ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ
ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ

ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ
ΠΟΛΙΤΙΚΗΣ

Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ

Β΄ Λυκείου

ΤΟΜΟΣ 3ος

Αθήνα 2014

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ
ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ
«ΔΙΟΦΑΝΤΟΣ»

ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ
Πρόεδρος: **Σωτήριος Γκλαβάς**

ΓΡΑΦΕΙΟ ΕΡΕΥΝΑΣ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΕΦΑΡΜΟΓΩΝ Β΄
Προϊστάμενος:
Παύλος Φ. Μάραντος

ΣΥΓΓΡΑΦΕΙΣ:
Δρ. **Σπυρίδων Δουκάκης**, Πληροφορικός, Μαθηματικός, PIERCE-Αμερικανικό Κολλέγιο Ελλάδος
Χρήστος Δουληγέρης, Καθηγητής Τμήματος Πληροφορικής Πανεπιστημίου Πειραιώς
Δρ. **Θεόδωρος Καρβουνίδης**, Εκπαιδευτικός ΠΕ19
Χρήστος Κοίλιας, Καθηγητής Τμήματος Μηχανικών Πληροφορικής Τ.Ε. ΤΕΙ Αθήνας

Δρ. Αθανάσιος Πέρδος, Πληροφορικός, Φυσικός, Ελληνογαλλική Σχολή Καλαμαρί

ΣΥΝΤΟΝΙΣΤΗΣ: Χρήστος Κοίλιας

ΣΥΛΛΟΓΗ - ΕΠΕΞΕΡΓΑΣΙΑ ΥΛΙΚΟΥ:
Δημήτριος Κοτσιφάκος, Εκπαιδευτικός, ΠΕ 1708

ΚΡΙΤΕΣ-ΑΞΙΟΛΟΓΗΤΕΣ:
Παναγιώτης Βαρζάκας, Μέλος ΔΕΠ (συντονιστής)
Σοφία Τζελέπη, Σχολική Σύμβουλος, ΠΕ19
Πέτρος Ματζάκος, Εκπαιδευτικός, ΠΕ19

ΦΙΛΟΛΟΓΙΚΗ ΕΠΙΜΕΛΕΙΑ:
Μαρία Κοίλια

ΕΞΩΦΥΛΛΟ: Γιώργος Σκούφος

ΣΕΛΙΔΟΠΟΙΗΣΗ: Γιώργος Σκούφος

ΑΝΑΔΟΧΟΣ: ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ

Στουρνάρη 49Α, 106 82,
Αθήνα, Τηλ. 210-38.45.594
Fax: 210-38.08.009
E-mail:
contact@newtech-publications.gr
URL:
www.newtech-pub.com

«ΔΗΜΙΟΥΡΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΓΙΑ ΤΑ ΝΕΑ ΜΑΘΗΜΑΤΑ ΤΟΥ ΓΕΝΙΚΟΥ ΛΥΚΕΙΟΥ» της Πράξης «ΝΕΟ ΣΧΟΛΕΙΟ (ΣΧΟΛΕΙΟ 21ου αιώνα) - ΝΕΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ» ΜΕ ΚΩΔ. ΟΠΣ 295450, των Αξόνων Προτεραιότητας 1, 2 και 3 - ΟΡΙΖΟΝΤΙΑ ΠΡΑΞΗ του ΕΠΙΧΕΙΡΗΣΙΑΚΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ «ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ», που συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση - Ευρωπαϊκό Κοινωνικό Ταμείο και από Εθνικούς Πόρους (ΕΣΠΑ 2007 - 2013).

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Η αξιολόγηση, η κρίση των προσαρμογών και η επιστημονική επιμέλεια του προσαρμοσμένου βιβλίου πραγματοποιείται από τη Μονάδα Ειδικής Αγωγής του Ινστιτούτου Εκπαιδευτικής Πολιτικής.

Η προσαρμογή του βιβλίου για μαθητές με μειωμένη όραση από το ΙΤΥΕ – ΔΙΟΦΑΝΤΟΣ πραγματοποιείται με βάση τις προδιαγραφές που έχουν αναπτυχθεί από ειδικούς εμπειρογνώμονες για το ΙΕΠ.

ΠΡΟΣΑΡΜΟΓΗ ΤΟΥ ΒΙΒΛΙΟΥ ΓΙΑ ΜΑΘΗΤΕΣ ΜΕ ΜΕΙΩΜΕΝΗ ΟΡΑΣΗ

ΙΤΥΕ - ΔΙΟΦΑΝΤΟΣ

Προγραμματισμός

Στόχοι του κεφαλαίου αυτού είναι να μπορούν οι μαθητές:

- ✓ **να δημιουργούν ευκρινές γνωσιακό και οργανωμένο νοητικό σχήμα που να περιλαμβάνει τα είδη και τεχνικές προγραμματισμού, με βάση την πρότερη εμπειρία τους.**
- ✓ **να συνδυάζουν αλγοριθμικές δομές και δεδομένα/δομές δεδομένων για να δημιουργούν κώδικα/πρόγραμμα.**
- ✓ **να διαπιστώνουν ότι οι σημερινές εφαρμογές είναι αρκετά**

πολύπλοκες και η δημιουργία τους ακολουθεί συγκεκριμένα μοντέλα ανάπτυξης εφαρμογών λογισμικού που εξελίσσονται σε συγκεκριμένες φάσεις.

Προερωτήσεις

- Η δημιουργία του αλγορίθμου αρκεί για να επιλύσεις ένα πρόβλημα στον υπολογιστή;
- Γνωρίζεις κάποιες γλώσσες προγραμματισμού;
- Έχεις ακολουθήσει κάποια μεθοδολογία ή τεχνική για να επιλύσεις ένα πρόβλημα;
- Ποια νομίζεις ότι είναι η δουλειά του προγραμματιστή υπολογιστών;

2.3.1 Αναφορά σε γλώσσες προγραμματισμού και «Προγραμματιστικά Υποδείγματα»

2.3.1.1 Πρόγραμμα και Γλώσσες Προγραμματισμού

Για να αναπαρασταθούν οι αλγόριθμοι σε μορφή κατανοητή από τον υπολογιστή αναπτύσσονται προγράμματα.

Πρόγραμμα είναι το σύνολο των εντολών που χρειάζεται να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος.

Η εργασία σύνταξης των προγραμμάτων σε κάποια γλώσσα προγραμματισμού ονομάζεται **προγραμματισμός** και τα άτομα που γράφουν και συντάσσουν ένα πρόγραμμα ονομάζονται **προγραμματιστές.** Βασικό στοιχείο του προγράμματος, εκτός από τον αλγόριθμο που υλοποιεί, είναι τα δεδομένα και οι δομές δεδομένων που επεξεργάζεται.

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου (προγραμματιστή) με τη μηχανή (υπολογιστή). Ο υπολογιστής κάνει στοιχειώδεις ενέργειες σε ακολουθίες των δύο ψηφίων 0 και 1 (**δυαδικά ψηφία,** bits), αλλά αυτές τις ενέργειες τις

εκτελεί με ασύλληπτη ταχύτητα. Συγκεκριμένα μπορεί να αποθηκεύει στη μνήμη τις ακολουθίες των δυαδικών ψηφίων, να τις ανακτά, να κάνει στοιχειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει.

Αρχικά τα προγράμματα γράφονταν σε **γλώσσα μηχανής**, δηλαδή ακολουθίες δυαδικών ψηφίων, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες. Ο συγκεκριμένος τρόπος γραφής προγραμμάτων είναι επίπονος και ελάχιστοι μπορούν να τον κατανοήσουν και να τον υλοποιήσουν, αφού απαιτεί βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.

Εντολές σε γλώσσα μηχανής που καταχωρούν το άθροισμα των τιμών δύο θέσεων μνήμης σε μία άλλη.

```
0000001001011010
0000101001011110
0000011011011110
```

Η περιγραφή των παραπάνω εντολών είναι η εξής:

- **Μετάφερε στον καταχωρητή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011010.**
- **Πρόσθεσε στο περιεχόμενο του καταχωρητή το περιεχόμενο της θέσης μνήμης με διεύθυνση 01011110.**

- **Μετάφερε και αποθήκευσε το περιεχόμενο του καταχωρητή στη θέση μνήμης με διεύθυνση 11011110.**

Στη συνέχεια αναπτύχθηκαν οι συμβολικές γλώσσες οι οποίες κάνουν χρήση εντολών που αποτελούνται από συμβολικά ονόματα τα οποία αντιστοιχούν σε εντολές της γλώσσας μηχανής. Το έργο της μετάφρασης των εντολών σε γλώσσα μηχανής το αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (assembler).

Οι συμβολικές γλώσσες ήταν σαφώς μια εξέλιξη αλλά παραμένουν στενά συνδεδεμένες με την αρχιτεκτονική

του κάθε υπολογιστή. Επιπλέον η έλλειψη εντολών σύνθετων λειτουργιών στις παραπάνω γλώσσες οδηγεί σε μακροσκελή προγράμματα που είναι δύσκολο να γραφούν και να συντηρηθούν. Ακόμη, δεν είναι δυνατό να μεταφερθούν και να εκτελεστούν σε υπολογιστή διαφορετικής αρχιτεκτονικής.

Εντολές σε **συμβολική γλώσσα** που καταχωρούν το άθροισμα των τιμών δύο θέσεων μνήμης σε μία άλλη.

```
LDA B
ADD C
STA A
```

Η περιγραφή των παραπάνω εντολών είναι η εξής:

- Μετάφερε στον καταχωρητή το περιεχόμενο της θέσης μνήμης με όνομα Β.
- Πρόσθεσε στο περιεχόμενο του καταχωρητή το περιεχόμενο της θέσης μνήμης με όνομα C.
- Μετάφερε και αποθήκευσε το περιεχόμενο του καταχωρητή στη θέση μνήμης με όνομα Α.

Οι παραπάνω ανεπάρκειες και η προσπάθεια για καλύτερη επικοινωνία ανθρώπου – μηχανής οδήγησαν στην εμφάνιση των **γλωσσών υψηλού επιπέδου**. Σε σχέση με τις συμβολικές γλώσσες στις γλώσσες

υψηλού επιπέδου:

- είναι φυσικότερος και πιο ανθρώπινος ο τρόπος έκφρασης των προβλημάτων.
- υπάρχει δυνατότητα μεταφοράς, «μεταφερσιμότητα» δηλαδή, εκτέλεσης των προγραμμάτων σε οποιοδήποτε υπολογιστή.
- είναι εύκολη η εκμάθηση, η διόρθωση των λαθών και η συντήρηση των προγραμμάτων.

Έτσι αναπτύχθηκαν γλώσσες όπως οι ακόλουθες:

- **FORTRAN** (FORrmula TRANslation, Μετάφραση Τύπων). Το 1957 η IBM ανέπτυξε

την πρώτη γλώσσα υψηλού επιπέδου. Αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων.

- **COBOL** (COmmon Business Oriented Language, Κοινή Γλώσσα Προσανατολισμένη στις Επιχειρήσεις). Κατάλληλη για ανάπτυξη εμπορικών και γενικά διαχειριστικών εφαρμογών. Χρησιμοποιείται από επιχειρήσεις και από τη δημόσια διοίκηση.

- **LISP** (LISt Processor, Επεξεργαστής Λίστας). Συναρτησιακή γλώσσα η οποία προσανατολίζεται σε χειρισμό λιστών από σύμβολα. Χρησιμοποιείται στο χώρο της τεχνητής νοημοσύνης, σε

έμπειρα συστήματα, παιχνίδια, επεξεργασία φυσικών γλωσσών.

- **PROLOG** (PROgramming in LOGic, Λογικός Προγραμματισμός). Η γλώσσα PROLOG χρησιμοποιεί μεθόδους λογικής για να αναπαραστήσει τη γνώση και να επιλύσει προβλήματα. Χρησιμοποιείται όπως και η LISP στο χώρο της τεχνητής νοημοσύνης, σε έμπειρα συστήματα, παιχνίδια, επεξεργασία φυσικών γλωσσών.

- **BASIC** (Beginner's All Purpose Symbolic Instruction Code, Συμβολικός Κώδικας Εντολών Γενικής Χρήσης για Αρχάριους). Γλώσσα που αναπτύχθηκε για την εκπαίδευση αρχαρίων στον

προγραμματισμό. Σχεδιάστηκε για να γράφονται σύντομα προγράμματα τα οποία εκτελούνται με τη βοήθεια διερμηνευτή. Η ανάπτυξη των μικροϋπολογιστών και η τυποποίησή της από τη Microsoft, την καθιέρωσε ως πρότυπο για ανάπτυξη εφαρμογών σε προσωπικούς υπολογιστές.

- **PASCAL.** Είναι μια γλώσσα γενικής χρήσης. Διέπεται από τις αρχές του δομημένου προγραμματισμού. Γνώρισε τεράστια εξάπλωση, και επηρέασε την ανάπτυξη άλλων γλωσσών όπως η ADA.

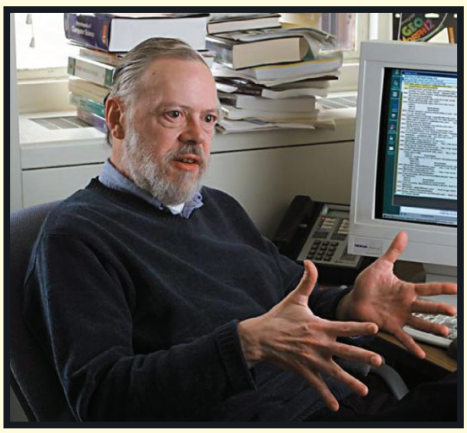
- **C και η μετεξέλιξη της C++.** Γλώσσα η οποία δημιουργήθηκε

στα τα εργαστήρια BELL και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος UNIX. Είναι γλώσσα με ισχυρά χαρακτηριστικά. Η C++ είναι γλώσσα αντικειμενοστρεφούς προγραμματισμού.

- **JAVA.** Γλώσσα αντικειμενοστρεφής που αναπτύχθηκε από τη SUN με σκοπό την ανάπτυξη εφαρμογών για το διαδίκτυο.

Εντολή στη **γλώσσα BASIC** που καταχωρεί το άθροισμα των τιμών δύο μεταβλητών B και C στη μεταβλητή A.

A = B + C



Εικόνα 2.25

Ντένις Ρίτσι (Dennis Ritchie). Ο δημιουργός της γλώσσας προγραμματισμού C και ο βασικός συντελεστής στην ανάπτυξη του λειτουργικού συστήματος UNIX.

«Το UNIX είναι βασικά ένα απλό λειτουργικό σύστημα, πρέπει όμως να είσαι ιδιοφυία για να καταλάβεις την απλότητά του».

Η εμφάνιση των γραφικών περιβαλλόντων εργασίας δημιούργησε την

ανάγκη για προγράμματα που να εκμεταλλεύονται τον γραφικό τρόπο επικοινωνίας χρήστη – υπολογιστή. Έτσι γλώσσες όπως η BASIC, η C++, η PASCAL που είναι μεν κειμενικές, εξελίχθηκαν (Visual Basic, Visual C++, Delphi) ώστε να διαθέτουν και οπτικό περιβάλλον προγραμματισμού.

Σε ένα οπτικό περιβάλλον προγραμματισμού υπάρχει η δυνατότητα να δημιουργείται γραφικά ολόκληρο το περιβάλλον της εφαρμογής, όπως για παράδειγμα τα πλαίσια διαλόγου ή τα μενού.

Αναπτύχθηκαν όμως και γλώσσες όπως η **SCRATCH – BYOB** και η **Google AppInvertor** που είναι αποκλειστικά οπτικές γλώσσες προγραμματισμού (Visual Programming Languages, VPL). Οι συγκεκριμένες δίνουν τη δυνατότητα στον προγραμματιστή να δημιουργήσει προγράμματα μέσα από το γραφικό χειρισμό προγραμματιστικών στοιχείων (αντί κειμένου).

Αρχικά υπήρχαν τα περιβάλλοντα Γραμμής Εντολών, όπου ο χρήστης είχε τη δυνατότητα να πληκτρολογεί τις εντολές και να τις βλέπει στην οθόνη ενώ στα γραφικά περιβάλλοντα εργασίας τα προγράμματα και οι πληροφορίες εμφανίζονται στην οθόνη με γραφικά και σχήματα.

Εικόνα 2.26
Λογότυπο της γλώσσας Google AppInvertor.

Η χρήση των υπολογιστών σχεδόν σε όλες τις εκφάνσεις της ανθρώπινης δραστηριότητας δημιούργησε την ανάγκη για γλώσσες κατάλληλες στην επίλυση συγκεκριμένων προβλημάτων. Έτσι αναπτύχθηκαν γλώσσες όπως η LOGO ή η GameMaker για εκπαιδευτικούς σκοπούς, η LabView που χρησιμοποιείται από τους επιστήμονες και τους μηχανικούς στο σχεδιασμό, τον έλεγχο και τη δοκιμή καταναλωτικών προϊόντων κ.ά..

Οι γλώσσες υψηλού επιπέδου χαρακτηρίζονται ως γλώσσες τρίτης γενιάς ενώ οι συμβολικές ως δεύτερης γενιάς ή χαμηλού επιπέδου. Από την άλλη ένα πρόγραμμα σε γλώσσα μηχανής είναι κωδικοποιημένο σε γλώσσα πρώτης γενιάς.

Οι παραπάνω γενιές γλωσσών προγραμματισμού απευθύνονται μόνο σε προγραμματιστές και ο χρήστης δεν έχει 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 τέταρτης γενιάς. Η **SQL** (Structured Query Language, Δομημένη Γλώσσα Ερωτοαποκρίσεων) είναι μία γλώσσα τέταρτης γενιάς η οποία χρησιμοποιείται για την ανάκτηση και τη διαχείριση δεδομένων καθώς και την παραγωγή πληροφοριών σε σχεσιακές βάσεις δεδομένων.

```
SELECT ENAME, JOB, SAL
FROM EMPLOYES
WHERE DEPTNO = 20
AND SAL > 1000;
```

Με την ερώτηση αυτή σε SQL εκτελείται αναζήτηση στη βάση δεδομένων EMPLOYES και επιστρέφει το όνομα, τη θέση και τον μισθό των υπαλλήλων της διεύθυνσης 20 που κερδίζουν πάνω από 1000 ευρώ.

2.3.1.2 Προγραμματιστικά Υποδείγματα

Αναφέρθηκε προηγουμένως ότι κάποιες γλώσσες ακολουθούν τον αντικειμενοστρεφή προγραμματισμό και άλλες είναι συναρτησιακές ή χρησιμοποιούν μεθόδους λογικής για να επιλύσουν προβλήματα. Η ανάπτυξη λοιπόν ενός προγράμματος σε κάποια γλώσσα προγραμματισμού βασίζεται σε ένα πρότυπο ή μία καθορισμένη μεθοδολογία.

Ως «Προγραμματιστικό Υπόδειγμα» εννοείται ένα πρότυπο ανάπτυξης προγραμμάτων, δηλαδή μία καθορισμένη μεθοδολογία με βάση την οποία αναπτύσσονται η δομή και τα στοιχεία του προγράμματος.

Οι δυνατότητες και οι μεθοδολογίες ανάπτυξης προγραμμάτων που παρέχει μία γλώσσα προγραμματισμού, καθορίζονται από το προγραμματιστικό υπόδειγμα που ακολουθεί. Υπάρχουν όμως γλώσσες που έχουν σχεδιαστεί να υποστηρίζουν περισσότερα από ένα υποδείγματα.

Τα κυριότερα προγραμματιστικά υποδείγματα είναι:

- **Ο προστακτικός προγραμματισμός** όπου τα προγράμματα αναπτύσσονται με απλές εντολές σε προστακτική (Διάβασε, Εμφάνισε, Επανάλαβε) που ζητούν από τον υπολογιστή να εκτελέσει συγκεκριμένες ενέργειες και να ακολουθήσει βήματα με μία λογική σειρά για να επιλύσει το πρόβλημα που έχει δοθεί. Γλώσσες, όπως η FORTRAN, η BASIC, η PASCAL, η C, ακολουθούν αυτό το υπόδειγμα.

- **Ο δηλωτικός προγραμματισμός** όπου, σε αντίθεση με τον προστακτικό προγραμματισμό, το πρόβλημα επιλύεται δηλώνοντας απλώς τις επιθυμητές ιδιότητες του αποτελέσματος. Το πρόγραμμα περιέχει λογικές εκφράσεις,

ενώ κατά την εκτέλεσή του γίνεται έλεγχος για το ποιες ακριβώς ικανοποιούνται. Παραδείγματα γλωσσών που τον ακολουθούν είναι η PROLOG και η SQL.

- Ο **συναρτησιακός προγραμματισμός** επιλύει το πρόβλημα με τη χρήση μαθηματικών συναρτήσεων. Οι συναρτήσεις παράγουν αποτελέσματα με βάση τα δεδομένα εισόδου τους. Παράδειγμα συναρτησιακής γλώσσας είναι η LISP.

