Πληροφορική 2

Αλγόριθμοι

Τι είναι αλγόριθμος;

- Αλγόριθμος είναι ένα διατεταγμένο σύνολο από σαφή βήματα το οποίο παράγει κάποιο αποτέλεσμα και τερματίζεται σε **πεπερασμένο** χρόνο.

- Ο αλγόριθμος δέχεται μια είσοδο, ακολουθεί η επεξεργασία της εισόδου και παράγει μια έξοδο.

ΠΕΠΕΡΑΣΜΕΝΟ = ΌΧΙ ΑΠΕΙΡΟ

Παράδειγμα αλγορίθμου

- Θέλουμε να κατασκευάσουμε έναν αλγόριθμο για την εύρεση του μεγαλύτερου ακεραίου από μια λίστα θετικών ακεραίων.
- Ο αλγόριθμος θα πρέπει να μπορεί να βρίσκει τον μεγαλύτερο ακέραιο από μια λίστα ακεραίων οποιουδήποτε μεγέθους (5, 1.000, 10.000, 1.000.000 κ.λπ.)
- Ο αλγόριθμος πρέπει να είναι γενικός και να μην εξαρτάται από το πλήθος των ακεραίων.

Ο αλγόριθμος δέχεται ως είσοδο μια λίστα από Ν ακεραίους.

Βήμα 1. Ο αλγόριθμος ορίζει ένα στοιχείο δεδομένων με όνομα ΜΕΓΙΣΤΟΣ και του δίνει ως τιμή τον πρώτο ακέραιο της λίστας

Βήμα 2. Αν ο επόμενος ακέραιος είναι μεγαλύτερος από την τιμή που υπάρχει στο ΜΕΓΙΣΤΟΣ όρισε τον ως νέα τιμή του ΜΕΓΙΣΤΟΣ

Βήμα 3. Αν η λίστα των ακεραίων έχει και άλλα στοιχεία πήγαινε στο Βήμα 2

Βήμα 4. Παρουσίασε ως έξοδο την τιμή του ΜΕΓΙΣΤΟΣ

Αριθμητικό παράδειγμα

ΛΙΣΤΑ ΤΙΜΩΝ {13, 6, 17, 14, 15}

ΜΕΓΙΣΤΟΣ = 13

6 > ΜΕΓΙΣΤΟΣ ➔ ΌΧΙ

17 > ΜΕΓΙΣΤΟΣ ➔ ΝΑΙ ➔ ΜΕΓΙΣΤΟΣ = 17

14 > ΜΕΓΙΣΤΟΣ ➔ ΌΧΙ

15 > ΜΕΓΙΣΤΟΣ ➔ ΌΧΙ

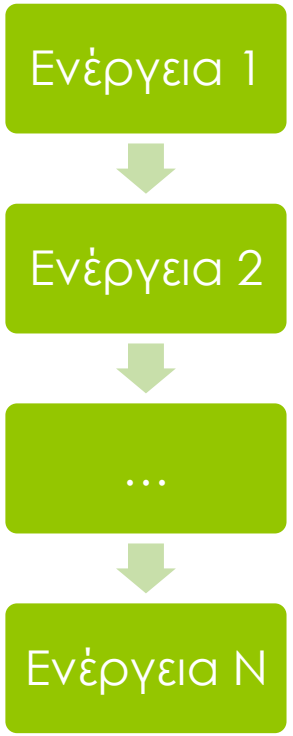
Συνεπώς ΜΕΓΙΣΤΟΣ = 17

Βασικές αλγοριθμικές δομές

- **Δομή ακολουθίας** (sequence)
  - Οι εντολές εκτελούνται στην σειρά η μια μετά την άλλη. Για να εκτελεστεί μια εντολή θα πρέπει να έχει ολοκληρωθεί η εκτέλεση της προηγούμενης.
- **Δομή απόφασης** (selection)
  - Αφορά περιπτώσεις που ελέγχεται μια συνθήκη. Αν το αποτέλεσμα του ελέγχου είναι η τιμή αληθής τότε εκτελείται μια ακολουθία εντολών ενώ αν είναι ψευδής εκτελείται μια διαφορετική ακολουθία εντολών.
- **Δομή επανάληψης** (repetition)
  - Επιτρέπει την επανάληψη της ίδιας ακολουθίας εντολών.

