# Ενότητα 8

# ΑΛΓΟΡΙΘΜΙΚΗ

**Ανάλυση προβλήματος**

---

**Η έννοια του αλγορίθμου**

---

**Γλώσσες προγραμματισμού**

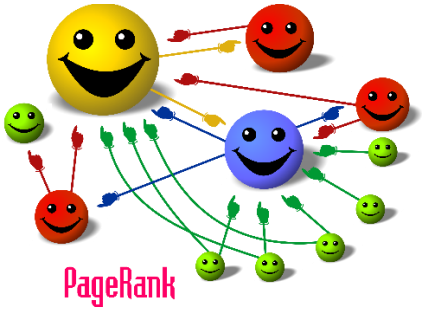
---

# Ενότητα 8. Αλγοριθμική

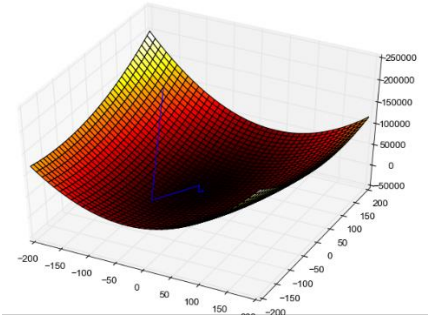
## 8.1 Εισαγωγή

Έχετε αναρωτηθεί ποτέ πώς μια μηχανή αναζήτησης κατατάσσει τα αποτελέσματα που επιστρέφει; Πώς το YouTube σας προτείνει συγκεκριμένα βίντεο, ή το Netflix συγκεκριμένες ταινίες; Πώς ο πλοηγός στο κινητό σας υπολογίζει τη συντομότερη διαδρομή από το σπίτι στο σχολείο; Όλες αυτές οι ερωτήσεις έχουν μια κοινή απάντηση: τον αλγόριθμο. Φανταστείτε τον αλγόριθμο σαν μια ακολουθία από βήματα για να φτάσετε στη λύση ενός προβλήματος. Η μελέτη των αλγορίθμων, γνωστή ως Αλγοριθμική, αποτελεί το θεμέλιο της επιστήμης της Πληροφορικής. Η επιστήμη της Πληροφορικής, αν και συνδέεται άμεσα με την ανάπτυξη των υπολογιστικών συστημάτων, έχει τις ρίζες της βαθιά στην αρχαιότητα. Πριν την εμφάνιση των ηλεκτρονικών υπολογιστών, μεγάλοι μαθηματικοί και φιλόσοφοι είχαν ήδη αναπτύξει αλγορίθμους που παραμένουν θεμελιώδεις στην επιστήμη των υπολογιστών. Ένα από τα πιο χαρακτηριστικά παραδείγματα είναι ο αλγόριθμος του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη (ΜΚΔ), που χρονολογείται περίπου στο 300 π.Χ. και εξακολουθεί να χρησιμοποιείται ευρέως λόγω της απλότητάς του και της αποδοτικότητάς του. Ένα άλλο σημαντικό παράδειγμα είναι το κόσκινο του Ερατοσθένη, που αναπτύχθηκε γύρω στο 200 π.Χ. και παρέχει έναν αποτελεσματικό τρόπο για την εύρεση όλων των πρώτων αριθμών μέχρι έναν δεδομένο αριθμό. Αυτοί τεκμηριώνουν ότι η αλγοριθμική σκέψη, που είναι η κεντρική έννοια της Πληροφορικής, αποτελεί αναπόσπαστο μέρος της ανθρώπινης διανόησης και προϋπήρχε πολύ πριν την εμφάνιση των πρώτων υπολογιστών.

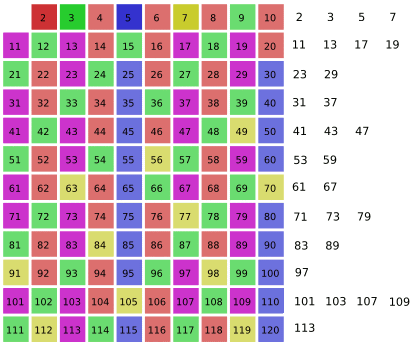
Σε αυτό το κεφάλαιο, θα εξερευνήσουμε τις βασικές έννοιες και αρχές που διέπουν την αλγοριθμική σκέψη. Θα ξεκινήσουμε με τον ορισμό του αλγορίθμου και θα προχωρήσουμε στη μελέτη των χαρακτηριστικών που τον καθιστούν αποδοτικό και χρήσιμο. Ένα από αυτά είναι ο τρόπος αναπαράστασής του. Για αυτόν τον λόγο επιλέγουμε μια γλώσσα προγραμματισμού. Σε αυτήν την ενότητα, θα δείτε πολλά παραδείγματα σύγχρονων γλωσσών προγραμματισμού.




**Εικόνα 8.1.** Ο αλγόριθμος PageRank με βάση τον οποίο κατατάσσει τις ιστοσελίδες η μηχανή αναζήτησης Google



**Εικόνα 8.2.** Ο αλγόριθμος Gradient Descent με τον οποίο μαθαίνουν τα νευρωνικά δίκτυα



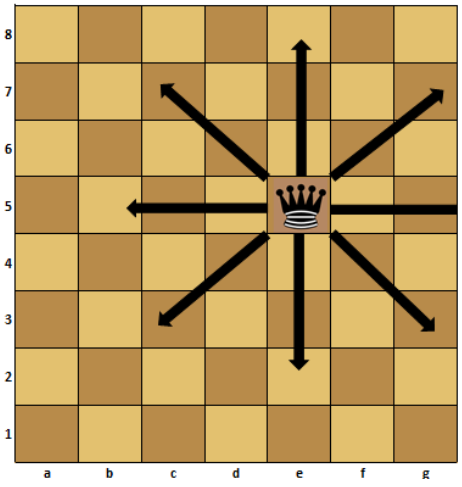
**Εικόνα 8.3.** Το κόσκινο του Ερατοσθένη για τον υπολογισμό των πρώτων αριθμών μέχρι το 120

## 8.2 Ανάλυση προβλήματος

Καθημερινά ερχόμαστε αντιμέτωποι με μια σειρά προβλημάτων διαφορετικής δομής και δυσκολίας. Υπάρχουν τα προβλήματα που καλούμαστε να λύσουμε στο πλαίσιο των μαθημάτων μας, όπως μια άσκηση στη Φυσική ή στα Μαθηματικά ή μια άσκηση γραμματικής στα Αρχαία Ελληνικά. Υπάρχουν, όμως, και τα προβλήματα που συναντάμε στην καθημερινότητά μας, όπως η εύρεση της συντομότερης διαδρομής από το σπίτι στο σχολείο, η οργάνωση του δωματίου μας ή η στελέχωση της ομάδας μπάσκετ του σχολείου. Εκτός από τα προβλήματα που αντιμετωπίζουμε στην καθημερινή μας ζωή, υπάρχουν και τα μεγάλα προβλήματα που αντιμετωπίζει η ανθρωπότητα, όπως το ενεργειακό πρόβλημα, οι πανδημίες ιών, οι κοινωνικές ανισότητες και πολλά άλλα. Τι εννοούμε όμως με την λέξη πρόβλημα;

**Πρόβλημα** είναι μια κατάσταση η οποία απαιτεί λύση που δεν είναι γνωστή.

Μια κατηγορία δύσκολων προβλημάτων είναι τα προβλήματα συνδυαστικής, τα οποία πρέπει να βρουν τον σωστό συνδυασμό που ικανοποιεί κάποια κριτήρια. Ένα πολύ γνωστό πρόβλημα αυτής της κατηγορίας, που μπορείτε να λύσετε με χαρτί και μολύβι, είναι το πρόβλημα των 8 βασιλισσών. Σύμφωνα με αυτό το πρόβλημα, πρέπει να τοποθετήσετε σε μια σκακιέρα 8 βασίλισσες χωρίς να απειλούνται μεταξύ τους. Δηλαδή, να μη βρεθούν δύο βασίλισσες στην ίδια γραμμή, στήλη ή διαγώνιο.



