6.

Εισαγωγή στον προγραμματισμό

Εισαγωγή

Στα προηγούμενα κεφάλαια αναφερθήκαμε αναλυτικά στην ανάπτυξη των αλγορίθμων και των διαφόρων τεχνικών. Στα επόμενα κεφάλαια θα ασχοληθούμε με τον προγραμματισμό δηλαδή τη διατύπωση των αλγορίθμων σε τέτοια μορφή, ώστε να μπορούν να υλοποιηθούν από τον υπολογιστή. Το κεφάλαιο αυτό ασχολείται με γενικές έννοιες που είναι απαραίτητες πριν από την ενασχόληση με τη διαδικασία του προγραμματισμού. Ορίζεται η έννοια του προγράμματος, παρουσιάζεται μία σύντομη ιστορία των γλωσσών προγραμματισμού και των ειδών προγραμματισμού, καθώς και οι τεχνικές που χρησιμοποιούνται για τη σωστή δημιουργία προγραμμάτων. Συγκεκριμένα παρουσιάζονται οι τεχνικές της ιεραρχικής σχεδίασης προγραμμάτων, του τμηματικού προγραμματισμού και κύρια οι αρχές του δομημένου προγραμματισμού. Επίσης περιγράφεται η διαδικασία που ακολουθείται από τη τη σύνταξη του προγράμματος μέχρι την τελική του εκτέλεση από τον υπολογιστή και τη λήψη των αποτελεσμάτων.

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι ο μαθητής :

- Να ορίζει τι είναι πρόγραμμα και να κατατάσσει και να συγκρίνει τις γλώσσες προγραμματισμού.
- Να αναγνωρίζει τα κυριότερα είδη προγραμματισμού και να περιγράφει τα βασικά χαρακτηριστικά των τεχνικών που χρησιμοποιούνται.
- Να διατυπώνει τα πλεονεκτήματα του δομημένου προγραμματισμού.
- Να περιγράφει τη διαδικασία εκτέλεσης ενός προγράμματος.
- Να αναφέρει τα βασικά προγράμματα που περιέχει ένα προγραμματιστικό περιβάλλον

Προερωτήσεις

- ✓ Η δημιουργία του αλγορίθμου αρκεί για να επιλύσουμε ένα πρόβλημα στον υπολογιστή;
- ✓ Πώς διαχειρίζεται τις πληροφορίες ο υπολογιστής;
- ✓ Γνωρίζεις κάποιες γλώσσες προγραμματισμού;
- ✓ Πώς και γατί εξελίσσονται οι γλώσσες;
- ✓ Η ύπαρξη συγκεκριμένων μεθοδολογιών και τεχνικών βοηθάει στην επίλυση των προβλημάτων;

6. 1 Η έννοια του προγράμματος

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.
- Τη διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή.

Ο προγραμματισμός ασχολείται με το τρίτο αυτό στάδιο, τη δημιουργία του προγράμματος δηλαδή του συνόλου 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 στοιχείο του είναι τα δεδομένα και οι δομές δεδομένων επί των οποίων ενεργεί. 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 τα πολύπλοκα προβλήματα.

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου (προγραμματιστή) με τη μηχανή (υπολογιστή)

Η εντύπωση αυτή όμως είναι απλώς μία ψευδαίσθηση. Ο υπολογιστής, ως γνωστόν, είναι μία μηχανή που καταλαβαίνει μόνο δύο καταστάσεις, οι οποίες αντιπροσωπεύονται με δύο αριθμούς το μηδέν και το ένα, τα ψηφία του δυαδικού συστήματος. Το μόνο πράγμα που κάνει ο υπολογιστής είναι στοιχειώδεις ενέργειες σε ακολουθίες αυτών των δύο ψηφίων, αλλά αυτές τις ενέργειες τις εκτελεί μα ασύλληπτη ταχύτητα. Ο υπολογιστής μπορεί απλά να αποθηκεύει στη μνήμη τις ακολουθίες των δυαδικών ψηφίων, να τις ανακτά, να κάνει στοιχειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει.

6.2 Ιστορική αναδρομή

Από τη δημιουργία του πρώτου υπολογιστή μέχρι σήμερα έχουν αλλάξει πάρα πολλά πράγματα. Οι πρώτοι υπολογιστές, τεράστιοι σε μέγεθος αλλά με πάρα πολύ περιορισμένες δυνατότητες και μικρές ταχύτητας επεξεργασίας εξελίχθηκαν σε πολύ μικρούς σε μέγεθος υπολογιστές με τεράστιες όμως δυνατότητες και ταχύτητες επεξεργασίας.

Ενώ λοιπόν το υλικό (hardware) των υπολογιστών βελτιώνεται, τελειοποιείται και ταυτόχρονα παρέχει νέες δυνατότητες επεξεργασίας, οι βασι-

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

*Ενα πρόγραμμα σε **γλώσσα μηχανής** είναι μια ακολουθία δυαδικών ψηφίων, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες.*

6.2.1 Γλώσσες μηχανής

Αρχικά για να μπορέσει ο υπολογιστής να εκτελέσει μία οποιαδήποτε λειτουργία, έπρεπε να δοθούν κατευθείαν οι κατάλληλες ακολουθίες από 0 και 1, δηλαδή εντολές σε μορφή κατανοητή από τον υπολογιστή αλλά ακατανόητες από τον άνθρωπο. Ο τρόπος αυτός ήταν επίπονος και ελάχιστοι μπορούσαν να τον υλοποιήσουν, αφού απαιτούσε βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.

Ο πρώτος υπολογιστής ο περίφημος ENIAC για να "προγραμματιστεί", ώστε να εκτελέσει κάποιους υπολογισμούς, έπρεπε να αλλάξουν θέση εκατοντάδες διακόπτες και να ρυθμιστούν αντίστοιχα όλες οι καλωδιώσεις, διαδικασία εξαιρετικά επίπονη και χρονοβόρα. Ο "προγραμματισμός" των πρώτων αυτών υπολογιστών, δεν ήταν ουσιαστικά προγραμματισμός με τη σημερινή έννοια του όρου. Ο υπολογιστής αναδιαρθρωνόταν, ώστε να εκτελέσει τους απαιτούμενους υπολογισμούς και στη συνέχεια έπρεπε να αλλάξει πάλι η διάρθρωσή του, ώστε να εκτελέσει έναν άλλο υπολογισμό.

Οι εντολές ενός προγράμματος και σήμερα μετατρέπονται σε ακολουθίες που αποτελούνται από 0 και 1, τις εντολές σε **γλώσσα μηχανής,** όπως ονομάζονται, οι οποίες εκτελούνται από τον υπολογιστή.

6.2.2 Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου

Από τα πρώτα χρόνια άρχισαν να γίνονται προσπάθειες για τη δημιουργία μίας 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 (πρόσθεσε) ακολουθούμενη από δύο αριθμούς, είναι κατανοητή από τον άνθρωπο και απομνημονεύεται σχετικά εύκολα. Η εντολή αυτή θα μεταφραστεί από τον υπολογιστή σε μία ακολουθία δυαδικών ψηφίων και στη συνέχεια μπορεί να εκτελεστεί. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο **συμβολομεταφραστής** (assembler).

Η χρήση των πρώτων αυτών συμβολικών γλωσσών, που συνεχίζουν να χρησιμοποιούνται για ειδικούς σκοπούς, ήταν σαφώς μια εξέλιξη από τις α-

κατανόητες ακολουθίες δυαδικών στοιχείων. Ωστόσο παρέμεναν στενά συνδεδεμένες με την αρχιτεκτονική του κάθε υπολογιστή. Επίσης δεν διέθεταν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προγράμματα, που ήταν δύσκολο να γραφούν και κύρια να συντηρηθούν. Ακόμη τα προγράμματα δεν μπορούν να μεταφερθούν σε άλλον διαφορετικό υπολογιστή, ακόμη και του ίδιου κατασκευαστή. Οι γλώσσες αυτές ονομάζονται συμβολικές ή γλώσσες χαμηλού επιπέδου, αφού εξαρτώνται από την αρχιτεκτονική του υπολογιστή.

