

# Программирование на алгоритмическом языке. Часть III

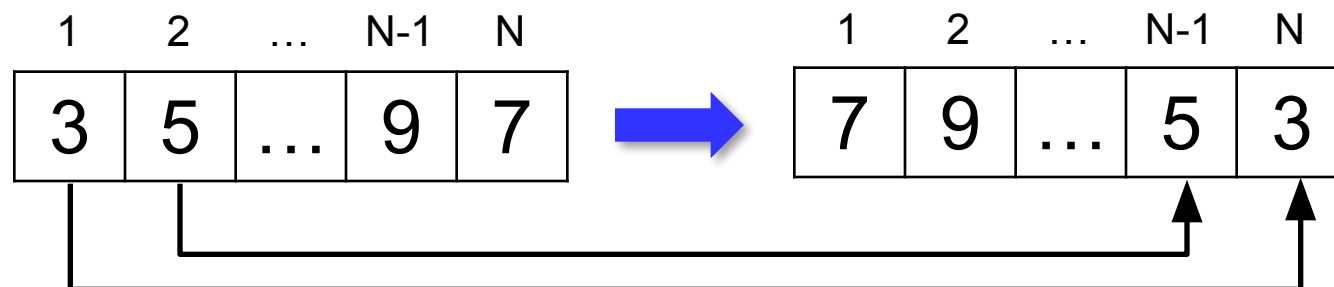
1. [Обработка массивов](#)
2. [Сортировка массивов](#)
3. [Двоичный поиск](#)
4. [Символьные строки](#)
5. [Матрицы](#)
6. [Файлы](#)

# Программирование на алгоритмическом языке. Часть II

## **Тема 1. Обработка массивов**

# Реверс массива

**Задача:** переставить элементы массива в обратном порядке.



**Алгоритм:**

поменять местами  $A[1]$  и  $A[N]$ ,  $A[2]$  и  $A[N-1]$ , ...

**Псевдокод:**

```

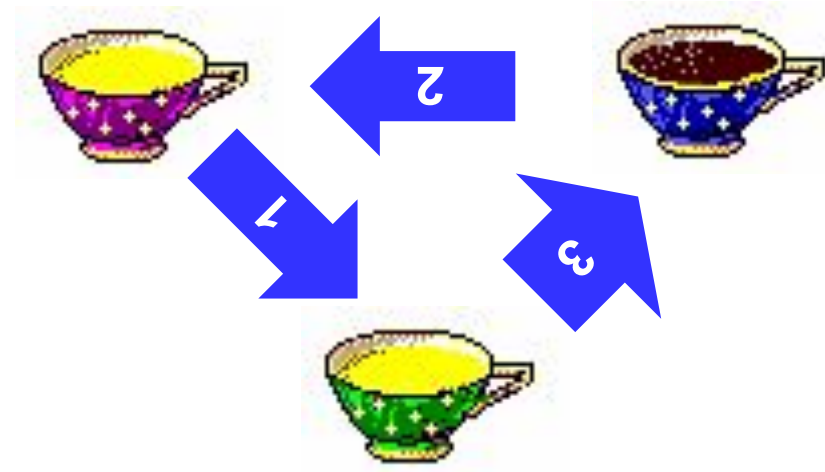
нц для  $i$  от 1 до  $\text{div}(N, 2)$ 
  | поменять местами  $A[i]$  и  $A[N+1-i]$ 
кц
  
```



Что неверно?

# Как переставить элементы?

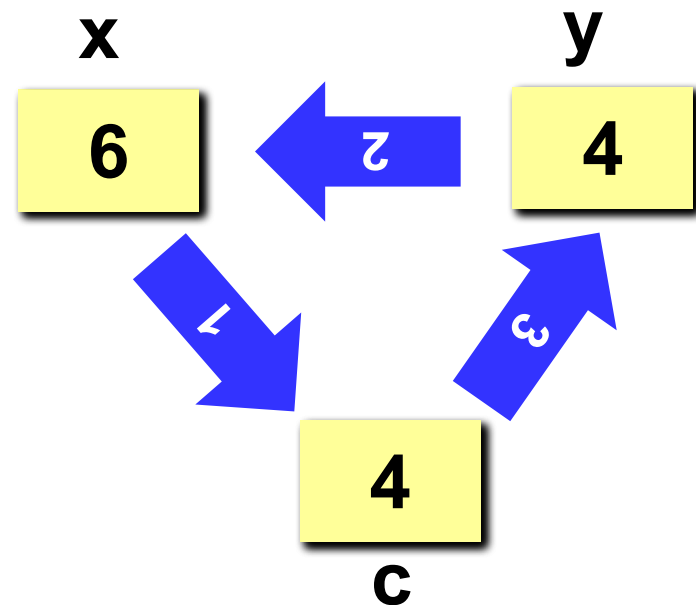
**Задача:** поменять местами содержимое двух чашек.



**Задача:** поменять местами содержимое двух ячеек памяти.

```
x := y
y := x
```

```
c := x
x := y
y := c
```



**?** Можно ли обойтись без **c**?

# Программа

алг Реверс

нач

цел  $i$ ,  $c$ ,  $N = 10$

целтаб  $A[1:N]$

| здесь нужно заполнить массив

нц для  $i$  от 1 до  $\text{div}(N, 2)$

$c := A[i]$

$A[i] := A[N+1-i]$

$A[N+1-i] := c;$

кц

| здесь нужно вывести результат

кон

# Задания

---

**«3»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-10..10]$  и сделать реверс всех элементов, кроме первого.

**Пример:**

Исходный массив:

4   -5   3   10   -4   -6   8   -10   1   0

Результат:

4   0   1   -10   8   -6   -4   10   3   -5

**«4»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-10..10]$  и сделать реверс отдельно для 1-ой и 2-ой половин массива.

**Пример:**

Исходный массив:

4   -5   3   10   -4   -6   8   -10   1   0

Результат:

-4   10   3   -5   4   |   0   1   -10   8   -6

# Задания

---

**«5»:** Заполнить массив из 12 элементов случайными числами в интервале  $[-12..12]$  и выполнить реверс для каждой трети массива.

**Пример:**

Исходный массив:

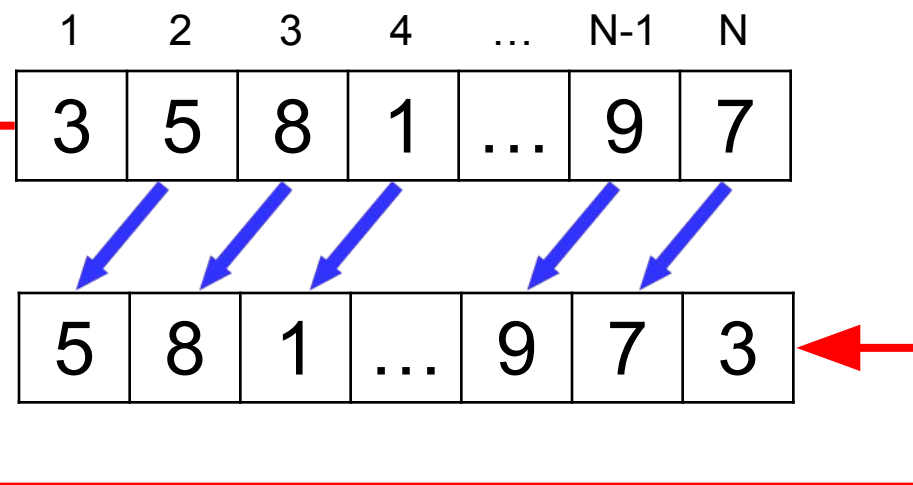
4    -5    3    10    |   -4    -6    8   -10    1    0    5    7

Результат:

10    3    -5    4    |   -10    8    -6    -4    7    5    0    1

# Циклический сдвиг

**Задача:** сдвинуть элементы массива влево на 1 ячейку, первый элемент становится на место последнего.



**Алгоритм:**

$A[1] := A[2] ; A[2] := A[3] ; \dots A[N-1] := A[N] ;$

```

нц для i от 1 до N-1
    A[i] := A[i+1]
кц
  
```

почему не N?



Что неверно?



# Программа

алг Циклический сдвиг влево

нач

цел  $i$ ,  $c$ ,  $N = 10$

целтаб  $A[1:N]$

| здесь нужно заполнить массив

```
c := A[1]
```

```
нц для  $i$  от 1 до  $N-1$ 
```

```
     $A[i] := A[i+1]$ 
```

```
кц
```

```
 $A[N] := c;$ 
```

| здесь нужно вывести результат

кон

# Задания

---

**«3»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-10..10]$  и выполнить циклический сдвиг влево *без первого элемента*.

**Пример:**

Исходный массив:

4   -5   3   10   -4   -6   8   -10   1   0

Результат:

4   0   -5   3   10   -4   -6   8   -10   1

**«4»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-10..10]$  и выполнить циклический сдвиг ВПРАВО.

**Пример:**

Исходный массив:

4   -5   3   10   -4   -6   8   -10   1   0

Результат:

0   4   -5   3   10   -4   -6   8   -10   1

# Задания

---

**«5»:** Заполнить массив из 12 элементов случайными числами в интервале  $[-12..12]$  и выполнить циклический сдвиг ВПРАВО на 4 элемента.

