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-02-19

Носкова А.В

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 20 (2)**

Задавать автоматически яркость заливки каждого столбца гистограммы в градациях серого в зависимости от высоты столбца. Чем больше столбец, тем темнее заливка.

Сделать это можно, передавая цвет в параметр fill в формате "#RGB" ([red, green, blue](http://htmlbook.ru/html/value/color)). "#111" — самый темный, "#222" — чуть менее темный, …, "#EEE" — практически белый, "#FFF" — белый. В лабораторной работе использовать диапазон цветов от "#111" для самого большого столбца до "#999" для самого маленького столбца. Поскольку используются градации серого, расчет сводится к вычислению только одного значения и дублированию этого значения в качестве цвета каждого из каналов (Red, Green, Blue). Для расчета цвета *i*-го столбца bins[i] использовать формулу (10 - (bins[i] \* 9) / max\_count). По ней мы получаем значение цвета одного канала (от 1 до 9), который затем записываем три раза.

Пример с текстом вместо SVG:

1|▮ — цвет #999  
5|▮▮▮▮▮ — цвет #111  
3|▮▮▮ — цвет #555

**Описание решения индивидуального задания:**

Записала функции для поиска столбцов с максимальной и минимальной длиной (find\_max\_column и find\_min\_column), а также функцию для автоматического определения яркости заливки каждого столбца гистограммы в градациях серого в зависимости от высоты столбца (create\_color\_column). Если столбец имеет максимальную длину, то у него будет цвет «#999», если минимальную – «#111», в другом случае для расчета цвета *i*-го столбца, я использовала формулу, предложенную в условии варианта. В последней функции использовала класс – ostringstream, так как он предоставляет удобный интерфейс потоков ввода-вывода и обрабатывает все встроенные типы данных. Результирующую строку получила с помощью метода str().

**Код:**

**Main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include "lab03.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 count)

{

vector<size\_t> result(count);

double max, min;

find\_minmax(numbers, min, max);

for (double number : numbers)

{

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

if (bin == count)

{

bin--;

}

result[bin]++;

}

return result;

}

int main()

{

size\_t number\_count;

cin >> number\_count;

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;

}

**Lab03.cpp**

#pragma once

#include <vector>

#include <iostream>

#include <vector>

#include "lab03.h"

using namespace std;

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

{

if (numbers.size()!=0)

{

min = numbers[0];

max = numbers[0];

for(double number: numbers)

{

if(min>number)

min=number;

if(max<number)

max=number;

}

}

}

**Lab03.h**

#ifndef LAB03\_H\_INCLUDED

#define LAB03\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

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

#endif // LAB03\_H\_INCLUDED

**Svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <string>

#include <vector>

#include <iostream>

using namespace std;

using namespace std;

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

void svg\_end();

void svg\_text(double left, double baseline, size\_t text);

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

size\_t find\_max\_column(const vector<size\_t>& bins);

size\_t find\_min\_column(const vector<size\_t>& bins);

string create\_color\_column(const vector<size\_t>& bins, size\_t max\_count, size\_t bin);

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

#endif // SVG\_H\_INCLUDED

**Svg.cpp**

#include "svg.h"

#include <vector>

#include <iostream>

#include <string>

#include <sstream> // for stringsteam

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, size\_t 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 << "' />";

}

size\_t find\_max\_column(const vector<size\_t>& bins)

{

size\_t max\_column = bins[0];

for (size\_t bin : bins)

{

if (max\_column<bin)

max\_column=bin;

}

return max\_column;

}

size\_t find\_min\_column(const vector<size\_t>& bins)

{

size\_t min\_column = bins[0];

for (size\_t bin : bins)

{

if (min\_column<bin)

min\_column=bin;

}

return min\_column;

}

string create\_color\_column(const vector<size\_t>& bins, size\_t max\_count, size\_t bin)

{

size\_t max = find\_max\_column(bins);

size\_t min = find\_min\_column(bins);

size\_t color;

if (bin == max)

{

color = 111;

}

else if (bin == min)

{

color = 999;

}

else

{

color = (10 - (bin \* 9) / max\_count)\*111;

}

ostringstream c;

c << color;

string color\_column = c.str();

return color\_column;

}

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;

const auto BLOCK\_WIDTH = 10;

const size\_t MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_LEFT - TEXT\_WIDTH;

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;

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

for (size\_t bin : bins)

{

string color\_column = create\_color\_column(bins, max\_count, bin);

cout << color\_column << endl;

size\_t height = bin;

if (scaling\_needed)

{

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

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

}

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "white", "#" + color\_column);

top += BIN\_HEIGHT;

}

svg\_end();

}

**Lab03-test/test.cpp**

#include "lab03.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({2, 2, 2}, min, max);

assert(min == 2);

assert(max == 2);

}

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);

}

void test\_homework\_1 ()

{

vector<size\_t> bins = {3,4,7,21};

assert(create\_color\_column(bins,21,bins[0]) == "999");

assert(create\_color\_column(bins,21,bins[1]) == "888");

assert(create\_color\_column(bins,21,bins[2]) == "777");

assert(create\_color\_column(bins,21,bins[3]) == "111");

}

void test\_homework\_2 ()

{

vector<size\_t> bins = {1,9,7};

assert(create\_color\_column(bins,9,bins[0]) == "999");

assert(create\_color\_column(bins,9,bins[1]) == "111");

assert(create\_color\_column(bins,9,bins[2]) == "333");

}

void test\_homework\_3 ()

{

vector<size\_t> bins = {5,6};

assert(create\_color\_column(bins,6,bins[0]) == "999");

assert(create\_color\_column(bins,6,bins[1]) == "111");

}

int

main()

{

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

}

**Ссылка:** <https://github.com/NoskovaAV/cs-lab03>