ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΑΘΗΜΑ: ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ
ΕΡΓΑΣΙΑ 1η ΜΕΡΟΣ Α3

ΤΜΗΜΑ: Β2
ΟΜΑΔΑ: 1η
ΜΕΛΗ ΟΜΑΔΑΣ:

1. Φράγκος Μαρίνος (ΠΑΔΑ – 20390255) 8ο Εξάμηνο
2. Φριλίγκος Γρηγόριος (ΠΑΔΑ – 20390258) 8ο Εξάμηνο
3. Βροχάρης Αντώνιος (ΠΑΔΑ – 20390030) 8ο Εξάμηνο
4. Φάσσου Κοντοδημάκη Ιφιγένεια Γεωργία (ΠΑΔΑ – 20390249) 8ο Εξάμηνο
5. Διάννης Ιωάννης (ΠΑΔΑ – 21390053) 6ο Εξάμηνο

Περιεχόμενα

Εισαγωγή

Στην δεύτερη εργαστηριακή άσκηση καλούμαστε να ολοκληρώσουμε τον πρότυπο ημιτελή κώδικα simple-flex-code.l. Ο συγκεκριμένος κώδικας αφορά την αναγνώριση ακέραιων και μεταβλητών. Πιο συγκεκριμένα πρέπει να αντικαταστήσουμε τα απαραίτητα κομμάτια μέσα στον κώδικα flex καθώς και στο αντίστοιχο token.h αρχείο header με τον κώδικα που λείπει.

Εφόσον ολοκληρωθούν οι αντικαταστάσεις θα δοκιμάσουμε την εκτέλεση του αρχείου μας με τα συντακτικά ορθά λεξήματα που δίνονται στο αρχείο input.txt, θα συγκρίνουμε το αποτελέσματα της εκτέλεσης με αυτά που δίνονται στο αρχείο output.txt και σκοπός είναι να κατανοήσουμε τη λειτουργία της γεννήτριας.

Εφόσον ολοκληρωθούν τα παραπάνω βήματα ολοκληρώνουμε το λεκτικό αναλυτή ώστε να αναγνωρίζει όλες τις λεκτικές μονάδες, τα διαχωριστικά και τα σχόλια της γλώσσας που μας έχει δοθεί. Επίσης στον κώδικα που θα συμπληρώσουμε στο παραπάνω αρχείο .l θα δώσουμε τα απαραίτητα σχόλια χρησιμοποιώντας τις γνώριμες από τη γλώσσα C ακολουθίες χαρακτήρων για τα multiline comments.

Ο λεκτικός αναλυτής μας, στην τελική του μορφή, θα είναι σε θέση να αναγνωρίζει όλες τις κανονικές εκφράσεις τις οποίες μελετήσαμε και δημιουργήσαμε στο προηγούμενο μέρος της εργαστηριακής άσκησης, το μέρος Α2. Τέλος το αρχείο εξόδου θα πρέπει να ακολουθεί συγκεκριμένους κανόνες για την αποτύπωση των αποτελεσμάτων.

