

Основные вопросы:

1. Формирование нового массива.
2. Сортировка массива линейным методом и методом пузырька.
3. Проверка упорядоченности.
4. Составление программы для вычислений в одномерном массиве (теоретическая часть).

Первый вопрос: Формирование нового массива.

Заполнение массива.

1 Способ (заполнение с клавиатуры. Динамический ввод данных)

```

Var
  M:array[1..10] of integer;
  I: byte;
Begin
  For I:=1 To 10 Do Begin
    Write('Введите ',I,' значение ');
    ReadLn(M[I]);
  End;
End.

```

2 Способ (с использованием генератора случайных чисел)

```

Var
  M: array[1..25] of integer;
  I: byte;
Begin
  For I:=1 To 25 Do Begin
    M[I]:=Random(50);
    Write(M[I]:4);
  End;
End.

```

3 Способ (статический ввод данных)

```

Const
  M: array[1..12] of integer = (31,28,31,30,31,30,31,31,30,31,30,31);
Var
  I: Integer;
Begin
  For I:=1 To 9 Do
    Write(M[I]:3);
End.

```

Алгоритмы формирования одномерного массив.

1. Дан двумерный массив размерностью 5x6, заполненный целыми числами. Сформировать одномерный массив, каждый элемент которого соответственно равен сумме элементов строк. Оба массива вывести на экран.

USES Crt;

VAR

H:ARRAY[0..5,0..6] OF INTEGER; K:ARRAY[0..5] OF INTEGER;

I,J:BYTE; S:INTEGER;

BEGIN

ClrScr; Randomize;

WriteLn(' Значения двумерного массива');

FOR I:=1 TO 5 DO Begin

FOR J:=1 TO 6 DO Begin

H[I,J]:=Random(23); Write(H[I,J]:3);

End; WriteLn;

End; WriteLn;

{ Нахождение суммы элементов строк и заполнение одномерного массива }

WriteLn(' Значения одномерного массива');

FOR I:=1 TO 5 DO Begin

S:=0;

FOR J:=1 TO 6 DO

S:=S+H[I,J];

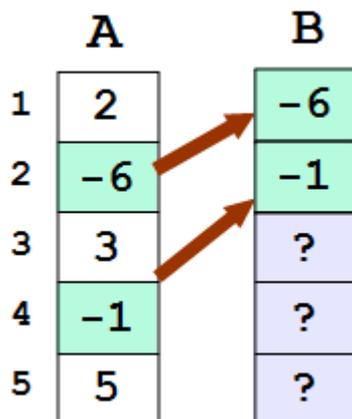
K[I]:=S; Write(K[I]:4);

End;

END.

Выбор элементов и сохранение в другой массив.

Пример: найти в массиве элементы, удовлетворяющие некоторому условию (например, отрицательные), и скопировать их в другой массив.



Решение:

Решение: подсчитывать количество найденных элементов с помощью счетчика count, очередной элемент устанавливать на место B. Переменной count необходимо присвоить 1 .

```

for i:=1 to N do
  if (A[i] < 0) then begin
    B[ count ] := A[i];
    count:=count+1;
  end;

```

Вывод массива B:

```
writeln("Выбранные элементы"); for i:=1 to count-1 do write(B[i], " ")
```

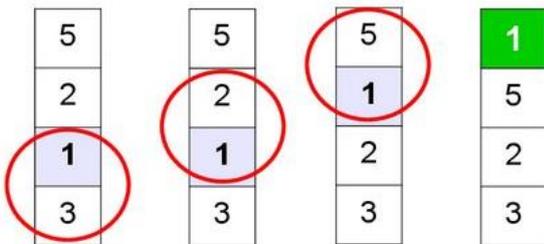
Второй вопрос: Сортировка массива линейным методом и методом пузырька.

Метод пузырька

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

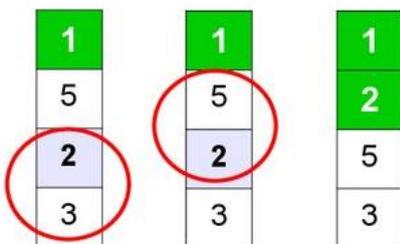
Для массивов – самый маленький («легкий» элемент перемещается вверх («всплывает»)).

