ΕΝΟΤΗΤΑ ΙΙΙ

Υλοποίηση σε Προγραμματιστικό Περιβάλλον

Σκοπός ενότητας:

Να αποκτήσουμε δεξιότητες υλοποίησης αλγορίθμων σε προγραμματιστικό περιβάλλον.

Ειδικοί σκοποί της ενότητας:

- ❑ Να μπορούμε να μετατρέπουμε έναν αλγόριθμο επίλυσης ενός προβλήματος σε πρόγραμμα.
- ❑ Να μπορούμε να κωδικοποιούμε τον αλγόριθμο σε κατάλληλο προγραμματιστικό περιβάλλον γλωσσών υψηλού επιπέδου.
- ❑ Να μπορούμε να διορθώνουμε, να βελτιώνουμε και να επεκτείνουμε τα προγράμματα που δημιουργούμε.

Περιεχόμενα:

- Είδη, Τεχνικές και Περιβάλλοντα Προγραμματισμού.
- Η Γλώσσα Pascal
- Βασικές Εντολές
- Εντολές Επιλογής και Αποφάσεων
- Εντολές Επανάληψης
- Υποπρογράμματα
- Τύποι Δεδομένων
- Στατικές Δομές Δεδομένων
- Αρχεία
- Δυναμικές Δομές Δεδομένων
- Έλεγχος - Εκσφαλμάτωση Προγράμματος

ΚΕΦΑΛΑΙΟ 7
ΕΙΔΗ, ΤΕΧΝΙΚΕΣ ΚΑΙ ΠΕΡΙΒΑΛΛΟΝΤΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σκοπός κεφαλαίου:

Να γνωρίσουμε τον τρόπο ανάπτυξης ενός προγράμματος, και τη διαδικασία μετάφρασης και εκτέλεσής του.

Ειδικοί σκοποί:

- ❑ Να γνωρίσουμε τα κυριότερα είδη προγραμματισμού και τα κυριότερα χαρακτηριστικά τους.
- ❑ Να γνωρίσουμε τη διαδικασία δημιουργίας εκτελέσιμου κώδικα.
- ❑ Να γνωρίσουμε τα διάφορα είδη γλωσσών και μεταφραστικών προγραμμάτων.
- ❑ Να γνωρίσουμε κατηγορίες προγραμματιστικών περιβαλλόντων.

7.1. ΑΝΑΠΤΥΞΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Όπως γνωρίζουμε, το πρόγραμμα είναι ένα σύνολο εντολών για την εκτέλεση ορισμένων λειτουργιών από τον υπολογιστή. Τα προγράμματα χρησιμοποιούνται για την επίλυση προβλημάτων από τον υπολογιστή και βασίζονται σε έναν ή περισσότερους αλγορίθμους.

Κατά τη διάρκεια της επίλυσης του προβλήματος με τον υπολογιστή, είναι απαραίτητο να χρησιμοποιήσουμε ορισμένες τυποποιημένες διαδικασίες που χαρακτηρίζονται ως ***Κύκλος Ανάπτυξης Προγράμματος*** και εμφανίζονται στον παρακάτω πίνακα.

κύκλος ανάπτυξης προγράμματος	
σχεδίαση προγράμματος	
	α. ***κατανόηση του προβλήματος***
	β. ***μέθοδοι επίλυσης του προβλήματος***
Λοιπές διαδικασίες	
	γ. ***κωδικοποίηση - μετάφραση του προγράμματος***
	δ. ***έλεγχος του προγράμματος***
	ε. ***τεκμηρίωση του προγράμματος***

Από τις διαδικασίες του κύκλου ανάπτυξης προγράμματος, οι δύο είναι γνωστές και ως ***Σχεδίαση Προγράμματος.***

Για να λύσουμε ένα πρόβλημα, είναι απαραίτητο πρώτα να το καταλάβουμε.

Πρέπει να προσδιορίσουμε τα δεδομένα, τα ζητούμενα και τους περιορισμούς ή τις συνθήκες που είναι απαραίτητες για την επίλυσή του. Μερικές φορές μάλιστα μπορούμε να χωρίσουμε ένα σύνθετο πρόβλημα σε άλλα επιμέρους απλούστερα προβλήματα.

Υπάρχουν διάφοροι τρόποι για την ανάπτυξη του αλγορίθμου. Οι πιο συνηθισμένες παραστάσεις αλγόριθμου είναι:
- Φραστική
- Ψευδοκώδικας
- Λογικό διάγραμμα

Από τους τρόπους επίλυσης ενός προβλήματος θα επιλέξουμε εκείνον που είναι ο πιο ***κατάλληλος*** για την υλοποίησή του στον υπολογιστή που διαθέτουμε. Για να βρει ο προγραμματιστής ποιος είναι ο πιο κατάλληλος τρόπος λαμβάνει, υπόψη του ***τον όγκο των δεδομένων, την πολυπλοκότητα των υπολογισμών και τις δυνατότητες του υπολογιστή τόσο σε υλικό όσο και σε λογισμικό***.

Αφού επιλεγεί ο τρόπος επίλυσης, αποτυπώνεται σε διαδοχικά βήματα ο αλγόριθμος για τη λύση του προβλήματος.

Μετά την αποτύπωση του αλγορίθμου με μία από τις προηγούμενες μεθόδους, ακολουθεί η κωδικοποίησή του σε μία από τις γλώσσες προγραμματισμού που είναι στη διάθεση του χρήστη για τον υπολογιστή που θα χρησιμοποιήσει.

Για την εκτέλεση κάθε προγράμματος κωδικοποιημένου σε μία γλώσσα προγραμματισμού, απαιτείται και το αντίστοιχο μεταφραστικό πρόγραμμα (compiler ή interpreter), το οποίο μετατρέπει τον κώδικα του προγράμματος σε κώδικα μηχανής.

7.2. ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Η εξέλιξη των υπολογιστών αποδεικνύει ότι η Επεξεργασία Δεδομένων εξαρτάται περισσότερο από την ανάπτυξη του Λογισμικού και λιγότερο από την ανάπτυξη του Υλικού. Μέρος της ανάπτυξης του Λογισμικού αποτελούν οι διάφορες γλώσσες προγραμματισμού, που έχουν ως στόχο τη διευκόλυνση των προγραμματιστών στη λύση των διαφόρων προβλημάτων τους.

7.2.1. Γλώσσες μηχανής

Κάθε υπολογιστής μπορεί να κατανοήσει και να εκτελέσει εντολές οι οποίες είναι γραμμένες με έναν καθορισμένο τρόπο, που έχει σχέση με τη δική του Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ). Οι εντολές αυτές είναι γραμμένες σε γλώσσα μηχανής. Με εντολές σε γλώσσα μηχανής ο προγραμματιστής μπορεί να καθοδηγήσει τον υπολογιστή να εκτελέσει τις στοιχειώδεις λειτουργίες του.

Οι εντολές σε γλώσσα μηχανής αποτελούνται από μία ακολουθία 0 και 1 σταθερού ή μεταβλητού πλήθους. Κάθε τέτοια ακολουθία είναι μία εντολή προς την ΚΜΕ να εκτελέσει μία στοιχειώδη λειτουργία, όπως πρόσθεση, αφαίρεση, αποθήκευση στην κεντρική μνήμη κλπ.

Τα προγράμματα που είναι γραμμένα σε γλώσσα μηχανής ενός υπολογιστή είναι εξαρτημένα από τη δομή του συγκεκριμένου υπολογιστή και μπορούν να ***τρέξουν*** μόνο σε υπολογιστές, που είναι όμοιοι ή έχουν συμβατές ΚΜΕ. Ένα πρόγραμμα γραμμένο σε γλώσσα μηχανής είναι εύκολα κατανοητό από τον υπολογιστή για τον οποίο γράφτηκε αλλά δύσκολα κατανοητό από τον άνθρωπο, ακόμη και απ' αυτόν που το έγραψε, μετά από κάποιο χρονικό διάστημα.

Ο προγραμματισμός σε γλώσσα μηχανής ήταν ο πρώτος τρόπος προγραμματισμού των υπολογιστών. Ο προγραμματισμός στη γλώσσα αυτή θεωρείται σήμερα ένας άθλος για τον προγραμματιστή της εποχής εκείνης. Οι πρώτοι υπολογιστές προγραμματίζονταν σε γλώσσα μηχανής με τη χρήση διακοπτών, των οποίων η μία θέση αντιπροσώπευε το **0 (μηδέν)** και η άλλη το **1 (ένα)**.

7.2.2. Συμβολικές Γλώσσες

Όπως αναφέραμε, ο προγραμματισμός σε γλώσσα μηχανής ήταν και είναι μία πολύ δύσκολη δουλειά για τον προγραμματιστή. Έτσι, πολύ σύντομα οι προγραμματιστές άρχισαν να αντιστοιχίζουν τις διάφορες **εντολές** του κώδικα μηχανής με **συντομογραφίες λέξεων της Αγγλικής** γλώσσας. Ταυτόχρονα εκτός από τα μνημονικά ονόματα σε εντολές, χρησιμοποιήθηκαν και **μνημονικά ονόματα σε διευθύνσεις** μνήμης και έτσι μειώθηκαν οι πιθανότητες λάθους από απροσεξία, τόσο στον καθορισμό των εντολών, όσο και στον καθορισμό των διευθύνσεων που χειριζόταν το πρόγραμμα.

Έτσι δημιουργήθηκε μία άλλη κατηγορία γλωσσών, οι συμβολικές γλώσσες, οι οποίες, επειδή είναι στενά συνδεδεμένες με την αρχιτεκτονική της μηχανής, ονομάστηκαν και ***γλώσσες χαμηλού επιπέδου***. Επειδή όμως, ο υπολογιστής αναγνωρίζει μόνο τη γλώσσα μηχανής, έπρεπε και τα προγράμματα που γράφονταν σε συμβολική γλώσσα, να μεταφραστούν σε γλώσσα μηχανής. Αρχικά αυτό γινόταν χειρογραφικά από ανθρώπους που έπαιρναν ένα πρόγραμμα γραμμένο σε συμβολική γλώσσα και το μετέφραζαν σε γλώσσα μηχανής. Οι **άνθρωποι** αυτοί ονομάσθηκαν **assemblers, (συναρμολογητές)**.

Γύρω στα 1950 διαπιστώθηκε ότι η μηχανική εργασία της μετάφρασης θα μπορούσε να γίνει από τους υπολογιστές πιο γρήγορα και με περισσότερη ακρίβεια απ' ό,τι γινόταν από τον άνθρωπο. Έτσι γράφτηκαν τα πρώτα ***μεταφραστικά προγράμματα που μετέτρεπαν ένα πρόγραμμα από συμβολική γλώσσα σε γλώσσα μηχανής***.

Τα προγράμματα σε **συμβολική (assembly)** γλώσσα είναι άμεσα συνδεδεμένα με τον υπολογιστή για τον οποίο γράφτηκαν και δεν μπορούν να μεταφερθούν σε διαφορετικό υπολογιστή. Η κύρια χρήση της γλώσσας χαμηλού επιπέδου προορίζεται για τη συγγραφή προγραμμάτων διαχείρισης

του συστήματος, όπως, λειτουργικά συστήματα, **βοηθητικά προγράμματα (utilities)**, συστήματα αυτόματου ελέγχου με υπολογιστή κλπ.

7.2.3. Γλώσσες Υψηλού Επιπέδου

Η χρήση της συμβολικής γλώσσας διευκόλυνε τους προγραμματιστές στην κατασκευή και στη συντήρηση των προγραμμάτων. Τα προγράμματα όμως σε συμβολική γλώσσα ήταν **εξαρτημένα από τον υπολογιστή**. Η ανάγκη να γράφει κανείς ένα πρόγραμμα, χωρίς να χρειάζεται να έχει γνώση της αρχιτεκτονικής του υπολογιστή, δηλαδή να γράφει σχεδόν ανεξάρτητα από τον τύπο του υπολογιστή στον οποίο θα υλοποιηθεί η εφαρμογή, δημιούργησε τις γλώσσες υψηλού επιπέδου.

Οι γλώσσες υψηλού επιπέδου βασίζονται σε λέξεις-κλειδιά της Αγγλικής γλώσσας και στηρίζονται σε μια γραμματική και σε ένα συντακτικό που καθορίζονται κατά τη φάση του σχεδιασμού της γλώσσας. Ο συνδυασμός των προκαθορισμένων λέξεων της γλώσσας και των λέξεων που παράγουμε εμείς, χρησιμοποιώντας κάποιο αλφάβητο -συνήθως το Λατινικό μαζί με ειδικούς χαρακτήρες- μας δίνει προτάσεις που υπακούουν στο συντακτικό και στη γραμματική της γλώσσας και αποτελούν τις εντολές του προγράμματος.

Χαρακτηριστικά γλωσσών υψηλού επιπέδου.

- Έχουν ένα καθορισμένο σύνολο από λέξεις, σύμβολα και προτάσεις.
- Οι εντολές που γράφονται σε γλώσσα υψηλού επιπέδου μεταφράζονται κατά τη διάρκεια της μετάφρασης σε πολλές εντολές του κώδικα μηχανής.
- Έχουν ορισμένους γραμματικούς και συντακτικούς κανόνες που πρέπει να γνωρίζει ο προγραμματιστής, όταν γράφει το πρόγραμμα.
- Η εξάρτηση της γλώσσας από τη μηχανή είναι ελάχιστη, πολλές φορές μάλιστα η γλώσσα είναι ανεξάρτητη της μηχανής.
- Η γλώσσα, συνήθως, είναι εφοδιασμένη με ένα μεγάλο αριθμό υποπρογραμμάτων που ονομάζονται **βιβλιοθήκη της γλώσσας**. Από τη βιβλιοθήκη αυτή ο προγραμματιστής έχει τη δυνατότητα να ενσωματώσει ένα ή περισσότερα υποπρογράμματα μέσα στα προγράμματα εφαρμογών. Επίσης, μπορεί να προσθέσει δικά του υποπρογράμματα στη βιβλιοθήκη αυτή.

Στον πίνακα που ακολουθεί κατατάσσονται οι γλώσσες υψηλού επιπέδου ανάλογα με τις λειτουργικές τους ιδιότητες. Ενδεικτικά αναφέρονται μερικές από τις πιο γνωστές γλώσσες προγραμματισμού υψηλού επιπέδου.

- επιστημονικές εφαρμογές, όπως η ***FORTRAN***, η **C** και η ***APL***.
- εμπορικές εφαρμογές, όπως η. ***COBOL*** και η ***RPG***.
- εκπαιδευτικές εφαρμογές όπως η ***BASIC*** και η ***LOGO***.
- ειδικές εφαρμογές, όπως η ***LISP*** και η ***PROLOG***.
- επιστημονικές - εμπορικές εφαρμογές, όπως η ***PL-1***, η ***PASCAL*** κλπ.

7.2.4. Γλώσσες 4ης γενιάς

Οι γλώσσες 4ης γενιάς δίνουν στο χρήστη ένα νέο δομημένο τρόπο για την ανάπτυξη Πληροφοριακών Συστημάτων. Είναι γλώσσες υψηλής παραγωγικότητας, γιατί με τη χρήση τους απαιτείται λιγότερος χρόνος για την ανάπτυξη των εφαρμογών απ' ό,τι με τις κλασικές γλώσσες προγραμματισμού. Επιπλέον η χρήση τους είναι δυνατή από άτομα που δεν έχουν γνώσεις προγραμματισμού, αλλά απλώς διαθέτουν μία εξοικείωση με τη λειτουργία του υπολογιστή.

Οι γλώσσες 4ης γενιάς έχουν μία κατηγορία ***δυναμικών εντολών***, που δίνουν στο χρήστη τη δυνατότητα εξαιρετική ευκολία στο χρήστη, για να δημιουργεί αρχεία, να τα ενημερώνει, να σχεδιάζει οθόνες για την εκμετάλλευση τους και να δημιουργεί εκτυπώσεις. Τα προηγούμενα επιτυγχάνονται με τη χρήση πολύ λίγων εντολών, η κάθε μία από τις οποίες θα αντιστοιχούσε σε ολόκληρο πρόγραμμα σε μία κλασσική γλώσσα προγραμματισμού.

Στις γλώσσες 4ης γενιάς ο χρήστης καθορίζει το είδος της επεξεργασίας, που απαιτεί η εφαρμογή, και όχι τον τρόπο εκτέλεσής της από τον υπολογιστή. Οι γλώσσες 4ης γενιάς είναι **διαλογικές (interactive)** και έχουν τη δυνατότητα εμφάνισης βοηθητικών μηνυμάτων που έχουν σχέση με τη χρήση των διαφόρων εντολών της γλώσσας από το χρήστη. Οι γλώσσες αυτές συνήθως αποτελούν εργαλεία πακέτων για Συστήματα Διοίκησης Βάσεων Δεδομένων, όπως πχ. INGRES, ORACLE κλπ.

Η υλοποίηση ενός πληροφοριακού συστήματος είναι ταχύτερη με τη χρήση γλωσσών 4ης γενιάς. Οι εντολές των γλωσσών 4ης γενιάς είναι λιγότερες, αλλά περισσότερο περιεκτικές από τις αντίστοιχες μίας συμβατικής γλώσσας προγραμματισμού (COBOL, FORTRAN κλπ.). Αυτό έχει ως αποτέλεσμα μικρότερα προγράμματα, πράγμα το οποίο σημαίνει λιγότερα λάθη, καθώς επίσης και μικρότερο χρόνο για την ανεύρεσή τους και για την μελλοντική συντήρησή τους.

7.3. ΕΙΔΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Στη διαρκώς επιταχυνόμενη εξέλιξη υλικού και λογισμικού εντάσσεται και η ανάπτυξη του προγραμματισμού. Οι απαιτήσεις για αξιοπιστία, ποιότητα και χαμηλό κόστος στις προσφερόμενες υπηρεσίες λογισμικού, είχαν ως αποτέλεσμα τη συστηματοποίηση του προγραμματισμού. Τις τελευταίες δεκαετίες δημιουργούνται συνεχώς νέες μέθοδοι, τεχνικές και πρότυπα στον προγραμματισμό, καθώς και νέες εξειδικευμένες γλώσσες προγραμματισμού.

7.3.1. Διαδικασιακός Προγραμματισμός

Στο **Διαδικασιακό Προγραμματισμό (Procedural Programming)** το πρόγραμμα είναι γραμμένο σε κάποια γλώσσα προγραμματισμού εξαρτημένη από τη **διαδικασία (procedure oriented)**. Η διαδικασία είναι το χαρακτηριστικό των περισσοτέρων γλωσσών της τρίτης γενιάς. Μια γλώσσα που είναι εξαρτημένη από τη διαδικασία, δίνει έμφαση στον προγραμματισμό των υπολογιστικών και λογικών διαδικασιών οι οποίες απαιτούνται για την επίλυση του προβλήματος. Χαρακτηριστικά παραδείγματα γλωσσών που είναι εξαρτημένες από τη διαδικασία είναι: COBOL, FORTRAN, PASCAL κλπ.

Προβλήματα που επιλύονται αλγοριθμικά, προσφέρονται για Διαδικασιακό Προγραμματισμό. Παραδείγματα τέτοιων προγραμμάτων είναι τα προγράμματα λογιστικών εφαρμογών, εμπορικών εφαρμογών, διοικητικών εφαρμογών (προσωπικού, μισθοδοσίας), μεταφορών (κράτηση θέσεων, έκδοση εισιτηρίων) κλπ.

Στο διαδικασιακό προγραμματισμό το πρόγραμμα είναι μια διαδικασία η οποία εκτελεί ένα προς ένα τα βήματα του αλγόριθμου επίλυσης του προβλήματος. Οι εντολές του προγράμματος εκτελούνται διαδοχικά, εκτός αν υπάρχουν συνθήκες, οπότε κάποιες εντολές παραλείπονται και δημιουργούνται διακλαδώσεις ή κάποιες εντολές επαναλαμβάνονται. Ανάλογα φαινόμενα επικρατούν για παράδειγμα κατά την κίνηση των οχημάτων όπου σύμφωνα με τις κυκλοφοριακές συνθήκες εκτελούνται διακλαδώσεις και επαναλήψεις δρομολογίων κατά ορισμένα χρονικά διαστήματα.

Στην περίπτωση που υπάρχουν εντολές επανάληψης κάποιο τμήμα εντολών του προγράμματος επαναλαμβάνεται σύμφωνα με έναν προκαθορισμένο αριθμό επαναλήψεων ή ανάλογα με το αποτέλεσμα κάποιας συνθήκης.

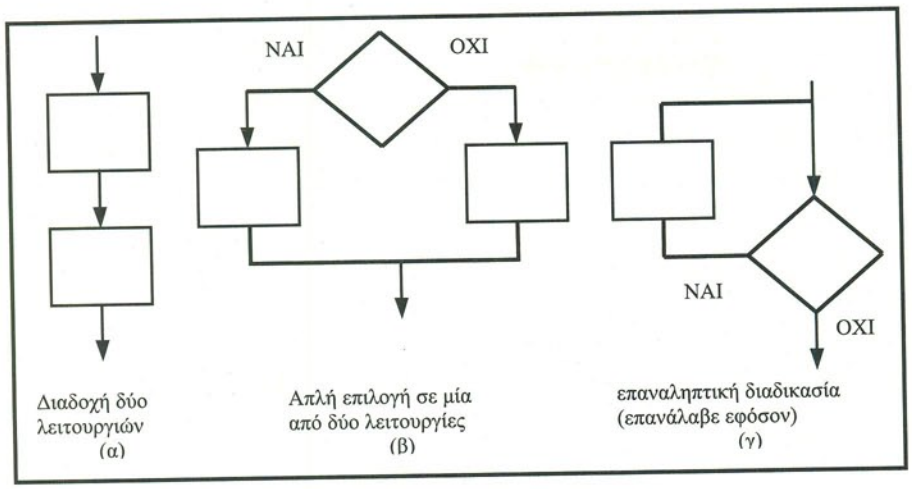
Οι γλώσσες της τρίτης γενιάς κατάφεραν να απομακρύνουν τον προγραμματισμό από το επίπεδο της μηχανής προς το επίπεδο του ανθρώπου. Το αποτέλεσμα ήταν να γίνει ο προγραμματισμός μια ευχάριστη και δημιουργική εργασία, στην οποία οι προγραμματιστές απελευθερωμένοι από δουλειές ρουτίνας πέτυχαν να δημιουργήσουν νέα εργαλεία και τεχνικές και η εξέλιξη της Πληροφορικής να συνεχίζεται από τότε με ιλιγγιώδεις ρυθμούς.

7.3.2. Δομημένος Προγραμματισμός

Ο **Δομημένος Προγραμματισμός (Structural Programming)** προϋποθέτει **δομημένη σχεδίαση** και **έλεγχο** ενός δομημένου προγράμματος, που αποτελείται από **ανεξάρτητα τμήματα (modules)**, με βάση ένα προκαθορισμένο σχέδιο.

Χρησιμοποιεί μία θεωρητική βάση για την κωδικοποίηση των προγραμμάτων, όπου η κύρια ιδέα είναι η χρησιμοποίηση των **βασικών αλγοριθμικών δομών** για τη δημιουργία πολύπλοκων προγραμμάτων. Οι δομές αυτές είναι η διαδοχή, η απλή επιλογή και η επανάληψη.

Εκτός από τις δομές αυτές χρησιμοποιούνται και τεχνικές ανάπτυξης και σχεδίασης όπως του ιεραρχικού και του τμηματικού προγραμματισμού.



Σχήμα **7-1** *Παράσταση βασικών δομών*

Η παράσταση αυτών των τεχνικών γίνεται με τα διαγράμματα **HIPO**. Στα διαγράμματα αυτά εμφανίζονται τα διάφορα επίπεδα ιεραρχίας και τα τμήματα προγράμματος με αριθμό σύμφωνα με τη σειρά εκτέλεσής τους.

Στο σημείο αυτό επισημαίνεται ότι το πρόγραμμα είναι η αναπαράσταση του αλγόριθμου στον υπολογιστή. Οι ίδιες τεχνικές του Δομημένου Προγραμματισμού εφαρμόζονται και στην ανάπτυξη και σχεδίαση αλγορίθμων όπου έχουν αναπτυχθεί αναλυτικά στην 2η ενότητα.

7.3.3. Παράλληλος Προγραμματισμός

Όπως στις περισσότερες περιπτώσεις εξέλιξης του λογισμικού, ο **Παράλληλος Προγραμματισμός** (**Parallel Programming**) οφείλει την καθιέρωσή του στην εξέλιξη του υλικού. Η εμφάνιση της αρχιτεκτονικής των πολλών επεξεργαστών οι οποίοι χρησιμοποιούν κοινή μνήμη είχε ως συνέπεια την ανάπτυξη των παραλλήλων αλγορίθμων οι οποίοι επέβαλαν την καθιέρωση του Παράλληλου Προγραμματισμού. Ο Παράλληλος Προγραμματισμός εκτός από τις δομές του παραδοσιακού διαδικασιακού προγραμματισμού διαθέτει δομές που επιτρέπουν την ταυτόχρονη εκτέλεση διαδικασιών από διαφορετικούς επεξεργαστές. Έτσι δίνεται η δυνατότητα ώστε διάφορα υποπρογράμματα ενός προγράμματος να εκτελούνται παράλληλα (ταυτόχρονα) από δύο ή περισσότερους επεξεργαστές του υπολογιστή.

