Информатика

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ. МАССИВЫ

ОСНОВНЫЕ СВЕДЕНИЯ ОБ АЛГОРИТМАХ

11 класс

ИЗДАТЕЛЬСТВО БИНОМ

Ключевые слова

- массив
- размерность массива
- описание массива
- типовые задачи обработки одномерных массивов за один просмотр
- сортировка массива: метод «пузырька», сортировка выбором

Массив

Массив – это поименованная совокупность однотипных элемен-тов, упорядоченных по индексам, определяющим положение элемента в массиве.

Массив в языке Pascal – это набор однотипных данных, причём количество этих данных фиксировано и определяется при описании массива. Все переменные, входящие в массив, имеют одно и то же имя – имя массива, а различаются они по индексу – номеру (месту) в массиве.

Массив **Result**:

Индекс	0	1	2	3	4	5	6	7
Значение элемента	90	100	84	72	45	92	45	53

Массив **Season**:

Индекс	1	2	3	4
Значение элемента	'зима'	'весна'	'лето'	'осень'

Описание массива

Pascal

Описание массива выглядит так:

array [<*тип индекса*>] **of** <*тип компонент*>

Здесь:

- **array** и **of** – служебные слова («массив» и «из»);
- <*тип индекса*> – описание индексации компонент (элементов) массива;
- <*тип компонент*> – тип величин, составляющих массив.

Упражнение. Запишите описание массива, ориентируясь на его назначение.

Массив с фамилиями учащихся 11-го класса (всего 25 учащихся).

```
const n=25;
var Name: array [1 .. n] of string;
```

Типовые задачи обработки одномерных массивов

- Поиск элементов с заданными свойствами
- Поиск максимумов и минимумов
- Подсчёт элементов, удовлетворяющих услови
- Проверка массива на упорядоченность
- Удаление из массива элемента с индексом *k*
- Вставка в массив элемента на место с индекс
- Перестановка элементов в обратном порядке
- Сортировка массива. Метод «пузырька»
- Сортировка выбором

Последовательный поиск в неупорядоченном массиве

Пример 3. Имеется массив $A\ [1..n]$. Найти элемент массива, равный p.

В алгоритмах поиска существует два возможных варианта окончания их работы: поиск может оказаться удачным – заданный элемент найден в массиве и определено его месторасположение, либо поиск может оказаться неудачным – необходимого элемента в данном объёме информации нет.

Возможный **алгоритм решения**:

1. Установить $i = 1$.
2. Если $A[i] = p$, алгоритм завершил работу успешно.
3. Увеличить i на 1.
4. Если $i \leq n$, то перейти к шагу 2. В противном случае алгоритм завершил работу безуспешно.

Программа

Последовательный поиск в неупорядоченном массиве

Pascal

Пример 3. Имеется массив *A* [*1..n*]. Найти элемент массива, равный *p*.

```
const n=5;
var A: array [1..n] of integer;
      i, p: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for  i := 1 to n do
     read (A[i]);
  write ('Ввод p: ');
  readln (p);
  i:=1;
  while (i<=n) and (A[i]<>p) do
    i:=i+1;
  if  i=n+1  then writeln ('Искомого элемента в массиве нет')
             else writeln ('Искомый элемент A[', i, '] = ', A[i])
end.
```

Поиск максимумов и минимумов

Пример 4. Имеется массив *A* [*1..n*]. Найти элемент массива с наименьшим значением.

Алгоритм поиска элемента с наименьшим значением в неупорядоченном массиве:

1. Установить значение текущего минимума равным первому исследуе-мому элементу.
2. Установить счетчик равным *2*.
3. Если исследованы ещё не все элементы (*i<=n*), то перейти к шагу *4*, иначе алгоритм окончен (минимальный элемент равен min).
4. Если рассматриваемый элемент меньше, чем текущий минимум, то минимуму присвоить значение текущего элемента.
5. Перейти к следующему элементу (увеличить *i* на единицу).
6. Перейти к шагу *3*.

