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 № 1**

**По курсу «Разработка ПО систем управления»**

**«Основы языка С++»**

Выполнил студент группы А-02-19

Шепелев Д.Д.

Проверили

Мохов А. С

Козлюк Д. А

24.04.2020

#### Постановка задачи:

#### 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.

1. **Описание логики решения варианта:**

Сначала пользователю дается право ввести ширину столбца с клавиатуры. Введенное значение проверяется оператором цикла с предисловием while: если величина zadanie более 800, менее 70 или менее трети количества чисел, умноженных на ширину блока, то программа выводит на экран сообщение с просьбой ввести значение, которое удовлетворяет выше перечисленным условиям. После этого пользователь еще раз вводит значение. Цикл продолжается до тех пор, пока пользователь не введет значение, удовлетворяющее условиям задачи. Сама величина zadanie присваивается IMAGE\_WIDTH.  
  
  
**3) Фрагмент программного кода:**  
size\_t zadanie;

cerr<<"image width: ";

cin >> zadanie;

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

{

cout << "Ooops! Try again: Width mustnt be less than 70, bigger than 800 or less than 1/3 of number count \* 10. Enter image width: ";

cin >> zadanie;

}

main.cpp

#include <iostream>

#include <vector>

#include "lab03.h"

#include "svg.h"

#include <string>

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, size\_t bin\_count)

{

double min;

double max;

find\_minmax(numbers,min,max);

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

for (double x : numbers)

{

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

if (bin\_index == bin\_count)

{

bin\_index--;

}

bins[bin\_index]++;

}

return bins;

}

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 maxbin = 0;

for (size\_t bin : bins) {

if (bin > maxbin) {

maxbin = bin;

}

}

double factor = MAX\_ASTERISK/maxbin;

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;

}

}

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 auto BLOCK\_WIDTH = 10;

double top = 0;

size\_t zadanie;

cerr<<"image width: ";

cin >> zadanie;

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

{

cout << "Ooops! Try again: Width mustnt be less than 70, bigger than 800 or less than 1/3 of number count \* 10. Enter image width: ";

cin >> zadanie;

}

const auto IMAGE\_WIDTH = zadanie;

svg\_begin(zadanie, 300);

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, "red", "#ffd700");

top += BIN\_HEIGHT;

}

svg\_end();

}

int main()

{

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

size\_t number\_count;

double min, max;

cerr << "Enter number count ";

cin >> number\_count;

const auto numbers = input\_numbers(number\_count);

size\_t bin\_count;

cerr << "Enter bucket count" << endl;

cin >> bin\_count;

find\_minmax(numbers, min, max);

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

show\_histogram\_svg (bins, number\_count);

return 0;

}

lab03.h

#ifndef LAB03\_H\_INCLUDED

#define LAB03\_H\_INCLUDED

#endif // LAB03\_H\_INCLUDED

#pragma once

#include <vector>

#include <string>

using namespace std;

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

lab03.cpp

#include "lab03.h"

#include <iostream>

#include <string>

using namespace std;

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

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x > max)

{

max = x;

}

else if (x < min)

{

min = x;

}

}

}

svg.h

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#endif // SVG\_H\_INCLUDED

#pragma once

#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\_text(double left, double baseline, string text);

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

svg.cpp

#include "lab03.h"

#include <iostream>

#include <string>

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>";

}

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<<"' />";

}

test.cpp

#include "lab03.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\_single(){

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void test\_nothing(){

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}