# Go-Reloaded

*by klampria*

Περιγραφή προβλήματος και καταγραφή κανόνων

https://platform.zone01.gr/intra/athens/div-01/go-reloaded

## Περιγραφή του προβλήματος

Το go-reloaded είναι ένα πρόγραμμα που δέχεται ένα αρχείο τύπου .txt και δημιουργεί ένα νέο αρχείο τύπου .txt. Το πρόγραμμα διαβάζει το αρχείο εισόδου, το επεξεργάζεται λαμβάνοντας υπόψιν συγκεκριμένους “κανόνες” (κυρίως συντακτικούς) και αποθηκεύει το αποτέλεσμα στο αρχείο που δημιουργεί. Η γλώσσα των αρχείων είναι αποκλειστικά τα αγγλικά.

## Καταγραφή των "κανόνων"

*https://platform.zone01.gr/intra/athens/div-01/go-reloaded*

Βάση του παραπάνω υπάρχουν 3 ήδη κανόνων. Κανόνες που συναντάνε κάποιο *(keyword)*, οι οποίοι μετατρέπουν την προηγούμενη λέξη ανάλογα με το είδος, κανόνες που ελέγχουν αν έχουν τοποθετηθεί με σωστά κενά τα σημεία στίξης και ένας κανόνας με σκοπό την σωστή χρήση του αγγλικού άρθρου ‘a’.

### *Κανόνας (keyword) - (hex)*

Όταν συναντάει το keyword-hex μετατρέπει τον προηγούμενο αριθμό από δεκαεξαδικό σε δεκαδικό σύστημα (και αφαιρείται το keyword από το κείμενο)

(Ex: "1E (hex) files were added" -> "30 files were added")

### *Κανόνας (keyword) - (bin)*

Όταν συναντάει το keyword-bin μετατρέπει τον προηγούμενο αριθμό από δυαδικό σε δεκαδικό σύστημα (και αφαιρείται το keyword από το κείμενο)

(Ex: "It has been 10 (bin) years" -> "It has been 2 years")

### *Κανόνας (keyword) - (up)*

Όταν συναντάει το keyword-up μετατρέπει σε κεφαλαία γράμματα την προηγούμενη λέξη (και αφαιρείται το keyword από το κείμενο)

(Ex: "Ready, set, go (up) !" -> "Ready, set, GO!")

Το (up) μπορεί να βρεθεί και ως (up, ‘X’) όπου ‘X’ ένας ακέραιος αριθμός. Σε αυτή την περίπτωση η μετατροπή θα γίνει στις ‘X’ προηγούμενες λέξεις.

(Ex: "This is so exciting (up, 2)" -> "This is SO EXCITING")

### *Κανόνας (keyword) - (low)*

Όταν συναντάει το keyword-low μετατρέπει σε πεζά γράμματα την προηγούμενη λέξη (και αφαιρείται το keyword από το κείμενο)

(Ex: "I should stop SHOUTING (low)" -> "I should stop shouting")

Το (low) μπορεί να βρεθεί και ως (low, ‘X’) όπου ‘X’ ένας ακέραιος αριθμός. Σε αυτή την περίπτωση η μετατροπή θα γίνει στις ‘X’ προηγούμενες λέξεις.

(Ex: "This is SO EXCITING  (low, 2)" -> "This is so exciting”)

### *Κανόνας (keyword) - (cap)*

Όταν συναντάει το keyword-cap  μετατρέπει το πρώτο γράμμα της προηγουμένης λέξης σε κεφαλαίο (και αφαιρείται το keyword από το κείμενο)

(Ex: "Welcome to the Brooklyn bridge (cap)" -> "Welcome to the Brooklyn Bridge")

Το (cap) μπορεί να βρεθεί και ως (cap, ‘X’) όπου ‘X’ ένας ακέραιος αριθμός. Σε αυτή την περίπτωση η μετατροπή θα γίνει στις ‘X’ προηγούμενες λέξεις.

