Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

Кафедра Управления и интеллектуальных технологий

**Отчёт по лабораторной работе № 1**

**По курсу «Разработка программного обеспечения систем управления»**

# «Библиотеки. Низкоуровневые операции»

Выполнил студент группы А-03-20

Львов М. Д.

Проверил

Мохов А. С

Москва 2021

Цели работы:

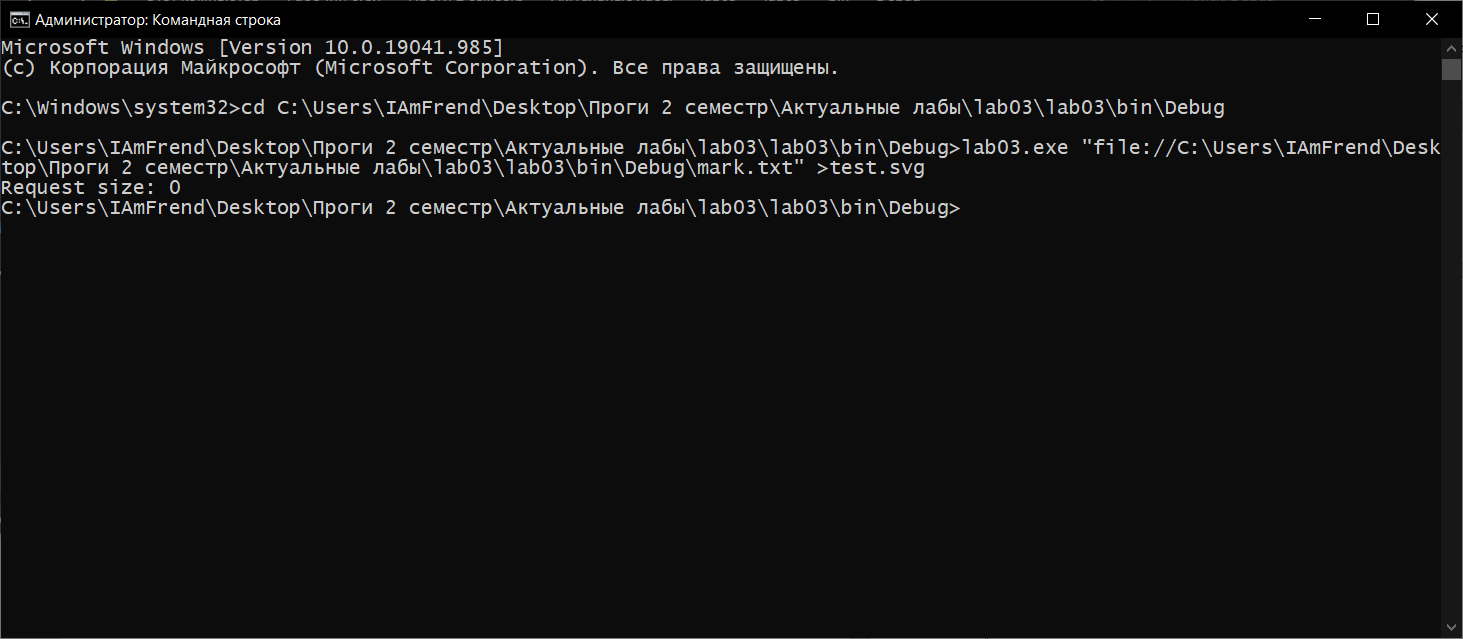
1. Уметь устанавливать и подключать к программе внешние библиотеки.
2. Уметь использовать типовые элементы API: функции обратного вызова, битовые флаги и маски, массивы и строки C.
3. Уметь работать с параметрами командной строки программы.
4. Уметь применять побитовые операции для типовых сценариев.
5. Уметь работать с API, принимающими указатели, в том числе строки C.
6. Знать характерные особенности документации на API библиотек.

