**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИСиТ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Судоку

|  |  |  |
| --- | --- | --- |
| Студент гр. 9374 |  | Ламов Д.Е. |
| Преподаватель |  | Синев В.Е. |

Санкт-Петербург

2019

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Ламов Д.Е. | | |
| Группа 9374 | | |
| Тема работы: Судоку | | |
| Исходные данные:  Создать программу, которая могла бы:   * Решить предложенное ей судоку * Сгенерировать уровни разной сложности * Сохранять уровень судоку и решение пользователя | | |
| Содержание пояснительной записки:   * Содержание * Введение * Заключение * Список использованных источников | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 01.01.2020 | | |
| Дата сдачи реферата: 09.05.2020 | | |
| Дата защиты реферата: 09.05.2020 | | |
| Студент |  | Ламов Д.Е. |
| Преподаватель |  | Синев В.Е. |

**Аннотация**

Проект представляет из себя консольное приложение, написанное на языке программирования C++. При создании программы использовалась каскадная модель разработки. По итогу полученная программа может: решать судоку различной сложности, при чем если входные данные будут неверными (судоку будет иметь больше одного решения или не будет иметь решений вообще) программа сообщит об этом; генерировать для пользователя уровни различной сложности; проверять решение пользователя; реализовывать функцию заметок; сохранять уровни и продолжать игру в них.

**Оглавление**

[ВВЕДЕНИЕ 6](#_Toc40013100)

[1. АЛГОРИТМ РЕШЕНИЯ СУДОКУ 7](#_Toc40013101)

[2. ГЕНЕРАЦИЯ УРОВНЕЙ РАЗНОЙ СЛОЖНОСТИ 10](#_Toc40013102)

[3. РЕАЛИЗАЦИЯ АЛГОРИТМА ИГРЫ 15](#_Toc40013103)

[4. ИНТЕРФЕЙС 16](#_Toc40013104)

[ЗАКЛЮЧЕНИЕ 26](#_Toc40013105)

### ВВЕДЕНИЕ

Цель: обобщение знаний и практических навыков по программированию, полученных за первый курс на примере генерации и решения судоку.

Основные правила судоку:

* Уровень имеет единственное решение
* Числа в рядах и столбцах не должны повторяться
* Числа в малых квадратах не должны повторяться

Основные задачи, поставленные перед программой:

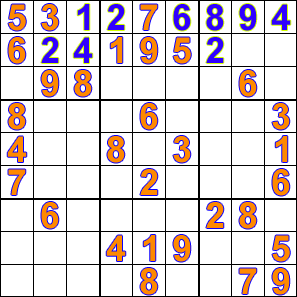
* Решать введенные пользователем судоку – методом перебора всех вариантов
* Генерировать уровни судоку разной сложности – путем элементарных преобразований «базовой» матрицы и удаления из нее части элементов
* Сохранять сгенерированные уровни и решения пользователя: в длительной памяти компьютера создается файл с матрицей решения и матрицей уровня
* Создавать заметки при решении судоку: в верхнем правом углу клетки пользователь может указать число, которое предположительно является решением
* Проверять решение пользователя: программа решает судоку и сравнивает единственно верное решение для уровня с решением пользователя.

### 1. АЛГОРИТМ РЕШЕНИЯ СУДОКУ

Программа решает судоку путем перебора всех возможных вариантов:

1. Находит пустую клетку.
2. Выбирает одно из чисел, которое удовлетворяет правилам игры и ставит его на поле. Если такое число нашлось возвращается к пункту 1, иначе переходит к пункту 3.
3. Если поле не заполнено полностью и нет чисел, которые можно поставить в пустую клетку и при этом не нарушить правила игры, программа возвращается к предыдущему числу, чтобы заменить его.
4. В результате мы получаем полностью решенное судоку.

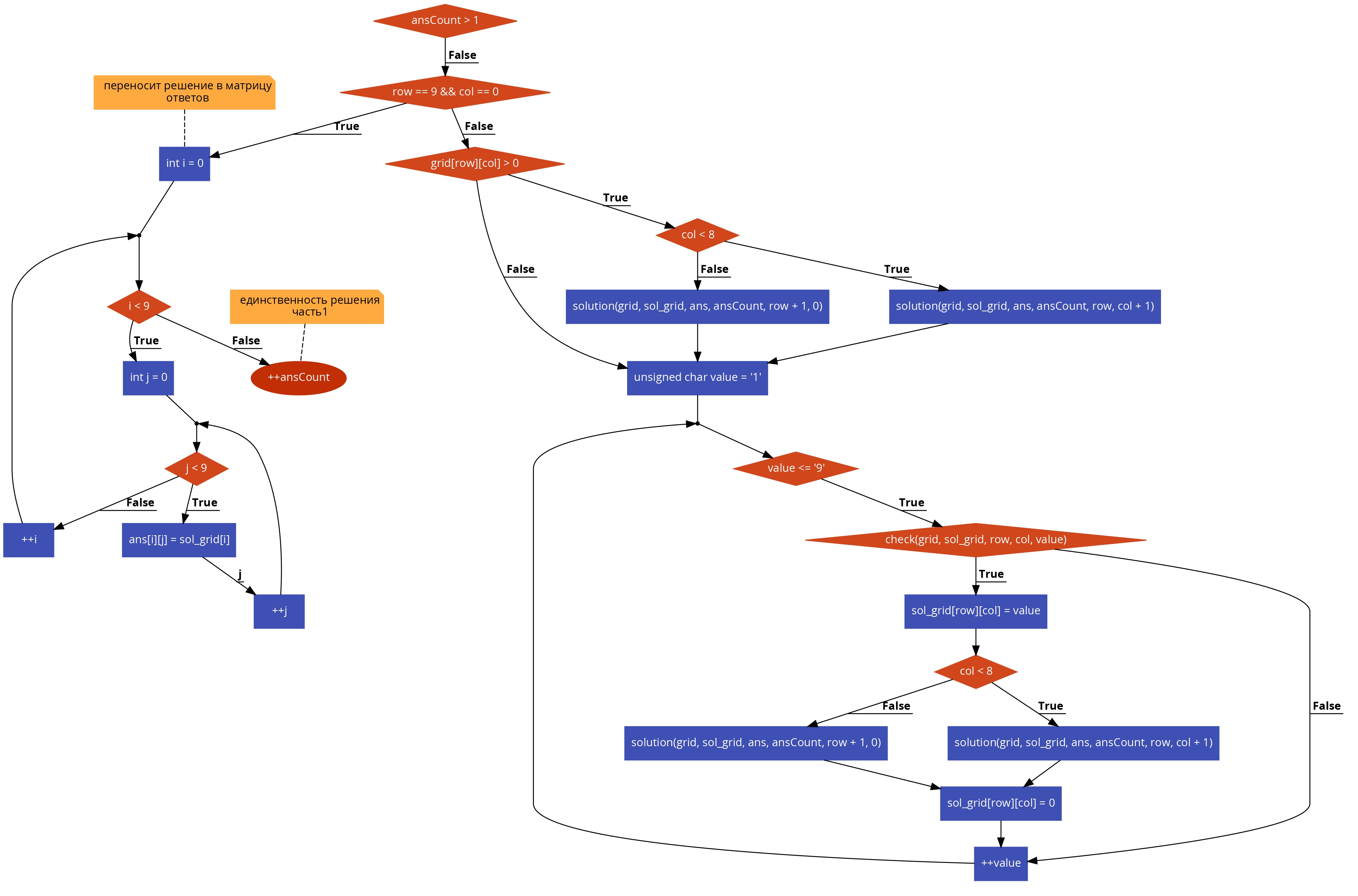
Работа алгоритма выглядит так:



(к сожалению, Word не поддерживает анимацию gif. Оригинал файла

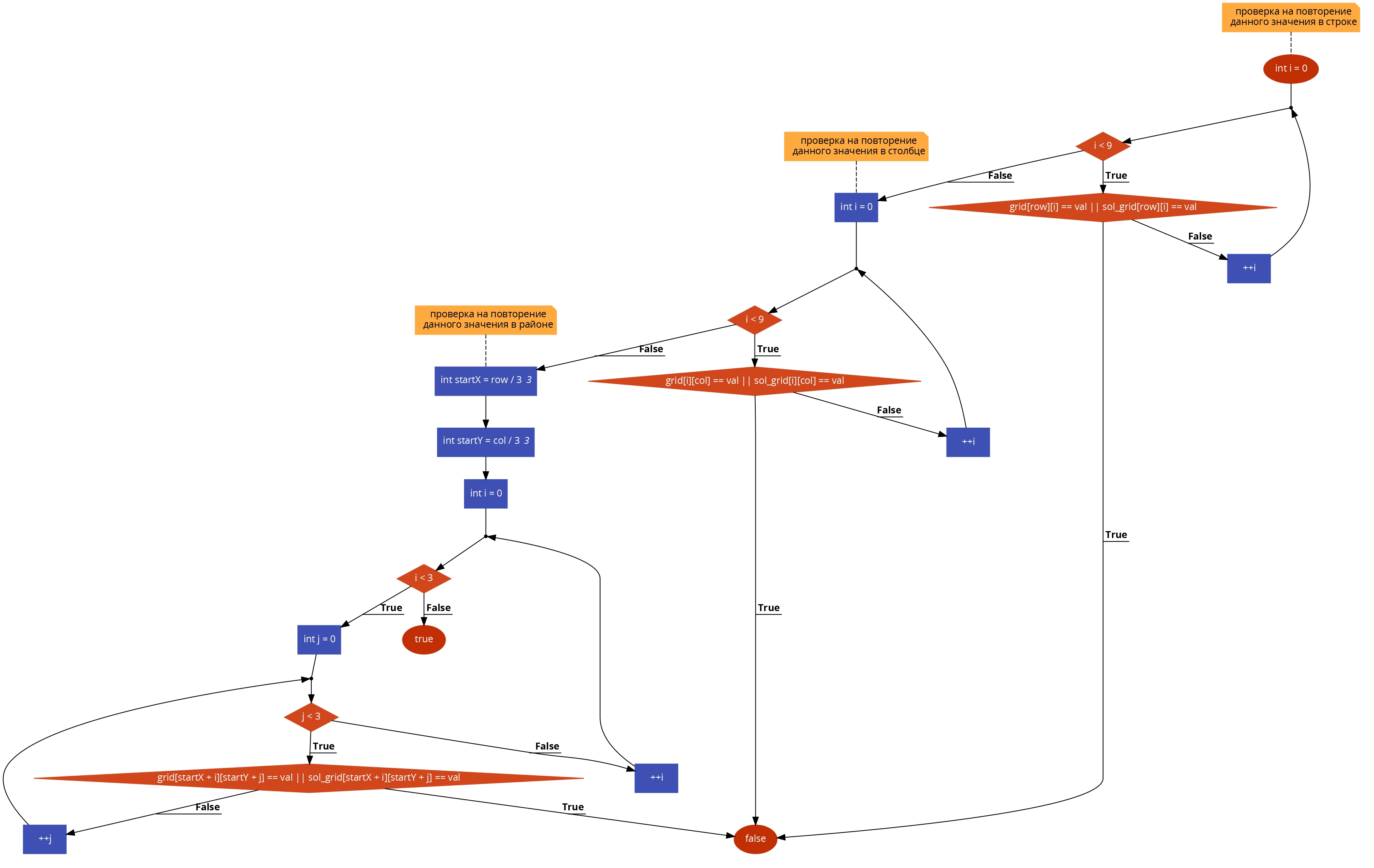
находится в папке Sudoku и имеет название Алгоритм работы.gif)

