



Μεταγλωττιστές 2018 Προγραμματιστική Εργασία #2

Ονοματεπώνυμο : Μαγουνάκη Ουρανία

A.M: Π2015140

Η παρούσα εργασία αφορά την κατασκευή top-down αναλυτή με την μέθοδο της αναδρομικής κατάβασης, ο οποίος αναγνωρίζει εντολές για την ανάθεση σε μεταβλητές και εκτύπωση λογικών εκφράσεων οι οποίες περιέχουν λογικές εκφράσεις, αναγνωριστικά ονόματα μεταβλητών και λογικούς τελεστές.

Η ζητούμενη γραμματική η οποία κατασκευάστηκε αρχικά, ήταν η δημιουργία κανόνων για κάθε τελεστή καθώς έχουν διαφορετική προτεραιότητα. Όμως δεν κατάφερα να δημιουργήσω τον κώδικα ο οποίος αντιστοιχεί στην συγκεκριμένη γραμματική καθώς παρουσιάστηκαν αρκετά προβλήματα στην φάση της υλοποίησης. Συνεπώς, η γραμματική που δημιουργήθηκε είναι η εξής :

```
Stmt_list -> Stmt Stmt_list|.
Stmt -> id assign Expr|print Expr.
Expr -> Term Term_tail.
Term_tail -> Or_And_op Term Term_tail|.
Term -> Factor Factor_tail.
Factor_tail -> Not_op Factor Factor_tail|.
Factor -> (Expr) |Boolconst |id|.
Or_And_op -> or|not.
Not_op -> not.
Boolconst -> true|false|t|f|0|1.
```

Όσον αφορά τον τελευταίο κανόνα (**Boolconst**) δεν δημιουργήθηκε αντίστοιχη συνάρτηση, τα **True** , **False** ορίζονται ως **tokens**. Τονίζεται ότι χρησιμοποιείται **plex.NoCase** προκειμένου να ληφθούν υπ' όψη κεφαλαία και μικρά.

Παρακάτω παρουσιάζεται σχετική εικόνα με τα αποτελέσματα ελέγχου για LL(1) συμβατότητα καθώς και ο πίνακας που περιέχει τα FIRST και FOLLOW sets.

```

Grammar
Stmt_list → Stmt Stmt_list
          | .
Stmt →    id assign Expr
          | print Expr.
Expr →    Term Term_tail.
Term_tail → Or_And_op Term Term_tail
           | .
Term →     Factor Factor_tail.
Factor_tail → Not_op Factor Factor_tail
            | .
Factor →   (Expr)
          | Boolconst
          | id
          | .
Or_And_op → or
          | not.
Not_op →   not.
Boolconst → true
          | false
          | t
          | f
          | 0
          | 1.

```

Some sentences generated by this grammar: (t, print, print f, print t, print 0, print 1, print id, id assign, print true, id assign 0, id assign f, id assign 1, id assign t, print false, print (Expr), id assign id, id assign true, id assign false, id assign (Expr), id assign (Expr) not (Expr))

- All nonterminals are reachable and realizable.
- The nullable nonterminals are: Stmt_list Term_tail Factor_tail Factor Term Expr.
- The endable nonterminals are: Boolconst Not_op Factor_tail Factor Or_And_op 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 or true false t f 0 1 not	id print	yes	yes
Term_tail	or not	id print	yes	yes
Term	(Expr) id true false t f 0 1 not	or not id print	yes	yes
Factor_tail	not	or not id print	yes	yes
Factor	(Expr) id true false t f 0 1	or not id print	yes	yes
AndOr_op	or not	(Expr) true false t f 0 1 or not id print	no	yes
Not_op	not	(Expr) true false t f 0 1 or not id print	no	yes
Boolconst	true false t f 0 1	or not id print	no	yes

Συνοπτική περιγραφή κώδικα

Επισημαίνεται ότι ο κώδικας στον οποίο βασίστηκα είναι από αντίστοιχο παράδειγμα που υλοποιήθηκε κατά την διάρκεια του εργαστηρίου.

