Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Ομαδικό Project (Μέρος Α): Δημιουργία Ανεξάρτητου Λεκτικού Αναλυτή με τη γεννήτρια FLEX

Εαρινό Εξάμηνο 2020-21
Τμήμα ΜΕΤ3, Ομάδα 4
Συντάκτης: Οικονομίδης Χαράλαμπος
Καρναβάς Απόστολος Α.Μ. 161050Εξάμηνο 10ο Π.Σ. ΠΑΔΑ cs161050@uniwa.gr
Λυκούδη ΔέσποιναΑ.Μ. 18390151Εξάμηνο 6ο Π.Σ. ΠΑΔΑ ice18390151@uniwa.gr
Οικονομίδης Χαράλαμπος Α.Μ. 18390049 Εξάμηνο 6ο Π.Σ. ΠΑΔΑ ice18390049@uniwa.gr
Πρώιος Παντελεήμων Α.Μ. 18390023 Εξάμηνο 6ο Π.Σ. ΠΑΔΑ ice18390023@uniwa.gr
Ομάδα Χρηστών: ΜΕΤ3: ΠΕΜΠΤΗ 12-2
Υπεύθυνος Καθηγητής: Ιορδανάκης Μιχάλης

Πίνακας περιεχομένων

1. Εισαγωγή

Στόχος της εξαμηνιαίας εργαστηριακής άσκησης είναι η δημιουργία ενός Μεταγλωττιστή (Compiler). Κατά τη διάρκεια υλοποίησης ενός Μεταγλωττιστή διακρίνουμε επτά φάσεις: Λεκτική Ανάλυση, Συντακτική Ανάλυση, Σημασιολογική Ανάλυση, Παραγωγή Ενδιάμεσου Κώδικα, Βελτιστοποίηση Ενδιάμεσου Κώδικα, Παραγωγή Τελικού Κώδικα και Βελτιστοποίηση Τελικού Κώδικα. Το Πηγαίο Αρχείο του μεταγλωττιστή μας αποτελεί η Uni-C και στόχος της ακόλουθης εργαστηριακής άσκησης είναι η δημιουργία ενός Λεκτικού Αναλυτή.

2. Λεκτική Ανάλυση – Λεκτικού Αναλυτή

Αρχικά, η Uni-C θα διαβάζεται από έναν Λεκτικό Αναλυτή. Η λειτουργία του Λεκτικού Αναλυτή (ή αλλιώς ΛΑ ή lexical analyzer) έχει ως ακολούθως, διαβάζει ένα-ένα τα λεξήματα εισόδου, μέσα από τη συμβολοσειρά εισόδου και να τα αναγνωρίζει ως λεκτικές μονάδες. Σκοπός του ΛΑ είναι ο διαχωρισμός και η αναγνώριση των λεξημάτων του πηγαίου κώδικα. Προτού όμως γίνει περαιτέρω ανάλυση για τις λεκτικές μονάδες, θα πρέπει να διασαφηνιστεί το αλφάβητο της Uni-C, καθώς και τα βασικά χαρακτηριστικά του πεπερασμένου αυτομάτου (ΠΑ), όπου θα δημιουργηθεί.
Το αλφάβητο **Σ** της γλώσσας, αποτελείται από όλους τους πεζούς και κεφαλαίους λατινικούς χαρακτήρες (**a-z A-Z**), από τα αριθμητικά ψηφία (**0-9**), από τους ειδικούς χαρακτήρες **! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~**, καθώς και whitespaces χαρακτήρες, το κενό (space**\s**), tab (**\t**), νέα γραμμή (NEWLINE**\n**), τέλος αρχείου (**EOF**). Επιπλέον, το πεπερασμένο αυτόματο θα πρέπει να χαρακτηρίζεται από ένα σύνολο πεπερασμένων καταστάσεων **S**, μία συνάρτηση μεταφοράς **T**, μία αρχική κατάσταση **SZ**, καθώς και ένα σύνολο **G** αποδεκτών καταστάσεων κατάληξης, το οποίο θα πρέπει να είναι υποσύνολο του **S**. Όλα όσα προαναφέρθηκαν, είναι τα βασικά στοιχεία περιγραφής ενός πεπερασμένου αυτομάτου **M** και μπορούν να γραφτούν ομαδοποιημένα ως εξής **M=(S, Σ, T, SZ, G)**.

3. Περιγραφή των επιμέρους Λεκτικών Μονάδων

Αρχικά, μία Λεκτική Μονάδα(token), προσδιορίζεται όταν οι πηγαίες λέξεις διαχωρίζονται και στη συνέχεια αναγνωρίζονται επιτυχώς, ικανοποιώντας ένα συγκεκριμένο πρότυπο αναγνώρισης.

3.1 Αναγνωριστικά (Ονόματα)

Τα λεξήματα των αναγνωριστικών αποτελούνται από έναν ή περισσότερους κεφαλαίους ή και πεζούς χαρακτήρες (a-z A-Z),είτε το underscore (_), είτε τα ψηφία (0-9) αρκεί όμως να μην βρίσκονται στη θέση του πρώτου χαρακτήρα. Η παραπάνω περιγραφή αποτελεί το πρότυπο αναγνώρισης, για τα αναγνωριστικά ή ονόματα (identifiers ή names), τα οποία αναγνωρίζονται και επιστρέφονται ως λεκτικές μονάδες, με το αναγνωριστικό όνομα identifiers.
Έστω A το πεπερασμένο αυτόματο περιγραφής αναγνωριστικών, τότε θα περιγράφεται ως εξής A=(I, Σ, δ, I1, G), όπου I={I1, I2, I3, I4}, Σ={a-z, A-Z, _, 0-9, \n}, I1∈I, G={I3} ⊆I.
Η Κανονική έκφραση που τα περιγράφει είναι η εξής: **([a-z]|[A-Z]|_)+(\d|[a-z]|[A-Z]|_)*\n**.
Μερικά αποδεκτά παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης είναι:

- α_123 -> Αποδεκτή συμβολοσειρά, δεν ξεκινάει με ψηφίο (0-9) και δεν περιέχει κάποιο σύμβολο που δεν ανήκει στο αλφάβητο του αυτόματου.
- Tolis ->Αποδεκτή συμβολοσειρά, δεν ξεκινάει με ψηφίο (0-9) και περιέχει μόνο λατινικούς (πεζούς και κεφαλαίους) χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου.

- _MET3_4 -> Αποδεκτή συμβολοσειρά, όλοι οι χαρακτήρες περιέχονται στο αλφάβητο του αυτομάτου και δεν ξεκινάει με ψηφίο (0-9).

Μερικά λανθασμένα παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης:

- 12a_e-23 -> Λανθασμένη συμβολοσειρά, ο πρώτος χαρακτήρας είναι ψηφίο και επίσης περιέχει και την παύλα (-), η οποία δεν ανήκει στο αλφάβητο του αυτόματου.
- ab12_#$ -> Λανθασμένη συμβολοσειρά, μέχρι την κάτω παύλα (_) η συμβολοσειρά είναι αποδεκτή όταν φτάνει όμως στη δίεση (#), γίνεται μη αποδεκτή καθώς αυτό το σύμβολο δεν ανήκει στο αλφάβητο του αυτομάτου.

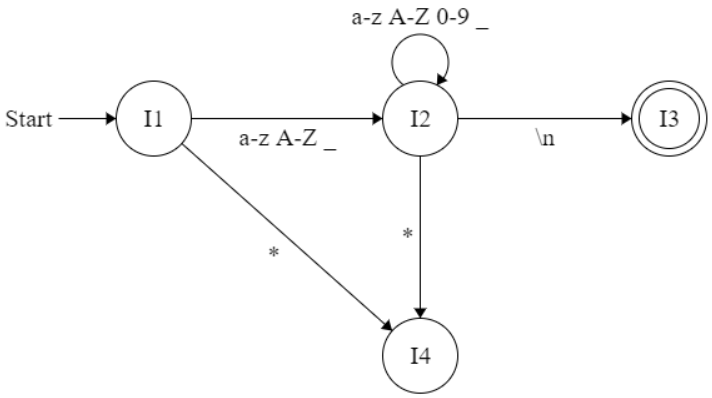
BNF έκφραση:

identifier ::= (letter | Underscore) (letter | digit | Underscore)*
letter ::= [a-z] | [A-Z]
digit ::= [0-9]
Underscore ::= '_'

Ο Πίνακας Μεταβάσεων του αυτομάτου Α είναι ο παρακάτω:

	a-z	A-Z	0-9	_	\n	OTHERS
I1	I2	I2		I2		
I2	I2	I2	I2	I2	I3	
I3						

Με τον όρο OTHERS στον πίνακα μετάβασης, εννοούμε όλους τους υπόλοιπους χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου αλλά δεν αναφέρονται στις προηγούμενες στήλες. Όσον αφορά το **\n**, το χρησιμοποιούμαι σαν το κενό (ε), για να μεταβούμε σε μία συγκεκριμένη τελική κατάσταση. Ακολουθεί το αυτόματο.



