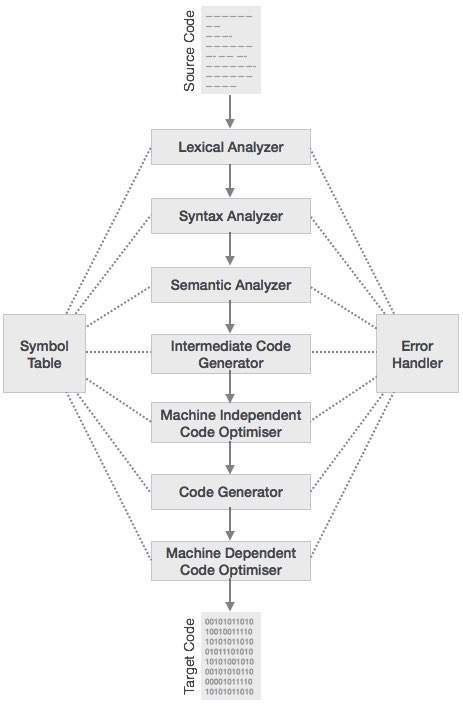
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΥ802 - ΜΕΤΑΦΡΑΣΤΕΣ

ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ



**ΜΕΛΗ ΟΜΑΔΑΣ**

ΚΩΝΣΤΑΝΤΙΝΟΣ ΠΑΠΑΝΤΩΝΙΟΥ-ΧΑΤΖΗΓΚΙΟΣΗΣ - 4769

ΜΙΧΟΥ ΝΑΤΑΛΙΑ - 4922

ΙΩΑΝΝΙΝΑ - ΜΑΙΟΣ 2024

# ΕΙΣΑΓΩΓΗ

Η παρούσα εργασία έχει ως στόχο την δημιουργία ενός πληρους μεταφραστή σε RISC-V assembly για την εκπαιδευτική γλώσσα cpy. Ο μεταφραστής, γραμμένος σε γλώσσα Python, πρέπει τελικά να μπορεί να πάρει ως είσοδο ένα πρόγραμμα σε cpy και να δημιουργήσει τελικά ένα αρχείο με κώδικα σε RISC-V assembly το οποίο μπορεί να τρέξει κανονικά σε προσομοιωτή και να δώσει τα σωστά αποτελέσματα.

Η σχεδίαση και υλοποίηση του μεταφραστή έγινε σε στάδια όπως μας ζητήθηκε. Συγκεκριμένα τα στάδια αυτά είναι:

1. Σχεδίαση και υλοποίηση Λεκτικού Αναλυτή (Lexer)
2. Σχεδίαση και υλοποίηση Συντακτικού Αναλυτή (Parser)
3. Παραγωγή Ενδιάμεσου Κώδικα (Intermediate Code)
4. Παραγωγή Πίνακα Συμβόλων (Symbol Table)
5. Παραγωγή Τελικού Κώδικα (Final Code)

Στα πλαίσια της εκπαιδευτικής αυτής εργασίας δεν μας ζητήθηκε βελτιστοποίηση του παραγόμενου κώδικα σε κανένα από τα δύο στάδια οπού αυτό θα ήταν δυνατό (στάδια 3 και 5)

# 1. ΛΕΚΤΙΚΟΣ ΑΝΑΛΥΤΗΣ

## Γενική Ιδέα

Στο κεφάλαιο αυτό παρουσιάζουμε τον Λεκτικό Αναλυτή που δημιουργήσαμε. Ο σκοπός του είναι να αναλύει τη γραμματική ενός αρχείου πηγαίου κώδικα και να παράγει μια σειρά από tokens, που αντιπροσωπεύουν τα αναγνωρισμένα σύμβολα της γλώσσας προγραμματισμού cpy. Αυτός ο λεκτικός αναλυτής είναι το πρώτο βήμα στη διαδικασία της μετάφρασης από υψηλού επιπέδου κώδικα σε εντολές που μπορεί να εκτελέσει ένας υπολογιστής.

Ο λεκτικός αναλυτής που έχουμε δημιουργήσει λειτουργεί μέσω ενός finite state machine (FSM), και επεξεργάζεται τον πηγαίο κώδικα χαρακτήρα προς χαρακτήρα. Ανάλογα με τον χαρακτήρα που διαβάζεται και την τρέχουσα κατάσταση του finite state machine, ο αναλυτής καθορίζει τον τύπο του token, καθώς και τη μετάβαση σε μια νέα κατάσταση. Τα αναγνωρισμένα tokens αποθηκεύονται σε μια λίστα και καταγράφονται στο αρχείο εξόδου. Αν βρεθεί κάποιο λάθος, καταγράφεται στο αρχείο εξόδου και ο αναλυτής συνεχίζει.

Αυτή η διαδικασία γίνεται μέσω ενός βρόχου που διατρέχει κάθε χαρακτήρα του αρχείου.

Ο σχεδιασμός του αυτομάτου που καθόρισε το FMS έγινε με βάση τις πληροφορίες που μας δόθηκαν στο μάθημα καθώς και τις διαφάνειες και το ηλεκτρονικό σύγγραμμα. Το σχεδιάγραμμα που δημιουργήσαμε για το αυτόματο μας παρατίθεται παρακάτω.

