Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ИГРА “Судоку”

БГУИР КП 1-40 02 01 214 ПЗ

                                                                 Студент: гр. 250502 Кисель Д.Д.

Руководитель: Богдан Е. В.

МИНСК 2023

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

–––––––––––––––––––2023 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту  *Киселю Даниил Дмитриевичу*

Тема проекта *Игра “Судоку”* 2. Срок сдачи студентом законченного проекта *15 декабря 2023 г.*  3.Исходные данные к проекту*-Игра “Судоку”*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

1. Лист задания.

2. Введение.

3. Обзор литературы.

3.1 Постановка задачи.

3.2 Постановка задачи.

3.1. Обзор методов и алгоритмов решения поставленной задачи.

4. Функциональное проектирование.

4.1. Структура входных и выходных данных.

4.2. Разработка диаграммы классов.

4.3. Описание классов.

5. Разработка программных модулей.

5.1. Разработка схем алгоритмов (два наиболее важных метода)

5.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух методов

6. Результаты работы

7. Заключение

8. Литература

9. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов.*

*2. Схема алгоритма annealing( )*

*3. Схема алгоритма solve( )*

6. Консультант по проекту (с обозначением разделов проекта) Богдан Е. В.

7. Дата выдачи задания –––––*15.09.2023г.––––––––––––––––––––––   –*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала – 15 %;––––––––––––––––––––––––––––*

*разделы 2, 3 – 10 %;–––––––––––––––––––––––––––––––––––––*

*разделы 4 к –20 %;–––––––––––––––––––––––––––––––––––––––*

*разделы 5 к – 35 %;––––––––––––––––––––––––––––––––––––––*

*раздел 6,7,8 – 5 %;–––––––––––––––––––––––––––––––––––––––*

*раздел 9 к – 5%;–––––––––––––––––––––––––––––––––––––––––*

*оформление пояснительной записки и графического материала к 15.12.22 – 10 %*

*Защита курсового проекта с 21.12 по 28.12.23г.–––––––––––––––––––––––––*

РУКОВОДИТЕЛЬ Богдан Е. В.

(подпись)

Задание принял к исполнению *–––––––\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––*

(дата и подпись студента)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc152224749)

[1 ОБЗОР ЛИТЕРАТУРЫ 6](#_Toc152224750)

[1.1 Анализ существующих аналогов 6](#_Toc152224751)

[1.2 Постановка задачи 6](#_Toc152224752)

[1.3 Обзор методов и алгоритмов решения поставленной задачи 6](#_Toc152224753)

[2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 8](#_Toc152224754)

[2.1 Структура входных и выходных данных 8](#_Toc152224755)

[2.2 Разработка диаграммы классов 9](#_Toc152224756)

[2.3 Описание классов 9](#_Toc152224757)

[2.3.1 Класс Field 9](#_Toc152224758)

[2.3.2 Класс Cell 9](#_Toc152224759)

[2.3.3 Класс Generator 10](#_Toc152224760)

[2.3.4 Класс Solver 10](#_Toc152224761)

[2.3.5 Класс Gamewidget 10](#_Toc152224762)

[3 Разработка программных модулей 13](#_Toc152224763)

[3.1 Разработка схем алгоритмов 13](#_Toc152224764)

[3.2 Разработка алгоритмов 13](#_Toc152224765)

[3.2.1 Разработка алгоритма функции annealing( ) 13](#_Toc152224766)

[3.2.2 Разработка алгоритма метода solve() 13](#_Toc152224767)

[4 Результат работы 14](#_Toc152224768)

[ЗАКЛЮЧЕНИЕ 17](#_Toc152224769)

[СПИСОК ЛИТЕРАТУРЫ 18](#_Toc152224770)

[ПРИЛОЖЕНИЕ А 19](#_Toc152224771)

[ПРИЛОЖЕНИЕ Б 20](#_Toc152224772)

[ПРИЛОЖЕНИЕ В 21](#_Toc152224773)

[ПРИЛОЖЕНИЕ Г 22](#_Toc152224774)

# ****ВВЕДЕНИЕ****

C++ является мощным и эффективным языком программирования с широкими возможностями. Его особенности, такие как высокая производительность, а также возможность работы с объектно-ориентированным и процедурным программированием, делают его идеальным выбором для разработки игр. C++ обеспечивает высокую скорость выполнения программы, что важно для быстрой обработки игровой логики.

