

# Μεταγλωττιστές

ΕΡΓΑΣΙΑ 2019-2020

Αργυροπούλου Μαρία | ΑΜ: Π18011

Δημητρίου Δημήτρης | ΑΜ: Π18036

Στεργίου Χρήστος | ΑΜ: Π18147

27/1/2020

## Σχετικά με την εργασία:

Το αρχείο .zip περιέχει το παρόν αρχείο Μεταγλωττιστές.pdf στο οποίο παρατίθενται οι απαντήσεις των ασκήσεων, διευκρινήσεις για τον τρόπο ανάπτυξης της εκάστοτε εφαρμογής και ενδεικτικά runtime screenshots. Ο πηγαίος κώδικας των εφαρμογών βρίσκεται σε φάκελο με το όνομα της άσκησης στην οποία αντιστοιχεί. Οι πληροφορίες για το compilation και execution του κάθε προγράμματος βρίσκονται παρακάτω στην περιγραφή της λύσης του προβλήματος.

## Άσκηση (Α)

Για να γίνει η υλοποίηση ενός Ντετερμινιστικού Αυτομάτου Στοιβάς (ΝΑΣ) πρέπει να δημιουργήσουμε έναν μηχανισμό καταστάσεων μεταβάσεων και χρειαζόμαστε την μνήμη του αυτόματου, με την μορφή στοιβάς (που θα χρησιμοποιούμε για να αποθηκεύουμε τα σύμβολα).

Έτσι μπορούμε να ορίσουμε:

Ένα αυτόματο  $M=(K,T,V,p,k_1,\$,F)$  όπου

1.  $K$  είναι ένα πεπερασμένο **σύνολο καταστάσεων**.  $K = \{k_1, k_2\}$ .
2.  $T$  είναι ένα πεπερασμένο **αλφάβητο εισόδου**.  $T = \{x, y, e\}$ .

Όπου  $e$  συμβολίζει το κενό.

3.  $V$  είναι ένα πεπερασμένο **αλφάβητο συμβόλων στοιβάς**.  $V = \{x, \$\}$ .
4.  $P$  είναι μια **συνάρτηση στοιβάς** από το  $K \times V \times (T \cup \{e\})$  σε πεπερασμένα υποσύνολα του  $K \times V^*$ .
  - 1)  $p(k_1, \$, x) = (k_1, \$x)$
  - 2)  $p(k_1, x, x) = (k_1, xx)$
  - 3)  $p(k_1, x, y) = (k_1, \epsilon)$
  - 4)  $p(k_1, \$, \epsilon) = (k_2, \epsilon)$
5.  $k_1 \in K$  είναι μια **αρχική κατάσταση**.
6.  $\$ \in V$  είναι ένα **αρχικό σύμβολο στοιβάς** το οποίο αρχικά είναι το μόνο σύμβολο στην στοιβά.
7.  $F \subseteq K$  είναι ένα σύνολο **τελικών καταστάσεων**.  $F = \{k_2\}$ .

Η συνάρτηση μπορεί να παρασταθεί σε ένα πίνακα που ονομάζεται **πίνακας ελέγχου**:

V/T	X	y	e
X	BAΛΕ(x)	BΓAΛΕ(x)	
\$	BAΛΕ(x)		k <sub>2</sub>

## ΕΠΟΜΕΝΩΣ Η ΚΥΡΙΑ ΛΟΓΙΚΗ ΜΕ ΤΗΝ ΟΠΟΙΑ ΑΝΑΠΤΥΣΣΕΤΑΙ ΤΟ ΠΡΟΓΡΑΜΜΑ ΕΙΝΑΙ:

- Για κάθε  $x$  που διαβάζεται (από αριστερά προς τα δεξιά) θα γίνεται προσθήκη ενός  $x$  στην στοίβα.
- Για κάθε  $y$  που διαβάζεται θα αφαιρείται ένα  $x$  από την στοίβα.
- Το πρόγραμμα θα επιστρέφει την ένδειξη «YES» μονάχα εάν
  1. Έχει διαβαστεί όλη η είσοδος.
  2. Η Στοίβα είναι κενή ( περιέχει μόνο το \$ ).
  3. Είμαστε σε τελική κατάσταση.

Έτσι εάν δοθεί στο αυτόματο η συμβολοσειρά  $xxyxy$  το αυτόματο θα τα διαβάσει και θα καταλήξει σε τελική κατάσταση οπότε την αναγνωρίζει , όπως δείχνει ο Πίνακας 1.1 και η Εικόνα 2. Ενώ η έκφραση  $xyyx$  όχι. Εικόνα 3

## ΛΕΙΤΟΥΡΓΙΑ ΠΡΟΓΡΑΜΜΑΤΟΣ

- Η υλοποίηση του προγράμματος έχει γίνει σε γλώσσα C.
- Το πρόγραμμα ονομάζεται NAS.c.
- Για το compilation χρησιμοποιείται η εντολή “gcc NAS.c -o NAS” καθώς και για την εκτέλεση η εντολή “NAS.exe” (Εικόνα 1).
- Παραδοχή: Εάν η έκφραση που δίνεται σαν είσοδος είναι μεγαλύτερη από 15 χαρακτήρες τότε εμφανίζονται μόνο οι 3 πρώτοι , οι 3 τελευταίοι και ο αριθμός των χαρακτήρων της συμβολοσειράς. Ακόμα εάν τα σύμβολα μέσα στην στοίβα είναι παραπάνω από 10 εμφανίζονται τα 2 πρώτα , το τελευταίο και το μέγεθός της.

Ξεκινώντας το πρόγραμμα ζητάει από τον χρήστη να εισάγει τη συμβολοσειρά που θέλει να ελέγξει εάν αναγνωρίζεται ή όχι. Το πρόγραμμα διαβάζει τους χαρακτήρες (που δίνονται σαν είσοδος) μέχρι ο χρήστης να πατήσει το πλήκτρο enter. Η συμβολοσειρά πρέπει να αποτελείται μόνο από χαρακτήρες «x» και «y» και το μέγεθος της να μην ξεπερνάει τους 100 χαρακτήρες. Διαφορετικά ενημερώνει τον χρήστη ότι έβαλε λάθος είσοδο, σταματάει να διαβάζει , και αφού εμφανίσει «NO» το πρόγραμμα τερματίζεται. Ομοίως και στην περίπτωση που ο χρήστης δεν βάλει

είσοδο. Εάν η είσοδος είναι αποδεκτή (χαρακτήρες «x» και «y») αυξάνει έναν μετρητή και αποθηκεύει τους χαρακτήρες σε έναν πίνακα από χαρακτήρες.

Έπειτα ακολουθεί η υλοποίηση της στοίβας του Ντετερμινιστικού Αυτομάτου Στοίβας. Δημιουργείται μια δομή στοίβας η οποία αποτελείται από έναν πίνακα χαρακτήρων και την κορυφή της στοίβας. Η αναφορά στην δομή γίνεται με τη λέξη STACK. Η συνάρτηση ST\_init αρχικοποιεί την στοίβα, δηλαδή θέτει το top(την κορυφή της στοίβας) να είναι -1. Μια άλλη συνάρτηση που χρειάζεται είναι η ST\_empty η οποία ελέγχει εάν το top είναι -1, τότε η στοίβα είναι κενή και επιστρέφει "1" (θεωρώ ότι το "1" συμβολίζει το λογικό true και το "0" το λογικό false). Αλλιώς εάν είναι διάφορο του -1 επιστρέφει "0". Η ST\_full ομοίως με την ST\_empty ελέγχει εάν είναι γεμάτη η στοίβα, δηλαδή εάν η κορυφή της στοίβας ισούται με το πλήθος των χαρακτήρων μείον έναν χαρακτήρα και επιστρέφει false αντίστοιχα. Η ST\_push κάνει ώθηση ενός χαρακτήρα στην στοίβα. Το κάνει αυτό ελέγχοντας εάν είναι γεμάτη η στοίβα (καλεί την ST\_full) και εάν είναι, επειδή δεν μπορεί να εισάγει άλλο στοιχείο στην στοίβα επιστρέφει "0". Αλλιώς αυξάνει την κορυφή της στοίβας και βάζει σε αυτή την θέση του πίνακα το στοιχείο, και επιστρέφει true. Η ST\_pop είναι η εξαγωγή στοιχείου από την στοίβα. Ελέγχει εάν η στοίβα είναι κενή οπότε δεν μπορεί να βγάλει κάτι από αυτή άρα επιστρέφει false ειδάλλως μειώνει την κορυφή της στοίβας και επιστρέφει true.

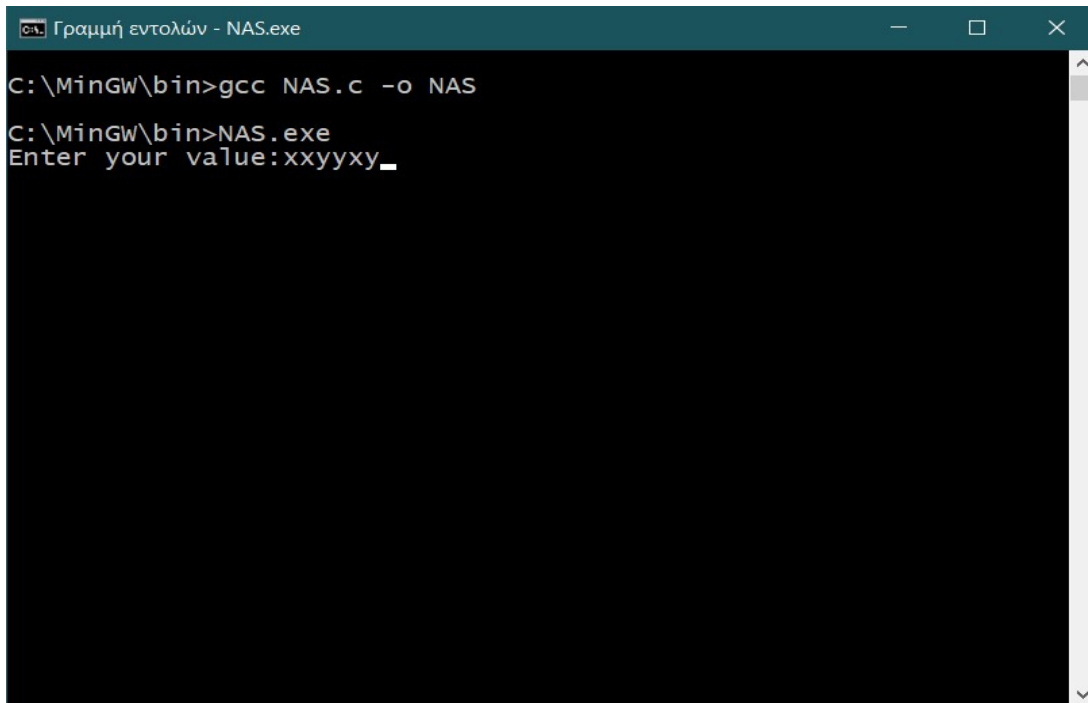
Υλοποίηση του ντετερμινιστικού αυτομάτου στοίβας