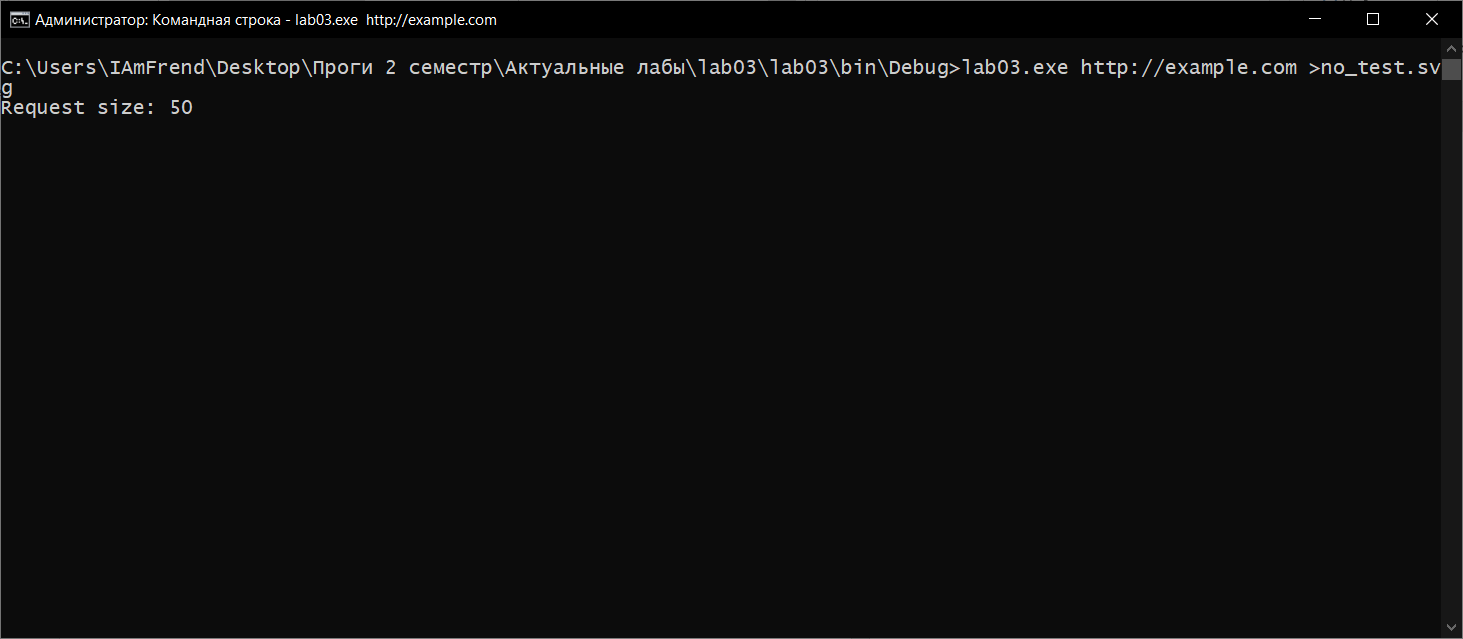
Задачи работы:

