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** студент группы А-01-19

Юрченков Г. А.

**Проверили**

Мохов А. С

Козлюк Д. А

Москва 2020

**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**

В int main(int argc, char\* argv[]) будем проверять задается ли опция   
-generate и идет ли после нее число. По результатам проверки вызываем функцию download. В read\_input будем формировать data.numbers с помощью функции rand() , а в number\_count занесем число после -generate.

**Ссылка на репозитарий -** <https://github.com/YurchenkovGA/lab_03_plus>

**Код программы**

**main.cpp**

#include <iostream>

#include <vector>

#include <curl/curl.h>

#include <cstdio>

#include <string>

#include <sstream>

#include "histogram.h"

#include "svg.h"

#include <stdlib.h>

#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, int argc, char\* argv[], bool flag\_control) {

Input data;

size\_t number\_count;

if (prompt)

{

cerr << "Enter number count: ";

in >> number\_count;

cerr << "Enter numbers: ";

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

cerr << "Enter bin count : ";

in >> data.bin\_count;

}

else

{

if (flag\_control){

srand(time(0));

int number\_count=0;

int random;

for (int i=0; i<argc; i++)

if (string(argv[i])=="-generate"){

number\_count=atoi(argv[i+1] );}

vector<double>random\_numbers(number\_count);

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

random=0;

for (int j=0; j<12; j++)

random+=rand();

random\_numbers[i]=random;}

data.numbers=random\_numbers;

in >> data.bin\_count;

}

else{

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) {

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, int argc, char\* argv[], bool flag\_control ) {

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 ){

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

exit(1);

}

curl\_easy\_cleanup(curl);

}

return read\_input(buffer, false, argc, argv,flag\_control );

}

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

Input input;

if (argc > 1){

bool flag=false;

bool flag\_fault=false;

int link;

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

if (string(argv[i])=="-generate"){

flag=true;

if (i==(argc-1)){

cerr<<"Please set number count after <-generate>";

flag\_fault=true;}

else if (string(argv[i+1]).size()>12){

cerr<<"Please set number count after <-generate>";

flag\_fault=true;}}

if (string(argv[i]).size()>10)

link=i;}

if (!flag && !flag\_fault)

input = download(argv[1],argc, argv, false);

if (!flag\_fault && flag)

input =download (argv[link],argc, argv, true);}

else

input = read\_input(cin, true, argc, argv, true);

const auto bins = make\_histogram(input);

show\_histogram\_svg(bins);

return 0;

}

**svg.cpp**

#include "svg.h"

#include "histogram.h"

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<<"'/>";}

double scaling(const vector<size\_t>& bins, double block, double IMAGE\_WIDTH , double TEXT\_LEFT ){

double max;

max=-1;

for (size\_t bin: bins){

if (bin>max)

max=bin;

}

block = ((IMAGE\_WIDTH-TEXT\_LEFT) /max);

return block;

}

size\_t text\_reflection\_check (size\_t IMAGE\_WIDTH , size\_t TEXT\_WIDTH , size\_t TEXT\_LEFT){

size\_t text\_reflection;

return text\_reflection=IMAGE\_WIDTH + TEXT\_WIDTH - TEXT\_LEFT;

}

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

const auto block= 10;

const auto IMAGE\_WIDTH = 820;

const auto TEXT\_LEFT = 40;

double BLOCK\_WIDTH = scaling(bins, block, IMAGE\_WIDTH, TEXT\_LEFT);

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 10;

size\_t text\_reflection = text\_reflection\_check (IMAGE\_WIDTH, TEXT\_WIDTH , TEXT\_LEFT);

size\_t histogram\_reflection;

const auto BIN\_HEIGHT = ((IMAGE\_HEIGHT-100)/bins.size());

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

double top = 0;

for (size\_t bin : bins) {

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

histogram\_reflection = IMAGE\_WIDTH-bin\_width-TEXT\_LEFT;

svg\_rect(histogram\_reflection, top, bin\_width, BIN\_HEIGHT);

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

top += BIN\_HEIGHT;

}

svg\_text(TEXT\_LEFT+TEXT\_WIDTH, top + TEXT\_BASELINE, make\_info\_text());

svg\_end();

}

**histogram.cpp**

#include "histogram.h"

#include <vector>

#include <iostream>

#include <windows.h>

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

{

if(numbers.size()==0)

{

return;

}

else{

min=numbers[0];

max=numbers[0];

for (size\_t i=1; i<numbers.size(); i++)

{

if (min>numbers[i])

{

min=numbers[i];

}

if (max<numbers[i])

{

max=numbers[i];

}}}

}

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

{

double min=0;

double max=0;

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

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

for (double number : data.numbers)

{

size\_t bin;

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

if (bin == data.bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;}

string make\_info\_text() {

stringstream buffer;

DWORD WINAPI GetVersion();

DWORD mask = 0x0000ffff;

DWORD mask2 = 0x000000ff;

DWORD info = GetVersion();

DWORD platform = info >> 16;

DWORD version = info & mask;

DWORD version\_major = version & mask2;

DWORD version\_minor = version >>8;

if ((version & 0x40000000) == 0) {

DWORD build = platform;

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

}

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

DWORD size = sizeof(computer\_name);

GetComputerNameA(computer\_name, &size);

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

return buffer.str();

}

**histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

struct Input {

vector<double> numbers;

size\_t bin\_count;

};

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

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

string make\_info\_text();

#endif // HISTOGRAM\_H\_INCLUDED

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

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);

size\_t text\_reflection\_check (size\_t IMAGE\_WIDTH , size\_t TEXT\_WIDTH , size\_t TEXT\_LEFT);

double scaling(const vector<size\_t>& bins, double block, double IMAGE\_WIDTH );

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

#endif // SVG\_H\_INCLUDED