- Προσθέτω ως τελευταίο στοιχείο της συμβολοσειράς το κενό ( $\epsilon = \epsilon$ )
- Δημιουργώ μία στοίβα και την αρχικοποιώ.
- Εισάγω ως πρώτο στοιχείο της στοίβας το \$.
- Εμφανίζω την στοίβα, την κατάσταση στην οποία βρισκόμαστε ( αφού την αρχικοποιήσω ως  $K_1$ ), και την συμβολοσειρά που δόθηκε ως είσοδος.
- Ορίζω σαν αριθμό συνάρτησης που χρησιμοποιώ το "0", και μία λογική μεταβλητή με αρχική τιμή false
- Όσο υπάρχουν ακόμα χαρακτήρες στον πίνακα εισόδου:

Εάν η είσοδος είναι  $x$ , η κορυφή της στοίβας  $s$  και η παρούσα κατάσταση είναι η  $k_1$  τότε χρησιμοποιούμε την 1) συνάρτηση στοίβας  $p(k_1, s, x) = (k_1, sx)$ . Εάν η είσοδος είναι  $x$ , η κορυφή της στοίβας είναι  $x$  και η παρούσα κατάσταση είναι η  $k_1$  τότε χρησιμοποιούμε την 2) συνάρτηση στοίβας  $p(k_1, x, x) = (k_1, xx)$ . Σύμφωνα λοιπόν και με τις δύο παραπάνω γίνεται ώθηση του  $x$  στην στοίβα και εκτυπώνονται στην οθόνη η στοίβα, η κατάσταση στην οποία βρισκόμαστε, τα υπόλοιπα σύμβολα εισόδου καθώς και ο αριθμός της συνάρτησης στοίβας που χρησιμοποιείται κάθε φορά. Εάν η είσοδος είναι  $y$  και η κορυφή της στοίβας είναι  $x$  και η παρούσα κατάσταση είναι η  $k_1$  τότε χρησιμοποιούμε την 3) συνάρτηση στοίβας  $p(k_1, x, y) = (k_1, \epsilon)$  και τότε βγαίνει

ένα  $x$  από την στοίβα και εκτυπώνεται η αλληλουχία βημάτων. Στην τελευταία 4) συνάρτηση στοίβας  $p(k_1, \$, \epsilon) = (k_2, \epsilon)$  γίνεται πρόσβαση όταν η είσοδος είναι  $\epsilon$  (δηλαδή έχει διαβαστεί όλη η είσοδος), η κορυφή της στοίβας είναι το  $\$$  και η παρούσα κατάσταση γίνεται από  $k_1, k_2$ . Τότε μόνο αναγνωρίζεται η συμβολοσειρά και εκτυπώνεται «YES!!», αλλιώς εκτυπώνεται «NO». Τέλος εκτυπώνεται και «Illegal Entry» όταν ο χρήστης έχει δώσει συμβολοσειρά στην οποία τα  $y$  κοιτάζοντας την έκφραση από αριστερά προς τα δεξιά είναι περισσότερα.

## ΕΝΔΕΙΚΤΙΚΑ SCREENSHOTS



```
C:\MinGW\bin>gcc NAS.c -o NAS
C:\MinGW\bin>NAS.exe
Enter your value:xxyxyx_
```

Εικόνα 0-1 .Compilation και prompt για input

Στοιβά	Κατάσταση	Είσοδος	Αριθμός Συνάρτησης p
\$	K1	xxyxyx	
\$x	K1	xyxyx	1
\$xx	K1	yxyx	2
\$x	K1	xyx	3
\$xx	K1	yx	2
\$x	K1	y	3
\$	K1	ε	3
\$	K1	ε	4

Πίνακας 1. Επιτυχής αναγνώριση του xxyxyx

```
Γραμμή εντολών
C:\MinGW\bin>a
Enter your value:xyyyxy
Stack      Status      Input      Function number
$          k1          xxyyyxy
$x         k1          xyyyxy    1
$xx        k1          yyxy      2
$x         k1          yxy       3
$          k1          xy        3
$x         k1          y         1
$          k1
$          k2          4

YES!!
C:\MinGW\bin>
```

Εικόνα 0-2. Επιτυχής αναγνώριση του xxyyxy



```
C:\MinGW\bin>a
Enter your value:xyyx
Stack      Status      Input      Function number
$          k1          xyyx
$xx        k1          yyx          1
$          k1          yx           3

Illegal Entry
NO
C:\MinGW\bin>
```

Εικόνα 0-3. Λειτουργία του προγράμματος με τη μη αναγνωρίσιμη έκφραση xyyx.

```
C:\MinGW\bin>a
Enter your value:xxxyy
Stack      Status      Input      Function number
$          k1          xxxyy
$xx        k1          xyx          1
$xxx       k1          yx           2
$xx        k1          x           3
$xxx       k1          2
NO
C:\MinGW\bin>a
Enter your value:xxxyyy
Stack      Status      Input      Function number
$          k1          xxxyyy
$xx        k1          xxyyy          1
$xxx       k1          xyyy          2
$xxxx      k1          yyy           2
$xx        k1          yy           3
$xx        k1          y            3
$          k1          3
$          k2          4
YES!!
C:\MinGW\bin>
```

Εικόνα 0-4. Άλλα παραδείγματα σωστής λειτουργίας του προγράμματος .(1)

Στοιβά	Κατάσταση	Είσοδος	Αριθμός
\$	K <sub>1</sub>	xyx	
\$x	K <sub>1</sub>	yx	1
\$xx	K <sub>1</sub>	yx	2
\$x	K <sub>1</sub>	x	3
\$xx	K <sub>1</sub>	ε	2

*Πίνακας 2. Συμβολοσειρά: xyx*

Στοιβά	Κατάσταση	Είσοδος	Αριθμός
\$	K <sub>1</sub>	xxxyy	
\$x	K <sub>1</sub>	xyyy	1
\$xx	K <sub>1</sub>	xyyy	2
\$xxx	K <sub>1</sub>	yyy	2
\$xx	K <sub>1</sub>	yy	3
\$x	K <sub>1</sub>	y	3
\$	K <sub>1</sub>	ε	3
\$	K <sub>2</sub>	ε	4

*Πίνακας 3. Συμβολοσειρά: xxxyy.*

```

C:\MinGW\bin>gcc NAS.c -o NAS

C:\MinGW\bin>NAS.exe
Enter your value:xyyyxy
Stack      Status      Input      Function number
$          k1          xyyyxy
$x         k1          yyyxy      1
$          k1          yyxy      3

Illegal Entry
NO
C:\MinGW\bin>NAS.exe
Enter your value:xyxyxy
Stack      Status      Input      Function number
$          k1          xyxyxy
$x         k1          xyxyy      1
$xx        k1          yxyy      2
$x         k1          xy      3
$xx        k1          yy      2
$x         k1          y      3
$          k1          y      3
$          k2          y      4

YES!!
C:\MinGW\bin>_

```

Εικόνα 0-5. Παραδείγματα σωστής λειτουργίας του προγράμματος .(2)

Στοιβά	Κατάσταση	Είσοδος	Αριθμός
\$	K1	xyyyxy	
\$x	K1	yyyxy	1
\$	K1	yyxy	3

Πίνακας 4. Συμβολοσειρά: xyyxy



## Άσκηση (B)

Σε αυτήν την άσκηση θα δημιουργήσουμε μια γεννήτρια συμβολοσειρών η οποία θα παράγει έγκυρες συμβολοσειρές σύμφωνα με την γραμματική της εκφώνησης. Η υλοποίηση της έγινε σε γλώσσα Java.

Αρχικά ορίζουμε τα εξής: Ex για την έκφραση(expression),SE για την υποέκφραση (subexpression),E1 για στοιχείο 1(element1),E2 για το στοιχείο 2(element2).

Η μεταγλώττιση είναι 1)javac Main.java Syntax.java 2)java Main

Η κλάση Syntax

Αρχικά δημιουργούμε ένα αντικείμενο r τύπου Random και ένα field της κλάσης Syntax, το string s στο οποίο εκχωρούμε το περιεχόμενο "Ex". Θεωρούμε ότι από το μη τερματικό σύμβολο Ex θα πραγματοποιείται η αρχή της του προγράμματος. Επιπλέον δημιουργούμε ένα αντικείμενο pr τύπου 1) StringBuilder δηλαδή ένα mutable string το οποίο θα αποδειχθεί στην συνέχεια αρκετά χρήσιμο καθώς μεταβάλλεται ανάλογα με τις αντικαταστάσεις με τους κανόνες παραγωγής. Το pr δημιουργείται σύμφωνα με τα περιεχόμενα του String s. Στην συνέχεια ορίζουμε τις εξής μεθόδους:

Public void Expression():

Η μέθοδος αυτή πραγματοποιεί αντικατάσταση του substring Ex που περιέχεται στο pr σύμφωνα με τον κανόνα παραγωγής. Αρχικά μέσω της μεθόδου indexOf(String) του StringBuilder γίνεται αναζήτηση του string μέσα στο pr. Την πρώτη φορά που η συμβολοσειρά εντοπίσει το string Ex θα επιστρέψει το index από το οποίο αρχίζει η συμβολοσειρά και βρίσκεται στο pr. Με βάση το index γίνονται αντικατάσταση δύο χαρακτήρες(δηλαδή το Ex) και στην θέση τους παρεμβάλλεται το '(SE)'. Στην συνέχεια μετατρέπουμε τα περιεχόμενα του StringBuilder σε String και τα εκχωρούμε στο s. Στην συνέχεια εμφανίζουμε τα περιεχόμενα του τροποποιημένου string.

