Εισαγωγή στο Flex

Μεταγλωττιστές,

Χειμερινό εξάμηνο 2017-2018

Λεκτική Ανάλυση

- Αποτελεί την πρώτη φάση της μετάφρασης.
- Αναγνώριση λεκτικών μονάδων ενός προγράμματος.
- Απόκρυψη των περιττών λεπτομερειών από τον συντακτικό αναλυτή.

- Που βοηθάει το Flex ?
  - Εργαλείο κατασκευής λεκτικού αναλυτή.
  - Αναγνωρίζει λεκτικές μονάδες από ένα αρχείο.
  - Δυνατότητα επιστροφής τιμών για κάθε λεκτική μονάδα.
  - Κατασκευάζει αυτόματα το ΔΜ.

Flex (“Fast”-Lexical Analyzer)

- Μετα-εργαλείο παραγωγής λεκτικών αναλυτών.
- Open source.
- https://github.com/westes/flex/tree/master/examples
- Linux OS
  - Download address: https://sourceforge.net/projects/flex/
- Windows OS
  - Διατίθεται μαζί με το περιβάλλον Cygwin
  - http://www.cygwin.com/

Εγκατάσταση Flex

- Αποσυμπίεση:
  - tar –zxvf flex-2.5.37.tar.gz

- Στον κατάλογο που δημιουργείται χρησιμοποιούμε:
  - ./configure
  - make
  - make install

- sudo apt-get install flex (from the command line)

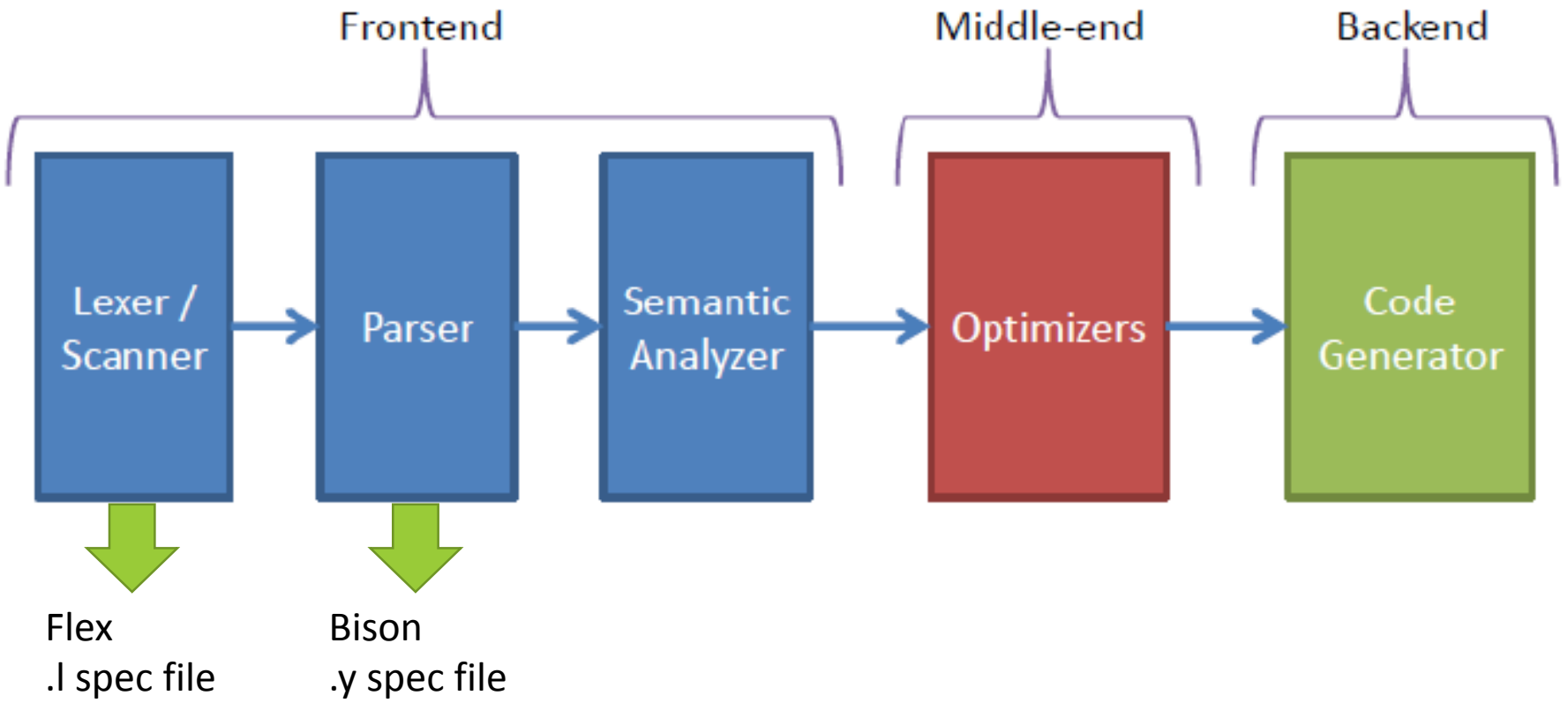
Παράδειγμα Λεκτικής Ανάλυσης

- Μετατροπής μιας έκφρασης από χαρακτήρες σε μια σειρά από tokens.

