Δημοκρίτειο Πανεπιστήμιο Θράκης

Τμήμα Δασολογίας και Διαχείρισης Περιβάλλοντος και Φυσικών Πόρων

Εργαστήριο Δασικής-Περιβαλλοντικής Πληροφορικής και Υπολογιστικής Νοημοσύνης

«Τεχνολογίες Πληροφορικής και Διαδικτύου»

Δρ Κωνσταντίνος Δεμερτζής

Βασικές Έννοιες Αλγορίθμων

Αλγόριθμος είναι μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.

- **Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια:**
  - **Είσοδος (input).** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.

- **Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια:**
  - **Είσοδος (input).** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
  - **Έξοδος (output).** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.

- **Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια:**
  - **Είσοδος (input).** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
  - **Έξοδος (output).** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
  - **Καθοριστικότητα (definiteness).** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης λαμβάνει μηδενική τιμή.

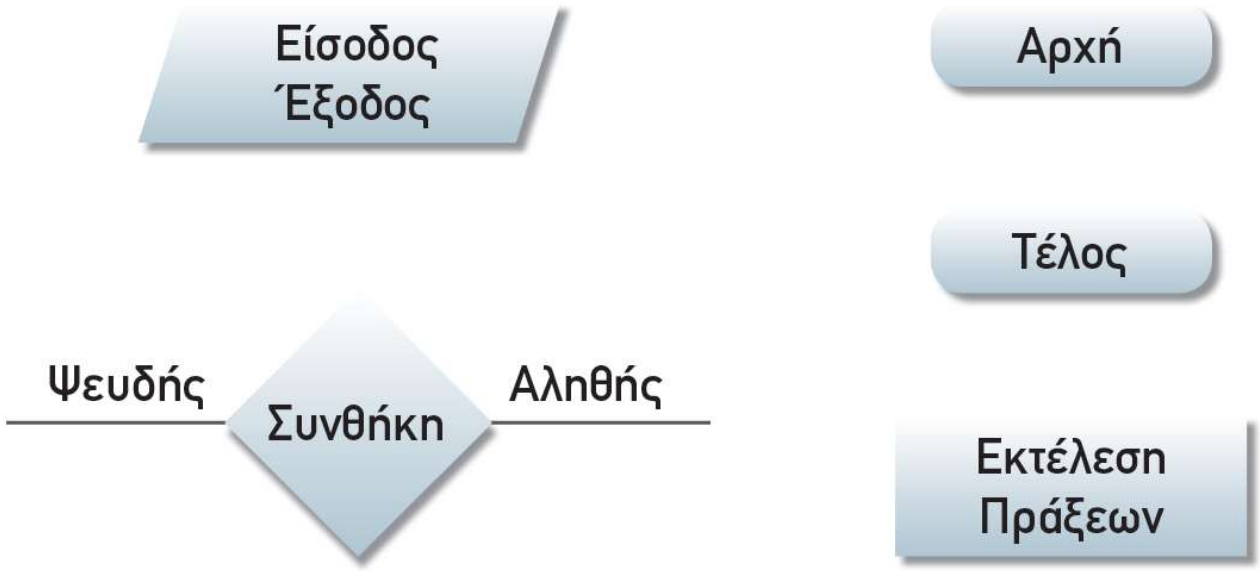
- **Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια:**
  - **Είσοδος (input).** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
  - **Έξοδος (output).** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
  - **Καθοριστικότητα (definiteness).** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης λαμβάνει μηδενική τιμή.
  - **Περατότητα (finiteness).** Ο αλγόριθμος να τελειώνει μετά από πε-περασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία.

- **Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια:**
  - **Είσοδος (input).** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
  - **Έξοδος (output).** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
  - **Καθοριστικότητα (definiteness).** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης λαμβάνει μηδενική τιμή.
  - **Περατότητα (finiteness).** Ο αλγόριθμος να τελειώνει μετά από πε-περασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία.
  - **Αποτελεσματικότητα (effectiveness).** Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

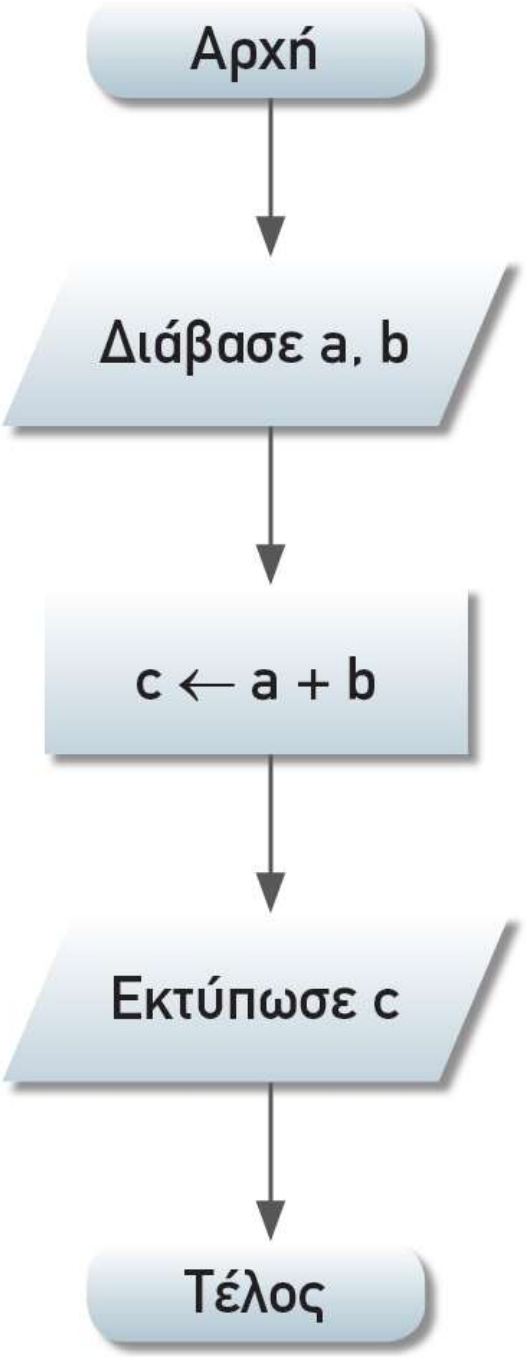

Η λάμπα δεν δουλεύει
Είναι στη πρίζα;
Όχι
Βάλε στη πρίζα
Ναι
Καμένος λαμπτήρας;
Ναι
Αντικατάστησε λαμπτήρα
Όχι
Επισκεύασε τη λάμπα

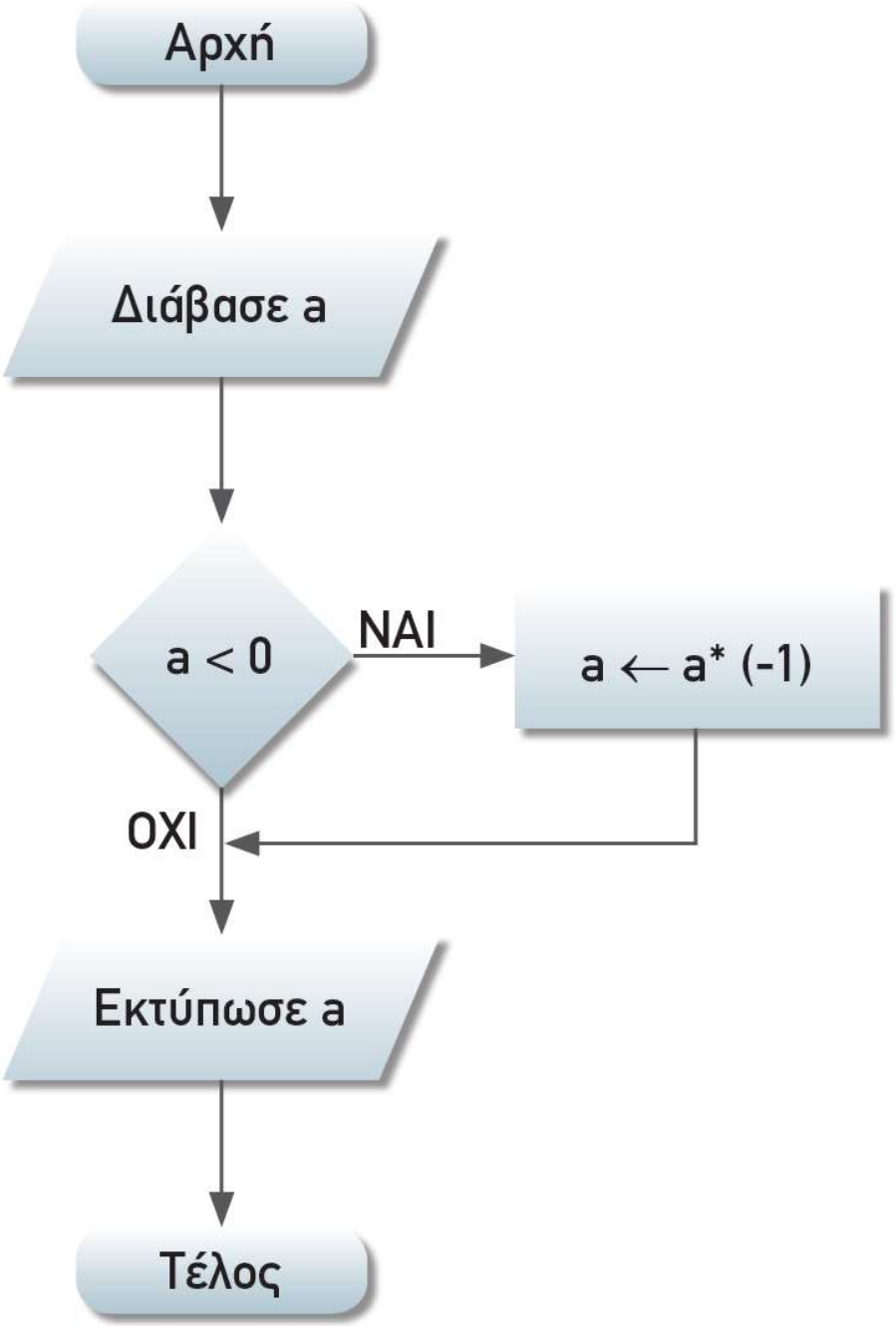
- **Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής.**
- **Η Πληροφορική μπορεί να ορισθεί ως η επιστήμη που μελετά τους αλγορίθμους από τις ακόλουθες σκοπιές:**
  - Υλικού (hardware). Η ταχύτητα εκτέλεσης ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού, δηλαδή από τον τρόπο που είναι δομημένα σε μία ενιαία αρχιτεκτονική τα διάφορα συστατικά του ΗΥ.
  - Γλωσσών Προγραμματισμού (programming languages). Το είδος της γλώσσας που χρησιμοποιείται (χαμηλότερου ή υψηλότερου επιπέδου) αλλάζει τη δομή και τον αριθμό των εντολών ενός αλγορίθμου.
  - Θεωρητική (theoretical). Το ερώτημα που συχνά τίθεται είναι αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος. Η εξέταση αυτού του ερωτήματος είναι να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του.
  - Αναλυτική (analytical). Μελετώνται οι υπολογιστικοί πόροι που απαιτούνται από έναν αλγόριθμο, όπως για παράδειγμα το μέγεθος της κύριας και της δευτερεύουσας μνήμης, ο χρόνος για λειτουργίες CPU και για λειτουργίες εισόδου/εξόδου κ.λπ.

