"When you have eliminated the impossible,
whatever remains, however improbable,
must be the truth"

Sir Arthur Conan Doyle, *The Sign of the Four*, 1890

1 ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΜΕ Η/Υ

1.1 Εισαγωγη

Αν και ο αριθμός των προβλημάτων που επιλύονται με Ηλεκτρονικό Υπολογιστή (Η/Υ) αυξάνει καθημερινά και μάλιστα με εκρηκτικούς ρυθμούς, εντούτοις είναι γνωστό ότι υπάρχουν προβλήματα τα οποία δεν δύνανται ακόμη να επιλυθούν, όπως και προβλήματα για τα οποία έχει αποδειχθεί ότι δεν θα μπορέσουν να επιλυθούν ποτέ. Στο παρόν κεφάλαιο θα προσδιορίσουμε τα χαρακτηριστικά των προβλημάτων που μπορούν να επιλυθούν με τη βοήθεια Η/Υ καθώς και τη διαδικασία επίλυσής τους. Στο πλαίσιο αυτό θα εισαχθούν οι έννοιες των υπολογιστικών ή αλγοριθμικών προβλημάτων, δηλαδή των προβλημάτων εκείνων που μπορούν να επιλυθούν με Η/Υ, καθώς και η έννοια του αλγορίθμου που χρησιμοποιείται ως διαδικασία επίλυσής τους. Θα παρουσιάσουμε επίσης μία ψευδογλώσσα περιγραφής των αλγορίθμων, η οποία έχει τη δομή μιας δομημένης γλώσσας προγραμματισμού χωρίς να περιλαμβάνει τις λεπτομέρειές της. Με την ψευδογλώσσα αυτή αναδεικνύονται τα βασικά στοιχεία της δομής ενός αλγορίθμου και διευκολύνεται ο προσδιορισμός των συναρτήσεων υπολογιστικής πολυπλοκότητάς του, ο έλεγχος της ορθότητάς του και ο έλεγχος του τερματισμού του.

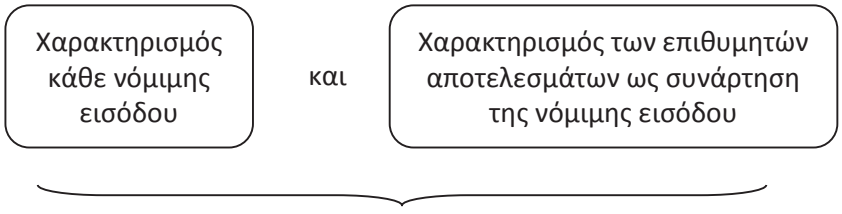
1.2 ΑΛΓΟΡΙΘΜΟΙ-ΑΛΓΟΡΙΘΜΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

Ένα πρόβλημα εκφράζεται συνήθως ως μία εντολή-οδηγία που πρέπει να εκτελεστεί ή, ακόμη πιο συχνά, ως ένα ερώτημα που θα πρέπει να απαντηθεί.

Στην περίπτωση της εντολής-οδηγίας αυτό μπορεί να λάβει μία μορφή όπως η ακόλουθη: «Να προσδιοριστούν οι τιμές των μεταβλητών που ικανοποιούν το ακόλουθο σύστημα εξισώσεων…». Στην περίπτωση ερωτήματος, μία ενδεικτική μορφή μπορεί να είναι η: «ποιες είναι οι τιμές των μεταβλητών που ικανοποιούν το πιο κάτω σύστημα εξισώσεων;»

- **Εξειδίκευση προβλήματος.**

Το πρώτο βήμα για την επίλυση ενός προβλήματος με Η/Υ είναι η *εξειδίκευσή* του (specification). Με την εξειδίκευση ενός προβλήματος προσδιορίζονται κατά τον πλέον σαφή και ευκρινή τρόπο τα χαρακτηριστικά και οι ιδιότητες που πρέπει να πληρούν τα δεδομένα εισόδου του, τα οποία βέβαια πρέπει να είναι συναφή προς το περιεχόμενό του. Τα δεδομένα αυτά ονομάζονται **αποδεκτά** ή **νόμιμα** δεδομένα. Ορίζονται επίσης επακριβώς τα επιθυμητά ή επιδιωκόμενα αποτελέσματα ως συνάρτηση των νόμιμων δεδομένων εισόδου, που αποτελούν την απάντηση στην εντολή ή στο ερώτημα που τίθεται από το πρόβλημα και στοιχειοθετούν μία λύση ή τη λύση αυτού. Με την εξειδίκευση των αποδεκτών δεδομένων εισόδου και τον ορισμό των αποτελεσμάτων εξόδου το πρόβλημα ονομάζεται **αλγοριθμικό** (algorithmic) ή **υπολογιστικό** (computational). Σχηματικά έχουμε:



ΣΧΗΜΑ 1.1: ΣΧΗΜΑΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΑΛΓΟΡΙΘΜΙΚΟΥ – ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Παράδειγμα 1.1. Να ταξινομηθούν οι αριθμοί $a_1, a_2, a_3, ...a_n$ ενός καταλόγου αριθμών K, σε αύξουσα τάξη μεγέθους τους.

Στο πρόβλημα αυτό τα δεδομένα εισόδου είναι μία συλλογή n αριθμών και η απάντηση (λύση) αποτελείται από τους αριθμούς αυτούς ταξινομημένους σε μία αύξουσα ακολουθία.

◆

Παράδειγμα 1.2. Να ευρεθεί αν ο αριθμός x βρίσκεται μέσα σε έναν κατάλογο $K = \{a_1, a_2, a_3, ...a_n\}$, n αριθμών.

Στο πρόβλημα αυτό τα δεδομένα εισόδου είναι μία συλλογή Κ, από n αριθμούς, και ένας αριθμός x. Η απάντηση είναι ναι εάν ο x βρίσκεται εντός του Κ και όχι εάν βρίσκεται εκτός αυτού.

◆

Παράδειγμα 1.3. Να υπολογιστεί το εσωτερικό γινόμενο $\langle \mathbf{a}, \mathbf{b} \rangle$ των διανυσμάτων $\mathbf{a} = (a_1, a_2, a_3, ...a_n)$ και $\mathbf{b} = (b_1, b_2, b_3, ...b_n)$, καθώς και το συνημίτονο $cos\theta$ της γωνίας που σχηματίζουν.

Στο πρόβλημα αυτό τα δεδομένα εισόδου είναι τα διανύσματα $\mathbf{a}$ και $\mathbf{b}$ και ο αριθμός των στοιχείων τους n. Επιδιωκόμενα αποτελέσματα είναι οι αριθμοί (i) του εσωτερικού γινομένου τους $\langle \mathbf{a}, \mathbf{b} \rangle$ και (ii) του συνημίτονου $cos\theta$ της μεταξύ τους γωνίας.

◆

- **Στιγμιότυπο προβλήματος.**

Οι μαθηματικές σχέσεις που προσδιορίζουν ένα πρόβλημα μπορεί να περιέχουν **μεταβλητές, σταθερές** και **παραμέτρους.**

Ειδικότερα, έστω ότι εξετάζουμε αλγεβρικές παραστάσεις και τα «συστατικά» τους. Για το σκοπό αυτό θεωρούμε ότι έχουμε ένα σύνολο Χ, τα στοιχεία του οποίου είναι αριθμοί. Μία **μεταβλητή** x είναι ένα οποιοδήποτε μέλος του συνόλου Χ, και το Χ είναι το **πεδίο ορισμού** (domain) της x. Έτσι λοιπόν, μια μεταβλητή είναι μέλος ενός επακριβώς προσδιορισμένου συνόλου (του πεδίου ορισμού της) οπότε μας είναι απόλυτα γνωστό ποιες τιμές μπορεί- επιτρέπεται να πάρει.

Σε μία αλγεβρική παράσταση, εκτός της μεταβλητής x υπάρχουν ορισμένοι αριθμοί που ονομάζονται **σταθερές** (constants) που συνδυάζονται με τη μεταβλητή βάσει των γνωστών πράξεων της άλγεβρας. Έτσι για παράδειγμα, στο πολυώνυμο $3x^2-2x+5$, οι 3 και -2 είναι σταθερές. Όμως, η παράσταση $3x^2-2x+5$ είναι ένα **στιγμιότυπο** (instance) μιας ολόκληρης κλάσης τετραγωνικών πολυωνύμων με ρητές σταθερές ως συντελεστές, που έχει τη γενική μορφή $y=ax^2+bx+c$. Στο πολυώνυμο αυτό, x είναι η μεταβλητή και a,b,c είναι οι **παράμετροι** (parameters). Έτσι λοιπόν, μπορούμε να πούμε πως οι σταθερές είναι παράμετροι οι τιμές των οποίων έχουν εξειδικευθεί. Στην περίπτωση αυτή, θεωρούμε το $y=ax^2+bx+c$ ως ένα πολυώνυμο, χωρίς να προσδιορίζουμε συγκεκριμένη μορφή για αυτό. Από μια άλλη άποψη, οι παράμετροι δύνανται να θεωρηθούν μεταβλητές όταν θεωρήσουμε ολόκληρη την κλάση των τετραγωνικών πολυωνύμων. Είναι σημαντικό να αναφέρουμε ότι σε κάθε παραμετρική μορφή θα πρέπει να προσδιορίζουμε το πεδίο ορισμού των παραμέτρων. Σύμφωνα με τα παραπάνω, έχουμε ότι:

- **Στιγμιότυπο ενός προβλήματος είναι κάθε εκχώρηση συγκεκριμένων τιμών στις παραμέτρους του.**
- **Το σύνολο των στιγμιότυπων ενός προβλήματος, που προφανώς μπορεί να είναι και απειροσύνολο, αποτελεί το πεδίο ορισμού του (problem domain).**
- **Η απάντηση στο ερώτημα ή στα ερωτήματα που τίθενται από το πρόβλημα αποτελεί τη λύση ενός στιγμιότυπου του προβλήματος.**

Με βάση όσα αναφέρθηκαν, παρουσιάζουμε τα ακόλουθα παραδείγματα:

Παράδειγμα 1.4. Οι παράμετροι του προβλήματος που δόθηκε στο παράδειγμα 1.1 είναι ο κατάλογος K, και ο αριθμός των στοιχείων του n, που αντιπροσωπεύει το μέγεθός του.

Ένα στιγμιότυπο του προβλήματος αυτού είναι το:

$$K=\{15,2,3,10,6,13,9,1\} \text{ με n=8.}$$

Η λύση στο στιγμιότυπο αυτό είναι $K=\{1,2,3,6,9,10,13,15\}$

◆

Παράδειγμα 1.5. Οι παράμετροι του προβλήματος που δόθηκε στο παράδειγμα 1.2 είναι τρεις: ο κατάλογος K, ο αριθμός των στοιχείων του n, και ο ζητούμενος αριθμός x.

Ένα στιγμιότυπο του προβλήματος αυτού είναι το $K = \{15,2,3,10,6,13,9,1\}$ n=8, και x=12. Η λύση στο στιγμιότυπο είναι «όχι, ο x δεν ανήκει στον K».

◆

Παράδειγμα 1.6. Στο πρόβλημα του παραδείγματος 1.3 παράμετροι είναι τα πραγματικά διανύσματα **a** και **b**, και ο αριθμός των στοιχείων τους n. Ένα στιγμιότυπο του προβλήματος είναι αυτό που αντιστοιχεί στις τιμές **a=**(2,5,1,3,2,7,6,4), **b**=(3,5,1,4,7,6,5,8), με n=8.

Η λύση στο πρόβλημα αυτό είναι:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{8} a_i b_i = 162 \text{ και } \cos\theta = \frac{\sum_{i=1}^{8} a_i b_i}{\|a\| \ \|b\|} = \frac{162}{12 \cdot 15} = 0.9$$

Στο παράδειγμά μας, εάν **a** είναι ένα διάνυσμα τιμών προϊόντων και **b** είναι ένα διάνυσμα των αντίστοιχων ποσοτήτων, τότε το εσωτερικό τους γινόμενο μας δίνει την αξία των ποσοτήτων των n προϊόντων που πωλούνται στις αντίστοιχες τιμές.

◆

Θα πρέπει να επισημανθεί ότι οι ιδιότητες που χαρακτηρίζουν τα δεδομένα εισόδου θα πρέπει να είναι αρκούντως ακριβείς, λεπτομερείς και ευκρινείς, αφού ο έλεγχος νομιμότητάς τους γίνεται από τον Η/Υ. Έτσι, το πρόβλημα της εξειδίκευσης ενός προβλήματος δεν είναι πάντα μία εύκολη διαδικασία.