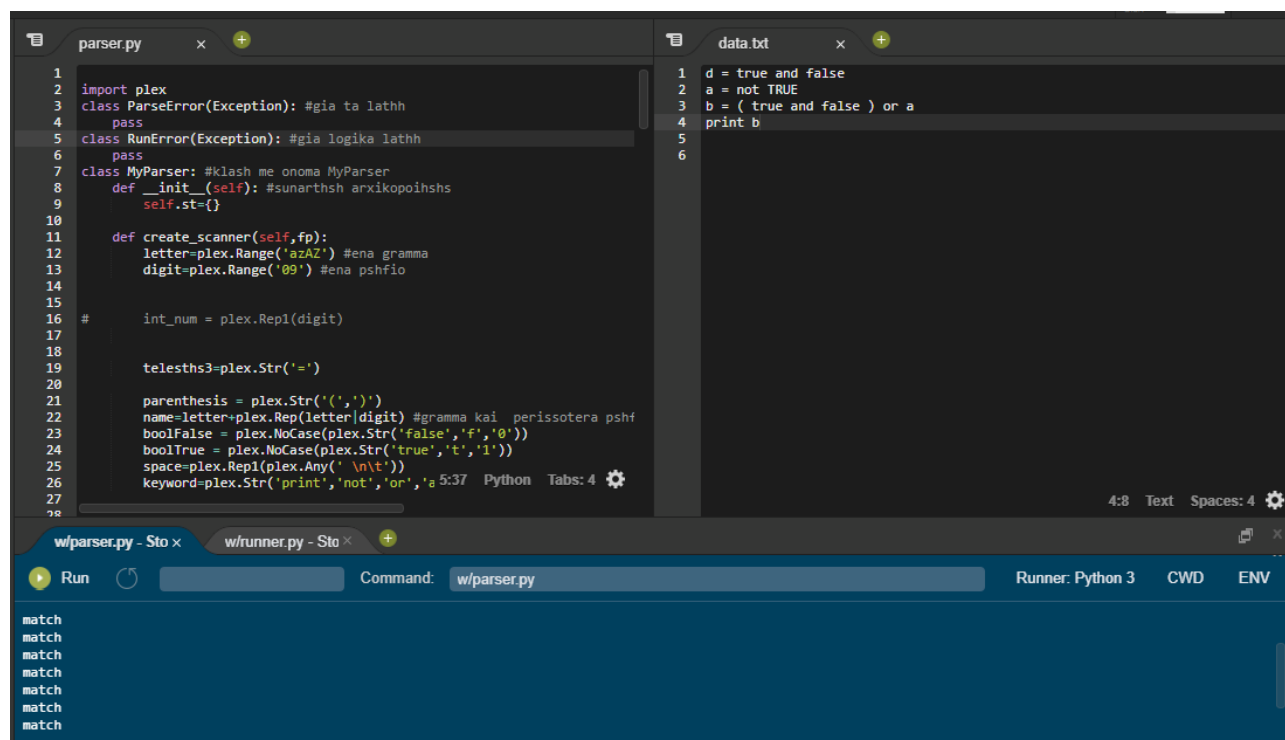
Όσον αφορά τον parser χρησιμοποιεί το **module plex**. Αναλυτικότερα, αρχικά έχει δημιουργηθεί η κλάση **ParseError** η οποία εκτυπώνει μηνύματα λάθους. Η κλάση **MyParser** περιέχει μία συνάρτηση αρχικοποίησης **init**, το **create_scanner** εντός του οποίου ορίζονται τα γράμματα (**letters**), τα ψηφία (**digits**), ο τελεστής ισότητας, οι παρενθέσεις, τα ονόματα μεταβλητών, οι λογικές τιμές (**boolFalse**, **boolTrue**), τα κενά (**spaces**), τα απαραίτητα **keywords** (**print**, **not**, **and**, **or**) και η αφαίρεση των σχολίων.

Έπειτα, ορίζεται το λεξικό με τα απαραίτητα **tokens** ενώ το **scanner** προσπελαύνει το αρχείο και δημιουργούμε τις μεταβλητές **self.la** και **self.val**. Η συνάρτηση **match()** ελέγχει τα **tokens** εάν υπάρχουν, ώστε να γίνει η αντιστοίχιση και στην συνέχεια ορίζονται οι συναρτήσεις που αντιστοιχούν στους κανόνες της γραμματικής βάση των **first** και **follow**.