Программа

Поиск минимума

Pascal

Пример 4. Имеется массив *A* [*1..n*]. Найти элемент массива с наименьшим значением.

```
const n=5;
var A: array [1..n] of integer;
      i, min: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for i := 1 to n do
    read (A[i]);
  min := A[1];
  i := 2;
  while (i<=n) do
  begin
     if A[i] < min then min := A[i];
     I := i+1
  end;
  writeln ('Минимум=', min)
end.
```

Подсчёт элементов массива, удовлетворяющих некоторому условию

Зачастую бывает важно выяснить, сколько элементов, обладающих определённым свойством, содержится в массиве.

Пример 5. Имеется массив *A* [*1*..*n*]. Подсчитать количество элементов массива кратных некоторому числу *p*.

Алгоритм решения:

1. Присвоить нулевое значение переменной (счётчику), введённой для подсчёта количества элементов, удовлетворяющих заданному условию.

2. Организовать просмотр всех элементов массива: если просматри-ваемый элемент удовлетворяет заданному условию, значение счётчика увеличивать на *1*.

Программа

Подсчет элементов массива, удовлетворяющих некоторому условию

Pascal

Пример 5. Имеется массив *A* [*1..n*]. Подсчитать количество элементов массива кратных некоторого числа *p*.

```
const n=5;
var A: array [1..n] of integer;
      i, p, k: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for i := 1 to n do
     read (A[i]);
  writeln ('Ввод числа p:');
  readln (p);
  k := 0;
  for  i := 1 to n do
     if  A[i] mod p = 0 then k := k + 1;
  writeln ('k=', k)
end.
```

Проверка массива на упорядоченность

Пример 7. Имеется массив *A* [*1..n*]. Определить, упорядочены ли элементы массива по неубыванию, т. е. каждый элемент массива с 1-го по (n – 1)-й не больше последующего.

Алгоритм решения

Самый простой путь решения этой задачи – проверить, есть ли в массиве такие пары элементов, что *A*[*i*] > *A*[*i + 1*]. Если подобные пары элементов есть, то массив не упорядочен по неубыванию, а если таких пар нет, то – упорядочен.

В программе будем использовать логическую переменную *flag*:

- если *flag = true*, то массив упорядочен;
- если *flag = false*, то массив неупорядочен.

Программа

Проверка массива на упорядоченность

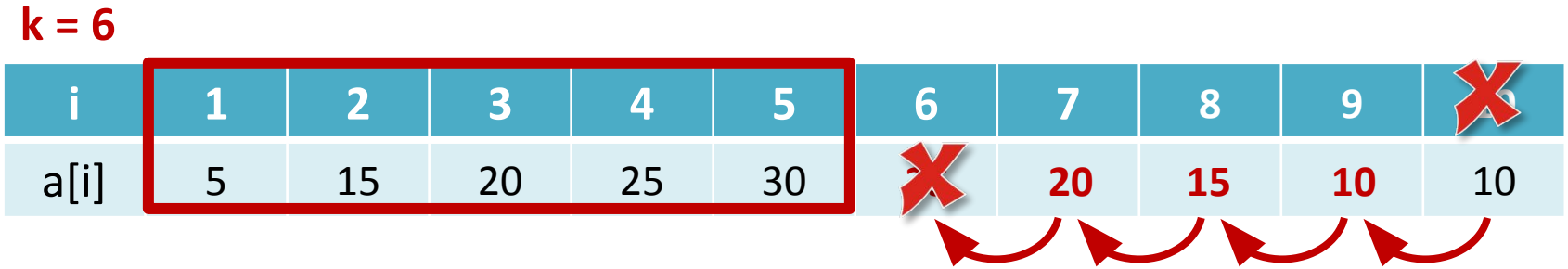
Pascal

Пример 7. Имеется массив *A* [*1..n*]. Определить, упорядочены ли элементы массива по неубыванию.

```
const n=5;
var A: array [1..n] of integer;
      i: integer;  flag: boolean;
begin
  writeln ('Ввод значений элементов массива:');
  for  i := 1 to n do
     read (A[i]);
  flag := true;
  for  i := 1 to n-1 do
     if a[i]>a[i+1] then flag:=false;
  if flag then writeln ('упорядочен')
          else writeln ('неупорядочен')
end.
```

Удаление из массива элемента с индексом k

Пример 8. Имеется массив a[1..n]. Удалить элемент с индексом *k*.



При удалении из массива любого из элементов размерность массива уменьшается на 1.

