ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΑΘΗΜΑ: ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΕΡΓΑΣΙΑ 1η ΜΕΡΗ Α2

ΤΜΗΜΑ: Β2

ΟΜΑΔΑ: 1η

ΜΕΛΗ ΟΜΑΔΑΣ:

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

Πίνακας περιεχομένων

1. Εισαγωγή

Στην πρώτη εργαστηριακή άσκηση καλούμαστε να υλοποιήσουμε το μέρος Α-2 το οποίο ασχολείται με την κωδικοποίηση αυτόματων πεπερασμένων καταστάσεων FSM. Αρχικά, πρέπει να σχεδιάσουμε τις κανονικές εκφράσεις των λεκτικών μονάδων της γλώσσας καθώς και τα αντίστοιχα πεπερασμένα αυτόνομα συστήματα αναγνώρισής τους.

Στη συνέχεια, αφού τα δημιουργήσουμε, θα πρέπει να τα συνενώσουμε σε ένα ενιαίο αυτόματο σύστημα στο οποίο θα βασιστούμε ώστε να το προσομοιώσουμε σε έναν γενικό Πίνακα Μεταβάσεων, ο οποίος θα περιέχει τις διαδρομές των καταστάσεων που υπάρχουν στο ενιαίο αυτόματο.

Με τη χρήση του εργαλείου FSM , θα κωδικοποιήσουμε τον γενικό Πίνακα Μεταβάσεων με σκοπό να ελέγξουμε την ορθή αναγνώριση των λεκτικών μονάδων της γραμματικής. Σε αυτή τη διαδικασία της σύνταξης του κώδικα FSM, θα προχωρήσουμε σε κατάλληλες δοκιμές οι οποίες θα μας επιτρέψουν να ελέγξουμε την λειτουργικότητα και την αποτελεσματικότητα του ενιαίου αυτόματου.

Τέλος, στην εργασία θα αποδώσουμε ένα γενικό συμπέρασμα για το σύνολο των ενεργειών που υλοποιήσαμε καθώς και για τυχόν παραδοχές ή προβλήματα που συναντήσαμε στον κώδικα ή το ενιαίο αυτόματο.

2. Λεκτικές Μονάδες

Τα στοιχεία μιας γλώσσας διαχωρίζονται και όταν με βάση ένα πρότυπο αναγνώρισης αναγνωριστούν επιτυχώς, χωρίζονται σε λεκτικές μονάδες. Αυτός ο χωρισμός σε λεκτικές μονάδες συχνά δεν είναι προκαθορισμένος. Σε περιπτώσεις όπου υπάρχει ασάφεια κατά το διαχωρισμό λέξεων, ο διαχωρισμός περιλαμβάνει την μακρύτερη πιθανή συμβολοσειρά στην οποία σχηματίζεται ένα αποδεκτό token όταν η συμβολοσειρά εισόδου διαβάζεται κατά το διαχωρισμό της από αριστερά προς τα δεξιά.

2.1 Αναγνωριστικά:

Ονόματα μεταβλητών που δέχονται χαρακτήρες [a-z] [A-Z] ,_ και εκτός από τον πρώτο χαρακτήρα τους αριθμούς [0-9] με όριο μήκους 32 χαρακτήρων.

Κώδικας σε Regex:

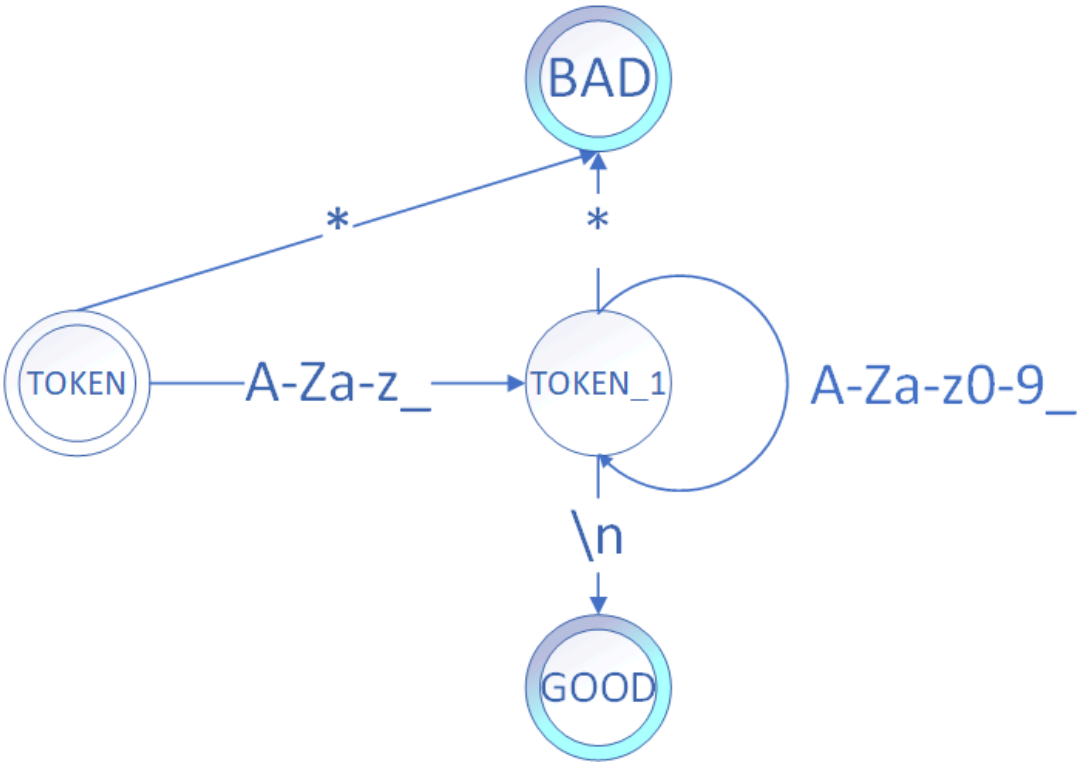
```
[A-Za-z_][A-Za-z0-9_]{0,32}
```

Ενδεικτικά παραδείγματα από το περιβάλλον RegEx Pal:

```
x
sum
_value
myVariable55
__counter
MAX_SIZE
myFunction
3variable
!x
```

Αποδεκτά γίνονται όλα εκτός από το 3variable διότι ξεκινάει με αριθμό και το !x διότι ξεκινά με σύμβολο διαφορετικό του «_».