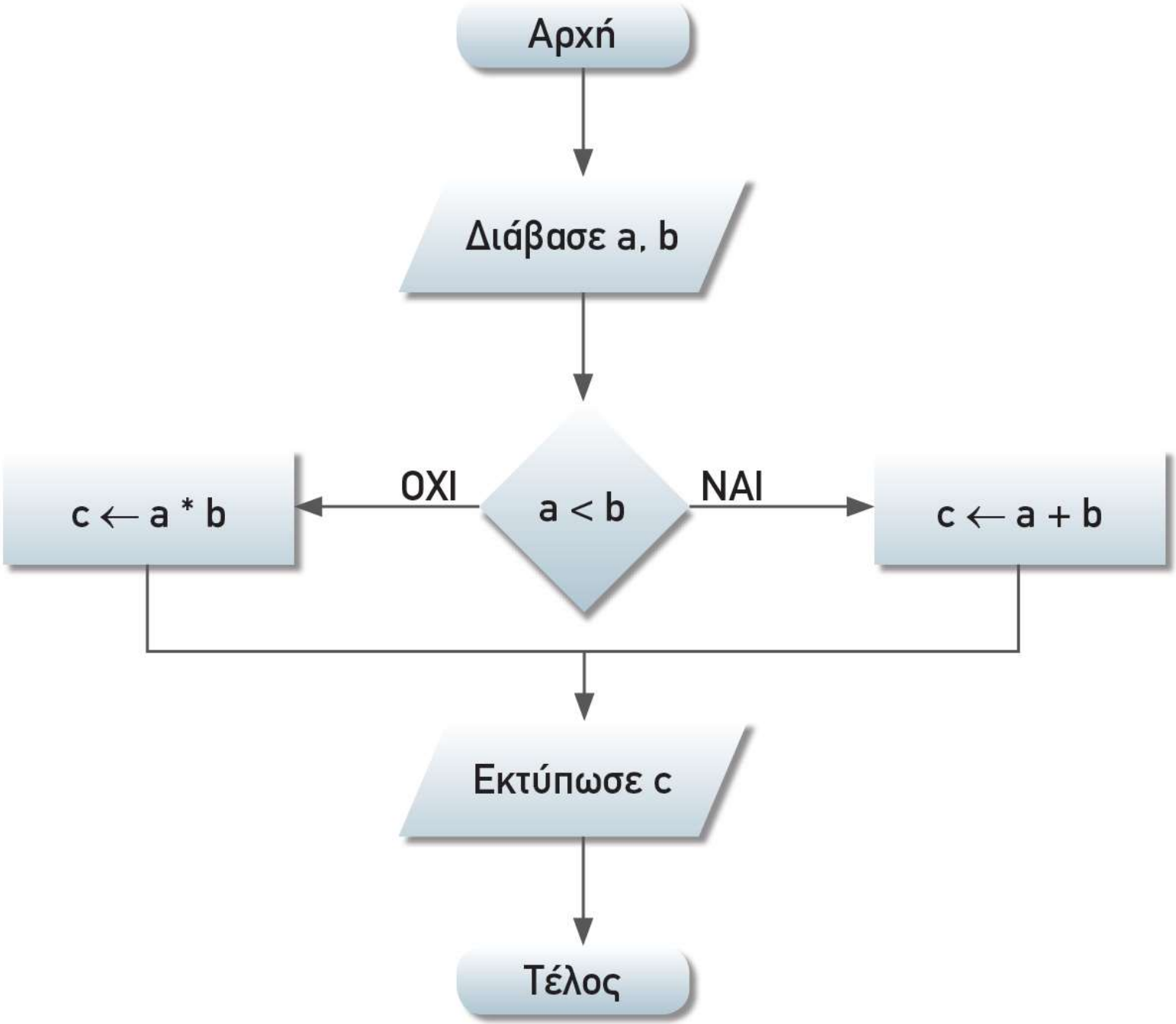
- **Στη βιβλιογραφία συναντώνται διάφοροι τρόποι αναπαράστασης ενός αλγορίθμου:**
  - **με ελεύθερο κείμενο (free text),** που αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Έτσι εγκυμονεί τον κίνδυνο ότι μπορεί εύκολα να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα.
  - **με διαγραμματικές τεχνικές (diagramming techniques),** που συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγορίθμου. Από τις διάφορες διαγραμματικές τεχνικές που έχουν επινοηθεί, η πιο παλιά και η πιο γνωστή ίσως, είναι το διάγραμμα ροής (flow chart). Ωστόσο η χρήση διαγραμμάτων ροής για την παρουσίαση αλγορίθμων δεν αποτελεί την καλύτερη λύση, γι' αυτό και εμφανίζονται όλο και σπανιότερα στην πράξη.
  - **με φυσική γλώσσα (natural language) κατά βήματα.** Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, όπως προσδιορίσθηκε προηγουμένως, δηλαδή το κριτήριο του καθορισμού.
  - **με κωδικοποίηση (coding),** δηλαδή με ένα πρόγραμμα γραμμένο είτε σε μία ψευδογλώσσα είτε σε κάποια γλώσσα προγραμματισμού που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

