ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»**

Московский институт электроники и математики им. А.Н. Тихонова

Департамент прикладной математики

Отчёт
по лабораторной работе №9
по курсу «Алгоритмизация и программирование»
Задание № 13

ФИО студента	Номер группы	Дата
Кейер Александр Петрович	БПМ-231	3 марта 2024 г.

Москва, 2023

1 Задание (вариант № 13)

1. Реализовать указанные в варианте алгоритмы сортировки для массива объектов в соответствии с вариантом. Структура объекта описана в варианте ЛР8. Определить функции сравнения объектов по следующему принципу: приоритет сравнения определяется порядком поля в структуре, т.е. если структурный тип X содержит (в описании варианта) поля с именами a,b,c,d, то при сравнении двух объектов типа X надо сравнить поля с именем a, в случае их равенства сравнить поля b и т.д.

2. Каждый алгоритм сортировки должен поддерживать сортировку в обоих направлениях: по неубыванию и по невозрастанию.

3. Выбор и запуск требуемого алгоритма и направления сортировки осуществляется через меню на этапе выполнения.

4. Провести сортировку каждым алгоритмом массивов следующих размеров: 100, 1000, 5000, 10000, 20000, 50000, 100000 Засечь (программно) время сортировки каждым алгоритмом. По полученным точкам построить графики зависимости времени сортировки от размерности массива для каждого из алгоритмов сортировки на одной оси координат. Полученные графики включить в отчет к работе.

Реализовать сортировки:

- B – вставками
- D – шейкер
- E – слиянием

2 Структура заголовочного файла

```c
#define fieldLength 64
#define fieldSize fieldLength * sizeof(char)
#define entryLength 6

struct footballerType {
  char fullName[fieldLength];
  char clubName[fieldLength];
  char role[fieldLength];
  int age;
  int numberOfGames;
  int numberOfGoals;
};

```

3 Решение

```c
#include "football.h" // Footballer info.

#include <stdio.h> // Input/output library.
#include <stdlib.h> // Memory allocation.
#include <time.h> // Time library.
#include <assert.h> // Assertion library.
#include <string.h> // String functions library.

// Function getting direction of sorting by user.
void getSortDirectionByUser(int* direction) {
  printf("Enter sort direction (1 - increasing, -1 -
decreasing): ");

  fflush(stdin); // Clear stdin flow.
  scanf("%d", direction);
}

// Function printing horizontal line.
void printHr(int length) {
  for (int j = 0; j < length; j++) {
    printf("- ");
  }
  printf("\n");
}

