Ζάχος Παναγιώτης 2117057

Terminal Name	Lexer.h	Parser.tab.h	Δ.Μ.
T_EOF	0	0	213
T_PROGRAM	1	258	8
T_CONST	2	259	14
T_TYPE	3	260	19
T_ARRAY	4	261	25
T_LIST	5	262	30
T_SET	6	263	34
T_OF	7	264	37
T_RECORD	8	265	44
T_VAR	9	266	48
T_FUNCTION	10	267	57
T_PROCEDURE	11	268	87
T_INTEGER	12	269	75
T_REAL	13	270	60
T_BOOLEAN	14	271	76
T_CHAR	15	272	80
T_FORWARD	16	273	94
T_LENGTH	17	274	100
T_NEW	18	275	104
T_BEGIN	19	276	109
T_END	20	277	113
T_IF	21	278	115
T_THEN	22	279	119
T_ELSE	23	280	123
T_WHILE	24	281	128
T_DO	25	282	132
T_CASE	26	283	137
T_OTHERWISE	27	284	146
T_FOR	28	285	139
T_TO	29	286	149
T_DOWNTO	30	287	154
T_WITH	31	288	160
T_READ	32	289	161
T_WRITE	33	290	166
T_ID	34	291	169
T_ICONST	35	294	215
T_RCONST	36	295	225
T_BCONST	37	296	197
T_CCONST	38	297	193
T_RELOP	39	298	171
T_ADDOP	40	299	170
T_OROP	41	300	183
T_MULDIVANDOP	42	301	172
T_NOTOP	43	302	182
T_INOP	44	303	179
T_LISTFUNC	45	293	248

T_STRING	46	292	255
T_LPAREN	47	304	202
T_RPAREN	48	305	203
T_SEMI	49	306	204
T_DOT	50	307	205
T_COMMA	51	308	206
T_EQU	52	309	207
T_COLON	53	310	208
T_LBRACK	54	311	209
T_RBRACK	55	312	210
T_ASSIGN	56	313	256
T_DOTDOT	57	314	212

Grammar conflicts as per bison .output file:

State 59 conflicts: 6 shift/reduce

(Βρίσκουμε το State 59 στο ίδιο αρχείο)

State 59

```
    8 expression: expression • "> or >= or < or <= or <>" expression
    9           | expression • "=" expression
   10           | expression • "IN" expression
   11           | expression • "OR" expression
   12           | expression • "+ or -" expression
   13           | expression • "* or / or DIV or AND or MOD" expression
   14           | "+ or -" expression •

    "> or >= or < or <= or <>"          shift, and go to state 65
    "+ or -"                            shift, and go to state 66
    "OR"                                shift, and go to state 67
    "* or / or DIV or AND or MOD"       shift, and go to state 68
    "IN"                                shift, and go to state 69
    "="                                 shift, and go to state 70

    "> or >= or < or <= or <>"          [reduce using rule 14 (expression)]
    "+ or -"                            [reduce using rule 14 (expression)]
    "OR"                                [reduce using rule 14 (expression)]
    "* or / or DIV or AND or MOD"       [reduce using rule 14 (expression)]
    "IN"                                [reduce using rule 14 (expression)]
    "="                                 [reduce using rule 14 (expression)]
    $default                            reduce using rule 14 (expression)
```

State 60 conflicts: 6 shift/reduce
State 121 conflicts: 6 shift/reduce
State 122 conflicts: 6 shift/reduce
State 123 conflicts: 6 shift/reduce
State 124 conflicts: 6 shift/reduce
State 125 conflicts: 6 shift/reduce
State 126 conflicts: 6 shift/reduce

State 208 conflicts: 1 shift/reduce

Ξεκινώντας από την πρώτη κατάσταση, είναι προφανές οτι υπάρχουν συγκρούσεις ολίσθησης/ελάττωσης για τους τελεστές της γλώσσας.

Μπορούμε να λύσουμε τις συγκρούσεις μετατρέποντας τη γραμματική μας σε διφορούμενη ή χρησιμοποιώντας την προτεραιότητα και προσεταιριστικότητα των τελεστών της γλώσσας, ώστε η ανάλυση να οδηγεί σε μοναδικό συντακτικό δέντρο. Επιλέγουμε να χρησιμοποιήσουμε τη δεύτερη μέθοδο επειδή είναι ευκολότερη χρησιμοποιώντας λειτουργίες του bison.

Από την σελίδα 17 του αρχείου της περιγραφής της γλώσσας, βλέπουμε την προτεραιότητα και προσεταιριστικότητα των τελεστών της γλώσσας. Στο bison, η προτεραιότητα γράφεται ανάποδα, δηλαδή θα ξεκινήσουμε γράφοντας την εντολή προσεταιριστικότητας για τους τελεστές INOP, RELOP και ‘=’. Εφ’όσον δεν έχουν καμία προσεταιριστικότητα, η εντολή θα είναι η:

```
%nonassoc     T_INOP        T_RELOP      T_EQU
```

Συνεχίζουμε προς τα πάνω στον πίνακα και έχουμε ADDOP, OROP με αριστερή προσεταιριστικότητα άρα γράφουμε την εντολή:

```
%left         T_ADDOP     T_OROP
```

Έπειτα MULDIVANDOP με επίσης αριστερή προσεταιριστικότητα:

```
%left         T_MULDIVANDOP
```

Αφού προσθέσαμε τις παραπάνω εντολές, ξανατρέχουμε το πρόγραμμα για να δούμε εάν επιλύσαμε τις συγκεκριμένες συγκρούσεις. Η έξοδος του bison πλέον μας βγάζει ότι υπάρχουν 7 συγκρούσεις απο τις 49 που είχαμε αρχικά, άρα επιλύσαμε 42 συγκρούσεις με τις 3 παραπάνω εντολές.

