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

# Козлов И. А

# Проверили

# Мохов А. С.

# Козлюк Д. А

2020

1.

Смотрим содержание marks.svg

C:\Users\Игорь\Desktop\lab03\bin\Debug>type marks.svg

<?xml version='1.0' encoding='UTF-8'?>

<svg width='400' height='300' viewBox='0 0 400 300' xmlns='http://www.w3.org/2000/svg'>

</svg>

2.

svg\_text(double left, double baseline, string text) {

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

}

void

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

svg\_begin(400, 300);

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

svg\_end();

}

После запуска в консоли за записи результатов в marks.svg с помощью команды lab01.exe <marks.txt 2>NUL >marks.svg открываем файл и в браузере отображается цифра 2 на заданных нами координатах.

3.

Пишем функцию для прямоугольника

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

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

**}**

Проверяем её добавляя строчку (\*\*)

**void**

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

**svg\_begin(400, 300);**

**int i=1;**

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

**svg\_rect(50, 0, bins[0]\*10, 30);** (\*\*)

**svg\_end();**

**}**

Получаем следующий результат и убеждаемся, что всё работает.



4.

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;

Заменяем «магические константы» и проверяем работу программы. Всё работает так же.

5.

void

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

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;

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

double top=0;

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

{

const double bin\_width = BLOCK\_WIDTH \* bins[i];

svg\_text(TEXT\_LEFT, top+TEXT\_BASELINE, to\_string(bins[i]));

cout << endl;

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

cout << endl;

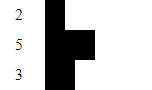
top+=BIN\_HEIGHT;

}

svg\_end();

}

Вписав этот цикл получаем уже готовую гистограмму



6.

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black")

Вводим дополнительные параметры для изменения цветов гистограммы

Редактируем функцию

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black") {

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

}

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

Можно оставить так, но чтобы поменять цвет нужно дополнить вызов

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "green", "#0000CD");

Теперь гистограмма стала цветной

