Μεταγλωττιστές Εργασία 2019-2020

Δημητρίου Δημήτρης Π18036

Άσκηση Β)

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 δημιουργείτε σύμφωνα με τα περιεχόμενα του String s.Στην συνέχεια ορίζουμε τις εξής μεθόδους:

Public void Expression():

Η μέθοδος αυτή πραγματοποιεί αντικατάστασης 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 αντικατάστασης 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 εντοπίσει 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():

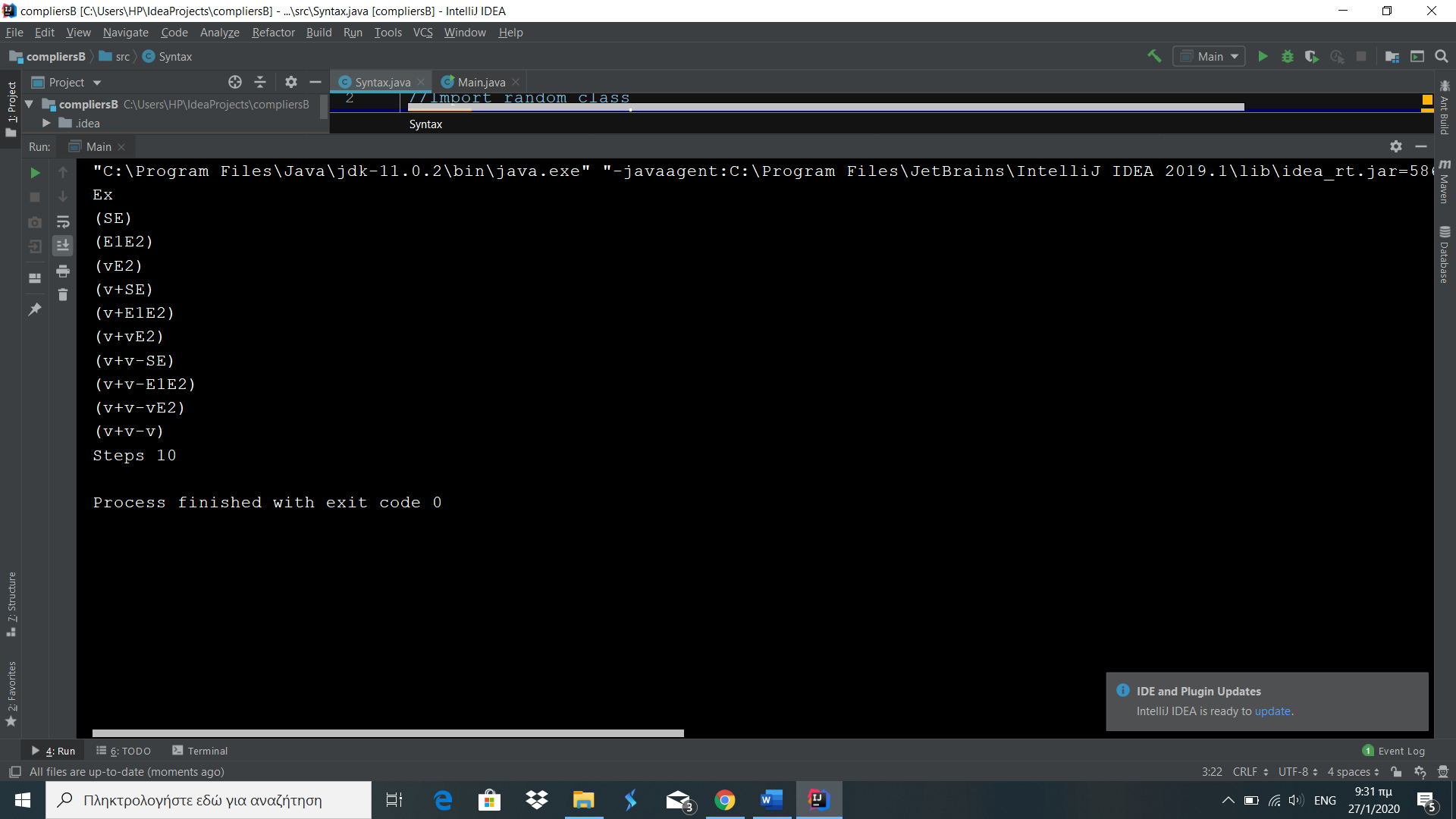
Με βάση την βιβλιοθήκη random παράγουμε έναν τυχαίο αριθμό από το 1-3.Παρόμοια όπως προαναφέραμε ήδη γίνεται αντικατάσταση του E2 σύμφωνα με το index μέσα στο string με βάση την επιλογή του choice δηλαδή του αριθμού που επιλέχθηκε μέσα από την Random.Αν το choice είναι 1 τότε το E2 αντικαθίστανται από 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:

