Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών: Εργαστηριακή Άσκηση 2012-2013

27 Μαρτίου 2013

Περίληψη

Σκοπός της παρούσας εργασίας είναι η εξοικείωσή σας με τις θεμελιώδεις θεωρητικές και πρακτικές πτυχές της λεκτικής και συντακτικής ανάλυσης. Η εργασία αποτελείται από ένα θεωρητικό μέρος που αφορά τα βασικά θεωρητικά εργαλεία που πρέπει κανείς να γνωρίζει πριν καταπιαστεί με την υλοποίηση ενός λεκτικού/συντακτικού αναλυτή και έπειτα σας ζητά να φτιάξετε με την βοήθεια των μεταεργαλείων flex και bison έναν parser για μία απλή γλώσσα προγραμματισμού.

1 Θεωρητικό Τμήμα

1. Δίνεται η παρακάτω γραμματική:

$$
\begin{array}{lcl}
<S> & ::= & \mathbf{a}\ <A> \mid \mathbf{b} \mid <B> \\
<A> & ::= & <C>\ \mathbf{a} \mid <D>\ \mathbf{b} \\
<B> & ::= & <C>\ \mathbf{b} \mid <D>\ \mathbf{a} \\
<C> & ::= & <E> \\
<D> & ::= & <E> \\
<E> & ::= & \epsilon
\end{array}
$$

   Να υπολογίσετε τα σύνολα FIRST, FOLLOW και PREDICT για την παραπάνω γραμματική και να φτιάξετε τον πίνακα συντακτικής ανάλυσης για την παραπάνω γλώσσα (όπως στην εικόνα 2.19 του βιβλίου του Scott). Είναι η γλώσσα LL(1)? Δικαιολογήστε την απάντησή σας.

2. Θεωρείστε την παρακάτω LL(1) γραμματική:

$$
\begin{array}{lcl}
<A> & ::= & \mathbf{y}\ <B> \mid \mathbf{x} \mid <B>\ <C> \\
<B> & ::= & \mathbf{z}\ <B> \mid \mathbf{u} \\
<C> & ::= & \mathbf{s}
\end{array}
$$

   Τί είναι η *σύγκρουση ολίσθησης-ελάττωσης* (shift-reduce conflict); Πώς επιλύεται στα διάφορα είδη συντακτικών αναλυτών της οικογένειας LR; Να κατασκευάσετε τη *χαρακτηριστική μηχανή πεπερασμένης κατάστασης* (όπως στην εικόνα 2.25 του Scott) καθώς και τον πίνακα συντακτικής ανάλυσης SLR(1) (όπως στην εικόνα 2.27 του Scott). Είναι η παραπάνω γραμματική LR(0)? Είναι SLR(1)?

3. (**bonus**) Να κατασκευάσετε ένα παράδειγμα μιας γλώσσας που είναι LL(1), αλλά όχι SLR(1), ούτε LALR(1). Να κατασκευάσετε ένα παράδειγμα μίας γλώσσας που είναι SLR(1) αλλά όχι LL(1). Εξηγήστε το σκεπτικό σας.

2 Υλοποίηση: Parser της γλώσσας Simon

Η γλώσσα που θα της οποίας τον parser θα υλοποιήσετε είναι μια απλή φανταστική αντικειμενοστρεφής γλώσσα προγραμματισμού με το όνομα **Simon** η οποία περιγράφεται αμέσως μετά.

2.1 Δομή προγραμμάτων Simon

Κάθε πρόγραμμα Simon αποτελείται από ΜΙΑ ή περισσότερες classes. Κάθε class έχει την πιο κάτω μορφή:

```
class classname {
/* C-like Sxolia se
 opoiodhpote shmeio... */
<variable-declaration>
<constructor>
<method-declaration>
}
```

