ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΑΘΗΝΑΣ

ATHENS UNIVERSITY OF APPLIED SCIENCES

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΡΙΚΗΣ

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΠΑΡΙΣ ΛΙΖΑΙ

Πίνακας περιεχομένων

ΕΙΣΑΓΩΓΗ

Η λεκτική ανάλυση είναι η πρώτη φάση ενός μεταγλωττιστή.
Η κύρια λειτουργία του είναι να διαβάσει τους χαρακτήρες εισόδου, να τους χωρίσει σε λεκτήματα και να παρουσιάσει σαν έξοδο μια ακολουθία από σύμβολα της γραμματικής που αναπαριστά την γλώσσα προγραμματισμού στην οποία είναι γραμμένο το πρόγραμμα, τα οποία αντιστοιχούν στα λεκτήματα εισόδου. Εκτός από την κύρια αυτή λειτουργία, ο λεκτικός αναλυτής είναι συχνά επιφορτισμένος και με κάποιες δευτερεύουσες, πλην όμως σημαντικές λειτουργίες:

- την αφαίρεση των σχολίων
- των κενών που εμφανίζονται (κενοί χαρακτήρες, αλλαγές γραμμής, κ.λπ.)
- συσχέτιση μηνυμάτων λάθους που τυχόν παράγει ο μεταγλωττιστής με το πρόγραμμα εισόδου.

Όπως είπαμε και προηγουμένως, οι λεκτικοί αναλυτές είναι ουσιαστικά πεπερασμένα αυτόματα, τα οποία χρησιμοποιώντας κανονικές εκφράσεις προσπαθούν να ταιριάξουν κάθε σύμβολο εισόδου με κάποιους κανόνες οι οποίοι περιγράφουν την εκάστοτε γλώσσα προγραμματισμού για την οποία επιθυμούμε να φτιάξουμε τον λεκτικό αναλυτή. Για την κατασκευή αυτών των αυτομάτων, στην πράξη χρησιμοποιούμε γεννήτορες λεκτικών αναλυτών, δηλαδή προγράμματα στα οποία δίνουμε την περιγραφή της γραμματικής μας και παίρνουμε στην έξοδο το επιθυμητό αυτόματο.

ΚΕΦΑΛΑΙΟ 1

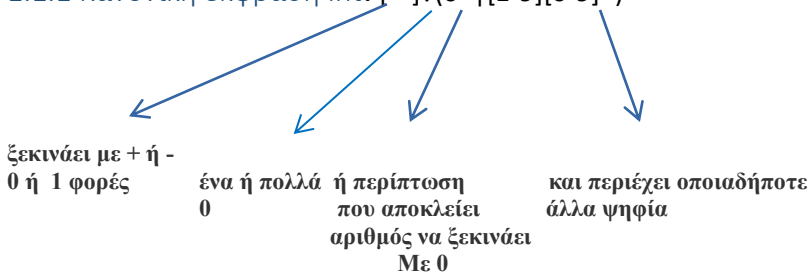
1.1 ΑΚΕΡΑΙΟΙ

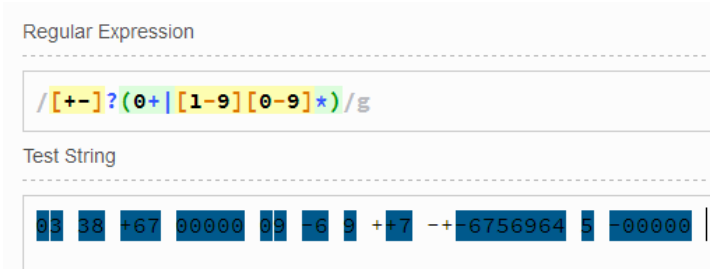
1.1.0 Περιγραφή

Ένα ακέραιος αποτελείται από ένα ή περισσότερα ψηφία 0-9. Ένας ακέραιος με μήκος 2 ή μεγαλύτερο δεν μπορεί να αρχίζει από 0.

- Παραδείγματα ακέραιων είναι τα 0, 2, +34, -1000 κλπ.