foo = 1-3**2

Lex	**Token type**
foo	Variable
=	Assignment Operator
1	Number
-	Subtraction operator
3	Number
**	Power operator
2	Number

FLEX/BISON στο flowchart ενός compiler



Λειτουργία Flex

- Το Flex και Bison λειτουργούν σχεδόν με τον ίδιο τρόπο.
- Καθορισμός των εκφράσεων προς αναγνώριση.
  - Σε αυτήν την περίπτωση χρησιμοποιούνται κανονικές εκφράσεις.
- Καθαρισμός ενεργειών μετά την αναγνώριση.

Δομή Αρχείου Εισόδου

- Το αρχείο εισόδου στο Flex αποτελείται από 3 μέρη.
  - Ορισμοί.
  - Κανόνες.
  - Συναρτήσεις.
- Δομή:

```
/*Statements block*/
%%
/*Rules block*/
%%
/*User Functions*/
```

Τμήμα Ορισμών

- Το τμήμα ορισμών περιλαμβάνει:
  - Επιλογές του εργαλείου:
    - % option noyywrap //tells Flex to read only one input file
    - %option case-insensitive (Αυτό σημαίνει ότι η λέξη “ClaSS” θα είναι ίδια με τη λέξη “class”)
  - Δηλώσεις αρχικών καταστάσεων.
    - %x STATE_COMMENT
  - Ορισμός αναγνωριστικών για Κ.Ε.
    - Ανάθεση ονομάτων σε σύνολα από αναγνωριστικά
    - Letterdigit [A-Za-z0-9]
    - Whitechar [\t\n]
  - Literal blocks – περικλείεται μέσα σε %{ code %} και περιέχει κώδικα C.
    - Header file inclusion.
    - Define constants and global variables.

Τμήμα Κανόνων

- Περιλαμβάνει τις κανονικές εκφράσεις και τους διάφορους κανόνες.
- Μορφή κανόνων:
  - Κανονική έκφραση {C code}
  - “0”|(“-”?[1-9][0-9]*)) {return (INTEGER);}
- Κώδικας εντός των %{..%} αντιγράφεται αυτούσιος.
- Η εντολή **return** επιστρέφει στο συντακτικό αναλυτή το αναγνωριστικό της τελευταίας λεκτικής μονάδας, που αναγνωρίστηκε.
- **ΠΡΟΣΟΧΗ!!!** Το Flex θα χρησιμοποιήσει το πρώτο κανόνα που κάνει match σε περίπτωση σύγκρουσης.(παραδείγματα παρακάτω)

Τμήμα Συναρτήσεων Χρήστη

- Περιλαμβάνει τις οριζόμενες από τον χρήστη συναρτήσεις.
- Παραδείγματα:
  - main()
  - yyerror()
  - yywrap()
- Ουσιαστικά δημιουργούμε όποια συνάρτηση θέλουμε εμείς για να επεξεργαστούμε τυχόν δεδομένα.

Ειδικοί Χαρακτήρες

Character	Operation
.	Οποιοσδήποτε χαρακτήρας εκτός του ‘\n’
[]	Κλάσεις χαρακτήρων, π.χ. [a-z] ,[0-9]
^	Αναγνώριση της αρχής μίας γραμμής
$	Αναγνώριση του τέλους μια γραμμής
{}	Πλήθος εμφανίσεων ή αναφορά σε ένα pattern
\	Escape sequence όπως στη C
*	Μηδέν ή περισσότερες εμφανίσεις της Κ.Ε. που προηγείται

Ειδικοί Χαρακτήρες (συνέχεια)

Character	Operation
+	Μία ή περισσότερες εμφανίσεις της Κ.Ε. που προηγείται
?	Μηδέν ή μία εμφάνιση της Κ.Ε. που προηγείται
|	Τελεστής διάζευξης
“…”	Αναγνώριση κυριολεκτικών
()	Ομαδοποίηση Κ.Ε.
/	Αναγνώριση βασισμένη σε συμφραζόμενα
<<EOF>>	Αναγνώριση τέλους αρχείου

Αναγνώριση Εκφράσεων

- Αναγνώριση της έκφρασης με το μεγαλύτερο μήκος συμβολοσειράς.
- Σε περίπτωση σύγκρουσης επιλέγεται ο προγενέστερος κανόνας.
- Απλά παραδείγματα:
  - “if” {return(IF);}
  - [a-zA-Z0-9]+ {return(ID);}

