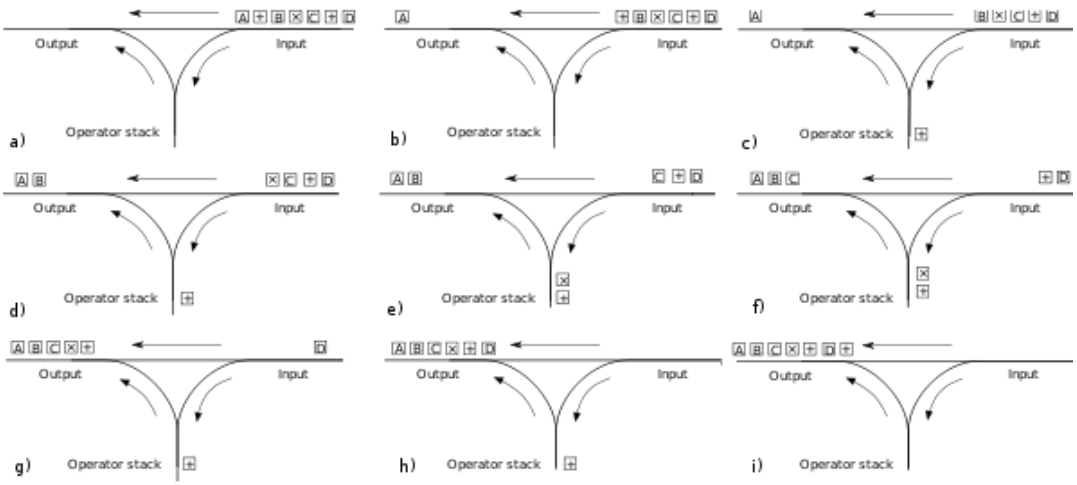
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

Ανάπτυξη εφαρμογής για τη μετατροπή Ενδοθεματικών (Infix) σε Μεταθεματικές (Postfix) παραστάσεις και τον υπολογισμό αυτών σε οπτικοποιημένο περιβάλλον



Πτυχιακή Εργασία της

Μαριάνθης Ζάμπα (1794)

Επιβλέπων : Ευάγγελος Γ. Ούτσιος, Καθηγητής Εφαρμογών

ΣΕΡΡΕΣ, ΜΑΙΟΣ 2012

Υπεύθυνη Δήλωση *: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.*

Περίληψη

Στόχος της παρούσας ατομικής πτυχιακής εργασίας ήταν η ανάπτυξη ενός λογισμικού που μετατρέπει μια έκφραση ενδοθεματικής μορφής (Infix notation) σε μεταθεματική μορφή (Postfix notation) και στη συνέχεια υπολογίζει το αποτέλεσμα ενώ ταυτόχρονα αναπαριστά με γραφικό τρόπο τις λειτουργίες μετατροπής και υπολογισμού βήμα - βήμα.

Για την υλοποίηση αυτού του λογισμικού μελετήθηκαν οι τρεις μορφές στις οποίες μπορεί να γραφεί μια αριθμητική έκφραση δηλαδή η ενδοθεματική μορφή (infix notation), η μεταθεματική μορφή (postfix notation) και η προθεματική μορφή (prefix notation). Επίσης μελετήθηκαν ο αλγόριθμος μετατροπής ενδοθεματικής έκφρασης σε μεταθεματική, ο αλγόριθμος υπολογισμού μεταθεματικών εκφράσεων και η δομή της στοίβας που είναι η βασική δομή δεδομένων για την υλοποίηση των δυο αλγορίθμων.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Ευχαριστίες

Με την ολοκλήρωση της παρούσας εργασίας θα ήθελα να ευχαριστήσω θερμά τον κ. Ευάγγελο Ούτσιο, καθηγητή Εφαρμογών του τμήματος Πληροφορικής και Επικοινωνιών του Τ.Ε.Ι. Σερρών και επιβλέποντα της πτυχιακής μου, για την εμπιστοσύνη που μου έδειξε με την ανάθεση της συγκεκριμένης εργασίας, για την καθοδήγηση που μου προσέφερε κατά την εκπόνησής της και για την υπομονή του.

Επίσης θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου για την ηθική και οικονομική συμπαράσταση όχι μόνο κατά τη διάρκεια της εκπόνησης της πτυχιακής μου αλλά και καθ’ όλη τη διάρκεια των σπουδών μου.

1. Εισαγωγή

Η νέα εκπαιδευτική πραγματικότητα επιβάλλει την υιοθέτηση από την πλευρά τόσο των εκπαιδευτικών όσο και των εκπαιδευόμενων, νέων διδακτικών προσεγγίσεων και μεθόδων διδασκαλίας.

Αξιοποιώντας νέες τεχνολογίες και εργαλεία μας δίνεται η δυνατότητα να βοηθήσουμε τον εκπαιδευόμενο να κατανοήσει πλήρως πολύπλοκα μαθηματικά μοντέλα. Έτσι με τη δημιουργία του παρόντος λογισμικού οπτικοποιούμε μαθηματικές μεθόδους απλοποιώντας σε σημαντικό βαθμό το ήδη δύσκολο έργο του διδάσκοντα. Παρ' όλα αυτά το έργο αποτελεί συμπληρωματικό βοήθημα στα υπάρχοντα συγγράμματα & παρεμφερή λογισμικά που σχετίζονται με τους αλγορίθμους μετατροπής ενδοθεματικών εκφράσεων σε μεταθεματικές και υπολογισμού μεταθεματικών εκφράσεων

2. Θεωρητικό Μέρος

Για να μπορέσουμε να υλοποιήσουμε ένα λογισμικό που θα σχετίζεται με την μετατροπή ενδοθεματικών εκφράσεων σε μεταθεματικές και τον υπολογισμό αυτών θα πρέπει πρώτα να εξετάσουμε σε θεωρητική βάση την στοίβα, τις αριθμητικές εκφράσεις, τις μορφές συγγραφής των αριθμητικών εκφράσεων και τους αλγορίθμους που σχετίζονται άμεσα με το θέμα μας.

2.1. Στοίβα

Η δομή της στοίβας είναι η βασική δομή δεδομένων για την υλοποίηση των αλγορίθμων μετατροπής ενδοθεματικών εκφράσεων σε μεταθεματικές και υπολογισμού μεταθεματικών εκφράσεων.

2.1.1. Τι είναι Στοίβα;

Στοίβα είναι μια στατική δομή δεδομένων της οποία τα δεδομένα που βρίσκονται στην κορυφή της λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στη βάση της λαμβάνονται τελευταία. Αυτή η μέθοδος επεξεργασίας ονομάζεται Τελευταίο μέσα, πρώτο έξω ή απλούστερα με την αγγλική συντομογραφία LIFO(Last-In-First-Out).

2.1.2. Λειτουργίες

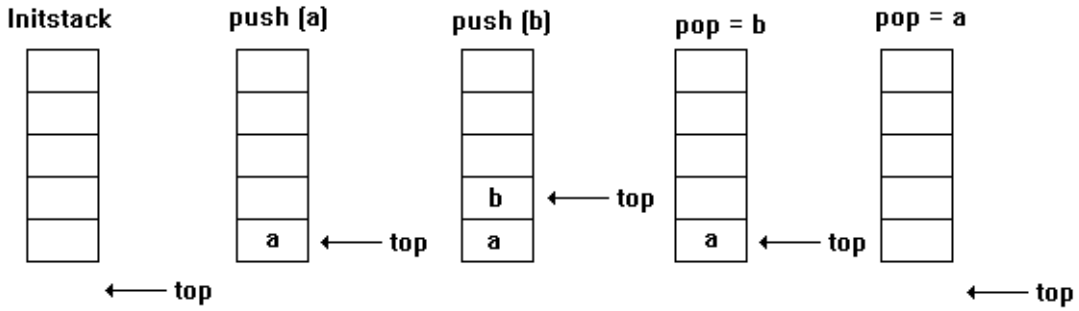
Δύο είναι οι κύριες λειτουργίες σε μία στοίβα:

- η ώθηση (push) στοιχείου στην κορυφή της στοίβας, και
- η απώθηση (pop) στοιχείου από τη στοίβα.

Η διαδικασία της ώθησης πρέπει οπωσδήποτε να ελέγχει, αν η στοίβα είναι γεμάτη, οπότε λέγεται ότι συμβαίνει υπερχείλιση (overflow) της στοίβας. Αντίστοιχα, η διαδικασία απώθησης ελέγχει, αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα, δηλαδή ελέγχει αν γίνεται υποχείλιση (underflow) της στοίβας.

2.1.3. Υλοποίηση

Μια στοίβα μπορεί να υλοποιηθεί πολύ εύκολα με τη βοήθεια ενός μονοδιάστατου πίνακα. Μια βοηθητική μεταβλητή (με όνομα συνήθως top) χρησιμοποιείται για να δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας. Για την εισαγωγή ενός νέου στοιχείου στη στοίβα (ώθηση) αρκεί να αυξηθεί η μεταβλητή top κατά ένα και στη θέση αυτή να εισέλθει το στοιχείο. Αντίθετα για την εξαγωγή ενός στοιχείου από τη στοίβα (απώθηση) εξέρχεται πρώτα το στοιχείο που δείχνει η μεταβλητή top και στη συνέχεια η top μειώνεται κατά ένα για να δείχνει τη νέα κορυφή.



Εικόνα 1 Η Στοίβα

2.1.4. Εφαρμογές

Οι στοίβες έχουν πολλές εφαρμογές. Βλέπουμε στοίβες στην καθημερινή μας ζωή, από τα βιβλία στη βιβλιοθήκη μας εως τη δέσμη φύλλων που διατηρούμε στη θήκη του εκτυπωτή μας. Όλα αυτά ακολουθούν την Last In First Out (LIFO) λογική, δηλαδή όταν προσθέτουμε ένα βιβλίο σε ένα σωρό από βιβλία, θα το προσθέσουμε στην κορυφή του σωρού, ενώ όταν αφαιρούμε ένα βιβλίο από το σωρό, θα το αφαιρέσουμε από την κορυφή του σωρού.

Μερικές εφαρμογές της στοίβας στον κόσμο των υπολογιστών είναι οι παρακάτω:

- Κλήση υποπρογραμμάτων
- Αναδρομικές τεχνικές

- Μετατροπή ενός δεκαδικού αριθμού σε δυαδικό αριθμό
- Πύργοι του Ανόι (Towers of Hanoi)
- Επεξεργασία και Υπολογισμός Αριθμητικών Εκφράσεων
- Μετατροπή μιας έκφρασης Infix σε έκφραση Postfix
- Quicksort

Παρακάτω θα αναλύσουμε την επεξεργασία και τον υπολογισμό αριθμητικών εκφράσεων.

2.2. Εκφράσεις

Όταν μια τιμή προκύπτει από υπολογισμό τότε αναφερόμαστε σε εκφράσεις. Οι τελεστές μαζί με τους τελεστέους διαμορφώνουν τις εκφράσεις. Η διεργασία αποτίμησης μιας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση των πράξεων. Οι εκφράσεις χωρίζονται σε δύο κατηγορίες: τις αριθμητικές και τις λογικές.

2.2.1. Αριθμητικές Εκφράσεις

Για την σύνταξη μιας αριθμητικής έκφρασης χρησιμοποιούνται:

- Σταθερές
  - Π.χ. 13,7,45
- Μεταβλητές
  - Π.χ. x,y
- Τελεστές
  - + : Πρόσθεση
  - – : Αφαίρεση
  - * : Πολλαπλασιασμός
  - / : Διαίρεση
  - ^ : Ύψωση στην δύναμη
  - Div : Ακέραια διαίρεση
  - Mod : Υπόλοιπο ακέραιας διαίρεσης

- Παρενθέσεις
    - (: Άνοιγμα παρένθεσης
    -) : Κλείσιμο παρένθεσης

Κάθε έκφραση παριστάνει μία αριθμητική τιμή η οποία βρίσκεται μετά την εκτέλεση των πράξεων. Γι' αυτό είναι απαραίτητο όλες οι μεταβλητές που εμφανίζονται σε μία έκφραση να έχουν πάρει προηγούμενα κάποια τιμή.

Η εκτέλεση των πράξεων γίνεται με βάση την ιεραρχία των αριθμητικών τελεστών.

Η ιεραρχία είναι:

1. ^
2. *, /, div, mod
3. +, -

Αν συναντάμε παρενθέσεις προηγούνται οι τελεστές εντός των παρενθέσεων. Όσοι τελεστές έχουν ίδια ιεραρχία εκτελούνται από αριστερά προς τα δεξιά.

2.2.2. Σημειογραφία Αριθμητικών Εκφράσεων

Στις γλώσσες προγραμματισμού υπάρχει η δυνατότητα να επιτυγχάνονται διαφορετικές λειτουργίες στην ίδια έκφραση ανάλογα με τη θέση των τελεστών ανάμεσα στους τελεστέους. Για το λόγο αυτό έχουν αναπτυχθεί τρεις σημειογραφίες για τους δυαδικούς τελεστές:

- Ενδοθεματική μορφή (Infix notation), ο τελεστής (operator) τοποθετείται μεταξύ των τελεστέων (operands), όπως στην έκφραση x+y.
- Προθεματική μορφή (Prefix notation), ο τελεστής (operator) τοποθετείται πριν από τους τελεστέους (operands), όπως στην έκφραση +xy.

- Μεταθεματική μορφή (Postfix notation), ο τελεστής τοποθετείται μετά από τους τελεστέους, όπως στην έκφραση xy+.

2.2.3. Ενδοθεματική μορφή (Infix notation)

Η έκφραση ενδοθεματικής μορφής είναι η κοινή αριθμητική έκφραση κατά την οποία οι τελεστές τοποθετούνται μεταξύ των τελεστέων π.χ. 3+4*5.

Κάθε έκφραση ενδοθεματικής μορφής χρησιμοποιεί προτεραιότητες στη σειρά εκτέλεσης των πράξεων. Π.χ. στην παράσταση 3+4*5 έχει προτεραιότητα η πράξη του πολλαπλασιασμού. Φυσικά οι προτεραιότητες τροποποιούνται με τη χρήση παρενθέσεων, π.χ. 2*(3+4) προτεραιότητα έχει η πράξη μέσα στις παρενθέσεις, δηλαδή η πρόσθεση.

Αν όμως ο μεταφραστής προσπαθήσει να εκτελέσει μία παράσταση στη ενδοθεματική μορφή της, όπως π.χ. την 3+4*5, όταν φθάσει στον τελεστή της πρόσθεσης δεν γνωρίζει αν θα πρέπει να εκτελέσει την πρόσθεση ή να την αναβάλει για αργότερα, γιατί πιθανόν να προηγείται κάποια άλλη πράξη. Για το λόγο αυτό ο μεταφραστής ακολουθεί τις ακόλουθες δύο διαδικασίες για το σωστό υπολογισμό των εκφράσεων:

- Η ενδοθεματική μορφή της αριθμητικής έκφρασης μεταφράζεται σε μεταθεματική (postfix) μορφή, όπου οι τελεστές εμφανίζονται μετά τους τελεστέους. Π.χ:
    - η παράσταση 2+3 γίνεται 23+,
    - και η παράσταση 3+4*5 μετατρέπεται σε 345*+
- Η μεταθεματική μορφή χρησιμοποιείται κατόπιν για τον υπολογισμό της έκφρασης.

2.2.4. Προθεματική μορφή (Prefix notation)

Η έκφραση προθεματικής μορφής (prestfix notation) ή πολωνικός συμβολισμός (polish notation) είναι η αριθμητική έκφραση κατά την οποία οι τελεστές συνίσταται στο να τοποθετούνται πριν από τους τελεστέους χωρίς να γίνετε χρήση παρενθέσεων.

Για παράδειγμα αντί για τις παραστάσεις:

- α+β γράφουμε +αβ
- (α+β)*γ γράφουμε: *+αβγ
- α+(β*γ) γράφουμε: +α*βγ
- α*β+γ*δ γράφουμε: +*αβ*γδ

Στον πολωνικό συμβολισμό η χρήση παρενθέσεων δεν είναι απαραίτητη με την προϋπόθεση ότι κάθε τελεστής έχει συγκεκριμένο αριθμό τελεστέων. Έτσι η έκφραση +αβ έχει το ίδιο νόημα με την α+β, αφού γνωρίζουμε ότι ο τελεστής + έχει δύο τελεστέους που ακολουθούν τον τελεστή.

Ο συμβολισμός αυτό προτάθηκε το 1951 από τον Πολωνό μαθηματικό Jan Lukasiewicz εφαρμόστηκε στην άλγεβρα καθώς και σε άλλα συστήματα τελεστών-τελεστέων και έγινε γνωστός με το όνομα πολωνικός συμβολισμός (Polish notation) προς τιμή του Πολωνού μαθηματικού.

2.2.5. Μεταθεματική μορφή (Postfix notation)

Η έκφραση μεταθεματικής μορφής (postfix notation) ή Αντίστροφη Πολωνική Μορφή (Reverse Polish Notation – RPN) είναι η αριθμητική έκφραση κατά την οποία οι τελεστές συνίσταται στο να τοποθετούνται μετά από τους τελεστέους χωρίς να γίνετε χρήση παρενθέσεων.

