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»

**Выполнила** студентка группы А-01-19

Афанасьева А.М.

**Проверили**

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 15**

Добавьте горизонтальную шкалу под гистограммой по аналогии с заданием лабораторной работы 1. Шкалу нужно разбить на интервалы, размер которых вводит пользователь. Допустимы размеры от 2 до 9 BLOCK\_WIDTH, при некорректном вводе печатайте сообщение со словом «ERROR» и завершайте работу программы. Под нулевой, первой и последней отметкой шкалы требуется напечатать соответствующие числа. Шкала должна быть во всю ширину гистограммы.

**Логика решения**

* Заведём переменную *star\_factor,* которая будет хранить в себе ширину одной звезды для вывода шкалы и гистограммы. Вычислим ее, разделив максимальную ширину вывода гистограммы (равно ширине с большим числом звезд) на количество этих звезд. Уменьшим максимальную ширину для построения изображения , чтобы туда «влез» интервал, который может выходить за границы гистограммы.
* Заведем новую переменную *interval*, которая будет хранить в себе размер интервала, вводимого пользователем.

Выясним количество выводимых интервалов (переменная *interval\_count*), причем учтем, что происходит деление двух целых чисел - прибавим единицу, если остаток от деления не равен нулю.

* Рассчитаем ширину одного интервала, *interval\_width,* умножив ширину одной звезды на их количество (*star\_factor\*interval).*
* С помощью цикла *for* выведем построение одного интервала *interval\_count* раз с учетом смены абсциссы (координаты x).
* Вызовем функцию вывода текста для отображения начала шкалы, размера одного интервала, общего количества звезд (цифр) входящих в шкалу.

(переменная *star\_factor* будет задействована и в построении гистограммы, на нее каждый раз будет умножаться количество переменных в каждой корзине)

**Код программы**

**Lab03.cpp:**

**main.cpp**  
  
#include <iostream>

#include <vector>

#include <string>

#include "histogram.h"

#include "svg.h"

using namespace std;

vector<double> input\_numbers(const 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, const size\_t count) {

vector<size\_t> result(count);

double min;

double max;

find\_minmax(numbers, min, max);

for (double number : numbers) {

size\_t bin = (size\_t)((number - min) / (max - min) \* count);

if (bin == count) {

bin--;

}

result[bin]++;

}

return result;

}

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';

}

}

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

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

// show\_interval\_svg (interval);

return 0;

}

**histogram.cpp**

#include<vector>

#include"histogram.h"

using namespace std;

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

if (numbers.size() != 0)

{

min = numbers[0];

max = numbers[0];

for (double number : numbers) {

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

}

**svg.cpp**

#include "svg.h"

#include <vector>

#include <string>

#include <iostream>

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 << "' />";

}

size\_t calculation\_star\_factor (size\_t MAX\_WIDTH, size\_t max\_count)

{

size\_t star\_factor=MAX\_WIDTH/max\_count;

return (star\_factor);

}

size\_t calculation\_interval\_width (size\_t star\_factor, size\_t interval)

{

size\_t interval\_width=star\_factor\*interval;

return(interval\_width);

}

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

{

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;

size\_t star\_factor;

//const auto BLOCK\_WIDTH = 10;

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

double top = 0;

size\_t max\_count=0;

for (size\_t bin : bins)

{

if(max\_count<bin)

max\_count=bin;

}

double MAX\_WIDTH=IMAGE\_WIDTH-4.5\*TEXT\_WIDTH;

size\_t interval;

cerr <<"Enter interval: ";

cin >> interval;

size\_t interval\_count;

if (max\_count%interval!=0)

interval\_count=max\_count/interval+1;

else

interval\_count=max\_count/interval;

star\_factor=calculation\_star\_factor(MAX\_WIDTH, max\_count);

size\_t interval\_width=calculation\_interval\_width(star\_factor,interval);

for (size\_t bin : bins)

{

const double bin\_width = star\_factor \* bin;

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

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

top += BIN\_HEIGHT;

}

size\_t otstup= (double)bin\_count\*BIN\_HEIGHT+15;

if (interval<4 || interval>9)

svg\_text(TEXT\_WIDTH, otstup, "ERROR!");

else

{

size\_t abscissa=TEXT\_WIDTH;

for (int i=1; i<=interval\_count; i++)

{

svg\_rect(abscissa, otstup, interval\_width, BIN\_HEIGHT, "black", "#90EE90");

abscissa+=interval\_width;

}

svg\_text(TEXT\_WIDTH, otstup+50, "0");

svg\_text(TEXT\_WIDTH+interval\_width, otstup+50, to\_string(interval));

svg\_text(TEXT\_WIDTH+interval\_count\*interval\_width, otstup+50, to\_string(interval\*interval\_count));

}

svg\_end ();

}

**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.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

using namespace std;

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

void svg\_end();

void svg\_text(double left, double baseline, size\_t text);

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

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

size\_t calculation\_star\_factor (size\_t MAX\_WIDTH, size\_t max\_count);

size\_t calculation\_interval\_width (size\_t star\_factor, size\_t interval);

#endif // SVG\_H\_INCLUDED

**Lab03-test RT.cpp:**

**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(min == -3);

assert(max == -1);

}

void test\_same()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

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\_calculation\_star\_factor ()

{

size\_t test\_star=0;

test\_star=calculation\_star\_factor(250, 2);

assert(test\_star == 125);

}

void test\_interval\_width ()

{

size\_t test\_interval=0;

test\_interval=calculation\_interval\_width (30, 4);

assert(test\_interval == 120);

}

int

main()

{

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

test\_calculation\_star\_factor ();

test\_interval\_width ();

}

Подключены файлы histogram.h, svg.h  
Добавлены файлы svg.cpp, histogram.cpp