Περιεχόμενα

[Εργασία - Θέμα 1 1](#_Toc63981125)

[Εκφώνηση 1](#_Toc63981126)

[Dependencies, τρόπος εκτέλεσης προγράμματος και τρόπος εκκίνησης εκτελέσιμου 1](#_Toc63981127)

[Παράδειγμα εκτέλεσης 2](#_Toc63981128)

[3](#_Toc63981129)

[Ντετερμινιστικό Αυτόματο Στοίβας 4](#_Toc63981130)

[Περιγραφή αρχείων πηγαίου κώδικα 4](#_Toc63981131)

[Τρόπος λειτουργίας αυτόματου – Αλγοριθμική ανάπτυξη της προγραμματιστικής υλοποίησης 5](#_Toc63981132)

[Βιβλιογραφία 6](#_Toc63981133)

# Εργασία - Θέμα 1

## Εκφώνηση

Υλοποιήστε Ντετερμινιστικό Αυτόματο Στοίβας (ΝΑΣ) που αναγνωρίζει εκφράσεις αποτελούμενες από τους χαρακτήρες ‘x’ και ‘y’, κατά τρόπον ώστε:

- όσοι χαρακτήρες ‘x’ εμφανίζονται συνολικά, άλλοι τόσοι χαρακτήρες ‘y’ εμφανίζονται συνολικά

- κοιτάζοντας την έκφραση από αριστερά προς τα δεξιά οι χαρακτήρες ‘y’ δεν είναι ποτέ περισσότεροι από τους χαρακτήρες ‘x’. Να τυπώνεται η αλληλουχία βημάτων που οδήγησαν στην αναγνώριση (ή στην απόρριψη) της έκφρασης.

## Dependencies, τρόπος εκτέλεσης προγράμματος και τρόπος εκκίνησης εκτελέσιμου

Το πρόγραμμα υλοποιήθηκε σε C++, σε Linux. Χρησιμοποιήθηκαν οι βιβλιοθήκες της C++ <stack> για χρήση στοίβας, <string> για χρήση string, <queue> για χρήση ουράς, <iostream> για την εκτύπωση μηνυμάτων στον χρήστη.

Σε περιβάλλον τερματικού (terminal):

* Εκτέλεση προγράμματος **g++ Subject1.cpp -o Subject1.out**
* Εκκίνηση εκτελέσιμου αρχείου (εφόσον το working directory είναι το directory στο οποίο δημιουργήθηκε το εκτελέσιμο με την παραπάνω εντολή) **./Subject1.out inputString**

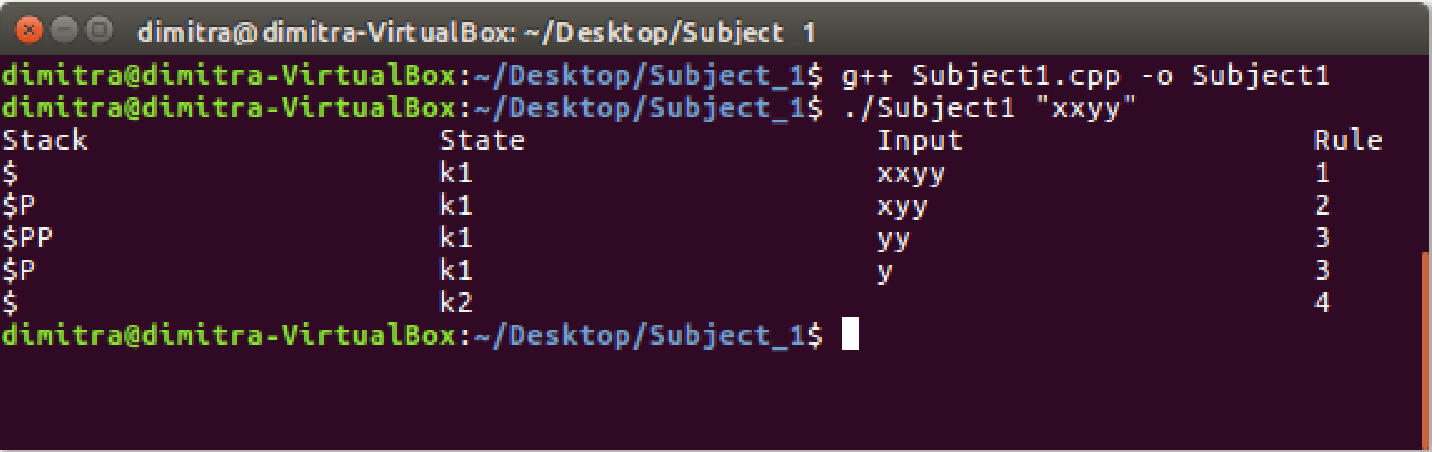
Όπου inputString = String με χαρακτήρες «x» και «y», η είσοδος που δίνει ο χρήστης για αναγνώριση από το Ντετερμινιστικό Αυτόματο Στοίβας. (Προσοχή! Η συμβολοσειρά εισόδου πρέπει να εσωκλειστεί σε “ ”). Επίσης ενδείκνυται ο χρήστης να χρησιμοποιεί λατινικούς χαρακτήρες στην είσοδο. Τέλος το extension του εκτελέσιμου μπορεί και να παραληφθεί επειδή εργαζόμαστε σε Linux.

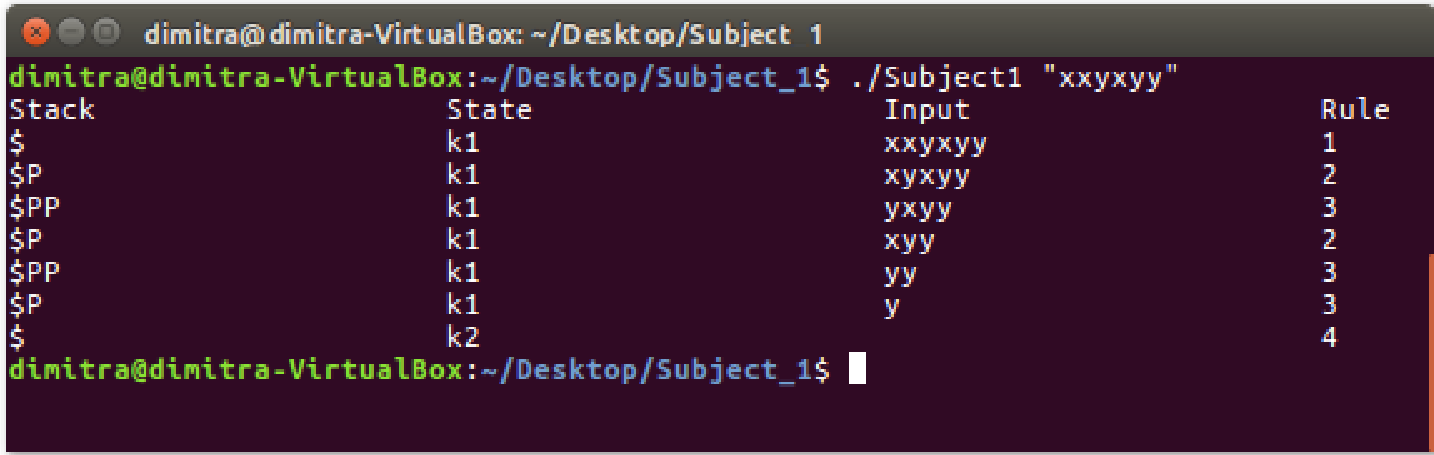
## Παράδειγμα εκτέλεσης

Σε περιβάλλον τερματικού καλούμε το εκτελέσιμο αρχείο (Subject1) και δίνουμε σαν όρισμα τη συμβολοσειρά εισόδου που επιθυμούμε. (παραδείγματος χάριν command 🡪 ./Subject1 “xxyy”) Παρακάτω επισυνάπτουμε ενδεικτικά screenshots τριών παραδειγμάτων εκτέλεσης.

Στην 1η στήλη εμφανίζονται τα περιεχόμενα (ή αλλιώς βάθος) στοίβας, στη 2η στήλη εμφανίζεται η τρέχουσα κατάσταση στην οποία βρίσκεται το αυτόματο, στη 3η στήλη εμφανίζεται η συμβολοσειρά εισόδου (διαγράφεται κάθε φορά το σύμβολο που αναγνωρίστηκε έπειτα και από την εφαρμογή κατάλληλου κανόνα) και τέλος στην 4η στήλη εμφανίζεται ο κανόνας που χρησιμοποιήθηκε (η αρίθμηση των κανόνων είναι ίδια με αυτή που διατυπώνεται παρακάτω).

