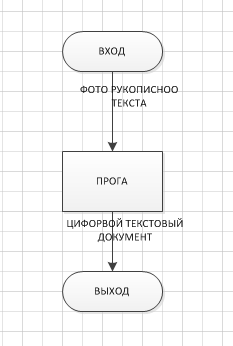
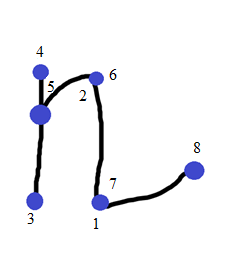
Задача: преобразовать рукописный текст на фото в цифровой текст.



**Шаблон буквы**

Шаблон представляет собой ***вектор положений*** характерных точек буквы относительно друг друга. Точки определяются положением выше или ниже относительно остальных. Первой точкой буквы считается самая правая характерная точка.

Например, для буквы «п» вектор положений будет выглядеть вот так:



В – выше, Н – ниже, Р – рядом (наравне, выше либо ниже).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | Н | Р | Н | Н | Н | Н | Н |
| 3 |  | В | Р | Р | Н | В | Р |
| 4 |  |  | Н | Н | Н | Р | Р |
| 5 |  |  |  | В | Р | В | Р |
| 6 |  |  |  |  | Н | Р | Р |
| 7 |  |  |  |  |  | В | Р |
| 8 |  |  |  |  |  |  | Н |

В длину:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2Н | Р | Н | Н | Н | Н | Н | 3В | Р | Р | Н | В | Р | 4Н | Н | Н | Р | Р | 5В | Р | В | Р | 6Н | Р | Р | 7В | Р | 8Н |

Вектор будет являться ***уникальным идентификатором*** символа.

**Распознаем букву**

1. Делаем шаблон
   1. Шаблон состоит из имени буквы и вектора положений
   2. составляем XML
   3. загружаем шаблоны из XML
2. Получаем изображение буквы на вход
   1. Загружаем изображение из файла
3. Обрабатываем изображение буквы
   1. Бинаризуем
   2. *удаляем шумы (чёрные точки)?*
4. Ищем характерные точки
   1. Ищем первую точку справа, потому что «хвост буквы» найти проще (первый экстремум)
   2. Проходим по контуру буквы, ищем локальные экстремумы, сохраняем их координаты и порядковый номер (точки буквы)
5. Сравниваем точки буквы с шаблоном
   1. по количеству точек
      1. Ищем среди шаблонов тот, размерность которого совпадает с кол-вом точек
   2. по положениям точек относительно друг друга
      1. для лучшего распознавания переопределить позицию «рядом», как +- половина размера буквы (размер определяется как «max(вектор точек) – min(вектор точек)» ), и добавить позицию «не знаю» (W – whatever), которая не проверяется
      2. для овальных экстремумов точка снизу всегда будет левее точки сверху, только надо уметь находить, что эти две точки принадлежат одному экстремуму
6. Выводим результат
   1. Имя буквы

**Распознаем слово**

1. Получаем изображение слова на вход
   1. Загружаем изображение из файла
2. Разделяем на буквы
   1. Проходим по изображению, начиная с конца по низу
   2. Находим точку экстремума-максимума
   3. Разделяем слово в этой точке
   4. двигаемся дальше по полученному куску, получая верхние характерные точки
   5. пытаемся распознать кусок как букву
   6. возвращаемся в точку разреза
   7. если кусок распознан как буква, то к п. b
   8. иначе
      1. если кусок не распознан 3 раза
         1. если позади нет букв длиной меньше 3, то оставляем букву пустой
         2. иначе
            1. возвращаемся к концу предыдущей буквы,
            2. удаляем верхние экстремумы и к п. b
      2. иначе, удаляем верхние экстремумы и к п. b
   9. если после точки разреза белое поле, то стоп
3. Выводим слово из букв
4. Ищем слово в словаре
5. если не найдено, то
   1. предложить варианты исправления?
   2. пытаться распознать заново?

Класс «Буква»

имя: string

координаты конца буквы: point

кол-во верхних точек: int

ширина (кол-во макс-экстремумов снизу): int

**Фичи**

1. Распознавание заглавных букв
2. Распознавание знаков препинания