

# Информатика (основы программирования)

Лекция 12.

Алгоритмы сортировки массивов

Автор: Бабалова И.Ф.

Доцент, каф.12

2023 г.

# Быстрая сортировка

- Для ускорения процесса обмена предлагается обмениваться не упорядоченными элементами в последовательности на больших расстояниях.
- Выбирается средний элемент  $X$  и весь массив просматривается с 1-ого элемента - слева направо, пока не будет найден элемент последовательности  $a_i$ , больший выбранного для сравнения  $X$ .
- Затем начинается просмотр последовательности с другого конца - движение справа налево, пока не будет найден элемент  $a_i < X$ . Теперь можно поменять местами эти элементы.
- Первый же просмотр обеспечивает распределение элементов в последовательности таким образом, что слева находятся все числа, меньшие  $X$ , а справа все элементы большие  $X$ .
- Между собой элементы не упорядочены, что можно увидеть на числовом примере.
- Просмотр правой и левой подпоследовательностей продолжается до тех пор, пока не возникнет сравнение среднего элемента, то есть деление последовательностей будет завершено.

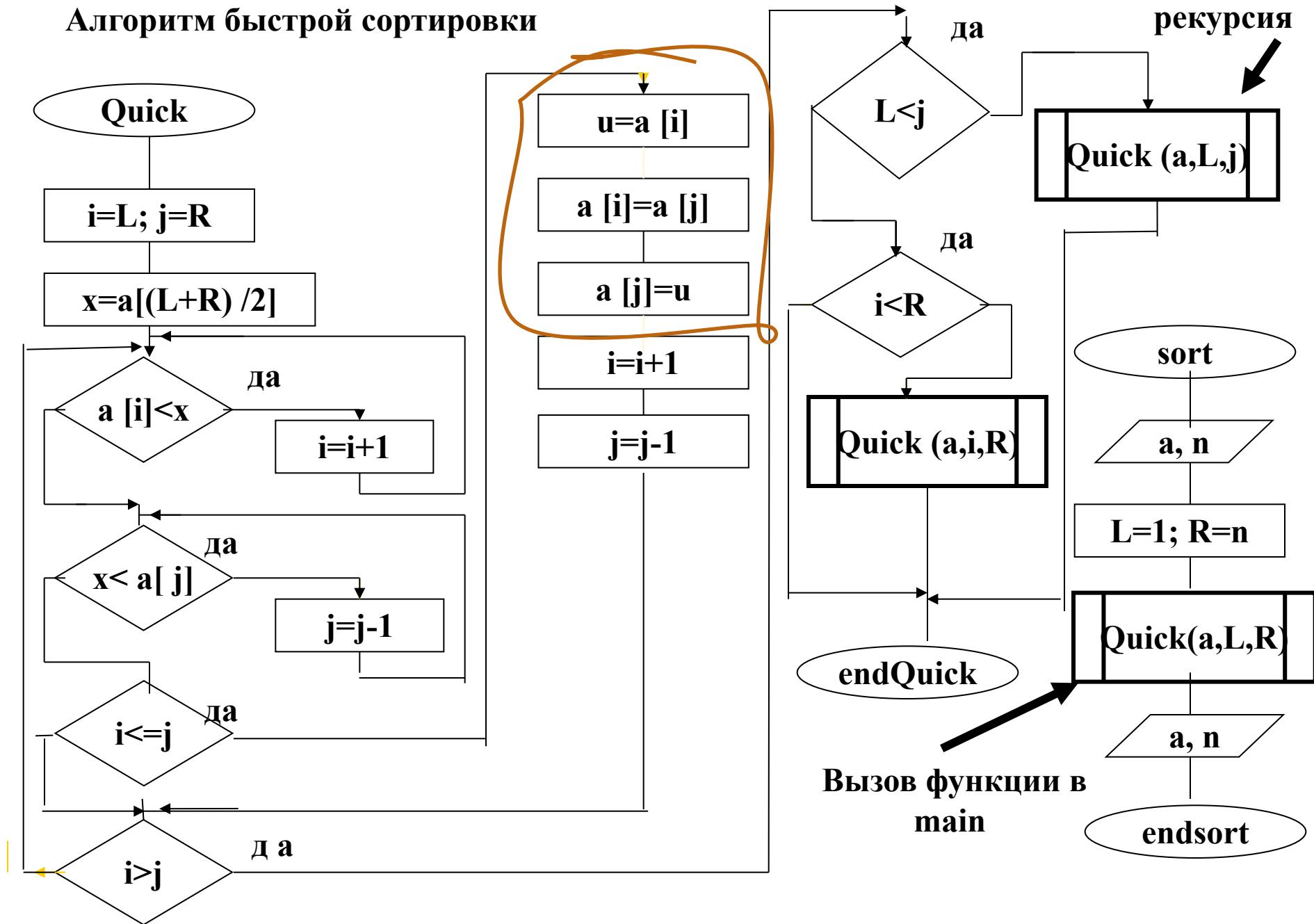
# Анализ быстрой сортировки

- Этот алгоритм обеспечивает уменьшение числа обменов за счет обменов на больших расстояниях. Берем последовательность. Выбираем некоторый средний элемент последовательности. Все меньшие его значения помещаются перед ним, а большие за ним.

43	5	5	5
55	17	12	12
12	12	17	17
41	41	41	41
93	93	93	43
17	43	43	55
5	55	55	64
64	64	64	93

- Окончание сортировки при  $\text{left} > \text{right}$
- Оценка времени выполнения быстрой сортировки в среднем  $\sim O(n \cdot \log(n))$
- Для других вариантов сортировки  $\sim O(n^2)$
- Решение задачи на языке программирования по этому алгоритму записывается созданием рекурсивной процедуры с новыми значениями левых и правых индексов сортируемой последовательности. Но такое использование метода имеет ограничения.

# Алгоритм быстрой сортировки



# Шейкер-сортировка (1)

Метод предлагает изменять направления сортировки массивов.

- Можно начать сортировку с конца последовательности.
- Наименьший элемент в этом случае окажется в начале последовательности.
- Затем движение начинаем со второго элемента и до конца последовательности. Наибольший элемент станет последним.
- Когда правая и левая граница совпадут, тогда массив будет отсортирован.
- Известно, что при частично упорядоченных последовательностях шейкер-сортировка имеет лучшие временные оценки, чем простые обменные сортировки.

