

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 Β3

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and curve upwards and to the right.

ΤΜΗΜΑ Β1

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

Περιεχόμενα

Στόχος εργασίας.....	2
Κώδικας FLEX	3
Κώδικας BISON.....	5
Αρχείο εισόδου	8
Αρχείο εξόδου	9
Ανάλυση Αρμοδιοτήτων	10
Βιβλιογραφία	11

Στόχος εργασίας

Στην παρακάτω εργασία, ως ομάδα, αναλύουμε τα συντακτικά και λεκτικά λάθη που μπορούν να προκύψουν, κατά την μεταγλώττιση ενός κώδικα στην γλώσσα “Uni-CLIPS”.

Υπάρχουν ξεχωριστά οι κώδικες FLEX και BISON, μαζί με 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. %x error
3. %{
4. #include <stdlib.h>
5. #include "syntax.tab.h"
6.
7. int      cw = 0; /* correct words */
8. int      ww = 0; /* wrong words */
9. int      lineno = 1; /* current line */
10. int     errcnt = 0; /* error count */
11. %}
12.
13. /*
14.  * Με βάση το μέρος A2, υλοποιούμε τις κανονικές εκφράσεις και τις
15.  * αντιστοιχούμε στα κατάλληλα tokens.
16.  */
17. ARITH      "+" | "-" | "*" | "/" | "="
18. DELIM      [ \t ]+
19. INT        0 | [+-]? [1-9] + [0-9]*
20. FLOAT      [+-]? [0-9] + ( ( \. [0-9] + ) ( [eE] [+-]? [0-9]* ) ? | ( [eE] [+-]? [0-9]* ) ? )
21. STR        \" [^\"\\] * ( ? : \\. [^\"\\] * ) * \"
22. DEFIN      [A-Za-z] + [A-Za-z0-9_ - ] *
23. VAR        \? [A-Za-z0-9] +
24. COMMENT    ; . *
25.
26. /*
27.  * Όταν βρίσκει οποιοδήποτε token, επιστρέφει την αντίστοιχη κατηγορία στο
28.  * οποίο ανήκει
29.  */
30. %%
31. "deffacts"      { cw++; return DEFFACTS; }
32. "defrule"       { cw++; return DEFRULE; }
33. "bind"          { cw++; return BIND; }
34. "read"          { cw++; return READ; }
35. "printout"      { cw++; return PRINT; }
36. "test"          { cw++; return TEST; }
37. "="            { cw++; return COMP; }
38. "("            { return LPAR; }
39. ")"            { return RPAR; }
40. "->"           { return ARROW; }
41. {ARITH}         { cw++; return ARITH; }
42. {INT}           { cw++; return INT; }
43. {FLOAT}        { cw++; return FLOAT; }
44. {STR}           { cw++; return STR; }
45. {DEFIN}         { cw++; return DEFIN; }
46. {VAR}           { cw++; return VAR; }
47. {DELIM}         { /* ignore whitespace */ }
48. {COMMENT}       { /* skip comments */ }
49. "\\n"           { lineno++; return NEWLINE; }
50. .               {
51.     printf("error: line %d: unrecognized token: %s\\n",
52.           lineno, yytext);
53.     errcnt++;
54.     ww++;
55.     BEGIN(error);
56.     return UNKNOWN;
57. }
58. <error>[ \t] +  { BEGIN(0); }
59. <error>.        { }

```

```
60. <error>\n    { BEGIN(0); }  
61. %%
```

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  lineno;
13. extern int  errcnt;
14. extern int  cw;          /* correct words */
15. extern int  ww;          /* wrong words */
16. int         ce = 0;      /* correct expressions */
17. int         we = 0;      /* wrong expressions */
18. int         warncnt = 0; /* warning count */
19.
20. void        yyerror(const char *);
21. %}
22.
23. /* Tokens declared from flex. */
24. %token DEFFACTS DEFRULE BIND READ PRINT TEST ARITH INT FLOAT COMP
25. %token STR DEFIN VAR LPAR RPAR ARROW NEWLINE UNKNOWN
26.
27. %start prog
28.
29. %%
30. /* Start here. */
31. prog:
32. | prog NEWLINE
33. | prog expr NEWLINE
34. | prog error NEWLINE {
35.     printf("error: line %d: unexpected token\n", lineno);
36.     errcnt++;
37.     we++;
38.     yyerrok; }
39. ;
40.
41. /*
42.  * Declare numbers. Variables only accept numerical values so add them here as
43.  * well.
44.  */
45. num:
46.     INT
47.     | FLOAT
48.     | VAR
49.     ;
50.
51. /* Accept any number of strings (for use in printout) */
52. str:
53.     STR
54.     | str STR
55.     ;
56.
57. /* (= (expr)) */
58. cmp:
59.     LPAR COMP expr expr RPAR

```

```

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

```

```
123.
124.     /* Parse all input files in reverse order. */
125.     while (argc-- > 1) {
126.         if ((yyin = fopen(argv[argc], "r")) == NULL)
127.             err(1, "fopen(%s)", argv[argc]);
128.         /* Parse file */
129.         if (yyparse() == 0)
130.             fprintf(yyout, "%s: parse: success\n", argv[argc]);
131.         else
132.             fprintf(yyout, "%s: parse: failure\n", argv[argc]);
133.         fclose(yyin);
134.     }
135.
136.     /* Print results. */
137.     fprintf(yyout, "\n");
138.     fprintf(yyout, "correct words: %d\n", cw);
139.     fprintf(yyout, "correct expressions: %d\n", ce);
140.     fprintf(yyout, "wrong words: %d\n", ww);
141.     fprintf(yyout, "wrong expressions: %d\n", we);
142.     fprintf(yyout, "\nwarnings: %d\n", warncnt);
143.     fprintf(yyout, "fatal errors: %d\n", errcnt);
144.     /* If we encountered more than 1 error, print appropriate message. */
145.     if (errcnt > 0)
146.         fprintf(yyout, "\ncompilation failed\n");
147.     else
148.         fprintf(yyout, "\ncompilation succeeded\n");
149.
150.     fclose(yyout);
151.
152.     return (0);
153. }
```


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. (++ 1 2)
8. ($ 2 1 % ^)
9. (+ 1 2) *
10. asdasd
11. ads
12.
13. (defrule move-up (+ 1 2) (- 1 (+ 1 (* 1 2))) (test (= 1 2)) ->
    (printout "hello"))
```

Αρχείο εξόδου

```
1. input.txt: parse: success
2.
3. correct words: 50
4. correct expressions: 12
5. wrong words: 3
6. wrong expressions: 4
7.
8. warnings: 1
9. fatal errors: 9
10.
11. compilation failed
```

Ανάλυση Αρμοδιοτήτων

- Πετράκη Βασιλική Ευανθία

Συγγραφή παραδοτέου, debugging κώδικα

- Παπαχριστοδούλου Αικατερίνη

Debugging 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