

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «Прикладное программирование»**

**Тема: «Модуль конструктора для Химической викторины на Android»**

Выполнил студент

Суровых Андрей Романович

Группа П1-16

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

**Королёв, 2019 г.**

**Приложение к заданию на курсовой проект**

**Краткое описание программы:**

Приложение-викторина на мобильные телефоны под управлением ОС Android, написанное с использованием языка Kotlin и библиотек Android.

**Разработать для программы:**

1. Режим «Викторина».

Пользователю задаются 10 случайных вопросов из базы данных. На каждый вопрос есть 4 варианта ответа. При ответе цветом выделяются верные и неверные ответы, через секунду приложение переходит к следующему вопросу. После ответа на все вопросы выводится количество верных ответов.

1. Режим «Собери слово из элементов».

Пользователю задаются 10 случайных вопросов из базы данных. На каждом вопросе выводится 12 случайных элементов таблицы Менделеева, среди которых есть нужные для построения слова. Также выводится картинка, в которой загадано нужное слово. При верном ответе выводится краткая информация о каждом использованном элементе и перевод слова на русский язык. Далее ожидается нажатие в любое место экрана, чтобы перейти к следующему вопросу.

1. Экраны результатов викторины.

На экране результатов любого режима викторины должно выводиться количество правильных ответов из максимально возможного. Также должна быть кнопка перехода в главное меню.

1. Экраны логотипов разработчиков.

На этих экранах должна быть картика с логотипом разработчика приложения. Для перехода обратно на главный экран нужно нажать на экран телефона.

**Входные данные:**

На вход программе поступает статичная база данных, включающая в себя:

1. Вопросы для режимов с вариантами ответов, названиями картинок для вопросов. У каждого режима своя таблица.
2. Таблица Менделеева (Номер в таблице, краткие и полные названия)

Структура нашей базы данных представлена на рисунке ниже:

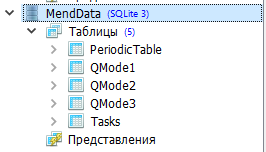


Рис. 1. Структура базы данных

**Оглавление**

[Введение 4](#_Toc12262098)

[1. Теоретическая часть 6](#_Toc12262099)

[1.1 Обычная викторина 6](#_Toc12262100)

[1.2 Исключи лишнее 7](#_Toc12262101)

[1.3 Расстановка в правильном порядке 8](#_Toc12262102)

[1.4 Собери слово из элементов 10](#_Toc12262103)

[2. Проектная часть 11](#_Toc12262104)

[1.1 Описание инструментов разработки программы 11](#_Toc12262105)

[1.2 Описание модулей программы 13](#_Toc12262106)

[3. Организационная часть 19](#_Toc12262107)

[3.1 Порядок установки программного средства 19](#_Toc12262108)

[3.2 Руководство оператора 19](#_Toc12262109)

[3.2.1 Минимальный состав технических и программных средств 19](#_Toc12262110)

[3.2.2 Требования к пользователю 19](#_Toc12262111)

[3.3 Запуск программы 19](#_Toc12262112)

[3.4 Игровой процесс 20](#_Toc12262113)

[Заключение 28](#_Toc12262114)

[Литература и источники 29](#_Toc12262115)

[Приложение 1 30](#_Toc12262116)

[Приложение 2 31](#_Toc12262117)

[Приложение 3 37](#_Toc12262118)

[Приложение 4 38](#_Toc12262119)

[Приложение 5 39](#_Toc12262120)

[Приложение 6 41](#_Toc12262121)

[Приложение 7 45](#_Toc12262122)

[Приложение 8 46](#_Toc12262123)

# **Введение**

Для выполнения данного курсового проекта необходимо разработать программу для решения поставленных задач, указав последовательное выполнение соответствующих модулей для получения необходимых результатов. Разработка программы будет происходить на языке программирования Kotlin[3]. В программе должны быть учтены моменты, позволяющие пользователям легко освоить все возможности программы, для этого необходимо создать удобный интерфейс программы.

В первой части будут рассмотрены: предметная область химия, а также жанр игр «Викторина».

Во второй части будут рассмотрены инструменты и модули, которые были разработаны, структура программной части и листинги ключевых частей программных модулей.

В третьей части будет освещено взаимодействие пользователя с программой, приведены изображения графического интерфейса программы и сообщения оператору. В заключительной части будут сделаны общие выводы о получившемся проекте и полученных знаниях при разработке проекта.

Цель работы заключается в том, чтобы составить алгоритм поставленной задачи по разработке программы химической викторины на Android[9], разработать и отладить программу, реализующую разработанный алгоритм.

# **Теоретическая часть**

**Типы игр жанра «Викторина»**

**Викторина** – [игра](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0), заключающаяся в ответах на устные или письменные вопросы из различных областей знания.

Викторины в основном отличаются друг от друга правилами, определяющими очерёдность хода, тип и сложность вопроса, порядок определения победителей, вознаграждение за правильный ответ.

У этого жанра есть много различных проявлений. Вот те из них, которые есть в нашей игре:

1. Обычная викторина
2. Исключи лишнее
3. Расстановка в правильном порядке
4. Собери слово из элементов

## **1.1 Обычная викторина**

В обычной викторине вам задаётся вопрос и несколько вариантов ответов. Вы должны выбрать один верный ответ.

Примером такого приложения является “История России Викторина”[4]

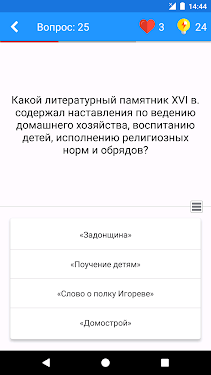


Рис. 1. Игра “История России Викторина”

В нашей игре за один тур задаётся 10 вопросов один за другим, связанных с химией. В каждом вопросе 4 варианта ответа. После каждого вопроса вы сразу можете увидеть верный ответ. После ответа на каждый вопрос выводится количество верных ответов.

С нашей точки зрения, этот режим самый классический из всех, что есть в нашей игре.

## **1.2 Исключи лишнее**

В этом режиме вам даётся ряд слов или словосочетаний, из которых вы должны выбрать то, которое является лишним в этом ряду.

Хорошим примером такой игры является игра “Угадай лишнее слово!”[6]



Рис. 2. Игра “Угадай лишнее слово!”

В нашей игре все варианты ответа так или иначе связаны с химией. После каждого ответа вы увидите верный ответ. В этом режиме также 10 раундов, после прохождения которых вы увидите сколько вы дали верных ответов.

Наше мнение об этом режиме:

Данный режим интереснее обычной викторины и хватает его на большее время.

## **1.3 Расстановка в правильном порядке**

В этом режиме вам также даются слова или словосочетания, но их уже нужно расположить их в правильном порядке.

Подобным режимом обладает обучающая программа “Duolingo”[7]