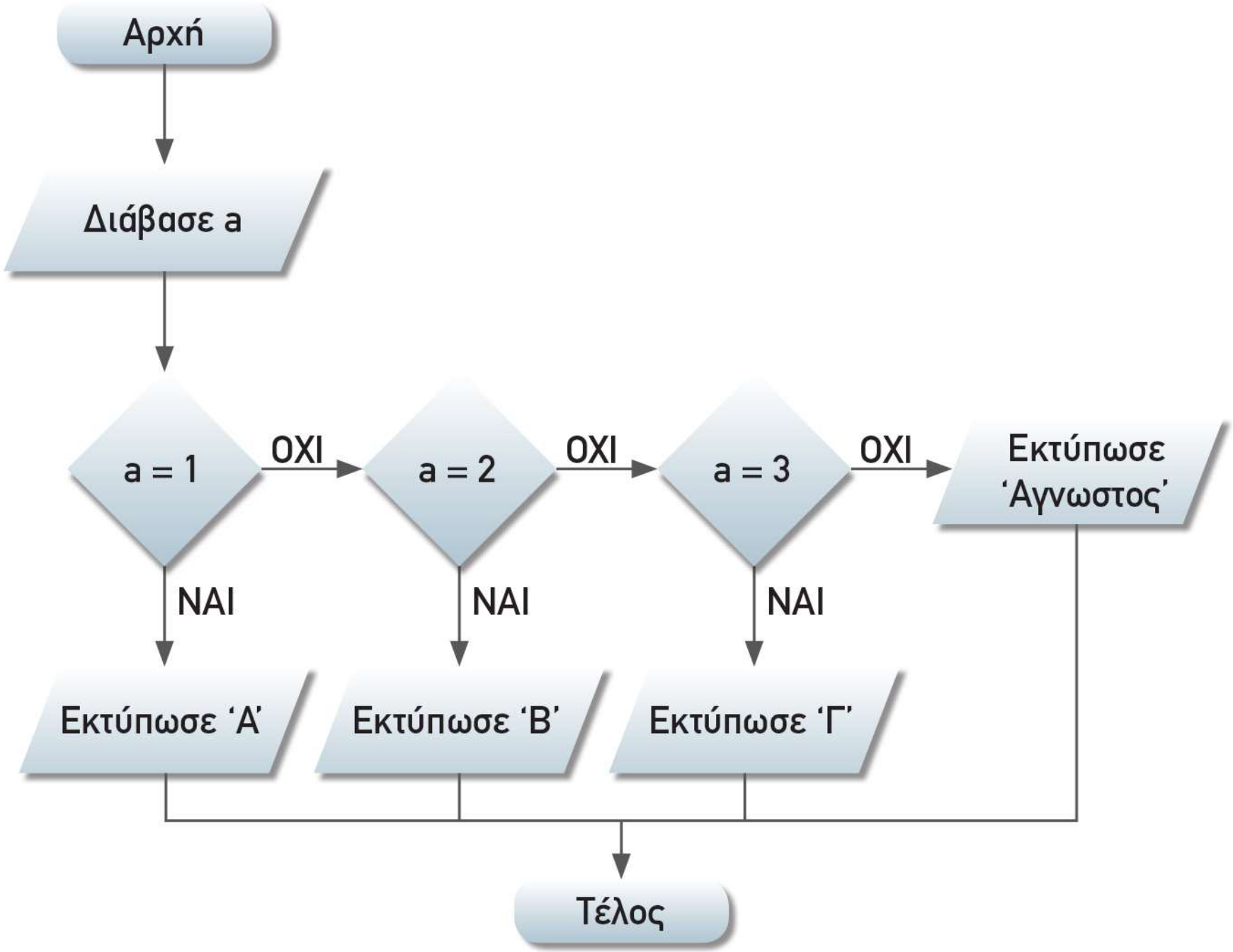


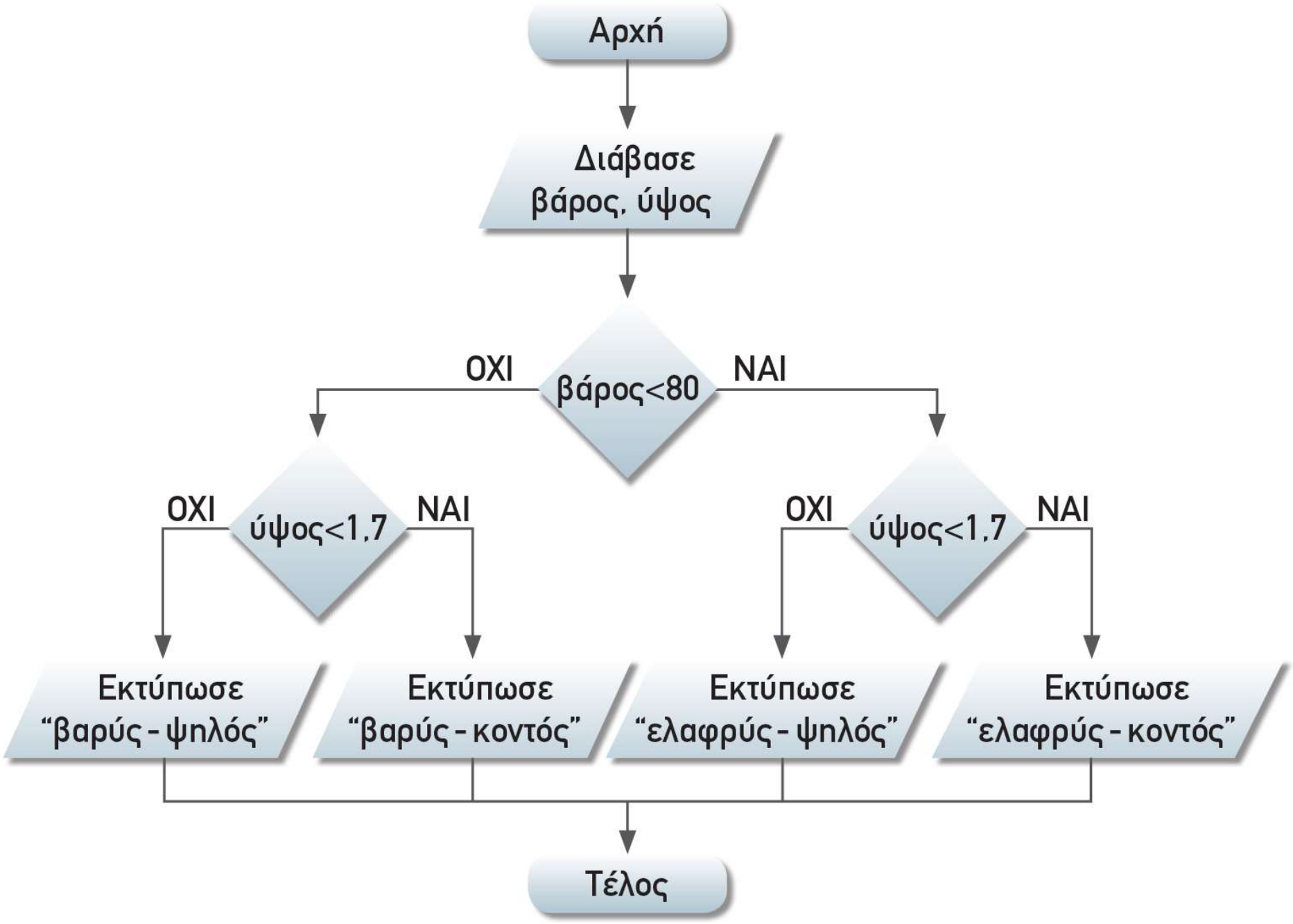
Μερικά από τα χρησιμοποιούμενα γεωμετρικά σύμβολα στα διαγράμματα ροής.


Αρχή
Διάβασε a, b
c ← a + b
Εκτύπωσε c
Τέλος


Αρχή
Διάβασε a
a < 0
ΝΑΙ
a ← a* (-1)
ΟΧΙ
Εκτύπωσε a
Τέλος


Αρχή
Διάβασε a, b
a < b
ΟΧΙ
ΝΑΙ
c ← a * b
c ← a + b
Εκτύπωσε c
Τέλος


Αρχή
Διάβασε a
a = 1
ΟΧΙ
a = 2
ΟΧΙ
a = 3
ΟΧΙ
Εκτύπωσε 'Αγνωστος'
ΝΑΙ
ΝΑΙ
ΝΑΙ
Εκτύπωσε 'Α'
Εκτύπωσε 'Β'
Εκτύπωσε 'Γ'
Τέλος

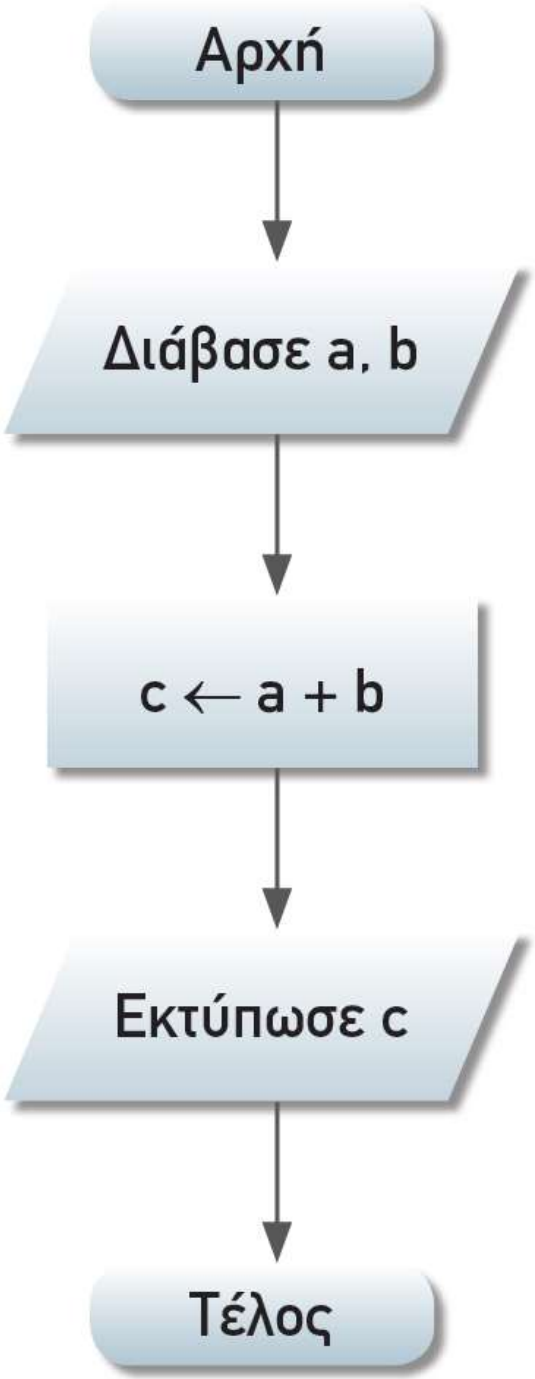

Αρχή
Διάβασε
βάρος, ύψος
βάρος<80
ΟΧΙ
ΝΑΙ
ύψος<1,7
ΟΧΙ
ΝΑΙ
ύψος<1,7
ΟΧΙ
ΝΑΙ
Εκτύπωσε
“βαρύς - ψηλός”
Εκτύπωσε
“βαρύς - κοντός”
Εκτύπωσε
“ελαφρύς - ψηλός”
Εκτύπωσε
“ελαφρύς - κοντός”
Τέλος

ΠΑΡΑΔΕΙΓΜΑ 1. Ανάγνωση και εκτύπωση αριθμών

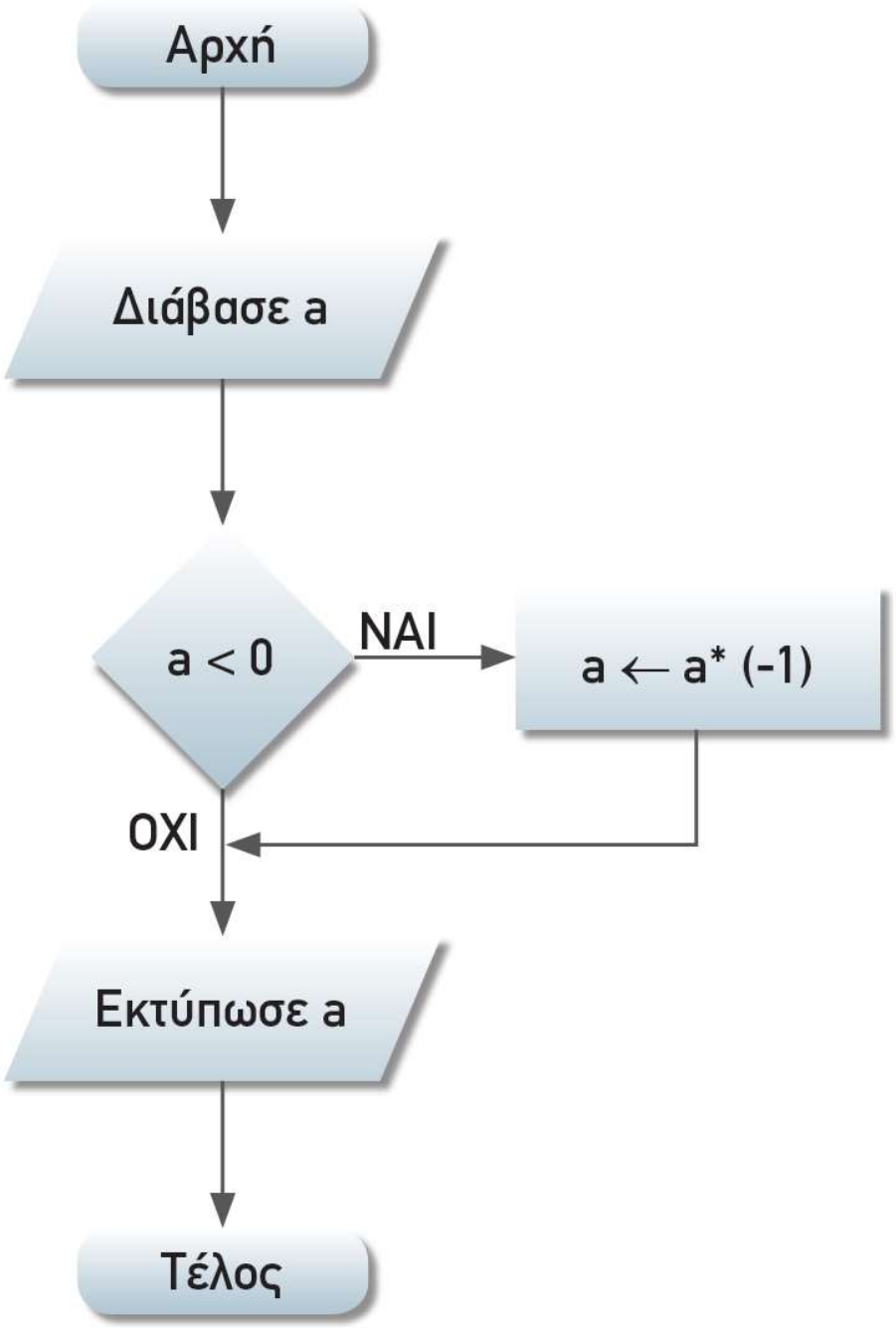
Να διαβασθούν δύο αριθμοί, να υπολογισθεί και να εκτυπωθεί το άθροισμά τους.

