ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

###### Московский институт электроники и математики

**им. А.Н.Тихонова НИУ ВШЭ**

Щербинин Дмитрий Игоревич, группа БИВ172

**Разработка клиентской части приложения для мониторинга свободных и занятых аудиторий корпуса учебного заведения.**

Междисциплинарная курсовая работа

по направлению 09.03.01 Информатика и вычислительная техника

студентов образовательной программы бакалавриата  
«Информатика и вычислительная техника»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.И.Щербини

подпись И.О. Фамилия

|  |  |
| --- | --- |
|  | Руководитель  ученая степень, звание (при наличии)  С.В.Рысаков |
| Москва 2019 г. | |

Оглавление

[Введение 3](#_Toc8125175)

[1. Условие задачи 7](#_Toc8125176)

[1.1. Основная задача 7](#_Toc8125177)

[1.2. Требования к техническим средствам 7](#_Toc8125178)

[1.3. Требования к программной части 7](#_Toc8125179)

[2. Постановка задачи 8](#_Toc8125180)

[2.1. Описание необходимых алгоритмов 8](#_Toc8125181)

[2.2. Описание графического интерфейса пользователя 8](#_Toc8125182)

[2.3. Создание установочного пакета 8](#_Toc8125183)

[3. Обзор аналогов 9](#_Toc8125184)

[4. Описание интерфейса. 10](#_Toc8125185)

[4.1. Основное окно программы 10](#_Toc8125186)

[4.2. Главное меню 11](#_Toc8125187)

[4.3. Статус-бар 12](#_Toc8125188)

[4.4. Игровое поле 12](#_Toc8125189)

[4.5. Дополнительные элементы интерфейса 13](#_Toc8125190)

[5. Описание алгоритмов. 14](#_Toc8125191)

[5.1. Алгоритмы поиска пути. 14](#_Toc8125192)

[5.1.1. Формирование поля ходов 14](#_Toc8125193)

[5.1.2. Анализ существования пути 14](#_Toc8125194)

[5.1.3. Поиск траектории 15](#_Toc8125195)

[5.2. Алгоритмы рандомизации цвета и местоположения pre-шарика. 15](#_Toc8125196)

[5.2.1. Используемые обозначения 15](#_Toc8125197)

[5.2.2. Рандомизация цвета pre-шарика 15](#_Toc8125198)

[5.2.3. Рандомизация местоположения pre-шарика 16](#_Toc8125199)

[5.3. Алгоритмы поиска и удаления линий шариков 16](#_Toc8125200)

[5.3.1. Используемые обозначения 16](#_Toc8125201)

[5.3.2. Поиск линий от заданного шарика 17](#_Toc8125202)

[5.3.3. Удаление линий, образованных заданным шариком 17](#_Toc8125203)

[6. Тесты 19](#_Toc8125204)

[7. Выводы 20](#_Toc8125205)

[8. Список используемых программных продуктов 21](#_Toc8125206)

[9. Список используемой литературы 22](#_Toc8125207)

# Введение

Основной целью данной междисциплинарной курсовой работы (далее просто курсовая работа) является реализация системы для отслеживания загруженности аудиторий учебного заведения, с целью получения достоверной информации о занятости определенной аудитории в любой момент времени, а конкретно реализация клиентской части данной системы.

**Описание и актуальность проблемы**

Данная МКР посвящена созданию приложения для отслеживания загруженности аудиторий учебного заведения. Каждый день студенты различных учебных заведений во время «окон» между занятиями и иногда после уроков ищут свободные аудитории для того, чтобы позаниматься в тихой, спокойной и удобной обстановке. Также и преподаватели нередко нуждаются в бронировании свободной аудитории на определенное время. Соответственно возникает необходимость в поиске свободных аудиторий. При условиях слабой загрузки и небольшого размера учебного корпуса это не представляется проблемой, но с увеличением площади и этажности здания, а также количества единовременных занятий, поиск свободной аудитории становится непростой задачей, на которую приходится потратить определенное время.

С другой стороны, задача усложняется тем, что не во всех вузах информация о загруженности аудиторий находится в свободном доступе. Соответственно необходим независимый от учебной части способ собирать информацию о занятых аудиториях.

