**Введение**

В современном мире накопились большие объемы рукописных и печатных книг, не имеющих цифровых аналогов. А так как мы живем в мире, где большинство процессов представляют из себя автоматизированные системы, то важным компонентом нашего существования является обеспечение целостности связи между поколениями в виде сохранности существующей информации. У людей мало времени и сил на копирование вручную. Поэтому сейчас как можно скорее необходимо задуматься об оцифровке, т.е. о переводе научно-технической, художественной литературы, архивных документов и других бумажных носителей информации в электронный вид. Электронная версия информации – это гарантия сохранности, несмотря на возможные бедствия. Электронные копии книг могут образовывать [электронные библиотеки](https://ru.wikipedia.org/wiki/Электронные_библиотеки) и распространяться в глобальной сети Интернет. Цифровые книги можно легко распространять, воспроизводить и читать на экране. Во всем этом помогают новомодные гаджеты, которые стали неотъемлемой частью нашей жизни. Большой функционал, мобильность, аккуратность и многое другое делают эти устройства лидерами рынка. С их помощью люди экономят время на поиск необходимой информации в библиотеках и архивах, соответственно, усваивают быстрее и больше нужного материала.

На сегодняшний момент времени существует методика оцифровки текста, но она, к сожалению, имеет свои недостатки. Процесс оцифровки включает в себя два подхода.

1. Обязательный: получение копий страниц в виде графических изображений, путем сканирования или фотографирования с последующей обработкой и сохранение в одном из форматов графических файлов. При этом оригинальная верстка книги сохраняется, исключаются разного рода ошибки.
2. Опциональный: применение технологии «оптического распознавания символов – OCR» с последующим сохранением распознанного текста в одном из форматов электронных книг. Открывается возможность полнотекстового поиска по книге и индексация больших массивов электронных книг, но затрудняется воспроизведение оригинальной верстки, изображений, схем и формул.

Тема оцифровки информации с бумажных носителей является невероятно актуальной. Доставая том книги с полки, мы даже не задумываемся, что держим в руках не просто вещь, а результат многовековой эволюции человеческой культуры. С древнейших времен человек стремился оставить за собой след. Первыми носителями информации стали стены пещер и камни. Создавались наскальные рисунки и петроглифы. Далее были придуманы глиняные и восковые таблички. И только после появления папируса была изобретена бумага. Теперь же главной целью у человечества должно быть формирование электронного фонда цифровой информации.

В соответствии с вышеизложенным, **цель работы** –

Объект исследования –

Предмет исследования –

Основные задачи, намеченные для достижения цели:

**ГЛАВА 1**

Итак, перед нами стоит задача распознать текст с изображения. Для обеспечения масштабируемости и читабельности имеет смысл разбить основной процесс на подзадачи, и реализовать каждую по-отдельности.

Исходя из нашей задачи можно выделить следующие этапы:

1. Перевод изображения в монохромный формат
2. Сегментация
3. Выравнивание
4. Распознавание

Разберем каждый из них.

**Перевод изображения в монохромный формат**

В основе этого этапа будет алгоритм, который анализирует каждый пиксель и перезаписывает его в одно из двух состояния: белый и черный (0 и 1 соответственно).

Разберем этот механизм подробнее. Известно, что стандартные форматы изображений, такие как “PNG” И “JPG” описывают каждый пиксель тремя каналами: красным, зеленым и синим. Каждый канал в свою очередь задается значением от 0 до 255. Это значение определяет меру присутствия соответствующего цвета в конечном цвете. Так, чтобы описать светлый оттенок, нам нужно установить как можно большее значение каждому каналу. Чистый белый цвет получается из значений 255, 255, 255 красного, зеленого и синего соответственно (R, G, B), а черный из значений 0, 0, 0. Пример цвета со значениями каналов приведены на рис. 1.

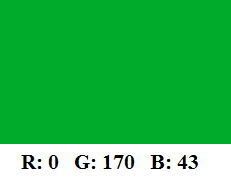


Рисунок . Пример цвета

Контуры символов соответствуют темным оттенкам, исходя из чего мы можем устанавливать каждому пикселю черный цвет (0,0,0), если значение каждого его канала будет меньше какого-либо заранее установленного предела и белый (255, 255, 255) в противном случае. Предел определяет границу между черным и белым и служит регулятором “чувствительности” алгоритма. Значение этого предела лежит в промежутке от 0 до 255. Целесообразней установить его в середине этого промежутка, но точное его значение определяется опытным путем при тестировании программы.

**Сегментация**

По итогу предыдущего алгоритма мы получим матрицу из нулей и единиц, описывающих черные и белые цвета соответственно. Перевод изображения в монохромный формат нужен прежде всего для корректной работы последующих этапов программы, одним из которых является сегментация.

Смысл этого этапа заключает в себя обрамление каждого символа и последующего сохранения этой информации в массив. Разберем этот процесс подробнее.