Από την εκφώνηση προκύπτει αμέσως ο επόμενος αλγόριθμος

```
Αλγόριθμος Παράδειγμα_1
Διάβασε a
Διάβασε b
c ← a + b
Εκτύπωσε c
Τέλος Παράδειγμα_1
```


Αρχή
Διάβασε a, b
c ← a + b
Εκτύπωσε c
Τέλος

```
Αλγόριθμος Παράδειγμα_2
Διάβασε a
Αν a < 0 τότε a ← a*(-1)
Εκτύπωσε a
Τέλος Παράδειγμα_2
```


Αρχή
Διάβασε a
a < 0
ΝΑΙ
a ← a* (-1)
ΟΧΙ
Εκτύπωσε a
Τέλος

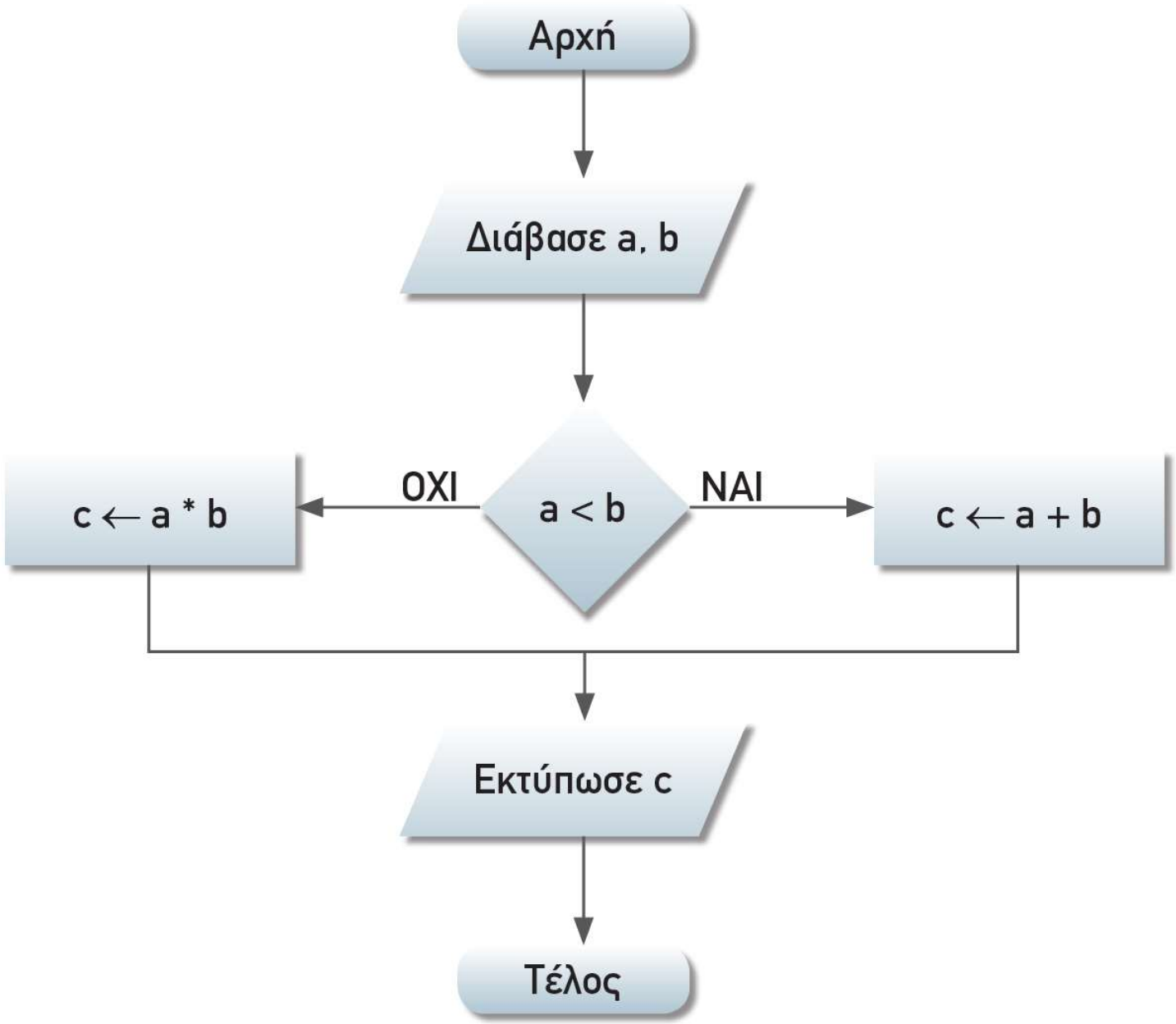
ΠΑΡΑΔΕΙΓΜΑ 3. Σύγκριση αριθμών με σύνθετη επιλογή

Να διαβασθούν δύο αριθμοί και σε περίπτωση που ο πρώτος αριθμός είναι μικρότερος του δεύτερου, να υπολογισθεί και να εκτυπωθεί το άθροισμά τους, διαφορετικά να υπολογισθεί και να εκτυπωθεί το γινόμενό τους.

```
Αλγόριθμος Παράδειγμα_3
Διάβασε a, b
Αν a < b τότε
   c ← a + b
αλλιώς
   c ← a * b
Τέλος_αν
Εκτύπωσε c
Τέλος Παράδειγμα_3
```

Στο παράδειγμα αυτό χρησιμοποιείται η γενική μορφή της εντολής επιλογής, που είναι:

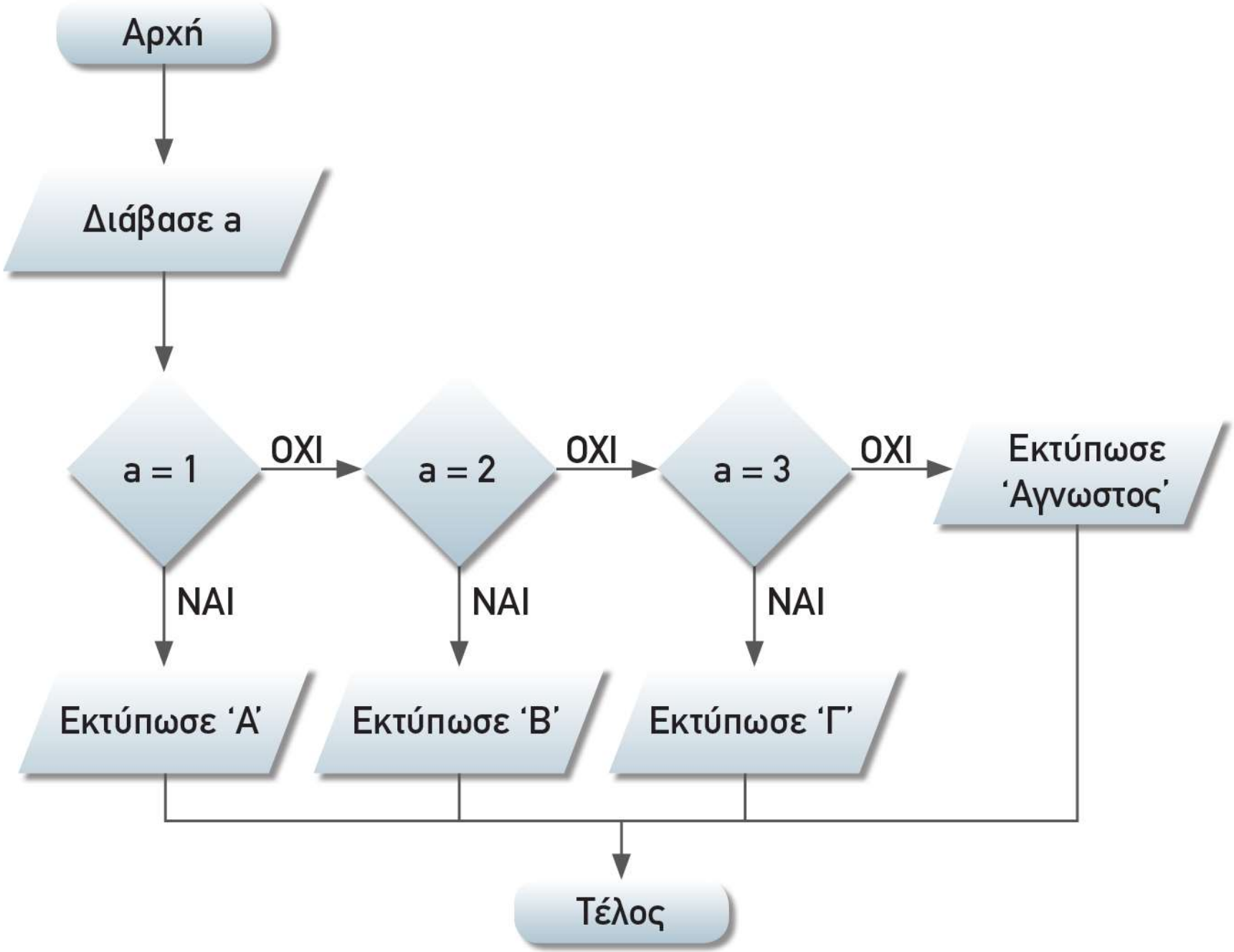
```
Αν συνθήκη τότε
   εντολή ή εντολές
αλλιώς
   εντολή ή εντολές
Τέλος_αν
```


