Εργασία 1η

Η εργασία 1 μας ζητάει να φτιάξουμε ένα ντετερμινιστικό αυτόματο στοίβας (ΝΑΣ) το οποίο θα αναγνωρίζει εκφράσεις που αποτελούνται από ‘x’ , ‘y’

Περιορισμοί :

- 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’.

Επίσης ζητείται η εκτύπωση των βημάτων που οδήγησαν στην αναγνώριση

Όνομα αρχείου : ASKHSH1.cpp

Λύση

Για να πραγματοποιηθεί το ζητούμενο ΝΑΣ αρχικά δημιουργήσαμε κάποιους κανόνες που μας οδηγούν στην αναγνώριση των δοσμένων συμβολοσειρών :

M=(K,T,V,p,k1,A1,F)
K= {k1,k2}
T= {“x”,”y”,ε}
A1={$}
F={k2}
V={1,0,$}

- p(k1,$,x)={k1,$1}
- p(k1,1,x)={k1,11}
- p(k1,1,y)={k1,ε}
- p(k2,$,ε)={k2,ε}

Όσον αφορά τον κώδικα

Έλεγχος για τον αριθμό των x και y

- **counterx(),countery()** : Αρχικά όσον αφορά τον έλεγχο για τον αριθμό των x και y , δημιουργήσαμε 2 συναρτήσεις με παρόμοια λειτουργία (μια για τα x και μια για τα y), την counterx() και την countery() , οι οποίες παίρνουν σαν όρισμα ένα string s και επιστρέφουν τον αριθμό των x και y ,αυτές καλούνται ύστερα στην main μέσω των μεταβλητών xcount και ycount , οι οποίες μετά αντιστοιχούν στον αριθμό των x και y .

Χρήση **counterx(), countery()** : Οι μεταβλητές xcount και ycount , εφόσον τους εκχωρείται η τιμή που δίνουν αντίστοιχα οι παραπάνω μεταβλητές ελέγχονται παρακάτω στην main.

Πως ελέγχω αν η συμβολοσειρά εν τέλει είναι σωστή σε μορφή(αποτελείται δηλαδή μόνο από x και y , αρχίζει από x και ο αριθμός των x είναι ίσος με τον αριθμό των y)

- Στην main() πριν κληθεί η συνάρτηση που κάνει τον έλεγχο της συμβολοσειράς υπάρχει μια εντολή ελέγχου (if) με συνθήκη ελέγχου :
    - Αριθμός x = αριθμός y (αυτό γίνεται ελέγχοντας αν οι μεταβλητές xcount=ycount) **ΚΑΙ** το πρώτο στοιχείο της συμβολοσειράς δεν ισούται με y **ΚΑΙ** σύνολο στοιχείων της συμβολοσειράς ισούται με το άθροισμα των xcount + ycount
- Αν δεν καλυφθούν οι παραπάνω συνθήκες στην οθόνη , εκτυπώνεται μήνυμα που πληροφορεί τον χρήστη ότι η δοσμένη συμβολοσειρά είναι λάθος

Κύρια συνάρτηση automata(): Η συνάρτηση αυτή δέχεται σαν όρισμα την συμβολοσειρά , πληροφορεί αρχικά τον χρήστη για το ποια είναι οι δοσμένη συμβολοσειρά και :

1. Δημιουργείται ένας vector ο οποίος λειτουργεί σαν στοίβα όπου αν του προσθέτω 8 το αντιμετωπίζω σαν το αρχικό σύμβολο($) , τα υπόλοιπα είναι αντίστοιχα 1
2. Δημιουργείται μια ακόμη συμβολοσειρά (string) που λειτουργεί σαν την τρέχουσα κατάσταση όπου για state=1 => 1=k1(2=k2)
3. Εκτυπώνεται στον χρήστη η συμβολοσειρά που δόθηκε
4. Δημιουργείται μια επαναληπτική διαδικασία μέχρι οι επαναλήψεις να γίνουν κατά ένα περισσότερες από το μέγεθος της δοσμένης συμβολοσειράς

Επαναληπτική διαδικασία

Στην πρώτη επανάληψη θέτουμε την τρέχουσα κατάσταση σαν 1 = k1 και βάζουμε σαν πρώτο σύμβολο της στοίβας το 8 ($). Ύστερα στις επόμενες επαναλήψεις εκτός της τελευταίας με βάση τους παραπάνω κανόνες θα ελέγχετε το τρέχον σύμβολο συμβολοσειράς, η τρέχουσα κατάσταση και το τελευταίο σύμβολο της στοίβας και ανάλογα με αυτά διαγράφω η προσθέτω σύμβολο στην στοίβα , αλλάζω η παραμένω στην κατάσταση που πρέπει και προχωράω στην επόμενη επανάληψη. Στην τελευταία επανάληψη όπου ο μετρητής επαναλήψεων ισούται με το μέγεθος της συμβολοσειράς +1 , ελέγχετε αν το τελευταίο σύμβολο της στοίβας ισούται με 8 ($), η τρέχουσα κατάσταση να είναι 1 (k1) τότε πάλι με βάση τους κανόνες εκτελώ τις ανάλογες μετατροπές και εκτυπώνω στον χρήστη ότι αναγνωρίστηκε η συμβολοσειρά , αλλιώς αν δεν ισχύουν τα παραπάνω για την τελευταία επανάληψη τότε εκτυπώνεται μήνυμα μη αναγνώρισης της συμβολοσειράς. Σε κάθε έλεγχο με βάση τους κανόνες εκτυπώνεται και ανάλογο μήνυμα που πληροφορεί των χρήστη για τα βήματα της αναγνώρισης

