ΌΝΟΜΑ: Αικατερίνη

ΕΠΩΝΥΜΟ: Μηλιώτη

ΑΜ: 1115201900112

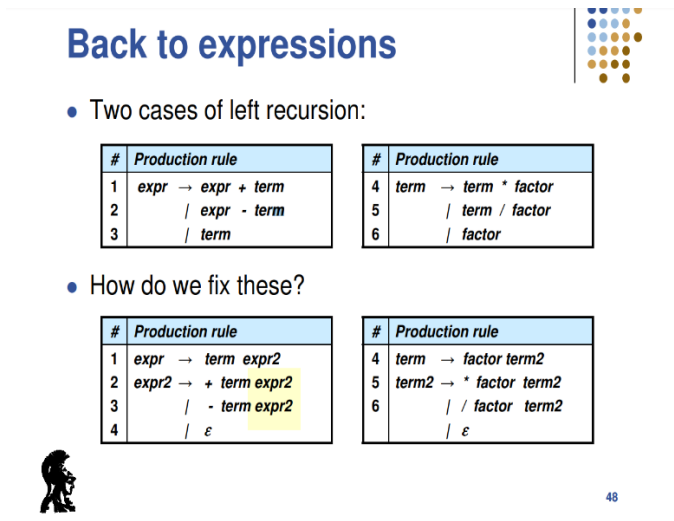
ΕΠΕΞΗΓΗΣΗ PART 1

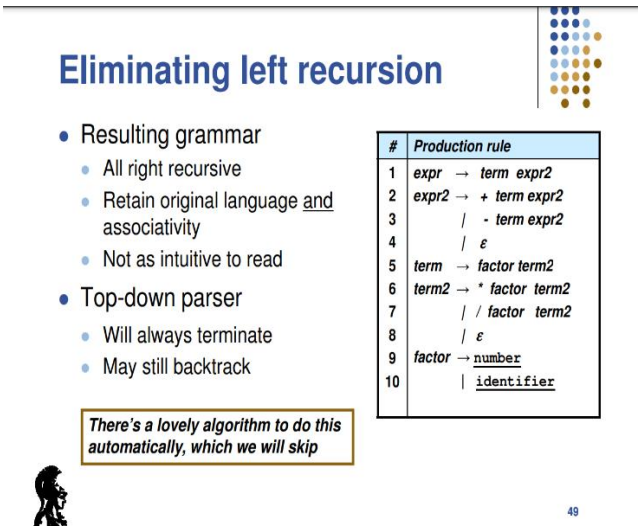
- Πως έφτιαξα τη γραμματική μου:

1. Αρχικά, αφαίρεσα την αριστερή αναδρομή(left recursion) χρησιμοποιώντας τον εξής κανόνα : όταν A -> Aα | β έχουμε αριστερή αναδρομή και αφαιρώντας την αριστερή αναδρομή θα γίνει:
   A-> βA1
   A1-> αA1 | ε
   Παρατήρησα ότι η γραμματική της εκφώνησης έχει αριστερή αναδρομή στο σημείο exp-> num | exp op exp | (exp) και με βάση τον παραπάνω κανόνα έγινε
   exp-> num exp2 | (exp) exp2
   exp2-> op exp exp2 | ε
   Το op και το num παρέμειναν προς το παρόν έτσι όπως ήταν.

2. Έπειτα παρατήρησα ότι η γραμματική δεν δίνει προτεραιότητα στους τελεστές(ο τελεστής & έχει μεγαλύτερη προτεραιότητα από τον τελεστή ^). Οπότε, έφτιαξα την γραμματική μου : exp -> term exp2 | (exp) exp2
   exp2-> ^ term exp2 | ε
   term-> factor term2
   term2-> & factor term2 | ε
   factor -> num
   num-> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

