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»

# «Библиотеки и низкоуровневые операции»

# *Выполнил:*

# Студенет группы А-03-20

# Уданенко А.Д.

# *Проверили:*

# Мохов А. С

# Козлюк Д. А

**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.
4. Уметь применять побитовые операции для типовых сценариев.
5. Уметь работать с API, принимающими указатели, в том числе строки C.
6. Знать характерные особенности документации на API библиотек.

# Задание:

# Часть 1. Библиотеки

Добавить возможность построения гистограммы по данным из файла из сети. Адрес файла задается аргументом командной строки программы. Если адрес не задан, читать данные со стандартного ввода, как раньше.

Работу нужно вести на основе кода общего задания к ЛР № 3 в старом репозитарии в отдельной ветке без создания нового проекта. По этой причине во всех примерах используется lab03.exe.

# Часть 2. Низкоуровневые операции

# Код, который будет написан в этой части работы, должен в итоге формировать строку, которую нужно отобразить снизу итогового SVG в формате:

Windows v5.1 (build 1234)

Computer name: My-Comp

**Индивидуальное задание**

**Вариант 5**

Добавьте программе опцию -generate, при указании которой программа будет генерировать числа вместо их считывания. Количество столбцов считывать по-прежнему нужно. Например, lab03.exe -generate 100 генерирует 100 чисел. Опция может стоять до или после URL: lab03.exe -generate 10 http://... или lab04.exe http://... -generate 10. Если после -generate не стоит числа, нужно печатать сообщение с подсказкой, как запускать программу, и завершать работу.

**Логика решения варианта:**

В основной программе мы проверяем есть ли -generate   
if (strcmp(argv[1], "-generate") == 0)  
если она есть мы считываем количество чисел, которые нужно загенерировать .  
n=atoi(argv[2])  
Указываем на ошибку, если после generate нет числа или оно равно 0, и заканчиваем программу.  
if (n==0){

cerr << "Enter the number of generated numbers after ' -generate' ";

return 0;

}

также добавляем bool flag, он будет отвечать за то, надо ли считывать в число элементов и массив .  
Если все вышеперечисленное выполняется присваиваем значение false  
flag=false   
Также flag должен быть передан во все функции для считывания   
input = download(argv[1],flag,n);

Если -generate cтоит после ссылки принцип решения не меняется, изменяется лишь элемент массива, который мы обрабатываем  
if (strcmp(argv[2], "-generate") == 0){

n=atoi(argv[3]);

if (n==0){

cerr << "Enter the number of generated numbers after ' -generate' ";

return 0;

}

flag=false;

input = download(argv[1],flag,n);}

}  
Теперь чтение функции выглядит так  
Input

read\_input(istream& in, bool prompt, bool flag, int n) {

Input data;

if (prompt)

cerr << "Enter column count: ";

size\_t bin\_count;

in >> data.bin\_count;

if (prompt)

cerr << "Enter number count: ";

size\_t number\_count;

if (flag)

in >> number\_count;

else

number\_count=n;

if (prompt)

cerr << "Enter numbers: ";

if (flag)

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

else

{

int A = 0;

int B = 100;

vector<double> result(n);

for (int i=0; i<n; i++){

result[i]= A + rand() % ((B + 1) - A);

}

data.numbers=result;

}

return data;

}

Сначала считывается число корзин для удобства, далее в зависимости от значения flag считываются или не считываются все остальные переменные

**Исходный код всех модулей**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <sstream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <windows.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 prompt, bool flag, int n) {

Input data;

if (prompt)

cerr << "Enter column count: ";

size\_t bin\_count;

in >> data.bin\_count;

if (prompt)

cerr << "Enter number count: ";

size\_t number\_count;

if (flag)

in >> number\_count;

else

number\_count=n;

if (prompt)

cerr << "Enter numbers: ";

if (flag)

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

else

{

int A = 0;

int B = 100;

vector<double> result(n);

for (int i=0; i<n; i++){

result[i]= A + rand() % ((B + 1) - A);

}

data.numbers=result;

}

return data;

}

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

double max;

double min;

find\_minmax(data, min, max);

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

for (double number : data.numbers) {

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

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

bin--;

}

bins[bin]++;

}

return bins;

