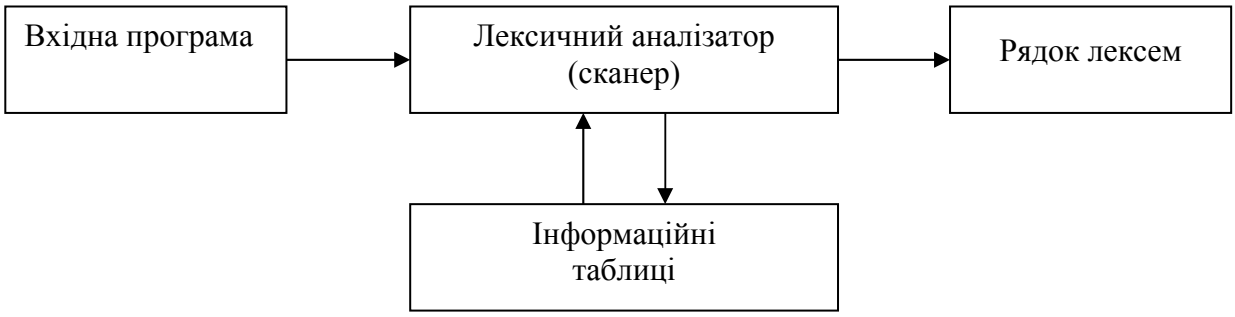
Лекція №3

Програмування лексичного аналізатора (сканера)

Загальні положення

Функції лексичного аналізатора:

1. Виділення та згортання лексем (токенів) із заміною їх символьного виду числовими кодами.
2. Видалення коментарів.
3. Побудова інформаційних таблиць (таблиць ідентифікаторів, числових констант, рядкових констант, символьних констант).
4. Виявлення лексичних помилок (як правило виявляються усі помилки, а не тільки перша).



Приклад побудови лексичного аналізатора (сканера)

1. Допустимі лексеми, що виділяються лексичним аналізатором (ЛА) даного прикладу:
   - Ключові слова;
   - цілі десяткові константи;
   - ідентифікатор
   - одиночні роздільники і знаки операцій;

Крім того, ЛА розпізнає ознаки початку і кінця коментарів (* текст коментаря*) і пропускає текст коментаря без формування лексем.

