ΚΕΦΑΛΑΙΟ 6 – Εισαγωγή στον προγραμματισμό.

Ο **προγραμματισμός** είναι η διατύπωση του αλγορίθμου σε μορφή κατανοητή από τον Η/Υ ώστε να τον εκτελέσει («τρέξει» όπως λέμε στην ορολογία της Πληροφορικής).
Η διατύπωση γίνεται χρησιμοποιώντας μία γλώσσα προγραμματισμού.

Κατηγορίες γλωσσών προγραμματισμού.

Γλώσσα μηχανής	Το πρόγραμμα περιέχει εντολές που είναι σε δυαδική μορφή, άμεσα κατανοητή από τον Η/Υ (όχι όμως από τον άνθρωπο). Δηλαδή, το πρόγραμμα αποτελείται από ακολουθίες 0 και 1 π.χ. 10101000 00001010 11000000 00000001 …… **Πλεονεκτήματα** : ▪ Ταχύτατη εκτέλεση των εντολών. ▪ Δεν απαιτείται μεταφραστικό πρόγραμμα. **Μειονεκτήματα** : ▪ Το γράψιμο του προγράμματος είναι μία ιδιαίτερα επίπονη και χρονοβόρα διαδικασία. ▪ Απαιτείται βαθιά γνώση της αρχιτεκτονικής του Η/Υ. ▪ Το πρόγραμμα «τρέχει» μόνο στο συγκεκριμένο τύπο του Η/Υ.
Γλώσσες χαμηλού επιπέδου ή Συμβολικές γλώσσες	Οι εντολές που είναι σε μορφή 0 και 1 αντικαθίστανται από μνημονικά (συμβολικά) ονόματα. Για παράδειγμα, η εντολή 100001100 αντικαθίστανται από το ADD. Ένα δείγμα χρήσης θα ήταν : INDEX =$01 {βάλε στην INDEX την τιμή 1} ADD INDEX {πρόσθεσε την τιμή της INDEX στον συσσωρευτή} LDA #10 {φόρτωσε στο συσσωρευτή την τιμή 10} CLA {καθάρισε το συσσωρευτή} …….. **Πλεονεκτήματα** : ▪ Ταχύτατη εκτέλεση των εντολών. ▪ Η μορφή του προγράμματος είναι καλύτερα κατανοητή από τον άνθρωπο σε σχέση με τη γλώσσα μηχανής. **Μειονεκτήματα** : ▪ Η αντιστοιχία 1 προς 1 με τις εντολές της γλώσσας παρέμενε. ▪ Απαιτείται η χρήση ενός μεταφραστικού προγράμματος ώστε οι συμβολικές εντολές να μετατραπούν στις αντίστοιχες δυαδικές. Το ειδικό αυτό πρόγραμμα ονομάζεται ***συμβολομεταφραστής (assembler).*** ▪ Το γράψιμο του προγράμματος εξακολουθεί να είναι μία ιδιαίτερα επίπονη και χρονοβόρα διαδικασία. ▪ Απαιτείται βαθιά γνώση της αρχιτεκτονικής του Η/Υ. ▪ Το πρόγραμμα «τρέχει» μόνο στο συγκεκριμένο τύπο του Η/Υ.
Γλώσσες υψηλού επιπέδου	Λέγονται έτσι διότι τα προγράμματα διατυπωμένα σε μία τέτοια γλώσσα είναι άμεσα κατανοητά από τον άνθρωπο (αλλά όχι από τον Η/Υ) αφού χρησιμοποιείται μία γλώσσα που είναι αρκετά περιγραφική όπως μία φυσική γλώσσα. Παράδειγμα, INPUT "Δώσε την τελική τιμή" ; N SUM = 0 For INDEX = 1 to N SUM = SUM + INDEX Next