```

```
25      // Function printing table header.
26      void printTableHeader() {
27        printHr(65);
28        printf("%4s|%30s|%30s|%30s|%10s|%10s|%10s|\n", "ID", "
        FULL NAME", "CLUB NAME", "ROLE", "AGE", "GAMES", "GOALS");
29        printHr(65);
30      }
31
32      // Function printing footballer.
33      void printFootballer(struct footballerType footballer,
        int id) {
34        printf("%4d|%30s|%30s|%30s|%10d|%10d|%10d|\n", id,
        footballer.fullName, footballer.clubName, footballer.role,
         footballer.age, footballer.numberOfGames, footballer.
        numberOfGoals);
35        printHr(65);
36      }
37
38      // Function printing footballers.
39      void printFootballers(struct footballerType* footballers,
        int length) {
40        if (footballers == NULL) {
41          printf("Incorrect array.\n");
42          return;
43        }
44
45        printTableHeader();
46
47        for (int i = 0; i < length; i++) {
48          printFootballer(footballers[i], i);
49        }
50      }
51
52      // Function generating string.
53      char* generateString(int length, int countOfUsedSymbols)
        {
54        assert(countOfUsedSymbols <= 26);
55
56        char* out = (char*)malloc(sizeof(char) * (length + 1));
57        char alphabet[26] = {
58          'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k'
        , 'l', 'm',
59          'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x'
        , 'y', 'z'
60        };
```

```
61
62      for (int i = 0; i < length; i++) {
63        out[i] = alphabet[rand() % countOfUsedSymbols];
64      }
65
66      out[length] = '\0';
67
68      return out;
69    }
70
71    // Function generating footballer.
72    struct footballerType generateFootballer() {
73      struct footballerType out = {
74        .fullName=generateString(1, 1),
75        .clubName=generateString(1, 1),
76        .role=generateString(1, 1),
77        .age=(rand() % 100),
78        .numberOfGames=(rand() % 100),
79        .numberOfGoals=(rand() % 100),
80      };
81
82      return out;
83    }
84
85    // Function generating footballer array.
86    struct footballerType* generateFootballersArray(int
    length) {
87      struct footballerType* arr = (struct footballerType*)
    malloc(sizeof(struct footballerType) * length);
88
89      for (int i = 0; i < length; i++) {
90        arr[i] = generateFootballer();
91      }
92
93      printf("Successfully generated %d footballers array.\n"
    , length);
94      return arr;
95    }
96
97    // Function comparing footballers.
98    int compareFootballers(struct footballerType footballer1,
    struct footballerType footballer2, int direction) {
99      int cmpFullNames = strcmp(footballer1.fullName,
    footballer2.fullName);
100
```

```
101       if (cmpFullNames != 0) {
102         return direction * cmpFullNames;
103       }
104
105       int cmpClubNames = strcmp(footballer1.clubName,
    footballer2.clubName);
106
107       if (cmpClubNames != 0) {
108         return direction * cmpClubNames;
109       }
110
111       int cmpRole = strcmp(footballer1.role, footballer2.role
    );
112
113       if (cmpRole != 0) {
114         return direction * cmpRole;
115       }
116
117       if (footballer1.age != footballer2.age) {
118         return direction * (footballer1.age - footballer2.age
    );
119       }
120
121       if (footballer1.numberOfGames != footballer2.
    numberOfGames) {
122         return direction * (footballer1.numberOfGames -
    footballer2.numberOfGames);
123       }
124       if (footballer1.numberOfGoals != footballer2.
    numberOfGoals) {
125         return direction * (footballer1.numberOfGoals -
    footballer2.numberOfGoals);
126       }
127     }
128
129     // Bubble sotring function.
130     void bubbleSort(struct footballerType* arr, int n, char
    direction) {
131       if (arr == NULL) {
132         printf("Incorrect array.\n");
133         return;
134       }
135
136       struct footballerType tmp;
137
```

```
138       clock_t start = clock();
139
140       for (int i = n - 1; i >= 0; i--) {
141         for (int j = 0; j < i; j++) {
142           if (compareFootballers(arr[j], arr[j + 1],
    direction) > 0) {
143             tmp = arr[j + 1];
144             arr[j + 1] = arr[j];
145             arr[j] = tmp;
146           }
147         }
148       }
149       clock_t stop = clock();
150
151       printf("%5s%6d: %.03fs\n", "n=", n, (double)(stop -
    start) / CLOCKS_PER_SEC);
152     }
153
154     // Function sorting by insertions.
155     void insertSort(struct footballerType* arr, int n, char
    direction) {
156       if (arr == NULL) {
157         printf("Incorrect array.\n");
158         return;
159       }
160
161       int j;
162       struct footballerType tmp;
163
164       clock_t start = clock();
165
166       for (int i = 1; i < n; i++) {
167         tmp = arr[i];
168
169         for (j = i - 1; j >= 0 && compareFootballers(arr[j],
