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 Прикладной оптики

Отчет по практике

Выполнил: студент группы B3316
Рафиков А. И.
Проверил: Кудрявцев А. С.

Санкт-Петербург
2019

Оглавление

ЦЕЛЬ ПРОВЕДЕНИЯ ПРАКТИКИ

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

Ссылка на репозиторий студента в системе контроля версий GitHub: https://github.com/Mazzeich/Practice

Лабораторная работа 1. Распределенная система контроля версий Git

1. Задачи
    1. Установить Git-Bash
    2. Потренироваться, пройдя курс https://githowto.com/ru
    3. Завести аккаунт на https://github.com/
    4. Создать репозиторий для лабораторных работ
    5. Склонировать репозиторий из https://github.com/ на свой компьютер
    6. Создать папку lab1
    7. Написать в папке lab1 программу, вычисляющую функцию факториала
    8. 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.
    9. Сделать по крайней мере два коммита. Отправить зафиксированные изменения на https://github.com/
    10. Привести результаты работы «git log»
    11. Показать при помощи «git diff <ваш_файл_ваши_коммиты>» изменение любого файла между двумя коммитами
2. Текст программы

```cpp
#include <iostream>

int factorial(int j)
{
    if (j == 0) return 1;
    else return j * factorial(j - 1);
}

int main()
{
    int i;

    std::cin >> i;
    std::cout << factorial(i) << "\n";
    return 0;
}

```

3. Ход работы

1. Создание репозитория для практики

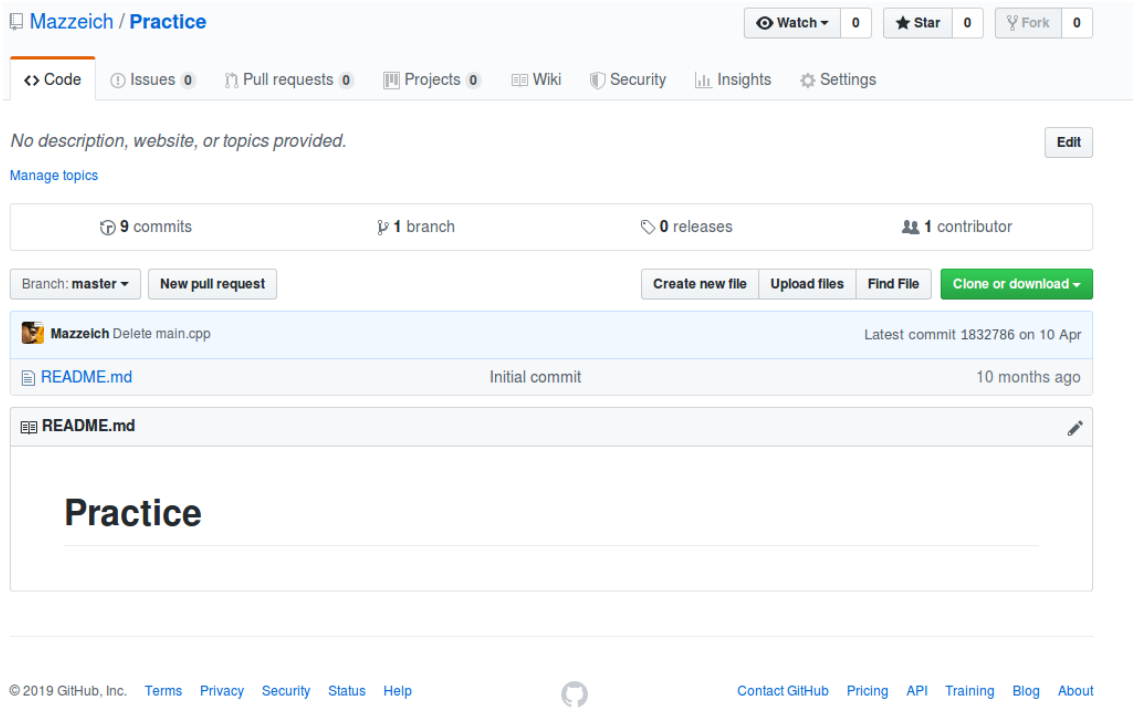

Рис. 1 Созданный репозиторий для заданий по практике

2. Клонирование репозитория из https://github.com/ на компьютер

```
oem@Melissa ~ $ git clone https://github.com/Mazzeich/Practice
Клонирование в «Practice»…
remote: Enumerating objects: 21, done.
remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 21
Распаковка объектов: 100% (21/21), готово.
Проверка соединения… готово.
oem@Melissa ~ $ ls
bin             Downloads  proggear.ru      Telegram   Изображения
CLionProjects   intel      PycharmProjects  Видео      Музыка
CSharp_projects OS_labs    Qt projects      Документы  Общедоступные
Desktop         Practice   Scripts          Загрузки   Шаблоны
oem@Melissa ~ $ cd Practice/
oem@Melissa ~/Practice $
```

Рис. 2 Резльтат работы утилиты git clone

3. Создание .gitignore-файла

```
oem@Melissa ~/Practice $ git add .gitignore
oem@Melissa ~/Practice $ git status
На ветке master
Ваша ветка обновлена в соответствии с «origin/master».
Изменения, которые будут включены в коммит:
  (используйте «git reset HEAD <файл>…», чтобы убрать из индекса)

        новый файл:    .gitignore
```

Рис. 3 Добавление .gitignore-файла в репозиторий

4. Сделать как минимум два коммита

```
oem@Melissa ~/Practice $ git commit -m "First commit"
[master 1d47b7d] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore
oem@Melissa ~/Practice $
```

Рис. 4 Первый коммит

```
oem@Melissa ~/Practice $ git status
На ветке master
Ваша ветка обновлена в соответствии с «origin/master».
Изменения, которые будут включены в коммит:
  (используйте «git reset HEAD <файл>…», чтобы убрать из индекса)

        новый файл:    fact/.gitignore
        новый файл:    fact/fact.pro
        новый файл:    fact/main.cpp

Изменения, которые не в индексе для коммита:
  (используйте «git add/rm <файл>…», чтобы добавить или удалить файл из индекса)
  (используйте «git checkout -- <файл>…», чтобы отменить изменения
   в рабочем каталоге)

        изменено:      .gitignore
        удалено:       README.md

oem@Melissa ~/Practice $ git commit -m "Second commit"
[master 80e2cd8] Second commit
 3 files changed, 103 insertions(+)
 create mode 100644 fact/.gitignore
 create mode 100644 fact/fact.pro
 create mode 100644 fact/main.cpp
oem@Melissa ~/Practice $
```

