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

# Маркелов Н.А.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 2020

**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:



# Часть 3. Модульное тестирование

Написать модульный тест для функции поиска минимума и максимума.

**Индивидуальное задание**

**Вариант 10**

Отображайте гистограмму вертикально с подписями сверху (Аналогично заданию из ЛР №1)

3 5 2

\* \* \*

\* \* \*

\* \*

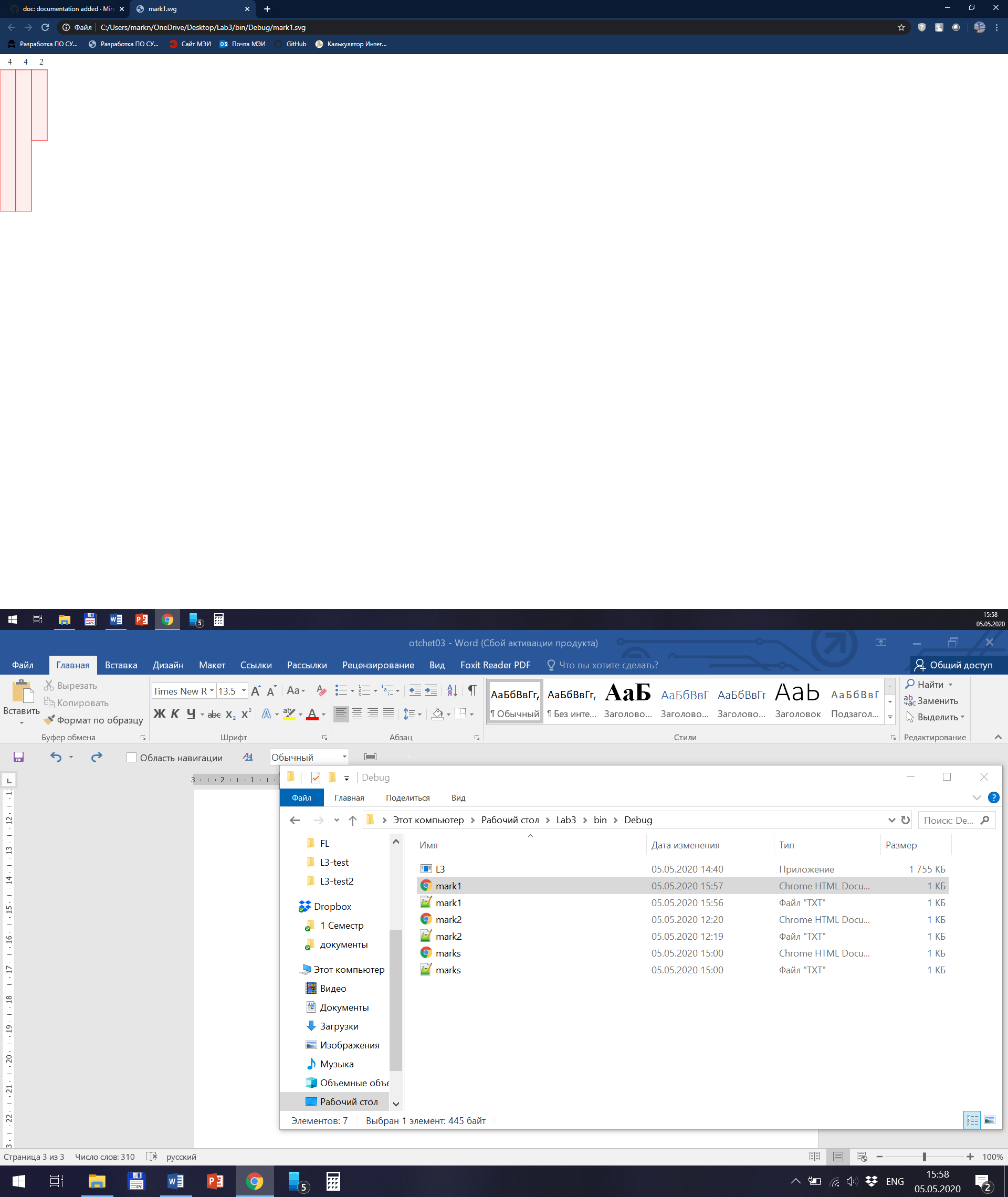
\*

Предусмотреть расчет IMAGE\_HEIGHT таким образом, чтобы вся гистограмма вмещалась в область рисунка.