Αναγνώριση Εκφράσεων

- [0-9]+ { printf(“An integer %s \n”, yytext); }
- [a-z][a-z0-9]* { printf(“An identifier %s \n”, yytext);}
- if | then | begin | end | function { printf(“A Keyword %s \n”, yytext); }

Παραδείγματα Συνέχεια

- Rules
  - [a-zA-Z0-9]+ {printf(“Hello”);}
  - “post” {printf(“World”);}
- Input
  - post
- Output
  - Θα κάνει match με τον πρώτο κανόνα και άρα θα τυπώσει “Hello”

Παραδείγματα Συνέχεια (ειδικοί χαρακτήρες)

- [ab-e] → αναγνωρίζει τους χαρακτήρες a ή b ή c ή d ή e.
- [^xyz] → αναγνωρίζει **ΟΛΟΥΣ** τους χαρακτήρες, εκτός από x,y,z
- [ab-e]* → περιγράφει τα ε, a, c, ac, cab, bbbe, cbaead…
- [ab-e]+ → περιγράφει τα a, c, ac, cab, bbbe, cbaead…
- “abc”+ → αναγνωρίζει τις abc, abcabc, ...
- \\, \”, \(→ οι ίδιοι οι χαρακτήρες \, ”, (
- .{2, 3} → περιγράφει τις “λέξεις” που αποτελούνται από 2 ή 3 χαρακτήρες, πχ. ant, ///, ..., 3/2, 23, 2b, or, not,...
- [a-z]”foo”[1-3] → αναγνωρίζει όλες τις λέξεις 5 χαρακτήρων που ξεκινούν με πεζό γράμμα, περιέχουν το string“foo” και τελειώνουν με έναν αριθμό από το 1 μέχρι το 3, πχ. afoo1, lfoo3, ...

Παραδείγματα Συνέχεια (ειδικοί χαρακτήρες)

- (r) → (abc)+ αναγνωρίζει τις abc, abcabc, ...
- r|s → η (”abc” | “ABC”){2,3} αναγνωρίζει τις abcabc, abcABC, ABCABC, abcabcabc, ...
- ^r → ικανοποιείται όταν ικανοποιείται η r και η ακολουθία που την ικανοποιεί βρίσκεται στην αρχή της γραμμής
- r$ → ικανοποιείται όταν ικανοποιείται η r και η ακολουθία που την ικανοποιεί βρίσκεται στο τέλος της γραμμής
- . (τελεία) → αναγνωρίζει οποιονδήποτε χαρακτήρα ή σύμβολο πλην του newline

Προτεραιότητα Κανονικών Εκφράσεων

- Η προτεραιότητα είναι:
  - character, ., [ ], [^], *, +, ?, { }, \, ( ), |, ^, $

- • Η foo|bar* είναι ισοδύναμη με την (foo)|(ba(r*)), επειδή ο τελεστής '*' έχει μεγαλύτερη προτεραιότητα από την παράθεση και αυτή από το τελεστή '|‘ (alternation).
  - Ο τελεστής * έχει μεγαλυτέρη προτεραιότητα απο την ομαδοποίηση ().

Προκαθορισμένες κλάσεις χαρακτήρων

- Το flex υποστηρίζει κάποιες προκαθορισμένες κλάσεις χαρακτήρων που μπορούν να χρησιμοποιηθούν σε κανονικές εκφράσεις. Ο συμβολισμός των κλάσεων αυτών έχει τη μορφή [:X:], όπου 'X' το όνομα της κλάσης.
- Συγκεκριμένα οι χαρακτήρες της κάθε κλάσης, ικανοποιούν τις συναρτήσεις 'isX(int c)' της επικεφαλίδας “ctype.h”. Οι κυριότερες από αυτές είναι:
  - [:alnum:] - όλα τα αλφαριθμητικά: [a-zA-Z0-9] στο c (default) locale
  - [:alpha:] - όλα τα γράμματα του αλφαβήτου: [a-zA-Z] στο c locale
  - [:digit:] - όλα τα ψηφία του δεκαδικού συστήματος: [0-9]
  - [:lower:] - όλα τα πεζά γράμματα του αλφαβήτου: [a-z]
  - [:upper:] - όλα τα κεφαλαία γράμματα του αλφαβήτου: [A-Z]

Αρχικές Καταστάσεις

- Χρησιμοποιούνται για αναγνώριση με χρήση συμφραζομένων.
- Προσθήκη του <STATE_NAME> πριν από κάθε κανόνα της κατάστασης.
- Δήλωση των καταστάσεων που χρησιμοποιούμε, στο τμήμα ορισμών:
  - %x: Αποκλειστικές καταστάσεις
    - Ενεργοί είναι μόνο οι κανόνες που ανήκουν στην κατάσταση.
  - %s: Κοινόχρηστες καταστάσεις.
    - Ενεργοί είναι οι κανόνες που ανήκουν στην κατάσταση όσο και όσοι δεν ονοματίζουν κάποια κατάσταση.