1.1.1 Κανονική έκφραση Int: [+-]?(0+|[1-9][0-9]*)





Σωστά: 0 , 3 +67, 00000, 9, -9
Λάθος: 09, +, -+

1.1.2 ΛΑΘΗ

int_error1 (0+[1-9]+)
int_error2 [0-9]+[^0-9\. \t\r\n]{no_whites}*
{prefix}({int_error1}|{int_error2})

Δύο είναι τα λάθη που μπορεί να εντοπισθούν στους ακεραίους. Πιο συγκεκριμένα ένας ακέραιος δεν μπορεί να ξεκινάει με μηδέν όπως 09. Το δεύτερο λάθος εντοπίζεται όταν ένας ακέραιος ακολουθείται από οτιδήποτε δεν είναι delimiter πχ 5+ 0λ 45^{ε}

Το \. υπάρχει επειδή όταν εντοπίζεται λάθος τύπου float θα πρέπει να το διαχειρίζεται η αντίστοιχη έκφραση.

ΠΑΡΑΔΕΙΓΜΑ

IINPUT

```
+2
13231
000
-0
02
```

OUTPUT

```
INTEGER  FOUND      Line=1      +2
INTEGER  FOUND      Line=2      13231
INTEGER  FOUND      Line=3      000
INTEGER  FOUND      Line=4      -0
INTEGER  NOT VALID IN LINE=5 READ:02
```

1.2 ΕΚΘΕΤΙΚΟΙ

1.2.0 Περιγραφή

Ένας αριθμός κινητής υποδιαστολής αποτελείται από ένα ακέραιο μέρος (ακέραιος αριθμός) και ένα δεκαδικό μέρος (ακολουθία από ψηφία 0-9) που διαχωρίζονται από μια τελεία. Ένα από τα δυο μέρη μπορεί να λείπει αλλά όχι και τα δυο μαζί.

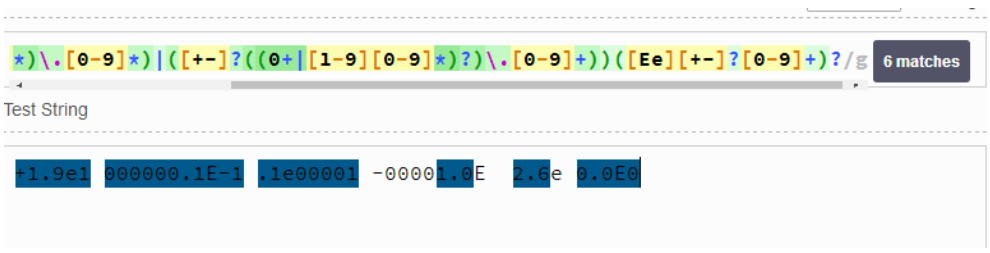
- Παραδείγματα αριθμών κινητής υποδιαστολής (ή floats για συντομία) είναι οι 2.0, 1.23, 100., +.0123 και -52.3E-4. Το σύμβολο E δηλώνει δύναμη του 10. Σε αυτή την περίπτωση, το -52.3E-4 σημαίνει -52.3 * 10-4 και αρμόδιος να το γνωρίζει και να το διαχειριστεί είναι ο σημασιολογικός αναλυτής του μεταγλωττιστή.

1.2.1Κανονική Έκφραση Float:

([+-]?(0+|[1-9][0-9]*)\.[0-9]*|[+-]?\.[0-9]+)([Ee][+-]?[0-9]+)?

Αναζήτηση για δεκαδικό

Αναζήτηση για E ή e (εάν υπάρχει), μπορεί να ακολουθείται από πρόσημο και ένα ή παραπάνω ψηφία.



Σωστά: +1.1, -21.34, 1.9e-2. 1.9E9
Λάθος: .e-2 , 2. , -2. , +.E-5

1.2.2 ΛΑΘΗ

float_error1 {prefix}{int_error1}\.{no_whites}* όπου int_error1: (0+[1-9]+)

float_error2 ({integer}\.[0-9]*[Ee])|({prefix}\.[0-9]*[Ee])

float_error3 (({integer}\.[0-9]*)|({prefix}\.[0-9]+))[^0-9Ee \t\r\n]{no_whites}*

({float_error1})|({float_error2})|({float_error3})

Καθώς τα λάθη τύπου float είναι πάρα πολλά και δεν θα ήταν δυνατό να εντοπισθούν όλα τα πιθανά σφάλματα θεωρείται σκόπιμο να εστιάσουμε τη προσοχή μας σε τρεις μεγάλες κατηγορίες.

