Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт
по лабораторной работе № 3

Название: Алгоритмы сортировки

Дисциплина: Анализ алгоритмов

Студент ИУ7-55Б _____ Д.О. Склифасовский
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподователь _____ Л.Л. Волкова
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Оглавление

Введение

Цель работы: изучение алгоритмов сортировки массивов. В данной лабораторной работе рассматриваются 3 алгоритма:

1) сортировка пузырьком;
2) сортировка шейкером;
3) сортировка вставками.

Также требуется изучить рассчет сложности алгоритмов. В ходе лабораторной работы необходимо:

1) изучить алгоритмы сортировки;
2) дать теоритическую оценку сортировок пузырьком, шейкером и вставками;
3) реализовать три алгоритма сортировки на одном из языков программирования;
4) сравнить алгоритмы сортировки.

1 Аналитическая часть

В данном разделе представлено описание алгоритмов сортировки массивов.

1.1 Сортировка пузырьком

Сортировка пузырьком — один из самых известных алгоритмов сортировки. Здесь нужно последовательно сравнивать значения соседних элементов и менять числа местами, если предыдущее оказывается больше последующего. Таким образом элементы с большими значениями оказываются в конце списка, а с меньшими остаются в начале.

Этот алгоритм считается учебным и почти не применяется на практике из-за низкой эффективности: он медленно работает на тестах, в которых маленькие элементы (их называют «черепахами») стоят в конце массива. Однако на нём основаны многие другие методы, например, шейкерная сортировка и сортировка расчёской.

1.2 Сортировка шейкером

Шейкерная сортировка отличается от пузырьковой тем, что она двунаправленная: алгоритм перемещается не строго слева направо, а сначала слева направо, затем справа налево.

1.3 Сортировка вставками

При сортировке вставками массив постепенно перебирается слева направо. При этом каждый последующий элемент размещается так, чтобы он оказался между ближайшими элементами с минимальным и максимальным значением.

1.4 Вывод

Было представлено описание алгоритмов сортировки массивов. В основном все алгоритмы сортировок основаны на алгоритме сортировки пузырьком.

2 Конструкторская часть

В данном разделе представлены съемы разработанных алгоритмов. Также оценивается трудоемкость алгоритмов.

2.1 Разработка алгоритмов

На рисунке 1 изображена схема алгоритма сортировки пузырьком.