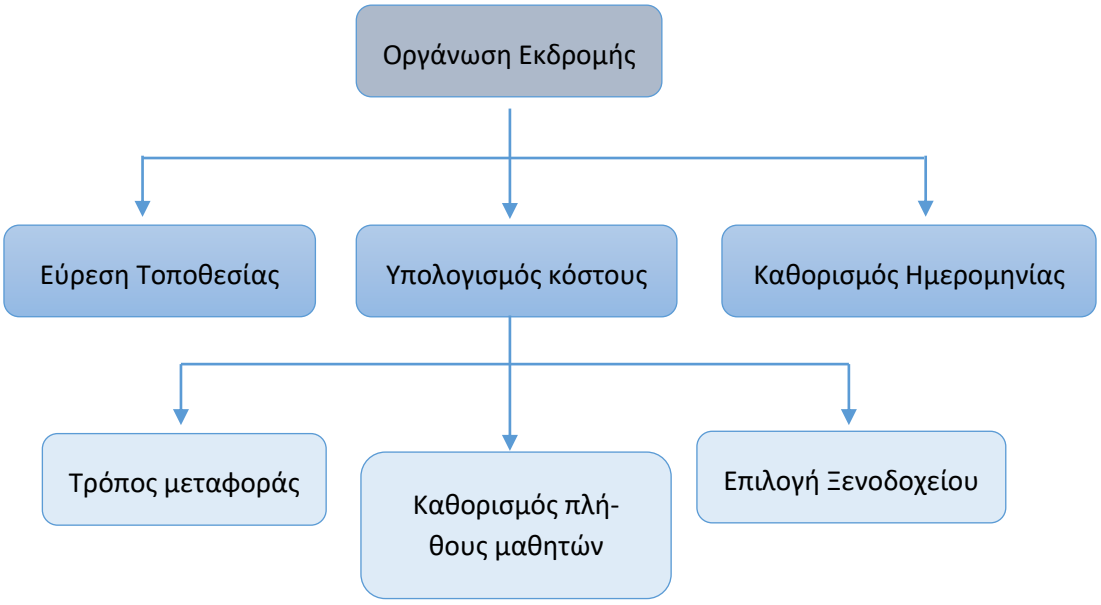

Δοκιμάστε να λύσετε το πρόβλημα στο τετράδιό σας. Στη συνέχεια, να συγκρίνετε τη λύση που βρήκατε με αυτές των συμμαθητών και συμμαθητριών σας. Τι παρατηρείτε;

Ας εξετάσουμε πρώτα ένα απλό και ευχάριστο πρόβλημα, όπως αυτό της οργάνωσης μιας σχολικής εκδρομής: Για να επιλύσουμε ένα πρόβλημα ξεκινάμε από την καταγραφή της ανάλυσης του προβλήματος και των βημάτων που απαιτούνται για την επίλυσή του.

Αρχικά, θέτουμε κάποια ερωτήματα στα οποία θα πρέπει να δοθούν απαντήσεις που είναι απαραίτητες για τον σχεδιασμό της εκδρομής, όπως :

- Πότε θα γίνει η εκδρομή;
- Πού θα πάμε;
- Πώς θα πάμε;
- Πόσο θα κοστίσει;

Για να απαντηθεί το τελευταίο ερώτημα, πρέπει πρώτα να απαντηθούν και άλλα ερωτήματα, όπως πόσοι μαθητές και μαθήτριες θα εκδηλώσουν ενδιαφέρον για την εκδρομή, σε ποιο ξενοδοχείο θα διαμείνουν και για πόσες μέρες. Αυτό είναι το στάδιο της **κατανόησης του προβλήματος**. Αφού απαντήσουμε στα παραπάνω ερωτήματα, θα πρέπει να καταστρώσουμε ένα σχέδιο. Θα πρέπει να θέσουμε τις ενέργειες που πρέπει να γίνουν σε μια σειρά. Σε αυτές τις περιπτώσεις, μπορεί να βοηθήσει ένας πίνακας ή ένα επεξηγηματικό διάγραμμα. Αυτό είναι το στάδιο του **σχεδιασμού της λύσης του προβλήματος**.




**Εικόνα 8.4.** Διαγραμματική αναπαράσταση της οργάνωση μιας σχολικής εκδρομής

Ακολουθεί η υλοποίηση του σχεδίου της λύσης του προβλήματος. Κατά την υλοποίηση υπάρχει πιθανότητα να εμφανιστούν διάφορα προβλήματα, όπως για παράδειγμα κάποιοι μαθητές και μαθήτριες να αλλάξουν γνώμη για την εκδρομή ή να κλείσει το ξενοδοχείο που έχουμε επιλέξει. Η επίλυση αυτών των προβλημάτων απαιτεί επανασχεδιασμό της λύσης.

Ένα από τα πιο γνωστά μοντέλα επίλυσης προβλήματος είναι το μοντέλο του Polya:

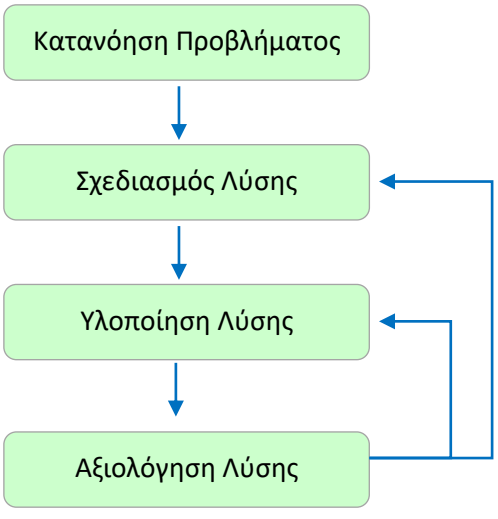
Σύμφωνα με το μοντέλο αυτό η διαδικασία της ανάλυσης και επίλυσης ενός προβλήματος μπορεί να δομηθεί σε τέσσερα βήματα:

**1)** Κατανόηση προβλήματος που περιλαμβάνει την καταγραφή των δεδομένων και των ζητούμενων.

**2)** Επινόηση - Δημιουργία ενός σχεδίου.

**3)** Εκτέλεση – Υλοποίηση του σχεδίου.

**4)** Αξιολόγηση της λύσης – Ανασκόπηση.




Στο τελευταίο στάδιο, γίνεται αναθεώρηση της αποτελεσματικότητας και της αποδοτικότητας της στρατηγικής επίλυσης που ακολουθήθηκε με σκοπό τη βελτίωση της λύσης. Για παράδειγμα, μπορεί να βρεθεί συντομότερος δρόμος για τη μετάβαση από το ξενοδοχείο στο γήπεδο ποδοσφαίρου, λόγω αυξημένης κίνησης στο αρχικό δρομολόγιο.

