1. Цель работы

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

2. Задание

Согласно варианту №10:

Использовать неупорядоченный массив A, содержащий n целочисленных элементов. Величина n определяется по согласованию с преподавателем.

Найти количество различных чисел среди элементов массива	Распределением

3. Листинг программы

```
// Вариант 10
#include <iostream>

using namespace std;

int get_num_int() // Запрос и проверка числа на корректность
{
int x;

cin >> x;
while (cin.fail() || (cin.peek() != '\n')) // Проверка на корректность
{
cin.clear(); // Очищение флага ошибки
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Очистка буфера запроса
cout << "Повторите ввод: ";
cin >> x;
}

return x;
}

unsigned int get_u_int()
{
int n;

n = get_num_int();

while (n <= 0) // Проверка на корректный ввод
{
cout << "Повторите ввод: ";
n = get_num_int();
}

return n;
}

void full_array(int* array, unsigned int n)
{
for (unsigned int i = 0; i < n; i++) // Заполнение массива
{
array[i] = get_num_int();
}
}

void show_array(int* array, unsigned int n)
{
```

```
46. for (unsigned int i = 0; i < n; i++)
47. {
48. cout << array[i] << "\t";
49. }
50. cout << endl;
51. }
52.
53. int find_elem(int* array, unsigned int n, int find)
54. {
55. for (unsigned int i = 0; i < n; i++)
56. {
57. if (array[i] == find)
58. {
59. return i;
60. }
61. }
62. return -1;
63. }
64.
65. void delete_elem(int*& array, unsigned int &n, int find)
66. {
67. int* arr = new int[n - 1];
68.
69. for (unsigned int i = 0; i < n; i++)
70. {
71. if (i >= find)
72. arr[i] = array[i+1];
73. else
74. arr[i] = array[i];
75.
76. }
77. delete[] array;
78. array = arr;
79. n--;
80. }
81.
82. int ar_min(int* array, unsigned int n)
83. {
84. int mins = pow(10, 5);
85. for (unsigned int i = 0; i < n; i++)
86. {
87. if (array[i] < mins)
88. {
89. mins = array[i];
90. }
91. }
92. return mins;
93. }
94.
95. int ar_max(int* array, unsigned int n)
96. {
97. int maxs = pow(-10, 5);
98. for (unsigned int i = 0; i < n; i++)
99. {
100.       if (array[i] > maxs)
101.       {
102.       maxs = array[i];
103.       }
104.       }
105.       return maxs;
106.       }
107.
108.       int count_sort(int*& array, unsigned int n) // Counting sort
109.       {
110.       int ar_mins = ar_min(array, n);
```

```
111.      int ar_maxs = ar_max(array, n);
112.
113.      int* arr = new int[ar_maxs - ar_mins + 1];
114.      int* arrs = new int[n];
115.
116.      for (unsigned int x = 0; x < (ar_maxs - ar_mins + 1); x++)
117.      {
118.      arr[x] = 0;
119.      }
120.      for (unsigned int y = 0; y < n; y++)
121.      {
122.      arr[array[y] - ar_mins] += 1;
123.      }
124.
125.      unsigned int j;
126.      unsigned int smeh = 0;
127.      for (unsigned int i = 0; i < (ar_maxs - ar_mins + 1); i++)
128.      {
129.
130.      int ind = arr[i];
131.      for (j = 0; j < (ind); j++)
132.      {
133.      arrs[smeh] = i + ar_mins;
134.      smeh++;
135.      }
136.      }
137.
138.      int unic = 0;
139.      for (unsigned int i = 0; i < n; i++)
140.      {
141.      if (arr[i] == 1)
142.      unic += 1;
143.      }
144.
145.      delete[] arr;
146.      delete[] array;
147.      array = arrs;
148.
149.      return unic;
150.      }
151.
152.      void add_elem(int*& array, unsigned int& n, int find)
153.      {
154.      int* arr = new int[n + 1];
155.
156.      for (unsigned int i = 0; i < n; i++)
157.      {
158.      arr[i] = array[i];
159.      }
160.      arr[n] = find;
161.      delete[] array;
162.      array = arr;
163.      n++;
164.      }
165.
166.      int main()
167.      {
168.      setlocale(LC_ALL, "RUS");
169.      int choice;
170.
171.      cout << "Добро пожаловать в программу сортировки." << endl;
172.      cout << "Введите длину массива: " << endl;
173.      unsigned int n = get_u_int();
174.      int* array = new int[n];
175.      bool full = false;
```

```
176.
177.       while (true)
178.       {
179.       cout << "Выберите дальнейшее действие" << endl;
180.       cout << "1. Задать массив" << endl;
181.       if (full)
182.       {
183.       cout << "2. Вывести массив" << endl;
184.       cout << "3. Добавить элемент" << endl;
185.       cout << "4. Удалить элемент" << endl;
186.       cout << "5. Найти элемент" << endl;
187.       cout << "6. Отсортировать массив" << endl;
188.       }
189.       cout << "7. Выход" << endl;
190.       choice = get_u_int();
191.
192.       if (choice == 1)
193.       {
194.       system("cls");
195.       cout << "Заполните массив: " << endl;
196.       full_array(array, n);
197.       full = true;
198.       }
199.       else if (choice == 2)
200.       {
201.       system("cls");
202.       cout << "Ваш массив: " << endl;
203.       show_array(array, n);
204.       }
205.       else if (choice == 3)
206.       {
207.       system("cls");
208.       cout << "Введите элемент для добавления: " << endl;
209.       int choc;
210.       choc = get_num_int();
211.       add_elem(array, n, choc);
212.       count_sort(array, n);
213.       }
214.       else if (choice == 4)
215.       {
216.       system("cls");
217.       cout << "Введите элемент для удаления: " << endl;
218.       int choc;
219.       choc = get_num_int();
220.       int i = find_elem(array, n, choc);
221.       if (i >= 0)
222.       {
223.       if (n == 1)
224.       {
225.       delete[] array;
226.       full = false;
227.       cout << "Массив удалён" << endl;
228.       }
229.       else
230.       {
231.       delete_elem(array, n, i);
232.       cout << "Удалён элемент " << choc << " на позиции " << i << endl;
233.       }
234.       }
235.       else
236.       {
237.       cout << "Не найдено." << endl;
238.       }
239.       }
240.       else if (choice == 5)
```

```
241.      {
242.      choice:
243.      system("cls");
244.      cout << "Выберите дальнейшее действие" << endl;
245.      cout << "1. Поиск по позиции" << endl;
246.      cout << "2. Поиск по содержимому" << endl;
247.      cout << "3. Назад" << endl;
248.      int choc;
249.      choice = get_u_int();
250.      if (choice == 1)
251.      {
252.      system("cls");
253.      cout << "Введите позицию элемента для вывода: " << endl;
254.      choc = get_num_int();
255.      if ((choc > n) || (choc < 0))
256.      {
257.      cout << "Выход за границы массива." << endl;
258.      }
259.      else
260.      {
261.      cout << "Найденный элемент: " << array[choc] << endl;
262.      }
263.
264.      }
265.      else if (choice == 2)
266.      {
267.      system("cls");
268.      cout << "Введите содержимое для поиска: " << endl;
269.      choc = get_num_int();
270.      int i = find_elem(array, n, choc);
271.      if (i >= 0)
272.      {
273.      cout << "Элемент " << choc << " на позиции " << i << endl;
274.      }
275.      else
276.      {
277.      cout << "Не найдено." << endl;
278.      }
279.
280.      }
281.      else if (choice == 3)
282.      {
283.      system("cls");
284.      }
285.      else
286.      {
287.      goto choice;
288.      }
289.      }
290.      else if (choice == 6)
291.      {
292.      system("cls");
293.      int unic = count_sort(array, n);
294.      cout << "Массив отсортирован." << endl;
295.      cout << "Количество уникальных элементов: " << unic << endl;
296.      }
297.      else if (choice == 7)
298.      {
299.      cout << "Заверщение работы..." << endl;
300.      break;
301.      }
302.      else
303.      {
304.      system("cls");
305.      }
```

```
306.        }
307.
308.        delete[] array;
309.        }
```