Мы видим, что элементы с индексами от *1* до *k – 1* не изменились.

На место элемента с индексом *k* (*6*) переместился элемент, имевший индекс *k + 1* (*7*), на место элемента с индексом *k + 1* (*8*) переместился элемент, имевший индекс *k + 2* (*8*) и т. д.

Фрагмент программы удаления из массива элемента с индексом k и последующим сдвигом всех расположенных справа от него элементов на одну позицию влево имеет вид:

```
for  i := k to n-1 do
     A[i] := A[i+1];
```

Программа

Удаление из массива элемента с индексом k

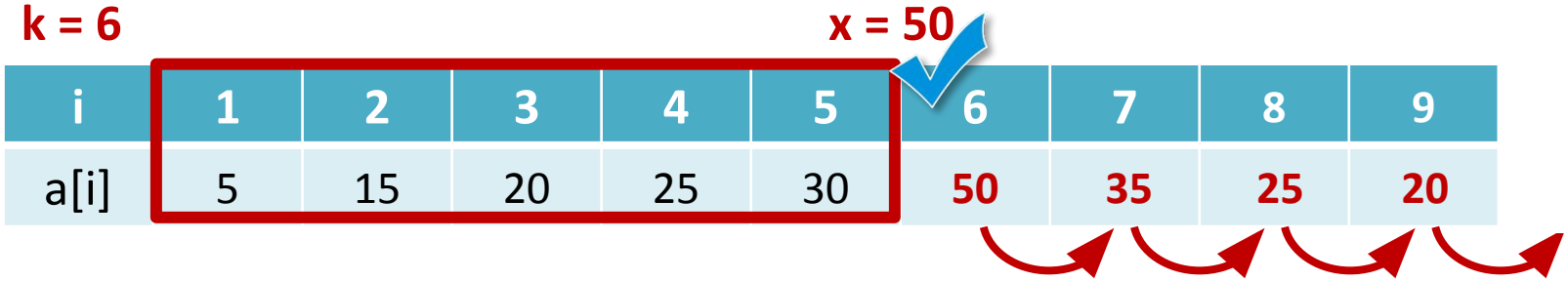
Pascal

Пример 8. Имеется массив *A* [*1..n*]. Удалить элемент с индексом *k*.

```
const n=10;
var A: array [1..n] of integer;
      i, k: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for i := 1 to n do
     read (a[i]);
   write ('Ввод индекса k: ');
   readln (k);
   for   i := k to n-1
         do A[i] := A[i+1];
 writeln('Массив после обработки:');
  for  i := 1 to n-1 do
     write (A[i], ' ')
end.
```

Вставка элемента на место с индексом *k*

Пример 9. Добавить в массив элемент *X* на место с индексом *k*.



i	1	2	3	4	5	6	7	8	9
a[i]	5	15	20	25	30	50	35	25	20

При вставке в массив ещё одного элемента размерность массива увеличивается на *1*. Это надо учесть при описании массива.

Элементы с индексами от *1* до *k – 1* не изменились.

На место элемента с индексом *k* (***6***) должен переместиться элемент, имевший индекс *k + 1* (***7***), на место элемента с индексом *k + 1* (***8***) – элемент, имевший индекс *k + 2* (***8***) и т. д. Поскольку при присваивании нового значения элементу старое пропадает, *замену надо производить с конца*. После чего заменить значение элемента с индексом *k*.

Фрагмент программы:

```
for i :=n downto k+1 do A[i] := A[i-1];
A[k] := X;
```

Программа

Вставка элемента на место с индексом *k*

Pascal

Пример 9. Добавить в массив элемент *X* на место с индексом *k*.

```
const n=10;
var A: array [1..n] of integer;
      i, k, X: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for  i := 1 to n-1 do
     read (A[i]);
  write ('Ввод индекса k: '); readln (k);
  write ('Ввод числа X: '); readln (X);
  for i:=n downto k+1 do
      A[i] := A[i-1];
  A[k] := X;
  writeln('Массив после обработки: ' );
  for i:=1 to n do
     write (A[i], ' ')
end.
```

