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-01-20

# Болотникова А. А.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2021

**Цель работы:**

* Уметь структурировать программу при помощи функций.
* Уметь писать модульные тесты.

**Порядок выполнения:**

# Часть 1. Декомпозиция программы функциями

* Требуется вынести часть кода с вводом чисел в массив в отдельную функцию, которую назовет input\_numbers.

vector<double>

input\_numbers(size\_t count) {

vector<double> result(count);

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

cin >> result[i];

}

**return** result;

}

* **С учетом того, что numbers в дальнейшем не изменяется (тип данных можно объявить как const) ввод данных в массив будет происходить так:**

constvector<double> numbers = input\_numbers(count);

В данном случае выгоднее использовать ключевое слово auto, так как оно сокращает часть написанного кода, в итоге ввод данных в массив будет выглядеть так:

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

**Объявим функцию find\_minmax()для поиска минимального и максимального элемента массива.**

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

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

min = x;

if (x > max)

max = x;

}

**Создадим функцию, производящую расчет чисел в столбцах гистограммы, назовем ее make\_histogram()**

vector<size\_t>

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

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

double min, max, bin\_size;

find\_minmax(numbers, min, max);

bin\_size = (max - min) / bin\_count;

for (double i : numbers)

{

bool found = false;

for (size\_t j = 0; (j < bin\_count - 1) && !found; j++)

{

auto lo = min + j \* bin\_size;

auto hi = min + (j + 1) \* bin\_size;

if ((lo <= i) && (i < hi))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[bin\_count - 1]++;

}

}

return bins;

}

**Также создадим функцию, выводящую гистограмму на экран, пусть она называется show\_histogram\_text()**

void

show\_histogram\_text(const vector<size\_t>& bins) {

size\_t max\_count = bins[0];

for (size\_t j : bins)

{

if (max\_count < j)

max\_count = j;

}

for (size\_t j : bins)

{

if (j < 10)

cout << " " << j << "|";

else if (j < 100 && j>10)

cout << " " << j << "|";

else if (j >= 100)

cout << j << "|";

if (max\_count == j)

{

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

cout << "\*";

}

else

{

size\_t height = MAX\_ASTERISK \* (static\_cast<double>(j) / max\_count);

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

cout << "\*";

}

cout << endl;

}

return;

}

# Часть 2. Вывод гистограммы как изображения в формате SVG

**Отработаем цикл модификации и проверки программы на примере пустого изображения, заменим функцию show\_hostogram\_text() на функцию show\_histogram\_svg()**

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";

}

**Для вывода подписей к столбцам напишем функцию вывода текста в SVG**

void svg\_text(doubleleft, doublebaseline, stringtext)

{

cout <<"<text x='"<<left<<"' y='"<<baseline<<"'>"<<text<<"</text>";

}

**Для проверки выведем высоту первого столбца гистограммы:**

void show\_histogram\_svg(constvector<size\_t>&bins)

{

svg\_begin(400, 300);

svg\_text(20, 20, to\_string(bins[0]));

svg\_end();

}

**Напишем функцию, выводящую прямоугольник SVG:**

void svg\_rect(double x, double y, double width, double height,

string stroke="black", string fill="red")

{

cout << "<rect x='" << x << "' y='" << y << "' width='" << width << "' height='" << height << "' stroke='" << stroke << "' fill='" << fill << "'/>";

}

**Заменим "магические" числа на константные переменные с понятным названием в функции show\_histogram\_svg()**

void

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

const auto IMAGE\_WIDTH = 500;

const auto IMAGE\_WIDTH\_H = 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;

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

**Реализуем вывод гистограммы при помощи SVG:**

void

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

const auto IMAGE\_WIDTH = 500;

const auto IMAGE\_WIDTH\_H = 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;

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

double top = 0;

for (size\_t bin : bins) {

constdouble bin\_width = BLOCK\_WIDTH \* bin;

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT);

top += BIN\_HEIGHT;

}

svg\_end();

}

**Добавим значения параметров по умолчанию функции svg\_rect().**

**void svg\_rect(double x, double y, double width, double height,**

**string stroke="black", string fill="red")**

**Часть 3. Модульное тестирование**

**Перенесем функцию find\_minmax() в histogram.h, а также добавим подключение библиотеки векторов и стандартного пространства имен.**