Η ανάγκη της ταυτόχρονης εκτέλεσης καθώς και της επικοινωνίας μεταξύ των εκτελουμένων διεργασιών καθορίζει κατ τα πλαίσια των νέων απαιτήσεων στο χώρο του προγραμματισμού. Για την κάλυψη των νέων αναγκών έγινε επέκταση είτε σε κάποιες κλασικές γλώσσες διαδικασιακού προγραμματισμού όπως οι Ada, Modula 2 Concurrent C είτε στο σχεδιασμό νέων γλωσσών. Ταυτόχρονα δόθηκε έμφαση στον παράλληλο προγραμματισμό για πιο αποτελεσματική εκμετάλλευση των νέων δυνατοτήτων στα πλαίσια της αρχιτεκτονικής των παραλλήλων επεξεργαστών. Αντιπροσωπευτική γλώσσα σχεδιασμένη για παράλληλη επεξεργασία είναι η γλώσσα Occam.

7.3.4. Αντικειμενοστρεφής Προγραμματισμός

Ο **Αντικειμενοστρεφής Προγραμματισμός (Object Oriented Programing)** είναι μια τεχνική στην οποία υπάρχει ενσωμάτωση των δεδομένων και του τρόπου χειρισμού αυτών μέσα από την έννοια του αντικειμένου.

Ένα αντικείμενο αποτελείται από μια σειρά δεδομένων που αποτελούν τα χαρακτηριστικά του και μία σειρά μεθόδων ή ενεργειών που σχετίζονται με την επεξεργασία των δεδομένων και καθορίζουν την συμπεριφορά του αντικειμένου στο πρόγραμμα. Οι μέθοδοι χειρισμού των δεδομένων μπορεί να είναι διαδικασίες ή συναρτήσεις του χρήστη στο κυρίως πρόγραμμα.

Τα αντικείμενα σε ένα αντικειμενοστρεφές πρόγραμμα μπορεί να σχηματίζουν κλάσεις ιεραρχικά δομημένες. Με την ιεραρχική δόμηση των κλάσεων οι υποκλάσεις κληρονομούν χαρακτηριστικά και ιδιότητες των κλάσεων από τις οποίες προήλθαν, έχοντας επιπλέον την δυνατότητα να προστεθούν σε αυτές (τις υποκλάσεις) νέες ιδιότητες και χαρακτηριστικά.

7.3.5. Συναρτησιακός Προγραμματισμός

Στον **Συναρτησιακό Προγραμματισμό (Functional Programming)** οι εντολές και οι δομές ελέγχου είναι συναρτήσεις. Ως ορίσματα των συναρτήσεων μπορεί να είναι δεδομένα ή άλλες συναρτήσεις. Έτσι ο συναρτησιακός προγραμματισμός βασίζεται στην έννοια της συνάρτησης όπως την έχουμε γνωρίσει στα Μαθηματικά. Η συνάρτηση δίνει τη δυνατότητα απεικόνισης του πεδίου ορισμού σ' ένα πεδίο τιμών.

Μέσα από τις συναρτήσεις στις οποίες εφαρμόζονται δεδομένα, υλοποιούνται οι διάφορες δομές όπως η διακλάδωση, η επανάληψη κλπ. Οι έννοιες της μεταβλητής και της εκχώρησης του διαδικασιακού προγραμματισμού αντικαθίστανται από την έννοια της δέσμευσης ονομάτων σε τιμές κατά την εφαρμογή των συναρτήσεων. Οι ίδιες οι συναρτήσεις μπορεί να αντιμετωπίζονται από το σύστημα ως δεδομένα και επομένως να εμφανίζονται ως είσοδος ή έξοδος άλλων συναρτήσεων.

Το θεωρητικό υπόβαθρο πολλών γλωσσών προγραμματισμού, όπως η Lisp, αποτελεί ο **Λογισμός Λάμδα (Lamda Calculus)** ο οποίος συχνά χρησιμοποιείται για να συνδέσει θέματα από τη Λογική και την Πληροφορική. Άλλες γνωστές γλώσσες προγραμματισμού είναι η FORTH και η LOGO με πολλές εφαρμογές στο χώρο της εκπαίδευσης.

7.3.6. Λογικός Προγραμματισμός

Ο **Λογικός Προγραμματισμός (Programming in Logic)** διαφέρει από τον διαδιακασιακό προγραμματισμό ως προς το εξής: Στο πρόγραμμα δεν περιγράφεται ο αλγόριθμος επίλυσης του σχετικού προβλήματος, αλλά η γνώση

που σχετίζεται με το πρόβλημα. Έτσι ο Λογικός Προγραμματισμός, σε αντίθεση με το διαδικασιακό, στηρίζεται στη γνώση. Η γνώση περιγράφεται υπό μορφή γεγονότων και κανόνων ενώ ορίζεται μία πρόταση - στόχος προς απόδειξη. Για την απόδειξη αυτής της πρότασης-στόχου, το πρόγραμμα εφαρμόζει τους κανόνες πάνω στα γεγονότα.

Ο Λογικός Προγραμματισμός εμφανίστηκε στα τέλη της δεκαετίας του 1970. Υπήρξε το αντικείμενο ερευνών σε περιοχές της επιστήμης, όπως η αυτόματη απόδειξη θεωρημάτων και η Τεχνητή Νοημοσύνη. Αποτέλεσμα των ερευνών αυτών ήταν και η δημιουργία κατάλληλων γλωσσών λογικού προγραμματισμού με τυπική εκπρόσωπο την Prolog.

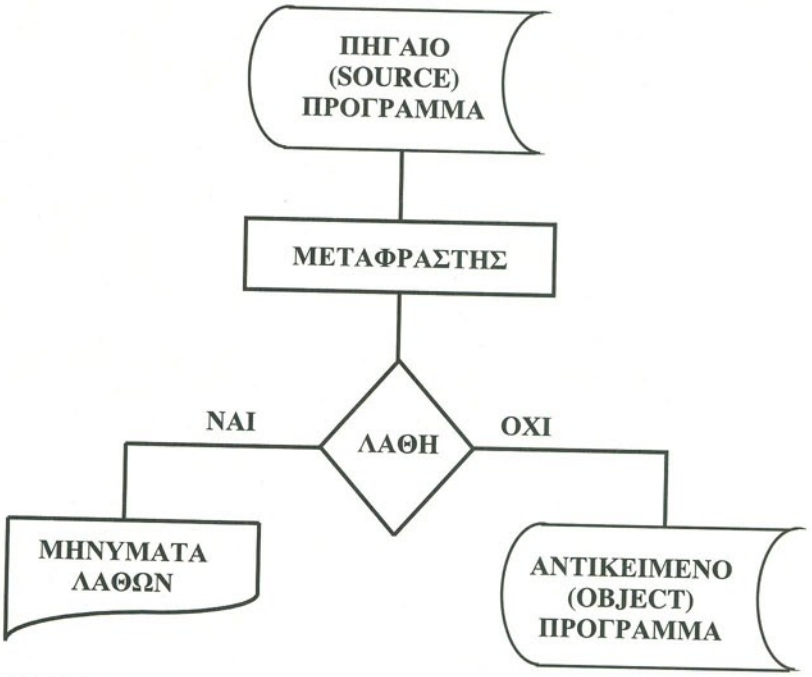
Στο Διαδικασιακό Προγραμματισμό ο προγραμματιστής προσδιορίζει τα βήματα του αλγόριθμου για την επίλυση κάποιου προβλήματος. Στο Λογικό Προγραμματισμό ο προγραμματιστής περιγράφει τη γνώση του σχετικά με κάποιο θέμα και προσδιορίζει τους στόχους. Ένα πρόγραμμα σύμφωνα με το Λογικό Προγραμματισμό περιλαμβάνει ένα σύνολο γνωστών γεγονότων και κανόνων και μία πρόταση-στόχο η οποία πρέπει να αποδειχθεί αληθής με τη χρήση των κανόνων. Η εφαρμογή των κανόνων για την απόδειξη του στόχου δεν αποτελεί μέρος του προγραμματισμού αλλά εκτελείται αυτόματα.

7.4. ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

Όπως έχει αναφερθεί ένα πρόγραμμα κωδικοποιείται από τον προγραμματιστή σε μια γλώσσα προγραμματισμού, ενώ ο υπολογιστής μπορεί να εκτελέσει ένα πρόγραμμα τότε μόνον, εάν αυτό έχει γραφεί σε κώδικα (γλώσσα) μηχανής. Για τη μετατροπή ενός προγράμματος από μια γλώσσα προγραμματισμού σε γλώσσα μηχανής χρησιμοποιείται κάποιο από τα μεταφραστικά προγράμματα που περιγράφεται παρακάτω. Στη συνέχεια επίσης θα περιγραφούν και οι φάσεις που περνάει ένα πρόγραμμα γραμμένο σε μια γλώσσα προγραμματισμού μέχρι να εκτελεστεί.

7.4.1. Μεταγλωττιστής

Ο μεταγλωττιστής είναι ένα ειδικό πρόγραμμα που μεταφράζει ένα άλλο πρόγραμμα, από μία γλώσσα προγραμματισμού σε γλώσσα μηχανής. Το πρόγραμμα προς μετάφραση, ονομάζεται **πηγαίο** (***source***) πρόγραμμα, ενώ το μεταφρασμένο **αντικείμενο** (***object***) πρόγραμμα.



Σχήμα 7-2 Μεταφραστικό πρόγραμμα

Οι μεταγλωττιστές διακρίνονται σε:

COMPILER (Μεταγλωττιστής γλώσσας υψηλού επιπέδου).

INTERPRETER (Διερμηνέας. δηλ. Μεταγλωττιστής γλώσσας υψηλού επιπέδου με ταυτόχρονη εκτέλεση του προγράμματος).

COMPILER

Ο **Compiler** διαβάζει ένα πρόγραμμα γραμμένο σε μια γλώσσα **υψηλού επιπέδου** (π.χ. FORTRAN, COBOL κλπ.) και το μεταφράζει σε **κώδικα μηχανής** κάποιου υπολογιστή. Ο μεταγλωττιστής ελέγχει αρχικά το πρόγραμμα για ορθογραφικά και συντακτικά λάθη. Εφόσον υπάρχουν λάθη, διορθώνονται με ένα άλλο πρόγραμμα που ονομάζεται συντάκτης-διορθωτής κειμένου (editor). Η διαδικασία ελέγχου-διόρθωσης επαναλαμβάνεται όσες φορές απαιτηθεί μέχρι να μηδενισθούν τα λάθη. Μόνο τότε ο μεταγλωττιστής θα μας δώσει τον αντικείμενο κώδικα (κώδικα σε γλώσσα μηχανής). Ο **αντικείμενος κώδικας (object code)** που παράγεται από το μεταγλωττιστή δεν είναι εκτελέσιμος από τον υπολογιστή και γι αυτό περνάει από κάποιες άλλες διαδικασίες μέχρι να γίνει εκτελέσιμος όπως θα εξηγήσουμε αναλυτικά σε επόμενη παράγραφο. Στον πίνακα που ακολουθεί συνοψίζουμε τις εργασίες που εκτελεί ο μεταγλωττιστής.

- Κάνει συντακτική και γραμματική ανάλυση του πηγαίου προγράμματος με εκτύπωση των λαθών (diagnostics).
- Κάνει την κατάληψη του απαραίτητου χώρου στην κύρια μνήμη για να μπορεί να εκτελεστεί το πρόγραμμα.
- Φυλάσσει το αντικείμενο (object) πρόγραμμα στο δίσκο.
- Παράγει, κατά βούληση του χρήστη, ένα αρχείο όπου υπάρχει η ανάπτυξη του προγράμματος σε κώδικα μηχανής (16-δική μορφή) και η ανάπτυξη κάθε εντολής του πηγαίου προγράμματος σε γλώσσα χαμηλού επιπέδου (assembly).
- Παράγει πίνακες στη μνήμη (πίνακες σταθερών, βρόχων κλπ.) που είναι απαραίτητοι για τη μετάφραση του προγράμματος.

INTERPRETER

Ο **Interpreter** είναι ένας μεταγλωττιστής που ελέγχει συντακτικά και γραμματικά το πηγαίο πρόγραμμα εξετάζοντας τη μια εντολή μετά την άλλη, κατά τη σειρά εκτέλεσής τους. Αν δεν υπάρχουν λάθη στην εντολή, παράγει την αντίστοιχη εντολή σε κώδικα μηχανής, την εκτελεί και συνεχίζει με τη μετάφραση της επόμενης εντολής.

Ο interpreter διαφέρει από τον compiler στα εξής χαρακτηριστικά:

- Δεν ελέγχει τη σύνταξη ολόκληρου του πηγαίου προγράμματος πριν από την εκτέλεση.
- Επαναλαμβάνει τη διαδικασία μετάφρασης - ελέγχου κάθε φορά που εκτελείται το πηγαίο πρόγραμμα. Επιπλέον, γίνεται συντακτικός έλεγχος και μετάφραση σε μία εντολή κάθε φορά που συναντάται, κατά την εκτέλεση ενός προγράμματος
- Ο έλεγχος και η διόρθωση των λαθών γίνονται ευκολότερα, γιατί ο προγραμματιστής πληροφορείται αμέσως για κάθε συντακτικό ή γραμματικό λάθος, το διορθώνει και συνεχίζει την εκτέλεση του προγράμματος.
- Η εκτέλεση ενός προγράμματος με Interpreter είναι πολύ πιο αργή σε σύγκριση με την εκτέλεση του ίδιου προγράμματος, μεταφρασμένου με compiler.

Η χρήση Interpreter είναι γνωστή από τη γλώσσα BASIC. Η γλώσσα αυτή έγινε γρήγορα δημοφιλής στο χώρο των μικροϋπολογιστών, κυρίως λόγω της ευκολίας προγραμματισμού, η οποία οφείλεται στον Διερμηνέα και στο **διαλεκτικό (interactive)** περιβάλλον που αυτός δημιουργεί.

7.4.2. Γραφή Εκτέλεση Προγράμματος

Το πηγαίο πρόγραμμα γράφεται σε κάποια γλώσσα προγραμματισμού με ένα απλό πρόγραμμα επεξεργασίας κειμένου, τον εκδότη-διορθωτή (**editor**) κειμένων. Το κείμενο προγράμματος το οποίο παράγεται ονομάζεται **πηγαίος κώδικας (source code)**.

Στη συνέχεια, ο Μεταγλωττιστής ελέγχει το πηγαίο πρόγραμμα για ορθογραφικά ή συντακτικά λάθη. Εφόσον υπάρχουν λάθη ο Μεταγλωττιστής δίνει αντίστοιχα μηνύματα. και ο εκδότης– διορθωτής (editor) χρησιμοποιείται για τη διόρθωσή τους. Η διαδικασία της διόρθωσης επαναλαμβάνεται όσες φορές χρειαστεί μέχρι να μηδενιστούν τα λάθη. Μόνο όταν δεν υπάρχουν λάθη, ο μεταγλωττιστής παράγει το αντικείμενο πρόγραμμα.

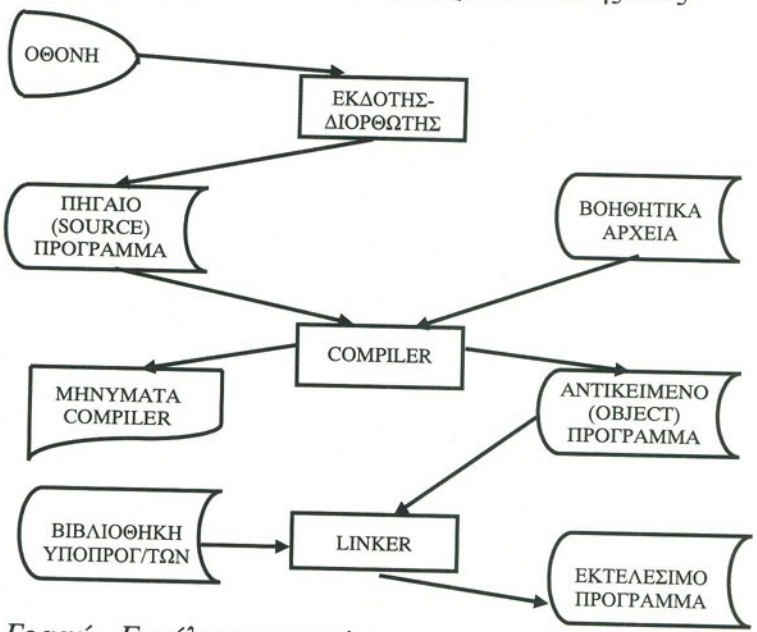
Πρέπει να σημειωθεί ότι το αντικείμενο (object) πρόγραμμα που παράγει ένας μεταγλωττιστής, εάν το πηγαίο πρόγραμμα δεν έχει λάθη, δεν είναι κατ' ανάγκην εκτελέσιμο. Πολλές φορές είναι απαραίτητο, ανάλογα με τον μεταγλωττιστή, το αντικείμενο πρόγραμμα να συνδεθεί με υποπρογράμματα της βιβλιοθήκης της γλώσσας ή του προγραμματιστή. Αυτό επιτυγχάνεται με τη βοήθεια ενός άλλου προγράμματος του συστήματος, του **συνδέτη** (**LINKER**). Μετά τη σύνδεση αυτή παίρνουμε τον κώδικα που είναι εκτελέσιμος από τον υπολογιστή. Έτσι έχουμε τη διαδικασία:

γραφή – μεταγλώττιση – σύνδεση –εκτέλεση Μερικές φορές αντί του συνδέτη χρησιμοποιείται ένα άλλο πρόγραμμα, ο **φορτωτής** (**loader**). Ο φορτωτής συνδέει το αντικείμενο πρόγραμμα με υποπρογράμματα της βιβλιοθήκης της γλώσσας ή του προγραμματιστή και φορτώνει στη μνήμη για εκτέλεση τον κώδικα μηχανής που προκύπτει. Στην περίπτωση αυτή έχουμε τη διαδικασία:

γραφή – μεταγλώττιση – φόρτωση

Στο σχεδιάγραμμα που ακολουθεί φαίνονται τα διάφορα στάδια από τα οποία περνάει ένα πρόγραμμα από τη γραφή μέχρι την εκτέλεσή του.

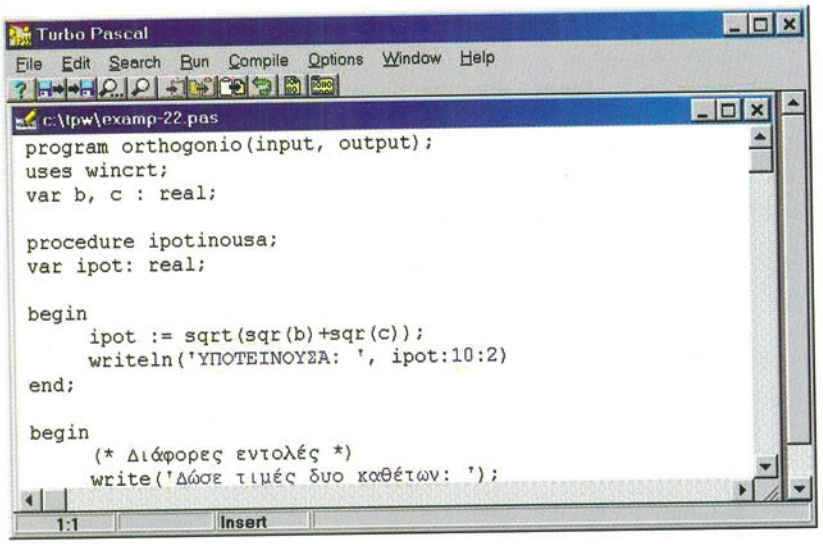
Στην περίπτωση που το μεταφραστικό πρόγραμμα είναι διερμηνέας, η μεταγλώττιση, η σύνδεση και η εκτέλεση γίνονται εντολή (όχι ολόκληρο το πηγαίο πρόγραμμα) προς εντολή κατά τη σειρά εκτέλεσής τους.



Σχήμα 7-3 Γραφή - Εκτέλεση προγράμματος

7.4.3. Σύγχρονα Προγραμματιστικά Περιβάλλοντα

Σήμερα υπάρχουν πολλές δυνατότητες για την ανάπτυξη προγραμμάτων. Θα αναφέρουμε συνοπτικά τα ολοκληρωμένα και τα γραφικά περιβάλλοντα. Σ' ένα **Ολοκληρωμένο Περιβάλλον**, για την ανάπτυξη ενός προγράμματος συνυπάρχουν διάφορα από τα προγράμματα που ανάπτυξης που αναφέρθηκαν παραπάνω. Τα προγράμματα αυτά συνεργάζονται στο ίδιο περιβάλλον, πράγμα το οποίο διευκολύνει και επιταχύνει την ολοκλήρωση της ανάπτυξης του προγράμματος καθώς και τη μετάβαση στα διάφορα στάδια.



Σχήμα 7-4 Ολοκληρωμένο περιβάλλον της Turbo Pascal

Στο προηγούμενο σχήμα εμφανίζονται μερικά από προγράμματα που διαθέτει το ολοκληρωμένο περιβάλλον της Pascal, όπως ο **διορθωτής (editor)**, ο **μεταγλωττιστής (compiler)**, ο **εκσφαλματωτής (debugger)** και άλλα, τα οποία μπορούμε να δούμε επιλέγοντας από τα μενού. Τα μενού εκτός από τα προγράμματα περιέχουν και διάφορα άλλα εργαλεία χρήσιμα για την ανάπτυξη του προγράμματος. Παρόμοια εικόνα εμφανίζει και το ολοκληρωμένο περιβάλλον της Qbasic.

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης προγραμμάτων είναι δυνατόν να διαθέτει εργαλεία σχεδίασης με τη βοήθεια γραφικών. Ένα περιβάλλον με αυτές τις ευκολίες χαρακτηρίζεται ως **γραφικό περιβάλλον**. Τα εργαλεία γραφικών μπορεί να χρησιμεύσουν για να σχεδιάσει ο προγραμματιστής στην οθόνη την εφαρμογή του.

Ανακεφαλαίωση

Στο κεφάλαιο αυτό είδαμε τα διάφορα στάδια στην ανάπτυξης προγράμματος, τις κατηγορίες γλωσσών προγραμματισμού και τα προγραμματιστικά περιβάλλοντα που χρησιμοποιούνται για την ανάπτυξη των προγραμμάτων.

Γνωρίσαμε τα διάφορα μεταφραστικά προγράμματα τα οποία μετατρέπουν τον κώδικα του προγράμματος σε γλώσσα μηχανής. Γνωρίσαμε τις διάφορες κατηγορίες προγραμματισμού και τις ιδιότητές τους.

Ερωτήσεις

1. Να αναφέρετε διαφορές μεταξύ γλωσσών μηχανής και γλωσσών χαμηλού επιπέδου.
2. Να αναφέρετε διαφορές μεταξύ γλωσσών υψηλού επιπέδου και γλωσσών χαμηλού επιπέδου.
3. Πώς μπορούμε να χαρακτηρίσουμε τις γλώσσες υψηλού επιπέδου ανάλογα με τις λειτουργικές τους ιδιότητες;
4. Τί γνωρίζετε για τις γλώσσες τέταρτης γενιάς;
5. Τί γνωρίζετε για το διαδικασιακό προγραμματισμό;
6. Να δώσετε μερικά παραδείγματα γλωσσών που είναι εξαρτημένες από τη διαδικασία.
7. Ποιές είναι οι βασικές αλγοριθμικές δομές και ποιες οι τεχνικές σχεδίασης στο δομημένο προγραμματισμό;
8. Πού χρησιμοποιούνται τα διαγράμματα HIPO;
9. Πού οφείλει την καθιέρωσή του ο παράλληλος προγραμματισμός;
10. Τί είναι ο αντικειμενοστρεφείς προγραμματισμός;
11. Ποιά είναι τα χαρακτηριστικά και ποιες οι μέθοδοι στον αντικειμενοστρεφή προγραμματισμό;
12. Τί είναι ο συναρτησιακός προγραμματισμός;
13. Αναφέρετε παραδείγματα γλωσσών συναρτησιακού προγραμματισμού.
14. Ποια η διαφορά του λογικού από το διαδικασιακό προγραμματισμό;
15. Να εξηγήσετε τους όρους πηγαίο πρόγραμμα, αντικείμενο πρόγραμμα και μεταγλωττιστής;
16. Να αναφέρετε ομοιότητες και διαφορές μεταξύ compiler και interpreter.
17. Να εξηγήσετε τη διαδικασία γραφής –εκτέλεσης προγράμματος.
18. Ποια είναι τα σύγχρονα προγραμματιστικά περιβάλλοντα;
19. Να συμπληρώσετε τα κενά με τη λέξη που λείπει:

    α. Κατά τη διάρκεια της επίλυσης του προβλήματος με τον υπολογιστή είναι απαραίτητο να χρησιμοποιήσουμε ορισμένες τυποποιημένες διαδικασίες που χαρακτηρίζονται ως ***κύκλος*** _____
    _____

β. Μετά την αποτύπωση του αλγορίθμου ακολουθεί η _____ του σε μία από τις γλώσσες προγραμματισμού.

γ. Οι εντολές σε γλώσσα μηχανής αποτελούνται από μία _____0 και 1 σταθερού ή _____ πλήθους.

δ. Σύντομα οι προγραμματιστές άρχισαν να αντιστοιχίζουν τις διάφορες **εντολές** του κώδικα μηχανής με _____ **της Αγγλικής** γλώσσας. Ταυτόχρονα εκτός από τα μνημονικά ονόματα σε εντολές, χρησιμοποιήθηκαν και **μνημονικά ονόματα σε** _____ μνήμης και έτσι μειώθηκαν οι πιθανότητες λάθους από απροσεξία, τόσο στον καθορισμό των εντολών, όσο και στον καθορισμό των διευθύνσεων που χειριζόταν το πρόγραμμα. Έτσι, δημιουργήθηκε μία άλλη κατηγορία γλωσσών, οι _____ γλώσσες, οι οποίες, επειδή είναι στενά συνδεδεμένες με την αρχιτεκτονική της μηχανής, ονομάστηκαν και ***γλώσσες*** _____ ***επιπέδου***.

