки автоматичної гри та наступного кроку активні, автоматичне розв’язування не відбувається
Вхідні дані	інформація про натискання на кнопку наступного кроку; інформація про натискання на кнопку автоматичної гри

Продовження таблиці 5.11

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

Таблиця 5.12 – Тестування роботи програми при натисканні на кнопки наступного кроку і автоматичного розв’язання під час гри, коли автоматичне розв’язування увімкнене

Мета тесту	Перевірка виконання переміщення при натисканні на кнопку наступного кроку під час автоматичного розв’язування; Перевірка можливості зупинити автоматичне розв’язування
Початковий стан програми	Працює таймер, кнопки наступного кроку та автоматичної гри активні, виконується автоматичне розв’язування
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 розв’язування зупиняється
Стан програми після проведення випробувань	Збігається з очікуваним

Таблиця 5.13 – 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 способу 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 очікуванням

Таблиця 5.14 – 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 способу 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

Продовження таблиці 5.14

Стан програми після проведення випробувань	Збігається з очікуваним
---	-------------------------

Таблиця 5.15 – 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

Таблиця 5.16 – 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

Таблиця 5.17 – 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

Таблиця 5.18 – 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

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Робота з програмою

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

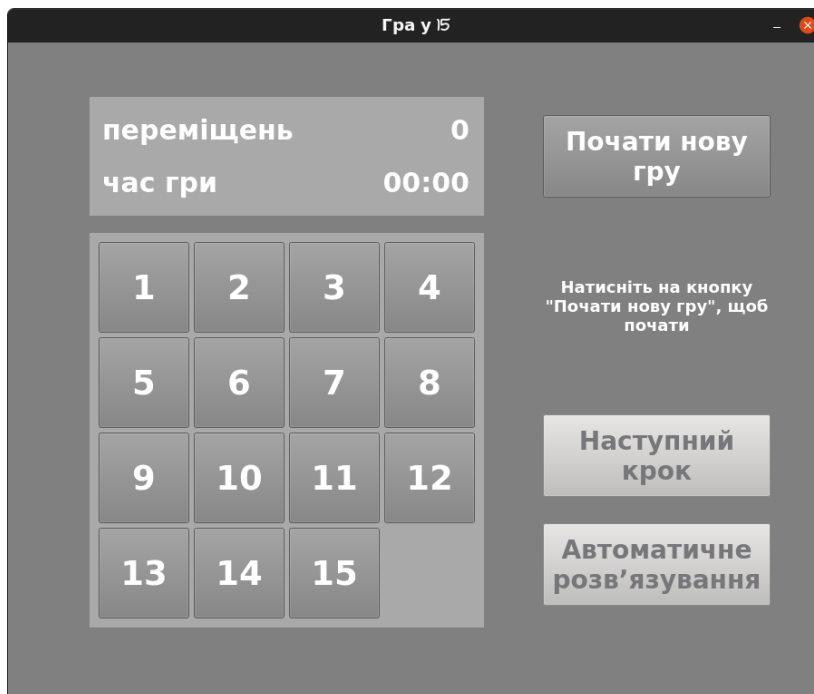


Рисунок 6.1 – Головне вікно програми

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

Відкриється вікно способу задавання стартового поля (рисунок 6.2).