[ΣΧΕΔΙΑΓΡΑΜΜΑ]

## Περιγραφή των κλάσεων

**Token**

Για να αναπαραστήσουμε τα tokens, χρησιμοποιούμε την κλάση “Token”, η οποία είναι αρκετά απλή. Περιλαμβάνει δύο πεδία:

* “Token”: Το σύμβολο ή η λέξη που αναγνωρίζεται
* “tokenType”: ο τύπος του token (π.χ. λέξη-κλειδί, αριθμός, identifier κλπ.)

**LexicalAnalyzer**

Η κλάση “LexicalAnalyzer” είναι υπεύθυνη για την ανάλυση του αρχείου πηγαίου κώδικα και την παραγωγή των tokens. Η κλάση αυτή παίρνει μέσω argument το όνομα του αρχείου που θα αναλυθεί, ορίζει τις καταστάσεις του αυτόματου πεπερασμένου καταστάσεων (finite state machine) που χρησιμοποιείται για την ανάλυση και εκτελεί την λεκτική ανάλυση μέσω ενός loop που διατρέχει το αρχείο εισόδου μέχρι να συναντήσει EOF. Επιπλέον, δημιουργεί ή αν υπάρχει ήδη ανοίγει το αρχείο εξόδου (lexicalAnalyzer.txt), στο οποίο καταγράφονται τα αποτελέσματα της λεκτικής ανάλυσης

## Καταστάσεις του Finite State Machine

Το finite state machine έχει διάφορες καταστάσεις που καθορίζουν την τρέχουσα διαδικασία αναγνώρισης ενός token. Οι καταστάσεις αυτές καθώς και οι αντίστοιχες καταστάσης του αυτόματου είναι:

stateBegin: Αρχική κατάσταση (0)

Στην αρχική κατάσταση, ο αναλυτής ελέγχει τον χαρακτήρα και αποφασίζει σε ποια κατηγορία ανήκει. Ανάλογα με την κατηγορία, αλλάζει την κατάσταση. Παραμένει σε αυτή την κατάσταση όσο συναντά λευκούς χαρακτήρες.

stateLetter: Αναγνωρίζει αλφαβητικούς χαρακτήρες και λέξεις (1)

Αν ο χαρακτήρας είναι γράμμα ή ψηφίο, προστίθεται στο τρέχον token. Αν το token υπερβαίνει τους 30 χαρακτήρες, αναγνωρίζεται ως λάθος.

stateDigit: Αναγνωρίζει αριθμητικούς χαρακτήρες (2)

Αν ο χαρακτήρας είναι ψηφίο, προστίθεται στο τρέχον token. Αν ο αριθμός υπερβαίνει τα όρια -32767 και 32767, αναγνωρίζεται ως λάθος.

stateSharp: Αναγνωρίζει σχόλια και σύμβολα ομαδοποίησης (3, 4, 5)

Αν αναγνωρίσει τα σύμβολα ομαδοποίησης ‘#{‘, ‘#}’ μεταβαίνει στην κατάσταση αποδοχής. Αν αναγνωρίσει ‘##’ ξεκινάει την αναγνώριση σχολίων, δηλαδή διαβάζει έναν έναν όλους τους χαρακτήρες μέχρι να συναντήσει ξανά ‘##’ χωρίς να τους καταγράφει.

stateDivision: Αναγνωρίζει τη διαίρεση (6)

Αν αναγνωριστεί το σύμβολο ‘//’ μεταβαίνει στην κατάσταση αποδοχής. Αν βρει μόνο ‘/’ αναγνωρίζεται ως λάθος.

stateLess, stateMore, stateEqual, stateDifferent: Αναγνωρίζουν συγκριτικές πράξεις (αντίστοιχα 7, 8, 9, 10)

Αυτές οι καταστάσεις χειρίζονται τις συγκριτικές πράξεις (`<`, `>`, `==`, `!=`). Μόλις αναγνωριστεί ένα από τα παραπάνω σύμβολα η αντίστοιχη κατάσταση μεταβαίνει στην κατάσταση αποδοχής.

stateERROR: Αναγνωρίζει λάθη.

Όταν εντοπίζεται ένα λάθος, το καταγράφει και επαναφέρει την κατάσταση στην αρχική.

stateOK: Κατάσταση αποδοχής.

Δημιουργεί αντικείμενο τύπου Token και το προσθέτει στη λίστα με τα tokens. Επίσης καταγράφει το αναγνωρισμένο token στο αρχείο εξόδου και επιστρέφει στην αρχική κατάσταση

stateEOF

Αν ο αναλυτής συναντήσει EOF (εκτός σχολίων), μεταβαίνει στην κατάσταση αποδοχής και ολοκληρώνει επιτυχώς την λεκτική ανάλυση.

Σε περίπτωση που ο επόμενος χαρακτήρας είναι ‘+’ (συν), ‘-’ (μείον), ‘\*’ (αστερίσκος/επί), ‘%’, ‘,’ (κόμμα), ‘:’ (άνω κάτω τελεία), ‘(‘ (αριστερή παρένθεση), ‘(δεξιά παρένθεση) αναγνωρίζεται και ο αναλυτής επιστρέφει στην αρχική κατάσταση.

