**ВВЕДЕНИЕ**

Целью данного проекта является разработать мобильного приложения «Учим английские слова».

Мобильное приложение - программное изделие, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных устройствах.

Мобильного приложения «Учим английские слова» представляет собой направление в области образования. Современный мир характеризуется повышенным интересом к изучению иностранных языков, а мобильные приложения стали незаменимым инструментом в этом процессе благодаря своей доступности и удобству использования. Проект ориентирован на разработку приложения, которое поможет пользователям улучшить свой уровень владения английским языком.

Для достижения цели необходимо выполнить ряд задач:

* Анализ требований
* Разработка теоретического материала
* Проектирование структуры базы данных приложения
* Проектирование интерфейса
* Разработка функциональности приложения
* Тестирование и отладка

Среда разработки используемая в процессе создания программного продукта – «Android Studio». «Android Studio» — интегрированная среда разработки для работы с платформой «Android», анонсированная 16 мая 2013 года на конференции «Google I/O». В последней версии «Android Studio» поддерживается «Android» 4.1 и выше.

Язык, используемый для разработки – «Kotlin». «Kotlin» — статически типизированный, объектно-ориентированный язык программирования, работающий на основе «Java Virtual Machine» и разрабатываемый компанией «JetBrains».

Результатом курсового проекта станет разработанная информационно-поисковая система, предоставляющая теоретический материал, задания разных типов для закрепления пройденного материала и статистка прохождения заданий.

**1 СИМВОЛЬНЫЙ И СТРОКОВЫЙ ТИП ДАННЫХ**

Символьный тип — тип данных, предназначенный для хранения одного символа (управляющего или печатного) в определённой кодировке. Может являться как однобайтовым (для стандартной таблицы символов), так и многобайтовым (к примеру, для Юникода).

Строковый тип — тип данных, значениями которого является произвольная последовательность символов алфавита. Каждая переменная такого типа может быть представлена фиксированным количеством байтов либо иметь произвольную длину.

Некоторые языки программирования накладывают ограничения на максимальную длину строки, но в большинстве языков подобные ограничения отсутствуют. При использовании «Unicode» каждый символ строкового типа может требовать двух или четырёх байтов для своего представления.

Основные проблемы в машинном представлении строкового типа:

* строки могут иметь достаточно существенный размер (до нескольких десятков мегабайтов);
* изменяющийся со временем размер — возникают трудности с добавлением и удалением символов.

В представлении строк в памяти компьютера существует два принципиально разных подхода: представление массивом символов, метод «завершающего байта», в виде списка.

При представлении массивом символов размер массива хранится в отдельной (служебной) области. От названия языка «[Pascal](https://ru.wikipedia.org/wiki/Pascal" \o "Pascal)», где этот метод был впервые реализован, данный метод получил название «Pascal strings».

Более оптимизированным вариантом этого метода является т. н. формат «c-addr u». В отличие от «Pascal strings», здесь размер массива хранится не совместно со строковыми данными, а является частью [указателя](https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) на строку.

Преимущества представления в виде массива символов:

* программа в каждый момент времени содержит сведения о размере строки, поэтому операции добавления символов в конец, копирования строки и собственно получения размера строки выполняются достаточно быстро;
* строка может содержать любые данные;
* возможно на программном уровне следить за выходом за границы строки при её обработке;
* возможно быстрое выполнение операции вида «взятие N-ого символа с конца строки».

Недостатки представления в виде массива символов:

* проблемы с хранением и обработкой символов произвольной длины;
* увеличение затрат на хранение строк — значение «длина строки» также занимает место и в случае большого количества строк маленького размера может существенно увеличить требования алгоритма к оперативной памяти;
* ограничение максимального размера строки. В современных языках программирования это ограничение скорее теоретическое, так как обычно размер строки хранится в 32-битовом поле, что даёт максимальный размер строки в 4 294 967 295 байт (4 гигабайта);
* при использовании [алфавита](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) с переменным размером символа (например, «[UTF-8](https://ru.wikipedia.org/wiki/UTF-8)»), в размере хранится не количество символов, а именно размер строки в байтах, поэтому количество символов необходимо считать отдельно.