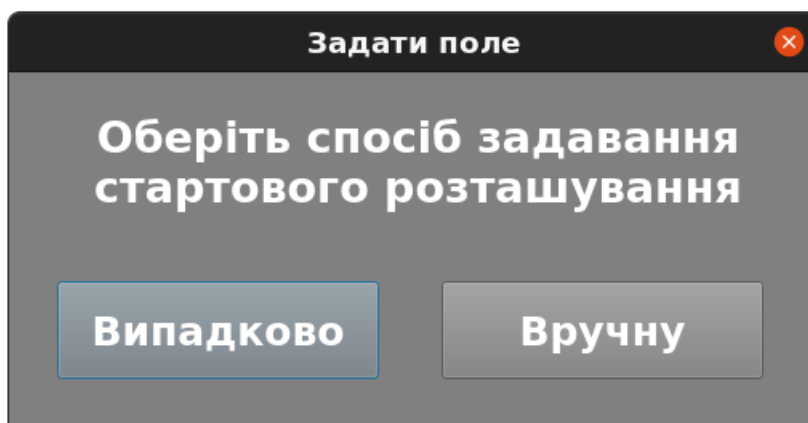


Рисунок 6.2 – Вікно вибору способу задавання стартового поля

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



Рисунок 6.3 – Головне вікно під час гри

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

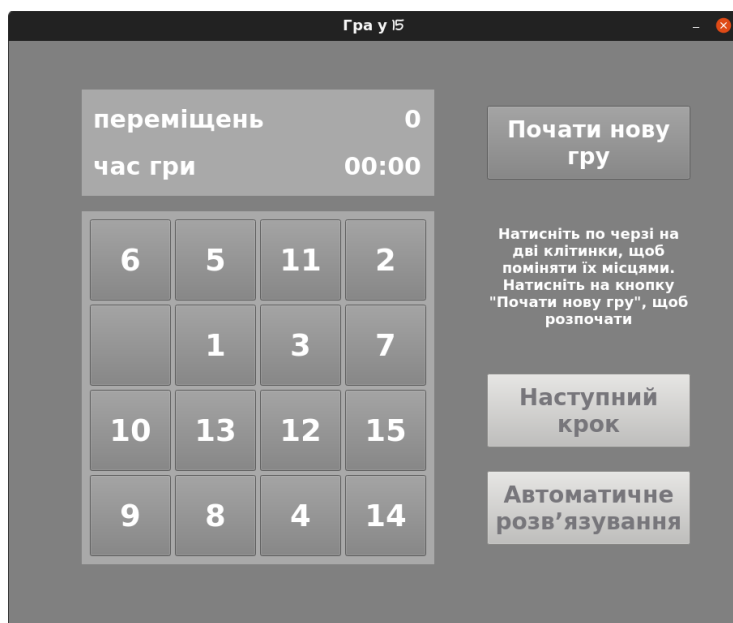


Рисунок 6.4 – Головне вікно під час користувацького задавання поля

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

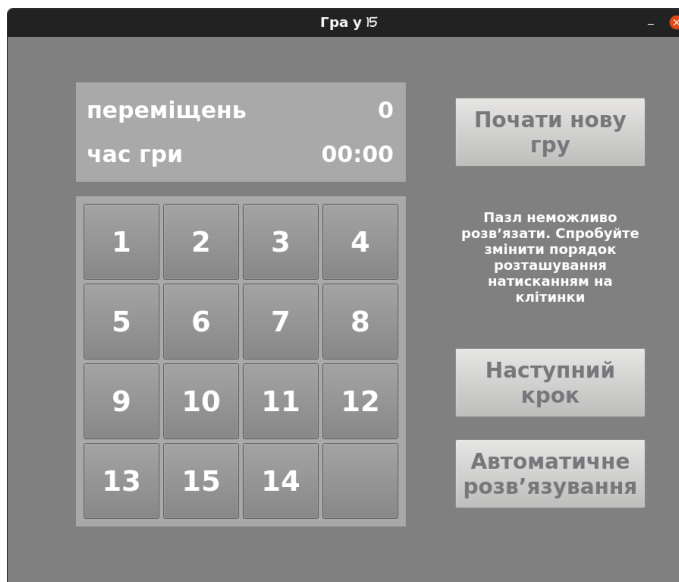


Рисунок 6.5 – Головне вікно під час користувацького задавання поля у випадку відсутності розв'язку

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 розв'язування. Після натискання відповідної кнопки пазл почне розв'язуватися самостійно (рисунок 6.6).



