ΚΕΦΑΛΑΙΟ VI

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

ΚΕΦΑΛΑΙΟ VI

Εισαγωγή στον προγραμματισμό

Η επίλυση ενός προβλήματος με τον υπολογιστή περιλαμβάνει, όπως έχει ήδη αναφερθεί, τρία εξίσου σημαντικά στάδια.

- Τον ακριβή προσδιορισμό του προβλήματος.
- Την ανάπτυξη του αντίστοιχου αλγορίθμου.
- Την διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή.

Ο **προγραμματισμός** ασχολείται με το τρίτο αυτό στάδιο, τη δημιουργία του προγράμματος.
Πρόγραμμα καλείται το σύνολο των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος. Το πρόγραμμα, το οποίο γράφεται σε κάποια γλώσσα προγραμματισμού, δεν είναι απλά η υλοποίηση του αλγορίθμου, αλλά βασικό του στοιχείο είναι τα δεδομένα και οι δομές δεδομένων επί των οποίων επενεργεί. *Αναφέρθηκε ήδη ότι οι αλγόριθμοι και οι δομές δεδομένων είναι μια αδιάσπαστη ενότητα.*

Ο προγραμματισμός είναι αυτός που δίνει την εντύπωση ότι οι υπολογιστές είναι έξυπνες μηχανές που επιλύουν πολύπλοκα προβλήματα. Η εντύπωση αυτή είναι μια ψευδαίσθηση. Το μόνο που μπορεί να κάνει ένας υπολογιστής είναι στοιχειώδεις ενέργειες σε ακολουθίες των ψηφίων 0 και 1 (πρόσθεση, αποθήκευση σύγκριση και μετατόπιση), απλά αυτές τις ενέργειες τις εκτελεί με ασύλληπτη ταχύτητα.

ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Γλώσσες μηχανής (1^{ης} γενιάς)

Αρχικά για να μπορέσει ο υπολογιστής να εκτελέσει μια οποιαδήποτε λειτουργία, έπρεπε να δοθούν κατευθείαν οι κατάλληλες ακολουθίες από 0 και 1. Οι εντολές αυτές ήταν κατανοητές από τον υπολογιστή, αλλά εντελώς ακατανόητες από τον άνθρωπο. Ο τρόπος αυτός ήταν επίπονος και απαιτούσε βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή. Γι αυτό το λόγο υπήρχαν πολύ λίγοι με τις ικανότητες να υλοποιήσουν αυτό τον τρόπο.

Παράδειγμα:

1010100000001010
1000110000000001
….

Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου ή Assembly (2^{ης} γενιάς)

Από τα πρώτα χρόνια άρχισαν να γίνονται προσπάθειες για τη δημιουργία μιας συμβολικής γλώσσας, η οποία ενώ θα έχει έννοια για τον άνθρωπο, θα μετατρέπεται εσωτερικά από τους υπολογιστές στις αντίστοιχες ακολουθίες από 0 και 1. Για παράδειγμα η λέξη ADD είναι κατανοητή από τον άνθρωπο και απομνημονεύεται σχετικά εύκολα. Η εντολή αυτή θα μεταφραστεί από τον υπολογιστή σε μια ακολουθία δυαδικών ψηφίων και στη συνέχεια θα μπορεί να εκτελεστεί. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (assembler).

Οι γλώσσες αυτές, που συνεχίζουν να χρησιμοποιούνται μέχρι σήμερα, παραμένουν στενά συνδεδεμένες με την αρχιτεκτονική του κάθε υπολογιστή. Επίσης διαθέτουν μόνο εντολές απλών ενεργειών και όχι σύνθετων οδηγώντας έτσι σε μακροσκελή προγράμματα, που ήταν δύσκολο να γραφούν και να συντηρηθούν. Επίσης τα προγράμματα αυτά δεν μπορούν να μεταφερθούν σε διαφορετικό υπολογιστή. Οι γλώσσες αυτές ονομάζονται γλώσσες χαμηλού επιπέδου επειδή εξαρτώνται από την αρχιτεκτονική του υπολογιστή.