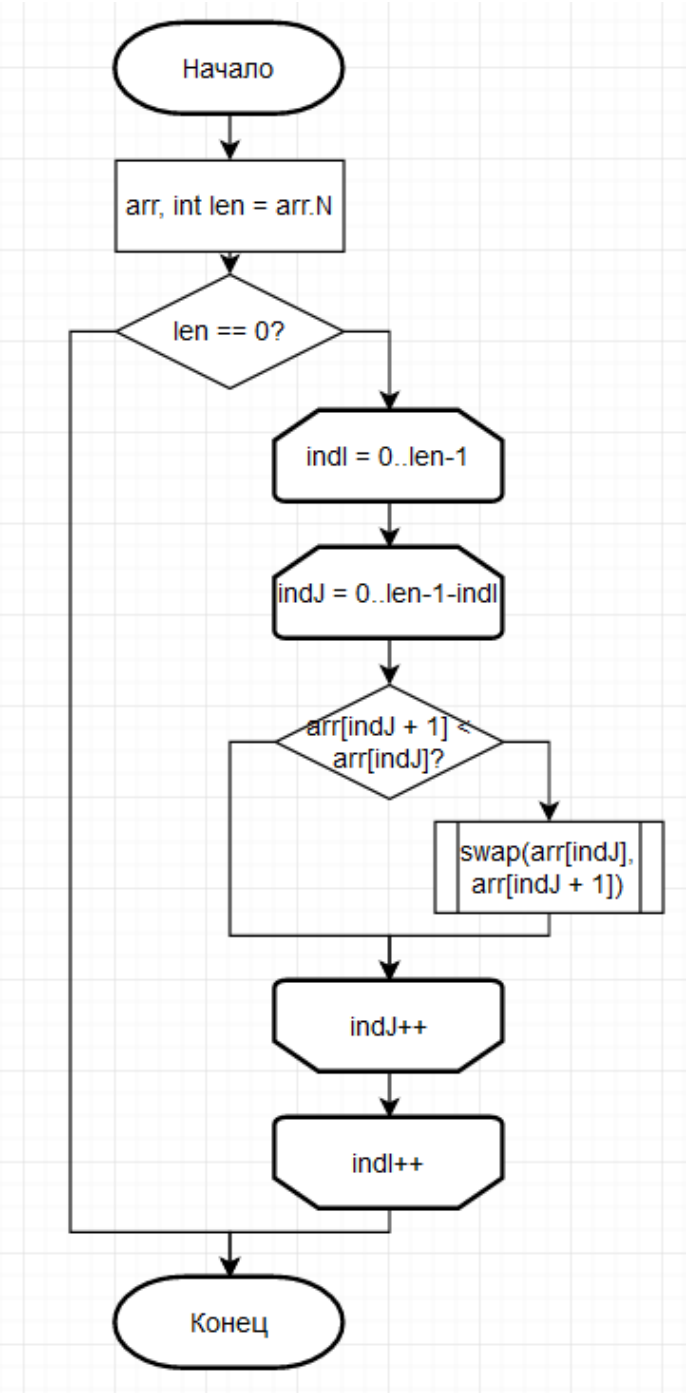

Рисунок 1. Схема алгоритма сортировки пузырьком

На рисунке 2 изображена схема алгоритма сортировки шейкером.

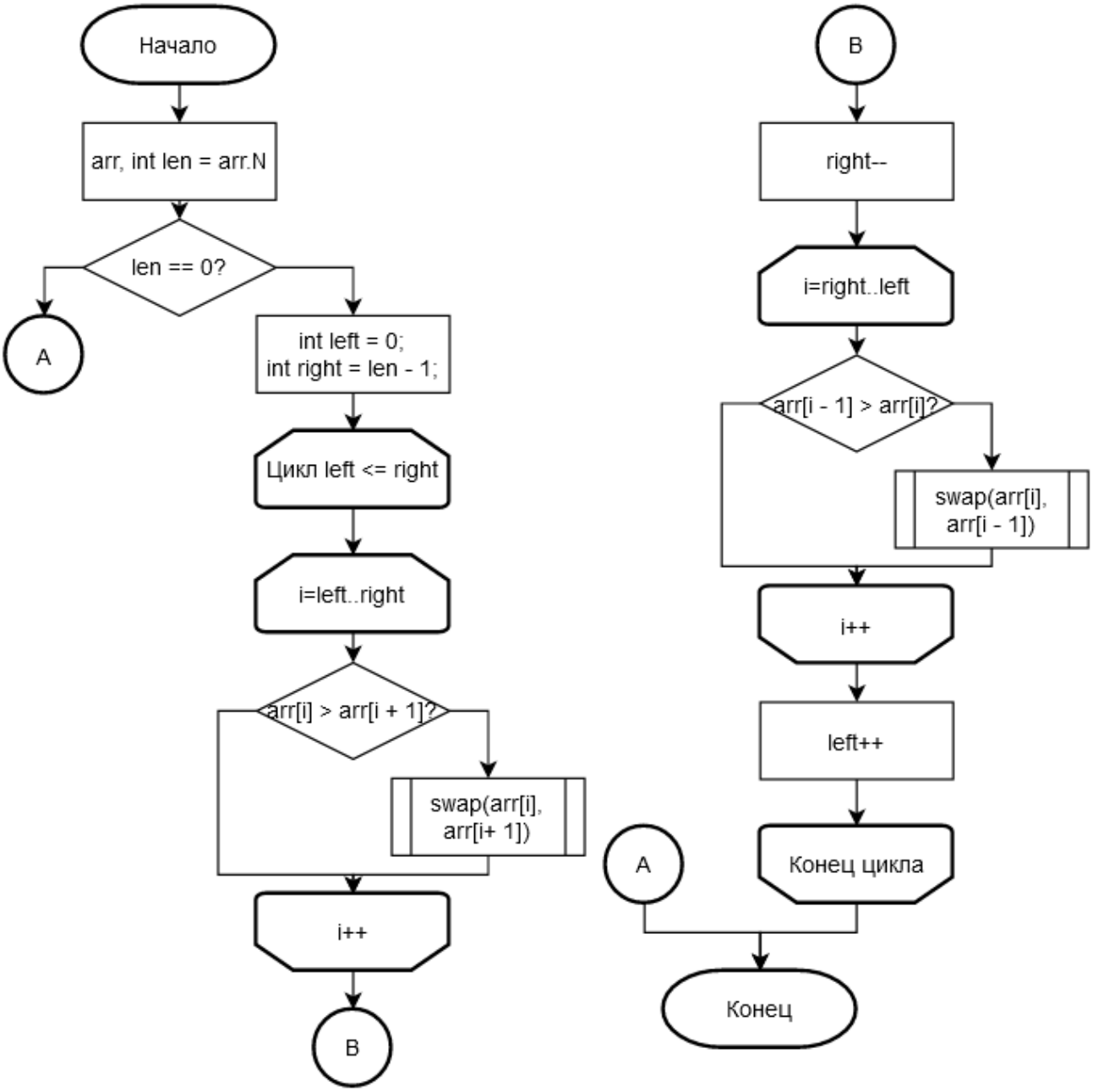

Рисунок 2. Схема алгоритма сортировки шейкером

На рисунке 3 изображена схема алгоритма сортировки вставками.

