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 программы»

# 

# Выполнила студентка группы А-01-20

# Никитина Н.К.

# Проверили

# Мохов А. С

# Козлюк Д. А

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 15

Добавьте горизонтальную шкалу под гистограммой по аналогии с заданием лабораторной работы 1. Шкалу нужно разбить на интервалы, размер которых вводит пользователь. Допустимы размеры от 2 до 9 BLOCK\_WIDTH, при некорректном вводе печатайте сообщение со словом «ERROR» и завершайте работу программы. Под нулевой, первой и последней отметкой шкалы требуется напечатать соответствующие числа. Шкала должна быть во всю ширину гистограммы.

**Код программы**

***Laba3.cpp***

#include <iostream>

#include <string>

#include <vector>

#include "Header.h"

#include "svg.h"

using namespace std;

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

{

cerr << "Enter numbers: ";

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 count, double min, double max)

{

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;

}

void show\_histogram\_text(vector<size\_t> bins)

{

// Вывод данных

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';

}

}

int main()

{

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

// Ввод чисел заменен вызовом функции:

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

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

double min = numbers[0];

double max = numbers[0];

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

show\_histogram\_svg(bins);

return 0;

}

***Histogram.h***

#pragma once

#include <vector>

using namespace std;

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

**Histogram.cpp**

#include "Header.h"

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

{

if (numbers.size() != 0)

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

**Svg.h**

#pragma once

#include <vector>

#include <string>

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 = "red", string fill = "green");

void

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

***Svg.cpp***

*#include "svg.h"*

*#include <iostream>*

*#include <string>*

*using namespace std;*

*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\_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*

*svg\_text(double left, double baseline, string text) {*

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

*}*

*void*

*show\_histogram\_svg(const vector<size\_t>& bins) {*

*const auto IMAGE\_WIDTH = 410;*

*const auto MAX\_WIDTH = 350;*

*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;*

*svg\_begin(400, 300);*

*svg\_text(20, 20, to\_string(bins[0]));*

*svg\_rect(50, 0, bins[0] \* 10, 30, "red");*

*double top = 0; string red = "red"; string black = "black";*

*size\_t gap;*

*cout << "Enter gap size"; cin >> gap;*

*size\_t max\_count = 0;*

*for (size\_t count : bins) {*

*if (count > max\_count) {*

*max\_count = count;*

*}*

*}*

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

*for (size\_t bin : bins)*

*{*

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

*size\_t width = bin\_width;*

*if (scaling\_needed) {*

*const double scaling\_factor = (double)MAX\_WIDTH / (max\_count \* BLOCK\_WIDTH);*

*width = (bin\_width \* scaling\_factor);*

*}*

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

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

*cout << endl;*

*top += BIN\_HEIGHT;*

*}*

*cout << "<line x1 = '" << TEXT\_WIDTH << "' x2 = '" << TEXT\_WIDTH + (max\_count \* BLOCK\_WIDTH) << "' y1 = '" << top + 10 << "' y2 = '" << top + 10 << "' stroke = 'orange' stroke-width = '5'/>";*

*svg\_text(TEXT\_WIDTH, top + 25, "0");*

*//const double bin\_width1 = BLOCK\_WIDTH \* bins[0];*

*//svg\_text(bin\_width1, top + 25, to\_string(9));*

*size\_t final\_gap;*

*if (scaling\_needed)*

*{*

*final\_gap = 378;*

*}*

*else*

*{*

*final\_gap = max\_count \* BLOCK\_WIDTH;*

*}*

*svg\_text(TEXT\_WIDTH + 30, top + 25, "3");*

*svg\_text(final\_gap, top + 25, to\_string(max\_count));*

*svg\_end();*

*}*

***Test.cpp***

#include "histogram.h"

#include "string"

#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 same\_numbers()

{

double min = 0;

double max = 0;

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

assert(min == 5);

assert(max == 5);

}

void one\_number()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void empty\_array()

{

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

void gap\_error()

{

string str = "str";

int gap = 10;

if (gap < 2 || gap > 9)

{

str = "ERROR";

}

assert(str == "ERROR");

}

void scalling\_test()

{

bool scalling\_needed = false;

const int BLOCK\_WIDTH = 10;

const int MAX\_WIDTH = 350;

int max\_count = 20;

if ((max\_count \* BLOCK\_WIDTH) > MAX\_WIDTH)

{

scalling\_needed = true;

}

assert(scalling\_needed);

}

int

main() {

test\_positive();

test\_negative();

same\_numbers();

one\_number();

empty\_array();

}