Собственно, это и делает данную задачу актуальной, и именно для ее решения задумывалась данная курсовая.

**Выбор платформы и основных архитектурных решений**

Система представляет собой приложение, построенное на архитектуре «клиент-сервер». Поскольку данная курсовая работа описывает именно разработку клиентской части приложения, то функционирование сервера будет затронуто лишь в необходимом для понимания работы клиентской части объеме.

Клиентская часть системы является приложением для платформы Android (начиная с версии Android 4.4 “KitKat” и выше). Данная операционная система была выбрана по нескольким причинам:

1. огромная популярность на рынке операционных систем;
2. легкость входа в мобильную разработку на данной платформе;
3. большое число поддерживаемых устройств;
4. обилие готовых решений для рутинных задач;
5. наличие официальной среды разработки от компании разработчика ОС Android (Android Studio).

Отсутствие поддержки старых версий Android (Android 4.3 “Jelly Bean” и ниже) обусловлено низкой популярностью данных на момент начала разработки данного приложения.

Выбор клиент-серверной архитектуры также обусловлен несколькими причинами:

1. простота реализации;
2. хранение данных в едином центре, что обеспечивает простоту доступа к данным, а также гарантирует поддержание их в актуальном состоянии;
3. гарантия идентичности данных для всех пользователей;
4. возможность четкого разграничения прав пользователей.

