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 программного обеспечения систем управления»**

# «Библиотеки и низкоуровневые операции»

Выполнила студентка группы А-02-20

Андриянова Кристина Борисовна

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

Ссылка не репозитарий: https://github.com/AndrianovaKB/lab03/tree/double

**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.
4. Уметь применять побитовые операции для типовых сценариев.
5. Уметь работать с API, принимающими указатели, в том числе строки C.
6. Знать характерные особенности документации на API библиотек.

# Задание:

# Часть 1. Библиотеки

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.

# Часть 2. Низкоуровневые операции

# 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

**Индивидуальное задание**

**Вариант 6**

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок суммарное время (total), затраченное на загрузку файла по сети.

Решение:

Для решения задачи я использовала функцию CURLINFO\_TOTAL\_TIME пример использования функции посмотрела в интернете и вставила в функцию Input download.

**Исходный код:**

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <math.h>

#include <curl/curl.h>

#include "histogram.h"

#include "histogram.cpp"

#include "svg.h"

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

using namespace std;

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 promt) {

Input data;

if(promt)

cerr << "Enter number count: ";

size\_t number\_count;

in >> number\_count;

if(promt)

cerr << "Enter numbers: ";

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

if(promt)

cerr << "Enter bin count: ";

in >> data.bin\_count;

return data;

}

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

{

double min;

double max;

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

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 show\_histogram\_text(vector <size\_t> bins, const size\_t MAX\_ASTERISK)

{

size\_t bin\_count = bins.size();

size\_t Max = bins[0];

for (size\_t j=1; j<bin\_count; j++)

{

if (bins[j]>Max)

Max = bins[j];

}

for (size\_t j=0; j<bin\_count; j++)

{

if(bins[j]>=100)

cout<<bins[j];

if (bins[j]<10)

cout <<" "<< bins[j];

else if (bins[j]<100)

cout<< " "<< bins[j] ;

cout << "|";

if(Max>MAX\_ASTERISK)

{

size\_t height = MAX\_ASTERISK \* (static\_cast<double>(bins[j]) / Max);

for (size\_t a=0; a<height; a++)

{

cout << "\*";

}

}

else

for(size\_t a=0; a<bins[j]; a++)

cout << "\*";

cout << endl;

}

}

size\_t

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

size\_t data\_size = item\_size \* item\_count;

stringstream\* buffer = reinterpret\_cast<stringstream\*>(ctx);

buffer->write(reinterpret\_cast<const char\*>(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);

curl\_easy\_cleanup(curl);

if(res)

{

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

exit(1);

}

else

{

double curtime = 0;

curtime= curl\_easy\_getinfo(curl, CURLINFO\_TOTAL\_TIME, &curtime);

if(!res)

{

cout << "total time spent downloading the file:" << curtime << "\n";

}

}

}

return read\_input(buffer, false);

}

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

{

Input input;

size\_t number\_count;

if (argc > 1)

{

input = download(argv[1]);

}

else

{

input = read\_input(cin, true);

}

size\_t bin\_count;

const auto bins = make\_histogram(input);

string stroke;

string fill;

show\_histogram\_svg(bins, bin\_count, number\_count,stroke,fill);

return 0;

}

**histogram.cpp**

#include "histogram.h"

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

{

if (numbers.size() == 0)

return;

else

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}**histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

#endif // HISTOGRAM\_H\_INCLUDED

**svg.cpp**

#include "svg.h"

#include <string>

#include <sstream>

#include <windows.h>

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

void

svg\_begin(double width, double height)

{

//cout <<"width="<<width<<endl;

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\_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<<"'/>";

}

void

svg\_end()

{

cout << "</svg>\n";

}

/\*double input\_image\_width(size\_t number\_count, double BLOCK\_WIDTH, double& IMAGE\_WIDTH,istream& in)

{

//double IMAGE\_WIDTH;

cerr << "Enter IMAGE\_WIDTH:";

in >> IMAGE\_WIDTH;

image\_width (number\_count, BLOCK\_WIDTH, IMAGE\_WIDTH, cin);

return IMAGE\_WIDTH;