Δομή ακολουθίας

- **Οι εντολές εκτελούνται στην σειρά η μια μετά την άλλη**
- Για να εκτελεστεί μια εντολή θα πρέπει να έχει ολοκληρωθεί η εκτέλεση της προηγούμενης
- Χρησιμοποιείται στην λύση απλών προβλημάτων



Δομή απόφασης

- **Αφορά περιπτώσεις που ελέγχεται μια συνθήκη**
- Αν το αποτέλεσμα του ελέγχου είναι η τιμή αληθής (true) τότε εκτελείται μια ακολουθία εντολών ενώ αν είναι ψευδής (false) εκτελείται μια διαφορετική ακολουθία εντολών

Αν συνθήκη **τότε**

σειρά ενεργειών 1

Αλλιώς

σειρά ενεργειών 2

Τέλος_αν

Συνθήκη είναι μια λογική έκφραση που μπορεί να είναι είτε αληθής είτε ψευδής (π.χ. ταχύτητα > 60)

Δομή επανάληψης

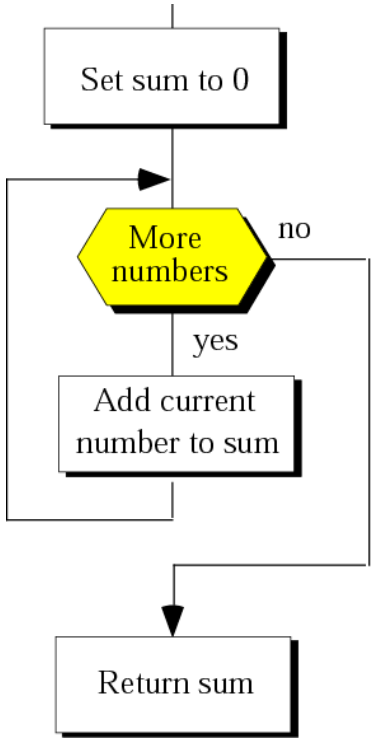
- **Επιτρέπει την επανάληψη της ίδιας ακολουθίας εντολών**
- Η δομή επανάληψης ονομάζεται και **βρόχος επανάληψης** (loop)

Όσο συνθήκη **επανάλαβε**
σειρά ενεργειών
Τέλος_επανάληψης

Τρόποι αναπαράστασης αλγορίθμων

- **Διάγραμμα Ροής** (flowchart)
  - Είναι μια σχηματική παράσταση ενός αλγορίθμου. Δεν περιέχει τις λεπτομέρειες του αλγορίθμου αλλά προσπαθεί να δώσει την γενική ιδέα του.
- **Ψευδοκώδικας** (pseudo code)
  - Είναι μια αναπαράσταση ενός αλγορίθμου **σε φυσική γλώσσα** (απλό κείμενο). Αν και δεν υπάρχει κάποιο πρότυπο για τον ψευδοκώδικα σε γενικές γραμμές θυμίζει την σύνταξη μιας πραγματικής γλώσσας προγραμματισμού όπως η PASCAL.
- **Κωδικοποίηση** (coding)
  - Αφορά την αναπαράσταση ενός αλγορίθμου σε μια πραγματική γλώσσα προγραμματισμού όπως η C, η Java, η Basic κοκ **τηρώντας όλους τους συντακτικούς κανόνες που επιβάλει η γλώσσα**.

