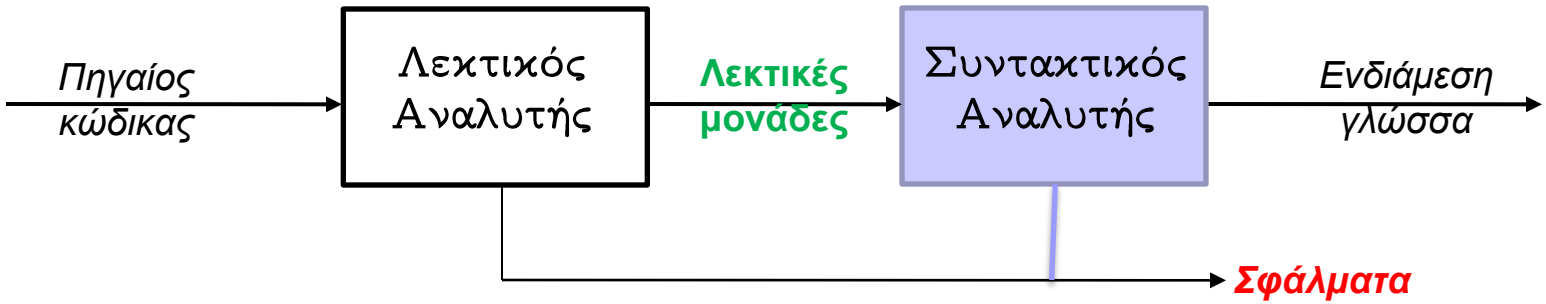
Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών 2010-2011

Compiler Tools:

Yacc/Bison

Compiler Front End



Parser (Συντακτικός Αναλυτής)

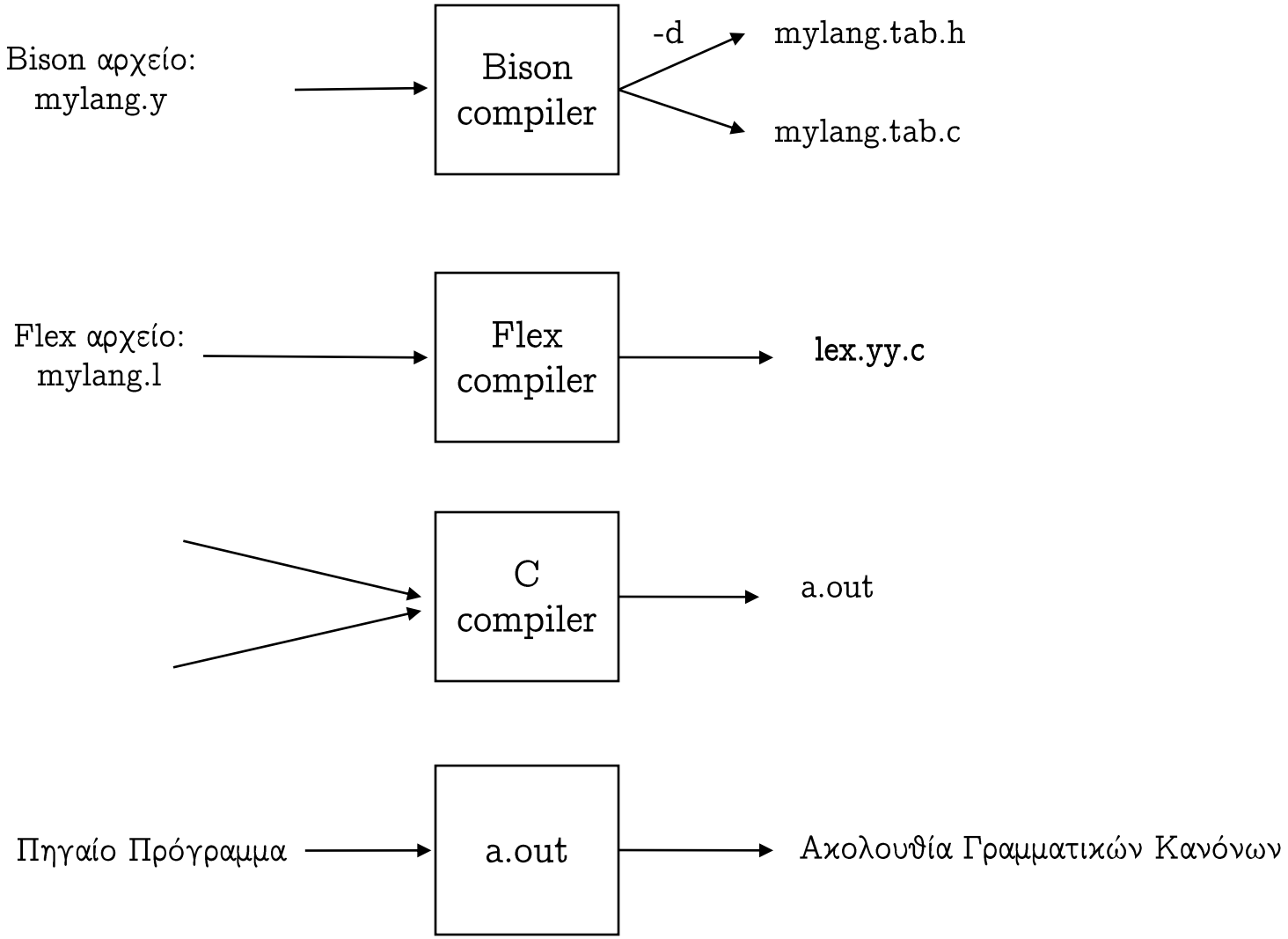
- Ελέγχει τη ακολουθία των λεκτικών μονάδων *(μέρη του λόγου)* για γραμματική ορθότητα
- Καθοδηγεί τον έλεγχο σε βαθύτερα επίπεδα από τη συντακτική ανάλυση
- Χτίζει ουσιαστικά μία IR αναπαράσταση του κώδικα

