Η ειδοποιός διαφορά του υπολογιστή από τις άλλες ηλεκτρονικές κατασκευές του ανθρώπου είναι η δυνατότητα **προγραμματισμού** του, με το πρόγραμμα μάλιστα να καθορίζει κάθε στιγμή τη λειτουργία του. Η πληθώρα των θεμάτων που σχετίζονται με τον προγραμματισμό του υπολογιστή είναι τόσο μεγάλη που, όπως είναι φυσικό, αποτέλεσε ένα πολύ μεγάλο κλάδο της επιστήμης των υπολογιστών.
Σε αυτό το κεφάλαιο θα αναφερθούμε σε βασικά θέματα του προγραμματισμού των υπολογιστών και πώς αυτά αντιμετωπίζονται. Θα διαπιστώσουμε ότι, ουσιαστικά, ο προγραμματισμός είναι μια διαδικασία επίλυσης προβλημάτων με τη βοήθεια του υπολογιστή.

7.1 Η προγραμματιζομένη μηχανή

Στο πρώτο κεφάλαιο γνωρίσαμε μερικές από τις εφαρμογές της Πληροφορικής στο σύγχρονο κόσμο. Εάν εξετάσουμε τους υπολογιστές που χρησιμοποιούνται σε αυτές ως προς το υλικό τους μέρος, θα διαπιστώσουμε πως, παρ' όλο που η εφαρμογή είναι διαφορετική, σε πολλές περιπτώσεις το υλικό των χρησιμοποιούμενων υπολογιστών δεν διαφοροποιείται.
Ο ίδιος υπολογιστής μπορεί να χρησιμοποιηθεί από ένα πολυκατάστημα αλλά και από ένα λογιστικό γραφείο ή από ένα νοσοκομείο, με μόνη πιθανή διαφορά στις περιφερειακές συσκευές. Αυτό που επιτρέπει στο υλικό του υπολογιστή να προσαρμόζεται σε τόσο διαφορετικές απαιτήσεις είναι τα προγράμματά του, το λογισμικό.
Ήδη επισημάναμε ότι μια από τις βασικότερες διαφορές ανάμεσα στον υπολογιστή και στις περισσότερες ηλεκτρονικές συσκευές είναι η δυνατότητα προγραμματισμού του. Πρακτικά αυτό σημαίνει ότι η λειτουργία του υπολογιστή δεν καθορίζεται μόνο από τα ηλεκτρονικά του εξαρτήματα και τη μεταξύ τους συνδεσμολογία, αλλά κυρίως από το πρόγραμμα που κατά περίπτωση αυτός εκτελεί. Το γεγονός ότι ο υπολογιστής, ως ηλεκτρονική συσκευή, δεν είναι κατασκευασμένος να κάνει κάτι συγκεκριμένο, δεν είναι αδυναμία του αλλά το πλεονέκτημά του. Ανάλογα με το πρόγραμμα που εκτελεί μετασχηματίζεται στα μάτια του χρήστη σε μια διαφορετική μηχανή. Το λογισμικό εφαρμογών, το οποίο είδαμε σε προηγούμενο κεφάλαιο, είναι εκείνο που καθιστά τον υπολογιστή στα χέρια του αρχιτέκτονα ένα εξελιγμένο σχεδιαστικό εργαλείο, στα χέρια του συγγραφέα μια ευέλικτη γραφομηχανή ή στα χέρια του λογιστή ένα λογιστικό εργαλείο, κ.ο.κ.

7.2 Οι χρήστες

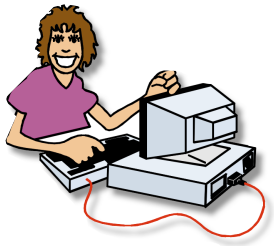
Τους χρήστες των υπολογιστικών συστημάτων μπορούμε να τους κατατάξουμε σε δύο μεγάλες κατηγορίες:

- σε αυτούς που χρησιμοποιούν τον υπολογιστή στη δουλειά τους, όπως είναι οι αρχιτέκτονες, οι συγγραφείς, οι λογιστές, κλπ. και
- σε αυτούς που η δουλειά τους σχετίζεται με τον ίδιο τον υπολογιστή.

Οι χρήστες της πρώτης κατηγορίας ονομάζονται «τελικοί χρήστες», ενώ στη δεύτερη υπάγονται οι επαγγελματίες της Πληροφορικής. Μια μεγάλη μερίδα των τελευταίων κατασκευάζει τα προγράμματα -το λογισμικό- που χρησιμοποιούνται από όλους τους άλλους.

Αυτή η ιδιότητα του κατασκευαστή και του χρήστη δεν είναι απόλυτα μονοσήμαντη. Ο κατασκευαστής λογισμικού, για να κάνει τη δουλειά του, χρησιμοποιεί με τη σειρά του τα προγράμματα άλλων κατασκευαστών, οπότε γίνεται και ο ίδιος με τη σειρά του χρήστης. Ωστόσο ο όρος **τελικός χρήστης** (end user) συνηθίζεται να δηλώνει το χρήστη που δεν είναι ειδικός στην Πληροφορική, αλλά χρησιμοποιεί τις εφαρμογές της Πληροφορικής στην εργασία του.

Ενδιαφέρον παρουσιάζει όχι μόνο η οπτική γωνία από την οποία βλέπει τον υπολογιστή ο προγραμματιστής αλλά και τα εργαλεία λογισμικού που αυτός χρησιμοποιεί στη δουλειά του.