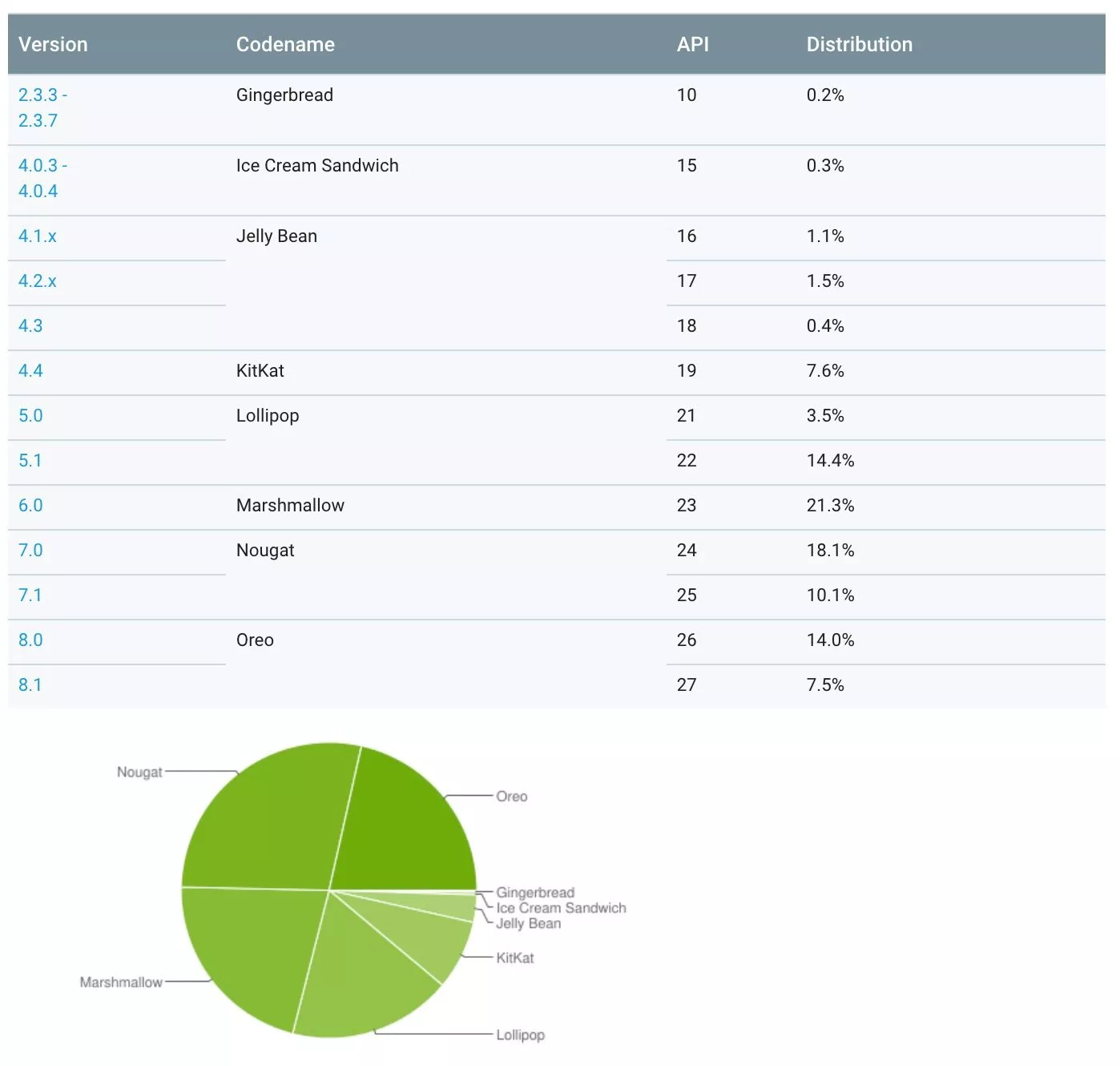


Рис. 1. Распределение используемых версий ОС Android по данным на октябрь 2018 года

**Краткое описание приложения**

Для упрощения дальнейшего описания введем следующие обозначения:

* клиент – клиентская часть приложения;
* сервер – серверная часть приложения.

Приложение состоит из двух частей – клиентской и серверной. Сервер решает задачу хранения, обработки и обновления данных, в то время как клиент предназначен для предоставления доступа к данным на сервере, а также отсылке запросов на их изменение.

Цикл работы клиента представлен на схеме ниже.

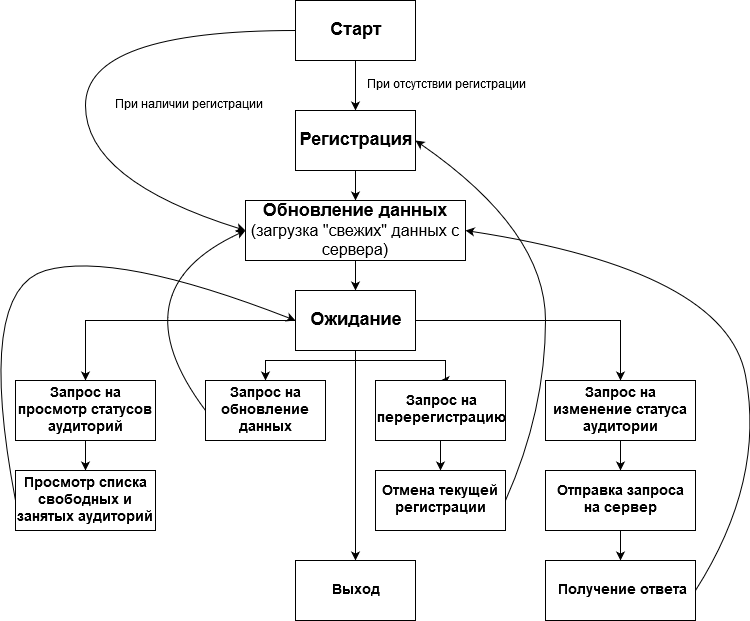


Рис. 2. Цикл работы клиентской-части приложения

Из схемы (рис. 2.) видно, что клиент не автономен и для его правильного функционирования необходима связь с сетью Интернет. Приложение может работать в обход фазы **«Обновление данных»**, используя данные, кэшированные на устройстве, но тогда не гарантируется актуальность предоставляемых данных.

**Средства разработки**

Для разработки приложения для ОС Android мной была выбрана официальная IDE для разработки на Android – Android Studio. Выбор пал на данную среду разработки ввиду ее доступности, просты для освоения и наличия поддержки со стороны создателей.

В качестве основного языка программирования был выбран Java. Наряду с Kotlin, этот язык является основным инструментом разработки под Android с огромной базой готовых решений и большим сообществом, что сильно упрощает разработку.

# Условие задачи

## Основная задача

Необходимо разработать программу для ОС Android, позволяющую отслеживать занятость аудиторий учебного корпуса и поддерживающую актуальность данных о статусах аудиторий.

## Требования к техническим средствам

Программа должна удовлетворять следующим требованиям к техническим средствам.