2. Συντακτικός Αναλυτής

2.1 Γενική Ιδέα

Στο κεφάλαιο αυτό παρουσιάζουμε τον συντακτικό αναλυτή. Ο συντακτικός αναλυτής ελέγχει την ορθότητα του προγράμματος βάσει μιας προκαθορισμένης γραμματικής και καθοδηγεί την παραγωγή κώδικα στις επόμενες φάσεις. Ο στόχος του είναι να διασφαλίσει ότι το πρόγραμμα που έχει γραφεί στη γλώσσα CPY είναι συντακτικά σωστό, ακολουθώντας τους κανόνες και τις δομές της γραμματικής της γλώσσας. Ο συντακτικός αναλυτής χρησιμοποιεί τα tokens που έχουν παραχθεί από τον λεξικό αναλυτή και τα οργανώνει σε μια δομή που είναι εύκολα διαχειρίσιμη για περαιτέρω επεξεργασία και εκτέλεση.

Ο συντακτικός αναλυτής ακολουθεί μια αναλυτική προσέγγιση για να επαληθεύσει την κάθε συντακτική μονάδα του προγράμματος, από τις δηλώσεις και τις εντολές, μέχρι τις εκφράσεις και τις παραμέτρους συναρτήσεων. Σε περίπτωση που εντοπιστεί κάποιο συντακτικό λάθος, Σταματάει την ανάλυση του προγράμματος και αναφέρει στον προγραμματιστή το λάθος και το σημείο που υπήρξε αυτό.

2.2 Περιγραφή των κλάσεων

Η κλάση που χρησιμοποιείται για την υλοποιήση του συντακτικού αναλυτή είναι η SyntaxAnalyzer και είναι υπεύθυνη για την λήψη και ανάλυση των tokens από τον λεκτικό αναλυτή κσθώς και για την ενσωμάτωση του ενδιάμεσου κώδικα και του πίνακα συμβόλων στην λειτουργία της.

Μέθοδοι Κλάσης:

**1. \_\_init\_\_(self, tokens)**

* **Περιγραφή**: Αρχικοποιεί την κλάση με τη λίστα των tokens και τη συμβολοπίνακα.
* **Λεπτομέρειες**:
  + Αποθηκεύει τα tokens.
  + Θέτει τον δείκτη token στο 0.
  + Ορίζει το τρέχον token.
  + Δημιουργεί έναν νέο συμβολοπίνακα.

2. **nextToken(self)**

* **Περιγραφή:** Μεταβαίνει στο επόμενο token στην λίστα των tokens.
* **Λεπτομέριες:**
  + Αυξάνει τα tokens κατά ένα.
  + Εάν δεν υπάρχουν άλλα tokens ορίζει το τρέχον στο κενό.

3. **tokenCheck(self,expectedToken)**

* **Περιγραφή**: Ελέγέγχει αν το τρέχον token είναι ίδιο με το αναμενόμενο.
* **Λεπτομέρειες:**
  + Αν το τρέχον token είναι το επιθυμητό τότε καλεί την nextToken().
  + Αν δεν ταιριάζει σταματάει το πρόγραμμα και ρίχνει εξαίρεση συντακτικού λάθους.

4. **startRule(self):**

* **Περιγραφή**: Αρχίζει την ανάλυση με τον αρχικό κανόνα της γραμματικής.
* **Λεπτομέρεις:**
  + Ελέγχει αν το τρέχον token είναι ανάμεσα στα token εκκίνησης (‘#int’, ‘def’, ‘#def’,) και καλεί τις αντίστοιχες μεθόδους (‘declarations’, ‘def\_function’, ‘def\_main’).
  + Επαναλαμβάνει αναδρομικά ώσπου να φτάσει στο τέλος του αρχείου ή να βρεθεί συντακτικό λάθος.

5. **def\_main(sefl):**

* **Περιγραφή:** Ανάλυση και επεξεργασία της δήλωσης της κύριας συνάρτησης main στο πρόγραμμα.
* **Λεπτομέρειες:**
  + Ξεκινάει μια καινούρια Procedure την οποία την τοποθετεί στον πίνακα συμβόλων.
  + Εκκινεί ένα scope που θα έχει όλες τις απαραίτητες μεταβλητές.
  + Ελέγχει ένα πρώτα ορίζεται κάποια συνάρτηση ή νεα μεταβλητή μέσα της για να καλέσει τις κατάλληλες διαδικασίες.
  + Παράγει τις κατάλληλες τετράδες για τον ενδίαμεσο κώδικα.
  + Ενημερώνει τον πίνακα συμβόλων.
  + Καλεί την code\_block για να υλοποιήσει τις υπόλοιπες λειτουργίες που πιθάνων να υπάρχουν μέσα στην main.