Οι εντολές σε συμβολική γλώσσα αποτελούνται από συμβολικά ονόματα που αντιστοιχούν σε εντολές της γλώσσας μηχανής. Οι συμβολικές γλώσσες είναι συνδεδεμένες με την αρχιτεκτονική κάθε υπολογιστή

```
10101000 00001010          INDEX=$01    sum = 0
10001100 00000001          SUM=$02      FOR index=1 TO 10
00111100                   LDA #10         sum=sum+index
01010001 00000001          STA INDEX    NEXT index
01000011 00000001          CLA          END
11000000 11111010    LOOP  ADD INDEX
10001100 00000010          DEC INDEX
11111111                   BNE LOOP
                           STA SUM
                           BRK
```

Σχ. 6.1. Åíá ðñüãñáììá óå ãëþóóá ìç÷áíÞò, óå óõìâïëéêÞ ãëþóóá êáé óå ãëþóóá õøçëïý åðéðÝáïõ ãéá ôïí õðïëïãéóìü ôïõ áèñïßóìáôïò ôùí áñéèìþí 1 Ýùò 10

6.2.3 Γλώσσες υψηλού επιπέδου

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 1957 η IBM ανάπτυξε την πρώτη γλώσσα υψηλού επιπέδου τη **FORTRAN.** Το όνομα FORTRAN προέρχεται από τις λέξεις FORmula TRANslation, που σημαίνουν μετάφραση τύπων. Η FORTRAN αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων. Το πρόγραμμα που γράφεται σε FORTRAN ή σε οποιαδήποτε άλλη 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 FORTRAN μπορεί να εκτελεστεί σε οποιοδήποτε άλλο υπολογιστή, αρκεί να υπάρχει ο αντίστοιχος μεταγλωττιστής για τον υπολογιστή αυτό. Η γλώσσα FORTRAN μετά από πολλές αλλαγές, προσθήκες και βελτιώσεις χρησιμοποιείται ακόμη και σήμερα για επιστημονικές εφαρμογές.

```
C   PROGRAM EQUATION
    READ(*,1) A,B
1   FORMAT(F5.1)
    IF (A.EQ.0) GO TO 20
    X=(-1.)*B/A
    WRITE(*,2) X
2   FORMAT('X=',F10.2)
    GO TO 50
20  IF (B.EQ.0) WRITE(*,3)
    IF (B.NE.0) WRITE(*,4)
3   FORMAT('ÁÏÑÉÓÔÇ')
4   FORMAT('ÁÄÕÍÁÔÇ')
50  STOP
    END
```

Σχ.6.2. Ç ãëþóóá FORTRAN õðÞñîå ç ðñþôç ãëþóóá ðñïãñáììáôéóìïý õøçëïý åðéðÝäéõ. Ðñüêåéôáé ãéá ãëþóóá êáôÜëëçëç ãéá ðñïëïãéóìïýò, åíþ ôóôåñåß óôç äéá÷åßñéóç áñ÷åßùí äåäïìÝíùí êáé ãåíéêüôåñá áëöáñéèìçôéêþí ðëçñïöïñéþí. Ãíþñéóå ðïëëÝò âåëôéþóåéò ìå êõñéüôåñïõò óôáèìïýò ôéò åêäüóåéò 4, 77, 90/95 êáé Visual FORTRAN. Ôï ðñüãñáììá ôïõ ðáñáäåßãìáôïò åðéëýåé ôçí åîßóùóç á' âáèìïý.

Η FORTRAN παρά τα ισχυρά χαρακτηριστικά της και τις συνεχείς αλλαγές που τη καθιστούσαν συνεχώς αποτελεσματικότερη, δεν μπορούσε να καλύψει τις απαιτήσεις σε όλους τους τομείς δραστηριοτήτων, όπως και καμία άλλη γλώσσα προγραμματισμού δεν κατάφερε. Έτσι αναπτύχθηκαν και συνεχίζουν να αναπτύσσονται πολλές γλώσσες προγραμματισμού για διάφορες περιοχές δραστηριοτήτων.

Το 1960 αναπτύχθηκε μία άλλη γλώσσα, σταθμός στον προγραμματισμό η γλώσσα COBOL. Η **COBOL** όπως δηλώνει και το όνομα της (COmmon Business Oriented Language - Κοινή γλώσσα προσανατολισμένη στις επιχειρήσεις) είναι κατάλληλη για ανάπτυξη εμπορικών εφαρμογών, και γενικότερα διαχειριστικών εφαρμογών, τομέας όπου η FORTRAN υστερούσε. Η COBOL καθιερώθηκε ως πρότυπο και χρησιμοποιήθηκε από πολλές επιχειρήσεις καθώς και από όλη τη δημόσια διοίκηση. Η γλώσσα γνώρισε πολλές εκδόσεις και πάρα πολλές εφαρμογές βρίσκονται σε χρήση ακόμη και σήμερα.

Μια από τις σημαντικότερες γλώσσα προγραμματισμού με ελάχιστη πρακτική εφαρμογή αλλά που επηρέασε ιδιαίτερα τον προγραμματισμό και τις επόμενες γλώσσες, είναι η **ALGOL** (ALGOrithmic Language – Αλγοριθμική γλώσσα). Αναπτύχθηκε από Ευρωπαίους επιστήμονες, αρχικά το

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  EQUATION.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-PC.
       OBJECT-COMPUTER. IBM-PC.
       SPECIAL-NAMES. DECIMAL-POINT  IS COMMA.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77  X          PIC S9(6)V9.
       77  A          PIC S9(6).
       77  B          PIC S9(6).
       77  W-X        PIC -(6),-.
       PROCEDURE DIVISION.
       ARXH.
           DISPLAY ‘ ÄÙÓÅ Á’.
           ACCEPT  A.
           DISPLAY ‘ ÄÙÓÅ Â’.
           ACCEPT  B.
           DISPLAY ‘ ‘.
           IF  A = 0   GO  TO  ROYT-1.
           COMPUTE X = B * (- 1) / A.
           MOVE  X  TO W-X.
           DISPLAY ‘ H ËÕÓÇ ÅÉÍÁÉ : ‘  W-X.
           STOP  RUN.
      ROYT-1.
           IF  B = 0
               DISPLAY ‘ ÁÏÑÉÓÔÇ’
           ELSE
               DISPLAY ‘ ÁÄÕÍÁÔÇ’.
           STOP  RUN.
```

Σχ. 6.3. Ç ãëþóóá COBOL äçìéïõñãÞèçêå áðü ôçí Grace Marray Hopper áîéùìáôéêü ôïõ ðïëåìéêïý íáõôéêïý ôùí ÇÐÁ ôï 1960. Ç COBOL Ýêáíå äõíáôÞ ôçí áîéïðïßçóç ôùí õðïëïãéóôþí áðü ôéò åðé÷åéñÞóåéò êáé ôïõò ïñãáíéóìïýò ðáñÝ÷ïíôáò éó÷õñüôáôåò äõ íáôüôçôåò äéá÷åßñéóçò áñ÷åßùí äåäïìÝíùí. Åíá ðñüãñáììá COBOL äéáèÝôåé ôÝóóåñéò õðïäéáéñÝóåéò (divisions). Ç ãëþóóá ÷ñçóéìïðïéåß ðåñéãñáöéêü ôñüðï ãéá ôç óýíôáîç ôùí åíôïëþí ìå ÷ñÞóç ñçìÜôùí ôçò áããëéêÞò ãëþóóáò, üðùò ADD, MULTIPLY, MOVE ê.ëð. ìå áðïÝëåóìá ôç äçìéïõñãßá ãåíéêÜ ìáêñïêåëþí ðñïãñáììÜôùí. ÊÜèå åíôïë Þ ôçò ãëþóóáò ôåñìáôßæåôáé ìå ôåëåßá. Ôï ðñüãñáììá ôïõ ðáñáäåßãìáôïò åðéëýåé ôçí åíßóùóç á’ âáèìïý.

