Μεταγλωττιστές

Άσκηση Μέρος 4

Σημασιολογικός Αναλυτής

Έχοντας ολοκληρώσει την υλοποίηση του συντακτικού αναλυτή, συνεχίζουμε υλοποιώντας τον σημασιολογικό αναλυτή. Η δράση του ολοκληρώνεται σε τρία περάσματα. Αρχικά δημιουργούμε μία στοίβα (Deque) από πίνακες (SymTable), οι οποίοι περιέχουν εγγραφές (SymTableEntry) για κάθε ένα επίπεδο (scope). Οι εγγραφές αυτές αντιπροσωπεύουν τους AST κόμβους που δημιουργεί ο συντακτικός αναλυτής στο προηγούμενο βήμα. Στο επόμενο βήμα, συλλέγονται όλα τα σύμβολα μέσα στους κόμβους και ελέγχεται η περίπτωση για διπλά σύμβολα. Στο τρίτο και τελευταίο βήμα γίνονται σημασιολογικοί έλεγχοι για όλους τους κόμβους.

Αναλυτικότερα, στη πρώτη φάση, η κλάση SymTableBuilderASTVisitor είναι υπεύθυνη για την εκχώρηση, αντιπροσωπευτικών για τους συντακτικούς κόμβους, αντικειμένων της κλάσης που υλοποιεί το SymTable, συγκεκριμμένα το HashSymTable, διαμορφώνοντας σωστά τα επίπεδα κάθε scope. Αυτό γίνεται εφικτό με δύο μεθόδους την pushEnviroment() και την popEnviroment(). Κάθε φορά που ένα Compound Statement κάνει accept τον ASTVisitor από μία δήλωση συνάρτησης δημιουργείται ένα καινούργιο επίπεδο. Στο αρχικό επίπεδο θα πρέπει να βρίσκονται όλες οι συναρτήσεις, όπως και η main συνάρτηση. Στο τέλος της φάσης, προσθέτουμε τον βασικό ορισμό της print, που αποτελεί το “standard library”, να είναι διαθέσιμο στο δέντρο.
Στη δεύτερη φάση, η κλάση CollectSymbolsASTVisitor είναι υπεύθυνη για την δήλωση των μεταβλητών και των συναρτήσεων στο SymTable, που αποτελεί τον πίνακα συμβόλων. Κυρίως δίνεται βάση στους ορισμούς τους και την αρκιχή δήλωση τύπων. Σε αυτή τη φάση λάθη όπως διπλή δήλωση μεταβλήτών ή συναρτήσεων στο ίδιο scope ανιχνεύονται. Υποστηρίζεται όμως η δυνατότητα να έχουμε μια μεταβλητή και μια συνάρτηση με το ίδιο αναγνωριστικό. Στο τέλος της φάσης ελέγχουμε εάν υπάρχει η main συνάρτηση με return value type ως void και καθόλου παραμέτρους.
Στην τρίτη φάση, η κλάση CollectTypesASTVisitor είναι υπεύθυνη για τον υπολογισμό του τύπου κάθε οντότητας, όχι μόνο μεταβλητών και συναρτήσεων, αλλά και εκφράσεων. Έτσι μπορεί να γίνουν και οι έλεγχοι τύπων που ταιριάζουν για κάθε περίπτωση. Οι αριθμητικές πράξεις γίνονται μόνο μεταξύ αριθμών. Οι σχεσιακοί τελεστές εφαρμόζονται μόνο μεταξύ αριθμών και χαρακτήρων. Στις προτάσεις while και if οι εκφράσεις πρέπει να είναι ακέραιου τύπου. Οι τελεστές λογικού AND, OR και NOT εφαρμόζονται μόνο σε εκφράσεις ακέραιου τύπου. Η έκφραση σε return συμφωνεί με τον τύπο επιστροφής της συνάρτησης, με τρόπο ανάλογο της ανάθεσης. Στις προτάσεις ανάθεσης ο τύπος στα αριστερά της ανάθεσης πρέπει είναι μεγαλύτερος ή ίσος τύπος από τον τύπο δεξιά της ανάθεσης. Θεωρούμε πως οι τύποι από τον μεγαλύτερο στον μικρότερο είναι float, integer, character. Προσοχή στο ότι δεν υποστηρίζουμε overloading, οπότε εάν βρεθεί συνάρτηση με ίδιο αναγνωριστικό τότε τυπώνεται μήνυμα λάθους από την προηγούμενη φάση. Κλήση σε συνάρτηση έχει τον σωστό αριθμό παραμέτρων και κάθε παράμετρος να συμφωνεί με τον τύπο της παραμέτρου. Η πρόταση break μπορεί να εμφανίζεται μόνο μέσα σε επανάληψη καθώς και η πρόταση continue. Όλα αυτά είναι εγγυημένα από το τρίτο πέρασμα.

