2017

Глава 3 Структурные типы данных

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

3.1 Массивы

Массив – это упорядоченная совокупность *однотипных данных*. Каждому элементу массива соответствует один или несколько *индексов порядкового типа*, определяющих положение элемента в массиве.

a

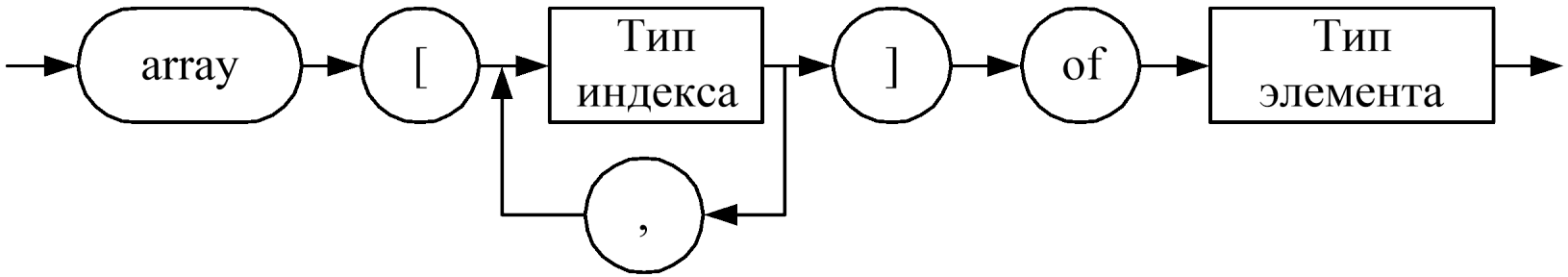
-5	0	12	54	-8
1	2	3	4	5

c

	-5	-4	-3
A	-5	0	13
B	46	83	-8
C	54	0	93

b

A	N	D		O	R	...	T
0	1	2	3	4	5	...	255



Количество *типов* индексов задает ***размерность*** массива.

Тип индекса – порядковый – определяет доступ к элементу.

Тип элемента – любой кроме файла, в том числе массивы, строки и т.п.

Массив в памяти не может занимать более 2 Гб.

Примеры объявления массивов

```
Var a:array[1..5] of integer;
    c:array['A'..'C',-5..-3] of byte;
    b:array[byte] of char;

Type mas=array[1..10] of integer;
Var a:mas;
```

Инициализация массива при объявлении

```
Var a:array[1..5]of real=(0,-3.6,7.8,3.789,5.0);
    b: array[boolean, 1..5] of real=
                          ((0,-3.6,7.8,3.789,5.0),
                           (6.1,0,-4.56,8.9,3.0));
```

Операции над массивами

1. Операция присваивания (только для массивов одного типа):

Пример:

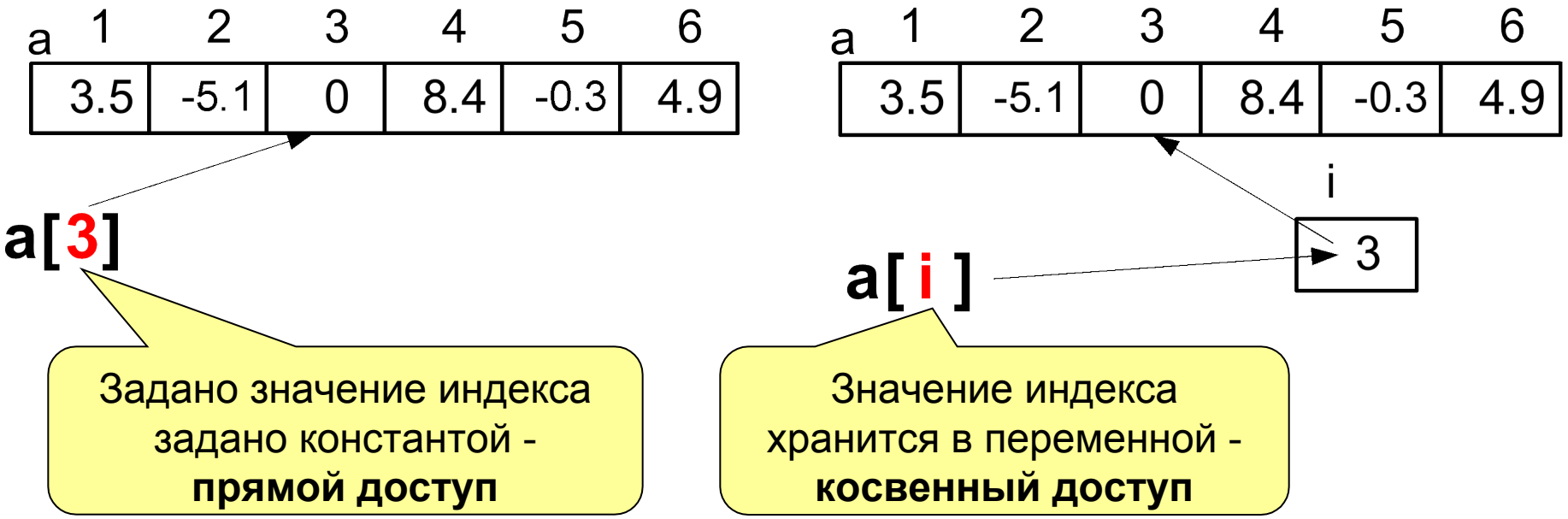
```
Var a, b:array[boolean] of real;
   ...
   a:=b;
```

2. Доступ к элементу массива:

Пример:

```
Var a:array[char,boolean] of real;
   ...
   a['A',true]:=5.1; {прямой доступ}
   ...
   Ch:='B'; b:=false;
   a[Ch,b]:=3; {косвенный доступ: значения индексов
                находятся в переменных}
```

Косвенный доступ к элементам массива



Косвенный доступ позволяет реализовать **последовательную обработку элементов массивов**:

```
for i:=1 to 6 do a[i]:=i*i;
```

или

```
for i:=6 downto 1 do a[i]:=i*i;
```

Операции над массивами (2)

3. Ввод/вывод массивов осуществляется поэлементно:

Пример 1. Ввод элементов одномерного массива

```
Var a:array[1..5] of real;
   ...
    for i:=1 to 5 do Read(a[i]);
    ReadLn; {обрабатывает последнее Enter}
```

Значения вводятся через пробел, Tab(→) или Enter(↵), например:

а) 2 -6 8 56 34 ↵

б) 2 ↵

-6 → 8 ↵

56 ↵

34 ↵

Операции над массивами (3)

Пример 2. Вывод матрицы.

