**Γενικά**

* Για την άμεση παραγωγή των έτοιμων αρχείων του sableCC (framework) καθώς και για τη μεταγλώττιση των .java έχει υλοποιηθεί ένα script.  
  *> ./myscript.sh*
* Έχει υλοποιηθεί Makefile.
* Ενδεικτική εκτέλεση: *> java Main ./examples/hello.grace*
* Έχουμε συμπεριλάβει στο παραδοτέο το sableCC.jar για την επιτυχή εκτέλεση του script και συνεπώς για τη διευκόλυνση σας.
* Επειδή διαλέξαμε την υλοποίηση ενός symbol table (hashtable & stack) του οποίου οι “εμβέλειες’ καταστρέφονται την ώρα που βγαίνουμε, κατα το traversal, από αυτές, χωρίς να κρατώνται οι πληροφορίες των εμβελειών αυτών σε κάποια άλλη δομή, η χρήση παραπάνω visitor από έναν δεν ήταν εφικτή.  
    
  **Χρησιμοποιείται ένας και μοναδικός visitor για όλες τις φάσεις της μεταγλώττισης.**  
    
  Αυτό κανει τον κώδικά μας λιγάκι πιο δυσανάγνωστο, ωστόσο, έχει άλλα πλεονεκτήματα, όπως το ότι η υλοποίηση αυτή είναι πιο γρήγορη/αποδοτική.

Η αδυναμία επιστροφής πληροφορίας από έναν κόμβο στο γονιό του μέσω του return value κάθε μεθόδου, εξαιτίας της δημιουργίας void μεθόδων από το sableCC, καθιστά απαραίτητη τη χρήση επιπρόσθετων members  
στον visitor, κάτι που επίσης καθιστά τον κώδικά μας λιγότερο ευανάγνωστο.

* Μεταξύ των δύο τρόπων υλοποίησης του μηχανισμού επικοινωνίας των παιδιών κόμβων με τους γονείς τους που συζητήθηκαν στο piazza, επιλέξαμε αυτόν κατά τον οποίο κάθε κόμβος αφήνει πληροφορία για τον γονέα του σε members του visitor.
* Για την επικοινωνία κόμβων - παιδιών με τους γονείς τους για την κατασκευή του symbol table χρησιμοποιήθηκαν τα members:  
    
   Boolean reference;  
   String datatype;  
   Key name;  
   Key funname;  
   LinkedList <Param> params;  
   LinkedList <Integer> arraylist;  
   LinkedList <Key> idlist;  
   String retvalue;
* Για την επικοινωνία κόμβων - παιδιών με τους γονείς τους για το type checking καθώς και για τους άλλους ελέγχους της σημασιολογικής ανάλυσης χρησιμοποιείται μια δομή stack.   
   LinkedList <TypeCheck> typeCheck;  
    
  Κάθε κόμβος παίρνει την πληροφορία που χρειάζεται από τα παιδιά του κάνοντας pop ένα ή περισσότερα elements από τη στοίβα και στο τέλος κάνει push τη δική του πληροφορία σε αυτήν, έτσι ώστε να την εκμεταλλευτεί αργότερα ο δικός του γονέας.  
    
  Τα elements της λίστας περιέχουν τόσο τον τύπο της έκφρασης/τιμής που “επιστρέφεται” όσο και άλλες διάφορες χρήσιμες πληροφορίες για τη σημασιολογική ανάλυση.  
    
  Γίνεται χρήση άλλης μιας δομής stack:  
   LinkedList <Node> funDefinition;  
  στην οποία αποθηκεύονται μόνο οι συναρτήσεις που έχουν ορισθεί με τη σειρά που ορίζονται. Αυτές αποθηκεύονται και στο symbol table. Η χρήσιμότητα της funcDefinition είναι για την ανίχνευση σημασιολογικών λαθών που αφορούν στην τιμή επιστροφής μιας συνάρτησης.  
  Κάθε φορά που φτάνουμε σε κάποιο κόμβο “return”, δηλαδή κάθε φορά που γίνεται return; / return expr; στο σώμα μιας συνάρτησης, λαμβάνεται (pop χωρίς remove) το τελευταίο element της στοιβας που αντιστοιχεί στη συνάρτηση, στης οποίας το σώμα βρισκόμαστε τώρα και γίνονται οι απαραίτητοι έλεγχοι/συγκρίσεις.  
  Ένα element αυτής της στοίβας γίνεται pop (με remove) κατά την έξοδο από τον ορισμό της συνάρτησης, δηλαδή στη   
   **void** **outAFuncdefLocalDef**(AFuncdefLocalDef node)