6. **def\_function(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται την δήλωση βοηθητικών στυναρτήσεων.
* **Λεπτομέριες:**
  + Ελέγχει συντακτικά εάν υπάρχουν τα κατάλληλα tokens για την ροή του προγράμματος σύμφωνα με τον πίνακα συμβόλων.
  + Ενημερώνει για νέες εκχωρήσεις στον πίνακα συμβόλων και ορατότητας καθώς και για την έξοδο από αυτά.
  + Ελέγχει και επεξεργάζεται τις παραμέτρους της συνάρτησεις και τις εκχωρεί στον πίνακα συμβόλων.
  + Ελέγχει και επεξεργάζεται πιθανές ορίσεις νέων μεταβλητών καθώς και τυχόν καθολικές μεταβλητές.
  + Δημιουργεί ένα χρειάζεται εκφωλιασμένες συναρτήσεις αναδρομικά.
  + Δημιουργεί νεες τετράδες είτε για εκκίνηση του block είτε για τερματισμό.
  + Καλεί την code\_block για την υλοποίηση της λογικής της βοηθητικής συνάρτησης.

7. **declarations(self)**

* **Περιγραφή:** Η συνάρτηση αυτή οριζεί νέες μεταβλητές.
* **Λειτουργία:**
  + Ελέγχει εάν το τρέχον token είναι λέξη ‘identifier’ και προχωράει στο επόμενο token.
  + Ελέγχει μετά εάν υπάρχει το ‘,’ στην περίπτωση που έχουμε πολλαπλές ορίσεις μεταβλητών.
  + Δημιουργεί μια οντότητα ‘Variable’ για κάθε μεταβλητή που έχει ορίσει.

8. **formal\_pars(self)**

* **Περιγραφή:** Ορίζει τις παραμέτρους μιας συνάρτησης.
* **Λειτουργία:**
  + Ελέγχει εάν το τρέχον τόκεν είναι ‘identifier’.
  + Αποθηκέυει σε έναν προσωρινό πίνακα την παράμετρο.
  + Εαν το επόμενο ‘token’ είναι το ‘,’ τότε σε έναν βρόχο ελέγχει για πιθανές άλλες παραμέτρους για να αποθηκέυσει στον πίνακα.
  + Επιστρέφει τις παραμέτρους κάθε συνάρτησης.

9. **statements(self):**

* **Περιγραφή:**Αναγνωρίζει και επεξεργάζεται διάφορους τύπους δηλώσεων.
* **Λειτουργία:**
  + Καθορίζει τον τύπο της δήλωσης (π.χ. ανάθεση, if, while, print) και καλεί τις αντίστοιχες μεθόδους (assignment\_statements, if\_statements, while\_statements, print\_statements).

10. assignment\_statements(self)

* Περιγραφή: Επεξεργάζεται δηλώσεις εκχώρησης.
* Λεπτομέρειες:
  + Αναγνωρίζει το όνομα της μεταβλητής στην αριστερή πλευρά της εκχώρησης.
  + Ελέγχει για το σύμβολο ‘=’ και το αναγνωρίζει ως τελεστή εκχώρησης.
  + Αναγνωρίζει την έκφραση στη δεξιά πλευρά της εκχώρησης και την επεξεργάζεται καλώντας τη μέθοδο expression.
  + Παράγει τα κατάλληλα quadruples για την εκχώρηση της τιμής.

11. **other\_statements(self,result)**

* **Περιγραφή:** Επεξεργάζεται άλλους τύπους δηλώσεων(if,while,print,return,int).
* **Λειτουργίες:**
  + Ελέγχει το πρώτο token της δήλωσης και καλεί τις αντίστοιχες μεθόδους (if\_statements, while\_statements, print\_statements, return\_statements, input\_statements) ανάλογα με τον τύπο της δήλωσης.

12. **if\_statements(self):**

* **Περιγραφή:** Επεξεργάζεται τις δηλώσεις ‘if’.
* **Λεπτομέρειες:** 
  + Ελέγχει συντακτικά αμα η λέξεις είναι η σωστές για την ροή της δήλωσης.
  + Καλεί την condition() για να διαβάσει την συνθήκη της ‘if’.
  + Παράγει τις κατάλληλες τετράδες για την λογική τις ‘if’.
  + Ενημερώνει τις τετράδες για τον προορισμό του άλματος για τις αληθείς και ψευδείς εκφράσεις.
  + Ελέγχει και καλεί εαν υπάρχουν ‘elif’,’else’ εκφράσεις.

13.**elif\_statment(self):**

* **Περιγραφή:** Επεξεργάζεται την δήλωση ‘elif’.
* **Λεπτομέριες:**
  + Ελέγχει συντακτικά αμα η λέξεις είναι η σωστές για την ροή της δήλωσης.
  + Καλεί την condition() για να διαβάσει την συνθήκη της ‘elif’.
  + Ενημερώνει τις τετράδες για τον προορισμό του άλματος για τις αληθείς και ψευδείς εκφράσεις.

14.**else\_statment(self):**

* **Περιγραφή:** Επεξεργάζεται την δήλωση ‘else’.
* **Λεπτομέριες:**
  + Ελέγχει συντακτικά αμα η λέξεις είναι η σωστές για την ροή της δήλωσης.

15.  **while\_statment(self):**

* **Περιγραφή:** Επεξεργάζεται δηλώσεις while.
* **Λεπτομέριες:**
  + Ελέγχει συντακτικά αμα η λέξεις είναι η σωστές για την ροή της δήλωσης.
  + Αναγνωρίζει την έκφραση της συνθήκης και την επεξεργάζεται καλώντας τη μέθοδο condition.
  + Παράγει τα κατάλληλα quadruples για το μπλοκ κώδικα του while.
  + Ενημερώνει τις τετράδες για τον προορισμό του άλματος για τις αληθείς και ψευδείς εκφράσεις.