**Пример:**

Исходный массив:

4   -5   3   10   -4   -6   8   -10   |   1   0   5   7

Результат:

1   0   5   7   |   4   -5   3   10   -4   -6   8   -10

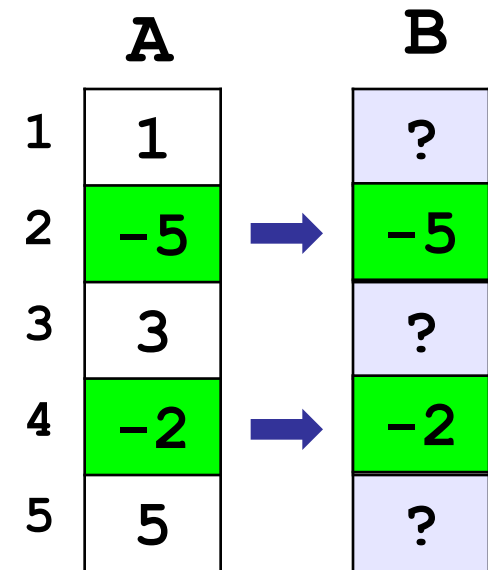
# Выбор нужных элементов

**Задача** – найти в массиве элементы, удовлетворяющие некоторому условию (например, отрицательные), и скопировать их в другой массив.

**Примитивное решение:**

```

цел  $i$ ,  $N = 5$ 
целтаб  $A[1:N]$ ,  $B[1:N]$ 
  | здесь заполнить массив A
нц для  $i$  от 1 до  $N$ 
  если  $A[i] < 0$  то
     $B[i] := A[i]$ 
  все
кц
  
```

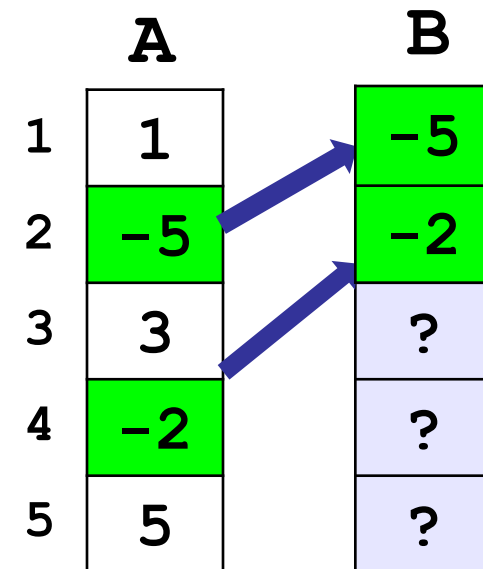


# Выбор нужных элементов

**Решение:** ввести счетчик найденных элементов `count`, очередной элемент ставится на место `B[count]`.

```

цел  $i$ ,  $N = 5$ ,  $count = 0$ 
целтаб  $A[1:N]$ ,  $B[1:N]$ 
  | здесь заполнить массив  $A$ 
нц для  $i$  от 1 до  $N$ 
  если  $A[i] < 0$  то
     $count := count + 1$ 
     $B[count] := A[i]$ 
  все
кц
  
```



# Как вывести массив В?

## Примитивное решение:

```
вывод "Выбранные элементы: ", нс  
нц для i от 1 до N  
    вывод В[i], " "  
кц
```



Что плохо?

## Правильное решение:

```
вывод "Выбранные элементы: ", нс  
нц для i от 1 до  t  
    вывод В[i], " "  
кц
```

# Задания

---

**«3»:** Заполнить массив случайными числами в интервале  $[-10, 10]$  и записать в другой массив все положительные числа.

**Пример:**

**Исходный массив :**

0   -5   3   7   -8

**Положительные числа :**

3   7

**«4»:** Заполнить массив случайными числами в интервале  $[20, 100]$  и записать в другой массив все числа, которые оканчиваются на 0.

**Пример:**

**Исходный массив :**

40   57   30   71   84

**Заканчиваются на 0 :**

40   30

# Задания

---

**«5»:** Заполнить массив случайными числами и выделить в другой массив все числа, которые встречаются более одного раза.

**Пример:**

**Исходный массив:**

4 1 2 1 11 2 34

**Результат:**

1 2



# Программирование на алгоритмическом языке. Часть II

## Тема 2. Сортировка массивов

# Сортировка

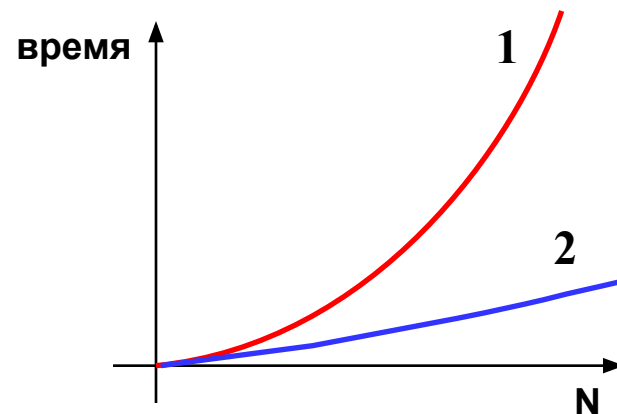
**Сортировка** – это расстановка элементов массива в заданном порядке (по возрастанию, убыванию, последней цифре, сумме делителей, ...).

**Задача:** переставить элементы массива в порядке возрастания.

сложность  $O(N^2)$

**Алгоритмы сортировки:**

- 1) простые и понятные, но медленно работающие для больших массивов
  - метод пузырька
  - метод выбора
- 2) сложные, но быстрые («быстрая сортировка» и др.)

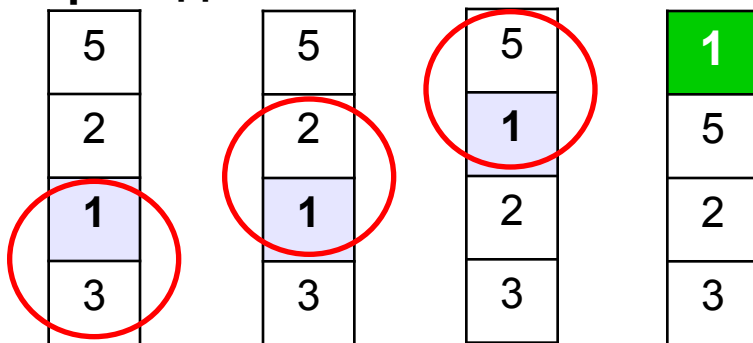


# Метод пузырька

**Идея** – пузырек воздуха в стакане воды поднимается со дна вверх.

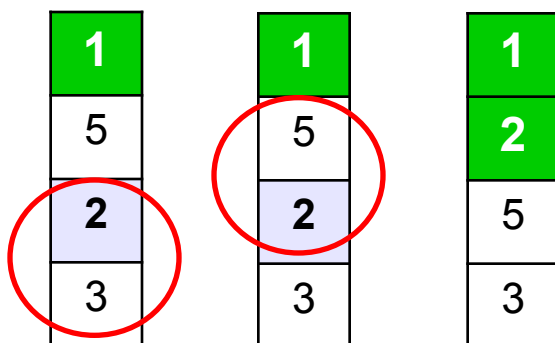
**Для массивов** – самый маленький («легкий» элемент перемещается вверх («всплывает»)).

**1-ый проход**

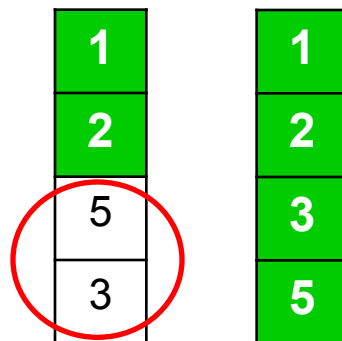


- начиная снизу, сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву **один** элемент (самый маленький) становится на свое место

**2-ой проход**



**3-ий проход**



Для сортировки массива из  $N$  элементов нужен  $N-1$  проход (достаточно поставить на свои места  $N-1$  элементов).

# Программа

$A[j]$  и  $A[j+1]$

1-ый проход:

1	5
2	2
...	...
N-1	6
N	3

сравниваются пары

$A[N-1]$  и  $A[N]$ ,  $A[N-2]$  и  $A[N-1]$ , ...,  $A[1]$  и  $A[2]$

нц для  $j$  от  $N-1$  до **1** шаг  $-1$

если  $A[j] > A[j+1]$  то

$c := A[j]; A[j] := A[j+1]; A[j+1] := c$

все

кц



$A[1]$  уже на своем месте!

2-ой проход

1	1
2	5
...	...
N-1	3
N	6

нц для  $j$  от  $N-1$  до **2** шаг  $-1$

если  $A[j] > A[j+1]$  то

$c := A[j]; A[j] := A[j+1]; A[j+1] := c$

все

кц

$i$ -ый проход

нц для  $j$  от  $N-1$  до  **$i$**  шаг  $-1$

...

# Программа

алг **Сортировка**

нач

цел  $N = 5$ ,  $i$ ,  $j$ ,  $c$

целтаб  $A[1:N]$

| здесь нужно заполнить массив

нц для  $i$  от  $1$  до  $N-1$

нц для  $j$  от  $N-1$  до  $i$  шаг  $-1$

если  $A[j] > A[j+1]$  то

$c := A[j]$ ;  $A[j] := A[j+1]$ ;  $A[j+1] := c$

все

кц

кц

| здесь нужно вывести полученный массив

кон



Почему цикл по  $i$  до  $N-1$ ?

элементы выше  $A[i]$   
уже поставлены

# Задания

---

**«3»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-10..10]$  и отсортировать его по убыванию.

**Пример:**

Исходный массив:

4 5 -8 3 -7 -5 3 1 0 9

Результат:

9 5 4 3 3 1 0 -5 -7 -8

**«4»:** Заполнить массив из 10 элементов случайными числами в интервале  $[0..100]$  и отсортировать его по последней цифре.