Εν κατακλείδι, δημιουργείται ένα αντικείμενο της κλάσης **MyParser** προκειμένου να γίνει η προσπέλαση του αρχείου.

Ωστόσο, επισημαίνεται ότι κατά την εκτέλεση του **parser** παρατηρήθηκε ότι δεν αντιστοιχεί όλα τα **tokens**.

Ορισμένα αποτελέσματα έπειτα από την εισαγωγή διάφορων συνδυασμών εισόδου:



```
1 import plex
2 class ParseError(Exception): #για τα lathh
3     pass
4 class RunError(Exception): #για logika lathh
5     pass
6 class MyParser: #klash me onoma MyParser
7     def __init__(self): #sunarthsh arxikopoihs
8         self.st={}
9
10    def create_scanner(self,fp):
11        letter=plex.Range('a-zAz') #ena gramma
12        digit=plex.Range('09') #ena pshfio
13
14        #
15        int_num = plex.Repl(digit)
16
17        #
18        telesths3=plex.Str('=')
19
20        #
21        parenthesis = plex.Str('(','')
22        name=letter+plex.Rep(letter|digit) #gramma kai perissotera pshf
23        boolFalse = plex.NoCase(plex.Str('false','f','0'))
24        boolTrue = plex.NoCase(plex.Str('true','t','1'))
25        space=plex.Repl(plex.Any(' \n\t'))
26        keyword=plex.Str('print','not','or','e 5:37 Python Tabs: 4
27
28
1 d = true and false
2 a = not TRUE
3 b = ( true and false ) or a
4 print b
5
6
```

```
w/parser.py - Sto x w/runner.py - Sto x
Run Command: w/parser.py Runner: Python 3 CWD ENV
match
match
match
match
match
match
match
```

Ορισμένα αποτελέσματα έπειτα από την εισαγωγή διάφορων συνδυασμών εισόδου:

The image shows a code editor with two tabs: 'runner.py' and 'data.txt'.

runner.py (lines 139-166):

```

139         return op, not c
140     elif self.la == 'and' or self.la == 'or' or self.la == 'print': c
141         return
142     else:
143         raise ParseError('error ,not')
144
145
146     def factor(self):
147         if self.la == '(':
148             self.match('(')
149             e = self.expr()
150             self.match(')')
151             return e
152
153         elif self.la == 'IDENTIFIER':
154             varname = self.val
155             self.match('IDENTIFIER')
156             if varname in self.st:
157                 return self.st[varname]
158
159         elif self.la == 'true':
160
161             self.match('true')
162             return True
163
164         elif self.la == 'false':
165
166

```

data.txt (lines 1-8):

```

1 a = true
2 print a
3 b = 1
4 c = 0
5 e = b or c
6 print e
7
8

```

The bottom status bar shows the command 'w/runner.py' and the runner 'Python 3'. The output area shows 'True' and 'True'.