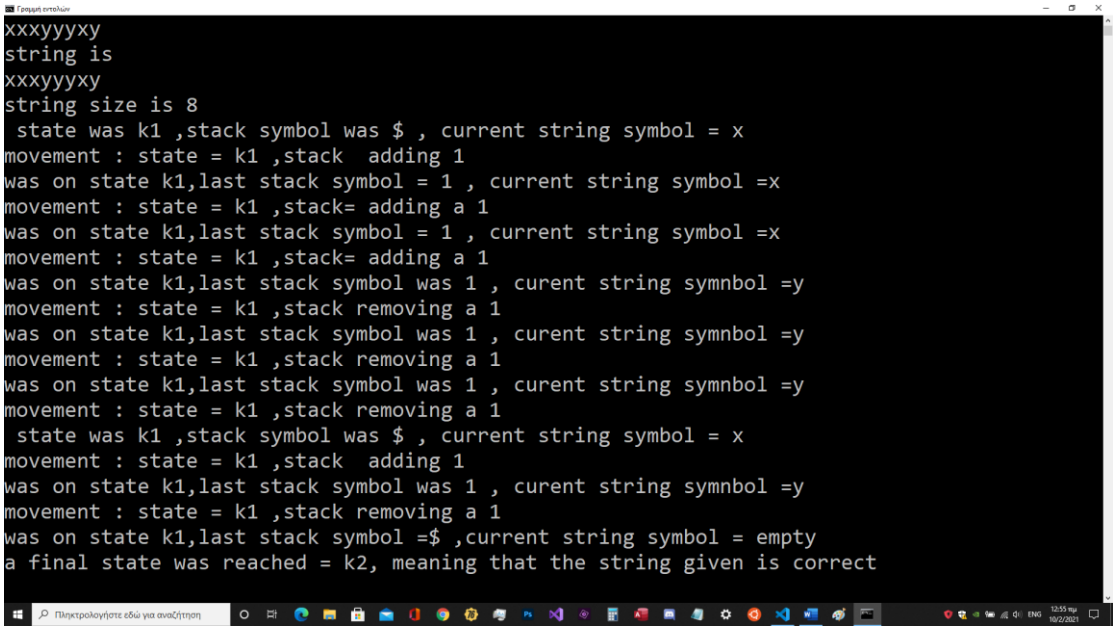
Οδηγίες χρήσης : Για να γίνει χρήση του προγράμματος , αρκεί η μεταγλώττιση του και η εκτέλεση του ανάλογου παραγόμενου εκτελέσιμου, ύστερα μέσω μνήματος που θα τυπωθεί στο τερματικό , ο χρήστης θα κληθεί να δώσει την συμβολοσειρά που θέλει να ελέγξει ως προς την ορθότητα της . Γίνεται ο έλεγχος από το πρόγραμμα και τυπώνονται τα ανάλογα μνήματα.

Μεταγλώττιση σε terminal: g++ -o a ASKHSH1.cpp

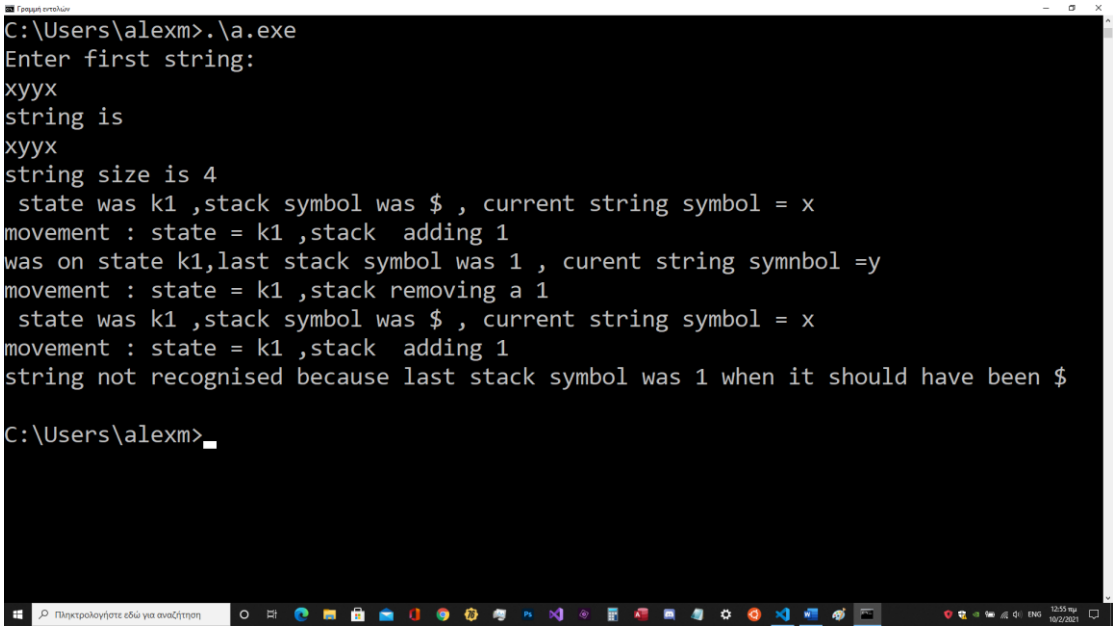
a.exe ή ./a.exe

Παράδειγμα χρήσης :

Σωστή συμβολοσειρά :



Λάθος συμβολοσειρά:



Εργασία 2η

Η εργασία 2η μας ζητάει να φτιάξουμε μία γεννήτρια συμβολοσειρών με βάση την δοσμένη γραμματική.

<E>::=(<Y>)
<Y>::=<A><B>
<A>::=v|<E>
<B>::=-<Y>|+<Y>|ε

Η διαδικασία να τερματίζεται , και τα βήματα των παραγωγών να τυπώνονται

Όνομα αρχείου : ASKHSH2.cpp

Όσων αφορά τον κώδικα

Συνάρτηση search(): Η λειτουργία της συνάρτησης search συνοψίζεται στην αναζήτηση μέσω επαναληπτικής διαδικασίας που επαναλαμβάνεται όσο οι επαναλήψεις είναι μικρότερες από το μέγεθος της τρέχουσας συμβολοσειράς ή μέχρι να βρεθεί το ανάλογο τερματικό σύμβολο μέσα σε αυτήν (η συμβολοσειρά συνεχίζει και παράγεται και η συνάρτηση καλείται για κάθε νέα μετατροπή της συμβολοσειράς μέσα από την συνάρτηση symbo , που θα αναλυθεί παρακάτω).Η συνάρτηση δέχεται σαν όρισμα έναν χαρακτήρα που αντιπροσωπεύει ένα μη τερματικό σύμβολο που αναζητείται μέσα στην συμβολοσειρά , και την τρέχουσα συμβολοσειρά . Επιστρέφει τον αριθμό των επαναλήψεων που εκτελέστηκαν μέχρι να βρεθεί μη τερματικό σύμβολο που ουσιαστικά είναι και ο αριθμός που δηλώνει την θέση του μη τερματικού μέσα στην συμβολοσειρά.