Перестановка всех элементов массива в обратном порядке

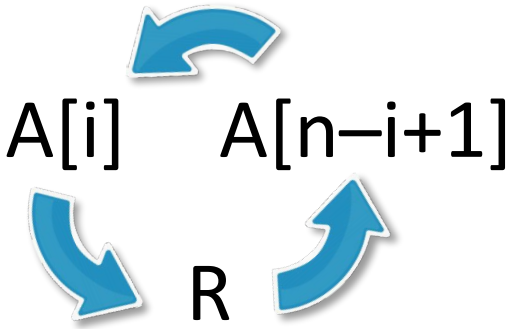
Пример 10. Имеется массив *A* [*1..n*]. Перевернуть его, т.е. что поменять местами 1-й и последний элементы, 2-й и предпоследний и т. д.

i	1	2	3	4	5	6	7
a[i]	40	35	30	25	20	15	5

В общем случае, меняются местами элементы *A*[*i*] и *A*[*n – i + 1*].

Фрагмент программы по перестановке в обратном порядке всех элементов массива:

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



Программа

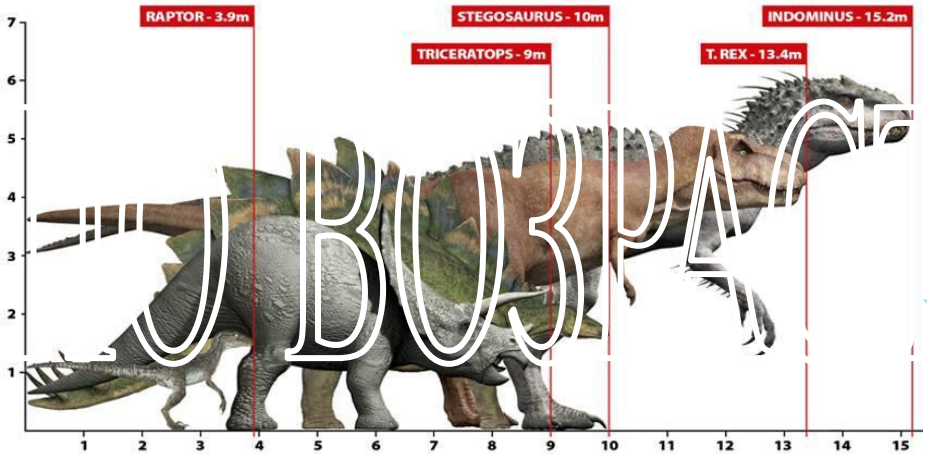
Перестановка всех элементов массива в обратном порядке

Pascal

Пример 10. Имеется массив *A* [*1..n*]. Перевернуть его.

```
const n=7;
var A: array [1..n] of integer;
      i, r: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for  i := 1 to n do
    read (A[i]);
  for  i := 1 to n div 2 do
  begin
    R := A[i];
    A[i] := A[n-i+1];
    A[n-i+1] := R
  end;
  writeln ('Массив после обработки:');
  for  i := 1 to n do  write (A[i], '  ')
end.
```

Сортировка массива



Сортировка – это распределение элементов массива в соответствии с определёнными правилами.

Сортировка методом «пузырька»

Своё название алгоритм получил благодаря следующей ассоциации: если сортировать этим алгоритмом массив по неубыванию, то максимальный элемент «тонет», а «лёгкие» элементы поднимаются на одну позицию к началу массива на каждом шаге алгоритма.

Пусть *n*– количество элементов в неупорядоченном массиве.