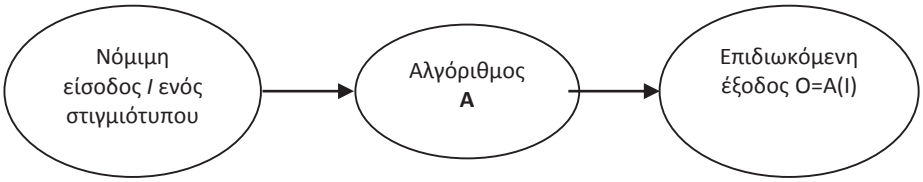
Ερχόμαστε τώρα στο πιο σημαντικό στάδιο, αυτό της επίλυσης του αλγοριθμικού προβλήματος, δηλαδή της ανάπτυξης μίας διαδικασίας η οποία κατά συστηματικό και αλάνθαστο τρόπο θα επιλύει κάθε στιγμιότυπο του προβλήματος μετασχηματίζοντας κάθε νόμιμη είσοδό του στην επιθυμητή λύση. Αν το πρόβλημα είναι απλό και μικρού μεγέθους, τότε η διαδικασία αυτή θα μπορούσε να γίνει «με το χέρι». Στην περίπτωση αυτή θα χρησιμοποιούνταν η νοημοσύνη, οι γνώσεις, η εμπειρία καθώς και η διαίσθηση του επιλύοντος το πρόβλημα, οπότε, αν και θα ήταν επιθυμητό, η διαδικασία αυτή δεν θα χρειάζονταν να περιγράφεται αυστηρά, λεπτομερειακά και με ακρίβεια. Έτσι θα μπορούσε κανείς να βρει τη λύση στο στιγμιότυπο του προβλήματος 1.4 σχεδόν αμέσως.

Είναι όμως προφανές ότι εάν η τιμή της παραμέτρου n ήταν αντί του n=8 το n=1000, τότε, αν και η επίλυση του προβλήματος θα εξακολουθούσε να είναι απλή, ο ανθρώπινος εγκέφαλος δεν θα ήταν σε θέση να εφαρμόσει την ίδια μέθοδο. Είναι επίσης βέβαιο ότι μία τέτοια μέθοδος δεν θα μπορούσε να μετατραπεί σε πρόγραμμα Η/Υ που θα χειριζόταν αρκούντως μεγάλα σύνολα αριθμών.

Για να δημιουργήσουμε λοιπόν ένα πρόγραμμα Η/Υ που θα μπορεί να επιλύσει κάθε στιγμιότυπο ενός προβλήματος, θα πρέπει να αναπτύξουμε μία αντίστοιχη, γενική, βήμα προς βήμα διαδικασία η οποία χωρίς αμφισημίες και πρόσθετες εξηγήσεις, θα μετασχηματίζει τη νόμιμη είσοδο για κάθε στιγμιότυπο του προβλήματος, μονοσήμαντα και σε πεπερασμένο χρόνο, σε μία επιδιωκόμενη έξοδο. Μία τέτοια διαδικασία που ορίζει το περιεχόμενο και τη σειρά των ενεργειών (πράξεων κλπ) που πρέπει να ακολουθηθούν για να λυθεί ένα πρόβλημα, ονομάζεται αλγόριθμος.

Έτσι λοιπόν ένας αλγόριθμος είναι μία ακολουθία υπολογιστικών βημάτων που εφόσον την ακολουθήσουμε με αφετηρία τη νόμιμη είσοδο, θα μας οδηγήσει στην έξοδο, δηλαδή στη λύση του προβλήματος. Στην περίπτωση αυτή λέμε ότι ο αλγόριθμος επιλύει το πρόβλημα υλοποιώντας την αντίστοιχη *αλγοριθμική συνάρτηση*, έστω A, που απεικονίζει μονοσήμαντα κάθε νόμιμη είσοδο, I, σε μία έξοδο : $O = A(I)$.

Έτσι, η επίλυση ενός αλγοριθμικού προβλήματος λαμβάνει την ακόλουθη σχηματική μορφή



Είναι σημαντικό να αναγνωριστεί ότι η ικανοποιητική επίλυση αλγοριθμικών προβλημάτων, η ανάπτυξη δηλαδή κατάλληλων αλγορίθμων, είναι γενικά μία άκρως δύσκολη και απαιτητική διαδικασία, η οποία μπορεί να οδηγήσει σε διαφορετικές «διαδρομές» που οδηγούν από την είσοδο στην έξοδο, δηλαδή σε διαφορετικούς αλγορίθμους που επιλύουν το ίδιο πρόβλημα. Αλγοριθμικά προβλήματα μπορεί να είναι απίστευτα πολύπλοκα και η ικανοποιητική επίλυσή τους ενδέχεται να χρειαστεί χρόνια ερευνητικής προσπάθειας. Και σαν να μην φτάνει αυτό, όπως θα δούμε και παρακάτω, ένας μεγάλος αριθμός προβλημάτων αντιστέκεται σε ικανοποιητικές λύσεις, ενώ άλλα είναι μη

επιλύσιμα, δηλαδή έχει αποδειχθεί ότι δεν θα υπάρξει ποτέ αλγόριθμος επίλυσής τους! Τέλος, υπάρχουν πολλά προβλήματα για τα οποία η ύπαρξη αποτελεσματικών αλγορίθμων δεν έχει ακόμη αποσαφηνιστεί, παρά τη μέχρι στιγμής τεράστια ερευνητική προσπάθεια πολλών επιστημόνων.

Το θέμα της μέτρησης της αποτελεσματικότητας αλγορίθμων (δηλαδή της εύρεσης της «βέλτιστης διαδρομής» μεταξύ στοιχείων εισόδου και εξόδου ενός προβλήματος) θα μας απασχολήσει σε επόμενο κεφάλαιο. Θα πρέπει να πούμε ότι η μαγεία και η τεράστια σημασία της επίλυσης ενός αλγοριθμικού προβλήματος, έγκειται στο γεγονός ότι άπαξ και αναπτυχθεί ένας αποτελεσματικός αλγόριθμος, έστω Α, για την επίλυσή του, τότε η επίλυση κάθε στιγμιοτύπου του με είσοδο I μπορεί να γίνεται μηχανικά από έναν Η/Υ, υλοποιώντας τον μετασχηματισμό $O = A(I)$. Και το πιο σημαντικό, η διαδικασία αυτή εκτελείται με απίστευτη ταχύτητα, απόλυτη ακρίβεια, και από οποιονδήποτε, ακόμη και μη ειδικό, που μπορεί να μην γνωρίζει τίποτα από τη δομή και τις ιδιαιτερότητες του αλγορίθμου.

Μετά την ανάπτυξη ενός αποτελεσματικού αλγορίθμου που αποτελεί τη διαδικασία επίλυσης του αλγοριθμικού προβλήματος, ακολουθεί η αποτύπωσή του σε μία γλώσσα που μπορεί να τον περιγράψει και να τον καταστήσει πλήρως κατανοητό στον άνθρωπο (την ψευδογλώσσα). Πολλές φορές βέβαια περνάμε απευθείας στο επόμενο βήμα που είναι η δημιουργία-γραφή του αντίστοιχου προγράμματος Η/Υ σε μία γλώσσα προγραμματισμού. Ακολουθώντας τους «γραμματικούς» και «συντακτικούς» κανόνες μίας τέτοιας γλώσσας, περιγράφουμε με τρόπο πλήρως κατανοητό για τον Η/Υ την «διαδρομή» (αλγόριθμο) που θέλουμε να ακολουθήσουμε για την επίλυση ενός προβλήματος. Με την γλώσσα προγραμματισμού θα πούμε στον υπολογιστή πως θα διαβάσει τα δεδομένα του προβλήματος, πως θα τα αποθηκεύσει στη μνήμη του, πώς να εκτελέσει τα βήματα του αλγορίθμου, και πώς να παρουσιάσει τη λύση του προβλήματος, δηλαδή τα επιδιωκόμενα αποτελέσματα.

Ο μετασχηματισμός του αλγορίθμου σε μορφή κατανοητή από έναν υπολογιστή ονομάζεται προγραμματισμός και το αποτέλεσμα του μετασχηματισμού αυτού πρόγραμμα, το οποίο αποτελεί το **λογισμικό** (software) για τους Η/Υ. Το πρόγραμμα αυτό μετατρέπεται από τους **μεταγλωττιστές** (compilers) ή **διερμηνευτές** (interpreters) σε ακολουθίες εντολών μηχανής μέσω ενός συνόλου υπορουτινών (οι οποίες είναι και αυτές προφανώς αλγόριθμοι). Το σύνολο των υπορουτινών αυτών ονομάζεται **λειτουργικό σύστημα** (operating system) και η ανάπτυξη και βελτίωσή του αποτελεί σημαντική ερευνητική

περιοχή της επιστήμης των υπολογιστών. Εάν η διαδικασία της μετάφρασης σε εντολές-γλώσσα μηχανής γίνεται πριν από την εκτέλεση του προγράμματος η διαδικασία ονομάζεται μεταγλώττιση (compilation) και το πρόγραμμα που την εκτελεί μεταγλωττιστής (compiler). Αν μέρος της μετάφρασης γίνεται κατά την εκτέλεση του προγράμματος, η διαδικασία ονομάζεται διερμηνεία (interpretation) και το πρόγραμμα διερμηνευτής (interpreter).

Οι μεταγλωττιστές και οι διερμηνευτές είναι αμφότεροι λογισμικό, δηλαδή προγράμματα που μετατρέπουν ένα πρόγραμμα γραμμένο σε γλώσσα προγραμματισμού (πηγαίο κώδικα-source code), σε μία γλώσσα (γλώσσα «στόχο»-target language) που γίνεται «κατανοητή» από τον Η/Υ και που συχνά αναφέρεται ως γλώσσα μηχανής. Η διαφορά τους είναι ότι οι μεταγλωττιστές μεταφράζουν πρώτα όλο το πρόγραμμα στη γλώσσα «στόχο» και μετά το εκτελούν, ενώ οι διερμηνευτές εκτελούν μια-μια τις εντολές-γραμμές κώδικα του πηγαίου προγράμματος.

1.3 ΤΟ ΜΑΘΗΜΑΤΙΚΟ ΜΟΝΤΕΛΟ

Προκειμένου να επιλύσουμε με Η/Υ ένα πρόβλημα Μηχανικού, αλλά και πιο γενικά ένα πρόβλημα από οποιαδήποτε άλλη επιστημονική περιοχή, θα πρέπει αρχικά να δημιουργήσουμε το αντίστοιχο **μαθηματικό μοντέλο** (mathematical model), και στη συνέχεια να αναπτύξουμε έναν κατάλληλο αλγόριθμο, που να επιλύει το πρόβλημα, ει δυνατόν κατά τον πιο αποτελεσματικό τρόπο.

Ένα τέτοιο μοντέλο αποτελείται συνήθως (ορίζεται) από ένα σύστημα ανισοεξισώσεων που εκφράζουν τις κυριότερες, τις πιο σημαντικές σχέσεις μεταξύ των βασικών μεγεθών του προβλήματος, καθώς και τις υποθέσεις συμπεριφοράς των μεγεθών αυτών.

Υπάρχουν όμως πολυάριθμα, ποικίλα και άκρως ενδιαφέροντα προβλήματα, τα οποία μοντελοποιούνται ως γραφήματα. Ένα χαρακτηριστικό πρόβλημα αυτής της κατηγορίας είναι το πρόβλημα του περιοδεύοντος πωλητή, το οποίο παρουσιάζεται σε λεπτομέρεια στο κεφ. 2.2.

