Προγραμματιστική άσκηση στο μάθημα των Μεταφραστών

Γεώργιος Ευάγγελος Θεοδώρου - 3231
Ιωάννης Πρίφτη - 3321

Το πρόβλημα

Ζητείται πρόγραμμα το οποίο μεταφράζει προγράμματα της γλώσσας Minimal++ κάνοντας λεκτική, συνταντακτική και σημασιολογική ανάλυση, παραγωγή ενδιάμεσου κώδικα, δημιουργία πίνακα συμβόλων και ισοδύναμου προγράμματος σε γλώσσα C καθώς και παραγωγή τελικού εκτελέσιμου κώδικα σε assembly.

Υλοποίηση

Λεκτικός αναλυτής

Ο λεκτικός αναλυτής ανοίγει αρχεία κειμένου με την κατάληξη .min διαβάζει χαρακτήρα-χαρακτήρα το αρχείο αγνοώντας κενούς χαρακτήρες και χαρακτήρες αλλαγής γραμμής και ενοποιεί τους χαρακτήρες σε λεκτικές μονάδες τις οποίες και ελέγχει αν ανοίξουν σε μια από τις παρακάτω κατηγορίες:
- Reserved words
- Relational operants
- Multiplication operants
- Add operants
- Delimiters
- Grouping symbols
- Comments
- Assignment operant
- Αριθμός (in range of -32767,32767)
- Όνομα μεταβλητής/συνάρτησης/διαδικασίας/προγράμματος

Μετά επιστρέφει την λεκτική αυτή μονάδα για να χρησιμοποιηθεί από τον συντακτικό αναλυτή. Ο λεκτικός αναλυτής ακόμα αναλαμβάνει να μετράει και τις γραμμές ώστε να γνωρίζει ο συντακτικός αναλυτής σε ποια βρίσκεται. Επίσης αναλαμβάνει να προσπερνά χαρακτήρες που βρίσκονται μέσα σε σχόλια και να ελέγχει αν αυτά είναι ορισμένα σωστά. Στην υλοποίηση μας ο λεκτικός αναλυτής αποτυγχάνει μερικές φορές στην περίπτωση που τα σχόλια δεν κλείνουν ποτέ. Τέλος ο λεκτικός αναλυτής είναι ικανός να εντοπίζει το τέλος του αρχείου (EOF) και να το κλείνει.

Συντακτικός αναλυτής

Ο συντακτικός αναλυτής αποτελείται από επιμέρους συναρτήσεις καθεμία από τις οποίες είναι μια έκφραση της γραμματικής που ορίζει την γλώσσα. Συγκεκριμένα:

- Η συνάρτηση program ελέγχει αν υπάρχει η δεσμευμένη λέξη program. Επίσης ελέγχει αν το id του προγράμματος υπάρχει ή είναι δεσμευμένη λέξη και αν το block περικλείεται από αγκύλες. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση declarations ελέγχει αν εμφανίζεται τουλάχιστον μια φορά η δεσμευμένη λέξη declare και αν στο τέλος της δήλωσης υπάρχει ερωτηματικό(;) . Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση varlist ελέγχει αν υπάρχουν id και αν υπάρχουν τουλάχιστον δυο να χωρίζονται με κόμμα. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση subprogram ελέγχει αν υπάρχει η δεσμευμένη λέξη function/precedure. Επίσης ελέγχει αν το id του αντίστοιχου υποπρογράμματος υπάρχει ή είναι δεσμευμένη λέξη. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση funcbody ελέγχει αν το block περικλείεται από αγκύλες. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.