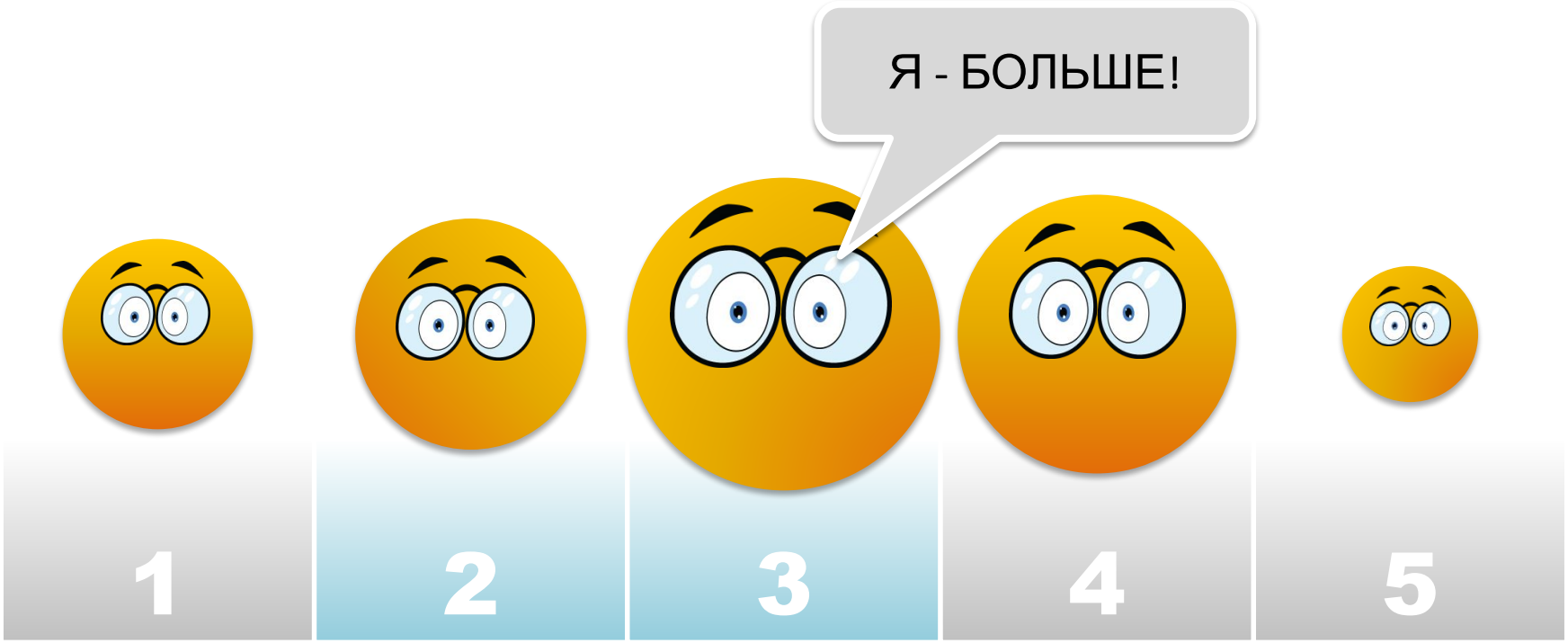
1. Поместим на место *n*-го элемента наибольший элемент массива. Для этого:

   1) положим *i = 1*;

   2) пока не обработана последняя пара элементов: сравниваем *i*-й и (*i + 1*)-й элементы массива; если *A*[*i*] > *A*[*i + 1*] (элементы расположены не по порядку), то меняем элементы местами; переходим к следующей паре элементов, сдвинувшись на один элемент вправо.