	Πλεονεκτήματα : ▪ Η μορφή του προγράμματος είναι εύκολα κατανοητή από τον άνθρωπο σε σχέση με τη γλώσσα μηχανής ή τη συμβολική γλώσσα. ▪ Το γράψιμο του προγράμματος δεν είναι πλέον μία ιδιαίτερα επίπονη και χρονοβόρα διαδικασία όπως συμβαίνει με τη γλώσσα μηχανής ή τη συμβολική γλώσσα. ▪ Δεν απαιτείται σχεδόν καμία γνώση της αρχιτεκτονικής του Η/Υ. Συνεπώς, είναι ανεξάρτητα από την αρχιτεκτονική του Η/Υ. ▪ Το πρόγραμμα «τρέχει» σε όλους τους τύπους Η/Υ αρκεί να υπάρχει το κατάλληλο μεταφραστικό πρόγραμμα. Συνεπώς ένα χαρακτηριστικό τους είναι η ***μεταφερσιμότητα***, δηλαδή ένα πρόγραμμα υψηλού επιπέδου να εκτελείται, με ελάχιστες μετατροπές, σε πολλούς τύπους Η/Υ. ▪ Η εκμάθηση της γλώσσας είναι εύκολη. ▪ Η διόρθωση λαθών και η συντήρηση των προγραμμάτων είναι ευκολότερη. **Μειονεκτήματα** : ▪ Απαιτείται η χρήση ενός μεταφραστικού προγράμματος ώστε οι εντολές να μετατραπούν σε πολλές δυαδικές εντολές (δεν έχουμε εδώ αντιστοιχία 1 προς 1). Έχουμε δύο ειδών μεταφραστικά προγράμματα: τους ***μεταγλωττιστές (compilers)*** και τους ***διερμηνείς (interpreters)***. ▪ Το πρόγραμμα «τρέχει» πιο αργά σε σχέση με τα προγράμματα των συμβολικών γλωσσών ή της γλώσσας μηχανής.

Κατηγοριοποίηση γλωσσών υψηλού επιπέδου

- **Διαδικασιακές ή αλγοριθμικές γλώσσες (Procedural)** : λέγονται έτσι διότι επιτρέπουν την εύκολη υλοποίηση αλγορίθμων π.χ. Pascal, Basic.
- **Αντικειμενοστραφείς γλώσσες (object – oriented)** π.χ. C++
- **Συναρτησιακές γλώσσες (functional)** π.χ. LISP
- **Μη-διαδικασιακές γλώσσες** π.χ. PROLOG
- **Γλώσσες ερωταπαντήσεων (Query languages) ή 4ης γενιάς** π.χ. SQL

Μία άλλη κατηγοριοποίηση των γλωσσών υψηλού επιπέδου είναι η εξής:

- **Γλώσσες γενικής χρήσης** : Σκοπός τους είναι να επιλύουν πάσης φύσεως προβλήματα (αριθμητικά, εμπορικά, επιστημονικά). Τέτοιες είναι η Basic, Pascal. Μερικές γλώσσες, όμως, έχουν δημιουργηθεί αποκλειστικά για να επιλύουν ευκολότερα συγκεκριμένους τύπους προβλημάτων όπως :
  - ***Γλώσσες επιστημονικής κατεύθυνσης*** : π.χ. FORTRAN
  - ***Γλώσσες εμπορικής κατεύθυνσης*** : π.χ. COBOL.
- **Γλώσσες προγραμματισμού συστημάτων** π.χ. C
- **Γλώσσες τεχνητής νοημοσύνης** π.χ. LISP, PROLOG.
- **Γλώσσες ειδικής χρήσης**. Σκοπός τους είναι να επιλύουν εδικού τύπου προβλήματα όπως διαχείριση Βάσεων Δεδομένων κ.α. π.χ. SQL .

Διαδικασιακές γλώσσες κατά χρονολογική σειρά εμφάνισης.