Public void SubExpression():

Η μέθοδος αυτή ουσιαστικά πραγματοποιεί αντικατάσταση του substring SE που περιέχεται στο pr σύμφωνα με τον κανόνα παραγωγής. Αρχικά μέσω της μεθόδου indexOf(String) του StringBuilder γίνεται αναζήτηση του string μέσα στο pr. Την πρώτη φορά που η συμβολοσειρά εντοπίζει το string SE(το μικρότερο Index) θα επιστρέψει το index από το οποίο αρχίζει η συμβολοσειρά και βρίσκεται στο pr. Με βάση το index γίνονται αντικατάσταση δύο χαρακτήρες(δηλαδή το SE) και στην θέση τους τοποθετείται το “E1E2”. Στην συνέχεια μετατρέπουμε τα περιεχόμενα του StringBuilder σε String και τα εκχωρούμε στο s. Στην συνέχεια εμφανίζουμε τα περιεχόμενα του αλλαγμένου string.

Public void Element1():

Με βάση την βιβλιοθήκη random παράγουμε έναν τυχαίο αριθμό από το 1-2. Παρόμοια όπως προαναφέραμε και από πάνω γίνεται αντικατάσταση του E1 σύμφωνα με το index μέσα στο string με βάση την επιλογή του choice δηλαδή του αριθμού που επιλέχθηκε μέσα από την Random. Αν το choice είναι 1 τότε το E1 αντικαθίστανται από το “v” αλλιώς από το “Ex”. Έπειτα μετατρέπουμε τα περιεχόμενα του StringBuilder σε String και τα εκχωρούμε στο s. Στην συνέχεια εμφανίζουμε τα περιεχόμενα του string που μόλις άλλαξε.

Public void Element2():

Με βάση την βιβλιοθήκη random παράγουμε έναν τυχαίο αριθμό από το 1-3. Παρόμοια όπως προαναφέραμε ήδη γίνεται αντικατάσταση του E2 σύμφωνα με το index μέσα στο string με βάση την επιλογή του choice δηλαδή του αριθμού που επιλέχθηκε μέσα από την Random. Αν το choice είναι 1 τότε το E2 αντικαθίστανται από το “-SE”, αν το choice είναι 2 από το “+SE” αλλιώς από το “” δηλαδή το κενό. Τέλος αλλάζουμε τα δεδομένα του stringBuilder σε string και τα εκχωρούμε στο s και εμφανίζονται τα περιεχόμενα του.

Στην κλάση Main:

Μέσα στην μέθοδο `main`, δημιουργούμε ένα `object m` τύπου `Syntax()`. Επίσης ορίζουμε τις ακέραιες `local` μεταβλητές `count` που μετράει τους τερματικούς χαρακτήρες και `steps` που εκφράζει τον αριθμό των αντικαταστάσεων που γίνονται στο `string` σύμφωνα με τους κανόνες παραγωγής της γραμματικής. Όσο το `z` είναι θετικό δηλαδή το `string` που θα παράγει η γεννήτρια δεν αποτελείται από μόνο τερματικούς χαρακτήρες τότε μέσα σε ένα καινούργιο `loop for` όπου αυτό θα διατρέχει όλο το μήκος του `string` εφόσον τα `steps` δεν είναι μεγαλύτερο του 70 θα ελέγχουμε διαδοχικά τις εξής καταστάσεις: Αν ο χαρακτήρας στην `i` θέση είναι τερματικός τότε αυξάνουμε το `count`, αλλιώς αν είναι 'Ε' και ο χαρακτήρας που ακολουθεί στην `i+1` θέση είναι 'χ', τότε θα πρέπει να κάνουμε αντικατάσταση με βάση τον κανόνα του `Expression` και `break` από το `for loop`, αλλιώς αν είναι 'S' και ακολουθείται από 'E' τότε αντίστοιχα θα πρέπει να γίνει αντικατάσταση `Subexpression` στο `string` και `break`, ή αν είναι 'E' και έπεται αμέσως το 'i' εκτελείται ο κανόνας του `Element1` και `break` από το `for loop`, ή αλλιώς γίνεται αντικατάσταση με βάση το `Element2` ή τέλος εμφανίζεται αναγνωριστικό λάθους. Μετά το τέλος του `for loop` αν ο μετρητής `count` ισούται με το αριθμό του μήκους της συμβολοσειράς τότε σημαίνει ότι το `string` αποτελείται μόνο από τερματικά σύμβολα και επομένως το `string` που θέλουμε να φτιάξουμε είναι έτοιμο επομένως το `z` θα γίνει `false` για να ολοκληρωθεί το `while-loop`. Θεωρούμε σαν σύμβαση ότι ο αριθμός των αντικαταστάσεων θα είναι μέχρι το 70 με σκοπό να αντιμετωπιστεί το πρόβλημα της αναδρομικότητας. Αν ο αριθμός των αντικαταστάσεων υπερβεί το 70 το πρόγραμμα ολοκληρώνεται.

Ενδεικτικά screenshots σε 3 διαφορετικές εκτελέσεις του προγράμματος:

α) Για 10 συνεχείς αντικαταστάσεις

β) Για 15 συνεχείς αντικαταστάσεις

γ) Για 71 όπου ολοκληρώνεται λόγω του ορίου 70 που θέσαμε εμείς

α)

The screenshot shows the IntelliJ IDEA interface with a project named 'compilers8'. The 'Run' tab is active, displaying the execution steps of a Java program. The program starts with 'Ex' and proceeds through a series of steps: (SE), (E1E2), (vE2), (v+SE), (v+E1E2), (v+vE2), (v+v-SE), (v+v-E1E2), (v+v-vE2), and (v+v-v). The process finishes with exit code 0. The IDE version is 2019.1, and the Java version is 11.0.2.

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2019.1\lib\idea_rt.jar=588"
Ex
(SE)
(E1E2)
(vE2)
(v+SE)
(v+E1E2)
(v+vE2)
(v+v-SE)
(v+v-E1E2)
(v+v-vE2)
(v+v-v)
Steps 10
Process finished with exit code 0
```

β)

The screenshot shows the IntelliJ IDEA interface with a project named 'compilers8'. The 'Run' tab is active, displaying the execution steps of a Java program. The program starts with 'Ex' and proceeds through a series of steps: (SE), (E1E2), (vE2), (v-SE), (v-E1E2), (v-ExE2), (v-(SE)E2), (v-(E1E2)E2), (v-(ExE2)E2), (v-((SE)E2)E2), (v-((E1E2)E2)E2), (v-((vE2)E2)E2), (v-((v)E2)E2), (v-((v)E2)), and (v-((v))). The process finishes with exit code 0. The IDE version is 2019.1, and the Java version is 11.0.2.

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2019.1\lib\idea_rt.jar=588"
Ex
(SE)
(E1E2)
(vE2)
(v-SE)
(v-E1E2)
(v-ExE2)
(v-(SE)E2)
(v-(E1E2)E2)
(v-(ExE2)E2)
(v-((SE)E2)E2)
(v-((E1E2)E2)E2)
(v-((vE2)E2)E2)
(v-((v)E2)E2)
(v-((v)E2))
(v-((v)))
Steps 15
Process finished with exit code 0
```



The screenshot shows the IntelliJ IDEA IDE with a Java project named 'compilersB'. The main file is 'Syntax.java'. The code in the editor is as follows:

```

"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2019.1\lib\idea_rt.jar=58
Ex
(SE)
(E1E2)
(ExE2)
((SE)E2)
((E1E2)E2)
((vE2)E2)
((v+SE)E2)
((v+E1E2)E2)
((v+ExE2)E2)
((v+(SE)E2)E2)
((v+(E1E2)E2)E2)
((v+(ExE2)E2)E2)
((v+((SE)E2)E2)E2)
((v+(((SE)E2)E2)E2)E2)
((v+(((E1E2)E2)E2)E2)E2)
((v+(((ExE2)E2)E2)E2)E2)
((v+(((SE)E2)E2)E2)E2)E2)
((v+(((E1E2)E2)E2)E2)E2)E2)
((v+(((vE2)E2)E2)E2)E2)E2)
((v+(((v)E2)E2)E2)E2)E2)
((v+(((v)-(SE)E2)E2)E2)E2)E2)
((v+(((v)-(E1E2)E2)E2)E2)E2)E2)

```

The IDE interface includes the Project view on the left, the Run configuration window at the top, and a terminal window at the bottom. The terminal window shows the command to run the Java program.



Οι 3 τελευταίες εικόνες παρουσιάζουν την περίπτωση ο αριθμός των αντικαταστάσεων να ξεπεράσει το 70.

### Άσκηση (Γ)

Για να αποδειχθεί εάν είναι η γραμματική  $LL(1)$ , αρχικά θα υπολογισθούν τα σύνολα  $FIRST$ ,  $FOLLOW$ ,  $EMPTY$  και  $LOOKAHEAD$  και έπειτα θα γίνει παράθεση του συντακτικού πίνακα.

Αναλύοντας τους κανόνες παραγωγής της γραμματικής της εκφώνησης έχουμε τους παρακάτω κανόνες:

- 1)  $S \rightarrow (X)$
- 2)  $X \rightarrow YZ$
- 3)  $Y \rightarrow \alpha$
- 4)  $Y \rightarrow \beta$
- 5)  $Y \rightarrow S$
- 6)  $Z \rightarrow *X$
- 7)  $Z \rightarrow -X$
- 8)  $Z \rightarrow +X$
- 9)  $Z \rightarrow \epsilon$

Σύνολα ***FIRST***:

$$FIRST(S) = \{ ( \}$$

$$I. \quad FIRST(Y) - \{ \epsilon \} \in FIRST(X)$$

$$II. \quad FIRST(Y) = \{ \alpha, \beta, ( \}$$

$$\text{Από I και II} \Rightarrow FIRST(X) = \{ \alpha, \beta, ( \}$$

$$FIRST(Z) = \{ *, -, +, \epsilon \}$$

Σύνολα ***FOLLOW***:

Σύμφωνα με τους κανόνες υπολογισμού του FOLLOW (που βρίσκονται στο βιβλίο ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ, Μ . Κ . ΒΙΡΒΟΥ, ΕΚΔΟΣΕΙΣ ΒΑΡΒΑΡΗΓΟΥ, ΚΕΦ. 5.2, ΣΕΛ. 177).

