# Программирование на языке Паскаль

# Программирование на языке Паскаль

## § 62. Массивы

# Что такое массив?


Как ввести 10000 переменных?

**Массив** – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка в массиве имеет уникальный номер.

**Надо**:

- выделять память
- записывать данные в нужную ячейку
- читать данные из ячейки

# Выделение памяти (объявление)

минимальный индекс

максимальный индекс

```
var A: array[1..5] of integer;
  V: array[0..5] of real;
  L: array[-5..5] of boolean;
  S: array[65..90] of char;
```

размер через константу

```
const N = 10;
var A: array[1..N] of integer;
```

? Зачем?

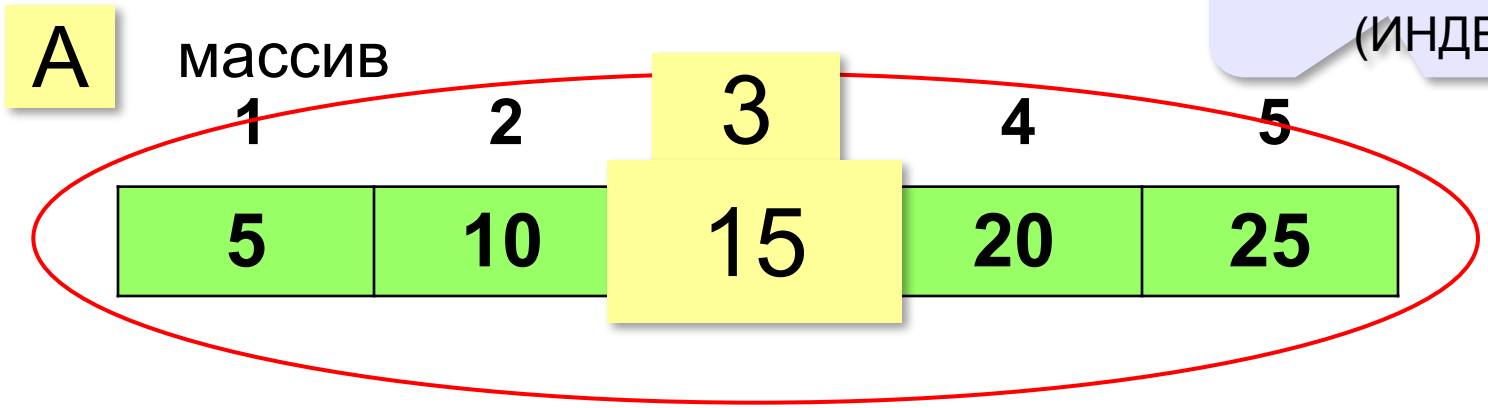
# Что неправильно?

```
var A: array[1..10] of integer;
...
A[5] := 4.5;
```

```
var A: array[1..10] of integer;
...
A[15] := 'a'
```

```
var a: array['a'..'z'] of integer;
...
  A['b'] := 15;
```

# Обращение к элементу массива




НОМЕР элемента массива (ИНДЕКС)

A[1] A[2] A[5]

ЗНАЧЕНИЕ элемента массива

НОМЕР (ИНДЕКС) элемента массива: 2

A[2]

ЗНАЧЕНИЕ элемента массива: 10

# Как обработать все элементы массива?

**Объявление**:

```
const N = 5;
var A: array[1..N] of integer;
```

**Обработка**:

```
{ обработать A[1] }
{ обработать A[2] }
{ обработать A[3] }
{ обработать A[4] }
{ обработать A[5] }
```

1) если N велико (1000, 1000000)?

2) при изменении N программа не должна меняться!

# Как обработать все элементы массива?

## Обработка с переменной:

```
i := 1;
{ обработать A[i] }
i := i + 1;
{ обработать A[i] }
i := i + 1;
{ обработать A[i] }
i := i + 1;
{ обработать A[i] }
i := i + 1;
{ обработать A[i] }
```

```
i := i + 1;
```

## Обработка в цикле:

```
i := 1;
while i <= N do begin
  { обработать A[i] }
  i := i + 1;
end;
```

## Цикл с переменной:

```
for i:=1 to N do begin
  { обработать A[i] }
end;
```

# Заполнение массива

```
program Arr;
const N = 10;
var A: array[1..N] of integer;
    i: integer;
begin
  for i:=1 to N do
    A[i]:= i*i;
end.
```

Чему равен **A[9]**?

# Ввод с клавиатуры и вывод на экран

## Объявление:

```
const N = 5;
var A: array[1..N] of integer;
    i: integer;
```

## Ввод с клавиатуры:

```
for i:=1 to N do begin
  write('A[', i, ']=');
  read ( A[i] )
end;
```

? Почему **write**?

```
A[1] = 5
A[2] = 12
A[3] = 34
A[4] = 56
A[5] = 13
```

## Вывод на экран:

```
writeln('Массив A:');
for i:=1 to N do
  write(A[i]:4);
```

? Зачем «:4»?

# Заполнение случайными числами

*Задача*. Заполнить массив (псевдо)случайными целыми числами в диапазоне от 20 до 100.

```
for i:=1 to N do begin
  A[i]:= 20 + random(81);
  write(A[i],' ')
end;
```

# Перебор элементов

## Общая схема:

```
for i:=1 to N do begin
  ... { сделать что-то с A[i] }
end;
```

## Подсчёт нужных элементов:

*Задача*. В массиве записаны данные о росте баскетболистов. Сколько из них имеет рост больше 180 см, но меньше 190 см?

```
count:= 0;
for i:=1 to N do
  if (180 < A[i]) and (A[i] < 190) then
    count:= count + 1;
```

# Перебор элементов

## Среднее арифметическое:

```
count:= 0; sum:= 0;
for i:=1 to N do
  if (180 < A[i]) and (A[i] < 190)
  then begin
    count:= count + 1;
    sum:= sum + A[i]
  end;
write(sum/count);
```

среднее арифметическое

# Задачи

**«A»:** Заполните массив случайными числами в интервале [0,100] и найдите среднее арифметическое его значений.

**Пример:**

```
Массив:
1 2 3 4 5
Среднее арифметическое 3.000
```

**«B»:** Заполните массив случайными числами в интервале [0,100] и подсчитайте отдельно среднее значение всех элементов, которые <50, и среднее значение всех элементов, которые ≥50.

**Пример:**

```
Массив:
3 2 52 4 60
Ср. арифм. элементов [0,50): 3.000
Ср. арифм. элементов [50,100]: 56.000
```

# Задачи

**«С»:** Заполните массив из N элементов случайными числами в интервале [1,N] так, чтобы в массив обязательно вошли все числа от 1 до N (постройте случайную перестановку).

**Пример:**

```
Массив:
3 2 1 4 5
```

# Программирование на языке Паскаль

## § 63. Алгоритмы обработки массивов

# Поиск в массиве

## Найти элемент, равный X:

```
i := 1;
while A[i] <> X do
  i := i + 1;
write('A[',i,']=',X);
```

? Что плохо?

```
i := 1;
while (i <= N) and (A[i] <> X) do
  i := i + 1;
if i <= N then
    write('A[',i,']=',X)
else write('Не нашли!');
```

должно быть первым!

? Что если такого нет?

# Поиск в массиве

## Вариант с досрочным выходом:

```
nX:= 0;
for i:=1 to N do
  if A[i] = X then begin
    nX:= i;
    break
  end;
if nX > 0 then
    write('A[',i,']=',X)
else write('Не нашли!');
```

досрочный выход из цикла

# Задачи

**«А»:** Заполните массив случайными числами в интервале [0,5]. Введите число X и найдите все значения, равные X.

**Пример:**

```
Массив:
1 2 3 1 2
Что ищем:
2
Нашли: A[2]=2, A[5]=2
```

**Пример:**

```
Массив:
1 2 3 1 2
Что ищем:
6
Ничего не нашли.
```

# Задачи

**«B»:** Заполните массив случайными числами в интервале [0,5]. Определить, есть ли в нем элементы с одинаковыми значениями, стоящие рядом.

**Пример:**

```
Массив:
1 2 3 3 2 1
Есть: 3
```

**Пример:**

```
Массив:
1 2 3 4 2 1
Нет
```

# Задачи

**«С»:** Заполните массив случайными числами. Определить, есть ли в нем элементы с одинаковыми значениями, не обязательно стоящие рядом.

**Пример:**

```
Массив:
3 2 1 3 2 5
Есть: 3, 2
```

**Пример:**

```
Массив:
3 2 1 4 0 5
Нет
```

# Максимальный элемент

```
M:=A[1];
for i:=2 to N do
  if A[i]>M then
    M:=A[i];
write(M);
```

? Как найти его номер?

```
M:=A[1]; nMax:=1;
for i:=2 to N do
  if A[i] > M then begin
    M:=A[i];
    nMax:=i;
  end;
write('A[',nMax,']=',M);
```

? Что можно улучшить?

# Максимальный элемент и его номер

По номеру элемента можно найти значение!

```
nMax:=1;
for i:=2 to N do
  if A[i]>A[nMax] then
    nMax:=i;
write('A[',nMax,']=',A[nMax]);
```

# Задачи

**«A»:** Заполнить массив случайными числами и найти минимальный и максимальный элементы массива и их номера.

**Пример**:

```
Массив:
1 2 3 4 5
Минимальный элемент: A[1]=1
Максимальный элемент: A[5]=5
```

**«B»:** Заполнить массив случайными числами и найти два максимальных элемента массива и их номера.

**Пример**:

```
Массив:
5 5 3 4 1
Максимальный элемент: A[1]=5
Второй максимум: A[2]=5
```

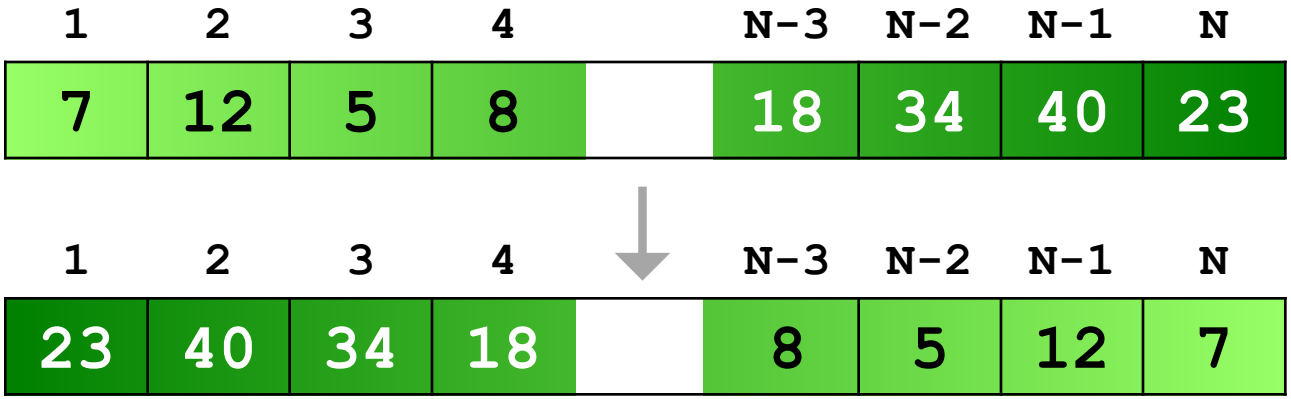
# Задачи

**«С»:** Введите массив с клавиатуры и найдите (за один проход) количество элементов, имеющих максимальное значение.

**Пример:**

```
Массив:
3 4 5 5 3 4 5
Максимальное значение 5
Количество элементов 3
```

# Реверс массива




**«Простое» решение:**

остановиться на середине!

```
for i:=1 to N div 2 do
  поменять местами A[i] и A[N+1-i]
```