2.1.1 Αυτόματο



2.1.2 Πίνακας Μεταβάσεων Αυτόματου

Χαρακτήρες Εισόδου					
Καταστάσεις		A-Z a-z _	0-9	*	\n
	TOKEN	TOKEN_2			
	TOKEN_2	TOKEN_2	TOKEN_2		GOOD

2.1.3 Κώδικας FSM

```
//Ξεκινάει από την κατάσταση TOKEN
START = TOKEN

TOKEN:
    //Αποδεκτοί χαρακτήρες για τον πρώτο χαρακτήρα της μεταβλητής
είναι το _
    //και οι λατινικοί χαρακτήρες πεζά και κεφαλαία
    a-z A-Z _ -> TOKEN_2
    //οποιοδήποτε άλλος χαρακτήρας οδηγεί στην κατάσταση bad
    * -> BAD

TOKEN_2:
    //Αποδεκτοί χαρακτήρες για τους υπόλοιπους χαρακτήρες είναι οι
παραπάνω και τα ψηφία 0-9
    a-z A-Z _ 0-9 -> TOKEN_2
    \n -> GOOD
```

```
    //οποιοδήποτε άλλος χαρακτήρας οδηγεί στην κατάσταση bad
    * -> BAD

BAD:
    //Έξοδος bad
    * -> BAD

GOOD:
    //Έξοδος good
    * -> GOOD
    ^D -> EXIT
EXIT:
```

2.2 Λέξεις κλειδιά:

Σύμφωνα με την εκφώνηση όσο αφορά τις λέξεις κλειδιά δεν απαιτούνται κανονικές εκφράσεις Κ.Ε. καθώς είναι καταχωρημένες ήδη στον πίνακα συμβόλων.

2.3 Λεκτικά Κυριολεκτικά:

Κώδικας σε Regex:

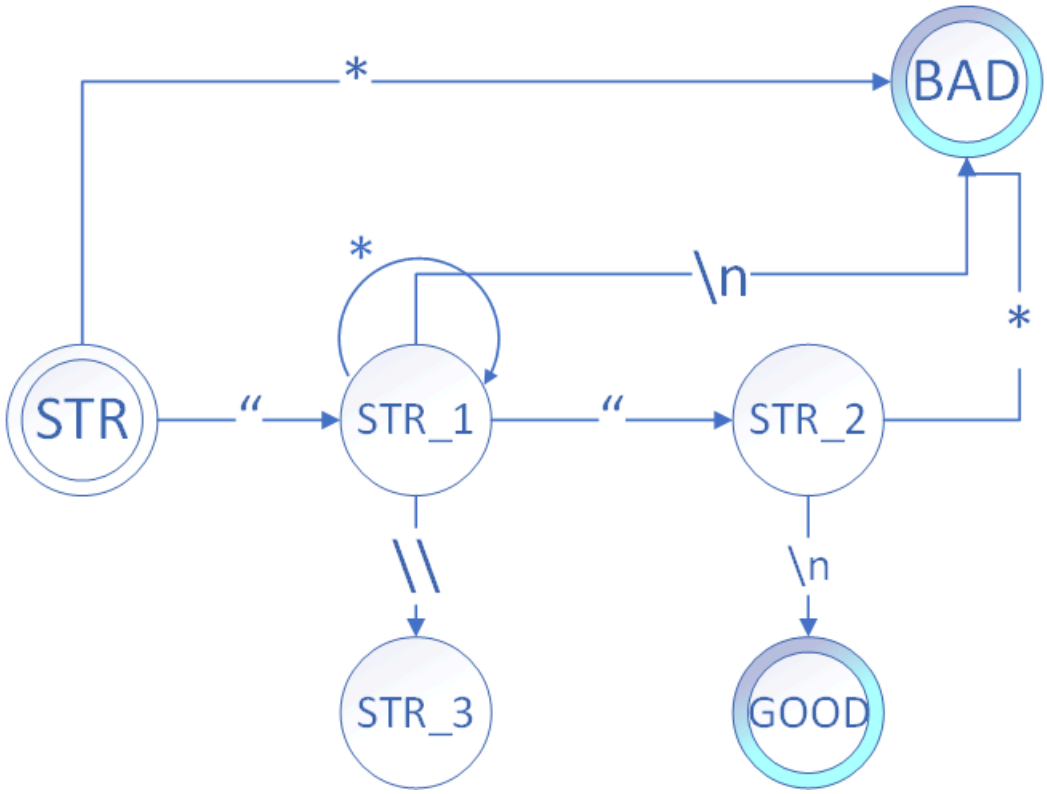
```
["]([^"\\\n]|\\.|\\\n)*["]
```

Ενδεικτικά παραδείγματα από το περιβάλλον RegEx Pal:

```
""
" "
"Test"
"Mark said, \"Boo!\"\n"
"\"
""\""
```

Αποδεκτά γίνονται όλα εκτός από το "\" καθώς το σκέτο backslash προκαλεί error και το ""\"" διότι παρόλο που είναι αποδεκτά τα " μέσα σε άλλα " προκαλείται error λόγω του backslash.

2.3.1 Αυτόματο



2.3.2 Πίνακας Μεταβάσεων Αυτόματου

Χαρακτήρες Εισόδου						
Καταστάσεις		A-Z a-z _	0-9	"	\n	\
	STR			STR_1		
	STR_1	STR_1	STR_1	STR_2		STR_3
	STR_2				GOOD	
	STR_3	STR_1	STR_1	STR_1		

2.3.3 Κώδικας FSM

```
//Ξεκινάει από την κατάσταση STR

START = STR
STR:
    //Οπωσδήποτε ξεκινάει με " και συνεχίσει στην κατάσταση STR_1
    " -> STR_1
    //Σε άλλη περίπτωση βγάζει bad
    * -> BAD

STR_1:
    //Αποδέχεται όλα τα σύμβολα
    * -> STR_1
    //Αν βρεθεί " μεταβαίνει στην κατάσταση STR_2
    " -> STR_2
```

```
    //Αν δοθεί ο χαρακτήρας \ μεταβαίνει στην κατάσταση STR_3 για τη
χρήση του " ως χαρακτήρας
    \\ -> STR_3
    //Αν δοθεί η αλλαγή γραμμής πριν το τελικό " μεταβαίνει στην BAD
    \n -> BAD

STR_2:
    //Δεν Αποδέχεται κανένα χαρακτήρα μετά το πέρας των "" εκτός από
το \n
    * -> BAD
    \n -> GOOD

STR_3:
    //Αν δοθεί το " τότε λαμβάνει το \" ως αποδεκτό χαρακτήρα
    " -> STR_1
    //Οτιδήποτε άλλο δοθεί επιστρέφει την συνέχιση του σχολίου
    * -> STR_1
BAD:
    //Έξοδος bad
    * -> BAD

GOOD:
    //Έξοδος good
    * -> GOOD
    ^D -> EXIT
EXIT:
```