Το μοντέλο αυτό περιλαμβάνει σταθερές, παραμέτρους, και μεταβλητές. Οι μεταβλητές εκφράζουν μεγέθη που μεταβάλλονται, και οι τιμές που παίρνουν κατά την επίλυση του προβλήματος αποτελούν μια λύση του. Οι παράμετροι είναι και αυτές μεταβλητές, οι τιμές των οποίων όμως δεν μεταβάλλονται μέσα στο ίδιο στιγμιότυπο, αλλά από στιγμιότυπο σε στιγμιότυπο εντός της ίδιας κλάσης (κατηγορίας) προβλημάτων. Οι παράμετροι δηλαδή χρησιμοποιούνται

για να εκφράσουν το μοντέλο στην πιο γενική του μορφή. Έτσι, με κάθε εκχώρηση συγκεκριμένων τιμών στις παραμέτρους ενός μοντέλου, παίρνουμε το αντίστοιχο στιγμιότυπό του. Είναι προφανές ότι κάτι που επιλέγεται ως παράμετρος σε μία κατηγορία προβλημάτων μπορεί να θεωρηθεί ως μεταβλητή σε μία άλλη και το αντίστροφο.

Από τα παραπάνω προκύπτει ότι ένα μαθηματικό μοντέλο αποτελεί ταυτόχρονα μία γενίκευση αλλά και μία απλοποίηση της πραγματικότητας. Επιπλέον, η δημιουργία ενός κατάλληλου μοντέλου είναι πολύ σημαντική και ταυτόχρονα άκρως απαιτητική ενασχόληση, ειδικά για την επιστήμη του Μηχανικού, αλλά και ευρύτερα για κάθε άλλη επιστήμη.

Ειδικότερα, για τη δημιουργία ενός μαθηματικού μοντέλου απαιτούνται συνήθως τα εξής:

- ✓ Ο καθορισμός των θεμελιωδών αρχών που διέπουν το πρόβλημα (φυσικοί και άλλοι νόμοι και κανόνες και μαθηματικές περιγραφές τους)
- ✓ Ο σχεδιασμός υποβοηθητικών σχημάτων και διαγραμμάτων για την καλύτερη κατανόησή του.
- ✓ Ο καθορισμός των απαραίτητων μεταβλητών, σταθερών και παραμέτρων, αλλά και του συμβολισμού που θα χρησιμοποιηθεί.
- ✓ Ο μετασχηματισμός του προβλήματος, έτσι ώστε να μπορεί να περιγραφεί με τη χρήση μαθηματικών όρων.
- ✓ Η εξαγωγή των βασικών «δομικών» στοιχείων του προβλήματος που προκύπτουν μέσα από την φυσική περιγραφή του προβλήματος (τμήμα εισαγωγής δεδομένων, τμήμα υπολογισμών και υποτμήματά του, τμήμα εξαγωγής αποτελεσμάτων).
- ✓ Η αναγνώριση και αιτιολόγηση των υποθέσεων και περιορισμών που θα ενυπάρχουν στο φυσικό μοντέλο που περιγράφει το πρόβλημα. Κοινά παραδείγματα τέτοιων υποθέσεων περιλαμβάνουν στοιχεία όπως:
    - Η θεώρηση πως η μάζα ενός σώματος είναι συγκεντρωμένη στο κέντρο μάζας του.
    - Η υπόθεση της ομοιογενούς κατανομής μάζας σώματος εντός του όγκου του.

- Η αμελητέα αντίσταση αέρα κατά την κίνηση σώματος εντός αυτού.
- Οι ενιαίες θερμοφυσικές ιδιότητες (πυκνότητα, αγωγιμότητα κ.λπ.) εντός μάζας ρευστού.
- Η αμελητέα επίδραση τριβών στην κίνηση, κλπ.

1.3.1 ΠΑΡΑΔΕΙΓΜΑ

Να κατασκευάσετε μαθηματικό μοντέλο για το ακόλουθο πρόβλημα και να αναπτύξετε έναν αλγόριθμο επίλυσής του:

Ένα μικρό σώμα βάλλεται από το έδαφος με ταχύτητα 50 μίλια/ώρα υπό γωνία 30 μοιρών από τον ορίζοντα. Να βρεθεί ο χρόνος που χρειάζεται το σώμα και η (οριζόντια) απόσταση που διανύει μέχρι να ξαναβρεθεί στο έδαφος.

Το παραπάνω πρόβλημα είναι καλά ορισμένο, μια και η περιγραφή του περιλαμβάνει το σύνολο των δεδομένων που απαιτούνται για την δημιουργία του κατάλληλου μαθηματικού μοντέλου[2]. Τα παρακάτω είναι ορισμένα από τα σημεία που μπορούν να παρατεθούν για την καλύτερη κατανόηση του προβλήματος.

- ✓ Οι ιδιότητες του σώματος και του μέσου (π.χ. αέρας) στο οποίο γίνεται η βολή, ενδέχεται να επηρεάσουν την τροχιά του. Έτσι, αν το αντικείμενο έχει μικρή μάζα, σχετικά μεγάλη επιφάνεια και βάλλεται στον αέρα, η αντίσταση του μέσου και η ανομοιογένεια στην πυκνότητά του (λόγω, π.χ., διαφορών θερμοκρασίας) μπορεί να επηρεάσουν την τροχιά του. Η κίνηση του αντικειμένου επηρεάζεται επιπλέον από την ταχύτητα του ανέμου. Αν λοιπόν δεν είναι διαθέσιμες τέτοιου είδους πληροφορίες, ή το πρόβλημα που θέλουμε να επιλύσουμε δεν αφορά τον συνυπολογισμό αυτών των παραμέτρων, τότε μπορούμε να θεωρήσουμε ότι το μέσο εντός του οποίου

[2] Επισημαίνεται ότι αυτό δεν είναι πάντα προφανές κατά την έναρξη της επίλυσης ενός προβλήματος, και ότι είναι πιθανό πως θα πρέπει να επανέλθουμε στο βήμα αυτό εφόσον κατά τη διάρκεια της επίλυσης προκύψει ότι μέρος των δεδομένων και πληροφοριών που απαιτούνται για την επίλυση δεν περιλαμβάνεται στον ορισμό του προβλήματος. Στην περίπτωση αυτή, θα συμπληρώσουμε τις ελλείψεις με τη βοήθεια παραδοχών και στη βάση του φυσικού μοντέλου του προβλήματος.

λαμβάνει χώρα η κίνηση δεν έχει κάποια επίδραση στην τροχιά του σώματος[3].

- ✓ Ένα άλλο μέγεθος που επηρεάζει την τροχιά είναι η επιτάχυνση της βαρύτητας. Αν δεν δίνονται άλλες πληροφορίες, είναι λογικό να θεωρήσουμε ότι η βολή λαμβάνει χώρα στην επιφάνεια της Γης, και να χρησιμοποιήσουμε την αντίστοιχη τιμή της επιτάχυνσης της βαρύτητας. Η μεταβολή του μεγέθους αυτού συναρτήσει των γεωγραφικών συντεταγμένων του τόπου και του ύψους, δεν λαμβάνονται υπόψη παρά μόνο εάν επηρεάζουν την τροχιά του σώματος που μελετάται.
- ✓ Απαιτείται αριθμητική ακρίβεια των τιμών της αρχικής ταχύτητας και της γωνίας γιατί αυτή θα καθορίσει την ακρίβεια των αποτελεσμάτων. Προσοχή: η τάξη μεγέθους της επιθυμητής ακρίβειας των αποτελεσμάτων, είναι γενικά αυτή που καθορίζει και την τάξη μεγέθους της ακρίβειας των δεδομένων εισόδου. Είναι λοιπόν επόμενο πως, εφόσον για παράδειγμα επιθυμούμε αριθμητική ακρίβεια του αποτελέσματος ισοδύναμη του ενός δεκαδικού ψηφίου, αυτή θα πρέπει να είναι και η επιδιωκόμενη ακρίβεια των στοιχείων εισόδου. Σημειώνεται ότι όσο αυξάνει η επιθυμητή ακρίβεια των στοιχείων εισόδου, τόσο αυξάνει και το κόστος απόκτησης αυτών των δεδομένων (είτε το υπολογιστικό είτε το πειραματικό). Επίσης, η ακρίβεια των αποτελεσμάτων προσδιορίζεται (αλλά και περιορίζεται) από την ακρίβεια των υπολογιστικών μεθόδων που θα χρησιμοποιηθούν, όπως θα εξηγηθεί σε επόμενο κεφάλαιο.
- ✓ Πολλές φορές είναι απαραίτητη η μετατροπή των μονάδων όπως στην περίπτωσή μας: 1 μίλι = 5280 πόδια = 1609.4 m, 1 ώρα = 60 λεπτά = 3600 δευτερόλεπτα, 360 μοίρες = 2π ακτίνια. Η ετερογένεια των μονάδων σε ένα πρόβλημα αυξάνει την πιθανότητα λάθους κατά την επίλυσή του. Επιπροσθέτως, η ομοιογένεια των μονάδων μέτρησης δίνει τη δυνατότητα στον Μηχανικό να προβεί σε μία γρήγορη επαλήθευση μαθηματικών σχέσεων που χρησιμοποιούνται κατά την επίλυση ενός προβλήματος. Εδώ για παράδειγμα, γνωρίζουμε ότι το πρόβλημά μας ανήκει στην κατηγορία των προβλημάτων κίνησης εντός πεδίου που ακολουθεί το νόμο του αντιστρόφου τετραγώνου, όπου η κατακόρυφη μετατόπιση είναι

[3] Η ενασχόληση με παραμέτρους του προβλήματος οι οποίες θεωρούνται γνωστές, χαρακτηρίζεται από κάποιους σχολαστικισμός. Το να θεωρούμε όμως κάποια πράγματα δεδομένα, γνωστά και κοινώς αποδεκτά από όλους, προϋποθέτει ότι όλοι σκέφτονται και ενεργούν όπως εμείς, μια υπόθεση που απέχει από τη πραγματικότητα εντός της οποίας καλείται να λειτουργήσει ένας Μηχανικός.

συνάρτηση του γινομένου της επιτάχυνσης του πεδίου και του τετραγώνου του χρόνου της κίνησης. Άρα η μετατόπιση θα δίνεται σε [m/s^2] * s^2 δηλαδή θα εκφράζεται σε [m].

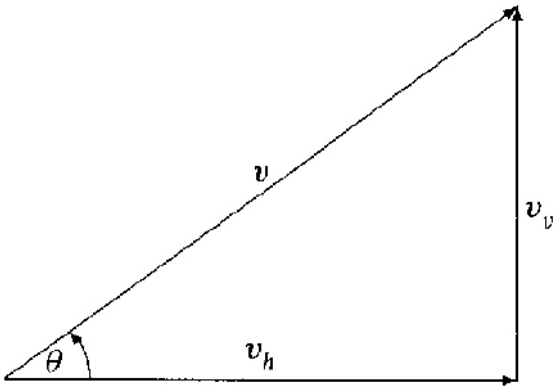
- ✓ Έξοδος ή αποτελέσματα που θα προκύψουν: Είναι σαφές από την παρουσίαση του προβλήματος ότι τα αποτελέσματα που θα προκύψουν θα είναι ο χρόνος και η απόσταση της βολής.
- ✓ Θεωρητική και πρακτική γνώση που θα εφαρμοστεί: Η θεωρία που θα εφαρμοστεί είναι αυτή της βολής (κίνησης) σώματος εντός βαρυτικού πεδίου.
- ✓ Δεδομένα ή πληροφορίες εισόδου: η αρχική ταχύτητα των 50 μιλίων/ώρα και η γωνία 30 μοιρών ως προς το έδαφος, το οποίο θεωρείται οριζόντιο (δηλαδή χωρίς υψομετρική διαφορά κατά μήκος του άξονα της οριζόντιας συνιστώσας της κίνησης).

- **Η δημιουργία του μαθηματικού μοντέλου**

Για να εισάγουμε το μαθηματικό μοντέλο που θα περιγράφει το πρόβλημα, πρέπει πρώτα να καθορίσουμε τον συμβολισμό που θα χρησιμοποιήσουμε:

- ✓ Χρόνος: t(s), με t=0 τη στιγμή που βάλλεται το σώμα.
- ✓ Μέτρο αρχικής ταχύτητας: υ=50 μίλια/ώρα.
- ✓ Αρχική γωνία: θ=30°.
- ✓ Οριζόντια και κατακόρυφη μετατόπιση της μπάλας x(t) και y(t) αντίστοιχα, εκφρασμένη σε πόδια (ft).
- ✓ Επιτ. βαρύτητας: g=32.2 ft/s^2 με κατεύθυνση αντίθετη του άξονα y.

Το σημαντικό στην ανάπτυξη του μαθηματικού μοντέλου είναι η ανάλυση της κίνησης σε δύο συνιστώσες· μία κατά τον οριζόντιο και μία κατά τον κατακόρυφο άξονα. Με αυτόν τον τρόπο αναλύεται η αρχική ταχύτητα, όπως φαίνεται στο Σχήμα 1.2.