Συνάρτηση search(): Η συνάρτηση αυτή αποτελεί την κύρια συνάρτηση του προγράμματος , δέχεται σαν όρισμα μια συμβολοσειρά , (η οποία στην main αρχικοποιείται σαν "f") αρχικά η συμβολοσειρά θα ισούται με Ε και δημιουργούνται 2 μεταβλητές , μια για έλεγχο ύπαρξης μη τερματικού συμβόλου (temp όπου θα αντιστοιχίζεται με τον αριθμό σημείου συμβολοσειράς που βρίσκεται το μη τερματικό ,καλώντας την search μέσω αυτής της μεταβλητής μετά από κάθε μετατροπή), και μια Boolean (flag) για έλεγχο στην επαναληπτική διαδικασία, που παράγει και μετασχηματίζει σύμβολα , για το αν η συμβολοσειρά αποτελείται μόνο από τερματικά. Ύστερα με βάση τους κανόνες και χρήση τυχαίων παραγόμενων αριθμών και της συνάρτησης search βρίσκουμε σε ποιο σημείο υπάρχει μη τερματικό σύμβολο , το μετατρέπουμε στο ανάλογο (αν έχουμε πάνω από 1 επιλογή τότε επιλέγεται στην τύχη μία από όλες) και συνεχίζουμε τους ελέγχους

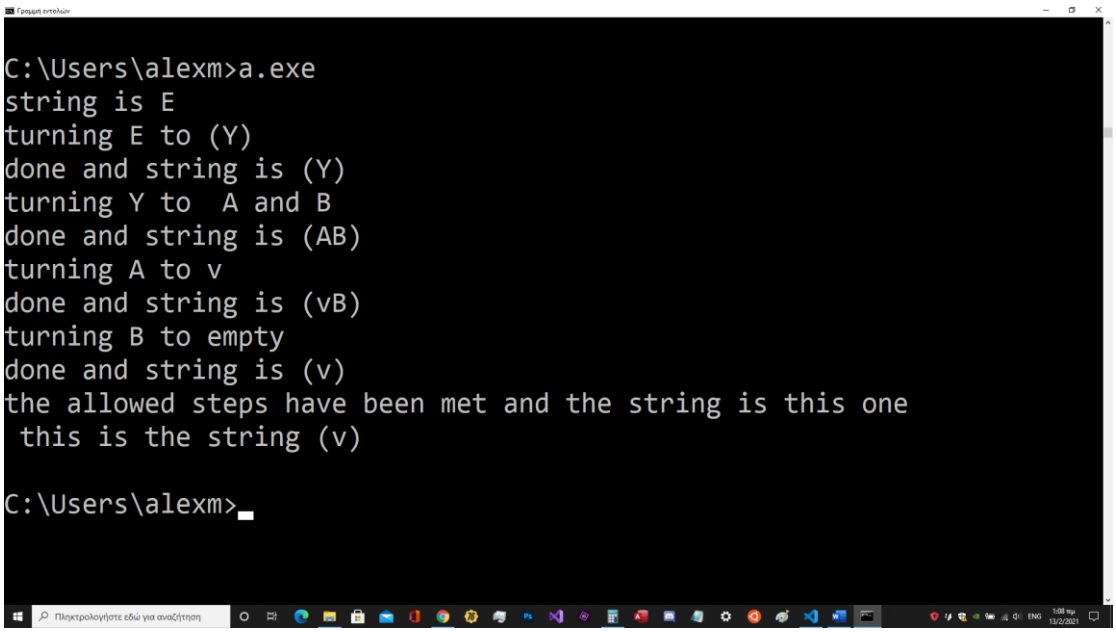
- Η μετατροπή γίνεται βρίσκοντας το σημείο του μη τερματικού συμβόλου , χωρίζουμε την τρέχουσα συμβολοσειρά σε 2 και διαγράφουμε και αντικαθιστούμε ανάλογα το τελευταίο σύμβολο (επιθυμητό για αλλαγή) της πρώτης από τις 2 συμβολοσειρές , ύστερα της ξαναενώνουμε και τυπώνουμε το ανάλογο μήνυμα
- Όταν βρεθεί σύμβολο Β τότε η συνάρτηση ευνοεί την παραγωγή κενών (δηλαδή απλά την διαγραφή του συμβόλου Β) ώστε να υπάρξει όσο γίνεται πιο σύντομη παραγωγή συμβολοσειράς
- Τέλος όταν μέσω των ελέγχων δεν βρεθεί άλλο μη τερματικό σύμβολο στην συμβολοσειρά τότε εκτυπώνεται η συμβολοσειρά στην τελική της μορφή μαζί με κατάλληλο μήνυμα και η συνάρτηση έχει ολοκληρώσει την εκτέλεση της

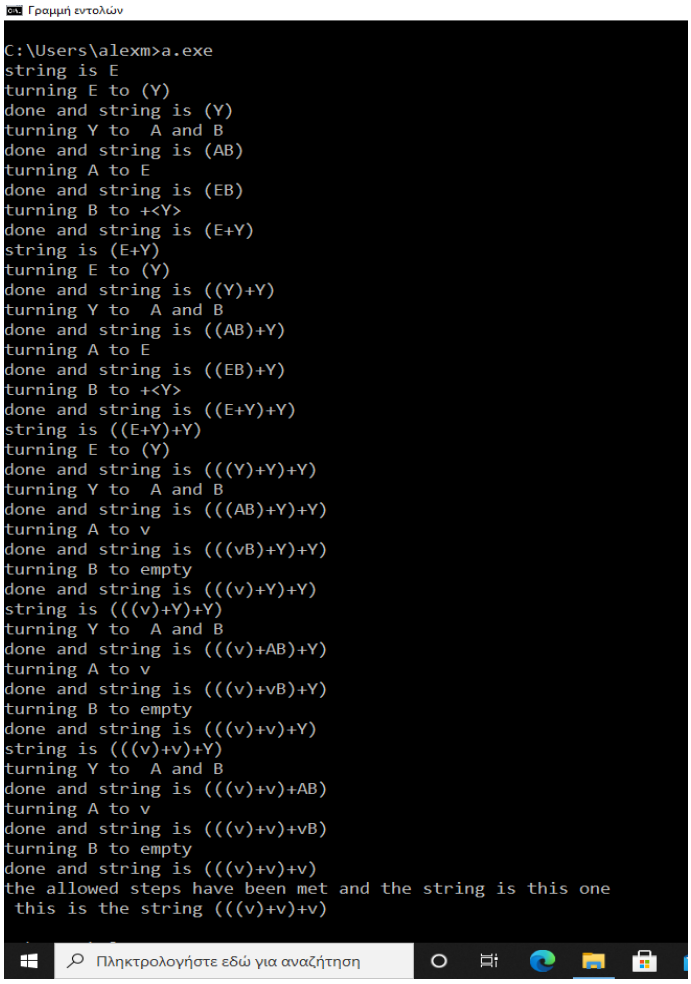
main(): Στην main απλά αρχικοποιείται μια συμβολοσειρά σαν "f" και ύστερα καλείται η search() με όρισμα αυτήν την συμβολοσειρά