Η νέα έξοδος του bison (αρχείο .output):

```
State 90 conflicts: 6 shift/reduce
State 265 conflicts: 1 shift/reduce
```

Βρίσκουμε το State 90 στο ίδιο αρχείο:

```
State 90

   12 expression: expression • "> or >= or < or <= or <>" expression
   13           | expression • "=" expression
   14           | expression • "IN" expression
   15           | expression • "OR" expression
   16           | expression • "+ or -" expression
   17           | expression • "* or / or DIV or AND or MOD" expression
   19           | "NOT" expression •

    "> or >= or < or <= or <>"     shift, and go to state 95
    "+ or -"                       shift, and go to state 96
    "OR"                           shift, and go to state 97
    "* or / or DIV or AND or MOD"  shift, and go to state 98
    "IN"                           shift, and go to state 99
    "="                            shift, and go to state 100

    "> or >= or < or <= or <>"     [reduce using rule 19 (expression)]
```

"+ or -" [reduce using rule 19 (expression)]
"OR" [reduce using rule 19 (expression)]
"* or / or DIV or AND or MOD" [reduce using rule 19 (expression)]
"IN" [reduce using rule 19 (expression)]
"=" [reduce using rule 19 (expression)]
$default reduce using rule 19 (expression)

Οι συγκρούσεις φαίνονται ίδιες με τις προηγούμενες αλλά παρατηρούμε ότι τώρα υπάρχει και ο τελεστής NOT μέσα στα σύμβολα σύγκρουσης. Στο πινακάκι προτεραιότητας και προσεταιριστικότητας βλέπουμε οτι ο τελεστής NOTOP δεν έχει κάποια προσεταιριστικότητα αλλά έχει μεγαλύτερη προτεραιότητα από τον MULDIVANDOP.

Άρα προσθέτουμε την εντολή προτεραιότητας κάτω απο τις υπόλοιπες:
%precedence T_NOTOP

Ξανατρέχουμε το πρόγραμμα και παίρνουμε έξοδο πως πλέον υπάρχει μόνο μια σύγκρουση, άρα η σύγκρουση επιλύθηκε.

Η τελευταία σύγκρουση είναι η: State 265 conflicts: 1 shift/reduce
(Βρίσκουμε το State 265 στο ίδιο αρχείο .output)
State 265

112 if_statement: "if" expression "then" $@3 statement $@4 • if_tail

"else" shift, and go to state 282

"else" [reduce using rule 115 (if_tail)]
$default reduce using rule 115 (if_tail)

if_tail go to state 283

Παρατηρούμε πως η τελευταία σύγκρουση είναι η περίπτωση του **ξεκρέμαστου else**.
Δηλαδή, σε ποιο if ανήκει ένα else, σε μία τέτοια περίπτωση:

if a then if b then s else s2

Η παραπάνω έκφραση παράγει δύο συντακτικά δέντρα:
1) if a then (if b then s else s2)
2) if a then (if b then s) else s2

Συγκεκριμένα ,το πρόβλημα που αντιμετωπίζει ο Σ.Α. είναι εάν θα πρέπει να κάνει ολίσθηση ή ελάττωση όταν βρίσκεται στο σημείο:

if_statement : T_IF expression T_THEN statement ***** if_tail

Δηλαδή εάν θα κάνει ολίσθηση χρησιμοποιώντας την πρώτη παραγωγή του κανόνα if_tail ή ελάττωση με τον δεύτερο κανόνα που είναι η κενή συμβολοσειρά ε.

Συνήθως, θέλουμε το else να ανήκει στο πιο κοντινό if άρα θέλουμε να κάνουμε ολίσθηση στην πρώτη παραγωγή του if_tail. Αλλιώς, σημαίνει οτι περιμένουμε το else να ανήκει σε ένα προηγούμενο if το οποίο μπορεί να μην υπάρχει. Επομένως θέλουμε στη σύγκρουση ολίσθησης/ελάττωσης, ο συντακτικός αναλυτής να επιλέγει την ολίσθηση σε ELSE.

Ένας απο τους τρόπους για να γίνει αυτό είναι να προσθέσουμε στην δεύτερη παραγωγή του κανόνα if_tail ένα καινούριο τερματικό σύμβολο, έστω LOWER_THAN_ELSE και να θέσουμε την προτεραιότητά του χαμηλότερη από αυτή του T_ELSE.

Έτσι ο κανόνας **if_tail** μετατρέπεται σε:

```
if_tail  :      T_ELSE statement
         |      %empty %prec LOWER_THAN_ELSE {}
         ;
```

και προσθέτουμε την προτεραιότητα κάτω απο τις προηγούμενες εντολές, άρα συνολικά οι εντολές που προσθέσαμε για την επίλυση συγκρούσεων είναι:

```
%nonassoc    T_INOP        T_RELOP     T_EQU
%left        T_ADDOP    T_OROP
%left        T_MULDIVANDOP
%precedence  T_NOTOP
%precedence  LOWER_THAN_ELSE
%precedence  T_ELSE
```