#pragmaonce

#include<vector>

usingnamespace std;

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

{

min = numbers[0];

max = numbers[0];

for (double number : numbers) {

if (number <min) {

min = number;

}

if (number >max) {

max = number;

}

}

}

**Необходимо подключить файлHeader.h (в самом начале файла main.cpp)**

#include"Header.h"

**Добавим к проекту файл реализации .cpp. Также подключим там Header.h.**

**Перенесем в histogram.cpp определение функции find\_minmax()**

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

{

min = numbers[0];

max = numbers[0];

for (double number : numbers) {

if (number <min) {

min = number;

}

if (number >max) {

max = number;

}

}

}

**Создадим новый проект типа Emptyproject.**

**Добавим к проекту histogram.cppи Header.h, а также файл релизации test.cpp.**

Файл test.cpp представляет собой полноценную тестирующую программу, в которой подключен тестируемый модуль, библиотека с assert()

#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);

}

**Убедимся, что программа завершается без ошибок, то есть тест проходит.**

**Добавим новые тесты на разные случаи:**

* массив из трех отрицательных чисел;
* массив из трех одинаковых чисел;
* массив из одного числа.

void

test\_positive1() {

double min = 0;

double max = 0;

find\_minmax({-1, -2, -3}, min, max);

assert(min == -3);

assert(max == -1);

}

void

test\_positive2() {

double min = 0;

double max = 0;

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

assert(min == 3);

assert(max == 3);

}

void

test\_positive3() {

double min = 0;

double max = 0;

find\_minmax({0}, min, max);

assert(min == 0);

assert(max == 0);

}

**Убедимся, что все тесты проходят.**

**Попробуем написать тест для пустого массива, он не пройдет.**

void

test\_positive4() {

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

**Для того, чтобы этого избежать добавим проверку на наличие элементов в массиве в функции find\_minmax()**

void

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

if(numbers.size()>0){

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

min = x;

if (x > max)

max = x;

}

}

else

;

return;

}

**Пересоберем программу и убедимся в том, что она не выдает никаких ошибок.**

**Сделаем коммит со всеми измененными файлами и отправим его на GitHub.**

**Индивидуальное задание.**

1. **Подсчитайте процент элементов, попавших в столбец, как целое двузначное число с % в конце и отображайте этот процент после столбца гистограммы с выравниванием:**

**3|\*\*\* 25%**

**5|\*\*\*\*\* 42%**

**4|\*\*\*\* 33%**

1. Воспользуемся целочисленной переменной **size\_t.** Для вывода процентов в строку нам нужно выполнить следующее действие:

**size\_t percent=(bin\*100)/number\_count**

Здесь мы находим произведение корзин и 100 (для перевода в проценты), а также частное этого произведения с количеством всех чисел.

**3.** После чего идет разветвление:

**if**(bin==max\_bin){

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

svg\_rect(TEXT\_WIDTH, top,IMAGE\_WIDTH\_H, BIN\_HEIGHT);

svg\_text(TEXT\_LEFT+BIN\_HEIGHT+IMAGE\_WIDTH\_H,top+TEXT\_BASELINE, to\_string(percent)+'%');

}

**else**{

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

**const double** IMAGE\_WIDTH\_bin=IMAGE\_WIDTH\_H\*(bin\_width/max\_bin\_width);

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

svg\_rect(TEXT\_WIDTH, top, IMAGE\_WIDTH\_bin, BIN\_HEIGHT,"black","green");

svg\_text(TEXT\_LEFT+BIN\_HEIGHT+IMAGE\_WIDTH\_bin,top+TEXT\_BASELINE, to\_string(percent)+'%');

}

1) Если в корзине находится максимальное число, то столбец заливается красным цветом.

2) Если в корзину попадает не максимальное число, то столбец заливается зеленым цветом.

В обоих случаях дописывается процент после столбца.

**4. Проверим работоспособность программы, убедимся в отсутстви ошибок. Сделаем коммит, отправим изменения на GitHub.**