**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](#_Toc498360231)

[Задание №1. Знакомство с системой контроля версий Git 4](#_Toc498360232)

[Задание №2 …… 12](#_Toc498360233)

[Приложение А. Слайды презентации с семинара по С++11 на тему: …………………….. 18](#_Toc498360234)

# 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

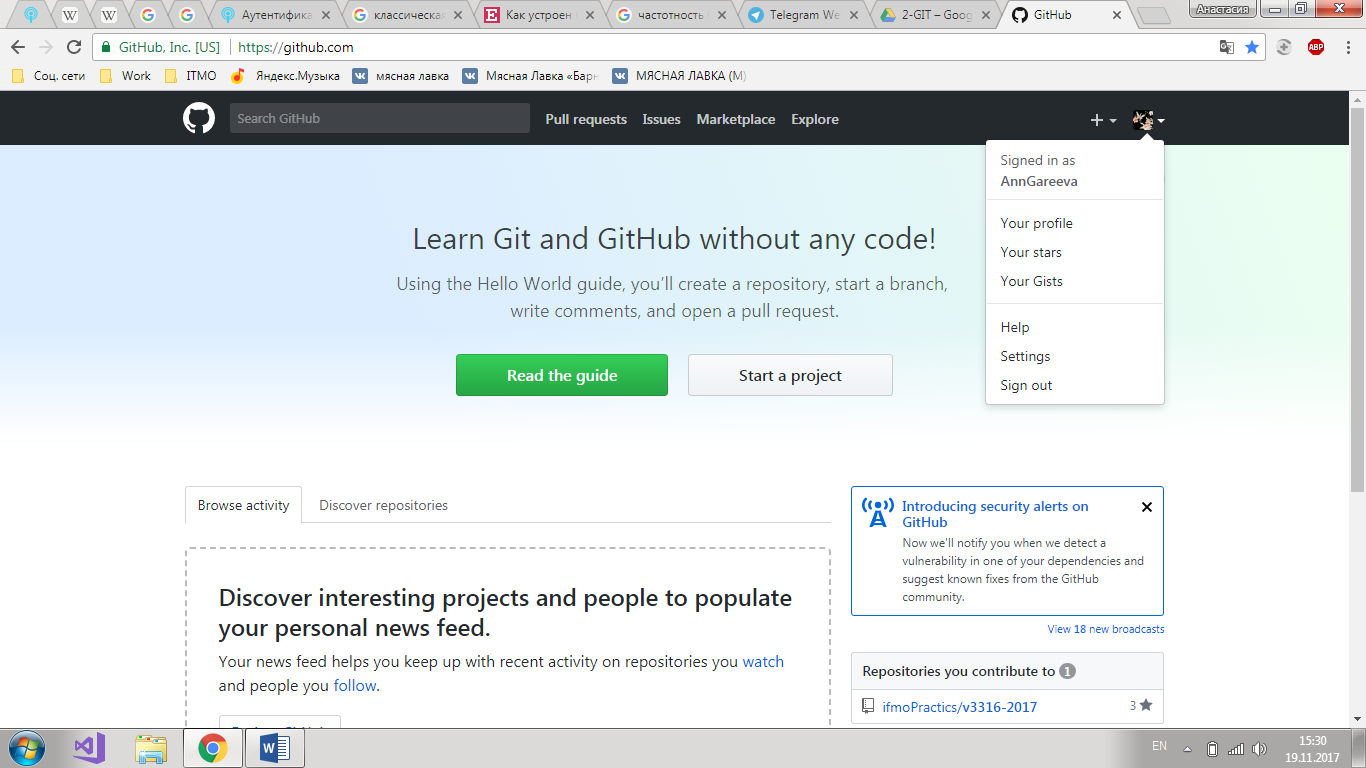
1. Завести аккаунт на github.com. На рис.1 скриншот с сайта github.com.  
   

