Περιεχόμενα

Λίγα λόγια για το μαθητή 9

1. Ανάλυση προβλήματος 11

2. Βασικές έννοιες αλγορίθμων 17

3. Δομές Δεδομένων και Αλγόριθμοι 29

4. Τεχνικές σχεδίασης αλγορίθμων 37

5. Ανάλυση αλγορίθμων 47

6. Εισαγωγή στον προγραμματισμό 57

7. Βασικές έννοιες προγραμματισμού 61

8. Επιλογή και επανάληψη 71

9. Πίνακες 87

10. Υποπρογράμματα 99

11. Σύγχρονα προγραμματιστικά περιβάλλοντα 113

12. Σχεδίαση διεπαφής χρήστη 151

13. Εκσφαλμάτωση προγράμματος 157

14. Αξιολόγηση Τεκμηρίωση 167

Παραρτήματα 179

- Turbo Pascal 181
- Quick Basic 191
- Visual Basic 201
- Delphi 227

Απαντήσεις στα τεστ αυτοαξιολόγησης 235

Λίγα λόγια για το μαθητή

Αγαπητέ μαθητή,

Στα χέρια σου κρατάς το Τετράδιο Μαθητή, ένα συμπληρωματικό, αλλά ταυτόχρονα και λειτουργικά απαραίτητο σύγγραμμα για τη διδασκαλία του μαθήματος "*Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον*". Σκοπός του μαθήματος δεν είναι να σε διδάξει και να εμβαθύνεις σε κάποια συγκεκριμένη γλώσσα προγραμματισμού. Η έμφαση και η προσπάθεια γίνεται στο να μπορέσεις :

- να αναπτύξεις αναλυτική σκέψη και συνθετική ικανότητα,
- να καλλιεργήσεις αυστηρότητα στη διατύπωση,
- να αναπτύξεις δημιουργικότητα και φαντασία στο σχεδιασμό,
- να αποκτήσεις ικανότητες μεθοδολογικού χαρακτήρα,
- να αναπτύξεις δεξιότητες αλγοριθμικής προσέγγισης,
- να μπορείς να επιλύεις προβλήματα και να υλοποιείς απλά τη λύση τους με χρήση βασικών γνώσεων προγραμματιστικού περιβάλλοντος.

Η θεωρητική πλευρά του μαθήματος καλύπτεται από το βιβλίο μαθητή. Σκοπός του *τετραδίου* αυτού είναι να σου προσφέρει μέσα από τα έτοιμα παραδείγματα που σου παρουσιάζει, αλλά και τις δραστηριότητες που σου προτείνει, μια πρακτική προσέγγιση των γνώσεων που παραθέτονται στο βιβλίο μαθητή. Με μια σειρά από τρόπους, μεθόδους και απλές τεχνικές χρήσης διαφόρων προγραμματιστικών περιβαλλόντων, σε βοηθάει στη μοντελοποίηση και επίλυση πραγματικών ή ιδεατών προβλημάτων.

Τα παραδείγματα και οι δραστηριότητες που προτείνονται περιγράφονται είτε σε μια υποθετική γλώσσα προγραμματισμού, τη ΓΛΩΣΣΑ, είτε σε πραγματικές γλώσσες προγραμματισμού, την QuickBasic, την Turbo Pascal, τη Visual Basic και την Delphi. Κάποιες από τις γλώσσες αυτές, είναι αυτές που θα χρησιμοποιήσεις στο εργαστήριο Πληροφορικής του σχολείου σου. Η αναφορά των ρεπερτορίων εντολών και των τεχνικών κάθε μιας από τις γλώσσες αυτές δεν γίνεται διεξοδικά, αφού σκοπός του μαθήματος, όπως προείπαμε, δεν είναι η εκμάθηση κάποιας γλώσσας προγραμματισμού. Όμως για τη βοήθειά σου έχουμε εντάξει στο τέλος του τετραδίου εργασιών, ένα παράρτημα που περιλαμβάνει τέσσερα συνοπτικά εγχειρίδια χρήσης, καθένα από τα οποία αναφέρεται στις παραπάνω γλώσσες προγραμματισμού.

Τα παραδείγματα που παρουσιάζονται και οι δραστηριότητες που προτείνονται:

- αντλούν ιδέες από πραγματικές καταστάσεις και από εμπειρίες της καθημερινής ζωής ή θίγουν πολιτιστικά, πολιτισμικά και κοινωνικά θέματα ευρύτερου ενδιαφέροντος, δίνοντας σου έτσι το ερέθισμα για περαιτέρω προβληματισμό,
- συνδέονται αρκετές φορές με άλλα μαθήματα όπως μαθηματικά, φυσική, χημεία, βιολογία, για να σου υπενθυμίζουν έμμεσα ότι, ο υπολογιστής δεν είναι αυτοσκοπός, αλλά εργαλείο επίλυσης προβλημάτων,
- δίνουν μεγαλύτερη έμφαση στην ανάλυση του προβλήματος και στο σχεδιασμό της λύσης, παρά στην υλοποίησή της, υποδηλώνοντάς σου έτσι συνέχεια ότι η προσπάθεια που θα πρέπει να καταβάλεις, δεν είναι προς κατεύθυνση της καλ-

λιέργεια τεχνικής, αλλά προς εκείνη της ανάπτυξης αναλυτικής και συνθετικής σκέψης.

Συμβάσεις

Για την καλύτερη αναγνωσιμότητα του τετραδίου έχουν χρησιμοποιηθεί και μερικά γνωστά εικονίδια από το βιβλίο μαθητή. Εκτός από αυτά, για τη διαβάθμιση των προτεινόμενων δραστηριοτήτων και ασκήσεων χρησιμοποιήθηκαν και τα παρακάτω:

 για μέτρια

 για προωθημένη

Ευχαριστίες

Η συγγραφική ομάδα θα ήθελε να επισημάνει τη συμβολή της Ελληνικής Εταιρείας Επιστημόνων Η/Υ και Πληροφορικής (ΕΠΥ) για την επίτευξη αυτού του σημαντικού και δύσκολου έργου που της ανέθεσε.

Για τη δημιουργία των εγχειριδίων χρήσης των Turbo Pascal και Delphi βοήθησαν οι *Κώστας Αντωνακόπουλος* και *Βαγγέλης Χαραλαμπόπουλος,* τους οποίους ευχαριστούμε θερμά και από τη θέση αυτή.

Οι συγγραφείς

ΚΕΦΑΛΑΙΟ 1ο

ΑΝΑΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

1.1. Προσδοκώμενα αποτελέσματα

Η μελέτη αυτού του πρώτου κεφαλαίου αναμένεται ότι θα σου καταστήσει σαφή την έννοια του προβλήματος. Η σωστή αντιμετώπιση ενός προβλήματος προϋποθέτει την καταρχήν πλήρη κατανόησή του. Η βάση της κατανόησης είναι η σαφής διατύπωσή του και αυτό απαιτεί σωστή χρήση του γραπτού και του προφορικού λόγου. Μέσα από τα παραδείγματα που αναφέρονται και τις δραστηριότητες που προτείνονται, θα μπορέσεις να καλλιεργήσεις την αναλυτική σου ικανότητα, ώστε να είσαι σε θέση να προσδιορίζεις τα συστατικά μέρη ενός προβλήματος και να το αναλύεις στη συνέχεια σε απλούστερα. Θα μάθεις να αναγνωρίζεις τα δεδομένα ενός προβλήματος και να προσδιορίζεις τα ζητούμενα αποτελέσματα στην επιθυμητή μορφή. Τέλος, θα είσαι σε θέση να θέσεις ο ίδιος προβλήματα διατυπώνοντάς τα με πληρότητα και ακρίβεια.

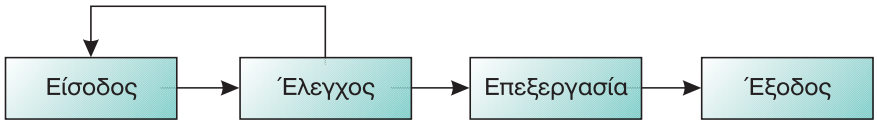
1.2. Επιπλέον παραδείγματα

Παράδειγμα 1

Στο κεφάλαιο 1.4 Καθορισμός απαιτήσεων του βιβλίου του μαθητή, παρουσιάζεται το πρόβλημα "Αποτελέσματα φοίτησης μαθητών Γ' τάξης Τεχνολογικής Κατεύθυνσης στα μαθήματα ειδικότητας". Εδώ θα προτείνουμε μια επέκτασή του προχωρώντας καταρχήν σε μια πιο αυστηρή διατύπωση του προβλήματος:

"Δίδονται οι βαθμολογίες όλων των μαθητών Γ' Λυκείου Τεχνολογικής Κατεύθυνσης του σχολικού έτους 1999/2000 στα τέσσερα μαθήματα ειδικότητας. Ζητείται να εκδοθούν στατιστικά αποτελέσματα κατά μάθημα, που περιλαμβάνουν (α) πίνακα συχνοτήτων, (β) τη μέση τιμή και την τυπική απόκλιση".

Οι απαιτούμενες ενέργειες για την αντιμετώπιση του προβλήματος είναι αυτές που φαίνονται στο σχήμα 1.1.



Σχ. 1.1.

- *Καταχώριση δεδομένων*. Οι βαθμολογίες όλων των μαθητών για ένα μάθημα συγκεντρώνονται και καταγράφονται.
- *Έλεγχος δεδομένων*. Τα δεδομένα ελέγχονται ως προς την ορθότητά τους και γίνονται οι απαραίτητες διορθώσεις, αν απαιτείται.
- *Επεξεργασία δεδομένων*. Γίνονται οι απαραίτητοι υπολογισμοί προκειμένου να βρεθούν τα ζητούμενα αποτελέσματα.
- *Εξαγωγή αποτελεσμάτων*. Δημιουργείται ο πίνακας συχνοτήτων (βλέπε παρ. 1.4 του βιβλίου), σχεδιάζεται το γράφημα και αποτυπώνεται η μέση τιμή και η τυπική απόκλιση.

Από τα παραπάνω μέρη του προβλήματος δεν είναι αρκετά σαφές τι περιλαμβάνει η επεξεργασία δεδομένων. Δηλαδή ποιοι ακριβώς είναι οι απαραίτητοι υπολογισμοί για την εύρεση των αποτελεσμάτων. Έτσι το μέρος αυτό κρίνεται ότι πρέπει να αναλυθεί περισσότερο, όπως στη συνέχεια.

Οι απαιτούμενοι υπολογισμοί είναι:

1. Βρίσκεται το πλήθος όλων των μαθητών, έστω N.
2. Καταμετρείται το πλήθος των μαθητών που έχει βαθμολογία ίση ή μικρότερη του 9 έστω K1, από 10 έως 13 έστω K2, κ.ο.κ.
3. Το ποσοστό των απορριπτόμενων μαθητών βρίσκεται από τον τύπο K1/N*100.
4. Αθροίζονται όλες οι βαθμολογίες και έστω S το άθροισμα. Η μέση τιμή μ υπολογίζεται από τη σχέση

   $$\mu = \frac{S}{N}$$

5. Αθροίζονται επίσης τα τετράγωνα των βαθμολογιών και έστω S2 το άθροισμα αυτό. Η τυπική απόκλιση σ βρίσκεται από τον τύπο

   $$\sigma^2 = \frac{S2}{N} - \mu^2$$

Το παράδειγμα εδώ ολοκληρώνεται σε θεωρητικό επίπεδο. Απομένει η υλοποίησή του με πειραματικά ή πραγματικά δεδομένα, όπως ζητείται από τη δραστηριότητα ΔΣ5.

1.3. Συμβουλές - υποδείξεις

Η επιτυχής προσπάθεια αντιμετώπισης ενός προβλήματος εξαρτάται σε πολύ μεγάλο βαθμό από τη σωστή κατανόησή του. Επομένως, πριν αρχίσεις οποιαδήποτε προσπάθεια επίλυσης ενός προβλήματος θα πρέπει να ασχοληθείς επισταμένως με την εκφώνησή του, ώστε να είσαι απολύτως βέβαιος ότι έχεις κατανοήσει σωστά και σε όλο τους το εύρος τα ζητούμενα.

Σημαντική διευκόλυνση για σένα θα είναι να μπορέσεις να αναλύσεις το πρόβλημα σε άλλα απλούστερα. Η αντιμετώπιση απλούστερων προβλημάτων είναι βέβαια πιο εύκολη. Θα πρέπει να έχεις υπόψη σου ότι κάθε κανόνας έχει τις εξαιρέσεις του, οπότε δεν θα πρέπει να εκπλαγείς, αν μετά την ανάλυση του προβλήματος κάποιο από τα επιμέρους προβλήματα που προέκυψαν, είναι πολύ δύσκολο να αντιμετωπιστεί. Θα πρέπει όμως να είσαι βέβαιος, πως η δυσκολία αυτή δεν είναι μεγαλύτερη από αυτήν που έχει το κύριο πρόβλημα.

Αφού κάποτε θα βρεθείς οπωσδήποτε στη θέση να διατυπώσεις ένα πρόβλημα, θα πρέπει να δώσεις προσοχή στη διατύπωσή του, έτσι ώστε να μην δημιουργεί παρερμηνείες και συγχύσεις σε κάποιον που θα κληθεί να το αντιμετωπίσει. Ιδιαίτερα μεγάλη προσοχή απαιτείται, αν το πρόβλημα "εκφράζεται" προς υπολογιστή, αφού η μηχανή δεν έχει την ευχέρεια να καταλάβει αυτά που θέλεις να δηλώσεις, αν δεν είναι απόλυτα σωστά διατυπωμένο.

1.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Η διεθνής αντιρατσιστική οργάνωση *SOS Ρατσισμός*, στην προσπάθειά της να συμβάλλει στην καταπολέμηση της ξενοφοβίας, διεξήγαγε μια ενημερωτική καμπάνια σε όλες τις ευρωπαϊκές πρωτεύουσες της Ενωμένης Ευρώπης. Κεντρικό σημείο της προσπάθειας αυτής ήταν η ενημέρωση των πολιτών σε θέματα φυλετικών διακρίσεων, αλλά και η υποβολή ερωτημάτων στους πολίτες εκ μέρους της οργάνωσης, με σκοπό τη συλλογή σχετικών απαντήσεων που θα μπορούσαν στη συνέχεια να χρησιμοποιηθούν σαν στατιστικά στοιχεία. Στην Αθήνα τα ενημερωτικά κιόσκια στήθηκαν σε 3 κεντρικά σημεία, όπου μοιραζόντουσαν ενημερωτικά φυλλάδια και ετίθετο και ερωτήσεις στους διερχόμενους. Οι απαντήσεις των ερωτηθέντων χρησιμοποιήθηκαν για την εξαγωγή συμπερασμάτων και τα αποτελέσματα φαίνονται στον παρακάτω πίνακα:

Ερώτηση	Ναι	Όχι	Δεν απάντησαν
Θα στέλνατε το παιδί σας σε ένα σχολείο με μεγάλο αριθμό παιδιών Αλβανών μεταναστών;	27%	61%	12%
Θα νοικιάζατε το διαμέρισμά σας σε μετανάστες της πρώην ανατολικής Ευρώπης ή του τρίτου κόσμου;	65%	32%	3%
Θα παίρνατε για κάποια πρόχειρη δουλειά (βάψιμο, οικιακή βοηθός κλπ) έναν/μια μετανάστη;	89%	7%	4%

Όπως στην Αθήνα, έτσι και στις άλλες ευρωπαϊκές πρωτεύουσες στήθηκαν παρόμοια κιόσκια και τέθηκαν παρόμοιες ερωτήσεις. Τα αποτελέσματα των ερευνών έδωσαν ανάλογα συμπεράσματα για κάθε χώρα.

Στο τέλος θεωρήθηκε σκόπιμο να βγουν κάποια συμπεράσματα συνολικά για την Ευρωπαϊκή Ένωση. Για το σκοπό αυτό χρησιμοποιήθηκαν οι πίνακες αποτελεσμάτων, όπως ο παραπάνω, για τις 15 ευρωπαϊκές πρωτεύουσες.

α) Να εντοπίσετε σε όλη την παραπάνω περιγραφόμενη ενέργεια ποια στοιχεία αποτελούν δεδομένα και ποια πληροφορίες για ποια διαδικασία. Σχολιάστε τις απαντήσεις.

β) Να αναλύσετε και να σχολιάσετε το πρόβλημα Ρατσισμός και να εκφράσετε λεκτικά και διαγραμματικά την ανάλυσή σας.

ΔΤ2. Το ενιαίο ευρωπαϊκό νομισματικό σύστημα είναι προ των πυλών. Φανταστείτε πως θα μπορούσαν οι ευρωπαϊκές χώρες να εφαρμόσουν στην πράξη το σύστημα αυτό αντιμετωπίζοντας με επιτυχία τα προβλήματα αποδοχής που θα συναντήσει από τους υπηκόους τους, οι οποίοι θα αντιδρούν στην αντικατάσταση του εθνικού τους νομίσματος. Αναλύστε καταρχήν το πρόβλημα φραστικά και στη συνέχεια κάντε την διαγραμματική του αναπαράσταση.

ΔΤ3. Μία σειρά από γραφειοκρατικές διαδικασίες διαφόρων υπηρεσιών έχουν αρχίσει ήδη να εξαλείφονται με τη χρήση υπολογιστών. Οι πολίτες δεν είναι πλέον υποχρεωμένοι να πηγαίνουν στις δημόσιες υπηρεσίες και να σχηματίζουν ουρές προκείμενου να παραλάβουν ένα πιστοποιητικό. Καλώντας ένα τηλεφωνικό αριθμό, μπορούν να δίνουν κάποια προσωπικά τους στοιχεία και να παραλαμβάνουν το πιστοποιητικό ταχυδρομικά στο σπίτι τους. Προσδιορίστε τα απαραίτητα δεδομένα που θα πρέπει να δίνει τηλεφωνικά ο πολίτης στην περίπτωση που θέλει να πάρει α) πιστοποιητικό γέννησης για έκδοση διαβατηρίου, β) φορολογική ενημερότητα για αγορά αυτοκινήτου.

ΔΤ4. Να επιλέξετε κάποιο από τα σοβαρότερα προβλήματα που απασχολούν τη μαθητική κοινότητα του σχολείου σας ή συνολικά τη μαθητική κοινότητα της χώρας μας. Να το διατυπώσετε με ακρίβεια και πληρότητα. Στη συνέχεια να προσδιορίσετε τα δεδομένα και τα ζητούμενα αποτελέσματα. Τέλος, μέσω της ανάλυσης του, να προτείνετε λύση του.

ΔΤ5. Ας υποθέσουμε ότι σήμερα είναι η γιορτή του Αγίου Γεωργίου και ότι θέλετε να τηλεφωνήσετε σε όλους τους φίλους σας και τις φίλες σας που γιορτάζουν για να τους ευχηθείτε. Θα πρέπει λοιπόν να ψάξετε στο προσωπικό σας σημειωματάριο για να βρείτε τα ονόματα και τα τηλέφωνα όλων όσων γιορτάζουν. Μια σκέψη που μπορεί να κάνατε μεγαλόφωνα είναι : "Να ψάξω να βρω όλους όσους λέγονται Γιώργος και Γεωργία".

Σχολιάστε τη διατύπωση αυτής της σκέψης.

Προβληματιστείτε για τα αποτελέσματα, αν ακριβώς τη σκέψη σας αυτή την μεταφράζατε σε μία γλώσσα προγραμματισμού και βάζατε τον υπολογιστή να βρει τους εορτάζοντες από το ηλεκτρονικό σας σημειωματάριο που κρατάτε σε αυτόν.

ΔΤ6. Να σχολιασθεί η άποψη: "*Οι υπολογιστές δεν είναι ούτε κατάρα, ούτε πανάκεια*".

Δεν είναι στη "φύση" των νέων τεχνολογιών να δημιουργούν προβλήματα ή να συντελούν στην ανθρώπινη πρόοδο. Οι τρόποι χρήσης είναι αυτοί που επηρεάζουν την ανθρώπινη ζωή και τις κοινωνίες. Να διατυπώσετε προβλήματα που δημιουργούνται αλλά και προβλήματα που λύνονται με τη χρήση των υπολογιστών.

Στο σπίτΨ

ΔΣ1. Η χρήση κεντρικών υπολογιστικών συστημάτων δημιουργεί και μια σειρά από κοινωνικά προβλήματα, ένα από τα οποία είναι ο κίνδυνος καταπάτησης του ιδιωτικού απόρρητου. Η δυνατότητα πρόσβασης σε στοιχεία των πολιτών, από τεχνικής πλευράς, είναι δύσκολο να αποκλειστεί. Τα καταχωρημένα στοιχεία μπορούν να αφορούν ατομικά στοιχεία (ονοματεπώνυμο, έτος και τόπος γέννησης, διεύθυνση κατοικίας, κλπ), κοινωνικοπολιτικά στοιχεία (θρήσκευμα, πολιτική τοποθέτηση, συνδικαλιστική δράση, κλπ), οικονομικά στοιχεία (ΑΦΜ, στοιχεία φορολογικών δηλώσεων, δάνεια, πιστωτικές κάρτες, κλπ), ιατρικά στοιχεία (ασθένειες, νοσηλείες, θεραπείες κλπ) καθώς και άλλα διάφορα στοιχεία (κλήσεις τροχαίας, αεροπορικά ταξίδια, κλπ). Καλείστε να προτείνετε τρόπους αντιμετώπισης του προβλήματος.

ΔΣ2. Οι κίνδυνοι εθισμού και εξάρτησης από την αλόγιστη χρήση των υπολογιστών, ειδικά για τα παιδιά και τους εφήβους είναι μεγάλοι. Πως θα μπορούσε να αντιμετωπιστεί δραστικά αυτό το πρόβλημα; Ποιες είναι οι γενεσιουργές του αιτίες; Αυτά είναι, μεταξύ πολλών άλλων, μερικά από τα ερωτήματα που θα πρέπει να σας απασχολήσουν ώστε να μπορέσετε να προσδιορίσετε σωστά το πρόβλημα και να το αναλύσετε σε επιμέρους προβλήματα. Καταγράψτε την φραστική ανάλυση για την αντιμετώπισή του και στη συνέχεια κάντε και τη διαγραμματική του αναπαράσταση.

ΔΣ3. Υποθέστε ότι είστε μέλος της σχολικής επιτροπής του σχολείου σας και ότι πρέπει να αντιμετωπίσετε το θέμα της αγοράς εξοπλισμού υπολογιστικών συστημάτων για τις διαφορετικές ανάγκες του σχολείου. Πρέπει να επιλέξετε εξοπλισμό για το ερ-

γαστήριο πληροφορικής, για το εργαστήριο καλλιτεχνικών σπουδών, για τη γραμματεία του σχολείου και για την αίθουσα εκδηλώσεων/παρουσιάσεων. Τι είδους εξοπλισμό θα διαλέγατε για να ικανοποιήσετε τις ανάγκες καθενός από αυτούς τους χώρους; Καταγράψτε τον βασικό εξοπλισμό και τις απαραίτητες περιφερειακές μονάδες για κάθε έναν χώρο ξεχωριστά.

ΔΣ4. Επιλέξτε ένα πρόβλημα από τον προσωπικό σας χώρο ή από τον κοινωνικό χώρο που σας απασχολεί. Διατυπώστε το με ακρίβεια και πληρότητα έτσι ώστε παρουσιάζοντάς το στη συνέχεια στην τάξη σας να γίνει απόλυτα κατανοητό από όλους.

ΔΣ5. Με τη συνεργασία του καθηγητή σας συγκεντρώστε στοιχεία βαθμολογίας μαθητών προηγούμενων ετών του σχολείου σας και πραγματοποιήστε την ανάλυση προβλήματος (για τέσσερα μαθήματα της επιλογής σας) που αναφέρεται στο κεφάλαιο 1.4 του βιβλίου και συνεχίζεται στο παράδειγμα του τετραδίου.

1.5. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες λέξεων. Σε κάθε μια από αυτές, να βάλεις τις λέξεις στη σωστή σειρά.

1. Επίλυση, ανάλυση, κατανόηση (αναφορά σε πρόβλημα)
2. Επεξεργασία, έλεγχος, έξοδος, είσοδος (αναφορά σε δεδομένα)

Συμπλήρωσε τα κενά με το σωστή λέξη ή λέξεις που λείπει(ουν)

3. Η επίλυση ενός προβλήματος ξεκινά από την _____του.
4. _____είναι το αποτέλεσμα επεξεργασίας δεδομένων.
5. Σημαντικός παράγοντας στην κατανόηση ενός προβλήματος είναι η _____.
6. Με τον όρο _____ προβλήματος αναφερόμαστε στα συστατικά μέρη που το αποτελούν.
7. Για να μπορέσουμε να επιλύσουμε ένα πρόβλημα θα πρέπει να γίνει ο καθορισμός _____.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

8. Πρόβλημα είναι μια οποιαδήποτε κατάσταση που πρέπει να αντιμετωπίσουμε.
9. Ο ανθρώπινος εγκέφαλος είναι ένας μηχανισμός επεξεργασίας δεδομένων.
10. Για την παραγωγή πληροφοριών απαιτούνται δεδομένα.
11. Ο υπολογιστής και το πρόβλημα είναι έννοιες αλληλένδετες.
12. Ένα πρόβλημα μπορεί να αναπαρασταθεί είτε διαγραμματικά, είτε φραστικά.

ΚΕΦΑΛΑΙΟ 2ο

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΩΝ

2.1. Πζοσδοκώμενα αποτελέσματα

Ολοκληρώνοντας αυτό το κεφάλαιο, θα έχεις κατανοήσει ακριβώς την έννοια του αλγορίθμου. Θα έχεις συνειδητοποιήσει τη σπουδαιότητα των αλγορίθμων ως μεθοδολογία σκέψης και ως εργαλείο αντιμετώπισης των προβλημάτων. Θα έχεις διαπιστώσει μέσα από τα παρουσιαζόμενα παραδείγματα και από τις ασκήσεις που θα λύσεις, την αναγκαιότητα αλγοριθμικής προσέγγισης κατά τη διαδικασία επίλυσης των προβλημάτων. Θα έχεις μπορέσει να εξασκηθείς στη μορφοποίηση αλγορίθμων με χρήση συγκεκριμένων τεχνικών. Έτσι λοιπόν εισάγεσαι στα "εργαλεία" ανάπτυξης αλγορίθμων, δηλαδή στη μεθοδολογία επίλυσης προβλημάτων με προγραμματισμό.

2.2. Επιπλέον παζαδείγματα

Παράδειγμα 1. Μετατροπή από βαθμούς Φαρενάιτ σε βαθμούς Κελσίου

Η μετατροπή μίας θερμοκρασιακής τιμής από βαθμούς Φαρενάιτ σε βαθμούς Κελσίου γίνεται με βάση τον τύπο

$$C = \frac{5(F-32)}{9}$$

όπου οι μεταβλητές C και F συμβολίζουν τις αντίστοιχες τιμές. Η μετατροπή αυτή γίνεται εύκολα με τον επόμενο αλγόριθμο που έχει ακολουθιακή δομή.

```
Αλγόριθμος Θερμοκρασία
Διάβασε farenheit
celsius ← (farenheit-32) * 5 / 9
Εκτύπωσε celsius
Τέλος Θερμοκρασία
```

Παράδειγμα 2. Υπολογισμός γεωμετρικών μεγεθών

'Εστω ότι δεδομένης του μήκους της ακτίνας θέλουμε να υπολογίσουμε το εμβαδόν του αντίστοιχου κύκλου, το εμβαδόν του τετραγώνου που είναι περιγεγραμμένο στο δεδομένο κύκλο και το μήκος της διαγωνίου του τετραγώνου αυτού. Ο επόμενος αλγόριθμος επιλύει το γεωμετρικό αυτό πρόβλημα, όπου τα ονόματα των μεταβλητών είναι προφανή. Τέλος, διευκρινίζεται ότι ο ακόλουθος αλγόριθμος καλεί έναν αλγόριθμο ονομαζόμενο Ρίζα, που επιστρέφει την τετραγωνική ρίζα ενός θετικού αριθμού.

```
Αλγόριθμος Γεωμετρικός
Διάβασε aktina
emvadon ← 3.14 * aktina * aktina
plevra ← 2 * aktina
tetragwno ← plevra * plevra
diagwnios ← Ρίζα(2 * tetragwno)
Εκτύπωσε emvadon, tetragwno, diagwnios
Τέλος Γεωμετρικός
```

Παράδειγμα 3. Τιμές θερμοκρασίας από Μετεωρολογικό Κέντρο

Σε ένα μετεωρολογικό κέντρο χρειάζεται να βρεθεί η μέγιστη και η ελάχιστη θερμοκρασία από τις μέσες ημερήσιες θερμοκρασίες ενός μήνα. Να γραφεί ένας αλγόριθμος που θα διαβάζει τη μέση ημερήσια θερμοκρασία για κάθε ημέρα ενός μήνα 30 ημερών και θα υπολογίζει την ελάχιστη και τη μέγιστη από αυτές τις θερμοκρασίες

Για τον υπολογισμό ελάχιστης και μέγιστης θερμοκρασίας είναι βασικό να δοθούν αρχικές τιμές στις μεταβλητές που θα κρατήσουν τις τιμές για να μπορεί να γίνει σωστά η σύγκριση. Εάν για παράδειγμα στη μεταβλητή MIN δώσουμε αρχική τιμή 0, δεν θα καταλήξουμε σε σωστή ελάχιστη θερμοκρασία, εφ' όσον στο μήνα δεν υπάρχουν αρνητικές θερμοκρασίες. Αντίθετα εάν στο MAX δώσουμε αρχική τιμή 0, δεν θα καταλήξουμε σε σωστή μέγιστη θερμοκρασία, στην περίπτωση που όλος ο μήνας είχε καθημερινή αρνητική μέση θερμοκρασία. Επομένως είναι χρήσιμο η MIN να έχει αρκετά υψηλή θερμοκρασία ως αρχική τιμή, ενώ αντίθετα η MAX να έχει αρκετά χαμηλή θερμοκρασία ως αρχική τιμή.

```
Αλγόριθμος  Ελάχιστη_Μέγιστη1
MIN ← 100
MAX ← -100
Για i από 1 μέχρι 30
   Διάβασε THEP
   Αν THEP < MIN τότε MIN ← THEP
   Αν THEP > MAX τότε MAX ← THEP
Τέλος_επανάληψης
Αποτελέσματα // MIN, MAX//
Τέλος Ελάχιστη_Μέγιστη1
```

Παράδειγμα 4. Επίλυση δευτεροβάθμιας εξίσωσης

Η περίπτωση της δευτεροβάθμιας εξίσωσης είναι παρόμοια. Αρχικά είναι απαραίτητο η τιμή του Α να είναι μη μηδενική, πράγμα που ελέγχεται κατά την είσοδο. Στη συνέχεια, για την εύρεση πραγματικών ριζών της εξίσωσης $Ax^2+Bx+\Gamma=0$, πρέπει να ελεγχθεί αν η διακρίνουσα είναι θετική. Και πάλι καλείται ο αλγόριθμος Ρίζα, που επιστρέφει την τετραγωνική ρίζα ενός θετικού αριθμού.

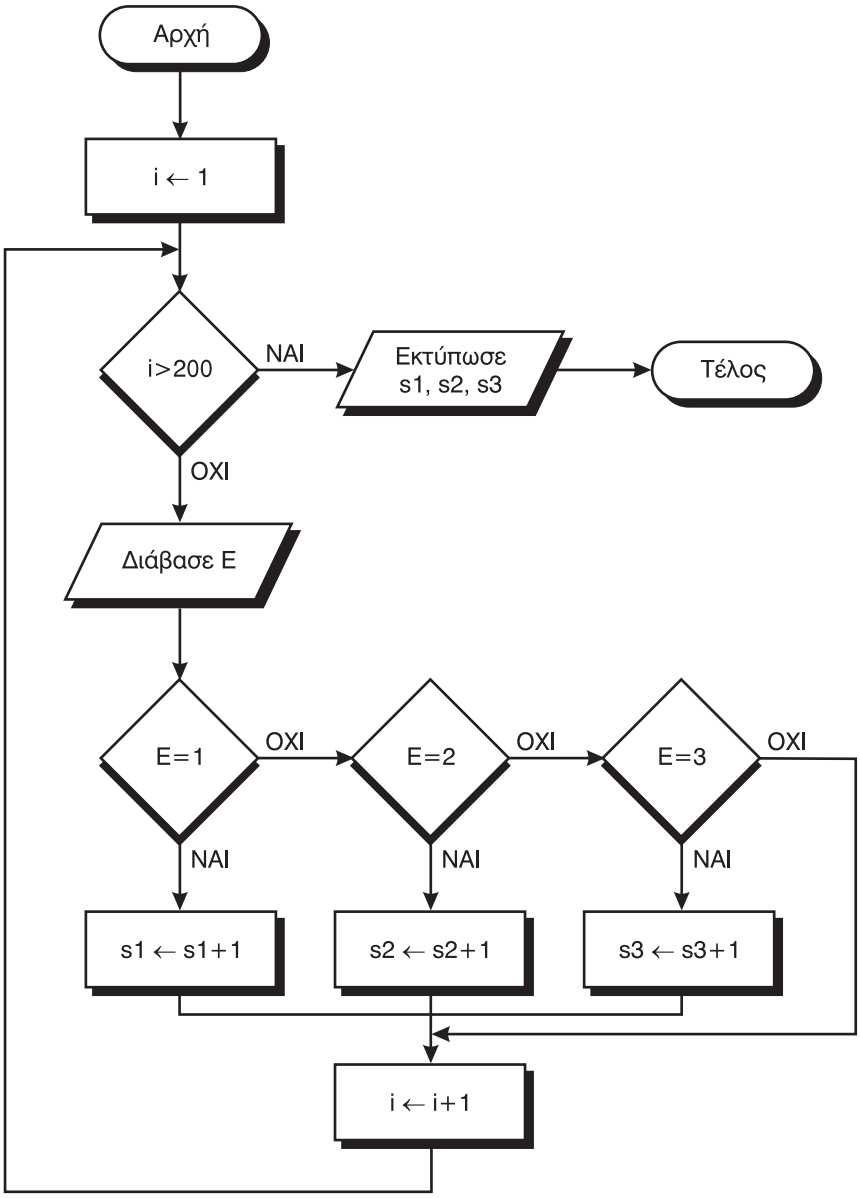
```
Αλγόριθμος ΕξίσωσηΒ
Αρχή_επανάληψης
   Διάβασε a
Μέχρις ότου a≠0
Διάβασε b
Διάβασε c
delta ← b*b-4*a*c
Αν delta ≥ 0 τότε
   solution1 ← (-b+Ρίζα(delta))/(2*a)
   solution2 ← (-b-Ρίζα(delta))/(2*a)
   Εκτύπωσε solution1,solution2
Τέλος_αν
Τέλος ΕξίσωσηΒ
```

Παράδειγμα 5. Φοίτηση στο Πανεπιστήμιο

Σε κάποια Σχολή υπάρχει ένα 3ετές Τμήμα με διαφορετικό αριθμό φοιτητών / φοιτητριών ανά έτος φοίτησης. Συνολικά το Τμήμα αυτό έχει 200 φοιτητές. Να σχεδιασθεί ένα διάγραμμα ροής και να γραφεί ένας αλγόριθμος που θα διαβάζει το έτος κάθε φοιτητή του Τμήματος και θα υπολογίζει τον αριθμό των φοιτητών για κάθε έτος φοίτησης.

Είναι χρήσιμο εδώ να χρησιμοποιηθεί η διαδικασία των πολλαπλών επιλογών διότι είναι ένα πρόβλημα όπου χρειάζεται να γίνει ξεχωριστός υπολογισμός για τις διακριτές τιμές 1, 2, 3 που είναι τα έτη φοίτησης στο συγκεκριμένο Τμήμα.

Διάγραμμα ροής



Αλγόριθμος

```
Αλγόριθμος  Φοιτητές_Ετος
s1 ← 0 s2 ← 0 s3 ← 0
Για i από 1 μέχρι 200
    Διάβασε Ε
    Αν Ε = 1 τότε s1 ← s1+1
    αλλιώς_αν Ε = 2 τότε s2 ← s2+1
    αλλιώς_αν Ε = 3 τότε s3 ← s3+1
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // s1, s2, s3 //
Τέλος Φοιτητές_Ετος
```

Παράδειγμα 6. Διοφαντική ανάλυση

Να εκπονηθεί ένας αλγόριθμος για την εύρεση όλων των ακεραίων λύσεων της εξίσωσης

$$3x + 2y - 7z = 5$$

για τιμές των x, y, z μεταξύ των 0 και 100. Η επίλυση τέτοιων εξισώσεων με πολλές μεταβλητές που επιδέχονται πολλές λύσεις, ονομάζεται διοφαντική ανάλυση. Αλγοριθμικά το πρόβλημα αντιμετωπίζεται ως εξής.

```
Αλγόριθμος Διοφαντική
Για x από 0 μέχρι 100
    Για y από 0 μέχρι 100
        Για z από 0 μέχρι 100
            Αν 3x+2y-7z=5 τότε Εκτύπωσε x,y,z
        Τέλος_επανάληψης
    Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος Διοφαντική
```

2.3. Συμβουλές - υποδείξεις

Από την αρχή της ενασχόλησής σου με τους αλγόριθμους, είναι χρήσιμο να μάθεις να ακολουθείς κάποιους κανόνες και κάποιες γενικές αρχές, έτσι ώστε να μπορείς να λύσεις πραγματικά προβλήματα με μεθοδικό τρόπο και να βρίσκεις την καλύτερη τεχνική για την επίλυση ενός προβλήματος. Τη σπουδαιότητα των αλγορίθμων καθώς και την αναγκαιότητά τους για την επίλυση προβλημάτων θα την καταλαβαίνεις όλο και καλύτερα όσο τα προβλήματα γίνονται περισσότερο σύνθετα και πολύπλοκα.

- Ο αλγόριθμός σου πρέπει να είναι απλός και να προτείνει την εξυπνότερη δυνατή λύση σε ένα πρόβλημα. Είναι χρήσιμο να προσπαθείς κάθε φορά να εντάξεις ένα πρόβλημα σε ένα σύνολο από διαδοχικά βήματα σε φυσική γλώσσα και στη συνέχεια να καταγράφεις αυτά τα βήματα σε κάποια αλγοριθμική δομή.
- Θα πρέπει να χρησιμοποιείς επαναληπτικές δομές για προβλήματα στα οποία μία ακριβώς ίδια ενέργεια γίνεται για ένα σύνολο από παρόμοιες οντότητες (π.χ. για 100 μαθητές, για 20 αυτοκίνητα κλπ). Είναι χρήσιμο να αναγνωρίσεις την αλγοριθμική δομή που βολεύει ανάλογα με την εκφώνηση του προβλήματος.

2.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Ο υπολογισμός της περιόδου του εκκρεμούς δίνεται από τον τύπο:

$$T = 2\pi\sqrt{\frac{L}{g}}$$

όπου L είναι το μήκος του εκκρεμούς και g είναι η επιτάχυνση της βαρύτητας. Να γραφεί αλγόριθμος που να υλοποιεί τον τύπο αυτό.

ΔΤ2 . Να γράψετε με βήματα αλγορίθμου τη διαδικασία μετατροπής των παρακάτω νομισμάτων σε ευρώ, με δεδομένο ότι έχετε τις παρακάτω πληροφορίες :

1. Το ευρώ έχει τιμή πώλησης 330 δρχ.
2. Η λίρα Αγγλίας έχει τιμή πώλησης 550 δρχ.
3. Το δολάριο Αμερικής έχει τιμή πώλησης 280 δρχ.
4. Το μάρκο Γερμανίας έχει τιμή πώλησης 100 δρχ.

Στη συνέχεια να υπολογίσετε σε δραχμές το σύνολο από 1025 λίρες Αγγλίας, 2234 δολάριο Αμερικής και 3459 μάρκα Γερμανίας

ΔΤ3 . Να γράψετε με βήματα αλγορίθμου και με διάγραμμα ροής τα παρακάτω

1. Το μέσο όρο ηλικιών μίας ομάδας 100 ανθρώπων.
2. Το σύνολο βαθμολογίας όλων των ομάδων που έχουν πάρει περισσότερο από 100 βαθμούς σε ένα διαγωνισμό.

ΔΤ4. Τι τύπου αλγοριθμική συνιστώσα πρέπει να χρησιμοποιήσετε για τα παρακάτω στοιχεία υπολογισμού ; Γράψετε το αντίστοιχο τμήμα δηλώσεων.

1. Το σύνολο ποσού για μία λίστα από 100 αντικείμενα.
2. Τη βαθμολογία ενός μαθητή εάν έχει περάσει τα μαθήματά του
3. Το μέσο όρο βαθμολογίας 100 μαθητών.
4. Διάβασε όνομα και τηλέφωνο ενός μαθητή.
5. Διάβασε όνομα, διεύθυνση και τηλέφωνο 25 μαθητών.
6. Τον αριθμό που προκύπτει όταν ρίξουμε ένα ζάρι.

ΔΤ5. Να διαβάζονται δύο αριθμοί που αντιστοιχούν στο ποσοστό του διοξειδίου του άνθρακα και του αζώτου μίας ημέρας, όπως έχει καταγραφεί στα ειδικά μηχανήματα

καταγραφής στην ατμόσφαιρα της πόλης. Να εκτυπώνεται ότι η ατμόσφαιρα είναι «καθαρή», αν το ποσοστό του διοξειδίου του άνθρακα είναι κάτω από 0.35, ή να εκτυπώνεται «μολυσμένη» στην αντίθετη περίπτωση. Επίσης να εκτυπώνεται «διαυγής», αν το άζωτο είναι κάτω από 0.17, αλλιώς να εκτυπώνεται «αδιαυγής».

ΔΤ6. Εστω ότι ένας Πανελλήνιος Διαγωνισμός στα Μαθηματικά δίνει δικαίωμα συμμετοχής στο 1% των μαθητών μίας τάξης με την προϋπόθεση ότι ο μέσος όρος της βαθμολογίας στα Μαθηματικά των μαθητών αυτής της τάξης είναι μεγαλύτερος από 18. Να γραφεί ένας αλγόριθμος που θα ελέγχει τη δυνατότητα συμμετοχής σε έναν τέτοιο διαγωνισμό και να παρακολουθήσετε τον αλγόριθμο για τα δεδομένα της τάξης σας.

ΔΤ7. Οι υπάλληλοι μίας εταιρείας συμφώνησαν για το μήνα Δεκέμβριο να κρατηθούν από το μισθό τους δύο ποσά, ένα για την ενίσχυση του παιδικού χωριού SOS και ένα για την ενίσχυση των σκοπών της UNICEF. Ο υπολογισμός του ποσού των εισφορών εξαρτάται από τον αρχικό μισθό του κάθε υπαλλήλου και υπολογίζεται με βάση τα παρακάτω όρια μισθών :

Μισθός	Εισφορά 1	Εισφορά 2
Εως 150.000 δρχ	5%	4%
150.001 – 250.000	7.5%	6%
250.001 – 400.000	9,5%	8%
μεγαλύτερο από 400.000	12%	11%

Να γραφεί αλγόριθμος που να δέχεται ως είσοδο το μισθό του και στη συνέχεια να υπολογίζει το ποσό των δύο εισφορών και το καθαρό ποσό που θα πάρει ο υπάλληλος.

ΔΤ8. Σε 10 σχολεία της περιφέρειας έχουν εγκατασταθεί πειραματικά 10 ηλεκτρονικοί υπολογιστές (εξυπηρέτες) που περιέχουν πληροφοριακές «σελίδες» του Internet και μπορεί να προσπελάσει κανείς την πληροφορία τους μέσα από οποιοδήποτε ηλεκτρονικό υπολογιστή στον κόσμο. Να γραφεί ένας αλγόριθμος που θα διαβάζει τον συνολικό αριθμό των προσπελάσεων που πραγματοποιήθηκε σε κάθε έναν από τους εξυπηρέτες αυτούς για διάστημα μιας ημέρας. Να βρεθεί ο εξυπηρέτης με το μικρότερο αριθμό προσπελάσεων καθώς και ο εξυπηρέτης με το μεγαλύτερο αριθμό προσπελάσεων.

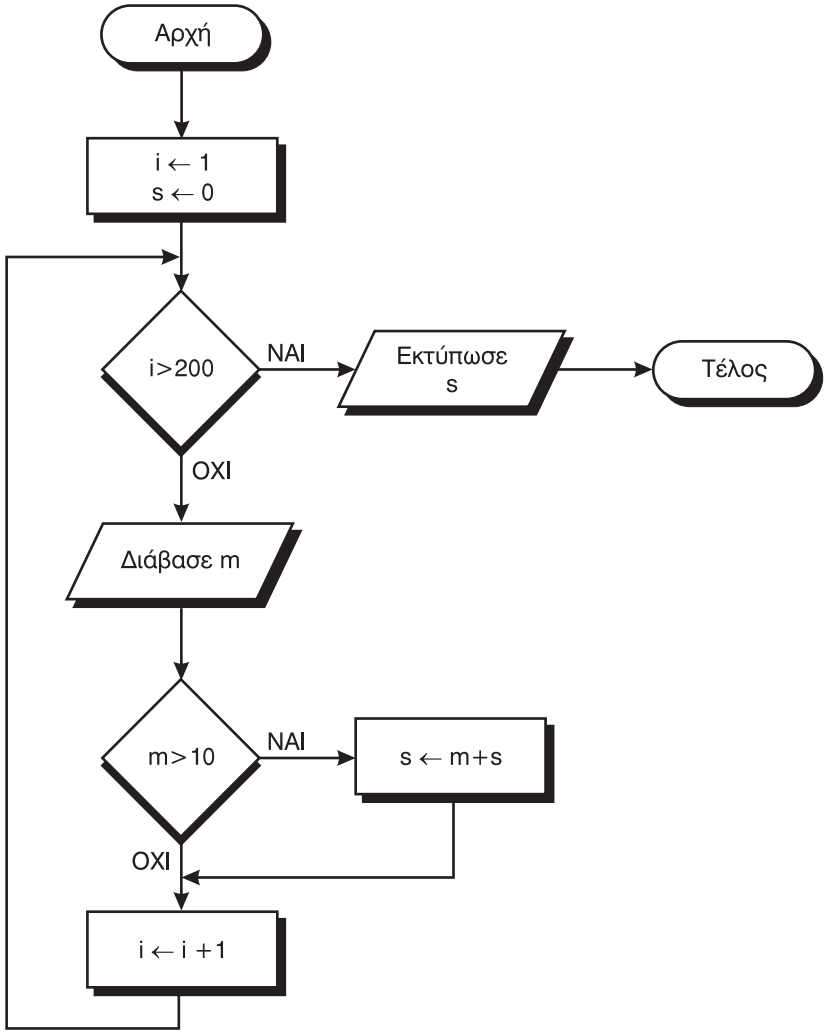
ΔΤ9. Σε ένα φυτώριο υπάρχουν 3 είδη δένδρων που θα δοθούν για δενδροφύτευση. Το $1^{ο}$ είδος δένδρου θα δοθεί στην περιοχή της Μακεδονίας, το $2^{ο}$ στην περιοχή της Θράκης, και το $3^{ο}$ είδος στην περιοχή της Πελοποννήσου. Να σχεδιασθεί το διάγραμμα ροής και να γραφεί ένας αλγόριθμος που θα διαβάζει τον αριθμό του είδους του δένδρου και θα εκτυπώνει την περιοχή στην οποία θα γίνει η δενδροφύτευση.

ΔΤ10. Σε ένα μουσείο υπάρχουν 10 διαφορετικές αίθουσες που περιέχουν διάφορα έργα της ελληνιστικής περιόδου. Κάθε αίθουσα έχει το δικό της αριθμό που είναι από 101, 102, ..., έως 110. Να γράψεις έναν αλγόριθμο που θα διαβάζει τον αριθμό των επισκεπτών κάθε αίθουσας για μία ημέρα και θα υπολογίζει το μέσο όρο των επισκεπτών από όλες τις αίθουσες. Στη συνέχεια ο αλγόριθμος θα πρέπει να εκτυπώνει τους αριθμούς των αιθουσών που είχαν περισσότερους επισκέπτες από το μέσο όρο των επισκεπτών.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Δίνεται το παρακάτω διάγραμμα ροής :



Να δώσετε την εκφώνηση του προβλήματος που εκφράζεται με το συγκεκριμένο διάγραμμα ροής.

ΔΣ2. Εστω ότι σου έχουν δώσει ένα μεταχειρισμένο ηλεκτρονικό υπολογιστή για 6 μήνες. Θέλεις να διαπραγματευτείς την τιμή αυτού του υπολογιστή για να δεις αν μπορείς να τον αλλάξεις με κάποιο άλλο μοντέλο. Η αρχική τιμή του υπολογιστή που πήρες είναι 295.600 δρχ. και σου τον προσφέρουν για 256.000 δρχ. Είναι χρήσιμο να να υπολογίσεις το ποσοστό της απαξίωσης για τον υπολογιστή αυτό δεδομένου ότι το ετήσιο ποσοστό υποτίμησης υπολογίζεται από τον παρακάτω τύπο :

$$\text{Ποσοσό\_Απαξίωσης} = 1 - \left(\frac{\textit{Τιμή\_προσφοράς}}{\textit{Αρχική\_Τιμή}} \right)^{\frac{1}{\textit{Αριθμός\_ετών}}}$$

Να σχεδιασθεί το διάγραμμα ροής και να γραφεί ένας αλγόριθμος που θα υπολογίζει το ποσοστό απαξίωσης για τον υπολογιστή που πήρες για τους 6 μήνες. Στη συνέχεια να γενικεύσεις τον αλγόριθμο, έτσι ώστε να δουλεύει επαναληπτικά για έναν αριθμό από διαφορετικά είδη των οποίων ξέρεις το αρχικό ποσό, το ποσό της προσφοράς και το χρονικό διάστημα για το οποίο θέλεις να υπολογίσεις τα ποσοστά απαξίωσης.

ΔΣ3. Ενας καταναλωτής πηγαίνει στο πολυκατάστημα και έχει στη τσέπη του 5.000 ευρώ. Ξεκινά να αγοράζει διάφορα είδη και ταυτόχρονα κρατά το συνολικό ποσό στο οποίο έχει φθάσει κάθε στιγμή που αγοράζει κάποιο είδος. Οι τιμές των ειδών που αγοράζει είναι σε δραχμές και είναι δεδομένο ότι 1 ευρώ=330 δραχμές. Να γραφεί σε φυσική γλώσσα, με ακολουθία βημάτων και με διάγραμμα ροής ένας αλγόριθμος για τον υπολογισμό του ποσού από τα ψώνια που έγιναν και να σταματά η αγορά ειδών έτσι ώστε να μην ξεπεραστεί το ποσό που έχει διαθέσιμο ο καταναλωτής.