Γλώσσες υψηλού επιπέδου (3ης γενιάς)

Οι παραπάνω ανεπάρκειες των συμβολικών γλωσσών και η προσπάθεια για καλύτερη επικοινωνία ανθρώπου-μηχανής, οδήγησαν στα τέλη της δεκαετίας του 50 στην εμφάνιση των πρώτων γλωσσών προγραμματισμού υψηλού επιπέδου. Το πρόγραμμα που γράφεται σε μια γλώσσα υψηλού επιπέδου, μεταφράζεται από τον ίδιο τον υπολογιστή στις ακολουθίες των εντολών της μηχανής με τη βοήθεια ενός ειδικού προγράμματος, που ονομάζεται *μεταγλωττιστής*. Το ίδιο πρόγραμμα μπορεί να εκτελεστεί σε οποιοδήποτε άλλο υπολογιστή, αρκεί να υπάρχει ο αντίστοιχος μεταγλωττιστής. Παρακάτω αναφέρονται μερικές από τις κυριότερες γλώσσες προγραμματισμού.

- **FORTRAN**. Το 1957 η IBM ανέπτυξε την πρώτη γλώσσα υψηλού επιπέδου. Το όνομά της προέρχεται από τις λέξεις FORmula TRANslation. Η FORTRAN αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών εφαρμογών. Οι πιο γνωστές εκδόσεις της είναι η FORTRAN 77 και η FORTRAN 90. Η γλώσσα αυτή συνεχίζει να χρησιμοποιείται ακόμα και σήμερα για επιστημονικές εφαρμογές.

- **COBOL**. Το 1960 αναπτύχθηκε μια άλλη γλώσσα σταθμός στον προγραμματισμό, η γλώσσα COBOL. Όπως δηλώνει και το όνομά της (Common Business Oriented Language) είναι κατάλληλη για ανάπτυξη εμπορικών εφαρμογών, τομέας στον οποίο υστερούσε η FORTRAN.

- **ALGOL**. Μια από τις σημαντικότερες γλώσσες προγραμματισμού με ελάχιστη πρακτική εφαρμογή, αλλά που επηρέασε ιδιαίτερα τον προγραμματισμό στις επόμενες γλώσσες είναι η γλώσσα ALGOL (ALGOrithmic Language). Δημιουργήθηκε από ευρωπαίους επιστήμονες το 1960. Η ALGOL είχε σκοπό τη δημιουργία γενικής φύσης προγραμμάτων που να μη συνδέονται με συγκεκριμένες εφαρμογές όπως οι δύο προηγούμενες γλώσσες.

- **PL/1**. Στα μέσα της δεκαετίας του 1960 αναπτύχθηκε η γλώσσα PL/1 (Programming Language 1), η οποία προσπάθησε, χωρίς επιτυχία, να καλύψει όλους τους τομείς προγραμματισμού αντικαθιστώντας την COBOL και την FORTRAN.

- **PROLOG, LISP**. Στον χώρο της τεχνητής νοημοσύνης αναπτύχθηκαν δύο γλώσσες αρκετά διαφορετικές από τις άλλες. Στα μέσα του 60 αναπτύχθηκε στο MIT η γλώσσα LISP (LISt Processor), η οποία προσανατολίζεται σε χειρισμό λιστών από σύμβολα και στις αρχές του 70 η PROLOG (Programming in LOGic). Οι δύο αυτές γλώσσες χρησιμοποιούνται σε προβλήματα Τεχνητής νοημοσύνης (έμπειρα συστήματα, παιγνίδια, επεξεργασία φυσικών γλωσσών κλπ.).

