Flex (strings, errors, handlers, hash tables)

Μεταγλωττιστές,

Χειμερινό εξάμηνο 2018-2019

Παραδοτέα λεκτικής ανάλυσης

- Χειρόγραφο μέρος:
  - σχεδιάσμος του Διαγράμματος Μετάβασης (ΔΜ) της γλώσσας της εργασίας (JFLAP ή χειρόγραφα)
  - τοποθετήστε πρώτα τις λέξεις-κλειδιά, στη συνέχεια προσθέστε τις σταθερές και τα αναγνωριστικά, και τέλος συμπληρώστε τους τελεστές και τα διαχωριστικά σύμβολα
  - κατασκευη ενός πίνακα με όλες τις τελικές καταστάσεις του ΔΜ, όπου να σημειώσετε τις ΛΜ που επιστρέφονται
  - Χρήση οπισθοδρόμησης όπου πρέπει να γίνεται σε κάθε τελική κατάσταση

Παραδοτέα λεκτικής ανάλυσης

- Προγραμματιστικό μέρος (υλοποίηση του ΛΑ):
  - Να επιστρέφει τον κωδικό της ΛΜ που βρίσκει κάθε φορά που καλείται
  - Να επιστρέφει τα ονόματα των αναγνωριστικών που συναντά, ή εναλλακτικά να εισάγει στον Πίνακα Συμβόλων (ΠΣ) τα αναγνωριστικά που συναντά
  - Να επιστρέφει τις τιμές των σταθερών που συναντά. Για μη αριθμητικές σταθερές να επιστρέφει τη συμβολοσειρά της λέξης που αναγνώρισε
  - Να χειρίζεται σφάλματα λεκτικής ανάλυσης με εκτύπωση μηνύματος που να περιλαμβάνει τον αριθμό γραμμής και την ίδια τη γραμμή του αρχείου εισόδου στην οποία σημειώθηκε το σφάλμα.

Strings

- Μπορούν να ορισθούν με μια αποκλειστική κατάσταση
  - %x STRINGS

- Όταν βρεθεί η τελική κατάσταση πρέπει να προστεθεί το ‘\0’ στο buffer που αποθηκεύεται το string
  - <STRING>\“ {*string_buf = '\0'; BEGIN(INITIAL); return T_STRING;}

- Προσοχή στην διαφορά \n της νέας γραμμής με τους χαρακτήρες
  - <STRINGS>\n {yyerror("Illegal input in string");}
  - <STRINGS>\\n {*string_buf++ = '\n';}

Χειρισμός λαθών

- Χρήση της yyerror(), και δήλωση στο πρώτο τμήμα ορισμών του flex:
  - void yyerror(char *message);
- Κλήση της συνάρτησης στο τμήμα κανόνων με κατάλληλο μήνυμα:
  - yyerror(“illegal input in string");
- Χρήση της yyterminate() σε περίπτωση ολοκλήρωσης της λεκτικής ανάλυσης

yyerror(char *message)

- Υλοποίηση της συνάρτησης στο τμήμα συναρτήσεων χρήστη.
- Μπορεί να μετράει τα λάθη και π.χ. να σταμάταει η λεκτική ανάλυση όταν αυτά ξεπεράσουν τα 5.
  - Χρήση πάλι μιας global variable int errors; στο τμήμα ορισμών

```
void yyerror(char *message)
{
error_count++;
printf("%d errors found with %s at token %s in line %d", errors, message, yytext, lineno);
if(MAX_ERRORS == error_count)
{
printf(“max errors detected");
exit(-1);
}}
```

Εντοπισμός λάθων

- Εντοπίζονται κυρίως με τον τελευταίο κανόνα → .
  - . {yyerror("Illegal character");}

- Χρήση της void yyless(int n) ώστε ο λεκτικός αναλυτής να προσπαθεί να συνεχίσει σβήνοντας το λανθασμένο χαρακτήρα
  - επιστρέφει όλους εκτός από τους πρώτους n χαρακτήρες πίσω στην ροή εισόδου.
  - π.χ. “foobar” σαν input η yyless(3) θα επιστρέψει στο το “bar” πίσω στην ροή εισόδου

Εντοπισμός λάθων

- Λεκτικά λάθη μπορούν να υπάρξουν και σε αποκλειστικές καταστάσεις
  - %x STRING, %x COMMENTS etc...