    tmp, direction) > 0; j--) {
170           arr[j + 1] = arr[j];
171         }
172
173         arr[j + 1] = tmp;
174       }
175
176       clock_t stop = clock();
177
178       printf("%5s%6d: %.03fs\n", "n=", n, (double)(stop -
```

```
   start) / CLOCKS_PER_SEC);
179  }
180
181  // Shaker sorting function.
182  void shakerSort(struct footballerType* arr, int n, char
   direction) {
183    if (arr == NULL) {
184      printf("Incorrect array.\n");
185      return;
186    }
187
188    struct footballerType tmp;
189    int  l = 1, r = n;
190    int j;
191
192    clock_t start = clock();
193
194    do {
195      for(j = --r; j >= l; j--) {
196        if (compareFootballers(arr[j - 1], arr[j],
   direction) > 0) {
197          tmp = arr[j - 1];
198          arr[j - 1] = arr[j];
199          arr[j] = tmp;
200        }
201      }
202
203      for (j = ++l; j <= r; j++) {
204        if (compareFootballers(arr[j - 1], arr[j],
   direction) > 0) {
205          tmp = arr[j - 1];
206          arr[j - 1] = arr[j];
207          arr[j] = tmp;
208        }
209      }
210    } while (l < r);
211
212    clock_t stop = clock();
213
214    printf("%5s%6d: %.03fs\n", "n=", n, (double)(stop -
   start) / CLOCKS_PER_SEC);
215  }
216
217  // Functio mergin two arrays.
218  void merge(struct footballerType* arr, int l, int m, int
```

```
r, char direction) {
219     int i = l;
220     int j = m + 1;
221     int k = 0;
222 
223     struct footballerType* tmp = (struct footballerType*)
    malloc(sizeof(struct footballerType) * (r - l + 1));
224 
225     while (i <= m && j <= r) {
226       if (compareFootballers(arr[i], arr[j], direction) >=
    0) {
227         tmp[k] = arr[j];
228         j++;
229       } else {
230         tmp[k] = arr[i];
231         i++;
232       }
233 
234       k++;
235     }
236 
237     if (i > m) {
238       while (j <= r) {
239         tmp[k] = arr[j];
240 
241         j++;
242         k++;
243       }
244     } else {
245       while (i <= m) {
246         tmp[k] = arr[i];
247 
248         i++;
249         k++;
250       }
251     }
252 
253     for (k = 0; k <= r - l; k++) {
254       arr[l + k] = tmp[k];
255     };
256 
257     free(tmp); // Free allocated memory.
258   }
259 
260   // Function spliting and merging sorting array.
```

```
261   void splitAndMerge(struct footballerType* arr, int l, int
      r, char direction) {
262     if (l < r) {
263       int m = (l + r) / 2;
264       splitAndMerge(arr, l, m, direction);
265       splitAndMerge(arr, m + 1, r, direction);
266       merge(arr, l, m, r, direction);
267     }
268   }
269
270   // Merge sotring function.
271   void mergeSort(struct footballerType* arr, int n, char
    direction) {
272     if (arr == NULL) {
273       printf("Incorrect array.\n");
274       return;
275     }
276
277     clock_t start = clock();
278     splitAndMerge(arr, 0, n - 1, direction);
279     clock_t stop = clock();
280
281     printf("%5s%6d: %.03fs\n", "n=", n, (double)(stop -
    start) / CLOCKS_PER_SEC);
282   }
283
284   // Function printing main operations codes list.
285   void printMainMenuOperationsList() {
286     printf("\n");
287     printf("%30s %3s", "generate array:", "1\n");
288     printf("%30s %3s", "sort by insertions:", "2\n");
289     printf("%30s %3s", "bubble sort:", "3\n");
290     printf("%30s %3s", "shaker sort:", "4\n");
291     printf("%30s %3s", "merge sort:", "5\n");
292     printf("%30s %3s", "print array:", "6\n");
293     printf("%30s %3s", "exit program:", "7\n");
294     printf("\n\n");
295   }
296
297   // Function starting main menu.
298   void startMainMenu(struct footballerType* arr, int *pn) {
299     printf("\n");
300     printMainMenuOperationsList();
301
302     int operationCode = 0;
```

```
303         int direction = 1;
304 
305         printf("Enter correct operation code: ");
306 
307         fflush(stdin); // Clear stdin flow.
308         scanf("%d", &operationCode);
309 
310         switch (operationCode) {
311           case 1: {
312             printf("Enter footballers count: ");
313 
314             fflush(stdin);
315             scanf("%d", pn);
316 
317             arr = generateFootballersArray(*pn);
318 
319             break;
320           }
321 
322           case 2: {
323             getSortDirectionByUser(&direction);
324             insertSort(arr, *pn, direction);
325             break;
326           }
327 
328           case 3: {
329             getSortDirectionByUser(&direction);
330             bubbleSort(arr, *pn, direction);
331             break;
332           }
333 
334           case 4: {
335             getSortDirectionByUser(&direction);
336             shakerSort(arr, *pn, direction);
337             break;
338           }
339 
340           case 5: {
341             getSortDirectionByUser(&direction);
342             mergeSort(arr, *pn, direction);
343             break;
344           }
345 
346           case 6: {
347             printFootballers(arr, *pn);
```

