Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Отчёт по лабораторной работе №4**

**Вариант 8**

*Задание выполнил: Ларчев В.И.*

*Студент факультета ИВТИ*

*направления УТС*

*группы A-03-19*

*Проверил:*

*Оценка:*

Москва 2020 НИУ «МЭИ»

**Постановка задачи (ПЗ)**

# Часть 1. Библиотеки

*Цель работы:*

1. Уметь устанавливать и подключать к программе внешние библиотеки.
2. Уметь использовать типовые элементы API: функции обратного вызова, битовые флаги и маски, массивы и строки C.
3. Уметь работать с параметрами командной строки программы.

*Задание:*

Добавить возможность построения гистограммы по данным из файла из сети. Адрес файла задается аргументом командной строки программы. Если адрес не задан, читать данные со стандартного ввода, как раньше.

Работу нужно вести на основе кода общего задания к ЛР № 3 в старом репозитарии в отдельной ветке без создания нового проекта. По этой причине во всех примерах используется lab03.exe.

Пример строки запуска:

lab03.exe http://uii.mpei.ru/study/courses/cs/lab03/marks.txt >marks.svg

# Часть 2. Низкоуровневые операции

*Цель работы:*

1. Уметь применять побитовые операции для типовых сценариев.
2. Уметь работать с API, принимающими указатели, в том числе строки C.
3. Знать характерные особенности документации на API библиотек.

*Задание:*

Код, который будет написан в этой части работы, должен в итоге формировать строку, которую нужно отобразить снизу итогового SVG в формате:

Windows v5.1 (build 1234)

Computer name: My-Comp

**Описание логики решения**

Для выполнения ИДЗ первой части ЛР был изучен API библиотеки и после этого для функции curl\_easy\_getinfo() выбран вторым аргументом CURLINFO\_PRIMARY\_IP, а также был переделан под исходную программу пример из документации.

Для добавления решения ИДЗ из третьей ЛР не было никаких изменений в его реализации, возможных из-за использования библиотеки.

Слияние второй части ЛР с первой не вызвало серьёзных конфликтов, разрешение большей части из них заключалось в последовательном представлении кода из разных частей, а в функции вывода гистограммы добавилась строка вывода необходимой информации о компьютере, которая передаётся строкой в формальные параметры функции.

**Программа на C++**

Histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <iostream>

#include <winsock2.h>

#include <windows.h>

#include <string>

#include <sstream>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

};

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

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

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

#endif // HISTOGRAM\_H\_INCLUDED

Svg.h

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

#include <windows.h>

#include <string.h>

#include <sstream>

using namespace std;

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

void svg\_end();

void svg\_text(double left, double baseline, string text, string font\_size = "12");

void svg\_rect(double x = 0, double y = 0, double width = 100, double height = 200, string stroke = "black", string fill = "black");

void show\_histogram\_svg(const vector<size\_t>& bins, string information, size\_t font\_size);

bool check\_size(size\_t font\_size);

#endif // SVG\_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;

}

}

}

}

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

{

double min = data.numbers[0];

double max = data.numbers[0];

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

vector<size\_t> result(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--;

}

result[bin]++;

}

return result;

}

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';

}

}

svg.cpp

#include "svg.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, string text, string font\_size)

{

cout << "<text x='" << left << "' y='" << baseline << "' font-size='" << font\_size << "'>"<< 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 <<"'/>";

}

void show\_histogram\_svg(const vector<size\_t>& bins, string info ,size\_t font\_size)

{

const auto IMAGE\_WIDTH = 500;

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 SCREEN\_WIDTH = 40;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 5;

size\_t max\_count = 0;

for (size\_t bin : bins)

{

if (bin > max\_count)

{

max\_count = bin;

}

}

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

for (size\_t bin : bins)

{

size\_t height = bin;

if (scaling\_needed)

{

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

double lenght\_rect = bin \* scaling\_factor;

height = (size\_t)(lenght\_rect);

}

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

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

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

top += BIN\_HEIGHT;

}

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

svg\_end();

}

bool check\_size(size\_t font\_size)

{

if(font\_size < 8 || font\_size > 32)

{

cerr << "Invalid input, font\_size => 8 and =< 32 !" << endl;

return false;

}

else return true;

}

test.cpp

#include "histogram.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\_const() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

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);

}

int main()

{

test\_positive();

test\_negative();

test\_const();

test\_one();

test\_empty();

}

test\_variant.cpp

#include "svg.h"

#include <cassert>

void test()

{

assert(check\_size(3) == false );

assert(check\_size(33) == false );

assert(check\_size(8) == true );

assert(check\_size(32) == true );

assert(check\_size(16) == true );

}

int main()

{

test();

}

main.cpp

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

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 column count: ";

size\_t bin\_count;

in >> bin\_count;

data.bin\_count = bin\_count;

return data;

}

string make\_info\_text()

{

stringstream buffer;

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD build;

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & 0xff;

DWORD version\_minor = version >> 8;

//printf("M\_version10 = %lu\n",version\_major);

//printf("M\_version16 = %08lx\n",version\_major);

//printf("m\_version10 = %lu\n",version\_minor);

//printf("m\_version16 = %08lx\n",version\_minor);

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

{

build = platform;

}

else printf("minor\_bit = %u",1);

//printf("Windows v%lu.%lu (build %lu)\n",version\_major,version\_minor,build);

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

DWORD Size = sizeof(system\_name);

GetComputerNameA(system\_name, &Size);

//printf("System name: %s\n", system\_name);

buffer << "Windows v" << version\_major << "." << version\_minor << " (build " << build << ") " << "Computer name: " << system\_name;

return buffer.str();

}

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

{

size\_t data\_size;

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) {

char \*ip;

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) && !curl\_easy\_getinfo(curl, CURLINFO\_PRIMARY\_IP, &ip) && ip) cerr << "IP:" << ip << endl;

else {

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

exit(1);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

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

string info = make\_info\_text();

Input data;

if(argc > 1)

{

data = download(argv[1]);

}

else {

data = read\_input(cin, true);

}

size\_t font\_size;

bool wrong\_size;

while (!wrong\_size)

{

cerr << "Enter font size: ";

cin >> font\_size;

wrong\_size = check\_size(font\_size);

}

const auto bins = make\_histogram(data);

show\_histogram\_svg(bins, info, font\_size);

return 0;

}