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:*

# Студентка группы А-01-20

# Маврина А.А.

# *Проверили:*

# Мохов А. С

# Козлюк Д. А

**Москва 2021**

**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 10**

Добавьте программе опцию -verbose, при указании которой нужно включать отладочный вывод cURL с помощью CURLOPT\_VERBOSE. Опция может стоять до или после URL: lab03.exe -verbose http://... или lab04.exe http://... -verbose. Если встретился иной аргумент, начинающийся с дефиса, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Логика решения варианта:**

Была создана структура, состоящая из 3х полей, первые 2 отвечают за необходимость отладочного вывода и вывода подсказки для пользователя, 3е хранит в себе возможный URL страницы ,с которой будет производиться загрузка данных, для построения гистограммы.

Структура заполняется с помощью функции , в которую передаются аргументы программы. Изначально в функции всем логическим аргументам передается “false”, а затем идет перебор аргументов , и , если какой-то из них начинается с – (дефиса) ,то проверятся , являются ли остальные символы аргумента verbose с помощью функции strcmp (если нет , то выводится подсказка)

В теле основной программы тоже были сделаны изменения. Теперь, если поле, отвечающее за необходимость подсказки равно «true», то программа выводит подсказку и завершается досрочно. Если в параметры был передан URL, то программа вызывает видоизменённую версию функции download (в параметры добавлен bool verbose), иначе ввод идет с клавиатуры.

**Исходный код всех модулей**

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

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 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 bin count:";

}

in>>data.bin\_count;

return data;

}

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,bool verbose) {

stringstream buffer;

curl\_global\_init(CURL\_GLOBAL\_ALL);

CURL \*curl = curl\_easy\_init();

if(curl)

{

CURLcode res;

if (verbose)

{

curl\_easy\_setopt(curl, CURLOPT\_VERBOSE, 1L);

}

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)

{

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

exit(1);

}

}

curl\_easy\_cleanup(curl);

return read\_input(buffer, false);

}

Verb input\_with\_verb(int argc, char\* argv[])

{

Verb use;

use.verbose=false;

use.help=false;

use.urladress=0;

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

{

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

{

if (strcmp(argv[i], "-verbose") == 0)

{

use.verbose=true;

}

else

{

use.help=true;

}

}

else

{

use.urladress=argv[i];

}

}

return use;

}

int

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

Input input;

auto data=input\_with\_verb(argc,argv);

if (data.help)

{

cerr <<"\nHELP:\n \*For making input from web enter URL of page to parameters"<<

"\n \*For using CURLOPT\_VERBOSE mode enter ''-verbose''" << "\n";

exit(2);

}

if (data.urladress)

{

input=download(data.urladress,data.verbose);

}

else

{

input=read\_input(cin, true);

}

const auto bins=make\_histogram(input);

// Вывод данных

size\_t bin\_height;

show\_histogram\_svg(bins,bin\_height);

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;

}

}

}

vector<size\_t>

make\_histogram (const Input& data)

{

double min , max ;

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';

}

}

**histogram.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

struct Verb

{

bool verbose;

bool help;

char\* urladress;

};

void

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

vector<size\_t>

make\_histogram (const Input& data);

void

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

**Часть 4.2**

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <windows.h>

using namespace std;

vector<double>

input\_numbers(size\_t count)

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

int

main() {

// Ввод данных

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

cerr << "Enter numbers: ";

const auto numbers=input\_numbers(number\_count);

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

// Обработка данных

const auto bins=make\_histogram(numbers,bin\_count);

// Вывод данных

size\_t bin\_height;

show\_histogram\_svg(bins,bin\_height);

return 0;

}

**svg.cpp**

#include "svg.h"

string

make\_info\_text() {

stringstream buffer;

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD mask\_2 = 0x0000ff;

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<<")";

}

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

DWORD size = MAX\_COMPUTERNAME\_LENGTH+1;

GetComputerNameA(computer\_name, &size);

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

// TODO: получить версию системы, записать в буфер.

// TODO: получить имя компьютера, записать в буфер.

return buffer.str();

}

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)

{

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

{

if (bins.size() == 0)

{

return;

}

const auto IMAGE\_WIDTH = 500;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 15;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_HEIGHT = 30;

const auto BIN\_WIDTH = 30;

const auto TEXT\_TOP=10;

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

double left =0;

size\_t max\_count=bins[0];

for (size\_t count : bins)

{

if ( count >max\_count)

{

max\_count=count;

}

}

for (size\_t bin: bins)

{

const double scaling\_factor = (double)(IMAGE\_HEIGHT - TEXT\_HEIGHT-TEXT\_TOP) / max\_count;

bin\_height=(size\_t)(bin\*scaling\_factor);

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

svg\_rect(left, TEXT\_HEIGHT+TEXT\_TOP, BIN\_WIDTH, bin\_height);

left += BIN\_WIDTH;

}

svg\_text(0, TEXT\_TOP, make\_info\_text());

svg\_end();

}

**svg.h**

#pragma once

#include <vector>

#include <iostream>

#include <sstream>

#include <windows.h>

using namespace std;

string

make\_info\_text();

void

svg\_begin(double width, double height);

void

svg\_end();

void

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

void

svg\_rect(double x, double y, double width, double height, string stroke= "red" , string fill= "#ffeeee" );

void

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