**Логика решения варианта:**

Для того, чтобы гистограмма отображалась вертикально, были изменены аргументы, подаваемые в функции svg\_text и svg\_rect. Теперь, при каждом вызове этих функций координата текста и прямоугольников меняется по оси X, вместо оси Y, как это было ранее. В соответствии с такими изменениями, были изменены названия некоторых констант, отвечающих за положение и размеры всей гистограммы.

Что касается масштабирования, теперь оно проводится всегда. Максимальная корзина теперь занимает всю длину картинки, а остальные корзины, в соответствии с отношением к максимальной, подстраиваются под неё.

Пример того, как выглядит сейчас гистограмма

**Тесты для индивидуального задания:**

Индивидуальное задание включает в себя работу только с svg-модулем, поэтому тесты были написаны для него. Оба теста проверяют правильную работу масштабирования, а именно: 1) правильно ли смасштабируются 2 гистограммы, в которых одинаковое количество элементов в корзинах 2) правильное ли соотношение высот блоков в корзинах с разным количеством элементов. Для реализации была немного изменена функция показа гистограммы, в параметры теперь входит высота блока. Чтобы получать значения из функции в тестовый модуль, используется ссылка.

**Исходный код всех модулей:**

**=========================**

**main.cpp**

**=========================**

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include "histogram.h" |
|  | #include "svg.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | int main() |
|  | { |
|  | 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; |
|  |  |
|  | const auto bins=make\_histogram(numbers, bin\_count); |
|  |  |
|  | double bin\_height; |
|  | show\_histogram\_svg(bins, bin\_height); |
|  |  |
|  | return 0; |
|  | } |

**=========================**

**histogram.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; |
|  | } |
|  |  |
|  | void find\_minmax(vector<double> numbers, double& min, double& max) |
|  | { |
|  | min = numbers[0]; |
|  | max = numbers[0]; |
|  | for (auto number : numbers) |
|  | { |
|  | if (number < min) |
|  | { |
|  | min = number; |
|  | } |
|  | if (number > max) |
|  | { |
|  | max = number; |
|  | } |
|  | } |
|  | return; |
|  | } |
|  |  |
|  | vector<size\_t> make\_histogram(vector<double>numbers, size\_t bin\_count) |
|  | { |
|  | double min, max; |
|  | find\_minmax(numbers, min, max); |
|  |  |
|  | vector<size\_t>bins(bin\_count); |
|  | 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 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'; |
|  | } |
|  | return; |
|  | } |

**=========================**

**histogram.h**

**=========================**

|  |
| --- |
| #pragma once |
|  |  |
|  | #include <iostream> |
|  | #include <vector> |
|  | using namespace std; |
|  |  |
|  | vector<double> input\_numbers(size\_t count); |
|  | void find\_minmax(vector<double> numbers, double& min, double& max); |
|  | vector<size\_t> make\_histogram(vector<double>numbers, size\_t bin\_count); |
|  | void show\_histogram\_text(vector<size\_t> bins); |

