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 и контроль корректности программы»**

Выполнил студент группы А-02-19

Морозов Г.З.

Проверили

Мохов А. С

Козлюк Д. А

**Main.cpp:**

#include "histogram.h"

#include "svg.h"

#include <iostream>

#include <vector>

using namespace std;

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

{

vector<size\_t> result(count);

double min;

double max;

find\_minmax(numbers, min, max);

for (double number : numbers)

{

size\_t bin = (size\_t)((number - min) / (max - min) \* count);

if (bin == count)

{

bin--;

}

result[bin]++;

}

return result;

}

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

for (size\_t bin : bins)

{

if (bin < 100)

{

cout << ' ';

}

if (bin < 10)

{

cout << ' ';

}

cout << bin << "|";

size\_t height = bin;

if (scaling\_needed)

{

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';

}

}

int main()

{

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

cerr << "Enter numbers: ";

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

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

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

show\_histogram\_svg(bins);

return 0;

}

**Svg.cpp:**

#include "svg.h"

#include <vector>

#include <string>

#include <iostream>

using namespace std;

size\_t find\_max(const vector<size\_t>& bins)

{

size\_t max = bins[0];

for (const auto& bin : bins)

{

if (bin > max)

{

max = bin;

}

}

return max;

}

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, size\_t text)

{

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

}

void svg\_rect(double x, double y, double width, double height, string stroke, string filll)

{

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

}

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

{

double TEXT\_WIDTH = 0;

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 500;

const auto TEXT\_LEFT = 10;

const auto TEXT\_BASELINE = 120;

const auto CONST\_AUTO = 20;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 10;

const size\_t MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_LEFT - TEXT\_WIDTH;

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

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

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

double HEIGHT\_MAXIMUM = find\_max(bins);

double top = 0;

for (size\_t bin : bins)

{

size\_t height = bin;

if (scaling\_needed)

{

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(top + TEXT\_LEFT,TEXT\_BASELINE, bin);

svg\_rect(TEXT\_WIDTH, CONST\_AUTO\*(HEIGHT\_MAXIMUM - height), BIN\_HEIGHT, height\*CONST\_AUTO,"red","#aab5ff");

TEXT\_WIDTH += BIN\_HEIGHT;

top += BIN\_HEIGHT;

}

svg\_end();

}

Svg.h:

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <vector>

#include <string>

using namespace std;

void svg\_begin(double width, double height);

void svg\_end();

void svg\_text(double left, double baseline, size\_t text);

void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black");

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

#endif // SVG\_H\_INCLUDED

Histogram.cpp:

#include "histogram.h"

#include <string>

#include <vector>

#include <iostream>

using namespace std;

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

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

Histogram.cpp:

#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);

#endif // HISTOGRAM\_H\_INCLUDED

Test.histogram.cpp:

#include "histogram.h"

#include <string>

#include <vector>

#include <iostream>

using namespace std;

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

{

min = numbers[0];

max = numbers[0];

for (double number : numbers)

{

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

Test.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);

#endif // HISTOGRAM\_H\_INCLUDED

Test.cpp:

#include "histogram.h"

#include <cassert>

#include "svg.h"

void

test\_positive() {

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

void

test\_max()

{

double max = 0;

find\_max({2,5,3})

assert (max == 5);

}

int

main() {

test\_positive();

test\_max();

}

Домашнее задание:

Формулировка:   
Отображайте гистограмму вертикально с подписями снизу.

\*

\*

\* \*

\* \*

\* \* \*

\_ \_ \_

3 5 1

**Ход мыслей:**

1. Чтобы перeвернуть Гистограмму мне потребовалось поменять местами значения x и y.
2. Поменял ширины и высоты столбцов с учётом того, что ось Y направлена вниз
3. Так же из-за особенностей построения гистограммы мне требуется находить высоту максимального столбца. Что я и делаю в отдельной функции.
4. После данных действий я столкнулся с проблемой расположения цифр, её я решил подбором переменных, а так же выделением новой переменной, отвечающей за передвижение тех самых чисел. ( в Лаб.3 у нас была одна переменная, которая отвечает и за гистограмму и за числа, у меня их стало две, одна за числа, вторая за гистограмму )

В итоге гистограмма стала строиться по такому принципу:

Пользователь указывает точку отсчёта прямоугольника, его высоту и ширину. Точка отсчёта будет каждый раз сдвигаться на ширину столбца, которую мы знаем, а высоту мы можем узнать из bins[i].

Ссылка на репозитарий: <https://github.com/MorozovGZ/lab03>