|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**по курсу объектно-ориентированное программирование**

Студент Камалов Антон Павлович

Группа РК6-25Б

Тип задания Лабораторная работа №8

Вариант F15

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Камалов А. П.**

*подпись, дата фамилия, и.о.*

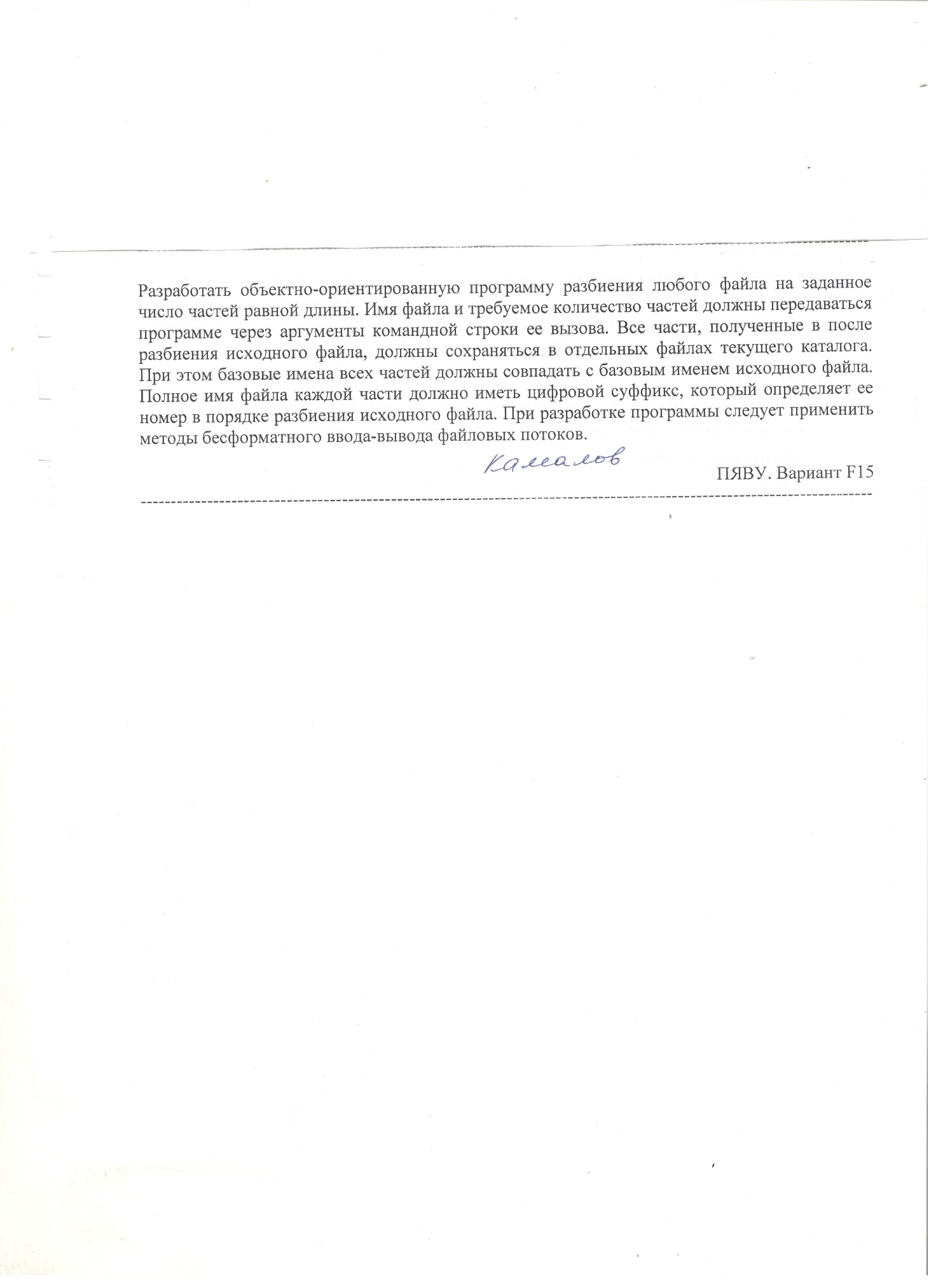
Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бочаров В. А.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г.*

**Задание**

****

**Описание программы**

**Входные данные:** имя файла и количество частей.

**Выходные данные:** части исходного файла с равными длинами.

**Класс File**

*Информационные поля:*

name (string) – название входного файла;

parts (int) – количество частей равной длины;

*Конструкторы и деструкторы:*

File (const string &, int) – конструктор инициализации;

*Методы:*

get\_name (string) – приватный метод получения имени нового файла;

split\_file () – публичный метод разделения исходного файла на равные части.