Σχήμα 1: Αυτόματο identifier

Κώδικας fsm επιμέρους αυτόματου:

START=I1
I1: a-z A-Z _ -> I2
*** -> I4**
I2: a-z A-Z 0-9 _ ->I2
*** -> I4**
\n -> I3

I3(OK):

Στον κώδικα fsm χρησιμοποιείται το **\n**, ως εναλλακτικό «κενό». Με αυτό τον τρόπο μπορούμε να μεταβαίνουμε σε μία τελική κατάσταση. Επίσης, όπου υπάρχουν κενά κελία σημαίνει ότι από την αντίστοιχη κατάσταση, με την συγκεκριμένη είσοδο πηγαίνει σε κατάσταση Λάθους.

3.2 Λεκτικά Κυριολεκτικά

Τα λεκτικά Κυριολεκτικά (strings) περικλείονται μέσα σε διπλές αποστρόφους (“) και περιλαμβάνουν οποιονδήποτε χαρακτήρα εκτός του backslash (\), της νέας γραμμής (\n) ή της διπλής αποστρόφου (“). Τα Λεκτικά Κυριολεκτικά αναγνωρίζονται και επιστρέφονται ως Λεκτική Μονάδα, με το αναγνωριστικό όνομα String.

Έστω B το πεπερασμένο αυτόματο περιγραφής αναγνωριστικών, τότε θα περιγράφεται ως εξής B=(S, Σ, δ, SZ, G), όπου S={SZ, S0, S1, GOOD, BAD}, Σ={a-z, A-Z, ! “ # % & ’ ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~}, SZ ∈ S, G={GOOD} ⊆S.

Η κανονική έκφραση που τα περιγράφει είναι η εξής: **"([\w\d\s]|(\\\\)|(\\n)|(\\")|[!@#$%^&*(),.?:;'{}|<>/~`\[\]\-\+\=])*"**.

Μερικά αποδεκτά παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης είναι:

- "test" ->Αποδεκτή συμβολοσειρά, οι τέσσερις λατινικοί χαρακτήρες εμπεριέχονται μέσα σε διπλές αποστρόφους.
- "\"\\" -> Αποδεκτή συμβολοσειρά, οι χαρακτήρες \” είναι αναγνωρισμένοι συνδυασμοί σειρών διαφυγής μέσα στα strings.
- "harris*_*-_/\\n&5" ->Αποδεκτή συμβολοσειρά, η ακολουθία χαρακτήρων εμπεριέχεται μέσα σε διπλές αποστρόφους, και έχουμε ξανά την ακολουθία \\, η οποία είναι αναγνωρισμένος συνδυασμός σειρών διαφυγής μέσα στα strings.

Μερικά λανθασμένα παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης:

- "\" ->Μη αποδεκτή συμβολοσειρά, περιέχει τον χαρακτήρα \, όπου δεν είναι αποδεκτός στο συγκεκριμένο αυτόματο.

BNF έκφραση:

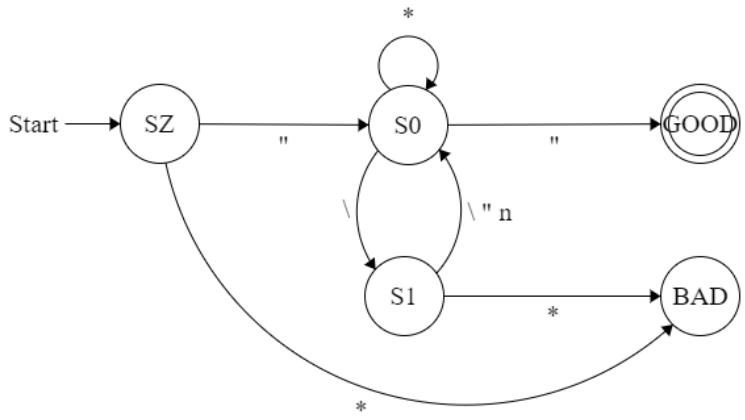
```
strings ::= "" alpha* ""
alpha ::= (letter | digit | symbols | escapechar)
symbols ::= '[' | ']' | '{' | '}' | '(' | ')' | '<' | '>' | '"' | '=' | '|' | '.' | ',' | ';' | '_'
escapechar ::= '\n' | '\\' | '\"'
letter ::= [a-z] | [A-Z]
digit ::= [0-9]
```

Ο Πίνακας Μεταβάσεων του αυτομάτου B είναι ο παρακάτω:

	“	\	n	OTHERS
SZ	S0			
S0	GOOD	S1	S0	S0
S1	S0	S0	S0	
GOOD				

Με τον όρο OTHERS στον πίνακα μετάβασης, εννοούμε όλους τους υπόλοιπους χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου αλλά δεν αναφέρονται στις προηγούμενες στήλες. Επίσης, όπου

υπάρχουν κενά κελία σημαίνει ότι από την αντίστοιχη κατάσταση, με την συγκεκριμένη είσοδο πηγαίνει σε κατάσταση Λάθους.



Σχήμα 2: Αυτόματο string

Κώδικας fsm επιμέρους αυτόματου:

```
SZ: "          ->    S0
    *          ->    BAD
S0: \\         ->    S1
    "          ->    GOOD
    *          ->    S0
S1: \\ " n     ->    S0
    *          ->    BAD
GOOD(OK):
```

3.3 Αριθμητικά Κυριολεκτικά: Ακέραιοι

Οι ακέραιοι περιλαμβάνουν δεκαδικούς, δεκαεξαδικούς και οκταδικούς ακέραιους. Οι δεκαδικοί ακέραιοι ξεκινάνε με μη μηδενικό ψηφίο (1-9) και ακολουθεί προαιρετικά αριθμός ψηφίων 0-9. Για τους δεκαεξαδικούς αριθμούς ο πρώτος χαρακτήρας είναι υποχρεωτικά το μηδέν (0), στη συνέχεια ακολουθεί ο λατινικός χαρακτήρας x πεζός ή κεφαλαίος (xX), και τέλος ακολουθούν ένα ή περισσότερα δεκαεξαδικά ψηφία (0-9 A-F). Τέλος, ένας οκταδικός ακέραιος ξεκινάει με 0 και ακολουθεί ένα ή περισσότερα οκταδικά ψηφία (0-7).

Οι ακέραιοι αναγνωρίζονται και επιστρέφονται ως tokens με τον αναγνωριστικό όνομα integer.

Η κανονική έκφραση που τους περιγράφει είναι η εξής: **([1-9]\d*\n)|(0[x|X](\d|[A-F])+\n)|(0[0-7]*\n)**.

Έστω Γ το πεπερασμένο αυτόματο περιγραφής αναγνωριστικών, τότε θα περιγράφεται ως εξής Γ=(N, Σ, δ, N1, G), όπου N={N1, N1, N3, N4, N5, N6, GOOD}, Σ={0-9, x, X, A-F, \n}, N1∈N, G={GOOD} ⊆N.

Μερικά αποδεκτά παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης είναι:

- 12345 -> Αποδεκτό ως ακέραιος αριθμός
- 00 -> Αποδεκτός ως οκταδικός αριθμός
- 0XA1232 ->Αποδεκτό ως δεκαεξαδικός αριθμός

Μερικά λανθασμένα παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης:

- 0x -> Λανθασμένο, καθώς μετά το x πρέπει να ακολουθεί τουλάχιστον ένα ψηφίο (0-9) ή κάποιο γράμμα ανάμεσα στα A-F.

