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»

Выполнил студент группы А-03-20

Коротков Егор Дмитриевич

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

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.

Код файла main.cpp:

#include <iostream>

#include <vector>

#include <iomanip>

#include "histogram.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

#include <windows.h>

#include <stdio.h>

#include <math.h>

using namespace std;

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 F)

{

Input data;

if (F) cerr << "Enter number count: ";

size\_t number\_count;

in >> number\_count;

if (F) cerr << "Enter numbers: ";

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

if (F) cerr << "Enter bin count: ";

in >> data.bin\_count;

return data;

}

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';

}

}

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

{

size\_t number\_count=data.numbers.size();

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

cerr<< "cerr "<<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 < 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;

}

}

// цикл по numbers не закончился!

if (!found)

{

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

}

} // конец цикла по numbers

return bins;

}

Input

download(const string& address)

{

stringstream buffer;

const char\* line;

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

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

char \*curl\_version();

cerr << " "<< \*curl\_version();

const char \*ssl\_version;

if(curl)

{

CURLcode res;

curl\_easy\_setopt(curl, CURLOPT\_URL, address);

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

const char \*version;

curl\_version\_info\_data \*d = curl\_version\_info(CURLVERSION\_NOW);

/\* compare with the 24 bit hex number in 8 bit fields \*/

if(d->version\_num >= 0x072100) {

/\* this is libcurl 7.33.0 or later \*/

cerr << "Succcess\n";

}

else {

cerr << "A too old version\n";

}

curl\_version\_info\_data \*ver = curl\_version\_info(CURLVERSION\_NOW);

ssl\_version = curl\_version\_info (CURLVERSION\_NOW)->ssl\_version;

cerr <<"Version "<< ssl\_version;

res = curl\_easy\_perform(curl);

if (res != 0)

{

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

exit(1);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false);

}

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);

}

BOOL GetComputerNameA(

LPSTR lpBuffer,

LPDWORD lpnSize

);

string make\_info\_text()

{

stringstream buffer;

// TODO: получить версию системы, записать в буфер.

DWORD WINAPI GetVersion(void);

DWORD info = 0;

info = GetVersion();

printf("n = %08x\n",info); //16

printf("%u \n", info);//10

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

DWORD version = info & mask;

cerr << " "<<version<<endl;

DWORD mask1 = 0x0000ffff;

DWORD platform = info >> 16;

DWORD maskVision = 0b00000000'11111111;

DWORD version\_major = version & maskVision;

DWORD mask2 = 0x0000ffff;

DWORD version\_minor = version >> 8;

if ((info & 0x1000ffff) == 0)

{

version\_major = 0;

}

DWORD build = platform;

printf("Wndows v.%u.%u(build %u)\n", version\_major, version\_minor,build);

//TODO:

char system\_dir[MAX\_PATH];

GetSystemDirectory(system\_dir, MAX\_PATH);

printf("System directory: %s", system\_dir);

TCHAR bufferrr[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD ssize = sizeof(bufferrr) / sizeof(TCHAR);

GetComputerName(bufferrr, &ssize);

cout <<"Computer name: " <<bufferrr <<endl;

return buffer.str();

}

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;

}

Код файла histogram.cpp:

#include "histogram.h"

#include <iomanip>

#include <iostream>

#include <string>

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;

}

}

}

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\_end()

{

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

}

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

{

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;

const size\_t MAX\_ASTERISK = IMAGE\_HEIGHT - TEXT\_BASELINE \* 2;

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;

svg\_begin( 300, 400);

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 = BIN\_HEIGHT \* height;

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

svg\_rect(top + TEXT\_LEFT, TEXT\_BASELINE \*2 , BIN\_HEIGHT, bin\_width, "black", "blue");

top += BIN\_HEIGHT ;

}

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

svg\_end();

}

Код файла histogram.h:

#ifndef HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <string>

#define HISTOGRAM\_H\_INCLUDED

using namespace std;

struct Input { vector<double> numbers; size\_t bin\_count;};

string make\_info\_text();

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

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

void svg\_end();

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

#endif // HISTOGRAM\_H\_INCLUDED

Код файла test.cpp:

#include "histogram.h"

#include <cassert>

void test\_positive()

{

double min = 0;

double max = 0;

find\_minmax({1, 2, 3}, min, max);

assert(min == 1);

assert(max == 3);

}

void test\_negative()

{

double min = 0;

double max = 0;

find\_minmax({-1, -2, -3}, min, max);

assert(min == -3);

assert(max == -1);

}

void test\_equally()

{

double min = 0;

double max = 0;

find\_minmax({1, 1, 1}, min, max);

assert(min == 1);

assert(max == 1);

}

void test\_alone()

{

double min = 0;

double max = 0;

find\_minmax({1}, min, max);

assert(min == 1);

assert(max == 1);

}

void test\_none()

{

double min = 0;

double max = 0;

find\_minmax({}, min, max);

assert(min == 1);

assert(max == 2);

}

int main()

{

test\_positive();

test\_negative();

test\_equally();

test\_alone();

test\_none();

}

Вариант 9.

…

# Низкоуровневые операции

**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.

Программа main.cpp

string make\_info\_text()

{

stringstream buffer;

// TODO: получить версию системы, записать в буфер.

DWORD WINAPI GetVersion(void);

DWORD info = 0;

info = GetVersion();

printf("n = %08x\n",info); //16

printf("%u \n", info);//10

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

DWORD version = info & mask;

cerr << " "<<version<<endl;

DWORD mask1 = 0x0000ffff;

DWORD platform = info >> 16;

DWORD maskVision = 0b00000000'11111111;

DWORD version\_major = version & maskVision;

DWORD mask2 = 0x0000ffff;

DWORD version\_minor = version >> 8;

if ((info & 0x1000ffff) == 0)

{

version\_major = 0;

}

DWORD build = platform;

printf("Wndows v.%u.%u(build %u)\n", version\_major, version\_minor,build);

//TODO:

char system\_dir[MAX\_PATH];

GetSystemDirectory(system\_dir, MAX\_PATH);

printf("System directory: %s", system\_dir);

TCHAR bufferrr[MAX\_COMPUTERNAME\_LENGTH + 1];

DWORD ssize = sizeof(bufferrr) / sizeof(TCHAR);

GetComputerName(bufferrr, &ssize);

cout <<"Computer name: " <<bufferrr <<endl;

return buffer.str();

}