```
10 REM ÅÐÉËÕÓÇ ÂÎÉÓÙÓÇÓ Á'ÂÁÈÌÏÕ
20 INPUT "A=",A
30 INPUT "B=",B
40 IF A=0 THEN 100
50 X=-B/A
60 PRINT "X=";X
70 END
100 IF B=0 THEN PRINT "ÁÏÑÉÓÔÇ" ELSE PRINT "ÁÄÕÍÁÔÇ"
110 END
```

Σχ. 6.4. Ç ãëþóóá BASIC äçìéïõñãÞèçêå ôï 1964 óôï Dartmouth College áðü ôïõò êá èçãçôÝò Kemeny êáé Kurtz. Óôü÷ïò ôùí äçìéïõñãþí ôçò Þôáí ç õëïðïßçóç ìéáò áðëÞò ãëþóóáò ðñïãñáììáôéóìïý ãéá åêðáéäåõôéêïýò óêïðïýò. Ç BASIC ãíþñéóá åêðëçêôé êÞ äéÜäïóç ìå ôçí åìöÜíéóç ôùí ìéêñïõðïëïãéóôþí (1975). Ï ðñþôïò äéåñìçíåõôÞò ôçò ãëþóóáò óå 8Ê ROM Þôáí Ýñãáï ôùí Bill Gates êáé Paul Allen. Áò óçìåéùèåß üôé, ç å ôáéñßá Microsoft éäñýèçêå áðü ôïí Bill Gates ãéá ôçí åìðïñéêÞ åêìåôÜëåõóç áõôïý ôïõ äéåñìçíåõôÞ. Ç Ýêäïóç GWBASIC ðáñïõóéÜæåôáé ôï 1981 ìå ôïí IBM PC. Êýñéï ÷áñáêôçñéóôéêü ôçò ãëþóóáò åßíáé ç ýðáñîç áñéèìþí ôùí ãñáììþí ôïõ ðñïãñáììÜôïò, ïé ïðïßïé ÷ñçóéìïðïéýíôáé êáé ùò èÝóåéò ðñïãñáììýìáôïò åíôïëÞí äéáêëÜùóçò. ÅíÝëéîç ôçò ãëþóóáò åßíáé ïé åêäüóåéò QuickBASIC êáé Visual BASIC. Ôï ðñüãñáììá ôïõ ðáñáäåßãìáôïò åðéëýåé ôçí åíßóùóç á' âáèìýí.

1960, με σκοπό τη δημιουργία γενικής φύσης προγραμμάτων που να μη συνδέονται με συγκεκριμένες εφαρμογές.

Στα μέσα της δεκαετίας του 60 αναπτύχθηκε η γλώσσα **PL/1** (Programming Language/1 – Γλώσσα Προγραμματισμού υπ΄ αριθμόν 1) που προσπάθησε, χωρίς επιτυχία να καλύψει όλους τους τομείς του προγραμματισμού, επιστημονικούς και εμπορικούς, αντικαθιστώντας τόσο τη FORTRAN όσο και την COBOL .

Στο χώρο της Τεχνητής Νοημοσύνης αναπτύχθηκαν δύο γλώσσες αρκετά διαφορετικές από όλες τις άλλες. Στα μέσα του 60 αναπτύχθηκε στο MIT η **LISP** (LISt Processor- Επεξεργαστής Λίστας), γλώσσα η οποία προσανατολίζεται σε χειρισμό λιστών από σύμβολα και η **PROLOG** (PROgramming LOGic –Λογικός Προγραμματισμός) στις αρχές του 70. Οι δύο αυτές γλώσσες χρησιμοποιούνται σε προβλήματα Τεχνητής νοημοσύνης (έμπειρα συστήματα, παιγνίδια, επεξεργασία φυσικών γλωσσών κ.λπ.).

Δύο σημαντικότατες γλώσσες γενικού σκοπού, οι οποίες αναπτύχθηκαν τη δεκαετία του 60 αλλά χρησιμοποιούνται πάρα πολύ στις ημέρες μας, είναι η BASIC και η PASCAL.

```
TO KYBOS :A
REPEAT 4 [FD :A RT 90]
PU SETPOS [20 20] PD
REPEAT 4 [FD :A RT 90]
PU HOME PD
REPEAT 2 [FD :A RT 45 FD 29 RT 135]
PU SETX :A SETY 0 PD
REPEAT 2 [FD :A RT 45 FD 29 RT 135]
HOME
END
```

Σχ. 6.5. Ç ãëþóóá ðñïãñáììáôéóìïý LOGO ïëïêëçñþèçêå ôï 1967 óôç Âïóôþíç áðü ôïí Seymoyr Papert. Ôï üíïìÜ ôçò ðñïÝñ÷åôáé áðü ôçí åëëçíéêÞ ëÝîç "ëüãïò". Ðñü êåéôáé ãéá ãëþóóá êáôÜëëçëç ãéá ôçí åéóáãùãÞ óôïí ðñïãñáììáôéóìü-ìáèçôþí ìé êñÞò çëéêßáò. Ôï ðñüãñáììá ôïõ ðáñáäåßãìáôïò ó÷åäéÜæåé Ýíáí êýâï áêìÞò Á ìå äéá äï÷éêÝò êéíÞóåéò ôçò ÷åëþíáò.

Η γλώσσα προγραμματισμού **BASIC** (Beginner's All Purpose Symbolic Instruction Code – Συμβολικός Κώδικας Εντολών Γενικής Χρήσης για Αρχάριους) αρχικά αναπτύχθηκε, όπως δηλώνει και το όνομα της, ως γλώσσα για την εκπαίδευση αρχαρίων στον προγραμματισμό. Σχεδιάστηκε για να γράφονται σύντομα προγράμματα, τα οποία εκτελούνται με τη βοήθεια διερμηνευτή (interpreter). Η ανάπτυξη όμως των μικροϋπολογιστών και οι συνεχείς εκδόσεις της γλώσσας βοήθησαν στην εξάπλωσή της, τόσο ώστε να γίνει ίσως η δημοφιλέστερη γλώσσα στους προσωπικούς υπολογιστές. Η τυποποίηση της δε από τη Microsoft με τις εκδόσεις QuickBasic και κύρια με τη Visual Basic, καθιέρωσε τη γλώσσα ως πρότυπο για ανάπτυξη εφαρμογών σε προσωπικούς υπολογιστές

Η γλώσσα **PASCAL** (δημιούργημα του καθηγητή Niklaus Wirth) έφερε μεγάλες αλλαγές στον προγραμματισμό. Παρουσιάστηκε το 1970 και στηρίχτηκε πάνω στην ALGOL. Είναι μία γλώσσα γενικής χρήσης, η οποία είναι κατάλληλη τόσο για την εκπαίδευση όσο και τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Χαρακτηριστικό της γλώσσας είναι η η καταλληλότητα για τη δημιουργία δομημένων προγραμμάτων. Η PASCAL γνώρισε και συνεχίζει να γνωρίζει τεράστια εξάπλωση ειδικά στο χώρο των μικροϋπολογιστών και αποτέλεσε τη βάση για την ανάπτυξη άλλων ισχυρότερων γλωσσών όπως η ADA και η Modula-2.

Στα μέσα του 1960 παρουσιάστηκε για πρώτη φορά μία τεχνική σχεδίασης προγραμμάτων που έμελλε να αλλάξει ριζικά τον τρόπο ανάπτυξης προγραμμάτων καθώς και τις ίδιες τις γλώσσες προγραμματισμού. Η τεχνική του **δομημένου προγραμματισμού** η οποία εξασφαλίζει τη δημιουργία

```
(DEFUN a-exisosi (a b)
  (setf apot (- (/ b a)))
  (princ “Ç åîßóùóç  ”)
  (princ a)
  (princ “x + ”)
  (princ b)
  (princ “ = 0  Ý÷åé óáí ëýóç ÷ = ”)
  (princ apot))
```

Σχ. 6.6. Ç ãëþóóá LISP äçìéïõñãÞèçêå ôï 1959 óôï ÌÉÔ. Ðñüêåéôáé ãéá ìç äéáäéêáóéá êÞ ãëþóóá ðïõ ðñïïñßæåôáé ãéá ôçí åðåîåñãáóßá óõìâïëéêþí äåäïìÝíùí. Âáóéêüò ôýðïò äåäïìÝíùí, áðü ôïí ïðïßï åî Üëëïõ ðÞñå êáé ôï üíïìÜ ôçò, åßíáé ç óõíäåäåìÝ íç ëßóôá. Óôï ðáñÜäåéãìá öáßíåôáé ìéá óõíÜñôçóç ôçò ãëþóóáò, ðïõ åðéëýåé ôçí åîß óùóç á’ âáèìïý. Ôï ðñüãñáììá åêôåëåßôáé äßíïíôáò óôç ãñáììÞ åíôïëþí ð.÷. (a-exisosi 2 5).

προγραμμάτων απλών στη συγγραφή και την κατανόηση και εύκολων στη διόρθωση. Ο δομημένος προγραμματισμός και τα χαρακτηριστικά του θα παρουσιαστούν εκτενώς σε επόμενη παράγραφο.

Μία ακόμη γλώσσα που γνώρισε μεγάλη διάδοση είναι η γλώσσα **C**. Η C αναπτύχθηκε στα εργαστήρια της εταιρείας BELL και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix, γλώσσα με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal κατάλληλη για ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Η C εξελίχτηκε στη γλώσσα C++, που είναι αντικειμενοστραφής. Η ιδέα του **αντικειμενοστραφούς προγραμματισμού** παρουσιάστηκε για πρώτη φορά στη δεκαετία του 70 και συνεχίζει ακόμη να απλώνεται αλλάζοντας τον παραδοσιακό προγραμματισμό. Λόγω της σημασίας του αντικειμενοστραφούς προγραμματισμού μερικά στοιχεία του παρουσιάζονται σε ξεχωριστή παράγραφο.

Τα τελευταία χρόνια χρησιμοποιείται ιδιαίτερα, ειδικά για προγραμματισμό στο Διαδίκτυο (Internet), η **JAVA.** Η JAVA είναι μία αντικειμενοστραφής γλώσσα που αναπτύχθηκε από την εταιρεία 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, προσωπικούς ή μεγάλα συστήματα με διαφορετικά λειτουργικά συστήματα χωρίς αλλαγές.

Η εμφάνιση των γραφικών περιβαλλόντων εργασίας δημιούργησε την ανάγκη για ανάπτυξη προγραμμάτων που να εκμεταλλεύονται τον γραφι-

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    float a, b;

    printf(“A = ”);
    scanf(“%f”, &a);
    printf(“B = ”);
    scanf(“%f”, &b);
    if (a == 0) {
        if (b == 0) {
            printf(“ÁÏÑÉÓÔÇ\n”);
        }
        else {
            printf(“ÁÄÕÍÁÔÇ\n”);
        }
    }
    else {
        printf(“X = %f\n”, -b/a);
    }

    return 0;
}
```