   με βάση αυτές τις διαφάνειες του μαθήματος





3. Στη συνέχεια, άρχισα να φτιάχνω τον parser , αλλά παρατήρησα ότι κάτι δεν πάει καλά με τις παρενθέσεις όποτε σκέφτηκα να αλλάξω την παραπάνω γραμματική και να γίνει :

```
1   exp -> term exp2
2   exp2-> ^ term exp2
3          | ε
4   term-> factor term2
5   term2-> & factor term2
6             | ε
7   factor -> num
8             | (exp)
9-18 num-> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

καθώς το factor μπορεί να είναι είτε μονοψήφιος αριθμός, είτε εκφράσεις μέσα σε παρενθέσεις.

- Έλεγχος αν η γραμματική που έφτιαξα είναι LL(1):

FIRST

- First(#1) = {0,1,2,3,4,5,6,7,8,9,(}
- First(#2) = {^}
- First(#3) = {ε}
- First(#4) = {0,1,2,3,4,5,6,7,8,9,(}
- First(#5) = {&}
- First(#6) = {ε}
- First(#7) = {0,1,2,3,4,5,6,7,8,9}
- First(#8) = {(}
- Για συντομία First(num) = {0,1,2,3,4,5,6,7,8,9}

FOLLOW

- Follow(exp) = {$,)}
- Follow(exp2) = {$,)}
- Follow(term) = {^,$,)}
- Follow(term2) = {^,$,)}
- Follow(factor) = {&,^,$,)}
- Follow(num) = {&,^,$,)}

FIRST +

- First +(exp -> term exp2) = {0,1,2,3,4,5,6,7,8,9,(}
- First +(exp2-> ^ term exp2) = {^}
- First+(exp2->ε) = {$,)}

- First+(term-> factor term2) = {0,1,2,3,4,5,6,7,8,9,(}
- First+(term2-> & factor term2) = {&}
- First+(term2-> ε) = {^,$,)}
- First+(factor -> num) = {0,1,2,3,4,5,6,7,8,9}
- First+(factor->(exp)) = {(}

Για συντομία τα First+(num->....) δεν θα γραφθούν, καθώς είναι προφανή.

Παρατηρούμε ότι για κάθε A->α και A->β First(a) ∩ First(β) = ∅, οπότε η γραμματική είναι LL(1).

➢ <u>Δημιουργία LL(1) Parsing Table:</u>

	(	)	^	&	0…9	$
exp	#1	error	error	error	#1	error
exp2	error	exp2->ε	#2	error	error	exp2-> ε
term	# 4	error	error	error	#4	error
term2	error	term2->ε	term2->ε	#5	error	term2->ε
factor	#8	error	error	error	#7	error
num	error	error	error	error	#(9-18)	error

➢ <u>Εξηγήσεις στον κώδικα</u>

Για να φτιάξω τον parser χρησιμοποίησα το παραπάνω LL(1) Parsing Table, το οποίο με βοήθησε πάρα πολύ. Οπότε, αυτό που κάνω και στον κώδικα είναι ότι ακριβώς λέει και το παραπάνω πινακάκι. Αν για παράδειγμα είμαι στη συνάρτηση exp() όταν δω το σύμβολο (πρέπει να γίνει η πρώτη παραγωγή οπότε το μεταφράζω αυτό σε κώδικα Java. Με αυτή τη λογική έγραψα τον parser. Επίσης, όταν η συνάρτηση exp2() δει το σύμβολο ^ , πρέπει να διαβάσει τον επόμενο χαρακτήρα, για αυτό έχουμε consume('^'). Το ίδιο ισχύει αντίστοιχα και για το σύμβολο & στη συνάρτηση term2(). Τέλος, στη συνάρτηση factor() , μόλις δει το σύμβολο (το κάνει consume ,έπειτα καλεί τη συνάρτηση exp() και μετά αν δεν εμφανισθεί κανένα parse error πρέπει να κάνει consume το σύμβολο) , σύμφωνα με την παραγωγή #8.

- Για μεταγλώττιση make compile.
- Για να τρέξει make execute.
- Οι εκφράσεις πρέπει να δίνονται από τον χρήστη.

<u>ΣΗΜΕΙΩΣΗ:</u> <u>Έχω χρησιμοποιήσει τον κώδικα του φροντιστηρίου που δόθηκε για αντίστοιχο παράδειγμα.</u>

ΕΠΕΞΗΓΗΣΗ PART 2

Αρχικά, οι προτεραιότητες που έχω είναι precedence left IF;

precedence left PLUS;

precedence left REVERSE;

πράγμα που δείχνει ότι precedence(if)<precedence(concat)<precedence(reverse)

Έχω την παραγωγή programme :: = ….. , η οποία δέχεται όλους τους πιθανούς συνδυασμούς που μπορεί να δοθεί ένα input.Για παράδειγμα, να δοθεί ένα άδειο αρχείο, ένα αρχείο που έχει πρώτα εκφράσεις μετά δήλωση συναρτήσεων, κλπ. Όμως , δεν μπορεί να δεχθεί περιπτώσεις όπου για παράδειγμα έχουμε μια κλήση συνάρτησης, μετά ένα string, μετά πάλι μια κλήση συνάρτησης(ανακατεμένα) κλπ. , καθώς δεν έχει αναδρομή. Επιπλέον, έχω δύο παραγωγές μια για το τι μπορεί να έχει η main (main ::=…) και μια για το τι μπορεί να έχει ο κώδικας μιας συνάρτησης (functionCode ::=…) , γιατί όταν τα είχα μαζί αρχικά μου έβγαζε conflicts και δεύτερον στη main δεν μπορώ να έχω μεταβλητές(identifiers). Επίσης , για τον λόγο το ότι μου έβγαζε conflicts όταν τα είχα μαζί έχω δύο παραγωγές μια για την κλήση συναρτήσεων στη main (call :: = …)και μια για την κλήση συναρτήσεων μέσα στο σώμα των συναρτήσεων (functionCallufunction ::=….).Επιπλέον, στις παραγωγές με το if έχω τις παρενθέσεις με κόκκινο χρώμα {:RESULT= "(" + "(" + e + ")?" + f1 + " : " + f2 + ")";:} για να τηρείται η προτεραιότητα για εκφράσεις όπως, "yes" + if ("y" prefix "y") " correct" else " incorrect" + " input" . Επιπροσθέτως, τα if else εμφανίζονται με τη μορφή (a>b) ? a:b

ΣΗΜΕΙΩΣΗ: Έχω χρησιμοποιήσει τον κώδικα του φροντιστηρίου που δόθηκε για αντίστοιχο παράδειγμα.

- Για μεταγλώττιση make compile.
- Για να τρέξει make execute < όνομα αρχείου.txt