Για παράδειγμα αντί για τις παραστάσεις:

- α+β γράφουμε αβ+

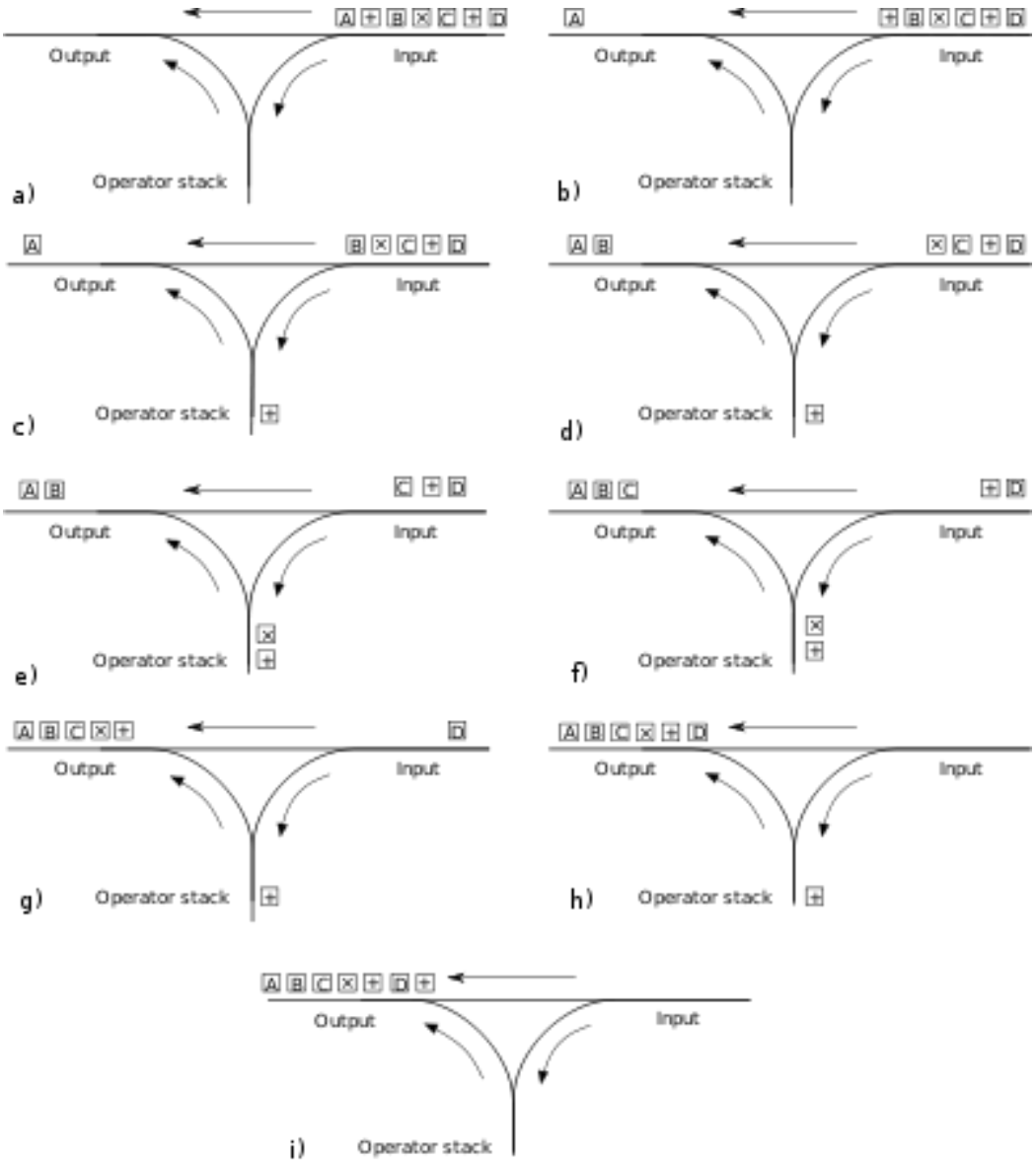
- (α+β)*γ γράφουμε αβ+γ*
- α+(β*γ) γράφουμε αβγ*+
- α*β+γ*δ γράφουμε αβ*γδ*+

Παραστάσεις γραμμένες σε αντίστροφη πολωνική μορφή μπορούν να υπολογιστούν πολύ εύκολα από τα αριστερά προς τα δεξιά, αρκεί όταν συναντάται κάποιος τελεστής να γίνεται η αντίστοιχη πράξη με τελεστέους, αυτούς που υπολογίστηκαν πιο πριν.

2.3. Αλγόριθμοι

2.3.1. Μετατροπή ενδοθεματικής μορφής σε μεταθεματική

Για τη μετατροπή μιας έκφρασης ενδοθεματικής μορφής σε μεταθεματική χρησιμοποιείτε ο αλγόριθμος Shunting-yard που εφευρέθηκε από τον Edsger Dijkstra.



Εικόνα 2 Γραφική απεικόνιση του αλγορίθμου Shunting-yard

Η διαδικασία που ακολουθείται είναι η εξής:

- Αν το στοιχείο είναι τελεστέος, τότε τοποθετείται στα δεξιά της μεταθεματικής μορφής.
- Αν το στοιχείο είναι τελεστής, τότε κατευθύνεται στη στοίβα. Εκεί συγκρίνεται με τον τελεστή της κορυφής της στοίβας. Αν ο εισερχόμενος τελεστής έχει μεγαλύτερη προτεραιότητα από τον τελεστή της κορυφής της στοίβας, τότε ο τελεστής τοποθετείται στη στοίβα (push). Αν η προτεραιότητα του εισερχόμενου τελεστή

είναι μικρότερη ή ίση από την προτεραιότητα του τελεστή της κορυφής της στοίβας, τότε:

- Εξάγονται όλοι οι τελεστές της στοίβας με προτεραιότητα μεγαλύτερη ή ίση από την προτεραιότητα του νέου τελεστή (pop) και τοποθετούνται στα δεξιά της μεταθεματικής μορφής.
- Ο νέος τελεστής τοποθετείται στη στοίβα (push).

Όταν η αριθμητική έκφραση έχει παρενθέσεις, τότε ακολουθούνται οι παρακάτω κανόνες:

- Η αριστερή παρένθεση τοποθετείται αμέσως στη στοίβα (push).
- Η δεξιά παρένθεση προκαλεί την εξαγωγή (pop) όλων των τελεστών μέχρι να συναντηθεί η αριστερή παρένθεση στη στοίβα.

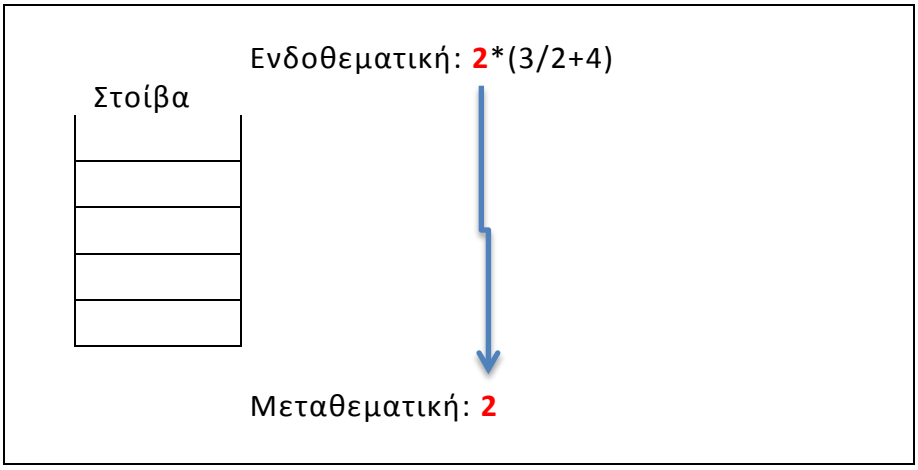
Ύστερα, οι δύο παρενθέσεις αγνοούνται.

2.3.1.1. Παράδειγμα μετατροπής

Η παράσταση 2*(3/2+4) μετατρέπεται σε 232/4+*

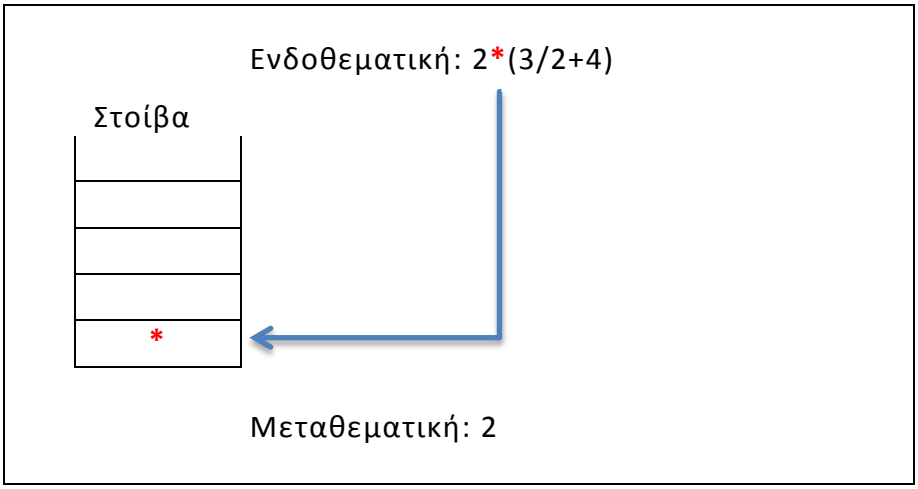
- *1ο βήμα:*

  Το πρώτο στοιχείο είναι τελεστέος οπότε τοποθετείται στα δεξιά της μεταθεματικής μορφής



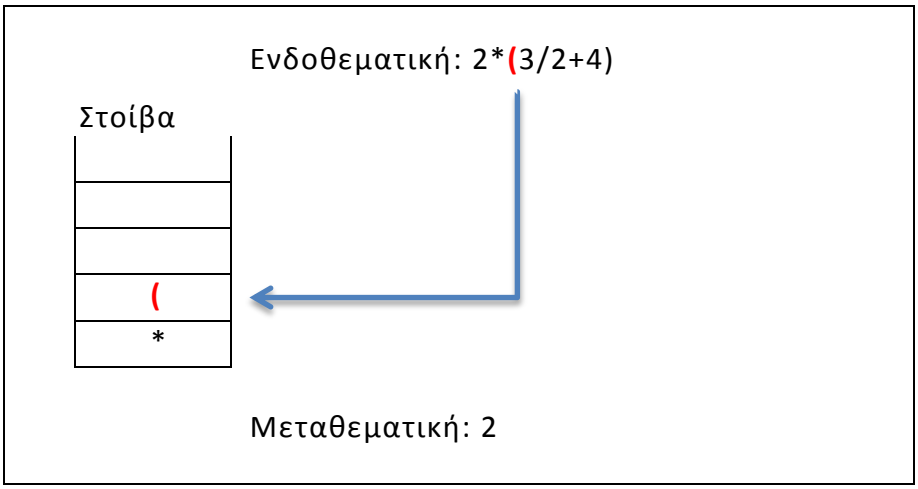
- *2ο βήμα:*

  Το επόμενο στοιχείο είναι τελεστής, επειδή η στοίβα είναι κενή, τοποθετείται αμέσως στη στοίβα (push)



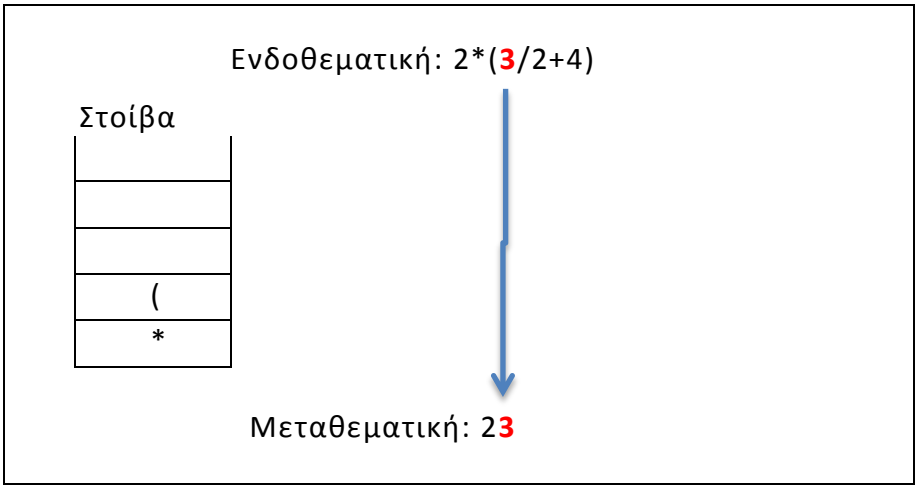
- *3ο βήμα:*

  Το επόμενο στοιχείο είναι αριστερή παρένθεση οπότε τοποθετείται στη στοίβα (push)



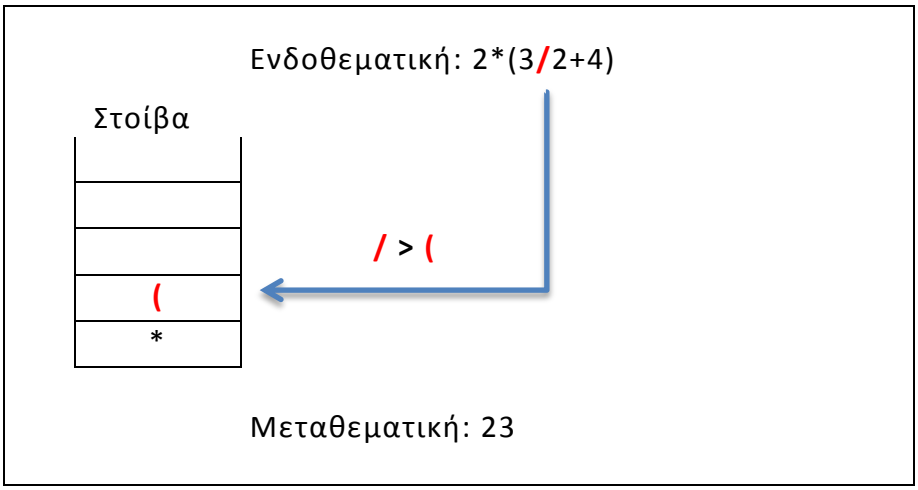
- *4ο βήμα:*

  Το επόμενο στοιχείο είναι τελεστέος οπότε τοποθετείται στα δεξιά της μεταθεματικής μορφής



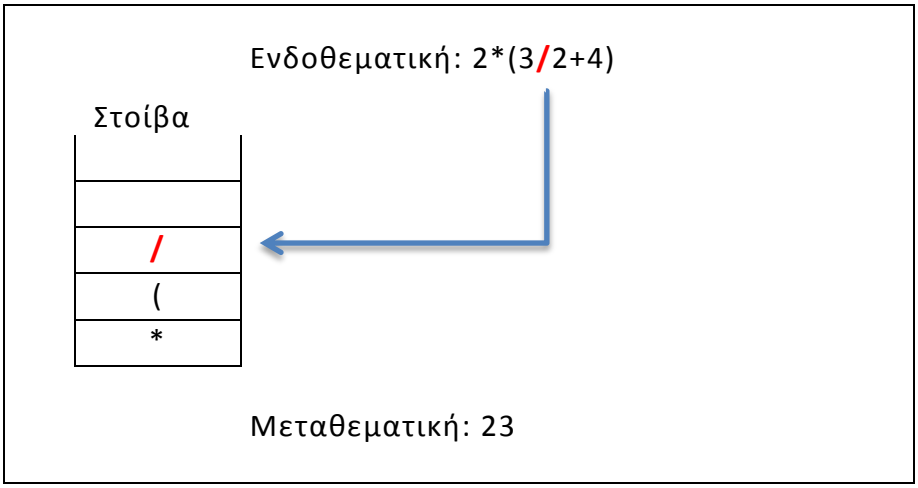
- *5ο βήμα*

  Το επόμενο στοιχείο είναι τελεστής και συγκρίνεται με τον τελεστή της κορυφής της στοίβας. Ο εισερχόμενος τελεστής έχει μεγαλύτερη προτεραιότητα από τον τελεστή της κορυφής της στοίβας



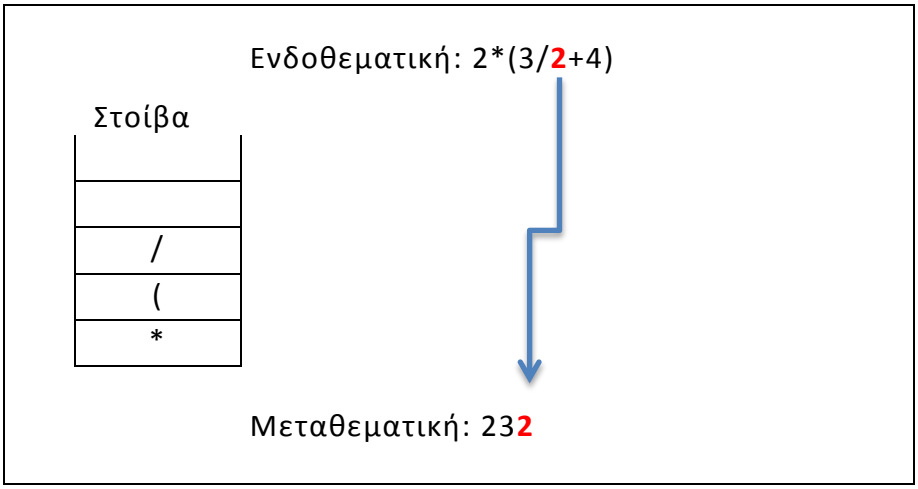
- *6ο βήμα*

  Ο τελεστής τοποθετείται στη στοίβα (push)



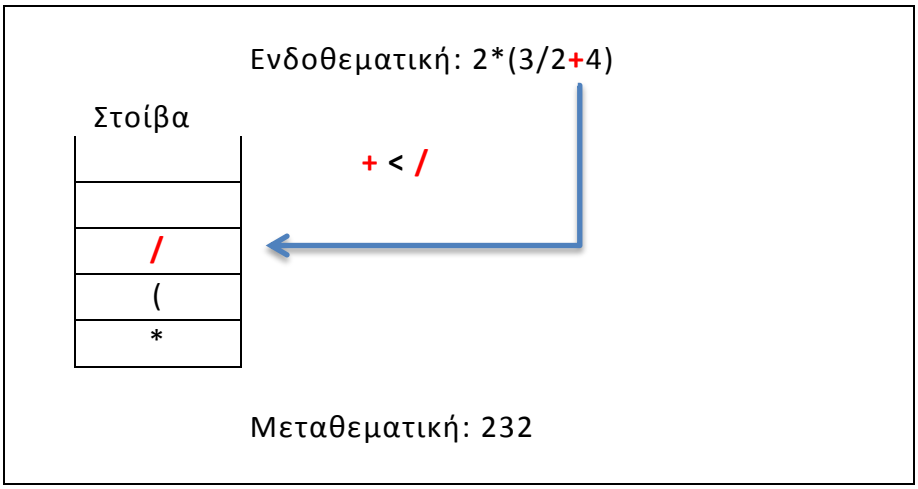
- *7ο βήμα*

  Το επόμενο στοιχείο είναι τελεστέος οπότε τοποθετείται στα δεξιά της μεταθεματικής μορφής