7.3 Πρόγραμμα - Γλώσσες προγραμματισμού

7.3.1 Το Πρόγραμμα

Ήδη προαναφέρθηκε ότι ο υπολογιστής είναι μια προγραμματιζόμενη μηχανή. Δηλαδή, για να εκτελέσει ακόμη και την πιο απλή εργασία, θα πρέπει να του έχουν δοθεί λεπτομερείς οδηγίες για να την επιτελέσει. Αυτές οι οδηγίες αποτελούν το πρόγραμμα και εκτελούνται από την κεντρική μονάδα επεξεργασίας (ΚΜΕ) του υπολογιστή.

Όπως είδαμε στο κεφάλαιο του υλικού των υπολογιστών, οι σύγχρονοι υπολογιστές είναι βασισμένοι στις αρχές που διατύπωσε ο Von Neumann και η ομάδα του κατά τη δεκαετία του 1940. Η μηχανή του Von Neumann χαρακτηρίζεται από ένα μεγάλο αριθμό κελιών μνήμης και μια μονάδα επεξεργασίας που περιέχει ένα μικρό σχετικά αριθμό κελιών, τους καταχωρητές. Η ΚΜΕ μπορεί να «φορτώσει» στοιχεία από τη μνήμη στους καταχωρητές, να εκτελέσει κάποιες αριθμητικές (πρόσθεση, αφαίρεση, κλπ.) και λογικές (AND, OR, NOT, κλπ.) πράξεις με το περιεχόμενο των καταχωρητών και να αποθηκεύσει τις τιμές από τους καταχωρητές πίσω στη μνήμη.

Για μια μηχανή Von Neumann το **πρόγραμμα** αποτελείται από μια σειρά οδηγιών, που ονομάζονται **εντολές**, για την εκτέλεση τέτοιου είδους πράξεων, καθώς επίσης και από ένα

Πρώτος προγραμματιστής θεωρείται η Augusta Ada Byron, κόμμισα του Lovelace (1815 - 1852). Ήταν κόρη του Λόρδου Βύρωνα με αξιόλογη μόρφωση και με ταλέντο στα Μαθηματικά. Όταν ο Charles Babbage έφερε τη διαφορική μηχανή του στο σπίτι της μητέρας της, η Augusta γοητεύτηκε από τις δυνατότητές της, με αποτέλεσμα να συνεργαστεί για χρόνια με τον Babbage στον προγραμματισμό της. Ωστόσο ο Babbage δεν κατάφερε να κάνει τη διαφορική μηχανή του να δουλέψει. Μαζί ανέπτυξαν ένα σύστημα για να κερδίζουν σε ιπποδρομίες. Αλλά ούτε αυτό πέτυχε, με αποτέλεσμα να αναγκαστεί να πουλήσει οικογενειακά κοσμήματα για να καλύψει τα χρέη από τον ιππόδρομο. Η Augusta πέθανε από καρκίνο στα 37 της χρόνια, στην ίδια ηλικία με το διάσημο πατέρα της. Προς τιμήν της έχει ονομαστεί Ada μια γλώσσα προγραμματισμού που υιοθετήθηκε στις αρχές του 1980 από το Υπουργείο Άμυνας των Η.Π.Α.

σύνολο πρόσθετων οδηγιών ελέγχου, που επηρεάζουν τη σειρά με την οποία εκτελούνται οι εντολές.
Είναι γεγονός ότι το ρεπερτόριο αυτών των βασικών λειτουργιών της ΚΜΕ είναι εξαιρετικά περιορισμένο, όμως είναι αρκετό, ώστε με τον κατάλληλο συνδυασμό τους να μπορούμε να δώσουμε στον υπολογιστή τις απαραίτητες οδηγίες για να κάνει όλα όσα βλέπουμε ότι γίνονται από υπολογιστές γύρω μας.

7.3.2 Γλώσσα μηχανής

Επειδή ο κάθε τύπος ΚΜΕ έχει διαφορετικό ρεπερτόριο εντολών, τα προγράμματα που εκτελεί πρέπει να είναι διατυπωμένα στη δική της γλώσσα μηχανής. Αυτά τα προγράμματα αποτελούνται από ακολουθίες 0 και 1.

Οι εντολές που αναγνωρίζει μια τυπική ΚΜΕ ανήκουν σε μια από τις πιο κάτω κατηγορίες:

- εντολές μεταφοράς δεδομένων μεταξύ της κεντρικής μνήμης και των καταχωρητών της ΚΜΕ
- εντολές μεταφοράς δεδομένων μεταξύ των καταχωρητών
- εντολές αριθμητικών πράξεων
- εντολές λογικών πράξεων
- εντολές ελέγχου της ροής εκτέλεσης των εντολών
- διάφορες βοηθητικές εντολές.

Κάθε εντολή στη γλώσσα μηχανής αποτελείται από δύο τμήματα, τον **κωδικό λειτουργίας** (operation code, OP code) και τον **τελεστέο** (operand).

Κωδικός λειτουργίας	Τελεστέος

Ο κωδικός λειτουργίας προσδιορίζει τη λειτουργία της εντολής, για παράδειγμα «πρόσθεσε στο συσσωρευτή» ή «αποθήκευσε το περιεχόμενο του συσσωρευτή στην τάδε θέση μνήμης», και αντιστοιχεί σε μια από τις οδηγίες του συνόλου οδηγιών της ΚΜΕ. Ο τελεστέος αφορά τα δεδομένα στα οποία δρα η εντολή ή τη διεύθυνση στην οποία βρίσκονται αποθηκευμένα. Το μέγεθος (ο αριθμός των bit) κάθε εντολής μπορεί να είναι σταθερός ή μεταβλητός.

