Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

Отчёт по лабораторной работе

Сравнение сортировок

Выполнил:

студент ф-та ИТММ гр. 3821Б1ПМ3

Попов Д. Д.

Проверил:

Заведующий лабораторией СК и ВВ

Лебедев И. Г.

Нижний Новгород

2021 г.

Содержание

[1. Введение 3](#_Toc89812339)

[2. Постановка задачи 4](#_Toc89812340)

[3. Руководство пользователя 5](#_Toc89812341)

[4. Руководство программиста 6](#_Toc89812342)

[1) Описание структуры программы 6](#_Toc89812343)

[2) Описание структур данных 8](#_Toc89812344)

[3) Описание алгоритмов 10](#_Toc89812345)

[5. Эксперименты 14](#_Toc89812346)

[6. Заключение 16](#_Toc89812347)

[7. Литература 17](#_Toc89812348)

[8. Приложение 1 18](#_Toc89812349)

[9. Приложение 2 19](#_Toc89812350)

# Введение

Программирование — неотъемлемая часть нашей жизни. это возможность взаимодействия человека с машиной, возможность общения друг с другом на понятном языке. Всё, что мы делаем на компьютере, так или иначе связано с ними.  
Данная лабораторная работа поможет разобраться с тем, как устроены сортировки данных, какие виды существуют и какие из них эффективней.

# Постановка задачи

Реализовать три типа сортировки массивов данных (пузырьком, вставкой и быстрый), сравнить их время работы. Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение. Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

# Руководство пользователя

Данная программа состоит из двух подпрограмм.

В первой программе можно ввести количество элементов, которые будут напечатаны в текстовом документе, и их минимальное и максимальное значение. (рис. 1)

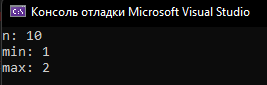


Рисунок 1. Программа 1

Во второй программе можно выбрать одно из четырех действий: вывод массива на экран, сортировка его данных по возрастанию, сброс значений до изначальных и выход из программы. (рис. 2)

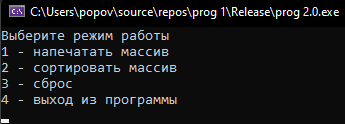


Рисунок 2. Программа 2

Также имеется возможность выбора определенного типа сортировки. Можно выбрать нажатием соответствующей кнопки сортировку пузырьком, вставками, а также быстрый тип. (рис. 3)

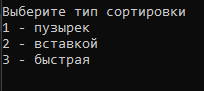


Рисунок 3. Выбор сортировки во второй программе

# Руководство программиста

## Описание структуры программы

Первая программа состоит из функции main(). В нее входят такие функции, как prinf (для вывода информации на экран), scanf (для ввода данных с клавиатуры), циклы if и for, malloc (для задания размера динамического массива), fopen (для открытия текстового файла), fprintf (для записи данных в тестовый файл), fclose (для закрытия текстового документа) и free (для освобождения памяти после работы с массивом).

Вторая программа состоит из функции scanN() для считывания количества элементов в файле, в которую входит fscanf (рис. 4), inizmas() для инициализации массива, в которую входит цикл for (рис. 5), scanmas() для считывания массива, в которую входит цикл for (рис. 6), printmas() для выода массива на экран, в которую входит цикл for (рис. 7), prisvoenie() для того, чтобы приравнять первый массив ко второму, в которую входит цикл for (рис. 8), puzyryok() для сортировки пузырьком, в которую входят циклы if и for (рис. 9), vstavka() для сортировки вставками, в которую входят циклы if и for, а также функция break (рис. 10), qSort() для быстрой сортировки, в которую входят циклы do while, while, if (рис. 11), bystraya() для вызова функции qSort() в удобном виде (рис. 12) и main(), в которую входят циклы while, if.

|  |
| --- |
| void scanN(FILE\* a, int\* n)  {  fscanf(a, "n = %d\n", n);  } |

Рисунок 4. функция scanN()

|  |
| --- |
| void inizmas(int n, float\* mas)  {  int i;  for (i = 0; i < n; i++)  {  mas[i] = 0;  }  } |

