# Εισαγωγή

…

# Άσκηση Α

## Εισαγωγή

Η πρώτη άσκηση έχεις ως στόχο την υλοποίηση ενός προγράμματος που δέχεται ως είσοδο ένα αρχείο πηγαίου κώδικα και μέσω μιας συγκεκριμένης διαδικασίας αναγνωρίζει τον αριθμό των παρενθέσεων και αποφαίνεται για την ορθότητά τους.

Το πρόγραμμα σε αρχικό στάδιο δέχεται ως όρισμα τη διαδρομή του αρχείο που θα λάβει ως είσοδο ώστε να αρχίσει την επεξεργασία. Έπειτα, διαβάζει τα περιεχόμενά του και αφαιρεί όλους τους χαρακτήρες εκτός των παρενθέσεων. Το πρόβλημα που καλείται το πρόγραμμα να αντιμετωπίσει έγκειται στον τρόπο με τον οποίο θα αναγνωριστούν οι ανοιχτές και κλειστές παρενθέσεις λαμβάνοντας υπ’ όψη τη σειρά με την οποία βρίσκονται.

## Επίλυση

Με τη χρήση ενός Ντετερμινιστικού Αυτόματου Στοίβας (ΝΑΣ) το ζήτημα αντιμετωπίζεται αρκετά πιο εύκολα. Σύμφωνα με τον ορισμό ενός ΝΑΣ, στη δική μας περίπτωση ορίζεται ως εξής:

**Μ = (K, T, V, p, k1, A1, F)**

**Όπου:**

Σύνολο Καταστάσεων (K)  
{ **k0**, **k1**, **k2** }

Αλφάβητο Εισόδου (T)  
{ **(**, **)** }

Αλφάβητο Συμβόλων (V)  
{ **(**, **)**, **$** }

Συνάρτηση Στοίβας (p)  
f(**k0**, **ε**, **ε**) = { (**k1**, **$**) }  
f(**k1**, **ε**, **(**) = { (**k1**, **(**) }  
f(**k1**, **(**, **)**) = { (**k1**, **ε**) }  
f(**k1**, **$**, **ε**) = { (**k2**, **ε**) }

Αρχική Κατάσταση (k1)  
**k0**

Αρχικό Σύμβολο Στοίβας (A1)  
**$**

Σύνολο Τελικών Καταστάσεων (F)  
{ **t2** }

Με άλλα λόγια, κατά την έναρξη του προγράμματος, αφού διαβαστούν τα περιεχόμενα του αρχείου, η στοίβα σύμφωνα με τη συνάρτησή της, αρχικοποιείται και μεταβαίνει στην κατάσταση **k1** με το σύμβολο **$** στο πυθμένα της στοίβας. Στη συνέχεια, εάν δεχτεί η στοίβα το σύμβολο **(** εισάγει τη παρένθεση και παραμένει στην κατάσταση **k1** μέχρις ότου το σύμβολο **)** να διαβαστεί. Σε αυτή τη περίπτωση, αφαιρείται η προηγούμενη παρένθεση **(** από τη στοίβα. Συνεχίζουμε αυτή τη διαδικασία μέχρι να καταναλωθεί η συμβολοσειρά από παρενθέσεις και να μεταβούμε στη τελική κατάσταση **k2**.

Κατά την κατανάλωση της συμβολοσειράς, εάν διαπιστωθεί πως δεν υπάρχει καθορισμένη μετάβαση σύμφωνα με τη συνάρτηση της στοίβας, η διαδικασία σταματάει και οι παρενθέσεις δεν αναγνωρίζονται.

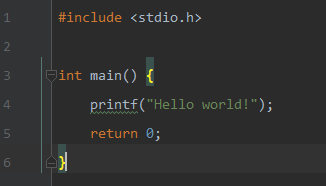
## Εκτέλεση

Το πρόγραμμα έχει υλοποιηθεί στη **Python 3** και δέχεται **1** όρισμα.

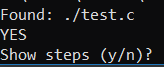
Ενδεικτικά, μέσω της γραμμής εντολών το πρόγραμμα εκτελείται ως εξής:

* Κάνουμε **cd compilers-project**
* Κάνουμε **cd exercise-1**
* Εκτελούμε το πρόγραμμα **python parentheses.py ./test.c**

Αν υποθέσουμε ότι το αρχείο **test.c** περιέχει τον πηγαίο κώδικα για ένα από hello world πρόγραμμα στη C και βρίσκεται στον ίδιο φάκελο με το αρχείο **parentheses.py**:



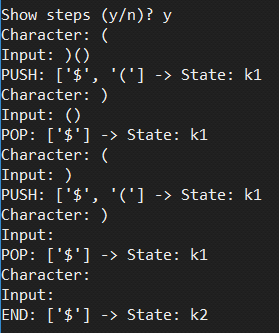
Τότε παρατηρούμε ότι το πρόγραμμά μας ανταποκρίνεται ως εξής:



Το πρόγραμμα βρήκε επιτυχώς το αρχείο **test.c** και αφού ανέλυσε τα περιεχόμενα και τις παρενθέσεις, τυπώνει «**YES**» στη γραμμή εντολών. Η ένδειξη αυτή σημαίνει πως οι παρενθέσεις που βρίσκονται στο πηγαίο κώδικα του test.c είναι σωστές.