2.1.1 Αναγνωριστικά

Τα αναγνωριστικά (identifiers) της γλώσσας Simon απαρτίζονται από γράμματα (A..Z, a..z), ψηφία (0..9) και τον ειδικό χαρακτήρα _ (underscore). Ένα αναγνωριστικό δεν μπορεί να αρχίζει από ψηφίο. Τα πεζά γράμματα διαφέρουν από τα αντίστοιχα κεφαλαία. Τα παρακάτω αναγνωριστικά είναι δεσμευμένα και δεν επιτρέπεται να χρησιμοποιούνται ως κοινά αναγνωριστικά. Η σημασία τους εξηγείται σε επόμενες παραγράφους.
char, else, if, integer, class, new, return, void, while

2.1.2 Βασικοί Τύποι Δεδομένων

Οι βασικοί τύποι δεδομένων που υποστηρίζει η γλώσσα Simon είναι οι εξής: **integer, char**. Οι ακέραιες σταθερές αποτελούνται από ένα προαιρετικό πρόσημο ακολουθούμενο από ένα ή περισσότερα δεκαδικά ψηφία. Οι σταθερές τύπου χαρακτήρα αποτελούνται από έναν απλό ή ειδικό χαρακτήρα που περικλείεται από απλά εισαγωγικά. Ως απλοί χαρακτήρες θεωρούνται όλοι οι εκτυπώσιμοι χαρακτήρες πλην των εισαγωγικών (απλών και διπλών) και της ανάστροφης καθέτου \(backslash). Οι ειδικοί χαρακτήρες χρησιμοποιούνται όπως στη γλώσσα C. Παριστάνονται ως ζεύγη χαρακτήρων, με πρώτο την ανάστροφη κάθετο. Οι επιτρεπόμενοι ειδικοί χαρακτήρες είναι οι εξής:

Χαρακτήρας	Περιγραφή
\n	χαρακτήρας νέας γραμμής (new line)
\”	χαρακτήρας ” (διπλό εισαγωγικό)
\’	χαρακτήρας ’ (απλό εισαγωγικό)
\0	χαρακτήρας με ASCII κωδικό 0
\t	χαρακτήρας αλλαγής στήλης (TAB)
****	χαρακτήρας (backslash)

2.1.3 Δήλωση Μεταβλητών

Η δήλωση των Μεταβλητών, ακολουθεί τον τρόπο που δηλώνονται οι μεταβλητές στην Java ενώ οι επιτρεπόμενοι τύποι για τις απλές μεταβλητές είναι αυτοί που αναφέραμε στην προηγούμενη παράγραφο.

Εκτός από τους βασικούς τύπους δεδομένων, η γλώσσα Simon υποστηρίζει πίνακες (arrays) αποτελούμενους από στοιχεία των βασικών τύπων. Κατά τον ορισμό μιας μεταβλητής τύπου

πίνακα, πρέπει να ορίζεται το μήκος του, δηλαδή το πλήθος των στοιχείων που τον αποτελούν. Ένας πίνακας π.χ. 20 ακεραίων με όνομα anArray θα ορίζεται ως εξής:

```
anArray = new integer[20];  /* create an array of integers */
```

2.2 Μέθοδοι και Constructor

Ο constructor, όπως και οι επιπλέον μέθοδοι, ακολουθούν τον τρόπο ορισμού που ισχύει στην java. Να τον τεκμηριώσετε και να φαίνεται ξεκάθαρα στον ορισμό του BNF που θα δώσετε. Στο σώμα οποιασδήποτε μεθόδου υπάρχει αρχικά η δήλωση Μεταβλητών (όμοια με παραπάνω) και στη συνέχεια οι εντολές. Η εντολή return χρησιμοποιείται για την επιστροφή από τη μέθοδο. Όταν πρόκειται για μέθοδο που επιστρέφει κάποιο τύπο, το return ακολουθείται από έκφραση.

2.2.1 Εντολές

Οι εντολές της γλώσσας Simon έχουν ακριβώς την ίδια μορφή με τις αντίστοιχες εντολές της γλώσσας C. Η εντολή εκχώρησης έχει την ίδια σύνταξη με την αντίστοιχη εντολή της C. Επιτρέπεται μόνο η εκχώρηση απλών τύπων: απαγορεύεται η εκχώρηση ολόκληρων πινάκων, αλλά επιτρέπεται η εκχώρηση σε στοιχεία πίνακα.