1-ый проход

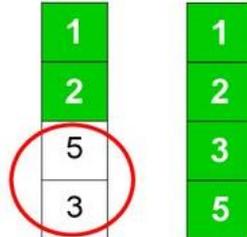


- начиная снизу, сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву **один** элемент (самый маленький) становится на свое место

2-ой проход



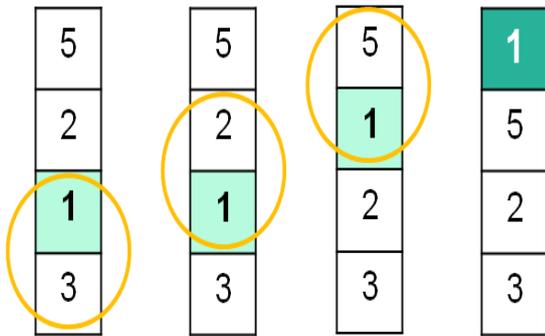
3-ий проход



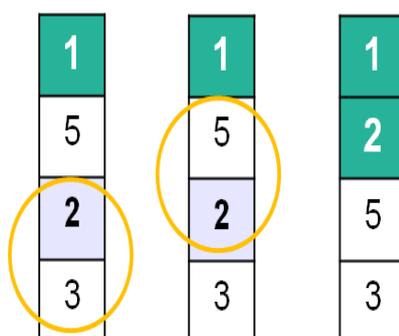
Для сортировки массива из N элементов нужен N-1 проход (достаточно поставить на свои места N-1 элементов).

- При сортировке методом «пузырька» массив представляется в виде воды, маленькие элементы — пузырьки в воде, которые всплывают вверх (самые легкие).
- При первой итерации цикла элементы массива попарно сравниваются между собой: предпоследний с последним, пред предпоследний с предпоследним и т.д. Если предшествующий элемент оказывается больше последующего, то производится их обмен.
- При второй итерации цикла нет надобности сравнивать последний элемент с предпоследним. Последний элемент уже стоит на своем месте, он самый большой. Значит, число сравнений будет на одно меньше. То же самое касается каждой последующей итерации.

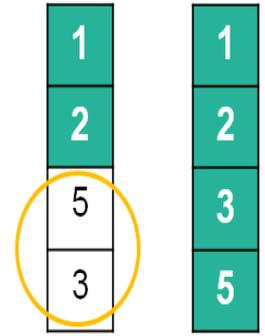
1-ая итерация



2-ая итерация



3-я итерация



Выполнение на Паскале:

```

for i: = 1 to N- 1
do begin
  for j: = N- 1 downto i do
    if A[ j ] > A[ j+ 1 ] then
      begin
        c := A[ j ] ;
        A[ j ] := A[ j+ 1 ] ;
        A[ j+ 1 ] := c;
      end ;
end ;

```

Метод "пузырька"

Текст упорядочения массива $M[1..N]$ на языке программирования Pascal:

```
begin
  for j:=1 to N-1 do
    for i:=1 to N-j do
      if M[i] > M[i+1] then
        begin
          t:= M[i];
          M[i]:= M[i+1];
          M[i+1]:= t;
        end;
      end;
    end;
  end;
```

Задание Array 12.

Заполнить массив из 10 элементов случайными числами в интервале и отсортировать первую половину массива по возрастанию, а вторую – по убыванию (методом 'Пузырька').

Пример: Исходный массив: 14 25 13 30 76 58 32 11 41 97

Результат: 13 14 25 30 76 97 58 41 32 11

```
begin
  c:= A[ i ] ;
  A[ i ] := A[ min ] ;
  A[ min ] := c;
  end ;
  for i:= 1 to N-1 do
    begin
      min:= i ;
      for j:= i+1 to N do
        if A[j] < A then
          min:=j;
      if min <> i then
        begin c:=A[i]; A[i]:=A; A:=c;
        end;
      end;
    end;
```

