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 программного обеспечения систем управления»**

# «Декомпозиция программы»

Выполнил студент группы А-03-20

Меденцев Алексей Николаевич

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

**Цель работы:**

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

**Код файла main.cpp:**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 3 - 1;

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

{

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

double min, max;

find\_minmax(numbers, min, max);

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

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

{

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 <= numbers[i]) && (numbers[i] < hi))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[bin\_count - 1]++;

}

}

return bins;

}

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

{

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

for (size\_t x : bins)

if (x > max\_count)

max\_count = x;

if (max\_count > 76)

{

for (size\_t x : bins)

{

if (x < 100)

cout << " ";

if (x < 10)

cout << " ";

cout << x << "|";

size\_t height = 76 \* (static\_cast<double> (x) / max\_count);

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

cout << "\*";

cout << endl;

}

}

else

{

for (size\_t x : bins)

{

if (x < 100)

cout << " ";

if (x < 10)

cout << " ";

cout << x << "|";

for (size\_t j = 0; j < x; j++)

cout << "\*";

cout << endl;

}

}

return bins;

}

vector<double> input\_numbers(size\_t count)

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

int main()

{

size\_t i;

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

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

size\_t bin\_count;

cerr << "bin\_count: ";

cin >> bin\_count;

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

show\_histogram\_svg (bins);

return 0;

}

**Код файла histogram.cpp:**

#include <iostream>

#include "histogram.h"

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;

}

else if (x > max)

{

max = x;

}

}

}

}

**Код файла histogram.h:**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

**Код файла svg.cpp:**

#include <iostream>

#include "svg.h"

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 3 - 1;

void foo(int Dash\_length, int Void\_length, double top, int BIN\_HEIGHT, int max\_bin\_count)

{

cout << "<line x1='50' y1='" << top + BIN\_HEIGHT + 5 <<"' x2='" << max\_bin\_count \* 50 << "' y2='" << top + BIN\_HEIGHT + 5

<<"' stroke='black' stroke-dasharray='" << Dash\_length << " " << Void\_length << "' />";

}

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\_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 fill) //За цвет линий в SVG отвечает атрибут stroke, а за цвет заливки — fill.

{

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

}

void svg\_end()

{

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

}

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

{

//Индивидуальное задание

int Dash\_length = 0;

int Void\_length = 0;

cerr << "Enter the length of the dash : ";

cin >> Dash\_length;

cerr << "Enter the length of the skip : ";

cin >> Void\_length;

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;

double top = 0;

//Маcштабирование

int max\_bin\_count = bins[0];

for (int i = 0; i < bins.size(); i++)

if (bins[i] > max\_bin\_count)

max\_bin\_count = bins[i];

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

{

int height = bins[i];

if (max\_bin\_count > MAX\_ASTERISK)

height = 76 \* (static\_cast<double>(bins[i])/ max\_bin\_count);

}

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

for (size\_t bin : bins)

{

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

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

foo(Dash\_length, Void\_length, top, BIN\_HEIGHT, max\_bin\_count);

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "lime", "#00ff00");

top += BIN\_HEIGHT +10;

}

svg\_end();

}

**Код файла svg.h:**

#include <iostream>

#include "svg.h"

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 3 - 1;

void foo(int Dash\_length, int Void\_length, double top, int BIN\_HEIGHT, int max\_bin\_count)

{

cout << "<line x1='50' y1='" << top + BIN\_HEIGHT + 5 <<"' x2='" << max\_bin\_count \* 50 << "' y2='" << top + BIN\_HEIGHT + 5

<<"' stroke='black' stroke-dasharray='" << Dash\_length << " " << Void\_length << "' />";

}

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\_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 fill) //За цвет линий в SVG отвечает атрибут stroke, а за цвет заливки — fill.

{

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

}

void svg\_end()

{

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

}

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

{

//Индивидуальное задание

int Dash\_length = 0;

int Void\_length = 0;

cerr << "Enter the length of the dash : ";

cin >> Dash\_length;

cerr << "Enter the length of the skip : ";

cin >> Void\_length;

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;

double top = 0;

//Маcштабирование

int max\_bin\_count = bins[0];

for (int i = 0; i < bins.size(); i++)

if (bins[i] > max\_bin\_count)

max\_bin\_count = bins[i];

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

{

int height = bins[i];

if (max\_bin\_count > MAX\_ASTERISK)

height = 76 \* (static\_cast<double>(bins[i])/ max\_bin\_count);

}

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

for (size\_t bin : bins)

{

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

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

foo(Dash\_length, Void\_length, top, BIN\_HEIGHT, max\_bin\_count);

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, "lime", "#00ff00");

top += BIN\_HEIGHT +10;

}

svg\_end();

}

**Код файла test.cpp:**

#include "histogram.h"

#include <cassert>

void test\_positive() //тест положительные числа

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

void test\_negative() //тест отрицательные числа

{

double min = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void test\_equally() //тест одикавые числа

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_alone() //массив из одного числа

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_none() //обработка пустого массива

{

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

int main()

{

test\_positive();

test\_negative();

test\_equally();

test\_alone();

test\_none();

}

**Вариант 14**

Задание:

Разделяйте каждый столбец пунктирными линиями длиной IMAGE\_WIDTH. Запрашивайте у пользователя шаблон пунктира. Шаблон пунктира задается в атрибуте stroke-dasharray блока <line> в виде stroke-dasharray = '20 10', где 20 означает длину черточки, 10 - длину промежутка. У пользователя нужно запросить как длину черточки, так и длину промежутка.

Реализация:

В функции show\_histagram я создаю переменные «Dash\_length» и «Void length».

Также передаю эти переменные в функцию «foo», которая при помощи html функций dasharray происходит печатание пунктиров.

Для этих пунктиров создаю начальные координаты, которые также передаю в функцию.

Строчка кода функции:

void foo(int Dash\_length, int Void\_length, double top, int BIN\_HEIGHT, int max\_bin\_count)

{

cout << "<line x1='50' y1='" << top + BIN\_HEIGHT + 5 <<"' x2='" << max\_bin\_count \* 50 << "' y2='" << top + BIN\_HEIGHT + 5

<<"' stroke='black' stroke-dasharray='" << Dash\_length << " " << Void\_length << "' />";

}