**Разработка программного обеспечения систем управления**

**Лабораторная работа №3  
“Декомпозиция программы”**

**Вариант 1  
(Для домашнего задания)**

**Строганов Максим Юрьевич**

# **Домашнее задание:** (постановка задачи)

Дайте пользователю возможность задавать произвольную ширину столбца гистограммы вместо 400. Считайте некорректной ширину менее 70, более 800 или менее трети количества чисел, умноженных на ширину блока (BLOCK\_WIDTH) — предлагайте пользователю ввести ширину заново с указанием причины.

**Логика решения моего варианта домашнего задания:**

Для повторного ввода необходимо было задать цикл, но цикл for не подходит из-за ограниченного количества действий, я написал цикл do … while и ввёл 3 логические переменные для проверки каждого из трёх условий, цикл будет повторяться до тех пор, пока одна из переменных будет удовлетворять ошибочному вводу.

do

{ bool flag1 = false;

bool flag2 = false;

bool flag3 = false;

cout << "Enter the image\_width";

cin >> IMAGE\_WIDTH;

if (IMAGE\_WIDTH < 70)

flag1 = true;

if (IMAGE\_WIDTH > 800)

flag2 = true;

if (IMAGE\_WIDTH < number\_count/3 \* BLOCK\_WIDTH)

flag3 = true;

} while (!flag1 or !flag2 or !flag3)

**Код всей программы:**

**main.cpp:**

#include <iostream>

#include <vector>

#include "Head.h"

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(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;

}

int

main()

{

// Ввод данных

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;

// Обработка данных

double min = numbers[0];

double max = numbers[0];

find\_minmax(numbers, min, max);

vector<size\_t> bins = make\_histogram(numbers, bin\_count, min, max);

// Вывод данных

show\_histogram\_svg(bins);

return 0;

}

**Выполняемый файл для всех функций, касающихся SVG:**

#include "Head.h"

using namespace std;

void

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

{

for (double number : numbers)

{

if (number < min)

{

min = number;

}

if (number > max)

{

max = number;

}

}

}

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\_end()

{

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

}

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

{

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

}

void

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

{

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;

do

{ bool flag1 = false;

bool flag2 = false;

bool flag3 = false;

cout << "Enter the image\_width";

cin >> IMAGE\_WIDTH;

if (IMAGE\_WIDTH < 70)

flag1 = true;

if (IMAGE\_WIDTH > 800)

flag2 = true;

if (IMAGE\_WIDTH < number\_count/3 \* BLOCK\_WIDTH)

flag3 = true;

} while (!flag1 or !flag2 or !flag3)

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

const bool scalei = (max\_count \* BLOCK\_WIDTH) > (IMAGE\_WIDTH - TEXT\_WIDTH);

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

const double controled = (IMAGE\_WIDTH - TEXT\_WIDTH) / (max\_count \* BLOCK\_WIDTH)

public double top = 0;

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

if (!scalei)

{

controled = bin\_width;

}

for (size\_t bin : bins)

{

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

svg\_rect(TEXT\_WIDTH, top, controled, BIN\_HEIGHT, "red", "darkblue");

top += BIN\_HEIGHT;

}

svg\_end();

}

**Заголовочный файл для всех функций, входящих в предыдущий пункт:**

#ifndef HEAD\_H\_INCLUDED

#define HEAD\_H\_INCLUDED

#include <vector>

using namespace std;

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

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

void svg\_text(double left, double baseline, string text);

void svg\_end();

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

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

#endif // HEAD\_H\_INCLUDED

**Ссылка на репрозитарий:**

[**https://github.com/Stroganov-A0319/lab03**](https://github.com/Stroganov-A0319/lab03)