**Παράδειγμα 1**

Ένας βαρκάρης θέλει να περάσει ένα πρόβατο, έναν λύκο και ένα καφάσι με λάχανο στην απέναντι όχθη ενός ποταμού. Η βάρκα όμως είναι μικρή και μπορεί να μεταφέρει, εκτός από τον ίδιο, άλλο ένα από τα ζώα ή το καφάσι. Αν, όμως, μείνει ο λύκος μόνος του με το πρόβατο, πιθανόν να το φάει. Επίσης, το πρόβατο μπορεί να φάει το λάχανο, αν δεν υπάρχει ένας άνθρωπος να το σταματήσει.

Μπορείτε να δώσετε οδηγίες στον βαρκάρη για το πώς πρέπει να κάνει τη μεταφορά τους;
Μπορείτε να πειραματιστείτε με την προσομοίωση του προβλήματος στην παρακάτω εφαρμογή στο φωτόδεντρο: http://photodentro.edu.gr/v/item/ds/8521/760

**Απάντηση**

Αν πάρουμε τον λύκο, τότε το πρόβατο θα φάει το λάχανο, ενώ αν πάρουμε το λάχανο τότε ο λύκος θα φάει το πρόβατο. Η μόνη μας επιλογή είναι, στην αρχή να μεταφέρουμε το πρόβατο αφού ο λύκος μπορεί να μείνει μόνος του με το λάχανο. Τα πρώτα βήματα της λύσης του προβλήματος φαίνονται παρακάτω:

| | | |
|---|---|---|
| 1 |  | Βάλε το πρόβατο στη βάρκα |
| 2 |  | Πήγαινε στην απέναντι όχθη |
| 3 |  | Άφησε το πρόβατο στην όχθη |
| 4 |  | Πήγαινε στην αρχική όχθη |
| 5 |  | Βάλε τον λύκο στη βάρκα |
| 6 |  | Πήγαινε στην απέναντι όχθη |
| 7 |  | Άφησε το λύκο στην όχθη |
| 8 |  | Βάλε το πρόβατο στη βάρκα |

Μπορείτε να συμπληρώσετε τα παραπάνω βήματα, έτσι ώστε να οδηγούν στη λύση του προβλήματος;

Μελετώντας τη λύση του παραπάνω προβλήματος παρατηρούμε τα εξής δύο βασικά σημεία:

**1.** Η λύση του προβλήματος είναι μια ακολουθία από βήματα-οδηγίες.

**2.** Τα βήματα περιγράφονται με δύο διαφορετικούς τρόπους, δύο διαφορετικές αναπαραστάσεις. Και στις δύο περιπτώσεις υπάρχει ένα **σύνολο επιτρεπτών εντολών**. Το σύνολο αυτό περιλαμβάνει τρεις βασικές εντολές: { Άφησε, Πήγαινε, Βάλε}.

**Παράδειγμα 2**

Η Ηλέκτρα έχει στη διάθεσή της ένα μπιτόνι που χωράει ακριβώς 5 λίτρα, ένα μπιτόνι που χωράει ακριβώς 3 λίτρα και μια βρύση με άφθονο νερό. Η Ηλέκτρα μπορεί να γεμίσει ένα μπιτόνι με νερό από τη βρύση, να μεταφέρει το νερό από ένα μπιτόνι σε ένα άλλο, ή να αδειάσει όλο το νερό από ένα μπιτόνι. Πώς θα μετρήσει ακριβώς ένα λίτρο;

Για να περιγράψουμε πιο αποτελεσματικά τη λύση του προβλήματος σε βήματα, χρησιμοποιούμε τις εξής εντολές:

**Γέμισε**(N): Γεμίζει το μπιτόνι με χωρητικότητα N μέχρι πάνω

**Άδειασε**(N): Αδειάζει όλο το μπιτόνι με χωρητικότητα N.

**Μετακίνησε**(M, N): Αδειάζει το μπιτόνι με χωρητικότητα M σε αυτό με χωρητικότητα N, μέχρι το μπιτόνι με χωρητικότητα N να γεμίσει. Για παράδειγμα, η εντολή Μετακίνησε(5,3), αν υποθέσουμε ότι το μπιτόνι των 5 λίτρων είναι γεμάτο και το μπιτόνι των 3 λίτρων έχει 1 λίτρο, θα μεταφέρει 2 λίτρα στο τρίλιτρο και θα μείνουν 3 λίτρα στο μπιτόνι των 5 λίτρων.

| Εντολή | Νερό στο μπιτόνι | |
|---|---|---|
| | **5 lt** | **3 lt** |
| | 0 | 0 |
| Γέμισε(5) | 5 | 0 |
| Μετακίνησε(5,3) | 2 | 3 |
| Άδειασε(3) | 2 | 0 |
| Μετακίνησε (5,3) | 0 | 2 |
| Γέμισε(5) | 5 | 2 |
| Μετακίνησε (5,3) | 4 | 3 |
| Άδειασε(3) | 4 | 0 |
| Μετακίνησε (5,3) | 1 | 3 |

Και σε αυτήν την περίπτωση, η λύση του προβλήματος είναι μια ακολουθία από αυστηρά καθορισμένα και πεπερασμένα βήματα, 8 σε αυτήν την περίπτωση. Τέτοιες λύσεις ονομάζονται **αλγόριθμοι**. Οι αλγόριθμοι περιγράφονται σε κάποια γλώσσα προγραμματισμού. Στο παράδειγμα αυτό η γλώσσα προγραμματισμού έχει μόνο τρεις εντολές: {Γέμισε, Άδειασε, Μετακίνησε}. Αυτό είναι το σύνολο εντολών της.

**Δραστηριότητα 1**

Μετρήστε ακριβώς 6 λίτρα αν έχετε στην διάθεσή σας ένα δοχείο των 5 λίτρων, ένα των 7 λίτρων και μια πηγή με άφθονο νερό. Μπορείτε μόνο να γεμίζετε μέχρι πάνω και να αδειάζετε εντελώς τα δοχεία όσες φορές θέλετε.

**Δραστηριότητα 2**

Έχετε τρεις κανάτες, μία των 10 λίτρων, μία των 7 λίτρων και μία των 3 λίτρων. Αυτή που χωράει 10 λίτρα είναι γεμάτη και οι άλλες δύο άδειες. Πώς μπορούμε να βάλουμε σε μία από τις κανάτες ακριβώς 5 λίτρα νερό, χωρίς ζυγαριά, κάνοντας μόνο μεταφορές νερού από τη μία στην άλλη;

## 8.3 Εισαγωγή στην έννοια του αλγορίθμου

Στην προηγούμενη παράγραφο, μελετήσαμε την επίλυση προβλημάτων τα οποία επιδέχονται αλγοριθμική λύση. Δηλαδή, η λύση τους δεν είναι ένας αριθμός ή μία ποσότητα αλλά μια ακολουθία από βήματα τα οποία οδηγούν σε μια συγκεκριμένη κατάσταση. Η ακολουθία αυτή ονομάζεται **αλγόριθμος**.

Ο **αλγόριθμος (algorithm)** είναι μια ακολουθία από αυστηρά καθορισμένα βήματα που είναι εκτελέσιμα σε πεπερασμένο χρόνο και έχουν στόχο την επίλυση ενός προβλήματος.

