МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА  
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., канд. техн. наук |  |  |  | Курицын К.А. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ «ПЕРИФЕРИЯ» |
| по дисциплине: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 7541ВЦ |  |  |  | Шешнев А.В. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

Содержание

1. Задание ……………………………………………………………………………...2
2. Функциональные требования……………………………………………………...2
3. Требование к оборудованию и стороннему ПО…………………………….…….2
4. Технические спецификации……………………………………………………..2-3
5. Взаимодействие с внешними и внутренними системами………………………..3
6. Программа и методика испытаний……………………………………………......3
7. Литература…………………………………………………………………………..4

Приложение 1 – Описание формата……………………………………………....…...5

Приложение 2 – Листинг………………………………………………………….....6-14

Приложение 3 – Примеры, скриншоты…………………………………………….15-21

1. **Задание**

Определить класс «Периферия». Базовый класс определяет интерфейс для построения периферийного оборудования для персонального компьютера, но они могут быть специализированы для конкретного устройства (принтер, сканер, МФУ, графический планшет). Каждое устройство обладает набором характеристик: производитель, год выпуска, цветность печати, стоимость, цвет устройства, интерфейсы, максимальный формат, скорость печати, площадь сканирования. Использовать паттерн «Одиночка». Результат и периферийное оборудование записываются в отдельные выходные файлы. Пользователь может загрузить все данные о ранее созданных устройствах из файлов.

1. **Функциональные требования**

2.1) Выход из приложения

Реализовать выход из приложения.

2.2) Добавление периферийных устройств.

Реализовать добавление новых периферийных устройств в файл-каталог. Добавление должно происходить либо из консоли, либо из файла.

2.3) Запись результата

Результат записывается в txt файл формата указанного в Приложении 1. Для каждого из типов периферийных устройств (принтер, сканнер, МФУ, графический планшет) должен быть предусмотрен отдельный файл-каталог. Файл должен быть доступен для просмотра в любое время.

2.4) Вывод результата в консоль из файла для конкретного устройства.

2.5) Вывод результата в консоль из файла для всех устройств.

2.6) Удобный интерфейс

Интерфейс должен быть удобным и понятным. Необходимо очищать консоль от лишних элементов.

2.7) Язык С++

Приложение должно быть реализовано при помощи языка С++ и использовании классов.

1. **Требования к оборудованию и стороннему ПО**

Консольное приложение. Операционная система Windows 10 и Visual Studio 2017.

1. **Технические спецификации**

4.1) При запуске приложения перед пользователем отображается меню, где предложены варианты действий: |1| - Создать устройство; |2| -Загрузить устройство из файлов; |0| - Выход.

4.2) Нулевой пункт отвечает за выход из приложения.

4.3) Первый пункт отвечают за создание одного из типов периферийных устройств (1 – принтер, 2 – сканер, 3 – МФУ, 4 – графический планшет). Когда выбран один из вариантов, пользователь может ввести все данные о периферийном устройстве, при этом такие параметры, как производитель, год выпуска, цветность печати, стоимость, цвет устройства, интерфейсы, максимальный формат, скорость печати, площадь сканирования. Результаты запишутся в файл-каталог и выведутся на экран автоматически. После завершения ввода все данные будут записаны в соответствующий каталог, а пользователю будет предложено: «создать устройство», «загрузить устройство из файлов» или «выход».

4.4) Второй пункт отвечает за вывод результата из файлов в консоль. После выбора действия «Загрузить устройство из файлов» будет предложено выбрать загрузить конкретное устройство или все сразу.

4.5) Все внесенные данные должны автоматически сохранятся в файл-каталог

1. **Взаимодействия с внешними и внутренними системами**

Взаимодействие типа консоль-файл. Приложение взаимодействует с четырьмя файлами: printer.txt, skaner.txt, mfu.txt, Graph\_plan.txt.

1. **Программа и методика испытаний**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Сценарий проверки | Ожидаемый результат | ФТ | Тех. Спец. | Результат |
| 1 | Включение приложения | Отображение названия программы и всех пунктов для работы с файлом каталога:  0)Выход  1) Создать устройство  2) Загрузить устройство из файлов | - | 4.1 | Готово  Рисунок 1 |
| 2 | В меню выбрать 0 | Выход из приложения | 2.1 | 4.2 | Готово |
| 3 | В меню выбрать 1 | Создание периферийного устройства | 2.2 | 4.3 | Готово  Рисунок 2  Рисунок 3  Рисунок 4 Рисунок 5 |
| 4 | В меню выбрать 2 | Загрузить устройство из файлов | 2.4  2.5 | 4.4 | Готово  Рисунок 6  Рисунок 7 |
| 5 | Возможность просмотра файла-каталога, рабочего файла | Отображение актуальной информации | 2.3 | 4.5 | Готово  Рисунок 8  Рисунок 9  Рисунок 10  Рисунок 11 |
| 6 | Возможность создавать новые периферийные устройства, не выходя из приложения | Создание, не приводящее к ошибкам | 2.2  2.6 | 4.1  4.2 | Готово |

1. **Литература**

* https://support.microsoft.com
* http://www.cyberforum.ru
* https://ru.stackoverflow.com
* http://cppstudio.com
* http://www.c-cpp.ru
* Б. Керниган и Д. Ритчи – Язык программирования Си
* Шилдт Г. Самоучитель С++
* Бьерн Страуструп - Язык программирования С++

**Приложение №1 Описание формата**

Формат файла должен отвечать следующим требованиям:

1. Разделение периферийных устройств

Для разделения периферийных устройств необходимо использовать специальный разделитель, на отдельной строке в начале и в конце каждой записи:

«------------------------------»

1. Информация о периферийном устройстве

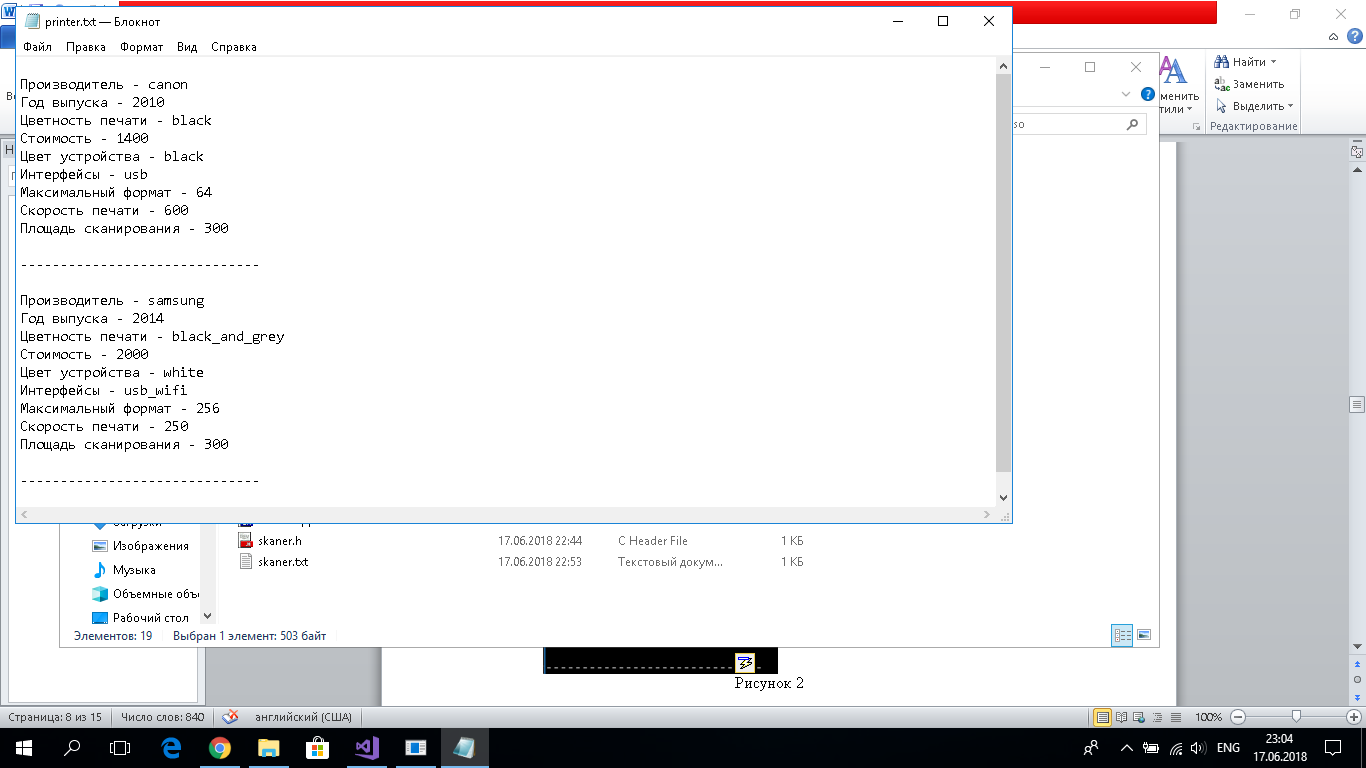
Информация о каждом периферийном устройстве должна быть представлена в виде Набора данных разделённого построчно. Каждая строка отвечает за отдельный тип информации.

