



**ΠΑΝΕΠΙΣΤΗΜΙΟ  
ΠΕΙΡΑΙΩΣ**

## **ΕΡΓΑΣΙΑ ΜΕΤΑΓΛΩΤΙΣΤΕΣ 2018-2019**

Φοιτητές: Μίμογλου Γεώργιος - Π17073  
Ντόστης Χρήστος - Π16189  
Παπαθεωχάρους Ιωάννης - Π17002

# Περιεχόμενα

- Εργασία Α.
  - Η λογική του προγράμματος.
  - Screenshots.
- Εργασία Β.
  - Η λογική του προγράμματος.
  - Screenshots.
- Εργασία Γ.
  - Κανόνες Παραγωγής.
  - Σύνολα FIRST.
  - Σύνολα FOLLOW.
  - Σύνολα EMPTY.
  - Σύνολα LOOKAHEAD.
  - Απόδειξη  $LL(1)$ .
  - Συντακτικός πίνακας.
  - Επίδειξη.
  - Η λογική του προγράμματος.
  - Screenshots.
- Εργασία Δ.
  - Συντακτικό διάγραμμα.
  - EBNF σύνταξη.
  - Το πρόγραμμα Flex.
  - Screenshots.
- Εργασία Ε.
  - Το πρόγραμμα Flex.
  - Screenshots.

# Εργασία Α

Η πρώτη εργασία είχε ως στόχο την κατασκευή προγράμματος που ελέγχει τη συντακτική ορθότητα των παρενθέσεων ενός αρχείου κειμένου. Επιλέχθηκε η γλώσσα προγραμματισμού C++ για την υλοποίηση.

Παρακάτω ακολουθεί περεταίρω ανάλυση και επίδειξη screenshots.

# Η λογική του προγράμματος

Για τις ανάγκες του προγράμματος, δημιουργήθηκε η συνάρτηση `bool stackAutomata`, η οποία δέχεται ως είσοδο τις παραμέτρους:

`string &state`: Η κατάσταση.

`char input`: Το σύμβολο εισόδου.

`stack<char> &myStack`: Η στοίβα.

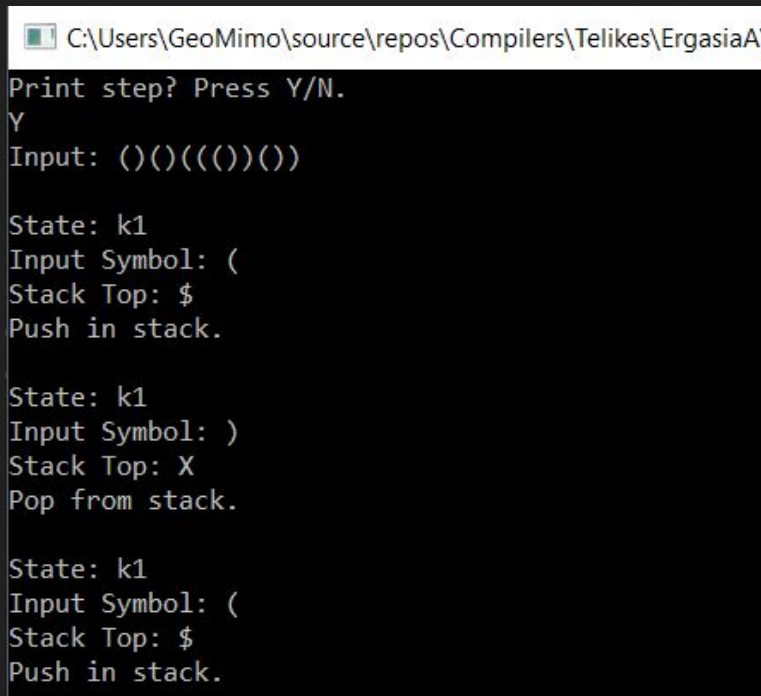
`bool verbose`: Εκτύπωση ή όχι.

Και επιστρέφει αν έγινε σωστά κάποια παραγωγή / κατανάλωση ή όχι, ενώ παράλληλα αλλάζει την κατάσταση και τη στοίβα.

Η παραπάνω συνάρτηση ελέγχει την κατάσταση, το σύμβολο εισόδου και την κορυφή της στοίβας και στη συνέχεια είτε προσθέτει ένα στοιχείο στην κορυφή της στοίβας, είτε αφαιρεί, είτε αλλάζει την κατάσταση, είτε επιστρέφει σφάλμα αναγνώρισης, είτε επιτυχία.

Κατά την εκκίνηση του προγράμματος, ζητείται από το χρήστη να δώσει το όνομα του αρχείου προς ανάλυση. Αν είναι έγκυρο, αφαιρούνται από το κείμενο όλοι οι χαρακτήρες διάφοροι των συμβόλων '(' και ')'. Έτσι η μεταβλητή `input` περιέχει μόνο αυτούς τους χαρακτήρες. Στη συνέχεια, ζητείται από τον χρήστη αν θέλει να εκτυπώνονται τα βήματα (`verbose`) και αρχικοποιείται η στοίβα και η κατάσταση. Έπειτα, καλείται η συνάρτηση `stackAutomata` για κάθε σύμβολο του `input` και ελέγχεται η επιστρεφόμενη τιμή. Τέλος, γίνεται έλεγχος αν η συμβολοσειρά αναγνωρίστηκε επιτυχώς και εμφανίζεται κατάλληλο μήνυμα.

