L#6

Основы алгоритмизации и программирования. Массивы

Педагогическое образование, 3 семестр

Mayer Svetlana Fyodorovna

Ввод массива

```
var a:=ReadArrInteger(5);
var a:=ReadArrReal(5);
```

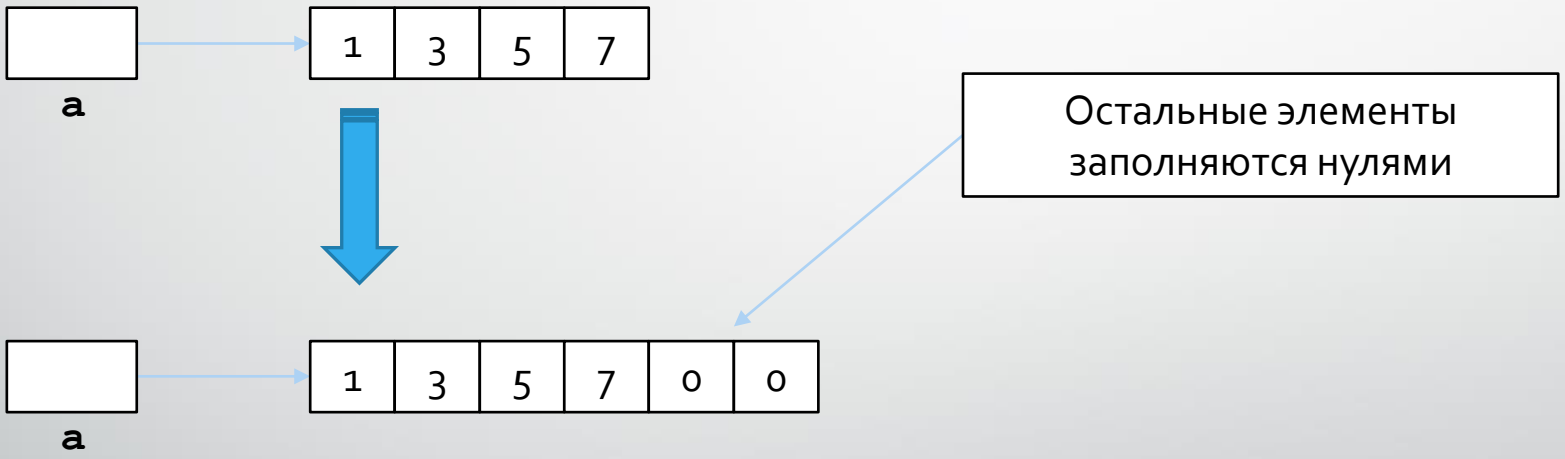
Заполнение случайными числами

```
var a:=new integer[10];
a:=arrRandomInteger(10); //  [0;9]
```

Динамическое расширение массива.

Процедура установки длины **SetLength**

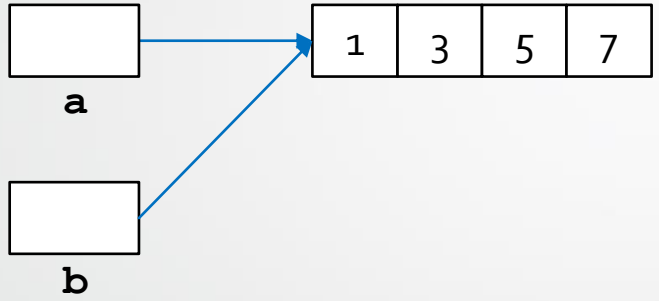
```
begin
  var a := Arr(1,3,5,7);
  SetLength(a,6);
  Print(a);
end.
```



Стандартная процедура **SetLength** выделяет необходимую память для размещения элементов массива

Переприсваивание:

```
var a: array of integer := (1,3,5,7);
var b:=a; // [1,3,5,7]
```



Но!

