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-03-20

Тюрин Егор

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Индивидуальное задание:

После запроса количества столбцов запросить цвет линий для каждого столбца. Проверять ввод: цвет должен либо начинаться с #, либо не иметь внутри пробелов.

**Main**

#include <iostream>

#include <vector>

#include "histogram.h"

#include "svg.h"

#include <sstream>

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;

}

int main()

{

size\_t number\_count, bin\_count;

cerr<<"Enter number count:";

cin>>number\_count;

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

cerr<<"Enter bin count:";

cin>>bin\_count;

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

show\_histogram\_svg(bins);

return 0;

}

**Histogram.cpp**

#include "histogram.h"

#include <iostream>

using namespace std;

void

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

min = numbers[0];max=numbers[0];

for (double x:numbers){

if(x<min) {

min=x;

}

else if (x>max){

max=x;

}

}

}

vector<size\_t>

make\_histogram(vector<double> numbers,size\_t bin\_count){

double min,max;

find\_minmax(numbers,min,max);

double bin\_size= (max-min)/bin\_count;

double max\_count=0;

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

size\_t number\_count=numbers.size();

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

bool found = false;

for (size\_t j=0;(j<bin\_count) && !found;j++){

auto low=min+j\*bin\_size;

auto hi=min+(1+j)\*bin\_size;

if((numbers[i]>=low)&&(numbers[i]<hi)){

bins[j]++;

found=true;

}

if (bins[j]>max\_count) max\_count=bins[j];

}

if (!found) {

bins[bin\_count - 1]++;

}

}

return bins;

}

void

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

size\_t bin\_count=bins.size();

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

if (bins[i]<100)

cout<<" ";

if (bins[i]<10)

cout<<" ";

cout<<bins[i]<<"|";

for (size\_t j=0;j<bins[i];j++)

cout<<"\*";

cout<<endl;

}

}

**Histogram.h**

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <vector>

using namespace std;

void

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

vector <size\_t>

make\_histogram(vector<double> numbers,size\_t bin\_count);

void

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

#endif // HISTOGRAM\_H\_INCLUDED

**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, string text) ;

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

void

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

void

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

string fillcolor(istream& in);

#endif // SVG\_H\_INCLUDED

**Svg.cpp**

#include "svg.h"

#include <iostream>

const size\_t SCREEN\_WIDTH = 80;

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

using namespace std;

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, string text) {

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

}

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

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

}

string fillcolor(istream& in){

string filler;

do{

// in.ignore();

cerr<<"Enter color:";

getline(in,filler, ';');

} while((!(filler.find(' ')==std::string::npos)||(filler[0]=='#'))&&!(filler[0]=='#'));

return filler;

}

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;

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

svg\_text(TEXT\_LEFT, TEXT\_BASELINE, to\_string(bins[0]));

double top = 0;

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

for (size\_t bin : bins)

if (bin > max\_count)

max\_count = bin;

for (size\_t bin : bins) {

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

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

top += BIN\_HEIGHT;

}

top=0;

cin.ignore();

if (max\_count > MAX\_ASTERISK ){

for(size\_t bin : bins){

string fill=fillcolor(cin);

const double bin\_width = static\_cast <double> (IMAGE\_WIDTH - TEXT\_LEFT)/max\_count \* bin;

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,fill);

top+=BIN\_HEIGHT;

}

}

else{

for(size\_t bin : bins){

string fill=fillcolor(cin);

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,fill);

top+=BIN\_HEIGHT;

}

}

svg\_end();

}

**Test.cpp**

#include "svg.h"

#include<iostream>

#include <cassert>

#include <sstream>

void check1(){

stringstream colour (" gfg; black;black");

string col=fillcolor(colour);

assert(col=="black");

}

void check2(){

stringstream colour("#ffeeee");

string col=fillcolor(colour);

assert(col=="#ffeeee");

}

int

main() {

check1();

check2();

}