# Screenshots



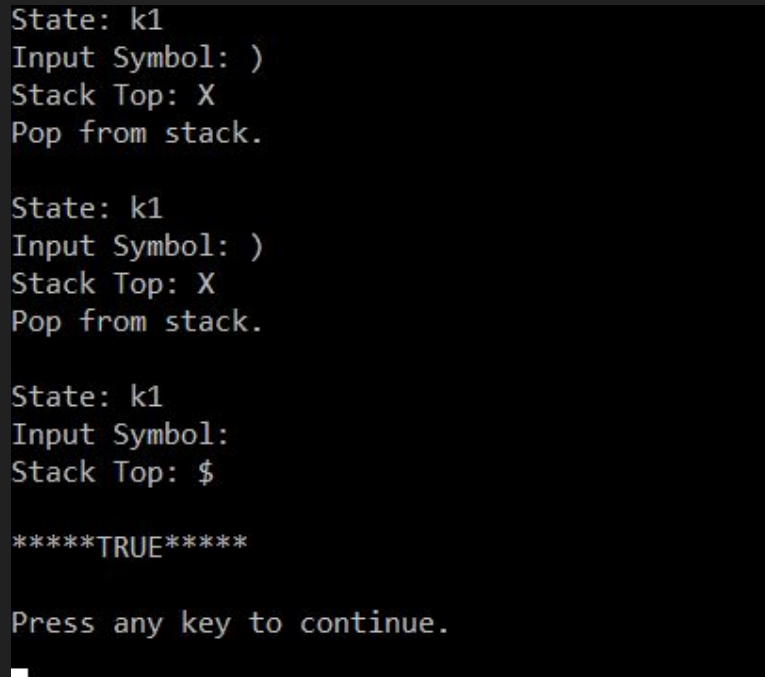
```
C:\Users\GeoMimo\source\repos\Compilers\Telikes\ErgasiaA
Print step? Press Y/N.
Y
Input: ()()((()))()

State: k1
Input Symbol: (
Stack Top: $
Push in stack.

State: k1
Input Symbol: )
Stack Top: X
Pop from stack.

State: k1
Input Symbol: (
Stack Top: $
Push in stack.
```

Εικόνα 1. Εκκίνηση ανάλυσης.



```
State: k1
Input Symbol: )
Stack Top: X
Pop from stack.

State: k1
Input Symbol: )
Stack Top: X
Pop from stack.

State: k1
Input Symbol:
Stack Top: $

*****TRUE*****

Press any key to continue.
█
```

Εικόνα 2. Επιτυχής αναγνώριση.

# Εργασία Β

Η δεύτερη εργασία είχε ως στόχο τη δημιουργία γεννήτριας συμβολοσειρών της ενότητας 3.2.4. Για την ανάπτυξη του προγράμματος επιλέχθηκε η γλώσσα C++.

Ουσιαστικά, το πρόγραμμα ξεκινάει με μια αρχική κατάσταση και με τυχαία επιλογή παραγωγής, παράγεται βήμα-βήμα η τελική έκφραση που ακολουθεί τους δοθέντους κανόνες.

# Η λογική του προγράμματος

Αρχικά, ορίζεται η συνάρτηση `vector<string> chooseRule(string token)`, η οποία επιστρέφει ένα `vector` που περιέχει τους κανόνες παραγωγής, ανάλογα με το εκάστοτε `token`.

Κατά την εκκίνηση του προγράμματος αρχικοποιούνται το `str` που περιέχει κάθε φορά την εξέλιξη της παραγωγής, το `token` που είναι ο μη-τερματικός από τον οποίο θα γίνει η παραγωγή και το `production` όπου περιέχει την εκάστοτε παραγωγή από το `token`. Έπειτα, ξεκινάει η διαδικασία της παραγωγής, η οποία επαναλαμβάνεται μέχρι να μην υπάρχει άλλο `token` για παραγωγή είτε μέχρι τις 100 παραγωγές ως δικλείδα ασφαλείας. Ακολουθεί η επιλογή κανόνα βάσει του `token` και η τυχαία επιλογή παραγωγής από τον κανόνα αυτόν. Τέλος, γίνονται ενημερώσεις των παραπάνω μεταβλητών, ενώ παράλληλα εκτυπώνονται τα βήματα.



# Screenshots

```
C:\Users\GeoMimo\source\repos\Compilers\Telikes\ErgasiaB_CPP\Debug\ErgasiaB_CPP.exe
1)      <expression>      (<expression> => <expression>+<term>)
2)      <expression>+<term>      (<expression> => <expression>+<term>)
3)      <expression>+<term>+<term>      (<expression> => <term>)
4)      <term>+<term>+<term>      (<term> => <term>*<factor>)
5)      <term>*<factor>+<term>+<term>      (<term> => <factor>)
6)      <factor>*<factor>+<term>+<term>      (<factor> => c)
7)      c*<factor>+<term>+<term>      (<factor> => a)
8)      c*a+<term>+<term>      (<term> => <factor>)
9)      c*a+<factor>+<term>      (<factor> => a)
10)     c*a+a+<term>      (<term> => <term>*<factor>)
11)     c*a+a+<term>*<factor>      (<term> => <factor>)
12)     c*a+a+<factor>*<factor>      (<factor> => c)
13)     c*a+a+c*<factor>      (<factor> => c)
14)     c*a+a+c*c
Would you like to repeat? [y/n]
```

Εικόνα 1. Τυχαία παραγωγή.

# Εργασία Γ

Η τρίτη εργασία είχε ως στόχο την κατασκευή ενός top-down συντακτικού αναλυτή που αναγνωρίζει συμβολοσειρές που εισάγει ο χρήστης αν αυτές ακολουθούν τους απαραίτητους συντακτικούς κανόνες. Το πρόγραμμα έχει υλοποιηθεί με τη γλώσσα προγραμματισμού C++.

Παρακάτω, εξετάζεται αν η δοθείσα γραμματική είναι LL(1).