Рисунок 1- создание своего аккаунта на сайте github.com

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

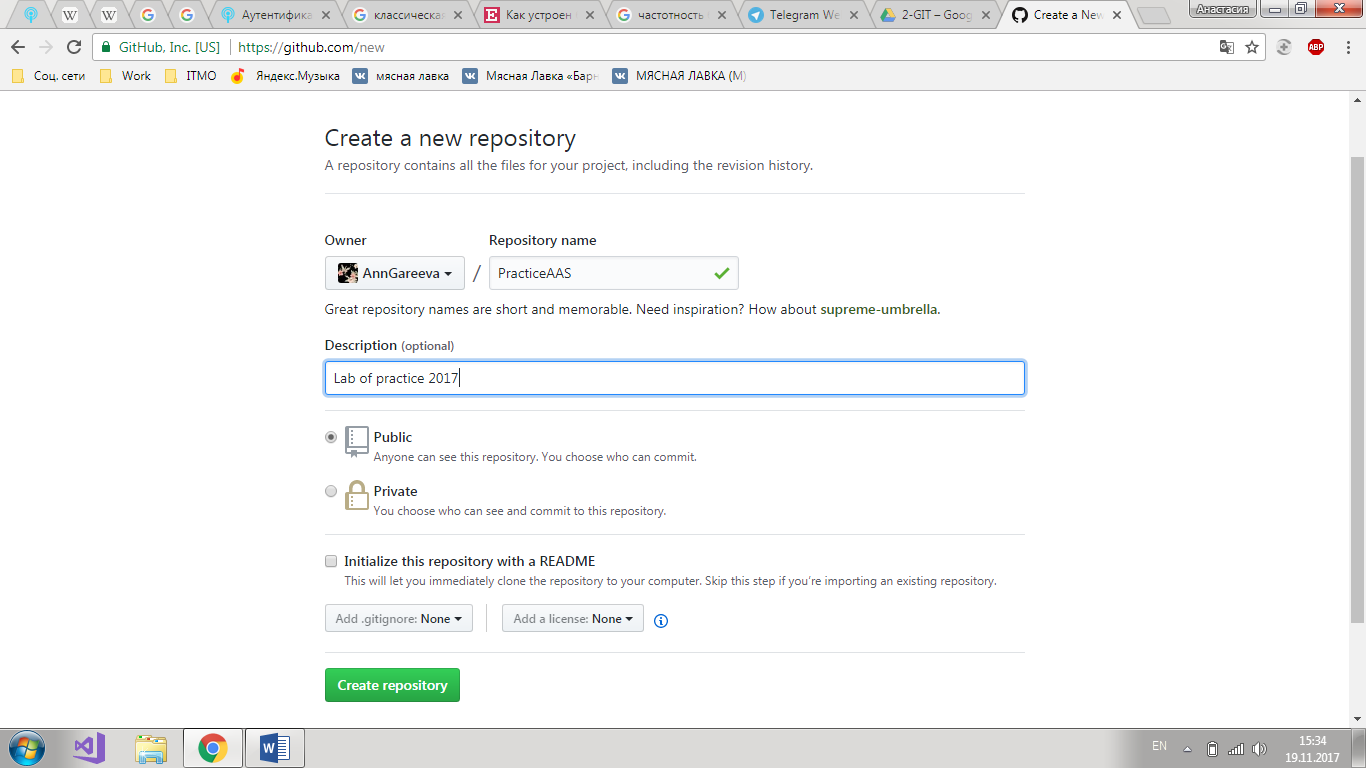


Рисунок 2-создание репозитория на сайте

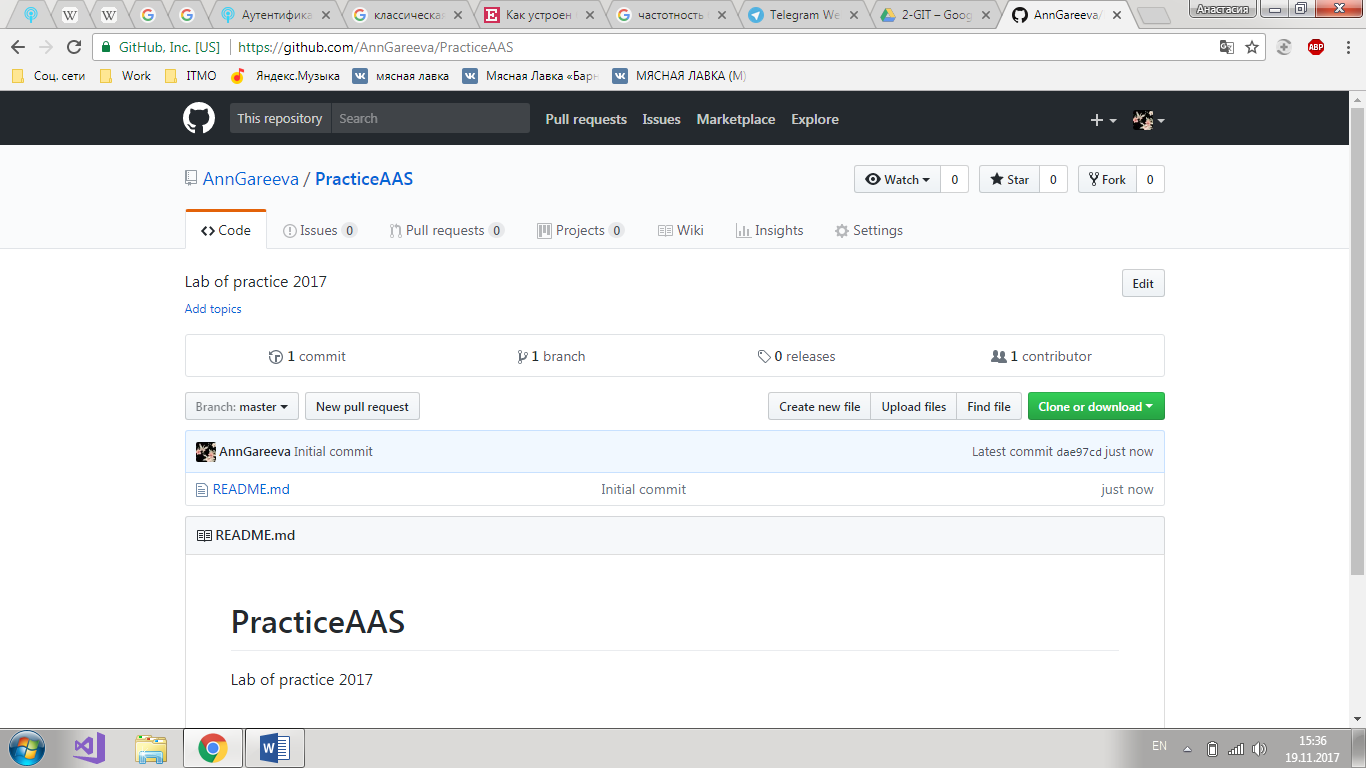