Рисунок 5. функция inizmas

|  |
| --- |
| void scanmas(FILE\* a, int n, float\* mas)  {  int i;  for (i = 0; i < n; i++)  {  fscanf(a, "%f\n", &mas[i]);  }  } |

Рисунок 6. фунция scanmas

|  |
| --- |
| void printmas(int n, float\* mas)  {  int i;  for (i = 0; i < n; i++)  {  printf("%f\n", mas[i]);  }  } |

Рисунок 7. функция printmas

|  |
| --- |
| void prisvoenie(float\* mas, float\* mas1, int n)  {  int i;  for (i = 0; i < n; i++)  {  mas1[i] = mas[i];  }  } |

Рисунок 8. функция prisvoenie

|  |
| --- |
| void puzyryok(int n, float\* mas)  {  int i, x;  float p;  for (i = 0; i < n; i++)  {  for (x = 0; x < n - i - 1; x++)  {  if (mas[x] > mas[x + 1])  {  p = mas[x];  mas[x] = mas[x + 1];  mas[x + 1] = p;  }  }  }  } |

Рисунок 9. сортировка пузырьком

|  |
| --- |
| void vstavka(int n, float\* mas)  {  int i, x;  float p;  for (i = 1; i < n; i++)  {  p = mas[i];  for (x = i - 1; x >= 0; x--)  {  if (mas[x] < p)  {  break;  }  else  {  mas[x + 1] = mas[x];  }  }  mas[x + 1] = p;  }  } |

Рисунок 10. сортировка вставками

|  |
| --- |
| void qSort(float\* mas, int l, int u)  {  int i = l, j = u;  float t = 0;  float x = mas[(int)(l + u) / 2];  do  {  while (mas[i] < x)  ++i;  while (mas[j] > x)  --j;  if (i <= j)  {  t = mas[i];  mas[i] = mas[j];  mas[j] = t;  i++;  j--;  }  } while (i < j);  if (l < j)  {  qSort(mas, l, j);  }  if (i < u)  {  qSort(mas, i, u);  }  } |

Рисунок 11. быстрая сортировка

|  |
| --- |
| void bystraya(float\* mas, int n)  {  qSort(mas, 0, n - 1);  } |

Рисунок 12. вызов быстрой сортировки

## Описание структур данных

В работе первой программы используются переменные типа int (рис. 13), double (рис. 14), float\* для массива (рис. 15) и FILE\* для текстового файла (рис. 16).

|  |
| --- |
| int n, i; |

Рисунок 13. Объявление переменных типа int

|  |
| --- |
| double max, min; |

Рисунок 14. Объявление переменных типа double

|  |
| --- |
| float\* mas |

Рисунок 15. Объявление переменных типа float\*

|  |
| --- |
| FILE\* a |

Рисунок 16. Объявление переменной типа FILE\*

Также для работы используются следующие библиотеки. (рис. 17)

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <locale.h>  #include <math.h>  #pragma warning(disable : 4996) |

Рисунок 17. Библиотеки

Во второй программе используются переменные типа int (рис. 18), FILE\* (рис. 19) и float\* (рис. 20).

|  |
| --- |
| int n, d = 0, y = 0; |

Рисунок 18. объявление переменных типа int

|  |
| --- |
| FILE\* a |

Рисунок 19. объявление переменных типа FILE\*

|  |
| --- |
| float\* mas  float\* mas1 |

Рисунок 20. объявление переменных типа float\*

Также для работы используются следующие библиотеки. (рис. 21)

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <locale.h>  #include <time.h>  #pragma warning(disable : 4996) |

Рисунок 21. Библиотеки

## Описание алгоритмов

Представим алгоритмы, которые используются в программе с помощью блок схем.

