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 № 1**

**По курсу «Разработка программного обеспечения систем управления»**

# «Декомпозиция программы»

Выполнил студент группы А-03-20

Львов М. Д.

Проверил

Мохов А. С

Москва 2021

Цели работы:

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

Задачи работы:

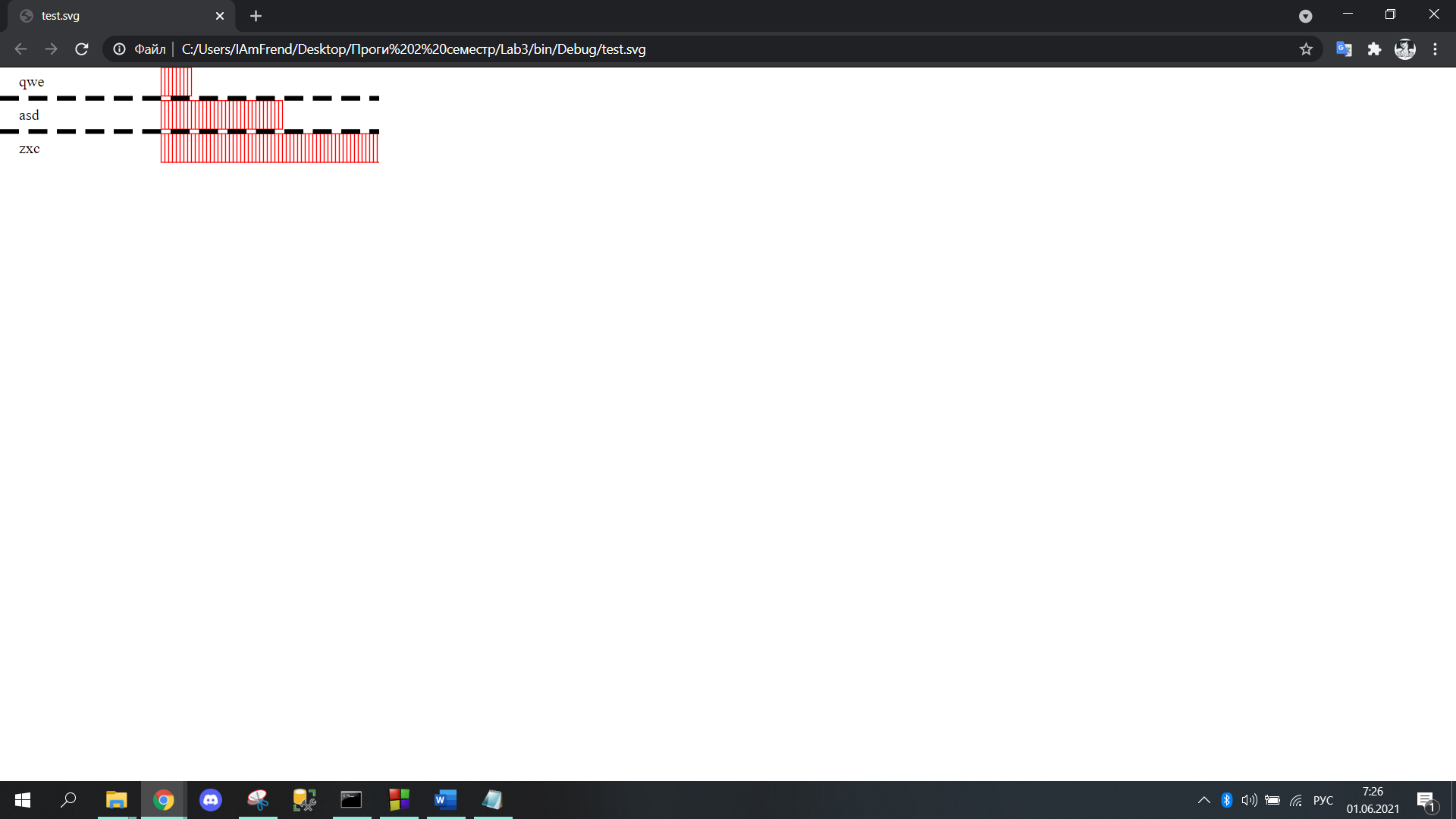
1. Произвести декомпозицию программы из Лр1
2. Создать возможность создания svg-файлов в программе.
3. Создать модульные тесты для программы
4. Доработать программу в соответствии с вариантом:

Вариант 14: Разделяйте каждый столбец пунктирными линиями длиной IMAGE\_WIDTH. Запрашивайте у пользователя шаблон пунктира. Шаблон пунктира задается в атрибуте stroke-dasharray блока <line> в виде stroke-dasharray = '20 10', где 20 означает длину черточки, 10 - длину промежутка. У пользователя нужно запросить как длину черточки, так и длину промежутка.