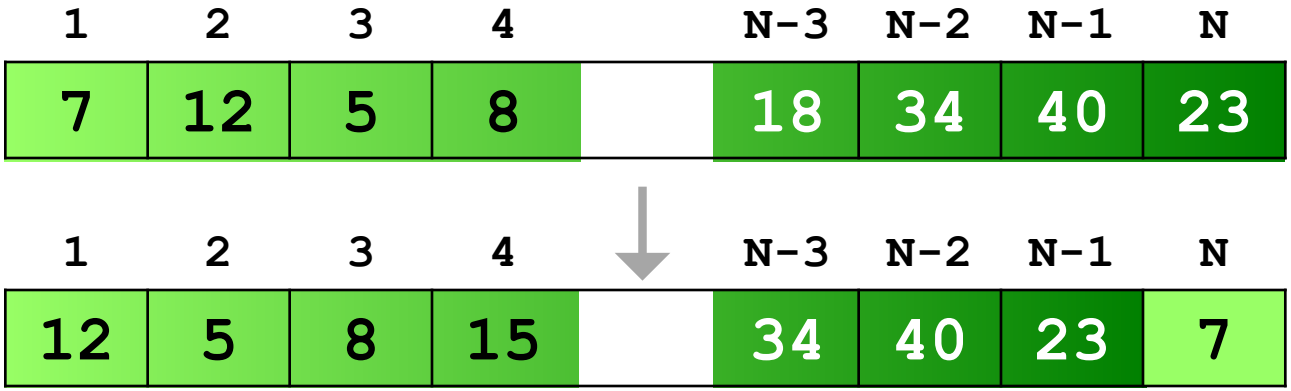
? Что плохо?

# Реверс массива

```
for i:=1 to N div 2 do begin
  c:=A[i];
  A[i]:=A[N+1-i];
  A[N+1-i]:=c;
end;
```

? Как обойтись без переменной c?

# Циклический сдвиг элементов




## «Простое» решение:

```
for i:=1 to N-1 do
  A[i]:=A[i+1]
```

? Почему не до N?

? Что плохо?

# Задачи

**«A»:** Заполнить массив случайными числами и выполнить циклический сдвиг элементов массива вправо на 1 элемент.

**Пример:**

```
Массив:
1 2 3 4 5 6
Результат:
6 1 2 3 4 5
```

**«B»:** Массив имеет четное число элементов. Заполнить массив случайными числами и выполнить реверс отдельно в первой половине и второй половине.

**Пример:**

```
Массив:
1 2 3 4 5 6
Результат:
3 2 1 6 5 4
```

# Задачи

**«С»:** Заполнить массив случайными числами в интервале [-100,100] и переставить элементы так, чтобы все положительные элементы стояли в начала массива, а все отрицательные и нули – в конце. Вычислите количество положительных элементов.

**Пример:**

```
Массив:
20 -90 15 -34 10 0
Результат:
20 15 10 -90 -34 0
Количество положительных элементов: 3
```

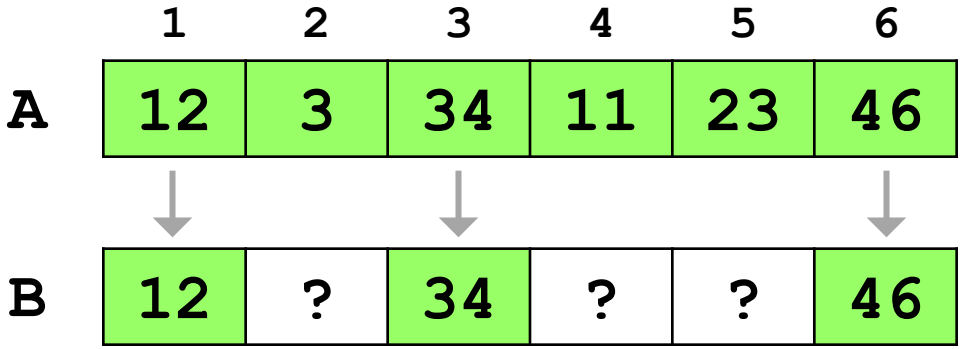
# Отбор нужных элементов

*Задача*. Отобрать элементы массива **A**, удовлетворяющие некоторому условию, в массив **B**.

**«Простое» решение:**

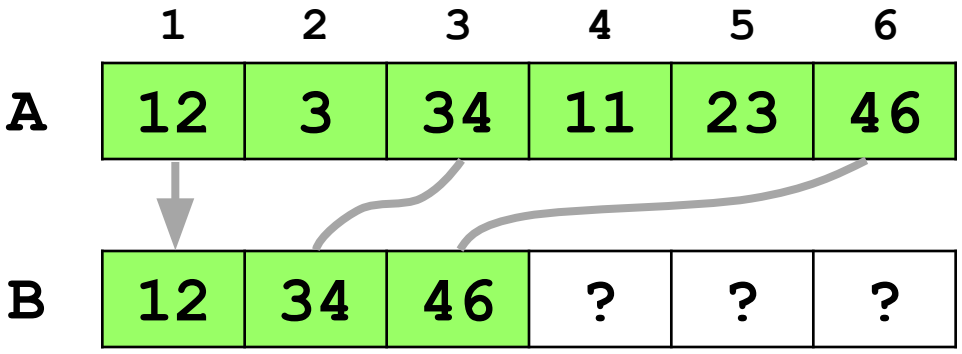
```
for i:=1 to N do
  if условие выполняется для A[i] then
    B[i]:=A[i];
```

? Что плохо?




выбрать чётные элементы

# Отбор нужных элементов




выбрать чётные элементы

```
count:= 0; { счётчик }
for i:= 1 to N do
  if A[i] mod 2 = 0 then begin
    count:= count + 1;
    B[count]:= A[i];
  end;
```

? Если A и B – один и тот же массив?

? Как вывести на экран?

```
for i:= 1 to count do
  write(B[i], ' ');
```

# Задачи

**«A»:** Заполнить массив случайными числами в интервале [-10,10] и отобрать в другой массив все чётные отрицательные числа.

**Пример:**

```
Массив A:
-5 6 7 -4 -6 8 -8
Массив B:
-4 -6 -8
```

**«B»:** Заполнить массив случайными числами в интервале [0,100] и отобрать в другой массив все простые числа. Используйте логическую функцию, которая определяет, является ли переданное ей число простым.

**Пример:**

```
Массив A:
12 13 85 96 47
Массив B:
13 47
```

# Задачи

**«С»:** Заполнить массив случайными числами и отобрать в другой массив все числа Фибоначчи. Используйте логическую функцию, которая определяет, является ли переданное ей число числом Фибоначчи.

**Пример:**

```
Массив A:
12 13 85 34 47
Массив B:
13 34
```

# Программирование на языке Паскаль

## § 64. Сортировка

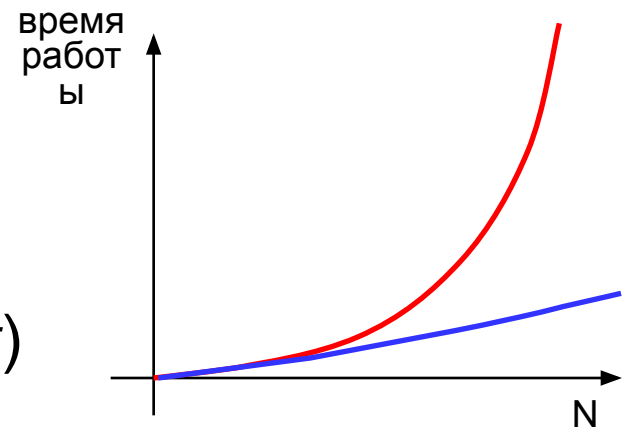
# Что такое сортировка?

**Сортировка** – это расстановка элементов массива в заданном порядке.

…по возрастанию, убыванию, последней цифре, сумме делителей, по алфавиту, …

**Алгоритмы:**

- простые и понятные, но неэффективные для больших массивов
  - **метод пузырька**
  - **метод выбора**
- сложные, но эффективные
  - **«быстрая сортировка»** (*QuickSort*)
  - сортировка «кучей» (*HeapSort*)
  - сортировка слиянием (*MergeSort*)
  - пирамидальная сортировка

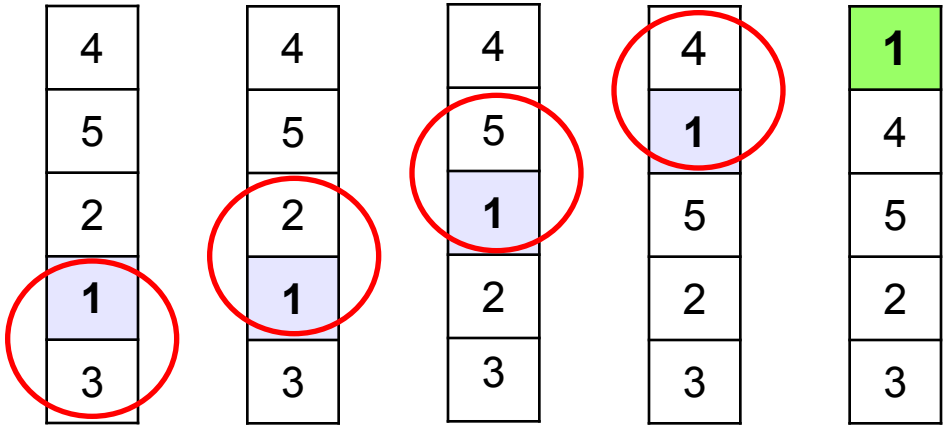


# Метод пузырька (сортировка обменами)

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

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

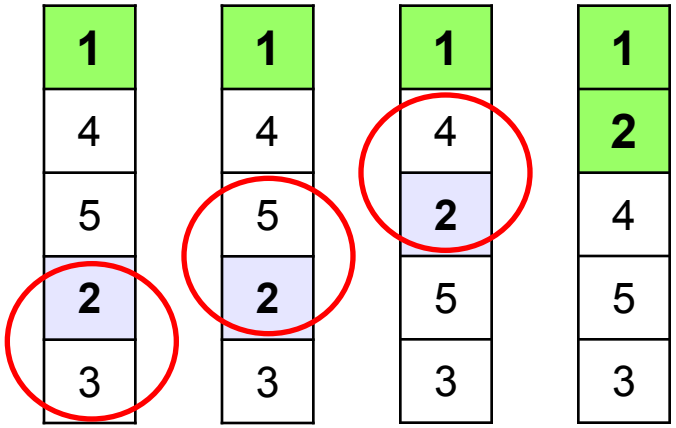
## 1-й проход:



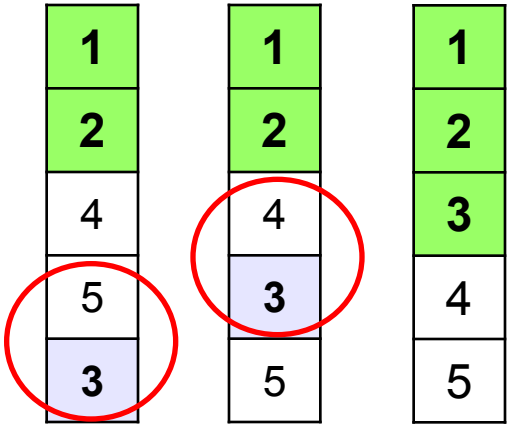

- сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву один элемент (самый маленький) становится на свое место

# Метод пузырька

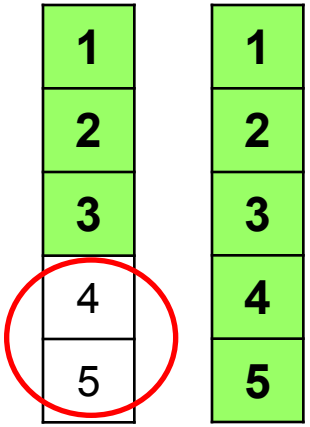
## 2-й проход:




## 3-й проход:




## 4-й проход:




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

# Метод пузырька

## 1-й проход:

```
for j:=N-1 downto 1 do
  if A[j+1]< A[j] then begin
    { поменять местами A[j] и A[j+1] }
  end;
```

единственное отличие!

## 2-й проход:

```
for j:=N-1 downto 2 do
  if A[j+1]< A[j] then begin
    { поменять местами A[j] и A[j+1] }
  end;
```

# Метод пузырька

```
for i:=1 to N-1 do
  for j:=N-1 downto i do
    if A[j+1]< A[j] then begin
      { поменять местами A[j] и A[j+1] }
    end;
```

? Как написать метод «камня»?

? Как сделать рекурсивный вариант?

# Задачи

**«A»:** Напишите программу, в которой сортировка выполняется «методом камня» – самый «тяжёлый» элемент опускается в конец массива.

**«B»:** Напишите вариант метода пузырька, который заканчивает работу, если на очередном шаге внешнего цикла не было перестановок.

**«C»:** Напишите программу, которая сортирует массив по убыванию суммы цифр числа. Используйте функцию, которая определяет сумму цифр числа.

# Метод выбора (минимального элемента)

*Идея*: найти минимальный элемент и поставить его на первое место.

```
for i:=1 to N-1 do begin
  { найти номер nMin минимального элемента
    из A[i]..A[N] }
  if i <> nMin then begin
    { поменять местами A[i] и A[nMin] }
  end
end;
```

# Метод выбора (минимального элемента)

```
for i:=1 to N-1 do begin
  nMin:=i;
  for j:=i+1 to N do
    if A[j]<A[nMin] then
      nMin:=j;
  if i<>nMin then begin
    { поменять местами A[i] и A[nMin] }
  end
end;
```

? Как поменять местами два значения?

# Задачи

**«А»:** Массив содержит четное количество элементов. Напишите программу, которая сортирует первую половину массива по возрастанию, а вторую – по убыванию. Каждый элемент должен остаться в «своей» половине.

**Пример**:

```
Массив:
5 3 4 2 1 6 3 2
После сортировки:
2 3 4 5 6 3 2 1
```

# Задачи

**«B»:** Напишите программу, которая сортирует массив и находит количество различных чисел в нем.

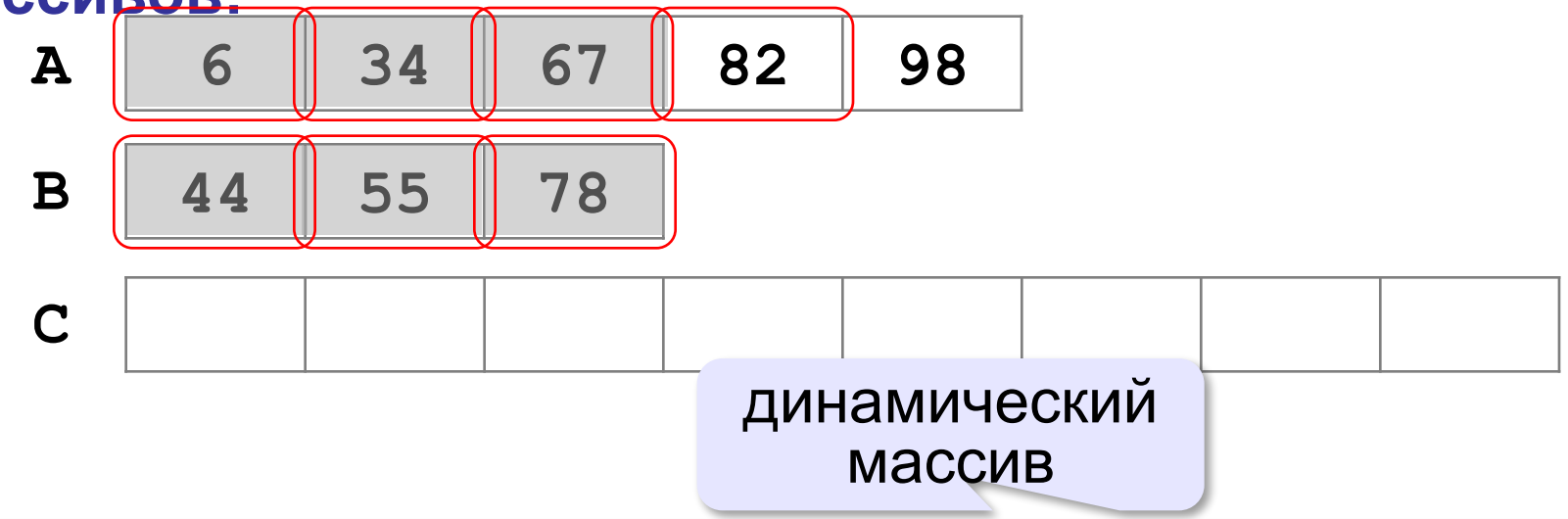
**Пример**:

```
Массив:
5 3 4 2 1 6 3 2 4
После сортировки:
1 2 2 3 3 4 4 5 6
Различных чисел: 5
```

**«C»:** Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком» и методом выбора. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.

# Сортировка слиянием

## Слияние отсортированных массивов:




```
type TIntArray = array of integer;
var iA, iB, nA, nB, j: integer;
    A, B, C: TIntArray;
```

# Процедура слияния (I)

```
iA:= 0; iB:= 0;   { позиции в A и B }
j:= 0;            { позиции в C }
nA:= Length(A);
nB:= Length(B);
setLength(C, nA + nB);

while (iA < nA) and (iB < nB) do begin
  if A[iA] <= B[iB] then begin { взять из A }
    C[j]:= A[iA]; Inc(iA)
  end
  else begin { взять из B }
    C[j]:= B[iB]; Inc(iB)
  end;
  Inc(j)  { к следующей ячейке C }
end;
```

# Процедура слияния (II)

Дописываем «хвост» одного массива:

```
while iA < nA do begin
  C[j]:= A[iA];
  Inc(iA);
  Inc(j)
end;
while iB < nB do begin
  C[j]:= B[iB];
  Inc(iB);
  Inc(j)
end;
```

# Процедура слияния (III)

## Оформляем функцию:

```
function merge(A, B: TIntArray): TIntArray;
var iA, iB, nA, nB, j: integer;
    C: TIntArray;
begin
  { всё, что написано раньше }
  Result := C;
end;
```

# Сортировка слиянием

```
function mergeSort(A: TIntArray): TIntArray;
var L, R: TIntArray;
    i, mid: integer;
begin
  if Length(A) = 1 then begin
    Result = A;
    Exit;
  end;
  ...
end;
```

выход из рекурсии

# Сортировка слиянием

```
function mergeSort(A: TIntArray): TIntArray;
var L, R: TIntArray;
    i, mid: integer;
begin
  { выход из рекурсии }
  mid:= Length(A) div 2;
  setLength(L, mid);
  setLength(R, Length(A)-mid);
  for i:=0 to High(L) do
    L[i]:= A[i];
  for i:=0 to High(R) do
    R[i]:= A[mid+i];
  ...
end;
```

номер среднего элемента

левая половина

правая половина

# Сортировка слиянием

```
function mergeSort(A: TIntArray): TIntArray;
var L, R: TIntArray;
    i, mid: integer;
begin
  { выход из рекурсии }
  { построение левой и правой частей}
  L:= mergeSort(L);
  R:= mergeSort(R);
  Result:= merge(L, R);
end;
```

рекурсивные вызовы

слияние

результат функции

# Сортировка слиянием

- работает быстро

- нужен дополнительный массив

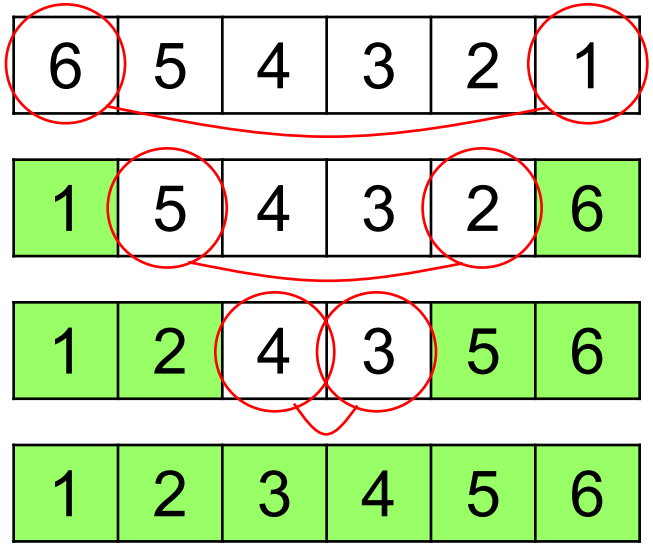
**«Разделяй и властвуй» (*divide and conquer*):**

1) задача разбивается на несколько подзадач меньшего размера;
2) эти подзадачи решаются с помощью рекурсивных вызовов того же (или другого) алгоритма;
3) решения подзадач объединяются, и получается решение исходной задачи.

# Быстрая сортировка (*QuickSort*)

Ч.Э.Хоар

*Идея*: выгоднее переставлять элементы, который находятся дальше друг от друга.