Κώδικας flex

```
%option noyywrap


%{


#include <stdio.h>

#include <string.h>

#include <stdlib.h>

 /*Εισαγωγή του header file που περιέχει τα tokens*/

#include "token.h"

 /*Δήλωση μεταβλητής για τον αριθμό της γραμμής*/

int line = 1;


%}

 /*Δήλωση των regular expressions για τα διάφορα tokens*/

DELIMITER       [ \t]+

OPERATOR                                (\+{1,2}|-{1,2}|\*|\/|%|=|\+=|-
=|\*=|\/=|!|&&|\|\||==|!=|<|>|<=|>=|&)

KEYWORD
(break|case|func|const|continue|do|double|else|float|for|if|int|long
|return|short|sizeof|struct|switch|void|while)

INT             0[xX][0-9a-fA-F]+|[1-9][0-9]*|0[0-7]+|0\b

ID              [A-Za-z_][A-Za-z0-9_]{0,32}

FLOAT               [0-9]+((\.[0-9]+)([eE][+-]?[0-9]*)?|([eE][+-]?[0-
9]*)+)

STR             ["]([^"\\\n]|\\.|\\\n)*["]

COM_S           \/\/.*$

COM_M           ^(\/\*).*(\*\/)$

ASSIGN          (.+){OPERATOR}$|{OPERATOR}(.+)|(.+){OPERATOR}(.+)

CATCHALL        [^ {DELIMITER}\n]+
```

```
%%
 /*Κανόνες για την αναγνώριση των tokens*/
;               {}
{DELIMITER}     {}
{OPERATOR}      { return TK_OPERATOR; }
{KEYWORD}       { return TK_KEYWORD; }
{INT}           { return TK_INT; }
{ID}            { return TK_ID; }
{FLOAT}         { return TK_FLOAT; }
{STR}           { return TK_STR; }
{COM_S}         {}
{COM_M}         {}
{ASSIGN}        { return TK_ASSIGN; }
\n              { line++; printf("\n"); }
{CATCHALL}      { return TK_CATCHALL; }
<<EOF>>         { printf("#END-OF-FILE#\n"); exit(0); }


%%
 /* Πίνακας με τα ονόματα των tokens*/
char     *tk[]     =     {"DELIMITER","INT",     "ID",     "FLOAT",
"STR","KEYWORD","OPERATOR","ASSIGN","CATCHALL"};
 /*Είσοδος του κώδικα σε C*/
int main(int argc, char **argv){
        int token;
        /*Έλεγχος ορισμάτων εντολής*/
        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;
                }
        }
        /*Ανάγνωση tokens από τον lexer*/
        while( (token=yylex()) >= 0)
      {
                /*Εμφάνιση πληροφοριών σχετικά με τα tokens*/
                if(token==TK_STR)
                {
                fprintf(yyout, "\tLine=%d, token=%s, value=%s\n",
line, tk[token], yytext); /*We do this to avoid double quotes.*/
                continue;
                }
                if(token==TK_CATCHALL)
                {
                fprintf(yyout, "\tLine=%d, UNKNOWN TOKEN,
value=\"%s\"\n", line, yytext);
```

```
                continue;
                }
            fprintf(yyout,  "\tLine=%d,  token=%s,  value=\"%s\"\n",
line, tk[token], yytext);
        }
        return 0;
}
```

Σχολιασμός κώδικα

Αυτός ο κώδικας είναι ένα πρόγραμμα lexer που διαβάζει ένα αρχείο κώδικα σε μία γλώσσα προγραμματισμού και αναγνωρίζει διάφορα tokens, όπως δεσμευμένες λέξεις, αριθμούς, αναγνωριστικά, συμβολοσειρές, τελεστές, σχόλια και κενά διαστήματα.

Ο κώδικας χρησιμοποιεί το Lex/Flex, ένα εργαλείο που διευκολύνει την ανάλυση των αρχείων κειμένου. Οι βασικές του συνιστώσες είναι: Ορισμός regular expressions για τα tokens: Ο κώδικας ξεκινά με τη δήλωση regular expressions για διάφορα είδη tokens όπως διαχωριστικά, τελεστές, λέξεις-κλειδιά, αριθμούς, αναγνωριστικά, σχόλια και κενά διαστήματα παραβλέποντας τα δύο τελευταία. Ορισμός κανόνων αναγνώρισης tokens: Στη συνέχεια, ορίζονται οι κανόνες για την αναγνώριση των διάφορων tokens, χρησιμοποιώντας τις παραπάνω regular expressions. Ορισμός επεξεργαστή: Τέλος, ορίζεται ο επεξεργαστής, ο οποίος διαβάζει τον κώδικα εισόδου, εφαρμόζει τους κανόνες αναγνώρισης tokens και εκτυπώνει τα tokens και τις πληροφορίες τους σε ένα αρχείο κειμένου.

Κώδικας token.h

```
#define TK_INT 1

#define TK_ID 2

#define TK_FLOAT 3

#define TK_STR 4

#define TK_KEYWORD 5

#define TK_OPERATOR 6

#define TK_ASSIGN 7

#define TK_CATCHALL 8
```

