**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

Мещерякова А.К.

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 9**
* Запросите у пользователя ширину одного «блока» гистограммы BLOCK\_WIDTH. Не позволяйте вводить ширину блока менее 3px и более 30px. В этом случае предлагайте пользователю ввести ее заново с указанием причины.

1. Описание логики решения.

Для начала выделили части программы в функции. После этого перенесли все подпрограммы в отдельный модуль histogram.cpp. Затем сделали отдельный модуль svg.cpp, который визуализирует эту гистограмму (вместо прежнего способа отображения гистограммы – звездочек). Добавили отдельный проект, цель которого – тестировать исходный код на выявление ошибок. Затем внесли изменения в svg.cpp для изменения прозрачности столбцов гистограммы. Каждый шаг коммитили на GitHub.

В решении варианта подразумевается, что BLOCK\_WIDTH будет постоянно изменяться. Вместе с этим IMAGE\_WIDTH тоже необходимо изменять. Если это не делать, то будет получаться так, что гистограмма либо не поместиться на картинке, либо не вся картинка будет заполнена гистограммой. Поэтому необходимо закомментировать наши константы и ввести новые BLOCK\_WIDTH и IMAGE\_WIDTH.

1. Ссылка на репозитарий (GitHub).

<https://github.com/MeshcheryakovaAK/lab-03>

1. Код программы.

Main.cpp

1. #include <iostream>
2. #include <vector>
3. #include "histogram.h"
4. #include "for\_svg.h"
5. using namespace std;
6. vector<double> input\_numbers(size\_t count)
7. {
8. vector<double> result(count);
9. for (size\_t i = 0; i < count; i++)
10. {
11. cin >> result[i];
12. }
13. return result;
14. }
15. int main()
16. {
17. size\_t number\_count;
18. cerr << "Enter number count: ";
19. cin >> number\_count;
20. cerr << "Enter numbers: ";
21. const auto numbers=input\_numbers(number\_count);
22. size\_t bin\_count;
23. cerr << "Enter column count: ";
24. cin >> bin\_count;
25. double min,max;
26. find\_minmax(numbers,min,max);
27. const auto bins = make\_histogram(numbers, bin\_count,min,max);
28. show\_histogram\_svg(bins);
29. return 0;
30. }

Histogram.cpp

1. #include "histogram.h"
2. #include <iostream>
3. void find\_minmax(const vector<double>& numbers, double& min, double& max)
4. {
5. if (numbers.size()==0) return;
6. min = numbers[0];
7. max = numbers[0];
8. for (double number : numbers)
9. {
10. if (number < min)
11. {
12. min = number;
13. }
14. if (number > max)
15. {
16. max = number;
17. }
18. }
19. }
20. vector <size\_t> make\_histogram(const vector<double>& numbers,size\_t bin\_count,double min,double max)
21. {
22. vector<size\_t> bins(bin\_count);
23. for (double number : numbers)
24. {
25. size\_t bin = (size\_t)((number - min) / (max - min) \* bin\_count);
26. if (bin == bin\_count)
27. {
28. bin--;
29. }
30. bins[bin]++;
31. }
32. return(bins);
33. }
34. void show\_histogram\_text(const vector<size\_t>& bins)
35. {
36. const size\_t screen\_width=80;
37. const size\_t width=screen\_width-4;
38. size\_t max\_count = 0;
39. for (size\_t count : bins)
40. {
41. if (count > max\_count)
42. {
43. max\_count = count;
44. }
45. }
46. const bool scaling\_needed = max\_count > width;
47. for (size\_t bin : bins)
48. {
49. if (bin < 100)
50. {
51. cout << ' ';
52. }
53. if (bin < 10)
54. {
55. cout << ' ';
56. }
57. cout << bin << "|";
58. size\_t height = bin;
59. if (scaling\_needed)
60. {
61. const double scaling\_multiplier = (double)width / max\_count;
62. height = (size\_t)(bin \* scaling\_multiplier);
63. }
64. for (size\_t i = 0; i < height; i++)
65. {
66. cout << '\*';
67. }
68. cout<<' ';
69. cout << endl;
70. }
71. }