- **BASIC.** Η γλώσσα BASIC (Beginner's All Purpose Symbolic Instruction Code) αρχικά αναπτύχθηκε, όπως δηλώνει και το όνομά της, ως γλώσσα για την εκπαίδευση αρχαρίων στον προγραμματισμό. Σχεδιάστηκε για να γράφονται σύντομα προγράμματα τα οποία εκτελούνται με τη βοήθεια διερμηνευτή (interpreter) και όχι μεταφραστή (Compiler) όπως οι υπόλοιπες γλώσσες. Η ανάπτυξη των μικροϋπολογιστών τύπου IBM βοήθησαν στην εξάπλωσή της. Ο Bill Gates (πρόεδρος της Microsoft) είχε σχεδιάσει τον πρώτο διερμηνευτή Basic σε 8K ROM για μικροϋπολογιστές. Η εταιρία του είχε ιδρυθεί για την εκμετάλλευση αυτού του προϊόντος.

⇨ **PASCAL.** Η γλώσσα Pascal, δημιούργημα του καθηγητή Niklaus Wirth ο οποίος της έδωσε το όνομα του μεγάλου μαθηματικού, φυσικού και θεολόγου Blaise Pascal, έφερε μεγάλες αλλαγές στον προγραμματισμό. Παρουσιάστηκε το 1970 και στηρίχτηκε πάνω στην ALGOL. Είναι μια γλώσσα γενικής χρήσης, κατάλληλη τόσο για την εκπαίδευση όσο και τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Η PASCAL γνώρισε και συνεχίζει να γνωρίζει μεγάλη εξάπλωση ειδικά στον χώρο τον μικροϋπολογιστών και αποτέλεσε τη βάση για την ανάπτυξη άλλων ισχυρότερων γλωσσών όπως η ADA και η Modula-2.

⇨ **C.** Η γλώσσα C αναπτύχθηκε στα εργαστήρια της BELL από τον Dennis Ritchie και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix. Είναι γλώσσα με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal για την ανάπτυξη δομημένων εφαρμογών, αλλά έχει και πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Έχει γνωρίσει τεράστια εξάπλωση και από πολλούς θεωρείται σαν η πιο «ισχυρή» γλώσσα υψηλού επιπέδου. Η C εξελίχθηκε στη γλώσσα C++, που είναι αντικειμενοστραφής.

⇨ **JAVA**. Η JAVA είναι μια αντικειμενοστραφής γλώσσα που αναπτύχθηκε από την εταιρία SUN με σκοπό την ανάπτυξη εφαρμογών, που θα εκτελούνται σε κατανεμημένα περιβάλλοντα, δηλαδή σε διαφορετικούς υπολογιστές οι οποίοι είναι συνδεδεμένοι στο διαδίκτυο. Τα προγράμματα αυτά μπορούν να εκτελούνται από διαφορετικούς υπολογιστές με διαφορετικά λειτουργικά συστήματα χωρίς αλλαγές.

Για την ανάπτυξη προγραμμάτων που να εκμεταλλεύονται τον γραφικό τρόπο επικοινωνίας χρήστη-υπολογιστή εμφανίστηκαν γλώσσες ή νέες εκδόσεις γλωσσών που υλοποιούσαν τις έννοιες του *οδηγούμενου από το γεγονός* προγραμματισμού και του *οπτικού προγραμματισμού*.

⇒ **Οπτικός προγραμματισμός.** Εννοούμε τη δυνατότητα να δημιουργούμε γραφικά ολόκληρο το περιβάλλον της εφαρμογής (πλαίσια διαλόγου, μενού κλπ.)

⇒ **Οδηγούμενος από το γεγονός προγραμματισμός.** Εννοούμε τη δυνατότητα να ενεργοποιούνται λειτουργίες του προγράμματος με την εκτέλεση ενός γεγονότος (π.χ. το κλικ του ποντικιού, την επιλογή μιας εντολής από το μενού κλπ).

Πλεονεκτήματα των γλωσσών υψηλού επιπέδου

⇒ Ο φυσικότερος και πιο «ανθρώπινος» τρόπος έκφρασης των προβλημάτων. Τα προγράμματα σε γλώσσα υψηλού επιπέδου είναι πιο κοντά στα προβλήματα που επιλύουν.

