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-02-19

Баснак Е.А.

Проверили

Мохов А. С.

Козлюк Д. А.

Москва 2020

**Вариант 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.

***Логика решения:***

Я написал 3 функции: find\_min, find\_max и make\_color. Первые 2 находят минимум и максимум вектора bins, т.е. максимальную и минимальную длину столбца. make\_color выдает цвет для введённой длины столбца. Если столбец минимальной длины, то цвет “#999”, если максимальной – “#111”, иначе цвет вычисляется по формуле.

***Код:***

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include "histogram.h"

#include "svg.h"

using namespace std;

vector<double> input\_numbers(const 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, const size\_t count) {

vector<size\_t> result(count);

double min;

double max;

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;

}

void show\_histogram\_text(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() {

// Ввод данных

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;

// Обработка данных

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

// Вывод данных

show\_histogram\_svg(bins);

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.cpp**

#include<vector>

#include"histogram.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 (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

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");

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

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

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

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

#endif // SVG\_H\_INCLUDED

**svg.cpp**

#include "svg.h"

//#include "histogram.h"

#include <vector>

#include <string>

#include <iostream>

#include <sstream>

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

size\_t min = bins[0];

for (const auto& bin : bins)

{

if (bin < min)

{

min = bin;

}

}

return min;

}

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

size\_t max = bins[0];

for (const auto& bin : bins)

{

if (bin > max)

{

max = bin;

}

}

return max;

}

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

ostringstream os;

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

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

size\_t x;

if (bin == max)

{

x = 1;

}

else if (bin == min)

{

x = 9;

}

else

{

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

}

os << x;

string color = os.str();

color += color + color;

return color;

}

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;

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) {

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

cout << endl << color << endl;

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, bin);

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

top += BIN\_HEIGHT;

}

svg\_end();

}

**test.cpp**

#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\_same() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_one() {

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);

assert(min == 0);

assert(max == 0);

}

void test\_color() {

// Test 1

vector<size\_t> bins = {1, 5, 3};

assert(make\_color(bins, bins[0], 5) == "999");

assert(make\_color(bins, bins[1], 5) == "111");

assert(make\_color(bins, bins[2], 5) == "555");

//Test 2

bins = {10, 6, 3, 5};

assert(make\_color(bins, bins[0], 10) == "111");

assert(make\_color(bins, bins[1], 10) == "555");

assert(make\_color(bins, bins[2], 10) == "999");

assert(make\_color(bins, bins[3], 10) == "666");

}

void test\_all() {

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

test\_color();

}

int

main() {

test\_all();

}

***Ссылка на репозитарий:***

https://github.com/BasnakEA/cs-lab03