**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

Синицин Д.И.

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 17:

Задавать автоматически прозрачность заливки каждого столбца гистограммы в зависимости от высоты столбца. Чем больше столбец, тем темнее заливка. Сделать это можно, передавая процент прозрачности в параметр fill-opacity в формате "0.7". 1 соответствует отсутствию прозрачности, 0 соответствует полной прозрачности (отсутствию цвета)

Для расчета прозрачности каждого i-го столбца bins[i] использовать формулу (bins[i]) / max\_count).

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.

1. Ссылка на репозитарий (GitHub).

<https://github.com/SinitsinDI/lab3>

1. Код программы.
2. main.cpp
3. #include <iostream>
4. #include <vector>
5. #include "histogram.h"
6. #include "svg.h"
7. using namespace std;
8. int main()
9. {
10. size\_t number\_count;
11. cerr << "Enter number count: ";
12. cin >> number\_count;
13. cerr << "Enter numbers: ";
14. const auto numbers=input\_numbers(number\_count);
15. size\_t bin\_count;
16. cerr << "Enter column count: ";
17. cin >> bin\_count;
18. double min,max;
19. find\_minmax(numbers,min,max);
20. const auto bins = make\_histogram(numbers, bin\_count,min,max);
21. show\_histogram\_svg(bins);
22. return 0;
23. }
24. histogram.h
25. #ifndef HISTOGRAM\_H\_INCLUDED
26. #define HISTOGRAM\_H\_INCLUDED
27. #include <vector>
28. using namespace std;
29. void find\_minmax(const vector<double> numbers, double& min, double& max);
30. double bin\_opacity17(double bin, size\_t max\_count);
31. vector<double> input\_numbers(size\_t count);
32. vector <size\_t> make\_histogram(vector <double> numbers,size\_t bin\_count,double min,double max);
33. void show\_histogram\_text(vector<size\_t>bins);
34. #endif // HISTOGRAM\_H\_INCLUDED
35. histogram.cpp
36. #include "histogram.h"
37. #include "svg.h"
38. #include <iostream>
39. void find\_minmax(vector<double> numbers, double& min, double& max)
40. {
41. min = numbers[0];
42. max = numbers[0];
43. for (double number : numbers)
44. {
45. if (number < min)
46. {
47. min = number;
48. }
49. if (number > max)
50. {
51. max = number;
52. }
53. }
54. }
55. double bin\_opacity17(double bin, size\_t max\_count)
56. {
57. return bin/max\_count;
58. }
59. vector<double> input\_numbers(size\_t count)
60. {
61. vector<double> result(count);
62. for (size\_t i = 0; i < count; i++)
63. {
64. cin >> result[i];
65. }
66. return result;
67. }
68. vector <size\_t> make\_histogram(vector <double> numbers,size\_t bin\_count,double min,double max)
69. {
70. vector<size\_t> bins(bin\_count);
71. for (double number : numbers)
72. {
73. size\_t bin = (size\_t)((number - min) / (max - min) \* bin\_count);
74. if (bin == bin\_count)
75. {
76. bin--;
77. }
78. bins[bin]++;
79. }
80. return(bins);
81. }
82. void show\_histogram\_text(vector<size\_t>bins)
83. {
84. const size\_t screen\_width=80;
85. const size\_t width=screen\_width-4;
86. size\_t max\_count = 0;
87. for (size\_t count : bins)
88. {
89. if (count > max\_count)
90. {
91. max\_count = count;
92. }
93. }
94. const bool scaling\_needed = max\_count > width;
95. for (size\_t bin : bins)
96. {
97. if (bin < 100)
98. {
99. cout << ' ';
100. }
101. if (bin < 10)
102. {
103. cout << ' ';
104. }
105. cout << bin << "|";
106. size\_t height = bin;
107. if (scaling\_needed)
108. {
109. const double scaling\_multiplier = (double)width / max\_count;
110. height = (size\_t)(bin \* scaling\_multiplier);
111. }
112. for (size\_t i = 0; i < height; i++)
113. {
114. cout << '\*';
115. }
116. cout<<' ';
117. cout << endl;
118. }
119. }
120. svg.h
121. #ifndef SVG\_H\_INCLUDED
122. #define SVG\_H\_INCLUDED
123. #include <vector>
124. #include <string>
125. using namespace std;
126. void svg\_begin(double width, double height);
127. void svg\_end();
128. void svg\_rect(double x, double y, double width, double height, string stroke, string fill);
129. void svg\_text(double left, double baseline, string text);
130. void show\_histogram\_svg(const vector<size\_t>& bins);
131. #endif // SVG\_H\_INCLUDED
132. svg.cpp
133. #include "histogram.h"
134. #include "svg.h"
135. #include <iostream>
136. void svg\_begin(double width, double height)
137. {
138. cout << "<?xml version='1.0' encoding='UTF-8'?>\n";
139. cout << "<svg ";
140. cout << "width='" << width << "' ";
141. cout << "height='" << height << "' ";
142. cout << "viewBox='0 0 " << width << " " << height << "' ";
143. cout << "xmlns='http://www.w3.org/2000/svg'>\n";