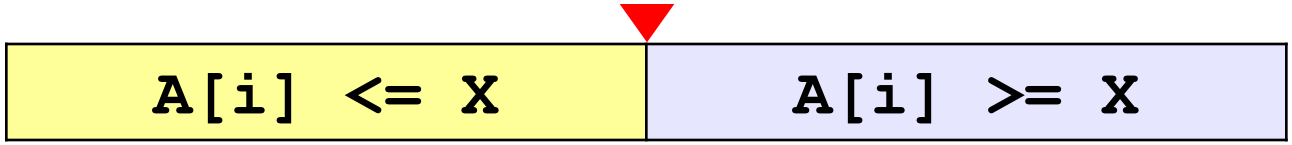



Для массива из **N** элементов нужно всего **N/2** обменов!

# Быстрая сортировка

**Шаг 1**: выбрать некоторый элемент массива X

**Шаг 2**: переставить элементы так:




при сортировке элементы не покидают « свою область»!

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

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


? Как лучше выбрать X?

**Медиана** – такое значение **X**, что слева и справа от него в отсортированном массиве стоит одинаковое число элементов (*это довольно сложно…*).

# Быстрая сортировка

**Разделение**:

1)выбрать любой элемент массива (**X=67**)

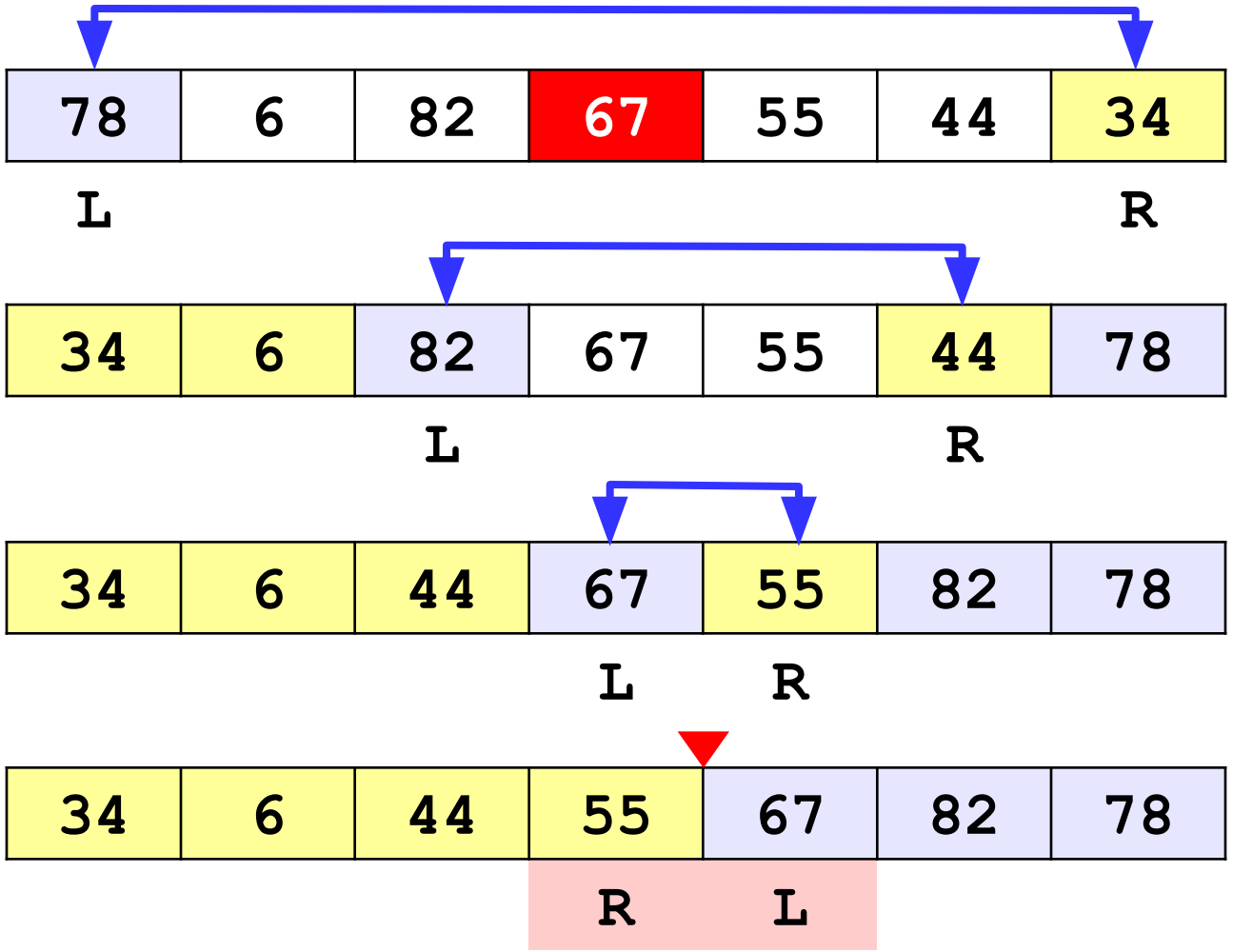

2)установить **L = 1, R = N**

3)увеличивая **L**, найти первый элемент **A[L]**, который **>= X** (должен стоять справа)

- уменьшая **R**, найти первый элемент **A[R]**, который **<= X** (должен стоять слева)

1)если **L<=R** то поменять местами **A[L]** и **A[R]** и перейти к п. 3
иначе **стоп**.

# Быстрая сортировка




**L > R** : разделение закончено!

# Быстрая сортировка

## Основная программа:

```
program QuickSort;
const N = 7;
var A: array[1..N] of integer;
...
begin
   { заполнить массив }
   qSort(1, N); { сортировка }
   { вывести результат }
end;
```

глобальные данные

# Быстрая сортировка

```
procedure qSort(nStart, nEnd: integer);
var L, R, c, X: integer;
begin
  if nStart >= nEnd then Exit;
  L:= nStart; R:= nEnd;
  X:= A[(L+R) div 2]; { или X:= A[L+random(R-L+1)] }
  while L <= R do begin { разделение }
    while A[L] < X do L:= L + 1;
    while A[R] > X do R:= R - 1;
    if L <= R then begin
      c:= A[L]; A[L]:= A[R]; A[R]:= c;
      L:= L+1; R:= R-1
    end
  end;
  qSort(nStart, R); { рекурсивные вызовы }
  qSort(L, nEnd)
end;
```

# Быстрая сортировка

**Сортировка массива случайных значений**:

| N | метод пузырька | метод выбора | быстрая сортировка |
|---|---|---|---|
| 1000 | 0,24 с | 0,12 с | 0,004 с |
| 5000 | 5,3 с | 2,9 с | 0,024 с |
| 15000 | 45 с | 34 с | 0,068 с |

# Задачи

**«B»:** Напишите программу, которая сортирует массив и находит количество различных чисел в нем. Используйте алгоритм быстрой сортировки.

**Пример**:

```
Массив:
5 3 4 2 1 6 3 2 4
После сортировки:
1 2 2 3 3 4 4 5 6
Различных чисел: 5
```

# Задачи

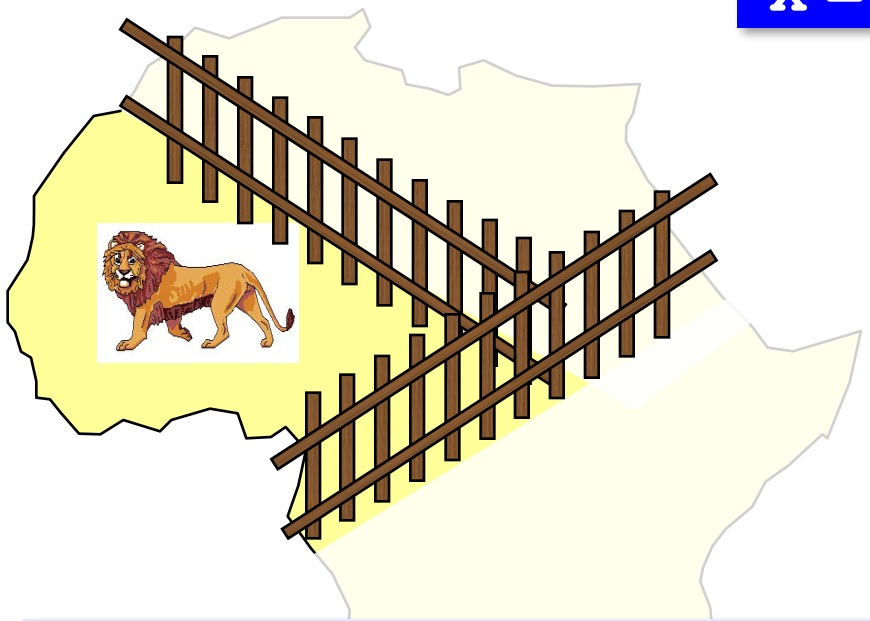
**«C»:** Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком», методом выбора и алгоритма быстрой сортировки. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.

**«D»:** Попробуйте построить массив из 10 элементов, на котором алгоритм быстрой сортировки показывает худшую эффективность (наибольшее число перестановок). Сравните это количество перестановок с эффективностью метода пузырька (для того же массива).

