МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский

технический университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

Кафедра автоматизированных систем обработки информации и управления

09.03.02 «Информационные системы и технологии»

Курсовая работа

по дисциплине: «Цифровая обработка изображений»

на тему: программа определения заполненности полей в печатном   
документе

Обучающийся 4309 Погасий К.В.

(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель: доцент Медведев М.В.

Курсовая работа зачтена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

(подпись, дата)

Казань 2018

MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION

federal State Budget Educational Institution of Higher and Professional Learning

«Kazan National Research Technical University named after A.N. Tupolev»

(KNITU-KAI)

Institute of computer technology and information security

Department of automated information processing and control systems

09.03.02 " Information systems and technologies»

Coursework

For Digital image processing

on the topic: the field fill checker program

Completed 4309 Погасий К. В.

(group number) (signature, date)

Adopted: associate Professor Медведев М.В.

Coursework credited with the assessment

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

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

(signature, date)

Kazan 2018

**аннотация**

**аннотация**

**СОДЕРЖАНИЕ**

|  |
| --- |
| 1. Постановка задачи……………………………………………………… |
| 2. Описание решения задачи |
| 2.1. Общий алгоритм решения задачи |
| 2.2. Методы цифровой обработки изображений, применяемые для решения поставленной задачи |
| 2.2.1. Функции двустороннего сглаживания |
| 2.2.2. HSV – компоненты |
| 3. Спецификация программы |
| 3.1. Инструкция по работе с приложением |
| 3.2. Внутренне представление программы |
| 3.3. Проверка достоверности результатов работы приложения |
| 4. Заключение |
| 5. Используемые источники |
| 6. Приложения |

**1. Постановка задачи.**

Задание №13: программа определения заполненности полей в печатном документе.

Программа позволяет пользователю обрабатывать заполненные вручную и отсканированные документы, определяя какие поля в документе заполнены, а какие нет. После запуска программы появляется окно, отображающее исходный документ и командное окно, в которое выводятся результаты проверки. При решении поставленной задачи были приняты следующие допущения: программа использует документы, созданные по специальному шаблону: пункты, обязательные для заполнения, должны обозначаться нижним подчеркиванием и начинаться с отступом 0,5 от края листа. Остальной текст документа должен начинаться с отступом не менее 1,5. Межстрочный интервал должен составлять 1.15. Так же, необходимо контролировать правильную ориентацию загружаемого документа.

**2. Описание решения задачи**

**2.1. Общий алгоритм решения задачи**

Для решения поставленной задачи был разработан и реализован алгоритм определения заполненности полей. В основе алгоритма лежит последовательное применение методов цифровой обработки изображений, а так же попиксильный анализ.

**Алгоритм:**

1.Загрузка отсканированного изображения в программу.

2.Биннаризация изображения.

4.Подсчет количества черных точек в каждой строке.

5.Поиск обязательных для заполнения полей.

6.Проверка заполнения полей.

7.Вывод результата в командную строку.

На рисунке 1 приведена блок-схема данного алгоритма.