ε. Οι γλώσσες υψηλού _____ βασίζονται σε λέξεις-κλειδιά της _____γλώσσας και στηρίζονται σε μια _____ και σε ένα _____ που καθορίζονται κατά τη φάση του σχεδιασμού της γλώσσας. Ο συνδυασμός των προκαθορισμένων λέξεων της γλώσσας και των λέξεων που παράγουμε εμείς χρησιμοποιώντας κάποιο αλφάβητο (συνήθως το Λατινικό μαζί με ειδικούς χαρακτήρες), μας δίνει _____ που υπακούουν στο συντακτικό και στη γραμματική της γλώσσας και αποτελούν τις _____ του προγράμματος.

ζ. Στις γλώσσες ____γενιάς ο χρήστης καθορίζει το είδος της επεξεργασίας που απαιτεί η εφαρμογή και όχι τον τρόπο _____ της από τον υπολογιστή. Οι γλώσσες 4ης γενιάς είναι _____ (interactive) και έχουν τη δυνατότητα εμφάνισης βοηθητικών μηνυμάτων που έχουν σχέση με τη χρήση των διαφόρων _____ της γλώσσας από το χρήστη.

20. Να συμπληρώσετε τα κενά με τη λέξη που λείπει

α. Στο διαδικασιακό προγραμματισμό το πρόγραμμα είναι γραμμένο σε κάποια γλώσσα προγραμματισμού εξαρτημένη από τη _____ (procedure oriented). Η διαδικασία είναι το χαρακτηριστικό των περισσοτέρων γλωσσών _____ _____..

β. Ο δομημένος προγραμματισμός χρησιμοποιεί για την κωδικοποίηση των προγραμμάτων τις βασικές _____ δομές για τη δημιουργία πολύπλοκων προγραμμάτων. Οι δομές αυτές είναι: η _____, η απλή _____ και η _____. Εκτός από τις δομές αυτές χρησιμοποιούνται και τεχνικές ανάπτυξης και σχεδίασης όπως του _____ και του _____ προγραμματισμού. Η παράσταση αυτών των τεχνικών γίνεται με τα διαγράμματα _____ (**Hierarchical Input Output Processing**), που στα Ελληνικά μεταφράζεται ως _____ ***εισόδου -***

_____ - ***εξόδου***. Στα διαγράμματα αυτά εμφανίζονται τα διάφορα επίπεδα ιεραρχίας και τα τμήματα προγράμματος με αριθμό σύμφωνα με τη σειρά εκτέλεσής τους.

γ. Η εμφάνιση της αρχιτεκτονικής των _____ επεξεργαστών οι οποίοι χρησιμοποιούν _____ μνήμη είχε ως συνέπεια την ανάπτυξη των _____ αλγορίθμων οι οποίοι επέβαλαν την καθιέρωση του παράλληλου _____.

δ. Ο αντικειμενοστρεφής προγραμματισμός είναι μια τεχνική στην οποία υπάρχει ενσωμάτωση των δεδομένων και του τρόπου _____ αυτών μέσα από την έννοια του _____. Ένα αντικείμενο αποτελείται από μια σειρά δεδομένων που αποτελούν τα _____ _____ του και μία σειρά μεθόδων ή ενεργειών που σχετίζονται με την _____ των δεδομένων και καθορίζουν την συμπεριφορά του _____ στο πρόγραμμα. Οι μέθοδοι χειρισμού των δεδομένων μπορεί να είναι _____ ή συναρτήσεις του χρήστη στο κυρίως πρόγραμμα.

ε. Ο μεταγλωττιστής είναι ένα ειδικό πρόγραμμα που _____ ένα άλλο πρόγραμμα, από μία γλώσσα _____ σε γλώσσα _____. Το πρόγραμμα προς μετάφραση, το λέμε _____ (***source***) πρόγραμμα, ενώ το μεταφρασμένο _____ (***object***) πρόγραμμα.

Δραστηριότητες

Να εμφανίσετε το ολοκληρωμένο περιβάλλον της **QBasic** στο περιβάλλον των Windows. Να εξετάσετε τις διάφορες επιλογές και να συγκρίνετε το ολοκληρωμένο περιβάλλον της **Qbasic** με αυτό της **Turbo Pascal για Windows**.

ΚΕΦΑΛΑΙΟ 8
Η ΓΛΩΣΣΑ PASCAL

Σκοπός κεφαλαίου:
Να γνωρίσουμε τις αρχές και τις δυνατότητες της γλώσσας Pascal.

Ειδικοί σκοποί:
- ❑ Να γνωρίσουμε τη χρησιμότητα της γλώσσας στη διδασκαλία του προγραμματισμού.
- ❑ Να γνωρίσουμε τις δυνατότητές της για την υλοποίηση του δομημένου προγραμματισμού.
- ❑ Να γνωρίσουμε τη δομή ενός προγράμματος σε γλώσσα Pascal.
- ❑ Να γνωρίσουμε τις ευκολίες που παρέχει για τη χρησιμοποίηση και τη δημιουργία προγραμμάτων.

8.1. ΕΙΣΑΓΩΓΗ

Η γλώσσα PASCAL σχεδιάστηκε από τον Nicklaus Wirth, διάσημο Ελβετό επιστήμονα της Πληροφορικής, το 1968 και αναθεωρήθηκε το 1972.Ο Wirth σχεδίασε την PASCAL προκειμένου να ξεπεραστούν τα μειονεκτήματα των γλωσσών προγραμματισμού της δεκαετίας του 1960. Πήρε το όνομά της προς τιμή του μαθηματικού και φιλοσόφου Blaise Pascal.

Η PASCAL σχεδιάστηκε με στόχο να χρησιμοποιηθεί ως διδακτικό εργαλείο των αρχών του προγραμματισμού. Λόγω όμως της πληρότητάς της, της απλότητας και της ευκολίας στην εκμάθησή της .χρησιμοποιείται ευρέως στις επιχειρήσεις, τη βιομηχανία και τους προσωπικούς υπολογιστές.

Η PASCAL είναι γλώσσα γενικής χρήσης και υποστηρίζει τις αρχές του δομημένου και του τμηματικού προγραμματισμού. Μερικά από τα ιδιαίτερα χαρακτηριστικά της είναι τα εξής:

- Η δυνατότητα που δίνεται στον προγραμματιστή να δημιουργεί δικούς του τύπους δεδομένων.
- Η χρήση μεταβλητών τύπου **δείκτη (pointer)** και η δυνατότητα της δυναμικής διαχείρισης της κεντρικής μνήμης.
- Η **σύνθετη εντολή (compound statement)**, δηλαδή η χρήση μιας σειράς εντολών ως μία εντολή.

Γλώσσα προγραμματισμού
Είναι ένα σύνολο κανόνων, συμβόλων και ειδικών λέξεων που χρησιμοποιούνται για τη δημιουργία ενός προγράμματος.

Όπως οι περισσότερες γλώσσες, έτσι και αυτή, πέρασε από πολλές εκδόσεις διαφόρων κατασκευαστών που κάθε μια εμπλουτιζόταν με περισσότερες δυνατότητες. Αρχικά είχε μία αδυναμία στον αποτελεσματικό χειρισμό των αρχείων και των αλφαριθμητικών δεδομένων ή **συμβολοσειρών (strings)** που όμως έχει ήδη ξεπεραστεί σε μεταγενέστερες εκδόσεις. Η έκδοσή της για περιβάλλον Windows θεωρείται ως πρότυπο στο χώρο των μικροϋπολογιστών.

Το **αλφάβητο** της γλώσσας Pascal, αποτελείται από βασικά σύμβολα, όπως γράμματα του Ελληνολατινικού αλφαβήτου, τα αριθμητικά ψηφία (0 - 9) και τα ειδικά σύμβολα, όπως +, -, *, /, \, ., ; κλπ.

Συντακτικό(Syntax)
Είναι ένα σύνολο τυπικών κανόνων οι οποίοι προσδιορίζουν πώς γράφονται έγκυρες εντολές σε μία γλώσσα προγραμματισμού.

Η Pascal μας επιτρέπει να δίνουμε ταυτότητες ή **ονόματα (identifiers)** τα οποία αναφέρονται σε σταθερές, μεταβλητές, τύπους δεδομένων, διαδικασίες, συναρτήσεις κλπ. Ένα **όνομα** αποτελείται από μια σειρά χαρακτήρων (**γράμματα** του λατινικού αλφαβήτου., **αριθμοί** ή _), πρέπει να αρχίζει πάντοτε με γράμμα και δεν πρέπει να περιέχει κενά.

Σημασιολογία (Semantics)
Είναι ένα σύνολο κανόνων οι οποίοι προσδιορίζουν τη σημασία των εντολών που γράφονται σε μία γλώσσα προγραμματισμού.

Προκειμένου να εξασφαλίσουμε ένα ευανάγνωστο πρόγραμμα, δίνουμε ονόματα ενδεικτικά του περιεχομένου τους (meaningful) και αντί κενού βάζουμε τον χαρακτήρα "_". Ο χαρακτήρας αυτός δεν μπορεί να είναι ο τελευταίος του ονόματος. Πχ. basikos_mistos, kratiseis, pl_poso, είναι σωστά ονόματα.

Πολλές φορές για λόγους **τεκμηρίωσης** χρειάζεται να γράφουμε **σχόλια** στο πρόγραμμα. Τα σχόλια μπορεί να καταλαμβάνουν όσες γραμμές θέλουμε ή να εμφανίζονται μεταξύ των στοιχείων μιας εντολής. Τα σχόλια περιέχονται μεταξύ δύο παρενθέσεων της μορφής **{...} ή (*...*)**. Κάθε σχόλιο θεωρείται ως ένα κενό.

8.2. ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τύπος Δεδομένων (Data Type)
Είναι μια κατηγορία δεδομένων με ορισμένη απεικόνιση και ένα σύνολο λειτουργιών που μπορούν να εφαρμοστούν στο σύνολο των τιμών τους.

Ένα πρόγραμμα επεξεργάζεται δεδομένα τα οποία μπορεί να είναι αποθηκευμένα εσωτερικά στη μνήμη ή εξωτερικά στο δίσκο ή τη δισκέτα ή να γίνεται η εισαγωγή τους από το πληκτρολόγιο ή το σαρωτή κλπ. Στην Pascal κάθε δεδομένο πρέπει να είναι ορισμένου τύπου. Οι **τύποι δεδομένων** προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή καθώς και το είδος της επεξεργασίας τους από τον υπολογιστή. Μερικοί τύποι δεδομένων χρησιμοποιούνται τόσο συχνά που η Pascal τους έχει προσδιορίσει και έτσι υπάρχουν έτοιμοι για το χρήστη. Στον προγραμματισμό όμως, ο προγραμματιστής προσδιορίζει και άλλους τύπους δεδομένων ανάλογα με το είδος της επεξεργασίας.

Οι τύποι δεδομένων χρησιμοποιούνται για τον ορισμό των μεταβλητών ή των συναρτήσεων που ορίζει ο χρήστης. Μία μεταβλητή θα είναι πάντοτε ενός συγκεκριμένου τύπου. Εάν προσπαθήσουμε να δώσουμε τιμή εκτός

των ορίων του τύπου της τότε τα αποτελέσματα είναι απρόβλεπτα. Υπάρχουν απλοί ή στοιχειώδεις και σύνθετοι τύποι δεδομένων.

Οι προσδιορισμένοι από την Pascal **απλοί** ή στοιχειώδεις τύποι δεδομένων είναι οι:

- Ο ακέραιος τύπος.
- Ο πραγματικός τύπος.
- Ο λογικός τύπος.
- Ο χαρακτήρας.

Σύνθετοι τύποι δεδομένων είναι αυτοί που ορίζονται από απλούς τύπους ή και από άλλους σύνθετους που ορίστηκαν παραπάνω. Στην ενότητα αυτή θα αναφερθεί ο αλφαριθμητικός τύπος (string).

8.2.1. Ακέραιος

Οι ακέραιοι τύποι είναι οι γνωστοί μας ακέραιοι που μπορεί να είναι θετικοί ή αρνητικοί, πχ. 10 100 –10009 +51 –52 κλπ. Αν δεν υπάρχει σημείο, ο αριθμός θεωρείται θετικός. Θεωρητικά ένας ακέραιος μπορεί να έχει οποιοδήποτε πλήθος ψηφίων. Πρακτικά όμως το πλήθος των ψηφίων περιορίζεται ανάλογα με τον τύπο του υπολογιστή. Η μεταβλητή **MaxInt** (μέγιστος ακέραιος) προσδιορίζει το εύρος του διαστήματος των ακεραίων, από **–MaxInt-1** έως **MaxInt** Στην Turbo Pascal MaxInt=32767 δηλαδή, το εύρος του διαστήματος των ακεραίων είναι από –32768 έως 32767.

Έτσι οι **ακέραιοι (integer)** της Pascal, όπως και στις περισσότερες γλώσσες προγραμματισμού, είναι ένα υποσύνολο των ακεραίων, που γνωρίζουμε από τα Μαθηματικά. Στις περισσότερες εκδόσεις της Pascal παίρνουν τιμές στο διάστημα [-32768, 32767]. Με τους ακεραίους γίνονται όλες οι γνωστές από τα Μαθηματικά πράξεις αλλά τα αποτελέσματα πρέπει να βρίσκονται στο σύνολο τιμών του ακέραιου τύπου.

Μερικές εκδόσεις της Pascal, όπως η Turbo Pascal, υποστηρίζουν διάφορους τύπους ακεραίων. Κάθε τέτοιος τύπος καθορίζεται από το πεδίο τιμών του, την ύπαρξη ή μη πρόσημου και το πλήθος των bytes (8bits/byte) που καταλαμβάνει στην κεντρική μνήμη (βλέπε Πίνακα 8-1).

Πίνακας 8-1
Ακέραιοι τύποι

Δήλωση τύπου	Διάστημα τιμών	πρόσημο	πλήθος bytes
shortint	-128 .. 127	ΝΑΙ	1
integer	-32768 .. 32767	ΝΑΙ	2
longint	-2148483648 .. 2147483647	ΝΑΙ	4
byte	0 .. 255	ΟΧΙ	1
word	0 .. 65535	ΟΧΙ	2

Οι επιτρεπτές πράξεις ακεραίων είναι:		Παραδείγματα
+	πρόσθεση	
-	αφαίρεση	
*	πολλαπλασιασμός	
div	ακέραια διαίρεση (πηλίκο)	27 div 6 = 4, 16 div 17 = 0 36 div 6 = 6
mod	υπόλοιπο διαίρεσης	27 mod 6 = 3 16 mod 17 = 16 36 mod 6 = 0

8.2.2. Πραγματικός

Πραγματικοί **τύποι**
- real
- single
- double
- extended
- comp

Ο **πραγματικός (real)** τύπος χρησιμοποιείται εκεί που οι αριθμητικές τιμές δεν είναι ακέραιοι αριθμοί ή οι αναμενόμενες τιμές τους είναι εκτός ορίων του ακέραιου τύπου. Επειδή οι αριθμοί αυτοί αποθηκεύονται με διαφορετικό τρόπο από τους ακέραιους, δίνουν μεγαλύτερο εύρος τιμών και μπορεί να χρησιμοποιηθούν για πολύ μεγάλους ή πολύ μικρούς αριθμούς (ακέραιους ή δεκαδικούς).

Οι πραγματικοί αριθμοί της Pascal ορίζονται ως ένα υποσύνολο των πραγματικών αριθμών, που ξέρουμε από τα Μαθηματικά. Περιέχονται στο διάστημα από 10^{-38} μέχρι 10^{+38} περίπου και έχουν από 6 μέχρι και 20 σημαντικά ψηφία, ανάλογα με την έκδοσή της Pascal και την επιμέρους επιλογή του πραγματικού τύπου. Τα σημαντικά ψηφία εκφράζονται με τη μορφή κινητής υποδιαστολής που διαθέτει ο υπολογιστής. Τα αποτελέσματα αριθμητικών πράξεων προσεγγίζονται με το πλήθος των σημαντικών ψηφίων που διαθέτει η έκδοση της Pascal.

Π.χ. ο αριθμός 4.934456E+04 αντιστοιχεί στο δεκαδικό αριθμό 49344,56 ($4.934456 \times 10^{+4}$).

Μερικές εκδόσεις Pascal όπως η Turbo Pascal υποστηρίζει διάφορους τύπους πραγματικών. Κάθε τέτοιος τύπος καθορίζεται από το πεδίο τιμών του, το πλήθος των σημαντικών ψηφίων του και το πλήθος των bytes που καταλαμβάνει στην κεντρική μνήμη (βλέπε Πίνακα 8-2).

Πίνακας 8-2

Τύποι πραγματικών αριθμών

Δήλωση τύπου	Διάστημα τιμών	πρόσημο	πλήθος bytes
real	$-2.9 \times 10^{-39} .. 1.7 \times 10^{38}$	11-12	6
single	$-1.5 \times 10^{-45} .. 3.4 \times 10^{38}$	7-8	4
double	$-5.0 \times 10^{-324} .. 1.7 \times 10^{308}$	15-16	8
extended	$-3.4 \times 10^{-4932} .. 1.1 \times 10^{4932}$	19-20	10
comp	$-263^{+1} .. 263^{-1}$	19-20	8

Επειδή η μορφή αναπαράστασης των πραγματικών αριθμών είναι περίπλοκη, πράξεις που περιλαμβάνουν πραγματικούς αριθμούς απαιτούν περισσότερο χρόνο για εκτέλεση από αυτές των ακεραίων, οι οποίοι αποθηκεύονται απλά σε ισοδύναμη δυαδική μορφή.

Οι επιτρεπτές πράξεις πραγματικών αριθμών είναι:

\+ πρόσθεση

\- αφαίρεση

* πολλαπλασιασμός

/ διαίρεση

Αριθμητικές εκφράσεις είναι οι απεικονίσεις αριθμητικών παραστάσεων που μπορεί να περιέχουν σταθερές, μεταβλητές, συναρτήσεις, αριθμητικά σύμβολα και παρενθέσεις. Κατά την εκτέλεση των πράξεων η προτεραιότητα των μαθηματικών τελεστών φαίνεται παρακάτω

Χαμηλότερη	*Υψηλότερη*
Προτεραιότητα	***Προτεραιότητα***
+ {πρόσθεση)	* (πολλαπλασιασμός)
- (αφαίρεση)	/ (διαίρεση)
	DIV (διαίρεση ακέραια)
	MOD (υπόλοιπο ακέραιας διαίρεσης)

Αυτό σημαίνει ότι κάθε πράξη, με τα σύμβολα *, /, DIV, MOD εκτελείται πρώτη, εκτός αν υπάρχει παρένθεση, οπότε εκτελείται πρώτα η πράξη που υπάρχει μέσα στην παρένθεση. Η πράξη του πολλαπλασιασμού, σε σχέση με τη διαίρεση, έχει την ίδια προτεραιότητα. Η πρόσθεση επίσης έχει την ίδια προτεραιότητα με την αφαίρεση. Όταν τα σύμβολα έχουν την ίδια προτεραιότητα οι πράξεις εκτελούνται από αριστερά προς τα δεξιά. Μερικά παραδείγματα εκφράσεων είναι τα παρακάτω:

Έκφραση	*Αποτέλεσμα*
20 DIV 3 * 4	24
20 MOD 3 * 4	8
6 * 3 / 2 * 4	36
6 * 3 / (2 * 4)	2,25
6 + 3 / (2 * 4)	1,125

8.2.3. *Λογικός*

Ο **λογικός τύπος (Boolean)** έχει δύο μόνο τιμές, την **Αληθή (true)** και την **Ψευδή (false).** Πολλές φορές ο σκοπός μιας μεταβλητής λογικού τύπου είναι η καταγραφή του αποτελέσματος ενός ελέγχου. Εάν σε κάποιο σημείο του προγράμματος χρειαστεί να εξετάσουμε το αποτέλεσμα ενός ελέγχου, αρκεί να εξετάσουμε αν η τιμή της λογικής μεταβλητής είναι **Αληθής (true)** ή **Ψευδής (false)**.

Οι πράξεις που μπορούν να γίνουν με μεταβλητές ή εκφράσεις λογικού τύπου είναι η σύζευξη, η διάζευξη, η αποκλειστική διάζευξη και η άρνηση και γίνονται με χρήση των λογικών τελεστών **and**, **or**, **xor** και **not** αντίστοιχα.

Στον πίνακα 8-3 παρουσιάζεται τα αποτελέσματα των παραπάνω πράξεων μεταξύ δύο λογικών μεταβλητών με όλους του συνδυασμούς των τιμών τους.

Πίνακας 8-3

Η σύζευξη, η διάζευξη, η αποκλειστική διάζευξη και η άρνηση με χρήση δύο λογικών μεταβλητών P, Q

P	Q	P and Q	P or Q	P xor Q	not P
True	True	True	True	False	False
True	False	False	True	True	False
False	True	False	True	True	True
False	False	False	False	False	True

Λογικές εκφράσεις είναι οι απεικονίσεις παραστάσεων που μπορεί να περιέχουν σταθερές, μεταβλητές, συναρτήσεις, αριθμητικά σύμβολα και παρενθέσεις και μπορούν να πάρουν μια λογική τιμή (true ή false). Μία λογική έκφραση παράγεται από δύο μεταβλητές ή σταθερές μέσω των σχεσιακών τελεστών (βλέπε πίνακα 8-4).

Πίνακας 8-4

Σχεσιακοί τελεστές

Περιγραφή	Μαθηματικά	Pascal
Μεγαλύτερο από	>	>
Μικρότερο από	<	<
Μεγαλύτερο ή ίσο	≥	>=
Μικρότερο ή ίσο	≤	<=
Διάφορο	≠	<>
Ανήκει	∈	in

Ο Λογικός τελεστής **and** δέχεται δύο λογικές εκφράσεις και δίνει τιμή true, μόνο όταν και οι δύο λογικές εκφράσεις έχουν τιμή true. Ο τελεστής and έχει χαρακτηριστεί ως τελεστής λογικού πολλαπλασιασμού.

Η έκφραση (x>1) and (x<10) έχει τιμή true, μόνο όταν ο x παίρνει τιμές από 2 μέχρι και 9. Ο τελεστής and έχει προτεραιότητα σε σχέση με τους <, >, γι' αυτό οι παρενθέσεις είναι απαραίτητες. Αν στην προηγούμενη σχέση γράψουμε x>1 and x<9, αυτή είναι ισοδύναμη με την x>(1 and x)<9, η οποία στερείται νοήματος.

Ο Λογικός τελεστής **or** δέχεται δύο λογικές εκφράσεις και δίνει τιμή true, όταν τουλάχιστον μία από τις δύο λογικές εκφράσεις έχει τιμή true. Ο τελεστής or έχει χαρακτηριστεί ως τελεστής λογικής πρόσθεσης και επομένως έχει μικρότερη προτεραιότητα από τον τελεστή and. 'Ετσι η σχέση:

if ((x>=0) and (x<=10)) or (x=20) then...

είναι ισοδύναμη με τη σχέση if (x>=0) and (x<=10) or (x=20) then...

Ο Λογικός τελεστής **xor** δέχεται δύο λογικές εκφράσεις και δίνει τιμή true, όταν μόνο μία από τις δύο λογικές εκφράσεις έχει τιμή true. Ο τελεστής xor έχει την ίδια προτεραιότητα με τον τελεστή or.

Ο τελεστής **not** είναι ο λογικός τελεστής που δέχεται ως παράμετρο μία μόνο λογική μεταβλητή ή έκφραση και επιστρέφει την αντίθετη τιμή της δηλαδή δίνει τιμή true, αν η παράμετρος έχει τιμή false, ή δίνει false, αν η παράμετρος έχει τιμή true.

8.2.4. Χαρακτήρας

Ο **τύπος** (**char**) περιγράφει δεδομένα ενός χαρακτήρα μέσα από το σύνολο των χαρακτήρων του υπολογιστή. Σε ένα πρόγραμμα Pascal μια τιμή ενός δεδομένου αυτού του τύπου γράφεται: 'A','B','$','&' κλπ.

Ο τύπος χαρακτήρας Char είναι ένας διατεταγμένος τύπος ο οποίος περιλαμβάνει το σύνολο των χαρακτήρων που διαθέτει ο υπολογιστής μας. Η διάταξη του συνόλου των χαρακτήρων διαφέρει από υπολογιστή σε υπολογιστή. Γενικά πάντως τα ψηφία 0,1,2,...9 είναι συνεχόμενα. Το ίδιο και τα γράμματα A,B,C,D,...Z, a,b,c,d,...,z και ακολουθούν οι Ελληνικοί χαρακτήρες A,B,...,Ω, α,β,...,ω.

8.2.5. Αλφαριθμητικός Τύπος

Ο **Αλφαριθμητικός (String)** τύπος δεν συναντάται στην Standard Pascal. Ο αλφαριθμητικός τύπος είναι μία σειρά από 255, το πολύ χαρακτήρες. Εάν στη δήλωση αυτού του τύπου δεν έχει αναφερθεί το μήκος, τότε λαμβάνεται το μέγιστο μήκος, δηλαδή 255 χαρακτήρες. Το περιεχόμενο μιας μεταβλητής αυτού του τύπου μπορεί να είναι μία λέξη ή μία φράση, όπως ονοματεπώ-

νυμο, διεύθυνση κατοικίας κλπ. τα οποία περιλαμβάνονται μεταξύ εισαγωγικών. Για παράδειγμα επιτρεπτές τιμές του τύπου αυτού μπορεί να είναι:

- 'Turbo Pascal',
- 'ΤΕΕ Τεχνικά Επαγγελματικά Εκπαιδευτήρια',
- 'Τομέας Πληροφορικής' κλπ.