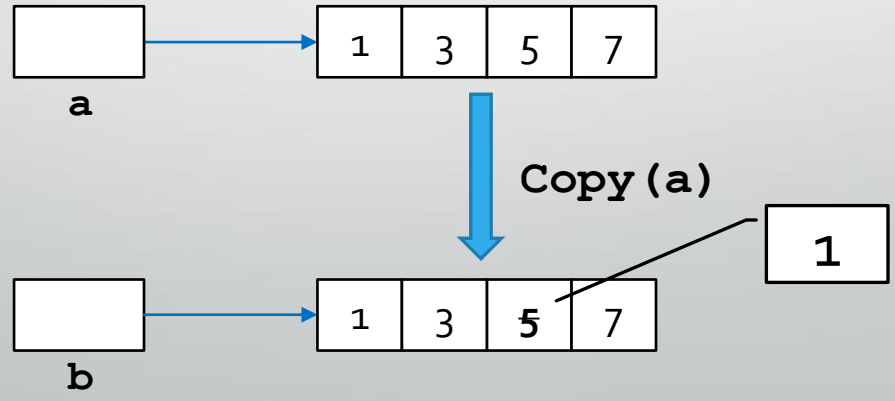
Если мы теперь переназначим значения элементов массива b, то массив a также изменится:

```
var a: array of integer := (1,3,5,7);
var b:=a; b[2]:=1;
print(a); //[1,3,1,7]
```

Функция Copy

Чтобы избежать этой ситуации, вам нужно создать массив **b** как копию массива **a** :

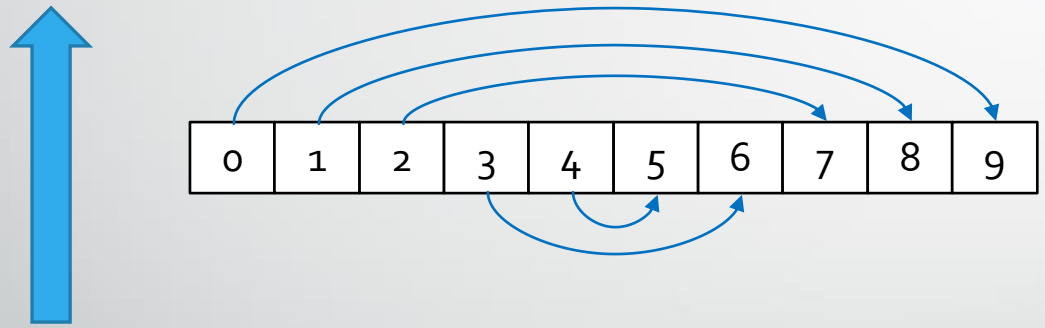
```
var a: array of integer := (1,3,5,7);
var b:=Copy(a);
b[2]:=1;

print(a); //[1,3,5,7]
```



Стандартные процедуры и методы

Перестановка (ревёрс) массива

```
Begin
var a: array of integer := (1,3,5,7);
var n := a.Length;
for var i:=0 to n div 2 - 1 do
    Swap(a[i],a[n-i-1]);
End.
```




```
begin
var a:=new integer[10];
a:=arrRandomInteger(10);
print(a);// [41,81,84,63,12,26,88,25,36,72]
Reverse(a);
print(a) //[72,36,25,88,26,12,63,84,81,41]
end.
```

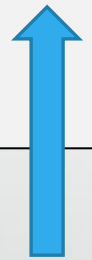
Стандартная процедура **Reverse(a)** имеет следующий алгоритм. Т.е. мы можем не вставлять этот код в программу, а просто пользоваться процедурой.

We can use slices: a := a[::-1]

Алгоритм линейного поиска

```
var a: array of integer := (1,3,5,7);
var x := readInteger('что ищем?');
Result := -1;
  for var i := 0 to a.Length - 1 do
    if a[i] = x then
    begin
      Result := i;
      break;
    end;
```

```
begin
  var a := new integer[10];
  a := arrRandomInteger(5,0,5); //[1,3,5,4,5]
  print(a.IndexOf(3)) // 1
end.
```



Есть стандартные методы **a.IndexOf(x)** и **a.LastIndexOf(x)**

Для поиска элемента в массиве, можно также использовать:

1. **a.Contains(x)**
2. x **in** a

Трансформация элементов массива

Задача: изменение элементов массива по правилу **x -> f(x)**

```
begin
  var a := new integer[5];
  a := arrRandomInteger(5); // [4,36,93,36,29]
  a.Transform(a,a -> a mod 2 = 0 ? a-1 : a+1);
  print(a) // [3,35,94,35,30]
end.
```

Существует стандартный метод **a.Transform**(x -> x*x)

для трансформации элементов массива по условию:

a.Transform(x -> x **mod** 2 = 0 ? x – 1 : x + 1)

Количество определенных элементов по условию

```
begin
  var a := new integer[5];
  a := arrRandomInteger(5); // [18,10,91,47,35]
  print(a);
  print(a.Count(a->odd(a))) // 3
end.
```