ΔΣ4. Δίνεται ο παρακάτω αλγόριθμος :

```
Αλγόριθμος  Ελεγχος_Ανάθεσης
Διάβασε x
Όσο x > 1 επανάλαβε
    Αν x είναι άρτιος τότε
        x ← x/2
    αλλιώς
        x ← 3*x+1
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα  // x //
Τέλος Ελεγχος_Ανάθεσης
```

Να γράψεις τα αποτελέσματα αυτού του αλγορίθμου για x=13, x=9 και x=22. Τι παρατηρείς ;

ΔΣ5. Σε ένα Λύκειο κάθε μαθητής αξιολογείται με βάση το μέσο όρο που θα έχει σε 5 βασικά μαθήματα. Να γραφεί ένας αλγόριθμος που θα διαβάζει τη βαθμολογία για καθένα από τα 5 αυτά μαθήματα και θα υπολογίζει το μέσο όρο του μαθητή.

Να αναλυθεί το πρόβλημα και να προταθεί λύση του με ακολουθία βημάτων και με διάγραμμα ροής.

Υπόδειξη

Για τον υπολογισμό του συνολικού μέσου όρου η χρήση επαναληπτικής δομής είναι σημαντική λόγω της "ελάφρυνσης" του κώδικα από παρόμοιες εντολές και από χρήση πολλαπλών μεταβλητών.

ΔΣ6. Πηγαίνεις σε ένα πολυκατάστημα και παρατηρείς τις παρακάτω τιμές για 4 διαφορετικά είδη γάλακτος.

Είδος	Τιμή	Ποσότητα
ΓΑΛΑ_Α	195 δρχ	300ml
ΓΑΛΑ_Β	205 δρχ	400ml
ΓΑΛΑ_Γ	400 δρχ	500ml
ΓΑΛΑ_Δ	450 δρχ	550ml

Να γράψεις έναν αλγόριθμο που θα υπολογίζει και θα εμφανίζει το είδος γάλακτος που έχει την πλέον συμφέρουσα τιμή.

ΔΣ7. Εστω ότι θέλεις να υπολογίσεις το ποσό που θα έχεις στο μέλλον με βάση το ποσό που τώρα έχεις αποταμιεύσει στην τράπεζα. Δίνεται ο παρακάτω τύπος υπολογισμού :

$$\text{Τελικό\_Ποσό} = \text{Αρχικό\_Ποσό} \cdot \left(1 + \frac{\frac{\text{επιτόκιο}}{100}}{2}\right)^{2 \cdot \text{χρόνια}}$$

Να γράψεις έναν αλγόριθμο που να υπολογίζει το ποσό που θα έχεις μετά από 5 χρόνια με δεδομένο ότι το ετήσιο επιτόκιο είναι 6,5 %. Να επεκτείνεις τον αλγόριθμο έτσι ώστε να υπολογίζει το ποσό που θα έχεις για 5 διαφορετικά ποσά που έχει κρατήσει σε ξεχωριστούς τραπεζικούς λογαριασμούς. Να βρεθεί και το τελικό ποσό που θα έχεις από όλους αυτούς τους λογαριασμούς.

ΔΣ8. 'Εστω ότι έχεις να επεκτείνεις το πρόβλημα της δενδροφύτευσης που δόθηκε στις δραστηριότητες για την τάξη (ΔΤ9). Να επεκτείνεις τον αλγόριθμο έτσι ώστε να διαβάζεις ένα σύνολο από 100 τιμές που αφορούν το είδος του δένδρου και να υπολογίζεις πόσα από τα δένδρα αυτά θα φυτευτούν στη Μακεδονία, πόσα στη Θράκη και πόσα στην Πελοπόννησο.

ΔΣ9. 'Εστω ότι θέλεις να οργανώσεις μία εκδήλωση για την παγκόσμια ημέρα περιβάλλοντος και έχεις τη χωρητικότητα (σε αριθμό ατόμων) και τις τιμές που θα κοστίσει η ενοικίαση χώρου από 3 διαφορετικούς χώρους στους οποίους μπορεί να γίνει η εκδήλωση. Επιπλέον έχεις προσφορές από 5 διαφορετικούς χορηγούς που διαθέτουν

χρήματα για την υποστήριξη της εκδήλωσης. Να γραφεί ένας αλγόριθμος που θα υπολογίζει πόσοι χορηγοί μπορούν να καλύψουν το κόστος της αίθουσας με τη δυνατή μεγαλύτερη χωρητικότητα.

2.5. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες προτάσεων. Σε κάθε μία από αυτές, να κάνετε τις απαραίτητες διορθώσεις ώστε να ισχύουν οι προτάσεις

1. Η αναπαράσταση αλγορίθμου με ελεύθερο κείμενο (free text) αποτελεί τον πιο καλά δομημένο τρόπο παρουσίασης αλγορίθμου.
2. Τα διαγράμματα ροής (flow charts) αποτελούν έναν ακολουθιακό τρόπο παρουσίασης ενός αλγορίθμου με χρήση βημάτων.
3. Η κωδικοποίηση (coding) ενός αλγορίθμου γίνεται με ένα πρόγραμμα που όταν εκτελεσθεί μπορεί και να μη δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

4. Η _____ δομή (σειριακών βημάτων) χρησιμοποιείται πρακτικά για την αντιμετώπιση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών.
5. Η δομή της _____ χρησιμοποιείται όταν υπάρχει αναγκαιότητα απόφασης μεταξύ ενός συνόλου περιπτώσεων.
6. Η _____ ενός αλγορίθμου γίνεται με ένα πρόγραμμα που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.
7. Τα _____ αποτελούν ένα γραφικό τρόπο παρουσίασης ενός αλγορίθμου.
8. Οι _____ διαδικασίες συνδυάζουν και χρησιμοποιούν περισσότερες από μία περιπτώσεις αλγοριθμικών συνιστωσών.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

9. Η αλγοριθμική υποστήριξη βοηθά στην επίλυση προβλημάτων.
10. Οι αλγοριθμικές δομές αποτελούνται από ένα ενιαίο κομμάτι και διαφέρουν μόνο στα στοιχεία εισόδου.
11. Για τον υπολογισμό ενός αθροίσματος ακεραίων μπορώ να χρησιμοποιήσω τη δομή της επιλογής.
12. Οι διαδικασίες πολλαπλών επιλογών χρησιμοποιούνται για τις διαφορετικές ενέργειες που πρέπει να γίνουν με βάση τον αριθμό των διακριτών ακεραίων τιμών μίας μεταβλητής.

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

13. Τα χρησιμοποιούμενα γεωμετρικά σχήματα για την αναπαράσταση των διαγραμμάτων ροής είναι τα εξής :

    Α) έλλειψη

    Β) ρόμβος

    Γ) ορθογώνιο

    Δ) κύκλος

14. Ποιά από τα παρακάτω είναι δεκτά ως αλγοριθμικές δομές :

    Α) επιλογή

    Β) εκτύπωση

    Γ) ανάγνωση

    Δ) υπολογισμός

    Ε) επανάληψη

Βάλε έναν κύκλο στα σωστά

15. Οι αλγοριθμικές συνιστώσες περιλαμβάνουν :

    Α) Επιλογή

    Β) Επανάληψη

    Γ) Ανάγνωση

    Δ) Πολλαπλή Εκτύπωση

16. Ο πολλαπλασιασμός αλά ρωσικά περιλαμβάνει :

    Α) πολλαπλασιασμό επί 4

    Β) πολλαπλασιασμό επί 2

    Γ) διαίρεση δια 4

    Δ) διαίρεση δια 2

17. Η Πληροφορική είναι η επιστήμη που μελετά τους αλγορίθμους από τις ακόλουθες σκοπιές :

    Α) Υλικού

    Β) Θεωρητική

    Γ) Πιθανολογική

    Δ) Αναλυτική

3.1. Πζοσδοκώμενα αποτελέσματα

Στο τέλος αυτού του κεφαλαίου προσδοκάται ότι θα έχεις συνειδητοποιήσει τη σπουδαιότητα των δεδομένων για την επίλυση ενός προβλήματος. Θα έχεις ενστερνισθεί τη θεώρηση ότι οι αλγόριθμοι και οι δομές δεδομένων αποτελούν αδιάσπαστη ενότητα. Θα μπορείς να χειρίζεσαι με ευχέρεια προβλήματα σχετικά με εργασίες με πίνακες. Ακόμα θα μπορείς να κάνεις μια περιληπτική αναφορά σε άλλες δομές δεδομένων (στοίβα, ουρά, λίστα, δένδρο). Τέλος εκτιμάται ότι θα έχεις κατανοήσει τη λειτουργία της αναδρομής. Έτσι έρχεσαι σε επαφή με ένα πανόραμα δομών και αλγορίθμων, που αποτελεί ένα ικανοποιητικό σύνολο εργαλείων για την επίλυση πρακτικών προβλημάτων.

3.2. Επιπλέον παζαδείγματα

Παράδειγμα 1. Υπολογισμός μέγιστου ποσού

Σε μία εταιρεία εργάζονται 200 υπάλληλοι και είναι γνωστός ο μισθός του καθενός. Να χρησιμοποιηθεί η δομή του πίνακα για να αποθηκεύονται οι μισθοί των υπαλλήλων και να βρεθεί ο κατάλληλος αλγόριθμος υπολογισμού του μεγαλύτερου μισθού.

```
Αλγόριθμος  Μεγαλύτερος_Μισθός
Διάβασε  MIS[1]
MAX ← MIS[1]
Για i από 2 μέχρι  200
   Διάβασε MIS[i]
   Αν MIS[i] > MAX τότε
      MAX ← MIS[i]
   Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // MAX//
Τέλος Μεγαλύτερος_Μισθός
```

Ένα παρόμοιο παράδειγμα είχε χρησιμοποιηθεί στο προηγούμενο κεφάλαιο (παράδειγμα 3). Η δομή του πίνακα χρησιμοποιείται ώστε οι μισθοί των υπαλλήλων να αποθηκεύονται στις θέσεις του πίνακα και να μπορούν να χρησιμοποιηθούν και στη συνέχεια. Αντίθετα, στο προηγούμενο κεφάλαιο δεν αποθηκεύονταν κάπου οι τιμές των θερμοκρασιών αλλά απλά χρησιμοποιούνταν στις συγκρίσεις για την αναγνώριση της μέγιστης θερμοκρασίας.

Παράδειγμα 2. Υπολογισμός αριθμού συνδυασμών

Είναι γνωστό από τα μαθηματικά ότι ο αριθμός των συνδυασμών των n πραγμάτων ανά k δίνεται από τον τύπο

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Να προταθεί ένας αλγόριθμος για τον υπολογισμό του αριθμού των συνδυασμών αυτών. Ένας απλός τρόπος είναι να εφαρμοσθεί ο προηγούμενος μαθηματικός τύπος. Έτσι δεδομένων των παραλλαγών του αλγορίθμου για τον υπολογισμό του n!, προκύπτει ο επόμενος αλγόριθμος που καλεί κάποια από αυτές τις παραλλαγές.

```
Αλγόριθμος Συνδυασμός
Διάβασε n
Διάβασε k
a ← Παραγοντικό(n)
b ←  Παραγοντικό(k)
c ← Παραγοντικό(n-k)
combination ← a /(b*c)
Γραψε combination
Τέλος Συνδυασμός
```

Αν και ο αλγόριθμος αυτός είναι ιδιαίτερα απλός στην κατανόηση και στον προγραμματισμό του, εντούτοις δεν είναι ο καλύτερος δυνατός από την άποψη της αποτελεσματικότητας γιατί εκτελεί περιττούς πολλαπλασιασμούς. Αυτό φαίνεται ανάγλυφα αν θεωρήσουμε και πάλι το μαθηματικό τύπο του αριθμού των συνδυασμών. Στο κλάσμα του τύπου αυτού μπορεί να γίνει κάποια απλοποίηση μεταξύ αριθμητή και παρονομαστή. Για παράδειγμα, για τον υπολογισμό του αριθμού των συνδυα-

σμών των 10 πραγμάτων ανά 5, με τον προηγούμενο τρόπο θα εκτελέσουμε 9 πολλαπλασιασμούς για τον υπολογισμό του αριθμητή (10!), και 4+4 πολλαπλασιασμούς για τον υπολογισμό του παρονομαστή (δύο φορές το 5!). Ενώ εύκολα προκύπτει ότι

$$\frac{10!}{5!5!} = \frac{10 \cdot 9 \cdot 8 \cdot 7 \cdot 6}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5}$$

όπου εκτελούνται από τέσσερις πολλαπλασιασμοί σε αριθμητή και παρονομαστή. Ο αντίστοιχος αλγόριθμος έχει ως εξής.

```
Αλγόριθμος Συνδυασμός2
Διάβασε n
Διάβασε k
a ← 1
Για i από n μέχρι n-k+1 με_βήμα -1
    a ← a*i
Τέλος_επανάληψης
b ← Παραγοντικό(k)
combination ← a/b
Εκτύπωσε combination
Τέλος Συνδυασμός2
```

Παράδειγμα 3. Υπολογισμός μέσου όρου

Σε ένα Λύκειο υπάρχουν τρία τμήματα για την Γ′ Λυκείου και κάθε τμήμα έχει 35 μαθητές. Να γραφεί ένας αλγόριθμος που θα διαβάζει το μέσο όρο βαθμολογίας κάθε μαθητή και θα υπολογίζει το γενικό μέσο όρο βαθμολογίας για όλη την τάξη της Γ′ Λυκείου.

Ο αλγόριθμος που ακολουθεί υπολογίζει τον παραπάνω μέσο όρο με χρήση της δομής του πίνακα.

```
Αλγόριθμος  Μέσος_Ορος
S ← 0
Για i από 1 μέχρι 105
    Διάβασε M[i]
    S ← S+M[i]
Τέλος_επανάληψης
MO ← S/105
Αποτελέσματα //  MO //
Τέλος Μέσος_Ορος
```

Παράδειγμα 4. Χρήση δισδιάστατων πινάκων

Έστω ότι δίνονται δύο δισδιάστατοι πίνακες Α και Β διαστάσεων 5Χ5 ο καθένας. Να γραφεί ένας αλγόριθμος που θα διαβάζει τα στοιχεία των πινάκων και θα υπολογίζει το άθροισμα των πινάκων, το οποίο θα αποθηκεύεται σε ένα νέο πίνακα.

```
Αλγόριθμος Αθροισμα_Πινάκων
Για i από 1 μέχρι 5
   Για  j από 1 μέχρι 5
      Διάβασε A[i,j], B[i,j]
      C[i,j] ← A[i,j] + B[i,j]
   Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // C //
Τέλος Αθροισμα_Πινάκων
```

Παράδειγμα 5. Αραιοί πίνακες

Ένας πίνακας λέγεται ***αραιός*** (sparse) αν ένα μεγάλο ποσοστό των στοιχείων του έχουν μηδενική τιμή. Δεν υπάρχει ακριβές ποσοστό σε σχέση με τον αριθμό των μηδενικών στοιχείων, επάνω από το οποίο ένας πίνακας χαρακτηρίζεται ως αραιός. Αρκεί όμως, για παράδειγμα, να πούμε ότι με περισσότερο από 80% μηδενικά ένας πίνακας χαρακτηρίζεται ως αραιός.

Αραιοί πίνακες συναντώνται συχνά σε μεγάλα επιστημονικά προβλήματα (επίλυση εξισώσεων κλπ). Το πρόβλημα με τη διαχείριση των αραιών πινάκων είναι ότι δαπανάται πολύ χώρος για την αποθήκευση μηδενικών. Άρα πρέπει να βρεθεί ένας οικονομικός τρόπος αποθήκευσης των αραιών πινάκων. Στην πράξη έχουν προταθεί αρκετοί τρόποι. Ένας από αυτούς τους τρόπος περιγράφεται στη συνέχεια. Έστω, λοιπόν, ότι δίνεται ο επόμενος πίνακας, που θέλουμε να τον διαχειρισθούμε ως αραιό.

0	7	0	0	0
1	2	0	0	-3
0	0	4	0	0
12	0	0	0	0

Αντί να αποθηκεύσουμε αυτόν το δισδιάστατο πίνακα 4x5, θα θεωρήσουμε ένα μονοδιάστατο πίνακα όπου θα τοποθετήσουμε μόνο τα μη μηδενικά στοιχεία, για τα οποία όμως χρειαζόμαστε τα στοιχεία των αντίστοιχων γραμμών και στηλών. Έτσι καταλήγουμε κάθε μη μηδενικό στοιχείο να αντιπροσωπεύεται από μία τριάδα στοιχείων, δηλαδή <γραμμή,στήλη,τιμή>. Για το λόγο αυτό δημιουργούμε ένα μονοδιάστατο πίνακα 18 θέσεων για τα 6 μη μηδενικά στοιχεία του αρχικού πίνακα. Ο νέος πίνακας έχει τη μορφή

1	2	7	2	1	1	2	2	2	2	5	-3	3	3	4	4	1	12

Πλέον, το πρόβλημα έγκειται στην αναγνώριση της τιμής μίας θέσης του παλαιού πίνακα, δεδομένου ότι ο πίνακας είναι αποθηκευμένος με τη νέα του μορφή. Ο επόμενος αλγόριθμος “Αραιός” επιστρέφει την τιμή του στοιχείου που βρίσκεται στη θέση <γραμμή l, στήλη m> του αρχικού πίνακα επεξεργαζόμενος τη νέα μορφή του πίνακα που αποτελείται από 3n θέσεις, όπου n ο αριθμός των μη μηδενικών στοιχείων.

```
Αλγόριθμος Αραιός
Δεδομένα // sparse, n //
flag ← 0
k ← 0
Οσο flag=0 επανάλαβε
    i ← sparse[3*k+1]
    j ← sparse[3*k+2]
    Αν i=L και j=M τότε
        result ← sparse[3*k+3]
        flag ← 1
    αλλιώς_αν i > L ή (i=L και j > M) τότε
        result ← 0
        flag ← 1
    αλλιώς
        k ← k+1
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // result //
Τέλος Αραιός
```

3.3. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Σε μία κατασκήνωση υπάρχουν 300 παιδιά και καθένα από αυτά έχει μοναδικό αριθμό από το 1 έως και το 300 που του αντιστοιχεί. Για κάθε παιδί είναι γνωστή η ηλικία του. Να χρησιμοποιηθεί η δομή του πίνακα για να αποθηκεύονται οι ηλικίες των παιδιών και να βρεθεί ο κατάλληλος αλγόριθμος υπολογισμού του μικρότερου και μεγαλύτερου σε ηλικία παιδιού και να εκτυπώνεται τόσο η ηλικία όσο και ο κωδικός του μικρότερου και μεγαλύτερου παιδιού.

ΔΤ2. Ο αλγόριθμος της φυσσαλίδας όπως διατυπώθηκε στην παράγραφο 3.7 έχει το μειονέκτημα ότι δεν είναι αρκετά "έξυπνος" ώστε να διαπιστώνει στην αρχή ή στο μέσο της διαδικασίας αν ο πίνακας είναι ταξινομημένος. Να σχεδιασθεί μία παραλλαγή του αλγορίθμου αυτού που να σταματά όταν διαπιστωθεί ότι τα στοιχεία του πίνακα είναι ήδη ταξινομημένα.

Υπόδειξη: Να χρησιμοποιήσετε μία βοηθητική μεταβλητή που να ελέγχει το τέλος κάθε επανάληψης του εξωτερικού βρόχου ("Για i από 2 μέχρι n") αν για την τρέχουσα τιμή του i έγιναν αντιμεταθέσεις στοιχείων.

ΔΤ3. Να δοθούν οι αλγόριθμοι Ώθηση (Push) και Απώθηση (Pop) που αντίστοιχα εκτελούν τις προφανείς λειτουργίες σε μία στοίβα. Να δοθεί ένα παράδειγμα, στο οποίο να χρησιμοποιείται μία στοίβα από ακέραιους. Η στοίβα αντιπροσωπεύεται από έναν πίνακα μέχρι 100 θέσεων.

ΔΤ4. Να δοθούν οι αλγόριθμοι Εισαγωγή_σε_Ουρά (Enqueue) και Εξαγωγή_από_Ουρά (Dequeue) που αντίστοιχα εκτελούν τις προφανείς λειτουργίες σε μία ουρά. Να δοθεί ένα παράδειγμα, στο οποίο να χρησιμοποιείται μία ουρά από ακέραιους. Η ουρά αντιπροσωπεύεται από έναν πίνακα μέχρι 100 θέσεων.

ΔΤ5. Εστω ότι η τάξη σας θα συμμετάσχει στην ημερήσια εθελοντική αιμοδοσία που πραγματοποιεί ο Δήμος της πόλης σας. Είναι γνωστό το επίθετο κάθε μαθητή και όλοι οι μαθητές θα συμμετάσχουν στην αιμοδοσία. Να γραφεί αλγόριθμος για τη δημιουργία ουράς των μαθητών έξω από το Κέντρο αιμοδοσίας με δεδομένο ότι η ουρά θα δημιουργηθεί με βάση την αλφαβητική σειρά των επιθέτων των μαθητών.

ΔΤ6. Μία οικολογική οργάνωση διαθέτει στοιχεία για το ποσοστό δασών για 50 διαφορετικές χώρες. Χρειάζεται να πάρει απόφαση για να διοργανώσει μία εκδήλωση διαμαρτυρίας στις 10 χώρες που έχουν το χαμηλότερο ποσοστό δασών. Να δοθεί αλγόριθμος που θα ταξινομεί τα ποσοστά δασών των χωρών με χρήση της μεθόδου της ευθείας ανταλλαγής και θα εκτυπώνει τις 10 χώρες στις οποίες θα διοργανωθούν οι εκδηλώσεις.

Στο σπίτι

Στο τετράδιό σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Δίνεται ο παρακάτω πίνακας από αντιστοιχίσεις νομισμάτων διαφόρων κρατών:

Νόμισμα Χώρας	Αγορά	Πώληση
Δολλάριο ΗΠΑ	274,161	281,091
Δολλάριο Καναδά	178,238	182,743
Λιρέττα Ιταλίας	0,164	0,172
Φράγκο Γαλλίας	49,442	50,692
Μάρκο Γερμανίας	165,806	169,997

Να γραφεί ένας αλγόριθμος που θα κάνει μετατροπές ενός ποσού από τα ξένα νομίσματα σε δραχμές και από δραχμές στο αντίστοιχο ξένο νόμισμα.

ΔΣ2. Κατά τη διάρκεια ενός πρωταθλήματος μπάσκετ καταγράφεται ο αριθμός των πόντων που έχουν βάλει 5 παίκτες σε 5 διαφορετικά παιχνίδια. Να γραφεί αλγόριθμος που θα σε βοηθήσει να κρατήσεις σε ένα δισδιάστατο πίνακα αυτά τα στοιχεία και στη συνέχεια να υπολογίσεις τον παίκτη που έχει πετύχει το μεγαλύτερο αριθμό πόντων από όλα τα παιχνίδια.

ΔΣ3. Εστω ότι θέλουμε να διατάξουμε τους μαθητές μίας τάξης κατά φθίνουσα σειρά ύψους. Η τεχνική που θα ακολουθήσουμε είναι η εξής. Αρχικά, τοποθετούμε τους μα-

θητές σε μία τυχαία σειρά. Κατόπιν συγκρίνουμε το δεύτερο με τον πρώτο και αν χρειασθεί τους αντιμεταθέτουμε ώστε πρώτος να είναι ο ψηλότερος. Στη συνέχεια θεωρούμε τον τρίτο και τον τοποθετούμε στη σωστή σειρά σε σχέση μεν πρώτο και το δεύτερο. Κατ' αυτόν τον τρόπο συνεχίζουμε μέχρι να τοποθετήσουμε στη σωστή σειρά όλους τους μαθητές. Να σχεδιασθεί ένας αλγόριθμος που να υλοποιεί αυτή τη μέθοδο ταξινόμησης.

ΔΣ4. Ενας μαθητής έχει μία συλλογή από δίσκους CD και για κάθε CD έχει καταγράψει στον υπολογιστή τον τίτλο και την χρονιά έκδοσής του. Να ταξινομηθούν τα CD με βάση την χρονιά τους και να υπολογισθεί ο αριθμός των CD που έχει ο μαθητής με χρονολογία έκδοσης πριν από το 1995.

ΔΣ5. Ας υποθέσουμε ότι έχετε αναλάβει να μοιράσετε ένα σύνολο από βιβλία στους συμμαθητές σας. Αν ορίσετε μία ημέρα για το μοίρασμα των βιβλίων και οι συμμαθητές σας φθάνουν ο ένας μετά τον άλλο φτιάχνοντας μία ουρά πώς θα ρυθμίσετε την είσοδο και την έξοδο τους από την ουρά ; Να δώσετε το σχετικό αλγόριθμο εισαγωγής και εξαγωγής από την ουρά.

3.4. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες προτάσεων. Σε κάθε μία από αυτές, να κάνετε τις απαραίτητες διορθώσεις ώστε να ισχύουν οι προτάσεις

1. Δομή Δεδομένων είναι ένα σύνολο δεδομένων τα οποία δεν υφίσταται επεξεργασία από λειτουργίες, που καλούνται από το υπόλοιπο πρόγραμμα.
2. Οι δυναμικές δομές δεδομένων αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης (dynamic memory allocation).
3. Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση τριών βασικών δομών: της αναδρομής, της στοίβας και της ουράς.

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

4. Η _____ είναι η πράξη κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.
5. Η _____ είναι η πράξη κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.
6. Ο _____ αποτελεί την αντίστροφη πράξη της συγχώνευσης.
7. Δύο είναι οι κύριες λειτουργίες σε μία στοίβα: η _____ στοιχείου στην κορυφή της στοίβας και η _____ στοιχείου από τη στοίβα.
8. Δύο είναι οι κύριες λειτουργίες σε μία ουρά: η _____ στοιχείου στο πίσω άκρο της ουράς και η _____ στοιχείου από το εμπρός άκρο της ουράς.

9. Η τακτοποίηση των κόμβων μίας δομής με μία ιδιαίτερη σειρά είναι μία ιδιαίτερη σημαντική λειτουργία που ονομάζεται _____.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

10. Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: τις στατικές και τις δυναμικές.

11. Δύο είναι οι κύριες λειτουργίες που εκτελούνται σε μία ουρά: εισαγωγή και η διαγραφή.

12. Η τακτοποίηση των κόμβων μίας δομής με μία ιδιαίτερη σειρά είναι μία ιδιαίτερη σημαντική λειτουργία που ονομάζεται εξαγωγή.

13. Η μέθοδος της ταξινόμησης ευθείας ανταλλαγής βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων, μέχρις ότου διαταχθούν όλα τα στοιχεία.

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

14. Οι βασικές λειτουργίες (ή αλλιώς πράξεις) επί των δομών δεδομένων είναι οι ακόλουθες:

Α) Προσπέλαση	Γ) Εισαγωγή	Ε) Αναζήτηση
Β) Ανάγνωση	Δ) Διαγραφή	Ζ) Εκτύπωση
		Η) Ταξινόμηση

15. Η σειριακή μέθοδος αναζήτησης δικαιολογεί τη χρήση της μόνο σε περιπτώσεις όπου:

Α) ο πίνακας είναι αταξινόμητος

Β) ο πίνακας αποτελείται από ακέραιους

Γ) ο πίνακας είναι μικρού μεγέθους

Δ) ο πίνακας δεν είναι δισδιάστατος

Ε) η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια

Ζ) η αναζήτηση γίνεται με βάση την τιμή δευτερεύοντος κλειδιού

Βάλε έναν κύκλο στα σωστά

16. Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση των παρακάτω δομών δεδομένων :

Α) ουράς	Γ) στοίβας
Β) λίστας	Δ) επιλογής

4.1. Πζοσδοκώμενα αποτελέσματα

Ολοκληρώνοντας αυτό το κεφάλαιο προσδοκάται πως θα έχεις λάβει εκείνες τις γνώσεις, ώστε να τεκμηριώνεις την αναγκαιότητα ανάλυσης των προβλημάτων και σχεδίασης των κατάλληλων αλγορίθμων. Θα μπορείς να διατυπώνεις σύγχρονες τεχνικές σχεδίασης αλγορίθμων και να περιγράφεις την ακολουθία βημάτων για την ανάλυση των αλγορίθμων. Ακόμα θα μπορείς να περιγράφεις τις κυριότερες προσεγγίσεις επίλυσης και ανάλυσης προβλημάτων. Τέλος, θα έχεις αποκτήσει εκείνες τις δεξιότητες ώστε να μπορείς να επιλύεις προβλήματα με χρήση των κυριοτέρων προσεγγίσεων.

4.2. Επιπλέον παζαδείγματα

Παράδειγμα 1. Ταξινόμηση με επιλογή

Σε μία τηλεφωνική εταιρεία χρειάζεται να γίνει ταξινόμηση σε αύξουσα σειρά των αριθμών τηλεφώνων με βάση την εξής παρατήρηση : «Από τους αριθμούς τηλεφώνων που δεν έχουν ταξινομηθεί στη σωστή σειρά, να βρεθεί ο μικρότερος αριθμός και να τοποθετηθεί στη σειρά των αριθμών που έχουν ήδη ταξινομηθεί». Να καταγραφεί ο σχετικός αλγόριθμος για την ταξινόμηση 1000 αριθμών τηλεφώνων.

Για να υλοποιήσεις τον αλγόριθμο πρέπει να υπάρξει κάποια απόφαση σχετικά με τις δομές δεδομένων που θα χρειασθείς. Είναι δεδομένο ότι θα έχεις 1000 αριθμούς τηλεφώνων που πρέπει να βάλεις σε σωστή αύξουσα σειρά. Στις περιπτώσεις προβλη-

μάτων όπου είναι δεδομένος ο συνολικός αριθμός από «αντικείμενα» που θα έχει το πρόβλημα χρησιμοποιούμε τη δομή του πίνακα, έτσι ώστε κάθε αντικείμενο να έχει τη δική του θέση. Επομένως γίνεται αρχικά η ανάθεση των αριθμών τηλεφώνων σε κάθε μία από τις θέσεις του πίνακα, με τον τρόπο που έχει περιγραφεί σε προηγούμενα κεφάλαια. Ο παρακάτω αλγόριθμος υποθέτει ότι ο πίνακας THL έχει πάρει τους αριθμούς τηλεφώνων σε τυχαία σειρά και επομένως δεν είναι ταξινομημένος.

```
Αλγόριθμος Αριθμοί_Τηλεφώνων
Δεδομένα // THL//
Για i από 1 μέχρι 1000
    j ← i
    Για k από i+1 μέχρι 1000
       Αν THL[k] < THL[j] τότε
          j ← k
       Τέλος_αν
    Τέλος_επανάληψης
    Αντιμετάθεσε THL[i], THL[j]
Τέλος_επανάληψης
Αποτελέσματα // THL //
Τέλος Αριθμοί_Τηλεφώνων
```

Ο παραπάνω αλγόριθμος αποτελεί μία απλή πρόταση για την ταξινόμηση στοιχείων και είναι γνωστός ως αλγόριθμος **ταξινόμησης με επιλογή** (selection sort). Η ονομασία του οφείλεται στη λογική που χρησιμοποιεί για την ταξινόμηση, η οποία βασίζεται στην επιλογή του μικρότερου στοιχείου από αυτά που δεν έχουν ταξινομηθεί σε κάθε βήμα.

Παράδειγμα 2. Ανεπιτυχής δυαδική αναζήτηση

Σε ένα παραθεριστικό κέντρο υπάρχουν πολλά καταστήματα και εστιατόρια. Ένας επιχειρηματίας θέλει να ανοίξει ένα κατάστημα και θέλει να του δώσει το όνομα «Άνοιξη». Πρέπει πρώτα να ερευνήσει εάν αυτό το όνομα έχει ήδη δοθεί σε κάποιο άλλο κατάστημα. Έστω ότι όλα τα ονόματα καταστημάτων του παραθεριστικού κέντρου έχουν καταγραφεί σε ένα πίνακα 50 θέσεων. Να προτείνετε τον κατάλληλο αλγόριθμο που θα δώσει την απάντηση στον επιχειρηματία για το εάν μπορεί να ανοίξει το κατάστημα με αυτό το όνομα ή όχι

Για να βρεις τον κατάλληλο αλγόριθμο πρέπει να χρησιμοποιήσεις την ιδέα της δυαδικής αναζήτησης που έχει παρουσιασθεί με παράδειγμα στο βιβλίο σου (Κεφάλαιο 4). Η διαφοροποίηση σε εκείνο τον αλγόριθμο έχει να κάνει με το ότι, εδώ χρειάζεται να υπάρξει κάποια μέριμνα για το εάν το στοιχείο που αναζητούμε, βρέθηκε στον πίνακα ή όχι. Στις περιπτώσει αυτές συνηθίζεται να υπάρχει κάποια μεταβλητή που αναλαμβάνει αυτόν το ρόλο και ενημερώνεται με την κατάλληλη τιμή. Συχνά οι μεταβλητές αυτές εκφράζουν δύο καταστάσεις (π.χ. εδώ έχουμε βρέθηκε / δεν βρέθηκε) και για αυτό χαρακτηρίζονται ως δυαδικές μεταβλητές «σημαίες» (boolean flags). Η μεταβλητή found στον παρακάτω αλγόριθμο έχει αυτόν τον ρόλο. Εάν η τιμή της found είναι 0, δεν έχει βρεθεί το όνομα που ψάχνεις, αν η τιμή της είναι 1, έχει βρεθεί

και επομένως το κατάστημα δεν μπορεί να πάρει το όνομα που έχεις δώσει. Επιπλέον, στον αλγόριθμο που ακολουθεί χρησιμοποιείται ο πίνακας ΚΑΤ, στοιχεία του οποίου είναι τα ονόματα των 50 καταστημάτων που έχουν διαβασθεί όπως έχει περιγραφεί σε προηγούμενα κεφάλαιο και ΟΝΟΜΑ "Ανοιξη" είναι το όνομα που αναζητούμε για το κατάστημα.

```
Αλγόριθμος Δυαδική_αναζήτηση
Δεδομένα // ONOMA, KAT //
low ← 0
high ← 50
found ← 0
όσο low ≤ high επανάλαβε
    mid ← (low + high)/2
    Αν KAT[mid] < ONOMA τότε
        low ← mid+1
    αλλιώς_αν KAT[mid] > ONOMA τότε
        high ← mid-1
    αλλιώς
        found ← 1
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // found //
Τέλος Δυαδική_αναζήτηση
```

Παράδειγμα 3. Εύρεση δύο μικρότερων αριθμών.

Σε ένα Τμήμα μίας επιχείρησης χρειάζεται να βρεθούν οι δύο χαμηλότεροι μισθοί με δεδομένο ότι το Τμήμα απασχολεί 50 υπαλλήλους και οι μισθοί τους αποθηκεύονται σε κάποιον πίνακα. Να γραφεί ένας αλγόριθμος που θα υπολογίζει τους δύο μικρότερους μισθούς με δεδομένο τον πίνακα των μισθών των υπαλλήλων.

Το πρόβλημα της ανεύρεσης των δύο μικρότερων στοιχείων ενός πίνακα επιδέχεται διάφορες τεχνικές και τρόπους σχεδίασης. Ο αλγόριθμος που παρουσιάζεται στη συνέχεια είναι αρκετά απλός και δεν έχει καλή αποδοτικότητα.

```
Αλγόριθμος Δύο_Μικρότεροι
Δεδομένα // M //
low1 ← M[1]
pos ← 1
Για i από 2 μέχρι 50
    Αν M[i] < low1 τότε
        low1 ← M[i]
        pos ← i
    Τέλος_αν
Τέλος_επανάληψης
```

```
Αν pos <> 1 τότε
    low2 ← M[1]
αλλιώς
    low2 ← M[2]
Τέλος_αν
Για i από 2 μέχρι 50
    Αν (i <> pos και M[i] < low2) τότε
        low2 ← M[i]
    Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // low1 , low2 //
Τέλος Δύο_Μικρότεροι
```

4.3. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Να παρακολουθήσετε την πορεία των αλγορίθμων που προτείνεται στο Παράδειγμα 3 για παραδείγματα πινάκων κάποιων θέσεων. Να συζητήσετε την πιθανότητα να προταθεί κάποια βελτίωση και διαφορετικότητα στην τεχνική σχεδίασης του προτεινόμενου αλγορίθμου.

ΔΤ2. Εστω ότι ο κατάλογος των μουσείων της πόλης σας υπάρχει αποθηκευμένος σε έναν πίνακα ο οποίος περιέχει το όνομα κάθε μουσείου. Εστω ότι κάποιος επισκέπτης θέλει να μάθει εάν κάποιο συγκεκριμένο μουσείο (π.χ. Λαογραφικό) υπάρχει στην πόλη σας. Να προτείνετε δύο τρόπους για την αναζήτηση ενός συγκεκριμένου μουσείου από αυτόν τον πίνακα και να συζητήσετε στην τάξη τη διαφορά και τον τρόπο λειτουργίας κάθε αλγορίθμου.

ΔΤ3. Να συζητηθεί και να αναλυθεί ο αλγόριθμος που υλοποιεί τη δημιουργία ενός μαγικού τετραγώνου όπως αυτό περιγράφεται στη συνέχεια:

‘Ενα μαγικό τετράγωνο είναι ένας n x n πίνακας από ακέραιους από το 1 μέχρι το n^2 που έχει κατασκευασθεί έτσι ώστε το άθροισμα κάθε γραμμής, κάθε στήλης και κάθε διαγωνίου να είναι το ίδιο, όπως φαίνεται και στο παρακάτω σχήμα :

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

Στο παραπάνω σχήμα παρουσιάζεται ένα μαγικό τετράγωνο με n=5, στο οποίο κάθε γραμμή ή στήλη ή διαγώνιος έχει άθροισμα 65. 'Εχει προταθεί κάποιος συγκεκριμένος κανόνας για τη δημιουργία ενός μαγικού τετραγώνου για n περιττό αριθμό. Ο κανόνας αυτός συνοψίζεται στα εξής :

«Ξεκινούμε τοποθετώντας τον αριθμό 1 στη μεσαία θέση της πρώτης γραμμής. Στη συνέχεια προχωρούμε αναθέτοντας τους αριθμούς 2, 3, 4, ... κλπ μετακινούμενοι συνεχώς προς τα επάνω και αριστερά μέχρι να γεμίσει το μαγικό τετράγωνο. 'Οταν κάποια μετακίνηση προς τα επάνω ή προς τα αριστερά μας οδηγεί εκτός των ορίων του τετραγώνου πηγαίνουμε στο αντι-διαμετρικό άκρο της γραμμής ή της στήλης στην οποία βρεθήκαμε. Επίσης αν η μετακίνηση μας οδηγεί σε κατειλημμένη θέση, τότε επιλέγεται η θέση κάτω από αυτήν όπου έγινε η τελευταία ανάθεση».

Ο παρακάτω αλγόριθμος υλοποιεί αυτόν τον κανόνα χρησιμοποιώντας τη δομή ενός δισδιάστατου πίνακα (square) για να κρατηθούν οι τιμές των στοιχείων του μαγικού τετραγώνου

```
Αλγόριθμος Μαγικό_τετράγωνο
Δεδομένα // n //
Για i από 1 μέχρι n
    Για j από 1 μέχρι n
        square[i,j] ← 0
    Τέλος_επανάληψης
Τέλος_επανάληψης
i←1
j←(n+1)/2
square[i,j]←1
Για key από 2 μέχρι n*n
    Αν i>1 τότε
        k←i-1
    αλλιώς
        k←n-1
    Τέλος_αν
    Αν j>1 τότε
        l←j-1
    αλλιώς
        l←n-1
    Τέλος_αν
    Αν square[k,l]>0 τότε
        i←i+1
        Αν i=n+1 τότε i←1
    αλλιώς
        i←k
        j←l
    Τέλος_αν
    square[i,j]←key
Τέλος_επανάληψης
Αποτελέσματα // square //
Τέλος Μαγικό_τετράγωνο
```

ΔΤ4. Ενα πρακτορείο ταξιδιών διοργανώνει μία εκδρομή για το γύρο του κόσμου σε 80 ημέρες. Για να κάνει το σχεδιασμό του ταξιδιού χρειάζεται να επιλέξει κάποιες πόλεις και συγκεκριμένη διαδρομή. Να συζητήσετε στην τάξη και να καταγράψετε τα βασικά βήματα ενός αλγορίθμου που θα σχεδιάζει τη διαδρομή που θα πρέπει να ακολουθήσει ο ταξιδιώτης ξεκινώντας από μία πόλη και καταλήγοντας πάλι σε αυτήν αφού περάσει μία φορά από τις πόλεις που έχουν επιλεγεί. Είναι χρήσιμο στο σχεδιασμό της διαδρομής να παίρνει κανείς την απόφαση για τη μικρότερη δυνατή διαδρομή.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Εστω ότι έχεις να παίξεις ένα παιχνίδι όπου προσπαθείς να μαντέψεις ένα αριθμό από το 1 μέχρι το 10 που έχει γράψει κάποιος συμμαθητής σου σε ένα χαρτί. Σε κάθε προσπάθεια, ο συμμαθητής σου απαντά δηλώνοντας αν ο αριθμός του είναι μικρότερος, μεγαλύτερος ή ίσος με το δικό σου. Να γράψεις έναν αλγόριθμο που θα σε οδηγήσει με γρήγορο τρόπο στο να βρεις τον αριθμό που έγραψε ο συμμαθητής σου. Πόσα βήματα θα χρειαστείς μέχρι να βρεις τον αριθμό;

ΔΣ2. Στο προηγούμενο κεφάλαιο είχες ασχοληθεί με την ταξινόμηση των δίσκων των CD σου σε χρονολογική σειρά. Να επεκτείνεις τον αλγόριθμο έτσι ώστε να υπάρχει δυνατότητα να βρίσκεις εάν ένα CD με συγκεκριμένο τίτλο υπάρχει στη συλλογή σου ή όχι, δίνοντας τον τίτλο του (δυαδική αναζήτηση).

ΔΣ3. Εστω ότι έχεις τον παρακάτω αλγόριθμο :

```
Αλγόριθμος Expo
Δεδομένα // x, n //
m ← n
pow ← 1
z ← x
Οσο m > 0 επανάλαβε
    Οσο ( m MOD 2 ) = 0 επανάλαβε
        m ← m/2
        z ← z * z
    Τέλος_επανάληψης
    m ← m-1
    pow ← pow*z
Τέλος_επανάληψης
Αποτελέσματα // pow //
Τέλος Expo
```

Η «πράξη» MOD έχει ως αποτέλεσμα το υπόλοιπο της ακέραιης διαίρεσης δύο αριθμών (π.χ. 14 MOD 3 = 2, 4 MOD 3 = 1), ενώ το m/2 αναφέρεται στον αμέσως μικρότερο ακέραιο από τον αριθμό που προκύπτει ως αποτέλεσμα της διαίρεσης (π.χ. 13/2 = 6, 25/3 = 8).

Να παρακολουθήσεις την πορεία του αλγόριθμου για τα εξής ζεύγη τιμών:

x=4, n=2 x=2, n=4 x=3, n=3 x=5, n=2

Με βάση την παρακολούθηση που έκανες ποιό είναι το συμπέρασμά σου για το αποτέλεσμα του παραπάνω αλγορίθμου ;

ΔΣ4. Στη βιβλιοθήκη ενός σχολείου υπάρχουν πολλά βιβλία σχετικά με τη γεωγραφία και τα ταξίδια. 'Εστω ότι κάθε βιβλίο έχει ένα μοναδικό κωδικό και καταχωρείται σε ηλεκτρονικό υπολογιστή ο τίτλος και ο συγγραφέας κάθε βιβλίου. Να γραφεί αλγόριθμος που θα διαβάζει το όνομα ενός συγγραφέα και θα βρίσκει τον κωδικό (ή τους κωδικούς) και τον τίτλο (ή τους τίτλους) των βιβλίων αυτού του συγγραφέα που υπάρχουν στη βιβλιοθήκη.

ΔΣ5. 'Ενας διαγωνισμός τραγουδιού στην Ευρώπη διεξάγεται ως εξής. Γίνεται μία πρώτη ακρόαση των τραγουδιών κάθε χώρας από την Κριτική Επιτροπή η οποία δίνει κάποιους βαθμούς σε κάθε τραγούδι (από 1- 100). 'Εστω ότι είναι γνωστοί οι βαθμοί που δόθηκαν στο τραγούδι κάθε χώρας. Να γραφεί ένας αλγόριθμος που θα επιλέγει για τη συνέχεια στη δεύτερη φάση του διαγωνισμού τις χώρες με τη μεγαλύτερη βαθμολογία κάθε φορά ώστε το άθροισμα της βαθμολογίας όλων των τραγουδιών που θα προχωρήσουν στη δεύτερη φάση να είναι μικρότερο από 1000 βαθμούς.

ΔΣ6. Να παρακολουθήσεις το πρόβλημα για τον «Γύρο του κόσμου» που δόθηκε στις δραστηριότητες για την τάξη (ΔΤ4) και να κάνεις ένα σχήμα για 10 πόλεις και των μεταξύ τους αποστάσεων με δεδομένο ότι υπάρχει αεροπορική σύνδεση για κάποιες από αυτές. Στη συνέχεια να δώσεις σχηματικά τη λύση για τη μικρότερη δυνατή διαδρομή.

4.4. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες προτάσεων. Σε κάθε μία από αυτές, να κάνετε τις απαραίτητες διορθώσεις ώστε να ισχύουν οι προτάσεις

1. Οι μέθοδοι λύσης ενός προβλήματος που προκύπτουν από την υλοποίηση του σε συγκεκριμένο υπολογιστικό σύστημα, οδηγούν στη σχεδίαση ενός αλγορίθμου που συνιστά την ακολουθία βημάτων που πρέπει να ακολουθηθούν για να επιλυθεί το πρόβλημα.
2. Κατά την επίλυση ενός προβλήματος, δεν γίνεται σύγκριση των χαρακτηριστικών και των ιδιοτήτων διαφορετικών τεχνικών σχεδίασης ενός αλγορίθμου αλλά επιλέγεται η πλέον εκτενής τεχνική.
3. Η μέθοδος του Δυναμικού Προγραμματισμού για τη σχεδίαση αλγορίθμων χρησιμοποιείται κυρίως για την επίλυση προβλημάτων υποδιαιρέσεων σε μικρότερα μεγέθη προβλημάτων και κυρίως κατά την ταξινόμηση.

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

4 Η _____ μέθοδος προχωρά με την παραδοχή ότι σε κάθε βήμα γίνεται επιλογή της τρέχουσας βέλτιστης επιλογής.

5 Η τεχνικής της _____ εντάσσεται στην κατηγορία αντιμετώπισης προβλημάτων που είτε περιλαμβάνουν την αναζήτηση ενός συνόλου λύσεων, είτε αναζητούν τη βέλτιστη λύση υπό κάποιες προϋποθέσεις.

6 Η_____ διευκολύνει την αποδοτική ανεύρεση στοιχείου από πίνακα, υποδιαιρώντας τον πίνακα σε δύο μέρη σε κάθε βήμα και συνεχίζοντας με τον κατάλληλο από τους δύο υπο-πίνακες.

7 Δύο γνωστές τεχνικές για την ταξινόμηση είναι η _____ και η _____.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

8 Γενικότερα, κάθε τεχνική σχεδίασης χρειάζεται να υποστηρίζει τα εξής :

- να αντιμετωπίζει με τα δικό της τρόπο τα δεδομένα.
- να έχει τη δική της ακολουθία εντολών.
- να διαθέτει τη δική της αποδοτικότητα.

9. Η γραμμική αναζήτηση διευκολύνει την αποδοτική ανεύρεση στοιχείου από πίνακα και δεν υπάρχει καλύτερος τρόπος αναζήτησης στοιχείου από πίνακα.

10. Η τεχνική του Δυναμικού Προγραμματισμού είναι πιο αποδοτική από την τεχνική Διαίρει και Βασίλευε.

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

11. Κατά την ανάλυση ενός προβλήματος θα πρέπει να δοθεί απάντηση σε κάθε μία από τις επόμενες ερωτήσεις:

Α) Ποιά είναι τα δεδομένα και το μέγεθος του προβλήματος

Β) Ποιές είναι τα περιφερειακά του συστήματος στο οποίο θα επιλυθεί ο αλγόριθμος

Γ) Ποιά είναι η χρησιμότητα του αλγορίθμου

Δ) Πώς θα καταγραφεί η λύση σε ένα πρόβλημα; (π.χ. σε ψευδογλώσσα)

Ε) Ποιός είναι ο τρόπος υλοποίησης στο συγκεκριμένο υπολογιστικό σύστημα (π.χ. επιλογή γλώσσας προγραμματισμού).

12. Κάθε τεχνική σχεδίασης αλγορίθμου χρειάζεται να υποστηρίζει τα εξής :

Α) Να υλοποιείται σε συγκεκριμένη γλώσσα προγραμματισμού.

Β) Να αντιμετωπίζει με το δικό της τρόπο τα δεδομένα.

Γ) Να έχει τη δική της ακολουθία εντολών.

Δ) Να δέχεται συγκεκριμένη είσοδο από πληκτρολόγιο.

Ε) Να διαθέτει τη δική της αποδοτικότητα.

13. Η περιγραφή της μεθόδου Διαίρει και Βασίλευε για τη σχεδίαση αλγορίθμων μπορεί να αποδοθεί με τα επόμενα βήματα:

Α) Δίνεται το όνομα του χρήστη του αλγορίθμου.

Β) Δίνεται για επίλυση ένα στιγμιότυπο ενός προβλήματος.

Γ) Το συνολικό μέγεθος του προβλήματος διαιρείται δια 2.

Δ) Υποδιαίρεση του στιγμιότυπου του προβλήματος σε υπο-στιγμιότυπα του ίδιου προβλήματος.

Ε) Δίνεται ανεξάρτητη λύση σε κάθε ένα υπο-στιγμιότυπο.

Ζ) Συνδυάζονται όλες οι μερικές λύσεις που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η συνολική λύση του προβλήματος.

Η) Εκτυπώνεται κάθε φορά πολλά είδη λύσεων.