⇒ Η ανεξαρτησία από τον τύπο του υπολογιστή. Προγράμματα σε μία γλώσσα υψηλού επιπέδου μπορούν να εκτελεστούν σε οποιονδήποτε υπολογιστή με ελάχιστες ή καθόλου μετατροπές. Η δυνατότητα της μεταφερσιμότητας των προγραμμάτων είναι σημαντικό προσόν.

⇒ Η ευκολία της εκμάθησης και εκπαίδευσης ως απόρροια των προηγουμένων.

⇒ Η διόρθωση των λαθών και η συντήρηση προγραμμάτων σε γλώσσα υψηλού επιπέδου είναι πολύ ευκολότερο έργο.

Γλώσσες 4ης γενιάς

Οι γλώσσες υψηλού επιπέδου γνώρισαν μεγάλη επιτυχία λόγων των πλεονεκτημάτων που παρουσιάζουν. Ωστόσο απευθύνονται μόνο σε προγραμματιστές. Ο χρήστης δεν είχε τη δυνατότητα να επιφέρει αλλαγές σε κάποιο πρόγραμμα, προκειμένου να ικανοποιήσει μια νέα ανάγκη του. Σταδιακά όμως εμφανίστηκαν γλώσσες εφοδιασμένες με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης. Με αυτές ο απλός χρήστης μπορεί να επιλύει μόνος του μικρά προβλήματα εφαρμογών. Αυτή η αυξανόμενη τάση απόκρυψης της αρχιτεκτονικής του υλικού και της τεχνικής του προγραμματισμού οδήγησε στις γλώσσες 4ης γενιάς. Στις γλώσσες αυτές ο υπολογιστής έχει τη δυνατότητα, σχετικά εύκολα, να υποβάλλει ερωτήσεις στο σύστημα ή να αναπτύσσει εφαρμογές που ανακτούν πληροφορίες από βάσεις δεδομένων και να καθορίζει τον ακριβή τρόπο εμφάνισης αυτών των πληροφοριών. Τέτοιες γλώσσες είναι π.χ. η SQL.

Ταξινόμηση γλωσσών προγραμματισμού

⇒ **Διαδικασιακές (αλγοριθμικές) γλώσσες**. Είναι σχεδιασμένες για να επιτρέπουν την υλοποίηση αλγορίθμων. Εδώ ανήκουν οι περισσότερες γλώσσες. ALGOL, Pascal, C, Basic κλπ.

⇒ **Αντικειμενοστραφείς γλώσσες**. C++, JAVA κλπ.

⇒ **Συναρτησιακές γλώσσες**. LISP κλπ.

⇒ **Μη διαδικασιακές γλώσσες**. Λέγονται και γλώσσες πολύ υψηλού επιπέδου. Η PROLOG είναι το πιο χαρακτηριστικό παράδειγμα.

⇒ **Γλώσσες ερωτοαπαντήσεων**. SQL.

Ταξινόμηση με βάση τη περιοχή χρήσης.

⇒ **Γλώσσες γενικής χρήσης**. Θεωρητικά κάθε τέτοια γλώσσα μπορεί να χρησιμοποιηθεί για την επίλυση οποιουδήποτε προβλήματος. Στην πράξη όμως κάθε γλώσσα έχει σχεδιαστεί για να ανταποκρίνεται καλύτερα σε ορισμένη κατηγορία προβλημάτων.
- ✓ **Γλώσσες επιστημονικής κατεύθυνσης**. Π.χ. FORTRAN,
- ✓ **Γλώσσες εμπορικής κατεύθυνσης.** Π.χ. COBOL

Γλώσσες όπως οι PASCAL, BASIC, C τα καταφέρνουν εξίσου καλά και στους δύο τομείς.

⇒ **Γλώσσες προγραμματισμού συστημάτων**. Π.χ. η C.

⇒ **Γλώσσες τεχνητής νοημοσύνης**. Π.χ. LISP, PROLOG.