Εικόνα 1 Επιτυχής αναγνώριση συμβολοσειράς "xxyy"





Εικόνα 2 Επιτυχής αναγνώριση συμβολοσειράς "xxyxyy"

## 

Εικόνα 3 Ανεπιτυχής αναγνώριση συμβολοσειράς "xxyx", η συμβολοσειρά δεν αναγνωρίστηκε

## Ντετερμινιστικό Αυτόματο Στοίβας

Το ζητούμενο Ντετερμινιστικό Αυτόματο Στοίβας χαρακτηρίζεται από την επτάδα M = (K, T, V, p, k1, $, {k2}) όπου:

* K το σύνολο καταστάσεων, Κ = {k1, k2}
* T το αλφάβητο εισόδου, T = {“x”, “y”, e} όπου e συμβολίζει το κενό
* V το αλφάβητο συμβόλων στοίβας, V = {P, $}
* Το p αποτελεί τη συνάρτηση στοίβας με τους εξής κανόνες:

1. p(k1, $, “x”) = (k1, $P)
2. p(k1, P, “x”) = (k1, PP)
3. p(k1, P, “y”) = (k1, e)
4. p(k1, e, “y”) = (k2, e)

* η k1 αποτελεί την αρχική κατάσταση
* το $ αποτελεί το αρχικό σύμβολο στοίβας
* το {k2} αποτελεί το σύνολο των τελικών καταστάσεων, με μία αποδεκτή τελική κατάσταση την k2

Πίνακας ελέγχου αυτομάτου

|  |  |  |  |
| --- | --- | --- | --- |
| **V/T** | **x** | **y** | **e** |
| **P** | Push(P) | Pop |  |
| **$** | Push(P) |  | k2 |

## Περιγραφή αρχείων πηγαίου κώδικα

**Subject1.cpp** περιέχει την Main συνάρτηση της λύσης, δέχεται ως όρισμα ένα string που περιέχει τους χαρακτήρες «x» και «y», τους τοποθετεί σε μια δομή σειράς (Queue) και ξεκινά την διαδικασία αναγνώρισης.

**Automaton.h** header file που περιέχει την υλοποίηση του Ντετερμινιστικού Αυτόματου Στοίβας, ο τρόπος λειτουργίας θα αναπτυχθεί λεπτομερώς παρακάτω.

**State.h** header file που περιέχει την υλοποίηση της κατάστασης που χρησιμοποιείται από το αυτόματο (περιλαμβάνει βασικά attributes που καθορίζουν έναν κανόνα και απαιτούνται για την επιλογή του κατάλληλου κανόνα από το αυτόματο, όπως τρέχουσα κατάσταση, κατάσταση μετάβασης, χαρακτήρας που υπάρχει στην κορυφή της στοίβας κ. α.) .

**PrettyPrint.h** header file που χρησιμοποιείται καθαρά και μόνο για την εκτύπωση του πίνακα καταστάσεων, με όσο το δυνατό πιο ευανάγνωστο τρόπο, στον χρήστη.

## Τρόπος λειτουργίας αυτόματου – Αλγοριθμική ανάπτυξη της προγραμματιστικής υλοποίησης

Ο τρόπος λειτουργίας του αυτόματου είναι αρκετά απλός. Μέσω της μεθόδου Start περνάμε τη σειρά χαρακτήρων που εισήγαγε ο χρήστης, αφού πρώτα έχουμε τοποθετήσει τη συμβολοσειρά εισόδου σε ένα queue (χρησιμοποιήθηκε ουρά σαν δομή, διότι κάθε φορά το αυτόματο επιχειρεί να αναγνωρίσει τον 1ο από τα αριστερά χαρακτήρα της συμβολοσειράς, και εφόσον τον αναγνωρίσει τον διαγράφει από την ουρά, αξιοποιήθηκε η ιδιότητα της δομής queue First-In-First-Out).

Στη συνέχεια, καλώντας κάθε φόρα τη μέθοδο Feed και τροφοδοτώντας τη με ένα χαρακτήρα, το αυτόματο ελέγχει αν υπάρχει κανόνας που να ικανοποιεί την καινούργια είσοδο. Για την εύρεση του κατάλληλου κανόνα, ελέγχεται:

1. η κατάσταση στην οποία βρίσκεται το αυτόματο
2. το σύμβολο που υπάρχει στην κορυφή της στοίβας
3. καθώς και το σύμβολο εισόδου (κάθε φορά δίνεται σαν όρισμα στη Feed() ο πρώτος χαρακτήρας που βρίσκεται στην ουρά που περιλαμβάνει τη συμβολοσειρά εισόδου

Αν υπάρχει λοιπόν κανόνας που οι παραπάνω τρεις προδιαγραφές του ταιριάζουν με τα δεδομένα της στοίβας, του αυτομάτου και της συμβολοσειράς τη συγκεκριμένη χρονική στιγμή, τότε εφαρμόζεται ο κανόνας ως εξής:

1. Ανάλογα με τον κανόνα που πρέπει να εφαρμοστεί το αυτόματο είτε παραμένει στην ίδια κατάσταση ή μεταβαίνει σε καινούρια
2. Στη συνέχεια είτε τοποθετείται νέο σύμβολο στην κορυφή της στοίβας, είτε αφαιρείται το σύμβολο που υπάρχει στην κορυφή της στοίβας, αντικαθίσταται δηλαδή με το κενό
3. Και τέλος αφαιρείται από την ουρά το 1ο σύμβολο, δηλαδή ο χαρακτήρας που μόλις αναγνωρίστηκε.

Το αυτόματο συνεχίζει κατά τον ίδιο τρόπο μέχρι το πέρας της σειράς χαρακτήρων που έδωσε ο χρήστης, όπου ολοκληρώνεται και η διαδικασία αναγνώρισης. Συγκεκριμένα, η διαδικασία της αναγνώρισης ολοκληρώνεται όταν:

* η στοίβα περιέχει μόνο το αρχικό της σύμβολο, δηλαδή το $
* έχουν καταναλωθεί (δηλαδή επεξεργαστεί και αναγνωριστεί) όλοι οι χαρακτήρες από τη συμβολοσειρά εισόδου
* όταν κατάσταση του αυτόματου είναι η τελική κατάσταση, δηλαδή η k2

Μόνο εφόσον ισχύουν και οι 3 παραπάνω προϋποθέσεις, η συμβολοσειρά εισόδου έχει αναγνωριστεί επιτυχώς. Αν όμως κατά τη διάρκεια αναγνώρισης δεν υπάρχει κανόνας που να ικανοποιεί την είσοδο, δηλαδή τα δεδομένα της στοίβας, του αυτομάτου και της συμβολοσειράς για τη συγκεκριμένη χρονική στιγμή δεν ταιριάζουν με κανενός κανόνα τις προδιαγραφές, τότε το αυτόματο επιστρέφει μήνυμα σφάλματος στον χρήστη και η διαδικασία αναγνώρισης σταματά.

# Εργασία - Θέμα 2

## Εκφώνηση

Υλοποιήστε μια γεννήτρια συμβολοσειρών για την παρακάτω γραμματική:

<Ε>::=(<Y>)

<Y>::=<A><B>

<A>::=v|<E>

<B>::=-<Y>|+<Y>|ε

Λάβετε μέριμνα ώστε η διαδικασία να τερματίζεται. Το πρόγραμμά σας θα πρέπει να τυπώνει τα βήματα της παραγωγής.

## Dependencies, τρόπος εκτέλεσης προγράμματος και τρόπος εκκίνησης εκτελέσιμου

Το πρόγραμμα υλοποιήθηκε σε C++, σε Linux. Χρησιμοποιήθηκαν οι βιβλιοθήκες της C++ <vector> και <list> για χρήση δυναμικών δομών δεδομένων, <string> για χρήση string, <random>, <stdlib> και <time> για χρήση ψευδοτυχαίων γεννητριών, <iostream> για την εκτύπωση μηνυμάτων στον χρήστη και <algorithm> για την χρήση εύχρηστων συναρτήσεων σχετικά με την προσπέλαση στοιχείων σε lists και vectors. Επίσης για την επιτυχή μεταγλώττιση του προγράμματος απαιτείται να δηλωθεί η c++ version 11 ή 14 ώστε να χρησιμοποιηθεί από τον compiler καθώς τα παραπάνω libraries έχουν αναπτυχθεί μεταγενέστερα και δεν υπάρχουν στο version 98 (default c++ version που χρησιμοποιεί ο compiler της, o g++).