Рисунок 3-создание репозитория на сайте

1. Склонировать репозиторий на компьютер.

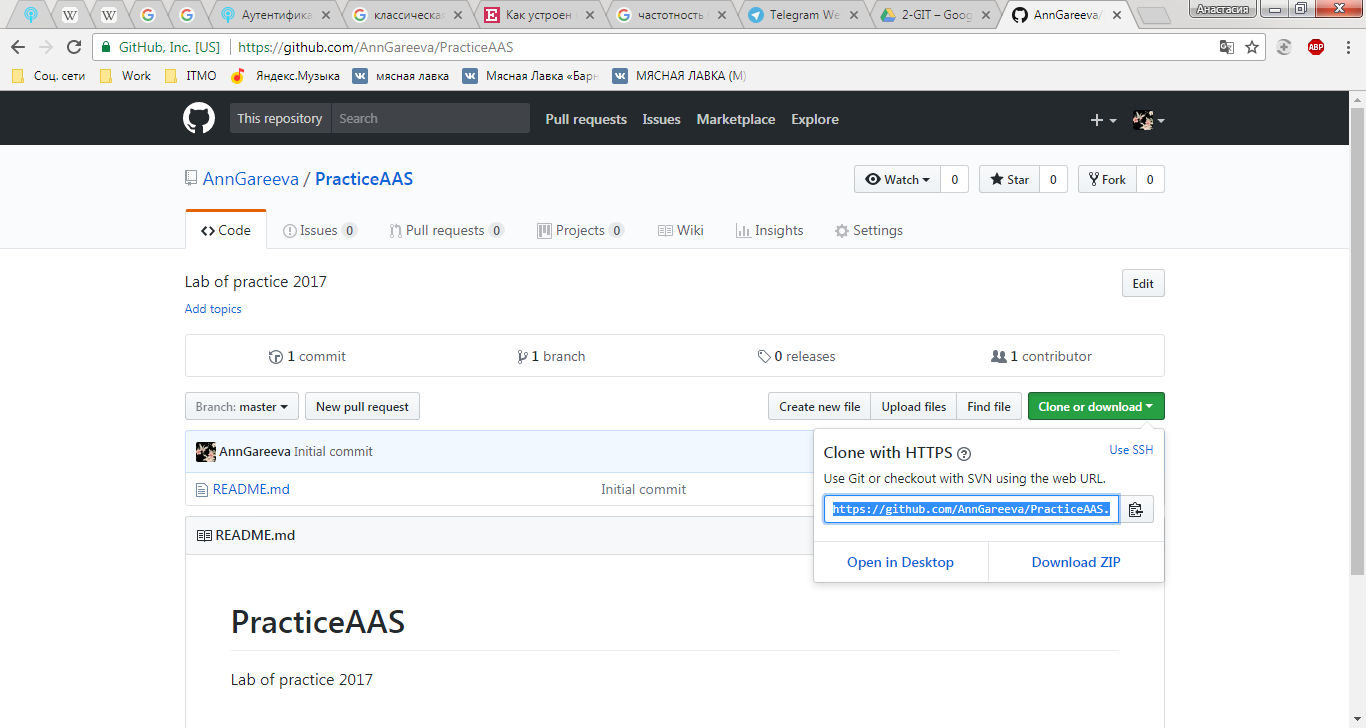
Для того, чтобы клонировать созданный репозиторий, необходимо скопировать на него ссылку. Именно это демонстрирует рис. 4.  