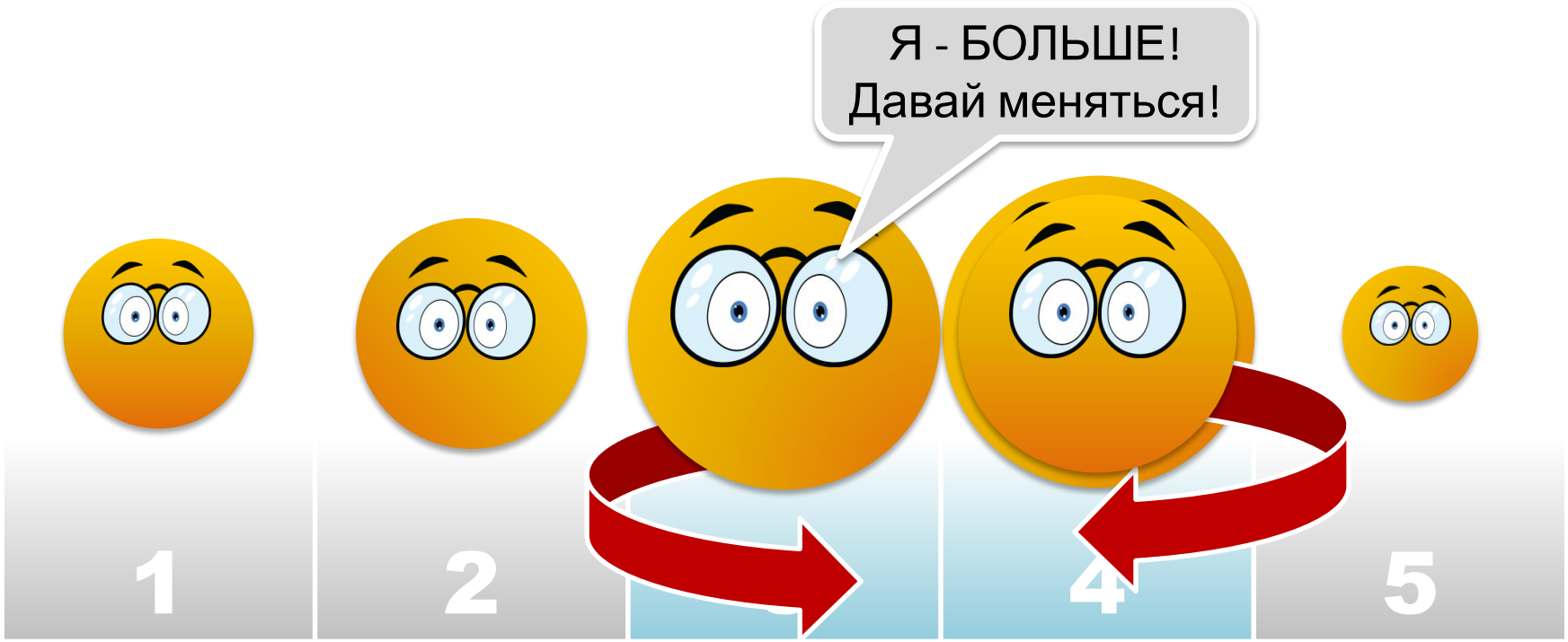
2. Повторяем пункт *1*, каждый раз уменьшая размерность неупорядо-ченного массива на *1*, до тех пор, пока не будет обработан массив из одной пары элементов (таким образом, на *k*-м просмотре будут сравниваться первые (*n – k*) элементов со

Сортировка методом «пузырька»



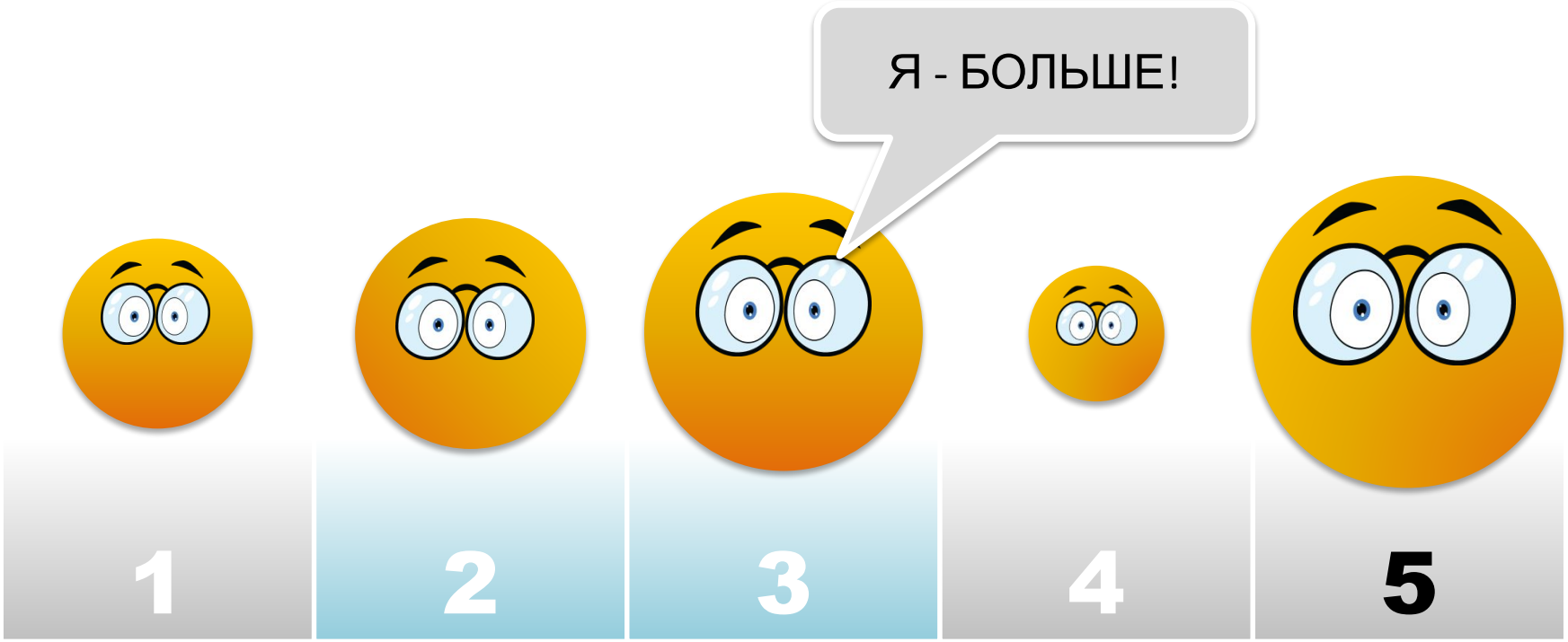

```
If A[i] > A[i+1]  then
begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
```

