

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

**Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики**
Факультет Фотоники и оптоинформатики
Кафедра Компьютерной фотоники и видеоинформатики

Отчет по практике

Выполнила:
Старобыховская А. А.

Группа: V3316

Преподаватель: Кудрявцев А. С.

Санкт-Петербург, 2017

Оглавление

Цель проведения практики	3
Задание №1. Знакомство с системой контроля версий Git	4
Задание №2. Форматирование и стиль.	12
Задание 3. TDD: Разработка через тестирование.....	16
Проект на тему «Спектры. Сравнение».....	20
Приложение А. Слайды презентации с семинара по C++11 на тему: “User-defined literals”.	31

Цель проведения практики

Освоение навыков использования C++ и изучение приемов разработки программного обеспечения. Практика проходит в компьютерном классе и состоит из лекционных занятий и практических заданий.

Задание №1. Знакомство с системой контроля версий Git

1. Завести аккаунт на github.com. На рис.1 скриншот с сайта github.com.

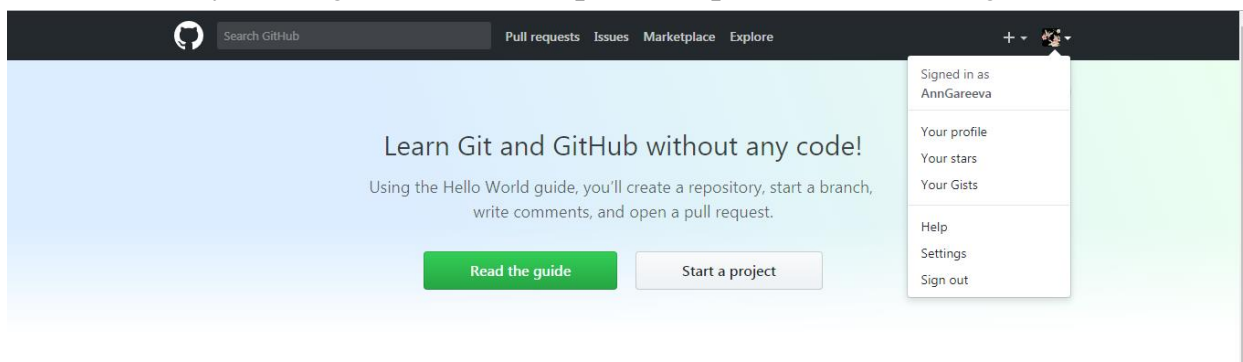


Рисунок 1- создание своего аккаунта на сайте github.com

2. Создать репозиторий для лабораторных работ. Рис.2 демонстрирует процесс создания репозитория, а рис.3 факт наличия его.

A screenshot of the "Create a new repository" form on GitHub. The form includes fields for "Owner" (AnnGareeva) and "Repository name" (PracticeAAS). Below these is a "Description (optional)" field containing "Lab of practice 2017". The "Visibility" section has "Public" selected. There is an option to "Initialize this repository with a README". At the bottom, there are dropdowns for ".gitignore" (None) and "License" (None), followed by a "Create repository" button.

Рисунок 2-создание репозитория на сайте

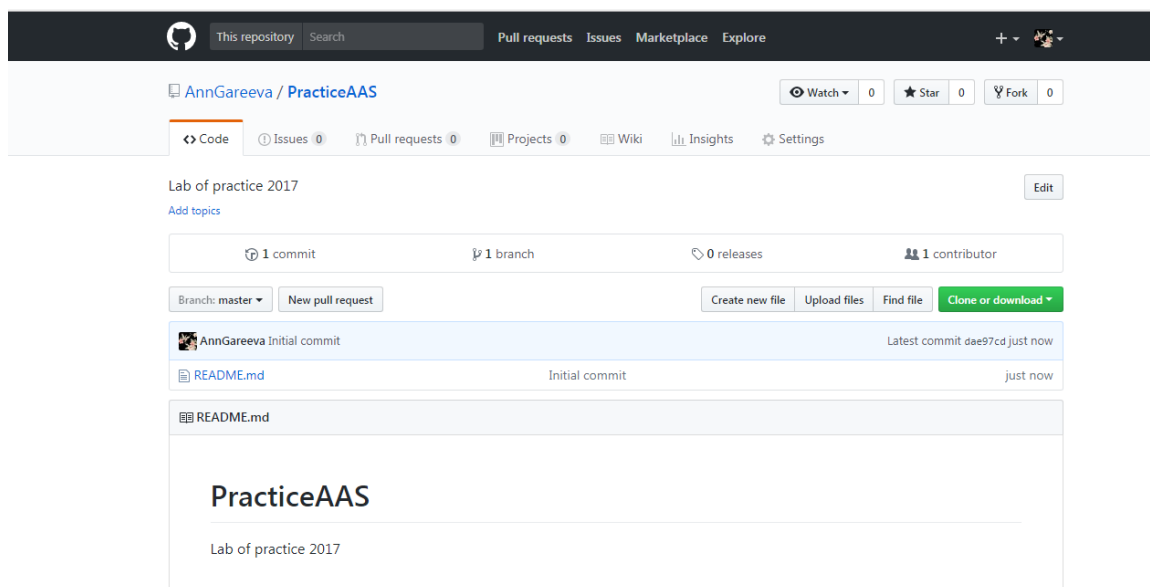


Рисунок 3-создание репозитория на сайте

3. Склонировать репозиторий на компьютер.

Для того, чтобы клонировать созданный репозиторий, необходимо скопировать на него ссылку. Именно это демонстрирует рис. 4.

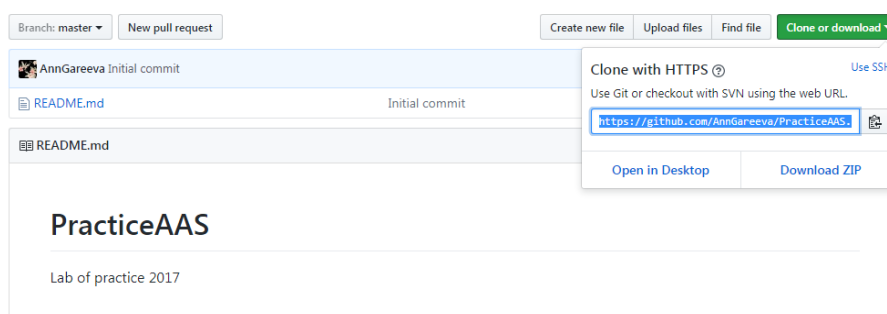
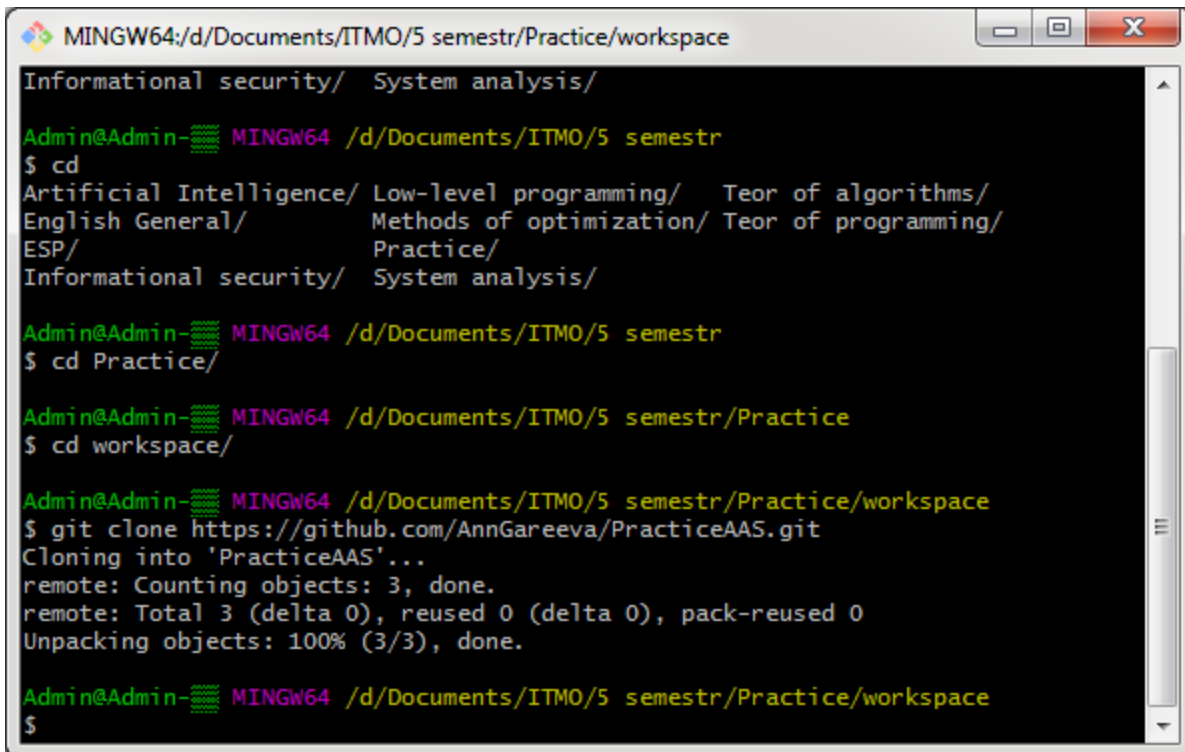


