**Разработка программного обеспечения систем управления**

# **Лабораторная работа №3**

# **“Декомпозиция программ”**

**Вариант 2**

**Подготовил:**

**студент группы А-03-19**

**Зимовин Михаил Юрьевич**

**27.04.2020**

**Реализация для гистограммы в SVG масштабирования**

Реализую масштабирование аналогично масштабированию звездочек из ЛР №1. Сначала нахожу максимальное кол-во чисел в корзине. В цикле прорисовки каждого столбца проверяю нужно ли масштабирование. Если оно нужно, нахожу коэффициент и умножаю на bin, отвечающий за размер прямоугольника (значение bin вывожу до масштабирования).

…

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;

double scaling\_factor = 1;

for (size\_t bin : bins)

{

fillcolor=color\_choice(COLOR,max\_count,bin);

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

if (scaling\_needed)

{

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

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

}

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

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

top += BIN\_HEIGHT;

…

**Реализация тестов**

Для реализации мы создаем test.cpp и подключаем к нему histogram.h и histogram.cpp, с помощью которых мы сможем использовать функцию find\_minmax для проверки. Также мы подключаем cassert это модуль для проверки ошибок, в частности мы используем функцию assert, которая сравнивает ожидаемые значения с реальными. Всего мы имеем 5 тестов: массив положительных чисел, отрицательных, массив из одинаковых чисел, из одного числа, и пустой массив.

#include "histogram.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\_one() {

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);

}

int

main() {

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

}

**Реализация автоматической заливки каждого столбца**

**(2 вариант)**

Для определения яркости каждого столбца создаю отдельную функцию color\_choice(). Из этой функции возвращается значение в формате "#XXX". Далее передаю эту переменную в параметр fill. Исходя из тестов решил, что для случая, если корзина пустая буду присваивать этой переменной значение #999.

string

color\_choice (size\_t color, size\_t max\_count, size\_t bin)

{

string shade;

if(bin==0)

shade="#999";

else

shade='#'+ to\_string(color\*(size\_t)(10-(bin \* 9.0)/max\_count));

return shade;

}

void

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

{

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 COLOR = 111;

const size\_t SCREEN\_WIDTH = 80;

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

string fillcolor;

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

double scaling\_factor = 1;

for (size\_t bin : bins)

{

fillcolor=color\_choice(COLOR,max\_count,bin);

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

if (scaling\_needed)

{

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

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

}

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

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

top += BIN\_HEIGHT;

}

svg\_end();

}

**Тесты для моего варианта**

#include "svg.h"

#include <cassert>

test\_empty\_bin(){

string shade;

size\_t color = 111;

size\_t max\_count = 100;

shade=color\_choice(color,max\_count,0);

assert(shade == "#999");

}

test\_max\_count(){

string shade;

size\_t color = 111;

size\_t max\_count = 100;

shade=color\_choice(color,max\_count,100);

assert(shade == "#111");

}

int main(){

test\_empty\_bin();

test\_max\_count();

}

**Код всех модулей**

**histogram.cpp**

#include "histogram.h"

void

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

if (numbers.size() != 0)

{

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

**main.cpp**

#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(vector <double> numbers,size\_t bin\_count,double min,double max){

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

for (double number : numbers)

{

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

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

return(bins);

}

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;

double min, max;

find\_minmax(numbers, min, max);

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

show\_histogram\_svg(bins);

return 0;

}

**svg.cpp**

#include "svg.h"

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, string fill)

{

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

}

void

svg\_text(double left, double baseline, string text,size\_t bin)

{

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

}

string

color\_choice (size\_t color, size\_t max\_count, size\_t bin)

{

string shade;

if(bin==0)

shade="#999";

else

shade='#'+ to\_string(color\*(size\_t)(10-(bin \* 9.0)/max\_count));

return shade;

}

void

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

{

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 COLOR = 111;

const size\_t SCREEN\_WIDTH = 80;

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

string fillcolor;

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

double scaling\_factor = 1;

for (size\_t bin : bins)

{

fillcolor=color\_choice(COLOR,max\_count,bin);

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

if (scaling\_needed)

{

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

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

}

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

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

top += BIN\_HEIGHT;

}

svg\_end();

}

**svg.h**

#ifndef SVGHEADER\_H\_INCLUDED

#define SVGHEADER\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

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

void svg\_end();

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

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

string color\_choice (size\_t color, size\_t max\_count, size\_t bin);

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

#endif // SVGHEADER\_H\_INCLUDED

**test.cpp**

#include "histogram.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\_one() {

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);

}

int

main() {

test\_positive();

test\_negative();

test\_same();

test\_one();

test\_empty();

}

**test\_svg.cpp**

#include "svg.h"

#include <cassert>

test\_empty\_bin(){

string shade;

size\_t color = 111;

size\_t max\_count = 100;

shade=color\_choice(color,max\_count,0);

assert(shade == "#999");

}

test\_max\_count(){

string shade;

size\_t color = 111;

size\_t max\_count = 100;

shade=color\_choice(color,max\_count,100);

assert(shade == "#111");

}

int main(){

test\_empty\_bin();

test\_max\_count();

}

**Ссылка на репозитарий**

https://github.com/ZimovinMY/cs-lab3.git