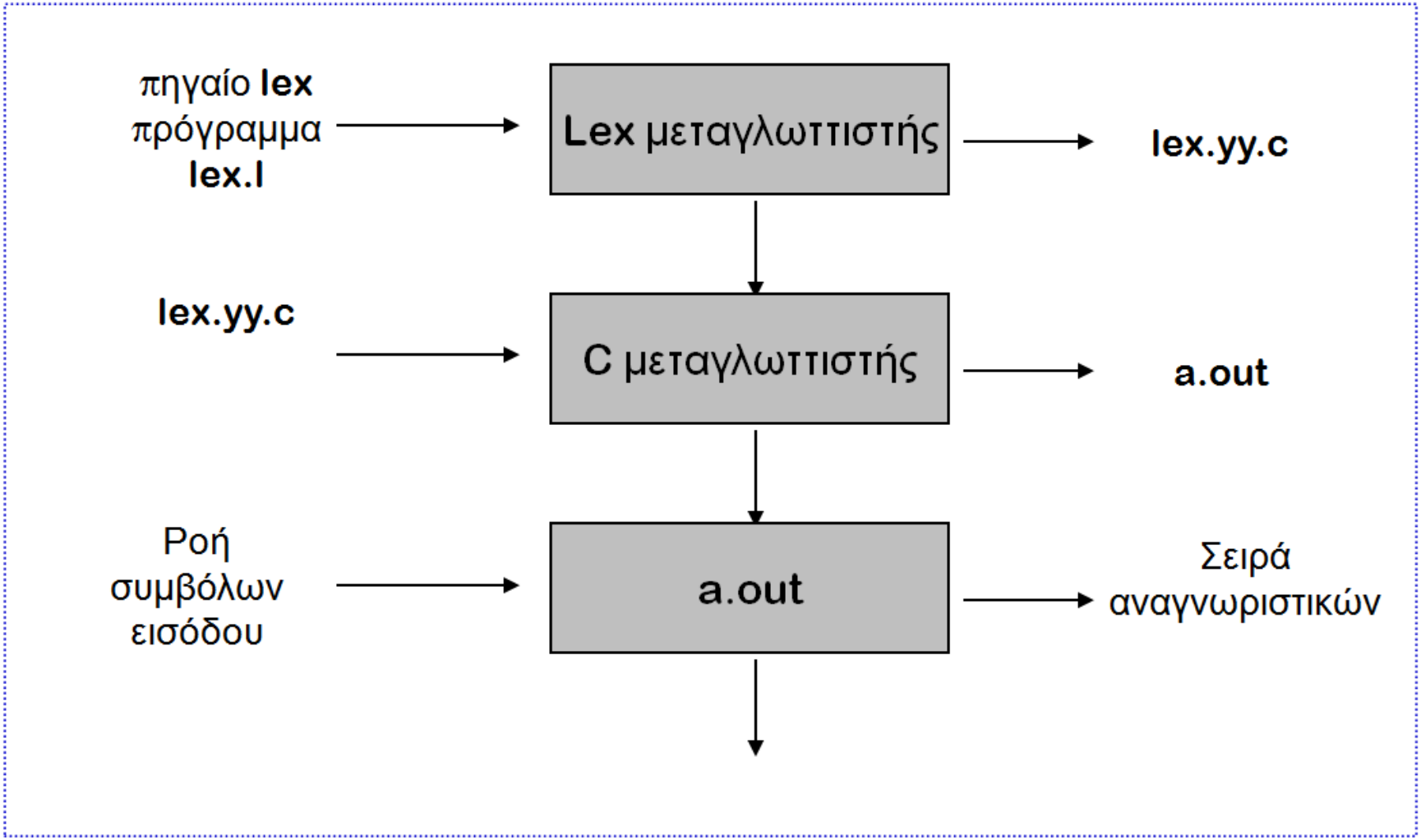
Παρουσίαση εργαλείου Flex

& Μέρος Α-3 της εργασίας εξαμήνου

Εισαγωγή στο εργαλείο flex

- Το flex είναι μια γεννήτρια λεκτικών αναλυτών, δηλαδή ένα εργαλείο που
  - Βοηθάει τον προγραμματιστή να γράψει (πιο) εύκολα τον δικό του λεκτικό αναλυτή
  - Βασίζεται στη γλώσσα C αλλά χρησιμοποιεί κανονικές εκφράσεις για την αναγνώριση λεκτικών μονάδων
- Το εγχειρίδιο χρήσης υπάρχει στο eClass
  - 3-ΕΡΓΑΣΤΗΡΙΟ » 2-FLEX
    - 2-FLEX manual

Βήματα δημιουργίας Λ.Α.



