Σχεδίαση αλγορίθμων

2η γνωστική ενότητα : σχεδίαση αλγορίθμων

1. Ανασκόπηση αλγοριθμικών δομών ελέγχου
2. Στοιχειοποίηση
3. Αναδρομή

Παράδειγμα καθορισμένου πλήθους επαναλήψεων

```
{υπολόγισε το παραγοντικό ενός φυσικού αριθμού N >=1}
start
    μετρητής := 1
    παραγοντικό := 1
    while μετρητής ≤ N
        παραγοντικό := παραγοντικό * μετρητής
        μετρητής := μετρητής +1
    end
end
```

- μπορεί να υλοποιηθεί με προ-ελεγχόμενη επανάληψη

Παράδειγμα καθορισμένου πλήθους επαναλήψεων (συνέχ.)

```
{υπολόγισε το παραγοντικό ενός φυσικού αριθμού N >=1}
start
    μετρητής := 0
    παραγοντικό := 1
    repeat
        μετρητής := μετρητής +1
        παραγοντικό := παραγοντικό * μετρητής
    until  μετρητής ≤ N
end
```

- μπορεί να υλοποιηθεί με μετα-ελεγχόμενη επανάληψη και διαφορετική αρχικοποίηση τιμών

Η δομή της καθορισμένης επανάληψης

- καθορισμένη επανάληψη : μια δομή ελέγχου η οποία συσχετίζει ένα σύνολο αλγοριθμικών βημάτων (σώμα επανάληψης) με ένα σύνολο στιγμιότυπων ενός αντικειμένου ή τιμών μιας μεταβλητής (τιμές επανάληψης)

- συμβολισμός μιας δομής καθορισμένης επανάληψης :

```
for τιμές επανάληψης
        σώμα επανάληψης
end
```

Παράδειγμα καθορισμένης επανάληψης

```
{υπολόγισε το παραγοντικό ενός φυσικού αριθμού N >=1}
start
    παραγοντικό := 1
    for μετρητής = 1 έως N
        παραγοντικό := παραγοντικό * μετρητής
    end
end
```

- οι τιμές επανάληψης μπορούν να παρατίθενται εξαντλητικά ή, για βαθμωτές μεταβλητές με διακριτές και διατεταγμένες τιμές (π.χ. φυσικοί αριθμοί), να υπονοούνται

Αξιολόγηση της δομής καθορισμένης επανάληψης

- επιτρέπει την αναπαράσταση με πεπερασμένο πλήθος δηλώσεων ενός καθορισμένου πλήθους αλγοριθμικών βημάτων

- η εκτέλεση μιας δομής καθορισμένης επανάληψης τερματίζεται πάντοτε, μόλις ολοκληρωθεί η επανάληψη για όλες τις τιμές

- η λίστα των τιμών επανάληψης πρέπει να περιέχει όλες και μόνο εκείνες τις τιμές για τις οποίες απαιτείται επανάληψη
- χρειάζεται προσοχή στις οριακές (πρώτες και τελευταίες) τιμές επανάληψης

Ισοδυναμία των δομών επανάληψης

- οι δομές μετα-ελεγχόμενης και προ-ελεγχόμενης επανάληψης είναι
  α) εκφραστικά ισοδύναμες μεταξύ τους, και
  β) εκφραστικά ισχυρότερες από την καθορισμένη επανάληψη

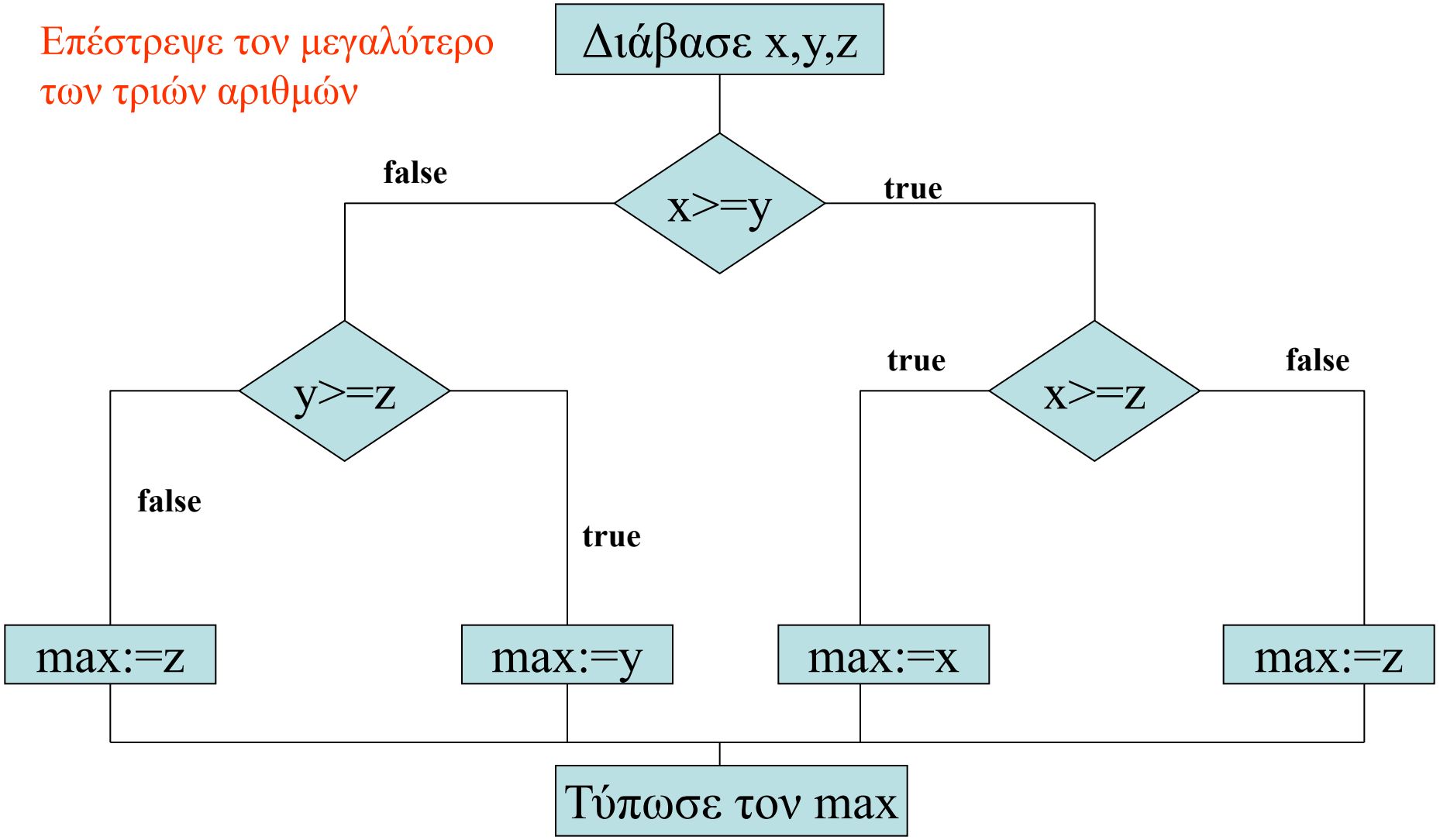
- κάθε μία από τις δομές επανάληψης διευκολύνει ως προς
  α) την οικονομία της αναπαράστασης, και
  β) την αποφυγή λαθών σε συγκεκριμένες περιπτώσεις

Εμφωλευμένες δομές επανάληψης

- το σώμα μιας δομής επανάληψης μπορεί να περιέχει το ίδιο δομές επανάληψης χαμηλότερου επιπέδου
- δομές επανάληψης μπορούν να εμφωλευτούν σε κλάδους δομών επιλογής και αντιστρόφως δομές επιλογής μπορούν να εμφωλευτούν σε σώματα δομών επανάληψης
- η σημασιολογία εκτέλεσης μιας δομής επανάληψης δε δημιουργεί εναλλακτικά μονοπάτια
- η χρήση ή εμφώλευση μιας δομής επανάληψης δε μεταβάλλει το πλήθος αλλά αυξάνει το μήκος των διαδρομών εκτέλεσης (το πλήθος των εκτελούμενων βημάτων)