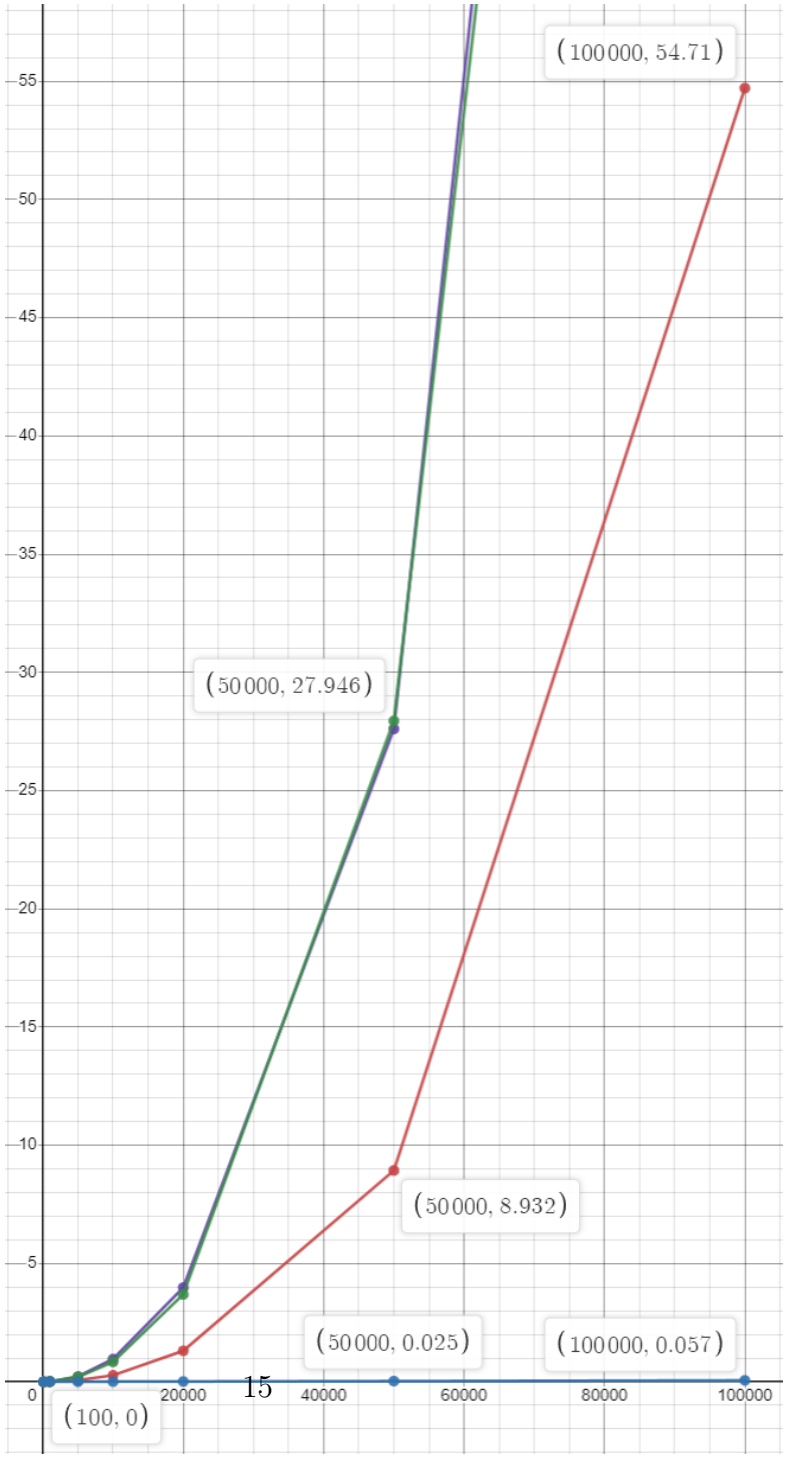
```
348           break;
349         }
350
351         case 7: {
352           return;
353         }
354       }
355
356       printf("\n");
357       startMainMenu(arr, pn);
358     }
359
360     // Function running tests.
361     void runTests() {
362       printf("Sort by insertions.\n\n");
363
364       insertSort(generateFootballersArray(100), 100, 1);
365       insertSort(generateFootballersArray(1000), 1000, 1);
366       insertSort(generateFootballersArray(5000), 5000, 1);
367       insertSort(generateFootballersArray(10000), 10000, 1);
368       insertSort(generateFootballersArray(20000), 20000, 1);
369       insertSort(generateFootballersArray(50000), 50000, 1);
370       insertSort(generateFootballersArray(100000), 100000, 1)
    ;
371
372       printf("\n----------\n\n");
373
374       printf("Bubble sort.\n\n");
375
376       bubbleSort(generateFootballersArray(100), 100, 1);
377       bubbleSort(generateFootballersArray(1000), 1000, 1);
378       bubbleSort(generateFootballersArray(5000), 5000, 1);
379       bubbleSort(generateFootballersArray(10000), 10000, 1);
380       bubbleSort(generateFootballersArray(20000), 20000, 1);
381       bubbleSort(generateFootballersArray(50000), 50000, 1);
382       bubbleSort(generateFootballersArray(100000), 100000, 1)
    ;
383