Рис. 5 Второй коммит

```
oem@Melissa ~/Practice/fact $ git status
На ветке master
Ваша ветка обновлена в соответствии с «origin/master».
Изменения, которые будут включены в коммит:
  (используйте «git reset HEAD <файл>…», чтобы убрать из индекса)

        переименовано: main.cpp -> main1.cpp

Изменения, которые не в индексе для коммита:
  (используйте «git add/rm <файл>…», чтобы добавить или удалить файл из индекса)
  (используйте «git checkout -- <файл>…», чтобы отменить изменения
   в рабочем каталоге)

        изменено:      ../.gitignore
        удалено:       ../README.md
        изменено:      fact.pro
        изменено:      main1.cpp

oem@Melissa ~/Practice/fact $ git commit -m "Third commit"
[master dca32ea] Third commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename fact/{main.cpp => main1.cpp} (100%)
oem@Melissa ~/Practice/fact $ git push
Username for 'https://github.com': mazzeich
Password for 'https://mazzeich@github.com':
Подсчет объектов: 3, готово.
Delta compression using up to 4 threads.
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 303 bytes | 0 bytes/s, готово.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Mazzeich/Practice
   80e2cd8..dca32ea  master -> master
oem@Melissa ~/Practice/fact $
```

Рис. 6 Третий коммит

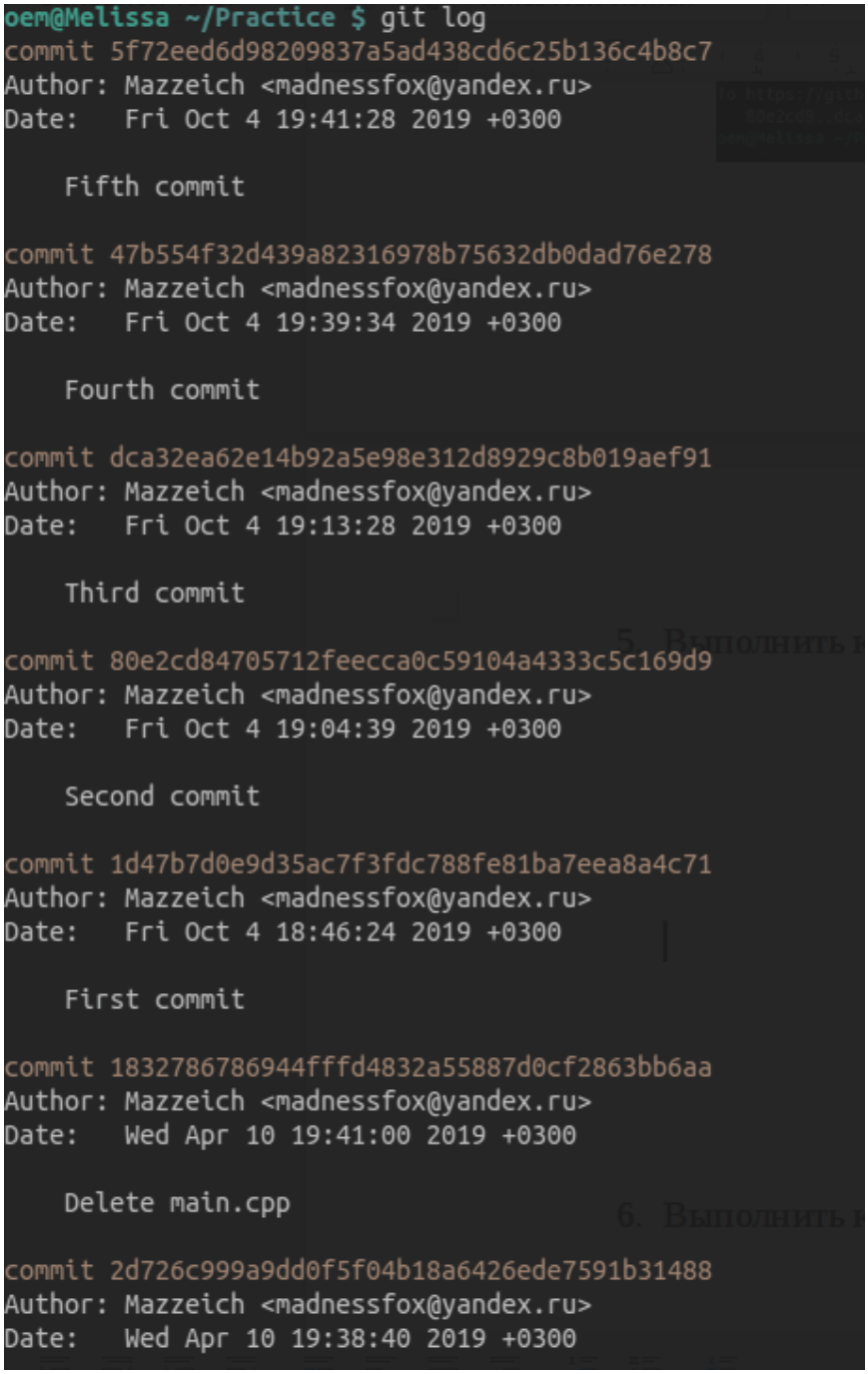

Рис. 7 Результат работы утилиты git log

5. Выполнить команду git diff

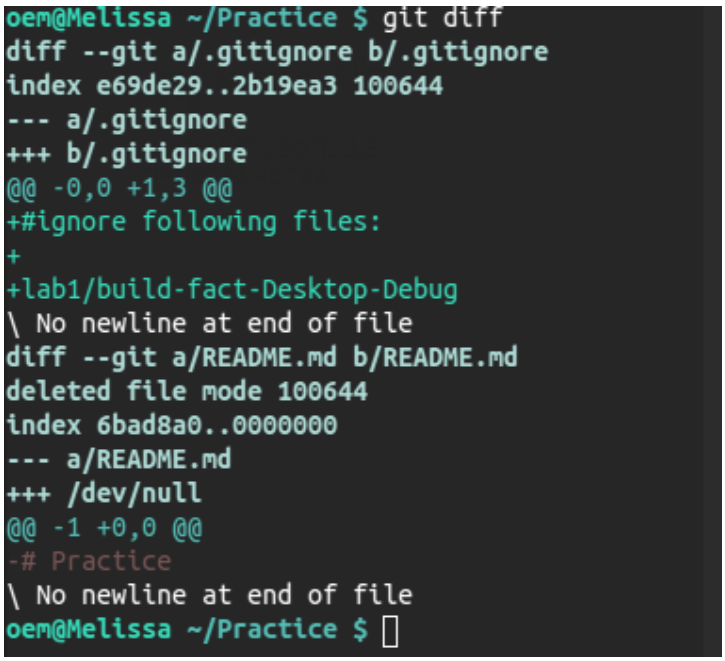

Рис. 8 Результат работы утилиты git diff

Лабораторная работа 2. Форматирование и стиль

Задачи

1. Разработать программу, оформленную в соответствии с правилами в файле CodeRules.pdf. Разработка осуществляется на языке C++
2. Сделать скриншот с именем и успешным результатом проверки
3. Поместить результаты работы — программу (файлы с расширениями cpp, h, sln, vxcproj, filters), а также скриншот с сайта timus в свой репозиторий