Одной из реализаций данного алгоритма является “раздувание” рамки при обнаружении закрашенных пикселей. Абстрагируемся от символов в пользу произвольных объектов и представим робота, который бегает по матрице и проверяет значение каждого пикселя. Обнаружив закрашенный, робот строит рамку вокруг этого пикселя (рис 2.).

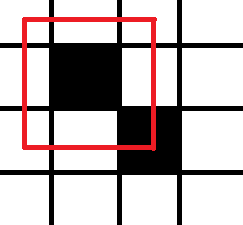


Рисунок . Построение рамки вокруг найденного пикселя

Далее робот проходит по периметру текущей рамки и анализирует каждый пиксель. Если на границе окажется закрашенный пиксель, робот расширит рамку, чтобы этот пиксель оказался внутри. Затем цикл повторяется, и так, пока на рамке не окажется закрашенных пикселей. В итоге для предыдущего примера картина окажется следующей (рис 3).

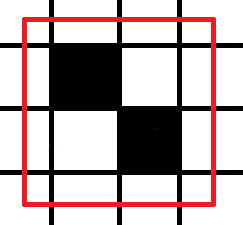


Рисунок . Результат работы алгоритма

Таким образом, каждый объект на изображении будет обрамлен. Вот живой пример заключения в рамки нескольких объектов на изображении (рис 4.)

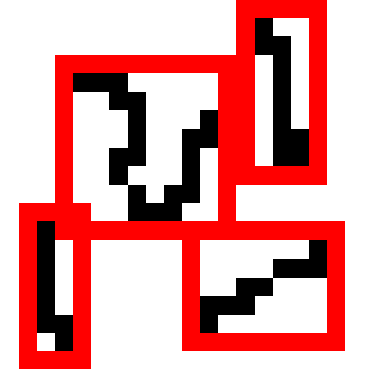


Рисунок . Пример работы алгоритма

Нам останется лишь сохранить информацию о каждом прямоугольнике, а именно позицию по ширине, позицию по высоте, ширину и высоту прямоугольника, в массив.

**Выравнивание**

В результате работы предыдущего этапа мы получаем карту изображения, которая описывает местоположение каждого объекта. Но давайте вернемся от объектов к нашей первоначальной задаче: символам. Тут нас ожидает проблема. Дело в том, что робот из предыдущего алгоритма, выполняя свою задачу, спускается вниз строка за строкой. И так как символы имеют разные размеры по высоте, то распознавание будет вестись хаотично, а если конкретней, в порядке уменьшения высоты символов. В результате ответ программы будет содержать набор букв, никак не связанный со словами на исходном изображении (рис 5.).

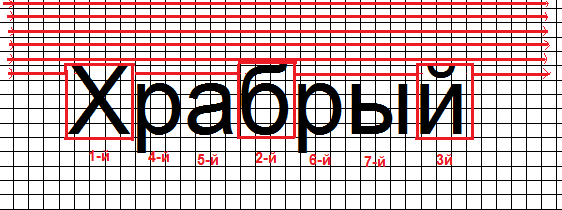


Рисунок . Результат программы без выравнивания

Так, для примера на рисунке 5 результатом программы будет “Хбйрары”, что, конечно же, неправильно.

Чтобы решить эту проблему нужно сделать выравнивание сначала построчно, затем посимвольно для каждой строки. Таким образом этот этап разбивается на две подзадачи. Разберем каждый по-отдельности.

Выравнивание по строкам можно реализовать путем сравнения уровней высоты двух идущих друг за другом букв. Рисунок 6 иллюстрирует данную методику.

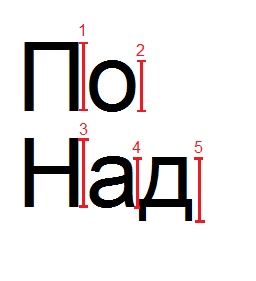


Рисунок . Способ выравнивания по строкам

Символы 1 и 2 располагаются на одном уровне, поэтому относим их к одной строке. А вот 2 и 3 находятся на разных уровнях, от чего мы образно переносим символ 3 на следующую строку и продолжаем сравнивать уровни следующих символов. В результате для примера из рисунка 6 мы получим две строки. Для структуризации всей собранной информации имеет смысл хранить контейнеры – строки, в которых мы и будем помещать символы с соответствующими номерами строк.

Когда первый под-этап отработал, и мы имеем структуру из строк нам не составит труда отсортировать символы по горизонтали в каждой строке. Для этого мы можем воспользоваться методом пузырька.

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются {\displaystyle N-1} раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

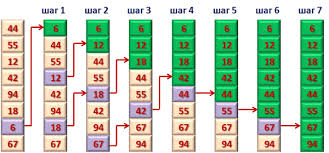


Рисунок . Сортировка методом пузырька

В нашем случае объектами сравнения будут позиции символов по горизонтали. По итогу этого этапа мы получим структуру строк и включенных в них в первозданном порядке символов.

**Распознавание**

Теперь, когда все готово, мы можем приступить к основному процессу, а именно к распознаванию каждого символа. Есть множество технологий, которые позволяют осуществить эту задачу.