}\*/

double image\_width (size\_t number\_count, double BLOCK\_WIDTH, istream& in)

{

int image\_width;

in >> image\_width;

while(image\_width < 70 || image\_width > 800 || image\_width < 1/3\*(number\_count\*BLOCK\_WIDTH))

{

cerr << "invalid input, please enter again" << endl;

cerr << "Enter SCREEN\_WIDTH:";

in >> image\_width;

}

return image\_width;

}

string

make\_info\_text() {

stringstream buffer;

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD mask\_2 = 0x000000ff;

DWORD platform = info >> 16;

DWORD version = info & mask;

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

{

DWORD version\_major = version & mask\_2;

DWORD version\_minor = version >> 8;

DWORD build = platform;

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

}

DWORD size = MAX\_COMPUTERNAME\_LENGTH+1;

char computer\_name[size];

GetComputerNameA(computer\_name, &size);

buffer << "Computer name: " << computer\_name;

return buffer.str();

}

void show\_histogram\_svg(const vector <size\_t>& bins, double bin\_count, size\_t number\_count, string& stroke, string& fill)

{

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 IMAGE\_WIDTH = image\_width(number\_count, BLOCK\_WIDTH, cin);

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

double top = 0;

size\_t Max = 0;

for (size\_t count : bins)

{

if (count > Max)

Max = count;

}

const bool scaling\_up = (Max\*BLOCK\_WIDTH) > (IMAGE\_WIDTH-TEXT\_WIDTH);

if (scaling\_up)

{

const double scaling = (double)(IMAGE\_WIDTH-TEXT\_WIDTH) / (Max\*BLOCK\_WIDTH);

cerr << "scaling="<<scaling;

cerr << "MAX\_ASTERISK="<<MAX\_ASTERISK;

cerr << "Max="<<Max;

for (size\_t bin : bins)

{

cerr <<"bin="<<bin<<"\n";

auto height = (size\_t)(bin \* scaling);

cerr << "height=" << height<<"\n";

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

cerr << "bin\_width=" << bin\_width<<"\n";

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

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

top += BIN\_HEIGHT;

}

}

else

{

for (size\_t bin : bins)

{

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

//cerr << "bin\_width=" << bin\_width;

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "blue", "#ffeeee");//stroke, fill);

top += BIN\_HEIGHT;

}

}

svg\_text(0, top + TEXT\_BASELINE, make\_info\_text());

svg\_end();

}

**svg.h**

##ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <windows.h>

using namespace std;

void

svg\_begin(double width, double height);

void

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

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

void

svg\_end();

//double input\_image\_width(size\_t number\_count, double BLOCK\_WIDTH, istream& in);

double image\_width (size\_t number\_count, double BLOCK\_WIDTH, istream& in);

string make\_info\_text();

void show\_histogram\_svg(const vector <size\_t>& bins, double bin\_count, size\_t number\_count, string& stroke, string& fill);

#endif // SVG\_H\_INCLUDED

**test.cpp**

#include "histogram.h"

#include "svg.h"

#include <cassert>

#include <math.h>

#include <sstream>

#include <string>

#include <iostream>

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\_similar()

{

double min = 0;

double max = 0;

find\_minmax({3, 3, 3}, min, max);

assert(min == 3);

assert(max == 3);

}

void

test\_single()

{

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);

}

//модульные тесты

void

test ()

{

//double IMAGE\_WIDTH=400;

stringstream iw ("400");

double res = image\_width (5, 10, iw);

assert(res == 400);

}

void

test1 ()

{

//double IMAGE\_WIDTH = 1000;

stringstream iw ("1000 350");

double res = image\_width (5, 10, iw);

assert(res == 350);

}

void

test2 ()

{

//double IMAGE\_WIDTH = 30;

stringstream iw ("30 750");

double res = image\_width (5, 10, iw);

assert(res == 750);

}

int

main1()

{

double IMAGE\_WIDTH;

test\_positive();

test\_negative();

test\_single();

test\_empty();

test();

test1();

test2();

}