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 № 4**

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

# «Библиотеки. Низкоуровневые операции»

Выполнила студентка группы А-02-19

Носкова А.В

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 20 (2)**

Добавьте программе опцию -format, которая определяет формат вывода: lab03.exe -format text — текст, lab03.exe -format svg — изображение. Опция может стоять до или после URL: lab03.exe -format svg http://... или lab04.exe http://... -format svg. Если после -format не стоит text или svg, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Описание решения индивидуального задания:**

В заголовочном файле lab03.h создала структуру Option для определения формата вывода. Определение структуры представляет собой функцию Option format, в которой путем сравнения строк определяется, в каком формате должен быть представлен результат программы.

Если пользователь ввел в командную строку –format txt, то результат будет представлен в виде текста, если же –format svg, то в виде изображения. Иначе, если после –format не стоит txt или svg, тогда будет выведено сообщение с подсказкой: "Enter option -format txt or -format svg", после чего программа завершит работу.

**Код:**

**Main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include "lab03.h"

#include "svg.h"

#include <curl/curl.h>

#include <sstream>

#include <windows.h>

using namespace std;

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;

}

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

{

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

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

for (size\_t bin : bins)

{

if (bin < 100)

{

cout << ' ';

}

if (bin < 10)

{

cout << ' ';

}

cout << bin << "|";

size\_t height = bin;

if (scaling\_needed)

{

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

height = (size\_t)(bin \* scaling\_factor);

}

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

{

cout << '\*';

}

cout << '\n';

}

cout << information\_system();

}

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

{

Input data;

size\_t number\_count;

if (promt)

{

cerr << "Enter number count: ";

in >> number\_count;

cerr << "Enter numbers: ";

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

cerr << "Enter column count: ";

in >> data.bin\_count;

}

else

{

in >> number\_count;

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

in >> data.bin\_count;

}

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

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

return data\_size;

}

Input download(const string& address) {

stringstream buffer;

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 != CURLE\_OK)

{

cout << curl\_easy\_strerror(res);

exit (1);

}

curl\_easy\_cleanup (curl);

}

return read\_input(buffer, false);

}

Option format(int argc, char\*\* argv)

{

Option form;

form.txt = false;

form.svg = false;

form.help = false;

form.url = 0;

for (int i = 1; i<argc; i++)

{

if (strcmp(argv[i], "-format") == 0)

{

if (strcmp(argv[i+1], "txt") == 0)

{

form.txt = true;

i++;

}

else if (strcmp(argv[i+1], "svg") == 0)

{

form.svg = true;

i++;

}

else {

form.help = true;

}

}

else {

form.url = argv[i];

}}

return form;

}

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

Option output = format(argc, argv);

if (output.help) {

cerr << "Enter option -format txt or -format svg" ;

exit(2);

}

Input input;

if (output.url)

{

input = download(output.url);

} else

{

input = read\_input(cin, true);

}

const auto bins = make\_histogram(input);

if (output.txt)

{

show\_histogram\_text(bins);

}

if (output.svg)

{

show\_histogram\_svg(bins);

}

return 0;

}

**Lab03.h**

#ifndef LAB03\_H\_INCLUDED

#define LAB03\_H\_INCLUDED

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

};

struct Option

{

bool txt;

bool svg;

bool help;

char \*url;

};

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

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

string information\_system();

#endif // LAB03\_H\_INCLUDED

**Lab03.cpp**

#pragma once

#include <vector>

#include <iostream>

#include <vector>

#include "lab03.h"

#include <windows.h>

#include <sstream>

#include <string>

using namespace std;

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

{

if (numbers.size()!=0)

{

min = numbers[0];

max = numbers[0];

for(double number: numbers)

{

if(min>number)

min=number;

if(max<number)

max=number;

}

}

}

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

{

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

double max, min;

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

for (double number : data.numbers)

{

size\_t bin = (size\_t)((number - min) / (max - min) \* data.bin\_count);

if (bin == data.bin\_count)

{

bin--;

}

result[bin]++;

}

return result;

}

string information\_system()

{

stringstream buffer;

DWORD mask = 0x0000ffff;

DWORD mask\_major = 0b00000000'00000000'00000000'11111111;

DWORD info = GetVersion();

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & mask\_major;

DWORD version\_minor = version >> 8;

if ((info & 0x40000000) == 0)

{

DWORD build = platform;

buffer << "Windows:" << version\_major << "." << version\_minor << " (build" << build << ")\n";

}

char get\_computer\_name[MAX\_COMPUTERNAME\_LENGTH+1];

DWORD size = sizeof(get\_computer\_name);