# Шейкер-сортировка (2)

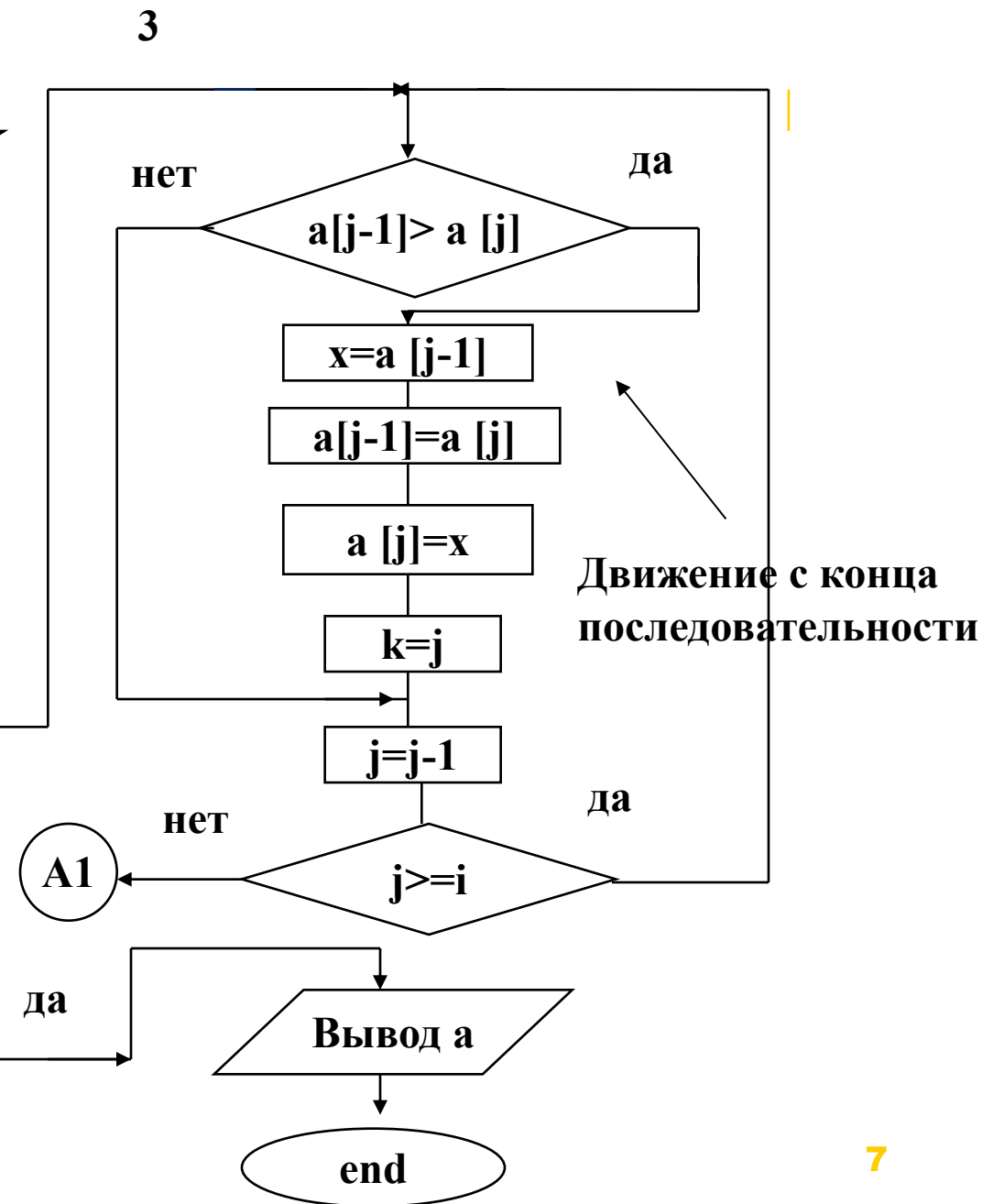
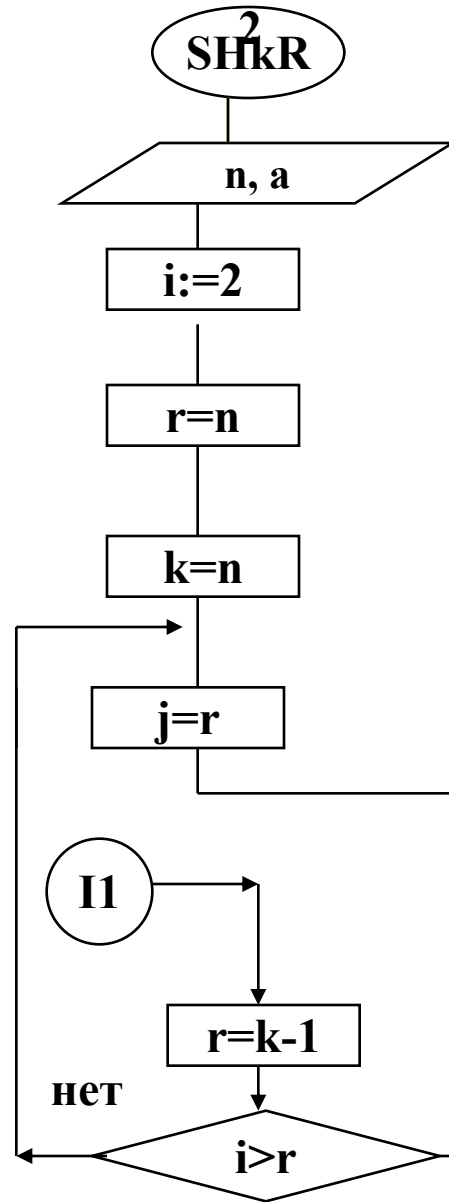