Сортировка методом «пузырька»




```
If A[i] > A[i+1]  then
begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
```

Сортировка методом «пузырька»




```
for i := 1  to 3  do
      If A[i] > A[i+1]  then
      begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
```

Сортировка методом «пузырька»

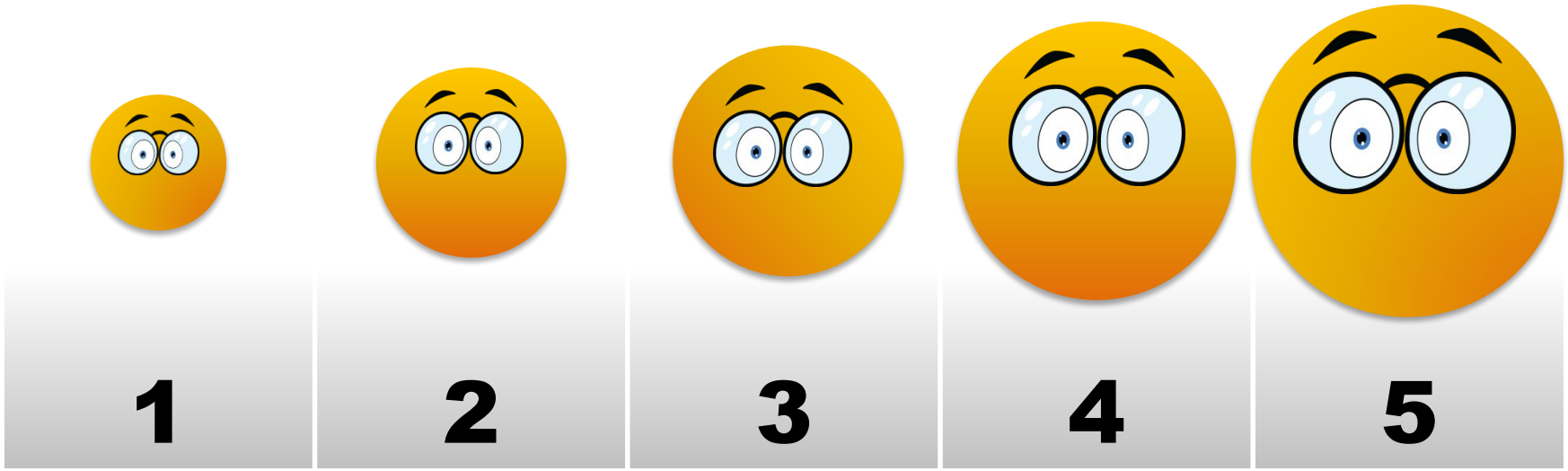

```
for i := 1  to 2  do
      If A[i] > A[i+1]  then
      begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
```

Сортировка методом «пузырька»


```
for i := 1  to 2  do
      If A[i] > A[i+1]  then
      begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
```

Сортировка методом «пузырька»

n = 5




```
for k := n-1  downto  1  do
      for i := 1  to k  do
            If A[i] > A[i+1]  then
            begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
```

Программа

Сортировка методом «пузырька»