Блок-схема кода, решающего судоку:



Код данного блока находится в приложении.

В данном фрагменте задействована функция, проверяющая можно ли поставить на данное место данную цифру. Алгоритм ее работы основан на проверке нахождения данной цифры в том же столбце, или строке, или квадрате. Блок схема данной части кода:



Код блока, отвечающего за проверку, находится в приложении.

### 2. ГЕНЕРАЦИЯ УРОВНЕЙ РАЗНОЙ СЛОЖНОСТИ

За основу берется «базовая» матрица – полностью заполненная матрица, удовлетворяющая всем правилам судоку и выглядящая следующим образом:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Далее при помощи элементарных преобразований мы «перемешиваем» базовую матрицу. Главная особенность элементарных преобразований – они гарантируют, что после их применения матрица будет удовлетворять всем правилам судоку. К элементарным преобразованиям относятся:

* Транспонирование матрицы – элементы отражаются относительно главной диагонали:

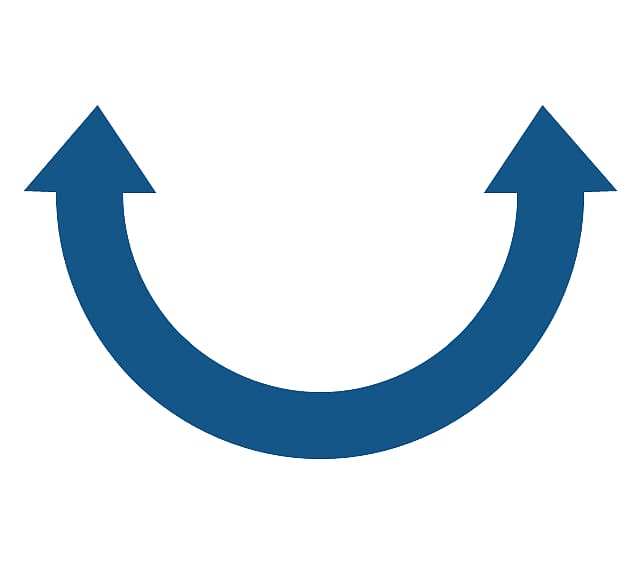
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 7 | 2 | 5 | 8 | 3 | 6 | 9 |
| 2 | 5 | 8 | 3 | 6 | 9 | 4 | 7 | 1 |
| 3 | 6 | 9 | 4 | 7 | 1 | 5 | 8 | 2 |
| 4 | 7 | 1 | 5 | 8 | 2 | 6 | 9 | 3 |
| 5 | 8 | 2 | 6 | 9 | 3 | 7 | 1 | 4 |
| 6 | 9 | 3 | 7 | 1 | 4 | 8 | 2 | 5 |
| 7 | 1 | 4 | 8 | 2 | 5 | 9 | 3 | 6 |
| 8 | 2 | 5 | 9 | 3 | 6 | 1 | 4 | 7 |
| 9 | 3 | 6 | 1 | 4 | 7 | 2 | 5 | 8 |

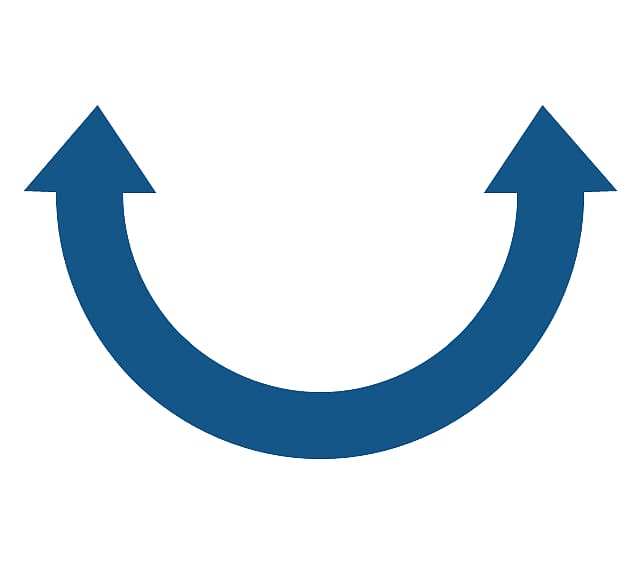
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

* Обмен строк в пределах одного района

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

****

****

* Обмен столбцов в пределах одного района

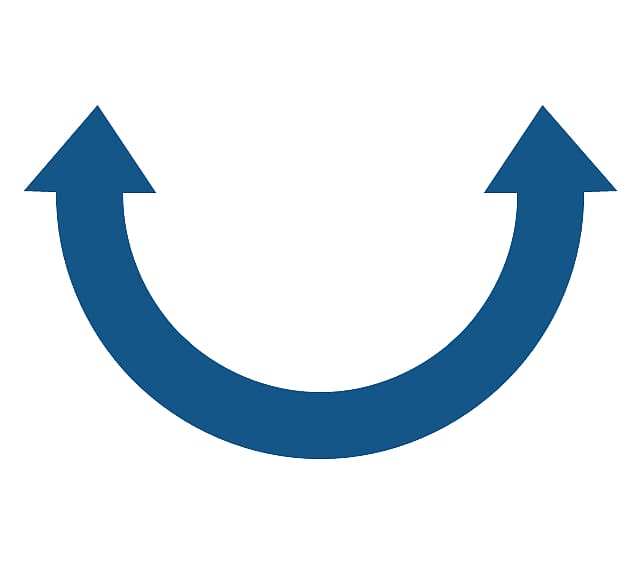
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

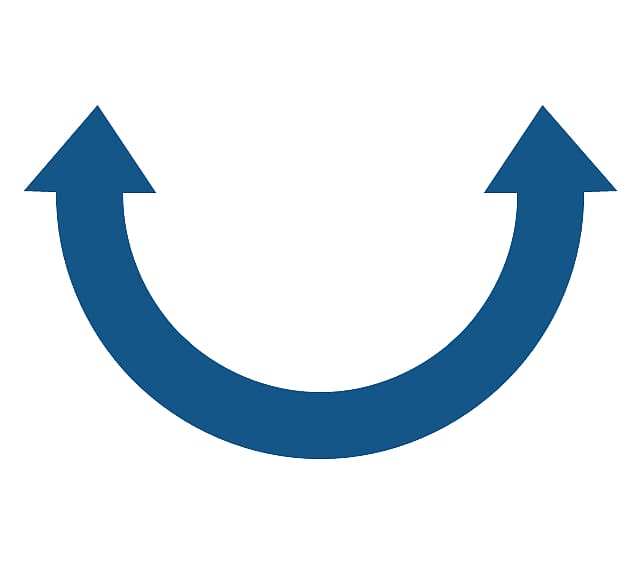
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 9 | 8 | 7 |
| 4 | 5 | 6 | 7 | 8 | 9 | 3 | 2 | 1 |
| 7 | 8 | 9 | 1 | 2 | 3 | 6 | 5 | 4 |
| 2 | 3 | 4 | 5 | 6 | 7 | 1 | 9 | 8 |
| 5 | 6 | 7 | 8 | 9 | 1 | 4 | 3 | 2 |
| 8 | 9 | 1 | 2 | 3 | 4 | 7 | 6 | 5 |
| 3 | 4 | 5 | 6 | 7 | 8 | 2 | 1 | 9 |
| 6 | 7 | 8 | 9 | 1 | 2 | 5 | 4 | 3 |
| 9 | 1 | 2 | 3 | 4 | 5 | 8 | 7 | 6 |