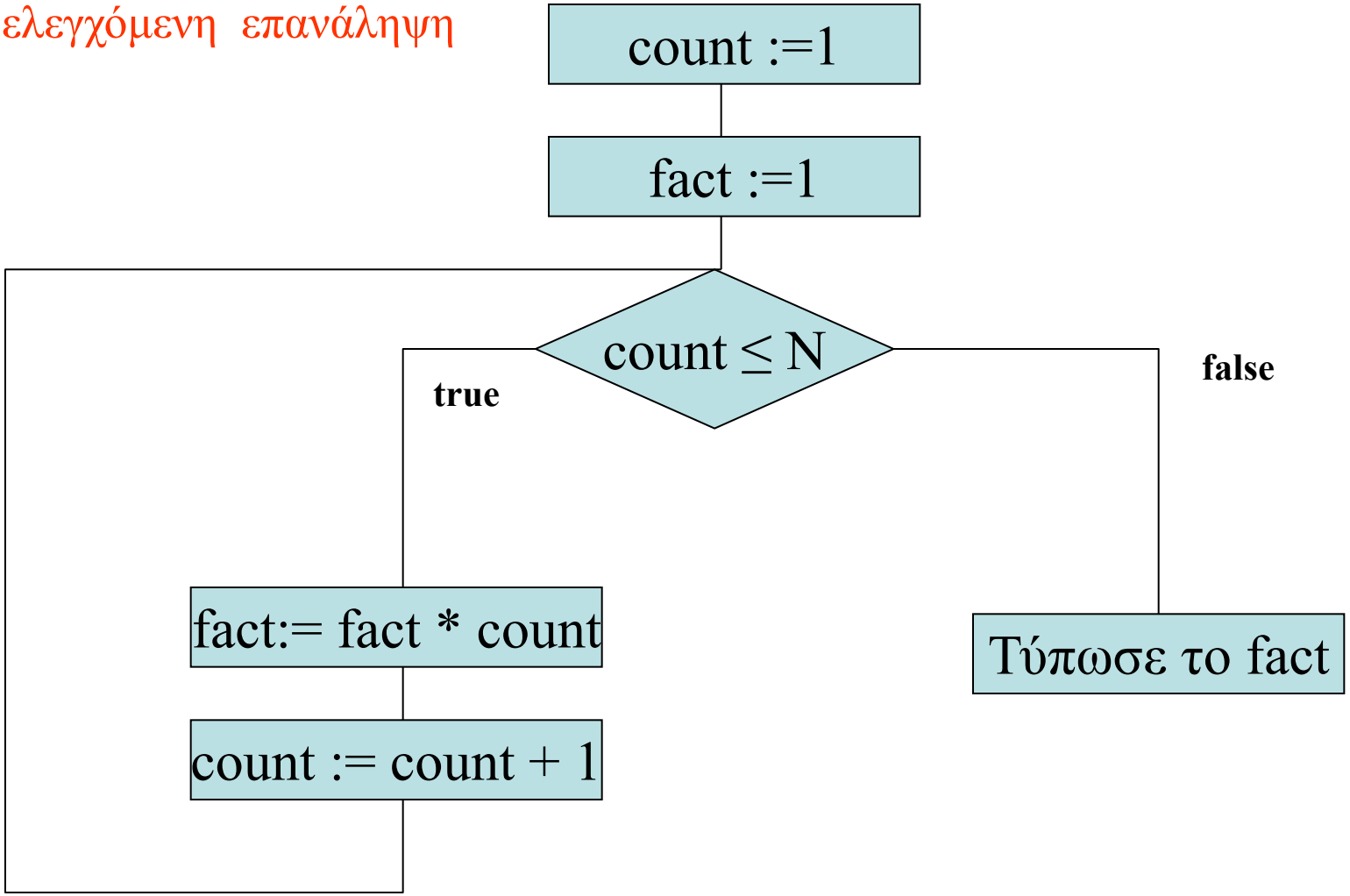
Διαγράμματα ροής

Επέστρεψε τον μεγαλύτερο των τριών αριθμών



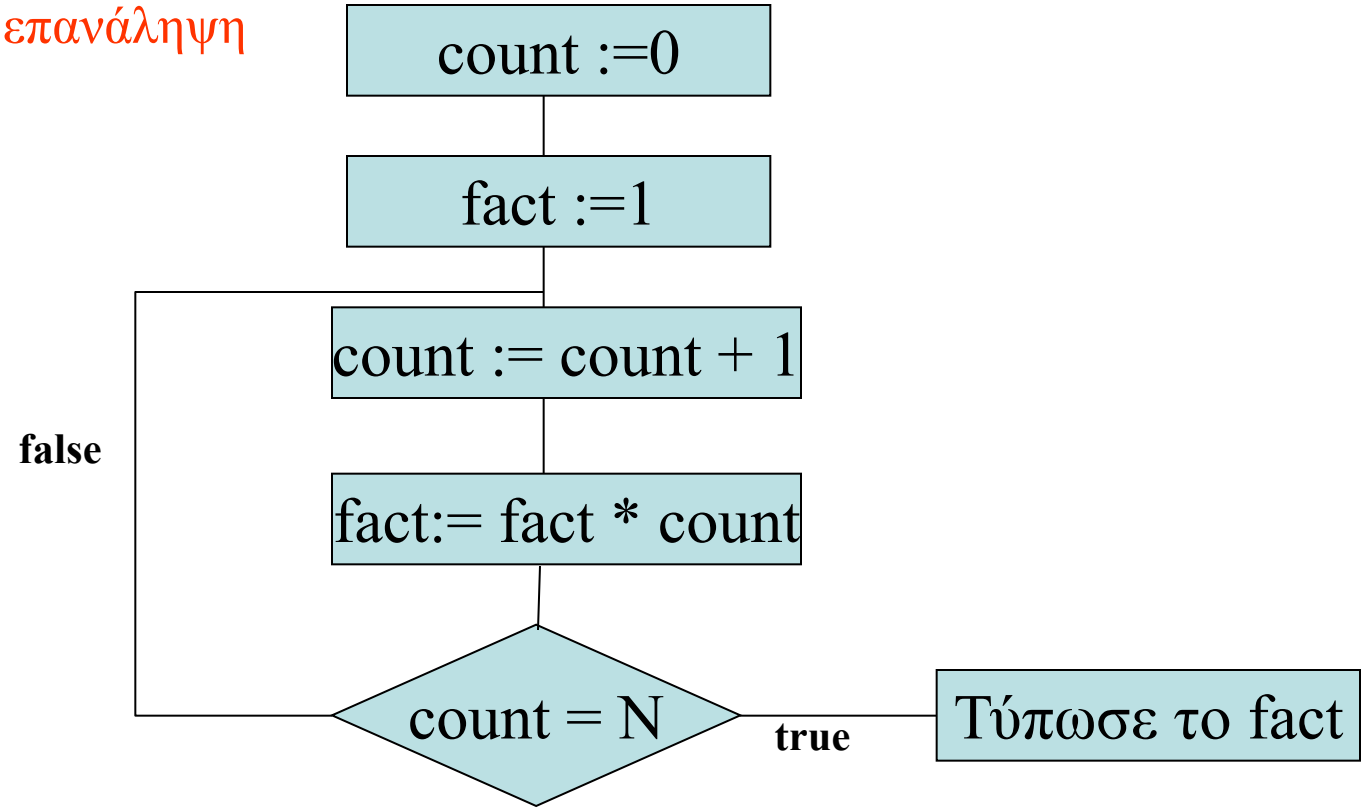
Διαγράμματα ροής

Υπολογισμός του παραγοντικού N
με προ-ελεγχόμενη επανάληψη



Διαγράμματα ροής

Υπολογισμός του παραγοντικού N
με μετα-ελεγχόμενη επανάληψη



Εκφραστική ισχύς δομών ελέγχου

- έχει αποδειχθεί (Dijkstra) ότι οι τρεις δομές :
  α) ακολουθίας,
  β) διπλής επιλογής και
  γ) ελεγχόμενης ακαθόριστης επανάληψης
  είναι ικανές να εκφράσουν, με κατάλληλους συνδυασμούς και εμφωλεύσεις, οποιονδήποτε υπολογίσιμο αλγόριθμο

- οποιεσδήποτε άλλες δομές δεν προσθέτουν σε εκφραστική ισχύ αλλά διευκολύνουν σε
  α) οικονομία αναπαράστασης
  β) αποφυγή σφαλμάτων μεταγραφής του αλγορίθμου σε πρόγραμμα