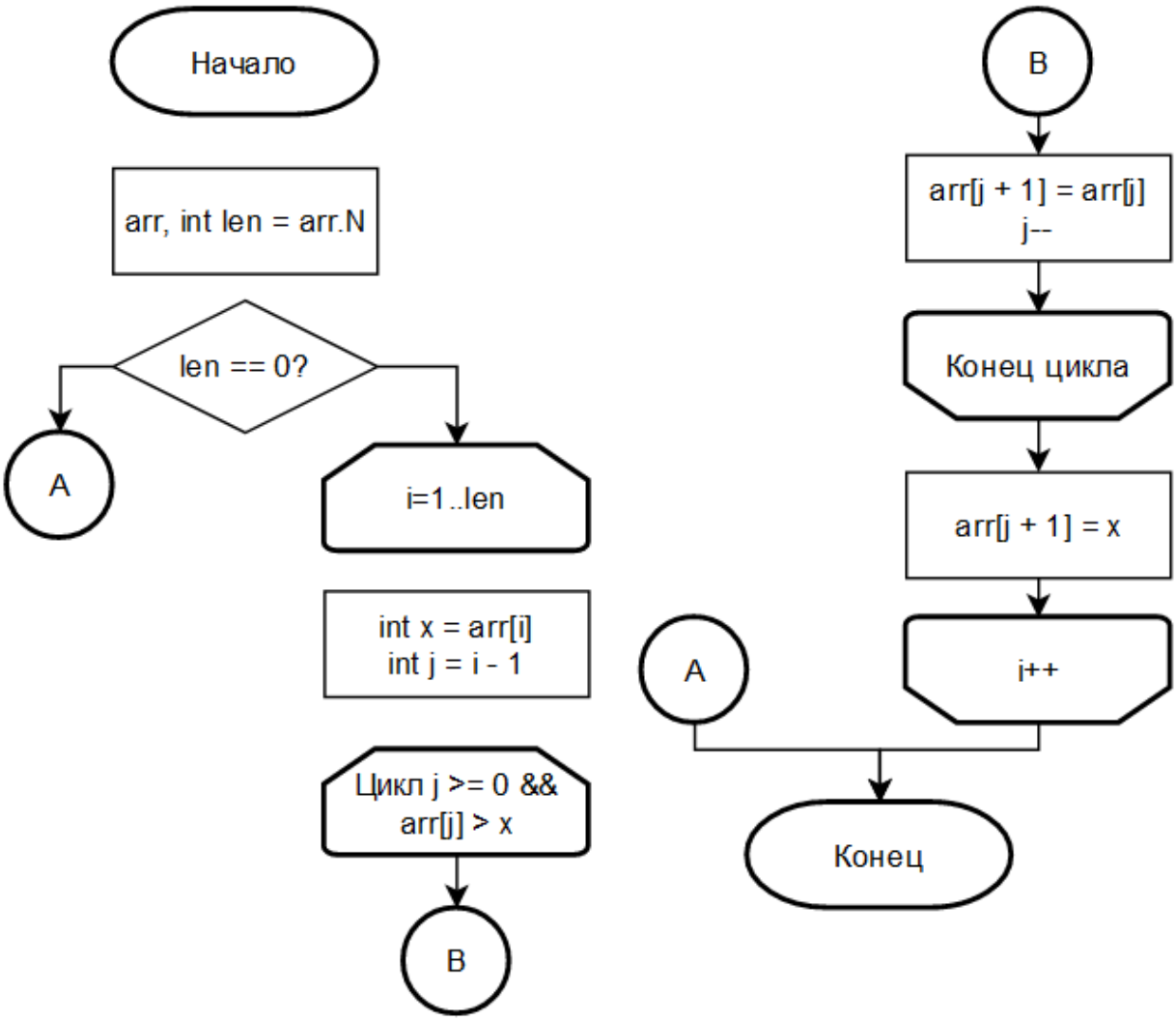

Рисунок 1. Схема алгоритма сортировки вставками

2.2 Модель трудоемкости

Модель трудоемкости для оценки алгоритмов:

1) стоимость базовых операций единица:

   $=, +, *, \simeq, <, >, \geq, \leq, ==, !=, [], +=, -=, *=, /=, ++, --;$

2) стоимость цикла:

   $f_{for} = f_{init} + f_{comp} + M(f_{body} + f_{increment} + f_{comp})$

   Пример: $for(i = 0, i < M; i++) / * body * /$

Результат: $2 + M(2 + f_{body})$;

3) стоимость условного оператора

Пусть goto (переход к одной из ветвей) стоит 0, тогда

$$f_f = \begin{cases} min(f_A, f_B), & \text{лучший случай} \\ max(f_A, f_B), & \text{худший случай} \end{cases}$$

4) операция обращения к ячейки матрицы [i, j] имеет трудоёмкость равную двум.

