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»

«Основы языка С++»

# Выполнил студент группы А-01-19

# Кононенко Александр Вячеславович

# Проверили:

# Мохов А. С

# Козлюк Д. А

Москва 2020

Задача: разработать программу для работы с svg файлом, построить в нём гистограмму из ЛР1.

Логика решения индивидуального варианта 7:

Задача состояла в том, чтобы найти среднее значение и проверить по количеству элементов в каждой корзине, превышает ли оно среднее значение. Если превышает, то красим столбец в красный, нет – в зелёный.

Среднее значение равно частному количества чисел и количества корзин: avg\_bin = number\_count / bin\_count. Это значение передаётся в функцию show\_histogram\_svg и в ней проверяется условие: количество элементов в корзине меньше или равны, среднему значению. Если условие выполняется, то меняем цвет заливки и контура на зелёный, нет – на красный.

Код программы:

**main.cpp**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "avg\_bin+scaling.h"

using namespace std;

int main()

{

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;

double min, max;

find\_minmax(numbers, min, max);

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

double scaling = scale(bins);

size\_t avg\_bin = average\_bin(number\_count, bin\_count);

show\_histogram\_svg(bins, scaling, avg\_bin);

return 0;

}

**histogram.cpp**

#include "histogram.h"

#include "svg.h"

#include <iostream>

#include <vector>

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 = numbers[0];

double max = numbers[0];

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;

}

void show\_histogram\_svg(const vector<size\_t>& bins, double scaling, size\_t &avg\_bin)

{

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;

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

double top = 0;

for (size\_t bin : bins)

{

string stroke, fill;

if (bin <= avg\_bin)

{

stroke = "green";

fill = "#00FF00";

}

else

{

stroke = "red";

fill = "#FF0000";

}

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

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

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

top += BIN\_HEIGHT;

}

svg\_end();

}

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;

}

}

}

else cerr << "ERROR\n";

}

**avg\_bin+scaling.cpp**

#include <iostream>

#include <vector>

#include "avg\_bin+scaling.h"

double scale (const vector <size\_t>& bins)

{

const size\_t MAX\_ASTERISK = 40;

size\_t max\_count = 0;

for (size\_t bin:bins)

{

if (max\_count < bin)

{

max\_count = bin;

}

}

double scaling = 1;

if (max\_count > MAX\_ASTERISK)

{

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

}

return scaling;

}

size\_t average\_bin (size\_t number\_count, size\_t bin\_count)

{

size\_t avg\_bin = number\_count / bin\_count;

return avg\_bin;

}

**svg.cpp**

#include <string>

#include <iostream>

#include <fstream>

#include "svg.h"

#define SVG "bin\\debug\\marks.svg"

using namespace std;

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

{

ofstream fout(SVG, ios\_base::app);

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

fout << "\n";

fout.close();

}

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

{

ofstream fout(SVG, ios\_base::app);

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

fout << "\n";

fout.close();

}

void svg\_begin(double width, double height)

{

ofstream fout(SVG);

if (!fout.is\_open())

{

cout << "ERROR2";

}

else

{

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

fout << "<svg ";

fout << "width='" << width << "' ";

fout << "height='" << height << "' ";

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

fout << "xmlns='http://www.w3.org/2000/svg'>\n";

}

fout.close();

}

void svg\_end()

{

ofstream fout(SVG, ios\_base::app);

fout << "</svg>\n";

fout.close();

}

**test.cpp //тестирование осуществляется в codeblocks**

void

test\_empty()

{

double min = 0;

double max = 0;

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

}

void

test\_equal()

{

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void

test\_double()

{

double min = 0;

double max = 0;

find\_minmax({2.5,3.33,5.89},min,max);

assert(min == 2.5);

assert(max == 5.89);

}

void

test\_svg(ifstream &fin)

{

size\_t number\_count;

fin >> number\_count;

vector <double> numbers (number\_count, 0);

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

fin >> numbers[i];

size\_t bin\_count;

fin >> bin\_count;

fin.close();

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

double scaling = scale(bins);

size\_t avg\_bin = average\_bin(number\_count, bin\_count);

show\_histogram\_svg(bins, scaling, avg\_bin);

}

int

main()

{

test\_positive();

test\_negative();

test\_equal();

test\_double();

test\_empty();

ifstream fin1("bin\\Debug\\marks1.txt");

test\_svg(fin1);

Sleep(10000); //в течение 10 секунд тестировщик должен посмотреть результат вывода для файла fin1, затем, после окончания действия программы, снова проверить svg файл

ifstream fin2("bin\\Debug\\marks2.txt");

test\_svg(fin2);

}