Рисунок 6.6 – Автоматичне розв'язання

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

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

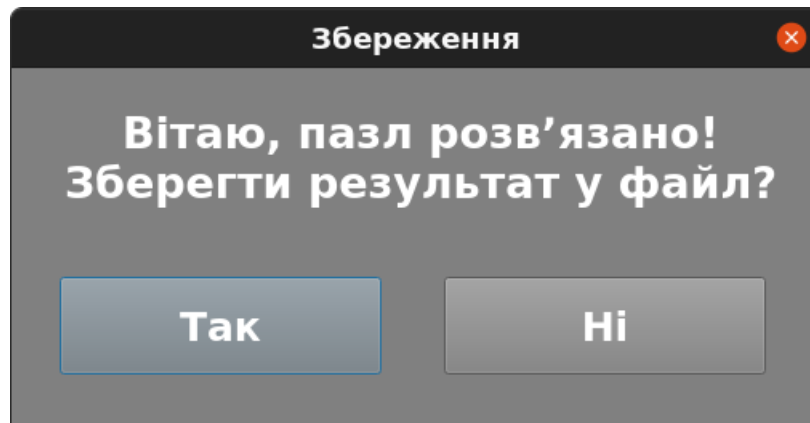


Рисунок 6.7 – Діалогове вікно збереження у файл

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

6.2 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.

6.3 Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	Windows 7/ Windows 8/Windows 10 (з останніми оновленнями)	Windows 10 (з останніми оновленнями)

Продовження таблиці 6.1

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	1600x1024	1920x1080 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 розв'язання задачі.

У розділі 3 викладено покрокове виконання основних алгоритмів, що використовуються для пошуку розв'язку головоломки.

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

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

У розділі 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» квітня 2022 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 розв'язок;
- 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 01.06.2022р.)
- 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

Тексти програмного коду програмного забезпечення

Вирішення та генерація головоломки «Гра у 15»

(Найменування програми (документа))

Електронний носій

(Вид носія даних)

23 арк, 23 Кб

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

студента групи ІП-15 І курсу

Мельника Данила Євгенійовича

31 арк, 23 Кб

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

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):                # Пересуває клітинку, якщо це можливо
        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):          # Випадкова генерація поля
    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):           # Перехід до режиму користувацького задавання
поля
    self.__view.switch_to_reorder(self.__field.space_ind)
    self.__update_rebase_view()

```

```

def chose_reorder_cell(self, cell_ind):  # Зміна клітинок місцями під час
користувацького задавання
    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):        # Оновлення інтерфейсу в залежності від
наявності розв'язання
    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):    # Вихід із режиму користувацького задавання та
    початок гри
        self.__view.switch_flat(self.__field.space_ind)
        self.__view.finish_reorder()
        self.__start_game()

    def __start_game(self):    # Початок гри
        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):    # Завершення гри
        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):    # Перемога у грі
        self.__view.set_frog()
        self.__end_dialog.exec()

    def chose_start(self):    # Натискання на кнопку початку нової гри
        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):    # Збереження даних у файл
        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):    # Оновлення таймера
    self.__sec_count += 1
    self.__view.timer_update(self.__sec_count)

def __moves_update(self):    # Оновлення кількості переміщень
    self.__moves_count += 1
    self.__view.moves_count_update(self.__moves_count)

def __turn_off_solver(self):    # Вимкнення автоматичного розв'язання
    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):    # Увімкнення автоматичного розв'язання
    self.__validate_solution()
    self.__solution_timer.start(SOLVE_INTERVAL)
    self.__view.solver_start()
    self.__view.set_hint(SOLVER_HINT)

def switch_solver(self):    # Перемикання режиму автоматичного розв'язання
    if self.__solution_timer.isActive():
        self.__turn_off_solver()
    else:
        self.__turn_on_solver()

def __generate_solution(self):    # Генерація розв'язку
    self.__solver = Solver(deepcopy(self.__field))
    self.__solution = self.__solver.solve()

def __validate_solution(self):    # Перевірка актуальності розв'язку
    if not self.__solution_is_valid:
        self.__generate_solution()
        self.__solution_is_valid = True

def step_pushed(self):    # Натискання на кнопку наступного кроку
    self.__validate_solution()
    self.__make_solution_step()

```