(Ex: "This is so exciting (cap, 2)" -> "This is So Exciting")

### *Κανόνας για σημεία στίξης*

Όταν συναντήσουμε σημεία στίξης  ‘.’  ‘,’  ‘!’ ‘?’ ‘:’ ‘;’ ‘...’ ‘!?’ το πρόγραμμα τα τοποθετεί κολλητά στην προηγούμενη λέξη (αφαιρεί τα κενά δηλαδή αν υπάρχουν) και τοποθετεί ένα κενό πριν την επόμενη λέξη.

(Ex: "I was sitting over there ,and then BAMM !!" -> "I was sitting over there, and then BAMM!!")
(Ex: "I was thinking ... You were maybe right  !?" -> "I was thinking... You were maybe right!?")

Ομοίως για τα μονά και διπλά “αυτάκια”  (' ') (“ ”) θα πρέπει να βρίσκονται κολλητά με την επόμενη λέξη στο άνοιγμα και την προηγούμενη λέξη στο κλείσιμο ακόμα και αν βρίσκονται παραπάνω από μία λέξεις ενδιάμεσα.

(Ex: "I am exactly how they describe me: ' awesome '" -> "I am exactly how they describe me: 'awesome'")

(Ex: "As Elton John said: ' I am the most well-known homosexual in the world '" -> "As Elton John said: 'I am the most well-known homosexual in the world'")

*Κανόνας για το άρθρο 'a'*

Στην αγγλική γλώσσα το άρθρο 'an' χρησιμοποιείται όταν η επόμενη λέξη ξεκινάει από φωνήεν (a, e, i, o, u) ή το γράμμα (h) αντί του άρθρου 'a'. Αντίστοιχα, το πρόγραμμα εντοπίζει το άρθρο 'a' και το μετατρέπει σε 'an' όπου χρειάζεται.

(Ex: "There it was. A amazing rock!" -> "There it was. An amazing rock!")

## 3. Σύγκριση Pipeline / Fsm αρχιτεκτονικής

*'Η αρχιτεκτονική προγραμματισμού αφορά τη δομή ενός συστήματος λογισμικού, η οποία καθορίζει πώς τα διάφορα στοιχεία του αλληλεπιδρούν και πώς το σύστημα λειτουργεί σε υψηλό επίπεδο.'*
(Gemini @ google.com)

Συνήθως για να γράψουμε ένα κείμενο χρησιμοποιούμε κάποιο συγκεκριμένο μοντέλο/αρχιτεκτονική. Σε μία έκθεση, για παράδειγμα, ξεκινάμε με έναν πρόλογο, συνεχίζουμε στο κυρίως θέμα και τέλος έχουμε τον επίλογο. Αν το κείμενο μας είναι μία μεσαία παράγραφος μπορούμε τον πρόλογο και τον επίλογο να τις ξεχωρίσουμε σε μία πρόταση το καθένα. Στην περίπτωση που το κείμενο μας όμως αποτελείται από δύο προτάσεις δηλαδή ανεπίσημο ή μικρο περιεχόμενο δεν έχει νόημα να ακολουθήσουμε το παραπάνω μοντέλο/αρχιτεκτονική.

Ομοίως στον προγραμματισμό, σε προγράμματα πολύ μικρής κλίμακας που έχουμε φτιάξει μέχρι τώρα στο piscine σε ατομικό επίπεδο, δεν σκεφτόμασταν αρχιτεκτονικές, αφού το περιεχόμενο ήταν μικρό (μοναδική λειτουργία, πρόγραμμα σε 15 σειρές κώδικα). Όσο αυξάνεται όμως το μέγεθος και οι λειτουργίες του προγράμματος, τόσο αυξάνεται και η ανάγκη για την εύρεσή συγκεκριμένου τρόπου συγγραφής και υλοποιήσης του κώδικα. Ειδικά όταν μιλάμε για ένα ομαδικό project, όπου διαφορετικά άτομα ενώνουν κομμάτια κώδικα για να δημιουργηθεί ένα πρόγραμμα, προκειμένου να επιτευχθεί επικοινωνία σε ανθρώπινο και προγραμματιστικό επίπεδο, πρέπει να υπάρχει συγκεκριμένη αρχιτεκτονική ως γνώμονας για την δημιουργία του κώδικα.

