

- Κανόνες της Γραμματικής:

Stmt\_list -> Stmt Stmt\_list

|.

Stmt -> id = Expr

| print Expr.

Expr -> Term Term\_tail.

Term\_tail -> Aoop Term Term\_tail

|.

Term -> Factor Factor\_tail.

Factor\_tail -> Nop Factor Factor\_tail

|.

Factor -> (Expr)

| id

| tf01.

Aoop -> and

| or.

Nop -> not.

- Από το παρακάτω στιγμιότυπο και με τη βοήθεια του online εργαλείου <http://smlweb.cpsc.ucalgary.ca/start.html>, επιβεβαιώνεται ότι η γραμματική που σχεδιάστηκε είναι LL(1).

- All nonterminals are reachable and realizable.
- The nullable nonterminals are: Stmt\_list Term\_tail Factor\_tail.
- The endable nonterminals are: 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
Expr	(Expr) id tf01	id print	no	yes
Term_tail	and or	id print	yes	yes
Term	(Expr) id tf01	and or id print	no	yes
Factor_tail	not	and or id print	yes	yes
Factor	(Expr) id tf01	not and or id print	no	yes
Aoop	and or	(Expr) id tf01	no	no
Nop	not	(Expr) id tf01	no	no

The grammar is LL(1).

- Επιπλέον, είναι διακριτά και τα FIRST / FOLLOW sets της:

- FIRST sets:

- Stmt\_list: id =
    - Stmt: id = print
    - Expr: (Expr) id tf01
    - Term\_tail: and or
    - Term: (Expr) id tf01
    - Factor\_tail: not
    - Factor: (Expr) id tf01
    - Aoop: and or
    - Nop: not

- FOLLOW sets:

- Stmt\_list
    - Stmt: id print
    - Expr: id print
    - Term\_tail: id print
    - Term: and or id print
    - Factor\_tail: and or id print
    - Factor: not and or id print
    - Aoop: (Expr) id tf01
    - Nop: (Expr) id tf01

- Συνοπτική Περιγραφή του κώδικα *parser.py*:

Αρχικά, ο κώδικας ξεκινάει, δημιουργώντας το scanner, και έπειτα, δημιουργούμε το λεξικό, στο οποίο θα βασιστεί το πρόγραμμα. Βάζω τους χαρακτήρες *t, f, 0, 1* και τις λέξεις *true, false*. Επιπλέον, χρησιμοποιούνται και οι τελεστές *not, and, or*. Ξεκινώντας, μέσω της συνάρτησης *parse*, ξεκινάει δημιουργείται ο *scanner* και προχωράει στην συνάρτηση *stmt\_list*. Έπειτα, ψάχνει κάποιο *Identifier*, δηλαδή το όνομα μίας μεταβλητής, που το δίνουμε στο αρχείο που διαβάσει. Εφόσον διαβάσει το *identifier* ή τον χαρακτήρα '=', συνεχίζει είτε στην *stmt\_list* ή προχωράει στην *stmt*. Από την *stmt*, διαβάσει από το *stmt\_list*, το όνομα κάποιας μεταβλητής, = ή προχωράει και διαβάσει την εντολή *print*. Μετά, προχωράει στην συνάρτηση *expr*, στη συνέχεια στην συνάρτηση, *term()*, η οποία συνεχίζει στην συνάρτηση *factor()*, η οποία ανάλογα με το input, διαβάσει είτε ένα expression μέσα από παρενθέσεις, ή κάποιο identifier, η κάποιο από τα (*true, t, 1*), (*false, f, 0*). Το πρόγραμμα συνεχίζει έτσι, χρησιμοποιώντας της συναρτήσεις για να αναγνωρίσει το δοσμένο input. Συνεχίζοντας, το πρόγραμμα μέσω της *factor\_tail()* αν εντοπίσει το *not* τότε συνεχίζει την ανάγνωση. Αν όχι, τότε προχωράει μέσω της *expr()* στο *term\_tail()*, και εκεί βλέπει αν υπάρχει κάποιο από τα *or, and*. Καταλυτικό ρόλο στο πρόγραμμα παίζουν οι συναρτήσεις *match()*, *next\_token()*, οι οποίες μέσω του προγράμματος, όταν αυτό διαβάσει έναν σωστό δοσμένο χαρακτήρα, τον κάνει *match* με κάποιο από τα σύμβολα που έχουμε βάλει από την αρχή στο λεξικό του προγράμματος, και έπειτα φέρνει τον επόμενο προς ανάγνωση χαρακτήρα, μέσω της *next\_token*, η οποία "σκανάρει" και τον διαβάσει.

- Στιγμιότυπα Δοκιμών:

The top window is a terminal titled "Τερματικό". It shows the execution of a Python script "parser.py". The script contains a recursive descent parser for a simple grammar. The error message is: "plex.errors.UnrecognizedInput: '', line 2, char 16: Token not recognised in state ''". The bottom window is a text editor titled "data.txt (~/Desktop/ex2)". It contains the following text: "a = true or false", "b = 0 or (1 and 4)", "c = T and (T and not F)".

```
self.term()
File "parser.py", line 97, in term
    self.factor()
File "parser.py", line 115, in factor
    self.expr()
File "parser.py", line 81, in expr
    self.term_tail()
File "parser.py", line 87, in term_tail
    self.aoop()
File "parser.py", line 130, in aoop
    self.match('and')
File "parser.py", line 51, in match
    self.la, self.val = self.next_token()
File "parser.py", line 56, in next_token
    return self.scanner.read()
File "/usr/local/lib/python3.4/dist-packages/plex/scanners.py", line 94, in read
    self.text, action = self.scan_a_token()
File "/usr/local/lib/python3.4/dist-packages/plex/scanners.py", line 138, in scan_a_token
    raise errors.UnrecognizedInput(self, self.state_name)
plex.errors.UnrecognizedInput: '', line 2, char 16: Token not recognised in state ''
aristotle@reggaeshark ~/Desktop/ex2 $
```

data.txt (~/Desktop/ex2)

```
a = true or false
b = 0 or (1 and 4)
c = T and (T and not F)
```

The top window is a terminal titled "Τερματικό". It shows the execution of a Python script "parser.py". The script contains a recursive descent parser for a simple grammar. The error message is: "plex.errors.UnrecognizedInput: '', line 2, char 16: Token not recognised in state ''". The bottom window is a text editor titled "data.txt (~/Desktop/ex2)". It contains the following text: "p".

```
aristotle@reggaeshark ~/Desktop/ex2 $ python3 parser.py
aristotle@reggaeshark ~/Desktop/ex2 $ python3 parser.py
Match Error
aristotle@reggaeshark ~/Desktop/ex2 $ python3 parser.py
Match Error
aristotle@reggaeshark ~/Desktop/ex2 $
```

data.txt (~/Desktop/ex2)

```
p
```