Рисунок 4-копирование ссылки на репозиторий для его клонирования

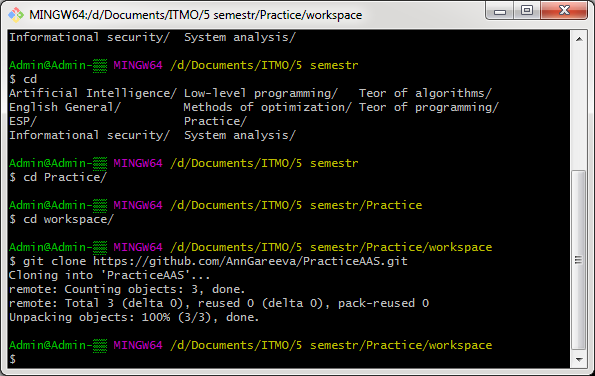
Скриншот Git-Bush показывает результаты выполнения команды git clone <ссылка> (рис. 5)  


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.



Рисунок 6- ветка master

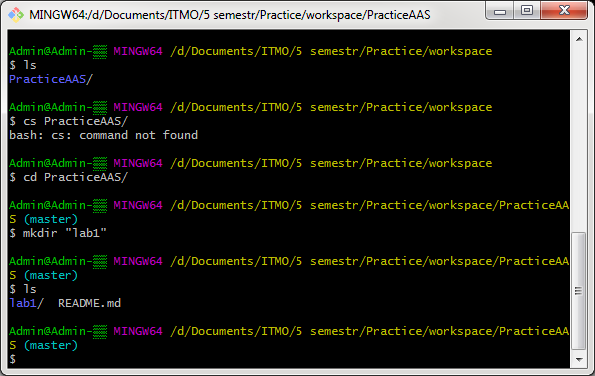
1. Создать папку lab1. С помощью команды mkdir “название”, создаем папку (рис.7) и переходим в нее.  
   

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

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

Код программы:

int Factorial(int a) {

return (a == 1) ? 1 : Factorial(a - 1)\*a;

}

1. 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.

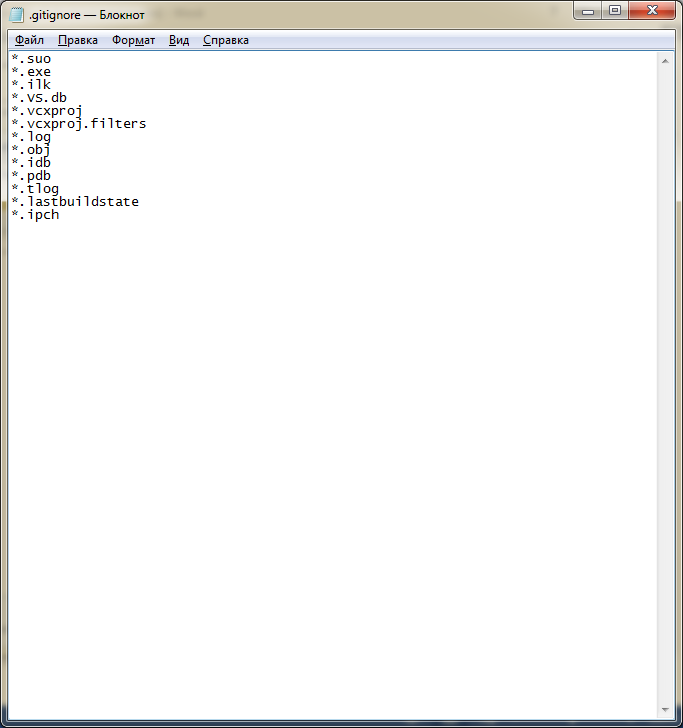
На рис.8 показан созданный файл .gitignore и его содержание, то есть указаны расширения, которые будут игнорироваться и не загружаться.   