2.4 Ακέραιοι:

Επιτρεπτές τιμές ακεραίων αριθμών. Οι δεκαδικοί αριθμοί πρέπει να ξεκινάνε από μη μηδενικό αριθμό (1-9) ακολουθώντας μετά οποιοσδήποτε αριθμός ψηφίων (0-9), με εξαίρεση τον αριθμό 0. Οι δεκαεξαδικοί αριθμοί πρέπει πάντα να ξεκινάνε από 0X ή 0x και να ακολουθούνται ένα ή περισσότερα ψηφία (0-9,A-F). Οι οκταδικοί αριθμοί πρέπει πάντα να ξεκινάνε από 0 και ακολουθεί ένα ή περισσότερα στοιχεία (0-7).

Κώδικας σε Regex:

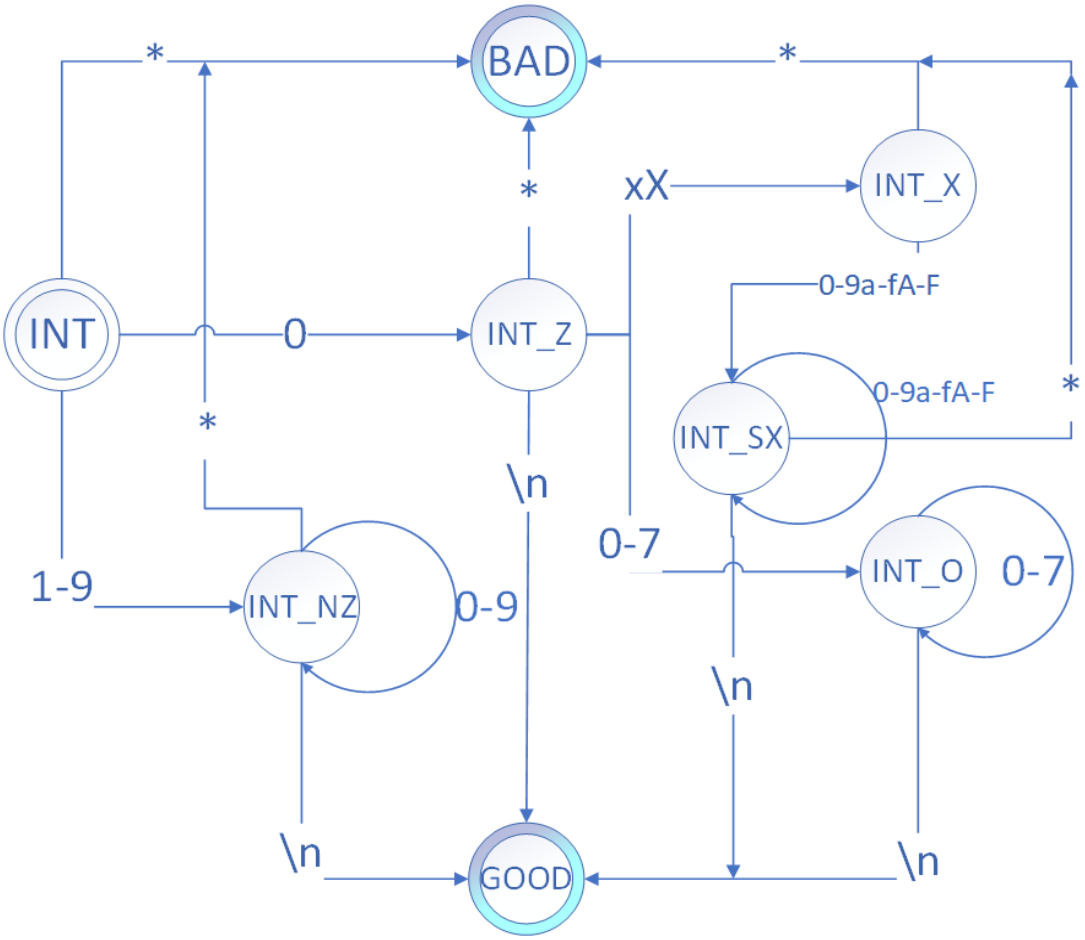
```
0[xX][0-9a-fA-F]+|[1-9][0-9]*|0[0-7]+|0\b
```

Ενδεικτικά παραδείγματα από το περιβάλλον RegEx Pal:

```
3
214748
0x4F
0XFF42
063
0147
00
07
09
0x98xAC
```

Αποδεκτά γίνονται όλα εκτός από το 09 διότι το 9 δεν ανήκει στο οκταδικό οπότε θεωρείται δεκαδικός, και οι δεκαδικοί με εξαίρεση το 0 δεν επιτρέπεται να ξεκινούν με μηδενικό ψηφίο και το 0x98xAC διότι οι δεκαεξαδικοί αριθμοί ξεκινούν πάντα με 0x ή 0X όπου το x δεν μπορεί να επαναληφθεί.

2.4.1 Αυτόματο



2.4.2 Πίνακας Μεταβάσεων Αυτόματου

Χαρακτήρες Εισόδου								
Καταστάσεις		0	1-7	8 9	x X	a-f A-F	*	\n
	INT	INT_Z	INT_NZ	INT_NZ				
	INT_NZ	INT_NZ	INT_NZ	INT_Z				
	INT_Z	INT_O	INT_O		INT_X			GOOD
	INT_X	INT_SX	INT_SX	INT_SX		INT_SX		
	INT_SX	INT_SX	INT_SX	INT_SX		INT_SX		GOOD
	INT_O	INT_O	INT_O					GOOD