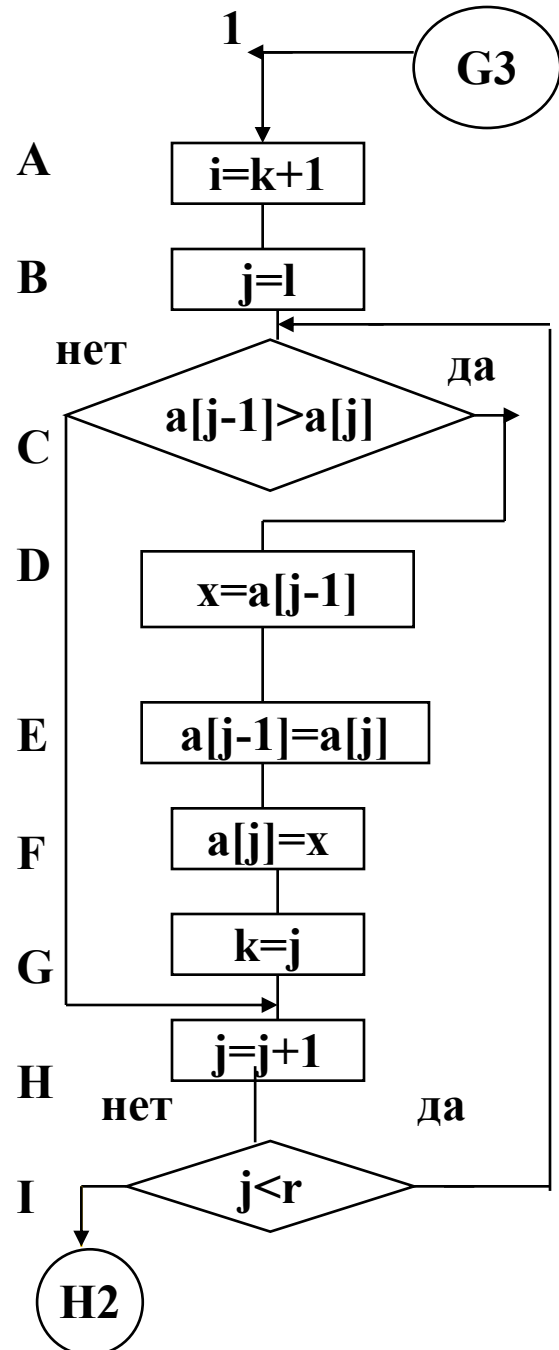
- В этом алгоритме изменяется направление движения по упорядочиваемой последовательности.