Qt - это мощный фреймворк с открытым исходным кодом, который предоставляет разработчикам широкие возможности для создания кроссплатформенных графических приложений. Он включает в себя богатую библиотеку инструментов для работы с GUI (графическим интерфейсом пользователя), обработки событий, создания окон, кнопок, таблиц и других элементов интерфейса.

Qt также предоставляет удобные инструменты для создания пользовательского интерфейса игры. С его помощью можно легко создавать графические элементы, обрабатывать события взаимодействия пользователя с интерфейсом, анимировать элементы и отображать информацию в удобном для игроков виде.

Сочетание C++ и Qt предоставляет разработчикам мощный инструментарий для создания игры с высокой производительностью, эффективностью использования ресурсов компьютера и удобным пользовательским интерфейсом. Этот стек технологий позволяет легко реализовать игровую логику, управление данными и создание приятного и интуитивно понятного интерфейса для игроков.

# ****ОБЗОР ЛИТЕРАТУРЫ****

## Анализ существующих аналогов

“Судоку” является одной из самых известных головоломок.

Существует множество аналогов начиная от простых браузерных вариантов игры и заканчивая необычными вариантами на ПК с различными режимами, например “Sudoku Jigsaw”.

Моя реализация имеет ряд недостатков по сравнению с различными аналогами. Например, простой визуальный интерфейс, который по сравнению с аналогами может показаться неприятным глазу. Моя реализация также имеет недостаток в виде ограниченного числа уровней.

Однако, при должной доработке, моя реализация может конкурировать с остальными аналогами.

## Постановка задачи

“Судоку” – логическая игра, представляющая собой игровое поле, размером 9 на 9 клеток, некоторые клетки которого заполнены числами от 1 до 9; поле разделено на 9 секторов, представляющих собой квадраты, размером 3 на 3 клетки. Правила игры: в каждом секторе, строке и столбце не может быть повторяющихся чисел. Цель игрока: заполнить игровое поле, не нарушив правила игры.

Программа должна иметь удобный визуальный интерфейс. В программе должна быть предусмотрена возможность загрузить заранее сгенерированный уровень выбранной сложности, получить подсказку в решении уровня.

## ****Обзор методов и алгоритмов решения поставленной задачи****

Для хранения поля игры используется двумерный массив. Процесс генерации поля состоит из двух этапов: заполнение поля числами от 1 до 9 и удаление чисел из поля.

В начале генерации поля, двумерный массив заполняется случайными числами таким образом, чтобы в каждой строке не было повторяющихся чисел. Для приведения поля к состоянию, в котором нет повторяющихся чисел в каждом столбце и секторе, был разработан алгоритм, основанный на методе имитации отжига.

Метод имитации отжига работает следующий образом: пусть имеется

некоторая функция 𝑓(𝑥) от состояния 𝑥, которую мы хотим минимизировать или максимизировать. Возьмём в качестве базового решения какое-то состояние 𝑥0 и будем пытаться его улучшать.

Введём температуру 𝑡 – какое-то действительное число, которое будет изменяться в течении оптимизации и влиять на вероятность перейти в соседнее состояние.

Пока не придём к оптимальному решению, будем повторять следующие шаги:

– уменьшим температуру;

– выберем случайного соседа 𝑥 - то есть какое-то состояние 𝑦, которое может быть получено из x каким-то минимальным изменением;

– с вероятностью 𝑝 (𝑓 (𝑥), 𝑓 (𝑦)) сделаем присвоение 𝑥 = 𝑦.