1. Создать два противоположных по значению модульных теста.

Описание решения варианта:

В начале решения работы указан метод создания строки, а в варианте – задания пунктира. Для соответствия стандарту принято решение создать ещё одну подпрограмму для создания линий. Для обеспечения тестируемости она создана в виде функции, возвращающей строковое значение. Данная функция впоследствии применяется в процедуре создания диаграммы со значениями, зависящими от предопределённых. Также в последней процедуре учитывается, что разделители не должны идти перед первым и после последнего столбца. Модульные тесты проверяют корректность формируемой строки – позитивный тест сравнивает фрагмент из задания с требуемым, негативный – проверяет, что последовательность значений штриха и пропуска соответствует требуемым.



Код программы:

* Модуль main.cpp:

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include "histogram.h"

using namespace std;

vector<double> input\_numbers(size\_t count)

{

vector<double> result(count);

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

cerr << "Enter number " << i + 1 << ": ";

cin >> result[i];

}

return result;

}

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

{

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

double min = numbers[0];

double max = numbers[0];

find\_minmax(numbers, min, max);

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

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

{

bool found = false;

for (size\_t j = 0; (j < bin\_count - 1) && !found; j++)

{

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

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

if ((lo <= numbers[i]) && (numbers[i] < hi))

{

bins[j]++;

found = true;

}

}

if (!found)

{

bins[bin\_count - 1]++;

}

}

return bins;

}

void show\_histogramm\_text(vector<size\_t> bins, size\_t bin\_count, vector<vector<char>> bin\_naming\_list)

{

vector<char[80]> bin\_namings(bin\_count);

size\_t iter = 0;

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

{

for (char l : bin\_naming\_list[iter])

{

bin\_namings[iter][strlen(bin\_namings[iter])] = l;

}

}

double max\_len = strlen(bin\_namings[0]);

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

{

if (strlen(bin\_namings[i]) > max\_len)

{

max\_len = strlen(bin\_namings[i]);

}

}

double max\_count = bins[0];

const size\_t SCREEN\_WIDTH = 80;

size\_t MAX\_ASTERISK = SCREEN\_WIDTH - max\_len - 1;

for (double x : bins)

{

if (x > max\_count)

{

max\_count = x;

}

}

double height = 1;

if (max\_count > MAX\_ASTERISK)

{

height = static\_cast<double>(MAX\_ASTERISK) / static\_cast<double>(max\_count);

}

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

{

size\_t tab = max\_len - strlen(bin\_namings[i]);

for (size\_t j = 0; j < tab; j++)

{

cout << " ";

}

cout << bin\_namings[i] << "|";

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

{

cout << "\*";

}

cout << endl;

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

size\_t number\_count;

cerr << "Enter number count: ";

cin >> number\_count;

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

size\_t bin\_count;

cerr << "Enter bin count: ";

cin >> bin\_count;

cin.get();

vector<char>naming(80);

vector<vector<char>> bin\_namings(bin\_count,naming);

vector<char[80]> bin\_naming\_list(bin\_count);

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

{

cerr << "Enter bin " << i + 1 << " naming: ";

cin.getline(bin\_naming\_list[i], 80);

}

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

{

for (size\_t i = 0; i <= strlen(bin\_naming\_list[iter]); i++)

{

char\* name = bin\_naming\_list[iter];

bin\_namings[iter].push\_back(name[i]);

}

}

cerr << "Enter separator dash: ";

double dash;

cin >> dash;

cerr << "Enter separator dasharray: ";

double dasharray;

cin >> dasharray;

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

show\_histogram\_svg(bins, bin\_count, bin\_namings,dash,dasharray);

return 0;

}

* Модуль histogram.cpp

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include "histogram.h"

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;

}

}

}

* Модуль histogram.h

#ifndef HISTOGRAM\_H\_INCLUDED

#define HISTOGRAM\_H\_INCLUDED

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

using namespace std;