Παράδειγμα διαγράμματος ροής



- Παράδειγμα άθροισης μιας ακολουθίας αριθμών

Ψευδοκώδικας

- **Μεταβλητή**: Ποσότητα που η τιμή της μπορεί να μεταβάλλεται
- **Εκχώρηση**: Αναθέτει το αποτέλεσμα μιας έκφρασης σε μια μεταβλητή
  - Χ←2+3
- **Εντολή εισόδου**: Δέχεται μια τιμή από τον χρήστη
  - Διάβασε x
- **Εντολή εξόδου**: Εμφανίζει ένα αποτέλεσμα στην μονάδα εξόδου (οθόνη, εκτυπωτής κ.α.)
  - Εμφάνισε x

Παράδειγμα ψευδοκώδικα

```
Αλγόριθμος Άθροισμα
sum←0
k←1
Όσο k <= 10 επανάλαβε
        Διάβασε x
        sum ← sum + x
        k ← k + 1
Τέλος_επανάληψης
Εμφάνισε sum
Τέλος Άθροισμα
```

- Παράδειγμα άθροισης δέκα τιμών που εισάγονται από τον χρήστη

Παράδειγμα κωδικοποίησης σε γλώσσα C++

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
int sum, k, x;
sum=0;
k=1;
while (k<=10)
{
                cin >> x;
                sum = sum + x;
                k = k + 1;
}
cout << "RESULT" << sum;
system("pause");
return 0;
}
```

- Παράδειγμα άθροισης δέκα τιμών που εισάγονται από το χρήστη

Αντιμετάθεση μεταβλητών (swap)

- **Λανθασμένη αντιμετάθεση**

```
…
Διάβασε α, β
α ← β;
β ← α;
Εμφάνισε α, β
…
```

- Λάθος! οι τιμές των α και β στο τέλος θα είναι ίδιες

- Έστω ότι ο χρήστης δίνει στο α την τιμή 5 και στο β την τιμή 10

α	β
5	10
10	

- Άρα οι τιμές και των 2 μεταβλητών όταν εκτελεστούν οι εντολές είναι 10

Αντιμετάθεση μεταβλητών (swap)

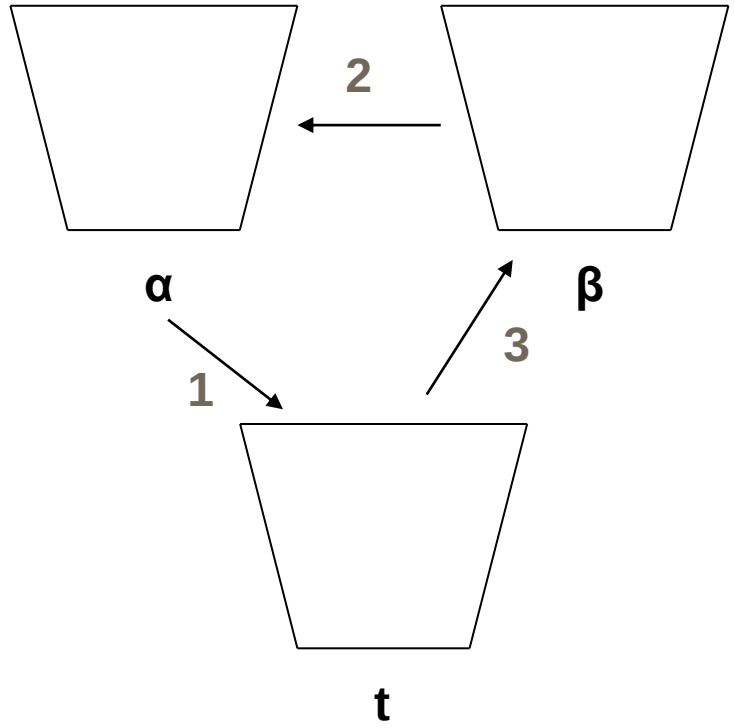
- **Σωστή αντιμετάθεση:**

με την χρήση βοηθητικής μεταβλητής

…