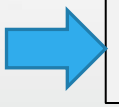
Стандартная процедура **a.Count(condition)**

Минимальный элемент и его индекс

Два решения:

```
…
var (min, minind) := (a[0], 0);
for var i:=1 to a.Length-1 do
  if a[i]<min then
    (min, minind) := (a[i], i);
Result := (min, minind);
```

Стандартная процедура **a.Min**, **a.IndexMin**



```
begin
  var a := new integer[5];
  a := arrRandomInteger(5); // [86,37,41,45,76]
  print(a.Min,a.IndexMin); // 37  1
end.
```

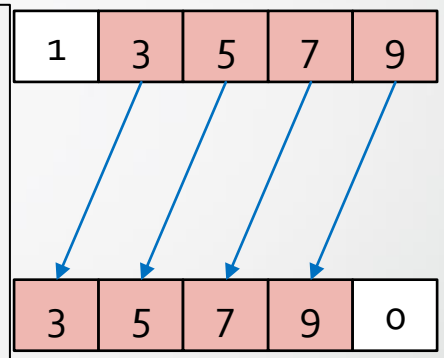
```
…
var (min, minind) := (real.MaxValue, 0);
for var i:=0 to a.Length-1 do
  if a[i]<min then
    (min, minind) := (a[i], i);
Result := (min, minind);
```

Сдвиг элементов

Сдвиг влево

Задача: Выполните в массиве сдвиг элементов влево

```
begin
  var a := new integer[5];
  a := arrRandomInteger(5); // [56,28,33,57,25]
  for var i := 0 to a.Length - 2 do
    a[i] := a[i + 1];
  a[a.Length - 1] := 0;
  print(a)// [28,33,57,25,0]
end.
```

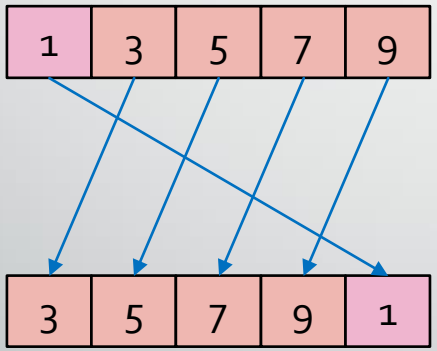


With slices:

a := a[1:] + Arr(0);

Циклический сдвиг влево

```
…
  var v := a[0];
  for var i:=0 to a.Length-2 do
    a[i] := a[i+1];
  a[a.Length-1] := v;
```

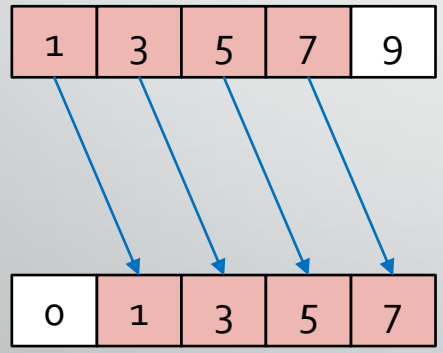


With slices:

a := a[1:] + a[:1];

Сдвиг вправо

```
…
  for var i:=a.Length-1 downto 1 do
    a[i] := a[i-1];
  a[0] := 0;
```

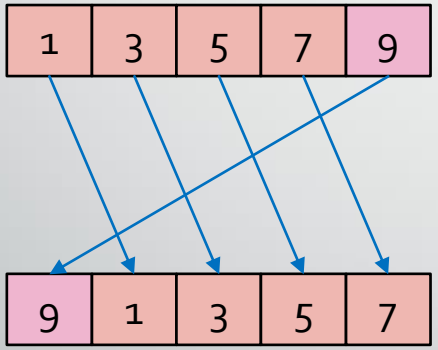


Using slices:

a := Arr(0) + a[:a.Length-1];

Циклический сдвиг вправо

```
…
var v := a[a.Length-1];
for var i:=a.Length-1 downto 1 do
  a[i] := a[i-1];
a[0] := v;
```