1803. Винтовки белочехов

Ограничение времени: 3.0 секунды
Ограничение памяти: 64 МБ

Белочехи решили прекратить ожесточённые бои в Сибири и вернуться домой. Но покинуть Россию было непросто — чтобы добраться на кораблях из Владивостока в Европу, были нужны деньги. Белочехи решили раздобыть необходимую сумму, продав свои винтовки наступающим войскам красных. Всего на продажу они выставили n винтовок. Первые две винтовки были самыми обычными, и белочехи просили за каждую из них только один рубль. Зато i-я винтовка ($i \geq 3$) стоила столько же, сколько ($i-1$)-я и ($i-2$)-я вместе взятые.

В стране имеют хождение только банкноты с номиналами, равными степеням числа k (банкноты в один рубль, k рублей, k^2 рублей и т. д.) В распоряжении большевиков уже достаточно типографий, поэтому напечатать нужное количество денег не составило труда. Они заплатили за каждую винтовку минимально возможное количество банкнот, в сумме дающее ровно столько, сколько просили за эту винтовку белочехи.

Когда винтовки попали в руки красных, Чапаев попросил Анку упорядочить их по возрастанию количества банкнот, которое было за них заплачено. Если за две винтовки было заплачено одинаковое количество банкнот, то раньше должна идти винтовка с меньшим номером. Помогите Анке выполнить просьбу Чапаева.

Исходные данные

В единственной строке через пробел записаны целые числа k и n ($2 \leq k \leq 10$; $3 \leq n \leq 50000$).

Результат

Выведите перестановку чисел от 1 до n — номера винтовок, упорядоченные по возрастанию количества уплаченных за них банкнот.

Пример

исходные данные	результат
10 8	1 2 3 4 8 7 5 6

Замечания

Когда Анка выполнит просьбу Чапаева, стоимости винтовок в рублях будут выглядеть так: {1, 1, 2, 3, 21, 13, 5, 8}.

Рис. 9 Условие задания 1803 с timus

4. Алгоритм:
   Представить каждую цену в виде массива номиналов, и при суммировании двух цен складываем вместе сумму стоимостей каждого номинала, заменяя на более высокие номиналы, когда это возможно. Оптимизируем алгоритм, комбинируя [1, p ^ 1, p ^ 2, p ^ 3, p ^ 4, ..., p ^ x] для некоторого x в новый «супербазис», который заменяет наш предыдущий p. Предварительно вычисляем «мапу», которая переводит, сколько реальных стоимостей соответствует каждому количеству стоимостей в этом «супербазисе».

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
8567405	22:03:52 10 окт 2019	Mazzeich	1803. Винтовки белочехов	Visual C++ 2017	Accepted		0.468	904 КБ

Рис. 10 Результат проверки работы алгоритма на timus

Исходный код программы

```
#include <algorithm>

std::pair<int, int> answer[50000] = { { 1, 1 }, { 1, 2 } };

int array_1[35000];
int array_2[35000];
int pre[100001];
int Ap[20];

int l;
int number_rifle;
int k_rubles;
int current_rubles;

int add(int* a, int* b)
{
    int ret = 0;

    for(int i = 0; i < l; i++)
    {
        b[i] += a[i];

        if(b[i] >= current_rubles)
        {
            b[i] -= current_rubles;
            b[i + 1]++;
            l = std::max(i + 2, l);
        }
        ret += pre[b[i]];
    }

    return ret;
}

void calcpre()
{
    Ap[0] = pre[0] = 0;
    for(int i = 1; i < current_rubles; i++)
    {
        int j = 0;
        pre[i] = pre[i-1];
        Ap[j]++;

        while(true)
        {
            pre[i]++;
            if(Ap[j] >= k_rubles)
            {
                Ap[j] -= k_rubles;
                Ap[j+1]++;
                pre[i] -= k_rubles;
                j++;
            }
            else
                break;
```

```
            }
        }
    }

    int main()
    {
        scanf("%d %d", &k_rubles, &number_rifle);

        array_1[0] = 1;
        array_2[0] = 1;

        l = 1;

        int* a = &array_1[0];
        int* b = &array_2[0];

        current_rubles = k_rubles;

        while(current_rubles * k_rubles < 100000)
            current_rubles *= k_rubles;

        calcpre();

        for(int i = 2; i <= number_rifle; i++)
        {
            answer[i].first  = add(a, b);
            answer[i].second = i + 1;

            std::swap(a, b);
        }

        std::sort(answer, answer + number_rifle);

        for(int i = 0; i < number_rifle; i++)
            printf("%d ", answer[i].second);
    }
```

Лабораторная работа 3. TDD: разработка через тестирование

2095. Скрам

Ограничение времени: 1.0 секунды

Ограничение памяти: 64 МБ

Зима в Екатеринбурге — самое длинное время года. И каждый коротает долгие зимние вечера по-своему. Программист Иван любит работать. Настолько, что может просидеть весь день до глубокой ночи в своём уголке в офисе, не отвлекаясь от написания кода на всякие посторонние дела. Жаль только, начальство не сильно его в этом поддерживает, всё время заставляя его участвовать во всяких совещаниях и презентациях.

С первого января нового года по всей организации вводится новый порядок отчётов сотрудников перед руководителями. Теперь каждый второй день каждый программист обязан отчитываться перед своим непосредственным начальником. Из оставшихся дней каждый третий — перед начальником своего начальника. Из оставшихся каждый четвёртый — перед начальником начальника своего начальника. И так далее. Организация очень крупная, потому можно считать, что для простых программистов в ней бесконечно много уровней начальства. И хотя вся эта бюрократия Ивана совсем не радует, деваться некуда — нужно учиться с ней жить. В частности, нужно спланировать отпуск так, чтобы на него выпало как можно больше отчётных дней и как можно меньше дней, в которые можно спокойно поработать. Иван хочет посчитать, сколько дней без отчётов выпадет на период запланированного им отпуска, чтобы всё взвесить и, возможно, перенести отпуск на другое время.

Исходные данные

В первой строке даны целые числа l и r — номера первого и последнего дней запланированного Иваном отпуска, считая со дня введения нового порядка ($1 \le l \le r \le 10^9$).

Результат

Выведите единственное число — количество дней в период запланированного отпуска, когда Ивану не нужно будет ни перед кем отчитываться.