1. Σωστό ακέραιο~ Λάθος δεκαδικό → float_error2,float_error3
2. Λάθος ακέραιο ~Σωστό δεκαδικό → float_error1
3. Λάθος ακέραιο ~Λάθος δεκαδικό → float_error1

Το float_error1 καλείται όταν ένας float ξεκινάει απο 0 και έπειτα ακολουθεί ένα τουλάχιστον ψηφίο πχ 03.4 -006.2e-3

Το float_error2 καλείται όταν ένας float τερματίζει με E ή e δίχως να ακολουθεί κάποιο άλλο ψηφίο πχ 3.3e +45.e

Το float_error3 καλείται όταν ένας ακέραιος δεν ακολουθείται από delimiter πχ 3.1+ +.45. -

ΠΑΡΑΔΕΙΓΜΑ

INPUT

```
+2.2
+2.2e-2
5.1
6.2E+1
+2.
2.
+.2e+55
+.2
02.
2.e-2
2.a
```

OUTPUT

```
FLOAT     FOUND     Line=1      +2.2
FLOAT     FOUND     Line=2      +2.2e-2
FLOAT     FOUND     Line=3      5.1
FLOAT     FOUND     Line=4      6.2E+1
FLOAT     FOUND     Line=5      +2.
FLOAT     FOUND     Line=6      2.
FLOAT     FOUND     Line=7      +.2e+55
FLOAT     FOUND     Line=8      +.2
FLOAT NOT VALID IN LINE=9 READ:02.
FLOAT     FOUND     Line=10     2.e-2
FLOAT NOT VALID IN LINE=11 READ:2.a
```

1.3 ΜΕΤΑΒΛΗΤΕΣ

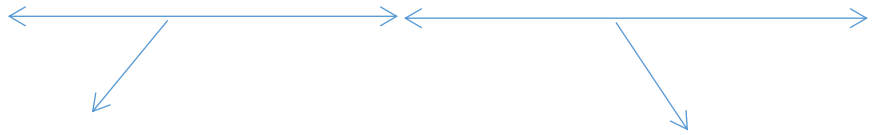
1.3.0 Πειργραφή

Οι μεταβλητές αποτελούν συμβολικά ονόματα θέσεων μνήμης. Το όνομα μιας μεταβλητής μπορεί να περιλαμβάνει λατινικούς χαρακτήρες a .. Z, A .. Z αριθμούς 0.. 9 και _ Το όνομα μιας μεταβλητής δεν μπορεί να αρχίζει όμως με αριθμητικό ψηφίο. Στο όνομα ενός αναγνωριστικού διακρίνονται τα πεζά από τα κεφαλαία (case-sensitive). Μπορείτε να χρησιμοποιείτε ελληνικές λέξεις ως ονόματα αναγνωριστικών.

- Αποδεκτά ονόματα μεταβλητών: X, αβ, abcdef, A12345, X1Yssss, my_Name, ο_όνομά_μου, _abcde, a_bc_123, _12345, _12345_ .

1.3.1Κανονική Έκφραση Variable:

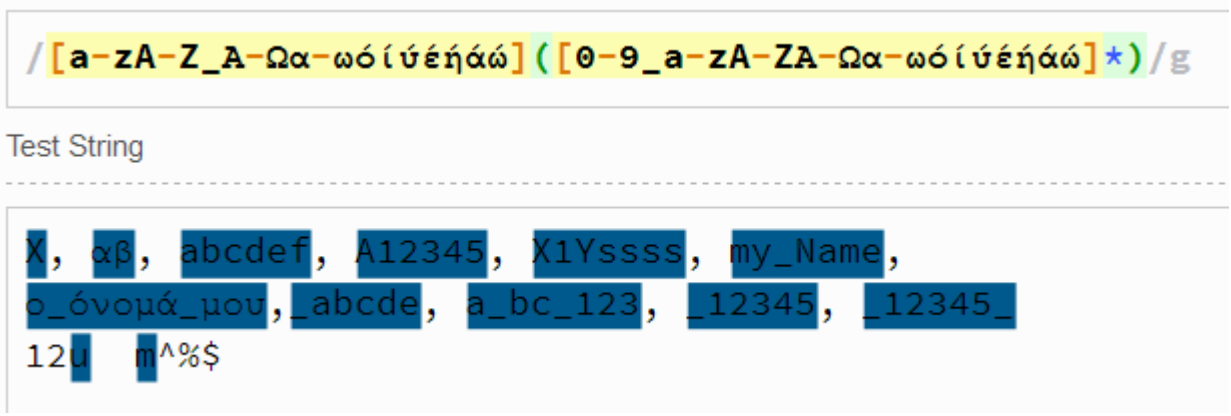
[a-zA-Z_A-Ωα-ωόίύέήάώ]([0-9_a-zA-ZA-Ωα-ωόίύέήάώ]*)