Z80: Επεξεργαστής της εταιρείας Zilog στο τέλος της δεκαετίας του '70

Για παράδειγμα, στον επεξεργαστή Z80, η εντολή 1110011010110011, κατευθύνει την ΚΜΕ να προσθέσει στο περιεχόμενο του καταχωρητή με το όνομα accumulator τον αντίστοιχο δυαδικό του $179_{(10)}$ δεκαδικό αριθμό 179. Τα πρώτα 8 ψηφία είναι ο κωδικός λειτουργίας, ενώ τα 8 επόμενα ο τελεστέος της εντολής, ο αριθμός $179_{(10)}$.
Ένα πρόγραμμα γραμμένο σε γλώσσα μηχανής θα έμοιαζε κάπως έτσι:

```
1110011010110011
1011011010111011
1010011110110110100100010
0101011010010011
1110011010110110
1100011010110111110110100
1101011000110101
10101110
1100011010110010
```

(Κωδικός Λειτουργίας/Τελεστέος)

Η κάθε εντολή θα μπορούσε, αντί να παρασταθεί στο δυαδικό σύστημα, να παρασταθεί στο οκταδικό ή στο δεκαεξαδικό, ώστε να είναι μικρότερος ο αριθμός των ψηφίων της.
Τα πρώτα προγράμματα γράφονταν σε γλώσσα μηχανής. Ωστόσο η γραφή προγραμμάτων άμεσα στη γλώσσα μηχανής του υπολογιστή είναι μια πολύ δύσκολη και αντιπαραγωγική εργασία. Δύο από τις βασικότερες αιτίες είναι η δυσκολία απομνημόνευσης των κωδικών των εντολών και η δυσκολία να εντοπιστεί κάποιο πιθανό λάθος ανάμεσα σε όλα αυτά τα 0 και 1.
Γι΄ αυτό, από πολύ νωρίς, αναζητήθηκαν τρόποι να γράφονται τα προγράμματα σε γλώσσα πιο προσιτή στον άνθρωπο. Η επιδίωξη αυτή, οδήγησε στην ανάπτυξη των διαφόρων γλωσσών προγραμματισμού, καθώς επίσης και στην ανεύρεση τρόπων για την εκτέλεση από τον υπολογιστή των προγραμμάτων που γράφονται σε αυτές τις γλώσσες.

7.3.3 Συμβολικές γλώσσες

Μια πρώτη προσπάθεια για τη διευκόλυνση της γραφής προγραμμάτων ήταν η γραφή τους σε **συμβολική γλώσσα** (assembly). Στις συμβολικές γλώσσες, σε κάθε εντολή της γλώσσας μηχανής αντιστοιχίζεται μια μνημονική λέξη η οποία θυμίζει το σκοπό της εντολής. Επί πλέον δίνεται η δυνατότητα να χρησιμοποιούνται, στη θέση αριθμών, ονόματα που αντιπροσωπεύουν σταθερές ή διευθύνσεις μνήμης. Έτσι μπορούμε να γράφουμε HLIKIA αντί για 18.
Η εντολή του προηγουμένου παραδείγματος της γλώσσας μηχανής του Z80 με τη μορφή 11100110 10110011, θα μπορούσε σε συμβολική γλώσσα να έχει τη μορφή:

ADD B3h

B3 είναι η δεκαεξαδική μορφή του 179. Το h δηλώνει ότι πρόκειται για δεκαεξαδικό αριθμό και όχι για όνομα B3

Στη συνέχεια βλέπουμε τμήμα ενός προγράμματος σε συμβολική γλώσσα, για επεξεργαστή της σειράς 80X86 της Intel.

```
mov   ax,DGROUP
mov   ds,ax
mov   ds:[STKHQQ],0
mov   es:[cdataseg],ax
mov   ax,sp
add   ax,bpregz+pgdata+14h
mov   [_atopsp],ax
mov   bx,seg STACK
sub   bx,DGROUP
mov   cl,4
shl   bx,cl
add   ax,bx
mov   [_abrktb].sz,ax
dec   ax
mov   [_asizds],ax
```

7.3.4 Γλώσσες υψηλού επιπέδου

Οι συμβολικές γλώσσες διευκόλυναν τη γραφή προγραμμάτων. Όμως οι εντολές τους εξακολουθούν να είναι σε άμεση (μία προς μία) αντιστοιχία με αυτές της γλώσσας μηχανής, με αποτέλεσμα ο προγραμματισμός να εξακολουθεί να είναι μια εξαιρετικά επίπονη διαδικασία. Επιπλέον τα προγράμματα που γράφονται σε συμβολική γλώσσα ή γλώσσα μηχανής μπορούν να εκτελεστούν μόνο από τον τύπο υπολογιστή για τον οποίο είναι γραμμένα. **Αυτό σημαίνει πως για να «τρέξει» το ίδιο πρόγραμμα σε έναν άλλο τύπο υπολογιστή θα πρέπει να γραφεί από την αρχή.**