Αντιπαραβολή Εισόδου – Εξόδου

input.txt

```
1234

50

115

3.14

10.0

05

09

5a


89e5

0.0001

3.14e-10

0e0

.67


0xac784

00
```

```
063

004

0Xff4

0xxaf3

00xaf3


backitup

BeepBeep

_test

test_3

45test

Test_t6

space man

ignore      whitespace


//Comment

/Comment

/*Better Comment*/

/*Better Comment*


""

" "

"   "

"Test\n"

"Hello world"

"Mark said, \"Boo!\""

"hello !"

"hello !


#unknown ?2 ? ?hello ?world
```

```
a+b
a+5
5+a
a+5;
a-b
a-5
a-5;
a*b
a*5
5*a
a*5;
a/b
a/5
5/a
a/5;
a%b
a%6
a%4;
a=8;
a="test"
a=false
a=6
a+=2
a+=b
a+=b;
a-=9
a-=b
a-=b;
```

```
a*=2
a*=b
a*=b;
a/=2
a/=b
a/=b;
!a
a=!b;
hex || "String"
hex ||| "String"
hex && "String"
hex & "String"
hex != "String"
hex !== "String"
a++
b--
a++;
b--;
a>8
a>9;
a>h
a<8
a<9;
a<h
a<=8
a<=9;
a<=h
a>=8
a>=9;
a>=h
```

```
&a

a===6

5-------a


break

case

func

const

continue

do

double

else

float

for

if

int

long

return

short

sizeof

struct

switch

void

while
```

output.txt

```
Line=1, token=INT, value="1234"
Line=2, token=INT, value="50"
Line=3, token=INT, value="115"
Line=4, token=FLOAT, value="3.14"
Line=5, token=FLOAT, value="10.0"
Line=6, token=INT, value="05"
Line=7, UNKNOWN TOKEN, value="09"
Line=8, UNKNOWN TOKEN, value="5a"
Line=10, token=FLOAT, value="89e5"
Line=11, token=FLOAT, value="0.0001"
Line=12, token=FLOAT, value="3.14e-10"
Line=13, token=FLOAT, value="0e0"
Line=14, UNKNOWN TOKEN, value=".67"
Line=16, token=INT, value="0xac784"
Line=17, token=INT, value="00"
Line=18, token=INT, value="063"
Line=19, token=INT, value="004"
Line=20, token=INT, value="0Xff4"
Line=21, UNKNOWN TOKEN, value="0xxaf3"
Line=22, UNKNOWN TOKEN, value="00xaf3"
Line=24, token=ID, value="backitup"
Line=25, token=ID, value="BeepBeep"
Line=26, token=ID, value="_test"
Line=27, token=ID, value="test_3"
Line=28, UNKNOWN TOKEN, value="45test"
Line=29, token=ID, value="Test_t6"
Line=30, token=ID, value="space"
Line=30, token=ID, value="man"
Line=31, UNKNOWN TOKEN, value="ignore     whitespace"
```

```
Line=34, token=ASSIGN, value="/Comment"
Line=36, token=ASSIGN, value="/*Better Comment*"
Line=38, token=STR, value=""
Line=39, token=STR, value=" "
Line=40, token=STR, value="   "
Line=41, token=STR, value="Test\n"
Line=42, token=STR, value="Hello world"
Line=43, token=STR, value="Mark said, \"Boo!\""
Line=44, token=STR, value="hello !"
Line=45, UNKNOWN TOKEN, value=""hello"
Line=45, token=OPERATOR, value="!"
Line=47, UNKNOWN TOKEN, value="#unknown"
Line=47, UNKNOWN TOKEN, value="?2"
Line=47, UNKNOWN TOKEN, value="?"
Line=47, UNKNOWN TOKEN, value="?hello"
Line=47, UNKNOWN TOKEN, value="?world"
Line=50, token=ASSIGN, value="a+b"
Line=51, token=ASSIGN, value="a+5"
Line=52, token=ASSIGN, value="5+a"
Line=53, token=ASSIGN, value="a+5;"
Line=54, token=ASSIGN, value="a-b"
Line=55, token=ASSIGN, value="a-5"
Line=56, token=ASSIGN, value="a-5;"
