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 № 3**

# По курсу «Разработка ПО систем управления»

# «Декомпозиция программы»

# Выполнил студент группы А-01-19

# Юрченков Г.А.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

**Задание.**

# 1) Написать программу для построения гистограммы массива чисел как изображения в формате SVG

2) Доработать программу в соответствии с вариантом.

Вариант 5  
Отображайте гистограмму зеркально по аналогии с заданием этого варианта в лабораторной работе № 1.  
  
**Пояснение к программе.**

Изначально выделяю пространство в конце изображения под текст, затем отступаю от текста и еще на величину bin\_width (размер столбца с учетом масштабирования ) и строю столбец от этого отступа до текста, получается отраженная гистограмма.   
  
MAIN.CPP  
  
#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.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()

{

size\_t number\_count;

cerr<< "enter number count";

cin>>number\_count;

const auto numbers=input\_numbers(number\_count);

size\_t bin\_count;

cerr<< "enter bin count";

cin>>bin\_count;

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

show\_histogram\_svg(bins);

return 0;

}  
  
HISTOGRAM.CPP  
  
#include "histogram.h"

#include <vector>

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 vector<double>& numbers, size\_t bin\_count){

double min, max;

find\_minmax(numbers, min, max);

vector<size\_t>count(bin\_count,0);

for (double x: numbers){

size\_t bin\_index=(size\_t)(x-min)\*(bin\_count)/(max-min);

if (bin\_index==bin\_count)

{

bin\_index--;

}

count[bin\_index]++;}

return count;

}  
  
HISTOGRAM.H  
  
 #ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

vector<size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count);

#endif // HISTOGRAM\_H\_INCLUDED

SVG.CPP  
  
#include <iostream>

#include <vector>

#include "svg.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;

}

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;

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

size\_t text\_reflection = IMAGE\_WIDTH + TEXT\_WIDTH - TEXT\_LEFT;

size\_t histogram\_reflection;

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\_end();

}  
  
SVG.H  
  
#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <iostream>

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);

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  
  
  
TEST.CPP  
  
#include "histogram.h"

#include "svg.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(max == -1);

assert(min == -3);

}

void test\_same()

{

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void test\_one()

{

double min = 0;

double max = 0;

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

assert(min == 2);

assert(max == 2);

}

void test\_empty()

{

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

void test\_text\_reflection()

{

size\_t text\_reflection;

show\_histogram\_svg({1,1,1});

assert(text\_reflection == 0);

}

void test\_histogram\_reflection()

{

size\_t histogram\_reflection;

show\_histogram\_svg({1,1,1});

assert(histogram\_reflection == 0);

}

int main()

{

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

test\_histogram\_reflection();

test\_text\_reflection();

}  
  
<https://github.com/YurchenkovGA/lab_03_plus>