Μεταφερσιμότητα προγράμματος

Για να αντιμετωπιστούν τέτοιου είδους προβλήματα, άρχισαν να δημιουργούνται και να εξελίσσονται νέες γλώσσες προγραμματισμού πιο απλές και ανεξάρτητες από το συγκεκριμένο τύπο υπολογιστή, που ονομάστηκαν γλώσσες **υψηλού επιπέδου** (high level languages). Αυτές οι γλώσσες, καθεμία σε διαφορετικό βαθμό, κρύβουν από τον προγραμματιστή τη γλώσσα μηχανής και προσφέρουν ένα πιο φιλικό σύνολο εντολών με τις οποίες συντάσσεται το κάθε πρόγραμμα. Προγράμματα αυτού του είδους με μικρές πιθανόν αλλαγές στον πηγαίο κώδικα μπορούν να εκτελεστούν στη συνέχεια και σε άλλους τύπους υπολογιστών, εκτός από αυτόν για τον οποίο αρχικά κατασκευάστηκαν. Αυτό το χαρακτηριστικό ενός προγράμματος ονομάζεται **μεταφερσιμότητα.**

7.3.5 Ένα συγκριτικό παράδειγμα

Έστω ότι θέλουμε να προσθέσουμε το περιεχόμενο δύο θέσεων μνήμης και το αποτέλεσμα να το καταχωρίσουμε σε μια τρίτη. Για λόγους σύγκρισης, ας δούμε πώς θα μπορούσε αυτό να γραφεί σε γλώσσα υψηλού επιπέδου, σε συμβολική γλώσσα και σε γλώσσα μηχανής:

Γλώσσα υψηλού επιπέδου

Εντολή	Περιγραφή
`A:= B + C`	Πρόσθεσε το περιεχόμενο των μεταβλητών B και C και το αποτέλεσμα καταχώρισέ το στη μεταβλητή A

Συμβολική γλώσσα

Εντολή	Περιγραφή
`LDA B`	Μετάφερε στο συσσωρευτή το περιεχόμενο της θέσης μνήμης με όνομα B
`ADD C`	Πρόσθεσε στο περιεχόμενο του συσσωρευτή το περιεχόμενο της θέσης μνήμης με όνομα C
`STA A`	Μετάφερε και αποθήκευσε το περιεχόμενο του συσσωρευτή στη θέση μνήμης με όνομα A

Γλώσσα μηχανής

Εντολή	Περιγραφή
0000001001011010	Μετάφερε στο συσσωρευτή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011010
0000101001011110	Πρόσθεσε στο περιεχόμενο του συσσωρευτή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011110
0000011011011110	Μετάφερε και αποθήκευσε το περιεχόμενο του συσσωρευτή στη θέση μνήμης με διεύθυνση 11011110

Όπως φαίνεται, για να πετύχουμε το ίδιο αποτέλεσμα, απαιτήθηκε μια εντολή σε γλώσσα υψηλού επιπέδου και αντίστοιχα τρεις σε συμβολική γλώσσα και σε γλώσσα μηχανής. Επιπλέον, ενώ στη γλώσσα υψηλού επιπέδου αναφερόμαστε σε μεταβλητές A, B, C, χωρίς να γίνεται αναφορά ούτε σε συγκεκριμένες θέσεις μνήμης ούτε στη διαδικασία της πρόσθεσης, στις άλλες δύο περιπτώσεις συμβαίνει ακριβώς το αντίθετο. Έτσι είναι απαραίτητο να αναφερθούμε και στη διαδικασία της πρόσθεσης και σε συγκεκριμένες θέσεις μνήμης είτε με όνομα είτε με διεύθυνση.

7.3.8 Μεταφραστές

Όπως ήδη αναφέραμε, η ΚΜΕ αναγνωρίζει και εκτελεί εντολές σε γλώσσα μηχανής. Το ερώτημα λοιπόν είναι: πώς τα προγράμματα, που είναι γραμμένα σε οποιαδήποτε άλλη μορφή εκτός της γλώσσας μηχανής, εκτελούνται από την ΚΜΕ του υπολογιστή;
Θα πρέπει τα προγράμματα που είναι γραμμένα σε κάποια γλώσσα υψηλού επιπέδου να «μεταφραστούν» σε γλώσσα μηχανής. Το ρόλο αυτό τον αναλαμβάνουν ειδικά προγράμματα. Στην περίπτωση των συμβολικών γλωσσών, χρησιμοποιούνται οι **συμβολομεταφραστές**, ενώ στις γλώσσες υψηλού επιπέδου, οι **μεταγλωττιστές** και οι **διερμηνευτές**.

Συμβολομεταφραστές

Σε προηγούμενο παράδειγμα διαπιστώσαμε ότι οι εντολές των συμβολικών γλωσσών βρίσκονται σε αντιστοιχία μία προς μία με αυτές της γλώσσας μηχανής. Η μετατροπή των προγραμμάτων από συμβολική γλώσσα σε γλώσσα μηχανής γίνεται από ειδικά προγράμματα που ονομάζονται συμβολομεταφραστές (assemblers). Η βασική τους λειτουργία συνίσταται στο να αντικαθιστούν με τη χρήση ενός λεξικού τις συμβολικές εντολές με τις αντίστοιχες της γλώσσας μηχανής.
Οι σύγχρονοι συμβολομεταφραστές, χωρίς να απομακρύνονται από αυτήν τη βασική λειτουργία, προσφέρουν διάφορες ευκολίες, όπως:

- να δίνονται ονόματα σε θέσεις μνήμης ή σε αριθμητικές αξίες και ο συμβολομεταφραστής αναλαμβάνει να κάνει τις μετατροπές
- να επιτρέπουν την κατασκευή μακροεντολών.