2.4.3 Κώδικας FSM

```
//Ξεκινάει από την κατάσταση INT
START=INT

INT:
    //Αποδεκτοί αριθμοί για δεκαδικό 1-9
    1-9    -> INT_NZ
    //Το δεκαεξαδικό ξεκινάει με 0
    0 -> INT_Z
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    *    -> BAD
INT_NZ:
    //Από το δεύτερο ψηφίο και μετά είναι αποδεκτά όλα τα ψηφία
    0-9    -> INT_NZ
    \n    -> GOOD
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    *    -> BAD
INT_Z:
    //Δεύτερο ψηφίο του δεκαεξαδικού αριθμού
    x X -> INT_X
    //μεταβαίνει σε οκταδικό αριθμό
    0-7 -> INT_O
    \n -> GOOD
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
INT_X:
    //Αποδεκτοί αριθμοί δεκαεξαδικού
    0-9 a-f A-F -> INT_SX
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
INT_SX:
    //Συνέχιση του δεκαεξαδικού μέχρι το πέρας του
    0-9 a-f A-F -> INT_SX
    \n -> GOOD
```

```
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
INT_O:
    //αποδεκτά ψηφία από το δεύτερο και έπειτα
    0-7 -> INT_O
    \n -> GOOD
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
BAD:
    //Έξοδος bad
    * -> BAD

GOOD:
    //Έξοδος good
    * -> GOOD
    ^D -> EXIT
EXIT:
```

2.5 Αριθμοί Κινούμενης υποδιαστολής:

Επιτρεπτές τιμές αριθμών κινούμενης υποδιαστολής. Το ακέραιο μέρος του αριθμού μπορεί να είναι (0-9) όπως και το πραγματικό, χωρισμένα με (.) . Στο πραγματικό μέρος μπορεί να εμπεριέχεται και ο αριθμός ‘e’ ή ‘E’ όπου οι αριθμοί που ακολουθούν μετά το e υψώνονται σε ακέραια δύναμη.

Κώδικας σε Regex:

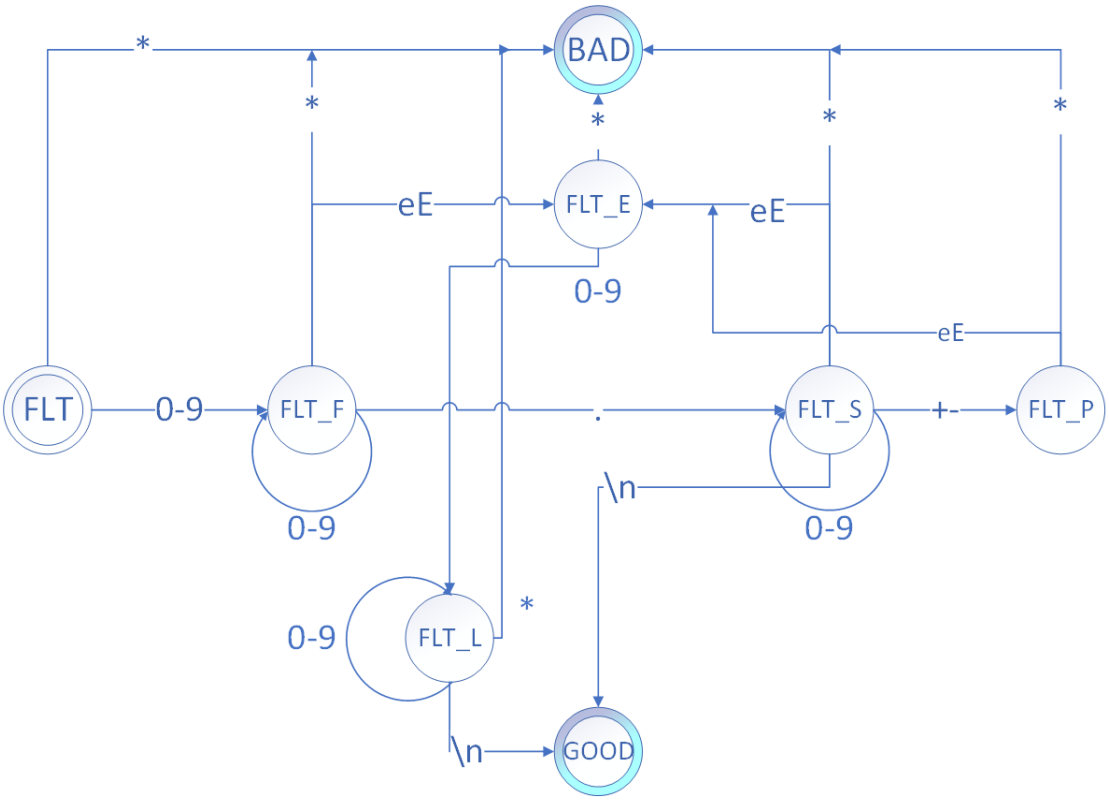
```
[0-9]+((\.[0-9]+)([eE][+-]?[0-9]*)?|([eE][+-]?[0-9]*)+)
```

Ενδεικτικά παραδείγματα από το περιβάλλον RegEx Pal:

```
3.14
10.0
0.001
1e100
3.14e-10
0e0
3e.6
```

Αποδεκτά γίνονται όλα εκτός από το 3e.6 καθώς είναι μη αποδεκτή μορφή του e.

2.5.1 Αυτόματο



2.5.2 Πίνακας Μεταβάσεων Αυτόματου

Χαρακτήρες Εισόδου							
Καταστάσεις		0-9	e E	.	+ -	*	\n
	FLT	FLT_F					
	FLT_F	FLT_F	FLT_E	FLT_S			
	FLT_S	FLT_S	FLT_E				GOOD
	FLT_P	FLT_L					
	FLT_E	FLT_L			FLT_P		
	FLT_L	FLT_L					GOOD

2.5.3 Κώδικας FSM

```
//Ξεκινάει από την κατάσταση FLT
START=FLT

FLT:
    //Για το πρώτο ψηφίο αποδεκτά είναι τα ψηφία 0-9
    0-9    -> FLT_F
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
FLT_F:
    0-9    -> FLT_F
```

```
    //Από το δεύτερο ψηφίο και έπειτα είναι αποδεκτό και το e και
μεταβαίνει στην κατάσταση FLT_E
    e E    -> FLT_E
    //Αν δοθεί . μεταβαίνει στο πραγματικό μέρος και μεταβαίνει στην
κατάσταση FLT_S
    . -> FLT_S
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
FLT_S:
    //αποδεκτοί αριθμοί για το πραγματικό μέρος
    0-9    -> FLT_S
    e E    -> FLT_E
    \n -> GOOD
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
FLT_P:
    0-9    -> FLT_L
    * -> BAD
FLT_E:
    //αποδεκτά σύμβολα και αριθμοί για το e
    0-9 -> FLT_L //μεταβαίνει στην κατάσταση FLT_L διότι μετά
επιτρέπονται μόνο αριθμοί αν δοθεί αριθμός μετά το e
    + \- -> FLT_P
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
FLT_L:
    0-9    -> FLT_L
    \n -> GOOD
    //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
    * -> BAD
BAD:
    //Έξοδος bad
    * -> BAD

GOOD:
   //Εξοδος good
    * -> GOOD
    ^D -> EXIT
EXIT:
```

2.6 Τελεστές:

Σύμφωνα με την εκφώνηση για όσο αφορά τους τελεστές δεν απαιτούνται κανονικές εκφράσεις Κ.Ε. καθώς είναι καταχωρημένοι ήδη στον πίνακα συμβόλων.