L	2	3	3	4	4
R	8	7	6	5	4
	43	5	5	5	5
→	55	43	43	12	12
	12	55	12	43	17
	41	12	41	17	41
	93	41	55	41	43
	17	93	17	55	55
	5	17	64	64	64
→	64	64	93	93	93

- Среднее число проходов, определенное Кнутом, равно  $\sim n - k_1 \sqrt{n}$ .
- Число обменов остается таким же, как и в простых методах сортировки.
- Управление процессом сортировки обеспечивается неравенством  $\text{left} < \text{right}$  ( $L < R$ ).
- При равенстве левой и правой границ массива процесс сортировки завершается.

# Алгоритм шейкер - сортировки

Движение с  
начала  
последова-  
тельности



# Функция шейкер-сортировки

```
void ShakerSort(int *a, int Start, int N)
{
    int Left, Right, i;
    Left=Start;
    Right=N-1;
    while (Left<=Right)
    {
        for (i=Right; i>=Left; i--)
            if (a[i-1]>a[i]) Swap(a, i);
        Left++;
        for (i=Left; i<=Right; i++)
            if (a[i-1]>a[i]) Swap(a, i);
        Right--;
    }
}
```



# Сортировка Шелла (1)

4	1	6	5	2	3	1 проход
4	1	3	5	2	6	2 проход
3	1	2	5	4	6	3 проход
1	2	3	4	5	6	результат


- Данный метод классифицируется как слияние с обменом. Часто называется сортировкой с убывающим шагом.
- В 1959 году американский ученый Дональд Шелл опубликовал алгоритм сортировки, который впоследствии получил его имя – «Сортировка Шелла». Этот алгоритм может рассматриваться как обобщение пузырьковой сортировки, так и сортировки вставками.