1. Порядок представления информации

Информация о периферийном устройстве должна быть представлена в следующем порядке (сверху вниз, построчно):

Производитель, год выпуска, цветность печати, стоимость, цвет устройства, интерфейсы, максимальный формат, скорость печати, площадь сканирования.

1. Текстовый файл-каталог имеет стандартную кодировку ANSI
2. Перевод строки осуществляется с помощью \n
3. Пример файла: printer.txt



Все строчные переменные должны быть представлены в виде одного слова. Для разделения слов использовать символ: «\_».

**Приложение №2 Листинг**

Ссылка на проект в Github: https://github.com/Sheshnev/kursovik

Main.cpp

#include "Pereferia.h"

#include <iostream>

#include <locale>

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

Pereferia s;

int button = 1;

while (button) {

cout << "Выберите действие:\n";

cout << "|1| - Создать устройство\n";

cout << "|2| - Загрузить устройства из файлов\n";

cout << "|0| - Выход.\n";

cin >> button;

switch (button) {

case 1: s.Create\_Pereferia(); break;

case 2: s.Read(); break;

case 0: break;

}

}

return 0;

}

Pereferia.cpp

#include "printer.h"

#include "skaner.h"

#include "mfu.h"

#include "Graph\_plan.h"

#include "Pereferia.h"

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

Pereferia::Pereferia() {}

void Pereferia::Create\_Pereferia() {

cout << "Выберите устройство:\n";

cout << "|1| - Принтер\n";

cout << "|2| - Сканер\n";

cout << "|3| - МФУ\n";

cout << "|4| - Графический планшет\n";

cin >> peref;

switch (peref) {

case 1: {

printer \* per = new printer();

per->Vivod\_in\_not\_fail();

per->Vivod\_in\_fail();

delete per;

break;

}

case 2: {

skaner \* per = new skaner();

per->Vivod\_in\_not\_fail();

per->Vivod\_in\_fail();

delete per;

break;

}

case 3: {

mfu \* per = new mfu();

per->Vivod\_in\_not\_fail();

per->Vivod\_in\_fail();

delete per;

break;

}

case 4: {

Graph\_plan \* per = new Graph\_plan();

per->Vivod\_in\_not\_fail();

per->Vivod\_in\_fail();

delete per;

break;

}

case 0: break;

}

}

void Pereferia::Read() {

cout << "Вы хотите загрузить конкретного устройство или все сразу:\n";

cout << "|1| - Конкретное\n";

cout << "|2| - Все сразу\n";

cin >> peref;

switch (peref) {

case 1: {

cout << "Выберите устройство:\n";

cout << "|1| - Принтер\n";

cout << "|2| - Сканер\n";

cout << "|3| - МФУ\n";

cout << "|4| - Графический планшет\n";

cin >> peref;

switch (peref) {

case 1: cout << "\n\tПринтеры\n"; Read("printer.txt"); break;

case 2: cout << "\n\tСканеры\n"; Read("skaner.txt"); break;

case 3: cout << "\n\tМФУ\n"; Read("mfu.txt"); break;

case 4: cout << "\n\tГрафические планшеты\n"; Read("Graph\_plan.txt"); break;

}

break;

}

case 2: {

cout << "\n\tПринтеры\n"; Read("printer.txt");

cout << "\n\tСканеры\n"; Read("skaner.txt");

cout << "\n\tМФУ\n"; Read("mfu.txt");

cout << "\n\tГрафические планшеты\n"; Read("Graph\_plan.txt");

break;

}

}

}

void Pereferia::Read(const char name1[]) {

ifstream file2(name1, ios\_base::in);

while (!file2.eof()) {

file2 >> qqqq;

if (qqqq == '.') cout << endl;

else if (qqqq == '\32') cout << " ";

else {

if (qqqq != '-') {

if (qqqq == 'Т' || qqqq == 'П' || qqqq == 'К' || qqqq == 'Ц' || qqqq == 'Д' || qqqq == 'В')

cout << endl;

cout << qqqq;

}

else cout << " " << qqqq << " " ;

}

}

file2.close();

}

Pereferia::~Pereferia() {

cout << "Конец работы!";

}

Pereferia.h

#ifndef Pereferia\_H

#define Pereferia\_H

class Pereferia

{

private:

int peref = 1;

char qqqq;

public:

Pereferia();

void Create\_Pereferia();

void Read();

void Read(const char name1[]);

Pereferia(const Pereferia&) = delete;

Pereferia & operator=(const Pereferia&) = delete;

void\* operator new(size\_t) = delete;

void\* operator new[](size\_t) = delete;

void operator delete(void\*) = delete;

void operator delete[](void\*) = delete;

~Pereferia();

};

#endif Pereferia\_H

Printer.cpp

#include "printer.h"

#include <iostream>

#include <fstream>

using namespace std;

printer::printer() {

cout << "Введите производителя\n";

cin >> proizv;

cout << "Год выпуска\n";

cin >> god;

cout << "Введите цветность печати\n";

cin >> cvet\_pechati;

cout << "Введите стоимость\n";

cin >> stoimost;

cout << "Введите цвет устройства\n";

cin >> cvet\_ustva;

cout << "Введите интерфейсы\n";

cin >> interfeci;

cout << "Введите максимальный формат\n";

cin >> max\_format;

cout << "Введите Скорость печати\n";

cin >> skorost\_pechati;

cout << "Введите площадь сканирования\n";

cin >> ploshad\_skana;

}

void printer::Vivod\_in\_not\_fail() {

cout << endl << "Производитель - " << proizv << endl;

cout << "Год выпуска - " << god << endl;

cout << "Цветность печати - " << cvet\_pechati << endl;

cout << "Стоимость - " << stoimost << endl;

cout << "Цвет устройства - " << cvet\_ustva << endl;

cout << "Интерфейсы - " << interfeci << endl;

cout << "Максимальный формат - " << max\_format << endl;

cout << "Скорость печати - " << skorost\_pechati << endl;

cout << "Площадь сканирования - " << ploshad\_skana << endl << endl;

cout << "------------------------------" << endl;