2.7 Σχόλια:

Στη Uni-C υπάρχουν τα σχόλια γραμμής που συμβολίζονται με // και ολοκληρώνονται με το τέλος της γραμμής \n. Και τα σχόλια πολλαπλών γραμμών που συμβολίζονται με /*(περιεχόμενο) */.

Κώδικας σε Regex:

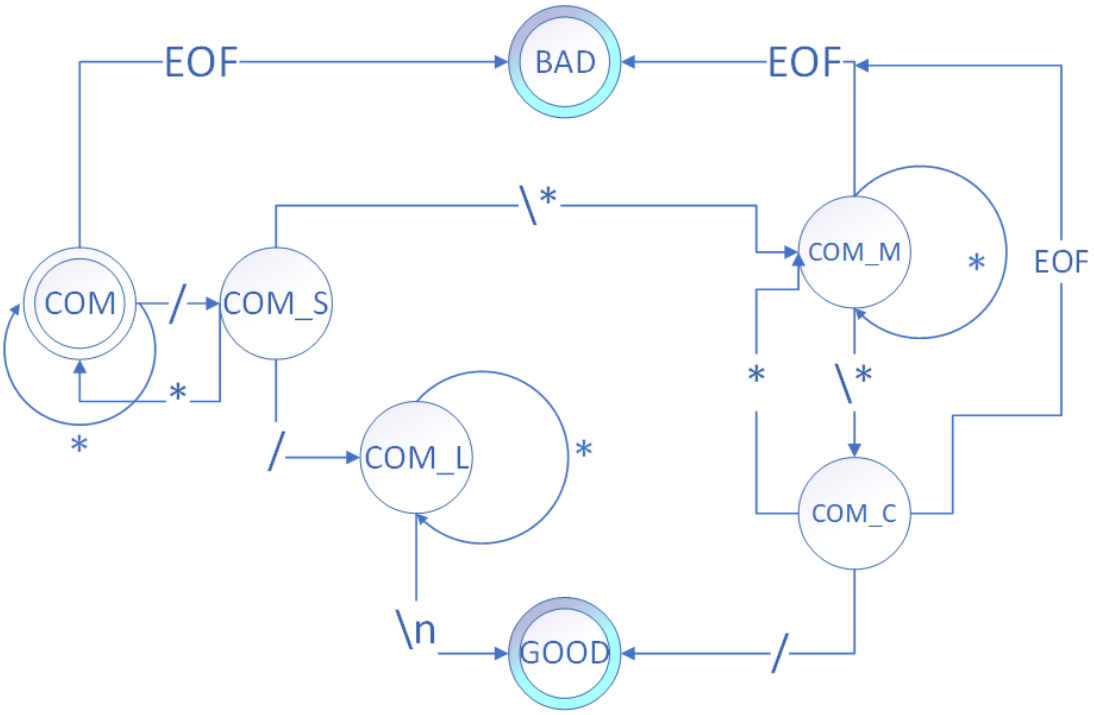
```
^//.*|\/\*(\*(?!\/)|[^*])*\*\/
```

Ενδεικτικά παραδείγματα από το περιβάλλον RegEx Pal:

```
//example1/*
/*example2 .1
example 2.2
example 2.3*/
/*example3*/
/*example//example
```

Αποδεκτά γίνονται όλα εκτός από το τελευταίο καθώς έχει ξεκινήσει σχόλιο πολλαπλών γραμμών που δεν είναι ολοκληρωμένο οπότε και το // δεν αναγνωρίζεται ως σχόλιο γραμμής αλλά σαν απλά σύμβολα.

2.7.1 Αυτόματο



2.7.2 Πίνακας Μεταβάσεων Αυτόματου

Χαρακτήρες Εισόδου						
Καταστάσεις		/	*	*	\n	EOF
	COM	COM_S	COM	COM	COM	BAD
	COM_S	COM_L	COM_M	COM	COM	COM
	COM_L	COM_L	COM_L	COM_L	GOOD	GOOD
	COM_M	COM_M	COM_C	COM_M	COM_M	BAD
	COM_C	GOOD	COM_M	COM_M	COM_M	BAD

2.7.3 Κώδικας FSM

```
START = COM

COM:
    * -> COM
    / -> COM_S
    EOF -> BAD
COM_S:
    //σχόλια πολλαπλών γραμμών
    * -> COM
    //σχόλια γραμμής
    / -> COM_L
    //τέλος σχολίων
    \* -> COM_M
COM_L:
    // περιεχόμενο σχολίων
    * ->COM_L
    \n -> GOOD
COM_M:
    //η μη ολοκλήρωση του σχολίου οδηγεί στην κατάσταση BAD
    * -> COM_M
    EOF -> BAD
    \* -> COM_C
COM_C:
    * -> COM_M
    / -> GOOD
    EOF -> BAD
BAD:
    //Έξοδος bad
    * -> BAD

GOOD:
    //Έξοδος good
    * -> GOOD
    ^D -> EXIT
EXIT:
```

2.8 White_spaces χαρακτήρες:

White spaces θεωρούνται τα απλά κενά τα tabs ή ο συνδυασμός και των δύο.