2.3 Оценка трудоемкости алгоритмов сортировки

Оценим трудоемкость алгоритмов.

2.3.1 Сортировка пузырьком

Лучший случай (массив отсортирован):

$$2 + 1 + 1 + 2 + (len - 1)(1 + 2+)$$

2.4 Вывод

В данном разделе были рассмотрены схемы алгоритмов сортировки массива, введена модель оценки трудоемкости алгоритма и были рассчитаны трудоемкости алгоритмов.

3 Технологическая часть

В данном разделе даны общие требования к программе, средства реализации и реализация алгоритмов.

3.1 Общие требования к программе

Требования к вводу:

1) вводится размер массива;
2) вводятся или автоматически генерируется массив.

1) при вводе неправильных размеров массива программа не должна завершаться аварийно;
2) должна выполняться корректная сортировка массива.

3.2 Средства реализации

В качестве языка программирования был выбран C#, так как я знаком с данным языком программирования, имею представление о способах тестирования программы. Средой разработки Visual Studio. Для замеров процессорного времени используется функция Stopwatch.

3.3 Сведения о модулях программы

Программа состоит из:

1) Program.cs - главный файл программы, в котором располагается точка входа в программу;

2) Array.cs - файл класса Array;

3) Sort.cs - файл класса Sort. В нем находятся алгоритмы сортировки массивов.

3.4 Листинг кода программы

Листинг 3.1: Класс Array для работы с массивами

```
1   class Array
2   {
3     private int[] array;
4     private int n;
5     public Array() { }
6
7     public Array(int n)
8     {
9       this.n = n;
10      array = new int[n];
11    }
12
13    public int N
14    {
15      get { return n; }
16      set { if (value > 0) n = 0; }
17    }
18
19    public int this[int i]
20    {
21      get { return array[i]; }
22      set { array[i] = value; }
```

```
23        }
24
25        public void Copy(Array arr)
26        {
27          for (int i = 0; i < n; i++)
28          {
29            arr[i] = array[i];
30          }
31        }
32
33        public void Read()
34        {
35          for (int i = 0; i < n; i++)
36          {
37            Console.Write(array[i] + "\t");
38          }
39          Console.WriteLine();
40        }
41
42        public void Fill()
43        {
44          Random rand = new Random();
45          for (int i = 0; i < n; i++)
46          {
47            array[i] = rand.Next(100);
48          }
49        }
50      }
```

Листинг 3.2: Алгоритм сортировки пузырьком

```
1       public static void BubbleSort(Array arr)
2           {
3               int len = arr.N;
4               if (len == 0)
5               {
6                   return;
7               }
8
```

```
            for (int indl = 0; indl + 1 < len; indl++)
            {
                for (int indJ = 0; indJ + 1 < len - indl; indJ++)
                {
                    if (arr[indJ + 1] < arr[indJ])
                    {
                        int tmp = arr[indJ];
                        arr[indJ] = arr[indJ + 1];
                        arr[indJ + 1] = tmp;
                    }
                }
            }
        }
```

Листинг 3.3: Алгоритм сортировки шейкером

```
    public static void ShakerSort(Array arr)
        {
            int len = arr.N;
            if (len == 0)
            {
                return;
            }
            int left = 0;
            int right = len - 1;
            while (left <= right)
            {
                for (int i = left; i < right; i++)
                {
                    if (arr[i] > arr[i + 1])
                    {
                        int tmp = arr[i];
                        arr[i] = arr[i + 1];
                        arr[i + 1] = tmp;
                    }
                }
                right--;

                for (int i = right; i > left; i--)
```

```
24                  {
25                      if (arr[i - 1] > arr[i])
26                      {
27                          int tmp = arr[i];
28                          arr[i] = arr[i - 1];
29                          arr[i - 1] = tmp;
30                      }
31                  }
32                  left++;
33              }
34          }
```

Листинг 3.4: Алгоритм сортировки вставками

```
1       public static void InsertionSort(Array arr)
2           {
3               int len = arr.N;
4               if (len == 0)
5               {
6                   return;
7               }
8               for (int i = 1; i < len; i++)
9               {
10                  int x = arr[i];
11                  int j = i - 1;
12                  for (; j >= 0 && arr[j] > x; j--)
13                  {
14                      arr[j + 1] = arr[j];
15                  }
16                  arr[j + 1] = x;
17              }
18          }
```

3.5 Вывод

В данном разделе были представлены сведения о модулях программы, а также реализованы три алгоритма сортировки массивов.

4 Экспериментальная часть

В данном разделе представлены результаты работы программы и приведен анализ времени работы каждого из алгоритмов.

4.1 Примеры работы программы