Η συντακτική ανάλυση είναι υπολογιστικά δυσκολότερη από την λεκτική. Είναι προτιμότερο να συμπεριλαμβάνουμε όσο περισσότερους κανόνες γίνεται στον λεκτικό αναλυτή (whitespace κλπ)

Τι είναι ο Bison?

- Ο bison είναι μια βελτίωση του εργαλείου yacc του Unix.
- Ο yacc είναι μια γεννήτρια συντακτικών αναλυτών:
  - δέχεται μια γραμματική χωρίς συμφραζόμενα (LALR(1)) και παράγει έναν συντακτικό αναλυτή σε C
  - Η παραγόμενη συνάρτηση yyparse αναγνωρίζει τις συμβολοσειρές εισόδου, κατασκευάζει το συντακτικό δέντρο και εκτελεί τις ενέργειες που περιγράφονται στο πρόγραμμα

Διαδικασία Παραγωγής Συντακτικού Αναλυτή



Δομή Προγράμματος Bison

%{

Κώδικας C

(μακροεντολές, τύποι δεδομένων, δηλώσεις μεταβλητών και συναρτήσεων)

%}

Δηλώσεις Bison

%%

Κανόνες παραγωγής γραμματικής

%%

Κώδικας C

(υλοποίηση συναρτήσεων, main())

Κανόνες Παραγωγής Γραμματικής(1/2)

- Η περιγραφή της γραμματικής της γλώσσας γίνεται με κανόνες παραγωγής διατυπωμένους σε BNF μορφή
- Γενική μορφή κανόνων: αριστερό_μέλος: δεξιό_μέλος;
- Το αριστερό_μέλος είναι ένα μη τερματικό σύμβολο
- Το δεξιό_μέλος μπορεί να περιέχει μηδέν ή περισσότερα τερματικά και μη τερματικά σύμβολα και εντολές C σε {}
  - Program : {count=0;} block_list
    {printf(" Counted %d block(s) \n",count);}
    ;
    block_list : /* nothing */
    | block_list block {count++;}
    ;
    block : BEGIN block_list END
    ;

Κανόνες Παραγωγής Γραμματικής(2/2)

Παράδειγμα με **actions**:

```
statement: NAME '=' expression
         | expression { printf("= %d\n", $1); }
         ;
$$       $1     $2   $3
expression: NUMBER '*' NUMBER    { $$ = $1 * $3 }
         | NUMBER                { $$ = $1 }
         ;
```

- $1, $3 αναφέρονται σε τιμές του δεξιού μέλους. Το $$ θέτει τιμή στο αριστερό μέλος. Η έκφραση, $$ = $1 * $3 θέτει την τιμή της (expression) σε NUMBER ($1) * NUMBER ($3)
- Το action ενός κανόνα εκτελείται όταν ο parser *reduces* αυτόν τον κανόνα
- Ο λεκτικός αναλυτής θα πρέπει να έχει επιστρέψει μία τιμή μέσω της yylval

Δηλώσεις Bison(1/3)

- Τμήμα Ορισμών
  - Τα Tokens της γραμματικής θα πρέπει να οριστούν. Παράδειγμα:
    - expression: NUMBER '+' NUMBER { $$ = $1 + $3; }
    - Το token NUMBER θα πρέπει να οριστεί:
    - %token NUMBER
  - Από τα ορισμένα tokens ο Bison θα δημιουργήσει ένα κατάλληλο header file
  - Χαρακτήρες σε απλά εισαγωγικά μπορούν να χρησιμοποιηθούν σαν tokens χωρίς να δηλωθούν, π.χ., '+', '=' κλπ.