Рисунок 4-копирование ссылки на репозиторий для его клонирования

Скриншот Git-Bush показывает результаты выполнения команды `git clone` [<ссылка>](#) (рис. 5)



```

MINGW64:/d/Documents/ITMO/5 semestr/Practice/workspace
Informational security/ System analysis/

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr
$ cd
Artificial Intelligence/ Low-level programming/ Teor of algorithms/
English General/ Methods of optimization/ Teor of programming/
ESP/ Practice/
Informational security/ System analysis/

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr
$ cd Practice/

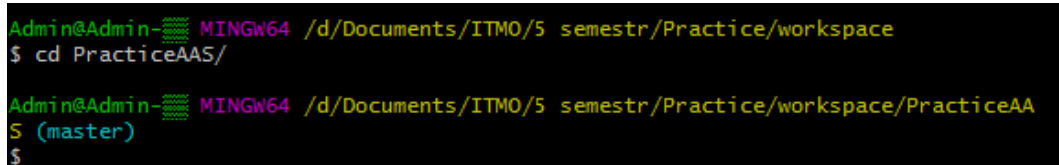
Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice
$ cd workspace/

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace
$ git clone https://github.com/AnnGareeva/PracticeAAS.git
Cloning into 'PracticeAAS'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace
$
  
```

Рисунок 5-результат выполнения команды `git clone`

После перехода в клонированную папку появляется `(master)`, дающий нам понять, что мы находимся и работаем в главной ветке. Это показано на рис.6.



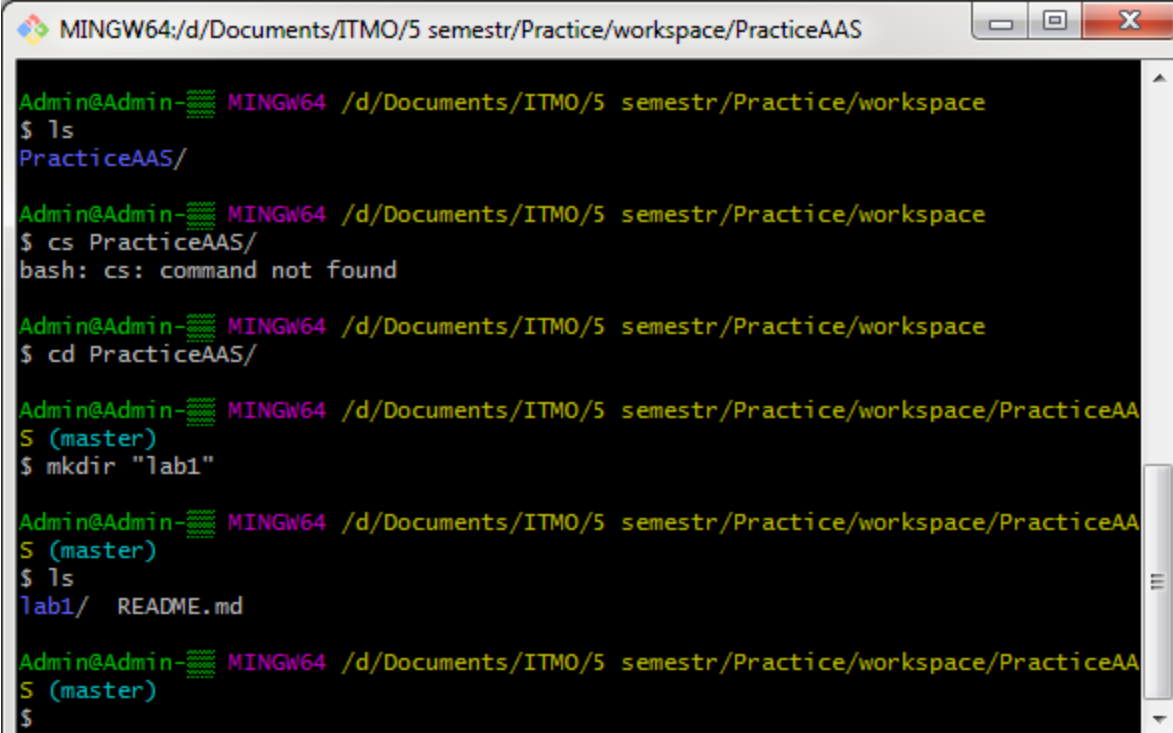
```

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace
$ cd PracticeAAS/

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAA
$ (master)
$
  
```

Рисунок 6- ветка master

4. Создать папку lab1. С помощью команды mkdir “название”, создаем папку (рис.7) и переходим в нее.



```

MINGW64:/d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace
$ ls
PracticeAAS/

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace
$ cs PracticeAAS/
bash: cs: command not found

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace
$ cd PracticeAAS/

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAA
S (master)
$ mkdir "lab1"

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAA
S (master)
$ ls
lab1/  README.md

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAA
S (master)
$
  
```

Рисунок 7-создание папки для выполнения л.р. 1

5. Написать в папке lab1 программу, вычисляющую функцию факториала.

Код программы:

```

int Factorial(int a) {
    return (a == 1) ? 1 : Factorial(a - 1)*a;
}
  
```

6. Создать .gitignore, добавив в игнорируемые файлы – промежуточные файлы компиляции (объектные файлы и другие временные файлы) и исполняемый файл. В результате должны остаться только файл проекта и исходные файлы.

На рис.8 показан созданный файл .gitignore и его содержание, то есть указаны расширения, которые будут игнорироваться и не загружаться.

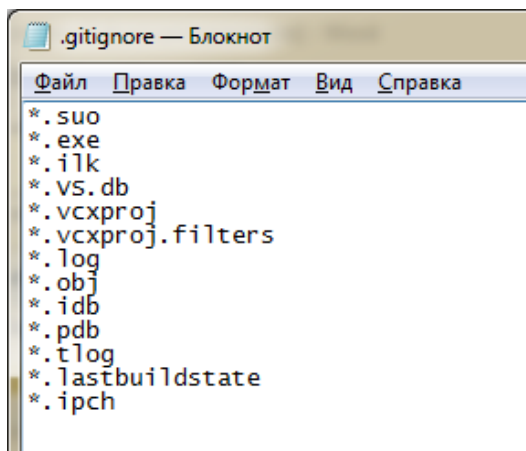


Рисунок 8-содержимое файла .gitignore

7. Сделать по крайней мере два коммита.

Скриншот ниже (рис. 8) показывает процесс создания коммита.

