

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the text 'ΕΡΓΑΣΤΗΡΙΟ'.

ΕΡΓΑΣΤΗΡΙΟ

# ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΜΕΡΟΣ Β2

## ΤΜΗΜΑ Β1

ΠΕΤΡΑΚΗ ΒΑΣΙΛΙΚΗ ΕΥΑΝΘΙΑ 19390193  
ΠΑΠΑΧΡΙΣΤΟΔΟΥΛΟΥ ΑΙΚΑΤΕΡΙΝΗ 19390185  
ΡΟΥΜΕΛΙΩΤΗΣ ΣΠΥΡΙΔΩΝ 19390205  
ΜΑΡΓΙΩΛΗΣ ΧΡΗΣΤΟΣ 19390133

## Περιεχόμενα

Στόχος εργασίας.....	2
Κώδικας FLEX .....	3
Κώδικας BISON.....	4
Αρχείο εισόδου .....	7
Αρχείο εξόδου .....	8
Ανάλυση Αρμοδιοτήτων .....	9
Βιβλιογραφία .....	10

## Στόχος εργασίας

---

Στην παρακάτω εργασία, ως ομάδα, αναλύουμε ένα εισαγωγικό κομμάτι των μεταγλωττιστών, συγκεκριμένα των συντακτικών αναλυτών. Αυτό θα γίνει πάνω στην εικονική γλώσσα προγραμματισμού «Uni-CLIPS», χρησιμοποιώντας το συντακτικό αναλυτή BISON σε συνεργασία με το FLEX.

Υπάρχουν ξεχωριστά ο ανανεωμένος κώδικας FLEX, ο κώδικας BISON και τα αρχεία εισόδου και εξόδου.

**Γίνεται compile και τρέχει κανονικά** σε 3 διαφορετικά μηχανήματα με Arch Linux, Fedora KDE και FreeBSD, και σε Virtual Machine με Ubuntu.

## Κώδικας FLEX

```

1. %option noyywrap
2. %{
3. #include <stdlib.h>
4. #include "syntax.tab.h"
5.
6. int      cw = 0; /* correct words */
7. int      ww = 0; /* wrong words */
8. %}
9.
10. /*
11.  * Με βάση το μέρος A2, υλοποιούμε τις κανονικές εκφράσεις και τις
12.  * αντιστοιχούμε στα κατάλληλα tokens.
13.  */
14. ARITH      "+" | "-" | "*" | "/" | "="
15. DELIM      [ \t\n]+
16. INT        0|[+-]?[1-9]+[0-9]*
17. FLOAT      [+-]?[0-9]+((\.[0-9]+)([eE][+-]?[0-9]*)?|([eE][+-]?[0-9]*)?)
18. STR        \"[^\\"\\]*(?:\\.[^\\"\\]*)*\
19. DEFIN      [A-Za-z]+[A-Za-z0-9_]*
20. VAR        \?[A-Za-z0-9]+
21. COMMENT    ;.*
22.
23. /*
24.  * Όταν βρίσκει οποιοδήποτε token, επιστρέφει την αντίστοιχη κατηγορία στο
25.  * οποίο ανήκει
26.  */
27. %%
28. "deffacts"      { cw++; return DEFFACTS; }
29. "defrule"       { cw++; return DEFRULE; }
30. "bind"          { cw++; return BIND; }
31. "read"          { cw++; return READ; }
32. "printout"      { cw++; return PRINT; }
33. "test"          { cw++; return TEST; }
34. "="            { cw++; return COMP; }
35. "("            { return LPAR; }
36. ")"            { return RPAR; }
37. "->"          { return ARROW; }
38. {ARITH}         { cw++; return ARITH; }
39. {INT}           { cw++; return INT; }
40. {FLOAT}         { cw++; return FLOAT; }
41. {STR}           { cw++; return STR; }
42. {DEFIN}         { cw++; return DEFIN; }
43. {VAR}           { cw++; return VAR; }
44. {DELIM}         { /* ignore whitespace */ }
45. {COMMENT}       { /* skip comments */ }
46. "\n"           { return NEWLINE; }
47. .               { ww++; return UNKNOWN; }
48. %%

```

## Κώδικας BISON

```

1.  %{
2.  #include <err.h>
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.
6.  /* Silence warnings... */
7.  extern int yylex(void);
8.
9.  /* Input and output files. */
10. extern FILE *yyin, *yyout;
11.
12. extern int cw;      /* correct words */
13. extern int ww;      /* wrong words */
14. int          ce = 0; /* correct expressions */
15. int          we = 0; /* wrong expressions */
16.
17. void          yyerror(const char *);
18. %}
19.
20. /* Tokens declared from flex. */
21. %token DEFFACTS DEFRULE BIND READ PRINT TEST ARITH INT FLOAT COMP
22. %token STR DEFIN VAR LPAR RPAR ARROW NEWLINE UNKNOWN
23.
24. %start prog
25.
26. %%
27. /* Start here. */
28. prog:
29.   | prog NEWLINE
30.   | prog expr
31.   ;
32.
33. /*
34.  * Declare numbers. Variables only accept numerical values so add them here as
35.  * well.
36.  */
37. num:
38.   INT
39.   | FLOAT
40.   | VAR
41.   ;
42.
43. /* Accept any number of strings (for use in printout) */
44. str:
45.   STR
46.   | str STR
47.   ;
48.
49. /* (= (expr)) */
50. cmp:
51.   LPAR COMP expr expr RPAR
52.   ;
53.
54. /* (test (= (expr))) */
55. test:
56.   LPAR TEST cmp RPAR
57.   ;
58.
59. /* (prinout (str)...) */