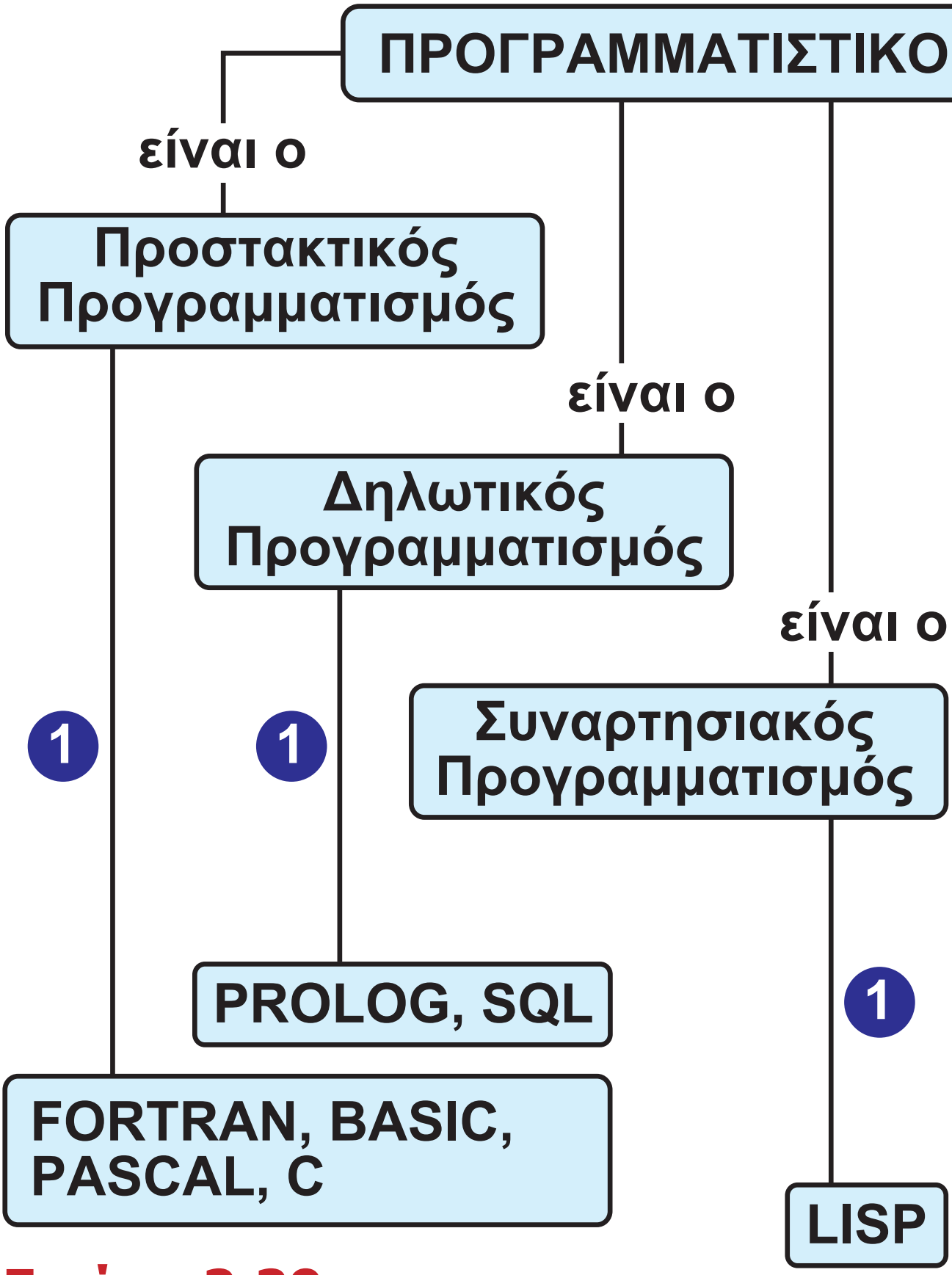
- Ο **αντικειμενοστρεφής προγραμματισμός** βασίζεται στην έννοια του αντικειμένου. Τα αντικείμενα δημιουργούνται από τις κλάσεις. Μία κλάση ορίζει τα χαρακτηριστικά και τη συμπεριφορά ενός

τύπου αντικειμένου, λειτουργεί δηλαδή ως πρότυπο. Ένα αντικείμενο είναι μία δομή δεδομένων η οποία περιέχει τόσο τα δεδομένα (χαρακτηριστικά που την περιγράφουν) όσο και τις διαδικασίες (μεθόδους) που επενεργούν σε αυτά. Τα αντικείμενα μπορούν να αλληλεπιδρούν μεταξύ τους. Αντικειμενοστραφείς γλώσσες είναι η C++ και η JAVA.

- **Ο λογικός προγραμματισμός** όπου τα προγράμματα είναι γραμμένα ως ένα σύνολο από προτάσεις σε μορφή λογικών εκφράσεων. Το συγκεκριμένο υπόδειγμα βασίζεται στα γεγονότα, στους κανόνες και στις ερωτήσεις και ακολουθείται κυρίως στο πεδίο της Τεχνητής Νοημοσύνης.

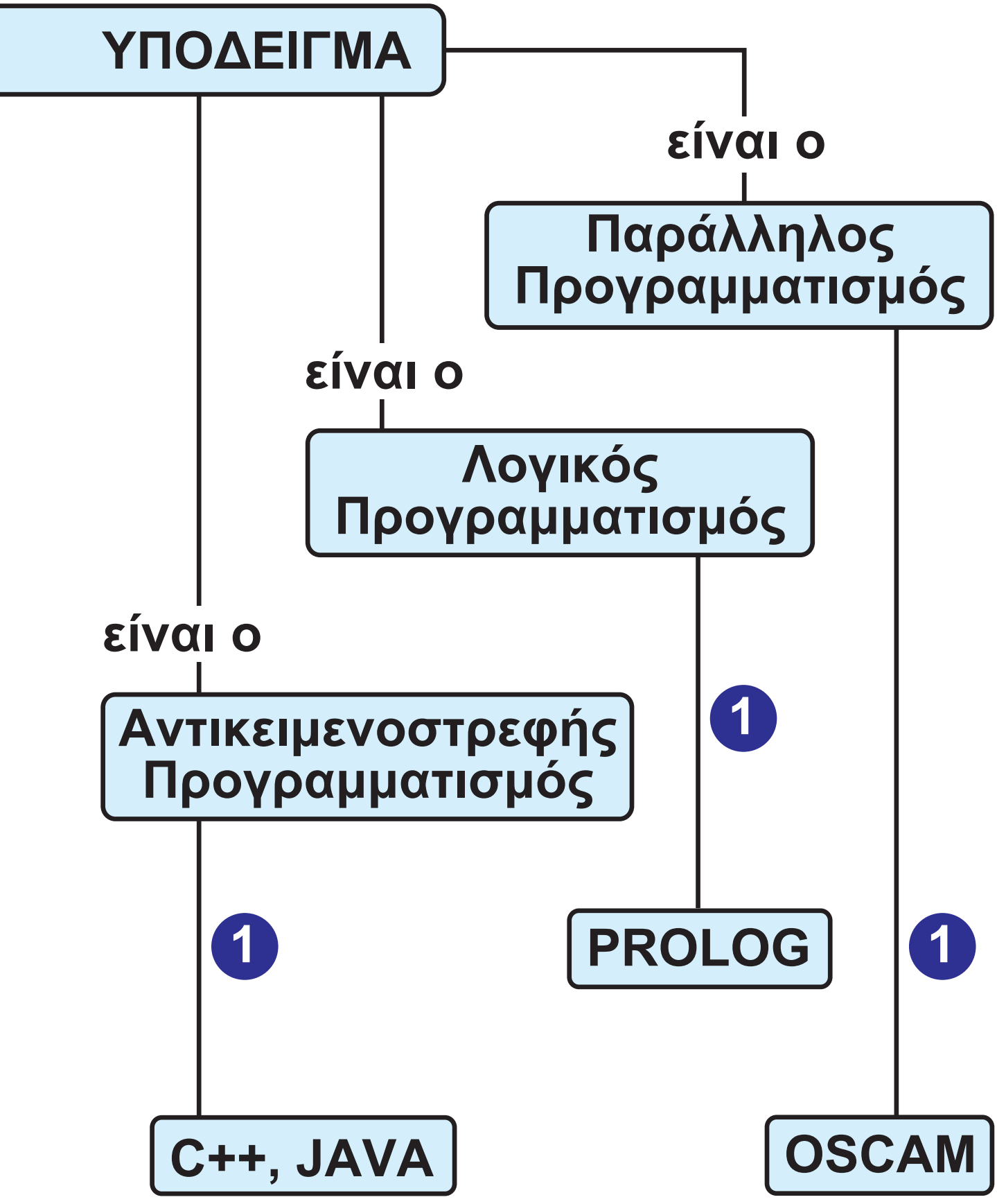
Παράδειγμα γλώσσας που τον ακολουθεί είναι η PROLOG.

- **Ο παράλληλος προγραμματισμός** στον οποίο τα προγράμματα εκμεταλλεύονται την ύπαρξη υπολογιστών που διαθέτουν περισσότερους από έναν επεξεργαστές. Έτσι επιτυγχάνεται η αύξηση των υπολογιστικών επιδόσεων και η μείωση του χρόνου εκτέλεσης της εφαρμογής. Θα πρέπει όμως το πρόβλημα προς επίλυση να διαιρεθεί σε τμήματα που μπορούν να επιλυθούν παράλληλα. Μία γλώσσα που υποστηρίζει τον παράλληλο προγραμματισμό είναι η Occam.



Εικόνα 2.29

Προγραμματιστικά Υποδείγματα.


ΥΠΟΔΕΙΓΜΑ
είναι ο
Παράλληλος Προγραμματισμός
είναι ο
Λογικός Προγραμματισμός
είναι ο
Αντικειμενοστρεφής Προγραμματισμός
1
1
1
PROLOG
C++, JAVA
OSCAM
1 ακολουθείται από

Εικόνα 2.27

Η Άντα Λάβλεϊς (Ada Lovelace), κόρη του Λόρδου Βύρωνα, έγραψε το πρώτο πρόγραμμα υπολογιστή κατά τον 19ο αιώνα για την Αναλυτική Μηχανή του Τσαρλς Μπάμπατζ (Charles Babbage), πολύ πριν από την εμφάνιση ηλεκτρονικών υπολογιστών. Η γλώσσα προγραμματισμού ADA έχει ονομαστεί έτσι προς τιμήν της.

Εικόνα 2.28

Γκρέις Χόπερ (Grace Hopper). Αμερικανίδα καθηγήτρια μαθηματικών και αξιωματικός του αμερικανικού ναυτικού, η οποία ηγήθηκε της ομάδας που ανέπτυξε την πρώτη προηγμένη γλώσσα προγραμματισμού με προορισμό τον επιχειρηματικό κόσμο, την Common Business Oriented Language (COBOL)

Εικόνα 2.30

Νίκλαους Βιρθ (Niklaus Wirth)

Δημιουργός της γλώσσας PASCAL, η οποία ονομάστηκε έτσι προς τιμήν του Γάλλου επιστήμονα Μπλεζ Πασκάλ (Blaise Pascal).

2.3.1.3 Δομημένος Προγραμματισμός

Μία μεθοδολογία ανάλυσης, σχεδίασης και συγγραφής προγραμμάτων που αναπτύχθηκε και διαδόθηκε ευρύτατα στον χώρο της πληροφορικής είναι ο δομημένος προγραμματισμός (structured programming) ο οποίος χρησιμοποιεί

- **την ιεραρχική σχεδίαση για την ανάπτυξη του αλγορίθμου που επιλύει το πρόβλημα.**
- **τον τμηματικό προγραμματισμό για τη σχεδίαση του προγράμματος και για τη δημιουργία των ενοτήτων του.**
- **τρεις βασικές συνιστώσες για τη συγγραφή των επιμέρους**

ενοτήτων που καθιστούν άσκοπη τη χρήση της εντολής GOTO (Πήγαινε).

Ο δομημένος προγραμματισμός προτάθηκε ως έννοια το 1966 από την ανάγκη να περιοριστεί η χρήση των εντολών GOTO (Πήγαινε).

Η **ιεραρχική σχεδίαση** ή ανάλυση «από πάνω προς τα κάτω» χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα. Αυτά είναι πιο εύκολο να επιλυθούν οδηγώντας τελικά στην επίλυση του αρχικού προβλήματος.

Ο τμηματικός προγραμματισμός υλοποιεί την ιεραρχική σχεδίαση όπου κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα που ονομάζεται υποπρόγραμμα. Τα υποπρογράμματα πρέπει να χαρακτηρίζονται από όσο το δυνατόν μεγαλύτερο βαθμό ανεξαρτησίας και από υψηλό βαθμό συνεκτικότητας, δηλαδή να υλοποιούν μια μόνο συγκεκριμένη διαδικασία ή λειτουργία. Το πρόγραμμα τελικά είναι η σύνθεση όλων των υποπρογραμμάτων.

Υποπρόγραμμα είναι το τμήμα προγράμματος που επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα.

Τέλος, όλες οι ενότητες του προγράμματος μπορούν να αναπτυχθούν μόνο με τρεις βασικές συνιστώσες: τη **δομή ακολουθίας**, τη **δομή επιλογής** και τη **δομή επανάληψης** ή συνδυασμό τους. Οι τρεις αυτές συνιστώσες ομαδοποιούν τις εντολές και καθορίζουν τη σειρά εκτέλεσης τους. Οι εντολές λοιπόν του προγράμματος οργανώνονται σε μία δομή.

Σε αντίθεση, η χρήση της εντολής GOTO έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, δηλαδή της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη. Έτσι αυξάνεται η δυσκολία στην αρχική σχεδίαση της λύσης, στην παρακολούθηση

και κατανόηση του προγράμματος και τέλος στη συντήρησή του.

Ο δομημένος προγραμματισμός αποτελεί λοιπόν μια μεθοδολογία σύνταξης προγραμμάτων που έχει σκοπό να βοηθήσει τον προγραμματιστή στην ανάπτυξη σύνθετων προγραμμάτων, να μειώσει τα λάθη, να διευκολύνει την κατανόηση των προγραμμάτων, τις διορθώσεις και τις αλλαγές σε αυτά.

Οι γλώσσες στις οποίες αναπτύσσονται προγράμματα με τη μεθοδολογία του δομημένου προγραμματισμού ακολουθούν κατά βάση το υπόδειγμα του προστακτικού προγραμματισμού.

Στον δομημένο προγραμματισμό γίνεται χρήση υποπρογραμμάτων όπως οι διαδικασίες και οι συναρτήσεις.

Τα υποπρογράμματα που αναπτύσσονται μπορούν να αποτελέσουν σύνθετες εντολές προς τον υπολογιστή.

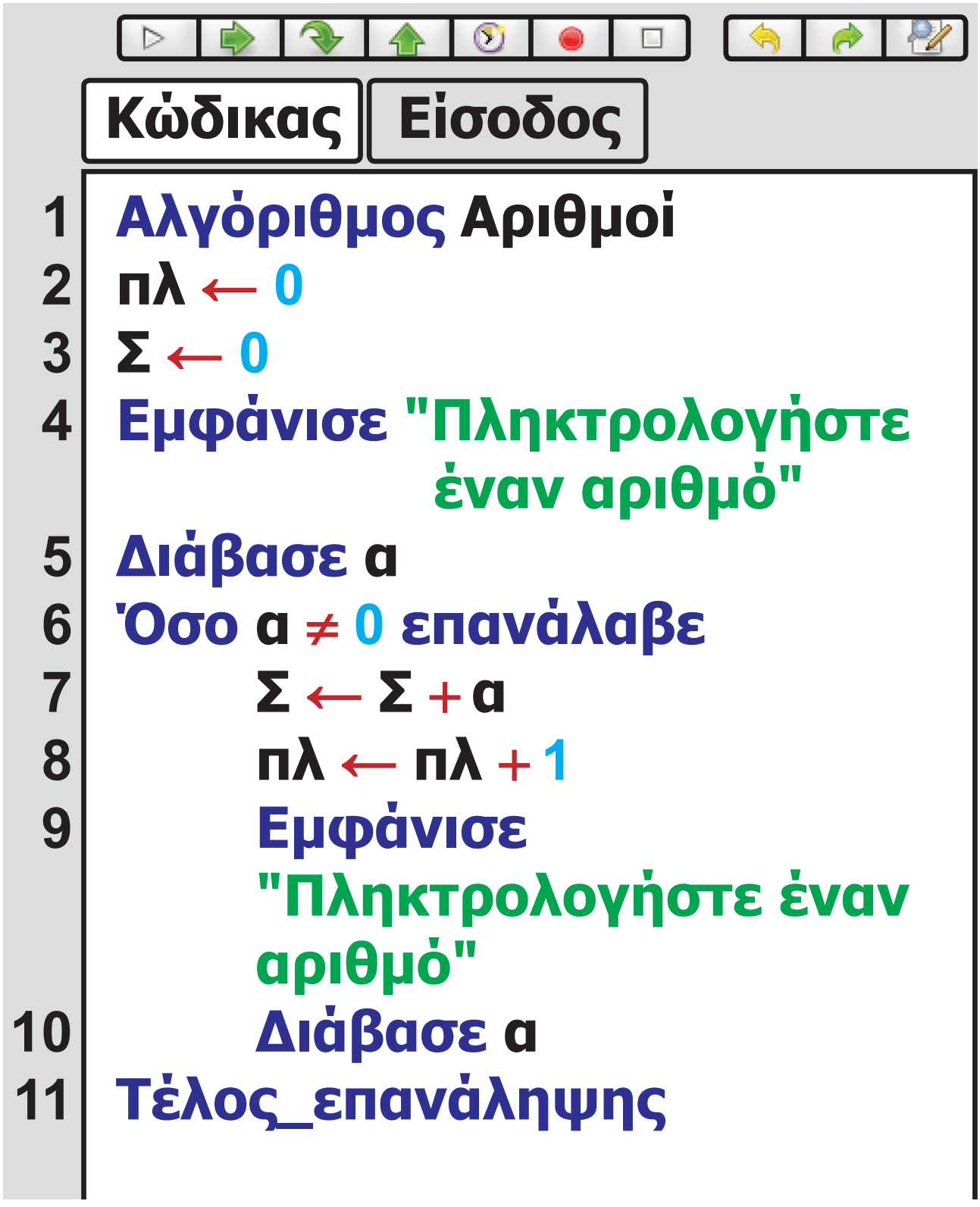
Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μόνο μία έξοδο. Δηλαδή, όταν ενεργοποιείται, δέχεται κατά βάση κάποια δεδομένα και, αφού ολοκληρώσει τη λειτουργία του, επιστρέφει κάποια αποτελέσματα. Κατά τη διάρκεια όμως της εκτέλεσής του δεν εκτελείται καμία άλλη εντολή του προγράμματος πέρα

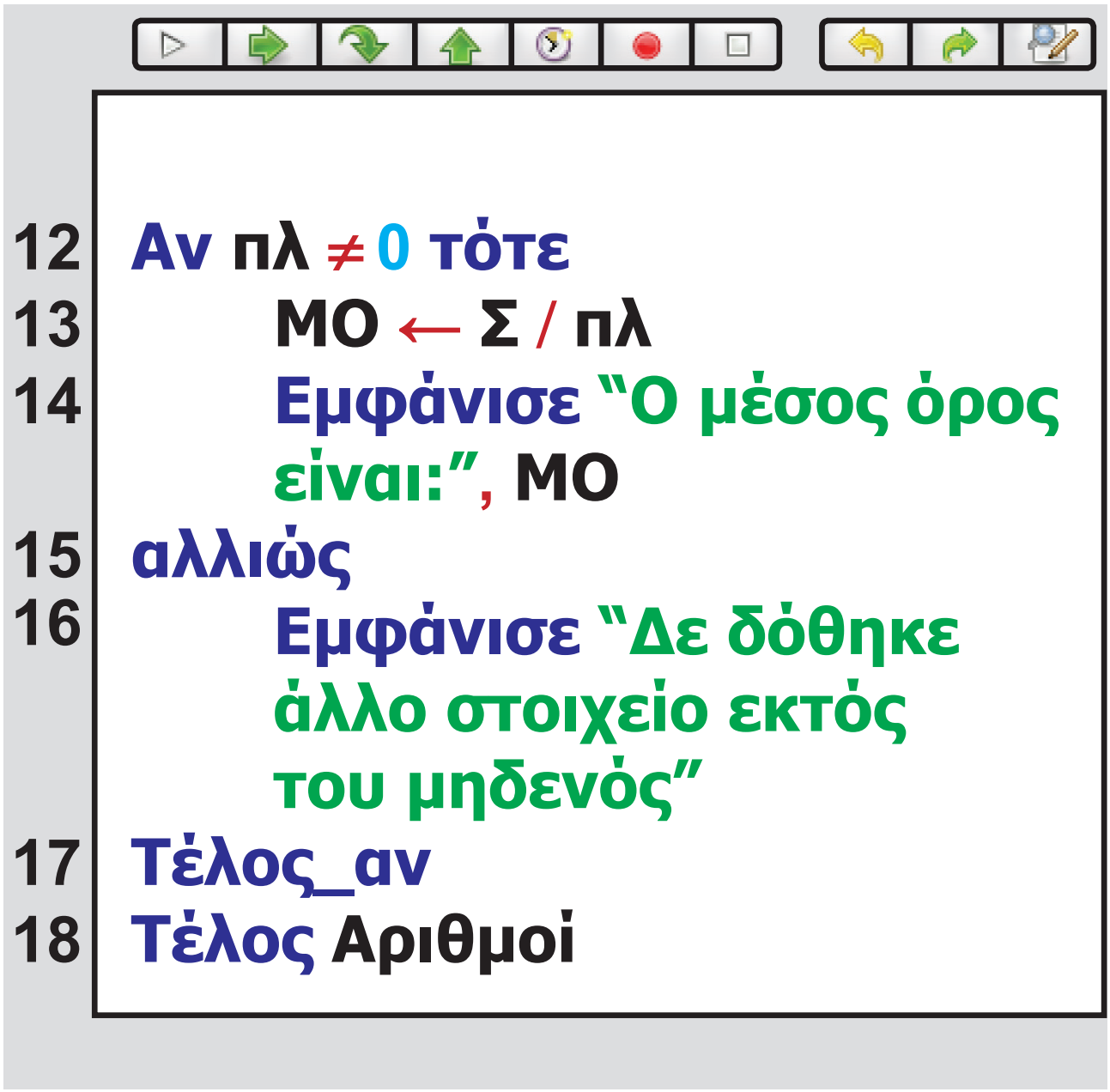
από αυτές που συμπεριλαμβάνονται στο συγκεκριμένο υποπρόγραμμα.

