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;

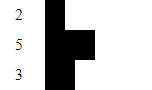
top+=BIN\_HEIGHT;

}

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 << "' />";

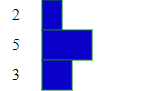
}

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



7.

Разделяем программу на модули, создаём файл histogram.h, переносим в него функцию find\_minmax и подключаем к main.cpp.

Убеждаемся, что программа работает.

8.

Создаём source файл histogram.cpp переносим в него функцию find\_minmax, оставляя в заголовочном только её объявление. Убеждаемся, что всё работает.

9.

Создаём test.cpp, пишем код, проверяем и убеждаемся, что она завершается без ошибок.

Меняем

assert(min == 1);

assert(max == 3);

на

assert(min == 3);

assert(max == 4);

Чтобы увидеть, что будет при провале проверки.

Выдаёт следующий результат.

Assertion failed: min == 3, file C:\Users\╚уюЁ№\Desktop\lab03\Lab03\test.cpp, line 10

То есть программа завершается при нахождении первой же ошибки.

10.

Весь код

#include <iostream>

#include <vector>

#include <string>

#include "histogram.h"

using namespace std;

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

min = numbers[0];

max = numbers[0];

for (int i=0; i < count; i++)

{

if (numbers[i] < min)

{

min = numbers[i];

}

if (numbers[i] > max)

{

max = numbers[i];

}

}

}

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 number\_count, size\_t bin\_count) {

double min, max;

find\_minmax(numbers, number\_count, min, max);

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

for (int i = 0; i < number\_count; i++)

{

size\_t bin = (size\_t)((numbers[i] - min) / (max - min) \* bin\_count);

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

return bins;

}

void

show\_histogram\_text(vector<size\_t> bins, const vector<double> numbers, size\_t number\_count, size\_t bin\_count) {

const size\_t SCREEN\_WIDTH = 80;

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

size\_t max\_count = 0;

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

{

if (bins[i] > max\_count)

{

max\_count = bins[i];

}

}

const bool scaling\_needed = max\_count > MAX\_ASTERISK;

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

{

if (bins[i] < 100)

{

cout << ' ';

}

if (bins[i] < 10)

{

cout << ' ';

}

cout << bins[i] << "|";

size\_t height = bins[i];

if (scaling\_needed)

{

const double scaling\_factor = (double)MAX\_ASTERISK / max\_count;

height = (size\_t)(bins[i] \* scaling\_factor);

}

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

{

cout << '\*';

}

cout << '\n';

}

}

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

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

}

double sred\_visota(vector<size\_t> bins, size\_t bin\_count) // изменения к Д/З вариваент 7. Добавлена функция sred\_visota

{

double sred\_visota;

for(int i=0; i < bin\_count; i++){

sred\_visota=sred\_visota+bins[i];

}

return sred\_visota/bin\_count;

}

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;

if (bins[i] > sred\_visota(bins, bin\_count)) // изменения к Д/З вариваент 7. добавлено if

{

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

cout << endl;

}

else

{

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

cout << endl;

}

top+=BIN\_HEIGHT;

}

svg\_end();

}

int main()

{

// Ввод данных

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

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

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

// Вывод данных

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

// show\_histogram\_text(bins, numbers, number\_count, bin\_count);

show\_histogram\_svg(bins, bin\_count);

return 0;

}