}

void printer::Vivod\_in\_fail() {

ofstream file1("printer.txt", ios\_base::app);

file1 << endl << "Производитель - " << proizv << endl;

file1 << "Год выпуска - " << god << endl;

file1 << "Цветность печати - " << cvet\_pechati << endl;

file1 << "Стоимость - " << stoimost << endl;

file1 << "Цвет устройства - " << cvet\_ustva << endl;

file1 << "Интерфейсы - " << interfeci << endl;

file1 << "Максимальный формат - " << max\_format << endl;

file1 << "Скорость печати - " << skorost\_pechati << endl;

file1 << "Площадь сканирования - " << ploshad\_skana << endl << endl;

file1 << "------------------------------" << endl;

file1.close();

}

printer::~printer() {}

Printer.h

#ifndef printer\_H

#define printer\_H

class printer {

char proizv[20];

int god;

char cvet\_pechati[20];

int stoimost;

char cvet\_ustva[20];

char interfeci[20];

int max\_format;

int skorost\_pechati;

int ploshad\_skana;

public:

printer();

void Vivod\_in\_not\_fail();

void Vivod\_in\_fail();

~printer();

};

#endif printer\_H

Skaner.cpp

#include "skaner.h"

#include <iostream>

#include <fstream>

using namespace std;

skaner::skaner() {

cout << "Введите производителя\n";

cin >> proizv;

cout << "Год выпуска\n";

cin >> god;

cout << "Введите цветность печати\n";

cin >> cvet\_pechati;

cout << "Введите стоимость\n";

cin >> stoimost;

cout << "Введите цвет устройства\n";

cin >> cvet\_ustva;

cout << "Введите интерфейсы\n";

cin >> interfeci;

cout << "Введите максимальный формат\n";

cin >> max\_format;

cout << "Введите Скорость печати\n";

cin >> skorost\_pechati;

cout << "Введите площадь сканирования\n";

cin >> ploshad\_skana;

}

void skaner::Vivod\_in\_not\_fail() {

cout << endl << "Производитель - " << proizv << endl;

cout << "Год выпуска - " << god << endl;

cout << "Цветность печати - " << cvet\_pechati << endl;

cout << "Стоимость - " << stoimost << endl;

cout << "Цвет устройства - " << cvet\_ustva << endl;

cout << "Интерфейсы - " << interfeci << endl;

cout << "Максимальный формат - " << max\_format << endl;

cout << "Скорость печати - " << skorost\_pechati << endl;

cout << "Площадь сканирования - " << ploshad\_skana << endl << endl;

cout << "------------------------------" << endl;

}

void skaner::Vivod\_in\_fail() {

ofstream file1("skaner.txt", ios\_base::app);

file1 << endl << "Производитель - " << proizv << endl;

file1 << "Год выпуска - " << god << endl;

file1 << "Цветность печати - " << cvet\_pechati << endl;

file1 << "Стоимость - " << stoimost << endl;

file1 << "Цвет устройства - " << cvet\_ustva << endl;

file1 << "Интерфейсы - " << interfeci << endl;

file1 << "Максимальный формат - " << max\_format << endl;

file1 << "Скорость печати - " << skorost\_pechati << endl;

file1 << "Площадь сканирования - " << ploshad\_skana << endl << endl;

file1 << "------------------------------" << endl;

file1.close();

}

skaner::~skaner() {}

Skaner.h

#ifndef skaner\_H

#define skaner\_H

class skaner {

char proizv[20];

int god;

char cvet\_pechati[20];

int stoimost;

char cvet\_ustva[20];

char interfeci[20];

int max\_format;

int skorost\_pechati;

int ploshad\_skana;

public:

skaner();

void Vivod\_in\_not\_fail();

void Vivod\_in\_fail();

~skaner();

};

#endif skaner\_H

Mfu.cpp

#include "mfu.h"

#include <iostream>

#include <fstream>

using namespace std;

mfu::mfu() {

cout << "Введите производителя\n";

cin >> proizv;

cout << "Год выпуска\n";

cin >> god;

cout << "Введите цветность печати\n";

cin >> cvet\_pechati;

cout << "Введите стоимость\n";

cin >> stoimost;

cout << "Введите цвет устройства\n";

cin >> cvet\_ustva;

cout << "Введите интерфейсы\n";

cin >> interfeci;

cout << "Введите максимальный формат\n";

cin >> max\_format;

cout << "Введите Скорость печати\n";

cin >> skorost\_pechati;

cout << "Введите площадь сканирования\n";

cin >> ploshad\_skana;

}

void mfu::Vivod\_in\_not\_fail() {

cout << endl << "Производитель - " << proizv << endl;

cout << "Год выпуска - " << god << endl;

cout << "Цветность печати - " << cvet\_pechati << endl;

cout << "Стоимость - " << stoimost << endl;

cout << "Цвет устройства - " << cvet\_ustva << endl;

cout << "Интерфейсы - " << interfeci << endl;

cout << "Максимальный формат - " << max\_format << endl;

cout << "Скорость печати - " << skorost\_pechati << endl;

cout << "Площадь сканирования - " << ploshad\_skana << endl << endl;

cout << "------------------------------" << endl;

}

void mfu::Vivod\_in\_fail() {

ofstream file1("mfu.txt", ios\_base::app);

file1 << endl << "Производитель - " << proizv << endl;

file1 << "Год выпуска - " << god << endl;

file1 << "Цветность печати - " << cvet\_pechati << endl;

file1 << "Стоимость - " << stoimost << endl;

file1 << "Цвет устройства - " << cvet\_ustva << endl;

file1 << "Интерфейсы - " << interfeci << endl;

file1 << "Максимальный формат - " << max\_format << endl;

file1 << "Скорость печати - " << skorost\_pechati << endl;

file1 << "Площадь сканирования - " << ploshad\_skana << endl << endl;

file1 << "------------------------------" << endl;

file1.close();

}

mfu::~mfu() {}

Mfu.h

#ifndef mfu\_H

#define mfu\_H

class mfu {

char proizv[20];

int god;

char cvet\_pechati[20];

int stoimost;

char cvet\_ustva[20];

char interfeci[20];

int max\_format;

int skorost\_pechati;

int ploshad\_skana;

public:

mfu();

void Vivod\_in\_not\_fail();

void Vivod\_in\_fail();

~mfu();

};

#endif mfu\_H

Graph\_plan.cpp

#include "Graph\_plan.h"

#include <iostream>

#include <fstream>

using namespace std;

Graph\_plan::Graph\_plan() {

cout << "Введите производителя\n";

cin >> proizv;

cout << "Год выпуска\n";

cin >> god;

cout << "Введите цветность печати\n";

cin >> cvet\_pechati;

cout << "Введите стоимость\n";

cin >> stoimost;

cout << "Введите цвет устройства\n";

cin >> cvet\_ustva;

cout << "Введите интерфейсы\n";

cin >> interfeci;

cout << "Введите максимальный формат\n";

cin >> max\_format;

cout << "Введите Скорость печати\n";

cin >> skorost\_pechati;

cout << "Введите площадь сканирования\n";

cin >> ploshad\_skana;

}

void Graph\_plan::Vivod\_in\_not\_fail() {

cout << endl << "Производитель - " << proizv << endl;

cout << "Год выпуска - " << god << endl;

cout << "Цветность печати - " << cvet\_pechati << endl;

cout << "Стоимость - " << stoimost << endl;

cout << "Цвет устройства - " << cvet\_ustva << endl;

cout << "Интерфейсы - " << interfeci << endl;

cout << "Максимальный формат - " << max\_format << endl;

cout << "Скорость печати - " << skorost\_pechati << endl;

cout << "Площадь сканирования - " << ploshad\_skana << endl << endl;

cout << "------------------------------" << endl;