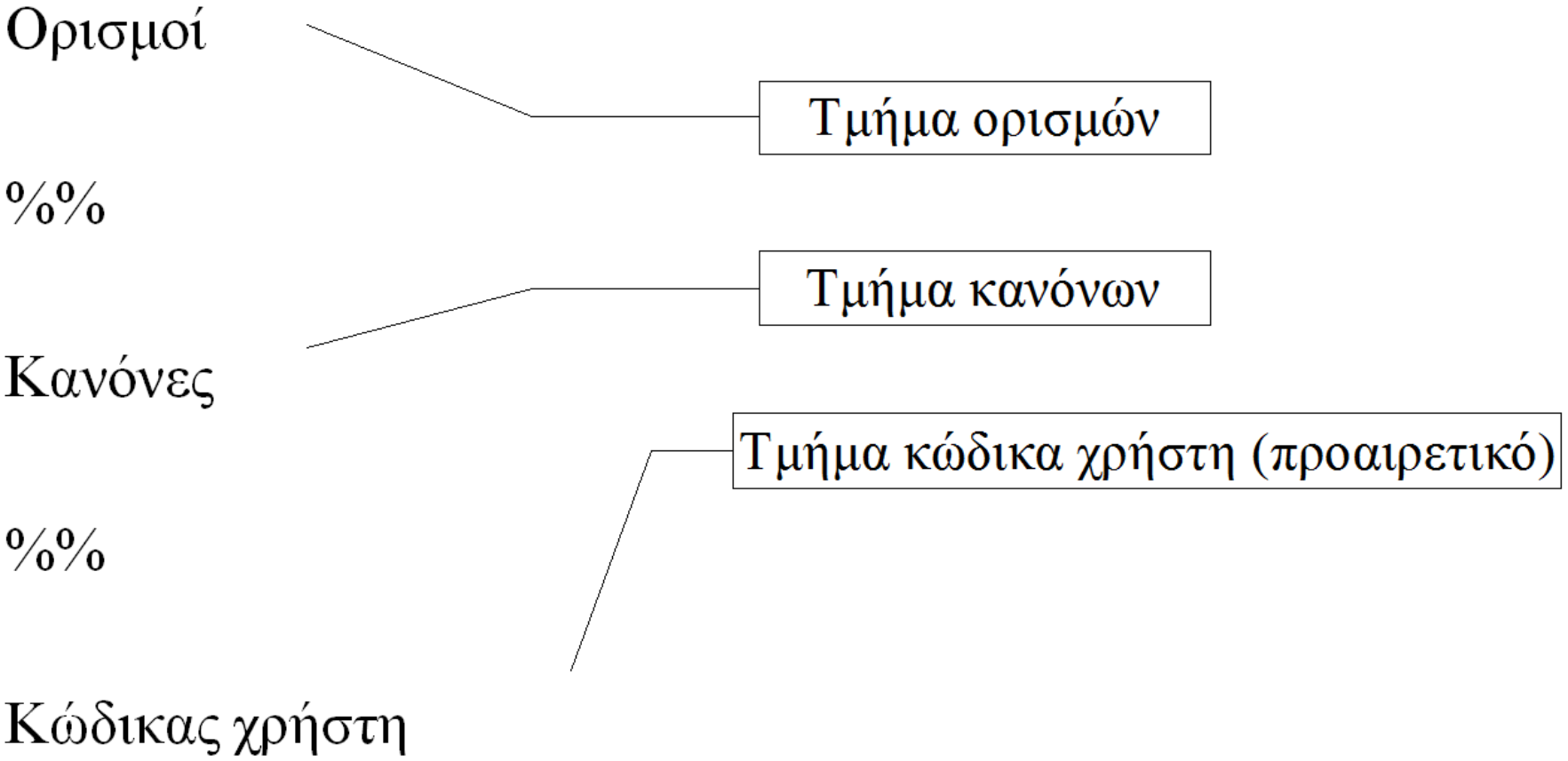
Γεννήτρια λεκτικών αναλυτών (1)