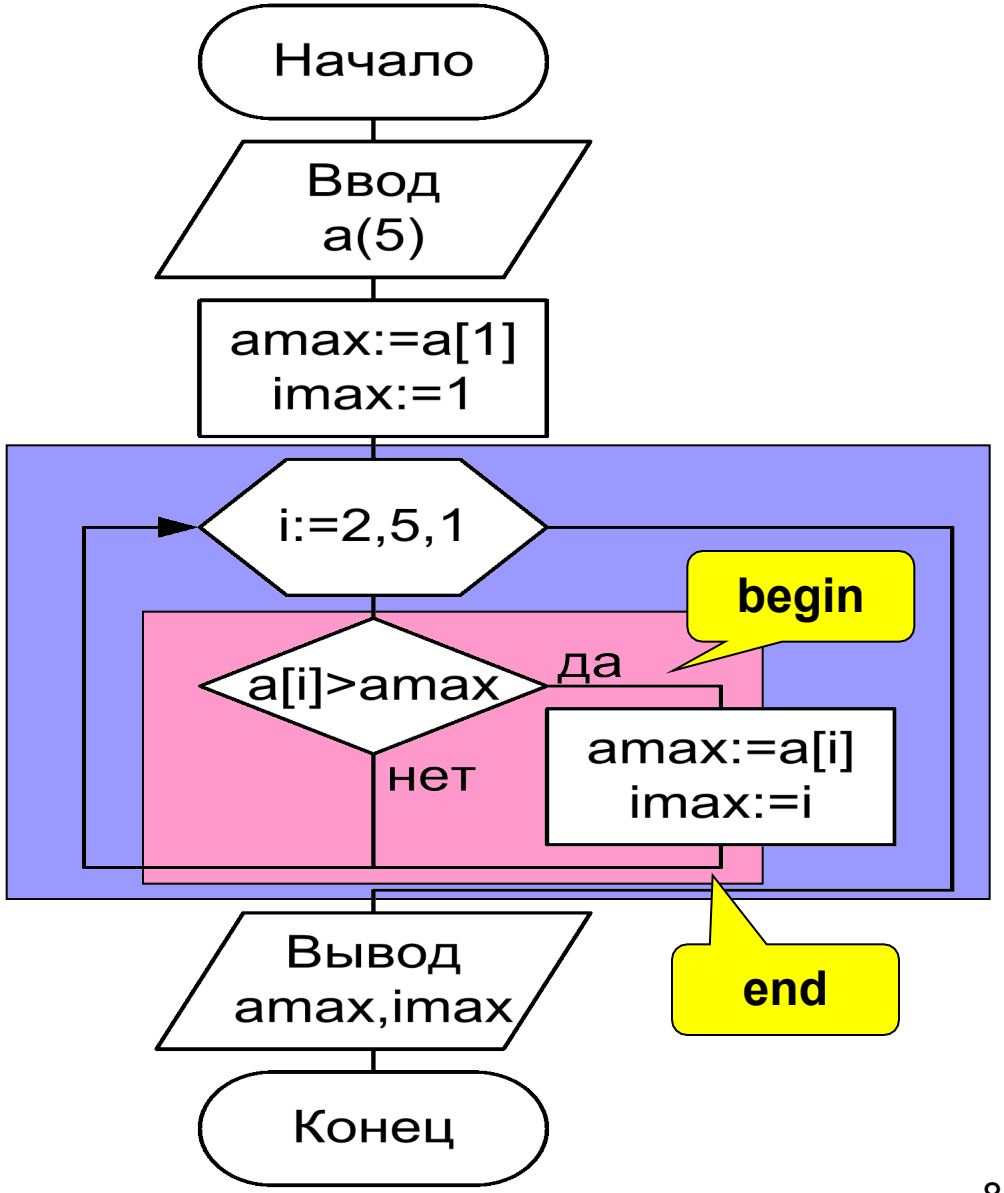
```
Var a:array[1..5, 1..7] of real;
Begin ...
    for i:=1 to 5 do
      begin
         for j:=1 to 7 do Write(a[i, j]);
                { a[i,1]  a[i,2]  a[i,3]  a[i,4]  a[i,5]  a[i,6]  a[i,7]}
         WriteLn; {переходим на следующую строку}
      end; ...
```