**Τμήματα Project – Πορεία Υλοποίησης**

* Λεκτική Ανάλυση  
   - Regular Expressions
* Συντακτική Ανάλυση  
   - Γραμματική (Productions section)  
   - Αφαίρεση ambiguities
* Μετατροπή CST -> AST
* Υλοποίηση Symbol Table
* Σημασιολογική Ανάλυση  
   - Type Checking  
   - Έλεγχοι Μοναδικότητας  
   . . . . . .  
   *Έχει συμπεριληφθεί αρχείο .txt μαζί με το παραδοτέο όπου αναφέρονται όλα τα printed errors που έχουν ληφθεί υπόψη μας κατα τη σημασιολογική ανάλυση.*
* Ενδιάμεσος Κώδικας  
   ***not implemented yet***

**Λεκτική Ανάλυση**

**Έχουν ληφθεί υπόψη μας όλες οι αναφερθείσες παρατηρήσεις από την εξέταση του πρώτου μέρους της άσκησης – λεκτική/συντακτική ανάλυση και έχουν διορθωθεί.**

Πιο συγκεκριμένα, πλέον:

* Δεν γίνονται δεκτά strings που είναι multiline.
* Δεν γίνονται δεκτά unmatched multiline comments  
  *Διευκρύνιση: Το πρόβλημα ήταν ότι γινόταν δεκτό το single line comment που αποτελείται από ένα μόνο $,  
  Δηλαδη:* ***$***
* Δεν γίνονται δεκτά inputs τύπου: 5mod 12.  
  *Λύση: Δημιουργήσαμε άλλο ένα regular expression που αναγνωρίζει  
  tokens τύπου* ***digit+ letter+****τα οποία απορρίπτονται στη συνέχεια από τον συντακτικό αναλυτή.*
* Δεν γίνονται δεκτά strings που περιέχουν backslash χωρίς να πρόκειται για escape character, πχ: “**\*Foo” , “*\*xLL“***

**Μετατροπή CST -> AST**

Υλοποιήθηκε κανονικά το Abstract Syntax Tree section.  
Έχουν αποκοπεί κόμβοι απο το cst που ήταν περιττοί για τις επόμενες φάσεις της μεταγλώττισης.  
Ωστόσο,κάποιοι άλλοι, φαινομενικά περιττοί, κρατήθηκαν ή προστέθηκαν για τη δημιουργία ενός δέντρου που θα ευνοεί τις φάσεις της μεταγλώττισης όπως τις υλοποιήσαμε προγραμματιστικά εμείς.

**Υλοποίηση Symbol Table**

Για την υλοποίηση του symbol table χρησιμοποιείται μια δομή hashtable καθώς και μια δομή stack. Η συνολική δομή είναι όμοια με αυτή που προτάθηκε στο piazza, καθώς και με αυτή που υπάρχει στις διαφάνειες της θεωρίας του μαθήματος.  
Έχουν υλοποιηθεί οι μέθοδοι:  
 **- enter()  
 - insert(<args>)  
 - lookup()  
 - exit()**

Για κάθε εισαγωγή στο symbol table μιας μεταβλητής/συνάρτησης κρατώνται οι παρακάτω πληροφορίες:  
 Key name; //Hashkey String type; **int** scope;  
 Boolean reference;  
 LinkedList <Param> params; LinkedList <Integer> arraylist;  
 String retvalue; //Type Of Returned Value Boolean defined; //Function Defined or Declared?  
  
*Some of the above fields concern only the function variables, while others concern only the simple variables. In case a field doesn’t concern a variable, it gets null as a value.*

**Συντακτική Ανάλυση**

* Έχει διορθωθεί ένα λάθος στη γραμματική που αφορούσε την προτεραιότητα του “not” σε σχέση με τα “or” , “and” .
* Τροποποιήθηκαν ελάχιστα τα productions l\_values και block της γραμματικής για λόγους που αφορούν στην παραγωγή ενός συντακτικού/αφηρημένου δέντρου που ευνοεί/εξυπηρετεί περισσότερο την κατασκευή του symbol table και τη σημασιολογική ανάλυση.