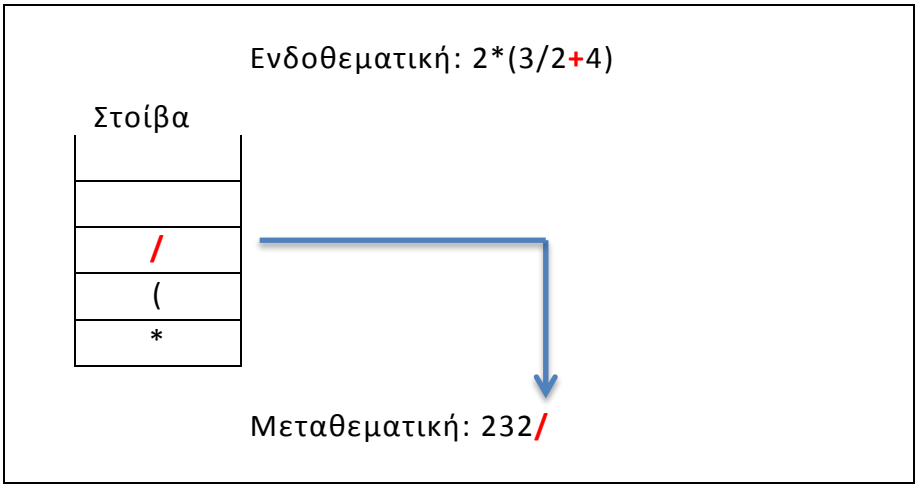
- *8ο βήμα*

  Το επόμενο στοιχείο είναι τελεστής και συγκρίνεται με τον τελεστή της κορυφής της στοίβας. Ο εισερχόμενος τελεστής έχει μικρότερη προτεραιότητα από τον τελεστή της κορυφής της στοίβας



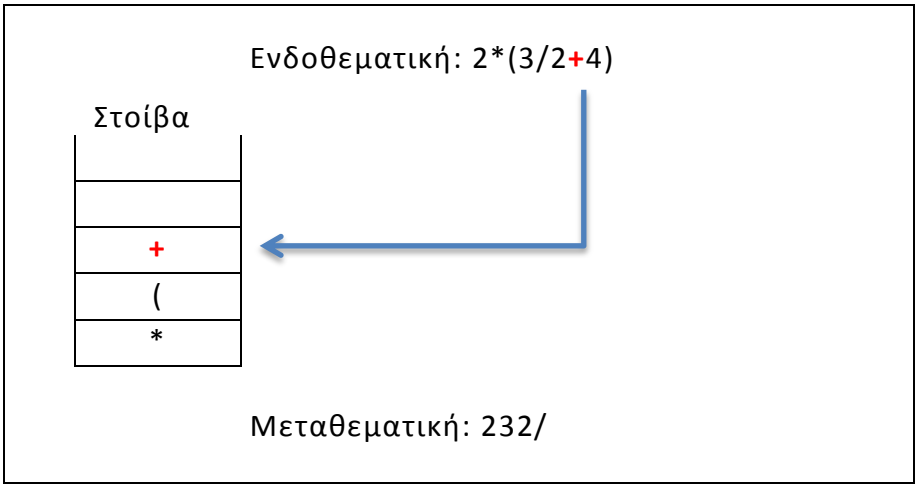
- *9ο βήμα*

  Εξάγεται (pop) ο τελεστής της στοίβας με προτεραιότητα μεγαλύτερη ή ίση από την προτεραιότητα του νέου τελεστή και τοποθετείται στα δεξιά της μεταθεματικής μορφής



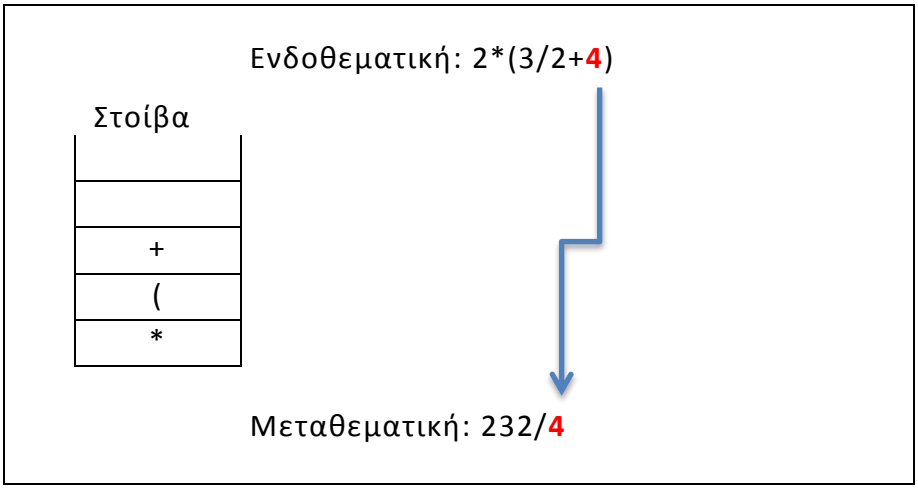
- *$10^{ο}$ βήμα*

  Ο νέος τελεστής τοποθετείται στη στοίβα (push)



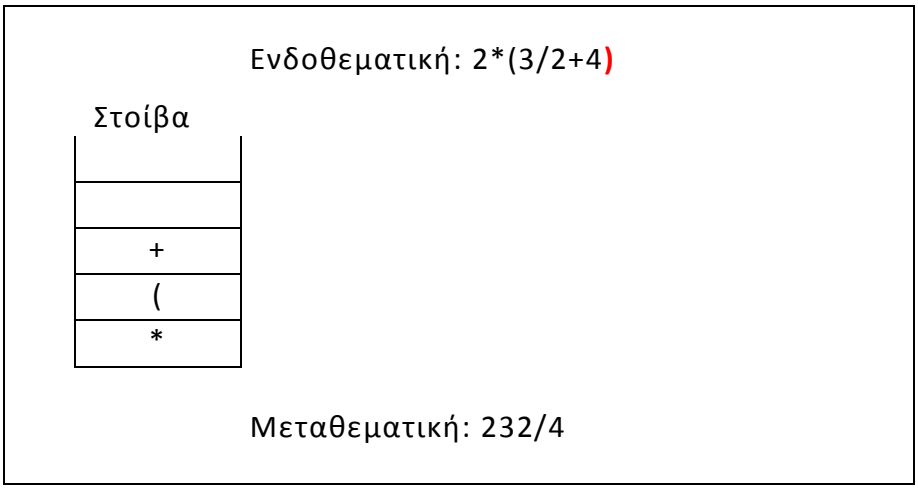
- *$11^{ο}$ βήμα*

  Το επόμενο στοιχείο είναι τελεστέος οπότε τοποθετείται στα δεξιά της μεταθεματικής μορφής



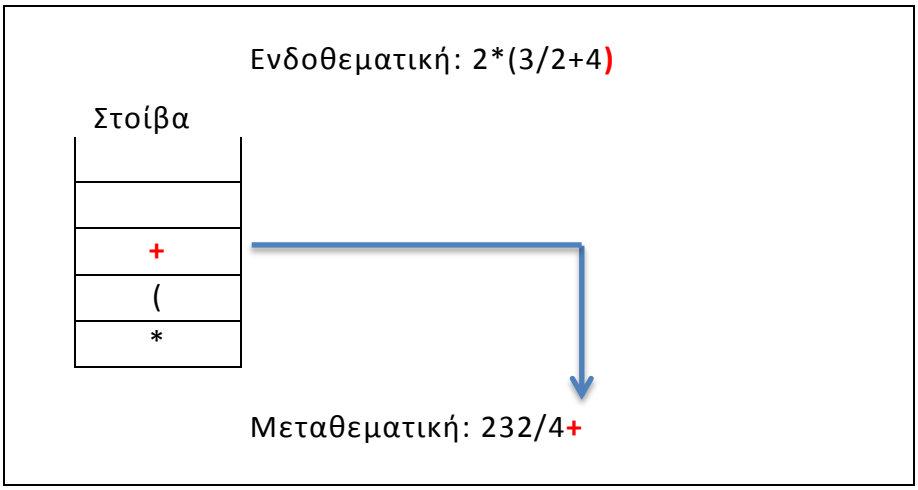
- *12^{ο} βήμα*

  Το επόμενο στοιχείο δεξιά παρένθεση, αυτό προκαλεί την εξαγωγή (pop) όλων των τελεστών μέχρι να συναντηθεί η αριστερή παρένθεση στη στοίβα. Ύστερα, οι δύο παρενθέσεις αγνοούνται.



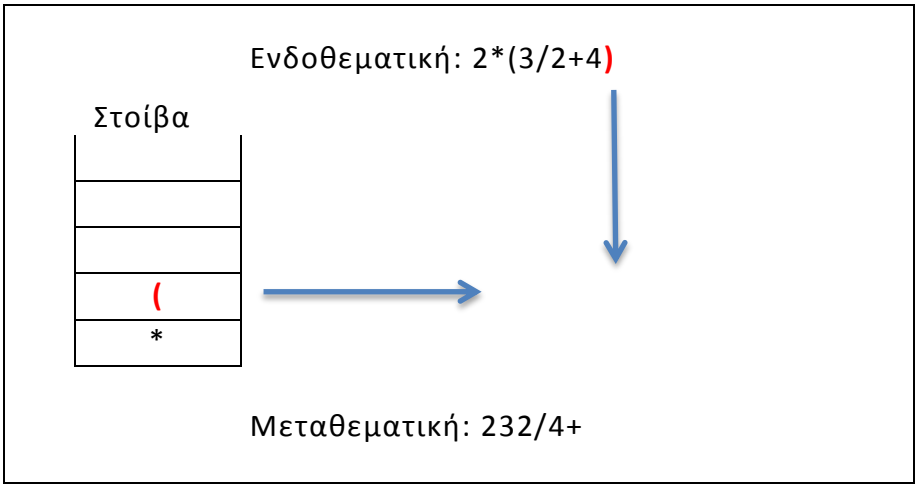
- *13^{ο} βήμα*

  Εξάγονται (pop) όλοι οι τελεστές μέχρι να συναντηθεί η αριστερή παρένθεση στη στοίβα.



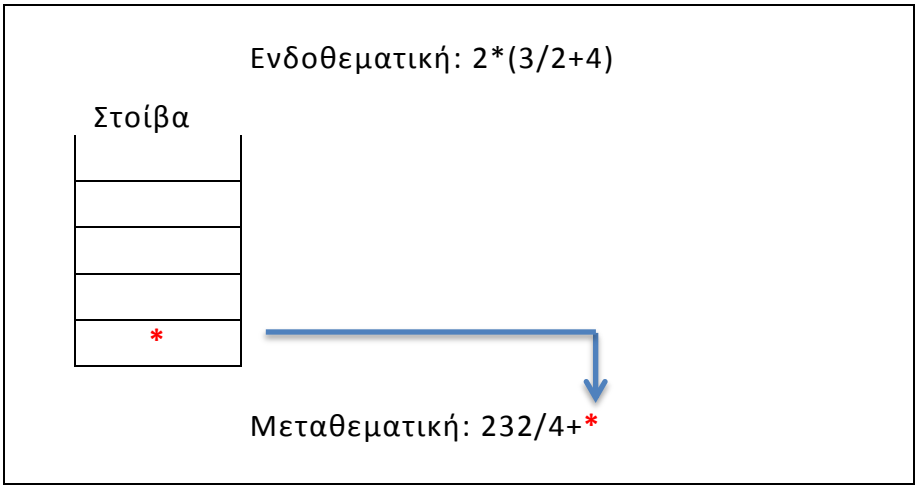
- *$14^{ο}$ βήμα*

  Οι δύο παρενθέσεις αγνοούνται



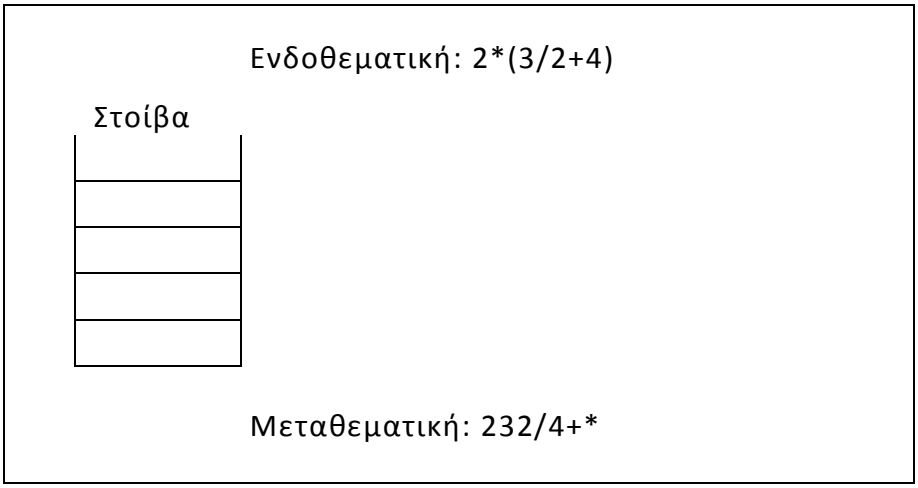
- *$15^{ο}$ βήμα*

  Τέλος εξάγεται (pop) ο τελευταίος τελεστής της στοίβας και τοποθετείται στα δεξιά της μεταθεματικής μορφής



- *$16^{ο}$ βήμα*

  Επιστρέφεται η μεταθεματική μορφή της έκφρασης



2.3.2. Υπολογισμός μεταθεματικής έκφρασης

Η διαδικασία υπολογισμού της αριθμητικής έκφρασης συνίσταται στη σάρωση της μεταθεματικής μορφής, με την εφαρμογή των ακόλουθων δύο κανόνων:

- Αν το στοιχείο είναι τελεσταίος, τότε τοποθετείται στη στοίβα (push).
- Αν το στοιχείο είναι τελεστής, τότε:
  - Εξάγονται διαδοχικά δύο τελεσταίοι από την κορυφή της στοίβας (pop)
  - Εκτελείται η πράξη που δηλώνει ο τελεστής
  - Το αποτέλεσμα τοποθετείται στη στοίβα (push)

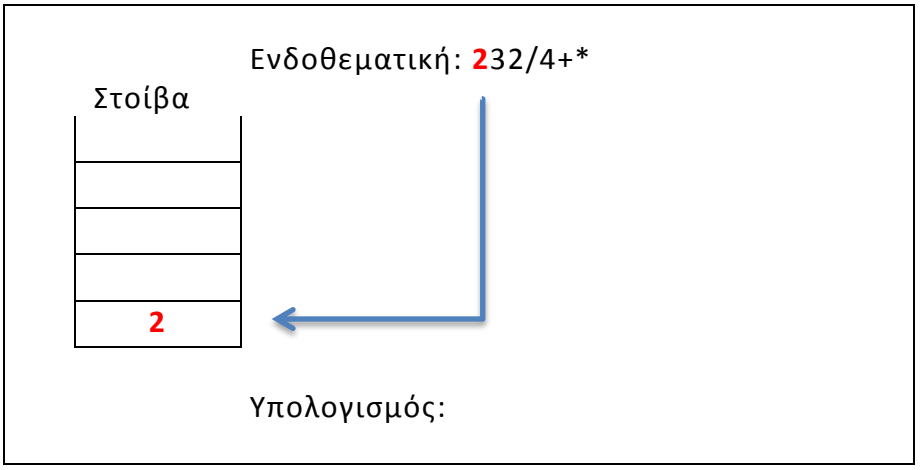
Όταν τελειώσει η σάρωση της μεταθεματικής μορφής, στη στοίβα έχει απομείνει το τελικό αποτέλεσμα, που είναι η τιμή της αριθμητικής παράστασης.

2.3.2.1. Παράδειγμα υπολογισμού

Η παράσταση 232/4+* δίνει το αποτέλεσμα 11 και υπολογίζεται με τον παρακάτω τρόπο:

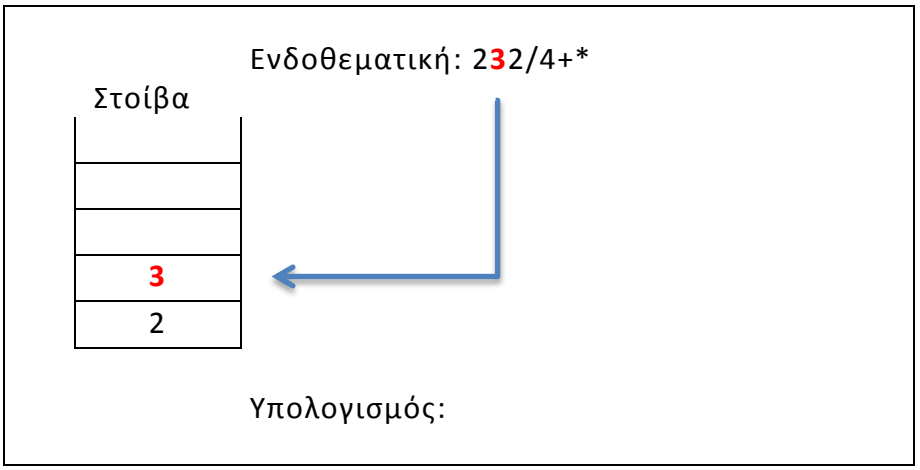
- *1^{ο} βήμα:*

  Το πρώτο στοιχείο είναι τελεστέος οπότε τοποθετείται στη στοίβα (push)



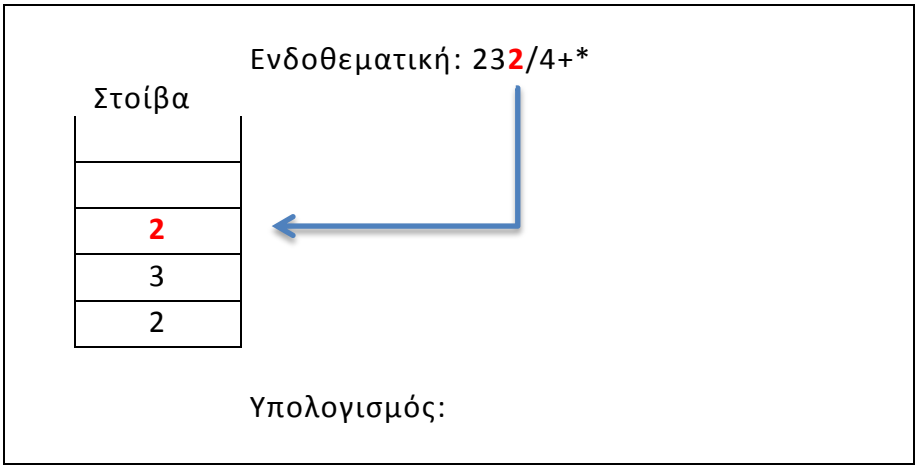
- *2^{ο} βήμα:*

  Το επόμενο στοιχείο είναι τελεστέος οπότε τοποθετείται στη στοίβα (push)