Οδηγίες χρήσης: Η χρήση του προγράμματος συνοψίζεται στην μεταγλώττιση του καθώς και την εκτέλεση του . Οπότε αρκεί η εκτέλεση του ανάλογου εκτελέσιμου

Μεταγλώττιση σε terminal: g++ -o a ASKHSH2.cpp
a.exe ή ./a.exe

Παραδείγματα λειτουργίας :





Εργασία 3η

Αρχικά μας ζητήθηκε να εξεταστεί αν η δοσμένη γραμματική ήταν LL(1) και να κατασκευάσουμε τον ανάλογο αναλυτή top-down που αναγνωρίζει μια δοσμένη συμβολοσειρά ή απαντά αρνητικά ως προς την ορθότητα της

ο έλεγχος αυτός καθώς και η εξήγηση της δημιουργίας του top-down αναλυτή θα γίνει παρακάτω παραθέτοντας τα σχετικά σύνολα FIRST , FOLLOW ,LOOKAHEAD καθώς και τον συντακτικό πίνακα της γραμματικής

Όνομα αρχείου: ASKHSH3.cpp

FIRST

FIRST(S)= { (}

FIRST(X) = { a,b,(}

FIRST(Y)={ a,b,(}

FIRST(Z)={ *,-,+,ε }

FOLLOW

FOLOW(S)={ $,*,-,+,) }

FOLLOW(X)={) }

FOLLOW(Y)={ *,-,+,) }

FOLLOW(Z)={) }

LOOKAHEAD

LOOKAHEAD(S->(X))={ (}

LOOKAHEAD(X->YZ)={ a,b,(}

LOOKAHEAD(Y-> a)={ a }

LOOKAHEAD(Y->b)={ b }

LOOKAHEAD(Y->S)={ (}

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		

Εν τέλει η γραμματική αυτή είναι LL(1) μιας και για κάθε ζεύγος διαφορέτικών παραγωγών (π.χ Y->a και Y->b) που έχουν ίδιο αριστερό μέλος ισχύει ότι :

Π.χ LOOKAHEAD(Y->a) ∩ LOOKAHEAD(Y->b) = Ø

Όσων αφορά τον κώδικα

Όσων αφορά την ορθότητα της της συμβολοσειράς ο έλεγχος γίνεται μέσω 6 συναρτήσεων που μετράνε η κάθε μια την ποσότητα των (,),α,b,-,+ και την επιστρέφουν αντίστοιχα . Η κλήση τους γίνεται στο κομμάτι της εισαγωγής της συμβολοσειράς από τον χρήστη ,ειδικότερα μέσω 6 πάλι μεταβλητών που αντιστοιχουν ο καθένας σε ένα τερματικό σύμβολο και ειδικότερα στον αριθμό που επιστρέφουν οι παραπάνω συναρτήσεις ελέγχεται πρίν κληθεί η βασική συνάρτηση του προγράμματος αν το μέγεθος της συμβολοσειράς χωρίς το σύμβολο $ είναι ίσο με το άθροισμα των παραπάνω μεταβλητών

Κωδικοποίηση τερματικών συμβόλων σε vector

Εφόσων δοθεί η συμβολοσειρά(πριν από τα παρακάτω δημιουργείται μια νέα συμβολοσειρά που είναι η αρχική συν το σύβολο $ στο τέλος της) το πρόγραμμα εισάγει έναν αριθμό αντίστοιχο με το ανάλογο τερματικό σύμβολο μέσα σε ένα πρώτο vector μέχρι να υπάρχουν όσοι αριθμοί όσο και σύμβολα στην συμβολοσειρα , ειδικότερα όσων αφορά το πρόγραμμα τα τερματικά και μη συμβολα , που χρησιμοποιούνται από τις ανάλογες στοίβες (τα vectors) η κωδικοποίηση τους έχει ως εξής

$=0

S=1

X=2

Y=3

Z=4

(=5

)=6

*=7

- = 8

+ = 9

a=13

b=14

Συνεχίζοντας ,το vector αυτό στο οποίο μπήκαν οι αριθμοί για μη τερματικά σύμβολα χρησιμοποιείται από την βασική συνάρτηση του προγράμματος για προσδιορισμό των παραγωγών .

Βασική συνάρτηση automata()

Η συνάρτηση αυτή που αποτελεί και το κύριο μέρος του προγράμματος μας , δέχεται σαν όρισμα την συμβολοσειρά που καταλήγει σε $ καθώς και το vector που περιέχει αριθμούς για τερματικά σύμβολα. Αρχικά δημιουργείται ένα ακόμη vector. Ύστερα ακολουθεί επαναληπτική διαδικασία στην οποία στο πρώτο της βήμα στο νεοδημιούργητο vector προστίθεται ο αριθμο 0 και μετά ο 1 , μετά από αυτό το βήμα αρχίζουν οι διάφοροι έλεγχοι .