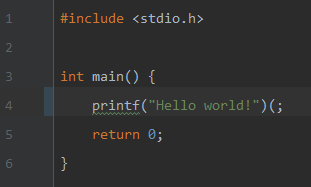
Πράγματι αφού έχουμε τις εξής παρενθέσεις: **()()**

Στη συνέχεια, παρατηρούμε πως το πρόγραμμα δίνει την επιλογή στο χρήστη να του εκτυπώσει αναλυτικά όλα τα βήματα που οδήγησαν σε αυτό το αποτέλεσμα.

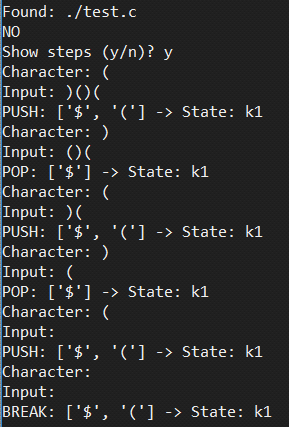


Στο τελευταίο βήμα, μπορούμε να διακρίνουμε πως η στοίβα περιέχει το σύμβολο **$** και βρίσκεται στην κατάσταση **k2**, άρα η συμβολοσειρά αναγνωρίζεται.

Σε μια άλλη περίπτωση που οι παρενθέσεις στο αρχείο του πηγαίου κώδικα δεν ήταν ισορροπημένες:



Το αποτέλεσμα θα ήταν «**ΝΟ**» αφού πράγματι θα είχαμε τις εξής παρενθέσεις: **()()(**



Έτσι, διαπιστώνουμε ότι η στοίβα παρέμεινε στη κατάσταση **k1** με μια εκκρεμής ανοιχτή παρένθεση.

# Άσκηση Β

## Εισαγωγή

Η δεύτερη άσκηση έχει ως θέμα την υλοποίηση ενός προγράμματος-γεννήτριας συμβολοσειρών σύμφωνα με την γραμματική που ορίζεται στην Ενότητα 3.2.4 του βιβλίου. Το πρόγραμμα όπως και στη πρώτη άσκηση έχει υλοποιηθεί στη γλώσσα προγραμματισμού Python, εφόσον είναι αρκετά ευέλικτη σε τέτοια θέματα παραγωγής συμβολοσειρών.

## Επίλυση

Ορίζουμε μια γραμματική G σύμφωνα με το παράδειγμα του βιβλίου.

**G = (S, P, N, T)**

Όπου:

Σύμβολο Εκκίνησης (S)  
**<έκφραση>**

Σύνολο Κανόνων Παραγωγής (P)  
**<έκφραση>** ::= <όρος> | <έκφραση> + <όρος>  
**<όρος>** ::= <παράγοντας> | <όρος> \* <παράγοντας>  
**<παράγοντας>** ::= a | b | c

Σύνολο μη Τερματικών Συμβόλων (N)  
{ **<έκφραση>**, **<όρος>**, **<παράγοντας>** }

Σύνολο Τερματικών Συμβόλων (Τ)  
{ **a**, **b**, **c** }

Η γραμματική G έχει οριστεί με BNF (Backus-Naur Form), μεταγλώσσα που χρησιμοποιείται για τον ορισμό γλωσσών χωρίς συμφραζόμενα. Η παρούσα γραμματική δεν έχει συμφραζόμενα, γι’ αυτό με τη χρήση BNF μπορούμε εύκολα να υλοποιήσουμε ένα πρόγραμμα για τη προτάσεων αυτής της γλώσσας.

Το πρόγραμμα σε αρχικό στάδιο, αρχίζει τη παραγωγή μιας πρότασης από την **<έκφραση>**. Έπειτα, επιλέγει τυχαία και εφαρμόζει τον κατάλληλο κανόνα για την αντικατάσταση του μη τερματικού συμβόλου. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου να μην υπάρχει η <έκφραση> στη πρότασή μας. Η ίδια διαδικασία εφαρμόζεται και για τα σύμβολα **<όρος>** και **<παράγοντας>**. Όσο υπάρχουν μη τερματικά σύμβολα, επαναλαμβάνεται η εφαρμογή των κανόνων της γραμματικής.