```

def __make_solution_step(self): # Виконання наступного кроку,
    self.__make_space_swap(self.__solution.pop(0))
    if not self.__solution:
        self.__end_game()
        self.__win_game()

def __make_space_swap(self, change): # Переміщення пробілу
    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()

```

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): # Підключення сигналів кнопок до методу
    for cell in self.__cells:
        cell.pressed.connect(partial(slot, self.__cells.index(cell)))

def __disconnect_cells(self): # Відключення методів від кнопок
    for cell in self.__cells:
        cell.disconnect()

def update_field(self, values): # оновлення поля
    for button, value in zip(self.__cells, values):
        button.setText(str(value).replace(str(FIELD_SIZE), ""))

def switch_to_reorder(self, space_ind): # Перемикання у режим користувацького
задавання поля
    self.__disconnect_cells()
    self.__connect_cells(self.__controller.chose_reorder_cell)
    self.__cells[space_ind].setFlat(False)

def switch_to_move(self): # Перемикання у режим переміщення
    self.__disconnect_cells()
    self.__connect_cells(self.__controller.try_move_cell)

def swap_text(self, ind1, ind2): # Обмін текстом між кнопками
    cell1 = self.__cells[ind1]
    cell2 = self.__cells[ind2]
    saved = cell1.text()
    cell1.setText(cell2.text())

```

```

cell2.setText(saved)

def enable_start(self): # увімкнення можливості почати гру
    self.__connect_button(self.__start_button, self.__controller.end_reorder)

def block_start(self): # блокування можливості почати гри
    self.__disconnect_button(self.__start_button)

def finish_reorder(self): # вихід із режиму користувацького задавання поля
    self.__connect_button(self.__start_button, self.__controller.chose_start)
    self.switch_to_move()

def solver_start(self): # зміна тексту кнопки автоматичного розв'язування при
    його увімкненні
    self.__solver_button.setText(OFF_SOLVER_BUTTON)

def solver_end(self): # зміна тексту кнопки автоматичного розв'язування при
    його вимиканні
    self.__solver_button.setText(ON_SOLVER_BUTTON)

def moves_count_update(self, moves_count): # оновлення тексту лічильника
    кроків
    self.__moves_label.setText(str(moves_count))

def timer_update(self, sec_count): # оновлення відображення таймера
    self.__time_label.setText(f'{sec_count // MIN_TO_SEC:02}:{sec_count %
MIN_TO_SEC:02}')

def connect_buttons(self): # підключення кнопок інтерфейсу
    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): # підключення окремої кнопки
    self.__disconnect_button(button)
    button.clicked.connect(slot)
    button.setStyleSheet("")

@staticmethod
def __disconnect_button(button): # відключення кнопки
    try:
        button.disconnect() # повертає помилку, якщо жодний сигнал не підключено
        button.setStyleSheet(BLOCKED_STYLESHEET)
    except TypeError:

```

```

    pass

def switch_flat(self, ind): # перемикає стан Flat кнопки
    cell = self.__cells[ind]
    cell.setFlat(not cell.isFlat())

def set_frog(self): # змінює текст порожньої клітинки
    self.__cells[-1].setText(FROG)

def remove_frog(self, ind): # прибирає текст порожньої клітинки
    self.__cells[ind].setText("")

def connect_solver(self): # підключення кнопок розв'язування
    self.__connect_button(self.__step_button, self.__controller.step_pushed)
    self.__connect_button(self.__solver_button, self.__controller.switch_solver)

def lock_solver(self): # підключення кнопок розв'язування
    self.__disconnect_button(self.__step_button)
    self.__disconnect_button(self.__solver_button)

def set_hint(self, text): # зміна тексту напису-підказки
    self.__hint_label.setText(text)

```

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): # розв'язування ігрового поля
        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): # перевірка чи відповідають вершини кінцевому
розташуванню
    for ind in arr_ind:
        if self.__field.arr[ind] != ind + 1:
            return False
    return True

def __add_moves(self, *moves): # додавання переміщень у результуючий масив
    for move in moves:
        self.__seq.append(move)
        self.__field.space_swap(move)

def __move_space_to(self, dest): # переміщення пробілу у вказану позицію
    prev = self.__field.space_ind
    path = self.__graph.shortest_path_search(prev, dest)
    for i in path:
        self.__add_moves(i - prev)
        prev = i

def __move_value_to(self, val, dest): # переміщення клітинки із вказаним
значенням у задану позицію
    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)
        self.__graph.open_vert(prev)
        self.__add_moves(prev - i)
        prev = i