# Сортировка Шелла (2)


- Математики доказали, что максимальное время сортировки Шелла не превосходит  $O(n^{1.5})$ , причем уменьшить показатель степени 1.5 нельзя. Большое число экспериментов с сортировкой Шелла провели Джеймс Петерсон и Дэвид Л. Рассел в Стэнфордском университете в 1971 г. Они пытались определить среднее число перемещений при  $100 < n < 250000$  для последовательности шагов  $2^k-1$ .
- Наиболее подходящими формулами для оценки сходимости оказались  $1.21n^{1.26}$  и  $0.39n(\ln n) - 2.33n(\ln n)$ .
- Но при изменении диапазона  $n$  оказалось, что коэффициенты в представлении степенной функции практически не изменяются, а коэффициенты в логарифмическом представлении изменяются довольно резко. Поэтому естественно предположить, что именно степенная функция описывает истинное асимптотическое поведение сортировки Шелла.
- Время сортировки пропорционально  $n^{1.5}$  при сортировке  $n$  элементов. А это уже существенное улучшение по сравнению с сортировками порядка  $n^2$ .

# Сортировка «расчёской»

- Программисты опытным путём установили оптимальное расстояние между элементами для сравнения — это длина массива, поделённая на 1,247. С этим числом алгоритм работает быстрее всего.
- Название метода ассоциируется с расчёской в связи с тем, что просматриваются сравниваются значения элементов последовательностей с убывающим шагом. Начинается сравнение элементов с шагом  $N/1.247$  (Для упрощения можно делить на 1.25).
- Пример. Есть 6 значений в последовательности. Первый шаг будет равен 5.



Шаг = 5	4	1	6	5	2	3
Результат	2	1	6	5	4	3



Шаг = 4	4	1	6	5	2	3
Результат	2	1	3	5	4	6

- Окончание сортировки будет с шагом 2, когда вы сравниваете пары соседних элементов