16. **return\_statement(self):**

* **Περιγραφή:** Επεξεργάζεται δηλώσεις ‘return’.
* **Λεπτομέρειες:** 
  + Ελέγχει συντακτικά αν οι λέξεις ακολουθούν την ορθή ροή των κανώνων.
  + Αναγνωρίζει την έκφραση που πρέπει να επιστραφεί και την επεξεργάζεται καλώντας τη μέθοδο expression.
  + Παράγει τα κατάλληλα quadruples για την εκτύπωση της έκφρασης.

17. **print\_statment(self):**

* **Περιγραφή:** Επεξεργάζεται δηλώσεις ‘print’.
* **Λεπτομέρειες:** 
  + Ελέγχει συντακτικά αν οι λέξεις ακολουθούν την ορθή ροή των κανώνων.
  + Αναγνωρίζει την έκφραση που πρέπει να εκτυπωθεί και την επεξεργάζεται καλώντας τη μέθοδο expression.
  + Παράγει τα κατάλληλα quadruples για την εκτύπωση της έκφρασης.

18. **input\_statements(self, id):**

* **Περιγραφή:** Επεξεργάζεται δηλώσεις ‘input’.
* **Λεπτομέρειες:** 
  + Ελέγχει συντακτικά αν οι λέξεις ακολουθούν την ορθή ροή των κανώνων.
  + Αναγνωρίζει την έκφραση που πρέπει να εισαχθεί και την επεξεργάζεται καλώντας τη μέθοδο expression.
  + Παράγει τα κατάλληλα quadruples για την εκτύπωση της έκφρασης.

19. **code\_block:**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται μπλοκ κώδικα που περιέχει δηλώσεις όπως if, while, print, return, input, global, int, ή αναγνωριστικά.
* **Λεπτομέρειες:** 
  + Επαναλαμβάνει την αναγνώριση και επεξεργασία δηλώσεων όσο το τρέχον token είναι ένα από τα συγκεκριμένα keywords ή ένα αναγνωριστικό.
  + Χρησιμοποιεί τη μέθοδο statements για την επεξεργασία της κάθε δήλωσης.

20. **condition(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται συνθήκες λογικών εκφράσεων.
* **Λεπτομέρειες:** 
  + Αρχικοποιεί τις λίστες cond\_true και cond\_false.
  + Αναγνωρίζει έναν λογικό όρο χρησιμοποιώντας τη μέθοδο bool\_term.
  + Αν το τρέχον token είναι ‘or’, ελέγχει το ‘or’ token και επεξεργάζεται τον επόμενο λογικό όρο.
  + Χρησιμοποιεί τη μέθοδο Quadruple.backpatch για να διορθώσει τα quadruples που δημιουργήθηκαν.
  + Ενοποιεί τις λίστες cond\_true και cond\_false με τα αποτελέσματα των λογικών όρων.
  + Επιστρέφει τις τελικές λίστες cond\_true και cond\_false.

21. **bool\_term(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται λογικούς όρους που συνδέονται με and.
* **Λεπτομέρειες:** 
  + Αρχικοποιεί τις λίστες B\_true και B\_false.
  + Αναγνωρίζει έναν λογικό παράγοντα χρησιμοποιώντας τη μέθοδο bool\_factor.
  + Αν το τρέχον token είναι and, ελέγχει το and token και επεξεργάζεται τον επόμενο λογικό παράγοντα.
  + Χρησιμοποιεί τη μέθοδο Quadruple.backpatch για να διορθώσει τα quadruples που δημιουργήθηκαν.
  + Ενοποιεί τις λίστες B\_true και B\_false με τα αποτελέσματα των λογικών παραγόντων.
  + Επιστρέφει τις τελικές λίστες B\_true και B\_false.

22. **bool\_factor(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται λογικούς παράγοντες που μπορεί να περιέχουν not, παρενθετικές εκφράσεις ή συγκρίσεις.
* **Λεπτομέρειες:** 
  + Αν το τρέχον token είναι not, ελέγχει το not token και αναγνωρίζει τη συνθήκη με τη μέθοδο condition.
  + Αν το τρέχον token είναι (, ελέγχει το άνοιγμα και το κλείσιμο των παρενθέσεων και αναγνωρίζει τη συνθήκη με τη μέθοδο condition.
  + Αν το τρέχον token είναι αναγνωριστικό ή αριθμός, αναγνωρίζει την έκφραση και τον τελεστή σύγκρισης, και δημιουργεί τα αντίστοιχα quadruples.
  + Επιστρέφει τις λίστες B\_true και B\_false.
  + Χρησιμοποιεί τη μέθοδο Quadruple.backpatch για να διορθώσει τα quadruples που δημιουργήθηκαν.
  + Ενοποιεί τις λίστες B\_true και B\_false με τα αποτελέσματα των λογικών παραγόντων.
  + Επιστρέφει τις τελικές λίστες B\_true και B\_false.

