Object Oriented Programming Course (CEID_NNY106)

Activity No 1

RPN Calculator σε C - Incremental Development

1. Στόχος

A) Εξοικείωση με:
- την Incremental Development τεχνική στην ανάπτυξη λογισμικού,
- την εφαρμογή RPNCalculator την οποία θα αναπτύξουμε στη συνέχεια με βάση την αντικειμενοστρεφή προσέγγιση.

B) Επανάληψη στη C η οποία είναι απαραίτητη για την μετάβαση στην Java.

Reverse Polish Notation Calculator

Η άσκηση βασίζεται στο παράδειγμα Reverse Polish Notation calculator που χρησιμοποιείται στο κεφάλαιο 8 "Οργάνωση προγράμματος" του βιβλίου «Διαδικαστικός προγραμματισμός – C». Την άσκηση μπορείτε να βρείτε στις παρακάτω πηγές:

1. Διαδικαστικός προγραμματισμός – C, Κ. Θραμπουλίδης, ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ & ΥΙΟΙ Α.Ε. (Κεφάλαιο 8 – Οργάνωση Προγράμματος)
2. Η Γλώσσα Προγραμματισμού C, Brian W. Kernighan, Dennis M. Ritchie, 2η/2008, Εκδόσεις ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ (κεφάλαιο 4)
3. Από τις Υπολογιστικές στις Κυβερνο-Φυσικές Διεργασίες και το IoT: Αντικείμενα και Υπηρεσίες, Κ. Θραμπουλίδης, ISBN 978-960-418-961-8, 2022, ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ & ΥΙΟΙ Α.Ε. https://sites.google.com/view/fromcomputationaltocyber-physi/home
4. στην ιστοσελίδα https://sites.google.com/view/objecttechnologycourse/courses-activities/activity-no-0

2. Οδηγίες Εκτέλεσης

Αν δεν είστε ήδη εξοικειωμένοι με την τεχνική της αυξητικής ανάπτυξης (incremental development) προγράμματος **θα πρέπει να το κάνετε το συντομότερο δυνατό**. Η δραστηριότητα αυτή έχει δομηθεί για να σας παρακινήσει να δουλέψετε υιοθετώντας την τεχνική αυτή.

Προσέξτε, η διαδικασία της αυξητικής ανάπτυξης του κώδικα σας δεν εφαρμόζεται μόνο σε επίπεδο εκδόσεων αλλά και στα πλαίσια της ανάπτυξης της κάθε έκδοσης, αν η λειτουργικότητα της έκδοσης είναι σύνθετη.

Με βάση τα παραπάνω σας συνιστώ να **εφαρμόσετε incremental development** για την ανάπτυξη μιας λειτουργούσας έκδοσης του προγράμματος RPNCalculator.

Την Δραστηριότητα θα δουλέψουμε βήμα προς βήμα στην 1η διάλεξη του μαθήματος. Εσείς θα συνεχίσετε την ανάπτυξη της, αξιοποιώντας την incremental development τεχνική, για να έχετε την δική σας λειτουργούσα έκδοση της εφαρμογής.

Προχωρήστε στην ανάπτυξη διαδοχικών εκδόσεων της εφαρμογής όπου η κάθε επόμενη θα προσθέτει επιπλέον λειτουργικότητα σε αυτήν της προηγούμενης έκδοσης.

3. Έκδοση No 1 (V1.0)

Ξεκινήστε με στόχο μια απλή έκδοση που θα υπολογίζει την τιμή πολύ απλών εκφράσεων με μονοψήφιους τελεστέους και με την υπόθεση πως ο χρήστης παρεμβάλει πάντα ένα κενό μεταξύ τελεστών και τελεστέων όπως στην παρακάτω απλή έκφραση

Απλή έκφραση: 3 6 + =

Α) Αξιοποιήστε το procedural abstraction και δουλέψτε σε πρώτη φάση μόνο με την main με βάση τον σκελετό που δώσαμε στην διάλεξη.

Ελέγξτε την σωστή λειτουργία της main πριν προχωρήσετε στην συγγραφή των άλλων συναρτήσεων (functions) της εφαρμογής.

Β) Δώστε την πιο απλή μορφή της getOp() η οποία θα σας επιτρέπει να ελέγξετε την σωστή λειτουργία της main().

Η εκτέλεση μια τέτοιας έκδοσης της εφαρμογής για είσοδο 3 6 + = θα μπορεί για παράδειγμα να έχει ως έξοδο την παρακάτω

```
RPNCalc started
putInStack
putInStack
add
presentResult
```

4. Έκδοση No 2 (V2.0)

Α) Αφού επιβεβαιώσετε τη σωστή λειτουργία της main(), προχωρήστε στην συγγραφή μιας getOp() που θα είναι ικανή να παίρνει τα στοιχεία της απλής έκφρασης.
Ελέγξτε τη λειτουργία της καθώς και την ορθή επικοινωνία της με την main().

Β) Προχωρήστε αναπτύσσοντας την πιο απλή στοίβα που θα σας δώσει τη δυνατότητα να έχετε μια λειτουργούσα έκδοση που θα υπολογίζει την τιμή της απλής έκφρασης μας.