Οι άνθρωποι χρησιμοποιούν καθημερινά διάφορους αλγορίθμους για τη διεκπεραίωση καθημερινών εργασιών. Ένας από τους πρώτους αλγορίθμους που μαθαίνουμε είναι ο αλγόριθμος της πρόσθεσης. Ένα παράδειγμα δίνεται παρακάτω:

```
                1                1                1
  1 9 9 9     1 9 9 9        1 9 9 9        1 9 9 9
+       3 9 + 0 0 3 9      + 0 0 3 9      + 0 0 3 9
-----------  -----------    -----------    -----------
              0 0 0 8        0 0 3 8        2 0 3 8
```

Τα βήματα του αλγορίθμου εκτελούνται με συγκεκριμένη σειρά. Πρώτα προσθέτουμε τις μονάδες, στη συνέχεια προχωράμε στις δεκάδες κ.ο.κ.

**Δραστηριότητα 3**

Να καταγράψετε τα βήματα του αλγορίθμου της πρόσθεσης αναλυτικά.

Ένα άλλο κλασσικό παράδειγμα αλγορίθμων αποτελούν οι συνταγές μαγειρικής. Για να μαγειρέψουμε ένα φαγητό, ακολουθούμε τις οδηγίες μιας συνταγής μαγειρικής αυστηρά, αλλιώς το αποτέλεσμα δε θα είναι το επιθυμητό. Εδώ, εκτός από τη σειρά με την οποία θα πρέπει να εκτελεστούν τα βήματα του αλγορίθμου, σημαντικό ρόλο παίζει και η ακρίβεια με την οποία θα εκτελεστούν οι εντολές. Η παραμικρή παρέκκλιση μπορεί να προκαλέσει σοβαρά προβλήματα στο τελικό αποτέλεσμα. Για παράδειγμα, αν ρίξουμε λίγο αλάτι ή πιπέρι παραπάνω, το αποτέλεσμα μπορεί να είναι πολύ διαφορετικό από το αναμενόμενο.

**Δραστηριότητα 4**

Να βάλετε στη σειρά τα βήματα εκτέλεσης μιας απλής συνταγής για μακαρόνια με σάλτσα.

1. Σερβίρετε τα μακαρόνια με σάλτσα ντομάτας και από πάνω ρίχνετε πιπέρι
2. Αφού στραγγίσετε τα μακαρόνια, τα βάζετε πίσω στην κατσαρόλα με λάδι και βούτυρο και ανακατεύετε.
3. Βράζετε τα μακαρόνια για 8 λεπτά.
4. Περιμένετε μέχρι να βράσει το νερό.
5. Κλείστε το μάτι της κουζίνας.
6. Ανοίξτε το μάτι της κουζίνας.

7. Βάλτε τα μακαρόνια στην κατσαρόλα.
8. Στραγγίστε τα μακαρόνια και στη συνέχεια σερβίρετε στα πιάτα.
9. Γεμίστε την κατσαρόλα με νερό από τη βρύση.
10. Τοποθετήστε την κατσαρόλα στο μάτι της κουζίνας.

Κάθε αλγόριθμος καθορίζεται από πέντε 5 χαρακτηριστικά :

- **Καθοριστικότητα**: Κάθε βήμα πρέπει να είναι αυστηρά καθορισμένο και εκτελέσιμο σε κάθε περίπτωση.
- **Περατότητα**: Ο αλγόριθμος πρέπει να τερματίζει μετά από την εκτέλεση πεπερασμένου πλήθους βημάτων.
- **Είσοδος**: Ένας αλγόριθμος μπορεί να έχει καμία, μία ή περισσότερες εισόδους, οι οποίες αντιστοιχούν στα δεδομένα που πρέπει να χρησιμοποιήσει για να επιλύσει το πρόβλημα.
- **Έξοδος**: Ένας αλγόριθμος, πρέπει να έχει τουλάχιστον μία έξοδο, η οποία αντιστοιχεί στη λύση του προβλήματος.
- **Αποτελεσματικότητα**: Κάθε βήμα του αλγορίθμου πρέπει να είναι αρκετά απλό, ώστε να μπορεί να εκτελεστεί από την υπολογιστική μηχανή για την οποία απευθύνεται. Για παράδειγμα, αν ο αλγόριθμος απευθύνεται σε ανθρώπους, θα πρέπει να μπορεί να εκτελεστεί με χαρτί και μολύβι από έναν άνθρωπο ακολουθώντας απλές και κατανοητές εντολές.


Η λέξη **αλγόριθμος** προέρχεται από το όνομα ενός Πέρση μαθηματικού, Abu Ja'far Mohammed ibn Musa al Khowarizmi, που έζησε τον 9ο αιώνα μ.Χ. Η λέξη «al Khowarizmi» σημαίνει «από το Khowarazm», που είναι η σημερινή πόλη Khiva του Ουζμπεκιστάν.

Ο al Khowarizmi έγραψε μια αραβική γλωσσική πραγματεία στο ινδουιστικό-αραβικό αριθμητικό σύστημα, το οποίο μεταφράστηκε στα λατινικά κατά τη διάρκεια του 12ου αιώνα. Το χειρόγραφο ξεκινά με τη φράση Dixit Algorizmi («Έτσι σπάσε τον Αλ-Κουαρίζμι»), όπου «Αλγκορίζμι» ήταν η λατινοποίηση του ονόματος του Αλ-Κουαρίζμι από τον μεταφραστή.

Μια διαδικασία, η οποία δεν πληροί το κριτήριο της περατότητας, ονομάζεται *υπολογιστική διαδικασία* (computational process). Σήμερα, σχεδόν όλες οι εφαρμογές που χρησιμοποιούμε δεν τερματίζουν ποτέ, αφού πρέπει να είναι διαθέσιμες να δεχτούν τα αιτήματα των χρηστών τους ανά πάσα στιγμή, όπως για παράδειγμα, είναι οι μηχανές αναζήτησης, οι εφαρμογές ηλεκτρονικού ταχυδρομείου, οι εφαρμογές νέφους, οι εφαρμογές μέσων κοινωνικής δικτύωσης κ.λπ. Όλες αυτές οι εφαρμογές ανήκουν σε μια ειδική κατηγορία υπολογιστικών διαδικασιών γνωστών ως reactive processes, διότι αναμένουν το επόμενο αίτημα προς διεκπεραίωση από τον χρήστη. Ωστόσο, υπάρχουν και περιπτώσεις που, για κάποιον λόγο, ένα πρόγραμμα εγκλωβίζεται σε μια μη αναμενόμενη ατέρμονη διαδικασία, εξαιτίας κάποιου σφάλματος στην κωδικοποίηση του αλγορίθμου.

**Αλγόριθμοι κρυπτογράφησης**

Την εποχή των γαλατικών πολέμων, ο Ιούλιος Καίσαρας έγραφε στον Κικέρωνα και σε άλλους φίλους του, αντικαθιστώντας τα γράμματα του κειμένου, με γράμματα που βρίσκονται έναν αριθμό θέσεων μπροστά στο Λατινικό Αλφάβητο, σε περίπτωση που το μήνυμα έπεφτε στα χέρια του εχθρού. Ένας από τους γνωστούς και απλούς τύπους κρυπτογραφικής υποκατάστασης που χρησιμοποιούσε ήταν η μετάθεση κατά τρεις θέσεις μπροστά στο αλφάβητο.