Αρχή
Διάβασε a, b
a < b
ΟΧΙ
ΝΑΙ
c ← a * b
c ← a + b
Εκτύπωσε c
Τέλος

ΠΑΡΑΔΕΙΓΜΑ 4. Ανάθεση γραμμάτων σε αριθμούς

Να διαβασθεί ένας ακέραιος και να εκτυπωθεί το αντίστοιχο γράμμα της αλφαβήτου αν ο ακέραιος έχει τιμή 1 ή 2 ή 3, διαφορετικά να εκτυπωθεί η λέξη "άγνωστος".

```
Αλγόριθμος Παράδειγμα_4
Διάβασε a
Αν a = 1 τότε εκτύπωσε "Α"
αλλιώς_αν a = 2 τότε εκτύπωσε "Β"
αλλιώς_αν a = 3 τότε εκτύπωσε "Γ"
αλλιώς εκτύπωσε "άγνωστος"
Τέλος_αν
Τέλος Παράδειγμα_4
```


Αρχή
Διάβασε a
a = 1
ΟΧΙ
a = 2
ΟΧΙ
a = 3
ΟΧΙ
Εκτύπωσε 'Αγνωστος'
ΝΑΙ
ΝΑΙ
ΝΑΙ
Εκτύπωσε 'Α'
Εκτύπωσε 'Β'
Εκτύπωσε 'Γ'
Τέλος

ΠΑΡΑΔΕΙΓΜΑ 5. Επιλογή ορίων

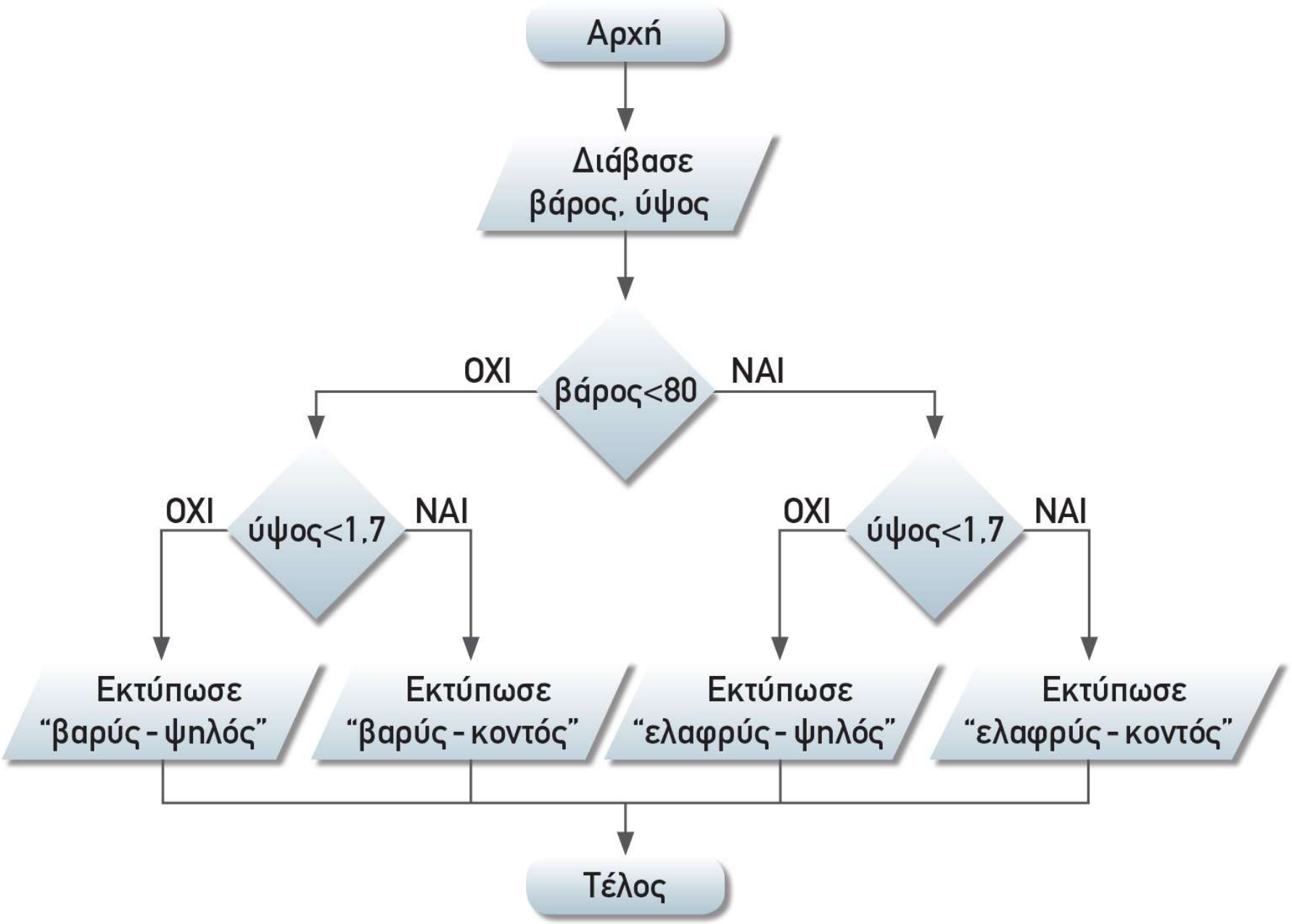
Να εισαχθεί ένας ακέραιος που αντιστοιχεί σε μια ηλικία και να βρεθεί σε ποια όρια εντάσσεται η δεδομένη ηλικία εμφανίζοντας σχετικό μήνυμα.

```
Αλγόριθμος Παράδειγμα 5.
Εμφάνισε "Σε ποια ηλικία άρχισες να μαθαίνεις
προγραμματισμό;"
Διάβασε age
Επίλεξε age
   Περίπτωση < 0
      Εμφάνισε "Είπαμε ηλικία ..."
   Περίπτωση < 5
      Εμφάνισε "Μάλλον τα παραλές !!"
   Περίπτωση < 60
      Εμφάνισε "Μπράβο"
   Περίπτωση < 100
      Εμφάνισε "Ποτέ δεν είναι αργά"
   Περίπτωση αλλιώς
      Εμφάνισε "Κάλλιο αργά παρά ποτέ"
Τέλος_επιλογών
Τέλος Παράδειγμα_5
```

ΠΑΡΑΔΕΙΓΜΑ 6. Χαρακτηρισμός ατόμων

Να διαβάζονται δύο αριθμοί που αντιστοιχούν στο ύψος και βάρος ενός άνδρα. Να εκτυπώνεται ότι ο άνδρας είναι "ελαφρύς", αν το βάρος του είναι κάτω από 80 κιλά, ή να εκτυπώνεται "βαρύς" στην αντίθετη περίπτωση. Επίσης να εκτυπώνεται "κοντός" αν το ύψος του είναι κάτω από 1.70, αλλιώς να εκτυπώνεται "ψηλός".

```
Αλγόριθμος Παράδειγμα_6
Διάβασε βάρος, ύψος
Αν βάρος < 80 τότε
   Αν ύψος < 1.70 τότε
      εκτύπωσε "ελαφρύς-κοντός"
   αλλιώς
      εκτύπωσε "ελαφρύς-ψηλός"
   Τέλος_αν
αλλιώς
   Αν ύψος < 1.70 τότε
      εκτύπωσε "βαρύς-κοντός"
   αλλιώς
      εκτύπωσε "βαρύς-ψηλός"
   Τέλος_αν
Τέλος_αν
Τέλος Παράδειγμα_6
```


Αρχή
Διάβασε
βάρος, ύψος
βάρος<80
ΟΧΙ
ΝΑΙ
ύψος<1,7
ΟΧΙ
ΝΑΙ
ύψος<1,7
ΟΧΙ
ΝΑΙ
Εκτύπωσε
“βαρύς - ψηλός”
Εκτύπωσε
“βαρύς - κοντός”
Εκτύπωσε
“ελαφρύς - ψηλός”
Εκτύπωσε
“ελαφρύς - κοντός”
Τέλος

```
Αλγόριθμος Παράδειγμα_7
i ← 1
Όσο i ≤ 100 επανάλαβε
    Εμφάνισε i
    i ← i + 1
Τέλος_επανάληψης
Τέλος Παράδειγμα_7
```

```
Αλγόριθμος Παράδειγμα_8
Διάβασε x
Όσο x > 0 επανάλαβε
    Εμφάνισε x
    Διάβασε x
Τέλος_επανάληψης
Τέλος Παράδειγμα_8
```

ΠΑΡΑΔΕΙΓΜΑ 9. Εκτύπωση θετικών αριθμών με εντολή: αρχή_επανάληψης...μέχρις_ότου

Να διαβάζονται και να εκτυπώνονται όσοι θετικοί αριθμοί δίνονται από το πληκτρολόγιο. Ο αλγόριθμος τελειώνει, όταν δοθεί ένας αρνητικός αριθμός.

