Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Лабораторная работа №4**

**«Библиотеки»**

Вариант 18

Задание выполнил: **Черемных Ю. А.**

Студент группы: **A-01-19**

Преподаватели: **Мохов А. С. , Козлюк Д. А.**

Москва

2020

# **Цель 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.

#### **Вариант 18**

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок время, затраченное на получение IP-адреса сервера по его имени (name lookup).

**Описание логики решения индивидуального варианта:**

Воспользовалсядокументацией <https://curl.haxx.se/libcurl/c/CURLINFO_NAMELOOKUP_TIME.html>

**Исходный код всех модулей**

**main.cpp**

*#include <iostream>*

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <sys/utsname.h>

using namespace std;

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

Input input;

if (argc > 1) {

input = download(argv[1]);

} else {

input = read\_input(cin, true);

}

if (check\_text\_parametr(input.text\_decoration) == false){

cerr << "Text parametr is incorrect. Please rewrite it and restart the program";

svg\_begin(1000,500);

svg\_text(20, 20, "Text parametr is incorrect. Please rewrite it and restart the program");

svg\_end();

return 1;

}

//MAKE HISTOGRAM

const auto bins = make\_histogram(input);

show\_histogram\_svg(bins,input.text\_decoration);

return 0;

}

**histogram.cpp**

#include "histogram.h"

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

#include <curl/curl.h>

using namespace std;

/\*

vector<double>

input\_numbers(size\_t count) {

vector<double> result(count);

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

cin >> result[i];

}

return result;

}

\*/

vector<double>

input\_numbers(istream& in, size\_t count) {

vector<double> result(count);

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

in >> result[i];

}

return result;

}

Input

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

Input data;

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

size\_t number\_count;

in >> number\_count;

if (prompt) cerr << "Enter numbers: ";

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

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

in >> data.bin\_count;

if (prompt) cerr << "Enter text decoration parametr";

string text\_decoration;

in >> data.text\_decoration;

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);

(\*buffer).write(reinterpret\_cast<const char\*>(items), data\_size);

return data\_size;

}

Input

download(const string& address) {

stringstream buffer;

double namelookup;

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

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

if(curl) {

CURLcode res;

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

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

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

res = curl\_easy\_perform(curl);

if (res) {

cerr << curl\_easy\_strerror(res) << endl;

exit(1);

}

if(CURLE\_OK == res) {

res = curl\_easy\_getinfo(curl, CURLINFO\_NAMELOOKUP\_TIME, &namelookup);

if(CURLE\_OK == res) {

cerr << "time lookup = " << namelookup << endl;

}

}

}

curl\_easy\_cleanup(curl);

return read\_input(buffer, false);

}

bool check\_text\_parametr(string text\_decoration){

if((text\_decoration != "none") && (text\_decoration != "underline") && (text\_decoration != "overline") && (text\_decoration != "line-through") && (text\_decoration != "")){

return false;

}

return true;

}

void

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

if (numbers.size() == 0){

return;

}

min = numbers[0];

max = numbers[0];

for (double x : numbers) {

if (x < min) {

min = x;

}

else if (x > max) {

max = x;

}

}

return;

}

vector<size\_t>

make\_histogram(const Input input){

vector<size\_t> bins(input.bin\_count, 0);

double min, max;

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

for (double x : input.numbers){

size\_t bin\_index = (size\_t)((x - min)/(max - min) \* input.bin\_count);

if (bin\_index == input.bin\_count){

bin\_index--;

}

bins[bin\_index]++;

}

return bins;

}

void

show\_histogram\_text(const vector<size\_t>& bins){

for(size\_t bin : bins){

if (bin < 100) {

cout << ' ';

}

if (bin < 10) {

cout << ' ';

}

cout << bin << "|";

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

cout << "\*";}

cout << endl;

}

return;

}

***histogram.h***

*#ifndef HISTOGRAM\_H\_INCLUDED*

#define HISTOGRAM\_H\_INCLUDED

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

string text\_decoration;

};

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

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

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

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

Input download(const string& address);

bool check\_text\_parametr(string text\_decoration);

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

vector<size\_t> make\_histogram(const Input input);

void show\_histogram\_text(const vector<size\_t>& bins);

#endif // HISTOGRAM\_H\_INCLUDED

**svg.cpp**

#include "histogram.h"

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

#include <sys/utsname.h>

using namespace std;

void

svg\_text(double left, double baseline, string text, string text\_decoration) {

cout << "<text x='" << left << "' y='" << baseline << "' text-decoration='" << text\_decoration << "' >" << text <<"</text>";

}

void

svg\_rect(double x, double y, double width, double height,

string stroke , string fil){

cout << "<rect x='" << x <<"' y='" << y << "' width='" << width << "' height='" << height

<< "' stroke='" << stroke << "' fill='" << fil << "' />";

}

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";

}

string

make\_info\_text() {

//Windows v5.1 (build 1234)

//Computer name: My-Comp

stringstream buffer;

struct utsname namesys;

uname(&namesys);

buffer << namesys.sysname << " ( " << namesys.version << " ) | ";

buffer << "Computer name: " << namesys.nodename;

return buffer.str();

}

void

show\_histogram\_svg(const vector<size\_t>& bins,string text\_decoration) {

const auto IMAGE\_WIDTH = 1000;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 10;

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

double top = 20;

size\_t max\_count = bins[0];

for(size\_t bin : bins){

if (bin > max\_count) {

max\_count = bin;

}

}

const size\_t SCREEN\_WIDTH = 100;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 50 - 20;

for (size\_t bin : bins) {

const bool scaling\_needed = max\_count < MAX\_ASTERISK;

size\_t binkoeff = bin;

if (scaling\_needed) {

const double koeff = (double)MAX\_ASTERISK / max\_count;

binkoeff= (size\_t)(bin \* koeff);

}

const double bin\_width = BLOCK\_WIDTH \* binkoeff;

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin), text\_decoration);

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "green", "#ffeeee");

top += BIN\_HEIGHT;

}

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, make\_info\_text(), text\_decoration);

// svg\_text(TEXT\_LEFT, TEXT\_BASELINE, to\_string(bins[0]));

// svg\_rect(TEXT\_WIDTH, 0, bins[0] \* BLOCK\_WIDTH, BIN\_HEIGHT);

svg\_end();

}

***svg.h***

*#ifndef SVG\_H\_INCLUDED*

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

void svg\_text(double left, double baseline, string text, string text\_decoration = "none");

void svg\_rect(double x, double y, double width, double height,

string stroke = "black", string fil = "black");

void svg\_begin(double width, double height);

void svg\_end();

void show\_histogram\_svg(const vector<size\_t>& bins,string text\_decoration);

#endif

Ссылка на репозитарий: https://github.com/yurchest/cslab3

(ветка lab4)