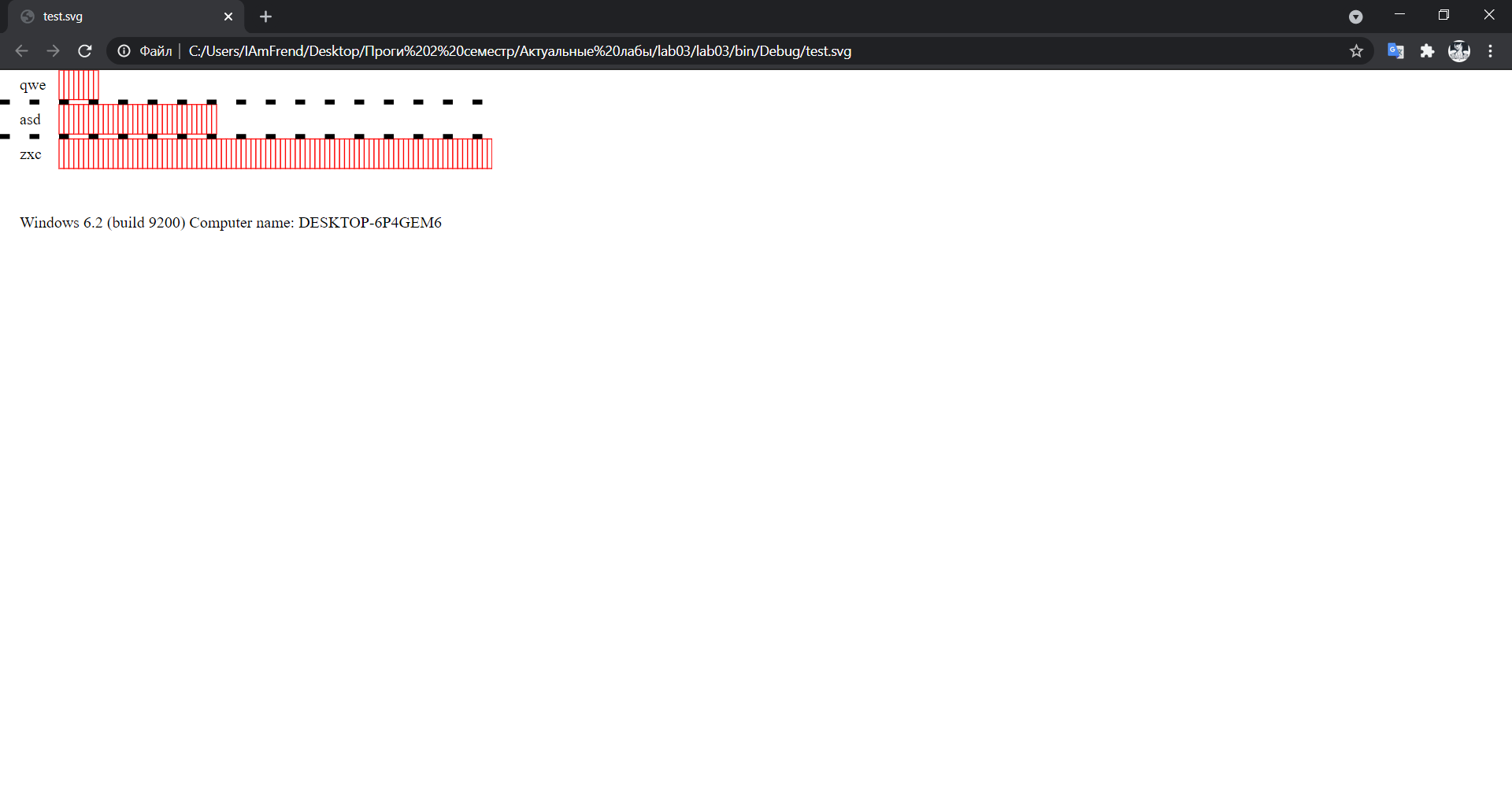
1. Добавить возможность построения гистограммы по данным из файла из сети. Адрес файла задается аргументом командной строки программы. Если адрес не задан, читать данные со стандартного ввода, как раньше.

Описание решения варианта:

Для вычисления верного параметра использовались данные документации: <https://curl.se/libcurl/c/curl_easy_getinfo.html> и <https://curl.se/libcurl/c/CURLINFO_REQUEST_SIZE.html>. На основании этих данных была составлена функция для вывода.







Код программы:

* Модуль main.cpp:

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include <sstream>

#include <string>

#include "histogram.h"

#include <curl/curl.h>

using namespace std;

vector<double> input\_numbers(istream& in, size\_t count,bool prompt)

{

vector<double> result(count);

for (size\_t i = 0; i < count; i++) {

if (prompt) cerr << "Enter number " << i + 1 << ": ";

in >> result[i];

}

return result;

}

Input read\_input(istream& in, bool prompt)

{

Input data;

if (prompt) cerr << "Enter number count: ";

in >> data.number\_count;

data.numbers = input\_numbers(in, data.number\_count,prompt);

if (prompt) cerr << "Enter bin count: ";

in >> data.bin\_count;

string str;

in.get();

in.get();

for (size\_t i = 0; i < data.bin\_count; i++)

{

if (prompt) cerr << "Enter bin " << i + 1 << " naming: ";

getline(in,str);

data.bin\_namings.push\_back(str);

}

if (prompt) cerr << "Enter separator dash: ";

in >> data.dash;

if (prompt) cerr << "Enter separator dasharray: ";

in >> data.dasharray;

return data;

}

size\_t write\_data(void\* items, size\_t item\_size, size\_t item\_count, void\* ctx)

