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»

# Выполнил студент группы А-02-19

# Викторов Г.О.

# Проверили

# Мохов А. С.

# Козлюк Д. А.

Москва 2020

# Цель работы

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

#### Вариант 3

Измените высоту изображения IMAGE\_HEIGHT = 700. Дайте пользователю возможность задавать высоту столбца гистограммы. Если итоговая высота гистограммы больше IMAGE\_HEIGHT, рассчитывать высоту столбца как (IMAGE\_HEIGHT / bins\_count).

Логика решения: написал функцию для ввода высоты столбца гистограммы , ее результат оправил в функцию по выводу изображения гистограммы, внутри функции поставил условие:

if(BIN\_HEIGHT\*bins.size()>IMAGE\_HEIGHT)

{

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

}

Ну а дальше выоста столбца идет как параметр в другие фунции

Ссылка на репозитарий: <https://github.com/ViktorovGO/laba3>

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(size\_t count,const vector<double>& numbers)

{ double min=0;

double max=0;

find\_minmax(numbers,min,max);

vector<size\_t> bins(count);

for (double number : numbers) {

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

if (bin == count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

int main()

{ double BIN\_HEIGHT=0;

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;

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

cin\_height(BIN\_HEIGHT);

show\_histogram\_svg(bins,BIN\_HEIGHT);

return 0;

}

**Histogram:**

#include "histogram.h"

#include <string>

#include <vector>

#include <iostream>

using namespace std;

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

{ int q=numbers.size();

if(q!=0)

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

}

**SVG:**

#include"svg.h"

#include <iostream>

#include <vector>

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<<"'/>";

}

void cin\_height(double &BIN\_HEIGHT)

{

cin>>BIN\_HEIGHT;

}

void show\_histogram\_svg(const vector<size\_t>& bins,double &BIN\_HEIGHT)

{ string stroke;

string fill;

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 700;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BLOCK\_WIDTH = 10;

if(BIN\_HEIGHT\*bins.size()>IMAGE\_HEIGHT)

{

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

}

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

double top=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,stroke="black",fill="black");

top+=BIN\_HEIGHT;

}

svg\_end();

}

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';

}

}

**TEST:**

#include "histogram.h"

#include "svg.h"

#include <cassert>

void

test\_positive()

{

double BIN\_HEIGHT=200;

show\_histogram\_svg({2,5,3},BIN\_HEIGHT);

assert(BIN\_HEIGHT == 200);

}

int main()

{

test\_positive();

return 0;

}

**TEST2**

#include "histogram.h"

#include "svg.h"

#include <cassert>

void

test\_positive()

{

double BIN\_HEIGHT=1000;

show\_histogram\_svg({2,5,3},BIN\_HEIGHT);

assert(BIN\_HEIGHT == 233);

}

int main()

{

test\_positive();

return 0;

}

**TEST\_FM1:**

#include "histogram.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);

}

int

main() {

test\_positive();

}

**TEST\_FM2:**

#include "histogram.h"

#include <cassert>

void

test\_positive() {

double min = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

int

main() {

test\_positive();

}

**TEST\_FM3:**

#include "histogram.h"

#include <cassert>

void

test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

int

main() {

test\_positive();

}

**TEST\_FM4:**

#include "histogram.h"

#include <cassert>

void

test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

int

main() {

test\_positive();

}

***TEST\_FM5:***

#include "histogram.h"

#include <cassert>

void

test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

int

main() {

test\_positive();

}