# Κανόνες Παραγωγής

Οι κανόνες παραγωγής είναι :

1.  $S \rightarrow [A]$
2.  $A \rightarrow BE$
3.  $B \rightarrow x \mid y \mid S$
4.  $E \rightarrow :A \mid +A \mid \varepsilon$

Οι κανόνες γράφονται πιο αναλυτικά ως εξής:

1.  $S \rightarrow [A]$
2.  $A \rightarrow BE$
3.  $B \rightarrow x$
4.  $B \rightarrow y$
5.  $B \rightarrow S$
6.  $E \rightarrow :A$
7.  $E \rightarrow +A$
8.  $E \rightarrow \varepsilon$

# Σύνολα FIRST

- $\text{FIRST}(S) = \text{FIRST}([A]) = \{ [ \}$
- $\text{FIRST}(A) = \text{FIRST}(B) - \{ \varepsilon \} \quad (1)$
- $\text{FIRST}(B) = \text{FIRST}(S) \cup \{ x, y \} = \{ x, y, [ \}$
- $\text{FIRST}(E) = \{ : \} \cup \{ + \} \cup \{ \varepsilon \} = \{ :, +, \varepsilon \}$

Άρα από (1)  $\Rightarrow \text{FIRST}(A) = \{ x, y, [ \}$

Επομένως:  $\text{FIRST}(S) = \{ [ \}$   
 $\text{FIRST}(A) = \text{FIRST}(B) = \{ x, y, [ \}$   
 $\text{FIRST}(E) = \{ :, +, \varepsilon \}$

# Σύνολα FOLLOW

ΚΑΝΟΝΑΣ 1:

$$\$ \in \text{FOLLOW}(S) \quad (1)$$

ΚΑΝΟΝΑΣ 2:

$$S \rightarrow [A] \Rightarrow \text{FIRST}([A]) - \{\epsilon\} \subseteq \text{FOLLOW}(A) \Rightarrow ] \in \text{FOLLOW}(A) \quad (2)$$

$$A \rightarrow BE \Rightarrow \text{FIRST}(E) - \{\epsilon\} \subseteq \text{FOLLOW}(B) \Rightarrow \{:, +\} \in \text{FOLLOW}(B) \quad (3)$$

ΚΑΝΟΝΑΣ 3:

$$A \rightarrow \epsilon BE \text{ και } \epsilon \in \text{FIRST}(E) \Rightarrow \text{FOLLOW}(A) \subseteq \text{FOLLOW}(B) \Rightarrow \text{FOLLOW}(A) \subseteq \text{FOLLOW}(B) \quad (4)$$

$$A \rightarrow BE \Rightarrow \text{FOLLOW}(A) \subseteq \text{FOLLOW}(E) \quad (5)$$

$$B \rightarrow \epsilon S \Rightarrow \text{FOLLOW}(B) \subseteq \text{FOLLOW}(S) \quad (6)$$

$$E \rightarrow :A \Rightarrow \text{FOLLOW}(E) \subseteq \text{FOLLOW}(A) \quad (7)$$

Επομένως έχουμε:

$$(5),(6) \Rightarrow \text{FOLLOW}( A ) = \text{FOLLOW}( E ) \quad (8)$$

$$(2) \Rightarrow \text{FOLLOW}( A ) = \{ ] \}$$

$$(8) \Rightarrow \text{FOLLOW}( E ) = \{ ] \}$$

$$(3) \Rightarrow \text{FOLLOW}( B ) = \{ :, +, ] \}$$

$$(3),(6) \Rightarrow \text{FOLLOW}( S ) = \{ \$, :, +, ] \}$$

# Σύνολα EMPTY

- $\text{EMPTY}([A]) = \text{FALSE}$
- $\text{EMPTY}(\text{BE}) = \text{FALSE}$
- $\text{EMPTY}(x) = \text{FALSE}$
- $\text{EMPTY}(y) = \text{FALSE}$
- $\text{EMPTY}(:A) = \text{FALSE}$
- $\text{EMPTY}(+A) = \text{FALSE}$
- $\text{EMPTY}(\varepsilon) = \text{TRUE}$

# Σύνολα LOOKAHEAD

- $\text{LOOKAHEAD}( S \rightarrow [A] ) = \{ [ ] \}$
- $\text{LOOKAHEAD}( A \rightarrow BE ) = \text{FIRST}( B ) = \{ x, y, [ ] \}$
- $\text{LOOKAHEAD}( B \rightarrow x ) = \text{FIRST}( x ) = \{ x \}$
- $\text{LOOKAHEAD}( B \rightarrow y ) = \text{FIRST}( y ) = \{ y \}$
- $\text{LOOKAHEAD}( B \rightarrow S ) = \text{FIRST}( S ) = \{ [ ] \}$
- $\text{LOOKAHEAD}( E \rightarrow :A ) = \{ : \}$
- $\text{LOOKAHEAD}( E \rightarrow +A ) = \{ + \}$
- $\text{LOOKAHEAD}( E \rightarrow \varepsilon ) = \text{FOLLOW}(E) = \{ ] \}$



# Απόδειξη LL(1)

Είναι:

1.  $\text{LOOKAHEAD}(E \rightarrow \varepsilon) \cap \text{LOOKAHEAD}(:A) = \emptyset$
2.  $\text{LOOKAHEAD}(E \rightarrow \varepsilon) \cap \text{LOOKAHEAD}(+A) = \emptyset$
3.  $\text{LOOKAHEAD}(B \rightarrow x) \cap \text{LOOKAHEAD}(B \rightarrow S) = \emptyset$
4.  $\text{LOOKAHEAD}(B \rightarrow y) \cap \text{LOOKAHEAD}(B \rightarrow S) = \emptyset$

Επομένως, η γραμματική είναι LL(1).