Стоит отметить, что в каждом шаге есть свобода реализации. Основные эвристические соображения следующие:

– в начале оптимизации наше решение плохое, поэтому мы можем позволить себе риск перейти в состояние хуже;

– в конце наоборот, наше решение почти оптимальное и мы не хотим терять прогресс, следовательно не можем позволить себе риск перейти в состояние хуже;

– температура должна быть высокой в начале и медленно уменьшаться

к концу;

– алгоритм будет работать лучше, если функция 𝑓 (𝑥) будет изменяться плавно.

Опишем более подробно детали, которые были изменены специально для текущей задачи: в качестве точки 𝑥 подразумевается текущее расположение чисел в строке. Значение функции 𝑓 (𝑥) в точке 𝑥 равняется числу повторяющихся чисел в каждом столбце и секторе.

Переход от точки 𝑥 к точке 𝑦 происходит путем перестановки двух чисел в строке.

Если 𝑓 (𝑦) < 𝑓 (𝑥), то выполняется присвоение 𝑥 = 𝑦. Иначе, присвоение не выполнятся.

Второй этап, удаление чисел, основан на поиске с возвратом.

Имеем какое-то число n – количество клеток, которые мы должны удалить. Выбирается и освобождается случайная клетка в игровом поле. Далее мы пробуем подставить в клетку все числа от 1 до 9. Если подставленное число удовлетворяет правилам игры, то мы делаем данную клетку пустой и заново повторяем данный ход действий пока не освободим n клеток, но уже с полем, в котором отсутствует клетка, освобожденная на предыдущем этапе алгоритма. Если на каком-то этапе алгоритма нам не удается подставить число в пустую клетку так, чтобы она удовлетворяла правилам игры, то мы возвращаемся к клетке, которая была освобождена до этой, и пытаемся подставить другое число, которое также удовлетворяло бы правилам игры, и повторяем данный алгоритм, пока не освободим n клеток.

# ****ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ****

## ****Структура входных и выходных данных****

На вход программы, в зависимости от выбранного уровня сложности, поступает игровое поле с некоторым количеством свободных клеток.

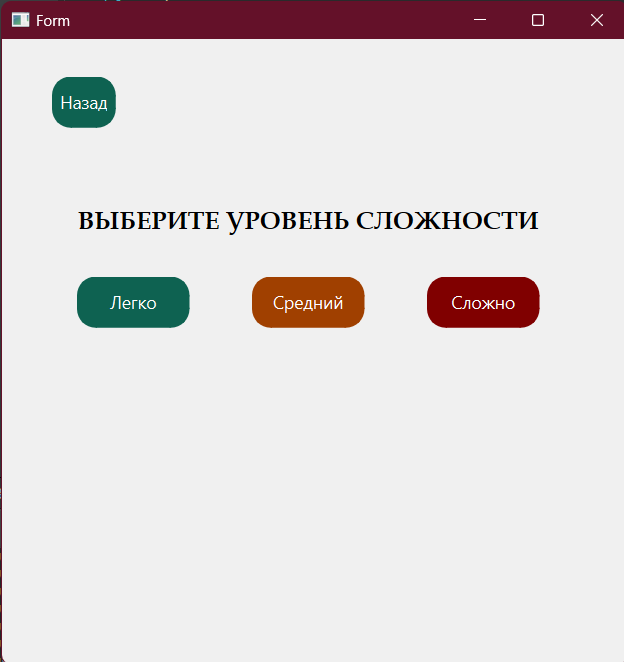


Рисунок 2.1.1 – Выбор уровня сложности

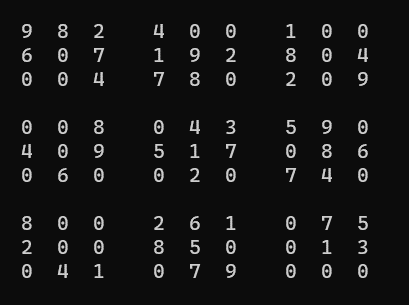


Рисунок 2.1.2 – Файл игрового поля “NFieldP.txt”

## ****Разработка диаграммы классов****

Диаграмма классов приведена в приложении А.

## ****Описание классов****

## Класс Field

Данный класс отвечает за хранение всей игры.