**Пример:**

Исходный массив:

14 25 13 30 76 58 32 11 41 97

Результат:

30 11 41 32 13 14 25 76 97 58

# Задания

---

**«5»:** Заполнить массив из 10 элементов случайными числами в интервале  $[0..100]$  и отсортировать первую половину по возрастанию, а вторую – по убыванию.

**Пример:**

Исходный массив:

14 25 13 30 76 | 58 32 11 41 97

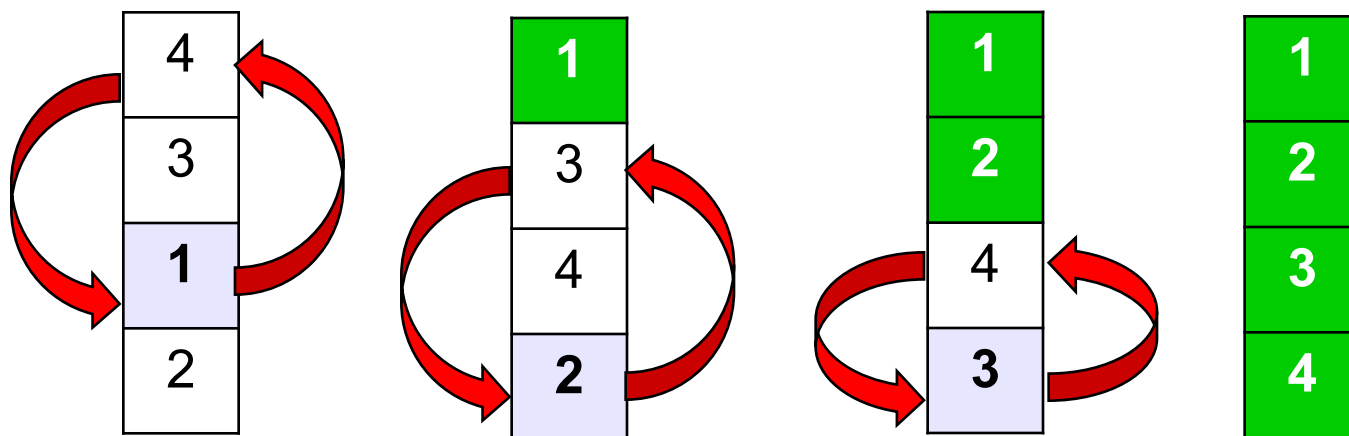
Результат:

13 14 25 30 76 | 97 58 41 32 11

# Метод выбора

## Идея:

- найти минимальный элемент и поставить на первое место (поменять местами с  $A[1]$ )
- **из оставшихся** найти минимальный элемент и поставить на второе место (поменять местами с  $A[2]$ ), и т.д.





# Метод выбора

нужно  $N-1$  проходов

нц для  $i$  от 1 до  $N-1$

$nMin := i$

нц для  $j$  от  $i+1$  до  $N$

если  $A[j] < A[nMin]$  то  $nMin := j$  все

кц

если  $nMin \neq i$  то

$c := A[i]$

$A[i] := A[nMin]$

$A[nMin] := c$

все

кц

ПОИСК МИНИМАЛЬНОГО  
ОТ  $A[i]$  ДО  $A[N]$

если нужно,  
переставляем



Можно ли убрать **если**?

# Задания

---

**«3»:** Заполнить массив из 10 элементов случайными числами в интервале [0..99] и отсортировать его по убыванию последней цифры.

**Пример:**

Исходный массив:

14 25 13 12 76 58 21 87 10 98

Результат:

98 58 87 76 25 14 13 12 21 10

**«4»:** Заполнить массив из 10 элементов случайными числами в интервале [0..99] и отсортировать его по возрастанию суммы цифр (*подсказка: их всего две*).

**Пример:**

Исходный массив:

14 25 13 12 76 58 21 87 10 98

Результат:

10 21 12 13 14 25 76 58 87 98

# Задания

---

**«5»:** Заполнить массив из 10 элементов случайными числами в интервале [0..100] и отсортировать первую половину по возрастанию, а вторую – по убыванию.

**Пример:**

Исходный массив:

14 25 13 30 76 58 32 11 41 97

Результат:

13 14 25 30 76 97 58 41 32 11

# «Быстрая сортировка» (*Quick Sort*)

**Идея** – более эффективно переставлять элементы, расположенные дальше друг от друга.

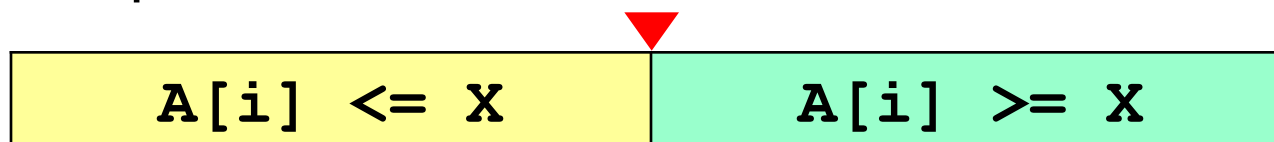


Сколько перестановок нужно, если массив отсортирован по убыванию, а надо – по возрастанию?

$$\text{div}(N, 2)$$

**1 шаг:** выбрать некоторый элемент массива  $X$

**2 шаг:** переставить элементы так:



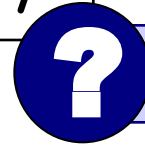
при сортировке элементы не покидают «свою область»!

**3 шаг:** так же отсортировать две получившиеся области

Разделяй и властвуй (англ. *divide and conquer*)

# «Быстрая сортировка» (Quick Sort)

78	6	82	67	55	44	34
----	---	----	----	----	----	----



Как лучше выбрать X?

**Медиана** – такое значение X, что слева и справа от него в отсортированном массиве стоит одинаковое число элементов (*для этого надо отсортировать массив...*).

## Разделение:

1) выбрать средний элемент массива ( $x=67$ )

78	6	82	67	55	44	34
----	---	----	----	----	----	----

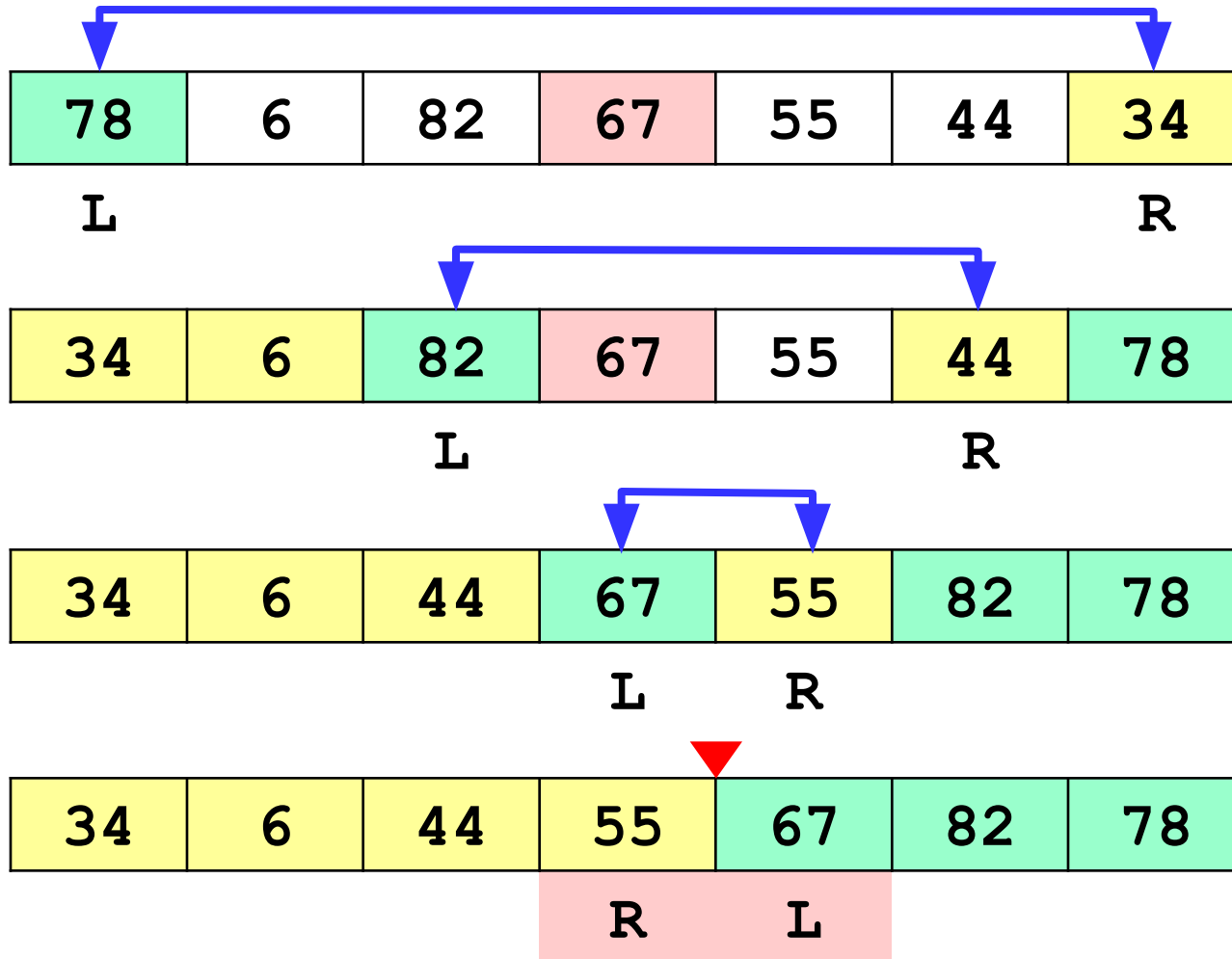
1) установить  $L := 1$ ,  $R := N$

2) увеличивая  $L$ , найти первый элемент  $A[L]$ , который  $\geq X$   
(должен стоять справа)