GetComputerNameA(get\_computer\_name, &size);

buffer << "Computer name:" << get\_computer\_name << "\n";

return buffer.str();

}

**Svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <string>

#include <vector>

#include <iostream>

using namespace std;

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

void svg\_end();

void svg\_text(double left, double baseline, size\_t text);

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black");

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

#endif // SVG\_H\_INCLUDED

**Svg.cpp**

#include "svg.h"

#include <vector>

#include <iostream>

#include <string>

#include <sstream> // for stringsteam

#include "lab03.h"

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, size\_t text)

{

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

}

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 << "' />";

}

size\_t find\_max\_column(const vector<size\_t>& bins)

{

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

for (size\_t bin : bins)

{

if (max\_column < bin)

max\_column = bin;

}

return max\_column;

}

size\_t find\_min\_column(const vector<size\_t>& bins)

{

size\_t min\_column = bins[0];

for (size\_t bin : bins)

{

if (min\_column > bin)

min\_column = bin;

}

return min\_column;

}

string create\_color\_column(const vector<size\_t>& bins, size\_t max\_count, size\_t bin)

{ if (bins.size()!=0)

{

size\_t min = find\_min\_column(bins);

size\_t max = find\_max\_column(bins);

size\_t color\_column;

if (bin == max)

{

color\_column = 111;

}

else if (bin == min)

{

color\_column = 999;

}

else

{

color\_column = (10 - (bin \* 9) / max\_count)\*111;

}

ostringstream c;

c << color\_column;

string color = c.str();

return color;

}

return "000";

}

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

{

const auto IMAGE\_WIDTH = 400;

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;

const size\_t MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_WIDTH - TEXT\_LEFT;

double top = 0;

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

const bool scaling\_needed = max\_count \* BLOCK\_WIDTH > MAX\_ASTERISK;

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

for (size\_t bin : bins)

{

string color = create\_color\_column(bins, max\_count, bin);

cout << color << endl;

size\_t height = bin;

if (scaling\_needed)

{

const double scaling\_factor = (double)MAX\_ASTERISK / (max\_count \* BLOCK\_WIDTH);

height = (size\_t)(bin \* scaling\_factor);

}

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

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, bin);

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "#CCFF00", "#" + color);

top += BIN\_HEIGHT;

}

cout << "<text x='" << TEXT\_LEFT << "' y='"<<top+BIN\_HEIGHT<<"'>"<<information\_system()<<"</text>";

svg\_end();

}

**Lab03-test/test.cpp**

#include "../lab03.h"

#include "../svg.h"

#include <cassert>

void test\_positive()

{

double min = 0;

double max = 0;

find\_minmax({1, 2, 3}, min, max);

assert(min == 1);

assert(max == 3);

}

void test\_negative()

{

double min = 0;

double max = 0;

find\_minmax({-1, -2, -3}, min, max);

assert(min == -3);

assert(max == -1);

}

void test\_same()

{

double min = 0;

double max = 0;

find\_minmax({2, 2, 2}, min, max);

assert(min == 2);

assert(max == 2);

}

void test\_one()

{

double min = 0;

double max = 0;

find\_minmax({1}, min, max);

assert(min == 1);

assert(max == 1);

}

void test\_empty()

{

double min = 0;

double max = 0;

find\_minmax({}, min, max);

assert(min == 0);

assert(max == 0);

}

void test\_homework\_4 ()

{

vector<size\_t> bins = {};

assert(create\_color\_column(bins,0,0) == "000");

}

void test\_homework\_1 ()

{

vector<size\_t> bins = {3,4,7,21};

assert(create\_color\_column(bins,21,bins[0]) == "999");

assert(create\_color\_column(bins,21,bins[1]) == "999");

assert(create\_color\_column(bins,21,bins[2]) == "777");

assert(create\_color\_column(bins,21,bins[3]) == "111");

}

void test\_homework\_2 ()

{

vector<size\_t> bins = {1,9,7};

assert(create\_color\_column(bins,9,bins[0]) == "999");

assert(create\_color\_column(bins,9,bins[1]) == "111");

assert(create\_color\_column(bins,9,bins[2]) == "333");

}

void test\_homework\_3 ()

{

vector<size\_t> bins = {5,6};

assert(create\_color\_column(bins,6,bins[0]) == "999");

assert(create\_color\_column(bins,6,bins[1]) == "111");

}

int

main()

{

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

test\_homework\_1();

test\_homework\_2();

test\_homework\_3();

test\_homework\_4();

}

**Ссылка:** <https://github.com/NoskovaAV/cs-lab03/tree/double>