ΣΧΗΜΑ 1.2:
ΑΝΑΛΥΣΗ ΤΗΣ ΑΡΧΙΚΗΣ ΤΑΧΥΤΗΤΑΣ U ΣΕ ΔΥΟ ΣΥΝΙΣΤΩΣΕΣ U_h (ΟΡΙΖΟΝΤΙΑ) ΚΑΙ U_v (ΚΑΘΕΤΗ).

Από την τριγωνομετρία γνωρίζουμε ότι:

$$\upsilon_h = \upsilon \cos\theta, \qquad \upsilon_v = \upsilon \sin\theta \tag{1.1}$$

Με βάση αυτές τις συνιστώσες της ταχύτητας μπορούμε να υπολογίσουμε την οριζόντια και κατακόρυφη μετατόπιση του σώματος συναρτήσει του χρόνου. Επειδή κατά την οριζόντια διεύθυνση δεν ασκείται καμία δύναμη στο σώμα, αυτό θα εκτελεί ευθύγραμμη ομαλή κίνηση με σταθερή ταχύτητα υ_h και η οριζόντια μετατόπισή του θα είναι:

$$x(t) = \upsilon t \cos\theta \tag{1.2}$$

Στην κατακόρυφη διεύθυνση, λόγω της δύναμη της βαρύτητας, το σώμα θα εκτελεί ευθύγραμμη ομαλά επιταχυνόμενη (επιβραδυνόμενη) κίνηση και η κατακόρυφη μετατόπισή του θα είναι:

$$y(t) = \upsilon t \sin\theta - \frac{1}{2} g t^2 \tag{1.3}$$

Από την παραπάνω ανάλυση προκύπτουν παράμετροι, μεταβλητές και σταθερές. Έτσι, στις σχέσεις (1.2) και (1.3) σταθερές είναι το 1/2, παράμετροι είναι η επιτάχυνση της βαρύτητας g , η αρχική ταχύτητα υ, και η γωνία βολής θ, ενώ μεταβλητές είναι ο χρόνος t αλλά και οι μετατοπίσεις (οριζόντια και κατακόρυφη).

- **Αλγόριθμος**

Με βάση το μαθηματικό μοντέλο που αναπτύχθηκε παραπάνω, μπορούμε να εξάγουμε σχέσεις που θα δίνουν τα επιθυμητά αποτελέσματα. Το σώμα φτάνει στο έδαφος όταν η κατακόρυφη μετατόπισή του είναι 0, δηλαδή όταν:

$$y(t) = 0 \Leftrightarrow \upsilon t \sin\theta - \frac{1}{2} g t^2 = 0$$

Η παραπάνω εξίσωση μας δίνει δύο λύσεις για τον χρόνο κίνησης:

$$t = 0 \quad \kappa\alpha\iota \quad t = \frac{2\upsilon \sin\theta}{g} \tag{1.4}$$

Η πρώτη λύση αντιστοιχεί στη χρονική στιγμή που γίνεται η βολή και η δεύτερη στο χρόνο που χρειάζεται το σώμα για να ξαναφτάσει στο έδαφος, αντιστοιχεί δηλαδή στο χρόνο που αναζητούμε.

$$t_g = \frac{2\upsilon \sin\theta}{g} \tag{1.5}$$

Η οριζόντια μετατόπιση για το χρόνο αυτό δίδεται από τη σχέση:

$$x(t_g) = \upsilon t_g \cos\theta \tag{1.6}$$

Έτσι, ένας αλγόριθμος για την επίλυση του προβλήματος αυτού θα είναι ο ακόλουθος:

```
Δεδομένα εισόδου: g,u,θ
Αρχή
   Υπολόγισε χρόνο κίνησης t, από σχ. (1.5)
   Υπολόγισε οριζ. μετατόπ. x,βάσει t και σχ. (1.6)
   Εμφάνισε αποτελέσματα υπολογισμών
Τέλος
```

- **Υλοποίηση αλγορίθμου**

Ο παραπάνω αλγόριθμος μπορεί εύκολα να υλοποιηθεί με τη χρήση μιας γλώσσας προγραμματισμού όπως η FORTRAN ή ενός προγραμματιστικού περιβάλλοντος, όπως το MATLAB. Ακολούθως παρατίθεται ένα τέτοιο πρόγραμμα γραμμένο για MATLAB, το οποίο επιλέγεται στο παράδειγμα αυτό χάρη στις αυξημένες δυνατότητές του για την εύκολη κατασκευή γραφικών. Ο κώδικας έχει ως εξής:

```
% Αρχικές τιμές
g = 32.2; % Επιτ. βαρύτητας, ft/s^2
v = 50 * 5280/3600; % Αρχική ταχύτητα, ft/s
theta = 30 * pi/180; % Γωνία βολής, radians
% Υπολογισμός και εξαγωγή αποτελεσμάτων
disp('Xronos ptisis (s):') % Λεζάντα για το χρόνο πτήσης
tg = 2 * v * sin(theta)/g % Χρόνος επιστροφής στο έδαφος, s
disp('Apostasi ptisis (ft):') % Λεζάντα για την απόσταση
xg = v * cos(theta) * tg % Διανυόμενη απόσταση
% Υπολογισμός και γραφική παράσταση αποτελεσμάτων
t = linspace(0,tg,256);
x = v * cos(theta) * t;
y = v * sin(theta) * t - g/2 * t.^2;
plot(x,y), axis equal, axis([ 0 150 0 30 ]), grid
xlabel('Apostasi(ft)')
ylabel('Ypsos(ft)')
title('Ixnos ptisis')
```

Στον παραπάνω κώδικα προγράμματος, σημειώνουμε ότι:

- ✓ Οι λέξεις που έπονται του συμβόλου % αποτελούν σχόλια που γράφονται για την καλύτερη αναγνωσιμότητα του προγράμματος.
- ✓ Μία λέξη που βρίσκεται αριστερά του συμβόλου «ίσον» αντιστοιχεί στο όνομα μίας μεταβλητής. Σε αυτήν εκχωρείται η τιμή ή οι τιμές που βρίσκονται δεξιά του «ίσον».
- ✓ Η εντολή **`t=linspace(0,tg,256)`** και η έκφραση **`t.^2`** θα επεξηγηθούν στο τρίτο μέρος του βιβλίου, όπου θα παρουσιάσουμε το περιβάλλον προγραμματισμού MATLAB.
- ✓ Επιπλέον των απαιτούμενων αποτελεσμάτων, υπολογίζεται και σχεδιάζεται η τροχιά που διαγράφει το σώμα, η οποία θα χρησιμοποιηθεί παρακάτω στην πιστοποίηση της λύσης. Αυτό γίνεται με χρήση της εντολής **`plot`**, η οποία δημιουργεί τη γραφική παράσταση του ύψους σε σχέση με την απόσταση και της δίνει ένα τίτλο, καθώς και λεζάντες στους άξονες x και y.

Όλα αυτά αποτελούν στοιχεία του προγραμματισμού, του «λεξιλογίου», των τεχνολογικών στοιχείων του και των ιδιαιτεροτήτων του, συναντώνται σε κάθε γλώσσα ή περιβάλλον προγραμματισμού, και θα μας απασχολήσουν στις επόμενες δύο ενότητες του βιβλίου.

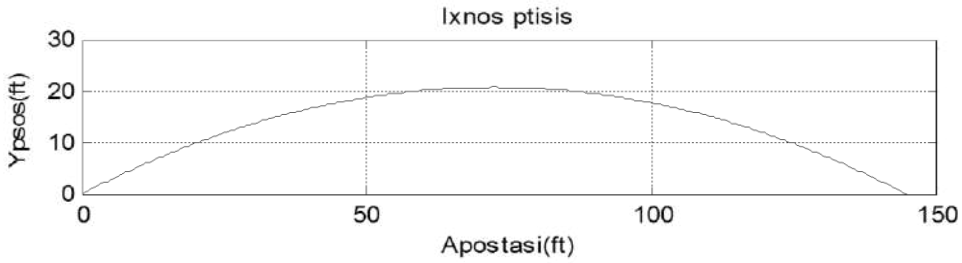
- **Δοκιμή και πιστοποίηση επίλυσης**

Το πρόβλημα αυτό είναι αρκετά απλό και η λύση του μπορεί να υπολογιστεί με το χέρι ή με τη χρήση μίας αριθμομηχανής.

Η εκτέλεση των εντολών MATLAB που παρουσιάστηκαν παραπάνω εκτυπώνει στην οθόνη τα ακόλουθα αποτελέσματα:

```
Xronos ptisis (s):
tg =
    2.2774
Apostasi ptisis (ft):
xg =
  144.6364
```

Ο υπολογισμός αυτών των μεγεθών με τη χρήση μίας αριθμομηχανής δίνει τα ίδια αποτελέσματα. Η γραφική παράσταση της τροχιάς του σώματος που σχεδιάζεται φαίνεται στο Σχήμα 1.3. Μέσω του σχήματος αυτού παρουσιάζεται ένα από τα πλεονεκτήματα του MATLAB, που είναι η δυνατότητα εύκολης παραγωγής γραφικών παραστάσεων. Η συγκεκριμένη γραφική παράσταση μας βοηθά στην πιστοποίηση της λύσης, αφού μπορούμε, βασιζόμενοι στην εμπειρία μια τη γνώση μας, να διαπιστώσουμε ότι αποτελεί μία λογική λύση τους προβλήματός μας, μια και απεικονίζει μία παραβολική τροχιά.



ΣΧΗΜΑ 1.3: ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΤΗΣ ΤΡΟΧΙΑΣ ΤΟΥ ΣΩΜΑΤΟΣ

1.4 ΜΕΛΕΤΗ ΑΛΓΟΡΙΘΜΩΝ

Από την παραπάνω ανάλυση προκύπτει ότι οι αλγόριθμοι αποτελούν τον πυρήνα και το πνεύμα της επιστήμης των Η/Υ. Αυτό δεν είναι καθόλου παράξενο, καθώς κάθε περιοχή των Η/Υ εξαρτάται ουσιωδώς από τον σχεδιασμό και την ανάλυση αποτελεσματικών αλγορίθμων.

Η μελέτη αλγορίθμων απαρτίζεται από δύο περιοχές: τη σχεδίαση ενός αλγορίθμου που θα επιλύει το συγκεκριμένο πρόβλημα, και την ανάλυση του αλγορίθμου αυτού. Οι δύο αυτές περιοχές είναι στενά συνδεδεμένες και αλληλοσυμπληρούμενες. Αυτό συμβαίνει διότι η ανάλυση αλγορίθμων μας εφοδιάζει με στοιχεία σχετικά με τη δομή τους, απαραίτητα για τον σχεδιασμό νέων, πιο αποτελεσματικών αλγορίθμων. Η ανάλυση αλγορίθμων περιλαμβάνει τις ακόλουθες περιοχές:

- **Ορθότητα αλγορίθμου (algorithm correctness).**

Όταν μας δίνεται ένας αλγόριθμος, είναι σημαντικό να γνωρίζουμε εάν παράγει τα σωστά αποτελέσματα για κάθε στιγμιότυπο του προβλήματος. Ο έλεγχος της ορθότητας αλγορίθμων αποτελεί μία ιδιαίτερα απαιτητική περιοχή έρευνας και χρησιμοποιεί αρκετά προχωρημένες λογικο-μαθηματικές τεχνικές. Το χαρακτηριστικό αυτό, σε συνδυασμό με το ότι ο έλεγχος της ορθότητας των αλγορίθμων με τους οποίους θα ασχοληθούμε στο βιβλίο αυτό δεν παρουσιάζει κάποια ιδιαίτερη δυσκολία, μας επιτρέπει να μην ασχοληθούμε περαιτέρω με το θέμα αυτό.

- **Τερματισμός αλγορίθμου (algorithm termination).**

Λέμε ότι ένας αλγόριθμος τερματίζει, εάν περατώνεται για κάθε μία από τις νόμιμες εισόδους του. Έτσι, ο τερματισμός είναι μία από τις βασικές ιδιότητες ενός αλγορίθμου, αφού διαφορετικά δεν παράγει τη λύση.