Παράδειγμα 2.32. Να γραφεί πρόγραμμα το οποίο θα διαβάζει αριθμούς και θα υπολογίζει το μέσο όρο τους. Η διαδικασία εισαγωγής αριθμών θα σταματά αν διαβαστεί ο αριθμός 0, χωρίς να συνυπολογίζεται. Αν ο πρώτος αριθμός που θα διαβαστεί είναι το μηδέν, θα εκτυπώνει το μήνυμα «Δεν δόθηκε άλλο στοιχείο εκτός του μηδενός». Το πρόγραμμα να υλοποιηθεί:

1. στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL).

2. στη γλώσσα προγραμματισμού PASCAL.





```
12  Αν πλ ≠ 0 τότε
13        ΜΟ ← Σ / πλ
14        Εμφάνισε "Ο μέσος όρος
          είναι:", ΜΟ
15  αλλιώς
16        Εμφάνισε "Δε δόθηκε
          άλλο στοιχείο εκτός
          του μηδενός"
17  Τέλος_αν
18  Τέλος Αριθμοί
```

Το προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) υποστηρίζει τις εντολές της ψευδογλώσσας

που ορίστηκε στο προηγούμενο κεφάλαιο και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

Στο πρόγραμμα αυτό θα εισαχθεί άγνωστο πλήθος αριθμών προς επεξεργασία. Αυτό θα επιτευχθεί με την εντολή **Όσο**. Επίσης είναι απαραίτητο να ελεγχθεί αν δόθηκε ένα τουλάχιστον στοιχείο διαφορετικό από το μηδέν ή όχι. Για το συγκεκριμένο έλεγχο θα χρησιμοποιηθεί η εντολή σύνθετης επιλογής.

Free Pascal IDE

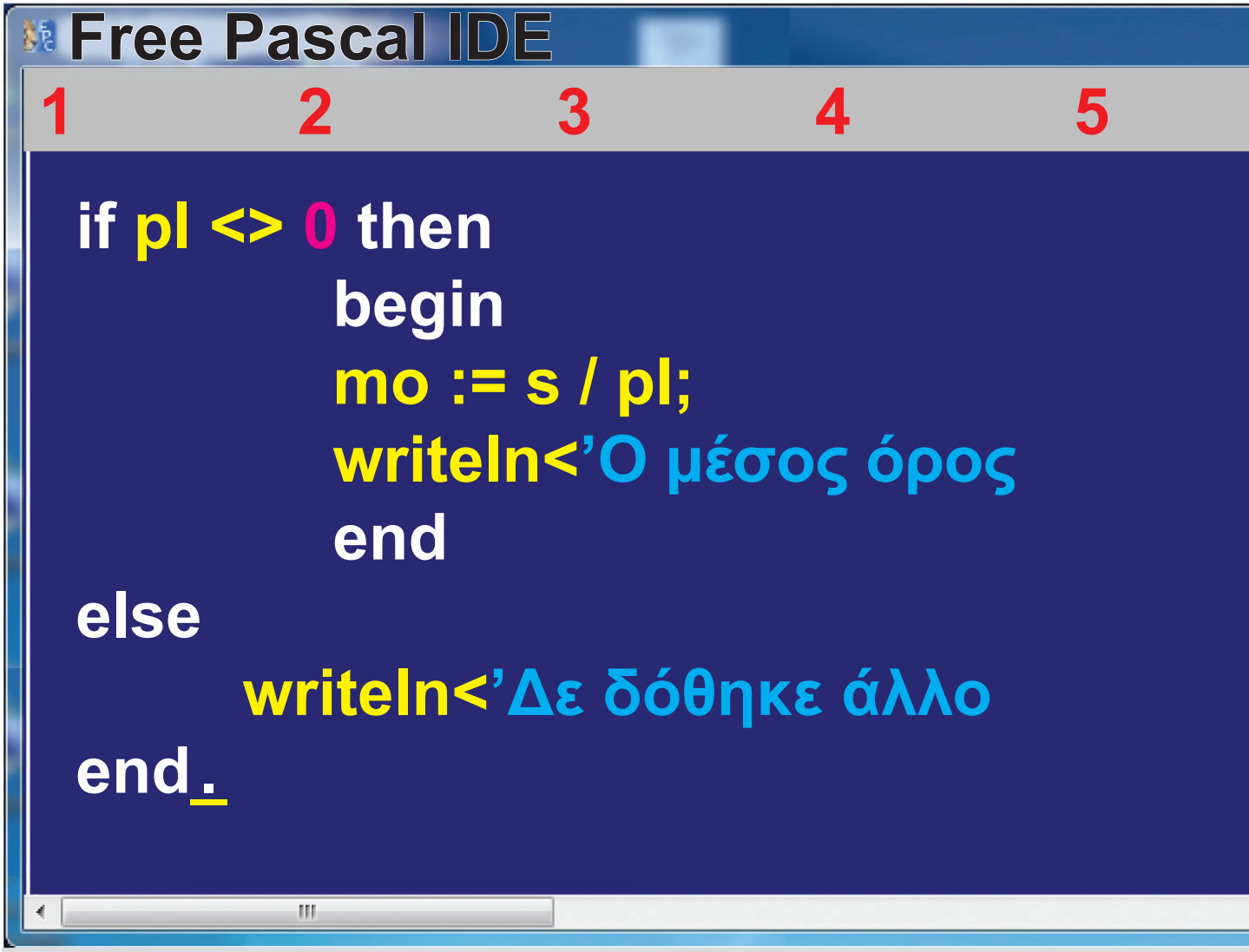
1 2 3 4 5

```
Program arithmoi;
VAR
pl: INTEGER;
a, s, mo: REAL;
begin
pl := 0; s := 0;
writeln<'Πληκτρολογήστε έναν
readln<a>;
while a<> 0 do
            begin
            s:= s + a;
            pl := pl + 1;
            writeln<'Πληκτρολογήστε
            readln<a>;
            end;
```

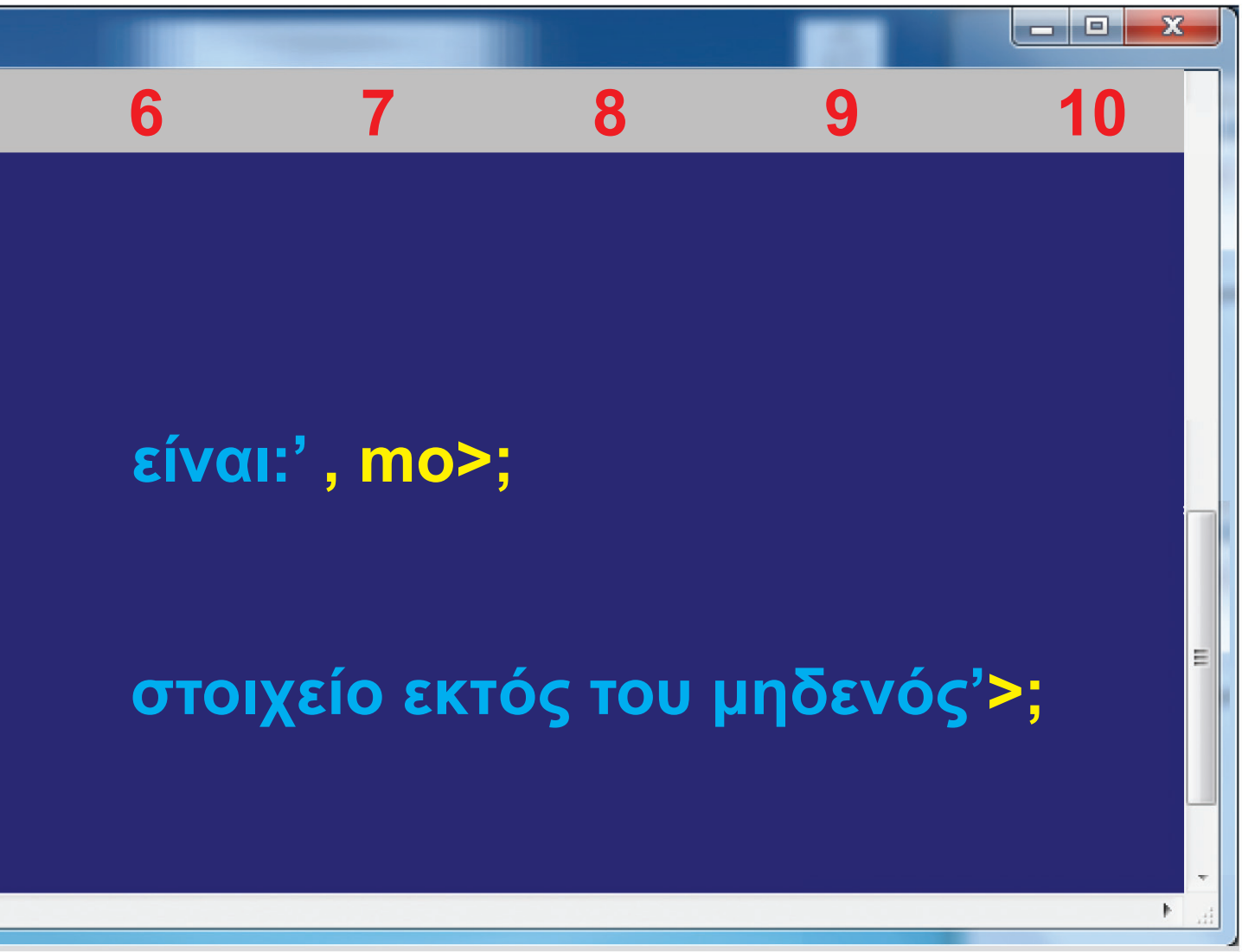
6 7 8 9 10

αριθμό’>;

έναν αριθμό’>;



1 File, 2 Edit, 3 Search, 4 Run,
7 Tools, 8 Options, 9 Window,



5 Compile, 6 Debug
10 Help

Το προγραμματιστικό περιβάλλον στο οποίο υλοποιήθηκε το πρόγραμμα στη γλώσσα PASCAL μπορεί να βρεθεί στη διεύθυνση **http://www.freepascal.org/.**

Η εντολή **Όσο** υλοποιείται με την εντολή **While** και η σύνθετη εντολή επιλογής με την εντολή **If...else**

Η σχεδίαση προγραμμάτων στη γλώσσα PASCAL ακολουθεί τον δομημένο προγραμματισμό.

2.3.2 Σχεδίαση και συγγραφή κώδικα

Οι περισσότερες γλώσσες προγραμματισμού υποστηρίζονται από προγραμματιστικά περιβάλλοντα τα οποία παρέχουν εργαλεία που διευκολύνουν την εργασία του προγραμματιστή. Για τη σύνταξη του **πηγαίου προγράμματος** χρησιμοποιείται ένα ειδικό πρόγραμμα το οποίο ονομάζεται **συντάκτης** (editor). Στη συνέχεια το πηγαίο πρόγραμμα πρέπει να μεταφραστεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή δηλαδή σε εντολές γλώσσας μηχανής. Το έργο της μετάφρασης το αναλαμβάνουν δύο προγράμματα ο **μεταγλωττιστής** ή ο **διερμηνευτής**.

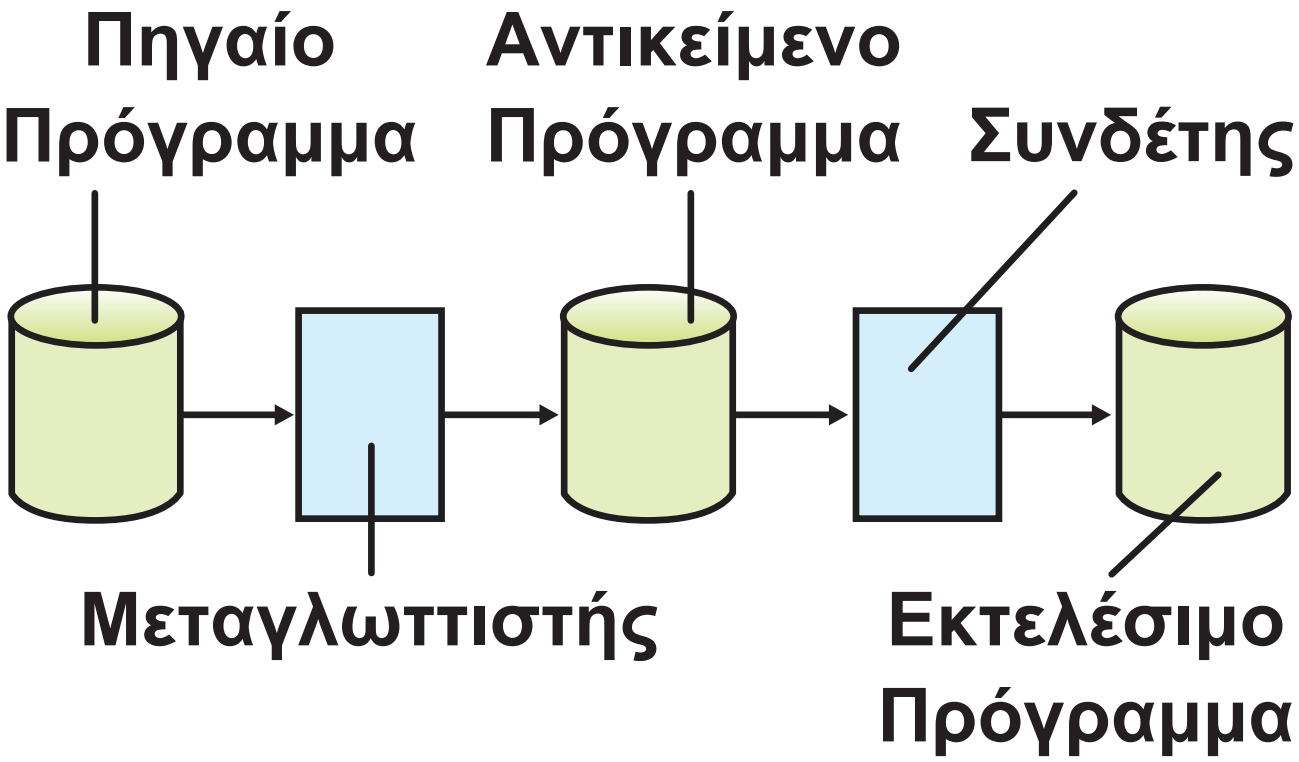
Το πρόγραμμα που γράφεται σε κάποια γλώσσα προγραμματισμού ονομάζεται **πηγαίο πρόγραμμα** (source program).

Οι **μεταγλωττιστές** (compilers) δέχονται στην είσοδο ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου και παράγουν ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το πρόγραμμα που παράγεται ονομάζεται **αντικείμενο** (object) πρόγραμμα. Το αντικείμενο πρόγραμμα δεν είναι σε θέση να εκτελεστεί. Χρειάζεται να συνδεθεί με άλλα τμήματα προγράμματος τα οποία 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 είναι η παραγωγή του **εκτελέσιμου** (executable) προγράμματος (Εικόνα 2.31). Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το πηγαίο πρόγραμμα.

Οι **διερμηνευτές** (interpreters) διαβάζουν μία προς μία τις εντολές του πηγαίου προγράμματος και για καθεμία εκτελούν αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.

Για να μεταφραστεί το πηγαίο πρόγραμμα σε εντολές γλώσσας μηχανής δεν θα πρέπει να ανιχνευθούν λάθη. Τα λάθη που εμφανίζονται κατά τη μετάφραση ονομάζονται **συντακτικά**. Τα συντακτικά λάθη μπορεί να οφείλονται σε αναγραμματισμούς, σε λανθασμένη σύνταξη εντολών, παράλειψη δήλωσης μεταβλητών κ.ά.. Ο μεταφραστής ανιχνεύει τα λάθη και εμφανίζει κατάλληλα διαγνωστικά μηνύματα. Στη συνέχεια ακολουθεί η διόρθωσή τους από τον προγραμματιστή.



Εικόνα 2.31. Μεταγλώττιση και Σύνδεση Προγράμματος.

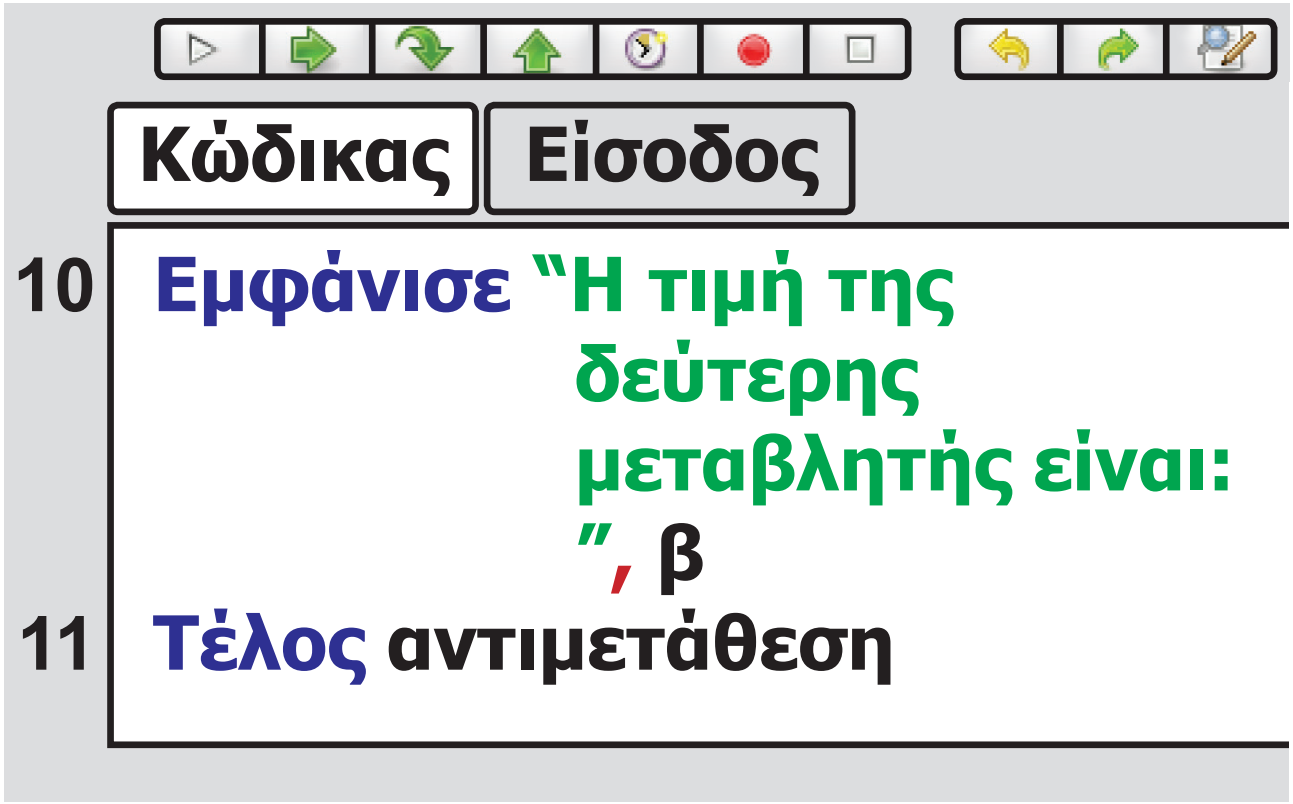
Πέρα όμως από τα συντακτικά λάθη υπάρχουν και τα **λογικά** που δεν είναι δυνατό να ανιχνευθούν από τα μεταφραστικά προγράμματα. Τα περισσότερα όμως προγραμματιστικά περιβάλλοντα παρέχουν εργαλεία εκσφαλμάτωσης που βοηθούν τον

προγραμματιστή να εκτελέσει το πρόγραμμα εντολή προς εντολή μέχρι συγκεκριμένο σημείο ή να παρακολουθεί τις τιμές των μεταβλητών έτσι ώστε να εντοπίσει τα λάθη στην υλοποίηση του αλγορίθμου.

Παράδειγμα 2.33. Με σκοπό να αναπτυχθεί πρόγραμμα το οποίο θα αντιμεταθέτει το περιεχόμενο δύο μεταβλητών γράφτηκαν οι ακόλουθες εντολές στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL).