Рисунок 8-содержимое файла .gitignore

1. Сделать по крайней мере два коммита.

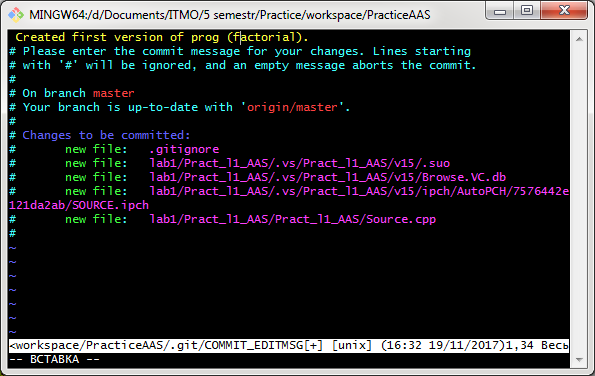
Скриншот ниже (рис. 8) показывает процесс создания коммита.

Рисунок 9- написание commit к первоначальному файлу программы

На рис.10 показан результат выполнения команды git commit.

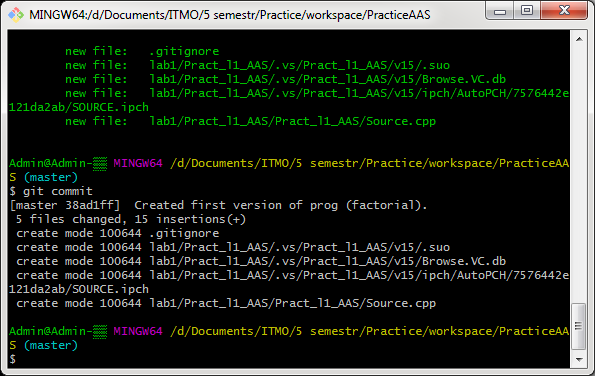
Создание второго коммита после изменения файла. Новый код:

Рисунок 10-результат выполнения команды git commit

#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;

}

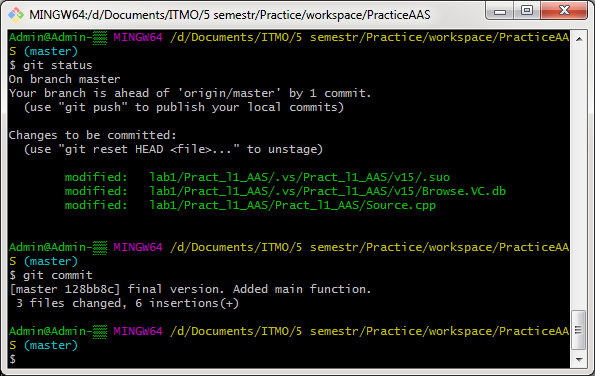
Результат выполения команды git commit показан на рис. 11.  


