Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский университет «Высшая школа экономики»

Факультет  [экономики, менеджмента и бизнес-информатики](http://perm.hse.ru/bi/)

Бизнес - информатика

Первый курс

Информационные системы в бизнесе

**О Т Ч Е Т**

**по прикладной практике**

Проект

“Нахождение на рисунке черного квадрата”

Выполнил студент гр. БИ-21-2

Полежаев Яков Вячеславович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

**Проверил:**

Еремин Евгений Александрович, доцент

кафедры информационных технологий в

бизнесе НИУ ВШЭ – Пермь

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

Пермь, 2022 год

**Оглавление**

[Введение. 2](#_Toc106144880)

[1.1 Описание предметной области 4](#_Toc106144881)

[1.2 Постановка задачи. 5](#_Toc106144882)

[1.3 Требования к разрабатываемому приложению. 6](#_Toc106144883)

[1.4 Преимущества и недостатки аналогичных приложений. 6](#_Toc106144884)

[1.5 Выбор программной среды. 7](#_Toc106144885)

[2.1 Проектирование интерфейса 8](#_Toc106144886)

[**2.1.1** **Интерфейс главного меню** 9](#_Toc106144887)

[**2.1.2** **Интерфейс меню для работы с квадратом.** 9](#_Toc106144888)

[2.2 Описание используемых алгоритмов. 10](#_Toc106144889)

[**2.2.1** **Алгоритм перемещения приложения по экрану.** 10](#_Toc106144890)

[**2.2.2** **Алгоритм открытия новых окон приложения.** 10](#_Toc106144891)

[**2.2.3** **Алгоритм бинаризации изображения.** 11](#_Toc106144892)

[**2.2.4** **Алгоритм поиска черного квадрата.** 12](#_Toc106144893)

[**2.2.5** **Алгоритм трансляции массива в текстовый файл** 14](#_Toc106144894)

[3.1 Разработка алгоритмов для работы с кнопками 14](#_Toc106144895)

[3.2 Разработка алгоритма для перемещения приложения. 15](#_Toc106144896)

[3.3 Разработка алгоритма для бинаризации изображения. 16](#_Toc106144897)

[3.4 Разработка алгоритма для поиска квадрата. 18](#_Toc106144898)

[4.1 Тестирование всплывающих окон 20](#_Toc106144899)

[4.2 Тестирование Переноса массива в txt 21](#_Toc106144900)

[4.3 Тестирование поиска квадрата 22](#_Toc106144901)

[Заключение. 23](#_Toc106144902)

[Список литературы 24](#_Toc106144903)

[Приложение А. Сервис для определения объектов на фото 27](#_Toc106144904)

[Приложение Б. Методы сегментации изображений 27](#_Toc106144905)

[Приложение В. Интерфейс приложения 28](#_Toc106144906)

# Введение

Стремительное развитие технологий позволило человечеству войти в новую информационную эру. Еще 30 лет назад устройства, которыми сейчас владеет практически каждый человек на земле, казались бы чем-то немыслимым, магией.

К таким устройствам в первую очередь относят компьютеры, которые с 1981 года и по сей день приобретают все более богатый функционал. Нередко компьютеры сравнивают с человеком и даже устраивают соревнования, например шахматная игра, где программа “[Deep Blue](https://translated.turbopages.org/proxy_u/en-ru.ru.0213f2f5-62a61889-c69aa884-74722d776562/https/en.wikipedia.org/wiki/Deep_Blue_(chess_computer))” одержала победу над тогдашним [чемпионом мира по шахматам](https://translated.turbopages.org/proxy_u/en-ru.ru.0213f2f5-62a61889-c69aa884-74722d776562/https/en.wikipedia.org/wiki/World_Chess_Champion) [Гарри Каспаровым](https://translated.turbopages.org/proxy_u/en-ru.ru.0213f2f5-62a61889-c69aa884-74722d776562/https/en.wikipedia.org/wiki/Garry_Kasparov) в 1997. Сегодня компьютеры способны почти на все, что может человек, а некоторые действия делают даже лучше.

Однако для выполнения таких сложных алгоритмов компьютеру требуются инструкции для работы с получаемой информацией. Одной из областей компьютерных наук, занимающихся разработкой данных инструкций, является анализ изображений. Данный анализ позволяет извлекать нужную информацию из цифровых изображений с помощью методов цифровой обработки. Множество инструментов данного анализа разработаны на основе моделей зрительного восприятия человека. Анализ изображений используется во многих сферах деятельности, начиная со считывания штрих кода и заканчивая распознаванием человека на фото.

Несмотря на широкий спектр использования анализа изображений, программ, способных распознавать фигуры, все еще мало. Такая простая фигура как черный квадрат может встретиться практически везде: черная клетка в шахматах и шашках, цензура, QR коды, квадрат Малевича, CAPTCHA, метки для передвижения роботов и т.д. К сожалению, из-за нехватки инструментов для распознавания черного квадрата автоматизация процессов, связанных с данными объектами невозможна. Именно поэтому я считаю проблему по разработке простейшего приложения для анализа изображения актуальной.

Во время поиска материалов по данной теме я натыкался на веб-страницы, являющиеся аналогом данному приложению. Не смотря на их схожесть были и различия. Их программы специализировались на широком поиске всех возможных объектов на фото, из-за чего страдала производительность и точность. На фотографии теста было загружено фото с девушкой и черным квадратом рядом с ней, и программа нашла девушку с вероятностью 73%, а обычный черный квадрат она восприняла как телевизор с вероятностью 31% [[см. ПРИЛОЖЕНИЕ А.1](#_Приложение_А._Сервис)].

*Целью проекта* является разработка простого приложения для нахождения черного квадрата на изображении, вывода размера его стороны и координат верхнего левого угла. Для достижения поставленной цели необходимо выполнить следующие *задачи*:

1. Проанализировать работу аналогов приложения
2. Спроектировать макет приложения
3. Освоить инструменты для создания десктоп приложений.
4. Разработать алгоритмы, на которых будет работать приложение.
5. Реализовать алгоритмы на языке программирования C#.
6. Провести тестирование.

Основные методы исследования для создания проекта по выбранной теме: наблюдение, сравнение, визуализация, моделирование. С помощью наблюдения изучаются программы анализа изображений, а именно реализованные в них методы анализа. Метод сравнения позволяет проанализировать выбранные приложения и на их основе выявить наиболее эффективные способы анализа. Визуализация на основе наблюдений помогает получить внешнее представление приложения по нахождению черного квадрата, а именно как будет выглядеть интерфейс, какое взаимодействие с пользователем потребуется. Метод моделирования позволяет разработать блок-схемы алгоритмов для отдельных функций системы.

1. **Анализ задач и инструментов, разработка требований**

## **Описание предметной области**

Зрение человека – это сильный инструмент для восприятия и анализа окружающего мира. Сегодня существует его аналог, созданный людьми, который называют компьютерным зрением. Из него же вытекает более узкая область – цифровая обработка и анализ изображений.

Сегментация изображений - это процесс присвоения метки каждому пикселю изображения. Результатом сегментации изображения является набор сегментов, которые в совокупности покрывают всю картинку. Каждый из пикселей в области похож по некоторой характеристике ([цвет](https://translated.turbopages.org/proxy_u/en-ru.ru.a056aa53-62a64ab6-22bc8ac0-74722d776562/https/en.wikipedia.org/wiki/Color), [интенсивность](https://translated.turbopages.org/proxy_u/en-ru.ru.a056aa53-62a64ab6-22bc8ac0-74722d776562/https/en.wikipedia.org/wiki/Luminous_intensity) или [текстура](https://translated.turbopages.org/proxy_u/en-ru.ru.a056aa53-62a64ab6-22bc8ac0-74722d776562/https/en.wikipedia.org/wiki/Image_texture)), а границы сегментов соответствуют контурам объектов.

Распознавание объектов – технология в области [компьютерного зрения](https://translated.turbopages.org/proxy_u/en-ru.ru.a056aa53-62a64ab6-22bc8ac0-74722d776562/https/en.wikipedia.org/wiki/Computer_vision) для поиска и идентификации объектов в изображении или видеоряде.