Στην κρυπτογραφία χρησιμοποιούνται συνήθως οι όροι *κανονικό αλφάβητο* για να αναφερθούν στο αλφάβητο στο οποίο έχει γραφεί το αρχικό μήνυμα και *κρυπτογραφικό αλφάβητο* που αναφέρεται στο κρυπτογραφημένο μήνυμα. Για εύκολη κρυπτογράφηση/αποκρυπτογράφηση τοποθετούμε το κανονικό πάνω από το κρυπτογραφικό αλφάβητο, ώστε να φαίνεται πως κρυπτογραφείται κάθε γράμμα, όπως φαίνεται στο παρακάτω σχήμα.

| Α | Β | Γ | Δ | Ε | Ζ | Η | Θ | Ι | Κ | Λ | ..... | Χ | Ψ | Ω |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⬇ | | ⬇ | | ⬇ | | ⬇ | | ⬇ | | ⬇ | | | ⬇ | |
| Χ | Ψ | Ω | Α | Β | Γ | Δ | Ε | Ζ | Η | Θ | ..... | Τ | Υ | Φ |

Ο αριθμός των θέσεων που μετακινείται (ολισθαίνει) το αλφάβητο είναι το μυστικό κλειδί που χρειαζόμαστε για να αποκρυπτογραφήσουμε το μήνυμα που θα λάβουμε. Πώς όμως θα κινηθούν οι κρυπταναλυτές του εχθρού, αν πέσει ένα τέτοιο μήνυμα στα χέρια τους χωρίς να ξέρουν το μυστικό κλειδί; Τι μπορεί να κάνει ο Καίσαρας για να κάνει δύσκολο το έργο τους;

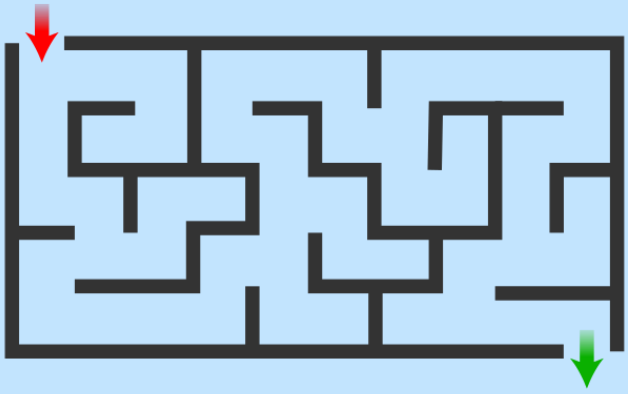
**Δραστηριότητα 5 - Κρυπτανάλυση**

Χωριστείτε σε ομάδες των 3-4 ατόμων στην τάξη. Επιλέξτε το μυστικό κλειδί της κρυπτογράφησης και κρυπτογραφήστε με αυτό μια φράση. Στη συνέχεια, να γράψετε στον πίνακα το κρυπτογραφημένο μήνυμα. Ο στόχος είναι να βρείτε τα κλειδιά των άλλων ομάδων και να αποκρυπτογραφήσετε τα μηνύματά τους.
Να καταγράψετε αναλυτικά την στρατηγική που θα ακολουθήσετε κατά την κρυπτογράφηση του δικού σας μηνύματος, αλλά και κατά τη διαδικασία της κρυπτανάλυσης των μηνυμάτων των άλλων ομάδων.
Πόσο χρόνο πιστεύετε ότι χρειάζεται ένας απλός υπολογιστής για να «σπάσει» των κώδικα του Καίσαρα; Υπάρχει τρόπος να τον δυσκολέψετε;

### Δραστηριότητα 6 - Έξοδος από Λαβύρινθο

Να περιγράψετε έναν αλγόριθμο για την έξοδο από τον παρακάτω λαβύρινθο, αν υποθέσουμε ότι το κόκκινο βέλος δείχνει την είσοδο και το πράσινο βέλος δείχνει την έξοδο.

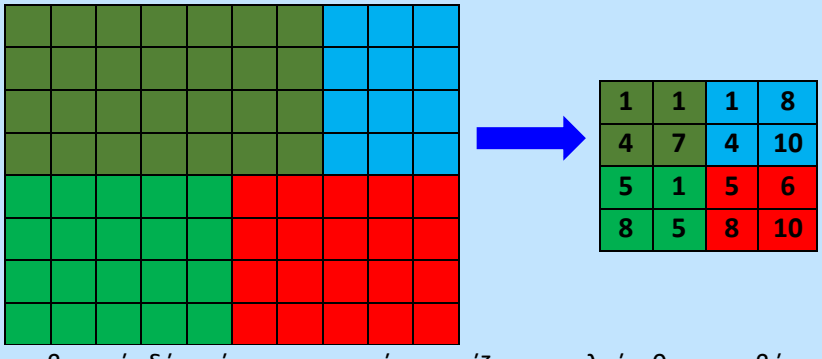
Μια παλιά στρατηγική που οδηγεί, σχεδόν πάντα, στην έξοδο του λαβυρίνθου είναι ο κανόνας του δεξιού χεριού. Τοποθετούμε το δεξί μας χέρι στον τοίχο, προχωράμε και δεν το αφήνουμε ποτέ. Με αυτόν τον τρόπο, σε κάθε στροφή θα πηγαίνουμε δεξιά. Έτσι, δε θα περάσουμε ποτέ δύο φορές από το ίδιο σημείο και αν υπάρχει έξοδος θα τη βρούμε σίγουρα. Η μόνη προϋπόθεση για να λειτουργήσει αυτός ο κανόνας είναι να μην υπάρχουν κενά στον τοίχο.



### Παράδειγμα 3 - Αλγόριθμοι Συμπίεσης Δεδομένων

Στην κάρτα μνήμης του κινητού ή στον δίσκο του υπολογιστή μας αποθηκεύουμε φωτογραφίες, βίντεο, ηλεκτρονικά βιβλία και άλλες πληροφορίες. Πολλές φορές, όμως, ο αποθηκευτικός χώρος δεν είναι αρκετός. Για αυτό καταφεύγουμε στη συμπίεση μεγάλων αρχείων, έτσι ώστε να εξοικονομηθεί χώρος. Η συμπίεση δεδομένων (data compression) ορίζεται ως «η μετατροπή (μετασχηματισμός) ενός ψηφιακού αρχείου σε αρχείο μικρότερου μεγέθους με τρόπο ώστε να είναι δυνατή η αναδόμηση του συμπιεσμένου αρχείου στο αρχικό». Η διαδικασία της συμπίεσης εφαρμόζεται συστηματικά στα υπολογιστικά συστήματα που χρησιμοποιούν και επεξεργάζονται μεγάλο όγκο ψηφιακών δεδομένων. Καθημερινά χρησιμοποιούμε συμπιεσμένα αρχεία πολυμέσων, όπως είναι τα μουσικά αρχεία mp3, τα βίντεο mp4 και οι εικόνες jpeg.

Ένα πολύ απλό παράδειγμα συμπίεσης εικόνας φαίνεται στο παρακάτω σχήμα.