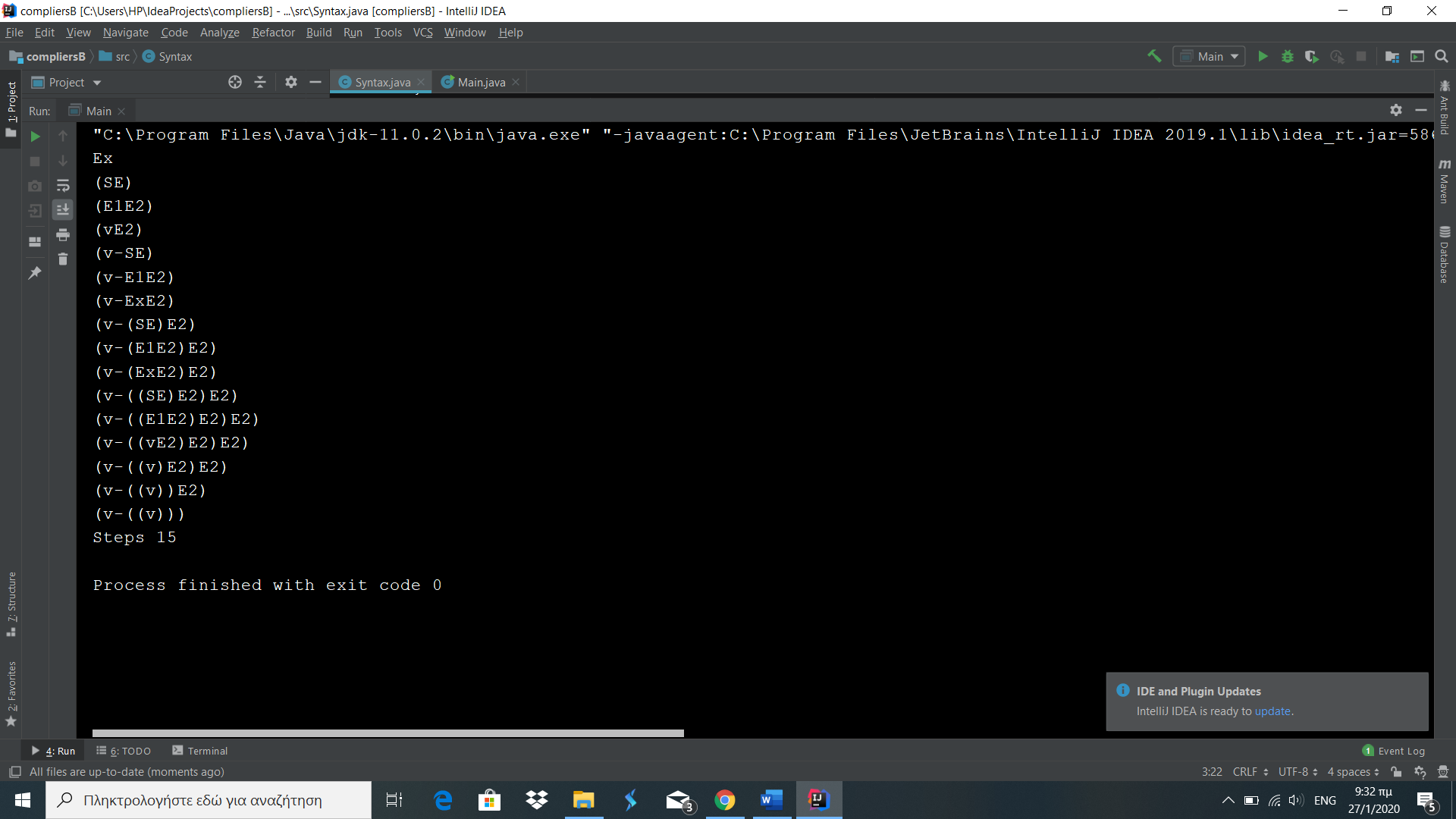
Μέσα στην μέθοδο main,δημιουργούμε ένα object m τύπου Syntax().Επίσης ορίζουμε τις ακέραιες local μεταβλητές count που μετράει τους τερματικούς χαρακτήρες και steps που εκφράζει τον αριθμό των αντικαταστάσεων που γίνονται στο string σύμφωνα με τους κανόνες παραγωγής της γραμματικής. Όσο το z είναι θετικό 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 ‘E’ και ο χαρακτήρας που ακολουθεί στην i+1 θέση είναι ‘x’,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 ‘1’ 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 τέλους 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 συνέχεις αντικαταστάσεις β)Για 15 συνεχείς αντικαταστάσεις γ)Για 71 όπου ολοκληρώνεται λόγω του ορίου 70 που θέσαμε εμείς