```
Διάβασε α, β
t ← α;
α ← β;
β ← t;
Εμφάνισε α, β
```

…



Παράδειγμα 1 (Δομή ακολουθίας)

- Αλγόριθμος που υπολογίζει το άθροισμα 2 ακεραίων

```
Αλγόριθμος Θ1
Διάβασε α, β
γ←α+β
Εμφάνισε γ
Τέλος Θ1
```

Παράδειγμα 2 (Δομή απόφασης)

- Αλγόριθμος που διαβάζει ένα βαθμό, αποφασίζει αν είναι προβιβάσιμος και εμφανίζει («Προάγεται») ή («Απορρίπτεται») αντίστοιχα.

```
Αλγόριθμος Θ2
Διάβασε βαθμός
Αν βαθμός >=5 τότε
        Εμφάνισε "Προάγεται"
Αλλιώς
        Εμφάνισε "Απορρίπτεται"
Τέλος_αν
Τέλος Θ2
```

Παράδειγμα 3 (Δομή επανάληψης)

- Αλγόριθμος που βρίσκει τη μεγαλύτερη ανάμεσα σε τιμές που δίνει ο χρήστης

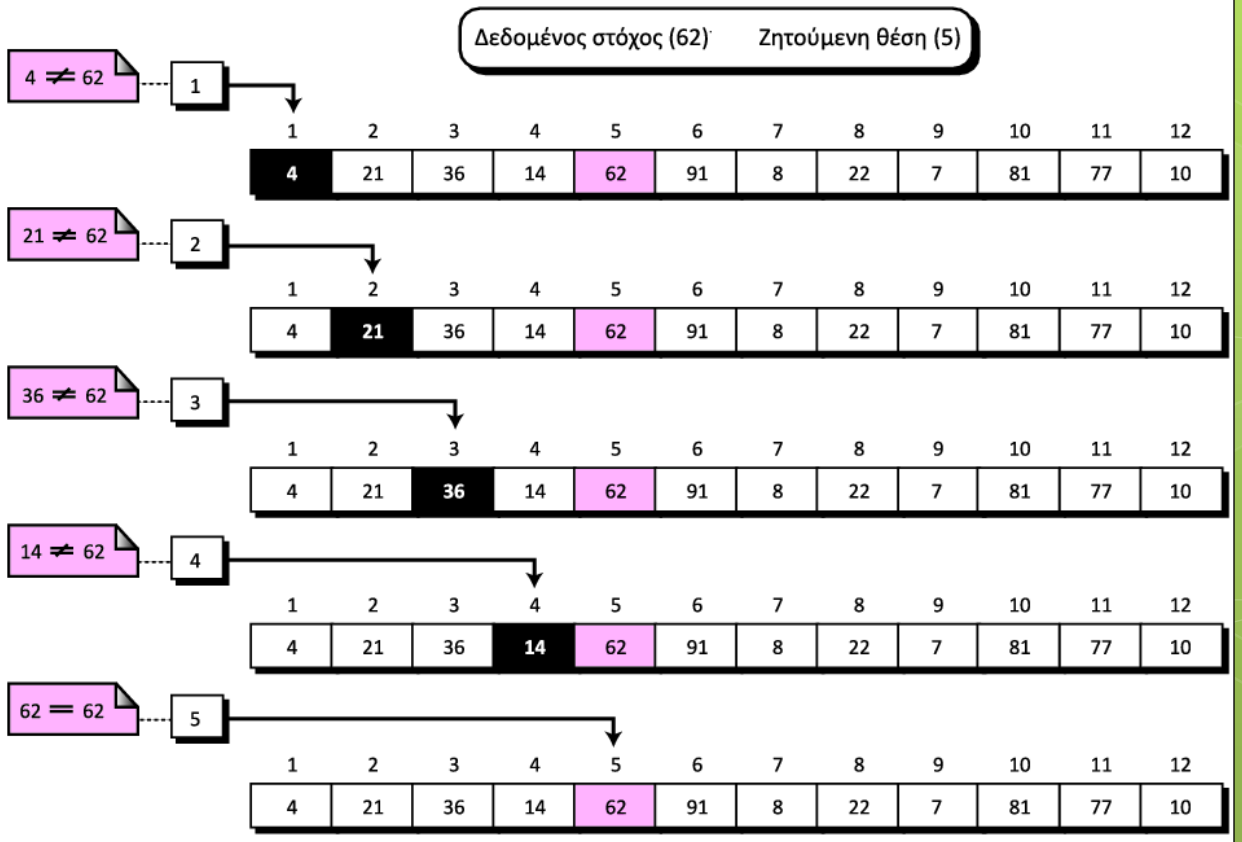
```
Αλγόριθμος Θ3
Διάβασε x
max←x
Όσο x<>0 επανάλαβε
        Αν x>max τότε
                max←x
        Τέλος_αν
        Διάβασε x
Τέλος_επανάληψης
Εμφάνισε max
Τέλος Θ3
```

Βασικοί αλγόριθμοι

- Άθροιση