Δηλώσεις Bison(2/3)

- Το header file που παράγει ο bison πρέπει να συμπεριληφθεί στο αρχείο του flex.
- Θα πρέπει να δηλώσετε το yylval ως extern

```
%{
#include "simpleCalc.tab.h
extern int yylval;
#include <math.h>
%}
```

Δηλώσεις Bison(3/3)

- Δηλώσεις τελεστών της γλώσσας και της *προτεραιότητας* και *προσεταιριστικότητας* τους

  %nonassoc '=' '<' '>'

  %left '+' '-'

  %left '*' '/' TK_div, TK_mod

- Ισχύουν τα εξής:
  - τα tokens που εμφανίζονται στην ίδια γραμμή έχουν την ίδια προτεραιότητα
  - η προτεραιότητα αυξάνεται από πάνω προς τα κάτω
    - Τα '+' και '-' έχουν μικρότερη προτεραιότητα από τα '*' και '/'
    - Η έκφραση a+b+c υπολογίζεται ως (a+b)+c
- Το %nonassoc χρησιμοποιείται για τελεστές που δεν μπορούν να συνδυαστούν μεταξύ τους, π.χ., το '<'
- Χρησιμοποιούνται για αποφυγή της παραγωγής μιας έκφρασης με περισσότερους του ενός τρόπους

Τμήμα Κώδικα C

- Συναρτήσεις yyerror , main και άλλες....

```
yyerror(char *errmsg)
{
    fprintf(stderr, "%s\n", errmsg);
}

main()
{
   yyparse();
}
```

Το περιβάλλον εκτέλεσης του Bison

- int yyparse(); → Η συνάρτηση αυτή υλοποιεί τον Συντ.Αναλυτή. Επιστρέφει 0 αν αναγνωρισθεί η συμβολοσειρά εισόδου ή την τιμή 1 σε περίπτωση συντακτικού λάθους.
- int yylex(); → Υλοποιεί τον λεκτικό αναλυτή. Καλείται από την yyparse κάθε φορά που πρέπει να διαβαστεί μια νέα λεκτική μονάδα από τη συμβολοσειρά εισόδου.
- void yyerror(const char * message); → Η συνάρτηση αυτή πρέπει να υλοποιείται υποχρεωτικά στο μεταπρόγραμμα του bison. Καλείται αυτόματα όταν εντοπιστεί κάποιο συντακτικό σφάλμα.
- Περισσότερα στο εγχειρίδιο του bison...

Παράδειγμα συνεργασίας Flex-Bison

- Θέλουμε να γράψουμε ένα μεταφραστή για μια απλή γλώσσα αριθμητικών εκφράσεων. Η είσοδος θα δίνεται από ένα αρχείο input και τα αποτελέσματα θα γράφονται σε ένα αρχείο output.

- Πχ, θα διαβάζει: 7+5*8 και θα γράφει στο Output 47.

Βήμα 1: σχεδιάζουμε το bnf της γλώσσας.

- Στην περίπτωσή μας θεωρούμε πως κάθε πρόγραμμα της γλώσσας πρέπει να αποτελείται από μόνο μία αριθμητική έκφραση, ενώ οι μόνες επιτρεπόμενες πράξεις είναι: '+', '*' μεταξύ ακεραίων.
- <πρόγραμμα> ::= <έκφραση>
  <έκφραση> ::= ΑΚΕΡΑΙΟΣ
  | <έκφραση> + <έκφραση>
  | <έκφραση> * <έκφραση>

Βήμα 2: μεταφέρουμε το bnf στον bison.

