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

Иванов Олег Борисович

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.

#### Вариант 6

После запроса количества столбцов запросите цвет для каждого столбца.

Решение:

Добавлена новая функция на ввод из потока, где человеку предлагается ввести цвет самому от 000000 до 999999. Строка с числом объединяется с символом #, и используется в качестве цвета для заливки гистограммы.

**Код main.cpp**

#include <iostream>

#include <vector>

using namespace std;

#include "histogram.h"

vector <size\_t>

make\_histogram (const vector<double>& numbers, const size\_t& number\_count, const size\_t& bin\_count, const double min, const double max, const double bin\_size)

{

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

for (size\_t i = 0; i < number\_count; 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<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 number\_count;

size\_t bin\_count;

const size\_t SCREEN\_WIDTH = 80;

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

cerr << "Enter number count: ";

cin >> number\_count;

cerr << "Enter bin count:";

cin >> bin\_count;

vector <string> color(bin\_count);

color\_add(bin\_count, color, cin);

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

double min, max;

find\_minmax(numbers, min, max);

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

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

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

getchar ();

return 0;

}

**Код Histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

#include <string>

using namespace std;

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

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, string fill);

void show\_histogram\_svg(const vector<size\_t>& bins, const size\_t bin\_count, vector <string>& color);

void color\_add(const size\_t bin\_count,vector <string>& color, istream& in);

#endif // HISTOGRAM\_H\_INCLUDED

**Код histogram.cpp**

#include "histogram.h"

#include <iostream>

using namespace std;

void color\_add(const size\_t bin\_count,vector <string>& color, istream& in)

{

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

{

//cerr << "Enter number color- "<< i+1 <<" ( 000000 - 999999) = "; Ввод если пользователь сам его выполняет, а не ввод из текстового документа.

in >> color[i];

}

}

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

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

{

min = x;

}

if (x > max)

{

max = x;

}

}

}

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 fill)

{

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

}

void

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

{

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;

const auto MAX\_WIGTH = IMAGE\_WIDTH - TEXT\_WIDTH;

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

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

{

if (bins [i] > max\_bins)

max\_bins = bins[i] ;

}

max\_bins = max\_bins \* BLOCK\_WIDTH;

double height;

if (max\_bins <= MAX\_WIGTH)

height = 1;

else

height = (static\_cast < double > (MAX\_WIGTH) / max\_bins );

double top = 0;

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

int help\_color = 0;

for (size\_t bin : bins )

{

string COLOR = "#" + color[help\_color];

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

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

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

top += BIN\_HEIGHT;

help\_color ++;

}

svg\_end();

}

**Код test.cpp**

#include "histogram.h"

#include <vector>

#include <cassert>

#include <iostream>

#include <sstream>

using namespace std;

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 = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void test\_equally()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_alone()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_color()

{

vector <string> colors (3);

string col0 = "000000";

string col1 = "111111";

string col2 = "222222";

stringstream input("000000 111111 222222");

color\_add(3, colors, input);

assert(colors[0] == "000000");

assert(colors[1] == "111111");

assert(colors[2] == "222222");

}

int main()

{

test\_color();

test\_positive();

test\_negative();

test\_equally();

test\_alone();

}