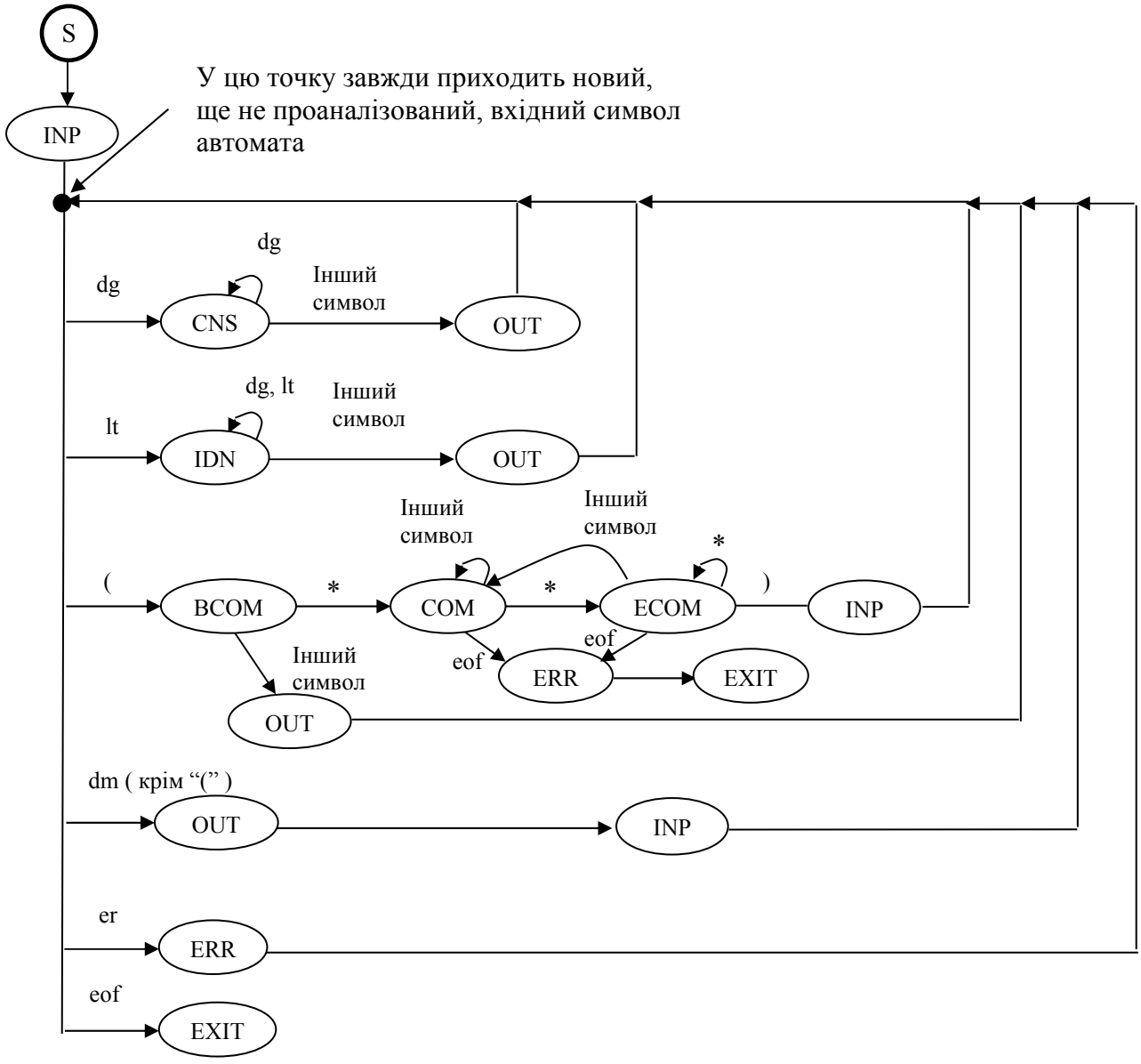
2. Діаграма переходів (граф) автомату ЛА:
   - Позначення станів автомата ЛА:
     - S – початковий стан;
     - INP – стан введення поточного символу програми;
     - CNS – стан виділення константи;
     - IDN – стан виділення ідентифікатора;
     - BCOM – стан визначення символів початоку коментаря;
     - COM – стан пропуску (видалення) символів коментаря;
     - ECOM – стан визначення символів кінеця коментаря;
     - ERR – стан обробки помилки та видача повідомлення про помилку;
     - OUT – стан виведення лексеми;
     - EXIT – кінцевий стан.

- Позначення вхідних символів:
  - dg – цифра (0..9), тобто вхідний символ dg встановлюється, якщо поточний прочитаний символ програми є цифрою;
  - lt – буква (A .. Z, a .. z), тобто вхідний символ lt встановлюється, якщо поточний прочитаний символ програми є буквою;
  - dm – роздільник, тобто вхідний символ dm встановлюється, якщо поточний прочитаний символ програми є роздільником;
  - er – помилковий символ, тобто вхідний символ er встановлюється, якщо поточний прочитаний символ програми є неприпустимим для даної мови (такими, як правило, є більшість керуючих (управляючих) символів ASCII з діапазону 0..31, а також іноді деякі друковані символи);
  - eof – символ кінця файлу.

У всіх станах, окрім OUT, виконується введення чергового символу програми і визначення вхідного символу автомата ЛА.

В стані OUT виконується пошук виділеного ідентифікатора в таблицях ключових слів ідентифікаторів та константи в таблиці констант і внесення їх до цих таблиць (якщо потрібно), виведення коду сформованої лексеми у вихідний масив(файл) лексем і передача наступного символу (якщо він був вже введений) на подальший аналіз.

В стані ERR лексема не формується, а виводиться повідомлення про помилку.



3. Опис змінних, використаних в прикладі програми сканера:
   - symbol.value – код ASCII поточного символу;
   - symbol.attr – клас лексем, до якого належить поточний символ symbol.value (кожному символу таблиці ASCII заздалегідь присвоюється атрибут, що визначає лексему (токен), яка може починатися з цього символу):
     (0) пробіл і прирівняні до нього символи (whitespace);
     (1) ціла константа;
     (2) ідентифікатор;
     (3) символ початку коментаря;
     (4) роздільник;
     (5) помилковий символ;
   - buf – буфер для накопичення символів поточної лексеми;
   - lexCode – код чергової лексеми;
   - Attributes – масив значень атрибутів для кожного символу ASCII;
   - FINP – файл початкової програми;
   - SuppressOutput – ознака того, що була виявлена послідовність пробілів або коментар, які не потрібно записувати у вихідний файл.