**Алгоритм решения:**

1. Проверка на количество аргументов. Если их меньше 3, то программа завершает свою работу.
2. Создаётся объект класса **File** с именем файла и количеством частей, переданные аргументами командной строки.
3. Вызов функции split\_file() для разделения входного файла на несколько других равной длины.
   1. Создаём файловый поток **in** для работы с входным файлом.
   2. С помощью метода open() открываем входной файл.
   3. Проверяется, успешность открытия файла и количество частей. Если одно из условий не прошло проверку, то выводится сообщение об ошибке, и программа завершает свою работу.
   4. Получаем размер **in** с помощью методов бесформатного ввода-вывода – seekg() и tellg().
   5. Проверяется, что размер **in** больше или равен количеству частей. В противном случае выводится сообщение об ошибке, и программа завершает свою работу.
   6. Вычисляется размер каждой части файла. Результат записывается в переменную **num\_parts**.
   7. Создаём файловый поток **out** для работы с выходным файлом. С помощью метода open() открываем выходной файл с именем, полученным с помощью метода get\_name(i+1), где i – цифровой суффикс.
   8. Записываем в буферную переменную **buf** **num\_parts** символов из исходного файла.
   9. С помощью метода write() записываем в поток **out num\_parts** символов из **buf**.
   10. Закрываем выходной поток out.
   11. Повторяем 3.7 – 3.11 до тех пор, пока не создастся нужное количество частей. После завершения закрываем поток **in**.

**Список литературы**

1. Пол Айра. Объектно-ориентированное программирование на C++: пер. с англ. 2-е изд. СПБ.: Невский Диалект; М.: Издательство БИНОМ, 2001. 462 с. [Pohl Ira. Object-Oriented Programming Using C++. 2nd ed. Addison-Wesley, 1996. 576 p.].
2. Страуструп Б. Язык программирования C++: пер. с англ / под ред. Н.Н. Мартынова. Специальное изд. М.: Бином, 2011. 1035 c. [Stroustrup B. The C++ Programming Language. Special ed. Addison-Wesley, 2000. 1029p.].
3. Шилдт Г. Самоучитель C++. 3-е изд. СПБ.: БХВ-Петербург, 2002. 688 с. [Schildt H. Teach Yourself C++. 3d ed. McGraw-Hill, 1998. 768 p.].
4. Волосатова Т.М., Родионов С.В. Объектно-ориентированное программирование на С++. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou (дата обращения 29.05.2023).

**Приложение 1**

Текст программы

#include <iostream>

#include <cstdlib>

#include <fstream>

#include <string>

using namespace std;

class File{

private:

string name;

int parts;

string get\_name(int index){

return name+ "\_" + to\_string(index);

}

public:

File (const string &file\_name, int p) : name(file\_name), parts(p) {};

void split\_file();

};

void File::split\_file(){

int part\_size, text\_length;

ifstream in;

in.open(name, ios::binary);

if (!in){

cerr << "ERROR! FILE DOESN'T EXIST!";

exit(-2);

}

if (parts < 1){

cerr << "ERROR! PARTS LESS THAN 1!";

exit(-3);

}

text\_length = in.seekg(0, ios::end).tellg();

in.seekg(0, ios::beg);

if (text\_length < parts){

cerr << "ERROR! SIZE OF FILE LESS THAN PARTS!";

exit(-4);

}

part\_size = text\_length / parts;

for (int i = 0; i < parts; i++){

ofstream out;

char buf[512];

out.open(get\_name(i+1));

in.read(buf, part\_size);

out.write(buf, part\_size);

out.close();

}

in.close();