Έχει παρατηρηθεί ότι σε ένα πρόγραμμα κάποιες ομάδες εντολών επαναλαμβάνονται σε διαφορετικά σημεία αυτούσιες ή με μικρές παραλλαγές. Ο μηχανισμός των μακροεντολών επιτρέπει να ορίσουμε ένα όνομα που να αντιπροσωπεύει την επαναλαμβανόμενη σειρά εντολών, την οποία καταγράφουμε αναλυτικά για μια φορά. Σε κάθε σημείο που χρειαζόμαστε την ίδια σειρά εντολών, αρκεί να γίνεται αναφορά σε αυτό το όνομα και ο συμβολομεταφραστής αναλαμβάνει την αντικατάσταση του ονόματος της μακροεντολής με τις εντολές που αυτή εμπεριέχει.

Μεταγλωττιστές - Διερμηνευτές

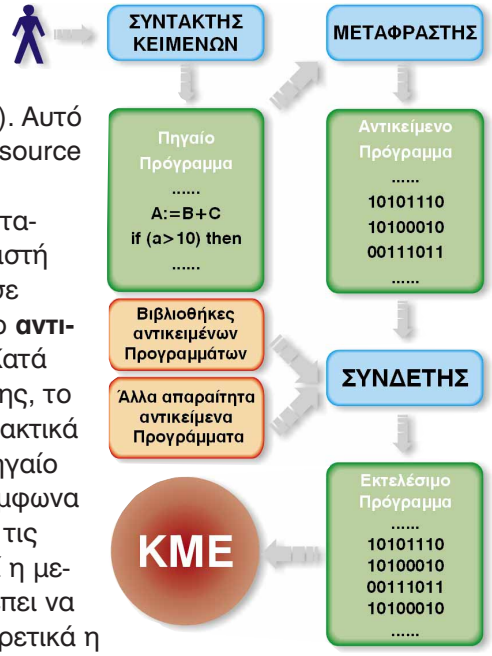
Υπάρχουν δύο τρόποι για την εκτέλεση από τον υπολογιστή προγραμμάτων που είναι γραμμένα σε γλώσσες υψηλού επιπέδου:

- με τη χρήση μεταγλωττιστή (compiler)
- με τη χρήση διερμηνευτή (interpreter).

Μεταγλωττιστής

Στο σχήμα φαίνεται η διαδικασία εκτέλεσης ενός προγράμματος με τη χρήση **μεταγλωττιστή**. Η διαδικασία περιλαμβάνει τα εξής βήματα:

- Γραφή του προγράμματος σε γλώσσα υψηλού επιπέδου -π.χ. PASCAL- με τη βοήθεια ενός συντάκτη κειμένων (editor). Αυτό είναι το **πηγαίο** πρόγραμμα (source code, source program).
- Το πηγαίο αυτό πρόγραμμα μεταγλωττίζεται, από το μεταγλωττιστή της συγκεκριμένης γλώσσας, σε γλώσσα μηχανής. Αυτό είναι το **αντικείμενο** (object) πρόγραμμα. Κατά τη διαδικασία της μεταγλώττισης, το πρόγραμμα ελέγχεται για συντακτικά λάθη, δηλαδή κατά πόσο το πηγαίο πρόγραμμα είναι γραμμένο σύμφωνα με τους συντακτικούς κανόνες τις γλώσσας. Για να ολοκληρωθεί η μεταγλώττιση το πρόγραμμα πρέπει να είναι συντακτικά σωστό, διαφορετικά η διαδικασία της μεταγλώττισης διακόπτεται.



- Το αντικείμενο πρόγραμμα, παρ' όλο που είναι σε γλώσσα μηχανής, δεν μπορεί να εκτελεστεί από την ΚΜΕ, γιατί δεν είναι αυτόνομο. Συνήθως του λείπει κάποιος κώδικας κοινής χρήσης, που βρίσκεται σε βιβλιοθήκες αντικειμένων προγραμμάτων. Όταν λέμε κώδικα κοινής χρήσης, συνήθως εννοούμε κώδικα που διαχειρίζεται τις λειτουργίες εισόδου εξόδου ή κώδικα που αφορά μαθηματικές συναρτήσεις (sin, cos, κλπ.). Τη σύνδεση του αντικειμένου προγράμματος με τα απαραίτητα προγράμματα από τις βιβλιοθήκες αναλαμβάνει ο **συνδέτης** (linker), ο οποίος παράγει και το αυτόνομο εκτελέσιμο πρόγραμμα. Αυτό το τελευταίο είναι που εκτελείται από την ΚΜΕ.

Τα βήματα δηλαδή είναι:

- Συγγραφή
- Μεταγλώττιση
- Σύνδεση
- Εκτέλεση.

Στο περιβάλλον του MS-DOS και των MS-Windows τα εκτελέσιμα προγράμματα διακρίνονται από την επέκταση **com** ή **exe** στο όνομα του αρχείου.

Τα βήματα αυτά μπορεί να γίνονται είτε σαν μια συνεχόμενη διαδικασία είτε το καθένα σε διαφορετικό χρόνο. Στο κάθε βήμα μπορεί να προκύψουν λάθη, οπότε διορθώνουμε το λάθος και η διαδικασία επαναλαμβάνεται.
Στην πιο πάνω διαδικασία ο συντάκτης κειμένων, ο μεταφραστής και ο συνδέτης, είναι προγράμματα τα οποία μας βοηθούν να παραγάγουμε το εκτελέσιμο πρόγραμμα. Από τη στιγμή που το εκτελέσιμο πρόγραμμα παραχθεί, μπορεί να εκτελεστεί αυτόνομα από την ΚΜΕ, χωρίς να είναι απαραίτητος ο συντάκτης κειμένων, ο μεταφραστής ή ο συνδέτης.
Τελείως διαφορετική σε αυτό το σημείο είναι η προσέγγιση της εκτέλεσης προγράμματος γραμμένου σε γλώσσα υψηλού επιπέδου με τη χρήση **διερμηνευτή**.