   Приклад можливого кодування лексем:
   - діапазон кодів роздільників: 0..255 (ASCII коди);
   - діапазон кодів рядків: 301..400;
   - діапазон кодів констант: 401..500;
   - діапазон кодів ідентифікаторів: 501..600.

4. Функції, використані в прикладі програми сканера:
   - Gets – читає поточний символ з вхідної програми, визначає його атрибут по масиву Attributes і повертає запис symbol;
   - ShowError – обробка помилок, видача повідомлення про помилки;
   - IdnTabForm – формування таблиці ідентифікаторів;
   - ConstTabForm – формування таблиці констант;
   - IdnTabSearch – пошук в таблиці ідентифікаторів;
   - KeyTabSearch – пошук в таблиці ключових слів;
   - ConstTabSearch – пошук в таблиці констант.

5. Структура програми сканера

```
program Scanner;

{$APPTYPE CONSOLE}

uses
  SysUtils;
type
  TSymbol = record
    value: Char;
    attr: Byte;
  end;
```

```
{ Опис типів таблиць }
var
  Attributes: array [Char] of Byte;
  symbol: TSymbol;
  lexCode: Word;
  buf: string;
  SuppressOutput: Boolean;
  FINP: TextFile;

{Опис таблиць}
function Gets: TSymbol;
begin
  Read(FINP, Result.value);
  Result.attr := Attributes[Result.value];
end;

begin
  (*відкриття файлу початкової програми*)
  (*початкове встановлення таблиць ідентифікаторів і констант*)
  FillAttributes;
  if eof(FINP) then
    ShowError('Empty file');
  repeat
    symbol := Gets;
    buf := '';
    lexCode := 0;
    SuppressOutput := False;
    case symbol.attr of
      0: (*whitespace*)
      begin
        while not eof(FINP) do
        begin
          symbol := Gets;
          if symbol.attr <> 0 then
            Break;
        end;
        SuppressOutput := True;
      end;
      1: (*константа*)
      begin
        while not eof(FINP) and (symbol.attr = 1) do
        begin
          buf := buf + symbol.value;
          symbol := Gets;
        end;
        if ConstTabSearch then
          lexCode := <код константи>
        else
        begin
          lexCode := <код наступної константи>
          ConstTabForm;
        end;
      end;
```

```
2: (*ідентифікатор*)
begin
  while not eof(FINP) and ((symbol.attr = 2)
    or (symbol.attr = 1)) do
  begin
    buf := buf + symbol.value;
    symbol := Gets;
  end;
  if KeyTabSearch then
    lexCode := <код ключового слова>
  else
    if IdnTabSearch then
      lexCode :=  <код ідентифікатора>
    else
    begin
      lexCode := <код наступного ідентифікатора>
      IdnTabForm;
    end;
end;

3: (*можливий коментар, тобто зустрінута '(' *)
begin
  if eof(FINP) then
    lexCode := <код відкриваючої дужки>
  else
  begin
    symbol := Gets;
    if symbol.value = '*' then
    begin
      if eof(FINP) then
        ShowError('*) expected but end of file found');
      else
      begin
        symbol := Gets;
        repeat
          while not eof(FINP) and (symbol.value <> '*') do
            symbol := Gets;
          if eof(FINP) then //якщо кінець файла
          begin
            ShowError('*) expected but end of file found');
            symbol.value = '+'; // все що завгодно, але не ')'
            Break;
          end
          else //була '*'  і немає кінця файла
            symbol := Gets;
        until symbol.value = ')';
        if symbol.value = ')' then
          SuppressOutput := True;
        if not eof(FINP) then
          symbol := Gets;
      end;
    end
```

```
          else
          begin
            lexCode := <код відкриваючої дужки>
          end;
        end;
      end;
      4: (*роздільник окрім '(' *)
      begin
        symbol := Gets;
        lexCode := <ASCII код односимвольного роздільника>
      end;
      5: (*помилка*)
      begin
        ShowError('Illegal symbol');
        symbol := Gets;
      end;
    end;       (*case*)
    if not SuppressOutput then
      writeln('Output: ',' ',lexCode);
  until eof(FINP);
  Readln;
end.
```

У випадку, якщо в граматиці є багатосимвольні роздільники, то вони заздалегідь вносяться до окремої таблиці і обробляються аналогічно ключовим словам. Якщо багатосимвольний роздільник не розпізнаний, то треба забезпечити повернення у тексті початкової програми до попереднього виділеного символу–роздільника.