FORTRAN (1957)	Κατάλληλη για επίλυση επιστημονικών προβλημάτων (αριθμητικές εφαρμογές).
COBOL (1960)	Κατάλληλη για επίλυση εμπορικών προβλημάτων (εφαρμογές μισθοδοσίας, λογιστικές κλπ).
ALGOL (1960)	Δημιουργήθηκε με σκοπό την ανάπτυξη προγραμμάτων κυρίως για επιστημονικές εφαρμογές ως ανταγωνιστική της Fortran. Δεν χρησιμοποιήθηκε, όμως, ευρέως στην πράξη. Πρωτοπαρουσίασε τις αρχές του δομημένου προγραμματισμού.
PL/1 (μέσα '60)	Προσπάθησε να συνδυάσει τις δυνατότητες των γλωσσών προσανατολισμένων για εμπορικές και επιστημονικές εφαρμογές χωρίς όμως να γνωρίσει επιτυχία.
BASIC (μέσα '60)	Δημιουργήθηκε με σκοπό την εκπαίδευση των αρχάριων στον προγραμματισμό. Σκοπός της BASIC είναι να γράφονται μικρά προγράμματα που κατόπιν εκτελούνται με τη βοήθεια διερμηνέα (interpreter). Στις μέρες μας αποτελεί μία πανίσχυρη, γενικής χρήσης γλώσσα.
PASCAL (1970)	Γλώσσα γενικής χρήσης. Στηρίχθηκε στην ALGOL. Είναι η καταλληλότερη γλώσσα για να μάθει κάποιος δομημένο προγραμματισμό.
C (αρχές '70)	Περιέχει αρκετά κοινά χαρακτηριστικά με τη Pascal για την ανάπτυξη δομημένων εφαρμογών αλλά παράλληλα ενσωματώνει και χαρακτηριστικά γλώσσας χαμηλού επιπέδου. Θεωρείται κατάλληλη για την ανάπτυξη λειτουργικών συστημάτων (π.χ. Unix)
Ada (1979)	Γλώσσα γενικής χρήσης που δίνει έμφαση στο θέμα της αξιοπιστίας των προγραμμάτων. Γι' αυτό χρησιμοποιήθηκε πρωτίστως για στρατιωτικές εφαρμογές.

Αντικειμενοστραφείς γλώσσες κατά χρονολογική σειρά εμφάνισης.

Smalltalk (αρχές ’80)	Η πρώτη αντικειμενοστραφής γλώσσα με ολοκληρωμένο μάλιστα περιβάλλον ανάπτυξης προγραμμάτων.
C++ (τέλη ’80)	Αποτελεί μία μετεξέλιξη της C στο χώρο του αντικειμενοστραφούς προγραμματισμού και χρησιμοποιείται αρκετά στην ανάπτυξη λειτουργικών συστημάτων (π.χ. Windows) αλλά και άλλου τύπου εφαρμογών. Θεωρείται σήμερα μία κορυφαία γλώσσα.
JAVA (μέσα ’90)	Γλώσσα ειδικά σχεδιασμένη για την ανάπτυξη εφαρμογών στο Internet. Σκοπός της είναι να γράφονται προγράμματα που θα εκτελούνται, χωρίς μετατροπές, σε Η/Υ με διαφορετικά λειτουργικά συστήματα. Περιέχει αρκετά στοιχεία από τη C++.
C# (2002)	Δημιουργήθηκε ως ανταγωνιστική της JAVA. Σκοπός της είναι να συνδυάσει την ευχρηστία της Basic και τη δυναμική της C++ για την ανάπτυξη εφαρμογών που θα εκτελούνται σε Η/Υ με διαφορετικά λειτουργικά συστήματα.

Συναρτησιακές γλώσσες κατά χρονολογική σειρά εμφάνισης.

LISP (μέσα ’60)	Δημιουργήθηκε για την ανάπτυξη προγραμμάτων στο χώρο της τεχνητής νοημοσύνης.

Μη – διαδικασιακές γλώσσες κατά χρονολογική σειρά εμφάνισης.

PROLOG (αρχές ’70)	Δημιουργήθηκε για την ανάπτυξη προγραμμάτων στο χώρο της τεχνητής νοημοσύνης.