Από τον κανόνα 1) υπολογισμού του FOLLOW

I.  $\$ \in \text{FOLLOW}(S)$

Από τον κανόνα 2) υπολογισμού του FOLLOW

II.  $1) \Rightarrow \text{FIRST}( ) - \{\epsilon\} \subseteq \text{FOLLOW}(X) \Rightarrow \{ ) \} \subseteq \text{FOLLOW}(X)$   
III.  $2) \Rightarrow \text{FIRST}(Z) - \{\epsilon\} \subseteq \text{FOLLOW}(Y) \Rightarrow \{*, -, +\} \subseteq \text{FOLLOW}(Y)$

Από τον κανόνα 3i) υπολογισμού του FOLLOW

IV.  $2) \Rightarrow \text{FOLLOW}(X) \subseteq \text{FOLLOW}(Z)$   
V.  $5) \Rightarrow \text{FOLLOW}(Y) \subseteq \text{FOLLOW}(S)$   
VI.  $6) + 7) + 8) \Rightarrow \text{FOLLOW}(Z) \subseteq \text{FOLLOW}(X)$

Από τον κανόνα 3ii) υπολογισμού του FOLLOW

VII.  $2) \Rightarrow \text{FOLLOW}(X) \subseteq \text{FOLLOW}(Y)$

Από τις παραπάνω σχέσεις προκύπτουν τα εξής:

$\text{II} \Rightarrow \text{FOLLOW}(X) = \{ ) \}$

$\text{II} + \text{IV} + \text{VI} \Rightarrow \text{FOLLOW}(Z) = \{ ) \}$

$\text{III} + \text{IV} \Rightarrow \text{FOLLOW}(Y) = \{*, -, +, )\}$

$\text{I} + \text{V} \Rightarrow \text{FOLLOW}(S) = \{*, -, +, ), \$\}$

Σύνολα **EMPTY**:

$\text{EMPTY}(S) = \text{FALSE}$

$\text{EMPTY}(X) = \text{FALSE}$

$\text{EMPTY}(Y) = \text{FALSE}$

$\text{EMPTY}(Z) = \text{TRUE}$

Σύνολα **LOOKAHEAD**:

$$\text{LOOKAHEAD}(S \rightarrow (X)) = \text{FIRST}( ) = \{ ) \}$$

$$\text{LOOKAHEAD}(X \rightarrow YZ) = \text{FIRST}(Y) = \{ \alpha, \beta, ) \}$$

$$\text{LOOKAHEAD}(Y \rightarrow \alpha) = \text{FIRST}(\alpha) = \{ \alpha \}$$

$$\text{LOOKAHEAD}(Y \rightarrow \beta) = \text{FIRST}(\beta) = \{ \beta \}$$

$$\text{LOOKAHEAD}(Y \rightarrow S) = \text{FIRST}(S) = \{ ( \}$$

$$\text{LOOKAHEAD}(Z \rightarrow *X) = \text{FIRST}(*) = \{ * \}$$

$$\text{LOOKAHEAD}(Z \rightarrow -X) = \text{FIRST}(-) = \{ - \}$$

$$\text{LOOKAHEAD}(Z \rightarrow +X) = \text{FIRST}(+) = \{ + \}$$

$$\text{LOOKAHEAD}(Z \rightarrow \epsilon) = \text{FOLLOW}(Z) = \{ ) \}$$

Επομένως η γραμματική είναι LL(1) διότι:

$$\text{LOOKAHEAD}(Y \rightarrow \alpha) \cap \text{LOOKAHEAD}(Y \rightarrow \beta) \cap \text{LOOKAHEAD}(Y \rightarrow S) = \emptyset$$

$$\text{LOOKAHEAD}(Z \rightarrow *X) \cap \text{LOOKAHEAD}(Z \rightarrow -X) \cap \text{LOOKAHEAD}(Z \rightarrow +X) \cap \text{LOOKAHEAD}(Z \rightarrow \epsilon) = \emptyset$$

V/T	(	)	$\alpha$	$\beta$	*	-	+	\$
S	S- >(X)							
X	X- >YZ		X- >YZ	X- >YZ				
Y	Y->S		Y-> $\alpha$	Y-> $\beta$				
Z		Z-> $\epsilon$			Z->*X	Z->-X	Z- >+X	

Στοιβά	Είσοδος	Στοιχεία πίνακα	Παραγωγή
\$S	$((\alpha-\beta)^*(\beta+\alpha))\$$	M(S, ( )	S->(X)
\$)X(	$((\alpha-\beta)^*(\beta+\alpha))\$$		
\$)X	$(\alpha-\beta)^*(\beta+\alpha))\$$	M(X, ( )	X->YZ
\$)ZY	$(\alpha-\beta)^*(\beta+\alpha))\$$	M(Y, ( )	Y->S
\$)ZS	$(\alpha-\beta)^*(\beta+\alpha))\$$	M(S, ( )	S->(X)
\$)Z)X(	$(\alpha-\beta)^*(\beta+\alpha))\$$		
\$)Z)X	$\alpha-\beta)^*(\beta+\alpha))\$$	M(X, $\alpha$ )	X->YZ
\$)Z)YZ	$\alpha-\beta)^*(\beta+\alpha))\$$	M(Y, $\alpha$ )	Y-> $\alpha$
\$)Z)Z $\alpha$	$\alpha-\beta)^*(\beta+\alpha))\$$		
\$)Z)Z	$-\beta)^*(\beta+\alpha))\$$	M(Z, -)	Z->-X
\$)Z)X-	$-\beta)^*(\beta+\alpha))\$$		
\$)Z)X	$\beta)^*(\beta+\alpha))\$$	M(X, $\beta$ )	X->YZ
\$)Z)ZY	$\beta)^*(\beta+\alpha))\$$	M(Y, $\beta$ )	Y-> $\beta$
\$)Z)Z $\beta$	$\beta)^*(\beta+\alpha))\$$		
\$)Z)Z	$)^*(\beta+\alpha))\$$	M(Z, ))	Z-> $\epsilon$
\$)Z)	$)^*(\beta+\alpha))\$$		

\$)Z	$*(\beta+\alpha))\$$	$M(Z, *)$	$Z \rightarrow^* X$
\$)X*	$*(\beta+\alpha))\$$		
\$)X	$(\beta+\alpha))\$$	$M(X, ($	$X \rightarrow YZ$
\$)ZY	$(\beta+\alpha))\$$	$M(Y, ($	$Y \rightarrow S$
\$)ZS	$(\beta+\alpha))\$$	$M(S, ($	$S \rightarrow (X)$
\$)Z)X(	$(\beta+\alpha))\$$		
\$)Z)X	$\beta+\alpha))\$$	$M(X, \beta)$	$X \rightarrow YZ$
\$)Z)ZY	$\beta+\alpha))\$$	$M(Y, \beta)$	$Y \rightarrow \beta$
\$)Z)Z\beta	$\beta+\alpha))\$$		
\$)Z)Z	$+\alpha))\$$	$M(Z, +)$	$Z \rightarrow +X$
\$)Z)X+	$+\alpha))\$$		
\$)Z)X	$\alpha))\$$	$M(X, \alpha)$	$X \rightarrow YZ$
\$)Z)ZY	$\alpha))\$$	$M(Y, \alpha)$	$Y \rightarrow \alpha$
\$)Z)Z \alpha	$\alpha))\$$		
\$)Z)Z	$)\$$	$M(Z, ))$	$Z \rightarrow \varepsilon$
\$)Z)	$)\$$		
\$)Z	$)\$$	$M(Z, ))$	$Z \rightarrow \varepsilon$
\$)	$)\$$		
\$	\$		

## ΛΕΙΤΟΥΡΓΙΑ ΠΡΟΓΡΑΜΜΑΤΟΣ

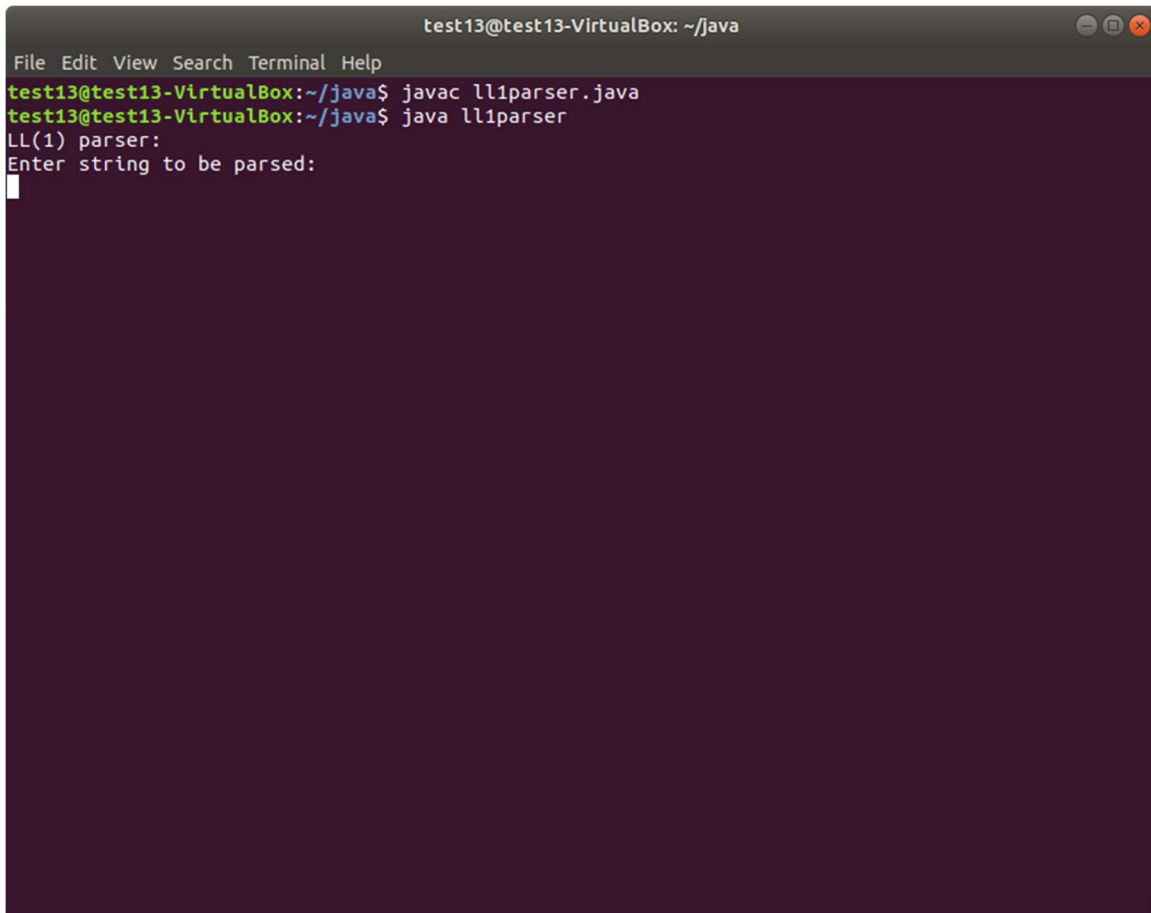
- Η υλοποίηση του προγράμματος έχει γίνει σε γλώσσα JAVA.
- Έχει αναπτυχθεί με βάση τους κανόνες παραγωγής της συγκεκριμένης εκφώνησης.
- Το πρόγραμμα ονομάζεται `ll1parser.java`.
- Για το compilation χρησιμοποιείται η εντολή `"javac ll1parser.java"` καθώς και για την εκτέλεση η εντολή `"java ll1parser"`. **Σημείωση:** Η ανάπτυξη του προγράμματος προϋποθέτει την ικανότητα αναγνώρισης και εμφάνισης των ελληνικών χαρακτήρων 'α' και 'β' στο *command line*.