}

void Graph\_plan::Vivod\_in\_fail() {

ofstream file1("Graph\_plan.txt", ios\_base::app);

file1 << endl << "Производитель - " << proizv << endl;

file1 << "Год выпуска - " << god << endl;

file1 << "Цветность печати - " << cvet\_pechati << endl;

file1 << "Стоимость - " << stoimost << endl;

file1 << "Цвет устройства - " << cvet\_ustva << endl;

file1 << "Интерфейсы - " << interfeci << endl;

file1 << "Максимальный формат - " << max\_format << endl;

file1 << "Скорость печати - " << skorost\_pechati << endl;

file1 << "Площадь сканирования - " << ploshad\_skana << endl << endl;

file1 << "------------------------------" << endl;

file1.close();

}

Graph\_plan::~Graph\_plan() {}

Graph\_plan.h

#ifndef Graph\_plan\_H

#define Graph\_plan\_H

class Graph\_plan {

char proizv[20];

int god;

char cvet\_pechati[20];

int stoimost;

char cvet\_ustva[20];

char interfeci[20];

int max\_format;

int skorost\_pechati;

int ploshad\_skana;

public:

Graph\_plan();

void Vivod\_in\_not\_fail();

void Vivod\_in\_fail();

~Graph\_plan();

};

#endif Graph\_plan\_H

**Приложение №3 Примеры**

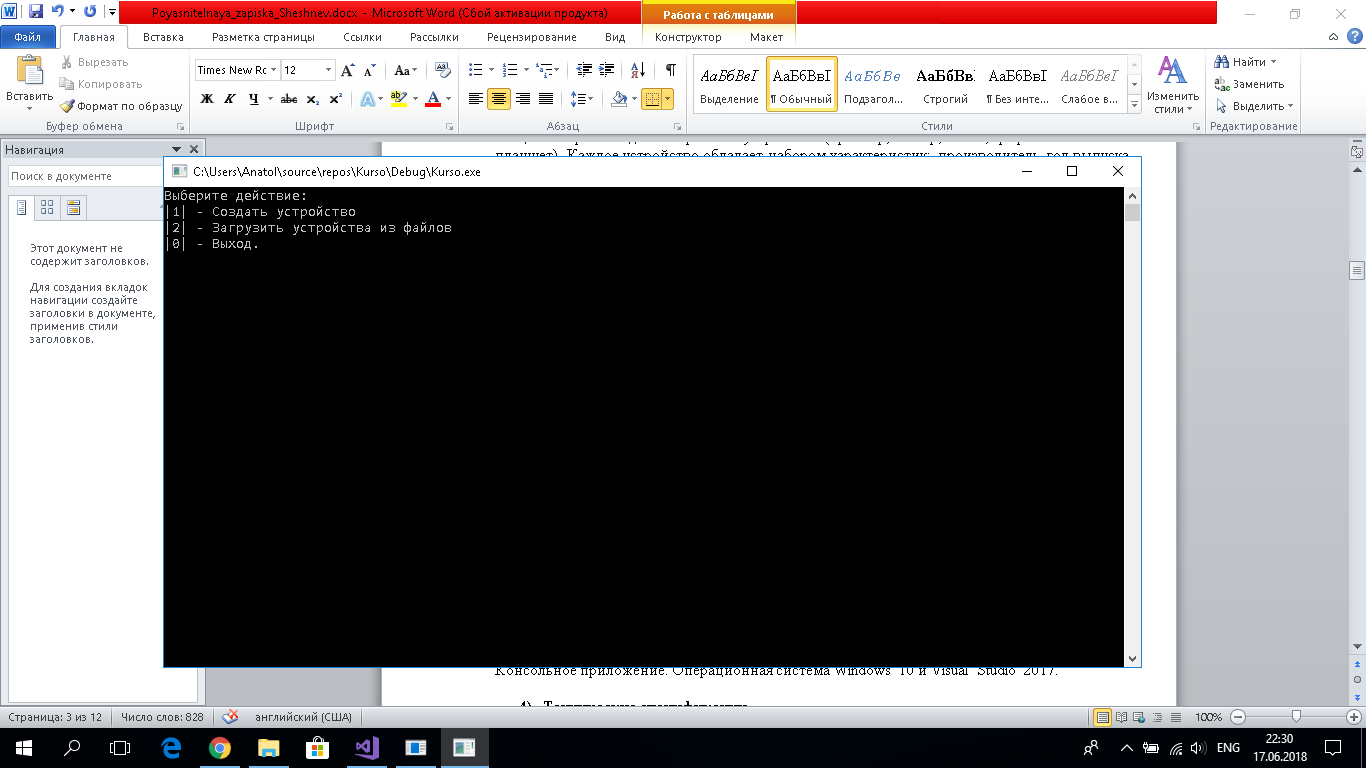


Рисунок 1