* Обмен районов по горизонтали

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

****

****

* Обмен районов по вертикали

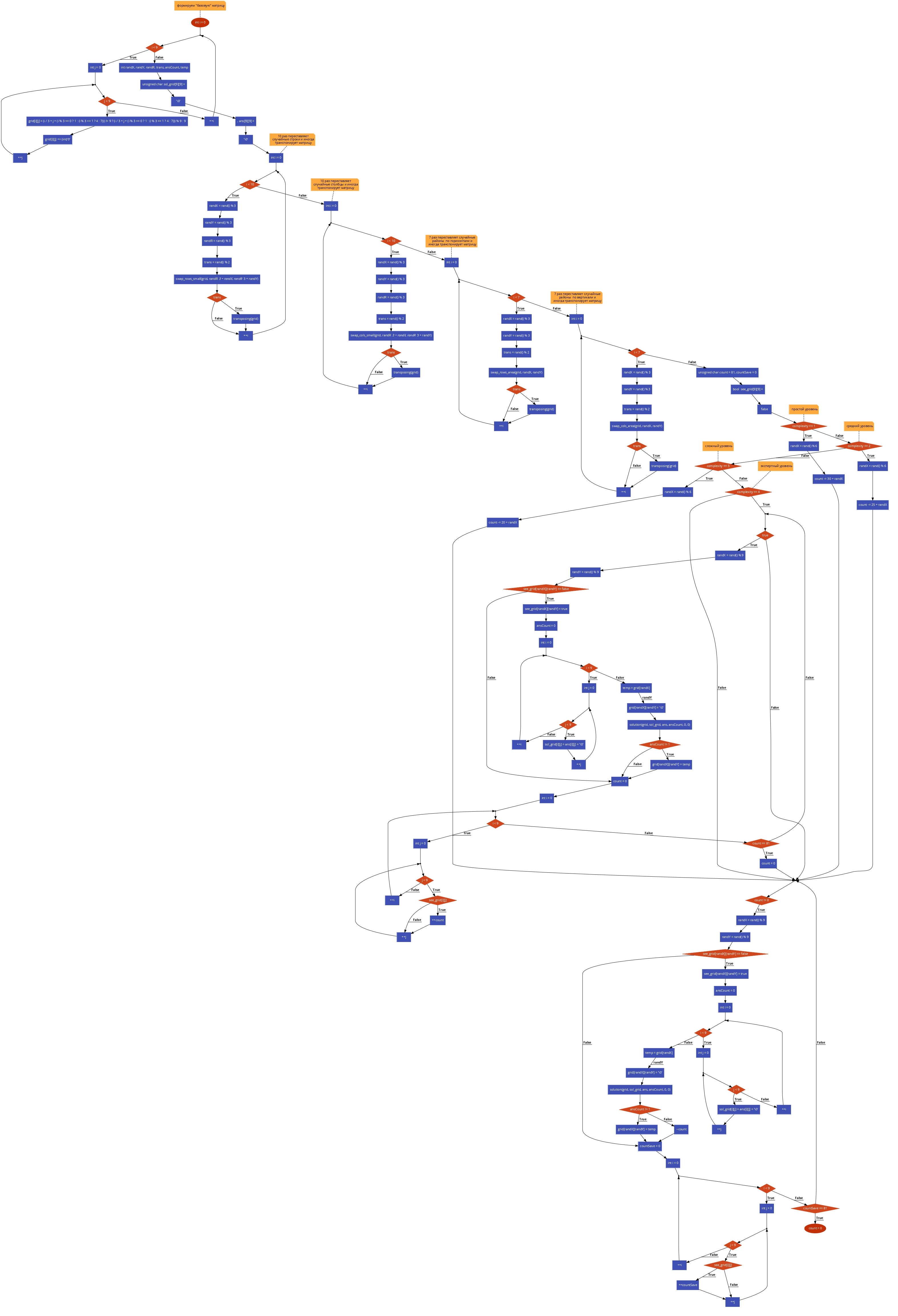
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 8 | 9 | 4 | 5 | 6 |
| 4 | 5 | 6 | 1 | 2 | 3 | 7 | 8 | 9 |
| 7 | 8 | 9 | 4 | 5 | 6 | 1 | 2 | 3 |
| 2 | 3 | 4 | 8 | 9 | 1 | 5 | 6 | 7 |
| 5 | 6 | 7 | 2 | 3 | 4 | 8 | 9 | 1 |
| 8 | 9 | 1 | 5 | 6 | 7 | 2 | 3 | 4 |
| 3 | 4 | 5 | 9 | 1 | 2 | 6 | 7 | 8 |
| 6 | 7 | 8 | 3 | 4 | 5 | 9 | 1 | 2 |
| 9 | 1 | 2 | 6 | 7 | 8 | 3 | 4 | 5 |

При помощи рандома применяются данные преобразования к случайным столбца, строкам и районам, при этом иногда происходит транспонирование матрицы. Так «базовая» матрица полностью перемешивается.

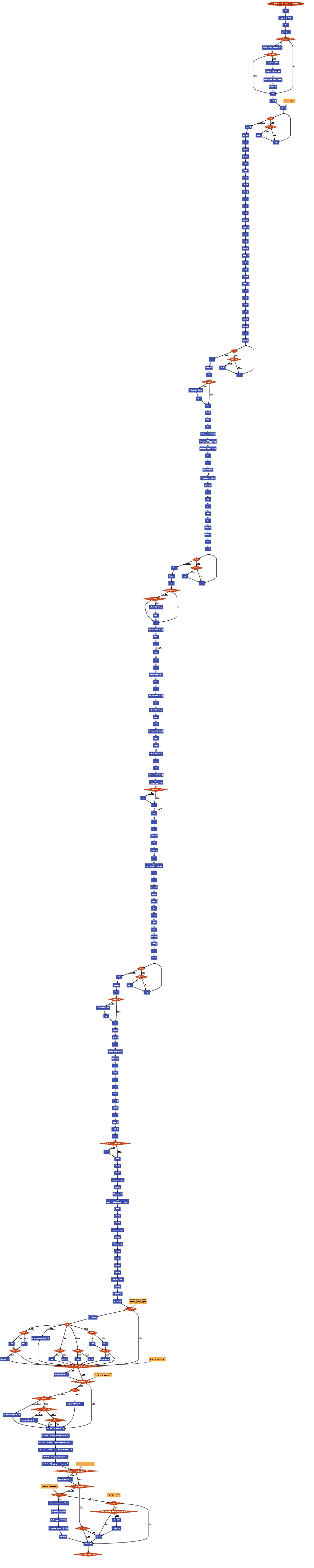
Далее в цикле случайным образом выбирается клетка. Если после ее удаления судоку имеет единственное решение, то выбирается следующая клетка, а если судоку стало иметь множество решений, в клетку возвращается ее старое значение. Так продолжается до тех пор, пока на поле не останется необходимое количество клеток (для легкого уровня это ~30, для среднего ~25, для сложного ~20, а в экспертном уровне сложности при удалении хоть одной клетки, судоку будет иметь бесконечное количество решений, то есть гарантируется, что на поле находится минимальное количество элементов для получения однозначного решения судоку).

Блок схема данного алгоритма:

Код данного блока находится в приложении.

### 3. РЕАЛИЗАЦИЯ АЛГОРИТМА ИГРЫ

Перемещения по полю, добавление на поле символов, проверка и сохранение уровней реализованы довольно просто. Блок схема поможет понять алгоритм работы данного блока (код находится в приложении).



### 4. ИНТЕРФЕЙС

При запуске программа автоматически откроется на весь экран и однократно пользователю будет показана инструкция.

Рисунок 4.

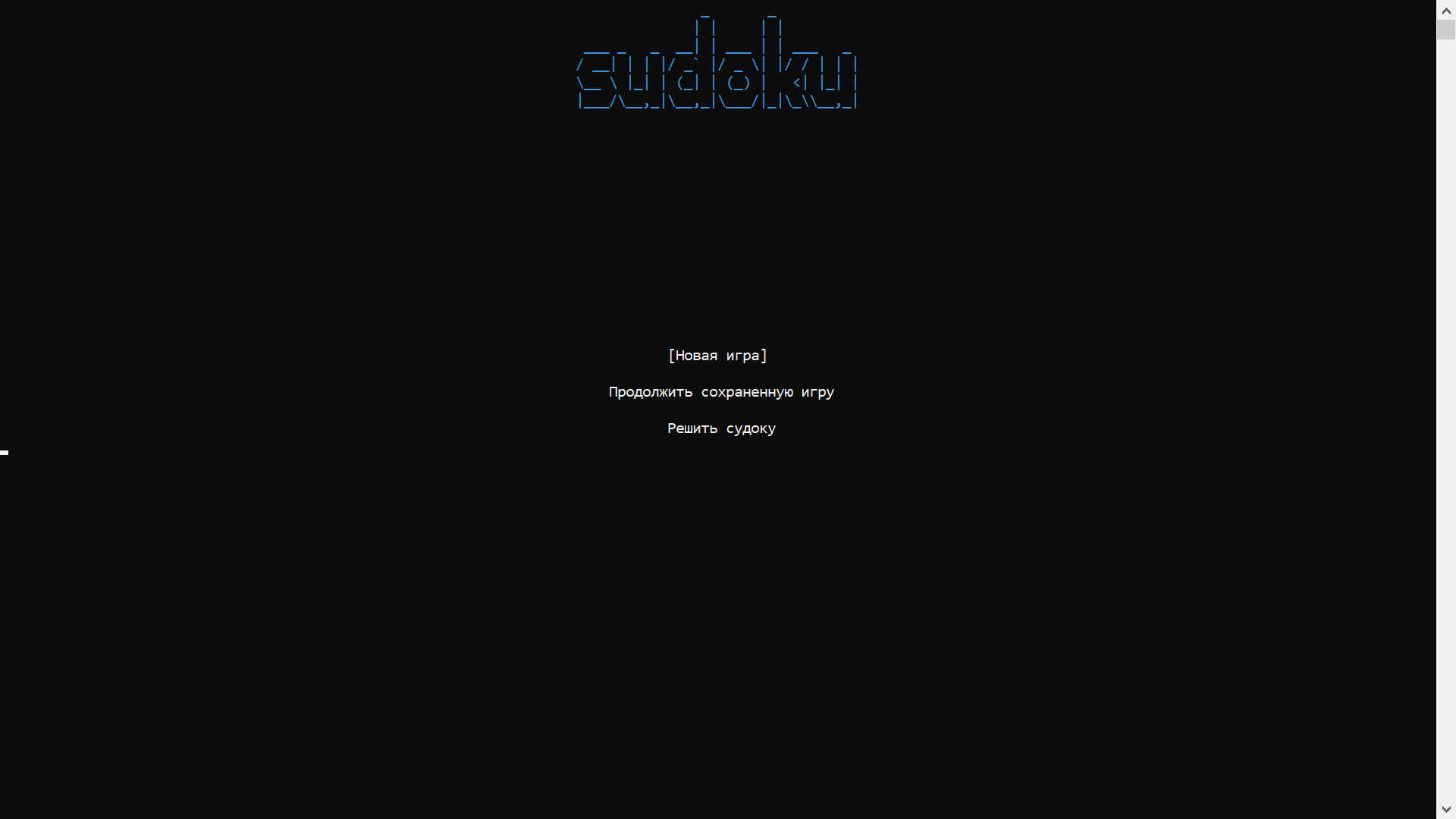
По нажатию клавиши Enter пользователь перейдет к основному меню.

