МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра
Автоматизованих систем обробки інформації та управління

Лабораторна робота №4

з дисципліни

«Теоретичні основи лексичного та синтаксичного аналізу»

на тему:

Розробка лексичного аналізатора

Варіант 16

Перевірив:

Селін Ю. М.

Виконав:

Педоренко О. Р.

Студент груп ІП-31,
факультету ІОТ,
залікова книжка ІП-3114

Київ 2016

1. Мета роботи

Побудувати лексичний аналізатор для процедурної або функціональної мови програмування.

2. Завдання на роботу

1. По варианту задания определить, какие классы лексем будут в вашем языке.

2. Составить контрольные примеры на реализуемом языке. Хотя бы один пример должен проверять поведение вашей программы при наличии недопустимых символов в транслируемом файле.

3. Запрограммировать и отладить модуль сканирования. Выполнить тестирование на контрольных примерах. Результатом работы должна быть таблица, содержащая лексемы и признаки их классов. Необходимо включить в результирующий файл информацию о номерах строк исходного текста транслируемой программы.

3. Варіант індивідуального завдання

16

Подмножество языка PASCAL включает:

<Программа> ::= <Объявление переменных> <Описание вычислений> .
<Описание вычислений> ::= Begin <Список присваиваний> End
<Объявление переменных> ::= Var <Список переменных>
<Список переменных> ::= <Идент> | <Идент> , <Список переменных>
<Список присваиваний>::= <Присваивание> |
<Присваивание> <Список присваиваний>
<Присваивание> ::= <Идент> = <Выражение>
<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>
<Подвыражение> :: = (<Выражение>) | <Операнд> |
< Подвыражение > <Бин.оп.> <Подвыражение>
<Ун.оп.> ::= "-"
<Бин.оп.> ::= "-" | "+" | "*" | "/"
<Операнд> ::= <Идент> | <Const>
<Идент> ::= <Буква> <Идент> | <Буква>
<Const> ::= <Цифра> <Const> | <Цифра>

На одной строке может быть только объявление переменных или один оператор присваивания.

4. Виконання роботи

1) З варіанту завдання видно, що у мові є наступні класи лексем: ключові слова, ідентифікатори, оператори та константи.

2) Текст прикладів наведено нижче.

a)

```
var a,b,c,d
begin
a = 10
b = 20
d = 30
d = a+b/c+10
end
```

b)

```
var a,b,c,d
begin
a = 10
b = 20
d = 30
d = a+b/c+10
(_)
end
```

3)

a) Результат роботи на прикладі a:

```
E:\DATA\Downloads\gplex-distro-1_2_2\gplex-distro-1_2_2\testfiles>lab4 a.pas
File: a.pas
keyword:     var (1:0)
identifier:  a (1:4)
identifier:  b (1:6)
identifier:  c (1:8)
identifier:  d (1:10)
keyword:     begin (2:0)
identifier:  a (3:0)
assignment:  = (3:2)
constant:    10 (3:4)
identifier:  b (4:0)
assignment:  = (4:2)
constant:    20 (4:4)
identifier:  d (5:0)
assignment:  = (5:2)
constant:    30 (5:4)
identifier:  d (6:0)
assignment:  = (6:2)
identifier:  a (6:4)
operator:    + (6:5)
identifier:  b (6:6)
operator:    / (6:7)
identifier:  c (6:8)
operator:    + (6:9)
constant:    10 (6:10)
keyword:     end (7:0)
Elapsed time:   24 msec
```

b) Результат роботи на прикладі b:

```
E:\DATA\Downloads\gplex-distro-1_2_2\gplex-distro-1_2_2\testfiles>lab4 b.pas
File: b.pas
keyword:     var (1:0)
identifier:  a (1:4)
identifier:  b (1:6)
identifier:  c (1:8)
identifier:  d (1:10)
keyword:     begin (2:0)
identifier:  a (3:0)
assignment:  = (3:2)
constant:    10 (3:4)
identifier:  b (4:0)
assignment:  = (4:2)
constant:    20 (4:4)
identifier:  d (5:0)
assignment:  = (5:2)
constant:    30 (5:4)
identifier:  d (6:0)
assignment:  = (6:2)
identifier:  a (6:4)
operator:    + (6:5)
identifier:  b (6:6)
operator:    / (6:7)
identifier:  c (6:8)
operator:    + (6:9)
constant:    10 (6:10)
error:       (_) (7:0)
keyword:     end (8:0)
Elapsed time:   31 msec
```

Код програми:

```
%namespace lab4
%option noparser, caseInsensitive

alpha [a-zA-Z]
digits [0-9]+
error [^a-zA-Z0-9\+\-\*\/=,\ \r\n]
operators [\+\-\*\/]
%%

begin |
end |
var             Console.WriteLine("keyword:     " + yytext + " ("+ yyline + ":" + yycol +")");
{alpha}+        Console.WriteLine("identifier:  " + yytext + " ("+ yyline + ":" + yycol +")");
=               Console.WriteLine("assignment:  " + yytext + " ("+ yyline + ":" + yycol +")");
{operators}     Console.WriteLine("operator:    " + yytext + " ("+ yyline + ":" + yycol +")");
{digits}        Console.WriteLine("constant:    " + yytext + " ("+ yyline + ":" + yycol +")");
{error}+        Console.WriteLine("error:       " + yytext + " ("+ yyline + ":" + yycol +")");

%%

   public static void Main(string[] argp) {
      DateTime start = DateTime.Now;
      int count = 0;
      if (argp.Length == 0)
         Console.WriteLine("Usage: WordCount filename(s), (wildcards ok)");
      DirectoryInfo dirInfo = new DirectoryInfo(".");
      for (int i = 0; i < argp.Length; i++) {
         string name = argp[i];
         FileInfo[] fInfo = dirInfo.GetFiles(name);
         foreach (FileInfo info in fInfo)
         {
                try {
                      int tok;
                      FileStream file = new FileStream(info.Name, FileMode.Open);
                      Scanner scnr = new Scanner(file);
                      Console.WriteLine("File: " + info.Name);
                      do {
                         tok = scnr.yylex();
                      } while (tok > (int)Tokens.EOF);
                      count++;
                   } catch (IOException) {
                      Console.WriteLine("File " + name + " not found");
                   }
         }
      }
      TimeSpan span = DateTime.Now - start;
      Console.WriteLine("Elapsed time: {0,4:D} msec", (int)span.TotalMilliseconds);
   }
```