Μεταγλωτιστές 2018

Προγραμματιστική Εργασία στους Μεταγλωττιστές #2

Ονοματεπώνυμο: Πλούμης Ευθύμιος

ΑΜ: Π2012054

Ερωτήματα :

1) ΚΑΝΟΝΕΣ ΓΡΑΜΜΑΤΙΚΗΣ

<Program> -> Stmt_list # (βοηθητικός κανόνας!)

Stmt_list -> Stmt Stmt_list | ε

Stmt -> id = Expr | print Expr

Expr -> OrExpresion OrExpresion_tail

OrExpresion_tail -> or OrExpresion OrExpresion_tail | ε

OrExpresion -> AndExpresion AndExpresion_tail

AndExpresion_tail -> and AndExpresion AndExpresion_tail | ε

AndExpresion -> NotExpresion NotExpresion_tail

NotExpresion -> not | ε

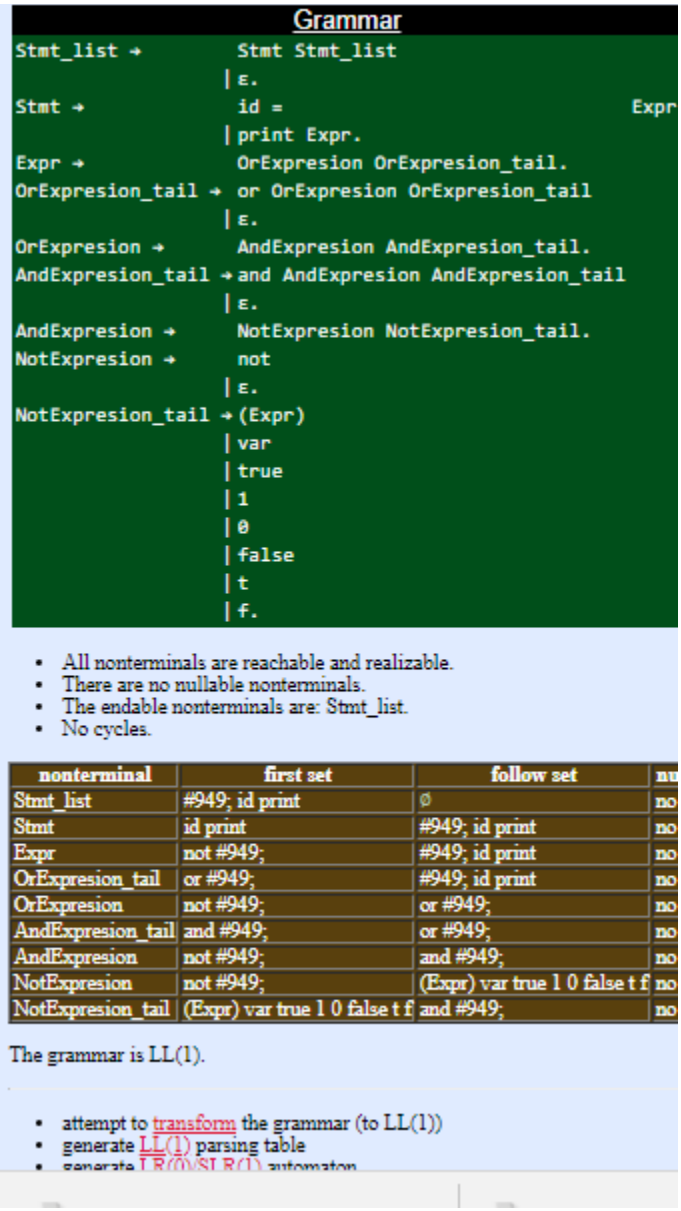
NotExpresion_tail -> (Expr) | var | true | 1 | 0 | false | t | f

print: το keyword 'print'

id, var: όνομα μεταβλητής

2. Το αποτέλεσμα του ελέγχου

φαίνεται παρακάτω:



Grammar

```
Stmt_list →         Stmt Stmt_list
                  | ε.
Stmt →              id =                                Expr
                  | print Expr.
Expr →              OrExpresion OrExpresion_tail.
OrExpresion_tail →  or OrExpresion OrExpresion_tail
                  | ε.
OrExpresion →       AndExpresion AndExpresion_tail.
AndExpresion_tail → and AndExpresion AndExpresion_tail
                  | ε.
AndExpresion →      NotExpresion NotExpresion_tail.
NotExpresion →      not
                  | ε.
NotExpresion_tail → (Expr)
                  | var
                  | true
                  | 1
                  | 0
                  | false
                  | t
                  | f.
```

- All nonterminals are reachable and realizable.
- There are no nullable nonterminals.
- The endable nonterminals are: Stmt_list.
- No cycles.

nonterminal	first set	follow set	nu
Stmt_list	#949; id print	∅	no
Stmt	id print	#949; id print	no
Expr	not #949;	#949; id print	no
OrExpresion_tail	or #949;	#949; id print	no
OrExpresion	not #949;	or #949;	no
AndExpresion_tail	and #949;	or #949;	no
AndExpresion	not #949;	and #949;	no
NotExpresion	not #949;	(Expr) var true 1 0 false t f	no
NotExpresion_tail	(Expr) var true 1 0 false t f	and #949;	no

The grammar is LL(1).

- attempt to transform the grammar (to LL(1))
- generate LL(1) parsing table
- generate LR(0)/SLR(1) automaton

3) **Πίνακες με τα FIRST και FOLLOW sets**

Rule	FIRST	FOLLOW
Stmt_list	id, print, ε	#
Stmt	id, print	id, print, #
Expr	NOT, (, var, true, 1, 0, false, t, f	), id, print, #
OrExpresion_tail	OR, ε	), id, print, #
OrExpresion	NOT, (, var, true, 1, 0, false, t, f	OR,) ,id , print, #
AndExpresion_tail	AND, ε	OR,) ,id , print,#
AndExpresion	NOT, (, var, true, 1, 0, false, t, f	AND, OR,) ,id , print, #
NotExpresion	NOT, ε	(, var, true, 1, 0, false, t, f
NotExpresion_tail	(, var, true, 1, 0, false, t, f	AND, OR,) ,id , print, #

4. ΚΩΔΙΚΑΣ

parser.py

Στον κώδικα parser, έχοντας σαν βάση τον κώδικα που δόθηκε στην εκφώνηση, υλοποιήσαμε έναν αναλυτή για την παραπάνω γραμματική. Ο αναλυτής αντιλαμβάνεται τις κώδικές λέξεις id και print και στην συνέχεια ελέγχει την κάθε μία λογική πράξη που δίνεται. Αν κάποια έκφραση είναι λάθος τότε σταματά και ενημερώνει στον συγκεκριμένο κανόνα τι αναμένει σαν είσοδο.

Αν όλη η έκφραση είναι σωστή τότε ο αναλυτής τερματίζει χωρίς μήνυμα λάθους.

runner.py

Στον κώδικα runner αφού πρώτα τρέξουμε τον κώδικα του parser και ελέγξουμε την λογική έκφραση, εκτελούμε την λογική έκφραση στην python και εκτυπώνουμε το αποτέλεσμα.

5 ΑΡΧΕΙΟ log εκτέλεσης.txt

Στο αρχείο log εκτέλεσης.txt υπάρχουν μερικά παραδείγματα εκτέλεσης, τέτοια όπου είτε ο parser και η python θεώρησαν ότι είναι λάθος , είτε άλλα όπου ο Parser θεώρησε ότι είναι σωστά αλλά η python δεν γνωρίζει τους τελεστές, πχ: τελεστής f

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