Εισαγωγή στον προγραμματισμό

Η έννοια του προγράμματος

Ο προγραμματισμός ασχολείται με τη δημιουργία του προγράμματος, δηλαδή του συνόλου εντολών που πρέπει να δοθούν στον υπολογιστή ώστε να υλοποιηθεί ένας αλγόριθμος για την επίλυση ενός προβλήματος. Το πρόγραμμα, το οποίο γράφεται σε κάποια γλώσσα προγραμματισμού, δεν είναι απλά η υλοποίηση του αλγόριθμου, αλλά βασικό στοιχείο του είναι τα δεδομένα και οι δομές δεδομένων επί των οποίων ενεργεί. Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου με τη μηχανή.

Ο υπολογιστής είναι μια μηχανή που καταλαβαίνει μόνο δυο καταστάσεις οι οποίες αντιπροσωπεύονται με τους αριθμούς μηδέν και ένα, τα ψηφία του δυαδικού συστήματος. Ο υπολογιστής μπορεί απλά να αποθηκεύει στη μνήμη τις ακολουθίες δυαδικών ψηφίων, να τις ανακτά, να κάνει στοιχειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει, με ασύλληπτη ταχύτητα.

Ιστορική αναδρομή

Οι πρώτοι υπολογιστές, τεράστιοι σε μέγεθος αλλά με πολύ περιορισμένες δυνατότητες και χαμηλές ταχύτητες επεξεργασίας, εξελίχθηκαν σε πολύ μικρούς με τεράστιες δυνατότητες και ταχύτητες επεξεργασίας.

Οι βασικές αρχές λειτουργίας των υπολογιστών που διατυπώθηκαν το 1945 από τον Φον Νόυμαν, δεν έχουν αλλάξει πρακτικά καθόλου. Οι γλώσσες προγραμματισμού αν και εξελίσσονται και συνεχώς εμπλουτίζονται με νέες δυνατότητες, τα χαρακτηριστικά τους και οι βασικές τους ιδιότητες ουσιαστικά παραμένουν τα ίδια.

Α. Γλώσσες μηχανής

Ένα πρόγραμμα σε γλώσσα μηχανής είναι μια ακολουθία δυαδικών ψηφίων που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες. Οι εντολές αυτές είναι κατανοητές από τον υπολογιστή αλλά ακατανόητες από τον άνθρωπο καθώς απαιτούν βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.

Β. Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου

Μια *συμβολική* γλώσσα ενώ έχει έννοια για τον άνθρωπο μετατρέπεται εσωτερικά από τον υπολογιστή στις αντίστοιχες ακολουθίες από 0 και 1. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα ο **συμβολομεταφραστής**.

Οι εντολές σε συμβολική γλώσσα αποτελούνται από συμβολικά ονόματα που αντιστοιχούν σε εντολές σε γλώσσα μηχανής.

Τα μειονεκτήματα των συμβολικών γλωσσών είναι τα εξής:

- Παραμένουν συνδεδεμένες στενά με την αρχιτεκτονική του υπολογιστή
- Δεν διαθέτουν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προγράμματα, που είναι δύσκολο να γραφούν και κυρίως να συντηρηθούν.
- Δεν μπορούν να μεταφερθούν σε άλλον διαφορετικό υπολογιστή ακόμη και του ίδιου κατασκευαστή.

Γ. Γλώσσες υψηλού επιπέδου (3ης γενιάς)

Οι γλώσσες υψηλού επιπέδου χρησιμοποιούν ως εντολές απλές λέξεις της αγγλικής γλώσσας ακολουθώντας αυστηρούς κανόνες σύνταξης. Οι εντολές αυτές μεταφράζονται από τον ίδιο τον υπολογιστή σε εντολές σε γλώσσα μηχανής. Οι κυριότερες γλώσσες υψηλού επιπέδου είναι οι εξής:

1) **FORTRAN**, 2) **COBOL**, 3) **ALGOL**, 4) **PL/1**, 5) **LISP**, 6) **PROLOG**, 7) **BASIC**, 8) **PASCAL**, 9) **C**, 10) **C++**, 11) **JAVA**.