- Το αρχείο περιγραφής διοχετεύεται ως είσοδος στη γεννήτρια κώδικα λεκτικής ανάλυσης
- Ο κώδικας που προκύπτει ως αποτέλεσμα περιλαμβάνεται στο αρχείο lex.yy.c και περιλαμβάνει τη συνάρτηση λεκτικής ανάλυσης yylex()
- Αν το αρχείο περιγραφής περιλαμβάνει συνάρτηση main(), τότε το πρόγραμμα που παράχθηκε μπορεί να λειτουργήσει αυτόνομα

Γεννήτρια λεκτικών αναλυτών (2)

- Για να λειτουργήσει αυτόνομα θα πρέπει να «περάσει» από έναν μεταγλωττιστή της γλώσσας C
- Αν το αρχείο περιγραφής δεν περιλαμβάνει τη main(), τότε για να χρησιμοποιηθεί ο παραγόμενος κώδικας στο πλαίσιο ενός άλλου προγράμματος χρειάζεται να συμπεριληφθεί με την οδηγία:
  - #include lex.yy.c

Δομή προγράμματος



Κώδικας για δοκιμή: test1.l

```
%{
    #include <stdio.h>
    int num_lines = 0, num_chars = 0;
%}

%%

\n    { num_lines++; num_chars++; }
.     { num_chars++; }

%%

    int main()
    {
        yylex();
        printf("# of lines = %d\n", num_lines);
        printf("# of chars = %d\n", num_chars);
    }
```

Κώδικας C

Περιοχή Κανόνων

Κώδικας C

Ορισμοί (Definitions) (1)

- Κώδικας ο οποίος δεν ανήκει σε συγκεκριμένες συναρτήσεις και είναι επιθυμητή η συμπερίληψή του στον παραγόμενο λεκτικό αναλυτή
- Ο κώδικας αυτός εμφανίζεται ανάμεσα στην ειδική ακολουθία χαρακτήρων %{ και %} και προηγείται των χαρακτήρων %% που διαχωρίζουν το πρώτο από το δεύτερο μέρος της περιγραφής

Ορισμοί (Definitions) (2)

- Ορισμοί ονομάτων κανονικών εκφράσεων. Κάθε ένα από αυτά ορίζεται σε ξεχωριστή γραμμή, όπου προηγείται το όνομα και μετά από ένα ή περισσότερα κενά ακολουθεί η κανονική έκφραση από την οποία περιγράφεται

Κανόνες αναγνώρισης (Rules)

- Στο δεύτερο μέρος του αρχείου περιγραφής του λεκτικού αναλυτή περιλαμβάνονται κανόνες αναγνώρισης διατυπωμένοι στη μορφή:
  - p_1 {ενέργεια-1}
  - p_2 {ενέργεια-2}
  - ... . . .
  - p_n {ενέργεια-n}
- Όπου κάθε p_i είναι μια κανονική έκφραση και κάθε ενέργεια μία ή περισσότερες εντολές της C που εκτελούνται κάθε φορά που αναγνωρίζεται η αντίστοιχη λεκτική μονάδα

Κώδικας χρήστη (User code)

- Στο τελευταίο μέρος της περιγραφής περιλαμβάνονται όποιες συναρτήσεις καλούνται από το δεύτερο μέρος και δεν ορίζονται αλλού.
- Αν θέλουμε το αποτέλεσμα της επεξεργασίας της γεννήτριας να λειτουργεί ως αυτόνομο πρόγραμμα, τότε στο μέρος αυτό περιλαμβάνεται και η συνάρτηση main()

