Μεταγλωττιστές 2019
Προγραμματιστική Εργασία #2

Ονοματεπώνυμο: Μιχαήλ-Χ. Παγκρακιώτης
Α.Μ.: Π2014035

- -

1. Γραμματική:

Stmt_list	**→ Stmt Stmt_list | ε**
Stmt	**→ id equals Expr | print Expr**
Expr	**→ Term Term_tail**
Term_tail	**→ xor Term Term_tail | ε**
Term	**→ Factor Factor_tail.**
Factor_tail	**→ or Factor Factor_tail | ε**
Factor	**→ Atom Atom_tail**
Atom_tail	**→ and Atom Atom_tail | ε**
Atom	**→ lPar Expr rPar | id | number**

2. Αποτελέσματα ελέγχου συμβατότητας LL(1): "The grammar is LL(1)", FIRST & FOLLOW sets:

Grammar

```
Stmt_list →   Stmt Stmt_list
             | .
Stmt →        id equals Expr
             | print Expr.
Expr →        Term Term_tail.
Term_tail →   xor Term Term_tail
             | .
Term →        Factor Factor_tail.
Factor_tail → or Factor Factor_tail
             | .
Factor →      Atom Atom_tail.
Atom_tail →   and Atom Atom_tail
             | .
Atom →        lPar Expr rPar
             | id
             | number.
```

Some sentences generated by this grammar: {ε, print id, print number, id equals id, print id and id, id equals number, print id and number, print number and id, id equals id and id, id equals number and id, print number and number, id equals id and number, id equals id and id and id, id equals number and number, id equals id and id and number, id equals id and number and id, id equals number and id and id, id equals id and number and number, id equals number and id and number, id equals number and number and id}

- All nonterminals are reachable and realizable.
- The nullable nonterminals are: Stmt_list Term_tail Factor_tail Atom_tail.
- The endable nonterminals are: Atom_tail Atom Factor_tail Factor Term_tail Term Expr Stmt_list Stmt.
- No cycles.

nonterminal	first set	follow set	nullable	endable
Stmt_list	id print	∅	yes	yes
Stmt	id print	id print	no	yes
Term_tail	xor	rPar id print	yes	yes
Term	lPar id number	rPar xor id print	no	yes
Factor_tail	or	rPar xor id print	yes	yes
Factor	lPar id number	rPar or xor id print	no	yes
Atom_tail	and	rPar or xor id print	yes	yes
Atom	lPar id number	rPar and or xor id print	no	yes
Expr	lPar id number	rPar id print	no	yes

The grammar is LL(1).

3. Αποτελέσματα εξόδου:

Αποτελέσματα εξόδου δεν έχω από τον **runner.py** καθώς δεν κατάφερα να τον φέρω σε σημείο που να μπορεί να εκτελεί πράξεις, να αποθηκεύει και να εμφανίζει τιμές.

Ο **parser.py** αναγνωρίζει τις συντακτικά σωστές εισόδους χωρίς να εμφανίζει μηνύματα λάθους. Αναφέρω μερικά ενδεικτικά παραδείγματα εισόδου παρακάτω:

3.1. Για συντακτικά σωστές εισόδους στον parser:

INPUT:
```
a = 0010100
b = 1000110
c = a xor b and 10001
print c
print a or b xor c and 100101
```

OUTPUT: **No errors**

INPUT:
```
a = 001
b = 101
c = a xor b
print c
```

OUTPUT: **No errors**

INPUT:
```
a = 10101
b = 11111
c = 10001
d = 11011
print a and ((b or c) xor d)
```

OUTPUT: **No errors**

3.2. Για συντακτικά λανθασμένες εισόδους στον parser:

INPUT:
```
a = 001
b = 101
c = a xor )
print c
```

OUTPUT: **Parser Error: in term: (, id or binary number expected at line 3 char 11**

INPUT:
```
a = 001
b = 101
c = print
print c
```

OUTPUT: **Parser Error: in expr: (, id or binary number expected at line 3 char 5**

INPUT:
```
a = 100
print b = a
```

OUTPUT: **Parser Error: in atom_tail: and,), or, xor, id or print expected at line 2 char 9**

4. Αναφορά σε πηγές που χρησιμοποίησα:

- Το online εργαλείο της εκφώνησης για την εύρεση των First και Follow sets της γραμματικής, από τα οποία προέκυψε ο κώδικας: http://smlweb.cpsc.ucalgary.ca/start.html