- Η συνάρτηση formalpars ελέγχει αν το formalparlist περικλείεται από παρενθέσεις. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση formalparitem ελέγχει αν υπάρχει η δεσμευμένη λέξη in/inout. Επίσης ελέγχει αν το id υπάρχει ή είναι δεσμευμένη λέξη. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση statements ελέγχει αν το statement περικλείεται από αγκύλες και αν υπάρχουν τουλάχιστον δύο το τελευταίο να μην έχει ερωτηματικό(;). Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος. Διευκρινίζουμε ότι η συνάρτηση δεν δουλεύει στην περίπτωση που υπάρχει ένα μόνο statement χωρίς αγκύλες.
- Η συνάρτηση assignmentStat ελέγχει αν το id υπάρχει ή είναι δεσμευμένη λέξη. Επίσης ελέγχει αν υπάρχει η δεσμευμένη λέξη :=. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση ifStat ελέγχει αν υπάρχει η δεσμευμένη λέξη if. Επίσης ελέγχει αν το condition περικλείεται από παρενθέσεις και αν υπάρχει η δεσμευμένη λέξη then. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάντηση elsePart ελέγχει αν υπάρχει ή όχι η δεσμευμένη λέξη else.
- Η συνάντηση loopStat ελέγχει αν υπάρχει η δεσμευμένη λέξη loop. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάντηση exitStat ελέγχει αν υπάρχει η δεσμευμένη λέξη exit. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση forcaseStat ελέγχει αρχικά αν υπάρχει η δεσμευμένη forcase. Έπειτα ελέγχει αν υπάρχουν μια ή περισσότερες προτάσεις της μορφής: δεσμευμένη λέξη when, condition που περικλείονται από παρενθέσεις, ακολουθούμενα από “:”. Τέλος ελέγχει αν υπάρχει και η δεσμευμένη λέξη default: . Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση incaseStat ελέγχει αρχικά αν υπάρχει η δεσμευμένη incase. Έπειτα ελέγχει αν υπάρχουν μια ή περισσότερες προτάσεις της μορφής: δεσμευμένη λέξη when, condition που περικλείονται από παρενθέσεις, ακολουθούμενα απο “:”.Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάντηση returnStat ελέγχει αν υπάρχει η δεσμευμένη λέξη return. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάντηση printStat ελέγχει αν υπάρχει η δεσμευμένη λέξη print. Επίσης ελέγχει αν το expression περικλείεται από παρενθέσεις Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάντηση inputStat ελέγχει αν υπάρχει η δεσμευμένη λέξη input. Επίσης ελέγχει αν το id υπάρχει ή είναι δεσμευμένη λέξη και αν περικλείεται από παρενθέσεις Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση actualpars ελέγχει αν το actualparlist περικλείεται από παρενθέσεις. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση actualparitem ελέγχει αν υπάρχει η δεσμευμένη λέξη in ή αν υπάρχει η δεσμευμένη λέξη λέξη inout ακολουθούμενη από το id το οποίο ελέγχει υπάρχει ή είναι δεσμευμένη λέξη. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση condition ελέγχει αν υπάρχουν περισσότερα από δύο boolterm να είναι χωρισμένα με την δεσμευμένη λέξη or. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση boolterm ελέγχει αν υπάρχουν περισσότερα από δύο boolfactor να είναι χωρισμένα με την δεσμευμένη λέξη and. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση boolfactor ελέγχει αν υπάρχει η δεσμευμένη λέξη not ακολουθούμενη από condition που περικλείονται από παρενθέσεις ή αν υπάρχουν condition που περικλείονται από παρενθέσεις ή κάτι άλλο που πρέπει να καλεσθεί. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση relationalOper ελέγχει αν υπάρχει μια από τις δεσμευμένες λέξεις =, <=, >=, >, <, <>. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση addOper ελέγχει αν υπάρχει μια από τις δεσμευμένες λέξεις +/-. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.
- Η συνάρτηση mulOper ελέγχει αν υπάρχει μια από τις δεσμευμένες λέξεις * ή /. Σε αντίθετη περίπτωση εμφανίζει αντίστοιχο μήνυμα λάθους και την αντίστοιχη γραμμή που είναι το λάθος.

Οι υπόλοιπες συναρτήσεις που δεν αναφέρθηκε αναλυτικά η λειτουργικότητα τους δεν κάνουν κάποιον έλεγχο απλά καλούν άλλες συναρτήσεις. Επίσης για το id ελέγχουμε πάντα αν είναι δεσμευμένη λέξη ή εάν το όνομα που δίνεται έχει ξαναχρησιμοποιηθεί.

