МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

«Изучение основ библиотек openCV и libusb»

студента 2 курса, группы 20203

Синюкова Валерия Константиновича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
доцент кафедры параллельных вычислений
Власенко Андрей Юрьевич

Новосибирск 2021

СОДЕРЖАНИЕ

ЦЕЛЬ

1) Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.

2) Ознакомиться с началами низкоуровневого программирования периферийных устройств на примере получения информации о доступных USB-устройствах с помощью библиотеки libusb.

ЗАДАНИЕ

1. Реализовать программу№1 с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.

2. Выполнить произвольное преобразование изображения (**кроме** указанных в computerlab5.pdf сглаживания и установки значений цветовых каналов в константу).

3. Измерить количество кадров, обрабатываемое программой в секунду.

Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.

4. Реализовать программу №2, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя, идентификатор изделия и серийный номер.

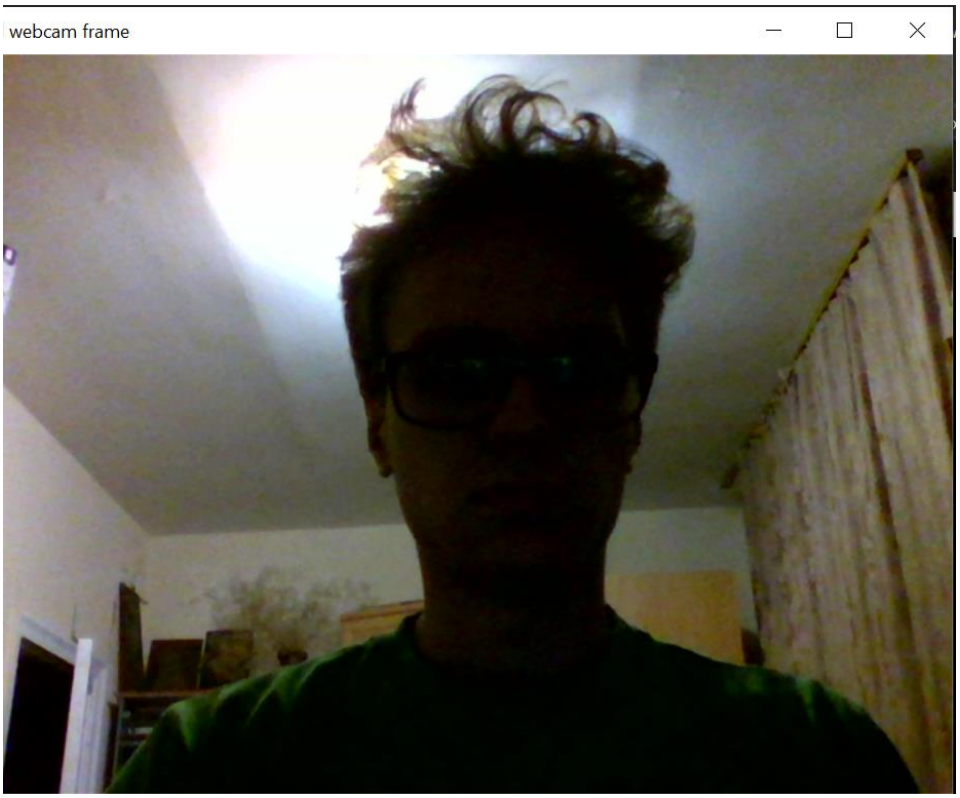
5. Составить отчет, который должен содержать:

- «чистое» неоткорректированное изображение, полученное с камеры;
- это же изображение в преобразованном виде;
- полный код программы №1, выполняющей преобразование изображения;
- оценку скорости обработки видео (кадров в секунду) и долю времени, затрачиваемого процессором на ввод, обработку и показ видеоданных;
- полный код программы №2, выводящей информацию по USB-устройствам;
- описание обнаруженных USB-устройств.