ΚΕΦΑΛΑΙΟ 5ο

ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

5.1. Πζοσδοκώμενα αποτελέσματα

Όταν θα έχεις ολοκληρώσει τη μελέτη αυτού του κεφαλαίου θα έχεις κατανοήσει τις τεχνικές ανάλυσης των αλγορίθμων. Θα μπορείς να μετράς την επίδοση των αλγορίθμων με βάση την αποδοτικότητά τους. Θα είσαι σε θέση χρησιμοποιώντας τις κατάλληλες μεθόδους να ελέγχεις τη ορθότητά τους. Ή ακόμα λαμβάνοντας υπόψη σου κάποια κριτήρια, να επιλέγεις τον προτιμότερο αλγόριθμο για το πρόβλημα που καλείσαι να αντιμετωπίσεις. Τέλος, θα έχεις κατανοήσει την έννοια της πολυπλοκότητας των αλγορίθμων.

5.2. Επιπλέον παζαδείγματα

Παράδειγμα 1

Έστω ότι έχουμε το παρακάτω πρόγραμμα υλοποίησης ενός αλγορίθμου :

```
Αλγόριθμος Ελεγχος_εκτέλεσης
a ← 1
b ← 2
Για  i  από  1 μέχρι 100
    a ← i
    b ← a * i
Τέλος_επανάληψης
Εκτύπωσε a
```

```
Εκτύπωσε b
Τέλος Ελεγχος_εκτέλεσης
```

Να υπολογισθεί η επίδοσή του με βάση τον αριθμό των πράξεων που θα εκτελεσθούν.

Εντολή αλγορίθμου	Αριθμός πράξεων
ανάθεση τιμών στα a και b	2
Βρόχος επανάληψης αρχική τιμή i έλεγχος i αύξηση i ανάθεση τιμών στο a ανάθεση τιμών στο b (2X100)	 1 101 100 100 200
Εκτύπωση a,b	2
ΣΥΝΟΛΟ	***506***

Με δεδομένο ότι ο βρόχος του προγράμματος θα εκτελεσθεί 100 φορές προκύπτει η παραπάνω ανάλυση, η οποία αποτελεί εκτίμηση και του χρόνου εκτέλεσης του προγράμματος αλγορίθμου. Είναι χρήσιμο εδώ να καταγραφεί το μέγεθος του προβλήματος και να εκφρασθεί το σύνολο των κριτηρίων επίδοσης σε σχέση με αυτό.

Παράδειγμα 2

Έστω ότι έχουμε τον παρακάτω αλγόριθμο ανάγνωσης και άμεσης εκτύπωσης των στοιχείων ενός δισδιάστατου πίνακα Α :

```
Αλγόριθμος Ανάγνωση_Εκτύπωση_Πίνακα
Δεδομένα  // n //
Για i από 1 μέχρι  n
   Για j από 1 μέχρι  n
      Διάβασε  A[i, j]
      Εκτύπωσε  A[i, j]
   Τέλος_επανάληψης
Τέλος_επανάληψης
Τέλος Ανάγνωση_Εκτύπωση_Πίνακα
```

Να υπολογισθεί ο χρόνος εκτέλεσης του αλγορίθμου αυτού και να σχολιασθεί η βαρύτητα των πράξεων επανάληψης σε σχέση με την απόφαση για την πολυπλοκότητα των αλγορίθμων.

Από ότι παρατηρούμε υπάρχουν δύο βρόχοι επανάληψης (ένας βρόχος για κάθε διάσταση του πίνακα). Για κάθε στιγμή επανάληψης μέσα στο εσωτερικό των δύο βρόχων γίνονται δύο απλές πράξεις (ανάγνωση και εκτύπωση) μοναδιαίου κόστους η καθεμία. Επομένως η πολυπλοκότητα του παραπάνω αλγορίθμου θα εκφράζεται με n * n * 2, δηλαδή ο αλγόριθμος είναι τετραγωνικός. Είναι φανερό ότι οι βρόχοι επανάληψης είναι εκείνοι που καθορίζουν την επιβάρυνση στο κόστος εκτέλεσης του αλγορίθμου.

Παράδειγμα 3. Ανάλυση αλγορίθμου ταξινόμησης.

Έστω ότι έχουμε την παρακάτω απλή μορφή για τον αλγόριθμο ταξινόμησης με ευθεία ανταλλαγή (bubblesort) :

```
Αλγόριθμος Ευθεία_Ανταλλαγή
Δεδομένα  // A //
Για i από 1 μέχρι n-1
    Για j από 1 μέχρι n-1
        Αν A[j+1] < A[j]  τότε
            Αντιμετάθεσε A[j+1], A[j]
        Τέλος_αν
    Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // A //
Τέλος Ευθεία_Ανταλλαγή
```

Έστω ότι έχουμε τον πίνακα Α με τα παρακάτω στοιχεία :

1	2	3	4
7	4	3	8

Να παρακολουθήσετε την πορεία του αλγορίθμου με καταγραφή της επίδοσής του που θα εκφράζεται από τον αριθμό των πράξεων που πρέπει να εκτελεσθούν.

Επίδοση Αλγορίθμου

Για να εκφρασθεί η απόδοση του αλγορίθμου χρειάζεται να μετρηθεί ο αριθμός των συγκρίσεων και ο αριθμός των ανταλλαγών που θα πραγματοποιηθούν για την ταξινόμηση. Στη συνέχεια παρουσιάζονται αναλυτικά αυτοί οι υπολογισμοί :

Αριθμός Συγκρίσεων

i ← 1 (1ο Πέρασμα)

1	2	3	4	
7	4	3	8	Σύγκριση 1ου και 2ου
4	7	3	8	Σύγκριση 2ου και 3ου
4	3	7	8	Σύγκριση 3ου και 4ου

i ← 2 (2ο Πέρασμα)

1	2	3	4	
4	3	7	8	Σύγκριση 1ου και 2ου
3	4	7	8	Σύγκριση 2ου και 3ου
3	4	7	8	Σύγκριση 3ου και 4ου

i ← 3 (3ο Πέρασμα)

1	2	3	4	
3	4	7	8	Σύγκριση 1ου και 2ου
3	4	7	8	Σύγκριση 2ου και 3ου
3	4	7	8	Σύγκριση 3ου και 4ου

Όπως φαίνεται από την παρακολούθηση της πορείας του αλγορίθμου η εκτέλεση του περιλαμβάνει 3 συνεχόμενα «περάσματα» πάνω από τα στοιχεία του πίνακα και σε κάθε πέρασμα συγκρίνονται ζευγάρια στοιχείων. Από ότι φαίνεται και παραπάνω γίνονται συνολικά 9 πράξεις σύγκρισης, αφού σε κάθε πέρασμα γίνονται 3 συγκρίσεις και όπως αναφέρθηκε υπάρχουν 3 περάσματα. Ο πίνακας του παραδείγματος μας έχει n=4 θέσεις. Αν γενικεύσουμε την καταγραφή των πράξεων της σύγκρισης παρατηρούμε ότι θα χρειαστούν **(n-1)*(n-1) συγκρίσεις** για έναν πίνακα n θέσεων.

Αριθμός Ανταλλαγών

Έστω τώρα θέλουμε να καταγράψουμε τον αριθμό των ανταλλαγών που γίνονται για τις ανάγκες της ταξινόμησης. Για να υπάρξει κριτήριο για την τυποποίηση της επίδοσης του αλγορίθμου θα πρέπει να μετρηθεί ο αριθμός των ανταλλαγών που γίνονται στη χειρότερη περίπτωση. Επομένως η χειρότερη περίπτωση για ένα πίνακα 4 θέσεων (που είναι και το μέγεθος του προβλήματός μας) είναι τα στοιχεία του να βρίσκονται ταξινομημένα κατά φθίνουσα τάξη έτσι ώστε όλα να πρέπει να αλλάξουν θέση. Έστω λοιπόν η πιο απλή περίπτωση αυτής της κατάστασης :

i ← 1 (1ο Πέρασμα)

4	3	2	1	1 αντα΄λ΄λαγή
3	4	2	1	1 αντα΄λ΄λαγή
3	2	4	1	1 αντα΄λ΄λαγή

i ← 2 (2ο Πέρασμα)

3	2	1	4	1 αντα΄λ΄λαγή
2	3	1	4	1 αντα΄λ΄λαγή
2	1	3	4	0 αντα΄λ΄λαγή

i ← 3 (3ο Πέρασμα)

2	1	3	4	1 αντα΄λ΄λαγή
1	2	3	4	0 αντα΄λ΄λαγή
1	2	3	4	0 αντα΄λ΄λαγή

Επομένως θα γίνουν συνολικά 3+2+1=6 ανταλλαγές για την ταξινόμηση 4 στοιχείων που είναι τοποθετημένα σε αντίθετη της κανονικής τάξης (χειρότερη περίπτωση). Αν γενικεύσουμε το παράδειγμα για πίνακα n θέσεων θα χρειασθούν στη χειρότερη περίπτωση **(n-1)+(n-2)+...+1 ανταλλαγές στοιχείων**.

Η πολυπλοκότητα του παραπάνω αλγορίθμου υπολογίζεται από το άθροισμα του «κόστους» των συγκρίσεων και των ανταλλαγών που είναι

$$(n-1)+(n-2)+\ldots+1+(n-1)\cdot(n-1)=\frac{3}{2}n^2-\frac{5}{2}n+1$$

Είναι φανερό ότι η τάξη της πολυπλοκότητας του αλγορίθμου υπολογίζεται από το μεγαλύτερο όρο του πολυωνύμου, ο οποίος είναι το n^2, και έτσι προκύπτει ότι ο παραπάνω αλγόριθμος έχει τετραγωνική πολυπλοκότητα

Παράδειγμα 4. Εύρεση του i-οστού αριθμού Fibonacci

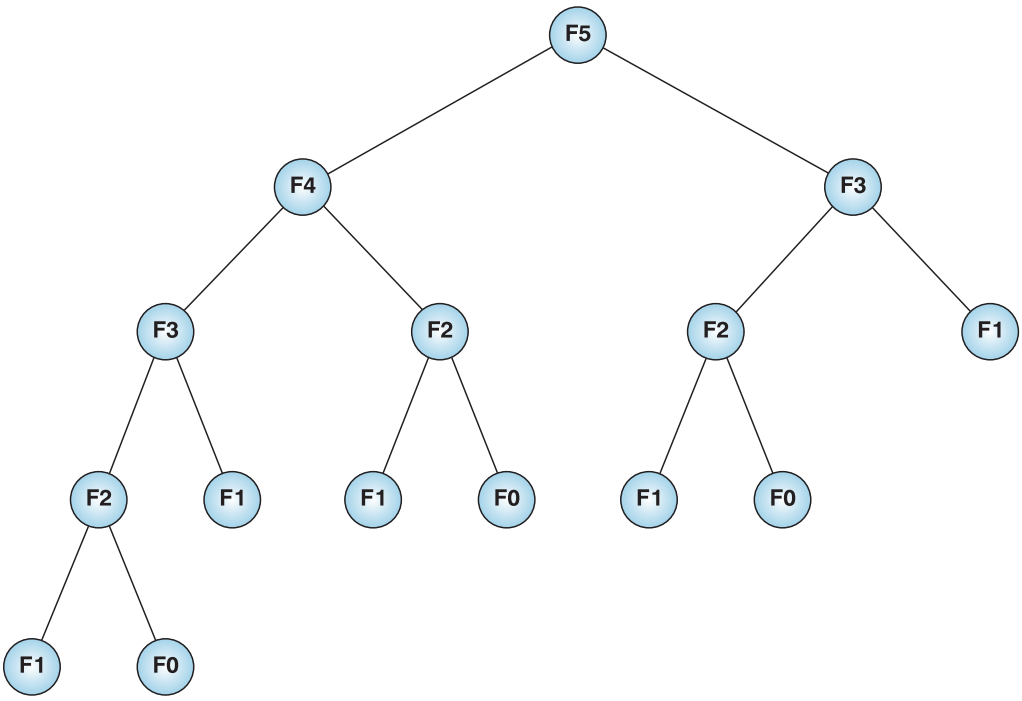
Στο κεφάλαιο του Βιβλίου σου περί αναδρομής συζητήσαμε τα πλεονεκτήματα και μειονεκτήματα των επαναληπτικών και των αναδρομικών συναρτήσεων. Μάλιστα καταλήξαμε στο συμπέρασμα ότι μία επαναληπτική διαδικασία πρέπει να προτιμάται έναντι μίας ισοδύναμης αναδρομικής όταν το βασικό μας κριτήριο είναι ο χρόνος εκτέλεσης/απόκρισης. Στο σημείο αυτό επανερχόμαστε ώστε να εξηγηθεί ότι συχνά μία αναδρομική πρέπει να αποφεύγεται και για ένα άλλο σημαντικό λόγο.

Οι επόμενοι δύο αλγόριθμοι υπολογίζουν τον i-οστό αριθμό Fibonacci. Ο πρώτος είναι επαναληπτικός ενώ ο δεύτερος είναι αναδρομικός.

```
Αλγόριθμος Fibonacci
Δεδομένα // n //
Αν n ≤ 1 τότε
    Fib ← n
αλλιώς
    j ← 0
    k ← 1
    Για i από 1 μέχρι n
        j ← j+k
        k ← j-k
    Τέλος_επανάληψης
    Fib ← j
Τέλος_Αν
Αποτελέσματα // Fib //
Τέλος Fibonacci
```

```
Αλγόριθμος Fibonacci
Δεδομένα // n //
Αν n ≤ 1 τότε
    Fib ← n
αλλιώς
    Fib ← Fib(n-1) + Fib(n-2)
Τέλος_Αν
Αποτελέσματα // Fib //
Τέλος Fibonacci
```

Η επαναληπτική διαδικασία, λοιπόν, βασίζεται σε ένα και μόνο βρόχο, άρα η πολυπλοκότητα της μεθόδου είναι τάξης O(n). Η δεύτερη διαδικασία υπολογίζει τις ίδιες τιμές πολλές φορές. Για παράδειγμα, κατά τον υπολογισμό του F_5 γίνονται οι κλήσεις Fib(4) και Fib(3). Όμως κατά τον υπολογισμό του Fib(4) καλείται για δεύτερη φορά η Fib(3). Με το σκεπτικό αυτό η Fib(2) θα κληθεί τρεις φορές, η Fib(1) θα κληθεί πέντε φορές και η Fib(0) θα κληθεί τρεις φορές. Έτσι συμπερασματικά προκύπτει ότι η πολυπλοκότητα της αναδρομικής μεθόδου είναι τάξης $O(F_n)$.



***Σχ. 5.1**. Κλήσεις για τον υπολογισμό του F_5.*

5.3. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Να συζητηθεί η επίδοση των παρακάτω κομματιών αλγορίθμων και να καταγραφεί για κάθε περίπτωση και η αντίστοιχη πολυπλοκότητα. Για να βρείτε την πολυπλοκότητα θα πρέπει να μετρήσετε τον αριθμό των πράξεων στη χειρότερη περίπτωση :

```
1.
Για i από 1 μέχρι  n -1 με βήμα 2
    a ← 2*i
Τέλος_επανάληψης

2.
Για i από 1 μέχρι  n
    Για j από 1 μέχρι  n
       a ← 2*i +j
    Τέλος_επανάληψης
Τέλος_επανάληψης


3.
Για  i από 1 μέχρι  n  με βήμα 2
    Για j από 1 μέχρι  n
       a ← 2*i +j
    Τέλος_επανάληψης
Τέλος_επανάληψης
```

ΔΤ2. Εστω ότι ένας πίνακας κρατά τα ποσά που έχουν δώσει οι μαθητές της τάξης σας για την ενίσχυση του παιδικού χωριού SOS της περιοχής σας. Να δώσετε έναν αλγόριθμο για τον υπολογισμό του συνολικού ποσού που θα διατεθεί και να σχολιάσετε την πολυπλοκότητά του.

ΔΤ3. Να σχολιασθεί και να παρακολουθήσετε βήμα-βήμα τον ακόλουθο επαναληπτικό αλγόριθμο υπολογισμού των αριθμών Fibonacci. Ποιά είναι η πολυπλοκότητα του ακόλουθου αλγόριθμου ;

```
Αλγόριθμος Fibonacci
i ← 1
j ← 0
k ← 0
l ← 1
Επανάλαβε όσο n > 0
    Αν n MOD 2 = 1 τότε m ← j*l
    j ← i*l+j*k+m
    i ← i*k+m
    m ← Ρίζα(l)  'Σχόλιο: Ρίζα είναι η συνάρτηση τετραγωνικής ρίζας'
    h ← 2*k*l+m
```

```
    k ← Ρίζα(k)+m
    n ← n DIV 2
Τέλος_επανάληψης
Fib ← j
Αποτελέσματα Fib
Τέλος Fibonacci
```

ΔΤ4. Σε μία αποθήκη ταινιών κινηματογράφου υπάρχει αρχειοθέτηση των ταινιών με βάση τη χρονιά που παρουσιάσθηκε κάθε ταινία. Ενας σύλλογος ενδιαφέρεται να κάνει ένα αφιέρωμα στις ταινίες της δεκαετίας του 1960 που διαθέτει η αποθήκη. Να προτείνετε ένα αλγόριθμο αναζήτησης των ταινιών αυτών και να σχολιάσετε την επίδοση και την πολυπλοκότητά του.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Να βρείς την πολυπλοκότητα των παρακάτω κομματιών αλγορίθμων :

1.
```
Επανάλαβε όσο i < 100
    a ← 2*i
Τέλος_επανάληψης
```

2.
```
Για i από 1 μέχρι n -1
    a ← 2*i
Τέλος_επανάληψης
```

3.
```
Για i από 1 μέχρι n -1 με βήμα 3
    a ← 2*i
Τέλος_επανάληψης
```

ΔΣ2. Έστω ότι έχουμε τον παρακάτω αλγόριθμο :

```
Αλγόριθμος  Ελεγχος_επίδοσης
x ← 10
c ← 20
Για i από 100 μέχρι 10 με_βήμα -10
    x ← i
    y ← 2*x + i
Τέλος_επανάληψης
Εκτύπωσε x
Εκτύπωσε y
Τέλος Ελεγχος_ επίδοσης
```

Να υπολογισθεί η επίδοσή του με βάση τον αριθμό των πράξεων που θα εκτελεσθούν.

ΔΣ3. Να παρακολουθήσεις την πορεία του αλγορίθμου της Δυαδικής αναζήτησης που δόθηκε στο προηγούμενο κεφάλαιο (Τετράδιο Μαθητή) για έναν πίνακα 10 θέσεων. Να καταγράφεις τον αριθμό των πράξεων που γίνονται σε κάθε βήμα και να καταλήξεις σε ένα συμπέρασμα για την πολυπλοκότητά του.

ΔΣ4. Έστω ότι έχεις τον παρακάτω αλγόριθμο :

```
Αλγόριθμος Ευθεία_Ανταλλαγή2
Δεδομένα  //  A //
Για i από 1 μέχρι  n-1
   Για j από 1 μέχρι  n-1
      Αν A[j+1] < A[j]  τότε
         Αντιμετάθεσε A[j+1], A[j]
      Τέλος_αν
   Τέλος_επανάληψης
   Εκτύπωσε A[j]
Τέλος_επανάληψης
Αποτελέσματα // A //
Τέλος Ευθεία_Ανταλλαγή2
```

Να σχολιάσεις την πολυπλοκότητα του αλγορίθμου. Χρησιμοποίησε το Παράδειγμα 2 και να ελέγξεις αν ο παραπάνω αλγόριθμος είναι ίδιας ή διαφορετικής πολυπλοκότητας από τον αλγόριθμο του Παραδείγματος 2.

ΔΣ5. Μπορείς να προτείνεις κάποιο διαφορετικό αλγόριθμο για την ανεύρεση των ταινιών κινηματογράφου από την αποθήκη που περιγράφηκε παραπάνω στις δραστηριότητες για την τάξη (ΔΤ5) ; Πώς θα σχολίαζες την πολυπλοκότητα των δύο διαφορετικών αλγορίθμων ;

5.4. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες προτάσεων. Σε κάθε μία από αυτές, να κάνετε τις απαραίτητες διορθώσεις ώστε να ισχύουν οι προτάσεις

1. Η χειρότερη περίπτωση ενός αλγορίθμου αφορά στο ελάχιστο κόστος εκτέλεσης του αλγορίθμου, κόστος που μετράται σε υπολογιστικούς πόρους.
2. Τα δεδομένα συνιστούν το μέγεθος της πολυπλοκότητας ενός αλγορίθμου.
3. Ο απλούστερος τρόπος μέτρησης της επίδοσης ενός αλγορίθμου είναι ο *εμπειρικός* ή αλλιώς ο λεγόμενος *εκ των προτέρων που* υλοποιείται και εφαρμόζεται σε ένα σύνολο δεδομένων ώστε να υπολογισθεί ο απαιτούμενος *χρόνος επεξεργασίας* και η πολυπλοκότητα.

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

4 Τα δεδομένα συνιστούν το μέγεθος της _____ ενός αλγορίθμου.

5 Ο συμβολισμός $O(n^2)$ εκφράζει την _____ πολυπλοκότητα.

6 Ο συμβολισμός O(logn) εκφράζει τη _____ πολυπλοκότητα

7 Ως _____ ορίζονται οι αλγόριθμοι που δε δίνουν την καλύτερη λύση, αλλά προτιμώνται για λόγους ταχύτητας.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

8 Αν η πολυπλοκότητα ενός αλγορίθμου είναι f(n), τότε λέγεται ότι ο αλγόριθμος είναι τάξης O(g(n)) αν υπάρχουν δύο θετικοί ακέραιοι c και n0, έτσι ώστε για κάθε n>=n0 να ισχύει: ¦f(n)¦ > c ¦g(n)¦

9 Ο συμβολισμός $O(n^2)$ εκφράζει την Κυβική πολυπλοκότητα και πρέπει να χρησιμοποιείται μόνο για προβλήματα μεγάλου μεγέθους.

10 Ο συμβολισμός O(n) εκφράζει τη γραμμική πολυπλοκότητα η οποία είναι η καλύτερη επίδοση για έναν αλγόριθμο που πρέπει να εξετάσει ή να δώσει στην έξοδο n στοιχεία.

11 Ευριστικοί λέγονται οι αλγόριθμοι που δεν ειναι τυποποιημένοι και δεν ανήκουν στις γνωστές οικογένειες αλγορίθμων.

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

12. Μία βασική πράξη μπορεί να είναι :

Α) βρόχος επανάληψης

Β) ανάθεση τιμής

Γ) σύγκριση μεταξύ δύο μεταβλητών

Δ) συνθήκη Αν..αλλιώς

Ε) οποιαδήποτε αριθμητική πράξη μεταξύ δύο μεταβλητών

13. Ο χρόνος εκτέλεσης κάθε αλγορίθμου εξαρτάται από ένα σύνολο παραγόντων που περιλαμβάνουν τα εξής:

Α) Τύπος ηλεκτρονικού υπολογιστή που θα εκτελέσει το πρόγραμμα του αλγορίθμου

Β) Χρονική στιγμή εκτέλεσης του αλγορίθμου

Γ) Συνθήκες ανταγωνισμού με άλλους αλγορίθμους

Δ) Γλώσσα προγραμματισμού που θα χρησιμοποιηθεί

Ε) Δομή προγράμματος και δομές δεδομένων που χρησιμοποιεί

Ζ) Αριθμός εντολών του αλγορίθμου

Η) Χρόνος για πρόσβαση στο δίσκο και στις ενέργειες εισόδου-εξόδου

Θ) Είδος συστήματος, ενός χρήστη ή πολλαπλών χρηστών.

ΚΕΦΑΛΑΙΟ 6ο

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

6.1. Πζοσδοκώμενα αποτελέσματα

Στο κεφάλαιο αυτό θα γνωρίσεις τις βασικές έννοιες και τις βασικές αρχές των τεχνικών και των μεθόδων που θα χρησιμοποιήσεις στη συνέχεια για την ανάπτυξη των προγραμμάτων σου. Θα γνωρίσεις τα χαρακτηριστικά των κυριοτέρων γλωσσών προγραμματισμού και τις τεχνικές της ιεραρχικής σχεδίασης και του τμηματικού προγραμματισμού. Ιδιαίτερο βάρος πρέπει να δώσεις στα χαρακτηριστικά και κυρίως στα πλεονεκτήματα του δομημένου προγραμματισμού.

Πρέπει να είσαι σε θέση να περιγράφεις όλη τη διαδικασία δημιουργίας και εκτέλεσης ενός προγράμματος και τέλος να εκτελέσεις ένα απλό έτοιμο πρόγραμμα στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σου.

6.2. Δζαστηζιότητες - ασκήσεις

Στην τάξη

ΔΤ1. Ο παρακάτω αλγόριθμος αποτελεί τμήμα μη δομημένου προγράμματος.

Να γράψεις αλγόριθμο σχεδιασμένο με τις αρχές του δομημένου προγραμματισμού, που να εκτελεί τις ίδιες λειτουργίες.

```
ΑΡΧΗ
ΟΣΟ συνθήκη1 ΕΠΑΝΑΛΑΒΕ
    Εντολή 2
    ΑΝ συνθήκη3 ΤΟΤΕ
           Εντολή4
           Πήγαινε στο Τέλος
    ΑΛΛΙΩΣ
           Εντολή5
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ
```

ΔΤ2. Σχεδίασε ένα διάγραμμα που να δείχνει τις διάφορες λειτουργίες και τις σχέσεις ανάμεσα σε αυτές τις λειτουργίες για το πρόγραμμα που συγκεντρώνει και επεξεργάζεται τα δεδομένα του προβλήματος της δραστηριότητας ΔΤ1 του πρώτου κεφαλαίου του τετραδίου σου, την έρευνα της *SOS Ρατσισμός*.

Το πρόβλημα αυτό περιλαμβάνει τη συγκέντρωση των απαντήσεων, την επεξεργασία τους και την παρουσίαση σε μορφή πίνακα των αποτελεσμάτων για όλες τις χώρες της Ευρωπαϊκής Ένωσης,

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Στο προγραμματιστικό περιβάλλον να εκτελέσετε ένα έτοιμο πρόγραμμα.

Να κάνετε αλλαγές με το συντάκτη σε μία εντολή, να το μεταγλωττίσετε, να σημειώσετε τα λάθη, να τα διορθώσετε (ουσιαστικά να επαναφέρετε το πρόγραμμα στην αρχική του μορφή) και τελικά να πάρετε τα αποτελέσματα.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Να καταγράψεις σε έναν κατάλογο τα ονόματα και κυρίως τα βασικά χαρακτηριστικά των γλωσσών προγραμματισμού. Συγκεκριμένα για κάθε γλώσσα προγραμματισμού να καταγράφεις: το όνομα της, τις διάφορες εκδόσεις της, το είδος των δραστηριοτήτων για την οποία χρησιμοποιείται, τη διάδοσή της σε προσωπικούς υπολογιστές, και το είδος του προγραμματισμού που υποστηρίζει, για παράδειγμα αντικειμενοστραφή, οπτικό, οδηγούμενο από τα γεγονότα κλπ.

Για τη δημιουργία του καταλόγου να ανατρέξεις στα βιβλία σου και σε όποια άλλη πηγή έχεις διαθέσιμη.

ΔΣ2. Ο παρακάτω αλγόριθμος αποτελεί τμήμα μη δομημένου προγράμματος.

Να γράψεις αλγόριθμο σχεδιασμένο με τις αρχές του δομημένου προγραμματισμού, που να εκτελεί τις ίδιες λειτουργίες.

```
ΑΡΧΗ
ΑΝ συνθήκη1 ΤΟΤΕ
       Εντολή1
          ΑΝ συνθήκη2 ΤΟΤΕ
       Εντολή2
       Εντολή3
       Πήγαινε στην Εντολή5
   ΤΕΛΟΣ_ΑΝ
   Εντολή4
   Εντολή5
   Πήγαινε στην Αρχή
ΤΕΛΟΣ_ΑΝ
Εντολή3
ΤΕΛΟΣ
```

6.3.Τεστ αυτοαξιολόγησης

Συμπλήρωσε τα κενά με τη σωστή λέξη που λείπει

1. Ο μεταγλωττιστής μεταγλωττίζει το _____ πρόγραμμα σε αντικείμενο πρόγραμμα.
2. Ο τμηματικός προγραμματισμός υλοποιεί την _____ σχεδίαση του προγράμματος.
3. Οι γλώσσες που υλοποιούν τον _____ και τον _____ διευκολύνουν την ανάπτυξη εφαρμογών σε γραφικά περιβάλλοντα.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

4. Η Visual Basic είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού.
5. Οι εντολές στις συμβολικές γλώσσες αποτελούνται από ακολουθίες 0 και 1.
6. Ο δομημένος προγραμματισμός εξασφαλίζει τη δημιουργία σωστών προγραμμάτων.
7. Οι γλώσσες 4^{ης} γενιάς είναι κατάλληλες για ανάπτυξη γενικών εφαρμογών.

Διάλεξε ένα μεταξύ των προτεινόμενων

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

   Α. Συντάκτης

   Β. Μεταγλωττιστής

   Γ. Διερμηνευτής

   Δ. Συνδέτης

9. Η Pascal είναι μία γλώσσα:

   Α. Μηχανής

Β. Υψηλού επιπέδου

Γ. Συμβολική

Δ. 4ης γενιάς

10. Ο μεταγλωττιστής επισημαίνει:

Α. Όλα τα λάθη του προγράμματος

Β. Μόνο τα λογικά λάθη του προγράμματος

Γ. Μόνο τα συντακτικά λάθη του προγράμματος

Δ. Μόνο τα λάθη που προέρχονται από αναγραμματισμό των εντολών

11. Ο δομημένος προγραμματισμός είναι:

Α. μία γενική μεθοδολογία ανάπτυξης προγραμμάτων

Β. ένας τρόπος προγραμματισμού που εφαρμόζεται μόνο από τη γλώσσα Pascal

Γ. η εξέλιξη του τμηματικού προγραμματισμού

Δ. ένας τρόπος να εξαλείψουμε τις εντολές GOTO από ένα πρόγραμμα.

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

12. Ποια από τα παρακάτω είναι χαρακτηριστικά ενός δομημένου προγράμματος:

Α. Δομικό στοιχείο είναι τα αντικείμενα.

Β. Έχει μία είσοδο και μία έξοδο.

Γ. Χρησιμοποιεί τις τρεις δομές: της ακολουθίας, της επιλογής και της επανάληψης.

Δ. Μπορεί να εκμεταλλευτεί τους παράλληλους υπολογιστές.

13. Κάθε φυσική γλώσσα προσδιορίζεται από:

Α. Το αλφάβητο της

Β. Το λεξιλόγιο της

Γ. Τη γραμματική της

Δ. Τη σημασιολογία της

14. Ποιες από τις παρακάτω γλώσσες χρησιμοποιούνται για ανάπτυξη εφαρμογών τεχνητής νοημοσύνης:

Α. LISP

Β. FORTRAN

Γ. COBOL

Δ. PROLOG

Ε. JAVA

ΚΕΦΑΛΑΙΟ 7ο

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

7.1. Πζοσδοκώμενα αποτελέσματα

Το κεφάλαιο αυτό ουσιαστικά αποτελεί την πρώτη σου επαφή με προγραμματιστικό περιβάλλον. Παρουσιάζονται τα βασικά στοιχεία προγραμματισμού και σταδιακά δομείται η προγραμματιστική τακτική. Ανεξάρτητα από το πραγματικό περιβάλλον προγραμματισμού που στη συνέχεια θα δουλέψεις, θεμελιώδεις κανόνες αρχές και έννοιες που πρωτοπαρουσιάζονται σε αυτό το κεφάλαιο, όπως οι έννοιες της μεταβλητής και της σταθεράς, η εκχώρηση τιμής είναι αναγκαίο να γίνουν κτήμα σου. Η σωστή αντίληψη και εικόνα που θα πρέπει να σχηματίσεις, θα σε βοηθήσει να ασχοληθείς με μεγαλύτερη ευκολία με άλλα προγραμματιστικά περιβάλλοντα.

Οι λυμένες ασκήσεις του κεφαλαίου παρουσιάζονται αρχικά στο περιβάλλον της ιδεατής γλώσσας προγραμματισμού ΓΛΩΣΣΑ και στη συνέχεια κάποιες από αυτές παρουσιάζονται στα πραγματικά προγραμματιστικά περιβάλλοντα Basic και Pascal.

7.2. Επιπλέον παζαδείγματα

Παράδειγμα 1

Μία μπάλα η οποία εκτοξεύεται στον αέρα ακολουθεί μία παραβολική τροχιά μέχρι να πέσει πάλι στη γη. Αν θεωρήσουμε την αντίσταση του αέρα αμελητέα, και αν αγνοήσουμε την καμπυλότητα της γης, τότε το ύψος της μπάλας σε κάθε χρονική στιγμή δίνεται από τον τύπο

$$Y(t) = y_0 + v_{y0}t + \frac{1}{2}gt^2$$

v_0 : αρχική ταχύτητα

y_0 : Το αρχικό ύψος από τη γη

g : Η επιτάχυνση της βαρύτητας

θ: η γωνία βολής

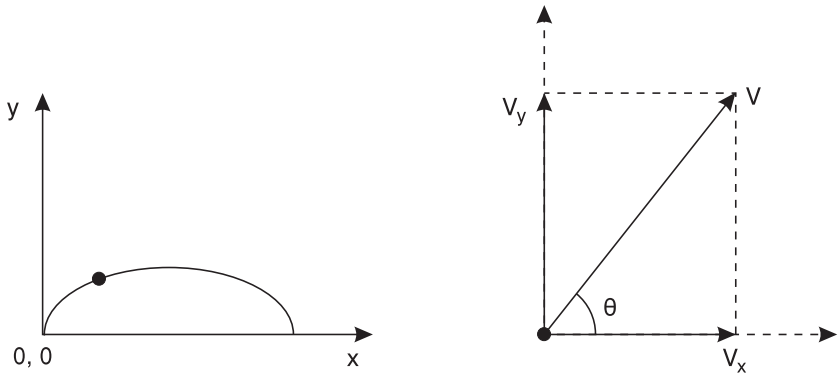
v_{y0} : Η προβολή της αρχικής ταχύτητας στον άξονα των y, $v_{y0} = v_0$ συνθ

ενώ η οριζόντια απόσταση είναι $x(t) = x_0 + v_{x0}t$

x_0 : Η αρχική οριζόντια θέση

$v_{0:}$ Η προβολή της αρχικής ταχύτητας στον άξονα των x, $v_{x0} = v_0$ ημθ

Αν θεωρήσουμε ότι η μπάλα εκτοξεύεται από το σημείο 0,0 τότε, όπως φαίνεται από τους τύπους παραπάνω, θα πέσει στη γη σε απόσταση $x = 2v_{x0}v_{y0}/g$, το οποίο ονομάζεται βεληνεκές.



Να γραφεί πρόγραμμα που να υπολογίζει την απόσταση x στην οποία η μπάλα θα πέσει στη γη (το βεληνεκές). Θεωρείστε ότι η μπάλα ξεκινάει από το σημείο με συντεταγμένες 0,0 και με αρχική ταχύτητα 20 m/sec. Η γωνία θ δίνεται από το πληκτρολόγιο κατά την ώρα της εκτέλεσης.

Για τον υπολογισμό των τριγωνομετρικών συναρτήσεων ημίτονο και συνημίτονο η γωνία βολής θ στο περιβάλλον της ΓΛΩΣΣΑΣ εκφράζεται σε μοίρες. Στα πραγματικά προγραμματιστικά περιβάλλοντα θα πρέπει να προσέξεις, αν υπάρχει δυνατότητα έκφρασης της γωνίας σε μοίρες, ή όπως συμβαίνει συνήθως, υποχρεωτικά θα πρέπει η γωνία να εκφράζεται σε ακτίνια. Υπενθυμίζεται ότι **2π ακτίνια = 360 μοίρες**.

Περιβάλλον προγραμματισμού ΓΛΩΣΣΑ

```
ΠΡΟΓΡΑΜΜΑ Τροχιά_μπάλας
! Πρόγραμμα υπολογισμού θέσης μπάλας
! Θεωρούμε την αντίσταση του αέρα αμελητέα και αγνοούμε την καμπυλότητα της γης

ΣΤΑΘΕΡΕΣ
G = 9.81
```

```
ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ:V0,VX0,VY0,Θ, Βεληνεκές

ΑΡΧΗ
! Αρχικές τιμές

V0 <- 20

! Εισαγωγή δεδομένων
ΓΡΑΨΕ 'Δώσε την γωνία
ΔΙΑΒΑΣΕ Θ

! Υπολογισμοί
VX0 <- V0*ΣΥΝ(Θ)
VY0 <- V0*ΗΜ(Θ)
Βεληνεκές <- 2*VX0*VY0/G

! Εμφάνιση αποτελεσμάτων
ΓΡΑΨΕ  'Το βεληνεκές για γωνία ',Θ,' είναι:', Βεληνεκές

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Τροχιά_Μπάλας
```

Περιβάλλον προγραμματισμού PASCAL

```
program ball;
const
   g=9.81;
   pi=3.14;
var
   a:integer;
   range,akt, v0,vx0,vy0:real;
begin
   {αρχικές τιμές}  vo:=20;
   {εισαγωγή δεδομένων}
   write('ΔΩΣΕ ΤΗ ΓΩΝΙΑ :');
   readln(a);
   {μετατροπή της γωνίας σε ακτίνια}
   akt:=a*pi/180;
   vx0:=v0*cos(akt);
   vy0:=v0*sin(akt);
   range:=2*vx0*vy0/g;
writeln('ΓΩΝΙΑ',A:3,' ΒΕΛΗΝΕΚΕΣ:',range:7:2);
end.
```

Περιβάλλον προγραμματισμού BASIC

```
'τροχιά μπάλας
g = 9.81:pi=3.14
v0 = 20
```

```
INPUT "Γωνία=", a
a = a * pi / 180
vx0 = v0 * COS(a)
vy0 = v0 * SIN(a)
vel = 2 * vx0 * vy0 / g
PRINT "Βεληνεκές ="; vel
END
```

Παράδειγμα 2

Η αγορά ενός αυτοκινήτου πολύ συχνά γίνεται με δόσεις. Ο υπολογισμός της δόσης εξαρτάται από την τιμή του αυτοκινήτου, την προκαταβολή, το επιτόκιο και τέλος την περίοδο αποπληρωμής.

Συγκεκριμένα δίνεται από τον τύπο:

$$a = i(p-d)\frac{(1+i)^m}{(1+i)^m - 1}$$

a: Μηνιαία δόση

p: Αρχική τιμή αυτοκινήτου

d: Ποσό προκαταβολής

i: Μηνιαίο επιτόκιο

m: Περίοδος αποπληρωμής σε μήνες

Να γραφεί πρόγραμμα το οποίο να διαβάζει την τιμή του αυτοκινήτου, το ποσό της προκαταβολής, το επιτόκιο και την περίοδο αποπληρωμής και στη συνέχεια να υπολογίζει το ποσό της κάθε δόσης καθώς και το ποσοστό επιβάρυνσης της τιμής του αυτοκινήτου από την αρχική του αξία.

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_δόσεων
!Υπολογισμός δόσεων αγοράς αυτοκινήτου
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:Τιμή, Προκαταβολή, Περίοδος, Υπόλοιπο
    ΠΡΑΓΜΑΤΙΚΕΣ: Ετήσιο_επιτόκιο, Επιτόκιο, Τελικό_ποσό, Δόση
!Τιμή:Αρχική τιμή αυτοκινήτου σε δρχ
!Προκαταβολή:Ποσό προκαταβολής σε δρχ
!Δόση: Μηνιαία δόση σε δρχ
!Ετήσιο_επιτόκιο:ετήσιο επιτόκιο σε ποσοστό %
!Επιτόκιο:Μηνιαίο επιτόκιο σε δεκαδικό αριθμό
!Περίοδος: Περίοδος αποπληρωμής σε μήνες
!Τελικό_ ποσό: Συνολικό ποσό όλων των δόσεων
!Ποσοστό:Ποσοστό διαφοράς  τιμής δόσεων με αρχική τιμή
ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε την αρχική τιμή: '
    ΔΙΑΒΑΣΕ Τιμή
    ΓΡΑΨΕ 'Δώσε την προκαταβολή: '
    ΔΙΑΒΑΣΕ Προκαταβολή
    ΓΡΑΨΕ 'Δώσε επιτόκιο: '
```

```
    ΔΙΑΒΑΣΕ Ετήσιο_επιτόκιο
    ΓΡΑΨΕ 'Δώσε περίοδο σε μήνες:' '
    ΔΙΑΒΑΣΕ Περίοδος
    Επιτόκιο <- (Ετήσιο_επιτόκιο/12)/100
    Υπόλοιπο <- Τιμή-Προκαταβολή
    Κλάσμα <- (1+Επιτόκιο)^Περίοδος/((1+Επιτόκιο)^Περίοδος-1)
    Δόση <- Επιτόκιο*Υπόλοιπο*Κλάσμα
    Τελικό_ποσό <- Δόση*Περίοδος+Προκαταβολή
    Ποσοστό <- Τελικό_ποσό/Τιμή*100
    ΓΡΑΨΕ ' Η μηνιαία δόση είναι:', Δόση
    ΓΡΑΨΕ ' Η συνολικό ποσό είναι:', Τελικό_ποσό,
    ΓΡΑΨΕ ' η αύξηση από το αρχικό είναι ', Ποσοστό ,'%'
ΤΕΛΟΣ-ΠΡΟΓΡΑΜΜΑΤΟΣ Υπολογισμός_δόσεων
```

7.3. Συμβουλές - υποδείξεις

Τώρα που αρχίζεις να γράφεις προγράμματα, καλό και χρήσιμο θα ήταν να μάθεις να ακολουθείς κάποιους κανόνες και κάποιες γενικές αρχές έτσι ώστε, η συγγραφή, η κατανόηση και η τροποποίηση των προγραμμάτων σου να γίνεται εύκολα και γρήγορα. Τη σημασία όλων αυτών θα τη νοιώθεις όλο και περισσότερο όσο τα προγράμματα γίνονται περισσότερο σύνθετα και πολύπλοκα.

- Τα προγράμματά σου πρέπει να είναι **απλά** και **κατανοητά**. Όχι μόνο για τους άλλους που θα χρειαστεί κάποια στιγμή να τα διαβάσουν και να τα καταλάβουν, αλλά και για σένα τον ίδιο που μετά από κάποιο καιρό θα επανέλθεις σε παλαιότερό σου πρόγραμμα για να το τροποποιήσεις ή να το επεκτείνεις.
- να χρησιμοποιείς ονόματα σταθερών και μεταβλητών που να υπονοούν τη χρήση τους.
- να γράφεις σχόλια μέσα στο πρόγραμμά σου, ειδικά σε εκείνα τα σημεία του που υπάρχουν σχετικές δυσκολίες στην κατανόηση.
- η χρήση κενών γραμμών διευκολύνει στην ανάγνωση του προγράμματος και οριοθετεί τις ενότητές του.
- η χρησιμοποίηση σταθερών σε διευκολύνει σε πιθανές επόμενες αλλαγές και σε προστατεύει από ενδεχόμενες αθέλητες τροποποιήσεις.
- Θα πρέπει να αποδίδεις αρχικές τιμές στις μεταβλητές που χρησιμοποιείς στο πρόγραμμα. Σε πολλά προγραμματιστικά περιβάλλοντα η ίδια η γλώσσα φροντίζει έτσι ώστε να αποδίδει αυτόματα αρχική τιμή ίση με μηδέν στις μεταβλητές. Η δυνατότητα αυτή όμως πολλές φορές μπορεί να οδηγήσει σε λάθος αποτελέσματα. Επιπλέον η απόδοση αρχικών τιμών βοηθάει στην καλύτερη κατανόηση του προγράμματος και στην ευκολότερη συντήρησή του.
- Να αποφεύγεις να χρησιμοποιείς μεγάλους υπολογισμούς. Η διάσπαση ενός υπολογισμού σε απλούστερους, διευκολύνει τους άλλους στην κατανόηση του προγράμματος και σένα στην αποφυγή λαθών.

7.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. **Να μετατρέψετε σε κώδικα προγράμματος τις παρακάτω παραστάσεις**

1. Η περίοδος γραμμικής αρμονικής ταλάντωσης είναι

$$T = 2\pi\sqrt{\frac{m}{D}}$$

2. Η κινητική ενέργεια ενός σώματος είναι :

$$E_{κιν} = \frac{1}{2}mv^2$$

3. Συνισταμένη δύο δυνάμεων που ενεργούν στο ίδιο σημείο και σχηματίζουν γωνία φ δίνεται από τον τύπο :

$$F = \sqrt{F_1^2 + F_2^2 + 2F_1F_2\sigma\upsilon\nu\varphi}$$

4. Η μία λύση εξίσωσης Β′ βαθμού είναι :

$$\frac{-\beta + \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha}$$

ΔΤ2. Τι τύπου μεταβλητές πρέπει να χρησιμοποιήσετε για τα παρακάτω στοιχεία του μαθητολόγιου του σχολείου μας; Γράψετε το αντίστοιχο τμήμα δηλώσεων.

1. Το όνομα ενός μαθητή.
2. Ο αριθμός μαθητολογίου του μαθητή.
3. Τη βαθμολογία του μαθητή.
4. Το τηλέφωνο ενός μαθητή.
5. Τη διεύθυνση ενός μαθητή.
6. Το φύλο ενός μαθητή (πώς μπορεί να οριστεί με χρήση λογικής μεταβλητής;)

ΔΤ3. Γράψτε το πρόγραμμα για το παρακάτω πρόβλημα και στη συνέχεια πραγματοποιήστε εικονική εκτέλεσή του έτσι ώστε να βεβαιωθείτε ότι λειτουργεί σωστά.

Δίδονται οι πλευρές ενός τριγώνου και υπολογίζεται το εμβαδόν του τριγώνου με τον τύπο του Ήρωνα

$$E = \sqrt{\tau(\tau-\alpha)(\tau-\beta)(\tau-\gamma)}$$

όπου τ είναι η ημιπερίμετρος του τριγώνου, δηλαδή $= \frac{\alpha+\beta+\gamma}{2}$.

Προβληματιστείτε πάνω στο ερώτημα "Μπορεί ο υπολογισμός αυτός να γίνεται για κάθε τριάδα αριθμών". Προσπαθήστε να δικαιολογήσετε τη απάντησή σας όσο καλύτερα μπορείτε.

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Γράψτε τον κώδικα και εκτελέστε το πρόγραμμα για το παράδειγμα 2, "Υπολογισμός δόσεων αγοράς αυτοκινήτου". Εκτελέστε το πρόγραμμα για διάφορες τιμές προκαταβολής και περιόδους αποπληρωμής.

ΔΕ2. Η απόσταση μεταξύ δύο σημείων (x_1,y_1) και (x_2,y_2) ενός Καρτεσιανού συστήματος συντεταγμένων υπολογίζεται από τον τύπο

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Γράψτε πρόγραμμα το οποίο να υπολογίζει και να εκτυπώνει την απόσταση δύο σημείων των οποίων οι συντεταγμένες δίνονται από το χρήστη.

ΔΕ3. Να γράψετε πρόγραμμα το οποίο διαβάζει το ονοματεπώνυμο, την τάξη και τη βαθμολογία σε τρία μαθήματα ενός μαθητή και υπολογίζει τον μέσο όρο του σε αυτά τα μαθήματα. Στη συνέχεια εκτυπώνει το όνομα του μαθητή, το τμήμα του και το μέσο όρο.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Η μετατροπή της θερμοκρασίας από βαθμούς Celsius σε Fahrenheit δίνεται από τον τύπο:

$$F = \frac{9}{5}C + 32$$

Να γραφεί πρόγραμμα το οποίο να διαβάζει τη θερμοκρασία σε βαθμούς Celsius και να την υπολογίζει και να την τυπώνει σε βαθμούς Fahrenheit.

ΔΣ2. Η περίοδος ενός εκκρεμούς δίνεται από τον τύπο

$$T = 2\pi\sqrt{\frac{L}{g}}$$

όπου L το μήκος του εκκρεμούς και g η επιτάχυνση της βαρύτητας.

Γράψτε πρόγραμμα το οποίο να υπολογίζει την περίοδο του εκκρεμούς. Το μήκος του εκκρεμούς θα δίνεται από το χρήστη κατά την εκτέλεση του προγράμματος.

ΔΣ3. Για να υπολογίσουμε τη ροή του αίματος στον ανθρώπινο οργανισμό χρησιμοποιούμε τον τύπο ροής υγρών σε σωλήνες. Για παράδειγμα, η ροή του αίματος στην αορτή (την βασική αρτηρία που μεταφέρει αίμα σε όλα τα όργανα εκτός από τους πνεύμονες) υπολογίζεται από τον τύπο ΡΟΗ= 5500πρ^4, όπου ρ η ακτίνα της αορτής. Μία υγιής αορτή έχει διάμετρο περίπου 0,02m. Η μείωση της διαμέτρου (στένωση) της αορτής προκαλεί σοβαρά καρδιαγγειακά νοσήματα αφού οποιαδήποτε στένωση προκαλεί πολύ μεγάλη μείωση της ροής αίματος. Για παράδειγμα, στένωση κατά 33% της αορτής προκαλεί μείωση κατά 80% της ροής του αίματος, με πολύ σοβαρές επιπλοκές στην υγεία του ανθρώπου.

Να γράψετε πρόγραμμα το οποίο να υπολογίζει τη ροή του αίματος σε μία φυσιολογική αορτή (με ακτίνα 0.01m) και την ποσοστιαία μεταβολή της ροής που επέρχεται με μείωση της ακτίνας της αορτής κατά 10%, 33% και 50%.

7.5. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες προτάσεων. Σε κάθε μια από αυτές, να βάλετε τις προτάσεις στη σωστή σειρά με την οποία θα πρέπει να γράφονται σε ένα πρόγραμμα

1.

Α. Δήλωση μεταβλητών
Β. Δήλωση σταθερών
Γ. Επικεφαλίδα προγράμματος
Δ. Εντολή εισόδου ΔΙΑΒΑΣΕ

2.

Α. ΓΡΑΨΕ 'Η συνολική τιμή είναι', Τιμή
Β. ΔΙΑΒΑΣΕ Ν
Γ. Κοστος <- Ν * 100
Δ. Τιμη <- Κοστος + Κοστος * 0.18

Συμπλήρωσε τα κενά με τη σωστή λέξη που λείπει

3. Τα στοιχεία προγράμματος των οποίων η τιμή μπορεί να μεταβληθεί κατά τη διάρκεια εκτέλεσης ενός προγράμματος, ονομάζονται _____

4. Η τελευταία εντολη κάθε προγράμματος είναι _____

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

5. Η δήλωση των μεταβλητών που χρησιμοποιούνται σε ένα πρόγραμμα είναι υποχρεωτική.

6. Το σύμβολο της εντολής εκχώρησης είναι το ίσον =.

7. Κατά τον υπολογισμό μιας αριθμητικής παράστασης, πρώτα εκτελείται ο πολλαπλασιασμός και στη συνέχεια η πρόσθεση.

8. Οι λογικές μεταβλητές δέχονται μόνο δύο τιμές.

Διάλεξε ένα μεταξύ των προτεινόμενων

9. Ποιες από τις παρακάτω εντολές δίνουν σαν αποτέλεσμα εκτέλεσης το μήνυμα: Η τιμή είναι 100

   Α.
   ```
   Τιμή <— 100
   ΓΡΑΨΕ 'Η τιμή είναι' 100
   ```

   Β.
   ```
   ΓΡΑΨΕ 'Η τιμή είναι', Τιμή
   ```

   Γ.
   ```
   Τιμή <— 100
   ΓΡΑΨΕ 'Η τιμή είναι', 100
   ```

   Δ.
   ```
   Τιμή <— 100
   ΓΡΑΨΕ 'Η τιμή είναι', Τιμή
   ```

10. Μετά την εκτέλεση της εντολής `Y <- 5*(X-3)+X^3-2+Z` ποια είναι η τιμή της μεταβλητής Υ, αν Χ=5 και Ζ=1

    Α. 35 Β. 134 Γ. 22 Δ. 148

11. Τι θα τυπώσουν οι παρακάτω εντολές

    ```
    A <- 100
    X <- (2+T_P(A)*3/10)^2-(A+50)/5
    ΓΡΑΨΕ X
    ```

    Α. 22 Β. -3 Γ. 10.7 Δ. 25

12. Σε ένα πρόγραμμα έχουμε μία μεταβλητή Πλήθος την οποία θέλουμε να την αυξήσουμε κατά μία μονάδα. Ποια από τις εντολές εχει σαν αποτέλσμα την αύξηση αυτή

    Α. `Πλήθος +1 <- Πλήθος`

    Β. `Πλήθος <- Πλήθος+1`

    Γ. `Πλήθος <- +1`

    Δ. `Πλήθος = Πλήθος+1`

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

13. Τα είδη μεταβλητών που υποστηρίζει η ΓΛΩΣΣΑ είναι

    Α. ακέραιες Γ. μιγαδικές Ε. ημερομηνίες

    Β. πραγματικές Δ. χαρακτήρες Ζ. λογικές

14. Ποια από τα παρακάτω είναι δεκτά σαν ονόματα σταθερών:

    Α. Α Γ. 1Στοιχείο Ε. Τιμή-σε-$

    Β. Στοιχείο1 Δ. Φύλο μαθητή Ζ. ΤΑΧΥΤΗΤΑ

ΚΕΦΑΛΑΙΟ 8ο

ΕΠΙΛΟΓΗ ΚΑΙ ΕΠΑΝΑΛΗΨΗ

8.1. Πζοσδοκώμενα αποτελέσματα

Σε αυτό το κεφάλαιο θα γνωρίσεις τις σημαντικότερες εντολές που θα χρησιμοποιείς σε όλα τα προγράμματα σου. Οι εντολές αυτές εκφράζουν τις βασικές δομές του δομημένου προγραμματισμού: τη δομή της επιλογής και τη δομή της επανάληψης ή ανακύκλωσης όπως συχνά θα την ακούσεις να λέγεται. Η επιλογή υλοποιείται με την εντολή **ΑΝ** και τις διάφορες μορφές της καθώς και με την εντολή **ΕΠΙΛΕΞΕ**, ενώ η επανάληψη με τις εντολές , **ΟΣΟ_ΕΠΑΝΑΛΑΒΕ** και **ΜΕΧΡΙΣ ΟΤΟΥ** καθώς και με την εντολή **ΓΙΑ**. Η σωστή γνώση της χρήσης αυτών των εντολών και η γνώση των διαφορών που παρουσιάζουν, σου επιτρέπουν να επιλέγεις την καταλληλότερη για κάθε συγκεκριμένο πρόγραμμα.

Ο σκοπός σου δεν είναι να γράψεις απλά ένα πρόγραμμα το οποίο επιλύει το πρόβλημα, αλλά να χρησιμοποιήσεις τις εντολές που επιτρέπουν την σύνταξη του πιο απλού, σύντομου, κατανοητού και τελικά αποδοτικότερου προγράμματος.

Οι λυμένες ασκήσεις του κεφαλαίου αυτού, όπως και του προηγούμενου, παρουσιάζονται στο περιβάλλον της ιδεατής γλώσσας προγραμματισμού ΓΛΩΣΣΑ και μερικές από αυτές παρουσιάζονται στα πραγματικά προγραμματιστικά περιβάλλοντα Basic και Pascal.

8.2. Επιπλέον παραδείγματα

Παράδειγμα 1

Για τη μέτρηση της ποιότητας της ατμόσφαιρας στην Αθήνα, όπως και σε κάθε μεγάλης πόλης που έχει πρόβλημα μόλυνσης της ατμόσφαιρας μετρούνται συνεχώς τα επίπεδα συγκεκριμένων βλαβερών συστατικών της, που είναι γνωστοί ως ρύποι . Οι ρύποι αυτοί είναι το διοξείδιο του αζώτου (NO_2), το μονοξείδιο του άνθρακα (CO), το διοξείδιο του θείου (SO_2) το όζον (O_3) και ο καπνός.

Για τον περιορισμό της ρύπανσης σε περιπτώσεις που σημειώνεται σημαντική αύξηση των τιμών των ρύπων χρησιμοποιούνται τα όρια εκτάκτων μέτρων.

Τα όρια αυτά που ισχύουν για την περιοχή της Αθήνας για δύο από τους πλέον συχνά εμφανιζόμενους ρύπους O_3 και NO_2 παρουσιάζονται στον παρακάτω Πίνακα.

Ρύπος	Στάδιο Προειδοποίησης	Στάδιο λήψης μέτρων Α! βαθμίδας	Στάδιο λήψης μέτρων Β! βαθμίδας
NO_2 (μg/m3)	400	500	700
O_3 (μg/m3)	250	300	500

Να γραφεί πρόγραμμα το οποίο διαβάζει τις τιμές του NO_2 και του O_3 και να τυπώνει το αντίστοιχο μήνυμα σύμφωνα με το παρακάτω πίνακα.

Κάτω από το στάδιο προειδοποίησης	Στάδιο Προειδοποίησης	Στάδιο λήψης μέτρων Α! βαθμίδας	Στάδιο λήψης μέτρων Β! βαθμίδας
ΡΥΠΟΙ ΜΕΣΑ ΣΤΑ ΟΡΙΑ	ΠΡΟΣΟΧΗ ΥΨΗΛΟΙ ΡΥΠΟΙ	ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΕΚΤΑΚΤΑ ΜΕΤΡΑ	ΠΑΡΑ ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΑΠΑΓΟΡΕΥΣΗ ΚΥΚΛΟΦΟΡΙΑΣ

