# Опис алгоритмів

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

Таблиця 3.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
|  | Унікальний логін користівача |
| bool combinationFromFileCanBeAssembled | Ознака розв’язності комбінації, зчитаної з файлу |
| WayOfInput wayofinput | Перелічувальний тип, ознака походження комбінації(генерація/з файлу) |

## Загальний алгоритм

1. ПОЧАТОК
2. Зчитати ім’я користувача:
   1. ЯКЩО ім’я порожнє, вивести відповідне повідомлення та запросити його заново.
3. Запросити вибрати спосіб отримання комбінації(з файлу або згенерувати).
4. ЯКЩО вибрано спосіб отримання комбінації «з файлу», ТО зчитати з файлу input.txt комбінацію, ІНАКШЕ – згенерувати її.
5. ЯКЩО комбінація розв’язна, ТО запустити гру. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 7.
6. Гра запущена:
   1. ЯКЩО гравець натискає на костяшку, з якої можливий хід, ТО відбувається хід
      1. ІНАКШЕ ЯКЩО натиснуто кнопку підказки, то програма видає підказку щодо наступного ходу,
      2. ІНАКШЕ ЯКЩО натиснуто кнопку виходу з гри, ТО
         1. ЯКЩО комбінація зібрана, вивести повідомлення про перемогу. ІНАКШЕ вивести повідомлення про програш.
7. КІНЕЦЬ

## Алгоритм перевірки на зібраність комбінації (Field::isAssembled)

1. ПОЧАТОК
2. Створити зібрану матрицю ideal 6\*6 елементів типу int:
   1. Заповнити її значеннями -1.
   2. ЦИКЛ i проходу від 2-ї строки до передостанньої
   3. ЦИКЛ j проходу від 2-го стовпця до передостаннього
      1. Ideali, j = (i - 1) x 4 + j
   4. ЦИКЛ i проходу від 2-ї строки до передостанньої
   5. ЦИКЛ j проходу від 2-го стовпця до передостаннього
      1. ЯКЩО елемент з індексами i, j не дорівнює відповідному елементу у вхідній матриці, ТО повернути FALSE та перейти до кроку 3.
   6. За замовчуванням повернути TRUE.
3. КІНЕЦЬ

## Алгоритм підрахування евристики (Field::getHeuristic)

1. ПОЧАТОК
2. ЦИКЛ i проходу від 2-ї строки до передостанньої
3. ЦИКЛ j проходу від 2-го стовпця до передостаннього
4. ЦИКЛ i проходу від 2-ї строки до передостанньої
5. ЦИКЛ j проходу від 2-го стовпця до передостаннього
7. ЯКЩО
   1. ЯКЩО , ТО
8. ЯКЩО
   1. ЯКЩО , ТО
9. ЯКЩО
   1. ЯКЩО , ТО
10. КІНЕЦЬ

## Алгоритм встановлення можливості вирішення комбінації (Field::canBeAssembled)

1. ПОЧАТОК
2. ЦИКЛ i проходу від 2-ї строки до передостанньої
3. ЦИКЛ j проходу від 2-го стовпця до передостаннього
   1. ЯКЩО , ТО
4. ЦИКЛ i проходу від 2-ї строки до передостанньої
5. ЦИКЛ j проходу від 2-го стовпця до передостаннього
   1. ЯКЩО , ТО
6. ЦИКЛ i проходу від 1-го до передостаннього елемента snake
7. ЦИКЛ j проходу від i+1-го до останнього елемента snake
8. Повернути
9. КІНЕЦЬ

## Алгоритм змінення матриці гри(хід)(Field::changeMap)

1. ПОЧАТОК
2. ЯКЩО , ТО
   1. обміняти місцями та
   2. Повернути TRUE
   3. Перейти на крок 7
3. ЯКЩО , ТО
   1. обміняти місцями та
   2. Повернути TRUE
   3. Перейти на крок 7
4. ЯКЩО , ТО
   1. обміняти місцями та
   2. Повернути TRUE
   3. Перейти на крок 7
5. ЯКЩО , ТО
   1. обміняти місцями та
   2. Повернути TRUE
   3. Перейти на крок 7
6. Повернути FALSE
7. КІНЕЦЬ

## Алгоритм генерації комбінації (Game::generateCombination)

1. ПОЧАТОК
2. ЦИКЛ i проходу від 2-ї строки до передостанньої
3. ЦИКЛ j проходу від 2-го стовпця до передостаннього
4. ЦИКЛ 20 разів:
   1. РОБИТИ
   2. ПОКИ

* 1. Обміняти місцями та

1. КІНЕЦЬ

## Алгоритм видачі підказки (Game::solve)

1. ПОЧАТОК
2. ЯКЩО madeStepAfterHelp == false && helpCalls.size>0, ТО
   1. Повернути останній елемент helpCalls[]
   2. Перейти до кроку !!!!!!!!!!!
3. ІНАКШЕ:
   1. Покладемо максимальну глибину спуску(MAXDEPTH) рівною 17
   2. Множина хешей пройдених вершин
   3. Додамо start до q
   4. Додамо до allVertoxes start
   5. ПОКИ черга q не порожня:
      1. Запишемо у curr перший елемент черги.
      2. ЯКЩО and :
         1. ЯКЩО curr є зібраною
            1. Вийти з циклу
         2. ІНАКШЕ:
4. КІНЕЦЬ