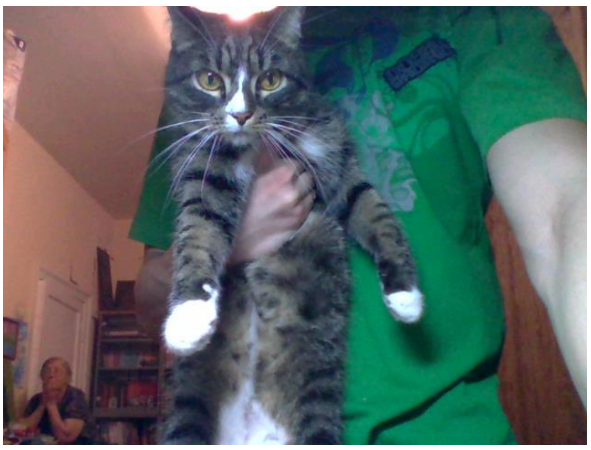
ОПИСАНИЕ РАБОТЫ

OpenCV

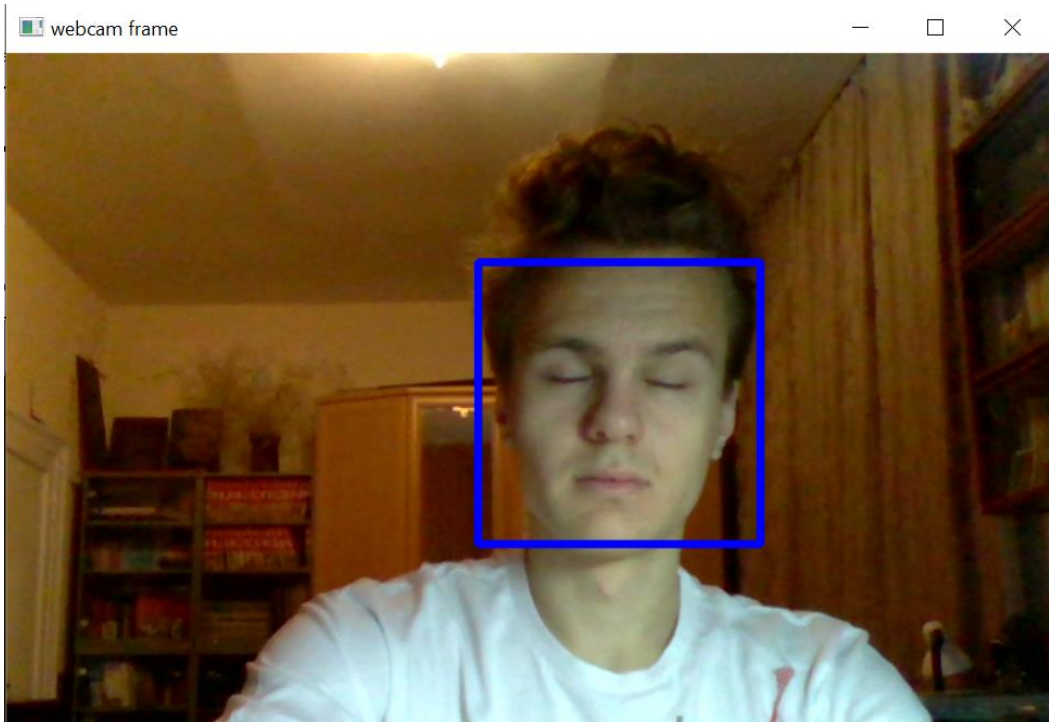
1) На языке C++ была реализована программа №1 с использованием openCV, которая получает поток видеоданных с камеры и выводит его на экран (см. код программы №1 в соответствующем приложении).



2) Было выполнено два преобразования изображений и видеоданных:
   1. Программа №2: кадр сглаживается, на нем выделяются края, выделенные края увеличиваются в 3 раза, левая и правая четверть изображения обрезаются (см. код программы №2 в соответствующем приложении).

Оригинальная фотография	Отредактированная фотография
	

2. Программа №3: программа, получающая поток видеоданных с камеры, определяет лицо человека и обводит его прямоугольником, после чего выводит на экран видео с выделенным лицом (см. код программы №3 в соответствующем приложении)



2) Было измерено количество кадров, обрабатываемое программой №1 в секунду. Так как в цикле тела `while(1)`, функция `waitKey(int delay = 0)` вызывается с параметром 33, то время, которое проходит перед получением следующего кадра равно 33 миллисекунды. Следовательно, количество кадров обрабатываемое программой в секунду равно $\frac{\mathbf{1000}\ \text{мс}}{\mathbf{33}\ \frac{\text{мс}}{\text{кадр}}} \approx 30$ кадров.

3) С помощью функции `clock()` было измерено время, затрачиваемое процессором на обработку видеоданных (конфигурация программы: Release в Visual Studio 2019).