Το πρόγραμμα κατά την εκτέλεσή του παραπέμπει τον χρήστη να εισάγει ένα σύνολο συμβόλων το οποίο επιθυμεί να ελέγξει εάν αναγνωρίζεται από την συγκεκριμένη γλώσσα. Η είσοδος αποθηκεύεται σε μία μεταβλητή `input` τύπου `String`. Έπειτα γίνεται η αρχικοποίηση των πινάκων. Ο δισδιάστατος πίνακας `ptable` είναι ο συντακτικός πίνακας του προγράμματος με κάθε γραμμή να αντιστοιχεί σε ένα μη-τερματικό σύμβολο και με κάθε στήλη σε ένα τερματικό. Όπου δεν αντιστοιχίζεται κανόνας μεταξύ μη-τερματικού και τερματικού συμβόλου υπάρχει `null`. Στη συνέχεια γίνεται η εκτύπωση των πινάκων με τα σύμβολα της γραμματικής. Ο αλγόριθμος που χρησιμοποιείται βρίσκεται στη σελίδα 202 του κεφαλαίου 5.2.3 του βιβλίου ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ, Μ . Κ . ΒΙΡΒΟΥ, ΕΚΔΟΣΕΙΣ ΒΑΡΒΑΡΗΓΟΥ. Υπάρχει υλοποίηση της στοίβας που χρησιμοποιεί τις λειτουργίες: `stack.peek()` για την ανάγνωση της κορυφής της στοίβας, `stack.pop()` για την αφαίρεση του στοιχείου που βρίσκεται στην κορυφή της και της `stack.push()` για την εισαγωγή στοιχείου. Η αρχικοποίηση της στοίβας γίνεται με την είσοδο του "\$" και του αρχικού συμβόλου `S`. Σε κάθε επανάληψη εμφανίζονται στην οθόνη ο αριθμός της επανάληψης, ποιο σύμβολο αφαιρείται (εάν αυτό απαιτείται από το βήμα του αλγορίθμου), ο κανόνας που χρησιμοποιείται, το σύνολο των συμβόλων που εισέρχεται στη στοίβα μετά την αντικατάσταση με βάση τους κανόνες, η κορυφή της στοίβας καθώς και το σύμβολο το οποίο βρίσκεται στο `input` προς πιθανή



αντικατάσταση. Σε περίπτωση σφάλματος εμφανίζεται το αντίστοιχο μήνυμα λάθους. Εάν το input αναγνωριστεί τότε στην οθόνη εμφανίζονται επιτυχώς τα βήματα που έγιναν και οι pointers της στοίβας και του input ταυτίζονται περιέχοντας και οι δύο το “\$”. Στην περίπτωση που το input δεν αναγνωριστεί ο χρήστης θα λάβει στην οθόνη του το μήνυμα “ Input not recognised by grammar!!!”.

## ΕΝΔΕΙΚΤΙΚΑ SCREENSHOTS



```
test13@test13-VirtualBox: ~/java
File Edit View Search Terminal Help
test13@test13-VirtualBox:~/java$ javac ll1parser.java
test13@test13-VirtualBox:~/java$ java ll1parser
LL(1) parser:
Enter string to be parsed:

```

Εικόνα 1. Compilation και prompt για input.

```
test13@test13-VirtualBox: ~/java
File Edit View Search Terminal Help
LL(1) parser:
Enter string to be parsed:
((α-β)*(β+α))
String to be parsed is:((α-β)*(β+α))$
Displaying terminal characters: ( ) α β * - + $
Displaying non-terminal characters: S X Y Z
Displaying parsing table:
-----
| (X) | null | null | null | null | null | null |
-----
| YZ | null | YZ | YZ | null | null | null |
-----
| S | null | α | β | null | null | null |
-----
| null | | null | null | *X | -X | +X |
-----
Stack pointer: S
Input pointer: (
-----
Repetition number: 1
Rule is: S->(X)
Revprod is: )
Revprod is: )X
Revprod is: )X(
```

Εικόνα 2. Εμφάνιση των πινάκων.

```
test13@test13-VirtualBox: ~/java
File Edit View Search Terminal Help
-----
Repetition number: 11
- is popped out of the stack
-----
Repetition number: 12
Rule is: X->YZ
Revprod is: Z
Revprod is: ZY
top on stack: Y
Input pointer: β
-----
Repetition number: 13
Rule is: Y->β
Revprod is: β
top on stack: β
Input pointer: β
-----
Repetition number: 14
β is popped out of the stack
-----
Repetition number: 15
Rule is: Z->
top on stack: )
Input pointer: )
```

Εικόνα 3. Εμφάνιση ενδεικτικών βημάτων.

```
File Edit View Search Terminal Help
-----
Repetition number: 30
a is popped out of the stack
-----
Repetition number: 31
Rule is: Z->
top on stack: )
Input pointer: )
-----
Repetition number: 32
) is popped out of the stack
-----
Repetition number: 33
Rule is: Z->
top on stack: )
Input pointer: )
-----
Repetition number: 34
) is popped out of the stack
-----
top on stack: $
Input pointer: $
LL1 parser shutting down ...
test13@test13-VirtualBox:~/java$
```

Εικόνα 4. Παράδειγμα τερματισμού επιτυχημένης αναγνώρισης.

```

File Edit View Search Terminal Help
LL(1) parser:
Enter string to be parsed:
(a+β()-a(())$
String to be parsed is:(a+β()-a(())$
Displaying terminal characters: ( ) a β * - + $
Displaying non-terminal characters: S X Y Z
Displaying parsing table:
-----
| (X) | null | null | null | null | null | null |
-----
| YZ | null | YZ | YZ | null | null | null |
-----
| S | null | a | β | null | null | null |
-----
| null | | null | null | *X | -X | +X |
-----
Stack pointer: S
Input pointer: (
-----
Repetition number: 1
Rule is: S->(X)
Revprod is: )
Revprod is: )X
Revprod is: )X(

```

Εικόνα 5. Παράδειγμα λειτουργίας με μη αναγνωρίσιμη είσοδο.(1)

```

File Edit View Search Terminal Help
+ is popped out of the stack
-----
Repetition number: 8
Rule is: X->YZ
Revprod is: Z
Revprod is: ZY
top on stack: Y
Input pointer: β
-----
Repetition number: 9
Rule is: Y->β
Revprod is: β
top on stack: β
Input pointer: β
-----
Repetition number: 10
β is popped out of the stack
-----
Repetition number: 11
Input not recognised by grammar!!!
top on stack: Z
Input pointer: (
LL1 parser shutting down ...
test13@test13-VirtualBox:~/java$

```

Εικόνα 6. Παράδειγμα λειτουργίας με μη αναγνωρίσιμη είσοδο.(2)

```

File Edit View Search Terminal Help
Enter string to be parsed:
@)β+)α((-β)
String to be parsed is:@)β+)α((-β)$
Displaying terminal characters: ( ) α β * - + $
Displaying non-terminal characters: S X Y Z
Displaying parsing table:
-----
| (X) | null | null | null | null | null | null |
-----
| YZ | null | YZ | YZ | null | null | null |
-----
| S | null | α | β | null | null | null |
-----
| null | | null | null | *X | -X | +X |
-----
Stack pointer: S
Input pointer: @
-----
Repetition number: 1
No terminal character
top on stack: S
Input pointer: @
LL1 parser shutting down ...
test13@test13-VirtualBox:~/java$

```

Εικόνα 7. Παράδειγμα διακοπής λειτουργίας λόγω μη αποδεκτού χαρακτήρα σύμφωνα με το αλφάβητο της γλώσσας.

## Άσκηση (Δ)

Για το συντακτικό διάγραμμα.

Θεωρούμε σαν σύμβαση ότι μια μεταβλητή μπορεί να αποτελείται και από γράμματα(κεφαλαία η μικρά) και από αριθμούς σε οποιοδήποτε αριθμό αλλά υποχρεωτικά θα ξεκινάει από γράμμα.