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

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");

string svg\_line(double x1, float y1, double x2, float y2, string stroke, float stroke\_width, double stroke\_dash, double stroke\_dasharray);

void show\_histogram\_svg(vector<size\_t> bins, size\_t bin\_count, vector<vector<char>> bin\_naming\_list,double dash, double dasharray);

#endif // HISTOGRAM\_H\_INCLUDED

* Модуль svg.cpp

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#include "histogram.h"

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 stroke, string fill)

{

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

}

string svg\_line(double x1, float y1, double x2, float y2, string stroke, float stroke\_width, double stroke\_dash, double stroke\_dasharray)

{

return "<line x1='"+to\_string(x1)+"' y1='"+to\_string(y1)+"' x2='"+to\_string(x2)+"' y2='"+to\_string(y2)+

"' stroke='"+stroke+"' stroke-width='"+to\_string(stroke\_width)

+"' stroke-dasharray = '"+to\_string(stroke\_dash)+" "+to\_string(stroke\_dasharray)+"'/>";

}

void show\_histogram\_svg(vector<size\_t> bins, size\_t bin\_count, vector<vector<char>> bin\_naming\_list,double dash, double dasharray)

{

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 LINE\_HEIGHT = 5.0;

double BLOCK\_WIDTH = 10;

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

vector<char[80]> bin\_namings(bin\_count);

size\_t iter = 0;

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

{

for (char l : bin\_naming\_list[iter])

{

bin\_namings[iter][strlen(bin\_namings[iter])] = l;

}

}

double max\_len = strlen(bin\_namings[0]);

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

{

if (strlen(bin\_namings[i]) > max\_len)

{

max\_len = strlen(bin\_namings[i]);

}

}

double max\_count = bins[0];

for (double x : bins)

{

if (x > max\_count)

{

max\_count = x;

}

}

if (max\_count\*BLOCK\_WIDTH > IMAGE\_WIDTH)

{

BLOCK\_WIDTH = static\_cast<double>(IMAGE\_WIDTH) / static\_cast<double>(max\_count);

}

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

{

svg\_text(TEXT\_LEFT, i\*(BIN\_HEIGHT+LINE\_HEIGHT) + TEXT\_BASELINE, bin\_namings[i]);

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

{

svg\_rect(TEXT\_LEFT+TEXT\_WIDTH\*strlen(bin\_namings[i])+(BLOCK\_WIDTH\*(j-1)), i\*(BIN\_HEIGHT+LINE\_HEIGHT), BLOCK\_WIDTH, BIN\_HEIGHT,"red","white");

}

if (i>0)

{

cout << svg\_line(0,i\*(BIN\_HEIGHT)+(i-1)\*LINE\_HEIGHT+LINE\_HEIGHT/2,IMAGE\_WIDTH,i\*(BIN\_HEIGHT)+(i-1)\*LINE\_HEIGHT+LINE\_HEIGHT/2,"black",LINE\_HEIGHT,dash,dasharray);

}

cout << endl;

}

svg\_end();

}

* Модуль test.cpp

#include <iostream>

#include <math.h>

#include <cscobj.h>

#include <vector>

#include <Windows.h>

#include <iterator>

#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\_equal()

{

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\_null()

{

double min = 0;

double max = 0;

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

assert(min == 0);

assert(max == 0);

}

void svg\_test\_positive()

{

double dash = 10;

double dasharray = 20;

string line = "";

line = svg\_line(0.0,0.0,10.0,10.0,"black",1.0,dash,dasharray);

size\_t pos = line.find("stroke-dasharray =)"+to\_string(dash)+" "+to\_string(dasharray));

assert(pos>0);

}

void svg\_test\_negative()

{

double dash = 10;

double dasharray = 20;

string line = "";

line = svg\_line(0.0,0.0,10.0,10.0,"black",1.0,dash,dasharray);

size\_t pos = line.find("stroke-dasharray =)"+to\_string(dasharray)+" "+to\_string(dash));

assert(pos=string::npos);

}

int main()

{

test\_positive();

test\_negative();

test\_equal();

test\_one();

svg\_test\_positive();

svg\_test\_negative();

}

Ссылка на репозиторий:

https://github.com/LvovMD/cs-lab03