	Программа №1 (с)	**Программа №2 (с)**	**Программа №3 (с)**
1)	0.014	0.159	0.15
2)	0.003	0.017	0.089
3)	0.124	0.009	0.07
4)	0.012	0.023	0.065
5)	0.028	0.009	0.061
6)	0.013	0.007	0.069
7)	0.012	0.009	0.068
8)	0.012	0.008	0.07

9)	0.014	0.008	0.064
10)	0.012	0.006	0.062
среднее арифметическое	0.024	0.025	0.078

Из данной таблицы мы можем сделать вывод, что на распознавание лиц тратится намного больше времени, чем на ввод, обработку и вывод видеоданных.

libusb

1) Программа, данная в тексте лабораторной работы №6 была модифицирована, одной из модификаций является добавление функции `void printSerialNumberAndProductDescription(libusb_device *dev, libusb_device_descriptor desc, FILE * out)`, выполняющая вывод в файл серийный номер и описание продукта. (*см. код модифицированной программы в соответствующем приложении*):

Оригинальная программа	Модифицированная программа
Выводит: 1) количество найденных устройств 2) количество возможных конфигураций *Для каждого устройства:* 1) класс устройства 2) идентификатор производителя 3) идентификатор устройства 4) количество интерфейсов 5) количество альтернативных настроек *Для каждого интерфейса:* 1) номер интерфейса 2) количество конечных точек *Для каждой конечной точки:* 1) тип дескриптора 2) адрес конечной точки	*Выводит:* то же самое, что и оригинальная программа *Для каждого устройства:* 1) Серийный номер устройства 2) Описание устройства

Содержимое файла, в которое записывается описание всех обнаруженных устройств:

```
==========================================================
======================

количество конфигураций: 01
класс устройства: 09
идентификатор производителя: 32903
идентификатор изделия: 32768
```

количество интерфейсов: 001
Ошибка: серийный номер устройства не получен, код: -2
Ошибка: описание продукта не получено, код: -2
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 09
идентификатор производителя: 7531
идентификатор изделия: 0002
количество интерфейсов: 001
Серийный номер USB устройства: 0000:00:1d.0
Продукт: EHCI Host Controller
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 09
идентификатор производителя: 32903
идентификатор изделия: 32776
количество интерфейсов: 001
Ошибка: серийный номер устройства не получен, код: -2
Ошибка: описание продукта не получено, код: -2
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 09
идентификатор производителя: 7531
идентификатор изделия: 0002
количество интерфейсов: 001
Серийный номер USB устройства: 0000:00:1a.0
Продукт: EHCI Host Controller
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 09
идентификатор производителя: 7531
идентификатор изделия: 0003
количество интерфейсов: 001
Серийный номер USB устройства: 0000:00:14.0
Продукт: xHCI Host Controller
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 00
идентификатор производителя: 1112
идентификатор изделия: 0058

количество интерфейсов: 001
Ошибка: серийный номер устройства не получен, код: -2
Продукт: USB Optical Mouse
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 00
идентификатор производителя: 7247
идентификатор изделия: 0038
количество интерфейсов: 002
Ошибка: серийный номер устройства не получен, код: -2
Продукт: USB Keyboard
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
2. количество различных установок интерфейса: 01
1) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000130

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 239
идентификатор производителя: 1133
идентификатор изделия: 2085
количество интерфейсов: 004
Серийный номер USB устройства: 95410D90
Ошибка: описание продукта не получено, код: -2
1. количество различных установок интерфейса: 01

1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000135
2. количество различных установок интерфейса: 12
1) номер интерфейса: 01
количество конечных точек: 00
2) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
3) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
4) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
5) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
6) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
7) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
8) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
9) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
10) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
11) номер интерфейса: 01
количество конечных точек: 01

1. тип дескриптора: 05
адрес конечной точки: 000000129
12) номер интерфейса: 01
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000129
3. количество различных установок интерфейса: 01
1) номер интерфейса: 02
количество конечных точек: 00
4. количество различных установок интерфейса: 05
1) номер интерфейса: 03
количество конечных точек: 00
2) номер интерфейса: 03
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000134
3) номер интерфейса: 03
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000134
4) номер интерфейса: 03
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000134
5) номер интерфейса: 03
количество конечных точек: 01
1. тип дескриптора: 05
адрес конечной точки: 000000134

==========
==========
==========
==========

количество конфигураций: 01
класс устройства: 09
идентификатор производителя: 7531
идентификатор изделия: 0002
количество интерфейсов: 001
Серийный номер USB устройства: 0000:00:14.0
Продукт: xHCI Host Controller
1. количество различных установок интерфейса: 01
1) номер интерфейса: 00
количество конечных точек: 01
1. тип дескриптора: 05

адрес конечной точки: 000000129

ЗАКЛЮЧЕНИЕ

В результате нашей работы были изучены основы библиотеки openCV, была реализована программа, получающая поток видеоданных с камеры и выводящая их на экран, программа, обрабатывающая изображения, и программа, обнаруживающая лица.
Также нами были изучены основы библиотеки libusb и реализована программа, получающая информацию о подключенных usb-устройствах и выводящая эту информацию на экран.