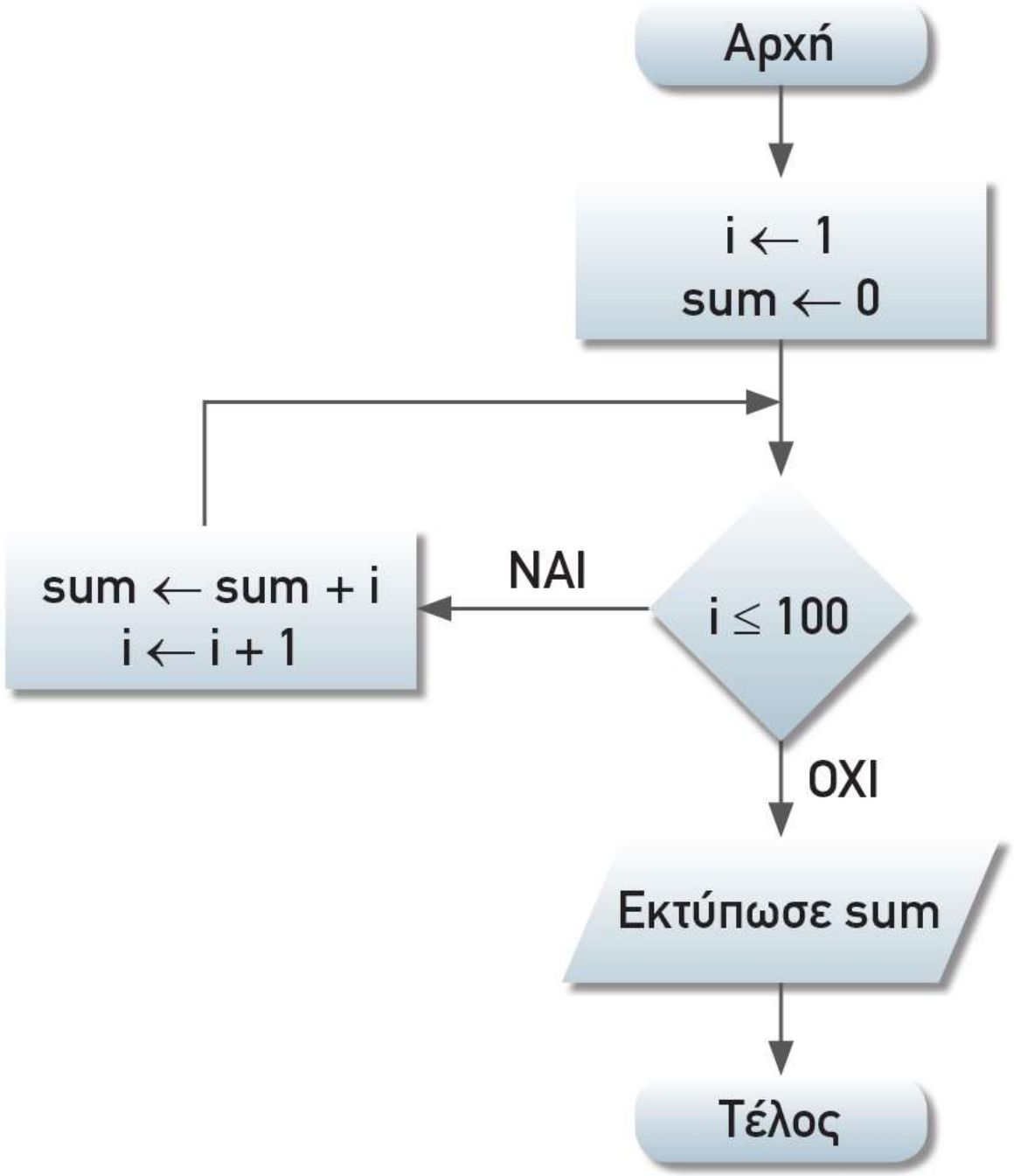
```
Αλγόριθμος Παράδειγμα_9
Αρχή_επανάληψης
    Διάβασε x
    Εμφάνισε x
Μέχρις_ότου x < 0
Τέλος Παράδειγμα_9
```

ΠΑΡΑΔΕΙΓΜΑ 10. Υπολογισμός αθροίσματος αριθμών με επαναληπτική εντολή: για...από...μέχρι

Να βρεθεί και να εκτυπωθεί το άθροισμα των 100 ακεραίων από το 1 μέχρι το 100.

Όταν ο αριθμός των φορών που θα εκτελεστεί μια επαναληπτική διαδικασία είναι γνωστός εκ των προτέρων, τότε είναι προτιμότερο να χρησιμοποιείται η εντολή Για...από...μέχρι. Έτσι ο ζητούμενος αλγόριθμος είναι.

```
Αλγόριθμος Παράδειγμα_10
Sum ← 0
Για i από 1 μέχρι 100
    Sum ← Sum + i
Τέλος_επανάληψης
Εκτύπωσε Sum
Τέλος Παράδειγμα_10
```


Αρχή
i ← 1
sum ← 0
i ≤ 100
ΝΑΙ
sum ← sum + i
i ← i + 1
ΟΧΙ
Εκτύπωσε sum
Τέλος

ΠΑΡΑΔΕΙΓΜΑ 11. Υπολογισμός αθροίσματος με επαναληπτική εντολή: για...από...μέχρι...με _βήμα

Να βρεθεί και να εκτυπωθεί το άθροισμα των άρτιων αριθμών από το 1 μέχρι το 100.

Η λύση αυτού του προβλήματος είναι παρόμοια με αυτή του προηγούμενου. Η μόνη αλλαγή είναι στην εντολή επανάληψης όπου προσδιορίζεται η ποσότητα *βήμα*, η οποία κάθε φορά προστίθεται στην τιμή της μεταβλητής i. Έτσι έχουμε

```
Αλγόριθμος Παράδειγμα_11
άθροισμα ← 0
Για i από 2 μέχρι 100 με_βήμα 2
    άθροισμα ← άθροισμα + i
Τέλος_επανάληψης
Εκτύπωσε άθροισμα
Τέλος Παράδειγμα_11
```

$$
\begin{array}{r}
45 \\
\times \quad 19 \\
\hline
405 \\
+ \ 45 \\
\hline
855
\end{array}
$$

Ας θεωρήσουμε την πράξη του πολλαπλασιασμού δύο ακεραίων αριθμών και ας θυμηθούμε πώς αυτή υλοποιείται χειρωνακτικά. Τοποθετούμε, λοιπόν, τους δύο αριθμούς τον ένα κάτω από τον άλλο και πολλαπλασιάζουμε κάθε ψηφίο του κάτω αριθμού με όλα τα ψηφία του επάνω αριθμού. Πιο συγκεκριμένα, για κάθε ψηφίο του κάτω αριθμού παράγεται ένα μερικό γινόμενο, ενώ τα μερικά γινόμενα τοποθετούνται το ένα κάτω από το άλλο με μία μετατόπιση από τα δεξιά προς τα αριστερά καθώς θεωρούμε διαδοχικά τα ψηφία των μονάδων, των δεκάδων, των εκατοντάδων κ.λπ. Στη συνέχεια γίνεται η πρόσθεση των επιμέρους γινομένων, αφού τα τοποθετήσουμε στην κατάλληλη διά-ταξη όπως φαίνεται στο σχήμα

45	19	45
90	9	90
180	4	
360	2	
720	1	720
Άθροισμα = 855		

Έστω, λοιπόν, ότι δίνονται δύο θετικοί ακέραιοι αριθμοί, οι αριθμοί 45 και 19. Οι αριθμοί γράφονται δίπλα-δίπλα και ο πρώτος διπλασιάζεται, ενώ ο δεύτερος υποδιπλασιάζεται αγνοώντας το δεκαδικό μέρος. Στο σχήμα παρουσιάζεται η επαναλαμβανόμενη διαδικασία, που συνεχίζεται μέχρις ότου στη δεύτερη στήλη να προκύψει μονάδα. Τελικώς, το γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης, όπου αντίστοιχα στη δεύτερη στήλη υπάρχει περιττός αριθμός. Για το παράδειγμά μας, τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη.

Πολλαπλασιασμός αλά ρωσικά

45	19	45
90	9	90
180	4	
360	2	
720	1	720
Άθροισμα = 855		

Πολλαπλασιασμός αλά ρωσικά

Αλγόριθμος: Πολλαπλασιασμός δύο θετικών ακεραίων (αλά ρωσικά)	
Είσοδος:	Δύο ακέραιοι M1 και M2, όπου M1, M2 ≥ 1
Έξοδος:	Το γινόμενο P=M1*M2
Βήμα 1	Θέσε P=0
Βήμα 2	Αν M2>0, τότε πήγαινε στο Βήμα 3, αλλιώς πήγαινε στο Βήμα 7
Βήμα 3	Αν ο M2 είναι περιττός, τότε θέσε P=P+M1
Βήμα 4	Θέσε M1=M1*2
Βήμα 5	Θέσε M2=M2/2 (θεώρησε μόνο το ακέραιο μέρος)
Βήμα 6	Πήγαινε στο Βήμα 2
Βήμα 7	Τύπωσε τον P

Πολλαπλασιασμός αλά ρωσικά

```
Αλγόριθμος Πολλαπλασιασμός_αλά_ρωσικά
Δεδομένα // Μ1,Μ2 ακέραιοι //
Ρ ← 0
Όσο Μ2 > 0 επανάλαβε
    Αν Μ2 mod 2 = 1 τότε Ρ ← Ρ+Μ1
     Μ1 ← Μ1*2
     Μ2 ← Μ2 div 2
Τέλος_επανάληψης
Αποτελέσματα // Ρ, το γινόμενο των ακεραίων Μ1,Μ2 //
Τέλος Πολλαπλασιασμός_αλά_ρωσικά
```

ΟΛΙΣΘΗΣΗ (SHIFT)

Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή, δηλαδή 0 και 1, ανεξάρτητα από το πώς τα ορίζει ο προγραμματιστής, όπως ακεραίους ή πραγματικούς σε δεκαδικό σύστημα, ή ακόμη χαρακτήρες κ.λπ. Έτσι ο αριθμός 17 του δεκαδικού συστήματος ισοδυναμεί με τον αριθμό 00010001 του δυαδικού συστήματος, ο οποίος μπορεί να αποθηκευθεί σε ένα byte. Αν μετακινήσουμε τα ψηφία αυτά κατά μία θέση προς τα αριστερά, δηλαδή αν προσθέσουμε ένα 0 στο τέλος του αριθμού και αγνοήσουμε το αρχικό 0, τότε προκύπτει ο αριθμός 00100010 του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 34 του δεκαδικού συστήματος. Επίσης, με παρόμοιο τρόπο, αν μετακινήσουμε τα ψηφία κατά μία θέση δεξιά, δηλαδή αποκόψουμε το τελευταίο ψηφίο 1 και θεωρήσουμε ένα ακόμη αρχικό 0, τότε προκύπτει ο αριθμός 00001000 του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 8 του δεκαδικού συστήματος. Άρα η ολίσθηση προς τα αριστερά ισοδυναμεί με πολλαπλασιασμό επί δύο, ενώ η ολίσθηση προς τα δεξιά ισοδυναμεί με την ακέραια διαίρεση διά δύο.