Σχ. 6.7. Ç ãëþóóá ðñïãñáììáôéóìïý C áíáðôý÷èçêå óôá åñãáóôÞñéá Bell-ôçò áìåñé êáíéêÞò åôáéñßáò Áô&Ô ôï 1972 áðü ôïí Dennis Ritchie. Ç ãëþóóá C åßíáé ìéá äïìçìÝíç äéáäéêáóéáêÞ ãëþóóá ãåíéêÞò ÷ñÞóçò ðïõ ÷áñáêôçñßæåôáé áðü ëéôüôçôá óôçí Ýêôñá óç êáé ìéá ðëïýóéá óõëëïãÞ ôåëåóôþí êáé ôýðùí äåäïìÝíùí. ÓôåíÜ óõíäåäåìÝíç ìå ôï UNIX, ç ãëþóóá C ÷ñçóéìïðïéåßôáé åõñýôáôá ãéá ôç äçìéïõñãßá ëåéôïõñãéêþí óõ óôçìÜôùí êáé Üëëùí ðáêÝôùí ëïãéóìéêïý. Ôï ðñüãñáììá ôïõ ðáñáäåßãìáôïò åðéëýåé ôçí åíßóùóç á’ âáèìïý.

κό αυτό τρόπο επικοινωνίας χρήστη-υπολογιστή. Στα περισσότερα προγραμματιστικά περιβάλλοντα που υπήρχαν, ήταν πολύ δύσκολη έως αδύνατη η ανάπτυξη εφαρμογών, ικανών να εκμεταλλεύονται τα γραφικά αυτά χαρακτηριστικά.

Έτσι εμφανίστηκαν γλώσσες ή νέες εκδόσεις των γλωσσών που υλοποιούσαν τις έννοιες του **οδηγούμενου από το γεγονός προγραμματισμού** (object driven programming) και του **οπτικού προγραμματισμού** (visual programming).

```
CLEAR
? “1.  ÅÉÓÁÃÙÃÇ ÓÔÏÉ×ÁÉÙÍ”
? “2. ÅÊÔÕÐÙÓÇ ÅÔÉÊÅÔÙÍ”
? “3. ÐÑÏÂÏËÇ”
? “4. ÔÅËÏÓ”
INPUT “Åðéëýîôå [1..4] : “ TO CHOICE
DO CASE
   CASE CHOICE=1
      APPEND
   CASE CHOICE=2
      LABEL FORM PELATES
   CASE CHOICE=3
      BROWSE
   OTHERWISE
      QUIT
END CASE
```

Σχ. 6.8. Ç dBASE ðáñïõóéÜóôçêå óôá ôÝëç ôçò äåêáåßáò ôïõ 70 áðü ôçí åôáéñßá Ashton-Tate áñ÷éêÜ ãéá ìéêñïõðïëïãéóôÝò 8-bit êáé áñãüôåñá ãéá ðñïóùðéêïýò õðïëïãéóôÝò. Ç dBASE õðÞñîå ï óðïõäáéüôåñïò åêðñüùðïò åîåëéãìÝíùí ãëùóóþí êáé åñãáëåßùí ðñïãñáììáôéóìïý ôçò åðï÷Þò ìå êýñéï ÷áñáêôçñéóôéêü ôéò åîáéñåôéêÝò äõíáôüôçôåò äéá÷åßñéóçò áñ÷åßùí (âÜóåùí) äåäïìÝíùí. Ç dBASE ìðïñïýóå íá ëåéôïõñãåß ìå Ümåóç åêÔëåóç ôùí åíôïëþí ôçò ìÝóá áðü ìéá äéáëïãéêÞ äéåðáöÞ ÷ñÞóôç, ìå óõíÝðåéá íá ìðïñåß íá ÷ñçóéìïðïéçèåß êáé áðü Üôïìá ìå åëÜ÷éóôåò ãíþóåéò ðñïãñáììáôéóìý. Ôï ãåãïíüò áõôü óõíÝâáëå óôçí åõñåßá äéÜäïóç ôùí ðñïüðïèÝóåùí õðïëïãéóôþí. Ç ðëÝïí äéáäåäïìÝíç Ýêäïóç Þôáí ç dBASE III Plus. ÅíÝëåíÞ ôçò õðÞñîå ï Clipper ìåôáãëùôôéæùìåíç ãëþóóá, ìå ôçí ïðßá äçìéïõñãÞèçêáí ðëÞèïò åìðïñé êþí åöáñìïãþí óå ðñïóùðéêïýò õðïëïãéóôÝò. Ôá ðñïúüíôá áõôÜ, ðïõ óõ÷íÜ åðïíï ìÜæïíôáé xBASE, õðÞñîáí ðñïÜãåëé ôùí óçìåñéíþí ðáêÝôùí äéá÷åßñéóçò âÜóåùí äåäïìÝíùí üðùò ç Access. Ôï áðüóðáóìá ðñïãñÜììáôïò ôïõ ðáñáäåßãìáôïò ðáñïõ óéÜæå Ýíá ìåíïý åðéëïãþí êáé åêôåëåß äéáêëÜùóç óôçí áíôßóôïé÷ç ëåéôïõñãßá.

Οι γλώσσες υψηλού επιπέδου χρησιμοποιούν ως εντολές απλές λέξεις της αγγλικής γλώσσας ακολουθώντας αυστηρούς κανόνες σύνταξης, οι οποίες μεταφράζονται από τον ίδιο τον υπολογιστή σε εντολές σε γλώσσα μηχανής

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 μίας εντολής από ένα μενού ή το κλικ του ποντικιού.

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

Πλεονεκτήματα των γλωσσών υψηλού επιπέδου

Στα πλεονεκτήματα των γλωσσών προγραμματισμού υψηλού επιπέδου σε σχέση με τις συμβολικές μπορούν να αναφερθούν:

- 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 λαθών και η συντήρηση προγραμμάτων σε γλώσσα υψηλού επιπέδου είναι πολύ ευκολότερο έργο.

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

6.2.4 Γλώσσες 4ης γενιάς

Οι γλώσσες υψηλού επιπέδου (γλώσσες 3ης γενιάς) γνώρισαν μεγάλη επιτυχία λόγω των πλεονεκτημάτων που παρουσιάζουν. Ωστόσο απευθύνονται μόνο σε προγραμματιστές. Ο χρήστης ενός υπολογιστή δεν είχε τη δυνατότητα να επιφέρει αλλαγές σε κάποιο πρόγραμμα, προκειμένου να ικανοποιήσει μια νέα ανάγκη του. Σταδιακά όμως πολλές γλώσσες εφοδιάστηκαν με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης και με αυτά ο 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 χρήστης ενός υπολογιστή έχει τη δυνατότητα, σχετικά εύκολα, να υποβάλει ερωτήσεις στο σύστημα ή να αναπτύσσει ε-

Ταξινόμηση γλωσσών προγραμματισμού

Όλες οι γλώσσες προγραμματισμού που έχουν αναπτυχθεί μέχρι σήμερα αντιπροσωπεύουν διάφορες ιδέες πάνω στον προγραμματισμό και η κάθε μία είναι συνήθως καλύτερα προσαρμοσμένη σε ορισμένες κατηγορίες προβλημάτων. Η μεγάλη πλειοψηφία των γλωσσών ανήκει στην κατηγορία των **διαδικασιακών** (procedural) γλωσσών. Είναι γνωστές επίσης και ως **αλγοριθμικές** γλώσσες, γιατί είναι σχεδιασμένες για να επιτρέπουν την υλοποίηση αλγορίθμων. Αλλες κατηγορίες γλωσσών υψηλού επιπέδου είναι:

- **Αντικειμενοστραφείς γλώσσες** (object -oriented languages)
- **Συναρτησιακές γλώσσες** (functional languages) π.χ. LISP
- **Μη διαδικασιακές γλώσσες** (non procedural languages) π.χ. PROLOG. Χαρακτηρίζονται επίσης και ως γλώσσες πολύ υψηλού επιπέδου.
- **Γλώσσες ερωταπαντήσεων** (query languages) π.χ. SQL.