Пример

исходные данные	результат
1 10	3

Замечания

В дни с номерами 2, 4, 6, 8 и 10 Иван будет отчитываться перед своим непосредственным начальником. В 5-й день — перед начальником своего начальника, в 9-й — перед начальником начальника своего начальника. Так что спокойными остаются только дни 1, 3 и 7.

Для решения задачи будем использовать алгоритм повторного деления.

main.cpp

```
#include "MyClass.h"
#include <gtest/gtest.h>


int main(int argc, char* argv[])
{
	testing::InitGoogleTest(&argc, argv);
	return RUN_ALL_TESTS();
}
```

MyClass.h

```
#ifndef H_MYCLASS
#define H_MYCLASS

int MyClass(long long a, long long b);

bool checkFirstDay(long long a);
```

```
bool checkSecondDay(long long a);
bool firstDayDouble(double a, long long b);
bool secondDayDouble(long long a, double b);
bool moreThanTwoArgs(long long a, long long b, ...);
bool inputNegative(long long a, long long b);
bool secondDayIsGreater(long long a, long long b);
bool noArguments();

#endif
```

MyClass.cpp

```
#include "MyClass.h"
#include <exception>

int MyClass(long long a, long long b)
{
	auto f = [](int n)
	{
		for (int i = 2; n >= i; n -= n / i++);
		return n;
	};

	return (f(b) - f(a - 1));
}

bool checkFirstDay(long long a)
{
	if (a < 1 || a > 1000000000)
		return false;

	return true;
}

bool checkSecondDay(long long a)
{
	if (a < 1 || a > 1000000000)
		return false;

	return true;
}

bool firstDayDouble(double a, long long b)
{
	return false;
}

bool secondDayDouble(long long a, double b)
{
	return false;
}

bool moreThanTwoArgs(long long a, long long b, ...)
{
	return false;
}

bool inputNegative(long long a, long long b)
{
	if(a < 0 || b < 0)
		return false;

	return true;
}
```

```
bool secondDayIsGreater(long long a, long long b)
{
	if(b > a)
		return false;

	return true;
}

bool noArguments()
{
	return false;
}
```

test.cpp

```
#include "pch.h"
#include "X:\Coding\C++\timus_2095\timus_2095\MyClass.cpp"
#include "gtest/gtest.h"
#include <string>

TEST(VacationTest, CorrectAnswer)
{
  EXPECT_EQ(MyClass(1, 10), 3);
  EXPECT_EQ(MyClass(2, 10), 2);
}

TEST(VacationTest, UpperBound)
{
	EXPECT_TRUE(checkSecondDay(999999999));
	EXPECT_TRUE(checkFirstDay(999999999));
}

TEST(VacationTest, LowerBound)
{
	EXPECT_TRUE(checkFirstDay(1));
	EXPECT_TRUE(checkSecondDay(1));
}

TEST(VacationTest, UnderflowValue)
{
	EXPECT_FALSE(checkFirstDay(0));
	EXPECT_FALSE(checkSecondDay(0));
}

TEST(VacationTest, OverflowValue)
{
	EXPECT_FALSE(checkFirstDay(1000000001));
	EXPECT_FALSE(checkSecondDay(1000000001));
}

TEST(VacationTest, InvalidDataType)
{
	EXPECT_FALSE(firstDayDouble(5.5, 200));
	EXPECT_FALSE(secondDayDouble(4, 100.3));
}

TEST(VacationTest, ZeroArguments)
{
	EXPECT_FALSE(noArguments());
}
```

```
TEST(VacationTest, BelowZero)
{
	EXPECT_FALSE(inputNegative(-5, 10));
	EXPECT_FALSE(inputNegative(2, -6));
}

TEST(VacationTest, SecondDayIsGreaterOrEqual)
{
	EXPECT_TRUE(2, 10);
	EXPECT_TRUE(2, 2);
	EXPECT_TRUE(10, 2);
}

TEST(VacationTest, NoInputData)
{
	EXPECT_FALSE(noArguments());
}
```

Результат прохождения тестов:

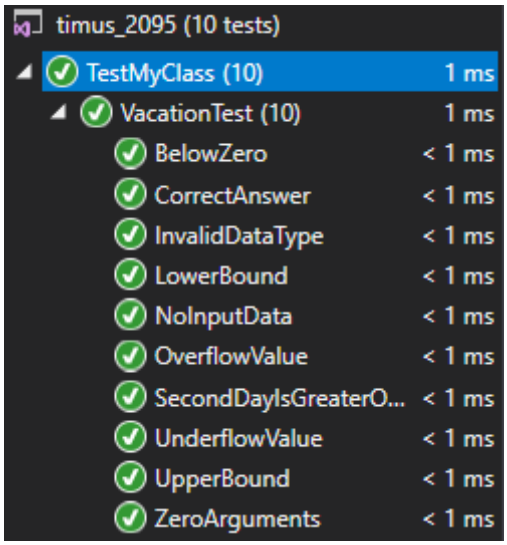

Рис. 11 Прохождение google-тестов программы

Последние попытки

Автор: Mazzeich • Задача: Скрам

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
8599043	00:05:15 28 окт 2019	Mazzeich	2095. Скрам	Visual C++ 2017	Accepted		0.015	304 КБ

Рис. 12 Результат проверки на timus

ОСНОВНОЕ ЗАДАНИЕ

Подсчитать количество зерен на фотографии, используя язык C++ и алгоритмы из готовых библиотек OpenCV.

Ссылка на репозиторий: https://github.com/Mazzeich/Practice

В работе будет использован алгоритм распознавания образов Watershed

Этим алгоритмом обрабатываются 500 фотографий с изображением зерен риса.