```
ΠΡΟΓΡΑΜΜΑ Ρύποι
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:NO2, O3

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε την τιμή του Διοξειδίου του αζώτου'
    ΔΙΑΒΑΣΕ  NO2
    ΓΡΑΨΕ 'Δώσε την τιμή του Όζοντος'
    ΔΙΑΒΑΣΕ  O3
    ΑΝ NO2>700 Ή O3 > 500 ΤΟΤΕ
        ΓΡΑΨΕ  'ΠΑΡΑ ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΑΠΑΓΟΡΕΥΣΗ ΚΥΚΛΟΦΟΡΙΑΣ'
    ΑΛΛΙΩΣ_ΑΝ NO2 > 500 Ή O3 > 300 ΤΟΤΕ
        ΓΡΑΨΕ  'ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΕΚΤΑΚΤΑ ΜΕΤΡΑ'
    ΑΛΛΙΩΣ_ΑΝ NO2 > 400 Ή O3 > 250 ΤΟΤΕ
        ΓΡΑΨΕ  'ΠΡΟΣΟΧΗ ΥΨΗΛΟΙ ΡΥΠΟΙ'
```

```
    ΑΛΛΙΩΣ
        ΓΡΑΨΕ ' Ρύποι μέσα στα όρια'
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Ρύποι
```

Η επιλογή του επιπέδου των ρύπων μπορεί να γίνει με πολλούς άλλους τρόπους. Μπορούν να χρησιμοποιηθούν εμφωλευμένα ΑΝ ή δύο διαφορετικές εντολές ΑΝ- ΑΛΛΙΩΣ_ΑΝ, ένα ΑΝ για το όζον και ένα δεύτερο για το διοξείδιο ή ακόμη και δύο εντολές ΕΠΙΛΕΞΕ.

Ο καλύτερος τρόπος λύσης εξαρτάται από το πρόβλημα και τα ζητούμενα αποτελέσματα. Η λύση που δόθηκε είναι η πιο σύντομη και η πιο απλή για το συγκεκριμένο πρόβλημα.

Περιβάλλον προγραμματισμού PASCAL

```
program rypoi;
var
    no2,o3:real;

begin
    write ('ΔΩΣΕ ΤΗΝ ΤΙΜΗ ΤΟΥ ΔΙΟΞΕΙΔΙΟΥ:'); readln (no2);
    write('ΔΩΣΕ ΤΗΝ ΤΙΜΗ ΤΟΥ ΟΖΟΝΤΟΣ: '); readln (o3);
    if (no2 > 700) or (o3 > 500) then
write('ΠΑΡΑ ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΑΠΑΓΟΡΕΥΣΗ ΚΥΚΛΟΦΟΡΙΑΣ')
    else if (no2 > 500) or (o3 > 300) then
write ('ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΕΚΤΑΚΤΑ ΜΕΤΡΑ')
    else if (no2 > 400) or (o3 > 250) then
write ('ΠΡΟΣΟΧΗ ΥΨΗΛΟΙ ΡΥΠΟΙ')
    else
write (' Ρύποι μέσα στα όρια')
    endif
end.
```

Περιβάλλον προγραμματισμού BASIC

```
' rypoi
INPUT "NO2=", NO2
INPUT "O3=", O3
IF NO2 > 700 OR O3 > 500 THEN
   PRINT "'ΠΑΡΑ ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΑΠΑΓΟΡΕΥΣΗ ΚΥΚΛΟΦΟΡΙΑΣ"
ELSEIF NO2 > 500 OR O3 > 300 THEN
   PRINT "'ΠΟΛΥ ΥΨΗΛΟΙ ΡΥΠΟΙ ΕΚΤΑΚΤΑ ΜΕΤΡΑ2"
ELSEIF NO2 > 400 OR O3 > 250 THEN
   PRINT "'ΠΡΟΣΟΧΗ ΥΨΗΛΟΙ ΡΥΠΟΙ"
ELSE
   PRINT "'Ρύποι μέσα στα όρια'"
END IF
END
```

Παράδειγμα 2

Ο λογαριασμός του νερού είναι τριμηνιαίος και υπολογίζεται με βάση την κατανάλωση νερού. Η αξία του νερού υπολογίζεται από τον παρακάτω πίνακα

Κατανάλωση/μήνα σε κυβικά μέτρα	Τιμή σε δρχ
0-5	117
5-20	178
20-27	514
27-35	720
>35	900

Στην αξία του νερού προστίθεται το πάγιο (έστω 500 δρχ), η αποχέτευση 40% της αξίας του νερού, άλλες επιβαρύνσεις 1% καθώς και το ΦΠΑ που είναι 18% στο σύνολο του λογαριασμού.

Να γραφεί πρόγραμμα που διαβάζει το ονοματεπώνυμο του καταναλωτή, τον αριθμό του μετρητή νερού την κατανάλωση (ανά τρίμηνο) και να υπολογίζει και να τυπώνει τα ποσά του λογαριασμού.

Η διαδικασία επαναλαμβάνεται συνεχώς για διάφορους καταναλωτές και τερματίζεται με την είσοδο του 0 ως αριθμού μετρητή.

```
ΠΡΟΓΡΑΜΜΑ ΛΟΓΑΡΙΑΣΜΟΣ_ΝΕΡΟΥ
ΣΤΑΘΕΡΕΣ
    ΦΠΑ=0.18
    ΤΙΜΗ1=117
    ΤΙΜΗ2=178
    ΤΙΜΗ3=514
    ΤΙΜΗ4=720
    ΤΙΜΗ5=900

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:Κωδικός, Πάγιο
    ΠΡΑΓΜΑΤΙΚΕΣ:Κατανάλωση, Τιμή, Αποχέτευση, Άλλα, Αξία_ΦΠΑ, Αξία,
    Τελική_Τιμή
    ΧΑΡΑΚΤΗΡΕΣ: Όνομα, Επώνυμο

ΑΡΧΗ
    ΓΡΑΨΕ 'Δώσε Αριθμό του μετρητή (0 για τέλος)'
    ΔΙΑΒΑΣΕ Μετρητής
    ΟΣΟ Μετρητής <> 0 ΕΠΑΝΑΛΑΒΕ
        ΓΡΑΨΕ 'Δώσε το Ονοματεπώνυμο'
        ΔΙΑΒΑΣΕ Επώνυμο, Όνομα
        ΓΡΑΨΕ 'Δώσε την Κατανάλωση'
        ΔΙΑΒΑΣΕ Κατανάλωση
        ΕΠΙΛΕΞΕ Κατανάλωση
```

```
        ΠΕΡΙΠΤΩΣΗ =< 15
            Αξία <- Κατανάλωση*ΤΙΜΗ1
        ΠΕΡΙΠΤΩΣΗ =< 60 ΤΟΤΕ
            Αξία <- 15*ΤΙΜΗ1+(Κατανάλωση-15)*ΤΙΜΗ2
        ΠΕΡΙΠΤΩΣΗ =< 81 ΤΟΤΕ
            Αξία <- 15*ΤΙΜΗ1+ +45*ΤΙΜΗ2+(Κατανάλωση-60)*ΤΙΜΗ3
        ΠΕΡΙΠΤΩΣΗ =< 105 ΤΟΤΕ
            Αξία <- 15*ΤΙΜΗ1+ 45*ΤΙΜΗ2+21*ΤΙΜΗ3+(Κατανάλωση-81)*ΤΙΜΗ4
        ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
            Αξία <- 15*ΤΙΜΗ1+ 45*ΤΙΜΗ2+21*ΤΙΜΗ3+24*ΤΙΜΗ4+Κατανάλωση
                                                        -105)*ΤΙΜΗ5
        ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
        Αποχέτευση <- Αξία*0.4
        Άλλα <- Αξία*0.01
        Τιμή <- Αξία+ Αποχέτευση+ Άλλα
        Αξία_ΦΠΑ <- Τιμή* ΦΠΑ
        Τελική_τιμή <- Τιμή + Αξία_ΦΠΑ
        ΓΡΑΨΕ 'Ο λογαριασμός του', Επώνυμο,' είναι ', Τελική_τιμή
        ΓΡΑΨΕ 'Αξία νερού:', Αξία,
        ΓΡΑΨΕ 'άλλα:', Αποχέτευση+ Άλλα, 'ΦΠΑ:', Αξία_ΦΠΑ
        ΓΡΑΨΕ 'Δώσε Αριθμό του επόμενου μετρητή (0 για τέλος)'
        ΔΙΑΒΑΣΕ Μετρητής
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παράδειγμα 3

Στο παράδειγμα 1 του προηγούμενου κεφαλαίου με την μπάλα που εκτοξεύεται στον αέρα η γωνία βολής μπορεί να μεταβάλλεται από 20 έως 80 μοίρες σε βήματα των 10 μοιρών . Επίσης η αρχική ταχύτητα μπορεί να μεταβάλλεται από 10μ/sec έως 40 μ/sec σε βήματα των 10 μ/sec.

Να γραφεί πρόγραμμα που να υπολογίζει την οριζόντια απόσταση (το βεληνεκές) για κάθε συνδυασμό γωνίας και αρχικής ταχύτητας.

Περιβάλλον προγραμματισμού ΓΛΩΣΣΑ

```
ΠΡΟΓΡΑΜΜΑ Τροχιά_μπάλας2

ΣΤΑΘΕΡΕΣ
    G = 9.81

ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ:V0,VX0,VY0,Θ, Βεληνεκές

ΑΡΧΗ
    ΓΙΑ Θ ΑΠΟ 20 ΜΕΧΡΙ 80 ΜΕ_ΒΗΜΑ 10
        ΓΡΑΨΕ  'Γωνία:',Θ
        ΓΙΑ V0 ΑΠΟ 10 ΜΕΧΡΙ 40 ΜΕ_ΒΗΜΑ 10
            VX0 <- V0*ΣΥΝ(Θ)
```

```
            VY0 <- V0*ΗΜ(Θ)
            Βεληνεκές <- 2*VX0*VY0/G
            ΓΡΑΨΕ  'Ταχύτητα:',V0,'Βεληνεκές:',Βεληνεκές
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Τροχιά_Μπάλας2
```

Περιβάλλον προγραμματισμού PASCAL

```
program ball_2;
const
   g=9.81;
   pi=3.14;
var
    v01,a:integer;
    range,akt,v,v0,vx0,vy0:real;
begin
 for a:=2 to 8 do
  begin
   {μετατροπή της γωνίας σε ακτίνια}
   akt:=a*10*pi/180;
   writeln ('ΓΩΝΙΑ :',a*10:5);
   for v01:=1 to 4 do
   begin
    v0:=v01*10;
    vx0:=v0*cos(akt);
    vy0:=v0*sin(akt);
    range:=2*vx0*vy0/g;
    writeln('ΜΕ ΑΡΧΙΚΗ ΤΑΧΥΤΗΤΑ:',v0:5,'ΒΕΛΗΝΕΚΕΣ:',range:7:2);
 end;
 end;
end.
```

Επειδή η Pascal δεν επιτρέπει το καθορισμό του βήματος για τις επαναλήψεις που υλοποιούνται με την εντολή For, το βήμα είναι πάντα 1 ή –1, πρέπει να χρησιμοποιηθεί κάποιο τέχνασμα.

Έτσι αντί η η γωνία βολής να μεταβάλλεται από 10 έως 80 με βήματα των 10 μοιρών, η μεταβλητή a μεταβάλλεται από 2 έως 8, αυξανόμενο κατά μονάδα σε κάθε επανάληψη και στη συνέχεια πολλαπλασιάζεται με 10 στην μετατροπή σε ακτίνια: akt:=a*10*pi/180.

Αντίστοιχα για την αρχική ταχύτητα η μεταβλητή v01 παίρνει τιμές από 1 ως 4 και στη συνέχεια πολλαπλασιάζεται με το 10 για να δώσει την αρχική ταχύτητα, v0:=v01*10.

Περιβάλλον προγραμματισμού BASIC

```
' Τροχιά μπάλλας 2
g=9.81
FOR a = 20 TO 80 STEP 10
  akt = a * ATN(1) * 4 / 180
  PRINT "Γωνία ";akt
  FOR v0 = 10 TO 40 STEP 10
    vx0 = v0 * COS(akt)
    vy0 = v0 * SIN(akt)
    vel = 2 * vx0 * vy0 / g
    PRINT "ΜΕ ΑΡΧΙΚΗ ΤΑΧΥΤΗΤΑ";v0
    PRINT "ΒΕΛΗΝΕΚΕΣ = "; vel
  NEXT v0
NEXT a
END
```

Η συνάρτηση ATN επιστρέφει το τόξο εφαπτομένης. Αρα ATN(1)=π/4, αφού εφ(π/4)=1.

8.3. Συμβουλές - υποδείξεις

Εφόσον όπως έχουμε αναφέρει πολλές φορές κάθε πρόγραμμα μπορεί να υλοποιηθεί με τη χρήση των τριών δομών της ακολουθίας, της επιλογής και της επανάληψης, αν μάθεις να χρησιμοποιείς σωστά τις εντολές επιλογές και επανάληψης, μπορείς να υλοποιήσεις σχεδόν οποιονδήποτε αλγόριθμο. Στην πραγματικότητα όμως μόνο η εξάσκηση και η πείρα θα σου εξασφαλίσουν τη δυνατότητα να συντάσσεις εύκολα και γρήγορα σωστά προγράμματα. Οι παρακάτω συμβουλές θα σε βοηθήσουν στη συγγραφή σωστών προγραμμάτων αποφεύγοντας μερικά από τα πιο συνηθισμένα λάθη που παρουσιάζονται.

- Όταν χρησιμοποιείς σύνθετες λογικές εκφράσεις, να προσέχεις την ιεραρχία των τελεστών. Είναι καλύτερο να χρησιμοποιείς πάντα παρενθέσεις, έστω και αν δεν είναι απαραίτητο, σε προφυλάσσει από πιθανά λάθη και αβλεψίες, ενώ ταυτόχρονα κάνει το πρόγραμμα πιο εύκολο στην κατανόηση του.
- Πριν χρησιμοποιήσεις εμφωλευμένα ΑΝ, σκέψου μήπως το ίδιο πρόγραμμα μπορεί να υλοποιηθεί απλούστερα με σύνθετες λογικές εκφράσεις, την εντολή ΑΝ-ΑΛΛΙΩΣ_ΑΝ ή κάποια άλλη εντολή επιλογής που πιθανόν να προσφέρει το υπολογιστικό περιβάλλον που χρησιμοποιείς.
- Οι μεταβλητές που ελέγχουν την επανάληψη του βρόχου ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ πρέπει υποχρεωτικά να αλλάζουν τιμή μέσα στο σώμα του βρόχου, αλλιώς ή δεν εκτελείται ποτέ ή συνηθέστερα δεν σταματάει η εκτέλεση του (ατέρμων βρόχος).
- Οι επαναλήψεις που υλοποιούνται με την εντολή ΟΣΟ, μπορεί να μην εκτελεστούν ούτε μία φορά, αφού ο έλεγχος γίνεται στην είσοδο του βρόχου, αντίθετα οι επαναλήψεις ΜΕΧΡΙΣ_ΟΤΟΥ θα πραγματοποιηθούν τουλάχιστον μία φορά.

- Η εντολή ΓΙΑ χρησιμοποιείται μόνο για προκαθορισμένο αριθμό επαναλήψεων. Αν λοιπόν ξέρεις τον αριθμό των επαναλήψεων ή μπορείς να τον υπολογίσεις, τότε να χρησιμοποιείς την εντολή ΓΙΑ.
- Ποτέ μη χρησιμοποιείς εντολές που αλλάζουν την αρχική τιμή, την τελική τιμή, το βήμα ή τη μεταβλητή που ελέγχει την επανάληψη μέσα σε ένα βρόχο ΓΙΑ. Αν και μερικές γλώσσες προγραμματισμού επιτρέπουν αυτές τις αλλαγές, να τις αποφεύγεις, γιατί οδηγούν σε προγράμματα δυσνόητα και συνήθως λανθασμένα.

8.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Αν η μεταβλητή Α έχει την τιμή 10, η μεταβλητή Β έχει την τιμή 5 και η μεταβλητή Γ έχει την τιμή 3 ποιες από τις παρακάτω εκφράσεις είναι αληθείς και ποιες ψευδείς.

Α. ΟΧΙ (Α >Β)

Β. Α > Β ΚΑΙ Α<Γ Η Γ=<Β

Γ. Α>Β ΚΑΙ (Α<Γ Η Γ=<Β)

Δ. Α = Β Η (Γ-Β) < 0

Ε. (Α > Β ΚΑΙ Γ< Β) Η (Β <> Γ ΚΑΙ Α< Γ)

ΔΤ2. Να γράψεις τις εντολές για τα παρακάτω

Α. Αν η Βαθμολογία (ΒΑΘΜΟΣ) είναι μεγαλύτερη από τον Μέσο όρο (ΜΟ) τότε να τυπώνει "Πολύ καλά", αν είναι ίση ή μικρότερη του Μέσου όρου μέχρι και 2 μονάδες να τυπώνει "Καλά" και όταν είναι μικρότερη του Μέσου όρου περισσότερο από 2 μονάδες να τυπώνει "Μέτρια".

Β. Αν το τμήμα (ΤΜΗΜΑ) είναι Γ1 και η βαθμολογία (ΒΑΘΜΟΣ) είναι μεγαλύτερη από 15 τότε να τυπώνει το επώνυμο (ΕΠΩΝΥΜΟ).

Γ. Αν η απάντηση (ΑΠΑΝΤΗΣΗ) δεν είναι Ν ή ν ή Ο ή ο τότε να τυπώνει το μήνυμα "Λάθος απάντηση...".

Δ. Αν ο αριθμός Χ είναι αρνητικός ή το ΗΜ(Χ)=0 τότε να τυπώνεται το μήνυμα "Λάθος δεδομένα...", αλλιώς να υπολογίζεται η παράσταση (Χ^2+5*Χ)/(Τ_Ρ(Χ)* ΗΜ(Χ)).

ΔΤ3. Τι αλλαγές πρέπει να γίνουν στο πρόγραμμα του παραδείγματος 1 ώστε να τυπώνει και ποιος από τους δύο ρύπους υπερέβη τα όρια λήψης μέτρων.

ΔΤ4. Εστω το παρακάτω τμήμα προγράμματος:

```
Κ <- 0
ΓΙΑ Ι ΑΠΟ 0 ΜΕΧΡΙ 100 ΜΕ_ΒΗΜΑ 5
   Α <- Ι^3
   Κ <- Κ+Α
   ΓΡΑΨΕ Ι, Α
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Κ
```

Πόσες φορές θα εκτελεστεί ο βρόχος;

Ποια η λειτουργία των εντολών;

Γράψτε τις παραπάνω εντολές χρησιμοποιώντας την εντολή επανάληψης ΟΣΟ και την εντολή επανάληψης ΜΕΧΡΙΣ_ΟΤΟΥ. Ποιον από τους τρεις τρόπους προτιμάς και γιατί.

ΔΤ5. Διάβασε προσεκτικά τα παρακάτω τμήματα προγράμματος. Ποια είναι τα λάθη; Διόρθωσέ τα, ώστε να λειτουργούν σωστά.

Α.

```
ΔΙΑΒΑΣΕ Μισθός
ΟΣΟ Μισθός <>0 ΕΠΑΝΑΛΑΒΕ
   Άθροισμα <- 0
   ΑΝ Μισθός > Μέγιστος ΤΟΤΕ
      Μέγιστος <- Μισθός
   ΤΕΛΟΣ_ΑΝ
   ΑΝ Μισθός < Ελάχιστος ΤΟΤΕ
      Ελάχιστος <- Μισθός
   ΤΕΛΟΣ_ΑΝ
   Άθροισμα <- Άθροισμα+Μισθός
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Β.

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
   Άθροισμα <- 0
   ΑΝ Μισθός > Μέγιστος ΤΟΤΕ
      Μέγιστος <- Μισθός
   ΤΕΛΟΣ_ΑΝ
   ΑΝ Μισθός < Ελάχιστος ΤΟΤΕ
      Ελάχιστος <- Μισθός
   ΤΕΛΟΣ_ΑΝ
   Άθροισμα <- Άθροισμα+Μισθός
   ΔΙΑΒΑΣΕ Μισθός
ΜΕΧΡΙΣ_ΟΤΟΥ Μισθός<>0
```

Γ.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
   Άθροισμα <- 0
   ΔΙΑΒΑΣΕ Μισθός
```

```
    ΑΝ Μισθός > Μέγιστος ΤΟΤΕ
       Μέγιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    ΑΝ Μισθός < Ελάχιστος ΤΟΤΕ
       Ελάχιστος <- Μισθός
    ΤΕΛΟΣ_ΑΝ
    Άθροισμα <- Άθροισμα+Μισθός
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Εκτέλεσε εικονικά τις εντολές στο χαρτί και σημείωνε τα αποτελέσματα που προκύπτουν. Με αυτόν τον τρόπο θα δεις τα λάθη και στη συνέχεια θα κάνεις τις διορθώσεις.

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Να γραφεί πρόγραμμα που να διαβάζει το βαθμό ενός μαθητή και να υπολογίζει την αντίστοιχη αξιολόγηση του με βάση το βαθμό του και σύμφωνα με τον παρακάτω πίνακα:

17,5 -20	Άριστα
15,5 –17,4	Πολύ καλά
13,5 – 15,4	Καλά
9,5 – 13,4	Μέτρια
0 – 9,4	Απορρίπτεται

Το πρόγραμμα να γραφεί με τους ακόλουθους τρόπους:

- Με εντολές ΑΝ ... ΤΟΤΕ
- Με εντολές ΑΝ ... ΤΟΤΕ ... ΑΛΛΙΩΣ_ΑΝ
- Με εμφωλευμένα ΑΝ.
- Με την εντολή ΕΠΙΛΕΞΕ

ΔΕ2. Στο κεφάλαιο 2 του βιβλίου σου παρουσιάστηκε και συζητήθηκε αναλυτικά ο Πολλαπλασιασμός αλλά Ρωσικά. Να γράψεις πρόγραμμα που να υλοποιεί τον αλγόριθμο αυτό. Το πρόγραμμα να εκτελεστεί για διάφορα ζεύγη τιμών.

ΔΕ3. Να γραφεί πρόγραμμα το οποίο θα εκτελεί κάποια από τις βασικές πράξεις πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση ανάμεσα σε δύο ακέραιους αριθμούς και θα εμφανίζει το αποτέλεσμα στην οθόνη.

Το πρόγραμμα θα ελέγχεται από το παρακάτω μενού επιλογής και θα σταματάει όταν ο χρήστης επιλέξει από το μενού την επιλογή έξοδο.

1. Πρόσθεση

2. Αφαίρεση
3. Πολλαπλασιασμό
4. Διαίρεση
5. Έξοδος

Δώσε επιλογή:__

ΔΕ4. Να επεκτείνεις το παράδειγμα 1, τον υπολογισμό της ατμοσφαιρικής ρύπανσης, έτσι ώστε να παίρνει 6 τιμές ανά ώρα από 5 διαφορετικούς σταθμούς μέτρησης για τους δύο ρύπους. Το πρόγραμμα

- να υπολογίζει τη μέση τιμή κάθε ρύπου ανά ώρα και ανά σταθμό
- να βρίσκει τη μέγιστη μέση τιμή για κάθε ρύπο
- να ελέγχει τις μέγιστες αυτές τιμές με τα όρια που δόθηκαν

Το πρόγραμμα να εκτελεστεί με δεδομένα τις πραγματικές τιμές ρύπων που μετρήθηκαν τη χθεσινή ημέρα. Οι τιμές αυτές δίδονται από το τμήμα ποιότητας της ατμόσφαιρας του ΥΠΕΧΩΔΕ και βρίσκονται στη διεύθυνση: www.minenv.gr

ΔΕ5. Να γραφεί πρόγραμμα το οποίο να υπολογίζει τη συνολική χωρητικότητα πυκνωτών και τη συνολική αντίσταση αντιστάσεων. Η συνολική αντίσταση R και η συνολική χωρητικότητα C δίνεται από τους τύπους

Σε σειρά

$$R = R_1 + R_2 + R_3 + \ldots$$

$$c = \frac{1}{c_1} + \frac{1}{c_2} + \frac{1}{c3} + \ldots$$

Σε παραλληλία

$$c = c_1 + c_2 + c_3 + \ldots$$

$$R = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \ldots$$

Το πρόγραμμα θα ελέγχεται από μενού επιλογής και θα τερματίζεται όταν ο χρήστης επιλέξει έξοδο.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Η φορολογία εισοδήματος φυσικών προσώπων υπολογίζεται από τις αρμόδιες υπηρεσίες του υπουργείου των Οικονομικών κλιμακωτά, με τη βοήθεια του παρακάτω πίνακα.

ΚΛΙΜΑΚΑ ΥΠΟΛΟΓΙΣΜΟΥ ΦΟΡΟΥ ΕΙΣΟΔΗΜΑΤΟΣ ΦΥΣΙΚΩΝ ΠΡΟΣΩΠΩΝ ΟΙΚΟΝ. ΕΤΟΥΣ 1999

Κλιμάκιο εισοδήματος	Φορολογικός συντελεστής	Φόρος κλιμακίου	Σύνολο	
			εισοδήματος	φόρου
1.055.000	0	0	1.055.000	0
1.582.500	5	79.125	2.637.500	79.125
1.582.500	15	237.375	4.220.000	316.500
3.165.000	30	949.500	7.385.000	1.266.000
8.440.000	40	3.376.000	15.825.000	4.642.000
Υπερβάλλον	45			

Για κάθε φορολογούμενο δίνονται τα εξής στοιχεία: αριθμός φορολογικού μητρώου (ΑΦΜ), όνομα φορολογούμενου, φορολογητέο εισόδημα

Να γραφτεί πρόγραμμα το οποίο:

Να διαβάζει τα στοιχεία των φορολογουμένων, να υπολογίζει και να τυπώνει το φόρο που τους αντιστοιχεί. Το πρόγραμμα θα διαβάζει τα στοιχεία πολλών φορολογουμένων και θα τελειώνει όταν διαβάζει για ΑΦΜ τον αριθμό 0.

ΔΣ2. Να γραφεί πρόγραμμα που να υπολογίζει τις ρίζες της δευτεροβάθμιας εξίσωσης $\alpha x^2+\beta x+\gamma=0$. Αν δεν υπάρχουν πραγματικές ρίζες, να εκτυπώνει αντίστοιχο μήνυμα.

ΔΣ3. Να γραφεί πρόγραμμα το οποίο διαβάζει το όνομα ενός μαθητή, τους βαθμούς του σε τρία μαθήματα και υπολογίζει και τυπώνει το μέσο όρο. Το πρόγραμμα να σταματάει, όταν για όνομα δοθεί το κενό.

ΔΣ4.Να γράψετε πρόγραμμα που να υπολογίζει τη συνάρτηση $y(x)=x^2-3x+2$ για όλες τις τιμές του x από –1 έως 3 σε βήματα του 0.1.

ΔΣ5.Ένας τρόπος υπολογισμού των τριγωνομετρικών συναρτήσεων, που χρησιμοποιείται συχνά από τους υπολογιστές είναι με τον υπολογισμό των παρακάτω σειρών:

$$\eta\mu x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

$$\sigma\upsilon\nu x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Να γράψεις πρόγραμμα το οποίο να διαβάζει τη γωνία x σε μοίρες και να υπολογίζει το ημίτονο και το συνημίτονο της σύμφωνα με τους παραπάνω τύπους.

Ποια μπορεί να είναι τα κριτήρια για διακοπή των επαναλήψεων;

Υπόδειξη: Να μετατρέψεις αρχικά τη γωνία x σε ακτίνια..

ΔΣ6. Να γραφεί ένα πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό και να τον αναλύει σε γινόμενο πρώτων παραγόντων.

8.5. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες εντολές. Σε κάθε μια από αυτές, να βάλετε τις εντολές στη σωστή σειρά με την οποία θα πρέπει να γράφονται σε ένα πρόγραμμα

1.

Α. ΓΡΑΨΕ 'Δεν υπάρχει ρίζα'

Β. ΑΝ Α>0 ΤΟΤΕ

Γ. ΤΕΛΟΣ_ΑΝ

Δ. ΑΛΛΙΩΣ

Ε. Ρίζα<-Τ_Ρ(Α)

2.

Α. ΜΕΧΡΙΣ_ΟΤΟΥ (Απάντηση='Ν' Ή Απάντηση='ν')

Β. ΔΙΑΒΑΣΕ Απάντηση

Γ. ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

Δ. ΓΡΑΨΕ 'Δώσε απάντηση :'

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

3. Οι εντολές που βρίσκονται σε ένα βρόχο ΟΣΟ ΕΠΑΝΑΛΑΒΕ εκτελούνται τουλάχιστον μία φορά.

4. Η τιμή του βήματος στην εντολή ΓΙΑ είναι υποχρεωτική να αναγράφεται.

5. Κάθε εντολή ΑΝ πρέπει να έχει την αντίστοιχη εντολή ΤΕΛΟΣ_ΑΝ.

6. Κάθε βρόχος που υλοποιείται με την εντολή ΟΣΟ ΕΠΑΝΑΛΑΒΕ μπορεί να γραφεί και με χρήση της εντολής ΓΙΑ.

7. Αν το Α έχει την τιμή 5 και το Β την τιμή 6 τότε η λογική έκφραση Α>5 Ή Α<3 ΚΑΙ Β>5 είναι ψευδής.

Διάλεξε ένα μεταξύ των προτεινόμενων

8. Ποιο από τα παρακάτω υπολογίζει το άθροισμα των 100 πρώτων περιττών αριθμών

Α.

```
Άθροισμα <- 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100
   Άθροισμα <- Άθροισμα+Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Β.

```
Άθροισμα <- 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ_ΒΗΜΑ 2
   Άθροισμα <- Άθροισμα+ Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Γ.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ_ΒΗΜΑ 2
   Άθροισμα <- 0
   Άθροισμα <- Άθροισμα+ Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Δ.

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ_ΒΗΜΑ 2
   Άθροισμα <- Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

9. Τι θα εκτυπώσει το παρακάτω τμήμα προγράμματος

```
Α <- 0
ΓΙΑ Ι ΑΠΟ 10 ΜΕΧΡΙ 20 ΜΕ_ΒΗΜΑ 10
   Α <- Α+Ι^2
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Α
```

Α. 0 Β. 100 Γ. 500 Δ. 400

10. Πόσες φορές θα εκτελεστεί η παρακάτω επανάληψη

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
   Α <- 0
   ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
      Α <- Α-1
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΜΕΧΡΙΣ_ΟΤΟΥ Α=0
```

Α. 10 Β. 0 Γ. 5 Δ. Άπειρες

11. Δίνονται οι παρακάτω εντολές

```
Α <- 1
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10 ΜΕ_ΒΗΜΑ 2
   Α <- Α*Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Ποιες από τις επόμενες ομάδες εντολών δίνουν στο Α την ίδια τιμή

Α.

```
Α <- 1
Ι <- 1
ΟΣΟ Ι<=10  ΕΠΑΝΑΛΑΒΕ
   Ι <- Ι+2
   Α <- Α*Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Β.

```
Α <- 1
Ι <- 1
ΟΣΟ Ι <=10 ΕΠΑΝΑΛΑΒΕ
       Α <- Α*Ι
       Ι <- Ι+2
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Γ.

```
Α <- 1
Ι <- 1
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
   Α <- Α*Ι
   Ι <- Ι+2
ΜΕΧΡΙΣ_ΟΤΟΥ Ι<10
```

Δ.

```
Α <- 1
Ι <- 1
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
       Α <- Α*Ι
       Ι <- Ι+2
ΜΕΧΡΙΣ_ΟΤΟΥ Ι=10
```

12. Πόσες φορές θα εκτελεστεί η παρακάτω επανάληψη

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 2 ΜΕ_ΒΗΜΑ 3
   ΓΡΑΨΕ 'Μήνυμα'
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Α. 2 Β. 0 Γ. 1 Δ. Άπειρες

13. Ποια η λειτουργία του παρακάτω τμήματος προγράμματος

```
Β <- 10
ΔΙΑΒΑΣΕ  Α
Β <- Α
ΑΝ Α < 0 ΤΟΤΕ
   Β <- -Α
ΤΕΛΟΣ_ΑΝ
Α <- 0
ΓΡΑΨΕ Β
```

Α. Τυπώνει τον αριθμό που διάβασε

Β. Τυπώνει την απόλυτη τιμή του αριθμού που διάβασε

Γ. Τυπώνει πάντα την τιμή 0

Δ. Τυπώνει πάντα την τιμή 10

ΚΕΦΑΛΑΙΟ 9ο

ΠΙΝΑΚΕΣ

9.1. Πζοσδοκώμενα αποτελέσματα

Στο κεφάλαιο αυτό θα μάθεις να χρησιμοποιείς στα προγράμματα σου τους πίνακες για την αποθήκευση μεγάλου αριθμού δεδομένων ιδίου τύπου.

Αρχικά πρέπει να αποφασίζεις, αν η χρήση της δομής του πίνακα σε βοηθάει στην υλοποίηση του προγράμματός σου.

Στη συνέχεια πρέπει να επιλέγεις το είδος του πίνακα που χρειάζεται και να μπορείς να τον ορίσεις σωστά, αλλά και να χειριστείς σωστά τα στοιχεία του. Συγκεκριμένα πρέπει να μπορείς να εισάγεις, να επεξεργάζεσαι και να τυπώνεις τα στοιχεία ενός πίνακα τόσο μονοδιάστατου όσο και δισδιάστατου.

Οι επεξεργασίες που απαιτούνται σε ένα πίνακα είναι συνήθως η αναζήτηση, η ταξινόμηση και η συγχώνευση. Μερικούς από τους αλγόριθμους για τις βασικές αυτές επεξεργασίες τις γνώρισες στο κεφάλαιο 3 και 4, εδώ θα έχεις την ευκαιρία να τους υλοποιήσεις σε προγραμματιστικό περιβάλλον.

Οι λυμένες ασκήσεις του κεφαλαίου αυτού, όπως και των προηγουμένων, παρουσιάζονται στο περιβάλλον της ιδεατής γλώσσας προγραμματισμού ΓΛΩΣΣΑ και μερικές από αυτές παρουσιάζονται στα πραγματικά προγραμματιστικά περιβάλλοντα Basic και Pascal.

9.2. Επιπλέον παραδείγματα

Παράδειγμα 1

Να γραφεί πρόγραμμα το οποίο διαβάζει τα ονόματα 50 αεροπορικών εταιρειών και τις αντίστοιχες εισπράξεις τους. Να τυπώνει τα ονόματα των εταιρειών που έχουν εισπράξεις περισσότερες από τον μέσο όρο.

```
ΠΡΟΓΡΑΜΜΑ Αεροπορικές_εταιρείες
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:Ν,Ι, Εισπράξεις[50], Σύνολο
    ΠΡΑΓΜΑΤΙΚΕΣ:ΜΟ
    ΧΑΡΑΚΤΗΡΕΣ:Εταιρεία[50]

ΑΡΧΗ
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΓΡΑΨΕ 'Αριθμός εταιρειών.. (μικρότερο από 50)'
        ΔΙΑΒΑΣΕ Ν
    ΜΕΧΡΙΣ_ΟΤΟΥ Ν<=50
    Σύνολο <- 0
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΓΡΑΨΕ 'Δώσε αεροπορική εταιρεία …'
        ΔΙΑΒΑΣΕ Εταιρεία[Ι]
        ΓΡΑΨΕ 'Δώσε εισπράξεις …'
    ΔΙΑΒΑΣΕ Εισπράξεις [Ι]
    Σύνολο <- Σύνολο+Εισπράξεις[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΜΟ <- Σύνολο/Ν
    ΓΡΑΨΕ 'Μεγαλύτερες από τον μέσο όρο'
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΑΝ Εισπράξεις[Ι]> ΜΟ ΤΟΤΕ
            ΓΡΑΨΕ Εταιρεία[Ι]
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Αεροπορικές_εταιρείες
```

Οι παραπάνω πίνακες λέγονται παράλληλοι. Δύο οι περισσότεροι πίνακες λέγονται παράλληλοι, αν σε αυτούς έχουμε αποθηκεύσει τα χαρακτηριστικά οντοτήτων με τέτοιο τρόπο ώστε τα δεδομένα κάθε οντότητας να βρίσκονται σε στοιχεία με την ίδια τιμή δείκτη.

Στο παραπάνω παράδειγμα οι πίνακες Εισπράξεις και Εταιρεία είναι παράλληλοι αφού τα στοιχεία που αναφέρονται σε κάθε γραμμή τους, δηλαδή το όνομα και οι εισπράξεις, αφορούν την ίδια εταιρεία.

Περιβάλλον προγραμματισμού PASCAL

```
program air_co;
var
  n,i,sum:integer;
```

```
  ave:real;
  tickets:array[1..50] of integer;
  company:array[1..50] of string;
begin
  repeat
    write('ΑΡΙΘΜΟΣ ΕΤΑΙΡΕΙΩΝ :');readln(n);
  until (n<=50);
  sum:=0;
  for i:=1 to n do
  begin
    write('ΕΤΑΙΡΕΙΑ :'); readln(company[i]);
    write('ΕΙΣΠΡΑΞΕΙΣ :');readln(tickets[i]);
    sum:=sum+tickets[i];
  end;
  ave:=sum/n;
  for i:=1 to n do
  begin
    if tickets[i] > ave then
       writeln (company[i]);
   end;
  end.
```

Περιβάλλον προγραμματισμού Basic

```
' Αεροπορικές εταιρίες
DIM company$(50), E(50)
DO
  INPUT "Αριθμός εταιριών:", n
LOOP UNTIL n <= 50
sum = 0
FOR i = 1 TO n
  PRINT "¨Εταιρία; i; " :";
  INPUT "", company$(i)
  PRINT "Εισπράξεις=";
  INPUT "", E(i)
  sum = sum + E(i)
NEXT i
MO = sum / n
PRINT "Μεγαλύτερες από το μέσο όρο"
PRINT "============================"
FOR i = 1 TO n
  IF E(i) > MO THEN PRINT company$(i)
NEXT i
END
```

Παράδειγμα 2

Μία εταιρεία κατασκευής αυτοκινήτων έχει μετρήσεις από το επίπεδο θορύβου όλων των μοντέλων της(σε decibel -dB). Οι μετρήσεις γίνονται για διαφορετικές ταχύτητες και δίνονται από το παρακάτω πίνακα

Μοντέλο	Ταχύτητα (km/h)				
	40	60	80	100	120
GX	88	90	93	105	112
LX	75	78	81	89	95
Gti	80	85	90	96	101
SX	68	78	85	102	105

Να γραφεί πρόγραμμα το οποίο θα υπολογίζει και θα τυπώνει το μέσο επίπεδο θορύβου για κάθε μοντέλο, το μέσο επίπεδο θορύβου για κάθε ταχύτητα και το συνολικό μέσο επίπεδο θορύβου όλων των αυτοκινήτων.

```
ΠΡΟΓΡΑΜΜΑ Αυτοκίνητα

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:Θόρυβος[4,5],I,J,Ταχύτητα[5],Άθροισμα, Συν_Αθροισμα
    ΧΑΡΑΚΤΗΡΕΣ:Μοντέλο[4]
    ΠΡΑΓΜΑΤΙΚΕΣ:ΜΟ,Συν_ΜΟ

!Εισαγωγή δεδομένων
    ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ 5
        ΓΡΑΨΕ 'Δώσε ταχύτητα.. '
        ΔΙΑΒΑΣΕ Ταχύτητα[I]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ 4
        ΓΡΑΨΕ 'Δώσε μοντέλο.. '
        ΔΙΑΒΑΣΕ Μοντέλο[I]
        ΓΡΑΨΕ 'Δώσε επίπεδα θορύβου.. '
        ΓΙΑ J ΑΠΟ 1 ΜΕΧΡΙ 5
            ΔΙΑΒΑΣΕ Θόρυβος[I,J]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

! Υπολογισμοί μέσων τιμών
    Συν_Άθροισμα <- 0
    ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ 4
        Άθροισμα <- 0
        ΓΙΑ J ΑΠΟ 1 ΜΕΧΡΙ 5
            Άθροισμα <- Άθροισμα+Θόρυβος[I,J]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        Συν_Άθροισμα <- Συν_Άθροισμα+Άθροισμα
        ΜΟ <- Άθροισμα/5
```

```
        ΓΡΑΨΕ Μοντέλο[Ι], ' :', ΜΟ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΙΑ J ΑΠΟ 1 ΜΕΧΡΙ 5
        Άθροισμα <- 0
        ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 4
            Άθροισμα <- Άθροισμα+Θόρυβος[Ι,J]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΜΟ <- Άθροισμα/4
        ΓΡΑΨΕ Ταχύτητα[J] ' :', ΜΟ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    Συν_ΜΟ <- Συν_Άθροισμα/20
    ΓΡΑΨΕ 'Συνολικό μέσο επίπεδο θορύβου :', Συν_ΜΟ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παράδειγμα 3

Δίνονται δύο ταξινομημένοι κατά αύξουσα σειρά μονοδιάστατοι πίνακες, ακεραίων αριθμών. Να γραφεί πρόγραμμα το οποίο να συγχωνεύει τους δύο πίνακες σε ένα τρίτο ο οποίος να είναι επίσης ταξινομημένος κατά αύξουσα σειρά. Οι δύο αρχικοί πίνακες δεν μπορούν να περιέχουν περισσότερα από 100 στοιχεία ο καθένας.

Η συγχώνευση είναι μία βασική λειτουργία των πινάκων και γενικότερα των δομών δεδομένων. Στη συνέχεια δίνεται ένας πολύ απλός αλγόριθμος συγχώνευσης δύο ταξινομημένων πινάκων σε ένα τρίτο ταξινομημένο πίνακα.

Θεωρείται ότι στην είσοδο του αλγορίθμου συγχώνευσης δίνονται δύο ταξινομημένοι, κατά αύξουσα σειρά, πίνακες Α και Β, μεγέθους Ν και Μ στοιχείων αντίστοιχα, ενώ στην έξοδο προκύπτει ένας τρίτος πίνακας Γ με Ν+Μ ταξινομημένα στοιχεία επίσης κατά αύξουσα σειρά.

Στο πρόγραμμα Συγχώνευση που ακολουθεί οι μεταβλητές i, j και k είναι δείκτες για την κίνηση μέσα στους πίνακες Α, Β και Γ. Η μέθοδος προχωρεί ως εξής:

Το μικρότερο στοιχείο από τους πίνακες Α και Β τοποθετείται στον πίνακα Γ με ταυτόχρονη αύξηση του αντίστοιχου δείκτη. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου τελειώσουν τα στοιχεία του ενός πίνακα.

Στη συνέχεια τα υπόλοιπα στοιχεία του άλλου πίνακα μεταφέρονται στον πίνακα Γ.