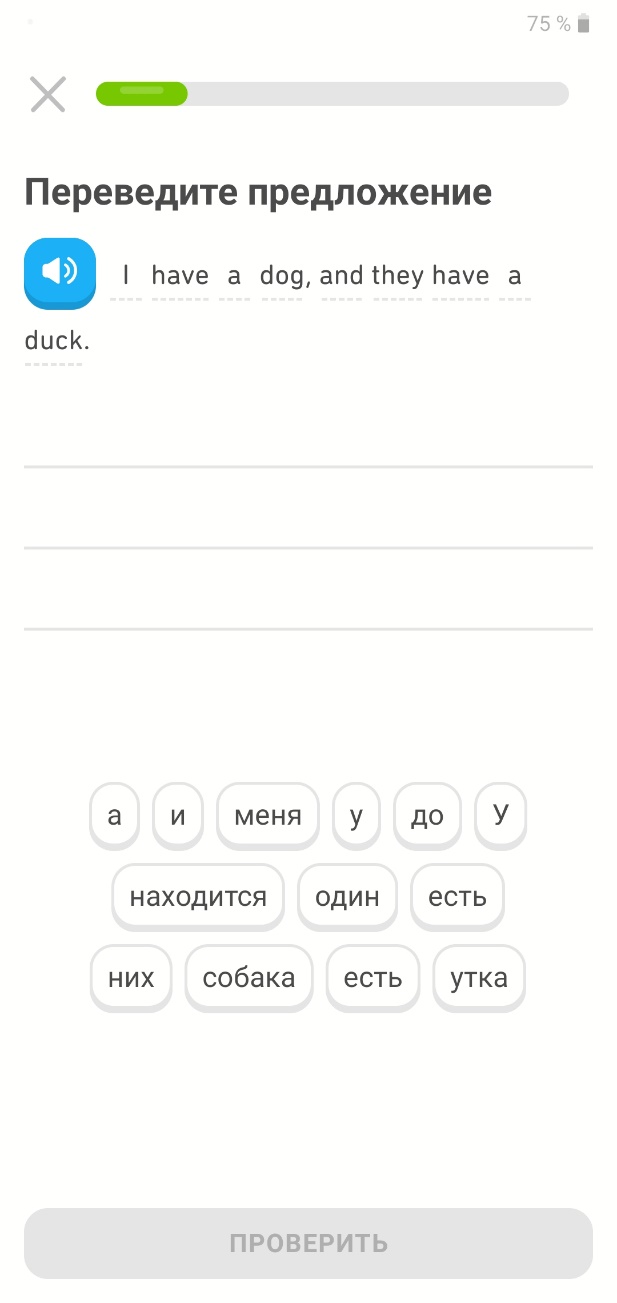


Рис. 3. Приложение “Duolingo”

В нашей игре вам даётся ряд химических терминов, которые надо расставить в соответствии со степенью их раскрытия в последующих понятиях. После того как вы используете все понятия, вы увидите верно ли вы ответили. После ответа на все такие вопросы вы увидите сколько из ваших ответов были верными.

С нашей точки зрения, этот режим интереснее предыдущих и заставляет больше думать. Здесь нет разных вопросов, но заставляет сильно задуматься в каком порядке должны стоять термины.

## **1.4 Собери слово из элементов**

Обычно в этом режиме вам даётся картинка и несколько букв. Из них вы должны составить слово, загаданное на картинке.

Примером такой игры является игра “4 фото 1 слово”[5]



Рис. 4. Игра “4 фото 1 слово”

В нашей игре мы пошли дальше. У нас есть картинка, на которой загадано некое слово. Также у нас есть 12 элементов периодической таблицы Менделеева. Из них вы должны составить загаданное на картинке слово, но это слово будет на английском языке, т. к. элементы таблицы состоят из английских букв. Таким образом мы затронули тему не только изучения химии, но и английского языка. После правильного ответа на каждый вопрос вы увидите небольшую информацию о каждом элементе таблицы, использованном в этом слове, а также перевод слова на русский язык.

Также в нашем режиме есть 3 вида сложности: простой, географический (Загадываются географические термины) и сложный.

С нашей точки зрения, это самый интересный режим из всех. Он не надоест так быстро, в отличие от остальных режимов.

# **Проектная часть**

## **1.1 Описание инструментов разработки программы**

**Язык программирования**

Данный курсовой проект был разработан с использованием языка программирования Kotlin. В сравнении класса Java[12] с эквивалентным классом Kotlin демонстрирует лаконичность кода Kotlin. Для той же операции, что выполняется в классе Java, класс Kotlinтребует меньше кода. Одно из основных различий между Java и Kotlin заключается в том, что в последнем нет условий для проверяемых исключений (checked exception). Следовательно, нет необходимости отлавливатьили объявлять какие-либо исключения. Несомненным преимуществом является полная совместимость с Java, в том числе и обратная. Все библиотеки для Java будут работать на Kotlin и наоборот. Также Kotlin позволяет разработчикам расширять класс новыми функциями с помощью функций расширения. Эти функции, хотя и доступны в других языках программирования, таких как C#, не доступны в Java. В отличие от Java, в Kotlin все типы по умолчанию являются не-nullable. Если разработчики попытаются присвоить или вернуть значение null в коде Kotlin, во время компиляции произойдет сбой. Классы Kotlin, в отличие от классов Java, могут иметь один или несколько вторичных конструкторов, в дополнение к первичному конструктору. Это делается путем включения этих вторичных конструкторов в объявление класса. И Java, и Kotlin имеют свои преимущества друг перед другом.

Kotlin - язык программирования, разработанный компанией JetBrains[8], работающий на платформе Java. Он использует JDK, как и сама Java, но имеет другой синтаксис. Решение не новое - уже существуют языки, которые так делают: Scala и Closure, например. Появились они, в основном, из-за проблем с самой Java в плане синтаксиса. Синтаксис языка использует элементы из Паскаля, TypeScript, Haxe, PL/SQL, F#, Go и Scala, C++, Java, C#, Rust и D. При объявлении переменных и параметров типы данных указываются после названия (разделитель - двоеточие). Точка с запятой как разделитель операторов так же необязательна, как в Scala и Groovy; в большинстве случаев перевода строки достаточно, чтобы компилятор понял, что выражение закончилось. Кроме объектно-ориентированного подхода, Kotlin также поддерживает процедурный стиль с использованием функций. Как и в [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), C++ и D, [точка входа](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%87%D0%BA%D0%B0_%D0%B2%D1%85%D0%BE%D0%B4%D0%B0) в программу – функция main, принимающая массив параметров командной строки. Программы на Kotlin также поддерживают [Perl](https://ru.wikipedia.org/wiki/Perl)- и [shell](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BE%D0%BB%D0%BE%D1%87%D0%BA%D0%B0_Unix)-стиль интерполяции строк (переменные, включённые в строку заменяются на их содержимое). Также поддерживается [вывод типов](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B2%D0%BE%D0%B4_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2).

Kotlin помог разработчикам писать программы с меньшим количеством кода. Помимо всего того, что есть в Java, он добавляют вещи из мира функционального программирования. Это значительно облегчает написание кода - делает его короче и выразительнее. Чем меньше кода мы пишем, тем меньше кода нужно поддерживать, писать меньше тестов. Собственно, большинство языков появилось по этой самой причине - поменять синтаксис Java, сделать его более удобным, более прагматичным.

**Среда разработки программного обеспечения**

**Android Studio[10]** – это интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, - официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, OS X и Linux. 17 мая 2017, на ежегодной конференции Google I/O, Google анонсировал поддержку языка Kotlin, используемого в Android Studio, как официального языка программирования для платформы Android в добавление к Java и С++.

**База данных**

**SQLite[11]** – компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние. В 2005 году проект получил награду Google-O’Reilly Open Source Awards.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу.

## **1.2 Описание модулей программы**

Всё приложение состоит из нескольких модулей, которые представляют собой классы экранов приложения. Они описывают то, как программа будет выглядеть на экране смартфона. Каждому классу соответствует также XML файл макета экрана, который далее подключается в классе. (Приложение 8)

Модули программы, разработанные мной:

* Экран логотипов разработчиков
* Экран режима «Собери слово»
* Экран результатов режима «Собери слово»
* Экран режима «Викторина»
* Экран результатов режимов викторины

**Экран логотипов разработчиков –** это простой информационный экран, на котором располагается только 1 картинка-логотип, которая выбирается приложением в зависимости от одной из 2-х нажатых кнопок-логотипов на стартовом экране приложения. (Приложение 1)

**Экран режима «Собери слово»** - этот экран состоит из 2-х больших частей.

Первая часть – конструктор слова. В этой части пользователь может видеть картинку, в которой загадано слово, которое нужно собрать. Также даются 12 кнопок с краткими названиями элементов таблицы Менделеева на английском языке (K, H, O и т. д.). В связи с этим само слово также загадано на английском языке. При нажатии на кнопку – элемент добавляется в строку слова и отмечает кнопку как выбранную. Если на кнопку нажать второй раз – элемент удалится из строки. В самой строке слова изначально можно увидеть из скольких элементов состоит слово. Если загадка слишком сложна для пользователя – он может воспользоваться подсказкой – вывести ответ на русском языке.

Вторая часть выводится только при отгадывании слова. В ней выводится краткая сводка информации по каждому химическому элементу, который был использован в слове, а также само слово на английском и русском.