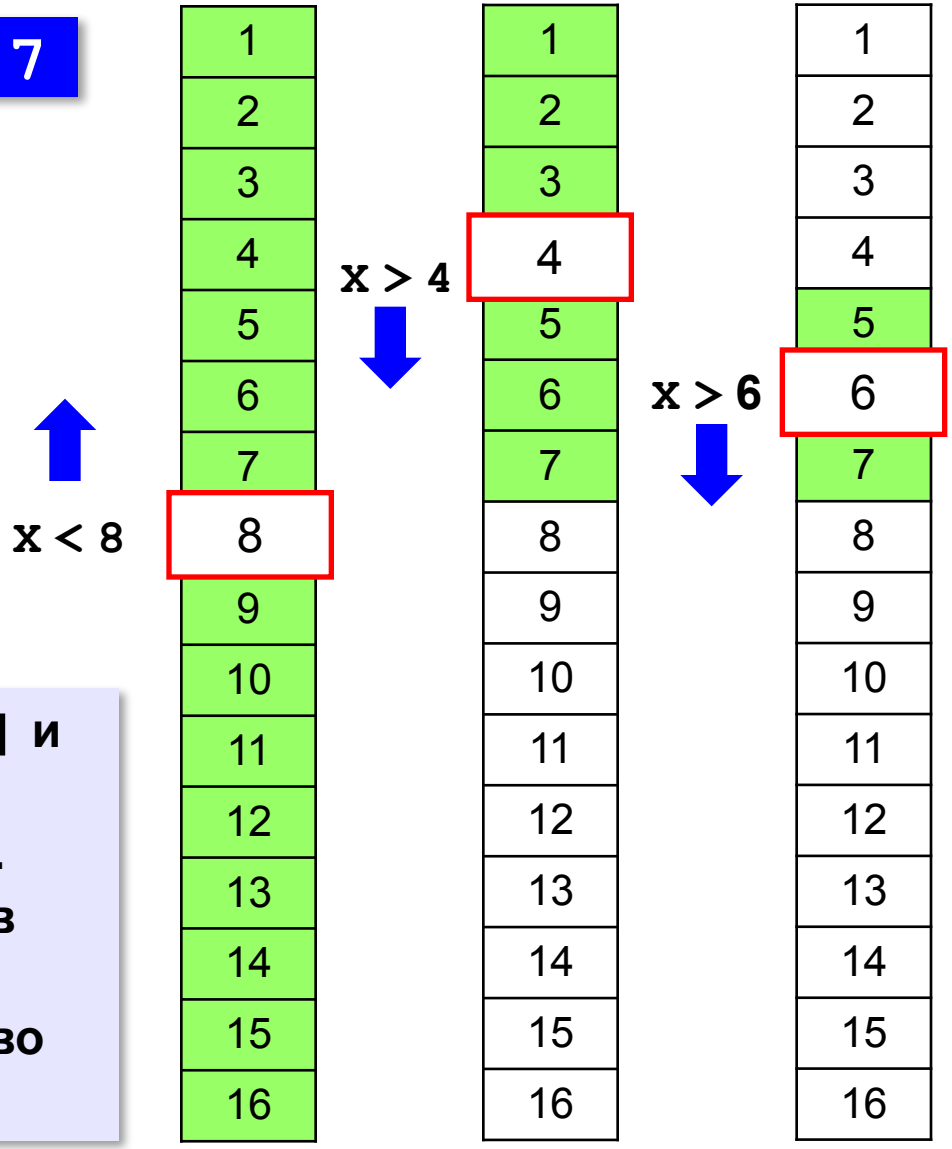
# Программирование на языке Паскаль

## § 65. Двоичный поиск

# Двоичный поиск

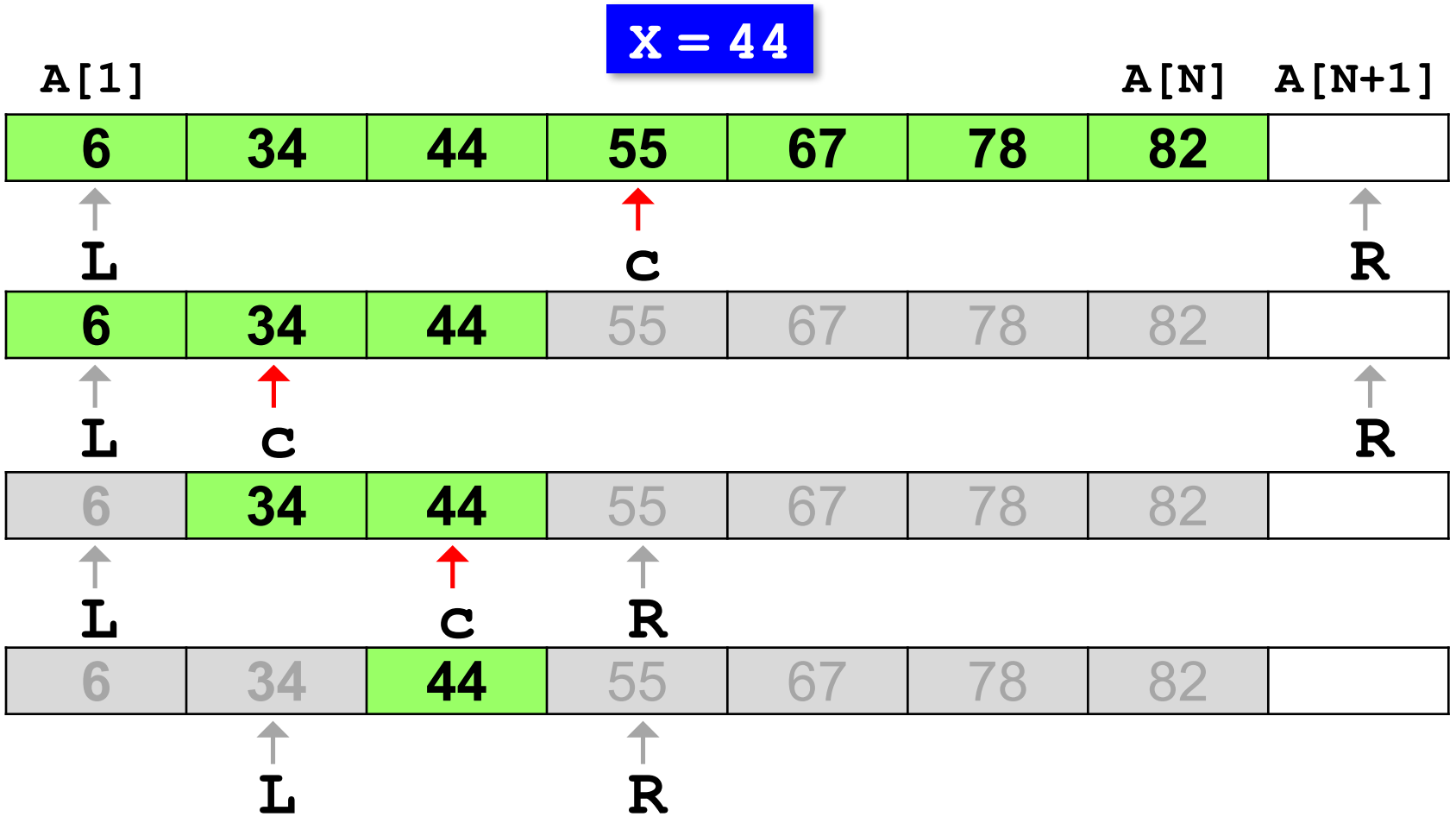


**X = 7**




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

# Двоичный поиск




**!** L = R−1 : поиск завершен!

# Двоичный поиск

```
var L, R, c: integer;
...
L:=1; R:=N+1;  { начальный диапазон }
while L<R-1 do begin
  c:=(L+R) div 2; { нашли середину }
  if X<A[c] then
       R:=c          { изменить диапазон }
  else L:=c;
end;
if A[L]=X then
     writeln('A[',L,']=',X)
else writeln('Не нашли!')
```

# Двоичный поиск

**Число сравнений**:

| N | линейный поиск | двоичный поиск |
|---|---|---|
| 2 | 2 | 2 |
| 16 | 16 | 5 |
| 1024 | 1024 | 11 |
| 1048576 | 1048576 | 21 |

- скорость выше, чем при линейном поиске

- нужна предварительная сортировка

Когда нужно применять?

# Задачи

**«А»:** Заполнить массив случайными числами и отсортировать его. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Подсчитать количество сравнений.

**Пример:**

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 7 9
Введите число X:
2
Число 2 найдено.
Количество сравнений: 2
```

# Задачи

**«B»:** Заполнить массив случайными числами и отсортировать его. Ввести число X. Используя двоичный поиск, определить, сколько чисел, равных X, находится в массиве.

**Пример:**

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 7 9
Введите число X:
4
Число 4 встречается 2 раз(а).
```

**Пример:**

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 7 9
Введите число X:
14
Число 14 не встречается.
```

# Задачи

**«С»:** Заполнить массив случайными числами и ввести число и отсортировать его. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Если такого числа нет, вывести число, ближайшее к X.

**Пример:**

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 12 19
Введите число X:
12
Число 12 найдено.
```

**Пример:**

```
Массив:
1 4 7 3 9 2 4 5 2
После сортировки:
1 2 2 3 4 4 5 12 19
Введите число X:
11
Число 11 не найдено. Ближайшее число 12.
```

# Программирование на языке Паскаль

## § 66. Символьные строки

# Зачем нужны символьные строки?

```
var s: array[1..80] of char;
       { массив символов }
```

- элементы массива – отдельные объекты
- сложно работать со строками переменной длины

**Хочется**:

- строка – единый объект
- длина строки может меняться во время работы программы

```
var s: string;  { символьная строка }
```

строка

# Символьные строки

**Присваивание**:

```
s:= 'Вася пошёл гулять';
```

```
var s: string;
```

**Ввод с клавиатуры**:

```
readln(s);
```

**Вывод на экран**:

```
writeln(s);
```

? А если массив?

**Отдельный символ**:

```
s[4]:= 'a';
```

**Длина строки**:

```
var n: integer;
...
n:= Length(s);
```

# Символьные строки

*Задача*: заменить в строке все буквы **'а'** на буквы **'б'**.

```
program ReplaceAB;
var s: string;
    i: integer;
begin
   writeln('Введите строку');
   readln(s);
   for i:=1 to Length(s) do
      if s[i]='а' then
         s[i]:='б';
   writeln(s);
end.
```

# Задачи

**«А»:** Ввести с клавиатуры символьную строку и заменить в ней все буквы «а» на «б» и все буквы «б» на «а» (заглавные на заглавные, строчные на строчные).

**Пример:**

```
Введите строку:
ааббААББссСС
Результат:
ббааББААссСС
```

# Задачи

**«B»:** Ввести с клавиатуры символьную строку и определить, сколько в ней слов. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

**Пример**:

```
Введите строку:
  Вася  пошел      гулять
Найдено слов: 3
```

# Задачи

**«С»:** Ввести с клавиатуры символьную строку и найдите самое длинное слово и его длину. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

**Пример:**

```
Введите строку:
  Вася   пошел  гулять
Самое длинное слово: гулять, длина 6
```

# Операции со строками

## Объединение *(конкатенация)* :

```
s1:= 'Привет';
s2:= 'Вася';
s  := s1 + ', ' + s2 + '!';
```

'Привет, Вася!'

## Срез:

```
s:= '123456789';
s1:= Copy(s, 3, 5);        { '34567' }
```

откуда

с какого символа

сколько символов

5

# Удаление и вставка символов

**Удаление**:

```
s:= '123456789';
Delete(s, 3, 6); { '129' }
```

с какого символа

сколько символов

**Вставка**:

```
s:= '123456789';
Insert('ABC', s, 3); { '12ABC3456789' }
```

что

куда

с какого символа

# Поиск в строках

```
s:= 'Здесь был Вася.';
          что  где
n:= Pos('с', s)
if n > 0 then
  write('Номер символа ', n)
else
  write('Символ не найден.');
```

Находит первое слева вхождение подстроки!

# Пример обработки строк

*Задача:* Ввести имя, отчество и фамилию. Преобразовать их к формату «фамилия-инициалы».

**Пример:**

```
Введите имя, отчество и фамилию:
Василий Алибабаевич Хрюндиков
Результат:
Хрюндиков В.А.
```

Алибабаевич Хрюндиков

**Алгоритм:**

- найти первый пробел и выделить имя
- удалить имя с пробелом из основной строки
- найти первый пробел и выделить отчество
- удалить отчество с пробелом из основной строки
- «сцепить» фамилию, первые буквы имени и фамилии, точки, пробелы…

Хрюндиков

Хрюндиков В.А.