Line=57, token=ASSIGN, value="a*b"
Line=58, token=ASSIGN, value="a*5"
Line=59, token=ASSIGN, value="5*a"
Line=60, token=ASSIGN, value="a*5;"
Line=61, token=ASSIGN, value="a/b"
Line=62, token=ASSIGN, value="a/5"
Line=63, token=ASSIGN, value="5/a"
```

```
Line=64, token=ASSIGN, value="a/5;"
Line=65, token=ASSIGN, value="a%b"
Line=66, token=ASSIGN, value="a%6"
Line=67, token=ASSIGN, value="a%4;"
Line=68, token=ASSIGN, value="a=8;"
Line=69, token=ASSIGN, value="a="test""
Line=70, token=ASSIGN, value="a=false"
Line=71, token=ASSIGN, value="a=6"
Line=72, token=ASSIGN, value="a+=2"
Line=73, token=ASSIGN, value="a+=b"
Line=74, token=ASSIGN, value="a+=b;"
Line=75, token=ASSIGN, value="a-=9"
Line=76, token=ASSIGN, value="a-=b"
Line=77, token=ASSIGN, value="a-=b;"
Line=78, token=ASSIGN, value="a*=2"
Line=79, token=ASSIGN, value="a*=b"
Line=80, token=ASSIGN, value="a*=b;"
Line=81, token=ASSIGN, value="a/=2"
Line=82, token=ASSIGN, value="a/=b"
Line=83, token=ASSIGN, value="a/=b;"
Line=84, token=ASSIGN, value="!a"
Line=85, token=ASSIGN, value="a=!b;"
Line=86, token=ASSIGN, value="a===6"
Line=87, token=ASSIGN, value="5-------a"
Line=88, token=ASSIGN, value="hex || "String""
Line=89, token=ASSIGN, value="hex ||| "String""
Line=90, token=ASSIGN, value="hex && "String""
Line=91, token=ASSIGN, value="hex & "String""
Line=92, token=ASSIGN, value="hex != "String""
Line=93, token=ASSIGN, value="hex !== "String""
```

```
Line=94, token=ASSIGN, value="a++"
Line=95, token=ASSIGN, value="b--"
Line=96, token=ASSIGN, value="a++;"
Line=97, token=ASSIGN, value="b--;"
Line=98, token=ASSIGN, value="a>8"
Line=99, token=ASSIGN, value="a>9;"
Line=100, token=ASSIGN, value="a>h"
Line=101, token=ASSIGN, value="a<8"
Line=102, token=ASSIGN, value="a<9;"
Line=103, token=ASSIGN, value="a<h"
Line=104, token=ASSIGN, value="a<=8"
Line=105, token=ASSIGN, value="a<=9;"
Line=106, token=ASSIGN, value="a<=h"
Line=107, token=ASSIGN, value="a>=8"
Line=108, token=ASSIGN, value="a>=9;"
Line=109, token=ASSIGN, value="a>=h"
Line=110, token=ASSIGN, value="&a"
Line=112, token=KEYWORD, value="break"
Line=113, token=KEYWORD, value="case"
Line=114, token=KEYWORD, value="func"
Line=115, token=KEYWORD, value="const"
Line=116, token=KEYWORD, value="continue"
Line=117, token=KEYWORD, value="do"
Line=118, token=KEYWORD, value="double"
Line=119, token=KEYWORD, value="else"
Line=120, token=KEYWORD, value="float"
Line=121, token=KEYWORD, value="for"
Line=122, token=KEYWORD, value="if"
Line=123, token=KEYWORD, value="int"
Line=124, token=KEYWORD, value="long"
```

```
Line=125, token=KEYWORD, value="return"
Line=126, token=KEYWORD, value="short"
Line=127, token=KEYWORD, value="sizeof"
Line=128, token=KEYWORD, value="struct"
Line=129, token=KEYWORD, value="switch"
Line=130, token=KEYWORD, value="void"
Line=131, token=KEYWORD, value="while"
```