Переход между этими двумя частями производится посредством анимации, прописанной в XML файле и запускаемой Kotlin кодом. (Приложения 2-4)

**Экран результатов режима «Собери слово»** - этот экран пользователь может увидеть только тогда, когда разгадает все слова на данной сложности этого режима. Если он проходил эту сложность в первый раз – он откроет доступ к новой сложности. Он увидит всю информацию об открытом режиме на экране. Также на экране будет кнопка перехода к списку режимов. (Приложение 5)

**Экран режима «Викторина»** – на этом экране пользователь видит вопрос (задание), а также 3 варианта ответа. Пользователь должен выбрать только один верный вариант ответа. При нажатии на одну из кнопок ответа, пользователь увидит верно ли он ответил на вопрос – все неверные ответы будут выделены красным цветом. Через 2 секунды появится новый вопрос и новые варианты ответа, а все кнопки вернут изначальный цвет. Так пользователь должен ответить на 10 вопросов. После ответа на последний вопрос ему покажется экран результатов режима. (Приложение 6)

Ниже приведён код функции для получения задания из базы данных и заполнения кнопок ответов, а также блок-схема этой функции.

Листинг «Получение задания из базы» (Файл QMode1Activity.kt):

fun getTask(){ // Считывание задания из базы данных и заполнение //кнопок

// Включаем кнопки  
 T\_QuizAnsw1.isClickable = true  
 T\_QuizAnsw2.isClickable = true  
 T\_QuizAnsw3.isClickable = true  
  
 // Назначаем кнопкам текст  
 T\_QuizAnsw1.setTextColor(Color.parseColor("#00ff00"))  
 T\_QuizAnsw2.setTextColor(Color.parseColor("#00ff00"))  
 T\_QuizAnsw3.setTextColor(Color.parseColor("#00ff00"))  
  
 // Делаем запрос в базу данных  
 var cursor : Cursor? = db!!.query(TABLE\_QMODE1, null, null, null, null, null, null)  
  
 // Узнаём сколько вопросов есть  
 cursor!!.moveToLast()  
 val lastPos : Int = cursor.position  
  
 // Выбор задания  
 var needRand = true  
 while (needRand) {  
 var taskPos = (0..lastPos).random()  
 cursor.moveToPosition(taskPos)  
 if (taskPos !in taskList) {  
 // Получение задания  
 T\_QuizTask.text = cursor.getString(1)  
 // Получение ответов  
 val answList = cursor.getString(2).split("|").toTypedArray()  
 shuffleArray(answList) // Перемешивание массива  
 // Заполнение кнопок текстами ответов  
 T\_QuizAnsw1.text = answList[0]  
 T\_QuizAnsw2.text = answList[1]  
 T\_QuizAnsw3.text = answList[2]  
 // Получение строки верного ответа  
 rightAnsw = cursor.getString(3)  
 // Запись номеров вопросов, которые уже использованы  
 taskList[taskCount] = taskPos  
 taskCount++  
 needRand = false  
 }  
 }

