###### 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 ЛАБОРАТОРНОЙ РАБОТЫ

«Низкоуровневая работа с периферийными устройствами»

студента 2 курса, 18205 группы

Гайдамака Андрея Владиславовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А. Ю. Власенко

Новосибирск 2019

ЦЕЛЬ 3

ЗАДАНИЕ 3

ОПИСАНИЕ РАБОТЫ 3

ЗАКЛЮЧЕНИЕ 4

Приложение 1. Код программы 5

# ЦЕЛЬ

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

# ЗАДАНИЕ

1. Реализовать программу, получающую список всех подключенных к машине USB устройств с использованием libusb. Для каждого найденного устройства напечатать его класс, идентификатор производителя и идентификатор изделия. За основу для разработки можно взять программу, приведенную в листинге 1.
2. Изучить состав и характеристики обнаруженных с помощью реализованной программ USB устройств.
3. Дополнить программу, реализованную в п. 2 функцией печати серийного номера USB устройства. Для написания функции рекомендуется использовать функции libusb\_open, libusb\_close, libusb\_get\_string\_descriptor\_ascii для печати поля iSerialNumber дескриптора устройства.

# ОПИСАНИЕ РАБОТЫ

В ходе лабораторной работы был получен список устройств подключенных по USB, в этом списке было единственное устройство – это хаб к которому подключаются конечные точки (клавиатура, трекпад итд), у каждой точки есть свой интерфейс и дескриптор (описание устройства), который содержит информацию о серийный номер, id производителя, id устройства. У каждой конечной точки есть свой адрес по которому хост может с ними общаться.

Строка компиляции: g++ -o executable main.cpp -lusb-1.0

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была изучена работа с устройствами usb и принцип работы интерфейса usb. Так же была изучена иерархия usb устройств и типы и структуры запросов в интерфейсе usb.

**Приложение 1. Код программы**

**#include <iostream>**

**#include "libusb.h"**

**using namespace std;**

**void printdev(libusb\_device \*dev);**

**int main() {**

**libusb\_device \*\*devs;**

**libusb\_context \*ctx = NULL;**

**int r;**

**ssize\_t cnt;**

**r = libusb\_init(&ctx);**

**if(r < 0) {**

**cout<<"Init Error "<<r<<endl;**

**return 1;**

**}**

**libusb\_set\_debug(ctx, 3);**

**cnt = libusb\_get\_device\_list(ctx, &devs);**

**if(cnt < 0) {**

**cout<<"Get Device Error"<<endl;**

**}**

**cout<<cnt<<" Devices in list."<<endl;**

**ssize\_t i;**

**for(i = 0; i < cnt; i++) {**

**printdev(devs[i]);**

**}**

**libusb\_free\_device\_list(devs, 1);**

**libusb\_exit(ctx);**

**return 0;**

**}**

**void printdev(libusb\_device \*dev) {**

**libusb\_device\_descriptor desc;**

**int r = libusb\_get\_device\_descriptor(dev, &desc);**

**if (r < 0) {**

**cout<<"failed to get device descriptor"<<endl;**

**return;**

**}**

**cout<<"Number of possible configurations: "<<(int)desc.bNumConfigurations<<" ";**

**cout<<"Device Class: "<<(int)desc.bDeviceClass<<" ";**

**cout<<"VendorID: "<<desc.idVendor<<" ";**

**cout<<"ProductID: "<<desc.idProduct<<endl;**

**libusb\_config\_descriptor \*config;**

**libusb\_get\_config\_descriptor(dev, 0, &config);**

**cout<<"Interfaces: "<<(int)config->bNumInterfaces<<" ||| ";**

**const libusb\_interface \*inter;**

**const libusb\_interface\_descriptor \*interdesc;**

**const libusb\_endpoint\_descriptor \*epdesc;**

**for(int i=0; i<(int)config->bNumInterfaces; i++) {**

**inter = &config->interface[i];**

**cout<<"Number of alternate settings: "<<inter->num\_altsetting<<" | ";**

**for(int j=0; j<inter->num\_altsetting; j++) {**

**interdesc = &inter->altsetting[j];**

**cout<<"Interface Number: "<<(int)interdesc->bInterfaceNumber<<" | ";**

**cout<<"Number of endpoints: "<<(int)interdesc->bNumEndpoints<<" | ";**

**for(int k=0; k<(int)interdesc->bNumEndpoints; k++) {**

**epdesc = &interdesc->endpoint[k];**

**cout<<"Descriptor Type: "<<(int)epdesc->bDescriptorType<<" | ";**

**cout<<"EP Address: "<<(int)epdesc->bEndpointAddress<<" | ";**

**}**

**}**

**}**

**libusb\_device\_handle \*dh;**

**unsigned char data[1000];**

**r = libusb\_open(dev, &dh);**

**if (r){**

**cout << "Cant open device" << endl;**

**}**

**r = libusb\_get\_string\_descriptor\_ascii(dh, desc.iSerialNumber , data, 1000);**

**cout << endl << endl << "Serial Number: " << data << endl;**

**libusb\_close(dh);**

**cout<<endl<<endl<<endl;**

**libusb\_free\_config\_descriptor(config);**

**}**