- Γινόμενο
- Ελάχιστο και μέγιστο
- Ταξινόμηση
- Αναζήτηση

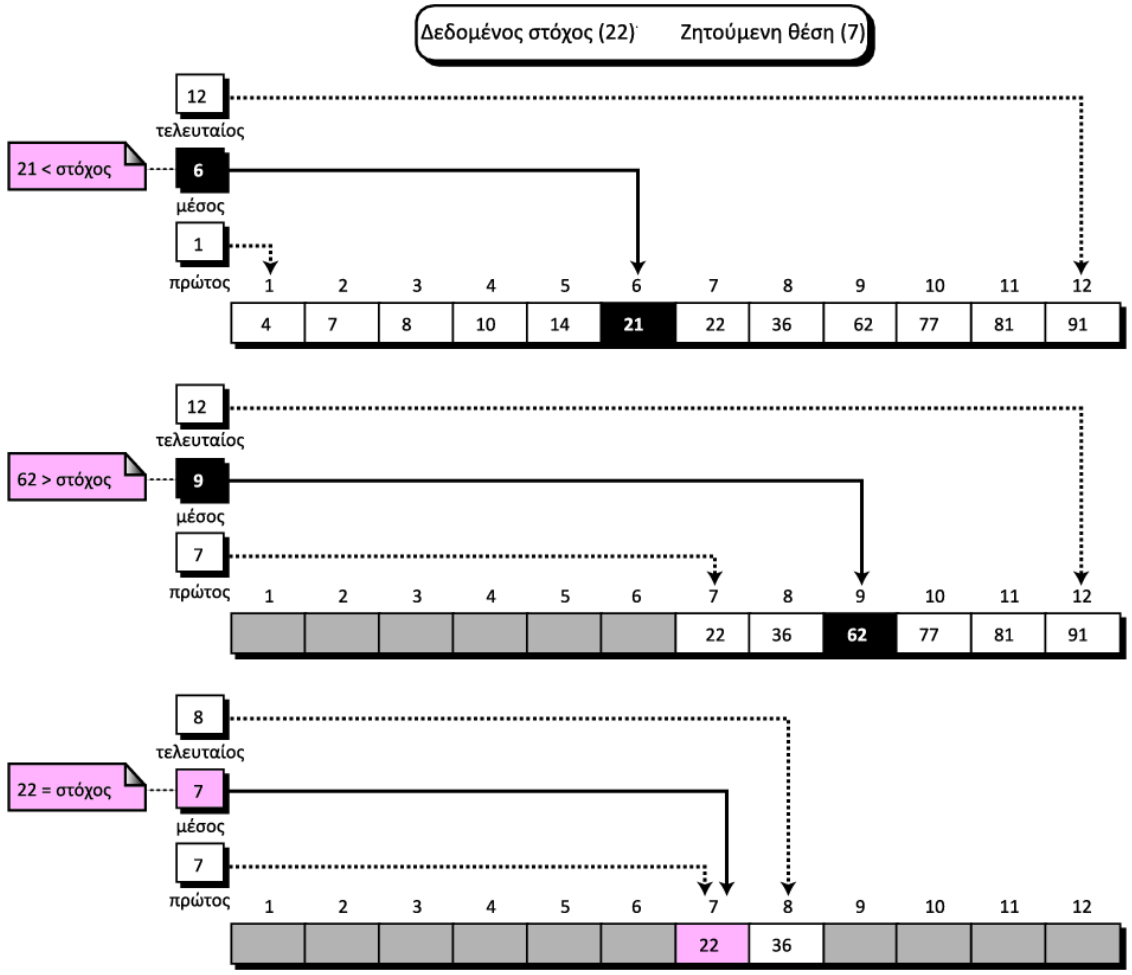
Ακολουθιακή αναζήτηση

- Αναζήτηση είναι η διαδικασία εντοπισμού της θέσης ενός στοιχείου σε μια λίστα στοιχείων
- Η ακολουθιακή αναζήτηση εφαρμόζεται όταν η λίστα δεν είναι ταξινομημένη
- Εξετάζει όλα τα στοιχεία ένα προς ένα μέχρι να βρεθεί το στοιχείο που ψάχνουμε ή να φτάσουμε στο τέλος της λίστας
- Δεν είναι αποδοτικός τρόπος αναζήτησης



Δυαδική αναζήτηση

- Απαιτεί η λίστα να είναι ταξινομημένη
- Είναι αποδοτικός τρόπος αναζήτησης
- Ελέγχει το μεσαίο στοιχείο της ταξινομημένης λίστας
- Εντοπίζει αν ο στόχος είναι στο δεξί μισό ή στο αριστερό μισό
- Απορρίπτει το μισό στο οποίο δεν μπορεί να βρίσκεται ο στόχος
- Επαναλαμβάνει την διαδικασία για το τμήμα που μένει



Ταξινόμηση φυσαλίδας (bubble sort)

- Η λίστα διαιρείται σε 2 υπό-λίστες (ταξινομημένη αριστερά και αταξινόμητη δεξιά)
