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 программного обеспечения систем управления»**

# «Декомпозиция программы»

Выполнил студент группы А-02-20

Дашин И.Н.

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

# Цель работы

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

# Часть 1. Декомпозиция программы функциями

## Задание

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.

# Часть 2. Вывод гистограммы как изображения в формате SVG

## Задача

Требуется вместо текстовой гистограммы рисовать картинку

# Часть 3. Модульное тестирование

Написать модульный тест для функции поиска минимума и максимума.

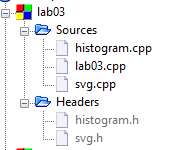
**Индивидуальное задание:**

#### Вариант 8

Запрашивать у пользователя размер шрифта. За размер шрифта отвечает атрибут font-size. Считать 12 значением по умолчанию. Не позволять вводить значения менее 8 и более 32. В этом случае предлагайте пользователю ввести значение заново с указанием причины.

**Логика решения:**

После декомпозиции программы структура файлов проекта имеет вид:



Функции поиска максимума и минимума и функция разделения чисел по корзинам находятся в файле histogram.cpp, функции, требуемые для вывода гистограммы в виде кода для вывода в формате svg выделены в файл svg.cpp

Для выполнения индивидуального задания (выбора пользователем размера шрифта чисел, обозначающих кол-во элементов в столбцах гистограммы) мы напишем функцию, которая запрашивает размер шрифта, и если размер не подходит под заданный диапазон, выводит сообщение с просьбой ввести правильный размер.

Функция возвращает результат – число, обозначающее размер шрифта. Его мы будем использовать как дополнительный аргумент функции svg\_text. Таким образом программа будет выводить код, содержащий информацию о размере шрифта чисел гистограммы.

**Модульные тесты :**

Для функции find\_minmax пишем тесты, требуемые по заданию лабораторной работы. В новом проекте lab03-test используем функции из основного проекта, чтобы в случае изменения функции в основном проекте можно было проверить функцию на правильность выполнения.

Для индивидуального задания нужно написать тесты для функции font\_size которая запрашивает размер шрифта. Так как функция взаимодействует с пользователем, нам придется видоизменить функцию чтобы представить поток как параметр функции. Так мы сможем напрямую передавать данные для проверки в тестах.

****

Для использования функции в основной программе ей в качестве аргумента будет передаваться поток стандартного ввода, и тогда функция будет получать данные напрямую от пользователя.

Для функции font\_size тесты будут проверять различные случае ввода размера шрифта. В том числе: проверка на ввод числа не из диапазона допустимых значений, ввод чисел на границе диапазона, многократный ввод недопустимых значений. В результате тестирования ошибок найдено не было.

**Код программы:**

**lab03.cpp**

#include <iostream>

#include <vector>

#include "svg.h"

#include "histogram.h"

using namespace std;

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 bin count: ";

cin >> bin\_count;

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

show\_histogram\_svg(bins);

return 0;

}

**Histogram.cpp**

#include "histogram.h"

#include <vector>

#include <string>

#include <iostream>

using namespace std;

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

vector<double> result(count);

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

cin >> result[i];

}

return result;

}

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

if (numbers.size()) {

max = numbers[0];

min = numbers[0];

if (numbers.size()) {

for (double x : numbers) {

if (x > max) {

max = x;

}

if (x < min) {

min = x;

}

}

}

}

}

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

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

double min, max;

find\_minmax(numbers, min, max);

if (min == max) {

for (double number : numbers) {

bins[0]++;

}

return bins;

}

else {

for (double number : numbers) {

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

if (bin == bin\_count) {

bin--;

}

bins[bin]++;

}

return bins;

}

}

**Histogram.h**

#pragma once

#include <iostream>

#include <vector>

using namespace std;

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

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

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

**Svg.cpp**

#include "svg.h"

#include <vector>

#include <string>

using namespace std;

int font\_size(istream& in) {

int size;

cerr << "Enter font size from 8 to 36:\n";

in >> size;

bool wrong = true;

while (wrong) {

if (size >= 8 && size <= 36) {

wrong = false;

}

else {

if (size < 8) {

cerr << "Enter again, size is too low\n";

in >> size;

}

if (size > 36) {

cerr << "Enter again, size is too big\n";

in >> size;

}

}

}

return size;

}

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, int size) {

cout << "<text x = '" << left << "' y = '" << baseline << "' font-size = '" << size << "' >" << 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 show\_histogram\_svg(const vector<size\_t>& bins) {

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 MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_WIDTH;

const int size = font\_size(cin);

size\_t max\_count = bins[0];

for (size\_t bin : bins) {

if (bin > max\_count) {

max\_count = bin;

}

}

const bool scaling\_needed = max\_count \* BLOCK\_WIDTH > MAX\_ASTERISK;

double scaling\_factor;

if (scaling\_needed)

{

scaling\_factor = static\_cast<double>(MAX\_ASTERISK) / (max\_count \* BLOCK\_WIDTH);

}

else {

scaling\_factor = 1;

}

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

double top = 50;

for (size\_t bin : bins) {

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "black", "#c72af7");

top += BIN\_HEIGHT;

}

svg\_end();

}

**Svg.h**

#pragma once

#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, string fill);

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

int font\_size(istream& in);

**main.cpp**

#include <cassert>

#include <sstream>

#include <iostream>

#include "svg.h"

#include "histogram.h"

using namespace std;

void test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

void test\_negative() {

double min = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void test\_constant() {

double min = 0;

double max = 0;

find\_minmax({ 99,99,99 }, max, min);

assert(min == 99);

assert(max == 99);

}

void test\_onenumber() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_empty() {

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

void test\_big() {

stringstream in("50 10");

assert(font\_size(in) == 10);

}

void test\_low() {

stringstream in("4 14");

assert(font\_size(in) == 14);

}

void test\_lot() {

stringstream in("1 1 1 1 1 14");

assert(font\_size(in) == 14);

}

void test\_loth() {

stringstream in("99 99 99 99 25");

assert(font\_size(in) == 25);

}

void test\_8() {

stringstream in("8");

assert(font\_size(in) == 8);

}

void test\_36() {

stringstream in("36");

assert(font\_size(in) == 36);

}

int main() {

// Тесты для функции find\_minmax

test\_positive();

test\_negative();

test\_constant();

test\_onenumber();

test\_empty();

// Тесты для функции font\_size

test\_big();

test\_low();

test\_lot();

test\_loth();

test\_8();

test\_36();

return 0;

}