**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ**

**КАФЕДРА УПРАВЛЕНИЯ И ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ**

**Лабораторная работа №3**

По дисциплине «Разработка программного обеспечения систем управления»

«Декомпозиция и контроль корректности программ»

Выполнил: студент I курса

Группы А-03-19

Симич А.Б.

Проверил:

Москва 2020

1. Постановка задачи.

Программа для построения гистограммы из ЛР № 1 состоит из одной функции main() на более чем 100 строк, из-за чего в ней неудобно ориентироваться. Необходимо выделить части программы в функции:

* Ввод чисел:
  + принимает количество чисел, которое необходимо ввести;
  + возвращает вектор чисел.
* Поиск наибольшего и наименьшего значения:
  + принимает вектор чисел;
  + возвращает два результата — min и max.
* Расчет гистограммы:
  + принимает вектор чисел и количество корзин;
  + возвращает вектор количеств чисел в каждой корзине;
  + *вызывает* в процессе работы функцию поиска min и max.

Вариант 16:

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

Сначала выделяем части программы в функции. После этого переносим все функции в отдельный модуль histogram.cpp. Затем создаём отдельный модуль svg.cpp, в который вставляем функции по визуализации гистограммы (вместо звездочек). Добавляем отдельный проект, цель которого – тестировать исходный код на выявление ошибок. Затем вносим изменения в svg.cpp для изменения цвета столбцов гистограммы. Каждый шаг коммитим на GitHub.

1. Ссылка на репозитарий (GitHub).

<https://github.com/Alexandr-Simich/lab03>

1. Код программы.

**main.cpp**

1. #include "histogram.h"
2. #include "svg.h"
3. using namespace std;
4. int main()
5. {
6. size\_t number\_count;
7. cerr << "Enter number count: ";
8. cin >> number\_count;
9. cerr << "Enter numbers: ";
10. const auto numbers = input\_numbers(number\_count);
11. size\_t bin\_count;
12. cerr << "Enter column count: ";
13. cin >> bin\_count;
14. double min;
15. double max;
16. find\_minmax(numbers,min,max);
17. auto bins = make\_histogram(numbers, bin\_count, min, max);
18. cerr << "Enter column colors: ";
19. const auto colors = input\_colors(bin\_count);
20. show\_histogram\_svg(bins, colors, bin\_count);
21. return 0;
22. }

**histogram.h**

1. #pragma once
2. #include <iostream>
3. #include <vector>
4. using namespace std;
5. void find\_minmax(const vector<double>& numbers, double& min, double& max);
6. vector<double> input\_numbers(size\_t count);
7. vector <size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count, double min, double max);
8. void show\_histogram\_text(vector <size\_t> bins);
9. vector <string> input\_colors(size\_t bin\_count);

**histogram.cpp**

1. #include "histogram.h"
2. void find\_minmax(const vector<double>& numbers, double& min, double& max)
3. {
4. if (numbers.size()!= 0) {
5. min = numbers[0];
6. max = numbers[0];
7. for (double number : numbers)
8. {
   1. if (number < min)
9. {
10. min = number;
11. }
12. if (number > max)
13. {
14. max = number;
15. }
16. }
17. }
18. }
19. vector<double> input\_numbers(size\_t count)
20. {
21. vector<double> result(count);
22. for (size\_t i = 0; i < count; i++)
23. {
24. cin >> result[i];
25. }
26. return result;
27. }
28. vector <size\_t> make\_histogram(const vector<double>& numbers, size\_t bin\_count, double min, double max)
29. {
30. vector<size\_t> bins(bin\_count);
31. for (double number : numbers)
32. {
33. size\_t bin = (size\_t)((number - min) / (max - min) \* bin\_count);
34. if (bin == bin\_count)
35. {
36. bin--;
37. }
38. bins[bin]++;
39. }
40. return (bins);
41. }
42. void show\_histogram\_text(vector <size\_t> bins)
43. {
44. const size\_t SCREEN\_WIDTH = 80;
45. const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;
46. size\_t max\_count = 0;
47. for (size\_t count : bins)
48. {
49. if (count > max\_count)
50. {
51. max\_count = count;
52. }
53. }
54. const bool scaling\_needed = max\_count > MAX\_ASTERISK;
55. for (size\_t bin : bins)
56. {
57. if (bin < 100)
58. {
59. cout << " ";
60. }
61. if (bin < 10)
62. {
63. cout << " ";
64. }
65. cout << bin << "|";
66. size\_t height = bin;
67. if (scaling\_needed)
68. {
69. const double scaling\_factor = (double)MAX\_ASTERISK / max\_count;
70. height = (size\_t)(bin \* scaling\_factor);
71. }
72. for (size\_t i = 0; i < height; i++)
73. {
74. cout << "\*";
75. }
76. cout << "\n";
77. }
78. }
79. vector <string> input\_colors(size\_t bin\_count)
80. {
81. vector<string> colors(bin\_count);
82. cerr << "Enter colour";
83. for (size\_t i=0; i<bin\_count; i++) {
84. bool flag = true;
85. while(colors[i][0] != '#' || flag)
86. {
87. cin >> colors[i];
88. if (colors[i][0] == '#'){
89. flag = false;
90. }
91. else {
92. cerr << "Error. Enter colour again";
93. }
94. }
95. cerr << "Enter next colour";
96. }
97. return colors;
98. }

**svg.h**

1. #pragma once
2. #include <iostream>
3. #include <vector>
4. using namespace std;
5. void svg\_begin(double width, double height);
6. void svg\_text(double left, double baseline, string text);
7. void svg\_rect(double x, double y, double wid, double heig, string stroke, string fill);
8. void svg\_end();
9. void show\_histogram\_svg(const vector<size\_t>& bins, const vector<string>& colors, size\_t bin\_count);

**svg.cpp**

#include "svg.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";

}

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

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

}

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

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

};

void svg\_end() {

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

}

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

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);

size\_t max\_count = 0;

for (size\_t count : bins)

{

if (count > max\_count)

{

max\_count = count;

}

}

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

const double scaling\_factor = (double)((IMAGE\_WIDTH - TEXT\_WIDTH)) / (double)((max\_count \* BLOCK\_WIDTH));

if (scaling\_needed)

{

double top = 0;

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

const double bin\_width = double(BLOCK\_WIDTH \* bins[i] \* scaling\_factor);

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bins[i]));

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

top += BIN\_HEIGHT;

}

}

else {

double top = 0;

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

const double bin\_width = BLOCK\_WIDTH \* bins[i];

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bins[i]));

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

top += BIN\_HEIGHT;

}

}

svg\_end();

}