Ως μία πρώτη προσέγγιση προς την κατεύθυνση αυτή, κάθε εντολή θα πρέπει να εκτελείται σε πεπερασμένο χρόνο, κάθε είσοδος θα πρέπει να έχει πεπερασμένο μήκος, και το κυριότερο, δεν επιτρέπονται ατέρμονοι βρόχοι, δηλαδή επαναλήψεις της ίδιας διαδικασίας, ή ανακυκλήσεις[4], που λαμβάνουν

[4] Ο αγγλικός όρος loop μεταφράζεται στα ελληνικά ως βρόχος, με την έννοια της κλειστής επαναλαμβανόμενης διαδρομής, ή ανακύκληση, με την έννοια της κυκλικής επανόδου, της

χώρα επ'άπειρον και δεν τερματίζουν ποτέ. Ο έλεγχος του τερματισμού είναι γενικά δύσκολος, αν και σε απλά προβλήματα δύναται να είναι και εξαιρετικά εύκολος.

Παράδειγμα 1.7 Η παρακάτω ακολουθία εντολών δεν τερματίζει.

Είσοδος x:=4

Όσο $x \neq 2$ **επανάλαβε**

Προσέθεσε στο x το 2, και διαίρεσε το αποτέλεσμα με το 2

Αντικατέστησε την προηγούμενη τιμή του x με το αποτέλεσμα που προέκυψε.

Δηλαδή θέσε $x \coloneqq \frac{x+2}{2}$

Τέλος όσο

Έξοδος x

Αν και η παραπάνω διαδικασία φαίνεται εκ πρώτης όψεως ότι τερματίζει, εντούτοις αυτό δεν συμβαίνει. Ο λόγος είναι πως η ακολουθία τιμών του x που παράγεται από την επαναληπτική διαδικασία, είναι η

4, 3, 2.5, 2.125, …,

η οποία έχει όριο το 2, δηλαδή συγκλίνει στο 2 χωρίς ποτέ να λαμβάνει αυτή την τιμή. Αυτό σημαίνει ότι η συνθήκη $x \neq 2$ της ανακύκλησης δεν ικανοποιείται ποτέ, οπότε ο βρόχος επαναλαμβάνεται επ'άπειρον, με αποτέλεσμα να μη μπορεί να παραχθεί ποτέ μία τελική τιμή της x.

◆

Ο έλεγχος τερματισμού του παραπάνω αλγορίθμου ήταν απλός. Δυστυχώς όμως αυτό δεν συμβαίνει πάντα. Για τον απλό αλγόριθμο που ακολουθεί, παρά τις τεράστιες προσπάθειες δεκάδων ετών, δεν επιτευχθεί ακόμη να αποδειχθεί ο τερματισμός του.

επιστροφής στην αρχή. Η χρήση των παρώνυμων «βρόγχος» ή «ανακύκλωση» αντίστοιχα είναι λανθασμένη.

Παράδειγμα 1.8 Να ελεγχθεί ο τερματισμός του παρακάτω αλγορίθμου (εικασία του Collatz[5])

```
όσο x ≠ 1 κάνε
    αν x είναι άρτιος
        τότε διαίρεσε τον x δια δύο και
            αντικατέστησε την αρχική τιμή του
            με το αποτέλεσμα, δηλαδή
            θέσε x ≔ x/2
        αλλιώς πολλαπλασίασε το x επί 3,
            πρόσθεσε 1 και αντικατέστησε το x
            με το αποτέλεσμα, δηλαδή
            θέσε x ≔ 3x + 1
    τέλος αν
τέλος όσο
```

Εφόσον ο αλγόριθμος αυτός ξεκινήσει με $x = 9$, θα παράξει την ακολουθία τιμών: [28,14,7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1] και θα τερματίσει.

Εάν τρέξουμε τον αλγόριθμο αυτό με διάφορους ακεραίους, θα δούμε ότι στο τέλος τερματίζει. Η ακολουθία των αριθμών που παράγεται πριν τον τερματισμό μπορεί να είναι ασυνήθης και να παλινδρομεί απρόβλεπτα πριν ικανοποιηθεί η συνθήκη τερματισμού $x = 1$. Το περίεργο με τον αλγόριθμο αυτό είναι ότι αν και έχει δοκιμαστεί σκληρά, και πάντα έχει μέχρι τώρα τερματίσει, εν τούτοις δεν βρέθηκε μέχρι στιγμής μαθηματική απόδειξη τερματισμού του για κάθε θετικό ακέραιο. Το πρόβλημα του τερματισμού αυτού του αλγορίθμου αποτελεί ένα δύσκολο ανοικτό πρόβλημα ενός κλάδου μαθηματικών που ονομάζεται θεωρία αριθμών.

◆

Με τα παραδείγματα αυτά είναι εμφανές το πόσο δύσκολο είναι το πρόβλημα του ελέγχου του τερματισμού ενός αλγορίθμου, ακόμη και στην περίπτωση

[5] Η εικασία του Collatz παραμένει ένα άλυτο μαθηματικό πρόβλημα από το 1937 οπότε και διατυπώθηκε από τον Γερμανό μαθηματικό Lother Collatz (1910-1990).). Η εικασία περιγράφει μία διαδικασία βάσει της οποίας κάθε φυσικός αριθμός, καταλήγει να γίνει ίσος με μονάδα.

πολύ απλών αλγορίθμων. Το χειρότερο δε είναι πως το γενικό πρόβλημα του τερματισμού αλγορίθμων δεν είναι απλώς ένα δυσεπίλυτο πρόβλημα, αλλά είναι δυστυχώς **μη επιλύσιμο** (unsolvable). Ειδικότερα, έστω ότι θέλουμε να γράψουμε έναν αλγόριθμο ο οποίος θα μπορεί να εξετάζει οποιονδήποτε άλλο αλγόριθμο και θα είναι σε θέση να διαπιστώνει εάν αυτός θα τερματίσει ή όχι. Είναι προφανές ότι αν ο εξεταζόμενος αλγόριθμος δεν περιέχει βρόχους, ή αναδρομικές κλήσεις υπορουτινών, τότε αναπόφευκτα κάποτε θα τερματίσει. Εάν όμως αυτό δεν ισχύει, η εκτέλεσή του μπορεί κάλλιστα να συνεχίζεται επ΄άπειρον. Αποδεικνύεται ότι δεν υπάρχει αλγόριθμος που να δύναται να εξετάζει έναν άλλον οποιονδήποτε αλγόριθμο και να αποφασίζει αν περιέχει ατέρμονους βρόχους ή όχι.

Το πρόβλημα αυτό, που φέρει τον τίτλο **πρόβλημα τερματισμού** (halting problem), το έθεσε πρώτος ο Alan Touring, και απέδειξε ότι είναι **μη υπολογίσιμο πρόβλημα** (uncomputable problem)[6]. Ειδικότερα, το θέμα δεν είναι ότι κανείς δεν κατόρθωσε μέχρι σήμερα να αναπτύξει έναν αλγόριθμο ελέγχου τερματισμού, αλλά ότι ένας τέτοιος αλγόριθμος είναι αδύνατο να υπάρξει[7]. Είναι επίσης αξιοσημείωτο ότι τα περισσότερα μη υπολογίσιμα προβλήματα που έχουν παρουσιαστεί στην πράξη είναι ισοδύναμα με το πρόβλημα του τερματισμού.

Άλλες περιοχές της ανάλυσης αλγορίθμων περιλαμβάνουν:

- **Πολυπλοκότητα χρόνου και χώρου** (time and space complexity).

Στην περίπτωση αυτή προσδιορίζονται οι υπολογιστικές απαιτήσεις σε χρόνο εκτέλεσης και χώρο μνήμης ενός αλγορίθμου, και επομένως η αποτελεσματικότητα του. Οι απαιτήσεις αυτές εκφράζονται ως συνάρτηση του

[6] Ο A. Touring απέδειξε ότι το πρόβλημα τερματισμού είναι μη αποφασίσιμο σε μηχανές Touring, μία γενικευμένη και θεωρητική αποτύπωση ενός Η/Υ.

[7] Αυτή η διατύπωση ίσως να ξαφνιάζει τον αναγνώστη, που ενδεχομένως θεωρεί ότι στο πεδίο της λογικής και των μαθηματικών ευρύτερα, δεν είναι δυνατό να υπάρχουν προτάσεις (αληθείς ή ψευδείς) επί των οποίων δεν μπορούμε να αποφανθούμε ως προς το εάν αυτές είναι αποδείξιμες-αποφασίσιμες ή όχι. Κάτι τέτοιο όμως είναι γεγονός, και απορρέει από το θεώρημα της μη πληρότητας του Gödel. Έτσι λοιπόν, είναι βέβαιο ότι υπάρχουν προτάσεις (οι οποίες μπορεί να είναι αληθείς), για τις οποίες όμως δεν μπορούμε να αποφανθούμε ως προς το εάν είναι αληθείς ή ψευδείς. Κατ'επέκταση, υπάρχουν προβλήματα με λύση, την οποία δεν θα μπορέσουμε να βρούμε ποτέ και άρα δεν θα μπορέσουμε ποτέ να δημιουργήσουμε τον αντίστοιχο αλγόριθμο.

μεγέθους του προβλήματος. Με το σημαντικό αυτό θέμα θα ασχοληθούμε στο επόμενο κεφάλαιο.

- **Σχεδίαση αλγορίθμων** (algorithm design).

Η περιοχή αυτή είναι περισσότερο τέχνη παρά επιστήμη. Όπως σε κάθε τέχνη (επιστήμη), έχουν αναπτυχτεί βασικές τεχνικές σχεδίασης αποτελεσματικών αλγορίθμων. Οι πιο βασικές και ευρέως χρησιμοποιούμενες από αυτές είναι: η μέθοδος της **απληστίας** (greedy method), η μέθοδος του **διαίρει και βασίλευε** (divide and conquer method), η μέθοδος της **οπισθοδρόμησης** (backtracking method), και η μέθοδος του **δυναμικού προγραμματισμού** (dynamic programming method). Σε κάθε περίπτωση, ένας αλγόριθμος πρέπει να περιγραφεί με τη βοήθεια μίας γλώσσας, η οποία θα αποτελέσει και το περιεχόμενο του επόμενου κεφαλαίου.

1.5 ΜΙΑ ΑΛΓΟΡΙΘΜΙΚΗ ΓΛΩΣΣΑ

Η εισαγωγή της έννοιας του αλγορίθμου για την περιγραφή της διαδικασίας επίλυσης ενός προβλήματος με τη βοήθεια Η/Υ προϋποθέτει ότι κάθε στοιχείο που αφορά το πρόβλημα μπορεί να αναπαρασταθεί με πληρότητα και σαφήνεια στον Η/Υ. Αυτό καθίσταται δυνατό χάρη στη χρήση της δυαδικής αναπαράστασης της πληροφορίας, η οποία βασίζεται στην άλγεβρα Boole. Στη συνέχεια, θα πρέπει να είμαστε σίγουροι ότι κάθε αλγόριθμος μπορεί να εκφραστεί πλήρως σε μία γλώσσα αντιληπτή για έναν Η/Υ, κάτι που έχει αποδειχθεί από τον Alan Turing, ο οποίος εισήγαγε και εξήγησε την έννοια και τις λειτουργίες των μηχανών Turing[8].

Μετά από αυτά είναι προφανές ότι για να μπορέσει να κατανοηθεί, να μελετηθεί και γενικότερα να αξιοποιηθεί ένας αλγόριθμος, θα πρέπει να είναι διατυπωμένος σε μία γλώσσα, δηλαδή σε ένα σύνολο συμβολισμών, λεξιλογίου και συντακτικών κανόνων για μία ακριβή και μονοσήμαντη περιγραφή του. Η γλώσσα αυτή λέγεται αλγοριθμική.

Η επιλογή της αλγοριθμικής γλώσσας θα πρέπει να γίνεται έτσι ώστε να διευκολύνεται η κατανόηση των βασικών στοιχείων της δομής του αλγορίθμου,

[8] Οι μηχανές Turing είναι απλές, θεωρητικής φύσης υπολογιστικές «συσκευές» που προτάθηκαν από τον Alan Turing για να διερευνήσουν το εύρος και τους περιορισμούς θεμάτων επιλυσιμότητας και υπολογισιμότητας προβλημάτων με μηχανιστικό, δηλαδή αλγοριθμικό τρόπο.