Σχολιασμός αποτελεσμάτων των δοκιμών

Από τη γραμμή **1-6** του αρχείου input.txt δίνονται έγκυροι ακέραιοι και πραγματικοί αριθμοί οι οποίοι αναγνωρίζονται από τον κώδικα. Στις γραμμές **7-8** δίνονται αρχικά ένας μη αποδεκτός οκταδικός αριθμός (09) και μετά ένας μη αποδεκτός τύπος αριθμού(5α) οι οποίοι αναγνωρίζονται από τον κώδικα και εμφανίζονται ως unknown token. Στις γραμμές **10-13** δίνονται διαφορετικές σωστές μορφές πραγματικών αριθμών οι οποίες γίνονται όλες ορθώς αποδεκτές. Στη γραμμή **14** δίνεται ένας μη ολοκληρωμένος τύπος πραγματικού αριθμού όπου υπολείπεται το ακέραιο μέρος του και γι' αυτό δεν αναγνωρίζεται και εμφανίζεται ως unknown token. Στις γραμμές **16-20** δίνονται μερικά παραδείγματα οκταδικών και δεκαεξαδικών αριθμών τα οποία είναι σωστά και γι' αυτό αναγνωρίζονται. Στις γραμμές **21-22** δίνονται δύο παραδείγματα μη αποδεκτών δεκαεξαδικών αριθμών (0xxaf3, 00xaf3) τα οποία εμφανίζονται ως unknown tokens. Στις γραμμές **24-27** και **29-30** δίνονται μερικά ορθά παραδείγματα αναγνωριστικών τα οποία γίνονται αποδεκτά ως id. Στη γραμμή **28** δίνεται ένα μη ορθό παράδειγμα αναγνωριστικού (45test) το οποίο δεν αναγνωρίζεται και εμφανίζεται ως unknown token. Στη γραμμή **31** δίνονται δύο αναγνωριστικά χωρισμένα με Tab τα οποία κακώς δεν γίνονται αποδεκτά. Στις γραμμές **33** και **35** δίνονται δύο ορθοί τύποι σχολίων μονής γραμμής και πολλαπλών γραμμών οι οποίοι αναγνωρίζονται και αγνοούνται. Στις γραμμές **34** και **36** δίνονται παραδείγματα ημιτελών σχολίων τα οποία αναγνωρίζονται και εμφανίζονται. Στις γραμμές **38-44** δίνονται ορθά παραδείγματα λεκτικών-κυριολεκτικών τα οποία αναγνωρίζονται και εμφανίζονται ως str. Στη γραμμή **45** δίνεται μία ημιτελής έκφραση λεκτικού-κυριολεκτικού ("hello !) το οποίο δεν γίνεται αποδεκτό το ("hello)και εμφανίζεται ως unknown token ενώ το (!) γίνεται αποδεκτό και εμφανίζεται ως operator. Στη γραμμή **47** δίνεται μία έκφραση με πολλά σφάλματα (#unknown ?2 ? ? hello ?world) σε αυτή την περίπτωση δίνονται πέντε λανθασμένα ορίσματα τα οποία διαχωρίζονται και εμφανίζονται ξεχωριστά ως unknown tokens. Στις γραμμές **50-108** δίνονται ορθά παραδείγματα χρήσης τελεστών τα οποία αναγνωρίζονται ως εκφράσεις και εμφανίζονται ως assign. Στις γραμμές **109-110** δίνονται δύο

παραδείγματα λανθασμένης χρήσης τελεστών (a===6, 5-------a) τα οποία κακώς αναγνωρίζεται ως ορθά. Στις γραμμές **112-131** δίνονται παραδείγματα λέξεων κλειδιών τα οποία αναγνωρίζονται και εμφανίζονται ως keyword. Στις γραμμές **9, 15, 23, 32,3 7, 46, 48, 49, 111** και **132** δίνονται κενά διαστήματα spaces ή TABS τα οποία αναγνωρίζονται και αγνοούνται όπως και στις περιπτώσεις όπου οι εκφράσεις τελειώνουν με (;) (αγνοείται το ; καθώς συμβολίζει το τέλος της έκφρασης).

Αναφορά ελλείψεων

Αρχικά ο κώδικας κάνει compile και εκτυπώνει σχεδόν στο έπακρον ορθά αποτελέσματα. Παρόλα αυτά σε κάποιες ειδικές περιπτώσεις τα αποτελέσματα δεν είναι τα αναμενόμενα. Στη γραμμή 31 δίνονται δύο αναγνωριστικά χωρισμένα με Tab τα οποία θα έπρεπε να παραβλέπεται το Tab όπως θα έπρεπε με το space και να λαμβάνει τα αναγνωριστικά ως δύο ξεχωριστά. Στις γραμμές 109-110 δέχεται ως ορθή έκφραση τελεστών, εκφράσεις με πολλούς ίδιους ή διαφορετικούς τελεστές ενώ δεν θα έπρεπε.

Κατανομή Εργασιών

Υπεύθυνος εργασίας Α-2:	Φράγκος Μαρίνος
Κώδικας:	Διάννης Ιωάννης, Βροχάρης Αντώνιος
Testing:	Φράγκος Μαρίνος
Σύνταξη pdf:	Φράγκος Μαρίνος, Φάσσου Κοντοδημάκη Ιφιγένεια Γεωργία, Φριλίγκος Γρηγόριος