Κώδικας | **Είσοδος**

```
1  Αλγόριθμος αντιμετάθεση
2  Εμφάνισε "Πληκτρολογήστε
            την τιμή της
            πρώτης
            μεταβλητής"
3  Διάβασε α
4  Εμφάνισε "Πληκτρολογήστε
            την τιμή της
            δεύτερης
            μεταβλητής"
5  Διάβασε β
6  temp ← α
7  α ← β
8  β ← temp
9  Εμφάνισε "Η τιμή της πρώτης
            μεταβλητής είναι:
            , α
```



Στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας, το χρώμα των γραμμάτων διαφοροποιείται σε μπλε για τις δεσμευμένες λέξεις, σε μαύρο για τις μεταβλητές και πράσινο για τις αλφαριθμητικές σταθερές ώστε να διακρίνονται εύκολα τα διάφορα στοιχεία του προγράμματος.

Τι θα συμβεί κατά τη μετάφραση του προγράμματος;

Απάντηση

Υπάρχουν δύο συντακτικά λάθη στις εντολές που έχουν γραφεί. Το πρώτο βρίσκεται στην πέμπτη γραμμή του προγράμματος και οφείλεται σε αναγραμματισμό της εντολής Διάβασε και το δεύτερο στην ένατη γραμμή και οφείλεται στην παράλειψη των διπλών εισαγωγικών στο τέλος της αλφαριθμητικής έκφρασης. Το προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) χρησιμοποιεί διερμηνευτή για τη μετάφραση του πηγαίου προγράμματος. Έτσι αρχικά θα εντοπιστεί το πρώτο

συντακτικό λάθος με ένα μήνυμα της μορφής:

Συντακτικό Λάθος - Περίμενα την εντολή εκχώρησης
Βρήκα: β

Ο διερμηνευτής αναγνωρίζει τη λέξη Διάβαεσ ως το όνομα κάποιας μεταβλητής και περιμένει στη συνέχεια να δει το αριστερό βέλος της εντολής εκχώρησης, κάτι το οποίο όμως δεν συμβαίνει. Επίσης εντοπίζει μόνο το πρώτο συντακτικό λάθος. Εφόσον αυτό διορθωθεί, ο διερμηνευτής βρίσκει το επόμενο λάθος και εμφανίζει ένα μήνυμα της μορφής:

Συντακτικό Λάθος - Περίμενα το χαρακτήρα. ”

Το ερώτημα αν οι μηχανές μπορούν να σκέφτονται έχει την ίδια σημασία με το ερώτημα αν τα υποβρύχια μπορούν να κολυμπούν.

Έντσγκερ Ντάικστρα (Edsger Dijkstra), Επιστήμονας Πληροφορικής

Παράδειγμα 2.34. **Με σκοπό να αναπτυχθεί πρόγραμμα το οποίο θα αντιμεταθέτει το περιεχόμενο δύο μεταβλητών γράφτηκαν οι ακόλουθες εντολές στο προγραμματιστικό περιβάλλον FreePascal.**

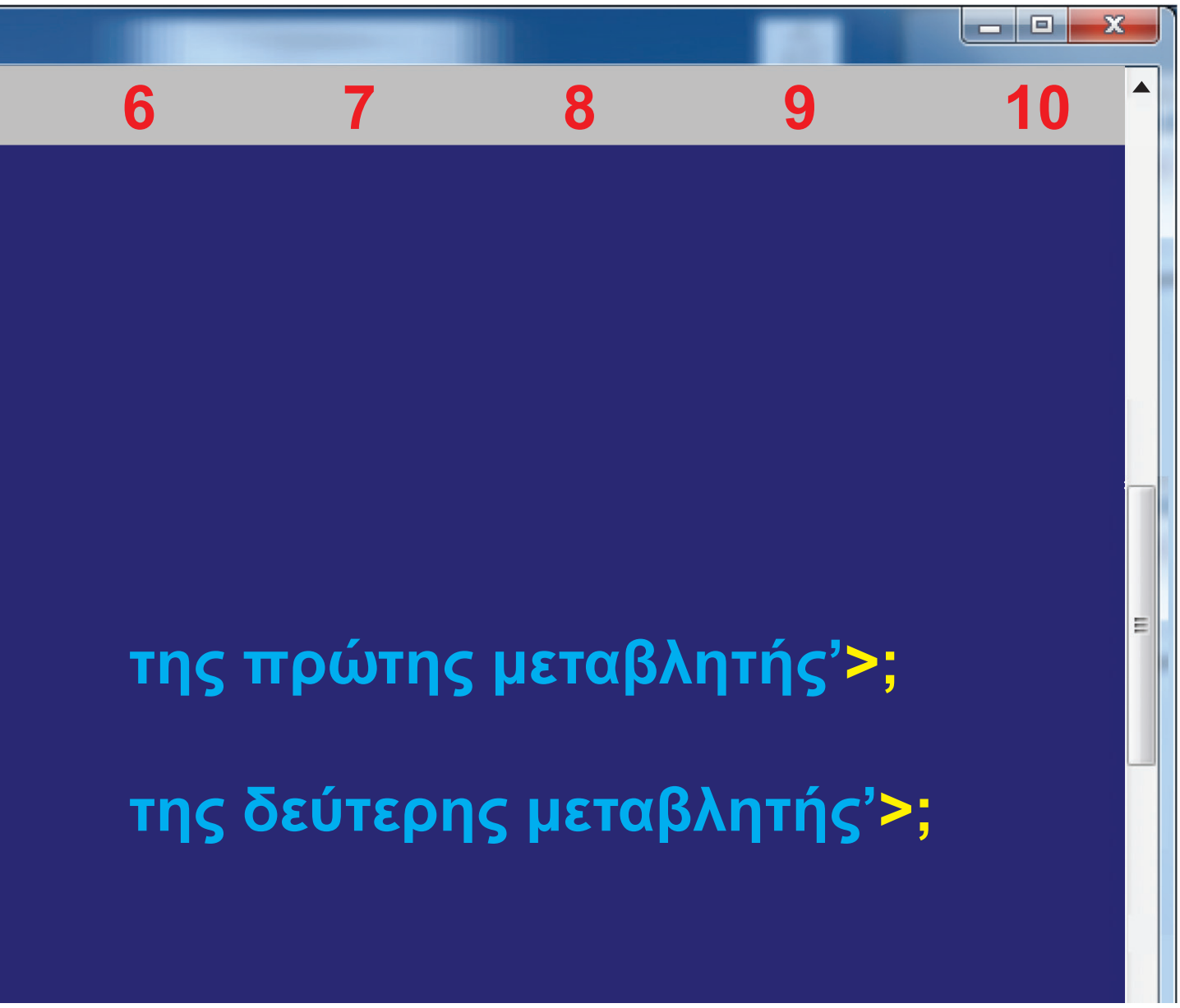
Η γλώσσα PASCAL είναι μία από τις γλώσσες για τη συγγραφή κώδικα στον Πανελλήνιο Διαγωνισμό Πληροφορικής.
http://www.pdp.gr

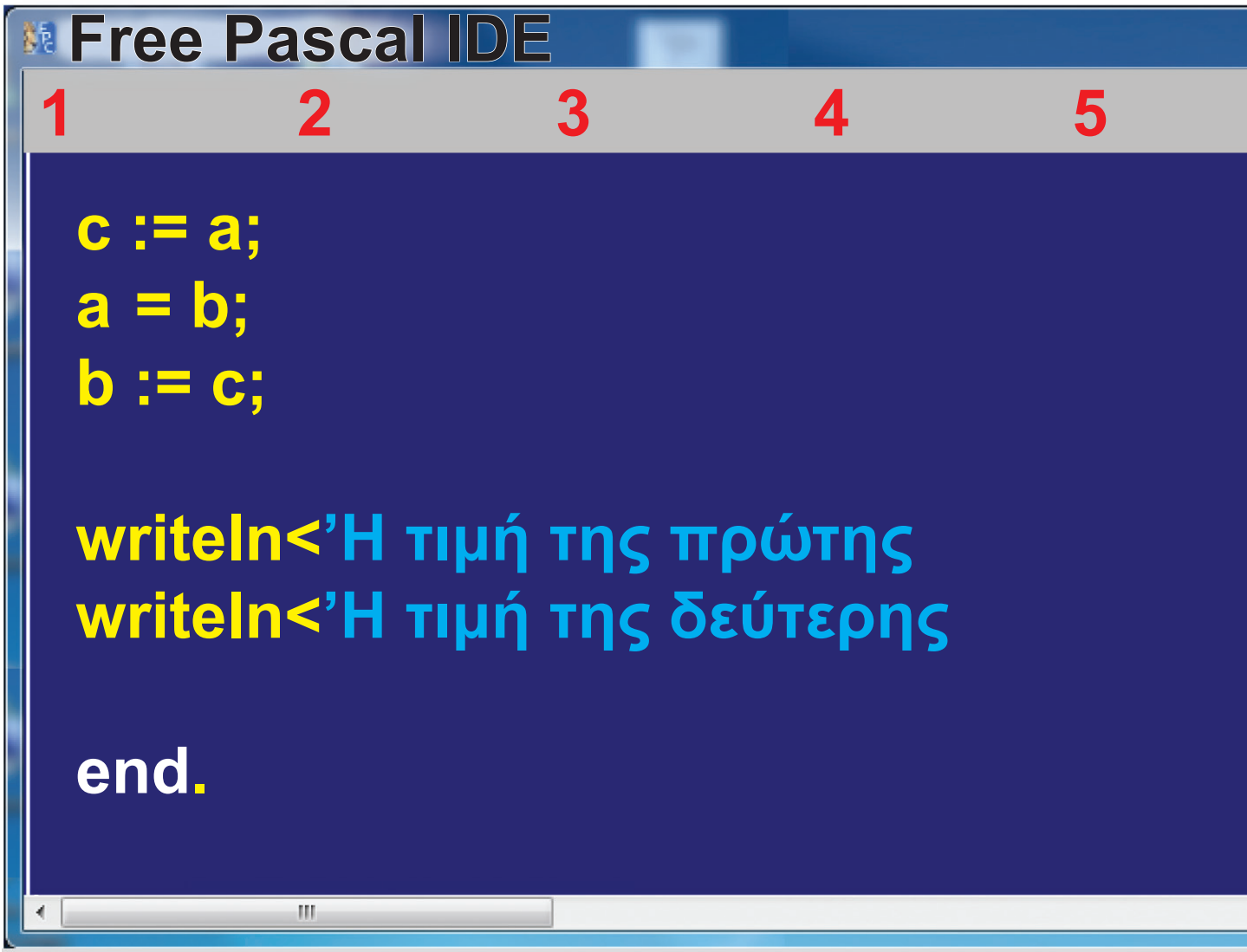
Free Pascal IDE

1 2 3 4 5

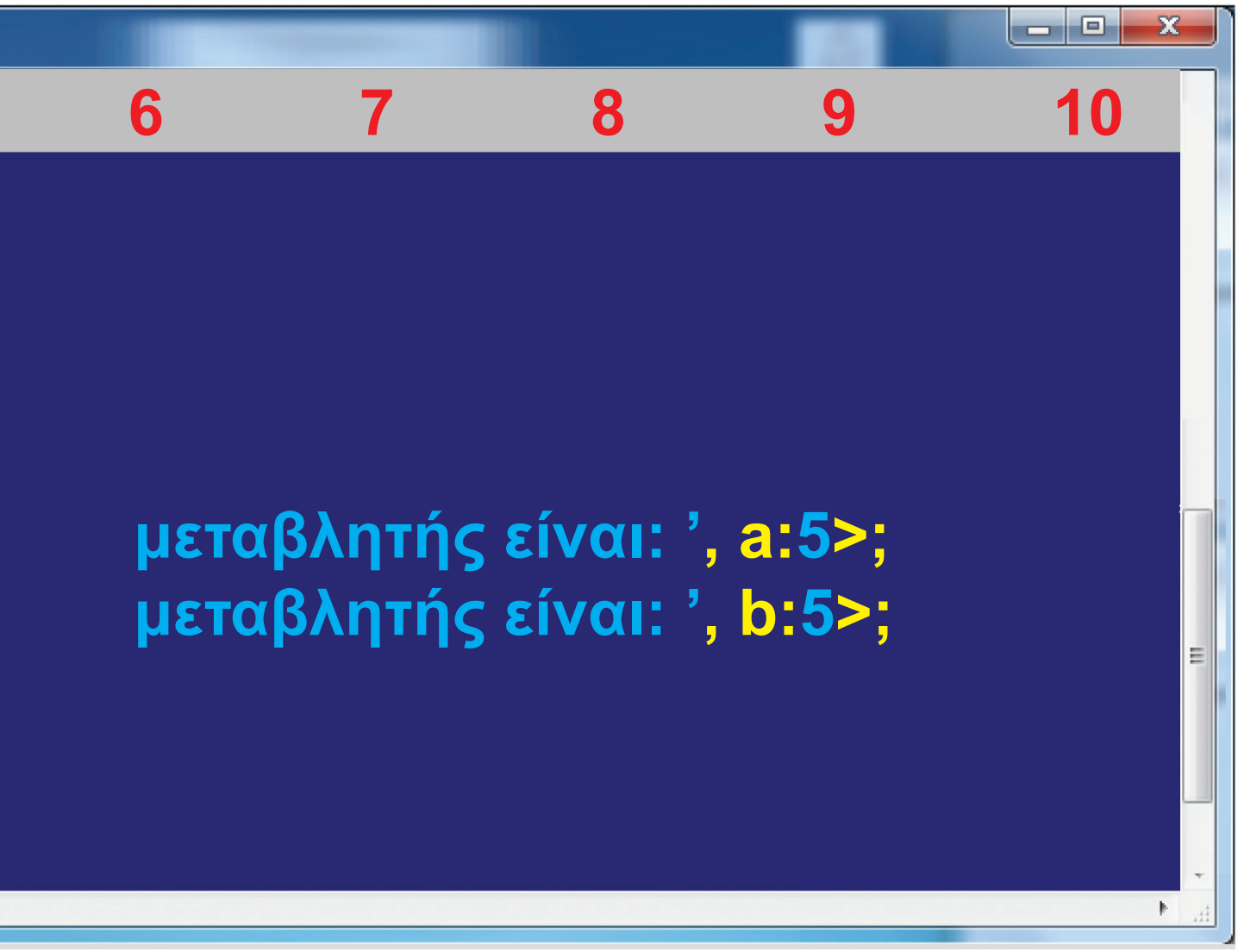
```
Program antimetathesi;
VAR
a, b: INTEGER;

begin

writeln<'Πληκτρολογήστε την τιμή
readln<a>;
writeln<'Πληκτρολογήστε την τιμή
readln<b>;
```

Στη γλώσσα προγραμματισμού PASCAL απαιτείται η δήλωση του τύπου των μεταβλητών που χρησιμοποιούνται από το πρόγραμμα.





1 File, 2 Edit, 3 Search, 4 Run,
7 Tools, 8 Options, 9 Window,



5 Compile, **6** Debug
10 Help

Τι θα συμβεί κατά τη μετάφραση του προγράμματος;

Απάντηση

Το μεταφραστικό πρόγραμμα στο συγκεκριμένο προγραμματιστικό περιβάλλον είναι μεταγλωττιστής. Έτσι θα εμφανιστεί το εξής μήνυμα.

```
antimetathesi.pas<12,3> Error:
antimetathesi.pas <13,6> Error:
antimetathesi.pas<14,7> Error:
antimetathesi.pas<20> Fatal:

antimetathesi.pas<0> Fatal:
```

Εντοπίζονται όλα τα συντακτικά λάθη που υπάρχουν. Τα δύο οφείλονται σε παράλειψη δήλωσης της μεταβλητής c και το άλλο οφείλεται σε λανθασμένη σύνταξη της εντολής εκχώρησης της τιμής της μεταβλητής b στη μεταβλητή a.

Identifier not found “c”

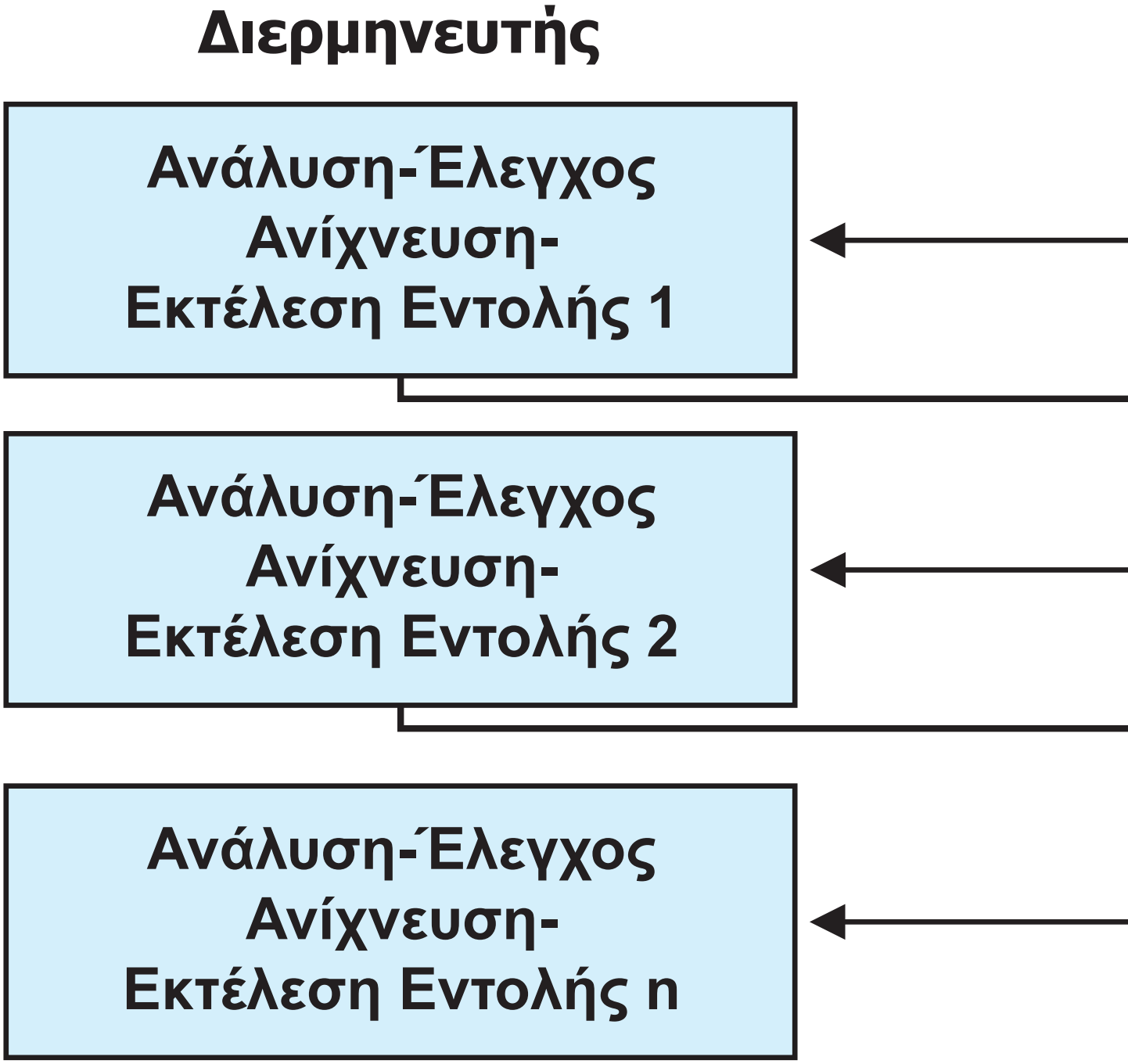
Illegal expression

Identifier not found “c”

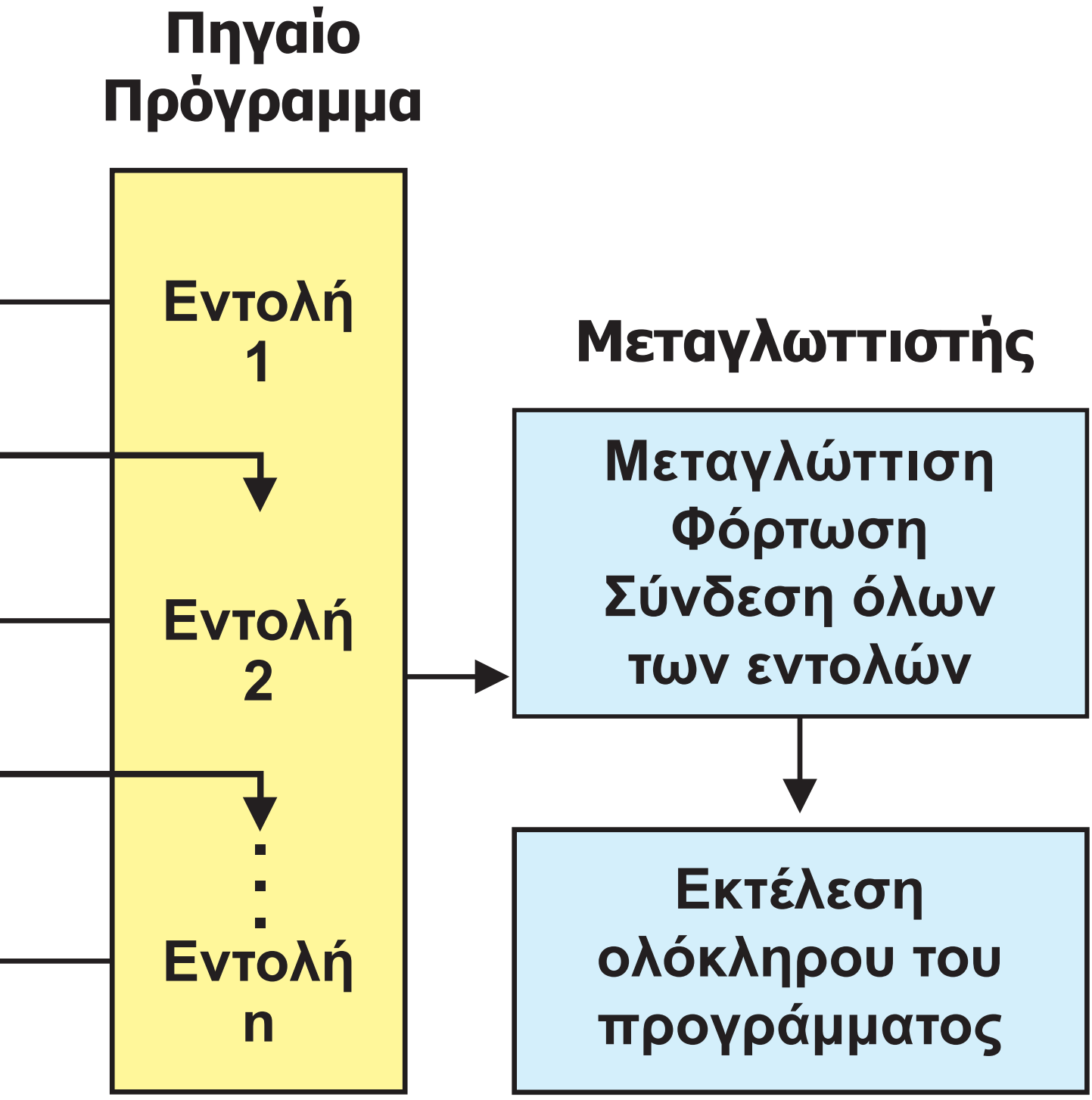
There were 3 errors compiling module,

Compilation aborted

Η διαδικασία μετάφρασης λοιπόν ενός προγράμματος σε προγραμματιστικά περιβάλλοντα που διαθέτουν είτε διερμηνευτή είτε μεταγλωττιστή φαίνεται στην εικόνα 2.32.