- Γίνονται συγκρίσεις διαδοχικών στοιχείων από το τέλος προς την αρχή και αντιμετάθεση στοιχείων αν δεν είναι στην σωστή σειρά
- Το μικρότερο στοιχείο της αταξινόμητης υπό-λίστας αναδύεται προς τα αριστερά

Είσοδος: 25,13,8,17,9

|25,13,8,17,9
|25,13,8,9,17
|25,13,8,9,17
|25,8,13,9,17
8,|25,13,9,17
8,|25,13,9,17
8,|25,9,13,17
8,9,|25,13,17
8,9,|25,13,17
8,9,13,|25,17
8,9,13,17,25|

Ταξινόμηση με επιλογή (selection sort)

- Η λίστα διαιρείται σε 2 υπό-λίστες (ταξινομημένη αριστερά και αταξινόμητη δεξιά)
- Θεωρούμε ότι οι δύο λίστες χωρίζονται από ένα φανταστικό τείχος
- Βρίσκουμε το μικρότερο στοιχείο της αταξινόμητης λίστας και το τοποθετούμε στην αρχή των αταξινόμητων δεδομένων.
- Το στοιχείο που αντικαθιστά παίρνει την θέση που είχε το μικρότερο στοιχείο.
- Επεκτείνουμε την ταξινομημένη λίστα κατά ένα στοιχείο προς τα δεξιά