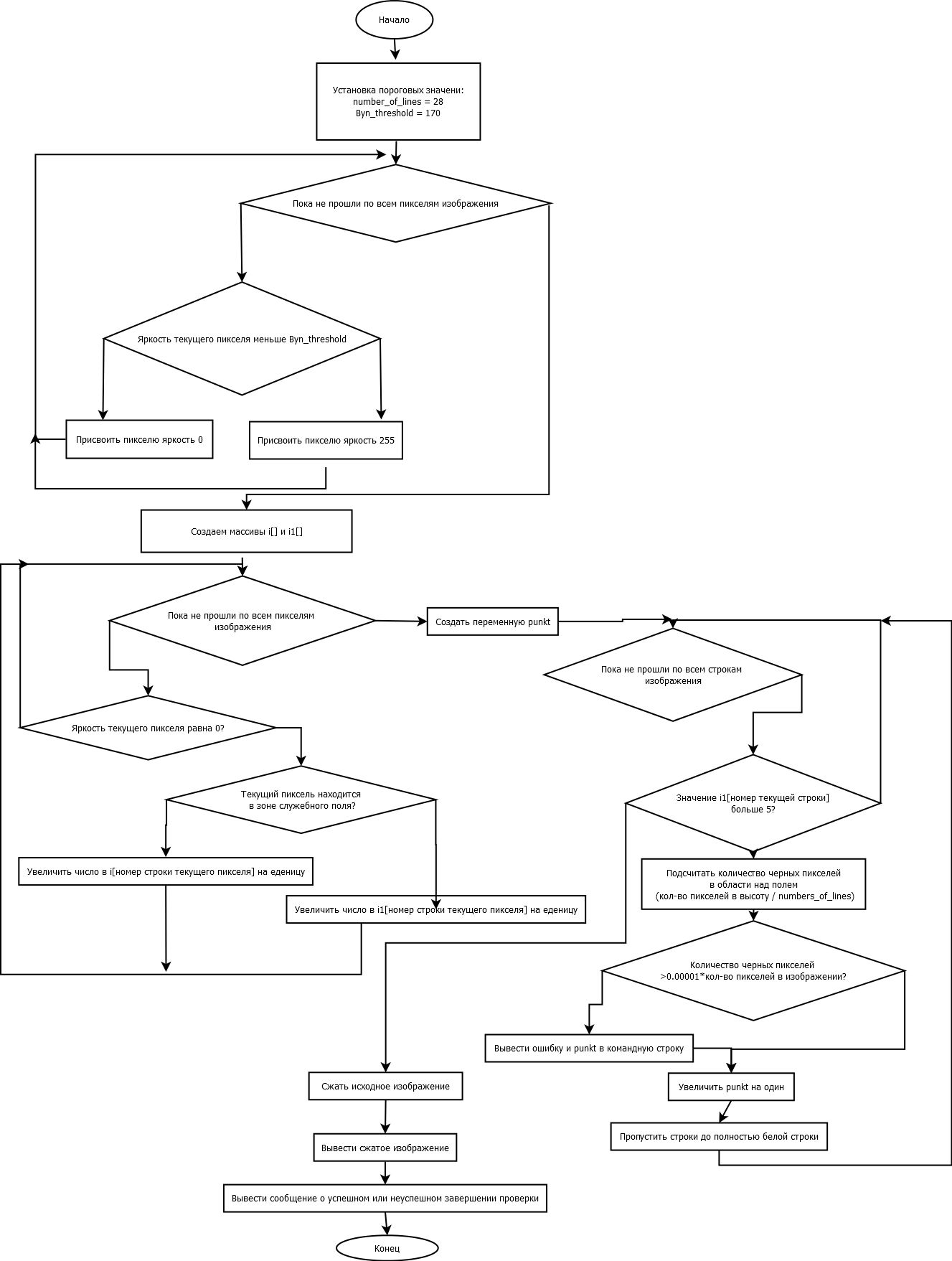


Рис 1. Блок-схема алгоритма

**2.2. Методы цифровой обработки изображений, применяемые для решения поставленной задачи**

Для решения поставленной задачи, был использован метод бинаризации, благодаря которому удалось отсечь серые полутона, появляющиеся при сканировании. Суть метода заключается в том, что каждая точка становится либо абсолютно черной (яркость равна 0), если исходная яркость ниже порогового значение, либо абсолютно белой (яркость равна 255), если исходная яркость выше порогового значения. В качестве порогового значения опытным путем было выбрано 170.

Так же, для увеличения удобства пользования программой, необходимо было сжать до размеров экрана пользователя входное изображение, поступающее зачастую в высоком разрешении, в связи с использованием современных сканеров. Для сжатия была применена библиотечная функция resize с параметром сжатия CV\_INTER\_CUBIC, позволяющим провести сжатие методом кубической интерполяции, дающим минимальные потери при сжатии, благодаря расчетам на основе 16 соседних точек. Такой метод дает приемлемое качество картинки и плавность линий со сжатием до 60%, что особенно важно для изображений, с которыми работает алгоритм.

**2.3 Алгоритмы, применяемые для решения поставленной задачи**

**2.3.1 Алгоритм поиска полей**

Для поиска полей был применен анализ заполнения части строки. Метод основан на подсчете черных пикселей в левой буферной области документа (красная зона на рисунке 2). Строка считается полем, если количество черных точек в ней превышает пороговое значение. Опытным путем было подобранно пороговое значение в 5 черных пикселей, для отсечения случайных точек.