Επειδή η εφαρμογή των κανόνων είναι τυχαία, μπορεί να καταλήξουμε σε μεγάλους βρόχους επανάληψης, δηλαδή σε μεγάλες προτάσεις που προέκυψαν έπειτα από την εφαρμογή πολλών κανόνων στη σειρά. Γι’ αυτό το λόγο, έχει τεθεί ένα όριο στον αριθμό των κανόνων που εφαρμόζονται. Έτσι αν έχουμε καταλήξει σε πρόταση που περιέχει μη τερματικά σύμβολα έπειτα από 50 εφαρμογές κανόνων, το πρόγραμμα σταματάει τη διαδικασία και εφαρμόζει τους πιο σύντομους κανόνες που θα μας οδηγήσουν σε τελικό αποτέλεσμα.

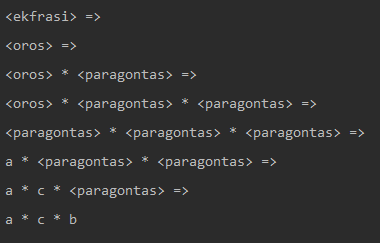
## Εκτέλεση

Το πρόγραμμα έχει υλοποιηθεί στη **Python 3** και δεν δέχεται ορίσματα.

Ενδεικτικά, μέσω της γραμμής εντολών το πρόγραμμα εκτελείται ως εξής:

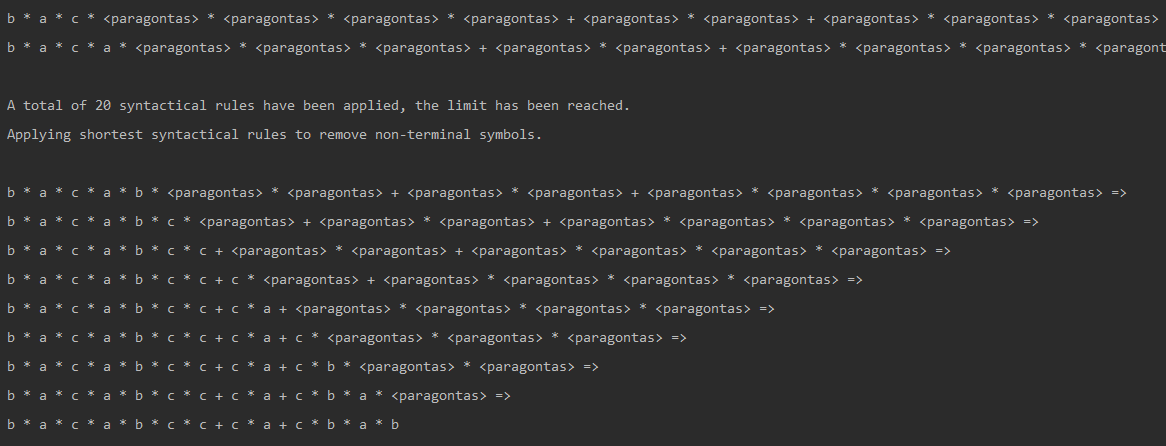
* Κάνουμε **cd compilers-project**
* Κάνουμε **cd exercise-2**
* Εκτελούμε το πρόγραμμα **python generator.py**

Αμέσως, με την εκτέλεση του προγράμματος παρατηρούμε πως έχει δημιουργηθεί μια πρόταση έπειτα από την εφαρμογή των κανόνων γραμματικής:



Σε αυτή τη περίπτωση, το πρόγραμμα επέλεξε να αντικαταστήσει την **<έκφραση>** με **<όρος>**, έπειτα **<όρος>** με **<όρος> \* <παράγοντας>** κ.ο.κ. Έτσι, φτάσαμε στη συμβολοσειρά **a \* c \* b**.

Υπάρχουν και άλλες περιπτώσεις όπου το πρόγραμμα μπορεί να συνεχίσει τη εφαρμογή αναδρομικών κανόνων αρκετές φορές:



Γι’ αυτό το λόγο, θέτουμε ένα όριο στην εφαρμογή των κανόνων ώστε αν έχουμε μεγάλους βρόχους επανάληψης, σταματάει η διαδικασία και καταλήγουμε όσο το συντομότερο δυνατό σε επιθυμητό αποτέλεσμα.

Στο παράδειγμά μας, το όριο έχει τεθεί σε 20 εφαρμογές κανόνων για τη παρουσίαση του παραδείγματος, όμως ρεαλιστικά το πρόγραμμα σταματάει μετά από **50 εφαρμογές**, που είναι αρκετά πιο σπάνιο.

# Άσκηση Γ

…

# Άσκηση Δ

…

# Άσκηση Ε

…