Παράδειγμα : ταξινόμηση

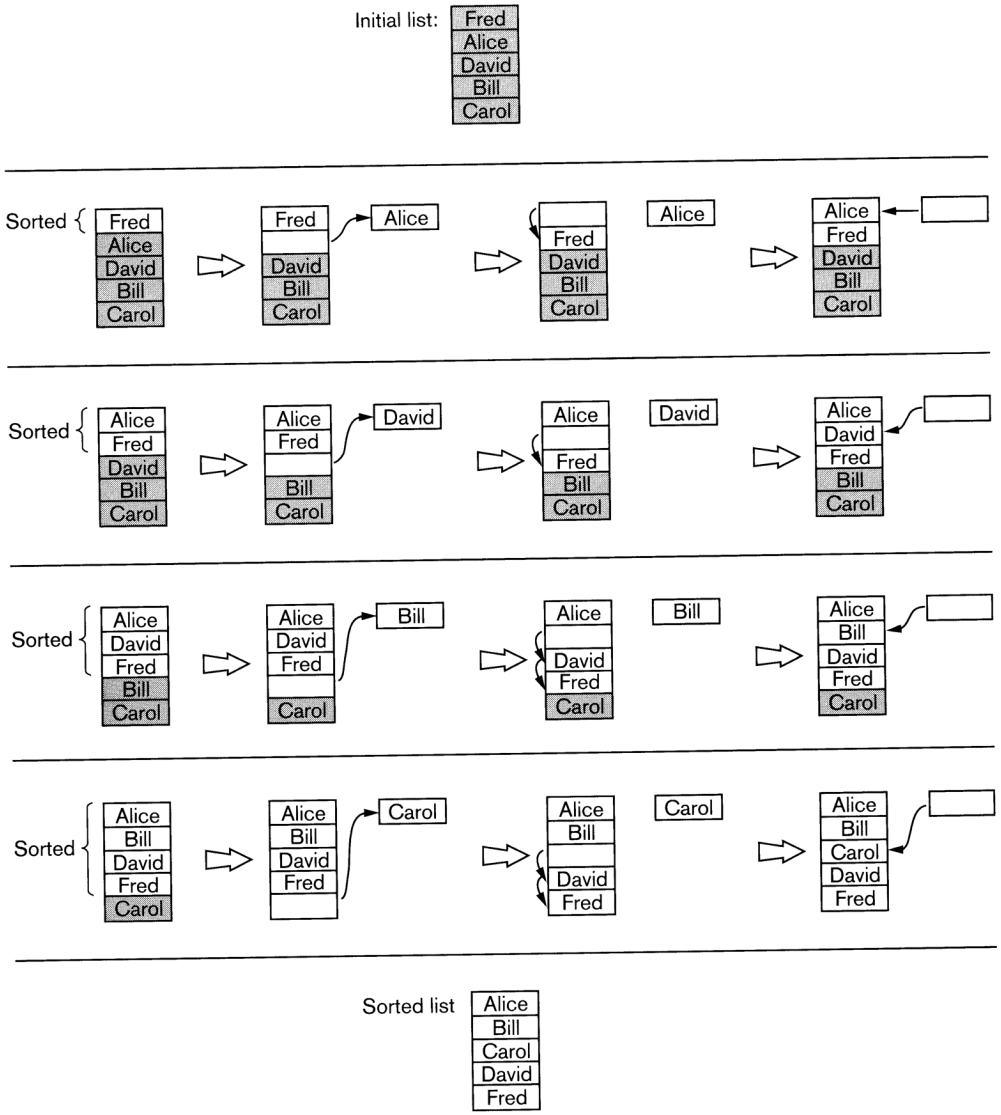
- ταξινόμηση : η διευθέτηση των εγγραφών μιας λίστας κατά τη διάταξη των τιμών ενός από τα πεδία τους (πεδίο ταξινόμησης)

- οι τιμές του πεδίου ταξινόμησης
  α) μπορεί να είναι συνεχείς (π.χ. πραγματικοί αριθμοί) ή διακριτές (π.χ. φυσικοί αριθμοί),
  β) μπορεί να προέρχονται από ένα πεπερασμένο (π.χ. ονοματεπώνυμα) ή άπειρο (π.χ. φυσικοί αριθμοί) πεδίο τιμών
  γ) πρέπει να έχουν κάποια μοναδική φυσική διάταξη ως προς την οποία μπορεί να γίνει η ταξινόμηση

παράδειγμα : ταξινόμηση (συνέχ.)

- η ταξινόμηση ως προς κάποιο πεδίο μπορεί να είναι αύξουσα ή φθίνουσα
- απλή ταξινόμηση : η ταξινόμηση μιας λίστας ως προς ένα και μόνο πεδίο
- σύνθετη ταξινόμηση : η ταξινόμηση μιας λίστας ως προς 2 ή περισσότερα πεδία με μια συγκεκριμένη διάταξη

Αλγόριθμος ταξινόμησης παρεμβολής



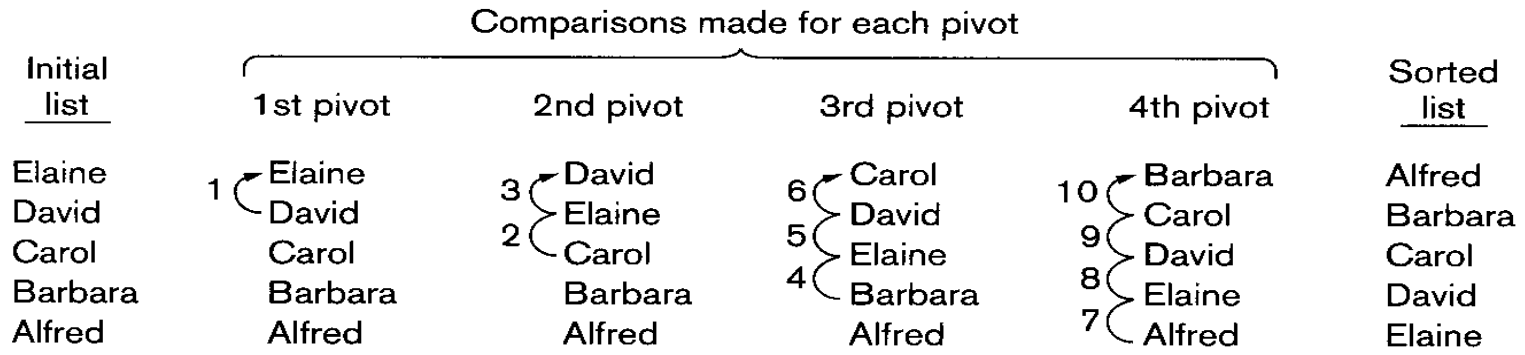
Αλγόριθμος ταξινόμησης παρεμβολής

start
θέσε στην μεταβλητή Ν την τιμή 2
while (η τιμή Ν <= του μήκους της λίστας)
επέλεξε το Νοστό στοιχείο της λίστας
μετέφερε το περιεχόμενο της θέσης Ν σε προσωρινή θέση
while (υπάρχει μεγαλύτερο όνομα πάνω από τη Ν)
μετακίνησε το όνομα μια θέση κάτω
τοποθέτησε το περιεχόμενο της προσωρινής θέσης στην κενή θέση
θέσε στο Ν την τιμή Ν+1
end