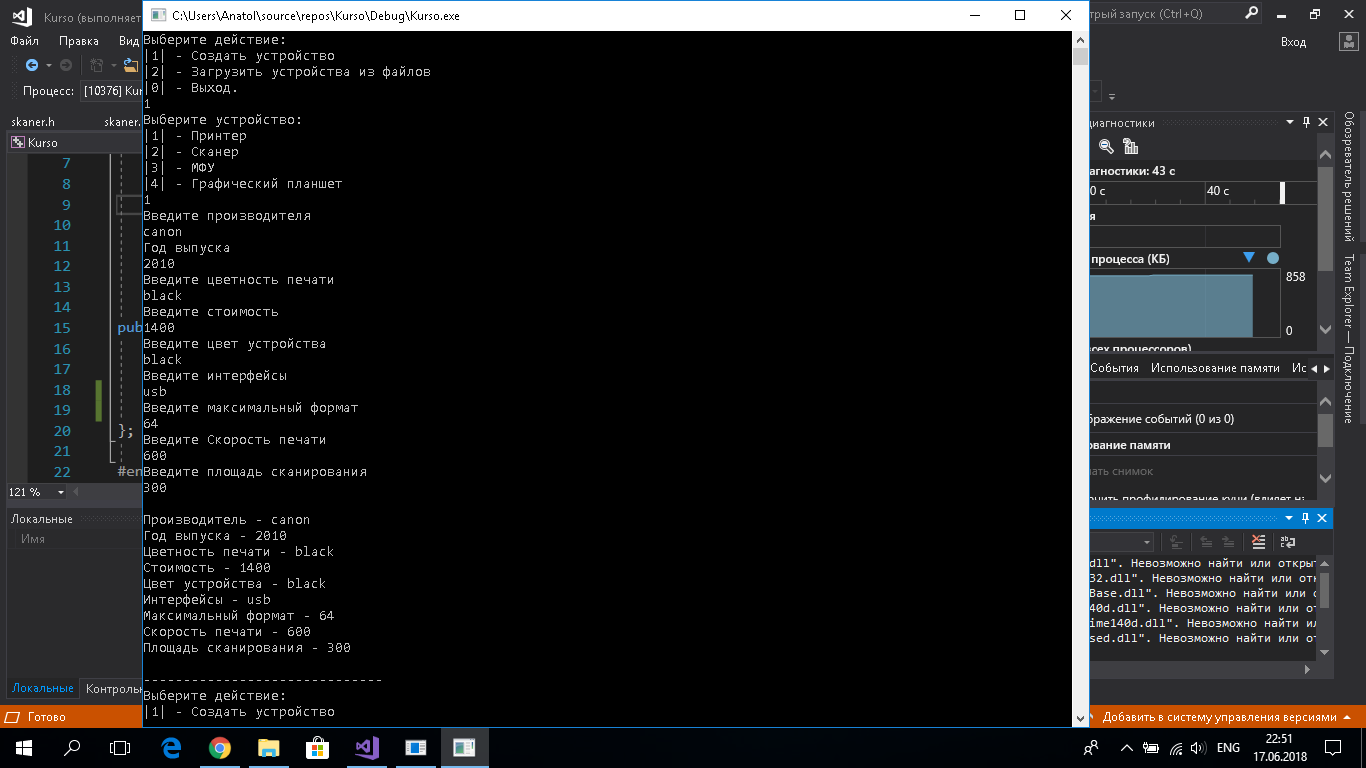


Рисунок 2

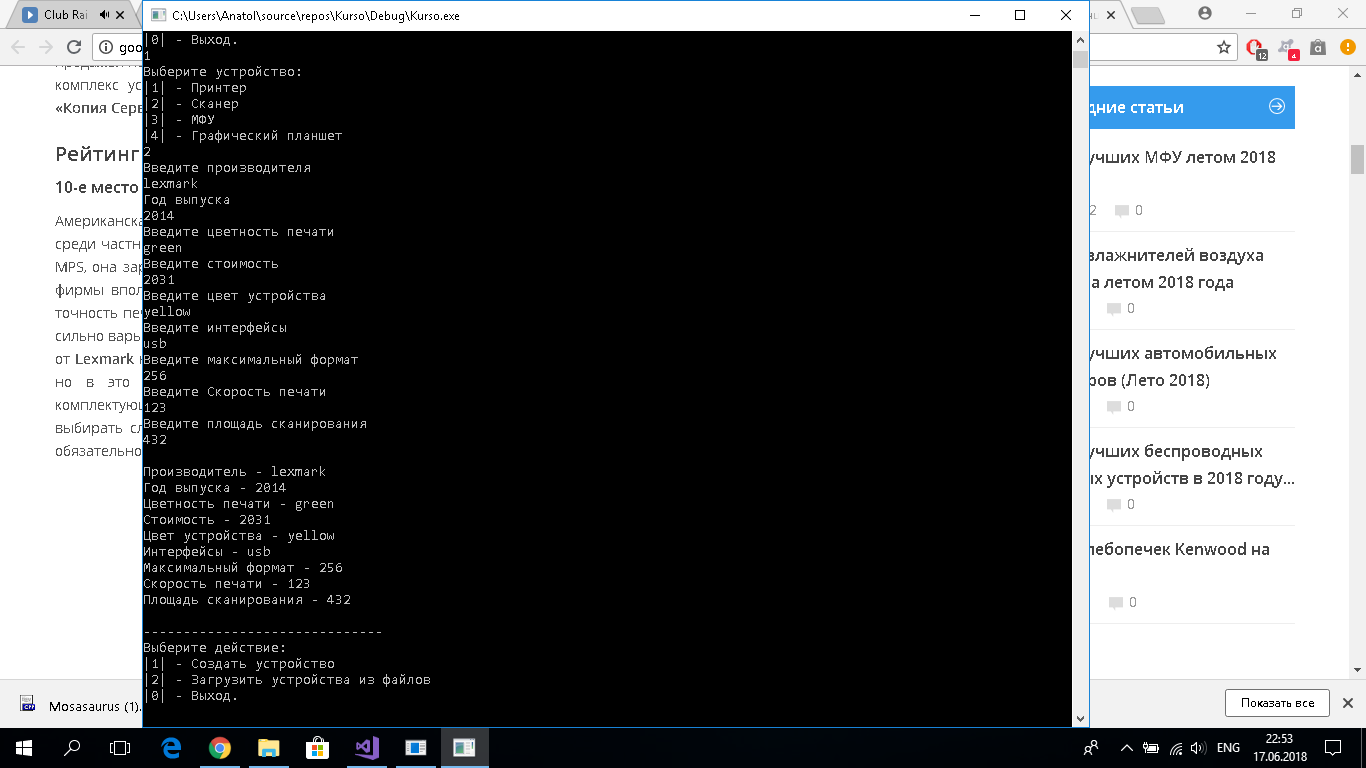


Рисунок 3

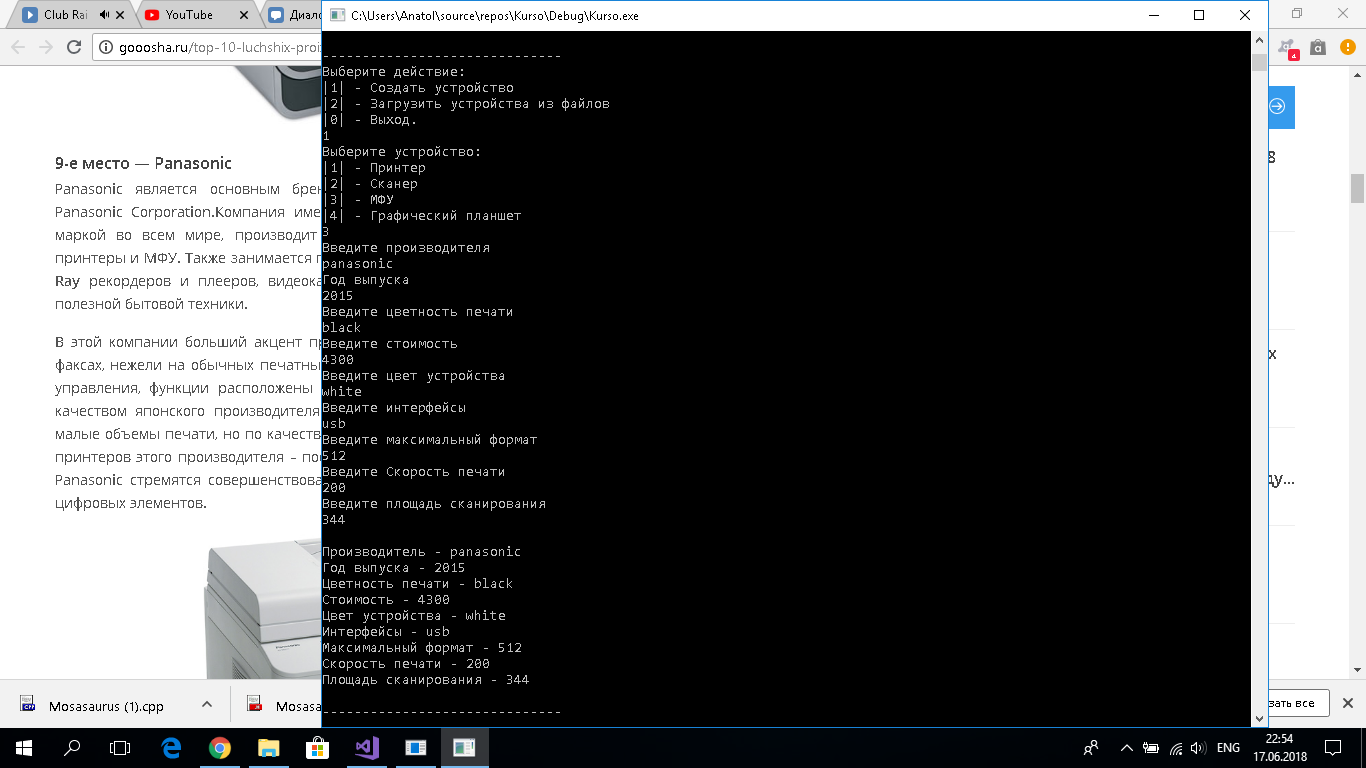


Рисунок 4

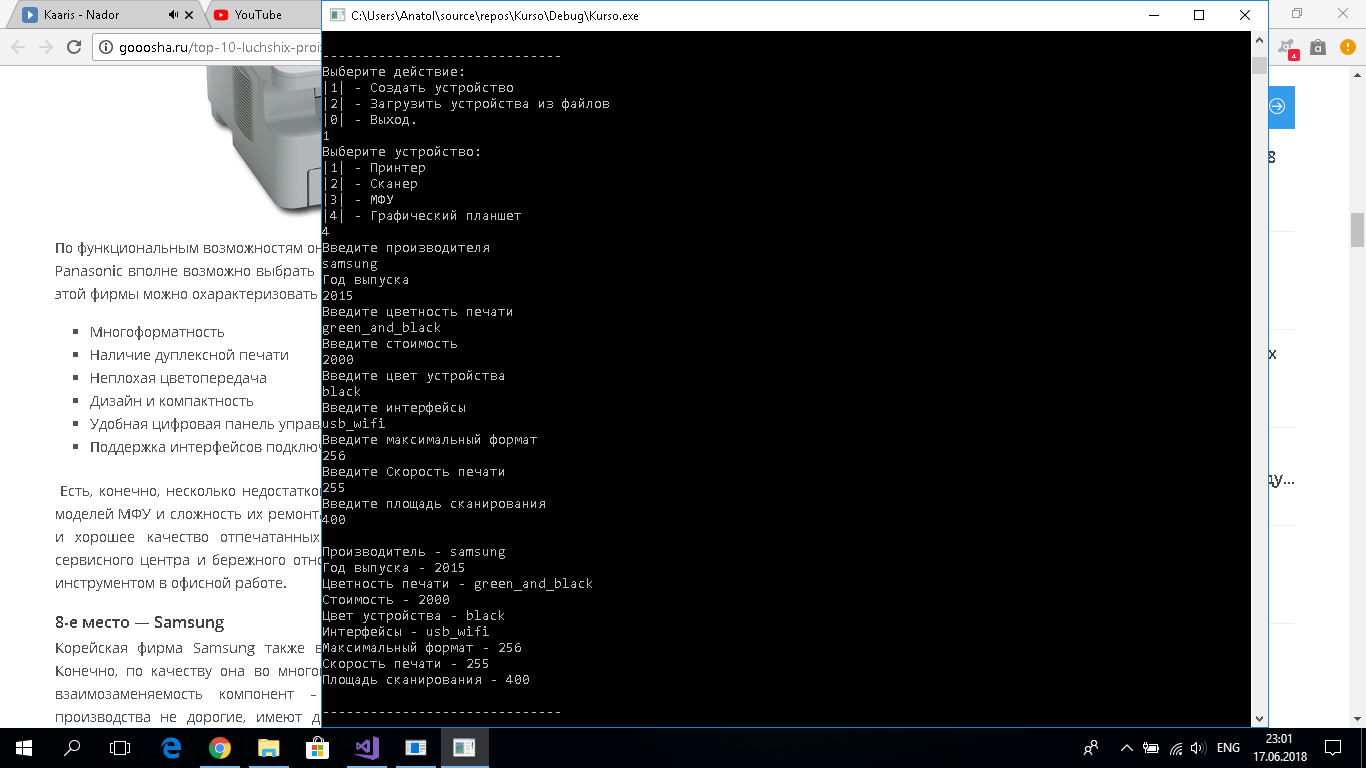


Рисунок 5

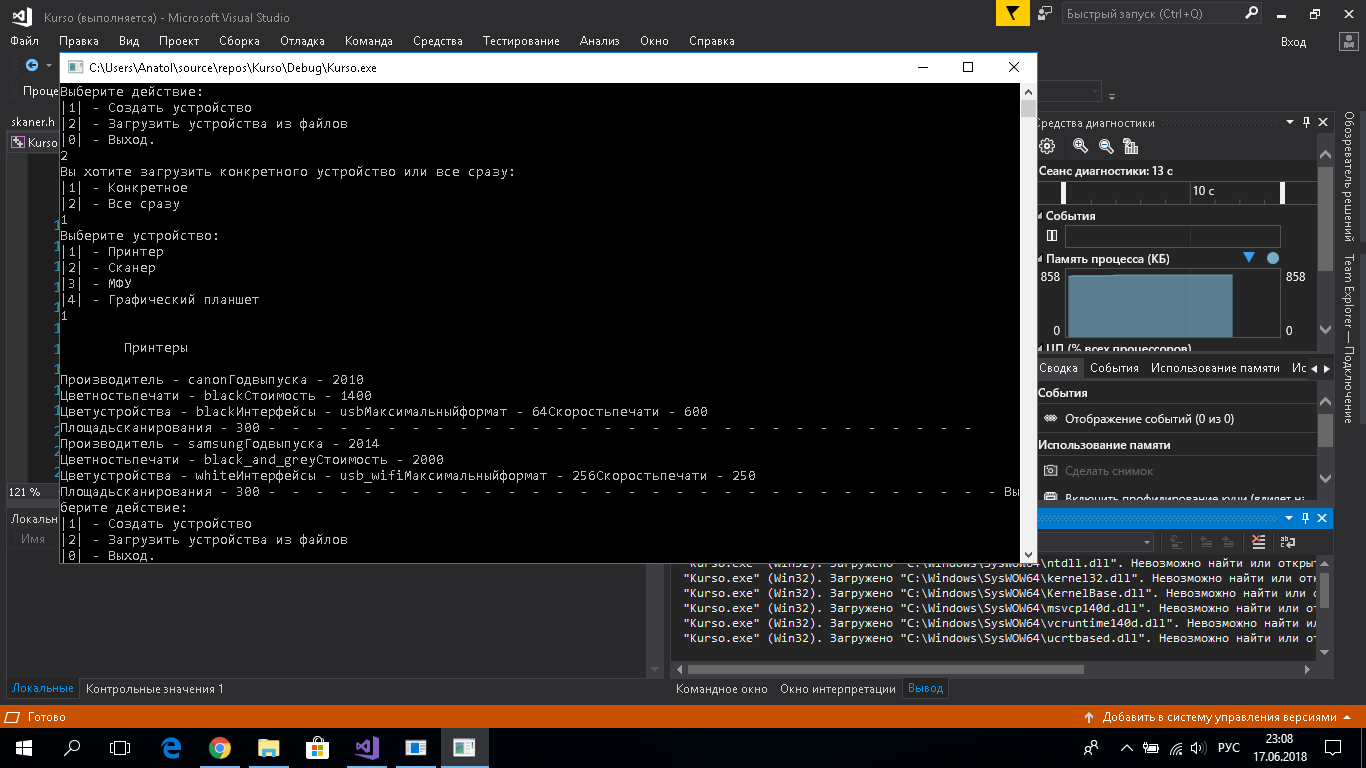


Рисунок 6

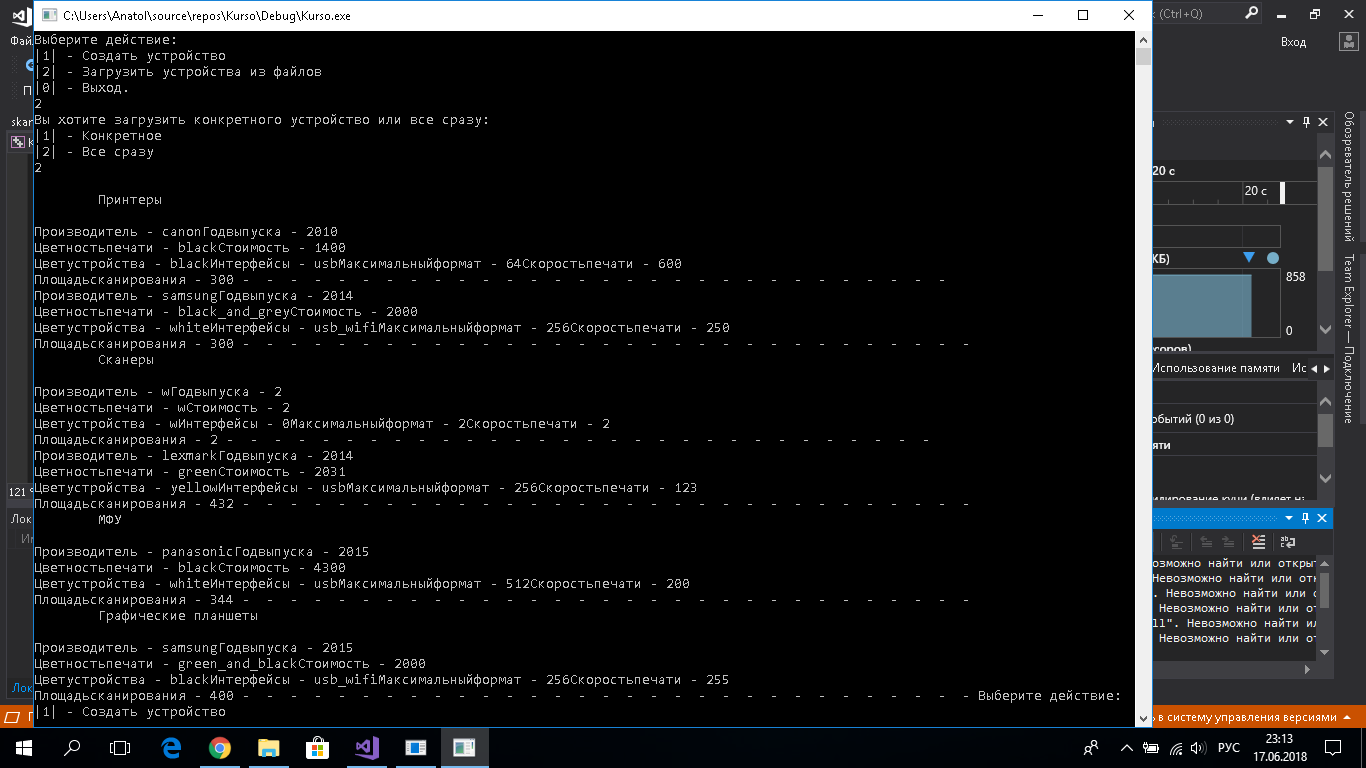


Рисунок 7

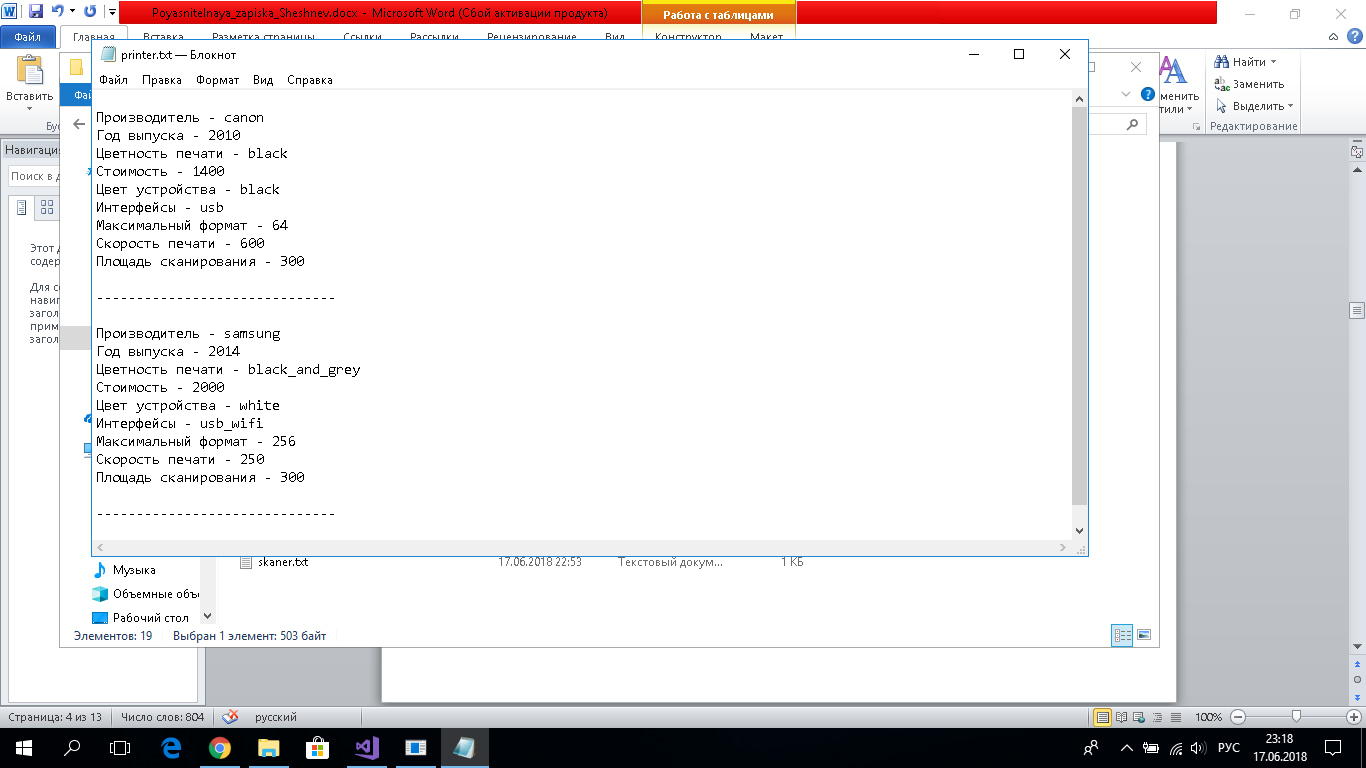
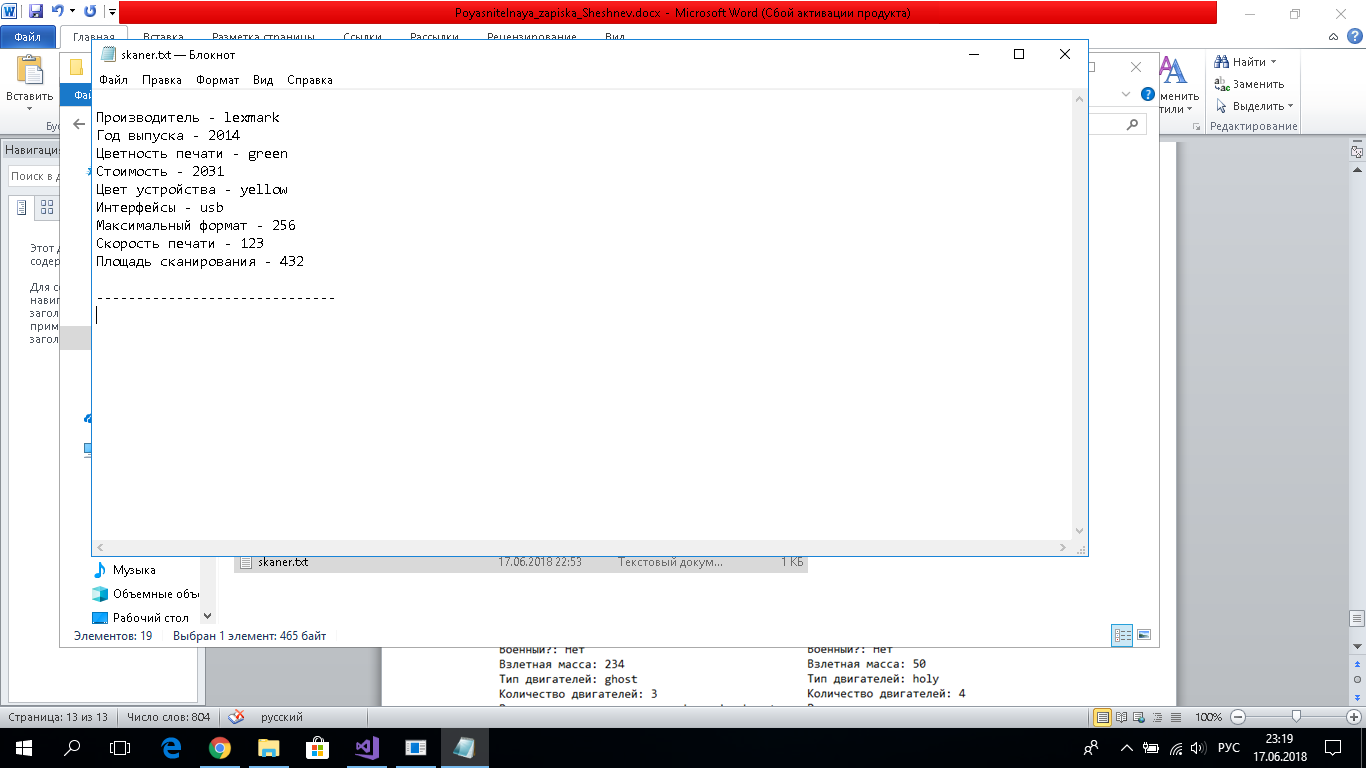
 