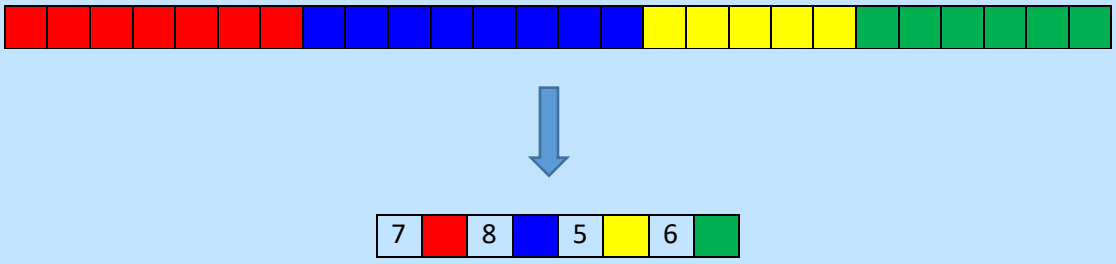

Να περιγράψετε τη βασική ιδέα πάνω στην οποία στηρίζεται ο αλγόριθμος με βάση τον οποίο έχει συμπιεστεί η παραπάνω εικόνα. Είναι αυτός ο αλγόριθμος αποτελεσματικός για όλες τις εικόνες;

*Προσπαθήστε να σκεφτείτε την απάντηση πριν κοιτάξετε στην επόμενη σελίδα*

**Απάντηση**

Υπάρχουν πολλοί αλγόριθμοι για να καταφέρουμε να μειώσουμε το μέγεθος μίας ακολουθίας. Ένας αλγόριθμος συμπίεσης μπορεί να επιφέρει μεγάλη ελάττωση του μήκους σε ακολουθίες δεδομένων με κάποιο ιδιαίτερο χαρακτηριστικό, ενώ ο ίδιος να είναι αναποτελεσματικός για άλλες ακολουθίες, αφήνοντάς τις στο ίδιο μήκος ή ακόμα και μεγαλώνοντάς το σε μερικές περιπτώσεις.

Ο *αλγόριθμος RLE (Run Length Encoding)* αποτελεί έναν από τους απλούστερους αλγορίθμους συμπίεσης δεδομένων. Σύμφωνα με τη μέθοδο αυτή, διατρέχεται η ακολουθία των συμβόλων που αποτελούν τα δεδομένα προς συμπίεση και καταγράφονται οι διαδοχικές επαναλήψεις του ίδιου συμβόλου. Στη συνέχεια, αντικαθίστανται οι συνεχόμενες επαναλήψεις με το πλήθος τους, ακολουθούμενο από το σύμβολο, όπως φαίνεται στο παρακάτω παράδειγμα:




Όταν αναφερόμαστε σε σύμβολο εννοούμε αριθμό, γράμμα ή σημείο στίξεως. Πολλές φορές αναφερόμαστε σε αυτό και σαν χαρακτήρα.

Ο παραπάνω αλγόριθμος συμπίεσης αποδίδει αρκετά καλά, όταν έχουμε συχνές επαναλήψεις χαρακτήρων. Για παράδειγμα, η ακολουθία χαρακτήρων

AAAAAAAAAAAAA BBBBBBBBBBBBBBBBBBBBBBBBBBBB ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ

**13Α** **30Β** **20Δ**

με μήκος 63 χαρακτήρες σε συμπιεσμένη μορφή χρειάζεται μόλις 9 χαρακτήρες.

Αν όμως δεν έχουμε επαναλαμβανόμενα σύμβολα, τότε με αυτήν τη μέθοδο συμπίεσης υπάρχει πιθανότητα να αυξηθεί το μέγεθος του συμπιεσμένου αρχείου αντί να μειωθεί.

## 8.4 Αναπαράσταση αλγορίθμων – Γλώσσες προγραμματισμού

Η έννοια του αλγόριθμου αποτελεί το βασικό θεμέλιο της επιστήμης της Πληροφορικής και αποδεικνύει ότι η επιστήμη της Πληροφορικής υπήρχε σε λανθάνουσα μορφή πολύ πριν εμφανιστούν οι υπολογιστές. Ένας από τους πιο γνωστούς αλγορίθμους είναι ο αλγόριθμος του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη δύο θετικών ακέραιων αριθμών, τον οποίο περιέγραψε ο Ευκλείδης σε ένα από τα βιβλία των Στοιχείων τον 3ο αιώνα π.Χ. Θυμηθείτε ότι ο μέγιστος κοινός διαιρέτης δύο αριθμών είναι ο μεγαλύτερος ακέραιος που διαίρει ακριβώς και τους δύο αριθμούς.

Ο Ευκλείδης σκέφτηκε έναν πολύ γρήγορο αλγόριθμο για την εύρεση του ΜΚΔ. Η βασική ιδέα είναι, ότι ο μέγιστος κοινός διαιρέτης δύο θετικών ακέραιων αριθμών **x** και **y** είναι ο ίδιος με τον μέγιστο κοινό διαιρέτη του **y** και του **x mod y**, όπου **mod** είναι ο τελεστής του υπολοίπου της ακέραιας διαίρεσης του **x** με το **y**. Δηλαδή, ο αλγόριθμος βασίζεται στην απλή παρατήρηση ότι:

**ΜΚΔ**( x, y ) = **ΜΚΔ**( y, x **mod** y )

Με διαδοχικές εφαρμογές της παραπάνω σχέσης έχουμε:

ΜΚΔ(512, 120) = ΜΚΔ(120, 32) = ΜΚΔ(32, 24) = ΜΚΔ(24, 8) = 8

| **x** | | **y** | | **υπόλοιπο** | **πηλίκο** | |
|---|---|---|---|---|---|---|
| 512 | : | 120 | = | 32 | 4 | 512 = 120 x 4 + 32 |
| 120 | : | 32 | = | 24 | 3 | 120 = 32 x 3 + 24 |
| 32 | : | 24 | = | 8 | 1 | 32 = 24 x 1 + 8 |
| 24 | : | **8** | = | 0 | 3 | 24 = 8 x 3 + 0 |

Ένας αλγόριθμος μπορεί να αναπαρασταθεί με διάφορους τρόπους. Κάποιοι από αυτούς είναι πιο κοντά στον ανθρώπινο τρόπο σκέψης και άλλοι πιο κοντά στον τρόπο λειτουργίας του υπολογιστή. Παρακάτω δίνονται τρεις εναλλακτικοί τρόποι αναπαράστασης του αλγορίθμου του Ευκλείδη σε φυσική γλώσσα, ψευδοκώδικα και διάγραμμα ροής.

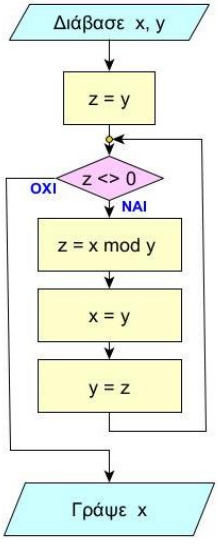
**Φυσική γλώσσα**

**1** ΔΙΑΒΑΣΕ x, y
**2** ΘΕΣΕ z = y
**3** ΑΝ z = 0 ΤΟΤΕ
ΠΗΓΑΙΝΕ ΣΤΗΝ 8
**4** ΘΕΣΕ z ← x mod y
**5** ΘΕΣΕ x = y
**6** ΘΕΣΕ y = z
**7** ΠΗΓΑΙΝΕ ΣΤΗΝ 3
**8** ΓΡΑΨΕ x

**Ψευδοκώδικας**

Διάβασε x, y
z ← y
Όσο z <> 0 Επανάλαβε
z ← x mod y
x ← y
y ← z
Τέλος_Επανάληψης
Γράψε x