Αποδοτικότητα του αλγόριθμου ταξινόμησης παρεμβολής

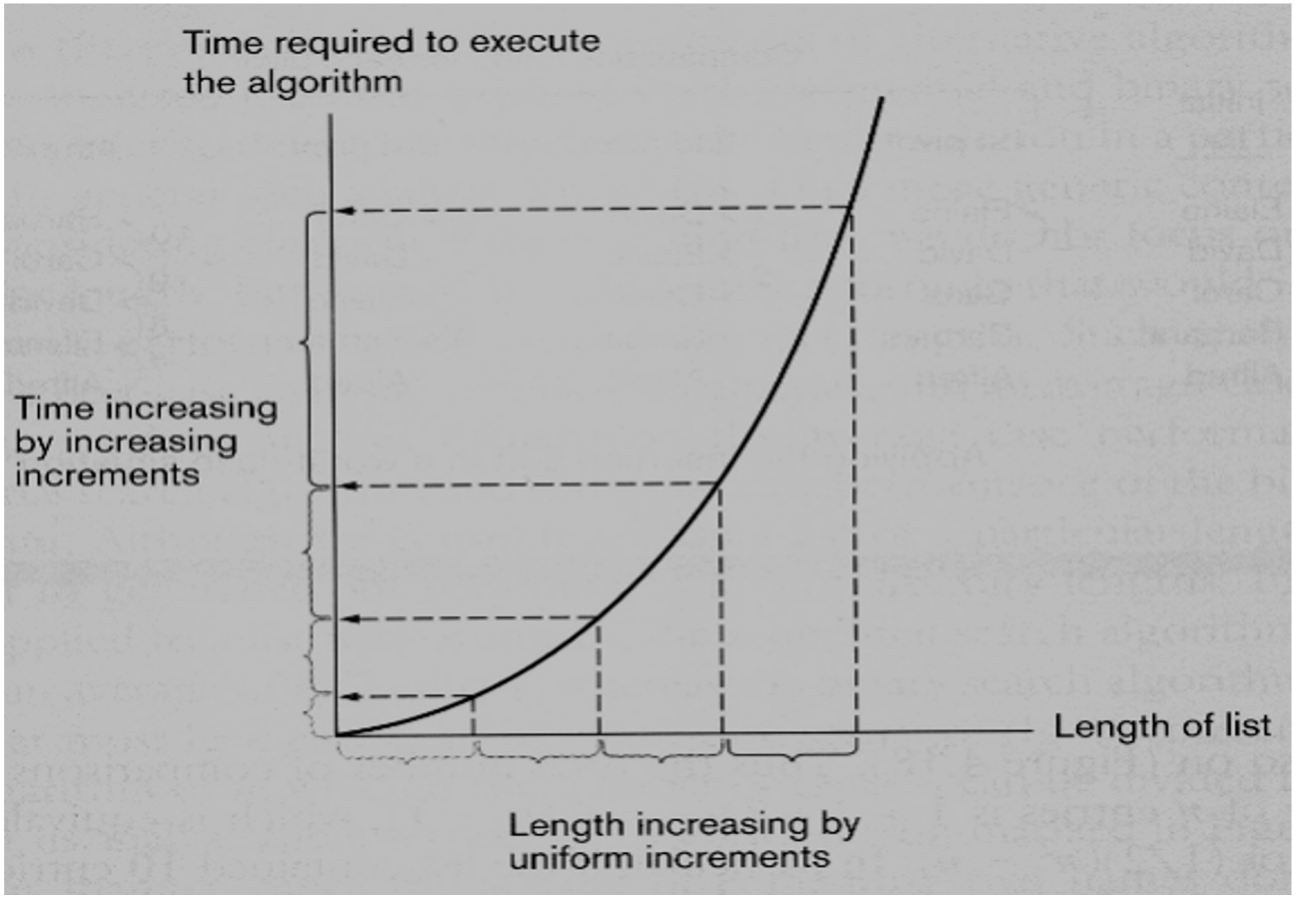
- Κυρίαρχο στοιχείο αποτελεί ο αριθμός συγκρίσεων κάθε θέσης με τα παραπάνω στοιχεία
- Καλύτερη περίπτωση: η λίστα είναι ταξινομημένη και άρα απαιτούνται μόνο N-1 συγκρίσεις
- Χειρότερη περίπτωση: σε κάθε θέση πρέπει να συγκρίνουμε με όλα τα παραπάνω στοιχεία.

Αποδοτικότητα του αλγόριθμου ταξινόμησης με εισαγωγή



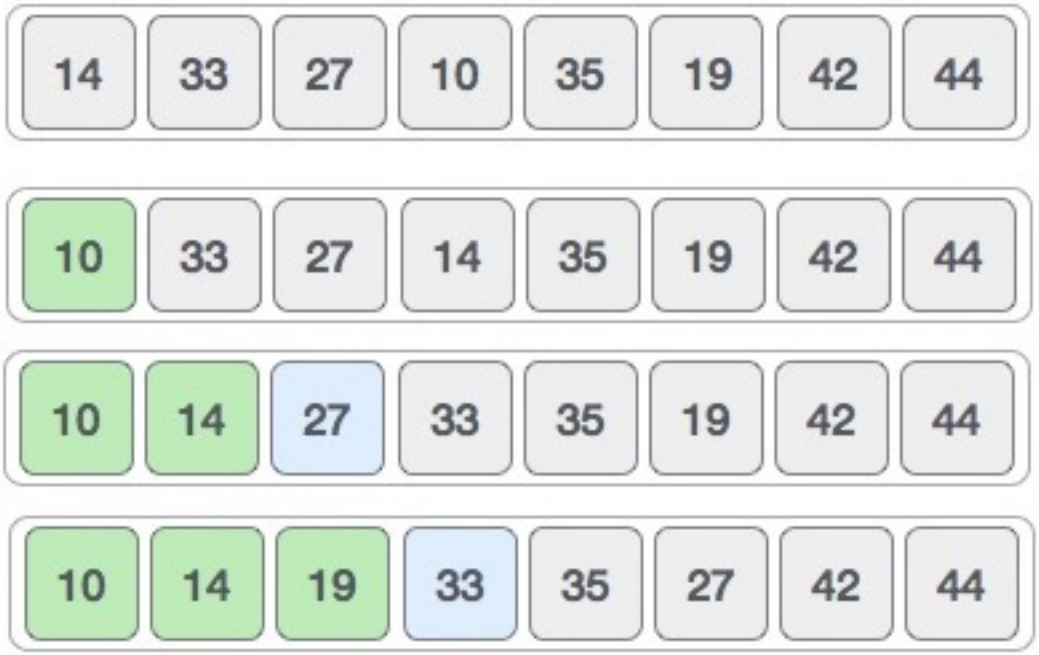
- Αριθμός συγκρίσεων για μία λίστα με N στοιχεία: 1 + 2 + 3 + ... + (N-1) = N*(N-1)/2
- Για παράδειγμα σε μία λίστα 10 στοιχείων θα χρειαζόμασταν 45 συγκρίσεις.

Αποδοτικότητα του αλγόριθμου ταξινόμησης με εισαγωγή



Εναλλακτικός Αλγόριθμος ταξινόμησης

- Επιλέγει την μικρότερη τιμή και τη βάζει στην πρώτη θέση
- Επιλέγει τη δεύτερη μικρότερη τιμή και τη βάζει στη δεύτερη θέση κ.ο.κ



Ταξινόμηση με τον αλγόριθμο φυσαλίδας

- έστω μια λίστα ονομάτων την οποία θέλουμε να ταξινομήσουμε αλφαβητικά σε αύξουσα διάταξη
- κάνουμε επαναληπτικά περάσματα της λίστας και σε κάθε πέρασμα, συγκρίνουμε κάθε όνομα με το επόμενό του :
  α) εάν το επόμενο έπεται και αλφαβητικά, το αφήνουμε έτσι και προχωράμε
  β) εάν το επόμενο προηγείται αλφαβητικά, αντιμεταθέτουμε τα δύο ονόματα και προχωράμε
- έτσι, τα ονόματα που πρέπει να προηγηθούν ανεβαίνουν σταδιακά (κατά μία θέση σε κάθε πέρασμα) μέχρι τη σωστή θέση τους
- τα περάσματα σταματούν όταν η λίστα βρεθεί ταξινομημένη

7.4. παράδειγμα ταξινόμησης φυσαλίδας

αρχική λίστα	λίστα μετά το 1ο πέρασμα (4 αλλαγές)	λίστα μετά το 2ο πέρασμα (3 αλλαγές)	λίστα μετά το 3ο πέρασμα (2 αλλαγές)
Ιωάννης	Ιωάννης	Γιώργος	Βασίλης
Κατερίνα	Γιώργος	Βασίλης	Γιώργος
Γιώργος	Βασίλης	Ιωάννης	Ιουλία
Βασίλης	Κατερίνα	Ιουλία	Ιωάννης
Στέλιος	Ιουλία	Κατερίνα	Κατερίνα
Ιουλία	Μαρία	Μαρία	Μαρία
Μαρία	Στέλιος	Στέλιος	Στέλιος

Ανάλυση του αλγορίθμου φυσαλίδας

```
{ταξινόμηση μιας λίστας με τον αλγόριθμο φυσαλίδας}
start
    while η λίστα δεν είναι ταξινομημένη
          κάνε ένα πέρασμα με αντιμεταθέσεις όπου χρειάζεται
    end
end
```

- ανάγκη για ανάλυση σώματος επανάληψης

Ανάλυση του αλγορίθμου φυσαλίδας (συνέχ.)

```
start
    while η λίστα δεν είναι ταξινομημένη
        πάρε το πρώτο όνομα της λίστας
        repeat
          if το όνομα έπεται αλφαβητικά από το επόμενο
          στη λίστα
                then αντιμετάθεσε το όνομα αυτό με το επόμενο
                πάρε το επόμενο όνομα της λίστας
        until να φτάσεις στο τέλος της λίστας
    end
end
```

- έλλειψη πληροφορίας για προ-έλεγχο συνθήκης

Ανάλυση του αλγορίθμου φυσαλίδας (συνέχ.)

```
start
    repeat
        πάρε το πρώτο όνομα της λίστας
        repeat
          if το όνομα έπεται αλφαβητικά από το επόμενο
          στη λίστα
                then αντιμετάθεσε το όνομα αυτό με το επόμενο
                        σημείωσε ότι έγινε αλλαγή στο πέρασμα αυτό
                πάρε το επόμενο όνομα της λίστας
        until να φτάσεις στο τέλος της λίστας
    until να μην έχει γίνει αλλαγή στο πέρασμα αυτό
τέλος
```