- Η πρώτη γλώσσα υψηλού επιπέδου, η **FORTRAN**, αναπτύχθηκε το 1957 ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων και μετά από πολλές αλλαγές, προσθήκες και βελτιώσεις χρησιμοποιείται ακόμη και σήμερα για επιστημονικές εφαρμογές. Η FORTRAN υστερεί στη διαχείριση αρχείων δεδομένων και γενικότερα αλφαριθμητικών πληροφοριών.
- Το 1960 αναπτύχθηκε η **COBOL** ως γλώσσα κατάλληλη για ανάπτυξη εμπορικών συναλλαγών. Η COBOL καθιερώθηκε ως πρότυπο και χρησιμοποιήθηκε από πολλές επιχειρήσεις και από όλη τη δημόσια διοίκηση. Πολλές εφαρμογές βρίσκονται σε χρήση ακόμη και σήμερα.
- Η γλώσσα **ALGOL** επηρέασε ιδιαίτερα τον προγραμματισμό και τις επόμενες γλώσσες. Αναπτύχθηκε με σκοπό τη δημιουργία προγραμμάτων γενικής φύσης που να μην συνδέονται με συγκεκριμένες εφαρμογές.
- Η γλώσσα **PL/1** προσπάθησε ανεπιτυχώς να αντικαταστήσει την FORTRAN και την COBOL.
- Στο χώρο της τεχνητής νοημοσύνης αναπτύχθηκαν δυο διαφορετικές γλώσσες, η **LISP** και η **PROLOG**.
- Η γλώσσα προγραμματισμού **BASIC** αναπτύχθηκε ως γλώσσα για την εκπαίδευση αρχάριων στον προγραμματισμό. Η ανάπτυξη των μικροϋπολογιστών και οι συνεχείς

εκδόσεις της την κατέστησαν την δημοφιλέστερη, ίσως, γλώσσα στους προσωπικούς υπολογιστές.

- Η γλώσσα **PASCAL** παρουσιάστηκε το 1970 και στηρίχτηκε πάνω στην ALGOL. Είναι γλώσσα γενικής χρήσης, κατάλληλη για εκπαίδευση αρχάριων αλλά και για τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Είναι κατάλληλη για δημιουργία δομημένων προγραμμάτων. Γνωρίζει τεράστια εξάπλωση στο χώρο των μικροϋπολογιστών.
- Η γλώσσα **C** γνώρισε μεγάλη διάδοση και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix. Είναι γλώσσα κατάλληλη για ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Έχει εξελιχθεί στη γλώσσα **C++** που είναι αντικειμενοστραφής.
- Τα τελευταία χρόνια χρησιμοποιείται ειδικά για προγραμματισμό στο Διαδίκτυο η γλώσσα **JAVA**. Η JAVA είναι αντικειμενοστραφής γλώσσα που αναπτύχθηκε με σκοπό την ανάπτυξη εφαρμογών που θα εκτελούνται σε κατανεμημένα περιβάλλοντα, δηλαδή σε διαφορετικούς υπολογιστές που είναι συνδεδεμένοι στο Διαδίκτυο. Τα προγράμματα αυτά μπορούν να εκτελούνται χωρίς αλλαγές από διαφορετικούς υπολογιστές, προσωπικούς ή μεγάλα συστήματα με διαφορετικά λειτουργικά συστήματα.

Η εμφάνιση των γραφικών περιβαλλόντων εργασίας οδήγησε στην εμφάνιση νέων γλωσσών ή νέων εκδόσεων παλιότερων γλωσσών που υλοποιούσαν τις έννοιες του **οδηγούμενου από το γεγονός προγραμματισμού** και του **οπτικού προγραμματισμού**.

