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-02-19

# Сметанин А.С.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 15.06.2020

#### **<https://github.com/SmetaninAS/lab03/tree/lab4>**

#### Вариант 14

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок размер запроса, который cURL отправила серверу (request size).

Нужно было печатать размер запроса cURL с помощью функции curl\_easy\_getinfo() .

**Main**

|  |
| --- |
| #include <curl/curl.h> |
| #include <iostream> | #include <iostream> |
| #include <sstream> | #include <sstream> |
| #include <windows.h> | #include <windows.h> |
| #include <curl/curl.h> | #include <curl/curl.h> |
| #include <iostream> | #include <iostream> |
| #include <sstream> | #include <sstream> |
| #include <tchar.h> | #include <tchar.h> |
| #include <string> | #include <string> |
| #include <vector> | #include <vector> |
| #include <iomanip> | #include <iomanip> |
| #include "histogram.h" | #include "histogram.h" |
| #include "svg.h" | #include "svg.h" |
| #define INFO\_BUFFER\_SIZE 32767 | #define INFO\_BUFFER\_SIZE 32767 |
| using namespace std; | using namespace std; |
|  |  |
| vector<double> input\_numbers(istream& in,size\_t count) | vector<double> input\_numbers(istream& in,size\_t count) |
| { | { |
| vector<double> result(count); | vector<double> result(count); |
| for (size\_t i = 0; i < count; i++) { | for (size\_t i = 0; i < count; i++) { |
| in >> result[i]; | in >> result[i]; |
| } | } |
| return result; | return result; |
| } | } |
| Input read\_input(istream& in,bool prompt) | Input read\_input(istream& in,bool prompt) |
| { | { |
| Input data; | Input data; |
| if (prompt==true) | if (prompt==true) |
| cerr << "Enter number count: "; | cerr << "Enter number count: "; |
| size\_t number\_count; | size\_t number\_count; |
| in >> number\_count; | in >> number\_count; |
| if (prompt==true) | if (prompt==true) |
| cerr << "Enter numbers: "; | cerr << "Enter numbers: "; |
| data.numbers = input\_numbers(in, number\_count); | data.numbers = input\_numbers(in, number\_count); |
| size\_t bin\_count; | size\_t bin\_count; |
| if (prompt==true) | if (prompt==true) |
| cerr << "Enter column count: "; | cerr << "Enter column count: "; |
| in >> data.bin\_count; | in >> data.bin\_count; |
| cerr << "enter length\_ch"; | cerr << "enter length\_ch"; |
| in >> data.length\_ch; | in >> data.length\_ch; |
| cerr << "enter length\_pr"; | cerr << "enter length\_pr"; |
| in >> data.length\_pr; | in >> data.length\_pr; |
| return data; | return data; |
| } | } |
|  |  |
| void show\_histogram\_text(const vector<size\_t> &bins) | void show\_histogram\_text(const vector<size\_t> &bins) |
| { | { |
|  |  |
| const size\_t SCREEN\_WIDTH = 80; | const size\_t SCREEN\_WIDTH = 80; |
| const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1; | const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1; |
|  |  |
| size\_t max\_count = 0; | size\_t max\_count = 0; |
| for (size\_t count : bins) { | for (size\_t count : bins) { |
| if (count > max\_count) { | if (count > max\_count) { |
| max\_count = count; | max\_count = count; |
| } | } |
| } | } |
| const bool scaling\_needed = max\_count > MAX\_ASTERISK; | const bool scaling\_needed = max\_count > MAX\_ASTERISK; |
|  |  |
| for (size\_t bin : bins) { | for (size\_t bin : bins) { |

**Histogram.cpp**

#include "histogram.h"

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

{

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(struct Input bindata)

{

double min;

double max;

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

vector<size\_t> bins(bindata.bin\_count,0);

for (double number : bindata.numbers)

{

size\_t bin;

bin = (number - min) / (max - min) \* bindata.bin\_count;

if (bin == bindata.bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}**Histogram h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

size\_t length\_ch; size\_t length\_pr;

};

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

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

#endif // HISTOGRAM\_H\_INCLUDED

**Svg.cpp**

#include <curl/curl.h>

#include <iostream>

#include <sstream>

#include <windows.h>

#include <curl/curl.h>

#include <iostream>

#include <sstream>

#include <tchar.h>

#include <string>

#include <vector>

#include <iomanip>

#include "histogram.h"

#include "svg.h"

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<<"'/>";

}

void svg\_line(double x1,double y1,double x2,double y2,string stroke\_1,struct Input bindata)

{

cout << "<line x1='"<<x1<<"' y1='"<<y1<<"' x2='"<<x2<<"' y2='"<<y2<<"' stroke='"<<stroke\_1<<"' stroke-dasharray='"<<bindata.length\_ch<<" "<<bindata.length\_pr<<"' />";

}

void show\_histogram\_svg(const vector<size\_t>& bins,struct Input bindata,string name)

{

const auto IMAGE\_WIDTH = 600;

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;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

double top = 0;

svg\_begin(600, 300);

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)

{

size\_t height = bin;

if (scaling\_needed)

{

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

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

}

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

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

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

if (top !=(bindata.bin\_count-1)\*BIN\_HEIGHT)

svg\_line(BLOCK\_WIDTH,top+BIN\_HEIGHT,IMAGE\_WIDTH,top+BIN\_HEIGHT,"red",bindata);

top += BIN\_HEIGHT;

}

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

svg\_end();

}