Τροποποίηση του σχεδιασμού σε βήματα

- η επιλογή μιας δομής ελέγχου σε ένα στάδιο της ανάλυσης σε βήματα μπορεί να αλλάξει σε επόμενο στάδιο, βλέποντας τις επί μέρους λεπτομέρειες

- συντηρητικός σχεδιασμός :
  α) αναβολή των σχεδιαστικών αποφάσεων για το στάδιο λεπτομέρειας που είναι απαραίτητες
  β) επανεξέταση σε κάθε βήμα των σχεδιαστικών αποφάσεων του προηγούμενου βήματος

Πλήθος βημάτων του αλγορίθμου φυσαλίδας

- καλύτερη περίπτωση : αν η λίστα είναι εξ' αρχής ταξινομημένη τότε θα χρειαστεί ένα μόνο πέρασμα
- χειρότερη περίπτωση : αν το τελευταίο στοιχείο τις λίστας πρέπει να φτάσει στην πρώτη θέση τότε θα χρειαστούν N-1 περάσματα, όπου N το μήκος της λίστας
- κατά μέσο όρο θα χρειαστούν N/2 περάσματα

- σε κάθε πέρασμα ελέγχονται N-1 διαδοχικά ζεύγη και κάποια από αυτά αντιμετατίθενται

- το συνολικό πλήθος βημάτων είναι κατά μέσο όρο (N-1)*N/2 άρα είναι ανάλογο του N^2

Στοιχειοποίηση (modularity)

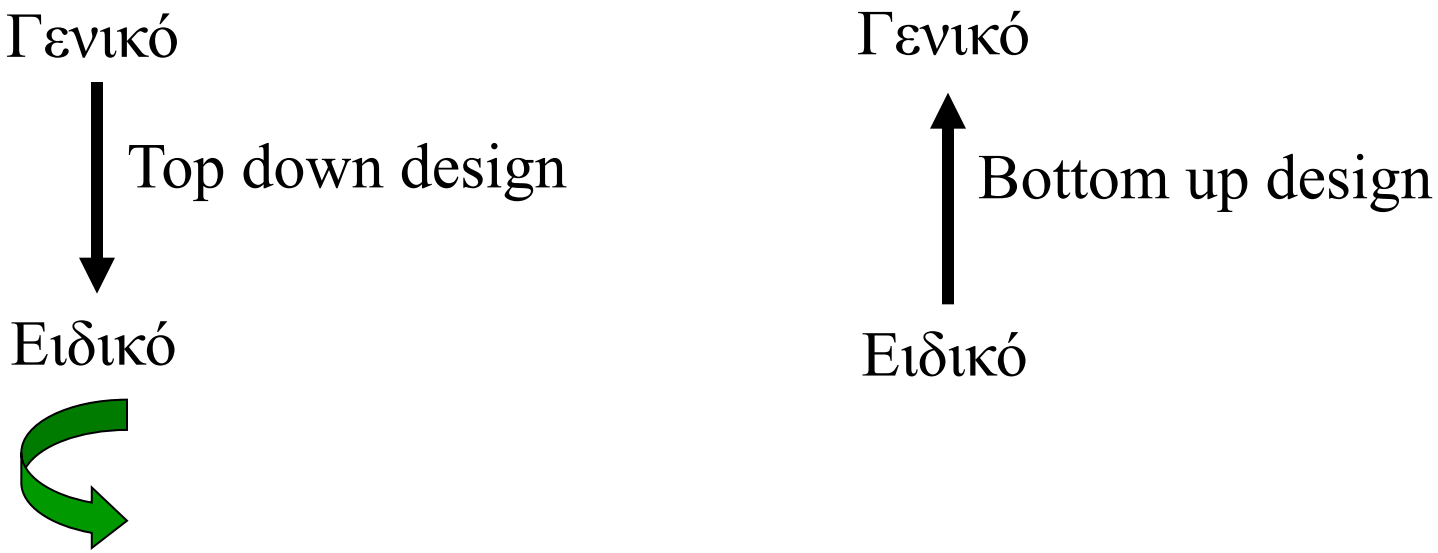
- ανάλυση ενός αλγορίθμου σε βήματα: η διατύπωσή του σε βήματα αρκετά λεπτομερή ώστε να μπορούν να εκτελεστούν από έναν επεξεργαστή
- συνθήκες τερματισμού της ανάλυσης κατά βήματα: παραγωγή (α) απλών και (β) εκτελέσιμων βημάτων

- κάθε ένα από τα παραγόμενα βήματα έχει έννοια εντός του συγκεκριμένου αλγορίθμου

Η έννοια του τμήματος

- τμήμα αλγορίθμου : ένα σύνολο αλγοριθμικών βημάτων το οποίο
  α) επιτελεί μία συγκεκριμένη λειτουργία
  β) δέχεται συγκεκριμένες εισόδους
  γ) παράγει συγκεκριμένες εξόδους
  δ) τεκμηριώνεται ως προς τη λειτουργία, τις εισόδους και τις εξόδους του
- η ανάλυση κατά βήματα μπορεί να οδηγεί στην παραγωγή τμημάτων
- ένα τμήμα αλγορίθμου εμφανίζει αυτοτέλεια κατανόησης και λειτουργίας ως προς τον κύριο αλγόριθμο από τον οποίο χρησιμοποιείται

Στοιχειοποίηση (modularity)



Δημιουργία plug-in διαδικασιών
που είναι ανεξάρτητες από τον κύριο αλγόριθμο

Παράδειγμα στοιχειοποίησης: Σχεδιασμός εικόνας από ρομπότ

1. Δυνατότητα μετακίνησης σε ένα χαρτί
2. Έχει μολύβι που μπορεί να το κατεβάζει για να ζωγραφίζει
3. Μπορεί να εκτελέσει τις εξής εντολές

μπροστά (Ε) : κίνηση της κεφαλής προς τα εμπρός κατά Ε εκατοστά

αριστερά (Μ) : περιστροφή της κεφαλής προς τα αριστερά κατά Μ μοίρες

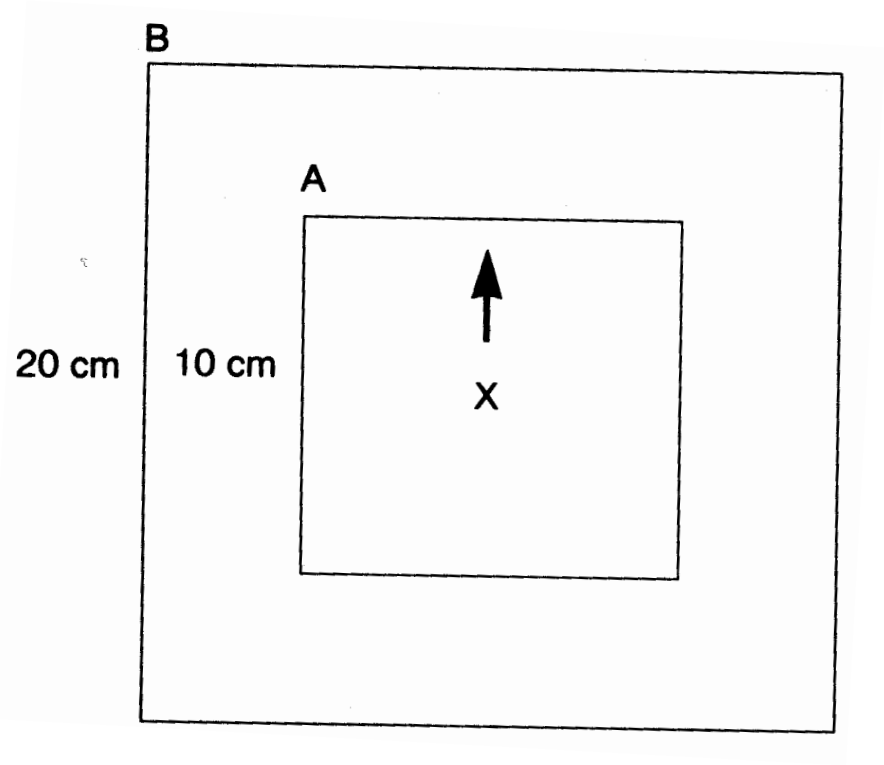
δεξιά (Μ) : περιστροφή της κεφαλής προς τα δεξιά κατά Μ μοίρες

πένα πάνω : ανέβασμα της πένας

πένα κάτω : κατέβασμα της πένας

Παράδειγμα στοιχειοποίησης: Σχεδιασμός εικόνας από ρομπότ

Ζητούμενο σχέδιο



Ανάλυση του αλγορίθμου κατά βήματα

{σχεδίαση δύο ομόκεντρων τετραγώνων}
1. κινήσου στο σημείο Α
2. σχεδίασε ένα τετράγωνο με διάσταση πλευράς 10 εκατοστά
3. κινήσου στο σημείο Β
4. σχεδίασε ένα τετράγωνο με διάσταση πλευράς 20 εκατοστά