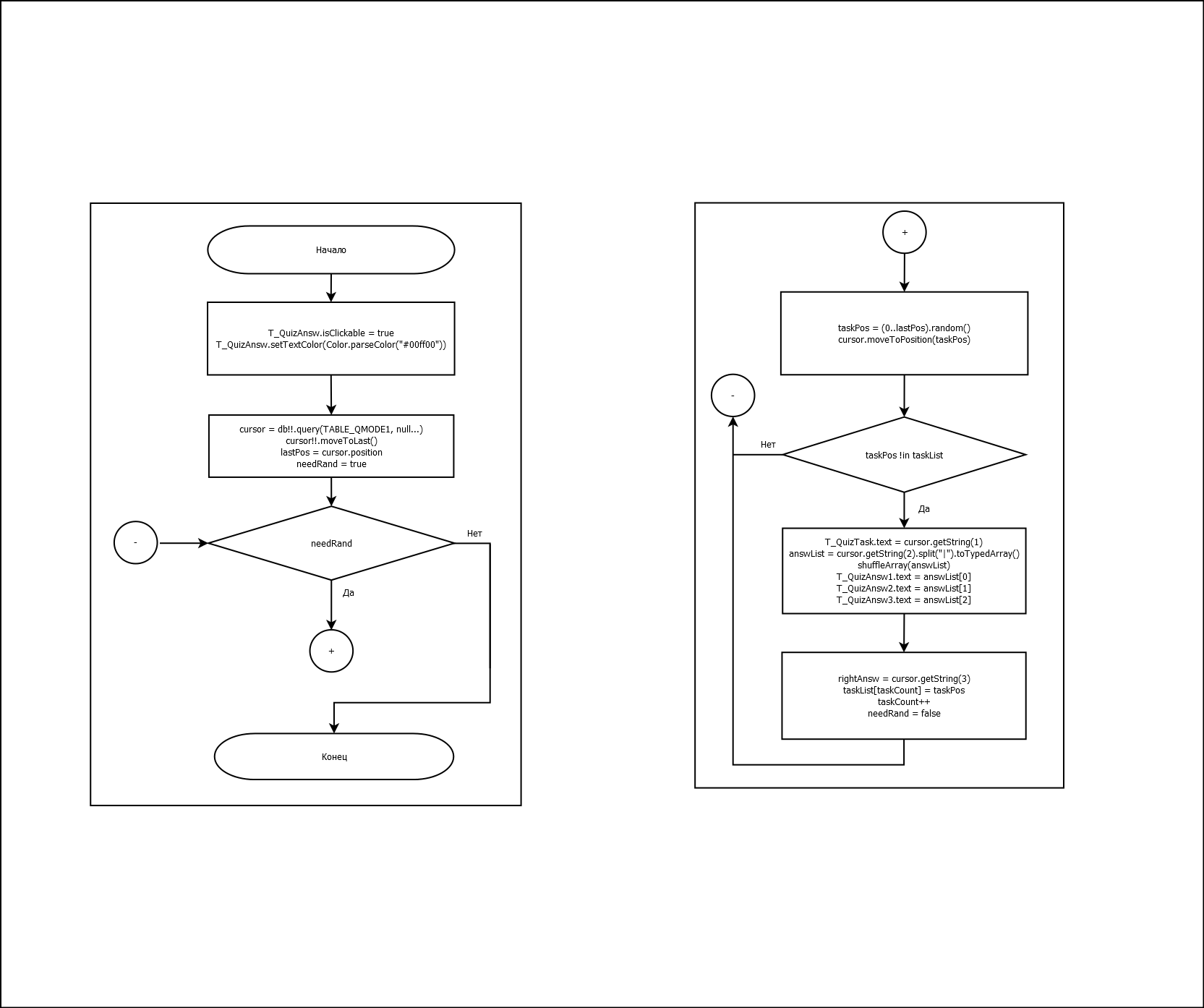


Рис. 2.1. Блок-схема алгоритма ч.1

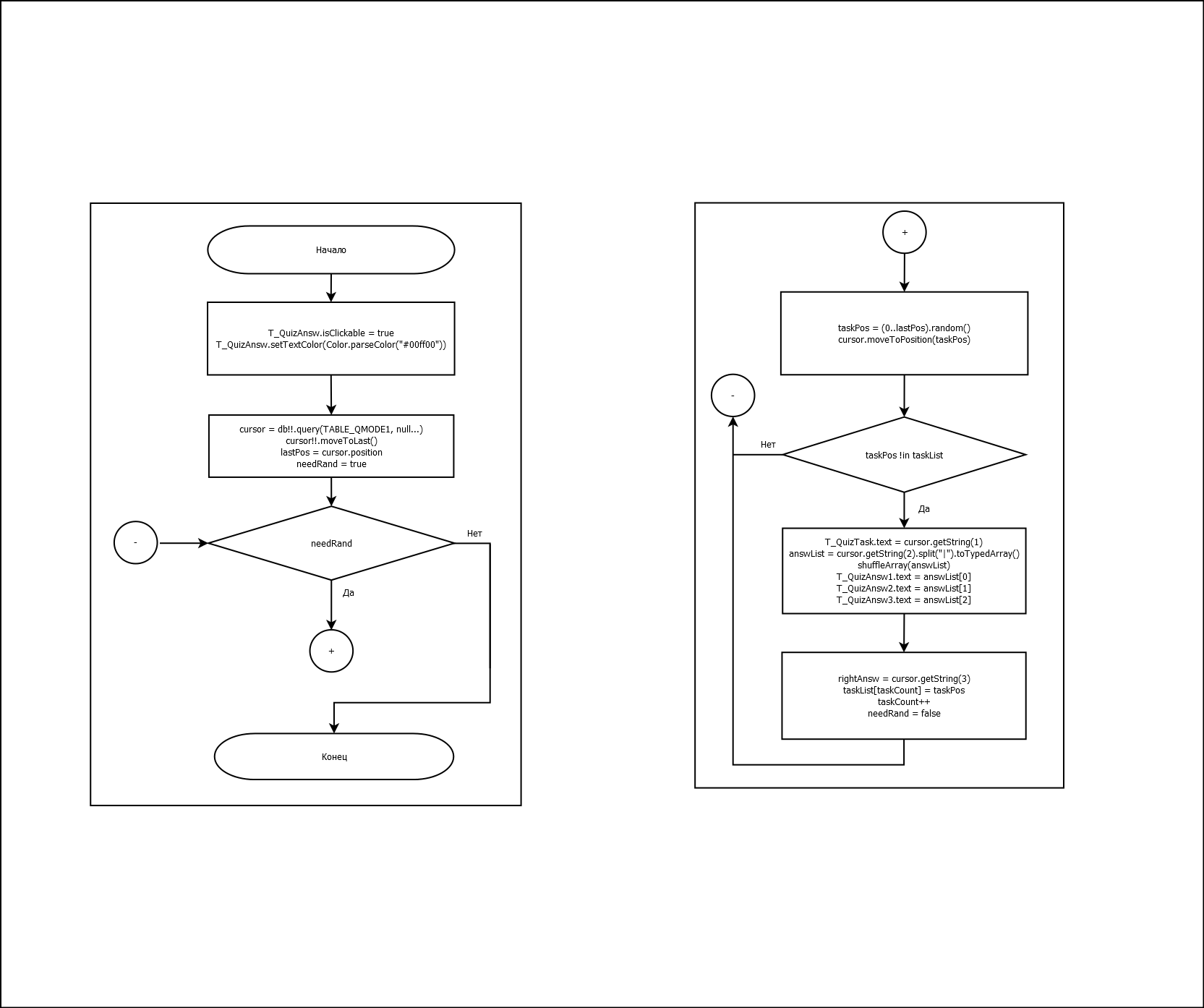


Рис. 2.2. Блок-схема алгоритма ч.2

**Экран результатов режимов викторины** – этот экран пользователь увидит после прохождения 10 заданий в одном из режимов викторины. Из каждого режима сюда передаётся количество верных ответов, данных пользователем и выводится в TextView в виде “X/10”, где X – количество верных ответов. Также на этом экране есть кнопка перехода в главное меню. (Приложение 7)

# **Организационная часть**

## **3.1 Порядок установки программного средства**

Приложение устанавливается обычными средствами операционной системы Android. Дополнительных разрешений не требует.

## **3.2 Руководство оператора**

**Назначение программы:**

Программа предназначена для изучения химии и английского языка, а также для развлечения пользователя.

#### 3.2.1 Минимальный состав технических и программных средств

В состав технических средств должен входить телефон, имеющий:

* ОС: Android 4.1 или выше.
* Оперативная память (RAM): 512 Мб;
* Жёсткий диск (доступного места на диске): 50 Мб;

#### 3.2.2 Требования к пользователю

Для успешной работы с программой пользователю необходимо:

* Уметь работать с системой Android
* Ознакомиться с данным руководством пользователя.

## **3.3 Запуск программы**

При запуске игры появляется главное меню с кнопкой «Играть»



Рис. 3.1. Главное меню.

## **3.4 Игровой процесс**

При нажатии на кнопку «Играть» открывается следующий экран – выбор режима игры.

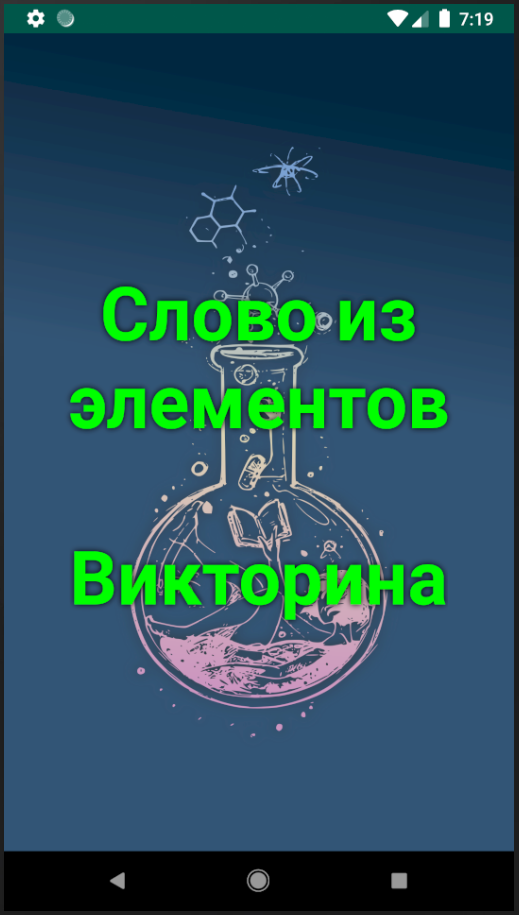


Рис. 3.2. Меню выбора режима

При нажатии на кнопку «Слово из элементов» открывается экран выбора сложности данного режима.



Рис. 3.3. Меню выбора сложности режима «Слово из элементов»

После выбора сложности открывается игровой экран.



Рис. 3.4. Экран режима «Слово из элементов»

Даются 12 кнопок с названиями элементов периодической таблицы Д.И.Менделеева. Из этих элементов пользователь должен собрать слово, загаданное на картинке выше. Это слово зашифровано на английском языке.

Если задание слишком сложное – пользователь может воспользоваться подсказкой – увидеть перевод слова на русский язык.

Когда пользователь соберёт слово, проиграется анимация перехода к результатам раунда. Теперь пользователь может видеть использованные в слове элементы периодической таблицы Д. И. Менделеева, краткую информацию о них, а также загаданное слово на английском и на русском языках.



Рис. 3.5. Результаты раунда

Далее пользователь может нажать на экран и перейти к следующему заданию. Затем действия повторяются. При желании пользователь может сменить режим.

Если пользователь пройдёт все задания на данной сложности, ему покажется экран результатов режима. Если он ранее не проходил данную сложность целиком – ему открывается новый уровень сложности и предлагается перейти в меню.

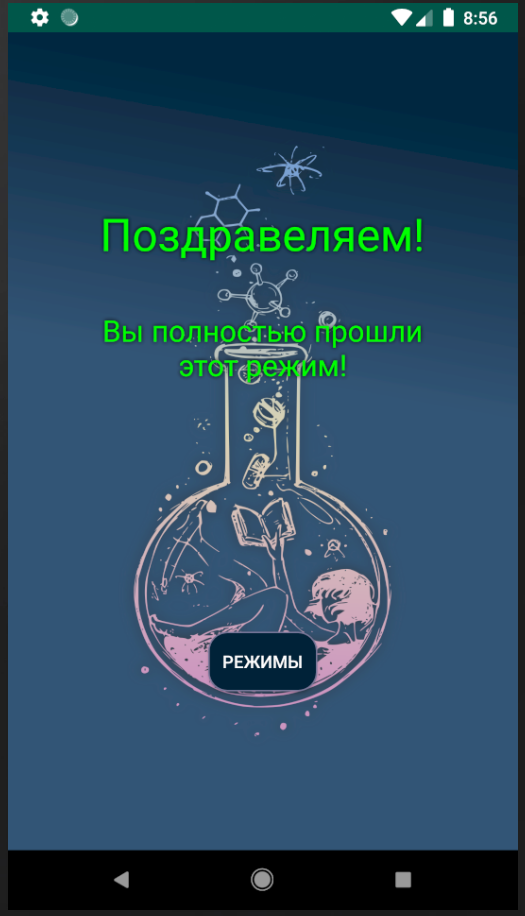


Рис. 3.6. Результаты режима «Собери слово из элементов»

Если на экране выбора режима пользователь выбирает режим викторины ему открывается экран выбора режима викторины.