# ΣΥΝΤΑΚΤΙΚΟΣ ΠΙΝΑΚΑΣ

	X	Y	:	+	[	]	$\epsilon$	\$
S					[A]			
A	BE	BE			BE			
B	X	Y			S			
E			:A	+A		$\epsilon$		

# Επίδειξη $[[y:x]+[x:y]]$

ΣΤΟΙΒΑ	ΕΙΣΟΔΟΣ	ΣΤΟΙΧΕΙΟ	ΠΑΡΑΓΩΓΗ
\$S	$[[y:x]+[x:y]]\$$	$M(S, [ )$	$S \rightarrow [A]$
\$]A[	$[[y:x]+[x:y]]\$$		
\$]A	$[y:x]+[x:y]]\$$	$M(A, [ )$	$A \rightarrow BE$
\$]EB	$[y:x]+[x:y]]\$$	$M(B, [ )$	$B \rightarrow S$
\$]ES	$[y:x]+[x:y]]\$$	$M(S, [ )$	$S \rightarrow [A]$
\$]E]A[	$[y:x]+[x:y]]\$$		
\$]E]A	$y:x]+[x:y]]\$$	$M(A, y )$	$A \rightarrow BE$
\$]E]EB	$y:x]+[x:y]]\$$	$M(B, y )$	$B \rightarrow y$
\$]E]Ey	$y:x]+[x:y]]\$$		

\$]E]E	:x]+[x:y]]\$	M( E, : )	E -> :A
\$]E]A:	:x]+[x:y]]\$		
\$]E]A	x]+[x:y]]\$	M( A, x )	A -> EB
\$]E]EB	x]+[x:y]]\$	M( B, x )	B -> x
\$]E]Ex	x]+[x:y]]\$		
\$]E]E	] +[x:y]]\$	M( E, ] )	E -> ε
\$]E]	] +[x:y]]\$		
\$]E	+ [x:y]]\$	M( E, + )	E -> +A
\$]A+	+ [x:y]]\$		
\$]A	[x:y]]\$	M( A, x )	A -> BE

\$]EB	[x:y]]\$	M( B, [ )	B -> S
\$]ES	[x:y]]\$	M( S, [ )	S -> [A]
\$]E]A[	[x:y]]\$		
\$]E]A	x:y]]\$	M( A, x)	A -> BE
\$]E]EB	x:y]]\$	M( B, x)	B -> x
\$]E]Ex	x:y]]\$		
\$]E]E	:y]]\$	M( E, : )	E -> :A
\$]E]A:	:y]]\$		
\$]E]A	y]]\$	M( A, y )	A -> EB
\$]E]EB	y]]\$	M( B, y)	B -> y

\$]E]Ey	y]]\$		
\$]E]E	]]\$	M( E, ] )	E -> ε
\$]E]	]]\$		
\$]E	]\$	M( E, ] )	E -> ε
\$]	]\$		
\$	\$		

Άρα, όπως είδαμε από τον πίνακα, έχουμε επιτυχία αναγνώρισης από τον αναλυτή.

# Λογική του προγράμματος

Αρχικά ζητείται από τον χρήστη να εισάγει την συμβολοσειρά προς ανάλυση. Η συμβολοσειρά αυτή, από `string` μετατρέπεται σε `List<char>`, προστίθεται στο τέλος το σύμβολο `$` και ύστερα αρχικοποιείται η στοίβα. Στη συνέχεια, ελέγχεται το στοιχείο στην κορυφή της στοίβας και ανάλογα με το εκάστοτε σύμβολο εισόδου γίνονται οι απαραίτητες παραγωγές / καταναλώσεις μέχρι το μέγεθος της στοίβας ή της λίστας εισόδου να γίνουν μικρότερες ίσες του 1. Τότε, ελέγχεται αν η κορυφή και των δύο λιστών είναι ίσες με το `$` και εμφανίζεται το κατάλληλο μήνυμα.

Επιπλέον, σε περίπτωση που το στοιχείο της κορυφής της στοίβας και το σύμβολο εισόδου δεν δίνουν κάποιο αποτέλεσμα, η ανάλυση σταματάει και εμφανίζεται το κατάλληλο μήνυμα. Σε κάθε βήμα εκτυπώνονται οι παραγωγές και οι καταναλώσεις.

# Screenshots

```
C:\Users\GeoMimo\source\repos\Compilers\Telikes\ErgasiaC\Debug\ErgasiaC.exe

Please give an expression to analyse.
[x:y]
$]A[      [x:y]$
$]A      x:y]$
$]EB      x::y]$
$]Ex      x:y]$
$]E       :y]$
$]A:      :y]$
$]A       y]$
$]EB      y]$
$]Ey      y]$
$]E       ]]$
$]        ]]$
$         $
Expression was successfully recognized!

Would you like to repeat? [y/n]
-
```