Ένα string πρέπει να γράφεται στην ίδια γραμμή, ειδάλλως η Turbo Pascal θα δώσει το μήνυμα STRING CONSTANT EXCEEDS LINE (η σταθερά string ξεπερνάει τη γραμμή) Η τιμή μιας μεταβλητής αλφαριθμητικού τύπου είναι η σειρά των χαρακτήρων που περιλαμβάνονται μεταξύ των εισαγωγικών. Για παράδειγμα **'Vathmos'** είναι η σειρά των χαρακτήρων που αποτελείται από τους χαρακτήρες **V a t h m o s** χωρίς τα εισαγωγικά. **Vathmos** χωρίς εισαγωγικά είναι μία μεταβλητή, το όνομα μίας περιοχής της μνήμης. Η τιμή του string '**2468**' είναι η σειρά των χαρακτήρων **2 4 6 8**. Αν γράψουμε **2468** χωρίς τα εισαγωγικά είναι ένας ακέραιος τον οποίο μπορούμε να χρησιμοποιήσουμε σε υπολογισμούς.

Ένα string μπορεί να αποτελείται, όπως αναφέραμε, από 0 έως 255 το πολύ χαρακτήρες. Ένα string με 0 χαρακτήρες είναι το κενό (null string) και δηλώνεται με δύο εισαγωγικά χωρίς κενό μεταξύ τους. Η Turbo Pascal παρέχει λειτουργίες για τη συνένωση αλφαριθμητικών τύπων, για την απομάκρυνση χαρακτήρων από ένα string και για τη σύγκριση των τιμών αλφαριθμητικών τύπων.

8.3. ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ PASCAL

8.3.1. Επικεφαλίδα

Πρόγραμμα Pascal
program όνομα προγράμματος;
δηλώσεις
begin
εκτελέσιμες εντολές
end.

Ένα πρόγραμμα Pascal αρχίζει με τη λέξη **program** που ακολουθείται από το **όνομα** του προγράμματος και μία **λίστα με ονόματα αρχείων** τα οποία μπορεί να είναι προαιρετικά μονάδες εισόδου / εξόδου ή τα αρχεία που θα χρησιμοποιήσει κατά την εκτέλεσή του. Η πρώτη γραμμή είναι η **επικεφαλίδα** (**heading**) του προγράμματος. Η μορφή της επικεφαλίδας είναι:

program όνομα προγράμματος(όνομα αρχείου, όνομα αρχείου,...);

Παραδείγματα:

program test_1 (input,output);

program test_2;

8.3.2. Δηλώσεις

Η επικεφαλίδα ακολουθείται από προτάσεις **δηλώσεων (declarations)** που αναφέρονται στους όρους που χρησιμοποιούνται στο πρόγραμμα. Οι προτάσεις δηλώσεων καταλήγουν σε ερωτηματικό (;). και ακολουθούν την εξής σειρά:

- Δηλώσεις **σταθερών (constants)**, χρησιμοποιούνται για τον ορισμό ονομάτων δεδομένων τα οποία παραμένουν σταθερά. Κάθε σταθερή τιμή που περιλαμβάνεται σε ένα πρόγραμμα ονομάζεται literal. Πριν από τον ορισμό των σταθερών αναγράφεται η λέξη **const.**

Παραδείγματα:

```
const
      pososto=18;
      nomos='ΑΘΗΝΩΝ';
      pi=3.141592653;
      e=2.718281828;
      max= 100;
      min=-100;
      part_no=1234567;
      Invalid_d=-9999999;
      letter='y';
      name= 'Blaise Pascal';
```

Ο τύπος της σταθεράς προσδιορίζεται από την τιμή της. Αν υπάρχει δεκαδικό σημείο η σταθερά θεωρείται τύπου πραγματικού. Αν δεν υπάρχει δεκαδικό σημείο εξετάζεται αν η σταθερά βρίσκεται στα όρια των ακεραίων θεωρείται τύπου Integer, διαφορετικά θεωρείται LongInt. Αν η τιμή βρίσκεται ανάμεσα σε εισαγωγικά είναι τύπου χαρακτήρα ή αλφαριθμητικού ανάλογα με το πλήθος των χαρακτήρων που περιλαμβάνει.

- Δηλώσεις **τύπων (type)**, εφόσον ορίζονται νέοι τύποι.
- Δηλώσεις **μεταβλητών (variables)**, χρησιμοποιούνται για τον ορισμό των ονομάτων δεδομένων τα οποία μεταβάλλονται και για τα οποία αναφέρεται ο τύπος τους. Δύο ή περισσότερες μεταβλητές μπορεί να είναι του ιδίου τύπου οπότε διαχωρίζονται με ένα κόμμα. Ο τύπος αναφέρεται στο τέλος της γραμμής μετά το σύμβολο (:). Πριν από τον ορισμό των μεταβλητών αναγράφεται η λέξη **var.**

Παραδείγματα

```
var
  pososto_forou,mistos,kath_apodoxes:real;
  eponymo, onoma : string[25];
```

- Δηλώσεις των υποπρογραμμάτων που ορίζει ο χρήστης
  Συναρτήσεις (functions) και
  Διαδικασίες (procedures) εφόσον υπάρχουν.

8.3.3. Κύριο Πρόγραμμα

Σύνθετη Εντολή (Compound statement)
Είναι μία ομαδοποίηση εντολών οι οποίες χρησιμοποιούνται όπως μία απλή εντολή και περικλείονται μεταξύ των λέξεων begin και end.

Η λέξη **begin** (**αρχή**) δηλώνει την αρχή των εντολών του προγράμματος που εκτελεί ο υπολογιστής, δηλαδή το **κύριο** πρόγραμμα, ενώ η λέξη **end** (**τέλος**) δηλώνει το τέλος του προγράμματος.

Το κύριο πρόγραμμα αποτελείται από μια σειρά εντολών, σύμφωνα με τον αλγόριθμο του προβλήματος. Κάθε εντολή που γράφεται τελειώνει υποχρεωτικά με ένα ερωτηματικό (;). Το σύμβολο αυτό είναι διαχωριστικό εντολών. Δεν είναι υποχρεωτικό να γραφεί μόνο όταν η επόμενη γραμμή αρχίζει με end (τέλος προγράμματος ή σύνθετης εντολής ή υποπρογράμματος ή προγράμματος). Μετά το end πρέπει να γραφεί μία τελεία (.) η οποία δηλώνει και το τέλος του προγράμματος

Παραδείγματα προγράμματος

1.

```
program thermokrasia;
{Το πρόγραμμα υπολογίζει τη μέση θερμοκρασία }
{πήξης και βρασμού                              }
 uses wincrt;
const  {δηλώσεις σταθερών}
  pixis_c=0;      {θερμοκρασία πήξης βαθμοί Κελσίου}
  vrasmou_c=100; {θερμοκρασία βρασμού βαθ.Κελσίου }
var    {δηλώσεις μεταβλητών}
  mesi_th_k :real; {μέση τιμή, πήξης - βρασμού}
{κύριο πρόγραμμα}
  begin
   writeln('θερμ. πήξης    ', pixis_c,    ' βαθμοί');
   writeln('θερμ. βρασμού ', vrasmou_c, ' βαθμοί');
   mesi_th_k:=( pixis_c + vrasmou_c)/2;
   writeln('μέση τιμή      ', vrasmou_c, ' βαθμοί');
  end.
```

2.

```
program mistos;
   uses wincrt;
   const  {δηλώσεις σταθερών}
        orio=2000000;    {όριο υπολογισμού φόρου}
        pos_forou=0.20; {ποσοστό υπολογισμού φόρου}
        wrom=1500;       {ποσό ανά υπερωριακή ώρα}
   var    {δηλώσεις μεταβλητών}
      bm,yper,foros,akath,plir:real;
        {bm     = βασικός μισθός
        yper   = ώρες υπερωρίας
        foros = φόρος
        akath = ακαθάριστες αποδοχές
        plir   = πληρωτέο ποσό}
        yp_yper:real;     {αποδοχές από υπερωρίες}

function ypol_ap(basikos,yp:real):real;
  {συνάρτηση υπολογισμού αποδοχών}
```

```
    begin
      yp_yper:=yp*wrom;              {υπολ. υπερωριών}
      ypol_ap:=yp_yper+basikos;      {υπολ. αποδοχών}
    end;
function ypol_forou(synolo:real):real;
   {συνάρτηση υπολογισμού φόρου}
    begin
      if synolo<=orio     {συνθήκη υπολογισμού φόρου}
       then ypol_forou:=0
       else ypol_forou:=synolo*pos_forou;
    end;
procedure  ektyposi;
   {διαδικασία εκτύπωσης αποτελεσμάτων}
 begin
    writeln(bm:6:0,' ',yp_yper:6:0,' ',akath:6:0,'
          ',foros:6:0,' ',plir:6:0);
 end;

{κύριο πρόγραμμα}
   begin
     writeln('δώσε βασικό μισθό, ώρες υπερωρίας ');
     readln(bm,yper);
     akath:=ypol_ap(bm,yper);     {ακαθάριστες αποδοχές}
     foros:=ypol_forou(akath);    {φόρος}
     plir:=akath-foros;           {πληρωτέο - καθαρά}
     ektyposi;                 {εκτύπωση αποτελεσμάτων}
end.
```

Ανακεφαλαίωση

Γνωρίσαμε την δομή και τη φιλοσοφία της γλώσσας Pascal. Γνωρίσαμε τους βασικούς τύπους δεδομένων της γλώσσας και τον τρόπο ορισμού σταθερών και μεταβλητών. Κατανοήσαμε τη δομή ενός προγράμματος σε γλώσσα Pascal. Γνωρίσαμε την έννοια της σύνθετης εντολής. Γνωρίσαμε τη δυνατότητες δημιουργίας τύπων δεδομένων από τον προγραμματιστή και της δυναμικής διαχείρισης της μνήμης με δείκτες.

Ερωτήσεις

1. Να αναφέρετε μερικά από τα ιδιαίτερα χαρακτηριστικά της Pascal.
2. Ποιό είναι το αλφάβητο της Pascal;
3. Ποιά είναι τα ονόματα- ταυτότητες και σε τι χρησιμεύουν;
4. Σε τί χρησιμεύει το Συντακτικό (Syntax) της γλώσσας;
5. Σε τί χρησιμεύει η Σημασιολογία (Semantics);
6. Ποιοί είναι οι απλοί ή στοιχειώδεις τύποι δεδομένων;
7. Πότε χρησιμοποιείται ο πραγματικός τύπος;
8. Ποιές είναι οι πράξεις που μπορεί να γίνουν με μεταβλητές ή εκφράσεις λογικού τύπου;
9. Να εξηγήσετε τη σημασία των: 2000, '2000', etos, 'etos' στη γλώσσα Pascal.
10. Ποιά είναι η δομή ενός προγράμματος Pascal;

Ασκήσεις

1. Να γράψετε με τον editor, να μεταφράσετε και να εκτελέσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τις κατάλληλες εντολές.

```
program charset(output);
{  το πρόγραμμα εκτυπώνει το σύνολο των χαρακτήρων
ASCII }
var
 ch:char;
begin
 for ch:=chr(32) to chr(255) do
    write(ch);
 writeln
end.
```

2. Να γράψετε με τον editor,να μεταφράσετε και να εκτελέσετε το παρακάτω πρόγραμμα χρησιμοποιώντας τις κατάλληλες εντολές.

```
program table(output);
{το πρόγραμμα τυπώνει τα τετράγωνα και τους κύβους
 των αριθμών 1 έως και 10 }
var
 i:integer;
begin
 writeln('Number':10,'square':10,'cube':10);
 for i:=1 to 10 do
writeln(i:10,sqr(i):10,i*sqr(i):10)
end.
```

Δραστηριότητες

Να γράψετε με τον editor, να **μεταφράσετε** και να εκτελέσετε χρησιμοποιώντας τις κατάλληλες εντολές τα παρακάτω προγράμματα στο περιβάλλον της **Qbasic.** Να συγκρίνετε τα αποτελέσματα με αυτά των αντίστοιχων ασκήσεων 1 και 2 που προηγήθηκαν.

```
'άσκηση 1 Κεφάλαιο 8  Η γλώσσα Pascal
'Το πρόγραμμα αυτό τυπώνει το σύνολο των χαρακτήρων
ASCII
CLS
FOR i= 32 to 255
   PRINT CHR$(i);
NEXT I
END
'άσκηση 2 Κεφάλαιο 8  Η γλώσσα Pascal
'Το πρόγραμμα αυτό τυπώνει τους αριθμούς 1 έως 10
'καθώς και τα τετράγωνα και τους κύβους τους.
CLS
PRINT  TAB(10); "αριθμός"; TAB(20); "τετράγωνο";
TAB(32); "κύβος"
FOR i= 1 to 10
   PRINT TAB(12); i; TAB(22); i^2; TAB(32); I^3;
   PRINT
NEXT I
END
```

ΚΕΦΑΛΑΙΟ 9
ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ

Σκοπός κεφαλαίου:

Να μπορούμε γρήγορα να υλοποιήσουμε ένα απλό πρόγραμμα σε γλώσσα Pascal.

Ειδικοί σκοποί:

- Να κατανοήσουμε τις εντολές εισόδου/εξόδου.
- Να κατανοήσουμε τη σημασία της εντολής αντικατάστασης.
- Να κατανοήσουμε τον τρόπο της διαδοχικής εκτέλεσης των εντολών ενός προγράμματος.

9.1. ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ / ΕΞΟΔΟΥ

Τα δεδομένα των προγραμμάτων λαμβάνονται από μια μονάδα εισόδου, όπως για παράδειγμα το πληκτρολόγιο. Για την εισαγωγή των δεδομένων η Pascal χρησιμοποιεί τις διαδικασίες (**procedures**) **read** ή **readln**. Για την εμφάνιση των περιεχομένων μεταβλητών ή σταθερών στην οθόνη, στον εκτυπωτή ή σ' ένα αρχείο χρησιμοποιούνται οι διαδικασίες **write** ή **writeln**.

Οι εντολές read, readln, write, writeln χρησιμοποιούνται και για ανάγνωση/εγγραφή από αρχεία τα οποία είναι καταχωρημένα στο δίσκο. Στην παρούσα ενότητα, θα αναφερθεί η χρήση των παραπάνω εντολών για ανάγνωση από το πληκτρολόγιο και εγγραφή στην οθόνη. Όταν οι εντολές εισόδου/εξόδου χρησιμοποιούνται στην Turbo Pascal για Windows, θα πρέπει να γραφεί αμέσως μετά τη δήλωση του προγράμματος η δήλωση **uses wincrt**.

Για πρακτικούς λόγους οι διαδικασίες εισόδου / εξόδου ονομάζονται και εντολές εισόδου / εξόδου.

Η χρήση των εντολών εισόδου/εξόδου σε αρχεία θα γίνει σε ξεχωριστό κεφάλαιο.

9.1.1. read, readln

***Μορφή*: read (parameter,parameter,...,)**

Ενέργεια: Η διαδικασία read ακολουθείται από μία ή περισσότερες παραμέτρους. Οι παράμετροι αυτές είναι τα ονόματα των διευθύνσεων της μνήμης όπου θα αποθηκευτούν τα δεδομένα, δηλαδή ονόματα **μεταβλητών**

που χωρίζονται με ',' (κόμμα) και περιέχονται σε μία παρένθεση. Όταν εκτελείται η διαδικασία, τα δεδομένα διαβάζονται και αποδίδονται, κατά τη σειρά που είναι γραμμένα, στις μεταβλητές όπως αυτές εμφανίζονται στη διαδικασία read. Όταν γίνεται εισαγωγή δεδομένων, και εφόσον αυτά είναι αριθμητικά και εμφανίζονται περισσότερα από ένα ανά γραμμή, πρέπει να χωρίζονται με ένα τουλάχιστον κενό. Τα δεδομένα τύπου char δεν πρέπει να χωρίζονται με κενά, γιατί το κενό είναι κι αυτό ένας χαρακτήρας ο οποίος θα διαβαστεί.

Αν για παράδειγμα οι παράμετροι της εντολής read είναι ακέραιοι για κάθε παράμετρο, αρχίζοντας από την πρώτη, εκτελούνται τα παρακάτω:

- 1. Διαβάζεται ένας αριθμός από το πληκτρολόγιο.
- 2. Αποθηκεύεται ο αριθμός στη μεταβλητή (αντίστοιχη διεύθυνση της μνήμης)
- 3. Επαναλαμβάνονται τα βήματα 1 και 2 για την επόμενη μεταβλητή.

Όταν αποθηκευτεί τιμή και στην τελευταία μεταβλητή της λίστας των παραμέτρων, τελειώνει και η εκτέλεση της εντολής **read.**

Θα πρέπει να είμαστε προσεκτικοί κατά την εισαγωγή των δεδομένων. Η εισαγωγή πραγματικού αριθμού σε ακέραια μεταβλητή δεν είναι επιτρεπτή. Μπορούμε όμως να διαβάσουμε μια ακέραια τιμή σε πραγματική μεταβλητή.

	Εντολή	Δεδομένα	Περιεχόμενα μετά την εκτέλεση
1	read(x)	22	x=22
2	read(a, b, c)	33 44 55	a=33 b=44 c=55
3	read(d, e)	66 77	d=66 e=77
4	read(f, g)	11.99 21 31 41 51	f=11.99 g=21
5	read(x, a, d)		x=31 a=41 d=51

Παρατηρήσεις:

- Τα δεδομένα της γραμμής 3 δεν είναι αρκετά για όλες της παραμέτρους κι έτσι η εισαγωγή δεδομένων συνεχίζεται στην επόμενη γραμμή.
- Οι μεταβλητές της εντολής read στη γραμμή 4 είναι λιγότερες από τα δεδομένα. Για το λόγο αυτό, τα επιπλέον δεδομένα είναι διαθέσιμα για την επόμενη εντολή read στη γραμμή 5.

Στα προηγούμενα παραδείγματα όλες οι μεταβλητές είναι **ακέραιες (integer)** εκτός από την f η οποία είναι **πραγματική(real)**.

Οι εντολές read(a), read(b) και read(a, b) είναι ισοδύναμες. Η λίστα των μεταβλητών διευκολύνει τον προγραμματιστή στην κωδικοποίηση.

***Μορφή*: readln(parameter, parameter,...,)**

Ενέργεια Η διαδικασία readln, μπορεί να ακολουθείται από μία ή περισσότερες παραμέτρους (**μεταβλητές**) που χωρίζονται με ',' (κόμμα) και περιέχονται σε μία παρένθεση. που είναι ισοδύναμη των read και readln. Κατ' αυτό τον τρόπο τα ζητούμενα διαβάζονται από την τρέχουσα γραμμή (read) και στη συνέχεια ο έλεγχος περνάει στην επόμενη γραμμή (readln). Η διαδικασία readln, όταν εμφανίζεται χωρίς παραμέτρους, έχει ως αποτέλεσμα να περνάει ο έλεγχος στην επόμενη γραμμή.

Ας υποθέσουμε ότι έχουμε δύο ακέραιες τιμές σε κάθε γραμμή δεδομένων. Ο πίνακας που ακολουθεί δείχνει τα περιεχόμενα των μεταβλητών μετά την εκτέλεση της εντολής readln.

	Εντολή	Δεδομένα	Περιεχόμενα μετά την εκτέλεση
1	readln(a) readln(b) readln(c)	10 20 30 40 50 60	a=10 b=30 c=50
2	readln(a,b,c) readln(d,e)	10 20 30 40 50 60	a=10 b=20 c=30 d=50 e=60
3	readln (a, b) readln(c,d) readln(e)	10 20 30 40 50 60	a=1 b=20 c=30 d40 e=50
4	readln(a) readln(b) readln(c,d,e)	10 20 30 40 50 60	a=10 b=30 c=50 d=60 e=?
5	readln(a,b) readln readln(c)	10 20 30 40 50 60	a=10 b=20 c=50

Παρατηρήσεις:

- Δεν υπάρχουν δεδομένα για τη μεταβλητή e η εκτέλεση σταματάει μέχρι να γίνει εισαγωγή από το πληκτρολόγιο.
- Μια εντολή readln χωρίς παραμέτρους έχει ως αποτέλεσμα να παραλειφθεί μία γραμμή δεδομένων.

9.1.2. write, writeln

***Μορφή*: write (parameter,parameter,...,)**

Ενέργεια: Οι παράμετροι μπορεί να είναι σταθερές, μεταβλητές ή εκφράσεις, σε αντίθεση με την εντολή read η οποία δέχεται ως παραμέτρους μόνο μεταβλητές. Η τιμή καθεμιάς παραμέτρου κατά σειρά, τυπώνεται στη γραμμή από αριστερά προς τα δεξιά Στην απλούστερη περίπτωση οι παράμετροι μπορεί να είναι αριθμοί, χαρακτήρες ή **αλφαριθμητικά στοιχεία (συμβολοσειρές) (strings)**. Εάν πρόκειται να γραφούν **συμβολοσειρές** σταθερών, αυτές θα πρέπει να περιέχονται μεταξύ απλών εισαγωγικών.

Παράδειγμα:

write ('αυτή είναι μία συμβολοσειρά')

***Μορφή*: writeln(parameter, parameter,...,)**

Ενέργεια: Η διαδικασία writeln, αν ακολουθείται από παραμέτρους, είναι ισοδύναμη των write και writeln. Η τιμή κάθε μιας παραμέτρου κατά σειρά, τυπώνεται στην γραμμή από αριστερά προς τα δεξιά και στη συνέχεια η επόμενη εντολή εισόδου / εξόδου εκτελείται στην αρχή της επόμενης γραμμής. Η διαδικασία writeln, όταν εμφανίζεται χωρίς παραμέτρους, έχει ως αποτέλεσμα να περνάει ο έλεγχος στην αρχή της επόμενης γραμμής.

Ο πίνακας που ακολουθεί δείχνει τα αποτελέσματα μετά την εκτέλεση εντολής **writeln.**

Περιεχόμενα παραμέτρων	Εντολή	Αποτελέσματα
i=2	writeln (i)	2
	writeln('i=', i)	i=2
	writeln('Error=',i)	Error=2
	writeln('Error Message')	Error Message
la ='Pascal'	writeln(ty, ' ',la)	TURBO Pascal
ty ='TURBO'	writeln(ty, la)	TURBOPascal
	writel writeln('Language=', la)	Language=Pascal

Η εισαγωγή των δεδομένων γίνεται με την εντολή read. Η εισαγωγή απλοποιείται, όταν με τη διαδικασία write το ίδιο το πρόγραμμα δίνει ένα προτρεπτικό μήνυμα, δηλαδή εμφανίζει στην οθόνη ποια είναι τα ζητούμενα δεδομένα.

Παράδειγμα:
```
write('Δώσε βασικό μισθό :  ');
readln(mistos);
```

Μορφοποιημένη εκτύπωση

Οι δύο βασικές τεχνικές για τη μορφοποίηση της εκτύπωσης είναι οι κενές γραμμές και η εισαγωγή κενών στις γραμμές εκτύπωσης. Για να έχουμε κενές γραμμές αρκεί να δώσουμε την εντολή writeln χωρίς παραμέτρους τόσες φορές όσες είναι οι κενές γραμμές που θέλουμε. Για την εισαγωγή κενών στη γραμμή μεταξύ των παραμέτρων περιλαμβάνουμε και όσα κενά θέλουμε μεταξύ εισαγωγικών.

Ο έλεγχος της μορφής των αποτελεσμάτων στη γραμμή εκτύπωσης γίνεται με πληροφορία που ακολουθεί την παράμετρο στη διαδικασία write ή writeln. Η πληροφορία αυτή δίνει το επιθυμητό εύρος του πεδίου δηλαδή τις θέσεις που θα καταλάβει η παράμετρος κατά την εκτύπωσή της..

Για σταθερές / μεταβλητές **Ακέραιου** ή **Αλφαριθμητικού** τύπου, η μορφή εκτύπωσης είναι **write(x:m).** Όπου:

x η μεταβλητή που θα τυπωθεί,

m το συνολικό πλήθος των θέσεων εκτύπωσης.

Η διαδικασία write(x:m), όπου **x** είναι ακέραια έκφραση, τυπώνει την τιμή της μεταβλητής x αρχίζοντας από τα δεξιά ενός πεδίου εύρους **m**. Αν το εύρος είναι μεγαλύτερο από το μήκος του πεδίου που πρόκειται να τυπωθεί, εμφανίζονται κενά στην αριστερή πλευρά της τιμής. Αν το εύρος είναι μικρότερο από το μήκος του πεδίου που πρόκειται να τυπωθεί, αγνοείται η οδηγία μορφοποίησης και χρησιμοποιείται το ελάχιστο απαιτούμενο εύρος.

Η μορφοποιημένη εκτύπωση μπορεί να χρησιμοποιηθεί και για σταθερές ή μεταβλητές αλφαριθμητικού τύπου (string). Η χρήση γίνεται όπως και στους ακεραίους, δηλαδή δίνεται το πλήθος των θέσεων εκτύπωσης της μεταβλητής ή της σταθεράς. Η στοίχιση γίνεται από δεξιά.