- Καλύτερη επιλογή είναι ο τερματισμός του προγράμματος, καθώς δεν υπάρχει νόημα να σβηστεί κάποιος χαρακτήρας
  - Το πιο συχνό λάθος είναι πρώιμο EOF που δεν μπορεί να αντιμετωπιστεί

- Χρήση της yyerror() με κατάλληλο μήνυμα και ακολούθως της yyterminate
  - <COMMENTS><<EOF>>{yyerror("Unterminated comment"); yyterminate();}
  - <STRINGS><<EOF>> {yyerror("Unterminated string"); yyterminate();}

Χειρισμός αριθμών

- Όλα οι ΛΜ επιστρέφονται σαν tokens μέσω της yytext ☹
- Εύκολη μετατροπή από το δεκαδικό μέσω των atoi, atof
- Ορισμός συναρτήσεων για τον χειρισμό αριθμών
  - int dectoInt(char *str); double dectoReal(char *str);
  - int hextoInt(char *str); double hextoReal(char *str);
  - int bintoInt(char *str); double bintoReal(char *str);

Ακέραιοι αριθμοί

- Σε οποιαδήποτε βάση εύκολη μετατρόπη ακεραίων μέσω της strtol
  - int strtol (const char *str, char **endptr, int base)

- Μετατρέπει το αρχικό τμήμα της συμβολοσειράς str σε μια τιμή int σύμφωνα με τη δεδομένη βάση, η οποία πρέπει να είναι μεταξύ 2 και 36

```
int hexIntHandle(char *str)
{
        int res;
        res = (int) strtol(&str[2], NULL, 16);
        return res;
}
```

Πραγματικοί αριθμοί

- Ειδικοί μεταχείριση του αριθμού με την εύρεση του ακεραίου και δεκαδικού μέρους

- Χρήση των βοηθητικών συναρτήσεων strlen και strchr
  - size_t strlen (const char * str) υπολογίζει το μήκος του str
  - char *strchr(const char *str, int c) αναζητά την πρώτη εμφάνιση του χαρακτήρα c και επιστρέφει έναν δείκτη στο str που δείχνει την πρώτη εμφάνιση του χαρακτήρα c

Πραγματικοί αριθμοί

- Συνεπώς υπολογισμός του πραγμάτικου κομματιού και του δεκαδικού σε δύο στάδια.
  - υπολογισμός του μήκος των επιμέρους συμοβολοσειρών

```
length_str = strlen(str);
character = strchr(str, '.');
length_real = strlen(character) - 1;
length_decimal = length_str - length_real + 1;
```

- Ο ακέραιος θα υπολογιστεί με την strtol

```
res = (int) strtol( &str[2], &character, 16);
```

Πραγματικοί αριθμοί

- Πρόσθεση του δεκαδικού αριθμού στον ακέραιο με αλγόριμο βάσης για χειρισμό δεκαδικών
  - Παράδειγμα double bintoReal(char *str);

```
double binRealHandle(char *str){
double res; int i; int length_real;
char *character;
character = strchr(str, '.');
length_real = strlen(character) - 1;
res = (int)strtol(&str[2], &character, 2);
for (i = 1; i <= length_real; i++)
{
	res = (character[i] - '0') * (1 / pow(2,i)) + res;
}
return res;}
```

Πίνακας συμβόλων

- Για την λεκτική ανάλυση χρειάζονται μόνο τα αναγνωριστικά (ID).

- include το header file στο πρώτο τμήμα και δήλωση του
  - #include ‘’hashtbl.h’’
  - HASHTBL *hashtbl;

- Δημιουργία και διαγραφή hast table στην main
  - hashtbl = hashtbl_create(10, NULL)
  - hashtbl_destroy(hashtbl);

Πίνακας συμβόλων

- Στο τμήμα κανόνων όταν βρεθεί αναγνωριστικό εισάγετε πληροφορία στον πίνακα συμβόλων:
  - hashtbl_insert(hashtbl, yytext, NULL ,0);

- Υπάρχει η δυνατότητα εκτύπωσης του πίνακα συμβόλων:
  - hashtbl_get(hashtbl, 0);

ΕΡΩΤΗΣΕΙΣ/ΑΠΟΡΙΕΣ