Алгоритмы анализа:

1. Бинаризация и выделения связанных компонент - это перевод цветного (или в градациях серого) изображения в двухцветное черно-белое. Главным параметром такого преобразования является пороговая фильтрация (Thresholding). По результатам сравнения, пикселю присваивается значение ниже порога: 0 или выше: 1. Существуют различные методы бинаризации, которые можно условно разделить на две группы – глобальные и локальные. В первом случае величина порога остается неизменной в течение всего процесса бинаризации. Во втором изображение разбивается на области, в каждой из которых вычисляется локальный порог. [[см. ПРИЛОЖЕНИЕ Б.1](#_Приложение_Б._Методы)]
2. Последовательное сканирование - это специальный метод анализа изображений, который позволяет получить все линии изображения за одно сканирование. Он заключается в последовательном просмотре значений критерия оптимальности в ряде точек и нахождении среди них такой точки, в которой критерий имеет оптимальное значение.
3. Кластеризация К-средних – это метод, который разделяет данные на множества непересекающихся кластеров. Перед началом кластеризации требуется определить точное количество кластеров, после чего алгоритм будет относить каждый объект к конкретному кластеру. При сегментации объектами являются пиксели. Таким образом можно проводить кластеризацию по яркости и цвету. [[см. ПРИЛОЖЕНИЕ Б.2](#_Приложение_Б._Методы)]

Шум в бинарных изображениях – замещение 1 на 0 с вероятностью (1-р) и замещение 0 на 1 с вероятностью (1-q). С ростом параметров шума растет искажение изображения. При сильном зашумлении человеческому глазу уже становится сложно определить контуры объектов. Существует множество фильтров, подавляющих шумы: Переход в координаты яркость-цвет, медианный фильтр, фильтры, управляющие величиной коррекции, фильтры, использующие спектральное представление сигнала и т.д. [[см. ПРИЛОЖЕНИЕ Б.3](#_Приложение_Б._Методы)]

## **Постановка задачи.**

Для реализации поставленной цели – разработать приложение для нахождения черного квадрата на изображении – необходимо выполнить следующие задачи:

1. Проанализировать работу аналогов приложения – на просторах интернета было найдено множество сервисов для обнаружения объектов на фото. Судя по их функционалу можно предположить, что они связаны с нейронными сетями, из-за чего требуют много ресурсов для работы (среднее время ожидания результатов 1,5 минуты).
2. Спроектировать макет приложения – интерфейс является неотъемлемой частью любого приложения. Именно он определяет первое впечатление пользователя. Поэтому интерфейс должен быть простым и удобным.
3. Освоить инструменты для создания десктоп приложений – для реализации данного проекта будут рассмотрены такие популярные и простые в использовании среды разработки, как Visual Studio и Unity.
4. Разработать алгоритмы, на которых будет работать приложение – для нахождения квадрата потребуются наиболее простые методы анализа изображения с множеством документации на просторах интернета.
5. Реализовать алгоритмы на языке программирования – для написания кода будет использован язык С#, который является чем-то средним между языком C++, который славится своей производительностью и языком Python, который в свою очередь легок в освоении.
6. Провести тестирование - во время разработки для исключения возможности у пользователей проблем с эксплуатацией приложения будут проводиться тесты приложения в зависимости от того, что реализовано на данный момент.

## **Требования к разрабатываемому приложению.**

Приложение для поиска черного квадрата должно быть предельно простым и производительным, чтобы оно могло спокойно запускаться на любом компьютере. Оно не должно быть загружено лишними функциями, чтобы не сбивать с толку пользователей. К основным же функциям относятся: загрузка изображений, преобразование изображений в массив единиц и нулей, сопровождение пользователя подсказками, поиск черного квадрата и наконец вывод информации о нем и его выделение на изображении в красную рамку. В качестве дополнительной функции следует добавить перепись массива изображения в текстовый файл для визуализации работы приложения.

## **Преимущества и недостатки аналогичных приложений.**

Как уже говорилось ранее существует множество приложений-аналогов, имеющих похожий функционал. Однако, они обладают определенными минусами, не мешающими достижению нашей цели. Ниже приведена таблица для наглядного сравнения наиболее популярных приложений:

**Таблица 1 – преимущества и недостатки аналогов приложения.**

|  |  |  |
| --- | --- | --- |
| **Название** | **Преимущества** | **Недостатки** |
| Nuance OmniPage Ultimate | индивидуальные системы настроек; высокая скорость; точность распознавания. | высокая цена; сложно разобраться начинающим пользователям; платные обновления. |
| Google Goggles | полностью бесплатно; современные алгоритмы обработки; высокая скорость. | точность распознавания лиц невысокая; ранжирование результатов в большинстве случаев ошибочно; очень много находит похожих объектов |
| Amazon Rekognition | удобный интерфейс; быстрая обработка; возможность сравнения характеристик. | больше нацелен на обработку неодушевленных объектов; нет русского интерфейса; ее ищет единичные объекты. - |
| Clarifai | уникальная система обработки данных; высокая скорость работы; пока бесплатно. | система еще тестируется; обработка изображений по конкретным серверам; глобальный поиск отсутствует. |

На данный момент ближе всего к нужному варианту относится приложение “Amazon recognition. Оно обладает высокой скоростью обработки приложения, удобным интерфейсом и специализируется на поиске одного конкретного объекта. Однако, не смотря на бесплатный пробный период на 12 месяцев, оно остается платным.

## **Выбор программной среды.**

Для разработки приложения выбрана среда “Visual Studio” по следующим причинам:

1. Стабильность: “Visual Studio” стабильный продукт с долгой историей обработки крупных проектов.
2. Функционал: “Visual studio” имеет массу бесплатных и платных плагинов, которые расширяют его функциональность. Популярным примером является “JetBrains ReSharper”, который автоматизирует ваши процедуры кодирования.
3. Удобный и понятный интерфейс: “Visual Studio” обладает множеством визуальных решений, которые упрощают разработку. Более того данная среда позволяют стилизовать свой код, что не может не радовать глаз.
4. Документация: Существует обширная документация для “Visual Studio”. В целом, во всем мире гораздо больше разработчиков используют “Visual Studio”, поэтому можно найти множество ссылок в интернете.
5. Опыт: Основным фактором было то, что у меня уже имеется опыт работы с “Visual Studio”. Исходя из этого, чтобы не тратить ценное время на освоение новой среды разработки лучше пользоваться уже известной средой.
6. **Проектирование приложения.**

## **Проектирование интерфейса**

Интерфейс является неотъемлемой частью программы. Именно он определяет кол-во элементов приложения для которых предстоит написать код, их расположение и будущий функционал. Именно потому проектирование приложения стоит начинать именно с него. Во время проектирования визуальной части был предпочтителен стиль минимализм: без множества кнопок, без ярких и вызывающих цветов и без огромного количества текстовой информации. Все это предпринято для того, чтобы не отвлекать пользователя от изображения, которое само по себе может быть пестрым и ярким.

Для общей темы приложения была выбрана Япония, а именно ее природа и культура. Не мало известный факт, что одним из символов Японии является цветущая сакура, которая завораживает приезжих своей красотой. Поэтому изображения для фона я выбирал именно по запросу “Цветущая сакура на фон”. После поиска образовался список из 6 фотографий, которые потом я отбирал путем загрузки каждой в свое приложение. В итоге главным фоном стало изображение с неярким задним планом, сакурой и красным солнцем. [[см. ПРИЛОЖЕНИЕ В.1](#_Приложение_В._Интерфейс)] Отсутствие множества объектов позволит пользователю не отвлекаться на фон, а неяркий задний план не будет давить на глаза и сливаться с загруженной юзером фотографией.

Что касается кнопок, для них был выбран особенный шрифт (MV Boli, наклонный,36), который хорошо сочетается с японской тематикой. Сами кнопки так же сильно не выделяются, например, для закрытия и возврата на главное меню были использованы текстовые лейблы, окрашенные в красный “X” и “ <=”.

### **Интерфейс главного меню**

При открытии приложения пользователя встречает форма c фоном сакуры и бирюзовой кнопкой по центру. В верхнем правом находится красный крестик для закрытия программы. [[см. ПРИЛОЖЕНИЕ В.2](#_Приложение_В._Интерфейс)]

На кнопке по центру написано “Start” и при нажатии на нее открывается Windows окно для загрузки изображения. Для удобства в Windows окно уже были встроены фильтры для поиска изображений нужного формата. При наведении курсора на любую рабочую кнопку будет выводиться информация о ней.