⇒ **Γλώσσες ειδικής χρήσης**.

Ποια είναι η καλύτερη γλώσσα προγραμματισμού; Η απάντηση στο ερώτημα αυτό δεν υπάρχει. Μπορούμε να πούμε με βεβαιότητα ότι μια γλώσσα προγραμματισμού που να είναι με αντικειμενικά κριτήρια καλύτερη από όλες τις άλλες δεν πρόκειται ποτέ να υπάρξει.
Οι γλώσσες προγραμματισμού, που είναι τεχνητές γλώσσες, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις φυσικές γλώσσες. Μια γλώσσα προσδιορίζεται από το αλφάβητό της, το λεξιλόγιό της, τη γραμματική της και τέλος τη σημασιολογία της.

Το αλφάβητο. Αλφάβητο μιας γλώσσας καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα. (π.χ. για την ελληνική γλώσσα αλφάβητο θεωρούνται όλα τα γράμματα από το α μέχρι το ω πεζά και κεφαλαία, οι αριθμοί από το 0 μέχρι το 9 και τα σημεία στίξης)

Το λεξιλόγιο. Το λεξιλόγιο αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία της αλφαβήτου, τις λέξεις που είναι δεκτές από τη γλώσσα. Για παράδειγμα στην ελληνική γλώσσα η ακολουθία των γραμμάτων ΑΒΓΑ είναι δεκτή αφού αποτελεί λέξη, αλλά η ακολουθία ΑΒΓΔΑ δεν αποτελεί λέξη της ελληνικής γλώσσας άρα δεν είναι δεκτή.

Η Γραμματική αποτελείται από το τυπικό ή σημασιολογικό και το συντακτικό.

⇒ **Τυπικό** είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μια λέξη είναι αποδεκτή. Για παράδειγμα στην ελληνική γλώσσα οι λέξεις γλώσσα γλώσσας είναι δεκτές, ενώ η λέξη γλώσσατ δεν είναι αποδεκτή.

⇒ **Συντακτικό** είναι το σύνολο των κανόνων που καθορίζει τη νομιμότητα της διάταξης και της σύνδεσης των λέξεων της γλώσσας για τη δημιουργία προτάσεων.

Η σημασιολογία είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατά επέκταση των εκφράσεων και προτάσεων που χρησιμοποιούνται σε μια γλώσσα. Στις γλώσσες προγραμματισμού ο δημιουργός της γλώσσας αποφασίζει τη σημασιολογία των λέξεων της γλώσσας.

Διαφορές φυσικών και τεχνητών γλωσσών

Μια βασική διαφορά είναι η συχνότητα εξέλιξης των φυσικών γλωσσών. Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις δημιουργούνται, κανόνες γραμματικής και σύνταξης αλλάζουν κλπ. Αντίθετα οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα. Ωστόσο συχνά βελτιώνονται και μεταβάλλονται από τους δημιουργούς τους, με σκοπό να διορθωθούν αδυναμίες ή να καλύψουν μεγαλύτερο εύρος εφαρμογών. Αλλάζουν σε επίπεδο διαλέκτου (π.χ. GW-Basic και QuickBasic) ή σε επίπεδο επέκτασης (π.χ. Basic και Visual Basic).

Τεχνικές Σχεδίασης Προγραμμάτων.

⇒ **Ιεραρχική σχεδίαση προγράμματος** (από επάνω προς τα κάτω). Σκοπός της ιεραρχικής σχεδίασης είναι η διάσπαση του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.

⇒ **Τμηματικός προγραμματισμός**. Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον τμηματικό προγραμματισμό. Μετά την ανάλυση του προβλήματος σε αντίστοιχα υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα, που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος.