* Мобильное устройство с экраном и устройством управления, не включая носимые устройства, автомобильные системы и телевизоры.
* Наличие сетевого оборудования для подключения с сети Интернет.
* Не менее 50 Мб свободного дискового пространства на внутренней памяти устройства.
* Процессор с тактовой частотой 800МГц или выше.
* 256Мб ОЗУ.

## Требования к программной части

Программа должна удовлетворять следующим требованиям к программной части: операционная система линейки Android (4.4 “KitKat” или новее).

Приложение должно устанавливаться на мобильное устройством посредством .APK‑файла.

# Постановка задачи

## Описание необходимых алгоритмов

Для правильной реализации приложения требуется разработать и реализовать алгоритмы обработки следующих событий.

* регистрация в системе;
* обновление локальной базы данных на устройстве;
* просмотр статусов аудиторий учебного заведения;
* запрос на обновление данных;
* отправка запроса на сервер с целью изменения статуса аудитории;
* запрос на перерегистрацию;
* запрос на изменение статуса аудитории.

## Описание графического интерфейса пользователя

## Создание установочного пакета

После выпуска стабильной версии программы необходимо создать ее установочный .APK-файл, включающий в себя:

1. весь код приложения (.DEX файлы).
2. ресурсы;
3. активы (assets);
4. файл манифеста (AndroidManifest.xml);
5. нативные библиотеки;

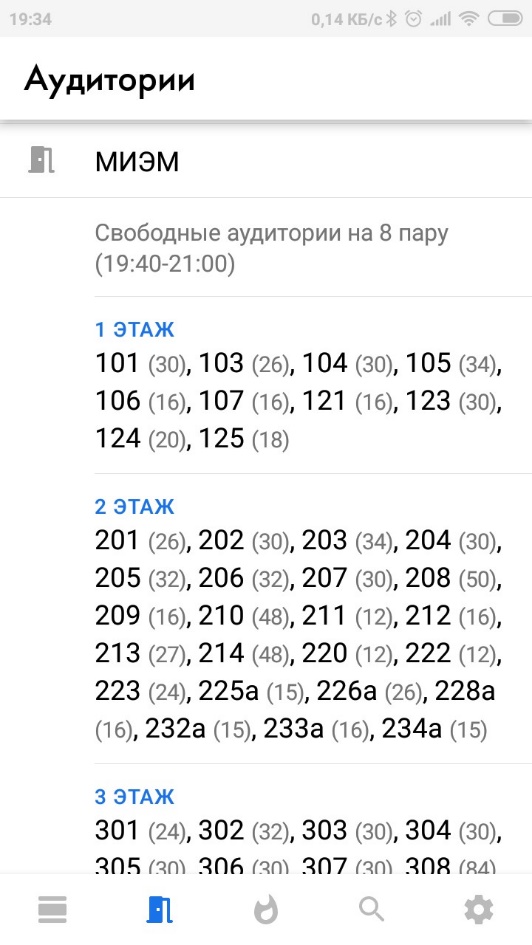
# Обзор аналогов

Данным вопросом не могли не заинтересоваться различные разработчики, что привело к тому, что на данный момент уже существуют готовые решения в данной области. Здесь конкретно рассматривается приложение HSE APP, которое предоставляет функционал для поиска свободных аудиторий в корпусах НИУ ВШЭ в заданный промежуток времени.

Приложение имеет вполне логичный интерфейс, соответствующий современному стилю Android приложений: присутствует выбор корпуса, даты, а также промежутка времени, который нас интересует.

После ввода информации приложение связывается с серверами, на которых находится РУЗ (Расписание Учебных Занятий), находит свободные аудитории подходящие по заданным критериям, и выводит их в удобно читаемом виде по этажам, дополнительно выводя на сколько мест рассчитана каждая аудитория, что позволяет сразу оценить количество человек, которые поместятся в данной аудитории.

Отметим, что работы данного функционала приложения нет необходимости в какой-либо регистрации в системе, но необходим доступ в Интернет для связи с серверами НИУ ВШЭ. Приложение может работать и без подключения с Интернету, но тогда нет никакой гарантии актуальности данных на устройстве.