### **Интерфейс меню для работы с квадратом.**

После выбора фото в нижней половине приложения появится картинка пользователя, которая, к сожалению, может растянуться из-за разницы в разрешении у фотографии и области, куда она загружается. В нижнем правом углу картинки появится кнопка “Find Quad” похожая на прошлую из главного меню. Слева от картинки появится красная кнопка “ <=”, которая позволяет пользователю вернуться в главное меню. [[см. ПРИЛОЖЕНИЕ В.3](#_Приложение_В._Интерфейс)]

При нажатии на кнопку “Find Quad” программа начнет переводить загруженное фото в массив из единиц и нулей и искать квадрат. После этого всплывет небольшое windows окно с настоящим статусом квадрата. Если на фотографии был квадрат, то выведутся координаты его верхнего левого угла и размер его стороны. [[см. ПРИЛОЖЕНИЕ В.4](#_Приложение_В._Интерфейс)] Если его не было, то будет сообщение “квадрата нет”. После закрытия окна вокруг квадрата на изображении появится красная рамка, обозначающая его границы (при условии, что квадрат был).[[см. ПРИЛОЖЕНИЕ В.5](#_Приложение_В._Интерфейс_1)]

По завершению программы на рабочем столе можно обнаружить текстовый файл с названием “cherny kvadrat”, где будет находиться бинарное представление изображения в виде массива из единиц и нулей, где 0 показывает черную область, а 1 любую другую. Однако, отношение его высоты и ширины не будут соответствовать реальным из-за особенностей блокнота в который записывается массив, а именно расстояние между строками больше чем между столбцами. [[см. ПРИЛОЖЕНИЕ В.6](#_Приложение_В._Интерфейс_1)]

## **Описание используемых алгоритмов.**

Как уже было сказано выше во многих аналогах для распознавания объектов использовались нейронные сети, обученные на больших массивах изображений. Однако, встречались и программы, использующие метод бинаризации и выделения связанных компонент. Именно эти программы будут взяты в качестве примера для построения алгоритмов.

### **Алгоритм перемещения приложения по экрану.**

Во время разработки приложения пришлось столкнуться с проблемой, что стандартные рамки приложения никак не сочетались с интерфейсом, поэтому было решено их убрать. Однако, при удалении этих рамок пропала возможность передвигать приложение и пришлось разрабатывать новый алгоритм перемещения. К счастью, на данную тему много документации и найти нужный алгоритм не составило труда.

Сам алгоритм заключается в том, что при нажатии мышью программа запоминает ее координаты x, y. Если мышь зажата и перемещается, то к координатам приложения прибавляется разница между текущими координатами курсора и начальными координатами. Например, изначально координаты верхнего левого угла приложения были (20;20) и пользователь зажал мышку на (25;10) после чего передвинул ее вправо на единицу. Разница координат равна (26-25=1; 10-10=0), следовательно, координата приложения по X увеличится на 1, а по Y не изменится.

X += x (2) – x (1); Y += y (2) – y (1)

### **Алгоритм открытия новых окон приложения.**

Изначально предполагалось, что для каждого нового этапа работы приложения будет создаваться новая форма с кнопками и изображениями, но позже был обнаружен более простой алгоритм. Его суть заключалась в том, чтобы работать с одной единственной формой, а ее элементы в зависимости от этапа просто прятать или наоборот показывать. Данный алгоритм позволяет избежать затормаживаний во время работы приложения, ведь все элементы будут созданы в самом начале. Кроме этого упрощается написание кода [Рисунок 1].

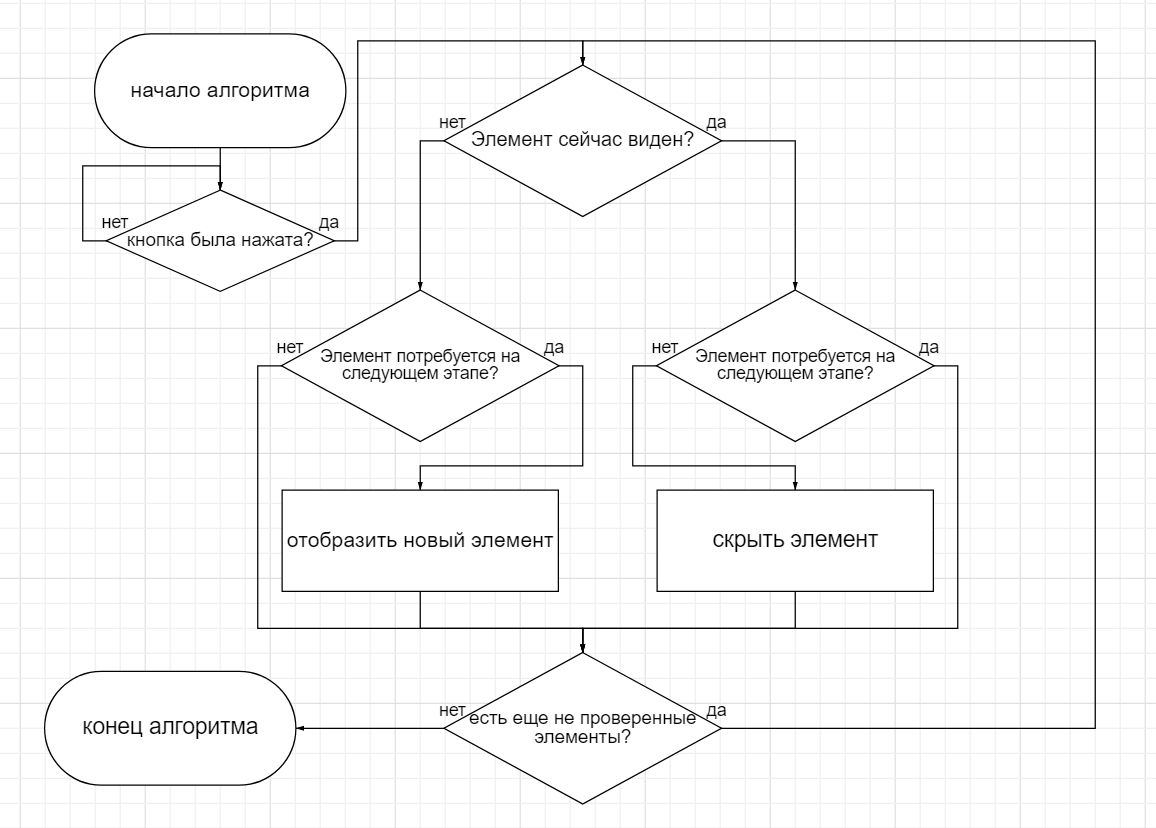


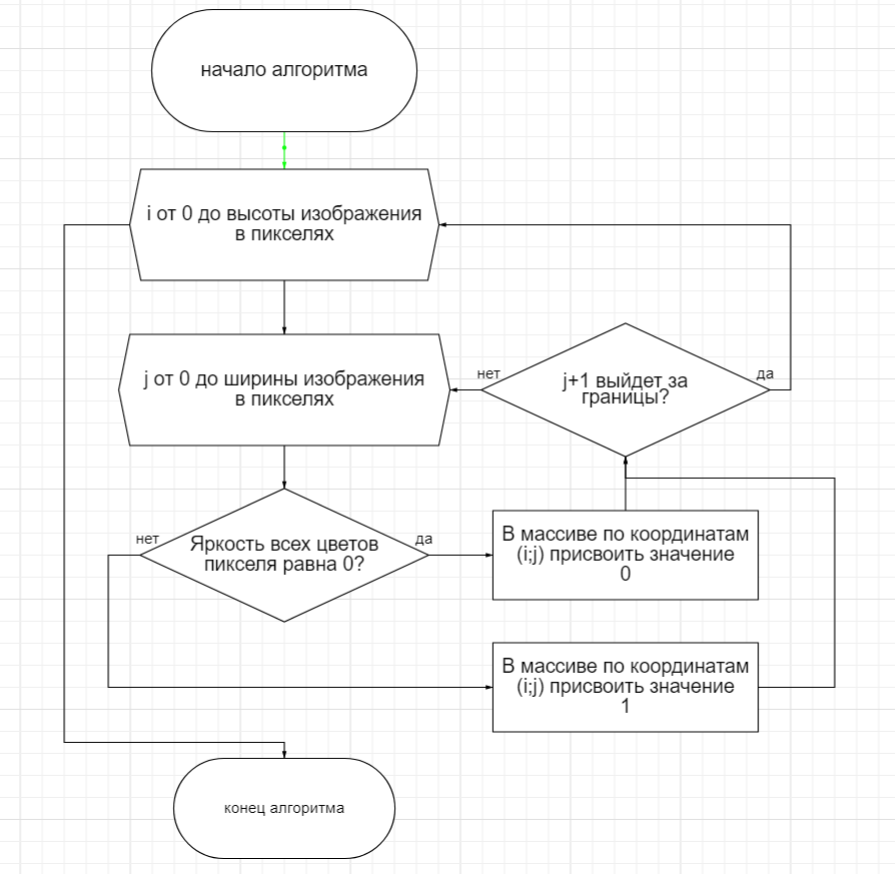
Рисунок 1 Блок схема Алгоритма отображения элементов

### **Алгоритм бинаризации изображения.**

Результат преобразования картинки представляет собой числовой массив единиц и нулей, где нули показывают черные участки. Всего для достижения данного результата было реализовано два алгоритма.

Первый алгоритм представлял собой перевод цветного изображения в градацию серого путем присвоения каждому пикселю усредненного значения RGB. После этого можно было задавать порог присвоения единицы или нуля для каждого участка картинки, ниже порога 0 выше 1.