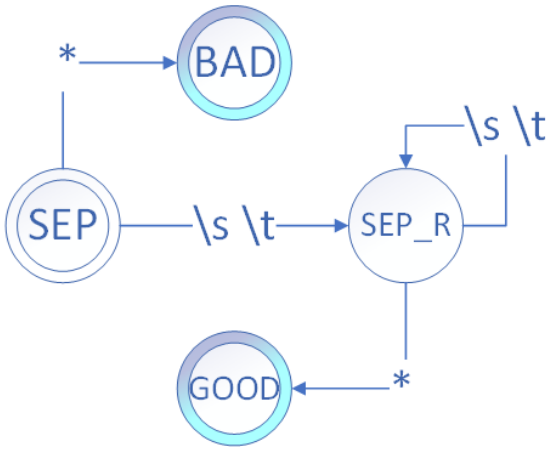
Κώδικας σε Regex:

```
\s
```

Ενδεικτικά παραδείγματα από το περιβάλλον RegEx Pal:

```
ex_ample
```

2.8.1 Αυτόματο



2.8.2 Πίνακας Μεταβάσεων Αυτόματου

Χαρακτήρες Εισόδου				
Καταστάσεις		\s	\t	*
	SEP	SEP_R	SEP_R	GOOD
	SEP_R	SEP_R	SEP_R	GOOD

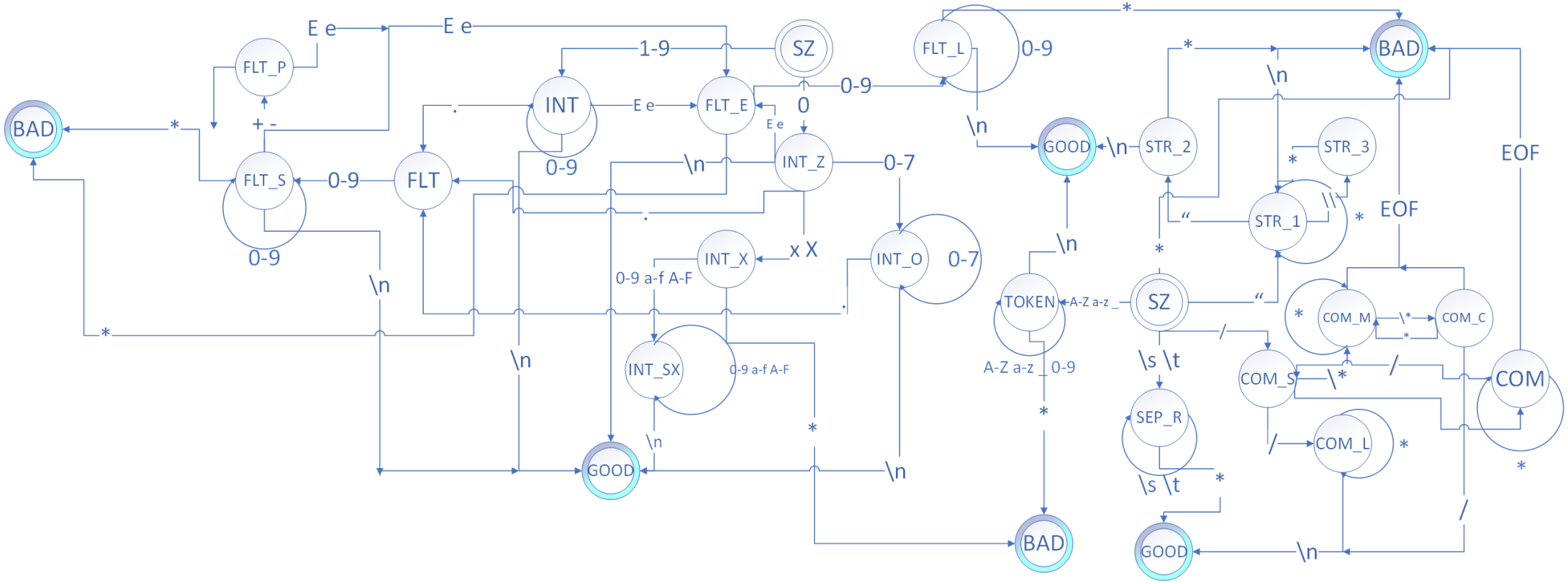
2.8.3 Κώδικας FSM

```
//Ξεκινάει από την κατάσταση SEP
START=SEP
//Δέχεται το χαρακτήρα space και tab οτιδήποτε άλλο οδηγεί στην
κατάσταση bad
SEP:    \s    -> SEP_R
        \t    -> SEP_R
        *     -> GOOD
SEP_R:  \s    -> SEP_R
        \t    -> SEP_R
        *     -> GOOD

BAD:
    // Έξοδος bad
    * -> BAD

GOOD:
    // Έξοδος good
```

```
    * -> GOOD
    ^D -> EXIT
EXIT:
```

3 Ενιαίο Αυτόματο

Αυτόματο



Πίνακας Μεταβάσεων Αυτόματου

Καταστάσεις	Χαρακτήρες Εισόδου																	
		1-9	a-z A-Z	\s	\t	"	/	0	*	*	\n	e E	.	+ \-	x X	1-7	a-f A-F	\\
	SZ	INT	TOKEN	SEP_R	SEP_R	STR_1	COM_S	INT_Z										
	COM						COM_S		COM									
	COM_S						COM_L		COM	COM_M								
	COM_L								COM_L		GOOD							
	COM_M									COM_C								
	COM_C						GOOD		COM_M									
	FLT	FLT_F																
	FLT_F	FLT_F										FLT_E	FLT_S					
	FLT_S	FLT_S									GOOD	FLT_E						
	FLT_P	FLT_L																
	FLT_E	FLT_L												FLT_P				
	FLT_L	FLT_L									GOOD							
	INT	INT_NZ						INT_Z				FLT_E	FLT_S					
	INT_NZ	INT_NZ						INT_NZ			GOOD							
	INT_Z							INT_O			GOOD	FLT_E	FLT_S		INT_X	INT_O		
	INT_X	INT_SX						INT_SX									INT_SX	
	INT_SX	INT_SX						INT_SX			GOOD						INT_SX	
	INT_O							INT_O			GOOD					INT_O		
	STR					STR_1												
	STR_1					STR_2			STR_1									STR_3
	STR_2										GOOD							
	STR_3					STR_1			STR_1									
	SEP			SEP_R	SEP_R													
	SEP_R			SEP_R	SEP_R				SEP_R		GOOD							
	TOKEN	TOKEN_2	TOKEN_2					TOKEN_2			GOOD							
	TOKEN_2																	