- Ελέγχεται αν υπάρχουν τερματικό σύμβολο στο τέλος του νεοδημιούργητου vector, αν ναι τότε ελέγχεται ο vector τερματικών συμβόλων ώστε να διαπιστωθεί πιο τερματικό σύμβολο έχουν ίδιο(τελευταίο για τον νεοδημιούργητο , ανάλογα με την επανάληψη για των τερματικών συμβόλων) , ανάλογα με το σύμβολο γίνεται διαγραφή από τον νεοδημιούργητο vector του τελευταίου του συμβόλου εκτυπώνονται τα κατάλληλα μυνήματα και προστίθεται +1 στον μετρητή επαναλήψεων

- Ελέγχεται αν το τελευταίο σύμβολο του νεοδημιούργητου vector είναι μη τερματικό , αναλογα με το ποιο είναι αυτό το σύμβολο ΚΑΙ ανάλογα με τον μετρητή επανάληψης που προσδιορίζει το σύμβολο με το οποίο ισούται (από το vector των τερματικών) εκτυπώνεται κατάλληλο μήνυμα και ανάλογα την παραγωγή π.χ έιχαμε S στον παραπάνω vector , τότε διαγράφεται αυτό το σύμβολο και η παραγωγή του εισάγεται στον ίδιο αλλά με ανάποδη σειρά και γίνεται συνέχιση της συνάρτησης χωρίς προσθήκη στον μετρητή επαναλήψεων , αν δε το τρέχων σύμβολο σε συνάρτηση με το σύμβολο μη τερματικών δεν αντιστοιχούν σε κάποια παραγωγή τότε έχουμε λάθως συμβολοσειρα

- Ελέγχεται αν το τελευταίο σύμβολο του νεοδημιούργητου vector ισούται με το σύμβολο $ στο σημείο του vector των τερματικών συμβόλων που αντιστοιχεί με τον μετρητή επαναλήψεων , ειδικότερα μας ενδιαφέρει αυτός ο έλεγχος όταν βρισκόμαστε στην τελευταία επανάληψη καθώς αν ισχύουν τα παραπάνω τότε η συμβολοσειρά είναι σωστή

Εν τέλει στην main δίνεται μια παραδειγματική χρήση των παραπάνω μέσω της συμβολοσειράς ((b-a)*(a+b)) και ύστερα δίνεται η δυνατότητα στον χρήστη να εισάγει την δικία του για έλεγχο

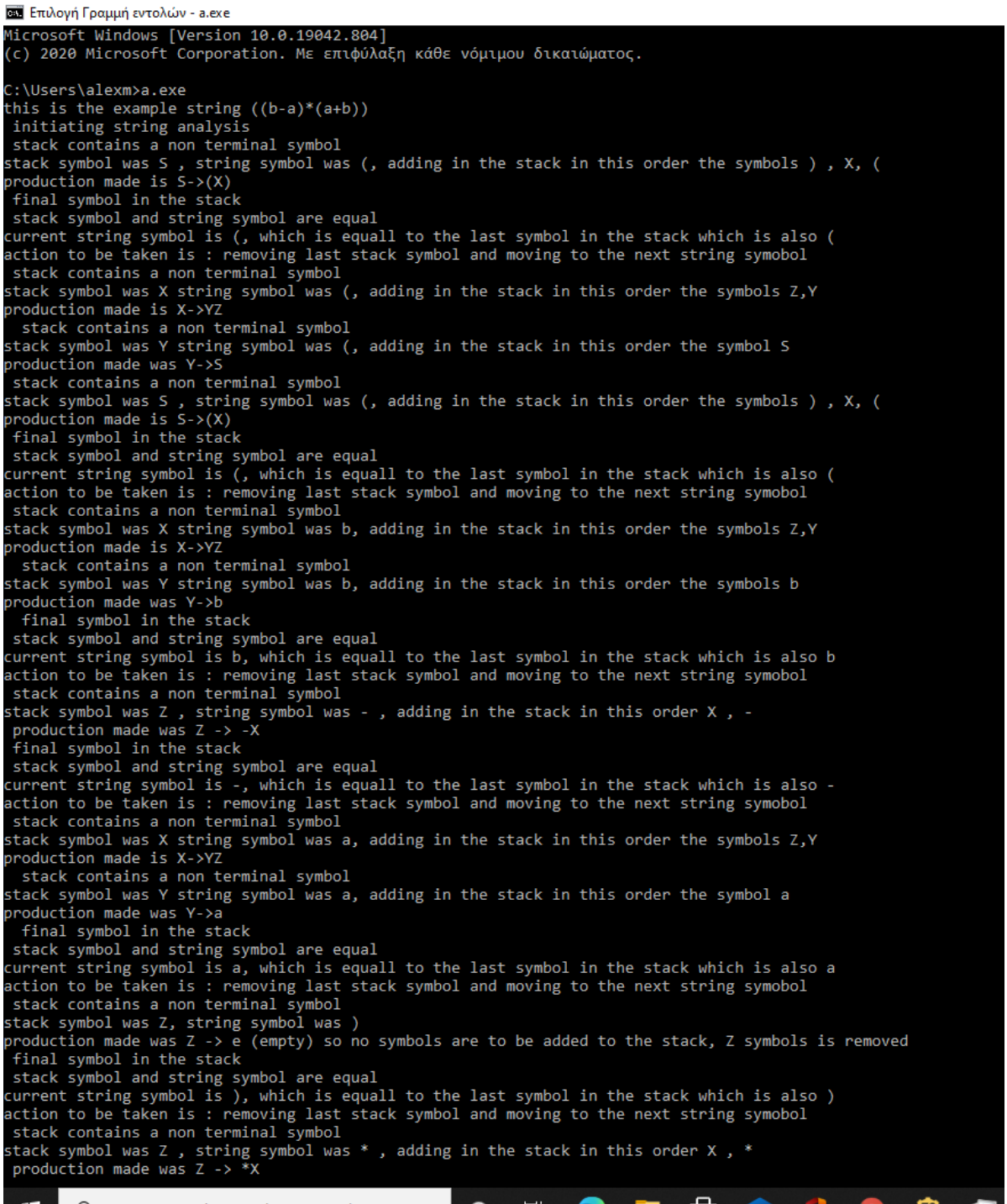
Οδηγίες χρήσης

Αρκεί η μεταγλώτιση του προγράμματος και η εκτέλεση του ανάλογου εκτελέσιμου καθώς το πρόγραμμα θα ξεκινήσει και θα δώσει το παράδειγμα ανάλυσης της συμβολοσειράς ((b-a)*(a+b)) και ύστερα θα δωθεί η δυνατότητα στον χρήστη να εισάγει όποια συμβολοσειρά θέλει να αναλύσει