⇒ **Δομημένος προγραμματισμός**. Ο δομημένος προγραμματισμός παρουσιάστηκε στα μέσα του 1960. Παλιότερα τα προγράμματα που δεν είχαν γραφεί «δομημένα» χρησιμοποιούσαν την εντολή **GOTO** και ήταν ιδιαίτερα μεγάλα και πολύπλοκα με αποτέλεσμα να ξοδεύεται πάρα πολύς χρόνος τόσο για τη συγγραφή τους όσο και στη διόρθωση και στη μετέπειτα συντήρησή

τους. Ο δομημένος προγραμματισμός είναι μια μεθοδολογία σύνταξης προγραμμάτων που έχει σκοπό να βοηθήσει τον προγραμματιστή στην ανάπτυξη σύνθετων προγραμμάτων, να μειώσει τα λάθη, να εξασφαλίσει την εύκολη κατανόηση των προγραμμάτων και να διευκολύνει τις διορθώσεις και τις αλλαγές σε αυτά. Ο δομημένος προγραμματισμός στηρίζεται στη χρήση τριών και μόνο στοιχειωδών λογικών δομών, τη δομή της ακολουθίας, τη δομή της επιλογής και τη δομή της επανάληψης. Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μόνο μια είσοδο και μόνο μια έξοδο.

Πλεονεκτήματα του δομημένου προγραμματισμού

- ⇨ Δημιουργία απλούστερων προγραμμάτων.
- ⇨ Άμεση μεταφορά των αλγορίθμων σε προγράμματα.
- ⇨ Διευκόλυνση ανάλυσης του προγράμματος σε τμήματα.
- ⇨ Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος.
- ⇨ Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.
- ⇨ Ευκολότερη διόρθωση και συντήρηση.

Αντικειμενοστραφής προγραμματισμός

Η ιδέα του αντικειμενοστραφούς προγραμματισμού ή της αντικειμενοστραφούς σχεδίασης έχει τις ρίζες της σε μια απλοϊκή ιδέα. Ένα πρόγραμμα περιγράφει «ενέργειες» (επεξεργασία) που εφαρμόζονται πάνω σε δεδομένα. Ένα βασικό ερώτημα που τίθεται είναι αν η φιλοσοφία, η δομή του προγράμματος είναι προτιμότερο να στηρίζεται στις «ενέργειες» ή στα δεδομένα; *Οι παραδοσιακές προγραμματιστικές τεχνικές εκλαμβάνουν ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τις «ενέργειες», ενώ η αντικειμενοστραφής σχεδίαση τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα*. Τα προγράμματα που δημιουργούνται έτσι είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα. Φυσικά ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

Παράλληλος προγραμματισμός

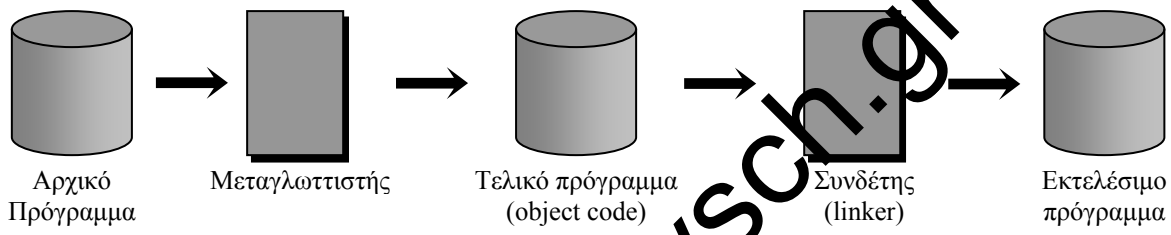
Σχετικά πρόσφατα εμφανίστηκαν υπολογιστές που ξεφεύγουν από την κλασική αρχιτεκτονική και διαθέτουν περισσότερους από έναν επεξεργαστές. Οι επεξεργαστές αυτοί μοιράζονται την ίδια μνήμη και λειτουργούν παράλληλα εκτελώντας διαφορετικές εντολές του ίδιου προγράμματος. Οι υπολογιστές αυτοί θεωρητικά μπορούν να πετύχουν ταχύτητες ασύλληπτες για τους τυπικούς υπολογιστές με έναν επεξεργαστή. Για να εκμεταλλευτούμε όμως την ταχύτητα που προσφέρει η αρχιτεκτονική τους πρέπει το πρόβλημα να διαιρεθεί σε τμήματα που εκτελούνται παράλληλα και στη συνέχεια να προγραμματιστεί σε ένα προγραμματιστικό περιβάλλον που να επιτρέπει τον παράλληλο προγραμματισμό.