Рисунок 4.

Для перемещения по пунктам меню нужно воспользоваться стрелками, для выбора пункта меню нажать Enter, для завершения работы программы нажать Esc.

При выборе первого пункта меню пользователю будет предложено выбрать уровень сложности. Для перемещения в данном меню также будет необходимо воспользоваться стрелками, для выбора пункта меню нажать Enter, для возврата в основное меню нажать Backspace.

Рисунок 4.

После выбора пункта меню пользователю однократно будет показана инструкция для самой игры. 

Рисунок 4.

Сам игровой интерфейс выглядит следующим образом. Синим отмечены клетки, которые нельзя изменять и с опорой на которые мы заполняем свободные клетки. В верхнем правом углу в некоторых свободных клетках находятся пользовательские заметки.

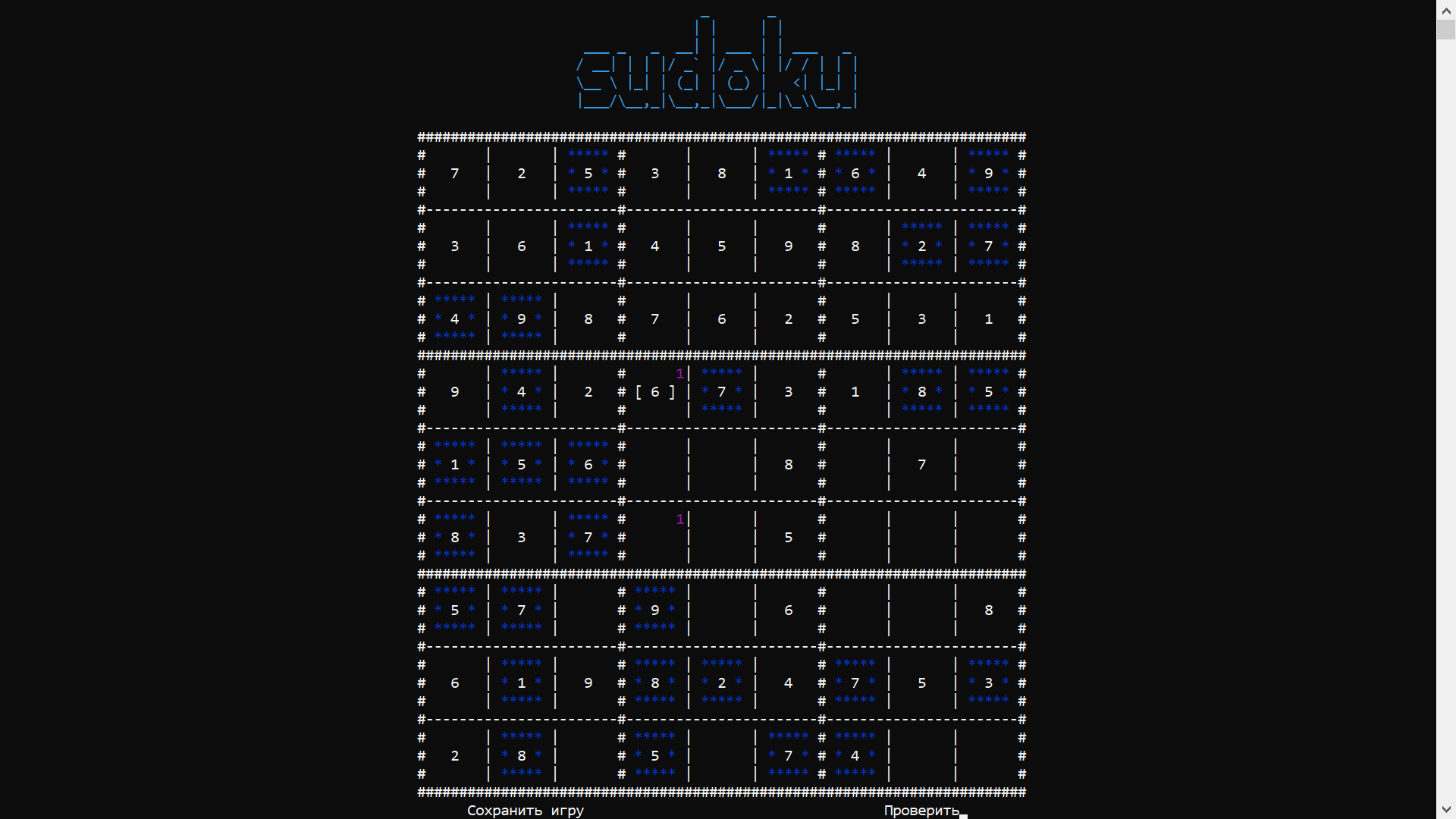


Рисунок 4.

При выборе пункта «проверить» на 3 секунды красным цветом будут подсвечены те клетки, в которых находятся неверные значения.

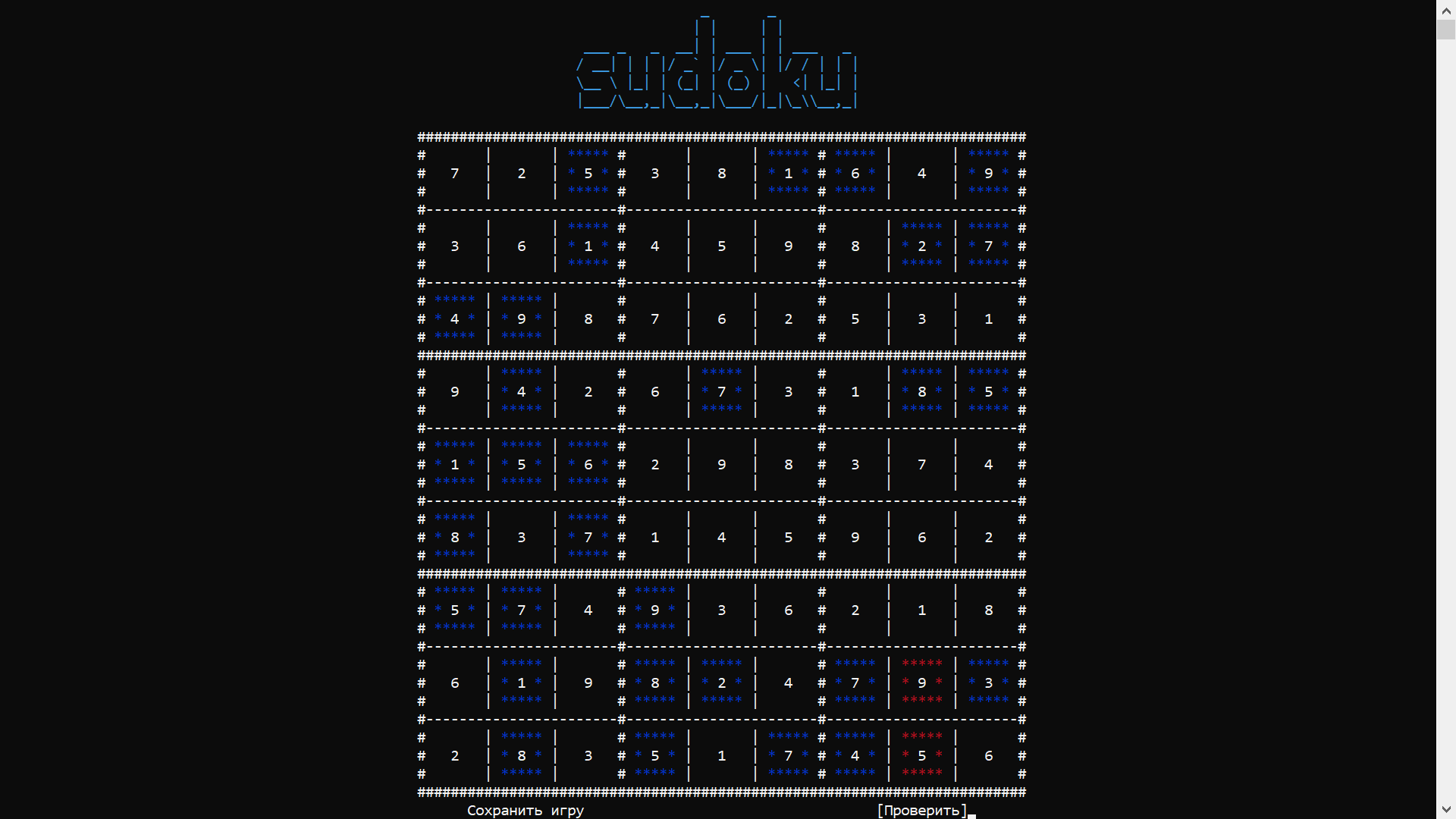


Рисунок 4.

Когда все поле будет полностью заполнено и будет удовлетворять всем правилам судоку при нажатии на кнопку проверить на 3 секунды будет выведено уведомление о прохождении уровня, а после пользователь окажется в главном меню.