# Пример обработки строк

```
program FIO;
var s, name, name2: string;
    n: integer;
begin
   write('Введите имя, отчество и фамилию: ');
   readln(s);
   n:= Pos(' ', s);
   name:= Copy(s, 1, n-1); { взять имя }
   Delete(s, 1, n);
   n:= Pos(' ', s);
   name2:= Copy(s, 1, n-1);{ взять отчество }
   Delete(s, 1, n);          { осталась фамилия }
   s:= s + ' ' + name[1] + '.' + name2[1] + '.';
   writeln(s)
end.
```

# Задачи

**«А»:** Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

**Пример**:

```
Введите фамилию, имя и отчество:
Иванов Петр Семёнович
П.С. Иванов
```

# Задачи

**«B»:** Ввести адрес файла и «разобрать» его на части, разделенные знаком `'/'`. Каждую часть вывести в отдельной строке.

**Пример**:

```
Введите адрес файла:
C:/Фото/2013/Поход/vasya.jpg
C:
Фото
2013
Поход
vasya.jpg
```

# Задачи

**«С»:** Напишите программу, которая заменяет во всей строке одну последовательность символов на другую.

**Пример**:

```
Введите строку:
(X > 0) and (Y < X) and (Z > Y) and (Z <> 5)
Что меняем: and
Чем заменить: &
Результат
(X > 0) & (Y < X) & (Z > Y) & (Z <> 5)
```

# Преобразования «строка» – «число»

## Из строки в число:

0 или номер неверного символа

```
var N: integer;
    X: real;
    s: string;
    r: integer;
```

```
s:='123';
Val(s,N,r); { N = 123 }
s:='123.456';
Val(s,X,r); { X = 123.456 }
```

## Из числа в строку:

```
N:=123;
Str(N,s);          { s = '123' }
X:=123.456;
Str(X,s);          { s ='1.234560E+002' }
Str(X:10:3,s);     { s = '   123.456' }
```

# Задачи

**«A»:** Напишите программу, которая вычисляет сумму трех чисел, введенную в форме символьной строки. Все числа целые.

**Пример**:

```
Введите выражение:
12+3+45
Ответ: 60
```

**«B»:** Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются только знаки «+» или «–»). Выражение вводится как символьная строка, все числа целые.

**Пример**:

```
Введите выражение:
12-3+45
Ответ: 54
```

# Задачи

**«С»:** Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются знаки «+», «–», «*» и «/»). Выражение вводится как символьная строка, все числа целые. Операция «/» выполняется как целочисленное деление (**div**).

**Пример**:

```
Введите выражение:
12*3+45
Ответ: 81
```

# Задачи

**«D»:** Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются знаки «+», «–», «*» и «/») **и круглых скобок**. Выражение вводится как символьная строка, все числа целые. Операция «/» выполняется как целочисленное деление (**div**).

**Пример:**

```
Введите выражение:
2*(3+45)+4
Ответ: 100
```

# Строки в процедурах и функциях

*Задача*: построить процедуру, которая заменяет в строке **s** все вхождения слова-образца **wOld** на слово-замену **wNew**.

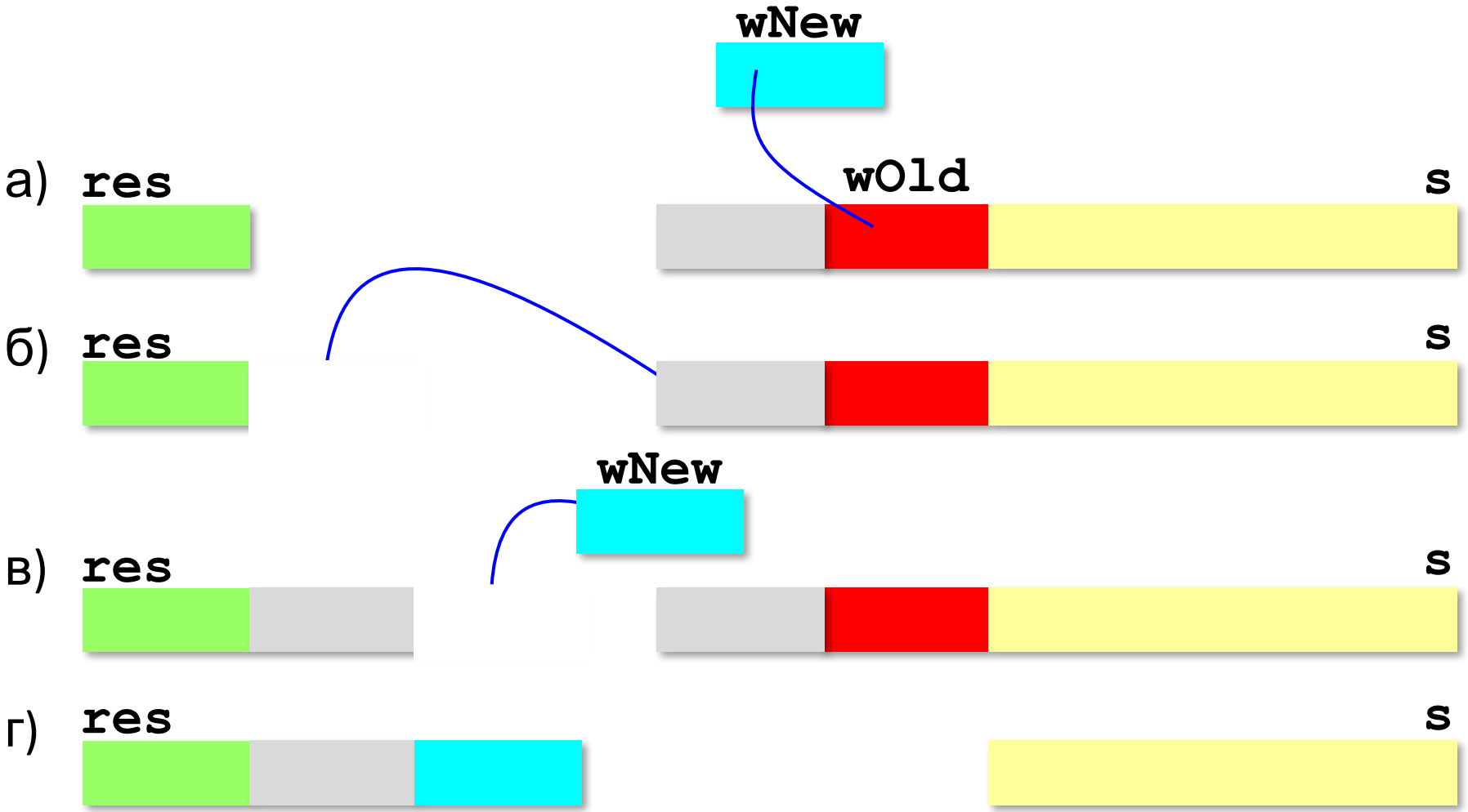
```
пока { слово wOld есть в строке s }
   { удалить слово wOld из строки }
   { вставить на это место слово wNew }
```


Что плохо?

```
wOld: '12'
wNew: 'A12B'
```

зацикливание

# Замена всех экземпляров подстроки



# Замена всех экземпляров подстроки

```
program Replace;
var s: string;
... { здесь будет процедура }
begin
  s:='12.12.12';
  replaceAll(s, '12', 'A12B');
  writeln(s)
end;
```

| рабочая строка **s** | результат **res** |
|---|---|
| '12.12.12' | '' |
| | |
| | |

# Замена всех экземпляров подстроки

```
procedure replaceAll(var s: string; wOld, wNew: string);
var res: string;
    p, len: integer;
begin
  len:=Length(wOld);
  res:='';
  while Length(s)>0 do begin
    p:=Pos(wOld, s);
    if p=0 then begin res:=res+s; break; end;
    if p>1 then res:=res+Copy(s,1,p-1);
    res:=res+wNew;
    if p+len>Length(s) then
        s:=''
    else s:=Copy(s,p+len,Length(s));
  end;
  s:=res
end;
```

найти слово **wOld**

не нашли…

добавить **wNew**

добавить то, что перед ним

взять «хвост»

# Замена: из процедуры в функцию

```
program Replace;
var s: string;
 function replaceAll(s,
                wOld, wNew: string): string;
 ...
 begin
   ... { тело процедуры }
   replaceAll:=res
 end;
begin
   s:='12.12.12';
   s:=replaceAll(s, '12', 'A12B');
   writeln(s)
end;
```

# Задачи

**«А»:** Напишите функцию, которая возвращает первое слово переданной ей символьной строки.

**Пример**:

```
Введите строку: Однажды в студёную зимнюю пору...
Первое слово: Однажды
```

# Задачи

**«B»:** Напишите функцию, которая заменяет расширение файла на заданное новое расширение.

**Пример**:

```
Введите имя файла: qq
Введите новое расширение: tmp
Результат: qq.tmp
```

**Пример**:

```
Введите имя файла: qq.exe
Введите новое расширение: tmp
Результат: qq.tmp
```

**Пример**:

```
Введите имя файла: qq.work.xml
Введите новое расширение: tmp
Результат: qq.work.tmp
```

# Задачи

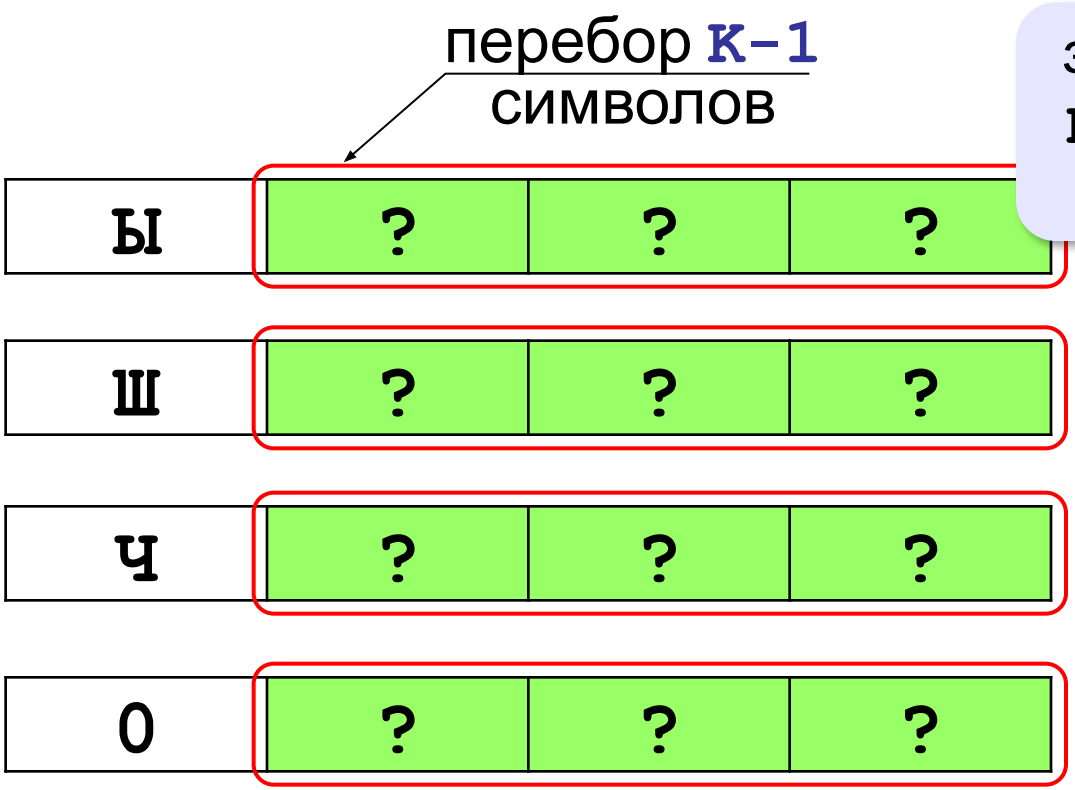
**«С»:** Напишите функцию, которая заменяет во всей строке все римские числа на соответствующие десятичные числа.

