Омский Научный центр Сибирского отделения Российской академии наук

Региональная общественная организация «Омский совет ректоров»

Омское региональное отделение Всероссийской общественной организации

«Русское географическое общество»

Детская областная общественная организация

«Научное общество учащихся «Поиск»

Автономная некоммерческая организация дополнительного профессионального образования «Центр обучения «Махаон»

52-я

Межрегиональная научно-практическая конференция

школьников и учащейся молодежи

Тема: «Мобильное приложение для решения судоку»

Учебно-исследовательская работа

Научное направление: Информатика и программирование

*Выполнил:*

ученик 10 класса

Школы “Интеллект”

Барайщук Леонид Васильевич

*Научный руководитель:*

педагог д.о. АНО ДПО “ЦО “Махаон”

Морозов Антон Дмитриевич

Омск – 2020

Оглавление

[Введение 3](#_Toc35677162)

[Цель 3](#_Toc35677163)

[Задачи 3](#_Toc35677164)

[Описание проекта 4](#_Toc35677165)

[Правила игры в судоку 4](#_Toc35677166)

[Аналоги 4](#_Toc35677167)

[Алгоритмы решения судоку 5](#_Toc35677168)

[Техническая реализация 5](#_Toc35677169)

[Средства реализации 5](#_Toc35677170)

[Интерфейс 6](#_Toc35677171)

[Описание алгоритмов 7](#_Toc35677172)

[Решение судоку 8](#_Toc35677173)

[Решение каждой ячейки отдельно 9](#_Toc35677174)

[Генератор судоку 9](#_Toc35677175)

[Выводы и перспективы 11](#_Toc35677176)

[Источники: 12](#_Toc35677177)

# Введение

Головоломки не только позволяют развивать логику, мышление, внимательность, но и позволяют отвлечься от рутины, проводя время с пользой. Хороший пример такой головоломки является судоку. Эту игру изобрёл Леонард Эйлер в XVIII веке, изначально она называлась “Латинский квадрат”. Настоящую популярность игра завоевала в 1980 —1990-х годах, когда японские журналы начали регулярно публиковать её на своих страницах. [1].

Судоку достаточно популярная головоломка, поэтому для её решения созданы различные программы, веб-сервисы и мобильные приложения. Тем не менее, эти продукты часто используют готовые базы головоломок, что приводит к тому, что при частом решении задания начинают повторяться. К тому же эти сервисы могут проверить правильность готового решения, но не всегда могут дать пользователю подсказки при возникновении у него затруднений в ходе решения.

Поэтому я решил создать мобильную версию этой игры с возможностью подсказок для игрока и случайной генерацией заданий.

# Цель

Создание мобильного приложения для решения судоку с возможностями случайной генерации заданий и подсказок пользователю в ходе решения.

# Задачи

1. Выбрать площадку и язык программирования для создания.
2. Изучить среду разработки Android Studio.
3. Изучить язык программирования Kotlin.
4. Продумать структуру и создать дизайн приложения.
5. Выбрать и реализовать алгоритм для решения судоку.
6. Написать генератор заданий.
7. Создать рабочую версию приложения
8. Тестирование и отладка приложения.

# Описание проекта

## Правила игры в судоку

Игровое поле представляет собой квадрат размером 9×9, разделённый на меньшие квадраты со стороной в 3 клетки. Таким образом, всё игровое поле состоит из 81 клетки. В них уже в начале игры стоят некоторые числа (от 1 до 9), называемые подсказками. От игрока требуется заполнить свободные клетки цифрами от 1 до 9 так, чтобы в каждой строке, в каждом столбце и в каждом малом квадрате 3×3 каждая цифра встречалась бы только один раз.