Π.χ. Η εντολή `writeln ('Αριθμός μητρώου.':20);` θα έχει ως αποτέλεσμα να αφήσει 4 κενές θέσεις εκτύπωσης από την αρχή της γραμμής ή από την τελευταία θέση εκτύπωσης και μετά να γράψει το περιεχόμενο της σταθεράς δηλαδή: `Αριθμός μητρώου.`

Για σταθερές / μεταβλητές **Πραγματικού** τύπου η μορφή εκτύπωσης είναι **write(x:m:n)**. Όπου:

x η μεταβλητή πραγματικού τύπου που θα τυπωθεί,

m το συνολικό πλήθος των θέσεων εκτύπωσης,
συμπεριλαμβανομένης της υποδιαστολής και

n το πλήθος των δεκαδικών ψηφίων.

Με τον τρόπο αυτό γίνεται η στρογγυλοποίηση και ο αριθμός εμφανίζεται με τη γνωστή από τα Μαθηματικά, μορφή σταθερής υποδιαστολής. Π.χ. `write(3.14159265:10:4)` θα δώσει ως αποτέλεσμα `3.1416`. Γίνεται δηλαδή στρογγυλοποίηση στο τέταρτο δεκαδικό ψηφίο και εμφανί-ζονται και 4 κενά πριν από τον αριθμό, για να συμπληρωθεί το πλήθος των 10 θέσεων.

9.2. ΕΝΤΟΛΗ ΑΝΤΙΚΑΤΑΣΤΑΣΗΣ

***Μορφή*: μεταβλητή := έκφραση**

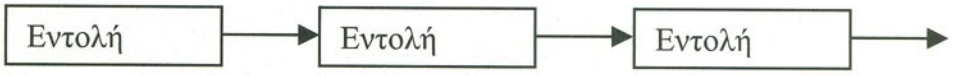
Ενέργεια: Το περιεχόμενο της μνήμης που ορίζεται με το όνομα της μεταβλητής που βρίσκεται αριστερά του := αντικαθίσταται από την τιμή που ορίζεται από την έκφραση η οποία βρίσκεται δεξιά του := (συμβόλου αντικατάστασης). Ο συμβολισμός := θα πρέπει να θεωρείται ως ένα μόνο σύμβολο και για το λόγο αυτό δεν επιτρέπονται κενά. Η έκφραση του δεξιού μέρους μπορεί να είναι και μία σταθερά ή άλλη μεταβλητή και τότε η τιμή τους αντικαθιστά την τιμή της μεταβλητής που βρίσκεται αριστερά.

Παραδείγματα:

```
x:=1;          {η μεταβλητή x λαμβάνει την τιμή 1}
x:=x+1;        {το περιεχόμενο της μεταβλητής x αυξάνει  κατά 1 και
                το αποτέλεσμα καταχωρείται στη μεταβλητή x}
y:=(3*x+5)/(x+3);
               {το περιεχόμενο της μεταβλητής y αντικαθίσταται
          από την τιμή της αλγεβρικής παράστασης (3x+5)/(x+3)
onoma:='ΑΡΗΣ';
               {Η μεταβλητή onoma, η οποία πρέπει να έχει
                ορισθεί ως string, λαμβάνει την τιμή ΑΡΗΣ}
star := '*' ; {Η μεταβλητή star, η οποία πρέπει να έχει ορισθεί
                ως  string ή  char λαμβάνει την τιμή *}
```

9.3. ΑΚΟΛΟΥΘΙΑ

Ακολουθία είναι μια σειρά από εντολές οι οποίες εκτελούνται η μία μετά την άλλη.



Το παρακάτω πρόβλημα είναι ένα χαρακτηριστικό παράδειγμα ακολουθίας:

Με δεδομένα την προφορική βαθμολογία των δύο τετραμήνων και τη γραπτή βαθμολογία, να υπολογιστεί ο μέσος όρος της βαθμολογίας.

Όπως φαίνεται, το πρόγραμμα που ακολουθεί αποτελείται από διαδοχικές εντολές, εισόδου, εξόδου και αντικατάστασης.

```
program mo_bathmwn;
{Μέσος όρος βαθμολογίας}
uses wincrt;
var pr1, pr2, gr, sum     : integer;
    mo, mo_pr             : real;

begin
  write('δώσε βαθμολογία προφορικών Α ΤΕΤΡΑΜΗΝΟΥ  ');
  readln(pr1);
  write ('δώσε βαθμολογία προφορικών Β ΤΕΤΡΑΜΗΝΟΥ  ');
  readln(pr2);
  write ('δώσε βαθμολογία ΓΡΑΠΤΩΝ                 ');
  readln(gr);
  mo_pr:=(pr1+pr2)/2;
  mo:= (mo_pr+ gr)/2;
  writeln('προφορικά =',mo_pr :3:1);
  writeln('γραπτά    =',gr :2);
  writeln('μέσος όρος=',mo :3:1);
end.
```

Αναχεφαλαίωση

Γνωρίσαμε τη σύνταξη και τον τρόπο χειρισμού των εντολών εισόδου στοιχείων στον υπολογιστή από το πληκτρολόγιο. Γνωρίσαμε τη σύνταξη και τον τρόπο χειρισμού των εντολών εξόδου στοιχείων στην οθόνη του υπολογιστή. Κατανοήσαμε τη χρήση και τη σημασία των εντολών αυτών μέσα από χαρακτηριστικά παραδείγματα. Γνωρίσαμε τις δυνατότητες να δημιουργούμε μορφοποιημένες εκτυπώσεις. Επίσης κατανοήσαμε την έννοια της εντολής αντικατάστασης και της ακολουθίας εντολών. Τέλος χρησιμοποιήσαμε τις εντολές αυτές για τη δημιουργία των πρώτων εντολών Pascal.

Ερωτήσεις

1. Να εξηγήσετε πώς λειτουργούν οι διαδικασίες
   read(parameter, parameter,...) και readln(parameter,parameter,...);
   Ποιες είναι οι ομοιότητες και ποιες είναι οι διαφορές τους;
2. Να εξηγήσετε πώς λειτουργούν οι διαδικασίες write(parameter, parameter,...) και writeln(parameter,parameter,...);
   Ποιές είναι οι ομοιότητες και ποιές οι διαφορές τους;

3. Να εξηγήσετε πώς λειτουργούν οι διαδικασίες write(x : m) και writeln(x :m :n); Ποιες είναι οι ομοιότητες και ποιες οι διαφορές τους;
4. Ποιο θα είναι το αποτέλεσμα της εντολής write(3.14159265:10:3);
5. Ποια είναι τα αποτελέσματα των παρακάτω εντολών, αν αυτές εκτελούνται διαδοχικά x:=2; x:=2*x+4; y:=(4*x-1)*(5*x-4);
6. Να συμπληρώσετε τα κενά στον πίνακα που ακολουθεί.

	Εντολή	Δεδομένα	Περιεχόμενα μετά την εκτέλεση
1	read(x)	122	x=
2	read(a, b, c)	133 244 355	a= b= c=
3	read(d, e)	166 277 3888	d= e=
4	read(f, g)	11.99 21 31 41 51	f= g= h=
5	read(x, a, d)	10 20 30	x= a= d=

7. Να συμπληρώσετε τα κενά στον πίνακα που ακολουθεί.

Περιεχόμενα παραμέτρων	Εντολή	Αποτελέσματα
i=2	writeln (i) writeln('i=', i) writeln('Error=',i) writeln('Error Message')	i= _____ _____ _____= ____ _____ ___
la ='name=' ty ='Tttt'	writeln(ty, ' ',la) writeln(ty, la) writel writeln('Language=', la)	_____ _____ _____

Ασκήσεις

1. Να γράψετε πρόγραμμα που να τυπώνει:
   στη γραμμή 1 αυτό είναι το πρώτο μου πρόγραμμα
   στη γραμμή 2 το όνομά σας
   στη γραμμή 3 ημερομηνία
2. Να γράψετε πρόγραμμα στο οποίο να γίνεται εισαγωγή των δεδομένων μήκος, πλάτος, ύψος έτσι, ώστε να υπολογίζονται και να τυπώνονται τα παρακάτω αποτελέσματα:

   το εμβαδόν της βάσης είναι .....τετρ.εκ.
   το εμβαδόν της έδρας με διαστάσεις .., ..
   είναι .....τετρ.εκ.

το εμβαδόν της έδρας με διαστάσεις .., ..
είναι .....τετρ.εκ.
ο όγκος του παραλληλεπιπέδου είναι .....κυβ. εκ.

3. Να γράψετε πρόγραμμα το οποίο να διαβάζει έναν αριθμό και να υπώνει το διπλάσιο και το τριπλάσιό του. Να γίνει η ίδια διαδικασία για τους 2 επόμενους απ΄ αυτόν αριθμούς. Η μορφή της εκτύπωσης να είναι σύμφωνα με το παρακάτω υπόδειγμα:

8 16 24
9 18 27
10 20 30

4. Να γράψετε πρόγραμμα στο οποίο να γίνεται εισαγωγή των ψήφων τριών ατόμων για το καθένα χωριστά. Να βρεθεί ο μέσος όρος των ψήφων και να τυπωθεί η διαφορά των ψήφων καθενός από το μέσο όρο. Οι αριθμοί κατά την εκτύπωση να συνοδεύονται από τις κατάλληλες επεξηγήσεις σύμφωνα με το υπόδειγμα.

Η μορφή της εκτύπωσης θα είναι:
Ο Γιώργος πήρε ..... ψήφους
Ο Χάρης πήρε ..... ψήφους
Ο Χρίστος πήρε ..... ψήφους
_ _
ο μέσος όρος των ψήφων είναι ..... ψήφοι
_ _
Η διαφορά από το μέσο όρο είναι:
για το Γιώργο ..... ψήφοι
για το Χάρη ..... ψήφοι
για το Χρίστο ..... ψήφοι

5. Να γράψετε πρόγραμμα που να διαβάζει το μήκος της ακτίνας ενός κύκλου και να τυπώνει τη διάμετρο, το μήκος και το εμβαδόν αυτού του κύκλου.

αποτελέσματα:
για κύκλο ακτίνας **** εκ.
η διάμετρος είναι ***** εκ.
το μήκος της περιφέρειας είναι ***** εκ.
το εμβαδόν του κύκλου είναι ******* τετρ.εκ.

6. Να γράψετε πρόγραμμα που να διαβάζει ένα τριψήφιο ακέραιο αριθμό και να τον τυπώνει ανάστροφα. Π.χ.

ο αριθμός είναι 123
ο ανάστροφος είναι 321

7. Να γράψετε το ίδιο πρόγραμμα για έναν τετραψήφιο αριθμό.
8. Να γράψετε πρόγραμμα που να διαβάζει έναν πραγματικό αριθμό και να τυπώνει τις 5 πρώτες δυνάμεις του και το άθροισμα των 3 πρώτων δυνάμεών του.

Τα αποτελέσματα να έχουν τη μορφή:
αριθμός :
1η δύναμη του αριθμού :
2η δύναμη του αριθμού :
3η δύναμη του αριθμού :
4η δύναμη του αριθμού :
5η δύναμη του αριθμού :

άθροισμα των δυνάμεων 1,2,3 :

9. Να γράψετε πρόγραμμα που να διαβάζει θερμοκρασία σε βαθμούς Fahrenheit και να τη μετατρέπει σε βαθμούς Κελσίου. Ο τύπος μετατροπής είναι:

$$\frac{(f-32)}{9}=\frac{c}{5}$$

10. Να γράψετε πρόγραμμα που να διαβάζει τους συντελεστές α,β, της πρωτοβάθμιας εξίσωσης αx+β=0 και να υπολογίζει τη ρίζα της.
(Υποτίθεται ότι α, β διάφορα του μηδενός).

11. Να γράψετε πρόγραμμα που να διαβάζει ημερομηνία με τη μορφή μέρα/μήνας/έτος και να την τυπώνει με τη μορφή έτος/μήνας/μέρα
αποτελέσματα:
διαβάστηκε η ημερομηνία ../../..
που μετατρέπεται στην ../../..

Δραστηριότητες

1. Να γράψετε 3 δικά σας προβλήματα όπου θα χρησιμοποιήσετε εντολές εισόδου/εξόδου και αντικατάστασης και θα κωδικοποιήσετε τον αλγόριθμο καθενός από αυτά

2. Δίνεται ως παράδειγμα η κωδικοποίηση της άσκησης 9 σε Qbasic

```
'άσκηση 9, Κεφάλαιο 9 ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ
'Μετατροπή θερμοκρασίας από Fahrenheit σε Κελσίου
CLS
INPUT "δώσε τη θερμοκρασία σε βαθμούς Fahrenheit ";
fa!
Ce!=(5*(fa-32)) / 9
PRINT  " η θερμοκρασία σε βαθμούς Κελσίου είναι :"'
ce!
END
```

Να γράψετε με τον editor, να **μεταφράσετε** και να εκτελέσετε χρησιμοποιώντας τις κατάλληλες εντολές τρεις από τις προηγούμενες ασκήσεις του κεφαλαίου 9 στο περιβάλλον της **Qbasic.**

ΚΕΦΑΛΑΙΟ 10
ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ ΚΑΙ ΑΠΟΦΑΣΕΩΝ

Σκοπός κεφαλαίου:
Να κατανοήσουμε τις δυνατότητες της αλλαγής της διαδοχικής εκτέλεσης των εντολών ενός προγράμματος ανάλογα με τα αποτελέσματα συνθηκών που περιλαμβάνονται στις ίδιες τις εντολές.

Ειδικοί σκοποί:
- ❑ Να κατανοήσουμε τη σύνταξη και τη λειτουργία της εντολής if σε όλες της δυνατές μορφές.
- ❑ Να κατανοήσουμε τη σύνταξη και τη λειτουργία της εντολής case καθώς και τον τρόπο με τον οποίο αυτή χρησιμοποιείται για την καλύτερη δόμηση του προγράμματος.

Αλλαγή σειράς εκτέλεσης των εντολών

Όπως είναι γνωστό, οι εντολές ενός προγράμματος εκτελούνται διαδοχικά, η μία μετά την άλλη. Αυτό δεν είναι πάντοτε επιθυμητό. Η σειρά εκτέλεσης των εντολών μπορεί να αλλάξει με τη χρήση των εντολών ελέγχου και επιλογής. Με τις εντολές αυτές επιτυγχάνεται η εκτέλεση ορισμένων εντολών υπό συνθήκες.

10.1. IF

***Μορφή*:** **if Λογική έκφραση**
then εντολή-1

Ενέργεια: Αν ή τιμή της λογικής έκφρασης είναι true (σωστό – αληθής), τότε εκτελείται ή εντολή που ακολουθεί το then.

***Μορφή*:** **if Λογική έκφραση**
then εντολή-1
else εντολή-2

Ενέργεια: : Αν ή τιμή της λογικής έκφρασης είναι true (σωστό – αληθής), τότε εκτελείται ή εντολή-1 που ακολουθεί το then, αλλιώς εκτελείται η εντολή-2 που ακολουθεί το else.

Παραδείγματα:

α.
```
if  wres  <= 40
    then  plir:=wr_apoz  * wres
    else  plir:=wr_apoz  ( 40+ (wres-40)*1.5);
```

Δηλαδή αν οι ώρες εργασίας δεν είναι περισσότερες από 40 πολλαπλασιάζουμε την ωριαία αποζημίωση επί τις ώρες και βρίσκουμε το πληρωτέο ποσό. Οι επί πλέον των 40 ωρών θεωρούνται υπερωριακές και πολλαπλασιάζονται με το συντελεστή 1.5. Μερικές φορές η εντολή if χρησιμοποιείται και για έλεγχο των δεδομένων. Για παράδειγμα, πριν γίνει διαίρεση πρέπει να γίνει έλεγχος μήπως ο διαιρέτης έχει τιμή μηδέν οπότε δεν είναι δυνατή η διαίρεση και τυπώνεται έτσι το κατάλληλο μήνυμα, όπως φαίνεται στο παράδειγμα που ακολουθεί.

β.
```
if diaireths <>0
    then  piliko:=diaireteos div diaireths
    else  writeln('ΛΑΘΟΣ διαίρεση με μηδέν δε γίνε-
                   ται');
```

Αν μετά το then ή το else ακολουθούν περισσότερες από μία εντολές, θεωρούμε ότι αποτελούν μία ακολουθία από εντολές. Τις εντολές αυτές βάζουμε μεταξύ των begin – end και το μεταφραστικό πρόγραμμα τις θεωρεί ως μία εντολή, **σύνθετη** εντολή **compound** statement. Το προηγούμενο παράδειγμα θα μπορούσε να έχει την εξής μορφή:

```
if diaireths <>0
then
 begin
  piliko:=diaireteos div diaireths´;
  writeln('διαίρεση δυνατή');
 end
 else
  begin
   writeln('ΛΑΘΟΣ  διαίρεση με μηδέν δε γίνεται');
   piliko:=maxint;
  end;
```

Η εντολή η οποία ακολουθεί το then ή το else μπορεί να είναι μια άλλη εντολή επιλογής (ένα άλλο if). Τότε λέμε ότι έχουμε **φωλιά** (**nest**) επιλογών ή φωλιά από (nested) if.

Παραδείγματα:

α.
```
if  a1
then  a2
else  if  'a3
       then a4
```

γ.
```
program exisosi_a_bathmou;
var
   a,b,x:real;
begin
  write('δώσε το a: ');
```

```
        else a5

β.

 if a=b
then write ('a=b')
else
 if a>b
 then
writeln('a>b')
     else
    writeln('a<b');
```

```
    readln(a);
    write('δώσε το b: ');
    readln(b);
    if a<>0
    then
      begin
      x:= a/b;
      writeln('x =', x:8:2);
      end
    else
      if b<>0
      then writeln('αδύνατη
      εξίσωση')
      else writeln('αόριστη
      εξίσωση');
end.
```

10.2. CASE

Μορφή: **case** έκφραση **of**

case label, case label ...: εντολή-1;

case label, case label ...: εντολή-2;

.

.

.

else εντολή-λ

end;

Ενέργεια: Η εντολή case προσφέρει δυνατότητα πολλαπλής επιλογής.

Επιλέγεται για εκτέλεση η εντολή-κ, όταν η έκφραση έχει ως τιμή μια ετικέτα (label) από τα στοιχεία της λίστας ετικετών της εντολής-κ.

Η **ετικέτα** (**case label**) μπορεί να είναι:

μια καθορισμένη τιμή (literal)

μια σταθερά με όνομα

οποιοσδήποτε διατεταγμένος τύπος (ακέραιος, χαρακτήρας κλπ)

Η **έκφραση** είναι ο επιλογέας (selector) της case.

Η λίστα ετικετών της case είναι μια λίστα διατεταγμένων τιμών του ίδιου τύπου με τον επιλογέα της case.

Αν η έκφραση δεν πάρει καμιά από τις αναφερόμενες τιμές, τότε εκτελείται η εντολή-λ, εφόσον έχει δηλωθεί μέσω της else. Αν σε μία επιλογή πρέπει να

εκτελεστούν περισσότερες εντολές, τότε χρησιμοποιούμε τη σύνθετη εντολή μέσω των begin-end.

Ακολουθούν χαρακτηριστικά παραδείγματα για την κατανόηση της case.

Μία πρόταση case μπορεί να είναι της μορφής:

α. case boolean έκφραση of
    true: εντολή1
    false: εντολή2
  end {case}

Εδώ επιλογέας είναι μια λογική έκφραση. Οι τιμές της μπορεί να είναι true ή false. Εκτελείται η εντολή που έχει ως ετικέτα την τιμή της λογικής έκφρασης, δηλαδή η εντολή1, αν η τιμή της λογικής έκφρασης είναι true ή η εντολή2, αν η τιμή της λογικής έκφρασης είναι false.

Μία τέτοια μορφή που ελέγχεται από μία λογική έκφραση μπορεί βέβαια να έχει το πολύ δύο δυνατές επιλογές. Η μορφή αυτή είναι ισοδύναμη της εντολής if.

Παραδείγματα:

α.

```
 case letter of
      'x'          : εντολή-1
      'l', 'm'     : εντολή-2
      's'          : εντολή-3
 end; {case}
εντολή4
```

Εδώ επιλογέας είναι ο διατεταγμένος τύπος γράμμα (letter). Οι ετικέτες είναι literal του ιδίου τύπου με τον επιλογέα της **case.** Υπάρχει μόνο μία λίστα με ετικέτες που αντιστοιχούν στην εντολή-2. Αν ο επιλογέας έχει ως τιμή μια από τις ετικέτες της λίστας, δηλαδή l ή m, τότε εκτελείται η εντολή-2.

β.

```
 case vathmologia  of
  'A','B'      : writeln ('πολύ καλά'); {εντολή-1}
  'C','D'      : writeln ('καλά');      {εντολή-2}
  'E','F','G'  : begin                   {εντολή-3}
                   writeln ('καλά);
                   new_test:=new-test+1;
                 end;
else writeln(' βαθμολογία εκτός ορίων')
 end; {case}
εντολή-4
```

Εδώ επιλογέας είναι **vathmologia** τύπου Char (χαρακτήρα), δηλαδή διατεταγμένου τύπου. Οι ετικέτες είναι literal του ιδίου τύπου με τον επιλογέα

της **case.** Υπάρχουν τρεις λίστες με ετικέτες που αντιστοιχούν σε κάθε μία από τις τρεις εντολές.. Η τρίτη εντολή είναι σύνθετη. Αν ο επιλογέας έχει ως τιμή μια από τις ετικέτες της λίστας, τότε εκτελείται η αντίστοιχη εντολή, αλλιώς τυπώνεται βαθμολογία εκτός ορίων.

γ.

```
var
     arxika_epilogis:char;
.........
case arxika_epilogis of
'Ε': writeln('ΕΙΣΑΓΩΓΗ ');
'Μ': writeln('ΜΕΤΑΒΟΛΗ');
'Δ': writeln('ΔΙΑΓΡΑΦΗ');
'Τ': writeln('ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ -  ΕΞΟΔΟΣ');
else writeln('Λάθος αρχικό επιλογής');
end{case}
```

δ.

```
var i:integer;
     case  i of
           1:  a:=a+1;
           2:  b:=b+1;
           3:  c:=c+1;
     end {case}
```

Η εντολή case επιτρέπει την απαρίθμηση διαφόρων εναλλακτικών λύσεων και την επιλογή μίας εξ αυτών για εκτέλεση.

Ανακεφαλαίωση

Κατανοήσαμε τη σύνταξη των εντολών επιλογής και τον τρόπο χειρισμού τους για τη λήψη απόφασης μέσα σε ένα πρόγραμμα Pascal. Κατανοήσαμε τις διάφορες μορφές σύνταξης της εντολής **if** και τον έλεγχο της συνθήκης που μπορεί να έχει μία ή δύο επιλογές. Κατανοήσαμε την ανάγκη για σύνθετες συνθήκες με περισσότερες επιλογές με τη χρήση εμφωλευμένων δομών if. Τέλος κατανοήσαμε τη χρησιμότητα της εντολής **case** η οποία βοηθάει τη δόμηση ενός προγράμματος και το καθιστά πιο ευανάγνωστο και κατανοητό αντικαθιστώντας πολλά if.

Ερωτήσεις

1. Πώς μπορεί να αλλάξει η σειρά εκτέλεσης των εντολών;
2. Πώς επιτυγχάνεται η εκτέλεση ορισμένων εντολών υπό συνθήκες;
3. Να εξηγήσετε την παρακάτω εντολή και να την τροποποιήσετε ώστε να περιλαμβάνει και την περίπτωση εκείνη στην οποία ο αριθμός είναι πε-

ριττός. Να δώσετε παραδείγματα και τα αποτελέσματα της εκτέλεσης των εντολών.

```
if arithm mod 2 = 0
then writeln(' ο αριθμός είναι άρτιος ')';
```

4. Να εξηγήσετε την παρακάτω εντολή και να δώσετε παραδείγματα και τα αποτελέσματα της εκτέλεσης της εντολής.

```
if diairetis <>0
then pililko:=diaireteos div diairetis
else writeln('διαίρεση με μηδέν δεν είναι επιτρεπτή');
```

5. Να χρησιμοποιήσετε την εντολή if για τον έλεγχο της ηλικίας ενός ψηφοφόρου και να δώσετε το κατάλληλο μήνυμα ανάλογα με την περίπτωση, για ηλικία > 65 , για ηλικία <18 και για 18<ηλικία<65

6. Να χρησιμοποιήσετε την εντολή case για τον έλεγχο της ηλικίας ενός ψηφοφόρου και να δώσετε το κατάλληλο μήνυμα ανάλογα με την περίπτωση, για ηλικία > 65, για ηλικία <18 και για 18<ηλικία<65

7. Να συγκρίνετε την εντολή case με την if και να δώσετε δύο παραδείγματα χρήσης της εντολής case.

8. Να γράψετε την επόμενη κωδικοποίηση χρησιμοποιώντας την εντολή case αντί της if

```
if  n=5
then   tm5:=tm5+1
else   if  n=15
          then  tm15:=tm15+1
          else  if n=25
                   then  tm25:=tm25+1;
```

9. Να γράψετε μία εντολή case η οποία ανάλογα με το χαρακτηρισμό της βαθμολογίας τυπώνει το αντίστοιχο μήνυμα. Αν ο χαρακτηρισμός=1 τύπωσε άριστα, 2 λίαν καλώς, 3 καλώς, 4 σχεδόν καλώς, 5 απορρίπτεται, αλλιώς λάθος χαρακτηρισμός. Να επαναλάβετε τα προηγούμενα με την εντολή if.