The image shows a code editor with two files open. The left file, 'runner.py', contains a Python class with methods for parsing expressions. The right file, 'data.txt', contains a list of variables and their values. The bottom status bar shows the command 'w/runner.py' and the output 'False error ,not'.

```

runner.py
137
140         elif self.la == 'and' or self.la == 'or' or self.la == 'print':
141             return
142         else:
143             raise ParseError('error ,not')
144
145     def factor(self):
146         if self.la == '(':
147             self.match('(')
148             e = self.expr()
149             self.match(')')
150             return e
151
152         elif self.la == 'IDENTIFIER':
153             varname = self.val
154             self.match('IDENTIFIER')
155             if varname in self.st:
156                 return self.st[varname]
157
158         elif self.la == 'true':
159             self.match('true')
160             return True
161
162         elif self.la == 'false':
163             self.match('false')
164             return False
165
166         else:
167             raise ParseError('error ,not')
168
169     def parse(self):
170         self.match('start')
171         e = self.expr()
172         self.match('end')
173         return e
174
175     def __str__(self):
176         return str(self.st)
177
178     def __repr__(self):
179         return repr(self.st)
180
181     def __len__(self):
182         return len(self.st)
183
184     def __getitem__(self, i):
185         return self.st[i]
186
187     def __setitem__(self, i, v):
188         self.st[i] = v
189
190     def __delitem__(self, i):
191         del self.st[i]
192
193     def __iter__(self):
194         return iter(self.st)
195
196     def __contains__(self, v):
197         return v in self.st
198
199     def __eq__(self, other):
200         return self.st == other.st
201
202     def __neq__(self, other):
203         return self.st != other.st
204
205     def __lt__(self, other):
206         return self.st < other.st
207
208     def __gt__(self, other):
209         return self.st > other.st
210
211     def __le__(self, other):
212         return self.st <= other.st
213
214     def __ge__(self, other):
215         return self.st >= other.st
216
217     def __hash__(self):
218         return hash(self.st)
219
220     def __call__(self):
221         return self
222
223     def __getattr__(self, attr):
224         return getattr(self.st, attr)
225
226     def __setattr__(self, attr, value):
227         setattr(self.st, attr, value)
228
229     def __delattr__(self, attr):
230         delattr(self.st, attr)
231
232     def __copy__(self):
233         return self
234
235     def __deepcopy__(self, memo):
236         return self
237
238     def __reduce__(self):
239         return self
240
241     def __sizeof__(self):
242         return len(self.st)
243
244     def __str__(self):
245         return str(self.st)
246
247     def __repr__(self):
248         return repr(self.st)
249
250     def __len__(self):
251         return len(self.st)
252
253     def __getitem__(self, i):
254         return self.st[i]
255
256     def __setitem__(self, i, v):
257         self.st[i] = v
258
259     def __delitem__(self, i):
260         del self.st[i]
261
262     def __iter__(self):
263         return iter(self.st)
264
265     def __contains__(self, v):
266         return v in self.st
267
268     def __eq__(self, other):
269         return self.st == other.st
270
271     def __neq__(self, other):
272         return self.st != other.st
273
274     def __lt__(self, other):
275         return self.st < other.st
276
277     def __gt__(self, other):
278         return self.st > other.st
279
280     def __le__(self, other):
281         return self.st <= other.st
282
283     def __ge__(self, other):
284         return self.st >= other.st
285
286     def __hash__(self):
287         return hash(self.st)
288
289     def __call__(self):
290         return self
291
292     def __getattr__(self, attr):
293         return getattr(self.st, attr)
294
295     def __setattr__(self, attr, value):
296         setattr(self.st, attr, value)
297
298     def __delattr__(self, attr):
299         delattr(self.st, attr)
300
301     def __copy__(self):
302         return self
303
304     def __deepcopy__(self, memo):
305         return self
306
307     def __reduce__(self):
308         return self
309
310     def __sizeof__(self):
311         return len(self.st)
312
313     def __str__(self):
314         return str(self.st)
315
316     def __repr__(self):
317         return repr(self.st)
318
319     def __len__(self):
320         return len(self.st)
321
322     def __getitem__(self, i):
323         return self.st[i]
324
325     def __setitem__(self, i, v):
326         self.st[i] = v
327
328     def __delitem__(self, i):
329         del self.st[i]
330
331     def __iter__(self):
332         return iter(self.st)
333
334     def __contains__(self, v):
335         return v in self.st
336
337     def __eq__(self, other):
338         return self.st == other.st
339
340     def __neq__(self, other):
341         return self.st != other.st
342
343     def __lt__(self, other):
344         return self.st < other.st