Описание полей класса:

Cell\*\* field – игровое поле

Solver\* slv – поле ответственное за решение головоломки

Generator\* gen – поле ответсвенное за генерацию головоломки

Описание методов класса:

Field() – конструктор класса

~Field() – деструктор класса

bool check\_field() – проверка на корректность игрового поля

bool load(std::string file\_name) – загрузить уровень с заданным названием

bool save(std::string file\_name) – сохранить уровень с заданным названием

## Класс Cell

Данный класс отвечает за клетку игрового поля.

Описание полей класса:

int value – значение клетки

int is\_locked – отвечает за возможность изменения значения клетки

Описание методов класса:

Cell( ) – конструктор класса

## Класс Generator

Данный класс отвечает за генерацию игрового поля.

Описание методов класса:

Generator( ) – конструктор класса

void generate(Field::Cell\*\* &field, int difficulty) – генерация игрового поля заданного уровня сложности.

## Класс Solver

Данный класс отвечает за решение головоломки.

Описание методов класса:

Solver( ) – конструктор класса

bool solve(Field::Cell\*\* &field) – метод, который решает головоломку

## Класс Gamewidget

Данный класс представляет собой графический интерфейс, а также описывает некоторые методы игрового процесса, которые должны отображаться на экране.

Все использованные виджеты наследуются от встроенного в Qt класса QWidget. В данных виджетах вместе с модификаторами public, private, protected используются методы, имеющие модификатор signals (сигналы) и модификатор public/private slots (слоты). Сигнал вырабатывается, когда происходит определенное событие. Слот – это функция, которая вызывается в ответ на определенный сигнал.

Описание методов класса:

explicit gamewidget(QWidget \*parent = nullptr) – коструктор класса

~gamewidget() – деструктор класса

void set\_value() – установка значения клетки

void onCellClicked() – запоминание последней нажатой клетки

void load() – загрузка игрового уровня

void check() – проверка на корректность поля

void solve() – решение уровня

void reset() – сброс решения

void timerSlot() – отвечает за работу секундомера

void play() – начало игры

void exit() – выход из игры

void easy() – установить уровень сложности на легкий

void medium() – установить уровень сложности на средний

void hard() – установить уровень сложности на сложный

void back\_to\_menu() – вернуться в главное меню

void back\_to\_difficulty() – вернуться к выбору уровня сложности

void hint() – получить посказку

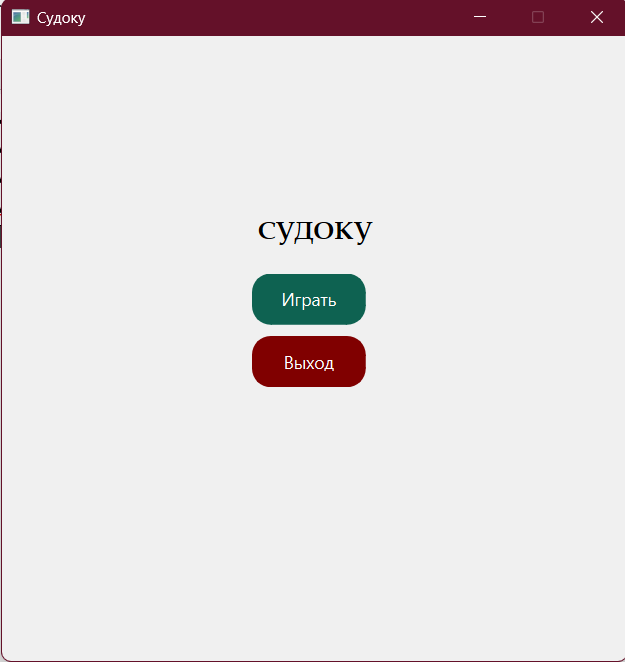


Рисунок 2.3.5.1 – Интерфейс главного меню

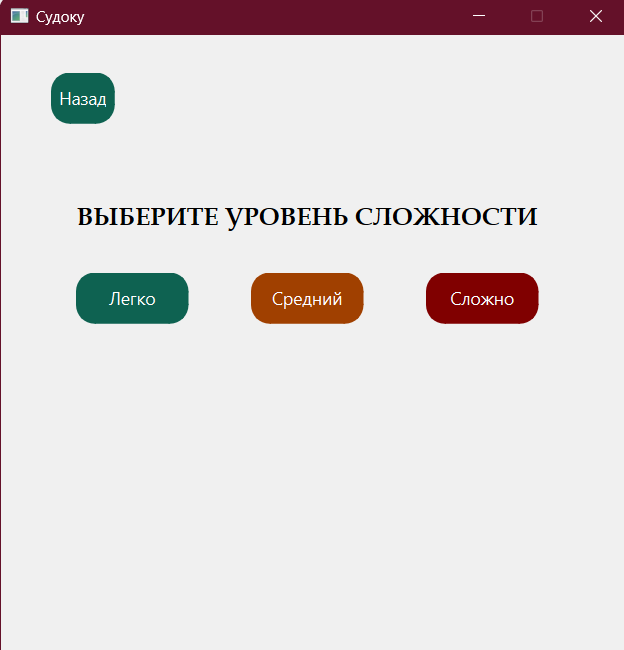


Рисунок 2.3.5.2 – Интерфейс раздела выбора уровня сложности

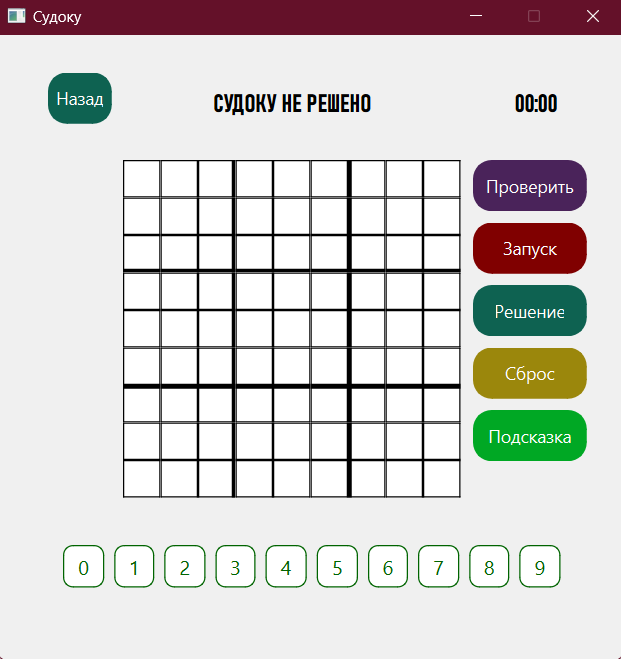


Рисунок 2.3.5.3 – Игровой раздел

# Разработка программных модулей

## Разработка схем алгоритмов

Функция annealing() производит генерацию игрового поля. Схема алгоритма приведена в приложении Б.

Метод solve() класса Solver решает уровень. Схема алгоритма приведена в приложении В.

## Разработка алгоритмов

## Разработка алгоритма функции annealing( )

Функция annealing() производит генерацию поля без удаления клеток игрового поля.

Шаг 1. Начало.

Шаг 2. Устанавливаем начальное значение температуры, конечное значение температуры, коэффициент остывания.

Шаг 3. Инициализируем игровое поле с помощью функции field\_init(curr).

Шаг 4. Создаем временное хранилище игрового поля, которое будем изменять в процессе алгоритма, и помещаем в него текущее игровое поле.

Шаг 5. Запоминаем количество совпадающих чисел в игровом поле.

Шаг 6. Перемешиваем два случайных числа в случайной строке и сравниваем количество совпадающих чисел.

Шаг 7. Если оно меньше предыдущего, то переходим в это состояние поля. Если больше или равно – оставляем изначальное состояние поля.

Шаг 8. Уменьшаем значение температуры на коэффициент остывания.

Шаг 9. Повторяем шаги 5 – 8 пока текущая температура не станет равна конечной.

Шаг 10. Сохраняем текущее состояние игрового поля.

Шаг 11. Конец.