Максимальный элемент массива и его номер

A

45	34	56	2	-3

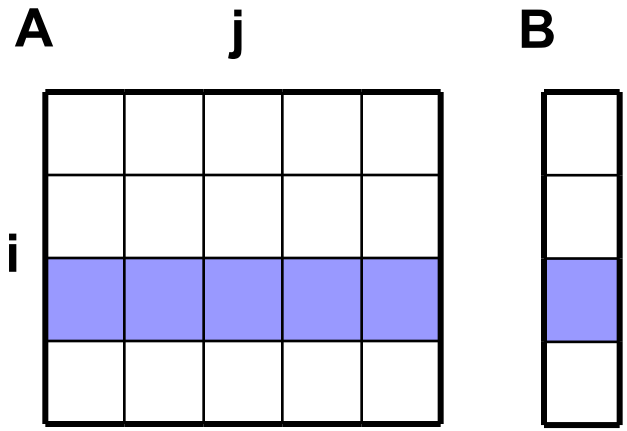
AMAX	IMAX	i
56	3	5

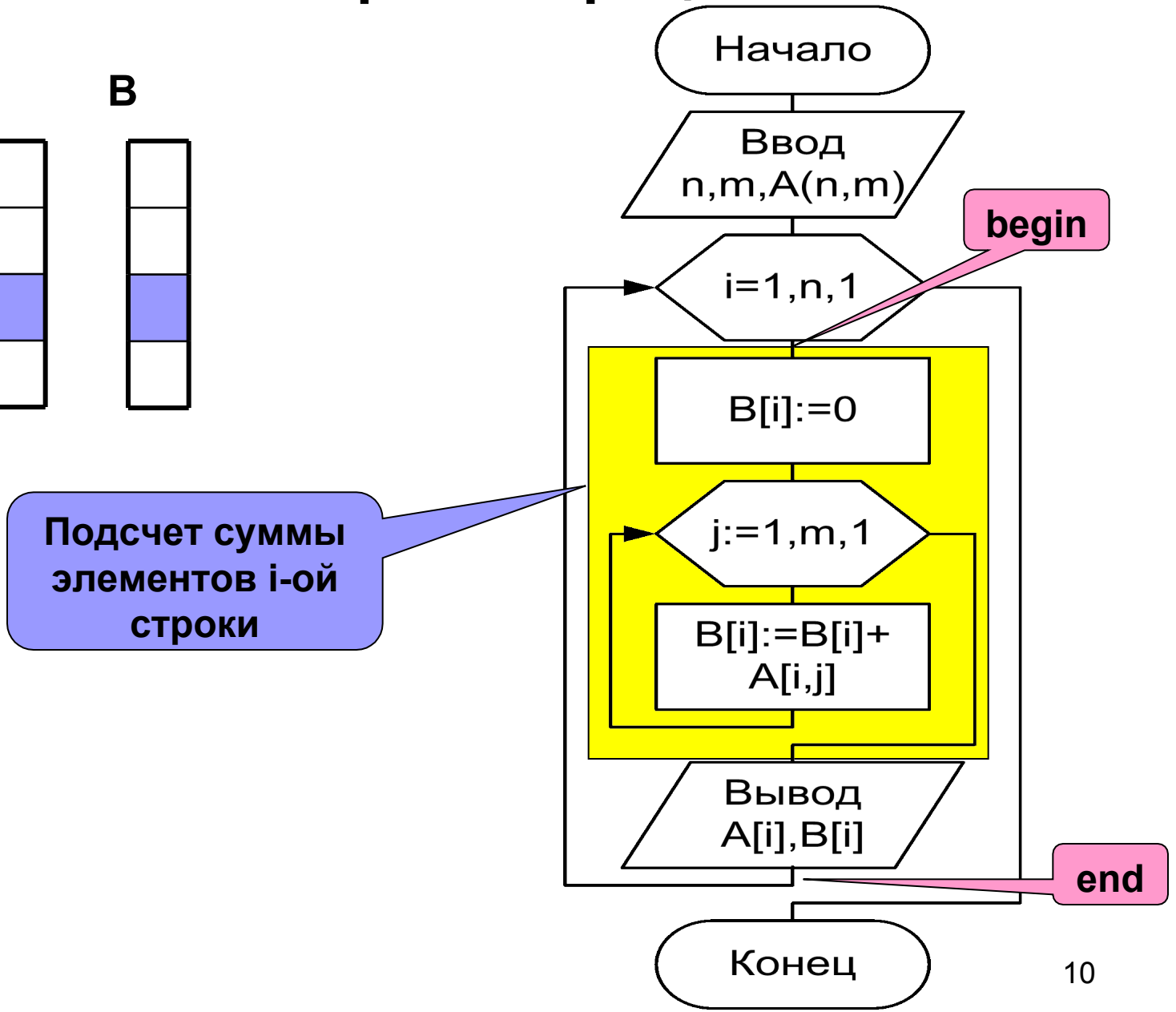


Программа

```
Program Ex3_1;
{$APPTYPE CONSOLE}
Uses SysUtils;
Var a:array[1..5] of single; amax:single;i, imax:byte;
Begin   WriteLn('Input 5 values:');
     for i:=1 to 5 do Read(a[i]);  ReadLn;
     amax:=a[1];
     imax:=1;
     for i:=2 to 5 do
          if a[i]>amax then
               begin  amax:=a[i];  imax:=i;   end;
     WriteLn('Values:');
     for i:=1 to 5 do Write(a[i]:7:2); WriteLn;
     WriteLn('Max =', amax:5:2, ', number=', imax);
     ReadLn;
 End.
```

Сумма элементов строк матрицы



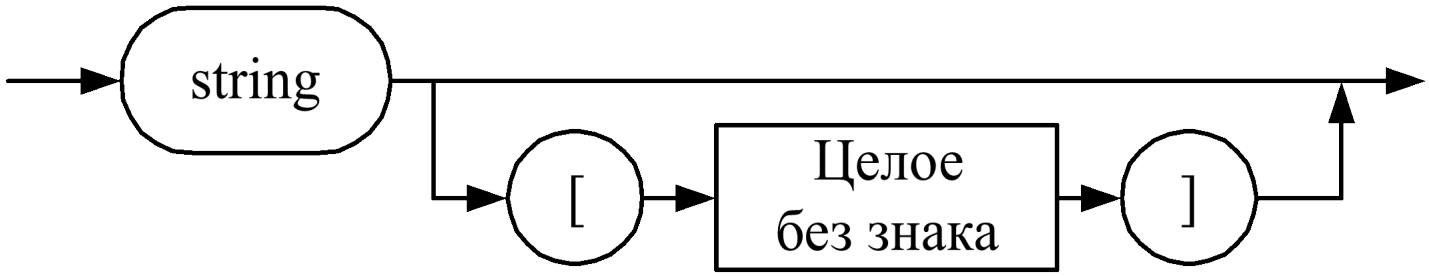


Программа суммирования элементов строк

```
Program Ex3_2;
{$APPTYPE CONSOLE}
uses SysUtils;
Var A:array[1..10,1..10] of real;
    B:array[1..10] of real;     n,m,i,j:byte;
Begin WriteLn('Input n,m'); ReadLn(n,m);
     WriteLn('Input matrics n*m values:');
     for i:=1 to n do
         begin  for j:=1 to m do Read(A[i,j]);  ReadLn;  end;
     WriteLn('Results:');
     for i:=1 to n do
       begin   B[i]:=0;
           for j:=1 to m do B[i]:=B[i]+A[i,j];
           for j:=1 to m do Write(A[i,j]:7:2);
           WriteLn(' Sum= ',B[i]:7:2);
       end;
      ReadLn;
End.
```

3.2 Строки

Строка – последовательность символов.



Целое – максимальная длина строки.

Внутренний формат:

Примеры описания строк

1) `Var S1,S2:string[40]; S3:string;`

2) С предварительным объявлением типов:

```
Type S40 = string[40];
     ST = string;
Var  S1,S2: S40;
     S3:ST;
```

3) С инициализацией

```
Var S:string[40]='Строковая константа';
    S1:string = '';
```

Операции над строками

1. Присваивание строк:

```
S1:='ABCD';
S1:=S2;
S1:='A';
S1:='';{пустая строка}
```

2. Обращение к элементу:

```
S1[5] - прямое
S1[i] - косвенное
```

3. Конкатенация (сцепление) строк:

```
St:=St + 'A';
St:='A' + 'B';
```

4. Операции отношения – выполняется попарным сравнением кодов символов, результат определяется по отношению кодов первых различных символов:

```
b:= S1 > S2;
'T' < 'Ta'
```

5. Ввод-вывод строк:

```
ReadLn(S1);
{Строка вводится до Enter
 или указанной длины}

WriteLn(S1);
```

Стандартные процедуры и функции

1. Функция **Length(st):word** – возвращает длину строки st:

```
n:=Length(st1);
```

2. Процедура **Delete(st, index, count)** – удаляет count символов строки st, начиная с символа с номером index:

```
S1: = 'dddddsssssfffff';
Delete(S1,6,5);
```

‘dddddfffff’

3. Процедура **Insert(St2,St1,index)** – вставляет подстроку символов St2 в строку St1, начиная с символа с номером index:

```
S1 = 'ddddddddddd';
S2 = 'aaaaaa' ;
Insert(S2,S1,6);
```

‘dddddaaaaaaddddd’

```
Insert('Pas',S1,6);
```

‘dddddPasaaaaaaddddd’

Стандартные процедуры и функции (2)

4. Процедура **Str(x[:w[:d]],St)** – преобразует результат выражения x, в строку st, содержащую запись этого числа в виде последовательности символов (как при выводе).

```
x:=-5.67;
Str(x:7:3,s1);
```

' -5.670'

5. Процедура **Val(St,x,Code)** – преобразует строку St с записью числа в виде последовательности символов во внутреннее представление целого или вещественного числа и помещает его в переменную x. В целочисленной переменной Code процедура возвращает код ошибки:

```
Var S:string; Code:integer; a:real;   ...
...repeat
      Write('Input a:');
      ReadLn(S);
      Val(S,a,Code);
      if Code<>0 then WriteLn('Input error');
   until Code=0; ...
```

Стандартные процедуры и функции (3)

6. Функция **`Copy(St,index,count):string`** – возвращает фрагмент строки St, длиной count символов, начиная с символа с номером index:

```
S1 = 'qqqEEEEEEuuuuu';
S:= Copy(S1,4,6);
```

'EEEEEE'

