ΕΙΣΑΓΩΓΗ ΣΤΟΝ BISON

Τι είναι ο Bison

Ο bison είναι μια βελτίωση του εργαλείου yacc του Unix.

Ο yacc είναι μια γεννήτρια συντακτικών αναλυτών που:

- δέχεται μια γραμματική χωρίς συμφραζόμενα (LALR(1)) και παράγει έναν συντακτικό αναλυτή σε C/C++
- Η παραγόμενη συνάρτηση **yyparse** αναγνωρίζει τις συμβολοσειρές εισόδου, κατασκευάζει το συντακτικό δέντρο και εκτελεί τις ενέργειες που περιγράφονται στο πρόγραμμα

Δομή Προγράμματος

```
%{
      Κώδικας C
      (μακροεντολές, τύποι δεδομένων, δηλώσεις μεταβλητών και
            συναρτήσεων)
%}
      Δηλώσεις bison
%%
      Κανόνες παραγωγής γραμματικής
%%
      Κώδικας C
      (υλοποίηση συναρτήσεων, main() )
```

Δηλώσεις Bison

Δηλώσεις λεκτικών μονάδων

```
%token SID
%token SIF, SELSE
```

Δηλώσεις τελεστών της γλώσσας και της προτεραιότητας και προσεταιριστικότητας τους.

```
%nonassoc '=' '<' '>'

%left '+' '-'

%left '*' '/' TK_div, TK_mod
```

Δήλωση του συνόλου των σημασιολογικών τιμών και του τύπου κάθε συμβόλου (στην επόμενη φάση)

Προτεραιότητα και Προσεταιριστικότητα Τελεστών

Τα **%left**, **%right** δηλώνουν τις προσεταιριστικότητες των tokens που τα ακολουθούν. Η σειρά δήλωσής τους καθορίζει την προτεραιότητα των αντίστοιχων τελεστών.

Ισχύει ότι:

- τα tokens που εμφανίζονται στην ίδια γραμμή έχουν την ίδια προτεραιότητα
- η προτεραιότητα αυξάνεται από πάνω προς τα κάτω.

π.χ.
```
%left '+' '-'
%left '*' '/' SDIV SMOD
```

- 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 '=' .
- Το **%prec** δηλώνει τη προτεραιότητα μέσα σε έναν κανόνα

Κανόνες Παραγωγής Γραμματικής

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, π.χ.

```
expression: term SPLUS expression ;
```

✓ Τα τερματικά σύμβολα (tokens) παριστάνονται με κεφαλαία ενώ τα μη τερματικά με πεζά κατά σύμβαση.
✓ Μπορούν να δίνονται περισσότερα εναλλακτικά δεξιά μέλη:

```
arithmetic_expr:   NUM
                 | arithmetic_expr '+' arithmetic_expr
                 | arithmetic_expr '-' arithmetic_expr
                 | arithmetic_expr '*' arithmetic_expr
                 | arithmetic_expr '/' arithmetic_expr
```

```
                    | '-' arithmetic_expr %prec UMINUS
                    | '(' arithmetic_expr  ')'
                    ;
id_list: /*empty*/
                  | id
                  | id SCOMMA id_list;
```

Flex και Bison

Στο πρώτο μέρος του αρχείου bison (myfile.y) δηλώνεται η συνάρτηση για το χειρισμό λαθών:

```
void yyerror(const char *)'
```

Η προκαθορισμένη συνάρτηση λεκτικής ανάλυσης που καλείται από τη **yyparse()** είναι η **yylex()**

Στο flex αρχείο:

- αφαιρούμε τις δηλώσεις των tokens
- προσθέτουμε το αρχείο C που παράγεται από το bison

```
#include "myfile.tab.h"
```

- αφαιρούμε τη main()

Διαδικασία Παραγωγής Συντακτικού Αναλυτή

Για δημιουργία κώδικα συντακτικής ανάλυσης, μεταγλώττισής του και εκτέλεσης του συντακτικού αναλυτή που θα προκύψει κάνουμε τα κάτωθι :

$ bison myfile .y *που μας δημιουργεί ως αρχείο εξόδου τον κώδικα της συντακτικής ανάλυσης σε C μέσα στο αρχείο εξόδου myfile.tab.c*

$ gcc -lm myfile.tab.c *που μας δημιουργεί τον εκτελέσιμο κώδικα της συντακτικής ανάλυσης βάσει της γραμματικής που ορίσαμε*

$ ./a.out *για να εκτελέσουμε τον συντακτικό αναλυτή.*

Αν όμως οι απαιτήσεις είναι μεγαλύτερες και απαιτείται ο συντακτικός αναλυτής να καλέσει τον λεκτικό αναλυτή, τότε η ακολουθία εντολών είναι η κάτωθι:

$ bison myfile .y

$ flex myfile.l

$ gcc -lm myfile.tab.c lex.yy.c *που μας δημιουργεί τον εκτελέσιμο κώδικα της συντακτικής ανάλυσης βάσει της γραμματικής που ορίσαμε*

$ ./a.out *για να εκτελέσουμε τον συντακτικό αναλυτή.*