Рис. 3.7. Меню выбора режима викторины

В каждом режиме викторины даётся 10 заданий одного типа. При ответе на каждый вопрос цветом выделяются верные и неверные ответы.

В режиме «Викторина» даётся вопрос из области химии и 3 варианта ответа, среди которых только 1 правильный.

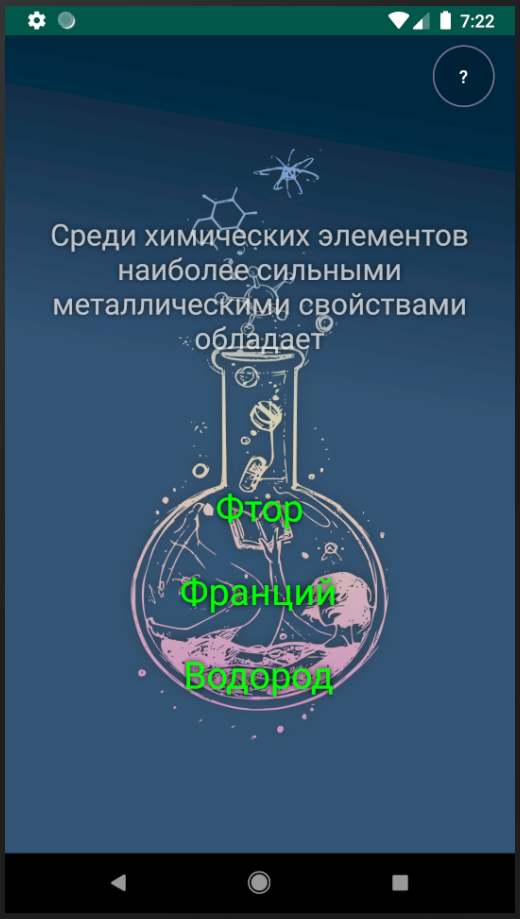


Рис. 3.8. Экран режима «Викторина»

В режиме «Исключить лишнее» даётся несколько химических терминов, из которых пользователь должен выбрать лишний.



Рис. 3.9. Экран режима «Исключить лишнее»

В режиме «Порядок» также даётся несколько химических терминов, связанных между собой. Пользователь должен расставить эти термины в соответствии со степенью их раскрытия в следующем.



Рис. 3.10. Экран режима «Порядок»

После прохождения всех 10 заданий пользователю открывается экран результатов викторины.

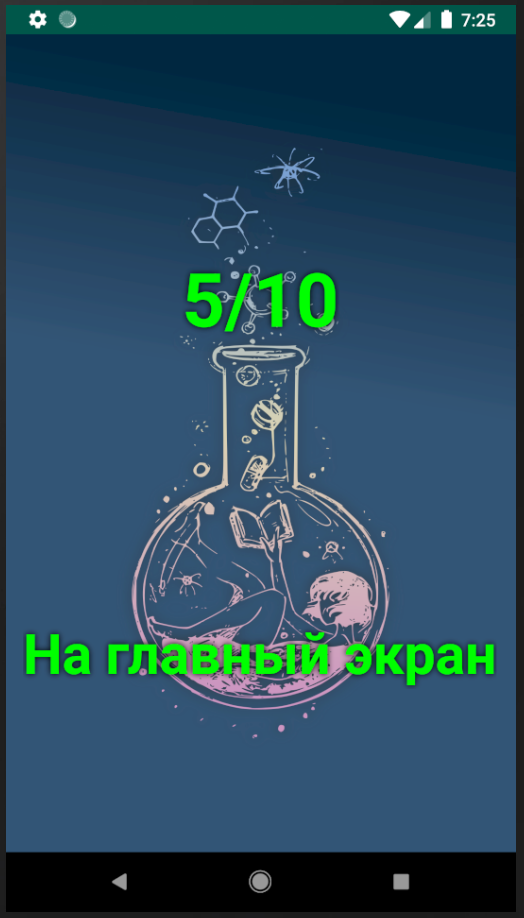


Рис. 3.11. Экран результатов викторины

Также на главном экране пользователь может увидеть логотипы разработчиков. При нажатии на эти логотипы – они откроются в полном размере и выведется ник разработчика.

Рис. 3.12. Экран логотипа разработчика

# **Заключение**

Данный курсовой проект был выполнен в полном соответствии с поставленным заданием. В ходе выполнения курсовой работы была разработана программа для химической викторины на Android с графическим интерфейсом.

Программа может быть использована в образовательных целях, помогая изучать химию и английский язык. Все вопросы и режимы связаны с химией, а в режиме «Слово из элементов» все слова загаданы на английском.

В качестве дальнейшего развития возможна доработка программы с целью добавления новых режимов, подсказок в режим «Викторина», организации таймера и ограничения времени решения на каждый из вопросов, возможности пропустить некоторые вопросы, чтобы вернуться к ним и ответить на них в случае наличия времени.

В результате выполнения данной курсовой работы, убедились в широких возможностях языка программирования Kotlin.

# **Литература и источники**

1. Kotlin for Android Developers А. Лейва 7-е издание 2019
2. Документация по Android API: https://developer.android.com/reference
3. Документация по языку Kotlin: https://kotlinlang.org/docs/reference/
4. Игра “История России Викторина”: https://play.google.com/store/apps/details?id=com.yugyd.russianhistoryquiz&hl=ru
5. Игра “4 фото 1 слово”: https://apkga.com/ru/com.originalgames.word1\_pics4\_en
6. Игра “Угадай лишнее слово”: https://play.google.com/store/apps/details?id=com.sayrex.wordup
7. Приложение “Duolingo”: https://play.google.com/store/apps/details?id=com.duolingo
8. Компания JetBrains: https://www.jetbrains.com/company/?fromMenu
9. ОС Android: https://ru.wikipedia.org/wiki/Android
10. Среда разработки Adroid Studio: https://ru.wikipedia.org/wiki/Android\_Studio
11. Статья о СУБД SQLite: https://habr.com/ru/post/149356/
12. Язык программирования Java: https://www.java.com/ru/about/
13. Youtube видеоурок базовый курс по программированию для Android: https://www.youtube.com/playlist?list=PL0lO\_mIqDDFW13-lP3IgK9lZoM1M-oPl4
14. Youtube видеоурок базовый курс по языку Kotlin: https://www.youtube.com/playlist?list=PLIU76b8Cjem4ZOt3tlWykUX1AjL9zE19t
15. Youtube видеоурок курс по разработке для Android на Kotlin: https://www.youtube.com/playlist?list=PLyfVjOYzujuj20Y-3kVhT3Zro9CrMNgNS

# **Приложение 1**

Листинг «Логотипы»:

package com.enoti.enotimendeleev  
// Класс экрана логотипа разработчика  
  
import android.content.pm.ActivityInfo  
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
import android.view.MotionEvent  
import kotlinx.android.synthetic.main.activity\_logo.\*  
  
class LogoActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_logo)  
 requestedOrientation = ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT  
 if (intent.getStringExtra("logo") == "enoti") { // Получение и высвет нужного логотипа  
 @Suppress("DEPRECATION")  
 Logo.setImageDrawable(resources.getDrawable(R.drawable.enotioriginal))  
 T\_Name.text = "Enoti"  
 } else if (intent.getStringExtra("logo") == "nejdan") {  
 @Suppress("DEPRECATION")  
 Logo.setImageDrawable(resources.getDrawable(R.drawable.nejdanoriginal))  
 T\_Name.text = "He}|{gaH4uk"  
 }  
 }  
  
 override fun onTouchEvent(event: MotionEvent): Boolean { // Нажатие на экран - выход из Activity  
 finish()  
 return true  
 }  
}