Метод «завершающего байта» заключается в том что одно из возможных значений символов алфавита (как правило, это символ с кодом 0) выбирается в качестве признака конца строки, и строка хранится как последовательность байтов от начала до конца. Есть системы, в которых в качестве признака конца строки используется не символ 0, а байт «0xFF» (255) или код символа «$».

Метод имеет три названия — «ASCIIZ», «C-strings» и метод нуль-терминированных строк.

Преимущества метода «завершающего байта»:

* отсутствие дополнительной служебной информации о строке (кроме завершающего байта);
* возможность представления строки без создания отдельного типа данных;
* отсутствие ограничения на максимальный размер строки;
* экономное использование памяти;
* простота получения суффикса строки;
* простота передачи строк в функции (передаётся указатель на первый символ);

Недостатки метода «завершающего байта»:

* долгое выполнение операций получения длины и конкатенации строк;
* отсутствие средств контроля за выходом за пределы строки, в случае повреждения завершающего байта возможность повреждения больших областей памяти, что может привести к непредсказуемым последствиям — потере данных, краху программы и даже всей системы;
* невозможность использовать символ завершающего байта в качестве элемента строки.
* невозможность использовать некоторые кодировки с размером символа в несколько байт (например, «UTF-16»), так как во многих таких символах, например «Ā» (0x0100), один из байтов равен нулю (в то же время, кодировка «UTF-8» свободна от этого недостатка).

Метод представление в виде списка делает язык более «теоретически элегантным» за счёт соблюдения ортогональности в системе типов, но приносит существенные потери быстродействия.

Процедуры и функции:

Простейшие операции со строками:

* получение символа по индексу — в большинстве языков это тривиальная операция;
* конкатенация строк.

Производные операции:

* получение подстроки по индексам начала и конца;
* проверка вхождения одной строки в другую (поиск подстроки в строке);
* проверка на совпадение строк (с учётом или без учёта регистра символов);
* получение длины строки;
* замена подстроки в строке.

Операции при трактовке строк как списков:

* свёртка;
* отображение одного списка на другой;
* фильтрация списка по критерию.

Продвинутые операции:

* нахождение минимальной надстроки, содержащей все указанные строки;
* поиск в двух массивах строк совпадающих последовательностей.

До появления стандарта Юникод в 1991 году, один символ обычно кодировался одним байтом из 8 двоичных битов или меньше — 7-битные, 6-битные. 8-битые кодировки позволяли представлять 256 возможных значений. Однако для полноценного представления символов алфавитов нескольких языков 256 символов недостаточно. Для решения этой проблемы применялись разные подходы.

Переключение языка управляющими кодами. Метод не стандартизирован и лишает текст самостоятельности (то есть последовательность символов без управляющего кода в начале теряет смысл);

Использование двух или более байт для представления каждого символа («UTF-16», «UTF-32»). Главным недостатком этого метода является потеря совместимости с предыдущими библиотеками для работы с текстом при представлении строки как «ASCIIZ». Например, концом строки должен считаться уже не байт со значением 0, а два или четыре подряд идущих нулевых байта, в то время как одиночный байт «0» может встречаться в середине строки, что сбивает библиотек.

Использование кодировки с переменным размером символа. Например, в «UTF-8» часть символов представляется одним байтом, часть двумя, тремя или четырьмя. Этот метод позволяет сохранить частичную совместимость со старыми библиотеками, но приводит к невозможности прямой адресации символа в памяти по номеру его позиции в строке.

* 1. **ТРЕБОВАНИЕ К ПРОГРАММНОМУ ПРОДУКТУ**

Программное средство должно обрабатывать базу данных, иметь возможность создания новой записи или просмотра существующих записей.

