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 и низкоуровневые операции»**

Выполнила студентка

группы А-01-19

Шашерина А.В.

Проверили

Мохов А.С.

Козлюк Д.А.

Москва 2020

**Код:**

**main.cpp**

#include <iostream>

#include <vector>

#include <curl/curl.h>

#include <sstream>

#include <string>

#include <windows.h>

#include "histogram.h"

#include "svg.h"

using namespace std;

vector<double> input\_numbers(istream& in, const 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;

cerr << "Enter numbers: ";

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

cerr << "Enter column count: ";

in>>data.bin\_count;

}

else

{

size\_t number\_count;

in >> number\_count;

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

in>>data.bin\_count;

}

return data;

}

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

{

double min, max;

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

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

for (double number : data.numbers)

{

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

if (bin == data.bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

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';

}

}

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

{

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

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

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

buffer->write(new\_items, 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) << endl;

exit(1);

}

}

curl\_easy\_cleanup(curl);

return read\_input(buffer, false);

}

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

{

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);

return 0;

}

**histogram.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(const vector<double> numbers, double& min, double& max) ;

#endif // HISTOGRAM\_H\_INCLUDED

**histogram.cpp**

#include "histogram.h"

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 (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

#include <string>

using namespace std;

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

void svg\_end();

void svg\_text(double left, double baseline, string 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 <windows.h>

#include <string>

#include<sstream>

void svg\_begin(double width, double height)

{

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

cout << "<svg "

<< "width='" << width << "' "

<< "height='" << height << "' "

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

<< "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>";

}

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 << "' />";

}

string show\_name\_version ()

{

stringstream buffer;

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD mask2 = 0x000000ff;

DWORD version\_major = version & mask2;

DWORD version\_minor = version >>8;

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

{

DWORD build = platform;

}

DWORD build = platform;

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

char computer\_name[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD size = sizeof(computer\_name);

GetComputerNameA(computer\_name, &size);

buffer << "Computer name: " << computer\_name << "\n";

return buffer.str();

}

void

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

{

const auto IMAGE\_WIDTH = 600;

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;

double top = 0;

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

const size\_t MAX\_VIDTH = IMAGE\_WIDTH- TEXT\_WIDTH;

const size\_t MAX\_ASTERISK = 35;

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

for (size\_t bin: bins)

{

double bin\_factor ;

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

if (scalling\_needed)

{

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

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

}

else {bin\_factor=bin;}

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

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

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

top += BIN\_HEIGHT;

}

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

svg\_end();

}

**.gitignore**

/curl

/bin

/obj