# **Приложение 2**

Листинг «Режим собери слово»:

package com.enoti.enotimendeleev  
// Класс экрана режима “Собери слово”  
  
import android.content.Intent  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.graphics.Color  
import android.graphics.Point  
import android.graphics.Typeface  
import android.os.Build  
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
import android.view.MotionEvent  
import android.view.View  
import android.view.animation.AnimationUtils  
import android.widget.Button  
import android.widget.TextView  
import kotlinx.android.synthetic.main.activity\_game.\*  
import java.util.\*  
  
  
class GameActivity : AppCompatActivity() {  
  
 // Константы  
 // Информация о таблице PeriodicTable в базе данных  
 private val TABLE\_PERIODIC = "PeriodicTable"  
 private val KEY\_ID\_P = "ID"  
 private val KEY\_SHORTNAME = "ShortName"  
 private val KEY\_FULLNAME = "FullName"  
 private val KEY\_RUNAME = "RuName"  
  
 // Информация о таблице Tasks в базе данных  
 private val TABLE\_TASKS = "Tasks"  
 private val KEY\_ID = "\_id"  
 private val KEY\_TYPE = "Type"  
 private val KEY\_PICTURE = "Picture"  
 private val KEY\_HINT = "Hint"  
 private val KEY\_ANSWERS = "Answers"  
 private val KEY\_FULLANSW = "FullAnsw"  
 private val KEY\_ANSWRU = "AnswRu"  
  
 // Кол-во вопросов разных режимов  
 private val questCountEasy = 91  
 private val questCountGeography = 17  
 private val questCountHard = 24  
  
 // Флаг - все вопросы закончились  
 var allTasksEnded = false  
  
 // База данных  
 var db : SQLiteDatabase? = null  
  
 // Массив Boolean всех заданий. true - задание было использовано  
 var allTasks : BooleanArray = BooleanArray(0)  
  
 // Текущие элементы и их номера  
 var curElements = Vector<String?> (0)  
 var curElementNums = Vector<Int> (0)  
  
 // Информация о задании (Файл картинки, подсказка, ответы, полный ответ, ответ на русском)  
 var Picture = ""  
 var Hint = ""  
 var Answers : List<String?> = listOf()  
 var FullAnsw = ""  
 var AnswRu = ""  
  
 // Номер режима  
 var Mode = 0  
  
 // Главный вариант ответа. На случай, если можно ответить разными способами  
 var mainAnswer : String? = ""  
  
 // Флаг ответил ли пользователь на вопрос  
 var answered = false  
  
 // Текущее задание  
 var curTaskPos = -1  
  
 // Событие создания экрана  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_game)  
 // Получение выбранного ранее режима  
 @Suppress("RECEIVER\_NULLABILITY\_MISMATCH\_BASED\_ON\_JAVA\_ANNOTATIONS")  
 Mode = intent.extras.getInt("mode")  
  
 var SP = getSharedPreferences("com.enoti.enotimendeleev", MODE\_PRIVATE)  
  
 // Получаем массив правильных ответов в зависимости от // режима  
 when(Mode) {  
 1 -> {  
 if (!SP.getBoolean("easyModeCompleted", false)) {  
 allTasks = BooleanArray(questCountEasy)  
 for (i in (0..questCountEasy)) {  
 if (SP.getBoolean("easyQuest\_$i", false)) {  
 allTasks[i] = true  
 }  
 }  
 }  
 }  
 2 -> {  
 if (!SP.getBoolean("geographyModeCompleted", false)) {  
 allTasks = BooleanArray(questCountGeography)  
 for (i in (0..questCountGeography)) {  
 if (SP.getBoolean("geographyQuest\_$i", false)) {  
 allTasks[i] = true  
 }  
 }  
 }  
 }  
 3 -> {  
 if (!SP.getBoolean("hardModeCompleted", false)) {  
 allTasks = BooleanArray(questCountHard)  
 for (i in (0..questCountHard)) {  
 if (SP.getBoolean("hardQuest\_$i", false)) {  
 allTasks[i] = true  
 }  
 }  
 }  
 }  
 }  
  
 // Открытие базы  
 db = SQLiteDatabase.openDatabase(getDatabasePath("Data").path, null, SQLiteDatabase.OPEN\_READONLY)  
  
 // Получение размеров экрана  
 var size = Point(0, 0)  
 windowManager.defaultDisplay.getSize(size)  
 // Выставление ширины экрана для полного сдвижения после // ответа  
 QuestResConstraint.translationX = size.x.toFloat()  
  
 getTask()  
  
 addListenerOnButton()  
 }

override fun onBackPressed() { } // Отключение возвращения //на предыдущий экран  
  
 fun addListenerOnButton() { // Назначение функции, которая //будет вызываться на созданной кнопке  
 Var1.setOnClickListener {  
 ElementButtonClicked(1, Var1)  
 }  
 Var2.setOnClickListener {  
 ElementButtonClicked(2, Var2)  
 }  
 Var3.setOnClickListener {  
 ElementButtonClicked(3, Var3)  
 }  
 Var4.setOnClickListener {  
 ElementButtonClicked(4, Var4)  
 }  
 Var5.setOnClickListener {  
 ElementButtonClicked(5, Var5)  
 }  
 Var6.setOnClickListener {  
 ElementButtonClicked(6, Var6)  
 }  
 Var7.setOnClickListener {  
 ElementButtonClicked(7, Var7)  
 }  
 Var8.setOnClickListener {  
 ElementButtonClicked(8, Var8)  
 }  
 Var9.setOnClickListener {  
 ElementButtonClicked(9, Var9)  
 }  
 Var10.setOnClickListener {  
 ElementButtonClicked(10, Var10)  
 }  
 Var11.setOnClickListener {  
 ElementButtonClicked(11, Var11)  
 }  
 Var12.setOnClickListener {  
 ElementButtonClicked(12 , Var12)  
 }  
 HintBut.setOnClickListener {  
 // Показать рекламу :)  
 T\_Hint.visibility = View.VISIBLE  
 }  
 }

fun shuffleArray(ar: IntArray) {  
 for (i in ar.size - 1 downTo 1) {  
 val index = (0..11).random()  
 // Простой обмен значений  
 val a = ar[index]  
 ar[index] = ar[i]  
 ar[i] = a  
 }  
 }  
  
 fun ElementButtonClicked (element : Int, button : Button) { // Функция для всех кнопок. element - номер элемента таблицы  
 // Добавление/Удаление из строки ответа  
 if (!button.isSelected) {  
 curElements.addElement("${button.text}")  
 curElementNums.addElement(element)  
 button.isSelected = true  
 button.refreshDrawableState()  
 }  
 else {  
 var index = curElementNums.indexOf(element)  
 curElements.removeAt(index)  
 curElementNums.removeAt(index)  
 button.isSelected = false  
 button.refreshDrawableState()  
 }  
  
 // Обновление строки ответа  
 UpdateString()  
  
 // Проверка на правильность ответа  
 if (T\_Answ.text.toString().replace("-", "").replace("\_", "").toLowerCase() == FullAnsw.toLowerCase()) {  
 // Заполнение страницы результата текущего задания // данными об элементах и полном ответе  
 T\_ResAnsw.text = "$FullAnsw - $AnswRu"  
 var Answer = T\_Answ.text.toString().replace("\_", "").split("-")  
 LinearElems.removeAllViews()  
 for (i in Answer) {  
 val selection = "${KEY\_SHORTNAME} = '${i}'"  
 var cursor : Cursor? = db!!.query(TABLE\_PERIODIC, null, selection, null, null, null, null)  
 var newText = TextView(this)  
 cursor!!.moveToFirst()  
 newText.typeface = Typeface.SANS\_SERIF  
 newText.setBackgroundResource(R.drawable.tables)  
 newText.setTextColor(Color.parseColor("#aaaa00"))  
 newText.textSize = 22F  
 newText.setShadowLayer(10.toFloat(), 0.toFloat(), 0.toFloat(), Color.BLACK)  
 newText.text = "${cursor.getString(cursor.getColumnIndex(KEY\_ID\_P))} - ${i} - ${cursor.getString(cursor.getColumnIndex(KEY\_FULLNAME))} - ${cursor.getString(cursor.getColumnIndex(KEY\_RUNAME))}"  
 LinearElems.addView(newText)  
 cursor.close()  
 }

