Белорусский государственный университет информатики и радиоэлектроники

Кафедра информатики

Лабораторная работа № 1

**Лексический анализатор**

Выполнил:

Проверил:

Шиманский В.В.

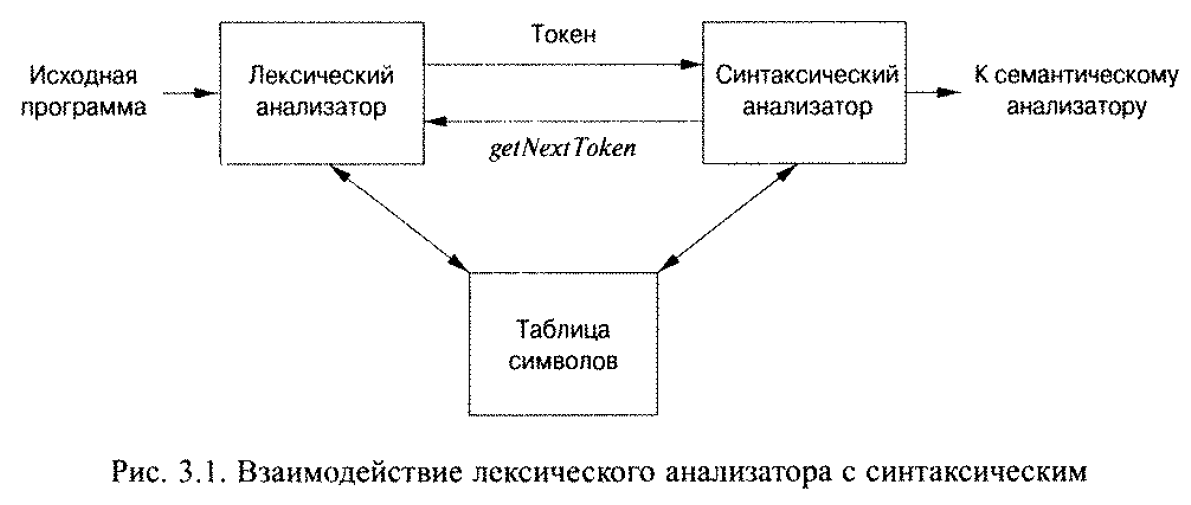
Минск, 2013

1. ***Постановка задачи:***

Освоение работы с существующими лексическими анализаторами. Разработать собственный лексический анализатор языка/подмножества языка программирования для чего определить лексические правила и выполнить перевод потока символов в поток лексем (токенов).

1. ***Теория:***

Лексический анализатор представляет собой первую фазу компиляции, его основная задача состоит в чтении входных символов исходной программы, их группировании в лексемы и вывод последовательностей токенов для всех лексем исходной программы. Поток токенов пересылается синтаксическому анализатору для разбора. Обычно при работе лексический анализатор взаимодействует также с таблицей символов. Когда лексический анализатор встречается с лексемой, составляющей идентификатор, эту лексему требуется внести в таблицу символов. В некоторых случаях лексический анализатор может получать из таблицы символов некоторую информацию об идентификаторах, которая может помочь ему верно определить передаваемый синтаксическому анализатору токен.



Поскольку лексический анализатор является частью компилятора, которая читает исходный текст, он может заодно выполнять и некоторые другие действия, помимо идентификации лексем. Одной из таких задачявляется отбрасывание комментариев и пробельных символов.

1. ***Программа и комментарии:***

Первая вещь, которую мы делаем – это определяем набор названий для наших токенов.  
RESERVED – определяет зарезервированное слово или оператор  
NUMBER – определяет число  
STRING – определяет строку  
ID – для идентификаторов

RESERVED = 'RESERVED'

NUMBER = 'NUMBER'

STRING = 'STRING'

ID = 'ID'

Далее определяются выражения, которые будут использоваться в лексере. Первые 2 выражения определяют пробелы и комментарии. Далее, у нас определены все зарезервированные слова и операторы. В конце у нас выражения для строковых и числовых данных, а также идентификаторов.

token\_exprs = [

(r'[ \n\t]+', None),

(r'#[ ^\n]\*', None),

(r'\=', RESERVED),

(r'\,', RESERVED),

(r'\(', RESERVED),

(r'\)', RESERVED),

(r';', RESERVED),

(r'\+', RESERVED),

(r'\-', RESERVED),

(r'\\*', RESERVED),

(r'\\', RESERVED),

(r'%', RESERVED),

(r'<=', RESERVED),

(r'<', RESERVED),

(r'>=', RESERVED),

(r'>', RESERVED),

(r'!=', RESERVED),

(r'and', RESERVED),

(r'or', RESERVED),

(r'not', RESERVED),

(r'if', RESERVED),

(r'else', RESERVED),

(r'print', RESERVED),

(r'while', RESERVED),

(r'for', RESERVED),

(r'in', RESERVED),

(r'\:', RESERVED),

(r'\[', RESERVED),

(r'\]', RESERVED),

(r'\"([^\"]+)\"', STRING),

(r'\'([^\"]+)\'', STRING),

(r'[0-9]+', NUMBER),

(r'[A-Za-z][A-Za-z0-9\_]\*', ID),

]

Определяем функцию лексера: перевод потока символов в поток лексем

def lex(characters, token\_exprs):

pos = 0

tokens = []

while pos < len(characters):

match = None

for token\_expr in token\_exprs:

pattern, tag = token\_expr

regex = re.compile(pattern)

match = regex.match(characters, pos)

if match:

text = match.group(0)

if tag:

token = (text, tag)

tokens.append(token)

break

if not match:

sys.stderr.write('Illegal character: %s\n' % characters[pos])

else:

pos = match.end(0)

return tokens

Вызов функции

def lex\_an(characters):

return lex(characters,token\_exprs)

Открываем файл с кодом и переводим.

filename="E:/input.txt"

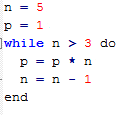
file = open(filename)

characters = file.read()

file.close()

tokens = lex\_an(characters)

print(tokens)

Исходный текст программы:  
  
Результат:  