For\_svg.cpp

1. #include "for\_svg.h"
2. #include <iostream>
3. int variant\_9()
4. {
5. int BLOCK\_WIDTH=0;
6. cerr<<"Enter the width of one block(int 3...30):";
7. cin>>BLOCK\_WIDTH;
8. while (BLOCK\_WIDTH<3 or BLOCK\_WIDTH>30)
9. {
10. cerr<<"Wrong! The width must be between 3 and 30! Enter the width again!";
11. cin>>BLOCK\_WIDTH;
12. }
13. return BLOCK\_WIDTH;
14. }
15. void svg\_begin(double width, double height)
16. {
17. cout << "<?xml version='1.0' encoding='UTF-8'?>\n";
18. cout << "<svg ";
19. cout << "width='" << width << "' ";
20. cout << "height='" << height << "' ";
21. cout << "viewBox='0 0 " << width << " " << height << "' ";
22. cout << "xmlns='http://www.w3.org/2000/svg'>\n";
23. }
24. void svg\_end()
25. {
26. cout << "</svg>\n";
27. }
28. void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black")
29. {
30. cout << "<rect x='"<<x<<"' y='"<<y<<"' width='"<<width<<"' height='"<<height<<"' stroke='"<<stroke<<"' fill='"<<fill<<"' />";
31. }
32. void svg\_text(double left, double baseline, string text)
33. {
34. cout << "<text x='" << left << "' y='"<<baseline<<"'>"<<text<<"</text>";
35. }
36. void show\_histogram\_svg(const vector<size\_t>& bins)
37. {
38. const size\_t screen\_width=80;
39. const size\_t width=screen\_width-4;
40. int BLOCK\_WIDTH=variant\_9(); //comment for lab3
41. const auto IMAGE\_WIDTH=400\*BLOCK\_WIDTH/10; //comment for lab3
42. //const auto IMAGE\_WIDTH = 400; //comment for variant 9
43. const auto IMAGE\_HEIGHT = 300;
44. const auto TEXT\_LEFT = 20;
45. const auto TEXT\_BASELINE = 20;
46. const auto TEXT\_WIDTH = 50;
47. const auto BIN\_HEIGHT = 30;
48. //const auto BLOCK\_WIDTH = 10; //comment for variant 9
49. const auto GRAPH\_WIDTH = IMAGE\_WIDTH-TEXT\_WIDTH;
50. svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);
51. double top = 0;
52. string stroke="black";
53. string fill="red";
54. size\_t max\_count = 0;
55. for (size\_t count : bins)
56. {
57. if (count > max\_count)
58. {
59. max\_count = count;
60. }
61. }
62. const bool scaling\_needed = max\_count > width;
63. if (scaling\_needed)
64. {
65. const double scaling\_multiplier = (double)width / max\_count;
66. if(max\_count\*BLOCK\_WIDTH/GRAPH\_WIDTH>1)
67. {
68. for (size\_t bin : bins)
69. {
70. const double new\_scaling\_multiplier=(double)GRAPH\_WIDTH/(max\_count\*BLOCK\_WIDTH\*scaling\_multiplier);
71. const size\_t new\_bin\_width = (size\_t)BLOCK\_WIDTH\*bin\*scaling\_multiplier\*new\_scaling\_multiplier;
72. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
73. svg\_rect(TEXT\_WIDTH, top, new\_bin\_width, BIN\_HEIGHT,stroke,fill);
74. top += BIN\_HEIGHT;
75. }
76. }
77. else
78. {
79. for (size\_t bin : bins)
80. {
81. const double bin\_width = BLOCK\_WIDTH\*bin\*scaling\_multiplier;
82. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
83. svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill);
84. top += BIN\_HEIGHT;
85. }
86. }
87. }
88. else
89. {
90. if(max\_count\*BLOCK\_WIDTH/GRAPH\_WIDTH>1)
91. {
92. for (size\_t bin : bins)
93. {
94. const double new\_scaling\_multiplier=(double)GRAPH\_WIDTH/(max\_count\*BLOCK\_WIDTH);
95. const size\_t new\_bin\_width = (size\_t)BLOCK\_WIDTH\*bin\*new\_scaling\_multiplier;
96. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
97. svg\_rect(TEXT\_WIDTH, top, new\_bin\_width, BIN\_HEIGHT,stroke,fill);
98. top += BIN\_HEIGHT;
99. }
100. }
101. else
102. {
103. for (size\_t bin : bins)
104. {
105. const double bin\_width = BLOCK\_WIDTH \* bin;
106. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
107. svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill);
108. top += BIN\_HEIGHT;
109. }
110. }
111. }
112. svg\_end();
113. }