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-01-20

# Болотникова 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 4**

Добавьте программе опцию -bins, которая позволяет задать количество корзин при запуске. Считывать его в read\_input() при этом не нужно. Например: lab03.exe -bins 3. Опция может стоять до или после URL: lab03.exe -bins 3 http://... или lab04.exe http://... -bins 3. Если после -bins не стоит числа, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Логика решения варианта:**

|  |
| --- |
| Input |
|  | f\_bins(istream& in,int argc, char\* argv[]){ |
|  | Input data; |
|  | string \_bins=argv[1]; |
|  | if(\_bins=="-bins" && argc>3) { |
|  | size\_t bin\_count; |
|  | data.bin\_count=atoi(argv[2]); |
|  | } |
|  | else if(argc<3){ |
|  | cout<<"You need to enter the quantity\n"; |
|  | } |
|  | return data; |
|  | } |

**Исходный код всех модулей**

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 3 - 1;

vector<double>

input\_numbers(istream& in, size\_t count, bool prompt)

{

vector<double> result(count);

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

{

if (prompt == true)

{

cerr << "Enter " << i + 1 << " number:\t";

}

in >> result[i];

}

return result;

}

Input

read\_input(istream& in, bool prompt)

{

Input data;

size\_t number\_count;

if (prompt == true)

{

cerr << "Enter number count: ";

}

in >> number\_count;

if (prompt == true)

{

cerr << "Enter numbers: ";

}

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

if (prompt == true)

{

cerr << "Enter quantity of bins: ";

}

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((const char\*) items, data\_size);

return data\_size;

}

Input

download(const string& address)

{

stringstream buffer;

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

if(curl)

{

CURLcode res;

curl\_easy\_setopt(curl, CURLOPT\_URL, address.c\_str());

/\* ask libcurl to show us the verbose output \*/

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

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_data);

curl\_easy\_setopt(curl, CURLOPT\_SSL\_VERIFYPEER, FALSE);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, &buffer);

res = curl\_easy\_perform(curl);

curl\_easy\_cleanup(curl);

if (res != CURLE\_OK)

{

cerr << "error: " << curl\_easy\_strerror(res);

exit(1);

}

double namelookup;

res = curl\_easy\_getinfo(curl, CURLINFO\_NAMELOOKUP\_TIME, &namelookup);

if(CURLE\_OK == res)

{

cerr << "Name lookup time: " << namelookup << " s" << endl;

}

}

return read\_input(buffer, false);

}

void

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

{

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

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

{

if (bins[i] > max\_count)

{

max\_count = bins[i];

}

}

if (max\_count > MAX\_ASTERISK)

{

size\_t height;

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

{

height = MAX\_ASTERISK \* (static\_cast<double>(bins[i]) / max\_count);

if (bins[i] < 100)

cout << " ";

if (bins[i] < 10)

cout << " ";

cout << bins[i] << "|";

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

{

cout << "\*";

}

cout << endl;

}

}

else

{

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

{

if (bins[i] < 100)

cout << " ";

if (bins[i] < 10)

cout << " ";

cout << bins[i] << "|";

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

{

cout << "\*";

}

cout << endl;

}

}

}

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);

}

**histogram.cpp**

#include <vector>

#include "histogram.h"

const vector<size\_t>

make\_histogram(Input input)

{

double min;

double max;

vector<size\_t> bins(input.bin\_count);

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

double bin\_size = (max - min) / bins.size();

for (size\_t i = 0; i < input.numbers.size(); i++)

{

bool found = false;

for (size\_t j = 0; (j < bins.size() - 1) && !found; j++)

{

auto low = min + j \* bin\_size;

auto high = min + (j + 1) \* bin\_size;

if ((input.numbers[i] >= low) && (input.numbers[i] < high))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[bins.size() - 1]++;

}

}

return bins;

}

void

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

{

if (numbers.size() == 0) return;

min = numbers[0];

max = numbers[0];

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

{

if (numbers[i] < min)

{

min = numbers[i];

}

if (numbers[i] > max)

{

max = numbers[i];

}

}

}

**histogram.h**

#pragma once

#include <vector>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

};

const vector <size\_t> make\_histogram (Input input);

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

**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 = 0xff;

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);