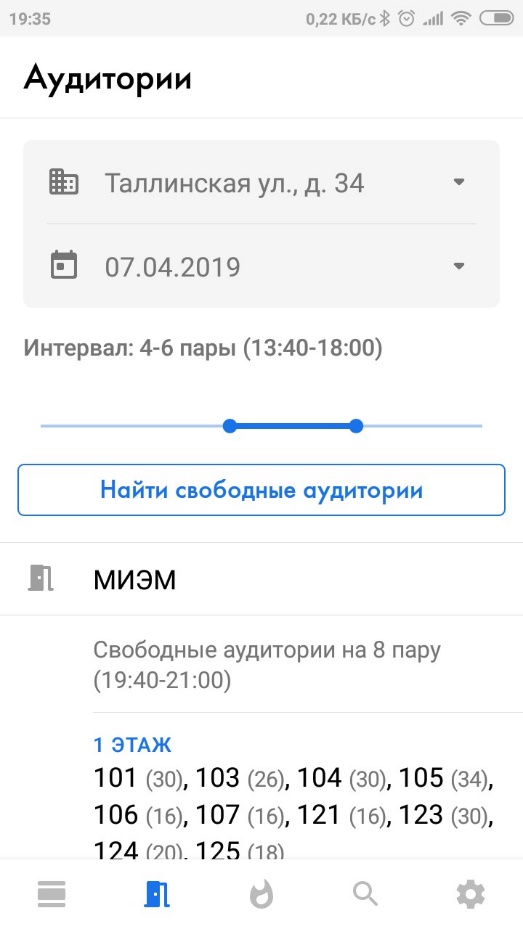


Рис. 3. Интерфейс функционала по поиску свободных аудиторий приложения HSE APP

# Описание интерфейса.

# Описание алгоритмов.

В данном разделе будут приведены основные алгоритмы, необходимые для реализации всех заявленных функций. Все алгоритмы, описанные в данном разделе, используют связь с сервером посредством протокола **HTTP** (RFC 1945, RFC 2616, RFC 7231), доступ к которому осуществляется на клиенте через HTTP клиент **OkHttp**. Все операции, связанные с загрузкой данных на сервер или выгрузкой данных с сервера, осуществляются в фоновом потоке (не в UI потоке). Для передачи данных используется текстовый формат обмена данными - **JSON**.

## Алгоритмы поиска пути.

### Формирование поля ходов

Данный алгоритм использует переменную **SIZE** – размер игрового поля (**SIZE** = 9).

1. Создаем двумерный целочисленный массив *mfield* размером (**SIZE**+2)x(**SIZE**+2). Получим поле **SIZE**x**SIZE**, обернутое в рамку из клеток.
2. В каждую клетку рамки устанавливаем значение -1.
3. Затем для каждой клетки поля внутри рамки проделываем следующее преобразование.
   * Если клетка игрового поля пуста, то в соответствующую ей клетку поля *mfield* записываем 81;
   * Если клетка игрового поля занята шариком, то в соответствующую ей клетку поля *mfield* записываем -1.
4. Записываем 1 в клетку, соответствующую начальной клетке с шариком.

### Анализ существования пути

1. Начинаем обход от начальной клетки поля *mfield*.
2. Смотрим верхнюю клетку. Если значение в ней больше, чем значение в текущей клетке плюс 1, то записываем в неё (верхнюю клетку) значение в текущей клетке плюс 1 и рекурсивно вызываем от нее алгоритм, начиная с шага 2 до шага 5. Иначе переходим к следующему пункту.
3. Аналогично поступаем с нижней клеткой.
4. С правой клеткой.
5. С левой клеткой.
6. Если после завершения рекурсии в клетке, соответствующей целевой пустой клетке, осталось значение 81, то пути не существует. Иначе путь найден.

### Поиск траектории

Запускаем данный алгоритм только при условии, что путь существует. Для этого сначала запускаем алгоритм из пункта 5.2.2.