Η εκτέλεση μια τέτοιας έκδοσης της εφαρμογής για είσοδο 3 6 + = θα μπορεί για παράδειγμα να έχει ως έξοδο την παρακάτω

```
RPNCalc started
3 6 + =
putInStack 3
putInStack 6
adding
res= 9
```

Η παρακάτω ενδεικτική έξοδος δίνει περισσότερη πληροφορία για τον τρόπο εκτέλεσης της εφαρμογής

```
Rpn Calculator V1
getOp() running
3 6 + =
input =3        51      inputType = 1
putInStack() executed
getOp() running
input =         32      inputType = 32
getOp() running
input =6        54      inputType = 1
putInStack() executed
getOp() running
input =         32      inputType = 32
getOp() running
input =+        43      inputType = 43
add() executed
6       3
getOp() running
input =         32      inputType = 32
getOp() running
input ==        61      inputType = 61
getOp() running
input =
        10      inputType = 10
getOp() running
```

Δώστε την εξήγηση σας για την μορφή της παραπάνω εξόδου.

5. Έκδοση No 3 (V3.0)

Α) Συνεχίστε προσθέτοντας για κάθε επόμενη έκδοση επιπλέον λειτουργικότητα.

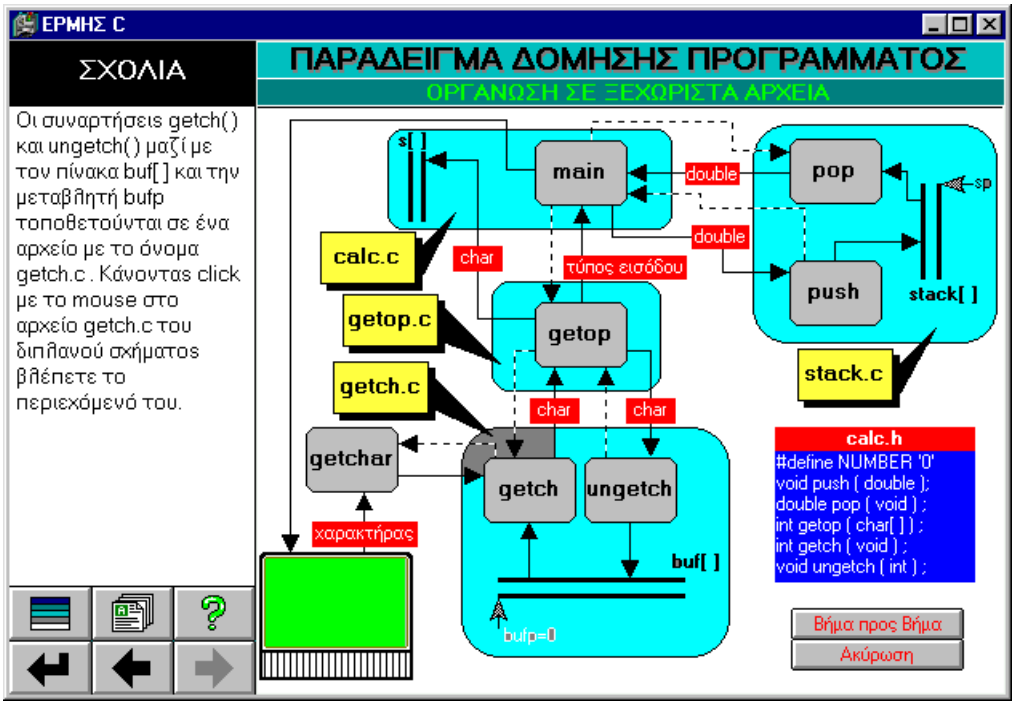
Β) Αξιοποιήστε την getOp() που θα βρείτε στις διαφάνειες ή στις πηγές που σας δόθηκαν στο πλαίσιο και προσπαθήστε να κατανοήσετε τον κώδικα και την λειτουργικότητα της.

Γ) Τροποποιήστε την getOp() ώστε να χρησιμοποιεί την getchar() αντί της getch(). Δώστε την έκφραση 12 10+= (δεν υπάρχει κενό μετά το 10) και παρατηρήστε την λειτουργικότητα της getOp(). Αυτό θα σας βοηθήσει να κατανοήσετε το λόγο χρήσης των getch() και ungetcth() από τους K&R.

6. Έκδοση No 4 (V4.0)

Κάντε μια προσπάθεια να δομήσετε το πρόγραμμα σας σε περισσότερα του ενός αρχεία.

Η παρακάτω εικόνα από το κεφάλαιο 8 του βιβλίου «Διαδικαστικός προγραμματισμός – C» δίνει ένα σενάριο δόμησης του κώδικα σε αρχεία. Είναι μια καλή άσκηση για οργάνωση προγράμματος. Το πρόγραμμα ΕΡΜΗΣ C μπορείτε να βρείτε στο συνοδευτικό CD του βιβλίου.



7. Έκδοση No 5 (V5.0)

Τροποποιήστε την εφαρμογή σας ώστε να υπολογίζει την τιμή εκφράσεων με τελεστέους δεκαεξαδικούς αριθμούς.

Ίσως στο τέλος νοιώσετε όπως ο ναυαγός που ανάβει φωτιά στο νησί.