Εικόνα 1.  
Επιτυχημένη ανάλυση - Αναγνωρίστηκε

```
C:\Users\GeoMimo\source\repos\Compilers\Telikes\ErgasiaC\Debug\ErgasiaC.exe

Please give an expression to analyse.
[x:]
$]A[      [x:]]$
$]A       x:]]$
$]EB      x:]]$
$]Ex      x:]]$
$]E       :]]$
$]A:      :]]$
$]A       ]]$
$]EB      ]]$
Expression was not recognized!

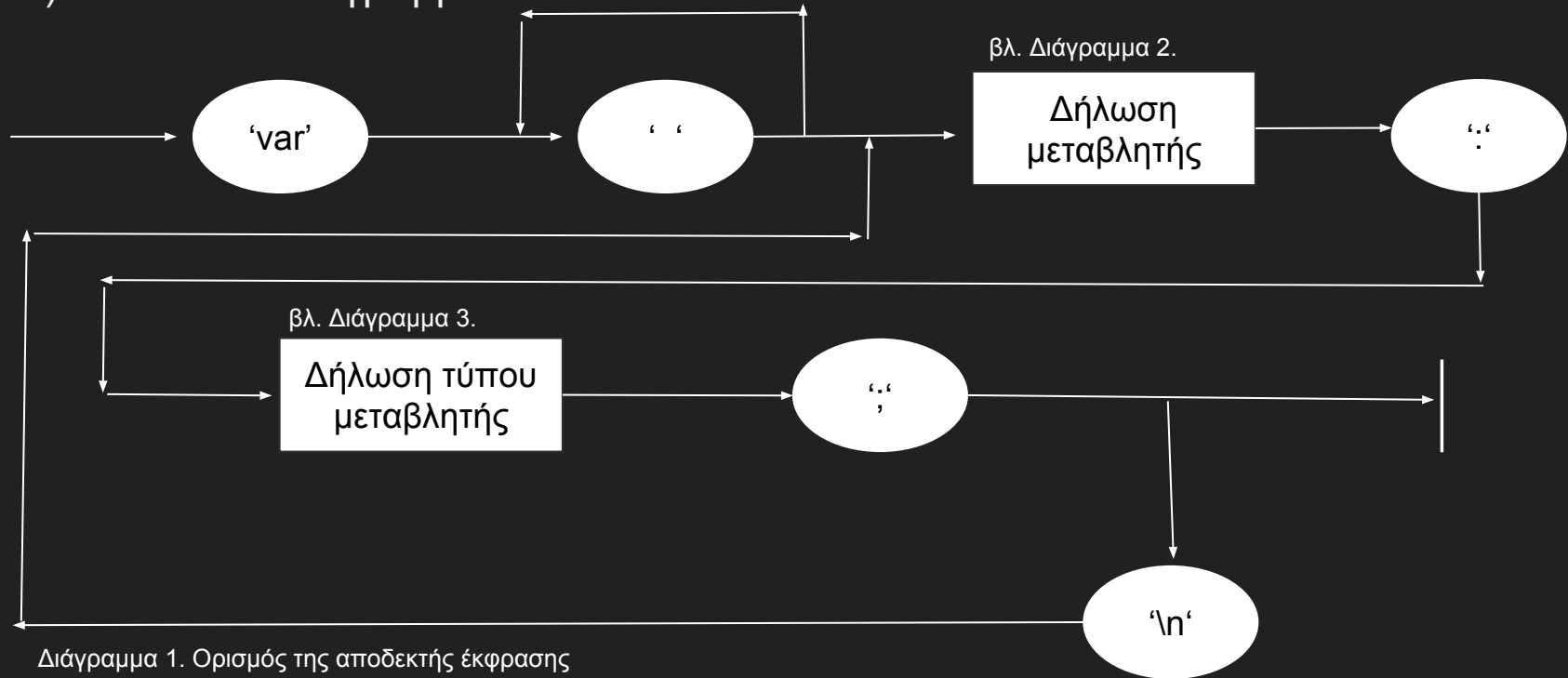
Would you like to repeat? [y/n]
-
```