Με τον όρο *οπτικό* εννοούμε τη δυνατότητα να δημιουργούμε γραφικά ολόκληρο το περιβάλλον της εφαρμογής για παράδειγμα τα πλαίσια διαλόγου ή τα μενού.

Με τον όρο *οδηγούμενο από το γεγονός* εννοούμε τη δυνατότητα να ενεργοποιούνται λειτουργίες του προγράμματος με την εκτέλεση ενός γεγονότος, για παράδειγμα την επιλογή μιας εντολής από ένα μενού ή το κλικ του ποντικού.

Οι πιο διαδεδομένες γλώσσες προγραμματισμού σε γραφικό περιβάλλον για προσωπικούς υπολογιστές είναι η Visual Basic, η Visual C++ και η Java.

Τα πλεονεκτήματα των γλωσσών υψηλού επιπέδου σε σχέση με τις συμβολικές είναι τα εξής:

- Ο φυσικότερος και πιο 'ανθρώπινος' τρόπος έκφρασης των προβλημάτων.
- Η ανεξαρτησία από τον τύπο του υπολογιστή και η δυνατότητα **μεταφερσιμότητας** που αυτή συνεπάγεται.
- Η ευκολία εκμάθησης και εκπαίδευσης.

- Η ευκολία διόρθωσης λαθών και συντήρησης των προγραμμάτων.

Συνολικά οι γλώσσες υψηλού επιπέδου ελάττωσαν σημαντικά το χρόνο και το κόστος παραγωγής νέων προγραμμάτων αφού λιγότεροι προγραμματιστές μπορούν σε λιγότερο χρόνο να αναπτύξουν προγράμματα που χρησιμοποιούνται σε περισσότερους υπολογιστές.

Δ. Γλώσσες 4ης γενιάς.

Οι γλώσσες 4ης γενιάς, αντίθετα από τις γλώσσες 3ης γενιάς, είναι γλώσσες εφοδιασμένες με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης και με αυτά ο χρήστης μπορεί να επιλύει μόνος του μικρά προβλήματα εφαρμογών.

Στις γλώσσες αυτές μπορεί ο χρήστης ενός υπολογιστή σχετικά εύκολα να υποβάλλει ερωτήσεις στο σύστημα ή να αναπτύσσει εφαρμογές που αποκτούν πληροφορίες από βάσεις δεδομένων και να καθορίζει τον ακριβή τρόπο εμφάνισης αυτών των πληροφοριών.

Ταξινόμηση γλωσσών προγραμματισμού.

1. Ανάλογα με το είδος προγραμματισμού.

Α) Διαδικασιακές ή αλγοριθμικές

Β) Αντικειμενοστραφείς

Γ) Συναρτησιακές, π.χ. LISP

Δ) Μη διαδικασιακές (ή πολύ υψηλού επιπέδου), π.χ. PROLOG

Ε) Γλώσσες ερωταπαντήσεων, π.χ. SQL

2. Ανάλογα με την περιοχή χρήσης (τομέα εφαρμογών)

Α) Γλώσσες γενικής χρήσης, π.χ. BASIC, PASCAL

Α1) Γλώσσες επιστημονικής κατεύθυνσης, π.χ. FORTRAN

Α2) Γλώσσες εμπορικής κατεύθυνσης, π.χ. COBOL

Β) Γλώσσες προγραμματισμού συστημάτων, π.χ. C

Γ) Γλώσσες τεχνητής νοημοσύνης, π.χ. LISP, PROLOG

Δ) Γλώσσες ειδικής χρήσης

Αυτή τη στιγμή χρησιμοποιούνται μερικές εκατοντάδες γλώσσες ενώ συνολικά έχουν αναπτυχθεί μερικές χιλιάδες γλώσσες. Μια γλώσσα που να είναι αντικειμενικά καλύτερη από τις άλλες δεν υπάρχει ούτε πρόκειται να υπάρξει. Η επιλογή της γλώσσας για την ανάπτυξη μιας εφαρμογής εξαρτάται:

1) από το είδος της εφαρμογής,
2) το υπολογιστικό περιβάλλον στο οποίο θα εκτελεστεί,
3) τα προγραμματιστικά περιβάλλοντα που διαθέτουμε, και
4) τις γνώσεις του προγραμματιστή.