Οι εντολές ελέγχου που επιτρέπει η γλώσσα είναι οι εξής :

```
if ( condition ) statement [ else statement ]
while ( condition ) statement
```

Οι τελεστές της γλώσσας Simon δίνονται παρακάτω. Η προτεραιότητα και η προσεταιριστικότητα των τελεστών αυτών είναι η ίδια με αυτή των τελεστών της γλώσσας C.

Αριθμητικοί Τελεστές Οι αριθμητικοί τελεστές με ένα τελούμενο της γλώσσας Simon είναι οι εξής:

Τελεστής	Σημασία
μοναδιαίοι	
+	Θετικό πρόσημο (ταυτότητα)
-	Αρνητικό πρόσημο (αντίθετος)
Δυαδικοί	
+	Πρόσθεση ακεραίων
-	Αφαίρεση ακεραίων
*	Πολλαπλασιασμός ακεραίων
/	Πηλίκο ακέραιας διαίρεσης
%	Υπόλοιπο ακέραιας διαίρεσης

Τα τελούμενα των αριθμητικών τελεστών πρέπει να είναι έγκυρες ακέραιες εκφράσεις.

Σχεσιακοί Τελεστές Οι σχεσιακοί τελεστές της γλώσσας Simon φαίνονται στον παρακάτω πίνακα. Τα τελούμενά τους πρέπει να είναι έγκυρες ακέραιες εκφράσεις ή χαρακτήρες.

Τελεστής	Σημασία
==	ίσο
!=	διάφορο
>	μεγαλύτερο
<	μικρότερο
>=	μεγαλύτερο ή ίσο
<=	μικρότερο ή ίσο

Τα αποτελέσματα των σχεσιακών τελεστών είναι λογικές εκφράσεις. Οι εκφράσεις αυτές μπορούν να χρησιμοποιηθούν μόνο σε εντολές if και while. Δεν μπορούν να εκχωρηθούν σε μεταβλητές, να περάσουν ως παράμετροι ούτε να επιστραφούν ως αποτελέσματα συναρτήσεων.

Λογικοί Τελεστές Οι λογικοί τελεστές της γλώσσας Simon είναι τρεις:

Τελεστής	Σημασία
||	Λογική διάζευξη (ή)
&&	Λογική σύζευξη (και)
!	Λογική άρνηση (όχι)

Οι τελεστές || και && είναι δυαδικοί, ενώ ο τελεστής ! δέχεται ένα τελούμενο. Τα τελούμενα πρέπει να είναι έγκυρες λογικές εκφράσεις. Το αποτέλεσμα είναι επίσης λογική έκφραση.

Ερωτήματα

1. (80%) Δώστε σε BNF τον συντακτικό ορισμό της γλώσσας, και χρησιμοποιώντας τα μεταεργαλεία Flex και Bison, υλοποιήστε έναν λεκτικό και συντακτικό αναλυτή, ο οποίος θα παίρνει ως είσοδο ένα αρχείο γραμμένο στη γλώσσα Simon που περιγράφηκε πιο πάνω και θα ελέγχει αν το πρόγραμμα είναι συντακτικά ορθό. Το πρόγραμμά σας θα καλείται από τη γραμμή εντολών ως εξής:

   ```
   prompt> myParser.exe  file.txt
   ```

   και θα επιστρέφει διαγνωστικό μήνυμα για το αν ήταν ορθώς γραμμένο, ή κατάλληλο μήνυμα σφάλματος (πρέπει να φαίνεται η γραμμή όπου υπάρχει το σφάλμα).