}  
 }  
  
 fun UpdateString () { // Обновление строки ответа для показа // того, сколько осталось дописать элементов.  
 var str = ""  
 if (curElements.count() > 0) {  
 var curLetters = 0  
 for (i in curElements) {  
 str += "-$i"  
 curLetters++  
 }  
 while (curLetters < mainAnswer!!.split(" ").count()) {  
 str += "-\_"  
 curLetters++  
 }  
 }  
 else {  
 for(i in (1..mainAnswer!!.split(" ").count())) {  
 str += "-\_"  
 }  
 }  
 str = str.removeRange(0..0)  
 T\_Answ.text = str  
 }  
}

# **Приложение 3**

Листинг «Результаты раунда»:

// Анимация перехода на страницу результата  
var anim = AnimationUtils.loadAnimation(applicationContext, R.anim.toleft)  
QuestQonstraint.startAnimation(anim)  
QuestResConstraint.startAnimation(anim)  
answered = true  
allTasks[curTaskPos] = true  
  
// Проверка на полное прохождение режима, если он не был пройден // ранее  
var SP = getSharedPreferences("com.enoti.enotimendeleev", MODE\_PRIVATE)  
var tasksEnded = true  
var count = 0  
var mode = ""  
when(Mode) {  
 1 -> {  
 count = questCountEasy-1  
 mode = "easy"  
 }  
 2 -> {  
 count = questCountGeography-1  
 mode = "geography"  
 }  
 3 -> {  
 count = questCountHard-1  
 mode = "hard"  
 }  
}  
for (i in (0..count)) {  
 tasksEnded = allTasks[i]  
 if (!tasksEnded) {  
 break  
 }  
}  
  
// Запись информации о том, что этот вопрос задавался  
SP.edit().putBoolean("${mode}Quest\_$curTaskPos", true).apply()  
// Если режим был пройден - ставим флаг, что все вопросы //закончились  
if (tasksEnded) {  
 allTasksEnded = true  
}

# **Приложение 4**

Листинг «Обратная анимация»:

// Событие нажатия на экран  
override fun onTouchEvent(event: MotionEvent?): Boolean {   
 // Если верный ответ получен  
 if (!answered) return false  
 // Обнуляем переменную  
 answered = false  
 // Если вопросы не закончились - ищем новый вопрос и //возвращаем все поля к стандартному виду  
 if (!allTasksEnded) {  
 T\_Answ.text = ""  
 curElements.clear()  
 curElementNums.clear()  
 getTask()  
 Var1.isSelected = false  
 Var2.isSelected = false  
 Var3.isSelected = false  
 Var4.isSelected = false  
 Var5.isSelected = false  
 Var6.isSelected = false  
 Var7.isSelected = false  
 Var8.isSelected = false  
 Var9.isSelected = false  
 Var10.isSelected = false  
 Var11.isSelected = false  
 Var12.isSelected = false  
  
 // Обратная анимация (Возвращение к странице вопроса)  
 var anim = AnimationUtils.loadAnimation(applicationContext, R.anim.toright)  
 QuestQonstraint.startAnimation(anim)  
 QuestResConstraint.startAnimation(anim)  
 }  
 else { // Если вопросы закончились - показываем окно //результатов  
 var resultIntent = Intent(this, ResultActivity::class.java)  
 resultIntent.putExtra("mode", Mode)  
 resultIntent.flags = Intent.FLAG\_ACTIVITY\_CLEAR\_TOP  
 startActivity(resultIntent)  
 finish()  
 }  
  
 return super.onTouchEvent(event)  
}

# **Приложение 5**

Листинг «Результаты режима»:

package com.enoti.enotimendeleev  
// Класс экрана с результатами режима “Собери слово”  
  
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
import android.view.View  
import kotlinx.android.synthetic.main.activity\_result.\*  
  
class ResultActivity : AppCompatActivity() {  
  
 val questCountEasy = 91  
 val questCountGeography = 17  
 val questCountHard = 24  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
// Тут же надо где-то очищать наш массив ответов, чтобы всё было // false  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_result)  
  
 @Suppress("RECEIVER\_NULLABILITY\_MISMATCH\_BASED\_ON\_JAVA\_ANNOTATIONS")  
 val Mode = intent.extras.getInt("mode")  
  
 var SP = getSharedPreferences("com.enoti.enotimendeleev", MODE\_PRIVATE)  
 var unlockedMode = ""  
 var mode = ""  
 var count = 0  
 var unlocked = ""  
  
 when(Mode) {  
 1 -> {  
 if (!SP.getBoolean("geographyModeUnlocked", false)) {  
 unlockedMode = "География"  
 mode = "easy"  
 count = questCountEasy  
 unlocked = "geography"  
 }  
 }  
 2 -> {  
 if (!SP.getBoolean("hardModeUnlocked", false)) {  
 unlockedMode = "Сложный"  
 mode = "geography"  
 unlocked = "hard"  
 count = questCountGeography  
 }  
 }  
 3 -> {  
 count = questCountHard  
 mode = "hard"  
 }  
 }  
  
 if (unlockedMode.isNullOrEmpty()) {  
 T\_ModeUnlocked.visibility = View.INVISIBLE  
 T\_UnlockedMode.visibility = View.INVISIBLE  
 }  
 else {  
 T\_UnlockedMode.text = unlockedMode  
 }  
  
 // Очистка массива и Complete переменной в SP  
  
 SP.edit().putBoolean("${mode}ModeCompleted", false).commit()  
 SP.edit().putBoolean("${unlocked}ModeUnlocked", true).commit()  
  
 for (i in (0..count)) {  
 SP.edit().putBoolean("${mode}Quest\_${i}", false).commit()  
 }  
  
 addListenerOnButton()  
 }  
  
 fun addListenerOnButton() {  
 ModesBut.setOnClickListener {  
 finish()  
 }  
 }  
}

# **Приложение 6**

Листинг «Режим викторина»:

package com.enoti.enotimendeleev  
// Класс экрана режима “викторина”  
  
import android.content.Intent  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.graphics.Color  
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
import android.os.CountDownTimer  
import kotlinx.android.synthetic.main.activity\_qmode1.\*  
  
class QMode1Activity : AppCompatActivity() {  
  
 // Константы  
 // Информация о таблице QMode1 в базе данных  
 private val TABLE\_QMODE1 = "QMode1"  
  
 var rightAnsw = ""  
 var taskList = IntArray(10) {-1}  
 var taskCount = 0  
 var rightAnswCount = 0  
  
 var db : SQLiteDatabase? = null  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_qmode1)  
 addListenerOnButton()  
  
 // Открываем базу данных  
 db = SQLiteDatabase.openDatabase(getDatabasePath("Data").path, null, SQLiteDatabase.OPEN\_READONLY)  
  
 getTask()  
 }  
  
 override fun onBackPressed() { }  
  
 fun getTask() { // Считывание задания из базы данных и //заполнение кнопок  
 // Включаем кнопки  
 T\_QuizAnsw1.isClickable = true  
 T\_QuizAnsw2.isClickable = true  
 T\_QuizAnsw3.isClickable = true  
  
 // Назначаем кнопкам текст  
 T\_QuizAnsw1.setTextColor(Color.parseColor("#00ff00"))  
 T\_QuizAnsw2.setTextColor(Color.parseColor("#00ff00"))  
 T\_QuizAnsw3.setTextColor(Color.parseColor("#00ff00"))   
  
 // Делаем запрос в базу данных  
 var cursor : Cursor? = db!!.query(TABLE\_QMODE1, null, null, null, null, null, null)  
  
 // Узнаём сколько вопросов есть  
 cursor!!.moveToLast()  
 val lastPos : Int = cursor.position  
  
