Индивидуальное домашнее задание к лабораторной работе №3

«Декомпозиция программ»

По дисциплине Разработка программного обеспечения систем управления

Кочурова Виктория Артёмовна, А-03-19

Задание: После запроса количества столбцов запросите цвет для каждого столбца.

Поскольку в реализации задания использовался тип string, вначале была подключена библиотека <string>.

Переменная colors имеет тип vector<string>, поскольку надо вывести вектор строк. Она зависит от переменной bin\_count (кол-во корзин).

Кол-во корзин равно кол-ву столбцов, которые надо вывести. Поэтому в цикле for со счётчиком перебираем все столбцы с нулевого до тех пор, пока индекс значение индекса i меньше, чем bin\_count, и для каждого из них осуществляем запрос цвета: пользователь вводит цвет столбцов.

Реализуем заполнение гистограммы заданным цветом. В функции в аргументы функции show\_histogram\_svg() в круглых скобочках запишем переменную colors.

Main.cpp:

#include "histogram.h"

#include "svg\_module.h"

#include <iostream>

#include <vector>

#include <string>

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(const 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(const 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;

if (number\_count == 0) {

cerr << "ERROR: Empty vector";

} else {

cerr << "Enter numbers: ";

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

size\_t bin\_count;

cerr << "Enter column count: ";

cin >> bin\_count;

vector<string> colors(bin\_count);

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

cerr << "Enter column's number" << i << " color: ";

cin >> colors[i];

}

double min, max;

find\_minmax(numbers, min, max);

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

show\_histogram\_svg(bins, colors);

}

return 0;

}

Svg\_module.cpp:

#include <vector>

#include <string>

#include <iostream>

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\_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)

{

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

}

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

{

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 size\_t SCREEN\_WIDTH = 80;

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

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

double top = 0;

string stroke = "black";

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;

unsigned int color\_count = 0;

for (size\_t bin : bins)

{

if (scaling\_needed)

{

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

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

}

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

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

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT, stroke, colors[color\_count]);

top += BIN\_HEIGHT;

color\_count++;

}

svg\_end();

}

Svg\_module.h:

#pragma once

#include <vector>

#include <string>

using namespace std;

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