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

Меденцев Алексей Николаевич

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

**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.

**Код файла main.cpp:**

#include <iostream>

#include <vector>

#include <curl/curl.h>

#include <sstream>

#include <string>

#include "histogram.h"

#include "svg.h"

#include <windows.h>

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

vector<size\_t>make\_histogram(Input& name)

{

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

double min, max;

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

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

for (size\_t i = 0; i < name.numbers.size(); i++)

{

bool found = false;

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

{

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

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

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

{

bins[j]++;

found = true;

}

}

if (!found)

{

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

}

}

return bins;

}

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

{

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

for (size\_t x : bins)

if (x > max\_count)

max\_count = x;

if (max\_count > 76)

{

for (size\_t x : bins)

{

if (x < 100)

cout << " ";

if (x < 10)

cout << " ";

cout << x << "|";

size\_t height = 76 \* (static\_cast<double> (x) / max\_count);

for (size\_t j = 0; j < height; j++)

cout << "\*";

cout << endl;

}

}

else

{

for (size\_t x : bins)

{

if (x < 100)

cout << " ";

if (x < 10)

cout << " ";

cout << x << "|";

for (size\_t j = 0; j < x; j++)

cout << "\*";

cout << endl;

}

}

return bins;

}

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

{

auto data\_size = item\_size \* item\_count;

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

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

return data\_size;

}

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;

cerr << "Enter numbers: ";

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

if(prompt)

cerr << "Enter bin count: ";

in >> data.bin\_count;

return data;

}

Input download(const string& address)

{

stringstream buffer;

CURL \*curl = curl\_easy\_init();

if(curl)

{

CURLcode res;

curl\_easy\_setopt(curl, CURLOPT\_URL, &buffer);

res = curl\_easy\_perform(curl);

if (res != 0)

{

cerr<< "curl\_easy\_perform() failed: %s\n"<< curl\_easy\_strerror(res);

exit(1);

}

if(res == CURLE\_OK)

{

long req;

res = curl\_easy\_getinfo(curl, CURLINFO\_REQUEST\_SIZE, &req);

if(!res)

fprintf(stderr,"Request size: %ld bytes\n", req);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

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

{

Input name;

if (argc > 1)

{

name = download(argv[1]);

}

else

{

name = read\_input(cin, true);

}

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

vector<size\_t>bins = make\_histogram(name);

show\_histogram\_svg (bins);

return 0;

}

**Код файла svg.cpp:**

#include <iostream>

#include "svg.h"

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

void foo(int Dash\_length, int Void\_length, double top, int BIN\_HEIGHT, int max\_bin\_count)

{

cout << "<line x1='50' y1='" << top + BIN\_HEIGHT + 5 <<"' x2='" << max\_bin\_count \* 50 << "' y2='" << top + BIN\_HEIGHT + 5

<<"' stroke='black' stroke-dasharray='" << Dash\_length << " " << Void\_length << "' />";

}

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\_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) //За цвет линий в SVG отвечает атрибут stroke, а за цвет заливки — fill.

{

cout << "<rect x=' " <<x<< "' y='" << y << "' width='" << width << "' height='" << height << "' stroke='" << stroke << "' fill='#" << fill << " '/>";

}

void svg\_end()

{

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

}

string make\_info\_text(int n)

{

stringstream buffer;

DWORD dwVersion = 0;

dwVersion = GetVersion();

DWORD mask = 0x40000000;

if ((dwVersion & mask) == 0)

{

mask = 0xFFFF;

DWORD version = dwVersion & mask;

DWORD version\_major = 0;

DWORD version\_minor = 0;

mask = 0xFF;

version\_major = dwVersion & mask;

version\_minor = (dwVersion>>8) & mask;

DWORD platform = dwVersion >> 16;

char infoBuf[INFO\_BUFFER\_SIZE];//массив

DWORD bufCharCount = INFO\_BUFFER\_SIZE;

GetUserName(infoBuf, &bufCharCount);//почему infoBuf передаеться по указателю, а bufCharCount по ссылке?

if(n == 0)

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

if(n == 1)

buffer << "Computer name: " << infoBuf;

return buffer.str();

}

}

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

{

//Индивидуальное задание

int Dash\_length = 0;

int Void\_length = 0;

cerr << "Enter the length of the dash : ";

cin >> Dash\_length;

cerr << "Enter the length of the skip : ";

cin >> Void\_length;

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;

double top = 0;

//Маcштабирование

int max\_bin\_count = bins[0];

for (int i = 0; i < bins.size(); i++)

if (bins[i] > max\_bin\_count)

max\_bin\_count = bins[i];

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

{

int height = bins[i];

if (max\_bin\_count > MAX\_ASTERISK)

height = 76 \* (static\_cast<double>(bins[i])/ max\_bin\_count);

}

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

for (size\_t bin : bins)

{

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

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

foo(Dash\_length, Void\_length, top, BIN\_HEIGHT, max\_bin\_count);

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "lime", "#00ff00");

top += BIN\_HEIGHT +10;

}

for(int i = 0; i < 2; ++i)

{

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

top += 16;

}

svg\_end();

}

**Код файла svg.h:**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include "histogram.h"

#include <string>

#include <iostream>

#include <vector>

#include <curl/curl.h>

#include <sstream>

#define INFO\_BUFFER\_SIZE 32767

void foo(int Dash\_length, int Void\_length, double top, int BIN\_HEIGHT, int max\_bin\_count);

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

void svg\_begin(double width, double height);

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 svg\_end();

#endif // SVG\_H\_INCLUDED

**Индивидуальное здание:**

**Вариант №14.**

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

Чтобы вывести в стандартный вывод ошибки размера запроса, который cURL отправляет серверу, нужно обратить внимание на функцию: curl\_easy\_getinfo().

Если переменная res возвращает значение без ошибок, то оно равняется значению CURL\_OK , соответственно выполняется дальнейший код.

Дальше я объявляю переменные и при помощи функции curl\_easy\_getinfo и внутренней функции request size помещаем переменную req, которая будет записывать байты.

И при помощи функции fprintf вывожу в консоль.

if(res == CURLE\_OK)

{

long req;

res = curl\_easy\_getinfo(curl, CURLINFO\_REQUEST\_SIZE, &req);

if(!res)

fprintf(stderr,"Request size: %ld bytes\n", req);

}