- 0139 -> Λανθασμένο, εφόσον οι ακέραιοι δεν ξεκινάνε με μηδέν γίνεται έλεγχος για το εάν ο αριθμός είναι οκταδικός, ωστόσο επειδή περιέχει το ψηφίο 9 δεν είναι.

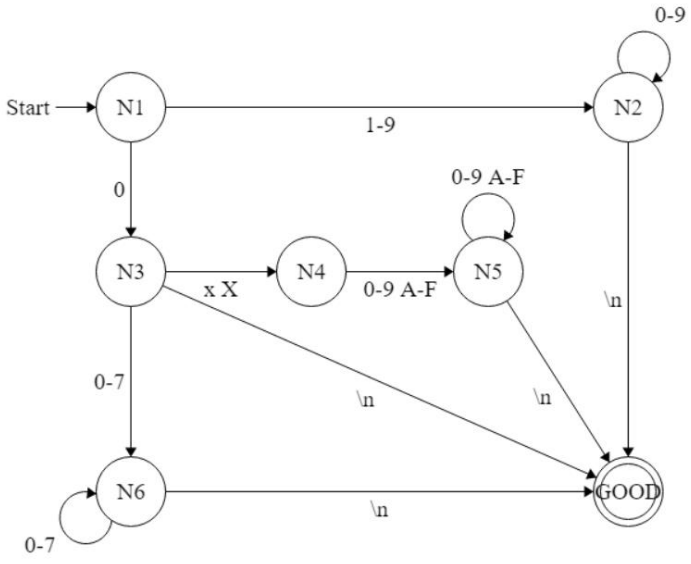
BNF έκφραση:

integer ::= natural Digits* | Zero ((Hex1 (Digits | Hex2)+) |octal *)
Hex1 ::= [x X]
Hex2 ::= [A-F]
Zero ::= '0'
Digits ::= [0-9]
natural ::=[1-9]
octal ::= [0-7]

Ο Πίνακας Μεταβάσεων του αυτομάτου Γ είναι ο παρακάτω:

	0	1-7	8-9	A-F	x X	\n	OTHERS
N1	N3	N2	N2				
N2	N2	N2	N2			GOOD	
N3	N6	N6			N4	GOOD	
N4	N5	N5	N5	N5			
N5	N5	N5	N5	N5		GOOD	
N6	N6	N6				GOOD	
GOOD							

Με τον όρο OTHERS στον πίνακα μετάβασης, εννοούμε όλους τους υπόλοιπους χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου αλλά δεν αναφέρονται στις προηγούμενες στήλες. Επίσης, όπου υπάρχουν κενά κελία σημαίνει ότι από την αντίστοιχη κατάσταση, με την συγκεκριμένη είσοδο πηγαίνει σε κατάσταση Λάθους. Ακόμα, όταν στο αλφάβητο αναγράφουμε δύο χαρακτήρες με κενό ενδιάμεσα, σημαίνει είτε ο ένας, είτε ο άλλος, δηλαδή το κενό λειτουργεί ως το λογικό OR. Όσον αφορά το \n, το χρησιμοποιούμαι σαν το κενό (ε), για να μεταβούμε σε μία συγκεκριμένη τελική κατάσταση.



Σχήμα 3: Αυτόματο integer

Κώδικας fsm επιμέρους αυτόματου:

START=N1
N1: 1-9 ->N2
0 ->N3

```
N2:     0-9     ->N2
        \n      ->GOOD
N3:     x X     ->N4
        \n      ->GOOD
        0-7     ->N6
N4:     0-9 A-F ->N5
N5:     0-9 A-F ->N5
        \n      ->GOOD
N6:     0-7     ->N6
        \n      ->GOOD
GOOD(OK):
```

Στον κώδικα fsm χρησιμοποιείται το **\n**, ως εναλλακτικό «κενό». Με αυτό τον τρόπο μπορούμε να μεταβαίνουμε σε μία τελική κατάσταση.

3.4 Αριθμητικά Κυριολεκτικά: Αριθμοί Κινούμενης Υποδιαστολής

Οι αριθμοί κινούμενης υποδιαστολής αποτελούνται από το ακέραιο και το δεκαδικό μέρος, τα οποία διαχωρίζονται με μία τελεία (.) Το ακέραιο και δεκαδικό μέρος περιγράφεται βάση τους κανόνες του δεκαδικού αριθμητικού συστήματος, οι οποίοι περιγράφτηκαν στην υποενότητα 3.3 Αριθμητικά Κυριολεκτικά: Ακέραιοι. Ωστόσο, στους αριθμούς κινούμενης υποδιαστολής υπάρχει και η δυνατότητα ορισμού δυνάμεων, με τη χρήση του λατινικών χαρακτήρων e ή E.

Οι αριθμοί κινητής υποδιαστολής αναγνωρίζονται και επιστρέφονται ως tokens με το αναγνωριστικό όνομα float.

Έστω Δ το πεπερασμένο αυτόματο περιγραφής αναγνωριστικών, τότε θα περιγράφεται ως εξής Δ=(F, Σ, δ, FZ, G), όπου F={F1, F2, F3, F4, F5, F6, F7, GOOD}, Σ={0-9, E, e, -, ., \n}, F1∈F, G={GOOD} ⊆F.

Η κανονική έκφραση που τους περιγράφει είναι η εξής: **(0(((\.[\d]+)?([Ee]-?[\d]+))|(\.[\d]+)))|([1-9][\d]*(((\.[\d]+)?([Ee]-?[\d]+))|(\.[\d]+)))\n**

Μερικά αποδεκτά παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης είναι:

- 103.0e-0 ->Αποδεκτή συμβολοσειρά, μετά την τελεία ακολουθεί ψηφίο και το πλην είναι μετά το e.
- 103e0 ->Αποδεκτή συμβολοσειρά, το εείναι ανάμεσα σε δύο ψηφία (0-9).
- 103.0 -> Αποδεκτή συμβολοσειρά.

Μερικά λανθασμένα παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης:

- 0e. ->Μη αποδεκτή συμβολοσειρά, εφόσον η τελεία είναι μετά το εκαι δεν υπάρχει το δεκαδικό μέρος
- 5 -> Μη αποδεκτή συμβολοσειρά, δεν είναι αριθμός κινητής υποδιαστολής, αλλά ακέραιος
- 1.e3 -> Μη αποδεκτή συμβολοσειρά, διότι θα έπρεπε να υπάρχει κάποιο ψηφίο (0-9) ανάμεσα στην τελεία (.) και στο e.

BNF έκφραση:

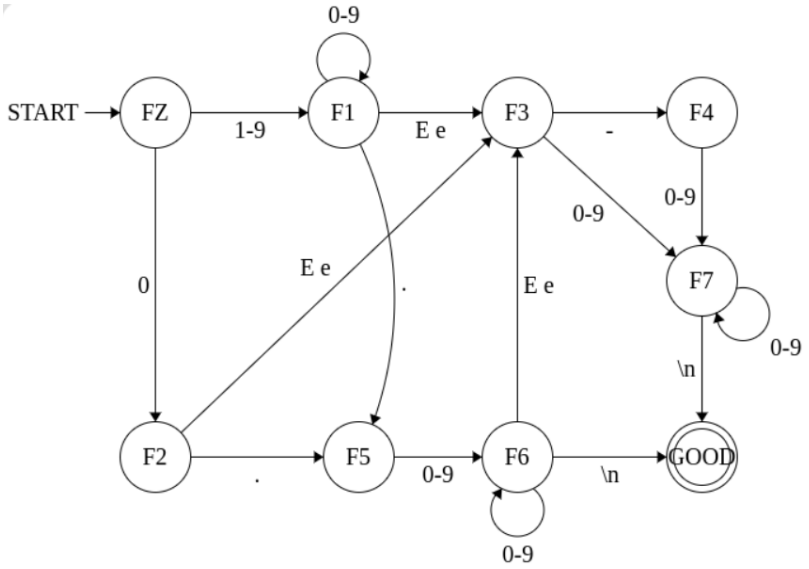
```
float ::= (0 ( ( ('.' digit+)? ([Ee]-?[digit]+) ) | ('.' digit+) ) ) | ( [1-9] digit* ( ( ( '.' digit+)? ([Ee]-? digit+) ) | ('.' digit+ ) ) )
digit ::= [0-9]
```