144. }
145. void svg\_end()
146. {
147. cout << "</svg>\n";
148. }
149. void svg\_rect(double x, double y, double width, double height, string stroke = "black", string fill = "black", double fill\_opacity=1.0)
150. {
151. cout << "<rect x='"<<x<<"' y='"<<y<<"' width='"<<width<<"' height='"<<height<<"' stroke='"<<stroke<<"' fill='"<<fill<<"' fill-opacity='"<<fill\_opacity<<"' />";
152. }
153. void svg\_text(double left, double baseline, string text)
154. {
155. cout << "<text x='" << left << "' y='"<<baseline<<"'>"<<text<<"</text>";
156. }
157. void show\_histogram\_svg(const vector<size\_t>& bins)
158. {
159. const size\_t screen\_width=80;
160. const size\_t width=screen\_width-4;
161. const auto IMAGE\_WIDTH = 400;
162. const auto IMAGE\_HEIGHT = 300;
163. const auto TEXT\_LEFT = 20;
164. const auto TEXT\_BASELINE = 20;
165. const auto TEXT\_WIDTH = 50;
166. const auto BIN\_HEIGHT = 30;
167. const auto BLOCK\_WIDTH = 10;
168. const auto GRAPH\_WIDTH = IMAGE\_WIDTH-TEXT\_WIDTH;
169. svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);
170. double top = 0;
171. string stroke="black";
172. string fill="red";
173. size\_t max\_count = 0;
174. for (size\_t count : bins)
175. {
176. if (count > max\_count)
177. {
178. max\_count = count;
179. }
180. }
181. const bool scaling\_needed = max\_count > width;
182. if (scaling\_needed)
183. {
184. const double scaling\_multiplier = (double)width / max\_count;
185. if(max\_count\*BLOCK\_WIDTH/GRAPH\_WIDTH>1)
186. {
187. for (size\_t bin : bins)
188. {
189. const double new\_scaling\_multiplier=(double)GRAPH\_WIDTH/(max\_count\*BLOCK\_WIDTH\*scaling\_multiplier);
190. const size\_t new\_bin\_width = (size\_t)BLOCK\_WIDTH\*bin\*scaling\_multiplier\*new\_scaling\_multiplier;
191. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
192. svg\_rect(TEXT\_WIDTH, top, new\_bin\_width, BIN\_HEIGHT,stroke,fill,bin\_opacity17(bin,max\_count));
193. top += BIN\_HEIGHT;
194. }
195. }
196. else
197. {
198. for (size\_t bin : bins)
199. {
200. const double bin\_width = BLOCK\_WIDTH\*bin\*scaling\_multiplier;
201. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
202. svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill,bin\_opacity17(bin,max\_count));
203. top += BIN\_HEIGHT;
204. }
205. }
206. }
207. else
208. {
209. if(max\_count\*BLOCK\_WIDTH/GRAPH\_WIDTH>1)
210. {
211. for (size\_t bin : bins)
212. {
213. const double new\_scaling\_multiplier=(double)GRAPH\_WIDTH/(max\_count\*BLOCK\_WIDTH);
214. const size\_t new\_bin\_width = (size\_t)BLOCK\_WIDTH\*bin\*new\_scaling\_multiplier;
215. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
216. svg\_rect(TEXT\_WIDTH, top, new\_bin\_width, BIN\_HEIGHT,stroke,fill,bin\_opacity17(bin,max\_count));
217. top += BIN\_HEIGHT;
218. }
219. }
220. else
221. {
222. for (size\_t bin : bins)
223. {
224. const double bin\_width = BLOCK\_WIDTH \* bin;
225. svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
226. svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,stroke,fill,bin\_opacity17(bin,max\_count));
227. top += BIN\_HEIGHT;
228. }
229. }
230. }
231. svg\_end();
232. }
233. test.cpp
234. #include "histogram.h"
235. #include <cassert>
236. void test\_bin\_opacity()
237. {
238. double bin=5;
239. size\_t max\_count=10;
240. assert(bin\_opacity17(bin,max\_count)==0.5);
241. }
242. void test\_bin\_opacity2()
243. {
244. double bin=0;
245. size\_t max\_count=10;
246. assert(bin\_opacity17(bin,max\_count)==0.0);
247. }
248. void test\_positive()
249. {
250. double min = 0;
251. double max = 0;
252. find\_minmax({1,2,3}, min, max);
253. assert(min == 1);
254. assert(max == 3);
255. }
256. void test\_negative()
257. {
258. double min = 0;
259. double max = 0;
260. find\_minmax({-1,-2,-3}, min, max);
261. assert(min == -3);
262. assert(max == -1);
263. }
264. void test\_equal()
265. {
266. double min = 0;
267. double max = 0;
268. find\_minmax({1,1,1}, min, max);
269. assert(min == 1);
270. assert(max == 1);
271. }
272. void test\_one\_number()
273. {
274. double min = 0;
275. double max = 0;
276. find\_minmax({1}, min, max);
277. assert(min == 1);
278. assert(max == 1);
279. }
280. void test\_empty()
281. {
282. double min = 0;
283. double max = 0;
284. find\_minmax({}, min, max);
285. assert(min == 0);
286. assert(max == 0);
287. }
288. int
289. main()
290. {
291. test\_bin\_opacity();
292. test\_bin\_opacity2();
293. test\_positive();
294. test\_negative();
295. test\_equal();
296. test\_one\_number();
297. test\_empty();
298. }