Μια άλλη ταξινόμηση μπορεί να προκύψει με βάση την περιοχή χρήσης. Με αυτό το κριτήριο διακρίνουμε:

- **Γλώσσες γενικής χρήσης.** Θεωρητικά κάθε γλώσσα γενικής χρήσης μπορεί να χρησιμοποιηθεί για την επίλυση οποιουδήποτε προβλήματος. Στην πράξη ωστόσο κάθε γλώσσα έχει σχεδιαστεί για να ανταποκρίνεται καλύτερα σε ορισμένη κατηγορία προβλημάτων. Διακρίνονται σε:
  - ✓ **Γλώσσες επιστημονικής κατεύθυνσης** (science-oriented languages) π.χ. FORTRAN
  - ✓ **Γλώσσες εμπορικής κατεύθυνσης** (business-oriented languages) π.χ. COBOL.

  Ας σημειωθεί ότι ορισμένες γλώσσες τα καταφέρνουν εξίσου καλά και στους δύο πρηγούμενους τομείς π.χ. BASIC, Pascal.
- **Γλώσσες προγραμματισμού συστημάτων** (system programming languages) π.χ. C.
- **Γλώσσες τεχνητής νοημοσύνης** (artificial intelligence languages) π.χ. LISP, PROLOG.
- **Γλώσσες ειδικής χρήσης.** Πρόκειται για γλώσσες που χρησιμοποιούνται σε ειδικές περιοχές εφαρμογών όπως π.χ. στα γραφικά με υπολογιστή, στη ρομποτική, στη σχεδίαση ολοκληρωμένων κυκλωμάτων, στα Συστήματα Διοίκησης Βάσεων Δεδομένων, στην εκπαίδευση μέσω υπολογιστή κ.α.

φαρμογές που ανακτούν πληροφορίες από βάσεις δεδομένων και να καθορίζει τον ακριβή τρόπο εμφάνισης αυτών των πληροφοριών, όπως στο παράδειγμα που ακολουθεί.

```
SELECT ENAME, JOB, SAL
    FROM EMPLOYES
    WHERE DEPTNO=20
    AND SAL > 300000;
```

Η ερώτηση αυτή σε SQL εκτελεί αναζήτηση στη βάση δεδομένων EMPLOYES και επιστρέφει το όνομα, τη θέση και το μισθό των υπαλλήλων της διεύθυνσης 20 που κερδίζουν πάνω από 300.000 δρχ.

Ποια είναι η καλύτερη γλώσσα προγραμματισμού

Στην ιστορία του προγραμματισμού έχουν αναπτυχθεί χιλιάδες γλώσσες και αυτή τη στιγμή χρησιμοποιούνται μερικές εκατοντάδες. Υπάρχουν γλώσσες κατάλληλες για ανάπτυξη ειδικών εφαρμογών και άλλες κατάλληλες για γενική χρήση. Υπάρχουν γλώσσες κατάλληλες για εκπαίδευση και άλλες για ανάπτυξη εμπορικών εφαρμογών. Γλώσσες που επιτρέπουν την εύκολη ανάπτυξη εφαρμογών σε γραφικό περιβάλλον και άλλες που εκμεταλλεύονται τα παράλληλα συστήματα. Υπάρχουν γλώσσες πολύ ισχυρές αλλά πολύπλοκες και γλώσσες χωρίς μεγάλες δυνατότητες αλλά απλές και εύκολες στην εκμάθηση. Ο προγραμματιστής καλείται να επιλέξει την "καλύτερη" γλώσσα για να υλοποιήσει το πρόγραμμα.

Μπορούμε να ισχυριστούμε με βεβαιότητα ότι μία γλώσσα προγραμματισμού που να είναι αντικειμενικά καλύτερη από τις άλλες δεν υπάρχει, ούτε πρόκειται να υπάρξει.

Η επιλογή της γλώσσας για την ανάπτυξη μιας εφαρμογής εξαρτάται από το είδος της εφαρμογής, το υπολογιστικό περιβάλλον στο οποίο θα εκτελεστεί, τα προγραμματιστικά περιβάλλοντα που διαθέτουμε και κυρίως τις γνώσεις του προγραμματιστή. Συνήθως ο προγραμματιστής επιλέγει μία γλώσσα, που φυσικά επιτρέπει και διευκολύνει την ανάπτυξη του είδους της εφαρμογής στο συγκεκριμένο περιβάλλον με βάση όμως τις προσωπικές του γνώσεις και προτιμήσεις.

6.3 Φυσικές και τεχνητές γλώσσες

Οι γλώσσες προγραμματισμού αναπτύχθηκαν, για να μπορεί ο προγραμματιστής να δίνει τις εντολές που πρέπει να εκτελέσει ο υπολογιστής. Χρησιμοποιούνται δηλαδή για την επικοινωνία του ανθρώπου και της μηχανής, όπως αντίστοιχα οι φυσικές γλώσσες χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων. Οι γλώσσες προγραμματισμού, πού είναι τεχνητές γλώσσες, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις φυσικές γλώσσες.

Μία γλώσσα προσδιορίζεται από το αλφάβητό της, το λεξιλόγιό της, τη γραμματική της και τέλος τη σημασιολογία της.

Το αλφάβητο

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

Για παράδειγμα η ελληνική γλώσσα περιέχει τα εξής στοιχεία: Τα γράμματα του αλφαβήτου πεζά και κεφαλαία 48 δηλαδή χαρακτήρες (Α-Ω και α-ω), τα 10 ψηφία (0-9) και όλα τα σημεία στίξης. Αντίστοιχα η αγγλική γλώσσα περιλαμβάνει τα γράμματα του αγγλικού αλφαβήτου (A-Z και a-z) καθώς και τα ψηφία και όλα τα σημεία στίξης που χρησιμοποιούνται.

Το λεξιλόγιο

Το λεξιλόγιο αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου, τις λέξεις που είναι δεκτές από την 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

Η Γραμματική αποτελείται από το **τυπικό** ή **τυπολογικό** (accidence) και το **συντακτικό** (syntax).

Τυπικό είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μία λέξη είναι αποδεκτή. Για παράδειγμα στην ελληνική γλώσσα οι λέξεις γλώσσα, γλώσσας, γλώσσες 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 γνώση του συντακτικού επιτρέπει τη δημιουργία σωστών προτάσεων στις φυσικές γλώσσες ενώ στις γλώσσες προγραμματισμού τη δημιουργία σωστών εντολών.

Η σημασιολογία

Κάθε γλώσσα προσδιορίζεται από το αλφάβητο της, το λεξιλόγιο της, τη γραμματική της και τη σημασιολογία της.

Η σημασιολογία (Semantics) είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατά επέκταση των εκφράσεων και προτάσεων που χρησιμοποιούνται σε μία γλώσσα.

Στις γλώσσες προγραμματισμού οι οποίες είναι τεχνητές γλώσσες, ο δημιουργός της γλώσσας αποφασίζει τη σημασιολογία των λέξεων της γλώσσας.

Διαφορές φυσικών και τεχνητών γλωσσών.

Μία βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η δυνατότητα εξέλιξής τους. Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις δημιουργούνται, κανόνες γραμματικής και σύνταξης αλλάζουν με την πάροδο του χρόνου και αυτό γιατί η γλώσσα χρησιμοποιείται για την επικοινωνία μεταξύ ανθρώπων, που εξελίσσονται και αλλάζουν ανάλογα με τις εποχές και τον κοινωνικό περίγυρο.

Αντίθετα οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα, αφού κατασκευάζονται συνειδητά για ένα συγκεκριμένο σκοπό.

Ωστόσο συχνά οι γλώσσες προγραμματισμού βελτιώνονται και μεταβάλονται από τους δημιουργούς τους, με σκοπό να διορθωθούν αδυναμίες ή να καλύψουν μεγαλύτερο εύρος εφαρμογών ή τέλος να ακολουθήσουν τις νέες εξελίξεις. Οι γλώσσες προγραμματισμού αλλάζουν σε επίπεδο διαλέκτου (για παράδειγμα GW-Basic και QuickBasic) ή σε επίπεδο επέκτασης (για παράδειγμα Basic και Visual Basic).

6.4 Τεχνικές σχεδίασης προγραμμάτων

Από την αρχή της εμφάνισης των υπολογιστών γίνονται συνεχείς προσπάθειες ανάπτυξης μεθοδολογιών και τεχνικών προγραμματισμού, που θα εξασφαλίζουν τη δημιουργία απλών και κομψών προγραμμάτων, την

εύκολη γραφή τους όσο και την κατανόησή τους.