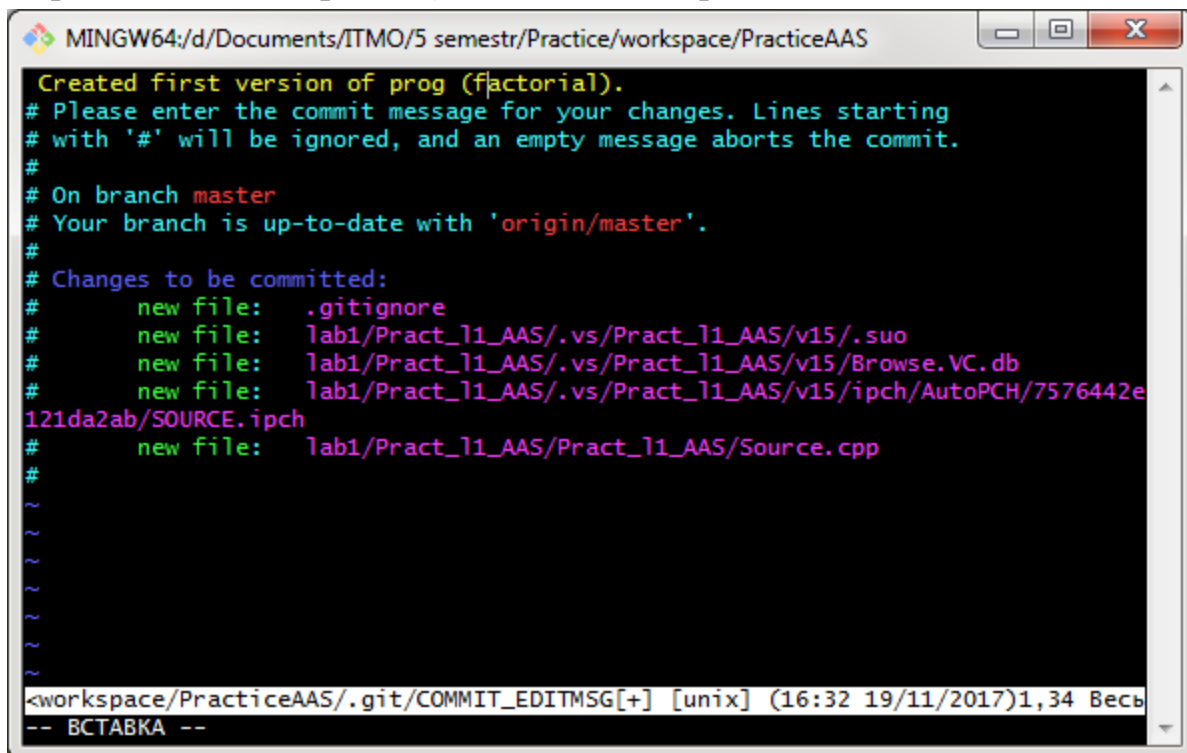


Рисунок 9- написание commit к первоначальному файлу программы

На рис.10 показан результат выполнения команды git commit.


```

new file:   .gitignore
new file:   lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/.suo
new file:   lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/Browse.VC.db
new file:   lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/ipch/AutoPCH/7576442e
121da2ab/SOURCE.ipch
new file:   lab1/Pract_11_AAS/Pract_11_AAS/Source.cpp

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS (master)
$ git commit
[master 38ad1ff] Created first version of prog (factorial).
5 files changed, 15 insertions(+)
create mode 100644 .gitignore
create mode 100644 lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/.suo
create mode 100644 lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/Browse.VC.db
create mode 100644 lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/ipch/AutoPCH/7576442e
121da2ab/SOURCE.ipch
create mode 100644 lab1/Pract_11_AAS/Pract_11_AAS/Source.cpp

Admin@Admin-MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS (master)
$

```

Рисунок 10-результат выполнения команды git commit

Создание второго коммита после изменения файла. Новый код:

```

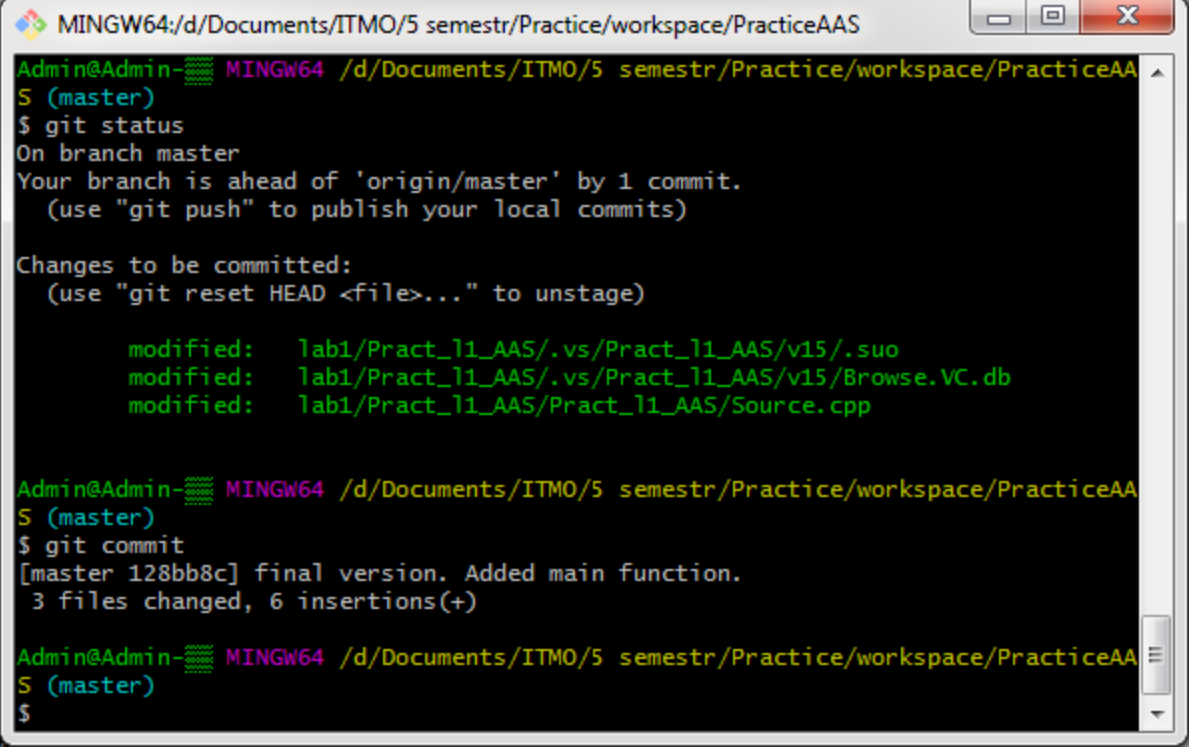
#include <iostream>
int Factorial(int a) {
    return (a == 1) ? 1 : Factorial(a - 1)*a;
}

#Function testing

int main() {
    int x = 12;
    std::cout << Factorial(x) << std::endl;
}

```

Результат выполнения команды `git commit` показан на рис. 11.



```

MINGW64:/d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

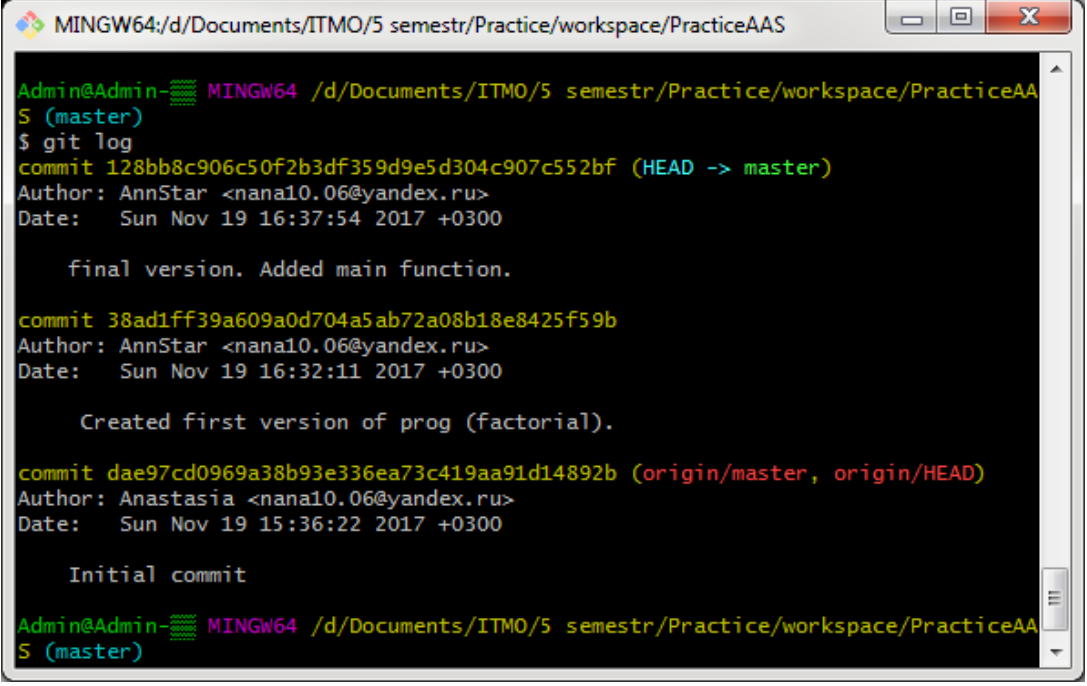
        modified:   lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/.suo
        modified:   lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/Browse.VC.db
        modified:   lab1/Pract_11_AAS/Pract_11_AAS/Source.cpp

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
$ git commit
[master 128bb8c] final version. Added main function.
 3 files changed, 6 insertions(+)

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
$
  