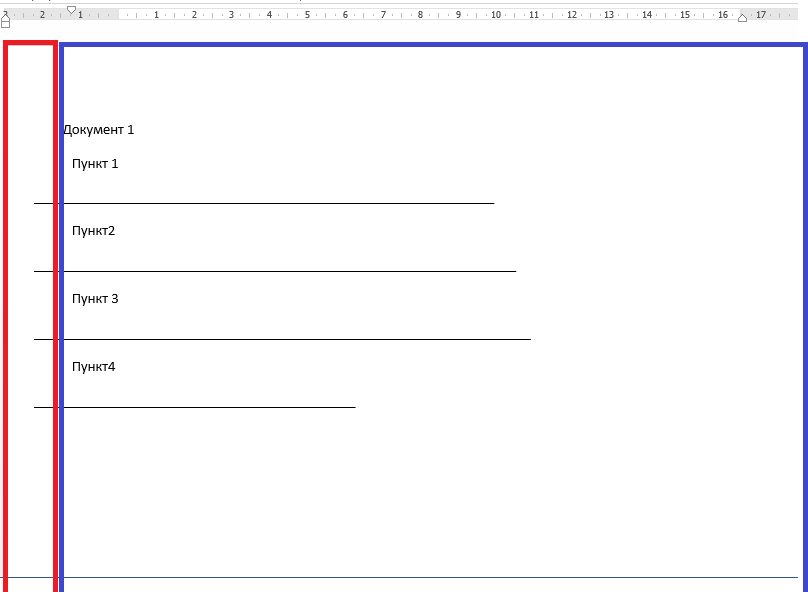


Рис. 2 Схема распознавания документа

**2.3.2 Алгоритм определения заполненности полей**

Для определения заполненности поля, по аналогии с поиском полей, был применен анализ количества черных точек в области поля. Областью поля будем называть область над линией, обозначающей начало поля, где нижней границей является сама линия, боковыми границами – границы области документа, исключая левое служебное поле, а верхняя граница рассчитывается по формуле y - image->height / Number\_of\_lines, где y – номер пикселя нижней границы, image->height - высота изображения в пикселях, а Number\_of\_lines – количество новых строк умещающихся на листе (опытным путем установлено, что при оговоренных параметрах текста, это число равно 28). После подсчета, количество черных точек сравнивается с пороговым значением, вычисленным по формуле q <= image->height\*image->width\*0.00001 , где image->height\*image->width – количество пикселей в изображении, а 0,00001 – величина, обратная количеству символов на одной странице текста, при оговоренных параметрах и шрифте до 11pt.

**3. Спецификация программы**

**3.1. Инструкция по работе с приложением**

Для работы с приложением необходимо:

1. Скачать и распаковать архив с программой в нужную директорию;

2. Открыть программный файл;

3. Запустить программу;

После запуска программы пользователь увидит исходное сжатое изображение и командную строку, в которую будут выводиться результаты проверки. Результаты работы программы представлены на рисунке 3 и 4

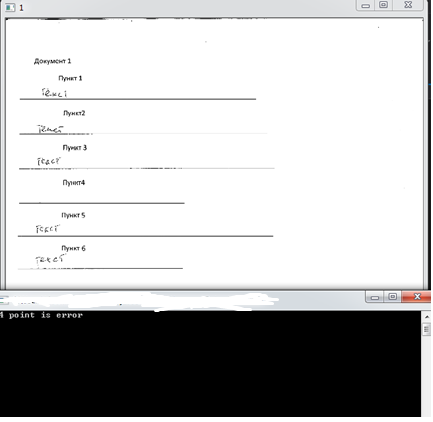


Рис. 3. Результат работы программы

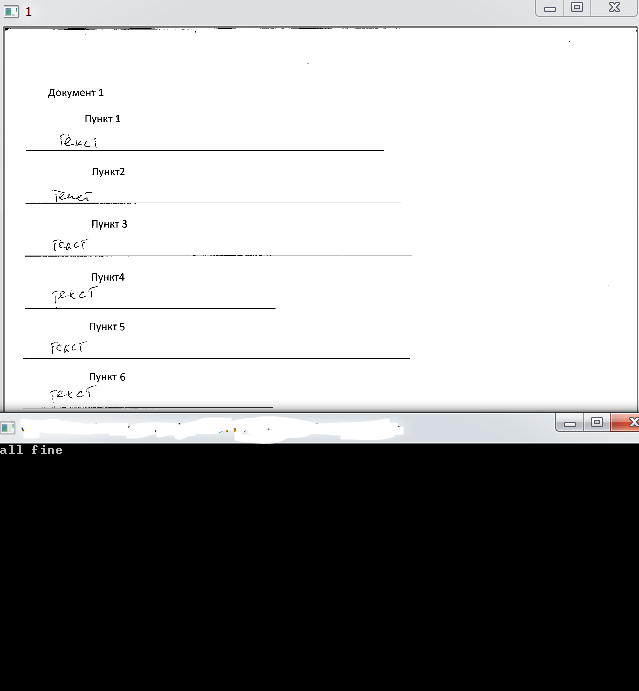


Рис. 4. Результат работы программы

**3.2. Внутренне представление программы**

1. Объявляем ссылку:

IplImage\* image;

2. Задаем ключевые значения:

int number\_of\_lines = 28;

int Byn\_threshold = 170;

3. Загружаем изображение:

image=cvLoadImage("C:\\Users\\kirushka\\Documents\\учеба\\ПЧЕЛКА\\курсач\\Doc2.png",CV\_LOAD\_IMAGE\_GRAYSCALE);

4. Создаем и обнуляем массивы

int \*i = new int[image->height];//для основного поля документа

int \*i1 = new int[image->height];//для служебного поля документа

for (int k = 0; k < image->height; k++)

{

i[k] = 0;

i1[k] = 0;

}

5.Проводим бинаризацию попиксельно

for (int y = 0; y < image->height; y++)

{

unsigned char\* ptr = (unsigned char\*)(image->imageData +

y \* image->widthStep);

for (int x = 0; x < image->width; x++)

{

if (ptr[x] < Byn\_threshold)

ptr[x] = 0;

else

ptr[x] = 255;

}

}

6. Считаем черные точки в каждой строке

for (int y = 20; y < image->height-20; y++)

{

unsigned char\* ptr = (unsigned char\*)(image->imageData +

y \* image->widthStep);

for (int x = 20; x < image->width-20; x++)

{

if (ptr[x] == 0)

i[y]++; //для основного поля

if (x>(image->width / 100) \* 0.15 && x < (image->width/100)\*6 && ptr[x] == 0)

i1[y]++; //для служебного поля

}

}

7. Проверяем поля на заполнение

7.1. Проходимся по служебному массиву:

for (int y = 1; y < image->height; y++)

{

if (i1[y] > 5)

{ punkt++;

7.2. При нахождении поля проверяем заполнение:

7.2.1 Считаем количество точек в области поля

int q = 0;

for (int k = y-3; k > (y - image->height / number\_of\_lines +30); k--)

q += i[k];

7.2.2 Сравниваем с порогом и выводим сообщение об ошибке, если недостаточно черных точек

if (q <= image->height\*image->width\*0.00001)

{

char w[5];

\_itoa(punkt, w, 10);

check = true;

printf("%d point is error\n", punkt);

7.3 Пропускаем строки до следующей пустой строки

while (i[y] > 2)

{

y++;

}

continue;

}

else

while (i[y] > 10)

{

y++;

}

}

}

8 Выводим сообщение об успешной проверке, если ошибок не найдено

if (!check)

{

printf("all fine\n");

}

9 Сжимаем и выводим исходное изображение

IplImage\* image1 = cvCreateImage(cvSize(image->width / 4, image->height / 4), image->depth, image->nChannels);

cvResize(image, image1, CV\_INTER\_CUBIC);

cvShowImage("1", image1);

**3.3. Проверка достоверности результатов работы программы**

Для проверки результатов работы программы было распечатано, заполнено экспертом с разной степенью полноты и отсканировано 10 бланков. После результаты работы программы были сравнены с изначальным планом заполнения эксперта. Совпадение пунктов, не заполненных по плану и обнаруженных программой означало успешное прохождение теста.

В таблице 1 приведено сравнение плана, составленного экспертом и результатов работы программы.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Незаполненные пункты (Эксперт) | Незаполненные пункты  (программа) | Количество ошибок |
| 1 | 2 | 2 | 0 |
| 2 | - | - | 0 |
| 3 | 3 | 3 | 0 |
| 4 | 3,4 | 3,4 | 0 |
| 5 | - | - | 0 |
| 6 | 3 | 3 | 0 |
| 7 | 1,5,6 | 1,5,6 | 0 |
| 8 | 2 | 2 | 0 |
| 9 | 4 | 4 | 0 |
| 10 | 1,2,3,4,5,6 | 1,2,3,4,5,6 | 0 |

Согласно результатам проверки, программа не допустила ни одной ошибки, что говорит о корректности работы программы.

**Заключение**

В данной курсовой работе была разработана программа, которая позволяет определять какие поля не были заполнены в отсканированном документе.

Для проверки достоверности результатов работы программы была произведена обработка десяти отсканированных документов, заполненных экспертом по определенному плану и сверка результатов работы программы с планом эксперта.

Согласно результатам работы подпрограммы, отраженных в таблице 1, программа по определению незаполненных полей работает корректно.

**Conclusion**

In this course work it was developed a program that allows you to find field, what didn’t fill.

