Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Лабораторная работа №3**

**«Декомпозиция программы»**

Вариант 18

Задание выполнил: **Черемных Ю. А.**

Студент группы: **A-01-19**

Преподаватели: **Мохов А. С. , Козлюк Д. А.**

Москва

2020

# Цель работы

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

#### Вариант 18

Позволять пользователю делать оформление текста - подчеркивание, надчеркивание, зачеркивание текста. За оформление шрифта отвечает атрибут text-decoration. Сделать 'none' значением по умолчанию. Допустимые значения: none, underline, overline, line-through. Проверять введенной пользователем значение, и если оно не соответствует допустимым, запрашивать значение заново, выдавая предупреждение.

**Описание логики решения индивидуального варианта:**

Добавим в функцию *svg\_text* параметр text\_decoration, который отвечает за оформление текста. Предложим пользователю в main.cpp вводить это значение. Тесты *est\_text\_parametr\_false* и *est\_text\_parametr\_true* проверяеть корректноть рабоы функции *check\_text\_parametr*, которая возвращает true, если введены корректные значение оформления текста, и false — если некорректные.

**Исходный код всех модулей**

***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 massive"<<endl;

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

size\_t bin\_count;

cerr<<"Enter bin count ";

cin>>bin\_count;

string text\_decoration;

cerr << "Enter text decoration parametr";

cin >> text\_decoration;

if (check\_text\_parametr(text\_decoration) == false){

cerr << "Text parametr is incorrect. Please rewrite it and restart the program";

svg\_begin(1000,500);

svg\_text(20, 20, "Text parametr is incorrect. Please rewrite it and restart the program");

svg\_end();

return 1;

}

//MAKE HISTOGRAM

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

show\_histogram\_svg(bins,text\_decoration);

return 0;

}

------------------------------------------------------------------------------------------------------------------

histogram.cpp

#include "histogram.h"

#include <iostream>

#include <vector>

using namespace std;

vector<double>

input\_numbers(size\_t count) {

vector<double> result(count);

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

cin >> result[i];

}

return result;

}

bool check\_text\_parametr(string text\_decoration){

if((text\_decoration != "none") && (text\_decoration != "underline") && (text\_decoration != "overline") && (text\_decoration != "line-through")){

return false;

}

return true;

}

void

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

if (numbers.size() == 0){

return;

}

min = numbers[0];

max = numbers[0];

for (double x : numbers) {

if (x < min) {

min = x;

}

else if (x > max) {

max = x;

}

}

return;

}

vector<size\_t>

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

vector<size\_t> bins(bin\_count, 0);

double min, max;

find\_minmax(numbers, min, max);

for (double x : numbers){

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

if (bin\_index == bin\_count){

bin\_index--;

}

bins[bin\_index]++;

}

return bins;

}

void

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

for(size\_t bin : bins){

if (bin < 100) {

cout << ' ';

}

if (bin < 10) {

cout << ' ';

}

cout << bin << "|";

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

cout << "\*";}

cout << endl;

}

return;

}

--------------------------------------------------------------------------------------------------------------

histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

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

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

bool check\_text\_parametr(string text\_decoration);

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

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

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

#endif // HISTOGRAM\_H\_INCLUDED

-------------------------------------------------------------------------------------------------------------------------------------------------------

svg.h

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

void svg\_text(double left, double baseline, string text, string text\_decoration = "none");

void svg\_rect(double x, double y, double width, double height,

string stroke = "black", string fil = "black");

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

void svg\_end();

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

#endif

---------------------------------------------------------------------------------------------

svg.cpp

#include "histogram.h"

#include <iostream>

#include <vector>

using namespace std;

void

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

cout << "<text x='" << left << "' y='" << baseline << "' text-decoration='" << text\_decoration << "' >" << 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

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,string text\_decoration) {

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;

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 scaling\_needed = max\_count < MAX\_ASTERISK;

size\_t binkoeff = bin;

if (scaling\_needed) {

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

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

}

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

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

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

top += BIN\_HEIGHT;

}

// svg\_text(TEXT\_LEFT, TEXT\_BASELINE, to\_string(bins[0]));

// svg\_rect(TEXT\_WIDTH, 0, bins[0] \* BLOCK\_WIDTH, BIN\_HEIGHT);

svg\_end();

}

test.cpp

#include "histogram.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;

double max;

find\_minmax({-3, -7,-125}, min, max);

assert(min == -125);

assert(max == -3);

}

void

test\_one\_number(){

double min;

double max;

find\_minmax({532}, min, max);

assert(min == 532);

assert(max == 532);

}

void

test\_empty\_vector(){

double min =5;

double max =5 ;

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

assert(min == 5);

assert(max == 5 );

}

void

test\_same\_numbers(){

double min;

double max;

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

assert(min == 2);

assert(max == 2);

}

void

test\_text\_parametr\_true(){

assert(check\_text\_parametr("underline") == true);

}

void

test\_text\_parametr\_false(){

assert(check\_text\_parametr("podcherknuto") == false);

}

int main() {

test\_positive();

test\_negative();

test\_one\_number();

test\_empty\_vector();

test\_same\_numbers();

test\_text\_parametr\_true();

test\_text\_parametr\_false();

}

Ссылка на репозитарий: <https://github.com/yurchest/cslab3>