ο έλεγχος της ορθότητας και του τερματισμού, καθώς και ο προσδιορισμός της υπολογιστικής πολυπλοκότητάς του. Μία πρώτη σκέψη θα ήταν το να γίνεται η περιγραφή ενός αλγορίθμου στη φυσική, καθομιλούμενη γλώσσα, όπως στα παραδείγματα 1.6 και 1.7. Η προσέγγιση αυτή όμως έχει τα εξής δύο βασικά μειονεκτήματα: Πρώτο, είναι ιδιαίτερα δύσκολο να περιγράψουμε έναν περίπλοκο αλγόριθμο με τον τρόπο αυτό, και είναι ακόμη δυσκολότερο να κατανοήσουμε τα βασικά στοιχεία της δομής του. Δεύτερο, δεν είναι καθόλου εύκολο να γράψουμε ένα πρόγραμμα Η/Υ από μία τέτοια περιγραφή του αλγορίθμου.

Μία επόμενη σκέψη θα ήταν να περιγραφεί ο αλγόριθμος σε μία γλώσσα προγραμματισμού όπως η FORTRAN, η BASIC, η PASCAL, η C++ κλπ. Ένα βασικό μειονέκτημα της προσέγγισης αυτής είναι ότι εξαρτά την περιγραφή του αλγορίθμου από λεπτομέρειες και ιδιαιτερότητες της επιλεγόμενης γλώσσας, γεγονός που περιορίζει την κατανόησή του μόνο από άτομα που γνωρίζουν τη γλώσσα αυτή. Ένα επίσης σημαντικό μειονέκτημα είναι ότι η χρησιμοποίηση μίας γλώσσας προγραμματισμού εισάγει πρόσθετη πολυπλοκότητα με αποτέλεσμα να χάνονται συχνά βασικά στοιχεία της λογικής και της δομής του αλγορίθμου, και να δυσχεραίνεται έτσι η μελέτη του.

Ένας βέλτιστος συνδυασμός των παραπάνω είναι μία **ψευδογλώσσα** ή **ψευδοκώδικας** (pseudocode) που διατηρεί τη δομή και τους συντακτικούς κανόνες μίας δομημένης γλώσσας προγραμματισμού χωρίς τις λεπτομέρειές της, μαζί με φράσεις από την καθομιλουμένη γλώσσα. Ο ψευδοκώδικας αυτός χρησιμοποιεί τη μεθοδολογία του δομημένου προγραμματισμού, όπως αυτή αναπτύχθηκε αρχικά από τους E.W. Dijkstra και N.Wirth, και επομένως μπορεί να μετατραπεί άμεσα και χωρίς καμία δυσκολία σε ένα πρόγραμμα μιας οποιασδήποτε γλώσσας προγραμματισμού.

Στην ψευδογλώσσα αυτή οι διάφορες πράξεις γράφονται υπό μορφή εντολών μια και τελικά ο υπολογιστής είναι αυτός που εντέλλεται να τις εκτελέσει. Ορισμένες από τις λέξεις της γλώσσας αυτής που έχουν αυστηρά καθορισμένο νόημα και τρόπο χρήσης ονομάζονται **δεσμευμένες λέξεις** (reserved words), και επειδή πρέπει να ξεχωρίζουν μέσα στο κείμενο του αλγορίθμου, έχει καθιερωθεί να γράφονται με έντονα γράμματα. Η πρώτη δεσμευμένη λέξη είναι η **αλγόριθμος.** Προκειμένου να αναγνωρίζεται κάθε αλγόριθμος, πρέπει να έχει ένα όνομα, που επιλέγεται συνήθως έτσι ώστε να υπενθυμίζει το αλγοριθμικό πρόβλημα. Το όνομα του αλγορίθμου γράφεται στην αρχή του κειμένου, και αμέσως μετά τη δεσμευμένη λέξη **αλγόριθμος**. Έπονται οι δεσμευμένες λέξεις

δεδομένα εισόδου, ή απλά **δεδομένα**, και **αποτελέσματα,** ακολουθούμενες από τα ονόματά τους.

Οι εντολές του αλγορίθμου που αποτελούν το σώμα του τοποθετούνται μεταξύ των δεσμευμένων λέξεων **αρχή** και **τέλος,** γράφονται διαδοχικά και χωρίζονται μεταξύ τους με το σύμβολο «;». Έτσι, η γενική μορφή ενός αλγορίθμου διατυπωμένη στην αλγοριθμική γλώσσα του προτεινόμενου ψευδοκώδικα έχει ως εξής:

```
αλγόριθμος: όνομα αλγορίθμου
δεδομένα
αποτελέσματα
αρχή
        εντολή 1
        εντολή 2
        ……..
        εντολή k
τέλος
```

Σώμα του αλγορίθμου

Πέρα από τις γνωστές πράξεις των μαθηματικών, για τις οποίες διατηρείται ο ίδιος συμβολισμός, χρησιμοποιείται η **εντολή εκχώρησης ή ανάθεσης** (assignment instruction), η οποία έχει την ακόλουθη συντακτική μορφή:

<όνομα μεταβλητής>:=<τιμή ή παράσταση>

(οτιδήποτε τοποθετείται ανάμεσα στα σύμβολα ανισότητας είναι υποχρεωτικό για τη σύνταξη της εντολής). Η σημασιολογία της εντολής αυτής έχει ως έξης: Υπολογίζεται πρώτα η τιμή της παράστασης και μετά το αποτέλεσμα εκχωρείται (ανατίθεται) στη μεταβλητή του αριστερού σκέλους του τελεστή εκχώρησης «:=», που πολλές φορές γράφεται και ως «←». Επειδή η τιμή της μεταβλητής στο αριστερό σκέλος του τελεστή αντικαθίσταται με το αποτέλεσμα υπολογισμού της παράστασης του δεξιού σκέλους, η εντολή αυτή ονομάζεται και **εντολή αντικατάστασης** (replacement statement).

Έτσι, η εντολή $x := x^2 + 1$ υλοποιείται ως εξής: Υπολογίζεται αρχικά η τιμή της παράσταση στο δεξιό μέρος του τελεστή εκχώρησης, υψώνοντας την παλαιά τιμή της x στο τετράγωνο, και προσθέτοντας 1. Στη συνέχεια, αντικαθίσταται η παλαιά τιμή της x με το αποτέλεσμα που προέκυψε. Θα πρέπει να τονιστεί ότι

το σύμβολο «:=» της εντολής εκχώρησης δεν είναι το γνωστό σύμβολο ισότητας των μαθηματικών. Χρησιμοποιείται στους αλγόριθμους για να αποθηκεύσει τις τιμές ενδιάμεσων μεγεθών που προκύπτουν κατά την εκτέλεση των υπολογισμών και τα οποία δεν είναι ούτε δεδομένα εισόδου αλλά ούτε και αποτελέσματα.

Παράδειγμα 1.9 Να κατασκευαστεί αλγόριθμος υπολογισμού της αριθμητικής τιμής της παράστασης $y = -3(2x-8)^2$

Στο πρόβλημα αυτό δεδομένα είναι η x και αποτέλεσμα η y, η οποία μπορεί να υπολογιστεί με τον ακόλουθο αλγόριθμο:

```
αλγόριθμος τιμή παράστασης
    δεδομένα x
    αποτελέσματα y
    αρχή
      A:=2*x (πολ/σίασε το 2 επί x και αποθήκευσε
      το αποτέλεσμα στην μεταβλητή Α)
      A:=A-8 (αφαίρεσε από την Α το 8 και αντικα-
      τέστησε την παλαιά τιμή της Α με το
      αποτέλεσμα που προέκυψε)
      A:=A*A (πολ/σίασε την Α με τον εαυτό της και
      αντικατέστησε την παλαιά τιμή με το αποτέ-
      λεσμα που προέκυψε)
      y:=(-3)*A (πολλαπλασίασε την Α επί -3 και
      αποθήκευσε το αποτέλεσμα στη μεταβλητή y)
    τέλος
```

◆

Οι εντολές ενός αλγορίθμου διακρίνονται σε **εντολές δράσης** (action statements) και **εντολές ελέγχου** (control statements). Με τις εντολές δράσης εκτελούνται κάποιες συγκεκριμένες πράξεις ή υπολογισμοί και παράγονται τα αντίστοιχα αποτελέσματα. Δεν υπάρχει κανένας περιορισμός ως προς το είδος των υπολογισμών ή των πράξεων που ζητείται να εκτελεστούν από μία εντολή δράσης. Έτσι, εντολές όπως οι παρακάτω, αποτελούν νόμιμες εντολές δράσης της ψευδογλώσσας:

- Υπολόγισε το εμβαδό ενός ελάσματος σχήματος τραπεζίου με βάσεις α και β και ύψος υ, και αποθήκευσέ το στη μεταβλητή Ε.

- Βρες το μεγαλύτερο ή μικρότερο στοιχείο ενός συνόλου αριθμών S που εκφράζει μετρήσεις θερμοκρασίας και διέγραψέ το.
- Αφαίρεσε μία κορυφή και όλες τις ακμές που συνδέονται με αυτή σε ένα γράφημα που αποδίδει εποπτικά ένα δίκτυο πωλήσεων.
- Βρες το συντομότερο μονοπάτι που συνδέει τις κορυφές x και y ενός γραφήματος.

Είναι προφανές ότι η εκτέλεση μιας εντολής δράσης μπορεί να αποτελέσει έναν ξεχωριστό αλγόριθμο. Γενικά, κάθε είδους εντολή καλώς ορισμένη και κατάλληλα εκφρασμένη, μπορεί να αποτελέσει μία νόμιμη εντολή της ψευδογλώσσας, αρκεί να είναι κατανοητή από τα άτομα στα οποία απευθύνεται. Στη γενικότητά της αυτή οφείλει την ευελιξία και την περιγραφική της δύναμη μία ψευδογλώσσα, έναντι μιας αντίστοιχης γλώσσας προγραμματισμού Η/Υ.

1.5.1 ΕΝΤΟΛΕΣ ΑΚΟΛΟΥΘΙΑΣ

Οι εντολές ενός οποιουδήποτε αλγορίθμου μπορούν να εκτελούνται **ακολουθιακά**, **επιλεγόμενα** (selectively) και **επαναληπτικά** (iteratively). Επιπρόσθετα, οι Boehm και Jacopini απέδειξαν το 1966[9] το εξής σημαντικό αποτέλεσμα:

Οι εντολές ακολουθίας, επιλογής, και επανάληψης, και μόνον αυτές, αρκούν για να περιγράψουμε κάθε αλγόριθμο που επιλύει ένα οποιοδήποτε αλγοριθμικό πρόβλημα.

Τις εντολές αυτές θα παρουσιάσουμε, παρακάτω, σε μορφή ψευδογλώσσας.

