

# Ιόνιο Πανεπιστήμιο



Μεταγλωττιστές  
Διδάσκων: Στεφανιδάκης Μιχαήλ

Προγραμματιστική εργασία 2: Κατασκευή  
top-down συντακτικού αναλυτή με τη  
μέθοδο της αναδρομικής κατάβασης

Κωνσταντίνος Σιμώνης Π2016113

# Εισαγωγή

Η συγκεκριμένη άσκηση έγινε στα πλαίσια του μαθήματος Μεταγλωττιστές του ΣΤ΄ εξαμήνου του τμήματος Πληροφορικής του Ιονίου Πανεπιστημίου.

Σκοπός της άσκησης ήταν η δημιουργία/τροποποίηση κώδικα ώστε να δημιουργηθεί top-down συντακτικός αναλυτής αλλά και διερμηνέας για λογικές εκφράσεις. Η πλήρης εκφώνηση της άσκησης μπορεί να βρεθεί [εδώ](#).

## Δημιουργία κατάλληλης γραμματικής

Για την δημιουργία της κατάλληλης γραμματικής τροποποιήθηκε η υπάρχουσα η οποία εξυπηρετεί συντακτικό αναλυτή για αριθμητικές εκφράσεις. Η συγκεκριμένη μπορεί να βρεθεί [εδώ](#).

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

Stmt_list      ->      Stmt Stmt_list | ε
Stmt           ->      id = Expr | print Expr
Expr           ->      Term Term_tail
Term_tail      ->      Addop Term Term_tail | ε
Term           ->      Factor Factor_tail
Factor_tail    ->      Multop Factor Factor_tail | ε
Factor         ->      (Expr) | id | number
Addop          ->      + | -
Multop         ->      * | /

print: το keyword 'print'
id: όνομα μεταβλητής
number: αριθμητική σταθερά
```

Εικόνα 1: Η αρχική γραμματική

Με στόχο την δημιουργία LL(1) γραμματικής που να υποστηρίζει τις λογικές εκφράσεις And, Or και Xor, προσθέσαμε τις καταστάσεις Atom και Atom\_tail ώστε σε κάθε έκφραση να αντιστοιχεί ένα ζευγάρι καταστάσεων και κανόνων. Καθώς η προτεραιότητα μας είναι ξεκάθαρη και δεν υπάρχουν τελεστές με ίδια προτεραιότητα οι κανόνες Addop και Multop δεν χρειάζονται πλέον και αφαιρούνται. Έτσι προκύπτει η εξής γραμματική (Γραμμένη σύμφωνα με το πρότυπο του [εργαλείου](#)):

Stmtlist -> Stmt Stmtlist

| .

Stmt -> id equal 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 -> anparen Expr klparen

|id

|number.

Στη συγκεκριμένη γραμματική:

id = όνομα μεταβλητής

equal = τελεστής ισότητας

print = To keyword 'print'

xor = ο λογικός τελεστής Xor

or = ο λογικός τελεστής Or

and = ο λογικός τελεστής And

anparen = Η ανοιχτή παρένθεση '('

klparen = Η κλειστή παρένθεση ')'

Παρατηρούμε επίσης ότι οι τελεστές με την μεγαλύτερη προτεραιότητα έχουν τοποθετηθεί στα βαθύτερα 'στρώματα' της γραμματικής και αυτό γιατί η μέθοδος της αναδρομικής κατάβασης εκτελεί τα βαθύτερα 'στρώματα' πρώτα.

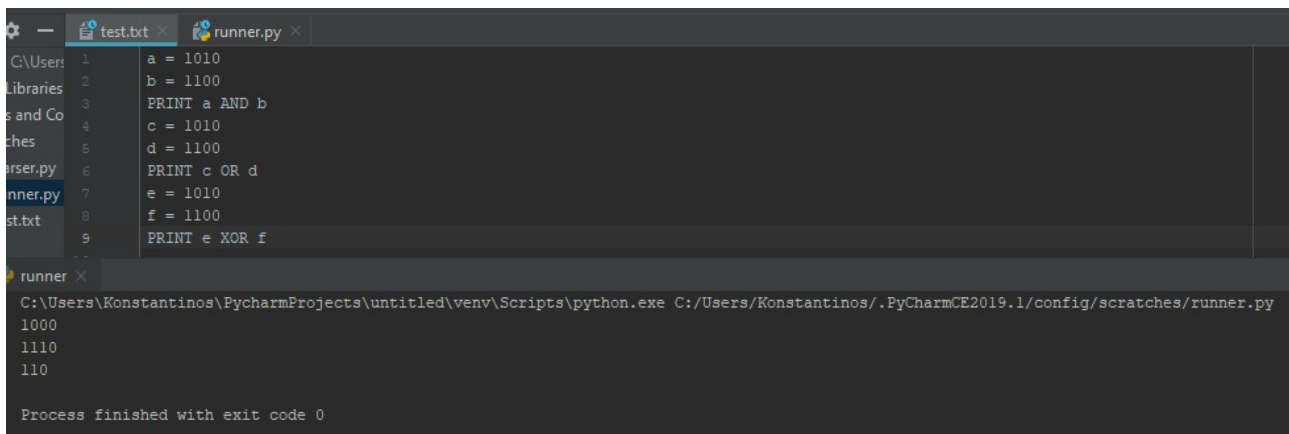
Έπειτα, με χρήση του προαναφερθέντος [εργαλείου](#) βγάλαμε τα first και τα follow sets τα οποία βοηθούν στην υλοποίηση της γραμματικής σε εκτελέσιμο κώδικα. Τα first και τα follow sets είναι:

| nonterminal | first set         | follow set                  |
|-------------|-------------------|-----------------------------|
| Stmntlist   | id print          | ∅                           |
| Stmnt       | id print          | id print                    |
| Term_tail   | xor               | klparen id print            |
| Term        | anparen id number | klparen xor id print        |
| Factor_tail | or                | klparen xor id print        |
| Factor      | anparen id number | klparen or xor id print     |
| Atom_tail   | and               | klparen or xor id print     |
| Atom        | anparen id number | klparen and or xor id print |
| Expr        | anparen id number | klparen id print            |

## Τροποποίηση κώδικα και αποτελέσματα

Αφού δημιουργήσαμε τη γραμματική προσθέσαμε στον υπάρχοντα κώδικα τις συναρτήσεις `atom` και `atom_tail` που αναπαριστούν τους αντίστοιχους κανόνες και κάναμε κάποιες επιπλέον μετατροπές ώστε να αναγνωρίζονται οι λογικοί τελεστές αντί για τους αριθμητικούς που αναγνωρίζονταν προηγουμένως.

Ο κώδικας `parser.py` δεν έχει τίποτα σαν έξοδο καθώς δεν εκτελεί και καθήκοντα `interpreter`. Από την άλλη ο `runner.py` δίνει τα εξής:



```
1 a = 1010
2 b = 1100
3 PRINT a AND b
4 c = 1010
5 d = 1100
6 PRINT c OR d
7 e = 1010
8 f = 1100
9 PRINT e XOR f
```

```
C:\Users\Konstantinos\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/Konstantinos/.PyCharmCE2019.1/config/scratches/runner.py
1000
1110
110

Process finished with exit code 0
```