Σε περιβάλλον τερματικού (terminal):

* Εκτέλεση προγράμματος με χρήση του c++ version 11 **g++ -std=c++11 Subject2.cpp -o Subject2.out**
* Εκτέλεση προγράμματος με χρήση του c++ version 14 **g++ -std=c++14 Subject2.cpp -o Subject2.out**
* Εκκίνηση εκτελέσιμου αρχείου (εφόσον το working directory είναι το directory στο οποίο δημιουργήθηκε το εκτελέσιμο με την παραπάνω εντολή) **./Subject2.out**

Τέλος το extension του εκτελέσιμου μπορεί και να παραληφθεί επειδή εργαζόμαστε σε περιβάλλον Linux.

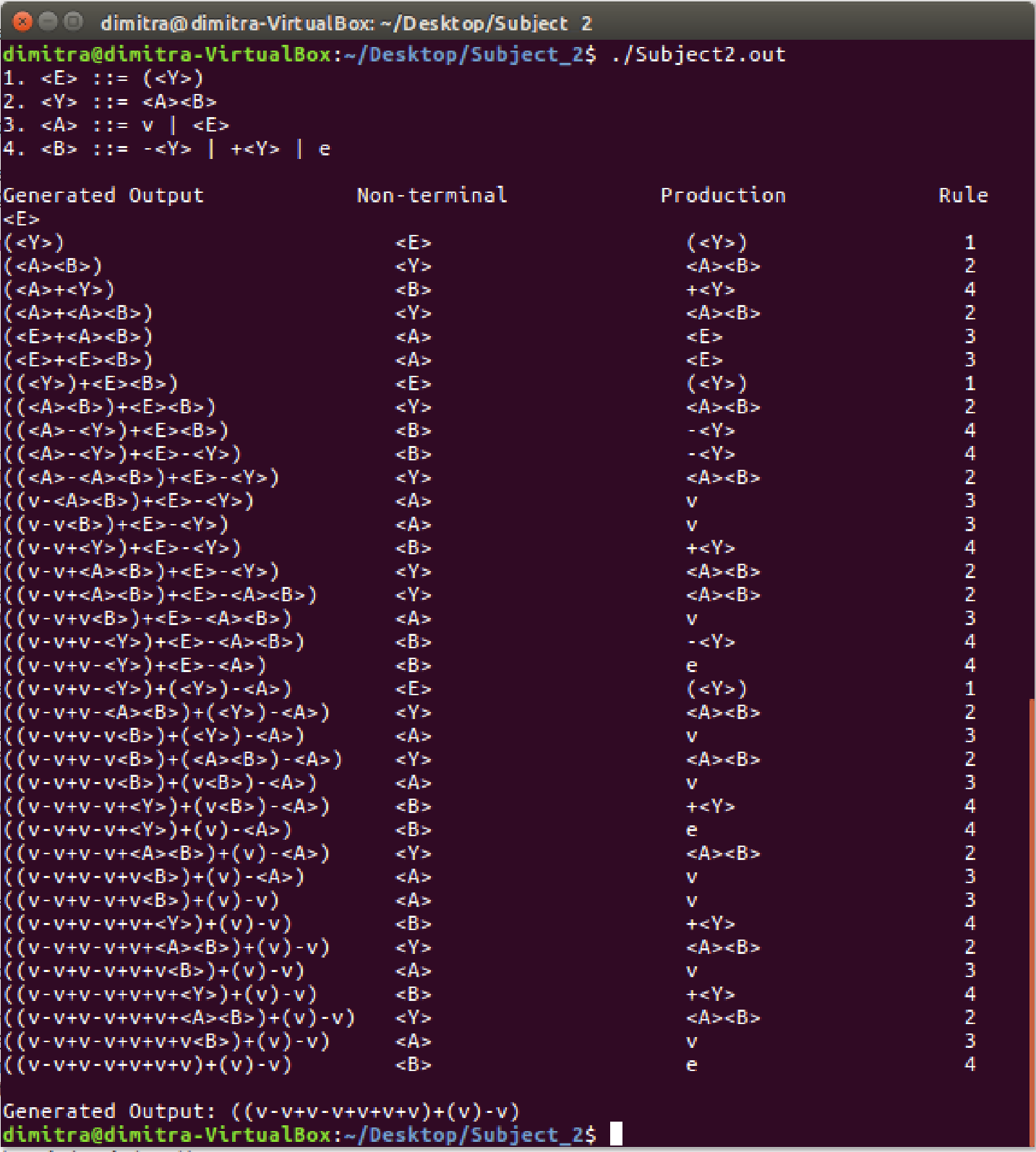
## Παράδειγμα εκτέλεσης

Σε περιβάλλον τερματικού καλούμε το εκτελέσιμο αρχείο (Subject2). Παρακάτω επισυνάπτουμε ενδεικτικά screenshots παραδειγμάτων εκτέλεσης.

