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**

**РПОСУ**

**«Декомпозиция программы»**

Выполнил студент группы А-03-20

Шилкина Мария Игоревна

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

**Цель работы:**

* Уметь структурировать программу при помощи функций.
* Уметь писать модульные тесты.

**Код программы**

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <cstring>

#include <curl/curl.h>

#include <Windows.h>

#include "svg.h"

#include "histogram.h"

#include <windows.h>

using namespace std;

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

auto data\_size = item\_size \* item\_count;

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

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

return data\_size;

}

Input

download(const string& address) //использование библиотеки curl

{

stringstream buffer;

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

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

if(curl)

{

CURLcode res;

double connect;

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(CURLE\_OK == res)

{

res = curl\_easy\_getinfo(curl, CURLINFO\_CONNECT\_TIME, &connect);

cerr <<"Connect: " << connect;

}

curl\_easy\_cleanup(curl);

if(res)

{

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

exit(1);

}

}

return read\_input(buffer, false);

}

struct ConsoleArguments {

bool valid;

bool use\_remote\_file;

const char\* remote\_file\_address;

bool use\_svg\_output;

};

ConsoleArguments parse\_arguments(int argc, char\* argv[]) {

ConsoleArguments res;

res.valid = false;

res.use\_remote\_file = false;

res.use\_svg\_output = true;

int i = 1;

while (i < argc) {

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

if ((i+1) < argc) {

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

res.use\_svg\_output = true;

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

res.use\_svg\_output = false;

else return res; //

} else return res;

i += 1;

} else {

res.use\_remote\_file = true;

res.remote\_file\_address = argv[i];

}

++i;

}

res.valid = true;

return res;

}

void print\_usage\_message(const char\* progname) {

cerr << "Неправильно введены аргументы : " << progname << " [-format {svg|txt}] [input URL]"

<< endl;

}

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

setlocale(LC\_ALL, "Russian");

ConsoleArguments args = parse\_arguments(argc, argv);

if (!args.valid) {

print\_usage\_message(argv[0]);

return 0;

}

Input input; //библиотека для скачивания

if (args.use\_remote\_file) {

input = download(args.remote\_file\_address);

} else {

input = read\_input(cin, true);

}

const auto bins = make\_histogram(input);

if (args.use\_svg\_output)

show\_histogram\_svg(bins);

else

show\_histogram\_text(bins);

return 0;

}

**Svg.cpp**

#include <iostream>

#include <string>

#include <vector>

#include "svg.h"

using namespace std;

string

add\_text\_to()

{

stringstream buffer;//начало хуеты

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD mask\_2 = 0x0000ff;

if ((info & 0x80000000) == 0)

{

DWORD version\_major = version & mask\_2;

DWORD version\_minor = version >> 8;

DWORD build = platform;

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

}

char computer\_name[MAX\_COMPUTERNAME\_LENGTH + 1];//помогли одногруппники это мой комьютер так зовут

DWORD size = MAX\_COMPUTERNAME\_LENGTH+1;

GetComputerNameA(computer\_name, &size);

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

return buffer.str();

return 0;

}

//конец

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\_rect(double x, double y, double width, double height, string stroke, string fill) {//сам квадратик гистограммы

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

}

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

{

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

}

string brightness (size\_t bin,size\_t max\_count)

{

int pr;

string s="#";

if (max\_count == 0) return "Error";

else {

pr=10 - (bin \* 9) / max\_count;

for(int i=0;i<3;i++)

{

if (pr > 9) pr = 9;

s += to\_string(pr);

}

return s;

}}

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;

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;

double scaling\_factor;

if (scaling\_needed)

{

scaling\_factor = static\_cast<double>(MAX\_ASTERISK) / (max\_count \* BLOCK\_WIDTH);

}

else {

scaling\_factor = 1;

}

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

double top = 50;

for (size\_t bin : bins) {

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

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

string fill;

fill=brightness(bin,max\_count);

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"orange", fill);

top += BIN\_HEIGHT;

}

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

svg\_end();

}

**Histogram.cpp**

#include <iostream>

#include <vector>

#include "histogram.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;

}

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;

return data;

}

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

if (numbers.size()) {

max = numbers[0];

min = numbers[0];

if (numbers.size()) {

for (double x : numbers) {

if (x > max) {

max = x;

}

if (x < min) {

min = x;

}

}

}

}

}

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

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

double min, max;

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

if (min == max) {

for (double number : data.numbers) {

bins[0]++;

}

return bins;

}

else {

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(const 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';

}

}

**Histogram.h**

#pragma once

#include <vector>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

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

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

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

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

Svg.h

#include <iostream>

#include <vector>

#include <string>

#include <windows.h>

#include <sstream>

using namespace std;

string add\_text\_to();

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

void svg\_end();

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

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

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

string brightness (size\_t bin,size\_t max\_count);