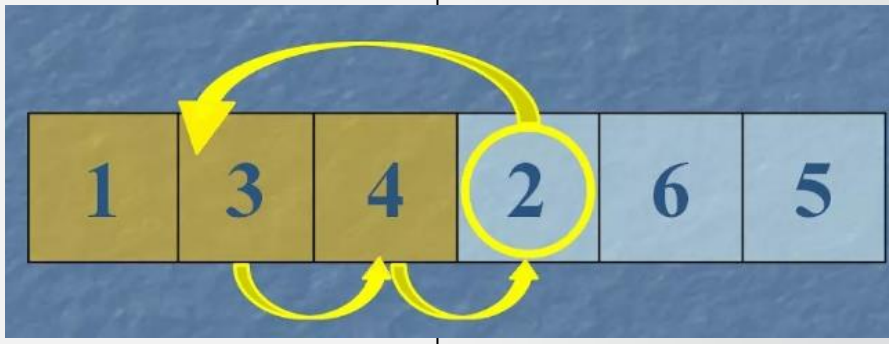
Using slices:

```
var m := a.Length-1;
a := a[m:] + a[:m];
```

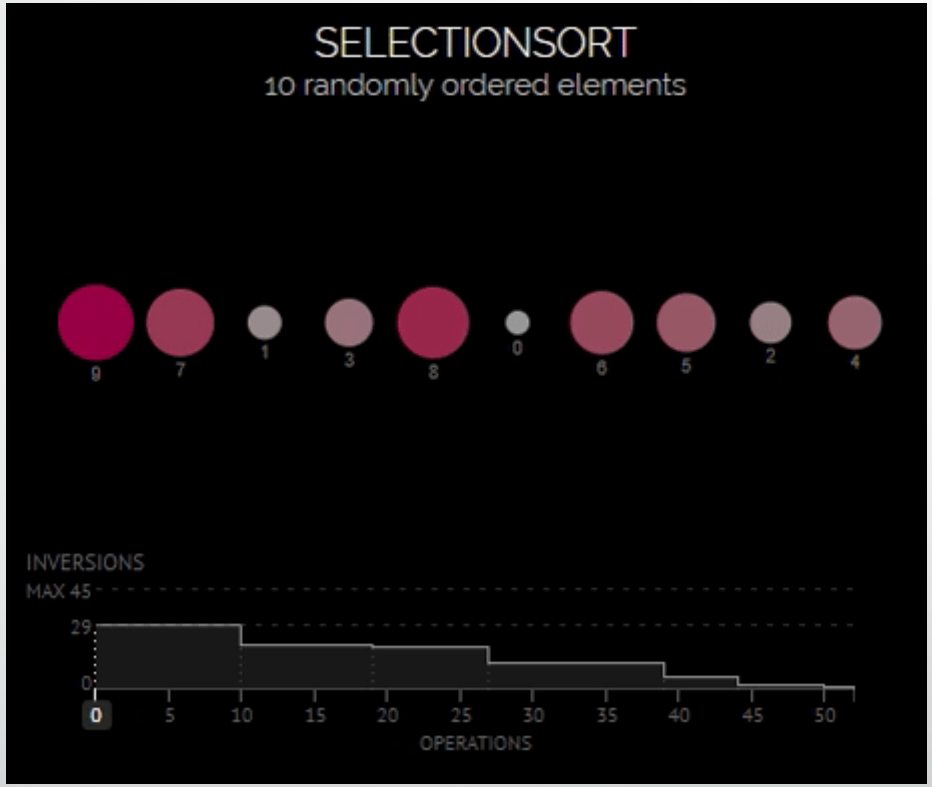
Алгоритмы сортировки массива

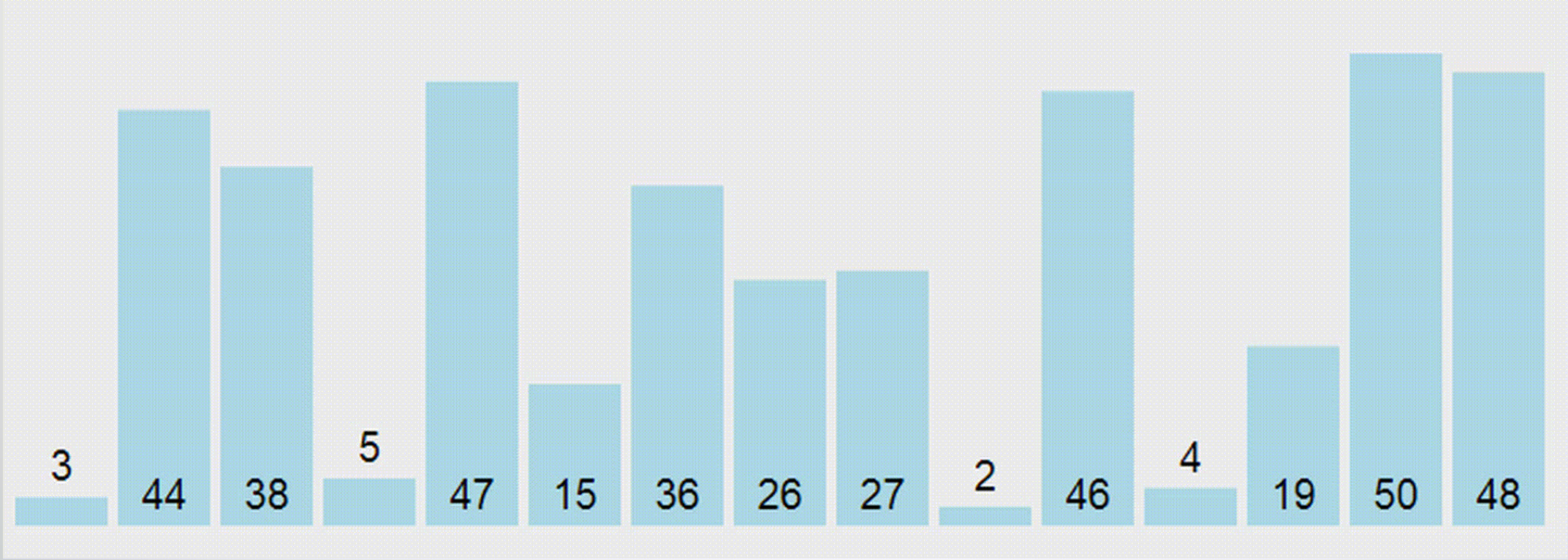
Сортировка обменом

```
…
  for var i := 0 to a.High-1 do
  begin
    var (min,imin) := (a[i],i);
    for var j := i + 1 to a.High do
      if a[j] < min then
        (min,imin) := (a[j],j);
    Swap(a[imin],a[i]);
  end;
```



- Алгоритм проходится по массиву снова и снова, вставляя элемент на правильную позицию
- В каждой итерации мы находим минимальный элемент и вставляем его на нужное место, меняя местами с тем элементом, который там находится
- Таким образом на i-й итерации, первые i элементов расположатся на свои места.


SELECTIONSORT
10 randomly ordered elements
9
7
1
3
8
0
6
5
2
4
INVERSIONS
MAX 45
29
0
0
5
10
15
20
25
30
35
40
45
50
OPERATIONS