```

```

60. print:
61.   LPAR PRINT str RPAR
62.   ;
63.
64. fact:
65.   expr
66.   | fact expr
67.   ;
68.
69. /* We match expressions here. */
70. expr:
71.   num                                     /* numbers */
72.   | cmp                                { ce++; } /* comparisons */
73.   | test                               { ce++; } /* test keyword */
74.   | print                             { ce++; } /* (printout "str"... ) */
75.   | LPAR READ RPAR                    { ce++; } /* (read) */
76.   | LPAR ARITH expr expr RPAR         { ce++; } /* (arithmetic_op (expr)... ) */
77.   | LPAR BIND VAR expr RPAR           { ce++; } /* (bind ?var (expr)) */
78.   | LPAR DEFFACTS DEFIN fact RPAR     { ce++; } /* (deffacts DEF facts...) */
79.   /* (defrule DEF
80.      *      (facts)
81.      *      ...
82.      *      (test)
83.      *      ->
84.      *      (printout))
85.   */
86.   | LPAR DEFRULE DEFIN fact test ARROW print RPAR { ce++; }
87.   | error                                         { we++; if (ce > 0) ce--; }
88.   ;
89. %%
90.
91. /* Print errors. */
92. void
93. yyerror(const char *s)
94. {
95.   fprintf(stderr, "%s\n", s);
96. }
97.
98. int
99. main(int argc, char *argv[])
100. {
101.   /* We need at least 1 input and 1 output file... */
102.   if (argc < 3) {
103.     fprintf(stderr, "usage: %s input... output\n", *argv);
104.     return (-1);
105.   }
106.
107.   /* Open last file as output. */
108.   if ((yyout = fopen(argv[--argc], "w")) == NULL)
109.     err(1, "fopen(%s)", argv[argc]);
110.
111.   /* Parse all input files in reverse order. */
112.   while (argc-- > 1) {
113.     if ((yyin = fopen(argv[argc], "r")) == NULL)
114.       err(1, "fopen(%s)", argv[argc]);
115.     /* Parse file */
116.     if (yyvsparse() == 0)
117.       fprintf(yyout, "%s: success\n", argv[argc]);
118.     else
119.       fprintf(yyout, "%s: failure\n", argv[argc]);
120.     fclose(yyin);
121.   }
122.

```

```
123.      /* Print results. */
124.      fprintf(yyout, "\n");
125.      fprintf(yyout, "correct words: %d\n", cw);
126.      fprintf(yyout, "correct expressions: %d\n", ce);
127.      fprintf(yyout, "wrong words: %d\n", ww);
128.      fprintf(yyout, "wrong expressions: %d\n", we);
129.
130.      fclose(yyout);
131.
132.      return (0);
133. }
```

Δεν έχει υλοποιηθεί σωστά η δομή του defrule-deffacts.

## Αρχείο εισόδου

---

```
1. (printout "hello" "hello")
2. (bind ?var 1)
3. (bind ?var (+ 1 2))
4. (test (= 1 2))
5. (= 1 (+ 2 3))
6.
7. (defrule move-up
8.   (+ 1 2)
9.   (- 1 (+ 1 (* 1 2))))
10. (test (= 1 2))
11. ->
12. (printout "hello"))
```



## Αρχείο εξόδου

---

```
1. input.txt: success
2.
3. correct words: 38
4. correct expressions: 12
5. wrong words: 0
6. wrong expressions: 0
```

## Ανάλυση Αρμοδιοτήτων

---

- Πετράκη Βασιλική Ευανθία

Συγγραφή παραδοτέου, debugging κώδικα

- Παπαχριστοδούλου Αικατερίνη

Συγγραφή κώδικα

- Ρουμελιώτης Σπυρίδων

Debugging κώδικα

- Μαργιώλης Χρήστος

Συγγραφή κώδικα

Η συγγραφή της εργασίας πραγματοποιούταν σε πραγματικό χρόνο, μέσω ομαδικής κλήσης, οπότε όλα τα μέλη είναι ενημερωμένα για όλα τα κομμάτια της.

## Βιβλιογραφία

---

- <https://stackoverflow.com/>
- <https://eclass.uniwa.gr/courses/CS118/>
- «Μεταγλωττιστές» - Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman
- “Regular Expressions Cheat Sheet” by DaveChild