10. Να συμπληρώσετε με **Σωστό – Λάθος** τις παρακάτω προτάσεις

    i. Η έκφραση της εντολής case μπορεί να δώσει τιμές τύπου ακέραιου, πραγματικού ή λογικού. _____

    ii. Οι τιμές που παίρνουν οι ετικέτες στην εντολή case μπορεί να έχουν οποιαδήποτε σειρά, αλλά οι ετικέτες δεν επαναλαμβάνονται. _____

    iii. Όλες οι δυνατές τιμές της έκφρασης σε μια εντολή case πρέπει να περιλαμβάνονται στις λίστες τιμών των ετικετών. _____

Ασκήσεις

1. Να γράψετε πρόγραμμα που να ελέγχει το ποσό ανάληψης ενός καταθέτη σε μία Τράπεζα. Σε περίπτωση που η ανάληψη είναι μεγαλύτερη από τις καταθέσεις τυπώνει απαγορευτικό μήνυμα, αλλιώς τυπώνει το ποσό ανάληψης και το υπόλοιπο των καταθέσεων.

2. Να γράψετε πρόγραμμα που να διαβάζει 4 βαθμούς, να βρίσκει το μέσο όρο τους και, αν αυτός είναι μεγαλύτερος από 15, τυπώνει ΕΠΙΤΥΧΩΝ.

3. Η ωριαία αμοιβή εργαζομένου είναι 2000 δρχ. Αν οι ώρες εργασίας είναι περισσότερες από 18, παίρνει επιπλέον υπερωριακή αποζημίωση 1000 δρχ γιά κάθε υπερωριακή ώρα. Να γράψετε πρόγραμμα που να διαβάζει τις ώρες εργασίας και να υπολογίζει τις αποδοχές του εργαζομένου. Τα αποτελέσματα να ακολουθήσουν το υπόδειγμα:

   κανονική αμοιβή .....
   αμοιβή υπερωριών .....
   ---------- ----------
   συνολικές αποδοχές .....

   Υπόδειξη: Να γίνουν δύο τουλάχιστον εκτελέσεις του προγράμματος, μία με υπερωρίες και μία χωρίς.

4. Να μετατρέψετε τα προηγούμενα προγράμματα if -- then--else κάνοντας χρήση της εντολής case.

5. Να γράψετε πρόγραμμα που να διαβάζει το αρχικό ενός ονόματος και να τυπώνει το αντίστοιχο όνομα (ονόματα: Πέτρος, Χριστίνα, Νίκος, 'Αννα, Γιώργος).

6. Να γράψετε πρόγραμμα που να εμφανίζει menu με τις 4 πράξεις:

   1 πρόσθεση
   2 αφαίρεση
   3 πολλαπλασιασμός
   4 διαίρεση,

   να διαβάζει τον κωδικό της πράξης και τους δύο αριθμούς και να εμφανίζει το αποτέλεσμα.(Διαίρεση με μηδέν δε γίνεται).

7. Να γράψετε πρόγραμμα που να διαβάζει δύο ηλικίες και να δίνει τη διαφορά τους. Η εισαγωγή στοιχείων και τα αποτελέσματα να είναι σύμφωνα με το υπόδειγμα:

   Δεδομένα : 9 6
   Αποτελέσματα:
   Ο Νίκος είναι 9 ετών
   Ο Γιώργος είναι 6 ετών
   Ο Νίκος είναι 3 έτη μεγαλύτερος απ' το Γιώργο

   Να γίνουν άλλες δύο εκτελέσεις με δεδομένα 6, 9 και 9, 9.

8. Για να ψηφίσει ένας πολίτης, πρέπει να είναι τουλάχιστον 18 ετών. Αν όμως είναι άνω των 70, δεν υποχρεώνεται να ψηφίσει. Να γράψετε πρόγραμμα που να διαβάζει την ηλικία και να δίνει το κατάλληλο μήνυμα.

Υπόδειξη: Να χρησιμοποιήσετε το δυαδικό τελεστή and. Να γίνουν 3 διαφορετικές εκτελέσεις με αποτελέσματα:

Είναι – – χρονών υποχρεώνεται να ψηφίσει.
Είναι – – χρονών δεν υποχρεώνεται να ψηφίσει.
Είναι – – χρονών δεν μπορεί να ψηφίσει.

Δραστηριότητες:

Να κωδικοποιήσετε σε γλώσσα Pascal τους αλγόριθμους των ασκήσεων 3,4, και 5 του ΚΕΦΑΛΑΙΟΥ 4 **ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΩΝ**

Δίνεται ως παράδειγμα η κωδικοποίηση της άσκησης 3 σε Qbasic

```
'άσκηση 3, Κεφάλαιο 10 ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ ΚΑΙ
ΑΠΟΦΑΣΕΩΝ
'Υπολογίζεται η κανονική αμοιβή και προσθέτουμε
'και τις υπερωρίες, αν υπάρχουν.
CLS
'εισαγωγή δεδομένων
INPUT  "δώσε τις ώρες εργασίας ";  we
'υπολογισμοί
IF we > 18
THEN
   Yper= we - 18
   AmYper = Yper *1000
ELSE
   AmYper=0
END IF
KanoAm=we*2000
SynolAm= KanoAm+AmYper
'Εκτύπωση αποτελεσμάτων
PRINT   "Κανονική Αμοιβή:";   TAB(20); KanoAm
PRINT   "Αμοιβή Υπερωριών:"; TAB(20); AmYper
PRINT   "                   "; TAB(20);
PRINT   "Συνολική Αμοιβή:";   TAB(20); SynolAm
END
```

Να γράψετε με τον editor, να **μεταφράσετε** και να εκτελέσετε τρεις από τις προηγούμενες ασκήσεις του κεφαλαίου 10 στο περιβάλλον της **Qbasic.** χρησιμοποιώντας τις κατάλληλες εντολές

ΚΕΦΑΛΑΙΟ 11
ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

Σκοπός κεφαλαίου:
Να κατανοήσουμε τη χρησιμότητα της δομής επανάληψης, που παρέχει ο προγραμματισμός.

Ειδικοί σκοποί:
- ❑ Να κατανοήσουμε τη λειτουργία της εντολής **while** η οποία είναι μία από τις τρεις βασικές δομές του δομημένου προγραμματισμού.
- ❑ Να κατανοήσουμε τη λειτουργία της εντολής **repeat** και πώς αυτή διαφοροποιείται σε σχέση με την while.
- ❑ Να κατανοήσουμε τη λειτουργία της εντολής **for** και πώς αυτή διαφοροποιείται από τις δύο προηγούμενες δομές επανάληψης.

11.1. Η ΕΝΝΟΙΑ ΤΗΣ ΕΠΑΝΑΛΗΨΗΣ

Συχνά, ορισμένοι υπολογισμοί σε ένα πρόγραμμα είναι αναγκαίο να εκτελούνται περισσότερες από μία φορές. Υπάρχουν δύο τύποι επαναλήψεων:

- Οι προκαθορισμένοι, όπου το πλήθος των επαναλήψεων είναι δεδομένο πριν αρχίσουν οι επαναλήψεις.
- Οι μη προκαθορισμένοι ή απροσδιόριστοι, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος επανάληψης.

11.2. WHILE

Η εντολή while χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων, όπου υπάρχει περίπτωση να μην εκτελεστούν οι επαναλήψεις και όπου ο έλεγχος γίνεται πριν από την εκτέλεση των εντολών επανάληψης.

Μορφή: **while** boolean expression **do** statement
εφόσον λογική έκφραση **εκτέλεσε** εντολή

Ενέργεια: Κάθε φορά, ακόμη και την πρώτη, πριν εκτελεστεί η εντολή που αποτελεί το σώμα της επανάληψης (το statement μετά το do), εξετάζεται η λογική έκφραση η οποία ακολουθεί τη λέξη while και, μόνο σε περίπτωση που δίνει τιμή **true**, εκτελείται η προς επανάληψη εντολή. Αν δώσει τιμή

false, η εντολή που αποτελεί το σώμα της επανάληψης δεν εκτελείται και ο έλεγχος από το while περνάει στην επόμενη εντολή. Το σώμα της επανάληψης είναι μία εντολή. Σε περίπτωση που θέλουμε να εκτελέσουμε περισσότερες από μία εντολές τότε αυτές θα πρέπει να αποτελέσουν μία σύνθετη εντολή με τη χρήση των **begin - end**.

Υπάρχουν δύο τύποι επαναλήψεων. Ο ένας τύπος χρησιμοποιεί κάποιο ***μετρητή επαναλήψεων (count-controlled loop)***. Ο άλλος τύπος χρησιμοποιεί κάποιο γεγονός και η επανάληψη συνεχίζεται μέχρι να συμβεί το γεγονός στη διάρκεια της επανάληψης (***event-controlled loop***).

Στο παράδειγμα που ακολουθεί χρησιμοποιείται ένας μετρητής για τον έλεγχο των επαναλήψεων. Ο μετρητής παίρνει αρχική τιμή έξω από τον κύκλο των επαναλήψεων, και αυξάνει κατά 1 στην τελευταία εντολή μέσα στον κύκλο των επαναλήψεων έτσι ώστε να μπορεί να γίνει ο έλεγχος και να σταματήσουν οι επαναλήψεις, όταν η λογική έκφραση πάρει τιμή ***false*** δηλαδή, όταν ο μετρητής γίνει ίσος με 101.

Παράδειγμα με μετρητή

Για τον έλεγχο των επαναλήψεων (count-controlled loop).

```
metritis:=1;      {αρχική τιμή στη μεταβλητή ελέγχου}
while count <= 100 do   {έλεγχος της επανάληψης}
begin
   **
   **                   {εντολές επανάληψης}
                        {εντολές επανάληψης}
   metritis:=metritis+1;   {αύξηση του μετρητή}
end
```

Παράδειγμα με μεταβλητή

Για τον έλεγχο των επαναλήψεων (event-controlled loop).

Για κάποια επεξεργασία χρειαζόμαστε ημερομηνίες μήνα και ημέρα. Διαβάζουμε την πρώτη ημερομηνία και για τον έλεγχο χρησιμοποιούμε την ίδια μεταβλητή. Η επανάληψη θα σταματήσει όταν η μεταβλητή πάρει τιμή μη παραδεκτή πχ. 31 Φεβρουαρίου. Όταν συμβεί το γεγονός αυτό η επανάληψη θα σταματήσει.

```
read(mina,imera);                  {αρχική ημερομηνία}
while not ((month=2) and (day=31)) do {έλεγχος
                                       επανάληψης}
   begin
     *
     *
      read(mina,imera);            {επεξεργασία}
   end                             {επόμενη ημερομηνία}
```

Το παράδειγμα που ακολουθεί είναι ένα πλήρες πρόγραμμα όπου εφαρμόζονται όσα αναφέραμε παραπάνω. Η επανάληψη των εντολών σταματάει όταν συμβεί το γεγονός " το άθροισμα να γίνει μεγαλύτερο από το δεδομένο όριο".

Παράδειγμα:

```
program sum1;
{Ανάγνωση αριθμών μέχρις ότου το άθροισμά τους γίνει
μεγαλύτερο από ένα δεδομένο αριθμό}
var
orio,ar,sum :real;
begin
     read(orio);
     sum:=0;
     while sum <= orio do
       begin
        readln(ar);
        sum:=sum+ar;
       end;
     writeln(sum:10:4);
end.
```

11.3. REPEAT - UNTIL

Η εντολή repeat χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων. Ο έλεγχος για την επανάληψη γίνεται στο τέλος του κύκλου (**loop**) των εντολών που επαναλαμβάνονται. Όπως φαίνεται και από τη μορφή της εντολής που ακολουθεί

Μορφή: **repeat**
εντολή [;
εντολή ;
.
.
.....]
until (boolean expression);

- Οι ορθογώνιες παρενθέσεις δηλώνουν ότι η δεύτερη καθώς και οι επόμενες εντολές επανάληψης μπορεί να μην υπάρχουν.
- Υπάρχει πάντοτε μία τουλάχιστον εντολή επανάληψης.
- Δεν απαιτούνται begin, end γιατί αντί αυτών υπάρχουν οι λέξεις **repeat**, **until**

Επειδή ο έλεγχος επανάληψης γίνεται στο τέλος, οι εντολές επανάληψης εκτελούνται τουλάχιστον μια φορά ακόμη και αν η λογική έκφραση είχε εξ αρχής την τιμή true.

Ενέργεια: Αφού εκτελεστούν για πρώτη φορά οι εντολές που αποτελούν το σώμα της επανάληψης (μεταξύ των repeat και until), εξετάζεται η λογική έκφραση η οποία ακολουθεί την λέξη until, και μόνο σε περίπτωση που δίνει τιμή false εκτελούνται και πάλι οι εντολές επανάληψης. Αν δώσει τιμή true, ο έλεγχος από το repeat, περνάει στην επόμενη του repeat-until εντολή. Όπως

φαίνεται από τον ορισμό της repeat οι εντολές εκτελούνται απαραίτητα τουλάχιστον μία φορά ακόμη και όταν η λογική τιμή είναι αληθής σε αντίθεση με την while, που δεν εκτελούνται ποτέ, αν η λογική έκφραση έχει τιμή false.

Ακολουθεί μια σύγκριση των εντολών repeat until και while do πρώτα για επαναλήψεις ελεγχόμενες από μετρητή και στη συνέχεια για επαναλήψεις ελεγχόμενες από κάποιο γεγονός.

Παράδειγμα με μετρητή

Για τον έλεγχο των επαναλήψεων (count-controlled loop).

```
metritis:=1;
while metritis <= 100 do
begin
   εντολή1
   εντολή2
      ...
    metritis:=metritis+1;

end
```

```
metritis:=1;

repeat
εντολή1
   εντολή2
      ...
metritis:=metritis+1
until metritis > 100
```

Στο **while** ο έλεγχος εκτελείται πριν από τον κύκλο των εντολών επανάληψης.

Στο **repeat** η επανάληψη των εντολών συνεχίζεται όσο η λογική έκφραση έχει τιμή false, ενώ στο **while** η επανάληψη των εντολών συνεχίζεται όσο η λογική έκφραση έχει τιμή true..

Παραδείγματα:

```
program sum2;
{Ανάγνωση αριθμών μέχρις ότου το
άθροισμά τους γίνει μεγαλύτερο
από ένα δεδομένο αριθμό}
var
orio,ar,sum :real;
begin
   read(orio);
   sum:=0;
   repeat
      readln(ar);
      sum:=sum+ar;
   until  sum > orio
   writeln(sum:10:4);
end.
```

```
Program sum2;
{Ανάγνωση αριθμών μέχρις ότου το
άθροισμά τους γίνει μεγαλύτερο
από ένα δεδομένο αριθμό}


var
orio,ar,sum :real;

begin
   read(orio);
   sum:=0;
   while sum <= orio do
      begin
         readln(ar);
         sum:=sum+ar;
      end;
   writeln(sum:10:4);
end.
```

Στο προηγούμενο παράδειγμα και στο repeat και στο while η επανάληψη σταματάει, όταν το άθροισμα ***(sum)*** γίνει μεγαλύτερο από το όριο ***(orio)*** Οι λογικές εκφράσεις είναι συμπληρώματα η μία της άλλης (αντίθετες) (***sum > orio*** και ***sum*** <= orio).

Εδώ πρέπει να τονιστεί ότι το **while** χρησιμοποιεί τη λογική έκφραση για να συνεχίσει την επανάληψη, ενώ το **repeat/until** για να τη σταματήσει.

Επιλέγουμε το repeat/until όταν οι εντολές επανάληψης εκτελούνται οπωσδήποτε τουλάχιστον μια φορά αλλιώς επιλέγουμε το while.

Αν έχουμε τη δυνατότητα να χρησιμοποιήσουμε και τα δύο πρέπει να επιλέξουμε εκείνο που αντιπροσωπεύει σημασιολογικά το πρόβλημα. Αν το πρόβλημα αναφέρεται στη συνέχεια της επανάληψης καλύτερα είναι να χρησιμοποιήσουμε το while. Αν το πρόβλημα αναφέρεται στη διακοπή της επανάληψης καλύτερα είναι να χρησιμοποιήσουμε το repeat/until. Αν υπάρχει αμφιβολία καλύτερα να χρησιμοποιούμε το While.

11.4. FOR

Η εντολή **for** σχεδιάστηκε για την απλοποίηση του ελέγχου των επαναλήψεων με μετρητή (**count-controlled loop**).

Η εντολή **for** metritis **:=** 1 **to** n **do**
statement; ***σημαίνει:***

- βάλε 1 στη μεταβλητή ελέγχου (metritis).
- αν το n είναι μικρότερο από το 1 τότε δεν εκτελείται ο κύκλος των εντολών επανάληψης

αλλιώς εκτελείται ο κύκλος των εντολών επανάληψης

και ο μετρητής αυξάνεται κατά 1.

- ο κύκλος επαναλαμβάνεται μέχρι ο μετρητής να γίνει μεγαλύτερος από n

Η προηγούμενη εντολή **for** εμφανίζεται μαζί με την ισοδύναμή της while στον πίνακα που ακολουθεί.

```
for metritis := 1 to n do
    statement;
```

```
metritis:=1;
while metritis <= n do
  begin
    statement;
    i:=i+1;
  end;
```

Υπάρχουν δύο μορφές της εντολής αυτής στην Pascal, μία που η μεταβλητή αυξάνεται (με τη χρήση του **to**) και μία που η μεταβλητή ελαττώνεται (με τη χρήση του **downto**).

Μορφή:

for variable identifier: = expression **to** expression **do**
statement;

για μεταβλητή ελέγχου από αρχική τιμή **μέχρι** τελική τιμή **εκτέλεσε** εντολή

Ενέργεια: Η **εντολή** που ακολουθεί το do, που μπορεί να είναι **απλή** ή **σύνθετη** εντολή, εκτελείται μέχρις ότου η μεταβλητή ελέγχου αποκτήσει τιμή ίση με την τελική τιμή. Η αύξηση της μεταβλητής ελέγχου γίνεται με βήμα τη μονάδα (1). Αν η αρχική τιμή είναι μεγαλύτερη από την τελική τιμή, τότε οι επαναλαμβανόμενες εντολές που ακολουθούν το do δεν εκτελούνται.

Αν η μεταβλητή ελέγχου αρχίζει από τη μεγαλύτερη τιμή μέχρι τη μικρότερη, τότε η μορφή της εντολής είναι η παρακάτω:

for variable identifier: = expression **downto** expression **do**
statement;

για μεταβλητή ελέγχου από αρχική τιμή **μέχρι** τελική τιμή **εκτέλεσε** εντολή

Ενέργεια: Η εντολή που ακολουθεί το do εκτελείται, μέχρις ότου η μεταβλητή ελέγχου αποκτήσει τιμή μικρότερη από την τελική τιμή. Η ελάττωση της μεταβλητής ελέγχου γίνεται με βήμα 1. Αν η αρχική τιμή είναι μικρότερη από την τελική τιμή, τότε οι εντολές που ακολουθούν το do δεν εκτελούνται.

Στον πίνακα που ακολουθεί ο αριθμός των επαναλήψεων είναι ίδιος και για τις δύο μορφές της εντολής for

```
for metritis := katwtero_orio to
                anwtero_orio do
   statement;
```

```
for metritis := anwtero_orio
          downto katwtero _orio do
   statement;
```

Η αρχική και η τελική τιμή του κύκλου for μπορεί να είναι μια έκφραση οποιουδήποτε διατεταγμένου τύπου. Έτσι, εκτός από ακέραιες τιμές μπορούμε να χρησιμοποιήσουμε μεταβλητές με τιμές τύπου χαρακτήρα char ή λογικού τύπου (boolean).

Με το παράδειγμα που ακολουθεί τυπώνουμε το Αγγλικό αλφάβητο.

```
for letter:= 'A ' to 'Z' do
    write(letter);
```

Η εντολή for όπως και άλλες μορφές επανάληψης μπορεί να είναι και σε φωλιά -nested (η μία μέσα στην άλλη). Στον πίνακα που ακολουθεί υπάρχει ένα παράδειγμα φωλιάς for και τα αποτελέσματά του.

Παράδειγμα:

```
for teleytaio_gramma:='A' to 'G' do
   begin
     for typwse_gramma:= 'A ' to
          teleytaio_gramma ' do
            write(letter);
   end;
```

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
ABCDEFG
```

Η εντολή for είναι πολύ εύχρηστη. Πρέπει να θυμόμαστε ότι έχει σχεδιαστεί για επαναλήψεις ελεγχόμενες με μετρητή, για τον οποίο γνωρίζουμε την αρχική και την τελική τιμή. Για αποτελεσματική χρήση της εντολής:

- Η μεταβλητή ελέγχου δεν πρέπει να αλλάζει τιμές μέσα στον κύκλο της επανάληψης. Μπορεί να εμφανίζεται σε μία έκφραση, αλλά όχι στο αριστερό μέλος της εντολής αντικατάστασης, οπότε αλλάζει η τιμή της
- Η μεταβλητή ελέγχου μεταβάλλεται κατά βήματα παίρνοντας την επόμενη ή την προηγούμενη τιμή από το πεδίο τιμών της. Αν η μεταβλητή είναι ακέραιος, τότε το βήμα είναι 1. Αν χρειαζόμαστε άλλο βήμα, επιλέγουμε την εντολή while.
- Μετά το τέλος της επανάληψης η μεταβλητή ελέγχου έχει απροσδιόριστη τιμή. Αν τη χρησιμοποιήσουμε ή θα πάρουμε μήνυμα λάθους ή λάθος αποτελέσματα ανάλογα με το μεταφραστικό πρόγραμμα (compiler) που χρησιμοποιούμε.
- Ο κύκλος επανάληψης εκτελείται με τη μεταβλητή ελέγχου να παίρνει τιμές την αρχική, την τελική και τα ενδιάμεσα βήματα. Αν η αρχική τιμή είναι ίση με την τελική, ο κύκλος εκτελείται μια μόνο φορά. Στην περίπτωση το αν η αρχική τιμή είναι μεγαλύτερη από την τελική, ο κύκλος δεν εκτελείται. Στην περίπτωση downto αν η αρχική τιμή είναι μικρότερη από την τελική τιμή, ο κύκλος δεν εκτελείται.

Παράδειγμα

```
program sum3;
use wincrt;
{Ανάγνωση αριθμών και υπολογισμός του αθροίσματός
τους μέχρις ότου το πλήθος τους γίνει μεγαλύτερο από
ένα δεδομένο αριθμό}
var
ar,sum :real;
orio:integer;
begin
    read(orio);
    sum:=0;
    for k:=1 to orio do
    begin
    readln(ar);
    sum:=sum+ar;
    end;
    writeln(sum:10:4);
end.
```

Ανακεφαλαίωση

Κατανοήσαμε τη σύνταξη και τον τρόπο χειρισμού των εντολών επανάληψης. Διακρίναμε τρεις διαφορετικές εντολές επανάληψης και κατανοήσαμε τις διαφορές μεταξύ τους και την ανάγκη ύπαρξη τους μέσα από χαρακτη-

ριστικά παραδείγματα. Οι εντολές **while** και **repeat** χρησιμοποιούνται για μη προκαθορισμένο αριθμό επαναλήψεων και στις δύο ο έλεγχος για την επανάληψη γίνεται μέσα από μία συνθήκη. Στη περίπτωση της **while** οι εντολές επανάληψης εκτελούνται εφόσον το αποτέλεσμα της συνθήκης είναι αληθές ενώ στη **repeat** οι εντολές επανάληψης εκτελούνται όσο το αποτέλεσμα της συνθήκης είναι ψευδές. Σημειώνεται ότι στη **repeat** οι εντολές θα εκτελεστούν τουλάχιστον μία φορά διότι ο έλεγχος της συνθήκης γίνεται μετά το σώμα των εντολών επανάληψης. Η εντολής **for** χρησιμοποιείται για προκαθορισμένο αριθμό επαναλήψεων.

Ερωτήσεις

1. Να δώσετε τη μορφή της εντολής επανάληψης while, να εξηγήσετε τη λειτουργία της και να δώσετε δύο δικά σας παραδείγματα.
2. Να δώσετε τη μορφή της εντολής επανάληψης repeat until, να εξηγήσετε τη λειτουργία της και να δώσετε δύο δικά σας παραδείγματα.
3. Να δώσετε τη μορφή της εντολής επανάληψης for, να εξηγήσετε τη λειτουργία της και να δώσετε δύο δικά σας παραδείγματα.
4. Να συγκρίνετε τις εντολές επανάληψης και να αναφέρετε ποιες είναι οι ομοιότητες και ποιες οι διαφορές τους.
5. Να δώσετε τα αποτελέσματα από την εκτέλεση των παρακάτω εντολών

```
arithmos:=1;
  while arithmos < 11 do
  begin
    arithmos:=arithmos+1;
    write(arithmos:5);
  end;
```

6. Να αλλάξετε τη σειρά των εντολών της προηγούμενης ερώτησης ώστε το αποτέλεσμα να είναι η εκτύπωση των αριθμών 1 έως και 10.
7. Να χρησιμοποιήσετε τις εντολές επανάληψης repeat until και for ώστε να έχετε τα ίδια αποτελέσματα με την ερώτηση 6.
8. Πόσες επαναλήψεις θα έχουμε από την εκτέλεση των παρακάτω εντολών;

```
telos:= false;
  while not telos  do
  begin
    arithmos:=arithmos+2;
    if arithmos > 100
    then telos :=true;
  end;
```

9. Να χρησιμοποιήσετε τις εντολές επανάληψης repeat until και for, ώστε να έχετε τον ίδιο αριθμό επαναλήψεων με την ερώτηση 8.

10. Να δώσετε τα αποτελέσματα από την εκτέλεση των παρακάτω εντολών και να εξηγήσετε τη διαφορά που υπάρχει στη χρήση των μεταβλητών athroisma και metritis