Μεταγλώττιση σε terminal: g++ -o a ASKHSH3.cpp

a.exe ή ./a.exe

Παράδειγμα χρήσης



Επιλογή Γραμμή εντολών - a.exe

```
 final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is *, which is equall to the last symbol in the stack which is also *
action to be taken is : removing last stack symbol and moving to the next string symobol
 stack contains a non terminal symbol
stack symbol was X string symbol was (, adding in the stack in this order the symbols Z,Y
production made is X->YZ
  stack contains a non terminal symbol
stack symbol was Y string symbol was (, adding in the stack in this order the symbol S
production made was Y->S
 stack contains a non terminal symbol
stack symbol was S , string symbol was (, adding in the stack in this order the symbols ) , X, (
production made is S->(X)
 final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is (, which is equall to the last symbol in the stack which is also (
action to be taken is : removing last stack symbol and moving to the next string symobol
 stack contains a non terminal symbol
stack symbol was X string symbol was a, adding in the stack in this order the symbols Z,Y
production made is X->YZ
  stack contains a non terminal symbol
stack symbol was Y string symbol was a, adding in the stack in this order the symbol a
production made was Y->a
  final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is a, which is equall to the last symbol in the stack which is also a
action to be taken is : removing last stack symbol and moving to the next string symobol
 stack contains a non terminal symbol
stack symbol was Z , string symbol was + , adding in the stack in this order X , +
 production made was Z -> +X
 final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is + , which is equall to the last symbol in the stack which is also +
action to be taken is : removing last stack symbol and moving to the next string symobol
 stack contains a non terminal symbol
stack symbol was X string symbol was b, adding in the stack in this order the symbols Z,Y
production made is X->YZ
  stack contains a non terminal symbol
stack symbol was Y string symbol was b, adding in the stack in this order the symbols b
production made was Y->b
  final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is b, which is equall to the last symbol in the stack which is also b
action to be taken is : removing last stack symbol and moving to the next string symobol
 stack contains a non terminal symbol
stack symbol was Z, string symbol was )
production made was Z -> e (empty) so no symbols are to be added to the stack, Z symbols is removed
 final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is ), which is equall to the last symbol in the stack which is also )
action to be taken is : removing last stack symbol and moving to the next string symobol
 stack contains a non terminal symbol
stack symbol was Z, string symbol was )
production made was Z -> e (empty) so no symbols are to be added to the stack, Z symbols is removed
 final symbol in the stack
 stack symbol and string symbol are equal
current string symbol is ), which is equall to the last symbol in the stack which is also )
action to be taken is : removing last stack symbol and moving to the next string symobol
the string has been analyzed and it looks like  the stack ended with the $ symbol , and also the inputed
this was the example string being analyzed we will continue by asking you to input your own string
 Enter the string you wish to analyze
```

Σημείωση : Το πρόγραμμα μας δεν έχει δυνατότητα να εκτυπώσει το σχετικό δέντρο , παρόλα αυτά είναι ένας απολύτως ορθός και λειτουργικός συντακτικός αναλυτής top-down

Εργασία 4η

Στην εργασία 4 ζητήθηκε η δημιουργία σχετικής EBNF και BNF καθώς και του ανάλογου προγράμματος flex που αναγνωρίζει τις κανονικές εκφράσεις τύπου "x=3+a; " που περιγράφονται από τα παρακάτω βήματα :

- Στην αρχή πρέπει να εμφανίζεται το όνομα μιας μεταβλητής.
- Στη συνέχεια, θα πρέπει να ακολουθεί το σύμβολο "=".
- Μετά ακολουθεί το όνομα μιας μεταβλητής ή ένας αριθμός από 1 έως και 9, ύστερα ένα σύμβολο από τα εξής "+", "-", "*", "/", "%" και έπειτα ξανά ακολουθεί το όνομα μιας μεταβλητής ή ένας αριθμός από 1 έως και 9.
- Το προηγούμενο βήμα μπορεί να επαναληφθεί όσες φορές επιθυμούμε.
- Η κανονική έκφραση τελειώνει με τον χαρακτήρα ";"

Όνομα αρχείου : flex4.l

Παρακάτω δίνονται BNF,EBNF και ανάλογος συντακτικός πίνακας :

BNF

<CHR>::= a|b|…|y|z

<NUM>::=1|2|3|4|5|6|7|8|9

<OP>::= “*”|”/”|”+”|”-“|”%”

<EQUALS>::”=”

<SEMICOLON>:: ”;”

<KANA>::<NUM>|<CHR>

<EPANAL>::=<OP><KANA>

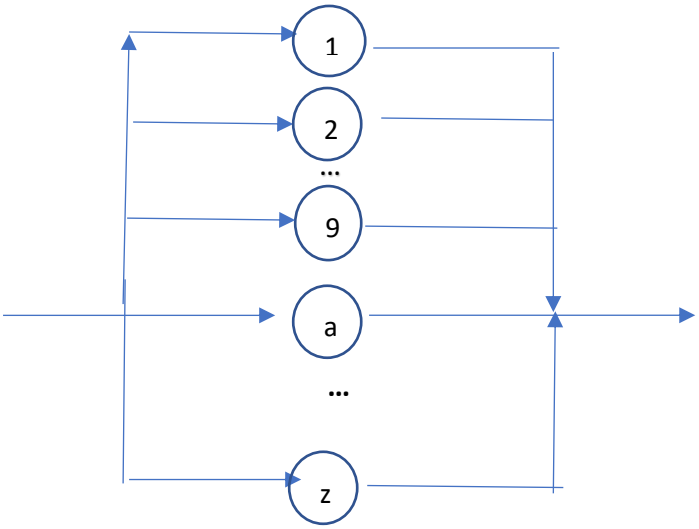
<EXPRESSION>::<KANA><EQUALS><EPANAL><SEMICOLON>