## Разработка алгоритма метода solve()

Метод solve() производит решение головоломки.

Шаг 1. Начало.

Шаг 2. Находим пустую клетку на поле.

Шаг 3. Пытаемся подставить каждое число от 1 до 9.

Шаг 4. Если игровое поле с данным числом в клетке не нарушает правил игры, то оставляем его и повторяем шаг 2 и 3, пока не заполним все пустые клетки или же не сможем подставить все числа от 1 до 9 так, чтобы не нарушить правила.

Шаг 4. Конец.

# Результат работы

После нажатия кнопки “Играть” в главном меню, мы попадаем в меню выбора сложности, где игрок может выбрать один из трех предложенных уровней сложности.

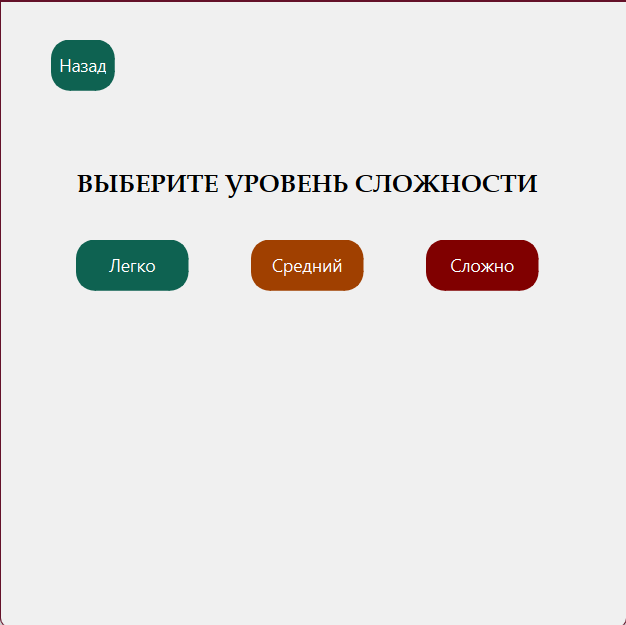


Рисунок 4.1 – Меню выбора уровня сложности

После выбора уровня сложности игрок попадает в игровую область (рисунок 4.2).

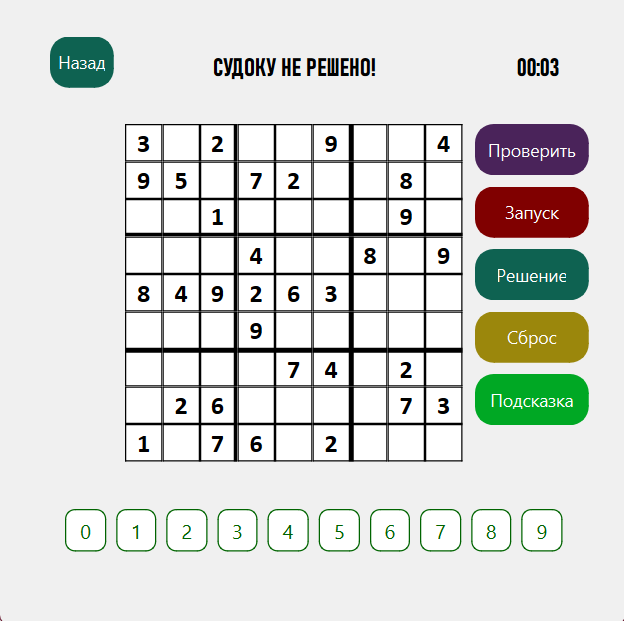


Рисунок 4.2 – Игровая область

Заполнение поля происходит следующим образом. Игрок нажимает на клетку, в которую хочет вставить число и нажимает на числа снизу. При полном заполнении игрового поля, игрок может нажать кнопку “Проверка”. Если игрок правильно решил уровень, то надпись над игровым полем покажет “Судоку решено!”. Иначе – “Судоку не решено!”.

При возникновении трудностей с прохождением уровня игрок может воспользоваться кнопкой “Подсказка”, которая открывает одну случайную клетку, или кнопкой “Решение” для полного решения всего уровня.

