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:*

# Студентка группы А-03-20

# Тюрин Е.О..

# *Проверили:*

# Мохов А. С

# .

**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**

С помощью функции curl\_easy\_getinfo() печатайте на стандартный вывод ошибок время, затраченное на установку соединения с сервером (connect).

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <curl/curl.h>

#include <string>

#include <windows.h>

#include <sstream>

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

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;

}

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);

cerr << "Enter column count: ";

in >> data.bin\_count;

}

else

{

size\_t number\_count;

in >> number\_count;

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

in >> data.bin\_count;

}

return data;

}

size\_t

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

{

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

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

buffer->write(reinterpret\_cast<const char\*>(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;

double connect;

CURLcode ret;

curl\_easy\_setopt(curl, CURLOPT\_URL, address.c\_str());

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

curl\_easy\_setopt(curl, CURLOPT\_FAILONERROR, 1L);

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

res = curl\_easy\_perform(curl);

ret = curl\_easy\_perform(curl);

if(ret == CURLE\_HTTP\_RETURNED\_ERROR) {

cerr << "error\n";

}

if(CURLE\_OK == res)

{

res = curl\_easy\_getinfo(curl, CURLINFO\_CONNECT\_TIME, &connect);

if(CURLE\_OK == res)

{

cerr <<"Connect: " << connect;

}

}

curl\_easy\_cleanup(curl);

if(res)

{

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

exit(1);

}

}

return read\_input(buffer, false);

}

vector<size\_t>

make\_histogram(Input a){

double min,max;

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

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

double max\_count=0;

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

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

for (size\_t i=0;i<number\_count;i++){

bool found = false;

for (size\_t j=0;(j<a.bin\_count) && !found;j++){

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

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

if((a.numbers[i]>=low)&&(a.numbers[i]<hi)){

bins[j]++;

found=true;

}

if (bins[j]>max\_count) max\_count=bins[j];

}

if (!found) {

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

}

}

return bins;

}

void

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

size\_t bin\_count=bins.size();

for (size\_t i=0;i<bin\_count;i++){

if (bins[i]<100)

cout<<" ";

if (bins[i]<10)

cout<<" ";

cout<<bins[i]<<"|";

for (size\_t j=0;j<bins[i];j++)

cout<<"\*";

cout<<endl;

}

}

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);

}

**histogram.cpp**

#include "histogram.h"

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

void

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

min = numbers[0];max=numbers[0];

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

for (double x:numbers){

if(x<min) {

min=x;

}

else if (x>max){

max=x;

}

}

}

**Svg.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <windows.h>

#include <sstream>

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

string make\_info\_text(){

stringstream buffer;

DWORD mask = 0x0000ffff;

DWORD info=GetVersion();

if ((info & 0xa000) == 0) {

DWORD version = info & mask;

DWORD platform = info >> 16;

DWORD mask\_minor=0x00ff;

DWORD version\_minor=version & mask\_minor;

DWORD version\_major=version >>8;

buffer<<"Version "<<version\_minor<<"."<<version\_major<<" (build "<<platform<<")";

char buff[MAX\_COMPUTERNAME\_LENGTH+1];

DWORD size;

size=sizeof(buff);

GetComputerName(buff,&size);

buffer<<buff;

}

else cout<<"net"<<endl;

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>";

}

void svg\_rect(double x, double y, double width, double height,string stroke = "black", string fill = "red"){

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

}

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;

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

svg\_text(TEXT\_LEFT, TEXT\_BASELINE, to\_string(bins[0]));

double top = 0;

size\_t max\_count = bins[0]; //поиск max из корзин

for (size\_t bin : bins)

if (bin > max\_count)

max\_count = bin;

for (size\_t bin : bins) {

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

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

top += BIN\_HEIGHT;

}

top=0;

if (max\_count > MAX\_ASTERISK ){

for(size\_t bin : bins){

const double bin\_width = static\_cast <double> (IMAGE\_WIDTH - TEXT\_LEFT)/max\_count \* bin;

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

top+=BIN\_HEIGHT;

}

}

else{

for(size\_t bin : bins){

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

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

top+=BIN\_HEIGHT;

}

}

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

svg\_end();

}

**Histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

void

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

#endif // HISTOGRAM\_H\_INCLUDED

**Svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

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\_text(vector<size\_t> bins);

void

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

#endif // SVG\_H\_INCLUDED