Κώδικας FSM

```
START=SZ

SZ:
      1-9 -> INT_NZ
      A-Z a-z _ -> TOKEN_2
      \s -> SEP_R
      \t -> SEP_R
      " -> STR_1
      / -> COM_S
      0 -> INT_Z
      * -> BAD


COM:
    * -> COM
    / -> COM_S
    EOF -> BAD
COM_S:
    //σχόλια πολλαπλών γραμμών
    * -> COM
    //σχόλια γραμμής
    / -> COM_L
    //τέλος σχολίων
    \* -> COM_M
COM_L:
    // περιεχόμενο σχολίων
    * ->COM_L
    \n -> GOOD
COM_M:
    //η μη ολοκλήρωση του σχολίου οδηγεί στην κατάσταση BAD
    * -> COM_M
    EOF -> BAD
    \* -> COM_C
COM_C:
    * -> COM_M
    / -> GOOD
    EOF -> BAD

FLT:
      //Για το πρώτο ψηφίο αποδεκτά είναι τα ψηφία 0-9
      0-9   -> FLT_F
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
```

```
FLT_F:
      0-9   -> FLT_F
      //Από το δεύτερο ψηφίο και έπειτα είναι αποδεκτό και το e και
μεταβαίνει στην κατάσταση FLT_E
      e E   -> FLT_E
      //Αν δοθεί . μεταβαίνει στο πραγματικό μέρος και μεταβαίνει
στην κατάσταση FLT_S
      . -> FLT_S
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
FLT_S:
      //αποδεκτοί αριθμοί για το πραγματικό μέρος
      0-9   -> FLT_S
      e E   -> FLT_E
      \n -> GOOD
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
FLT_P:
      0-9   -> FLT_L
      * -> BAD
FLT_E:
      //αποδεκτά σύμβολα και αριθμοί για το e
      0-9 -> FLT_L //μεταβαίνει στην κατάσταση FLT_L διότι μετά
επιτρέπονται μόνο αριθμοί αν δοθεί αριθμός μετά το e
      + \- -> FLT_P
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
FLT_L:
      0-9   -> FLT_L
      \n -> GOOD
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD

INT:
    e -> FLT_E
    . -> FLT_S
      //Αποδεκτοί αριθμοί για δεκαδικό 1-9
      1-9   -> INT_NZ
      //Το δεκαεξαδικό ξεκινάει με 0
      0 -> INT_Z
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      *     -> BAD
INT_NZ:
      //Από το δεύτερο ψηφίο και μετά είναι αποδεκτά όλα τα ψηφία
      0-9   -> INT_NZ
      \n    -> GOOD
```

```
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      *     -> BAD
INT_Z:
    e -> FLT_E
    . -> FLT_S
      //Δεύτερο ψηφίο του δεκαεξαδικού αριθμού
      x X -> INT_X
      //μεταβαίνει σε οκταδικό αριθμό
      0-7 -> INT_O
      \n -> GOOD
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
INT_X:
      //Αποδεκτοί αριθμοί δεκαεξαδικού
      0-9 a-f A-F -> INT_SX
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
INT_SX:
      //Συνέχιση του δεκαεξαδικού μέχρι το πέρας του
      0-9 a-f A-F -> INT_SX
      \n -> GOOD
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD
INT_O:
      //αποδεκτά ψηφία από το δεύτερο και έπειτα
      0-7 -> INT_O
      \n -> GOOD
      //Οτιδήποτε άλλο οδηγεί στην κατάσταση BAD
      * -> BAD

STR:
    //Οπωσδήποτε ξεκινάει με " και συνεχίσει στην κατάσταση STR_1
    " -> STR_1
    //Σε άλλη περίπτωση βγάζει bad
    * -> BAD

STR_1:
    //Αποδέχεται όλα τα σύμβολα
    * -> STR_1
    //Αν βρεθεί " μεταβαίνει στην κατάσταση STR_2
    " -> STR_2
    //Αν δοθεί ο χαρακτήρας \ μεταβαίνει στην κατάσταση STR_3 για τη
χρήση του " ως χαρακτήρας
    \\ -> STR_3
    //Αν δοθεί η αλλαγή γραμμής πριν το τελικό " μεταβαίνει στην BAD
    \n -> BAD
```

```
STR_2:
    //Δεν Αποδέχεται κανένα χαρακτήρα μετά το πέρας των "" εκτός από
το \n
    * -> BAD
    \n -> GOOD

STR_3:
    //Αν δοθεί το " τότε λαμβάνει το \" ως αποδεκτό χαρακτήρα
    " -> STR_1
    //Οτιδήποτε άλλο δοθεί επιστρέφει την συνέχιση του σχολίου
    * -> STR_1

SEP:    \s    -> SEP_R
        \t    -> SEP_R
        *     -> BAD
SEP_R:  \s    -> SEP_R
        \t    -> SEP_R
        *     -> SEP_R
        \n -> GOOD

TOKEN:
    //Αποδεκτοί χαρακτήρες για τον πρώτο χαρακτήρα της μεταβλητής
είναι το _
    //και οι λατινικοί χαρακτήρες πεζά και κεφαλαία
    a-z A-Z _ -> TOKEN_2
    //οποιοδήποτε άλλος χαρακτήρας οδηγεί στην κατάσταση bad
    * -> BAD

TOKEN_2:
    //Αποδεκτοί χαρακτήρες για τους υπόλοιπους χαρακτήρες είναι οι
παραπάνω και τα ψηφία 0-9
    a-z A-Z _ 0-9 -> TOKEN_2
    \n -> GOOD
    //οποιοδήποτε άλλος χαρακτήρας οδηγεί στην κατάσταση bad
    * -> BAD

BAD:
    //Έξοδος bad
    * -> BAD

GOOD:
    //Έξοδος good
    * -> GOOD
    ^D -> EXIT
EXIT:
```

4 Ενδεικτικά παραδείγματα

1ο παράδειγμα

```
Εκτελούμε το SZ.fsm με την εντολή .\SZ.fsm -trace /*Hello world*/
sz / -> com_s
com_s * -> com_m
com_m H -> com_m
com_m e -> com_m
com_m l -> com_m
com_m l -> com_m
com_m o -> com_m
com_m \s -> com_m
com_m w -> com_m
com_m o -> com_m
com_m r -> com_m
com_m l -> com_m
com_m d -> com_m
com_m * -> com_c
com_c / -> good
good \n -> good
^Z
good EOF -> good
```

