**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-01-19

# Медведев И. С.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

**Цель работы**

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

Программа для построения гистограммы из ЛР № 1 состоит из одной функции main() на более чем 100 строк, из-за чего в ней неудобно ориентироваться. Необходимо выделить части программы в функции:

* Ввод чисел:
  + принимает количество чисел, которое необходимо ввести;
  + возвращает вектор чисел.
* Поиск наибольшего и наименьшего значения:
  + принимает вектор чисел;
  + возвращает два результата — min и max.
* Расчет гистограммы:
  + принимает вектор чисел и количество корзин;
  + возвращает вектор количеств чисел в каждой корзине;
  + *вызывает* в процессе работы функцию поиска min и max.

#### (Домашнее задание) 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'

Описание логики решения:

Сначала я изменяю способ вызова функции вывода гистограммы добавив ещё один параметр функции bin\_count. Затем в SVG.cpp в функцию show\_histogram\_svg добавляю поиск максимального значения (maxbin) вектора bins. Так же задаю координаты (x1, x2, y1, y2), которые в дальнейшем будут служить координатами точек для вывода линий рамки. Изменяю вычисление переменной koeff с учетом рамки. Добавляю вычисление координат x2 и y2 с учетом x1, y1 и koeff. Изменяю вычисление переменной top с учетом рамки. Затем добавляю константы INDENT и WIDTH\_LINE отвечающие за отступ и толщину линии соответственно. Далее в SVG.cpp добавляю функцию svg\_line параметрами, которой являются координаты начальной точки (x\_1, y\_1), координаты конечной точки (x\_2, y\_2) и ширина линии (width). В функции show\_histogram\_svg добавляю вывод линий рамок. Изменяю в вызове функций svg\_text и svg\_rect параметр отвечающий за координату x с учетом рамки. В конце изменяем константы для более “красивого” вывода гистограммы с рамкой.

Ссылка на репозитарий:

<https://github.com/MedvedevIgS/lab-03.git>

Код программы

**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);**

**cerr << "Enter numbers: ";**

**for (size\_t i = 0; i < count; i++)**

**{**

**cin >> result[i];**

**}**

**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';**

**}**

**return;**

**}**

**int main()**

**{**

**size\_t number\_count;**

**cerr << "Enter number count: ";**

**cin >> number\_count;**

**if (number\_count==0)**

**return 1;**

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

**size\_t bin\_count;**

**cerr << "Enter column count: ";**

**cin >> bin\_count;**

**if (bin\_count==0)**

**return 1;**

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

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

**return 0;**

**}**

**Histigram.cpp**

**#include "histogram.h"**

**#include <vector>**

**using namespace std;**

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

**{**

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

**double max, min;**

**find\_minmax(numbers, min, max);**

**for (double number : numbers)**

**{**

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

**if (bin == bin\_count)**

**{**

**bin--;**

**}**

**bins[bin]++;**

**}**

**return bins;**

**}**

**void find\_minmax(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;**

**}**

**}**

**return;**

**}**

**Histigram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

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

#endif // HISTOGRAM\_H\_INCLUDED**SVG.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

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

void svg\_line(double x\_1, double y\_1, double x\_2, double y\_2, size\_t width);

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

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

void svg\_end();

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

#endif // SVG\_H\_INCLUDED

**SVG.cpp**

#include "SVG.h"

#include <iostream>

#include <vector>

using namespace std;

void svg\_line(double x\_1, double y\_1, double x\_2, double y\_2, size\_t width)

{

cout << "<line x1='"<<x\_1<<"' y1='"<<y\_1<<"' x2='"<<x\_2<<"' y2='"<<y\_2<<"' style='stroke-dasharray: 10 10' stroke='blue' stroke-width='"<<width<<"'/>\n";

}

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<<"'/>\n";

}

void svg\_text(double left, double baseline, string text)

{

cout << "<text x='" << left << "' y='"<< baseline <<"'>"<<text<<"</text>\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 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 = 5;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 40;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 20;

const auto INDENT = 5;

const auto WIDTH\_LINE = 3;

size\_t maxbin=bins[0];

for(size\_t j:bins)

if(maxbin<j)

maxbin=j;

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

double x1, x2, y1, y2;

x1=5;

y1=5;

double koeff;

if(((IMAGE\_WIDTH-x1-WIDTH\_LINE-TEXT\_WIDTH-INDENT-WIDTH\_LINE-WIDTH\_LINE)/BLOCK\_WIDTH)<maxbin)

koeff=(IMAGE\_WIDTH-x1-WIDTH\_LINE-TEXT\_WIDTH-INDENT-WIDTH\_LINE-WIDTH\_LINE)/maxbin;

else

koeff=BLOCK\_WIDTH;

x2=x1+WIDTH\_LINE+TEXT\_WIDTH+koeff\*maxbin+INDENT+WIDTH\_LINE;

y2=y1+WIDTH\_LINE+INDENT+BIN\_HEIGHT\*bin\_count+INDENT+WIDTH\_LINE;

double top = y1+WIDTH\_LINE+INDENT;

svg\_line(x1,y1,x2,y1, WIDTH\_LINE);

svg\_line(x1,y2,x2,y2, WIDTH\_LINE);

svg\_line(x1,y1,x1,y2, WIDTH\_LINE);

svg\_line(x2,y1,x2,y2, WIDTH\_LINE);

for (size\_t bin : bins)

{

const double bin\_width = koeff \* bin;

svg\_text(x1+WIDTH\_LINE+TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));

svg\_rect(x1+WIDTH\_LINE+TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "red", "red");

top += BIN\_HEIGHT;

}

svg\_end();

}

**Test.cpp**

#include "histogram.h"

#include <iostream>

#include <cassert>

using namespace std;

void test\_make\_histogram\_empty ()

{

size\_t bin\_count = 10;

auto bin = make\_histogram ({}, bin\_count);

assert(bin[0] > 0);

}

void test\_make\_histogram\_ones ()

{

size\_t bin\_count = 3;

auto bin = make\_histogram ({1}, bin\_count);

assert(bin[0] > 0);

}

void test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

void test\_otricala()

{

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\_ones()

{

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);

}

int main()

{

test\_positive();

test\_ones();

test\_same();

test\_empty();

}