Второй алгоритм является более упрощенным вариантом первого и успешно заменил его. [Рисунок 2] Упрощение состояло в том, что изображение уже не переводилось в серые оттенки. Теперь у каждого пикселя брались значения яркости RGB и, если все три показателя яркости равны 0, то присваивался 0, в противном случае 1. Данный алгоритм позволил избежать случаев, когда квадрат находится на темном фоне, ведь пока этот участок не является идеально черным, ему не присвоится 0. Более того это уменьшило нагрузку на компьютер.



**Рисунок 2 Блок схема Алгоритма бинаризации изображения.**

### **Алгоритм поиска черного квадрата.**

В процессе поиска идеального алгоритма поиска квадрата, который бы проходил все проверки было перепробовано множество вариантов. Начало было у всех одинаковым, перебор массива в поисках возможного наименьшего квадрата 2х2, так как в любом большом квадрате есть маленький. Данный размер был выбран, для избегания лишних проверок одиночных пикселей, а сам квадрат определялся путем смещения от выбранной координаты, которая должна быть верхним левым углом квадрата, на 1 вниз вверх и по диагонали и сравнением их значений с нулем.

Далее начиналась самая трудоемка часть – определение размеров квадрата и тесты, показывающие квадрат ли это. Один из самых успешных алгоритмов имел следующую реализацию:

1. определение длин верхней и левой стороны квадрата путем постепенного перемещения вниз и вправо от верхнего левого угла, пока не встретится 1.
2. Сравнение этих длин
3. Если длины были равны начиналась проверка внутренней части квадрата путем суммирования значений.
4. Если сумма была равна 0, то начиналась проверка внешних границ квадрата, чтобы вокруг не было больше чем 10 единиц. Данная проверка похожа на измерение сторон этого квадрата, но большего размера и при итерации суммировались значения каждой координаты.
5. При прохождении всех тестов программа останавливалась, в противном случае все элементы квадрата принимали значение 1, для исключения повторных проверок и ошибок, и продолжался перебор массива.

К сожалению, у данного алгоритма были некоторые ошибки и трудная логика, что усложнило бы процесс разработки. Однако, благодаря своему руководителю проекта был создан упрощенный, но не менее эффективный алгоритм, который менял только сам анализ квадрата. Было предложено постоянное наращивание стороны квадрата, пока не попадется сторона с единицей, после чего проверялось пространство вокруг фигуры. Данный метод значительно упростил написание кода и позволил избежать ошибок, которые попадались ранее [Рисунок 3]

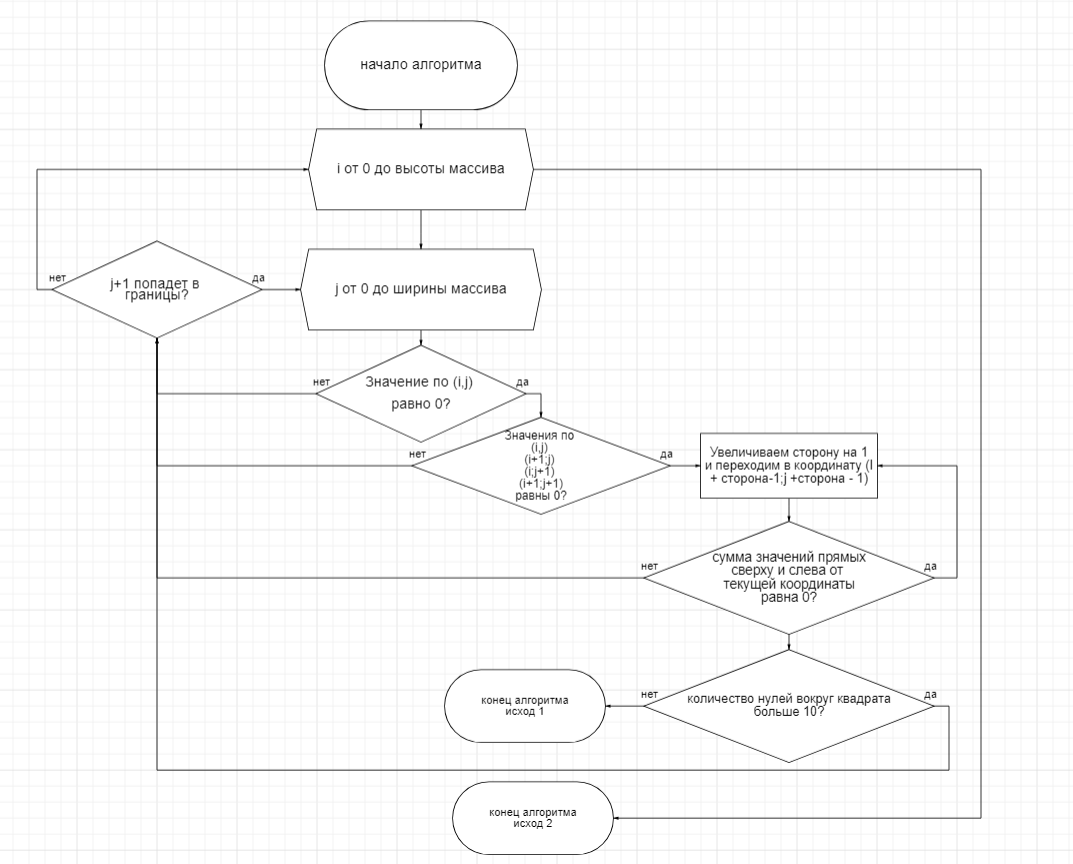


Рисунок 3 Блок схема Алгоритма поиска квадрата.

### **Алгоритм трансляции массива в текстовый файл**

Данный алгоритм с виду казался простым: в текстовый файл поочередно записываются значения массива, когда длина по икс заканчивается происходит переход на новую строку. Проблемы возникли с самим текстовым файлом, так как количество столбцов в блокноте было ограниченным (1000 шт.), что не соответствовало разрешению некоторых картинок. Из-за этого пришлось создавать копию изображения меньшего размера, подходящего для границ блокнота и уже его переводить в массив и записывать в блокнот.

1. **Процесс реализации приложения.**

Расстановка объектов интерфейса и описание их поведения в коде могут проходить непосредственно в Visual Studio, что делает сам процесс создания приложения крайне удобным. Однако, для реализации некоторых функций потребуется подключение сторонних библиотек, например, System.IO для переноса массива в текстовый документ.

## **Разработка алгоритмов для работы с кнопками**

Кнопки являются одним из самых важных объектов в приложении без которых оно даже не начнет работу. Сами кнопки относятся к классу “Button” и имеют большое количество свойств и методов. Выполнять их основную задачу, а именно что-то делать после нажатия, позволяет событие “Button-Click”, на которое уже и пишутся инструкции. Как уже говорилось ранее, во время работы приложения кнопки будут пропадать и появляться. За данное поведение отвечает свойство Visible, которое принимает значение “true” или “false”.

Кроме этого в конструкторе формы, как и в самом коде, кнопкам можно присваивать еще множество параметров. Например, у кнопки на главном меню есть определенный размер, расположение, текст, цвет заднего фона и т.д. Я нахожу очень удобным свойство, которое привязывает всплывающее окно с информацией “Tool tip”, при наведении курсора, к конкретной кнопке. С его помощью можно сделать подсказки пользователю [Рисунок 4].

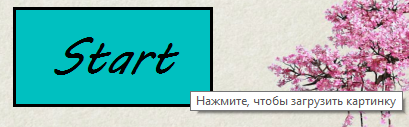


Рисунок 4 Демонстрация появления Tool Tip

Не стоит исключать тот факт, что текстовые окна так же можно сделать кнопками. Более того данная функция реализована в данном приложении. Данные кнопки упоминались выше: “X” и” <=”, они являются объектами типа “Label”, но тоже реализуют событие нажатия.

Ниже приведен пример использования свойства “Visible” в реализации нажатия кнопки “ <=”. [Рисунок 5]

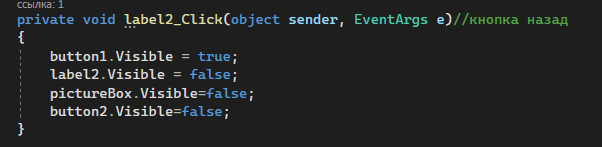


Рисунок 5 Фрагмент кода Событие нажатия на Label “<=”

## **Разработка алгоритма для перемещения приложения.**

Как уже говорилось ранее, возникла потребность в альтернативном способе перемещать окно приложения, и в качестве него было выбрано обычное прибавление разности координат начальной и конечной точек курсора. Начальные координаты курсора можно достать из события Menu\_MouseDown, в то время как для получения текущих координат надо создать событие Menu\_MouseMove и из него брать значения.