Συνάρτηση yylex

- Η σημαντικότερη συνάρτηση, μετά την main για τον λεκτικό σας αναλυτή
- Η yylex είναι αυτή που κάνει την λεκτική ανάλυση
  - Κάθε φορά που κάποιος την καλεί διαβάζει χαρακτήρες από την είσοδο μέχρι
    - να αναγνωρίσει κάποια λεκτική μονάδα
    - να φτάσει στο τέλος της εισόδου

Οδηγίες για τη δημιουργία ΛΑ

- Δημιουργία του αρχείο με τον κατάλληλο κώδικα για το εργαλείο flex
  - Το αρχείο πρέπει οπωσδήποτε να έχει επέκταση .l
- «Πέρασμα» του αρχείου .l από τον flex για τη δημιουργία κώδικα C
  - flex –o test1.c test1.l
- Μεταγλώττιση του αρχείου C
  - gcc –o test1 test1.c
- Εκτέλεση Λ.Α.
  - test1 [Windows] ή ./test1 [Linux]

Πιθανό σφάλμα

- Undefined reference to yywrap()
  - Η συνάρτηση yywrap καλείται μετά την αναγνώριση ενός ολόκληρου αρχείου εισόδου
    - Πρακτικά μετά την ανάγνωση του χαρακτήρα EOF
  - Αν το αρχείο είναι μονάχα ένα, τότε δεν έχει νόημα η κλήση της και γενικά η ύπαρξή της
  - Υπάρχουν 3 πιθανές λύσεις για την αντιμετώπιση του μηνύματος σφάλματος
    - Γράφουμε τη δική μας συνάρτηση yywrap
    - %option noyywrap
    - gcc -o test1 test1.c **-lfl**

Εκτέλεση ΛΑ

- Εκτέλεση του λεκτικού αναλυτή
  - ./test1 [Linux]
  - test1 [Windows]
- Ακολούθως γράφουμε την συμβολοσειρά εισόδου
- Ολοκληρώνουμε την διαδικασία πατώντας Ctrl+D (στο Linux) ή Ctrl+Z (στα Windows) που ενημερώνει το εργαλείο ότι δεν επιθυμούμε να δώσουμε άλλους χαρακτήρες για είσοδο

Κώδικας για δοκιμή: test2.l

```
%option noyywrap
%{
    #include <stdio.h>
%}

binary      [01]+

%%

{binary}    { ECHO; printf("\n"); }
.           { }

%%

    int main()
    {
        yylex();
    }
```

{ Κώδικας C

{ Περιοχή Ορισμών

{ Περιοχή Κανόνων

{ Κώδικας C

```
flex -o test2.c test2.l
gcc -o test2 test2.c
./test2     ή     test2
```

Παραδείγματα

- Μερικά χρήσιμα παραδείγματα
  - 3-ΕΡΓΑΣΤΗΡΙΟ » 2-FLEX » ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΡΧΕΙΩΝ FLEX
    - flex examples of simple input files.zip

- Υπενθυμίζεται η διαδικασία εκτέλεσης:
  - flex -o ex1.c ex1.l
  - gcc -o ex1 ex1.c
  - ./ex1 [Linux] ή ex1 [Windows]

1ο παράδειγμα (ex1.l)

- Δοκιμάστε το παρακάτω πρόγραμμα:

```
%option noyywrap
%{
        #include <stdio.h>
        int lineno = 1;
%}
line    .*\n

%%

{line}    { printf("%d %s", lineno++, yytext); }
%%

void main()
{ yylex(); }
```

- Υπενθύμιση:
  - flex -o ex1.c ex1.l
  - gcc -o ex1 ex1.c
  - ./ex1 ή ex1

Μεταβλητές yylval και yytext

- Για την επιστροφή τιμών ιδιοτήτων της λεκτικής μονάδας, μπορεί να χρησιμοποιηθεί η εσωτερική μεταβλητή yylval της γεννήτριας
- Ακόμη, ιδιαίτερα χρήσιμη είναι και η μεταβλητή yytext (τύπου char*) που αποθηκεύεται προσωρινά η συμβολοσειρά της λεκτικής μονάδας που αναγνωρίστηκε
- Χρήσιμη είναι και η μεταβλητή yyleng
  - Η τιμή της είναι ο αριθμός των χαρακτήρων (μήκος) της λεκτικής μονάδας που αναγνωρίστηκε

