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»

«Основы языка С++»

# Выполнил студент группы А-02-19

# Сон А.Е.

# Проверили:

# Мохов А. С

# Козлюк Д. А

Москва 2020

#### <https://github.com/SonAE1/cs.lab03>

#### 

#### Вариант 15

Добавьте горизонтальную шкалу под гистограммой по аналогии с заданием лабораторной работы 1. Шкалу нужно разбить на интервалы, размер которых вводит пользователь. Допустимы размеры от 2 до 9 , при некорректном вводе печатайте сообщение со словом «ERROR» и завершайте работу программы. Под нулевой, первой и последней отметкой шкалы требуется напечатать соответствующие числа. Шкала должна быть во всю ширину гистограммы.

#### Описание логики решения варианта:

Заведем новую переменную scale, которая будет хранить в себе размер интервала, вводимого пользователем .

Выясним количество выводимых интервалов (переменная interval), причем учтем, что происходит деление двух целых чисел - прибавим единицу, если остаток от деления не равен нулю.

С помощью цикла for выведем шкалу (конец одного интервала заканчивается, если параметр цикла делится на переменную scale без остатка).

С помощью цикла for выведем необходимые числа (нулевой элемент шкалы, размер одного интервала, общее количество элементов шкалы).

Используем некоторые константы, чтобы расположение шкалы было верное

Решение

**Main.cpp**

#include "histogram.h"

#include <iostream>

#include "svg.h"

#include <vector>

#include <cmath>

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

vector<double>

input\_numbers(size\_t count)

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

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

{

for(double x: numbers)

{

size\_t bin\_index=(x-min)/(max-min)\*bin\_count;

if (bin\_index==bin\_count)

{

bin\_index--;

}

bins[bin\_index]++;

}

}

int main()

{

size\_t number\_count;

cerr << "Enter number count:";

cin >> number\_count;

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

size\_t bin\_count;

cerr << "Enter bin count:";

cin >> bin\_count;

size\_t scale;

cerr << "Enter number of scale:";

cin >> scale;

vector <size\_t> bins(bin\_count, 0);

double min, max;

find\_minmax(numbers, min, max);

make\_histogram ( numbers, max, min, bin\_count, bins );

show\_histogram\_svg(bins, scale);

return 0;

}

**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);

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

#endif // HISTOGRAM\_H\_INCLUDED

**HISTOGRAM.CPP**

#include "histogram.h"

#include <iostream>

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

{

if (numbers.size() != 0)

{

min = numbers[0];

max = numbers[0];

for(double x: numbers)

{

if(min>x)

min=x;

if(max<x)

max=x;

}

}

}

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

{

const size\_t SCREEN\_WIDTH = 80;

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

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

const bool need = max\_count > MAX\_ASTERISK;

for (size\_t bin : bins)

{

if (bin < 100)

{

cout << ' ';

}

if (bin < 10)

{

cout << ' ';

}

cout << bin << "|";

size\_t height = bin;

if (need)

{

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

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

}

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

{

cout << '\*';

}

cout << '\n';

}

}

**SVG.H**

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <cmath>

#include <iostream>

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, string fill );

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t scale0);

void scale (const vector<size\_t>& bins, size\_t scale0,size\_t& interval);

#endif // SVG\_H\_INCLUDED

**SVG.CPP**

#include "svg.h"

#include <iostream>

#include <vector>

#include <cmath>

#include "histogram.h"

#include <sstream>

using namespace std;

const size\_t SCREEN\_WIDTH = 80;

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

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 = "black", string fill = "black")

{

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>";

}

void scale (const vector<size\_t>& bins, size\_t scale0, size\_t& interval)

{

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

for (size\_t bin : bins)

{

if (max\_bin<bin)

max\_bin=bin;

}

if (bins.size() != 0)

{

if (max\_bin%scale0!=0)

interval =max\_bin/scale0+1;

else

interval =max\_bin/scale0;

}

}

void show\_histogram\_svg(const vector<size\_t>& bins, size\_t scale0)

{

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 MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_WIDTH - TEXT\_LEFT;

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

double top = 0;

size\_t max\_count = 0;

for (size\_t count : bins) {

if (count > max\_count) {

max\_count = count;

}

}

const bool need = max\_count\* BLOCK\_WIDTH > MAX\_ASTERISK;

const double scaling\_factor1 = (double)MAX\_ASTERISK / max\_count;

size\_t interval;

if (scale0>9 || scale0<2)

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, "ERROR");

else

{

for (size\_t bin : bins)

{

size\_t height = bin;

if (need)

{

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

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,"#000000","#A52A2A");

top += BIN\_HEIGHT;

}

scale ( bins, scale0, interval);

svg\_text (TEXT\_WIDTH-2,top + TEXT\_BASELINE,"|");

for(int i=1; i<=interval\*scale0; i++)

{

if (i%scale0==0)

svg\_text(TEXT\_WIDTH-2+i\*scaling\_factor1,top +TEXT\_BASELINE,"|");

else

svg\_text(TEXT\_WIDTH-2+i\*scaling\_factor1,top+ TEXT\_BASELINE,"-");

}

svg\_text(TEXT\_WIDTH,top+ 2\*TEXT\_BASELINE, to\_string(0));

for(int i=1; i<interval\*scale0; i++)

{

if ((i%scale0!=0) || ((i % scale0== 0) && ((i > scale0))))

svg\_text(TEXT\_WIDTH-2+i\*scaling\_factor1,top+ 2\*TEXT\_BASELINE," ");

else

svg\_text(TEXT\_WIDTH-2+i\*scaling\_factor1,top+ 2\*TEXT\_BASELINE, to\_string(i));

}

svg\_text(TEXT\_WIDTH-5+interval \* scale0\*scaling\_factor1,top+ 2\*TEXT\_BASELINE, to\_string(interval \* scale0) );

}

}

**TEST.CPP**

#include "histogram.h"

#include "svg.h"

#include <cassert>

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\_same()

{

double min = 0;

double max = 0;

find\_minmax({1, 1, 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\_one()

{

double min = 0;

double max = 0;

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

assert(min == 2);

assert(max == 2);

}

void test\_scale\_empty()

{size\_t interval=1;

scale({},5,interval);

assert(interval == 1);

}

void test\_scale1()

{

size\_t interval;

scale({3,5,9},3,interval);

assert(interval==4);

}

int

main()

{

test\_positive();

test\_negative();

test\_same();

test\_empty();

test\_scale\_empty();

test\_scale1();

}