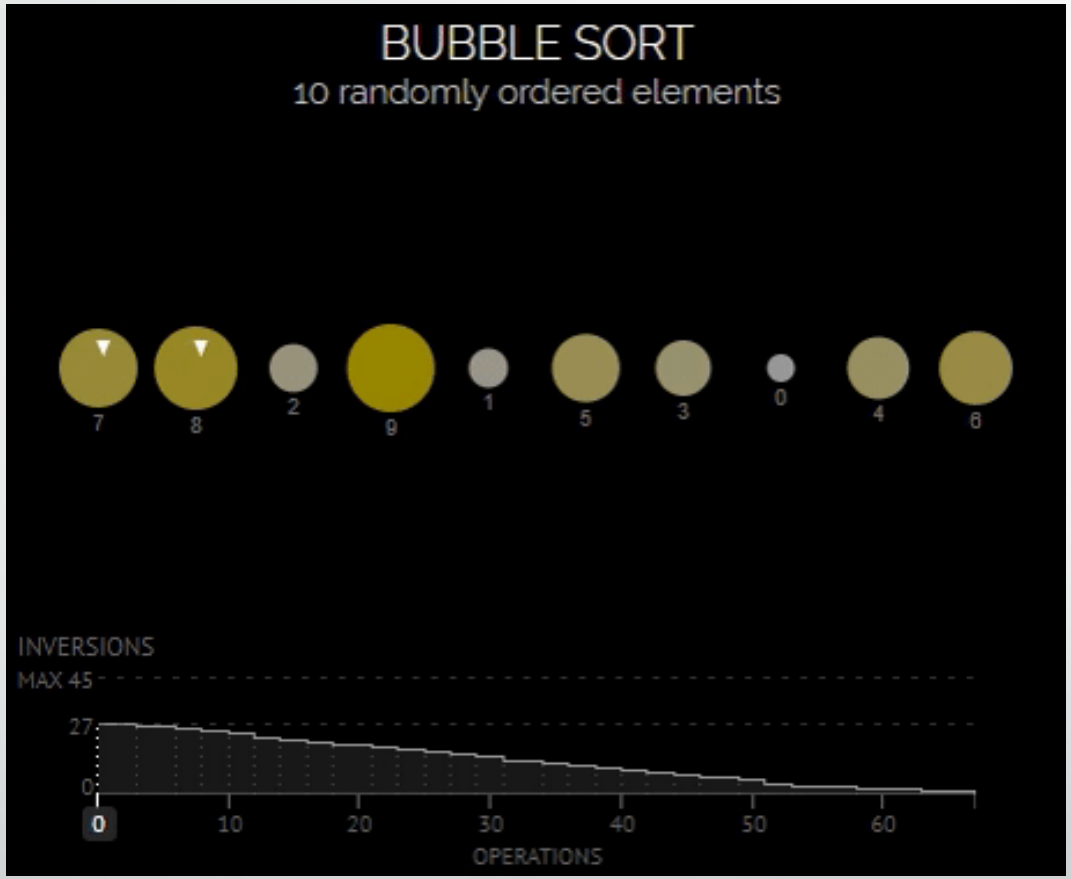


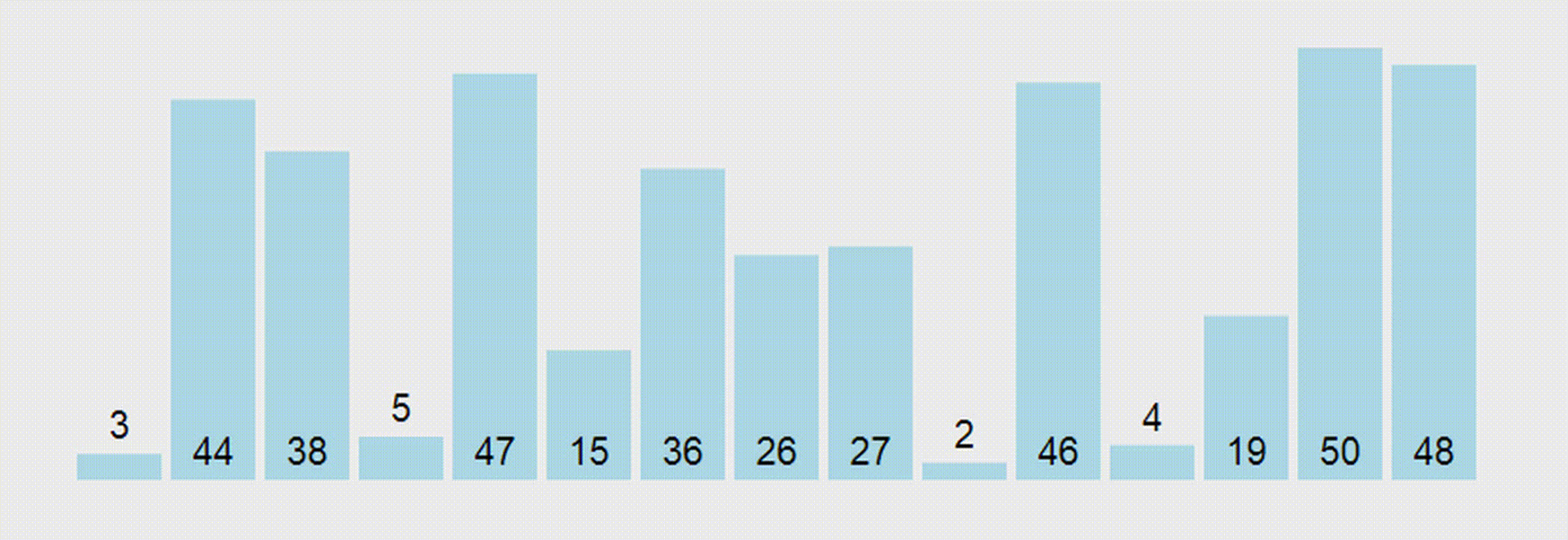

```
…
  for var i := 0 to a.High-1 do
  begin
    var (min,imin) := (a[i],i);
    for var j := i + 1 to a.High do
      if a[j] < min then
        (min,imin) := (a[j],j);
    Swap(a[imin],a[i]);
  end;
```

Сортировка методом «Пузырька»

```
...
   for var i := 0 to arr.High - 1 do
     for var j := arr.High - 1 downto i do
       if arr[j] > arr[j + 1] then
         Swap(arr[j], arr[j + 1]);
```

- Будем перебирать массива слева направо.
- Если текущий элемент больше следующего – меняем их.
- Делаем это до тех пор, пока массив не отсортируется.
- Заметьте, то после первой итерации самый большой элемент будет в конце массива, как и надо.
- После двух итераций, два самых больших элемента будут на правильном месте; и т.д..


