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-01-19

Кирьянов Б.Ю.

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 6**

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок размер файла, загружаемого по сети.

**Ссылка на репозитарий**

<https://github.com/KiryanovBY/cs-lab03>

**Код программы**

**main.cpp**

#include <iostream>

#include <sstream>

#include <vector>

#include <cmath>

#include <string>

#include "find\_minmax.h"

#include "input\_numbers.h"

#include "make\_histogram.h"

#include "show\_histogram\_text.h"

#include "svg\_begin.h"

#include "svg\_end.h"

#include "svg\_text.h"

#include "svg\_rect.h"

#include "show\_histogram\_svg.h"

#include <curl/curl.h>

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

/\*struct Input {

vector<double> numbers;

size\_t bin\_count;

};\*/

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

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

const char\* Items = reinterpret\_cast<const char\*>(items);

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

buffer->write(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);

if(res){

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

exit(1);

}

double dl;

res = curl\_easy\_getinfo(curl, CURLINFO\_SIZE\_DOWNLOAD, &dl);

if(!res) cerr<< "Downloaded "<<dl<<" bytes" << endl;

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);

return 0;

}

**make\_histogram.h**

#pragma once

#include <vector>

#include "find\_minmax.h"

#include "input\_numbers.h"

using namespace std;

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

|  |
| --- |
|  |

**make\_histogram.cpp**

#include "make\_histogram.h"

vector<size\_t> make\_histogram(Input Data)

//vector<size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count)

{

double min, max;

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

vector<size\_t>count(Data.bin\_count, 0);

for (double x : Data.numbers)

{

size\_t bin\_index = (size\_t)(x - min) \* (Data.bin\_count) / (max - min);

if (bin\_index == Data.bin\_count)

{

bin\_index--;

}

count[bin\_index]++;

}

return count;

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**show\_histogram\_svg.h**

#pragma once

#include <vector>

#include "svg\_begin.h"

#include "svg\_end.h"

#include "svg\_text.h"

#include "svg\_rect.h"

#include <string>

#include <sstream>

#include <cstdio>

#include <windows.h>

using namespace std;

string make\_info\_text();

void show\_histogram\_svg(const vector<size\_t>&);

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |

**show\_histogram\_svg.cpp**

#include "show\_histogram\_svg.h"

const auto IMAGE\_WIDTH = 400;

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;

string make\_info\_text() {

stringstream buffer;

DWORD info = GetVersion();

DWORD mask = 0x0000ff00;

int platform = info >> 16;

int version\_major = info & (mask>>8);

int version\_minor = (info & (mask))>>8;

buffer << "<tspan>";

buffer << "Windows v"<<version\_major<<"."<<version\_minor;

//printf("Windows v%d.%d",version\_major,version\_minor);

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

int build = platform;

buffer<<" (build "<<build<<")";

//printf(" (build %d)",build);

}

buffer<<"\n </tspan> \n <tspan x='"<<TEXT\_LEFT<<"' dy='1em'>";

//printf("\n");

char ComputerName[MAX\_COMPUTERNAME\_LENGTH+1];

DWORD mcl = MAX\_COMPUTERNAME\_LENGTH;

GetComputerNameA(ComputerName,&mcl);

//printf("%s\n",ComputerName);

buffer<<"Computer name: "<<ComputerName<<"\n </tspan> \n";

return buffer.str();

}

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

{

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

double top = 0;

for (size\_t bin : bins)

{

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

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

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

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

top += BIN\_HEIGHT;

}

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

svg\_end();

}

**show\_histogram\_text.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

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

**show\_histogram\_text.cpp**

#include "show\_histogram\_text.h"

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

{

for (size\_t bin : bins)

{

if (bin < 10)

cout << " ";

else if (bin < 100)

cout << " ";

cout << bin << "|";

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

{

cout << "\*";

}

cout << endl;

}

}

**svg\_begin.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_begin(double, double);

**svg\_begin.cpp**

#include "svg\_begin.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";

}

**svg\_end.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_end();

**svg\_end.cpp**

#include "svg\_end.h"

void svg\_end()

{

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

}

**svg\_rect.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_rect(double, double, double, double, string stroke = "black", string fill = "black");

**svg\_rect.cpp**

#include "svg\_rect.h"

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 << "' />\n";

}

**svg\_text.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_text(double, double, string);

**svg\_text.cpp**

#include "svg\_text.h"

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

{

cout << "<text x='" << left << "' y='" << baseline << "'>" << text << "</text>\n";

}

**find\_minmax.h**

#pragma once

#include <vector>

using namespace std;

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

**find\_minmax.cpp**

#include "find\_minmax.h"

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

{

if (numbers.size() > 0) {

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (min > x)

min = x;

if (max < x)

max = x;

}

}

}

**input\_numbers.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

vector<double> input\_numbers(istream&, size\_t);

Input read\_input(istream&, bool);

**input\_numbers.cpp**

#include "input\_numbers.h"

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;

}