Рисунок 11-результат выполнений команды git commit после изменения файла

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

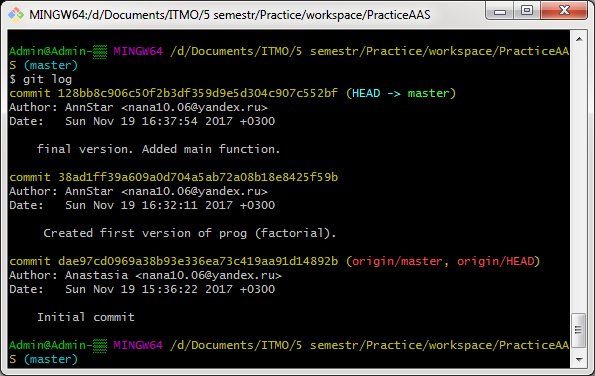


Рисунок 12-результат выполнения команды git log

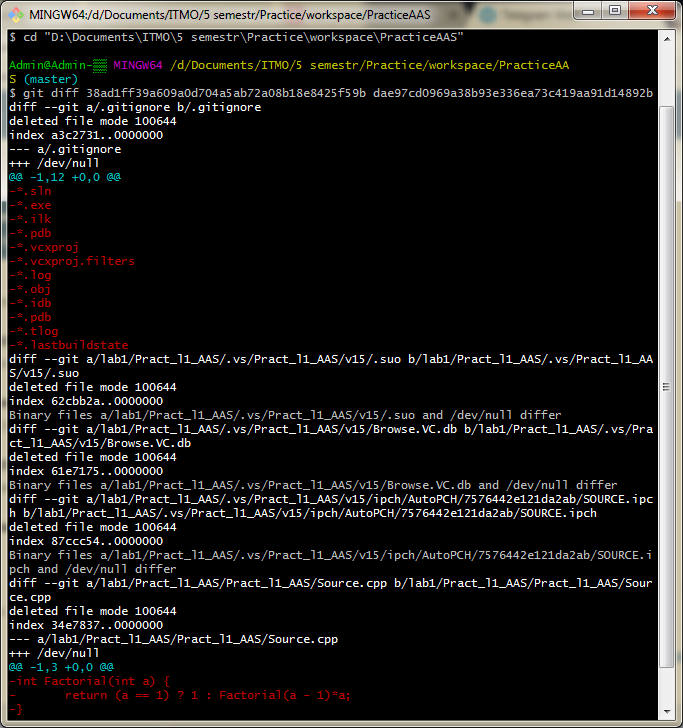
1. Показать при помощи “git diff” изменение любого файла между двумя коммитами. Результат выполнения данной команды показан на рис.13.  
   

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 *ai*, *i* = 0, 1, 2, …, удовлетворяющих следующим условиям:

* *a*0 = 0
* *a*1 = 1
* *a*2*i* = *ai*
* *a*2*i*+ 1 = *ai* + *ai*+ 1,

где для каждого *i* = 1, 2, 3, … .

Цель: написать программу, которая для заданного значения *n* находит максимальное среди чисел *a*0, *a*1, …, *an*.

Исходные данные:входные данные состоят из нескольких тестов (не более 10). Каждый тест представляет собой строку, в которой записано целое число *n* (1 ≤ *n* ≤ 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  10  0 | 3  4 |
| 10000  10000  0 | 512  512 |
| 0 |  |

## Алгоритм решения

В первую очередь, необходимо считать 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 хранится 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.