{

size\_t data\_size = item\_size \* item\_count;

stringstream\* buffer = reinterpret\_cast<stringstream\*>(ctx);

char\* item\_list = reinterpret\_cast<char\*>(items);

buffer->write(item\_list, data\_size);

return data\_size;

}

Input download(const string& address)

{

stringstream buffer;

CURL \*curl = curl\_easy\_init();

if(curl)

{

CURLcode res;

curl\_easy\_setopt(curl, CURLOPT\_SSL\_VERIFYPEER, FALSE);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_data);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, &buffer);

curl\_easy\_setopt(curl, CURLOPT\_URL, address.c\_str());

res = curl\_easy\_perform(curl);

if (res != 0)cout<<curl\_easy\_strerror(res);

long req;

curl\_easy\_getinfo(curl, CURLINFO\_REQUEST\_SIZE, &req);

cerr<<"Request size: "<< req;

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

vector<size\_t> make\_histogram(Input data)

{

vector<size\_t> bins(data.bin\_count);

double min = data.numbers[0];

double max = data.numbers[0];

find\_minmax(data.numbers, min, max);

double bin\_size = (max - min) / data.bin\_count;

for (size\_t i = 0; i < data.number\_count; i++)

{

bool found = false;

for (size\_t j = 0; (j < data.bin\_count - 1) && !found; j++)

{

auto lo = min + j \* bin\_size;

auto hi = min + (j + 1) \* bin\_size;

if ((lo <= data.numbers[i]) && (data.numbers[i] < hi))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[data.bin\_count - 1]++;

}

}

return bins;

}

void show\_histogramm\_text(vector<size\_t> bins, size\_t bin\_count, vector<vector<char>> bin\_naming\_list)

{

vector<char[80]> bin\_namings(bin\_count);

size\_t iter = 0;

for (size\_t iter = 0; iter < bin\_count; iter++)

{

for (char l : bin\_naming\_list[iter])

{

bin\_namings[iter][strlen(bin\_namings[iter])] = l;

}

}

double max\_len = strlen(bin\_namings[0]);

for (size\_t i = 0; i < bin\_count; i++)

{

if (strlen(bin\_namings[i]) > max\_len)

{

max\_len = strlen(bin\_namings[i]);

}

}

double max\_count = bins[0];

const size\_t SCREEN\_WIDTH = 80;

size\_t MAX\_ASTERISK = SCREEN\_WIDTH - max\_len - 1;

for (double x : bins)

{

if (x > max\_count)

{

max\_count = x;

}

}

double height = 1;

if (max\_count > MAX\_ASTERISK)

{

height = static\_cast<double>(MAX\_ASTERISK) / static\_cast<double>(max\_count);

}

for (size\_t i = 0; i < bin\_count; i++)

{

size\_t tab = max\_len - strlen(bin\_namings[i]);

for (size\_t j = 0; j < tab; j++)

{

cout << " ";

}

cout << bin\_namings[i] << "|";

for (size\_t j = 0; j < (height \* bins[i]); j++)

{

cout << "\*";

}

cout << endl;

}

}

int main(int argc, char\* argv[])

{

curl\_global\_init(CURL\_GLOBAL\_ALL);

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Input data;

if (argc>1)

{

data = download(argv[1]);

}

else

{

data = read\_input(cin,true);

}

const auto bins = make\_histogram(data);

show\_histogram\_svg(bins, data.bin\_count, data.bin\_namings,data.dash,data.dasharray);

return 0;

}

* Модуль histogram.cpp

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include "histogram.h"

using namespace std;

void find\_minmax(vector<double> numbers, double& min, double& max)

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

{

min = x;

}

else if (x > max)

{

max = x;

}

}

}

* Модуль histogram.h

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include "histogram.h"

using namespace std;

void find\_minmax(vector<double> numbers, double& min, double& max)

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

{

min = x;

}

else if (x > max)

{

max = x;

}

}

}

* Модуль svg.cpp

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include <sstream>

#include "histogram.h"

using namespace std;

void svg\_begin(double width, double height)

{

cout << "<?xml version='1.0' encoding='UTF-8'?>\n";

cout << "<svg ";

cout << "width='" << width << "' ";

cout << "height='" << height << "' ";

cout << "viewBox='0 0 " << width << " " << height << "' ";

cout << "xmlns='http://www.w3.org/2000/svg'>\n";