EBNF

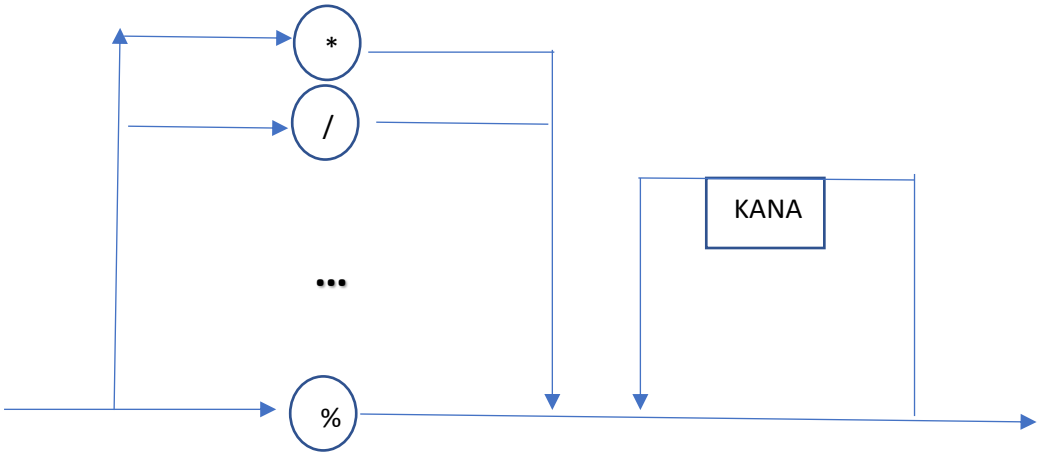
<EXPRESSION>::<KANA><EQUALS>{<OP><KANA>}<SEMICOLON>

Συντακτικό διάγραμμα

KANA::= 1|2|3|4|5|6|7|8|9|[a-z]



EPANAL::= “*”|”/”|”+”|”-“|”%” {KANA}



Όσων αφορά τον κώδικα

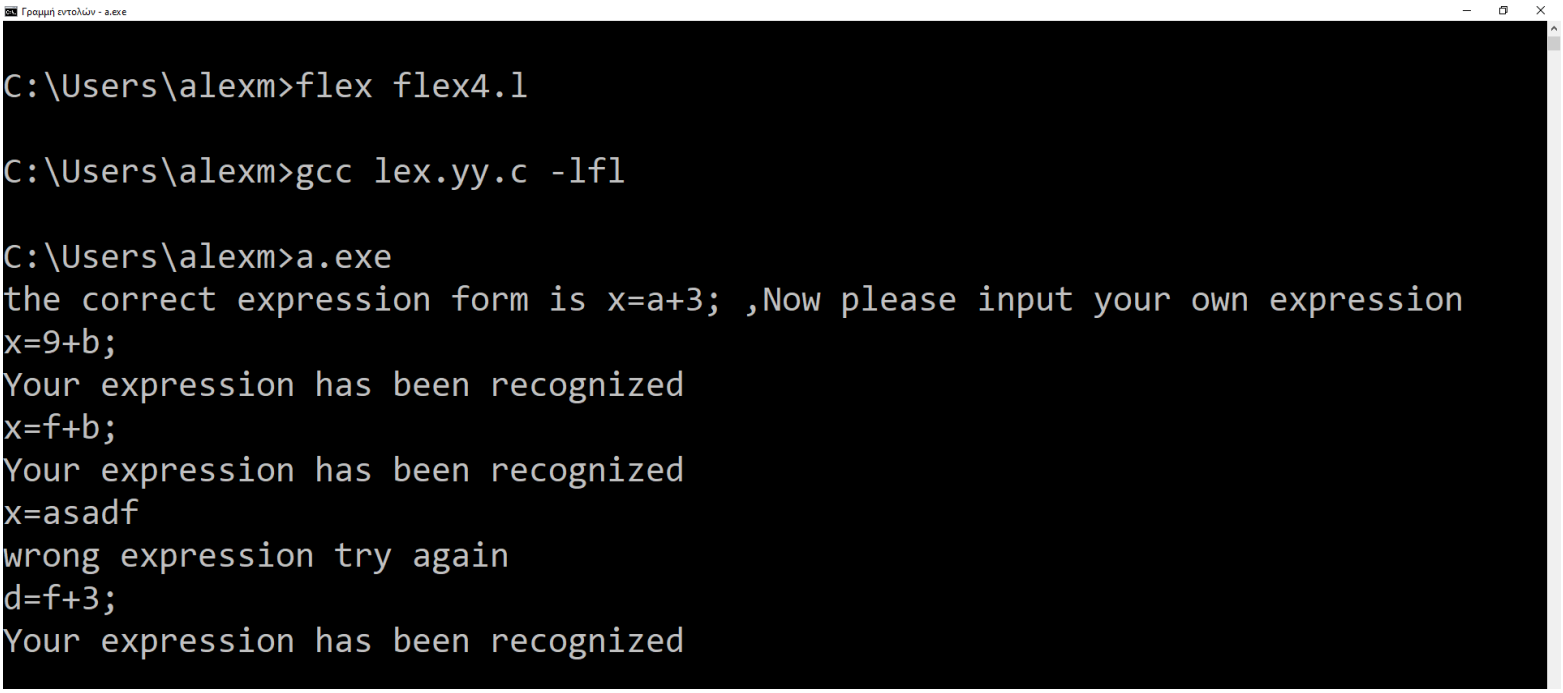
Όσων αφορά των κώδικα , στο τμήμα δηλώσεων δίνονται τα στοιχεία που αποτελούν τους κανόνες καθώς και οι ίδιοι κανόνες. Δίνεται ύστερα η έκφραση στην μορφή που πρέπει να πάρει και ρυθμίζοντας την δυνατότητα του κανόνα EPANAL να έχει επαναληπτικό χαρακτήρα. Όταν εισαχθεί κάποια έκφραση το πρόγραμμα θα κάνει την αναγνώριση της και είτε θα τυπώσει μήνυμα αναγνώρισης , είτε θα τυπώσει μήνυμα μη αναγνώρισης

Οδηγίες χρήσης

Η χρήση του προγράμματος συνοψίζεται στην μεταγλώττιση του και την εκτέλεση του ανάλογου εκτελέσιμου , ύστερα ακολουθεί εκτύπωση μνήματος που πληροφορεί τον χρήστη για την μορφή της έκφρασης που μπορεί να εισάγει προς αναγνώριση.