## Аналоги

Для игры в судоку был создан широкий спектр программ, веб-сервисов и мобильных приложений. Но многие из них имеют не достаточный функционал для комфортной игры. Ниже представлены примеры популярных сервисов для решения головоломки:

1. Sudoku.org.ua – этот веб-сервис имеет огромную базу с готовыми головоломками (примерно 2000 для каждой сложности). Именно поэтому у игры нет возможности давать подсказки, а может только проверить готовое решение. [2]
2. Sudoku.com – самое популярное приложение по головоломке судоку в “play market”. Данная игра имеет свой генератор заданий, а также может давать пользователю подсказки. Но если игроку нужно увидеть готовое решение сразу, то приходится просить подсказку для каждой клетки отдельно. [3]
3. Scanvord.net – этот веб-сервис предлагает выбрать судоку из готовой базы. У него нет функции, позволяющей давать подсказки, но зато может показать готовый ответ головоломки. [4]

После анализа аналогов я решил создать приложение, в котором можно решать судоку, генерировать задания различного уровня сложности или вводить их вручную, а также реализовать подсказки для клетки, которую запросит пользователь.

## Алгоритмы решения судоку

При выборе алгоритма для решения судоку, было рассмотрено несколько вариантов:

1. Решение судоку по правилам – этот алгоритм подразумевает собой решение задачи тем способом, каким решал бы его игрок. Главными недостатками этой идеи являются: сложность в реализации и сложность в оптимизации такого большого алгоритма [5].
2. Перебор всех вариантов – Этот метод самый простой в реализации, но перебор занимает очень много времени, и скорее всего на решение судоку уходило бы больше минуты.
3. Backtracking – Метод, который используют почти все приложения с возможностью подсказок. Это рекурсивный алгоритм поиска, который достаточно прост в реализации, а также оптимизирован и работает достаточно быстро, и игрок не замечает процесса решения [6].
4. Нейронные сети также используются для решения судоку. Но для этой головоломки, они не кажутся лучшим решением, так как имеют ряд недостатков. Например: обучение сети может привести к тупиковым ситуациям, а также нельзя получить информацию о решении [7].

Из найденных способов решения судоку я решил использовать рекурсивный алгоритм поиска Backtracking [6].

# Техническая реализация

## Средства реализации

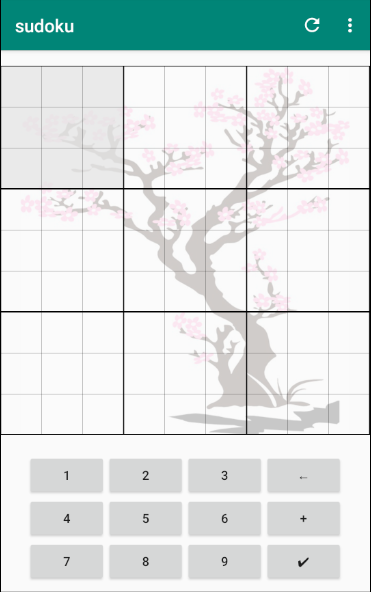
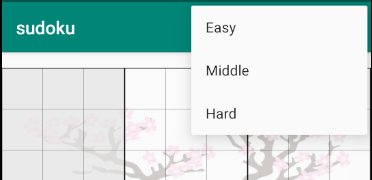
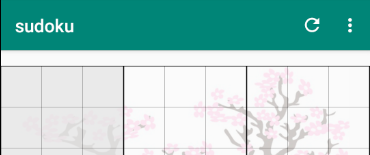
Для реализации проекта был выбран язык программирования “Kotlin” и среда разработки “Android studio”.

Kotlin - это язык программирования разработанный “JetBrains”. Он прагматичен и краток, благодаря чему написание кода превращается в приятный и эффективный процесс [8].

Среда разработки Android studio мне показалась лучшим решением для создания мобильной игры, т.к. имеет удобные инструменты для создания интерфейса и хороший эмулятор.