Εικόνα 2.32. Διαδικασία Μετάφρασης

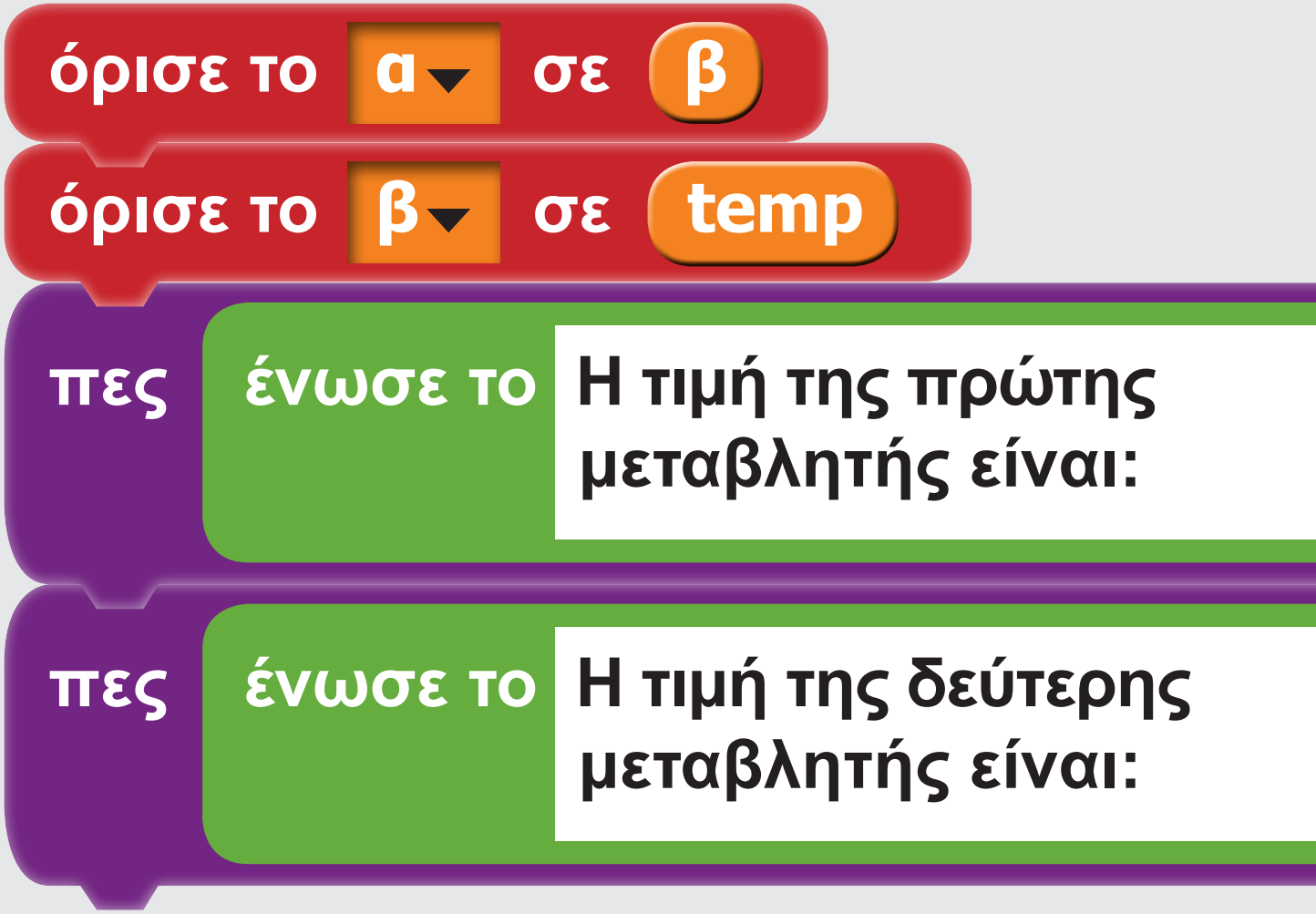


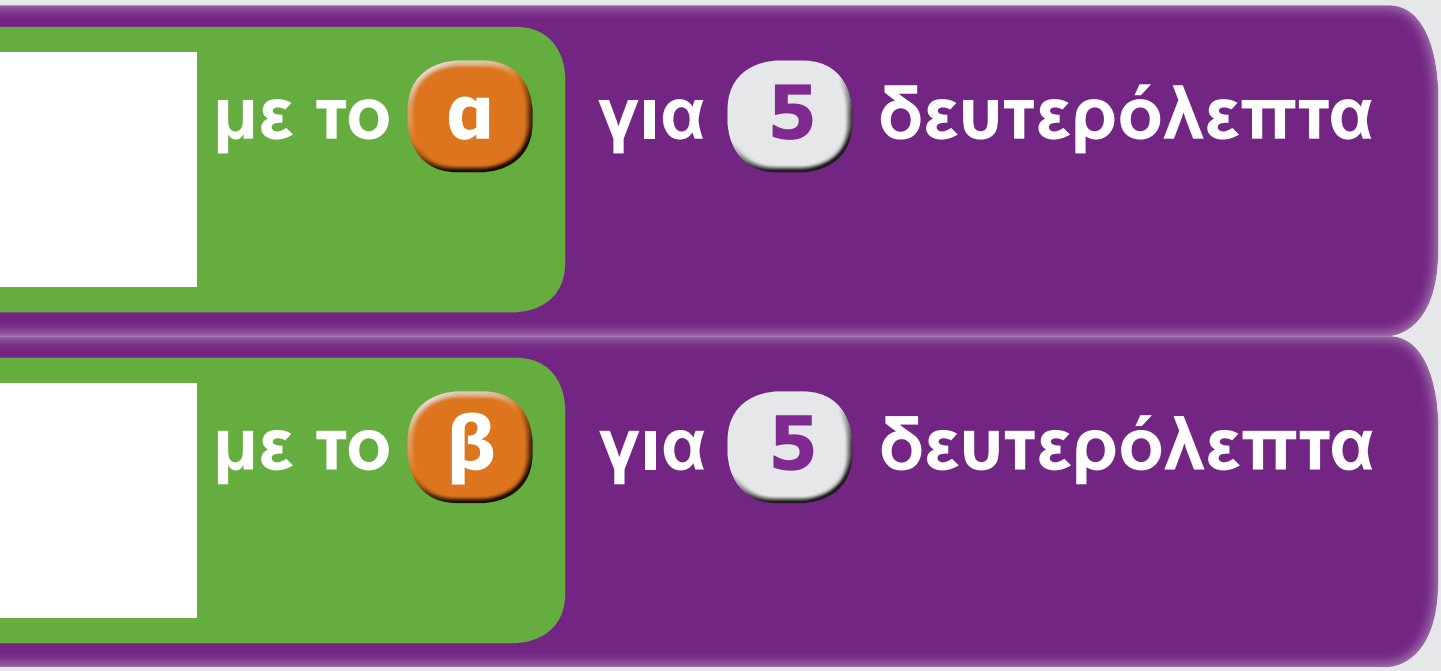
και Εκτέλεσης ενός προγράμματος.

Παράδειγμα 2.35. Να αναπτυχθεί πρόγραμμα το οποίο θα αντιμεταθέτει το περιεχόμενο δύο μεταβλητών στο προγραμματιστικό περιβάλλον SCRATCH.

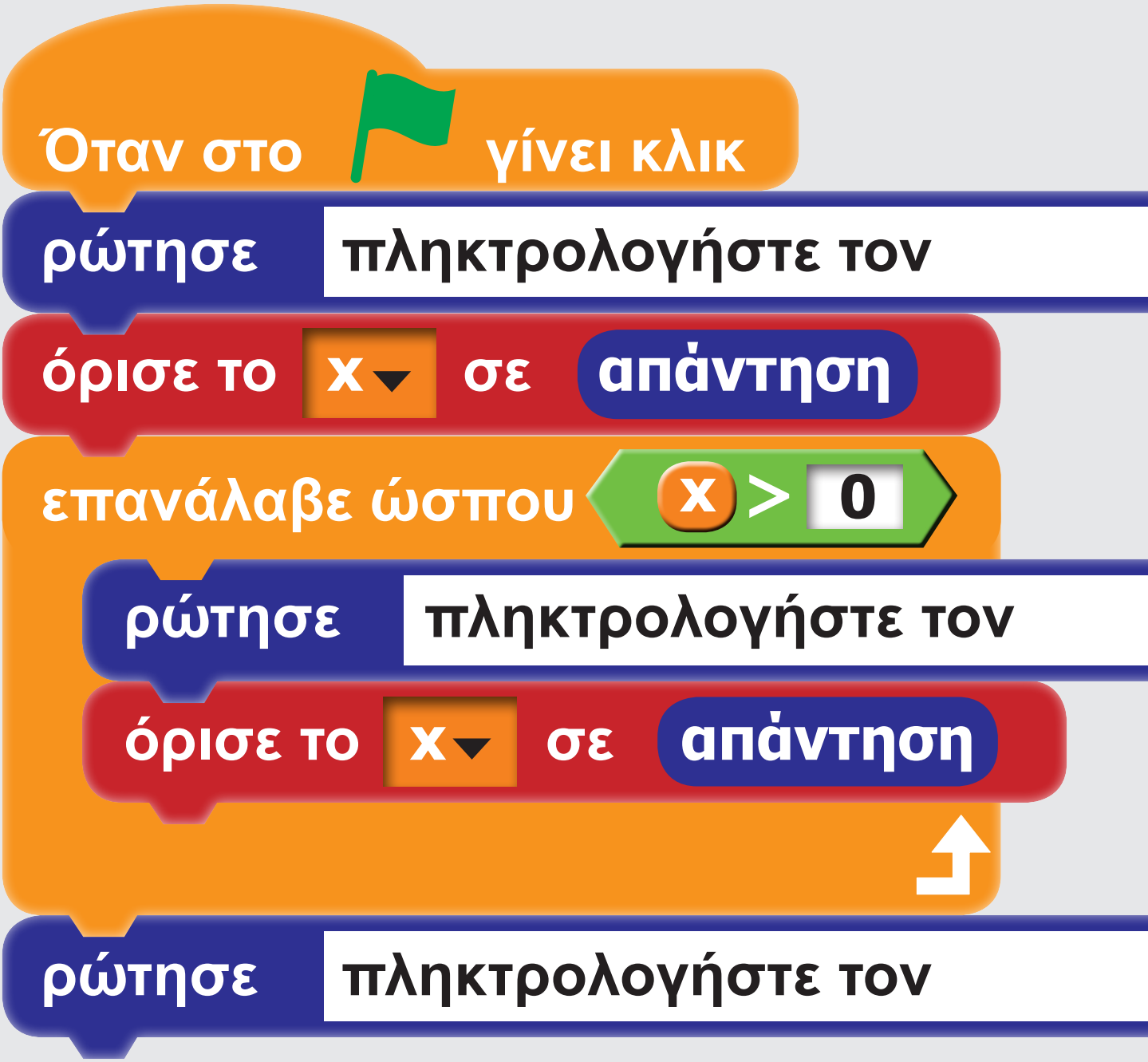
και περίμενε

και περίμενε


όρισε το α σε β
όρισε το β σε temp
πες ένωσε το Η τιμή της πρώτης μεταβλητής είναι:
πες ένωσε το Η τιμή της δεύτερης μεταβλητής είναι:


με το α
για 5 δευτερόλεπτα
με το β
για 5 δευτερόλεπτα

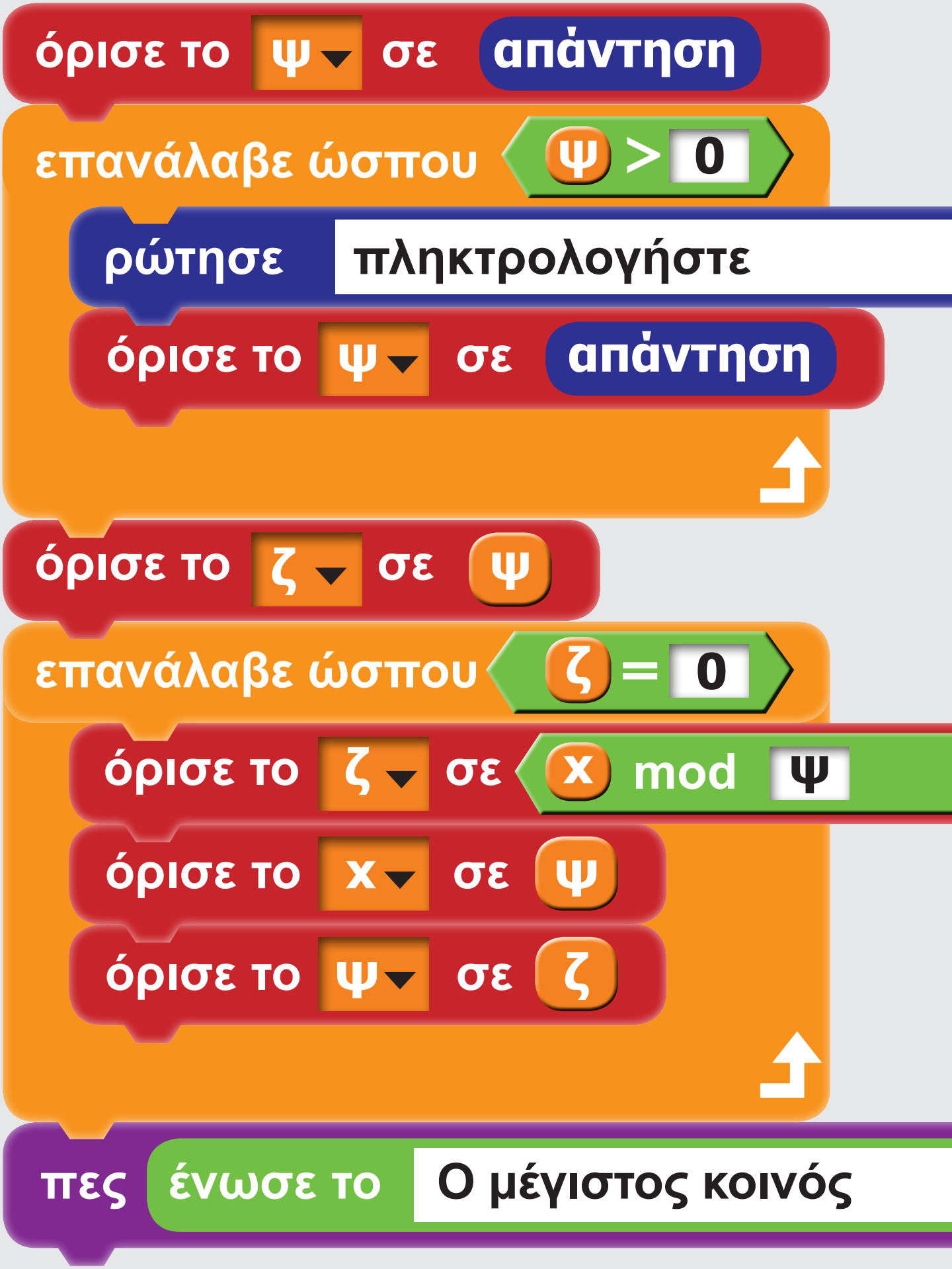
Παράδειγμα 2.36. Εύρεση μέγιστου κοινού διαιρέτη δύο θετικών ακέραιων αριθμών με τον επαναληπτικό αλγόριθμο του Ευκλείδη σε γλώσσα SCRATCH.



πρώτο αριθμό και περίμενε

πρώτο αριθμό και περίμενε

δεύτερο αριθμό και περίμενε


όρισε το ψ σε απάντηση
επανάλαβε ώσπου ψ > 0
ρώτησε πληκτρολογήστε
όρισε το ψ σε απάντηση
όρισε το ζ σε ψ
επανάλαβε ώσπου ζ = 0
όρισε το ζ σε x mod ψ
όρισε το x σε ψ
όρισε το ψ σε ζ
πες ένωσε το Ο μέγιστος κοινός

τον δεύτερο αριθμό και περίμενε

διαιρέτης είναι: με το X

Η γλώσσα οπτικού προγραμματισμού SCRATCH έχει δημιουργηθεί στο MIT.
http://scratch.mit.edu

Στη γλώσσα SCRATCH ο προγραμματισμός γίνεται με χρήση εντολών που μοιάζουν με κομμάτια ενός παζλ (Πλακίδια – blocks). Κάθε εντολή έχει το δικό της χαρακτηριστικό σχήμα και μπορεί να συνδυαστεί με άλλες μόνο με συγκεκριμένο τρόπο που αποκλείει τα συντακτικά λάθη.

Οι εντολές όταν συνδυάζονται δημιουργούν σενάρια ενεργειών που πρέπει να εκτελεστούν από αντικείμενα που λέγονται μορφές (sprites). Η πιο χαρακτηριστική μορφή είναι η γάτα.

Εικόνα 2.33. Η μορφή της γάτας.

Η εντολή Όσο της ψευδογλώσσας και η εντολή While της PASCAL δεν υπάρχει στη γλώσσα SCRATCH. Υπάρχει όμως η εντολή επανάλαβε ώσπου στην οποία οι εντολές εκτελούνται όσο η συνθήκη είναι ψευδής.

Το πρόγραμμα αρχικά εξασφαλίζει ότι οι τιμές που πληκτρολογούνται είναι θετικές.

Παράδειγμα 2.37. Να γραφεί πρόγραμμα για την εύρεση του μέγιστου κοινού διαιρέτη δύο θετικών ακεραίων αριθμών με τον αναδρομικό αλγόριθμο του Ευκλείδη στη γλώσσα LOGO. Ο κώδικας να αναπτυχθεί ως διαδικασία στο περιβάλλον MicroWorlds Pro.

```
για μκδ :χ :ψ
ΑνΔιαφορετικά :ψ = 0
[
   κάνε "ζ :χ
   ανακοίνωση (φρ[ο μέγιστος
   κοινός διαιρέτης είναι] :ζ)
]
[
   κάνε "χ υπόλοιπο :χ :ψ
   μκδ :ψ :χ
```

```
]
τέλος
```

Αρκετές φορές οι αλγόριθμοι χρειάζεται να τροποποιηθούν ώστε να μπορούν να εκτελεστούν από κάποια γλώσσα προγραμματισμού.

Για να εκτελεστεί το διπλανό πρόγραμμα και να ανακοινώσει το μέγιστο κοινό διαιρέτη των αριθμών 27 και 6, αρκεί να πληκτρολογήσει ο χρήστης στο κέντρο εντολών του περιβάλλοντος την εντολή:

μκδ 27 6

Οι μεταβλητές χ και ψ ονομάζονται παράμετροι.

Η σύνθετη επιλογή υλοποιείται με την εντολή

ΑνΔιαφορετικά.