Στο συντακτικό διάγραμμα 2) δείχνουμε τα τερματικά σύμβολα με στρογγυλά και τα μη τερματικά με κουτιά. Αρχικά γίνεται αναπαράσταση σε συντακτικό διάγραμμα της γενικής έκφρασης και στην συνέχεια γίνεται επιμέρους αναπαράσταση σε συντακτικό διάγραμμα του αριθμού ,του τελεστή, της υποέκφρασης και της μεταβλητής. Τα επιμέρους συντακτικά διαγράμματα συνθέτουν το τελικό συντακτικό διάγραμμα που είναι το ζητούμενο. Σε περιγραφή BNF προκύπτει το θέμα της αναδρομικότητας το οποίο αντιμετωπίζεται επιτυχώς με εισαγωγή βοηθητικών κανόνων αναγκαίων για την σωστή παραγωγή έγκυρων

κανονικών εκφράσεων. Τα μη τερματικά σύμβολα αναπαρίστανται με <> και το βελάκι με το ::=

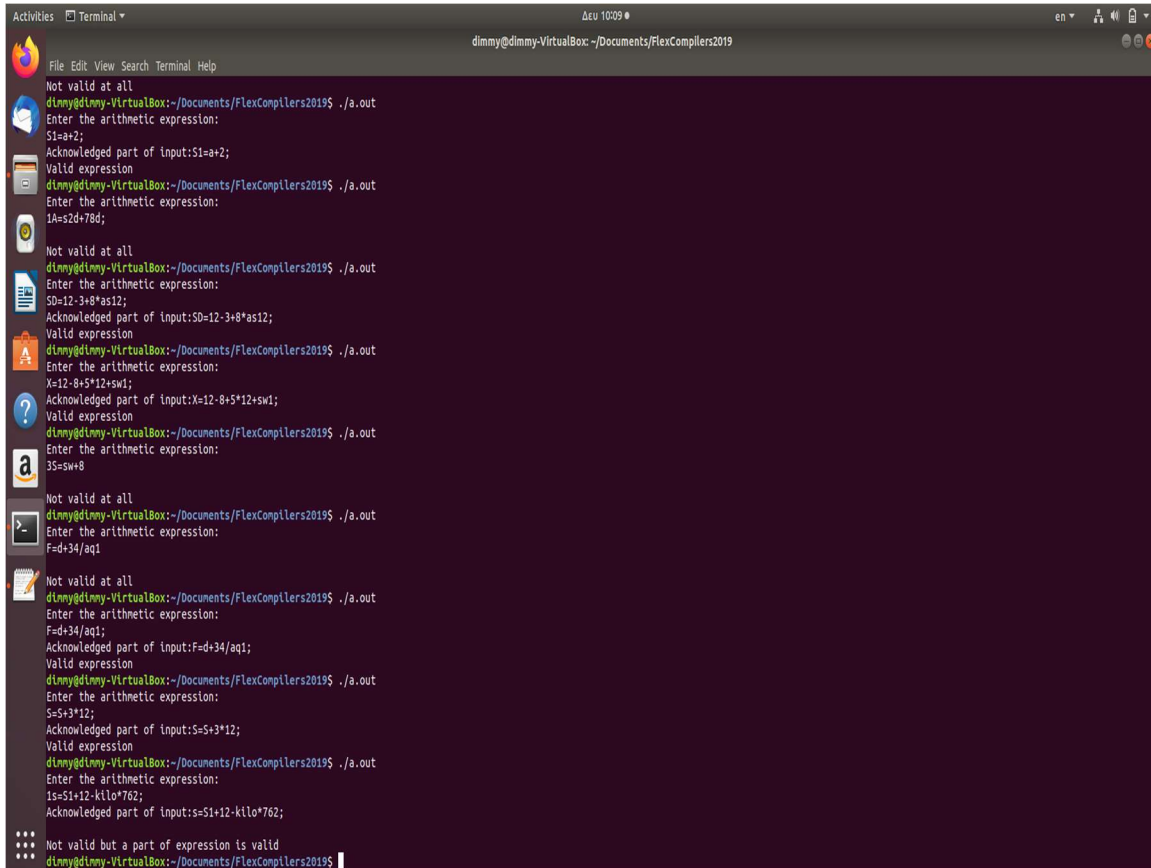
Σε EBNF ο προσδιορισμός σύνταξης γίνεται απλούστερος και εισάγεται και επαναληπτικότητα στους συντακτικούς κανόνες. Τα τερματικά σύμβολα μπαίνουν σε "", το βελάκι γίνεται με =, ενώ {} συμβάλλουν στην επανάληψη των περιεχομένων τους.

Για το πρόγραμμα Flex(reg.l)

Στο βοηθητικό τμήμα δηλώσεων ορίζουμε δύο ακέραιες μεταβλητές `invalid` και `part` τις οποίες εκχωρούμε με 0. Η μεταβλητή `invalid` εκφράζει τον αριθμό των χαρακτήρων που δεν αναγνωρίζονται από την κανονική έκφραση και η μεταβλητή `part` τον αριθμό των εκφράσεων που αναγνωρίζονται. Στο τμήμα δηλώσεων ορίζουμε το `NUMBER` το οποίο αποτελείται τουλάχιστον από ένα ψηφίο, το `VARIABLE` του οποίου η κανονική έκφραση αποτελείται από ένα γράμμα μικρό ή μεγάλο και μετά σε οποιοδήποτε αριθμό γράμματα ή αριθμοί, καθώς και τον `OPERATOR` που περιέχει τους τελεστές "+", "-", "\*", "/", "%" τους οποίους περικλείουμε σε εισαγωγικά καθώς είναι μεταχαρακτήρες. Με το %% τώρα ξεκινάει το μέρος του προγράμματος που σχετίζεται με τους κανόνες. Αν ο scanner διαβάσει έκφραση που ταιριάζει με το pattern `{(VARIABLE)}"="({NUMBER}|{VARIABLE})({OPERATOR}|{VARIABLE}|{NUMBER}))+";` τότε θα πραγματοποιηθεί δράση που είναι να εκτυπώνει στην κονσόλα το κομμάτι που αναγνωρίστηκε και να αυξάνει τον μετρητή `part`. Αν αλλάξουμε σε νέα γραμμή στην είσοδο ελέγχουμε την τιμή του `invalid`. Αν είναι 0 και το `part > 0` τότε σημαίνει ότι η είσοδος που δόθηκε ως κανονική έκφραση είναι έγκυρη αλλιώς αν είναι το `part > 0` σημαίνει ότι η είσοδος δεν είναι έγκυρη αλλά ένα κομμάτι της είναι έγκυρο αλλιώς η είσοδος που δόθηκε από τον χρήστη δεν είναι καθόλου έγκυρη. Τέλος αξίζει να αναφερθεί πως αν κάποιο κομμάτι της εισόδου δεν ταιριάζει με την κανονική έκφραση τότε οτιδήποτε άλλο και με την νέα γραμμή θα προκαλεί την αύξηση του μετρητή `invalid`. 3) Στην συνέχεια μέσα στην `main` με την βοήθεια της μεταβλητής τύπου `*File` ορίζουμε από που θα δεχθούμε είσοδο (από άλλο αρχείο εισόδου ή

κονσόλα). Τέλος καλούμε την `yylex()` η οποία είναι τύπου `int` (επιστρέφει το λεκτικό που αναγνωρίζεται) το η οποία παράγει κώδικα σύμφωνα με τους κανόνες που γράψαμε πιο πάνω.

Ενδεικτικά screenshots με διαφορετικές εισόδους από κονσόλα στο πρόγραμμα `reg.l`



```
dimmy@dimmy-VirtualBox: ~/Documents/FlexCompilers2019
File Edit View Search Terminal Help
Not valid at all
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
S1=a+2;
Acknowledged part of input:S1=a+2;
Valid expression
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
1A=s2d+78d;
Not valid at all
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
SD=12-3+8*as12;
Acknowledged part of input:SD=12-3+8*as12;
Valid expression
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
X=12-8+5*12+sw1;
Acknowledged part of input:X=12-8+5*12+sw1;
Valid expression
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
3S=sw+8
Not valid at all
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
F=d+34/aq1
Not valid at all
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
F=d+34/aq1;
Acknowledged part of input:F=d+34/aq1;
Valid expression
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
S=s+3*12;
Acknowledged part of input:S=s+3*12;
Valid expression
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$ ./a.out
Enter the arithmetic expression:
1s=S1+12-kilo*762;
Acknowledged part of input:s=S1+12-kilo*762;
*** Not valid but a part of expression is valid
dimmy@dimmy-VirtualBox:~/Documents/FlexCompilers2019$
```



B)

Σε BNF

<Έκφραση>::=<Μεταβλητή>"="<Υποέκφραση1><Υποέκφραση2>;

<Ψηφίο>::=0|1|2|3|4|5|6|7|8|9

<Αριθμός>::=<Ψηφίο>|<Ψηφίο><Αριθμός>

<Τελεστής>::=+|-|\*|/|%

<Υποέκφραση1>::=<Μεταβλητή>|<Αριθμός>

<Υποέκφραση2>::=<Τελεστής><Υποέκφραση1>|<Τελεστής><Υποέκφραση1><Υποέκφραση2>

<Γράμμα>::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

<Στοιχεία\_Μεταβλητής>::=<Γράμμα>|<Ψηφίο>|<Γράμμα><Στοιχεία\_Μεταβλητής>|<Ψηφίο><Στοιχεία\_Μεταβλητής>

<Μεταβλητή>::=<Γράμμα>|<Γράμμα><Στοιχεία\_Μεταβλητής>

Γ)

Σε EBNF

Έκφραση=Μεταβλητή"="Υποέκφραση1Υποέκφραση2";

Μεταβλητή=Γράμμα{Γράμμα| Ψηφίο}.

Υποέκφραση1=Μεταβλητή| Αριθμός.

Υποέκφραση2=Τελεστής Υποέκφραση1{Τελεστής Υποέκφραση1}.

Γράμμα=

"A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|"N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"|"a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|"m"|"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z".

