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 (ч.1 и ч.2)**

**По курсу «Разработка ПО систем управления»**

**«Библиотеки. Низкоуровневые операции»**

Выполнил студент группы А-02-19

Поташов С.Е.

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**4.1 Общее задание**

Адаптировать индивидуальное задание предыдущих ЛР к коду после рефакторинга при необходимости. Если требовалось ввести дополнительные данные, их нужно добавить в Input.

**4.2 Индивидуальные задания**

Всю необходимую информацию по функциям cURL нужно изучить по официальной документации и включить в отчет ссылку на конкретную страницу. В частности, если не указан конкретный параметр curl\_easy\_setopt(), нужно подобрать его по описанию в списке.

Во всех вариантах нужно обрабатывать ошибки cURL, как это делается в основной программе.

Если вариант требует ввода дополнительных данных, проверить работу с файлом «из сети» можно через локальный файл и адрес вида file://C:/Users/kozlyuk/Desktop/marks.txt для файла marks.txt на рабочем столе пользователя kozlyuk. Полный путь к файлу можно получить, щелкнув в адресной строке «Проводника» (так же, как перед запуском cmd).

#### **Вариант 13**

Добавьте программе опцию -stroke для задания цвета границ столбцов. Например, lab03.exe -stroke red или lab04.exe -stroke "#ff0000" делает их красными. Опция может стоять до или после URL: lab03.exe -stroke red http://... или lab04.exe http://... -fill red. Если после -stroke не стоит еще одного аргумента, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Пояснение к программе:**

Для выполнения индивидуального задания я добавил в определение функции “show\_histogram\_svg”, также сделал передачу значения переменной “stroke” в вызове функции “svg\_rect(TEXT\_WIDTH, top-bin\_width, BIN\_HEIGHT, bin\_width ,**stroke**,"#aab5ff")”. Также добавлена опция “-stroke” и учтены все частные случаи

**Текст программы:**

**Main.cpp**

#include <curl/curl.h>

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

#include <windows.h>

#include "histogram.h"

#include "svg.h"

using namespace std;

struct option {

string stroke;

bool stroke\_correct;

bool guide;

char\* url;

};

option stroke(int argc, char\*\* argv){

option str;

str.url = 0;

str.stroke\_correct=false;

str.guide = false;

for (int i = 1; i < argc; i++)

{

if (argv[i][0] == '-')

{

if(string(argv[i]) == "-stroke")

{

if(i+1<argc)

{

str.stroke = string(argv[i+1]);

if (str.stroke.size()!=0)

{

str.stroke\_correct=true;

i++;

}

else

{

str.guide=true;

}

}

else

{

str.guide = true;

}

}

}

else

{

str.url=argv[i];

}

}

return str;

}

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;

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;

}

return data;

}

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;

}

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,const option &str) {

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;

option str = stroke(argc,argv);

if (str.guide)

{

cerr<<"Error";

return 1;

}

if (str.url)

{

input = download(str.url,str);

}

else

{

input = read\_input(cin, true);

}

/\*if (argc > 1)

{

input = download(argv[1]);

} else

{

input = read\_input(cin, true);

}

\*/

const auto bins = make\_histogram(input);

show\_histogram\_svg(bins,str.stroke);

return 0;

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

#include "histogram.h"

using namespace std;

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

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

#endif // SVG\_H\_INCLUDED

**svg.cpp**

#include "svg.h"

#include <vector>

#include <string>

#include <iostream>

#include "histogram.h"

using namespace std;

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

cout << "<?xml version='1.0' encoding='Windows-1251'?>\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) {

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 << "' />";

}

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

if (bins.size() != 0)

{

size\_t max = bins[0];

for (const auto& bin : bins)

{

if (bin > max)

{

max = bin;

}

}

return max;

}

else

return 0;

}

void show\_histogram\_svg(const vector<size\_t>& bins, string stroke) {

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 10;

const auto TEXT\_BASELINE = 20;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 10;

const auto BLOCK\_HEIGHT = 100;

double TEXT\_WIDTH = 10;

const auto TEXT\_HEIGHT = 75;

const size\_t MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_LEFT - TEXT\_WIDTH;

const auto BAR\_HEIGHT = IMAGE\_HEIGHT - TEXT\_HEIGHT;

size\_t max\_count = 0;

for (size\_t count : bins) {

if (count > max\_count) {

max\_count = count;

}

}

const bool scaling\_needed = max\_count \* BLOCK\_HEIGHT > BAR\_HEIGHT;

double factor=1;

if (scaling\_needed){

factor = (double)BLOCK\_HEIGHT / (max\_count \* BLOCK\_HEIGHT);

}

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

double top = 0;

double height\_max = find\_max(bins)\*BLOCK\_HEIGHT\*factor+10 ;

for (size\_t bin : bins) {

size\_t height = bin\*BLOCK\_HEIGHT\*factor;

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

svg\_rect(TEXT\_WIDTH, height\_max - height, BIN\_HEIGHT, height,stroke,"#aab5ff");

top += BIN\_HEIGHT;

TEXT\_WIDTH+= BIN\_HEIGHT;

}

svg\_text(TEXT\_LEFT,top+BIN\_HEIGHT+height\_max, information\_system());

svg\_end();

}

**Histogram.cpp**

#include <vector>

#include <iostream>

#include "histogram.h"

#include <windows.h>

#include <sstream>

#include <string>

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;

}

}

}

}

string information\_system()

{

stringstream buffer;

DWORD mask = 0x0000ffff;

DWORD mask\_major = 0x000000ff;

DWORD info = GetVersion();

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & mask\_major;

DWORD version\_minor = version >> 8;

/\*printf("Windows decimal-version is %u.\n", version);

printf("Windows 16-x version is %x.\n", version);

printf("Platform is %u.\n", platform);

printf("Windows major version is %u.\n", version\_major);

printf("Windows minor version is %u.\n", version\_minor);\*/

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

{

DWORD build = platform;

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

}

char get\_computer\_name[MAX\_COMPUTERNAME\_LENGTH+1];

DWORD size = sizeof(get\_computer\_name);

GetComputerNameA(get\_computer\_name, &size);

buffer << "Computer name:" << get\_computer\_name << "\n";

return buffer.str();

}

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';

}

}

**Histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <sstream>

#include <string>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

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

string information\_system();

#endif // HISTOGRAM\_H\_INCLUDED