2ο παράδειγμα

```
/*Hello world *
sz / -> com_s
com_s * -> com_m
com_m H -> com_m
com_m e -> com_m
com_m l -> com_m
com_m l -> com_m
com_m o -> com_m
com_m \s -> com_m
com_m w -> com_m
com_m o -> com_m
com_m r -> com_m
com_m l -> com_m
com_m d -> com_m
com_m \s -> com_m
com_m * -> com_c
com_c \n -> com_m
^Z
com_m EOF -> bad
```

3ο παράδειγμα

```
//Hello /world
sz / -> com_s
com_s / -> com_l
com_l H -> com_l
com_l e -> com_l
com_l l -> com_l
com_l l -> com_l
com_l o -> com_l
com_l \s -> com_l
com_l / -> com_l
com_l w -> com_l
com_l o -> com_l
com_l r -> com_l
com_l l -> com_l
com_l d -> com_l
com_l \n -> good
^Z
good EOF -> good
```

4ο παράδειγμα

```
/Hello /world
sz / -> com_s
com_s H -> com
com e -> com
com l -> com
com l -> com
com o -> com
com \s -> com
com / -> com_s
com_s w -> com
com o -> com
com r -> com
com l -> com
com d -> com
com \n -> com
^Z
com EOF -> bad
```

5ο παράδειγμα

```
0.55e4
sz 0 -> int_z
int_z . -> flt_s
flt_s 5 -> flt_s
flt_s 5 -> flt_s
```

```
flt_s e -> flt_e
flt_e 4 -> flt_l
flt_l \n -> good
^Z
good EOF -> good
```

6ο παράδειγμα

```
1e100
sz 1 -> int
int e -> flt_e
flt_e 1 -> flt_l
flt_l 0 -> flt_l
flt_l 0 -> flt_l
flt_l \n -> good
^Z
good EOF -> good
```

7ο παράδειγμα

```
e5.45
sz e -> token
token 5 -> bad
bad . -> bad
bad 4 -> bad
bad 5 -> bad
bad \n -> bad
^Z
bad EOF -> bad
```

8ο παράδειγμα

```
5..74
sz 5 -> int
int . -> flt_s
flt_s . -> bad
bad 7 -> bad
bad 4 -> bad
bad \n -> bad
^Z
bad EOF -> bad
```

9ο παράδειγμα

```
4
sz 4 -> int_nz
int_nz \n -> good
```

```
^Z
good EOF -> good
```

10ο παράδειγμα

```
09
sz 0 -> int_z
int_z 9 -> bad
bad \n -> bad
^Z
bad EOF -> bad
```

11ο παράδειγμα

```
"Mark said, \"Boo!\"\n"
sz " -> str_1
str_1 M -> str_1
str_1 a -> str_1
str_1 r -> str_1
str_1 k -> str_1
str_1 \s -> str_1
str_1 s -> str_1
str_1 a -> str_1
str_1 i -> str_1
str_1 d -> str_1
str_1 , -> str_1
str_1 \s -> str_1
str_1 \ -> str_3
str_3 " -> str_1
str_1 B -> str_1
str_1 o -> str_1
str_1 o -> str_1
str_1 ! -> str_1
str_1 \ -> str_3
str_3 " -> str_1
str_1 \ -> str_3
str_3 n -> str_1
str_1 " -> str_2
str_2 \n -> good
^Z
good EOF -> good
```

12ο παράδειγμα

```
" "
sz " -> str_1
str_1 \s -> str_1
```

```
str_1 " -> str_2
str_2 \n -> good
^Z
good EOF -> good
```

13ο παράδειγμα

```
"Hello world"123\n
sz " -> str_1
str_1 H -> str_1
str_1 e -> str_1
str_1 l -> str_1
str_1 l -> str_1
str_1 o -> str_1
str_1 \s -> str_1
str_1 w -> str_1
str_1 o -> str_1
str_1 r -> str_1
str_1 l -> str_1
str_1 d -> str_1
str_1 " -> str_2
str_2 1 -> bad
bad 2 -> bad
bad 3 -> bad
bad \ -> bad
bad n -> bad
bad \n -> bad
^Z
bad EOF -> bad
```

14ο παράδειγμα

```
        //Έχει δοθεί space και tab
sz \s -> sep_r
sep_r \t -> sep_r
sep_r \n -> good
^Z
good EOF -> good
```

15ο παράδειγμα

```
        4 //Έχει δοθεί tab+4
sz \t -> sep_r
sep_r 4 -> sep_r
sep_r \n -> good
^Z
```

```
good EOF -> good
```

16ο παράδειγμα

```
_1varible
sz _ -> token_2
token_2 1 -> token_2
token_2 v -> token_2
token_2 a -> token_2
token_2 r -> token_2
token_2 i -> token_2
token_2 b -> token_2
token_2 l -> token_2
token_2 e -> token_2
token_2 \n -> good
^Z
good EOF -> good
```

17ο παράδειγμα

```
4virable
sz 4 -> int_nz
int_nz v -> bad
bad i -> bad
bad r -> bad
bad a -> bad
bad b -> bad
bad l -> bad
bad e -> bad
bad \n -> bad
^Z
bad EOF -> bad
```

18ο παράδειγμα

```
*virable
sz * -> bad
bad v -> bad
bad i -> bad
bad r -> bad
bad a -> bad
bad b -> bad
bad l -> bad
bad e -> bad
bad \n -> bad
^Z
bad EOF -> bad
```

5 Σχόλια:

Δεν εντοπίσαμε κάποιο σφάλμα με τις δοκιμές μας.

6 Κατανομή Εργασιών

Υπεύθυνος εργασίας Α-2:	Μαρίνος Φράγκος
Κανονικές εκφράσεις:	Μαρίνος Φράγκος
Επιμέρους αυτόματα:	Διάννης Ιωάννης
Ενιαίο Αυτόματο:	Διάννης Ιωάννης
Επιμέρους ΠΜ:	Φριλίγκος Γρηγόριος, Φάσσου – Κοντοδημάκη Ιφιγένεια - Γεωργία
Πίνακας Μεταβάσεων:	Φριλίγκος Γρηγόριος, Φάσσου – Κοντοδημάκη Ιφιγένεια - Γεωργία
Κώδικας FSM:	Βροχάρης Αντώνιος, Μαρίνος Φράγκος, Διάννης Ιωάννης
Σύνταξη εγγράφου:	Φάσσου – Κοντοδημάκη Ιφιγένεια – Γεωργία, Μαρίνος Φράγκος