

	Лексема	<програма>::=program <id> var <сп.ід.>: real; begin <сп.оп.> end.		
1	Program	< сп.ід.>::=<id> < сп.ід.>, <id>		
2	Var	<сп.оп.>::=<опер.> <сп.оп.>; <опер.>		
3	Begin	<опер.>::=read(<сп.ід.>) write(<сп.ід.>) <id>:=<вираз>		
4	end.	for <id>=<вираз> to <вираз> do < опер.>		
5	real	<вираз>::=<дод> <вираз>+<дод>		
6	for	<дод>::=<мн> <дод>+<мн>		
7	read	<мн>::=<id> (<вираз>) <const>		
8	write	<id>::=<буква> <id>< буква> <id><цифра>		
9	to	<const>::=<ціле> <ціле>. <цбз> <ціле>.<цбз>		
10	do	<ціле>::=<цбз> -<цбз>		
11	,	<цбз>::=<цифра> <цбз><цифра>		
12	:	<буква>::=a b c ... z		
13	;	<цифра>::=0 1 2 ... 9		
14	+	Таблиця класів		
15	*	.	.	
16	(a..z	Б	
17)	0..1	Ц	
18	=	;,+*()	OP	
19	:=	:	:	
20	id	=	=	
21	const	-	-	

figures: set of char =
['0','1','2'...];

letters: set of char =
['a','b','c'...];

OP: set of char =
[';',',','.',',','+', '*', ...];

Списіб 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;

```

Способ 2 (з мітками)

```
label 11, 12, 13 14, 15, 16, 17, 18;
IsRead : Boolean = false;
...
11: if eof then goto 18;
    <пропуск білих роздільників>
    s := '';
    if not IsRead then
        c := Nextchar();
        s := s + c;
        case c of
            'a'..'z': goto 12;
            '0'..'9': goto 13;
            '-': goto 15;
            '.': goto 16;
            ':': goto 17;
            ',', ';', '+', ...: begin
                addlex(c); IsRead := false;
                goto 11;
            end;
        else <помилка>
        end;
end;
```

```
end; // of case
12: c := Nextchar(); s := s + c;
    case c of
        'a'..'z', '0'..'9': goto 12;
    else begin
        IsRead := true;
        addlex(s, -1);
        goto 11;
    end;
13: c := Nextchar(); s := s + c;
    case c of
        '0'..'9': goto 12;
        '.': goto 14;
    else begin
        IsRead := true;
        addlex(s, CONCODE);
        goto 11;
    end;
...
18:
end; // of lexic analyzer
```

Способ 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;
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	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. <pre> s = s0; c = nextChar(); while (c != eof) { s = move(s,c); c = nextChar(); } if (s ∈ F) return "да"; else return "нет"; </pre>
1	Б	2		
	Ц	3		
	-	5		
	.	6		
	OP		[=] lex=OP	
	:	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:=	

Рис. 3.27. Моделирование ДКА