Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Отчёт по лабораторной работе №4**

**Вариант 9**

Выполнил Котухов М.Д.

Группа А-01-20

Проверил:

Мохов А.С.

Козлюк Д.А.

Москва 2021 НИУ МЭИ

**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 v6.2 (build 9200) | Computer name: HP**

**Индивидуальное задание**

**Вариант 9**

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

**Логика решения варианта:**

С помощью фунции

curl\_version\_info\_data\* inform= curl\_version\_info(CURLVERSION\_NOW)

мы вызываем структуру, в которой хранятся необходимые нам значения версий cURL и OpenSSl

Далее нам необходимо лишь обратиться к этим значениям, хранящимся в переменных: inform->version, inform->ssl\_version.

В итоговом виде наша строчка под гистограммой будет выглядеть следующим образом:

**Windows v6.2 (build 9200) | Computer name: HP| cURL Version: 7.77.0 | Open SSL Version: OpenSSL/1.1.1k (Schannel)**

**Исходный код всех модулей:**

**main.cpp**

**=================================**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <windows.h>

#include <cstdio>

using namespace std;

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

{

string info = make\_info\_text();

Input input;

if (argc > 1)

{

input = download(argv[1]);

}

else

{

input = read\_input(cin, true);

}

const auto bins = make\_histogram(input);

size\_t F;

cerr << "Enter BLOCK\_WIDTH behind 3 and 30" << endl;

cin >> F;

show\_histogram\_svg(bins, F);

return 0;

}

**==================================histogram.cpp**

**==================================**

#include "histogram.h"

#include <curl/curl.h>

Input read\_input(istream& in /\* произвольный поток ввода \*/, bool promt) // numbers и bin\_count должны храниться и передаваться вместе

{

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 column count: ";

}

in >> data.bin\_count;

return data;

}

size\_t write\_data(void\* items /\* указатель на принятые данные \*/, size\_t item\_size /\* размер одного блока данных \*/, size\_t item\_count /\* количество блоков данных \*/, void\* ctx /\* \*/) // call-back функция

{

size\_t data\_size = item\_size \* item\_count; // вычисление количества принятых байтов

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

(\*buffer).write(reinterpret\_cast<const char\*>(items), data\_size); // добавляем данные в буфер // buffer -> write(...)

return data\_size;

}

Input download(const string& address) // принимает строку с адресом файла

{

stringstream buffer; // поток который читает инф. из буффера

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

CURL\* curl = curl\_easy\_init(); //функция создания дескриптора

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)

{

cerr << curl\_easy\_strerror(res) /\* принимает код ошибки и возвращает строку с текстом ошибки \*/ << endl;

exit(1);

}

}

curl\_easy\_cleanup(curl);

return read\_input(buffer, false);

}

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;

}

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

{

if (numbers.size()==0)

{

min = 0;

max = 0;

return;

}

else

{

min = numbers[0];

max = numbers[0];

for (size\_t i = 1; i < numbers.size(); i++)

{

if (numbers[i] < min)

{

min = numbers[i];

}

if (numbers[i] > max)

{

max = numbers[i];

}

}

}

return;

}

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

{

double min, max;

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

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

for (double number : input.numbers)

{

size\_t bin = (size\_t)((number - min) / (max - min) \* input.bin\_count);

if (bin == input.bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

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;

}

**==================================**

**histogram.h**

#pragma once

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

#include <curl/curl.h>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

};

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

Input download(const string& address);

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

void find\_minmax(vector<double> numbers, double& min, double& max); // нахождение мин. и макс. значения

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

void show\_histogram\_text(vector<size\_t> bins); // отображает гистограмму в текстовом виде

Input read\_input(istream& in, bool promt);

**==================================svg.cpp**

#include "svg.h"

#include <windows.h>

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 fil)

{

cout << "<rect x='" << x << "' y='" << y << "' width='" << width << "' height='" << height

<< "' stroke='" << stroke << "' fill='" << fil << "' />";

}

string make\_info\_text()

{

stringstream buffer;

DWORD /\* тип возврщаемого значения double word 4 байта \*/ info = GetVersion(); // команда позволяет получить номер версии Windows

DWORD mask = 0x0000ffff; // маска для получения версии

DWORD version = info & mask;

DWORD platform = info >> 16; // сдвиг вправо на 16 бит

DWORD mask\_major = 0x0000ff;

DWORD version\_major = version & mask\_major;

DWORD version\_minor = version >> 8;

if ((info & 0x80000000) == 0) // проверка что старший бит = 0

{

DWORD build = platform;

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

}

else printf("minor\_bit = %u.\n", 1);

char system\_dir[MAX\_PATH];

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

DWORD size = MAX\_COMPUTERNAME\_LENGTH + 1;

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

// curl\_version\_info\_data \*c = curl\_version\_info.ssl\_version;

// const char s[100] = d.ssl\_version;

GetComputerNameA(computer\_name, &size); // получение имени компьютера

buffer << /\* поток, поэтому << \*/ "Computer name: " << computer\_name;

buffer << "| cURL Version: " << inform->version << " |" << " Open SSL Version: " << inform->ssl\_version;

return buffer.str();

}

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

{

const auto IMAGE\_WIDTH = 4000;

const auto IMAGE\_HEIGHT = 3000;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = SHIRINA\_BLOCKA;

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

double top = 0;

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

for (size\_t bin : bins)

{

if (bin > max\_count)

{

max\_count = bin;

}

}

const size\_t SCREEN\_WIDTH = 100;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 50 - 20;

for (size\_t bin : bins)

{

const bool scalling\_needed = max\_count < MAX\_ASTERISK;

size\_t bin\_koeff = bin;

if (scalling\_needed)

{

const double koeff = (double)MAX\_ASTERISK / max\_count;

bin\_koeff = (size\_t)(bin \* koeff);

}

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

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

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

top += BIN\_HEIGHT;

}

string Info=make\_info\_text();

svg\_text (TEXT\_LEFT, top + TEXT\_BASELINE,Info);

svg\_end();

}

**==================================svg.h**

#pragma once

#include <iostream>

#include <vector>

#include <sstream>

#include <windows.h>

#include <curl/curl.h>

using namespace std;

void svg\_begin(double width, double height); // вывод загаловка

void svg\_end(); // окончание cvg

void svg\_text(double left, double baseline, string text); // вывод подписей к столбцам

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fil = "black"); // вывод прямоугольника в svg

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t SHIRINA\_BLOCKA); // вывод гистограммы в svg

string make\_info\_text(); // строка под гистограммой svg

**==================================**

**Код скопирован в репозитарий:**

[**https://github.com/kotukhov/cs-lab04**](https://github.com/kotukhov/cs-lab04)