Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Отчёт по лабораторной работе №3**

**Вариант 9**

Выполнил Котухов М.Д.

Группа А-01-20

Проверил:

Мохов А.С.

Козлюк Д.А.

Москва 2021 НИУ МЭИ

**ЛР № 3. Декомпозиция программы**

# Цель работы

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

**Задача:**

**Часть 1. 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. Требуется вместо текстовой гистограммы рисовать картинку, например:



# Часть 3. Написать модульный тест для функции поиска минимума и максимума.

Индивидуальное задание:

#### Вариант 9

Запросите у пользователя ширину одного «блока» гистограммы BLOCK\_WIDTH. Не позволяйте вводить ширину блока менее 3px и более 30px.

Логика решения индивидуальной части задания:

Для решения данной задачи мы должны вручную ввести BLOCK\_WIDTH, который отвечает за ширину блока при выводе гистограммы в формате SVG.

Для этого введем дополнительную переменную, которую передадим в функцию show\_histogram\_svg( );

Таким образом гистограмма может быть меньше или больше в ширину

Программа

**Main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

using namespace std;

int main()

{

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(numbers, bin\_count);

size\_t F, BLOCK;

cerr << "Enter BLOCK\_WIDTH behind 3 and 30";

while ((F !> 3) && (F !< 30))

cin >> F;

BLOCK = F;

show\_histogram\_svg(bins, BLOCK);

return 0;

}

**Histogram.cpp**

#include "histogram.h"

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(vector<double> numbers, double& min, double& max)

{

min = numbers[0];

max = numbers[0];

for (auto number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

return;

}

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

{

double min, max;

find\_minmax(numbers, min, max);

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

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 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;

}

**Svg.cpp**

#include "svg.h"

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 fil)

{

cout << "<rect x='" << x << "' y='" << y << "' width='" << width << "' height='" << height

<< "' stroke='" << stroke << "' fill='" << fil << "' />";

}

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

{

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 = SHIRINA\_BLOCKA;

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

double top = 0;

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

for (size\_t bin : bins)

{

if (bin > max\_count)

{

max\_count = bin;

}

}

const size\_t SCREEN\_WIDTH = 100;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 50 - 20;

for (size\_t bin : bins)

{

const bool scalling\_needed = max\_count < MAX\_ASTERISK;

size\_t bin\_koeff = bin;

if (scalling\_needed)

{

const double koeff = (double)MAX\_ASTERISK / max\_count;

bin\_koeff = (size\_t)(bin \* koeff);

}

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

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

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

top += BIN\_HEIGHT;

}

svg\_end();

}

**Histogram.h**

#pragma once

#include <iostream>

#include <vector>

using namespace std;

vector<double> input\_numbers(size\_t count); // ввод значений

void find\_minmax(vector<double> numbers, double& min, double& max); // нахождение мин. и макс. значения

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

void show\_histogram\_text(vector<size\_t> bins); // отображает гистограмму в текстовом виде

**Svg.h**

#pragma once

#include <iostream>

#include <vector>

using namespace std;

void svg\_begin(double width, double height); // вывод загаловка

void svg\_end(); // окончание cvg

void svg\_text(double left, double baseline, string text); // вывод подписей к столбцам

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fil = "black"); // вывод прямоугольника в svg

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t SHIRINA\_BLOCKA); // вывод гистограммы в svg

**Test.cpp**

#include "histogram.h"

#include <cassert>

void test\_positive()//Случай массива полож. чисел

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

void test\_negative()//Случай массива отриц. чисел

{

double min;

double max;

find\_minmax({-1, -5,-228}, min, max);

assert(min == -228);

assert(max == -1);

}

void test\_one\_number()//массив из одного числа

{

double min;

double max;

find\_minmax({666}, min, max);

assert(min == 666);

assert(max == 666);

}

void test\_same\_numbers()//массив из трех одинаковых чисел

{

double min;

double max;

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

assert(min == 3);

assert(max == 3);

}

void test\_empty\_vector()//пустой массив

{

double min;

double max;

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

assert(min == 0);

assert(max == 0);

}

int main()//тестирование

{

test\_positive();

test\_negative();

test\_one\_number();

test\_same\_numbers();

test\_empty\_vector();

}