3.3. ЛЕКСИЧНИЙ АНАЛІЗ РЯДКІВ

Всі мови програмування містять у собі, як правило, підмови таких узагальнених понять, як «ідентификатор», «число» та ін. Якщо в основній граматиці виділити деякі підграматики, то опис основної граматики стає менш об'ємним і менш громіздким. Ці підграматики або конструкти називаються, за аналогією з природними мовами, *лексемами*. Виділення лексем має певний сенс для розроблювачів компіляторів (трансляторів). Один із модулів компілятора, називаний сканером, здійснює аналіз рядків на послівному (або лексичному) рівні. Це свідчить, що символьні рядки, що утворюють речення мови, аналізуються і групуються в лексеми, такі як, наприклад, `<ідентифікатор>`, `<число>`, `<ключове слово>` та ін.

Інший модуль компілятора, названий синтаксичним аналізатором, виконує граматичний аналіз, розглядаючи вже лексеми як термінальні символи. Таким чином, у компіляторах граматичний аналіз звичайно здійснюється на двох рівнях: рівні «слів» і рівні «речень». Якщо «слова» не задовольняють граматиці мови, то і саме речення не може задовольняти граматиці даної мови (природної мови або мови програмування). Зворотне не обов'язково вірне – слова можуть бути правильними, але речення, що складене з правильних слів (конструктів мови), може бути синтаксично неправильним.

Вибір переліку лексем залежить від розробника компілятора. В принципі, вони можуть і не виділятися, проте при великих обсягах початкового (вхідного) коду такі компілятори, як правило, витрачають набагато більше часу на синтаксичний аналіз.

Проілюструємо вищесказане на прикладі граматики оператора присвоювання арифметичних виразів (АВ). Нижче наводяться правила цієї граматики з одночасним виділенням лексем у вигляді відповідних підграматик. Оскільки на виході синтаксичного аналізатора повинно бути дерево граматичного розбору у вигляді послідовності застосованих правил, то в граматиці всі правила пронумеровані. Для зручності нумерації в правих частинах правил не записані альтернативні правила. Всі вони записані у вигляді окремих правил.

Граматика оператора присвоєння арифметичного виразу *(G4)*

```
1   <оператор присвоєння АВ>::=<ідентифікатор>:=<АВ>;
2   <АВ>::=<складове>
3   <АВ>::=<АВ><знак додавання/віднімання><складове>
4   <складове>::=<множник>
```

```
5   <складове>::=<складове><знак множення/ділення>
    <множник>
6   <множник>::=<ідентифікатор>
7   <множник>::=<число>
8   <множник>::=(<AB>)
9   <ідентифікатор>::=<літера>
10  <ідентифікатор>::=<ідентифікатор><літера>
11  <ідентифікатор>::=<ідентифікатор><цифра>
12  <літера>::=a
13  <літера>::=b
.....................
37  <літера>::=z
38  <цифра>::=0
39  <цифра>::=1
.....................
47  <цифра>::=9
48  <число>::=<число без знака>
49  <число без знака>::=<ціле без знака>.<ціле без
    знака>
50  <число без знака>::=<ціле без знака>
51  <ціле без знака>::=<цифра>
52  <ціле без знака>::=<ціле без знака><цифра>
53  <знак додавання/віднімання>::=+
54  <знак додавання/віднімання>::=-
55  <знак множення/ділення>::=*
56  <знак множ/діл>::=/
```

Рядки «a4:=(f+1.5/dd)*b2b» «wq:=e+5» є граматично правильними за таких правил, а «A4:=(f+1.5/dd)*b2b» «wq:=-e+5» є граматично неправильними при такій граматиці. У першому випадку неприпустимим є використання літери *«A»*, а у другому випадку в граматиці не передбачено наявності знака перед першим доданком арифметичного виразу. Обидва недоліки граматики можна усунути додаванням додаткових правил: додати до літер і великі літери та додати ще два правила, у яких перед першим доданком буде знак «+» або «-».

Тепер визначимо наступні лексеми (підграматики):

Лексема <ідентифікатор> – підграматика *G5= {9, 10, … , 47}.*

Лексема <число> – підграматика *G6= {48, 49, … , 52}.*

Лексема <знак додавання/віднімання> – підграматика *G7={53, 54}.*

Лексема <знак множення/ділення> – підграматика *G8= {55, 56}*.

Кожна лексема представлена множиною правил, номера яких наведено у фігурних дужках. Після лексичного аналізу виявлені лексеми вже можна розглядати як термінальні символи, тоді граматику оператора присвоєння арифметичного виразу *G4* можна представити наступними правилами (лексеми виділені жирним шрифтом):

1 <оператор присвоєння АВ>::=**G5**:=<АВ>;
2 <АВ>::=<складове>
3 <АВ>::=<АВ>**G7**<складове>
4 <складове>::=<множник>
5 <складове>::=<складове>**G8**<множник>
6 <множник>::=**G5**
7 <множник>::=**G6**
8 <множник>::=(<АВ>)

Таким чином, замість 56 правил граматики *G4* одержали 8 правил, що істотно полегшує синтаксичний аналіз. Нижче наводиться приклад програми мовою Pascal, призначеної для знаходження лексем *G5* (<ідентифікатор>).

```
function CheckIdentifier(St: string;
                         var Error: byte):boolean;
{Перевірка коректності формування ідентифікатора в
алгоритмічній мові Object Pascal. У програмі вико-
ристовуються такі основні об'єкти:
St - ідентифікатор, що перевіряється
Error - номер позиції у ідентифікаторі, де сталася
помилка(рахунок символів з 1)}
var
  I, //лічильник циклу
  L //довжина ідентифікатора
    : byte;
  Flag: Boolean; {якщо ідентифікатор сформовано
правильно, його значення - true, інакше - false}
begin
  I:=1;
  L:=Length(St);
  Error:=0;
```

```
   CheckIdentifier:=true;
   if St[I] in ['0'..'9', ' '..'/'] then
     begin {ідентифікатор починається із цифри -
помилка}
       Error:=I;
       CheckIdentifier:=false;
       Exit;
     end;
   for I:=2 to L do
     begin
       if not(St[I] in ['a'..'z', 'A'..'Z', '0'..'9',
'_']) then
         begin {якщо в ідентифікаторі присутні символи
окрім латинських літер, чисел, або знака підкреслення -
помилка}
           Error:=I; {номер позиції у рядку, яка
є помилковою}
           CheckIdentifier:=false;
           Exit;
         end;
     end;
end; //function CheckIdentifier
```

Програми знаходження лексем *G6, G7* та *G8* є простішими у порівнянні з наведеною. Хай на вході компілятора подано рядок:

```
tax=21.34*(income-deduction)+const5;
```

Результати лексичного аналізу з використанням підграматик *G5, G6, G7* та *G8* можна звести у таблицю типу табл. 3.1.

Таблиця 3.1

Результати лексичного аналізу рядка `tax=21.34*(income-deduction)+const5`

№ пор	Лексема	Тип лексеми	Позиція початку	Позиція кінця
1	tax	G5	1	3
2	21.34	G6	5	9
3	*	G8	10	10
4	income	G5	12	17
5	-	G7	18	18
6	deduction	G5	19	27
7	+	G7	29	29
8	const5	G5	30	35

Таким чином, в задачу сканера входить тільки розпізнавання лексичних одиниць. Тут також припускалося, що речення мови *G4* можуть розділятися між собою проміжками, що ігноруються при визначенні позиції символу.