Διερμηνευτής

Το πρόγραμμα γράφεται και σε αυτήν την περίπτωση με τη βοήθεια κάποιου συντάκτη κειμένων, ο οποίος είναι συνήθως ενσωματωμένος στο πρόγραμμα

του διερμηνευτή. Στη συνέχεια ο διερμηνευτής αναλαμβάνει να εκτελέσει μία μία τις εντολές του προγράμματος σαν να απευθύνονται σε αυτόν και όχι στην ΚΜΕ. Ο διερμηνευτής, δηλαδή, δεν μεταγλωττίζει το πρόγραμμα σε γλώσσα μηχανής όπως κάνει ο μεταγλωττιστής, αλλά εκτελεί ο ίδιος το πρόγραμμα. Συνήθως ο διερμηνευτής μετασχηματίζει το πρόγραμμά μας σε μια ενδιάμεση μορφή και στη συνέχεια το εκτελεί.

Στην περίπτωση αυτή το πρόγραμμά μας δεν εκτελείται άμεσα από την ίδια την ΚΜΕ αλλά από το διερμηνευτή. Η ΚΜΕ δεν εκτελεί άμεσα το πρόγραμμά μας αλλά εκτελεί το πρόγραμμα του διερμηνευτή, ο οποίος με τη σειρά του εκτελεί το πρόγραμμά μας.

Σε αντίθεση με την προσέγγιση της μεταγλώττισης, το πρόγραμμα δεν είναι αυτόνομο, δεν μπορεί να εκτελεστεί ανεξάρτητα, εφόσον απαιτείται η παρουσία του αντίστοιχου διερμηνευτή.

Η εκτέλεση ενός προγράμματος με αυτόν τον τρόπο είναι γενικά πιο αργή σε σχέση με τη μεταγλώττιση, επειδή κατά τη στιγμή της εκτέλεσης παρεμβάλλεται μεταξύ προγράμματος και ΚΜΕ το επίπεδο του διερμηνευτή. Συνήθως οι διερμηνευτές είναι διαλογικά προγράμματα.

7.6 Προγραμματίζοντας

Επειδή ο προγραμματισμός είναι κατά βάση μια δραστηριότητα επίλυσης προβλημάτων, στη συνέχεια θα παρουσιάσουμε μια συστηματική προσέγγιση στην επίλυση προβλημάτων με τη χρήση υπολογιστή, η οποία καλείται **μέθοδος ανάπτυξης λογισμικού.**

Τα βήματα αυτής της μεθόδου είναι:

α) Προσδιορισμός των απαιτήσεων του προβλήματος.
β) Ανάλυση του προβλήματος.
γ) Σχεδιασμός αλγόριθμου για την επίλυση του προβλήματος.
δ) Υλοποίηση του αλγόριθμου.
ε) Έλεγχος και επαλήθευση του τελικού προγράμματος.
στ) Συντήρηση και ενημέρωση του προγράμματος.
ζ) Τεκμηρίωση.

7.6.1 Καθορισμός προβλήματος

Ο προσδιορισμός των απαιτήσεων του προβλήματος είναι ένα στάδιο στο οποίο θα πρέπει να καθοριστεί με σαφήνεια το πρόβλημα και να υπάρξει ξεκάθαρη αντίληψη του είδους των απαιτήσεων για τη λύση του. Ο στόχος είναι να εξαλείψουμε τα ασήμαντα στοιχεία και να οδηγηθούμε στην ουσία του προβλήματος. Αυτό δεν είναι τόσο απλό, όσο ίσως φαίνεται. Μπορεί να σημαίνει ότι πρέπει να ζητήσουμε πρόσθετη πληροφορία από αυτόν ή αυτούς που έθεσαν το πρόβλημα.

7.6.2 Ανάλυση του προβλήματος

Η ανάλυση του προβλήματος περιλαμβάνει:

- τον καθορισμό των «εισόδων», δηλαδή των δεδομένων με τα οποία θα δουλέψουμε (inputs)
- τον καθορισμό των «εξόδων», δηλαδή των απαιτουμένων αποτελεσμάτων (outputs)
- την αναζήτηση άλλων απαιτήσεων ή περιορισμών.

Σε αυτό το στάδιο θα καθοριστεί και η μορφή στην οποία θέλουμε να δίνονται τα αποτελέσματα. Είναι επίσης χρήσιμο να προσδιορίσουμε τις παραμέτρους του προβλήματος και να καταγράψουμε πιθανές σχέσεις μεταξύ τους, ίσως με τη μορφή μαθηματικών σχέσεων.
Οποιαδήποτε παρανόηση είτε σε αυτό το επίπεδο είτε στο προηγούμενο μπορεί να οδηγήσει στην επίλυση άλλου προβλήματος από αυτό που στην πραγματικότητα απαιτείται.

7.6.3 Σχεδιασμός

Ο **αλγόριθμος** είναι μια σειρά από βήματα, τα οποία πρέπει να ακολουθήσουμε για να επιλύσουμε ένα συγκεκριμένο πρόβλημα. Τα βήματα πρέπει να είναι σαφή και να υπάρχει συγκεκριμένο σημείο περάτωσης της διαδικασίας.