3) уменьшая  $R$ , найти первый элемент  $A[R]$ , который  $\leq X$   
(должен стоять слева)

4) если  $L \leq R$ , поменять местами  $A[L]$  и  $A[R]$  и перейти к п. 3

# «Быстрая сортировка» (Quick Sort)



**!**  $L > R$ : разделение закончено

# «Быстрая сортировка» (Quick Sort)

алг `qSort` (аргрез **целтаб** `A[1:5]`, арг **цел** `iStart`, `iEnd`)

нач

**цел** `L`, `R`, `c`, `X`

если `iStart >= iEnd` то выход все

`L := iStart`; `R := iEnd`;

`X := A[div(L+R, 2)]`

нц пока `L <= R`

нц пока `A[L] < X`; `L := L+1` кц

нц пока `A[R] > X`; `R := R-1` кц

если `L <= R` то

`c := A[L]`; `A[L] := A[R]`; `A[R] := c`

`L := L+1`; `R := R-1`

все

кц

`qSort` (`A`, `iStart`, `R`)

`qSort` (`A`, `L`, `iEnd`)

кон

ограничение рекурсии

разделение

обмен

двигаемся дальше

сортируем две части:  
рекурсия!

# «Быстрая сортировка» (Quick Sort)

```
алг Сортировка Quick Sort
```

```
нач
```

```
  цел N = 5, i
```

```
  целтаб A[1:N]
```

```
  | заполнить массив и вывести на экран
```

```
  qSort(A, 1, N) | сортировка
```

```
  | вывести отсортированный массив
```

```
кон
```

```
алг qSort ( , arg цел iStart, iEnd)
```

```
нач
```

```
...
```

```
кон
```

для массивов  
любого размера

```
  цел N, argрез целтаб A[1:N]
```


**Вызов:** qSort(N, A, 1, N)




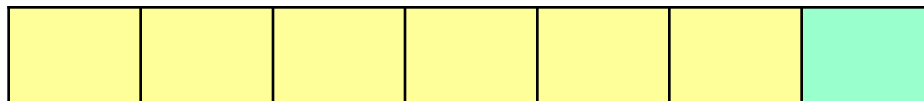
# Количество перестановок

(случайные данные)

$N$	<i>QuickSort</i>	«пузырек»
10	11	24
100	184	2263
200	426	9055
500	1346	63529
1000	3074	248547

 От чего зависит?

 Как хуже всего выбрать  $X$ ?



$O(N^2)$

## Задания

---

- «3»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-50..50]$  и отсортировать его с помощью алгоритма быстрой сортировки.
- «4»:** Заполнить массив из 10 элементов случайными числами в интервале  $[-50..50]$  и отсортировать его по убыванию с помощью алгоритма быстрой сортировки.
- «5»:** Заполнить массив из 500 элементов случайными числами в интервале  $[0..100]$ . Отсортировать его по возрастанию двумя способами – методом «пузырька» и методом «быстрой сортировки». Вывести на экран число перестановок элементов массива в том и в другом случае. Массив выводить на экран не нужно.

# Программирование на алгоритмическом языке. Часть II

## Тема 3. Двоичный поиск

# Поиск в массиве

---

**Задача** – найти в массиве элемент, равный **X**, или установить, что его нет.

**Решение:** для произвольного массива: **линейный поиск** (перебор)

недостаток: **низкая скорость**

**Как ускорить?** – заранее подготовить массив для поиска

- как именно подготовить?
- как использовать «подготовленный массив»?

# Линейный поиск

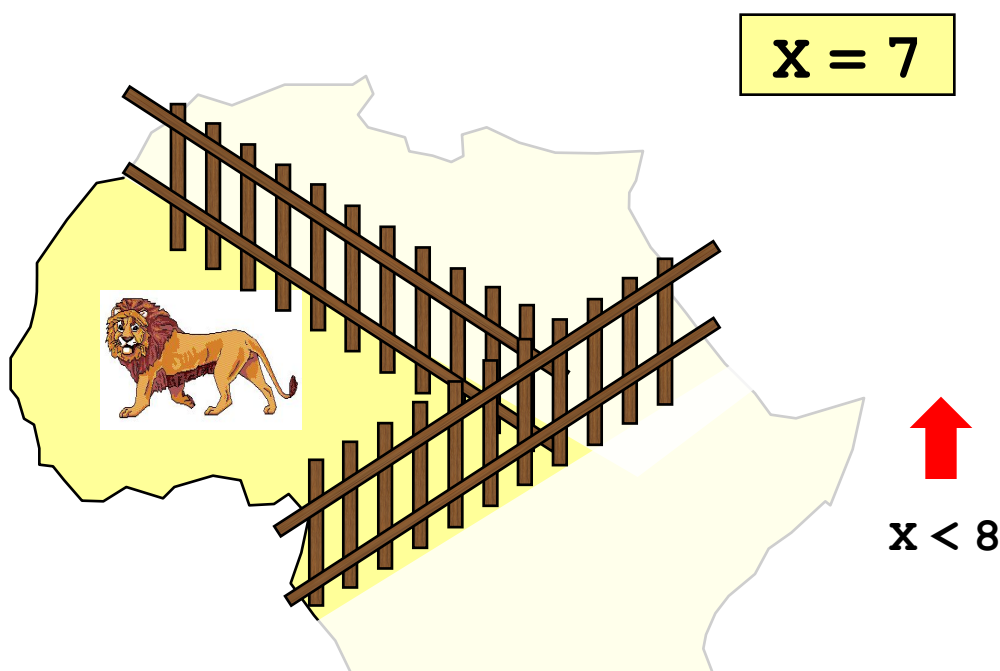
```
i := 1
нц пока i <= N и A[i] <> X
    i := i + 1
кц
если i <= N
    то вывод "A[" , i, "] = " , X
    иначе вывод "Нет такого"
все
```

$i$  – номер нужного  
элемента в массиве



Что плохо?

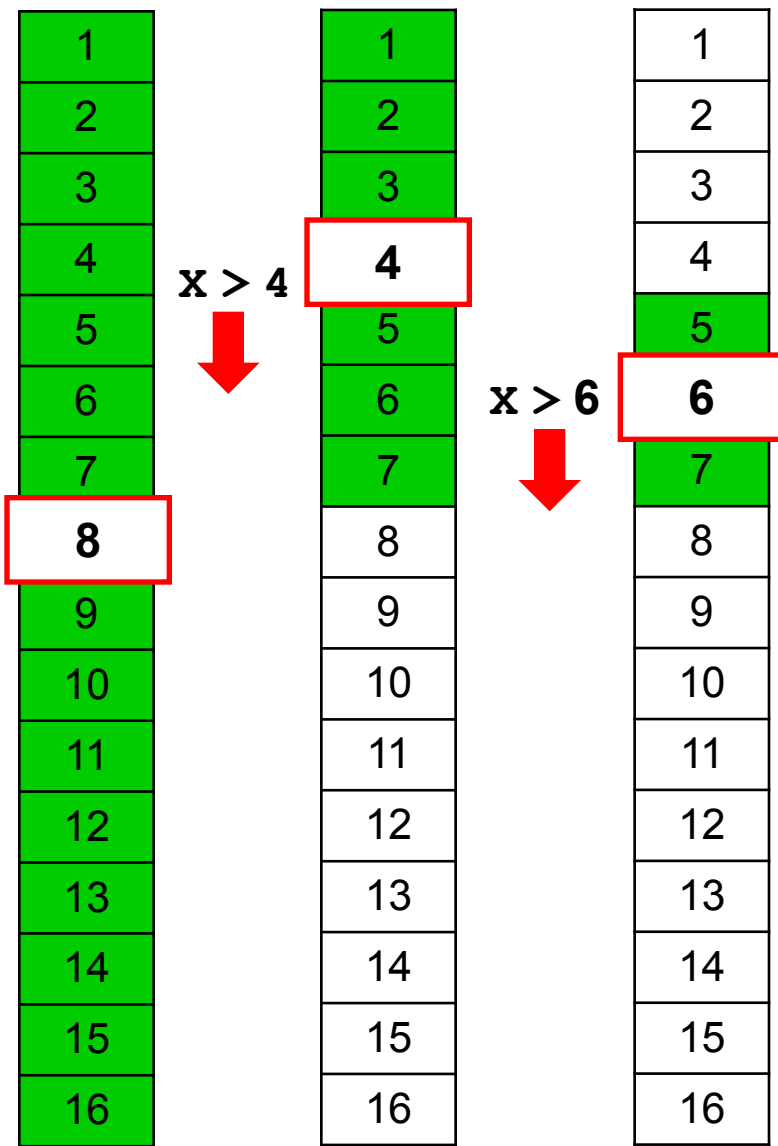
# Двоичный поиск



$X = 7$

$x < 8$

1. Выбрать средний элемент  $A[s]$  и сравнить с  $X$ .
2. Если  $X = A[s]$ , нашли (выход).
3. Если  $X < A[s]$ , искать дальше в первой половине.
4. Если  $X > A[s]$ , искать дальше во второй половине.