- program: expr { fprintf(yyout, "%i\n", $1); }
  ;
  expr: INT
  | expr '+' expr { $$ = $1 + $3; }
  | expr '*' expr { $$ = $1 * $3; }
  ;

- Ωστόσο η γλώσσα μας δεν είναι LL(1) . Συνεπώς θα πρέπει είτε να τροποποιήσουμε τη γραμματική ώστε να αποφύγουμε την αριστερή αναδρομή, είτε να ορίσουμε προτεραιότητες στους τελεστές ώστε να γνωρίζει ο bison πώς να αναλύσει τη συμβολοσειρά εισόδου.
- ΠΡΟΣΟΧΗ: πολλές φορές η αλλαγή της γραματικής είναι μονόδρομος (πχ αριστερή παραγοντοποίηση,...)

Βήμα 3 : Δηλώσεις & Προτεραιότητες

- %token INT
  %left '+'
  %left '*'

- Προσοχή το %left '*' τοποθετείται δεύτερο διότι ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα από την πρόσθεση.

Βήμα 4: Δημιουργία λεκτικού αναλυτή(1/2).

- Θα πρέπει έπειτα να γράψουμε έναν λεκτικό αναλυτή που να μετατρέπει τη συμβολοσειρά εισόδου σε μία ακολουθία από token. Οπότε έχουμε:

Στο προηγούμενο μάθημα απλά επιστρέφαμε tokens, όχι τιμές

- ```
  digit    [0-9]
  num      {digit}+
  %%
  {num}          { yylval = atoi(yytext); return INT; }
  "+"{ return '+'; }
  "*" { return '*'; }
  "\n"     { return '\n'; }
  .    ;
  %%
  ```

Θέτει τιμή για χρήση στα actions

Επιστρέφει το token που αναγνώρισε
  ```

Βήμα 4: Δημιουργία λεκτικού αναλυτή(2/2).

- Επίσης θα πρέπει να κάνουμε συμπερίληψη το header file που δημιούργησε ο bison και άλλες χρήσιμες βιβλιοθήκες. Οπότε:

- ```
  %{
  #include "y.tab.h"
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  %}
  ```
  ```

Βήμα 5: main,yyerror συναρτήσεις(1/2)

- Στο αρχείο του bison θα πρέπει επίσης να ορισθούν οι συναρτήσεις main και yyerror
- Επίσης θα πρέπει να ορίσουμε τις μεταβλητές που χειρίζονται τα αρχεία εισόδου/εξόδου

- ```
  %{
  #include <stdio.h>
  #include <math.h>
  void yyerror(char *);
  extern FILE *yyin;
  extern FILE *yyout;
  %}
  ```
  ```

Βήμα 5: main,yyerror συναρτήσεις(2/2)

- ```
  void yyerror(char *s) {
    fprintf(stderr, "%s\n", s);
  }
  ```
- ```
  int main ( int argc, char **argv  )   {
    ++argv; --argc;
    if ( argc > 0 )
          yyin = fopen( argv[0], "r" );
    else
          yyin = stdin;
    yyout = fopen ( "output", "w" );
    yyparse ();
    return 0;
  }
  ```

Ανάνηψη από σφάλματα

- Διαδικασία κατά την οποία ανιχνεύεται ένα συντακτικό σφάλμα και συνεχίζεται η ανάλυση για την ανίχνευση περισσότερων λαθών.

- Τεχνικές ανάνηψης:
  - Error Recovery
  - Error Repair
  - Error Correction

Error Recovery

1. Ανίχνευση συντακτικού λάθους.
2. Αγνόηση μέρους της εισόδου.
3. Αρχικοποίηση περιβάλλοντος αναλυτή (στοίβα, μεταβλητές) για την ανάλυση της υπόλοιπης συμβολοσειράς εισόδου.

- **ΠΡΟΣΟΧΗ**: Μπορεί να οδηγήσει σε ανίχνευση επόμενων λαθών που δεν είναι στην πραγματικότητα λάθη
- Οι φάσεις της σημασιολογικής ανάλυσης και της παραγωγής ενδιάμεσου κώδικα απενεργοποιούνται.
- **Panic Mode**: αναζήτηση για ένα ασφαλές σύμβολο (π.χ. ';') από το οποίο επανεκκινείται η συντακτική ανάλυση.

Error Recovery στον Bison

- Προκαθορισμένη συμπεριφορά κατά την ανίχνευση σφάλματος:
  - κλήση yyerror() για την εκτύπωση διαγνωστικού μηνύματος
  - τερματισμός
- %error-verbose: οδηγία για πιο κατατοπιστικά μηνύματα
- Η προκαθορισμένη συμπεριφορά δεν υλοποιεί Error Recovery.
- Βρείτε πληροφορίες για το ειδικό σύμβολο γραμματικής **error**....

Ευχαριστώ!