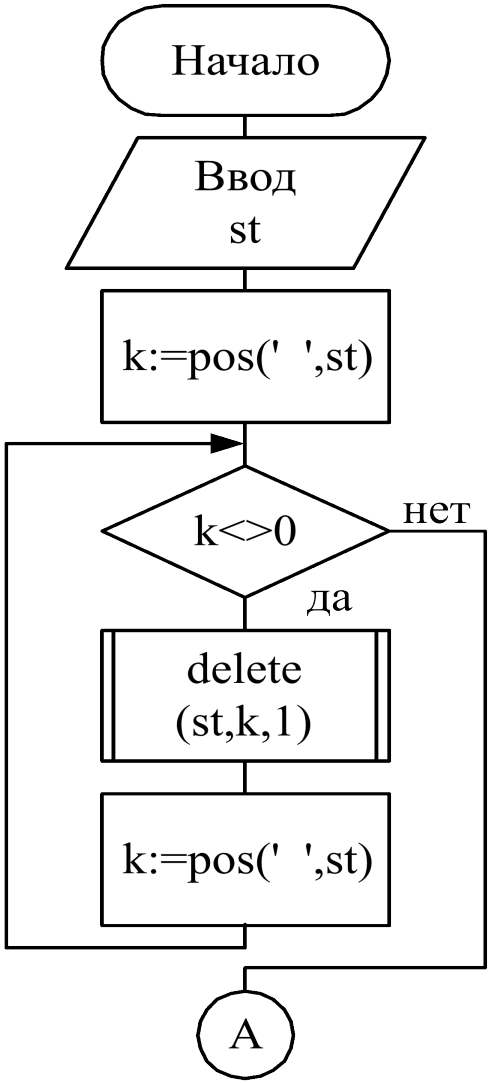
7. Функция **`Pos(St2,St1):integer`** – возвращает номер позиции первого вхождения подстроки St2 в строку St1. Если вхождение не найдено, то функция возвращает 0:

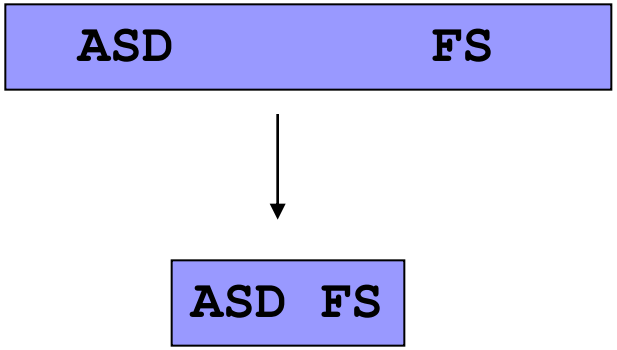
```
S1 = 'qqqEEppEEuuuuu';
i:= Pos('EE',S1);
```

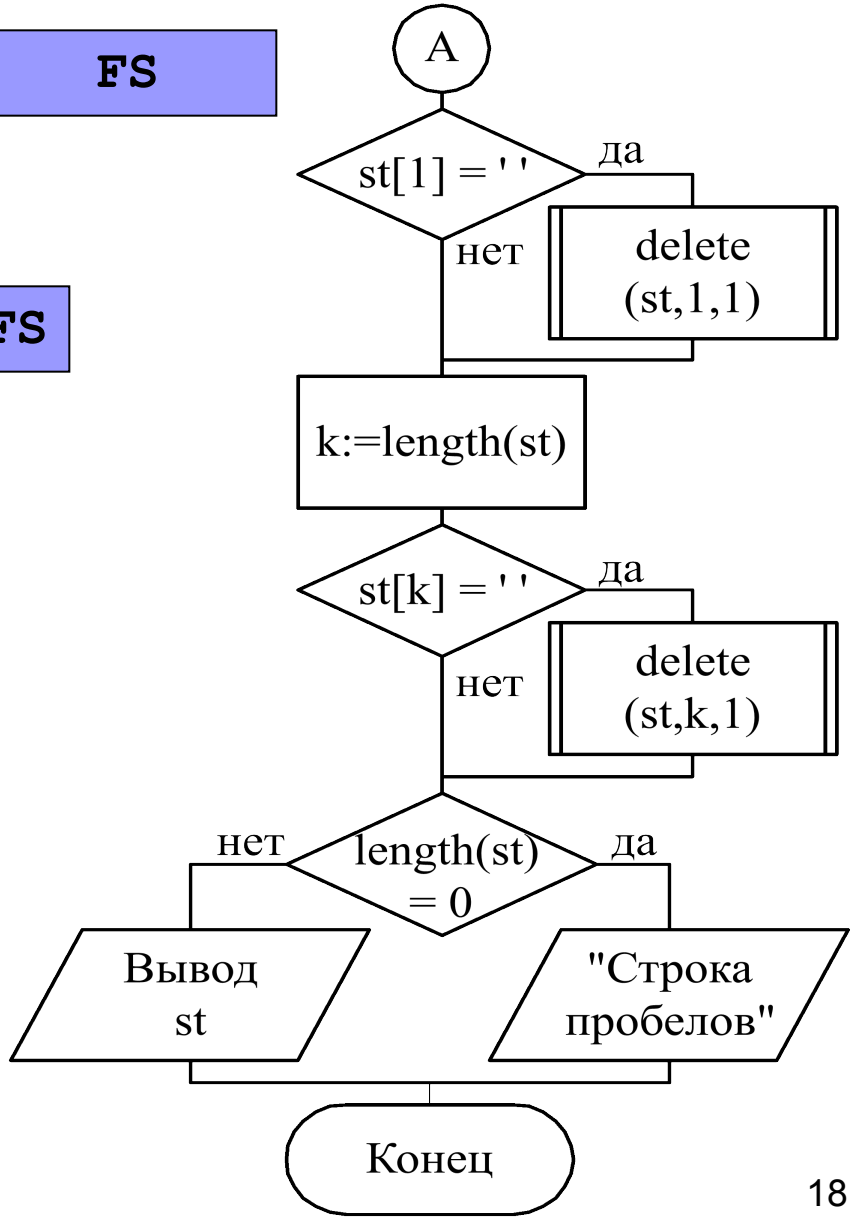
i=4

8. Функция **`UpCase(ch):char`** – возвращает символ, соответствующий символу верхнего регистра для ch, если таковой имеется, либо сам символ ch, если для него не определен символ верхнего регистра.