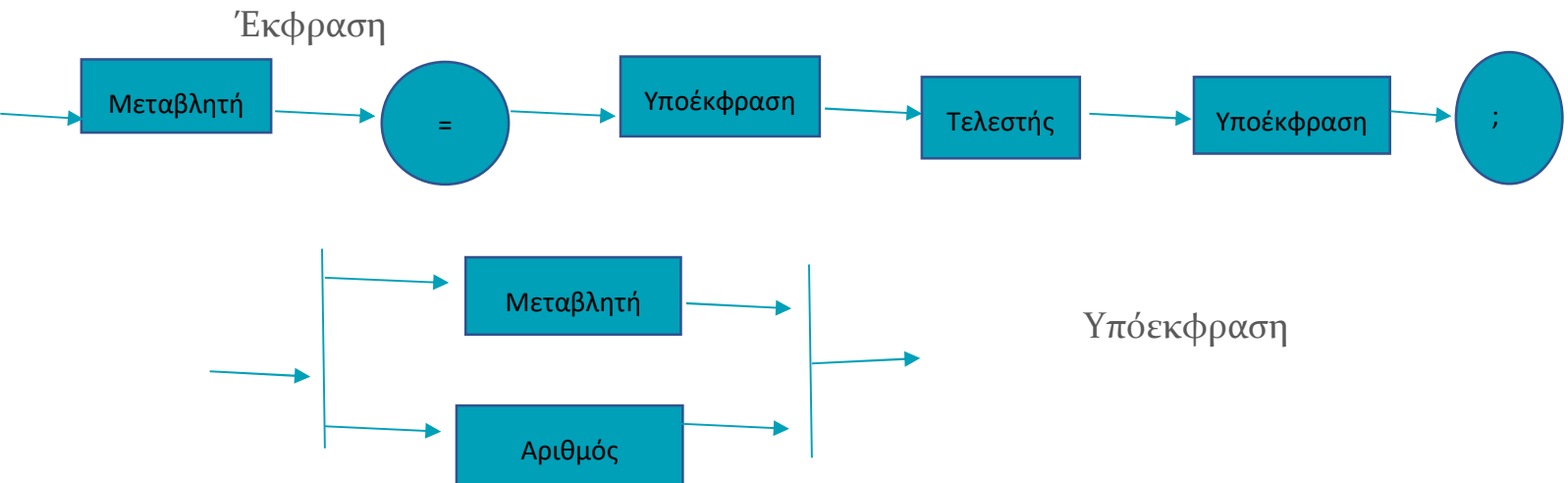


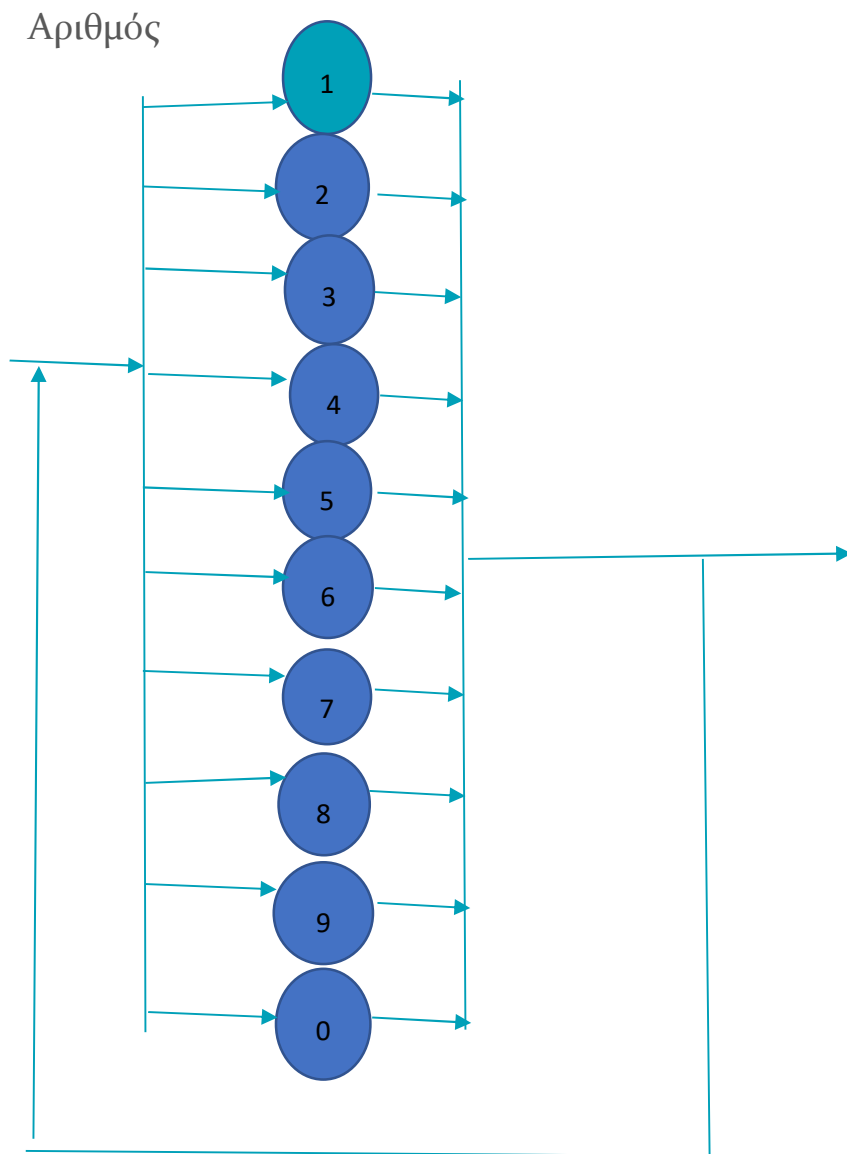
Ψηφίο="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".

Αριθμός=Ψηφίο{Ψηφίο}.

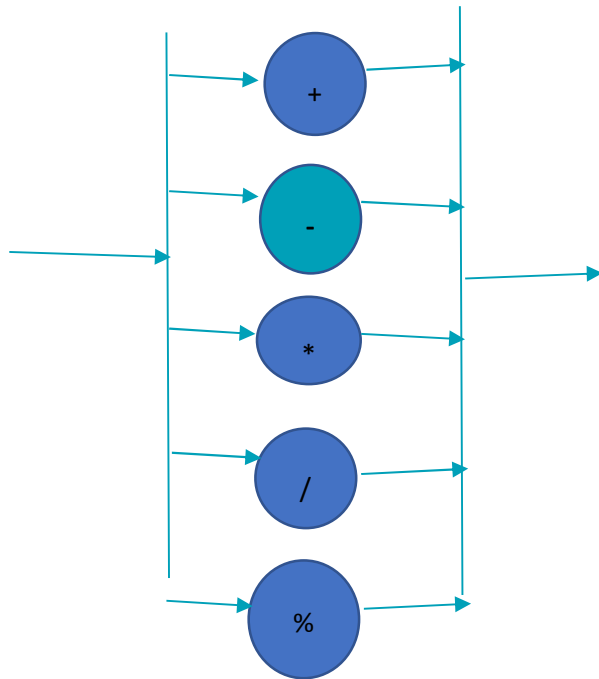
Τελεστής="+"|"-"|"\*"|"/"|"%".

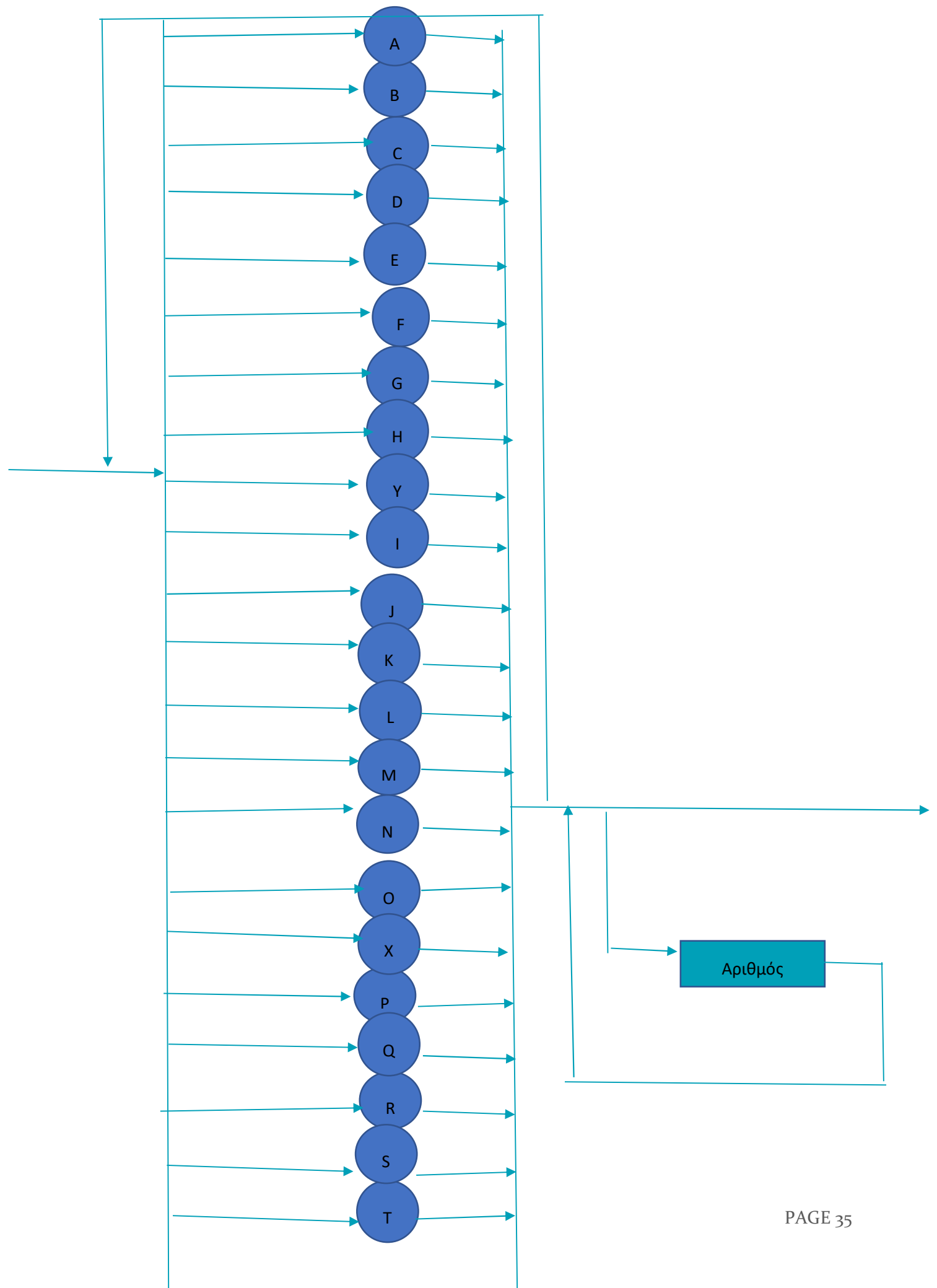
A) Συντακτικό διάγραμμα.

