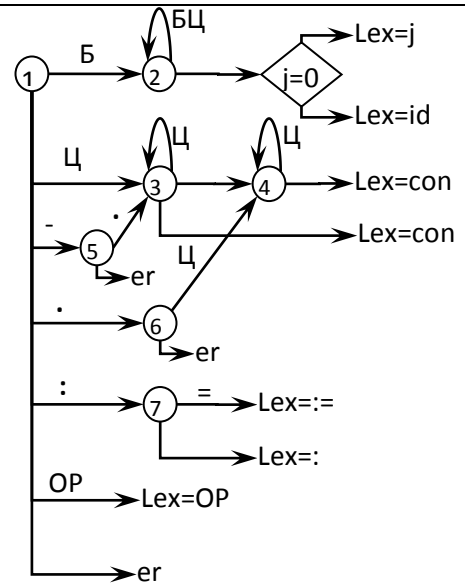
	Лексема
1	Program
2	Var
3	Begin
4	end.
5	real
6	for
7	read
8	write
9	to
10	do
11	,
12	:
13	;
14	+
15	*
16	(
17	)
18	=
19	:=
20	id
21	const

<програма>::=program <id> var <сп.ід.>: real; begin <сп.оп.> end.
< сп.ід.>::=<id>|< сп.ід.>, <id>
<сп.оп.>::=<опер.>| <сп.оп.>; <опер.>
<опер.>::=read(<сп.ід.>)|write(<сп.ід.>)|<id>:=<вираз>|
for <id>=<вираз> to <вираз> do < опер.>
<вираз>::=<дод>|<вираз>+<дод >
<дод>::=<мн>|<дод>+<мн>
<мн>::=<id>|(<вираз>)|<const>
<id>::=<буква>|<id>< буква >|<id><цифра>
<const>::=<ціле>|<ціле>.|.<цбз>|<ціле>.<цбз>
<ціле>::=<цбз>|-<цбз>
<цбз>::=<цифра>|<цбз><цифра>
<буква>::=a|b|c|…|z
<цифра>::=0|1|2|…|9

Таблиця класів

.	.
a..z	Б
0..1	Ц
;,+*()	ОР
:	:
=	=
-	-




```
figures: set of char =
['0','1','2'…];
letters: set of char =
['a','b','c'…];
OP: set of char =
[';',',','+','*',…];
```

Спосіб 1 (з case)

```
State : Integer;
…
Result := NONECODE;
S := ''; <пропуск білих роздільників>
Case State of
1:begin
    c := Nextchar(); s:=s+c;
    if c in figures then
      state := 2
    else if c in figures then
      state := 3
    else if c = '-' then
      state := 5
    else if c = '.' then
      state := 6
    else if c = ':' then
      state := 7
    else if c in OP then
      begin
        addlex(c);
        result := ACCEPTCODE;
      end
    else result := ERRORCODE;
  end;//of 1
2:begin
    c := Nextchar(); s:=s+c;
    if (c in figures + letters) then
      state := 2
    else begin
      retpos(); // возврат символа
      addlex(s, -1);
      result := ACCEPTCODE;
    end;
  end; //of 2
3:begin
    c := Nextchar(); s:=s+c;
    if (c in figures) then
      state := 3
    else if c = '.' then
      state := 4
    else begin
      retpos(); // возврат символа
      addlex(s, CONCODE);
      result := ACCEPTCODE;
    end;
  end; //of 3
4:begin
    c := Nextchar(); s:=s+c;
    if (c in figures) then
      state := 4
    else begin
      retpos(); // возврат символа
      addlex(s, CONCODE);
      result := ACCEPTCODE;
    end;
  end; //of 4
5:begin
    c := Nextchar(); s:=s+c;
    if (c in figures) then
      state := 3
    else result := ERRORCODE;
  end; //of 5
6:begin
    c := Nextchar(); s:=s+c;
    if (c in figures) then
      state := 4
    else result := ERRORCODE;
  end; //of 6
7:begin
    c := Nextchar(); s:=s+c;
    result := ACCEPTCODE;
    if (c = '=') then
      addlex(':=', 19)
    else addlex(':', 12);
  end; //of 7
 end;//of case
until result <> NONECODE;
```

Спосіб 2 (з мітками)

```
label l1, l2, l3 l4, l5, l6, l7, l8;
IsRead : Boolean = false;
…
l1: if eof then goto l8;
    <пропуск білих роздільників>
    s := '';
    if not IsRead  then
     c := Nextchar();
    s := s + c;
    case c of
    'a'..'z': goto l2;
    '0'..'9': goto l3;
    '-': goto l5;
    '.': goto l6;
    ':': goto l7;
    ',', ';', '+', …: begin
        addlex(c); IsRead := false;
        goto l1;
      end;
    else <помилка>
    end;
        end;//of case
l2: c := Nextchar(); s := s + c;
    case c of
    'a'..'z', '0'..'9': goto l2;
    else begin
      IsRead := true;
      addlex(s, -1);
      goto l1;
    end;
l3: c := Nextchar(); s := s + c;
    case c of
    '0'..'9': goto l2;
     '.': goto l4;
    else begin
      IsRead := true;
      addlex(s, CONCODE);
      goto l1;
     end;
…
l8:
end;// of lexic analizer
```

Спосіб 3 (з підпрограмами)

```
procedure p1(IsRead: boolean);
begin
  <пропуск білих розд.>
  if not IsRead then
    c := NextChar();
  s := s + c;
  case c of
    'a'..'z': p2;
    '0'..'9': p3;
    '-': p5;
    '.': p6;
    ':': p7;
    ',', ';', '+', …: begin
        addlex(c);
        p1(false);
      end;
  else <помилка, вихід>
end;

procedure p2();
begin
  c := NextChar(); s := s + c;
  case c of
    'a'..'z', '0'..'9': p2;
  else
    begin
      addlex(s, -1);
      p1(true);
    end;
end;

procedure p3();
begin
  case c of
    '0'..'9': p3;
    '.': p4;
  else
    begin
      addlex(s, CONCODE);
      p1(true);
    end;
  end;
end;
```

α	Символ	β	Семантическая подпрограмма
1	Б	2	
	Ц	3	
	-	5	
	.	6	
	ОР		[=] lex=ОР
	:	7	[≠] error
2	Б	2	
	Ц	2	[≠] lex = lex_j/id
3	Ц	3	
	.	4	[≠] lex=con
4	Ц	4	[≠] lex=con
5	Ц	3	[≠] error
6	Ц	4	[≠] error
7	=		[=] lex=**:=** [≠] lex=:

Алгоритм 3.18. Моделирование ДКА

Вход: входная строка x, завершенная символом конца файла eof, и детерминированный конечный автомат D с начальным состоянием sq, принимающими состояниями F и функцией переходов move.

Выход: ответ "да", если D принимает x, и "нет" в противном случае.

Функция move (s, c) дает состояние, в которое из состояния s ведет дуга при входном символе c. Функция nextChar возвращает очередной символ из входной строки x.

```
s = s0;
c = nextChar();
while ( c != eof ) {
  s = move(s,c);
  c = nextChar();
}
if ( s є F ) return "да";
else return "нет";
```

Рис. 3.27. Моделирование ДКА