Ο Πίνακας Μεταβάσεων του αυτομάτου Δ είναι ο παρακάτω:

	0	1-9	e E	.	-	\n	OTHERS
FZ	F2	F1					

F1	F1	F1	F3	F5		
F2			F3	F5		
F3	F7	F7			F4	
F4	F7	F7				
F5	F6	F6				
F6	F6	F6	F3			GOOD
F7	F7	F7				GOOD
GOOD						

Με τον όρο OTHERS στον πίνακα μετάβασης, εννοούμε όλους τους υπόλοιπους χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου αλλά δεν αναφέρονται στις προηγούμενες στήλες. Επίσης, όπου υπάρχουν κενά κελία σημαίνει ότι από την αντίστοιχη κατάσταση, με την συγκεκριμένη είσοδο πηγαίνει σε κατάσταση Λάθους. Ακόμα, όταν στο αλφάβητο αναγράφουμε δύο χαρακτήρες με κενό ενδιάμεσα, σημαίνει είτε ο ένας, είτε ο άλλος, δηλαδή το κενό λειτουργεί ως το λογικό OR. Όσον αφορά το **\n**, το χρησιμοποιούμαι σαν το κενό (ε), για να μεταβούμε σε μία συγκεκριμένη τελική κατάσταση.



Σχήμα 4: Αυτόματο float

Κώδικας fsm επιμέρους αυτόματου:

```
START=FZ
FZ:     1-9 ->  F1
        0       ->      F2
        *       ->      BAD
F1:     0-9     ->      F1
        E e     ->      F3
        .       ->      F5
        *       ->      BAD
F2:     E e      ->     F3
        .       ->      F5
        *       ->      BAD
F3:     -       ->      F4
        0-9     ->      F7
        *       ->      BAD
F4:     0-9     ->      F7
        *       ->      BAD
```

```
F5:     0-9     ->      F6
        *       ->      BAD
F6:     0-9     ->      F6
        E e     ->      F3
        \n      ->      GOOD
        *       ->      BAD
F7:     0-9     ->      F7
        \n      ->      GOOD
        *       ->      BAD
GOOD(OK):
```

Στον κώδικα fsm χρησιμοποιείται το **\n**, ως εναλλακτικό «κενό». Με αυτό τον τρόπο μπορούμε να μεταβαίνουμε σε μία τελική κατάσταση.

4. Λεξήματα που αναγνωρίζονται αλλά δεν επιστρέφονται ως tokens

4.1 White_spaces χαρακτήρες

White_spaces είναι το αναγνωριστικό όνομα, των ακολουθιών διαχωριστικών χαρακτήρων (κενά, tabs ή οι συνδυασμοί τους), που χρησιμοποιούνται για το διαχωρισμό λεξημάτων, οι οποίοι αναγνωρίζονται από πρότυπα, αλλά δεν επιστρέφονται ως tokens.
Η κανονική έκφραση που τους περιγράφει είναι η εξής: **\s+**.
Έστω Ε το πεπερασμένο αυτόματο περιγραφής αναγνωριστικών, τότε θα περιγράφεται ως εξής Ε=(S, Σ, δ, SZ, G), όπου S={SZ, Whitespace}, Σ={0-9, x, X, A-F, \n}, SZ∈S, G={Whitespace} ⊆S.
Παράδειγμα αποδεκτής συμβολοσειράς βάση της Κανονικής Έκφρασης είναι:

- -> Αποδεκτό, \s\t\t

Παράδειγμα λανθασμένης συμβολοσειράς βάση της Κανονικής Έκφρασης:

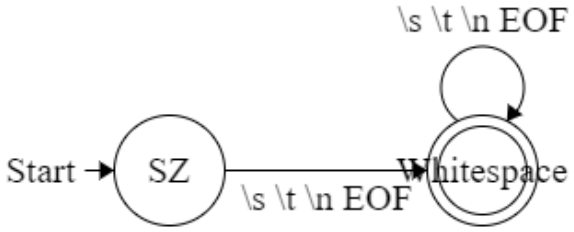
- ABC 123 -> Λανθασμένο, αφού περιέχει λατινικούς χαρακτήρες και ψηφία.

BNF έκφραση:
whitespace ::= '\s'+

Ο Πίνακας Μεταβάσεων του αυτομάτου Ε είναι ο παρακάτω:

	\s	\t	\n	EOF	OTHERS
SZ	Whitespace	Whitespace	Whitespace	Whitespace	
Whitespace	Whitespace	Whitespace	Whitespace	Whitespace	

Με τον όρο OTHERS στον πίνακα μετάβασης, εννοούμε όλους τους υπόλοιπους χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου αλλά δεν αναφέρονται στις προηγούμενες στήλες. Επίσης, όπου υπάρχουν κενά κελία σημαίνει ότι από την αντίστοιχη κατάσταση, με την συγκεκριμένη είσοδο πηγαίνει σε κατάσταση Λάθους.



Σχήμα 5: Αυτόματο white_spaces

Κώδικας fsm επιμέρους αυτόματου:

```
START=SZ
SZ:     \n \t \s EOF     ->SZ
        \n \t \s EOF     ->Whitespace
Whitespace(OK):
```

4.2 Σχόλια-Comment

Comment είναι το αναγνωριστικό όνομα των δύο ειδών σχολίων, τα οποία αναγνωρίζονται βάση προτύπου, αλλά δεν επιστρέφονται ως tokens. Τα σχόλια μπορεί να είναι μίας γραμμής, όπου υποχρεωτικά ξεκινάνε με δύο χαρακτήρες '/' στη σειρά, είτε μπορεί να είναι σχόλια πολλαπλών γραμμών, όπου αρχίζουν αποκλειστικά με τους χαρακτήρες '/*' και ολοκληρώνονται με τους χαρακτήρες '*/'.

Η κανονική έκφραση που τους περιγράφει είναι η εξής: **(\/{2}.*(\n|$))|(\/*(.|\s)**\/)**.

Έστω Z το πεπερασμένο αυτόματο περιγραφής αναγνωριστικών, τότε θα περιγράφεται ως εξής Z=(C, Σ, δ, SZ, G), όπου C={SZ, C1, C2, C3, C4, Comment}, Σ={a-z, A-Z, 0-9, ! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~}, SZ∈C, G={Comment} ⊆C.

Μερικά αποδεκτά παραδείγματα συμβολοσειρών βάση της Κανονικής Έκφρασης είναι:

- //Despina -> Αποδεκτό ως σχόλιο μίας γραμμής
- /*
  Padelis
  */ -> Αποδεκτό, ως σχόλιο πολλαπλών γραμμών

Λανθασμένο παραδείγματα συμβολοσειράς βάση της Κανονικής Έκφρασης:

- /* Met3_4 -> Λανθασμένο, καθώς δεν ολοκληρώνεται το σχόλιο πολλαπλών γραμμών, με την ακολουθία */.

BNF έκφραση:

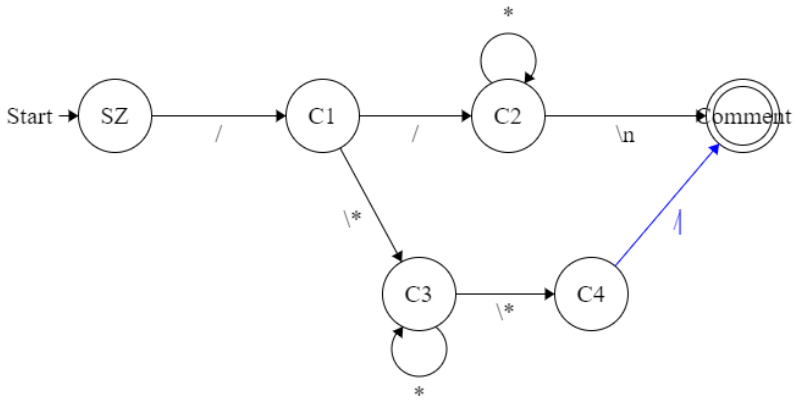
```
comment ::= (\/{2}alpha*('\n'|'EOF'))|(\/\*(alpha|'\s')*\*\/)
alpha ::= (letter | digit | symbols | escapechar)
symbols ::= '[' | ']' | '{' | '}' | '(' | ')' | '<' | '>' | '"' | '=' | '|' | '.' | ',' | ';' | '_'
escapechar ::= '\n' | '\\' | '\"'
letter ::= [a-z] | [A-Z]
digit ::= [0-9]
```

Ο Πίνακας Μεταβάσεων του αυτομάτου Z είναι ο παρακάτω:

	/	*	\n	OTHERS
SZ	C1			

C1	C2	C3		
C2	C2	C2	Comment	C2
C3	C3	C4	C3	C3
C4	Comment			
Comment				

Με τον όρο OTHERS στον πίνακα μετάβασης, εννοούμε όλους τους υπόλοιπους χαρακτήρες, οι οποίοι ανήκουν στο αλφάβητο του αυτομάτου αλλά δεν αναφέρονται στις προηγούμενες στήλες. Επίσης, όπου υπάρχουν κενά κελία σημαίνει ότι από την αντίστοιχη κατάσταση, με την συγκεκριμένη είσοδο πηγαίνει σε κατάσταση Λάθους.