# Двоичный поиск



```
L := 1; R := N; nX := 0
```

```
нц пока R >= L
```

```
  c := div(R+L, 2);
```

```
  если X = A[c]
```

```
    то nX := c; выход
```

```
  все
```

```
  если X < A[c] то R := c - 1 все
```

```
  если X > A[c] то L := c + 1 все
```

```
кц
```

```
если nX > 0
```

```
  то вывод "A[" , nX, "]=", X
```

```
  иначе вывод "Не нашли"
```

```
все
```

номер среднего  
элемента

нашли

ВЫЙТИ ИЗ  
ЦИКЛА

сдвигаем  
границы



Почему нельзя **нц пока R > L ... кц ?**

# Сравнение методов поиска

	Линейный	Двоичный
подготовка	нет	<b>отсортировать</b>
	<b>число шагов</b>	
$N = 2$	2	2
$N = 16$	<b>16</b>	5
$N = 1024$	<b>1024</b>	11
$N = 1048576$	<b>1048576</b>	21
$N$	<b><math>\leq N</math></b>	$\leq \log_2 N + 1$



## Задания

---

- «3»:** Написать программу, которая сортирует массив по возрастанию и ищет в нем элемент, равный  $X$  (это число вводится с клавиатуры).  
Использовать двоичный поиск.
- «4»:** Написать программу, которая сортирует массив **ПО УБЫВАНИЮ** и ищет в нем элемент, равный  $X$  (это число вводится с клавиатуры).  
Использовать двоичный поиск.
- «5»:** Написать программу, которая считает среднее число шагов в двоичном поиске для массива из 32 элементов в интервале  $[0, 100]$ . Для поиска использовать 1000 случайных чисел в этом же интервале.

# Программирование на алгоритмическом языке. Часть II

## Тема 4. Символьные строки

## Задачи на обработку строк

---

**Задача:** с клавиатуры вводится число  $N$ , обозначающее количество футболистов команды «Шайба», а затем –  $N$  строк, в каждой из которых – информация об одном футболисте таком формате:

*<Фамилия> <Имя> <год рождения> <голы>*

Все данные разделяются одним пробелом. Нужно подсчитать, сколько футболистов, родившихся в период с 1988 по 1990 год, не забили мячей вообще.

**Алгоритм (для каждой строки):**

- 1) выделить год (`year`) и количество голов (`goal`)
- 2) если  $1988 \leq \text{year} \leq 1990$  и `goal = 0`,  
то увеличить счётчик

# Программа

использовать **Строки**

алг **Футболисты**

ГОД

ГОЛ

Ы

СЧЁТЧИК

нач

**цел** N, i, p, year, goal, count=0

**лит** s

**ввод** N

**нц** для i от 1 до N

**ввод** s

| **разобрать строку, выделить год и голы**

**если** year >= 1988 и year <= 1990 и goal = 0

**то** count:=count+1

**все**

**кц**

**вывод** count

**кон**

# Разбор строки

## Пропуск фамилии:

```
p := найти (" ", s)
s := s[p+1 : длин(s)]
```

Иванов <sup>p</sup> Вася 1998 5  
 Вася 1998 5

## Пропуск имени:

```
p := найти (" ", s)
s := s[p+1 : длин(s)]
```

Вася <sup>p</sup> 1998 5  
 1998 5

## Ввод года рождения:

```
p := найти (" ", s)
year := лит_в_цел(s[1:p-1], ОК)
s := s[p+1 : длин(s)]
```

ЛОГ ОК

<sup>p</sup>  
 1998 5  
1998 5  
 5

## Ввод числа голов:

```
goal := лит_в_цел(s, ОК)
```

# Разбор строки

## Если фамилия нужна:

**лит** fam

```
p := найти(" ", s)
fam := s[1:p-1]
s := s[p+1:длин(s)]
```

Иванов Вася 1998 5  
 Иванов  
 Вася 1998 5

## Если нужны ВСЕ фамилии:

**литтаб** fam[1:N]

```
p := найти(" ", s)
fam[i] := s[1:p-1]
s := s[p+1:длин(s)]
```

# Задания

---

Информация о футболистах вводится так же, как и для приведенной задачи (сначала N, потом N строк с данными).

«3»: Вывести фамилии всех футболистов, которые забили больше двух голов.

Пример:

Иванов Василий

Семёнов Кузьма

«4»: Вывести фамилию и имя футболиста, забившего наибольшее число голов, и количество забитых им голов.

Пример:

Иванов Василий 25

# Задания

---

**«5»:** Вывести в алфавитном порядке фамилии и имена всех футболистов, которые забили хотя бы один гол. В списке не более 100 футболистов.

**Пример:**

**Васильев Иван**

**Иванов Василий**

**Кутузов Михаил**

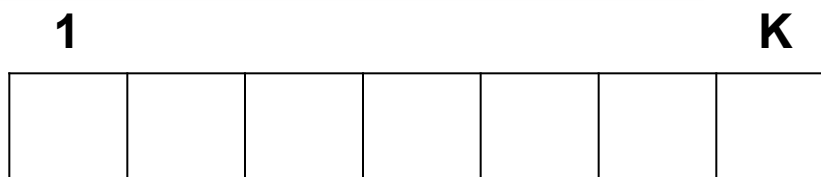
**Пупкин Василий**



# Рекурсивный перебор

**Задача:** Алфавит языка племени «тумба-юмба» состоит из букв **Ы**, **Ц**, **Щ** и **О**. Вывести на экран все слова из **K** букв, которые можно составить в этом языке, и подсчитать их количество. Число **K** вводится с клавиатуры.

в каждой ячейке может быть любая из 4-х букв



4 вари

4 варианта

4 варианта

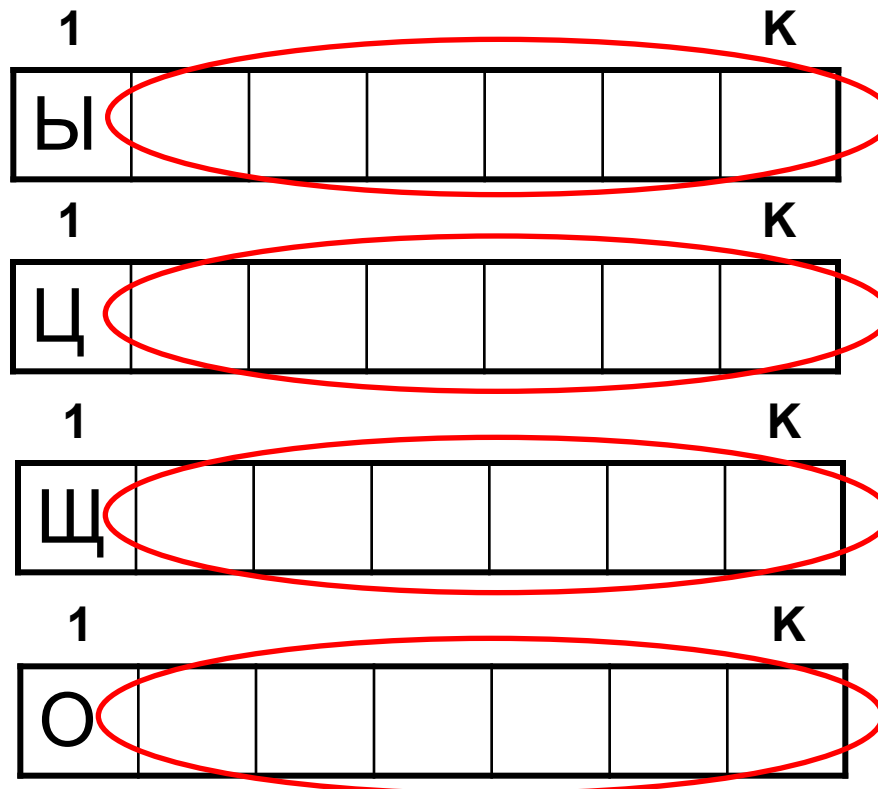
4 варианта

**Количество вариантов:**

$$N = 4 \cdot 4 \cdot 4 \cdot \square \cdot 4 = 4^K$$

# Рекурсивный перебор

**Рекурсия:** Решения задачи для слов из **K** букв сводится к 4-м задачам для слов из **K-1** букв.



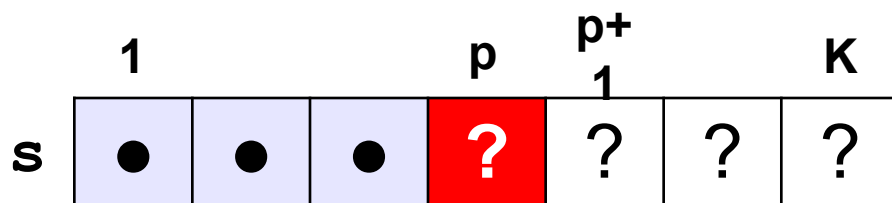
перебрать все варианты

перебрать все варианты

перебрать все варианты

перебрать все варианты

# Процедура



Глобальные переменные:

**лит** s

**цел** count, K

**алг** Рек (**цел** p)

**нач**

**если**  $p > K$  **то**

**вывод** s, **нс**

    count := count + 1

**выход**

**все**

s[p] := "Ы" ; Рек (p+1)

s[p] := "Ц" ; Рек (p+1)

s[p] := "Щ" ; Рек (p+1)

s[p] := "О" ; Рек (p+1)

**кон**

окончание рекурсии

рекурсивные вызовы



А если букв много?

