Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий

Кафедра Информатики

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
Разработка блока лексического анализа транслятора

Преподаватель _____ А.С. Кузнецов
подпись, дата

Студент гр. КИ10-11 031010132 _____ К.О. Васильев
подпись, дата

Красноярск 2013

1 Цель работы

Изучение методов лексического анализа с их программной реализацией.

2 Постановка задачи

Разработать программу, осуществляющую лексический анализ простого языка программирования. Входной язык содержит операторы цикла с параметром, разделенные символом точки с запятой (';'). Операторы цикла состоят из идентификаторов, целочисленных констант в шестнадцатеричной системе, знаков присваивания (':='), знаков операций сравнения (<, >, =, ==, <>, !=, >=, <=) и круглых скобок.

3 Полученная спецификация программы

```
# type names
SEMICOLON = 'SEMICOLON'
ID = 'ID'
FOR = 'FOR'
HCONST = 'HCONST'
ASSIGN = 'ASSIGN'
GREATER = 'GREATER'
LESS = 'LESS'
EQUALS = 'EQUALS'
DEQUALS = 'DEQUALS'
GRLESS = 'GRLESS'
NOTEQUALS = 'NOTEQUALS'
GREQUALS = 'GREQUALS'
LSEQUALS = 'LSEQUALS'
LPAR = 'LPAR'
RPAR = 'RPAR'

# reserved words
reserved = (FOR,)

tokens = reserved + (
   SEMICOLON,
   ID,
   HCONST,  #hex constant
```

```
    ASSIGN,
    GREATER,
    LESS,
    EQUALS,
    DEQUALS,  #double equals
    GRLESS,
    NOTEQUALS,
    GREQUALS,
    LSEQUALS,
    LPAR,
    RPAR
)

t_ignore = ' \t'


def t_NEWLINE(t):
    r'\n+'
    t.lexer.lineno += t.value.count('\n')


t_SEMICOLON = r';'
t_ASSIGN = r':='
t_LPAR = r'\('
t_RPAR = r'\)'
t_HCONST = r'-?0x[0-9a-fA-F]+'

# comparison operators
t_GREATER = r'>'
t_LESS = r'<'
t_EQUALS = r'='
t_DEQUALS = r'=='
t_GRLESS = r'<>'
t_NOTEQUALS = r'!='
t_GREQUALS = r'>='
t_LSEQUALS = r'<='

reserved_map = {}
for r in reserved:
    reserved_map[r.lower()] = r


def t_ID(t):
    r'[A-Za-z_][\w_]*'
    t.type = reserved_map.get(t.value, ID)
    return t

from ply.lex import LexError
class IllegalTokenException(LexError):
    def __init__(self, character, line_number):
```

```
        self.character = character
        self.line_number = line_number

    def __str__(self):
        return "Illegal character '{0}' at line {1}".format(self.character, self.line_number)


def t_error(t):
    t.lexer.skip(1)
    raise IllegalTokenException(t.value[0], t.lineno)
```

4 Тестовые примеры работы программы

Передадим программе текстовый файл с представленным кодом:

for (0xf;a!=b;

Тогда на выходе получим следующую таблицу:

```
FOR:        for         line:1
LPAR:       (           line:1
HCONST:     0xf         line:1
SEMICOLON:  ;           line:1
ID:         a           line:1
NOTEQUALS:  !=          line:1
ID:         b           line:1
SEMICOLON:  ;           line:1
```

Теперь передадим программе файл, содержащий код:

>

<

123

Результатом работы программы будет следующая таблица:

```
GREATER:    >           line:1
LESS:       <           line:2
Illegal character '1' at line 3
Illegal character '2' at line 3
Illegal character '3' at line 3
```

Если передать программе файл, содержащий следующий код:

a:=(for 0xf>0x0; !=a)

То на выходе будет получена следующая таблица:

ID:	a	line:1	
ASSIGN:	:=	line:1	
LPAR:	(	line:1	
FOR:	for	line:1	
HCONST:	0xf	line:1	
GREATER:	>	line:1	
HCONST:	0x0	line:1	
SEMICOLON:	;	line:1	
NOTEQUALS:	!=	line:1	
ID:	a	line:1	
RPAR:	)	line:1	