Υπάρχουν περιπτώσεις, όπου μία ενότητα εντολών ή και όλες οι εντολές ενός αλγορίθμου εκτελούνται ακολουθιακά, δηλαδή η μία μετά την άλλη, και μόνο μία φορά η κάθε μια, όπως για παράδειγμα συμβαίνει στην περίπτωση του υπολογισμού της τιμής μιας παράστασης όπως η $y=(\alpha x+\beta)(\gamma x+\delta)$. Στην περίπτωση αυτή η ενότητα των εντολών ονομάζεται **διαδικασία** (process) και αποτελεί βασική μονάδα της ψευδογλώσσας. Για να μην δημιουργείται καμία αμφιβολία σχετικά με την έναρξη και τον τερματισμό μιας διαδικασίας, χρησιμοποιούνται και σε αυτή οι δεσμευμένες λέξεις **αρχή** και **τέλος** που την οριοθετούν. Γράφουμε δηλαδή:

[9] Boehm, Corrado, and Jacopini, Giuseppe (1966), "Flow Diagrams, Turing Machines, and Languages with only Two Formation Rules", Communications of the ACM 9(5): 366-371.

```
αρχή
      εντολή 1;
      εντολή 2;
      .....
      εντολή k;
τέλος
```

Εναλλακτικά μπορούμε να γράψουμε

```
αρχή εντολή 1; εντολή 2;....εντολή k τέλος
ή
{ εντολή 1; εντολή 2;....εντολή k}
```

Έτσι για παράδειγμα ο υπολογισμός της τιμής της παράστασης $y = -3(2x-8)^2$ που γίνεται από τον αλγόριθμο του Παραδείγματος 1.9 είναι μία διαδικασία. Προφανώς αν μία διαδικασία περιέχει μία μόνο εντολή, δεν υπάρχει ανάγκη να χρησιμοποιηθούν οι λέξεις **αρχή, τέλος.**

1.5.2 ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ

Σε πολλές περιπτώσεις η εκτέλεση μιας διαδικασίας δεν είναι προκαθορισμένη, και η επιλογή της γίνεται μόνον όταν ικανοποιούνται κάποιες συνθήκες. Έτσι για παράδειγμα, μπορούμε να έχουμε την πρόταση «αν η ώρα είναι μεταξύ 14:30 και 15:30 χρησιμοποίησε την περιφερειακή οδό, αλλιώς χρησιμοποίησε την Εγνατία» Η πρόταση αυτή μπορεί να είναι μία εντολή ενός αλγορίθμου για τη συντομότερη μετάβαση από το ΑΠΘ στο αεροδρόμιο «Μακεδονία». Ελέγχοντας τη συνθήκη «η ώρα είναι μεταξύ 14:30 και 15:30» ο «εκτελών τις εντολές» αποφασίζει το επόμενό του βήμα (ποια οδό θα χρησιμοποιήσει). Η εντολή που χρησιμοποιείται για να περιγράψει αυτή τη διαδικασία ονομάζεται εντολή επιλογής (selection statement), αφού μέσω αυτής επιλέγεται η διαδικασία που θα εκτελεστεί, και έχει την εξής γενική μορφή:

αν(if)……**τότε**(then)….**αλλιώς** (else)

Η σύνταξη της εντολής ελέγχου είναι η ακόλουθη:

```
αν <συνθήκη>
 τότε διαδικασία Δ1
 αλλιώς διαδικασία Δ2
τέλος αν
```

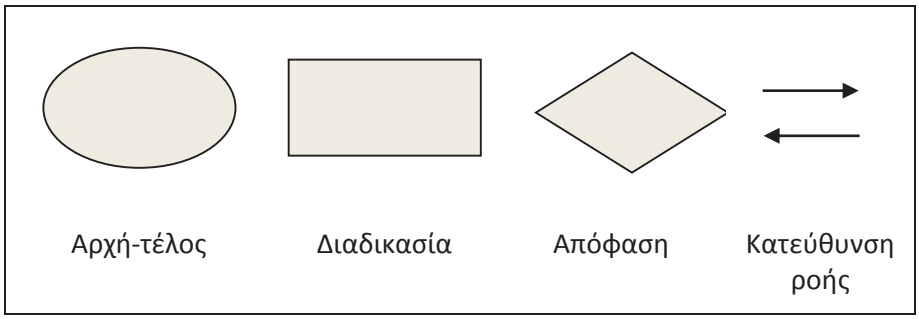
Η συνθήκη είναι το βασικό μέρος μιας εντολής επιλογής, αφού βάσει αυτής αποφασίζεται το σημείο στο οποίο μεταφέρεται η εκτέλεση του αλγορίθμου. Η συνθήκη εκφράζεται με λογικές προτάσεις, δηλαδή προτάσεις που μπορούν να πάρουν μία και μόνο από τις τιμές «αληθής» ή «ψευδής». Μία πρόταση είναι αληθής ή ψευδής ανάλογα με το εάν ικανοποιείται ή όχι η συνθήκη. Έτσι, υπολογίζεται πρώτα η τιμή της συνθήκης αντικαθιστώντας τις τρέχουσες τιμές των μεταβλητών που εμφανίζονται σε αυτή, και εάν βρεθεί αληθής, τότε εκτελείται μόνο η διαδικασία Δ1 και αγνοείται η Δ2. Αλλιώς δεν εκτελείται η Δ1 και εκτελείται μόνο η Δ2. Και στις δύο περιπτώσεις, η εκτέλεση του αλγορίθμου συνεχίζεται με την πρώτη εντολή που βρίσκεται μετά τη λέξη **τέλος αν**. Υπάρχουν περιπτώσεις, που το μέρος **αλλιώς** δεν υφίσταται οπότε εάν η συνθήκη είναι ψευδής, τότες δεν εκτελείται τίποτα. Έχουμε δηλαδή τη συντακτική μορφή:

```
αν <συνθήκη>
      τότε διαδικασία Δ1
τέλος αν
```

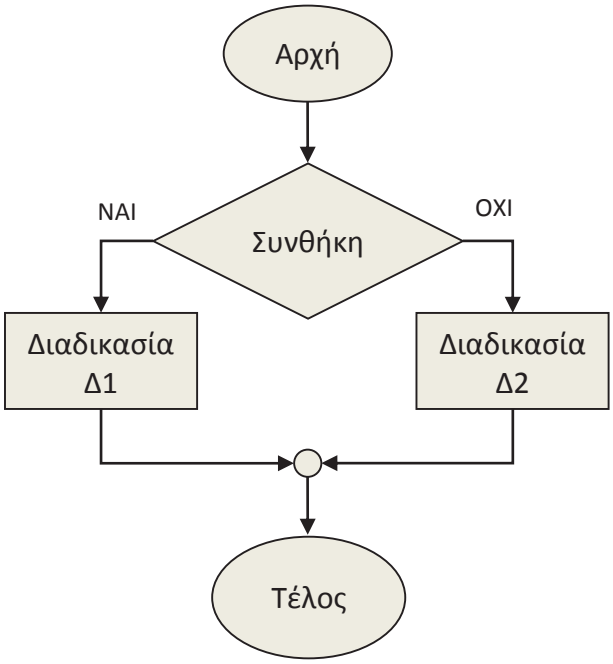
Η εντολή επιλογής ονομάζεται και **εντολή διακλάδωσης** (branching statement), αφού μέσω αυτής η εκτέλεση του αλγορίθμου (η ροή των πράξεων) διακλαδίζεται και μεταφέρεται σε άλλα τμήματα του αλγορίθμου. Λέγεται ακόμα και **εντολή απόφασης** (decision statement) αφού μέσω αυτής αποφασίζεται η διαδικασία που θα εκτελεστεί μετά.

1.5.2.1 Λογικά διαγράμματα

Λογικό διάγραμμα ή **διάγραμμα ροής** (flowchart) ενός αλγορίθμου ονομάζουμε ένα διάγραμμα που παρουσιάζει οπτικά τη δομή του αλγόριθμου και βοηθά στην καλύτερη κατανόηση της διαδικασίας επίλυσης του προβλήματος. Για το σκοπό αυτό έχει καθιερωθεί να χρησιμοποιούνται οι ακόλουθοι συμβολισμοί σχημάτων:



Έτσι, το τμήμα του διαγράμματος ροής ενός αλγορίθμου που αντιστοιχεί σε μία εντολή επιλογής έχει τη γενική μορφή:



Παράδειγμα 1.10. Για το παρακάτω τμήμα ενός αλγορίθμου, ποια είναι η τιμή της y για (i) $x = 5$ και (ii) $x = 3$

αν $(x \geq 5$ *ή* $x \leq 1)$

τότε$\{x \coloneqq x - 1; y \coloneqq x^2 + 1\}$

αλλιώς $\{x \coloneqq x + 1; y \coloneqq -3x \ \}$

τέλος αν

Λύση:

(i) όταν $x = 5$, η συνθήκη $x \geq 5$ ή $x \leq 1$ ικανοποιείται, οπότε εκτελείται η διαδικασία που ακολουθεί τη λέξη **τότε**. Έχουμε λοιπόν $x \coloneqq x - 1 = 4$ και $y = 4^2 + 1 = 17$.
(ii) αν $x = 3$, τότε δεν ικανοποιείται η συνθήκη, οπότε εκτελείται η διαδικασία που ακολουθεί τη λέξη **αλλιώς**. Έχουμε λοιπόν $x \coloneqq x + 1 = 4$ και $y = -3(4) = -12$

◆

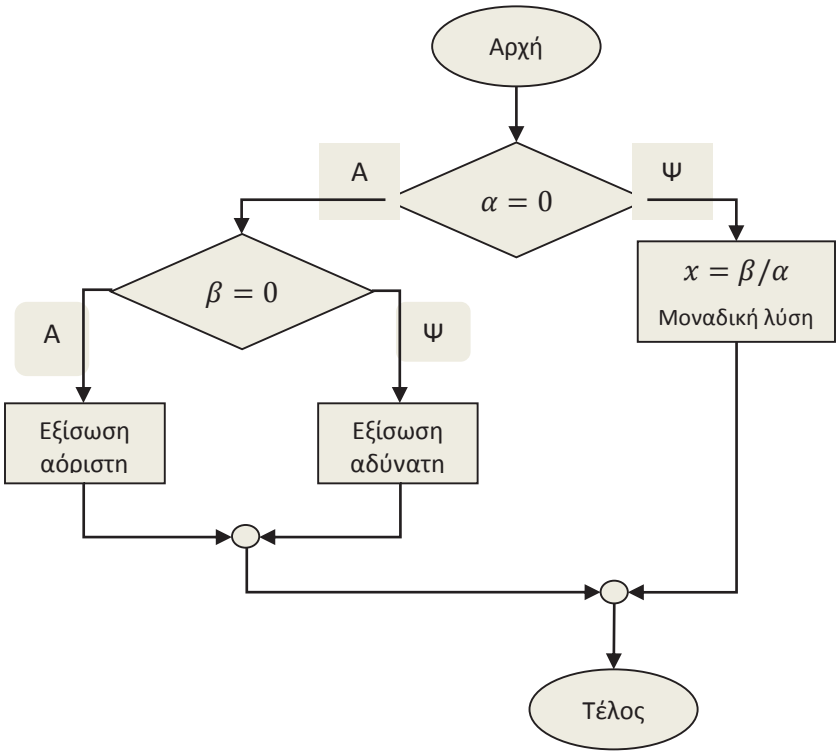
Παράδειγμα 1.11. Με τον αλγόριθμο του παραδείγματος 1.9 υπολογίσαμε την τιμή της παράστασης $y = -3(2x-8)^2$ με έναν αριθμό εντολών που εκτελέστηκαν ακολουθιακά. Έστω τώρα ότι έχουμε να υπολογίσουμε την τιμή της παράστασης $y = -3(2x-8)^2/(x-5)(x+12)$. Είναι προφανές ότι η παράσταση αυτή δεν ορίζεται για $x = 5$ και $x = -12$. Ο αλγόριθμος που ακολουθεί λαμβάνει υπόψη του το γεγονός αυτό και διακλαδώνει καταλλήλως την εκτέλεση των εντολών που υπολογίζουν την τιμή της παράστασης αυτής.

αλγόριθμος τιμή παράστασης
δεδομένα x
αποτελέσματα y
αρχή
  αν $x = 5$ ή $x = -12$
    τότε παράσταση δεν ορίζεται
    αλλιώς $y \coloneqq -3(2x-8)^2/(x-5)(x+12)$
  τέλος αν
τέλος

◆

1.5.2.2 Εμφωλευμενες δομες επιλογης

Οι διαδικασίες Δ1 και Δ2 σε μια εντολή επιλογής μπορεί να περιέχουν άλλες εντολές επιλογής. Στις περιπτώσεις αυτές λέμε πως έχουμε **εμφωλευμένες** (nested) εντολές επιλογής. Ένα απλό παράδειγμα τέτοιας δομής είναι η επίλυση της πρωτοβάθμιας εξίσωσης $ax = \beta$ (όπου α και β πραγματικοί αριθμοί), σύμφωνα με τον αλγόριθμο που έχει το ακόλουθο διάγραμμα ροής:



Ο αλγόριθμος αυτός περιγράφεται ως ακολούθως σε ψευδογλώσσα:

```
αλγόριθμος λύση πρωτοβάθμιας
δεδομένα α,β
αποτελέσματα x
    αρχή
        αν α=0
            τότε αν β=0
                τότε εξίσωση αόριστη (απειρία λύσεων)
```

```
            αλλιώς εξίσωση αδύνατη
        τέλος αν
    αλλιώς x:=β/α είναι η μοναδική λύση
    τέλος αν
τέλος
```