**Пример:**

```
Введите строку:
В MMXIII году в школе CXXIII состоялся очередной
    выпуск XI классов.
Результат:
В 2013 году в школе 123 состоялся очередной выпуск
    11 классов.
```

# Рекурсивный перебор

*Задача*. В алфавите языка племени «тумба-юмба» четыре буквы: «**Ы**», «**Ш**», «**Ч**» и «**О**». Нужно вывести на экран все слова, состоящие из **K** букв, которые можно построить из букв этого алфавита.




задача для слов длины **K** сведена к задаче для слов длины **K-1**!

# Рекурсивный перебор

```
Перебор К символов
var w: string;
begin
  w[1]:='Ы';
    { перебор последних К-1 символов }
  w[1]:='Ш';
    { перебор последних К-1 символов }
  w[1]:='Ч';
    { перебор последних К-1 символов }
  w[1]:='О';
    { перебор последних К-1 символов }
end;
```

# Рекурсивный перебор

```
program YSCHO;
var word: string;
 procedure TumbaWords(A, w: string; N: integer);
 var i: integer;
 begin
   if N = Length(w) then begin
     writeln(w);
     exit;
   end;
   for i:=1 to Length(A) do begin
     w[N+1]:= A[i];
     TumbaWords(A, w, N+1);
   end;
 end;
begin
  word:= '...';
  TumbaWords('ЫШЧО', word, 0);
end.
```

алфавит

слово

уже установлено

когда все символы уже установлены

по всем символам алфавита

# Задачи

**«A»:** В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых вторая буква «Ы». Подсчитайте количество таких слов.

**«B»:** В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых есть по крайней мере две одинаковые буквы, стоящие рядом. Подсчитайте количество таких слов.
Программа не должна строить другие слова, не соответствующие условию.

# Задачи

**«С»:** В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых есть по крайней мере две одинаковые буквы, не обязательно стоящие рядом.

Программа не должна строить другие слова, не соответствующие условию.

# Сравнение строк

**Пар ? пар ? парк**

Сравнение по кодам символов:

| | 0 | 1 | ... | 8 | 9 |
|---|---|---|---|---|---|
| CP-1251 | 48 | 49 | ... | 56 | 57 |
| UNCODE | 48 | 49 | ... | 56 | 57 |
| | **A** | **B** | ... | **Y** | **Z** |
| CP-1251 | 65 | 66 | ... | 89 | 90 |
| UNCODE | 65 | 66 | ... | 89 | 90 |
| | **a** | **b** | ... | **y** | **z** |
| CP-1251 | 97 | 98 | ... | 121 | 122 |
| UNCODE | 97 | 98 | ... | 121 | 122 |

# Сравнение строк

| | А | Б | ... | Ё | ... | Ю | Я |
|---|---|---|---|---|---|---|---|
| CP-1251 | 192 | 193 | ... | 168 | ... | 222 | 223 |
| UNCODE | 1040 | 1041 | ... | 1025 | ... | 1070 | 1071 |

| | а | б | ... | ё | ... | ю | я |
|---|---|---|---|---|---|---|---|
| CP-1251 | 224 | 225 | ... | 184 | ... | 254 | 255 |
| UNCODE | 1072 | 1073 | ... | 1105 | ... | 1102 | 1103 |

5STEAM < STEAM < Steam < steam

steam < ПАР < Пар < пАр < пар < парк

# Сортировка строк

```
const N = 10;
var i,j: integer;
    s1: string;
    S: array[1..N] of string;
begin
 for i:=1 to N do
   readln(S[i]);
 for i:=1 to N-1 do
   for j:=N-1 downto i do
     if S[j+1] < S[j] then begin
       s1:=S[j];
       S[j]:=S[j+1];
       S[j+1]:=s1;
     end;
 for i:=1 to N do
   writeln(S[i]);
end.
```

массив строк

# Задачи

**«А»:** Вводится 5 строк, в которых сначала записан порядковый номер строки с точкой, а затем – слово. Вывести слова в алфавитном порядке.

**Пример**:

```
Введите 5 строк:
1. тепловоз
2. арбуз
3. бурундук
4. кефир
5. урядник
Список слов в алфавитном порядке:
арбуз, бурундук, кефир, тепловоз, урядник
```

# Задачи

**«B»:** Вводится несколько строк (не более 20), в которых сначала записан порядковый номер строки с точкой, а затем – слово. Ввод заканчивается пустой строкой. Вывести введённые слова в алфавитном порядке.

**Пример**:

```
Введите слова:
1. тепловоз
2. арбуз

Список слов в алфавитном порядке:
арбуз, тепловоз
```

# Задачи

**«С»:** Вводится несколько строк (не более 20), в которых сначала записаны инициалы и фамилии работников фирмы. Ввод заканчивается пустой строкой. Отсортировать строки в алфавитном порядке по фамилии.

**Пример:**

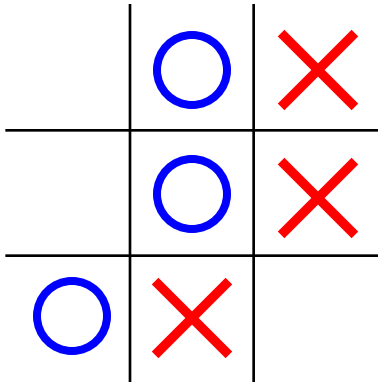
```
Введите ФИО:
А.Г. Урядников
Б.В. Тепловозов
В.Д. Арбузов

Список в алфавитном порядке:
В.Д. Арбузов
Б.В. Тепловозов
А.Г. Урядников
```

# Программирование на языке Паскаль

## § 67. Матрицы

# Что такое матрица?



нет знака

нолик

крестик

строка 2, столбец 3

? Как закодировать?

**Матрица** — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.). Каждый элемент матрицы имеет два индекса – номера строки и столбца.

# Объявление матриц

```
const N = 3; M = 4;
var A: array[1..N, 1..M] of integer;
    X: array[-3..0, -8..M] of double;
    L: a[1..N, 1..M] of boolean;
```

строки

столбцы

строки

столбцы

# Простые алгоритмы

## Заполнение случайными числами:

```
for i:=1 to N do begin
  for j:=1 to M do begin
    A[i,j]:= random(61) + 20;
    write(A[i,j]:3)
  end;
  writeln
end;
```

Вложенный цикл!

## Суммирование:

```
s:= 0;
for i:=1 to N do
  for j:=1 to M do
    s:= s + A[i,j];
```

# Задачи

**«А»:** Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале [10,99], и находит максимальный и минимальный элементы в матрице и их индексы.

**Пример**:

```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 11
40 12 35 15
Максимальный элемент A[2,2]=87
Минимальный элемент A[3,4]=11
```

# Задачи

**«B»:** Яркости пикселей рисунка закодированы числами от 0 до 255 в виде матрицы. Преобразовать рисунок в черно-белый по следующему алгоритму:

1) *вычислить среднюю яркость пикселей по всему рисунку*
2) *все пиксели, яркость которых меньше средней, сделать черными (записать код 0), а остальные – белыми (код 255)*

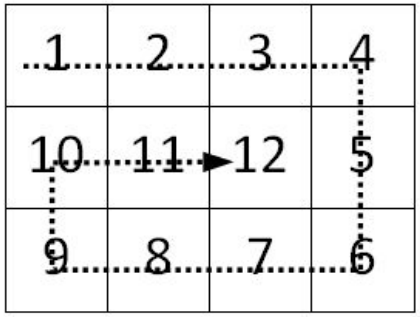
**Пример:**

```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 11
40 12 35 15
Средняя яркость 37.88
Результат:
  0   0 255 255
  0 255 255 255
255 255   0   0
255   0   0   0
```

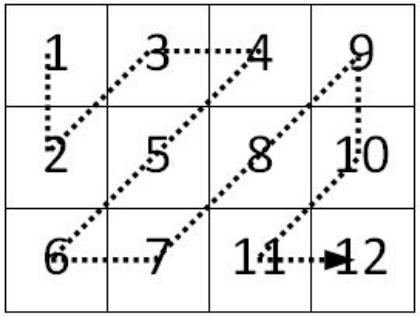
# Задачи

**«С»:** Заполните матрицу, содержащую N строк и M столбцов, натуральными числами по спирали и змейкой, как на рисунках:

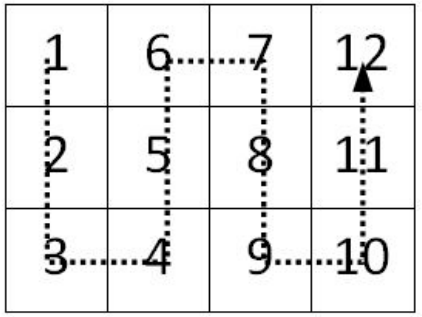
а)




б)



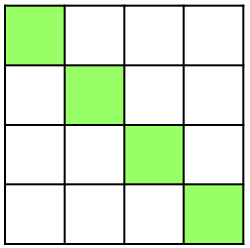

в)



# Перебор элементов матрицы

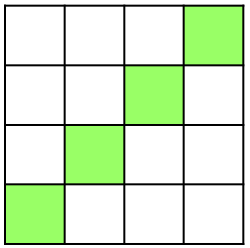
## Главная диагональ:

```
for i:=1 to N do begin
  { работаем с  A[i,i] }
end;
```



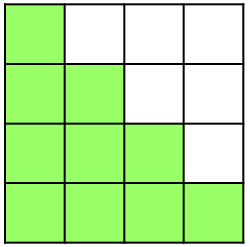
## Побочная диагональ:

```
for i:=1 to N do begin
  { работаем с  A[i,N+1-i] }
end;
```