Εικόνα 2.  
Αποτυχημένη ανάλυση - Δεν αναγνωρίστηκε

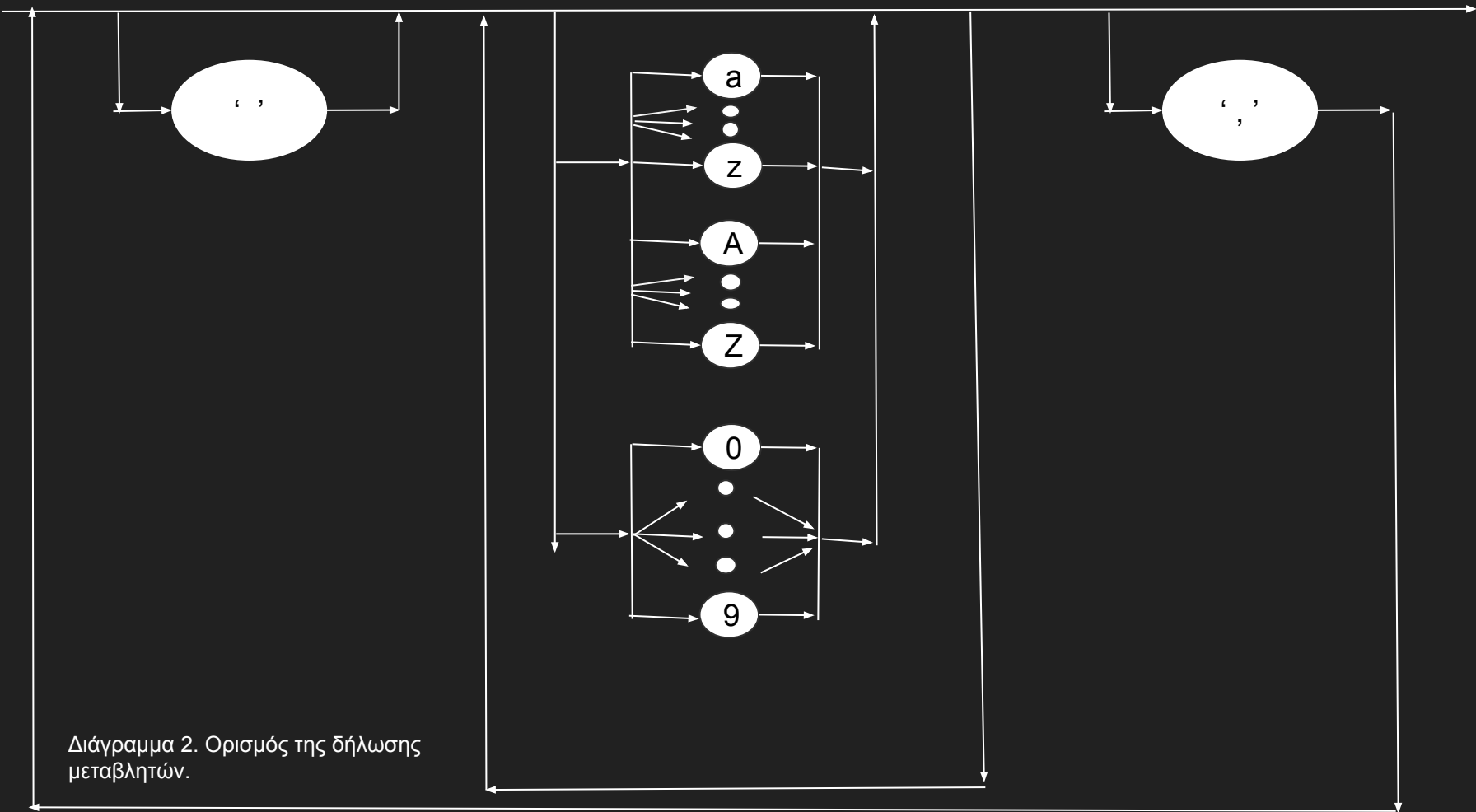


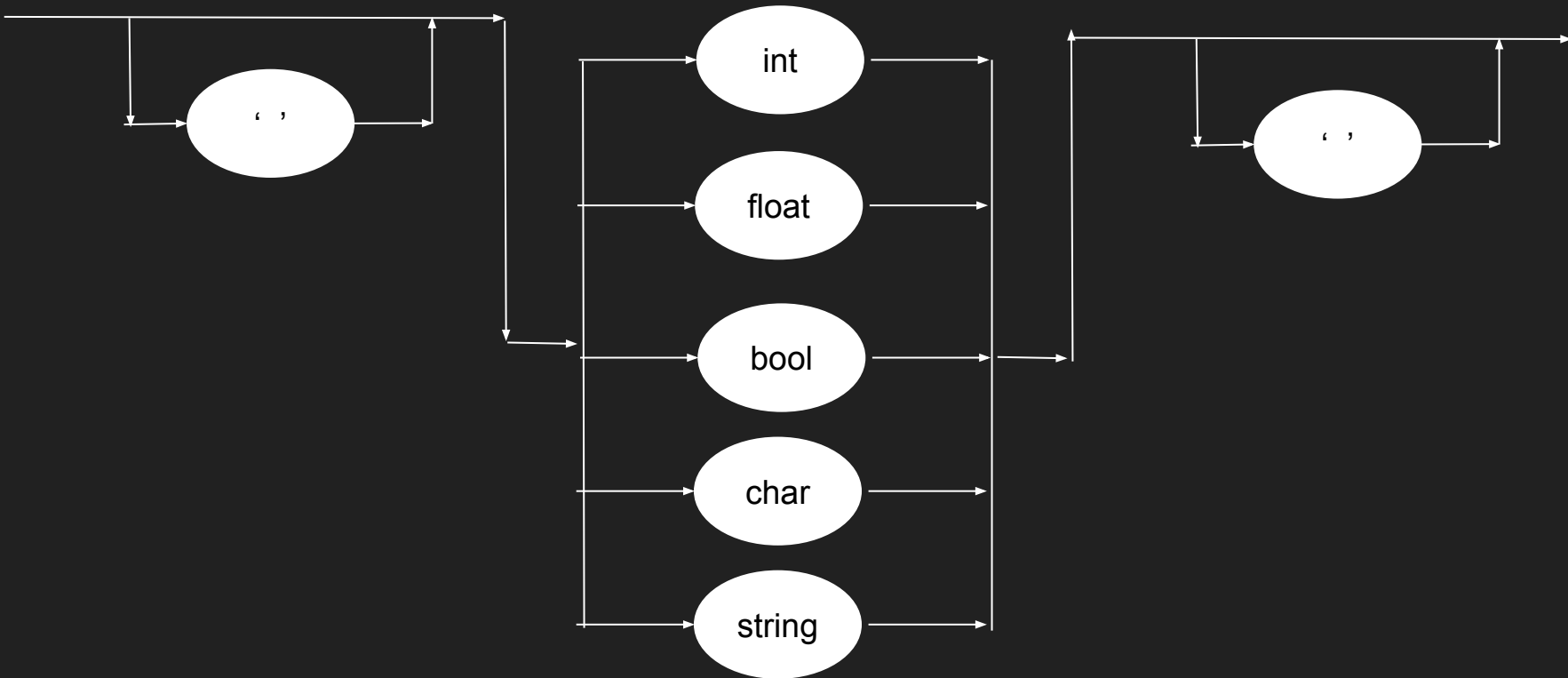
# Εργασία Δ

α) Συντακτικό διάγραμμα.



Διάγραμμα 1. Ορισμός της αποδεκτής έκφρασης του προγράμματος,





Διάγραμμα 3. Ορισμός της δήλωσης  
τύπου μεταβλητών.

β) EBNF σύνταξη.

1. Δήλωση ::= "var" ( " " ) { " " } ( Όνομα-μεταβλητής { " , " Όνομαμεταβλητής | { " " } } ) " : " { " " } Τύπος-Μεταβλητής { " " } " ; " "\n" { ( Όνομα-μεταβλητής { " , " Όνομα-μεταβλητής | { " " } } ) " : " { " " } Τύπος-Μεταβλητής { " " } " ; " "\n" }.
2. Όνομα-μεταβλητής ::= Γράμμα | Ψηφίο.
3. Τύπος-μεταβλητής ::= "int" | "float" | "bool" | "char" | "string".