Ниже приведена реализация данной логики, где “this.Left” и “this.Top” - координаты верхнего левого угла приложения; “e.X” , “e.Y” – текущие координаты курсора мыши; “lastPoint.X” “lastPoint.Y” – координаты точки нажатия. [Рисунок 6]

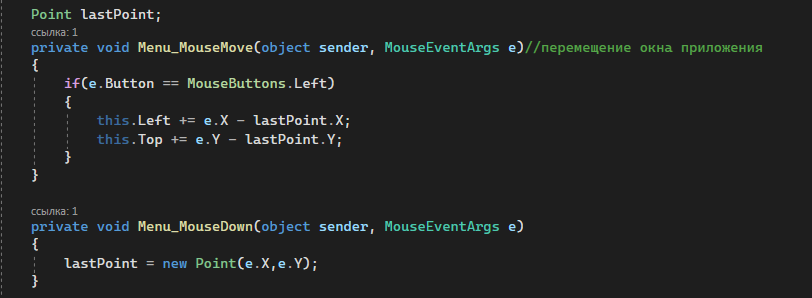


Рисунок 6 Фрагмент кода Перемещение окна

## **Разработка алгоритма для бинаризации изображения.**

Для изображения был отдельно создан класс “newImage” в котором прописаны все методы для работы с картинкой. В данном классе есть два статистических поля “Bitmap bitmap”, что является нашим изображением, и массив чисел “numsImage”. [Рисунок 7]



Рисунок 7 Фрагмент кода Поля класса “newImage”

Преобразование картинки в массив происходит в методе “ToNums()”. Изначально в этом методе выделяется память под массив “numsImage”, размеры которого равны размерам bitmap. Далее в методе происходит итерация каждого пикселя в изображении при помощи цикла for, где при условии, что яркость пикселя по RGB равна 0, в массив по таким же координатам записывается значение ноль, в противном случае 1.

Ниже продемонстрирован сам метод “ToNums()”. “Index” отвечает за номер элемента внутри массива “nums”, где находятся числа {0, 1}.[Рисунок 8]

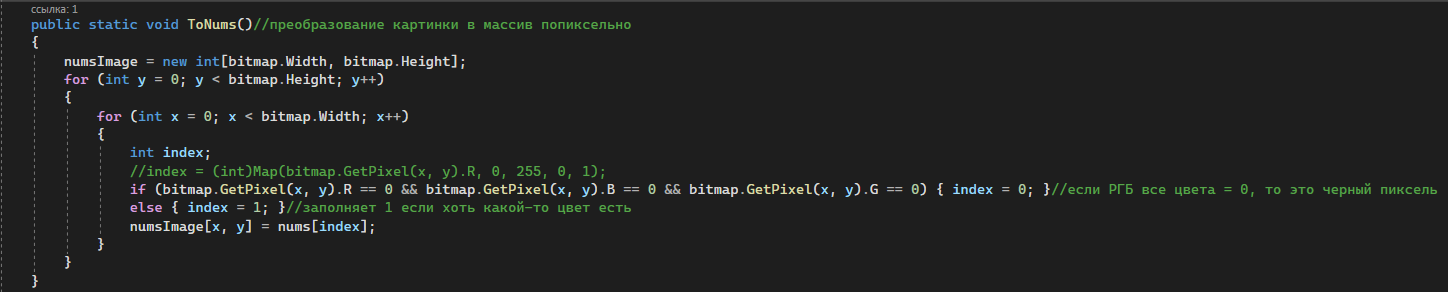


Рисунок 8 - Разработка алгоритма для переноса массива в .txt

После успешного выполнения бинаризации изображения требуется перенести массив в .txt файл на рабочем столе или если его нет создать новый. Так как блокнот имеет ограничение по количеству столбцов, требуется дополнительная проверка размеров массива. [Рисунок 9] При несоответствии размерам создается копия изображения с меньшим разрешением, и оно тоже проходит этап бинаризации в массив “sendArray”.

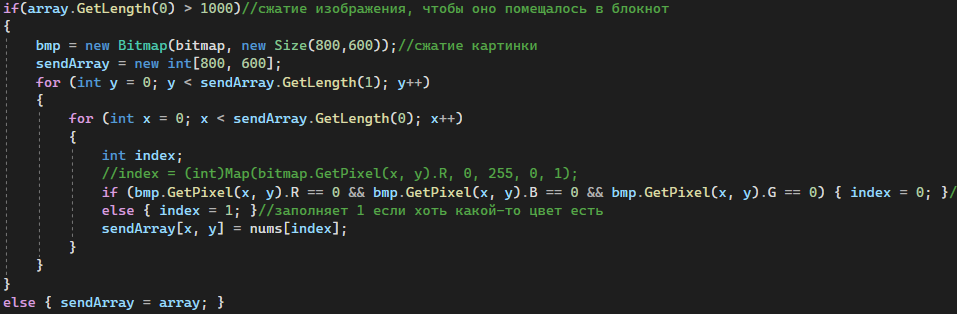


Рисунок 9 - Фрагмент кода Проверка размеров массива

После этого открывается поток для трансляции массива. В этом потоке создается объект класса “StreamWriter sw”, что в свою очередь создает на рабочем столе .txt файл если его нет или получает к нему доступ и перезаписывает. Для открытия рабочего стола требуется имя юзера, его получаем при помощи класса Environment. После завершения передачи данных поток закрывается и объект “sw” удаляется. [Рисунок 10, Рисунок 11]