Πολλές φορές κάποια προγράμματα μπορούν να χρησιμοποιήσουν κώδικα που έχει γραφτεί προηγουμένως. Η επαναχρησιμοποίηση κώδικα είναι αρκετά συνηθισμένη πρακτική η οποία περιορίζει τα λάθη και μειώνει το χρόνο που απαιτείται για τη συγγραφή του προγράμματος. Προγραμματιστικά γίνεται συνήθως με τη δημιουργία κατάλληλων υποπρογραμμάτων, δηλαδή διαδικασιών ή συναρτήσεων ανάλογα με τη

γλώσσα προγραμματισμού. Αρκετές φορές ο προγραμματιστής μπορεί να γράψει ή να χρησιμοποιήσει **βιβλιοθήκες** (libraries). Οι βιβλιοθήκες μιας γλώσσας προγραμματισμού είναι μία συλλογή από έτοιμα υποπρογράμματα που μπορούν να χρησιμοποιούνται κατά τη συγγραφή νέων προγραμμάτων. Μία βιβλιοθήκη μπορεί να χρησιμοποιηθεί τόσο από τον δημιουργό της όσο και από άλλους προγραμματιστές. Αποτελεί επίσης πρακτική πολλών προγραμματιστών να βελτιώνουν υπάρχουσες βιβλιοθήκες, ώστε να επεκτείνουν τις δυνατότητες των προγραμμάτων τους.

Παράδειγμα 2.38. Να γραφεί πρόγραμμα που θα δέχεται δύο θετικούς ακέραιους αριθμούς και θα εμφανίζει το μέγιστο κοινό διαιρέτη και το ελάχιστο κοινό πολλαπλάσιο τους. Η υλοποίηση να γίνει στο περιβάλλον MicroWorlds Pro.

```
για μκδ :χ :ψ
ΑνΔιαφορετικά :ψ = 0
[
   κάνε "ζ :χ
]
[
   κάνε "χ υπόλοιπο :χ :ψ
   μκδ :ψ :χ
]
τέλος
για εκπο :χ :ψ
μκδ :χ :ψ
```

```
κάνε “εκ :χ * :ψ / :ζ
τέλος

για υπολογισμός :χ :ψ
εκπο :χ :ψ

ανακοίνωση (φρ[ο μέγιστος
κοινός διαιρέτης είναι] :ζ)

ανακοίνωση (φρ[το ελάχιστο κοινό
πολλαπλάσιο είναι] :εκ)

τέλος
```

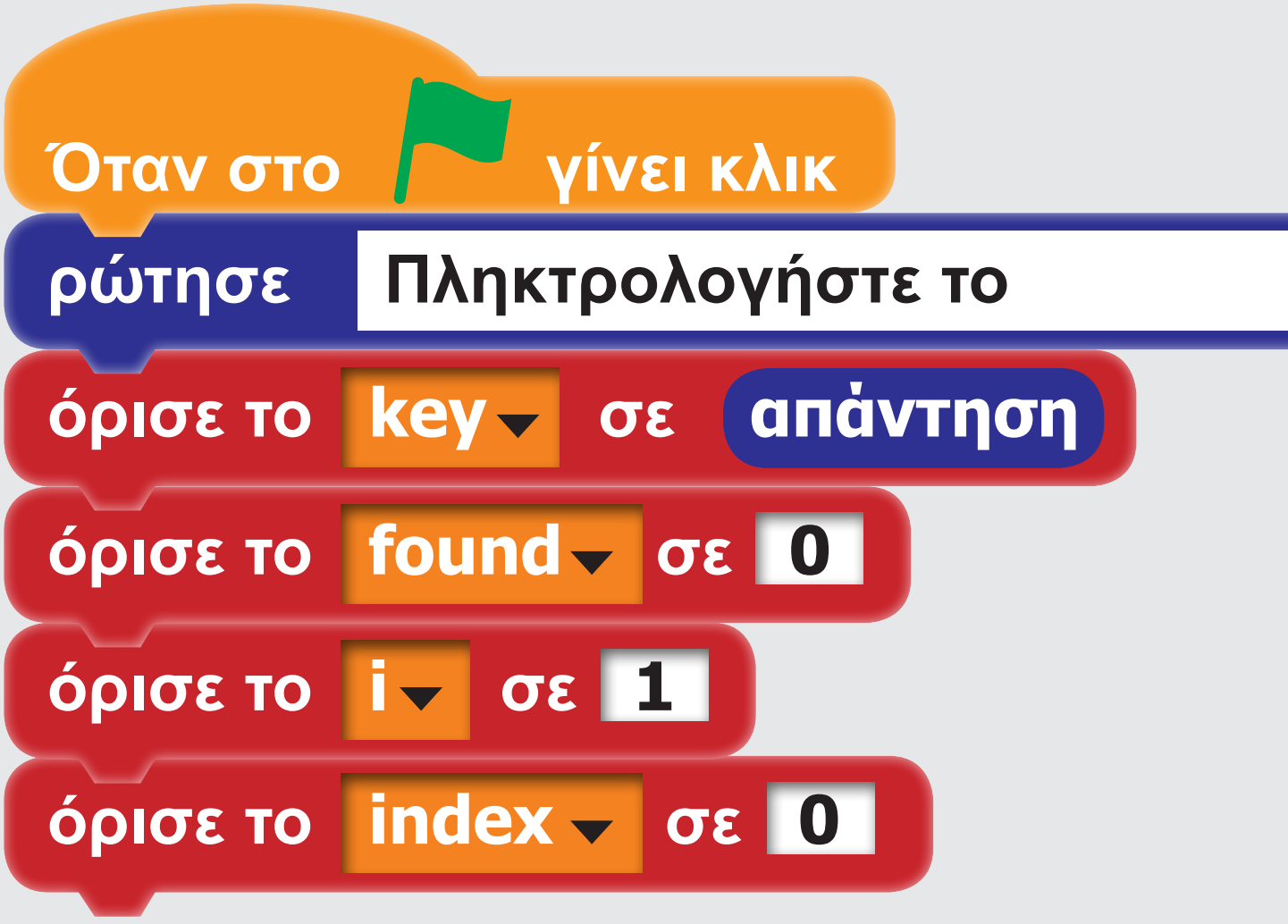
Αν χ, ψ είναι δύο θετικοί ακέραιοι τότε ΜΚΔ(χ,ψ)*ΕΚΠ(χ,ψ) = = χ*ψ

Το πρόγραμμα στη γλώσσα LOGO περιλαμβάνει τρεις διαδικασίες: μία για τον υπολογισμό του μέγιστου κοινού διαιρέτη η οποία επαναχρησιμοποιείται, μία για τον υπολογισμό του ελάχιστου κοινού πολλαπλάσιου που καλεί τη διαδικασία του μέγιστου κοινού διαιρέτη και μία που καλεί τη διαδικασία υπολογισμού του ελάχιστου κοινού πολλαπλάσιου και εμφανίζει τα αποτελέσματα.

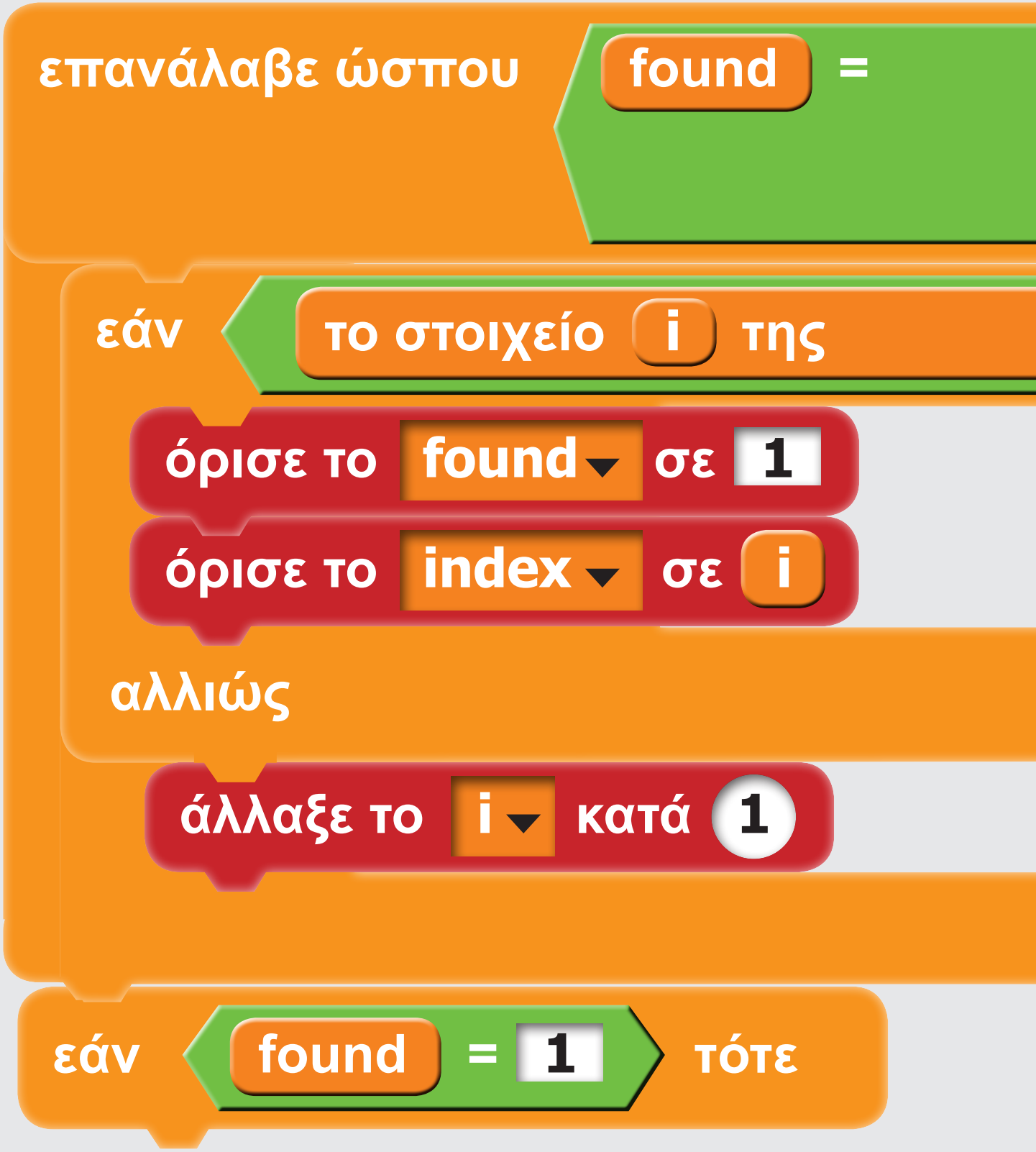
Οι τιμές των μεταβλητών στη LOGO είναι διαθέσιμες σε όλες τις διαδικασίες. Οι μεταβλητές αυτές ονομάζονται καθολικές. Σε άλλες γλώσσες

προγραμματισμού υπάρχουν και οι **τοπικές** μεταβλητές, οι οποίες μπορούν να προσπελαστούν μόνο στο πρόγραμμα ή το υποπρόγραμμα το οποίο δηλώνονται.

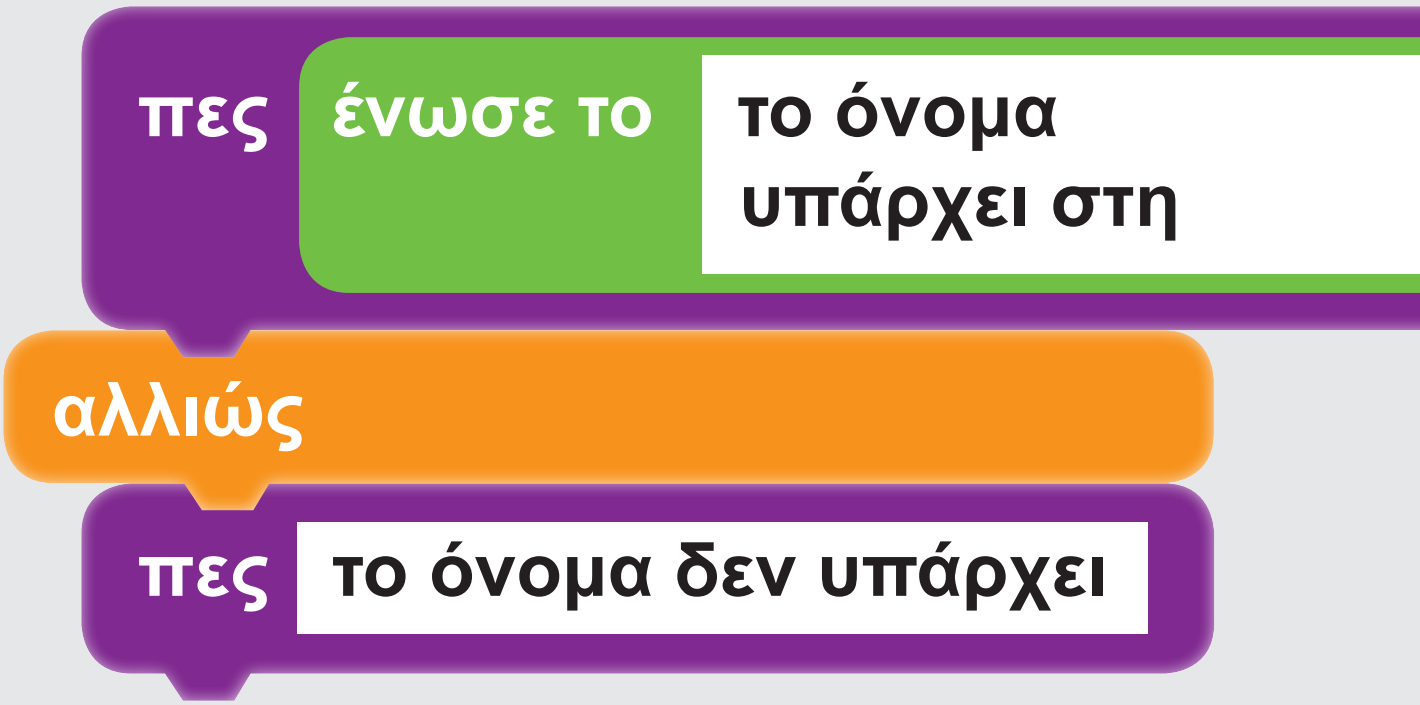
Παράδειγμα 2.39. Να γραφεί πρόγραμμα σε γλώσσα SCRATCH το οποίο θα αναζητά αν υπάρχει ένα όνομα σε μία λίστα και θα εμφανίζει κατάλληλο μήνυμα σχετικά με την εύρεσή του.



όνομα που αναζητάτε και περίμενε


επανάλαβε ώσπου found =
εάν το στοιχείο i της
όρισε το found σε 1
όρισε το index σε i
αλλιώς
άλλαξε το i κατά 1
εάν found = 1 τότε


1
ή
i
>
το μήκος της
λίστας names
λίστας
names
=
key
τότε


πες
ένωσε το
το όνομα
υπάρχει στη
αλλιώς
πες
το όνομα δεν υπάρχει

λίστα στη θέση με το index

Ο τρόπος επικοινωνίας μεταξύ των ενοτήτων ή υποπρογραμμάτων (διαδικασιών ή συναρτήσεων) και των προγραμμάτων διαφέρει από γλώσσα σε γλώσσα προγραμματισμού. Κατά βάση αυτή η επικοινωνία επιτυγχάνεται με τη χρήση των παραμέτρων, δηλαδή μεταβλητών που επιτρέπουν το πέρασμα της τιμής τους από ένα τμήμα προγράμματος σε ένα άλλο.

Η δομή δεδομένων, που είναι αποθηκευμένα τα ονόματα στο παράδειγμα 2.39, είναι μία λίστα με το όνομα names.

Ο αλγόριθμος που κωδικοποιείται είναι αυτός της σειριακής αναζήτησης κατάλληλα τροποποιημένος.

Παράδειγμα 2.40. **Σε ένα ραδιοφωνικό σταθμό το κόστος ενός διαφημιστικού μηνύματος σε σχέση με τα δευτερόλεπτα μετάδοσης, υπολογίζεται κλιμακωτά σύμφωνα με τον παρακάτω πίνακα.**

Χρονική Διάρκεια διαφήμισης (δευτερόλεπτα)	Κόστος (€ / δευτερόλεπτο)
Μέχρι 15	90
Μέχρι 30	75
Πάνω από 30	50

Α. Να αναπτύξετε πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο

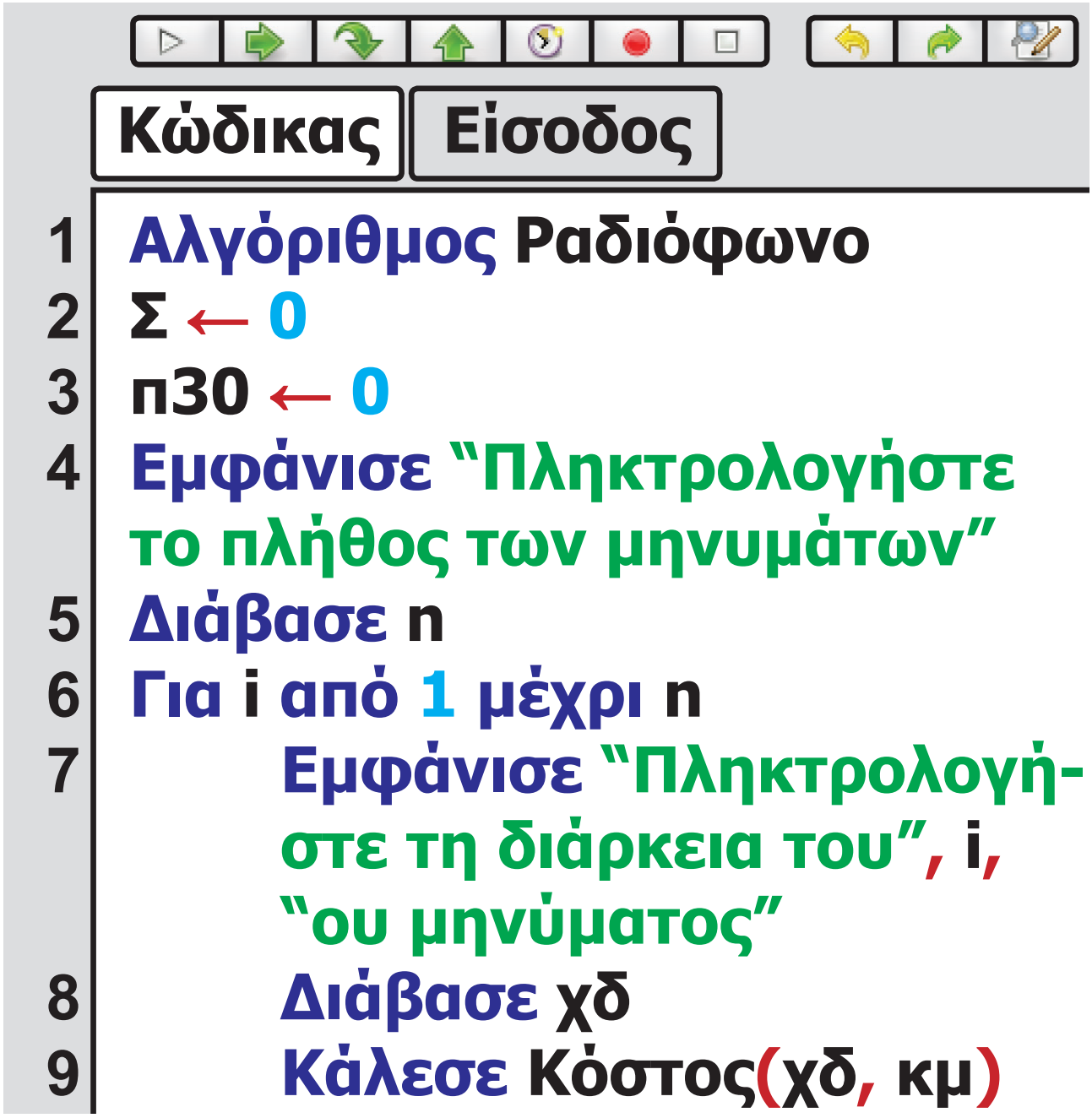
i. να διαβάζει το πλήθος των διαφορετικών μηνυμάτων που πρόκειται να μεταδώσει ο σταθμός την επόμενη εβδομάδα