```
ΠΡΟΓΡΑΜΜΑ Συγχώνευση
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:Α[100], Β[100], Γ[200], Ι, J, Κ, Ν, Μ, Λ
    ! Α καιΒ αρχικοί πίνακες
    ! Γ τελικός πίνακας
ΑΡΧΗ

    ! Διάβασε τα δεδομένα
    ΓΡΑΨΕ 'Δώσε το πλήθος των στοιχείων του πίνακα Α (<100)'
    ΔΙΑΒΑΣΕ Ν
```

```
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
        ΔΙΑΒΑΣΕ Α[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    ΓΡΑΨΕ 'Δώσε το πλήθος των στοιχείων του πίνακα Β(<100)'
    ΔΙΑΒΑΣΕ Μ
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Μ
        ΔΙΑΒΑΣΕ Β[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    ! Συγχώνευση πινάκων

    ! Ι είναι ο δείκτης για τον πίνακα Α
    ! Ϳ είναι ο δείκτης για τον πίνακα Β
    ! Κ είναι ο δείκτης για τον πίνακα Γ
    Ι <- 1
    Ϳ <- 1
    Κ <- 1
    ΟΣΟ Ι <= Ν ΚΑΙ Ϳ <= Μ ΕΠΑΝΑΛΑΒΕ
    ! Όσο και τα δύο έχουν στοιχεία
        ΑΝ Α[Ι] < Β[Ϳ] ΤΟΤΕ
            Γ[Κ] <- Α[Ι]
            Κ <- Κ+1
            Ι <- Ι+1
        ΑΛΛΙΩΣ
            Γ[Κ] <- Β[Ϳ]
            Κ <- Κ+1
            Ϳ <- Ϳ +1
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    ! Μεταφορά των υπολοίπων στοιχείων του Α ή του Β
    ΑΝ Ι > Ν ΤΟΤΕ
        ΓΙΑ Λ ΑΠΟ Κ ΜΕΧΡΙ Ν+Μ
            Γ[Λ] <- Β[Ϳ]
            Ϳ <- Ϳ +1
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΑΛΛΙΩΣ
        ΓΙΑ Λ ΑΠΟ Κ ΜΕΧΡΙ Ν+Μ
            Γ[Λ] <- Α[Ι]
            Ι <- Ι+1
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    ! Εκτύπωση τελικού πίνακα

    ΓΙΑ Λ ΑΠΟ 1 ΜΕΧΡΙ Ν+Μ
        ΓΡΑΨΕ Γ[Λ]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Συγχώνευση
```

Γλώσσα προγραμματισμού PASCAL

```
program merge;

var
    ι,j,k,l,n,m: integer;
    a,b: array[1..100] of integer;
    c: array[1..200] of integer;

{Α καιΒ αρχικοί ταξινομημένοι πίνακες C τελικός πίνακας}

begin
    write  ('Δώσε τη διάσταση του πίνακα Α (n) '); readln(n);
    for i:=1 to n do
        readln(a[i]);
    write ('Δώσε τη διάσταση του πίνακα Β  (m) ' '); readln (m);
    for i:=1 to m do
        readln (b[i]);
    i:=1; j:=1; k:=1;
    while (i<=n) and (j<=m) do
        if (a[i]<b[j]) then
        begin
            c[k]:=a[i];k:=k+1;i:=i+1;
        end
        else
        begin
            c[k]:=b[j]; k:=k+1; j:=j+1;
        end;
    if i>n then
        for l:=k to n+m do
        begin
            c[l]:=b[j]; j:=j+1;
        end
    else
        for l:=k to n+m do
        begin
            c[l]:=a[i]; i:=i+1;
        end;
    for l:=1 to n+m
        write (c[l]);
end.
```

Περιβάλλον προγραμματισμού Basic

```
' Merging
DIM a(100), b(100), c(200)
READ n
FOR i = 1 TO n: READ a(i): NEXT i
DATA 5
DATA 2,7,12,18,26
```

```
READ m
FOR i = 1 TO m: READ b(i): NEXT i
DATA 5
DATA 1,6,10,15,25
i = 1: j = 1: k = 1
WHILE i <= n AND j <= m
  IF a(i) < b(j) THEN
     c(k) = a(i): k = k + 1: i = i + 1
  ELSE
     c(k) = b(j): j = j + 1: k = k + 1
  END IF
WEND
IF i > n THEN
   FOR r = k TO n + m
     c(r) = b(j): j = j + 1
   NEXT r
ELSE
   FOR r = k TO n + m
     c(r) = a(i): i = i + 1
   NEXT r
END IF
FOR i = 1 TO m + n
  PRINT c(i)
NEXT i
END
```

9.3. Συμβουλές - υποδείξεις

Η χρήση των πινάκων είναι ένας βολικός τρόπος για την αποθήκευση μεγάλου αριθμού δεδομένων ιδίου τύπου. Συνήθως οι νέοι προγραμματιστές χρησιμοποιούν πίνακες ακόμη και όταν η χρήση τους δεν είναι απαραίτητη.

- Εξέτασε αν πραγματικά χρειάζεται πίνακας για την επίλυση του προβλήματος. Αν δεν είναι απαραίτητος μην τον χρησιμοποιείς. Να έχεις πάντα στο νου σου ότι οι πίνακες ξοδεύουν μεγάλα ποσά μνήμης.
- Για να αποφύγεις τα πλέον κοινά λάθη στη χρήση των πινάκων να προσέχεις πάντα:
  - ✓ Να δίνεις αρχικές τιμές σε όλους τους πίνακες.
  - ✓ Μην ξεπερνάς τα όρια του πίνακα σου. Το πιο συνηθισμένο λάθος στη χρήση των πινάκων είναι η προσπάθεια ανάγνωσης ή εκχώρησης τιμής έξω από τα όρια του πίνακα.
  - ✓ Η επεξεργασία γίνεται στα στοιχεία του πίνακα. Άρα σε όλες τις εντολές πρέπει να εμφανίζονται τα στοιχεία του πίνακα και όχι το όνομα του ίδιου του πίνακα.

- ✓ Όλα τα στοιχεία του πίνακα έχουν τον ίδιο τύπο, για παράδειγμα όλα είναι ακέραια ή όλα είναι χαρακτήρες όπως ορίστηκαν στο τμήμα δηλώσεων.
- ✓ Στην ταξινόμηση ή την αναζήτηση σε ένα πίνακα να χρησιμοποιείς πάντα τη μέθοδο που είναι πιο κατάλληλη.

9.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Να γράψετε τις δηλώσεις των παρακάτω πινάκων, καθώς και τις εντολές με τις οποίες εκχωρούνται οι τιμές σε αυτά.

Α. Πίνακας 5 στοιχείων που κάθε στοιχείο έχει την τιμή του δείκτη του.

Β. Πίνακας που θα περιέχει τα ψηφία.

Γ. Πίνακας που περιέχει τα ονόματα των συμμαθητών σου.

Δ. Πίνακας με 10 στοιχεία, πρώτο στοιχείο τον αριθμό 500 και κάθε επόμενο στοιχείο να είναι το μισό του προηγούμενο, δηλαδή το δεύτερο 250, το τρίτο 125 κοκ.

ΔΤ2. Έχουμε δύο πίνακες, ο ένας με τα μοντέλα των υπολογιστών και ο δεύτερος με τις τιμές τους. Να γράψετε τις εντολές που βρίσκουν και τυπώνουν το φθηνότερο μοντέλο καθώς και το ακριβότερο.

ΔΤ3. Να γράψετε τις εντολές που δίνουν τις ακόλουθες τιμές σε ένα πίνακα ακεραίων Α.

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

ΔΤ4. Να γραφούν οι εντολές που ανταλλάσσουν τα στοιχεία της τρίτης και της έκτης στήλης σε ένα πίνακα ακεραίων 5Χ6.

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Να γράψετε ένα πρόγραμμα το οποίο να διαβάζει των αριθμό των τερμάτων που σημειώθηκαν στους αγώνες ποδοσφαίρου μίας αγωνιστικής της Α κατηγορίας (9 τιμές), να υπολογίζει τον μέσο αριθμό τερμάτων καθώς και το εύρος των τερμάτων (δηλαδή τη διαφορά της μεγαλύτερης από την μικρότερη τιμή).

ΔΕ2. Να γράψετε το πρόγραμμα του παραδείγματος 2 (επίπεδα θορύβου αυτοκινήτων) και να το εκτελέσετε για τις τιμές που δίνονται στον πίνακα του παραδείγματος. Το πρόγραμμα σας να τυπώνει τον πίνακα με τα επίπεδα θορύβου για κάθε μοντέλο.

ΔΕ3. Να γράψετε την άσκηση ΔΕ4 (ρύπανση ατμόσφαιρας) του προηγουμένου κεφαλαίου χρησιμοποιώντας πίνακες για την αποθήκευση των τιμών καθώς και των ονομάτων των σταθμών μέτρησης.

ΔΕ4. Να γράψετε πρόγραμμα το οποίο να ταξινομεί τα μοντέλα αυτοκινήτων του παραδείγματος 2, κατά αύξουσα σειρά του μέσου επιπέδου θορύβου κάθε μοντέλου.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Να συμπληρώσετε το παράδειγμα 1 (εισπράξεις αεροπορικών εταιρειών), ώστε να τυπώνει και αυτές που έχουν εισπράξεις κάτω από τον μέσο όρο, να βρίσκει και να τυπώνει την εταιρεία με τις λιγότερες και με τις περισσότερες εισπράξεις.

ΔΣ2.Να γραφεί πρόγραμμα το οποίο να δέχεται δύο τετραγωνικούς δισδιάστατους πίνακες και να υπολογίζει το άθροισμα και το γινόμενο τους.

Υπόδειξη: Αν a και b είναι οι αρχικοί πίνακες και c ο τελικός, τότε ισχύει:

Πρόσθεση: $c_{ij} = a_{ij} + b_{ij}$

Πολ/σμός: $c_{ij} = \sum_{k=1}^{n} a_{ik} . b_{kj}$

ΔΣ3. Να γραφεί πρόγραμμα που να υπολογίζει το άθροισμα των κυρίων διαγωνίων τετραγωνικού πίνακα ΝΧΝ.

ΔΣ4. Να γραφεί ένα πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό d και μία βάση μετατροπής b, όπου $2 \le b \le 16$ και να μετατρέπει τον αριθμό d σε σύστημα αρίθμησης με βάση b.

ΔΣ5. Δίνεται ένας πίνακας Α που περιέχει Ν τυχαίους ακεραίους αριθμούς. Να γραφεί πρόγραμμα το οποίο να διαβάζει έναν αριθμό και να ελέγχει αν ο αριθμός υπάρχει στον πίνακα. Για την αναζήτηση να χρησιμοποιηθεί ο αλγόριθμος της σειριακής αναζήτησης που παρουσιάστηκε στο κεφάλαιο 3.

ΔΣ6. Δίνονται οι πίνακες Σ1(Κ,Κ) και Π1(Κ,Κ) που περιέχουν τα αποτελέσματα των αγώνων ομίλου του EuroBasket. Ο πίνακας Σ1 περιέχει τα αποτελέσματα των αγώνων (Ν (νίκη) ή Η (ήττα)), ενώ ο πίνακας Π1 τη διαφορά πόντων για κάθε αγώνα.

Να γραφεί πρόγραμμα το οποίο θα βρίσκει και θα εκτυπώνει την τελική βαθμολογία του ομίλου. Σε περίπτωση ισοβαθμίας προηγείται η ομάδα που έχει την καλύτερη διαφορά πόντων από τις ισόβαθμές της.

Τα στοιχεία της κύριας διαγωνίου δεν περιέχουν καμία πληροφορία (καμία ομάδα δεν παίζει με τον εαυτό της!).

Ο πίνακας περιέχει στοιχεία μόνο κάτω ή πάνω από τη διαγώνιο του, είναι δηλαδή τριγωνικός (κάθε ομάδα παίζει μόνο μία φορά με κάθε αντίπαλο).

9.5. Τεστ αυτοαξιολόγησης

Συμπλήρωσε τα κενά με τη σωστή λέξη που λείπει

1. Οι πίνακες οι οποίοι έχουν τα στοιχεία τους σε μία στήλη ονομάζονται _____
2. Οι πίνακες είναι μία _____δομή δεδομένων.
3. Το αποτέλεσμα από τις παρακάτω εντολές είναι ο υπολογισμός του αθροίσματος _____ του πίνακα Α

```
Άθροισμα <- 0
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Ν
   Αθροισμα <- Άθροισμα+Α[Ι,Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

4. Οι πίνακες πρέπει να χρησιμοποιούνται πάντα όταν αυτό είναι δυνατό.
5. Η δήλωση των πινάκων που χρησιμοποιούνται σε ένα πρόγραμμα είναι υποχρεωτική.
6. Για την ταξινόμηση ενός πίνακα 100 στοιχείων μπορεί να χρησιμοποιηθεί μόνο μία μέθοδος.
7. Η χρήση των πινάκων σε ένα πρόγραμμα αυξάνει την απαιτούμενη μνήμη.

Διάλεξε ένα μεταξύ των προτεινόμενων

8. Ποιες από τις παρακάτω εντολές τυπώνουν όλα τα στοιχεία ενός δισδιάστατου πίνακα Π, 2Χ2

Α.
```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 2
   ΓΡΑΨΕ Π[Ι,Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Β.
```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 2
   ΓΡΑΨΕ Π[Ι]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Γ.
```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 2
   ΓΙΑ J ΑΠΟ 1 ΜΕΧΡΙ 2
      ΓΡΑΨΕ Π[Ι,J]
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Δ.
```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 2
   ΓΙΑ J ΑΠΟ 1 ΜΕΧΡΙ 2
      ΓΡΑΨΕ Π
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

9. Ποιο το αποτέλεσμα των παρακάτω εντολών στον πίνακα Α 8Χ10:

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 8
   Άθροισμα <- 0
   ΓΙΑ J ΑΠΟ 1 ΜΕΧΡΙ 10
      Άθροισμα <- Άθροισμα+Α[Ι,J]
   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
   ΜΟ <- Άθροισμα/10
   ΓΡΑΨΕ ΜΟ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Α. Γράφει το μέσο όρο των στοιχείων του πίνακα

Β. Γράφει τον μέσο όρο των στοιχείων κάθε γραμμής

Γ. Γράφει το μέσο όρο των στοιχείων κάθε στήλης

Δ. Γράφει τον μέσο όρο της τελευταίας γραμμής

10. Ποιο είναι το αποτέλεσμα των παρακάτω εντολών

```
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 10
   Α[Ι] <- 10+Ι
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΣΥΝ <- 0
ΓΙΑ Κ ΑΠΌ 1 ΜΕΧΡΙ 10 ΜΕ_ΒΗΜΑ 2
   ΣΥΝ <- ΣΥΝ+Α[Κ]
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ΣΥΝ
```

Α. 75 Β. 155 Γ. 50 Δ. 125

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

11. Τυπικές επεξεργασίες σε έναν πίνακα είναι:

Α. Ταξινόμηση

Β. Πρόσθεση στοιχείων

Γ. Πολλαπλασιασμός στοιχείων

Δ. Συγχώνευση

Ε. Αναζήτηση

12. Η επιλογή του καλύτερου αλγόριθμου ταξινόμησης εξαρτάται από

Α. Τον τύπο δεδομένων που έχει ο πίνακας

Β. Τη διάσταση του πίνακα

Γ. Το πλήθος των στοιχείων του πίνακα

Δ. Την αρχική διάταξη των στοιχείων

ΚΕΦΑΛΑΙΟ 10ο

ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

10.1. Πζοσδοκώμενα αποτελέσματα

Με το κεφάλαιο αυτό κλείνει η ενότητα του προγραμματισμού με χρήση δομημένης γλώσσας προγραμματισμού. Το κεφάλαιο ασχολείται με την έννοια τμηματικού προγραμματισμού, πως δηλαδή αναλύεται το πρόγραμμα σε υποπρογράμματα και τον τρόπο με τον οποίο η ΓΛΩΣΣΑ χειρίζεται τα **υποπρογράμματα**.

Για να αναλύσεις σωστά ένα σύνθετο πρόγραμμα σε υποπρογράμματα πρέπει αρχικά να αποφασίζεις πότε θα χρησιμοποιήσεις **συναρτήσεις** και **διαδικασίες**, να γνωρίζεις τη δομή καθώς και τα βασικά χαρακτηριστικά αυτών των υποπρογραμμάτων. Κάθε γλώσσα προγραμματισμού έχει ελαφρώς διαφορετικό τρόπο με τον οποίο αντιμετωπίζει τα υποπρογράμματα και ειδικά τον τρόπο με τον οποίο χειρίζεται τις παραμέτρους. Στο βιβλίο σου παρουσιάστηκαν θεωρητικά οι αρχές επικοινωνίας των υποπρογραμμάτων με τη χρήση των παραμέτρων, στο εργαστήριο θα γνωρίσεις το συγκεκριμένο τρόπο που το δικό σου προγραμματιστικό περιβάλλον υλοποιεί αυτές τις αρχές και πως χρησιμοποιεί τις παραμέτρους.

Τέλος σε αυτό το κεφάλαιο θα γνωρίσεις τον τρόπο που υλοποιείται η αναδρομή τα πλεονεκτήματα από τη σύνταξη αναδρομικών προγραμμάτων αλλά και τα μειονεκτήματα της σε σχέση με τα επαναληπτικά προγράμματα.

Οι λυμένες ασκήσεις του κεφαλαίου αυτού παρουσιάζονται στο περιβάλλον της ιδεατής γλώσσας προγραμματισμού ΓΛΩΣΣΑ και επίσης στα πραγματικά προγραμματιστικά περιβάλλοντα Basic και Pascal.

10.2. Επιπλέον παραδείγματα

Παράδειγμα 1

Πολλά από τα προγράμματα που αναπτύχθηκαν στα προηγούμενα κεφάλαια μπορούν να γραφούν καλύτερα με τη χρήση υποπρογραμμάτων. Εδώ θα δούμε το πρόγραμμα που υπολογίζει τα βασικά στατιστικά μεγέθη τη μέση τιμή, την τυπική απόκλιση και τη διάμεσο τιμή που παρουσιάστηκε στο βιβλίο σου στο κεφάλαιο 9.

Το πρόγραμμα χρησιμοποιεί τις εξής διαδικασίες και συναρτήσεις:

Υπολόγισε_ΜΟ_ΤυπΑπ :Υπολογίζει τη μέση τιμή και την τυπική απόκλιση ακεραίων αριθμών. Το τμήμα αυτό θα μπορούσε να υλοποιηθεί και με δύο συναρτήσεις, μία για τον υπολογισμό της μέσης τιμής και μίας δεύτερης για τον υπολογισμό της τυπικής απόκλισης.

Ταξινόμησε: Η διαδικασία αυτή ταξινομεί τα στοιχεία του πίνακα χρησιμοποιώντας μία παραλλαγή του αλγορίθμου που παρουσιάστηκε στο βιβλίο σου.

Υπολογισμός_Διαμέσου: Πραγματική συνάρτηση η οποία υπολογίζει τη διάμεσο τιμή.

```
ΠΡΟΓΡΑΜΜΑ Στατιστική
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ : Ι, Πλήθος, Στοιχεία[100], Μέγιστο, Διάμεσος,
                          Άθροισμα, Άθροισμα_2, Βοηθητική
    ΠΡΑΓΜΑΤΙΚΕΣ: ΜΟ, Τυπ_Απόκλιση

ΑΡΧΗ
    ! Εισαγωγή στοιχείων
    ΓΡΑΨΕ 'Δώσε το πλήθος των αριθμών (μέγιστο 100)'
    ΔΙΑΒΑΣΕ Πλήθος
    ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ Πλήθος
        ΓΡΑΨΕ 'Δώσε έναν αριθμό'
        ΔΙΑΒΑΣΕ Στοιχεία[Ι]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΚΑΛΕΣΕ Υπολόγισε_ΜΟ_ΤυπΑπ(Στοιχεία, Πλήθος, ΜΟ, Τυπ_Απόκλιση)
    ΚΑΛΕΣΕ Ταξινόμησε(Στοιχεία, Πλήθος)
    Διάμεσος <- Υπολογισμός_Διαμέσου(Στοιχεία)
    ! Εκτύπωση αποτελεσμάτων
    ΓΡΑΨΕ 'ΑΠΟΤΕΛΕΣΜΑΤΑ ΓΙΑ ΤΟΥΣ ', Πλήθος, 'ΑΡΙΘΜΟΥΣ'
    ΓΡΑΨΕ 'ΜΕΣΟΣ ΟΡΟΣ:', ΜΟ
    ΓΡΑΨΕ 'ΤΥΠΙΚΗ ΑΠΟΚΛΙΣΗ: ', Τυπ_Απόκλιση
    ΓΡΑΨΕ 'ΔΙΑΜΕΣΟΣ:', Διάμεσος
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Στατιστική

ΔΙΑΔΙΚΑΣΙΑ Υπολόγισε_ΜΟ_ΤυπΑπ(Πίνακας, Ν, ΜΟ, ΤυπΑποκλ)
! Υπολογισμός μέσου όρου
!Υπολογισμός τυπικής απόκλισης
```

```
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:Πίνακας[100],N, I, Άθροισμα, Άθροισμα_2
    ΠΡΑΓΜΑΤΙΚΕΣ:ΜΟ, ΤυπΑποκλ
ΑΡΧΗ
    Άθροισμα <- 0
    Άθροισμα_2 <- 0
    ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ N
        Άθροισμα <- Άθροισμα+ Πίνακας[I]
        Άθροισμα_2 <- Άθροισμα_2+ Πίνακας[I]^2
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΜΟ <- Άθροισμα/N
    ΤυπΑποκλ <- Τ_Ρ((N* Άθροισμα_2-Άθροισμα^2)/(N*(N-1)))
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Υπολόγισε_ΜΟ_ΤυπΑπ

ΔΙΑΔΙΚΑΣΙΑ Ταξινόμησε(Πίνακας, N)
!Ταξινόμηση των στοιχείων του πίνακα
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:I,N1,T,Βοηθητική
ΑΡΧΗ
    N1 <- N
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        T <- 0
        ΓΙΑ I ΑΠΟ 1 ΜΕΧΡΙ N1-1
            ΑΝ Πίνακας[I] > Πίνακας[I+1] ΤΟΤΕ
                Βοηθητική <- Πίνακας[I]
                Πίνακας[I] <- Πίνακας[I+1]
                Πίνακας[I+1] <- Βοηθητική
                T <- I
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        N1 <- T
    ΜΕΧΡΙΣ_ΟΤΟΥ T=0
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Ταξινόμησε


ΣΥΝΑΡΤΗΣΗ Υπολογισμός_Διαμέσου(A,N):ΑΚΕΡΑΙΑ
ΑΚΕΡΑΙΕΣ:A[100],N

ΑΡΧΗ
    ΑΝ N MOD 2 =0 ΤΟΤΕ
        Υπολογισμός_Διαμέσου <- (A[N/2]+A[N/2+1])/2
    ΑΛΛΙΩΣ
        Υπολογισμός_Διαμέσου <- A[(N+1)/2]
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ Υπολογισμός_Διαμέσου
```

Προγραμματιστικό περιβάλλον Pascal.

```
program Statistiki;

type list=array[1..100] of integer;
```

```
var
   avg, st_dev:real;
   i,median,n:integer;
   a:list;
{--------------------}
procedure sort(var a:list;n:integer);
{Ταξινόμηση των στοιχείων του πίνακα Α}
var
   i,n1,t,temp:integer;
begin
   n1:=n;
   repeat
      t:=0;
      for i:=1 to n1-1 do
         if a[i]>a[i+1] then
         begin
            temp:=a[i];
            a[i]:=a[i+1];
            a[i+1]:=temp;
            t:=i;
         end;
      n1:=t;
   until t=0;
end;
{--------------------}
procedure ave_stdev(a:list;n:integer;var average,stddv:real);
{υπολογισμός μέσου όρου και τυπικής απόκλισης}
var
   i:integer;
   sum, sum_2:real;

begin
   sum:=0; sum_2:=0;
   for i:=1 to n do
   begin
      sum:=sum+a[i]; sum_2:=sum_2+sqr(a[i]);
   end;
   average:=sum/n;
   stddv:=sqrt((n*sum_2-sqr(sum))/(n*(n-1)));
end;
{--------------------}
function med(a:list; n:integer):real;
{υπολογισμός της διαμέσου τιμής}
begin
   if n mod 2=0 then
      med:=(a[n div 2]+a[(n div 2)+1])/2
   else
      med:=a[(n+1) div 2];
   end;
```

```
begin {κυρίως πρόγραμμα}
   write('ΔΩΣΕ ΤΟΝ ΠΛΗΘΟΣ ΤΩΝ ΑΡΙΘΜΩΝ (<100):');
   readln (n);
   for i:=1 to n do
   begin
      write('ΔΩΣΕ ΤΟΝ ', i:3,'ο ΑΡΙΘΜΟ  :');readln (a[i]);
   end;
   ave_stdev (a,n,avg,st_dev);
   qsort(a,1,n);
   median:= med(a,n);
   writeln ('Μέση τιμή :', avg, 'Τυπική απόκλιση :',st_dev);
   writeln('Διάμεσος τιμή :',median);
end.
```

Προγραμματιστικό περιβάλλον Basic.

```
DECLARE FUNCTION Median! (x!(), n!)
DECLARE SUB Sort (k!(), n!)
DECLARE SUB MeanAndStdDev (z!(), m!, s!)
' Στατιστική
DIM x(100)
DATA 7
DATA 1,5,7,12,9,13,6
READ n
FOR i = 1 TO n: READ x(i): NEXT i
'
CLS
CALL MeanAndStdDev(x(), mx, sx)
CALL Sort(x(), n)
med = Median(x(), n)
'
CLS
PRINT "***   Αποτελέσματα   ***"
PRINT "————————————"
PRINT "mx="; mx, "sx="; sx
PRINT "median="; med
END

SUB MeanAndStdDev (z(), m, s)
' Υπολογισμός μέσης τιμής και τυπικής απόκλισης
s1 = 0: s2 = 0
n = UBOUND(z)
FOR i = 1 TO n
  s1 = s1 + z(i)
  s2 = s2 + z(i) * z(i)
NEXT i
m = s1 / n
s = SQR(s2 / n - m * m)
END SUB
```

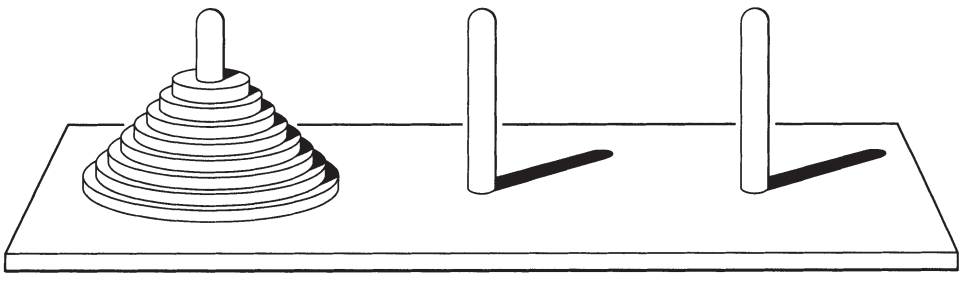
```
FUNCTION Median (x(), n)
IF n MOD 2 = 0 THEN
   Median = (x(n / 2) + x (n/2 + 1 ) /2
ELSE
   Median = x((n + 1) / 2)
END IF
END FUNCTION

SUB Sort (x(), n) STATIC
k = n
DO
  t = 0
  FOR i = 1 TO k - 1
    IF x(i) > x(i + 1) THEN
       SWAP x(i), x(i + 1)
       t = i
    END IF
  NEXT i
  k = t
LOOP UNTIL t = 0
END SUB
```

Παράδειγμα 2

Ένα χαρακτηριστικό πρόβλημα το οποίο λύνεται εύκολα με τη χρήση αναδρομής, ενώ είναι πολύ δύσκολο με επαναληπτική διαδικασία είναι οι πύργοι του **Ανόι**.

Στο πρόβλημα των πύργων του Ανόι υπάρχουν τρεις στύλοι και στον πρώτο από αυτούς βρίσκονται περασμένοι δίσκοι διαφορετικής διαμέτρου, έτσι ώστε οι διάμετροι των δίσκων να μικραίνουν από κάτω προς τα πάνω.

Όλοι οι δίσκοι, που βρίσκονται στον πρώτο στύλο, πρέπει να μεταφερθούν στο τρίτο ακολουθώντας τους εξής κανόνες:



- Όταν ένας δίσκος μεταφέρεται πρέπει να τοποθετηθεί σε έναν από τους τρεις στύλους.
- Μόνο ένας δίσκος μπορεί να μεταφερθεί κάθε φορά και πρέπει να βρίσκεται στην κορυφή του στύλου.
- Ένας μεγαλύτερος δίσκος δεν πρέπει να τοποθετηθεί πάνω από ένα μικρότερο.

Σύμφωνα με το μύθο το πρόβλημα δόθηκε στους μοναχούς του ιερού ναού του Μπενάρες. Στους μοναχούς δόθηκε μια χρυσή βάση με τρεις χρυσές βελόνες και εξήντα τέσσερις μικροί χρυσοί δίσκοι. Όταν οι μοναχοί κατορθώσουν να λύσουν το πρόβλημα, δηλαδή να μεταφέρουν τους εξήντα τέσσερις δίσκους από την πρώτη βελόνα στην τρίτη ακολουθώντας τους τρεις κανόνες που αναφέρθηκαν τότε θα έρθει η συντέλεια του Κόσμου (όπως θα δούμε στη συνέχεια δεν διατρέχουμε κανένα κίνδυνο).

Το παιγνίδι είναι σχετικά εύκολο να λυθεί για μικρό αριθμό δίσκων ,τρεις-τέσσερις, αλλά δυσκολεύει εξαιρετικά όσο ο αριθμός των δίσκων αυξάνεται.

Η γενική διατύπωση της λύσης όμως με χρήση αναδρομικής διαδικασίας είναι αρκετά απλή και περιγράφεται από τα παρακάτω βήματα:

- Αν υπάρχει μόνο ένας δίσκος τότε μεταφέρεται από τον Στύλο1 στο Στύλο3. Το πρόβλημα λοιπόν έχει λύση για Ν=1.
- Αν υπάρχουν δύο δίσκοι τότε χρειάζονται τρεις απλές κινήσεις:

  Ο πρώτος δίσκος από το Στυλο1 μεταφέρεται στο Στύλο2.

  Ο δεύτερος δίσκος από τον Στύλο2 μεταφέρεται στο Στύλο3.

  Ο δίσκος από το Στύλο2 μεταφέρεται στο Στύλο3.

Υποθέτουμε ότι η λύση υπάρχει για Ν-1 δίσκους, τότε για Ν δίσκους η λύση δίνεται αναδρομικά:

- Οι Ν-1 δίσκοι μεταφέρονται από τον Στύλο1 στο Στύλο2, χρησιμοποιώντας το Στύλο3 ως βοηθητικό.
- Ο τελευταίος δίσκος, που είναι ο τελευταίος άρα και ο μεγαλύτερος, μεταφέρεται από το Στύλο1 στο Στύλο3 .
- Οι Ν-1 δίσκοι μεταφέρονται από το Στύλο2 στο Στύλο3, χρησιμοποιώντας το Στύλο1 ως βοηθητικό.

Το πρόγραμμα που λύνει τους Πύργους του Ανόι είναι το ακόλουθο:

```
ΠΡΟΓΡΑΜΜΑ Πύργοι_του_Ανόι

ΣΤΘΕΡΕΣ
    Στύλος1='Α'
    Στύλος2='Β'
    Στύλος3='Γ'

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:N

ΑΡΧΗ
    ΓΡΑΨΕ 'δώσε τον αριθμό των δίσκων'
    ΔΙΑΒΑΣΕ N
    ΚΑΛΕΣΕ Μετακίνησε(N, Στύλος1, Στύλος2, Στύλος3)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ Μετακίνησε(N,ΣτύλοςΑ, ΣτύλοςΒ,ΣτύλοςΓ)

ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ:N
    ΧΑΡΑΚΤΗΡΕΣ: ΣτύλοςΑ, ΣτύλοςΒ,ΣτύλοςΓ
ΑΡΧΗ
    ΑΝ N=1 ΤΟΤΕ
        ΓΡΑΨΕ 'Μετακίνησε από τον', ΣτύλοςΑ, '  στον' , ΣτύλοςΓ
    ΑΛΛΙΩΣ
        ΚΑΛΕΣΕ Μετακινησε(N-1, ΣτύλοςΑ, ΣτύλοςΓ,ΣτύλοςΒ)
        ΓΡΑΨΕ 'Μετακίνησε από τον', ΣτύλοςΑ, '  στον' , ΣτύλοςΓ
        ΚΑΛΕΣΕ Μετακίνησε(N-1, ΣτύλοςΒ, ΣτύλοςΑ, ΣτύλοςΓ)
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Προγραμματιστικό περιβάλλον Pascal.

```
program anoi;
const
    st1='A';
    st2='B';
    st3='Γ';
var
    n:integer ;

procedure move(n:integer;sta,stb,stc:char);
begin
    if n=1 then
    begin
        writeln('ΜΕΤΑΚΙΝΗΣΕ ΑΠΟ ',sta,' ΣΤΟ ',stc)
    end
    else
    begin
        move(n-1,sta,stc,stb);
        writeln ('ΜΕΤΑΚΙΝΗΣΕ ΑΠΟ ', sta, ' ΣΤΟ ',stc);
```

```
        move (n-1,stb,sta,stc)
    end;
end;

begin

    write ('ΑΡΙΘΜΟΣ ΔΙΣΚΩΝ :'); readln(n);
    move (n,st1,st2,st3);
end.
```

Προγραμματιστικό περιβάλλον Basic.

```
DECLARE SUB Move (n!, a$, b$, c$)
' Anoi
INPUT "N=", n
CALL Move(n, "A", "B", "C")
END

SUB Move (n, a$, b$, c$)
IF n = 1 THEN
   PRINT "Move "; a$; " to "; c$
ELSE
   CALL Move(n - 1, a$, c$, b$)
   PRINT "Move "; a$; " to "; c$
   CALL Move(n - 1, b$, a$, c$)
END IF
END SUB
```

Η εκτέλεση του προγράμματος για 4 δίσκους δίνει τα εξής αποτελέσματα

Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Α στον Γ
Μετακίνησε από τον Β στον Γ
Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Γ στον Α
Μετακίνησε από τον Γ στον Β
Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Α στον Γ
Μετακίνησε από τον Β στον Γ
Μετακίνησε από τον Β στον Α
Μετακίνησε από τον Γ στον Α
Μετακίνησε από τον Β στον Γ
Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Α στον Γ
Μετακίνησε από τον Β στον Γ

Η λύση που δόθηκε θεωρητικά επιλύει το πρόβλημα για οποιοδήποτε αριθμό δίσκων. Ας μελετήσουμε όμως τον αριθμό των κινήσεων που απαιτείται.

Όπως είδαμε χρειάζονται 15 κινήσεις για την επίλυση του προβλήματος με 4 δίσκους. Για 5 δίσκους οι κινήσεις που χρειάζονται είναι 31.

Γενικά για Ν δίσκους η λύση δίνεται μετά από 2^N-1 κινήσεις. Αυτό σημαίνει ότι για 10 δίσκους χρειάζονται 1023 κινήσεις και για 20 δίσκους οι κινήσεις ξεπερνούν το εκατομμύριο.

Για τους 64 δίσκους που απασχολεί τους μοναχούς χρειάζονται $1,845 \times 10^{19}$, δηλαδή ένας αριθμός με 20 ψηφία. Πόσο μεγάλος είναι ένας τέτοιος αριθμός; Αν υποθέσουμε ότι εκτελούμε χίλιες κινήσεις το δευτερόλεπτο (όσες περίπου μπορεί να εκτελέσει ένας γρήγορος υπολογιστής), τότε χρειάζονται περίπου μισό εκατομμύριο χρόνια!!!

Η επίλυση του προβλήματος του πύργου του Ανόι αποτελεί χαρακτηριστική περίπτωση αλγορίθμων εκθετικής πολυπλοκότητας ($O(2^N)$).

Οι αλγόριθμοι της μορφής αυτής όπως αναφέρθηκε στο βιβλίο σου στο κεφάλαιο 5 "ανάλυση αλγορίθμων" είναι ουσιαστικά ακατάλληλοι για την πρακτική επίλυση προβλημάτων και χρήσιμοι μόνο για θεωρητικά προβλήματα, αφού κάθε αύξηση του Ν κατά μία μονάδα διπλασιάζει τον απαιτούμενο χρόνο εκτέλεσης.

10.3. Συμβουλές - υποδείξεις

Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και τις δικές της αρχές για τη σύνταξη και χρήση των υποπρογραμμάτων και πρέπει να μελετήσεις προσεκτικά πως υλοποιούνται τα υποπρογράμματα στο προγραμματιστικό περιβάλλον που χρησιμοποιείς. Υπάρχουν όμως κάποιοι γενικοί κανόνες που πρέπει να ακολουθείς.

- Πριν ξεκινήσεις να γράφεις το πρόγραμμα σου να μελετήσεις πώς το πρόγραμμα μπορεί να αναλυθεί σε επί μέρους τμήματα και να αποφασίσεις για τα αντίστοιχα υποπρογράμματα. Χρήσιμο είναι να κάνεις ένα διάγραμμα που θα δείχνει την ιεραρχία ανάμεσα στα υποπρογράμματα. Τότε να αναπτύσσεις τους αλγόριθμους για το κάθε υποπρόγραμμα και στη συνέχεια να γράφεις το πρόγραμμα.
- Να μελετάς αν ένα υποπρόγραμμα πρέπει να υλοποιηθεί με διαδικασία ή μπορεί να υλοποιηθεί με συνάρτηση.
- Εξέτασε αν κάποια υποπρογράμματα τα οποία έχεις ήδη γράψεις ή υπάρχουν σε έτοιμες βιβλιοθήκες προγραμμάτων μπορούν να χρησιμοποιηθούν. Θα γλιτώσεις χρόνο και κόπο.
- Κάθε υποπρόγραμμα να προσπαθείς να είναι όσο το δυνατόν πιο ανεξάρτητο από τα άλλα. Αυτό σε προφυλάσσει από λάθη στο πρόγραμμα σου και σου επιτρέπει τη χρήση του σε άλλα προγράμματα αργότερα.

Για να αποφύγεις τα πλέον κοινά λάθη να προσέχεις ιδιαίτερα:

- να ορίζεις τον τύπο της συνάρτησης. Οι συναρτήσεις παράγουν μόνο ένα αποτέλεσμα συγκεκριμένου τύπου, ακεραίου, πραγματικού κλπ που πρέπει να ορίζεται.

- Να μην υπάρχουν λάθη στην αντιστοίχιση τυπικών και πραγματικών παραμέτρων. Πρόσεξε ότι οι λίστες πρέπει να περιέχουν το ίδιο αριθμό παραμέτρων και κάθε τυπική παράμετρος με την αντίστοιχη πραγματική πρέπει να είναι του ιδίου τύπου.

10.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Τι είδους υποπρόγραμμα, διαδικασία ή συνάρτηση, πρέπει να χρησιμοποιήσεις για τα παρακάτω

Α) Εισαγωγή τριών δεδομένων.

Β) Εισαγωγή ενός δεδομένου.

Γ) Υπολογισμός του μικρότερου από πέντε ακεραίους.

Δ) Υπολογισμός των δύο μικροτέρων από πέντε ακεραίους.

Ε) Έλεγχος αν δύο αριθμοί είναι ίσοι.

Ζ) Να ταξινομεί, και να επιστρέφει ταξινομημένους, πέντε αριθμούς.

Η) Έλεγχος αν ένας χαρακτήρας είναι φωνήεν ή σύμφωνο.

ΔΤ2. Να γράψεις τα υποπρόγραμματα που υλοποιούν τα παρακάτω:

Α) Να διαβάζει ένα αριθμό και να επιστρέφει το τετράγωνο του.

Β) Να δέχεται δύο αριθμούς και να επιστρέφει το μικρότερο από δύο αριθμούς.

Γ) Να δέχεται την τιμή ενός προϊόντος και να υπολογίζει και να τυπώνει την αξία του ΦΠΑ .

Δ) Να ελέγχει αν ένας αριθμός είναι άρτιος.

ΔΤ3. Να σημειώσεις, στο τετράδιο σου, όλα τα βήματα για τον υπολογισμό του 4!, τόσο με τη χρήση επαναληπτικής διαδικασίας όσο και με τη χρήση αναδρομικής, σύμφωνα με τα προγράμματα που δίνονται στο βιβλίο σου.

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Να γράψεις πρόγραμμα το οποίο θα διαβάζει δύο αριθμούς, θα υπολογίζει το Μέγιστο Κοινό Διαιρέτη (ΜΚΔ) και το Ελάχιστο Κοινό Πολλαπλάσιο (ΕΚΠ) και τέλος θα τυπώνει τα αποτελέσματα.

Υπόδειξη. Για δύο αριθμούς x,y ισχύει: $x \cdot y = ΜΚΔ(x, y) \cdot ΕΚΠ(x, y)$

ΔΕ2. Να εκτελέσεις το πρόγραμμα του παραδείγματος 1.

ΔΕ3. Να γράψεις πρόγραμμα το οποίο να εκτελεί τις τέσσερεις πράξεις σε μιγαδικούς αριθμούς.

Για τους μιγαδικούς αριθμούς α+βi και γ+δi έχουμε

$(\alpha+\beta i)+(\gamma+\delta i)=(\alpha+\gamma)+(\beta+\delta)i$

$(\alpha+\beta i)-(\gamma+\delta i)=(\alpha-\gamma)+(\beta-\delta)i$

$(\alpha+\beta i)\cdot(\gamma+\delta i)=(\alpha\gamma-\beta\delta)+(\alpha\delta+\beta\gamma)i$

$$\frac{\alpha+\beta i}{\gamma+\delta i}=\left(\frac{\alpha\gamma+\beta\delta}{\gamma^2+\delta^2}\right)+\left(\frac{\beta\gamma-\alpha\delta}{\gamma^2+\delta^2}\right)i$$

Το πρόγραμμα θα οδηγείται από μενού επιλογής όπου ο χρήστης θα επιλέγει το είδος της πράξης.

Στην περίπτωση της διαίρεσης το γ και το δ πρέπει να είναι διάφορα του 0.

ΔΕ4. Να ξαναγράψεις το πρόγραμμα της ΔΕ1 χρησιμοποιώντας αναδρομικές συναρτήσεις.

Σύγκρινε τα δύο προγράμματα.

Στο σπίτι

ΔΣ1. Να γράψεις ένα πρόγραμμα το οποίο διαβάζει τη τιμή βιβλίων σε ΕΥΡΩ και μετατρέπει τις τιμές τους σε Δραχμές, Γερμανικά Μάρκα, Γαλλικά Φράγκα και Ιταλικές λιρέτες. Να χρησιμοποιήσεις για τις μετατροπές τις τρέχουσες ισοτιμίες των νομισμάτων.

ΔΣ2. Να ξαναγράψεις την άσκηση ΔΣ6 του κεφαλαίου 9, τα αποτελέσματα των αγώνων ομίλου του EuroBasket, χρησιμοποιώντας διαδικασίες και συναρτήσεις.

ΔΣ3. Να επεκτείνεις το παράδειγμα 1, ώστε να υπολογίζει την επικρατούσα τιμή, δηλαδή την τιμή που εμφανίζεται περισσότερες φορές.

ΔΣ4. Να γράψεις το πρόγραμμα ΔΕ5 που υπολογίζει τη συνολική χωρητικότητα πυκνωτών και τη συνολική αντίσταση αντιστάσεων με τη χρήση υποπρογραμμάτων.

ΔΣ5. Να γραφεί πρόγραμμα το οποίο να προσθέτει δύο κλάσματα. Το πρόγραμμα δέχεται τέσσερις ακεραίους αριθμούς τους παρανομαστές και τους αριθμητές των δύο κλασμάτων υπολογίζει και εκτυπώνει τον αριθμητή και τον παρανομαστή του αποτελέσματος.

Α/Β + Γ/Δ = Ε/Ζ

Υπόδειξη

Ενώ το πρόβλημα αρχικά φαίνεται απλό, η υλοποίησή του είναι αρκετά πολύπλοκη. Αρχικά πρέπει να απλοποιηθούν τα κλάσματα, στη συνέχεια να γίνουν ομώνυμα, να προστεθούν οι αριθμητές και τέλος να απλοποιηθεί το αποτέλεσμα.

Οι διαδικασίες αυτές απαιτούν τον υπολογισμό του ΜΚΔ (για την απλοποίηση) και του ΕΚΠ για τη μετατροπή των κλασμάτων σε ομώνυμα. Να χρησιμοποιήσετε τις συναρτήσεις της άσκησης ΔΕ1.

ΔΣ6. Δίνεται η εξίσωση $e^x - 2x - 1 = 0$. Να γραφεί πρόγραμμα το οποίο να βρίσκει μια ρίζα της εξίσωσης αυτής στο διάστημα [1,2] με τη μέθοδο της διχοτόμησης, όπως περιγράφηκε στην παράγραφο 4.3 του βιβλίου σου.

Τεστ αυτοαξιολόγησης

Συμπλήρωσε τα κενά με τη σωστή λέξη που λείπει

1. Η λίστα των παραμέτρων που υπάρχει στη δήλωση μίας διαδικασίας ονομάζεται λίστα _____παραμέτρων.
2. Τα δύο είδη υποπρογραμμάτων είναι οι _____ και οι _____
3. Οι μεταβλητές οι οποίες ισχύουν σε όλα τα υποπρογράμματα ενός προγράμματος και όχι μόνο σε αυτό που ορίστηκε λέγεται ότι έχουν _____εμβέλεια.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

4. Μια διαδικασία και μια συνάρτηση μπορούν να εκτελούν ακριβώς τις ίδιες λειτουργίες.
5. Το πλήθος των τυπικών και των πραγματικών παραμέτρων πρέπει να είναι ίδιο.
6. Οι αναδρομικές διαδικασίες είναι πάντοτε προτιμότερες από τις αντίστοιχες επαναληπτικές.
7. Η ενεργοποίηση μίας συνάρτησης γίνεται με την εντολή ΚΑΛΕΣΕ.

Διάλεξε ένα μεταξύ των προτεινόμενων

8. Όταν μία μεταβλητή ισχύει μόνο στο υποπρόγραμμα που ορίστηκε ονομάζεται

   Α. Τοπική Β. Καθολική Γ. Παράμετρος Δ. Τυπική

9. Τι θα τυπώσουν οι παρακάτω εντολές

```
.....
A <- 5
B <- 10
```

```
Γ <- 0
ΚΑΛΕΣΕ Διαδ1(Α, Β)
ΓΡΑΨΕ Α,Β,Γ
……
ΔΙΑΔΙΚΑΣΙΑ Διαδ1(Γ,Δ)
……
ΑΡΧΗ
Γ <- Γ-Δ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Διαδ1
```

Α. 5,10,0 Β. 5,10, -5 Γ. -5,10,0 Δ. -5,10,-5

10. Τι θα τυπώσουν οι παρακάτω εντολές

