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 ПО систем управления»**

# «Библиотеки. Низкоуровневые операции.»

Выполнил студент группы А-01-19

Шахов А. И.

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include "histogram.h"

#include "svg.h"

#include <windows.h>

#include <curl/curl.h>

using namespace std;

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

{

vector<double> result(count);

for (double& number: result)

{

in >> number;

}

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: ";

size\_t bin\_count;

in >> bin\_count;

data.bin\_count = bin\_count;

return data;

}

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

{

auto bin\_count = input.bin\_count;

auto numbers = input.numbers;

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

double numbers\_min, numbers\_max;

find\_minmax(numbers, numbers\_min, numbers\_max);

for (double number : numbers)

{

size\_t bin = (size\_t)((number - numbers\_min) / (numbers\_max - numbers\_min) \* bin\_count);

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

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);

string\* stringItems = reinterpret\_cast<string\*>(items);

buffer->write(stringItems->c\_str(), data\_size);

return 0;

}

Input

download(const string& address) {

stringstream buffer;

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) {

exit(1);

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

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

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

{

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

Input input;

if (argc > 1) {

input = download(argv[1]);

}

else {

input = read\_input(cin, true);

}

const auto bins = make\_histogram(input);

show\_histogram\_svg(bins);

}

**histogram.cpp**

#include "histogram.h"

#include <iostream>

#include <cassert>

using namespace std;

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

{

if (numbers.empty()) {

cerr << "vector is empty\n";

assert(false);

return;

}

else {

min = numbers[0];

max = numbers[0];

}

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

**svg.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.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" << '\n';

}

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

{

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

}

void svg\_end()

{

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

}

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 << "' />" << '\n';

}

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

{

const auto IMAGE\_WIDTH = 500;

const auto IMAGE\_HEIGHT = 400;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 50;

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

int top = 0;

for (size\_t bin : bins) {

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

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

switch (top){

case 0: svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "red", "red");

break;

case 30: svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "green", "green");

break;

case 60: svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "blue", "blue");

break;

}

top += BIN\_HEIGHT;

}

svg\_end();

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

using namespace std;

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

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

void svg\_end();

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

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

#endif // SVG\_H\_INCLUDED

**hictogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

#endif // HISTOGRAM\_H\_INCLUDED