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-01-19

# Уданенко А. Д

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

# Цель работы

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

# Часть 1. Декомпозиция программы функциями

Итак, имеется код для ввода массива чисел:

vector<double> numbers(number\_count);

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

cin >> numbers[i];

}

Требуется вынести его в отдельную функцию. Помещаем эту функцию над функцией main().

vector<double>

input\_numbers(size\_t count) {

vector<double> result(count);

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

cin >> result[i];

}

**return** result;

}

Теперь ввод всех чисел в массив numbers в блоке main выполняется одной строкой:

vector<double> numbers = input\_numbers(number\_count);

Но и эту строчку можно укоротить, так как этот массив после ввода изменяться не будет сделаем его постоянным, добавив const. Так же можно учесть, что дав постоянной numbers значение функции input\_numbers мы явно определяем тип numbers, это можно обозначить введя вместо типа **auto.** Новая укороченная строка выглядит так:

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

Подобным образом выносим функции find\_minmax(), которая будет выдавать минимум и максимум входящего в него массива, функцию make\_histogram(), которая будет проводить расчет количества чисел в столбцах гистограммы, и функцию show\_histogram\_text(), которая будет выводить гистограмму в текстовом формате.

Теперь файл с кодом состоит из 2ух функций обработки данных и 2х функций ввода и вывода, а основной блок main( ) выглядит так:

size\_t number\_count;

cin >> number\_count;

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

size\_t bin\_count;

cin >> bin\_count;

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

show\_histogram\_text(bins);

# Часть 2. Вывод гистограммы как изображения в формате SVG

Для отображения гистограммы в SVG формате напишем ряд функций которые выводят в консоль блоки из которых состоит SVG файл.

### **Вывод заголовка и окончания 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() {

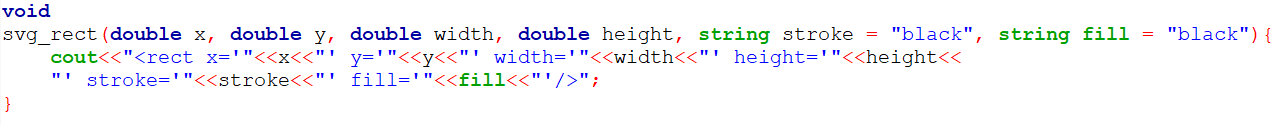
cout << "</svg>\n";

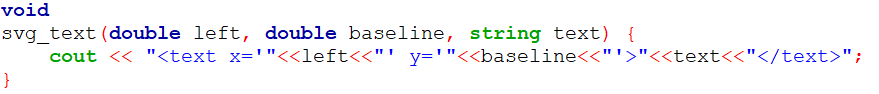
}

Эти две функции выводят в консоль коды, которые требуются в коде SVG изображения в начале и конце.

### **Функции вывода элементов SVG**

Допишем подобным образом фрагменты кода, которые отобразят прямоугольник в заданных координатах с заданными размерами и текст в заданных координатах.





Далее используя константы, описывающие стандартный размер и положение блоков изображения

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;

создадим функцию, принимающую на вход массив корзин выведет гистограмму.

void

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

size\_t max\_bin = bins[0];//поиск максимальной корзины

for (size\_t x : bins) {

if (x > max\_bin)max\_bin = x;

}

double numbers\_count = 0;//подсчет всех чисел

for (size\_t bin : bins) numbers\_count+=bin;

bool scale\_flag = false; //Условие введения изображения в формат масштабирования

if (max\_bin\*BLOCK\_WIDTH > IMAGE\_WIDTH-2\*TEXT\_WIDTH-5) scale\_flag = true;

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

double top = 0;

for (size\_t bin : bins) {

double bin\_width; //Определение размера прямоугольника

if(scale\_flag) bin\_width = (static\_cast<double>(bin)/max\_bin) \* (IMAGE\_WIDTH-2\*TEXT\_WIDTH-5);

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"black", "#e3256b");

double proc = round(((bin/numbers\_count)\*100)); //расчет и выведение процента.

string proc\_s = to\_string(static\_cast<int>(proc))+ "%";

svg\_text(TEXT\_WIDTH+bin\_width+5, top+TEXT\_BASELINE, proc\_s);

top += BIN\_HEIGHT;

}

svg\_end();

}

# Часть 3. Модульное тестирование

Напишем модульный тест для функции find\_minmax()

Для этого первым делом нужно вынести все функции связанные с созданием гистограммы в отдельный файл histogram.cpp и создать заголовочный файл histogram.h

Так же дополнительно вынесем в отдельный файл все функции создания SVG-гистограммы.

Теперь создадим новый проект в папке с текущим, он будет предназначен для написания теста модуля find\_minmax()

После создания проекта нужно добавить в обрабатываемые файлы histogram.cpp и histogram.h и можно начинать писать тесты.

Функция созданная для одного из тестов выглядит так:

void

test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

Если при выполнении команды assert вложенное в нее условие не выполняется, программа завершается.

Это тест на массив из трех положительных чисел, подобным образом напишем тест для массива из трех отрицательных, из трех одинаковых, из одного числа, для пустого массива.

Тест на пустой массив программа не пройдет, мы нашли баг.

Обновим функцию find\_minmax(), чтобы она сначала проверяла не дали ли ей пустой массив.

# Часть 4. Индивидуальное задание.

#### Вариант 5

Отображайте гистограмму зеркально по аналогии с заданием этого варианта в лабораторной работе № 1.



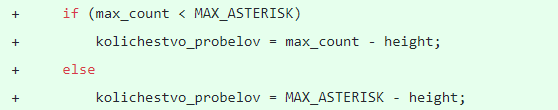
Для получения такой картинки для начала изменим длину строки



Далее добавим переменную для подсчета пробелов между наибольшем количеством символов и остальными



Считаем количество пробелов с учетом масштабирования



Вводим новую переменную для длины строки в svg, которую мы будем отображать в гистограмме и вводим их в функции, а также смещаем начало строки и текст в конец