Функционал программного продукта должен включать в себя функции:

* «CRUD»: «Create» - создания записи, «Read» - чтения записи, «Update» - обновления записи, «Delete» - удаления записи
* Поиска
* Сортировки
* Представление теоретического материала
* Выполнения заданий разных типов для закрепления пройденного материала
* Отслеживание статистки прохождения заданий

Программа должна иметь 2 формы:

* Приветственная с названием курсового проекта и автором
* Форма обработки файла

В программе должна осуществляться валидация введённых пользователем данных

База данных содержит таблицы:

* «words\_table» содержащая следующие поля:
  1. «ID» - уникальный не пустой целочисленный автоинкрементный основной столбец характеризующий номер строки в базе данных
  2. «ENGLISH\_NAME» ­­­­­­­­­­- уникальный не пустой строковый столбец, хранящий английское представление слова
  3. «RUSSIAN\_NAME» ­­­­­­­­­­- уникальный не пустой строковый столбец, хранящий русское представление слова
  4. «WORD\_DEFENITION» ­­­­­­­­­­- строковый столбец, хранящий английское представление слова
* «test find translation table», «test find word table», «test true false table» содержащие следующие поля:
  1. «ID» - целочисленное основное не пустое уникальное автоинкрементное поле
  2. «RESULT» - не пустое текстовое поле, содержащее результаты прохождения теста
  3. «RESULT\_PROCENT» - не пустое целочисленное поле, содержащее результаты прохождения теста в виде процентного соотношения
  4. «TIME\_START» - строковый столбец, хранящий начало теста
  5. «TIME\_FINISH» - строковый столбец хранящий конец теста
  6. «PROGRESS» - строковый столбец хранящий прогресс с последнего прохождения теста
  7. **ОПИСАНИЕ АЛГОРИТМОВ И ТЕХНОЛОГИЙ РЕШЕНИЯ**

Для создания записи необходимо нажать на кнопку и форма измениться на форму для добавления записи, после этого необходимо будет ввести данные, нажать на кнопку, при этом данные из изменяемых полей проверяются на содержание в базе данных, в случае если они присутствуют пользователь получит сообщение об этом, в случае если поля пустые пользователь получит сообщение об ошибке, в ином случае поля будут добавлены в новую запись в базе данных и пользователь получит сообщение о успешном, а запись будет добавлена в базу данных и форма измениться на предыдущую.

Для чтения записи необходимо выбрать соответствующую запись в списке и изменяемые поля заполнятся содержанием записи.

Для обновления записи необходимо нажать на соответствующую кнопку и содержимое записи с соответствующим номером будет проверено на содержание в базе данных и если оно не содержится, то заменено на данные из изменяемых полей, в ином случае пользователь получит сообщение об ошибке.удаления записи

Для поиска по теоретическому материалу необходимо нажать на соответствующую кнопку и ввести символы, содержащиеся в искомом слове, при этом все записи в базе данных проверяются на содержание введённых символов и в случае если они содержат, то содержимое списка записей обновляется, в списке отображаются только поля содержащие последовательность введённых символов, в случае ни одной записи не найдено пользователь получит сообщение.

Для сортировки списка необходимо нажать на соответствующую кнопку и порядок содержимого списка будет обновлен в соответствии с видом сортировки

Для просмотра теоретического материала пользователю необходимо нажать на кнопку, после этого на экране отобразится новая форма с теоретическим материалом, при этом с начала отображается верхнее меню и содержание выпадающего меню. Для отображения теоретического материала в списке все поля берутся из базы данных и отображаются в списке.

Для выполнения заданий разных типов для закрепления пройденного материала необходимо в выпадающем меню выбрать нужный вариант и на экране отобразится приветственная форма и параметры теста, для начала теста необходимо нажать на соответствующую кнопку и форма измениться на тестовые задания, при этом содержание параметров теста применяется к тесту, для выбора ответа необходимо нажать на кнопку, при этом ответ записывается в временное хранилище, после завершения теста результаты теста рассчитываются и вносятся в базу данных.

Для отслеживания статистки прохождения заданий необходимо выбрать соответствующий пункт и форма измениться на форму с историей прохождения тестов и средними значениями, при этом содержание базы данных заносятся в список пройденных тестов, а средние значения рассчитываются и заносятся в соответствующие пункты на форме.