Γλώσσες 4ης γενιάς ή ερωταπαντήσεων

SQL	Δεν απευθύνεται μόνο σε προγραμματιστές αλλά και χρήστες. Ο χρήστης μπορεί, σχετικά εύκολα, να υποβάλει ερωτήσεις στο σύστημα ή να αναζητά πληροφορίες από μία Βάση Δεδομένων. Παράδειγμα, SELECT Επώνυμο, Όνομα FROM Μαθητές WHERE Τάξη = ‘’Γ2’’ Η παραπάνω πρόταση της γλώσσας SQL θα κάνει μία αναζήτηση στη βάση δεδομένων των μαθητών και θα εμφανίσει τα ονοματεπώνυμα των μαθητών του Γ2 μόνο.

Από τι εξαρτάται η επιλογή μίας γλώσσας προγραμματισμού;

- Από το είδος του προβλήματος (εμπορικό, αριθμητικό κ.α.)
- Από τον Η/Υ στον οποίο θα εκτελεστεί το πρόγραμμα.
- Από τη διαθέσιμη γλώσσα ή προγραμματιστικό περιβάλλον στο οποίο θα αναπτυχθεί το πρόγραμμα.
- Από τις γνώσεις και την εμπειρία του προγραμματιστή.

Με την ευρεία διάδοση των γραφικών περιβαλλόντων επικοινωνίας (π.χ. Windows, MacOS κλπ) δημιουργήθηκαν παραλλαγές κάποιων γλωσσών που απευθύνονται σε αυτά. Τέτοιες είναι η Visual Basic, Visual C++, Delphi (Visual Pascal), C# κ.α. Αυτές οι γλώσσες ακολουθούν τη

φιλοσοφία του *οπτικού* και του *καθοδηγούμενου-από-γεγονότα προγραμματισμού* χωρίς να απορρίπτουν τις αρχές του δομημένου προγραμματισμού.

Φυσικές και τεχνητές γλώσσες

Οι γλώσσες προγραμματισμού είναι τεχνητές γλώσσες που απευθύνονται σε ανθρώπους που επιθυμούν να επικοινωνήσουν με τον Η/Υ.

Κάθε γλώσσα προγραμματισμού προσδιορίζεται από :
- Το αλφάβητό της
- Το λεξιλόγιό της
- Τη γραμματική της
- Τη σημασιολογία της (Semantics)

Αλφάβητο γλώσσας
Ως αλφάβητο ορίζουμε το σύνολο των αποδεκτών χαρακτήρων της γλώσσας. Από τους χαρακτήρες αυτούς σχηματίζονται οι λέξεις της γλώσσας.
Σε μία φυσική γλώσσα, όπως τα Ελληνικά, οι αποδεκτοί χαρακτήρες είναι τα γράμματα Α-Ω (κεφαλαία και πεζά), τα ψηφία 0-9 και τα σημεία στίξης.

Λεξιλόγιο γλώσσας
Το λεξιλόγιο μίας γλώσσας περιλαμβάνει όλες τις έγκυρες και αποδεκτές λέξεις. Στην ουσία, είναι ένα υποσύνολο από όλες τις δυνατές ακολουθίες που μπορούμε να σχηματίσουμε από τα στοιχεία του αλφαβήτου.
Σε μία φυσική γλώσσα, όπως τα Ελληνικά, η λέξη ΔΙΑΒΑΖΩ είναι αποδεκτή ενώ η λέξη ΖΩΒΑΓΩ όχι.

Η Γραμματική της γλώσσας.
Η γραμματική περιλαμβάνει το **τυπολογικό** και το **συντακτικό**.
Το **τυπολογικό** ορίζει τους κανόνες σύμφωνα με του οποίους μία λέξη θα είναι αποδεκτή.
Για παράδειγμα, στην ελληνική γλώσσα για τη λέξη ‘’δίνω’’, αποδεκτές μορφές είναι και το ‘’δίνεις’’, ‘’δίνουν’’ αλλά όχι το ‘’δίνουτ’’.
Το **συντακτικό** είναι ένα σύνολο κανόνων που ορίζει το πώς πρέπει να σχηματίζονται οι προτάσεις από τις λέξεις της γλώσσας ώστε οι προτάσεις αυτές να είναι έγκυρες και αποδεκτές.
Σε μία γλώσσα προγραμματισμού αυτό που ενδιαφέρει είναι η σωστή σύνταξη των εντολών.