384       printf("\n----------\n\n");
385
386       printf("Shaker sort.\n\n");
387
388       shakerSort(generateFootballersArray(100), 100, 1);
389       shakerSort(generateFootballersArray(1000), 1000, 1);
390       shakerSort(generateFootballersArray(5000), 5000, 1);
```

```
391       shakerSort(generateFootballersArray(10000), 10000, 1);
392       shakerSort(generateFootballersArray(20000), 20000, 1);
393       shakerSort(generateFootballersArray(50000), 50000, 1);
394       shakerSort(generateFootballersArray(100000), 100000, 1)
      ;
395
396       printf("\n----------\n\n");
397
398       printf("Merge sort.\n\n");
399
400       mergeSort(generateFootballersArray(100), 100, 1);
401       mergeSort(generateFootballersArray(1000), 1000, 1);
402       mergeSort(generateFootballersArray(5000), 5000, 1);
403       mergeSort(generateFootballersArray(10000), 10000, 1);
404       mergeSort(generateFootballersArray(20000), 20000, 1);
405       mergeSort(generateFootballersArray(50000), 50000, 1);
406       mergeSort(generateFootballersArray(100000), 100000, 1);
407     }
408
409     int main() {
410       srand(time(NULL)); // Init first random number.
411
412       struct footballerType* arr = NULL;
413       int pn;
414
415       startMainMenu(arr, &pn);
416       // runTests();
417
418       return 0;
419     }
420
```

