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»**

**«Декомпозиция программы»**

Выполнил студент группы А-02-19

Трофименко А.И.

Проверили

Мохов А. С.

Козлюк Д. А.

Москва 2020

#### Вариант 1

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок суммарное время (total), затраченное на загрузку файла по сети.

***Логика решения:***

Если программа запускается с параметром, то входные данные берутся из интернета, в противном же случае, ввод данных происходит вручную, так же, как и в Лабораторной работе №3. Также выводится версия ОС и сборка, а также имя пользователя компьютера.

***Код:***

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include <curl/curl.h>

#include<sstream>

//#include "histogram.h"

#include "svg.h"

#include <windows.h>

using namespace std;

vector<double> input\_numbers(istream& in, const 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: ";

in >> data.number\_count;

cerr << "Enter numbers: ";

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

cerr << "Enter column count: ";

in >> data.bin\_count;

cerr << "Enter width: ";

in >> data.width;

}

else

{

in >> data.number\_count;

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

in >> data.bin\_count;

in >> data.width;

}

return data;

}

size\_t

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

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

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

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

buffer->write(new\_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);

if (res != CURLE\_OK)

{

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

exit(1);

}

else

{

double total\_time;

res = curl\_easy\_getinfo(curl, CURLINFO\_TOTAL\_TIME, &total\_time);

if (res == CURLE\_OK)

{

cerr << "Total time: " << total\_time << endl;

}

}

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);

}

DWORD mask = 0x0000ffff;

DWORD mask\_major = 0x000000f;

DWORD info = GetVersion();

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & mask\_major;

DWORD version\_minor = version >>8;

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

{

DWORD build = platform;

}

char system\_dir[MAX\_PATH];

char computer\_name[MAX\_COMPUTERNAME\_LENGTH+1];

DWORD size = sizeof(computer\_name);

GetSystemDirectory(system\_dir, MAX\_PATH);

GetComputerName(computer\_name, &size);

// Обработка данных

const auto bins = make\_histogram(input);

// Вывод данных

show\_histogram\_svg(bins, input.number\_count, computer\_name, version\_major, version\_minor, platform);

return 0;

}

**histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include<vector>

#include<string>

#include<sstream>

struct Input {

std::vector<double> numbers;

size\_t bin\_count;

size\_t number\_count;

double width;

};

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

std::vector<size\_t> make\_histogram(const Input& data);

void show\_histogram\_text(std::vector<size\_t> bins, Input& data);

std::string custome\_width(double width, double BLOCK\_WIDTH, size\_t number\_count, size\_t min, size\_t max);

#endif

**histogram.cpp**

#include<vector>

#include"histogram.h"

#include<iostream>

using namespace std;

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

if (numbers.size() != 0)

{

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(const Input& data) {

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

double min;

double max;

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

for (double number : data.numbers) {

size\_t bin = (size\_t)((number - min) / (max - min) \* data.bin\_count);

if (bin == data.bin\_count) {

bin--;

}

result[bin]++;

}

return result;

}

string custome\_width(double width, double BLOCK\_WIDTH, size\_t number\_count, size\_t min, size\_t max) {

stringstream str;

if (width < min)

str << "Width < " << min << ". Enter the width again:";

else if (width > max)

str << "Width > " << max << ". Enter the width again:";

else if (width < BLOCK\_WIDTH\*number\_count/3)

str << "Width < number\_count/3. Enter the width again:";

return str.str();

}

void show\_histogram\_text(const vector<size\_t>& bins, Input& data) {

const size\_t SCREEN\_WIDTH = 80;

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

while (data.width < 7 || data.width > 80 || data.width < data.number\_count/3)

{

cerr << custome\_width(SCREEN\_WIDTH, 1, data.number\_count, 7, 8);

cin >> data.width;

}

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';

}

}

**svg.cpp**

#include "svg.h"

#include <vector>

#include <string>

#include <iostream>

#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, size\_t 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 custome\_width(double width, double BLOCK\_WIDTH, size\_t number\_count) {

string str = "";

if (width < 70)

str = "Width < 70. Enter the width again:";

else if (width > 800)

str = "Width > 800. Enter the width again:";

else if (width < BLOCK\_WIDTH\*number\_count/3)

str = "Width < BLOCK\_WIDTH\*number\_count/3. Enter the width again:";

return str;

}

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t number\_count, char computer\_name[], DWORD version\_major, DWORD version\_minor, DWORD build) {

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 double BLOCK\_WIDTH = 10;

double IMAGE\_WIDTH;

cerr << "Enter width:";

cin >> IMAGE\_WIDTH;

while (IMAGE\_WIDTH < 70 || IMAGE\_WIDTH > 800 || IMAGE\_WIDTH < BLOCK\_WIDTH\*number\_count/3)

{

cerr << custome\_width(IMAGE\_WIDTH, BLOCK\_WIDTH, number\_count);

cin >> IMAGE\_WIDTH;

}

const size\_t MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_LEFT - TEXT\_WIDTH;

size\_t max\_count = 0;

for (size\_t count : bins) {

if (count > max\_count) {

max\_count = count;

}

}

const bool scaling\_needed = max\_count \* BLOCK\_WIDTH > MAX\_ASTERISK;

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

double top = 0;

for (size\_t bin : bins) {

size\_t height = bin;

if (scaling\_needed) {

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

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

}

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "red", "green");

top += BIN\_HEIGHT;

}

cout << "<text x='" << left << "' y='" << top + TEXT\_BASELINE << "'>Computer name: " << computer\_name << "</text>";

cout << "<text x='" << left << "' y='" << top + 2\*TEXT\_BASELINE << "'>Windows v" << version\_major << "."

<< version\_minor << " (build " << build << ")</text>";

svg\_end();

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

#include"histogram.h"

#include <windows.h>

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

void svg\_end();

void svg\_text(double left, double baseline, size\_t text);

void svg\_rect(double x, double y, double width, double height, std::string stroke = "black", std::string fill = "black");

void show\_histogram\_svg(const std::vector<size\_t>& bins, size\_t number\_count, char computer\_name[], DWORD version\_major, DWORD version\_minor, DWORD build);

std::string custome\_width(double IMAGE\_WIDTH,double BLOCK\_WIDTH,size\_t number\_count);

#endif // SVG\_H\_INCLUDED

***Ссылка на репозитарий:***

https:// github.com/TrofimenkoAI/cs-lab04