Σημασιολογία της γλώσσας.
Είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και προτάσεων της γλώσσας.
Σε μία γλώσσα προγραμματισμού αυτό καθορίζεται από το δημιουργό της ενώ σε μία φυσική γλώσσα από αυτόν που εκφέρει την πρόταση.

Διαφορές μεταξύ φυσικών και τεχνητών γλωσσών

Φυσικές	Τεχνητές
Χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπων.	Χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπου και Η/Υ.
Έχουν μεγάλες δυνατότητες εξέλιξης. Νέες λέξεις μπορεί να εισαχθούν, κανόνες γραμματικής και σύνταξης να αλλάξουν κλπ.	Οι δυνατότητες εξέλιξης είναι περιορισμένες. Τις περισσότερες φορές η εξέλιξη αυτή αφορά την επέκταση του ρεπερτορίου των εντολών της γλώσσας (π.χ. Basic και Visual Basic).

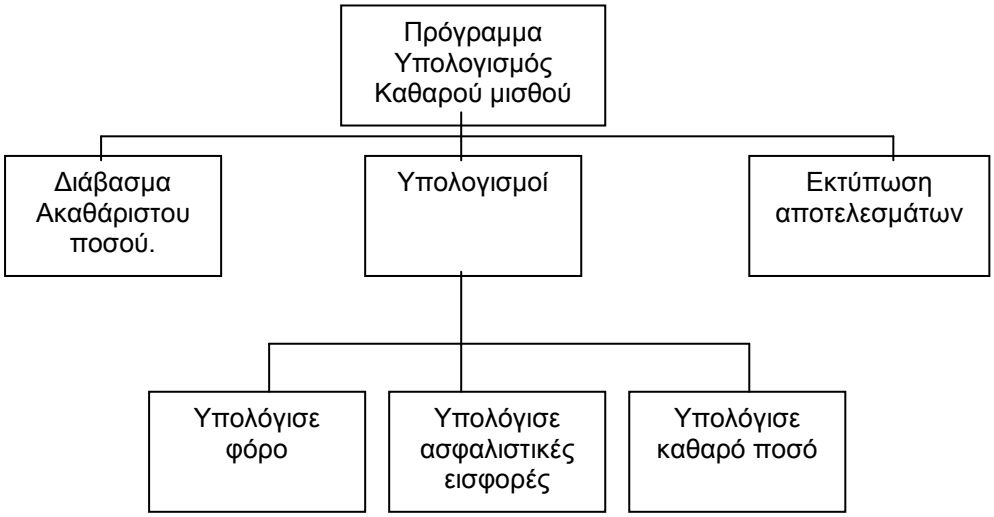
Τεχνικές σχεδίασης προγραμμάτων

Δεν αρκεί κάποιος να γνωρίζει απλά μία γλώσσα προγραμματισμού. Θα πρέπει να ακολουθήσει και μία τεχνική σχεδίασης του προγράμματός του.
Για την σύνταξη σωστών, κατανοητών και εύκολα συντηρήσιμων προγραμμάτων ακολουθήθηκαν διάφορες μεθοδολογίες ανάπτυξης που παρουσιάζονται παρακάτω:

- **Ιεραρχική σχεδίαση ή Top-down σχεδίαση**.

Η τεχνική αυτή συνιστά στον καθορισμό των βασικών λειτουργιών του προγράμματος σε ανώτερο επίπεδο και στη συνέχεια τη διάσπαση καθεμιάς σε μικρότερες και απλούστερες μέχρι του σημείου να είναι τόσο απλές που μπορούν να επιλυθούν άμεσα.
Συνεπώς, σκοπός της ιεραρχικής σχεδίασης είναι η διάσπαση του προβλήματος σε μικρότερα κι απλούστερα υπο-προβλήματα τα οποία είναι ευκολότερο να επιλυθούν.
Για την Top-down σχεδίαση χρησιμοποιούμε το ***ιεραρχικό διάγραμμα***.