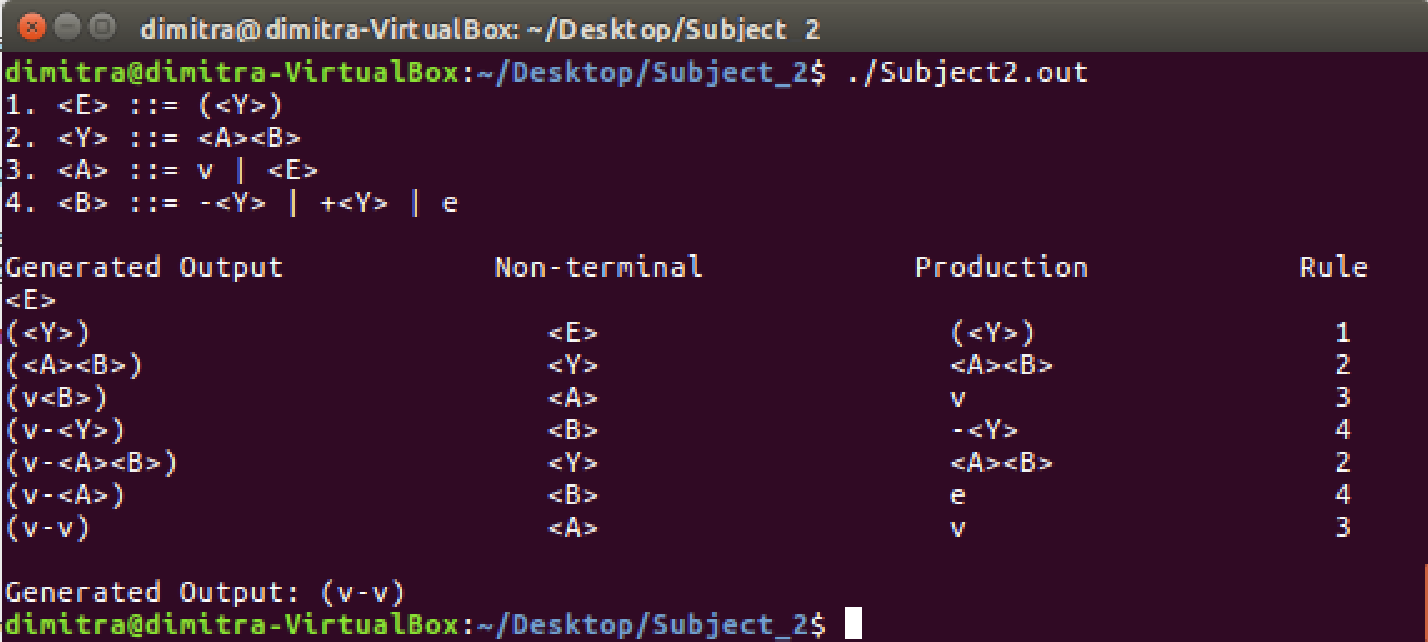
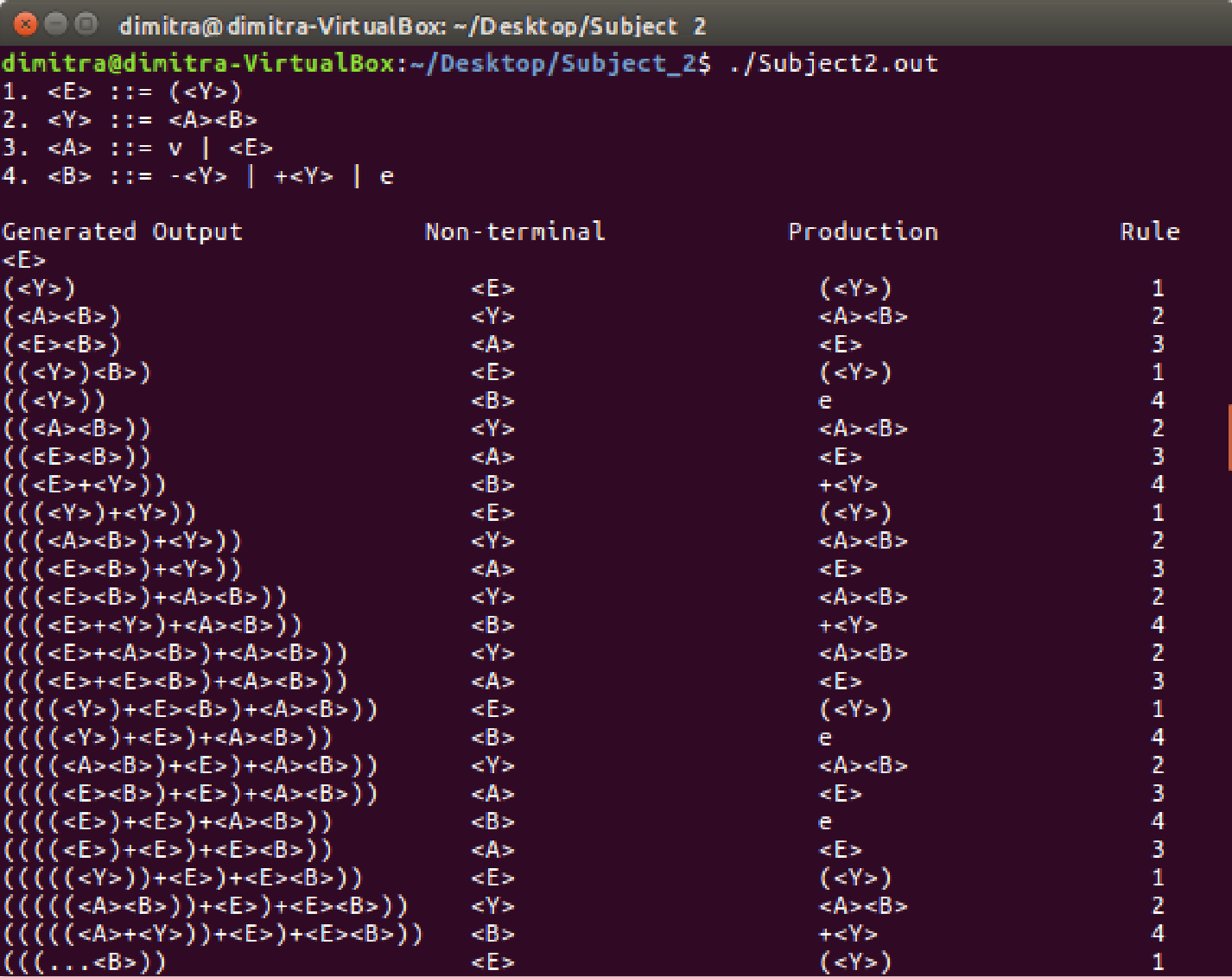
Αρχικά εμφανίζονται αριθμημένοι οι κανόνες παραγωγής της γραμματικής που έχει δοθεί. Ακολουθούν τα βήματα παραγωγής της συμβολοσειράς εξόδου. Στην 1η στήλη εμφανίζεται η συμβολοσειρά εξόδου που παράχθηκε κατόπιν εφαρμογής του εκάστοτε κανόνα παραγωγής. Στην 2η στήλη παρουσιάζεται το μη τερματικό σύμβολο της προηγούμενη συμβολοσειράς εξόδου, το οποίο έχει πλέον αντικατασταθεί (λόγω εφαρμογής του κανόνα). Στην 3η στήλη εμφανίζεται με τι αντικαταστάθηκε το μη τερματικό που επιλέχθηκε από την προηγούμενη συμβολοσειρά εξόδου (το τμήμα δηλαδή του κανόνα παραγωγής που εφαρμόστηκε). Στην 4η στήλη εμφανίζεται ο κανόνας παραγωγής που εφαρμόστηκε στην προηγούμενη συμβολοσειρά εξόδου και οδήγησε στην τρέχουσα παραγωγή της συμβολοσειράς εξόδου. (η αρίθμηση των κανόνων είναι ίδια με αυτή που διατυπώνεται παρακάτω).

Αν η συμβολοσειρά εξόδου που παράγεται είναι πολύ μεγάλη για να εκτυπωθεί, εκτυπώνονται τα 3 πρώτα σύμβολά της ακολουθούμενα από τρεις τελείες και τα 3 τελευταία σύμβολά της.

Όταν τελειώσει η διαδικασία παραγωγής εμφανίζεται η τελική συμβολοσειρά που παράχθηκε.

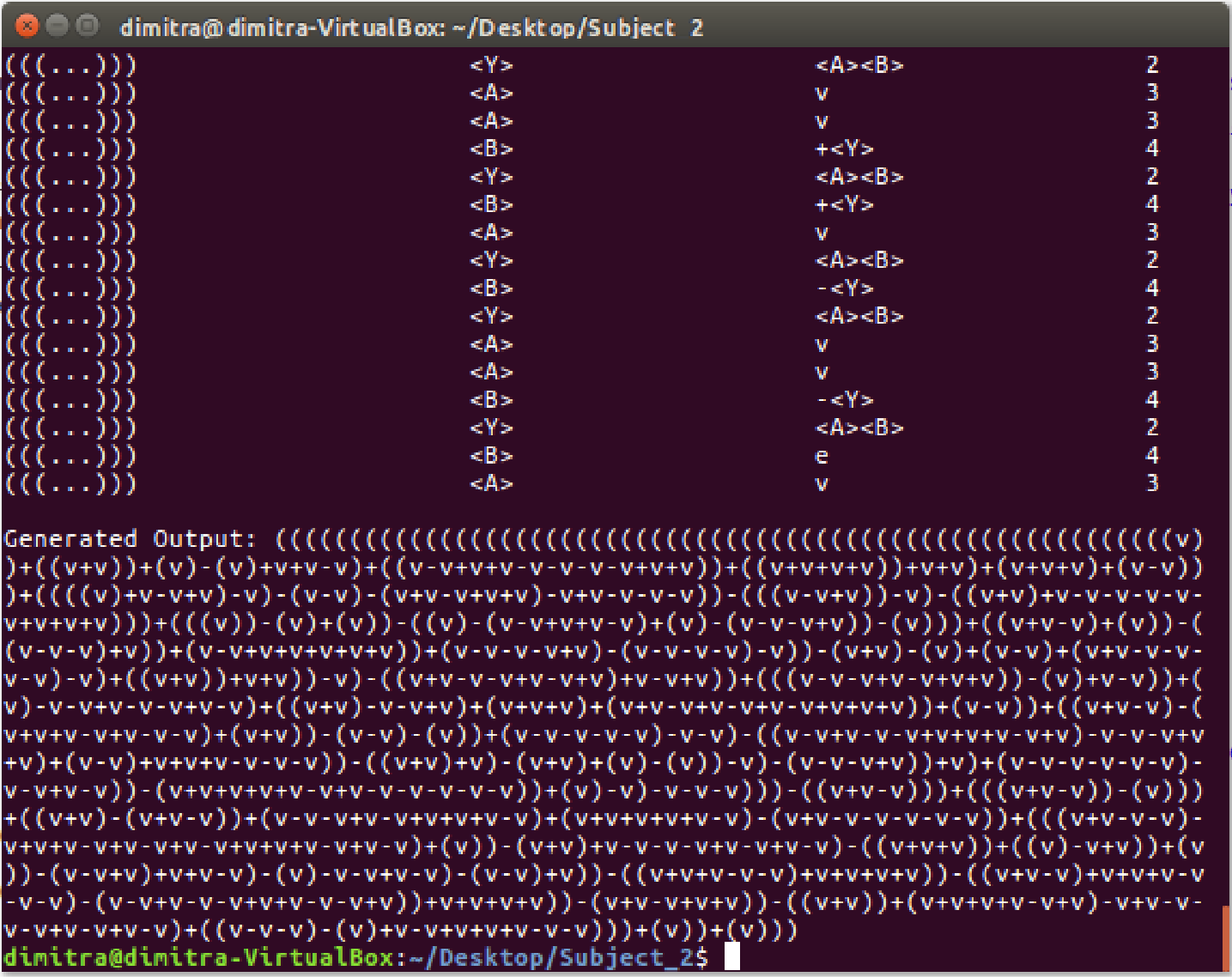


Εικόνα 4 Παραγωγή μέτριου μήκους συμβολοσειράς



Εικόνα 5 Παραγωγή μικρού μήκους συμβολοσειράς

Η παραγωγή της παραπάνω συμβολοσειράς συνεχίζεται και το τέλος απεικονίζεται παρακάτω.