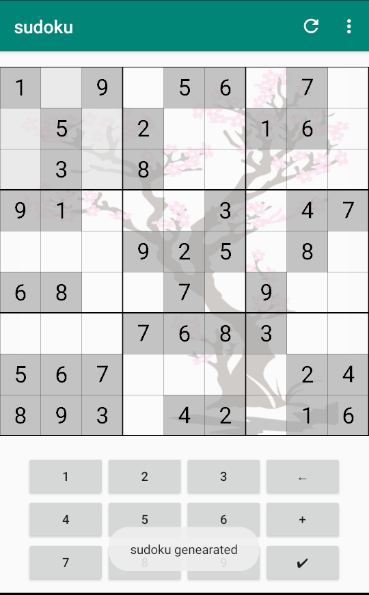
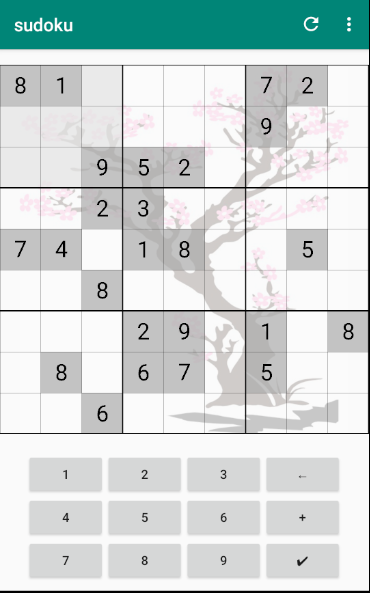
## Интерфейс

При запуске игры показывается пустое поле игре для игры (*рис. 1*). При желании игрок может ввести свою задачу. Под игровым полем есть 12 кнопок: кнопки от 1 до 9, кнопка стирания ячейки, кнопка для решения одной ячейки и кнопка для решения всего судоку.



*Рисунок 1 Рисунок 2*

С помощью кнопок на верхней панели (*рис. 2*) можно очистить всё поле, либо сгенерировать судоку одной из трёх предложенных сложностей.

После выбора сложности, на поле появляется сгенерированная головоломка с неизменяемыми ячейками.

*Рисунок 3, пример лёгкого судоку Рисунок 4, пример сложного судоку*

## Описание алгоритмов

Для проверки судоку, важно знать, что поставленная цифра должна подходить по 3 аспектам:

1. В ряду нет такой же цифры.
2. В столбце нет такой же цифры.
3. В малом квадрате нет такой же цифры (3х3 квадрат).

Алгоритм получает координаты поставленной цифры, после чего проверяет, не совпадает ли она с цифрами из её ряда, столбца и малого квадрата.

Если все условия совпали, то функция возвращает True.

fun valid(n: Int, r: Int, c: Int): Boolean{  
 for(i in 0..8){  
 val cell = board.getCell(r, i)  
 if(cell.value == n && i != c){  
 return false  
 }  
 }  
 for(i in 0..8){  
 val cell = board.getCell(i, c)  
 if(cell.value == n && i != r){  
 return false  
 }  
 }  
 val x = c / 3  
 val y = r / 3  
 for(i in y\*3..y\*3+2){  
 for (j in x\*3..x\*3+2){  
 val cell = board.getCell(i,j)  
 if (cell.value == n && i != r && j != c){  
 return false  
 }  
 }  
 }  
 return true  
}

## Решение судоку

Для того чтобы приложение могло давать подсказки пользователю, оно должно само уметь решать судоку.

Для решения головоломки был выбран рекурсивный алгоритм поиска [6]. Сначала алгоритм находит первую попавшуюся пустую клетку. После чего в найденную клетку подставляются в цифры от 1 до 9. Каждую цифру проверяет алгоритм проверки поставленных цифр. Как только подходящая цифра находится, алгоритм запускается снова. Если заново запущенный алгоритм не смог найти пустую клетку, то все поставленные цифры выводятся. Но если пустая клетка находится, то алгоритм повторяется. В случае если ни одна из 9 цифр не подошла, алгоритм возвращается к предыдущей клетке и подбирает другое число.

fun solver(): Boolean{  
 var empty: Boolean = false  
 var emptyR: Int = 0  
 var emptyC: Int = 0  
 for(r in 0..8){  
 for (c in 0..8){  
 val cell = board.getCell(r, c)  
 if (cell.value == 0){  
 empty = true  
 emptyR = cell.row  
 emptyC = cell.col  
 break  
 }  
 }  
 if (empty){  
 break  
 }  
 }  
 if (!empty){  
 return true  
 }  
 else{  
 for(i in 1..9){  
 if (valid(i, emptyR ,emptyC)){  
 val cell = board.getCell(emptyR, emptyC)  
 cell.value=i  
 if(solver()){  
 return true  
 }  
 else{  
 cell.value = 0  
 }  
 }  
  
 }  
 }  
 return false  
}

## Решение каждой ячейки отдельно

Алгоритм получает координаты клетки, в которой пользователь хочет увидеть подсказку. Перед решением головоломки, все цифры сохраняются в массив. После этого запускается алгоритм решения судоку. Как только задача решена, все клетки, кроме выбранной, восстанавливаются.