def __fill_line(self, num, vertical): # заповнення рядка або стовпця кінцевими
значеннями
    if vertical:
        along, across = DOWN, RIGHT # заповнення стовпця
        beg, end = (num + 1) * FIELD_SIDE + num, FIELD_SIDE*(FIELD_SIDE-1) +
num
    else:
        along, across = RIGHT, DOWN # заповнення рядка
        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): # заповнення третього стовпця з кінця
    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): # заповнення останніх трьох позицій
    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)

```

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): # оновлення параметрів індексів пробілу
        self.space_ind = self.value_ind(FIELD_SIZE)
        self.__update_space_coords()

    def __update_space_coords(self): # оновлення параметрів стовпця та рядка
        пробілу
        self.space_x, self.space_y = \
            self.space_ind % FIELD_SIDE, self.space_ind // FIELD_SIDE

    def value_ind(self, val): # повертає індекс певної клітинки
        return self.arr.index(val)

    def invar(self): # перевіряє, чи може поле бути розв'язаним
        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): # змінює пробіл із сусідньою клітинкою
        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): # змінює місцями дві клітинки за
        індексами
        self.arr[ind1], self.arr[ind2] = self.arr[ind2], self.arr[ind1]
        self.find_space()

    def next_to_space(self, ind): # перевіряє чи знаходиться клітинка поруч із
        пробілом
        return self.space_ind - ind in DIRECTIONS

```

```

def shuffle_arr(self): # змінює поле випадковим чином
    prev = self.arr.copy()
    while self.is_sorted() or self.arr == prev:
        shuffle(self.arr)
    self.find_space()

def is_sorted(self): # перевіряє чи відсортоване поле
    return self.arr == list(range(1, FIELD_SIZE + 1))

def matrix_view(self): # повертає рядок матричного представлення поля
    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

```

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(): # повертає список суміжності для гри у 15
        _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): # знаходить найкоротший шлях між
вершинами

```

```

    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): # видаляє вершину
    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): # повертає раніше видалену вершину
    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

```

InversionCounter.py

```

class InversionCounter:
    def __init__(self, arr):
        self.__arr = arr

    def num_inver(self): # обраховує кількість інверсій в масиві
        return self.__inv_merge_sort(0, len(self.__arr) - 1)

    def __inv_merge_sort(self, l, r): # обраховує кількість інверсій у підмасиві
        if l == r:
            return 0
        m = (l + r) >> 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): # об'єднує два підмасиви та повертає кількість
інверсій
        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

```

MyQueue.py

```

class MyQueue:
    def __init__(self):

```

```

self.__queue = []

def enqueue(self, el): # додає елемент в кінець черги
    self.__queue.append(el)

def dequeue(self): # видаляє та повертає перший елемент черги
    return self.__queue.pop(0)

```

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)

```

UiMainWindow.py

```

from PyQt5 import QtCore, QtWidgets

```

```

class UiMainWindow(object):
    def setupUi(self, MainWindow): # налаштує інтерфейс
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(900, 725)
        MainWindow.setMinimumSize(QtCore.QSize(900, 725))
        MainWindow.setMaximumSize(QtCore.QSize(900, 725))
        MainWindow.setSizeIncrement(QtCore.QSize(612, 826))
        MainWindow.setStyleSheet("QLabel{font:600 18pt \"FreeMono\" bold;\n"
            "                color: white}\n"
            "QLabel#label_5{font-size: 18px}\n"
            "QPushButton{font:600 22pt \"FreeMono\" bold;\n"
            "                color: white}\n"
            "QWidget{background-color: darkgray}\n"
            "QLabel{background-color: none}\n"
            "QWidget#centralwidget{background-color: gray}\n"
            "QPushButton{background-color: gray}\n"
            "QPushButton#pushButton_17{font:600 17pt \"FreeMono\"
bold;\n"
            "                color: white}\n"
            "QPushButton#pushButton_18{font:600 17pt \"FreeMono\"
bold;\n"
            "                color: white}\n"
            "QPushButton#pushButton_19{font:600 17pt \"FreeMono\"
bold;\n"
            "                color: white}")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayoutWidget_2 = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget_2.setGeometry(QtCore.QRect(90, 210, 435, 435))
        self.gridLayoutWidget_2.setObjectName("gridLayoutWidget_2")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.gridLayoutWidget_2)
        self.gridLayout_2.setContentsMargins(10, 10, 10, 10)
        self.gridLayout_2.setSpacing(5)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.pushButton_4 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

        sizePolicy.setHeightForWidth(self.pushButton_4.sizePolicy().hasHeightForWidth())
        self.pushButton_4.setSizePolicy(sizePolicy)