2. (10%) Τροποποιήστε τον κώδικά σας ώστε ο μεταγλωττιστής της γλώσσας Simon να επιτρέπει οδηγία **#include**. Η οδηγία αυτή θα επιτρέπει την ανάγνωση ενός εξωτερικού αρχείου σαν αυτό να ήταν τμήμα του προγράμματος, ενώ πρέπει να βρίσκεται υποχρεωτικά στην αρχή του προγράμματος εισόδου (να μην προηγούνται κενά διαστήματα). Η σύνταξή της είναι η εξής:

   ```
   #include "filename"
   ```

   Η οδηγία αυτή απευθύνεται ουσιαστικά στο λεκτικό αναλυτή. Σε περίπτωση που συναντήσει #include, θα πρέπει να σταματήσει την ανάγνωση του αρχείου προγράμματος, και να συνεχίσει με την επεξεργασία του αρχείου που ζητείται να συμπεριληφθεί. Μετά το τέλος αυτού του αρχείου, ο λεκτικός αναλυτής πρέπει να συνεχίσει από το σημείο του αρχείου προγράμματος, στο οποίο είχε σταματήσει. Φυσικά μεμονωμένες λεκτικές μονάδες καθώς και σχόλια πρέπει να περιέχονται πλήρως σε ένα αρχείο προγράμματος (δεν επιτρέπεται να αρχίζουν σε ένα αρχείο προγράμματος και να τελειώνουν σε κάποιο άλλο).

3. (10%) Τροποποιήστε τον κώδικα του συντακτικού αναλυτή σας ώστε να μπορεί να κάνει μία εκτίμηση του πλήθους των λαθών που υπάρχουν στο πρόγραμμα.

Παρατηρήσεις - Διαδικαστικά

Για τη χρήση των εργαλείων Flex και Bison μπορείτε να βρείτε πληροφορίες στη σελίδα του μαθήματος. Για την άσκηση μπορείτε να δουλέψετε σε ομάδες έως 4 ατόμων. Η βαθμολογία της άσκησης προκύπτει μετά από **ατομική προφορική εξέταση** που αφορά τόσο τις λεπτομέρειες της υλοποίησης όσο και την ύλη που καλύπτεται από το θεωρητικό τμήμα της άσκησης. Ως ημερομηνία παράδοσης της άσκησης ορίζεται η ημερομηνία γραπτής εξέτασης περιόδου Ιουνίου και Σεπτεμβρίου αντίστοιχα.

Παραδοτέα

- Γραπτή Αναφορά σε **pdf** που περιλαμβάνει:
  - Τις αναλυτικές λύσεις του θεωρητικού τμήματος μαζί με τις απαραίτητες επεξηγήσεις και τεκμηριώσεις όπου αυτό είναι απαραίτητο.
  - Τα αρχεία περιγραφής της γλώσσας, τα οποία δίνονται ως είσοδος στα μεταεργαλεία Flex και Bison.
  - Screenshots παραδειγμάτων εφαρμογής του parser.
- Ένα αρχείο **zip**, **rar**, **tar.gz** που περιλαμβάνει:
  - Την αναφορά σε ηλεκτρονική μορφή
  - Όλα τα αρχεία που αφορούν την υλοποίηση (συμπεριλαμβανομένων των αρχείων που δόθηκαν σαν είσοδο στον parser για να ελεγχθεί η σωστή λειτουργία του).

Το αρχείο zip (ή tar.gz) πρέπει να έχει όνομα τους αριθμούς μητρώου των ατόμων της ομάδας διαχωρισμένους με το χαρακτήρα " _ " , και διατεταγμένους από τον μικρότερο στο μεγαλύτερο (π.χ. 3500_3543_4788_4972.zip), και να σταλεί (ΥΠΟΧΡΕΩΤΙΚΑ) με email στο nikolako@... με θέμα "**ASKISI ARXES GLWSSWN 2013**". Στο σώμα του email θα πρέπει να αναφέρονται τα **ονοματεπώνυμα**, το **έτος** και οι αντίστοιχοι **αριθμοί μητρώου** των μελών της ομάδας.

Για τυχόν απορίες σχετικά με την άσκηση μπορείτε να χρησιμοποιείτε το forum του μαθήματος ή να έρχεστε στο γραφείο του Θάνου Νικολακόπουλου στα ΠΡΟΚΑΤ (δίπλα στο γραφείο του κ.Τσακαλίδη) κατά τις ώρες γραφείου (Δευτέρες 10:00-12:00).