```
athroisma:=0;
metritis :=0;
while metritis < 10  do
begin
   athroisma:=athroisma + metritis;
   writeln(metritis:10,athroisma:10);
   metritis:= metritis + 1;
end;
```

Να τροποποιήσετε τον προηγούμενο κώδικα ώστε, να δίνει τα ίδια αποτελέσματα με τη χρήση των εντολών repeat until και for

Ασκήσεις

1. Να γράψετε ένα πρόγραμμα που να υπολογίζει τη μικρότερη δύναμη του 2 που είναι μεγαλύτερη από ένα δεδομένο αριθμό και να τυπώνει ποια είναι η δύναμη αυτή.

   Τα αποτελέσματα να δοθούν με τη μορφή:
   Η μικρότερη δύναμη του 2
   μεγαλύτερη από τον αριθμό .....
   είναι .....
   δηλ. είναι το 2 στην ..... δύναμη

Υπόδειξη: Η ασκηση να λυθεί με δύο τρόπους (με χρήση WHILE και REPEAT-UNTIL).

2. Να γράψετε πρόγραμμα που να υπολογίζει το μικρότερο αριθμό Fibonacci που δεν είναι μικρότερος από ένα προκαθορισμένο όριο. Κάθε αριθμός της σειράς Fibonacci, εκτός από τους δύο πρώτους που είναι 0 και 1, σχηματίζεται από το άθροισμα των δύο προηγουμένων του.

Υποδείξεις:
Να χρησιμοποιηθεί η εντολή while.
Τα αποτελέσματα να δοθούν με τη μορφή:
αποτελέσματα:
ο μικρότερος αριθμός Fibonacci
που δεν είναι μικρότερος από .....
είναι ο αριθμός .....

3. Να γράψετε πρόγραμμα που να διαβάζει:
   α. το πλήθος των εργατών μίας εταιρείας
   β. πόσες είναι οι υποχρεωτικές ώρες εργασίας (κάθε επιπλέον ώρα είναι υπερωρία)

γ. την ωριαία αμοιβή

δ. την επιπλέον αμοιβή γιά κάθε υπερωριακή ώρα

ε. τις ώρες εργασίας κάθε εργάτη

και να υπολογίζει την αποζημίωση κάθε εργάτη.

Υποδείξεις: Να χρησιμοποιηθεί η εντολή while. Τα αποτελέσματα να δοθούν με τη μορφή:

πλήθος εργατών: - -
υποχρεωτικές ώρες εργασίας: - -
ωριαία αποζημίωση : - -
πρόσθετη υπερωριακή αποζημ: - -
αποδοχές εργατών

Η εκτύπωση να γίνει με τη μορφή:

α/α	ώρες εργ.	αποζημ.	ώρες υπερ.	υπερ. αποζ.	σύνολο
1					
2					
...	...	.....	...	.....	.....

Να δοθεί δυνατότητα να σταματήσει η επανάληψη, σε περίπτωση που θα δοθεί αριθμός αρνητικός για ώρες εργασίας ενός εργάτη.

4. Να τροποποιηθεί η προηγούμενη άσκηση, ώστε το πρόγραμμα να λειτουργεί και με την εντολή repeat.
5. Να γράψετε πρόγραμμα που να διαβάζει το πλήθος των μαθητών και το βαθμό του καθενός και να τυπώνει ένα χαρακτηρισμό ανάλογα με το βαθμό του.

   (9 < βαθμός < = 12 μέτρια
   12 < βαθμός < = 15 καλά
   15 < βαθμός < = 18 πολύ καλά
   18 < βαθμός < = 20 άριστα)

   Να χρησιμοποιήσετε την εντολή repeat και δυαδικούς τελεστές (and...). Στο τέλος να τυπωθεί το πλήθος κάθε κατηγορίας, δηλ. άριστα, πολύ καλά κλπ. καθώς και το σύνολο όλων των μαθητών.
6. Να γράψετε πρόγραμμα που να διαβάζει τις 4 πλευρές και μία γωνία ενός τετραπλεύρου και να υπολογίζει το είδος του (τετράγωνο, ρόμβο, ορθογώνιο, παραλληλόγραμμο, τυχαίο). Να χρησιμοποιήσετε την εντολή repeat και δυαδικούς τελεστές.

Δραστηριότητες

1. Να κωδικοποιήσετε σε γλώσσα Pascal τους αλγόριθμους των ασκήσεων 2,3 και 4 του ΚΕΦΑΛΑΙΟΥ 5 ΣΤΟΙΧΕΙΑ ΔΟΜΗΜΕΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ.

2. Δίνεται ως παράδειγμα η κωδικοποίηση της άσκησης 2 σε Qbasic

```
'άσκηση 2, Κεφάλαιο 11 ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ
'Υπολογίζεται ο μικρότερος αριθμός Fibonacci που
'δεν είναι μικρότερος από ένα προκαθορισμένο όριο.
CLS
'εισαγωγή δεδομένων
INPUT "δώσε το όριο για τη σειρά Fibonacci
:"''Oriof
'υπολογισμοί
fProPro=0
fPro   =1
fi=fProPro+fPro
DO
   FProPro=fPro
   fPro=fi
   fi=fProPro+fPro
LOOP WHILE (fi < OrioF)
'Εκτύωση αποτελεσμάτων
PRINT  "Αποτέλεσμα :";  TAB(20); fi
END
```

Να γράψετε με τον editor, να **μεταφράσετε** και να εκτελέσετε τρεις από τις προηγούμενες ασκήσεις του κεφαλαίου 11 στο περιβάλλον της **Qbasic** χρησιμοποιώντας τις κατάλληλες εντολές.

ΚΕΦΑΛΑΙΟ 12
ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Σκοπός κεφαλαίου:
Να κατανοήσουμε την υλοποίηση του τμηματικού προγραμματισμού με τη χρήση υποπρογραμμάτων (διαδικασιών και συναρτήσεων) που ορίζονται από τους χρήστες.

Ειδικοί σκοποί:
- ❑ Να κατανοήσουμε την έννοια του υποπρογράμματος
- ❑ Να κατανοήσουμε τον τρόπο δημιουργίας διαδικασιών.
- ❑ Να κατανοήσουμε τον τρόπο δημιουργίας συναρτήσεων.
- ❑ Να κατανοήσουμε τις έννοιες τοπικές και σφαιρικές μεταβλητές.
- ❑ Να κατανοήσουμε την είσοδο τιμών σε ένα υποπρόγραμμα με τη βοήθεια παραμέτρων.
- ❑ Να κατανοήσουμε την έξοδο τιμών από ένα υποπρόγραμμα.
- ❑ Να κατανοήσουμε τη διαφορά μεταξύ διαδικασιών και συναρτήσεων.

12.1. ΔΙΑΔΙΚΑΣΙΕΣ

12.1.1. Διαδικασίες οριζόμενες από τον χρήστη

Μία **διαδικασία (procedure)** είναι ένα υποπρόγραμμα και έχει την ίδια μορφή με ένα πρόγραμμα, με τη διαφορά ότι αντί της λέξης **program** στην πρώτη γραμμή, έχουμε τη λέξη **procedure** και μετά το τελευταίο **end** έχουμε "**;**" αντί "**.**".

Η διαδικασία είναι ένα τμήμα του προγράμματος. Η απόφαση αν θα παραμείνει τμήμα του προγράμματος ή αν θα κωδικοποιηθεί ως διαδικασία εξαρτάται από το αν αυτό διευκολύνει στην όλη κατανόηση του προγράμματος. Η απόφαση αυτή μπορεί να επηρεάζει τη συντήρηση του προγράμματος, επειδή μπορούμε ευκολότερα να αλλάξουμε και να ελέγξουμε τη διαδικασία, όχι όμως και τη λειτουργία του προγράμματος γενικά.

Παραδείγματα διαδικασιών:

ΔΙΑΔΙΚΑΣΙΑ χωρίς παραμέτρους	ΠΡΟΓΡΑΜΜΑ ίδιο με τη διαδικασία

Procedure athroisma;
var num, sum,i,m :**integer**;
begin
sum:=0;
write('αριθμός προσθετέων=');
readln(m);
for i:=1 **to** m **do**
begin
write('αριθμός=');
readln(num);
sum:=sum+num;
end;
writeln('ΑΘΡΟΙΣΜΑ=',sum);
end;
	Program athroisma;
	Var num, sum,i,m :**integer**;
	Begin
	sum:=0;
	write('αριθμός προσθετέων=');
	readln(m);
	for i:=1 **to** m **do**
	begin
	write('αριθμός=');
	readln(num);
	sum:=sum+num;
	end;
	writeln('ΑΘΡΟΙΣΜΑ=',sum);
	end.

ΚΛΗΣΗ διαδικασίας ***χωρίς παραμέτρους***	ΚΛΗΣΗ διαδικασίας ***με παραμέτρους***

Program test_subroutines1;
Uses wincrt;
 	
procedure athroisma;
var num, sum,i,m :**integer**;
 	
begin
sum:=0;
write('αριθμός προσθετέων=');
readln(m);
for i:=1 **to** m **do**
begin
write('αριθμός=');
readln(num);
sum:=sum+num;
end;
writeln('ΑΘΡΟΙΣΜΑ=',sum);
end;
 	
begin {*κλήση διαδικασίας *}
athroisma;
end.
	Program test_subroutines2;
	Uses wincrt;
	Var n :integer; athr :**real**;
	
	Procedure athroisma(m:**integer**;
	var sum :real);
	{υπολογίζει το άθροισμα m αριθμών. Το αποτέλεσμα στη μεταβλητή sum }
	var num, i :**integer**;
	begin
	sum:=0;
	for i:=1 **to** m **do**
	begin
	write('αριθμός=');
	readln(num);
	sum:=sum+num;
	end
	end;
	begin
	write('αριθμός προσθετέων=');
	readln(n);
	athroisma(n, athr);
	{*διαδικασίας με παραμέτρους*}
	writeln('άθροισμα=',athr:10:2);
	end.

Η πρώτη γραμμή (επικεφαλίδα) ενός προγράμματος ονομάζει το πρόγραμμα και προσδιορίζει τα αρχεία, από όπου παίρνουμε τα δεδομένα, και τα αρχεία, όπου γράφονται τα αποτελέσματα. Η πρώτη γραμμή μιας διαδικα-

σίας ονομάζει τη διαδικασία και προσδιορίζει τις μεταβλητές (παραμέτρους) που χρησιμοποιούνται για είσοδο των δεδομένων και έξοδο των αποτελεσμάτων. Οι παράμετροι αυτές λέγονται **τυπικές (formal)**. Οι **πραγματικές (actual)** αντικαθιστούν τις τυπικές παραμέτρους κατά την κλήση της διαδικασίας.

Παραδείγματα:

procedure **ypologismos (var athr:integer; m:integer)**;	
athr, m	είναι τυπικές (formal) παράμετροι
καλείται ως	**ypologismos(sum,3)**
sum, 3	είναι πραγματικές (actual) παράμετροι

Η διαδικασία υπολογίζει το άθροισμα (athr) m το πλήθος αριθμών, η εκτέλεση γίνεται με την κλήση ypologismos(sum,3), οπότε υπολογίζεται το άθροισμα τριών (3) αριθμών και το αποτέλεσμα δίνεται στο sum.

Για να μπορεί μια διαδικασία να χρησιμοποιηθεί, πρέπει να έχει δηλωθεί στο τμήμα προγράμματος. Καλείται με το όνομά της και, εάν υπάρχουν **τυπικές παράμετροι (formal)**, τότε οι **πραγματικές παράμετροι (actual)** αντικαθιστούν τις τυπικές παραμέτρους. Στην περίπτωση αυτή έχει σημασία η σειρά των παραμέτρων. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που έχει την ίδια σειρά. Η σειρά των procedures είναι χωρίς σημασία. Όταν όμως μία διαδικασία καλεί μία άλλη διαδικασία, τότε η καλούμενη διαδικασία πρέπει να προηγείται.

Οι τυπικές παράμετροι διακρίνονται σε:

- παραμέτρους τιμών (value parameters)
- παραμέτρους μεταβλητών (variable parameters)

■ Παράμετροι Τιμών

Η δήλωση των **παραμέτρων τιμών (value parameters)** είναι όμοια με τη δήλωση των μεταβλητών, μόνο που παραλείπεται το var. Μία τέτοια παράμετρος είναι **τοπική (local)** για τη διαδικασία. Η αντίστοιχη πραγματική παράμετρος πρέπει να είναι ίδιου τύπου με την τυπική παράμετρο.

Η τυπική παράμετρος ως τοπική μεταβλητή παίρνει αρχική τιμή από την πραγματική παράμετρο, όταν καλείται η διαδικασία. Χώρος για την τυπική παράμετρο κρατείται κάθε φορά που ενεργοποιείται η διαδικασία και ο χώρος αυτός αποδεσμεύεται, μόλις τελειώσει η εκτέλεση της διαδικασίας.

Οι παράμετροι τιμών μεταφέρουν δεδομένα μέσα στη διαδικασία, γι' αυτό πολλές φορές ονομάζονται παράμετροι εισόδου. Μία παράμετρος τιμής ορίζει τοπικά το όνομα μιας περιοχής **τυπικής (formal)** παραμέτρου. Η περιοχή αυτή παίρνει τιμή από την **πραγματική (actual)** παράμετρο, όταν καλείται η διαδικασία.

Στο προηγούμενο παράδειγμα η διαδικασία athroisma έχει δύο τυπικές (formal) παραμέτρους, m, sum.. Η m είναι παράμετρος τιμής (value) και αναφέρεται χωρίς το var, ενώ η sum είναι, όπως θα εξηγήσουμε λεπτομερέστερα παρακάτω, παράμετρος μεταβλητής και αναφέρεται με το var. Η διαδικασία καλείται με το όνομά της που συνοδεύεται από μια λίστα με τις πραγματικές (actual) παραμέτρους

athroisma(n, athr);

Οι πραγματικές παράμετροι αντιστοιχούν μία προς μία με τις τυπικές. Έτσι η πραγματική παράμετρος τιμής n είναι αυτή που δίνει τιμή στην τυπική παράμετρο αξίας m. Η αλλαγή της τιμής της τυπικής παραμέτρου τιμής μέσα στη διαδικασία δεν επηρεάζει την τιμή της πραγματικής παραμέτρου (actual parameter). Αυτό φαίνεται στο παράδειγμα που ακολουθεί.

```
Program     test_subroutines3;
uses wincrt;
var
      n     :integer;
      athr : real;
 procedure athroisma(m:integer; var sum :real);
{Η υπορουτίνα υπολογίζει το άθροισμα m αριθμών.
  Το   αποτέλεσμα στη  μεταβλητή sum }
var
  num, i :integer;
begin
 sum:=0;
 writeln(m);
 m:=m+2;
 writeln(m);
 for i:=1 to m do
   begin
     write(i, ' ');
     write('αριθμός=');
     readln(num);
     sum:=sum+num;
   end ;
   writeln(m);
end;

begin
  write('αριθμός προσθετέων=');
  readln(n);
  writeln('n=',n);
  writeln('ΑΘΡΟΙΣΜΑ=',athr:10:2);
  athroisma(n, athr);
  writeln('ΑΘΡΟΙΣΜΑ=',athr:10:2);
  writeln('n=',n);
end.
```

```
ΕΚΤΕΛΕΣΗ
αριθμός προσθετέων=3
n=3

ΑΘΡΟΙΣΜΑ=0.00
3
5
1 αριθμός 10
2 αριθμός 20
3 αριθμός 30
4 αριθμός 40
5 αριθμός 50
5
ΑΘΡΟΙΣΜΑ=150.00
n=3
```

■ Παράμετροι Μεταβλητών

Μερικές φορές είναι αναγκαίο να εφοδιάσουμε τη διαδικασία και με παράμετρο η οποία να μεταφέρει μια τιμή στη διαδικασία, η δε διαδικασία να αλλάζει την τιμή της παραμέτρου και η αλλαγή αυτή να μεταφέρεται στην πραγματική παράμετρο. Αυτόν ακριβώς το ρόλο παίζει η **παράμετρος μεταβλητής (variable parameter)**. Στην περίπτωση αυτή, η τυπική παράμετρος μπορεί να παίρνει τιμή από την πραγματική παράμετρο αλλά οποιαδήποτε αλλαγή της τιμής της τυπικής παραμέτρου έχει ως αποτέλεσμα αλλαγή και της τιμής της πραγματικής παραμέτρου. Οι παράμετροι μεταβλητών μεταφέρουν πληροφορίες έξω από τη διαδικασία, για το λόγο αυτό ονομάζονται και παράμετροι εξόδου. Για να δηλωθεί σε μία διαδικασία, ότι μια παράμετρος είναι αυτού του τύπου, προηγείται η λέξη **var**.

Η παράμετρος μεταβλητής ορίζει το όνομα μιας περιοχής για χρήση από την πραγματική μεταβλητή. Η περιοχή αυτή παραμένει δεσμευμένη στη διάρκεια της εκτέλεσης του προγράμματος ενώ, στη διάρκεια εκτέλεσης της διαδικασίας, η τυπική παράμετρος αναφέρεται στην περιοχή της πραγματικής μεταβλητής η οποία είναι αυτή που δέχεται τα αποτελέσματα.

Παραδείγματα:

procedure **synolikes_kratiseis** (***var krariseis,foros:real***; ***akatharista:real);***
kratiseis, foros είναι ***παράμετροι μεταβλητών*** (variable parameter)
Akatharista είναι ***παράμετρος τιμής*** (value parameter)

12.2. ΣΥΝΑΡΤΗΣΕΙΣ

12.2.1. Συναρτήσεις οριζόμενες από τον χρήστη

Υπάρχουν περιπτώσεις που μια επεξεργασία την οποία προσδιορίζει ο χρήστης, εκφράζεται καλύτερα με τη μορφή **συνάρτησης (function)**. Αυτό συμβαίνει όταν ισχύουν οι παρακάτω προϋποθέσεις:

- Υπάρχει μόνο ένα αποτέλεσμα από τους υπολογισμούς που περιέχει η διαδικασία.
- Η διαδικασία δεν επιφέρει αλλαγές σε μη τοπικό επίπεδο.

Μία συνάρτηση της Pascal πλησιάζει περισσότερο το γνωστό μας από τα Μαθηματικά ορισμό συνάρτησης. Η συνάρτηση δίνει ως αποτέλεσμα μια τιμή η οποία μπορεί να έχει ένα οποιοδήποτε τύπο από τους επιτρεπτούς της Pascal. Για να μην παραβούμε τις παραπάνω προϋποθέσεις, πρέπει οι τυπικές παράμετροι να είναι παράμετροι τιμών.

Μία συνάρτηση έχει την ίδια μορφή με μια διαδικασία με τη διαφορά ότι αντί της λέξης procedure στην πρώτη γραμμή, έχουμε τη λέξη **function**. Η πρώτη γραμμή μιας συνάρτησης καθορίζει το όνομα της συνάρτησης και ορίζει τις μεταβλητές που ονομάζονται τυπικές παράμετροι και χρησιμοποιούνται για την εισαγωγή των δεδομένων στη συνάρτηση, οπότε οι πραγματικές παράμετροι αντικαθιστούν τις τυπικές παραμέτρους. Ακόμη η συνάρτηση δίνει πάντοτε ένα αποτέλεσμα που η διεύθυνσή του ορίζεται από το όνομα της συνάρτησης. Γι' αυτόν το λόγο, εκτός από το όνομα και τις παραμέτρους, στην πρώτη γραμμή δίνεται και ο τύπος της συνάρτησης π.χ. integer, real κλπ.

Για να μπορεί μια συνάρτηση να χρησιμοποιηθεί, πρέπει να έχει δηλωθεί στο τμήμα δηλώσεων του προγράμματος. Η σειρά των συναρτήσεων είναι χωρίς σημασία. Συνήθως γράφονται αλφαβητικά για εύκολη ανεύρεση. Εάν όμως μία συνάρτηση καλεί μία άλλη τότε ή καλούμενη πρέπει να προηγείται. Κάθε συνάρτηση καλείται με το όνομά της και, αν υπάρχουν παράμετροι, οι πραγματικές παράμετροι αντικαθιστούν τις τυπικές παραμέτρους. Στην περίπτωση αυτή έχει σημασία η σειρά των παραμέτρων. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που έχει την ίδια σειρά.

Οι συναρτήσεις και οι διαδικασίες ονομάζονται και υποπρογράμματα. Κάθε υποπρόγραμμα μπορεί να καλεί ένα οποιοδήποτε άλλο υποπρόγραμμα ή ακόμη και τον ίδιο τον εαυτόν του. Στην τελευταία περίπτωση έχουμε αναδρομική συνάρτηση ή αναδρομική διαδικασία.

Ακολουθούν παραδείγματα συναρτήσεων.

Παραδείγματα συναρτήσεων:

```
program     test_function1;
uses wincrt;
var i       :integer;
    ba,ek   :integer;

function power(basi, kth:integer):real;
var
     apotel :real;
begin
 apotel:=1;
 for i:=1 to ekth do
    apotel:=apotel* basi;
    power:=apotel;
end ;

begin
  clrscr;
  writeln('δώσε βάση, εκθέτη(0 για
           τέλος)');
  readln(ba, ek);
  writeln('βάση ':11, 'εκθέτης ':10,
```

Το πρόγραμμα καλεί τη συνάρτηση power για τον υπολογισμό δυνάμεων και οι πραγματικές παράμετροι ba, ek αντικαθιστούν τις τυπικές παραμέτρους, ***basi, ekth***, για τον υπολογισμό των δυνάμεων αυτών.

Η συνάρτηση power υπολογίζει τη δύναμη ενός αριθμού με δεδομένα τη βάση και τον εκθέτη.

basi, ekth είναι οι τυπικές παράμετροι.

Με διάβασμα δίνονται τιμές στις πραγματικές παραμέτρους και καλείται η συνάρτηση.

Στον πίνακα που ακολουθεί εμφανίζονται τα δεδομένα (βάση, εκθέτης) και τα απο-

```
          δύναμη' :30 );
 while ba > 0 do
     begin
       writeln(ba :10, ek :10,
                    power(ba,ek):30 :1 );
       readln(ba, ek);
     end;
 end.
```

τελέσματα (δύναμη) που υπολογίζονται με κλήση της συνάρτησης.
δώσε βάση, εκθέτη(0 για τέλος)

βάση	εκθέτης	δύναμη
5	2	25.0
5	3	125.0
5	4	625.0
2	8	256.0
2	16	65536.0
2	32	4294967296.0
2	64	18446744074000000000.0

```
program     test_functon2;
uses wincrt;
var pl      :integer;
```

Το πρόγραμμα καλεί τη συνάρτηση tetragwno για τον υπολογισμό του εμβαδού με πραγματική παράμετρο pl που αντικαθιστά την τυπική παράμετρο plevra, για τον υπολογισμό του εμβαδού

```
function tetragwno
(plevra:integer):integer;
var  i :integer;
begin
  tetragwno:=plevra*plevra;
end ;
```

Η συνάρτηση υπολογίζει το εμβαδόν ενός τετραγώνου

plevra είναι η τυπική παράμετρος

```
Begin
  Write('πλευρά τετραγώνου=');
  Readln(pl);
  While pl > 0 do
    Begin
      Writeln('Εμβαδόν=',tetragwno(pl));
      Write('πλευρά τετραγώνου=');
      Readln(pl);
    End;
end.
```

Με διάβασμα δίνονται τιμές στην πραγματική παράμετρο (pl) και καλείται η συνάρτηση tetragwno
Στον πίνακα που ακολουθεί εμφανίζονται τα δεδομένα (πλευρά) και τα αποτελέσματα (εμβαδόν) που υπολογίζονται με κλήση της συνάρτησης tetragwno.

πλευρά =10	Εμβαδόν	=100
πλευρά=100	Εμβαδόν	=10000
πλευρά =25	Εμβαδόν	=625
πλευρά =19	Εμβαδόν	=361
πλευρά =22	Εμβαδόν	=484
πλευρά = 0		

```
Program     test_function3;
uses wincrt;
var
      g1,g2,g3:real;
```

```
Function trigwno (gwnia1,gwnia2,gwnia3:real):boolean;
begin
 trigwno:=abs(gwnia1+gwnia2+gwnia3 -180.0) <0.00000001
end ;
```

```
Begin
 write('δώσε τις τρεις γωνίες σε μοίρες=');
 readln(g1,g2,g3);
 if trigwno(g1,g2,g3)
  then writeln('οι τρεις γωνίες σχηματίζουν τρίγωνο ')
  else writeln('οι τρεις γωνίες δε σχηματίζουν τρίγωνο ')
end.
```

Το προηγούμενο πρόγραμμα καλεί τη συνάρτηση trigwno με πραγματικές παραμέτρους g1,g2,g3 που αντικαθιστούν τις τυπικές παραμέτρους gwnia1, gwnia2, gwnia3 για τον υπολογισμό του αθροίσματος των τριών γωνιών του τριγώνου.

Η συνάρτηση επιστρέφει true αν το άθροισμα των γωνιών = 180

gwnia1,gwnia2,gwnia3 είναι οι τυπικές παράμετροι.

Με διάβασμα δίνονται τιμές στις πραγματικές παραμέτρους(g1,g2,g3) και καλείται η συνάρτηση trigwno. Αν επιστρέφει true εμφανίζεται το μήνυμα 'οι τρεις γωνίες σχηματίζουν τρίγωνο' αλλιώς εμφανίζεται το μήνυμα 'οι τρεις γωνίες δεν σχηματίζουν τρίγωνο'

Κατά την εκτέλεση ανάλογα με τα δεδομένα μπορεί να εμφανιστούν τα παρακάτω αποτελέσματα:.

30 60 70

οι τρεις γωνίες δε σχηματίζουν τρίγωνο

60 70 50

οι τρεις γωνίες σχηματίζουν τρίγωνο

50 80 70

οι τρεις γωνίες δε σχηματίζουν τρίγωνο

```
program     test_functon4;
uses wincrt;
var
   ar1,ar2,ar3,z        :integer;
```

Το πρόγραμμα καλεί τη συνάρτηση max με πραγματικές παραμέτρους ar1,ar2,ar3 που αντικαθιστούν τις τυπικές παραμέτρους a, b, c για την εύρεση του μεγίστου τριών αριθμών.