345
346     def __gt__(self, other):
347         return self.st > other.st
348
349     def __le__(self, other):
350         return self.st <= other.st
351
352     def __ge__(self, other):
353         return self.st >= other.st
354
355     def __hash__(self):
356         return hash(self.st)
357
358     def __call__(self):
359         return self
360
361     def __getattr__(self, attr):
362         return getattr(self.st, attr)
363
364     def __setattr__(self, attr, value):
365         setattr(self.st, attr, value)
366
367     def __delattr__(self, attr):
368         delattr(self.st, attr)
369
370     def __copy__(self):
371         return self
372
373     def __deepcopy__(self, memo):
374         return self
375
376     def __reduce__(self):
377         return self
378
379     def __sizeof__(self):
380         return len(self.st)
381
382     def __str__(self):
383         return str(self.st)
384
385     def __repr__(self):
386         return repr(self.st)
387
388     def __len__(self):
389         return len(self.st)
390
391     def __getitem__(self, i):
392         return self.st[i]
393
394     def __setitem__(self, i, v):
395         self.st[i] = v
396
397     def __delitem__(self, i):
398         del self.st[i]
399
400     def __iter__(self):
401         return iter(self.st)
402
403     def __contains__(self, v):
404         return v in self.st
405
406     def __eq__(self, other):
407         return self.st == other.st
408
409     def __neq__(self, other):
410         return self.st != other.st
411
412     def __lt__(self, other):
413         return self.st < other.st
414
415     def __gt__(self, other):
416         return self.st > other.st
417
418     def __le__(self, other):
419         return self.st <= other.st
420
421     def __ge__(self, other):
422         return self.st >= other.st
423
424     def __hash__(self):
425         return hash(self.st)
426
427     def __call__(self):
428         return self
429
430     def __getattr__(self, attr):
431         return getattr(self.st, attr)
432
433     def __setattr__(self, attr, value):
434         setattr(self.st, attr, value)
435
436     def __delattr__(self, attr):
437         delattr(self.st, attr)
438
439     def __copy__(self):
440         return self
441
442     def __deepcopy__(self, memo):
443         return self
444
445     def __reduce__(self):
446         return self
447
448     def __sizeof__(self):
449         return len(self.st)
450
451     def __str__(self):
452         return str(self.st)
453
454     def __repr__(self):
455         return repr(self.st)
456
457     def __len__(self):
458         return len(self.st)
459
460     def __getitem__(self, i):
461         return self.st[i]
462
463     def __setitem__(self, i, v):
464         self.st[i] = v
465
466     def __delitem__(self, i):
467         del self.st[i]
468
469     def __iter__(self):
470         return iter(self.st)
471
472     def __contains__(self, v):
473         return v in self.st
474
475     def __eq__(self, other):
476         return self.st == other.st
477
478     def __neq__(self, other):
479         return self.st != other.st
480
481     def __lt__(self, other):
482         return self.st < other.st
483
484     def __gt__(self, other):
485         return self.st > other.st
486
487     def __le__(self, other):
488         return self.st <= other.st
489
490     def __ge__(self, other):
491         return self.st >= other.st
492
493     def __hash__(self):
494         return hash(self.st)
495
496     def __call__(self):
497         return self
498
499     def __getattr__(self, attr):
500         return getattr(self.st, attr)
501
502     def __setattr__(self, attr, value):
503         setattr(self.st, attr, value)
504
505     def __delattr__(self, attr):
506         delattr(self.st, attr)
507
508     def __copy__(self):
509         return self
510
511     def __deepcopy__(self, memo):
512         return self
513
514     def __reduce__(self):
515         return self
516
517     def __sizeof__(self):
518         return len(self.st)
519
520     def __str__(self):
521         return str(self.st)
522
523     def __repr__(self):
524         return repr(self.st)
525
526     def __len__(self):
527         return len(self.st)
528
529     def __getitem__(self, i):
530         return self.st[i]
531
532     def __setitem__(self, i, v):
533         self.st[i] = v
534
535     def __delitem__(self, i):
536         del self.st[i]
537
538     def __iter__(self):
539         return iter(self.st)
540
541     def __contains__(self, v):
542         return v in self.st
543
544     def __eq__(self, other):
545         return self.st == other.st
546
547     def __neq__(self, other):
548         return self.st != other.st
549
550     def __lt__(self, other):
551         return self.st < other.st
552
553     def __gt__(self, other):
554         return self.st > other.st
555
556     def __le__(self, other):
557         return self.st <= other.st
558
559     def __ge__(self, other):
560         return self.st >= other.st
561
562     def __hash__(self):
563         return hash(self.st)
564
565     def __call__(self):
566         return self
567
568     def __getattr__(self, attr):
569         return getattr(self.st, attr)
57
```