Приложение 1. *Код программы №1*

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <fstream>
#include <time.h>

using namespace cv;
using namespace std;

int main()
{
	clock_t t;
	VideoCapture cap(0);
	Mat frame;
	char number_of_time_fixes = 0;
	while (true)
	{
		t = clock();
		cap >> frame;
		imshow("webcam frame", frame);
		t = clock() - t;
		if (waitKey(16) == 27)
			break;
		if (number_of_time_fixes < 100)
		{
			cout << (int)(number_of_time_fixes) + 1 << ") " <<
(float)(t) / CLOCKS_PER_SEC << " sec " <<  endl;
			++number_of_time_fixes;
		}
	}
	destroyAllWindows();
	return 0;
}
```

Приложение 2. *Код программы №2*

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <fstream>
#include <time.h>

using namespace cv;
using namespace std;

int main()
{
	clock_t t;
	VideoCapture cap(0);
	Mat frame, frame_blur, frame_canny;
	Mat frame_dil, frame_dil_crop;
	Mat frame_gray;
	int frame_width, frame_height, number_of_time_fixes = 0;
	Mat kernel = getStructuringElement(MORPH_RECT, Size(3, 3));
	Point anchor(9, 9);
	while (true)
	{
```

```
            t = clock();
            cap >> frame;
            GaussianBlur(frame, frame_blur, anchor, 0);
            Canny(frame_blur, frame_canny, 60, 60);
            dilate(frame_canny, frame_dil, kernel);
            frame_width = frame.size().width;
            frame_height = frame.size().height;
            Rect roi(frame_width / 4, 0, frame_width / 2, frame_height);
            frame_dil_crop = frame_dil(roi);
            imshow("webcam frame", frame);
            imshow("dilated croped frame", frame_dil_crop);
            t = clock() - t;
            if (waitKey(20) == 27)
                  break;
            if (number_of_time_fixes < 100)
            {
                  cout << (int)(number_of_time_fixes)+1 << ") " << (float)(t)
/ CLOCKS_PER_SEC << " sec " << endl;
                  ++number_of_time_fixes;
            }

      }
      imwrite("original.png", frame);
      imwrite("edited1.png", frame_dil_crop);
      destroyAllWindows();
      return 0;
}
```

Приложение 3. *Код программы №3*

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <fstream>
#include <time.h>

using namespace cv;
using namespace std;

int main()
{
      clock_t t;
      VideoCapture cap(0);
      Mat frame;
      int i, number_of_faces;
      vector<Rect> faces;
      CascadeClassifier face_cascade;
      face_cascade.load("xml_files/haarcascade_frontalface_default.xml");
      char number_of_time_fixes = 0;
      if (face_cascade.empty())
            return 1;
      while (true)
      {
            t = clock();
            cap >> frame;
            face_cascade.detectMultiScale(frame, faces, 1.1, 10);
            number_of_faces = faces.size();
```

```
            for (i = 0; i < number_of_faces; ++i)

      rectangle(frame,faces[i].tl(),faces[i].br(),Scalar(255,0,0),3);
            imshow("webcam frame", frame);
            t = clock() - t;
            if (waitKey(33) == 27)
                  break;
            if (number_of_time_fixes < 100)
            {
                  cout << (int)(number_of_time_fixes)+1 << ") " << (float)(t)
/ CLOCKS_PER_SEC << " sec " << endl;
                  ++number_of_time_fixes;
            }

      }
      destroyAllWindows();
      return 0;
    }
```

Приложение 4. *Код программы, выводящей обнаруженные usb-устройства*

```
    #include <iostream>
    #include <libusb.h>
    #include <stdio.h>

    using namespace std;

    void printSerialNumberAndProductDescription(libusb_device *dev,
libusb_device_descriptor desc, FILE * out);
    void printdev(libusb_device *dev, FILE * out);

    int main()
    {
        FILE * out = fopen("devices_description.txt","w");
        if (!out)
            return 1;
        libusb_device **devs; // указатель на указатель на устройство,
                              // используется для получения списка устройств
        libusb_context *ctx = NULL; // контекст сессии libusb
        int r; // для возвращаемых значений
        ssize_t cnt; // число найденных USB-устройств
        ssize_t i; // индексная переменная цикла перебора всех устройств
        // инициализировать библиотеку libusb, открыть сессию работы с
libusb
        r = libusb_init(&ctx);
        if(r < 0)
        {
            fprintf(out,"Ошибка: инициализация не выполнена, код: %d.\n",
r);
            return 1;
        }
        // задать уровень подробности отладочных сообщений
        libusb_set_debug(ctx, 3);
        // получить список всех найденных USB- устройств
        cnt = libusb_get_device_list(ctx, &devs);
```