```
……
Α <- 5
Β <- 10
ΚΑΛΕΣΕ Διαδ1(Β, Α)
ΓΡΑΨΕ Α,Β
……
ΔΙΑΔΙΚΑΣΙΑ Διαδ1(Α,Β)
……
ΑΡΧΗ
ΓΡΑΨΕ Α,Β
Α <- Α-Β
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Διαδ1
```

Α. 5,10 Β. 10,5 Γ. 5,10 Δ. 10, 5

5,10 5,5 -5,10 5,10

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

11. Ο ορισμός κάθε αναδρομικού υποπρογράμματος περιλαμβάνει

Α. Αναδρομική σχέση.

Β. Τιμή βάσης.

Γ. Υπολογισμό παραγοντικού.

Δ. Επαναληπτική διαδικασία.

Ε. Καθολικές μεταβλητές.

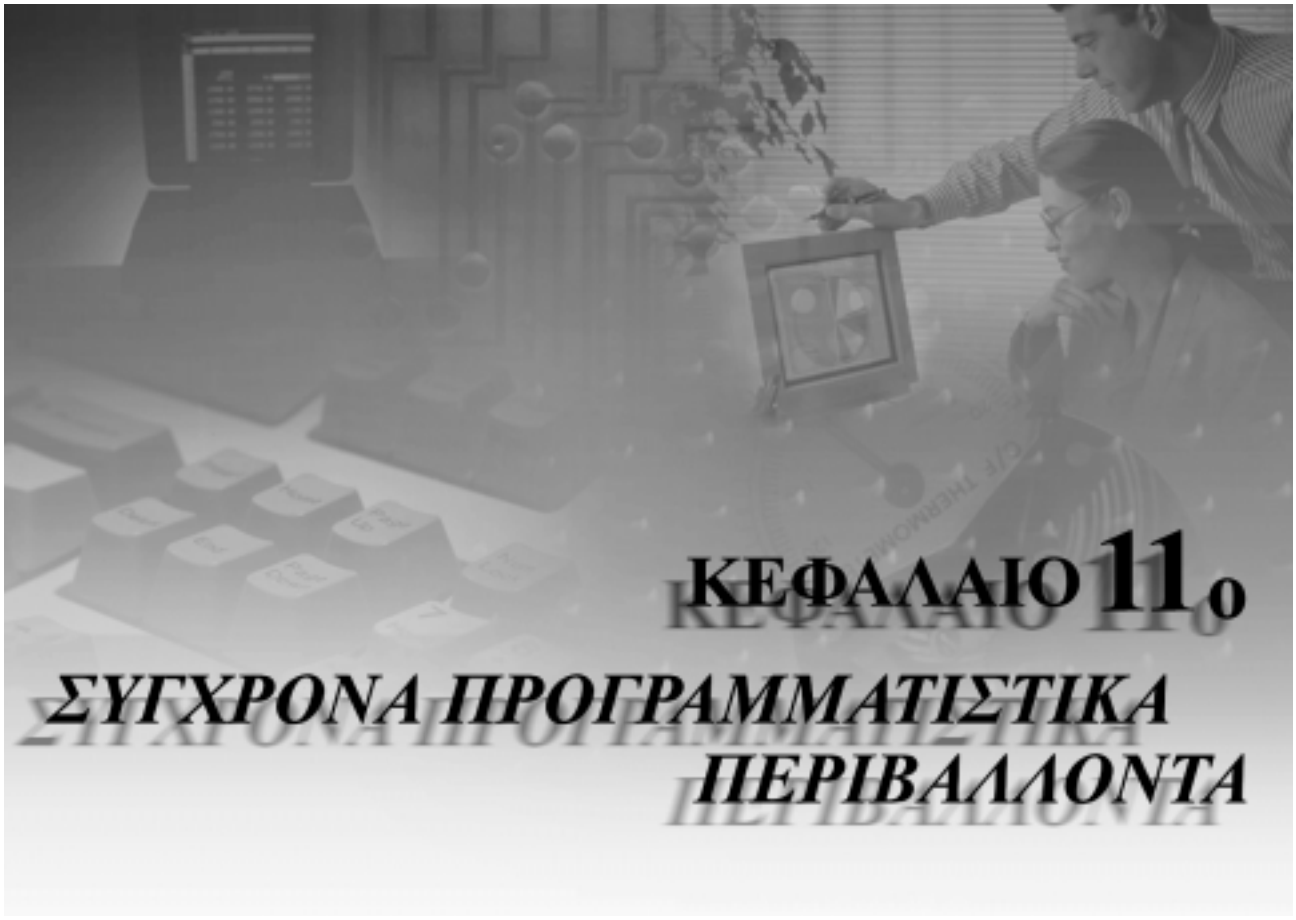
12. Μερικά από τα πλεονεκτήματα του τμηματικού προγραμματισμού είναι

Α. Λιγότερος χρόνος για την ανάπτυξη του προγράμματος.

Β. Ευκολότερη διόρθωση.

Γ. Ταχύτητα κατά την εκτέλεση.

Δ. Χρήση αναδρομικών διαδικασιών.



ΚΕΦΑΛΑΙΟ 11ο
ΣΥΓΧΡΟΝΑ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

11.1. Προσδοκώμενα αποτελέσματα

Μέχρι τώρα είχες εργαστεί σε ένα παραδοσιακό περιβάλλον προγραμματισμού, που κυρίως ενδιαφέρεται για το πρόγραμμα και όχι για το περιβάλλον εργασίας του χρήστη. Σε αυτό το κεφάλαιο έρχεσαι σε επαφή με σύγχρονα γραφικά περιβάλλοντα προγραμματισμού που παρέχουν ιδιαίτερες δυνατότητες επικοινωνίας του προγράμματος με το χρήστη αλλά και ειδικά εργαλεία προς το προγραμματιστή για την απλούστευση του προγραμματισμού.

Οι βασικές έννοιες προγραμματισμού που ήδη γνωρίζεις και χρησιμοποιείς, εξακολουθούν να εφαρμόζονται και σε ένα σύγχρονο προγραμματιστικό περιβάλλον, σε συνδυασμό όμως με τις τεχνικές του αντικειμενοστραφή και του οδηγούμενου από τα γεγονότα προγραμματισμού. Ο συνδυασμός αυτός έχει σαν στόχο την εκμετάλλευση όλων των δυνατοτήτων ενός σύγχρονου περιβάλλοντος με την προσομοίωση του χώρου εργασίας με τον πραγματικό φυσικό μας κόσμο, μέσα από την παραδοχή ότι κάθε γραφικό εργαλείο της διασύνδεσης του προγράμματος με το χρήστη αποτελεί ένα ανεξάρτητο αντικείμενο. Μέσα από τα παραδείγματα του κεφαλαίου, αναλύεται ο νέος τρόπος αντίληψης των εφαρμογών και επιδεικνύεται ο συνδυασμός των τεχνικών προγραμματισμού.

Σαν προγραμματιστικό περιβάλλον, για την παρουσίαση των λυμένων ασκήσεων, έχουμε επιλέξει τη Visual Basic for Windows, γιατί είναι απλό, εύχρηστο και δημοφιλές και κυρίως γιατί οι εντολές κώδικα είναι σχεδόν πανομοιότυπες με τη γλώσσα προγραμματισμού Basic που ήδη έχουμε χρησιμοποιήσει σε προηγούμενα παραδείγματα του τετραδίου. Εναλλακτικά, σε κάθε παράδειγμα συμπεριλαμβάνουμε και τον κώδι-

κα σε περιβάλλον Delphi. Στην υλοποίηση της διεπαφής χρήστη με το Delphi δεν αναφερόμαστε καθόλου, γιατί η φιλοσοφία και οι τεχνικές δημιουργίας του δεν διαφοροποιούνται από τις αντίστοιχες της Visual Basic.

11.2. Επιπλέον παραδείγματα

Παράδειγμα 1

Στο βιβλίο σου έχουμε ήδη παρουσιάσει το πρώτο παράδειγμα εφαρμογής σε υποθετικό περιβάλλον προγραμματισμού. Το παράδειγμα αναφέρεται σε μια παρουσίαση των τριών βημάτων ανάπτυξης μιας εφαρμογής σε σύγχρονο προγραμματιστικό περιβάλλον. Στο σημείο αυτό θεωρούμε ότι πρέπει να επιδείξουμε την υλοποίησή του σε πραγματικό περιβάλλον προγραμματισμού.

Περιβάλλον προγραμματισμού Visual Basic

Το πρώτο στοιχείο που πρέπει να σε απασχολήσει, είναι ο τρόπος επικοινωνίας του χρήστη με την εφαρμογή. Για το σχεδιασμό λοιπόν της διεπαφής χρήστη χρησιμοποιήσαμε τα παρακάτω γραφικά αντικείμενα:

- μια **φόρμα** (form) η οποία θα αποτελέσει το παράθυρο μέσα στο οποίο θα εκτελείται η εφαρμογή,
- τέσσερα **πλήκτρα εντολής** (command buttons) με τα οποία ο χρήστης θα καθοδηγεί και θα τερματίζει την εκτέλεσή της,
- τέσσερις **ετικέτες** (labels) που θα εμφανίζουν τις πληροφορίες,
- μια **γραμμή** (line) για να δώσουμε έμφαση στην ετικέτα με τον τίτλο της εφαρμογής.

Μόλις ξεκινήσουμε τη δημιουργία μιας εφαρμογής, η Visual Basic έχει ήδη δημιουργήσει μια φόρμα με το όνομα Form1. Επάνω σε αυτή τη φόρμα τοποθετούμε τα πλήκτρα εντολής, τις ετικέτες και το αντικείμενο γραμμή. Τα γραφικά αντικείμενα τα επιλέγουμε από την εργαλειοθήκη (toolbox) που μας παρέχει το περιβάλλον προγραμματισμού της Visual Basic.

Στις ιδιότητες των αντικειμένων της εφαρμογής αποδίδουμε τις επόμενες τιμές:

Αντικείμενο	Ιδιότητα	Τιμή
Φόρμα	Name	FrmMain
	Caption	Η πρώτη μας εφαρμογή σε σύγχρονο προγραμματιστικό περιβάλλον
Πλήκτρο εντολής 1	Name	CmdStep1
	Caption	1
Πλήκτρο εντολής 2	Name	CmdStep2
	Caption	2
Πλήκτρο εντολής 3	Name	CmdStep3
	Caption	3
Πλήκτρο εντολής 4	Name	CmdEnd
	Caption	(κενό)
	Style	1-Graphical
	Picture	\vb\graphics\icons\traffic\Trffc14.ico
Ετικέτα 0	Name	LblTitle
	Visible	True
	Caption	Βήματα ανάπτυξης μιας εφαρμογής
	ForeColor	&H000000C0&
Ετικέτα 1	Name	LblStep1
	Visible	False
	Caption	Σχεδιασμός του τρόπου επικοινωνίας χρήστη-εφαρμογής, επιλέγοντας τα κατάλληλα αντικείμενα
	ForeColor	&H8000000D& (Highlight)
Ετικέτα 2	Name	LblStep2
	Visible	False
	Caption	Καθορισμός της αρχικής συμπεριφοράς των αντικειμένων μέσω των ιδιοτήτων που τα χαρακτηρίζουν
	ForeColor	&H8000000D& (Highlight)
Ετικέτα 3	Name	LblStep3
	Visible	False
	Caption	Δημιουργία και εκσφαλμάτωση του κώδικα
	ForeColor	&H8000000D& (Highlight)
Γραμμή	Name	Lntitle
	ForeColor	&H000000C0&

Το τελευταίο βήμα που πρέπει να εκτελέσουμε για να ολοκληρωθεί ο σχεδιασμός της εφαρμογής, είναι η προσθήκη των εντολών κώδικα.

Στη Visual Basic και στο Delphi, για να μετατρέψουμε την τιμή μιας ιδιότητας ενός αντικειμένου χρησιμοποιούμε εντολές της μορφής :

Αντικείμενο.Ιδιότητα = Τιμή

Για παράδειγμα αν θέλουμε να εμφανίσουμε μέσα σε ένα αντικείμενο πλαίσιο κειμένου, με όνομα Text1, το κείμενο "Καλώς ήλθατε στο κόσμο των αντικειμένων" πρέπει να χρησιμοποιήσουμε μια εντολή της μορφής :

Text1.Text= "Καλώς ήλθατε στο κόσμο των αντικειμένων"

Για την εκτέλεση μιας μεθόδου χρησιμοποιούμε εντολές της μορφής :

Αντικείμενο.Μέθοδος

Ο καθαρισμός των περιεχομένων ενός αντικειμένου λίστας με όνομα List1, πραγματοποιείται με τη μέθοδο Clear:

List1.Clear

Στην εντολή εκτέλεσης των περισσοτέρων μεθόδων είναι δυνατό να συμπεριλάβουμε και ορίσματα (arguments) με τα οποία επιτυγχάνουμε παραμετροποίηση του αποτελέσματος. Σε αυτή τη περίπτωση η εντολή εκτέλεσης μιας μεθόδου συντάσσεται ως εξής :

Αντικείμενο.Μέθοδος Λίστα ορισμάτων

Όταν θέλουμε να μετακινήσουμε επάνω σε μία φόρμα ένα αντικείμενο εικόνα με όνομα Image1 θα χρησιμοποιήσουμε την μέθοδο Move ως εξής:

Image1.Move 100, 200

Το αποτέλεσμα της εντολής, θα είναι η μετακίνηση του αντικειμένου Image1 στο σημείο της φόρμας με απόλυτες συντεταγμένες (100, 200).

Μέσα από το παράθυρο κώδικα που παρέχει η Visual Basic πληκτρολογούμε τις παρακάτω εντολές στις διαδικασίες γεγονότων:

```
Private Sub CmdStep1_Click()
' Πλήκτρο εμφάνισης πρώτου βήματος
' Με την ιδιότητα Visible εμφανίζουμε ή αποκρύπτουμε από την οθόνη
' ένα γραφικό αντικείμενο
' Οι δυνατές τιμές της ιδιότητας Visible είναι αντίστοιχα True ή
' False.
 LblStep1.Visible = True
    LblStep2.Visible = False
    LblStep3.Visible = False
End Sub
Private Sub CmdStep2_Click()
    ' Πλήκτρο εμφάνισης δεύτερου βήματος
```

```
    LblStep1.Visible = False
    LblStep2.Visible = True
    LblStep3.Visible = False
End Sub
Private Sub CmdStep3_Click()
    ' Πλήκτρο εμφάνισης τρίτου βήματος
    LblStep1.Visible = False
    LblStep2.Visible = False
    LblStep3.Visible = True
End Sub
Private Sub CmdEnd_Click()
    ' Πλήκτρο τερματισμού
    End
End Sub
```

Περιβάλλον προγραμματισμού Delphi

```
program Prj1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {LblStep1};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TLblStep1, LblStep1);
  Application.Run;
end.
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
                                                          Dialogs,
  StdCtrls, Buttons;
type
  TLblStep1 = class(TForm)
    Title1: TLabel;
    CmdStep1: TButton;
    CmdStep2: TButton;
    CmdStep3: TButton;
    LblStep1: TLabel;
    LblStep2: TLabel;
    LblStep3: TLabel;
    CmdEnd: TBitBtn;
    procedure CmdStep1Click(Sender: TObject);
    procedure CmdStep2Click(Sender: TObject);
    procedure CmdStep3Click(Sender: TObject);
    procedure CmdEndClick(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;
var
  LblStep1: TLblStep1;
implementation
{$R *.DFM}
procedure TLblStep1.CmdStep1Click(Sender: TObject);
begin
     LblStep1.Visible := True;
     LblStep2.Visible := False;
     LblStep3.Visible := False;
end;
procedure TLblStep1.CmdStep2Click(Sender: TObject);
begin
     LblStep1.Visible := False;
     LblStep2.Visible := True;
     LblStep3.Visible := False;
end;
procedure TLblStep1.CmdStep3Click(Sender: TObject);
begin
     LblStep1.Visible := False;
     LblStep2.Visible := False;
     LblStep3.Visible := True;
end;
procedure TLblStep1.CmdEndClick(Sender: TObject);
begin
     close;
end;
end.
```

Παράδειγμα 2

Να κατασκευαστεί μια οθόνη εισαγωγής κωδικού πρόσβασης (password). Κατά την εισαγωγή του κωδικού στην οθόνη του υπολογιστή, για λόγους ασφάλειας, δεν πρέπει να εμφανίζονται οι χαρακτήρες που πληκτρολογεί ο χρήστης αλλά ο χαρακτήρας *. Όταν ο χρήστης πληκτρολογεί τον κωδικό να εμφανίζεται αντίστοιχα μήνυμα αποδοχής ή απόρριψης.

Περιβάλλον προγραμματισμού Visual Basic

Για την υλοποίηση της εφαρμογής προτείνουμε να χρησιμοποιηθούν τα παρακάτω αντικείμενα:

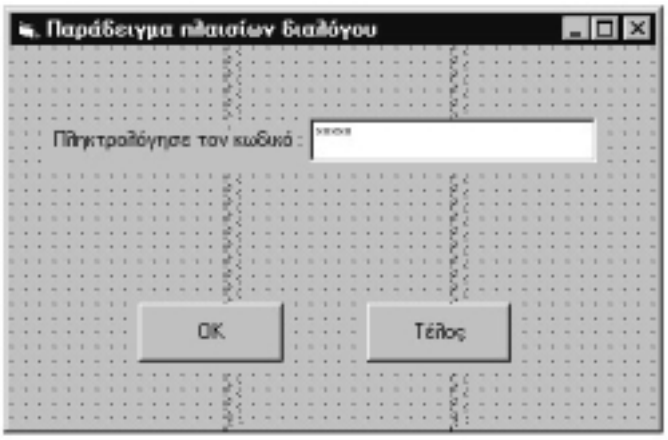
- μια **φόρμα** η οποία είναι το παράθυρο μέσα στο οποίο θα εκτελείται η εφαρμογή,
- ένα **πλαίσιο κειμένου** (text box) που θα εισάγετε ο κωδικός,
- μια **ετικέτα** (label), για την προτροπή του χρήστη ώστε να πληκτρολογήσει τον κωδικό,

- δύο **πλήκτρα εντολής** που θα χρησιμεύουν για την αποδοχή του κωδικού και τον τερματισμό της εφαρμογής.

Στις ιδιότητες των αντικειμένων της εφαρμογής αποδίδουμε τις παρακάτω τιμές:

Αντικείμενο	Ιδιότητα	Τιμή
Φόρμα	Name	Frmpwd
	Caption	Παράδειγμα πλαισίων διαλόγου
Ετικέτα 1	Name	Lblpwd
	Caption	Πληκτρολογήστε τον κωδικό :
Πλαίσιο κειμένου	Name	Txtpwd
	Passwordchar	*
	Text	(κενό)
Πλήκτρο εντολής 1	Name	CmdOK
	Caption	OK
Πλήκτρο εντολής 2	Name	CmdEnd
	Caption	Τέλος

Μετά την τοποθέτηση των αντικειμένων και την αντιστοίχιση των ιδιοτήτων έχουμε ολοκληρώσει την οθόνη εισαγωγής κωδικού (Σχήμα 11.1).



***Σχ.11.1.** Η οθόνη της εφαρμογής εισαγωγής κωδικού.*

Στο παράδειγμά μας θα χρησιμοποιήσουμε ως κωδικό την τρέχουσα ημερομηνία. Η ημερομηνία θα διαβάζεται από τον υπολογιστή με την συνάρτηση Date.

Τα μηνύματα προς το χρήστη θα γίνονται με τη χρήση προκαθορισμένων πλαισίων διαλόγου που προσφέρει η Visual Basic. Η εμφάνιση των προκαθορισμένων πλαισίων διαλόγου γίνεται με την συνάρτηση MsgBox, η οποία συντάσσεται ως εξής:

MsgBox(μήνυμα[, πλήκτρα εντολής] [, τίτλος πλαισίου] [, αρχείο βοήθειας, δείκτης αρχείου])

Τα περισσότερα ορίσματα της συνάρτησης MsgBox, όπως γίνεται αντιληπτό, είναι μη υποχρεωτικά και μπορούμε να τα παραλείψουμε. Όταν παραλείψουμε κάποιο όρισμα η Visual Basic χρησιμοποιεί τα δικά της εξ ορισμού ορίσματα, για παράδειγμα στον τίτλο πλαισίου τοποθετεί το όνομα της εφαρμογής.

Σε ένα προκαθορισμένο πλαίσιο διαλόγου έχουμε τη δυνατότητα να εμφανίσουμε πλήκτρα εντολής. Μόλις ο χρήστης πατήσει κάποιο από τα πλήκτρα εντολής του πλαισίου διαλόγου, αυτό φεύγει από την οθόνη και επιστρέφει στο πρόγραμμα μια τιμή που εκφράζει το πλήκτρο που επιλέχτηκε. Το όρισμα **πλήκτρα εντολής,** βασικά καθορίζει τον αριθμό και τον τύπο των πλήκτρων καθώς και το εικονίδιο που θα εμφανιστούν στο πλαίσιο διαλόγου. Οι τιμές του συνήθως ορίζονται με τη χρήση σταθερών και οι πιο βασικές είναι :

Σταθερά	Τιμή	Περιγραφή
vbOKOnly	0	Εμφάνιση OK πλήκτρου.
vbOKCancel	1	Εμφάνιση OK και Cancel πλήκτρων.
vbAbortRetryIgnore	2	Εμφάνιση Abort, Retry, και Ignore πλήκτρων.
vbYesNoCancel	3	Εμφάνιση Yes, No, και Cancel πλήκτρων.
vbYesNo	4	Εμφάνιση Yes και No πλήκτρων.
vbRetryCancel	5	Εμφάνιση Retry και Cancel πλήκτρων.
vbCritical	16	Εμφάνιση εικονίδιου κρίσιμης εργασίας.
vbQuestion	32	Εμφάνιση εικονίδιου ερώτησης.
vbExclamation	48	Εμφάνιση εικονίδιου προειδοποίησης.
vbInformation	64	Εμφάνιση εικονίδιου πληροφοριών.

Ένα όρισμα **πλήκτρα εντολής** μπορεί να οριστεί και με το άθροισμα δύο σταθερών. Για παράδειγμα δηλώνοντας στο όρισμα την τιμή vbOKCancel + vbExclamation θα εμφανιστούν στο πλαίσιο διαλόγου τα πλήκτρα OK και Cancel καθώς και το εικονίδιο προειδοποίησης.

Όταν ο χρήστης επιλέξει κάποιο από τα πλήκτρα του πλαισίου διαλόγου που εμφανίζεται με τη συνάρτηση MsgBox επιστρέφεται αντίστοιχα μια από τις παρακάτω τιμές :

Σταθερά	Τιμή	Περιγραφή
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

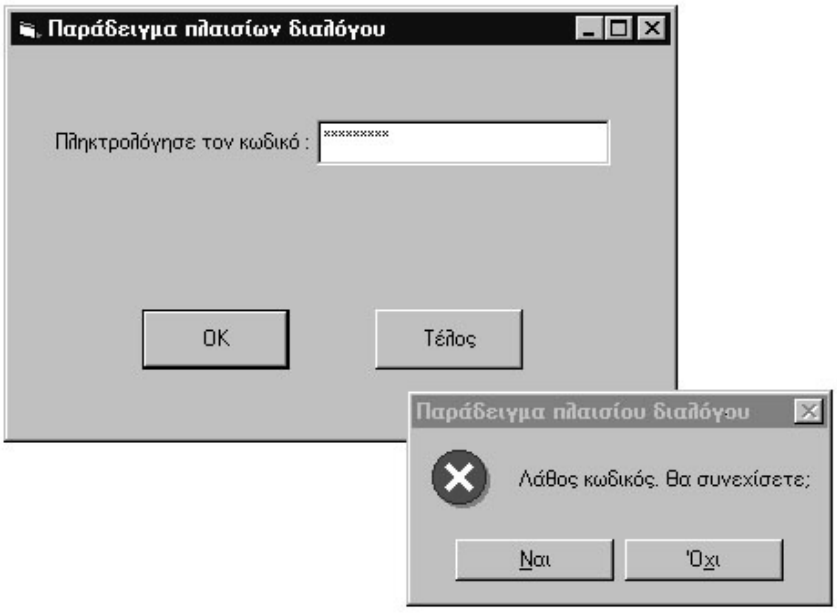
Ο προγραμματιστής, αξιολογεί στο πρόγραμμα την τιμή που επιστράφηκε και εκτελεί την κατάλληλη εργασία. Ο έλεγχος του πλήκτρου που επιλέχτηκε, μπορεί να γίνει απ' ευθείας με την τιμή ή με τη χρήση της αντίστοιχης σταθεράς που επέστρεψε ή συνάρτηση MsgBox. Για παράδειγμα οι παρακάτω δομές ελέγχου είναι ισοδύναμες :

```
If Response = vbOK Then
   Εντολές κώδικα
End IF

If Response = 1 Then
   Εντολές κώδικα
End IF
```

Στη συνέχεια ακολουθεί ο κώδικας της εφαρμογής :

```
Private Sub CmdOK_Click()
    If Txtpwd.Text = Date Then
        MsgBox ("Σωστός κωδικός !!!")
    Else
        Dim Response As String
        ' Πλαίσιο ερώτησης προς το χρήστη
        Response = MsgBox("Λάθος κωδικός. Θα συνεχίσετε;", vbYesNo +
                    vbCritical, "Παράδειγμα πλαισίου διαλόγου")
        If Response = vbYes Then
            ' Καθαρισμός και εστίαση του πλαισίου  κειμένου
            Txtpwd.Text = ""
            Txtpwd.SetFocus
        Else
            ' Εκτέλεση της διαδικασίας τερματισμού
            CmdEnd_Click
        End If
    End If
End Sub
Private Sub CmdEnd_Click()
   End
End Sub
```

Με τη μέθοδο Setfocus, μπορούμε κάθε φορά να εστιάσουμε προγραμματιστικά ένα αντικείμενο στην οθόνη του προγράμματος.



***Σχ. 11.2.** Η εφαρμογή εισαγωγής κωδικού κατά την εκτέλεση.*

Για να είναι κατανοητός ο κώδικας, φρόντισε να υπάρχει η εντολή τερματισμού του προγράμματος σε ένα μόνο σημείο. Δημιούργησε μια ρουτίνα τερματισμού του προγράμματος και κάλεσέ την, όπου χρειάζεται. Στο παραπάνω παράδειγμα, όταν ο χρήστης επιλέξει το πλήκτρο No στο πλαίσιο διαλόγου της συνάρτησης MsgBox, δεν τερματίζεται η εφαρμογή με την εντολή End, αλλά καλείται η ρουτίνα γεγονότος CmdEnd_Click που εκτελεί την εργασία. Σε μεγαλύτερα προγράμματα χρησιμοποίησε γενικές ρουτίνες τερματισμού του προγράμματος.

Περιβάλλον προγραμματισμού Delphi

```
program Prj2;
uses
  Forms,
  Unit2 in 'Unit2.pas' {Frmpwd};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TFrmpwd, Frmpwd);
  Application.Run;
end.
unit Unit2;
interface
```

```
uses Windows, SysUtils, Classes, Graphics, Forms, Controls,
StdCtrls,
  Buttons,Dialogs;
type
  TFrmpwd = class(TForm)
    Label1: TLabel;
    Txtpwd: TEdit;
    CmdOK: TButton;
    CmdEnd: TButton;
    procedure CmdEndClick(Sender: TObject);
    procedure CmdOKClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Frmpwd: TFrmpwd;
implementation
{$R *.DFM}
procedure TFrmpwd.CmdEndClick(Sender: TObject);
begin
     close;
end;
procedure TFrmpwd.CmdOKClick(Sender: TObject);
begin
     if Txtpwd.Text = DateToStr(Date) then
        ShowMessage('Σωστός κωδικός !!!')
     else
         begin
         if MessageDlg( 'Λάθος κωδικός. Θα συνεχίσετε ?',
            mtConfirmation , [mbYes, mbNo], 0 ) = mrYES then
            begin
              Txtpwd.Text := '';
              Txtpwd.SetFocus;
            end
         else
         close;
     end;
end;
end.
```

Παράδειγμα 3

Να γραφεί πρόγραμμα με το οποίο θα γίνεται η γραφική προσομοίωση της προσγείωσης ενός αεροπλάνου. Η προσομοίωση της προσγείωσης μπορεί να γίνει με την μετακίνηση ενός εργαλείου εικόνας προς το κάτω μέρος της φόρμας με παράλληλη αύξηση του μεγέθους του.

Περιβάλλον προγραμματισμού Visual Basic

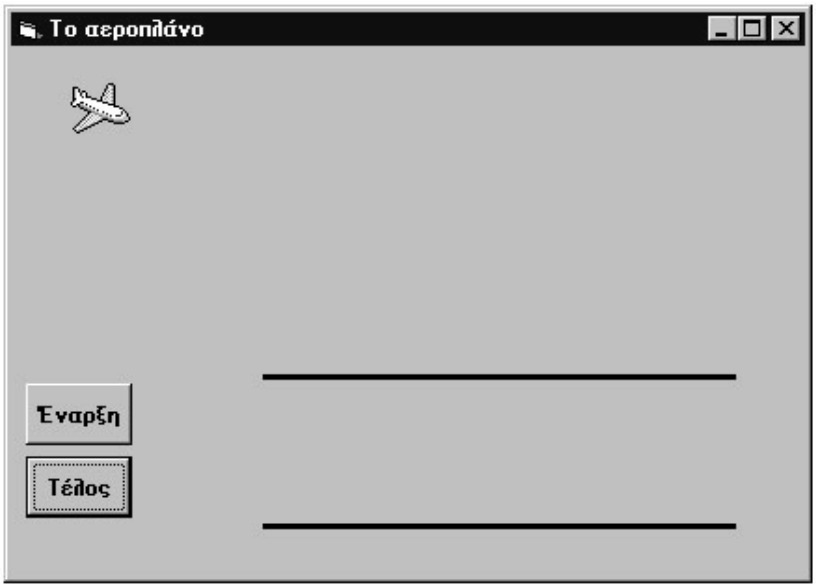
Για την υλοποίηση της εφαρμογής προτείνουμε να χρησιμοποιηθούν τα παρακάτω αντικείμενα:

- μια **φόρμα** η οποία είναι το παράθυρο μέσα στο οποίο θα εκτελείται η εφαρμογή,
- μια **εικόνα** (image) που θα περιέχει την εικόνα του αεροπλάνου,
- δύο **γραμμές** με τις οποίες θα δημιουργήσουμε το διάδρομο προσγείωσης,
- δύο **πλήκτρα εντολής** που θα χρησιμεύουν για την έναρξη και τον τερματισμό της εφαρμογής,
- ένα εργαλείο **χρονομέτρη** (timer). Με το χρονομέτρη σε συγκεκριμένα χρονικά διαστήματα θα μετακινείται και θα αλλάζει το μέγεθος του αεροπλάνου.

Στις ιδιότητες των αντικειμένων της εφαρμογής αποδίδουμε τις παρακάτω τιμές:

Αντικείμενο	Ιδιότητα	Τιμή
Φόρμα	Name	FrmPlane
	Caption	Το αεροπλάνο
Εικόνα	Name	imgplane
	Height	480
	Picture	\vb\graphics\icons\industry\plane.ico
	Stretch	True
	Width	480
Γραμμή1	Name	Line1
	BorderWidth	3
Γραμμή2	Name	Line2
	BorderWidth	3
Πλήκτρο εντολής 1	Name	CmdStart
	Caption	Έναρξη
Πλήκτρο εντολής 2	Name	CmdEnd
	Caption	Τέλος
Χρονομέτρης	Name	Tmrmove
	Interval	0

Μετά την τοποθέτηση των αντικειμένων και την αντιστοίχιση των ιδιοτήτων έχουμε ολοκληρώσει το τρόπο επικοινωνίας χρήστη εφαρμογής (Σχ. 11.3).



***Σχ. 11.3**. Η διασύνδεση του χρήστη με την εφαρμογή στο παράδειγμά μας.*

Ένα εργαλείο χρονομέτρη υποστηρίζει το γεγονός Timer, το οποίο δεν προκαλείται από το χρήστη αλλά από το ρολόι του συστήματος. Η ιδιότητα Interval, καθορίζει σε χιλιοστά του δευτερολέπτου, το χρονικό διάστημα που το σύστημα θα προκαλέσει ένα γεγονός Timer. Η τιμή 0 που αποδώσαμε κατά το χρόνο σχεδιασμού αρχικά απενεργοποιεί το εργαλείο.

Το εργαλείο χρονομέτρη δεν εμφανίζεται κατά την εκτέλεση της εφαρμογής, γι αυτό το λόγο δεν έχει σημασία σε ποιο σημείο της φόρμας θα τοποθετηθεί. Φυσικά προσπαθούμε να το τοποθετήσουμε σε κάποιο σημείο που δεν θα καλύπτεται από κάποιο άλλο εργαλείο, για να γίνεται αντιληπτή κατά το χρόνο σχεδιασμού η ύπαρξη του.

Για να επιτύχουμε την προσομοίωση της προσγείωσης, όταν προκληθεί ένα γεγονός Timer με την ενέργεια της μεθόδου **Move**, μετακινούμε το αεροπλάνο, μετατρέποντας τις ιδιότητες Left και Top και αυξάνουμε το μέγεθός του μετατρέποντας τις ιδιότητες Width και Height της Εικόνας. Οι ιδιότητες αυτές εκφράζουν αντίστοιχα :

Left : την απόσταση της πάνω αριστερά γωνίας του αντικειμένου από την αριστερή πλευρά της φόρμας

Top : την απόσταση της πάνω αριστερά γωνίας του αντικειμένου από την κορυφή της φόρμας

Width: το πλάτος του εργαλείου

Height: το ύψος του εργαλείου

Η προσγείωση του αεροπλάνου θα πραγματοποιηθεί ανάμεσα στις δύο γραμμές που περιγράφουν το διάδρομο προσγείωσης. Κατά συνέπεια πρέπει να γνωρίζουμε τη θέση τους για να τερματίσουμε τη προσγείωση. Η θέση ενός αντικειμένου γραμμή καθορίζεται από τις παρακάτω ιδιότητες :

X1 : η x συντεταγμένη του σημείου έναρξης της γραμμής

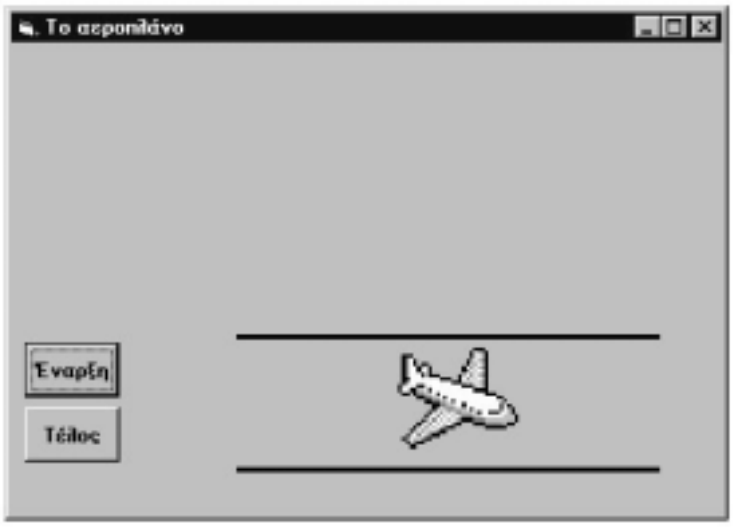
X2 : η x συντεταγμένη του σημείου τέλους της γραμμής

Y1 : η y συντεταγμένη του σημείου έναρξης της γραμμής

Y2 : η y συντεταγμένη του σημείου τέλους της γραμμής

Για την προσομοίωση της προσγείωσης του αεροπλάνου, πρέπει να συμπεριλάβουμε τις παρακάτω εντολές κώδικα στις αντίστοιχες διαδικασίες γεγονότων:

```
Private Sub CmdStart_Click()
' Πλήκτρο έναρξης
' Με την απόδοση της τιμής 5 στην ιδιότητα Interval  ενεργοποιούμε
' το εργαλείο χρονομέτρη
' κάθε πέντε  χιλιοστά του δευτερολέπτου
    Timer1.Interval = 5
End Sub
Private Sub Timer1_Timer()
    ' Έλεγχος αν η κορυφή του αεροπλάνου περάσει την πρώτη γραμμή
                                        του διαδρόμου προσγείωσης.
    ' Η ιδιότητα Y1 του εργαλείου γραμμής αναφέρεται στην y
                    συντεταγμένη του σημείου έναρξής της.
    If Imgplane.Top < Line1.Y1 Then
        Imgplane.Move Imgplane.Left + 25, Imgplane.Top + 20,
        Imgplane.Width + 5, Imgplane.Height + 5
    Else
        Timer1.Interval = 0
          ' Απενεργοποίηση του εργαλείου χρονομέτρη
    End If
End Sub
Private Sub CmdEnd_Click(Index As Integer)
    ' Πλήκτρο τερματισμού
    End
End Sub
```



***Σχ. 11.4.** Η προσγείωση του αεροπλάνου μετά την εκτέλεση του παραδείγματος*

Περιβάλλον προγραμματισμού Delphi

```
program Prj3;
uses
  Forms,
  Unit3 in 'Unit3.pas' {FrmPlane};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TFrmPlane, FrmPlane);
  Application.Run;
end.
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, Buttons, ExtCtrls;
type
  TFrmPlane = class(TForm)
    Imgplane: TImage;
    Panel1: TPanel;
    Panel2: TPanel;
    CmdStart: TBitBtn;
    CmdEnd: TBitBtn;
    Timer1: TTimer;
    procedure CmdStartClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure CmdEndClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FrmPlane: TFrmPlane;
implementation
{$R *.DFM}
procedure TFrmPlane.CmdStartClick(Sender: TObject);
begin
     Timer1.Interval := 1000;
end;
procedure TFrmPlane.Timer1Timer(Sender: TObject);
begin
     if ImgPlane.Top < Panel1.Top then
        begin
        ImgPlane.Top := ImgPlane.Top + 20;
        ImgPlane.Left := ImgPlane.Left + 25;
        ImgPlane.Height := ImgPlane.Height + 5;
        ImgPlane.Width  := ImgPlane.Width + 5;
```

```
        end;
end;
procedure TFrmPlane.CmdEndClick(Sender: TObject);
begin
     close;
end;
end.
```

Παράδειγμα 4

Να κατασκευαστεί μια εφαρμογή επεξεργασίας κειμένου στην οποία οι βασικές εργασίες θα πραγματοποιούνται μέσα από επιλογές ενός κεντρικού μενού επιλογών.

Περιβάλλον προγραμματισμού Visual Basic

Η υλοποίηση του επεξεργαστή κειμένου θα γίνει με τη χρήση των παρακάτω αντικείμενων:

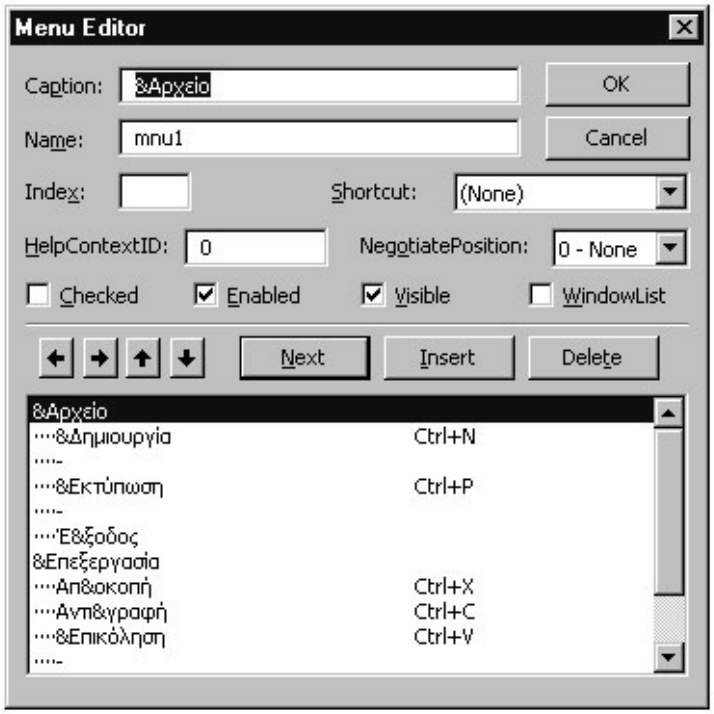
- μια **φόρμα** που θα είναι το παράθυρο μέσα στο οποίο θα εκτελείται η εφαρμογή,
- ένα **πλαίσιο κειμένου** στο οποίο θα γίνεται η πληκτρολόγηση του κειμένου.

Στις ιδιότητες των αντικειμένων της εφαρμογής αποδίδουμε τις παρακάτω τιμές:

Αντικείμενο	Ιδιότητα	Τιμή
Φόρμα	Name	FrmMain
	Caption	Ο επεξεργαστής κειμένου
Πλαίσιο κειμένου	Name	TxtEdit
	Multiline	True
	Text	(κενό)

Το επόμενο βήμα είναι η δημιουργία του κεντρικού μενού επιλογών. Η Visual Basic μας παρέχει ένα εργαλείο σχεδιασμού μενού επιλογών που καλείται **επεξεργαστής μενού** (menu editor) με το οποίο σχεδιάζουμε τις επιλογές του μενού (Σχήμα 11.5). Κάθε επιλογή που δημιουργούμε αποτελεί ανεξάρτητο αντικείμενο που υποστηρίζει ένα καθορισμένο αριθμό ιδιοτήτων και επιπλέον το γεγονός κλικ. Μόλις ο χρήστης επιλέξει κάποια από τις επιλογές του μενού, ανιχνεύεται το γεγονός κλικ και εκτελείται ο κώδικας που έχουμε συμπεριλάβει στη αντίστοιχη διαδικασία γεγονότος του αντικειμένου μενού.

Μόλις ενεργοποιήσουμε το παράθυρο του επεξεργαστή μενού, δημιουργούμε τις παρακάτω επιλογές :



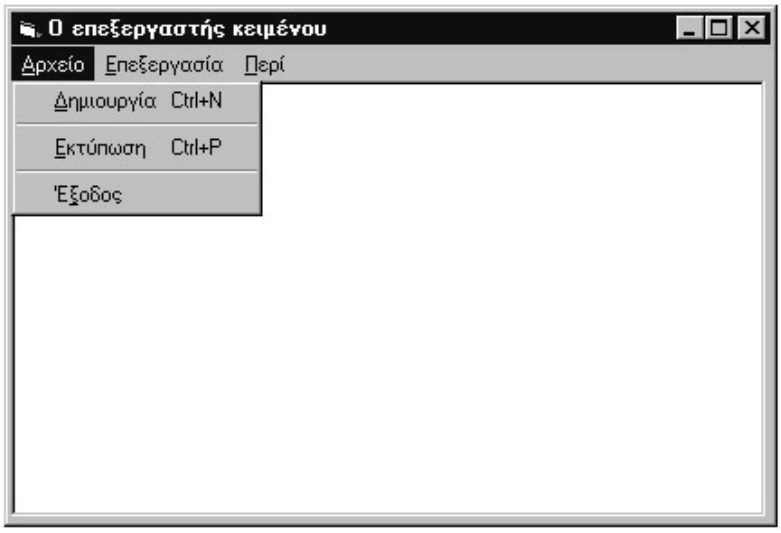
Σχ. 11.5. *Ο επεξεργαστής μενού της Visual Basic.*

Μενού	Επιλογή (Ιδιότητα Caption)	Όνομα (Ιδιότητα Name)	Πλήκτρο συντομίας (Ιδιότητα Shortcut)
&Αρχείο	&Δημιουργία	mnunew	Ctrl+N
	-	separator1	
	&Εκτύπωση	mnuprint	Ctrl+P
	-	separator2	
	Έ&ξοδος	mnuexit	
&Επεξεργασία	Απ&οκοπή	mnucut	Ctrl+X
	Αντι&γραφή	mnucopy	Ctrl+C
	&Επικόλληση	mnupaste	Ctrl+V
	-	separator3	
	Ε&πιλογή όλων	mnuSelall	Ctrl+A
&Περί		mnuAbout	

Η ιδιότητα **Caption** περιέχει το λεκτικό με το οποίο θα εμφανίζεται η επιλογή στο μενού, η ιδιότητα **Name** είναι το όνομα που θα αναφερόμαστε από τον κώδικα στο αντικείμενο μενού και η ιδιότητα **Shortcut** εκφράζει τα πλήκτρα συντομίας με τα οποία

είναι δυνατόν να εκτελέσουμε άμεσα την επιλογή χωρίς την χρήση του μενού.

Μετά από τη δημιουργία και του κεντρικού μενού επιλογών έχει ολοκληρωθεί η δημιουργία της βασικής φόρμας της εφαρμογής (Σχήμα 11.6).



Σχ. 11.6. Η βασική φόρμα της διασύνδεσης του χρήστη με τον επεξεργαστή κειμένου

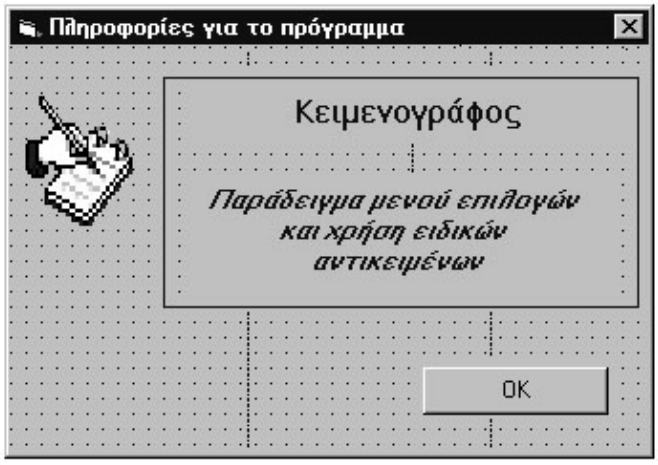
Με την επιλογή **Περί** του κεντρικού μενού επιλογών πρόκειται να εμφανίζουμε μια δεύτερη φόρμα που θα περιέχει πληροφοριακά στοιχεία για τη εφαρμογή. Αφού προσθέσουμε στο έργο (project) μια νέα φόρμα τοποθετούμε επάνω της τα παρακάτω αντικείμενα :

- δύο **ετικέτες** που θα περιέχουν τα πληροφοριακά στοιχεία της εφαρμογής,
- ένα **σχήμα** (shape) για την ομαδοποίηση των ετικετών,
- μια **εικόνα** ως διακοσμητικό,
- ένα **πλήκτρο εντολής** που θα κλείνει τη δεύτερη φόρμα.

Στις ιδιότητες των αντικειμένων της δεύτερης φόρμας αποδίδουμε τις παρακάτω τιμές:

Αντικείμενο	Ιδιότητα	Τιμή
Φόρμα	Caption	Πληροφορίες για το πρόγραμμα
	Name	FrmAbout
Ετικέτα1	Alignment	2-Center
	Caption	Κειμενογράφος
	ForeColor	&H8000000D& (Highlight)
	Name	Lblinfo1
Ετικέτα2	Alignment	2-Center
	Caption	Παράδειγμα μενού επιλογών και χρήση ειδικών αντικειμένων
	ForeColor	&H8000000D& (Highlight)
	Name	Lblinfo2
Σχήμα	ForeColor	&H8000000D& (Highlight)
	Name	Shape1
	Shape	0-Rectangle
Εικόνα	Name	Image1
	Picture	\vb\graphics\icons\writing\note16.ico
	Stretch	True
Πλήκτρο εντολής	Caption	OK
	Name	CmdBack

Με τον σχεδιασμό της δεύτερης φόρμας (Σχήμα 11.7) ολοκληρώθηκε η διεπαφή του χρήστη με τον επεξεργαστή κειμένου.



Σχ. 11.7. Η δεύτερη φόρμα του επεξεργαστή κειμένου

Στον κώδικα του συγκεκριμένου παραδείγματος πρέπει να χρησιμοποιήσουμε δύο ειδικά αντικείμενα που υποστηρίζει η Visual Basic, το **Printer** που εκφράζει τον εκτυπωτή και το **Clipboard** (Πρόχειρο) που εκφράζει μια περιοχή μνήμης, η οποία είναι προσπελάσιμη από όλες τις εφαρμογές του συστήματος.

Το ειδικό αντικείμενο Printer δέχεται δεδομένα με τη χρήση της μεθόδου **Print**. Όταν ολοκληρώσουμε την τοποθέτηση των δεδομένων στο ειδικό αντικείμενο Printer, στέλνουμε τα περιεχόμενά του για εκτύπωση με τη μέθοδο **EndDoc**.

Με τη χρήση του Clipboard μπορούμε να χειριστούμε δεδομένα κειμένου ή γραφικών. Όταν πρόκειται για αποστολή κειμένου από την εφαρμογή προς το Clipboard πρέπει να χρησιμοποιήσουμε τη μέθοδο SetText και για λήψη τη μέθοδο GetText, ενώ όταν πρόκειται για γραφικά χρησιμοποιούμε αντίστοιχα τις μεθόδους SetData και GetData.

Ακόμη για την επιλογή **Επιλογή όλων** του μενού θα χρησιμοποιήσουμε τη συνάρτηση **Len**, η οποία υπολογίζει τον αριθμό χαρακτήρων ενός αλφαριθμητικού και επιπλέον τις παρακάτω ιδιότητες που υποστηρίζει το πλαίσιο κειμένου :

- **SelLength** : επιστρέφει τον αριθμό των επιλεγμένων χαρακτήρων.
- **SelStart** : επιστρέφει το σημείο έναρξης ενός επιλεγμένου κειμένου ή τοποθετεί το σημείο εισαγωγής σε συγκεκριμένο σημείο.
- **SelText** : περιέχει το κείμενο που έχει επιλεγεί ή το μηδενικού μήκους αλφαριθμητικό (“”) όταν δεν έχει επιλεγεί τίποτα.

Τα βασικά τμήματα του κώδικα της εφαρμογής πρέπει να τοποθετηθούν στις διαδικασίες γεγονότων κλικ των αντικειμένων μενού.

```
Private Sub mnunew_Click()
    ' Για τον καθαρισμό του πλαισίου κειμένου αποδίδουμε το
        μηδενικού μήκους αλφαριθμητικό στην ιδιότητα Text.
    Text1.Text = ""
End Sub
Private Sub mnuprint_Click()
    ' Εκτύπωση των περιεχομένων του πλαισίου κειμένου TxtEdit
    ' με τη χρήση του ειδικού αντικειμένου Printer
    Printer.Print TxtEdit.Text
    Printer.EndDoc
End Sub
Private Sub mnuexit_Click()
    End
End Sub
Private Sub mnucut_Click()
    ' Αποστολή των περιεχομένων του πλαισίου κειμένου TxtEdit στο
                                        ειδικό αντικείμενο Clipboard.
    Clipboard.SetText TxtEdit.SelText
    ' Καθαρισμός των περιεχομένων του πλαισίου κειμένου TxtEdit
    TxtEdit.SelText = ""
End Sub
Private Sub mnucopy_Click()
```

```
    ' Αποστολή των περιεχομένων του πλαισίου κειμένου TxtEdit στο
                                    ειδικό αντικείμενο Clipboard.
    Clipboard.SetText TxtEdit.SelText
End Sub
Private Sub mnupaste_Click()
    ' Μεταφορά των περιεχομένων του Clipboard στο πλαισίου κειμένου
                                                              TxtEdit
    TxtEdit.SelText = Clipboard.GetText
End Sub
Private Sub mnuSelall_Click()
    ' Τοποθέτηση του σημείου εισαγωγής κειμένου στην αρχή του
                                    πλαισίου κειμένου TxtEdit.
    TxtEdit.SelStart = 0
    ' Επιλογή όλων των περιεχομένων του πλαισίου κειμένου TxtEdit.
    TxtEdit.SelLength = Len(TxtEdit.Text)
End Sub
Private Sub mnuAbout_Click()
    ' Με τη μέθοδο Show εμφανίζεται η δεύτερη φόρμα.
    FrmAbout.Show
End Sub
```

Στον κώδικα της δεύτερης φόρμας θα συμπεριλάβουμε στη διαδικασία γεγονότος κλικ του πλήκτρου εντολής OK τον κώδικα με τον οποίο θα αποκρύπτεται η φόρμα.