Εικόνα 6 Παραγωγή μεγάλου μήκους συμβολοσειράς

## Μη Ντετερμινιστικό Πεπερασμένο Αυτόματο

Αρχικά η δοθείσα γραμματική δίνεται σε BNF μορφή τα μη τερματικά σύμβολα της γραμματικής είναι τα E, Y, A, B ενώ τα τερματικά σύμβολα αποτελούνται από τους χαρακτήρες (, ), v, -, +

Δοθέντος μιας γραμματικής τα αυτόματα χρησιμοποιούνται είτε για την αναγνώριση μιας συμβολοσειράς που μπορεί να παραχθεί βάση των κανόνων παραγωγής που καθορίζονται από την εκάστοτε γραμματική, είτε μπορούν να χρησιμοποιηθούν ως γεννήτριες για την παραγωγή έγκυρων συμβολοσειρών που ανήκουν στην γλώσσα που περιγράφει η γραμματική του αυτομάτου. Στο συγκεκριμένο ερώτημα μας ζητείτε η 2η λειτουργία του αυτομάτου.

Το αυτόματο είναι μη Ντετερμινιστικό διότι σε δύο από τους κανόνες παραγωγές υπάρχει δυνατότητα επιλογής όσων αφορά το μέρος τους κανόνα που θα επιλεγεί να εφαρμοστεί. Επίσης εκ πρώτης όψεως το αυτόματο είναι μη πεπερασμένο λόγω της ύπαρξης του αναδρομικού κανόνα που μπορεί να οδηγήσει σε ατέρμων βρόχο αν εφαρμοστεί διαδοχικά πολλές φορές.

Η παραδοχή που λάβαμε για να μπορεί να τερματίζεται η διαδικασία είναι η εξής. Στον αναδρομικό κανόνα 3 την πρώτη φορά επιλέγεται τυχαία το μέρος του κανόνα που θα εφαρμοστεί. Στις επόμενες εφαρμογές του κανόνα μειώνουμε διαδοχικά την πιθανότητα επιλογής του 2ου μέρους. Έτσι κατά την κλήση του κανόνα 3:

* Την 1η φορά το v έχει πιθανότητα 50%, το <Ε> έχει πιθανότητα 50%
* Την 2η φορά εφαρμογής του κανόνα το v έχει πιθανότητα 60%, το <Ε> έχει πιθανότητα 40%
* Την 3η φορά εφαρμογής του κανόνα το v έχει πιθανότητα 70%, το <Ε> έχει πιθανότητα 30%
* Την 4η φορά εφαρμογής του κανόνα το v έχει πιθανότητα 80%, το <Ε> έχει πιθανότητα 20%
* Την 5η φορά εφαρμογής του κανόνα το v έχει πιθανότητα 90%, το <Ε> έχει πιθανότητα 10%

Εφόσον η πιθανότητα επιλογής του <Ε> στον κανόνα 3, φτάσει στο 10%, δεν την μειώνουμε άλλο. Επομένως εσκεμμένα πειράζουμε τον αλγόριθμο, ώστε να τερματίζει η διαδικασία.

Αρίθμηση των κανόνων η οποία ακολουθείται και στο πρόγραμμά μας:

1. <Ε>::=(<Y>)
2. <Y>::=<A><B>
3. <A>::=v|<E>
4. <B>::=-<Y>|+<Y>|ε

## Περιγραφή αρχείων πηγαίου κώδικα

**Subject2.cpp** περιέχει την Main συνάρτηση της λύσης, η οποία δημιουργεί ένα Automaton αντικείμενο και καλεί τον constructor ώστε να ξεκινήσει η διαδικασία παραγωγής.

**Automaton.h** header file που περιέχει την υλοποίηση του μη Ντετερμινιστικού Πεπερασμένου Αυτόματου, ο τρόπος λειτουργίας θα αναπτυχθεί λεπτομερώς παρακάτω.

**PrettyPrint.h** header file που χρησιμοποιείται καθαρά και μόνο για την εκτύπωση των κανόνων και των βημάτων παραγωγής, με όσο το δυνατό πιο ευανάγνωστο τρόπο, στον χρήστη.

## Τρόπος λειτουργίας αυτόματου – Αλγοριθμική ανάπτυξη της προγραμματιστικής υλοποίησης

Αρχικά στον constructor της κλάσης Automaton γίνονται οι απαραίτητες αρχικοποιήσεις, σαν αρχικό σύμβολο της συμβολοσειράς εξόδου τοποθετείται το <Ε> και εφαρμόζεται ο κανόνας παραγωγής 1. Η διαδικασία των διαδοχικών παραγωγών ξεκινάει με την κλήση της μεθόδου Start(). Αρχικά εντοπίζονται και καταχωρούνται σε μια ξεχωριστή δομή τα διαθέσιμα μη τερματικά σύμβολα που υπάρχουν στη συμβολοσειρά εξόδου και μπορούν να αναλυθούν παραπάνω. Αν μπορούν να εφαρμοστούν παραπάνω από ένα κανόνες, επιλέγεται τυχαία ένας κανόνας. Αν ο κανόνας παραγωγής περιέχει πάνω από ένα μέρος παραγωγής, επιλέγεται τυχαία το μέρος του κανόνα που θα εφαρμοστεί. Εξαιρείται ο κανόνας παραγωγής 3, εφόσον επιλεγεί για να εφαρμοστεί ο συγκεκριμένος κανόνας, η επιλογή του μέρους γίνεται όπως αναλύθηκε παραπάνω.

Η διαδικασία συνεχίζεται μέχρι η συμβολοσειρά εξόδου να περιέχει μόνο τερματικά σύμβολα.

Για την χρήση ισοπίθανων ψευδοτυχαίων επιλογών χρησιμοποιήθηκε random number generator της c++, ενώ για την χρήση ψευδοτυχαίων επιλογών με βάρη χρησιμοποιήθηκε η συνάρτηση rand() με seed τον χρόνο ώστε να παίρνουμε διαφορετικές επιλογές (με την μονότονη αύξηση του χρόνου).

# Εργασία - Θέμα 3

## Εκφώνηση

Δίνεται η γραμματική:

1. S 🡪(X)
2. X 🡪YZ
3. Y 🡪α | β | S
4. Z 🡪\*X | -X |+X|ε

Εξετάστε αν η γραμματική είναι LL(1). Αν είναι LL(1), τότε κατασκευάστε συντακτικό αναλυτή top-down που αναγνωρίζει την εκάστοτε συμβολοσειρά ή απαντά αρνητικά ως προς τη συντακτική της ορθότητα. Να επιστρέφεται το σχετικό δέντρο και να εκτυπώνεται. Να γίνει επίδειξη για την έκφραση ((β-α)\*(α+β)).

## Dependencies, τρόπος εκτέλεσης προγράμματος και τρόπος εκκίνησης εκτελέσιμου

Το πρόγραμμα υλοποιήθηκε σε C++, σε Linux. Χρησιμοποιήθηκαν οι βιβλιοθήκες της C++ <queue> για χρήση ουράς , <stack> για χρήση στοίβας, <vector> και <list> για χρήση δυναμικών δομών δεδομένων, <string> για χρήση string, <math> για χρήση της συνάρτησης ceil, <iostream> για την εκτύπωση μηνυμάτων στον χρήστη και <algorithm> για την χρήση εύχρηστων συναρτήσεων σχετικά με την προσπέλαση στοιχείων σε lists και vectors. Επίσης για την επιτυχή μεταγλώττιση του προγράμματος απαιτείται να δηλωθεί η c++ version 11 ή 14 ώστε να χρησιμοποιηθεί από τον compiler καθώς τα παραπάνω libraries έχουν αναπτυχθεί μεταγενέστερα και δεν υπάρχουν στο version 98 (default c++ version που χρησιμοποιεί ο compiler της, o g++).