22. **expression(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται αριθμητικές εκφράσεις.
* **Λεπτομέρειες:**
  + Αν το τρέχον token είναι μία δήλωση (if, elif, else, while, print, return, input), καλεί τη μέθοδο statements και στη συνέχεια την expression.
  + Αν το τρέχον token είναι #}, ελέγχει το κλείσιμο του μπλοκ.
  + Αναγνωρίζει το προαιρετικό πρόσημο και τον όρο της έκφρασης.
  + Αν το τρέχον token είναι ένας αριθμητικός τελεστής (+, -, =), αναγνωρίζει τον επόμενο όρο και δημιουργεί τα αντίστοιχα quadruples.
  + Επιστρέφει την τελική θέση της έκφρασης.

23. **optional\_sign(self):**

* **Περιγραφή:** Αναγνωρίζει προαιρετικά πρόσημα + ή -.
* **Λεπτομέρειες:**
  + Αν το τρέχον token είναι + ή -, επιστρέφει το αντίστοιχο πρόσημο.
  + Επιστρέφει το πρόσημο ή κενή συμβολοσειρά αν δεν υπάρχει πρόσημο.

24. **term(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται όρους εκφράσεων που περιέχουν πολλαπλασιαστικούς τελεστές (\*, //, %).
* **Λεπτομέρειες:**
  + Αναγνωρίζει τον παράγοντα της έκφρασης χρησιμοποιώντας τη μέθοδο factor.
  + Αν το τρέχον token είναι ένας πολλαπλασιαστικός τελεστής, αναγνωρίζει τον επόμενο παράγοντα και δημιουργεί τα αντίστοιχα quadruples.
  + Αν το τρέχον token είναι ένας πολλαπλασιαστικός τελεστής, αναγνωρίζει τον επόμενο παράγοντα και δημιουργεί τα αντίστοιχα quadruples.

25. **factor(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται παράγοντες εκφράσεων που μπορεί να είναι αριθμοί, keywords, παρενθετικές εκφράσεις ή αναγνωριστικά.
* **Λεπτομέρειες:**
  + Αν το τρέχον token είναι αριθμός ή keyword, επιστρέφει το token και προχωρά στο επόμενο.
  + Αν το τρέχον token είναι (, αναγνωρίζει την παρενθετική έκφραση.
  + Αν το τρέχον token είναι αναγνωριστικό, επεξεργάζεται το υπόλοιπο του αναγνωριστικού με τη μέθοδο idtail.

26. **idtail(self,name):**

* **Περιγραφή:** Επεξεργάζεται το υπόλοιπο ενός αναγνωριστικού που μπορεί να είναι κλήση συνάρτησης.
* **Λεπτομέρειες:**
  + Αν το τρέχον token είναι (, αναγνωρίζει τις παραμέτρους της συνάρτησης με τη μέθοδο actual\_pars.
  + Δημιουργεί τα κατάλληλα quadruples για την κλήση της συνάρτησης και την επιστροφή της τιμής.
  + Δημιουργεί τα κατάλληλα quadruples για την κλήση της συνάρτησης και την επιστροφή της τιμής.

27. **actual\_pars(self):**

* **Περιγραφή:** Αναγνωρίζει και επεξεργάζεται τις πραγματικές παραμέτρους μιας συνάρτησης.
* **Λεπτομέρειες:**
  + Αναγνωρίζει τις παραμέτρους χρησιμοποιώντας τη μέθοδο expression.
  + Δημιουργεί τα κατάλληλα quadruples για τις παραμέτρους της συνάρτησης.
  + Επιστρέφει το αποτέλεσμα της τελευταίας αναγνωρισμένης παραμέτρου.

27. **syntaxCorrect(self):**

* **Περιγραφή**: Ελέγχει αν η σύνταξη του προγράμματος είναι σωστή.
* **Λεπτομέρειες**:
  + Αν το τρέχον token είναι EOF, καταγράφει το τέλος του αρχείου και εκτυπώνει ότι η σύνταξη είναι σωστή.

# 4. ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ

## Γενική Ιδέα

Στο κεφάλαιο αυτό παρουσιάζουμε την υλοποίηση του πίνακα συμβόλων. Στον πίνακα αυτό κατά την διάρκεια της συντακτικής ανάλυσης και της παραγωγής ενδιάμεσου κώδικα αποθηκεύονται πληροφορίες σχετικά με τα συμβολικά ονόματα που χρησιμοποιούμε στο πρόγραμμα (μεταβλητές, συναρτήσεις, παράμετροι κλπ), πληροφορίες που χρησιμεύουν αργότερα στην σημασιολογική ανάλυση και την παραγωγή τελικού κώδικα.

Ο πίνακας αποτελείται από επίπεδα (scopes) και εγγραφές (entities). Το αρχικό/μηδενικό scope του πίνακα δημιουργείται αμέσως με την εκκίνηση της διαδικασίας συντακτικής ανάλυσης καθώς σε αυτό αποθηκεύονται μεταξύ άλλων οι καθολικές μεταβλητές, που με βάση την εκφώνηση της εργασίας μας ορίζονται στην αρχή ενός προγράμματος.