Οποιοδήποτε χαρακτήρας φορές

Οποιοσδήποτε χαρακτήρας από 0 έως πολλές

1 φορά εκτός από ψηφίο



1.3.2 ΛΑΘΗ

letters [a-zA-Z_A-Ωα-ωόίύέήάώ]
var_error {letters}({letters}+[0-9])*[^a-zA-Z_A-Ωα-ωόίύέήάώ0-9 \t\r\n]{no_whites}*
όπου letters: [a-zA-Z_A-Ωα-ωόίύέήάώ]

Μια μεταβλητή μπορεί να ξεκινάει από με οποιοδήποτε γράμμα πεζό ή κεφαλαίο (και ελληνικό) ή με τον χαρακτήρα [_] και στη συνέχεια με οποιοδήποτε αλφαριθμητικό.
Αν όμως εντοπισθεί σε σημείο του λεξήματος μη αλφαριθμητικός χαρακτήρας τότε το λέξημα δεν αναγνωρίζεται και χαρακτηρίζεται λάθος. Π.χ. : cv6$ c?

ΠΑΡΑΔΕΙΓΜΑ

INPUT

```
var1
vaasd_sad
cv6$
c?
_6λφγφφσγδφγ
```

OUTPUT

```
VARIABLE FOUND     Line=1      var1
VARIABLE FOUND     Line=2      vaasd_sad
VARIABLE NOT VALID IN LINE=3 READ:cv6$
VARIABLE NOT VALID IN LINE=4 READ:c?
VARIABLE FOUND     Line=5      _6λφγφφσγδφγ
```

1.4 ΣΧΟΛΙΑ

1.4.0 Περιγραφή

Τα σχόλια αρχίζουν με το σύμβολο # και συνεχίζουν μέχρι το τέλος της γραμμής.

- #This is a single line column

Αν επιθυμούμε σχόλια που να επεκτείνονται σε πολλές γραμμές θα πρέπει να τα περικλείσουμε μεταξύ δυο τριάδων διπλών εισαγωγικών """ (3 διπλά εισαγωγικά στην αρχή και το τέλος του σχολίου).

1.4.1Κανονική Έκφραση:

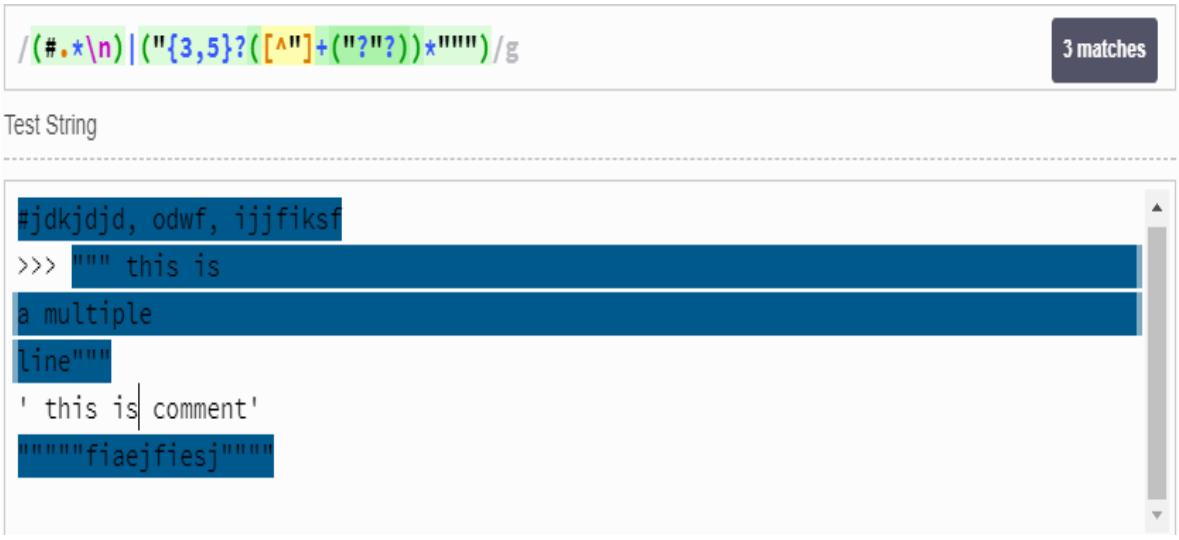
*ΠΑΡΑΤΗΡΗΣΗ(ΠΑΤΗΣΤΕ ΠΑΝΩ)

(#.*\n)|("{3,5}?([^"]+("?"?))*""")

Ψάχνει τα λιγότερα που μπορεί να βρει

Ψάχνει να βρει τον χαρακτήρα (#.)

