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»

**Выполнила** студентка группы А-01-19

Рымарева Т. А.

**Проверили**

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 15**

Установите callback-функцию CURLOPT\_HEADERDATA, откуда печатайте полученные данные на стандартный вывод ошибок.

Для работы с данной функцией изучила необходимые материалы.

Источник: <https://curl.haxx.se/libcurl/c/CURLOPT_HEADERDATA.html>  
дополнение <https://curl.haxx.se/libcurl/c/CURLOPT_HEADERFUNCTION.html>

**Логика решения**

* Создадим функцию header\_callback типа size\_t, которая будет возвращать  размер принятых данных в байтах, а также буфер buffer\_1
* С помощью CURLOPT\_HEADERFUNCTION в curl\_easy\_setopt() вызовем функцию header\_callback
* В curl\_easy\_setopt() вызовем CURLOPT\_HEADERDATA, данные будут записаны в buffer\_1
* В функции download осуществим вывод данных на стандартный вывод ошибок (cerr)

**Ссылка на репозитарий -** [**https://github.com/RymarevaTA/cs-lab03**](https://github.com/RymarevaTA/cs-lab03)

**Код программы**

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

#include <windows.h>

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 column count: ";

}

in >> data.bin\_count;

if (prompt)

{

cerr << "Enter interval: ";

}

in >> data.interval;

return data;

}

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;

}

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;

}

size\_t header\_callback(void\* items, size\_t item\_size, size\_t item\_count, void\* ctx)

{

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

stringstream\* buffer\_1 = reinterpret\_cast<stringstream\*>(ctx);

(\*buffer\_1).write(reinterpret\_cast<const char\*>(items), data\_size);

return data\_size;

}

Input download(const string& address)

{

stringstream buffer;

stringstream buffer\_1;

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);

curl\_easy\_setopt(curl, CURLOPT\_HEADERFUNCTION, header\_callback);

curl\_easy\_setopt(curl, CURLOPT\_HEADERDATA, &buffer\_1);

res = curl\_easy\_perform(curl);

string str;

while (buffer\_1) {

buffer\_1 >> str;

cerr << str << " ";

}

if (res)

{

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

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, input.bin\_count, input.interval);

return 0;

}

**svg.cpp**

#include "svg.h"

#include "histogram.h"

#include <vector>

#include <string>

#include <iostream>

#include <windows.h>

#include <cstdio>

using namespace std;

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 << "' />";

}

size\_t calculation\_star\_factor (size\_t MAX\_WIDTH, size\_t max\_count)

{

size\_t star\_factor=MAX\_WIDTH/max\_count;

return (star\_factor);

}

size\_t calculation\_interval\_width (size\_t star\_factor, size\_t interval)

{

size\_t interval\_width=star\_factor\*interval;

return(interval\_width);

}

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

{

const auto IMAGE\_WIDTH = 550;

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;

size\_t star\_factor;

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

double top = 0;

size\_t max\_count=0;

for (size\_t bin : bins)

{

if(max\_count<bin)

max\_count=bin;

}

double MAX\_WIDTH=IMAGE\_WIDTH-4.5\*TEXT\_WIDTH;

size\_t interval\_count;

if (max\_count%interval!=0)

interval\_count=max\_count/interval+1;

else

interval\_count=max\_count/interval;

star\_factor=calculation\_star\_factor(MAX\_WIDTH, max\_count);

size\_t interval\_width=calculation\_interval\_width(star\_factor,interval);

for (size\_t bin : bins)

{

const double bin\_width = star\_factor \* bin;

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"black","green");

top += BIN\_HEIGHT;

}

size\_t otstup= (double)bin\_count\*BIN\_HEIGHT+15;

if (interval<4 || interval>9)

svg\_text(TEXT\_WIDTH, otstup, "ERROR!");

else

{

size\_t abscissa=TEXT\_WIDTH;

for (size\_t i=1; i<=interval\_count; i++)

{

svg\_rect(abscissa, otstup, interval\_width, BIN\_HEIGHT, "black", "#90EE90");

abscissa+=interval\_width;

}

svg\_text(TEXT\_WIDTH, otstup+50, "0");

svg\_text(TEXT\_WIDTH+interval\_width, otstup+50, to\_string(interval));

svg\_text(TEXT\_WIDTH+interval\_count\*interval\_width, otstup+50, to\_string(interval\*interval\_count));

}

svg\_text(TEXT\_WIDTH,otstup+80,make\_info\_text());

svg\_end();

}

**histogram.cpp**

#include "histogram.h"

#include <string>

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

{

min = numbers[0];

max = numbers[0];

for (auto number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

return;

}

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;

}

string make\_info\_text()

{

stringstream buffer;

DWORD WINAPI GetVersion();

DWORD mask = 0x0000ffff;

DWORD mask\_2 = 0x000000ff;

DWORD info = GetVersion();

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & mask\_2;

DWORD version\_minor = version >>8;

if ((version & 0x40000000) == 0)

{

printf("equal to zero\n");

}

DWORD build = platform;

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

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

DWORD size = sizeof(computer\_name);

GetComputerNameA(computer\_name, &size);

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

return buffer.str(); }

**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 interval;

};

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

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

string make\_info\_text();

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

using namespace std;

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, string fill);

size\_t calculation\_star\_factor (size\_t MAX\_WIDTH, size\_t max\_count);

size\_t calculation\_interval\_width (size\_t star\_factor, size\_t interval);

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

#endif