Για να μπορεί να γίνει σωστά η ανάλυση με τους ελέγχους στις κλάσεις αυτές χρησιμοποιούμε τις κλάσεις ASTUtils και TypeUtils ως βοηθήματα. Επιπλέον κανόνες έχουν προστεθεί στην κλάση TypeUtils ώστε να γίνεται σωστά ο έλεγχος των τύπων. Επίσης για κάθε entry ή ζευγάρι από entries (1 για function 1 για variable) στο symbol table χρησιμοποιούμε μια wrapper class, την TableEntryWrapper. Το HashSymTable υποστηρίζει την ύπαρξη του wrapper. Σε όλες τις φάσεις διαχειριζόμαστε τους τύπους με την βιβλιοθήκη ASM, ώστε να μπορούμε να κάνουμε πιό εύκολα παραγωγή bytecode σε επόμενο μέρος.

Για να δούμε πως οι έλεγχοι γίνονται σωστά δίνουμε στον compiler λανθασμένα αρχεία με σκοπό να δούμε πως η μεταγλώττιση σταματά στο σωστό σημείο.

Παρακάτω φαίνονται μερικές δοκιμές των ελέγχων.

1. Έλεγχος για την ύπαρξη main συνάρτησης.

```
1 int x;
2 char y;
3 int myfuct(char r){
4     return 0;
5 }
```

Τα αποτελέσματα είναι τα εξής :

```
15:35:13.444 [main] INFO  Compiler - Scanning file nomaintest.txt
15:35:13.485 [main] INFO  Compiler - Constructed AST
Collect Symbols phase
15:35:13.498 [main] ERROR Compiler - 1:1: No main() function declared!
```

2. Έλεγχος για την μετατροπή ενός ακεραίου σε float.

```
1 void main(){
2     int number = 5;
3     float x = 5.5;
4
5     x=number;
6     number = x;
7 }
```

Στα αποτελέσματα φαίνεται πως το πρόβλημα είναι στην γραμμή 6, όταν δηλαδή ένας αριθμός float εκχωρείται σε έναν ακέραιο. Αντίθετα με την γραμμη 5 όπου εκχωρείται ένας ακέραιος σε έναν float.

```
15:57:07.198 [main] INFO  Compiler - Scanning file floatTOint.txt
15:57:07.243 [main] INFO  Compiler - Constructed AST
Collect Symbols phase
Collect Types phase
15:57:07.261 [main] ERROR Compiler - 6:2: Type mismatch: Cannot convert from D to I
```

3. Έλεγχος για την ύπαρξη των break και continue εντός επαναλήψεων:

```
1  void main(){
2      int x=0;
3      while(x<10){
4          if(x==5){
5              break;
6          }
7          x=x+1;
8      }
9      break;
10 }
```

Στα αποτελέσματα φαίνεται ο έλεγχος της εντολής break, η οποία πρέπει να είναι εντός κάποιας επανάληψης (όπως στην γραμμή 5) και όχι εκτώς(όπως στην γραμμή 9). Αυτός είναι και ο λόγος που εμφανίζεται το σφάλμα.

```
16:09:38.837 [main] INFO  Compiler - Scanning file randomBreakTest.txt
16:09:38.887 [main] INFO  Compiler - Constructed AST
Collect Symbols phase
Collect Types phase
16:09:38.904 [main] ERROR Compiler - 9:2: This is a random break.
```

4. Έλεγχος για την σωστή ανάθεση τιμών σε πίνακα.

```
1 void main(){
2 char[] charTable;
3     charTable = new char[5];
4     charTable[0] = 'T';
5     charTable[1] = 'Y';
6     charTable[2] = 100 ;
7     return;
8 }
```

Ενώ στην γραμμή 4 και 5 η εκχωρήσεις γίνονται κατά το δοκούν, στην γραμμή 6 υπάρχει σφάλμα και παίρνουμε το ανάλογο μήνυμα.

```
20:46:45.347 [main] INFO  Compiler - Scanning file charTableTest.txt
20:46:45.387 [main] INFO  Compiler - Constructed AST
Collect Symbols phase
Collect Types phase
20:46:45.410 [main] ERROR Compiler - 6:2: Type mismatch: Cannot convert from I to C
```

5. Έλεγχος για την εκχώρηση τιμής σε μη υπαρκτή θέση του πίνακα.

```
1 void main(){
2 char[] charTable;
3     charTable = new char[5];
4     charTable[0] = 'T';
5     charTable[1] = 'Y';
6     charTable[7] = 100 ;
7     return;
8 }
```

Στην γραμμή 6 γίνεται προσπάθεια να εκχωρηθεί μια τιμή σε μια θέση του πίνακα η οποία δεν υπάρχει.

```
21:11:50.729 [main] INFO  Compiler - Scanning file charTableTest.txt
21:11:50.769 [main] INFO  Compiler - Constructed AST
Collect Symbols phase
21:11:50.779 [main] ERROR Compiler - 6:2: Array index out of bounds.
```

Η εργασία ακολουθεί την δομή των εργαστηρίων, με εξαίρεση τα αρχεία java που προσθέτουμε όταν χρειαστεί. Για ελέγχους, όντας στον αρχικό του project:

mvn package

java -jar target/compiler-0.0.4.jar input1.txt

Ο κώδικας είναι στον κατάλογο compiler του παραδοτέου.