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-01-19

# Медведев И.С.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

# Задание:

1. 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.
2. 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

#### **Вариант 11**

С помощью функции curl\_version\_info() печатайте на стандартный вывод ошибок список протоколов, которые поддерживает cURL (protocols).

**Описание логики решения индивидуального задания:**

Чтобы получить список протоколов, которые поддерживает cURL, нужно вызвать указатель на структуру, где они содержатся. Для этого воспользуемся функцией curl\_version\_info(). Чтобы получить соответствующую версию нужно обратится к ней в структуре с помощью ( -> ) . И так как protocols это указатель на массив указатель, для вывода списка протоколов используем цикл с предусловием.

**Ссылка на репозитарий:**

<https://github.com/MedvedevIgS/lab-03.git>

**Код программы:**

Main.cpp

#include <iostream>

#include <vector>

#include "histogram.h"

#include "SVG.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

{

vector<double> result(count);

cerr << "Enter numbers: ";

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 bin count: ";

in >> data.bin\_count;

}

else

{

in >> number\_count;

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

in >> data.bin\_count;

}

return data;

}

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';

}

return;

}

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

{

vector<size\_t> bins(data.bin\_count, 0);

double max, min;

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--;

}

bins[bin]++;

}

return bins;

}

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;

curl\_version\_info\_data \*prot=curl\_version\_info(CURLVERSION\_NOW);

int i =0 ;

while (prot->protocols[i] !=0 )

{

cerr << prot->protocols[i] <<endl ;

i=i+1;

}

exit(1);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

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

{

Input data;

if (argc > 1)

{

data = download(argv[1]);

}

else

{

data = read\_input(cin, true);

}

const auto bins = make\_histogram(data);

show\_histogram\_svg(bins, data.bin\_count);

return 0;

}

SVG.h

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

void svg\_line(double x\_1, double y\_1, double x\_2, double y\_2, size\_t width);

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

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

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

void svg\_end();

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

#endif // SVG\_H\_INCLUDED

SVG.cpp

#include "SVG.h"

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <windows.h>

using namespace std;

void svg\_line(double x\_1, double y\_1, double x\_2, double y\_2, size\_t width)

{

cout << "<line x1='"<<x\_1<<"' y1='"<<y\_1<<"' x2='"<<x\_2<<"' y2='"<<y\_2<<"' style='stroke-dasharray: 10 10' stroke='blue' stroke-width='"<<width<<"'/>\n";

}

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<<"'/>\n";

}

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

{

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

}

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";

}

string info\_text ()

{

stringstream buff;

DWORD info=GetVersion();

DWORD mask = 0b00000000'00000000'11111111'11111111;

DWORD version = info & mask;

//printf("version %u\n",version);

DWORD mask\_minor = 0x000000ff;

DWORD mask\_major = 0x0000ff00;

DWORD platform = info >> 16;

DWORD version\_minor = info & mask\_minor;

DWORD version\_major = info & mask\_major;

DWORD version\_major16 = version\_major >> 8;

if ((info & 0b10000000'00000000'0000000'00000000) == 0)

{

DWORD build = platform;

buff<<"Windows v"<<version\_minor<<"."<<version\_major16<<" (build "<<build<<") ";

//printf("build %u.\n", build);

}

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

DWORD Size = sizeof(system);

GetComputerNameA(system, &Size);

buff<<"Computer name: "<<system<<"\n";

return buff.str();

}

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t bin\_count)

{

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 5;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 40;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 20;

const auto INDENT = 5;

const auto WIDTH\_LINE = 3;

size\_t maxbin=bins[0];

for(size\_t j:bins)

if(maxbin<j)

maxbin=j;

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

double x1, x2, y1, y2;

x1=5;

y1=5;

double koeff;

if(((IMAGE\_WIDTH-x1-WIDTH\_LINE-TEXT\_WIDTH-INDENT-WIDTH\_LINE-WIDTH\_LINE)/BLOCK\_WIDTH)<maxbin)

koeff=(IMAGE\_WIDTH-x1-WIDTH\_LINE-TEXT\_WIDTH-INDENT-WIDTH\_LINE-WIDTH\_LINE)/maxbin;

else

koeff=BLOCK\_WIDTH;

x2=x1+WIDTH\_LINE+TEXT\_WIDTH+koeff\*maxbin+INDENT+WIDTH\_LINE;

y2=y1+WIDTH\_LINE+INDENT+BIN\_HEIGHT\*bin\_count+INDENT+WIDTH\_LINE;

double top = y1+WIDTH\_LINE+INDENT;

svg\_line(x1,y1,x2,y1, WIDTH\_LINE);

svg\_line(x1,y2,x2,y2, WIDTH\_LINE);

svg\_line(x1,y1,x1,y2, WIDTH\_LINE);

svg\_line(x2,y1,x2,y2, WIDTH\_LINE);

for (size\_t bin : bins)

{

const double bin\_width = koeff \* bin;

svg\_text(x1+WIDTH\_LINE+TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));

svg\_rect(x1+WIDTH\_LINE+TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "red", "red");

top += BIN\_HEIGHT;

}

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

svg\_end();

}

Histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

Histogram.cpp

#include "histogram.h"

#include <vector>

using namespace std;

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

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

return;

}