6.4.1 Ιεραρχική σχεδίαση προγράμματος

*Η **ιεραρχική σχεδίαση** ή **ιεραρχικός προγραμματισμός** χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα*

Η τεχνική της ιεραρχικής σχεδίασης και επίλυσης ή η διαδικασία σχεδίασης "από επάνω προς τα κάτω" όπως συχνά ονομάζεται (top-down program design) περιλαμβάνει τον καθορισμό των βασικών λειτουργιών ενός προγράμματος, σε ανώτερο επίπεδο, και στη συνέχεια τη διάσπαση των λειτουργιών αυτών σε όλο και μικρότερες λειτουργίες, μέχρι το τελευταίο επίπεδο που οι λειτουργίες είναι πολύ απλές, ώστε να επιλυθούν εύκολα.

Σκοπός της ιεραρχικής σχεδίασης είναι η διάσπαση λοιπόν του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.

Για την υποβοήθηση της ιεραρχικής σχεδίασης χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές, όπως για παράδειγμα το διάγραμμα του σχήματος 6.4.

6.4.2 Τμηματικός προγραμματισμός

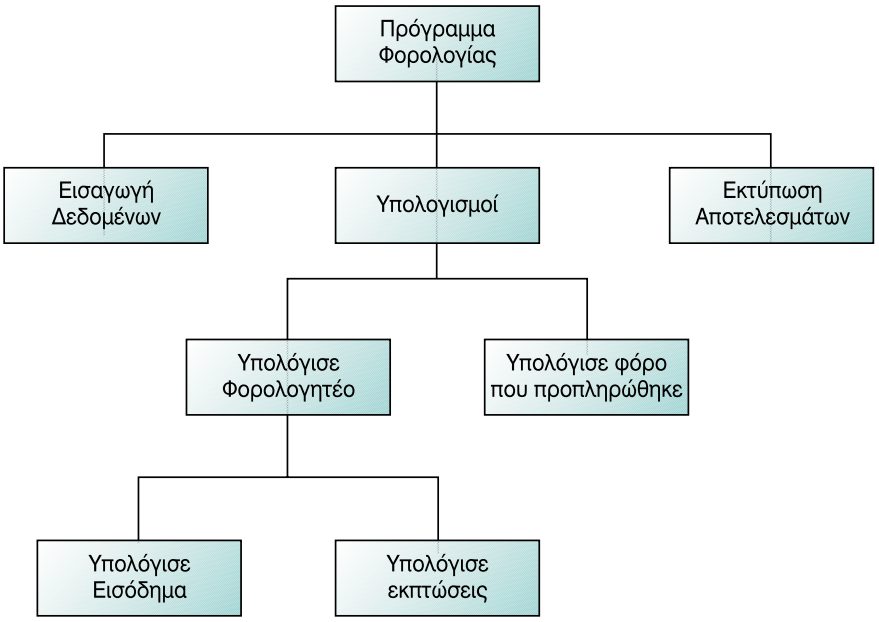
Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον τμηματικό προγραμματισμό. Μετά την ανάλυση του προβλήματος σε αντίστοιχα υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη **ενότητα** (module), που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος.

Η σωστή διαίρεση του αρχικού προβλήματος σε υποπροβλήματα και κατά συνέπεια του αρχικού προγράμματος σε τμήματα προγράμματος είναι μία διαδικασία αρκετά πολύπλοκη και θα εξεταστεί σε επόμενο κεφάλαιο.

Εδώ πρέπει να σημειωθεί ότι ο τμηματικός προγραμματισμός διευκολύνει τη δημιουργία του προγράμματος, μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρηση του προγράμματος από τρίτους.

6.4.3 Δομημένος προγραμματισμός

Η μεθοδολογία που σήμερα έχει επικρατήσει απόλυτα και σχεδόν όλες οι σύγχρονες γλώσσες προγραμματισμού υποστηρίζουν, είναι ο δομημένος



Σχ. 6.4. Éåñáñ÷éêÞ ó÷ääßáóç õðïëïãéóìïý ôïõ öüñïõ åéóïäÞìáôïò

προγραμματισμός (structured programming). Ο δομημένος προγραμματισμός παρουσιάστηκε στα μέσα του 1960.

Συγκεκριμένα το 1964 σε ένα συνέδριο στο Ισραήλ παρουσιάστηκε ένα κείμενο των Bohm και Jacopini με τις θεωρητικές αρχές του δομημένου προγραμματισμού. Οι απόψεις τους δεν έγιναν αρχικά ευρύτερα γνωστές και αποδεκτές, αλλά το 1968 ο καθηγητής Edsger Dijkstra δημοσίευσε ένα κείμενο που έκανε ιδιαίτερη αίσθηση και έμελλε να αλλάξει σταδιακά τον τρόπο προγραμματισμού καθώς και τις ίδιες τις γλώσσες προγραμματισμού. Ο τίτλος της μελέτης αυτής ήταν "GO TO Statement Considered Harmful - η εντολή GOTO θεωρείται επιβλαβής" και θεμελίωνε το δομημένο προγραμματισμό. Χρειάστηκε όμως να περάσουν αρκετά χρόνια, ώστε να αρχίσει να διαδίδεται η χρήση του δομημένου προγραμματισμού.

Την εποχή εκείνη δεν υπήρχε μία μεθοδολογία για την ανάπτυξη των προγραμμάτων, τα προγράμματα ήταν μεγάλα και ιδιαίτερα μπερδεμένα με αποτέλεσμα να ξοδεύεται πάρα πολύς χρόνος τόσο στην συγγραφή όσο κύρια στη διόρθωση και τη μετέπειτα συντήρηση τους. Βασικός λόγος για τα προβλήματα αυτά ήταν η αλόγιστη χρήση μίας εντολής, της εντολής GOTO πού χρησιμοποιούμενη άλλαζε διαρκώς τη ροή του προγράμματος. Ο δομημένος προγραμματισμός αναπτύχθηκε από την ανάγκη να υπάρχει

GOTO: Το μαύρο πρόβατο του προγραμματισμού

Στην ιστορία του προγραμματισμού καμία άλλη εντολή δεν συζητήθηκε τόσο πολύ όσο η εντολή **GOTO** (πήγαινε). Η εντολή GOTO έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη. Η εντολή αυτή χώρισε τους προγραμματιστές σε δύο αντιμαχόμενες ομάδες. Η μία αποτελείτο από φανατικούς υποστηρικτές της χρήσης του GOTO, οι οποίοι με τη χρήση της έλυναν εύκολα και αβασάνιστα προβλήματα της ανάπτυξης των προγραμμάτων τους και η δεύτερη με πολέμιους που έβλεπαν ότι η εντολή αυτή ήταν υπεύθυνη για τη δυσκολία στην αρχική σχεδίαση της λύσης, στην παρακολούθηση και κατανόηση του προγράμματος και τέλος στη συντήρηση. Ο δομημένος προγραμματισμός προήλθε από την ανάγκη του περιορισμού της ανεξέλεγκτης χρήσης του GOTO.

Η χρήση της εντολής αυτής θα παρουσιαστεί με ένα απλό παράδειγμα, ενώ για λόγους σύγκρισης δίνεται ο ψευδοκώδικας με χρήση της δομής επιλογής, όπως παρουσιάστηκε στα προηγούμενα κεφάλαια

```
,,,,,
ÁÍ Áñéèìüò>0 ÔÏÔÅ GOTO 1
ÁÍ Áñéèìüò=0 ÔÏÔÅ GOTO 2
 ÃÑÁØÅ ‘Áñíçôéêüò’
GOTO 4
1:ÃÑÁØÅ ‘Èåôéêüò’
GOTO 4
2: ÃÑÁØÅ ‘ÌçäÝí’
GOTO 4
4:! ÓõíÝ÷åéá,

,,,,
ÁÍ Áñéèìüò>0 ÔÏÔÅ ÃÑÁØÅ ‘Èåôéêüò’
ÁËËÉÙÓ_ÁÍ Áñéèìüò=0  ÔÏÔÅ ÃÑÁØÅ ‘ÌçäÝí’
ÁËËÉÙÓ ÃÑÁØÅ ‘Áñíçôéêüò’
ÔÅËÏÓ_ÁÍ
,,,
```

Η χρήση του GOTO κάνει ακόμα και αυτό το μικρό τμήμα προγράμματος δύσκολο στην κατανόηση του και στην παρακολούθησή του.

⇒

Όλες οι σύγχρονες γλώσσες προγραμματισμού, υποστηρίζουν το δομημένο προγραμματισμό και διαθέτουν εντολές που καθιστούν τη χρήση του GOTO περιττή. Για λόγους όμως συμβατότητας με τις παλιότερες εκδόσεις τους καθώς και για λόγους συντήρησης παλιών προγραμμάτων, μερικές τη διατηρούν στο ρεπερτόριο των εντολών τους.

Στη συνέχεια αυτού του βιβλίου η εντολή GOTO δεν θα μας απασχολήσει και καλό είναι να μη χρησιμοποιείται στην ανάπτυξη προγραμμάτων.

μία κοινή μεθοδολογία στην ανάπτυξη των προγραμμάτων και τη μείωση των εντολών GOTO που χρησιμοποιούνται στο πρόγραμμα.