Μεταγλώττιση σε terminal (windows OS): flex flex4.l
gcc lex.yy.c -lfl
a.exe ή ./a.exe

Παράδειγμα χρήσης



Εργασία 5η

Η άσκηση 5η μας ζητούσε να φτιάξουμε ένα πρόγραμμα flex το οποίο με βάση το αλφάβητο A,B,D,E,F,G,H αναγνωρίζει η δεν αναγνωρίζει εκφράσεις που αντιστοιχούν σε σχήματα ,Π.χ. point A, triangle ABC . Δεν επιτρέπονται οι επαναλήψεις συμβόλων καθώς και λάθος προτάσεις όπως triangle ABCD

Όνομα αρχείου: flex5.l

Όσων αφορά τον κώδικα

Όσων αφορά των κώδικα , στο τμήμα δηλώσεων δίνεται μια έκφραση που αποτελείται από τις παραπάνω ζητούμενες προτάσεις με χρήση επιλογής των ανάλογων χαρακτήρων από τους [A-H] .Ύστερα η όταν εισαχθεί κάποια έκφραση το πρόγραμμα θα κάνει την αναγνώριση της και είτε θα τυπώσει μήνυμα αναγνώρισης , είτε θα τυπώσει μήνυμα μη αναγνώρισης.

Όσων αφορά τον έλεγχο του σωστού αριθμού σημείων στα σχήματα ο έλεγχος γίνεται μέσω της χρήσης επιλογής του ανάλογου αριθμού χαρακτήρων.

Ο έλεγχος για την μοναδικότητα του κάθε χαρακτήρα στα σημεία των σχημάτων , δεν πραγματοποιήθηκε.

Οδηγίες χρήσης

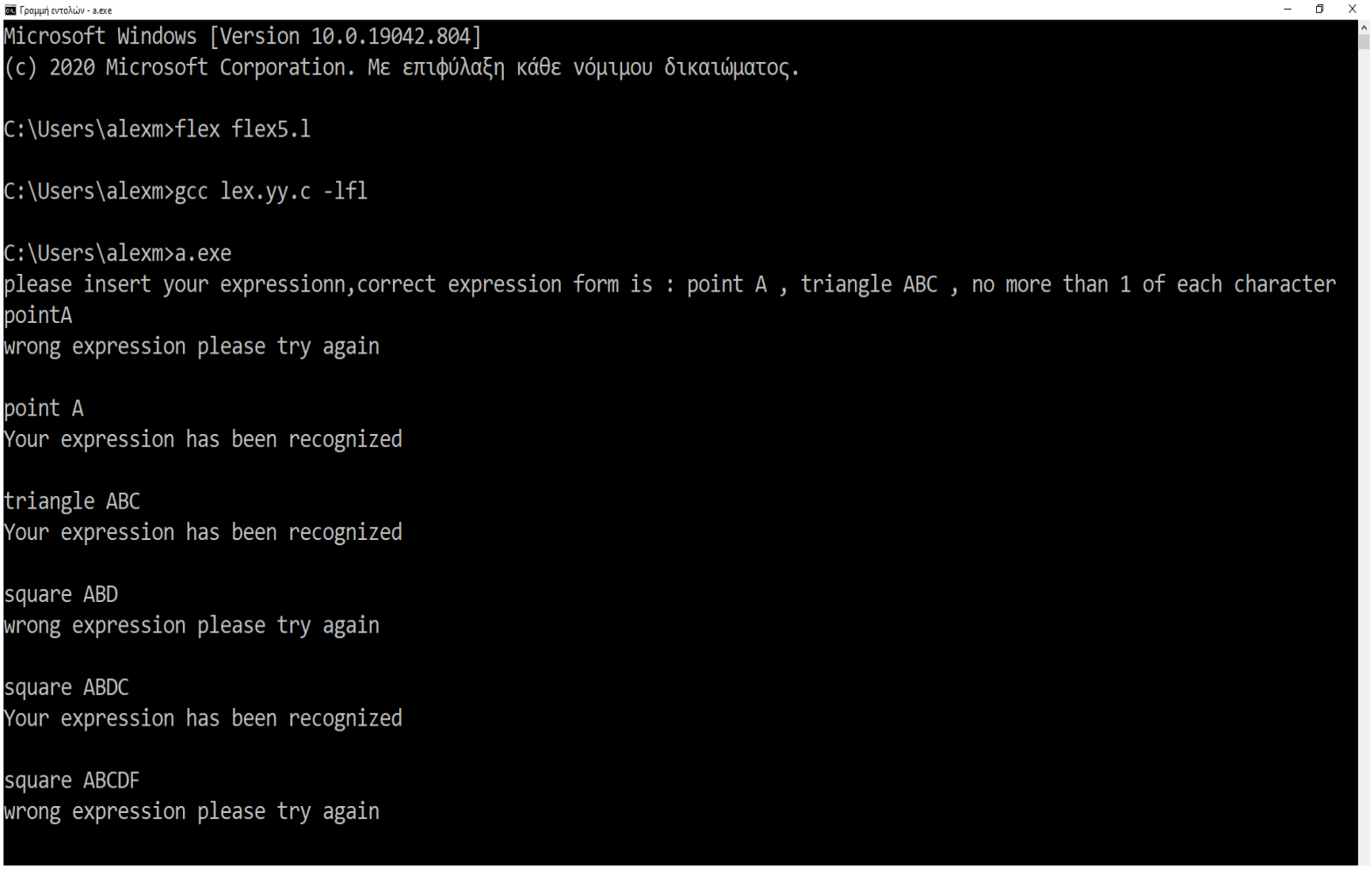
Μετά την μεταγλώττιση και εκτέλεση του ανάλογου εκτελέσιμου αρχείου ο χρήστης καλείται να εισάγει μια πρόταση/έκφραση με τον ανάλογο τύπο . Όσων αφορά την μορφή της έκφρασης , μια “σωστή” έκφραση έχει την παρακάτω μορφή:

σχήμα(κενό)σημεία σχήματος

σχήμα : point/line/triangle/square/pentagon/hexagon/heptagon/octagon

Μεταγλώττιση σε terminal (windows OS): flex flex5.l
gcc lex.yy.c -lfl
a.exe ή ./a.exe

Παραδείγματα χρήσης



Συντελεστές εργασίας

Τάσος Τσότρας Π19176

Αλέξανδρος Μανάκος Π19093

Σταύρος Ανδρομιδάς Π19011

Μιχάλης Φλυτζάνης Π19181