- βήματα 2 και 4 : η ίδια διαδικασία, η οποία κατανοείται και εκτελείται αυτοτελώς σε σχέση με τον κύριο αλγόριθμο
- γενική διαδικασία :
  σχεδίαση τετραγώνου με διάσταση πλευράς Δ εκατοστά

Ανάλυση του αλγορίθμου κατά βήματα

στοιχείο σχεδίαση τετραγώνου (διάσταση πλευράς)

```
{ Σχεδιάζει ένα τετράγωνο με μήκος πλευράς = διάσταση
  πλευράς εκφρασμένο σε εκατοστά. Το τετράγωνο
  σχεδιάζεται
  ξεκινώντας από την αρχική θέση και προσανατολισμό της
  κεφαλής. Η κεφαλή επιστρέφει στην αρχική θέση και
  προσανατολισμό, με την πένα πάνω. }
start
        πένα κάτω
        for μετρητής = 1 έως 4
                μπροστά (διάσταση πλευράς)
                αριστερά (90)
        πένα πάνω
end
```

Ανάλυση του αλγορίθμου κατά βήματα

{σχεδίαση δύο ομόκεντρων τετραγώνων}
{1. κινήσου στο σημείο Α}
1.1. αριστερά (45)
1.2. μπροστά (5√2)

2. αριστερά (135)

{3. σχεδίασε ένα τετράγωνο με διάσταση πλευράς 10 εκατοστά}
3.1. σχεδίαση τετραγώνου (10)

{4. κινήσου στο σημείο Β}
4.1. δεξιά (45)
4.2. μπροστά (5√2)

5. αριστερά (135)

{6. σχεδίασε ένα τετράγωνο με διάσταση πλευράς 20 εκατοστά}
6.1. σχεδίαση τετραγώνου (20)

Βηματική ανάλυση του αλγορίθμου

τμήμα μπροστά (απόσταση)

{Κινεί μπροστά το ρομπότ κατά «απόσταση χ εκατοστά» αφήνοντας ίδιο τον προσανατολισμό του. Η μετακίνηση είναι ανεξάρτητη με το αν η πένα είναι σηκωμένη ή χαμηλωμένη}

1. Αριθμός απαιτούμενων περιστροφών N := χ / περίμετρος τροχού
2. for N φορές
   κάνε μια περιστροφή όλων των τροχών

Ορισμός και κλήση ενός τμήματος

- ορισμός ενός τμήματος :

module όνομα τμήματος (λίστα τυπικών παραμέτρων)
start
    . . .
end

- κλήση ενός τμήματος

{αλγόριθμος}
start
    . . .
    όνομα τμήματος (λίστα πραγματικών παραμέτρων)
    . . .
end

Τυπικές παράμετροι ενός τμήματος

- **τυπική παράμετρος εισόδου :** μια μεταβλητή εισόδου του τμήματος, η οποία θα πάρει αρχική τιμή από μια πραγματική παράμετρο εισόδου όταν το τμήμα κληθεί, ώστε με την τιμή αυτή να καθοδηγηθεί η λειτουργία του τμήματος
- **τυπική παράμετρος εξόδου :** μια μεταβλητή εξόδου του τμήματος, η οποία θα δώσει τελική τιμή σε μια πραγματική παράμετρο εξόδου αφού το τμήμα εκτελεστεί, ώστε με την τιμή αυτή να καθοδηγηθεί η λειτουργία του κύριου αλγορίθμου

- **λίστα τυπικών παραμέτρων :** μια λίστα N >= 1 τυπικών παραμέτρων FP_1, FP_2, …, FP_N, κάθε μία από τις οποίες μπορεί να είναι παράμετρος εισόδου ή/και εξόδου

Πραγματικές παράμετροι μιας κλήσης

- πραγματική παράμετρος εισόδου : μια μεταβλητή του κύριου αλγορίθμου της οποίας η τιμή δίνεται σε μια τυπική παράμετρο εισόδου πριν την εκτέλεση του τμήματος
- πραγματική παράμετρος εξόδου : μια μεταβλητή του κύριου αλγορίθμου της οποίας η τιμή λαμβάνεται από μια τυπική παράμετρο εξόδου μετά την εκτέλεση του τμήματος

- λίστα πραγματικών παραμέτρων : μια λίστα N >= 1 πραγματικών παραμέτρων RP_1, RP_2, …, RP_N, κάθε μία από τις οποίες μπορεί να είναι είτε σταθερή τιμή είτε παράμετρος εισόδου ή/και εξόδου

Εμφωλευμένες κλήσεις τμημάτων

- ένα τμήμα μπορεί να καλεί άλλα τμήματα
- οι εμφωλευμένες κλήσεις εκτελούνται ακριβώς όπως και οι απλές, με το τμήμα που καλεί να λειτουργεί ως κύριος αλγόριθμος
- ένα τμήμα μπορεί να καλεί τον εαυτό του ;

Η έννοια του τμήματος σε γλώσσες προγραμματισμού

- διαδικασίες (procedures)
- ρουτίνες (routines)
- υπορουτίνες (subroutines)
- συναρτήσεις (functions)
- . . .

- κλήσεις τιμής
- κλήσεις αναφοράς
- . . .

Γενικότητα χρήσης ενός τμήματος

- όσο γενικότερη είναι η χρησιμότητα της λειτουργίας ενός τμήματος, σε τόσο περισσότερες περιπτώσεις μπορεί να χρησιμοποιηθεί
- τετράγωνο : ειδική περίπτωση κανονικού πολυγώνου

τμήμα σχεδίαση κανονικού πολυγώνου (Δ, Ν)

{ Σχεδιάζει ένα κανονικό πολύγωνο με διάσταση πλευράς = Δ
εκφρασμένο σε εκατοστά και πλήθος πλευρών = Ν.
Το πολύγωνο σχεδιάζεται ξεκινώντας από την αρχική θέση και
προσανατολισμό της κεφαλής. Η κεφαλή επιστρέφει στην
αρχική θέση και προσανατολισμό, με την πένα πάνω. }

```
start
    πένα κάτω
    for μετρητής = 1 έως Ν
        μπροστά(Δ)
        αριστερά (360/Ν)
    πένα πάνω
end
```

Γενικότητα χρήσης ενός τμήματος (συνέχ.)

{σχεδίασε ένα τετράγωνο με διάσταση πλευράς 10 εκατ. }
σχεδίαση κανονικού πολυγώνου (10, 4)

{σχεδίασε ένα ισόπλευρο τρίγωνο με διάσταση πλευράς 6 εκατ. }
σχεδίαση κανονικού πολυγώνου (6,3)

Πλεονεκτήματα της στοιχειοποίησης

- μείωση της πολυπλοκότητας ⇒
  α) μείωση του μόχθου σχεδίασης
  β) μείωση της πιθανότητας σφαλμάτων

- αυτοτέλεια τμήματος σε σχέση με τον κύριο αλγόριθμο ⇒

  α) ανεξαρτησία στη σχεδίαση από τον κύριο αλγόριθμο :
  - σχεδίαση από άλλους ανθρώπους
  - σχεδίαση σε άλλο χρόνο (πριν, παράλληλα, μετά)

- β) ανεξαρτησία στην κατανόηση από τον κύριο αλγόριθμο

Πλεονεκτήματα της στοιχειοποίησης (συνέχ.)

- σαφής ορισμός της λειτουργίας, των εισόδων και των εξόδων ενός τμήματος ⇒

  α) απλότητα κλήσης του τμήματος από τον κύριο αλγόριθμο

- β) δυνατότητα κλήσης του τμήματος και από άλλους αλγορίθμους, δηλ. δυνατότητα επανα-χρησιμοποίησης

- βιβλιοθήκες επανα-χρησιμοποιήσιμων τμημάτων

- βιβλιοθήκες εξαρτημάτων λογισμικού

Αναδρομή

```
{υπολόγισε το παραγοντικό ενός φυσικού αριθμού N >=1}
start
    παραγοντικό := 1
    for μετρητής = 1 έως N
        παραγοντικό := παραγοντικό * μετρητής
    end
end
```

- γενικά ισχύει :
  α) 1! = 1
  β) N! = N * (N-1) * … * 1 = N * (N-1)!