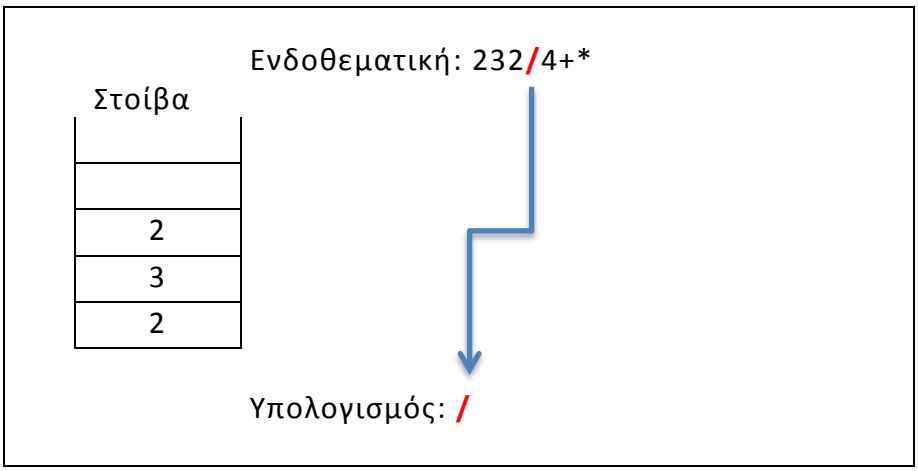
- *3ο βήμα:*

  Το επόμενο στοιχείο είναι τελεστέος οπότε τοποθετείται στη στοίβα (push)



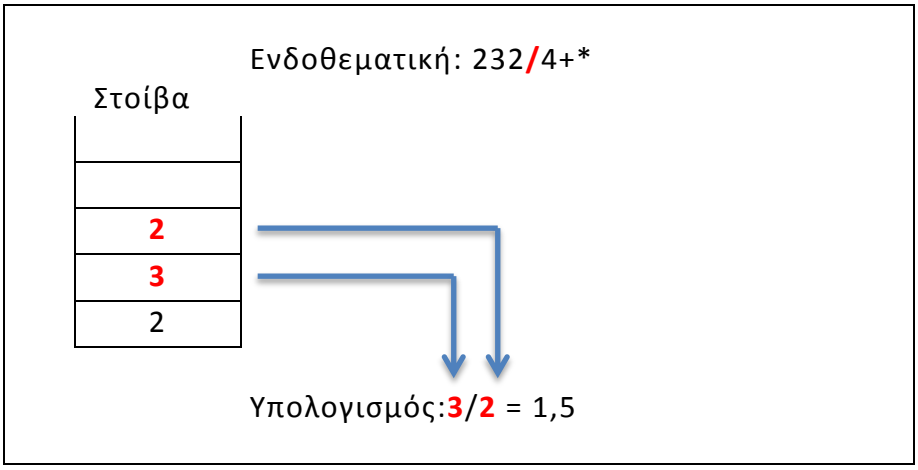
- *4ο βήμα:*

  Το επόμενο στοιχείο είναι τελεστής



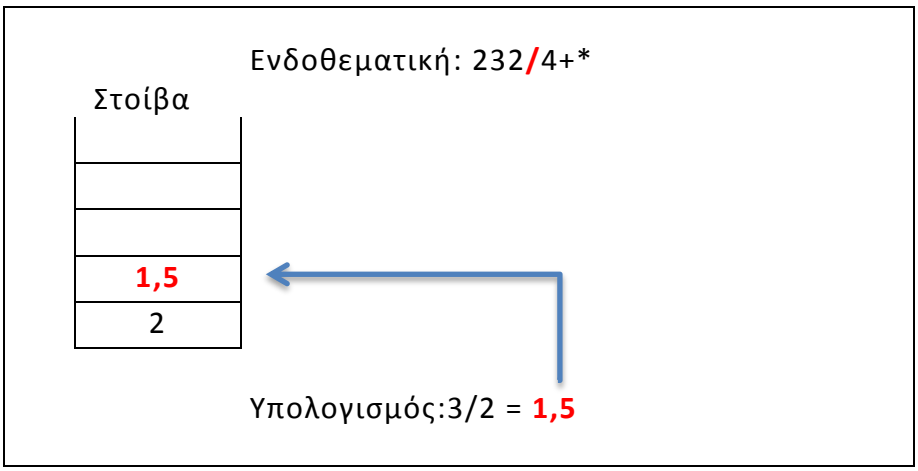
- *5ο βήμα:*

  Εξάγονται διαδοχικά δύο τελεσταίοι από την κορυφή της στοίβας (pop) και εκτελείται η πράξη που δηλώνει ο τελεστής



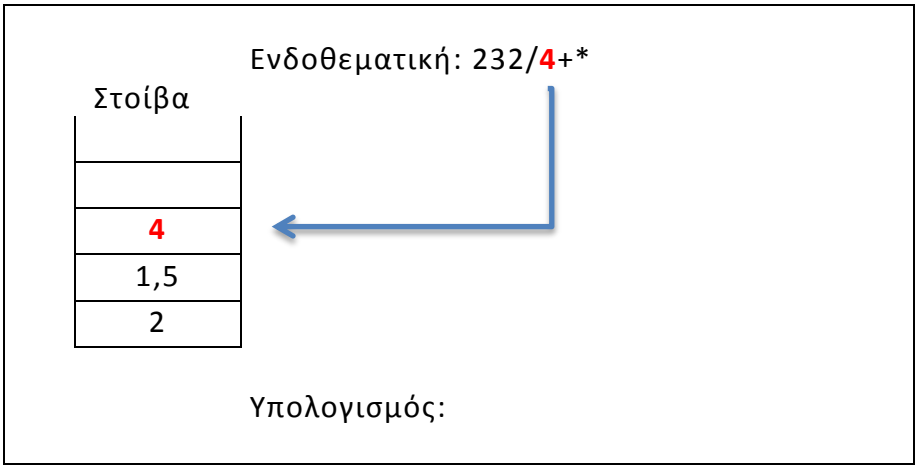
- *6ο βήμα:*

  Το αποτέλεσμα της πράξης τοποθετείται στη στοίβα (push)



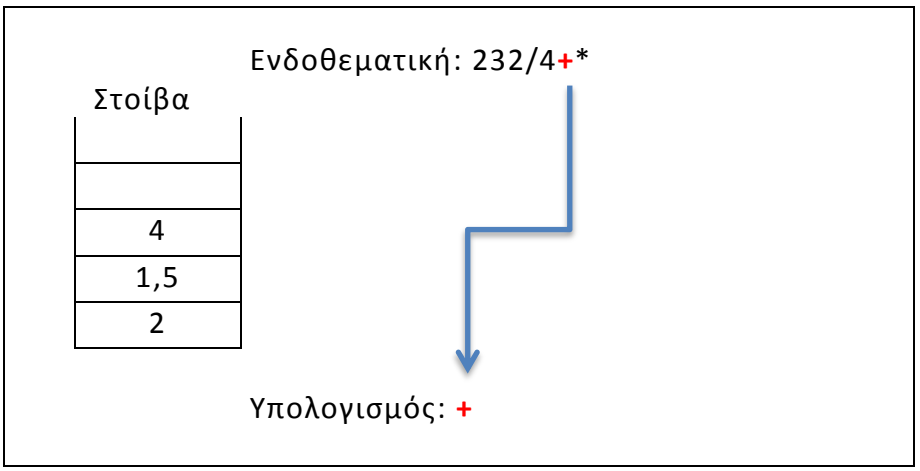
- *7ο βήμα:*

  Το επόμενο στοιχείο είναι τελεστέος οπότε τοποθετείται στη στοίβα (push)



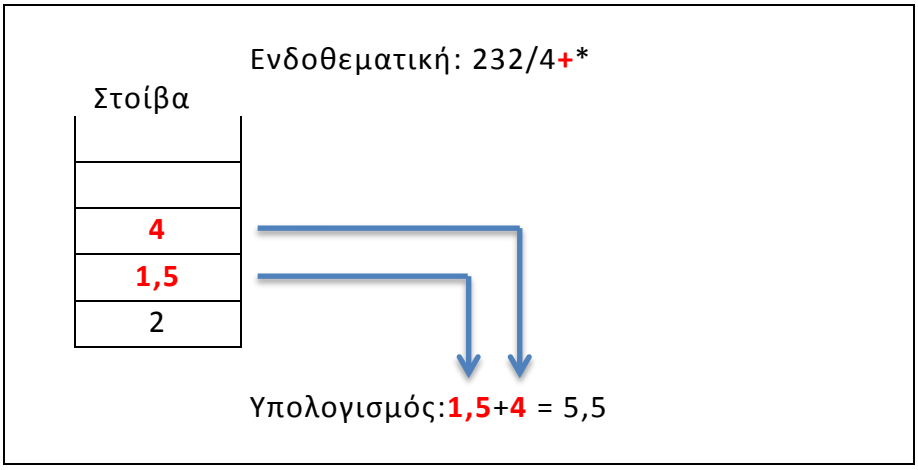
- *8ο βήμα:*

  Το επόμενο στοιχείο είναι τελεστής



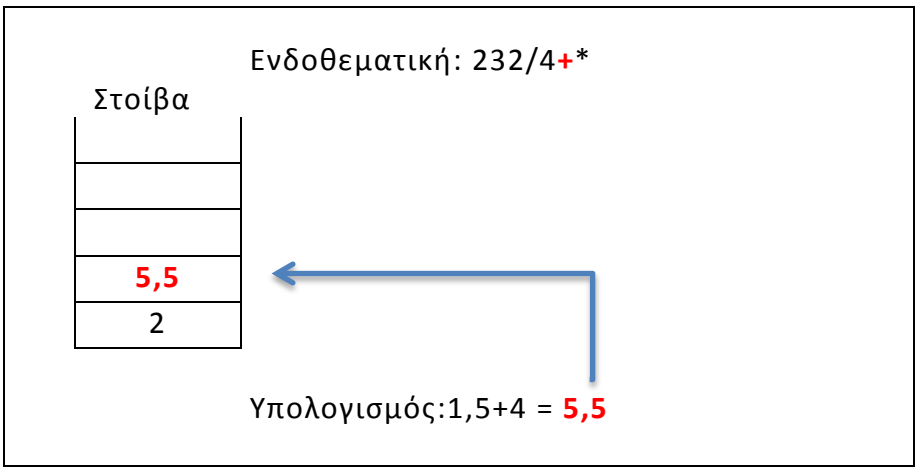
- *9^{ο} βήμα:*

  Εξάγονται διαδοχικά δύο τελεσταίοι από την κορυφή της στοίβας (pop) και εκτελείται η πράξη που δηλώνει ο τελεστής



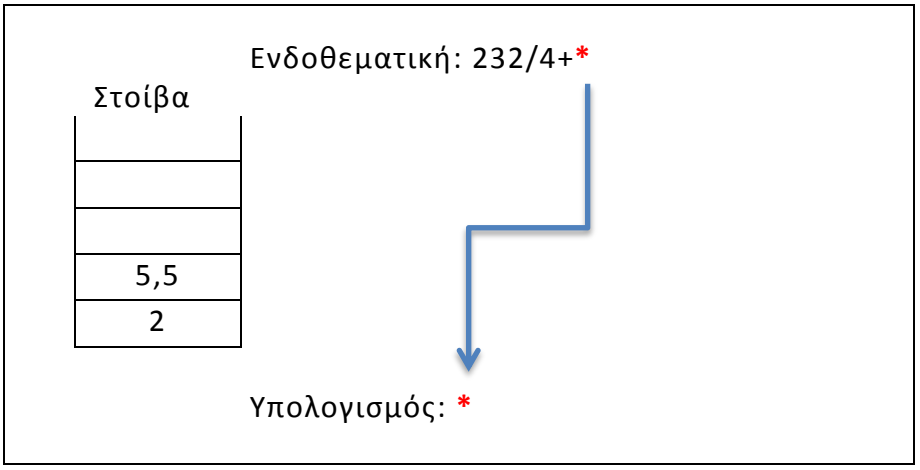
- *10^{ο} βήμα:*

  Το αποτέλεσμα της πράξης τοποθετείται στη στοίβα (push)



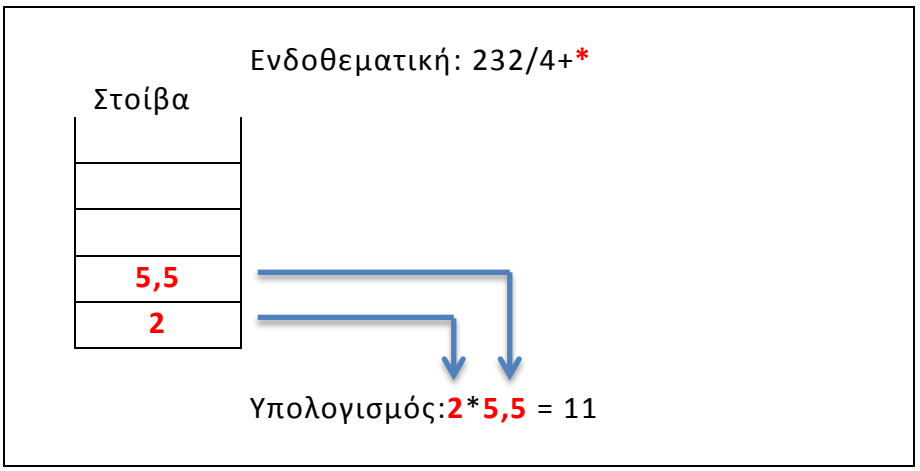
- *$11^{ο}$ βήμα:*

  Το επόμενο στοιχείο είναι τελεστής



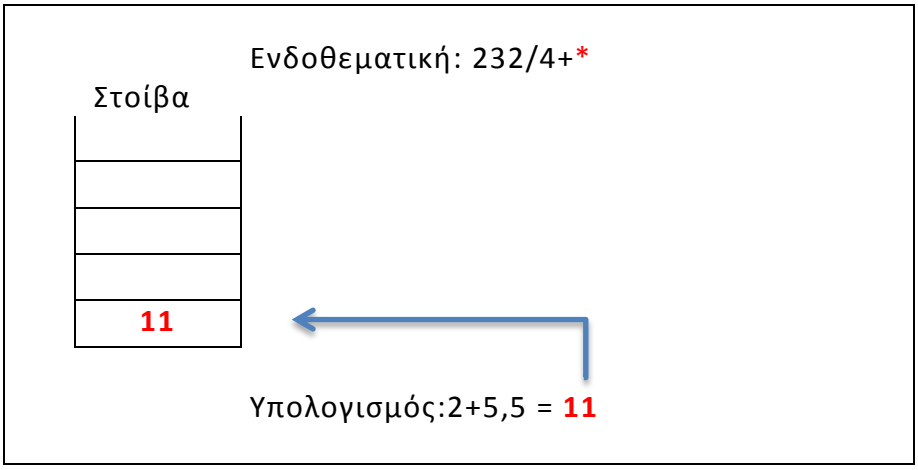
- *$12^{ο}$ βήμα:*

  Εξάγονται διαδοχικά δύο τελεσταίοι από την κορυφή της στοίβας (pop) και εκτελείται η πράξη που δηλώνει ο τελεστής



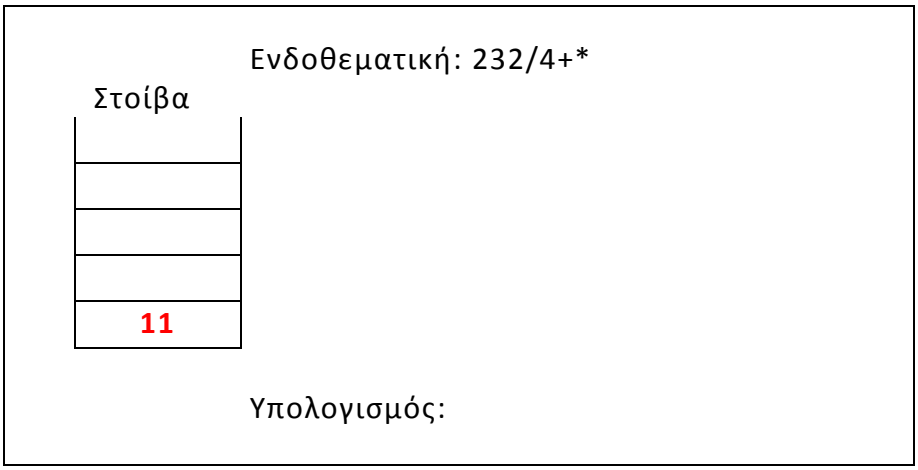
- *13^{ο} βήμα:*

  Το αποτέλεσμα της πράξης τοποθετείται στη στοίβα (push)



- *14^{ο} βήμα:*

  Έχει τελειώσει η σάρωση της μεταθεματικής μορφής και στη στοίβα έχει απομείνει το τελικό αποτέλεσμα που είναι η τιμή της αριθμητικής παράστασης



3. Εργαλεία Σχεδίασης και Ανάπτυξης

3.1. Διαδίκτυο

Το Διαδίκτυο ή Internet είναι ένα επικοινωνιακό δίκτυο, που επιτρέπει την ανταλλαγή δεδομένων μεταξύ οποιουδήποτε διασυνδεδεμένου υπολογιστή. Η τεχνολογία του είναι κυρίως βασισμένη στην διασύνδεση επιμέρους δικτύων ανά τον κόσμο και πολυάριθμα τεχνολογικά πρωτόκολλα. Στην πιο εξειδικευμένη και περισσότερο χρησιμοποιούμενη μορφή του, με τους όρους Διαδίκτυο, (με κεφαλαίο το αρχικό γράμμα) περιγράφεται το παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών και των υπηρεσιών και πληροφοριών που παρέχει στους χρήστες του.

Το Internet, σε συνδυασμό με την ολοένα αναπτυσσόμενη ψηφιακή τεχνολογία, έχει δημιουργήσει μία τεράστια αγορά γνώσεων/πληροφοριών και επηρέασε σημαντικά τον τρόπο διάθεσής τους.

3.2. Visual Studio 2010

Το Visual Studio 2010, είναι ένα εύχρηστο περιβάλλον ανάπτυξης εφαρμογών, που αναπτύχθηκε από την εταιρεία Microsoft Corporation. Υποστηρίζει την ανάπτυξη προγραμμάτων σε κονσόλα, οπτικές εφαρμογές, ιστοσελίδες, υπηρεσίες WEB και άλλα.

