Лексема					
1		Лексема	<програма>::=program <id> var <c< td=""><td>:п.ід.&gt;: real; begin &lt;сп.оп.&gt; end.</td><td><math>r \cdot r \cdot</math></td></c<></id>	:п.ід.>: real; begin <сп.оп.> end.	$r \cdot r \cdot$
3   Begin	1	Program	< cп.iд.>::= <id> &lt; cп.iд.&gt;, <id></id></id>		[1] [5] (i-0)
3   Begin	2	Var	<сп.оп.>::=<опер.>  <сп.оп.>;	lov-id	
Tor < d>	3	Begin	<опер.>::=read(<сп.ід.>) writ	e( <cп.iд.>) <id>:=&lt;вираз&gt; </id></cп.iд.>	I I ./∖Ц ./∖Ц
Seupas   Capa   Capa	4	end.	for <id>=&lt;вираз&gt; to &lt;ви</id>	Lex=con	
Tor	5	real	<вираз>::=<дод> <вираз>+<,	.11	
8 write	6	for	<дод>::=<мн> <дод>+<мн>	5 Lex=con	
9 to	7	read	<мн>::= <id> (&lt;вираз&gt;) <const></const></id>		
10 do	8	write	<id>::=&lt;буква&gt; <id>&lt; буква &gt; <id>&lt; цифра&gt;</id></id></id>		6
11 , <uds>::=<uudpa> <uds><uudpa>         12 :       <dyква>::=a b c  z         13 ;       <udpa>::=0 1 2  9         14 +       Таблиця класів         15 *       .         16 (       a.z Б         17 )       0.1 Ц         18 =       ;,+*() OP         19 :=       :         20 id       =     (op Lex=OP  (io','1','2'];  (letters: set of char = ['o','1','2'];  (op: set of char = ['a','b', 'c'];  (op: set of char = ['i',',',','+','*',];  (io','',','','','','','','','','','','',''</udpa></dyква></uudpa></uds></uudpa></uds>	9	to	<const>::=&lt;ціле&gt; &lt;ціле&gt;. .&lt;цбз&gt; &lt;ціле&gt;.&lt;цбз&gt;</const>		→er
12 :	10	do	<ціле>::=<цбз> -<цбз>		: = Lex=:=
12   :	11	,	<цбз>::=<цифра> <цбз><цифра>		
13   ;	12	:	<буква>::=a b c  z		l OP
15 *	13	;	<цифра>::=0 1 2  9		Lex=OP
16 ( az 5 figures: set of char = ['0','1','2'];  18 = ;,+*() OP letters: set of char = ['a','b', 'c'];  20 id = = ['i',',',','+','*',];	14	+	Таблиця	класів	
17 )  18 = ;,+*() ОР  19 := ::  20 id = = :    10','1','2'	15	*	•	•	<b>└──→</b> er
18 = ;,+*() OP letters: set of char = ['a','b', 'c'];  19 := : : OP: set of char = ['i',',',','+','*',];	16	(	az	Б	_
18 = ;;+'() OP  19 := ; :  20 id = = ['a','b', 'c'];  OP: set of char = [';',',','+','*',];	17	)	01	Ц	
19 := OP: set of char = [';',',','+','*',];	18	=	;,+*()	OP	
20 ld = = [\';',',',\'+','\*',];	19	:=	:	:	
21 const	20	id	=	=	
	21	const	-	-	] [ , , , , , , , ,],

## Спосіб 1 (з case)

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

## Спосіб 2 (з мітками)

```
label 11, 12, 13 14, 15, 16, 17, 18;
                                                    end;//of case
                                                12: c := Nextchar(); s := s + c;
IsRead : Boolean = false;
                                                    case c of
                                                    'a'..'z', '0'..'9': goto 12;
11: if eof then goto 18;
    <пропуск білих роздільників>
                                                    else begin
    s := '';
                                                      IsRead := true;
    if not IsRead then
                                                      addlex(s, -1);
     c := Nextchar();
                                                      goto 11;
    s := s + c;
                                                    end;
    case c of
                                                13: c := Nextchar(); s := s + c;
    'a'..'z': goto 12;
                                                    case c of
    '0'...'9': goto 13;
                                                    '0'...'9': goto 12;
    '-': goto 15;
                                                    '.': goto 14;
    '.': goto 16;
                                                    else begin
    \:': goto 17;
                                                      IsRead := true;
    `,', `;', `+', ...: begin
                                                      addlex(s, CONCODE);
         addlex(c); IsRead := false;
                                                      goto 11;
         goto 11;
                                                    end;
      end;
                                                ...
    else <помилка>
                                                18:
                                                end; // of lexic analizer
    end:
                                  Спосіб 3 (з підпрограмами)
                                                    'a'...'z', '0'...'9': p2;
procedure p1(IsRead: boolean);
begin
                                                  else
  <пропуск білих розд.>
                                                    begin
  if not IsRead then
                                                      addlex(s, -1);
    c := NextChar();
                                                      p1(true);
  s := s + c;
                                                    end;
  case c of
                                                end;
    'a'..'z': p2;
    '0'...'9': p3;
                                                procedure p3();
    '-': p5;
                                                begin
    '.': p6;
                                                  case c of
    ':': p7;
                                                    '0'...'9': p3;
                                                    '.': p4;
    `,', `;', `+', ...: begin
         addlex(c);
                                                  else
         p1(false);
                                                    begin
      end;
                                                      addlex(s, CONCODE);
  else <помилка, вихід>
                                                      p1(true);
                                                    end;
end;
                                                  end;
procedure p2();
                                                end;
begin
  c := NextChar(); s := s + c;
  case c of
```

case c or					
α	Символ	β	Семантическая		
			подпрограмма		
	Б	2			
	Ц	3			
1	-	5			
_	•	6			
	OP		[=] lex=OP		
	:	7	[≠] error		
2	Б	2			
	Ц	2	$[\neq]$ lex = lex <sub>j</sub> /id		
3	Ц	3			
	•	4	[≠] lex=con		
4	Ц	4	[≠] lex=con		
5	Ц	3	[≠] error		
6	Ц	4	[≠] error		
7	=		[=] lex=: <b>=</b> [≠] lex=:		

## Алгоритм 3.18. Моделирование ДКА

**Вход:** входная строка x, завершенная символом конца файла eof, и детерминированный конечный автомат D c 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
```