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-03-19

Бязров Г.Г.

Проверили

Москва 2020

**Вариант 1**

Дайте пользователю возможность задавать произвольную ширину столбца гистограммы вместо 400. Считайте некорректной ширину менее 70, более 800 или менее трети количества чисел, умноженных на ширину блока (BLOCK\_WIDTH) — предлагайте пользователю ввести ширину заново с указанием причины.

***Логика решения:***

При введении ширины столбца, не удовлетворяющей условиям из задания, вызывается функция custom-width. Функция custom-width выводит сообщение об ошибке.

***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, number\_count);**

**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");

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

string custome\_width(double IMAGE\_WIDTH,double BLOCK\_WIDTH,size\_t number\_count);

#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 << "' />";

}

string custome\_width(double width, double BLOCK\_WIDTH, size\_t number\_count) {

string str = "";

if (width < 70)

str = "Width < 70. Enter the width again:";

else if (width > 800)

str = "Width > 800. Enter the width again:";

else if (width < BLOCK\_WIDTH\*number\_count/3)

str = "Width < BLOCK\_WIDTH\*number\_count/3. Enter the width again:";

return str;

}

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

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 double BLOCK\_WIDTH = 10;

double IMAGE\_WIDTH;

cerr << "Enter width:";

cin >> IMAGE\_WIDTH;

while (IMAGE\_WIDTH < 70 || IMAGE\_WIDTH > 800 || IMAGE\_WIDTH < BLOCK\_WIDTH\*number\_count/3)

{

cerr << custome\_width(IMAGE\_WIDTH, BLOCK\_WIDTH, number\_count);

cin >> IMAGE\_WIDTH;

}

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) {

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", "green");

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\_width() {

assert(custome\_width(50, 10, 10) == "Width < 70. Enter the width again:");

assert(custome\_width(850, 10, 10) == "Width > 800. Enter the width again:");

assert(custome\_width(500, 10, 10) == "");

assert(custome\_width(75, 24, 10) == "Width < BLOCK\_WIDTH\*number\_count/3. Enter the width again:");

}

int main() {

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

test\_width();

return 0;

}