Φυσικές και τεχνητές γλώσσες

Οι *φυσικές* γλώσσες χρησιμοποιούνται για την επικοινωνία ανθρώπων. Οι *τεχνητές* γλώσσες χρησιμοποιούνται για την επικοινωνία ανθρώπου και μηχανής.

Μια γλώσσα προσδιορίζεται από:

1. το αλφάβητό της
2. το λεξιλόγιό της
3. τη γραμματική της
4. τη σημασιολογία της.

Αλφάβητο μιας γλώσσας καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα.

Το *λεξιλόγιο* αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου, τις λέξεις που είναι αποδεκτές από τη γλώσσα.

Η *γραμματική* αποτελείται από το **τυπικό** ή **τυπολογικό** και το **συντακτικό**.

> *Τυπικό* είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μια λέξη 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 κατ΄ επέκταση των εκφράσεων και προτάσεων που χρησιμοποιούνται σε μια γλώσσα.

Μια βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η δυνατότητα εξέλιξής τους. Οι φυσικές γλώσσες εξελίσσονται διαρκώς αντίθετα οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα.

Τεχνικές σχεδίασης προγραμμάτων.

Α. Ιεραρχική σχεδίαση προγράμματος.

Η *ιεραρχική σχεδίαση προγράμματος* ή η διαδικασία σχεδιασμού «από επάνω προς τα κάτω» χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε

υποπροβλήματα. Για την υποβοήθηση της ιεραρχικής σχεδίασης χρησιμοποιούνται διάφορες *διαγραμματικές τεχνικές*.

Β. Τμηματικός προγραμματισμός.

Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον *τμηματικό προγραμματισμό*. Μετά την ανάλυση του προβλήματος σε υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ***ενότητα*** που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος. Ο τμηματικός προγραμματισμός: μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρηση του προγράμματος από τρίτους.

Γ. Δομημένος προγραμματισμός.

Ο δομημένος προγραμματισμός αναπτύχθηκε από την ανάγκη α) να υπάρχει μια κοινή μεθοδολογία στην ανάπτυξη των προγραμμάτων και β) να μειωθεί η χρήση εντολών GOTO, εντολών που αλλάζουν τη ροή του προγράμματος.

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 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.

Ο όρος δομημένος προγραμματισμός περιέχει την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.

Τα πλεονεκτήματα του δομημένου προγραμματισμού είναι τα εξής:

- δημιουργία απλούστερων προγραμμάτων
- άμεση μεταφορά αλγορίθμων σε προγράμματα
- διευκόλυνση ανάλυσης του προγράμματος
- περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος
- διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους

- ευκολότερη διόρθωση και συντήρηση

Αντικειμενοστραφής προγραμματισμός.

Ο αντικειμενοστραφής προγραμματισμός είναι μια νέα μέθοδος για τη δόμηση ενός προγράμματος σε ιεραρχικά οργανωμένες τάξεις που περιγράφουν τα δεδομένα και τις λειτουργίες αντικειμένων, τα οποία μπορούν να αλληλεπιδρούν με άλλα αντικείμενα.

Η αντικειμενοστραφής σχεδίαση έχει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία με κατάλληλες μορφοποιήσεις δημιουργούνται τα αντικείμενα. Αυτή η σχεδίαση επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιούμενα.

Παράλληλος προγραμματισμός.