**Διάγραμμα Ροής**



Ο βασικός σκοπός της σχεδίασης ενός αλγορίθμου είναι να μπορεί να εκτελεστεί από έναν υπολογιστή, ο οποίος μπορεί να εκτελέσει δισεκατομμύρια πράξεις το δευτερόλεπτο. Για να γίνει αυτό, κάθε εντολή θα πρέπει να είναι αυστηρά καθορισμένη και εκτελέσιμη από τον υπολογιστή. Δηλαδή, να αναγνωρίζει ο υπολογιστής κάθε εντολή και να ξέρει πώς θα την εκτελέσει (κριτήριο αποτελεσματικότητας). Για τον σκοπό αυτό, χρησιμοποιούμε διάφορες γλώσσες προγραμματισμού, όπως φαίνεται παρακάτω:

**C++**

```
int z = y;
while (z != 0 )
{
   z = x % y;
   x = y;
   y = z ;
}
cout << x;
```

**Javascript**

```
var z = y;
while (z != 0 ) {
   z = x % y;
   x = y;
   y = z;
}
document.write
(x);
```

**Python**

```
z = y
while z != 0 :
   z = x % y
   x = y
   y = z

print( x )
```

**Lua**

```
while y ~= 0 :
 x , y = y, x % y

print( x )
```

**Scratch**


Μεταξύ της περιγραφής ενός αλγορίθμου σε φυσική γλώσσα και της περιγραφής σε μια γλώσσα προγραμματισμού, υπάρχει μια ενδιάμεση αναπαράσταση του αλγορίθμου σε μια μορφή γνωστή ως **ψευδοκώδικας** ή **ψευδογλώσσα**, η οποία λειτουργεί ως γέφυρα μεταξύ των δύο αναπαραστάσεων, όπως φαίνεται δίπλα. Παρατηρήστε πόσο μοιάζει η αναπαράσταση του αλγορίθμου σε Scratch με την αναπαράσταση σε ψευδογλώσσα. Μπορείτε να αντιστοιχήσετε τις εντολές σε ψευδογλώσσα με αυτές σε Scratch;

```
z ← y
Όσο z ≠ 0 Επανάλαβε :
   z ← x mod y
   x ← y
   y ← z

print( x )
```

Ο ψευδοκώδικας χρησιμοποιείται όταν θέλουμε να περιγράψουμε σύνθετους και πολύπλοκους αλγορίθμους, οι οποίοι σε κάποια γλώσσα προγραμματισμού δε θα ήταν εύκολα κατανοητοί, λόγω των συντακτικών ιδιαιτεροτήτων της κάθε γλώσσας.

**Grace Hopper: Μια πρωτοπόρος στον σχεδιασμό γλωσσών προγραμματισμού**

Η Grace Hopper, γνωστή και ως «Amazing Grace», ήταν μια Αμερικανίδα επιστήμονας υπολογιστών και ναύαρχος του Πολεμικού Ναυτικού των Ηνωμένων Πολιτειών. Θεωρείται ευρέως ως μια από τις πιο σημαντικές προσωπικότητες στην ιστορία της Πληροφορικής, καθώς συνέβαλε καθοριστικά στον σχεδιασμό και την ανάπτυξη της πρώτης γλώσσας προγραμματισμού υψηλού επιπέδου, της COBOL. Η Hopper θεωρείται η πρώτη που συνέλαβε την ιδέα μιας γλώσσας προγραμματισμού που θα χρησιμοποιούσε αγγλικές λέξεις αντί για τη γλώσσα μηχανής, όπου οι εντολές ήταν αριθμοί στο δυαδικό σύστημα.

Οι υπολογιστές, όμως, δεν καταλάβαιναν αγγλικό κείμενο, μόνο γλώσσα μηχανής. Έτσι, η Hopper πρότεινε την ιδέα του μεταγλωττιστή (compiler), ένα πρόγραμμα το οποίο μεταφράζει το αγγλικό κείμενο που είναι κατανοητό στους ανθρώπους σε γλώσσα μηχανής που είναι κατανοητή στους υπολογιστές. Ένας διερμηνέας, δηλαδή, μεταξύ ανθρώπου και μηχανής, κάτι που για εκείνη την εποχή φαινόταν ανέφικτο. Μάλιστα, αρκετοί συνάδελφοι της Hopper την προσγείωναν, λέγοντάς της ότι κάτι τέτοιο δεν μπορεί να γίνει. Ωστόσο, η Grace Hopper επέμεινε στο όνειρό της και τα κατάφερε. Σχεδίασε τον πρώτο μεταγλωττιστή ο οποίος στη συνέχεια αξιοποιήθηκε ως βάση για την πρώτη γλώσσα προγραμματισμού υψηλού επιπέδου, την COBOL. Παρακάτω δίνεται ένα πρόγραμμα το οποίο προσθέτει όλους τους ακέραιους αριθμούς από 1 έως και 10 σε γλώσσα μηχανής σε γλώσσα Assembly και σε Python. Η Assembly (συμβολική γλώσσα) είναι μια γλώσσα που βρίσκεται ένα επίπεδο πάνω από τη γλώσσα μηχανής. Η διαφορά είναι ότι έχουν δοθεί λεκτικές περιγραφές σε κάθε ακολουθία δυαδικών ψηφίων που αναπαριστά κάποια εντολή.

| Γλώσσα Μηχανής | Γλώσσα Assembly | Python |
|---|---|---|
| | INDEX = $01 | |
| | SUM = $02 | |
| 10101000 00001010 | LDA #10 | INDEX = 10 |
| 10001100 00000001 | STA INDEX | |
| 00111100 | CLA | SUM = 0 |
| | | |
| 01010001 00000001 | LOOP ADD INDEX | while INDEX > 0: |
| 01000011 00000001 | DEC INDEX | SUM += INDEX |
| 11000000 11111010 | BNE LOOP | INDEX -= 1 |
| 10001100 00000010 | | |
| 11111111 | STA SUM | print("Τελικό Άθροισμα:", SUM) |
| | BRK | |

Σήμερα, πολλές σύγχρονες γλώσσες προγραμματισμού παράγουν μια ενδιάμεση μορφή κώδικα για μια νοητή μηχανή (virtual machine) με σκοπό την ανεξαρτησία του προγράμματος από συγκεκριμένες πλατφόρμες. Έτσι, ένα πρόγραμμα γραμμένο σε μια γλώσσα μπορεί να εκτελείται το ίδιο σε διάφορα λειτουργικά συστήματα ή υπολογιστές διαφορετικών αρχιτεκτονικών. Η μορφή αυτή κώδικα μηχανής λέγεται συνήθως bytecode. Παρακάτω δίνεται ένα τέτοιο παράδειγμα για τη γλώσσα Python.