Удаление «лишних» пробелов из строки







Программа

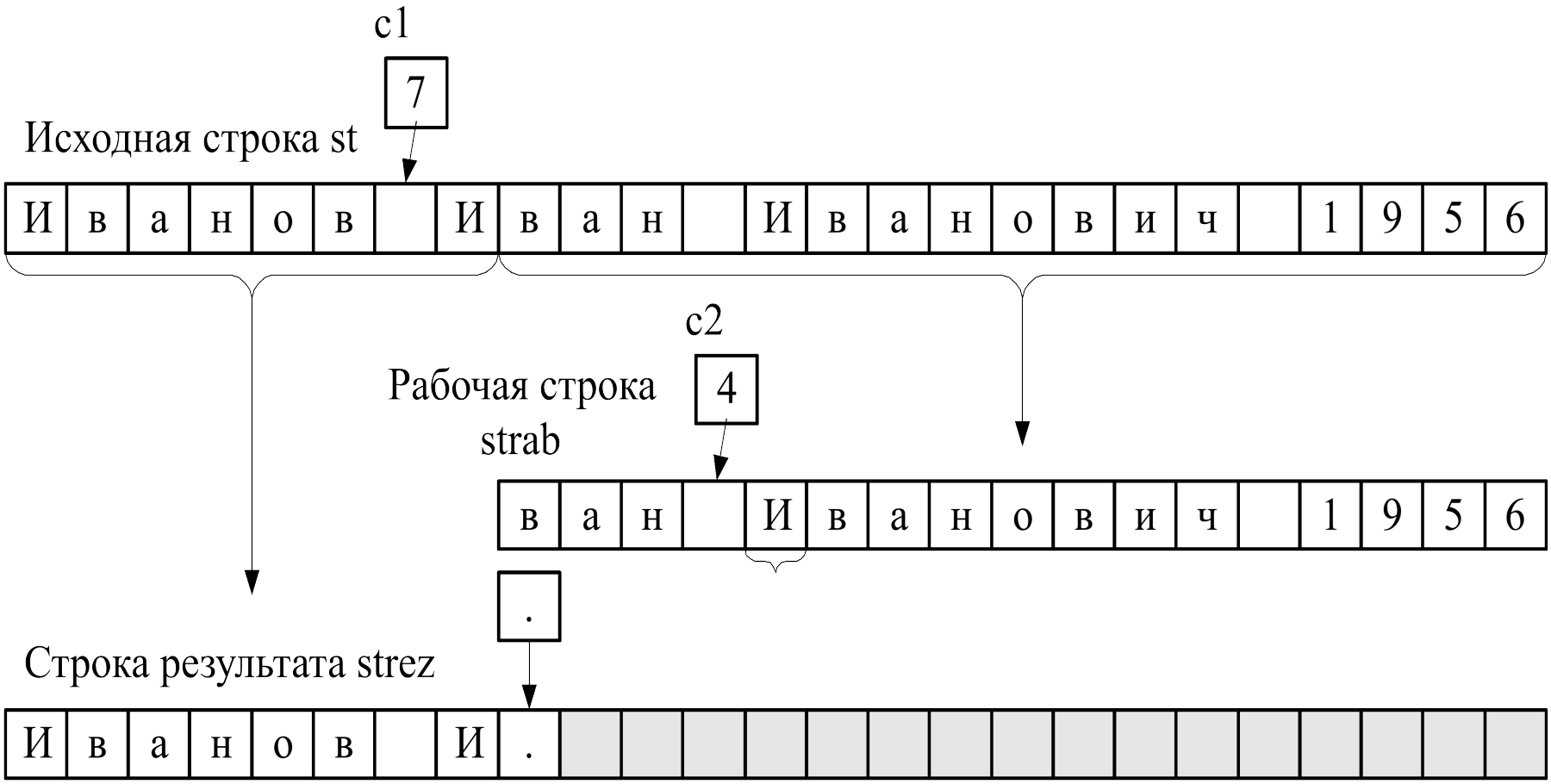
```
Program Ex3_3;
{$APPTYPE CONSOLE}
Uses SysUtils;
Var  st:string[40];   k:byte;
   Begin
      WriteLn('Input string <= 40 symbols');
      ReadLn(st);
      WriteLn('String:', st);
      k:=pos('  ',st);
      while k<>0 do
          begin delete(st,k,1); k:=pos('  ',st);  end;
      if st[1]= ' '  then delete(st,1,1);
      k:= length(st);
      if st[k]= ' '  then delete(st,k,1);
      if length(st)<>0 then WriteLn('Result:',st)
                       else WriteLn('Only spaces.');
      ReadLn;
End.
```

Преобразование последовательности строк

Вводится последовательность строк вида

`Иванов Иван Иванович 1956 ⇒ Иванов И.И. 45`

Завершение ввода – при чтении пустой строки.



Программа

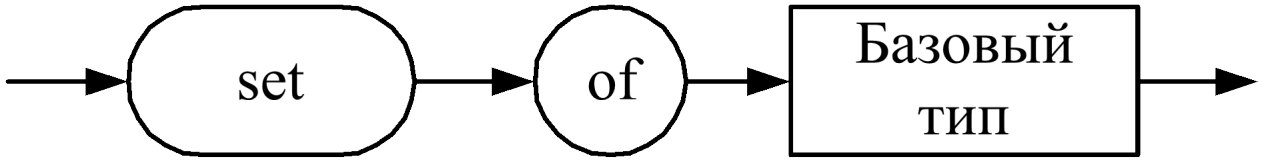
```
Program Ex3_4;
{$APPTYPE CONSOLE}
Uses SysUtils;
Var st,strez,strab:string[40];
    c1,c2,c3,n,old:word;code:integer;
 Begin
    WriteLn('Input string. End - empty string.');
    ReadLn(st);
    while st<>'' do
       begin
         c1:=Pos(' ',st);
         strez:=Copy(st,1,c1+1)+'.';
         strab:=Copy(st,c1+2,Length(st)-c1-1);
```

Программа (2)

```
        c2:=Pos(' ',strab);
        strez:=strez+strab[c2+1]+'.';
        Delete(strab,1,c2+1);
        c3:=Pos(' ',strab);
        Delete(strab,1,c3);
        Val(strab,n,code);
        old:=2001-n;
        Str(old,strab);
        strez:=strez+' '+strab;
        WriteLn(strez);
        WriteLn('Input string. End - empty string.');
        ReadLn(st);
    end;
End.
```

3.3 Множества

Множество – неупорядоченная совокупность неповторяющихся элементов.



Тип элементов – порядковый, кроме Word, Integer, SmallInt, LongInt. Количество элементов не должно превышать 256.

```
Type
   Digits = set of 1..100;
   Setchar = set of char;
   letter = set of 'a'..'z';
Var mychar: setchar;
    mydig: Digits;
    simst: letter;
или
Var number: set of 1..31;
    cif: set of  0..9;
    kods: set of #0..#255;
```

Конструкторы и инициализация множеств

Конструкторы множеств – константы множественного типа:

```
[] – пустое множество;
[2,3,5,7,11] – множество чисел;
['a','d','f','h'] – множество символов;
[1,k] – множество чисел, переменная k должна содержать число;
[2..100] – множество содержит целые числа из указанного интервала;
[k..2*k] – интервал можно задать выражениями;
[red,yellow,green]- множество перечисляемого типа
```

Инициализация множеств при объявлении:

```
Type setnum = set of byte;
Var S:setnum = [1..10];
```

Операции над множествами

1. Присваивание:

```
A:=B;
A:=[];
```

2. Объединение, пересечение и дополнение:

- ***A+B*** (A∪B) – объединение множеств A и B – множество, состоящее из элементов, принадлежащих множествам A и B
- ***A*B*** (A∩B) – пересечение множеств A и B – множество, состоящее из элементов, принадлежащих одновременно и множеству A и множеству B.
- ***A-B*** (A \ B) – дополнение множества A до B – множество, состоящее из тех элементов множества A, которые не принадлежат множеству B.

Примеры:

```
[1,2]+[3,4] = [1,2,3,4];
[1..10]*[3,8,9,15,23,45] = [3,8,9];
[1..15]-[3,8,9,15,23,45] = [1,2,4..7,10..14];
[red,blue,green,black]*[blue,magenta,yellow] = [blue];
```

Операции над множествами (2)

3. Операции отношения:

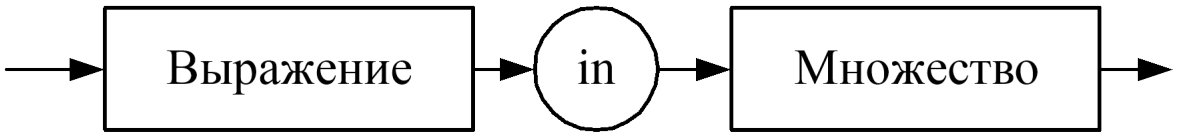
A = B – проверка совпадения множеств A и B (если совпадают – true)

A <> B – проверка не совпадения множеств A и B (не совпадают – true).