```

Рисунок 11-результат выполнений команды `git commit` после изменения файла

8. Привести результаты работы “`git log`” (рис. 12).



```

MINGW64:/d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
$ git log
commit 128bb8c906c50f2b3df359d9e5d304c907c552bf (HEAD -> master)
Author: AnnStar <nana10.06@yandex.ru>
Date:   Sun Nov 19 16:37:54 2017 +0300

    final version. Added main function.

commit 38ad1ff39a609a0d704a5ab72a08b18e8425f59b
Author: AnnStar <nana10.06@yandex.ru>
Date:   Sun Nov 19 16:32:11 2017 +0300

    Created first version of prog (factorial).

commit dae97cd0969a38b93e336ea73c419aa91d14892b (origin/master, origin/HEAD)
Author: Anastasia <nana10.06@yandex.ru>
Date:   Sun Nov 19 15:36:22 2017 +0300

    Initial commit

Admin@Admin- MINGW64 /d/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
$
  
```

Рисунок 12-результат выполнения команды `git log`

9. Показать при помощи “git diff” изменение любого файла между двумя коммитами. Результат выполнения данной команды показан на рис.13.

```

MINGW64;d:/Documents/ITMO/5 semestr/Practice/workspace/PracticeAAS
$ cd "D:\Documents\ITMO\5 semestr\Practice\workspace\PracticeAAS"

Admin@Admin-MINGW64 /d:/Documents/ITMO/5 semestr/Practice/workspace/PracticeAA
S (master)
$ git diff 38ad1ff39a609a0d704a5ab72a08b18e8425f59b dae97cd0969a38b93e336ea73c419aa91d14892b
diff --git a/.gitignore b/.gitignore
deleted file mode 100644
index a3c2731..0000000
--- a/.gitignore
+++ /dev/null
@@ -1,12 +0,0 @@
-*.sln
-*.exe
-*.ilk
-*.pdb
-*.vcxproj
-*.vcxproj.filters
-*.log
-*.obj
-*.idb
-*.pdb
-*.tlog
-*.lastbuildstate
diff --git a/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/.suo b/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/.suo
deleted file mode 100644
index 62cbb2a..0000000
Binary files a/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/.suo and /dev/null differ
diff --git a/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/Browse.VC.db b/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/Browse.VC.db
deleted file mode 100644
index 61e7175..0000000
Binary files a/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/Browse.VC.db and /dev/null differ
diff --git a/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/ipch/AutoPCH/7576442e121da2ab/SOURCE.ipch b/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/ipch/AutoPCH/7576442e121da2ab/SOURCE.ipch
deleted file mode 100644
index 87ccc54..0000000
Binary files a/lab1/Pract_11_AAS/.vs/Pract_11_AAS/v15/ipch/AutoPCH/7576442e121da2ab/SOURCE.ipch and /dev/null differ
diff --git a/lab1/Pract_11_AAS/Pract_11_AAS/Source.cpp b/lab1/Pract_11_AAS/Pract_11_AAS/Source.cpp
deleted file mode 100644
index 34e7837..0000000
--- a/lab1/Pract_11_AAS/Pract_11_AAS/Source.cpp
+++ /dev/null
@@ -1,3 +0,0 @@
-int Factorial(int a) {
-    return (a == 1) ? 1 : Factorial(a - 1)*a;
-}

```

Рисунок 13-результат выполнения команды git diff

Задание №2. Форматирование и стиль.

Условия задания

Необходимо реализовать задачу №1079 «Максимум».

Ограничение времени: 2.0 секунды

Ограничение памяти: 64 МБ

Рассмотрим последовательность чисел a_i , $i = 0, 1, 2, \dots$, удовлетворяющих следующим условиям:

- $a_0 = 0$
- $a_1 = 1$
- $a_{2i} = a_i$
- $a_{2i+1} = a_i + a_{i+1}$,

где для каждого $i = 1, 2, 3, \dots$.

Цель: написать программу, которая для заданного значения n находит максимальное среди чисел a_0, a_1, \dots, a_n .

Исходные данные: входные данные состоят из нескольких тестов (не более 10). Каждый тест представляет собой строку, в которой записано целое число n ($1 \leq n \leq 99\,999$). В последней строке входных данных записано число 0.

Результат: для каждого n во вводе выведите соответствующее максимальное значение.

Пример

Исходные данные	Результат
5	3
10	4
0	
10000	512
10000	512
0	
0	

Алгоритм решения

В первую очередь, необходимо считать и сохранить все введенные данные. Считываем данные до тех пор, пока введенное значение не равно 0. Если значение равно 0, то считывание завершается. Создаем вектор, в котором будет храниться ряд значений a . В данный вектор помещаем первые два элемента, равные 0 и 1. Считываем введенный элемент и заходим в цикл от 2 до значения элемента с шагом 1. Проверяем на четность значение элемента. Если значение четное, то в вектор ряда складываем следующий элемент, который будет равен элементу этого ряда с индексом, который равен половине текущего итератора в цикле. Если значение нечетно, то значение следующего элемента будет равно сумме элементов, индексы которых равны половине текущего и половине текущего плюс 1 соответственно. После того, как получили ряд для одного введенного числа, ищем максимальный его элемент и выводим. Аналогично поступаем для всех введенных тестовых данных.