Σχήμα 6: Αυτόματο comment

Κώδικας fsm επιμέρους αυτόματου:

```
START=SZ
SZ:     /           -> C1
        *           -> BAD
C1:     /           -> C2
        \*          -> C3
        *           -> BAD
C2:     *           -> C2
        \n EOF      -> GOOD
C3:      *          -> C3
        \*          -> C4
C4:      /          -> GOOD
        *           -> C3
GOOD(OK):
```

5. Πίνακας Μεταβάσεων Ενιαίου Αυτομάτου

Για τη δημιουργία του ενιαίου πεπερασμένου αυτομάτου, θα ενώσουμε όλα τα στοιχεία των επιμέρους αυτομάτων, με σκοπό να ορίσουμε το ενιαίο αυτόματο, την κανονική έκφρασή του, καθώς και τον πίνακα μεταβάσεων.

Έστω Ω το ενιαίο πεπερασμένο αυτόματο, τότε θα περιγράφεται ως εξής Ω=(Q, Σ, T, SZ, G), όπουQ={SZ, I2, S0, S1, N2, N3, N4, N5, N6, F3, F4, F5, F6, F7, IDENTIFIER, STRING, INTEGER, FLOAT, COMMENT, WHITESPACE }, Σ={a-z, A-Z, 0-9, ! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~}, T: Q×Σ ->Q συνάρτηση μετάβασης, SZ είναι η αρχική κατάσταση και G ={IDENTIFIER, STRING, INTEGER, FLOAT, COMMENT, WHITESPACE} είναι υποσύνολο του Q.

Η Κανονική έκφραση που περιγράφει το ενιαίο αυτόματο Ω είναι η εξής: **(([a-z]|[A-Z]|_)+(\d|[a-z]|[A-Z]|_)*\n)|("([\w\d\s]|(\\\\)|(\\n)|(\\")|[!@#$%^&*(),.?:;'{}|<>/~`\[\]\-\+\=])*")|(([1-9]\d*\n)|(0[x|X](\d|[A-F])+\n)|(0[0-7]*\n))|((0(((\.[\d]+)?([Ee]-?[\d]+))|(\.[\d]+)))|([1-9][\d]*(((\.[\d]+)?([Ee]-?[\d]+))|(\.[\d]+)))\n)|(\s+)|((\/{2}.*(\n|$))|(\/*(.|\s)**\/))**

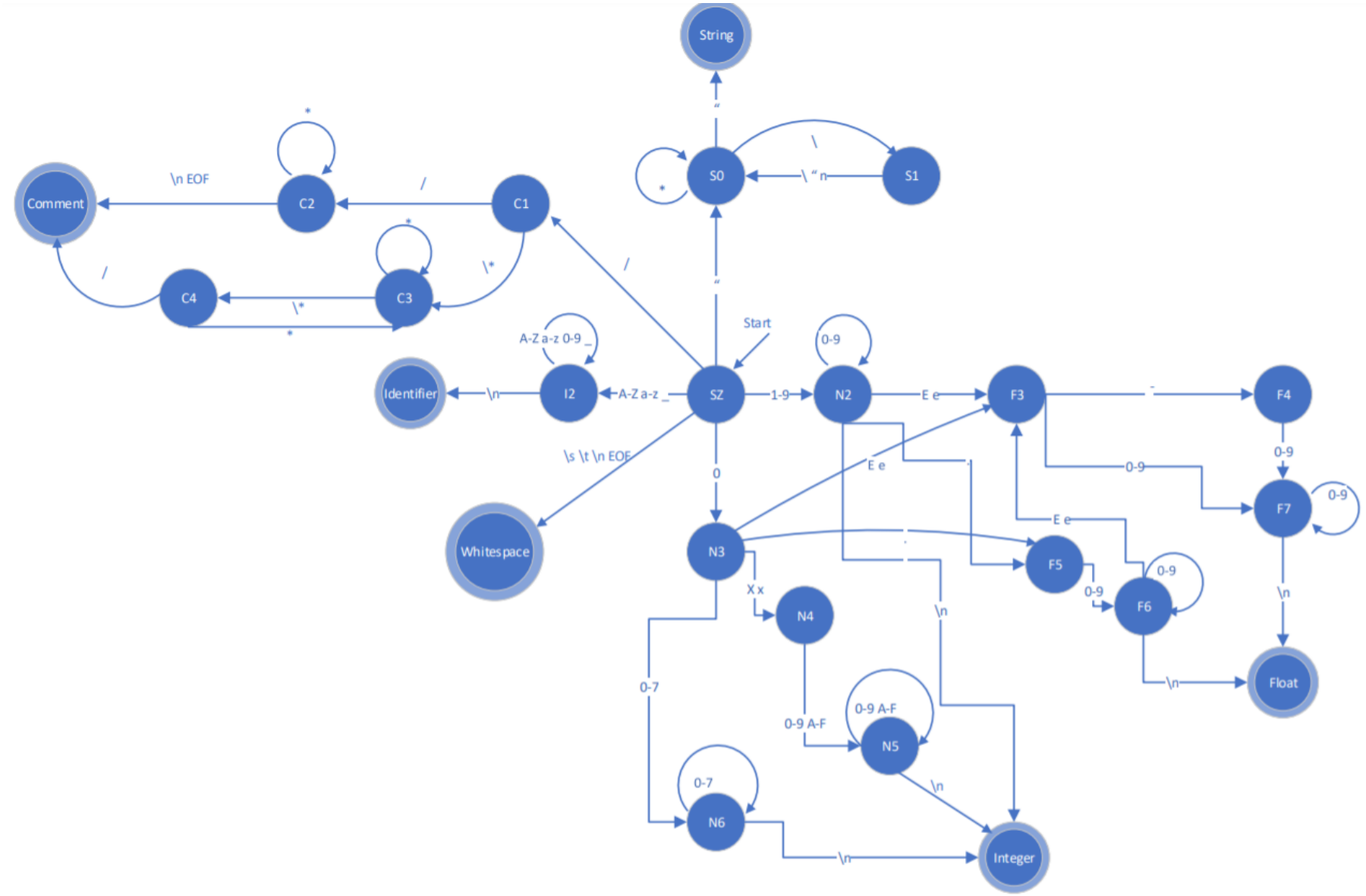
Για λόγους εξοικονόμησης χώρου στον Πίνακα Μεταβάσεων, τις τελικές καταστάσεις θα τις αναγράφουμε όπως φαίνεται από κάτω :

- IDENTIFIER -> G1
- STRING ->G2
- INTEGER ->G3
- FLOAT ->G4
- COMMENT ->G5
- WHITESPACE ->G6

Στην πρώτη γραμμή του πίνακα μεταβάσεων αναγράφονται όλοι οι χαρακτήρες του αλφαβήτου. Στην τελευταία στήλη υπάρχει ο χαρακτήρας OTHERS, ο οποίος συμπεριλαμβάνει όλους του χαρακτήρες του αλφαβήτου, οι οποίοι δεν έχουν αναφερθεί στις προηγούμενες στήλες. Όσον αφορά την πρώτη στήλη του πίνακα, περιέχει όλες τις καταστάσεις του ενιαίου αυτόματου. Όπου υπάρχουν κενά κελιά σημαίνει ότι, η εκάστοτε κατάσταση με το συγκεκριμένο σύμβολο οδηγεί σε αδιέξοδο. Τέλος, στις τελευταίες έξι γραμμές είναι οι τελικές μας καταστάσεις, οι οποίες δεν μεταβαίνουν ούτε σε άλλες καταστάσεις αλλά ούτε και στον εαυτό της, όποτε όλα τα κελιά στις συγκεκριμένες γραμμές είναι κενά.