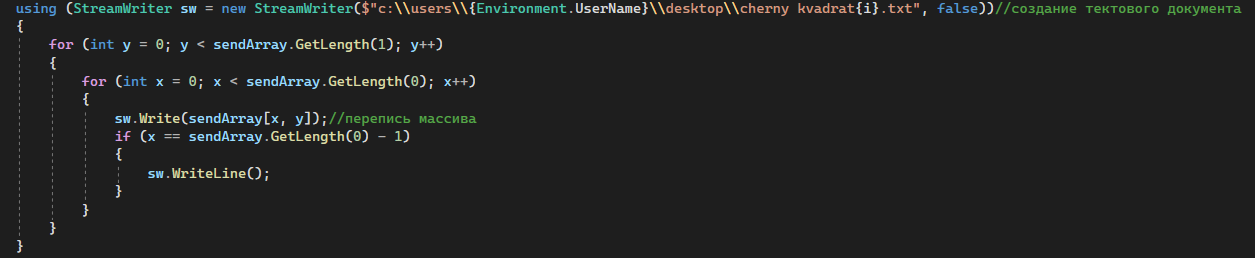


Рисунок 10 Фрагмент кода Передача данных в txt

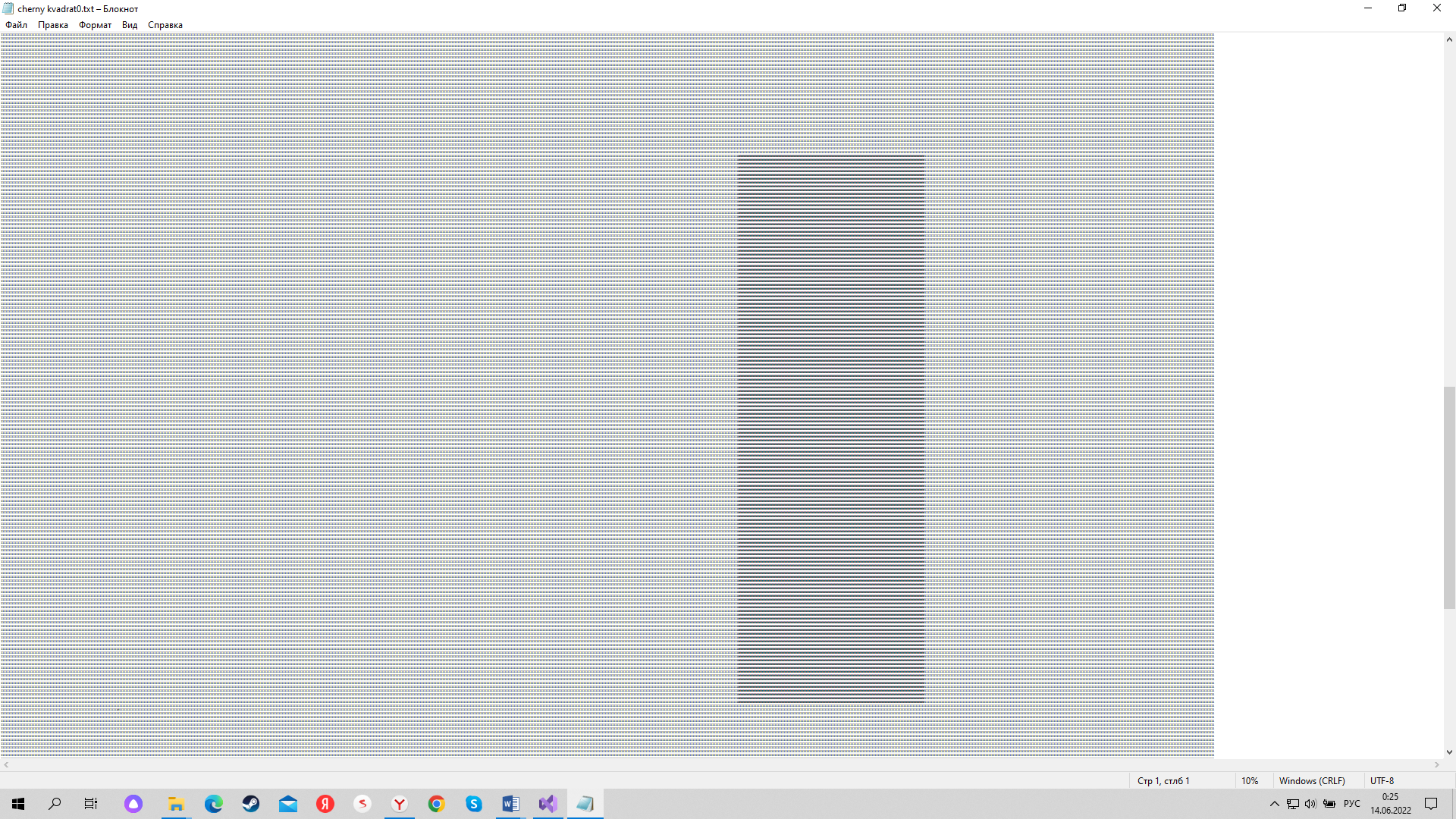


Рисунок 11 - Массив изображения в .txt файле

## **Разработка алгоритма для поиска квадрата.**

Процесс поиска квадрата реализуется в методе “FindQuad()”. Все происходит во время итерации массива “numsImage”, сначала двигается по строке потом спускается на нижнюю строку и т.д. До итерации создается переменная “checkQuad” типа “bool” равная “false”, которая получает значение “true” лишь при прохождении квадратом всех условий, после чего происходит выход из всех циклов. Первым условием для возможного нахождения квадрата является нахождение квадрата рядом с границей. Следующее условие — это значение 0 в текущей координате и координате “ (i+side - 1; j+side - 1)”, то есть проверяются нули в верхнем левом углу квадрата со стороной 2 и нижнем правом углу. [Рисунок 12]

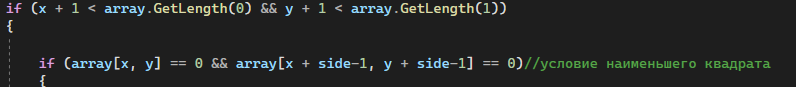


Рисунок 12 - Фрагмент кода Условие наименьшего Квадрата

Далее попадаем в бесконечный цикл “while” с постусловием для анализа квадрата. Внутри него первая проверка — это вновь выход за границу, ибо данное исключение часто вылетало в процессе тестирования. Следующим шагом от нижнего правого угла условно проводятся две линии вверх и влево и считаются суммы их значений. [*Рисунок 13*]

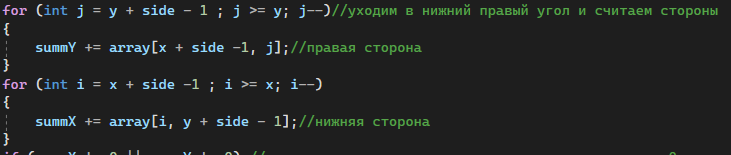


Рисунок 13 Фрагмент кода Проверка правой и нижней сторон

В качестве следующего условия для прохождения теста является равенство сумм значений нижней и правой сторон нулю. Если обе суммы равны нулю, то это означает, что можно попробовать взять квадрат побольше, сторона увеличивается и цикл начинается по новой. [Рисунок 14]



Рисунок 14 - Фрагмент кода Условие для увеличения стороны

Если же одна из сумм не равна 0, то значит у нас есть возможный квадрат, которому предстоит пройти проверку на окружение. При помощи циклов “for” происходит суммирование значений по периметру квадрата со стороной большей чем у текущего на единицу. [Рисунок 15]

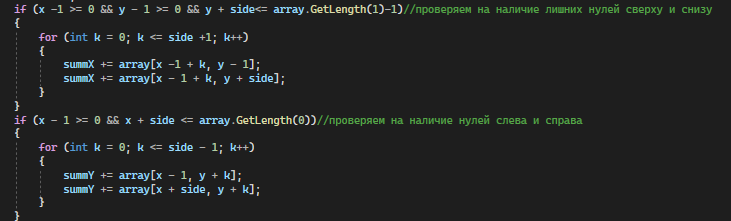


Рисунок 15 - Фрагмент кода Проверка на наличие нулей вокруг

После этого из вручную высчитанного при помощи математической формулы периметра окружающего квадрата мы вычитаем суммы этих значений. При идеальном варианте разность должна быть равна нулю, что означало бы, что вокруг квадрата только единицы. Однако, допускается присутствие не более 10 нулей вокруг. Кроме всего этого стоит ограничение на минимальный размер стороны квадрата 10 пикселей, как более или менее видимый человеческому глазу квадрат. После прохождения всех проверок выводится информация о квадрате, пикселям по контуру фигуры на изображении присваивается красный цвет, а переменная “checkQuad” принимает значение “true”. [Рисунок 16]

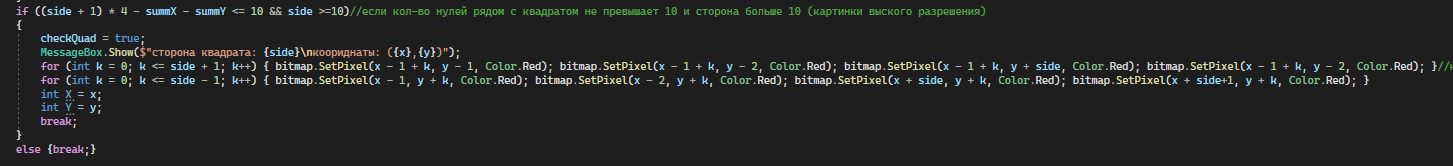


Рисунок 16 - Фрагмент кода Финальная проверка квадрата

1. **Тестирование приложений.**

Над разработанной системой был проведен ряд тестов, проверяющий работу переходов по сценам. То есть тестированию подлежали всплывающие окна (при выборе изображения, и выводе информации о квадрате) и работоспособность кнопок для перехода между этапами. Далее проверялись методы поиска квадрата и бинаризации изображения при загрузке изображений с разными форматами (после чего было решено установить фильтр на поиск картинок). Также проверялась работоспособность программы в спорных случаях расположения квадрата (квадрат с краю, пересечение двух квадратов, неровные края квадрата). Для проведения тестов с расположением фигуры был использован редактор Paint, как удобный инструмент для изменения фото.

## **Тестирование всплывающих окон**

В первую очередь были проведены тесты над всплывающим окном windows для выбора изображения, так как оно могло вызвать некоторые исключения. Одним из них было закрытие этого окна, которое не было предусмотрено в сценарии. После этого проблема была решена внедрением try catch, которое позволило возвращать главное меню при возникновении данной ошибки.

Следующая проблема была связана с изменением пропорций некоторых картинок при их загрузке в picturebox. Данный дефект был связан с разностью в размерах фотографии и picturebox, однако на него было принято решение закрыть глаза, так как оно не вызывает никаких исключений, но при исправлении потребовало бы пересмотреть интерфейс приложения.

Ниже приведена таблица возможных сценариев для выбора картинки:

**Таблица 2 - Тестирование загрузки изображения**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер** | **Сценарий** | **Ожидаемый результат** | **Реальный результат** | **Комментарии** |
| С1 | 1. Найти в файлах нужную картинку формата (bmp, png) 2. Выбрать ее | Картинка отобразится в приложении | <= | С картинкой появится красная стрелка для возврата и кнопка |
| С2 | 1. Закрыть окно выбора картинки 2. Вернуться в главное меню | Открытие главного меню и возобновление работы | <= | … |
| С3 | 1. Загрузить картинку низкого разрешения | Нормальное отображение картинки | Отображение картинки в пиксельном формате | Фото растягивается по всей форме |

## **Тестирование Переноса массива в txt**

Во время проведения тестов был замечен конфликт между кнопкой возврат на главное меню и записью массива в документ. После работы приложения приходилось каждый раз удалять txt файл, так как приложение еще не могло перезаписывать его. На помощь пришло свойство append у StreamWriter позволяющее перезаписывать файл. К сожалению, осталась другая проблема, не дающая увидеть изменения документа в реальном времени до его перезагрузки.