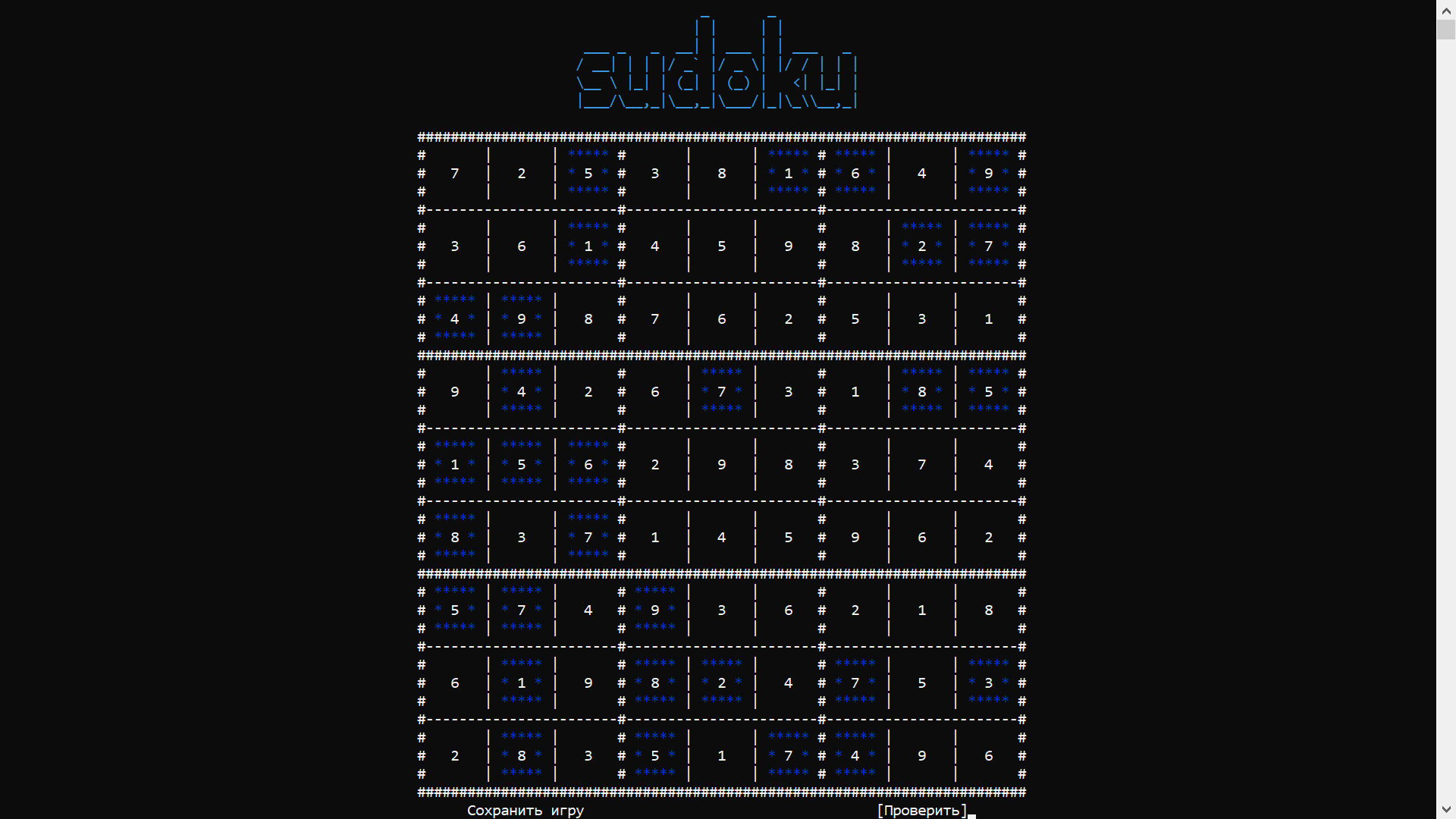


Рисунок 4.



Рисунок 4.

При выборе пункта «сохранить игру» игра будет сохранена, и пользователь окажется в главном меню.

При выборе второго пункта основного меню пользователь сможет продолжить решать уже сохраненный уровень.

При выборе третьего пункта меню, отвечающего за решение судоку полученного от пользователя, откроется подменю, в котором от пользователя потребуется выбрать необходимое ему действие. Для возврата к основному меню пользователю будет необходимо нажать клавишу Backspace.

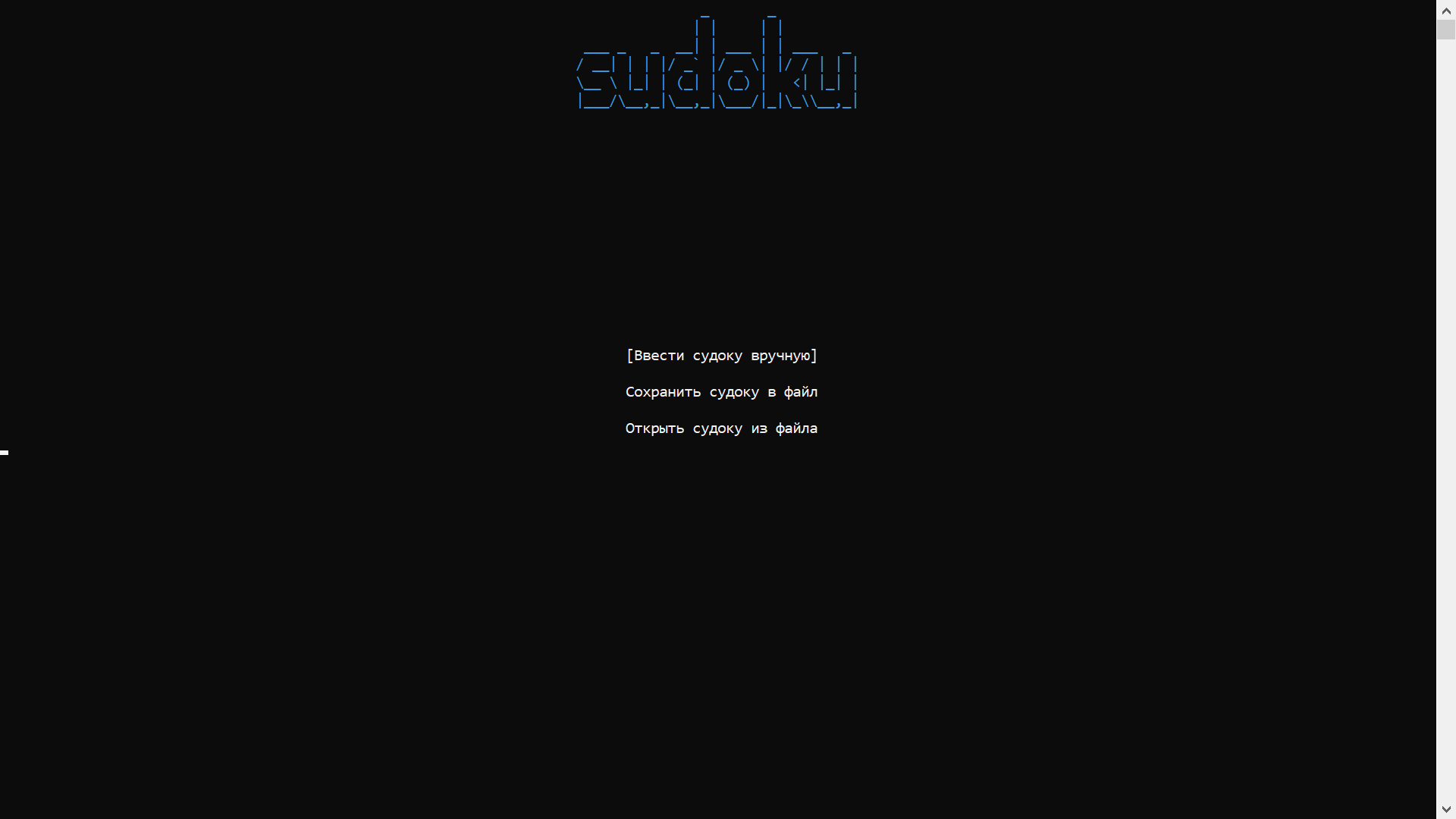
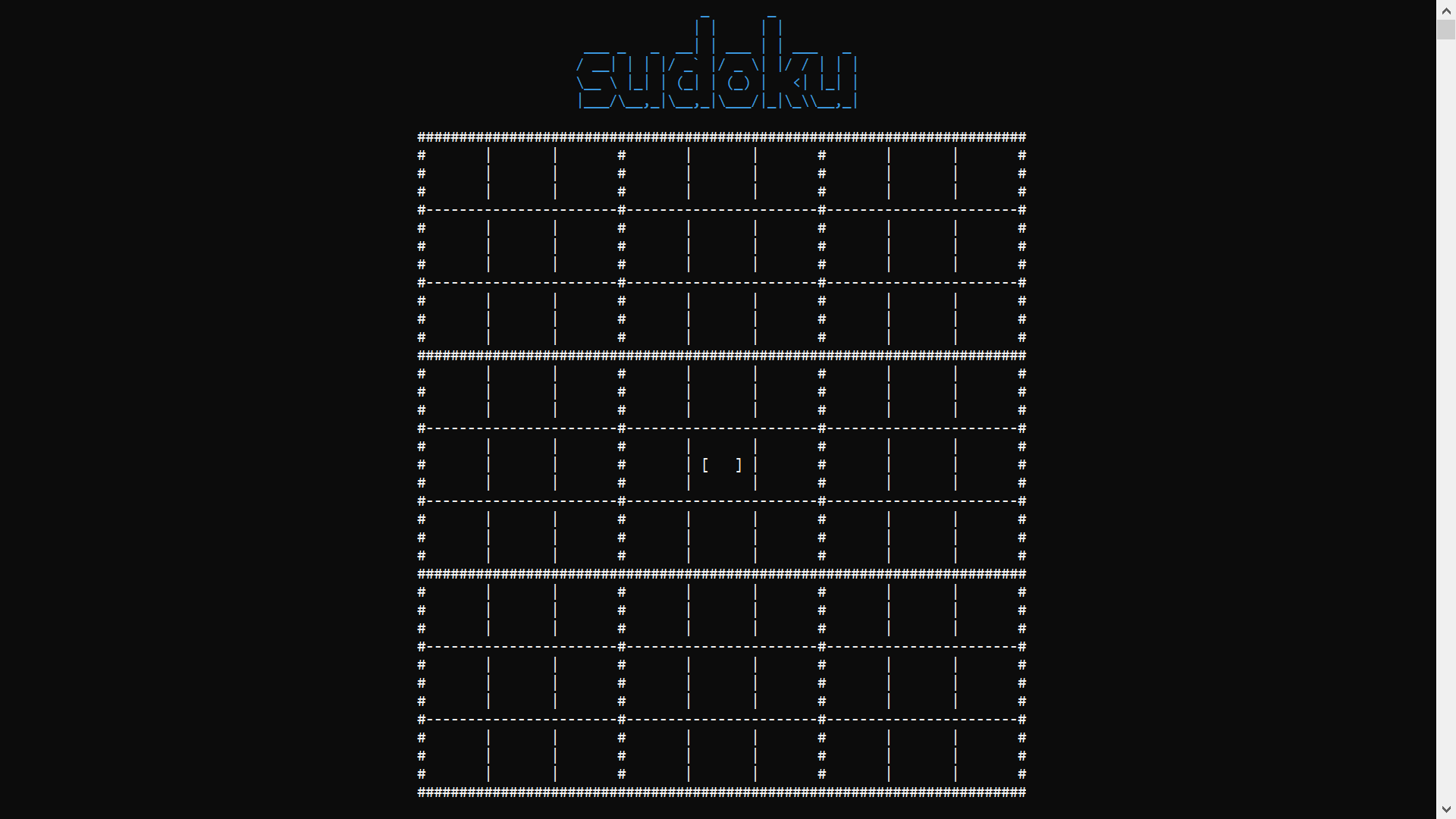
При выборе пункта ввести судоку вручную пользователь увидит следующее окно.