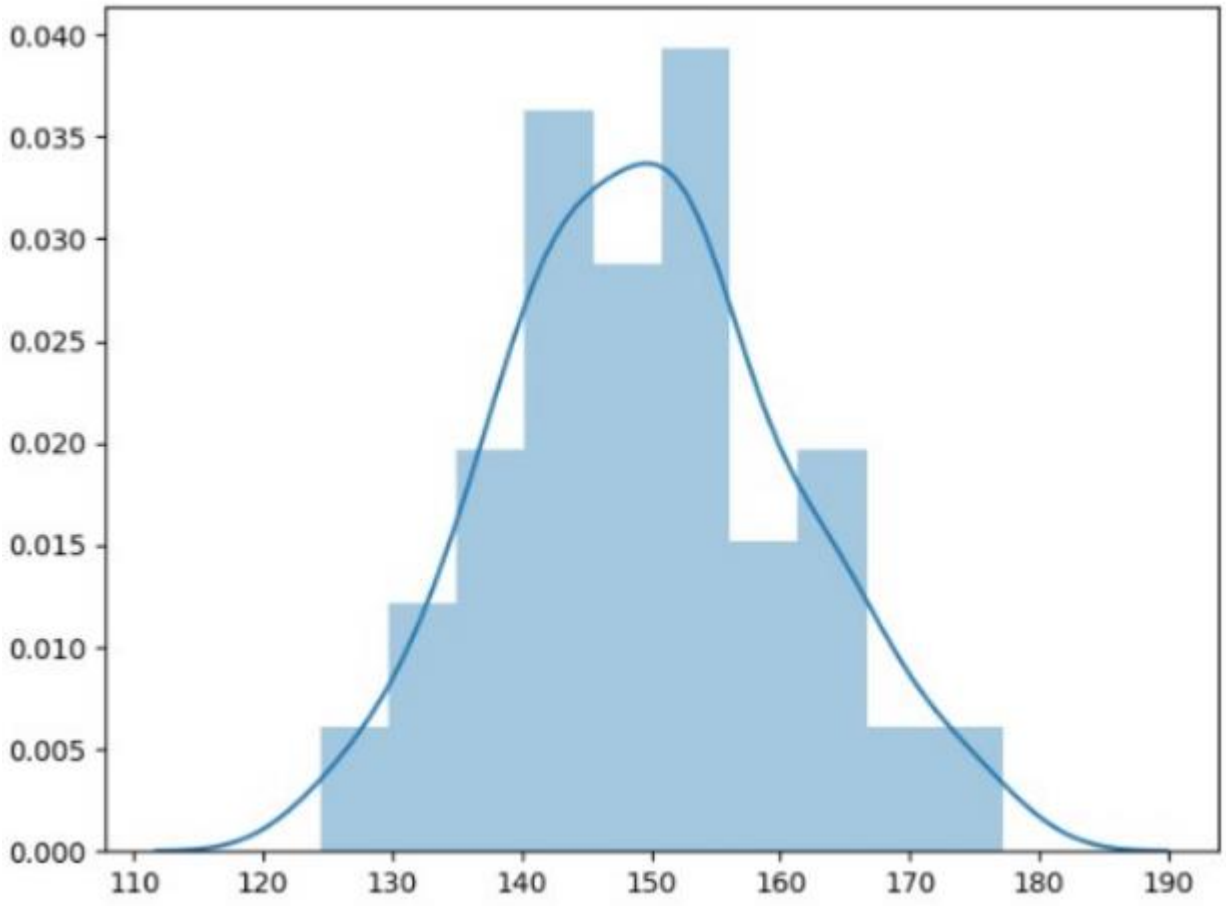

Рис. 13 График погрешности подсчета при работе алгоритма на 170 зернах

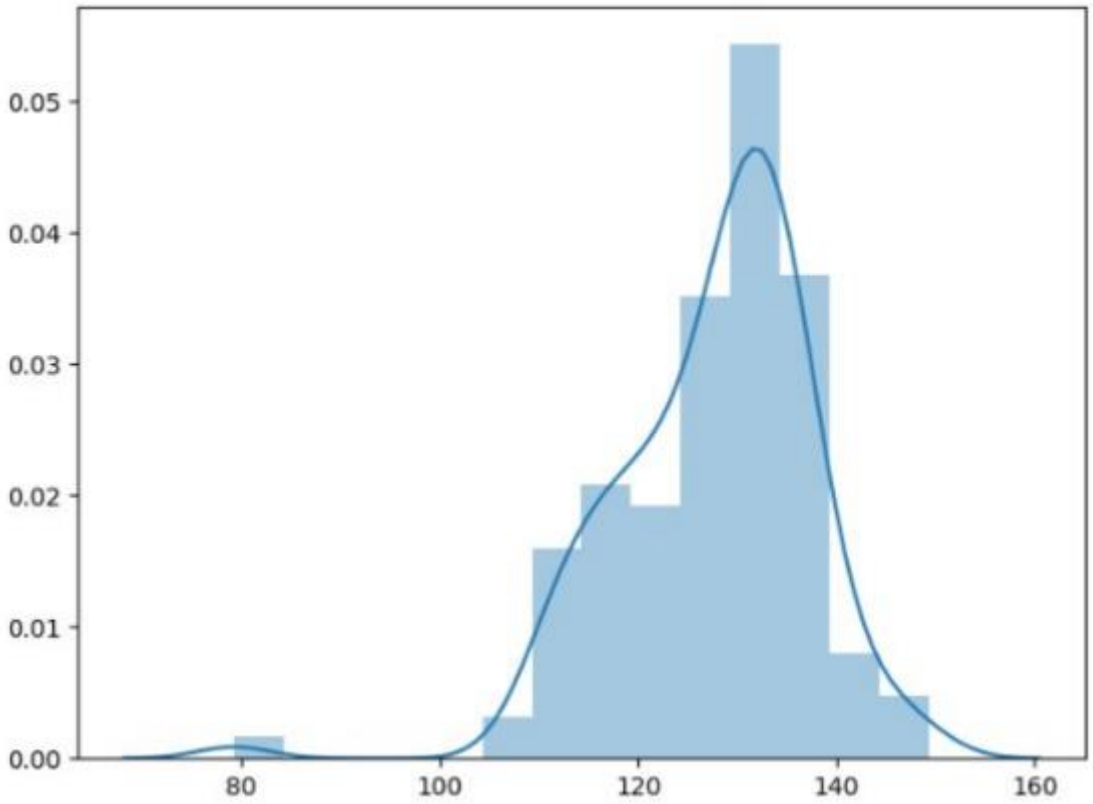

Рис. 14 График погрешности подсчета при работе алгоритма на 150 зернах

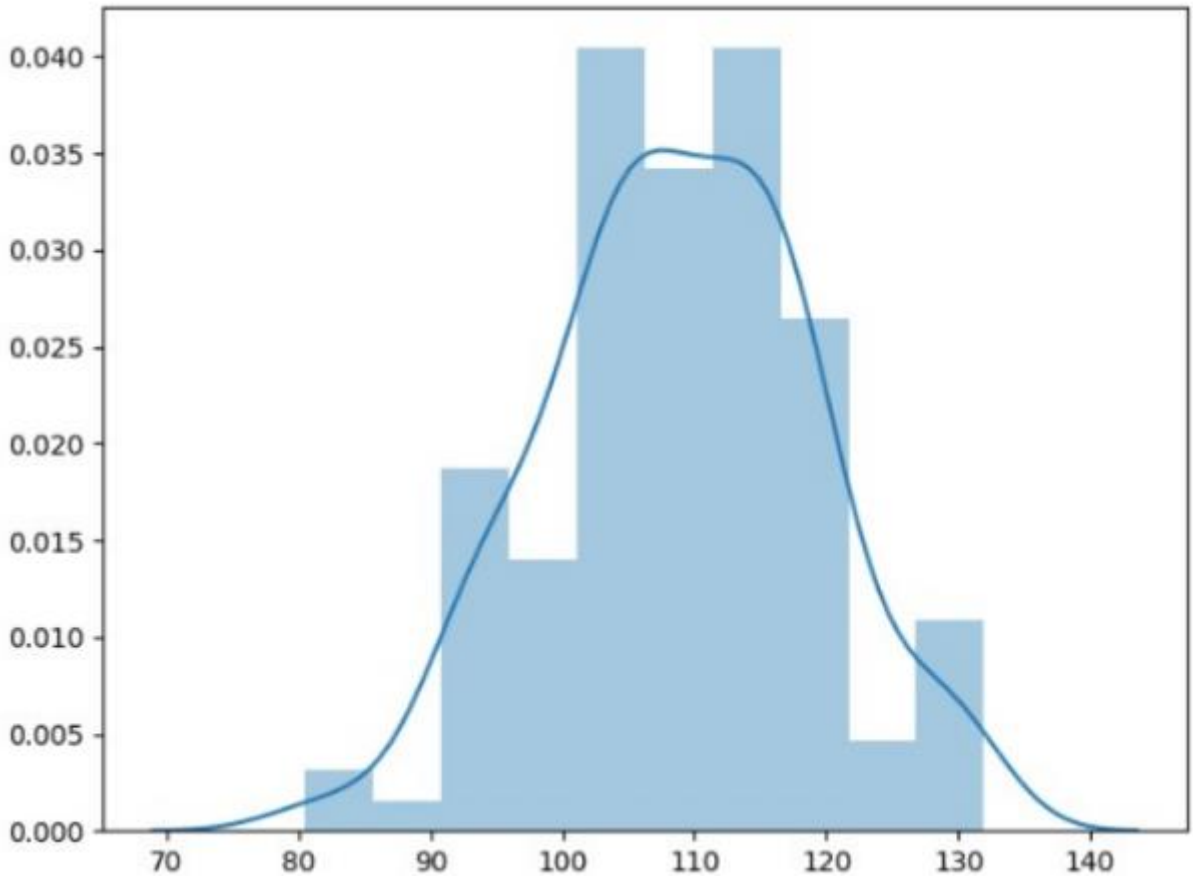

Рис. 15 График погрешности подсчета при работе алгоритма на 130 зернах