 // Выбор задания  
 var needRand = true  
 while (needRand) {  
 var taskPos = (0..lastPos).random()  
 cursor.moveToPosition(taskPos)  
 if (taskPos !in taskList) {  
 // Получение задания  
 T\_QuizTask.text = cursor.getString(1)  
 // Получение ответов  
 val answList = cursor.getString(2).split("|").toTypedArray()  
 shuffleArray(answList) // Перемешивание массива  
 // Заполнение кнопок текстами ответов  
 T\_QuizAnsw1.text = answList[0]  
 T\_QuizAnsw2.text = answList[1]  
 T\_QuizAnsw3.text = answList[2]  
 // Получение строки верного ответа  
 rightAnsw = cursor.getString(3)  
 // Запись номеров вопросов, которые уже использованы  
 taskList[taskCount] = taskPos  
 taskCount++  
 needRand = false  
 }  
 }  
 }  
  
 fun addListenerOnButton() { // Назначение функций, которые // будут вызываться на кнопках  
 T\_QuizAnsw1.setOnClickListener {  
 onAnswButtonClick()  
 if (rightAnsw == T\_QuizAnsw1.text.toString()) { // Если ответ верный - прибавляем 1 к счётчику  
 rightAnswCount++  
 }  
 }  
 T\_QuizAnsw2.setOnClickListener {  
 onAnswButtonClick()  
 if (rightAnsw == T\_QuizAnsw2.text.toString()) { // Если ответ верный - прибавляем 1 к счётчику  
 rightAnswCount++  
 }  
 }  
 T\_QuizAnsw3.setOnClickListener {  
 onAnswButtonClick()  
 if (rightAnsw == T\_QuizAnsw3.text.toString()) { // Если ответ верный - прибавляем 1 к счётчику  
 rightAnswCount++  
 }  
 }  
 }  
  
 fun onAnswButtonClick() { // Общая часть для всех кнопок  
 // Отключаем кнопки, чтобы нельзя было нажать их дважды  
 T\_QuizAnsw1.isClickable = false  
 T\_QuizAnsw2.isClickable = false  
 T\_QuizAnsw3.isClickable = false  
 // Показываем цветом верный ли ответ  
 showRightAnsw()  
 // Таймер, по окончании которого даётся новое задание, либо показываются результаты  
 object: CountDownTimer(1000, 1) {  
 override fun onTick(p0: Long) { }  
 override fun onFinish () {  
 if (taskCount == 10) {  
 // Переход на окно результатов  
 var ResultScreenIntent = Intent(this@QMode1Activity, QuizResultActivity::class.java)  
 ResultScreenIntent.putExtra("rightAnswersCount", rightAnswCount)  
 ResultScreenIntent.flags = Intent.FLAG\_ACTIVITY\_CLEAR\_TOP  
 startActivity(ResultScreenIntent)  
 finish()  
 }  
 else {  
 // Поиск нового вопроса  
 getTask()  
 }  
 }  
 }.start()  
 }  
  
 fun showRightAnsw() { // Отображение цветом верного ответа  
 if(rightAnsw == T\_QuizAnsw1.text.toString()) {  
 T\_QuizAnsw1.setTextColor(Color.parseColor("#ffff00"))  
 T\_QuizAnsw2.setTextColor(Color.parseColor("#ff0000"))  
 T\_QuizAnsw3.setTextColor(Color.parseColor("#ff0000"))  
 }  
 else if (rightAnsw == T\_QuizAnsw2.text.toString()) {  
 T\_QuizAnsw1.setTextColor(Color.parseColor("#ff0000"))  
 T\_QuizAnsw2.setTextColor(Color.parseColor("#ffff00"))  
 T\_QuizAnsw3.setTextColor(Color.parseColor("#ff0000"))  
 }  
 else {  
 T\_QuizAnsw1.setTextColor(Color.parseColor("#ff0000"))  
 T\_QuizAnsw2.setTextColor(Color.parseColor("#ff0000"))  
 T\_QuizAnsw3.setTextColor(Color.parseColor("#ffff00"))  
 }  
  
 }

// Функция перемешивания массива  
 fun shuffleArray(ar: Array<String>) {  
 for (i in ar.size - 1 downTo 1) {  
 val index = (0 until ar.size).random()  
 val a = ar[index]  
 ar[index] = ar[i]  
 ar[i] = a  
 }  
 }  
}

# **Приложение 7**

Листинг «Экран результатов викторины»:

package com.enoti.enotimendeleev  
// Класс экрана с результатами викторины  
  
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle  
import kotlinx.android.synthetic.main.activity\_quiz\_result.\*  
  
class QuizResultActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_quiz\_result)  
 @Suppress("RECEIVER\_NULLABILITY\_MISMATCH\_BASED\_ON\_JAVA\_ANNOTATIONS")  
 RightQuizCounter.text = "${intent.extras.getInt("rightAnswersCount")}/10"  
  
 addListenerOnButton()  
 }  
  
 fun addListenerOnButton() {  
 MainMenuButton.setOnClickListener {  
 finish()  
 }  
 }  
}

# **Приложение 8**

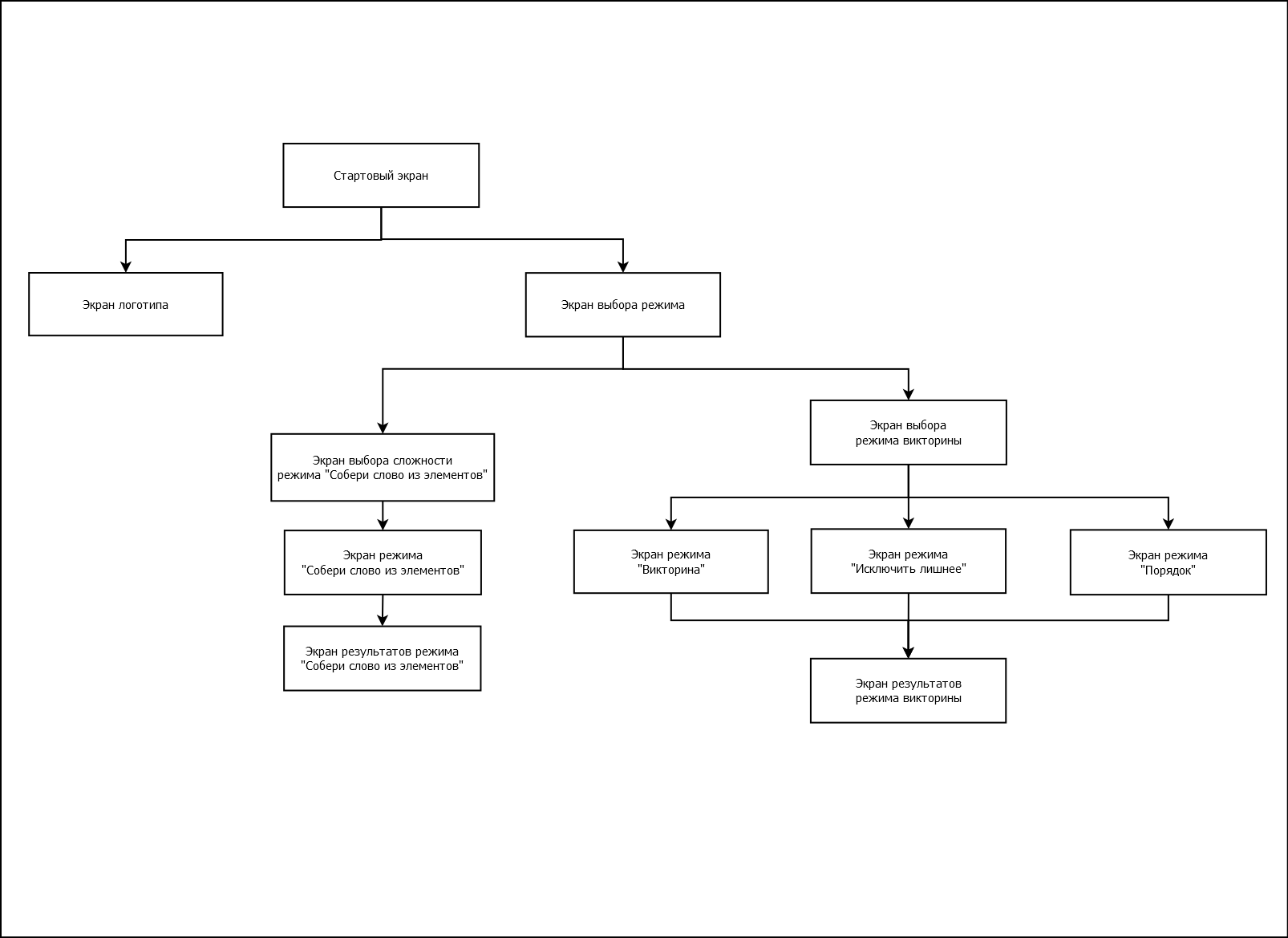


Рис. 1. Общая схема работы программы