Текст программы

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<long long> values;
    vector<long long> test;
    long long num = -1;
    if (num == 0) {
        return 1;
    }
    cin >> num;
    while (num != 0) {
        test.push_back(num);
        cin >> num;
    }
    system("cls");
    for (long long i = 0; i < test.size(); i++) {
        values.push_back(0);
        values.push_back(1);
        for (long long j = 2; j < (test[i] + 1); j++) {
            long long index = 0;
            if (j % 2 == 0) {
                index = j / 2;
                values.push_back(values[index]);
            } else {
                index = (j - 1) / 2;
                values.push_back(values[index] + values[index + 1]);
            }
        }
        long long maximum = 0;
        for (int j = 0; j < values.size(); j++) {
            if (maximum < values[j]) maximum = values[j];
        }
        cout << maximum;
        if (i != (test.size() - 1)) {
            cout << endl;
        }
        values.clear();
    }
    system("pause");
    return 0;
}
```

Результат прохождения тестов на сайте acm.timus.ru представлен на рисунке 14.

Результаты проверки решений

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
7653564	13:09:15 4 дек 2017	AAS	1079_Максимум	Visual C++ 2017	Accepted		0.031	1 740 КБ
	13:05:25							

Рисунок 14- результаты прохождения тестов программой.

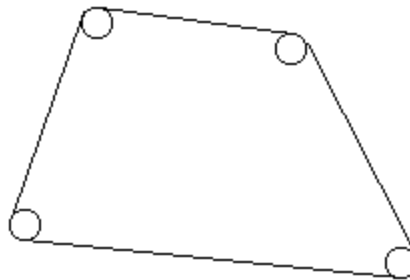
Задание 3. TDD: Разработка через тестирование

Задание, выбранное для реализации и тестирования.

Задача 1020. Ниточка.

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Злоумышленники варварски вбили в ни в чем не повинную плоскую поверхность N гвоздей, да так, что только шляпки остались. Мало того, они в своих подлых целях вбили все гвозди в вершины выпуклого многоугольника. После этого они... страшно сказать... они натянули ниточку вокруг всех гвоздей, так, что поверхности стало совсем больно! Вот как примерно они это сделали:



Ваша задача — определить длину этой ниточки.

Исходные данные

В первой строке входа к этой задаче находятся два числа — количество гвоздей N , $1 \leq N \leq 100$, и вещественное число R — радиус шляпок гвоздей. Все шляпки имеют одинаковый радиус. Далее на входе располагаются еще N строк, в каждой из которых записана через пробел пара вещественных координат центра очередного гвоздя; координаты не превосходят по абсолютной величине числа 100. Описания гвоздей приводятся в порядке обхода вершин многоугольника (либо по часовой стрелке, либо против часовой стрелки), начиная с произвольного. Шляпки разных гвоздей не накладываются друг на друга.

Результат

Выведите вещественное число, округлённое до двух знаков после запятой — длину ниточки, натянутой вокруг всех гвоздей.

Пример

исходные данные	результат
<pre>4 1 0.0 0.0 2.0 0.0 2.0 2.0 0.0 2.0</pre>	14.28

Описание алгоритма работы программы

В первую очередь необходимо считать в переменную количество гвоздей, радиус и дальше необходимо заполнить двумерный массив, в котором буду храниться координаты гвоздей. Разделим задачу на блоки. Для того, чтобы найти длину нитки между двумя центрами двух гвоздей необходимо воспользоваться формулой

$$l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Таким образом попарно мы сможем найти длину всей нитки без учета радиуса гвоздей путем суммирования отдельных расстояний между гвоздей.

Для получения окончательного результата к такой сумме нужно прибавить длину окружности шляпки гвоздя, умножив заданный радиус на 2 и число π .

Ошибочные ситуации

1. Некорректный ввод данных
2. Некорректное количество введенных данных
3. Совпадение координат двух гвоздей

Результаты решения задачи

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
7676359	03:07:45 18 дек 2017	AAS	1020. Ниточка	Visual C++ 2017	Accepted		0.001	284 КБ

Рисунок 15-результат тестирования программы сайт timus

Тестирование

Далее был написан тест, который проверяющий и оценивающий корректность работы программы. Необходимые файл можно найти, пройдя по ссылке:

Результаты тестирования представлены на рисунке 16.

LCOV - code coverage report

Current view: [top level](#) - [c/Temp/tdd/tst_example](#) - [thread.cpp](#) (source / [functions](#))

Test: [coverage.info](#)

Date: 2017-12-18 12:20:59

	Hit	Total	Coverage
Lines:	13	13	100.0 %
Functions:	4	4	100.0 %
Branches:	6	8	75.0 %

Branch data	Line data	Source code
1	:	: #include "thread.h"
2	:	: #include <iostream>
3	:	: #include <math.h>
4	:	:
5	:	110 : double LenPoints(double aX1, double aX2, double aY1, double aY2) {
6	:	110 : return sqrt((aX1 - aX2) * (aX1 - aX2) + (aY1 - aY2) * (aY1 - aY2));
7	:	: }
8	:	:
9	:	6 : double LenThread(int aCount, double aRad, double (*pCoord)[2]) {
10	[+ +]:	6 : if(aCount <= 0) {
11	:	2 : return -2;
12	:	: }
13	:	4 : const double PI = 3.14159;
14	:	4 : double sum = 0;
15	:	//
16	[+ +]:	108 : for(int i = 0; i < (aCount - 1); ++i)
17	:	104 : sum += LenPoints(pCoord[i][0], pCoord[i + 1][0], pCoord[i][1], pCoord[i + 1][1]);
18	:	//
19	:	4 : sum += LenPoints(pCoord[0][0], pCoord[aCount - 1][0], pCoord[0][1], pCoord[aCount - 1][1]) + 2 * PI * aRad;
20	:	4 : sum = trunc(sum * 100.0) / 100.0;
21	:	4 : return sum;
22	[+ - -] [+ - -]:	4 : }

Generated by: [LCOV version 1.13-13-gad399be](#)

Рисунок 16- результаты тестирования.

Текст программы

Файл thread.cpp

```
#include "thread.h"
#include <iostream>
#include <math.h>

double LenPoints(double aX1, double aX2, double aY1, double aY2) {
    return sqrt((aX1 - aX2) * (aX1 - aX2) + (aY1 - aY2) * (aY1 - aY2));
}

double LenThread(int aCount, double aRad, double (*pCoord)[2]) {
    if(aCount <= 0) {
        return -2;
    }
    const double PI = 3.14159;
    double sum = 0;
    //
    for(int i = 0; i < (aCount - 1); ++i)
        sum += LenPoints(pCoord[i][0], pCoord[i + 1][0], pCoord[i][1], pCoord[i + 1][1]);
    //
    sum += LenPoints(pCoord[0][0], pCoord[aCount - 1][0], pCoord[0][1], pCoord[aCount - 1][1]) + 2 * PI * aRad;
    sum = trunc(sum * 100.0) / 100.0;
    return sum;
}
```

Файл main.cpp

```
#include "thread.h"

#define MY_DEF_USE_LIBTAP
#ifdef MY_DEF_USE_LIBTAP

#define TAP_COMPILE
#include "libtap/cpp_tap.h"

using namespace std;
```

```

int main(int, char *[]) {
    //
    plan_tests(8);
    double arr[100][2];
    //
    arr[0][0] = 0.0;
    arr[0][1] = 0.0;
    arr[1][0] = 3.0;
    arr[1][1] = 0.0;
    ok(LenPoints(arr[0][0], arr[1][0], arr[0][1], arr[1][1]) == 3.0, "TestLenPoints1");
    //
    arr[0][0] = 10.0;
    arr[0][1] = 0.0;
    arr[1][0] = -10.0;
    arr[1][1] = 0.0;
    ok(LenPoints(arr[0][0], arr[1][0], arr[0][1], arr[1][1]) == 20.0, "TestLenPoints2");
    //
    arr[0][0] = 0.0;
    arr[0][1] = 0.0;
    arr[1][0] = 2.0;
    arr[1][1] = 0.0;
    arr[2][0] = 2.0;
    arr[2][1] = 2.0;
    arr[3][0] = 0.0;
    arr[3][1] = 2.0;
    ok(LenThread(4, 1, arr) == 14.28, "Test from timus");
    //
    arr[0][0] = 0.0;
    arr[0][1] = 0.0;
    arr[1][0] = 3.0;
    arr[1][1] = 0.0;
    arr[2][0] = 3.0;
    arr[2][1] = 3.0;
    ok(LenThread(3, 2, arr) == 22.80, "Test 1");
    //
    ok(LenThread(0, 2, arr) == -2, "Test 2");
    //
    ok(LenThread(-10, 3, arr) == -2, "Test 3");
    //
    arr[0][0] = 0.0;
    arr[0][1] = 0.0;
    ok(LenThread(1, 3, arr) == 18.84, "Test 4");
    //
    for(int i(0); i < 99; i++) {
        arr[i][0] = i;
        arr[i][1] = i;
    }
    arr[99][0] = 50.0;
    arr[99][1] = 99.0;
    ok(LenThread(100, 1, arr) == 303.79, "Test 5");
    //
    return exit_status();
    //
    return 0;
}

