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 программного обеспечения систем управления»**

# «Библиотеки. Низкоуровневые операции»

Выполнил студент группы А-03-20

Иванов Олег Борисович

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

**Часть 1:**

**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.

**Часть 2:**

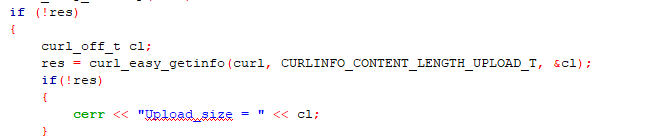
**Цель работы:**

1. Уметь применять побитовые операции для типовых сценариев.
2. Уметь работать с API, принимающими указатели, в том числе строки C.
3. Знать характерные особенности документации на API библиотек.

**Индивидуальное задание (вариант 6):**

С помощью функции curl\_easy\_getinfo () печатайте на стандартный вывод ошибок размер файла, загружаемого по сети. Оставить при этом 6 вариант из ЛР №3.

**Решение:**

В коде использовалась функция библиотеки curl если res отвечающий за код ошибки не имел какого-либо номера ошибки. Пример использования описан в методичке работы с curl.

**Код:**

**Main.cpp:**

#include <iostream>

#include <vector>

#include <sstream>

#include <string>

using namespace std;

#include "histogram.h"

#include <curl/curl.h>

Input read\_input (istream& in, bool prompt)

{

Input data;

if (prompt)

{

cerr << "Enter number count: ";

in >> data.number\_count;

cerr << "Enter numbers: ";

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

cerr << "Enter bin count: ";

in >> data.bin\_count;

cerr << "Enter color" << ": " << '\n';

data.color = colorsvg(in, data.bin\_count);

}

else

{

in >> data.number\_count;

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

in >> data.bin\_count;

data.color = colorsvg(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(reinterpret\_cast<const char\*>(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);

curl\_easy\_cleanup(curl);

if (!res)

{

curl\_off\_t cl;

res = curl\_easy\_getinfo(curl, CURLINFO\_CONTENT\_LENGTH\_UPLOAD\_T, &cl);

if(!res)

{

cerr << "Upload\_size = " << cl;

}

}

if(res)

{

cout << curl\_easy\_strerror(res);

exit(1);

}

}

return read\_input(buffer, false);

}

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

{

const size\_t SCREEN\_WIDTH = 80;

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

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);

getchar ();

return 0;

}

**histogram.cpp:**

#include "histogram.h"

#include <iostream>

using namespace std;

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

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

{

min = x;

}

if (x > max)

{

max = x;

}

}

}

vector <size\_t>

make\_histogram (Input data)

{

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

double min, max;

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

double bin\_size = (max - min) / data.bin\_count;

for (size\_t i = 0; i < data.number\_count; i++)

{

bool found = false;

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

{

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

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

if ((lo <= data.numbers[i]) && (data.numbers[i] < hi))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[data.bin\_count - 1]++;

}

}

return bins;

}

vector <string>

colorsvg (istream &in, size\_t bin\_count)

{

vector <string> result(bin\_count);

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

{

in >> result[i];

}

return result;

}

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;

}

**Код SVG.cpp:**

#include "histogram.h"

#include <iostream>

#include <windows.h>

#include <sstream>

#include <string>

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<<"' />";

}

string

make\_info\_text() {

stringstream buffer;

DWORD WINAPI GetVersion(void);

DWORD info = GetVersion();

DWORD mask = 0b00000000'00000000'11111111'11111111;

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD maska = 0b00000000'11111111;

if ((info & 0x40000000) == 0)

{

DWORD version\_major = version & maska;

DWORD version\_minor = version >> 8;

DWORD build = platform;

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

}

DWORD size = MAX\_COMPUTERNAME\_LENGTH + 1;

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

GetComputerNameA(name, &size);

buffer << "PC name: " << name << '\n';

return buffer.str();

}

void

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

{

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;

const auto MAX\_WIGTH = IMAGE\_WIDTH - TEXT\_WIDTH;

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

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

{

if (bins [i] > max\_bins)

max\_bins = bins[i] ;

}

max\_bins = max\_bins \* BLOCK\_WIDTH;

double height;

if (max\_bins <= MAX\_WIGTH)

height = 1;

else

height = (static\_cast < double > (MAX\_WIGTH) / max\_bins );

double top = 0;

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

int help\_color = 0;

for (size\_t bin : bins )

{

string COLOR = "#" + data.color[help\_color];

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

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

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

top += BIN\_HEIGHT;

help\_color ++;

}

svg\_text(1, top + TEXT\_BASELINE, make\_info\_text());

svg\_end();

}

**histogram.h:**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <string>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

size\_t number\_count;

vector<string> color;

};

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

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);

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

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

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

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

vector<string> colorsvg (istream &in, size\_t bin\_count);

#endif // HISTOGRAM\_H\_INCLUDED

**.gitignore:**

/curl/

/bin

/obj

**Ссылка на репозитарий:**

***https://github.com/Halssara/Lab\_3\_new***