```

```

self.pushButton_4.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_4.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_4.setSizeIncrement(QtCore.QSize(0, 0))
self.pushButton_4.setObjectName("pushButton_4")
self.gridLayout_2.addWidget(self.pushButton_4, 0, 3, 1, 1)
self.pushButton_8 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_8.sizePolicy().hasHeightForWidth())
self.pushButton_8.setSizePolicy(sizePolicy)
self.pushButton_8.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_8.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_8.setObjectName("pushButton_8")
self.gridLayout_2.addWidget(self.pushButton_8, 1, 3, 1, 1)
self.pushButton_7 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_7.sizePolicy().hasHeightForWidth())
self.pushButton_7.setSizePolicy(sizePolicy)
self.pushButton_7.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_7.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_7.setObjectName("pushButton_7")
self.gridLayout_2.addWidget(self.pushButton_7, 1, 2, 1, 1)
self.pushButton_11 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_11.sizePolicy().hasHeightForWidth())
self.pushButton_11.setSizePolicy(sizePolicy)
self.pushButton_11.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_11.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_11.setObjectName("pushButton_11")
self.gridLayout_2.addWidget(self.pushButton_11, 2, 2, 1, 1)
self.pushButton_14 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)

```

```
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_14.sizePolicy().hasHeightForWidth())
self.pushButton_14.setSizePolicy(sizePolicy)
self.pushButton_14.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_14.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_14.setObjectName("pushButton_14")
self.gridLayout_2.addWidget(self.pushButton_14, 3, 1, 1, 1)
self.pushButton_9 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_9.sizePolicy().hasHeightForWidth())
self.pushButton_9.setSizePolicy(sizePolicy)
self.pushButton_9.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_9.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_9.setObjectName("pushButton_9")
self.gridLayout_2.addWidget(self.pushButton_9, 2, 0, 1, 1)
self.pushButton_3 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_3.sizePolicy().hasHeightForWidth())
self.pushButton_3.setSizePolicy(sizePolicy)
self.pushButton_3.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_3.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_3.setSizeIncrement(QtCore.QSize(0, 0))
self.pushButton_3.setObjectName("pushButton_3")
self.gridLayout_2.addWidget(self.pushButton_3, 0, 2, 1, 1)
self.pushButton = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth())
self.pushButton.setSizePolicy(sizePolicy)
self.pushButton.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton.setObjectName("pushButton")
```

```

self.gridLayout_2.addWidget(self.pushButton, 0, 0, 1, 1)
self.pushButton_15 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_15.sizePolicy().hasHeightForWidth())
self.pushButton_15.setSizePolicy(sizePolicy)
self.pushButton_15.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_15.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_15.setObjectName("pushButton_15")
self.gridLayout_2.addWidget(self.pushButton_15, 3, 2, 1, 1)
self.pushButton_16 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_16.sizePolicy().hasHeightForWidth())
self.pushButton_16.setSizePolicy(sizePolicy)
self.pushButton_16.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_16.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_16.setText("")
self.pushButton_16.setFlat(True)
self.pushButton_16.setObjectName("pushButton_16")
self.gridLayout_2.addWidget(self.pushButton_16, 3, 3, 1, 1)
self.pushButton_5 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_5.sizePolicy().hasHeightForWidth())
self.pushButton_5.setSizePolicy(sizePolicy)
self.pushButton_5.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_5.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_5.setSizeIncrement(QtCore.QSize(0, 0))
self.pushButton_5.setObjectName("pushButton_5")
self.gridLayout_2.addWidget(self.pushButton_5, 1, 0, 1, 1)
self.pushButton_13 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)