2ο παράδειγμα (ex2.l)

- Υπενθύμιση:
  - flex –o ex2.c ex2.l
  - gcc –o ex2 ex2.c
  - ./ex2
- Προτεραιότητα;

```
%option noyywrap
%{
        #include <stdio.h>
%}
ends_with_a     .*a\n
begins_with_a   a.*\n

%%

{ends_with_a}   ECHO;
{begins_with_a} ECHO;
.*\n            ;

%%

void main()
{ yylex(); }
```

- Υπενθύμιση:
  - flex –o ex2.c ex2.l
  - gcc –o ex2 ex2.c
  - ./ex2 ή ex2

3ο παράδειγμα (ex3.l)

```
%{
        #include "header.h"
        void yyerror(char *s);
        int linenum = 1;
%}

digit           [0-9]
int_const       {digit}+

%%

{int_const}     { printf("%d\n", atoi(yytext)); return INTEGER_LITERAL; }
"+"             { printf("%s\n", yytext); return PLUS; }
"*"             { printf("%s\n", yytext); return MULT; }
[ \t]*          {}
[\n]            { linenum++; }
.               { yyerror("Error Message!"); exit(1); }

%%

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

- Υπενθύμιση:
  - flex -o ex3.c ex3.l
  - gcc -o ex3 ex3.c
  - ./ex3 ή ex3

Η συνάρτηση main που λείπει

```
void main()
{
	yylex();
}
```

```
int main()
{
	int out = -1;
	while(out != 0)
	{
		out = yylex();
		printf("yylex finished with '%d'\n", out);
	}
}
```

Εντολή return

- Η εντολή return επιστρέφει στον συντακτικό αναλυτή το αναγνωριστικό της τελευταίας λεκτικής μονάδας που αναγνωρίστηκε
- Σημαντική εντολή στη συνεργασία του flex με το εργαλείο δημιουργίας συντακτικών αναλυτών bison

header.h

- Γιατί υπάρχει το header.h;
- Τι σκοπό εξυπηρετεί;

```
#define INTEGER_LITERAL 1
#define PLUS 2
#define MULT 3
```

Συνάρτηση yyerror

- void yyerror(s)
  - Όπου s: const char *s
- Η συνάρτηση χειρισμού σφαλμάτων yyerror δίνεται προαιρετικά από τον χρήστη
- Η συμβολοσειρά s περιέχει το μήνυμα λάθους που παρουσιάστηκε

4ο παράδειγμα (ex4.l)

```
%{
        #include "header.h"
        void yyerror(char *s);
        int linenum = 1;
%}

digit           [0-9]
int_const       {digit}+

%%

{int_const}     { printf("\tRead: %d (2x=%d)\n", atoi(yytext), atoi(yytext)*2);
                  return INTEGER_LITERAL; }
"+"             { printf("\tLine=%d, Read: %s\n", linenum, yytext);
                  return PLUS; }
"*"             { printf("\tLine=%d, Read: %s\n", linenum, yytext);
                  return MULT; }
[ \t]*          {}
[\n]            { linenum++; }
.               { yyerror("Error: Unrecognized text"); exit(1); }

%%

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

Χρήσιμες μαθηματικές συναρτήσεις

- atof
  - ASCII to floating point
    - Μετατροπή συμβολοσειράς σε δεκαδικό αριθμό
- atoi
  - ASCII to integer
    - Μετατροπή συμβολοσειράς σε ακέραιο αριθμό

5ο παράδειγμα

```
%option noyywrap
%{
        #include <stdio.h>
%}

%%

[\t\n]              ;
[0-9]+\.[0-9]+  { printf("Float: %s\n", yytext); }
[0-9]+              { printf("Integer: %s\n", yytext); }
[a-zA-Z]+           { printf("Letters: %s\n", yytext); }
.                   { printf("Unknown: %s\n", yytext); }

%%

void main()
{
        yylex();
}
```

<- return ???