# Основная программа

```
лит s, цел count = 0, K
алг Рекурсивный перебор
нач
```

глобальные переменные

```
  вывод "Введите длину слов: "
```

```
  ввод K
```

```
  s := ""
```

```
  нц K раз s := s + " " кц
```

строка из K пробелов

```
  Рек(1)
```

```
  вывод "Всего ", count, " слов"
```

```
кон
```

```
алг Рек(цел p)
```

```
нач
```

```
  ...
```

```
кон
```

# Процедура (много букв)

алг **Рек** (**цел** p) **все** буквы

локальные переменные

нач

**лит** syms="ЫЩЦО", **цел** i

**если** p > K **то**

**вывод** s, **нс**

count := count + 1

**выход**

цикл по всем буквам

**все**

**нц** для i от 1 до **длин**(syms)

s[p] := syms[i]; **Рек**(p+1)

**кц**

**кон**

## Задания

---

Алфавит языка племени «тумба-юмба» состоит из букв **Ы**, **Ц**, **Щ** и **О**. Число **К** вводится с клавиатуры.

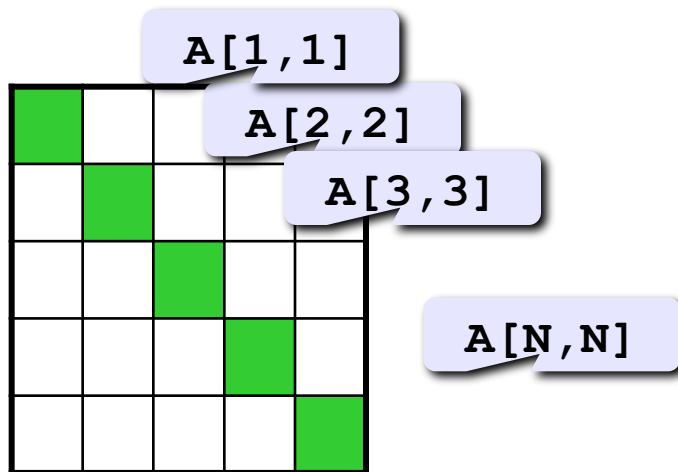
- «3»: Вывести на экран все слова из **К** букв, в которых первая буква – **Ы**, и подсчитать их количество.
- «4»: Вывести на экран все слова из **К** букв, в которых буква **Ы** встречается более 1 раза, и подсчитать их количество.
- «5»: Вывести на экран все слова из **К** букв, в которых есть одинаковые буквы, стоящие рядом (например, **ЫЩЩО**), и подсчитать их количество.

# Программирование на алгоритмическом языке. Часть II

## Тема 5. Матрицы

# Операции с матрицами

**Задача 1.** Вывести на экран главную диагональ квадратной матрицы из  $N$  строк и  $N$  столбцов.

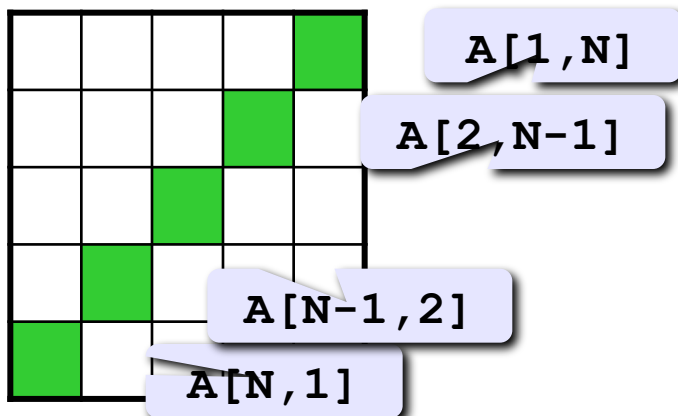


```
нц для i от 1 до N
  вывод A[i,i], " "
кц
```



А снизу вверх?

**Задача 2.** Вывести на экран вторую диагональ.



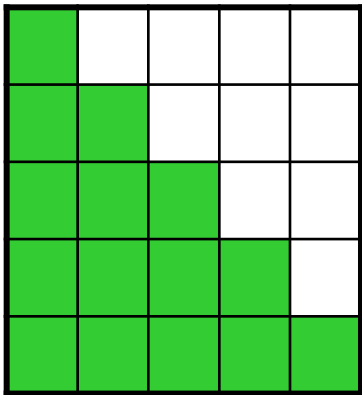
сумма номеров строки и столбца  $N+1$

```
нц для i от 1 до N
  вывод A[i, N+1-i], " "
кц
```



# Операции с матрицами

**Задача 3.** Найти сумму элементов, стоящих на главной диагонали и ниже ее.



Одиночный цикл или вложенный?

строка 1:  $A[1, 1]$

строка 2:  $A[2, 1] + A[2, 2]$

...

строка N:  $A[N, 1] + A[N, 2] + \dots + A[N, N]$

```
S := 0;
```

```
нц для i от 1 до N
```

```
нц для j от 1 до i
```

```
    S := S + A[i, j]
```

```
кц
```

```
кц
```

цикл по всем строкам

складываем нужные  
элементы строки i

# Операции с матрицами

**Задача 4.** Перестановка строк или столбцов. В матрице из  $N$  строк и  $M$  столбцов переставить 2-ую и 4-ую строки.

	j				
2	1	2	5	2	1
4	7	3	1	3	7

Callouts:  $A[2, j]$  (pointing to row 2) and  $A[4, j]$  (pointing to row 4). Red double-headed arrows indicate the swap of elements between rows 2 and 4 in each column.

нц для j от 1 до M

$c := A[2, j]$

$A[2, j] := A[4, j]$

$A[4, j] := c$

кц

**Задача 5.** К третьему столбцу добавить шестой.

нц для i от 1 до N

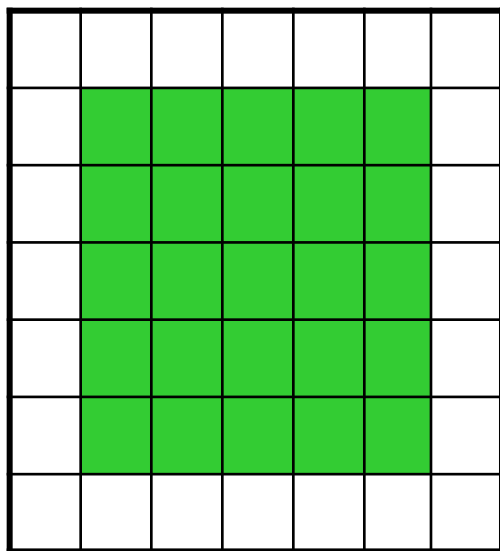
$A[i, 3] := A[i, 3] + A[i, 6]$

кц

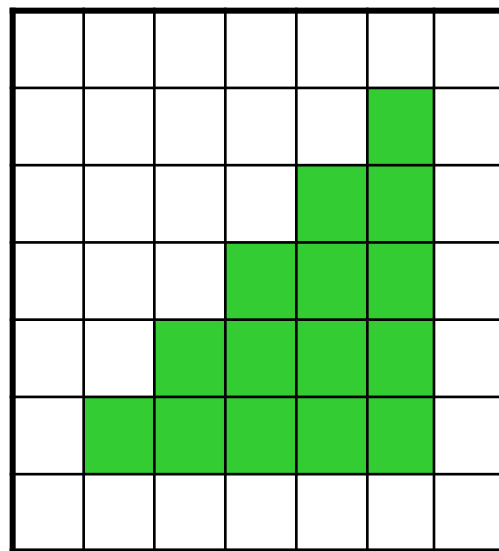
# Задания

Заполнить матрицу из 7 строк и 7 столбцов случайными числами в интервале  $[10,90]$  и вывести ее на экран. Заполнить элементы, отмеченные зеленым фоном, числами 99, и вывести полученную матрицу на экран.

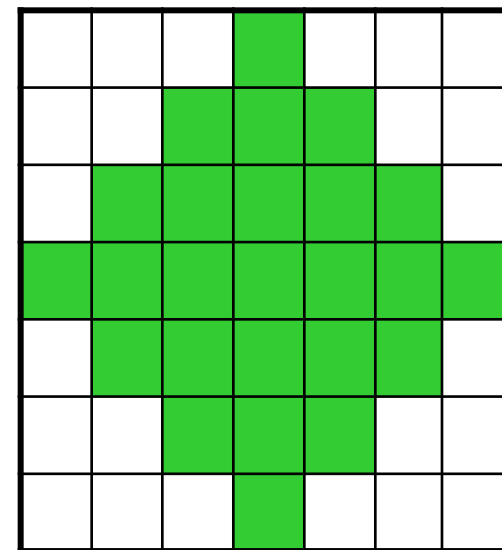
«3»:



«4»:



«5»:



# Программирование на алгоритмическом языке. Часть II

## Тема 6. **Файлы**

# Файлы

**Файл** – это данные на диске, имеющие имя.

**Файл**

**ы**

**Текстовы**

**е**

только текст без оформления,  
не содержат управляющих  
символов (с кодами < 32)

ASCII (1 байт на символ)

UNICODE (>1 байта на символ)