Το περιβάλλον ανάπτυξης Visual Studio 2010 βοηθά τον προγραμματιστή να αναπτύξει τα προγράμματά του με σχετική ευκολία, καθώς η τεχνολογία Microsoft IntelliSense βοηθά τον προγραμματιστή να κατανοήσει με ευκολία πιθανά λάθη του κώδικά του, υπογραμμίζοντας τα με κόκκινη γραμμή δυναμικά κατά το χρόνο συγγραφής του προγράμματος. Αυτή η τεχνολογία είναι ικανή να εντοπίσει λάθη τα οποία μπορεί να είναι είτε συντακτικά, όπως για

παράδειγμα η χρήση μιας εντολής με εσφαλμένο τρόπο, είτε λογικά, όπως για παράδειγμα η δήλωση ενός αντικειμένου χωρίς να αυτό να χρησιμοποιείται.

Το Visual Studio 2010 υποστηρίζει την ανάπτυξη προγραμμάτων στις C++, C#, Visual Basic, F# και τη μεταφορά προγραμμάτων από την μία γλώσσα στην άλλη. Με άλλα λόγια συγγραφεί ένα πρόγραμμα σε γλώσσα C++ μπορεί απλά και εύκολα να μετατραπεί αυτόματα σε κάποια εκ των γλωσσών που υποστηρίζει το Visual Studio.

3.3. ArgoUML

Το ArgoUML είναι ένα διαδραστικό, γραφικό περιβάλλον εργασίας σχεδίασης λογισμικού που υποστηρίζει το σχεδιασμό, την ανάπτυξη και τεκμηρίωση των αντικειμενοστραφής εφαρμογών. Υποστήριξη όλα τα τυπικά UML διαγράμματα όπως Use Case, Class Diagrams, Activity Diagrams κ.α.

3.4. Helpinator Professional

Το Helpinator είναι ένα εργαλείο συγγραφής Βοήθειας και εγχειριδίων. Μέσω αυτού μπορούμε με εύκολο τρόπο να δημιουργήσουμε αρχεία βοήθειας σε *.chm, *.pdf, και *.doc μορφή και να τα ενσωματώσουμε στην εφαρμογή μας.

4. Η ανάπτυξη της εφαρμογής

Η ανάπτυξη της εφαρμογής έχει βασιστεί στις ακόλουθες αρχές:

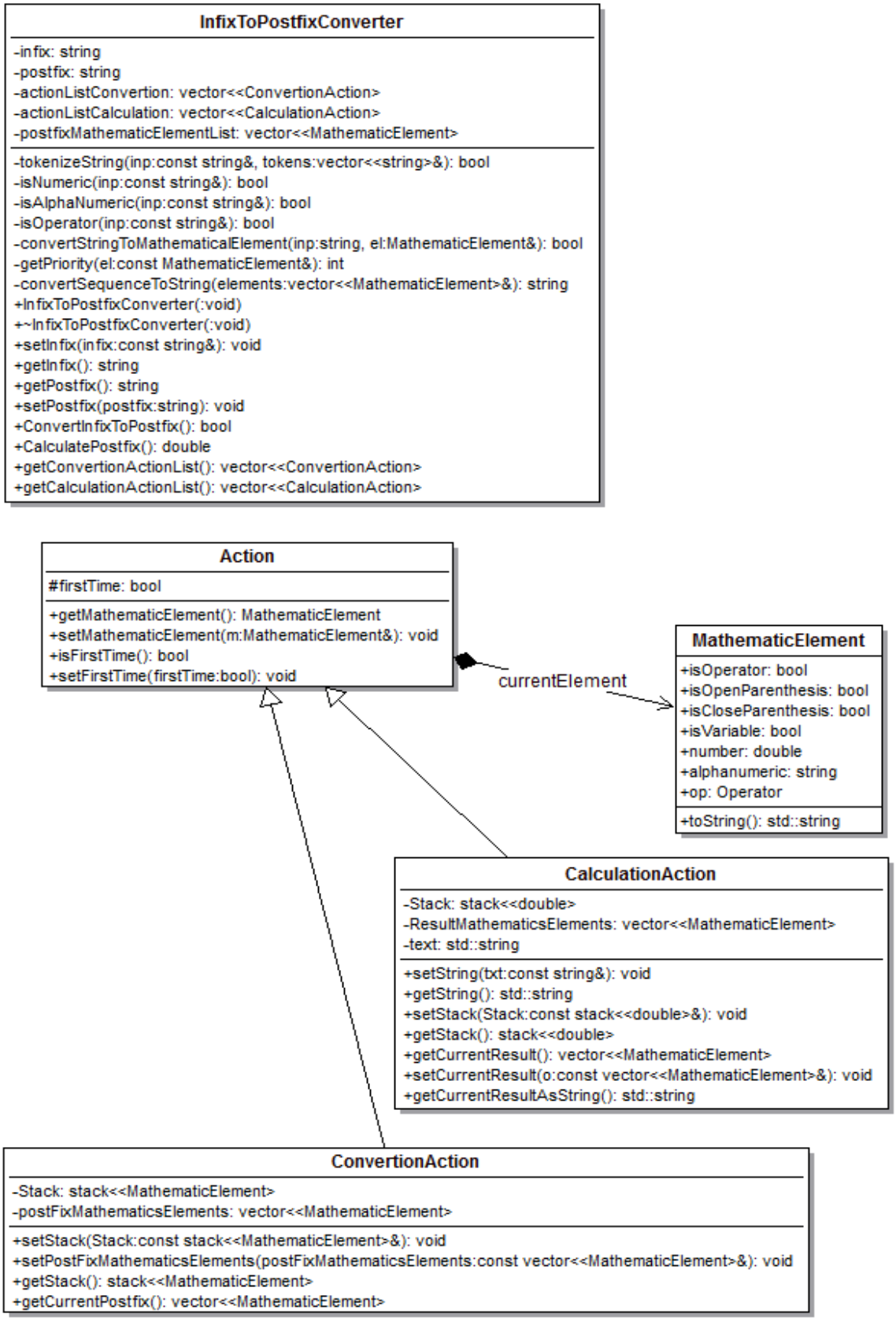
- Διαχωρισμός γραφικού περιβάλλοντος, από τα βασικά τμήματα της διαδικασίας
- Η διαδικασία θα υλοποιηθεί με μορφή επέκτασης στο γραφικό περιβάλλον. Για τον λόγο αυτό επιλέξαμε την δημιουργία μιας βασικής βιβλιοθήκης η οποία θα εκτελεί τις μετατροπές και τους υπολογισμούς. Η διασύνδεση της με το γραφικό περιβάλλον θα πραγματοποιηθεί μέσω του προτύπου τρόπου διασύνδεσης δυναμικών βιβλιοθηκών (Dynamic Link Libraries – DLL)
- Η βιβλιοθήκη θα αναπτυχθεί εξολοκλήρου σε φυσικό (native) κώδικα C++
- Στο γραφικό περιβάλλον, για να αξιοποιήσουμε τις βιβλιοθήκες που προσφέρονται από το .NET Framework, θα χρησιμοποιήσουμε managed κώδικα.
- Επειδή η διαδικασία θα επιδεικνύεται για εκπαιδευτικούς σκοπούς, θα υπάρξει κατάλληλη καταγραφή όλων των βημάτων μετατροπής ή υπολογισμού από την βιβλιοθήκη. Τα βήματα αυτά θα παρουσιάζονται με κατάλληλο γραφικό τρόπο στη γραφική διεπαφή.

Στη συνέχεια θα επικεντρωθούμε αρχικά στον σχεδιασμό και την υλοποίηση της βιβλιοθήκης, και έπειτα στον σχεδιασμό και την υλοποίηση του γραφικού περιβάλλοντος.

Στο σημείο αυτό, αξίζει να σημειωθεί ότι μετά την αρχική δημιουργία της βιβλιοθήκης, για τις ανάγκες του γραφικού περιβάλλοντος, αρκετές αλλαγές πραγματοποιήθηκαν κατά την διαδικασία της ανάπτυξης.

4.1. Βιβλιοθήκη μετατροπής και υπολογισμού εκφράσεων

Ο σχεδιασμός της βιβλιοθήκη έχει στηριχθεί σχεδόν εξολοκλήρου στις αρχές του αντικειμενοστραφούς προγραμματισμού. Έχουν δημιουργηθεί οι ακόλουθες κλάσεις αντικειμένων



Εικόνα 3 Διάγραμμα Κλάσεων

1. MathematicElement: Αποτελεί τη βασική κλάση της προσέγγισής μας. Κάθε στοιχείο που εμφανίζεται σε μια μαθηματική παράσταση, είτε είναι τελεστής, είτε αριθμός, είτε παρένθεση, δημιουργεί ένα αντικείμενο αυτού του τύπου. Η κλάση αυτή περιέχει μόνο μία συνάρτηση μετατροπής σε συμβολοσειρά για εκτύπωση

   Οι ιδιότητες της συγκεκριμένης κλάσης είναι :

   - alphanumeric: μεταβλητή string η οποία αποθηκεύει την μεταβλητή
   - isCloseParenthesis: boolean μεταβλητή η οποία, όταν είναι αληθής, δηλώνει ότι το συγκεκριμένο αντικείμενο εκφράζει κλείσιμο παρένθεσης
   - isOpenParenthesis: boolean μεταβλητή η οποία, όταν είναι αληθής, δηλώνει ότι το συγκεκριμένο αντικείμενο εκφράζει άνοιγμα παρένθεσης
   - isOperator: boolean μεταβλητή η οποία, όταν είναι αληθής, δηλώνει ότι το συγκεκριμένο αντικείμενο εκφράζει τελεστή
   - isVariable: boolean μεταβλητή η οποία, όταν είναι αληθής, δηλώνει ότι το συγκεκριμένο αντικείμενο εκφράζει μεταβλητή
   - number: αριθμητική μεταβλητή η οποία αποθηκεύει τον αριθμό που εκφράζει το μαθηματικό στοιχείο
   - op: Η μεταβλητή που αποθηκεύει τον τελεστή.

```
class MathematicElement
{
	public:
		bool isOperator;
		bool isOpenParenthesis;
		bool isCloseParenthesis;
		bool isVariable;
		double number;
		string alphanumeric;
		Operator op;

	std::string toString()
	{
		std::string returnString;
```

```
            if(isVariable && isOperator)
            {
                  return "end";
            }

            if(isOperator)
            {
                  switch(op)
                  {
                  case PLUS:
                        returnString = "+";
                        return returnString;
                  case MINUS:
                        returnString = "-";
                        return returnString;
                  case POWER:
                        returnString = "^";
                        return returnString;
                  case MULTIPLY:
                        returnString = "*";
                        return returnString;
                  case DIVIDE:
                        returnString = "/";
                        return returnString;
                  case MODULO:
                        returnString = "%";
                        return returnString;
                  }
            }

            if(isOpenParenthesis)
            {
                  returnString ="(";
                  return returnString;
            }




            if(isCloseParenthesis)
            {
                  returnString =")";
                  return returnString;
            }

            if(isVariable)
            {
                  returnString = alphanumeric;
                  return returnString;
            }
            char s[100];
      //floor(number): Επιστρέφει το μεγαλύτερο ακέραιο που δεν
υπερβαίνει την καθορισμένη αριθμητική έκφραση
```

```
            if(number-floor(number)<1e-7)
            {

                  sprintf(s,"%d ",(int)floor(number));
            }
            else
            {
                  sprintf(s,"%0.2lf ",number);
            }

            returnString = s;
            return returnString;
      }
};
```

2. Action: Η συγκεκριμένη κλάση είναι αφηρημένη, και χρησιμοποιείται ως βάση για τις δύο επόμενες.

```
   class  Action
   {
      protected:
            bool firstTime;
            MathematicElement currentElement;
      public:
            MathematicElement getMathematicElement()
            {
                  return currentElement;
            }

            void setMathematicElement(MathematicElement &m)
            {
                  this->currentElement=m;
            }

            bool isFirstTime()
            {
                  return firstTime;
            }

            void setFirstTime(bool firstTime)
            {
                  this->firstTime = firstTime;
            }
   };
```

3. ConvertionAction: Αναπαριστά ένα βήμα της διαδικασίας μετατροπής. Περιλαμβάνει την κατάσταση στη στοίβα, το τρέχον σημείο στην ενδοθεματική έκφραση καθώς και την τρέχουσα μορφή της μεταθεματικής έκφρασης

```
class  ConvertionAction: public Action
{
	stack<MathematicElement> Stack;
	vector<MathematicElement> postFixMathematicsElements;

	public:
		void setStack(const stack<MathematicElement> &Stack)
		{
			this->Stack=Stack;
		}
		void setPostFixMathematicsElements(const vector<MathematicElement> &postFixMathematicsElements)
		{
			this->postFixMathematicsElements = postFixMathematicsElements;
		}
		stack<MathematicElement> getStack()
		{
			return Stack;
		}
		vector<MathematicElement> getCurrentPostfix()
		{
			return postFixMathematicsElements;
		}
};
```

4. CalculationAction: Αναπαριστά ένα βήμα της διαδικασίας υπολογισμού. Περιλαμβάνει την κατάσταση στη στοίβα, το τρέχον σημείο στην έκφραση καθώς και την τρέχουσα μορφή του υπολογισμού

```
class CalculationAction: public Action
{
	stack<double> Stack;
	vector<MathematicElement> ResultMathematicsElements;
	std::string text;

	public:
		void setString(const string &txt)
		{
			this->text = txt;
		}

		std::string getString()
		{
			return text;
		}

		void setStack(const stack<double> &Stack)
		{
			this->Stack=Stack;
		}

		stack<double> getStack()
		{
			return Stack;
		}

		vector<MathematicElement> getCurrentResult()
		{
```

```
                return ResultMathematicsElements;
        }

        void setCurrentResult(const vector<MathematicElement> &o)
        {
                this->ResultMathematicsElements=o;
        }

        std::string getCurrentResultAsString()
        {
                std::string ret("");
                for(int i=0, l=ResultMathematicsElements.size(); i<l; i++)
                {
                        MathematicElement m = ResultMathematicsElements[i];
                        ret += m.toString();
                }
                return ret;
        }
};
```

5. InfixToPostfixConverter: Η κλάση αυτή αναλαμβάνει την μετατροπή των ενδοθεματικών εκφράσεων σε μεταθεματικές, καθώς και τον υπολογισμό των μεταθεματικών εκφράσεων.

```
class InfixToPostfixConverter
{
      string infix, postfix;
      vector<ConvertionAction>  actionListConvertion;
      vector<CalculationAction> actionListCalculation;
      vector<MathematicElement> postfixMathematicElementList;
      bool tokenizeString(const string &inp, vector<string> &tokens);
      bool isNumeric(const string &inp);
      bool isAlphaNumeric(const string &inp);
      bool isOperator(const string &inp);
      bool convertStringToMathematicalElement(string inp, MathematicElement &el);
      int getPriority(const MathematicElement &el);
      string convertSequenceToString(vector<MathematicElement> &elements);
   public:
      InfixToPostfixConverter(void);
      ~InfixToPostfixConverter(void){};
      void setInfix(const string &infix);
      string getInfix();
      void setPostfix(string postfix);
      string getPostfix();
      bool ConvertInfixToPostfix();
      double CalculatePostfix();
      vector<ConvertionAction>  getConvertionActionList();
      vector<CalculationAction>  getCalculationActionList();
};
```

Πιο συγκεκριμένα, η κλάση περιλαμβάνει τις ακόλουθες μεθόδους:

- getInfix & setInfix, getPostfix & setPostfix: Βοηθητικές συναρτήσεις που τροποποιούν και επιστρέφουν τις αντίστοιχες εκφράσεις (ενδοθεματική και μεταθεματική)

```
void setInfix(const string &infix)
	{
		this->infix = infix;
	};

	string getInfix()
	{
		return infix;
	};

	void setPostfix(string postfix)
		{
			this->postfix = postfix;
	};

	string getPostfix()
	{
		return postfix;
	};
```

- tokenizeString: Η συγκεκριμένη συνάρτηση διασπάει μια ενιαία ακολουθία χαρακτήρων σε επιμέρους ακολουθίες, όπου κάθε μια περιλαμβάνει μόνο ένα μαθηματικό στοιχείο

```
bool InfixToPostfixConverter::tokenizeString( const string &inp, vector<string> &tokens )
{
	int firstCharacter = 0, inpLength = (int)inp.length();
	tokens.clear();
	bool wasPreviousNumeric = false;
	bool wasPreviousAlphaNumeric = false;

	/*Ξεκινάμε και σαρώνουμε το string από τον πρώτο χαρακτήρα. Σταδιακά ανεβάζουμε το μήκος του ελεγχόμενου πεδίου.	Μόλις το πεδίο γίνει μη αποδεκτό, αποκόπτουμε το τελευταίο έγκυρο τμήμα από την αρχή του string και επαναλαμβάνουμε μέχρι το string να είναι κενό.*/

	for(int i = 0; i < inpLength; i++)
	{
		string s = inp.substr(firstCharacter,(1+i-firstCharacter));
		if(isOperator(s) || s.compare("(")==0 || s.compare(")")==0)
		{
			firstCharacter = i+1;
			tokens.push_back(s);
			continue;
		}
		if(isNumeric(s))
		{
			wasPreviousNumeric = true;
			continue;
		}

		if(isAlphaNumeric(s))
		{
			wasPreviousAlphaNumeric = true;
```

```
                        continue;
                }
                if(wasPreviousNumeric || wasPreviousAlphaNumeric)
                {
                        i--;
                        s = inp.substr(firstCharacter,(1+i-firstCharacter));
                        tokens.push_back(s);
                        firstCharacter = i+1;
                        wasPreviousNumeric=wasPreviousAlphaNumeric=false;
                        continue;
                }
                printf("ERROR");
                return false;
        }
        string s=inp.substr(firstCharacter);
        tokens.push_back(s);
        for(int i = tokens.size()-1; i>=0; i--)
        {
                if(tokens[i].length()==0)
                {
                        tokens.erase(tokens.begin()+i);
                }
        }
        return true;
}
```