Το string μέσα στα """...""" που ορίζουν τα σχόλια μία φορά.



1.4.2 ΛΑΘΗ

Σε αυτή την κατηγορία παρατηρούμε μια ‘ποικιλία’ από λάθη. Αρχικά υπάρχουν μερικές παραξενιές από μέρους του flex.

Το ειδικό σύμβολο ?(lazy) δεν αναγνωρίζεται άρα είναι αναγκαία η μετατροπή της κανονικής έκφρασης.

({quotes} (\"?\"?[^"]+)*{quotes}\"?\"?) όπου quotes: """

comm_error: ({quotes}(\"?\"?[^"]+)*{quotes}\"?\"?){no_whites}+

Το μόνο λάθος που παρατηρείται από γραμματική άποψη είναι όταν έχει ολοκληρωθεί το σχόλιο και αντί για delimiter ακολουθήσει οτιδήποτε άλλο.

Π.χ. """σχόλιο"""άσχετο,

""""4

2""1"

3"""λάθος

ΠΑΡΑΔΕΙΓΜΑ

INPUT

```
"""correct"""
#comment
"""33"""
"""34"""
"""43"""
"""45"""""
"""""53"""
"""""55"""""

"""31"
""23"""

"""32""2

""1"3"""

"""3
1"
3"""

"""wrong""wrong
"""31"
""23""again

"""32"""""ll
```

OUTPUT

```
Illegal  identifier (comment) at  line: 41       """wrong""wrong
Illegal  identifier (comment) at  line: 43       """31"
""23""again
Illegal  identifier (comment) at  line: 49       """32"""""ll
correct 11      wrong 3
```

1.5 ΤΕΛΕΣΤΕΣ

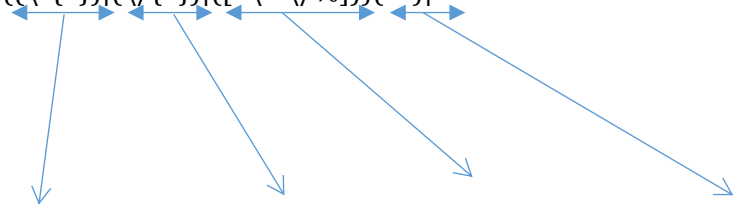
1.5.0 Περιγραφή

Οι αριθμητικοί τελεστές χρησιμοποιούνται για πράξεις μεταξύ δυο αριθμών (τελεστέων):

\+ →Προσθήκη 2 τελεστέων ή συν ως πρόσημο x + y, +2
\-→ Αφαίρεση δεύτερου τελεστέου από τον πρώτον ή μείον ως πρόσημο x – y, -2
*→ Πολλαπλασιασμός 2 τελεστέων x * y
/→ Διαίρεση αριστερού τελεστέου από τον δεξιό (πάντα δίνει αποτέλεσμα float) x / y
% → Υπόλοιπο – υπόλοιπο διαίρεσης αριστερού τελεστέου από τον δεξιό x % y (remainder of x/y)
// → Βάση διαίρεσης – ακέραιο μέρος διαίρεσης αριστερού τελεστέου από τον δεξιό x // y
** Εκθετικός – αριστερός τελεστέος υψωμένος στη δεξιά δύναμη x**y (x to the power y)

1.5.1 Κανονική Έκφραση Operator:

([*{2}]|(\/{2})|([+\-*\/%]))(=?)|=

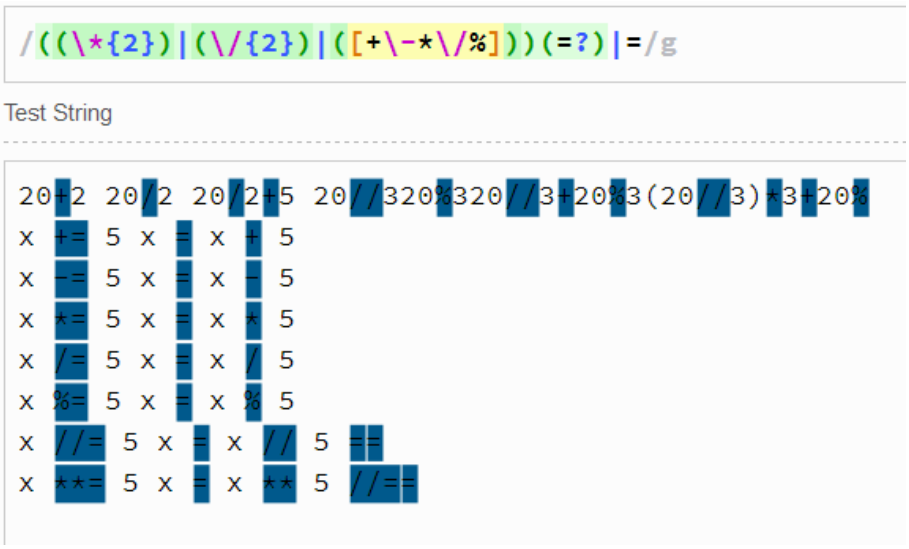


Τα σύμβολα ακολουθούνται * ακριβώς 2 φορές

Τα σύμβολα /ακριβώς 2 φορές

Όλοι οι γνωστοί τελεστές

Στην περίπτωση που από = (πχ +=)



1.5.2 ΛΑΘΗ

```
sym             [-+\*\/%]
op_error1       [^= \t\r\n]{no_whites}*
op_error2       {sym}{op_error1}            εδώ είναι τα λάθη όπως -κ *+

(\*\*{op_error1})|(\/\/{op_error1})|{op_error2}   εδώ προστίθενται λάθη στις
                                                  περιπτώσεις ** // όπως **+ //λ
```

Το σφάλμα προκύπτει όταν ένα σύμβολο δεν ακολουθείται από [=] ή delimiter
Έτσι προκύπτουν λάθη όπως +k **l %+ //*

ΠΑΡΑΔΕΙΓΜΑ

INPUT

```
+
-
/
*
=
%
%=
/=
-=
*=
+k
**]
%+
//*
```

OUTPUT

```
OPERATOR FOUND      Line=1      +
OPERATOR FOUND      Line=2      -
OPERATOR FOUND      Line=3      /
OPERATOR FOUND      Line=4      *
OPERATOR FOUND      Line=5      =
OPERATOR FOUND      Line=6      %
OPERATOR FOUND      Line=7      %=
OPERATOR FOUND      Line=8      /=
OPERATOR FOUND      Line=9      -=
OPERATOR FOUND      Line=10     *=
OPERATOR NOT VALID IN LINE=11 READ:+k
OPERATOR NOT VALID IN LINE=12 READ:**]
OPERATOR NOT VALID IN LINE=13 READ:%+
OPERATOR NOT VALID IN LINE=14 READ://*
```

1.6 ΣΥΜΒΟΛΟΣΕΙΡΕΣ

1.6.0 Περιγραφή

Μια συμβολοσειρά είναι μια ακολουθία από χαρακτήρες μέσα από το αλφάβητο της γλώσσας που περικλείονται μεταξύ μονών ή διπλών εισαγωγικών.

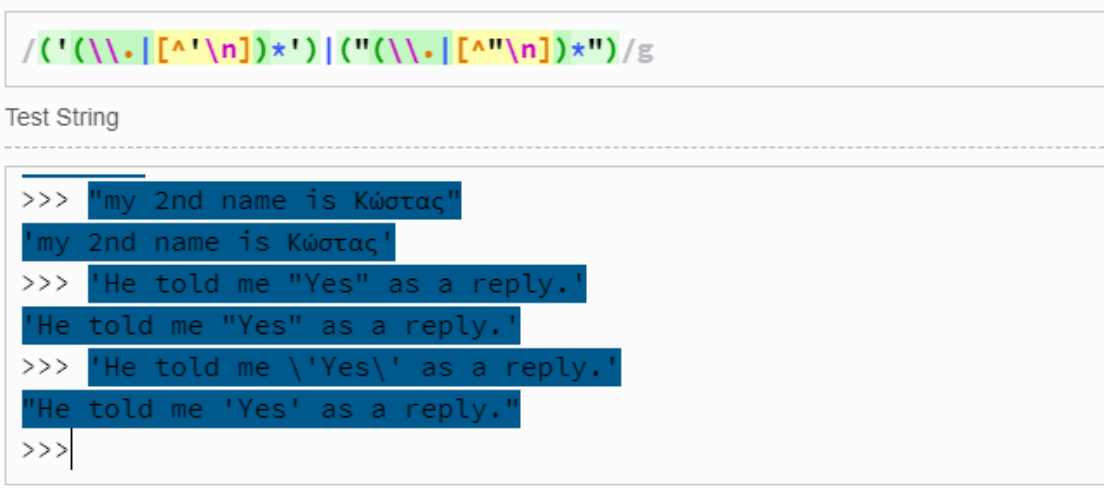
- Παραδείγματα συμβολοσειρών: 'Δημήτρης'
  "My 2nd name is Δημήτρης"
  'He told me "Yes" as a reply.'