Ο δομημένος προγραμματισμός δεν είναι απλώς ένα είδος προγραμματισμού, είναι μία 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 μία είσοδο και μόνο μία έξοδο.

Οι τρεις αυτές δομές παρουσιάστηκαν στο κεφάλαιο 2 και θα επαναληφθούν στα επόμενα κεφάλαια.

Αν και ο δομημένος προγραμματισμός αρχικά εμφανίστηκε σαν μία προσπάθεια περιορισμού των εντολών GOTO, σήμερα αποτελεί τη βασική μεθοδολογία προγραμματισμού.

Ο δομημένος προγραμματισμός ενθαρρύνει και βοηθάει την ανάλυση του προγράμματος σε επί μέρους τμήματα, έτσι ώστε σήμερα ο όρος δομημένος προγραμματισμός περιέχει τόσο την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.

Πλεονεκτήματα του δομημένου προγραμματισμού

Επιγραμματικά μπορούμε να αναφέρουμε τα εξής 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.

6.5 Αντικειμενοστραφής προγραμματισμός

Μία νέα ιδέα στον προγραμματισμό γεννήθηκε στις παγωμένες νορβηγικές ακτές στα τέλη της δεκαετίας του ‘70 και πέρασε πολύ γρήγορα στην άλλη μεριά του Ατλαντικού. Πρόκειται για μια νέα τάση αντιμετώπισης προγραμματιστικών αντιλήψεων και δομών που ονομάζεται **αντικειμενοστραφής** (object-oriented) προγραμματισμός. Την τελευταία δεκαετία έχει γίνει η επικρατούσα κατάσταση και έχει αλλάξει ριζικά τα μέχρι πριν από λίγα χρόνια γνωστά και σταθερά σημεία αναφοράς των προγραμματιστών.

Η ιδέα του αντικειμενοστραφούς προγραμματισμού ή της αντικειμενοστραφούς σχεδίασης έχει τις ρίζες της σε πολύ 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. Η απάντηση σε αυτό το ερώτημα προσδιορίζει και τη βασική διαφορά ανάμεσα στις παραδοσιακές προγραμματιστικές τεχνικές και στην αντικειμενοστραφή προσέγγιση.

Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα **αντικείμενα** (objects). Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα

που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα. Βέβαια, δημιουργούνται μία σειρά από εύλογα ερωτήματα, όπως "Τι ακριβώς είναι ένα αντικείμενο;", "Πώς προσδιορίζουμε και περιγράφουμε ένα αντικείμενο;", "Πώς το πρόγραμμα χειρίζεται τα αντικείμενα;" "Πώς τα αντικείμενα συσχετίζονται μεταξύ τους;". Απαντήσεις σε αυτά τα ερωτήματα καθώς και αναλυτική παρουσίαση του αντικειμενοστραφούς προγραμματισμού υπάρχουν στο κεφάλαιο 11.

Φυσικά ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

6.6 Παράλληλος προγραμματισμός

Μία άλλη μορφή προγραμματισμού που αναπτύσσεται τελευταία και πιθανόν στο μέλλον να γνωρίσει μεγάλη άνθηση, είναι ο **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 ιδίου προγράμματος. Οι υπολογιστές αυτοί εμφανίζονται θεωρητικά να πετυχαίνουν ταχύτητες, που είναι 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.

Μια γλώσσα προγραμματισμού που υποστηρίζει παράλληλο προγραμματισμό είναι η OCCAM.

'Οπως αναφέρθηκε και στο κεφάλαιο 5, ο παράλληλος προγραμματισμός αποτελεί μία σημαντική επιστημονική περιοχή, η οποία ξεφεύγει από τα όρια αυτού του βιβλίου.

6.7 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.

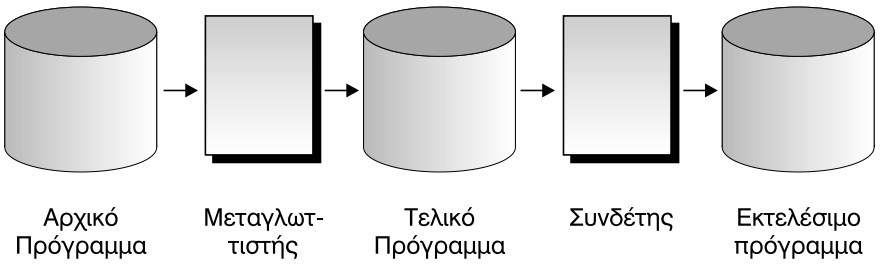
Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προγραμμά-

των, οι **μεταγλωττιστές** (compilers) και οι **διερμηνευτές** (interpreters). Ο μεταγλωττιστής 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. Αντίθετα ο διερμηνευτής διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για κάθε μια εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.

Το αρχικό πρόγραμμα λέγεται **πηγαίο** πρόγραμμα (source), ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται **αντικείμενο** πρόγραμμα (object).

Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατανοητή από τον υπολογιστή, αλλά συνήθως δεν είναι σε θέση να εκτελεστεί. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει 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. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται μεταγλώττιση και σύνδεση.

Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση, που το αρχικό πρόγραμμα δεν περιέχει λάθη. Τις περισσότερες φορές κάθε πρόγραμμα αρχικά θα έχει λάθη. Τα λάθη του προγράμματος είναι γενικά δύο ειδών, λογικά και συντακτικά. Τα λογικά λάθη εμφανίζονται μόνο στην εκτέλεση, ενώ τα συντακτικά λάθη στο στάδιο της μεταγλώττισης.

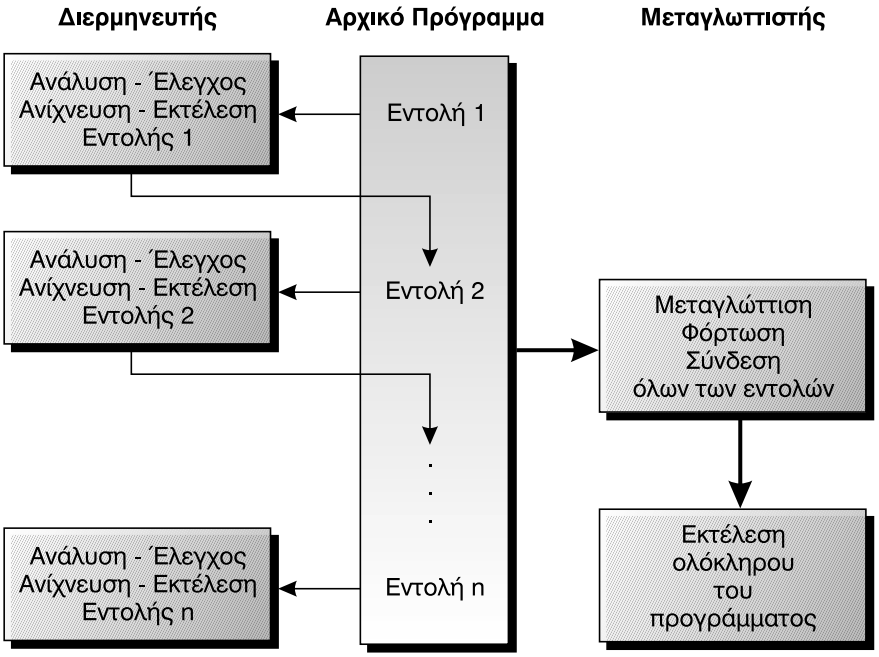


Σχ. 6.5. Ìåôáãëþôôéóç êáé óýíäåóç ðñïãñáììáôïò

Εκτενής παρουσίαση των λαθών και των τρόπων που αντιμετωπίζονται γίνεται σε επόμενο κεφάλαιο. Εδώ αναφέρουμε ότι τα λογικά λάθη που είναι τα πλέον σοβαρά και δύσκολα στη διόρθωση τους, οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου, ενώ τα συντακτικά οφείλονται σε αναγραμματισμούς ονομάτων εντολών, παράληψη δήλωσης δεδομένων και πρέπει πάντα να διορθωθούν, ώστε να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.

Ο μεταγλωττιστής ή ο διερμηνευτής ανιχνεύει λοιπόν τα λάθη και εμφανίζει κατάλληλα διαγνωστικά μηνύματα. Το στάδιο που ακολουθεί είναι η διόρθωση των λαθών. Το διορθωμένο πρόγραμμα επαναϋποβάλεται για μεταγλώττιση και η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου εξαληφθούν πλήρως όλα τα λάθη.