Рисунок 4. 10

Рисунок 4. 9

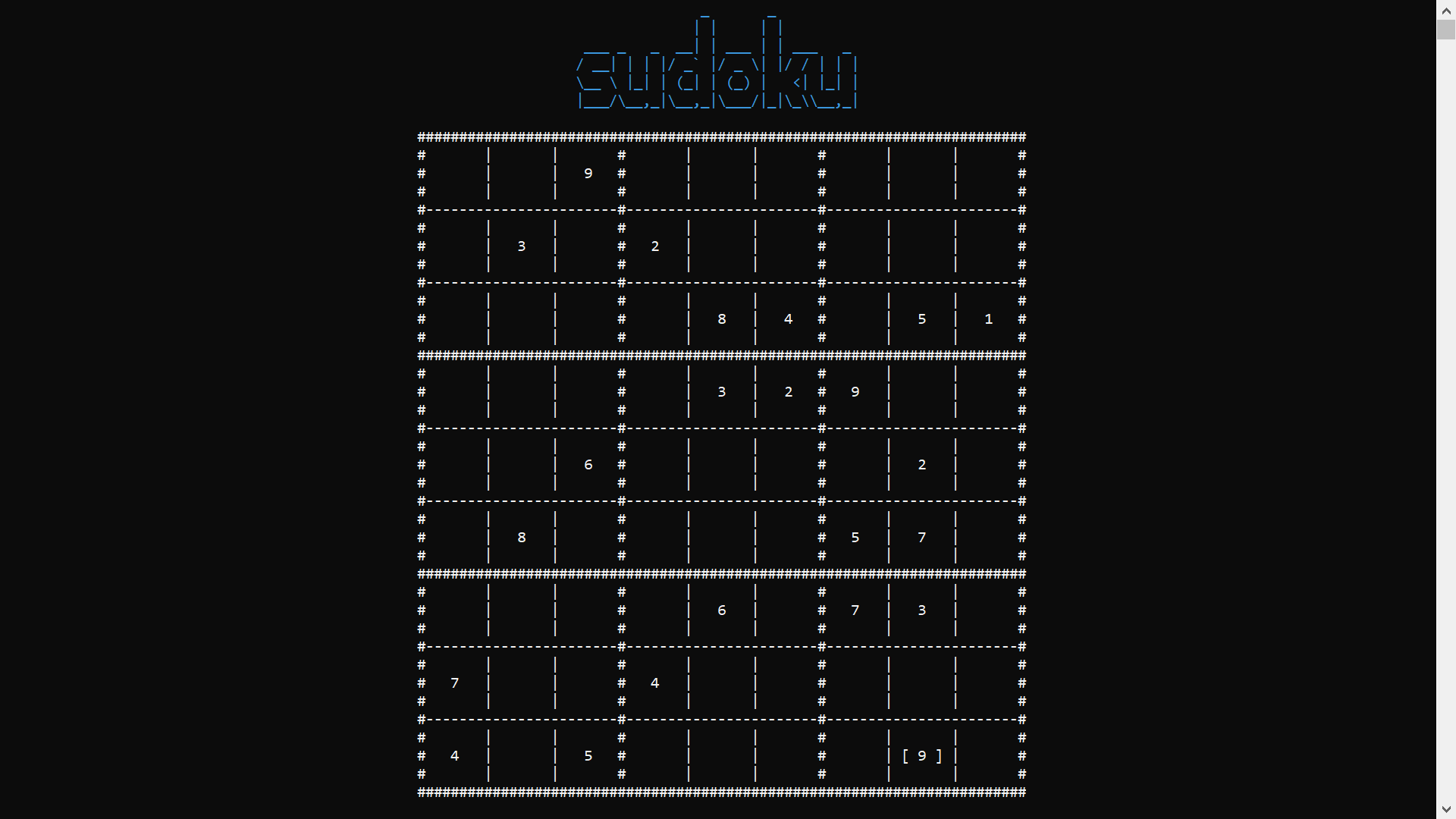
Далее пользователю необходимо ввести судоку, которое ему нужно решить, если пользователь передумает вводить судоку, то ему необходимо нажать клавишу Esc – она вернет его в подменю. Заполнение поля осуществляется с помощью стрелок, цифр и клавиши Backspace для удаления неверных цифр. Вот пример заполненного поля:

Рисунок 4. 11

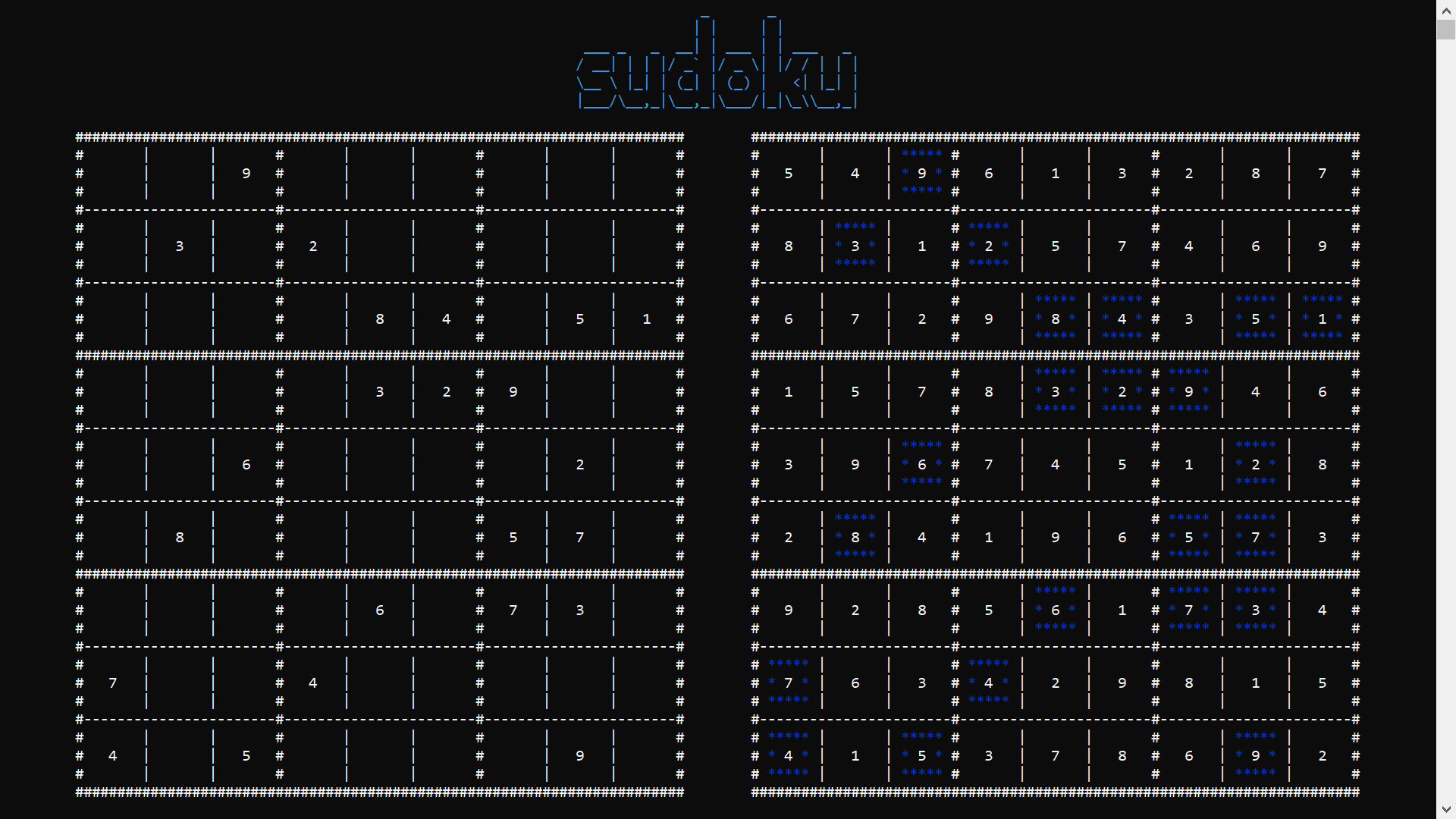
 После завершения ввода судоку пользователю необходимо нажать клавишу Enter, и программа моментально выведет решение данного судоку.

Рисунок 4. 12

Если судоку, введенное пользователем будет иметь более одного решения, следовательно не будет являться уровнем в судоку, то программа выведет следующее сообщение об ошибке:

Рисунок 4. 13

После нажатия любой клавиши пользователь окажется в подменю.