Προγραμματιστικά περιβάλλοντα

Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού, πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής.

Η μετατροπή αυτή γίνεται με χρήση ειδικών μεταφραστικών προγραμμάτων που μετατρέπουν τις εντολές τις γλώσσας προγραμματισμού σε γλώσσα μηχανής. Υπάρχουν δύο κατηγορίες τέτοιων προγραμμάτων, οι μεταγλωττιστές (compilers) και οι διερμηνευτές (interpreters).

Ο μεταγλωττιστής (compiler) δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το αρχικό πρόγραμμα. Το αρχικό πρόγραμμα (σε γλώσσα υψηλού επιπέδου) καλείται **πηγαίο πρόγραμμα** (source code), ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται **αντικείμενο πρόγραμμα** (object). Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατανοητή από τον υπολογιστή, αλλά δεν μπορεί να εκτελεστεί. Πρέπει να συμπληρωθεί και να συνδεθεί με τα υπόλοιπα τμήματα του προγράμματος που είτε έχει γράψει ο προγραμματιστής είτε βρίσκονται στις βιβλιοθήκες (libraries) της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται **συνδέτης – φορτωτής** (linker – loader). Το αποτέλεσμα του συνδέτη είναι η παραγωγή του εκτελέσιμου προγράμματος (executable) το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή.



Ο διερμηνευτής (interpreter) διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για κάθε μία εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής. Δεν δημιουργείται δηλαδή εκτελέσιμο πρόγραμμα.

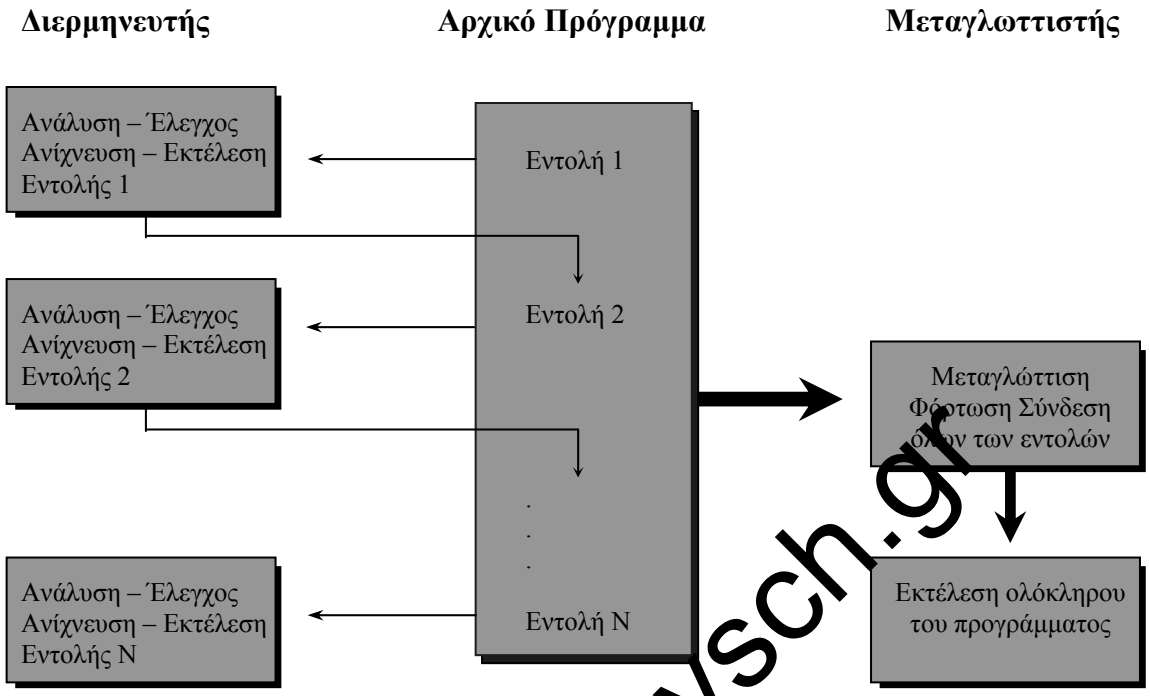
Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση που το αρχικό πρόγραμμα δεν περιέχει λάθη. Τα λάθη του προγράμματος είναι δύο ειδών, **λογικά** και **συντακτικά**. Τα λογικά εμφανίζονται μόνο στην εκτέλεση, ενώ τα συντακτικά στο στάδιο της μεταγλώττισης. Ο μεταγλωττιστής (και ο διερμηνευτής) ανιχνεύει τα λάθη και εμφανίζει κατάλληλα διαγνωστικά μηνύματα που μας βοηθούν να διορθώσουμε.