Проверка min и max на то, не больше ли min, чем max. (рис. 22)



Рисунок 22. Блок схема на проверку max>min

Проверка количества элементов на натуральное значение, т.е n>0. (рис. 23)



Рисунок 23. Блок схема на проверку n >0

Алгоритм сортировки пузырьком. (рис. 24)

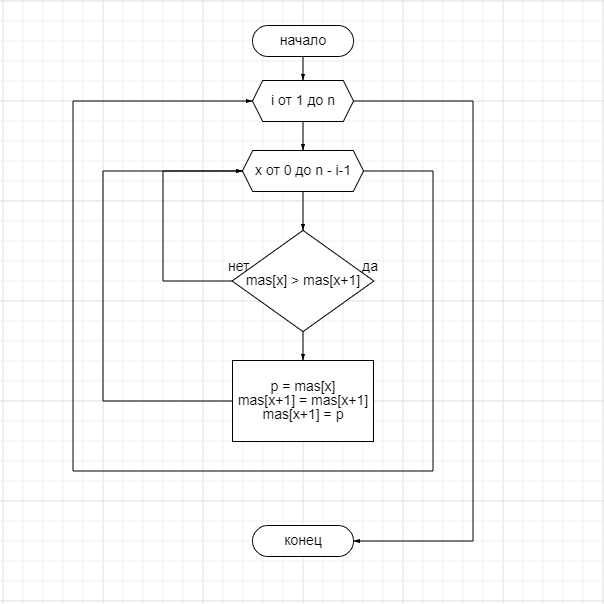


Рисунок 24. сортировка пузырьком

Алгоритм сортировки вставками. (рис. 24)

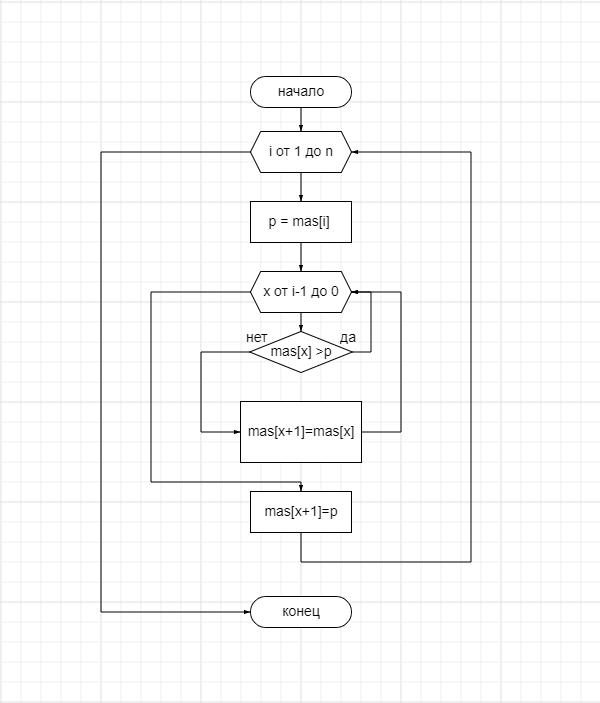


Рисунок 25. сортировка вствками

Алгоритм быстрой сортировки. (рис. 26)

