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 № 2**

# По курсу «Разработка ПО систем управления»

# «Декомпозиция и контроль корректности программ»

# Выполнил студент группы А-02-19:

# Каюк М.А.

# Проверили:

# Мохов А. С.

Козлюк Д. А

**Цель работы**

**1 часть:**

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.

**2 часть:**

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.

**Вариант 8**

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок IP-адрес сервера, с которого скачан файл (не local).

**Ссылка на страницу документации:**

<https://curl.haxx.se/libcurl/c/CURLINFO_PRIMARY_IP.html>

1

**1 часть:**

**Main.cpp:**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

}

;

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;

}

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

{

double min=0;

double max=0;

find\_minmax(Input, 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;

}

Input

read\_input(istream& in, bool prompt)

{

Input data;

size\_t number\_count;

if(prompt==true)

{

cerr << "Enter number count: ";

}

in >> number\_count;

if(prompt==true)

{

cerr << "Enter numbers: ";

}

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

size\_t bin\_count;

if (prompt==true)

{

cerr << "Enter column count: ";

}

in >>bin\_count;

data.bin\_count=bin\_count;

return data;

}

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

{

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

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

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

buffer->write(char\_items, data\_size);

return data\_size;

}

Input

download(const string& address)

{

stringstream buffer;

char \*ip;

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!=CURLE\_OK)

{

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

exit(1);

}

curl\_easy\_getinfo(curl, CURLINFO\_PRIMARY\_IP, &ip);

cerr << "IP:" << ip;

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;

}

**Histogram.h:**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <string>

#include <vector>

using namespace std;

struct Input;

void find\_minmax(struct Input Input, double& min, double& max);

#endif // HISTOGRAM\_H\_INCLUDED

**Histogram.cpp:**

#include "histogram.h"

#include <vector>

#include <iostream>

using namespace std;

struct Input

{

vector<double> numbers;

size\_t bin\_count;

}

;

void find\_minmax(struct Input Input, double& min, double& max)

{

int p=Input.numbers.size();

if(p!=0)

{

min = Input.numbers[0];

max = Input.numbers[0];

for (double number : Input.numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

**2 часть:**

**Main.cpp:**

#include "histogram.h"

#include "svg.h"

#include <iostream>

#include <vector>

#include <cstdio>

#include <windows.h>

#include <string>

#include <sstream>

using namespace std;

vector<double> input\_numbers(size\_t count)

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

vector<size\_t> make\_histogram(const vector<double>& numbers, size\_t count)

{

double min=0;

double max=0;

find\_minmax(numbers, min, max);

vector<size\_t> bins(count);

for (double number : numbers)

{

size\_t bin = (size\_t)((number - min) / (max - min) \* count);

if (bin == count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

void show\_histogram\_text(const 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';

}

}

int main()

{

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

cerr << "Enter numbers: ";

const auto numbers = input\_numbers(number\_count);

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

const auto bins = make\_histogram(numbers, bin\_count);

show\_histogram\_svg(bins);

return 0;

}

**Svg.h:**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <string>

#include <vector>

#include <windows.h>

#include <sstream>

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);

void svg\_text1(double left, double baseline, string text);

string make\_info\_text();

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

#endif // SVG\_H\_INCLUDED

**Svg.cpp:**

#include "svg.h"

#include "histogram.h"

#include <vector>

#include <iostream>

#include <windows.h>

#include <string>

#include <sstream>

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 = "black",string fill = "black")

{

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

}

void svg\_text1(double left, double baseline, string text)

{

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

}

string make\_info\_text()

{

stringstream buffer;

DWORD dwVersion = 0;

dwVersion = GetVersion();

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD platform = info >> 16;

DWORD version = info & mask;

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

{

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

DWORD version\_major = version & mask\_major;

DWORD version\_minor = version >> 8;

printf("Win 16-x version is %x\n", version);

printf("Win decimal-version is %u\n", version);

printf("Win major version is %x\n", version\_major);

printf("Win minor version is %u\n", version\_minor);

DWORD build = platform;

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

}

char system\_dir[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD size = sizeof(system\_dir);

GetComputerNameA(system\_dir, &size);

buffer << "Computer name: " << system\_dir << "\n";

return buffer.str();

}

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

{

string stroke;

string fill;

const auto IMAGE\_WIDTH = 500;

const auto IMAGE\_HEIGHT = 400;

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;

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,stroke="black",fill="black");

top+=BIN\_HEIGHT;

}

svg\_text1(TEXT\_LEFT, top+BIN\_HEIGHT, make\_info\_text());

svg\_end();

}

**Histogram.h:**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <string>

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

**Histogram.cpp:**

#include "histogram.h"

#include <vector>

#include <iostream>

#include <windows.h>

#include <string>

#include <sstream>

using namespace std;

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

{

int p=numbers.size();

if(p!=0)

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}