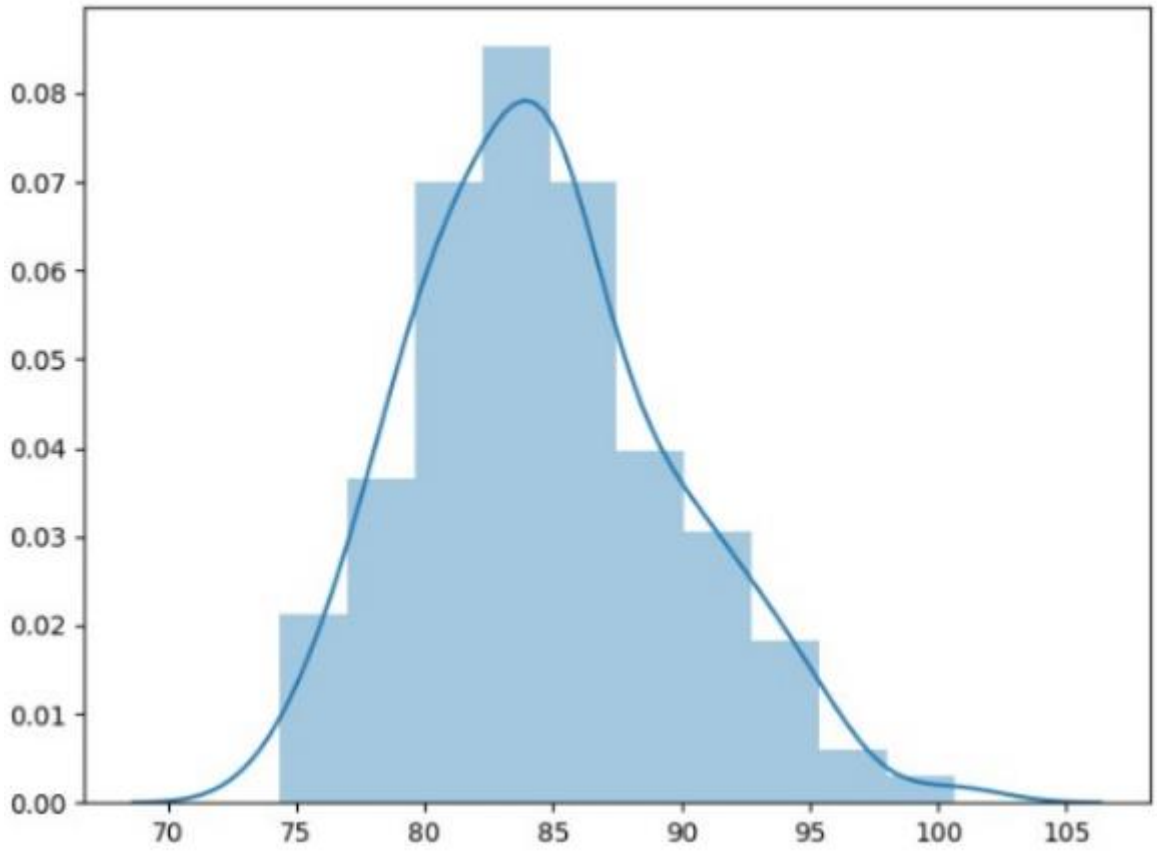

Рис. 16 График погрешности подсчета при работе алгоритма на 100 зернах

Таблица 1

Количество зерен	**Среднее**	**Стандартное отклонение**	**Количество элементов**
170	141.2	12.3	125
150	128.3	11.1	125
130	113.3	7.5	125
100	85.0	4.2	125

ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp

```
#include <gtest/gtest.h>
#include "computerVision.hpp"


using namespace cv;
using namespace std;
```

```
int main(int argc, char *argv[])
{
  testing::InitGoogleTest(&argc, argv);
  int code = RUN_ALL_TESTS();
  std::string dirName = "/home/oem/Practice/Rice/";
  std::vector<std::string> files= getFileList(dirName);

  ofstream myfile;
  myfile.open ("res.csv");
  for (int i = 0; i < files.size(); ++i)
  {
    vector<double> length;
    vector<double> width;
    vector<double> area;
    Mat image = imread(string(dirName+files.at(i)));
  }

  myfile.close();

  return 0;
}
```

computerVision.hpp

```
#include <opencv2/opencv.hpp>
#include "dirent.h"

using namespace cv;
```

```
using namespace std;

Mat imageRead(std::string path);
Mat laplacian(Mat &sharp);

Mat erode(Mat &subPlans, Mat &imgLaplacian);

Mat getMarkers(Mat subPlans, Mat &sureFigure, Mat &unknown) ;
vector<vector<Point> > drawContours(Mat markers, Mat sureFigure);

int mainAlg(Mat resultImage, std::vector<double> &length,
        std::vector<double> &width, std::vector<double> &area);


std::vector<std::string> getFileList(std::string path);
```

computerVision.cpp

```
#include "computerVision.hpp"
using namespace cv;
using namespace std;


const double PIXEL_SIZE = 0.102;

Mat imageRead(std::string path)
{
  Mat img = imread(path.c_str());
  if(!img.data)
```

```
  {
    throw std::invalid_argument("This file doesn't exist");
  }
  return imread(path.c_str());
}

Mat laplacian(Mat &sharp)
{
  Mat kernel = (Mat_<float>(3, 3) << 1, 1, 1, 1, -8, 1, 1, 1, 1);
  Mat imgLaplacian;
  filter2D(sharp, imgLaplacian, CV_32F, kernel);
  return imgLaplacian;
}


Mat erode(Mat &subPlans, Mat &imgLaplacian)
{
  subPlans.convertTo(subPlans, CV_8UC3);
  imgLaplacian.convertTo(imgLaplacian, CV_8UC3);


  blur(subPlans, subPlans, Size(3, 3));


  int erodeSz = 4;
  Mat structuringElement = cv::getStructuringElement(MORPH_ELLIPSE,
                                                     Size(2 * erodeSz + 1,
                                                          2 * erodeSz + 1),
```

```
                                    Point(erodeSz, erodeSz));

  erode(subPlans, subPlans, structuringElement);
  dilate(subPlans, subPlans, structuringElement);
  bitwise_not(subPlans, subPlans);


  threshold(subPlans, subPlans, 40, 255,
          CV_THRESH_BINARY | CV_THRESH_OTSU);

  return subPlans;
}

Mat getMarkers(Mat subPlans, Mat &sureFigure, Mat &unknown)
{
  Mat opening;
  Mat kernel_1(3, 3, CV_8U, Scalar(1));
  morphologyEx(subPlans, opening, MORPH_OPEN, kernel_1,
            Point(-1, -1), 1);

  Mat sure_bg;
  dilate(opening, sure_bg, kernel_1, Point(-1, -1), 3);

  Mat distTransform;
  distanceTransform(opening, distTransform, CV_DIST_L2, 5);

  double minVal, maxVal;
  Point minLoc, maxLoc;
```

```
  minMaxLoc(distTransform, &minVal, &maxVal, &minLoc, &maxLoc);

  threshold(distTransform, sureFigure, 0, 255, 0);

  Mat sureFigure1;
  sureFigure.convertTo(sureFigure1, CV_8UC1);
  subtract(sure_bg, sureFigure1, unknown);

  sureFigure.convertTo(sureFigure, CV_32SC1, 1.0);

  return Mat::zeros(sureFigure.rows, sureFigure.cols, CV_32SC1);
}


vector<vector<Point> > drawContours(Mat markers, Mat sureFigure)
{
  vector<vector<Point> > contours;
  vector<Vec4i> hierarchy;
  findContours(sureFigure, contours, hierarchy, RETR_CCOMP,
               CHAIN_APPROX_SIMPLE);

  int compCount = 0;

  for (int index = 0; index >= 0; index = hierarchy[index][0], compCount++)
    drawContours(markers, contours, index, Scalar::all(compCount + 1),
                 -1, 8, hierarchy, INT_MAX);
  return contours;
}
```

```
int mainAlg(Mat resultImage, std::vector<double> &length,
        std::vector<double> &width, std::vector<double> &area)
{

  Mat hsv;
  cvtColor(resultImage, hsv, CV_BGR2HSV);

  vector<Mat> channels;
  split(hsv, channels);


  Mat sharp = channels[1];
  Mat imgLaplacian = laplacian(sharp);
  imgLaplacian = laplacian(imgLaplacian);

  channels[1].convertTo(sharp, CV_32F);

  Mat subPlans = sharp - imgLaplacian;
  subPlans = erode(subPlans, imgLaplacian);


  Mat sureFigure, contour;

  Mat markers = getMarkers(subPlans, sureFigure, contour);

  vector<vector<Point> > contours = drawContours(markers, sureFigure);
```

```
markers = markers + 1;
for (int i = 0; i < markers.rows; i++)
{
  for (int j = 0; j < markers.cols; j++)
  {
    unsigned char &v = contour.at<unsigned char>(i, j);
    if (v == 255)
    {
      markers.at<int>(i, j) = 0;
    }
  }
}

watershed(resultImage, markers);

for (int i = 0; i < markers.rows; i++)
{
  for (int j = 0; j < markers.cols; j++)
  {
    int index = markers.at<int>(i, j);
    if (index == -1)
      resultImage.at<Vec3b>(i, j) = Vec3b(0, 255, 0);

  }
}

int count = 0;
```

```
  for (unsigned int i = 0; i < contours.size(); i = i + 2)
  {
    RotatedRect rect = minAreaRect(contours[i]);
    putText(resultImage, to_string(count + 1), rect.center,
            FONT_ITALIC, 2, Scalar(0, 0, 255), 1);

    length.push_back(rect.size.height * PIXEL_SIZE);
    width.push_back(rect.size.width * PIXEL_SIZE);
    area.push_back(contourArea(contours[i]) * PIXEL_SIZE * PIXEL_SIZE);
    count++;
  }

  return count;
}



std::vector<std::string> getFileList(std::string path)
{
  DIR *dir;
  struct dirent *ent;
  std::vector<std::string> files;
  if ((dir = opendir(path.c_str())) != NULL)
  {
    while ((ent = readdir(dir)) != NULL)
    {
      if (ent->d_name!= std::string("..") and ent->d_name!= std::string("."))
      {
```

```
      files.push_back(ent->d_name);
    }
   }
   closedir(dir);
  } else {
   throw std::invalid_argument("This directory doesn't exist");
  }
  return files;
}
```