Η χρήση μεταγλωττιστή έχει το μειονέκτημα, ότι προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της μεταγλώττισης και σύνδεσης. Από την άλλη μεριά η χρήση διερμηνευτή έχει το πλεονέκτημα της άμεσης εκτέλεσης και συνεπώς και της άμεσης διόρθωσης. Όμως η εκτέλεση του προγράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής. Πάντως τα σύγχρονα προγραμματιστικά περι-



Σχ. 6.6. Äéáäéêáóßá ìåôÜöñáóçò êáé åêôÝëåóçò åíüò ðñïãñÜììáôïò

Τα σύγχρονα ολοκληρωμένα προγραμματιστικά περιβάλλοντα δεν παρέχουν απλώς ένα μεταφραστή μιας γλώσσας προγραμματισμού. Περιέχουν όλα τα προγράμματα και τα εργαλεία που απαιτούνται και βοηθούν τη συγγραφή, την εκτέλεση και κύρια τη διόρθωση των προγραμμάτων.

βάλλοντα 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). Ο συντάκτης είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου, με δυνατότητες όμως που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων

Για τη δημιουργία, τη μετάφραση και την εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον τρία προγράμματα: ο συντάκτης, ο μεταγλωττιστής και ο συνδέτης. Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν αυτά τα προγράμματα με ενιαίο τρόπο.

Το κάθε προγραμματιστικό περιβάλλον έχει φυσικά διαφορετικά εργαλεία και ιδιότητες. Για παράδειγμα ένα περιβάλλον οπτικού (visual) προγραμματισμού πρέπει να περιέχει οπωσδήποτε και ειδικό συντάκτη που να διευκολύνει τη δημιουργία γραφικών αντικειμένων (για παράδειγμα φόρμες, λίστες, παράθυρα διαλόγου) παρέχοντας στον προγραμματιστή τα αντίστοιχα γραφικά εργαλεία.

Ανακεφαλαίωση

Δημιουργία προγράμματος είναι η μετατροπή του αλγορίθμου που επιλύει ένα πρόβλημα σε εντολές προγράμματος. Οι εντολές γράφονται σε κάποια από τις εκατοντάδες γλώσσες προγραμματισμού, που έχουν αναπτυχθεί με σκοπό να διευκολύνουν την επίλυση συγκεκριμένου τύπου προβλημάτων. Η επιλογή της καταλληλότερης γλώσσας εξαρτάται από το είδος της εφαρμογής, το υπολογιστικό περιβάλλον που θα εκτελεστεί, τις γνώσεις και τις προτιμήσεις του προγραμματιστή.

Οι τεχνικές που χρησιμοποιούνται για την ανάπτυξη προγραμμάτων είναι της ιεραρχικής σχεδίασης, του τμηματικού προγραμματισμού και του δομημένου προγραμματισμού, που επιτρέπουν τη δημιουργία προγραμμάτων κατανοητών και απλών, ενώ διευκολύνουν τη διόρθωση και τη συντήρηση των εφαρμογών. Ένα είδος προγραμματισμού που γνωρίζει ιδιαίτερη άνθηση τελευταία, είναι ο αντικειμενοστραφής προγραμματισμός.

Κάθε πρόγραμμα για να εκτελεστεί από τον υπολογιστή χρειάζεται πρώτα να μετατραπεί σε μορφή κατανοητή από αυτόν. Η μετατροπή αυτή γίνεται από τους μεταγλωττιστές ή τους διερμηνευτές, οι οποίοι επιση-

μαίνουν και τα συντακτικά λάθη, που έχει κάθε πρόγραμμα. Η σύνταξη του προγράμματος, η μετάφραση, η διόρθωση των λαθών και η εκτέλεση του γίνεται με τα ολοκληρωμένα προγραμματιστικά περιβάλλοντα που διαθέτουν πολλά εργαλεία για την υποβοήθηση της ανάπτυξης των εφαρμογών.

Ερωτήσεις - Θέματα για συζήτηση

- Τι ονομάζεται πρόγραμμα;
- Τι είναι οι γλώσσες μηχανής;
- Ποιες οι διαφορές των γλωσσών υψηλού επιπέδου από αυτές χαμηλού επιπέδου;
- Ποιες γλώσσες υψηλού επιπέδου γνωρίζεις;
- Τι ονομάζουμε οπτικό προγραμματισμό και τι οδηγούμενο από τα γεγονότα;
- Πώς προσδιορίζεται μία φυσική γλώσσα;
- Ποιες οι κυριότερες διαφορές των φυσικών και των τεχνητών γλωσσών;
- Πώς γίνεται η παράσταση της ιεραρχικής σχεδίασης προγράμματος;
- Ποιες οι αρχές του δομημένου προγραμματισμού;
- Ποια τα πλεονεκτήματα του δομημένου προγραμματισμού;
- Τι ονομάζεται αντικειμενοστραφής προγραμματισμός;
- Ποια η διαδικασία για την μετάφραση και εκτέλεση ενός προγράμματος;
- Ποιες οι διαφορές μεταγλωττιστή και διερμηνευτή;
- Ποια προγράμματα και εργαλεία περιέχει ένα προγραμματιστικό περιβάλλον;

Λέξεις κλειδιά

Πρόγραμμα, Γλώσσα μηχανής, Συμβολική γλώσσα, Γλώσσες υψηλού επιπέδου, Τμηματικός προγραμματισμός, Δομημένος προγραμματισμός, Αντικειμενοστραφής προγραμματισμός, Μεταγλωττιστής, Διερμηνευτής, Προγραμματιστικό περιβάλλον

Βιβλιογραφία

1. Ph. Breton, *Ιστορία της Πληροφορικής,* Εκδόσεις Δίαυλος, Αθήνα,
2. Γ. Μπαμπινιώτης, *Θεωρητική Γλωσσολογία,* Αθήνα, 1986.
3. Χρ. Κοίλιας-Στρ, Καλαφατούδης, *Το πρώτο βιβλίο της Πληροφορικής,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.
4. *Εγκυκλοπαίδεια Πληροφορικής και Τεχνολογίας Υπολογιστών,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1986.
5. Αθ.Τσουροπλής-Στ.Κλημόπουλος, *Από τη FORTRAN 77 στη FORTRAN 90,* Εκδόσεις Πελεκάνος, Αθήνα, 1995.
6. Χ. Κοίλιας-Στρ. Μαραγκός, *Η γλώσσα COBOL και οι εφαρμογές της,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, Αθήνα, 1992.
7. Κ. Μαρινάκης-Ν. Ιωαννίδης, *Structure & Advanced COBOL,* Εκδόσεις 'Ελιξ, Αθήνα, 1992.
8. Χ. Κοίλιας-Αλ. Τομαράς, *GWBASIC Θεωρία και Εφαρμογές,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.
9. Μ. Κατζουράκη-Μ. Γεργατσούλης-Σ. Κόκκοτος, *PROγραμματίζοντας στη LOGική,* 'Εκδοση ΕΠΥ, Αθήνα, 1991.
10. Αικ. Γεωργοπούλου, *LOGO Βήμα προς Βήμα,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1991.
11. Αλ. Τομαράς, *C Θεωρία και Πράξη,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1995.
12. Μ.Μαλιάππης, *SQL Περιβάλλοντα Ανάπτυξης Εφαρμογών 4ης Γενιάς,* Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1995.
13. E. Horowitz, *Βασικές αρχές γλωσσών προγραμματισμού,* Κλειδάριθμος, Αθήνα, 1995.
14. R. Shackelford, *Introduction to Computing and Algorithms,* Addison-Wesley, USA, 1998.
15. W. Hutching-H. Somers, *An Introduction to Machine Translation,* Academic Press, London, 1992.

Διευθύνσεις Διαδικτύου

- cuiwww.unige.ch/langlist

Κατάλογος όλων των γλωσσών προγραμματισμού που υπάρχουν. Περιέχει περισσότερες από 2000 γλώσσες και ενημερώνεται συνεχώς.

- www.swcp.com/~dodrill/

Περιέχει πληροφορίες αλλά και πολλές εκπαιδευτικές ασκήσεις για διάφορες γλώσσες προγραμματισμού.

- www.progsource.com

Γενικές πληροφορίες, πολλές εφαρμογές, χρήσιμα βοηθητικά προγράμματα καθώς και αναφορές σε άλλες διευθύνσεις για πολλές γλώσσες προγραμματισμού όπως Pascal, Delphi, C/C++, Java, Perl, Visual Basic.

- www.hensa.ac.uk/parallel/

Πληροφορίες για τον παράλληλο προγραμματισμό και τις γλώσσες που υποστηρίζουν τον παράλληλο προγραμματισμό.

- Softwaredesign.com/objects.html

Γενικές συνοπτικές πληροφορίες για το τι είναι αντικειμενοστραφής προγραμματισμός και τα βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού.

- Iamwww.unibe.ch/~scg/Ooinfo

Στοιχεία για τον αντικειμενοστραφή προγραμματισμό, τις γλώσσες που χρησιμοποιούνται και πολλές σχετικές διευθύνσεις.