## γ) Το πρόγραμμα Flex

Το πρόγραμμα που έγινε σε Flex υλοποιήθηκε ως εξής:

Στο αρχικό τμήμα δηλώσεων εισάγουμε κάποιες βιβλιοθήκες απο τη C που θα μας βοηθήσουν και δηλώνουμε κάποιες μεταβλητές που θα χρησιμεύσουν ως “διακόπτες”/boolean μεταβλητές παίρνοντας τιμές 0 ή 1. Έπειτα στο τμήμα με τους κανόνες έκφρασης έχουμε τον κανόνα που αναγνωρίζει την δήλωση μεταβλητών που ζητάει η άσκηση και τις λέξεις OK και EXIT που χρησιμεύουν για την μερική ή πλήρης ολοκλήρωση του προγράμματος επιστρέφοντας στις αντίστοιχες μεταβλητές ok ή exitProg την τιμή 1. Όταν πατάμε το κενό δεν θέλουμε να κάνει τίποτα και οποιαδήποτε άλλη έκφραση εκτός από την αποδεκτή θα επιστρέφει την τιμή 1 στη μεταβλητή wrong.

Έχοντας γράψει όλους τους κανόνες έκφρασης προχωράμε στο τμήμα με τις βοηθητικές διαδικασίες, στη `main()` πιο συγκεκριμένα, όπου με ένα `while` ελέγχουμε την μεταβλητή `end` αν έχει πάρει την τιμή 1 (εξηγήσαμε παραπάνω πότε συμβαίνει αυτό). Έπειτα, εμφανίζουμε τα κατάλληλα μηνύματα για να ενημερώσουμε το χρήστη ότι μπορεί να βάλει τις δηλώσεις του και με ένα ακόμη `while` κάνουμε έλεγχο εγκυρότητας του “`var`” με βάση τις μεταβλητές που επιστρέφουν οι κανόνες έκφρασης.

Αφού ολοκληρώσει τις δηλώσεις μεταβλητών ο χρήστης και εισαχθεί στο τέλος το `OK` ή το `EXIT` ανάλογα θα εμφανιστούν και τα κατάλληλα μηνύματα για το αν είναι αποδεκτή ή όχι η δήλωση μεταβλητών που έκανε από τη γλώσσα με βάση πάλι τις τιμές που θα πάρουν οι `start` και `wrong` (1 ή 0) από τους κανόνες έκφρασης.

# Screenshots

```
File Edit View Search Terminal Help  
geomimo@geomimo-VirtualBox:~/Downloads$ ./a.out
```

```
Enter your variable decleration.  
Type 'EXIT' in empty line to exit.  
Type 'OK' when you finish the decleration.  
var counter,i: int;  
name: string;  
flag, got: bool;  
  
OK
```

```
Your variable decleration is OK.
```

Εικόνα 1. Εκτέλεση του προγράμματος με αναγνωρίσιμη δήλωση μεταβλητών.

```
Enter your variable decleration.  
Type 'EXIT' in empty line to exit.  
Type 'OK' when you finish the decleration.  
var wrong: chat;  
OK
```

```
Not valid decleration.
```

Εικόνα 2. Εκτέλεση του προγράμματος με μη αναγνωρίσιμη δήλωση μεταβλητών (δεν αναγνωρίζει το chat)

```
Enter your variable decleration.  
Type 'EXIT' in empty line to exit.  
Type 'OK' when you finish the decleration.  
var test: int  
OK
```

```
Not valid decleration.
```

Εικόνα 3. Εκτέλεση του προγράμματος με μη αναγνωρίσιμη δήλωση μεταβλητών (δεν έχει ερωτηματικό στο τέλος)

# Εργασία Ε

Το πρόγραμμα Flex.

Το πρόγραμμα που έγινε σε Flex υλοποιήθηκε ως εξής:

Αρχικά είναι σημαντικό να σημειωθεί ότι χρησιμοποιήθηκαν μόνο αγγλικοί χαρακτήρες λόγω ασυμβατότητας με τους ελληνικούς.

Στο αρχικό τμήμα δηλώσεων εισάγουμε κάποιες βιβλιοθήκες απο τη C που θα μας βοηθήσουν και δηλώνουμε κάποιες μεταβλητές που θα χρησιμεύσουν ως “διακόπτες”/boolean μεταβλητές παίρνωντας τιμές 0 ή 1, καθώς και 2 πίνακες τύπου char που ο answer θα έχει κάθε φορά αυτό που έδωσε ο χρήστης και ο valid έχει τα γράμματα απο το Α-Ε μικρά και κεφαλαία.



Έπειτα έχουμε 4 κανόνες έκφρασης:

- 1) Σε οποιαδήποτε άλλη έκφραση εκτός από αυτή που υποστηρίζει η γλώσσα μας θα επιστρέφεται η τιμή 1 στη μεταβλητή `wrong`.
- 2) Με το `end` σε κενή γραμμή θα τερματίζεται το πρόγραμμα μας επιστρέφοντας την τιμή 1 στη μεταβλητή `finish`.
- 3) Κάθε φορά που πατάμε το κενό θα γίνεται έλεγχος αν η μεταβλητή `wrong` είναι ίση με 1 δηλαδή υπάρχει λάθος και αν υπάρχει τότε θα εμφανίζει το κατάλληλο μήνυμα και κάνοντας “reset” το πρόγραμμα.
- 4) Ο κανόνας που ορίζει την έκφραση που δέχεται η γλώσσα θα ξεκινάει με την λέξη `dinetai` μετά μία απο τις 3 γεωμετρικές οντότητες (`gonia`, `trigono`, `tetragono`) και μετά θα δέχεται δύο ή τρία ή τέσσερα γραμματα απο το Α εως το Ε και μετά οτιδήποτε άλλο τυχόν εισάγει ο χρήστης (θα εξηγήσουμε τη χρησιμότητα του παρακάτω).

