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 и контроль корректности программ**](http://uii.mpei.ru/study/courses/cs/lab03/index.html)

Выполнил студент группы А-02-19

Казиков А.П.

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**Цель работы**

1) Уметь структурировать программу при помощи функций.

2)Уметь писать модульные тесты.

**Индивидуальное задание Вариант 6**

После запроса количества столбцов запросите цвет для каждого столбца.

https://github.com/KazikovAP/cs-lab03

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include "histogram.h"

#include "lab03svg.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)

{

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;

}

void show\_histogram\_text(const 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';

}

}

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 = "black")

{

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

}

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

{

vector<string> colors\_vec(bin\_count);

colors\_vec=colors(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;

const auto BLOCK\_WIDTH = 10;

double top = 0;

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

size\_t t=0;

for (size\_t bin : bins)

{

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,colors\_vec[t],colors\_vec[t]);

top += BIN\_HEIGHT;

t++;

}

svg\_end();

}

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 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);

return 0;

}

**HISTOGRAM.CPP**

#include "histogram.h"

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

{

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>

#include <string>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

Задача была написать функцию вывода цвета для каждого столбца гистограммы

**Lab03svg.cpp**

#include <iostream>

#include "lab03svg.h"

vector<string> colors(size\_t bin\_count)

{

vector<string> cin\_colors(bin\_count);

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

{

getline(cin, cin\_colors[i]);

}

return cin\_colors;

}

**Lab03svg.h**

#ifndef LAB03SVG\_H\_INCLUDED

#define LAB03SVG\_H\_INCLUDED

#include <vector>

#include <string>

using namespace std;

vector<string> colors(size\_t bin\_count);

#endif // LAB03SVG\_H\_INCLUDED

**TEST.CPP**

#include "histogram.h"

#include "lab03svg.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\_single(){

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void test\_nothing(){

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void test\_colors(){

size\_t bin\_count = 0;

vector<string> colors\_vec(bin\_count);

colors\_vec=colors(bin\_count);

vector<string> colors();

assert(colors\_vec[0] == "green");

assert(colors\_vec[0] == "grey");

assert(colors\_vec[0] == "yellow");

}

void test\_colors1(){

size\_t bin\_count = 1;

vector<string> colors\_vec(bin\_count);

colors\_vec=colors(bin\_count);

vector<string> colors(bin\_count);

assert(colors\_vec[0] == "red");

}

int

main() {

test\_positive();

test\_negative();

test\_same();

test\_single();

test\_nothing();

test\_colors();

test\_colors1();

}