Παράδειγμα : Πρόγραμμα Υπολογισμός Καθαρού Μισθού



- **Τμηματικός προγραμματισμός**

Η ιεραρχική σχεδίαση υλοποιείται με τον τμηματικό προγραμματισμό.
Μετά την ανάλυση του προβλήματος σε μικρότερα και απλούστερα υπο-προβλήματα , κάθε υπο-πρόβλημα αποτελεί μία ξεχωριστή και ανεξάρτητη **ενότητα** (module). Τώρα, για κάθε ενότητα θα γραφτεί το κατάλληλο πρόγραμμα ή τμήμα προγράμματος.

- **Δομημένος προγραμματισμός**

Είναι η μεθοδολογία που έχει επικρατήσει σήμερα. Εμπεριέχει τις αρχές τις ιεραρχικής σχεδίασης και του τμηματικού προγραμματισμού. Επιπλέον, αναφέρει ότι για τη δημιουργία σωστών προγραμμάτων να χρησιμοποιούμε μόνο τις 3 στοιχειώδες δομές : Ακολουθίας, Επιλογής και Επανάληψης.
΄΄Όλα τα προγράμματα, οσοδήποτε μεγέθους, μπορούν να γραφτούν στηριζόμενα στη χρήση μόνο αυτών των δομών, αποφεύγοντας πλήρως τη χρήση τη δομής GOTO. Επίσης, κάθε ενότητα προγράμματος έχει μόνο μία είσοδο και μόνο μία έξοδο.

Πλεονεκτήματα του δομημένου προγραμματισμού.

- Άμεση υλοποίηση των αλγορίθμων σε πρόγραμμα.
- Διευκόλυνση της ανάλυσης του προγράμματος σε τμήματα (ενότητες) . Το κάθε τμήμα μπορεί να γραφτεί από διαφορετικές ομάδες προγραμματιστών.
- Ευκολότερη και συντομότερη ανάπτυξη προγραμμάτων.
- Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος.
- Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.
- Ευκολότερη διόρθωση και συντήρηση του προγράμματος.

Αντικειμενοστραφής προγραμματισμός (object-oriented programming)
Η φιλοσοφία του αντικειμενοστραφούς προγραμματισμού είναι η εξής: Να θεωρήσουμε τα δεδομένα και τις αποδεκτές ενέργειες που γίνονται πάνω σε αυτά ως ένα ενιαίο αντικείμενο (object). Αυτό το αντικείμενο θα μπορεί να χρησιμοποιηθεί πολύ εύκολα οπουδήποτε αλλού, δηλαδή θα είναι *επαναχρησιμοποιήσιμο*.
Για παράδειγμα, θα μπορούσαμε να θεωρήσουμε μία στοίβα ως ένα αντικείμενο : Ένας πίνακας Π[1:20] που θα δέχεται ακεραίους. Οι μόνες ενέργειες που θα ήταν επιτρεπτές θα ήταν η ώθηση (push) ενός στοιχείου στη στοίβα και η αφαίρεση (pop) ενός στοιχείου από τη στοίβα. Αυτό το αντικείμενο θα μπορούσαμε να το δώσουμε και σε όποιον άλλο το είχε ανάγκη.
Ο αντικειμενοστραφής προγραμματισμός ακολουθεί τις αρχές του δομημένου προγραμματισμού.

Οπτικός προγραμματισμός.
Είναι η δυνατότητα να δημιουργούμε, με γραφικό τρόπο, ολόκληρο το περιβάλλον της εφαρμογής όπως για παράδειγμα τα μενού και τα πλαίσια διαλόγου και άλλα παράθυρα της εφαρμογής.
Ο οπτικός προγραμματισμός εκμεταλλεύεται τις δυνατότητες των γραφικών περιβαλλόντων επικοινωνίας (π.χ Windows, MacOs κλπ).