test.cpp

```
#include <gtest/gtest.h>
#include "computerVision.hpp"

const double EPS = 5.0;

TEST(structTest, imageRead)
{
  EXPECT_THROW(imageRead("EFKMD3"), std::invalid_argument);
}

TEST(structTest, imageRead2)
{
  EXPECT_NO_FATAL_FAILURE(
        imageRead("/home/oem/Practice/Rice/1233.JPG"));
}

TEST(structTest, imageRead3)
```

```
{
  EXPECT_THROW(
        imageRead("/home/oem/Practice/Rice/1233.jpg"),
              std::invalid_argument);
}

TEST(structTest, imageRead4)
{
  EXPECT_THROW(
        imageRead("/home/oem/Practice/Rice/"), std::exception);
}

TEST(laplacianTest, test1)
{
  Mat kernel = (Mat_<float>(5, 5) <<
                          0, 0, 0, 0, 0,
        0, 255, 255, 255, 0,
        0, 255, 255, 255, 0,
        0, 255, 255, 255, 0,
        0, 0, 0, 0, 0);
  Mat image = laplacian(kernel);

  EXPECT_GT (image.at<double>(3, 3) + EPS, 0.0);

}

TEST(laplacianTest, test2)
{
```

```
  Mat kernel = (Mat_<float>(5, 5) <<
                          255, 255, 255, 255, 255,
        255, 0, 0, 0, 255,
        255, 0, 0, 0, 255,
        255, 0, 0, 0, 255,
        255, 255, 255, 255, 255);

  Mat image = laplacian(kernel);

  EXPECT_GT (0.0 + EPS, image.at<double>(3, 3));
}

TEST(erodeTest, test1)
{
  Mat kernel = (Mat_<float>(5, 5) <<
        255, 255, 255, 255, 255,
        255, 0, 0, 0, 255,
        255, 0, 100, 0, 255,
        255, 0, 0, 0, 255,
        255, 255, 255, 255, 255);
  Mat lap = laplacian(kernel);
  Mat image = erode(kernel, lap);

  EXPECT_GT (0.0 + EPS, image.at<double>(3, 3));
}

TEST(erodeTest, test2)
{
```

```
  Mat kernel = (Mat_<float>(5, 5) <<
        255, 255, 255, 255, 255,
        255, 255, 255, 255, 255,
        255, 255, 255, 255, 255,
        255, 255, 255, 255, 255,
        255, 255, 255, 255, 255);
  Mat lap = laplacian(kernel);
  Mat image = erode(kernel, lap);



  EXPECT_GT (0.0 + EPS, image.at<double>(3, 3));
}

TEST(erodeTest, test3)
{
  Mat kernel = (Mat_<float>(5, 5) <<
        0, 0, 0, 0, 0,
        0, 0, 255, 0, 0,
        0, 255, 255, 255, 0,
        0, 0, 255, 0, 0,
        0, 0, 0, 0, 0);
  Mat lap = laplacian(kernel);
  Mat image = erode(kernel, lap);



  EXPECT_GT (0.0 + EPS, image.at<double>(3, 3));
}
```

```
TEST(erodeTest, test4)
{
  Mat
  kernel = (Mat_<float>(6, 6) <<
        255, 255, 255, 255, 255, 255,
        255, 0, 0, 0, 255, 255,
        255, 0, 0, 0, 255, 255,
        255, 0, 0, 0, 255, 255,
        255, 255, 255, 255, 255, 255,
        255, 255, 255, 255, 255, 255);
  Mat lap = laplacian(kernel);
  Mat image = erode(kernel, lap);

  EXPECT_GT (0.0 + EPS, image.at<double>(3, 3));
}

TEST(markerTest, centerTest1)
{
  Mat img = imread("rice.png");
  Mat hsv;
  cvtColor(img, hsv, CV_BGR2HSV);

  vector<Mat> channels;
  split(hsv, channels);
  Mat sharp = channels[2];

  Mat imgLaplacian = laplacian(sharp);
```

```
  imgLaplacian = laplacian(imgLaplacian);



  channels[2].convertTo(sharp, CV_32F);

  Mat subPlans = sharp - imgLaplacian;
  subPlans = erode(subPlans, imgLaplacian);



  Mat sureFigure, unknown;
  Mat markers = getMarkers(subPlans, sureFigure, unknown);

  EXPECT_EQ(0, unknown.at<double>(30, 60));
  EXPECT_GT(0, unknown.at<double>(25, 5)-EPS);
}

TEST(markerTest, centerTest2)
{
  Mat img = imread("rice.png");
  Mat hsv;
  cvtColor(img, hsv, CV_BGR2HSV);

  vector<Mat> channels;
  split(hsv, channels);
  Mat sharp = channels[2];
```

```
  Mat imgLaplacian = laplacian(sharp);
  imgLaplacian = laplacian(imgLaplacian);



  channels[2].convertTo(sharp, CV_32F);

  Mat subPlans = sharp - imgLaplacian;
  subPlans = erode(subPlans, imgLaplacian);



  Mat sureFigure, unknown;
  Mat markers = getMarkers(subPlans, sureFigure, unknown);

  EXPECT_EQ(0, unknown.at<double>(100, 60));
}

TEST(markerTest, grainTest1)
{
  Mat img = imread("rice.png");
  Mat hsv;
  cvtColor(img, hsv, CV_BGR2HSV);

  vector<Mat> channels;
  split(hsv, channels);
  Mat sharp = channels[2];
```

```
  Mat imgLaplacian = laplacian(sharp);
  imgLaplacian = laplacian(imgLaplacian);



  channels[2].convertTo(sharp, CV_32F);

  Mat subPlans = sharp - imgLaplacian;
  subPlans = erode(subPlans, imgLaplacian);



  Mat sureFigure, unknown;
  Mat markers = getMarkers(subPlans, sureFigure, unknown);


  EXPECT_GT(0, unknown.at<double>(25, 5)-EPS);
}

TEST(markerTest, grainTest2)
{
  Mat img = imread("rice.png");
  Mat hsv;
  cvtColor(img, hsv, CV_BGR2HSV);

  vector<Mat> channels;
  split(hsv, channels);
  Mat sharp = channels[2];
```

```
  Mat imgLaplacian = laplacian(sharp);
  imgLaplacian = laplacian(imgLaplacian);



  channels[2].convertTo(sharp, CV_32F);

  Mat subPlans = sharp - imgLaplacian;
  subPlans = erode(subPlans, imgLaplacian);



  Mat sureFigure, unknown;
  Mat markers = getMarkers(subPlans, sureFigure, unknown);


  EXPECT_GT(0, unknown.at<double>(120, 5)-EPS);
}

TEST(getFileListTest, listTest1)
{
  std::string path = "EFKMD3";
  EXPECT_THROW(getFileList(path),std::invalid_argument);
}

TEST(getFileListTest, listTest2)
 {
```

```
  std::string path = "..";
  EXPECT_NO_THROW(getFileList(path));
}

TEST(getFileListTest, listTest3)
{
  std::string path = ".";
  EXPECT_NO_THROW(getFileList(path));
}

TEST(getFileListTest, listTest4)
{
  std::string path = "/home/oem/Practice/Rice/";
  EXPECT_NO_THROW(getFileList(path));
}
```