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 № 2**

# По курсу «Разработка ПО систем управления»

# «Основы языка С++»

# Выполнил студент группы А-01-19

# Афанасьева А.М.

# 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.

**Пояснение к программе**

Для написания программы я создала три новые функции, две из которых предназначены для поиска максимальной и минимальной длины столбца (find\_max, find\_min). Третья функция (color\_bins) предназначена для задания яркости заливки каждого столбца гистограммы. Существует определенный диапазон цветов от "#111" до "#999", который определяется в зависимости от дины столбца, то есть, если столбец минимальной длины, то его цвет "#111", а если столбец максимальной длины то его цвет "#999", в противном случае он рассчитывается по формуле, указанной в задании.

***Текст программы.***

**Main.cpp**

#include <iostream>

#include <vector>

#include <cmath>

#include <string>

#include "histogramm.h"

#include "svg.h"

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 3 - 1;

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

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

void make\_histogram(const vector<double>& numbers, double max, double min, size\_t bin\_count, vector <size\_t>& bins )

{

for(double x: numbers)

{

size\_t bin\_index=(x-min)/(max-min)\*bin\_count;

if (bin\_index==bin\_count)

{

bin\_index--;

}

bins[bin\_index]++;

}

}

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

{

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

for (size\_t bin:bins)

{

if (bin > max\_bin)

{

max\_bin=bin;

}

}

if (max\_bin > MAX\_ASTERISK)

{

double factor = MAX\_ASTERISK/static\_cast<double>(max\_bin);

for (size\_t bin:bins)

{

if (bin <100)

{

cout <<" ";

if (bin < 10)

{

cout <<" ";

}

}

cout <<bin <<"|";

size\_t height = bin\*factor;

for(int i=0; i<height; i++)

{

cout<< "\*";

}

cout<<endl;

}

}

else

{

for (size\_t bin:bins)

{

if (bin <100)

{

cout <<" ";

if (bin < 10)

{

cout <<" ";

}

}

cout <<bin <<"|";

for (int i = 0; i < bin; i++)

{

cout << "\*";

}

cout<<endl;

}

}

}

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 bin count:";

cin >> bin\_count;

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

double left;

double baseline;

double min, max;

find\_minmax(numbers, min, max);

make\_histogram ( numbers, max, min, bin\_count, bins );

show\_histogram\_svg(bins);

return 0;

}

**Histogramm.cpp**

#include "histogramm.h"

using namespace std;

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

{

min = numbers[0];

max = numbers[0];

for(double x: numbers)

{

if(min>x)

min=x;

if(max<x)

max=x;

}

}

**Histogramm.h**

#ifndef HISTOGRAMM\_H\_INCLUDED

#define HISTOGRAMM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAMM\_H\_INCLUDED

**Svg.cpp**

#include <iostream>

#include <vector>

#include <cmath>

#include <string>

#include <sstream> /\*библиотека, отвечающая за ввод и вывод строк\*/

#include "svg.h"

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\_max(const vector<size\_t>& bins)/\*определяем максимальную длину столбца \*/

{

size\_t max = bins[0];

for (const auto& bin : bins)

{

if (bin > max)

{

max = bin;

}

}

return max;

}

size\_t find\_min(const vector<size\_t>& bins)/\*определяем минимальную длину столбца \*/

{

size\_t min = bins[0];

for (const auto& bin : bins)

{

if (bin < min)

{

min = bin;

}

}

return min;

}

string color\_bins(const vector<size\_t>& bins, size\_t max\_count, size\_t bin) /\*функция, меняющая насыщенность заливки\*/

{

ostringstream tipe; /\*используем для записи в стоку\*/

size\_t a;

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

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

if (bin == min)

{

a = 9;

}

else if (bin == max)

{

a = 1;

}

else

{

a =10 - (bin \* 9) / max\_count; /\*используем для расчета цвета i-го столбца bins[i]\*/

}

tipe << a; /\*содержимое строки a \*/

string colors = tipe.str(); /\*содержимое а записывается в строку colors\*/

colors = colors + colors + colors;

return colors;

}

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 b : bins)

{

if (b > max\_count)

{

max\_count = b;

}

}

const bool need\_scaling = max\_count \* BLOCK\_WIDTH > MAX\_ASTERISK;

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

double top = 0;

for (size\_t bin : bins)

{

string colors = color\_bins(bins, max\_count, bin);

cout << colors << endl;

size\_t height = bin;

if (need\_scaling)

{

const double scal = (double)MAX\_ASTERISK / (max\_count \* BLOCK\_WIDTH);

height = (size\_t)(bin \* scal);

}

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

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

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

top += BIN\_HEIGHT;

}

svg\_end();

}

**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, string stroke , string fill );

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

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

string color\_bins(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

**Lab03-test**

#include "histogramm.h"

#include <cassert>

#include <string.h>

void test\_positive()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

void test1()

{

double min = 0;

double max = 0;

find\_minmax({-5, -8, -10}, min, max);

assert(min == -10);

assert(max == -5);

}

void test2()

{

double min = 0;

double max = 0;

find\_minmax({5, 5, 5}, min, max);

assert(min == 5);

assert(max == 5);

}

void test3()

{

double min = 0;

double max = 0;

find\_minmax({6}, min, max);

assert(min == 6);

assert(max == 6);

}

void test4()

{

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

void test5()

{

vector <size\_t> bins={20,5,25,30,24};

assert(color\_bins(bins, 15, bin[0])=="111");

assert(color\_bins(bins, 15, bin[1])=="555");

assert(color\_bins(bins, 15, bin[2])=="444");

assert(color\_bins(bins, 15, bin[3])=="999");

assert(color\_bins(bins, 15, bin[4])=="666");

}

void test6()

{

vector <size\_t> bins={5,10,15};

assert(color\_bins(0, 5, bin[0])=="999");

assert(color\_bins(0, 5, bin[1])=="111");

assert(color\_bins(0, 5, bin[2])=="555");

}

int main()

{

test\_positive();

test1();

test2();

test3();

test4();

test5();

test6();

}