Παραγωγή ενδιάμεσου κώδικα

Ως προετοιμασία για την διαδικασία της παραγωγής ενδιάμεσου κώδικα υλοποιήσαμε τις βοηθητικές συναρτήσεις nextquad(), genquad(op, x, y, z), newtemp(), emptylist(), makelist(x), merge(list1, list2), backpatch(list, z).

Μετά επεκτείναμε τις ακόλουθες συναρτήσεις έτσι ώστε να περιλαμβάνουν κλήσεις των παραπάνω συναρτήσεων:

- Στην συνάρτηση program κρατάμε τετράδες που σηματοδοτούν την λήξη του κυρίου προγράμματος.
- Στην συνάρτηση block κρατάμε τετράδες που σηματοδοτούν την έναρξη του κυρίου προγράμματος ή κάποιου υπό προγράμματος.
- Στην συνάρτηση subprogram κρατάμε τετράδες που σηματοδοτούν την λήξη του υπό προγράμματος.
- Στην συνάρτηση funcbody κρατάμε τετράδες που σηματοδοτούν την λήξη του υπό προγράμματος.
- Στην συνάρτηση ifStat δημιουργούμε δύο λίστες (bTrue, bFalse) οι οποίες περιέχουν την τετράδα στην οποία πρέπει να μεταφερθεί η εκτέλεση του προγράμματος στην περίπτωση που αληθεύει ή όχι η συνθήκη του if.
- Στην συνάρτηση whileStat δημιουργούμε δύο λίστες (bTrue, bFalse) οι οποίες περιέχουν την τετράδα στην οποία πρέπει να μεταφερθεί η εκτέλεση του προγράμματος στην περίπτωση που αληθεύει ή όχι η συνθήκη του while.
- Στην συνάρτηση forcaseStat δημιουργούμε δύο λίστες (bTrue, bFalse) οι οποίες περιέχουν την τετράδα στην οποία πρέπει να μεταφερθεί η εκτέλεση του προγράμματος στην περίπτωση που αληθεύει ή όχι η συνθήκη του forcase.
- Στην συνάρτηση returnStat κρατάμε τετράδα που περιέχει την επιστρεφόμενη τιμή.
- Στην συνάρτηση printStat κρατάμε τετράδα που περιέχει την τιμή που θα εκτυπωθεί.
- Στην συνάρτηση actualpars κρατάμε τετράδες που περιέχουν μια καινούργια προσωρινή μεταβλητή που επιστρέφει η newtemp() και τον τροπο περάσματος της και τετράδες που περιέχουν την τιμή κλήσης μιας συνάρτησης.
- Στην συνάρτηση actualparitem κρατάμε τετράδες που περιέχουν μια καινούργια προσωρινή μεταβλητή που επιστρέφει η newtemp() και τον τροπο περάσματος της και τετράδες που περιέχουν τον τρόπο περάσματος μιας μεταβλητής κατά την κλήση της συνάρτησης.
- Στην συνάρτηση inputStat κρατάμε τετράδα που περιέχει την τιμή που θα εισαχθεί.
- Στην συνάρτηση condition συμπληρώνουμε με την backpatch όσες τετράδες μπορούν να συμπληρωθούν μέσα στον κανόνα και κάνουμε merge τις τετράδες που δεν μπορούν ακόμα να συμπληρωθούν και αντιστοιχούν σε αποτίμηση λογικής παράστασης
- Στην συνάρτηση boolterm συμπληρώνουμε με την backpatch όσες τετράδες μπορούν να συμπληρωθούν μέσα στον κανόνα και κάνουμε merge τις τετράδες που δεν μπορούν ακόμα να συμπληρωθούν και αντιστοιχούν σε αποτίμηση λογικής παράστασης
- Στην συνάρτηση boolfactor δημιουργούμε μη συμπληρωμένες τετράδες για την αληθή και την μια αληθή αποτίμηση της relopTemp και τις εισάγουμε στην λίστα των μη συμπληρωμένων τετράδων
- Στην συνάρτηση expression δημιουργούμε τετράδες που κρατάνε τα ορίσματα και τους τελεστές της έκφρασης
- Στην συνάρτηση term δημιουργούμε τετράδες που κρατάνε τα ορίσματα και τους τελεστές της έκφρασης