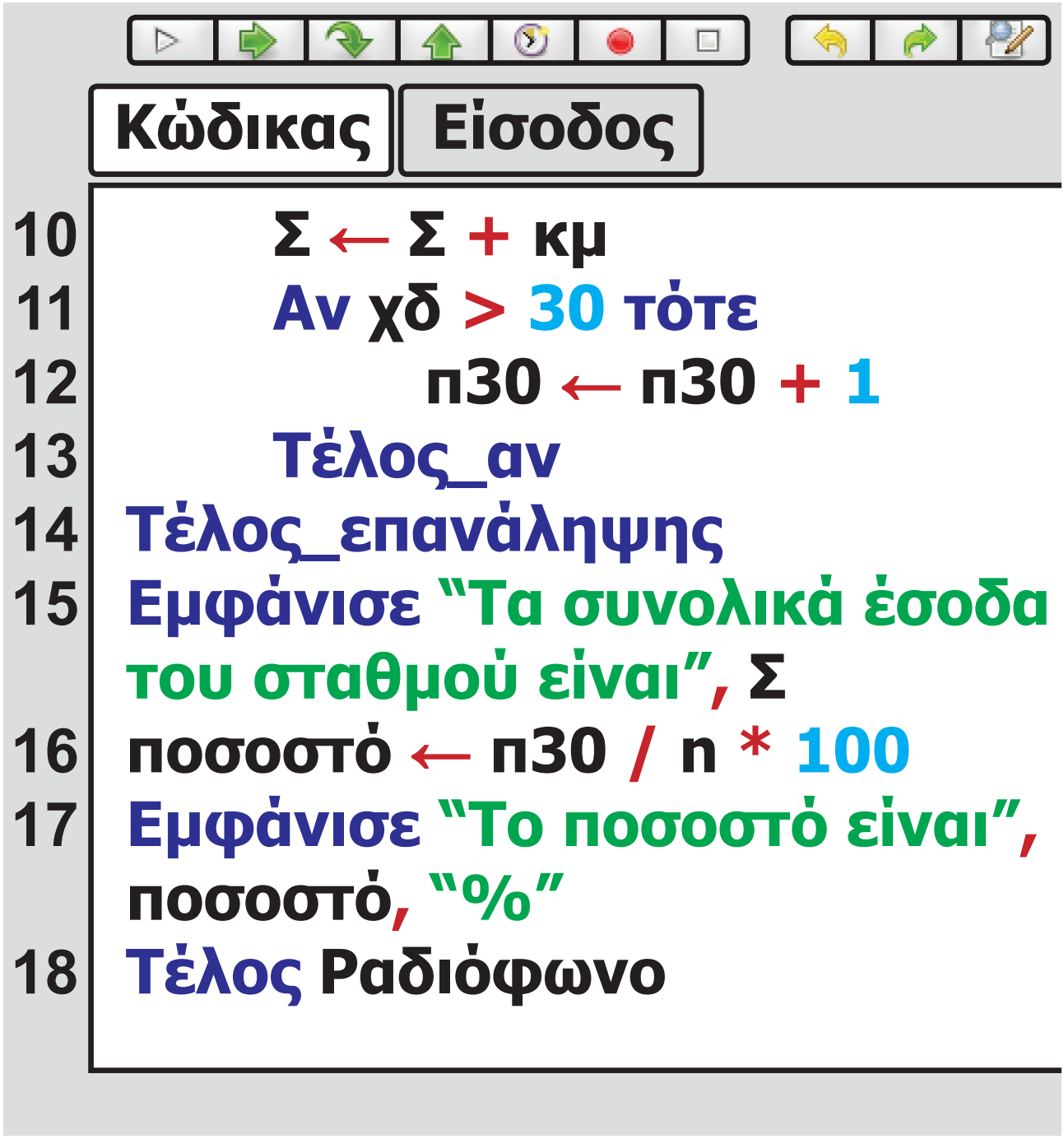
ii. να διαβάζει τη χρονική διάρκεια κάθε μηνύματος και να υπολογίζει καλώντας κατάλληλο υποπρόγραμμα το κόστος του

iii. να εμφανίζει με κατάλληλο μήνυμα τα συνολικά έσοδα του σταθμού καθώς και το ποσοστό (%) των μηνυμάτων με χρονική διάρκεια άνω των 30 δευτερολέπτων.

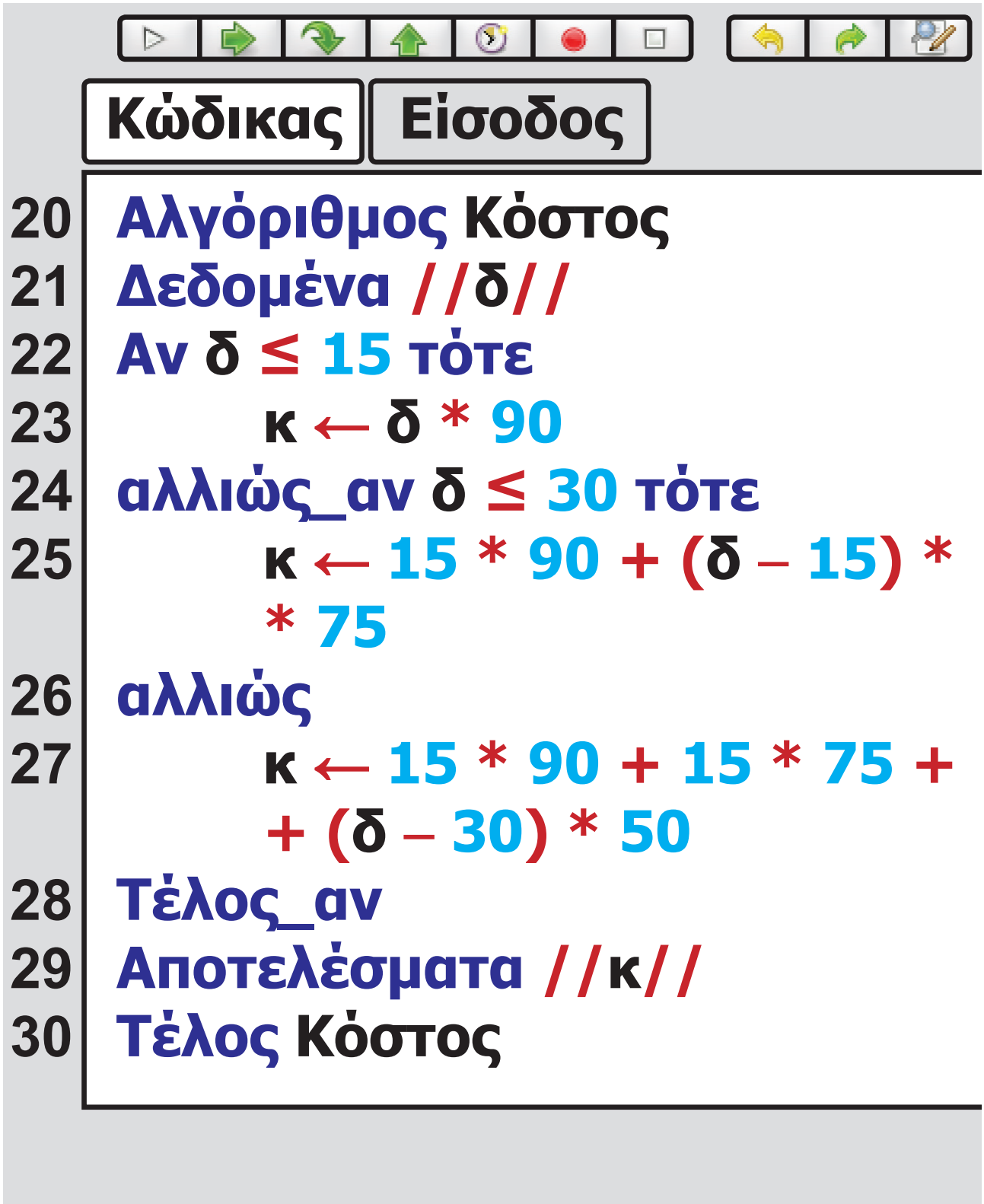
Β. Να γράψετε τον αλγόριθμο για τον υπολογισμό του κόστους του κάθε μηνύματος.

Ο κώδικας του προγράμματος είναι:




Κώδικας
Είσοδος
10 Σ ← Σ + κμ
11 Αν χδ > 30 τότε
12 π30 ← π30 + 1
13 Τέλος_αν
14 Τέλος_επανάληψης
15 Εμφάνισε "Τα συνολικά έσοδα του σταθμού είναι", Σ
16 ποσοστό ← π30 / n * 100
17 Εμφάνισε "Το ποσοστό είναι", ποσοστό, "%"
18 Τέλος Ραδιόφωνο

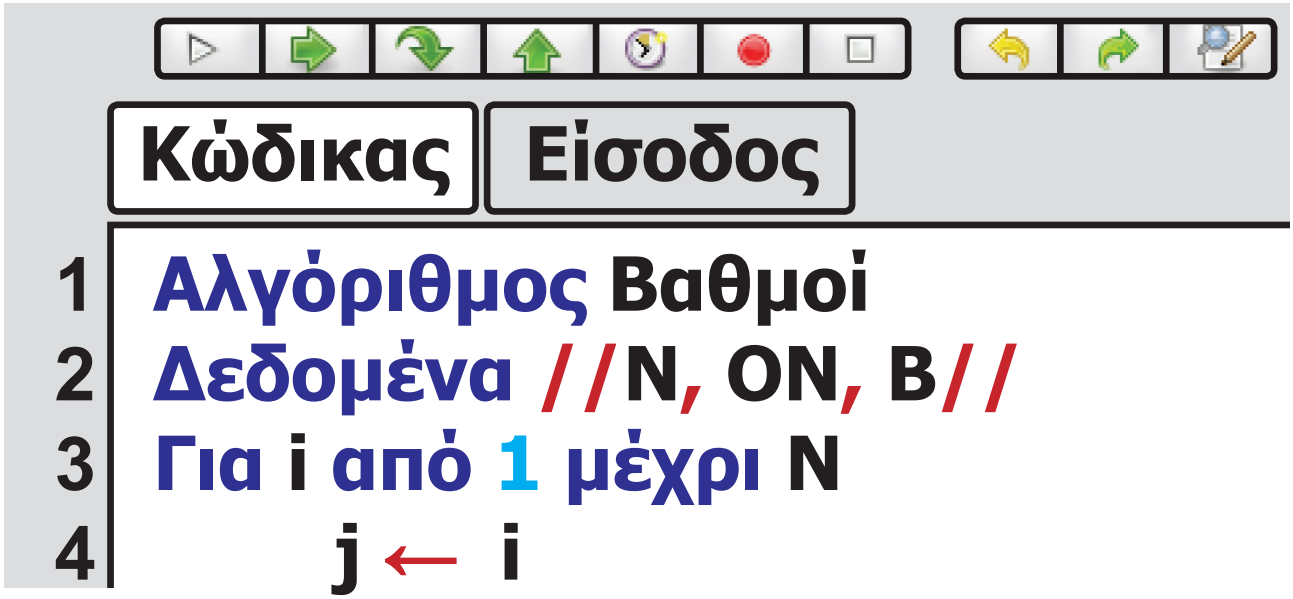
Ο κώδικας του υποπρογράμματος είναι:



Η χρήση της δομής του πίνακα, όταν αυτή δεν κρίνεται απαραίτητη, επαφίεται στη λογική του προγραμματιστή. Κατά βάση όμως στις εφαρμογές λογισμικού τα δεδομένα αποθηκεύονται σε δομές δεδομένων.

Κατά την κλήση του αλγορίθμου Κόστος, η μεταβλητή χδ μεταβιβάζει την τιμή της στη μεταβλητή δ και όταν ολοκληρώνεται η εκτέλεση του αλγορίθμου η μεταβλητή κ μεταβιβάζει την τιμή της στη μεταβλητή κμ.

Παράδειγμα 2.41. Να γραφεί πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο να δέχεται έναν πίνακα που περιέχει τα ονόματα των μαθητών ενός τμήματος και έναν παράλληλο πίνακα με το βαθμό απολυτηρίου τους. Το πρόγραμμα να ταξινομεί τον πίνακα με τα ονόματα σε φθίνουσα σειρά με βάση τους βαθμούς. Τέλος να εμφανίζει το όνομα του κάθε μαθητή και δίπλα τον βαθμό του.



Κώδικας | Είσοδος

```
5          max ← B[j]
6          Για k από i + 1 μέχρι N
7                Αν B[k]> max τότε
8                     j ← κ
9                     max ← B[j]
10               Τέλος_αν
11         Τέλος_επανάληψης
12         temp ← B[i]
13         B[i] ← B[j]
14         B[j] ← temp
15         tempX ← ON[i]
16         ON[i] ← ON[j]
17         ON[j] ← tempX
18   Τέλος_επανάληψης
19   Για i από 1 μέχρι N
20         Εμφάνισε ON[i], B[i]
21   Τέλος_επανάληψης
22   Τέλος Βαθμοί
```

Οι πίνακες στους οποίους χρησιμοποιούνται αντίστοιχοι δείκτες θέσης για την αποθήκευση συσχετιζόμενων τιμών ονομάζονται παράλληλοι πίνακες.

Με την εντολή
Δεδομένα //N, ON, B//
γνωστοποιούνται το μέγεθος και οι τιμές των πινάκων στο πρόγραμμα.

Στο πρόγραμμα του παραδείγματος 2.41 εφαρμόζεται ο αλγόριθμος της ταξινόμησης με επιλογή. Έχει τροποποιηθεί όμως σε σχέση με τον αλγόριθμο που παρουσιάστηκε στο προηγούμενο κεφάλαιο, ώστε να βρίσκει κάθε φορά το μέγιστο στοιχείο.

Παράδειγμα 2.42. Να αναπτυχθεί πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο να διαβάζει τους βαθμούς των μαθητών ενός τμήματος και να εμφανίζει με κατάλληλα μηνύματα το μέσο όρο των βαθμών καθώς και πόσοι από αυτούς είναι μεγαλύτεροι από το μέσο όρο. Να γίνεται έλεγχος ότι οι βαθμοί που εισάγονται είναι ακέραιες τιμές μεταξύ του 0 και του 20. Μετά την εισαγωγή κάθε βαθμού το πρόγραμμα να ερωτά τον χρήστη αν θέλει να συνεχίσει την καταχώρηση εμφανίζοντας το μήνυμα «Θα συνεχίσετε; Πληκτρολογήστε Ν ή Ο» και ανάλογα με την απάντησή του (αποδεκτές τιμές μόνο τα γράμματα του ελληνικού αλφαβήτου «Ν»,

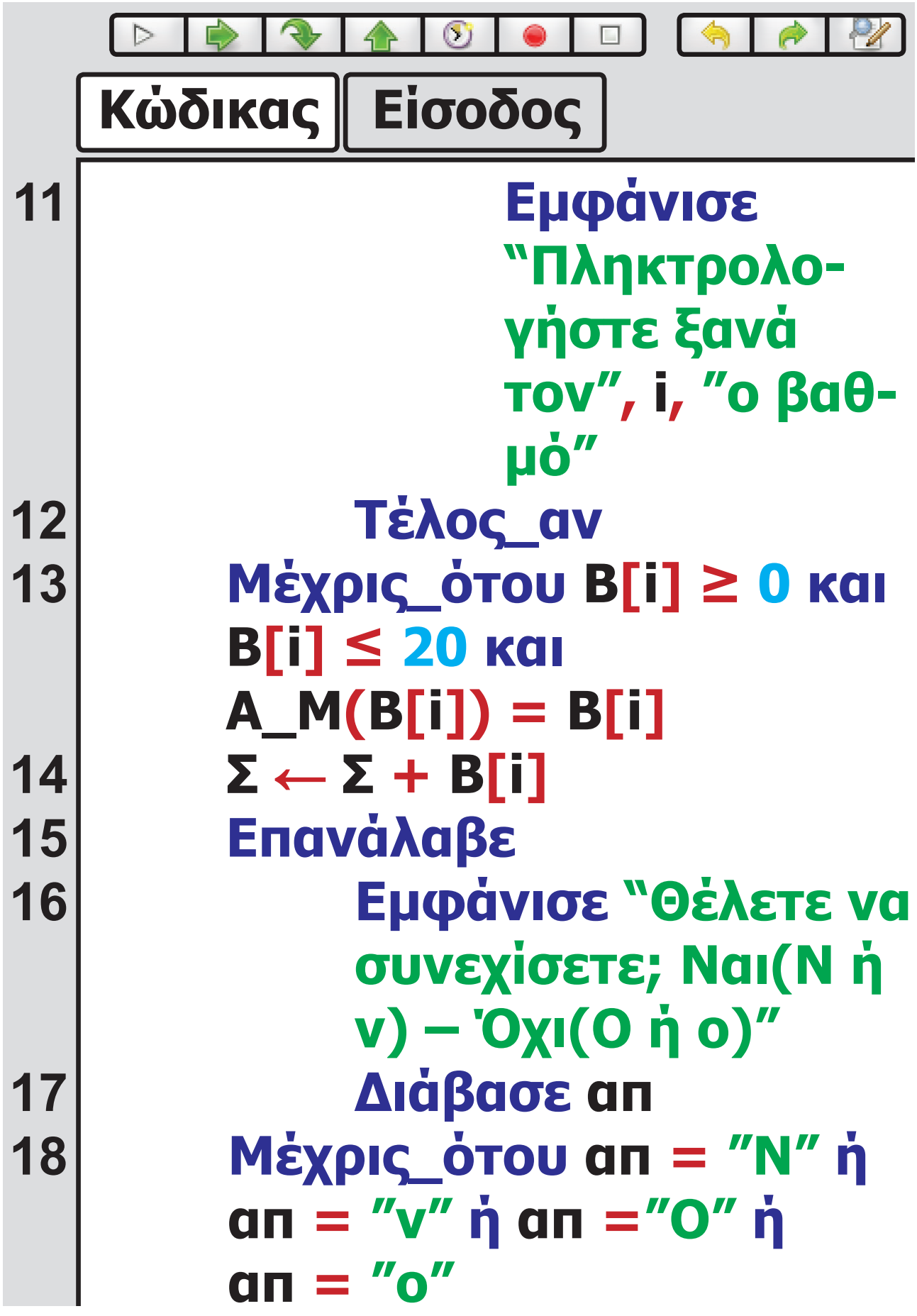
«ν», «Ο» και «ο»), να συνεχίζει την καταχώριση ή να εμφανίζει τα αποτελέσματα της επεξεργασίας.

Κατά την ανάπτυξη αυτού του προγράμματος θα χρησιμοποιηθεί η δομή δεδομένων του πίνακα, γιατί διαφορετικά δεν θα υπάρχει η δυνατότητα να γίνει η επεξεργασία για τον υπολογισμό του πλήθους αυτών που είναι πάνω από το μέσο όρο.

Γενικά η χρήση πινάκων είναι απαραίτητη αν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσης.

Κώδικας | **Είσοδος**

```
1   Αλγόριθμος Βαθμοί
2   i ← 0
3   Σ ← 0
4   Επανάλαβε
5       i ← i+1
6       Εμφάνισε "Πληκτρολογήστε τον", i, "ο βαθμό"
7       Επανάλαβε
8           Διάβασε Β[i]
9           Αν Β[i] < 0 ή
            Β[i] > 20 ή
            Α_Μ(Β[i]) ≠ Β[i]
            τότε
10              Εμφάνισε
                "Λάθος
                βαθμός."
```

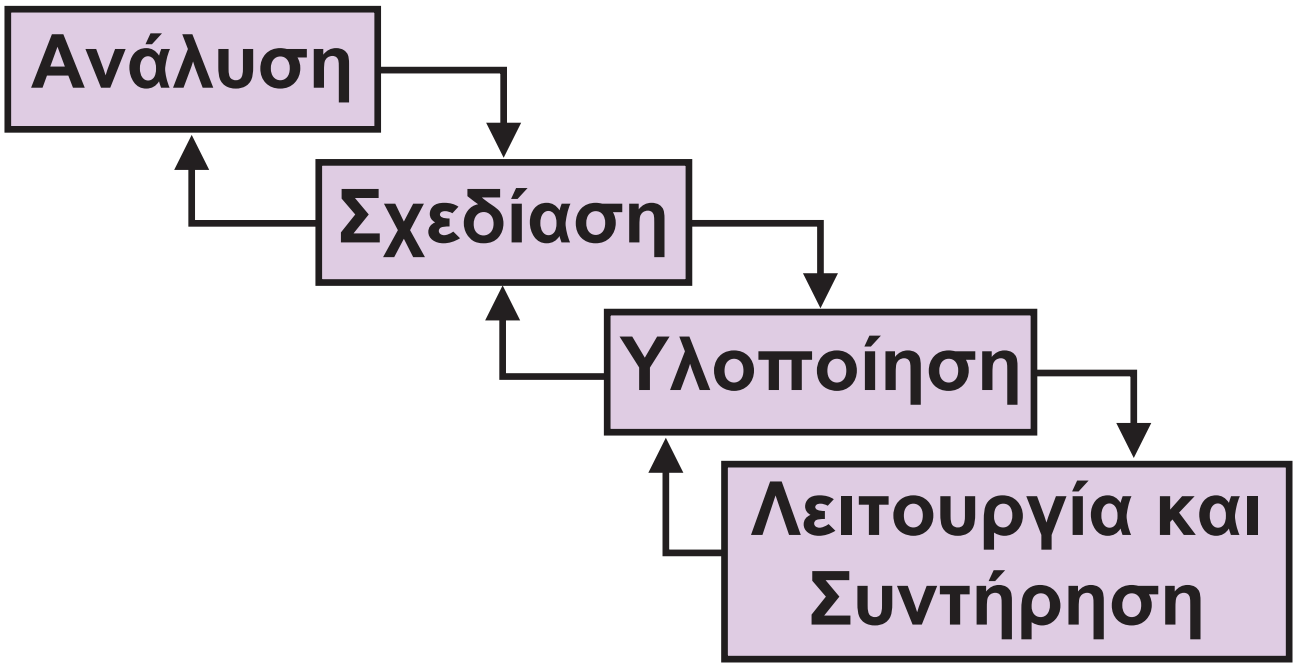

Κώδικας
Είσοδος
11
Εμφάνισε "Πληκτρολογήστε ξανά τον", i, "ο βαθμό"
12
Τέλος_αν
13
Μέχρις_ότου Β[i] ≥ 0 και Β[i] ≤ 20 και Α_Μ(Β[i]) = Β[i]
14
Σ ← Σ + Β[i]
15
Επανάλαβε
16
Εμφάνισε "Θέλετε να συνεχίσετε; Ναι(Ν ή ν) – Όχι(Ο ή ο)"
17
Διάβασε απ
18
Μέχρις_ότου απ = "Ν" ή απ = "ν" ή απ ="Ο" ή απ = "ο"

Κώδικας | Είσοδος

```
19  Μέχρις_ότου απ = "Ο" ή
    απ = "ο"
20  Ν ← i
21  ΜΟ ← Σ/Ν
22  πλ ← 0
23  Για i από 1 μέχρι Ν
24          Αν Β[i] > ΜΟ τότε
25                  πλ ← πλ + 1
26          Τέλος_αν
27  Τέλος_επανάληψης
28  Εμφάνισε "Ο μέσος όρος είναι:
    ", ΜΟ
29  Εμφάνισε "Υπάρχουν", πλ,
    "βαθμοί πάνω από το μέσο
    όρο"
30  Τέλος Βαθμοί
```

2.3.3 Κύκλος ζωής εφαρμογής λογισμικού

Ένα πρόγραμμα αρχίζει την ζωή του από την στιγμή που θα καθοριστούν οι απαιτήσεις του, οι προδιαγραφές του και παύει να ζει όταν εξαντληθούν όλα τα περιθώρια συντήρησής του (προσθήκες, αλλαγές, βελτιώσεις). Ο **Κύκλος Ζωής** ενός προγράμματος περιλαμβάνει διάφορες φάσεις οι οποίες αλληλεπιδρούν μεταξύ τους. Οι φάσεις απεικονίζονται στο διάγραμμα της εικόνας 2.34.



