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-02-19

# Ковалев А. В.

# Проверили:

# Мохов А. С.

# Козлюк Д. А.

Москва 2020

# Задание:

* 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.

#### **Вариант 9**

# Результат выполнения нужно оформить отдельным коммитом на GitHub. Доработку следует делать с использованием функции, для который нужно добавить unit-тесты из не менее двух существенно отличающихся случаев.

Запросите у пользователя ширину одного «блока» гистограммы BLOCK\_WIDTH. Не позволяйте вводить ширину блока менее 3px и более 30px. В этом случае предлагайте пользователю ввести ее заново с указанием причины.

**Описание логики решения индивидуального задания:**

Сделаем отдельный модуль для ввода ширины блока. Воспользуемся циклом while: зададим flag=1, и будем совершать цикл пока flag=0. При flag=0, должно выполняться **(BLOCK\_WIDTH>=3 && BLOCK\_WIDTH<=30).** Если же выполнится другое условие: **(BLOCK\_WIDTH<3 || BLOCK\_WIDTH>30),** то программа выведет сообщение с причиной неподходящего ввода ширины блока.

**Код программы:**

**Main.cpp:**

#include <iostream>

#include "histogram.h"

#include <vector>

#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;

}

vector<size\_t>

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

{

vector<size\_t>bins(bin\_count,0); /\* массив из индексов корзин \*/

for(double x : numbers) /\* мы каждому x присваиваем последовательно каждый элемент массива "numbers" \*/

{

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

if (bin\_index==bin\_count) /\* индекс корзины "bin\_index"\*/

{

bin\_index=bin\_index-1; /\* нумирация идет с 0\*/

}

bins[bin\_index]++;

}

return bins;}

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 column count: ";

cin >> bin\_count;

double min, max;

find\_minmax(numbers, min, max);

const auto bins=make\_histogram(numbers, max, min, 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 <iostream>

#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, string text);

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

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

#endif // SVG\_H\_INCLUDED

**Svg.cpp:**

#include "svg.h"

#include <vector>

#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, 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<<"'/>";

}

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

{

const size\_t MAX\_ASTERISK=30;

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 auto WIDTH\_TO\_PROCENT = 370;

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

double top = 0;

unsigned max\_count=0;

for (size\_t b : bins) /\* мы присваиваем значение количества элементов \*/

{

if(max\_count<b)

max\_count=b;

}

for (size\_t bin : bins)

{

size\_t height=30;

if(max\_count>MAX\_ASTERISK) /\* Если количество будет больше 35, то уменьшаем масштаб\*/

height=MAX\_ASTERISK\*((static\_cast<double>(bin))/max\_count); /\* это для того,чтобы было дробное число , получаем, используя "static\_cast<double>"\*/

else

{

height=bin;

}

height = BLOCK\_WIDTH \* height;

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

svg\_rect(TEXT\_WIDTH, top, height, BIN\_HEIGHT, "black", "yellow");

top += BIN\_HEIGHT;

}

svg\_end();

}

**block\_width.h:**

#ifndef BLOCK\_WIDTH\_H\_INCLUDED

#define BLOCK\_WIDTH\_H\_INCLUDED

#include <vector>

using namespace std;

#include <iostream>

void block\_width(double &BLOCK\_WIDTH,int flag);

#endif // BLOCK\_WIDTH\_H\_INCLUDED

**block\_width.cpp:**

#include "block\_width.h"

#include <iostream>

using namespace std;

void block\_width(double &BLOCK\_WIDTH, int flag)

{

while (flag==1)

{

cerr<<"Please enter the width of one block of the histogram\n";

cin>>BLOCK\_WIDTH;

if (BLOCK\_WIDTH<3 || BLOCK\_WIDTH>30)

cerr << "The width of one block of the histogram cannot be more then 30 px or less then 3 px\n";

if (BLOCK\_WIDTH>=3 && BLOCK\_WIDTH<=30)

flag=0;

}

}

**Test.cpp:**

#include "histogram.h"

#include <cassert>

#include "histogram.h"

#include "block\_width.h"

#include <cassert>

void

test\_1homework\_3()

{

double BLOCK\_WIDTH=2;

int flag=1;

block\_width(BLOCK\_WIDTH, flag);

assert(flag == 1);

}

void

test\_2homework\_30()

{

double BLOCK\_WIDTH=40;

int flag=1;

block\_width(BLOCK\_WIDTH, flag);

assert(flag == 1);

}

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({1}, min, max);

assert(min == 1);

assert(max == 1);

}

void

test\_nul()

{

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

int

main()

{

test\_1homework\_3();

test\_2homework\_30();

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_nul();

}