}

void show\_histogram\_text(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';

}

return;

}

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<char\*>(items), data\_size);

return data\_size;

}

Input

download(const string& address,bool flag, int n) {

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\_WRITEDATA, &buffer);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, write\_data);

res = curl\_easy\_perform(curl);

if (res)

{

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

exit(1);

}

}

curl\_easy\_cleanup(curl);

return read\_input(buffer, false, flag,n);

}

int

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

srand(time(NULL));

bool flag=true;

int n=0;

Input input;

if (argc > 1) {

if (strcmp(argv[1], "-generate") == 0 || strcmp(argv[2], "-generate" ) == 0){

if (strcmp(argv[1], "-generate") == 0){

n=atoi(argv[2]);

if (n==0){

cerr << "Enter the number of generated numbers after ' -generate' ";

return 0;

}

flag=false;

input = download(argv[3],flag,n);}

if (strcmp(argv[2], "-generate") == 0){

n=atoi(argv[3]);

if (n==0){

cerr << "Enter the number of generated numbers after ' -generate' ";

return 0;

}

flag=false;

input = download(argv[1],flag,n);}

}

else

input = download(argv[1],true,0);

}

else {

input = read\_input(cin, true, true,n);

}

// Обработка данных

const auto bins = make\_histogram(input);

// Вывод данных

show\_histogram\_svg(bins);

return 0;

}

**histogram.cpp**

#include "histogram.h"

#include <vector>

using namespace std;

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

if (t.numbers.size() == 0)

return;

min = t.numbers[0];

max = t.numbers[0];

for (double number : t.numbers) {

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

return ;

}

**histogram.h**

#pragma once

#include <vector>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

**Часть 4.2**

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <windows.h>

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;

}

int

main() {

// Ввод данных

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;

// Обработка данных

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

// Вывод данных

size\_t bin\_height;

show\_histogram\_svg(bins,bin\_height);

return 0;

}

**svg.cpp**

#include <iostream>

#include <vector>

#include <sstream>

#include <windows.h>

using namespace std;

string

make\_info\_text() {

stringstream buffer;

DWORD info = GetVersion();

DWORD mask = 0x0000ffff;

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD mask\_2 = 0x0000ff;

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

{

DWORD version\_major = version & mask\_2;

DWORD version\_minor = version >> 8;

DWORD build = platform;

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

}

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

DWORD size = MAX\_COMPUTERNAME\_LENGTH+1;

GetComputerNameA(computer\_name, &size);

buffer << " Computer name:" << computer\_name;

return buffer.str();

}

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>" << endl;

}

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 << "' />" <<endl;

}

void

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

const auto IMAGE\_WIDTH = 810;

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);

const size\_t SCREEN\_WIDTH = 80;

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

size\_t max\_count = 0;

for (size\_t count : bins) {

if (count > max\_count) {

max\_count = count;

}

}

size\_t kolichestvo\_probelov=0;

double top = 0;

for (size\_t bin : bins) {

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

size\_t height = bin;

if (scaling\_needed) {

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

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

}

if (max\_count < MAX\_ASTERISK)

kolichestvo\_probelov = max\_count - height;

else

kolichestvo\_probelov = MAX\_ASTERISK - height;

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

const double bin\_probel = BLOCK\_WIDTH \* kolichestvo\_probelov;

svg\_rect(0, top, bin\_probel, BIN\_HEIGHT, "white" , "#ffeceb");

svg\_rect(bin\_probel, top, bin\_width, BIN\_HEIGHT, "white" , "#c71585");

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

top += BIN\_HEIGHT;

}

svg\_text(0, top+BIN\_HEIGHT, make\_info\_text());

svg\_end();

}

**svg.h**

#pragma once

#include <iostream>

using namespace std;

string

make\_info\_text();

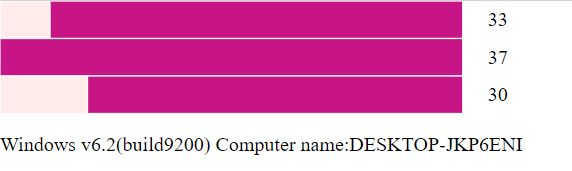
void svg\_begin(double width, double height);

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);

**Итоговый вид программы **