Παράδειγμα αναδρομής (συνέχ.)

```
{υπολόγισε το παραγοντικό ενός φυσικού αριθμού N >=1 με
αναγωγή στο παραγοντικό του N-1}
start
    if N = 1
        then    παραγοντικό του N := 1
        else παραγοντικό του N := N * παραγοντικό του N-1
    end
end
```

Η έννοια της αναδρομής

- υπάρχουν συγκεκριμένα προβλήματα των οποίων η λύση ανάγεται επαναληπτικά στη λύση υπο-προβλημάτων

- όλες οι επαναλαμβανόμενες αναγωγές γίνονται με την ίδια λογική
- τα διαδοχικά υπο-προβλήματα γίνονται όλο και απλούστερα

- τελικά αναγόμαστε σε ένα πολύ απλό πρόβλημα με γνωστή εκ των προτέρων λύση
- από τη λύση του τελικού υπο-προβλήματος σχηματίζονται, επιστρέφοντας προς την αρχή, οι λύσεις όλων των ενδιάμεσων προβλημάτων, και εν τέλει και του αρχικού

Η έννοια της αναδρομής (συνέχ.)

- αναδρομή : η έκφραση της λύσης ενός προβλήματος ως λύση του ίδιου προβλήματος για απλούστερα δεδομένα

- αναδρομική επίλυση ενός προβλήματος : η επίλυση ενός προβλήματος σύμφωνα με
  α) έναν κανόνα παραγωγής διαδοχικών αναδρομών
  β) μία συνθήκη τερματισμού της αναδρομικής διαδικασίας
  γ) μία εξ αρχής γνωστή λύση του τελευταίου προβλήματος
  δ) έναν κανόνα παραγωγής των προηγούμενων λύσεων μέχρι και της αρχικής

Παράδειγμα Ι – υπολογισμός παραγοντικού

```
module υπολογισμός παραγοντικού (N, F)
{Υπολογίζει αναδρομικά το παραγοντικό ενός φυσικού αριθμού
N >=1. Το αποτέλεσμα επιστρέφεται στην παράμετρο F.}
start
    if N = 1
        then    F := 1
        else
                υπολογισμός παραγοντικού (N-1, tempF)
                F := N * tempF
end
```

Παράδειγμα Ι – υπολογισμός παραγοντικού

- υπολογισμός παραγοντικού (3, F)

if 3=1 then …
else
    υπολογισμός παραγοντικού (2, $tempF_{(1)}$)
    if 2=1 then …
    else
        υπολογισμός παραγοντικού (1, $tempF_{(2)}$)
        if 1=1 then
            $F_{(3)} := 1$ $\quad F_{(3)} = 1 \Rightarrow$

$tempF_{(2)} = F_{(3)} = 1 \Rightarrow$

$F_{(2)} := 2 * tempF_{(2)}$ $\quad F_{(2)} = 2*tempF_{(2)} = 2*1 = 2 \Rightarrow$

$tempF_{(1)} = F_{(2)} = 2 \Rightarrow$

$F_{(1)} := 3 * tempF_{(1)}$ $\quad F_{(1)} = 3*tempF_{(1)} = 3*2 = 6$

όπου $V_{(i)}$ είναι η τιμή της μεταβλητής V κατά την i-οστή αναδρομική κλήση

Παράδειγμα ΙΙ – αντιστροφή λέξης

- αντιστροφή των γραμμάτων μιας λέξης
  π.χ. “star” ➔ “rats”

{αναδρομική αντιστροφή μιας λέξης}
1. αφαίρεσε το πρώτο γράμμα
2. αντέστρεψε την υπόλοιπη λέξη
3. πρόσθεσε το πρώτο γράμμα

- η πλέον απλή περίπτωση : λέξη με 1 μόνο γράμμα
- στην περίπτωση αυτή : “x” ➔ “x”

Παράδειγμα ΙΙ – αντιστροφή λέξης

```
module αντιστροφή λέξης (W, R)
{Αντιστρέφει αναδρομικά μια λέξη W, επιστρέφοντας την
αντίστροφη λέξη στην παράμετρο R.}
start
    if η λέξη W περιέχει μόνο ένα γράμμα “x”
        then    R := “x”
        else
            start
                letter := το πρώτο γράμμα της λέξης W
                restW := το υπόλοιπο τμήμα της λέξης W
                αντιστροφή λέξης (restW, restR)
                R := η λέξη restR με πρόσθεση του letter στο τέλος
            end
τέλος
```

Παράδειγμα ΙΙ – αντιστροφή λέξης

- αντιστροφή λέξης (“star”, R)

if η λέξη “star” περιέχει μόνο ένα γράμμα τότε …
else
    $letter_{(1)}$ = “s”, $restW_{(1)}$ = “tar”
    αντιστροφή λέξης (“tar”, $restR_{(1)}$)
    if η λέξη “tar” περιέχει μόνο ένα γράμμα τότε …
    else
        $letter_{(2)}$ = “t”, $restW_{(2)}$ = “ar”
        αντιστροφή λέξης (“ar”, $restR_{(2)}$)
        if η λέξη “ar” περιέχει μόνο ένα γράμμα τότε …
        else
            $letter_{(3)}$ = “a”, $restW_{(3)}$ = “r”
            αντιστροφή λέξης (“r”, $restR_{(3)}$)
            if η λέξη “r” περιέχει μόνο ένα γράμμα τότε …
                $R_{(4)}$ = “r”
            $R_{(3)}$:= $restR_{(3)}$ με προσθήκη $letter_{(3)}$
        $R_{(2)}$:= $restR_{(2)}$ με προσθήκη $letter_{(2)}$
    $R_{(1)}$:= $restR_{(1)}$ με προσθήκη $letter_{(1)}$

$R_{(4)}$ = “r” ⇒ $restR_{(3)}$ = “r” ⇒
$R_{(3)}$ = “r” + “a” = “ra” ⇒ $restR_{(2)}$ = “ra” ⇒
$R_{(2)}$ = “ra” + “t” = “rat” ⇒ $restR_{(1)}$ = “rat” ⇒
$R_{(1)}$ = “rat” + “s” = “rats”

Η ανάγκη της αναδρομικότητας

- οι δύο προηγούμενοι αλγόριθμοι μπορούν να εκφραστούν και επαναληπτικά

- η αναδρομική έκφραση παρέχει
  α) ευκολία κατανόησης
  β) οικονομία αναπαράστασης

- σε ορισμένες περιπτώσεις η μη αναδρομική έκφραση, με δομές επανάληψης, είναι τόσο δύσκολο να σχεδιαστεί και να κατανοηθεί ώστε η αναδρομική έκφραση αποτελεί την μόνο πρακτική λύση

Παράδειγμα ΙΙΙ - ΜΚΔ

```
Module ΜΚΔ (α,β)
start
    if a > b then
                    M = a
                    N =b
            else
                    M = b
                    N =a
    x = N
    while (x != 0) do
            M =N
            N =x
            x=M%N
    return N
end
```

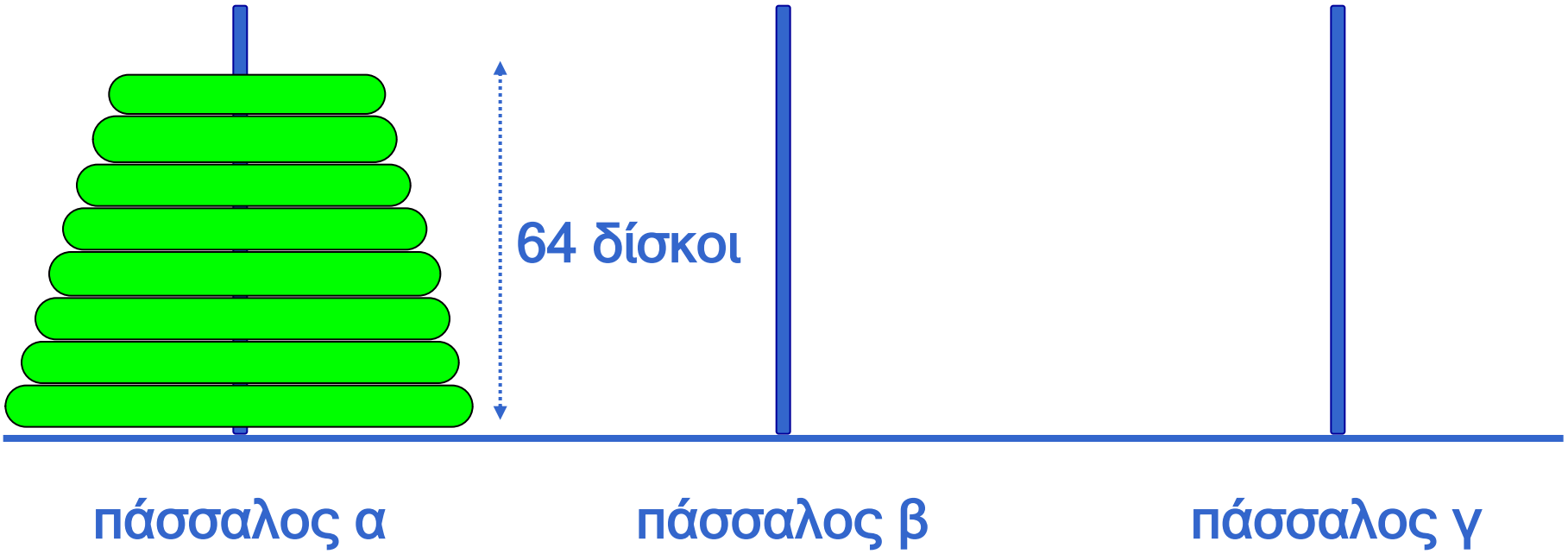
Παράδειγμα ΙΙΙ - ΜΚΔ

```
module MKD (x,y)
start
if x > y
        then
                M = x
                N = y
        else
                M = y
                N = x
if N = 0
        then return M
        else MKD (y, x%y)
end
```

Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι

- δοκιμασία μιας τάξης βουδιστών μοναχών
- απαιτεί εξαιρετική αυτοσυγκέντρωση επί πολύ μεγάλο χρονικό διάστημα
- το πρόβλημα : να μετακινηθούν 64 κυκλικοί δίσκοι διαδοχικού μεγέθους από έναν πάσσαλο σε έναν άλλο, κατά τρόπο ώστε ποτέ ένας δίσκος να μην τοποθετηθεί πάνω σε μικρότερο
- υπάρχει ένας ακόμη πάσσαλος ο οποίος μπορεί να χρησιμοποιηθεί βοηθητικά

Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι



Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι

- έστω α, β, γ οι τρεις πάσσαλοι (γ είναι ο βοηθητικός)
- έστω Δ_1, Δ_2, … οι δίσκοι (από το μικρότερο στο μεγαλύτερο)

- αν είχαμε 1 δίσκο, τότε
  1η κίνηση : Δ_1 α ➔ β (OK)
- αν είχαμε 2 δίσκους, τότε
  1η κίνηση : Δ_1 α ➔ γ
  2η κίνηση : Δ_2 α ➔ β
  3η κίνηση : Δ_1 γ ➔ β (OK)

- αν είχαμε 3 δίσκους ;
- αν είχαμε Ν δίσκους ;

Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι

- έχουμε γνωστή λύση για μια απλή μορφή (1 δίσκος)
- η λύση της αμέσως λιγότερο απλής μορφής (2 δίσκοι) μπορεί να εκφραστεί με αναγωγή στη λύση της προηγούμενης

```
{μετέφερε 2 δίσκους από τον α στο β}
start
    μετέφερε 1 δίσκο από τον α στον γ
    μετέφερε 1 δίσκο από τον α στον β
    μετέφερε 1 δίσκο από τον γ στον β
end
```

- η έκφραση αυτή μπορεί να γενικευτεί ώστε να λύνει το πρόβλημα για Ν δίσκους, όπου Ν >= 2

Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι

```
module μεταφορά πύργου (N, source, target, help)
{Μεταφέρει έναν πύργο Ν δίσκων από τον πάσσαλο source στον
πάσσαλο  target, χρησιμοποιώντας τον πάσσαλο help ως βοηθητικό.
Ο πάσσαλος help πρέπει αρχικά να είναι κενός και μετά τη
μεταφορά βρίσκεται και πάλι κενός.}
    if N = 1
    then
        μετέφερε 1 δίσκο από τον source στον target
    else
        μεταφορά πύργου (N-1, source, help, target)
        μετέφερε 1 δίσκο από τον source στον target
        μεταφορά πύργου (N-1, help, target, source)
    end
end
```

Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι

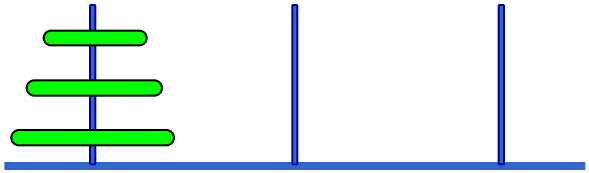
- χρησιμοποιούμε το N=1 ως βασική περίπτωση

- σε κάθε αναδρομική κλήση, οι τρεις πάσσαλοι α, β, γ χρησιμοποιούνται εκ περιτροπής ως αρχικοί, τελικοί και βοηθητικοί
- ο πάσσαλος που χρησιμοποιείται ως βοηθητικός διατηρείται πάντοτε κενός πριν και μετά τη χρήση

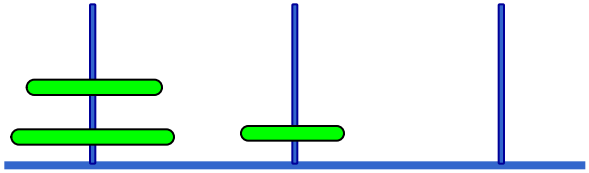
Παράδειγμα III – οι πύργοι του Ανόι

- μεταφορά πύργου (3, source, target, help)

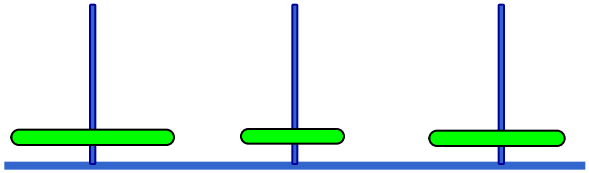
(1)



(2)

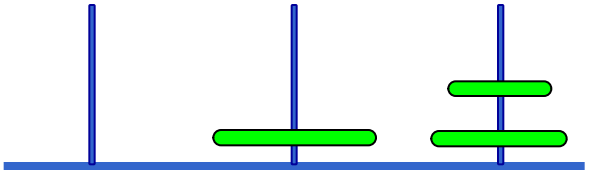


(3)

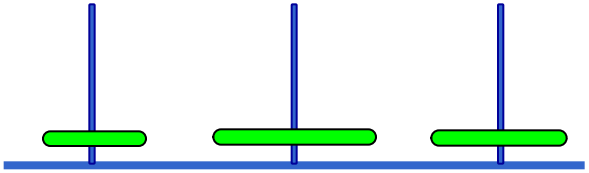


(4)

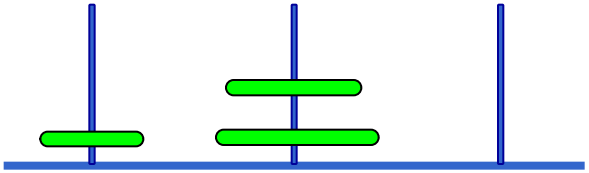
(5)



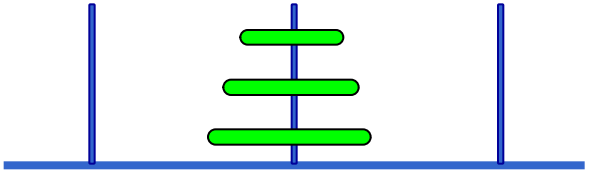
(6)



(7)



(8)



Παράδειγμα ΙΙΙ – οι πύργοι του Ανόι

- κάθε αναδρομική κλήση οδηγεί σε 2 καινούριες
- η εκτέλεση για N=64 θα οδηγήσει σε 2^{64}-1 δηλ. περίπου $32*10^{18}$ αναδρομικές κλήσεις του αλγορίθμου
- αν μπορούσαμε με το χέρι να κάνουμε 1 σωστή κίνηση/sec θα χρειάζονταν περίπου 600 δις έτη για να μετακινήσουμε τους 64 δίσκους

- υπάρχουν κάποιες επαναληπτικές εκφράσεις του συγκεκριμένου αλγορίθμου, οι οποίες όμως είναι εξαιρετικά δυσνόητες
- έχει ήδη αποδειχθεί ότι καμία άλλη έκφραση, επαναληπτική ή αναδρομική, δεν είναι γρηγορότερη

Σχεδίαση ενός αναδρομικού αλγορίθμου

- εντοπισμός της βασικής περίπτωσης και της γνωστής λύσης

- σχεδίαση της συνθήκης αναδρομής : οι είσοδοι δεν εμπίπτουν στη βασική περίπτωση (ή ισοδύναμα)
- σχεδίαση της συνθήκης διαφυγής : οι είσοδοι εμπίπτουν στη βασική περίπτωση

- σχεδίαση του σώματος αναδρομής
- σχεδίαση του μονοπατιού διαφυγής

- έλεγχος τερματισμού : είναι δυνατή, σε χρόνο εκτέλεσης, η διάψευση της συνθήκης αναδρομής (ή ισοδύναμα) η επαλήθευση της συνθήκης διαφυγής ;