Μια συμβολοσειρά μπορεί να επεκτείνεται σε περισσότερες της μιας γραμμές με τη χρήση ενός χαρακτήρα διαφυγής - \n στην αρχή της νέας γραμμής.
Ένας ακόμα χρήσιμος χαρακτήρας διαφυγής είναι ο στηλοθέτης (Tab) \t.

1.6.1 Κανονική έκφραση String: ('(\\.|[^'\n])*')|("(\\.|[^"\n])*")

Εντοπίζει string τα οποία μπορούν να περιέχουν οποιοδήποτε χαρακτήρα εκτός των \n "αλλά στην περίπτωση που εντοπίζει \ μπορεί να ακολουθεί οτιδήποτε μεταξύ άλλων και το "

εντοπιζει string τα οποία μπορούν να περιέχουν οποιοδήποτε χαρακτήρα εκτός των \n ‘αλλά στην περίπτωση που εντοπίζει \ μπορεί να ακολουθεί οτιδήποτε μεταξύ άλλων και το



1.6.2 ΛΑΘΗ

string1 '(\\.|[^'\n\r])*

string2 \"(\\.|[^"\n\r])*

({string1}[\r\n])|({string2}[\r\n])

Ένα σφάλμα τύπου string εντοπίζεται όταν ένα string δεν ολοκληρώνεται αλλά σε κάποιο σημείο βρεθεί χαρακτήρας νέας γραμμής

ΠΑΡΑΔΕΙΓΜΑ

INPUT

```
"this is a string"
"this \"is\" a string"

"this is an wrong string

'this is an another one string'

'this is an another \'one\' string'
'this is an wrong string
```

OUTPUT

```
STRING   FOUND     Line=1      "this is a string"
STRING   FOUND     Line=3      "this \"is\" a string"STRING NOT VALID IN LINE=6 READ:"this is an wrong string

STRING   FOUND     Line=9      'this is an another one string'
STRING   FOUND     Line=12     'this is an another \'one\' string'STRING NOT VALID IN LINE=14 READ:'this is an wrong string
```

ΚΕΦΑΛΑΙΟ 2

ΠΑΡΑΤΗΡΗΣΕΙΣ:

<<EOF>>:

Η flex μπορεί να εντοπίσει το τέλος αρχείου και έτσι δίνεται η δυνατότητα διαχείρισης.

Εμείς θελήσαμε να εμφανίζει κάποια στατιστικά στοιχεία και μετά να τερματίζει το πρόγραμμα.

ΕΠΕΞΗΓΗΣΕΙΣ

delimiter [\t\r\n]+ ⟶ αναγνωρίζει όλους του άσπρους χαρακτήρες και τους προσπερνάει [*αντίστοιχο της avoidblanks()*]

no_whites [^ \t\r\n] ⟶ αναγνωρίζει μη λευκό χαρακτήρα χρησιμοποιείται εκτενός στον εντοπισμό λαθών

Σημείωση:

<error>{delimiter} ⟶ *Σε περίπτωση λάθους είναι απαραίτητη η προσπέραση όλων των λευκών χαρακτήρων και έπειτα η συνέχιση της αναζήτησης κάποιου token.*

Το BEGIN(0); ⟶ *Χρησιμοποιείται διότι από τη στιγμή που θα εισέλθουμε σε κατάσταση λάθους δεν μπορούμε να "βγούμε" από αυτήν. Αντιθέτως μετά την αναγνώριση ενός σφάλματος επιθυμούμε την έξοδο μας από αυτή την κατάσταση και την αναγνώριση εκ νέου κάποιου άλλου token.*