Pascal

Пример 10. Отсортировать массив *A* [*1..n*] по возрастанию

Обратите внимание, что при каждой итерации внешнего цикла (с параметром *k*), не только один из элементов ставится на место, но и происходит частичное упорядочивание других элементов массива.
Подумайте, как можно улучшить алгоритм.

```
for k := n-1  downto  1  do
   for i := 1  to k  do
      If A[i] > A[i+1]  then
      begin  R := A[i];  A[i] := A[i+1];  A[i+1] := R end;
writeln ('Массив после обработки:');
for i := 1 to n do
   write (A[i], ' ')
end.
```

Сортировка выбором

Сортировка выбором (в порядке неубывания) осуществляется следующим образом:

1. В массиве выбирается минимальный элемент.
2. Минимальный и первый элементы меняются местами (первый элемент считается отсортированным).
3. В неотсортированной части массива снова выбирается минимальный элемент и меняется местами с первым неотсортированным элементом массива.
4. Действия, описанные в пункте 3, повторяются с неотсортированными элементами массива до тех пор, пока не останется один неотсортированный элемент (его значение будет

Попробуйте записать программу самостоятельно

Программа

Сортировка выбором

Pascal

Пример 10. Отсортировать массив *A* [*1..n*] по возрастанию.

```
const n=10;
var A: array [1..n] of integer;
      i, j, imin, R: integer;
begin
  writeln ('Ввод значений элементов массива:');
  for  i := 1 to n do read (A[i]);
  for i:=1 to n-1 do
  begin
      imin := i;
      for  j := i+1 to n do
         if  A[j] < A[imin] then imin:=j;
      R := A[i];  A[i] := A[imin];  A[imin] := R;
  end;
  writeln ('Отсортированный массив:');
  for  i := 1 to n do  write(A[i],'  ');
end.
```

Самое главное

Из элементов простых типов в языке Pascal можно образовывать составные типы данных (структуры данных). Примером таких структур являются одномерные массивы.

Массив в языке Pascal – это набор однотипных данных, причём количество этих данных фиксировано и определяется при описании массива. Все переменные, входящие в массив, имеют одно и то же имя – имя массива, а различаются они по индексу – номеру (месту) в массиве.

Перед использованием в программе массив должен быть описан, т. е. должно быть указано имя массива, количество элементов массива и их тип. Это необходимо для того, чтобы выделить в памяти под массив блок ячеек нужного типа.

Чаще всего массив обрабатывается в цикле for. Но при работе с массивами можно использовать и другие циклы.

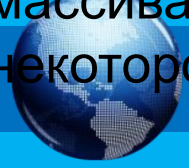
Самое главное

К типовым задачам обработки одномерных массивов, решаемым в процессе их однократного просмотра, относятся:

- задачи поиска элемента с заданными свойствами, в том числе максимумов и минимумов;
- проверка соответствия элементов массива некоторому условию (подсчёт количества или суммы элементов, удовлетворяющих некоторому условию; проверка массива на упорядоченность и др.);
- задачи на удаление и вставку элементов массива;
- задачи на перестановку всех элементов массива в обратном порядке

  и т. д.

Сортировка – один из наиболее распространённых процессов современной обработки данных. Под **сортировкой** (упорядочением) массива понимают перераспределение значений его элементов в некотором определённом порядке.



Информационные источники

- http://www.emu.dk/sites/default/files/boeger.jpg
- http://method-alfa.ru/ma/112.png
- http://stavka-nomer1.ru/photos/kalendar-izmeneniya-pogody-dlya-yasley-detskogo-sada-9563-large.jpg
- https://vertex-club.ru/upload/iblock/74f/74fcaeaa76602c56566253b269aed9e0.jpg
- http://3.bp.blogspot.com/-Fhpq8kknfLI/VXln4DBuKeI/AAAAAAAAEcE/4W9MklPvquI/s1600/indominus-t-rex-size-compare-chart.jpg
- https://ssec.si.edu/sites/default/files/ThinkstockPhotos-519386131.jpg
- http://omyworld.ru/wp-content/uploads/2012/11/highest-skyscrapers-of-the-world_2012-1.jpg
- http://iq230.com/images/sampledata/1/teacher-desk.jpg
- http://gamelion.ucoz.ru/photo/