Καθοδηγούμενος από γεγονότα προγραμματισμός
Είναι η δυνατότητα να εκτελούνται οι διάφορες λειτουργίες του προγράμματος με την ενεργοποίηση ενός γεγονότος. Για παράδειγμα, αν κάνουμε κλικ σε κάποια εντολή ενός μενού ή σε κάποιο κουμπί σε ένα παράθυρο της εφαρμογής τότε θα εκτελεστεί μία λειτουργία..

Τα σύγχρονα προγραμματιστικά περιβάλλοντα είναι κτισμένα πάνω στις αρχές του οπτικού και καθοδηγούμενου από γεγονότα προγραμματισμού.

Παράλληλος προγραμματισμός
Σήμερα υπάρχουν μεγάλοι Η/Υ που διαθέτουν στο εσωτερικό τους πολλούς επεξεργαστές. Οι επεξεργαστές αυτοί μοιράζονται την ίδια μνήμη και λειτουργούν παράλληλα. Έτσι, την ίδια χρονική στιγμή, μπορούν να εκτελούνται διαφορετικές εντολές του προγράμματος.
Για να εκμεταλλευτούμε αυτήν την ιδιαίτερη ισχύ των Η/Υ θα πρέπει τα προγράμματα να είναι φτιαγμένα με τέτοιον τρόπο ώστε διαφορετικά τμήματά του να εκτελούνται παράλληλα. Για τον σκοπό αυτό έχουν αναπτυχθεί ιδιαίτερες γλώσσες προγραμματισμού όπως η OCCAM

Προγραμματιστικά περιβάλλοντα
Ένα πρόγραμμα που φτιάχνεται σε μία γλώσσα υψηλού επιπέδου δεν είναι άμεσα κατανοητό από τον Η/Υ. Θα πρέπει να μεταφραστεί σε ισοδύναμο πρόγραμμα σε γλώσσα μηχανής (δυαδική μορφή). Την διαδικασία μετάφρασης την πραγματοποιούν τα μεταφραστικά προγράμματα. Είναι δύο ειδών:

- Μεταγλωττιστές (Compilers)
- Διερμηνευτές (Interpreters)

Πηγαίο πρόγραμμα (Source program): Το πρόγραμμα που είναι φτιαγμένο σε γλώσσα υψηλού επιπέδου.

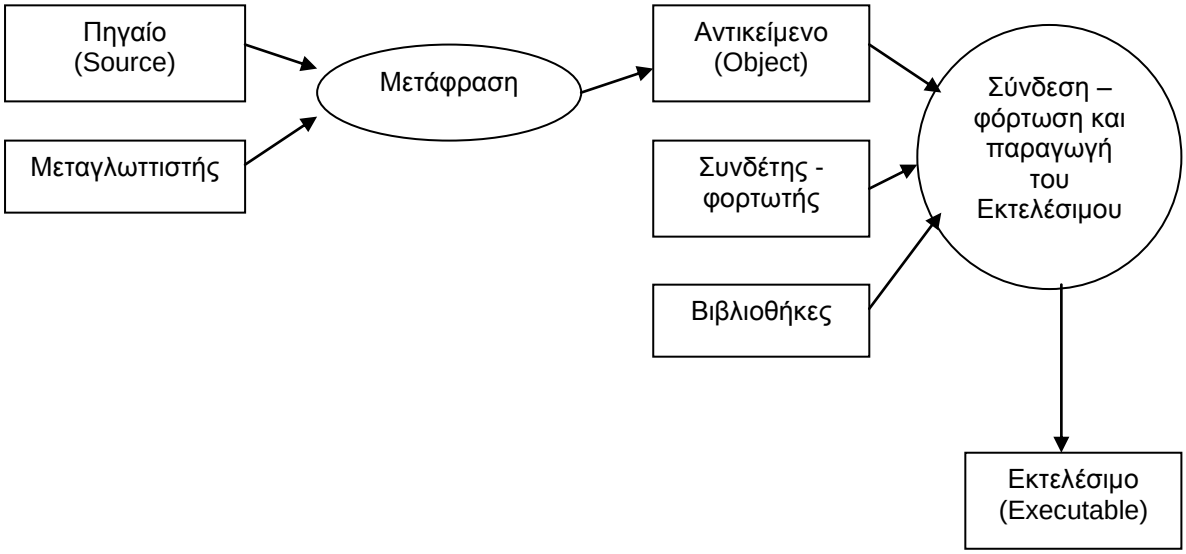
Αντικείμενο πρόγραμμα (Object program) : Το πρόγραμμα που είναι μεταφρασμένο σε γλώσσα μηχανής αλλά όχι άμεσα εκτελέσιμο.