- convertStringToMathematicalElement: Η συνάρτηση που μετατρέπει την συμβολοσειρά σε μαθηματικό στοιχείο

```
bool InfixToPostfixConverter::convertStringToMathematicalElement( string inp, MathematicElement &el )
{
	el.op=NONE;

	if(isOperator(inp))
	{
		el.isOperator=true;
		el.isVariable=el.isOpenParenthesis=el.isCloseParenthesis=false;
		if(inp.compare("+")==0)
			el.op=PLUS;
		else if(inp.compare("-")==0)
			el.op=MINUS;
		else if(inp.compare("*")==0)
			el.op=MULTIPLY;
		else if(inp.compare("/")==0)
			el.op=DIVIDE;
		else if(inp.compare("^")==0)
			el.op=POWER;
		else if(inp.compare("%")==0)
			el.op=MODULO;
		return true;
	}


if(inp.compare("(")==0)
	{
		el.isOpenParenthesis=true;
		el.isVariable=el.isCloseParenthesis=el.isOperator=false;
		return true;
	}
```

```
	if(inp.compare(")")==0)
	{
		el.isCloseParenthesis=true;
		el.isVariable=el.isOpenParenthesis=el.isOperator=false;
		return true;
	}
	if(isNumeric(inp))
	{
		el.number=atof(inp.c_str());
		el.isVariable=el.isOperator=el.isOpenParenthesis=el.isCloseParenthesis=false;
		return true;
	}
	if(isAlphaNumeric(inp))
	{
		el.isOperator=el.isOpenParenthesis=el.isCloseParenthesis=false;
		el.isVariable=true;
		el.alphanumeric = inp;
		return true;
	}
	return false;
}
```

- isAlphaNumeric: Η συνάρτηση ελέγχει αν η δοθείσα ακολουθία χαρακτήρων είναι το όνομα μιας μεταβλητής

```
bool InfixToPostfixConverter::isAlphaNumeric( const string &inp )
{
	int inpLength = (int)inp.length();
	const char *in=inp.c_str();
	if(!isalpha(in[0]))
	{
		return false;
	}
	for(int i = 1;i<inpLength;i++)
	{
		if(!isalnum(in[i]))
		{
			return false;
		}
	}
	return true;
}
```

- isNumeric: Η συνάρτηση ελέγχει αν η δοθείσα ακολουθία χαρακτήρων είναι μια αριθμητική τιμή

```
bool InfixToPostfixConverter::isNumeric( const string &inp )
{
	int inpLength = (int)inp.length();
	const char *in=inp.c_str();
	for(int i = 0;i<inpLength;i++)
	{
		if(!isdigit(in[i]) && in[i]!='.')
		{
			return false;
		}
	}
	double ss;
	int rr=sscanf(inp.c_str(), "%f", &ss);
	if(rr>0)
	{
		return true;
	}
	return false;
}
```

- isOperator: Η συνάρτηση ελέγχει αν η δοθείσα ακολουθία χαρακτήρων είναι ένας τελεστής

```
bool InfixToPostfixConverter::isOperator( const string &inp )
{
	if(inp.compare("+")==0) {
		return true;
	}

	if(inp.compare("-")==0) {
		return true;
	}

	if(inp.compare("*")==0) {
		return true;
	}

	if(inp.compare("/")==0) {
		return true;
	}

	if(inp.compare("%")==0) {
		return true;
	}

	if(inp.compare("^")==0) {
		return true;
	}
	return false;
}
```

- ConvertInfixToPostfix: Η συνάρτηση πραγματοποιεί την μετατροπή μιας ενδοθεματικής έκφρασης στην αντίστροφη πολωνική μορφή της. Η συγκεκριμένη συνάρτηση αποτελεί υλοποίηση του αλγορίθμου μετατροπής ενδοθεματικής έκφρασης σε μεταθεματική που περιγράψαμε στο 2 κεφάλαιο.

```
bool InfixToPostfixConverter::ConvertInfixToPostfix()
{
	//Έλεγχος αν υπάρχει έκφραση
	if(infix.length()==0)
	{
		return false;
	}

	//Αρχικοποιήσεις
	postfixMathematicElementList.clear();
	actionListConvertion.clear();
	int infixElementsNumber;
	vector<string> infixElements;

	//Διάσπαση της έκφρασης στα στοιχεία της --πίνακας από string
	if(!tokenizeString(infix, infixElements))
	{
		return false;
	}

	infixElementsNumber= (int)infixElements.size();
	vector<MathematicElement> infixMElements(infixElementsNumber);
	postfixMathematicElementList.reserve(infixElementsNumber);
	stack<MathematicElement> st;
	MathematicElement f;
	st.push(f);
	st.pop();
```

```
	//Μετατροπή των string σε MathematicElement
	for(int i = 0; i < infixElementsNumber; i++)
	{
		if(!convertStringToMathematicalElement(infixElements[i],infixMElements[i]))
			return false;
	}

	//Βασικός Αλγόριθμος μετατροπής σε RPN
	for(int i = 0; i < infixElementsNumber; i++)
	{
		bool firstTime = true;

		//Εαν είναι αριθμός ή μεταβλητή - > πάει κατευθειάν στην postfix
		if(!infixMElements[i].isCloseParenthesis && !infixMElements[i].isOpenParenthesis &&
!infixMElements[i].isOperator )
		{
			postfixMathematicElementList.push_back(infixMElements[i]);
		}
		else
		{
//Εαν η στοίβα είναι άδεια ή το πρώτο στοιχείο έχει χαμηλότερη προτεραιότητα,
			το στοιχείο μπαίνει στη στοίβα
			int priority = getPriority(infixMElements[i]);
			if(st.empty() || priority>getPriority(st.top()) || infixMElements[i].isOpenParenthesis)
			{
				st.push(infixMElements[i]);
			}
			//Διαφορετικά, αδειάζουμε τη στοίβα μέχρι να ικανοποιηθεί η παραπάνω συνθήκη και μετά
				προσθέτουμε στη στοίβα.
			else if(infixMElements[i].isCloseParenthesis)
			{
				while(!(st.top().isOpenParenthesis))
				{
					postfixMathematicElementList.push_back(st.top());
```

```
                        st.pop();

                        //Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
                        ConvertionAction currentStatus;
                        currentStatus.setPostFixMathematicsElements(postfixMathematicElementList);
                        currentStatus.setStack(st);
                        currentStatus.setMathematicElement(infixMElements[i]);
                        currentStatus.setFirstTime(firstTime);
                        firstTime=false;
                        this->actionListConvertion.push_back(currentStatus);
                  }
                  st.pop();
            }

            else
            {
                  while(!st.empty() && priority<=getPriority(st.top()))
                  {
                        postfixMathematicElementList.push_back(st.top());
                        st.pop();

                        //Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
                        ConvertionAction currentStatus;
                        currentStatus.setPostFixMathematicsElements(postfixMathematicElementList);
                        currentStatus.setStack(st);
                        currentStatus.setMathematicElement(infixMElements[i]);
                        currentStatus.setFirstTime(firstTime);
                        firstTime=false;
                        this->actionListConvertion.push_back(currentStatus);
                  }
                  st.push(infixMElements[i]);
            }
      }
```

```
			//Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
			ConvertionAction currentStatus;
			currentStatus.setPostFixMathematicsElements(postfixMathematicElementList);
			currentStatus.setStack(st);
			currentStatus.setMathematicElement(infixMElements[i]);
			currentStatus.setFirstTime(firstTime);
			firstTime=false;
			this->actionListConvertion.push_back(currentStatus);
		}

		while(!st.empty())
		{
			//Στο τέλος αδειάζουμε τη στοίβα στην postfix
			if(st.top().isOpenParenthesis)
				return false;
			postfixMathematicElementList.push_back(st.top());
			st.pop();

			//Προσθήκη της τελικής κατάστασης
			MathematicElement empt;empt.isOperator=empt.isVariable = true;
			ConvertionAction currentStatus;
			currentStatus.setPostFixMathematicsElements(postfixMathematicElementList);
			currentStatus.setStack(st);
			currentStatus.setMathematicElement(empt);
			this->actionListConvertion.push_back(currentStatus);
		}

		//Μετατροπή της τελικής έκφρασης σε string.
		postfix = convertSequenceToString(postfixMathematicElementList);
		return true;
}
```

- convertSequenceToString: Η συνάρτηση επαναφέρει τα MathematicalElements σε μορφή συμβολοσειράς.

```
string InfixToPostfixConverter::convertSequenceToString(vector<MathematicElement> &elements )
{
	string returnString = "";
	int l = (int)elements.size();
	int i = 0;
	for(i = 0; i < l; i++)
	{
		returnString += elements[i].toString();
	}
	return returnString;
}
```

- getPriority: Η συνάρτηση επιστρέφει την προτεραιότητα ενός τελεστή. Η προτεραιότητα αναπαρίσταται συμβατικά με μορφή ακεραίου. Μεγαλύτερη προτεραιότητα σημαίνει και μεγαλύτερος ακέραιος.

```
int InfixToPostfixConverter::getPriority( const MathematicElement &el )
{
	if(el.isOperator)
		switch(el.op)
		{
			case MODULO:
			case POWER:
				return 5;
			case MULTIPLY:
			case DIVIDE:
				return 4;
			case PLUS:
			case MINUS:
				return 3;
		}

	if(el.isOpenParenthesis)
		return 1;

	if(el.isCloseParenthesis)
		return 0;

	printf("Unknown operator\n");
	return -1;
}
```

- CalculatePostfix: Η συνάρτηση πραγματοποιεί τον υπολογισμό έκφρασης από την αντίστροφη πολωνική μορφή της. Η συγκεκριμένη συνάρτηση αποτελεί υλοποίηση του αλγορίθμου υπολογισμού μεταθεματικής έκφρασης που περιγράψαμε στο 2 κεφάλαιο.

```
double InfixToPostfixConverter::CalculatePostfix()
{
	int i;
	actionListCalculation.clear();
	int numberOfElements = (int)postfixMathematicElementList.size();
	stack<double> st;
	std::string str;

	//Βασικός αλγόριθμος
	for(i = 0; i < numberOfElements; i++)
	{
		bool isfirst = true;

		//Αν το στοιχείο είναι τελεστής, τότε:
		if(postfixMathematicElementList[i].isOperator)
		{
			//ανακαλούνται 1, ή 2 στοιχεία από τη στοίβα
			double x2 = st.top(),x1,y;
			//Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
			{
				CalculationAction action;
				action.setStack(st);
				action.setMathematicElement(postfixMathematicElementList[i]);
				action.setFirstTime(isfirst);
				isfirst = false;
				std::string txt ;
				txt = txt + postfixMathematicElementList[i].toString();
				action.setString(txt);
```

```
                        std::vector<MathematicElement> currentStatus;
                        MathematicElement m;
                        m.isCloseParenthesis=m.isOpenParenthesis=m.isOperator=m.isVariable=false;
                        m.number = x2;
                        currentStatus.push_back(m);
                        currentStatus.insert(currentStatus.end(),
postfixMathematicElementList.begin()+i+1,postfixMathematicElementList.end());
                        action.setCurrentResult(currentStatus);
                        actionListCalculation.push_back(action);
                  }

                  st.pop();
                  x1 = st.top();

                  //Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
                  {
                        CalculationAction action;
                        action.setStack(st);
                        action.setMathematicElement(postfixMathematicElementList[i]);
                        action.setFirstTime(isfirst);
                        isfirst = false;
                        std::string txt;
                        char chr[20];
                        if(x2-floor(x2)<1e-7)
                              sprintf(chr,"%d",(int)floor(x2));
                        else
                              sprintf(chr,"%0.2lf",x2);
                        txt = postfixMathematicElementList[i].toString() + chr;
                        action.setString(txt);
                        std::vector<MathematicElement> currentStatus;
                        MathematicElement m;
                        m.isCloseParenthesis=m.isOpenParenthesis=m.isOperator=m.isVariable=false;
                        m.number = x1;
                        currentStatus.push_back(m);
```

```
                        currentStatus.insert(currentStatus.end(),
postfixMathematicElementList.begin()+i+1,postfixMathematicElementList.end());
                        action.setCurrentResult(currentStatus);
                        actionListCalculation.push_back(action);
                  }
                  st.pop();
                  //υπολογίζεται το αποτέλεσμα ανάλογα τον τελεστή
                  switch(postfixMathematicElementList[i].op)
                  {
                  case  MODULO:
                        y= ((int)floor(x1+0.5))%((int)floor(x2+0.5));
                        break;

                  case POWER:
                        y=pow(x1,x2);
                        break;

                  case MULTIPLY:
                        y=x1*x2;
                        break;


                  case DIVIDE:
                        y=x1/x2;
                        break;

                  case PLUS:
                        y=x1+x2;
                        break;

                  case MINUS:
                        y=x1-x2;
                        break;
                  }
```

```
                  //Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
                        CalculationAction action;
                        action.setStack(st);
                        action.setMathematicElement(postfixMathematicElementList[i]);
                        action.setFirstTime(isfirst);
                        isfirst = false;
                        char chr[20];
                        if(x1-floor(x1)<1e-7) // ???
                              sprintf(chr,"%d",(int)floor(x1));
                        else
                              sprintf(chr,"%0.2lf",x1);
                        std::string txt = chr;

                        if(x2-floor(x2)<1e-7)
                              sprintf(chr,"%d",(int)floor(x2));
                        else
                              sprintf(chr,"%0.2lf",x2);
                        txt = txt + postfixMathematicElementList[i].toString() + chr;
                        action.setString(txt);
                        std::vector<MathematicElement> currentStatus;
                        MathematicElement m;
                        m.isCloseParenthesis=m.isOpenParenthesis=m.isOperator=m.isVariable=false;
                        m.number = y;
                        currentStatus.push_back(m);
                        currentStatus.insert(currentStatus.end(),
postfixMathematicElementList.begin()+i+1,postfixMathematicElementList.end());
                        action.setCurrentResult(currentStatus);
                        actionListCalculation.push_back(action);
                  //Το αποτέλεσμα επιστρέφει στη στοίβα
                  st.push(y);

                  if(y-floor(y)<1e-7)
                        sprintf(chr,"%d",(int)floor(y));
```

```
                    else
                          sprintf(chr,"%0.2lf",y);

                    str = chr;
              }
              //Διαφορετικα, το στοιχείο προστίθεται στη στοίβα
              else
              {
                    st.push(postfixMathematicElementList[i].number);
                    str ="";
              }
              //Αρχείο καταγραφής βημάτων -- Προσθήκη της κατάστασης
                    CalculationAction action;
                    action.setString(str);
                    action.setFirstTime(isfirst);
                    isfirst = false;
                    action.setStack(st);
                    action.setMathematicElement(postfixMathematicElementList[i]);
                    std::vector<MathematicElement> currentStatus;
                    MathematicElement m;
                    m.isCloseParenthesis=m.isOpenParenthesis=m.isOperator=m.isVariable=false;
                    m.number = st.top();
                    currentStatus.push_back(m);
                    currentStatus.insert(currentStatus.end(),
postfixMathematicElementList.begin()+i+1,postfixMathematicElementList.end());
                    action.setCurrentResult(currentStatus);
                    actionListCalculation.push_back(action);
      }
      CalculationAction action;
      action.setStack(st);
      actionListCalculation.push_back(action);
      return st.top();
}
```

- getConvertionActionList: Η συνάρτηση επιστρέφει τον αναλυτικό κατάλογο ενεργειών μετατροπής ενδοθεματικής έκφρασης στη μεταθεματική γραφή της. Η συνάρτηση αυτή αξιοποιείται από το γραφικό περιβάλλον για τις ανάγκες της παρουσίασης της μετατροπής βήμα προς βήμα.

```
vector<ConvertionAction> getConvertionActionList()
{
	return actionListConvertion;
}
```

- getCalculationActionList: Η συνάρτηση επιστρέφει τον αναλυτικό κατάλογο ενεργειών υπολογισμού της αντίστροφης πολωνικής έκφρασης. Η συνάρτηση αυτή αξιοποιείται από το γραφικό περιβάλλον για τις ανάγκες της παρουσίασης της διαδικασίας βήμα προς βήμα.

```
vector<CalculationAction> getCalculationActionList()
{
	return actionListCalculation;
}
```

5. Παρουσίαση της εφαρμογής

5.1. Οδηγίες Εγκατάστασης

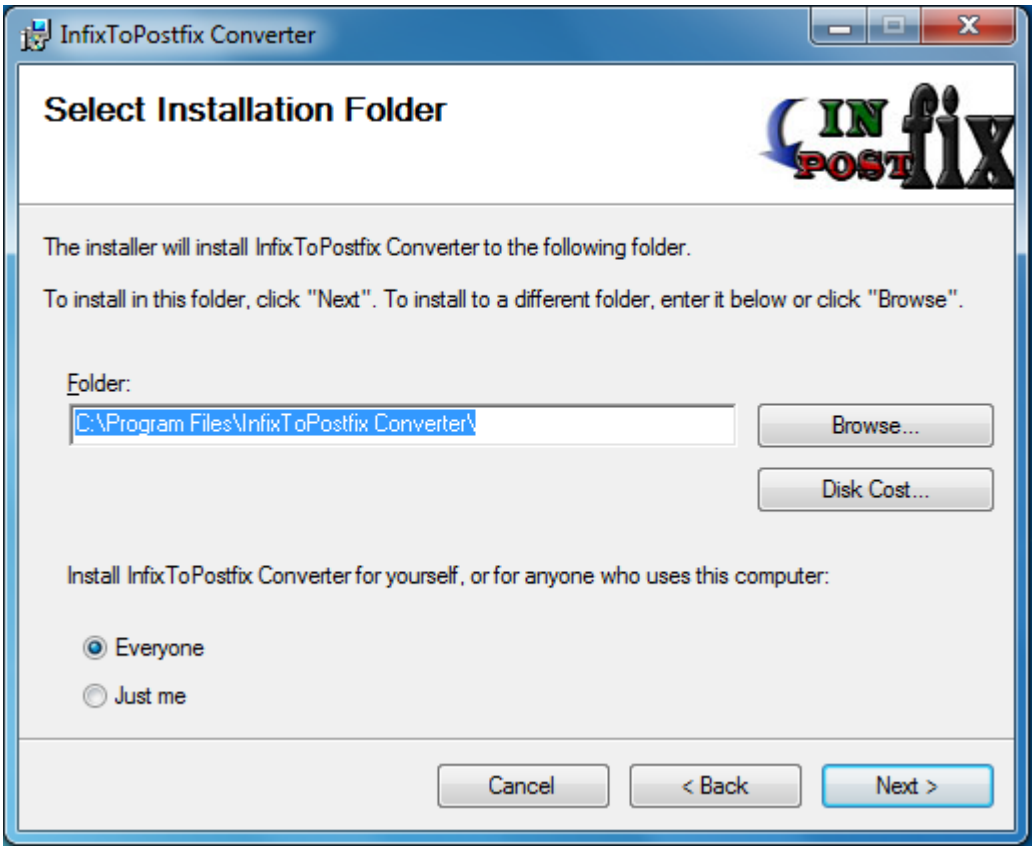
Για να εγκαταστήσετε την εφαρμογή τοποθετήστε το CD στον υπολογιστή σας και ανοίξτε το με την εξερεύνηση των Windows. Ανοίξτε τον φάκελο Setup και επιλέξτε τον φάκελο που αντιστοιχεί στον τύπο πλατφόρμας των Windows του υπολογιστή σας (32-bit (x 86) ή 64-bit (x 64)). Στη συνέχεια θα κάνετε διπλό κλικ στο αρχείο setup.exe που βρίσκεται στον φάκελο.