Αρχεία: Είσοδος και έξοδος (1)

- Πως γίνεται να δέχεται το πρόγραμμα είσοδο από αρχείο;
  - FILE *yyin
  - < input.txt
- Πως γίνεται να δίνει έξοδο σε αρχείο;
  - FILE *yyout
  - > output.txt

Αρχεία: Είσοδος και έξοδος (2)

- Είσοδος από αρχείο ή standard input
  - ./test input.txt
  - ./test

```
int main(int argc,char **argv)
{
        int i;
        if(argc == 2)
                yyin=fopen(argv[1],"r");
        else
                yyin=stdin;
      yylex();
}
```

Αρχεία: Είσοδος και έξοδος (3)

- Είσοδος από συγκεκριμένο αρχείο
  - ./test

```
main() {
      FILE *myfile = fopen("input.txt", "r");
      if (!myfile)     {
            {
            printf("I can't open input.txt !");
            return -1;
            }
      yyin = myfile;
      yylex();
        }
```

Αρχεία: Είσοδος και έξοδος (4)

- Είσοδος από αρχείο & έξοδος σε αρχείο

```
int main(int argc, char **argv)
{
      if(argc == 3){
            if(!(yyin = fopen(argv[1], "r"))) {
                  fprintf(stderr, "Cannot read file: %s\n", argv[1]);
                  return 1;
            }
            if(!(yyout = fopen(argv[2], "w"))) {
                  fprintf(stderr, "Cannot create file: %s\n", argv[2]);
                  return 1;
            }
      }
      else if(argc == 2){
              if(!(yyin = fopen(argv[1], "r"))) {
                  fprintf(stderr, "Cannot read file: %s\n", argv[1]);
                  return 1;
               }
            }
      yylex();
 }
```

Διαχείριση λαθών (1)

- Παράδειγμα με διαχείριση λαθών
  - 3-ΕΡΓΑΣΤΗΡΙΟ » 2-FLEX » ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΡΧΕΙΩΝ FLEX
    - flex-only-errscan
- Αποσυμπίεση σε τερματικό Linux
  - unrar x flex-only-errscan2.rar
- Makefile και make
  - Σημ: Καλό είναι το περιεχόμενο του rar να αποσυμπιεστεί σε ξεχωριστό κατάλογο

Διαχείριση λαθών (2)

```
%option noyywrap
%x error
. . . . . .
DELIMITERS              [ \t]+
INTEGER                 [0-9]+
INTEGER_ERROR           {INTEGER}[A-Za-z]+

%%

"+"                     { prn("PLUS"); return PLUS; }
{DELIMITERS}            { }
\n                      { line++; }
{INTEGER}               { prn("INTCONST"); return INTCONST; }
{INTEGER_ERROR}         { ERROR("Integer ends with letter!");
                                        return TOKEN_ERROR; }
.                       { ERROR("Unrecognized token error!");
                                BEGIN(error); return TOKEN_ERROR; }
<error>[ \t\n]          { BEGIN(0); }
<error>.                { }
. . . . . .
```

Λοιπές χρήσιμες συναρτήσεις

- void yymore()
  - Επιτρέπει στον λεκτικό αναλυτή να προχωρήσει στην αναγνώριση της επόμενης λεκτικής μονάδας, διατηρώντας στην μεταβλητή yytext τη συμβολοσειρά της λεκτικής μονάδας που αναγνωρίστηκε τελευταία

- void yyless(int n)
  - Οπισθοδρόμηση n χαρακτήρων στη συμβολοσειρά εισόδου της λεκτικής ανάλυσης

Εργασία Α-3 (& Α-2)

- Εργασία Μεταγλωττιστών
  - Μέρος Α-3
    - Συμπλήρωση πρότυπου κώδικα Flex
      - simple-flex-code.zip
    - Αναγνώριση όλων των λεκτικών μονάδων της Uni-C
    - Σχολιασμός κώδικα
  - Μέρος Α-2 (υποβολή μαζί με το μέρος Α-3!)

- Προσοχή στα ζητούμενα!
  - Διαβάστε προσεκτικά την εκφώνηση
  - Έγγραφο τεκμηρίωσης με όλα τα απαιτούμενα
    - Μην ξεχάσετε:
      - αναλυτική αναφορά σε ρόλους / αρμοδιότητες
      - αναφορά σε ελλείψεις και προβλήματα