Ισοδύναμο πρόγραμμα σε γλώσσα C

Για την δημιουργία του ισοδύναμου προγράμματος σε C υλοποιήσαμε την βοηθητική συνάρτηση minQuadToCquad η οποία διατρέχει τον πίνακα που αποθηκεύει τις τετράδες. Στις τετράδες αυτές προσθέτει τα κατάλληλα ορίσματα έτσι ώστε να αποτελούν ορθό κώδικα C. Τέλος η

συνάρτηση αυτή επιστρέφει κάθε έτοιμη τετράδα στην συνάρτηση MakeCfile η οποία αναλαμβάνει να τις τοποθετήσει σε ένα .c αρχείο το οποίο δημιουργεί.

Πίνακας Συμβόλων

Ως προετοιμασία για την υλοποίηση του πίνακα συμβόλων δημιουργήσαμε την κλάση Entity, την κλάση Scope, την κλάση Argument και την συνάρτηση scopeSearch.

- Η κλάση Entity περιέχει τα πεδία name, entityType, offset, startQuad, argumentList, framelength, parMode και nextEntity καθώς και setters, getters και constructors για κάποια απο αυτά τα πεδία.
- Η κλάση Scope περιέχει τα πεδία entityList, nestingLevel και enclosingScore καθώς και setters, getters και constructors για κάποια απο αυτά τα πεδία.
- Η κλάση Argument περιέχει τα πεδία parMode και argMode και τον constructor τους.

Μετά επεκτείναμε τις ακόλουθες συναρτήσεις έτσι ώστε να υλοποιούν το πίνακα συμβόλων:

- Στην συνάρτηση program αρχικοποιούμε το scope με βάση το nesting και δημιουργούμε το entity της main και το προσθέτουμε στο scope.
- Στην συνάρτηση block εάν καλείται από εσωτερική συνάρτηση ελέγχουμε αν το Entity ανήκει στην main ή σε κάποια άλλη συνάρτηση και καθορίζουμε που θα μεταφερθεί η ροή του προγράμματος
- Στην συνάρτηση varlist αφού ελέγξουμε αν υπάρχει μεταβλητή, προσθέτουμε ένα νέο Entity στο scope επειδή συναντήσαμε δήλωση μεταβλητής που δεν έχει ξαναχρησιμοποιηθεί.
- Στην συνάρτηση subprogram αρχικά αυξάνουμε το nesting κατά ένα. Μετά φτιάχνουμε ένα νέο Entity, ψάχνουμε στο scope αν υπάρχει ήδη η συνάρτηση ή η διαδικασία και αν δεν υπάρχει το προσθέτουμε στο καινούργιο scope με τα κατάλληλα ορίσματα. Επίσης όταν τελειώσει το υποπρόγραμμα διαγράφουμε την εγγραφή του scope, όλες τις λίστες με τα Entity και τα Argument που εξαρτώνται απο αυτήν και μειώνουμε το nesting κατά ένα.
- Στην συνάρτηση formalparitem δημιουργούμε ένα Entity για κάθε όρισμα της κληθείσας συνάρτησης ή διαδικασίας και το προσδίδουμε τις κατάλληλες ιδιότητες για κάθε μια από τις δύο περιπτώσεις(in, inout).
- Στην συνάρτηση assignmentStat ελέγχουμε αν η μεταβλητή υπάρχει στο scope, δηλαδή έχει δηλωθεί πιο πριν.
- Στην συνάρτηση inputStat ελέγχουμε αν η μεταβλητή υπάρχει στο scope, δηλαδή έχει δηλωθεί πιο πριν.
- Στην συνάρτηση actualpars αφού ελέγξουμε αν υπάρχει η μεταβλητή στο scope, προσθέτουμε ένα νέο Entity στο scope με τα κατάλληλα ορίσματα επειδή συναντήσαμε δήλωση μεταβλητής που δεν έχει ξαναχρησιμοποιηθεί.
- Στην συνάρτηση actualparlist δημιουργούμε ένα νέο Entity κάτω από το οποίο θα «κάτσουν» τα ορίσματα της συνάρτησης ή της διαδικασίας.
- Στην συνάρτηση actualparitem δημιουργούμε ένα Argument για τον τύπο περάσματος της μεταβλητής το οποίο και προσθέτουμε στο αντίστοιχο Entity
- Στην συνάρτηση expression ελέγχουμε αρχικά αν υπάρχει η μεταβλητή στο scope. Αν υπάρχει δημιουργούμε ένα Entity το οποίο και προσθέτουμε στο scope με το κατάλληλα ορίσματα. Αυτή η διαδικασία γίνεται επαναληπτικά όσο υπάρχει ο τελεστής +/-. **Σημείωση: αυτό γίνεται μέσα στα σχόλια που ξεχάσαμε να αφαιρέσουμε.**
- Στην συνάρτηση term ελέγχουμε αρχικά αν υπάρχει η μεταβλητή στο scope. Αν υπάρχει δημιουργούμε ένα Entity το οποίο και προσθέτουμε στο scope με το κατάλληλα ορίσματα. Αυτή η διαδικασία γίνεται επαναληπτικά όσο υπάρχει ο τελεστής * ή /.