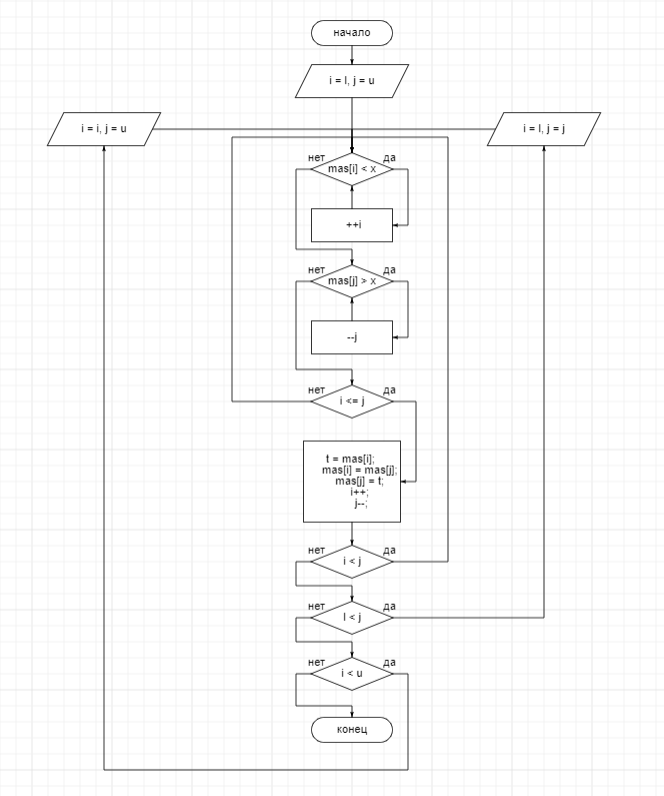


Рисунок 26. быстрая сортировка

# Эксперименты

Проведем некоторые эксперименты. Возьмем 10 элементов от 1 до 2 и посмотрим, какая сортировка будет быстрее. Как можно заметить, на 10 элементах все одинаково быстро отсортировали элементы. Теперь возьмем 10000 элементов от 1 до 100. Теперь уже видно наглядно, что самая быстрая — быстрая сортировка, а медленнее всех пузырьком. Возьмем на этот раз 1000000 элементов от 1 до 5, только сейчас без сортировки вставками. Как видно, при большем значение быстрая сортировка быстрее на полторы минуты.

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Пузырек, (секунд) | Вставками, (секунд) | Быстрая, (секунд) |
| 10 | 0,000000 | 0,000000 | 0,000000 |
| 10000 | 0,133000 | 0,007000 | 0,001000 |
| 1000000 | — | 91,962000 | 0,061000 |

# Заключение

В ходе данный работы мы узнали, как создавать алгоритмы сортировки, каких видов они бывают, как работать с текстовым файлом, насколько быстро работают сортировки и какие из них эффективнее.

# Литература

Керниган. Б. Язык программирования Си / Керниган Б., Ритчи Д. — «Вильямс»:

Издательский дом «Вильямс», 2015. — 304 с.