#endif

```

Проект на тему «Спектры. Сравнение»

Цель работы

Обнаружить и выделить образцы со спектральными линиями на изображении. Предположить и реализовать критерий схожести, основанный на совпадении линий спектра.

Описание выполненной работы

Распознавание и выделение образцов

В ходе изучения представленной базы образцов (рис. 17) было выведено два общих подхода к распознаванию «трубочек» (образцов, содержащих спектры). Главная их идея заключается в том, как анализируется и считывается изображение – вертикально или горизонтально. Таким образом было реализовано два алгоритма распознавания трубочек.

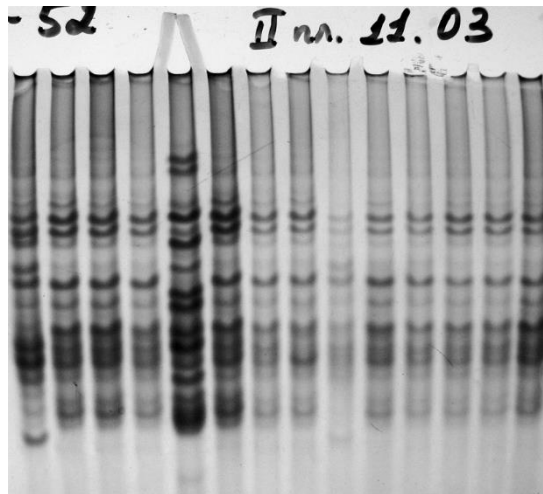


Рисунок 17- образец исходного изображения.

Первый алгоритм заключается в том, что вычисляется середина изображения по вертикали. Далее идет анализ ряда горизонтальных пикселей на данной высоте. Если пиксель удовлетворяет выведенному условию контрастности, то он считается «трубочкой», иначе фоном. После этого происходит анализ полученных данных, а именно, вычисляется ширина каждой выделенной предполагаемой «трубочки», далее вычисляется оптимальная ширина образца. На основании этого вычисления производится коррекция результатов, и те элементы, которые не проходят по условию оптимальности по ширине, удаляются. Аналогичным способом находится высота каждого образца: по центральной линии «трубки» алгоритм считывает верхнюю часть изображения от центра и определяет самый яркий перепад цвета, далее анализирует нижнюю часть изображения аналогично. Результат выделения продемонстрирован на рисунке 18. Таким образом было получено, что алгоритм выделяет корректно 70% трубочек.

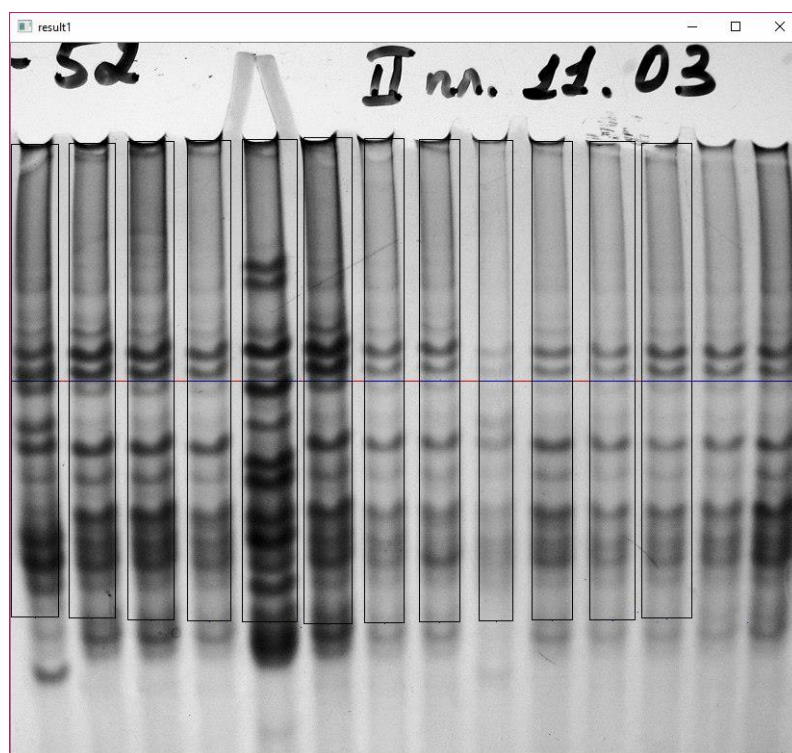


Рисунок 18- полученное изображение с выделенными спектрами после применения алгоритма 1.

Важно отметить, что образцы имеют тоже фон и он не является однородным и равномерным (наблюдаются шумы, засветы, тени). По этой причине существует вероятность, что линия предполагаемой середины, может попасть на строку пикселей, где эти искажения повлияют на результат и границы будут определены не верно. Именно поэтому алгоритм, описанный выше имеет результат не 100%. Для увеличения процента точности выделения был разработан второй способ выделения образцов.

Главное отличие нового алгоритма-анализ пикселей не только в одной средней строки, а блока строк. Т.е. также вычисляется середина изображения, но дальше считывается и анализируется не одна строка, а около 100 строк. Высота «трубочек» вычисляется также на основании блока столбцов пикселей. Таким образом был получен стопроцентный результат распознавания образцов. При этом были распознаны не только хорошо контрастные образцы, но и так же те, которые имеют сильные искажения. Результат применения данного алгоритма представлен на рисунке 19.

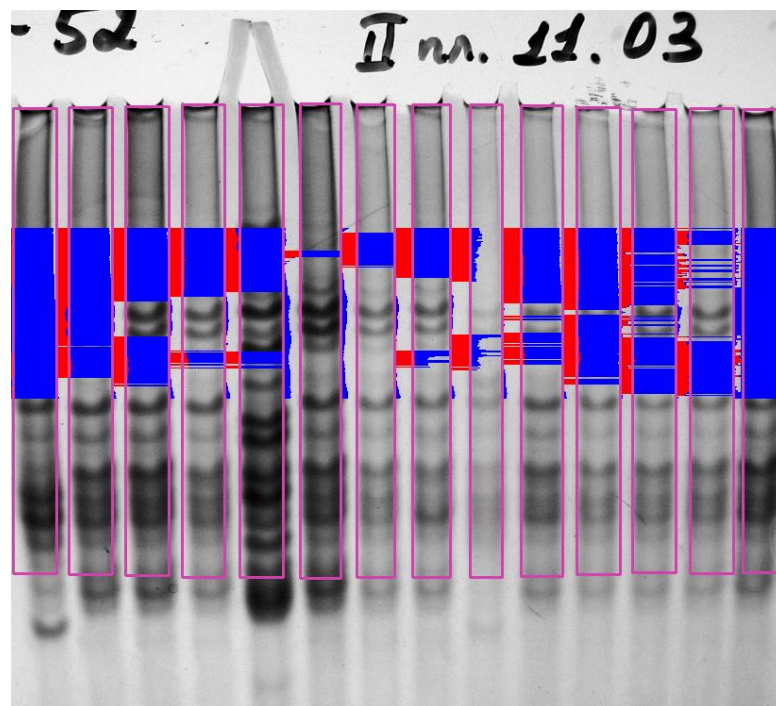


Рисунок 19- изображение, на котором показано выделение образцов, произведенное с помощью алгоритма 2.

Распознавание спектральных линий образцов

После того, как все образцы были выявлены и выделены, необходимо распознать спектральные линии. Как было описано ранее, фон трубочек неоднородный и не имеет одного цвета, такое же замечание можно сделать и по поводу спектральных линий. Величина яркости и контрастности каждой из них могут отличаться на порядок. На основании этого ряда причин были выделены 4 алгоритма.

Общей идеей всех этих алгоритмов является деление каждого образца на блоки по всей высоте и вычисление фона трубочки в каждом блоке отдельно. Такой метод уменьшает процент ошибки, так как фон некоторых образцов в разных частях «трубочки» различается в 2 раза. На дальнейших этапах алгоритмы имеют различия.

Алгоритм 1.