	0	1-7	8-9	A-D	E	F	G-W	X	Y-Z	a-d	e	f-m	n	o-w	x	y-z	-	_	/	.	*	“	\t	\s	EOF	\n	OTHERS
SZ	N3	N2	N2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2		I2	C1			S0	G6	G6	G6	G6	
C1																			C2		C2						
C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	G6	G6	C2
C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C4	C3	C3	C3		C3	C3
C4	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	G6	C3	C3	C3	C3	C3		C3	C3
I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2	I2		I2								G1	
S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S0	S1	S0	S0	G2	S0	S0			S0
S1													S1						S1			S1					
N2	N2	N2	N2		F3						F3									F5						G3	
N3	N6	N6			F3			N4			F3				N4					F5							
N4	N5	N5	N5	N5	N5	N5																					
N5	N5	N5	N5	N5	N5	N5																				G3	
N6	N6	N6																								G3	
F3	F7	F7	F7														F4										
F4	F7	F7	F7																								
F5	F6	F6	F6																								
F6	F6	F6	F6		F3						F3															G4	
F7	F7	F7	F7																							G4	
G1																											
G2																											
G3																											
G4																											
G5																											
G6																											

Πίνακας Μετάβασης Ενιαίου Αυτόματου.



Σχήμα 7: Ενιαίο Αυτόματο

6. Κώδικας FSM και σχολιασμός

```
START=SZ
SZ:     A-Z a-z _       -> I2
        "               -> S0
        0               -> N3
        1-9             -> N2
        /               -> C1
        \n \t EOF \s    -> WHITESPACE
        *               -> BAD
C1:     /               -> C2
        \*              -> C3
        *               -> BAD
C2:     \n EOF          -> COMMENT
        *               -> C2
C3:     \*              -> C4
        *               -> C3
C4:     /               -> COMMENT
        \*              -> C4
        *               -> C3
I2:     A-Z a-z 0-9 _   -> I2
        \n              -> IDENTIFIER
        *               -> BAD
S0:     \\              -> S1
        "               -> STRING
        *               -> S0
S1:     \\ " n          -> S0
        *               -> BAD
N2:     0-9             -> N2
        E e             -> F3
        .               -> F5
        \n              -> INTEGER
        *               -> BAD
N3:     X x             -> N4
        0-7             -> N6
        E e             -> F3
        .               -> F5
        \n              -> INTEGER
        *               -> BAD
N4:     0-9 A-F         -> N5
        *               -> BAD
N5:     0-9 A-F         -> N5
        \n              -> INTEGER
        *               -> BAD
```

N6:	**0-7**	**-> N6**
	\n	**-> INTEGER**
	*****	**-> BAD**
F3:	**-**	**-> F4**
	0-9	**-> F7**
	*****	**-> BAD**
F4:	**0-9**	**-> F7**
F5:	**0-9**	**-> F6**
	*****	**-> BAD**
F6:	**0-9**	**-> F6**
	E e	**-> F3**
	\n	**-> FLOAT**
	*****	**-> BAD**
F7:	**0-9**	**-> F7**
	\n	**-> FLOAT**
	*****	**-> BAD**

IDENTIFIER(OK):
STRING(OK):
INTEGER(OK):
FLOAT(OK):
COMMENT(OK):
WHITESPACE(OK):

7. Μηχανισμοί που δημιουργήθηκαν για τις μαζικές δοκιμές

7.1 combinations.c

Αρχικά, δημιουργήθηκε το πρόγραμμα combinations.c, το οποίο δέχεται ως πρώτο όρισμα ένα αρχείο με το αλφάβητο που μας ενδιαφέρει και ως δεύτερο όρισμα ένα αρχείου εξόδου το οποίο μπορεί να υπάρχει ή και όχι. Στην περίπτωση που υπάρχει ρωτάει για override ή όχι. Παράγει το νέο αρχείου όπου περιέχει λέξεις βάση του αλφαβήτου του πρώτου ορίσματος. Το αρχείο εισόδου στην πρώτη γραμμή πρέπει να έχει το πλήθος των υπόλοιπων γραμμών που ακολουθούν (δηλαδή το πλήθος του αλφαβήτου).
Παράδειγμα ενός αποδεκτού αρχείου εισόδου είναι το παρακάτω:

```
14
0
1
2
3
4
5
6
7
8
9
```

E

e

.

-

Κώδικας του combinations.c:

```
/* combinations.c */
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include <time.h>
#define STRSIZE 6   // Το μέγεθος τον string που θα δημιουργηθούν
#define TOTAL 30    // Το πλήθος τον strings που θα δημιουργηθούν
// Το STRSIZE καλό είναι να έχει μικρό αριθμό


// Έχει 2 παραμέτρους char pointer και ανοίγει κάπιο αρχείο
// είτε για εγγραφή είτε για ανάγνωση βάση του δεύτερου
// ορίσματος και το όνομα είναι η πρώτη παράμετρος
FILE *openFile(char *fname, char *opt){ // orpt gets w or r

    FILE *file = fopen(fname,opt);
    return file;
}



// Έχει 2 παραμέτρους, το αρχείο εισόδου και το αρχείο εξόδου
// ως πρώτη και δεύτερη παράμετρος αντοίστηχα
int main (int argc, char *argv[]){

    // Ελέγχει αν υπάρχουν όλοι οι παράμετροι
    if ( argc != 3) // it is always 1 so plus 2 is 3
    {
        puts("Usage: combinations <possible input file> <output>");
        exit(1);
    }


    FILE *in, *out;
    char ch;
    char accept = 'y';

    // Ανοίγει για ανάγνωση το αρχείο
    in = openFile(argv[1],"r");

    // Ελέχγει αν υπάρχει το αρχείο εισόδου
    if (!in)
    {
        printf("file \"%s\" dosen't exists...\n",argv[1]);
        exit (1);
    }

    // Ελέγχει αν υπάρχει το αρχείο εξόδου και αν υπάρχει ρωτάει
    // αν επιθυμείται να το κάνετε override
    if (!access(argv[2],F_OK))
    {
```

```
    printf("file \"%s\" exists... override it (y/n)?: ", argv[2]);
    scanf("%c", &ch);
    if ( ch != accept) exit(1);
}

// Ανοίγει για εγγραφή το αρχείο
out = openFile(argv[2],"w");


int len;

// Διαβάζει την πρώτη γραμμή του αρχείου όπου είναι το πλήθος
// τον αριθμών του αλφαβήτου (δηλαδή, των υπόλοιπων γραμμών που
// ακολουθούν) και γίνεται allocate βάση του αριθμού
fscanf(in,"%d",&len);

char *ptr = (char *)malloc(len*sizeof(int));

if (!ptr){
    perror("Couldn't allocate memory");
    exit(2);
}

int i = 0;

// Διαβάζει len γράμματα που είναι στο αρχείο ή λιγότερα απο len
while (i < len){

    ch = fgetc(in);

    if (ch == -1){
        printf("Ended before %d at %d\n", len, i);
        len = i;
        break;
    }

    // Αν είναι tab ή newline τα προσπερνάει (οχι τα spaces)
    if ( ch == '\n' || ch == '\t') continue;
    ptr[i]=ch;
    i++;
}

// Κλείνει το αρχείο ανάγνωσης
fclose(in);


// Στο str θα αποθηκευθή η τυχαία λέξη που θα παράξει
char str[STRSIZE];
srand(time(0));
int num, j, tmp;

// Παράγει συνολικά TOTAL λέξεις
for ( i = 0 ; i < TOTAL ; i++){


    // Επιλέγετε ένα τυχαίο μέγεθος συμβολοσειράς
    // μικρότερο ή ίσος με STRSIZE
    num = ( rand() % STRSIZE )+1;


    // Επιλέγει τυχαία γράμματα και τα προσθέτει στο string
```

```
        for (j = 0; j < num; j++){
            tmp = rand() % len;
            str[j] = ptr[ tmp ];
        }

        // Το str πρέπει να τελειώνει με null
        num = (num >= STRSIZE) ? num-1 : num;
        str[num]='\0';

        // Γράφει την λέξει στο αρχείο και έπειτα αλλάζει γραμμή
        fputs(str,out);
        fputs("\n",out);
    }

    // Εμφανίζει μήνυμα στο stdout πως τελείωσε
    puts("########### END ###########");

    // Κλείνη το αρχείο και ελευθερώνει την μνήμη
    fclose(out);
    free(ptr);


    return 0;
}
```