Ξεκινώντας η εγκατάσταση στην πρώτη οθόνη της ο οδηγός σας καλωσορίζει στην εγκατάσταση του InfixToPostfix Converter (Εικόνα 4)

Εικόνα 4 Εγκατάσταση: Οθόνη 1

Πατώντας Next περνάτε στην επόμενη οθόνη όπου εμφανίζεται η διαδρομή που θα εγκατασταθούν τα αρχεία του λογισμικού. Θα δείτε

ότι υπάρχει μία προεπιλεγμένη διαδρομή που καλό θα ήταν να διατηρήσετε. Αν έχετε 64-bit σύστημα επιλέξτε να εγκαταστήσετε την εφαρμογή στον φάκελο "C:***Program Files***\InfixToPostfix Converter\" και *όχι* στον "C:***Program Files (x86)***\InfixToPostfix Converter\" για να λειτουργεί σωστά η εφαρμογή (Εικόνα 5)



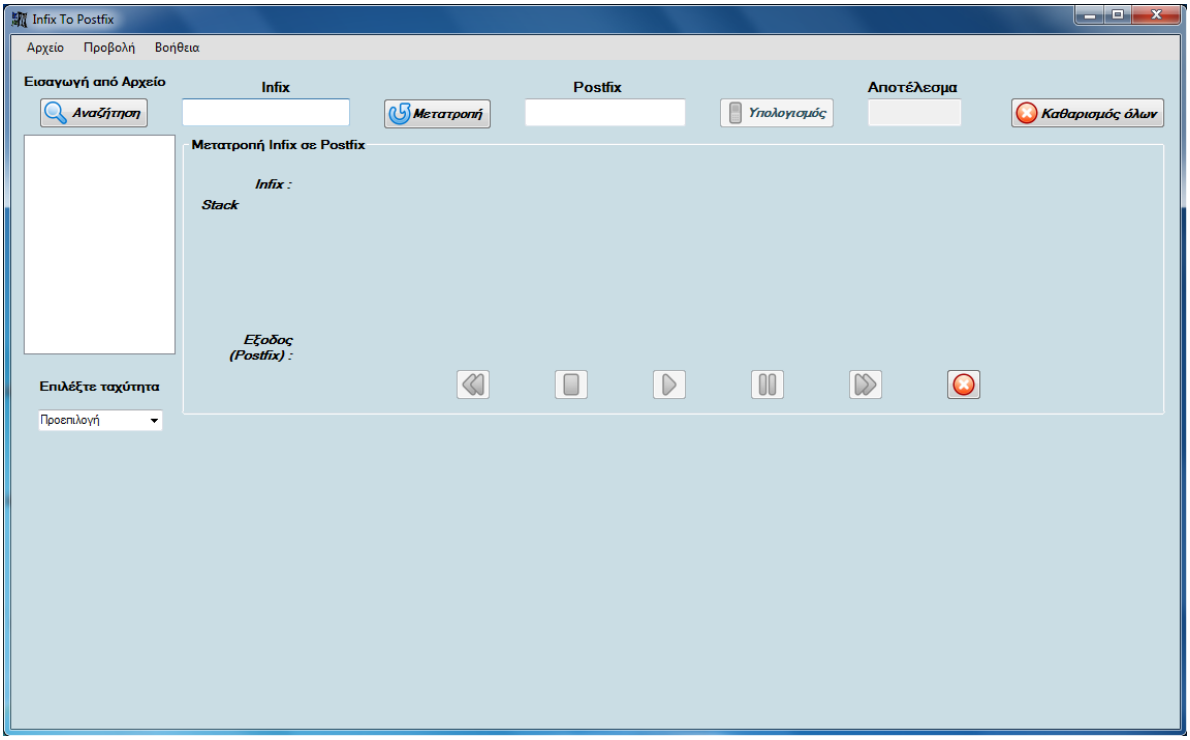
Εικόνα 5 Εγκατάσταση: Οθόνη 2

Πατώντας Next περνάτε στην επόμενη οθόνη όπου ο οδηγός σας ενημερώνει ότι είναι έτοιμος για να γίνει η εγκατάσταση του InfixToPostfix Converter. Για να συνεχίσετε πατάτε Next και πλέον αρχίζει η εγκατάσταση. Μόλις ολοκληρωθεί η εγκατάσταση πατήστε Close στην τελευταία οθόνη.

Στη συνέχεια θα προσέξετε ότι στην επιφάνεια εργασίας σας έχει εγκατασταθεί μία συντόμευση με όνομα ***InfixToPostfix Converter***

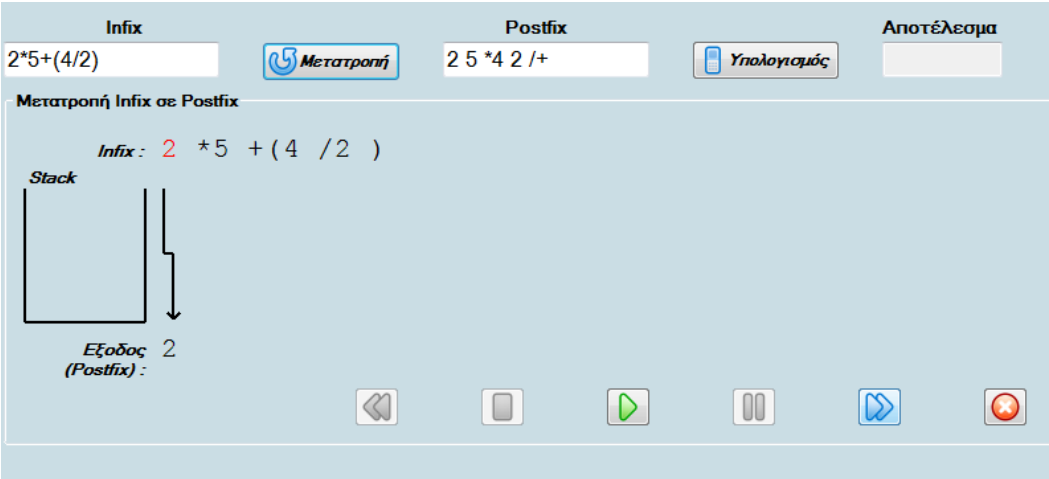
5.2. Οδηγίες Χρήσης

Εκκινώντας την εφαρμογή ο χρήστης βλέπει την κύρια φόρμα όπως φαίνεται στην Εικόνα 6



Εικόνα 6 Αρχική φόρμα της εφαρμογής

Στο πεδίο ***Infix***, πληκτρολογούμε μια αριθμητική έκφραση στην ενδοθεματική της μορφή (infix notation) όπως για παράδειγμα 2*5+(4/2). Κάνοντας κλικ στο κουμπί ***Μετατροπή*** γίνεται μετατροπή της έκφρασης στην μεταθεματική της μορφή δηλαδή 25*42/+ στο πεδίο ***Postfix*** και ταυτόχρονα σχεδιάζονται και γραφικά όπως φαίνεται στην Εικόνα 7



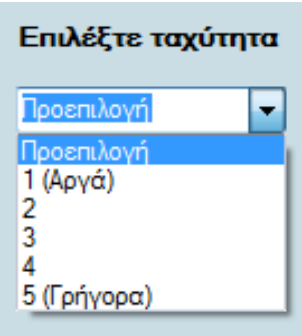
Εικόνα 7 Μετατροπή Infix σε Postfix

Με τα κουμπιά ελέγχου (βλ. Εικόνα 8) μπορούμε να δούμε βήμα-βήμα πως γίνεται η μετατροπή

Εικόνα 8 Κουμπιά ελέγχου

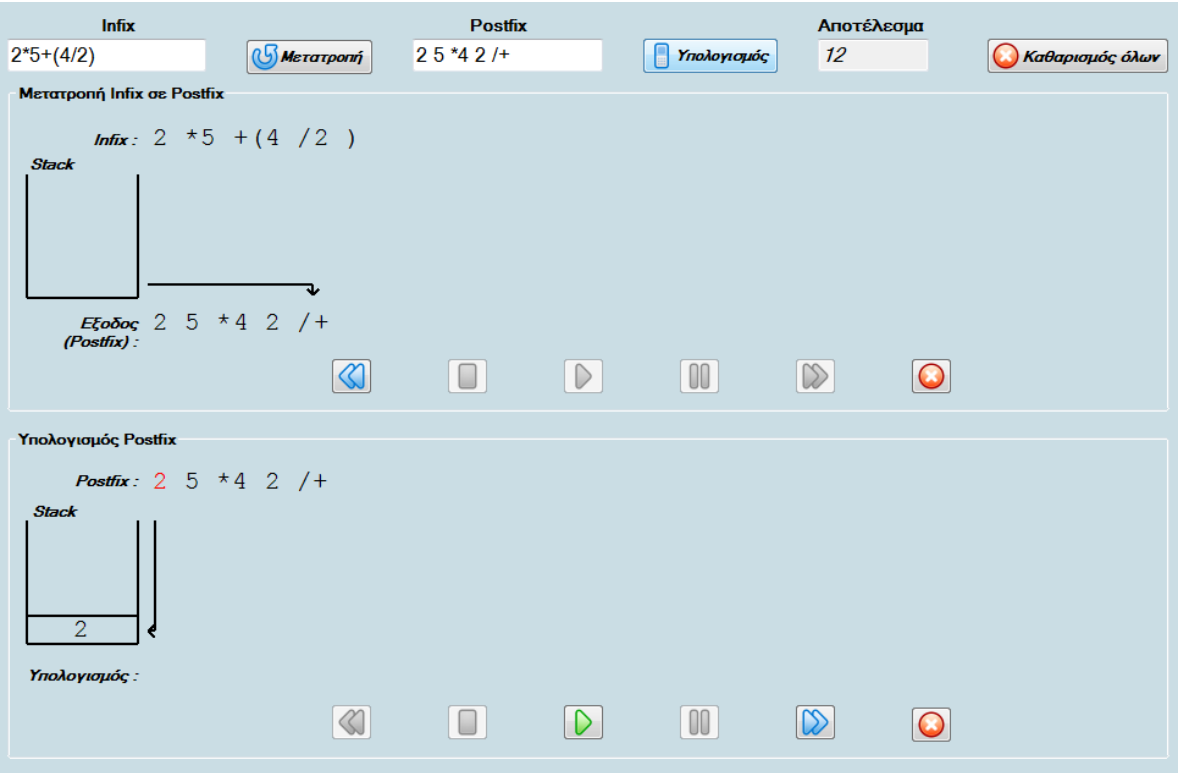
	Προηγούμενο βήμα
	Λήξη βηματικής εκτέλεσης με τη χρήση χρονοδιακόπτη
	Έναρξη βηματικής εκτέλεσης με τη χρήση χρονοδιακόπτη
	Παύση βηματικής εκτέλεσης με τη χρήση χρονοδιακόπτη
	Επόμενο βήμα
	Εκκαθάριση γραφικής αναπαράστασης

Επίσης έχουμε την δυνατότητα να επιλέξουμε την ταχύτητα του χρονοδιακόπτη όπως φαίνεται στην Εικόνα 9



Εικόνα 9 Επιλογή ταχύτητας χρονοδιακόπτη

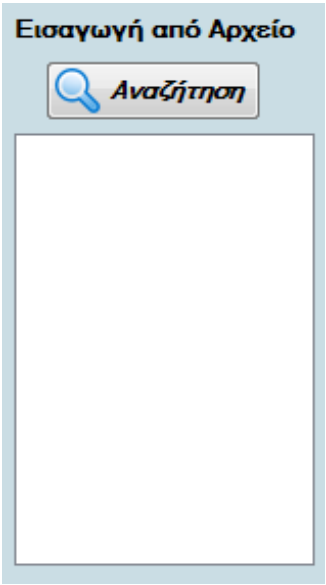
Αντίστοιχα ακολουθείται η ίδια διαδικασία κάνοντας κλικ στο κουμπί ***Υπολογισμός*** γίνεται ο υπολογισμός της έκφρασης στην μεταθεματική της μορφή δηλαδή 12 στο πεδίο ***Αποτέλεσμα*** και ταυτόχρονα σχεδιάζονται και γραφικά τα βήματα υπολογισμού όπως φαίνεται στην Εικόνα 10



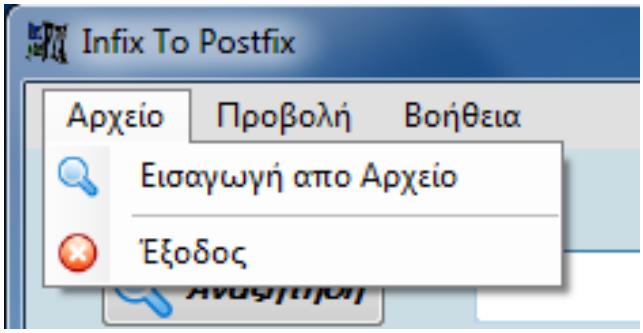
Εικόνα 10 Υπολογισμός έκφρασης Postfix

Κάνοντας κλικ στο κουμπί ***Καθαρισμός όλων*** επανέρχεται η φόρμα στην αρχική της μορφή όπως όταν άνοιξε η εφαρμογή.

Επίσης υπάρχει η δυνατότητα να εισάγουμε πολλές εκφράσεις infix από ένα αρχείο, είτε κάνοντας κλικ στο κουμπί ***Αναζήτηση*** (βλ. Εικόνα 11) είτε επιλέγοντας από το μενού ***Αρχείο*** την ***Εισαγωγή από Αρχείο*** (βλ Εικόνα 12)

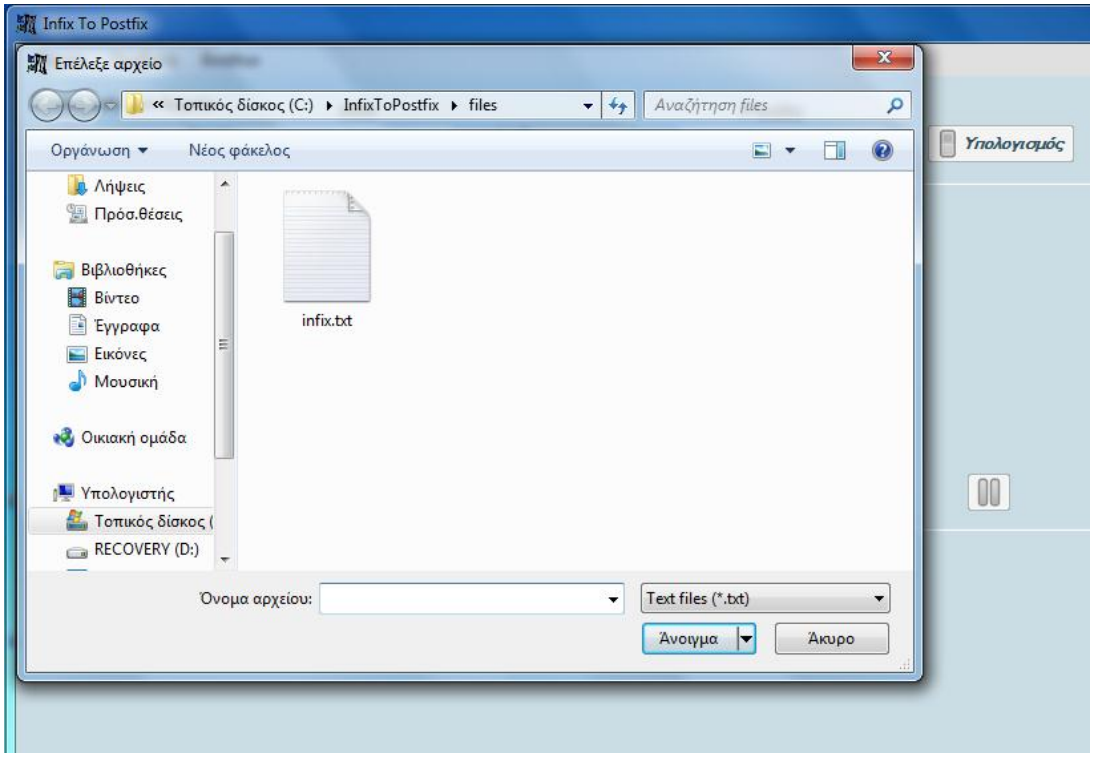


Εικόνα 11 Κουμπί "Εισαγωγή απο αρχείο"



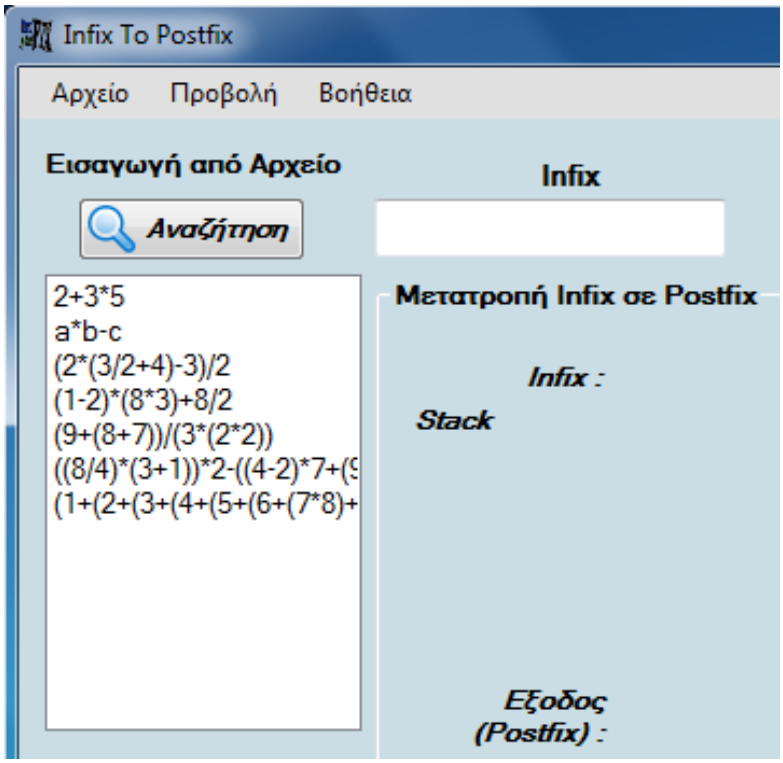
Εικόνα 12 Μενού "Εισαγωγή απο αρχείο"

Έπειτα εμφανίζεται το παράθυρο αναζήτηση αρχείου (βλ Εικόνα 13). Σ' αυτό το σημείο να τονίσουμε ότι μόνο από απλά αρχεία κειμένου (*.txt) μπορούμε να εισάγουμε εκφράσεις και κάθε έκφραση πρέπει να είναι γραμμένη σε διαφορετική γραμμή μέσα στο αρχείο.