1.5.3 ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

Επαναλαμβάνω σημαίνει κάνω κάτι επαναληπτικά. Έτσι, σε έναν αλγόριθμο, **επανάληψη** (iteration) ονομάζεται η επαναλαμβανόμενη εκτέλεση κάποιου τμήματός του. Είναι προφανές ότι η τεράστια αξία των Η/Υ έγκειται στην ικανότητά τους να εκτελούν τις ίδιες ενότητες εντολών πολλές φορές. Οι εντολές με τις οποίες γίνεται αυτό πράξη ονομάζονται **εντολές επανάληψης** (iteration statements) ή **εντολές βρόχου** ή **εντολές ανακύκλησης** (loop statements). Μία ενότητα εντολών που εκτελείται επαναληπτικά ονομάζεται και **σώμα της ανακύκλησης**, Υπάρχουν περιπτώσεις όπου ο αριθμός των επαναλήψεων είναι γνωστός εκ των προτέρων, όπως συμβαίνει για παράδειγμα όταν θέλουμε να αθροίσουμε n αριθμούς, οπότε ο αριθμός που δηλώνει την επανάληψη της πράξης είναι προφανώς το n. Σε άλλες περιπτώσεις, ο αριθμός των επαναλήψεων είναι άγνωστος, και προσδιορίζεται από το πόσες φορές ικανοποιείται μία συνθήκη. Έστω για παράδειγμα ότι μας δίνεται ένας αριθμός Ν και ζητείται να βρούμε τον αριθμό k για τον οποίο ισχύει πως $2^k \leq N$ και $2^{k+1} > N$. Στην περίπτωση αυτή ξεκινούμε με $k = 0$, $A = 2$, και επαναλαμβάνουμε τις πράξεις $A \coloneqq 2A$ και $k \coloneqq k + 1$, έως ότου $A > N$. Οι δύο αυτές περιπτώσεις επαναλήψεων παρουσιάζονται αναλυτικά ακολούθως:

- **Αριθμός επαναλήψεων γνωστός**

Στην περίπτωση αυτή χρησιμοποιείται η ακόλουθη εντολή επανάληψης:

```
για(for) …έως (to) …. κάνε (do)
```

Η εντολή αυτή συντάσσεται ως εξής:

```
για (for) i:=m έως N, κάνε (do)
      ομάδα εντολών (σώμα επανάληψης)
τέλος για (end do)
```

Όπου i είναι ένας μετρητής που παίρνει τις ακέραιες τιμές m, m+1,….,N για κάθε μία από τις οποίες εκτελείται η ομάδα εντολών του σώματος της επανάληψης. Πολλές φορές ο μετρητής i ξεκινά με τιμή m και μεταβάλλεται με βήμα k, οπότε παίρνει τις τιμές m, m+k, m+2k,…,N. Στις περιπτώσεις αυτές η σύνταξη των εντολών έχει ως εξής:

```
για i:=m ανά k έως N, κάνε
        ομάδα εντολών (σώμα επανάληψης)
τέλος για
```

Παράδειγμα 1.12. Να γραφεί ψευδοκώδικας που να υπολογίζει το συνημίτονο της γωνίας $\boldsymbol{\theta}$ μεταξύ των διανυσμάτων $\boldsymbol{\alpha} =< \alpha_1, \alpha_2, \alpha_3, \ldots . \alpha_n >$ και $\boldsymbol{\beta} =< \beta_1, \beta_2, \beta_3, \ldots . \beta_n >$

αλγόριθμος υπολογισμός συνημιτόνου

δεδομένα διαν. $\boldsymbol{\alpha} =< \alpha_1, \alpha_2, \alpha_3, \ldots . \alpha_n >$ και $\boldsymbol{\beta} =< \beta_1, \beta_2, \beta_3, \ldots . \beta_n >$

αποτελέσματα συνημίτονο $cos\theta = \frac{\sum_{i=1}^{n} \alpha_i \beta_i}{\sqrt{(\sum \alpha_i^2)}\sqrt{(\sum \beta_i^2)}}$

Θα χρησιμοποιήσουμε τις μεταβλητές x = το εσ. γιν. **<α,β>**, y= τετρ. ρίζα του αθροίσματος των τετραγώνων των α_i και z = τετρ. ρίζα του αθροίσματος των τετραγώνων των β_i.

```
αρχή
        x:=0; y:=0,z:=0
                για i:=1 έως n κάνε
                        x := x + α_i * β_i
                        y := y + α_i^2
                        z := z + β_i^2
                τέλος για
        y:=sqrt(y);z=sqrt(z)
        cosθ:=x/(y*z)
τέλος
```

όπου sqrt είναι η συνάρτηση υπολογισμού τετραγωνικής ρίζας

◆

- **Αριθμός επαναλήψεων άγνωστος**

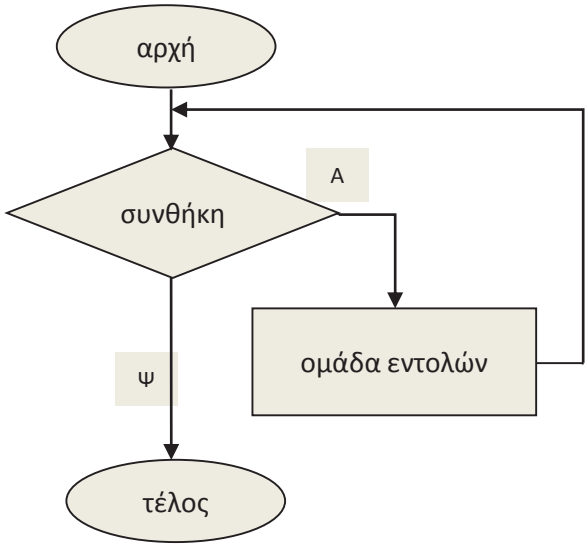
Στην περίπτωση αυτή το σώμα της επανάληψης εκτελείται μόνο όταν ισχύει μία συγκεκριμένη συνθήκη. Έτσι, ο αριθμός των επαναλήψεων είναι ίσος με τον αριθμό των περιπτώσεων για τις οποίες ικανοποιείται η συνθήκη. Αυτή η εντολή επανάληψης έχει την γενική μορφή

όσο(while)……**.κάνε** (do)

Και συντάσσεται ως εξής:

```
όσο <συνθήκη> κάνε
      ομάδα εντολών (σώμα επανάληψης)
τέλος όσο
```

Με τον τρόπο αυτό, στην αρχή κάθε επανάληψης ελέγχεται η συνθήκη, και εάν αυτή ισχύει, τότε εκτελείται το σώμα της επανάληψης. Σε διαφορετική περίπτωση, το σώμα της επανάληψης παρακάμπτεται, και η εκτέλεση του αλγορίθμου μεταφέρεται στην πρώτη εντολή αμέσως μετά τη λέξη **τέλος όσο.** Προφανώς, εάν η συνθήκη είναι από την αρχή ψευδής, το σώμα της επανάληψης δεν εκτελείται ποτέ. Στην αντίθετη περίπτωση, που η συνθήκη είναι πάντα αληθής, η ανακύκληση εκτελείται επ'άπειρον. Έπεται λοιπόν ότι για να υπάρξει τερματισμός της ανακύκλωσης, θα πρέπει το σώμα της επανάληψης να περιέχει εντολές με τις οποίες αναθεωρούνται οι τιμές των μεταβλητών που προσδιορίζουν την τιμή αληθείας της συνθήκης, έτσι ώστε σε κάποια ανακύκληση αυτή να πάρει την τιμή **ψευδής**. Το σχετικό διάγραμμα ροής παρατίθεται παραπλεύρως.



Παράδειγμα 1.13. Δίνεται ένας αριθμός N και ζητείται να βρούμε τον αριθμό k για τον οποίο ισχύει ότι: $2^k \leq N$ και $2^{k+1} > N$.

Ένας αλγόριθμος σε μορφή ψευδοκώδικα που επιλύει το πρόβλημα αυτό έχει ως εξής:

```
δεδομένα ακέραιος N
αποτέλεσμα ακέραιος k, τ.ώ. 2^k ≤ N και 2^(k+1) > N.
    αρχή
        k:=0, A:=2
        όσο A ≤ N κάνε
            A:=2 A
            k:=k+1
        τέλος όσο
    τέλος
```

◆

Παράδειγμα 1.14. Πολλαπλασιασμός ακεραίων.
Γνωρίζουμε ότι το γινόμενο δύο ακεραίων α και β ισούται με το άθροισμα του β, α φορές, είναι δηλαδή $\alpha\beta = \underbrace{\beta + \beta + \cdots \ldots \beta}_{\alpha\ \varphi o \rho \acute{\varepsilon} \varsigma}$.

Έτσι, το γινόμενο $\gamma = \alpha\beta$ προκύπτει ως αποτέλεσμα επαναληπτικής πρόσθεσης του β, α φορές, και αυτό μπορεί να γίνει με τον εξής αλγόριθμο σε μορφή ψευδοκώδικα:

```
αλγόριθμος πολλαπλασιασμός
δεδομένα ακέραιοι α,β
αποτελέσματα γινόμενο γ:=αβ
    αρχή
    γ:=0, k:=0
        όσο k<α κάνε
            γ:=γ+β
            k:=k+1
        τέλος όσο
    τέλος
```

◆

- **Εμφωλευμένες δομές επανάληψης**

Μία εντολή επανάληψης δύναται να περιλαμβάνει άλλες εντολές επανάληψης, έχουμε δηλαδή **εμφωλευμένες δομές επανάληψης** (embedded or nested loops). Μία τέτοια δομή παρουσιάζεται ακολούθως:

Παράδειγμα 1.15. Υπολογισμός του γινομένου $\boldsymbol{c = Ab}$ ενός m x n πίνακα $\boldsymbol{A} = [a_{i,j}]$ και ενός διανύσματος $\boldsymbol{b} = [b_1, b_2, b_3, \ldots\ldots, b_n]$. Στην περίπτωση αυτή ως γνωστό θα πρέπει να υπολογίσουμε επαναληπτικά m εσωτερικά γινόμενα διανυσμάτων, ένα για κάθε διάνυσμα γραμμή του πίνακα **A**, επί το διάνυσμα **b**. Καθένα όμως από αυτά, έστω το $\boldsymbol{\alpha_i} = (\alpha_{i1}, \alpha_{i2}, \ldots\ldots, \alpha_{in})$, υπολογίζεται από την εντολή επανάληψης που ακολουθεί, και κατόπιν φυλάσσεται στη θέση $\boldsymbol{i}$ του διανύσματος–στήλη $\boldsymbol{c}$:

```
S:=0
    για i:=1 έως n κάνε
        S:=S+A[i,j] b[i]
    τέλος για
C[i]:=S
```

Η εσωτερική αυτή εντολή επανάληψης ενθυλακώνεται εντός της εξωτερικής, οπότε η τελική μορφή του ψευδοκώδικα είναι η ακόλουθη:

αλγόριθμος πολλαπλασιασμός πινάκων

δεδομένα ένας mxn πίνακας $\boldsymbol{A} = [a_{i,j}]$, και ένα διάνυσμα $\boldsymbol{b} = [\beta_i]$

```
αποτελέσματα γινόμενο c = Ab
    αρχή
        για i:=1 έως m κάνε
            S:=0
            για i:=1 έως n κάνε
                S:=S+A[i,j]b[i]
            τέλος για
            C[i]:=S
        τέλος για
    τέλος
```

◆

Μέχρι τώρα εξετάσαμε τις βασικές υπολογιστικές δομές, της ακολουθίας, της επιλογής και της επανάληψης, που συνθέτουν κάθε αλγόριθμο, και παρουσιάσαμε τη λειτουργία τους με απλά παραδείγματα. Θυμίζουμε ότι, όπως έχει αποδεχθεί, οι τρείς αυτές δομές, και μόνον αυτές, αρκούν για να περιγράψουμε οποιονδήποτε αλγόριθμο. Έτσι, αν και έχουν αναπτυχθεί και άλλες δομές, για λόγους προγραμματιστικής ευκολίας, οι δομές αυτές είναι ισοδύναμες προς τις τρείς βασικές δομές. Παράδειγμα αποτελεί η δομή πολλαπλής επιλογής, που μπορεί ισοδύναμα να παρασταθεί με μία δομή εμφωλευμένων απλών δομών επιλογής. Τις δομές αυτές θα τις παρουσιάσουμε προγραμματιστικά σε γλώσσα FORTRAN και MATLAB στα επόμενα μέρη του βιβλίου αυτού.

Στο κεφάλαιο που ακολουθεί θα ασχοληθούμε με τον προσδιορισμό των υπολογιστικών απαιτήσεων σε χρόνο και χώρο μνήμης που συνεπάγεται η εκτέλεση ενός αλγορίθμου, κάτι που αποτελεί μία σημαντική περιοχή της αλγοριθμικής επιστήμης.