7.2 testbench.sh

Επιπρόσθετα, το script testbench.sh δέχεται ως πρώτη παράμετρο, το αρχείο με extension .fsm, ως δεύτερη παράμετρο το αρχείο εισόδου το οποίο περιέχει λέξεις προς έλεγχο, ως τρίτο όρισμα το αρχείο εξόδου και ως τέταρτο όρισμα y ή n, ανάλογα με το αν θέλουμε να τελειώνει με αλλαγή γραμμής ή όχι στο τέλος της λέξης αντίστοιχα. Προαπαιτούμενο να υπάρχει το αρχείο fsm.out

Κώδικας testbench.sh:

```
#!/bin/bash

# Ελέγχει αν υπάρχουν 4 ορίσματα
[ $# -ne 4 ] && echo "Usage:testbench <fsm file> <file input> <file output> <\\n
at the end (y/n)>" && exit 1

# Ελέγχει αν υπάρχουν τα αρχεία
[ -f $1 ]  || ( echo "Fsm file \"$1\" dosen't exits..." ; exit 2 )

[ -f $2 ]  || ( echo "Input file \"$2\" dosen't exits..." ; exit 2 )

# Ελέγχει αν υπάρχει το αρχείο εξόδου και αν υπάρχει ρωτάει αν θέλετε να γίνει
override
[ -f $3 ]  &&  read -p "File \"$3\" exits... override it?(y/n): " opt && [ $opt
!= "y" ] && exit 2


# Εάν υπάρχει διαγράφετε, αν δεν θέλατε να γίνει η διαγραφή το πρόγραμμα θα είχε
βγεί
# και δεν θα εκτελούταν ποτέ αυτή η εντολή
[ -f $3 ] && rm $3

# Δημιουργεί το αρχείο
touch $3

# Τελειώνει με new line το οποίο καθορίζεται από το τέταρτο όρισμα
```

```
end="\n"

# Αν το τέταρτο όρισμα είναι 'n' τότε δεν τελειώνει με new line
[ $4 = "n" ] && end=""

# Διαβάζει κάθε γραμμή του αρχείου, την εκτυπώνει στο αρχείο εξόδου
# βάση του τρίτου ορίσματος, προσθέτει το ' : ', στέλνει ως
# είσοδο στο fsm.out την γραμμή που δίαβασε για το αρχείο με την
# κατάληξει .fsm που έχει ορισθεί στο πρώτο όρισμα και κάνει
# redirect το stdout και stderr στο αρχείου εξόδου.
while read -r line
do
    echo -n "${line} : " >> $3
    echo -ne "${line}${end}" | fsm.out $1 >> $3 2>&1
done < "$2"

# Εκτυπώνει ότι τελείωσε και εμφανίζει τα αποτελέσματα του αρχείου
echo "######## end ########"
echo "######## display $3 ########"
cat $3
exit 0
```

7.3 addToStr.sh

Το script addToStr.sh δέχεται ως πρώτη παράμετρο ένα αρχείο εισόδου με λέξεις και ως δεύτερη παράμετρο ένα αρχείο εξόδου. Σε όλες τις λέξεις στο τέλος και στην αρχή προσθέτει double quotes (")

Προϋποθέσεις, να υπάρχουν:
Η εντολή yes, testbench.sh, fsm.out, addToStr.sh, combinations.out και τα αρχεία strStates.txt, intStates.txt, flaotStates.txt, idStates.txt στον ίδιο κατάλογο όπου είναι και το αρχείο με κατάληξη .fsm.

Κώδικας addToStr.sh:

```
#!/bin/bash

# Δέχεται 3 ορίσματα
[ $# -ne 3 ] && echo "Usage: addToStr <input file> <output file> <option>" &&
echo "options: 1 -> \"...\", 2 -> /*...*/, 3 -> //..." && exit 1

# Ελέγχει τα αρχεία αναλόγος
[ -f $1 ] || (echo "File \"$1\" dosen't exists..."; exit 2)
[ -f $2 ] && read -p "File \"$2\" exits... override it?(y/n): " opt && [ $opt !=
"y" ] && exit 2

# Αν υπάρχει το αρχείο το διαγράφει
[ -f $2 ] && rm $2

# Δημιουργεί το αρχείο
touch $2

if [ $3 -eq 1 ]
then
     beg="\""
     end="\""
elif [ $3 -eq 2 ]
then
    beg="/*"
    end="*/"
elif [ $3 -eq 3 ]
```

```
then
    beg="//"
    end=""
fi

# Διαβάζει κάθε γραμμή του αρχείου και προσθέτει στο
# τέλος και στην αρχεί αναλόγος του option.
while read -r line
do
    echo "$beg${line}$end" >> $2
done < "$1"

# Τυπώνει μήνυμα στο stdout ότι τερμάτησε
echo "#### END ####"
```

7.4 testing.sh

Τέλος, το script testing.sh δέχεται ως όρισμα το αρχείο με extension .fsm και κάνει μαζικό έλεγχο σε συνεργασία με τα προηγούμενα εργαλεία και παράγει 8 αρχεία, όπου τα 4 είναι οι λέξεις που δημιουργήθηκαν από το εργαλείο combinations.out και τα άλλα 4 είναι αυτά που παράχθηκαν από το εργαλείο testbench.sh.

Κώδικας testing.sh:

```
#!/bin/bash

# Δέχεται ως παράμετρο το αρχείο .fsm
[ $# -ne 1 ] && echo "Usage: testing <fsm file>" && exit 2

# Ελέγχει αν υπάρχει
[ -f $1 ] || ( echo "File \"$1\" dosen't exits..." ; exit 1 )


# Ελέγχει αν υπάρχουν τα αρχεία τα οποία περιέχουν το αλφάβητο για κάθε
διαφορετική
# λέξη που θέλουμε να σχηματίσουμε
[ -f strStates.txt ] || ( echo "File \"strStates.txt\" dosen't exits..." ; exit 1
)
[ -f idStates.txt ] || ( echo "File \"idStates\" dosen't exits..." ; exit 1 )
[ -f floatStates.txt ] || ( echo "File \"floatStates.txt\" dosen't exits..." ;
exit 1 )
[ -f intStates.txt ] || ( echo "File \"intStates.txt\" dosen't exits..." ; exit 1
)

# Η εντολή yes είναι build-in εντολή η οποία έχει ως output το γράμμα 'y'
πολλαπλές φορές γιατί το πρόγραμμα
# combinations.out, το addToStr.sh και το testbench.sh περιμένουν input από τον
user στην περίπτωση που το
# αρχείο εξόδου υπάρχει.

# Εκτελέιται το πρόγραμμα combinations.out και παράγει το αρχείο tmp.txt. Έπειτα
το script addToStr.sh
# του προσθέτει στο τέλος και στην αρχή double quotes (") και τέλος καλείται το
script testbench.sh
# με πρώτο όρισμα την πρώτη παράμετρο του παρών script και έπειτα το αρχείο
εισόδου που παρήγαγε το addToStr.sh
# output και τέλος χωρίς αλλαγή γραμμής και τέλος διαγράφεται το ενδιάμεσο αρχείο
tmp.txt που παράχθηκε προσορινά
yes | combinations.out strStates.txt tmp.txt && yes | addToStr.sh tmp.txt
strInput.txt 1 && yes | testbench.sh $1 strInput.txt strOutput.txt n
```

```
[ -f tmp.txt ] && rm tmp.txt

yes | combinations.out strStates.txt tmp.txt && yes | addToStr.sh tmp.txt
lCommInput.txt 3 && yes | testbench.sh $1 lCommInput.txt lCommOutput.txt n
[ -f tmp.txt ] && rm tmp.txt

yes | combinations.out strStates.txt tmp.txt && yes | addToStr.sh tmp.txt
bCommInput.txt 2 && yes | testbench.sh $1 bCommInput.txt bCommOutput.txt n
[ -f tmp.txt ] && rm tmp.txt

# Με την ίδια λογική δουλεύει και αυτό το κομμάτι αλλά χωρίς το addToStr.sh
yes | combinations.out idStates.txt idInput.txt && yes | testbench.sh $1
idInput.txt floatOutput.txt y
yes | combinations.out intStates.txt intInput.txt && yes | testbench.sh $1
intInput.txt intOutput.txt y
yes | combinations.out floatStates.txt floatInput.txt && yes | testbench.sh $1
floatInput.txt floatOutput.txt y
```