## Генератор судоку

Сначала поле очищается, после чего запускается рекурсивный алгоритм поиска. На этот раз, для каждой клетки перебираются числа в перемешанном порядке, причем для каждой ячейки порядок цифр разный.

Далее выбираются случайно выбранные клетки, которые будут известны игроку. Для этого генерируется список от 0 до 80 и перемешивается. После, берутся первые N чисел из списка, и клетки с такими индексами очищаются. Число N зависит от сложности судоку.

Лёгкий - N = 40; Средний - N = 50; Тяжёлый - N = 55

Также все оставшиеся цифры становятся неизменяемыми.

fun generate(): Boolean{  
 var empty: Boolean = false  
 var emptyR: Int = 0  
 var emptyC: Int = 0  
 for(r in 0..8){  
 for (c in 0..8){  
 val cell = board.getCell(r, c)  
 if (cell.value == 0){  
 empty = true  
 emptyR = cell.row  
 emptyC = cell.col  
 break  
 }  
 }  
 if (empty){  
 break  
 }  
 }  
 if (!empty){  
 return true  
 }  
 else {  
 var l: MutableList<Int> = *mutableListOf*(1,2,3,4,5,6,7,8,9)  
 l.*shuffle*()  
 for(i in 0..8){  
 if (valid(l[i], emptyR ,emptyC)){  
 val cell = board.getCell(emptyR, emptyC)  
 cell.value=l[i]  
  
 if(generate()){  
 return true  
 }  
 else{  
 cell.value = 0  
 }  
 }  
  
 }  
 }  
 return false  
}

fun cleaneasy(){  
 var l: MutableList<Int> = *mutableListOf*()  
 for (i in 0..80){  
 l.add(i)  
 board.cells[i].isStartingCell=true

}  
 l.*shuffle*()  
 for (i in 0..40){  
 val cell = board.getCell(l[i]/9, l[i]%9)  
 cell.value=0  
 board.cells[l[i]].isStartingCell=false

}  
}

Для версионного контроля проекта использовалась система контроля версий Git. С исходным кодом проекта можно ознакомиться на сервисе GitHub (ссылка на репозиторий https://github.com/le000nid/sudoku).

# Выводы и перспективы

Были изучены: язык программирования Kotlin и среда разработки Android Studio. С их помощью было создано приложение, позволяющее решать головоломки судоку, генерировать задания и выполнять подсказки пользователю при решении.

Исходный код проекта доступен на сервисе GitHub: https://github.com/le000nid/sudoku. Установочный файл для мобильных устройств с операционной системой Android можно скачать по ссылке: https://drive.google.com/open?id=1pFi\_df9iLlQa-agii34YO3vL1JUGTI\_U

В дальнейшем проект планируется дополнить функцией распознавания напечатанного варианта головоломки. Это позволит переносить головоломку из сборника или журнала в игровое поле приложения.

# Источники:

1. Sudoku // Wikipedia URL:

https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4%D0%BE%D0%BA%D1%83

1. Веб-сервис по решению судоку // sudoku.org.ua URL:

http://sudoku.org.ua/

1. Приложение по решению судоку // sudoku.com URL:

https://sudoku.com/

1. Веб-сервис по решению сканвордов // scanvord.net URL:

https://scanvord.net/

1. Описание алгоритма для решения судоку по правилам // habr.com URL:

https://habr.com/ru/post/173795/

1. Описание рекурсивного алгоритма поиска Backtracking // geekbrains.ru URL:

https://geekbrains.ru/events/1298

1. Описание использование нейронных сетей в судоку // issue.life URL:

https://issue.life/questions/44397123

1. Справочник языка Kotlin // kotlinglang.org URL:

https://kotlinlang.org/docs/reference/

1. Справочник Android Studio // developer.android.com URL:

https://developer.android.com/docs

1. Система вопросов и ответов // stackoverflow.com URL:

https://ru.stackoverflow.com/