В данном методе анализируется три столбца пикселей- центральный относительно ширины «трубочки», смещенный на некоторую дельта вправо и влево. Таким образом предполагается, что, если центр образца имеет сильный засвет, то смещенные линии распознают спектральную линию. Далее происходит обработка результатов, полученных предполагаемых спектральных линий. Этот метод дает общее представление о расположении спектральных линий, но нельзя однозначно определить их ширину и распознать отдельные линии в некоторых случаях.

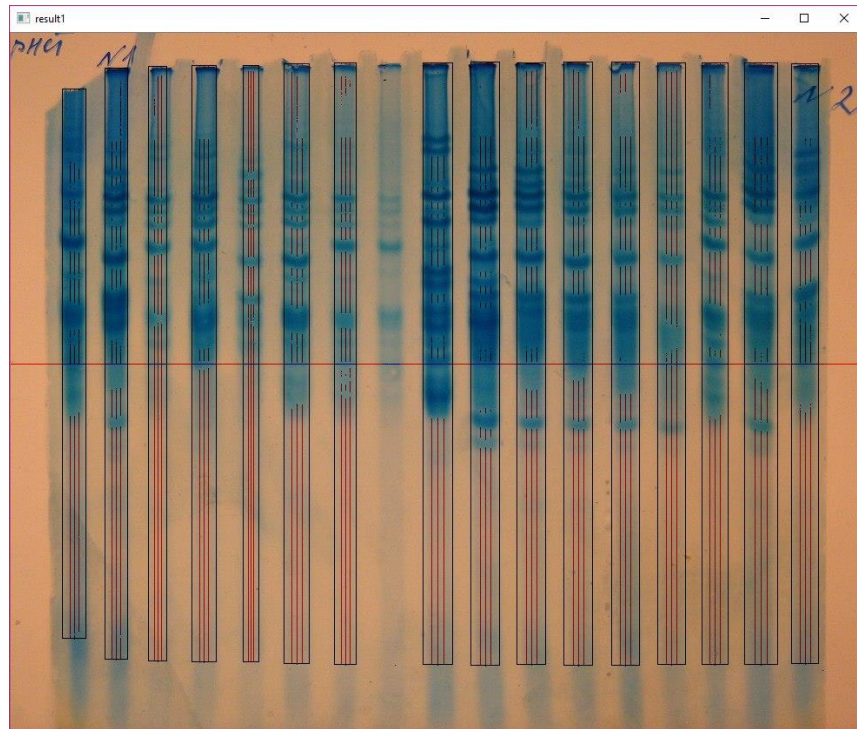


Рисунок 20-результат выделения спектральных линий методом 1.

Алгоритм 2.

Этот способ имеет сильно сходство со вторым методом выделения образцов. Горизонтальный анализ. По всей высоте «трубочки» считываются строки и проверяется условие: если продолжительность контрастной линии превышает половину ширины трубочки, тогда это спектральная линия, иначе фон. Результат представлен на рис.21. По тем же причинам, что и прошлый метод, были получены искаженные результаты.

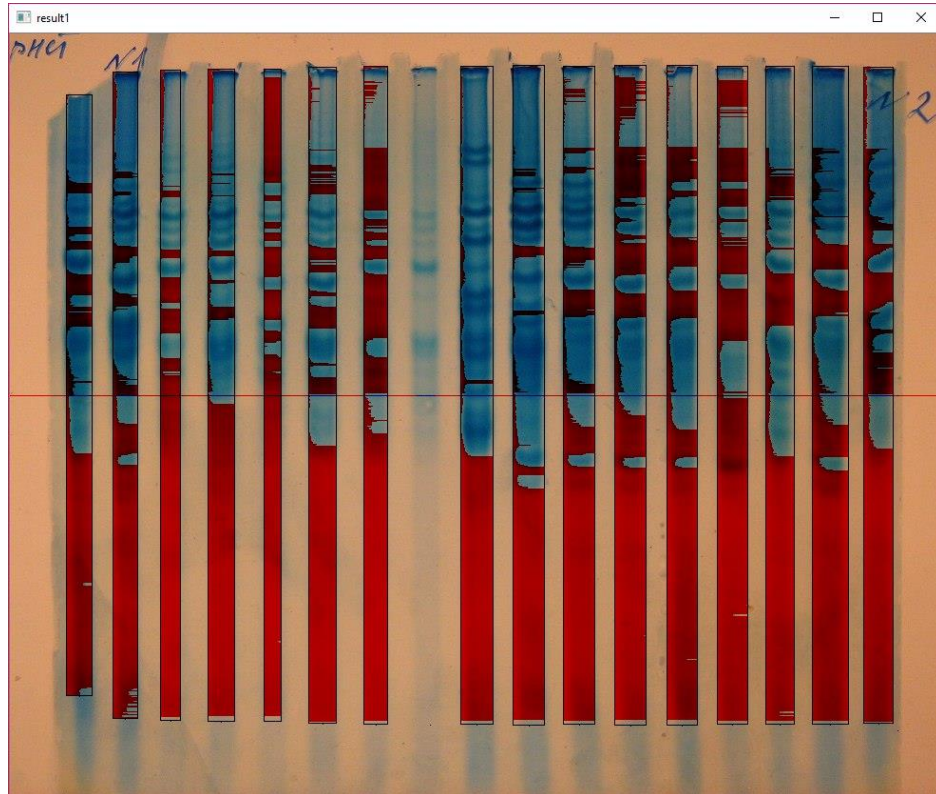


Рисунок 21- результат выделения спектральных линий методом 2.

Алгоритм 3.

Данный способ проводит анализ одного среднего столбца пикселей. На первый взгляд этот алгоритм ничем не отличается от первого, но их главное и весомое различие заключается в условиях распознавания спектральной линии и фона, а именно, что значение пикселя фона заменяется на значение предшествующего. Также критерии отбора сделаны более строгими и подобраны такие параметры, что с помощью этого метода можно определить больше половины спектральных линий. Но также этого недостаточно для определения схожих спектров.

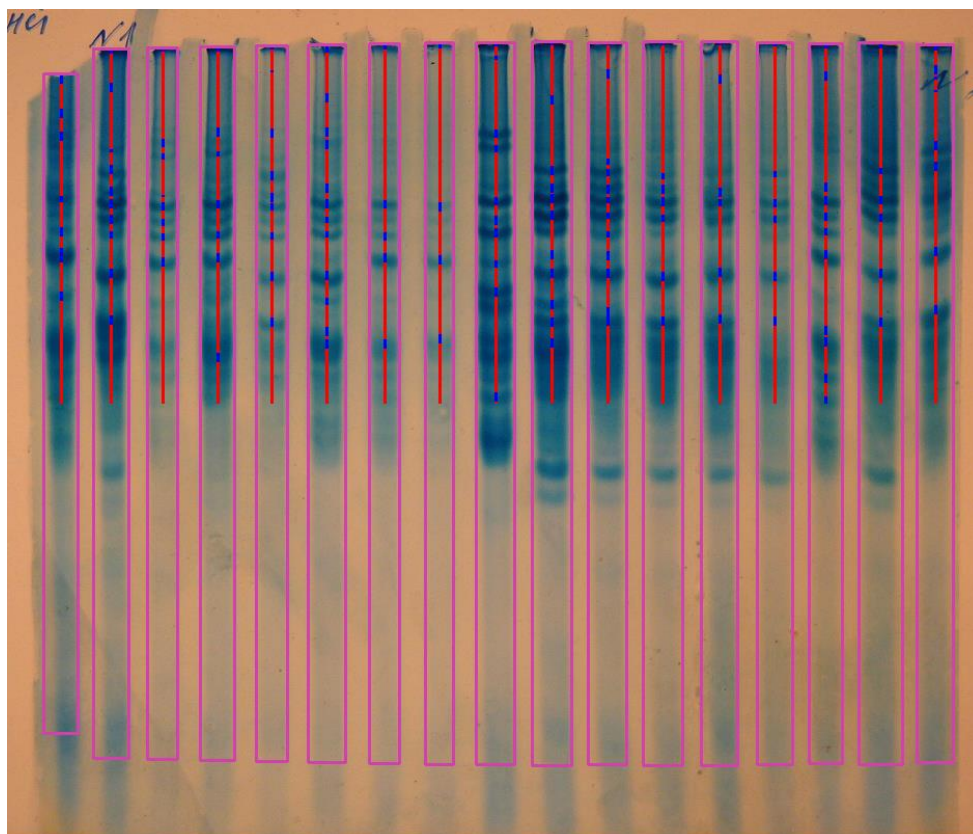


Рисунок 22- результат выделения спектральных линий методом 3.

Алгоритм 4.

Это алгоритм отличается больше всего от всех ранее предложенных. Его основная идея в первоначальной обработке изображения. А именно, выделяются основные контуры и получаем новое изображение, на котором изображены только контуры темных линий. Но этого оказалось также недостаточно, так как обнаружить метод для того, чтобы определять граничное значение фона и распознаваемых спектральных линий не удалось. Таким образом при различных настройках получается два исхода: потеря спектральных линий, но хорошая «чистка» фона, либо спектральные линии не удалены, но их не распознать, так как фон распознается как контур. При всех его недостатках и преимуществах был выбран именно он для детектирования спектральных линий и сравнения спектров.



Рисунок 23-результат выделения спектральных линий методом 4.

Критерии сравнения

Критерии для определения образца и спектральных линий различаются. Для выделения спектра по горизонтали главным условием является, что от фона образец должен отличаться больше, чем на 80%. Так как контрастность контуров в верхней части изображения и его центре различны, то и процент отличия был подвержен корректировке для определения высоты спектра. В этом случае значение процента составляет 90%.

После обработки полученных спектров было получено изображение (рис. 23), которое необходимо обработать для окончательного выделения спектральных линий. Контуром спектральной линии считается такой горизонтальный ряд в образце, где процентное отношение черных пикселей к ширине всего образца превышает 60%. Таким образом будет получено изображение 24. На основании его производится сравнение каждого образца с остальными, если процент одинаковых значений пикселей $> 90\%$, тогда спектры считаются схожими.

Алгоритм обработки спектральных линий

После того, как было получено изображение 23, была произведена отчистка остального фона. После, используя критерий различия фона и контуров было получено изображение (рис. 24).

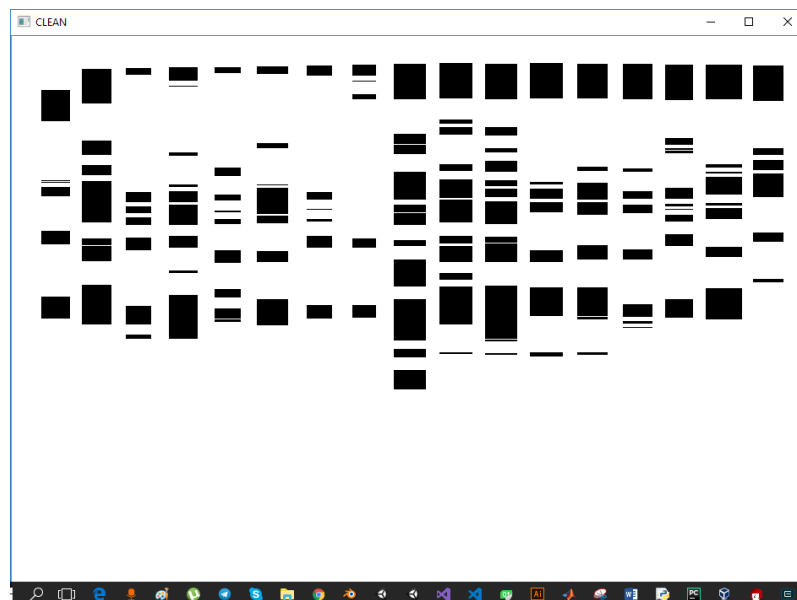
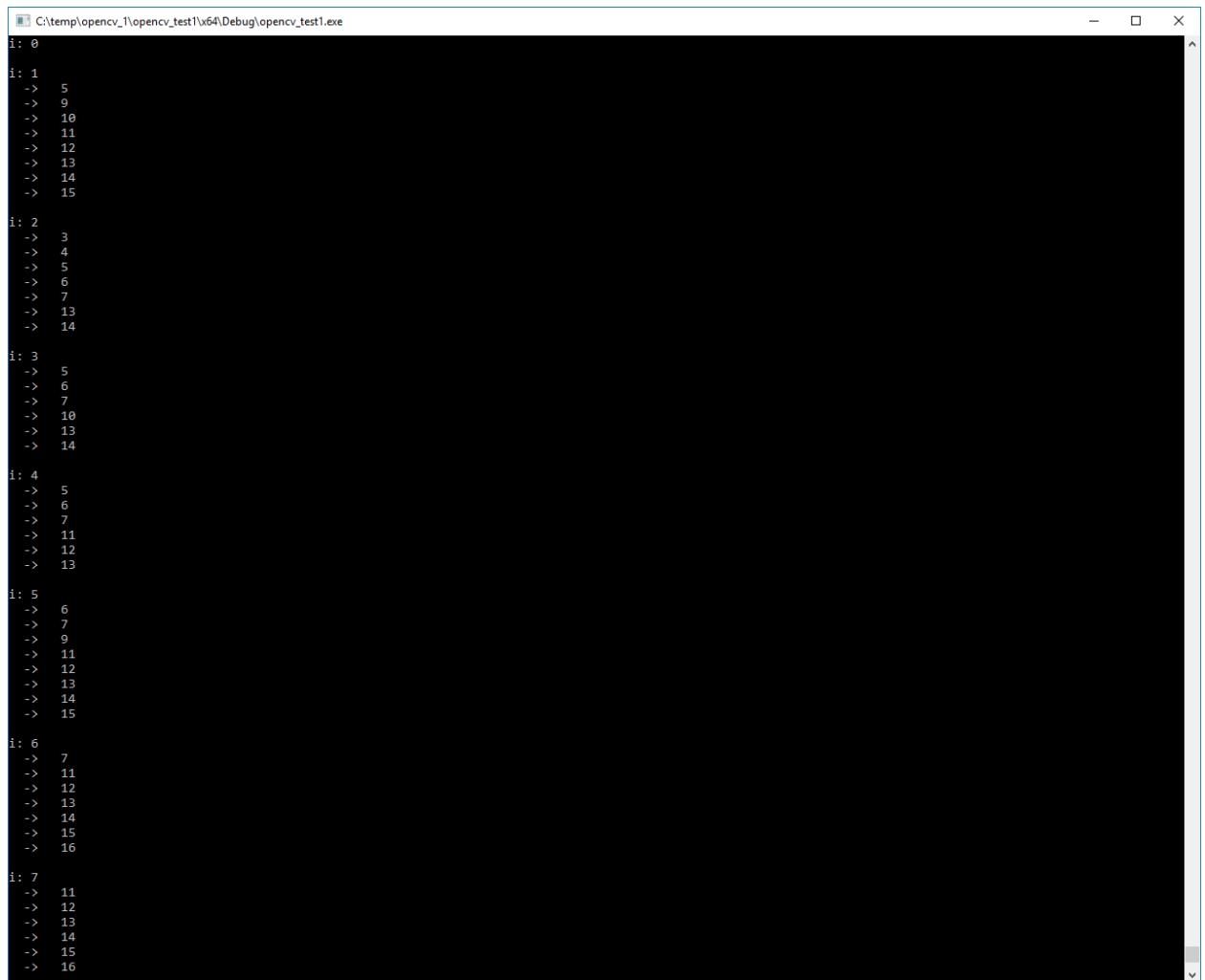


Рисунок 24- выделенные контуры спектральных линий

После этого проверяется каждый образец на основании спектральных линий на схожесть с другими. Полученные результаты выводятся. Пример результата работы программы на рис. 25. Исходный код программы и все файлы расположены в репозитории GitHub(<https://github.com/AnnGareeva/PracticeAAS>).



