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

# Сметанин А.С.

# Проверили

# Мохов А. С

# Козлюк Д. А

Москва 24.04.2020

#### **https://github.com/SmetaninAS/lab3dz**

#### Вариант 14

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.

Необходимо было вывести между столбцами пунктирную линию, параметры которой должен был вводить человек. Между столбцами линия помещается с помощью цикла с условием неравенства значения top значеию (bin\_count-1)\*BIN\_HEIGHT) чтобы линия не появлялась выше первого столбца и ниже последнего.

**Main**

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include "histogram.h" |
|  | #include "svg.h" |
|  | using namespace std; |
|  |  |
|  | 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 bin\_count) |
|  | { |
|  | const size\_t SCREEN\_WIDTH = 80; |
|  | const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1; |
|  | double min; |
|  | double max; |
|  | find\_minmax(numbers,min,max); |
|  | vector<size\_t> bins(bin\_count,0); |
|  | for (double number : numbers) |
|  | { |
|  | size\_t bin; |
|  | bin = (number - min) / (max - min) \* bin\_count; |
|  | if (bin == bin\_count) |
|  | { |
|  | bin--; |
|  | } |
|  | bins[bin]++; |
|  | } |
|  |  |
|  | return bins; |
|  | } |
|  | int main() |
|  | { |
|  | double length\_ch,length\_pr; |
|  | size\_t number\_count; |
|  | cerr << "Enter number count: "; |
|  | cin >> number\_count; |
|  | // Ââîä ÷èñëå çàìåíåí âûçîâîì ôóíêöèè: |
|  | const auto numbers = input\_numbers(number\_count); |
|  | size\_t bin\_count; |
|  | cerr << "Enter column count: "; |
|  | cin >> bin\_count; |
|  | // Îáðàáîòêà äàííûõ |
|  | double min, max; |
|  | // Ïîèñê ìàêñèìóìà è ìèíèìóìà ñ ïîìîáùüþ ïðîöåäóðû |
|  | find\_minmax(numbers,min,max); |
|  | const auto bins = make\_histogram(numbers, bin\_count); |
|  | // Âûâîä äàííûõ |
|  | cerr <<"Enter length\_chertochki and length\_probela "; |
|  | cin>> length\_ch >> length\_pr; |
|  |  |
|  | show\_histogram\_svg(bins,bin\_count,length\_ch,length\_pr); |
|  | return 0; |
|  | } |

**Histogram.cpp**

#include "histogram.h"

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

{

if (numbers.size()==0)

{

return;

}

else

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

}

**Histogram h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

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

#endif // HISTOGRAM\_H\_INCLUDED

**Svg.h**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#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 fill = "black");

void svg\_line(double x1,double y1,double x2,double y2,double length\_ch,double length\_pr,string stroke\_1 = "red");

void show\_histogram\_svg(const vector<size\_t>& bins,size\_t bin\_count,double length\_ch,double length\_pr);

#endif // SVG\_H\_INCLUDED

**Svg.cpp**

#include <iostream>

#include "svg.h"

using namespace std;

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 svg\_line(double x1,double y1,double x2,double y2,double length\_ch,double length\_pr,string stroke\_1)

{

cout << "<line x1='"<<x1<<"' y1='"<<y1<<"' x2='"<<x2<<"' y2='"<<y2<<"' stroke='"<<stroke\_1<<"' stroke-dasharray='"<<length\_ch<<" "<<length\_pr<<"' />";

}

void show\_histogram\_svg(const vector<size\_t>& bins,size\_t bin\_count,double length\_ch,double length\_pr)

{

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 size\_t SCREEN\_WIDTH = 80;

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

double top = 0;

svg\_begin(400, 300);

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)

{

size\_t height = bin;

if (scaling\_needed)

{

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

height = (size\_t)(bin \* scaling\_factor);

}

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"blue","#aaffaa");

if (top !=(bin\_count-1)\*BIN\_HEIGHT)

svg\_line(BLOCK\_WIDTH,top+BIN\_HEIGHT,IMAGE\_WIDTH,top+BIN\_HEIGHT,length\_ch,length\_pr,"red");

top += BIN\_HEIGHT;

}

svg\_end();

}

**Test1.cpp**

#include <iostream>

#include "test1.h"

#include <cassert>

#include "histogram.h"

#include "svg.h"

const auto BIN\_HEIGHT = 30;

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\_similar ()

{

double min = 0;

double max = 0;

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

assert(min == 2);

assert(max == 2);

}

void test\_negative()

{

double min = 0;

double max = 0;

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

assert(min == -3);

assert(max == -1);

}

void test\_one\_number()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 1);

}

void test\_empty()

{

double min = 0;

double max = 0;

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

assert(min ==0);

assert(max == 0);

}

void test\_lishnya\_liniya()

{

double top = 0;

show\_histogram\_svg({1,2,3},3,20,10);

assert (top=BIN\_HEIGHT);

}

void test\_bins0()

{

double top=0;

show\_histogram\_svg({},3,20,10);

assert (top=0);

}

**Test,cpp**

#include "histogram.h"

#include "test1.h"

#include <cassert>

int

main()

{

test\_positive();

test\_negative ();

test\_similar ();

test\_one\_number();

test\_lishnya\_liniya();

test\_bins0();

}

**Test.h**

#ifndef TEST1\_H\_INCLUDED

#define TEST1\_H\_INCLUDED

#include <iostream>

#include <vector>

void test\_positive();

void test\_similar ();

void test\_negative();

void test\_one\_number();

void test\_empty();

void test\_lishnya\_liniya();

void test\_bins0();

#endif // TEST1\_H\_INCLUDED