Лабораторная работа №3

Отчёт

**Разработка программного обеспечения систем управления**

Сазонов Павел

А-01-19

**Задание.**

# 1) Написать программу для построения гистограммы массива чисел как изображения в формате SVG

2) Доработать программу в соответствии с вариантом.

#### Вариант 16

После запроса количества столбцов запросить цвет линий для каждого столбца. Проверять ввод: цвет должен либо начинаться с #

**Пояснение к варианту**

Проверка выполняется сравнением первого элемента с символом ‘#’, при совпадении происходит замена стандартного значения ‘red’ для обводки и ‘ffeeee’ для заполнения на полученное значение.

**Текст программы:**

**Main.cpp**

**Svg.cpp**

**Svg.h**

**Colors.cpp**

**Colors.h**

**Доп. Тест. Модули:**

**Test.cpp**

**Histogram.cpp**

**Histogram.h**

**Test\_svg.cpp**

**Main.cpp**

#include <iostream>

#include <math.h>

#include <cstring>

#include "histogram.h"

#include "colors.h"

#include "svg.h"

//Ввод чисел

vector<double> input\_numbers

(size\_t count)

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}

//Нахождение количества чисел в каждой корзине

void make\_histogram

(vector<double> numbers,double min,double max,

vector<size\_t>& bins,size\_t number\_count,size\_t bin\_count)

{

size\_t bin\_index=0;

for(double x:numbers)

{

if(x==max)

{

(bins[bin\_count-1])++;

continue;

}

//Расчёт номера корзины для каждого числа с учетом min/max

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

(bins[bin\_index])++;

}

if(min==max)

{

bins.resize(1);

bins[0]=number\_count;

}

}

//Вывод гистограммы

void show\_histogram\_svg

(const vector<size\_t>& bins,size\_t bin\_count,

const vector<string>& stroke,const vector<string>& fill )

{

//Константы

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 = 50;

//Гистограмма

// Инициализация svg

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

double top = 0;

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

{

// Ввод констант и нужных переменных

const size\_t SCREEN\_WIDTH = 400;

const size\_t MAX\_in\_BIN = SCREEN\_WIDTH-TEXT\_WIDTH;

size\_t max\_count = 0;

//Нахождение максимального значения по корзинам

if (bins[i] > max\_count)

{

max\_count = bins[i];

}

//Проверка необходимости масштабирования

const bool scaling\_needed = max\_count > (size\_t)floor(MAX\_in\_BIN/BLOCK\_WIDTH);

//Масштабирование

size\_t height = bins[i];

if (scaling\_needed)

{

const double scaling\_factor = (double)MAX\_in\_BIN/BLOCK\_WIDTH / max\_count;

height = (size\_t)(bins[i] \* scaling\_factor);

}

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

//Гистограмма

// Вывод текста

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bins[i]));

//Гистограмма

// Вывод собственно гистограмм

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke[i],fill[i]);

top += BIN\_HEIGHT;

}

//Гистограмма

// Конец вывода

svg\_end();

}

int main()

{

//Ввод исходных данных

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

size\_t bin\_count;

cerr << "Enter bin count: ";

cin >> bin\_count;

//Ввод исходных данных

// Ввод чисел

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

//Ввод исходных данных

// Ввод цветов

vector<string> stroke(bin\_count,"red");

vector<string> fill(bin\_count,"#ffeeee");

input\_colors(stroke,fill,bin\_count);

//Обработка данных

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

//Обработка данных

// Нахождение минимального, максимального значений

double min;

double max;

find\_minmax(numbers,min,max);

//Обработка данных

// Нахождение количества чисел в каждой корзине

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

//Обработка данных

// Вывод гистограммы

show\_histogram\_svg(bins,bin\_count,stroke,fill);

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\_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\_end() {

cout << "</svg>\n";

}

**Svg.h**

#pragma once

#ifndef SVG\_H\_INCLUDED

#define SVG\_H\_INCLUDED

#include <iostream>

using namespace std;

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

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 svg\_end();

#endif // SVG\_H\_INCLUDED

**Colors.cpp**

#include "colors.h"

void input\_colors(vector<string>& stroke,vector<string>& fill,size\_t bin\_count)

{

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

{

string current\_stroke;

cerr<<"enter stroke color:";

cin>>current\_stroke;

if(current\_stroke[0]=='#')

{

stroke[i]=current\_stroke;

}

string current\_fill;

cerr<<"enter stroke color:";

cin>>current\_fill;

if(current\_fill[0]=='#')

{

fill[i]=current\_fill;

}

}

}

**Colors.h**

#pragma once

#ifndef COLORS\_H\_INCLUDED

#define COLORS\_H\_INCLUDED

#include <cstring>

#include <vector>

#include <iostream>

using namespace std;

void input\_colors(vector<string>& stroke,vector<string>& fill,size\_t bin\_count);

#endif // COLORS\_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\_alone()

{

double min = 0;

double max = 0;

find\_minmax({4}, min, max);

assert(min == 4);

assert(max == 4);

}

void test\_im\_empty()

{

double min = 0;

double max = 0;

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

assert(min == 1);

assert(max == 3);

}

int main()

{

test\_positive();

test\_negative();

test\_alone();

test\_im\_empty();

}

**Histogram.cpp**

#include "histogram.h"

void find\_minmax

(vector<double> numbers, double& min, double& max)

{

if (numbers.empty())

{

cout<<"empty vector";

}

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

{

min = x;

}

else if (x > max)

{

max = x;

}

}

}

**Histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <iostream>

#include <vector>

using namespace std;

void find\_minmax

(vector<double> numbers, double& min, double& max);

#endif // HISTOGRAM\_H\_INCLUDED

Test\_svg.cpp

#include "histogram.h"

#include <cassert>

//here we can test something if we want to

**Ссылка на репозитарий.**

**https://github.com/JustAnother-User/FirstLabDivided\_ThirdLab.git**