7.5 strStates.txt, intStates.txt, floatStates.txt, idStates.txt

Καλό είναι όλα τα εργαλεία που προαναφέρθηκαν, να υπάρχουν σε ένα directory που είναι στην μεταβλητή PATH, έτσι ώστε να μην μεταφέρονται και επιπλέον ένα ακόμα απαραίτητο βήμα είναι πως τα αρχεία strStates.txt, intStates.txt, flaotStates.txt, idStates.txt θα πρέπει να είναι στο ίδιο directory με το αρχείο .fsm. Τα αρχεία floatStates.txt, intStates.txt, strStates.txt, idStates.txt περιέχουν το αλφάβητο που χρειάζεται, όπου η πρώτη γραμμή του κάθε αρχείου περιέχει το πλήθος του αλφαβήτου.

8. Εξαντλητικές Δοκιμές Ορθού Ελέγχου

/*dg#*/ : YES
/*7Puel->q7*/ : YES
/*b1o^^o*/ : YES
/*y(miioT*/ : YES
/*$8U*/ : YES
/*NlSE*/ : YES
/*A*/ : YES
/*m2e>c*/ : YES
/*LR*/ : YES
/**B%&Go*/ : YES
/*2_B{b*/ : YES
/*Sd'?9IoX*/ : YES
/*\i]yM'H!*/ : YES
/*Q*/ : YES
/*/Tf|*/ : YES
/*x]bs?:*/ : YES
/*%@d<gTs*/ : YES
/*2\@*/ : YES
/*+FS,Q$*/ : YES
/*_V2*/ : YES
E.2 : fsm: in NewAll.fsm, state 'bad' input 2 not accepted
93396233e : NO

306248 : YES
E69e02. : fsm: in NewAll.fsm, state 'bad' input \n not accepted
8.E : fsm: in NewAll.fsm, state 'bad' input \n not accepted
5828 : YES
8 : YES
e0E26 : YES
41 : YES
.ee97. : fsm: in NewAll.fsm, state 'bad' input e not accepted
727.. : fsm: in NewAll.fsm, state 'bad' input \n not accepted
9eE618ee : fsm: in NewAll.fsm, state 'bad' input 6 not accepted
0e828e79 : fsm: in NewAll.fsm, state 'bad' input 7 not accepted
4 : YES
5245 : YES
e79E65 : YES
5329148 : YES
494 : YES
4331.9 : YES
215 : YES
h : YES
I : YES
g5KJQTZ : YES
nfBAeQ : YES
f : YES
sQwnX : YES
iu3z : YES
1FTfLm : fsm: in all.fsm, state 'bad' input T not accepted
jFTG0 : YES
RBmpXSdNq : YES
AObrGcq : YES
b5IGQ2A7 : YES
eOI : YES
nTZ : YES
yz : YES
Jpm : YES
BMJ : YES
y9GYCX : YES
bZxipb68N : YES
ZuTWthp : YES
118 : YES
X532b0Ea4 : YES
xfeX1e : YES
AfDeCFF : YES
F81 : YES
Fbdb : YES
E : YES

cxEa0 : YES
7D : fsm: in NewAll.fsm, state 'bad' input \n not accepted
034Ba9 : fsm: in NewAll.fsm, state 'bad' input a not accepted
5aFe8 : fsm: in NewAll.fsm, state 'bad' input F not accepted
xX69CXec : YES
8b91BF90 : fsm: in NewAll.fsm, state 'bad' input 9 not accepted
9 : YES
8CCA : fsm: in NewAll.fsm, state 'bad' input C not accepted
b3d734 : YES
cA70cec : YES
557 : YES
Xf8a39 : YES
BEa : YES
//dg# : YES
//7Puel->q7 : YES
//b1o^^o : YES
//y(miioT : YES
//$8U : YES
//NlSE : YES
//A : YES
//m2e>c : YES
//LR : YES
//*B%&Go : YES
//2_B{b : YES
//Sd'?9IoX : YES
//\i]yM'H! : YES
//Q : YES
///Tf| : YES
//x]bs?: : YES
//%@d<gTs : YES
//2\@ : YES
//+FS,Q$: YES
//_V2 : YES
"dg#" : YES
"7Puel->q7" : YES
"b1o^^o" : YES
"y(miioT" : YES
"$8U" : YES
"NlSE" : YES
"A" : YES
"m2e>c" : YES
"LR" : YES
"*B%&Go" : YES
"2_B{b" : YES
"Sd'?9IoX" : YES

"\i]yM'H!" : fsm: in NewAll.fsm, state 'bad' input] not accepted
"Q" : YES
"/Tf|" : YES
"x]bs?:" : YES
"%@d<gTs" : YES
"2\@" : fsm: in NewAll.fsm, state 'bad' input " not accepted
"+FS,Q$" : YES
"_V2" : YES

9. Ανάλυση Αρμοδιοτήτων

Καρναβάς Απόστολος: Σχεδιασμός του αυτόματου identifier, δημιουργία της BNF έκφρασης του και του κώδικα fsm του. Σχεδιασμός αυτομάτου για τα whitespaces και τα comments. Δημιουργία BNF εκφράσεων για τα αυτόματα strings και comments. Σχεδιασμός ενιαίου αυτόματου.

Λυκούδη Δέσποινα: Σχεδιασμός αυτομάτου integer, δημιουργία της BNF έκφρασης του και του κώδικα fsm του. Δημιουργία πινάκων μετάβασης του ενιαίου αυτομάτου και των επιμέρους. Τεκμηρίωση (σχολιασμός και παραδείγματα) ενοτήτων 1, 2, 3 και 4. Υπεύθυνη για το document.

Οικονομίδης Χαράλαμπος: Σχεδιασμός αυτομάτου για τα strings και δημιουργία του fsm κώδικα του. Εξαντλητικοί έλεγχοι για την σωστή λειτουργιά του κώδικα του ενιαίου αυτομάτου, καθώς και για την ορθή λειτουργία των επιμέρους αυτομάτων. Τελικός έλεγχος παραδοτέου. Υπεύθυνος μέρους Α2.

Πρώιος Παντελεήμων: Σχεδιασμός αυτομάτου float και δημιουργία του κώδικα fsm του. Δημιουργία κώδικα fsm ενιαίου αυτομάτου και του επιμέρους αυτόματου για τα comments. Δημιουργία BNF εκφράσεων για όλα τα επιμέρους αυτόματα. Δημιουργία εργαλείων για μαζικούς ελέγχους και τεκμηρίωση όλης της ενότητας 7.

Ωστόσο, η εργασία έγινε με πνεύμα ομαδικότητας και συνεργασίας. Όλα τα μέλη της ομάδας μας εκφράζαμε ελεύθερα και άμεσα τις ιδέες μας, αλλά και τους ενδοιασμούς μας, ώστε η εργασία μας να είναι όσο το δυνατόν καλύτερη.

10. Βιβλιογραφία & Αναφορές

Οτιδήποτε έχει χρησιμοποιηθεί στην τεκμηρίωση του εγγράφου είναι από το υλικό του μαθήματος, τη συνεπή παρακολούθηση των διαλέξεων της θεωρίας και του εργαστηρίου, αλλά και από το υλικό που κοινοποιείται από τους συμφοιτητές μας, μέσω της πλατφόρμας Συζητήσεις, του μαθήματος Μεταγλωττιστές. Επιπλέον, χρησιμοποιήθηκαν οι ακόλουθες ιστοσελίδες για την δημιουργία και τον έλεγχο διαφόρων μερών της εργασίας. Οι ιστοσελίδες είναι οι ακόλουθες:

- http://www.madebyevan.com/fsm/
- https://regex101.com
- https://www.regexpal.com
- https://regexr.com