**Python Bytecode**

```
 0 LOAD_CONST     1 (28)
 2 STORE_FAST     0 (X)

 4 LOAD_CONST     2 (496)
 6 STORE_FAST     1 (Y)

 8 LOAD_FAST      0 (X)
10 LOAD_FAST      1 (Y)
12 BINARY_ADD
14 STORE_FAST     2 (Z)
16 LOAD_CONST     0 (None)
18 RETURN_VALUE
```

**Πηγαίος κώδικας σε Python**

```
def test():
    X = 28
    Y = 496
    Z = X + Y
```

Όπως οι φυσικές γλώσσες, έτσι και οι γλώσσες προγραμματισμού έχουν κάποια βασικά χαρακτηριστικά, όπως:

| | | |
|---|---|---|
| **Αλφάβητο** | Είναι το σύνολο των χαρακτήρων (συμβόλων) που χρησιμοποιούνται από τη γλώσσα. | A, B, C, …, Z, a..z, 0,1..9, _ |
| **Λεξιλόγιο** | Το σύνολο των λέξεων (tokens) που αναγνωρίζει η γλώσσα ως αποδεκτά και έχουν κάποια σημασία. | **this**, **while**, quantum_bit |
| **Συντακτικό** | Το σύνολο των κανόνων που ακολουθούμε για να συνδέουμε λέξεις σε αποδεκτές από τη γλώσσα εκφράσεις και εντολές. | **if** sensor == “Rainy” **then**<br>roofTop = **True** |

Για να συντάξουμε ένα πρόγραμμα σε κάποια γλώσσα προγραμματισμού, θα πρέπει να χρησιμοποιήσουμε ένα περιβάλλον προγραμματισμού το οποίο περιλαμβάνει μια σειρά από εργαλεία, όπως είναι ο **συντάκτης** (editor) και ο **μεταγλωττιστής** (compiler). Ο συντάκτης είναι ένα πρόγραμμα επεξεργασίας κειμένου με ειδικές όμως λειτουργίες για τη σύνταξη και τη διόρθωση του κώδικα. Ο μεταγλωττιστής είναι το πρόγραμμα το οποίο μεταφράζει ένα πρόγραμμα σε γλώσσα προγραμματισμού στη γλώσσα που είναι κατανοητή από τη μηχανή, ώστε να μπορεί να την εκτελέσει.

Το πρόγραμμα σε γλώσσα μηχανής παράγεται μόνο αν δε βρεθούν συντακτικά ή άλλα λάθη στον αρχικό πηγαίο κώδικα (source code), όπως φαίνεται δίπλα όπου η μεταβλητή d δεν έχει οριστεί.

```
double a = 1, b = 2;
int c = d;
```

CS0103: The name 'd' does not exist in the current context

Show potential fixes (Alt+Enter or Ctrl+.)

Σήμερα, αρκετοί συντάκτες κώδικα έχουν αρκετά προχωρημένες λειτουργίες, όπως η πρόβλεψη της επόμενης εντολής που έχουμε στο μυαλό μας με χρήση εργαλείων τεχνητής νοημοσύνης, τα οποία αξιοποιούνται πλέον από αρκετούς προγραμματιστές.

```
{
    int a = 1;
    int temp = a;
}
```

Ένα άλλο είδος μεταφραστών πηγαίου κώδικα σε γλώσσα μηχανής είναι οι **διερμηνευτές,** οι οποίοι μεταφράζουν και εκτελούν τον κώδικά εντολή-εντολή. Ενώ οι μεταγλωττιστές ελέγχουν όλο το πρόγραμμα και το μεταφράζουν σε κώδικα μηχανής μόνο αν δε βρεθούν συντακτική λάθη, οι διερμηνευτές μπορεί να εκτελέσουν τις πρώτες εντολές και στη συνέχεια να σταματήσουν την εκτέλεση στη μέση επειδή βρέθηκε λάθος σε επόμενη εντολή.

Ένα πρόγραμμα το οποίο έχει περάσει από όλους τους ελέγχους του μεταγλωττιστή/διερμηνευτή δε σημαίνει απαραίτητα ότι έχει σωστή λειτουργία. Για παράδειγμα οι εντολές δίπλα εκτελούνται χωρίς πρόβλημα, αλλά το αποτέλεσμά τους είναι λανθασμένο. Αυτά τα λάθη λέγονται **λογικά** και αποτελούν τη δυσκολότερη κατηγορία λαθών στην ανίχνευση και διόρθωσή τους.

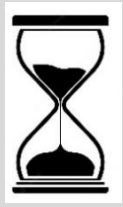
```
average = (a + b + c) / 2
```

```
if number>0 :
    print("Ο αριθμός είναι αρνητικός")
```

## 8.5 Ερωτήσεις - Ασκήσεις

**Ε.1: Διάσχιση Γέφυρας.** Πέντε άνθρωποι θέλουν να περάσουν μια γέφυρα. Είναι όμως νύχτα και διαθέτουν μόνο μια λάμπα με φυτίλι διάρκειας 30 λεπτών. Η γέφυρα αντέχει το πολύ δύο άτομα, οπότε δε μπορούν να περάσουν όλοι μαζί. Για να περάσουν τη γέφυρα χρειάζονται οπωσδήποτε τη λάμπα. Επίσης, κάθε ένας χρειάζεται διαφορετικό χρόνο για να περάσει τη γέφυρα. Ο πιο γρήγορος χρειάζεται 1 λεπτό, ο επόμενος 3 λεπτά, και υπόλοιποι 6, 8 και 12 λεπτά αντίστοιχα. Πώς θα περάσουν απέναντι;

**Ε.2:** Έχουμε ένα μπρίκι με νερό που βράζει και ένα αβγό που πρέπει να βράσουμε για 9 λεπτά ακριβώς. Δυστυχώς, δεν έχουμε κανένα ρολόι παρά μόνο δύο κλεψύδρες, η μία διάρκειας 7 και η άλλη 4 λεπτών. Ποιος είναι ο συντομότερος τρόπος για να μετρήσουμε 9 λεπτά;



**Ε.3:** Έχουμε 9 όμοια κέρματα από τα οποία το ένα είναι κάλπικο, δηλαδή πιο ελαφρύ. Επίσης, διαθέτουμε μια ζυγαριά ισορροπίας, όπως αυτή της εικόνας, με την οποία μπορούμε να συγκρίνουμε μεταξύ τους 2 βάρη.

**α)** Θέλουμε με 2 μόνο ζυγίσεις να εντοπίσουμε το κάλπικό κέρμα.

**β)** Πόσες ζυγίσεις θα χρειαστούμε για 81 κέρματα και πόσες για 36;

**Ε.4:** Μετρήστε ακριβώς 9 λίτρα, αν έχετε στη διάθεσή σας ένα δοχείο των 10 λίτρων, ένα των 7 λίτρων και μια πηγή με άφθονο νερό. Μπορείτε μόνο να γεμίζετε μέχρι πάνω και να αδειάζετε εντελώς τα δοχεία όσες φορές θέλετε.

**Ε.5:** Έχετε τρία δοχεία, ένα των 12 λίτρων, ένα των 8 και ένα των 5. Το δοχείο των 12 λίτρων είναι γεμάτο, ενώ τα άλλα δύο άδεια. Δεν έχετε άλλο νερό, μόνο τα 12 λίτρα. Να χωρίσετε την ποσότητα του νερού στη μέση, δηλαδή 6 λίτρα στο 12λιτρο και 6 λίτρα στο 8λιτρο. Μπορείτε μόνο να γεμίσετε ή να αδειάσετε τα δοχεία μέχρι τέλους, όσες φορές θέλετε. Τα δοχεία δεν έχουν καμία ογκομετρική ένδειξη επάνω τους. Σας δίνετε μια γλώσσα προγραμματισμού για το πρόβλημα των δοχείων η οποία έχει μόνο μια εντολή:

**Άδειασε**(Δοχείο1, Δοχείο2): Μετακινεί το περιεχόμενο του δοχείου 1 στο δοχείο 2.

π.χ. η εντολή Άδειασε(12 , 5) όταν αρχικά το 12λιτρο είναι γεμάτο και το 5λιτρο άδειο, αδειάζει το 12λιτρο στο 5λιτρο, άρα στο 12λιτρο θα μείνουν 7 λίτρα.

**Ε.6:** Αν είχατε μια κανάτα των 6 λίτρων και μια των 2 λίτρων, θα μπορούσατε να μετρήσετε ακριβώς 5 λίτρα; Αιτιολογήστε την απάντησή σας.