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

# Макаров К.О.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

# Задание 4.1 :

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.

# Задание 4.2 :

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 9**

С помощью функции curl\_version\_info() печатайте на стандартный вывод ошибок версию cURL и версию OpenSSL в строковом виде.

**Описание логики решения индивидуального задания:**

Чтобы получить версию cURL и OpenSSL 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 ( -> ).

**Ссылка на репозитарий:** [**https://github.com/MakarovKO/lab3.3.git**](https://github.com/MakarovKO/lab3.3.git)

**Код программы:**

***Main.cpp***

#include <iostream>

#include <curl/curl.h>

#include <vector>

#include <conio.h>

#include "histogram.h"

#include "svg.h"

#include <string>

#include <sstream>

#include <windows.h>

#include <cstdio>

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);

for (size\_t i=0; i<count; i++)

{

in>>result[i];

}

return result;

}

vector<size\_t>

make\_histogram(struct Input data)

{

double max=0, min=0;

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

vector<size\_t>bins(data.bin\_count,0); /\* массив из индексов корзин \*/

for(double x : data.numbers) /\* мы каждому x присваиваем последовательно каждый элемент массива "numbers" \*/

{

size\_t bin\_index=(size\_t)((x-min)\*data.bin\_count/(max-min));

if (bin\_index==data.bin\_count) /\* индекс корзины "bin\_index"\*/

{

bin\_index=bin\_index-1; /\* нумирация идет с 0\*/

}

bins[bin\_index]++;

}

return bins;

}

Input

read\_input(istream& in, bool promt)

{

size\_t number\_count;

Input data;

if(promt)

{

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;

}

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 0;

}

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);

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;

cout <<curl\_version\_info(CURLVERSION\_NOW)<<"\n";

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

cout <<asd->ssl\_version<<"\n";

exit(1);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

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

{

Input input;

if (argc > 1)

{

input = download(argv[1]);

}

else

{

input = read\_input(cin, true);

}

const auto bins=make\_histogram(input);

show\_histogram\_svg(bins);

return 0;

}

***Svg.h***

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

#include <windows.h>

using namespace std;

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

#endif // SVG\_H\_INCLUDED

***Svg.cpp***

#include "svg.h"

#include "block\_width.h"

#include <iostream>

#include <string>

#include <sstream>

#include <windows.h>

using namespace std;

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='red' fill='blue'/>";

}

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

make\_info\_text() {

stringstream buffer;

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

//printf("Hexadecimal version %x\n",GetVersion());

DWORD info=GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = info & mask;

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

DWORD platform = info >> 16;

DWORD mask\_minor = 0x000000ff;

DWORD mask\_major = 0x0000ff00;

DWORD version\_minor = info & mask\_minor;

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

DWORD version\_major1 = info & mask\_major;

DWORD version\_major = version\_major1 >> 8;

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

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

{

DWORD build = platform;

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

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

}

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

DWORD Size = sizeof(system);

GetComputerNameA(system, &Size);

//printf("System: %s", system);

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

return buffer.str();

}

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

{

double BLOCK\_WIDTH;

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;

int flag=1;

input\_BLOCK\_WIDTH(BLOCK\_WIDTH,flag);

size\_t MAX\_ASTERISK=(size\_t)(350/BLOCK\_WIDTH);

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

unsigned max\_count=0;

for (size\_t b : bins) /\* мы присваиваем значение количества элементов \*/

{

if(max\_count<b)

max\_count=b; /\* здесь мы ищем максимальное количество элементов в массиве\*/

}

double top = 0;

for (size\_t bin : bins)

{

size\_t height=35;

if(max\_count>MAX\_ASTERISK) /\* Если количество будет больше 35, то уменьшаем масштаб\*/

height=MAX\_ASTERISK\*((static\_cast<double>(bin))/max\_count); /\* это для того,чтобы было дробное число , получаем, используя "static\_cast<double>"\*/

else

{

height=bin;

}

height = BLOCK\_WIDTH \* height;

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

svg\_rect(TEXT\_WIDTH, top, height, BIN\_HEIGHT, "red", "#aaffaa");

top += BIN\_HEIGHT;

}

cout << "<text x='" << TEXT\_LEFT << "' y='"<<top+BIN\_HEIGHT<<"'>"<<make\_info\_text()<<"</text>";

svg\_end();

}

***Histogram.h***

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

***Histogram.cpp***

#include "histogram.h"

#include <iostream>

using namespace std;

#include <vector>

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

{

unsigned int n=numbers.size();

if(n!=0)

{

min = numbers[0];

max = numbers[0];

for(double x: numbers)

{

if(min>x)

min=x;

if(max<x)

max=x;

}

}

}

***Block\_width.h***

#ifndef BLOCK\_WIDTH\_H\_INCLUDED

#define BLOCK\_WIDTH\_H\_INCLUDED

#include <vector>

using namespace std;

#include <iostream>

void input\_BLOCK\_WIDTH(double &BLOCK\_WIDTH,int &flag);

#endif // BLOCK\_WIDTH\_H\_INCLUDED

***Block\_width.cpp***

#include "block\_width.h"

#include <iostream>

using namespace std;

#include <vector>

void input\_BLOCK\_WIDTH(double &BLOCK\_WIDTH, int &flag)

{

while (flag==1)

{

cerr<<"Please enter the width of one block of the histogram\n";

cin>>BLOCK\_WIDTH;

if (BLOCK\_WIDTH<3 || BLOCK\_WIDTH>30)

cerr << "The width of one block of the histogram cannot be more then 30 px or less then 3 px\n";

if (BLOCK\_WIDTH>=3 && BLOCK\_WIDTH<=30)

flag=0;

}

}