Μια μορφή προγραμματισμού που αναπτύσσεται τελευταία είναι ο *παράλληλος προγραμματισμός*. Πρόσφατα εμφανίστηκαν υπολογιστές που διαθέτουν περισσότερους από έναν επεξεργαστές. Οι επεξεργαστές αυτοί μοιράζονται την ίδια μνήμη και λειτουργούν παράλληλα εκτελώντας διαφορετικές εντολές του ίδιου προγράμματος. Οι υπολογιστές αυτοί εμφανίζονται θεωρητικά να επιτυγχάνουν ασύλληπτες ταχύτητες επεξεργασίας. Για να εκμεταλλευτούμε όμως τις δυνατότητες 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.

Μια γλώσσα προγραμματισμού που υποστηρίζει παράλληλο προγραμματισμό είναι η OCCAM.

Προγραμματιστικά περιβάλλοντα.

Ένα πρόγραμμα που γράφεται σε οποιαδήποτε γλώσσα προγραμματισμού πρέπει να μετατραπεί σε γλώσσα μηχανής προκειμένου να εκτελεστεί. Τα μεταφραστικά προγράμματα που χρησιμοποιούνται ανήκουν σε δυο μεγάλες κατηγορίες:

- τους **μεταγλωττιστές** ,και
- τους **διερμηνευτές**

Ο μεταγλωττιστής δέχεται σαν είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής, εκτελέσιμο *οποτεδήποτε* από τον υπολογιστή και τελείως ανεξάρτητα από το αρχικό πρόγραμμα.

Ο διερμηνευτής διαβάζει μια προς μια τις εντολές του προγράμματος και για κάθε μια εκτελεί *αμέσως* μια ισοδύναμη ακολουθία εντολών μηχανής.

Το αρχικό πρόγραμμα λέγεται **πηγαίο** ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται **αντικείμενο** πρόγραμμα.

Το αντικείμενο πρόγραμμα χρειάζεται συνήθως να συμπληρωθεί με άλλα τμήματα προγράμματος που βρίσκονται στις **βιβλιοθήκες**. Η σύνδεση αυτή γίνεται από ένα πρόγραμμα, τον **συνδετή – φορτωτή**. Το αποτέλεσμα του συνδετή είναι το **εκτελέσιμο πρόγραμμα**.

Αν κατά την μεταγλώττιση ή τη διερμήνευση ανιχνευτούν **συντακτικά** λάθη, π.χ. αναγραμματισμοί ονομάτων εντολών, παράληψη δήλωσης δεδομένων κ.α., εμφανίζονται κατάλληλα διαγνωστικά μηνύματα. Τα λάθη πρέπει πάντα να διορθωθούν προκειμένου να υποβληθεί εκ νέου το πρόγραμμα για μεταγλώττιση.

Η χρήση μεταγλωττιστή έχει το μειονέκτημα ότι για να χρησιμοποιηθεί το πρόγραμμα πρέπει να περάσει από την διαδικασία της μεταγλώττισης και της σύνδεσης. Η χρήση διερμηνευτή πλεονεκτεί στο σημείο αυτό καθώς έχουμε άμεση εκτέλεση άρα και άμεση διόρθωση του προγράμματος. Το μειονέκτημα του διερμηνευτή είναι η πιο αργή εκτέλεση του προγράμματος από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα χρησιμοποιούν διερμηνευτή κατά τη φάση της δημιουργίας του προγράμματος και μεταγλωττιστή για την τελική έκδοση και εκμετάλλευση του προγράμματος.

Για την σύνταξη των προγραμμάτων και τη διόρθωσή τους χρησιμοποιείται ένα ειδικό πρόγραμμα που ονομάζεται **συντάκτης**. Πρόκειται, ουσιαστικά περί ενός μικρού επεξεργαστή κειμένου με δυνατότητες, όμως, που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων.

Ανακεφαλαιώνοντας, για τη δημιουργία, τη μετάφραση και την εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον τρία προγράμματα:

1. ο συντάκτης
2. ο μεταγλωττιστής
3. ο συνδετής

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν αυτά τα προγράμματα με ενιαίο τρόπο.