Μειονεκτήματα χρήσης μεταγλωττιστή	Μειονεκτήματα χρήσης διερμηνευτή
⇨ Προτού χρησιμοποιηθεί το πρόγραμμα, πρέπει να περάσει από τη φάση μεταγλώττισης και σύνδεσης (που είναι χρονοβόρα). Ενώ ο διερμηνευτής εκτελεί αμέσως το πρόγραμμα και άρα η διόρθωση γίνεται γρηγορότερα.	⇨ Η εκτέλεση του προγράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου προγράμματος που παράγει ο μεταγλωττιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με μεικτές υλοποιήσεις, όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημιουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.

Για την αρχική σύνταξη των προγραμμάτων και τη διόρθωσή τους στη συνέχεια χρησιμοποιείται ένα ειδικό πρόγραμμα που ονομάζεται **συντάκτης (editor)** και είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου, με δυνατότητες που διευκολύνουν την γρήγορη γραφή των εντολών των προγραμμάτων.

Ο συντάκτης, ο μεταγλωττιστής και ο συνδέτης είναι τα τρία προγράμματα που απαιτούνται (τουλάχιστον) για τη δημιουργία ενός προγράμματος. Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν αυτά τα προγράμματα με ενιαίο τρόπο.



Ασκήσεις

1) Ο παρακάτω αλγόριθμος αποτελεί τμήμα μη δομημένου προγράμματος. Αν γράψετε αλγόριθμο σχεδιασμένο με τις αρχές του δομημένου προγραμματισμού, που να εκτελεί τις ίδιες λειτουργίες.

```
ΑΡΧΗ
ΟΣΟ συνθήκη1 ΕΠΑΝΕΛΑΒΕ
        Εντολή2
        ΑΝ συνθήκη3 ΤΟΤΕ
                Εντολή4
                Πήγαινε στο ΤΕΛΟΣ
        ΑΛΛΙΩΣ
                Εντολή5
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ
```

2) Ο παρακάτω αλγόριθμος αποτελεί τμήμα μη δομημένου προγράμματος. Αν γράψετε αλγόριθμο σχεδιασμένο με τις αρχές του δομημένου προγραμματισμού, που να εκτελεί τις ίδιες λειτουργίες.

```
ΑΡΧΗ
        ΑΝ συνθήκη1 ΤΟΤΕ
                Εντολή1
                ΑΝ συνθήκη2 ΤΟΤΕ
                        Εντολή2
                        Εντολή3
                        Πήγαινε στην Εντολή5
                ΤΕΛΟΣ_ΑΝ
                Εντολή4
                Εντολή5
                Πήγαινε στην Αρχή
        ΤΕΛΟΣ_ΑΝ
        Εντολή3
ΤΕΛΟΣ
```