Στοιχεία ψευδογλώσσας

1. Σταθερές

Αριθμητικές: χρησιμοποιούνται οι αριθμητικοί χαρακτήρες, το +, το – και το κόμμα ως δεκαδικό σημείο,

Αλφαριθμητικές: σχηματίζονται από οποιουσδήποτε χαρακτήρες εντός διπλών εισαγωγικών,

Λογικές: υπάρχουν δύο, οι Αληθής και Ψευδής.

2. Μεταβλητές

Για τη σύνθεση του ονόματος μιας μεταβλητής χρησιμοποιούνται οι αριθμητικοί χαρακτήρες, οι αλφαβητικοί χαρακτήρες πεζοί και κεφαλαίοι, καθώς και ο χαρακτήρας _ (underscore). Οι μεταβλητές μπορούν επίσης να είναι αριθμητικές, αλφαριθμητικές και λογικές.

3. Τελεστές

Αριθμητικοί: +, –, *, /, ^, **div**, **mod**

Συγκριτικοί: ≤, <, =, ≠, >, ≥

Λογικοί: **και** (σύζευξη), **ή** (διάζευξη), **όχι** (άρνηση).

4. Εκφράσεις

Σχηματίζονται από σταθερές, μεταβλητές, συναρτήσεις, τελεστές και παρενθέσεις.

5. Εντολή εκχώρησης

Μεταβλητή ← έκφραση

6. Σχήματα λογικών υποθέσεων

```
Αν <συνθήκη> τότε <εντολή>
```

```
Αν <συνθήκη> τότε
   <διαδικασία_1>
αλλιώς
   <διαδικασία_2>
Τέλος_αν
```

```
Επίλεξε έκφραση
   Περίπτωση 1
       Διαδικασία_1
    ............
   Περίπτωση ν
       Διαδικασία_ν
   Περίπτωση αλλιώς
       Διαδικασία_αλλιώς
Τέλος_επιλογών
```

```
Αν <συνθήκη_1> τότε
   <διαδικασία_1>
αλλιώς_αν <συνθήκη_2> τότε
   <διαδικασία_2>
 ......
αλλιώς_αν <συνθήκη_ν> τότε
   <διαδικασία_ν>
αλλιώς
   <διαδικασία_αλλιώς>
Τέλος_αν
```

όπου ως διαδικασία λαμβάνεται ένα σύνολο εντολών

7. Επαναληπτικές διαδικασίες

- Επαναληπτικό σχήμα με έλεγχο επανάληψης στην αρχή

```
Όσο <συνθήκη> επανάλαβε
    Διαδικασία
Τέλος_επανάληψης
```

- Επαναληπτικό σχήμα με έλεγχο επανάληψης στο τέλος

```
Αρχή_επανάληψης
    Διαδικασία
Μέχρις_ότου <συνθήκη>
```

- Επαναληπτικό σχήμα ορισμένων φορών επανάληψης

```
Για μεταβλητή από τ1 μέχρι τ2 με_βήμα β
   Διαδικασία
Τέλος_επανάληψης
```

8. Ρήματα σε προστακτική

Για παράδειγμα, “Διάβασε”, “Γράψε”, “Εκτέλεσε” κ.λπ.

9. Ουσιαστικά

Σε ορισμένες περιπτώσεις όταν οι ζητούμενες ενέργειες είναι πολλές ή προφανείς, καθορίζονται με τη χρήση ουσιαστικών αντί ρημάτων, όπως “εισαγωγή δεδομένων”, “εμφάνιση πεδίων στην οθόνη” κ.λπ.

10. Σχόλια

Προκειμένου να διαχωρίζονται οι επεξηγηματικές φράσεις από τις λέξεις-κλειδιά του αλγορίθμου, στις πρώτες προτάσσεται το σύμβολο !, για παράδειγμα !Σχόλια.

11. Πρώτη και τελευταία γραμμή ενός αλγορίθμου είναι αντίστοιχα

Αλγόριθμος <όνομα_αλγορίθμου> και **Τέλος** <όνομα_αλγορίθμου>

12. Δεδομένα και αποτελέσματα

Τα δεδομένα εισόδου (αν υπάρχουν) περιγράφονται στη δεύτερη γραμμή του αλγορίθμου εντός των συμβόλων //... //. Αντίστοιχα τα αποτελέσματα εξόδου δίνονται στην προτελευταία γραμμή του αλγορίθμου εντός των συμβόλων //... //.

Δομές δεδομένων και Αλγόριθμοι

- **Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:**
  - **Υλικού**. Το υλικό, δηλαδή η μηχανή, επιτρέπει στα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη και στις περιφερειακές συσκευές του υπολογιστή με διάφορες αναπαραστάσεις (representations). Τέτοιες μορφές είναι η δυαδική, ο κώδικας ASCII (βλ. παράρτημα), ο κώδικας EBCDIC, το συμπλήρωμα του 1 ή του 2 κ.λπ.
  - **Γλωσσών προγραμματισμού.** Οι γλώσσες προγραμματισμού υψη-λού επιπέδου (high level programming languages) επιτρέπουν τη χρήση διάφορων τύπων (types) μεταβλητών (variables) για να περι-γράψουν ένα δεδομένο. Ο μεταφραστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης, από πλευράς υλικού, κάθε μεταβλητής στον ΗΥ.
  - **Δομών δεδομένων.** Δομή δεδομένων (data structure) είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών. Για παράδειγμα, μία τέτοια δομή είναι η εγγραφή (record), που μπορεί να περιγράφει ένα είδος, ένα πρόσωπο κ.λπ. Η εγγραφή αποτελείται από τα πεδία (fields) που αποθηκεύουν χαρακτηριστικά (attributes) διαφορετικού τύπου, όπως για παράδειγμα ο κωδικός, η περιγραφή κ.λπ. Άλλη μορφή δομής δεδομένων είναι το αρχείο που αποτελείται από ένα σύνολο εγγραφών. Μία επιτρεπτή λειτουργία σε ένα αρχείο είναι η σειριακή προσπέλαση όλων των εγγραφών του.

- **Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:**
  - **Ανάλυσης δεδομένων.** Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα. Οι τεχνολογίες των Βάσεων Δεδομένων (Databases), της Μοντελοποίησης Δεδομένων (Data Modelling) και της Αναπαράστασης Γνώσης (Knowledge Representation) ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

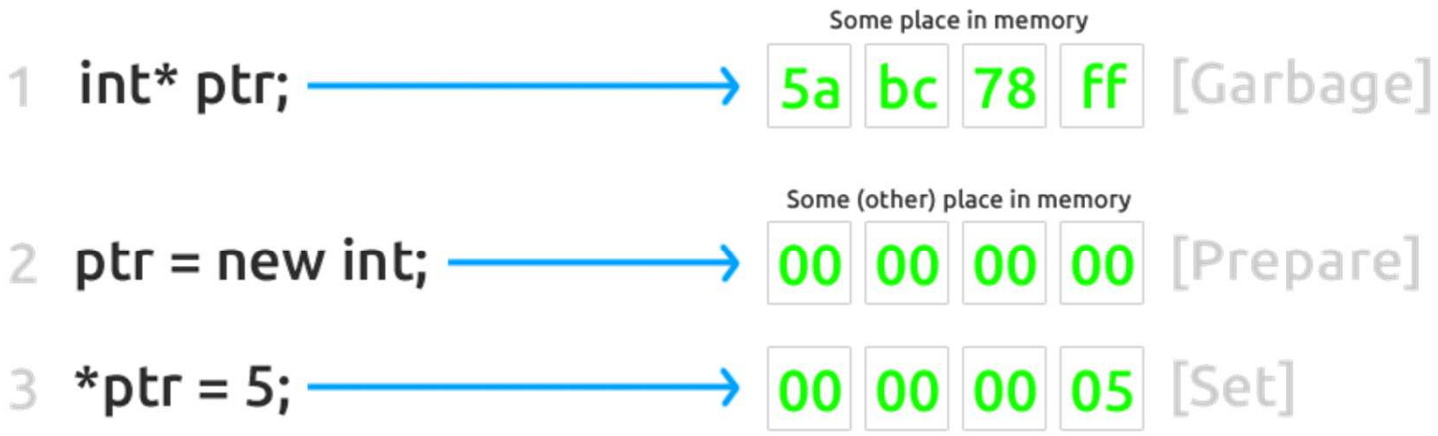
- **Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στη δευτερεύουσα μνήμη του.**
- **Η αποθήκευση αυτή δεν γίνεται κατά έναν τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή.**
- **Η έννοια της δομής δεδομένων (data structure) είναι σημαντική για την Πληροφορική και ορίζεται με τον ακόλουθο τυπικό ορισμό.**
- **Δομή Δεδομένων είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.**

- **Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes).**
- **Οι βασικές λειτουργίες (ή αλλιώς πράξεις) επί των δομών δεδομένων είναι οι ακόλουθες:**
  - **Προσπέλαση (access),** πρόσβαση σε έναν κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.
  - **Εισαγωγή (insertion),** δηλαδή η προσθήκη νέων κόμβων σε μία υπάρχουσα δομή.
  - **διαγραφή (deletion),** που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας κόμβος αφαιρείται από μία δομή.
  - **Αναζήτηση (searching),** κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν κόμβοι που έχουν μια δεδομένη ιδιότητα.
  - **ταξινόμηση (sorting),** όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
  - **Αντιγραφή (copying),** κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.
  - **Συγχώνευση (merging),** κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.
  - **διαχωρισμός (separation),** είναι η αντίστροφη πράξη της συγχώνευσης.