**Таблица 3 - Тестирование записи массива**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер** | **Сценарий** | **Ожидаемый результат** | **Реальный результат** | **Комментарии** |
| С1 | 1. Открыть текстовый файл с массивом после прошлой работы 2. Нажать кнопку “Find Quad” | Перезаписывание .txt файла на рабочем столе с массивом  Увидеть изменения в документе в реальном времени | Изменения не происходят | Изменения будут только после закрытия и повторного открытия файла, так как он пресоздается. |
| С2 | 1. Было загружено изображение без квадрата 2. Нажать кнопку “Find Quad” 3. Подождать не более 10 секунд 4. Открыть текстовый файл | Создание .txt файла на рабочем столе с записанным массивом | <= | … |

## **Тестирование поиска квадрата**

Для проведения данного рода тестов было задействовано множество изображений с различным расположением черного квадрата на них. Методом проб и ошибок код постепенно обрастал все большим количеством проверок, чтобы сейчас исключать практически все возможные ошибки. Самой частой проблемой было то, что код воспринимал другие маленькие объекты за нужный квадрат или вырезал квадрат внутри какой-то фигуры. Позже все это было исправлено.

Таблица 4 - Тестирование поиска квадрата

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер** | **Сценарий** | **Ожидаемый результат** | **Реальный результат** | **Комментарии** | **Комментарии** |
| Т1 | 1. Нажать кнопку “Find Quad” 2. Подождать не более 10 секунд 3. Получить информацию о квадрате 4. Увидеть текстовый документ с массивом | Показ небольшого окна с координатами и размером квадрата.  Вокруг квадрата красная рамка  Создание .txt файла на рабочем столе с массивом | <= | В зависимости от разрешения время ожидания меняется |
| Т2 | 1. Было загружено фото без квадрата. 2. Нажать кнопку “Find Quad” 3. Подождать не более 10 секунд 4. Получить информацию об отсутствии квадрата 5. Увидеть текстовый документ с массивом | Показ небольшого текстового окна “Квадрата нет”  Создание .txt файла на рабочем столе с массивом | <= | В зависимости от разрешения время ожидания меняется |
| Т3 | 1. Было загружено фото с двумя квадратами. 2. Нажать кнопку “Find Quad” 3. Подождать не более 10 секунд | Показ небольшого окна с координатами и размером первого попавшегося квадрата.  Вокруг квадрата красная рамка  Создание .txt файла на рабочем столе с массивом | <= | Задача предполагает наличие только одного квадрата |
| Т4 | 1. Было загружено изображение с очень большим разрешением и без квадрата 2. Нажать кнопку “Find Quad” 3. Подождать не более 10 секунд | Показ небольшого текстового окна “Квадрата нет”  Создание .txt файла на рабочем столе с массивом | Слишком долгое ожидание информации и нагрузка компьютера | Требуется аварийное закрытие |

# 

# Заключение.

В ходе выполнения работы цель – разработать приложение для поиска Черного Квадрата на изображении, была достигнута. Несмотря на то, что часто приходилось сталкиваться с трудностями во время создания приложения и освоения новой среды разработки Windows Forms, я считаю полученный опыт бесценным, и он обязательно пригодится в будущем. Кроме того, благодаря этому проекту мне удалось расширить свои знания в сфере анализа изображений и найти множество интересных статей для прочтения на эту тему.

Для достижения поставленной цели были выполнены следующие шаги:

1. Поиск информации о данной предметной области на тот момент будущего приложения для нахождения черного квадрата– программная среда Visual Studio 2022 на языке программирования C#.
2. Поиск алгоритмов для реализации базовых функций программы: преобразование фото в массив чисел, работа с этим массивом, перепись данного массива в текстовый файл.
3. Изучение материалов по работе в Windows Forms и самой теме курсовой, попытка разработать пробную версию приложения.
4. Выявление плюсов и минусов используемых алгоритмов в пробной версии приложения. Создание интерфейса для итоговой версии приложения.
5. Разработка итоговой версии с установленным ранее интерфейсом и с учетом устранения выявленных недочетов и недоработок алгоритмов из предыдущей версии.
6. Тестирование функций инструментов и алгоритмов в графическом редакторе.

Подводя итог, хочу сказать, что приложение соответствует всем ранее поставленным требованиям. В будущем его алгоритмы будут доработаны и возможно его “движок” будет использоваться для роботов, камер видеонаблюдения и т.д.

# Список литературы

1. [Компания Gil Algorithms](https://habr.com/ru/company/gilalgorithms/blog/). Очистка изображения от шума, некоторые методы [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/gilalgorithms/blog/67594/– (Дата обращения 15.05.2022)
2. Распознавание изображений: обзор лучших программ [Электронный ресурс]. – Режим доступа: <https://fb.ru/article/473366/raspoznavanie-izobrajeniy-obzor-luchshih-programm> – (Дата обращения 8.06.2022)
3. Бинаризация изображений: алгоритм Брэдли [(](https://habr.com/ru/hub/algorithms/)[Обработка изображений)](https://habr.com/ru/hub/image_processing/) [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/278435/ – (Дата обращения 8.06.2022)
4. Анализ изображений программ [Электронный ресурс]. – Режим доступа: https://translated.turbopages.org/proxy\_u/en-ru.ru.f4447726-62a61e5a-7b19c34c74722d776562/https/en.wikipedia.org/wiki/Computer\_image\_analysis– (Дата обращения 01.06.2022)
5. [ZlodeiBaal](https://habr.com/ru/users/ZlodeiBaal/)17. Пару слов о распознавании образов [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/208090/ – (Дата обращения 23.05.2022)
6. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB [2006] (Digital Image Processing using MATLAB) [Электронный ресурс]. – Режим доступа: https://studizba.com/files/sistemy-raspoznavaniya-obrazov/book/254893-gonsales-r.-vuds-r.-jeddins-s.-cifrovaja.html – (Дата обращения 20.05.2022)
7. Бутаков Е.А., Островский В.И., Фадеев И.Л. Обработка изображений на ЭВМ [книга] - М.: Радио и связь 1987. — 240 с. (Дата обращения 02.06.2022)
8. Jorgensen P.E.T. Analysis and Probability: Wavelets, Signals, Fractals [книга] - Springer, 2006. - 276 страниц. (Дата обращения 15.05.2022)
9. [iloveip](https://habr.com/ru/users/iloveip/). Как не облажаться с дизайном. Инструкция для чайников за 5 минут [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/iloveip/blog/328492/> – (Дата обращения 25.05.2022)
10. [Компания Joom](https://habr.com/ru/company/joom/blog/). Нахождение объектов на картинках [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/joom/blog/445354/ – (Дата обращения 27.05.2022)