Επόμενο βήμα είναι η σχεδίαση του αλγόριθμου για τη λύση του προβλήματος, δηλαδή των βημάτων που περιγράφουν τη λύση. Συχνά ο αλγόριθμος είναι το πιο δύσκολο τμήμα της μεθόδου.
Η λύση δεν χρειάζεται να περιγραφεί από την αρχή με κάθε λεπτομέρεια. Αντί γι' αυτό μπορεί να χρησιμοποιηθεί η «από πάνω προς τα κάτω προσέγγιση». Δηλαδή αρχικά να καταγραφούν τα βασικά βήματα ή υποπροβλήματα, που απαιτούνται για την επίλυση του προβλήματος, και στη συνέχεια να εστιάσουμε την προσοχή μας στο κάθε υποπρόβλημα χωριστά αναλύοντας περαιτέρω τα βήματα.

7.6.4 Υλοποίηση

Η υλοποίηση του αλγόριθμου αφορά τη συγγραφή του προγράμματος. Αυτό σημαίνει ότι κάθε βήμα του αλγόριθμου πρέπει να μετατραπεί σε μια ή περισσότερες εντολές κάποιας γλώσσας προγραμματισμού.
Ο δομημένος προγραμματισμός είναι μια πειθαρχημένη προσέγγιση στη διαδικασία συγγραφής, που οδηγεί στη δημιουργία προγραμμάτων, τα οποία όχι μόνο είναι εύκολο να διαβαστούν και να γίνουν κατανοητά, αλλά και να πε-

ριέχουν τα λιγότερα λάθη. Αυτό πρακτικά σημαίνει την εφαρμογή ορισμένων κανόνων που οδηγούν σε κώδικα ευανάγνωστο και εύκολο να συντηρηθεί.
Όπως έχουμε αναφέρει, η από πάνω προς τα κάτω λειτουργική αποσύνθεση είναι εκείνη που μας δίνει με ένα συστηματικό τρόπο τα βασικά τμήματα (modules) από τα οποία θα χτιστεί το τελικό πρόγραμμα.

7.6.5 Έλεγχος

Μια πολύ σοβαρή φάση της όλης ανάπτυξης του προγράμματος είναι αυτή του ελέγχου. Αφορά την εκτέλεση του προγράμματος για διάφορα -σωστά επιλεγμένα- σύνολα τιμών εισόδου και τον έλεγχο των αντίστοιχων εξόδων.

7.6.6 Συντήρηση

Ο όρος συντήρηση κατ' αρχάς είναι παράξενος, όταν αναφέρεται σε κάτι άυλο όπως είναι ένα πρόγραμμα. Το πρόγραμμα δεν χαλάει, δεν σκουριάζει, δεν φθείρεται και όμως χρειάζεται συντήρηση. Δύο είναι οι βασικοί λόγοι:

- για να διορθωθούν λάθη τα οποία εντοπίστηκαν, αφού το πρόγραμμα παραδόθηκε για χρήση, κατά τη διάρκεια δηλαδή χρήσης του προγράμματος
- για να γίνουν αλλαγές που προέκυψαν με την πάροδο του χρόνου, λόγω αλλαγών στο περιβάλλον χρήσης του προγράμματος. Για παράδειγμα, επειδή άλλαξε ο τρόπος υπολογισμού του Φ.Π.Α. για ένα πρόγραμμα τιμολόγησης.

Συναφής με το θέμα της συντήρησης του προγράμματος είναι η έννοια του αριθμού έκδοσης ή απλώς έκδοσης (version number, version).
Ας υποτεθεί ότι ένα πρόγραμμα με όνομα «`ΤΟ!πρόγραμμα`» παραδόθηκε για χρήση την 1/1/2000. Τους πρώτους μήνες της χρήσης του εντοπίστηκαν κάποια λάθη και την 1/8/2000 δόθηκε για χρήση το πρόγραμμα διορθωμένο. Θα πρέπει το διορθωμένο πρόγραμμα με κάποιο τρόπο να έχει διαφορετική ονομασία για να ξεχωρίζει από την αρχική έκδοση αλλά και συγχρόνως να φαίνεται ότι πρόκειται για το ίδιο πρόγραμμα. Κατά πάγια πρακτική στο χώρο του λογισμικού, αυτό επιτυγχάνεται με τη χρήση αρίθμησης για τον καθορισμό των εκδόσεων. Έτσι το αρχικό πρόγραμμα, ή για την ακρίβεια η αρχική έκδοση, πρέπει να λέγεται όχι απλώς «`ΤΟ!πρόγραμμα`», αλλά «`ΤΟ!πρόγραμμα έκδοση 1.0`». Η διορθωμένη τότε έκδοση μπορεί να λέγεται «`ΤΟ!πρόγραμμα έκδοση 1.1`». Μια επόμενη έκδοση, με πολλές αλλαγές, μπορεί να λέγεται «`ΤΟ!πρόγραμμα έκδοση 2.7`».
Η αρίθμηση γίνεται με παραπάνω από έναν αριθμούς, χωρισμένους με τελεία, και συνηθίζεται να είναι αύξουσα.
Όταν αγοράζουμε λογισμικό θα πρέπει να προσέχουμε ο αριθμός έκδοσης του προγράμματος να είναι ο πιο πρόσφατος.