Παραγωγή τελικού κώδικα

Για την παραγωγή του τελικού κώδικα υλοποιήσαμε τις βοηθητικές συναρτήσεις gnvlcode, loadvr και storerv όπως ορίζονται στις σημειώσεις του μαθήματος. Για την δημιουργία του τελικού κώδικα υλοποιήσαμε την βοηθητική συνάρτηση minToAsm η οποία διατρέχει τον πίνακα που αποθηκεύει τις τετράδες. Στις τετράδες αυτές προσθέτει τα κατάλληλα ορίσματα έτσι ώστε να αποτελούν ορθό κώδικα Assembly. Τέλος η συνάρτηση αυτή επιστρέφει κάθε έτοιμη τετράδα στην

συνάρτηση makeAsmFile η οποία αναλαμβάνει να τις τοποθετήσει σε ένα .asm αρχείο το οποίο δημιουργεί.

Ιδιαιτερότητες της υλοποίησης

Το πρόγραμμα μας δεν παράγει το αρχείο .asm λόγω σφάλματος σε κάποια από της συναρτήσεις που παράγουν τον τελικό κώδικα το οποίο πιθανόν προέρχεται από κάποια παράβλεψη στο πίνακα συμβόλων. Το πρόβλημα αυτό δεν καταφέραμε να το επιλύσουμε λόγω έλλειψης χρόνου.

Testing

```
program  testino
{
        declare a, c;
        function one(in numbers)
        {
            declare result, i;
            {
                i:=0;
                while(i<10)
                {
                        result:=numbers+i;
                        i:=i+1
                }
                return (result)
            }
        }
        {
                input (c);
                input (numbers);
                print (a)
        }
}
```

```
program  test_3
{
        declare a, c;
        procedure one(in arg, inout numbers)
        {
            declare result, i;
            {
                i:=0;
                while(i<10)
                {
                        result:=numbers+i;
                        i:=i+1
                }
                print (result)
        }
        input (result);
        if(arg>numbers and result>0) then
        {
                arg:=result+numbers
        }else
        {
                arg:=arg+result
                /* test_3 */
        }
        print(arg)
        }
        {
                input (c);
                c:=2;
                print (a)
        }
}
```

```
program  test_2
{
        declare a, c;
        procedure one(inout numbers)
        {
            declare result, i;
            {
                i:=0;
                while(i<10)
                {
                        result:=numbers+i;
                        i:=i+1
                        }
                print (result)
                }
        }
        {
        input (c);
        /* test_2 */
        c:=2;
        print (a)
        }
}
```

Τα παραπάνω προγράμματα χρησιμοποιήθηκαν για την αποσφαλμάτωση και τον έλεγχο του προγράμματος.