*Η έκφραση [^ \t\r\n]{no_whites}** όπου no_whites: [^ \t\r\n]

Σε πολλές εκφράσεις εμφανίζεται το παραπάνω ή παραλλαγές αυτού, κατά τις οποίες στο πρώτο σύνολο υπάρχει και κάποιος άλλος χαρακτήρας σχετικός με το token. Κατ' αυτόν τον τρόπο υλοποιούμε την συνάρτηση avoidchars(), δηλαδή από την στιγμή που θα εντοπιστεί κάποιος μη δεκτός χαρακτήρας όλο το υπόλοιπο token είναι λάθος προσπέρνα το. πχ {sym}[^= \t\r\n]{no_whites}* όπου sym: [-+*\/%] έτσι εντοπίζει λάθη όπως +κ= *. -λάθος4=5+

Prefix

Το [-] μπαίνει σκοπίμως πρώτα και μετά το [+] γιατί αλλιώς θα είχε ειδική σημασία εύρους. *(το εύρος θα ξεκίναγε από + και δεν θα έλεγε που τελειώνει)*

Ο ειδικός χαρακτήρας \r

Συχνά εμφανίζεται ο ειδικός χαρακτήρας \r όπως εδώ [^ \t\r\n].
Είναι ένας ειδικός χαρακτήρας \r που σημαίνει για κάποια λειτουργικά αλλαγή γραμμής.

ΣΥΝΑΡΤΗΣΕΙΣ

Void yyerror(int x);

Περιέχει μια ρουτίνα διαχείρισης λεκτικών σφαλμάτων, την yyerror, η οποία καλείται όταν μια ακολουθία χαρακτήρων της συμβολοσειράς εισόδου δεν μπορεί να αποκωδικοποιηθεί σε μορφή λεκτικής μονάδας. Η ρουτίνα αντιμετωπίζει το πρόβλημα με τη μέθοδο του πανικού τυπώνοντας κατάλληλο διευκρινιστικό μήνυμα και αυξάνοντας ένα μετρητή σημαντικών σφαλμάτων (fatal errors) κατά 1.

Void countlines();

Για την καταμέτρηση των γραμμών στο αρχείο, κατέστη αναγκαία η δημιουργία μιας βοηθητικής συνάρτησης countlines(); Η συνάρτηση ελέγχει έναν έναν όλους του χαρακτήρες του λεξήματος αναζητώντας για \n ή \r, χρησιμοποιώντας τις έτοιμες συναρτήσεις yyleng(Το μέγεθος του λεξήματος που ελέγχεται) και yytext(Δείκτης προς το λέξημα που διερευνάται).

ΣΥΜΠΕΡΑΣΜΑΤΑ

Το Flex σου προσφέρει την δυνατότητα να αναλύεις πολύ γρήγορα δεδομένα. Μάλιστα, δεν θα ήταν υπερβολή να πούμε ότι ίσως, το παραγόμενο αρχείο που υλοποιεί θα ήταν ταχύτερο από εκείνο που θα δημιουργούσαμε χειροκίνητα. Ταυτόχρονα εξασφαλίζεται η ορθότητα του Flex, εφόσον υφίσταται ως εργαλείο εδώ και πάρα πολύ καιρό. Συνάμα θα συναντήσει κανείς μια πολύ ικανοποιητική ποσότητα υποστηρικτικού υλικού αναφορικά με το εργαλείο. Αναμφισβήτητα, η επιδιόρθωση ή μετατροπή ενός προγράμματος καθίσταται δυσχερέστερη όταν πρέπει να επέμβεις εις βάθος στις καταστάσεις που έχει κάποιος ορίσει σε έναν κατασκευαστή, σε αντιδιαστολή με ένα κατασκευαστή του οποίου απλώς τροποποιείς τις εκφράσεις. Για να μην αναφέρουμε τα στάδια στα οποία πρέπει να προβεί κάποιος για να δημιουργήσει δικό του κατασκευαστή, καταστώντας χρονοβόρα την όλη διαδικασία. Ο χρόνος, η πολυπλοκότητα και επαγωγικά το ποσοστό λαθών αυξάνονται εκθετικά σε σχέση με την ποσότητα των καταστάσεων που δημιουργούμε. Συμπερασματικά, η χειροκίνητη δημιουργία κατασκευαστή είναι απαγορευτική εφόσον διαθέτουμε μέσα όπως το Flex.