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 № 2**

# По курсу «Разработка ПО систем управления»

# «Декомпозиция и контроль корректности программ»

# Выполнил студент группы А-02-19:

# Китайский А.С.

# Проверили:

# Мохов А. С.

# Козлюк Д. А.

**Цель работы**

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

**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.

**Описание хода решения:**

В блоке текста <text></text> оформил атрибут font-size, который меняет размер шрифта. Далее внутри функции show\_histogram\_svg сделал запрос на размер шрифта. Затем сделал функцию, которая проверяет: входит ли введенное пользователем значение в промежуток от 8 до 32. Если нет, то печатается ошибка. При вызове функции, передается значение либо true, либо false. Если flag=false, то рисуется гистограмма с размером шрифта. Также ввел значение по умолчанию.

**Ссылка на репозитарий:** <https://github.com/ArtemKitayskiy/cs-lab03>

**Svg.cpp:**

#include "svg.h"

#include <vector>

#include <iostream>

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 check\_size(double& font\_size, bool& flag)

{

flag=true;

if(font\_size < 8 || font\_size > 32)

{

cerr << "Error. Enter subject to conditions";

flag=false;

}

}

void svg\_text(double left,double baseline,string text,double font\_size=12)

{

cout<< "<text x='" << left << "' y='"<< baseline << "' font-size='" << font\_size <<"' > "<< text <<"</text>";

}

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black")

{

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

bool flag;

double font\_size;

cerr << "Enter font size: ";

cin >> font\_size;

check\_size(font\_size,flag);

if(flag=true)

{

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\_ASTERISK;

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

double top=0;

for(size\_t bin:bins)

{

size\_t height = bin;

if (scaling\_needed)

{

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

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

}

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

svg\_text(TEXT\_LEFT,top+TEXT\_BASELINE,to\_string(bin), 12);

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

top+=BIN\_HEIGHT;

}

svg\_end();

}

}

**Svg.h:**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <string>

#include <vector>

using namespace std;

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

void svg\_end();

void check\_size(double& font\_size, bool& flag);

void svg\_text(double left, double baseline, string text, double font\_size);

void svg\_rect(double x, double y, double width, double height, string stroke , string fill);

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

#endif // SVG\_H\_INCLUDED

**Histogram.cpp:**

#include "histogram.h"

#include <vector>

#include <iostream>

using namespace std;

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

{

int p=numbers.size();

if(p!=0)

{

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 <string>

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

**Main.cpp:**

#include "histogram.h"

#include "svg.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;

}

vector<size\_t> make\_histogram(const vector<double>& numbers, size\_t count)

{

double min=0;

double max=0;

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(const 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';

}

}

\*/

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);

show\_histogram\_svg(bins);

return 0;

}

**Tests:**

#include "histogram.h"

#include "svg.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\_equal()

{

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);

assert(min == 0);

assert(max == 0);

}

void test\_dz1()

{

double font\_size=8;

bool flag=true;

check\_size(font\_size, flag);

assert(flag=true);

}

void test\_dz2()

{

double font\_size=1;

bool flag=true;

check\_size(font\_size, flag);

assert(flag=true);

}

int main()

{

test\_positive();

test\_negative();

test\_equal();

test\_single();

test\_empty();

test\_dz1();

test\_dz2();

}