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»

# Выполнил студент группы А-01-19

# Антимонов М.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

**Задание.**

# 1) Написать программу для построения гистограммы массива чисел как изображения в формате SVG

2) Доработать программу в соответствии с вариантом.

#### Вариант 1

Дайте пользователю возможность задавать произвольную ширину столбца гистограммы вместо 400. Считайте некорректной ширину менее 70, более 800  или менее трети количества чисел, умноженных на ширину блока (BLOCK\_WIDTH) — предлагайте пользователю ввести ширину заново с указанием причины.

**Пояснение к программе**

Проверяю число, которое вводит пользователь. Сравниваю его с условием:

1. если число меньше 70, то выдаю ошибку "Число меньше 70".
2. Если число больше 800, то выдаю ошибку "Число больше 800 ".

КОД ПРОГРАММЫ

***main.cpp***

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

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 bin\_count)

{

double min;

double max;

find\_minmax(numbers,min,max);

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

for (double number : numbers)

{

size\_t bin;

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

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

int main()

{

size\_t number\_count;

size\_t IMAGE\_WIDTH;

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;

double min, max;

find\_minmax(numbers, min, max);

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

show\_histogram\_svg(bins, number\_count);

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);**

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

**#endif // HISTOGRAM\_H\_INCLUDED**

***HISTOGRAM.CPP***

#include "histogram.h"

#include <vector>

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;

}

}

}

}***SVG.H***

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <string.h>

#include <vector>

#include <iostream>

using namespace std;

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

void

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

void

svg\_begin(double width, double height);

void

svg\_end();

double three( size\_t number\_count, size\_t BLOCK\_WIDTH);

double shirina( size\_t numer\_count);

void

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

#endif // SVG\_H\_INCLUDED

***SVG.CPP***

#include <iostream>

#include <vector>

#include "svg.h"

using namespace std;

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

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

{

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

}

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";

}

double three( size\_t number\_count, size\_t BLOCK\_WIDTH)

{

return ((number\_count\*BLOCK\_WIDTH)/3.0) ;

}

double shirina( size\_t number\_count)

{

const auto BLOCK\_WIDTH = 10;

size\_t IMAGE\_WIDTH;

do

{

cerr << "Enter IMAGE\_WIDTH: ";

cin >> IMAGE\_WIDTH;

if ( IMAGE\_WIDTH < 70 )

{

cerr << "< 70" << endl;

}

if (IMAGE\_WIDTH > 800 )

{

cerr << "> 800 " << endl;

}

if ( IMAGE\_WIDTH < three(number\_count,BLOCK\_WIDTH ))

{

cerr << three(number\_count,BLOCK\_WIDTH ) << endl;

}

}

while (IMAGE\_WIDTH < 70 || IMAGE\_WIDTH > 800 || IMAGE\_WIDTH < (number\_count\*BLOCK\_WIDTH)/3.0);

return IMAGE\_WIDTH;

}

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t number\_count)

{

size\_t IMAGINE\_WIDTH=shirina (number\_count);

const auto BLOCK\_WIDTH = 10;

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;

double top = 0;

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

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,"chartreuse","#7FFF00");

top += BIN\_HEIGHT;

}

svg\_end();

}

***TEST.CPP***

#include "C:\Users\markus\Desktop\lab03\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\_onenumber()

{

double min = 0;

double max = 0;

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

assert(min == 2);

assert(max == 2);

}

void

test\_empty()

{

double min = 0;

double max = 0;

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

}

int

main()

{

test\_positive();

test\_negative();

test\_same();

test\_onenumber();

test\_empty();

}

***TEST\_DZ.CPP***

#include "C:\Users\marka\Desktop\lab03\svg.h"

#include <cassert>

#include <string.h>

void test1()

{

double abc;

abc=three (3,9);

assert(abc==9);

}

int main ()

{

test1();

}