- **Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή.**
- **Συνηθέστατα παρατηρείται το φαινόμενο μία δομή δεδομένων να είναι αποδοτικότερη από μία άλλη δομή με κριτήριο κάποια λειτουργία, για παράδειγμα την αναζήτηση, αλλά λιγότερο αποδοτική για κάποια άλλη λειτουργία, για παράδειγμα την εισαγωγή.**
- **Αυτές οι παρατηρήσεις εξηγούν αφ' ενός την ύπαρξη διαφορετικών δομών, και αφ' ετέρου τη σπουδαιότητα της επιλογής της κατάλληλης δομής κάθε φορά.**

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

- Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες:
  - τις στατικές (static) και
  - τις δυναμικές (dynamic).
- Οι δυναμικές δομές δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης (dynamic memory allocation).
- Με άλλα λόγια, οι δομές αυτές δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται αντίστοιχα.
- Όλες οι σύγχρονες γλώσσες προγραμματισμού προσφέρουν τη δυνατότητα δυναμικής παραχώρησης μνήμης.

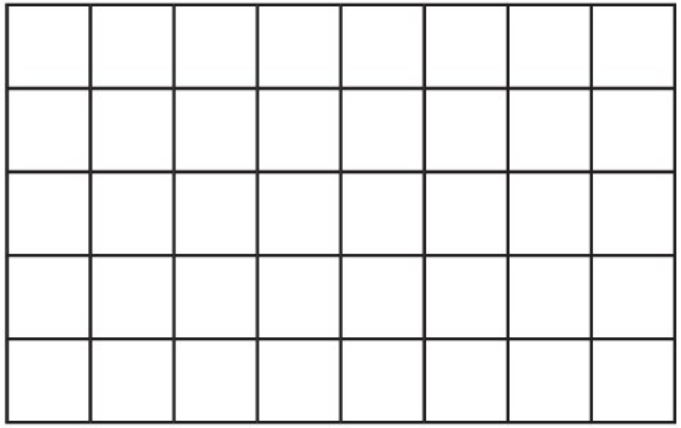


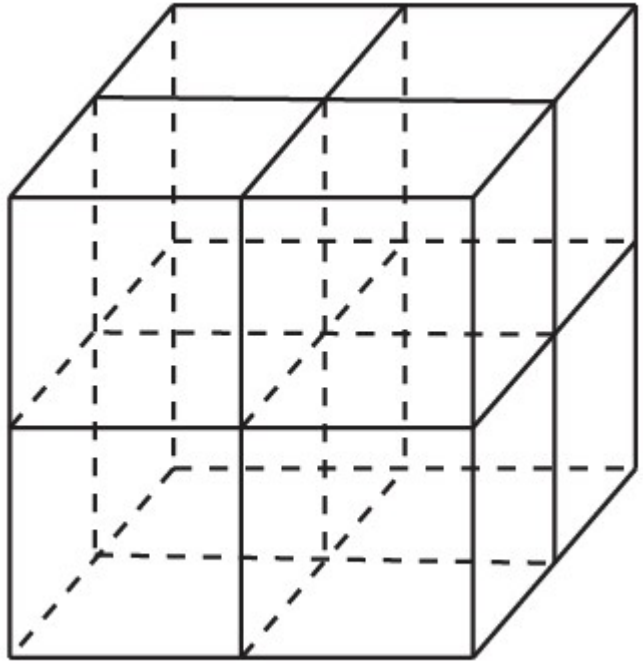
Πίνακες

- Με τον όρο στατική δομή δεδομένων εννοείται ότι το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους και κατά συνέπεια κατά τη στιγμή της μετάφρασής τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος.
- Μία άλλη σημαντική διαφορά είναι ότι τα στοιχεία των στατικών δομών αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.
- Στην πράξη, οι στατικές δομές υλοποιούνται με πίνακες και υποστηρίζονται από κάθε γλώσσα προγραμματισμού.
- Μπορούμε να ορίσουμε τον πίνακα ως μια δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς κ.λπ).
- Η δήλωση των στοιχείων ενός πίνακα και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται.
- Όμως, γενικά η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων δεικτών (indexes) σε παρένθεση ή αγκύλη.

Πίνακες

- Ένας πίνακας μπορεί να είναι μονοδιάστατος, δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος.
- Όσον αφορά στους δισδιάστατους πίνακες σημειώνεται ότι, αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται τετραγωνικός (square) και γενικά συμβολίζεται ως πίνακας nxn.
- Μάλιστα μπορούμε να θεωρήσουμε το δισδιάστατο πίνακα ότι είναι ένας μονοδιάστατος πίνακας, όπου κάθε θέση του περιέχει ένα νέο μονοδιάστατο πίνακα.



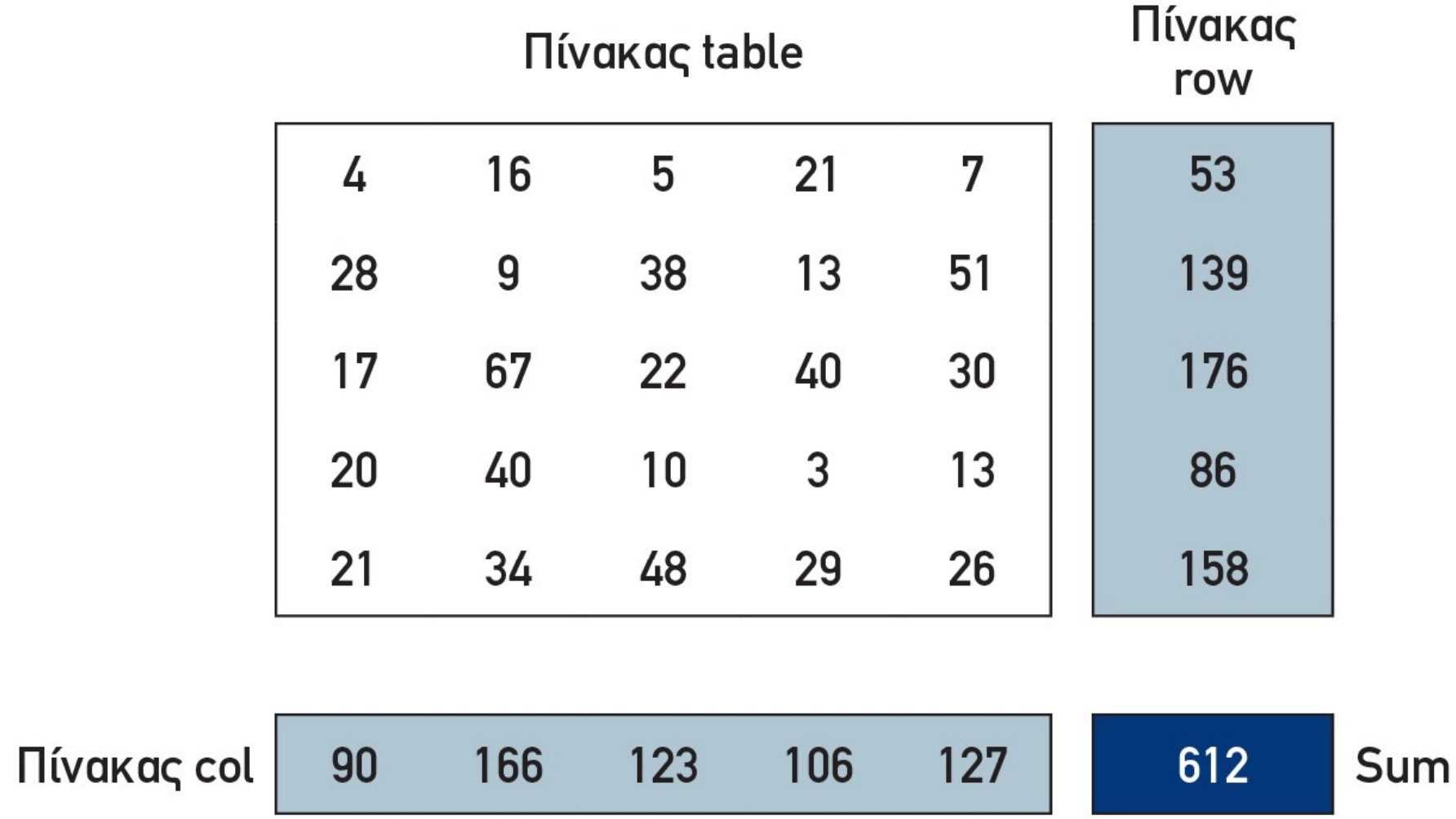


ΠΑΡΑΔΕΙΓΜΑ 1. Εύρεση του μικρότερου στοιχείου ενός μονοδιάστατου πίνακα

Δίνεται ένας μονοδιάστατος πίνακας table 100 στοιχείων. Να σχεδιασθεί αλγόριθμος που να βρίσκει το μικρότερο στοιχείο του.

```
Αλγόριθμος Ελάχ_Πίνακα
Δεδομένα // table //
Min ← table[1]
Για i από 2 μέχρι 100
   Αν table[i] < Min τότε Min ← table[i]
Τέλος_επανάληψης
Αποτελέσματα //Min//
Τέλος Ελάχ_Πίνακα
```

Στον αλγόριθμο αυτό αρχικά το πρώτο στοιχείο του πίνακα εκχωρείται στη μεταβλητή Min. Στη συνέχεια κάθε επόμενο στοιχείο του πίνακα εξετάζεται, αν είναι μικρότερο της Min και αν ναι, τότε αντικαθιστά το προηγούμενο. Έτσι στο τέλος θα υπάρχει στη μεταβλητή Min το μικρότερο στοιχείο όλου του πίνακα table.



Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση δύο σπουδαίων δομών, της στοίβας (stack) και της ουράς (queue), που θα εξεταστούν λεπτομερέστερα στη συνέχεια, επειδή χρησιμοποιούνται σε πληθώρα πρακτικών εφαρμογών.