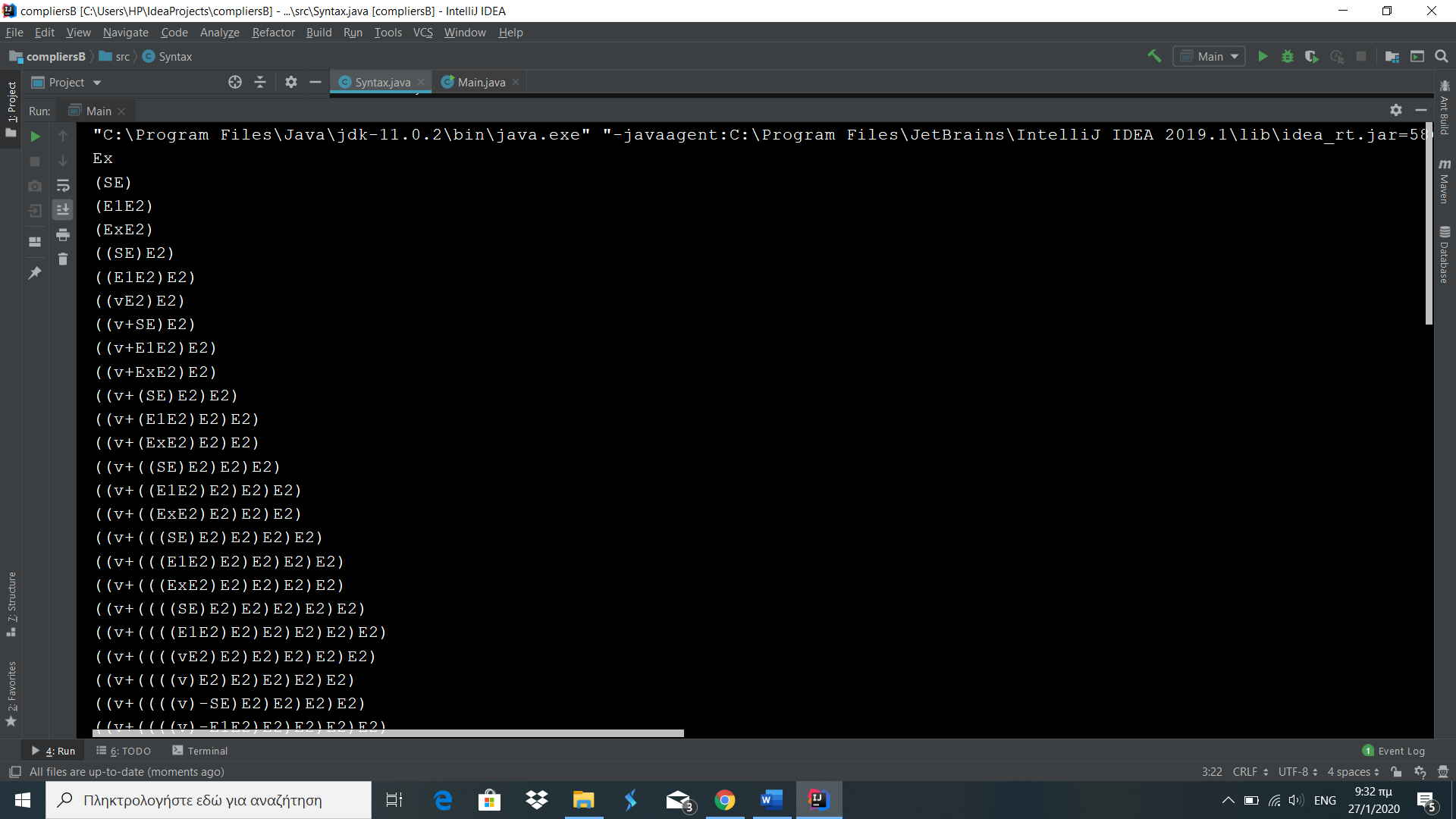
α)