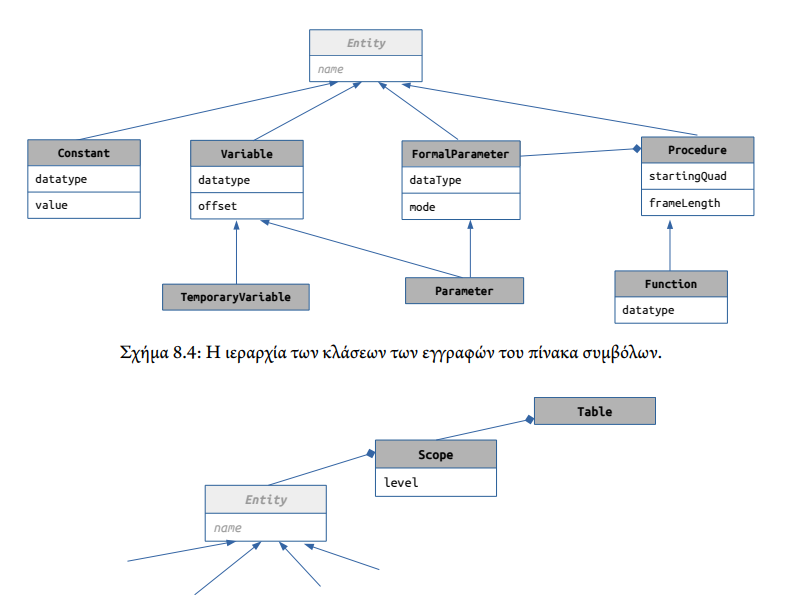
Στην συνέχεια, για κάθε συνάρτηση που ορίζεται εισάγουμε ένα νέο entity τύπου Function (ή Procedure, σε περίπτωση διαδικασίας) στον πίνακα και δημιουργούμε ένα νέο scope, στο οποίο εισάγονται οι πληροφορίες για την εν λόγω συνάρτηση όπως η αρχική τηε τετράδα, το μήκος του εγγραφήματος δραστηριοποίησης της, οι τοπικές της μεταβλητές, οι προσωρινές και καθολικές μεταβλητές που αυτή χρησιμοποιεί και οι παράμετροί της.

Μόλις ολοκληρωθεί η διαδικασία μετάφρασης της συνάρτησης, τυπώνουμε την τρέχουσα μορφή του πίνακα σε ένα εξωτερικό αρχείο (symbolTable.sym) και αφαιρούμε το scope που της αντιστοιχεί από τον πίνακα.

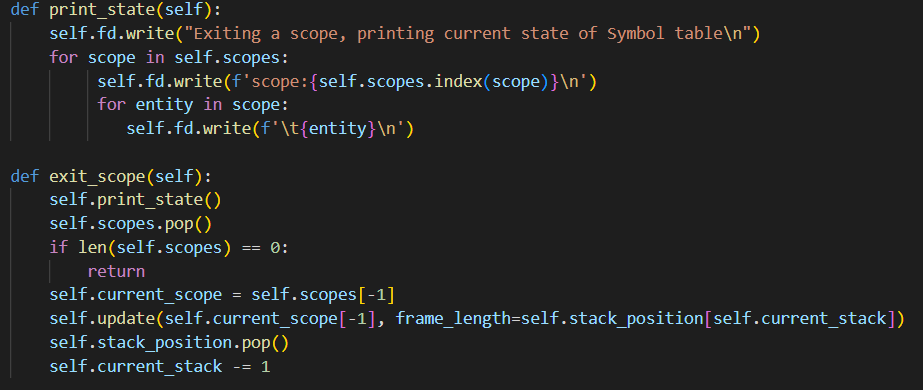
## Περιγραφή των κλάσεων

Με μία ματιά στον κώδικα, είναι εμφανές ότι ο πίνακας συμβόλων έχει αρκετά περισσότερες κλάσεις από ότι τα υπόλοιπα τμήματα του κώδικα.

Για την αναπαράσταση των στοιχείων του πίνακα ακολουθήσαμε μια καθαρά αντικειμενοστραφή σχεδίαση, όπως αυτή παρουσιάζεται στο ηλεκτρονικό σύγγραμμα που μας δώθηκε. Έτσι δημιουργήσαμε την αφηρημένη κλάση Entity με μοναδικό πεδίο το name, το μόνο κοινό στοιχείο κάθε εγγραφής του πίνακα και στη συνέχεια, εκμεταλλευόμενοι την κληρονομικότητα των δημιουργήσαμε την ιεραρχική σχεδίαση που προτείνεται στο σύγγραμμα όπως φαίνεται στην παρακάτω εικόνα

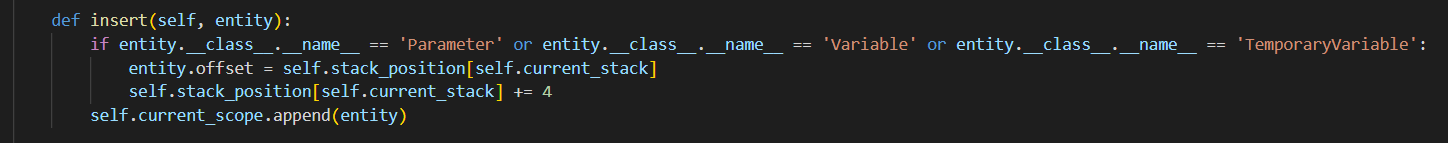


Για κάθε νέα συνάρτηση που καλείται, η enter\_scope φροντίζει να μετακινήσει τον sp κατά 12 bytes, 4 για καθένα εκ των