Σε περιβάλλον τερματικού (terminal):

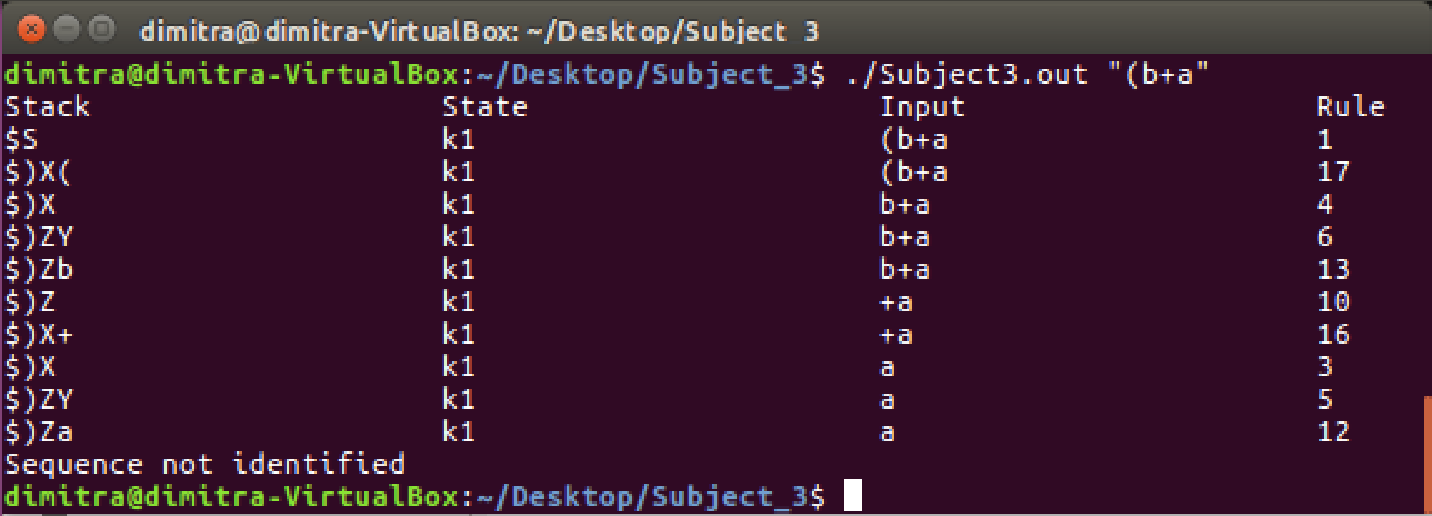
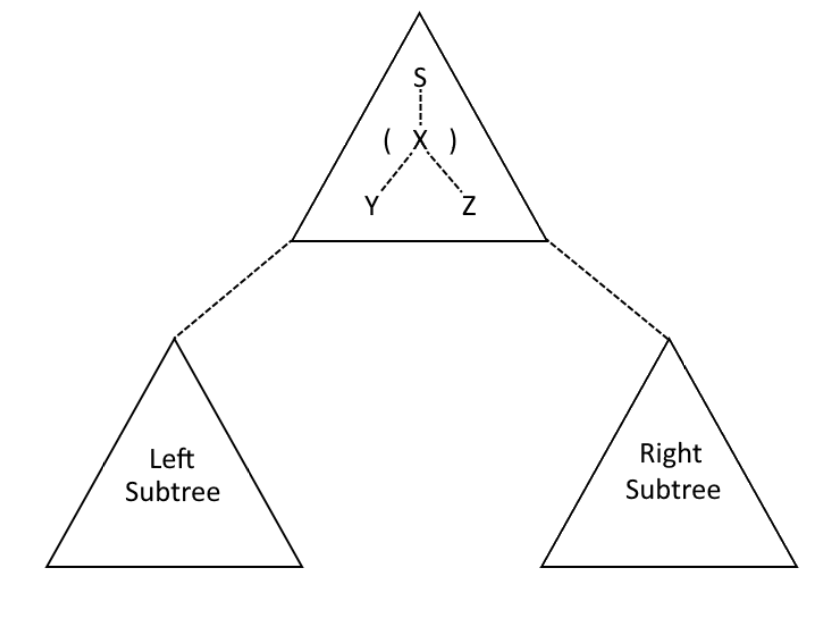
* Εκτέλεση προγράμματος με χρήση του c++ version 11 **g++ -std=c++11 Subject3.cpp -o Subject3.out**
* Εκτέλεση προγράμματος με χρήση του c++ version 14 **g++ -std=c++14 Subject3.cpp -o Subject3.out**
* Εκκίνηση εκτελέσιμου αρχείου (εφόσον το working directory είναι το directory στο οποίο δημιουργήθηκε το εκτελέσιμο με την παραπάνω εντολή) **./Subject3.out inputString**

Όπου inputString = String με χαρακτήρες μέσα σε διπλά εισαγωγικά, η είσοδος που δίνει ο χρήστης για αναγνώριση από το Ντετερμινιστικό Αυτόματο Στοίβας. Επίσης ενδείκνυται ο χρήστης να χρησιμοποιεί λατινικούς χαρακτήρες στην είσοδο. Τέλος το extension του εκτελέσιμου μπορεί και να παραληφθεί επειδή εργαζόμαστε σε Linux. Τέλος το extension του εκτελέσιμου μπορεί και να παραληφθεί επειδή εργαζόμαστε σε περιβάλλον Linux.

## Παράδειγμα εκτέλεσης

Σε περιβάλλον τερματικού καλούμε το εκτελέσιμο αρχείο (Subject3) και δίνουμε σαν όρισμα τη συμβολοσειρά εισόδου που επιθυμούμε. (παραδείγματος χάριν command 🡪 ./Subject3 “(a+b)” ) Παρακάτω επισυνάπτουμε ενδεικτικά screenshots παραδειγμάτων εκτέλεσης.

Στην 1η στήλη εμφανίζονται τα περιεχόμενα (ή αλλιώς βάθος) στοίβας, στη 2η στήλη εμφανίζεται η τρέχουσα κατάσταση στην οποία βρίσκεται το αυτόματο, στη 3η στήλη εμφανίζεται η συμβολοσειρά εισόδου (διαγράφεται κάθε φορά το σύμβολο που αναγνωρίστηκε έπειτα και από την εφαρμογή κατάλληλου κανόνα) και τέλος στην 4η στήλη εμφανίζεται ο κανόνας που χρησιμοποιήθηκε (η αρίθμηση των κανόνων είναι ίδια με αυτή που διατυπώνεται παρακάτω).

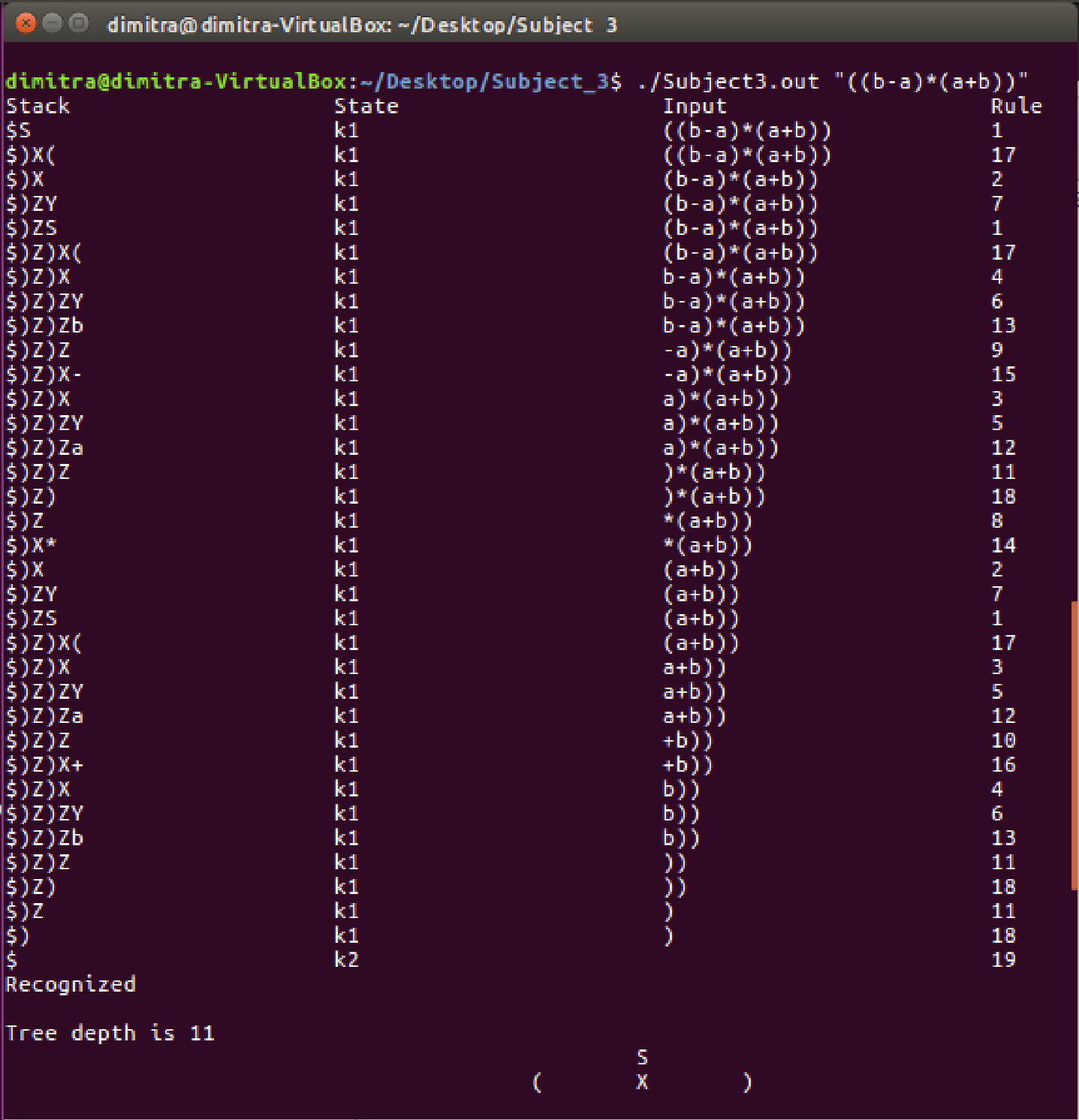
Το συντακτικό δέντρο εκτυπώνεται με την παρακάτω δομή. Στην κορυφή του δέντρου εκτυπώνονται οι κοινοί κόμβοι (λόγω επιλογής των δύο κανόνων θα παραμένουν ίδιοι για οποιαδήποτε συμβολοσειρά που θα δοθεί σαν είσοδος). Ως δεξί υποδένδρο θεωρούνται όλοι οι κόμβοι που εμφανίζονται κάτω από τον πρωτοεμφανιζόμενο κόμβο Z. Ως αριστερό υποδένδρο θεωρούνται όλοι οι κόμβοι που εμφανίζονται κάτω από τον πρωτοεμφανιζόμενο κόμβο Y. Ακολουθήσαμε αυτή τη δομή λόγω της δυσκολίας παράλληλης εκτύπωσης των κόμβων που ανήκαν σε διαφορετικά υποδένδρα.

