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

Поташов С. Е

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

**Вариант 13**

Отображайте гистограмму вертикально с подписями снизу.

\*

\*

\* \*

\* \*

\* \* \*

\_ \_ \_

3 5 1

Предусмотреть расчет IMAGE\_HEIGHT таким образом, чтобы вся гистограмма вмещалась в область рисунка.

**Решение**

 В show\_histogram\_svg() я поменял местами расчет координат X и Y, а также ширины и высоты столбцов. При фиксированной высоте изображения смещение Y для подписей всегда одинаково. Координата Y для левого верхнего угла и высота каждого столбца определяются значением bins[i]: чем больше bins[i], тем меньше Y и больше высота.

Сначала я расчитывал, какая высота рисунка мне нужна. Она равняется высоте самого большого столбца плюс какие-то отступы для подписей.  
Далее я указывал только точку отсчтета прямоугольника, его высоту и ширину. Точку отсчета я каждый раз сдвигал на ширину столбца, которая мне известна, а высота известна из bins[i].

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() {

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;

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

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\_H\_INCLUDED

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");

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

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

#endif // SVG\_H\_INCLUDED

Svg.cpp:

#include "svg.h"

#include <vector>

#include <string>

#include <iostream>

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 filll) {

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

}

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

size\_t max = bins[0];

for (const auto& bin : bins)

{

if (bin > max)

{

max = bin;

}

}

return max;

}

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

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 10;

const auto TEXT\_BASELINE = 20;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 10;

double TEXT\_WIDTH = 0;

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;

double height\_max = find\_max(bins)+20 ;

for (size\_t bin : bins) {

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 hight = BLOCK\_WIDTH \* bin;

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

svg\_rect(TEXT\_WIDTH, height\_max - height, BIN\_HEIGHT, height,"red","#aab5ff");

top += BIN\_HEIGHT;

TEXT\_WIDTH+= 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\_findmax() {

//TEST1

double max = find\_max({2, 5, 3});

assert(max == 5);

//TEST2

vector<size\_t> bins = {100, 4, 0, 25};

assert (find\_max(bins) == 100);

}

int

main() {

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

test\_findmax();

}