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-02-20

Андриянова Кристина Борисовна

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

Ссылка не рпозитарий: <https://github.com/AndrianovaKB/lab03/tree/master>

**Цель работы:**

1. Уметь структурировать программу при помощи функций.
2. Уметь писать модульные тесты.

# Задание:

# Часть 1. Декомпозиция программы функциями

Программа для построения гистограммы из ЛР № 1 состоит из одной функции main() на более чем 100 строк, из-за чего в ней неудобно ориентироваться. 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. Вывод гистограммы как изображения в формате SVG

Требуется вместо текстовой гистограммы рисовать картинку, например:

Изображение выглядит как стрела

Автоматически созданное описание

# Часть 3. Модульное тестирование

Написать модульный тест для функции поиска минимума и максимума.

**Индивидуальное задание. 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.

**Исходный код:**

**main.cpp**

#include <iostream>

#include <vector>

#include <string>

#include <math.h>

#include "histogram.h"

#include "svg.h"

const size\_t SCREEN\_WIDTH = 80;

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

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;

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

find\_minmax(numbers, min, max);

for (double number : numbers)

{

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

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

void show\_histogram\_text(vector <size\_t> bins, const size\_t MAX\_ASTERISK)

{

size\_t bin\_count = bins.size();

size\_t Max = bins[0];

for (size\_t j=1; j<bin\_count; j++)

{

if (bins[j]>Max)

Max = bins[j];

}

for (size\_t j=0; j<bin\_count; j++)

{

if(bins[j]>=100)

cout<<bins[j];

if (bins[j]<10)

cout <<" "<< bins[j];

else if (bins[j]<100)

cout<< " "<< bins[j] ;

cout << "|";

if(Max>MAX\_ASTERISK)

{

size\_t height = MAX\_ASTERISK \* (static\_cast<double>(bins[j]) / Max);

for (size\_t a=0; a<height; a++)

{

cout << "\*";

}

}

else

for(size\_t a=0; a<bins[j]; a++)

cout << "\*";

cout << endl;

}

}

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 bin\_count: ";

cin >> bin\_count;

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

//show\_histogram\_text(bins, MAX\_ASTERISK);

string stroke;

string fill;

show\_histogram\_svg(bins, MAX\_ASTERISK, number\_count,stroke,fill);

return 0;

}

**histogram.cpp**

#include "histogram.h"

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

{

if (numbers.size() == 0)

return;

else

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

**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

**svg.cpp**

#include "svg.h"

const size\_t SCREEN\_WIDTH = 80;

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

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\_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

svg\_end()

{

cout << "</svg>\n";

}

double input\_image\_width(size\_t number\_count, double BLOCK\_WIDTH, double& IMAGE\_WIDTH,istream& in)

{

cerr << "Enter IMAGE\_WIDTH:";

in >> IMAGE\_WIDTH;

image\_width (number\_count, BLOCK\_WIDTH, IMAGE\_WIDTH, cin);

}

double image\_width (size\_t number\_count, double BLOCK\_WIDTH, double& IMAGE\_WIDTH, istream& in)

{

while(IMAGE\_WIDTH < 70 || IMAGE\_WIDTH > 800 || IMAGE\_WIDTH < 1/3\*(number\_count\*BLOCK\_WIDTH))

{

cerr << "invalid input, please enter again";

cerr << "Enter SCREEN\_WIDTH:";

in >> IMAGE\_WIDTH;

}

}

void show\_histogram\_svg(const vector <size\_t>& bins, double bin\_count, size\_t number\_count, string& stroke, string& fill)

{

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 IMAGE\_WIDTH = image\_width(number\_count, BLOCK\_WIDTH);

double top = 0;

size\_t Max = bins[0];

for (size\_t j=1; j<bin\_count; j++)

{

if (bins[j]>Max)

Max = bins[j];

}

const bool scaling\_up = Max > MAX\_ASTERISK;

if (scaling\_up)

{

const double scaling = (double)MAX\_ASTERISK / Max;

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

for (size\_t bin : bins)

{

auto height = (size\_t)(bin \* scaling);

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,"blue", "#ffeeee");

top += BIN\_HEIGHT;

}

}

else

{

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, stroke, fill);

top += BIN\_HEIGHT;

}

}

svg\_end();

}

**svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

#include <vector>

#include <string>

using namespace std;

void

svg\_begin(double width, double height);

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

svg\_end();

double input\_image\_width(size\_t number\_count, double BLOCK\_WIDTH, istream& in);

double image\_width (size\_t number\_count, double BLOCK\_WIDTH, double& IMAGE\_WIDTH, istream& in);

void show\_histogram\_svg(const vector <size\_t>& bins, double bin\_count, size\_t number\_count, string& stroke, string& fill);

#endif // SVG\_H\_INCLUDED

**test.cpp**

#include "histogram.h"

#include "svg.h"

#include <cassert>

#include <math.h>

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\_similar()

{

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void

test\_single()

{

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);

}

void

test ()

{

double IMAGE\_WIDTH=400;

image\_width (5, 10, IMAGE\_WIDTH, cin);

}

void

test1 ()

{

double IMAGE\_WIDTH = 1000;

image\_width (5, 10, IMAGE\_WIDTH, cin);

}

void

test2 ()

{

double IMAGE\_WIDTH = 30;

image\_width (5, 10, IMAGE\_WIDTH, cin);

}

int

main()

{

double IMAGE\_WIDTH;

test\_positive();

test\_negative();

test\_single();

test\_empty();

test();

test1();

test2();

}