7.6.7 Τεκμηρίωση

Η τεκμηρίωση αφορά το σύνολο των εγγράφων που προκύπτουν καθ' όλη τη διάρκεια της ανάπτυξης του προγράμματος και βοηθούν το έργο της συντή-

ρησής του αλλά και τη χρήση του. Αυτό το είδος της τεκμηρίωσης λέγεται **εξωτερική**. Συνήθως, καθένα από τα στάδια που αναφέραμε παράγει και ένα έγγραφο που αναφέρεται στα θέματα του κάθε σταδίου. Π.χ. στο στάδιο του σχεδιασμού το αντίστοιχο έγγραφο θα πρέπει να περιέχει την περιγραφή του αλγόριθμου που επιλέχθηκε για τη λύση του προβλήματος.

Η **εσωτερική** τεκμηρίωση αφορά οποιαδήποτε μορφή τεκμηρίωσης βρίσκεται καταχωρισμένη στον πηγαίο κώδικα του προγράμματος. Συνήθως αποτελείται από σχόλια σε κατάλληλα σημεία του προγράμματος -π.χ. πριν από κάθε υποπρόγραμμα αδρή περιγραφή της λειτουργίας του- και από σωστή και πειθαρχημένη επιλογή των ονομάτων των διαφόρων οντοτήτων του προγράμματος -μεταβλητές, υπορουτίνες, κλπ. Είναι χρησιμότερο από άποψη τεκμηρίωσης το υποπρόγραμμα υπολογισμού εμβαδού τριγώνου να ονομάζεται «emvado_trigonou» παρά «f» ή «et».

Παρ' όλο που σε ορισμένες περιπτώσεις δεν δίνεται η βαρύτητα που πρέπει στη σωστή τεκμηρίωση, θα πρέπει αυτή να θεωρείται εξίσου σημαντική με το ίδιο το πρόγραμμα.

Ανακεφαλαίωση

Κύριο διακριτικό γνώρισμα του υπολογιστή είναι η δυνατότητα προγραμματισμού του, με τη λειτουργία του να καθορίζεται από το πρόγραμμα που κάθε στιγμή εκτελεί. Ο προγραμματισμός, αποτελεί έναν από τους βασικούς τομείς της επιστήμης των υπολογιστών.

Τα πρώτα προγράμματα γράφονταν στη γλώσσα μηχανής του υπολογιστή. Στη συνέχεια, για να διευκολυνθεί η διαδικασία του προγραμματισμού, σχεδιάστηκαν σταδιακά γλώσσες προγραμματισμού οι οποίες είναι πιο κοντά στον τρόπο έκφρασης τού ανθρώπου. Ένα πρόγραμμα, λοιπόν, μπορεί να είναι γραμμένο σε:

- Γλώσσα μηχανής.
- Συμβολική γλώσσα.
- Γλώσσα υψηλού επιπέδου.

Από τον πρώτο προγραμματιζόμενο υπολογιστή έως σήμερα, έχει επινοηθεί ένας πολύ μεγάλος αριθμός γλωσσών προγραμματισμού. Μερικές από τις πιο γνωστές και επιτυχημένες εμπορικά είναι η FORTRAN, η COBOL, η C και η SQL. Ανάλογα με το σκοπό για τον οποίο σχεδιάστηκαν, οι **γλώσσες προγραμματισμού** διακρίνονται σε:

- Ειδικού σκοπού, επειδή σχεδιάστηκαν για να καλύπτουν τις απαιτήσεις ενός συγκεκριμένου φάσματος προβλημάτων.
- Γενικού σκοπού, αυτές που σχεδιάστηκαν για να καλύπτουν ένα γενικότερο φάσμα προβλημάτων.

Τα προγράμματα γράφονται είτε σε συμβολικές γλώσσες είτε σε γλώσσες υψηλού επιπέδου. Τη μετατροπή τους σε γλώσσα μηχανής αναλαμβάνουν κατά περίπτωση, οι συμβολομεταφραστές, οι μεταγλωττιστές και οι διερμηνευτές.

Η συγγραφή των προγραμμάτων έχει εξελιχθεί σε μια δομημένη διαδικασία που διέπεται από τις αρχές **δομημένου προγραμματισμού**, σύμφωνα με τις οποίες ένα πρόγραμμα κατασκευάζεται από συνεργαζόμενα δομικά στοιχεία, τα οποία υλοποιούνται με βάση καθορισμένους τύπους προγραμματιστικών δομών. Οι δομές αυτές είναι:

- Η διαδοχή.
- Η επιλογή.
- Η επανάληψη.

Ανάλογα με το πώς προσεγγίζεται ο προγραμματισμός του υπολογιστή, έχουν δημιουργηθεί διάφορα πρότυπα προγραμματισμού. Αυτά είναι:

- Ο διαδικαστικός προγραμματισμός.
- Ο αντικειμενοστρεφής προγραμματισμός.
- Ο λογικός προγραμματισμός.
- Ο συναρτησιακός προγραμματισμός.

Η κατασκευή ενός προγράμματος ακολουθεί ορισμένα βήματα. Τα πιο βασικά είναι:

α) Προσδιορισμός των απαιτήσεων του προβλήματος.
β) Ανάλυση του προβλήματος.
γ) Σχεδιασμός αλγόριθμου για την επίλυση του προβλήματος.
δ) Υλοποίηση του αλγόριθμου.
ε) Έλεγχος και επαλήθευση του τελικού προγράμματος.
στ) Συντήρηση και ενημέρωση του προγράμματος.