A <= B – проверка нестрогого вхождения A в B (если входит – true).

A > B – проверка строгого вхождения B в A (если входит – true).

4. Проверка вхождения элемента во множество:



Пример:

```
if a in [2..6] then …
```

Ввод-вывод элементов множеств

Значения множественного типа нельзя вводить и выводить !

Ввод элементов множества:

```
Var S:set of 1..100; n:word; ...
   S:=[];
   Read(n);
   while n<>0 do
     begin
          S:=S+[n];
          Read(n);
     end;
  ReadLn;...
```

Вывод элементов множества:

```
Var S:set of 'a'..'z'; ...
  for i:='a' to 'z' do
     if  i in S then Write(i:3);
  WriteLn;
```

Определение множества цифр числа

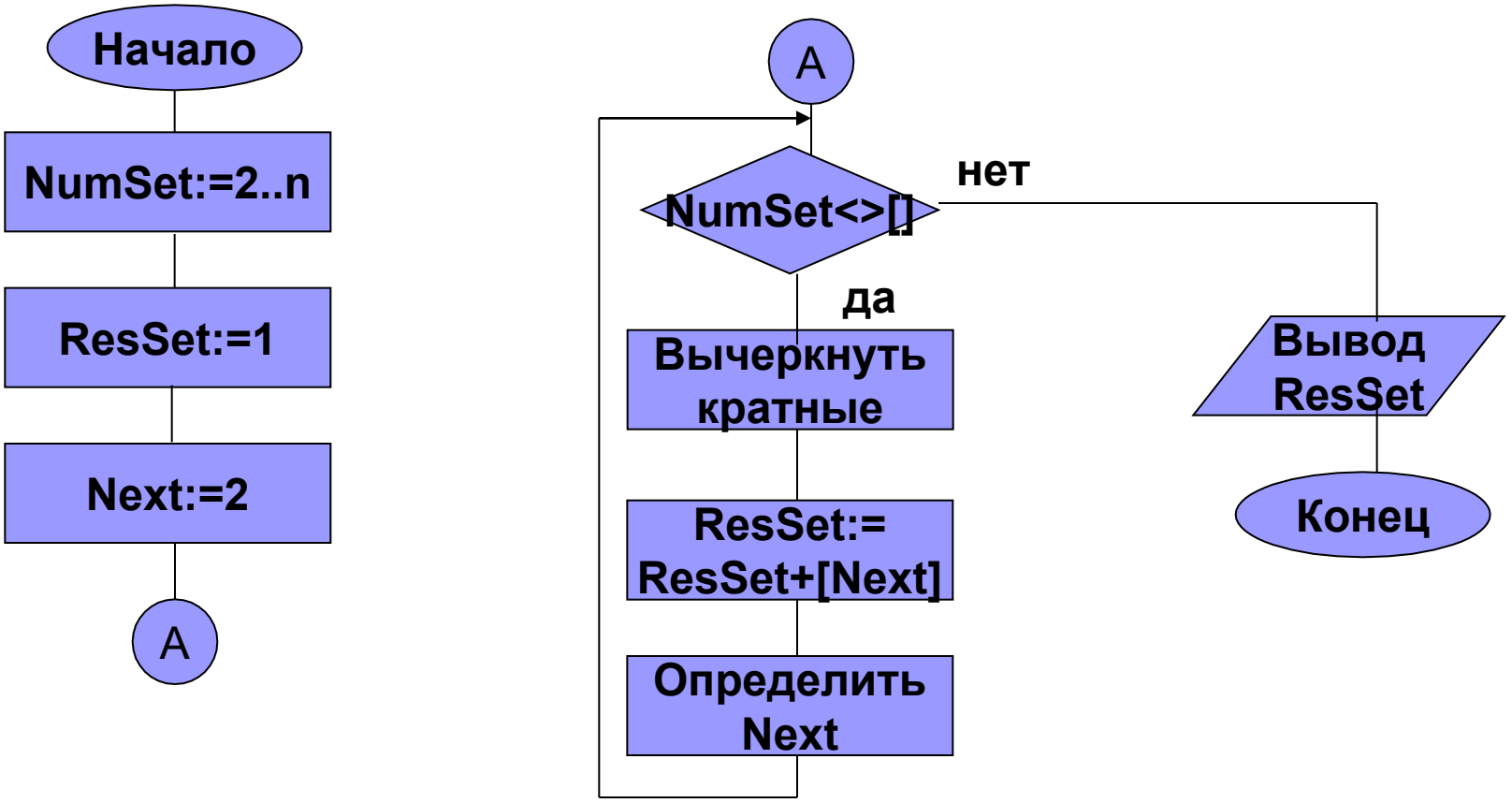
```
Program Ex3_5;
{$APPTYPE CONSOLE}
Uses SysUtils;
Var n:longint;
     st:string;
     mnoj:set of '0'..'9';
     i:integer;  j:char;
   Begin
      WriteLn('Input value:');
      ReadLn(n);
      Str(abs(n),st); // преобразуем число в строку
      mnoj:=[];
      for i:=1 to length(st) do
                mnoj:=mnoj+[st[i]]; //добавляем цифры
      WriteLn('String ',n,' includes ');
      for j:='0' to '9' do
           if j in mnoj then Write(j+' ');
      ReadLn;
   End.
```

«Решето Эратосфена» (простые числа)

Исходное состояние (NumSet):

`2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`

Результат (ResSet): `1`



«Решето Эратосфена» (2)

```
Program Ex3_6;
{$APPTYPE CONSOLE}
Uses SysUtils;
Const N = 100;
Type Numbers = set of 1..N;
Var  NumSet,ResSet:Numbers;
     Next,Nn,i:word;
Begin
     NumSet := [2..N];
     ResSet := [1];
     Next := 2;
```

«Решето Эратосфена» (простые числа)(3)

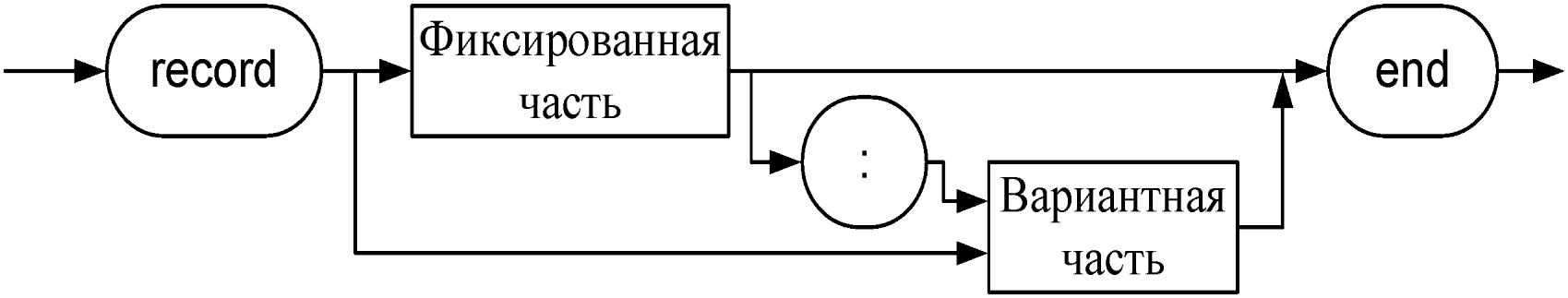
```
  while NumSet <> [] do
    begin
      Nn := Next;
      while Nn <= N  do
        begin
          NumSet := NumSet - [Nn];
          Nn := Nn + Next;
        end;
      ResSet := ResSet + [Next];
      repeat
         inc(Next);
      until (Next in NumSet) or (NumSet =[]);
    end;
  for i:=1 to N do
      if i in ResSet then write(i,' ');
  ReadLn;
End.
```