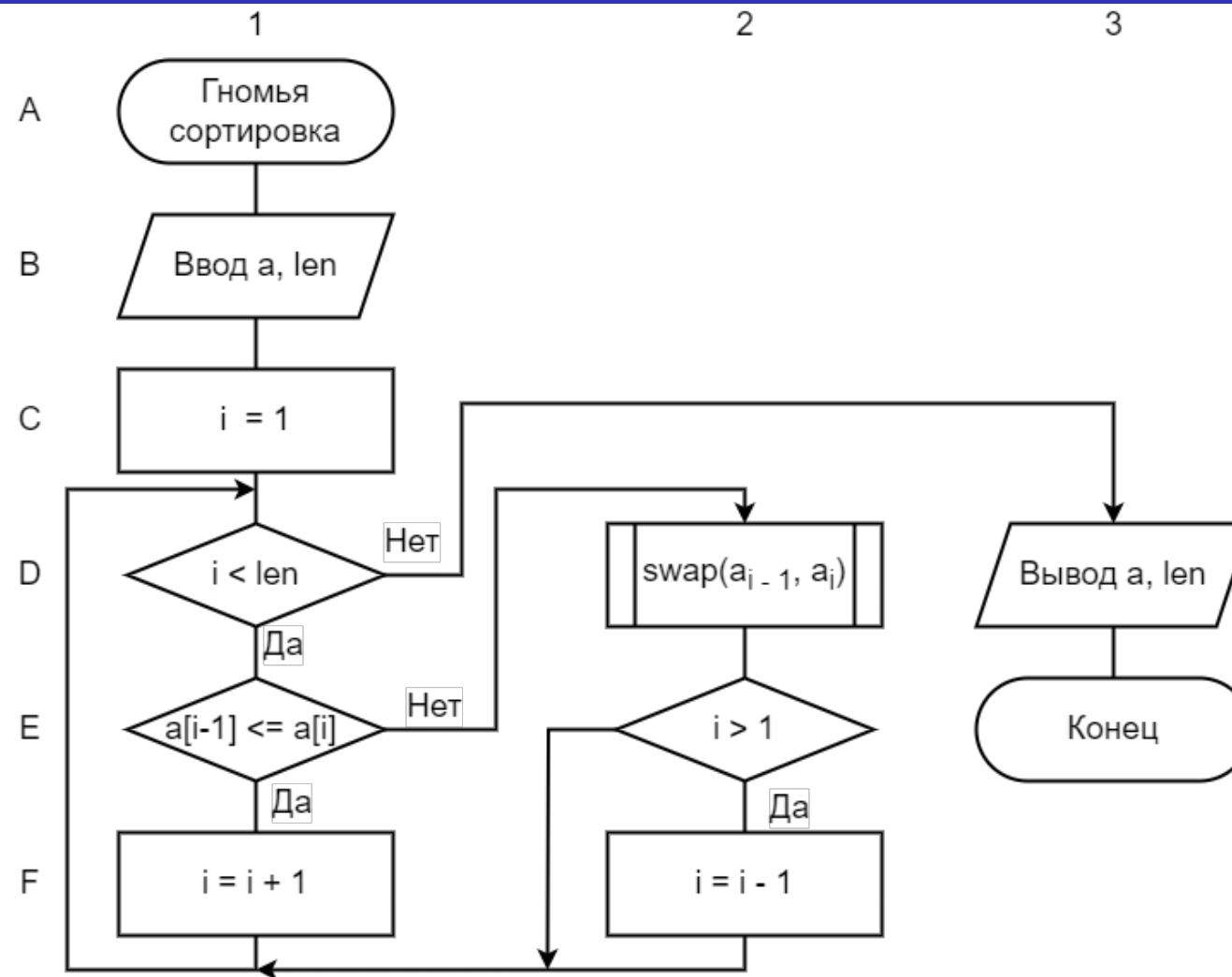
# Чётно-нечётная сортировка

- Это вариант обменной сортировки при движении слева-направо по последовательности. Сравниваем пары, у которых первый элемент – нечётный по индексу, а второй – чётный (т.е. первый и второй, третий и четвёртый, и т.д.). А потом наоборот – чётный-нечётный (второй и третий, четвёртый и пятый, и т.д.)
- В этом случае большие элементы последовательности на одной итерации одновременно делают один шаг вперед (в пузырьке самый крупный за итерацию доходит до конца, но остальные практически все остаются на месте).
- За два цикла сравнений последовательность будет упорядочена.

# «Гномья» сортировка

- Это также обменная сортировка, но при обмене индекс массива не возвращается в самое начало, а делает всего один шаг назад. Этого, оказывается, вполне достаточно, чтобы отсортировать последовательность значений. Движение назад будет до тех пор, пока остаются неупорядоченные элементы.

