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»

Выполнила студентка группы А-02-20

Ванина Александра Юрьевна

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

Ссылка на репозитарий: https://github.com/VaninaAY/cs-lab03

**Цель работы:**

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

# Задание:

# Часть 1. Декомпозиция программы функциями

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.

# Часть 2. Вывод гистограммы как изображения в формате SVG

Требуется вместо текстовой гистограммы рисовать картинку, например:

Изображение выглядит как стрела

Автоматически созданное описание

# Часть 3. Модульное тестирование

Написать модульный тест для функции поиска минимума и максимума.

**Индивидуальное задание. Вариант 6**

После запроса количества столбцов запросите цвет для каждого столбца.

**Логика решения варианта:**

Вводим 2 новые переменные типа string для цвета линий и цвета заливки, которым пользователь присваивает значения. Создаем функцию, которая проверят правильно ли введен цвет неверно, то программа завершается.

**Тесты для индивидуального задания:**

Тесты написаны для функции, которая проверяет правильно ли введен цвет пользователем. 1) тест, когда цвет введен верно 2) тест, когда цвет введен неверно

**Исходный код всех модулей:**

**main.cpp**

#include <iostream>

#include <vector>

using namespace std;

#include "histogram.h"

#include "svg.h"

#include "check.h"

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

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(const vector<double>& numbers, size\_t min, size\_t max, size\_t& bin\_count)

{

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

for (double number : numbers) {

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

if (bin == bin\_count) {

bin--;

}

result[bin]++;

}

return result;

}

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

{

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()

{

// Ввод данных

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

cerr << "Enter numbers: ";

const auto numbers = input\_numbers(number\_count); //функция ввовда чисел

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

string stroke;

cerr << "Enter stroke color (black, red, blue, green or yellow): ";

cin >> stroke;

string fill;

cerr << "Enter fill color (black, red, blue,green or yellow): ";

cin >> fill;

if(check\_color(stroke))

return 1;

if(check\_color(fill))

return 1;

// Обработка данных

// поиск min и max

double min, max;

find\_minmax(numbers, min, max);

// расчет количества чисел в столбцах гистограммы

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

// Вывод данных

// отображение гистограммы

show\_histogram\_svg(bins,stroke,fill);

return 0;

}

**histogram.cpp**

#include "histogram.h"

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

{

if (numbers.size() == 0)

return;

else

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

**histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

**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& fill)

{

cout << "<rect x='"<< x <<"' y='"<< y <<"' width='"<< width <<"' height='"<< height <<"' stroke='"<<stroke<<"' fill='"<< fill<<"'/>";

}

void show\_histogram\_svg(const vector<size\_t>& bins, string& stroke, string& fill)

{

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 SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

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

double top = 0;

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;

if (scaling\_needed)

{

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

for (size\_t bin : bins)

{

auto height = (size\_t)(bin \* scaling\_factor);

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, stroke, fill);

top += BIN\_HEIGHT;

}

}

else

{

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, fill);

top += BIN\_HEIGHT;

}

}

svg\_end();

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <iostream>

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, string& stroke, string& fill);

#endif // SVG\_H\_INCLUDED

**check.cpp**

#include "check.h"

bool check\_color (string color)

{

vector<string> colors {"black", "red", "blue", "green", "yellow"};

bool flag=true;

for(int i = 0; i < color.length(); i++)

{

color[i] = tolower(color[i]);

}

for (string x : colors)

{

if (color==x)

flag=false;

}

return flag;

}

**check.h**

#ifndef CHECK\_H\_INCLUDED

#define CHECK\_H\_INCLUDED

#include <vector>

#include <iostream>

using namespace std;

bool check\_color (string color);

#endif // CHECK\_H\_INCLUDED

**test.cpp**

#include "../histogram.h"

#include "../check.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);

}

void test\_negative()

{

double min = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void test\_same\_numbers()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_single()

{

double min=0;

double max=0;

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

assert(min == 1);

assert(max == 1);

}

void test\_empty ()

{

double min=0;

double max=0;

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

}

void test\_color\_1()

{

string stroke("RED");

bool result = check\_color(stroke);

assert(result == false);

}

void test\_color\_2()

{

string stroke("pink");

bool result = check\_color(stroke);

assert(result == true);

}

int main()

{

test\_positive();

test\_negative();

test\_same\_numbers();

test\_single();

test\_empty ();

test\_color\_1();

test\_color\_2();

}