3.4 Записи

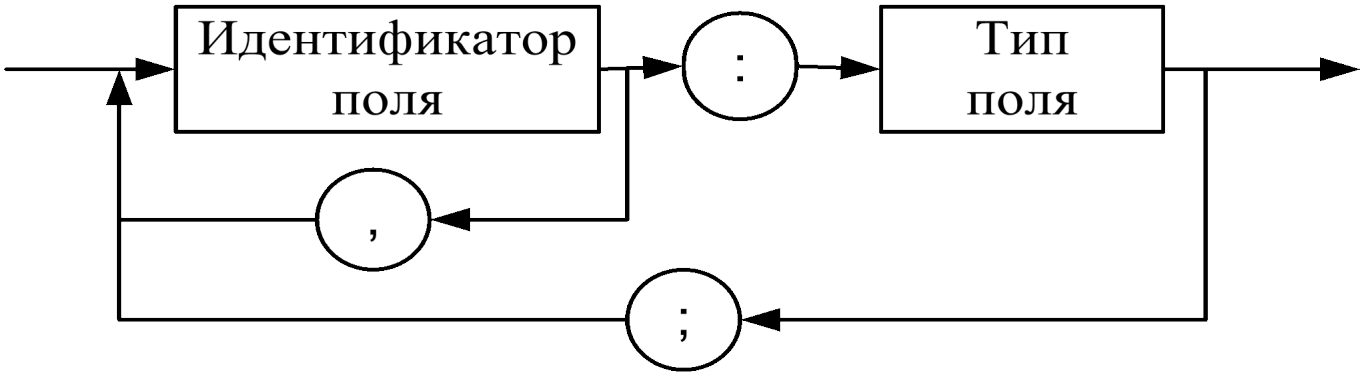
Запись – это структура данных, образованная фиксированным числом разнотипных компонентов, называемых **полями** записи.

Пример записи: Иванов Иван 20 лет студент 1 курса ⇒

Иванов | Иван | 20 | студент | 1



Фиксированная часть записи:



Объявление и инициализация записей

Примеры:

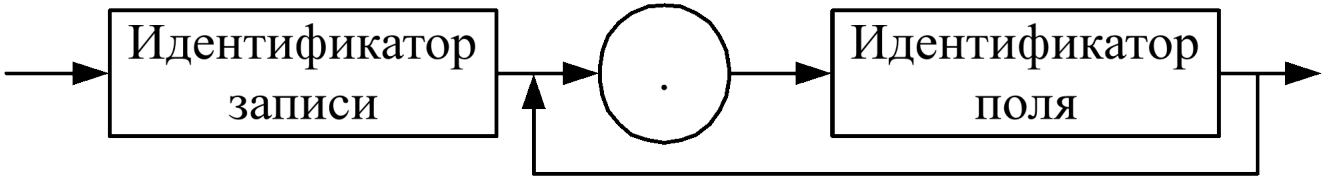
```
а) Var  Zap1: record
                  Day:1..31;
                  Month: 1..12;
                  Year: word;
               end;

б) Type Data = record
                  Day:1..31;
                  Month: 1..12;
                  Year: word;
               end;
   Var Zap1:Data;

в) Var BirthDay: Data = (Day:30; Month:6; Year:1973);
```

Операции над записями

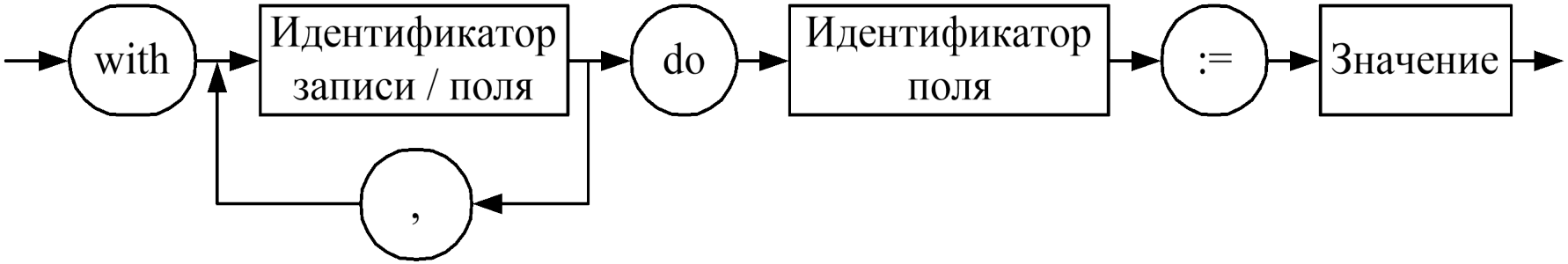
1. Присваивание записей одного типа:

```
Var A,B: record Day:1..31; Month: 1..12; Year: word; end;
   …
A:=B;
```

2. Доступ к полям записи:




```
A.Day:=21; {точечная нотация}
```




```
with A do Day := 21; {оператор доступа}
```

3. Ввод и вывод записей осуществляется по полям.

Массив записей

Задача.

Вводится список:

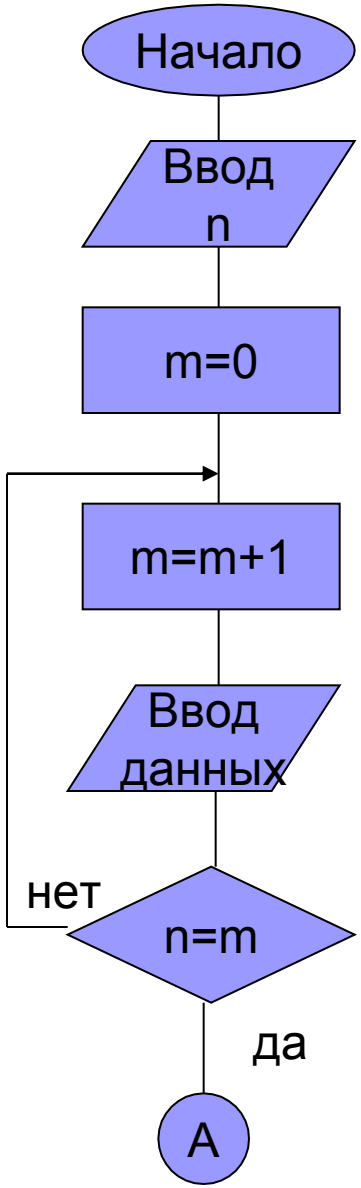
Ф.И.О.	Год р.	Месяц р.	Дата р.
Иванов Б.А.	**1986**	**11**	**26**
Петров М.А.	**1985**	**5**	**12**
Сидоров А.В.	**1986**	**4**	**8**

Определить дату рождения по фамилии и инициалам.