\*.txt, \*.log,

\*.htm, \*.html

**Двоичные**

могут содержать любые  
символы кодовой таблицы

\*.doc, \*.exe,

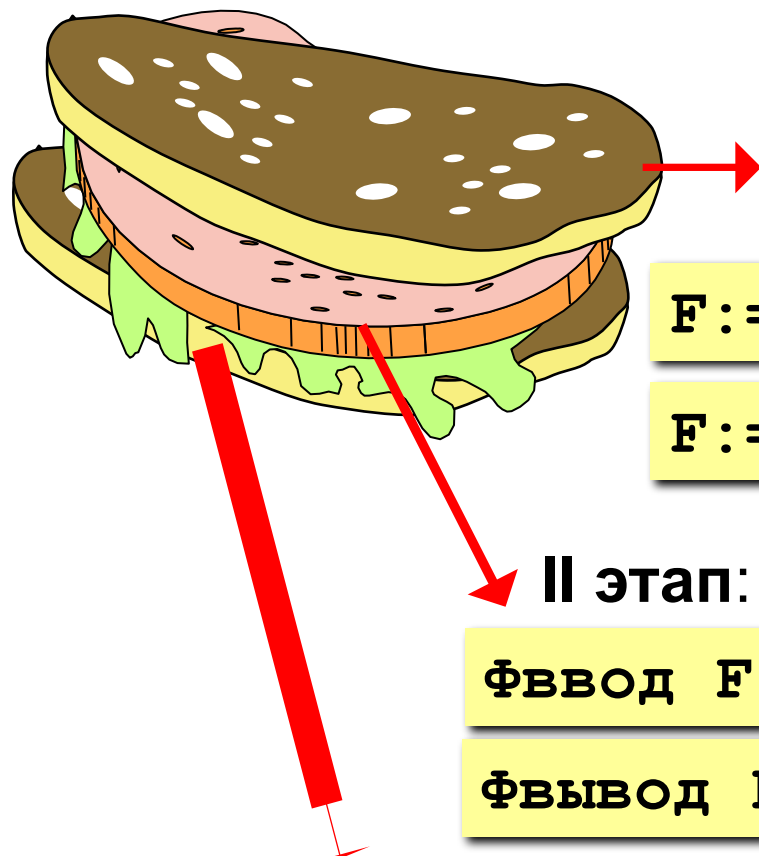
\*.bmp, \*.jpg,

\*.wav, \*.mp3,

\*.avi, \*.mpg

**Каталоги  
(папки)**

# Работа с файлами: принцип сэндвича



I этап. открыть файл:

- сделать его активным, приготовить к работе
- связать переменную **F** с

цел **F**

**F := открыть на чтение ("in.txt")**

**F := открыть на запись ("out.txt")**

II этап: работа с файлом

**ФВВОД F, a, b | ввести a и b**

**ФВВЫВОД F, a, b, nc | вывести a и b**

III этап: закрыть файл

**закреть (F)**

# Работа с файлами

---

## Особенности:

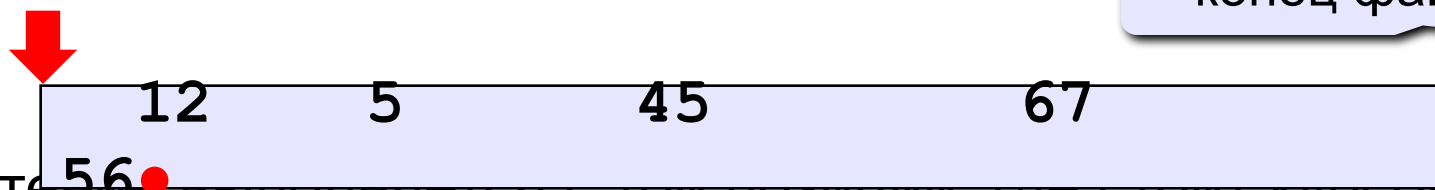
- имя файла упоминается только при открытии файла, работа с файлом – через файловую переменную
- файл, который открывается на чтение, должен **существовать**
- если файл, который открывается на запись, существует, старое содержимое **уничтожается**
- данные записываются в файл в текстовом виде
- при завершении программы все файлы закрываются автоматически
- после закрытия файла переменную **F** можно использовать еще раз для работы с другим файлом

# Последовательный доступ

- при открытии файла курсор устанавливается в начало

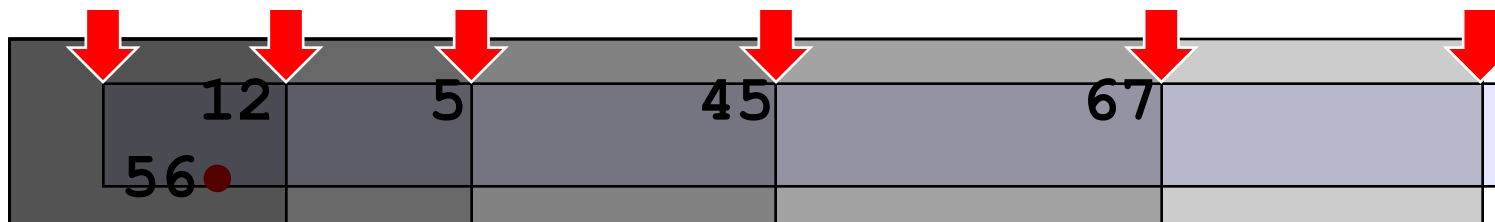
```
F := открыть на чтение ("qq.txt")
```

конец файла



- чтение выполняется с той позиции, где стоит курсор
- после чтения курсор сдвигается на первый непрочитанный СИМВОЛ

```
ФВВОД F, x
```





# Последовательный доступ

---

- как вернуться назад и начать с начала?

```
закреть ( F )
```

```
F := открыть на чтение ( "qq.txt" )
```

- не открывая файл заново

```
начать чтение ( F )
```

# Пример

**Задача:** в файле `input.txt` записаны числа (в столбик), сколько их – неизвестно. Записать в файл `output.txt` их сумму.

**Алгоритм:**



Можно ли обойтись без массива?

1. Открыть файл `input.txt` для чтения.
2.  $S := 0$
3. Если чисел не осталось, перейти к шагу 7.
4. Прочитать очередное число в переменную  $x$ .
5.  $S := S + x$
6. Перейти к шагу 3.
7. Закрыть файл `input.txt`.
8. Открыть файл `output.txt` для записи.
9. Записать в файл значение  $S$ .
10. Закрыть файл `output.txt`.

**ЦИКЛ «ПОКА ЕСТЬ  
ДААННЫЕ»**

# Программа: чтение данных из файла

использовать **Файлы П**

алг **Сумма чисел**

нач

**цел** F, S, x

F := **открыть на чтение** ("input.txt")

S := 0

нц пока не **конец файла** ( F )

**ФВвод** F, x

    S := S + x

кц

**закреть** ( F )

| **здесь нужно записать результат в файл**

кон

логическая функция,  
возвращает **да**, если  
достигнут конец файла

# Программа: запись результата в файл

---

## Особенности:

- файл, который открывается на запись, **не обязательно** должен **существовать**
- старое содержимое файла **уничтожается**

```
F := открыть на запись ("output.txt")  
Фвывод F, S  
закреть ( F )
```

## Задания

---

В файле `input.txt` записаны числа, сколько их – неизвестно.

- «3»: Найти сумму всех чётных чисел и записать её в файл `output.txt`.
- «4»: Найти минимальное и максимальное из всех чисел и записать их в файл `output.txt`.
- «5»: Найти длину самой длинной цепочки одинаковых чисел, идущих подряд, и записать её в файл `output.txt`.

# Обработка массивов

---

**Задача:** в файле `input.txt` записаны числа (в столбик), сколько их – неизвестно, но не более 100. Переставить их в порядке возрастания и записать в файл `output.txt`.



Можно ли обойтись без массива?

## Проблемы:

1. для сортировки надо удерживать в памяти все числа сразу (нужен массив!);
2. сколько чисел – неизвестно.

## Решение:

3. выделяем в памяти массив из 100 элементов;
4. записываем прочитанные числа в массив и считаем их с помощью переменной **N**;
5. сортируем первые **N** элементов массива;
6. записываем их в файл.

# Программа: чтение данных в массив

использовать **файлы П**

алг **Сортировка**

нач

**цел** F, MAX = 100, N = 0, i, j, c

**целтаб** A[1:MAX]

F := **открыть на чтение** ("input.txt")

**нц** пока не **конец файла** (F) и  $N < \text{MAX}$

    N := N + 1

**ФВвод** F, A[N]

**кц**

**закреть** (F)

| **отсортировать** первые N элементов

| **вывести** полученный массив

**кон**

цикл заканчивается, если достигнут конец файла или прочитали 100 чисел

## Программа: вывод массива в файл

```
F := открыть на запись ("output.txt")
```

```
нц для i от 1 до N
```

```
    Фвывод F, A[i], нс
```

```
кц
```

```
закреть (F)
```

зачем?



## Задания

---

В файле `input.txt` записаны числа (в столбик), известно, что их не более 100.

- «3»: Отсортировать массив по убыванию и записать его в файл `output.txt`.
- «4»: Отсортировать массив по убыванию последней цифры и записать его в файл `output.txt`.
- «5»: Отсортировать массив по возрастанию суммы цифр и записать его в файл `output.txt`.

# Обработка текстовых данных

---

**Задача:** в файле `input.txt` записаны строки, в которых есть слово-паразит «*короче*». Очистить текст от мусора и записать в файл `output.txt`.

**Файл `input.txt` :**

Мама, короче, мыла, короче, раму.

Декан, короче, пропил, короче, бутан.

А роза, короче, упала на лапу, короче, Азора.

Каждый, короче, охотник желает, короче, знать, где ...

**Результат - файл `output.txt` :**

Мама мыла раму.

Декан пропил бутан.

А роза упала на лапу Азора.

Каждый охотник желает знать, где сидит фазан.

# Обработка текстовых данных

---

## Алгоритм:

1. Прочитать строку из файла.
2. Удалить все сочетания ", *короче*".
3. Записать строку в другой файл.
4. Перейти к шагу 1.