**=========================**

**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) |
|  | { |
|  | cout << "<text x='" << left << "' y='" << baseline << "' >"<< text <<"</text>"; |
|  | } |
|  |  |
|  | void svg\_rect(double x, double y, double width, double height, string stroke, string fil) |
|  | { |
|  | cout << "<rect x='" << x <<"' y='" << y << "' width='" << width << "' height='" << height |
|  | << "' stroke='" << stroke << "' fill='" << fil << "' />"; |
|  | } |
|  |  |
|  | void show\_histogram\_svg(const vector<size\_t>& bins, double& bin\_height) |
|  | { |
|  |  |
|  | const auto IMAGE\_WIDTH = 400; |
|  | const auto IMAGE\_HEIGHT = 300; |
|  | const auto TEXT\_TOP = 20; |
|  | const auto TEXT\_HEIGHT = 30; |
|  | const auto BIN\_WIDTH = 30; |
|  | const auto TEXT\_BASELINE = 15; |
|  |  |
|  | svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT); |
|  |  |
|  | double left =0; |
|  |  |
|  | size\_t max\_count=bins[0]; |
|  |  |
|  | for (size\_t bin : bins) |
|  | { |
|  | if (bin>max\_count) |
|  | { |
|  | max\_count=bin; |
|  | } |
|  | } |
|  |  |
|  | for (size\_t bin: bins) |
|  | { |
|  | const double bin\_koeff = (double)(IMAGE\_HEIGHT - TEXT\_HEIGHT) / max\_count; |
|  | bin\_height = bin \* bin\_koeff; |
|  |  |
|  | svg\_text(left + TEXT\_BASELINE, TEXT\_TOP, to\_string(bin)); |
|  | svg\_rect(left, TEXT\_HEIGHT, BIN\_WIDTH, bin\_height, "red", "#ffeeee"); |
|  | left += BIN\_WIDTH; |
|  | } |
|  |  |
|  | svg\_end(); |
|  | } |

**=========================**

**svg.h**

**=========================**

|  |
| --- |
| #pragma once |
|  |  |
|  | #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); |
|  | void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fil = "black"); |
|  | void show\_histogram\_svg(const vector<size\_t>& bins, double& bin\_height); |

**=========================**

**test.cpp**

**=========================**

|  |
| --- |
| #include "C:\Users\markn\OneDrive\Desktop\Lab3\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; |
|  | double max; |
|  | find\_minmax({-1, -5,-228}, min, max); |
|  | assert(min == -228); |
|  | assert(max == -1); |
|  | } |
|  |  |
|  | void test\_one\_number() |
|  | { |
|  | double min; |
|  | double max; |
|  | find\_minmax({666}, min, max); |
|  | assert(min == 666); |
|  | assert(max == 666); |
|  | } |
|  |  |
|  | void test\_same\_numbers() |
|  | { |
|  | double min; |
|  | double max; |
|  | find\_minmax({3,3,3}, min, max); |
|  | assert(min == 3); |
|  | assert(max == 3); |
|  | } |
|  |  |
|  | void test\_empty\_vector() |
|  | { |
|  | double min; |
|  | double max; |
|  | find\_minmax({}, min, max); |
|  | assert(min == 0); |
|  | assert(max == 0); |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | test\_positive(); |
|  | test\_negative(); |
|  | test\_one\_number(); |
|  | test\_same\_numbers(); |
|  | test\_empty\_vector(); |
|  | } |

**=========================**

**test2.cpp**

**=========================**

|  |
| --- |
| #include "C:\Users\markn\OneDrive\Desktop\Lab3\svg.h" |
|  |  |
|  | #include <cassert> |
|  |  |
|  | void scalling\_same() |
|  | { |
|  |  |
|  | double height\_1; |
|  | show\_histogram\_svg({1,1,1},height\_1); |
|  | double height\_2; |
|  | show\_histogram\_svg({322,322,322}, height\_2); |
|  |  |
|  | assert(height\_1==height\_2); |
|  |  |
|  | } |
|  |  |
|  | void scalling\_correct() |
|  | { |
|  |  |
|  | double height\_1; |
|  | show\_histogram\_svg({1, 2, 3}, height\_1); |
|  |  |
|  | double height\_2; |
|  | show\_histogram\_svg({3, 2, 1}, height\_2); |
|  |  |
|  | assert(height\_1>height\_2); |
|  |  |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | scalling\_same(); |
|  | scalling\_correct(); |
|  | } |

Ссылка на репозитарий: <https://github.com/MarkelovNA/Lab3>