*'Η αρχιτεκτονική FSM (Finite State Machine) είναι ένα υπολογιστικό μοντέλο που περιγράφει τη συμπεριφορά ενός συστήματος που μπορεί να βρίσκεται σε πεπερασμένο αριθμό καταστάσεων. Το σύστημα μεταβαίνει από τη μία κατάσταση στην άλλη βάσει των εισροών και της τρέχουσας κατάστασής του, μέσω μεταβάσεων (transitions).'*
(Gemini @ google.com)

Στο go-reloaded σημαίνει πως στο main loop του προγράμματος θα διαβάζουμε και θα ψάχνουμε keywords-παραβάσεις όλων τον παραπάνω κανόνων ταυτόχρονα και θα μετατρέπουμε ότι χρειάζεται από το input.txt στο output.txt. Θεωρητικά σαν αποτέλεσμα θα έχουμε ένα συμφωνημένο, πολλών γραμμών κώδικα που θα βρίσκει λύση για όλους τους κανόνες με την μοναδική 'ανάγνωση' του input που θα κάνει. Αυτό μεταφράζεται σε μεγάλη ταχύτητα, μικρή κατανάλωση μνήμης αλλά κατά την γνώμη μου (και το επίπεδο μου) spagheti κώδικα, αφού θα είναι δυσανάγνωστο και δύσκολο στο debugging.

*'Η αρχιτεκτονική PIPELINE είναι ένα σχεδιαστικό πρότυπο που χρησιμοποιείται σε πολλούς τομείς της πληροφορικής, όπως η αρχιτεκτονική υπολογιστών. Η βασική της αρχή είναι η διαίρεση μιας σύνθετης διαδικασίας σε μια σειρά από διακριτά, διαδοχικά στάδια, όπου η έξοδος του ενός σταδίου γίνεται η είσοδος του επόμενου.'*
(Gemini @ google.com)

Πλεονέκτημα της αρχιτεκτονικής Pipeline είναι η διαίρεση όλης της διαδικασίας σε διαδοχικά στάδια όπου η έξοδος ενός σταδίου γίνεται η είσοδος στο επόμενο. Επομένως αποφεύγουμε τον έλεγχο και ταυτόχρονη τήρηση όλων των κανόνων όπως είδαμε στο fsm. Δημιουργώντας ξεχωριστές loops που η κάθε μία διαβάζει και εξετάζει κάθε κανόνα ξεχωριστά στα δεδομένα που εισάγουμε, μας δίνεται η ελευθερία να ομαδοποιήσουμε όπως θέλουμε εμείς τους κανόνες προκειμένου να δημιουργήσουμε μικρότερα loops που έχουμε τον απόλυτο έλεγχο (συγκριτικά με το main loop-all in one fsm). Προφανώς με αυτόν τον τρόπο, δίνοντας την έξοδο ενός σταδίου ως είσοδο σε άλλο στάδιο διαδοχικά, έχουμε όλα τα μειονεκτήματα που δημιουργούνται με το να διαβάζουμε ξανά και ξανά το input και να το "διορθώνουμε" κάθε φορά από λίγο, δηλαδή πιο αργοί χρόνοι εκτέλεσης και υψηλότερη χρήση μνήμης. Γιγάντιο βάρος πρέπει να δοθεί στο error handling, διότι αν κάποιο στάδιο δεν ολοκληρώσει έχουμε exit -1.

Ακόμη και ατομική αν ήταν η εργασία, θα επέλεγα για αρχή να υλοποιήσω το go-reloaded με την μέθοδο pipeline. Στα πλαίσια ομαδικής εργασία το pipeline είναι ιδανικό αφού, με ανάθεση διαφορετικών κανόνων σε κάθε μέλος, η ανάπτυξη του project γίνεται πολύ αποδοτική (integration των functions-αρχείων).