пока не  
кончились  
данные

## Особенность:

**надо одновременно держать открытыми два файла: один в режиме чтения, второй – в режиме записи.**

# Работа с двумя файлами одновременно

```
использовать Строки
использовать Файлы П
алг Обработка строк
нач
    цел fIn, fOut
    fIn := открыть на чтение ("input.txt")
    fOut := открыть на запись ("output.txt")
        | обработка файла
    закрыть (fIn)
    закрыть (fOut)
кон
```

# Цикл обработки файла

---

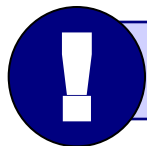
```
нц пока не конец файла (fIn)
```

```
    ФВвод fIn, s
```

```
        | обработка строки s
```

```
    ФВывод fOut, s, нс
```

```
кц
```



Оба файла открыты одновременно!

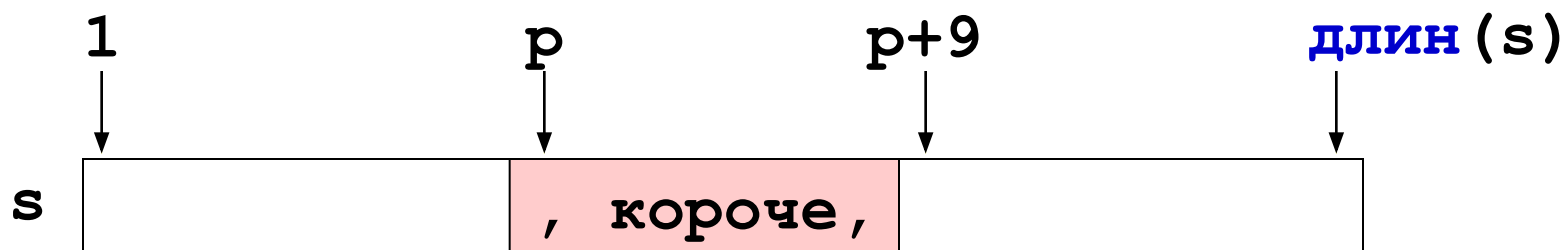
# Обработка одной строки

бесконечный цикл

```
нц пока да  
  р := найти (", короче,", s)  
  если р < 1 то выход все  
  s := s[1:р-1] + s[р+9:длин(s)]  
кц
```

ВЫХОД ИЗ  
ЦИКЛА

удалить 9 символов



## Задания

---

В файле `input.txt` записаны строки, сколько их – неизвестно.

- «3»: Заменить все слова «короче» на «в общем» и записать результат в файл `output.txt`.
- «4»: Вывести в файл `output.txt` только те строки, в которых есть слово «пароход». В этих строках заменить все слова «короче» на «в общем».
- «5»: Вывести в файл `output.txt` только те строки, в которых больше 5 слов (слова могут быть разделены несколькими пробелами).

# Сортировка списков

---

**Задача:** в файле `list.txt` записаны фамилии и имена пользователей сайта (не более 100). Вывести их в алфавитном порядке в файл `sort.txt`.

**Файл `list.txt` :**

Федоров Иван

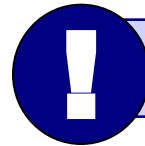
Иванов Федор

Анисимов Никита

Никитин Николай



Нужен ли массив!



Для сортировки нужен массив!

**Результат – файл `sort.txt` :**

Анисимов Никита

Иванов Федор

Никитин Николай

Федоров Иван



# Сортировка списков

---

## Алгоритм:

- 1) прочитать строки из файла в массив строк, подсчитать их в переменной **N**
- 2) отсортировать первые **N** строк массива по алфавиту
- 3) вывести первые **N** строк массива в файл

## Объявление массива (с запасом):

```
цел МАХ = 100
```

```
литтаб s [1:МАХ]
```

# Сортировка списков

## Ввод массива строк из файла:

```
f := открыть на чтение ("list.txt")
```

цел f, N

```
N := 0
```

```
нц пока не конец файла (f)
```

```
    N := N + 1
```

```
    ФВВОД f, s[N]
```

```
кц
```

```
закреть (f)
```

## Вывод первых N строк массива в файл:

```
f := открыть на запись ("sort.txt")
```

```
нц для i от 1 до N
```

```
    ФВВВОД f, s[i], нс
```

```
кц
```

```
закреть (f)
```

# Сортировка списков

## Сортировка первых N элементов массива:

```
нц для i от 1 до N-1
  nMin := i
  нц для j от i+1 до N
    если s[j] < s[nMin] то nMin := j все
  кц
  если i <> nMin то
    c := s[i]
    s[i] := s[nMin]
    s[nMin] := c
  все
кц
```

цел i, j, nMin  
лит c



Какой метод?

# Сортировка списков

Как сравниваются строки:

	245							
<b>s1</b>	П	а	р	о	х	о	д	ѐ
					?			
<b>s2</b>	П	а	р	о	в	о	з	ѐ
	226							



Что больше?

Кодовая таблица:

	А	Б	В	...	Я	а	б	в	...	х	...	я
<b>Win</b>	192	193	194	...	223	224	225	226	...	245	...	255
<b>UNICODE</b>	1040	1041	1042	...	1071	1072	1073	1074	...	1093	...	1103

**код("х") > код("в")**

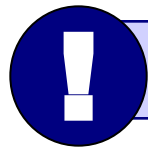
**"х" > "в"**

**"Паро**х**од" > "Паро**в**оз"**

# Сортировка списков

## Как сравниваются строки:

s1	П	а	р	о	х	о	Д	␣
				?				
s2	П	а	р	␣				



Любой символ больше пустого!

"х" > ␣

"Пароход" > "Пар"

# Сортировка списков

---

## Работа с отдельной строкой массива:

```
литтаб s[1:MAX]
лит с | вспомогательная строка
нц для i от 1 до N
    с := s[i]
    | работаем со строкой с, меняем ее
    s[i] := с
кц
```

# Задания

---

**«3»:** Добавить к списку нумерацию:

- 1) Анисимов Никита
- 2) Иванов Федор

**«4»:** Выполнить задачу на «3» и сократить имя до первой буквы:

- 1) Анисимов Н.
- 2) Иванов Ф.

**«5»:** Выполнить задачу на «4», но при выводе начинать с имени:

- 1) Н. Анисимов
- 2) Ф. Иванов

# Списки с числовыми данными

---

**Задача:** в файле `marks.txt` записаны фамилии и имена школьников и баллы, полученные ими на экзамене (0-100). В файле не более 100 строк. Вывести в файл `best.txt` список тех, кто получил более 75 баллов.

**Файл `marks.txt` :**

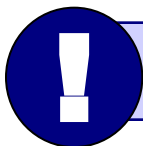
Федоров Иван 78  
Иванов Федор 63  
Анисимов Никита 90  
Никитин Николай 55



Нужен ли массив!

**Результат – файл `best.txt` :**

Федоров Иван 78  
Анисимов Никита 90



Используем два файла одновременно!



# Работа с двумя файлами одновременно

использовать **Файлы П**

использовать **Строки**

алг **Отметки на экзамене**

нач

**цел** fIn, fOut

fIn := **открыть на чтение** ("marks.txt")

fOut := **открыть на запись** ("best.txt")

| **обработка строк из файла**

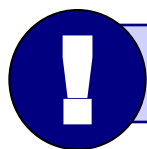
**закреть** (fIn)

**закреть** (fOut)

кон

# Цикл обработки файла

```
цел ball
нц пока не конец файла (fIn)
    Фввод fIn, s
    | обработка строки s
    | ball := результат на экзамене
если ball > 75 то
    Фвывод fOut, s, нс
все
кц
```



Оба файла открыты одновременно!

# Преобразования «строка»-«число»

## Из строки в число:

```
s := "123"
```

да или нет

```
N := лит_в_цел(s, ОК) | N = 123
```

```
если не ОК то вывод "Ошибка!" все
```

```
s := "123.456";
```

```
X := лит_в_вещ(s, ОК) | X = 123.456
```

```
если не ОК то вывод "Ошибка!" все
```

цел N, вещ X,  
лит s, лог ОК

## Из числа в строку:

```
N := 123
```

```
s := цел_в_лит(N) | "123"
```

```
X := 123.456
```

```
s := вещ_в_лит(X) | "123.456"
```

# Обработка строки

```
цел n
лит s, фамилия, имя
лог ОК
```

s:

8	2
---	---

```
n := найти (" ", s) | n := 7
фамилия := s [1:n-1] | фамилия := "Пупкин"
s := удалить (s, 1, n) | s := "Вася 82"
n := найти (" ", s) | n := 5
имя := s [1:n-1] | имя := "Вася"
s := удалить (s, 1, n) | s := "82"
ball := лит_в_цел (s, ОК) | ball := 82
```

# Обработка строки

```
n := найти (" ", s)      | n := 7
фамилия := s [1:n-1]     | фамилия := "Пупкин"
s := удалить (s, 1, n)   | s := "Вася 82"
n := найти (" ", s)      | n := 5
имя := s [1:n-1]         | имя := "Вася"
s := удалить (s, 1, n)   | s := "82"
ball := лит_в_цел (s, ОК) | ball := 82
если ball > 75 то
    ФВывод fOut, s, нс
все
```



Что плохо?

# Задания

---

**«3»:** Добавить к списку нумерацию:

- 1) Федоров Иван 78
- 2) Анисимов Никита 90

**«4»:** Выполнить задачу на «3» и сократить имя до первой буквы:

- 1) Федоров И. 78
- 2) Анисимов Н. 90

**«5»:** Выполнить задачу на «4», но отсортировать список по алфавиту.

- 1) Анисимов Н. 90
- 2) Федоров И. 78

**«6»:** Выполнить задачу на «4», но отсортировать список по убыванию отметки (балла).

# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**  
**д.т.н., учитель информатики высшей**  
**категории,**  
**ГОУ СОШ № 163, г. Санкт-Петербург**  
**[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)**