BUBBLE SORT
10 randomly ordered elements
7
8
2
9
1
5
3
0
4
6
INVERSIONS
MAX 45
27
0
0
10
20
30
40
50
60
OPERATIONS




```
…
for var i := 1 to arr.High - 1 do
    for var j := 0 to arr.High - i do
      if arr[j] > arr[j + 1] then
        Swap(arr[j], arr[j + 1]);
```

Сортировка Пузырьком 2

```
…
  var i := a.High;
  var q: boolean;
  repeat
    q := true;
    for var j := 0 to i - 1 do
      if a[j+1] < a[j] then
      begin
        Swap(a[j+1], a[j]);
        q := false;
      end;
    i -= 1;
  until q;
```

Стандартная процедура sort()

```
Sort(a);
SortByDescending(a);
```

Объединение двух отсортированных массивов

Объединение двух отсортированных массивов

Задача. Даны два отсортированных массива a и b.

Необходимо объединить их в третий отсортированный массив

Решение (неэффективное):

```
var c := a + b;
Sort(c);
```

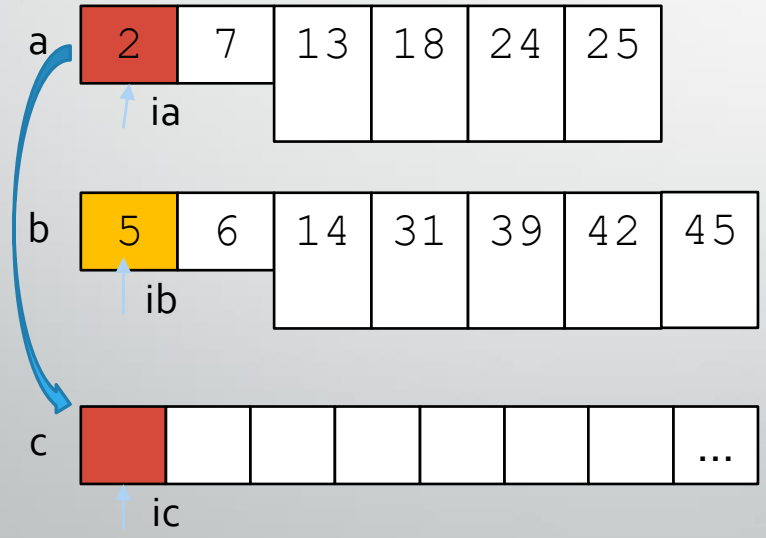
```
begin
  var a:=arr(1,3,11,19);
  var b:=arr(1,13,21);
  var c:=a+b; // [1,3,11,19,1,13,21]
  Sort(c); //[1,1,3,11,13,19,21]
end.
```

Объединение двух отсортированных массивов

Задача. Даны два отсортированных массива a и b.

Необходимо объединить их в третий отсортированный массив

Решение (частями). Будем использовать счетчик **ia** как индекс первого элемента массива **a** и счетчик **ib** – как индекс первого элемента **b**. Если **a[ia]<b[ib]**, тогда копируем **a[ia]** в **c** и увеличиваем счетчик **ia**. Иначе копируем **b[ib]** и увеличиваем **ib**.

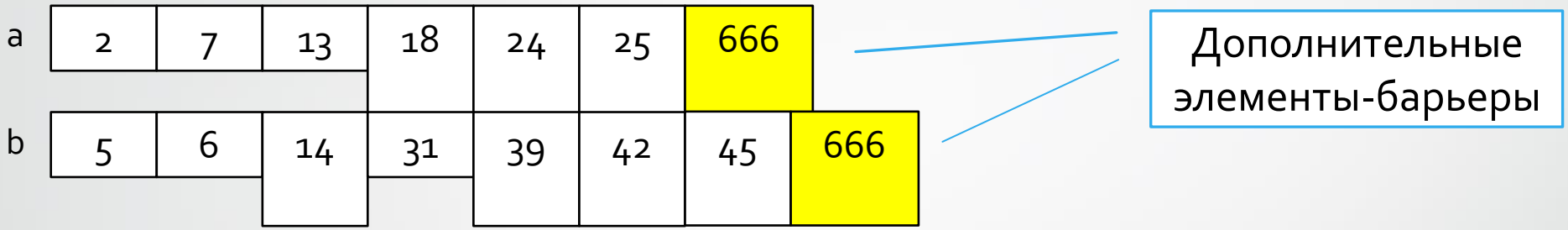


Это не работает! Почему?

```
var (ia,ib) := (0,0);
for var ic:=0 to c.Length-1 do
  if a[ia]<b[ib] then
  begin
    c[ic] := a[ia];
    ia += 1
  end
  else
  begin
    c[ic] := b[ib];
    ib += 1
  end;
```

Объединение двух отсортированных массивов

Решение (полное). Добавим элемент-барьер в конец каждого массива^



Предыдущий код будет работать!