Рисунок 8 Рисунок 9

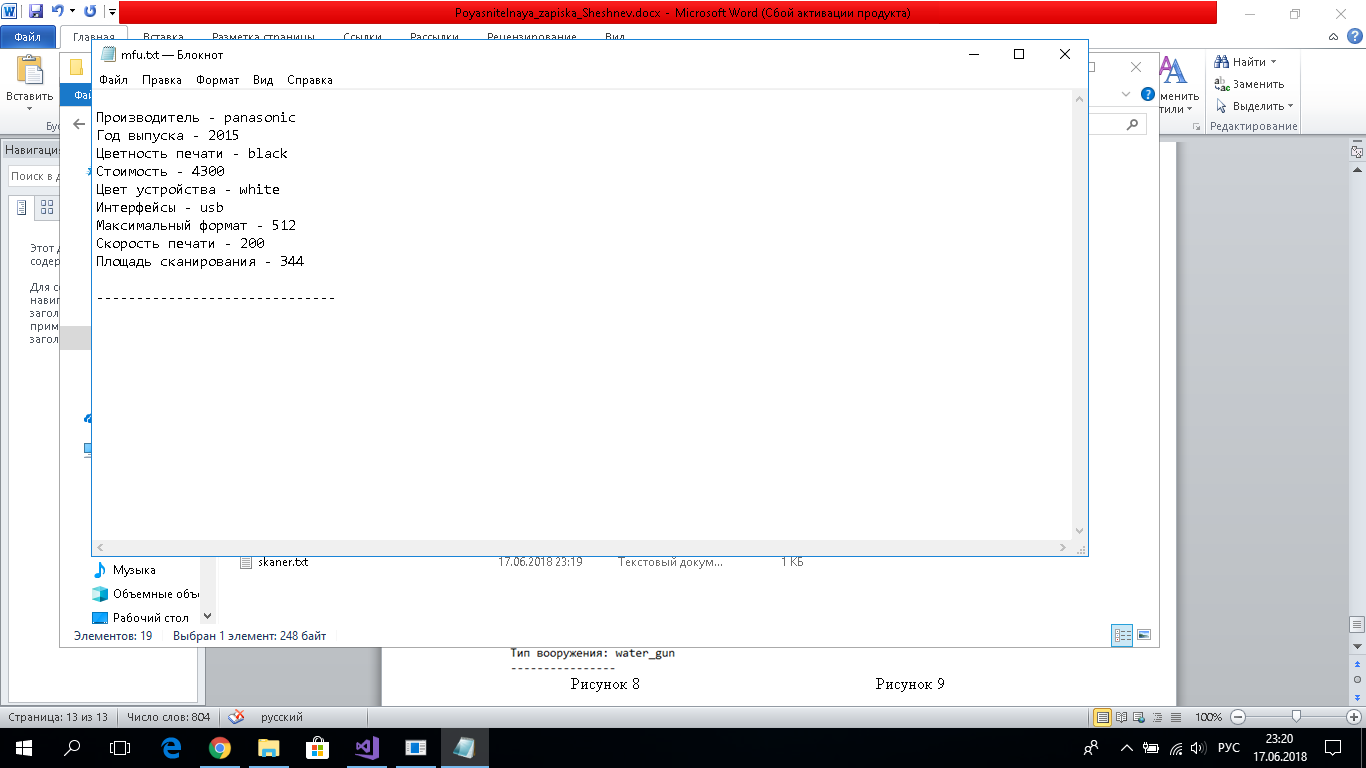
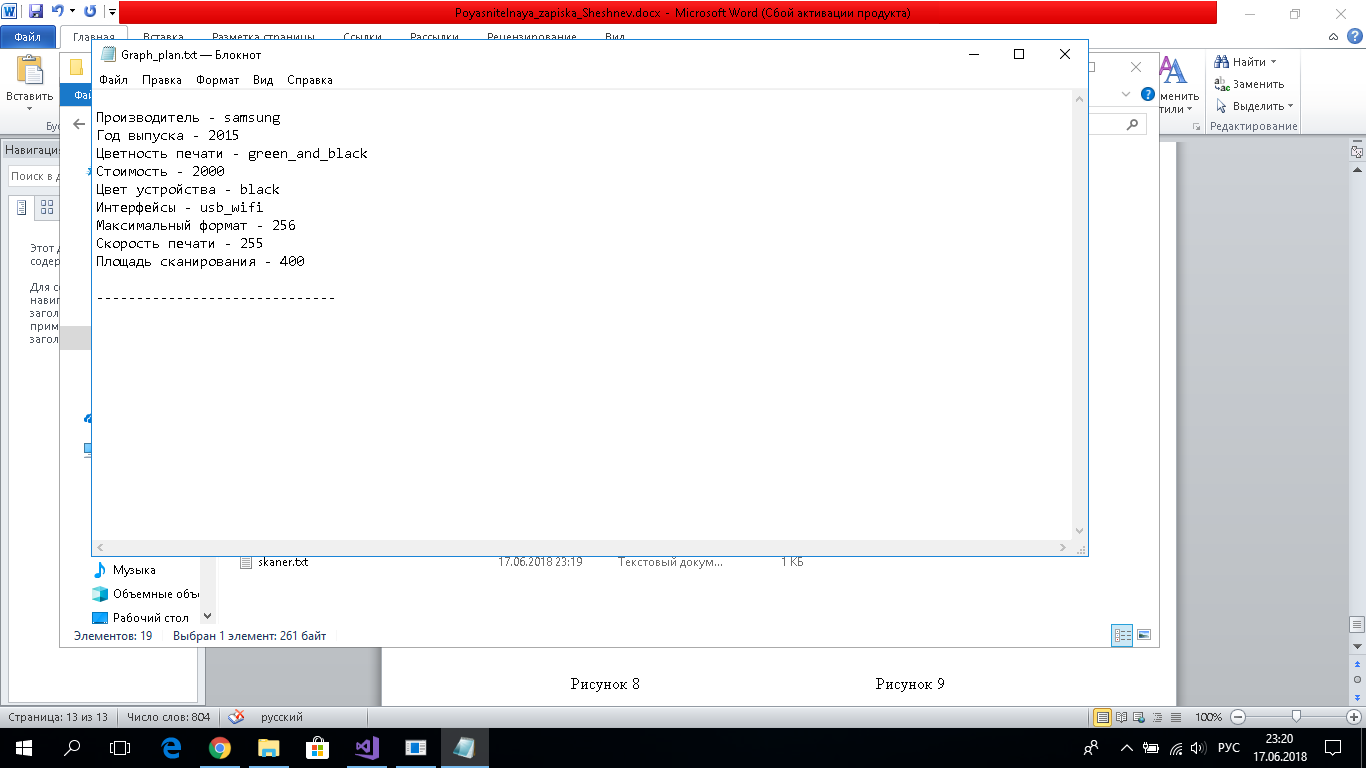
 

Рисунок 10 Рисунок 11