}

void svg\_end()

{

cout << "</svg>\n";

}

void svg\_text(double left, double baseline, string text)

{

cout << "<text x='"<<left<<"' y='"<<baseline<<"'>"<<text<<"</text>"<<endl;

}

void svg\_rect(double x, double y, double width, double height, string stroke, string fill)

{

cout << "<rect x='"<<x<<"' y='"<<y<<"' width='"<<width<<"' height='"<<height<<"' stroke='"<<stroke<<"' fill='"<<fill<<"' />"<<endl;

}

string svg\_line(double x1, float y1, double x2, float y2, string stroke, float stroke\_width, double stroke\_dash, double stroke\_dasharray)

{

return "<line x1='"+to\_string(x1)+"' y1='"+to\_string(y1)+"' x2='"+to\_string(x2)+"' y2='"+to\_string(y2)+

"' stroke='"+stroke+"' stroke-width='"+to\_string(stroke\_width)

+"' stroke-dasharray = '"+to\_string(stroke\_dash)+" "+to\_string(stroke\_dasharray)+"'/>";

}

string make\_info\_text()

{

stringstream buffer;

auto info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD maj\_mask = 0x000000ff;

DWORD major = info & maj\_mask;

DWORD minor = (info >> 8) & maj\_mask;

if ((info & 0x8000000) == 0)

{

DWORD build = platform;

buffer<<"Windows "<<major<<"."<<minor<<" (build "<<build<<")"<<endl;

}

TCHAR infoBuf[MAX\_COMPUTERNAME\_LENGTH];

DWORD bufCharCount = MAX\_COMPUTERNAME\_LENGTH + 1;

GetComputerName(infoBuf,&bufCharCount);

buffer<<"Computer name: "<<infoBuf;

return buffer.str();

}

void show\_histogram\_svg(vector<size\_t> bins, size\_t bin\_count, vector<string> bin\_namings,double dash, double dasharray)

{

const auto IMAGE\_WIDTH = 500;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 10;

const auto BIN\_HEIGHT = 30;

const auto LINE\_HEIGHT = 5.0;

double BLOCK\_WIDTH = 10;

svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);

double max\_len = bin\_namings[0].length();

for (size\_t i = 0; i < bin\_count; i++)

{

if (bin\_namings[i].length() > max\_len)

{

max\_len = bin\_namings[i].length();

}

}

double max\_count = bins[0];

for (double x : bins)

{

if (x > max\_count)

{

max\_count = x;

}

}

if (max\_count\*BLOCK\_WIDTH > IMAGE\_WIDTH)

{

BLOCK\_WIDTH = static\_cast<double>(IMAGE\_WIDTH) / static\_cast<double>(max\_count);

}

for (size\_t i = 0; i < bin\_count; i++)

{

svg\_text(TEXT\_LEFT, i\*(BIN\_HEIGHT+LINE\_HEIGHT) + TEXT\_BASELINE, bin\_namings[i]);

for (size\_t j = 0; j < (bins[i]); j++)

{

svg\_rect(TEXT\_LEFT+TEXT\_WIDTH\*(bin\_namings[i].length())+(BLOCK\_WIDTH\*(j-1)), i\*(BIN\_HEIGHT+LINE\_HEIGHT), BLOCK\_WIDTH, BIN\_HEIGHT,"red","white");

}

if (i>0)

{

cout << svg\_line(0,i\*(BIN\_HEIGHT)+(i-1)\*LINE\_HEIGHT+LINE\_HEIGHT/2,IMAGE\_WIDTH,i\*(BIN\_HEIGHT)+(i-1)\*LINE\_HEIGHT+LINE\_HEIGHT/2,"black",LINE\_HEIGHT,dash,dasharray)<<endl;

}

cout << endl;

}

svg\_text(TEXT\_LEFT, (bin\_count+1)\*(BIN\_HEIGHT+LINE\_HEIGHT) + TEXT\_BASELINE, make\_info\_text());

svg\_end();

}

Ссылка на репозиторий:

https://github.com/LvovMD/cs-lab03/tree/lab04