```
Private Sub CmndBack_Click()
    ' Με τη μέθοδο Hide κρύβεται η δεύτερη φόρμα.
    FrmAbout.Hide
End Sub
```

Στο πλαίσιο κειμένου έχουμε αποδώσει την τιμή True στην ιδιότητα Multiline. Έτσι όταν γεμίσει μια γραμμή του πλαισίου αυτόματα μεταφέρεται το σημείο εισαγωγής κειμένου στην επόμενη γραμμή. Αυτό ίσως μας δημιουργήσει προβλήματα κατά την εκτύπωση, γιατί η μέθοδος Print δεν στέλνει στο ειδικό αντικείμενο Printer κανένα χαρακτήρα αλλαγής γραμμής. Η λύση είναι να μεταφέρει ο χρήστης το σημείο εισαγωγής στην επόμενη γραμμή πατώντας το πλήκτρο Enter.

Περιβάλλον προγραμματισμού Delphi

```
program Prj4;
uses
  Forms,
  Unit4 in 'Unit4.pas' {Form1},
  Unit41 in 'Unit41.pas' {FrmAbout};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TFrmAbout, FrmAbout);
  Application.Run;
end.
unit Unit4;
```

```
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  Menus, StdCtrls, Printers;
type
  TForm1 = class(TForm)
    Memo1: TMemo;
    MainMenu1: TMainMenu;
    mnuEdit: TMenuItem;
    mnuFile: TMenuItem;
    mnuAbout: TMenuItem;
    mnuNew: TMenuItem;
    mnuPrint: TMenuItem;
    mnuExit: TMenuItem;
    mnuCut: TMenuItem;
    mnuPaste: TMenuItem;
    mnuCopy: TMenuItem;
    mnuSelAll: TMenuItem;
    procedure mnuExitClick(Sender: TObject);
    procedure mnuCutClick(Sender: TObject);
    procedure mnuCopyClick(Sender: TObject);
    procedure mnuPasteClick(Sender: TObject);
    procedure mnuNewClick(Sender: TObject);
    procedure mnuSelAllClick(Sender: TObject);
    procedure mnuAboutClick(Sender: TObject);
    procedure mnuPrintClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
uses Unit41;
{$R *.DFM}
procedure TForm1.mnuExitClick(Sender: TObject);
begin
     close;
end;
procedure TForm1.mnuCutClick(Sender: TObject);
begin
     Memo1.CutToClipboard;
end;
procedure TForm1.mnuCopyClick(Sender: TObject);
begin
     Memo1.CopyToClipboard;
end;
procedure TForm1.mnuPasteClick(Sender: TObject);
```

```
begin
     Memo1.PasteFromClipboard;
end;
procedure TForm1.mnuNewClick(Sender: TObject);
begin
     Memo1.Clear;
end;
procedure TForm1.mnuSelAllClick(Sender: TObject);
begin
     Memo1.SelStart := 0;
     Memo1.Sellength := 32767;
end;
procedure TForm1.mnuAboutClick(Sender: TObject);
begin
     FrmAbout.Show;
end;
procedure TForm1.mnuPrintClick(Sender: TObject);
var
   index : Integer;
begin
     Printer.BeginDoc;
     for index := 0 to Memo1.Lines.Count-1 do
         Printer.Canvas.TextOut(100,10+index*50,Memo1.Lines[index]);
     Printer.EndDoc;
end;
end
unit Unit41;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, ExtCtrls;
type
  TFrmAbout = class(TForm)
    CmndBack: TButton;
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Image1: TImage;
    procedure CmndBackClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FrmAbout: TFrmAbout;
implementation
```

```
{$R *.DFM}
procedure TFrmAbout.CmndBackClick(Sender: TObject);
begin
     FrmAbout.Hide;
end;
end.
```

Παράδειγμα 5

Να κατασκευάσετε μια εφαρμογή παρουσίασης των ελληνικών μεταλλίων στους Ολυμπιακούς Αγώνες της Ατλάντα. Δημιουργήστε ολοκληρωμένη παρουσίαση χρησιμοποιώντας και αρχεία βίντεο.

Περιβάλλον προγραμματισμού Visual Basic

Για την υλοποίηση της παρουσίασης θα χρησιμοποιήσουμε τα παρακάτω αντικείμενα:

- μια **φόρμα** που θα είναι το παράθυρο μέσα στο οποίο θα εκτελείται η εφαρμογή,
- τεσσάρων **πλήκτρων επιλογής** (option buttons) για την επιλογή του αθλήματος,
- ένα **πλαίσιο** (frame) στο οποίο θα γίνεται η ομαδοποίηση των πλήκτρων επιλογής,
- έξι **ετικετών** που θα χρησιμεύουν για την εμφάνιση των πληροφοριών της εφαρμογής,
- δύο **πλήκτρων εντολής** για την έναρξη του βίντεο και τον τερματισμό της εφαρμογής.

Στις ιδιότητες των αντικειμένων της εφαρμογής αποδίδουμε τις παρακάτω τιμές:

Αντικείμενο	Ιδιότητα	Τιμή
Φόρμα	Name	FrmMain
	Caption	Παραδείγματα video
Πλήκτρο επιλογής 1	Name	Option1
	Caption	Άρση βαρών
Πλήκτρο επιλογής 2	Name	Option2
	Caption	Γυμναστική
Πλήκτρο επιλογής 3	Name	Option3
	Caption	Ιστιοπλοΐα
Πλήκτρο επιλογής 4	Name	Option4
	Caption	Στίβος

Αντικείμενο	Ιδιότητα	Τιμή
Πλαίσιο	Name	FrOptions
	Caption	ΑΘΛΗΜΑΤΑ
Ετικέτα0	Name	LblTitle0
	Caption	Ελληνικά μετάλλια στην Ατλάντα
Ετικέτα1	Name	LblTitle1
	Caption	Αθλητής:
Ετικέτα2	Name	LblTitle2
	Caption	Αγώνισμα:
Ετικέτα3	Name	LblTitle3
	Caption	Μετάλλιο:
Ετικέτα4	Name	LblFullName
	BorderStyle	1-Fixed Single
	Caption	(κενό)
Ετικέτα5	Name	LblSport
	BorderStyle	1-Fixed Single
	Caption	(κενό)
Ετικέτα6	Name	LblMedal
	BorderStyle	1-Fixed Single
	Caption	(κενό)
Πλήκτρο εντολής 1	Name	CmdStart
	Caption	Έναρξη
Πλήκτρο εντολής 2	Name	CmdEnd
	Caption	Τέλος

Στη συνέχεια πρέπει να δώσουμε στην εφαρμογή μας και δυνατότητα χρήσης βίντεο. Για να το επιτύχουμε πρέπει να συνεργαστεί η εφαρμογή μας με κάποια εφαρμογή εκτέλεσης αρχείων βίντεο. Η συνεργασία των εφαρμογών επιτυγχάνεται με την αξιοποίηση της τεχνολογίας **OLE** (Object Linking and Embedding) που υποστηρίζει η Visual Basic και γενικότερα οι εφαρμογές που εργάζονται στο περιβάλλον των Windows.

Στο παράδειγμά μας θα επιδείξουμε μόνο το βίντεο του αθλήματος της ιστιοσανίδας και με τον ίδιο τρόπο μπορείς να υλοποιήσεις τα υπόλοιπα OLE εργαλεία. Για τη σύνδεση των εφαρμογών, επιλέγουμε από την εργαλειοθήκη και τοποθετούμε ένα αντικείμενο OLE επάνω στη φόρμα της εφαρμογής μας. Αμέσως εμφανίζεται το παράθυρο διαλόγου **Εισαγωγή αντικειμένου**. Στο παράθυρο διαλόγου επιλέγουμε **Δη-**

μιουργία από αρχείο και πατώντας το πλήκτρο εντολής **Αναζήτηση** εντοπίζουμε από το δίσκο του συστήματος το αρχείο .avi που πρόκειται να χρησιμοποιήσουμε. Στο παράδειγμά μας θα χρησιμοποιήσουμε το αρχείο windsurf2.avi. Εκτελώντας τις παραπάνω ενέργειες θα επιτύχουμε τη διασύνδεση των περιεχομένων του αρχείου σαν αντικείμενο της εφαρμογής μας. Όταν ενεργοποιήσουμε το αντικείμενο OLE το βίντεο θα εκτελεστεί μέσα από το πρόγραμμα που έχει δημιουργηθεί. Στο αντικείμενο OLE αντιστοιχούμε ακόμη τις παρακάτω ιδιότητες:

Αντικείμενο	Ιδιότητα	Τιμή
OLE	Name	OLE3
	Sizemode	2-Autosize
	Visible	False

Μετά τα δημιουργία του αντικειμένου OLE στην ιδιότητα **Class** έχει τοποθετηθεί η τιμή avifile και στην ιδιότητα **SourceDoc** έχει τοποθετηθεί η διαδρομή και το όνομα του αρχείου βίντεο που διασυνδέσαμε.

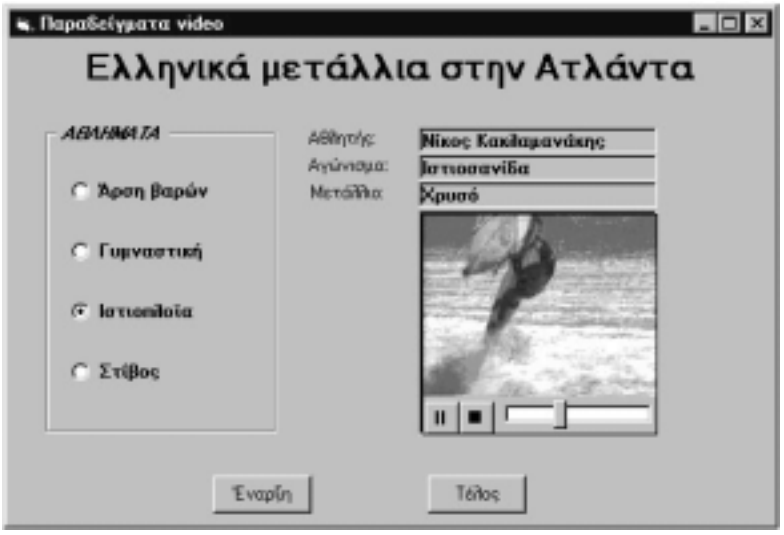
Μετά την αντιστοίχιση των ιδιοτήτων των αντικειμένων έχουμε ολοκληρώσει το τρόπο επικοινωνίας της εφαρμογής με το χρήστη.

Σχ. 11.8. Η φόρμα της εφαρμογής παρουσίασης

Στη συνέχεια πρέπει να γράψουμε το κώδικα της εφαρμογής:

```
Private Sub CmdStart_Click()
    ' Έλεγχος επιλεγμένου αθλήματος
    If Option3.Value = True Then
   ' Επιλέχτηκε το άθλημα της Ιστιοσανίδας
        ' Ενημέρωση ετικετών
        lblFullName.Caption = "Νίκος Κακλαμανάκης"
        lblSport.Caption = "Ιστιοσανίδα"
        lblMedal.Caption = "Χρυσό"
        'Ενεργοποίηση του αντικειμένου OLE
        OLE3.Action = 7
    End If
End Sub
Private Sub CmdEnd_Click()
    End
End Sub
```

Στον κώδικα της διαδικασίας CmdStart_Click() δεν χρησιμοποιούμε την ιδιότητα Visible για να εμφανίσουμε ή να αποκρύψουμε το εργαλείο OLE, γιατί επιτυγχάνεται αυτόματα με την ενεργοποίηση της εφαρμογής βίντεο.



***Σχ. 11.9.** Η εφαρμογή παρουσίασης κατά το χρόνο εκτέλεσης*

Κατά την εκτέλεση της εφαρμογής μαζί με το πλαίσιο προβολής βίντεο, εμφανίζονται δύο πλήκτρα ελέγχου που επιτρέπουν το πρώτο τη παύση ή τη συνέχιση της προβολής βίντεο και το δεύτερο τη διακοπή της.

Περιβάλλον προγραμματισμού Delphi

```
program Prj5;
uses
  Forms,
  Unit5 in 'Unit5.pas' {FrmMain};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TFrmMain, FrmMain);
  Application.Run;
end.
unit Unit5;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls, ExtCtrls, MPlayer;
type
  TFrmMain = class(TForm)
    OptionGroup: TRadioGroup;
    LblTitle0: TLabel;
    LblTitle1: TLabel;
    LblTitle2: TLabel;
    LblTitle3: TLabel;
    FullName: TEdit;
    Sport: TEdit;
    Medal: TEdit;
    CmdStart: TButton;
    MediaPlayer1: TMediaPlayer;
    Panel1: TPanel;
    CmdEnd: TButton;
    procedure CmdStartClick(Sender: TObject);
    procedure CmdEndClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FrmMain: TFrmMain;
implementation
{$R *.DFM}
procedure TFrmMain.CmdStartClick(Sender: TObject);
begin
     if OptionGroup.ItemIndex = 2 then
        begin
        Fullname.Text := ' Νίκος Κακλαμανάκης;
        Sport.Text := ' Ιστιοσανίδα';
        Medal.Text := 'Χρυσό';
```

```
        MediaPlayer1.Open;
        MediaPlayer1.Rewind;
        MediaPlayer1.Play;
        end;
end;
procedure TFrmMain.CmdEndClick(Sender: TObject);
begin
     close;
end;
end.
```

Παράδειγμα 6

Υλοποιήστε κατά το χρόνο σχεδιασμού μιας Visual Basic εφαρμογής τη σύνδεσή της με μια άλλη εφαρμογή του περιβάλλοντος των Windows, όπως το Microsoft Excel.

Περιβάλλον προγραμματισμού Visual Basic

Εάν θέλουμε μια εφαρμογή μας της Visual Basic να χρησιμοποιήσει δεδομένα που ανανεώνονται συνέχεια και προέρχονται από κάποια άλλη εφαρμογή, τότε θα πρέπει να χρησιμοποιήσουμε τις δυνατότητες που μας προσφέρει η Dynamic Data Exchange (DDE). Μα τι είναι η DDE; Είναι ένας μηχανισμός που υποστηρίζεται από τα Windows και επιτρέπει σε δύο εφαρμογές να ανταλλάσσουν αυτόματα μεταξύ τους δεδομένα. Οι δύο εφαρμογές που ανταλλάσσουν πληροφορίες εμπλέκονται σε μία DDE "συνομιλία". Η ανταλλαγή των δεδομένων προϋποθέτει κατ' αρχήν τη δημιουργία συνδέσμου. Η εφαρμογή η οποία αρχίζει τη "συνομιλία" ονομάζεται προορισμός (destination) και η εφαρμογή που ανταποκρίνεται ονομάζεται πηγή (source).

Κατά τη διάρκεια μιας DDE "συνομιλίας" πρέπει να προσδιοριστεί το θέμα της (topic). Κατά τη διάρκεια της "συνομιλίας" ο προορισμός και η πηγή μπορούν να ανταλλάσσουν πληροφορίες που αφορούν ένα ή περισσότερα άρθρα (items) σχετικά με το θέμα, τα οποία δεν είναι παρά αναφορές σε δεδομένα που ενδιαφέρουν και τις δύο εφαρμογές. Το τι ακριβώς μπορεί να είναι ένα άρθρο εξαρτάται από την εφαρμογή. Για παράδειγμα, το Excel αναγνωρίζει σαν άρθρο σε μία "συνομιλία", τις συντεταγμένες των κελιών (cells) ενός λογιστικού φύλλου.

Η Visual Basic υποστηρίζει διαδικασίες DDE και κάθε αντικείμενο πλαίσιο κειμένου, πλαίσιο εικόνας και ετικέτα, μπορεί να θεωρηθεί σαν προορισμός, ενώ τα προαναφερθέντα αντικείμενα και κάθε φόρμα θεωρείται σαν πηγή. Στην περίπτωση που μία φόρμα σε μία εφαρμογή της Visual Basic, είναι η πηγή μιας "συνομιλίας", το όνομα καθενός από τα αντικείμενα πλαίσιο κειμένου, πλαίσιο εικόνας και ετικέτα επάνω στη φόρμα, μπορεί να αποτελεί άρθρο για την DDE "συνομιλία". Σημειώνεται ότι μία εφαρμογή μπορεί να εμπλακεί ταυτόχρονα σε πολλές "συνομιλίες" λειτουργώντας σε άλλες σαν προορισμός και σε άλλες σαν πηγή.

Η δημιουργία συνδέσμου, μεταξύ της εφαρμογής της Visual Basic και της άλλης εφαρμογής με την οποία θα ανταλλάσσουν δεδομένα, μπορεί να γίνει είτε στη φάση σχεδίασης της εφαρμογής, είτε στη φάση της εκτέλεσής της. Οι σύνδεσμοι που δη-

μιουργούνται στη φάση της σχεδίασης είναι απλούστεροι, αλλά θέτουν κάποιους περιορισμούς :

- ✓ Κατ' αρχήν η άλλη εφαρμογή θα πρέπει να περιέχει στο μενού επιλογών Edit την επιλογή Paste Link.
- ✓ Επιπλέον, όταν εκτελείται η εφαρμογή της Visual Basic, η άλλη εφαρμογή θα πρέπει να είναι διαθέσιμη εκείνη τη στιγμή.
- ✓ Τέλος, επειδή η ενημέρωση του αντικειμένου της εφαρμογής της Visual Basic που περιέχει τα δεδομένα που προέρχονται από την άλλη εφαρμογή, γίνεται αυτόματα όποτε σημειώνεται κάποια αλλαγή, αυτό μπορεί να έχει σαν αποτέλεσμα την αποτυχία της ενημέρωσης (κάτω από ορισμένες συνθήκες), αν η άλλη εφαρμογή δεν έχει δυνατότητα κάτω από αυτές τις συνθήκες να ανανεώσει το περιεχόμενο του συνδέσμου.

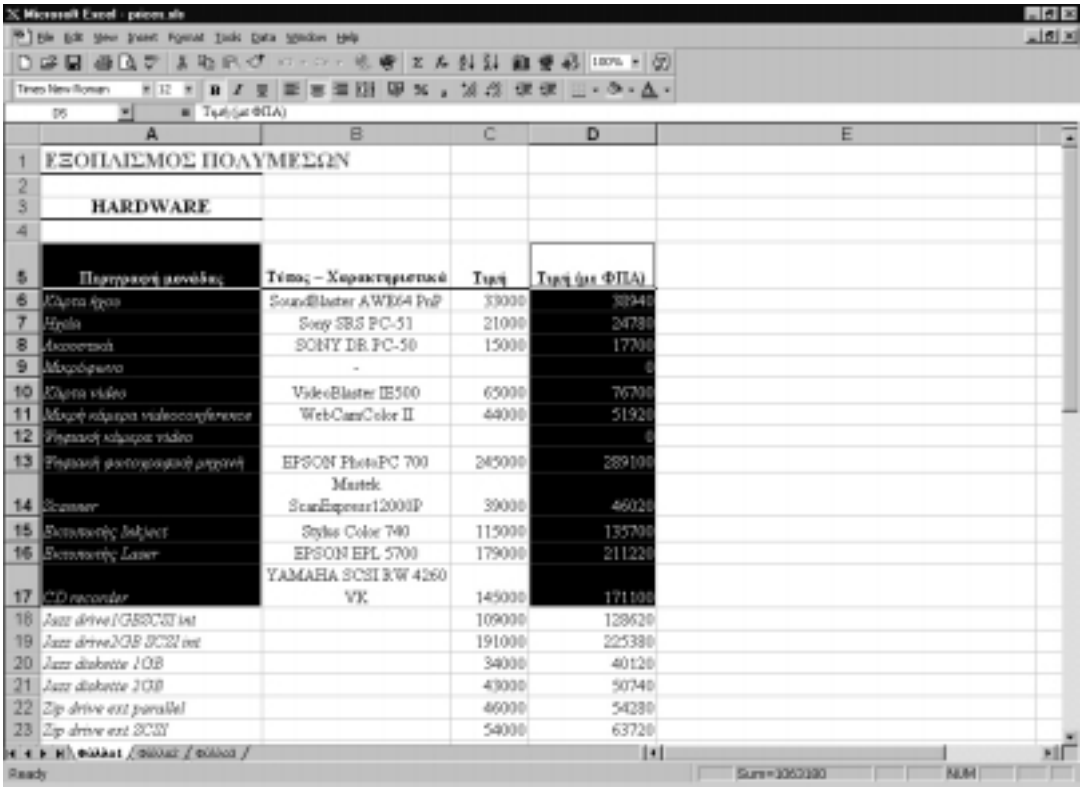
Σε μία εφαρμογή της Visual Basic όταν ένα αντικείμενο της είναι ο προορισμός μιας "συνομιλίας", η Visual Basic φροντίζει να καθορίσει το άρθρο της συνομιλίας θέτοντας ανάλογη τιμή στην ιδιότητα LinkItem αυτού του αντικειμένου.

Μία DDE "συνομιλία" πολύ συχνά καλείται σύνδεσμος (link) επειδή οι δύο εφαρμογές συνδέονται μέσω των δεδομένων που ανταλλάσσουν. Υπάρχουν τρία είδη συνδέσμων, με σημείο διαφοροποίησης τον τρόπο με τον οποίο ενημερώνεται ο προορισμός όταν αλλάζουν τα δεδομένα στην πηγή:

- ✓ Αυτόματος σύνδεσμος (automatic link): Η πηγή προμηθεύει με δεδομένα τον προορισμό κάθε φορά που αλλάζουν τα δεδομένα που προσδιορίζονται από το άρθρο σύνδεσης (link item).
- ✓ Χειροκίνητος σύνδεσμος (manual link): Η πηγή παρέχει πληροφορίες για τις τυχόν αλλαγές που έχουν επέλθει στα δεδομένα του άρθρου μόνο όταν το ζητήσει ο προορισμός.
- ✓ Σύνδεσμος γνωστοποίησης (notify link): Η πηγή γνωστοποιεί στον προορισμό ότι τα δεδομένα άλλαξαν, αλλά παρέχει αυτές τις αλλαγές των δεδομένων μόνο όταν της το ζητήσει ο προορισμός.

Ακριβώς αυτή η δυνατότητα που παρουσιάζει η Visual Basic, για δημιουργία συνδέσμων μεταξύ διαφορετικών εφαρμογών που υποστηρίζουν διαδικασίες DDE, προσδίδει δυνατότητες πλοήγησης και χαρακτηριστικά Hypermedia σε καλά οργανωμένες και δομημένες εφαρμογές της.

Έστω ότι θέλουμε να δημιουργήσουμε έναν σύνδεσμο ανάμεσα σε μία εφαρμογή της Visual Basic και σε ένα βιβλίο εργασίας του Excel από το οποίο παίρνει στοιχεία και ενημερώνεται για τις τιμές υλικού και λογισμικού πολυμέσων. Έχουμε ήδη ανοίξει και τις δύο εφαρμογές μας (της Visual Basic και του Excel), και βρισκόμαστε στην εφαρμογή του Excel, στο βιβλίο εργασίας prices.xls. Επιλέγουμε εκείνα τα δεδομένα που θέλουμε να συνδέσουμε με την εφαρμογή μας της Visual Basic (Σχήμα 11.10).



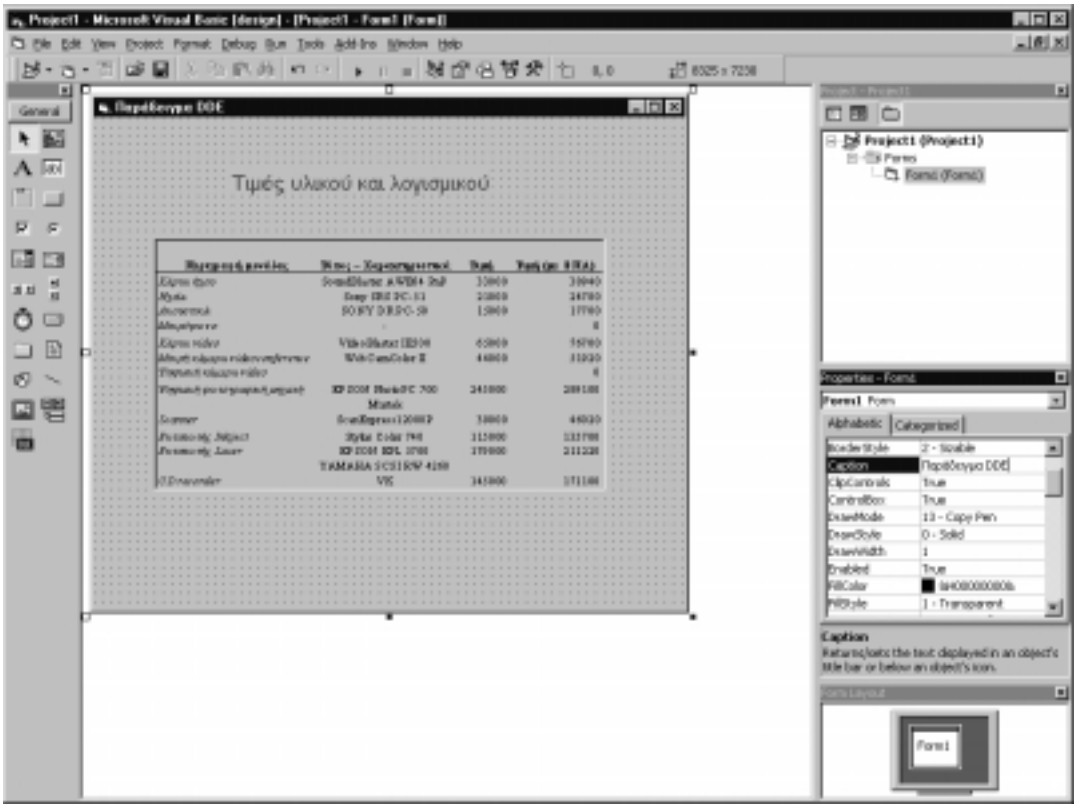
	A	B	C	D	E
1	ΕΞΟΠΛΙΣΜΟΣ ΠΟΛΥΜΕΣΩΝ				
2					
3	HARDWARE				
4					
5	Περιγραφή μενού	Τύπος – Χαρακτηριστικά	Τιμή	Τιμή (με ΦΠΑ)	
6	Κάρτα ήχου	SoundBlaster AWE64 PnP	33000	38940	
7	Ηχεία	Sony SRS PC-51	21000	24780	
8	Ακουστικά	SONY DR PC-50	15000	17700	
9	Μικρόφωνο	-		0	
10	Κάρτα video	VideoBlaster IE500	65000	76700	
11	Μικρή κάμερα videoconference	WebCamColor II	44000	51920	
12	Ψηφιακή κάμερα video			0	
13	Ψηφιακή φωτογραφική μηχανή	EPSON PhotoPC 700	245000	289100	
14	Scanner	Mustek ScanExpress12000P	39000	46020	
15	Εκτυπωτής Inkjet	Stylus Color 740	115000	135700	
16	Εκτυπωτής Laser	EPSON EPL 5700	179000	211220	
17	CD recorder	YAMAHA SCSI RW 4260 VK	145000	171100	
18	Jazz drive 1GB SCSI int		109000	128620	
19	Jazz drive 2GB SCSI int		191000	225380	
20	Jazz diskette 1GB		34000	40120	
21	Jazz diskette 2GB		43000	50740	
22	Zip drive ext parallel		46000	54280	
23	Zip drive ext SCSI		54000	63720	

Σχ. 11.10*. Επιλογή δεδομένων από την εφαρμογή του Excel*

Στη συνέχεια από τη γραμμή μενού του Excel επιλέγουμε Edit, μετά Copy. Στη συνέχεια επιστρέφουμε στην εφαρμογή μας της Visual Basic χωρίς να κλείσουμε την εφαρμογή στο Excel. Επιλέγουμε ένα εργαλείο, για παράδειγμα ένα πλαίσιο εικόνας, το οποίο μπορεί να χρησιμοποιηθεί σαν αυτό που θα λάβει τα δεδομένα και σχηματίζουμε το αντίστοιχο αντικείμενο. Από τη γραμμή μενού της Visual Basic επιλέγουμε Edit και μετά Paste Link. Εάν ο σύνδεσμος έχει πραγματοποιηθεί με επιτυχία, τα δεδομένα που είχαμε επιλέξει από το Excel, αντιγράφονται στην εφαρμογή μας της Visual Basic (Σχήμα 11.11).

Είναι αξιοσημείωτο πάντως, πως παρ' όλο που στην εφαρμογή μας στο Excel επιλέξαμε δύο ασυνεχή τμήματα, στην εφαρμογή μας της Visual Basic αντιγράφτηκε ένα ενιαίο τμήμα, στο οποίο συμπεριλαμβάνεται και το ενδιάμεσο μεταξύ των δύο επιλεγμένων τμημάτων τμήμα, που δεν είχαμε επιλέξει.

Παρατηρούμε τις τιμές που έχουν πάρει πλέον οι ιδιότητες του πλαισίου εικόνας το οποίο χρησιμοποιήσαμε για να πάρουμε τα δεδομένα από το Excel. Στην επιλογή LinkItem έχουν παρουσιαστεί οι συντεταγμένες που προσδιορίζουν το τμήμα του λογιστικού φύλλου που συνδέθηκε με την εφαρμογή μας της Visual Basic. Ακόμα, πάντα στο πλαίσιο ιδιοτήτων, στην επιλογή LinkMode εμφανίζεται η τιμή Automatic, στην επιλογή LinkTopic εμφανίζεται μαζί με την πλήρη διαδρομή και το όνομα της εφαρμογής του Excel και τέλος στην επιλογή Picture εμφανίζεται ο χαρακτηρισμός Metafile, που φανερώνει ακριβώς τη μορφή με την οποία απεικονίζονται τα δεδομένα από το Excel, στο πλαίσιο εικόνας που δημιουργήσαμε στην εφαρμογή της Visual Basic.



Σχήμα 11.11. *Μεταφορά των δεδομένων του Excel στην εφαρμογή της Visual Basic*

Έχοντας πάντοτε ανοικτές και τις δύο εφαρμογές, της Visual Basic και του Excel, μπορούμε να κάνουμε κάποιες αλλαγές στο Excel και να διαπιστώσουμε, γυρνώντας αμέσως μετά στην εφαρμογή της Visual Basic, ότι οι αλλαγές αυτές μεταφέρονται και σε στην εφαρμογή μας της Visual Basic.

Πειραματιστείτε αλλάζοντας στην εφαρμογή του Excel την τιμή κάποιων από τα υλικά που έχουμε απεικονίσει και στην εφαρμογή της Visual Basic και διαπιστώστε μόνοι σας ότι οι αλλαγές αυτές αμέσως στη συνέχεια αντικατοπτρίζονται και στην εφαρμογή της Visual Basic.

Η υλοποίηση αυτού του παραδείγματος δεν παρουσιάζεται και σε περιβάλλον Delphi, γιατί η τεχνική δημιουργίας DDE συνδέσεων κατά το σχεδιασμό μιας Delphi εφαρμογής με κάποια άλλη εφαρμογή στο περιβάλλον των Windows πραγματοποιείται επίσης μέσω του Clipboard. Στο περιβάλλον προγραμματισμού Delphi για την DDE συνομιλία χρειαζόμαστε τα DDEClientConv και DDEClientItem components.

11.3. Συμβουλές - υποδείξεις

- Κατά, το σχεδιασμό και την υλοποίηση των εφαρμογών είναι σκόπιμο να εκμεταλλευτείς όλες τις δυνατότητες που σου παρέχει ένα σύγχρονο περιβάλλον. Χρησιμοποίησε τα κατάλληλα εργαλεία και δημιούργησε ένα φιλικό, ευχάριστο και δη-

μιουργικό περιβάλλον εργασίας. Ακόμη μάθε να αξιοποιείς τις δυνατότητες που σου παρέχουν οι καινούργιες τεχνικές προγραμματισμού όπως ο αντικειμενοστραφής και οδηγούμενος από γεγονότα προγραμματισμός.

- Δημιούργησε ομοιόμορφες και εύχρηστες εφαρμογές. Φρόντισε ο τρόπος επικοινωνίας των εφαρμογών σου με το χρήστη να ακολουθεί τις γενικές αρχές εφαρμογών που γνωρίζουν οι χρήστες από εφαρμογές που ήδη χρησιμοποιούν (επεξεργαστές κειμένου, λογιστικά φύλλα κ.ά). Χρησιμοποίησε στις εφαρμογές σου μενού εντολών, κλασικά πλαίσια διαλόγου, εργαλειοθήκες και ότι νομίζεις ότι θα διευκολύνει το χρήστη.

- Κάθε εφαρμογή σου, κατά το χρόνο εκτέλεσης, πρέπει να συνοδεύεται από αρχεία βοήθειας και τεχνικές καθοδήγησης του χρήστη. Εμφάνιζε συχνά όπου είναι εφικτό επεξηγηματικά πλαίσια (tooltips), γραμμές εργαλείων (toolbars), πτυσσόμενα μενού επιλογών (popup menus) και γραμμές κατάστασης (status lines).

- Σε μια εφαρμογή μπορείς να χρησιμοποιήσεις μια ή περισσότερες φόρμες. Κάθε φόρμα αποτελεί ένα ξεχωριστό παράθυρο επικοινωνίας της εφαρμογής. Κατά το χρόνο εκτέλεσης όταν κλείσεις και την τελευταία φόρμα τερματίζεται η εκτέλεση της εφαρμογής. Για να αποφύγεις τη πολυπλοκότητα μιας εφαρμογής, μη χρησιμοποιείς πολλές φόρμες, εκτός αν είναι απαραίτητο.

- Ένα γεγονός μπορεί να προκληθεί και από το σύστημα. Στο παράδειγμα του αεροπλάνου η προσγείωση πραγματοποιείται με τη χρήση ενός γεγονότος που προκαλείται από το ρολόι του συστήματος. Εκμεταλλεύσου τα συμβάντα του συστήματος ή τις δυνατότητες του κώδικα για την αυτοματοποίηση διαδικασιών.

- Κάθε εξωτερικό μήνυμα είναι δυνατό να προκαλέσει περισσότερα του ενός γεγονότων στην εφαρμογή. Όταν ο χρήστης πατήσει ένα πλήκτρο από το πληκτρολόγιο καθώς το σημείο εισαγωγής βρίσκεται μέσα σε ένα πλαίσιο κειμένου τότε διαδοχικά προκαλούνται τα γεγονότα πάτημα πλήκτρου (keypress) και αλλαγή (change).

- Ένα γεγονός είναι δυνατό να προκληθεί και μέσα από εντολές κώδικα. Για παράδειγμα σε ένα αντικείμενο πλήκτρο εντολής της εφαρμογής μπορούμε να συμπεριλάβουμε μια εντολή με την οποία όταν ο χρήστης πατήσει με το αριστερό πλήκτρο του ποντικιού (γεγονός Κλικ) το πλήκτρο να αλλάζει το περιεχόμενο ενός πλαισίου κειμένου. Για να αντικαταστήσουμε κατά την εκτέλεση της εφαρμογής, το περιεχόμενο του πλαισίου κειμένου πρέπει να αποδώσουμε καινούργια τιμή στην ιδιότητά του **κείμενο**. Η διαδικασία γεγονότος θα έχει τη μορφή :

```
ΔΙΑΔΙΚΑΣΙΑ Πλήκτρο-Εντολής1_Κλικ
        Πλαίσιο-Κειμένου1.Κείμενο = "Καινούριο κείμενο"
ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ
```

Μετά την εκτέλεση της παραπάνω εντολής η μετατροπή του περιεχομένου του πλαισίου κειμένου, θα προκαλέσει διαδοχικά το γεγονός Αλλαγή του πλαισίου κειμένου το οποίο θα οδηγήσει στην εκτέλεση της επόμενης διαδικασίας γεγονότος:

```
ΔΙΑΔΙΚΑΣΙΑ Πλαίσιο-Κειμένου1_Αλλαγή
        ...
        Εντολές κώδικα
        ...
ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ
```

- Τα ειδικά αντικείμενα, δεν έχουν γραφικό ενδιάμεσο και συνήθως εκφράζουν εργαλεία του συστήματος. Ως ειδικά αντικείμενα στη Visual Basic μπορούμε να αναφέρουμε τον **εκτυπωτή** (Printer), το **πρόχειρο** (Clipboard), την **οθόνη** (Screen), την **εφαρμογή** (App) ή τα **αντικείμενα πρόσβασης δεδομένων** (Recordset).
- Οι επιλογές ενός μενού είναι ανεξάρτητα αντικείμενα και μπορούμε να μετατρέψουμε τα χαρακτηριστικά τους με την αλλαγή των τιμών των ιδιοτήτων τους. Έτσι για μια επιλογή μενού μπορούμε να ορίσουμε ένα αρχείο βοήθειας ή να την ενεργοποιήσουμε, να την μαρκάρουμε και να την αποκρύψουμε.
- Η εμφάνιση ενός αναδιπλωμένου μενού, γίνεται με την χρήση του δεξιού πλήκτρου του ποντικιού. Συνεπώς στο σημείο που θέλουμε να εμφανιστεί το αναδιπλωμένο μενού επιλογών πρέπει να ελέγχουμε αν ο χρήστης έχει πατήσει το δεξί πλήκτρο. Όταν η εφαρμογή ανιχνεύσει ότι ο χρήστης πάτησε και απελευθέρωσε το δεξί πλήκτρο πρέπει μέσα από την αντίστοιχη διαδικασία γεγονότος με την κατάλληλη μέθοδο να εμφανίσουμε το ανάλογο μενού. Στη Visual Basic η μέθοδο εμφάνισης αναδιπλωμένων μενού είναι η Popup.
- Ένα πλαίσιο διαλόγου μπορεί να εμφανιστεί ως **υποχρεωτικό** (modal) ή **μη υποχρεωτικό** (modeless). Τα υποχρεωτικά πλαίσια διαλόγου παραμένουν σαν ενεργά αντικείμενα της εφαρμογής μέχρι να τα κλείσει ο χρήστης. Με την τεχνική αυτή εξασφαλίζουμε ότι ο χρήστης θα αντιληφθεί και θα αποκριθεί στο μήνυμα που περιέχουν. Τα μη υποχρεωτικά πλαίσια μπορούν να μεταφερθούν ανοικτά στο παρασκήνιο της εφαρμογής και χρησιμοποιούνται κυρίως για την εμφάνιση βοηθητικών μηνυμάτων. Τα μη υποχρεωτικά πλαίσια διαλόγου δεν χρησιμοποιούνται ποτέ για την εκτέλεση κρίσιμων εργασιών. Ο ορισμός ενός προκαθορισμένου πλαισίου διαλόγου ως υποχρεωτικό γίνεται με την αντιστοίχηση των σταθερών vbApplicationModal ή vbSystemModal στο όρισμα πλήκτρα εντολής της συνάρτησης Msgbox. Όταν οριστεί το πλαίσιο διαλόγου ως SystemModal όλες οι εφαρμογές του συστήματος απενεργοποιούνται μέχρι να απαντήσει ο χρήστης στο πλαίσιο διαλόγου, ενώ όταν οριστεί ως ApplicationModal απενεργοποιείται μόνο η τρέχουσα εφαρμογή.
- Μια φόρμα της Visual Basic είναι δυνατόν να εμφανιστεί ως υποχρεωτικό πλαίσιο διαλόγου. Για να το επιτύχουνε πρέπει να συμπεριλάβουμε στην εντολή εμφάνισης της φόρμας το όρισμα 0 (μηδέν) :

