НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

МЭИ

ЛАБОРАТОРНАЯ РАБОТА

по дисциплине «Разработка программного обеспечения систем управления»

тема:

«Основы языка С++»

Студент Филиппова Ирина Вадимовна

Группа А-3-18

2019 г.

# Цель работы

1. Уметь структурировать программу при помощи функций и структур.

2. Уметь писать модульные тесты.

# Задание

Выделить части программы из ЛР №1 в функции, переделать вывод гистограммы на формат SVG, написать модульный тест для функции поиска минимума и максимума.

Реализовать для гистограммы в SVG масштабирование. Выделить в модуль все функции для работы с SVG.

**Логика решения**

Исходя из длины поля, рассчитывается коэффициент пропорциональности, на который домножается каждый элемент вектора bins, так что соотношение длины столбцов остается постоянным.

В заголовочный файл с функциями для работы с SVG помещаются стражи включения, объявления функций и необходимые библиотеки. В файл реализации подключается заголовочный файл и помещаются описания функций. В main.cpp подключается заголовочный файл.

**Репозитарий**

https://github.com/FilippovaIV/lab03

**Код**

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

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

{

double h;

double min,max;

find\_minmax(numbers, min, max);

h=(max-min)/bin\_count;

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

for (double number: numbers)

{

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

{

if ((number>=(min+h\*(i))) && (number<(min+h\*(i+1))))

{

bins[i]++;

}

if (i>=bin\_count)

{

i=bin\_count-1;

}

}

if (number==max)

{

bins [bin\_count-1]++;

}

}

return bins;

}

int main()

{

string da\_net;

da\_net="n";

size\_t number\_count;

cin >> number\_count;

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

size\_t bin\_count;

do

{

cerr<<"Enter bin count\n";

cin>>bin\_count;

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

show\_histogram\_svg(bins);

cerr<<"WI dowolny? (da/net)\n";

cin>>da\_net;

}

while (da\_net!="da");

return 0;

}

**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

**histogram.cpp**

#include "histogram.h"

void

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

{

if (numbers.size()==0)

{

min=0;

max=0;

}

else

{

min = numbers[0];

max=numbers[0];

for (int i=0; i<numbers.size(); i++)

{

if(min>numbers[i])

{

min=numbers[i];

}

if (max<numbers[i])

{

max=numbers[i];

}

}

}

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

double mashtab (vector <size\_t> bins);

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);

#endif // SVG\_H\_INCLUDED

**svg.cpp**

#include "svg.h"

double

mashtab (vector <size\_t> bins)

{

size\_t maxbin;

const size\_t MAX\_WiDTH = 350;

maxbin=0;

for (size\_t bin : bins)

{

if (bin > maxbin)

{

maxbin = bin;

}

}

double koef;

koef=1;

maxbin = maxbin \* 10;

if (maxbin > MAX\_WiDTH)

{

koef=(static\_cast<double>(MAX\_WiDTH)/maxbin);

}

return koef;

}

void

svg\_begin(double width, double height)

{

cout << "<?xml version='1.0' encoding='UTF-8'?>\n";

cout << "<svg width='" << width << "' height='" << height << "' "

<< "viewBox='0 0 " << width << " " << height << "' "

<< "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 = "red", string fill = "blue")

{

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

}

void

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

{

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;

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

const auto koef = mashtab(bins);

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

double top = 0;

for (size\_t bin : bins)

{

const double bin\_width = 10 \* bin \* koef;

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

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

top += 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 = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void

test\_same()

{

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void

test\_one()

{

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);

}

int

main()

{

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

}