

ГУАП

КАФЕДРА № 43

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

ассистент

\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

К.А. Кочин

\_\_\_\_\_  
инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

АЛГОРИТМЫ СОРТИРОВКИ

по курсу: Структуры и алгоритмы обработки данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4136

\_\_\_\_\_  
подпись, дата

Бобрович Н. С.

\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2022

## Цель работы

Целью работы является изучение алгоритмов внутренней сортировки и получение практических навыков их использования, и анализа их сложности.

## Задание

Вариант №11.

Использовать неупорядоченный массив  $A$ , содержащий  $n$  целочисленных элементов. Величина  $n$  определяется по согласованию с преподавателем. Дополнительно в программе должны быть реализованы следующие функции:

- 1) Поиск элемента либо по его порядковой позиции, либо по его содержимому;
- 2) Добавление/удаление элемента с последующей пересортировкой последовательности;
- 3) В программе должен быть реализован подсчет количества сравнений и перестановок, при осуществлении сортировки.

Найти количество повторяющихся чисел среди элементов массива сортировкой подсчётом.

## Листинг программы

```

1  #include <iostream>
2
3  using namespace std;
4
5  int gcount = 0;
6  const int n = 10;
7
8  int r(int min, int max)
9  {
10     int num = min + rand() % (max - min + 1);
11     return num;
12 }
13
14 void f1(int a, int* arr, const int n)
15 {
16     int count = 0;
17     for (int i = 0; i < n; i++) {
18         if (arr[i] == a) {
19             cout << "Элемент равный " << arr[i] << " находится на позиции " <<
20             count++;
21         }
22     }
23     if (count == 0) {
24         cout << "Элемента с таким значением нет." << endl;
25     }
26
27
28 void f2(int a, int* arr, const int n)
29 {
30     int count = 0;
31     for (int i = 0; i < n; i++) {
32         if (i == a) {
33             cout << "Элемент равный " << arr[i] << " находится на позиции " <<
34             count++;
35         }
36     }
37     if (count == 0) {
38         cout << "Элемента с такой позицией нет." << endl;
39     }
40
41
42 void sortch(int* arr, const int n)
43 {
44     int i, j, k, count;
45     count = 1;
46     int* brr = new int[n];
47     for (i = 0; i < n; i++) {
48         k = 0;
49         for (j = 0; j < n; j++) {
50             if ((arr[j] < arr[i]) || ((arr[j] == arr[i]) && (j < i))) {

```

```

51         k++;
52         gcount++;
53     }
54 }
55 brr[k] = arr[i];
56 }
57 for (i = 0; i < n; i++) {
58     arr[i] = brr[i];
59 }
60 i = 1;
61 while (i < n) {
62     if (arr[i] == arr[i - 1]) {
63         count++;
64         i++;
65         gcount++;
66     }
67     else {
68         cout << arr[i - 1] << " встречается " << count << " раз ";
69         count = 1;
70         i++;
71         gcount++;
72     }
73 }
74 cout << arr[i - 1] << " встречается " << count << " раз";
75 }

```

```

76
77 int main()
78 {
79     setlocale(LC_ALL, "Rus");
80     srand(time(NULL));
81
82     int n, m, k, l, i, x;
83     cout << "Введите количество элементов в массиве: ";
84     cin >> n;
85     int* a = new int[n];
86     cout << "Введите элементы массива, начиная с первого: ";
87     for (i = 0; i < n; i++) {
88         cin >> x;
89         a[i] = x;
90     }
91
92     cout << endl;
93     cout << "Если хотите узнать позицию элемента по номеру, нажмите 1.\nЕсли :
94     cin >> m;
95     if ((m != 1) && (m != 2) && (m != 0)) {
96         cout << "Error" << endl;
97     }
98     if (m == 1) {
99         cin >> k;
100         f1(k, a, n);

```

```

101     }
102     if (m == 2) {
103         cin >> k;
104         f2(k, a, n);
105     }
106     if (m == 0) {
107         cout << "Thanks" << endl;
108     }
109
110     sortch(a, n);
111
112     cout << endl;
113     cout << "Если хотите добавить элемент, нажмите 1.\nЕсли удалить элемент, н
114     cin >> m;
115     if ((m != 1) && (m != 2) && (m != 0)) {
116         cout << "Error" << endl;
117     }
118     if (m == 1) {
119         n++;
120         cout << "Введите значение элемента: " << endl;
121         cin >> l;
122         a[n - 1] = l;
123         sortch(a, n);
124     }
125     if (m == 2) {
126         cout << "Введите позицию элемента: " << endl;
127         cin >> l;
128         if (0 <= l < n) {
129             for (i = l; i < n; i++) {
130                 a[i - 1] = a[i];
131             }
132             n--;
133             sortch(a, n);
134         }
135         else {
136             cout << "Не попадает в количество элементов!" << endl;
137         }
138     }
139     if (m == 0) {
140         cout << "Thanks" << endl;
141     }
142
143     cout << "Общее количество сравнений и перестановок, при осуществлении сор
144
145     return 0;
146 }

```

## Контрольный пример

```
Консоль отладки Microsoft Visual Studio
Введите количество элементов в массиве: 5
Введите элементы массива, начиная с первого: 1
1
0
90
-2

Если хотите узнать позицию элемента по его значению, нажмите 1.
Если хотите узнать номер элемента по позиции, нажмите 2.
Если ничего не хотите, нажмите 0.
0
Thanks
-2 встречается 1 раз 0 встречается 1 раз 1 встречается 2 раз 90 встречается 1 раз
Если хотите добавить элемент, нажмите 1.
Если удалить элемент, нажмите 2.
Если ничего не хотите, нажмите 0.
1
Введите значение элемента:
5
-2 встречается 1 раз 0 встречается 1 раз 1 встречается 2 раз 5 встречается 1 раз 90 встречается 1 раз
Общее количество сравнений и перестановок, при осуществлении сортировки 34

C:\Users\User\source\repos\SAODLR5_100\x64\Debug\SAODLR5_100.exe (процесс 17532) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

## Количество выполненных операций сравнения и перестановок элементов массива

Переменная `gcount` в данном контрольном примере равна 34.

## Временную и пространственную сложность алгоритма

Разработанный алгоритм использует следующие данные:

- один массив размерностью  $n$ , один массив размерностью  $\max - \min$ ;
- семь переменных целого типа.

Значит, пространственная сложность алгоритма определяется следующим образом:

$v = n * \text{Cuint16\_t} + 7 * \text{Cuint16\_t}$ , где `Cuint16_t` – константа, характеризующая объем памяти, отводимый под переменную беззнакового 16-тиразрядного целого типа. Теоретическая пространственная сложность алгоритма составляет:

$$V(n) = O(v) = O(\max(O(n * \text{Cuint16\_t}), O(7 * \text{Cuint16\_t})))$$

Теоретическую временную сложность алгоритма определяем на основе анализа текста программы, реализующей данный алгоритм. Для начала рассчитаем теоретическую временную алгоритма сортировки:  
 $T_{\text{Sort}} = O(\max(O(K1), O(*K2))) = O(\max(O(1), O( ))) = O(n)$ , где  
 $K1$  – операции сравнения, присваивания, используемые в алгоритме и имеющие временную сложность «1»,  $K2$  – циклические операции, занимающие в наихудшем случае шагов.

## **Вывод**

Написан код, отвечающий всем требованиям. В итоге были изучены алгоритмы внутренней сортировки и получены практические навыки их использования, и анализа их сложности.