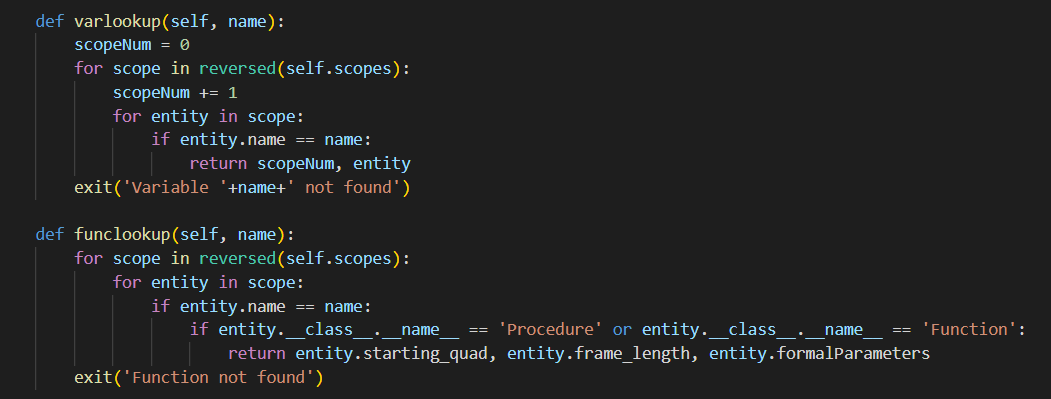


Κάθε φορά που ολοκληρώνεται η μετάφραση μιας συνάρτησης και πριν το αντίστοιχο scope αφαιρεθεί από τον πίνακα συμβόλων, καλείται η μέθοδος print\_state για να εκτυπωθεί η τρέχουσα μορφή του πίνακα.

Στην συνέχεια, η μέθοδος exit\_scope αφαιρεί το τρέχον scope από τον πίνακα και καθώς και το εγγράφημα δραστηριοποίησης της. Σε περίπτωση που δεν υπάρχουν πλέον scopes, δηλαδή ολοκληρώθηκε και η μετάφραση της main, επιστρέφει.



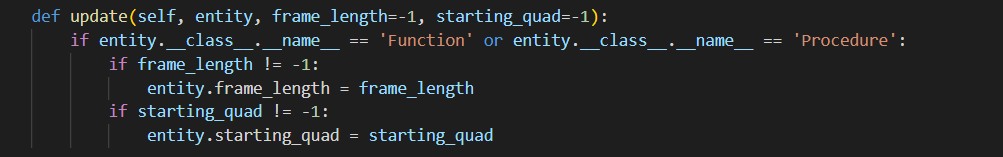
Η μέθοδος αυτή είναι υπεύθυνη για την προσθήκη των entities στον πίνακα. Δεσμεύει χώρο 4 bytes στην στοίβα μετακινώντας κατάλληλα τον pointer και στη συνέχεια προσθέτει το νέο entity.



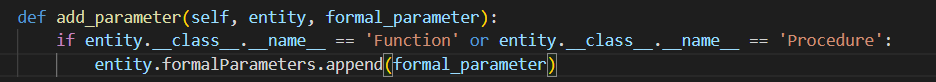
Οι συναρτήσεις αυτές εκτελούν την διαδικασία αναζήτησεις μεταβλητών ή συναρτήσεων/διαδικασιών αντίστοιχα.

Η μέθοδος varlookup αναζητά μία μεταβλητή ξεκινώντας από το τρέχων scope και μεταβαίνοντας στο προηγούμενο μέχρι να φτάσει και στο εξωτερικό/μηδενικό δηλαδή το global. Επιστρέφει την απόσταση από το τρέχων scope του scope στο οποίο βρέθηκε η μεταβλητή, δηλαδή 1 αν βρέθηκε στο τρέχων scope, 2 αν βρέθηκε στο αμέσως προηγούμενο κοκ. Σε περίπτωση που η μεταβλητή βρέθηκε στο αρχικό/μηδενικό scope επιστρέφει 0.

Η μέθοδος funclookup ακολουθεί παρόμοια διαδικασία. Αναζητά μία μεταβλητή ξεκινώντας από το τρέχων scope και μεταβαίνοντας στο προηγούμενο. Επιστρέφει την αρχική τετράδα, το frame length και την λίστα παραμέτρων της συνάρτησης ή διαδικασίας.



Η μέθοδος update, ενημερώνει τα πεδία μιας συνάρτησης/διαδικασίας συμπληρώνοντας τις σωστές τιμές. Καλείται στην αρχή και το τέλος της διαδικασίας μετάφρασης για να συμπληρώσει την αρχική τετράδα και το frame length αντίστοιχα.



Η μέθοδος add\_parameter, προσθέτει παραμέτρους κατάλληλα στη λίστα παραμέτρων μιας συνάρτησης ή μιας διαδικασίας