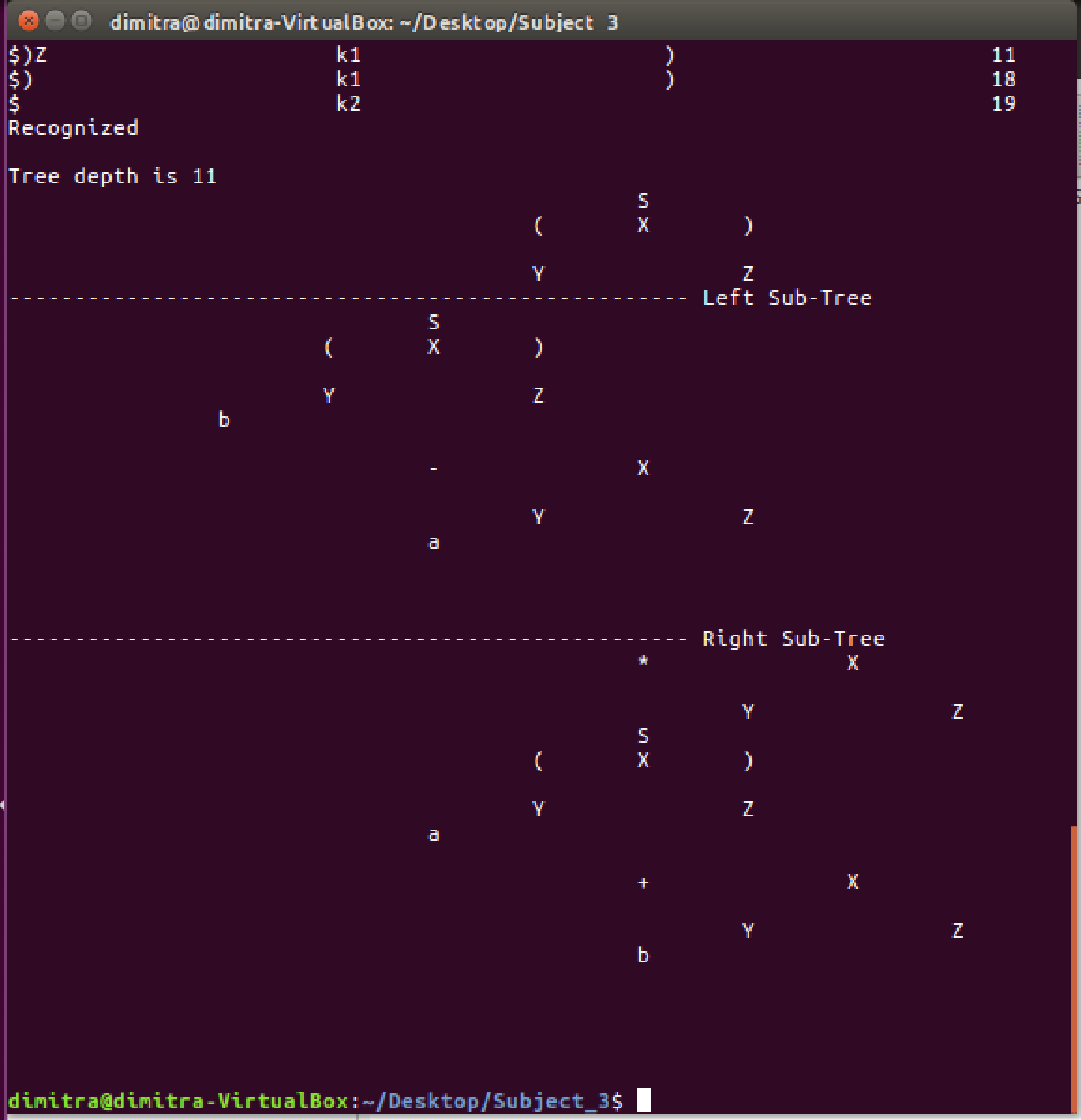
Εικόνα 9 Δομή εκύπωσης συντακτικού δέντρου

Εικόνα 7 Ανεπιτυχής αναγνώριση συμβολοσειράς



Εικόνα 10 Επιτυχής αναγνώριση συμβολοσειράς





Εικόνα 11 επιτυχής αναγνώριση συμβολοσειράς ((b-a)\*(a+b))

## Απόδειξη γραμματικής LL(1)

Για τη δημιουργία του συντακτικού πίνακα υπολογίζουμε τα σύνολα FIRST, FOLLOW:

* FIRST(S) = { ( }
* FIRST(X) = {a, b, ( }
* FIRST(Y) = {a, b, (}
* FIRST(Z) = {\*, -, +, ε}
* FOLLO(S) = {$, \*, -, +, ) }
* FOLLOW(X) = { ) }
* FOLLOW(Y) = {\*, -,+, ) }
* FOLLOW(Z) = { ) }

Θεωρούμε ότι ο υπό κατασκευή πίνακας λέγεται M. Έτσι παίρνουμε τους κανόνες παραγωγής έναν προς έναν:

1. S->(X) FIRST((X)) = { ( } M(S, ( ) = S -> (X)

2. X->YZ FIRST(YZ) = FIRST(Y) = {a, b, ( }

M(X, a) = X->YZ M(X, b) = X->YZ M(X, ( ) = X->YZ

Από τον κανόνα 2 προκύπτει ότι:

3.1 Υ -> a FIRST(a) = { a } M(Y, a) = Y -> a

3.2 Y -> b FIRST(b) = { b } M(Y, b) = Y -> b

3.3 Y->S FIRST(S) = { ( } M(Y, ( ) = Y -> S

Ομοίως συνεχίζουμε για τον κανόνα 4:

4.1 Z->\**X FIRST(\**X) = FIRST(\*) = { \* } M(Z, \*) = Z->\*X

4.2 Z->-X FIRST(-X) = FIRST(-) = { - } M(Z, -) = Z->-X

4.3 Z->+X FIRST(+X) = FIRST(+) = { + } M(Z, +) = Z->+X

4.4 Z->ε FIRST(ε) = { ε } => ε ∊ FIRST(ε)   
επομένως, εξετάζουμε το FOLLOW(Z) = { ) } M(Z, ) ) = Z -> ε

Συντακτικός πίνακας

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **V\T** | **(** | **)** | **\*** | **-** | **+** | **a** | **b** | **$** |
| **S** | S 🡪(X) |  |  |  |  |  |  |  |
| **X** | X 🡪YZ |  |  |  |  | X 🡪YZ | X 🡪YZ |  |
| **Y** | Y 🡪S |  |  |  |  | Y 🡪a | Y 🡪b |  |
| **Z** |  | Z 🡪ε | Z 🡪\*X | Z 🡪-X | Z 🡪+X |  |  |  |

Επειδή ο συντακτικός πίνακας δεν έχει επικαλύψεις, δηλαδή κάθε κελί του πίνακα περιέχει ένα ή κανένα κανόνα παραγωγής, η γραμματική είναι LL(1).