Όταν αναγνωριστεί η έκφραση θα γίνει έλεγχος μέσα σε ένα while ότι η μεταβλητή check είναι διάφορη του 2 (η μεταβλητή check παίρνει τιμες 0 όταν το πρόγραμμα πάρει την έκφραση πρώτη φορά για επεξεργασία, 1 όταν η επεξεργασία της έκφρασης που έδωσε ο χρήστης είναι εντελώς σωστή και 2 όταν υπάρχει κάποιο λάθος). Εφόσον και η μεταβλητή wrong δεν είναι 1 τότε το πρόγραμμα ανάλογα τη τιμή του check θα εκτελέσει και κάποιες διαδικασίες.

Πιο συγκεκριμένα αν check = 0 θα ξεκινήσει πρώτα μεταφέροντας στον πίνακα answer ό,τι έδωσε ο χρήστης. Μετά θα κάνει προσπάθεια του πίνακα μέχρι να βρεί κενό και μετά ξανά μέχρι να βρει χαρακτήρα. Μόλις βρει τη δεύτερη λέξη που είναι η γεωμετρική οντότητα που έδωσε ο χρήστης θα ελεγχει το δεύτερο της γράμμα για να καθορίσει ποια απο τις τρεις οντότητες είναι και θα δώσει την κατάλληλη τιμή στη μεταβλητή shape (2,3,4 γωνία τρίγωνο τετράγωνο).

Σε αυτό το σημείο είναι που θα εντοπιστεί αν υπάρχει κάποιο λάθος καθώς μόλις εντοπιστεί και η τρίτη λέξη (η ονομασία της οντότητας) θα γίνει έλεγχος για να διαπιστωθεί αν έχει επαναληφθεί κάποιο από τα 5 γράμματα που είναι δεκτά Α-Ε, αν το βρει μία φορά θα θέσει την τιμή της μεταβλητής που αντιστοιχεί σε αυτό το γράμμα ίσο με 1, αν το βρει και δεύτερη φορά, τότε η check θα γίνει 2 και η wrong 1.

Μετά θα γίνει ένας μικρός έλεγχος εγκυρότητας του τελευταίου γράμματος που έχει δώσει ο χρήστης κάνοντας προσπέλαση ενός πίνακα που έχει τα 5 γράμματα που γίνονται δεκτά και ελέγχοντας αν κάποιο από αυτά αντιστοιχεί στο γράμμα που ελέγχουμε, αν δεν αντιστοιχεί σε αυτά τότε η μεταβλητή found γίνεται 1.

Εφόσον ολοκληρωθεί και αυτός ο έλεγχος, αν η found είναι ακόμα 0, τότε η wrong θα γίνει 1 και η check 2, αλλιώς για κάθε γράμμα του ονόματος της οντότητας θα γίνει άλλος έλεγχος που θα ελέγχει αν υπάρχουν περισσότερα γράμματα από ότι επιτρέπεται για τη συγκεκριμένη οντότητα. Πχ τρίγωνο με 4 γράμματα είναι λάθος.

Σε αυτό θα μας βοηθήσει η μεταβλητή shape αφού θα ελέγξουμε πόσα γράμματα είναι το όνομα της οντότητας και την τιμή της shape, αν είναι λιγότερα ή περισσότερα, η check θα γίνει 2 και η wrong 1 αλλιώς η check θα γίνει 1 και θα ξαναγίνουν όλα από την αρχή με τη διαφορά όμως ότι τώρα αφού έχει ελεγχθεί η εισαγωγή από το χρήστη, το πρόγραμμα θα τυπώσει στην οθόνη όλα τα στοιχεία που ζήτηγε η άσκηση. Σε κάθε άλλη περίπτωση που η check ήταν 2 και η wrong 1 θα εμφανιζόντουσαν τα κατάλληλα μηνύματα.

Στο μέρος με τις βοηθητικές διαδικασίες μέσα σε ένα while όπου τερματίζει όταν το finish γίνει 1 έχουμε κάθε φορά εκκαθάριση του πίνακα answer και όλων των μεταβλητών για να μπορούν να γίνουν όλοι οι έλεγχοι σωστά.

# Screenshots

```
Please give your shape and sides.  
YOU CAN TYPE 'end' IN A EMPTY LINE ANYTIME TO EXIT.
```

```
dinetai trigono ABC  
dinetai : Verb  
trigono : Geometrical entity  
ABC : Name of geometrical entity
```

```
Please give your shape and sides.  
YOU CAN TYPE 'end' IN A EMPTY LINE ANYTIME TO EXIT.
```

Εικόνα 1. Εκτέλεση του προγράμματος και εκτύπωση στοιχείων γεωμετρικής οντότητας με σωστή σύνταξη.

```
Please give your shape and sides.  
YOU CAN TYPE 'end' IN A EMPTY LINE ANYTIME TO EXIT.
```

```
dinetai tetragono agcb
```

```
Not valid shape and sides please try again
```

```
Please give your shape and sides.  
YOU CAN TYPE 'end' IN A EMPTY LINE ANYTIME TO EXIT.
```

Εικόνα 2. Εκτέλεση του προγράμματος και εκτύπωση μυνήματος με λάθος σύνταξη (γράμμα g)

```
Please give your shape and sides.  
YOU CAN TYPE 'end' IN A EMPTY LINE ANYTIME TO EXIT.
```

```
dinetai trigono abcd
```

```
Not valid shape and sides please try again
```

Εικόνα 3. Εκτέλεση του προγράμματος και εκτύπωση μυνήματος με λάθος σύνταξη (γράμματα περισσότερα από 3 επειδή είναι τρίγωνο)