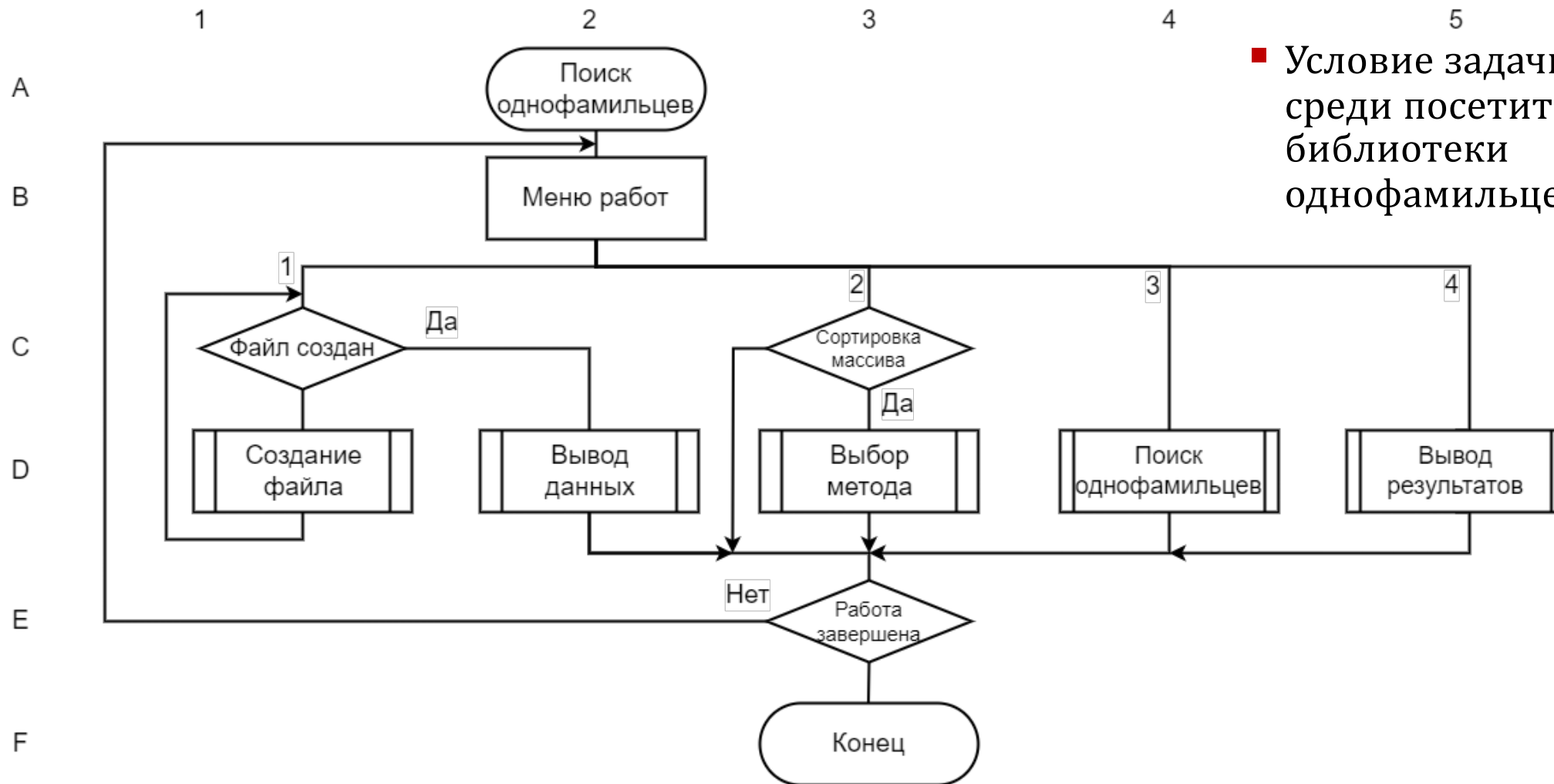
# Алгоритм «гномьей» сортировки



# Сравнительная таблица сходимости методов сортировки

Алгоритм	Лучшее значение	Среднее значение	Наихудший случай	Устойчивость
Простой обмен	$O(n)$	$O(n^2)$	$O(n^2)$	Да
Вставками	$O(n)$	$O(n^2)$	$O(n^2)$	Да
Выбором	$O(n)$	$O(n^2)$	$O(n^2)$	Да
Быстрая	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	Да / нет
Шелла	$O(n^{1.26})$	$O(n^{1.33})$	$O(n^{1.5})$	Нет
Слиянием	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Да
Поразрядная	$O(n)$	$O(n)$	$O(n)$	Да / нет

# Структура программы (пример)



- Условие задачи. Найти среди посетителей библиотеки однофамильцев



# Структуры данных (1)

- Структура – это составной объект, в который входят элементы любых типов, за исключением функций. В отличие от массива, который является однородным объектом, структура может быть неоднородной.

- Тип структуры определяется записью вида:

`struct { список определений };`

- В структуре обязательно должен быть указан хотя бы один компонент.

- Определение структур имеет следующий вид:

```
struct <Имя> {  
    Тип имя переменной;  
    Тип имя переменной; ← Поля данных  
    ...  
};
```

# Структуры данных (2)

```
typedef struct {  
    double x, y;  
} Point;  
...  
Point p1, p2, pm[10];
```

```
struct MyDate {  
    int year;  
    char month, day;  
};  
...  
struct MyDate data1, data2;
```

- Переменные p1, p2 определяются как структуры, каждая из которых состоит из двух полей: x и y.
- Доступ к полям структур: p1.x или p1.y
- Переменная pm определяется как массив из десяти структур.
- Каждая из двух переменных date1, date2 состоит из трех компонентов year, month, day.
- Имя - это имя типа структуры, а не имя переменной.
- Структурные переменные - это список имен переменных, разделенный запятыми.
- В блоках программы составное имя date1.year или date1.month и т.д. позволяет обрабатывать переменные разных типов.

# Структуры данных (3)

```
typedef struct data {  
    char *name;  
    int year;  
    int month, day;  
} data_1, data_2
```

В этом объявлении закладывается возможность использования только имени переменной в записи действий с компонентами структуры, без описателя `struct`:

```
data_1.year = 1976;  
data_2.month = 4;
```