```
Function max(a, b, c :integer):integer;
Begin
  if a>b
  then    if  c>a
          then        max:=c
          else        max:=a
  else    if c>b
          then        max:=c
          else        max:=b;
end ;
```

Η συνάρτηση υπολογίζει το μέγιστο τριών δεδομένων αριθμών.

a, b, c είναι οι τυπικές παράμετροι.

```
Begin
  Writeln('δώσε 3 αριθμούς
          ( 0 για τέλος) ');
  readln (ar1,ar2,ar3);
  writeln(ar1,ar2,ar3);

 while ar1 > 0 do
 begin
  z:=max(ar1,ar2,ar3 );
  writeln('μέγιστος ο ', z );
  writeln('δώσε 3 αριθμούς(0 τέλος)');
  readln(ar1,ar2,ar3);
 end;
end.
```

Με διάβασμα δίνονται τιμές στις πραγματικές παραμέτρους(ar1,ar2,ar3) και καλείται η συνάρτηση max.
Στον πίνακα που ακολουθεί εμφανίζονται τα δεδομένα (3 αριθμοί) και τα αποτελέσματα (ο μέγιστος) που υπολογίζεται με κλήση της συνάρτησης max.

100 49 80

μέγιστος ο 100

100 300 124

μέγιστος ο 300

100 200 500

μέγιστος ο 500

0

```
program test_function5;
uses wincrt;
   var
      r, y     :integer;
```

```
function kyklos(aktina:integer):real;
{Η συνάρτηση υπολογίζει το εμβαδόν ενός κύκλου }
begin
 kyklos:= aktina * aktina * pi;;
end ;
```

```
function kylindros(kyklos :real; ipsos:integer):real;
{Η συνάρτηση υπολογίζει τον όγκο   ενός κυλίνδρου }
begin
 kylindros:= kyklos * ipsos ;
end ;
```

```
begin
writeln('ακτίνα ύψος Ε κύκλου  V  κυλίνδρου. ');
  readln(r,y);
  while r > 0 do
     begin
       write(kyklos(r):20:2);
       writeln(kylindros(kyklos(r),y):15:2);
       readln(r,y);
     end;
end.
```

Το προηγούμενο πρόγραμμα καλεί

- τη συνάρτηση kyklos με πραγματική παράμετρο r, που αντικαθιστά την τυπική παράμετρο aktina για την εύρεση του εμβαδού ενός κύκλου.
- τη συνάρτηση kylindros με πραγματικές παραμέτρους kyklos(r),y, που αντικαθιστούν τις τυπικές παραμέτρους kyklos, ipsos: για την εύρεση του όγκου κυλίνδρου.

Η συνάρτηση kyklos υπολογίζει το εμβαδόν ενός κύκλου.
Τυπική παράμετρος aktina.

Η συνάρτηση kylindros υπολογίζει τον όγκο ενός κυλίνδρου.Τυπικές παράμετροι kyklos, ipsos Η παράμετρος kyklos είναι μία άλλη συνάρτηση που έχει οριστεί προηγούμενα.

Με διάβασμα δίνονται τιμές στις πραγματικές παραμέτρους r,y και καλείται η συνάρτηση kyklos καθώς και η συνάρτηση kylindros.
Τα αποτελέσματα (εμβαδόν κύκλου και όγκος κυλίνδρου) υπολογίζονται με κλήση των συναρτήσεων kyklos και. Kylindros.

ακτίνα	ύψος	Ε κύκλου	V κυλίνδρου.
100	100	31415.93	3141592.65
200	1000	125663.71	125663706.14
111	555	38707.56	21482697.51
10	99	314.16	31101.77
0	0		

Πότε χρησιμοποιούμε συναρτήσεις

Παρακάτω αναφέρονται ορισμένες προτάσεις για τη χρήση συναρτήσεων.

- Αν το τμήμα προγράμματος επιστρέφει περισσότερες από μία τιμές ή αλλάζει τιμές πραγματικών παραμέτρων τότε δεν χρησιμοποιούμε συνάρτηση.
- Αν το τμήμα προγράμματος εκτελεί είσοδο δεδομένων ή έξοδο αποτελεσμάτων΄ τότε δε χρησιμοποιούμε συνάρτηση
- Αν το τμήμα προγράμματος επιστρέφει μια τιμή και η τιμή αυτή είναι λογική (boolean) τότε χρησιμοποιούμε τη συνάρτηση
- Αν το τμήμα προγράμματος επιστρέφει μια τιμή και η τιμή αυτή χρησιμοποιείται άμεσα σε μια παράσταση τότε χρησιμοποιούμε τη συνάρτηση
- Αν υπάρχει αμφιβολία είναι προτιμότερο να χρησιμοποιήσουμε μία διαδικασία. Στη διαδικασία το ρόλο του ονόματος της συνάρτησης παίζει η μία παράμετρος μεταβλητής.
- Αν είναι δυνατόν να χρησιμοποιήσουμε και τις δύο δηλαδή διαδικασία ή συνάρτηση χρησιμοποιούμε αυτή με την οποία αισθανόμαστε πιο άνετα στην υλοποίηση.

Οι συναρτήσεις περιλαμβάνονται στη γλώσσα Pascal για προσομοίωση Μαθηματικών συναρτήσεων. Η Pascal διαθέτει ενσωματωμένες Μαθηματικές συναρτήσεις όπως Abs(x) Απόλυτη τιμή, Sin(x) Ημίτονο χ, Sqr(x) Τετράγωνο του χ, Sqrt(x) Τετραγωνική ρίζα, Ln(x) Λογάριθμος του x κλπ.

ΑΝΑΔΡΟΜΗ

Το πεδίο ορισμού ενός υποπρογράμματος αρχίζει με την επικεφαλίδα και τελειώνει με τη λέξη end του τμήματος το οποίο το ορίζει. Έτσι, ένα υποπρόγραμμα μπορεί μέσα από το πεδίο ορισμού του να καλεί ένα άλλο υποπρόγραμμα ή ακόμη και τον ίδιο τον εαυτό του. Μια διαδικασία ή συνάρτηση η οποία μεταξύ των διαφόρων εντολών περιέχει και μία τουλάχιστον κλήση στον εαυτό της, ονομάζεται διαδικασία ή συνάρτηση με αναδρομή.

Πολλοί ορισμοί, υπολογισμοί και αλγόριθμοι περιγράφονται ευκολότερα και με πιο συνοπτικό τρόπο με τη χρήση αναδρομής. Αυτό φανερώνει ότι η αναδρομή είναι αλγοριθμική δομή πληρέστερη από την επανάληψη.

Όλοι οι ορισμοί με αναδρομή έχουν δύο κοινά χαρακτηριστικά:

α. ένα στοιχείο που ελέγχει την αναδρομή.

β. ένα μηχανισμό που επιτρέπει την έξοδο από τη διαδικασία.

Παράδειγμα χρήσης αναδρομής

```
program recur_1;
{εύρεση του n!, με recursion και χωρίς recursion}
uses wincrt;
var
   n :integer;
   xx:char;
```

```
  function paragontiko(n:integer):Longint;
{παραγοντικό χωρίς αναδρομή}
var
   par,i:  integer;
   begin
      par:=1;
      for i:=2 to n do
            par:=par*i;
      paragontiko:=par;
   end;
```

```
function paragontiko_rec(n:integer):Longint;
{παραγοντικό με αναδρομή }
   begin
      if   n=0
      then paragontiko_rec:=1
      else paragontiko_rec:=n*paragontiko_rec(n-1);
   end;
```

```
begin
   clrscr;
   writeln('ΠΑΡΑΓΟΝΤΙΚΟ ΜΕ ΑΝΑΔΡΟΜΗ ');
   writeln('δώσε έναν αριθμό ή -1 για τέλος');
   readln (n);
   while n<>-1 do
      begin
         writeln(n,'!=',paragontiko_rec(n));
         writeln('δώσε έναν αριθμό ή -1 για τέλος');
         readln (n);
      end;
   writeln('πάτησε έναν πλήκτρο');
   readln;
   clrscr;
   writeln('ΠΑΡΑΓΟΝΤΙΚΟ ΧΩΡΙΣ ΑΝΑΔΡΟΜΗ ');
   writeln('δώσε έναν αριθμό ή -1 για τέλος');
   readln (n);
   while n<>-1 do
      begin
         writeln(n,'!=',paragontiko(n));
         writeln('δώσε έναν αριθμό ή -1 για τέλος');
         readln (n);
      end;
end.
```

Ανακεφαλαίωση

Στο κεφάλαιο αυτό κατανοήσαμε τη μορφή των υποπρογραμμάτων, διαδικασίες και συναρτήσεις, τα οποία ορίζονται από τον χρήστη και υποστηρίζονται από τη γλώσσα Pascal. Κατανοήσαμε τη διαφορά μεταξύ διαδικασίας και συνάρτησης, τον τρόπο με τον οποίο εισάγουμε τιμές σε μία διαδικασία και σε μία συνάρτηση καθώς και τον τρόπο με τον οποίο λαμβάνουμε τιμές από μία διαδικασία ή μία συνάρτηση. Η διαδικασία μπορεί να επιστρέψει περισσότερες από μία τιμές στο κύριο πρόγραμμα μέσα από παραμέτρους. Η συνάρτηση επιστρέφει μία τιμή μέσα από το όνομά της.

Ερωτήσεις

1. Να συγκρίνετε τη μορφή της διαδικασίας και του προγράμματος.
2. Τί είναι και σε τι χρησιμεύει μια διαδικασία;
3. Ποιές είναι οι τυπικές και ποιες οι πραγματικές παράμετροι μιας διαδικασίας;
4. Πώς εκτελείται μία διαδικασία;
5. Ποιές είναι οι παράμετροι τιμών και ποιες οι παράμετροι μεταβλητών σε μία διαδικασία; Να δώσετε παραδείγματα από τα προγράμματα διαδικασιών του βιβλίου σας.
6. Τί είναι συνάρτηση οριζόμενη από το χρήστη και ποια είναι η διαφορά της από τη διαδικασία;
7. Πώς καλείται μία συνάρτηση;
8. Να δώσετε δικά σας παραδείγματα κλήσεων συναρτήσεων από τα προγράμματα συναρτήσεων του βιβλίου σας ορίζοντας δικά σας δεδομένα και να προσδιορίσετε τα αποτελέσματα.
9. Να γράψετε μια συνάρτηση για το ημιάθροισμα δύο αριθμών.
10. Να γράψετε μια συνάρτηση για το γινόμενο δύο αριθμών.

Ασκήσεις

1. Να γράψετε πρόγραμμα που να υπολογίζει τον τελικό βαθμό ενός μαθητή σ' ένα μάθημα από την προφορική βαθμολογία των τριών τριμήνων και από το βαθμό των γραπτών.

Υποδείξεις

α. Ο υπολογισμός του βαθμού γίνεται με βάση τον τύπο:

$$\text{ΤελικόςΒαθμός} = \frac{\text{Α.τριμ} + \text{Β.τριμ} + \text{Γ.τριμ} + 2x\text{ΓραπτόςΒαθμός}}{5}$$

β. Να χρησιμοποιήσετε μία διαδικασία για την εισαγωγή δεδομένων, μία για την εύρεση του μέσου όρου και μια για την εκτύπωση των αποτελεσμάτων.

γ. Να χρησιμοποιηθούν παράμετροι τιμών καθώς και παράμετροι μεταβλητών(var) στις διαδικασίες.

2. Να γράψετε πρόγραμμα το οποίο να διαβάζει δύο αριθμούς και το είδος της πράξης που θα εκτελέσει (πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση). Να χρησιμοποιήσετε διαδικασίες (χωρίς παραμέτρους) για την εισαγωγή των δεδομένων, την επιλογή της πράξης, τον υπολογισμό και την εκτύπωση του αποτελέσματος. Να χρησιμοποιήσετε ένα μηχανισμό ελέγχου, ώστε οι διαδικασίες να επαναλαμβάνονται όσες φορές θέλουμε.

3. Να γραφεί πρόγραμμα που :

α. Διαβάζει το εμβαδόν τεσσάρων διαμερισμάτων, του καθενός χωριστά και τη συνολική δαπάνη πετρελαίου.

β. Υπολογίζει το ποσοστό συμμετοχής
(= εμβαδόν διαμερίσματος /συνολικό εμβαδόν)
και την αντίστοιχη χρέωση για κάθε διαμέρισμα.

γ. Τυπώνει τα αποτελέσματα σύμφωνα με το υπόδειγμα που ακολουθεί:

Υποδείξεις

α. Να γραφούν διαδικασίες
για την εισαγωγή των δεδομένων (εμβαδόν, κόστος πετρελαίου),
για τον υπολογισμό
και την εκτύπωση των δεδομένων

β. Να τυπωθούν τα αποτελέσματα σύμφωνα με το παρακάτω υπόδειγμα.

αξία πετρελαίου 100.000 δρχ.

διαμέρισμα	εμβαδόν	ποσοστά	χρέωση πετρελαίου
διαμέρισμα 1	150	0.349	34.900
διαμέρισμα 2	100	0.233	23.300
διαμέρισμα 3	80	0.186	18.600
διαμέρισμα 4	100	0.233	23.300
	430	1.000	100.000

4. Για τον υπολογισμό φόρου έστω ότι ισχύει ο παρακάτω πίνακας:

μισθός		ποσοστό φόρου
0	- 100.000 δρχ	12%
100.001	- 200.000 δρχ	18%
200.001	- 300.000 δρχ	25%
300.001	- 400.000 δρχ	35%
400.001	- 500.000 δρχ	50%
500.001	και άνω	60%

Να γραφεί πρόγραμμα το οποίο να χρησιμοποιεί τις διαδικασίες:

α. εισαγωγής δεδομένων (ονοματεπώνυμο, αποδοχές).

β. υπολογισμού.(φόρου, πληρωτέου ποσού).

γ. εκτύπωσης των αποτελεσμάτων σύμφωνα με το υπόδειγμα:

ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΑΠΟΔΟΧΕΣ	ΦΟΡΟΣ	ΚΑΘ.ΠΛΗΡ.
ΔΗΜΗΤΡΙΑΔΗΣ	ΝΙΚΟΛΑΟΣ	250.000	42.500	202.500

Υποδείξεις:

α. Ο φόρος να υπολογιστεί κλιμακωτά, πχ. για το ποσό των 250.000 δρχ έχουμε: 100.000 x 12% +100.000 x18% +50.000 x 25% =42.500 δρχ.

β. Οι διαδικασίες επαναλαμβάνονται, μέχρις ότου δοθεί τέλος δεδομένων (αποδοχές=0).

5. Να γράψετε πρόγραμμα που να διαβάζει δύο ακέραιους και θετικούς αριθμούς και να καλεί μία συνάρτηση για τον υπολογισμό του μέγιστου κοινού διαιρέτη. Η εργασία επαναλαμβάνεται όσο οι αριθμοί είναι και οι δύο θετικοί. Αν και οι δύο αριθμοί είναι μηδέν, η εργασία σταματάει και τυπώνεται κατάλληλο μήνυμα.

Υποδείξεις:

α. Να χρησιμοποιηθεί η εντολή while και παράμετροι για τη συνάρτηση.

β. Τα αποτελέσματα παρουσιάζονται με τη μορφή:

οι αριθμοί .... και .... έχουν
μέγιστο κοινό διαιρέτη .....
ή τέλος εργασίας

6. Να γράψετε πρόγραμμα που να διαβάζει έναν ακέραιο αριθμό και να καλεί μία συνάρτηση για τον υπολογισμό της τρίτης δύναμης του αριθμού αυτού. Η εργασία συνεχίζεται, αν ο αριθμός είναι διάφορος του μηδενός, αλλιώς η εργασία σταματάει και δίνεται και κατάλληλο μήνυμα.

Υποδείξεις:

α.: Να χρησιμοποιηθεί η εντολή repeat

β. Τα αποτελέσματα παρουσιάζονται με τη μορφή:

ο κύβος του αριθμού ***.** είναι *******.****
ή ο αριθμός είναι μηδέν
τέλος εργασίας

7. Να γράψετε πρόγραμμα που χρησιμοποιώντας κατάλληλες συναρτήσεις να υπολογίζει και να τυπώνει το μήκος του κύκλου και το εμβαδόν του κυκλικού δίσκου ακτίνας R. Η διαδικασία να επαναλαμβάνεται για τιμές της ακτίνας από 10cm έως 1.5 m με βήμα 10 cm.

8. Να γράψετε πρόγραμμα που να διαβάζει το ύψος και τη βάση ενός τριγώνου και να καλεί μία συνάρτηση που να υπολογίζει το εμβαδόν του. Να γίνει εκτύπωση των αποτελεσμάτων σύμφωνα με το υπόδειγμα:

βάση .....
ύψος .....
εμβαδόν .....

9. Να γράψετε πρόγραμμα που να διαβάζει:

α. ένα τόξο (ή γωνία) σε μοίρες, να μετατρέπει την τιμή αυτή σε ακτίνια καλώντας μία συνάρτηση και να τυπώνει στην οθόνη το αποτέλεσμα.

β. ένα τόξο (ή γωνία) σε ακτίνια, να μετατρέπει την τιμή αυτή σε μοίρες καλώντας μία συνάρτηση και να τυπώνει στην οθόνη το αποτέλεσμα.

Υπόδειξη:

Ο τύπος μετατροπής είναι:

$$\frac{ακτίνια}{\pi} = \frac{μοίρες}{180}$$

όπου π=3.14

10. Να γράψετε πρόγραμμα που να διαβάζει έναν αριθμό n, να καλεί μία συνάρτηση για τον υπολογισμό του n! (όπου n!=1*2*3*...*n) και να τυπώνει το αποτέλεσμα στην οθόνη. Η διαδικασία να επαναλαμβάνεται για 10 αριθμούς με τη χρησιμοποίηση της εντολής FOR.

11. Ο υπολογισμός της τιμής του ημιτόνου ενός τόξου x (σε ακτίνια) δίνεται από το άθροισμα της σειράς:

$$ημ(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ... = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Να γράψετε πρόγραμμα που να διαβάζει την τιμή του τόξου σε μοίρες και να υπολογίζει το ημίτονο του τόξου αυτού.

Υποδείξεις:

α. Η διαδικασία σταματά όταν η διαφορά δύο διαδοχικών υπολογισθέντων τιμών της συνάρτησης είναι μικρότερη από 0.00001.

β. Να χρησιμοποιήσετε τις δύο προηγούμενες συναρτήσεις για τον υπολογισμό του παραγοντικού και της μετατροπής ενός τόξου από μοίρες σε ακτίνια.

Δραστηριότητες:

1. Δίνεται ως παράδειγμα η κωδικοποίηση της άσκησης 1 σε Qbasic Να γράψετε με τον editor, να **μεταφράσετε** και να εκτελέσετε χρησιμοποιώντας τις κατάλληλες εντολές, το πρόγραμμα που ακολουθεί στο περιβάλλον της **Qbasic.**

```
'άσκηση 1, Κεφάλαιο 12 ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ
'Υπολογίζεται ο τελικός βαθμός ενός μαθητή, με χρήση
διαδικασιών
DECLARE SUB          eisaggwgi(a!, b!,c!,g!)
DECLARE SUB          ektypwsi(mo2!)
DECLARE SUB        ypologismoi(a1!, b1!,c1!,g1!,mo1!)
CLS
CALL          eisaggwgi(v1, v2, v3, gr)
CALL          ypologismoi (v1, v2, v3, gr, MesosOros)
CALL          ektypwsi (MesosOros)
END

SUB       eisaggwgi(a, b,c,g)
```

```
CLS
'εισαγωγή δεδομένων
INPUT    "δώσε βαθμό α τριμήνου : " ;a
INPUT    "δώσε βαθμό β τριμήνου : " ;b
INPUT    "δώσε βαθμό γ τριμήνου : " ;c
INPUT    "δώσε βαθμό γραπτών    : " ;g
END SUB

SUB         ektypwsi(mo2)
'εκτύπωση αποτελεσμάτων
PRINT  "τελική βαθμολογία ¨; TAB(25)´mo2
END SUB

SUB         ypologismoi(a1, b1,c1,g1,mo1)
mo1= (a1+b1+c1+2*g1)/5
END SUB
```

2. Να γράψετε με τον editor, να **μεταφράσετε** και να εκτελέσετε τρεις από τις προηγούμενες ασκήσεις του κεφαλαίου 12 στο περιβάλλον της **Qbasic** χρησιμοποιώντας τις κατάλληλες εντολές.

3. Να σχεδιαστεί και να γραφεί πρόγραμμα μισθοδοσίας σε PASCAL που να διαβάζει το πλήθος των εργαζομένων και να επαναλαμβάνει για καθέναν τις παρακάτω διαδικασίες:

Εισαγωγή δεδομένων.
- α. Ονοματεπώνυμο εργαζομένου.
- β. Βασικός μισθός.

Υπολογισμών .
- α. Κρατήσεων εργαζομένου
  - i. ΙΚΑ
  - ii. ΧΑΡΤΟΣΗΜΟΥ
  - iii. ΦΟΡΟΥ.
- β. Συνόλου κρατήσεων εργαζομένου
- γ. Καθαρού πληρωτέου για τον εργαζόμενο

Εκτύπωσης.

Σύμφωνα με το υπόδειγμα:

ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΑΣ.ΜΙΣΘΟΣ	ΙΚΑ|	ΧΡΤ	ΦΟΡΟΣ	ΚΡΑΤΗΣΕΙΣ	ΚΑΘΑΡΟ
θέσεις 30	10	4	4	6	9	9

Υποδείξεις:

α. Κρατήσεις ΙΚΑ. Υπολογίζονται με ποσοστό 10.25%.στο βασικό μισθό.

β. Κρατήσεις ΧΑΡΤΟΣΗΜΟΥ. Υπολογίζονται με ποσοστό 1%.στο βασικό μισθό.

γ. Κρατήσεις.ΦΟΡΟΥ. Υπολογίζονται με ποσοστό 10%.στο (Βασικός μισθός -(ΙΚΑ + ΧΑΡΤΟΣΗΜΟ)).

4. Να γράψετε πρόγραμμα για την καταμέτρηση των ψήφων δύο κομμάτων και για την εξαγωγή του τελικού αποτελέσματος.

Υποδείξεις:

α. Η εισαγωγή των ψήφων γίνεται ξεχωριστά για κάθε κόμμα.
β. Με -1 δηλώνεται τέλος δεδομένων για το κόμμα 1
γ. Με -2 δηλώνεται τέλος δεδομένων για το κόμμα 2
δ. Να χρησιμοποιήσετε μία procedure για τον υπολογισμό του συνόλου των ψήφων κάθε κόμματος.
ε. Η εκτύπωση να γίνει ως εξής:
Την πλειοψηφία κέρδισε το κόμμα ..... (1 ή 2)
Σε περίπτωση ισοψηφίας να τυπώνεται:
ισοψηφία, επανάληψη εκλογών.

Ακολουθεί ο αλγόριθμος λύσης του προβλήματος σε ψευδοκώδικα για διευκόλυνση ή για ενιαία λύση στο εργαστήριο.

```
Διαδικασία ΣυνολοΨηφων.
αρχή
   βάλε 0 στο σύνολο των ψήφων
   διάβασε τον αριθμό των ψήφων
   αν ο αριθμός των ψήφων <> από τον αριθμό τέλους ψήφων
    (-1 ή -2 αντίστοιχα για το κόμμα 1 ή το κόμμα 2)
   επανάλαβε
      πρόσθεσε τον αριθμό των ψήφων στο σύνολο
      διάβασε τον αριθμό των ψήφων
   μέχρις ότου ο αριθμός των ψήφων = με τον αριθμό τέλους ψήφων
τέλος(διαδικασία ΣυνολοΨηφων)

ΚΥΡΙΟ ΠΡΟΓΡΑΜΜΑ
αρχή
   {κόμμα 1}
      βάλε το -1 στον αριθμό τέλους ψήφων {για το κόμμα 1}
      δώσε τον αριθμό ψήφων η -1 για τέλος δεδομένων
      κάλεσε τη διαδικασία ΣυνολοΨηφων.
      βάλε το σύνολο των ψήφων στο σύνολο-1
      τύπωσε σύνολο-1 {τους ψήφους για το κόμμα 1}

   {κόμμα 2}
      βάλε το -2 στον αριθμό τέλους ψήφων {για το κόμμα 2}
      δώσε τον αριθμό ψήφων η -2 για τέλος δεδομένων
      κάλεσε τη διαδικασία ΣυνολοΨηφων.
      βάλε το σύνολο των ψήφων στο σύνολο-2
      τύπωσε σύνολο-2 {τους ψήφους για το κόμμα 2}

      αν σύνολο-1 = σύνολο-2
      τότε τύπωσε 'ισοψηφία, επανάληψη εκλογών'
      αλλιώς αν σύνολο-1 > σύνολο-2
                τότε τύπωσε την πλειοψηφία κέρδισε το κόμμα 1
                αλλιώς τύπωσε την πλειοψηφία κέρδισε το κόμμα 2
τέλος (κύριου προγράμματος)
```