```

```
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_13.sizePolicy().hasHeightForWidth())
self.pushButton_13.setSizePolicy(sizePolicy)
self.pushButton_13.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_13.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_13.setObjectName("pushButton_13")
self.gridLayout_2.addWidget(self.pushButton_13, 3, 0, 1, 1)
self.pushButton_12 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_12.sizePolicy().hasHeightForWidth())
self.pushButton_12.setSizePolicy(sizePolicy)
self.pushButton_12.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_12.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_12.setObjectName("pushButton_12")
self.gridLayout_2.addWidget(self.pushButton_12, 2, 3, 1, 1)
self.pushButton_2 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_2.sizePolicy().hasHeightForWidth())
self.pushButton_2.setSizePolicy(sizePolicy)
self.pushButton_2.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_2.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_2.setObjectName("pushButton_2")
self.gridLayout_2.addWidget(self.pushButton_2, 0, 1, 1, 1)
self.pushButton_10 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
```

```
sizePolicy.setHeightForWidth(self.pushButton_10.sizePolicy().hasHeightForWidth())
self.pushButton_10.setSizePolicy(sizePolicy)
self.pushButton_10.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_10.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_10.setObjectName("pushButton_10")
self.gridLayout_2.addWidget(self.pushButton_10, 2, 1, 1, 1)
```

```

self.pushButton_6 = QtWidgets.QPushButton(self.gridLayoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_6.sizePolicy().hasHeightForWidth())
self.pushButton_6.setSizePolicy(sizePolicy)
self.pushButton_6.setMinimumSize(QtCore.QSize(100, 100))
self.pushButton_6.setMaximumSize(QtCore.QSize(100, 100))
self.pushButton_6.setObjectName("pushButton_6")
self.gridLayout_2.addWidget(self.pushButton_6, 1, 1, 1, 1)
self.pushButton_18 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_18.setGeometry(QtCore.QRect(590, 410, 251, 91))
self.pushButton_18.setObjectName("pushButton_18")
self.pushButton_19 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_19.setGeometry(QtCore.QRect(590, 530, 251, 91))
self.pushButton_19.setObjectName("pushButton_19")
self.gridLayoutWidget_3 = QtWidgets.QWidget(self.centralwidget)
self.gridLayoutWidget_3.setGeometry(QtCore.QRect(90, 60, 435, 131))
self.gridLayoutWidget_3.setObjectName("gridLayoutWidget_3")
self.gridLayout_3 = QtWidgets.QGridLayout(self.gridLayoutWidget_3)
self.gridLayout_3.setSizeConstraint(QtWidgets.QLayout.SetMinimumSize)
self.gridLayout_3.setContentsMargins(10, 10, 10, 10)
self.gridLayout_3.setSpacing(5)
self.gridLayout_3.setObjectName("gridLayout_3")
self.label_2 = QtWidgets.QLabel(self.gridLayoutWidget_3)
self.label_2.setMinimumSize(QtCore.QSize(129, 0))
self.label_2.setMaximumSize(QtCore.QSize(129, 16777215))
self.label_2.setAlignment(QtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
self.label_2.setObjectName("label_2")
self.gridLayout_3.addWidget(self.label_2, 0, 1, 1, 1)
self.label = QtWidgets.QLabel(self.gridLayoutWidget_3)
self.label.setMinimumSize(QtCore.QSize(267, 0))
self.label.setMaximumSize(QtCore.QSize(267, 16777215))
self.label.setObjectName("label")
self.gridLayout_3.addWidget(self.label, 0, 0, 1, 1)
self.label_3 = QtWidgets.QLabel(self.gridLayoutWidget_3)
self.label_3.setMaximumSize(QtCore.QSize(420, 16777215))
self.label_3.setObjectName("label_3")
self.gridLayout_3.addWidget(self.label_3, 1, 0, 1, 1)
self.label_4 = QtWidgets.QLabel(self.gridLayoutWidget_3)
self.label_4.setMinimumSize(QtCore.QSize(129, 0))