Εικόνα 2.34. Κύκλος Ζωής Προγράμματος.

Στη φάση **Ανάλυσης** που ακολουθεί τον προσδιορισμό του προβλήματος από τον πελάτη, καταγράφονται αναλυτικά τα δεδομένα και τα ζητούμενα του προβλήματος και ζητούνται οι απαραίτητες διευκρινήσεις από τον πελάτη, σε όσα σημεία οι προδιαγραφές παρουσιάζουν ασάφεια.

Στη φάση της **Σχεδίασης** καθορίζεται η δομή του προγράμματος, οι ενότητες (υποπρογράμματα) από τις οποίες θα αποτελείται το πρόγραμμα και αναζητούνται έτοιμες ενότητες (modules) από παλιότερα προγράμματα που μπορούν να χρησιμοποιηθούν και σ' αυτό το πρόγραμμα. Ακόμη επιλέγονται οι αλγόριθμοι και οι δομές δεδομένων που θα χρησιμοποιηθούν σε κάθε ενότητα.

Στην επόμενη φάση της **Υλοποίησης** του προγράμματος επιλέγεται η κατάλληλη γλώσσα προγραμματισμού για το συγκεκριμένο πρόγραμμα όπου συντάσσεται το πηγαίο πρόγραμμα και μεταφράζεται από έναν μεταγλωττιστή, ώστε αυτό

να γίνει κατανοητό από τον υπολογιστή. Η μετάφραση θα εντοπίσει πιθανά συντακτικά λάθη. Τα λάθη διορθώνονται και ακολουθεί ξανά μετάφραση του προγράμματος, έως την οριστική εξάλειψή τους. Το πρόγραμμα που προκύπτει είναι το εκτελέσιμο πρόγραμμα.

Στην επόμενη φάση της **Λειτουργίας και Συντήρησης** θα γίνουν όλες οι προσαρμογές και βελτιώσεις που χρειάζονται προκειμένου το πρόγραμμα να συνεχίσει να χρησιμοποιείται. Αυτές ζητούνται όταν διαφοροποιούνται τα δεδομένα του προβλήματος, όταν ο χρήστης ζητήσει νέες λειτουργίες ή προκύψουν κάποια λογικά λάθη που δεν είχαν διαπιστωθεί στον

έλεγχο του αλγορίθμου και μπορεί, για παράδειγμα, να αφορούν την επικοινωνία των ενοτήτων. Για την υλοποίηση των αλλαγών μπορεί να επαναληφθεί η εκτέλεση της φάσης της ανάλυσης και σχεδίασης και άρα όλων των υπολοίπων φάσεων. Έτσι πραγματοποιείται συνολικός έλεγχος του προγράμματος που έχει ως συνέπεια και την καταγραφή σχετικών σχολίων για την τεκμηρίωση.

Στα διάφορα πακέτα λογισμικού, είτε είναι γλώσσες προγραμματισμού, είτε πακέτα εφαρμογών, είτε λειτουργικά συστήματα, δίπλα στο εμπορικό όνομα του λογισμικού, συνηθίζεται να υπάρχει ο αριθμός της έκδοσής του. Ο αριθμός έκδοσης του

πακέτου λογισμικού δείχνει ακριβώς τις αλλαγές που έχουν πραγματοποιηθεί από την αρχική του εμφάνιση.

Όταν οι αλλαγές είναι σημαντικές, δηλαδή έχουν προστεθεί νέες λειτουργίες, εντολές, προγράμματα, ο αριθμός έκδοσης αυξάνει κατά ακέραιο αριθμό (GeoGebra 3.0, GeoGebra 4.0), ενώ, όταν οι αλλαγές είναι μικρότερες, αυξάνεται κατά δέκατα ή εκατοστά (GeoGebra 4.1, GeoGebra 4.2).
http://www.geogebra.org

Οι εκδόσεις beta μιας εφαρμογής λογισμικού που αναπτύσσει μια εταιρεία δίνονται για έλεγχο συνήθως στους πελάτες της. Οι beta testers παίρνουν την τελική έκδοση δωρεάν ή με σημαντική έκπτωση.

Παράδειγμα 2.43. Ανάπτυξη μιας εφαρμογής αριθμομηχανής για κινητά τηλέφωνα με λειτουργικό Android.

Αρχικά στο στάδιο της ανάλυσης θα πρέπει να καταγραφούν οι απαιτήσεις από την αριθμομηχανή, δηλαδή τι είδους πράξεις θα είναι σε θέση να διεκπεραιώσει. Στη συνέχεια, στη φάση της σχεδίασης θα καθοριστούν οι ενότητες από τις οποίες θα αποτελείται η εφαρμογή, όπως ενότητα που θα υλοποιεί τις απλές αριθμητικές πράξεις ή ενότητα που θα υπολογίζει τριγωνομετρικούς αριθμούς ή τιμές μαθηματικών συναρτήσεων. Επίσης θα επιλεγούν οι αλγόριθμοι και οι δομές δεδομένων που θα χρησιμοποιηθούν σε

κάθε ενότητα. Στη φάση της υλοποίησης θα πρέπει να επιλεγεί αρχικά η γλώσσα προγραμματισμού στην οποία θα συνταχθεί το πρόγραμμα. Μία γλώσσα οπτικού προγραμματισμού κατάλληλη για ανάπτυξη εφαρμογών για Android είναι η Google AppInventor. Αφού συνταχθούν όλες οι ενότητες, θα γίνει έλεγχος για τη σωστή τους επικοινωνία. Η αριθμομηχανή είναι έτοιμη για να δοθεί στον τελικό χρήστη. Σε περίπτωση που στη φάση της Λειτουργίας και της Συντήρησης κάποιος χρήστης ζητήσει την επέκταση των δυνατοτήτων της αριθμομηχανής, όπως να προστεθεί στατιστική επεξεργασία δεδομένων, τότε θα πρέπει ο προγραμματιστής πιθανότατα να επαναλάβει όλες τις

φάσεις ώστε να καλύψει τις νέες απαιτήσεις.

Εικόνα 2.35. Λογότυπο του Λειτουργικού Συστήματος Android.

Στην ιστοσελίδα
http://www.code.org
υπάρχουν αρκετές δραστηριότητες κατάλληλες για την εκμάθηση οπτικού προγραμματισμού από μαθητές.

Ωστόσο μια εφαρμογή λογισμικού με σύνθετες λειτουργίες υλοποιείται κατά βάση από μία ομάδα

προγραμματιστών που εργάζονται παράλληλα ώστε να μειωθεί ο χρόνος και το κόστος της υλοποίησης. Το κάθε μέλος της ομάδας ανάπτυξης του λογισμικού αναλαμβάνει τη συγγραφή συγκεκριμένων ενοτήτων. Υπάρχουν ταυτόχρονα και μέλη τα οποία είναι υπεύθυνα για το συντονισμό της διαδικασίας ανάπτυξης, τη συνένωση των ενοτήτων σε μία ενιαία εφαρμογή ή για τους ελέγχους του προϊόντος. Τα μεγάλα έργα πληροφορικής εμπλέκουν πολλά διαφορετικά και εξειδικευμένα πρόσωπα, ενώ αποτελούνται από δραστηριότητες των οποίων η σειρά είναι σημαντική.

Ο συντονιστής του έργου ανάπτυξης μιας εφαρμογής λογισμικού

(project manager) δημιουργεί και ενημερώνει το πλάνο εργασίας, στελεχώνει το έργο, και παρακολουθεί και ελέγχει την πρόοδό του.

Οι εφαρμογές που αναπτύσσονται από εταιρείες λογισμικού είναι συνήθως **Κλειστού Κώδικα** (proprietary – closed source). Δεν υπάρχει η δυνατότητα για τον χρήστη να δει τον κώδικα ή να επιφέρει κάποια αλλαγή σε αυτόν. Οι βελτιώσεις στα κλειστά λογισμικά γίνονται μέσω αναβαθμίσεων που παρέχονται κατά καιρούς από τις εταιρείες κατασκευής τους, κυρίως μέσω διαδικτύου. Επίσης υπάρχει συνήθως κόστος απόκτησης και απαγόρευση της αναδιανομής της εφαρμογής.

Αντίθετα στις εφαρμογές **Ελεύθερου Λογισμικού / Λογισμικού Ανοικτού Κώδικα** (ΕΛ / ΛΑΚ free / open source) ο κώδικας είναι διαθέσιμος, συνεπώς ο καθένας μπορεί ελεύθερα να χρησιμοποιεί, να μελετά τον τρόπο λειτουργίας, να αντιγράφει, να διανέμει και να τροποποιεί την εφαρμογή προσθέτοντας δικές του βελτιώσεις ή νέες λειτουργίες. Με βάση αυτήν τη φιλοσοφία δημιουργούνται ομάδες προγραμματιστών που μοιράζονται τις αλλαγές που κάνουν στον κώδικα με σκοπό τη βελτίωσή του. Έτσι δημιουργείται ένα παγκόσμιο ανοικτό δίκτυο προγραμματιστών, οι οποίοι συνεργάζονται κυκλοφορώντας νέες εκδόσεις λογισμικού και συμβάλλοντας καθημερινά στη δημιουργία

νέων κοινών αγαθών. Το Διαδίκτυο αποτελεί το βασικό μέσο συνεργασίας των προγραμματιστών αλλά και του τρόπου πρόσβασης στο διαθέσιμο Ελεύθερο Λογισμικό.

Πληροφορίες σχετικά με το Λογισμικό Ανοικτού Κώδικα μπορούν να βρεθούν στις διευθύνσεις:

http://www.ellak.gr/

http://www.fsf.org/

http://opensource.org/

Εικόνα 2.36.
Λίνους Τόρβαλντς (Linus Torvalds)

Δημιουργός και συντονιστής της ομάδας εργασίας του λειτουργικού συστήματος ανοιχτού κώδικα Linux

http://www.linux.org/

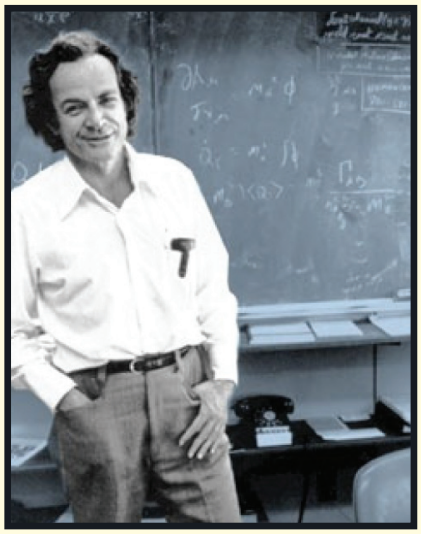
Ανακεφαλαίωση

Στην ενότητα αυτή επιχειρήθηκε να εξηγηθεί τι είναι πρόγραμμα και έγινε αναφορά στις διάφορες γενιές γλωσσών προγραμματισμού και στα βασικά Προγραμματιστικά Υποδείγματα. Επισημάνθηκαν οι αρχές του Δομημένου Προγραμματισμού και τα πλεονεκτήματα αυτής της μεθοδολογίας συγγραφής προγραμμάτων. Επίσης αναλύθηκε η διαδικασία μετάφρασης των προγραμμάτων και οι τρόποι εύρεσης

και διόρθωσης συντακτικών λαθών. Με παραδείγματα ανάπτυξης προγραμμάτων σε διάφορα προγραμματιστικά περιβάλλοντα παρουσιάστηκε η επαναχρησιμοποίηση κώδικα μέσω διαδικασιών ή συναρτήσεων. Ακόμη χρησιμοποιήθηκαν βασικοί αλγόριθμοι του προηγούμενου κεφαλαίου οι οποίοι κωδικοποιήθηκαν σε προγράμματα. Τέλος περιγράφηκε ο Κύκλος Ζωής ενός προγράμματος και έγινε διάκριση μεταξύ των εφαρμογών κλειστού και ανοικτού κώδικα.

Εικόνα 2.37. Σέυμουρ Παπέρτ (Seymour Papert)

Επινόησε τη γλώσσα προγραμματισμού LOGO ως ένα τεχνολογικό εργαλείο που θα βελτιώνει τους τρόπους με τους οποίους τα παιδιά σκέπτονται και επιλύουν προβλήματα.



Εικόνα 2.38. Ρίτσαρντ Φάινμαν (Richard Feynman) Βραβείο Νόμπελ Φυσικής.

«Η επιστήμη υπολογιστών δεν είναι τόσο παλιά όσο η φυσική, υστερεί χρονικά μερικούς αιώνες. Ωστόσο, αυτό δεν σημαίνει ότι υπάρχουν λιγότερα στο πιάτο του επιστήμονα των υπολογιστών απ' ότι σε αυτό του φυσικού: μπορεί να είναι νεότερη αλλά είχε μια πολύ πιο έντονη γέννηση!»

Λέξεις κλειδιά

Πρόγραμμα, Γλώσσες Προγραμματισμού, Προγραμματιστικό Υπόδειγμα, Δομημένος Προγραμματισμός, Συντάκτης, Μετάφραση Προγράμματος, Μεταγλωττιστής, Διερμηνευτής, Αντικείμενο Πρόγραμμα, Συνδέτης, Εκτελέσιμο Πρόγραμμα, Συντακτικά Λάθη, Διαδικασίες, Συναρτήσεις, Βιβλιοθήκες, Κύκλος Ζωής, Ανοικτό Λογισμικό

Ερωτήσεις - Θέματα προς συζήτηση - Δραστηριότητες

1. Τι είναι πρόγραμμα;

2. Ποια είναι τα πλεονεκτήματα των γλωσσών υψηλού επιπέδου σε σχέση με τις γλώσσες προηγούμενης γενιάς;

3. Ποια η διαφορά των γλωσσών του οπτικού προγραμματισμού με αυτές που διαθέτουν οπτικό περιβάλλον προγραμματισμού;

4. Τι εννοούμε με τη φράση «Προγραμματιστικό Υπόδειγμα»;

5. Τι είναι ο δομημένος προγραμματισμός και ποια είναι τα τρία κύρια χαρακτηριστικά του;

6. Ποια είναι τα πλεονεκτήματα του Δομημένου Προγραμματισμού;

7. Σε τι χρησιμεύουν τα μεταφραστικά προγράμματα;

8. Ποιες οι διαφορές του μεταγλωττιστή και του διερμηνευτή στην εύρεση των συντακτικών λαθών;

9. Τι εργαλεία πρέπει να περιέχει ένα προγραμματιστικό περιβάλλον;

10. Πώς επιτυγχάνεται η επαναχρησιμοποίηση κώδικα και ποια τα πλεονεκτήματά της;

11. Ποιες είναι φάσεις του Κύκλου Ζωής ενός προγράμματος;

12. Ποιες δυνατότητες έχει ο χρήστης με τα Λογισμικά Ανοικτού Κώδικα;

13. Να χαρακτηρίσετε με Σωστό ή Λάθος τις παρακάτω προτάσεις:
    - **Α.** Οι εντολές στις συμβολικές γλώσσες αποτελούνται από ακολουθίες 0 και 1.
    - **Β.** Η μεταφερσιμότητα είναι χαρακτηριστικό των γλωσσών υψηλού επιπέδου.
    - **Γ.** Η SQL αποκρύπτει τις τεχνικές του προγραμματισμού.
    - **Δ.** Η γλώσσα LOGO αναπτύχθηκε για εκπαιδευτικού σκοπούς.

Ε. Μία εφαρμογή λογισμικού με σύνθετες λειτουργίες υλοποιείται κατά βάση από μία ομάδα προγραμματιστών.

14. Να επιλέξετε για κάθε μία από τις παρακάτω φράσεις το γράμμα που οδηγεί σε σωστή πρόταση:

Α. Οι εντολές ενός προγράμματος γράφονται σε ένα πρόγραμμα που ονομάζεται:

i. Συντάκτης
ii. Μεταγλωττιστής
iii. Διερμηνευτής
iv. Συνδέτης

Β. Ο μεταγλωττιστής επισημαίνει:

i. όλα τα λάθη του προγράμματος

ii. μόνο τα λογικά λάθη του προγράμματος
iii. μόνο τα συντακτικά λάθη του προγράμματος
iv. μόνο τα λάθη που οφείλονται σε αναγραμματισμούς εντολών

Γ. Σε μία εφαρμογή ανοικτού κώδικα:
i. δεν είναι δυνατόν να προσπελάσει ο χρήστης τον κώδικα
ii. πρέπει ο χρήστης να περιμένει τις αναβαθμίσεις από την εταιρεία κατασκευής της.
iii. ο χρήστης δεν μπορεί να προσθέσει λειτουργίες.

iv. ο χρήστης μπορεί να δει τον κώδικα και να προσθέσει λειτουργίες.

15. Να πραγματοποιήσετε συζήτηση στην τάξη για το πώς θα μπορούσε να υλοποιηθεί προγραμματιστικά το υπολογιστικό πρόβλημα που περιγράφεται στο παράδειγμα 2.1. στο κεφάλαιο 2.1.

16. Να πραγματοποιήσετε συζήτηση στην τάξη σχετικά με το επάγγελμα του προγραμματιστή. Αναζητήστε πληροφορίες στη διεύθυνση http://www.eoppep.gr

17. Να αναπτύξετε πρόγραμμα στο προγραμματιστικό περιβάλλον Διερμηνευτής Ψευδογλώσσας (ViALGOL) το οποίο

Α. να δέχεται τα ονόματα 20 εταιρειών καθώς και τα έσοδα τους για το προηγούμενο έτος.

Β. να εκτυπώνει το μέσο όρο των εσόδων.

Γ. να εκτυπώνει το όνομα της εταιρείας με τα μεγαλύτερα έσοδα.

Δ. να διαβάζει το όνομα μιας εταιρείας και να εμφανίζει τα έσοδα της. Αν η εταιρεία δεν υπάρχει να εμφανίζει κατάλληλο μήνυμα.

18. Να αναπτύξετε κατάλληλα υποπρογράμματα για τα ερωτήματα Β και Γ της δραστηριότητας 17 και να γράψετε τον αντίστοιχο κώδικα του προγράμματος. Τα υποπρογράμματα θα πρέπει να επιστρέφουν το μέσο όρο και το όνομα της εταιρείας αντίστοιχα.

Περιεχόμενα

Βάσει του ν. 3966/2011 τα διδακτικά βιβλία του Δημοτικού, του Γυμνασίου, του Λυκείου, των ΕΠΑ.Λ. και των ΕΠΑ.Σ. τυπώνονται από το ΙΤΥΕ - ΔΙΟΦΑΝΤΟΣ και διανέμονται δωρεάν στα Δημόσια Σχολεία. Τα βιβλία μπορεί να διατίθενται προς πώληση, όταν φέρουν στη δεξιά κάτω γωνία του εμπροσθόφυλλου ένδειξη «ΔΙΑΤΙΘΕΤΑΙ ΜΕ ΤΙΜΗ ΠΩΛΗΣΗΣ». Κάθε αντίτυπο που διατίθεται προς πώληση και δεν φέρει την παραπάνω ένδειξη θεωρείται κλεψίτυπο και ο παραβάτης διώκεται σύμφωνα με τις διατάξεις του άρθρου 7 του νόμου 1129 της 15/21 Μαρτίου 1946 (ΦΕΚ 1946,108, Α').