Αρχικές Καταστάσεις

- Αρχική κατάσταση του αναλυτή είναι η INITIAL.
- Μετάβαση μεταξύ καταστάσεων:
  - BEGIN(STATE_NAME);
- Χρήση κατάστασης
  - <STATE_NAME>{…DERIVED_STATE_RULES….}

Παράδειγμα

- C-style comments:

```
“/*”                    {BEGIN(COMMENT);}
<COMMENT>”*/”           {BEGIN(INITTIAL);}
<COMMENT>\n             {lineno++;}
<COMMENT>.              ;
<COMMENT><<EOF>>        {printf(“Unterminated comment”);return 0;}
```

Χρήσιμες Μεταβλητές

- YYSTYPE yylval:
  - Είναι ίσως η σημαντικότερη μεταβλητή, καθώς χρησιμοποιείται για επικοινωνία του λεκτικού με τον συντακτικό αναλυτή.
  - Περιέχει την τιμή που θα περαστεί στον συντακτικό αναλυτή.
  - Μπορεί να περιέχει πολλαπλά πεδία αν ξαναοριστεί ως union τύπου YYSTYPE στο τμήμα ορισμών του αρχείου.
  - Parser (Bison) → έχει πρόσβαση στις yylval, yylen, και yytext.
- int yyleng:
  - Περιέχει το μήκος της αναγνωρισθείσας λεκτικής μονάδας.
- char *yytext:
  - Περιέχει την αναγνωρισθείσα λεκτική μονάδα.

Παράδειγμα yyleng

```
1  %%
2
3  {word}   { wordCount++; charCount += yyleng; }
4  {eol}    { charCount++; lineCount++; }
5  .        { charCount++; }
```

- Στην γραμμή 3, όταν αναγνωριστεί μια λέξη, η yyleng αυξάνει την τιμή του CharCount τόσο όσο είναι ο αριθμός των χαρακτήρων στη αναγνωρισμένη λέξη.

Χρήσιμες Συναρτήσεις

- int yylex():
  - Καλείται για την αναγνώριση της επόμενης λεκτικής μονάδας.
- void yymore():
  - Ενσωματώνει την επόμενη λεκτική μονάδα στην τρέχουσα.
- void yyless(int n):
  - Κρατάει τους n χαρακτήρες του λεκτικού και επιστρέφει τους υπόλοιπους.
- int yywrap():
  - Καθορίζει την λειτουργία του λεκτικού αναλυτή. Καλείται αυτόματα και μπορεί να επιστρέψει την αναγνώριση πολλαπλών αρχείων.

Χρήσιμες Συναρτήσεις (εκτός Flex)

- Συναρτήσεις μετατροπής:
  - Όταν αναγνωριστεί μία λεκτική μονάδα η yytext που την περιέχει είναι τύπου char. Για να επιστραφεί με χρήση της yyval χρειάζεται μετατροπή.
  - atof → char to float
  - atoi → char to int
  - Αν χρειαστούν άλλες βάσεις ??
    - Hint : int strtol(const char *str, char **endptr, int base)
- Debugging:
  - Χρησιμοποίηση της printf για να επιστρέψει τη αναγνωρίζει το Flex.

Μετρώντας Γραμμές με το Flex

```
%{
intlinecount= 0;
%}
Digit [0-9]
Identifier [a-zA-Z]{a-zA-Z0-9}*
%%
{Identifier}        { printf("%s: This is an identifier\n", yytext); }
\n                  { printf("The line number is %d\n", ++linecount); }
[\t ]+              ; /*Ignore spaces */
.                   { printf("Unrecognized character\n"); }
%%
main(){
yylex();}
```

Ορισμοί

Κανόνες

Συναρτήσεις Χρήστη

Εκτέλεση Flex

- Όνομα αρχείου → file_name.l
- Εντολή → flex file_name.l
  - Παράγεται το αρχείο lex.yy.c
  - Override με –o Output_File
- Compile του παραγόμενου αρχείου για παραγωγή εκτελέσιμου:
  - Με gcc
  - Με gcc και σύνδεση με την βιβλιοθήλη libfl: gcc lex.yy.c –o lexer_name -lfl
- Εκτέλεση λεκτικού αναλυτή:
  - ./lexer_name input_file_name

ΕΡΩΤΗΣΕΙΣ/ΑΠΟΡΙΕΣ