4. Расчёт сложности алгоритма

Разработанный алгоритм использует следующие данные:

- массив размерностью n;
- 14 переменных целого типа;
- 1 переменная логического типа;
- 14 переменных целового типа без знака.

Пространственная пространственная сложность:

$V = 14* C_{int} + 14* C_{uint} + 1*C_{bool} + n* C_{int}$

где C_{int} – Константа, характеризующая память, выделяемая под переменную целого типа, C_{uint} - Константа, характеризующая память, выделяемая под переменную целого типа без знака, C_{bool} - Константа, характеризующая память, выделяемая под переменную логического типа.

Теоретическая пространственная сложность:

$V(n) = O(n)$ =O(max(O(n*C_{int}), O(14 * C_{int}), O(14 * C_{uint}),), O(1 * C_{bool}))) =

O(max(O(n), O(1), O(1)), O(1))) = O(n)

Временную сложность алгоритма сортировки определяем на основе анализа текста, реализующей этот алгоритм сортировки.

t_{Sort} = t_{min} + t_{max} + K114 + K115 + CONST*K119 + n*K123 + K126 + K127 +

n*(K131 + n * (K134 + K135)) + K139 + n*(max(K143,0)) + K146 + K147 + K148

где K_i – константа, характеризующая время выполнения операций, помеченных i;

t_{min}, t_{max} – временные сложности функций `ar_min, ar_max`.

Теоретическая временная сложность:

$t_{Sort}(n) = O(t_{Sort})$ = O(max(O(n), O(n), O(1)…, O(n), O(n^2))) = O(n^2)

5. Выводы

В процессе лабораторной работы были алгоритмы внутренней сортировки, навыки их реализации, а так же оценки сложности алгоритма.