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»**

Выполнил студент группы А-03-20

Панкратов Евгений Павлович

Проверил

Мохов А.С.

Козлюк Д.А.

Москва 2021

Цель работы:

* Уметь структурировать программу при помощи функций.
* Уметь писать модульные тесты.

***Вариант 17***

***Задавать автоматически прозрачность заливки каждого столбца гистограммы в зависимости от высоты столбца. Чем больше столбец, тем темнее заливка. Сделать это можно, передавая процент прозрачности в параметр fill-opacity в формате "0.7". 1 соответствует отсутствию прозрачности, 0 соответствует полной прозрачности (отсутствию цвета)***

***Для расчета прозрачности каждого i-го столбца bins[i] использовать формулу (bins[i]) / max\_count).***

***Пример:***

***1|▮ — прозрачность 0.2***

***5|▮▮▮▮▮ — прозрачность 1.0***

***3|▮▮▮ — прозрачность 0.6***

**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 (const vector<double>& numbers, const size\_t& number\_count, const size\_t& bin\_count)

{

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

double min, max;

find\_minmax(numbers, min, max);

double bin\_size = (max - min) / bin\_count;

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

{

bool found = false;

for (size\_t j = 0; (j < bin\_count - 1) && !found; j++)

{

auto lo = min + j \* bin\_size;

auto hi = min + (j + 1) \* bin\_size;

if ((lo <= numbers[i]) && (numbers[i] < hi))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[bin\_count - 1]++;

}

}

return bins;

}

int

main()

{

size\_t number\_count;

cerr << "Enter 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, number\_count, bin\_count);

show\_histogram\_svg(bins);

return 0;

}

**Код 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

**Код histogram.cpp**

#include <vector>

using namespace std;

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

{

if (numbers.size() == 0) return;

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

**Код 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\_niggative()

{

double min = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void

test\_repeat()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void

test\_once()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void

test\_empty()

{

double min = -4;

double max = 6;

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

assert(min == -4);

assert(max == 6);

}

int

main()

{

test\_positive();

test\_niggative();

test\_repeat();

test\_once();

test\_empty();

}

***Код svg.cpp***

#include <iostream>

#include <vector>

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, string text)

{

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

}

void svg\_rect(double x, double y, double width, double height)

{

cout << "<rect x='"<< x <<"' y='"<< y <<"' width='"<< width <<"' height='"<< height <<"'/>\n";

}

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<<"'/>\n";

}

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

{

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

fill<<"' fill-opacity ='"<< fill\_opacity <<"'/>\n";

}

double

opacity\_factor (size\_t bin, size\_t max\_count)

{

double opacity = (double)bin / max\_count;

return opacity;

}

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 auto MAX\_COUNT = 34;

double top = 0;

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

size\_t count;

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

for (size\_t bin : bins)

{

if (bin > max\_count)

{

max\_count = bin;

}

}

if (max\_count > MAX\_COUNT)

{

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

for (size\_t bin : bins)

{

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

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "blue", "#ff2233", opacity\_factor(bin, max\_count));

top += BIN\_HEIGHT;

}

svg\_end();

}

else

{

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, "blue", "#aaffaa", opacity\_factor(bin, max\_count));

top += BIN\_HEIGHT;

}

svg\_end();

}

}

***Код 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, string text);

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

void svg\_rect(double x, double y, double width, double height,

/\*string stroke, string fill\*/ string stroke = "black", string fill = "black");

double

opacity\_factor (size\_t bin, size\_t max\_count);

void

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

#endif // SVG\_H\_INCLUDED

***Ссылка на репозитарий :***

[***https://github.com/PankratovEP/cs-lab03***](https://github.com/PankratovEP/cs-lab03)