Εκτελέσιμο πρόγραμμα (Executable program) : Το πρόγραμμα σε γλώσσα μηχανής που είναι έτοιμο πλέον να εκτελεστεί.

Μεταγλωττιστής : Παίρνει, ως είσοδο, το πηγαίο πρόγραμμα και αναλαμβάνει να το μεταφράσει εξ' ολοκλήρου παράγοντας το αντικείμενο πρόγραμμα.

Διερμηνευτής : Παίρνει, ως είσοδο, μία-μία εντολή του πηγαίου, την μεταφράζει και την εκτελεί αμέσως. Η λειτουργία του μοιάζει με τον άνθρωπο-διερμηνέα που μεταφράζει επί τόπου κάθε πρόταση.

Η διαδικασία μεταγλώττισης και σύνδεσης προγράμματος φαίνεται παρακάτω:



Βιβλιοθήκες (Libraries) : Έτοιμες ενότητες (modules) αντικείμενου προγράμματος της γλώσσας, απαραίτητες για την παραγωγή του εκτελέσιμου προγράμματος.

Συνδέτης – Φορτωτής (Linker-Loader) : Ειδικό πρόγραμμα που αναλαμβάνει να συνδέσει το αντικείμενο πρόγραμμα με τις βιβλιοθήκες και να παράγει το εκτελέσιμο.

Κατά τη δημιουργία ενός προγράμματος σχεδόν πάντα ενυπάρχουν λάθη. Τα λάθη τα χωρίζουμε σε δύο κατηγορίες :

- Συντακτικά λάθη
- Λογικά λάθη

Τα **συντακτικά λάθη** ανιχνεύονται κατά την διαδικασία της μεταγλώττισης ή διερμήνευσης. Αφορούν παραβιάσεις του τυπολογικού και συντακτικού της γλώσσας. (π.χ. μία εντολή έχει γραφτεί συντακτικά λάθος). Στην περίπτωση αυτή ο προγραμματιστής πρέπει να επιστρέψει στο πηγαίο πρόγραμμα, να διορθώσει τα λάθη και να το ξαναυποβάλλει για μεταγλώττιση.
Τα **λογικά λάθη** είναι και τα πλέον δύσκολα στην ανίχνευσή τους. Έχουν να κάνουν με σφάλματα στη λογική επίλυσης του προβλήματος ή λανθασμένης διατύπωσης του αλγορίθμου (π.χ. το πρόγραμμα παράγει άλλα αποτελέσματα κι όχι τα ζητούμενα!).

Για την σύνταξη των προγραμμάτων χρησιμοποιούμε ένα ειδικό πρόγραμμα που ονομάζεται **συντάκτης (editor).** Μοιάζει με επεξεργαστή κειμένου με επιπλέον δυνατότητες που διευκολύνουν την γρήγορη σύνταξη των πηγαίων προγραμμάτων.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα περιέχουν οπωσδήποτε 3 ειδών προγράμματα : Τον συντάκτη, το μεταγλωττιστή και το συνδέτη - φορτωτή. Πέραν αυτών, περιέχουν όλα εκείνα τα εργαλεία που διευκολύνουν την εύκολη, ταχύτατη σύνταξη, διόρθωση και συντήρηση των προγραμμάτων γι' αυτό και πολλές φορές ονομάζονται ως **RAD περιβάλλοντα** (**R**apid **A**pplication **D**evelopment).
Όλα αυτά παρέχονται με έναν ενιαίο και λειτουργικό τρόπο στον προγραμματιστή.