Программа

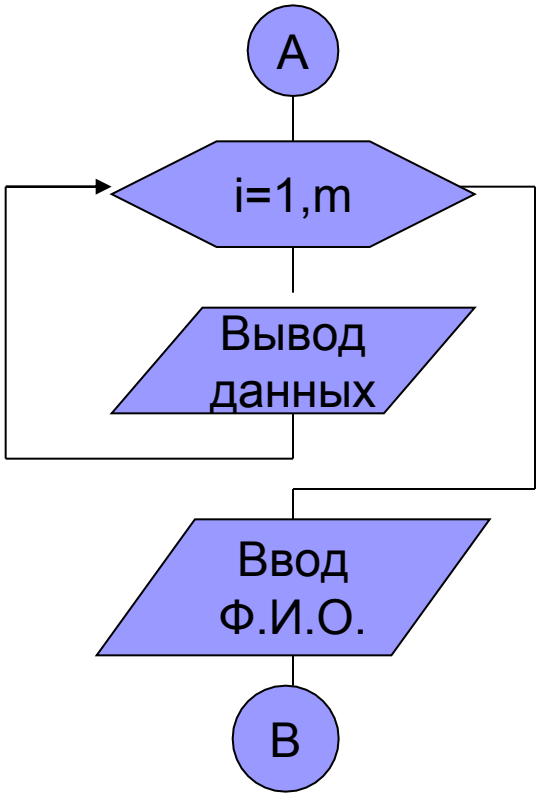
```
Program Ex3_7;
{$APPTYPE CONSOLE}
Uses SysUtils;
Type
   data=record
           year:word;
           month:1..12;
           day:1..31;
        end;
    zap=record
           fam:string[16];
           birthday:data;
        end;
Var  fb:array[1..25] of zap;
     fff:string[16];   i,j,m,n:byte;   key:boolean;
```

Ввод записей



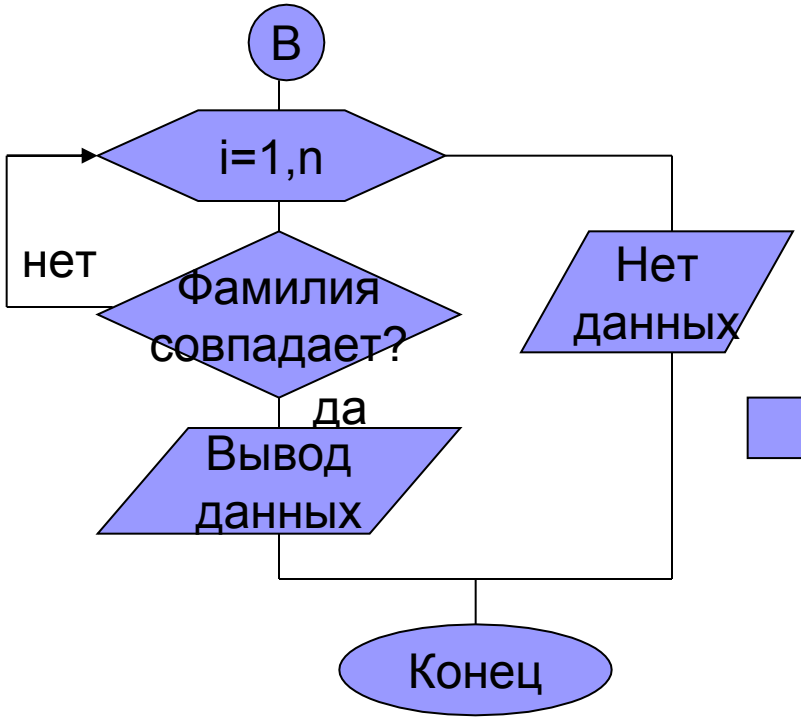

```
Begin
  WriteLn('Input n<=25');
  ReadLn(n);
  m:=0; {счетчик записей}
  repeat
     m:=m+1;
     Write('Input family  :');
     Readln(fb[m].fam);
     Write('Input year :  ' );
     Readln(fb[m].birthday.year);
     Write('Input month  :  ');
     ReadLn(fb[m].birthday.month);
     Write('Input date  :  ');
     ReadLn(fb[m].birthday.day);
   until n=m;
```

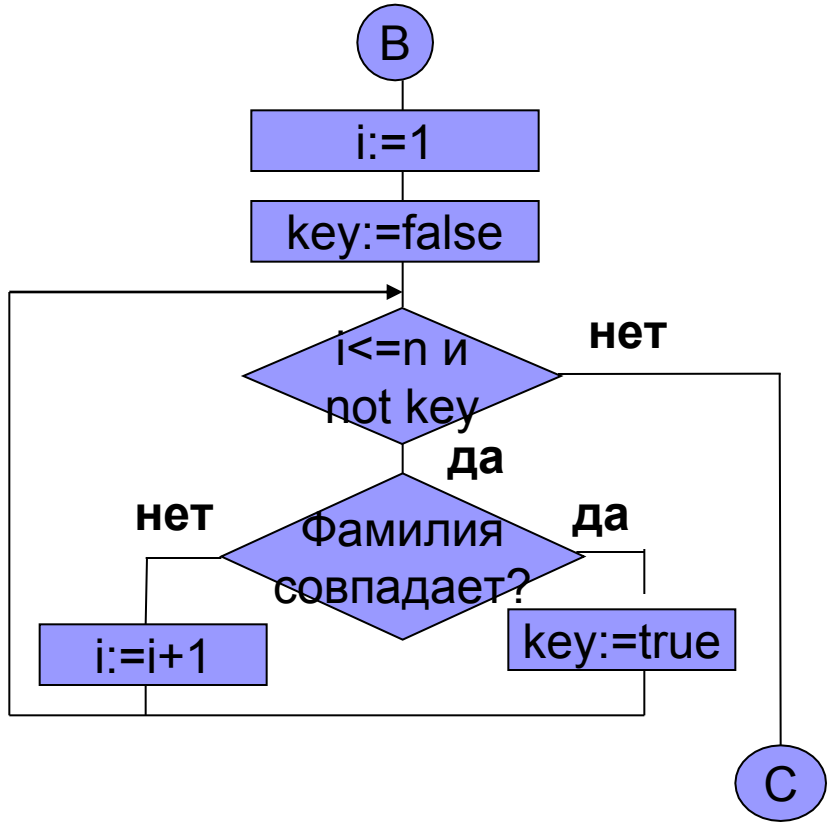
Вывод списка и ввод фамилии




```
WriteLn;
WriteLn('List of group ');
WriteLn;
 for i:=1 to m do
    with fb[i] do
      begin
        Write(i:2,fam:17);
        with birthday do
           WriteLn(year:6,
                   month:4,
                   day:4);
      end;
 WriteLn;
 Write('Input family:');
 ReadLn(fff);
```

Поиск. Программирование поискового цикла






```
i:=1;
key:=false;
while (i <= n) and (not key) do
    if fb[i].fam = fff then
        key:=true
    else i:=i+1;
```

Вывод результата

```
{вывод результата}
 if key then
   with fb[i] do
     begin
       WriteLn('Student :');
       Write(fam:18,'   ');
       with birthday do
          WriteLn(day:2,':',
                  month:2,':',
                  year:5);
     end
 else  WriteLn('No data about:',fff:18);
 ReadLn;
End.
```