Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Отчёт по лабораторной работе №3**

**Вариант 8**

*Задание выполнил: Ларчев В.И.*

*Студент факультета ИВТИ*

*направления УТС*

*группы A-03-19*

*Проверил:*

*Оценка:*

Москва 2020 НИУ «МЭИ»

**Постановка задачи (ПЗ)**

**Часть 1. Необходимо выделить части программы в функции:**

* Ввод чисел:
  + принимает количество чисел, которое необходимо ввести;
  + возвращает вектор чисел.
* Поиск наибольшего и наименьшего значения:
  + принимает вектор чисел;
  + возвращает два результата — min и max.
* Расчет гистограммы:
  + принимает вектор чисел и количество корзин;
  + возвращает вектор количеств чисел в каждой корзине;
  + *вызывает* в процессе работы функцию поиска min и max.

# Часть 2. Требуется вместо текстовой гистограммы рисовать картинку, например:

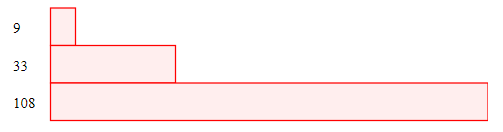


# Часть 3. Написать модульный тест для функции поиска минимума и максимума.

**Пример**

1. На вход поступают оценки 10 студентов: 4 4 3 5 3 4 5 5 4 4

Пусть требуется изобразить гистограмму на три столбца. Результат:



**Описание логики решения**

Чтобы изменять размер шрифта в подпрограмму svg\_text нужно передать число в текстовом виде, в моём решении через переменную font\_size, объявленную в функции main, и добавить атрибут font-size со значением из font\_size в элемент текста из подпрограммы svg\_text. Для отработки ввода была создана функция check\_size, которая возвращает true попадании font-size в диапазон [8 … 32], а в остальных случаях false. В тестах были проверки для пяти случаев : два для отработки граничных условий, два для отработки неподходящих значений в разных сторонах от диапазона, один для полностью подходящего значения.

**Программа на C++**

Histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <iostream>

using namespace std;

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

vector <size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count);

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

bool check\_size(size\_t font\_size);

#endif // HISTOGRAM\_H\_INCLUDED

Svg.h

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

void svg\_begin(double width, double height);

void svg\_end();

void svg\_text(double left, double baseline, string text, string font\_size = "12");

void svg\_rect(double x = 0, double y = 0, double width = 100, double height = 200, string stroke = "black", string fill = "black");

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

#endif // SVG\_H\_INCLUDED

Histogram.cpp

#include "histogram.h"

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;

}

}

}

}

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> result(bin\_count);

for (double number : numbers)

{

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

if (bin == bin\_count)

{

bin--;

}

result[bin]++;

}

return result;

}

void show\_histogram\_text(const 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';

}

}

bool check\_size(size\_t font\_size)

{

if(font\_size < 8 || font\_size > 32)

{

cerr << "Invalid input, font\_size => 8 and =< 32 !" << endl;

return false;

}

else return true;

}

svg.cpp

#include "svg.h"

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, string font\_size)

{

cout << "<text x='" << left << "' y='" << baseline << "' font-size='" << font\_size << "'>"<< 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 <<"'/>";

}

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t font\_size)

{

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;

double top = 0;

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

const size\_t SCREEN\_WIDTH = 40;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 5;

size\_t max\_count = 0;

for (size\_t bin : bins)

{

if (bin > max\_count)

{

max\_count = bin;

}

}

const bool scaling\_needed = max\_count > MAX\_ASTERISK;

for (size\_t bin : bins)

{

size\_t height = bin;

if (scaling\_needed)

{

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

double lenght\_rect = bin \* scaling\_factor;

height = (size\_t)(lenght\_rect);

}

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"red", "#ffeeee");

top += BIN\_HEIGHT;

}

svg\_end();

}

test.cpp

#include "histogram.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\_const() {

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({1}, min, max);

assert(min == 1);

assert(max == 1);

}

void test\_empty() {

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

int main()

{

test\_positive();

test\_negative();

test\_const();

test\_one();

test\_empty();

}

test\_variant.cpp

#include "histogram.h"

#include <cassert>

void test()

{

assert(check\_size(3) == false );

assert(check\_size(33) == false );

assert(check\_size(8) == true );

assert(check\_size(32) == true );

assert(check\_size(16) == true );

}

int main()

{

test();

}

main.cpp

#include "histogram.h"

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

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

int main() {

#include "histogram.h"

#include "svg.h"

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

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

int main() {

// Ввод данных

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;

size\_t font\_size;

bool wrong\_size;

while (!wrong\_size)

{

cerr << "Enter font size: ";

cin >> font\_size; cerr << endl;

wrong\_size = check\_size(font\_size);

}

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

show\_histogram\_svg(bins,font\_size); // Вывод данных

return 0;

}