# Приложение А. Сервис для определения объектов на фото

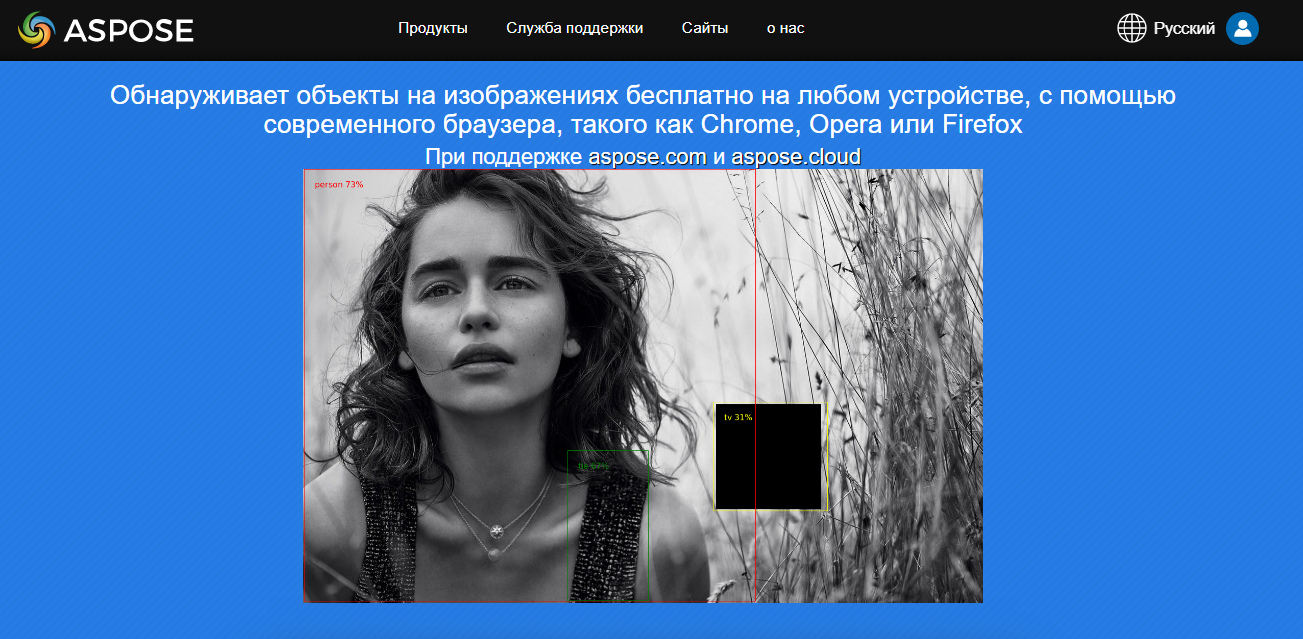


Рисунок А.1 Сервис для определения объектов на фото

# Приложение Б. Методы сегментации изображений



Рисунок Б.1 Бинаризация изображения

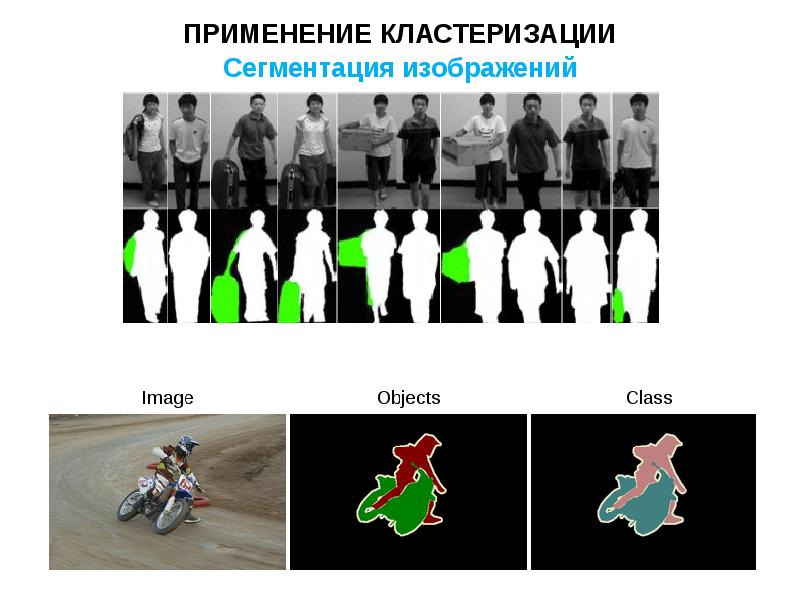


Рисунок Б.2 Пример Кластеризации

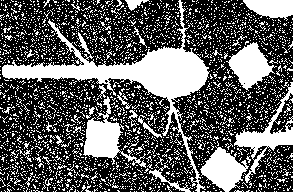


Рисунок Б.3 Пример шума на бинарном изображении

# Приложение В. Интерфейс приложения



Рисунок В.1 Изображение для главного фона

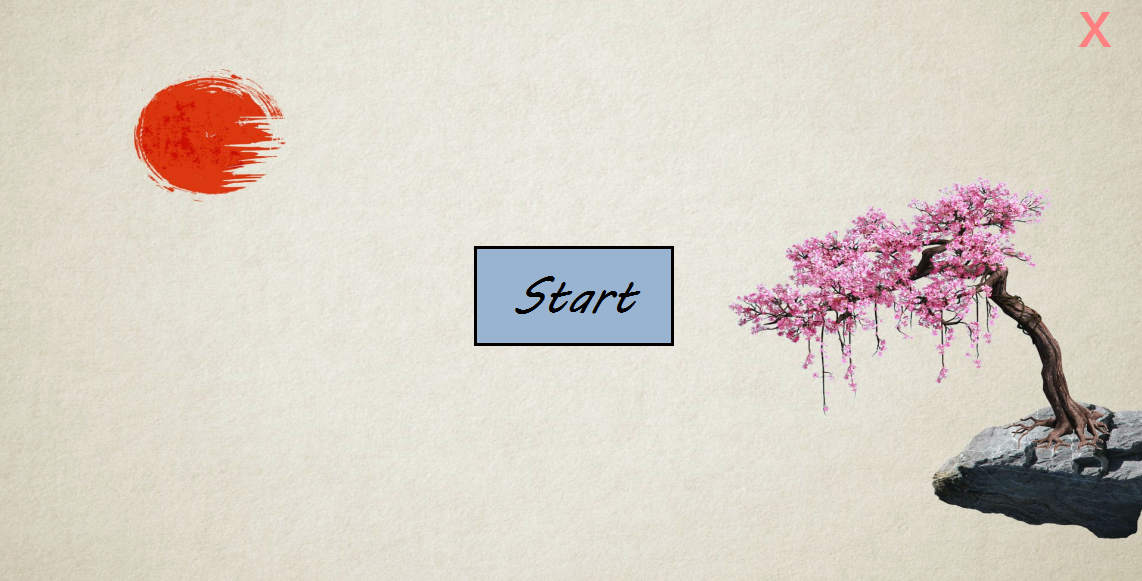


Рисунок В.2 Скриншот главного экрана

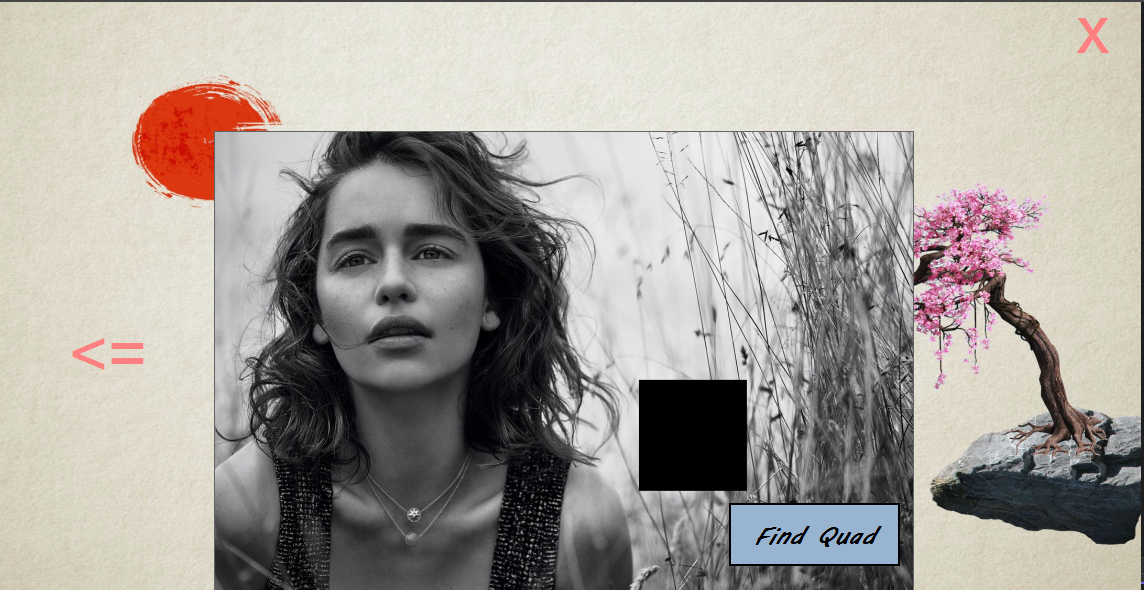


Рисунок В.3 Скриншот меню для работы с квадратом

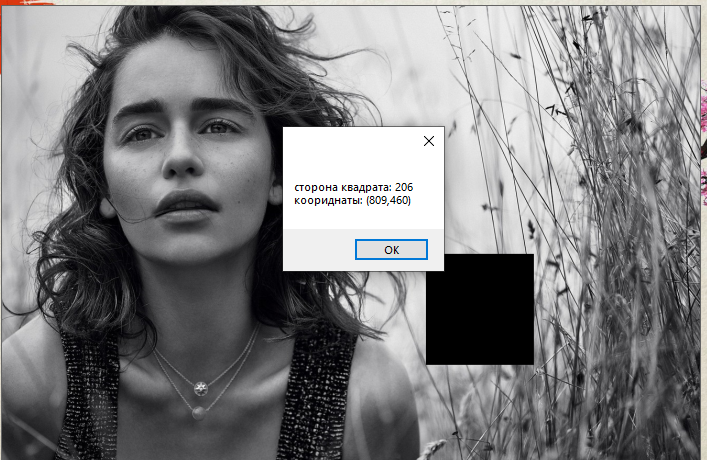


Рисунок В.4 Скриншот окна с информацией



Рисунок В.5 Фрагмент картинки с красной рамкой

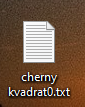


Рисунок В.6 Файл с массивом изображения