Рисунок 4. 14

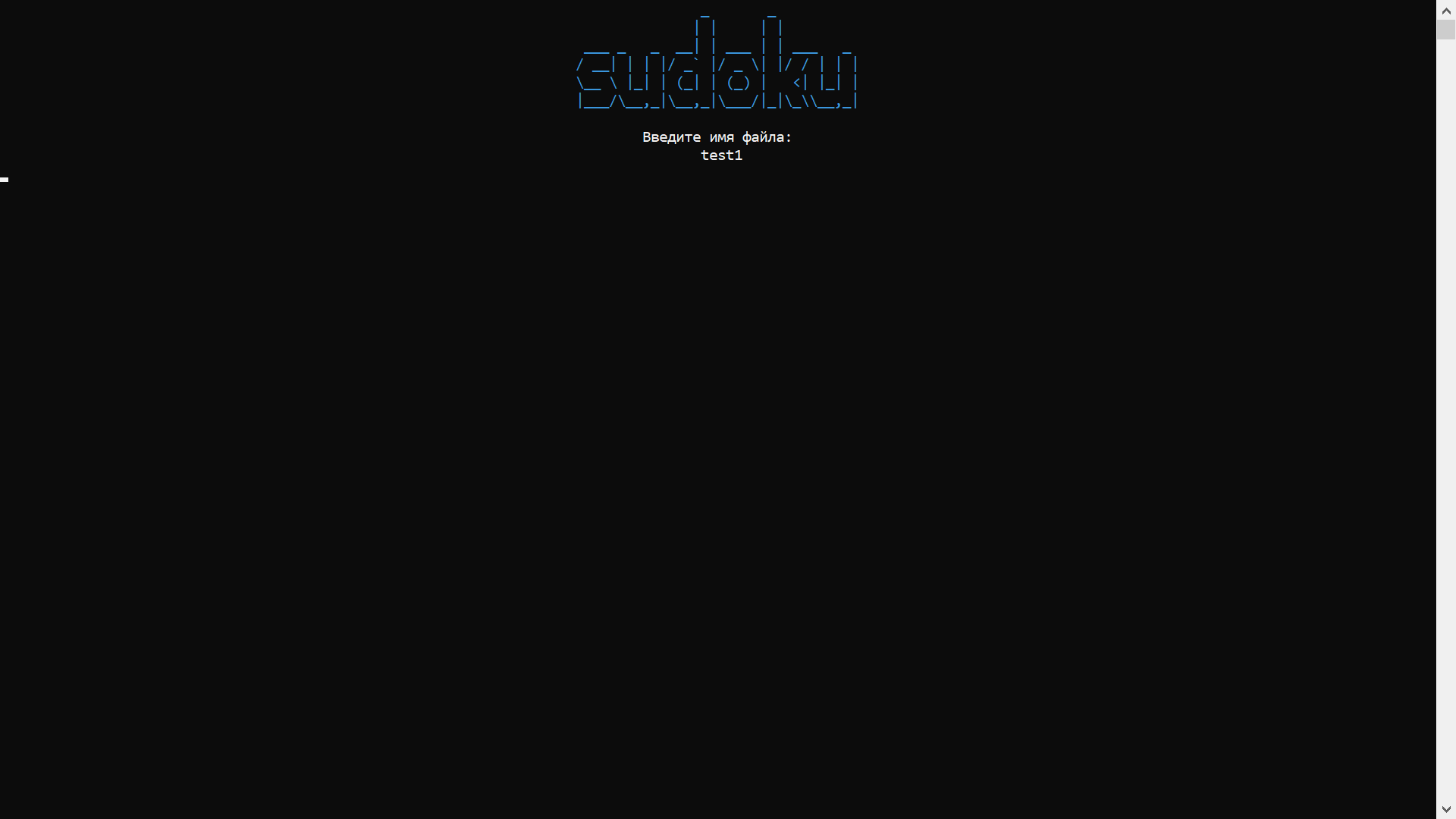
При выборе пункта подменю «сохранить судоку в файл» с помощью того же интерфейса и тех же команд управления пользователь заполняет поле. Далее пользователю необходимо нажать клавишу Enter, и программа попросит ввести имя файла.

Рисунок 4. 15

После ввода названия файла и нажатия клавиши Enter пользователь окажется в подменю.

При выборе пункта «открыть из файла» пользователю будет нужно указать название файла, который он хочет открыть. После будет показано созданное поле и его решение.

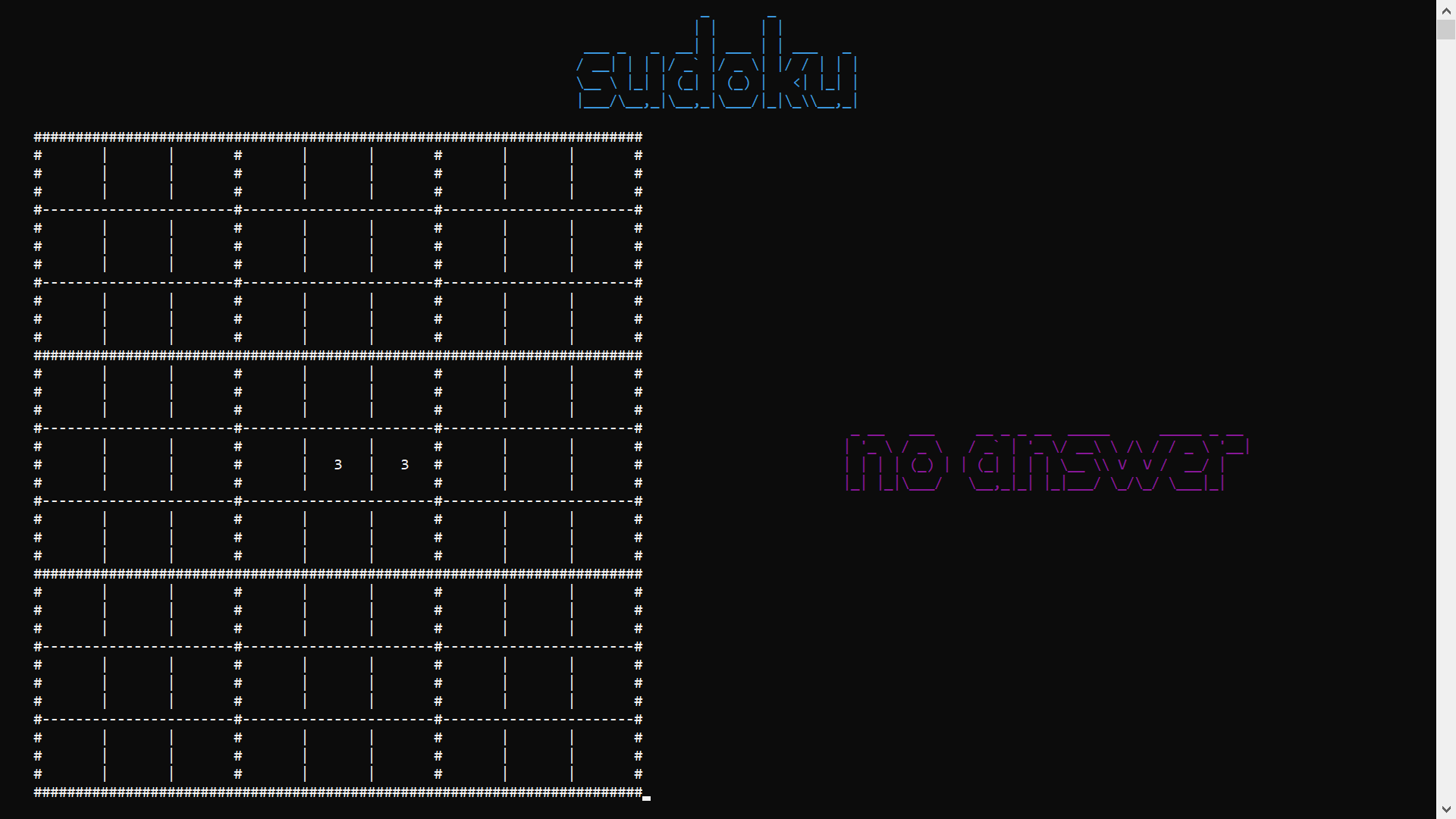
Если данное поле имеет множество решений или не имеет решения вообще, программа сообщит об этом.

Рисунок 4. 17

Рисунок 4. 16

Для возврата в подменю необходимо нажать Enter.

### ЗАКЛЮЧЕНИЕ

Была создана многофункциональная программа для решения судоку. В ходе работы были использованы списки и сохранения. Проект обобщает знания по программированию, полученные за первый курс. Список использованных источников

<https://habr.com/ru/post/192102/>

**приложение**

Код алгоритма, решающего судоку:

void solution(unsigned char grid[9][9], unsigned char sol\_grid[9][9], unsigned char ans[9][9], int& ansCount, const int& row, const int& col) {

if (ansCount > 1) return;

if (row == 9 && col == 0) {

//переносит решение в матрицу ответов

for (int i = 0; i < 9; ++i)

for (int j = 0; j < 9; ++j)

ans[i][j] = sol\_grid[i][j];

//единственность решения часть1

++ansCount;

return;

}

if (grid[row][col] > 0)

if (col < 8) solution(grid, sol\_grid, ans, ansCount, row, col + 1);

else solution(grid, sol\_grid, ans, ansCount, row + 1, 0);

for (unsigned char value = '1'; value <= '9'; ++value)

if (check(grid, sol\_grid, row, col, value)) {

sol\_grid[row][col] = value;

if (col < 8) solution(grid, sol\_grid, ans, ansCount, row, col + 1);

else solution(grid, sol\_grid, ans, ansCount, row + 1, 0);

sol\_grid[row][col] = 0;

}

}

Код алгоритма, проверяющего может ли данная цифра стоять в данной клетке:

bool check(unsigned char grid[9][9], unsigned char sol\_grid[9][9], const int& row, const int& col, const unsigned char& val) {

//проверка на повторение данного значения в строке

for (int i = 0; i < 9; ++i)

if (grid[row][i] == val || sol\_grid[row][i] == val) return false;