```
C:\temp\opencv_1\opencv_test\vs64\Debug\opencv_test1.exe
i: 0
i: 1
-> 5
-> 9
-> 10
-> 11
-> 12
-> 13
-> 14
-> 15
i: 2
-> 3
-> 4
-> 5
-> 6
-> 7
-> 13
-> 14
i: 3
-> 5
-> 6
-> 7
-> 10
-> 13
-> 14
i: 4
-> 5
-> 6
-> 7
-> 11
-> 12
-> 13
i: 5
-> 6
-> 7
-> 9
-> 11
-> 12
-> 13
-> 14
-> 15
i: 6
-> 7
-> 11
-> 12
-> 13
-> 14
-> 15
-> 16
i: 7
-> 11
-> 12
-> 13
-> 14
-> 15
-> 16
```

Рисунок 25- пример результата работы программы.

Вывод по практике

В курсе практики были освоены навыки использования C++ и изучены приемы разработки программного обеспечения. Были выполнены три лабораторные работы и выполнен проект. По результатам выполнения проекта, были изучены основы работы с OpenCV. С помощью нее был проведен анализ изображения, выделение контуров и сравнение полученных данных.

**Приложение А. Слайды презентации с семинара по C++11
на тему: “User-defined literals”.**

Презентация и доклад расположены по ссылке:
github.com/AnnGareeva/PracticeAAS/tree/master/Presentation