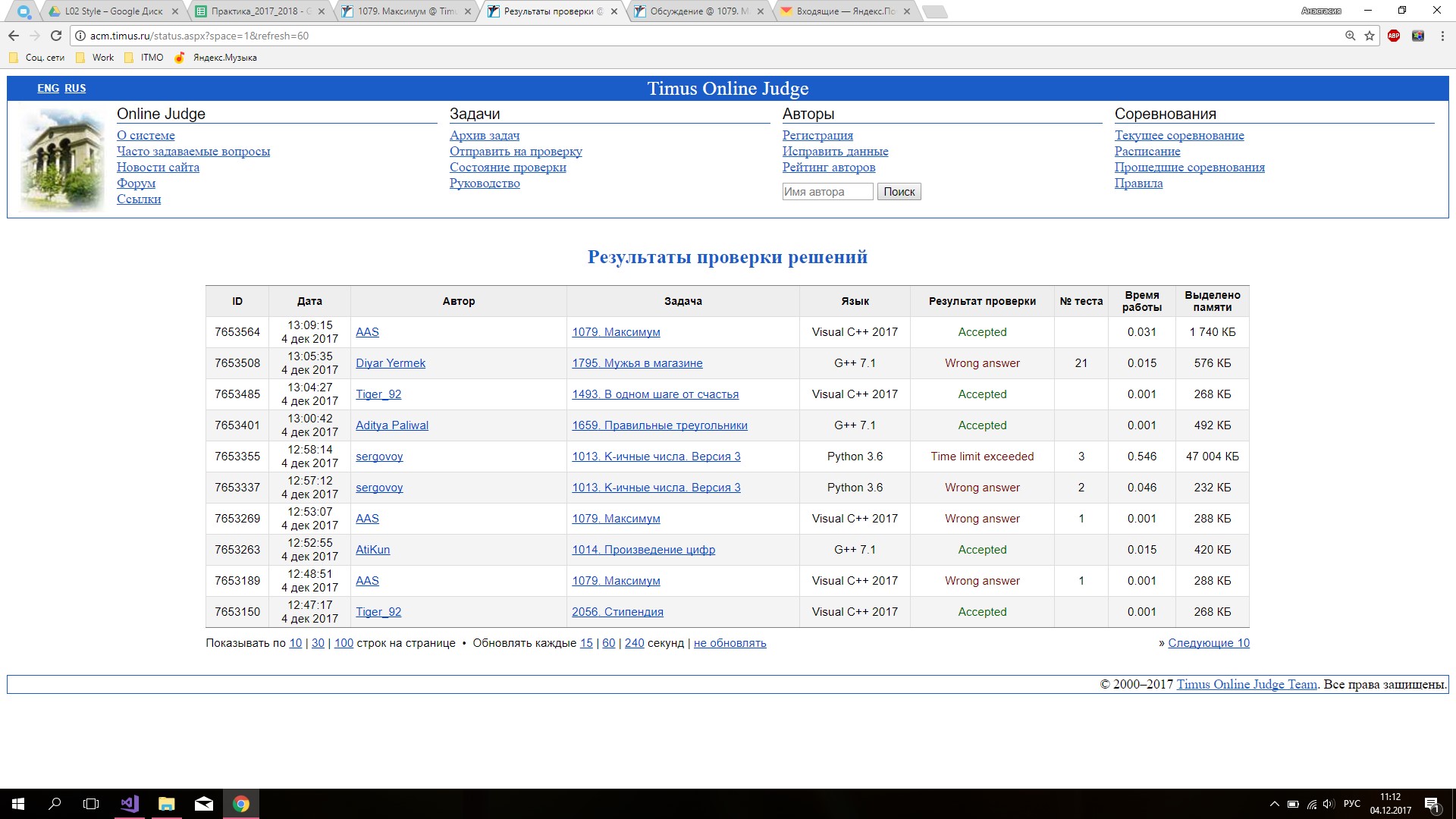


Рисунок 14- результаты прохождения тестов программой.

## Для 2й и последующих л/р:

1. Условия задания.
2. Схема алгоритма решения задачи (словесное описание пути решения и/или математическая модель задачи (формулы, алгоритмы …)).
3. Текст программы.
4. Исходные данные, используемые при вводе в программу.
5. Результаты выполнения программы.

## Дополнительные указания

1. Условие задания с сайта timus приводить полностью (включая номер и название задания, описание, исходные данные, результат и пример), лучше помещать задание в отчет как рисунок;
2. Помещать в отчет скриншот сайта timus с результатами проверки, включающий логин автора, номер задания и результат выполнения;
3. Предоставлять для проверки отчет, сохраненный в формате pdf (т.к. в разных версиях MS Word документ может отображаться по-разному, внося искажения в оформление отчета, что снижает оценку за оформление).
4. Бумажная версия отчета предоставляется в конце семестра. Оформление отчета по ГОСТ 7.32-200x.

## Основные требования и рекомендации по оформлению (составлено по наиболее частым ошибкам)

1. Отчет должен иметь титульный лист, оглавление и содержать отчеты по лабораторным работам, а также слайды презентации.
2. Страницы должны быть пронумерованы.
3. Для заголовков, подзаголовков, абзацев основного текста, списков и подписей рисунков/таблиц рекомендуется использовать стили. Это позволит обеспечить единообразное оформление и возможность быстрой корректировки оформления. Список стилей:

* Для заголовков: «Заголовок1», «Заголовок2», «Заголовок3».
* Для текста абзаца: «Обычный».
* Для рисунков и подписей: «Рисунок».
* Для списков: «Список-номер», «Список-тире».

1. Рисунки и таблицы должны иметь номера и названия. Не допускается расположение рисунка/таблицы и его подписи на разных страницах. Формулы должны быть пронумерованы. На рисунок и таблицу в тексте должны быть сноски. Примеры: (см. рис.1) (см. таблицу 1).
2. Презентацию доклада оформить в отчете как приложение.

# Приложение А. Слайды презентации с семинара по С++11 на тему: ……………………..