To verify the reliability of the program’s results, ten scanned documents were processed that were filled in by the expert according to a certain plan and the program’s results were checked against the expert’s plan.According to the results of the subprogram, reflected in table 1, the program for finding natural objects (rivers, fields) works correctly.

According to the results of the subprogram, reflected in table 1, the field fill checker program works correctly.

**4. Используемые источники**

1. Пространственные методы обработки изображений: пространственная фильтрация: методическое пособие к лабораторной работе №3 по дисциплине «Цифровая обработка изображений» / М.В. Медведев, М.П. Шлеймович. – Казанский национальный исследовательский университет им. А.Н. Туполева, 2012.

2. OpenCV [Электронный ресурс]. - https://opencv.org/

3. С.П. Шарый Курс вычислительных методов [Электронный ресурс]. - http://www.ict.nsc.ru/matmod/files/textbooks/SharyNuMeth.pdf

4. Пиксельное представление изображений. [Электронный ресурс]. - https://ru.wikipedia.org

5. Адриан Келер, Гэри Брэдски. Изучаем OpenCV 3. М.: Изд-во ДМК Пресс, 2017 г. – 826 с.

**Приложение 1 Листинг программы**

#include <opencv2/opencv.hpp>

#include <iostream>

#include <windows.h>

#include <opencv2/highgui/highgui.hpp>

#include <iostream>

#include <conio.h>

#include <stdio.h>

#include "opencv2/imgproc/imgproc\_c.h"

#include <string.h>

#include <opencv2/highgui/highgui.hpp>

#include <iostream>

#include <conio.h>

#include <stdio.h>

#include <math.h>

#include <atlbase.h>

using namespace cv;

IplImage\* image;

int number\_of\_lines = 28;

int Byn\_threshold = 170;

int main(int argc, char\* argv[])

{

cvNamedWindow("1", CV\_WINDOW\_AUTOSIZE);

image = cvLoadImage("C:\\Users\\kirushka\\Documents\\учеба\\ПЧЕЛКА\\курсач\\Doc2.png", CV\_LOAD\_IMAGE\_GRAYSCALE);

int \*i = new int[image->height];

int \*i1 = new int[image->height];

for (int k = 0; k < image->height; k++)

{

i[k] = 0;

i1[k] = 0;

}

//Бинаризация

for (int y = 0; y < image->height; y++)

{

unsigned char\* ptr = (unsigned char\*)(image->imageData +

y \* image->widthStep);

for (int x = 0; x < image->width; x++)

{

if (ptr[x] < Byn\_threshold)

ptr[x] = 0;

else

ptr[x] = 255;

}

}

//выборка по строкам

for (int y = 20; y < image->height-20; y++)

{

unsigned char\* ptr = (unsigned char\*)(image->imageData +

y \* image->widthStep);

for (int x = 20; x < image->width-20; x++)

{

if (ptr[x] == 0)

i[y]++;

if (x>(image->width / 100) \* 0.15 && x < (image->width/100)\*6 && ptr[x] == 0)

i1[y]++;

}

}

int punkt = 0;

bool check = false;

//определение заполнения

for (int y = 1; y < image->height; y++)

{

if (i1[y] > 5)

{

punkt++;

int q = 0;

for (int k = y-3; k > (y - image->height / number\_of\_lines +30); k--)

q += i[k];

if (q <= image->height\*image->width\*0.00001)

{

char w[5];

\_itoa(punkt, w, 10);

check = true;

printf("%d point is error\n", punkt);

while (i[y] > 2)

{

y++;

}

continue;

}

else

while (i[y] > 10)

{

y++;

}

}

}

if (!check)

{

printf("all fine\n");

}

IplImage\* image1 = cvCreateImage(cvSize(image->width / 4, image->height / 4), image->depth, image->nChannels);

cvResize(image, image1, CV\_INTER\_CUBIC);

cvShowImage("1", image1);

waitKey();

cvDestroyAllWindows();

return 0;

}

**Приложение 2. Скриншоты изображений**

|  |  |
| --- | --- |
|  | |
|  | |
|  |  |
| 1 тест | 2 тест |
|  |  |
| 3 тест | 4 тест |
|  | |
|  | |
|  |  |
| 5 тест | 6 тест |
|  |  |
| 7 тест | 8 тест |
|  | |
|  | |
|  |  |
| 9 тест | 10 тест |
|  |  |

**Приложение 3. Пример шаблона для заполнения**

Документ 1

Пункт 1

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

Пункт 2

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

Пункт 3

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

Пункт 4

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

Пункт 5

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

Пункт 6

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