```
Είσοδος: 25,33,18,17,11,23

|25,33,18,17,11,23
|11,33,18,17,25,23

11,|33,18,17,25,23
11,|17,18,33,25,23

11,17,|18,33,25,23
11,17,|18,33,25,23

11,17,18,|33,25,23
11,17,18,|23,25,33

11,17,18,23,|25,33
11,17,18,23,|25,33

11,17,18,23,25,|33
11,17,18,23,25,|33

11,17,18,23,25,33|
```

Ταξινόμηση με εισαγωγή (insertion sort)

- Η λίστα διαιρείται σε 2 υπό-λίστες (ταξινομημένη αριστερά και αταξινόμητη δεξιά)
- Επιλέγεται το πρώτο στοιχείο της αταξινόμητης λίστας και τοποθετείται στην σωστή θέση στην ταξινομημένη λίστα μετακινώντας όσα στοιχεία χρειαστεί προς τα δεξιά

```
Είσοδος: 25,13,8,17,9
|25,13,8,17,9
Επιλογή του 25
25,|13,8,17,9
Επιλογή του 13
13,25,|8,17,9
Επιλογή του 8
8,13,25,|17,9
Επιλογή του 17
8,13,17,25,|9
Επιλογή του 9
8,9,13,17,25|
```

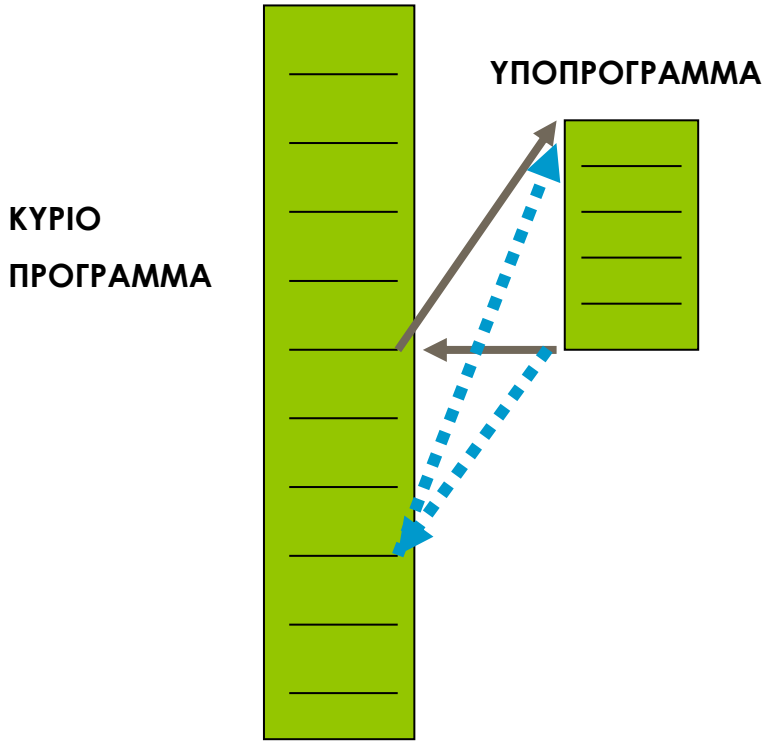
Υποαλγόριθμοι

- Ένας αλγόριθμος μπορεί να χωριστεί σε μικρότερες μονάδες που ονομάζονται υποαλγόριθμοι
- Το ίδιο ισχύει και για τους υποαλγόριθμους

- Πλεονεκτήματα υποαλγορίθμων
  - Ευκολότερη κατανόηση αλγορίθμου
  - Συντομότερη ανάπτυξη αλγορίθμου

Δομημένος προγραμματισμός

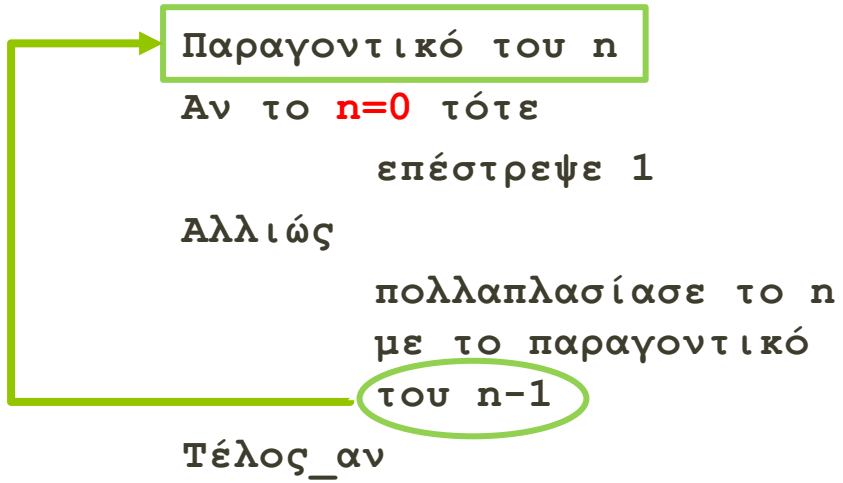