```
        if(cnt < 0)
        {
            fprintf(out, "Ошибка: список USB устройств не получен, код:
%d\n", r);
            return 1;
        }
        for(i = 0; i < cnt; i++)
        { // цикл перебора всех устройств
            printdev(devs[i],out); // печать параметров устройства
        }
        // освободить память, выделенную функцией получения списка устройств
        libusb_free_device_list(devs, 1);
        libusb_exit(ctx); // завершить работу с библиотекой libusb,
        // закрыть сессию работы с libusb
        return 0;
    }


    void printSerialNumberAndProductDescription(libusb_device *dev,
libusb_device_descriptor desc, FILE * out)
    {
        int r;
        unsigned char * serialNumber = new unsigned char [200];
        libusb_device_handle * devh;
        r = libusb_open(dev,&devh);
        if (r)
            fprintf(out,"\tОшибка: не удалось получить device_handle\n");
        r =
libusb_get_string_descriptor_ascii(devh,desc.iSerialNumber,serialNumber,200);
        if (r < 0)
            fprintf(out, "\tОшибка: серийный номер устройства не получен,
код: %d\n",r);
        else
            fprintf(out,"\tСерийный номер USB устройства: %s\n",
serialNumber);
        delete(serialNumber);
        unsigned char * product = new unsigned char [200];
        r =
libusb_get_string_descriptor_ascii(devh,desc.iProduct,product,200);
        if (r < 0)
            fprintf(out, "\tОшибка: описание продукта не получено, код:
%d\n",r);
        else
            fprintf(out,"\tПродукт: %s\n", product);
        delete(product);
        libusb_close(devh);
    }

    void printdev(libusb_device *dev, FILE * out)
    {
        libusb_device_descriptor desc; // дескриптор устройства
        libusb_config_descriptor *config; // дескриптор конфигурации объекта
        const libusb_interface *inter; // набор различных установок для
определенного USB интерфейса
        const libusb_interface_descriptor *interdesc; // дескриптор
интерфейса
```

```
        const libusb_endpoint_descriptor *epdesc;
        int i,j,k,r = libusb_get_device_descriptor(dev, &desc);

fprintf(out,"================================================================
==============\n\n");
        if (r < 0)
        {
            fprintf(out, "\tОшибка: дескриптор устройства не получен, код:
%d\n",r);
            return;
        }
        // получить конфигурацию устройства
        libusb_get_config_descriptor(dev, 0, &config);
        fprintf(out,"\tколичество конфигураций: %.2d\n\tкласс устройства:
%.2d\n\tидентификатор производителя: %.4d\n\tидентификатор изделия:
%.4d\n\tколичество интерфейсов: %.3d\n",
        (int)desc.bNumConfigurations,
        (int)desc.bDeviceClass,
        desc.idVendor,
        desc.idProduct,
        (int)config->bNumInterfaces
        );
        printSerialNumberAndProductDescription(dev,desc,out);
        for(i=0; i<(int)config->bNumInterfaces; i++) //цикл перебора всех
интерфейсов, поддерживаемых данной конфигурацией
        {
            inter = &config->interface[i];
            fprintf(out,"\t%d. количество различных установок интерфейса:
%.2d\n",i + 1, inter->num_altsetting);
            for(j=0; j<inter->num_altsetting; j++)
            {
                interdesc = &inter->altsetting[j];
                fprintf(out,"\t\t%d) номер интерфейса: %.2d\n\t\tколичество
""конечных точек"": %.2d\n", j + 1,
                (int)interdesc->bInterfaceNumber,
                (int)interdesc->bNumEndpoints
            );
                for(k=0; k<(int)interdesc->bNumEndpoints; k++)
                {
                    epdesc = &interdesc->endpoint[k];
                    fprintf(out,"\t\t\t%d. тип дескриптора: %.2d\n\t\t\tадрес
""конечной точки"": %.9d\n",k + 1,
                    (int)epdesc->bDescriptorType,
                    (int)epdesc->bEndpointAddress
                    );
                }
            }
        }

fprintf(out,"\n================================================================
================\n");
        libusb_free_config_descriptor(config);
    }
```