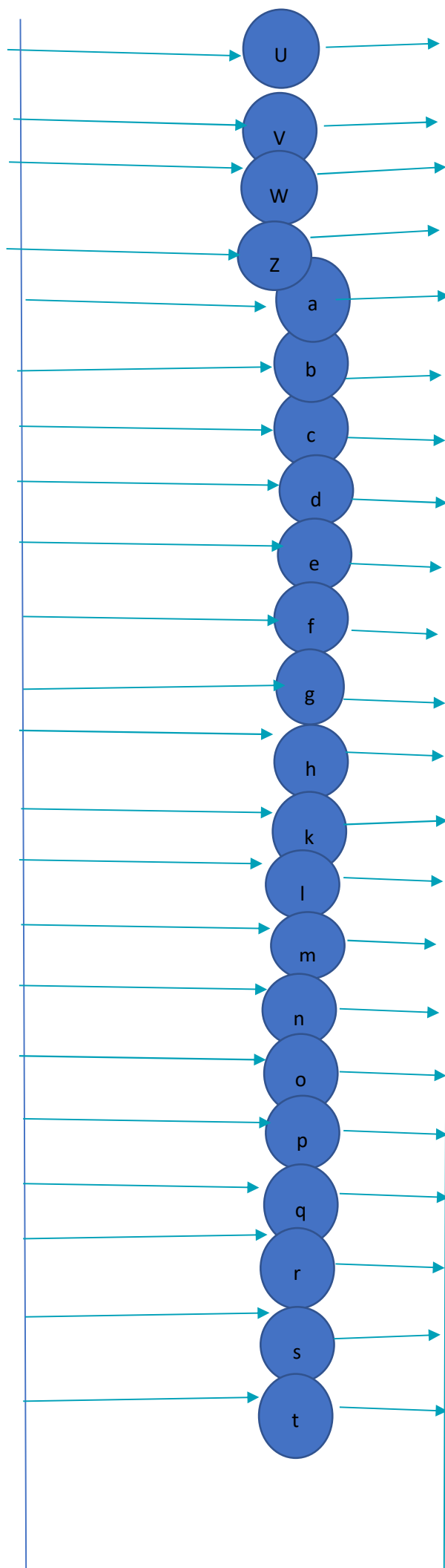


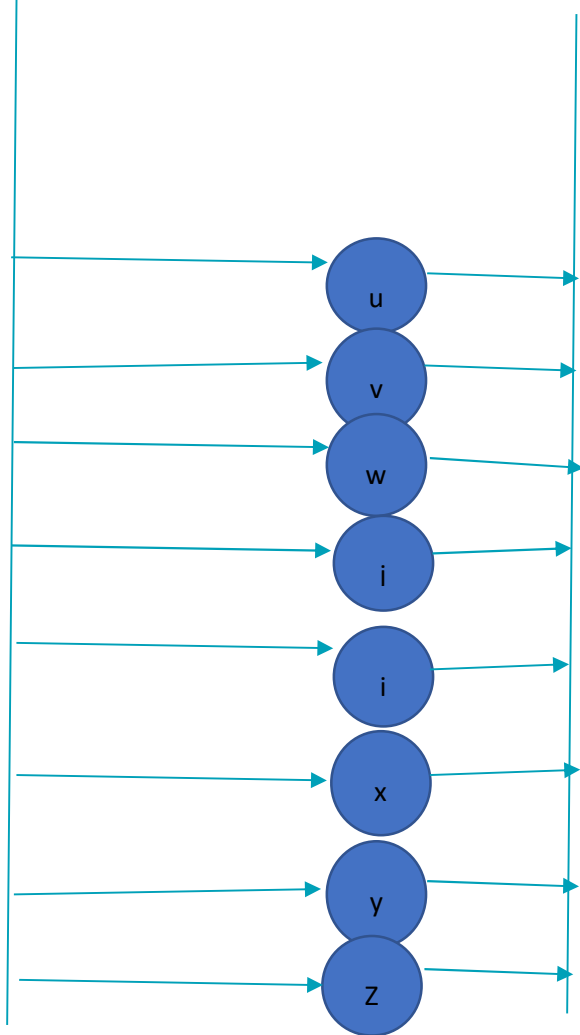


## Τελεστής









Μεταβλητή

ΒΙΒΛΙΟΓΡΑΦΙΑ

## Άσκηση (Ε)

Στο τμήμα δηλώσεων ορίζουμε τις εξής ακέραιες μεταβλητές : οι  $a_1, a_2, a_3, a_4, a_5$  οι οποίοι είναι μετρητές και εκφράζουν την ύπαρξη του σημείου, ευθείας, τρίγωνο, τετράγωνο και πεντάγωνο. Η μεταβλητή `verb` εκφράζει τον αριθμό των ρημάτων που αναγνωρίζονται από τον scanner της flex και η μεταβλητή `invalid` τον αριθμό των χαρακτήρων που δεν αναγνωρίζονται (σύμφωνα με την οριζόμενη γραμματική). Η μεταβλητή `counti` εκφράζει τον αριθμό των κανόνων που εκτελούνται. Τέλος ορίζουμε ένα ακέραιο πίνακα 5 θέσεων που θα παίζει το ρόλο της παρουσίας ή απουσίας ενός γεωμετρικού σχήματος στην πρόταση που δόθηκε από τον χρήστη .

Με το `%%` τώρα ξεκινάει το μέρος του προγράμματος που σχετίζεται με τους κανόνες. Αν ο scanner διαβάσει έκφραση που ταιριάζει με το pattern 'σημείο' τότε θα πραγματοποιηθεί δράση που είναι να εκτυπώνει στην κονσόλα το μήνυμα γεωμετρική οντότητα και το κομμάτι που να αναγνωρίστηκε και να αυξάνει τον μετρητή `a1`. Η ακριβώς ίδια διαδικασία ακολουθείται για τα 'ευθεία', 'τρίγωνο', 'τετράγωνο', 'πεντάγωνο' με την μόνη διαφορά ότι αυξάνονται οι αντίστοιχοι δικοί του μετρητές. Επίσης αν βρεθεί στην πρόταση που θα δώσει ο χρήστης το ρήμα δίνεται εκτυπώνει μήνυμα επιτυχίας και αυξάνει το `verb`. Αν δοθούν οποιαδήποτε 5 γράμματα A,B,Γ,Δ,E σε οποιαδήποτε σειρά θέτουμε το στοιχείο `i` του πίνακα `count` ίσο με 1, αν δοθούν 4 γράμματα A,B,Γ,Δ,E σε οποιαδήποτε σειρά τότε θέτουμε το στοιχείο 1 του πίνακα με 1, αν δοθούν 3 γράμματα A,B,Γ,Δ,E σε τυχαία σειρά θέτουμε το 2 στοιχείο με 1, αν δοθούν 2 γράμματα A,B,Γ,Δ,E,Z,H,Θ κάνουμε το `count[3]=1` αλλιώς αν δοθεί ένα γράμμα από τα A,B,Γ,Δ,E,Z,H,Θ το `count[4]` ίσο με 1. Σε όλες τις παραπάνω περιπτώσεις εκτυπώνεται το λεκτικό που βρέθηκε και αυξάνεται ο μετρητής `counti`.

Αν αλλάζουμε σε νέα γραμμή στην είσοδο ελέγχουμε την τιμή του `invalid`, `counti` και του `verb`. Αν είναι ο `invalid` τότε σημαίνει ότι η είσοδος που δόθηκε ως κανονική έκφραση είναι έγκυρη, αν είναι το `counti==1` τότε χρησιμοποιήθηκε μόνο ένα κανόνας δηλαδή μόνο ένα γεωμετρικό σχήμα που αποτελείται από τις προσδιοριζόμενες κορυφές που είναι και το ζητούμενο και αν το `verb` είναι ίσο με 1 τότε μόνο ένα ρήμα δόθηκε στην πρόταση. Αν ισχύει αυτό ελέγχουμε αν ταιριάζουν οι τιμές των  $a_1, a_2, a_3, a_4, a_5$  με τα στοιχεία 0,1,2,3,4,5 του `count`. (πχ δηλαδή αν δόθηκε η λέξη τρίγωνο να δόθηκαν και τρία σημεία όντως τρία σημεία). Αν υπάρχει αντιστοιχία η πρόταση αναγνωρίστηκε με επιτυχία. Αλλιώς η γεωμετρική οντότητα στην πρόταση δεν ταιριάζει με το πλήθος των κορυφών που όντως δόθηκε και εκφράζεται μήνυμα λάθους. Αν δεν είχε αληθής ούτε η αρχική συνθήκη στο πρώτο `if` τότε θα λείπει το ρήμα ή θα χρησιμοποιήθηκε γράμμα μη αποδεκτό με την γραμματική που αναγνωρίζεται. Γίνεται τερματισμός του προγράμματος. Τέλος αν

δόθηκε μη αναγνωρίσιμος χαρακτήρας (οτιδήποτε άλλο που δεν ταιριάζει με τους πάνω κανόνες) αυξάνουμε την τιμή του invalid κατά 1.

Τέλος καλούμε την `yylex()` (μέσα στην `main`) η οποία είναι τύπου `int` (επιστρέφει το λεκτικό που αναγνωρίζεται) το η οποία παράγει κώδικα σύμφωνα με τους κανόνες που γράψαμε πιο πάνω.

## BIBΛΙΟΓΡΑΦΙΑ

1. The StringBuilder Class Docs Oracle  
<https://docs.oracle.com/javase/tutorial/java/data/buffers.html>
2. Μεταγλωττιστές Μ.Βίρβου Κεφάλαιο 3 Τρόποι προσδιορισμού σύνταξης BNF,EBNF,συντακτικό διάγραμμα
3. Flex Manual <https://www.cs.virginia.edu/~cr4bd/flex-manual/>