}

int main(int argc, char \*\*argv){

if (argc < 3) return -1;

string name;

int num\_parts;

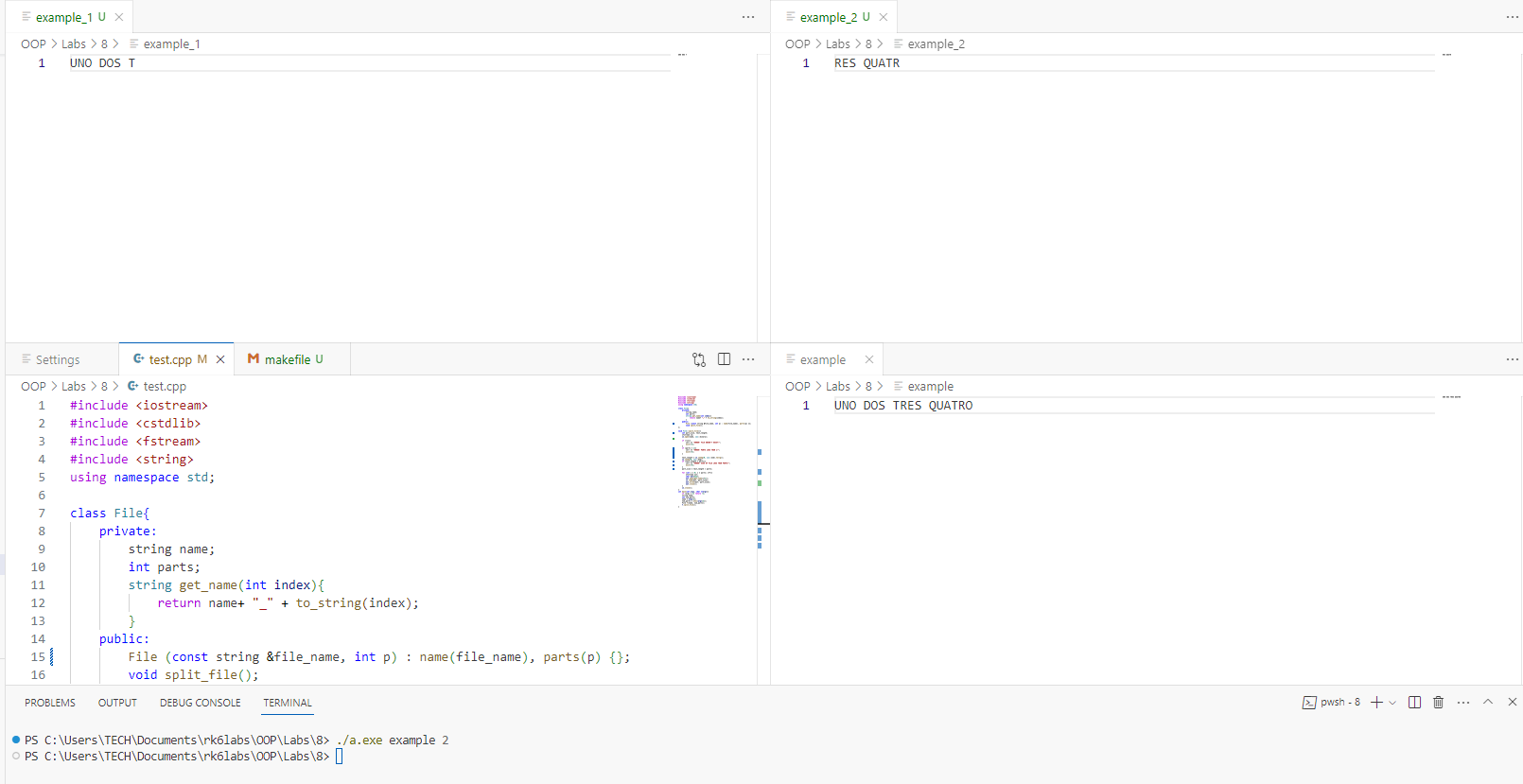
name = argv[1];

num\_parts = atoi(argv[2]);

File f(name, num\_parts);

f.split\_file();

**Приложение 2**

Результат работы программы