1. Берем поле ходов *mfield*, полученное после запуска алгоритма из пункта 5.2.2.
2. Создаем двумерный массив **steps** для записи координат клеток.
3. Создаем переменную **max\_steps** – длина пути. Записываем в max\_steps значение из целевой клетки поля *mfield*.
4. Записываем в steps координаты целевой клетки поля *mfield*, **max\_steps** = **max\_steps** – 1.
5. Начинаем обход от целевой клетки поля *mfield*.
6. Смотрим верхнюю клетку. Если в ней значение в ней плюс 1 равно значению в текущей клетке и **max\_steps** 0, то заносим ее координаты в steps, **max\_steps** = **max\_steps** – 1 и рекурсивно вызываем от нее алгоритм, начиная от шага 6 до шага 9.
7. Аналогично поступаем с нижней клеткой.
8. С правой клеткой поступаем аналогично пункту 6.
9. С левой клеткой поступаем аналогично пункту 6.

После завершения рекурсии в массиве **steps** будут находиться координаты всех клеток траектории.

## Алгоритмы рандомизации цвета и местоположения pre-шарика.

### Используемые обозначения

Алгоритмы, описанные в этом подразделе, используют следующие обозначения:

* Функция *rand*(**n**) – случайное число в диапазоне от 0 до **n**-1.
* Переменная **empty\_cell** – количество пустых клеток.
* **vacant\_id** – множество свободных id.
* **occupied\_id** – множество занятых id.

### Рандомизация цвета pre-шарика

1. Присваиваем каждому цвету шарика свой номер от 1 до 7.
2. Генерируем число *rand*(7) + 1 и возвращаем соответствующий ему цвет.

### Рандомизация местоположения pre-шарика

1. Генерируем число k = *rand*(**empty\_cell**).
2. Берем id = **vacant\_id**[k].
3. Запускаем поиск пустой клетки по её id. Поскольку каждый id уникален, все пустые клетки на поле имеют свой собственный id, но при этом мы не знаем точный порядок следования id пустых клеток на поле. Поэтому запускаем линейный поиск по id. Возвращаем клетку с совпадающим id.

## Алгоритмы поиска и удаления линий шариков

### Используемые обозначения

Алгоритмы, описанные в этом подразделе, используют следующие обозначения:

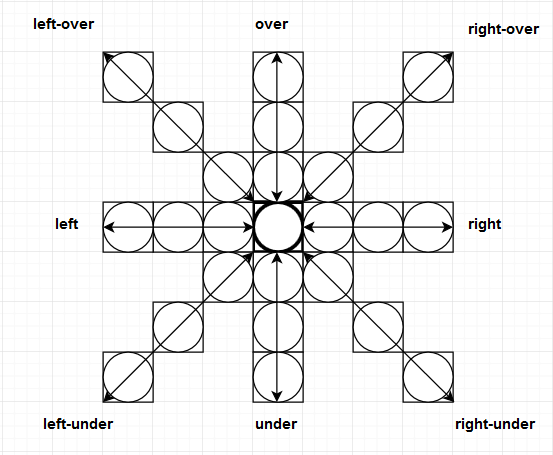
* **right** – правое отклонение от клетки.
* **left** – левое отклонение от клетки.
* **over** – верхнее отклонение от клетки.
* **under** – нижнее отклонение от клетки.
* **right-over** – правое верхнее отклонение от клетки.
* **right-under** – правое нижнее отклонение от клетки.
* **left-over** – левое верхнее отклонение от клетки.
* **left-under** – левое нижнее отклонение от клетки.
* **SIZE** – размер стороны игрового поля (SIZE = 9).

Рис. 2. Иллюстрация отклонений

### Поиск линий от заданного шарика

1. Устанавливаем все отклонения равными 0.
2. Начинаем обход от выбранной клетки игрового поля. Идем по циклу пока сверху не граница поля и шарик в верхней клетке того же цвета, что и шарик в начальной клетке, **over** = **over** + 1, смотрим следующую клетку сверху. Иначе завершаем цикл.
3. Аналогично для клеток снизу.
4. Для клеток справа поступаем аналогично пункту 2.
5. Для клеток слева поступаем аналогично пункту 2.
6. Начинаем обход от выбранной клетки игрового поля. Идем по циклу пока сверху не граница поля и слева не граница поля и шарик в верхней левой клетке того же цвета, что и шарик в начальной клетке,

**left\_over** = **left\_over** + 1, смотрим следующую клетку сверху. Иначе завершаем цикл.

1. Аналогично для клеток справа снизу.
2. Для клеток слева снизу поступаем аналогично пункту 6.
3. Для клеток справа сверху поступаем аналогично пункту 6.
4. Если хотя бы одно из следующих выражений справедливо, то линии найдены:
   * **over** + **under** + 1 5;
   * **left** + **right** + 1 5;
   * **left\_over** + **right\_under** + 1 5;
   * **left\_under** + **right\_over** + 1 5;

Иначе линии не найдены.