4 Тесты

4.1 Тест 1 (графики и оценка эффективности)

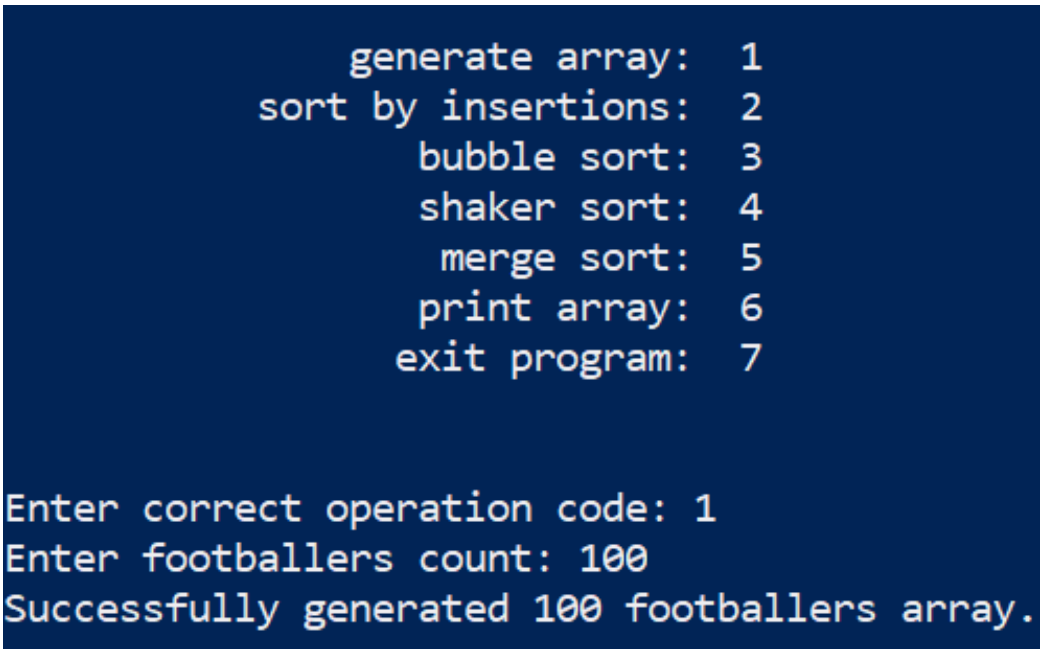
```
Sort by insertions.

   n=   100: 0.000s
   n=  1000: 0.003s
   n=  5000: 0.064s
   n= 10000: 0.272s
   n= 20000: 1.306s
   n= 50000: 8.932s
   n=100000: 54.710s

----------

Bubble sort.

   n=   100: 0.000s
   n=  1000: 0.008s
   n=  5000: 0.216s
   n= 10000: 0.957s
   n= 20000: 3.981s
   n= 50000: 27.609s
   n=100000: 165.531s

----------

Shaker sort.

   n=   100: 0.000s
   n=  1000: 0.008s
   n=  5000: 0.200s
   n= 10000: 0.840s
   n= 20000: 3.693s
   n= 50000: 27.946s
   n=100000: 156.804s

----------

Merge sort.

   n=   100: 0.000s
   n=  1000: 0.001s
   n=  5000: 0.002s
   n= 10000: 0.005s
   n= 20000: 0.009s
   n= 50000: 0.025s
   n=100000: 0.057s
```



По времени эффективнее всех оказывается сортировка слиянием. Даль-

ше по порядку идут: сортировка вставками, сортировка шейкером, пузырьковая сортировка. Естественно, по затратам памяти самой неэффективной оказывается сортировка слиянием, но затраты по памяти полностью компенсируются временными.

4.2 Тест 2



Успешная генерация массива футболистов соответствующего размера.

4.3 Тест 3

```
Enter correct operation code: 6
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ID|                    FULL NAME|                   CLUB NAME|                        ROLE|        AGE|      GAMES|      GOALS|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   0|                            a|                           a|                           a|         25|         81|         71|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   1|                            a|                           a|                           a|         53|         54|         16|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   2|                            a|                           a|                           a|         29|         74|          2|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   3|                            a|                           a|                           a|         87|         69|         94|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   4|                            a|                           a|                           a|         43|         40|          3|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   5|                            a|                           a|                           a|          1|         40|         96|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   6|                            a|                           a|                           a|         93|         13|         77|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   7|                            a|                           a|                           a|         96|         25|         72|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   8|                            a|                           a|                           a|         83|         77|         18|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   9|                            a|                           a|                           a|         61|         76|         82|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Успешный вывод массива футболистов соответствующего размера.

4.4 Тест 4

```
Enter correct operation code: 5
Enter sort direction (1 - increasing, -1 - decreasing): -1
   n=    10: 0.000s



               generate array:  1
           sort by insertions:  2
                  bubble sort:  3
                  shaker sort:  4
                   merge sort:  5
                  print array:  6
                 exit program:  7


Enter correct operation code: 6
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  ID|                    FULL NAME|                   CLUB NAME|                        ROLE|        AGE|      GAMES|      GOALS|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   0|                            a|                           a|                           a|         96|         25|         72|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   1|                            a|                           a|                           a|         93|         13|         77|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   2|                            a|                           a|                           a|         87|         69|         94|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   3|                            a|                           a|                           a|         83|         77|         18|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   4|                            a|                           a|                           a|         61|         76|         82|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   5|                            a|                           a|                           a|         53|         54|         16|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   6|                            a|                           a|                           a|         43|         40|          3|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   7|                            a|                           a|                           a|         29|         74|          2|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   8|                            a|                           a|                           a|         25|         81|         71|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   9|                            a|                           a|                           a|          1|         40|         96|
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Успешная сортировка массива футболистов в соответствующем направлении.