```



```

self.label_4.setMaximumSize(QQtCore.QSize(129, 16777215))
self.label_4.setAlignment(QQtCore.Qt.AlignRight | QtCore.Qt.AlignTrailing |
QtCore.Qt.AlignVCenter)
self.label_4.setObjectName("label_4")
self.gridLayout_3.addWidget(self.label_4, 1, 1, 1, 1)
self.pushButton_17 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_17.setGeometry(QtCore.QRect(590, 80, 251, 91))
self.pushButton_17.setObjectName("pushButton_17")
self.label_5 = QtWidgets.QLabel(self.centralwidget)
self.label_5.setGeometry(QtCore.QRect(590, 225, 251, 131))
self.label_5.setText("")
self.label_5.setAlignment(QtCore.Qt.AlignCenter)
self.label_5.setWordWrap(True)
self.label_5.setObjectName("label_5")
MainWindow.setCentralWidget(self.centralwidget)

```

```

QtCore.QMetaObject.connectSlotsByName(MainWindow)
MainWindow.setTabOrder(self.pushButton_17, self.pushButton_18)
MainWindow.setTabOrder(self.pushButton_18, self.pushButton_19)
MainWindow.setTabOrder(self.pushButton_19, self.pushButton)
MainWindow.setTabOrder(self.pushButton, self.pushButton_2)
MainWindow.setTabOrder(self.pushButton_2, self.pushButton_3)
MainWindow.setTabOrder(self.pushButton_3, self.pushButton_4)
MainWindow.setTabOrder(self.pushButton_4, self.pushButton_5)
MainWindow.setTabOrder(self.pushButton_5, self.pushButton_6)
MainWindow.setTabOrder(self.pushButton_6, self.pushButton_7)
MainWindow.setTabOrder(self.pushButton_7, self.pushButton_8)
MainWindow.setTabOrder(self.pushButton_8, self.pushButton_9)
MainWindow.setTabOrder(self.pushButton_9, self.pushButton_10)
MainWindow.setTabOrder(self.pushButton_10, self.pushButton_11)
MainWindow.setTabOrder(self.pushButton_11, self.pushButton_12)
MainWindow.setTabOrder(self.pushButton_12, self.pushButton_13)
MainWindow.setTabOrder(self.pushButton_13, self.pushButton_14)
MainWindow.setTabOrder(self.pushButton_14, self.pushButton_15)
MainWindow.setTabOrder(self.pushButton_15, self.pushButton_16)

```

```

_translate = QtCore.QCoreApplication.translate
MainWindow.setWindowTitle(_translate("MainWindow", "Тра y 15"))
self.pushButton_4.setText(_translate("MainWindow", "4"))
self.pushButton_8.setText(_translate("MainWindow", "8"))
self.pushButton_7.setText(_translate("MainWindow", "7"))
self.pushButton_11.setText(_translate("MainWindow", "11"))
self.pushButton_14.setText(_translate("MainWindow", "14"))
self.pushButton_9.setText(_translate("MainWindow", "9"))

```

```

self.pushButton_3.setText(_translate("MainWindow", "3"))
self.pushButton.setText(_translate("MainWindow", "1"))
self.pushButton_15.setText(_translate("MainWindow", "15"))
self.pushButton_5.setText(_translate("MainWindow", "5"))
self.pushButton_13.setText(_translate("MainWindow", "13"))
self.pushButton_12.setText(_translate("MainWindow", "12"))
self.pushButton_2.setText(_translate("MainWindow", "2"))
self.pushButton_10.setText(_translate("MainWindow", "10"))
self.pushButton_6.setText(_translate("MainWindow", "6"))
self.pushButton_18.setText(_translate("MainWindow", "Наступний \n"
                                     "крок"))
self.pushButton_19.setText(_translate("MainWindow", "Автоматичне\n"
                                     "розв'язування"))
self.label_2.setText(_translate("MainWindow", "0"))
self.label.setText(_translate("MainWindow", "переміщень"))
self.label_3.setText(_translate("MainWindow", "час гри"))
self.label_4.setText(_translate("MainWindow", "00:00"))
self.pushButton_17.setText(_translate("MainWindow", "Почати нову\n"
                                     "гру"))

```

main.py

```

from MainController import MainController

```

```

def main():
    MainController()

```

```

if __name__ == '__main__':
    main()

```