### Удаление линий, образованных заданным шариком

Запускаем данный алгоритм только при условии, что на поле находятся линии из 5+ шариков. Для этого сначала запускаем алгоритм из пункта 5.3.2.

1. Начинаем обход от клетки, отстоящей от начальной клетки на over сверху. Идем по циклу, пока текущая клетка не является начальной, удаляем шарик из текущей клетки и переходим на клетку вниз.
2. Аналогично для клеток снизу.
3. Для клеток справа поступаем аналогично пункту 1.
4. Для клеток слева поступаем аналогично пункту 1.
5. Начинаем обход от клетки, отстоящей от начальной клетки на **left\_over** сверху и **left\_over** снизу. Идем по циклу, пока текущая клетка не является начальной, удаляем шарик из текущей клетки и переходим на клетку вниз.
6. Аналогично для клеток справа снизу.
7. Для клеток слева снизу поступаем аналогично пункту 5.
8. Для клеток справа сверху поступаем аналогично пункту 5.
9. Удаляем шарик на начальной клетке.
10. Создаем переменную **total** равную сумме всех отклонений плюс 1.
11. Добавляем очки по следующей формуле: **score** = **score** + (**total**-4)\*5.

# Тесты

# Выводы

По результатам работы, мной было написано приложение, реализующее игру «Lines» и подчиняющееся всем основным правилам данной игры. Пользователю доступна сама игра, а также необходимый интерфейс взаимодействия с ней (сохранение, загрузка, настройки и т.д.). Вся программа, включая исполняемый файл, звуки и прочие необходимые компоненты свернуты в инсталлятор, что делает программу легко распространяемой на любой ПК. К тому же в исполняемый файл заключены все необходимые для работы приложения библиотеки, поэтому игра не требует никаких дополнительных условий для работы на компьютере пользователя, кроме установленной на нем операционной системы семейства Windows.

Как уже говорилось выше, вся графика игры создавалась без привлечения сторонних материалов, кроме встроенных в C++ Builder 6 элементов графики. С одной стороны, это позволило уменьшить размер установочного пакета и программы в целом, а с другой привела к одному последствию. Некоторые тестеры сообщали о том, что при прорисовке некоторых элементов графики (вращение шарика, прорисовка пути шарика от начальной клетки до целевой) наблюдается мерцание главного окна приложения. Скорее всего это связано с задержкой во времени при прорисовке и не каким образом не зависит от производительности компьютера. Одним из путей развития можно обозначить выпуск усовершенствованной версии игры, где данный баг будет устранен.

Также данное приложение можно в будущем портировать для работы на ПК с операционными системы, отличными от систем семейства Windows (Mac OS, Linux и т.д.). Одним из самых перспективных направлений в развитии данной игры можно назвать возможность создать её web-версию на основе уже существующей версии. Это сделает игру более простой в распространении, а также уберет необходимость устанавливать игру на ПК.

# Список используемых программных продуктов

* Microsoft Visual Studio 2017 Community.
* Borland C++ Builder 6.
* Smart Install Maker.

# Список используемой литературы

* + Герберт Шилдт. C++. Базовый курс. Москва. М..Вильямс, 2015.
  + Род Стивенс. Алгоритмы. Теория и практическое применение. Москва. М.Издательство «Э», 2017.
  + Программирование в C++ Builder. URL: <http://h-l-l.ru/> (дата обращения 05.05.2017).
  + Краткое руководство по работе с классом TCanvas для начинающих – C++ Builder. URL: <http://www.cyberforum.ru/cpp-builder/thread267813.html> (дата обращения 07.05.2017).
  + Интернет учебник C++ Builder. URL: <https://cubook.supernew.org/> (дата обращения 05.05.2017).