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»**

Выполнил студент группы А-02-19

Баснак Е.А.

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 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.

**Ход решения**

Перебираем массив аргументов, подающихся на вход программы, и ищем “-format”. При нахождении записываем в переменную format следующий аргумент, характеризующий тип вывода “text” или “svg”, и переменной num присваиваем позицию элемента с типом вывода и выходим из цикла. Далее проверяем правильность записанных данных, и в случае неправильного ввода, выводим ошибку. В конце проверяем позицию формата (num) и в зависимости от неё, выбираем элемент массива с ссылкой, откуда считваются входные данные.

**Ссылка на репозитарий**

https://github.com/BasnakEA/cs-lab03

**Код программы**

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include<curl/curl.h>

#include<sstream>

#include<cmath>

#include<cstdio>

#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;

}

size\_t zero\_bin\_count(const size\_t& number\_count) {

size\_t k = sqrt(number\_count);

if (k > 25)

{

k = 1 + log2(number\_count);

/\*cout << "Sterdjis rule" << endl;

return k;\*/

}

//cout << "Emper formula" << endl;

return k;

}

Input

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

Input data;

size\_t number\_count;

if (prompt)

{

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;

}

if (data.bin\_count == 0)

{

data.bin\_count = zero\_bin\_count(number\_count);

}

return data;

}

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;

char\* format;

int num;

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

{

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

{

if (i != argc - 1)

{

format = argv[i+1];

}

num = i+1;

break;

}

}

if (((strcmp(format, "text") != 0) && (strcmp(format, "svg") != 0)) || (num == argc))

{

cout << "You need to enter to '-format' and then the format type ('text' or 'svg')!";

exit(1);

}

if (argc > 1)

{

if (num == 2)

{

input = download(argv[3]);

}

else

{

input = download(argv[1]);

}

}

else

{

input = read\_input(cin, true);

}

const auto bins = make\_histogram(input);

if (strcmp(format, "text") == 0)

{

show\_histogram\_text(bins);

}

else

{

show\_histogram\_svg(bins);

}

return 0;

}

**histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include<vector>

#include<windows.h>

struct Input {

std::vector<double> numbers;

size\_t bin\_count;

};

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

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

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

void write\_version();

#endif // HISTOGRAM\_H\_INCLUDED

**histogram.cpp**

#include<vector>

#include<iostream>

#include"histogram.h"

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 (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

}

//<<<<<<< HEAD

//=======

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

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

double min;

double max;

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;

}

void write\_version()

{

DWORD dwVersion = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = dwVersion&mask;

DWORD platform = dwVersion >> 16;

DWORD mask2 = 0x00ff;

DWORD version\_major = version&mask2;

DWORD version\_minor = version >> 8;

char buffer[MAX\_COMPUTERNAME\_LENGTH+1]="";

if ((version & 0x80000000) == 0) {

DWORD size =MAX\_COMPUTERNAME\_LENGTH+1;

GetComputerNameA(buffer, &size);

}

DWORD build = platform;

cout << "Windows v" << version\_major << "."

<< version\_minor << " (build " << build << ")" << endl;

cout << "Computer name: " << buffer << endl;

}

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';

}

write\_version();

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include "histogram.h"

#include <vector>

#include <string>

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, std::string stroke = "black", std::string fill = "black");

std::string make\_color(const std::vector<size\_t>& bins, size\_t bin, size\_t max\_count);

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

size\_t find\_min(const std::vector<size\_t>& bins);

size\_t find\_max(const std::vector<size\_t>& bins);

void show\_version(double y);

void show\_computername(double y);

#endif // SVG\_H\_INCLUDED

**svg.cpp**

#include "svg.h"

#include <vector>

#include <string>

#include <iostream>

#include <sstream>

#include<cstdio>

//#include<windows.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";

}

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

size\_t min = bins[0];

for (const auto& bin : bins)

{

if (bin < min)

{

min = bin;

}

}

return min;

}

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

size\_t max = bins[0];

for (const auto& bin : bins)

{

if (bin > max)

{

max = bin;

}

}

return max;

}

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

ostringstream os;

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

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

size\_t x;

if (bin == max)

{

x = 1;

}

else if (bin == min)

{

x = 9;

}

else

{

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

}

os << x;

string color = os.str();

color += color + color;

return color;

}

void show\_version(double y, const double TEXT\_BSLN)

{

DWORD dwVersion = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = dwVersion&mask;

DWORD platform = dwVersion >> 16;

DWORD mask2 = 0x00ff;

DWORD version\_major = version&mask2;

DWORD version\_minor = version >> 8;

if ((version & 0x80000000) == 0) {

char buffer[MAX\_COMPUTERNAME\_LENGTH+1]="";

DWORD size =MAX\_COMPUTERNAME\_LENGTH+1;

GetComputerNameA(buffer, &size);

cout << "<text x='" << left << "' y='" << y + 2\*TEXT\_BSLN << "'>Computer name: " << buffer << "</text>";

}

DWORD build = platform;

cout << "<text x='" << left << "' y='" << y + TEXT\_BSLN << "'>Windows v" << version\_major << "."

<< version\_minor << " (build " << build << ")</text>";

}

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\_LEFT - 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;

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

double top = 0;

for (size\_t bin : bins) {

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

cout << endl << 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, "red", "#" + color);

top += BIN\_HEIGHT;

}

show\_version(top, TEXT\_BASELINE);

//show\_computername(top + 2\*TEXT\_BASELINE);

svg\_end();

}