//проверка на повторение данного значения в столбце

for (int i = 0; i < 9; ++i)

if (grid[i][col] == val || sol\_grid[i][col] == val) return false;

//проверка на повторение данного значения в районе

int startX = row / 3 \* 3;

int startY = col / 3 \* 3;

for (int i = 0; i < 3; ++i)

for (int j = 0; j < 3; ++j)

if (grid[startX + i][startY + j] == val || sol\_grid[startX + i][startY + j] == val) return false;

return true;

}

Код алгоритма, генерирующего уровень:

void Generation(unsigned char grid[9][9], const unsigned char& complexity) {

//формируем "базовую" матрицу

for (int i = 0; i < 9; ++i)

for (int j = 0; j < 9; ++j) {

grid[i][j] = (i / 3 + j + (i % 3 == 0 ? 1 : (i % 3 == 1 ? 4 : 7))) != 9 ? (i / 3 + j + (i % 3 == 0 ? 1 : (i % 3 == 1 ? 4 : 7))) % 9 : 9;

grid[i][j] += (int)'0';

}

int randX, randY, randR, trans, ansCount, temp;

unsigned char sol\_grid[9][9] = { '\0' }, ans[9][9] = { '\0' };

//10 раз переставляет случайные строки и иногда транспонирует матрицу

for (int i = 0; i < 10; ++i) {

randX = rand() % 3;

randY = rand() % 3;

randR = rand() % 3;

trans = rand() % 2;

swap\_rows\_small(grid, randR \* 3 + randX, randR \* 3 + randY);

if (trans) transposing(grid);

}

//10 раз переставляет случайные столбцы и иногда транспонирует матрицу

for (int i = 0; i < 10; ++i) {

randX = rand() % 3;

randY = rand() % 3;

randR = rand() % 3;

trans = rand() % 2;

swap\_cols\_small(grid, randR \* 3 + randX, randR \* 3 + randY);

if (trans) transposing(grid);

}

//7 раз переставляет случайные районы по горизонтали и иногда транспонирует матрицу

for (int i = 0; i < 7; ++i) {

randX = rand() % 3;

randY = rand() % 3;

trans = rand() % 2;

swap\_rows\_area(grid, randX, randY);

if (trans) transposing(grid);

}

//7 раз переставляет случайные районы по вертикали и иногда транспонирует матрицу

for (int i = 0; i < 7; ++i) {

randX = rand() % 3;

randY = rand() % 3;

trans = rand() % 2;

swap\_cols\_area(grid, randX, randY);

if (trans) transposing(grid);

}

unsigned char count = 81, countSave = 0;

bool see\_grid[9][9] = { false };

//простой уровень

if (complexity == 1) {

randX = rand() % 6;

count -= 30 + randX;

}

else

//средний уровень

if (complexity == 2) {

randX = rand() % 6;

count -= 25 + randX;

}

else

//сложный уровень

if (complexity == 3) {

randX = rand() % 6;

count -= 20 + randX;

}

else

//экспертный уровень

if (complexity == 4)

while (true) {

randX = rand() % 9;

randY = rand() % 9;

if (see\_grid[randX][randY] == false) {

see\_grid[randX][randY] = true;

ansCount = 0;

for (int i = 0; i < 9; ++i)

for (int j = 0; j < 9; ++j)

sol\_grid[i][j] = ans[i][j] = '\0';

temp = grid[randX][randY];

grid[randX][randY] = '\0';

solution(grid, sol\_grid, ans, ansCount, 0, 0);

if (ansCount != 1) grid[randX][randY] = temp;

}

count = 0;

for (int i = 0; i < 9; ++i)

for (int j = 0; j < 9; ++j)

if (see\_grid[i][j]) ++count;

if (count == 81) {

count = 0;

break;

}

}

while (count != 0) {

randX = rand() % 9;

randY = rand() % 9;

if (see\_grid[randX][randY] == false) {

see\_grid[randX][randY] = true;

ansCount = 0;

for (int i = 0; i < 9; ++i)

for (int j = 0; j < 9; ++j)

sol\_grid[i][j] = ans[i][j] = '\0';

temp = grid[randX][randY];

grid[randX][randY] = '\0';

solution(grid, sol\_grid, ans, ansCount, 0, 0);

if (ansCount != 1) grid[randX][randY] = temp;

else --count;

}

countSave = 0;

for (int i = 0; i < 9; ++i)

for (int j = 0; j < 9; ++j)

if (see\_grid[i][j]) ++countSave;

if (countSave == 81) {

count = 0;

break;

}

}

}

Код «игры»:

void Game(unsigned char grid[9][9], const unsigned char& modification) {

unsigned char row = 4, col = 4, us\_grid[9][9] = { '\0' }, us\_grid\_note[9][9] = { '\0' }, podType = 1;

if (modification == 2) {

FILE\* file = fopen("SavedLvl", "rb");

if (file != NULL)

{

fread(grid, 9, 9, file);

fread(us\_grid, 9, 9, file);

fread(us\_grid\_note, 9, 9, file);

fclose(file);

}

}

do {

sudoku();

//вывод матрицы

for (int i = 0; i < 9; ++i) {

if (i % 3 == 0) cout << setw(50) << setfill(' ') << ' ' << setw(73) << setfill('#') << "#";

else {

cout << setw(50) << setfill(' ') << ' ' << "#";

cout << setw(24) << setfill('—') << "#";

cout << setw(24) << setfill('—') << "#";

cout << setw(24) << setfill('—') << "#";

}

cout << endl;

cout << setw(50) << setfill(' ') << ' ';

for (int j = 0; j < 9; ++j) {

if (j % 3 == 0) cout << "#";

else cout << "|";

if (grid[i][j] != 0) {

SetColor(Blue, Black);

cout << ' ' << setw(6) << setfill('\*') << ' ';

SetColor(White, Black);

}

else

if (us\_grid\_note[i][j] != '\0') {

SetColor(Magenta, Black);

cout << " " << us\_grid\_note[i][j];

SetColor(White, Black);

}

else cout << " ";

}

cout << "#" << endl;

cout << setw(50) << setfill(' ') << ' ';

for (int j = 0; j < 9; ++j) {

if (j % 3 == 0) cout << "#";

else cout << "|";

if (grid[i][j] != '\0')

if (i == row && j == col) {

SetColor(Blue, Black);

cout << " \*";

SetColor(White, Black);

cout << '[' << grid[i][j] << ']';

SetColor(Blue, Black);

cout << "\* ";

SetColor(White, Black);

}

else {

SetColor(Blue, Black);

cout << " \* ";

SetColor(White, Black);

cout << grid[i][j];

SetColor(Blue, Black);

cout << " \* ";

SetColor(White, Black);

}

else

if (us\_grid[i][j] != '\0') {

if (i == row && j == col) cout << " [ " << us\_grid[i][j] << " ] ";

else cout << " " << us\_grid[i][j] << " ";

}

else

if (i == row && j == col) cout << " [ ] ";

else cout << setw(8) << setfill(' ');

}

cout << "#" << endl;

cout << setw(50) << setfill(' ') << ' ';

for (int j = 0; j < 9; ++j) {

if (j % 3 == 0) cout << "#";

else cout << "|";

if (grid[i][j] != 0) {

SetColor(Blue, Black);

cout << ' ' << setw(6) << setfill('\*') << ' ';

SetColor(White, Black);

}

else cout << " ";

}

cout << "#" << endl;

}

cout << setw(50) << setfill(' ') << ' ' << setw(73) << setfill('#') << "#";

if (row == 9 && podType == 1) cout << endl << setfill(' ') << setw(71) << "[Сохранить игру]" << setw(44) << "Проверить";

else

if (row == 9 && podType == 2) cout << endl << setfill(' ') << setw(70) << "Сохранить игру" << setw(46) << "[Проверить]";

else cout << endl << setfill(' ') << setw(70) << "Сохранить игру" << setw(45) << "Проверить";

ch = \_getch();

//перемещение по полю и удаление числа с поля в качесвте заметки

if (ch == 224) {

ch = \_getch();

switch (ch) {

case Up:

if (row != 0) --row;

else row = 9;

break;

case Down:

if (row != 9) ++row;

else row = 0;

break;

case Right:

if (col != 8) ++col;

else col = 0;

if (row == 9) podType = 2;

break;

case Left:

if (col != 0) --col;

else col = 8;

if (row == 9) podType = 1;

break;

case Delete:

us\_grid\_note[row][col] = '\0';

break;

}

}

//внесение числа на поле

if (ch >= '1' && ch <= '9' && grid[row][col] == '\0') us\_grid[row][col] = ch;

//внесение числа на поле в качестве заметки

if (grid[row][col] == '\0')

if (ch == '!') us\_grid\_note[row][col] = '1';

else if (ch == '@' || ch == '"') us\_grid\_note[row][col] = '2';

else if (ch == '#' || ch == 252) us\_grid\_note[row][col] = '3';

else if (ch == '$' || ch == ';') us\_grid\_note[row][col] = '4';

else if (ch == '%') us\_grid\_note[row][col] = '5';

else if (ch == '^' || ch == ':') us\_grid\_note[row][col] = '6';

else if (ch == '&' || ch == '?') us\_grid\_note[row][col] = '7';

else if (ch == '\*') us\_grid\_note[row][col] = '8';

else if (ch == '(') us\_grid\_note[row][col] = '9';

//удаление символов с поля

if (ch == Backspace && grid[row][col] == '\0') us\_grid[row][col] = '\0';

if (ch == Enter && row == 9) {

//сохраняет судоку в файл

if (podType == 1) {

FILE\* file = fopen("SavedLvl", "wb");

fwrite(grid, 1, 81, file);

fwrite(us\_grid, 1, 81, file);

fwrite(us\_grid\_note, 1, 81, file);

fclose(file);

}

else

//проверить судоку

if (podType == 2) {

if (check\_solution(grid, us\_grid, us\_grid\_note)) {

system("cls");

levelPassed();

break;

}

ch = 0;

}

}

else

if (ch == Enter) ch = 0;

system("cls");

} while (ch != Enter && ch != Esc);

}