Сортировка массива линейным методом (простым выбором)Сортировка простым выбором.

```

Program Simpl_Sort;
const n = 10;
var a : array [1 .. n] of integer;
      i, j, k, x : integer;
begin
  writeln('Введите ', n, ' элементов массива');
  for i := 1 to n do read(a[i]);

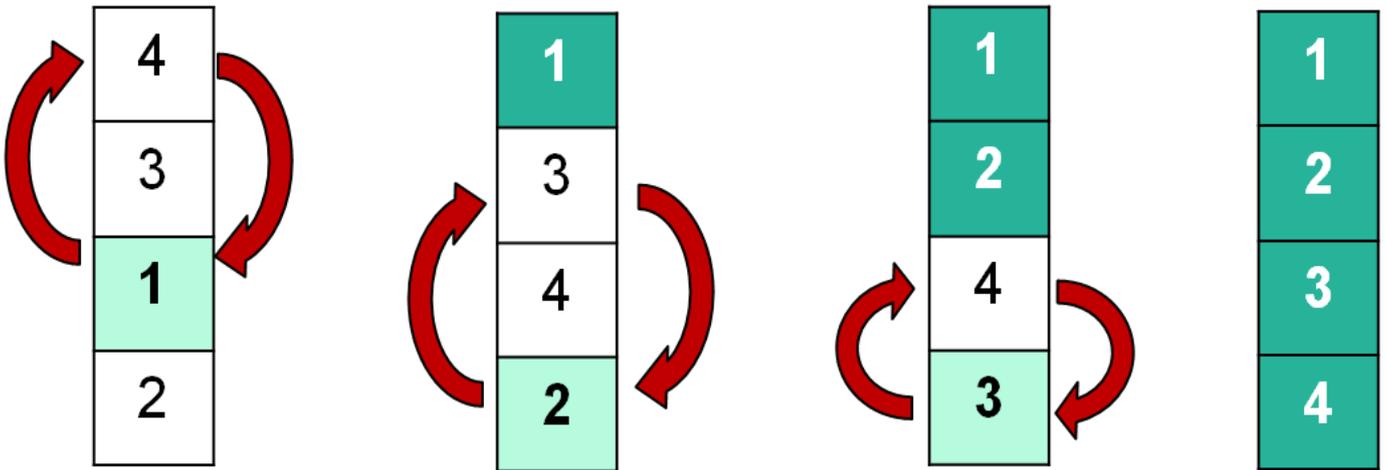
  for i := 1 to n - 1 do begin
    k := i; x:=a[i];
    for j := i + 1 to n do
      if a[j] < x then begin k := j; x:=a[k]; end;
      a[k] := a[i]; a[i] := x;
    end;
    . . .

  writeln('Упорядоченный массив:');
  for i := 1 to n do write(a[i]:5)
end.

```

Сортировка методом выбора:

- в массиве ищется минимальный элемент и ставится на первое место (меняется местами с A);
- среди оставшихся элементов также производится поиск минимального, который ставится на второе место (меняется местами с A) и т.д.



Задание Array 13: Заполнить массив из 10 элементов случайными числами в интервале и отсортировать его по возрастанию суммы цифр

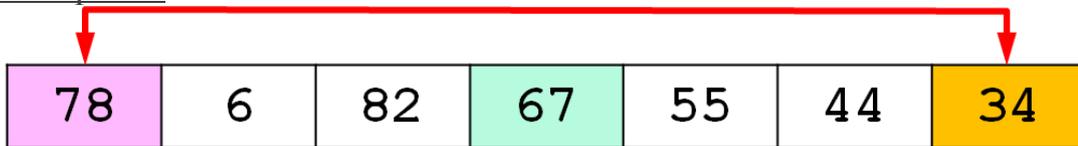
Пример:

Исходный массив: 14 25 13 12 76 58 21 87 10 98

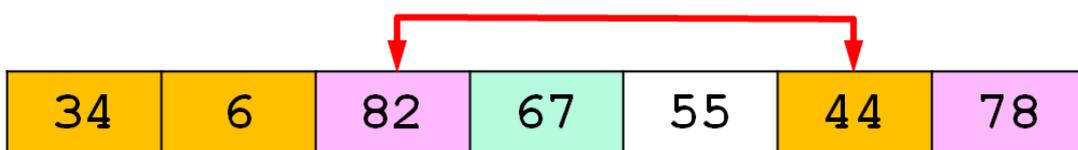
Результат: 10 21 12 13 14 25 76 58 87 98

Быстрая сортировка или quick sort

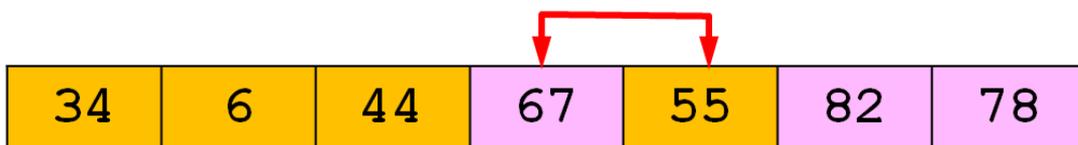
Алгоритм:



L R



L R



L R



R L

L стало больше R

Выполнение на Паскале:

```
procedure QSort ( first, last: integer);  
var L, R, c, X: integer;  
begin  
  if first < last then begin  
    X:=A[(first+last) div 2];  
    L:=first; R:=last;  
    while L <= R do begin  
      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(first, R);    QSort(L, last);  
  end;  
end.
```