В случае если игрок хочет начать уровень заново, он может воспользоваться кнопкой “Сброс”.

На игровом поле также присутствует секундомер, чтобы игрок мог отслеживать свой прогресс по скорости в прохождении уровня.

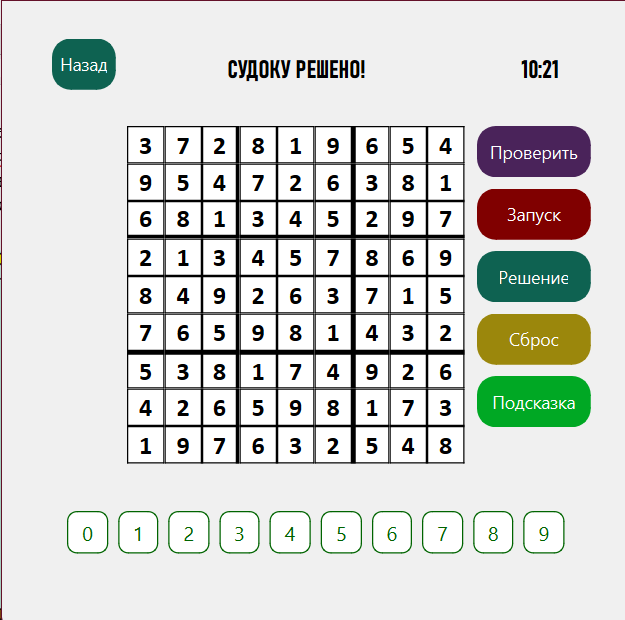


Рисунок 4.3 – Пример пройденного уровня

# ЗАКЛЮЧЕНИЕ

В процессе выполнения данной курсовой работы была разработана игра в “Судоку” на языке программирования C++ с использованием фреймворка Qt. В процессе работы были изучены основы создания графического интерфейса с помощью Qt, а также реализованы алгоритмы генерации и решения игрового поля.

Программа предоставляет пользователю удобный и интуитивно понятный интерфейс для игры в “Судоку”, позволяя выбирать уровень сложности и решать игровые уровни. Были реализованы функции генерации и решения уровней, проверки правильности заполнения игрового поля и подсказок для помощи игроку в решении головоломки.

Данная работа позволила углубить знания в области программирования на C++, ознакомиться с применением библиотеки Qt для создания графических приложений и практически применить алгоритмы решения головоломки “Судоку”.

Реализованный проект также предоставляет пользователю возможность не только играть в “Судоку”, но и изучать методы решения головоломки, следить за своими успехами и улучшать навыки логического мышления.

В целом, данная работа позволила углубить знания о языке программирования C++, понять особенности работы с фреймворком Qt и научиться применять полученные знания для разработки программ.

Код программы приведен в приложении Г.

# СПИСОК ЛИТЕРАТУРЫ

[1] Стивен Прата. Язык программирования С++. Лекции и упражнения: учеб.

пособие / С. Прата – СПб.: ООО «ДиаСофтЮП», 2003. – 1104с.

[2] Программирование на C++ [Электронный ресурс]. -  Режим доступа: <https://metanit.com/cpp/tutorial/>.  - Дата доступа: 23.11.2023.

[3] Qt Documentation [Электронный ресурс]. -Электронные данные.

-Режим доступа: <https://doc.qt.io/> Дата доступа: 15.11.2023

[4] А. В.Чеботарев Библиотека Qt 4. Программирование прикладных приложений в среде Linux / А. В.Чеботарев - Л.: Наука, 2013. - 821 с.

[5] Страуструп, Б. Язык программирования С++ / Б. Страуструп. - М. : БИНОМ, 2004.- 1098 с.

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Диаграмма классов

# ПРИЛОЖЕНИЕ Б

*(обязательное)*

Схема функции annealing()

# ПРИЛОЖЕНИЕ В

*(обязательное)*

Схема алгоритма метода solve()

# ПРИЛОЖЕНИЕ Г

*(обязательное)*

Код программы