# Приложение 1

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <locale.h>
4. #include <math.h>
5. #pragma warning(disable : 4996)
6. int main()
7. {
8. int n, i;
9. double max, min;
10. printf("n: ");
11. scanf("%d", &n);
12. if (n <= 0)
13. {
14. printf("Error, n <= 0");
15. return 0;
16. }
17. printf("min: ");
18. scanf("%lf", &min);
19. printf("max: ");
20. scanf("%lf", &max);
21. if (min > max)
22. {
23. printf("Error, min > max");
24. return 0;
25. }
26. float\* mas = ((float\*)malloc(n \* sizeof(float) + 1));
27. for (i = 0; i < n; i++)
28. {
29. mas[i] = (((float)rand()) / RAND\_MAX \* (max - min) + min);
30. }
31. FILE\* a = fopen("mas.txt", "w");
32. fprintf(a, "n = %d\n", n);
33. for (i = 0; i < n; i++)
34. {
35. fprintf(a, "%f\n", mas[i]);
36. }
37. fclose(a);
38. free(mas);
39. return 0;
40. }
41. **Приложение 2**
42. #include <stdio.h>
43. #include <stdlib.h>
44. #include <math.h>
45. #include <locale.h>
46. #include <time.h>
47. #pragma warning(disable : 4996)
48. void scanN(FILE\* a, int\* n)
49. {
50. fscanf(a, "n = %d\n", n);
51. }
52. void inizmas(int n, float\* mas)
53. {
54. int i;
55. for (i = 0; i < n; i++)
56. {
57. mas[i] = 0;
58. }
59. }
60. void scanmas(FILE\* a, int n, float\* mas)
61. {
62. int i;
63. for (i = 0; i < n; i++)
64. {
65. fscanf(a, "%f\n", &mas[i]);
66. }
67. }
68. void printmas(int n, float\* mas)
69. {
70. int i;
71. for (i = 0; i < n; i++)
72. {
73. printf("%f\n", mas[i]);
74. }
75. }
76. void prisvoenie(float\* mas, float\* mas1, int n)
77. {
78. int i;
79. for (i = 0; i < n; i++)
80. {
81. mas1[i] = mas[i];
82. }
83. }
84. void puzyryok(int n, float\* mas)
85. {
86. int i, x;
87. float p;
88. for (i = 0; i < n; i++)
89. {
90. for (x = 0; x < n - i - 1; x++)
91. {
92. if (mas[x] > mas[x + 1])
93. {
94. p = mas[x];
95. mas[x] = mas[x + 1];
96. mas[x + 1] = p;
97. }
98. }
99. }
100. }
101. void vstavka(int n, float\* mas)
102. {
103. int i, x;
104. float p;
105. for (i = 1; i < n; i++)
106. {
107. p = mas[i];
108. for (x = i - 1; x >= 0; x--)
109. {
110. if (mas[x] < p)
111. {
112. break;
113. }
114. else
115. {
116. mas[x + 1] = mas[x];
117. }
118. }
119. mas[x + 1] = p;
120. }
121. }
122. void qSort(float\* mas, int l, int u)
123. {
124. int i = l, j = u;
125. float t = 0;
126. float x = mas[(int)(l + u) / 2];
127. do
128. {
129. while (mas[i] < x)
130. ++i;
131. while (mas[j] > x)
132. --j;
133. if (i <= j)
134. {
135. t = mas[i];
136. mas[i] = mas[j];
137. mas[j] = t;
138. i++;
139. j--;
140. }
141. } while (i < j);
142. if (l < j)
143. {
144. qSort(mas, l, j);
145. }
146. if (i < u)
147. {
148. qSort(mas, i, u);
149. }
150. }
151. void bystraya(float\* mas, int n)
152. {
153. qSort(mas, 0, n - 1);
154. }
155. int main()
156. {
157. int n, d = 0, y = 0;
158. FILE\* a = fopen("C:\\Users\\popov\\source\\repos\\prog 1\\mas.txt", "r");
159. scanN(a, &n);
160. float\* mas = ((float\*)malloc(n \* sizeof(double)));
161. float\* mas1 = ((float\*)malloc(n \* sizeof(double)));
162. inizmas(n, mas);
163. inizmas(n, mas1);
164. scanmas(a, n, mas);
165. prisvoenie(mas, mas1, n);
166. setlocale(LC\_ALL, "Russian");
167. printf("Выберите режим работы\n");
168. printf("1 — напечатать массив\n");
169. printf("2 — сортировать массив\n");
170. printf("3 — сброс\n");
171. printf("4 — выход из программы\n");
172. while (d != 4)
173. {
174. scanf("%d", &d);
175. if (d == 1)
176. {
177. printmas(n, mas);
178. }
179. else if (d == 2)
180. {
181. printf("Выберите тип сортировки\n");
182. printf("1 — пузырек\n");
183. printf("2 — вставкой\n");
184. printf("3 — быстрая\n");
185. while ((y != 1) && (y != 2) && (y != 3))
186. {
187. scanf("%d", &y);
188. if (y == 1)
189. {
190. clock\_t t1 = clock();
191. puzyryok(n, mas);
192. clock\_t t2 = clock();
193. printf("Время работы сортировки пузырьком — %lf секунд\n", ((double)(t2 - t1)) / CLOCKS\_PER\_SEC);
194. }
195. else if (y == 2)
196. {
197. clock\_t t3 = clock();
198. vstavka(n, mas);
199. clock\_t t4 = clock();
200. printf("Время работы сортировки вставками — %lf секунд\n", ((double)(t4 - t3)) / CLOCKS\_PER\_SEC);
201. }
202. else if (y == 3)
203. {
204. clock\_t t5 = clock();
205. bystraya(mas, n);
206. clock\_t t6 = clock();
207. printf("Время работы быстрой сортировки — %lf секунд\n", ((double)(t6 - t5)) / CLOCKS\_PER\_SEC);
208. }
209. else
210. {
211. printf("Введено неверное значение, повторите\n");
212. }
213. }
214. }
215. else if (d == 3)
216. {
217. prisvoenie(mas1, mas, n);
218. }
219. else if (d == 4)
220. {
221. printf("Программа закончила работу");
222. return 0;
223. }
224. else
225. {
226. printf("Введите корректное значение\n");
227. }
228. y = 0;
229. }
230. }