## Περιγραφή αρχείων πηγαίου κώδικα

**Subject3.cpp** περιέχει την Main συνάρτηση της λύσης, δέχεται ως όρισμα ένα string που περιέχει τους χαρακτήρες που δόθηκαν ως είσοδος, τους τοποθετεί σε μια δομή σειράς (Queue) και ξεκινά την διαδικασία αναγνώρισης.

**Automaton.h** header file που περιέχει την υλοποίηση του Ντετερμινιστικού Αυτόματου Στοίβας, και την δημιουργία του συντακτικού δέντρου βάση των παραγωγών που εφαρμόστηκαν. Ο τρόπος λειτουργίας θα αναπτυχθεί λεπτομερώς παρακάτω.

**State.h** header file που περιέχει την υλοποίηση της κατάστασης που χρησιμοποιείται από το αυτόματο (περιλαμβάνει βασικά attributes που καθορίζουν έναν κανόνα και απαιτούνται για την επιλογή του κατάλληλου κανόνα από το αυτόματο, όπως τρέχουσα κατάσταση, κατάσταση μετάβασης, χαρακτήρας που υπάρχει στην κορυφή της στοίβας κ. α.) .

**PrettyPrint.h** header file που χρησιμοποιείται καθαρά και μόνο για την εκτύπωση του πίνακα καταστάσεων και του συντακτικού δέντρου, με όσο το δυνατό πιο ευανάγνωστο τρόπο, στον χρήστη.

**SymbolTree.h** περιέχει τη δημιουργία των κόμβων από τους οποίους θα αποτελείται το συντακτικό δέντρο.

## Τρόπος λειτουργίας αυτόματου – Αλγοριθμική ανάπτυξη της προγραμματιστικής υλοποίησης

Αρχικά λαμβάνουμε την είσοδο του χρήστη και με κατάλληλη επεξεργασία την τοποθετούμε σε μια ουρά (όπως και στο 1ο ερώτημα χρησιμοποιήθηκε ουρά σαν δομή, διότι κάθε φορά το αυτόματο επιχειρεί να αναγνωρίσει τον 1ο από τα αριστερά χαρακτήρα της συμβολοσειράς, και εφόσον τον αναγνωρίσει τον διαγράφει από την ουρά, αξιοποιήθηκε η ιδιότητα της δομής queue First-In-First-Out). Στη καλούμε τη μεθόδο Start τη σειρά χαρακτήρων εισόδου.

Έπειτα, καλώντας κάθε φόρα τη μέθοδο Feed και τροφοδοτώντας τη με ένα χαρακτήρα, το αυτόματο ελέγχει αν υπάρχει κανόνας που να ικανοποιεί την καινούργια είσοδο. Για την εύρεση του κατάλληλου κανόνα, ελέγχεται:

1. η κατάσταση στην οποία βρίσκεται το αυτόματο
2. το σύμβολο που υπάρχει στην κορυφή της στοίβας
3. καθώς και το σύμβολο εισόδου (κάθε φορά δίνεται σαν όρισμα στη Feed() ο πρώτος χαρακτήρας που βρίσκεται στην ουρά που περιλαμβάνει τη συμβολοσειρά εισόδου

Η διαδικασία αναγνώρισης που ακολουθείται με χρήση του Αυτόματου Στοίβας είναι η ίδια που αναλύθηκε στο 1ο ερώτημα. Παράλληλα με τη διαδικασία αναγνώρισης, κάθε φορά που εφαρμόζεται ένας κανόνας παραγωγής, εκχωρείται στην αντίστοιχη θέση του δέντρου καθένα από τα σύμβολα που ανήκαν στο production part του κανόνα και ενσωματώθηκαν στη στοίβα. Κάθε κόμβος του συντακτικού δέντρου περιέχει έναν χαρακτήρα που αποτελεί τα δεδομένα του κόμβου, και 3 pointers που δείχνουν στα πιθανά παιδιά του κόμβου. Σύμφωνα με τους κανόνες της δοθέντος γραμματικής προκύπτει ότι το δέντρο είναι διατεταγμένο με κάθε κόμβο να έχει από 0 μέχρι 3 παιδιά. Σαρώνοντας το δέντρο από τη ρίζα προς τα φύλλα φαίνονται τα βήματα παραγωγής, ενώ τα φύλλα του δέντρου αποτελούν τη συμβολοσειρά που αναγνωρίστηκε.

Εφόσον η συμβολοσειρά που δόθηκε σαν είσοδος αναγνωριστεί, εκτυπώνεται το βάθος του δέντρου (επίπεδα) και το συντακτικό δέντρο με τη δομή που προαναφέρθηκε.

# Εργασία - Θέμα 4

## Εκφώνηση

Μιααα

## Τρόπος εκτέλεσης προγράμματος και τρόπος εκκίνησης εκτελέσιμου

Μπμπμππ

## Παράδειγμα εκτέλεσης

Ππππππ

## Περιγραφή αρχείων πηγαίου κώδικα

ππππ

## Αλγοριθμική ανάπτυξη της προγραμματιστικής υλοποίησης

ππππ

# Εργασία - Θέμα 5

## Εκφώνηση

Έστω το αλφάβητο A, B, C, D, E, F, G, H. Σε ένα υποσύνολο φυσικής γλώσσας, τα ονόματα σημείων ορίζονται ως η παράθεση ενός μόνο συμβόλου, τα ονόματα ευθειών ορίζονται ως η παράθεση δύο συμβόλων, τα ονόματα τριγώνων ορίζονται ως η παράθεση τριών συμβόλων, κ.ο.κ, έως και την περίπτωση οκταγώνων. Δεν επιτρέπονται επαναλήψεις συμβόλων. Να γραφεί πρόγραμμα Flex που θα αναλύει προτάσεις της μορφής «τρίγωνο BCD», «τετράγωνο ΒCDA», κ.ο.κ. και θα αποδέχεται μόνο τους σωστούς ορισμούς. Παραδείγματα λάθος ορισμών είναι «τετράγωνο ΑΒ», «τρίγωνο ΑΑD», «γωνία ΒC».

## Τρόπος εκτέλεσης προγράμματος και τρόπος εκκίνησης εκτελέσιμου

Το πρόγραμμα υλοποιήθηκε σε Linux. Απαιτείται να είναι εγκατεστημένος ο λεκτικός αναλυτής flex.

Σε περιβάλλον τερματικού (terminal):

* Εκτέλεση προγράμματος αρχικά παράγεται η έξοδος του λεκτικού αναλυτή με το command **flex Subject5.l** Στη συνέχεια δίνουμε την έξοδο του λεκτικού αναλυτή (lex.yy.c) στον compiler της c ενσωματώνοντας τις απαραίτητες βιβλιοθήκες με το command **gcc lex.yy.c -lfl**
* Εκκίνηση εκτελέσιμου αρχείου (εφόσον το working directory είναι το directory στο οποίο δημιουργήθηκε το εκτελέσιμο με την παραπάνω εντολή) **./a.out** Για τον τερματισμό του προγράμματος πιέζουμε Ctrl + D.

Τέλος το extension του εκτελέσιμου μπορεί και να παραληφθεί επειδή εργαζόμαστε σε περιβάλλον Linux.

## Παράδειγμα εκτέλεσης

Ππππππ

## Περιγραφή αρχείων πηγαίου κώδικα

ππππ

## Αλγοριθμική ανάπτυξη της προγραμματιστικής υλοποίησης

ππππ

# Βιβλιογραφία

*<random>*. (2000-2020). Ανάκτηση από cplusplus: https://www.cplusplus.com/reference/random/

*std::mt19937*. (2000-2020). Ανάκτηση από cplusplus: https://www.cplusplus.com/reference/random/mt19937/

Χρυσαφιάδη, Δ. Κ. (χ.χ.). *Παραδείγματα για Αυτόματα Στοίβας.* Ανάκτηση από gunet2: https://gunet2.cs.unipi.gr/modules/document/file.php/TMB100/%ce%a3%cf%85%ce%bc%cf%80%ce%bb%ce%b7%cf%81%cf%89%ce%bc%ce%b1%cf%84%ce%b9%ce%ba%ce%ad%cf%82%20%ce%94%ce%b9%ce%b1%cf%86%ce%ac%ce%bd%ce%b5%ce%b9%ce%b5%cf%82%202017-2018/%ce%a0%ce%b1%cf%81%ce%b1%ce%b