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 ПО систем управления»**

**«Декомпозиция программы»**

Выполнил студент группы А-0!-19

Кирьянов Б.Ю.

Проверили

Мохов А. С.

Козлюк Д. А.

Москва 2020

**Вариант 6**

После запроса количества столбцов запросите цвет для каждого столбца.

**Ссылка на репозитарий:** <https://github.com/KiryanovBY/cs-lab03>

**Код программы:**

**main.cpp**

|  |
| --- |
|  |
|  | #include <iostream>  #include <vector> |
|  | #include <cmath> |
|  | #include <string> |
|  | #include "find\_minmax.h" |
|  | #include "input\_numbers.h" |
|  | #include "make\_histogram.h" |
|  | #include "show\_histogram\_text.h" |
|  | #include "svg\_begin.h" |
|  | #include "svg\_end.h" |
|  | #include "svg\_text.h" |
|  | #include "svg\_rect.h" |
|  | #include "show\_histogram\_svg.h" |
|  |  |
|  | using namespace std; |
|  | const size\_t SCREEN\_WIDTH = 80; |
|  | const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 3 - 1; |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | size\_t number\_count; |
|  | cerr << "enter number count > "; |
|  | cin >> number\_count; |
|  | if (!(number\_count > 0)) { |
|  | cerr << "ERROR - Invalid vector length!" << endl; |
|  | exit(0); |
|  | } |
|  | const auto numbers = input\_numbers(number\_count); |
|  |  |
|  | size\_t bin\_count; |
|  | cerr << "enter bin count"; |
|  | cin >> bin\_count; |
|  | const auto bins = make\_histogram(numbers, bin\_count); |
|  | show\_histogram\_svg(bins); |
|  | return 0; |
|  |  |
|  | } |

**make\_histogram.h**

#pragma once

#include <vector>

#include "find\_minmax.h"

using namespace std;

vector<size\_t> make\_histogram(const vector<double>&, size\_t);

|  |
| --- |
|  |

**make\_histogram.cpp**

#include "make\_histogram.h"

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

{

double min, max;

find\_minmax(numbers, min, max);

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

for (double x : numbers)

{

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

if (bin\_index == bin\_count)

{

bin\_index--;

}

count[bin\_index]++;

}

return count;

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**show\_histogram\_svg.h**

#pragma once

#include <vector>

#include "svg\_begin.h"

#include "svg\_end.h"

#include "svg\_text.h"

#include "svg\_rect.h"

#include <string>

using namespace std;

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

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |

**show\_histogram\_svg.cpp**

#include "show\_histogram\_svg.h"

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);

double top = 0;

for (size\_t bin : bins)

{

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, "red", "#ffeeee");

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

top += BIN\_HEIGHT;

}

svg\_end();

}

**show\_histogram\_text.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

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

**show\_histogram\_text.cpp**

#include "show\_histogram\_text.h"

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

{

for (size\_t bin : bins)

{

if (bin < 10)

cout << " ";

else if (bin < 100)

cout << " ";

cout << bin << "|";

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

{

cout << "\*";

}

cout << endl;

}

}

**svg\_begin.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_begin(double, double);

**svg\_begin.cpp**

#include "svg\_begin.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";

}

**svg\_end.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_end();

**svg\_end.cpp**

#include "svg\_end.h"

void svg\_end()

{

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

}

**svg\_rect.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_rect(double, double, double, double, string stroke = "black", string fill = "black");

**svg\_rect.cpp**

#include "svg\_rect.h"

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 << "' />\n";

}

**svg\_text.h**

#pragma once

#include <iostream>

using namespace std;

void svg\_text(double, double, string);

**svg\_text.cpp**

#include "svg\_text.h"

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

{

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

}

**find\_minmax.h**

#pragma once

#include <vector>

using namespace std;

void find\_minmax(const vector<double>&, double&, double&);

**find\_minmax.cpp**

#include "find\_minmax.h"

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

{

if (numbers.size() > 0) {

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (min > x)

min = x;

if (max < x)

max = x;

}

}

}

**input\_numbers.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

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

**input\_numbers.cpp**

#include "input\_numbers.h"

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

{

vector<double> result(count);

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

{

cin >> result[i];

}

return result;

}