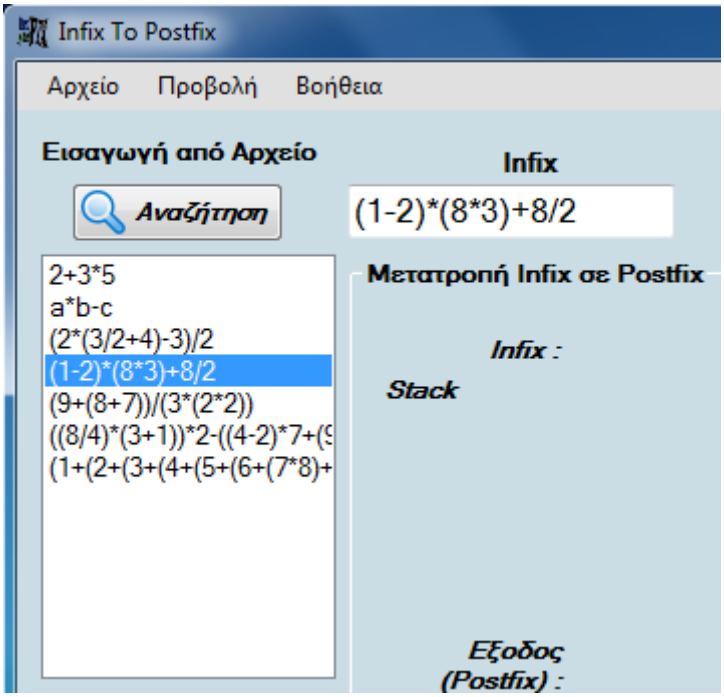
Εικόνα 13 Αναζήτηση Αρχείου

Στη συνέχεια οι εκφράσεις που περιείχε το αρχείο εισάγονται στην εφαρμογή (βλ Εικόνα 14).



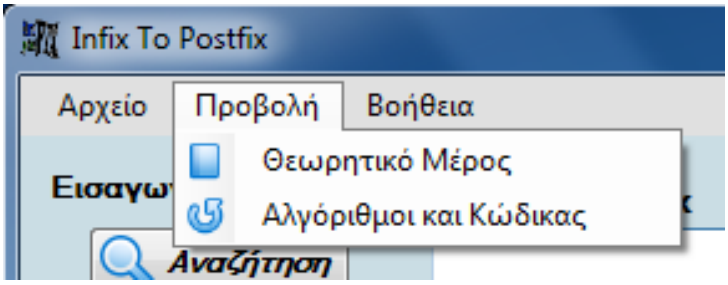
Εικόνα 14 Εισαγωγή εκφράσεων απο αρχείο

Για να τις χρησιμοποιήσουμε αρκεί να κάνουμε κλικ απλά πάνω στην έκφραση που θέλουμε και αυτή μεταφέρεται αυτομάτως στο πεδίο ***Infix*** (βλ. Εικόνα 15)

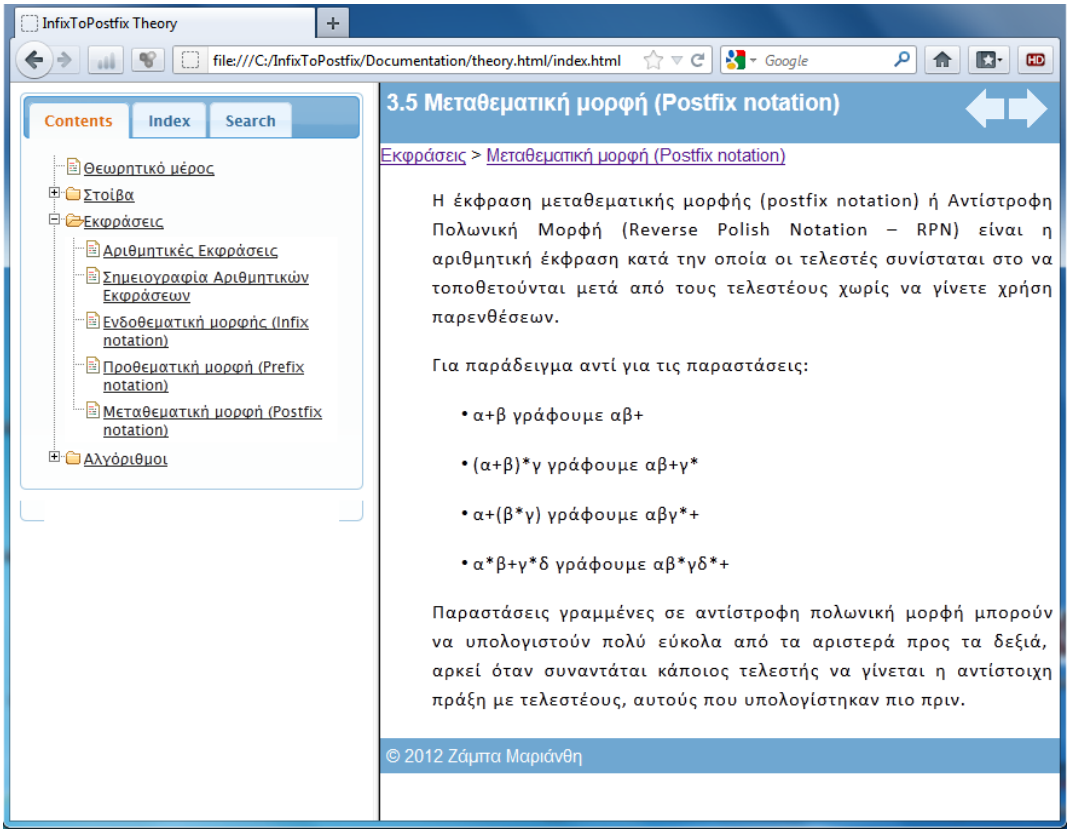


Εικόνα 15 Επιλογή εκφράσεων απο αρχείο

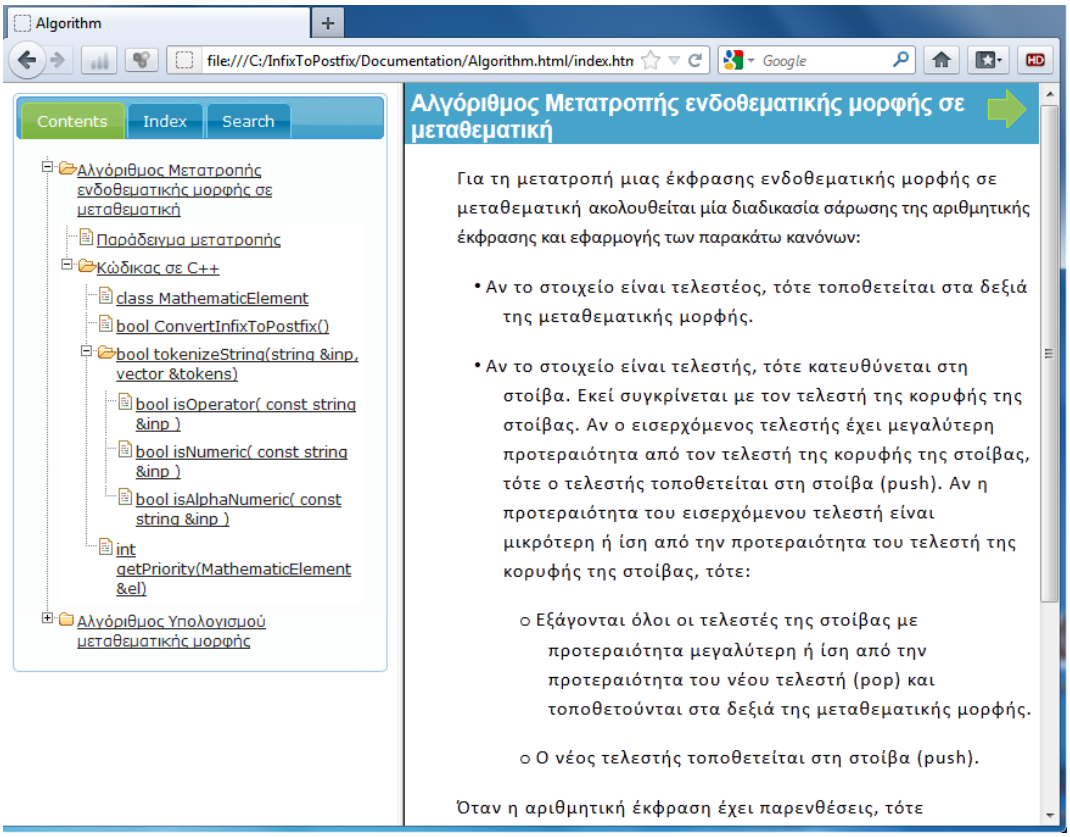
Από το μενού ***Προβολή*** (βλ Εικόνα 16) μπορούμε να επιλέξουμε να δούμε στον προεπιλεγμένο φυλλομετρητή (browser) του υπολογιστή μας το ***Θεωρητικό Μέρος*** (βλ Εικόνα 17) ή τους ***Αλγορίθμους και των Πηγαίο Κώδικα*** (βλ Εικόνα 18) πάνω στα οποία βασίστηκε η υλοποίηση της εφαρμογής.



Εικόνα 16 Μενού "Προβολή"

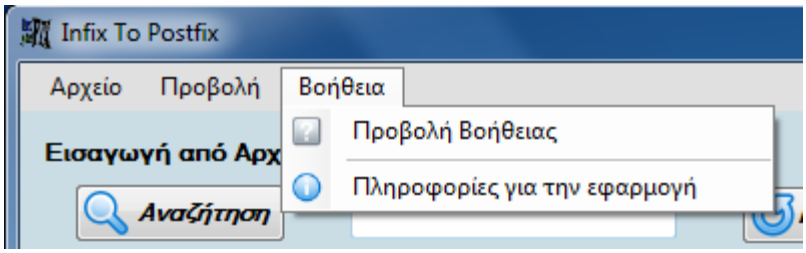


Εικόνα 17 Προβολή Θεωρητικού Μέρους

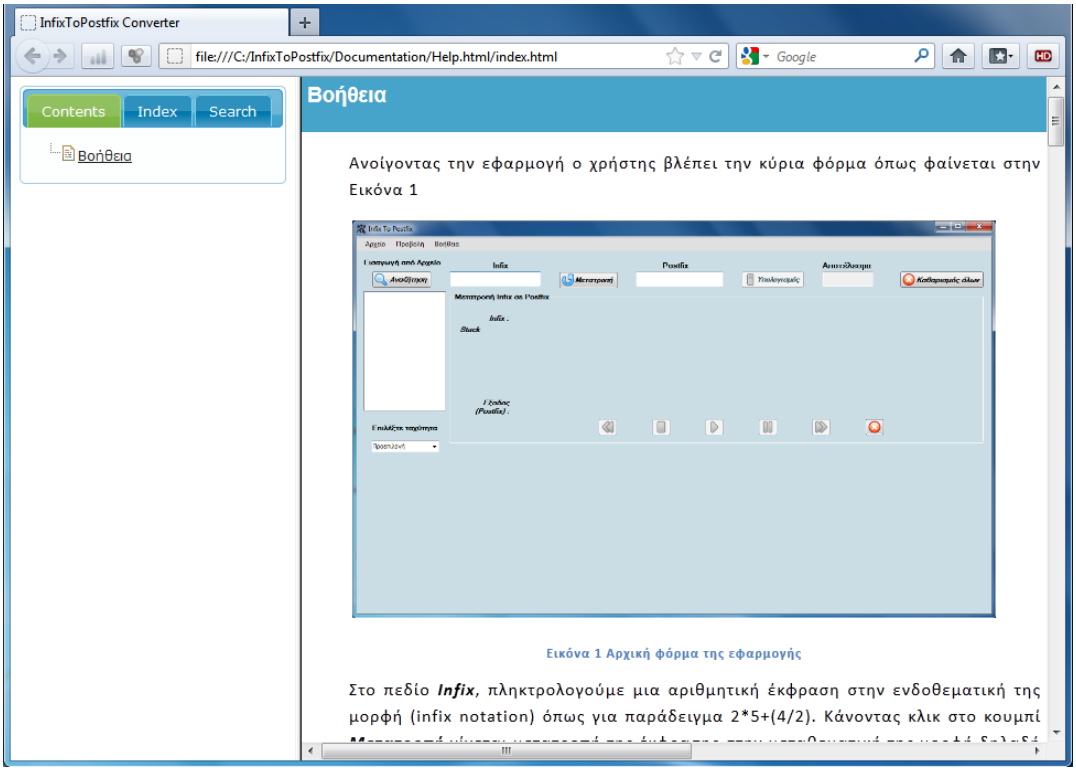


Εικόνα 18 Προβολή Αλγορίθμων και Κώδικα

Από το μενού ***Βοήθεια*** (βλ Εικόνα 19) μπορούμε να επιλέξουμε να δούμε στον προεπιλεγμένο φυλλομετρητή (browser) του υπολογιστή μας τη ***Βοήθεια*** (βλ Εικόνα 20) και κάποιες πληροφορίες σχετικά με την Εφαρμογή (βλ Εικόνα 21).



Εικόνα 19 Μενού "Βοήθεια"



Εικόνα 20 Προβολή Βοήθειας

Εικόνα 21 Σχετικά με την Εφαρμογή

6. Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.1. Συμπεράσματα

Η εφαρμογή InfixToPostfix Converter υλοποιήθηκε έχοντας ορισμένου κύριους στόχους. Οι πιο σημαντική από αυτούς ήταν:

- Μελέτη ενδοθεματικών (Infix) και μεταθεματικών (Posfix ή Reverse Polish Notation) εκφράσεων.
- Μελέτη αλγορίθμων μετατροπής ενδοθεματικών σε μεταθεματικές εκφράσεις και υπολογισμού μεταθεματικών εκφράσεων.
- Μελέτης της δομής της στοίβας, που χρησιμοποιείται ως βασική δομή δεδομένων για την υλοποίηση των προηγούμενων αλγορίθμων
- Υλοποίηση εφαρμογής που θα επιδεικνύει τις λειτουργίες μετατροπής και υπολογισμού βήμα-βήμα σε οπτικοποιημένο περιβάλλον

Οι παραπάνω στόχοι επιτεύχθηκαν σε ικανοποιητικό βαθμό, ωστόσο κατά τη διάρκεια κατασκευής και εξέλιξης του λογισμικού προέκυπταν συνεχώς νέες ανάγκες και στόχοι για νέες λειτουργίες. Χαρακτηριστικά παραδείγματα τέτοιων λειτουργιών είναι η εισαγωγή ενδοθεματικών εκφράσεων από αρχείο ή η επιλογή ταχύτητας της βηματικής εκτέλεσης των λειτουργιών μετατροπής και υπολογισμού.

6.2. Μελλοντικές επεκτάσεις

Παρ’ ότι η εφαρμογή έφτασε σε ένα ικανοποιητικό σημείο, θα μπορούσαν να γίνου ακόμη περισσότερες επεκτάσεις αλλά και διορθώσεις. Μερικές από τις πιο σημαντικές βελτιώσεις που θα μπορούσαν να γίνουν είναι:

- Σε περίπτωση που η ενδοθεματική έκφραση περιέχει μεταβλητές και όχι μόνο αριθμούς να ζητάτε από τον χρήστη με ανάλογο μήνυμα να δίνει τιμές στις μεταβλητές από το πληκτρολόγιο.
- Μετατροπή μιας μεταθεματικής έκφρασης σε ενδοθεματική. Να μπορεί δηλαδή να γίνετε αμφίδρομη μετατροπή όχι μόνο ενδοθεματικής σε μεταθεματική αλλά και μεταθεματικής σε ενδοθεματική.
- Απευθείας εισαγωγή μεταθεματικής έκφρασης και υπολογισμός της.
- Μετατροπή ενδοθεματικής έκφρασης σε προθεματική (Prefix) ανάλογα με την επιλογή του χρήστη. Δηλαδή να έχει τη δυνατότητα ο χρήστης να επιλέξει σε ποια από τις δύο μορφές, μεταθεματική ή προθεματική, θα μετατρέψει την ενδοθεματική έκφραση.

7. Βιβλιογραφία

[1] Χ. Σαμαρά, «Διαδικτυακός τόπος εξ αποστάσεως εκπαίδευσης πληροφορικής,» [Ηλεκτρονικό]. Available: http://users.sch.gr/vdrimtzias/index.php/2010-02-20-11-51-42/-3/2010-04-13-18-46-54. [Πρόσβαση 12 5 2012].

[2] Π. Α. Μαστοροκώστας, Προγραμματισμός Ι, Σέρρες: Τ.Ε.Ι. Σερρών, 2004.

[3] Ε. Γ. Ούτσιος, Αλγόριθμοι και δομές δεδομένων, Σέρρες: Τ.Ε.Ι. Σερρών, 2004.

[4] Wikipedia, "Polish notation," [Online]. Available: http://en.wikipedia.org/wiki/Polish_notation. [Accessed 13 5 2012].

[5] Wikipedia, "Stack," [Online]. Available: http://en.wikipedia.org/wiki/Stack_(abstract_data_type). [Accessed 12 5 2012].

[6] Ι. Γεωργουδάκης και Π. Μαστοροκώστας, Συνοπτικό εγχειρίδιο χρήσης του Microsoft Visual Studio 2010, Σέρρες: Τ.Ε.Ι. Σερρών, 2011, p. 16.

[7] Wikipedia, «Shunting-yard algorithm,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Shunting_yard_algorithm. [Πρόσβαση 19 05 2012].