для рекурсии

Делаем
обмен

идем дальше

сортировка двух
частей

Сортировка по возрастанию массива A из N целых чисел включением с линейным поиском.

```

Program Sort_Include1;
var A: array[1..100] of integer; N,i,k,x : integer;
begin
  write('количество элементов массива ');
  read(N);
  read(A[1]); {for i:=1 to n do read(A[i]);}
  {k - количество элементов в упорядоченной части массива}
  for k:=1 to n-1 do
    begin
      read(x); {x:=A[k+1];}
      i:=k;
      while (i>0)and(A[i]>x) do
        begin
          A[i+1]:=A[i];
          i:=i-1;
        end;
        A[i+1]:=x;
      end;
    for i:=1 to n do write(A[i], ' '); {упорядоченный массив}
  end.

```

Третий вопрос: Проверка упорядоченности.

1. Массив упорядочен по возрастанию, если каждый следующий его элемент НЕ МЕНЬШЕ предыдущего.
2. Когда речь заходит о предыдущем/следующем элементе массива, подразумевается попарное сравнение (n)-го и (n+1)-го элемента.
3. Там, где присутствует индексация элементов счетчиком (n), подразумевается цикл.
4. Поскольку мы заранее знаем, сколько элементов содержится в массиве, используем цикл FOR.

Функция проверки упорядоченности массива является живой иллюстрацией теоремы: массив упорядочен, если упорядочена любая пара соседних элементов.

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:

```

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.

```

Четвертый вопрос: Составление программы для вычислений в одномерном массиве (теоретическая часть).

Накопление суммы

```
s:=0;
For i:=1 To n Do
  s:=s+a[i];
```

Накопление произведения

```
p:=1;
For i:=1 To n Do
  p:=p*a[i];
```

Перестановка элементов

```
n:=a[i];
a[i]:=a[i+1];
a[i+1]:=n;
```

Нахождение min (max) элемента и его порядковый номер

```
min:=a[i];
For i:=2 To n Do
  if min>a[i] Then Begin min:=a[i]; minn:=i; End;
```

Удаление элемента

x-номер удаляемого
элемента

```
For i:=1 To n Do
  If (i<>x) Then
    Writeln(a[i]:3);
```

или

```
FOR I:=x To n-1 Do
  a[i]:=a[i+1];
```

**Учитывай
ранг**
Сдвиги
меняют
ранг.

Сортировка элементов

```
For i:=1 To n-1 Do
  If a[i+1]<a[i] Then
    Begin
      stek:=a[i];
      a[i]:=a[i+1];
      a[i+1]:=stek;
      i:=0;
    End;
```

**Возврат
на
начало
массива**

Типовые задачи обработки одномерных массивов

- Поиск элементов с заданными свойствами
- Поиск максимумов и минимумов
- Подсчёт элементов, удовлетворяющих условию
- Проверка массива на упорядоченность
- Удаление из массива элемента с индексом k
- Вставка в массив элемента на место с индексом k
- 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:

```

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 ('Искомое 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 исследуемому элементу.
2. Установить счетчик равным 2.
3. Если исследованы ещё не все элементы ($i \leq n$), 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:

```

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 просматриваемый элемент удовлетворяет заданному условию, значение счётчика увеличивать на 1.

Программа:

```

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:

```

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 .

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	25	20	10	10

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

$k = 6$

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	25	20	10	10

Мы видим, что элементы с индексами от 1 до $k - 1$ не изменились.

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	25	20	10	10

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.

$k = 6$

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	35	25	20	10

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:

```

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	35	25	20	10

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

$k = 6$ $x = 50$

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	25	20	10	?

Элементы с индексами от 1 до $k - 1$ не изменились.

$k = 6$ $x = 50$

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	25	20	10	?

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 .

$k = 6$ $x = 50$

i	1	2	3	4	5	6	7	8	9	10
a[i]	5	15	20	25	30	35	35	25	20	10

Фрагмент программы:

```

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

```

Программа:

```

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 $A [1..n]$.

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

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 $A[n - i + 1]$.

Вспомним, как можно произвести обмен значений между двумя переменными. Самый простой вариант – использование вспомогательной переменной:



```
R := A[i];
A[i] := A[n-i+1];
```

```
A[n-i+1] := R;
```

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:

```
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.

Существует 2 вида сортировки:

- по возрастанию;
- по убыванию.

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 неупорядоченного 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 своими соседями справа).

Фрагмент программы:

```

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;

```

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, в пункте 3, повторяются с неотсортированными элементами массива до тех пор, пока не останется один неотсортированный элемент (его значение будет максимальным).

5. Программа:

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.

```