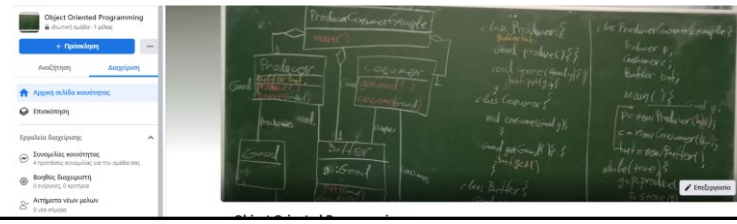
# Αντικειμενοστρεφής Προγραμματισμός (Object-Oriented Programming)

(CEID_NNY106)

## Προαπαιτούμενα: Functions and Program Structure (Δραστηριότητα No 1)

**FB Group**
https://www.facebook.com/groups/5894865063882886/



Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

# Εργαστηριακή Άσκηση No 1 – Στόχος

- Να θυμηθούμε την C.
- Να δούμε την διαδικασία ανάπτυξης λογισμικού.
- Να δούμε τους μηχανισμούς της C για χειρισμό πολυπλοκότητας.
- Να δούμε το πέρασμα στην ΟΟ προσέγγιση.

**Object Oriented Programming Course** (CEID_NNY106)
Activity No 1
RPN Calculator σε C - Incremental Development

### 1. Στόχος

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

Β) 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:

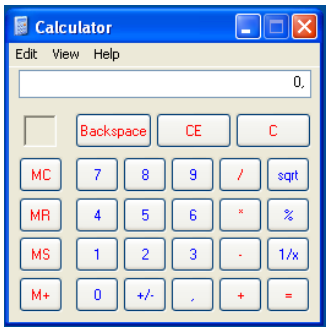
1. Διαδικαστικός προγραμματισμός – C, Κ. Θραμπουλίδης, ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ & ΥΙΟΙ Α.Ε. (Κεφάλαιο 8 – Οργάνωση Προγράμματος)
2. Η Γλώσσα Προγραμματισμού C, Brian W. Kernighan, Dennis M. Ritchie, 2η/2008, Εκδόσεις ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ (κεφάλαιο 4)
3. Από τις Υπολογιστικές στις Κυβερνο-Φυσικές Διεργασίες και 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.

# Εργαστηριακή Άσκηση No 1 – RPN Calculator

- Develop a program for the system to accept expressions based on the **Reverse Polish Notation** (RPN) and calculate their values.





free online calculator
http://www.math.com/students/calculators/source/scientific.htm

- Design Constraint
  - Stack has to be used to store operands.



# Reverse Polish Notation (RPN)

**Reverse Polish notation** (**RPN**), also known as (…,) **Polish postfix notation** or simply **postfix notation**, is a mathematical notation in which operators *follow* their operands, in contrast to Polish notation (PN), in which operators *precede* their operands. It does not need any parentheses as long as each operator has a fixed number of operands.

3 4 +
Postfix notation
("Reverse Polish")

In reverse Polish notation, the operators follow their operands; for instance, to add 3 and 4 together, one would write 3 4 + rather than 3 + 4. If there are multiple operations, operators are given immediately after their final operands (often an operator takes two operands, in which case the operator is written after the second operand); so the expression written 3 − 4 + 5 in conventional notation would be written 3 4 − 5 + in reverse Polish notation: 4 is first subtracted from 3, then 5 is added to it.
The concept of a stack, a last-in/first-out construct, is integral to these actions.

Source: Wikipedia

(10+6) * (18 - 16) ⟺ **10 6 + 18 16 - ***

# Στοίβα (Stack) – Last In First Out (LIFO)

**Stack of dishes**
adding or removing is only possible at the top.

**Stack (abstract data type)**
In computer science, a **stack** is an abstract data type that serves as a collection of elements, with two main operations:

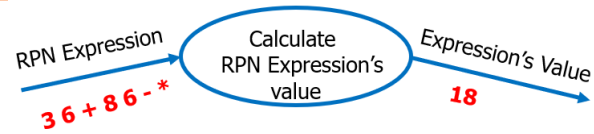
- **Push**, which adds an element to the collection, and
- **Pop**, which removes the most recently added element that was not yet removed.

Source: Wikipedia



# Development process

- Develop a program for the system **to accept expressions** based on the Reverse Polish Notation (RPN) and **calculate their values.**




Να γράψω την main()?

**Procedural Abstraction**
Identify basic processes

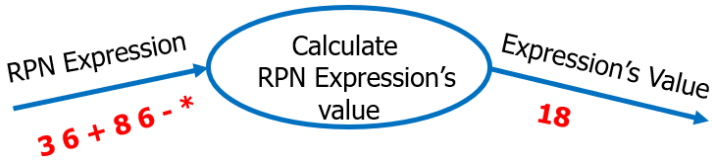
?

**Data Abstraction**
Data representation

Structured English of basic processes

```
main(){
....
...
}
```

# Identify basic processes




**1. Με ποια σειρά εκτελούνται οι συναρτήσεις;**
**2. Πως επικοινωνούν μεταξύ τους;**

## Βασικές διεργασίες

- Πάρε επόμενη συνιστώσα εισόδου (τελεστέο, τελεστή ή τέλος προγράμματος)
- Βάλε τελεστέο στην στοίβα
- Πάρε τελεστέο από την στοίβα
- Κάνε πρόσθεση
- Κάνε αφαίρεση
- Κάνε πολλαπλασιασμό
- Κάνε διαίρεση
- Δώσε το αποτέλεσμαστον χρήστη

## Functions

- getOp()
- putInStack()
- getFromStack()
- add()
- sub()
- mul()
- div()
- presentResult()

**main()**

```
main(){
....
...
}
```



# Structure English of program functions

**main()**

While the **next input** is not an indication of program termination
  In case it is a number
    put it in stack
  in case of +
    add
  in case of -
    sub
  ..
  in case of =
    present Result
  otherwise
    display error message.

**add()**
Get operands from stack
Apply the + operator
Put the result in stack

**sub()**
Get operands from stack
Apply the - operator
Put the result in stack

**presentResult()**
Get operand from stack
Display it