Объединение двух отсортированных массивов

```
function Merge(a, b: array of integer; n, m: integer): array of real;
begin
  Assert((0 < n) and (n < a.Length));
  Assert((0 < m) and (m < b.Length));
  a[n] := integer.MaxValue; // барьер
  b[m] := integer.MaxValue; // барьер
  SetLength(Result, m + n);
  var (ia, ib) := (0, 0);
  for var ic := 0 to n + m - 1 do
    if a[ia] < b[ib] then
    begin
      Result[ic] := a[ia];
      ia += 1;
    end
    else
    begin
      Result[ic] := b[ib];
      ib += 1;
    end;
end;
```

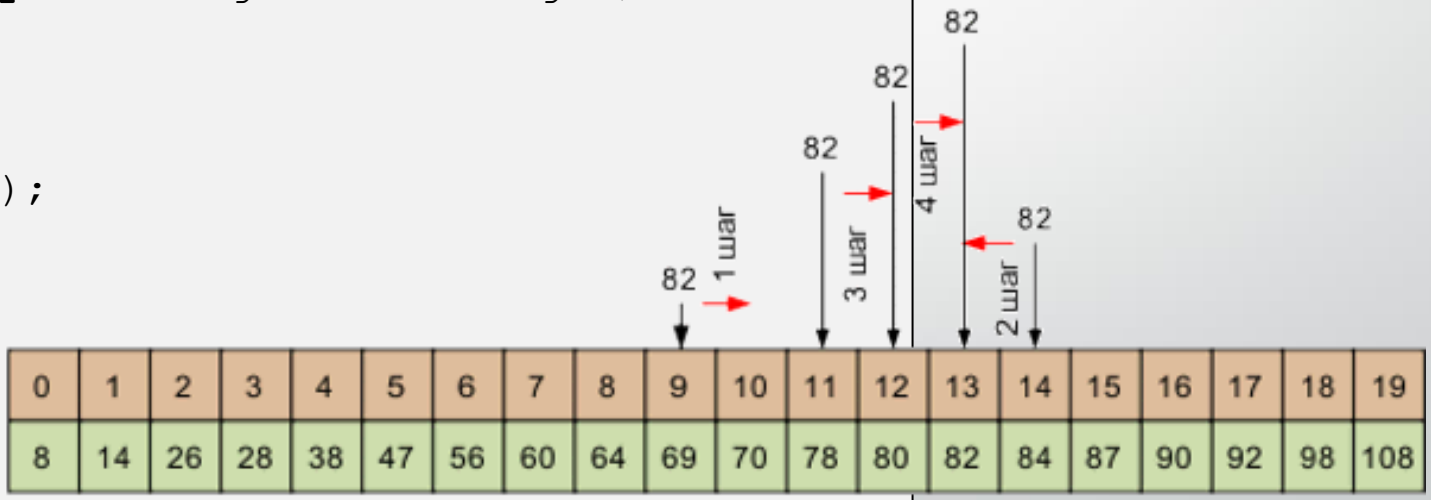
```
begin
  var a := arr(1, 3, 11, 19);
  var b := arr(1, 13, 21);
  setLength(a, 5); // для дополнительного элемента
  setLength(b, 4); // для дополнительного элемента
  print(Merge(a, b, 4, 3)) // [1,1,3,11,13,19,21]
end.
```

Бинарный поиск в отсортированном массиве

Стандартный метод **a.BinarySearch(x)**

Код метода (не рассматривать).

```
function BinarySearch(a: array of integer; x: integer):
integer;
begin
  var k: integer;
  var (l,r) := (0, a.Length-1);
  repeat
    k := (l+r) div 2;
    if x>a[k] then
      l := k+1
    else r := k-1;
  until (a[k]=x) or (l>r);
  Result := a[k]=x ? k : -1;
end;
```



Q & A