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-01-19

Шашерина А.В.

Проверили

Мохов А. С

Козлюк Д. А

Москва 2020

Цель работы

1.Уметь структурировать программу при помощи функций.
2.Уметь писать модульные тесты.

Вариант 2

Задавать автоматически яркость заливки каждого столбца гистограммы в градациях серого в зависимости от высоты столбца. Чем больше столбец, тем темнее заливка.

Сделать это можно, передавая цвет в параметр fill в формате "#RGB" (red, green, blue). "#111" — самый темный, "#222" — чуть менее темный, …, "#EEE" — практически белый, "#FFF" — белый. В лабораторной работе использовать диапазон цветов от "#111" для самого большого столбца до "#999" для самого маленького столбца. Поскольку используются градации серого, расчет сводится к вычислению только одного значения и дублированию этого значения в качестве цвета каждого из каналов (Red, Green, Blue). Для расчета цвета *i*-го столбца bins[i] использовать формулу (10 - (bins[i] * 9) / max_count). По ней мы получаем значение цвета одного канала (от 1 до 9), который затем записываем три раза.

Логика решения: Для решения задачи я создала две функции,find_minnmaxx и color_bins. Первая находит максимальное и минимальное числа в векторе bins (т.е max и min длины столбцов). Вторая определяет конкретный цвет столбца, для max и min “111” и 999”(далее они принимеп нормальный вид “#xxx”) соответственно, цветf остальных столбцов вычисляются по формуле.

Код:

main.cpp

```
#include <iostream>
#include <vector>
#include "histogram.h"
#include "svg.h"


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 bin_count)
{
    double min, max;
    find_minmax(numbers, min, 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;



    const auto bins = make_histogram(numbers, bin_count);

    show_histogram_svg(bins);

    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) ;

#endif // HISTOGRAM_H_INCLUDED
```

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;
            }
        }
    }
}
```

svg.h

```
#ifndef SVG_H_INCLUDED
#define SVG_H_INCLUDED

#include <iostream>
#include <vector>
#include <string>
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 find_minnmaxx(const vector<size_t>&bins, size_t& minn, size_t& maxx);
string color_bins(const vector<size_t>&bins, size_t max_count, size_t bin);
void show_histogram_svg(const vector<size_t>& bins);

#endif // SVG_H_INCLUDED
```

svg.cpp

```
#include "svg.h"
#include<string>
#include<sstream>



void svg_begin(double width, double height)
{
    cout << "<?xml version='1.0' encoding='UTF-8'?>\n";
    cout << "<svg "
         << "width='" << width << "' "
         << "height='" << height << "' "
```

```
        << "viewBox='0 0 " << width << " " << height << "' "
        << "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 find_minnmaxx(const vector<size_t>&bins, size_t& minn, size_t& maxx)
{

    for (double number : bins)
    {
        if (number < minn)
        {
            minn = number;
        }
        if (number > maxx)
        {
            maxx = number;
        }
    }
}
string color_bins(const vector<size_t>&bins, size_t max_count, size_t bin)
{
    size_t minn = bins[0];
    size_t maxx = bins[0];
    find_minnmaxx(bins, minn,maxx);

    ostringstream digit;
    size_t x;


    if (bin == minn)
    {
        x = 9;
    }
    else if (bin == maxx)
    {
        x = 1;
```

```
    }
    else
    {
        x =10 - (bin * 9) / max_count;
    }
    digit << x;
    string color = digit.str();
    color = color + color + color;
    return color;
}
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;
    double top = 0;

    svg_begin(IMAGE_WIDTH, IMAGE_HEIGHT);
    const size_t MAX_VIDTH = IMAGE_WIDTH- TEXT_WIDTH;
    const size_t MAX_ASTERISK = 35;
    size_t max_count = 0;
    for (size_t count : bins)
    {
        if (count > max_count)
        {
            max_count = count;
        }
    }

    for (size_t bin: bins)
    {
        string color = color_bins(bins, max_count, bin);
        cout << color << endl;

        double bin_factor ;
        const bool scalling_needed = max_count > MAX_ASTERISK;

        if (scalling_needed)
        {
            const double koeff = (double)MAX_ASTERISK / max_count;
            bin_factor= (size_t)(bin * koeff);
        }
        else
        {
            bin_factor=bin;
        }
```

```
        const double bin_width = BLOCK_WIDTH * bin_factor;



        svg_text(TEXT_LEFT, top + TEXT_BASELINE, to_string(bin));
        svg_rect(TEXT_WIDTH, top, bin_width, BIN_HEIGHT, "mistyrose", "#"+color);
        top += BIN_HEIGHT;
    }

    svg_end();
}
```

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_onenum()
{
    double min = 0;
    double max = 0;
    find_minmax({110.5}, min, max);
    assert(min == 110.5);
    assert(max == 110.5);
}
```

```
void test_emptymassif()
{
    double min = 0;
    double max = 0;
    find_minmax({}, min, max);
    assert(min == 0);
    assert(max == 0);
}
void test_color()
{
    vector <size_t> bins={2,4,6};
    assert(color_bins(bins, 6, bins[0])=="111");
    assert(color_bins(bins, 6, bins[1])=="444");
    assert(color_bins(bins, 6, bins[2])=="888");
}
int
main()
{
    test_positive();
    test_negative();
    test_same();
    test_onenum();
    test_emptymassif();
    test_color();

}
```