- Ο δομημένος προγραμματισμός ορίζει ότι ένας αλγόριθμος θα πρέπει να χωρίζεται σε μικρότερες υπομονάδες που ονομάζονται υποπρογράμματα.
- Κάθε υποπρόγραμμα με την σειρά του διαιρείται σε μικρότερα υποπρογράμματα μέχρι τα υποπρογράμματα να γίνουν στοιχειώδη και συνεπώς εύκολα υλοποιήσιμα



Αναδρομή (recursion)

- Αναδρομή ονομάζεται η διαδικασία κατά την οποία ένας αλγόριθμος καλεί τον εαυτό του
- Οι αναδρομικοί αλγόριθμοι συνήθως λύνουν «κομψά» ένα πρόβλημα

Παράδειγμα:
Υπολογισμός παραγοντικού ενός ακέραιου n

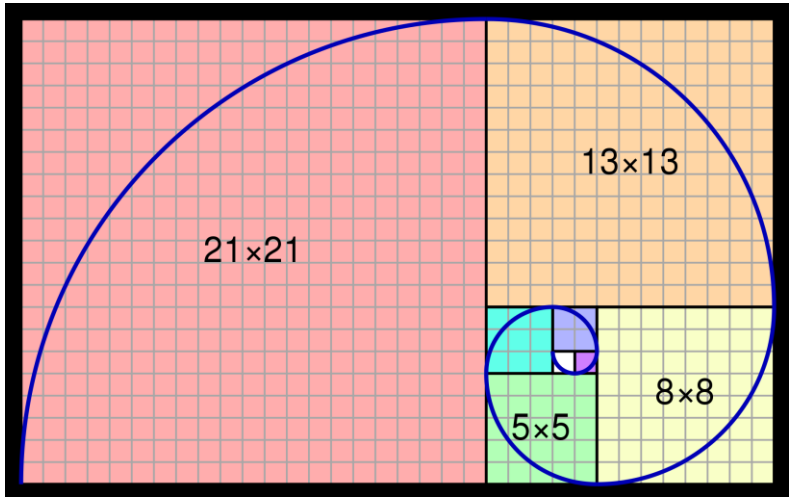


Παραγοντικό του n (συμβολίζεται n!)
Αν το n=0 τότε είναι 1
Αν το n>0 τότε είναι n*(n-1)*(n-2)*…*3*2*1

π.χ. 5! = 5*4*3*2*1=120

Αναδρομή (recursion)

- Ένα επίσης πολύ γνωστό παράδειγμα αριθμοσειράς που προκύπτει εύκολα με αναδρομή είναι οι αριθμοί Fibonacci:



Παράδειγμα:
Υπολογισμός σειράς Fibonacci ακέραιου n

```
Fibonacci του n
Αν το n=0 τότε
        επέστρεψε 0
Αλλιώς Αν το n=1 τότε
        επέστρεψε 1
Αλλιώς
        επέστρεψε
(Fibonacci(n-1)+Fibonacci(n-2))
Τέλος_αν
```

Οι αριθμοί Fibonacci ορίζονται από την αναδρομική σχέση
F(0)=0, F(1)=1, F(n)=(F(n-1)+F(n-2)) όταν n>2