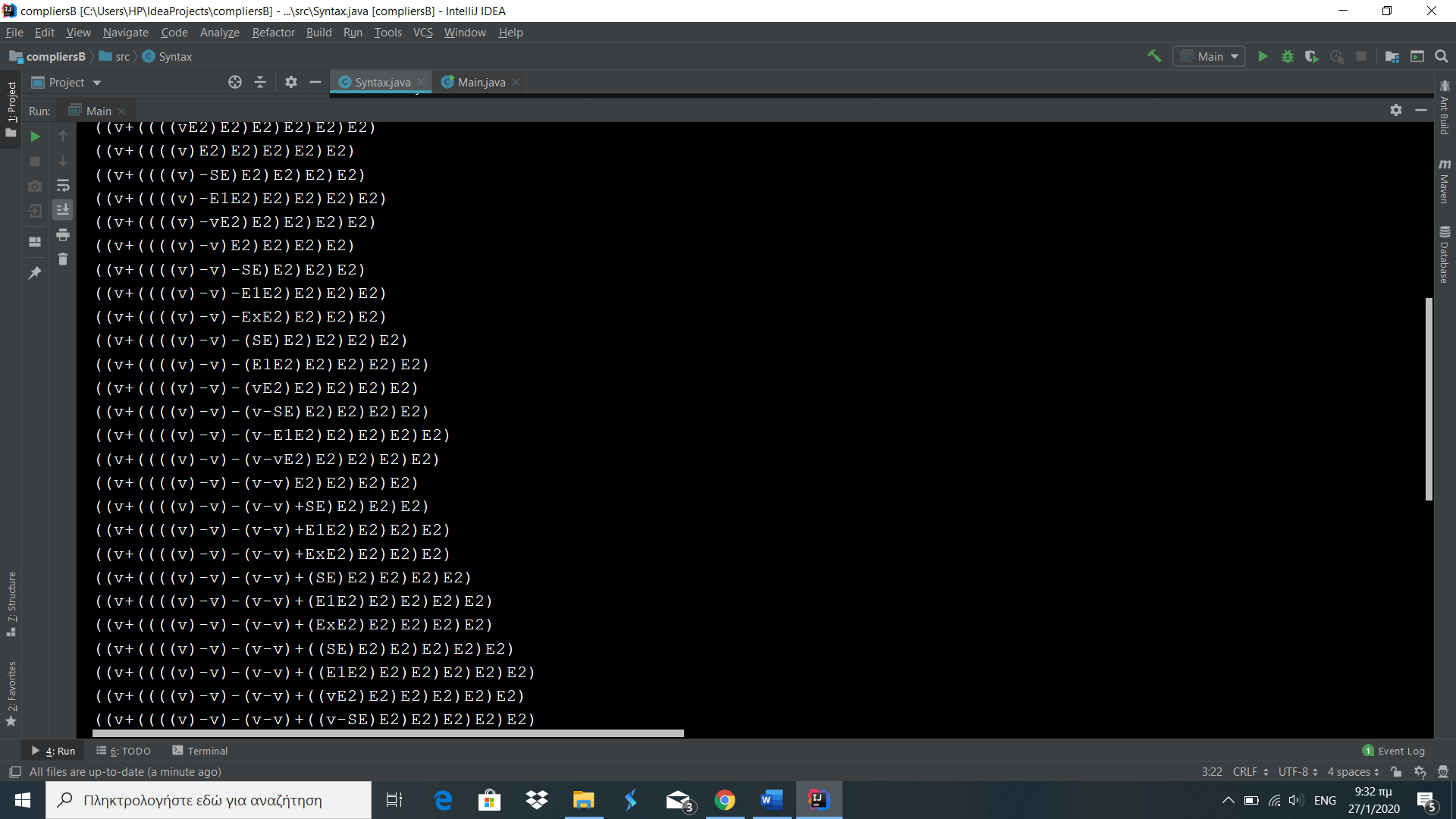


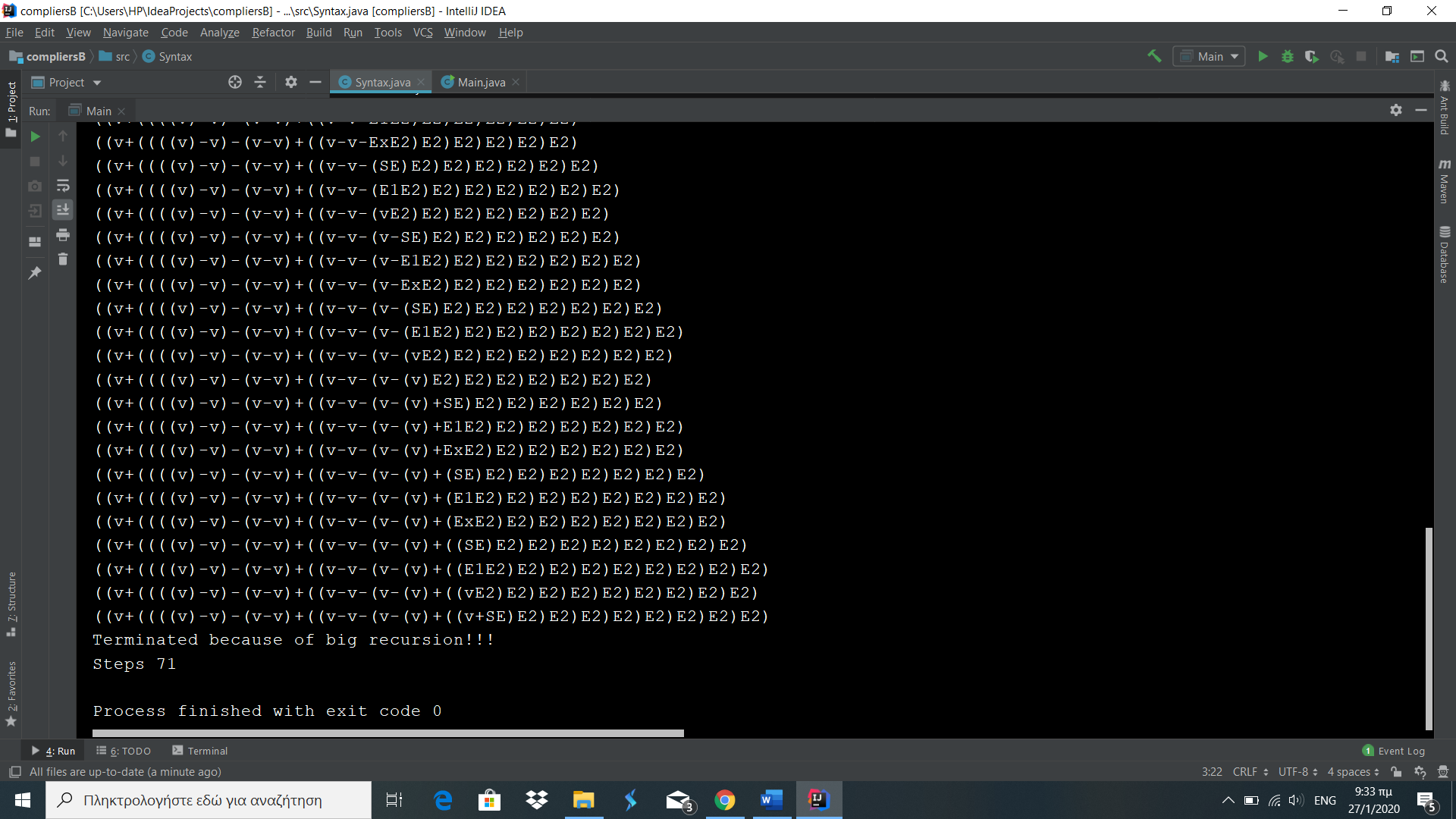
β)



γ)







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

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 η 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 κόμματι που αναγνωρίστηκε και να αυξάνει τον μετρητή part.Αν αλλάξουμε σε νέα γραμμή στην είσοδο ελέγχουμε την τιμή του invalid.Αν είναι 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 κόμματι 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.

Για την μεταγλώττιση 1)flex reg.l 2) Gcc lex.yy.c -lfl 3)./a.out

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

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

ΒΙΒΛΙΟΓΡΑΦΙΑ

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/