## Главная диагональ и под ней:

```
for i:=1 to N do
  for j:=1 to i do begin
    { работаем с  A[i,j] }
  end;
```

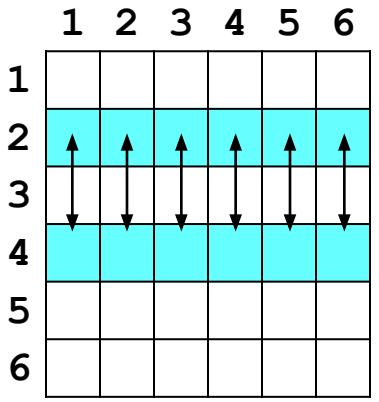


# Перестановка строк

## 2-я и 4-я строки:

```
for j:=1 to M do
  c:=A[2,j];
  A[2,j]:=A[4,j];
  A[4,j]:=c
end;
```

по всем столбцам



# Задачи

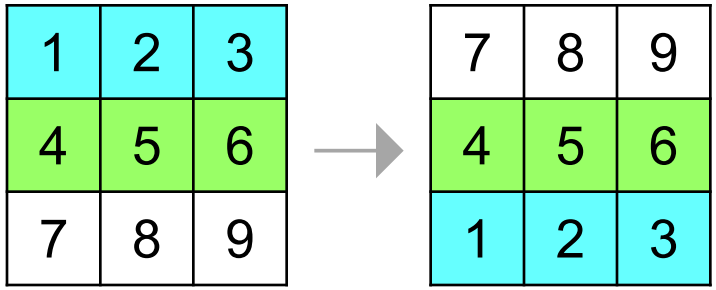
**«А»:** Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале [10,99], а затем записывает нули во все элементы выше главной диагонали. Алгоритм не должен изменяться при изменении размеров матрицы.

**Пример**:

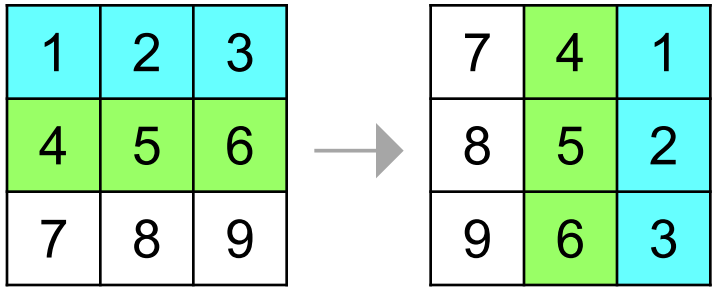
```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 30
40 12 35 65
Результат:
12  0  0  0
32 87  0  0
69 45 14  0
40 12 35 65
```

# Задачи

**«B»:** Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните отражение рисунка сверху вниз:



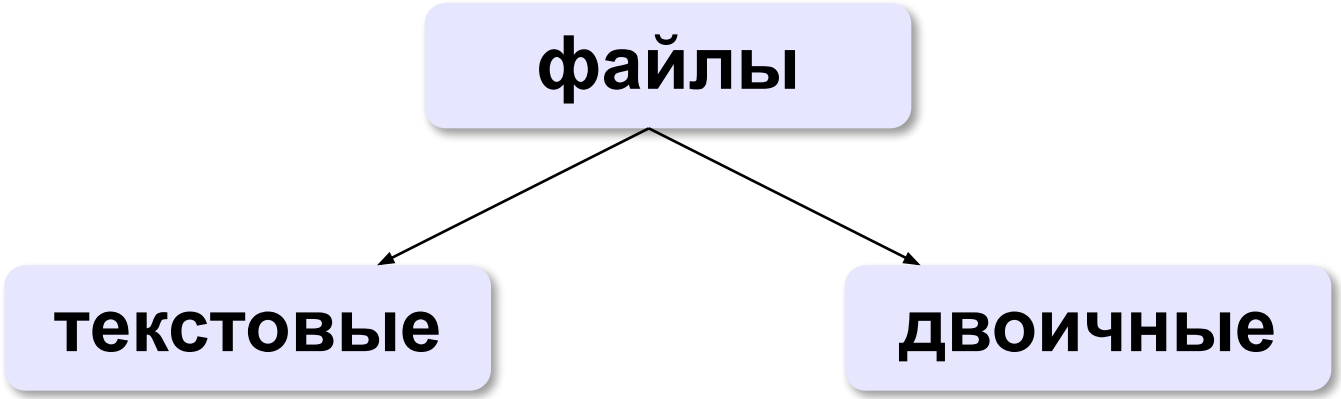

**«C»:** Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните поворот рисунка вправо на 90 градусов:



# Программирование на языке Паскаль

## § 68. Работа с файлами

# Как работать с файлами?




«*plain text*»:
- текст, разбитый на строки;
- из специальных символов только символы перехода на новую строку

- любые символы
- рисунки, звуки, видео, …

# Принцип сэндвича


файловые переменные

связать с файлами

```
var Fin, Fout: Text;
Assign(Fin, 'input.txt');
Assign(Fout, 'output.txt');
Reset(Fin);      { открыть на чтение }
Rewrite(Fout);   { открыть на запись }
   { здесь работаем с файлами }
Close(Fout);     { закрыть файлы }
Close(Fin);
```

# Ввод данных

```
var a, b: integer;
    Fin: Text;
...
Assign(Fin, 'input.txt');
Reset(Fin);
readln(Fin, a, b);
Close(Fin);
```

**Переход к началу открытого файла**:

```
Close(Fin);
Reset(Fin);
```

**Определение конца файла**:

```
if Eof(Fin) then  { end of file }
  write('Данные кончились');
```

# Вывод данных в файл

```
var a, b: integer;
    Fout: Text;
...
a:=1; b:=2;
Assign(Fout,'output.txt');
Rewrite(Fout);
writeln(Fout, a, '+', b, '=', a+b);
Close(Fout);
```

# Чтение неизвестного количества данных

*Задача*. В файле записано в столбик неизвестное количество чисел. Найти их сумму.

```
while { не конец файла } do begin
  { прочитать число из файла }
  { добавить его к сумме     }
end;
```

```
var x, S: integer;
    Fin: Text;
...
Assign(Fin,'input.txt');
Reset(Fin);
S:=0;
while not Eof(Fin) do begin
  readln(Fin, x);
  S:=S+x;
end;
Close(Fin);
```

# Задачи

**«A»:** Напишите программу, которая находит среднее арифметическое всех чисел, записанных в файле в столбик, и выводит результат в другой файл.

**«B»:** Напишите программу, которая находит минимальное и максимальное среди чётных положительных чисел, записанных в файле, и выводит результат в другой файл. Учтите, что таких чисел может вообще не быть.

**«C»:** В файле в столбик записаны целые числа, сколько их – неизвестно. Напишите программу, которая определяет длину самой длинной цепочки идущих подряд одинаковых чисел и выводит результат в другой файл.

# Обработка массивов

*Задача*. В файле записано не более 100 целых чисел. Вывести в другой текстовый файл те же числа, отсортированные в порядке возрастания.

? В чем отличие от предыдущей задачи?

! Для сортировки нужно удерживать все элементы в памяти одновременно.

```
const MAX = 100;
var A: array[1..MAX] of integer;
```

# Обработка массивов

## Ввод массива:

```
var N: integer;
    Fin: Text;
...
Assign(Fin,'input.txt');
Reset(Fin);
N:=0;
while (not Eof(Fin)) and (N<MAX) do
begin
  N:=N+1;
  readln(Fin, A[N]);
end;
Close(Fin);
```

счётчик прочитанных данных

? Зачем?

# Обработка массивов

## Вывод результата:

```
var Fout: Text;
...
Assign(Fout, 'output.txt');
Rewrite(Fout);
for i:=1 to N do
  writeln(Fout, A[i]);
Close(Fout);
```

# Задачи

**«A»:** В файле записано не более 100 чисел. Отсортировать их по возрастанию последней цифры и записать в другой файл.

**«B»:** В файле записано не более 100 чисел. Отсортировать их по возрастанию суммы цифр и записать в другой файл. Используйте функцию, которая вычисляет сумму цифр числа.

**«C»:** В двух файлах записаны отсортированные по возрастанию массивы неизвестной длины. Объединить их и записать результат в третий файл. Полученный массив также должен быть отсортирован по возрастанию.

# Обработка строк

*Задача*. В файле записано данные о собаках: в каждой строчке кличка собаки, ее возраст и порода:

```
Мухтар 4 немецкая овчарка
```

Вывести в другой файл сведения о собаках, которым меньше 5 лет.

```
пока { не конец файла }
  { прочитать строку из файла }
  { разобрать строку – выделить возраст }
  если возраст < 5 то
    { записать строку в файл Fout }
```

# Обработка строк

## Разбор строки:

```
{ найти в строке пробел }
{ удалить из строки кличку с первым пробелом }
{ найти в строке пробел }
{ выделить возраст перед пробелом }
{ преобразовать возраст в числовой вид }
```

```
var s, sAge: string;
    age, p, r: integer;
... { s = 'Мухтар 4 овчарка' }
p:= Pos(' ', s); { 'Мухтар 4 овчарка' }
Delete(s, 1, p); {     s = '4 овчарка' }
p:= Pos(' ', s); {         '4 овчарка' }
sAge:= Copy(s, 1, p-1); { sAge = '4' }
Val(sAge, age, r); { age = 4 }
```

# Обработка строк

```
var s, s0: string;
...
while not Eof(Fin) do begin
  readln(Fin, s0);
  s:=s0;
    { обработка строки s }
  if age < 5 then
    writeln(Fout, s0);
end;
```

Зачем s0?

# Задачи

**«A»:** В файле записаны данные о результатах сдачи экзамена. Каждая строка содержит фамилию, имя и количество баллов, разделенные пробелами:

```
<Фамилия> <Имя> <Количество баллов>
```

Вывести в другой файл фамилии и имена тех учеников, которые получили больше 80 баллов.

**«B»:** В предыдущей задаче добавить к полученному списку нумерацию, сократить имя до одной буквы и поставить перед фамилией:

```
1. Иванов
2. Петров
...
```

# Задачи

**«С»:** В файле записаны данные о результатах сдачи экзамена. Каждая строка содержит фамилию, имя и количество баллов, разделенные пробелами:

```
<Фамилия> <Имя> <Количество баллов>
```

Вывести в другой файл данные учеников, которые получили больше 80 баллов. Список должен быть отсортирован по убыванию балла. Формат выходных данных:

```
П. Иванов   98
И. Петров   96
...
```

# Конец фильма

**ПОЛЯКОВ Константин Юрьевич**
д.т.н., учитель информатики
ГБОУ СОШ № 163, г. Санкт-Петербург
kpolyakov@mail.ru

**ЕРЕМИН Евгений Александрович**
к.ф.-м.н., доцент кафедры мультимедийной дидактики и ИТО ПГГПУ, г. Пермь
eremin@pspu.ac.ru

# Источники иллюстраций

1. www.mcdonalds.com
2. иллюстрации художников издательства «Бином»
3. авторские материалы