**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-02-19

# Ли Егор Андреевич

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 24.04.2020

# Цель работы

1. Уметь структурировать программу при помощи функций.
2. Уметь писать модульные тесты.

Ссылка на репозитарий : <https://github.com/Eroproralee/lab-3>

Вариант 11:Добавьте рамку вокруг гистограммы, используя пунктирные линии. Для отрисовки пунктирной линии можно использовать стандартный элемент <line>, установив в нем атрибут stroke-dasharray = '10 10'

Логика решения: Необходимо было добавить вывод 4-х пунктирных линий вокруг гистограммы , для этого нужно было задать определенную высоту двум боковым линия а также их координаты и длину двум лицевым линиям . Для левой боковой линии и двух лицевых линий мы взяли за координату x const auto BLOCK\_WIDTH = 10; для правой боковой лини взяли величину TEXT\_WIDTH+lenght+5 , то есть длину самой высокой гистограммы ( её мы опеределили из цикла ) , константу TEXT\_WIDTH и +5 пиксилей для того , что бы граница была не вплотную . Координату y для верхней лицевой границе мы взяли за 0 , а за нижнюю ширину всех столбцов , для двух боковых линий y-нулевое брали за 0 , а y1 такое же как и для ниижней лицевой границе

Код:

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;

}

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

{

const size\_t SCREEN\_WIDTH = 80;

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

double min;

double max;

find\_minmax(numbers,min,max);

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

for (double number : numbers)

{

size\_t bin;

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

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

int main()

{

double length\_ch,length\_pr;

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

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

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

double min, max;

find\_minmax(numbers,min,max);

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

show\_histogram\_svg(bins,bin\_count,length\_ch,length\_pr);

return 0;

}

histogram.cpp

#include "histogram.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 (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

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);

#endif // HISTOGRAM\_H\_INCLUDED

svg.cpp

#include <iostream>

#include "svg.h"

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, string 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<<"'/>";

}

void svg\_line(double x1,double y1,double x2,double y2,double length\_ch,double length\_pr,string stroke\_1)

{

cout << "<line x1='"<<x1<<"' y1='"<<y1<<"' x2='"<<x2<<"' y2='"<<y2<<"' stroke='"<<stroke\_1<<"' stroke-dasharray='"<<10<<" "<<10<<"' />";

}

void show\_histogram\_svg(const vector<size\_t>& bins,size\_t bin\_count,double length\_ch,double length\_pr)

{

size\_t lenght = 0 ;

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;

const size\_t SCREEN\_WIDTH = 80;

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

double top = 0;

svg\_begin(400, 300);

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)

{

size\_t height = bin;

if (scaling\_needed)

{

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

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

}

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"blue","#aaffaa");

if ( lenght < bin\_width)

lenght = bin\_width ;

top += BIN\_HEIGHT;

}

svg\_line(BLOCK\_WIDTH,top ,TEXT\_WIDTH+lenght+5,top ,length\_ch,length\_pr,"red");

svg\_line(BLOCK\_WIDTH ,0 ,BLOCK\_WIDTH ,top ,length\_ch,length\_pr,"red");

// левая часть

svg\_line(TEXT\_WIDTH+lenght+5,0 ,TEXT\_WIDTH+lenght+5 ,top ,length\_ch,length\_pr,"red");

//

svg\_line(BLOCK\_WIDTH,0,TEXT\_WIDTH+lenght+5,0,length\_ch,length\_pr,"red");

svg\_end();

}

svg.h

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#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 = "black",string fill = "black");

void svg\_line(double x1,double y1,double x2,double y2,double length\_ch,double length\_pr,string stroke\_1 = "red");

void show\_histogram\_svg(const vector<size\_t>& bins,size\_t bin\_count,double length\_ch,double length\_pr);

#endif // SVG\_H\_INCLUDED

test.cpp

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#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 = "black",string fill = "black");

void svg\_line(double x1,double y1,double x2,double y2,double length\_ch,double length\_pr,string stroke\_1 = "red");

void show\_histogram\_svg(const vector<size\_t>& bins,size\_t bin\_count,double length\_ch,double length\_pr);

#endif // SVG\_H\_INCLUDED

test.h

#ifndef TEST\_H\_INCLUDED

#define TEST\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

void test\_positive();

void test\_similar ();

void test\_negative();

void test\_one\_number();

void test\_empty();

#endif // TEST\_H\_INCLUDED

test1.cpp

#include <iostream>

#include "test1.h"

#include <cassert>

#include "histogram.h"

#include "svg.h"

const auto BIN\_HEIGHT = 30;

using namespace std;

void

test\_positive() {

double min = 0;

double max = 0;

find\_minmax({1, 2, 3}, min, max);

assert(min == 1);

assert(max == 3);

}

void test\_similar ()

{

double min = 0;

double max = 0;

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

assert(min == 2);

assert(max == 2);

}

void test\_negative()

{

double min = 0;

double max = 0;

find\_minmax({-1, -2, -3}, min, max);

assert(min == -3);

assert(max == -1);

}

void test\_one\_number()

{

double min = 0;

double max = 0;

find\_minmax({1}, min, max);

assert(min == 1);

assert(max == 1);

}

void test\_empty()

{

double min = 0;

double max = 0;

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

assert(min ==0);

assert(max == 0);

}

void test\_lishnya\_liniya()

{

double top = 0;

show\_histogram\_svg({1,2,3},3,20,10);

assert (top=BIN\_HEIGHT);

}

void test\_bins0()

{

double top=0;

show\_histogram\_svg({},3,20,10);

assert (top=0);

}

test1.h

#ifndef TEST1\_H\_INCLUDED

#define TEST1\_H\_INCLUDED

#include <iostream>

#include <vector>

void test\_positive();

void test\_similar ();

void test\_negative();

void test\_one\_number();

void test\_empty();

void test\_lishnya\_liniya();

void test\_bins0();

#endif // TEST1\_H\_INCLUDED

Результаты тестирования