```
Όνομα_φόρμας.Show 0
```

Η εμφάνιση μιας φόρμας, ως υποχρεωτικό πλαίσιο διαλόγου δεν απενεργοποιεί τις υπόλοιπες εφαρμογές του συστήματος.

- Με τη χρήση των ιδιοτήτων Height, και Width μπορούμε να δημιουργήσουμε ισομεγέθη αντικείμενα και με τη χρήση των ιδιοτήτων Left και Top μπορούμε να τα στοιχίσουμε.

11.4. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Αναφέρατε αντικείμενα από το περιβάλλον σας.

ΔΤ2. Ποιες ιδιότητες και μεθόδους υποστηρίζουν τα αντικείμενα που περιγράψατε;

ΔΤ3. Περιγράψτε τα αντικείμενα που θα χρησιμοποιούσατε για το σχεδιασμό και την υλοποίηση του τρόπου επικοινωνίας:

α) ενός συστήματος διαχείρισης της προσωπικής σου βιβλιοθήκης.

β) μιας εφαρμογής ρολόι-ξυπνητήρι.

ΔΤ4. Ποια αντικείμενα και γεγονότα θα εκμεταλλευόσασταν από τα περιβάλλοντα που περιγράψατε στην προηγούμενη δραστηριότητα για την συγγραφή του κώδικα;

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σου:

ΔΕ1. Υλοποίησε σε γραφικό περιβάλλον το πρόγραμμα, που περιγράψαμε στο κεφάλαιο 7, για τη μετατροπή της θερμοκρασίας από τη κλίμακα Fahrenheit σε κλίμακα Celsius. Υπενθυμίζεται ότι η μετατροπή της θερμοκρασίας από βαθμούς Celsius σε Fahrenheit δίνεται από τον τύπο: $F=9C/5+32$.

ΔΕ2. Στο παράδειγμα της βολής, που περιγράψαμε στο κεφάλαιο 7, υλοποίησε τη λύση της άσκησης σε γραφικό προγραμματιστικό περιβάλλον. Να επεκτείνετε το παράδειγμα με την προσθήκη και της γραφικής προσομοίωσης της βολής.

ΔΕ3. Να γράψεις πρόγραμμα με το οποίο ο χρήστης θα έχει τη δυνατότητα να επιλέγει και να εκτελεί από μια λίστα αρχείων μουσικής κάποιο μουσικό κομμάτι.

Στο σπίτι

Στο τετράδιο σας αντιμετωπίστε τα παρακάτω προβλήματα :

ΔΣ1. Περιέγραψε αντικείμενα από ένα χώρο που περνάς τον ελεύθερο χρόνο σου. Για κάθε αντικείμενο ανάφερε ιδιότητες και μεθόδους που υποστηρίζει. Ποια γεγονότα είναι δυνατόν να συμβούν σε αυτό το περιβάλλον, από εσάς και ποια από άλλες αιτίες;

ΔΣ2. Σχεδίασε ένα πρόγραμμα για την επίλυση τριωνύμου. Ο χρήστης θα πληκτρολογεί τους τρεις συντελεστές και μόλις πατήσει κάποιο πλήκτρο εντολής θα εμφανίζεται η λύση.

ΔΣ3. Δημιούργησε μια εφαρμογή για τη μετατροπή αριθμών σε διαφορετικά αριθμητικά συστήματα. Χρησιμοποίησε το οκταδικό, δεκαδικό και δεκαεξαδικό σύστημα.

ΔΣ4. Στο παράδειγμα του επεξεργαστή κειμένου (παράδειγμα 4) που έχουμε περιγράψει παραπάνω, πρόσθεσε ένα επιπλέον μενού επιλογών, με τις επιλογές του οποίου θα διαμορφώνεις το κείμενο του πλαισίου κειμένου (γραμματοσειρά, μέγεθος και χρώμα). Στη συνέχεια πρόσθεσε και μια γραμμή εργαλείων μέσω της οποίας θα εκτελούνται άμεσα οι βασικές εργασίες του μενού. Η γραμμή εργαλείων μπορεί να δημιουργηθεί με εργαλεία πλήκτρων εντολής ή εικόνας.

11.4. Τεστ αυτοαξιολόγησης

1. Δίνονται οι παρακάτω λέξεις. Ποιες από αυτές εκφράζουν αντικείμενα σύμφωνα με αυτά που αναφέρονται στο βιβλίο σου.

Α. αυτοκίνητο	Ζ. πράσινο
Β. αγάπη	Η. βιβλίο
Γ. λουλούδι	Θ. σελίδα
Δ. μεταφορά	Ι. ανάβαση
Ε. σκέψη	Κ. δίσκος

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

2. Ο δομημένος προγραμματισμός καταργείται με την εμφάνιση νέων τεχνικών προγραμματισμού.
3. Τα γεγονότα προκαλούνται μετά από κάποια ενέργεια του χρήστη.
4. Σε ένα σύγχρονο περιβάλλον εργασίας μπορούμε να επιτύχουμε τη συνεργασία διαφορετικών εφαρμογών.
5. Σε μια εφαρμογή που έχει υλοποιηθεί με την τεχνική του οδηγούμενου από τα γεγονότα προγραμματισμού υπάρχουν μόνο διαδικασίες γεγονότων.

6. Μια μέθοδος δεν είναι συγκεκριμένη για ένα αντικείμενο και είναι δυνατό να εφαρμοστεί και σε κάποιο άλλο.

7. Δίνονται οι παρακάτω προτάσεις. Βάλτε τις προτάσεις σε σωστή σειρά ώστε να περιγράφουν τα βήματα ανάπτυξης μιας εφαρμογής σε σύγχρονο προγραμματιστικό περιβάλλον.

   Α. Δημιουργία και εκσφαλμάτωση του κώδικα.

   Β. Καθορισμός της αρχικής συμπεριφοράς των αντικειμένων μέσω των ιδιοτήτων που τα χαρακτηρίζουν.

   Γ. Σχεδιασμός του τρόπου επικοινωνίας χρήστη-εφαρμογής, επιλέγοντας τα κατάλληλα αντικείμενα.

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

8. Ποια από τα παρακάτω είναι ιδιότητες:

   Α. χρώμα
   Β. αριθμός
   Γ. πλάτος
   Δ. γραμματοσειρά
   Ε. βαλβίδα
   Ζ. στοίχιση
   Η. ανανέωση
   Θ. ενεργό
   Ι. μεταβλητή
   Κ. σκάλα

9. Τα παρακάτω χαρακτηριστικά συναντώνται μόνο σε ένα αντικειμενοστραφές περιβάλλοντος ανάπτυξης :

   Α. γραφικά
   Β. πολυμορφισμός
   Γ. κλάσεις
   Δ. κληρονομικότητα
   Ε. μενού επιλογών

10. Ένα γεγονός :

    Α. προκαλείται μόνο από το σύστημα

    Β. είναι δυνατό να προκαλέσει άλλα γεγονότα

    Γ. μπορεί να μην επιδράσει σε κανένα αντικείμενο

    Δ. είναι επακόλουθο της συνεργασίας διαφορετικών εφαρμογών

    Ε. προκαλεί την εκτέλεση γενικών διαδικασιών κώδικα

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

11. Ο σχεδιασμός του τρόπου επικοινωνίας χρήστη-εφαρμογής επιτυγχάνεται με την επιλογή των κατάλληλων _____

12. Μια _____ είναι η εφαρμογή μιας ενέργειας επάνω σε ένα αντικείμενο.
13. Τα χαρακτηριστικά ενός αντικειμένου καλούνται _____
14. Κάθε αντικείμενο κληρονομεί τις ιδιότητες και της μεθόδους από την _____
15. Κατά την συνεργασία διαφορετικών εφαρμογών με την _____τα δεδομένα αποθηκεύονται στην δική μας εφαρμογή.

Συνδυάστε τα περιεχόμενα που καταγράφονται στις παρακάτω λίστες :

16. γεγονός	Α. διαγραφή
17. ιδιότητα	Β. πάτημα πλήκτρου
18. μέθοδος	Γ. λίστα
19. αντικείμενο	Δ. πλήκτρο συντομίας
20. αντικείμενο μενού	Ε. ορατό

ΚΕΦΑΛΑΙΟ 12ο

ΣΧΕΔΙΑΣΗ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΗ

12.1. Πζοσδοκώμενα αποτελέσματα

Από τη μελέτη αυτού του κεφαλαίου προσδοκάται ότι καταρχήν θα κατανοήσεις την έννοια του user interface, της διεπαφής χρήστη, όπως είναι ο ελληνικός όρος. Θα μάθεις για τους διαφορετικούς τύπους διεπαφής χρήστη, δίνοντας έμφαση στον γραφικό τύπο (GUI). Θα καταλάβεις και θα ενστερνιστείς βασικούς κανόνες της εργονομίας λογισμικού. Θα μάθεις γενικές αρχές σχεδίασης διεπαφής χρήστη, αλλά και ειδικές αρχές που αναφέρονται στην οπτική και ηχητική του σχεδίαση. Οι γνώσεις αυτές θα σε βοηθήσουν ώστε να αναπτύξεις ο ίδιος δεξιότητες σχεδιασμού περιβάλλοντος διεπαφής, που να περιλαμβάνει όλα εκείνα τα στοιχεία που θα το χαρακτηρίζουν απλό, φιλικό, εργονομικό και εύχρηστο.

12.2. Συμβουλές - υποδείξεις

Το αποτέλεσμα της προσπάθειάς σου για υλοποίηση μιας εφαρμογής εξαρτάται σε μεγάλο βαθμό από τη σχεδίαση της διεπαφής χρήστη. Μια εφαρμογή που επιτελεί άψογα όλες τις λειτουργίες για τις οποίες σχεδιάστηκε, εν τούτοις μπορεί να χαρακτηριστεί αποτυχημένη και να μην έχει καμία χρηστικότητα, αν ο τρόπος με τον οποίο έρχεται σε επικοινωνία με το χρήστη δεν είναι σωστά σχεδιασμένος. Επομένως πριν αρχίσεις την προσπάθεια υλοποίησης ενός προγράμματος, θα πρέπει να ασχοληθείς επισταμένως με τη σχεδίαση της διεπαφής του χρήστη, έτσι ώστε να είσαι απολύτως βέβαιος ότι θα έχεις εξασφαλίσει τον καλύτερο τρόπο συνεργασίας του προγράμματος σου με το χρήστη. Έτσι θα αυξήσεις και στο μεγαλύτερο βαθμό την απόδοση του

προγράμματός σου.

Ο σημερινός κόσμος των υπολογιστών και της πληροφορικής χαρακτηρίζεται έντονα από την παρουσία των πολυμέσων. Εντυπωσιακά γραφικά, animations, ήχοι συνθέτουν ένα περιβάλλον διεπαφής ιδιαίτερα ελκυστικό. Όμως θα πρέπει να πιστέψεις ότι κάθε τι αισθητικά ωραίο δεν είναι απαραίτητα και λειτουργικά ωφέλιμο. Αντίθετα μάλιστα, πολλές φορές η παρουσία πολυμεσικών μορφών δεδομένων στην εφαρμογή, λειτουργεί σε βάρος της αποτελεσματικότητάς της. Ο χρήστης μπορεί να μαγνητίζεται από την παρουσία τους, η προσοχή του να παρασύρεται από την ελκυστική μορφή τους και το περιεχόμενο της εφαρμογής να περνάει σε δεύτερη μοίρα, χάνοντας έτσι ένα μικρό ή μεγάλο μέρος της λειτουργικότητάς της.

Ένας κανόνας που θα πρέπει να ακολουθείς στη χρησιμοποίηση πολυμεσικών μορφών δεδομένων στα προγράμματά σου είναι *"ότι είναι εξαιρετικά ωραίο, δεν είναι απαραίτητα και εξαιρετικά χρήσιμο"*.

12.3. Δραστηριότητες - ασκήσεις

Στην τάξη

ΔΤ1. Σε μία αλληλεπιδραστική εκπαιδευτική εφαρμογή ιστορίας, ο χρήστης μπορεί να πλοηγείται και να αναζητά διάφορα ευρήματα διαφορετικών ιστορικών εποχών σε διαφορετικούς γεωγραφικούς χώρους. Ποια μεταφορά βρίσκετε κατάλληλη για να χαρακτηρίσει το χρήστη, με άλλα λόγια ποιο ρόλο θα μπορούσατε να του αναθέσετε, και ποια μεταφορά θα χρησιμοποιούσατε για την οπτική απεικόνιση του δείκτη του ποντικιού;

ΔΤ2. Να προτείνετε περιπτώσεις τύπων εφαρμογών στις οποίες θεωρείτε ότι η παρουσία ήχου είναι επιβεβλημένη. Να καθορίσετε την κατάλληλη (ή τις κατάλληλες) μορφές ήχου.

Στο εργαστήριο

Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Να υλοποιήσετε την εξής εφαρμογή, της οποίας γενικά στοιχεία περιγράφονται στη συνέχεια:

Στοιχεία γραφικού περιβάλλοντος

Η εφαρμογή σας αποτελείται από 3 οθόνες.

- Στην εισαγωγική οθόνη υπάρχουν:

1. ένα πλαίσιο κειμένου με τον τίτλο του θέματος “Ρύπανση μέσα από τις τροφικές αλυσίδες”,
2. ένα πλαίσιο με 6 πλήκτρα επιλογής, καθένα από τα οποία αντιστοιχεί στα θέματα Μικροφάγα - μακροφάγα ψάρια, Παραγωγοί, Τροφική αλυσίδα, Φωτοσύνθεση, Ρύπανση, Μόλυνση,
3. ένα πλαίσιο κειμένου με τις πρώτες βασικές οδηγίες χρήσης της εφαρμογής,
4. ένα πλαίσιο εικόνας με σχετική με το θέμα εικόνα
5. τέσσερα πλήκτρα : εξόδου από την εφαρμογή, αφήγησης, απόκρυψης και εμφάνισης της παραπάνω εικόνας

Στις άλλες δύο οθόνες υπάρχουν:

1. θεματικός τίτλος,
2. επεξηγηματικό κείμενο,
3. σχετική με το θέμα εικόνα,
4. οτιδήποτε άλλο θεωρείτε απαραίτητο και σκόπιμο.

Έ&Ψουργίες

Είναι δυνατή η αμφίδρομη μετακίνηση από την αρχική οθόνη στις οθόνες που αντιστοιχούν στα θέματα Μικροφάγα - μακροφάγα ψάρια και Ρύπανση. Τα άλλα θέματα δεν θα αναπτυχθούν.

ΠαρατηρήσεΨ-οδηγίες

Χρησιμοποιήστε

- κατάλληλες χρωματικές αποχρώσεις παρασκηνίου και προσκηνίου

Στο σπίτι

ΔΣ1. Να καταγράψετε στο χαρτί τα πιο κατάλληλα κατά την άποψή σας μηνύματα λάθους που θα πρέπει να εμφανίζει ένα πρόγραμμά σας στις περιπτώσεις όπου :

α) τα δεδομένα που πληκτρολογείτε είναι αλφαριθμητικά, αντί αριθμητικά που είναι το ζητούμενο,

β) το αρχείο που επιχειρείτε να ανοίξετε είναι διαφορετικής μορφοποίησης από την ενδεδειγμένη,

γ) ενεργοποιείτε ένα φυλλομετρητή (browser) μέσα από το πρόγραμμά σας και του αναθέτετε off line να σας μεταφέρει σε κάποια ηλεκτρονική διεύθυνση.

ΔΣ2. Να καταγράψετε τις αναπαραστάσεις που σας προκαλούν τα εξής χρώματα: μαύρο, κόκκινο, πράσινο, κίτρινο, μπλε, άσπρο. Να συγκρίνετε τα αποτελέσματα της

δραστηριότητας με τα αποτελέσματα των συμμαθητών σας.

ΔΣ3. Να προετοιμάσετε σε επίπεδο σχεδίασης τη δραστηριότητα ΔΕ1.

12.4. Τεστ αυτοαξιολόγησης

Δίνονται οι παρακάτω ομάδες λέξεων. Σε κάθε μια από αυτές, να βάλεις τις λέξεις στη σωστή σειρά.

1. υλοποίηση, προσδιορισμός απαιτήσεων, έλεγχος, σχεδίαση

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει.

2. Ο _____ είναι ο περισσότερος φιλικός τύπος διεπαφής χρήστη.
3. Υποσυνείδητα για το χρήστη το χρώμα _____ σημαίνει απαγόρευση ή παύση

4. Ταίριαξε μεταξύ τους τις λέξεις από τις δύο στήλες

Προσκήνιο	**Παρασκήνιο**
Σκούρο μπλε	Μπλε
Άσπρο	Μπεζ
Μαύρο	Ανοικτό γκρι
Μπλε	Άσπρο

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος.

5. Ο χρήστης θα πρέπει να ακολουθεί τις συνήθειες του σχεδιαστή.
6. Τα μηνύματα λάθους πρέπει να έχουν επικριτικό τόνο προς το χρήστη.
7. Ο τόνος της φωνής του αφηγητή δεν παίζει κανένα ρόλο στην επίδραση της εφαρμογής στο χρήστη.

8. Διάλεξε ένα μεταξύ των προτεινόμενων.

Τα μηνύματα λάθους πρέπει να είναι:

α) γενικά

β) εξειδικευμένα

γ) αόριστα

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων.

9. Μερικές από τις βασικές αρχές που πρέπει να τηρούνται κατά τη σχεδίαση μιας λειτουργικής διεπαφής χρήστη είναι οι:

α) ελαχιστοποίηση παροχής άμεσης ανάδρασης

β) ποικιλία μορφών της ίδιας λειτουργίας

γ) ελαχιστοποίηση απομνημόνευσης

δ) διαφοροποίηση από τα συνήθη

ε) ελαχιστοποίηση ενεργειών χρήστη

10. Βασικά χαρακτηριστικά των γραφικών διεπαφών χρήστη είναι:

α) τα γραφικά αντικείμενα

β) το πληκτρολόγιο

γ) τα παράθυρα

δ) οι δείκτες του ποντικιού

ε) η επιφάνεια εργασίας

στ) τα γραφίστικα προγράμματα

13.

Εκσφαλμάτωση προγράμματος

Εισαγωγή

Ανεξάρτητα από το πόσο προσεκτικά έχει γράψει κάποιος ένα πρόγραμμα, τις περισσότερες φορές απρόβλεπτες καταστάσεις, κακός χειρισμός ή λογικά λάθη στο σχεδιασμό των προγραμμάτων, οδηγούν στην εμφάνιση λαθών. Τα λάθη που παρουσιάζονται κατά την εκτέλεση ενός προγράμματος, μερικές φορές μπορεί να είναι ασήμαντα, όπως για παράδειγμα ή εσφαλμένη στοίχιση των αποτελεσμάτων. 'Αλλες φορές όμως μπορεί να είναι ιδιαίτερα κρίσιμα, ώστε να οδηγούν στην κατάρρευση των εφαρμογών ή του συστήματος. Ένα πρόγραμμα πριν παραδοθεί για πραγματική λειτουργία, πρέπει να ελέγχεται και να είναι βέβαιο ότι εργάζεται απρόσκοπτα και παράγει σωστά αποτελέσματα. Κάθε προγραμματιστής οφείλει κατά τη φάση της υλοποίησης, να δοκιμάζει λεπτομερώς το πρόγραμμα σε διαφορετικές συνθήκες και με ποικιλία δεδομένων, να διορθώνει όποια λάθη παρουσιαστούν και να συγκρίνει τα παραγόμενα αποτελέσματα με τα αναμενόμενα, ώστε να προλαμβάνει απρόσμενες καταστάσεις λάθους, πριν εμφανιστούν σε πραγματική λειτουργία.

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι οι μαθητές:

- να αναγνωρίζουν ένα λάθος.
- να ορίζουν τις κατηγορίες λάθους.
- να αναφέρουν εργαλεία εκσφαλμάτωσης και τον τρόπο χειρισμού τους.
- να εφαρμόζουν τις τεχνικές χειρισμού λαθών κατά το χρόνο εκτέλεσης.

Προερωτήσεις

- ✓ ποια είναι τα πιθανά λάθη που μπορεί να εμφανιστούν κατά την υλοποίηση και την εκτέλεση ενός προγράμματος;
- ✓ ποια είναι τα πλεονεκτήματα της εκσφαλμάτωσης;
- ✓ υπάρχουν εργαλεία που βοηθούν τον προγραμματιστή για την εκσφαλμάτωση ενός προγράμματος;
- ✓ είναι εφικτό να χειριστούμε αναπάντεχες καταστάσεις που πιθανόν θα εμφανιστούν κατά το χρόνο εκτέλεσης του προγράμματος, έτσι ώστε να μην διακόπτεται η λειτουργία του;
- ✓ η εργασία της εκσφαλμάτωσης θεωρείται ότι είναι απαραίτητη για την υλοποίηση ενός προγράμματος;

13.1 Κατηγορίες λαθών

Ένας προγραμματιστής, ανεξάρτητα από πόσο ικανός είναι, όταν δημιουργεί ένα πρόγραμμα, είναι φυσικό να κάνει κάποιο λάθος. Ειδικά σε μεγάλες εφαρμογές που κατασκευάζει πολύπλοκες υπολογιστικές ρουτίνες ή χρησιμοποιεί αρκετές συσκευές υλικού, είναι αδύνατο να αποφύγει τέτοιες ανεπιθύμητες καταστάσεις.

Σε ένα πρόγραμμα είναι δυνατό να παρουσιαστούν διαφορετικής μορφής λάθη, τα οποία μπορούν να χωριστούν σε τρεις βασικές κατηγορίες:

Λάθη κατά την υλοποίηση

Τα λάθη κατά το χρόνο υλοποίησης, προκαλούνται κυρίως από λανθασμένη σύνταξη εντολών προγράμματος. Τέτοια λάθη μπορεί να είναι η λανθασμένη συγγραφή μιας δεσμευμένης λέξης της γλώσσας προγραμματισμού ή η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της.

Ένα λάθος που προκαλείται κατά τη συγγραφή του προγράμματος, ανιχνεύεται από το μεταγλωττιστή, ο οποίος εμφανίζει προς το προγραμματιστή κάποιο προειδοποιητικό μήνυμα. Αν το πρόγραμμα περιέχει ένα λάθος αυτής της μορφής, δεν επιτρέπεται η εκτέλεσή του, μέχρι να το διορθώσει ο προγραμματιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα μας προφυλάσσουν αυτόματα από τα λάθη κατά την υλοποίηση , αφού παρέχουν εργαλεία αυτόματου ελέγχου σύνταξης των εντολών και παρακολουθούν τον προγραμματιστή κατά τη συγγραφή του προγράμματος. Μόλις διαπιστώσουν κάποιο συντακτικό λάθος, σταματούν και απαιτούν τη διόρθωσή του. Συνήθως αντιλαμβάνονται ακριβώς το λάθος που δημιουργήθηκε και προτείνουν αναλυτικά τον τρόπο διόρθωσής του, εμφανίζοντας σε ενημερωτικό πλαίσιο την ορθή σύνταξη της εντολής που προκλήθηκε το λάθος.

Μια από τις πρώτες βλάβες που εμφανίστηκαν στους υπολογιστές ήταν ένα βραχυκύκλωμα που οφειλόνταν στον εγκλωβισμό ενός εντόμου (bug) στο εσωτερικό ενός υπολογιστή. Από το γεγονός αυτό, είναι πολύ πιθανό να προέρχεται ο όρος ***bugs****, με τον οποίο αναφερόμαστε στα λάθη που εμφανίζονται σε ένα πρόγραμμα.*

Λάθη κατά την εκτέλεση

Τα λάθη που προκαλούνται κατά το χρόνο εκτέλεσης του προγράμματος, είναι πιο επώδυνα γιατί συνήθως εμφανίζονται σε πραγματικό περιβάλλον εκτέλεσης και τις περισσότερες φορές προκαλούν τον αντικανονικό τερματισμό της εφαρμογής και το *κρέμασμα* (crash) του συστήματος.

Όταν ένα λάθος προκληθεί κατά την εκτέλεση της εφαρμογής, είναι δυνατό να αντιμετωπισθεί μόνο με τη χρήση εντολών προγράμματος που το παγιδεύουν και εκτελούν τις κατάλληλες διαδικασίες χειρισμού του.

Η πρόληψη τέτοιων λαθών είναι αρκετά δύσκολη, αφού συνήθως οφείλονται σε καταστάσεις που δεν είναι εύκολο να ελεγχθούν από τον προγραμματιστή, ενώ πολλές φορές εμφανίζονται μετά από μεγάλο χρονικό διάστημα. Τέτοια λάθη είναι δυνατό να προκληθούν από την κλήση μιας διαδικασίας με δεδομένα που δεν μπορεί να χειριστεί, όπως η αναζήτηση διαγραμμένων αρχείων, η προσπάθεια διαίρεσης ενός αριθμού με το μηδέν, η υπερχείλιση μιας αριθμητικής μεταβλητής ή από δυσλειτουργία του υλικού μέρους του υπολογιστή, όπως η καταστροφή του σκληρού δίσκου του συστήματος, ο τερματισμός μιας σύνδεσης δικτύου και η αποσύνδεση του εκτυπωτή.

Λογικά λάθη

Τα λογικά λάθη είναι συνήθως λάθη σχεδιασμού και δεν προκαλούν τη διακοπή της εκτέλεσης του προγράμματος. Ενώ ο μεταγλωττιστής της γλώσσας προγραμματισμού δεν ανιχνεύει κανένα συντακτικό λάθος και κατά την εκτέλεση του προγράμματος δεν παρουσιάζονται ανεπιθύμητες καταστάσεις σφαλμάτων, τελικά δεν παράγονται τα επιθυμητά αποτελέσματα.

Η ανίχνευση τέτοιων λαθών δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή και διαπιστώνονται μόνο με τη *διαδικασία ελέγχου* (testing) και την ανάλυση των αποτελεσμάτων των προγραμμάτων.

Το πιο δημοφιλές λάθος που παρουσιάστηκε στην ιστορία των υπολογιστών είναι το **πρόβλημα του έτους 2000** (millennium bug). Το πρόβλημα αυτό είναι ιδιόμορφο, γιατί οφείλεται σε συνδυασμένη προβληματική λειτουργία του λογισμικού και του υλικού μέρους του υπολογιστή και ακόμη απασχόλησε χρονικά την κοινωνία μας αρκετά πριν από την ουσιαστική εμφάνιση των συνεπειών του.

Άλλα σημαντικά προβλήματα που παρουσιάστηκαν στην ιστορία των υπολογιστών και αποδόθηκαν στη δυσλειτουργία του λογισμικού είναι :

- Το 1962 το διαστημικό όχημα Mariner 1 εκτοξεύτηκε από το ακρωτήρι Canaveral των Ηνωμένων Πολιτειών της Αμερικής για τον πλανήτη Αφροδίτη. Μετά την απογείωση ο πύραυλος που κατεύθυνε το διαστημικό όχημα έχασε την κατεύθυνσή του και οι τεχνικοί της διαστημικής υπηρεσίας των Η.Π.Α. (NASA) αναγκάστηκαν να τον ανατινάξουν, πριν αυτός καταστραφεί σε κάποιο σημείο της γης και κινδυ-

νεύσουν ανθρώπινες ζωές. Ευτυχώς το διαστημικό όχημα δεν είχε ανθρώπινο πλήρωμα. Η αναφορά της NASA απέδωσε το ατύχημα στην εσφαλμένη αντικατάσταση ενός θετικού πρόσημου με αρνητικό σε μια εντολή FORTRAN του λογισμικού πλοήγησης του πυραύλου. Το εσφαλμένο πρόσημο στοίχισε περίπου 80 εκατομμύρια δολάρια.

- Το 1990 ένα λάθος σε μια εντολή του λογισμικού των νέων συστημάτων δρομολόγησης τηλεφωνικών κλήσεων της εταιρείας τηλεπικοινωνιών AT&T, προκάλεσε το μπλοκάρισμα του μεγαλύτερου τμήματος του τηλεφωνικού δικτύου των Ηνωμένων Πολιτειών της Αμερικής. Περίπου 5 εκατομμύρια τηλεφωνικές γραμμές διακόπηκαν για εννέα ώρες.
- Ο επεξεργαστής Pentium το έτος 1994 παρουσίασε δυσλειτουργία, αφού σε σπάνιες περιπτώσεις έδινε εσφαλμένες απαντήσεις σε πολύπλοκες μαθηματικές εξισώσεις. Το πρόβλημα ανακαλύφτηκε από τον καθηγητή Thomas Nicely στο Πανεπιστήμιο Lynchburg στη Virginia των Ηνωμένων Πολιτειών της Αμερικής. Αρχικά η κατασκευάστρια εταιρεία αρνήθηκε την ύπαρξή του, αλλά στη συνέχεια υποχρεώθηκε να αντικαταστήσει τους προβληματικούς επεξεργαστές και σύμφωνα με εκτιμήσεις δαπάνησε γι αυτό το σκοπό περίπου 450 εκατομμύρια δολάρια
- Το έτος 1995 επρόκειτο να γίνουν τα εγκαίνια του νέου διεθνούς αεροδρομίου στο Denver των Ηνωμένων Πολιτειών της Αμερικής. Το αεροδρόμιο είχε κατασκευαστεί με σύγχρονη τεχνολογία και διέθετε ένα αυτόματο σύστημα μεταφοράς αποσκευών. Με την έναρξη λειτουργίας του το σύστημα μεταφοράς αποσκευών παρουσίασε λειτουργικά προβλήματα, με συνέπεια την καταστροφή αποσκευών επιβατών και του ιδίου του συστήματος. Το αεροδρόμιο διέκοψε τη λειτουργία του και επαναλειτούργησε δεκαέξι μήνες αργότερα και με κλασικό σύστημα μεταφοράς αποσκευών, επιβαρύνοντας τον προϋπολογισμό κατασκευής του κατά 3,2 εκατομμύρια δολάρια.

13.2 Εκσφαλμάτωση

Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος καλείται *εκσφαλμάτωση* (debugging). Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.

Η εργασία της εκσφαλμάτωσης δεν είναι εύκολη, απαιτεί βαθιά γνώση της γλώσσας προγραμματισμού και φυσικά αντίστοιχες ικανότητες από τον προγραμματιστή. Για τον εντοπισμό ενός λάθους δεν υπάρχουν ιδιαίτερα μυστικά και τρυκ. Η εκσφαλμάτωση είναι ένα πρόβλημα λογικής και όσο πιο καλά αντιλαμβάνεται ο προγραμματιστής τον τρόπο που εργάζεται το πρόγραμμα, τόσο πιο εύκολα και σύντομα θα εντοπίσει λάθη που προκαλούν δυσλειτουργίες.

Η εισαγωγή γραμμών με σχόλια σε ένα πρόγραμμα υποβοηθά σημαντικά την εκσφαλμάτωση.

Σε ένα σύγχρονο προγραμματιστικό περιβάλλον δεν χρειάζεται ιδιαίτερα μνεία για τα λάθη που παρουσιάζονται κατά το χρόνο σχεδιασμού, αφού αυτά, όπως αναφέρθηκε, είναι συντακτικά λάθη και τις περισσότερες φορές το περιβάλλον προγραμματισμού τα ανιχνεύει αυτόματα και προτείνει τη διόρθωσή τους. Ακόμη και αν το περιβάλλον δεν προτείνει τη διόρθωση, ο μεταγλωττιστής συλλαμβάνει και περιγράφει το λάθος και στη συνέχεια ο προγραμματιστής μπορεί πολύ εύκολα να το διορθώσει.

Τα λάθη που κυρίως μας απασχολούν στη φάση της εκσφαλμάτωσης είναι τα λογικά λάθη και τα λάθη που παρουσιάζονται κατά το χρόνο εκτέλεσης του προγράμματος. Η εκσφαλμάτωση τέτοιων λαθών μπορεί να γίνει μέσα από εργαλεία εκσφαλμάτωσης ή από ειδικές εντολές ή συναρτήσεις που προσφέρει το περιβάλλον προγραμματισμού.

13.3. Εργαλεία εκσφαλμάτωσης

Τα ονόματα των μεταβλητών πρέπει να ανάγουν στο περιεχόμενό τους. Έτσι διευκολύνεται η εκσφαλμάτωση.

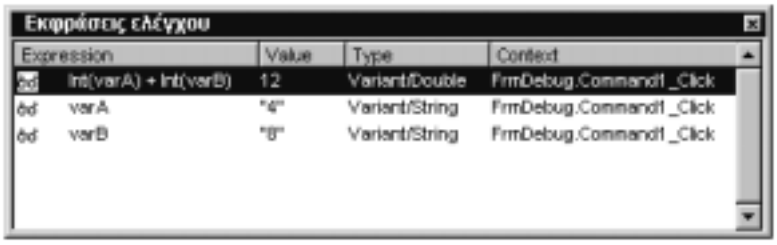
Πολλές φορές ο εντοπισμός ενός σφάλματος είναι ιδιαίτερα δύσκολος και για να επιτευχθεί χρειάζεται συστηματική παρακολούθηση και ανάλυση των δεδομένων του προγράμματος. Τα σύγχρονα εργαλεία προγραμματισμού δίνουν τη δυνατότητα στον προγραμματιστή να παρακολουθεί, τι συμβαίνει στο παρασκήνιο του προγράμματος κατά το χρόνο εκτέλεσης. Αυτό επιτυγχάνεται με τη χρήση κατάλληλων εργαλείων, που παρέχουν πολλές δυνατότητες ελέγχου κατά τη δοκιμαστική εκτέλεση ενός προγράμματος. Τα περισσότερα εργαλεία προγραμματισμού συνοδεύονται από *προγράμματα διόρθωσης* (debuggers).

Με τα εργαλεία εκσφαλμάτωσης είναι δυνατό να εμφανίζεται η τιμή μιας μεταβλητής ή μιας έκφρασης, να γίνεται δυναμική επέμβαση σε κάποιο σημείο δίνοντας διαφορετική τιμή σε μια μεταβλητή ή να εκτελείται βήμα προς βήμα το πρόγραμμα για τον εντοπισμό της εντολής που προκαλεί το σφάλμα.

Στη συνέχεια θα αναφέρουμε τις βασικές λειτουργίες εκσφαλμάτωσης που προσφέρουν τα περισσότερα προγραμματιστικά περιβάλλοντα :

Εκφράσεις ελέγχου

Οι *εκφράσεις ελέγχου* (watch expressions) είναι χρήσιμες για την άμεση παρατήρηση τιμών μεταβλητών ή εκφράσεων κατά την εκτέλεση ενός προγράμματος. Ακόμη με μια έκφραση ελέγχου, υπάρχει η δυνατότητα να διακοπεί η εκτέλεση ενός προγράμματος, όταν αλλάξει το περιεχόμενο μιας μεταβλητής ή δεχτεί συγκεκριμένη τιμή. Με τις εκφράσεις ελέγχου παρατηρούμε τη συμπεριφορά μιας μεταβλητής ή μιας έκφρασης σε όλη τη διάρκεια εκτέλεσης του προγράμματος.



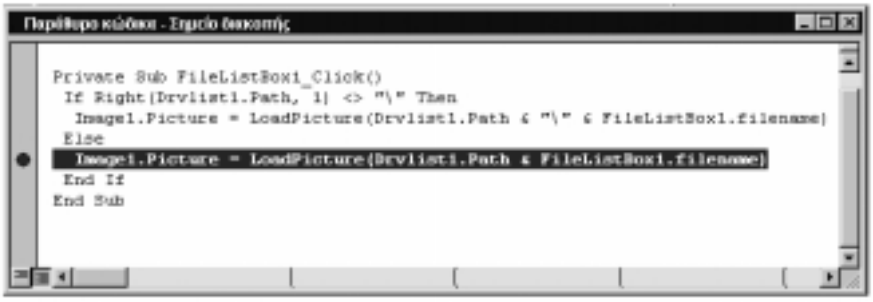
***Σχ. 13.1.** Εμφάνιση εκφράσεων ελέγχου*

Σημεία διακοπής

Ένα *σημείο διακοπής* (breakpoint) διακόπτει την εκτέλεση ενός προγράμματος ακριβώς πριν από μια συγκεκριμένη εντολή. Τα σημεία διακοπής χρησιμοποιούνται για τον έλεγχο της κατάστασης των δεδομένων σε συγκεκριμένο σημείο του προγράμματος. Ορίζονται στον πηγαίο κώδικα με το μαρκάρισμα κάθε εντολής, στην οποία θέλουμε να διακόψουμε την εκτέλεση του προγράμματος.

Βήμα προς βήμα εκτέλεση

Μετά τη διακοπή εκτέλεσης από ένα σημείο διακοπής, παρέχεται η δυνατότητα να συνεχιστεί βήμα προς βήμα η εκτέλεση του προγράμματος. Με τη λειτουργία αυτή μπορούμε να εξετάσουμε βηματικά τα αποτελέσματα κάθε εντολής ή ρουτίνας του προγράμματος.



***Σχ. 13.2.** Σημείο διακοπής στον κώδικα της εφαρμογής*

Ιστορικό

Με αυτή τη λειτουργία κατακρατείται *ιστορικό* (history) των τελευταίων εντολών που εκτελέστηκαν. Διαβάζοντας το ιστορικό αυτό μπορούμε να ελέγξουμε τη ροή του προγράμματος βαδίζοντας μπρος ή πίσω μέσα στον κώδικα.

Ιχνηλάτηση

Δίνει τη δυνατότητα αργής εκτέλεσης του προγράμματος, ενώ παράλληλα εμφανίζεται φωτισμένη η εντολή που εκτελείται κάθε φορά. Η *ιχνηλάτηση* (tracing) ενός προγράμματος ξεκινά από ένα σημείο διακοπής, που έχει εισαχθεί στο πρόγραμμα και λειτουργεί σε επίπεδο εντολής εκτελώντας κάθε φορά μια εντολή ή σε επίπεδο διαδικασίας εκτελώντας κάθε φορά όλη τη διαδικασία.

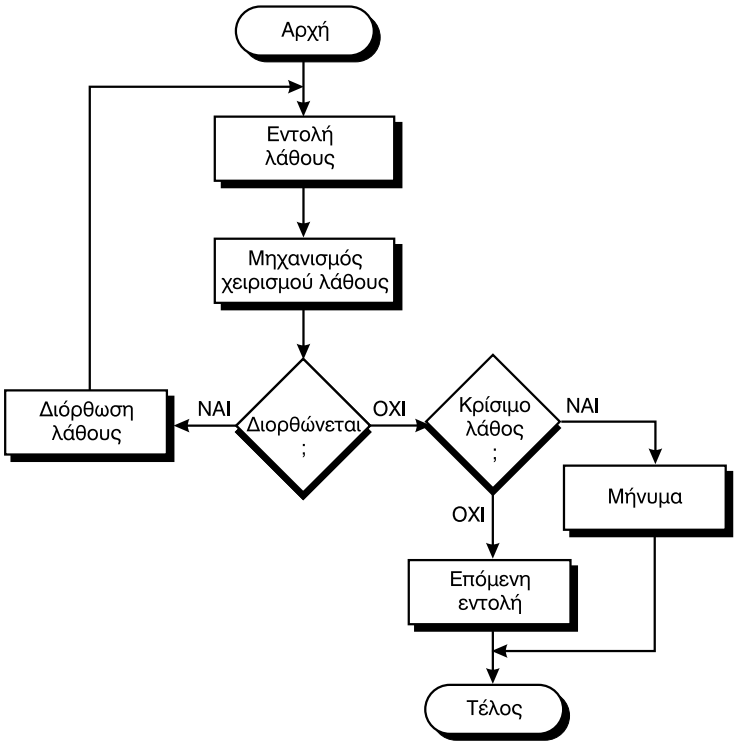
Λειτουργία άμεσης εκτέλεσης

Με τη λειτουργία *άμεσης εκτέλεσης* έχουμε τη δυνατότητα σε κάποιο σημείο διακοπής να πληκτρολογήσουμε και να εκτελέσουμε άμεσα οποιαδήποτε εντολή με σκοπό τη λογική αλλαγή της ροής εκτέλεσης του προγράμματος.

***Σχ. 13.3.** Εμφάνιση των περιεχομένων των εκφράσεων σε λειτουργία άμεσης εκτέλεσης*

Οι λειτουργίες των εργαλείων εκσφαλμάτωσης σε γραφικά περιβάλλοντα πραγματοποιούνται μέσα από ειδικά παράθυρα (π.χ. παράθυρο ελέγχου, παράθυρο άμεσης εκτέλεσης), ενώ στα υπόλοιπα εργαλεία από επιλογές μενού επιλογών που προσφέρει το περιβάλλον ανάπτυξης ή με ειδικές εντολές στον πηγαίο κώδικα.

Ένας προγραμματιστής που δεν είναι εξοικειωμένος και δεν γνωρίζει το χειρισμό των εργαλείων εκσφαλμάτωσης ή αν εργάζεται σε ένα περιβάλλον που δεν του παρέχει αντίστοιχα εργαλεία, μπορεί να υλοποιήσει αρκετές από τις λειτουργίες εκσφαλμάτωσης που αναφέρθηκαν, με την παρεμβολή εντολών εμφάνισης ή εκτύπωσης μηνυμάτων, τιμών μεταβλητών ή εκφράσεων.



13.4 Χειρισμός λαθών κατά το χρόνο εκτέλεσης

Τα λάθη που προκαλούνται κατά την εκτέλεση ενός προγράμματος μπορούν να προκληθούν από πάρα πολλές αιτίες. Ο προγραμματιστής είναι υποχρεωμένος μέσα από το πρόγραμμα του να προβλέψει κάθε πιθανό λάθος που μπορεί να συμβεί στο περιβάλλον εκτέλεσης, ώστε να το αντιμετωπίσει και να αποφύγει δυσάρεστα αποτελέσματα για το χρήστη.

Κάθε λάθος παράγει έναν κωδικό λάθους. Συνήθως ο προγραμματιστής δημιουργεί γενικές διαδικασίες χειρισμού λαθών, που τις εντάσσει σε βιβλιοθήκες.

Η διαδικασία χειρισμού λαθών κατά το χρόνο εκτέλεσης είναι απλή. Η εκτέλεση κάθε εντολής προγράμματος επιστρέφει μια ένδειξη που περιγράφει, αν ολοκληρώθηκε επιτυχώς ή παρουσιάστηκε κάποιο λάθος. Όταν παρουσιαστεί ένα λάθος, ο προγραμματιστής πρέπει να εκμεταλλευτεί αυτή την ένδειξη και μέσα από εντολές κώδικα να δώσει δυνατότητες διαφυγής στο πρόγραμμα.

Πιο αναλυτικά ο χειρισμός ενός λάθους κατά το χρόνο εκτέλεσης περιγράφεται ως εξής:

Στην ενότητα που θέλουμε να προσθέσουμε δυνατότητες χειρισμού λαθών, χρησιμοποιούμε τον αντίστοιχο μηχανισμό ανίχνευσης λαθών που προσφέρει το περιβάλλον προγραμματισμού.

Όταν προκληθεί ένα λάθος, ο μηχανισμός ανίχνευσης μεταφέρει τη ροή εκτέλεσης του προγράμματος σε ένα τμήμα κώδικα χειρισμού λάθους (exception ή error handler), στο οποίο περιγράφει το λάθος τις περισσότερες φορές με κάποιο αριθμό. Το συγκεκριμένο τμήμα κώδικα συνήθως βρίσκεται μέσα στην ίδια ενότητα που προκαλείται το λάθος.

Για την ανίχνευση και το χειρισμό των λαθών που εμφανίζονται κατά την εκτέλεση ενός προγράμματος στις γλώσσες προγραμματισμού Java, ADA και C++ χρησιμοποιείται ο μηχανισμός των εξαιρέσεων (exceptions), ενώ στη Visual Basic υπάρχει ειδική εντολή (On Error GoTo).

Η ρουτίνα χειρισμού του λάθους είναι διαφορετική κάθε φορά και δημιουργείται από τον προγραμματιστή ειδικά για το χειρισμό των λαθών που μπορεί να προκύψουν από την εργασία που εκτελείται εκείνη τη χρονική στιγμή, με στόχο φυσικά την ανώδυνη απεμπλοκή του προγράμματος από την ενδεχόμενη προβληματική κατάσταση.

Στη συνέχεια περιγράφουμε ορισμένες ενέργειες που θα μπορούσαμε να συμπεριλάβουμε στο τμήμα κώδικα που θα γίνει ο χειρισμός του λάθους:

- να διορθώσουμε μέσα από εντολές κώδικα το λάθος ή να δώσουμε τη δυνατότητα στο χρήστη να διορθώσει την αιτία πρόκλησης του λάθους. Στη συνέχεια να επιστρέψουμε τη ροή εκτέλεσης στην εντολή που προκάλεσε αυτή την κατάσταση λάθους. Μια τέτοια κατάσταση μπορεί να προκληθεί από την προσπάθεια του χρήστη να εκτυπώσει κάποιο αρχείο, ενώ ο εκτυπωτής είναι κλειστός. Μόλις ανιχνευτεί το λάθος, ενημερώνουμε το χρήστη ότι ο εκτυπωτής είναι κλειστός και αφού βεβαιωθούμε ότι διορθώθηκε το λάθος, επιστρέφουμε τη ροή εκτέλεσης του προγράμματος στην εντολή εκτύπωσης, η οποία ολοκληρώνεται με επιτυχία.
- να αξιολογήσουμε την εντολή που προκάλεσε το σφάλμα και αν θεωρήσουμε ότι η εκτέλεσή της δεν είναι κρίσιμη για την εφαρμογή, να μεταφέρουμε τη ροή εκτέλεσης στην επόμενη εντολή. Η εκτέλεση ενός αρχείου ήχου ταυτόχρονα με την εκτέλεση μιας εργασίας (π.χ. αντιγραφή αρχείων) δεν είναι σημαντική, εφόσον η συγκεκριμένη εργασία ολοκληρώνεται με επιτυχία.

- να θεωρήσουμε ιδιαίτερα κρίσιμο το σφάλμα και αφού ενημερώσουμε το χρήστη, να τερματίσουμε το πρόγραμμα. Αυτός ο τρόπος φυσικά πρέπει να αποφεύγεται, γιατί οδηγεί την εκτέλεση του προγράμματος σε απότομη διακοπή με απρόβλεπτες συνέπειες.

Όλες οι παραπάνω εργασίες πραγματοποιούνται μέσα από κατάλληλες εντολές που προσφέρουν σχεδόν όλα τα σύγχρονα περιβάλλοντα προγραμματισμού. Υπάρχουν εντολές που ανιχνεύουν ένα λάθος, δίνουν την περιγραφή του (αριθμητικά ή και με μήνυμα) και επιτρέπουν τη μεταβίβαση της ροής εκτέλεσης του προγράμματος.

Ανακεφαλαίωση

Το ζητούμενο κάθε χρήστη είναι ένα φιλικό και σταθερό πρόγραμμα που θα εργάζεται κάτω από οποιεσδήποτε συνθήκες και θα προλαβαίνει όλες τις εσφαλμένες του επιλογές. Η σωστή εκσφαλμάτωση ενός προγράμματος έχει ως αποτέλεσμα την κατασκευή τέτοιων προγραμμάτων που θα παρουσιάσουν όσο το δυνατό λιγότερα προβλήματα στο χρήστη.

Η εκσφαλμάτωση αποτελεί μια από τις βασικές εργασίες που πρέπει να κάνει ο προγραμματιστής για την ολοκλήρωση ενός προγράμματος. Κατά τη διαδικασία της εκσφαλμάτωσης ο προγραμματιστής πρέπει να εντοπίσει και να διορθώσει τα σημεία του προγράμματος που τυχόν προκαλούν λάθη. Για την επίτευξη του σκοπού αυτού πρέπει να κάνει τους κατάλληλους ελέγχους και να αναλύσει το πρόγραμμά του μέσα από τα εργαλεία εκσφαλμάτωσης που του παρέχει το περιβάλλον προγραμματισμού.

Τέλος ο προγραμματιστής για να δημιουργήσει ασφαλή και ποιοτικά προγράμματα, πρέπει να προβλέψει όλες τις καταστάσεις λάθους που μπορεί να συμβούν στο περιβάλλον εκτέλεσης από κακό χειρισμό ή προβληματικό εξοπλισμό και να δώσει στο πρόγραμμά του δυνατότητες αντιμετώπισης και τρόπους διαφυγής.

Λέξεις κλειδιά

Λάθος, έλεγχος, εκσφαλμάτωση, εργαλεία εκσφαλμάτωσης, χειρισμός λάθους.

Ερωτήσεις - Θέματα για συζήτηση

1. Να δοθούν παραδείγματα λαθών που εμφανίζονται συχνά στο λογισμικό.

2. Να γίνει συζήτηση σχετικά με τις επιπτώσεις των λαθών του λογισμικού.
3. Περιγράψτε το χειρισμό ενός λάθους που εμφανίζεται κατά το χρόνο εκτέλεσης ενός προγράμματος.
4. Περιγράψτε τη χρησιμότητα των εργαλείων εκσφαλμάτωσης.
5. Να γίνει συζήτηση για τις αιτίες που οδηγούν στην κατασκευή προβληματικού λογισμικού.

Βιβλιογραφία

1. Εγκυκλοπαίδεια Πληροφορικής και Τεχνολογίας Υπολογιστών, Εκδόσεις Νέων Τεχνολογιών, Τόμος 2, Αθήνα, 1986.
2. Παν. Πολίτης-Ηλ. Γιαννόπουλος: Προγραμματισμός με τη Visual Basic 4.0, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1997.
3. Ivars Peterson: Fatal Defect:Chasing killer Computer Bugs, Science News, USA, 1995.

Διευθύνσεις Διαδικτύου

⇨ http://www.alsplace.com/home/pages/famous_bugs.htm

Εντυπωσιακή ιστοσελίδα με σημαντικά ιστορικά σφάλματα που έχουν παρουσιαστεί στο χώρο των υπολογιστών.

⇨ http://www.bugnet.com

Περιέχονται πληροφοριακά δελτία για λάθη που παρουσιάζονται στο χώρο των υπολογιστών και τεχνικές αντιμετώπισής τους.

⇨ http://www.guide-p.infoseek.com

Περιέχει συνδέσεις με περιοδικά που παρέχουν πληροφορίες για λάθη που εμφανίζονται στους υπολογιστές.

⇨ http://www.netstuff.com/computerbug

Ιστοσελίδα με προϊόντα σχετικά με λάθη υπολογιστών.

⇨ http://www.techweb.com

Ενημερωμένη ιστοσελίδα με εκτιμήσεις, μελέτες, αποτελέσματα δοκιμών και εργαλείων για την αντιμετώπιση του προβλήματος του έτους 2000.

⇨ http://www.year2000.co.nz

Περιέχει πληροφορίες για το πρόβλημα του έτους 2000.

ΚΕΦΑΛΑΙΟ 14ο

ΑΞΙΟΛΟΓΗΣΗ - ΤΕΚΜΗΡΙΩΣΗ

14.1. Προσδοκώμενα αποτελέσματα

Σ' αυτό το τελευταίο κεφάλαιο του βιβλίου θα σου δοθεί η ευκαιρία να γνωρίσεις ποια είναι τα κριτήρια με τα οποία αξιολογούνται τα προγράμματά που κατασκευάζεις. Θα σου δοθεί η δυνατότητα μέσα από τις ασκήσεις που ακολουθούν να διατυπώσεις διαφορετικές λύσεις για το ίδιο πρόβλημα, εφαρμόζοντας τη μικρή πείρα που έχεις ήδη αποκτήσει και τις ιδέες που περιμένουμε από σένα.

Δοκιμάζοντας τις λύσεις που έχουν κατασκευάσει άλλοι, και προσπαθώντας να ανακαλύψεις τα όρια αξιοπιστίας των προγραμμάτων τους, θα συγκεντρώσεις χρήσιμες παρατηρήσεις, που θα μπορείς να εφαρμόσεις στα δικά σου προγράμματα.

14.2. Επιπλέον παραδείγματα

Παράδειγμα 1

Δίνεται ο δισδιάστατος πίνακας A(i,k) όπου στην πρώτη διάσταση περιέχει την κωδικό είδους ενός υλικού και στη δεύτερη την ποσότητα του υλικού. Ο πίνακας Α είναι ταξινομημένος κατά είδος. Ζητείται να εκτυπώνονται σύνολα ποσότητας, σε κάθε αλλαγή είδους και η συχνότητα εμφάνισης κάθε κωδικού. Στο τέλος της επεξεργασίας να τυπώνεται γενικό σύνολο ποσότητας.

Παρακάτω δίνεται ένας πίνακας με εικονικά δεδομένα και τα αποτελέσματα που πρέπει να παράγει το πρόγραμμά μας με αυτά τα δεδομένα.

Δεδομένα		Αποτελέσματα	
Κωδικός Είδους	Ποσότητα	Σύνολο Είδους	Γενικό Σύνολο
101	20		
101	30		
101	15		
101	20		
101	22	107	
105	1		
105	7	8	
200	17		
200	2		
200	3	22	
250	29	29	
280	12		
280	13	25	
310	6	6	
320	7	7	
330	9	9	213

Το πρόβλημα είναι παρόμοιο με το παράδειγμα 1 του βιβλίου. Εδώ τη θέση του αριθμού παίρνει ο κωδικός είδους. Επομένως θα αναζητήσεις την αλλαγή τιμής στο πεδίο του κωδικού είδους. Αν ακολουθήσεις την λογική της πρώτης λύσης του 1^{ου} παραδείγματος του βιβλίου, θα έχεις το παρακάτω πρόγραμμα σαν λύση του προβλήματος.

```
ΠΡΟΓΡΑΜΜΑ Σύνολα_Είδους
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Α, GS, i, S, Προηγ_Α
ΑΡΧΗ
GS <- 0
i <- 1
S <- 0
Προηγ_Α <- Α(1,1)
ΟΣΟ i<1001 ΕΠΑΝΑΛΑΒΕ
    ΑΝ Προηγ_Α<>Α(i,1) ΤΟΤΕ
        ΓΡΑΨΕ Προηγ_Α, S
        GS <- GS+S
        Προηγ_Α <- Α(i,1)
        S <- 0
    ΤΕΛΟΣ_ΑΝ
```

```
    S <- S+A(i,2)
    i <- i+1
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Προηγ_Α, S
GS <- GS+S
ΓΡΑΨΕ GS
ΤΕΛΟΣ Σύνολα_Είδους
```

Παράδειγμα 2

'Ενας αριθμός λέγεται πρώτος αν διαιρείται μόνο από το 1 και τον εαυτό του. 'Εστω, λοιπόν, ότι δίνεται ένας θετικός ακέραιος αριθμός n και ζητείται να διαπιστωθεί αν είναι πρώτος ή όχι. Μία πρώτη σκέψη είναι να αρχίσουμε να διαιρούμε τον αριθμό αυτόν διαδοχικά με 2, 3, 4, ..., n-1. 'Ετσι, αν ο αριθμός n δεν διαιρείται ακριβώς με κανένα από τους αριθμούς αυτούς, τότε πράγματι ο αριθμός n είναι πρώτος.

```
Αλγόριθμος ΠρώτοςΑριθμός
Διάβασε n
flag ← Ψευδής
i ← 2
Αρχή_επανάληψης
    Αν n mod i = 0 τότε flag ← Αληθής
    i ← i+1
Μέχρις_ότου flag=Αληθής ή i>n-1
Αποτελέσματα // flag //
Τέλος ΠρώτοςΑριθμός
```

Ο προηγούμενος αλγόριθμος χρησιμοποιεί μία συνάρτηση "mod" που επιστρέφει το υπόλοιπο της ακέραιης διαίρεσης. Αν αυτό είναι μηδέν τότε ο αριθμός δεν είναι πρώτος και η σημαία flag γίνεται Αληθής.

Με λίγη προσοχή και ελάχιστο κόπο, ο προηγούμενος αλγόριθμος μπορεί να γίνει ταχύτερος αν αλλάξει η συνθήκη της εντολής "Αρχή_επανάληψης ... μέχρις ότου" από i>n-1 σε i>Ρίζα(n). Αυτό μπορεί να γίνει με βεβαιότητα, καθώς αποδεικνύεται και μαθηματικά. Για παράδειγμα, αν θέλουμε να διαπιστώσουμε αν ο αριθμός 11 είναι πρώτος, τότε δεν είναι απαραίτητο να διαιρέσουμε το 11 δια 2, 3, κλπ μέχρι το 10, αλλά μέχρι το 4. Ο λόγος είναι ότι αν το 11 διαιρείτο ακριβώς δια του 5, του 6 κλπ, τότε θα διαιρείτο ακριβώς και δια του 11/5, 11/6 κοκ, γεγονός που θα είχε ήδη αποκαλυφθεί.

Παράδειγμα 3. Πίνακας Αποφάσεων

Είναι γνωστό ότι δίσεκτο έτος λέγεται αυτό που ο αριθμός του είναι πολλαπλάσιο του 4, αλλά όχι του 100, εκτός αν είναι πολλαπλάσιο του 400. Για παράδειγμα: το 1984 είναι δίσεκτο (πολλαπλάσιο του 4), το 1900 δεν είναι (πολλαπλάσιο του 4 αλλά και του 100), το 1600 είναι δίσεκτο (πολλαπλάσιο του 4, του 100, αλλά και του 400), τέλος το 1993 δεν είναι. Η διατύπωση της λύσης για ένα τέτοιο έλεγχο είναι γενικά δύσκολη υπόθεση. Ας δούμε ένα τμήμα προγράμματος που δίνει μία λύση.

```
.....
E4 <- E MOD 4
E100 <- E MOD 100
E400 <- E MOD 400
ΑΝ ε4=0 ΤΟΤΕ
    δις <- 1
ΤΕΛΟΣ_ΑΝ
ΑΝ ε100=0 ΤΟΤΕ
    δις <- 0
ΤΕΛΟΣ_ΑΝ
ΑΝ ε400=0 ΤΟΤΕ
    δις <- 1
ΤΕΛΟΣ_ΑΝ
.....
```

‘Οταν το δις είναι 1 το έτος είναι δίσεκτο, όταν είναι 0, δεν είναι.

Η κωδικοποίηση των **Αν** δείχνει ότι είναι ανεξάρτητα, όμως δεν είναι. Πράγματι αν αλλάξεις την σειρά των **Αν** και δοκιμάσεις το πρόγραμμα που θα προκύψει, θα διαπιστώσεις ότι δίνει διαφορετικά αποτελέσματα.

Αυτό συμβαίνει γιατί δόθηκε μία λύση, χωρίς προηγούμενα να έχει κατανοηθεί πλήρως η αλληλεξάρτηση των συνθηκών. Σ’ αυτές τις περιπτώσεις μπορείς να χρησιμοποιήσεις ένα άλλο τρόπο εύρεσης της λύσης, που θα σε βοηθήσει στην βαθύτερη κατανόηση του προβλήματος, άρα και στη σωστή προσέγγιση της λύσης. Μπορείς να χρησιμοποιήσεις ένα **Πίνακα Αποφάσεων**. ‘Ενας πίνακας αποφάσεων έχει την εξής μορφή:

Συνθήκες (που θα ελέγχονται)	Τιμή της Συνθήκης (Αληθής, Ψευδής)
Ενέργειες / Συμπέρασμα	Τιμή ενέργειας (Ναι, ‘Οχι)

Ο πίνακας αποφάσεων ενδείκνυται όταν υπάρχουν πολλές συσχετιζόμενες συνθήκες που προκαλούν διαφορετικές ενέργειες / συμπεράσματα. Η συμπλήρωση των στηλών γίνεται από όλους τους δυνατούς συνδυασμούς Α και Ψ των συνθηκών που πρέπει να ελεγχθούν μέσα στο πρόγραμμα. Αυτοί είναι σε πλήθος 2ν, όπου ν το πλήθος των διαφορετικών συνθηκών. Το τμήμα με τις ενέργειες / συμπεράσματα συγκροτείται από μία ή περισσότερες λογικές προτάσεις που είναι και ενέργειες του προγράμματος σου. Σε κάθε στήλη, αφού ελεγχθεί η λογική του συνδυασμού των Α και Ψ, γράφεται η τιμή της ενέργειας που πρέπει εκτελεστεί ή το συμπέρασμα που προκύπτει.

Για το συγκεκριμένο πρόβλημα θέλεις να γνωρίζεις αν το έτος είναι πολλαπλάσιο, δηλαδή αν διαιρείται ή δεν διαιρείται ακριβώς με το 4, το 100 ή το 400 προκείμενου να συμπεράνεις ότι το έτος είναι δίσεκτο. ‘Αρα στον παραπάνω πίνακα έχεις τρεις συνθήκες, που οι πιθανοί συνδυασμοί των απαντήσεων τους είναι 8. Ενώ το συμπέρασμα είναι ένα, αν δηλαδή το έτος είναι δίσεκτο ή όχι.

Ας σχηματίσεις λοιπόν τον πίνακα αποφάσεων για το δίσεκτο έτος, θα έχεις:

Συνθήκες	Περιπτώσεις							
	1η	2η	3η	4η	5η	6η	7η	8η
E mod 4 =0	Α	Α	Α	Α	Ψ	Ψ	Ψ	Ψ
E mod100=0	Α	Α	Ψ	Ψ	Α	Ψ	Α	Ψ
E mod 400=0	Α	Ψ	Α	Ψ	Α	Α	Ψ	Ψ
Ενέργειες/ Συμπεράσματα								
Δίσεκτο	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ

Από την παρατήρηση του πίνακα προκύπτει ότι μόνο η 1η και η 4η περίπτωση οδηγούν στο συμπέρασμα ότι το έτος είναι δίσεκτο, αυτό το συμπέρασμα στο πρόγραμμα θα το υλοποιήσεις με μία μεταβλητή δύο τιμών (πχ. 1:είναι, 0: δεν είναι). Επομένως τώρα γνωρίζεις πώς να συντάξεις τις σχετικές ερωτήσεις, και την σχέση που υπάρχει μεταξύ τους. Μία λύση δίνεται στο 4ο παράδειγμα του βιβλίου. Πρόσεξε ότι στην 4η περίπτωση του πίνακα, όταν E mod 100 ≠ 0 αναγκαστικά θα είναι και E mod 400 ≠ 0 και άρα δεν χρειάζεται να γίνει επιπλέον έλεγχος.

14.3. Συμβουλές

- Αν στο πρόγραμμα σου υπάρχουν τμήματα για τα οποία μπορείς να δώσεις ένα όνομα όπως "Υπολογισμός Παραγγελίας", "Διάβασμα και έλεγχος", "Αλλαγή κωδικού" και άλλα παρόμοια, τότε αυτά είναι κατάλληλα να γίνουν ενότητες.
- Αντιμετώπιζε παρεμφερή θέματα με τον ίδιο τρόπο. Αυτό θα δώσει "υπογραφή" στα προγράμματα σου. Αν η "υπογραφή" αποδειχτεί "κακή", αυτό θα το καταλάβεις γρήγορα.
- Να κρατάς (μαζεύεις) σημειώσεις όπως ο φιλάργυρος μαζεύει δεκάρες. Αργότερα θα ξεχωρίσεις και θα κρατήσεις αυτές που αξίζουν.
- Να ζητάς από άλλους να δοκιμάσουν το "σωστό" πρόγραμμα σου. Αν δεν είναι φίλοι ακόμα καλύτερα.
- Γράφε σχόλια ανάμεσα στις γραμμές του κώδικα.
- 'Οταν σχεδιάζεις τη λύση του προβλήματος, να σκέπτεσαι, "τι θα μου ζητήσουν μετά;"
- Αν το πρόγραμμα είναι μεγάλο σχεδίασε πρώτα ένα VTOC, είναι καλή ιδέα.
- Αν πρέπει να εξετάσεις πολύπλοκες συνθήκες, δεν κάνεις καλύτερα έναν πίνακα αποφάσεων;
- Να χρησιμοποιείς δοκιμαστικά δεδομένα για να ελέγχεις τη λύση. Αν κάπου βρήκες λάθη, ξανά έλεγξε. Εκεί που βρέθηκε ένα λάθος πιθανόν να υπάρχουν και άλλα.

- Μετά από κάθε αλλαγή στο πρόγραμμά σου, να το ξανά ελέγχεις με όλα τα δοκιμαστικά δεδομένα που είχες χρησιμοποιήσει και να συμπληρώνεις την τεκμηρίωση.
- Μια βιαστική λύση καταλήγει σε πρόγραμμα "μιας χρήσης".
- Μην γράφεις σύνθετες συνθήκες στα **Αν**, προτίμησε να γράφεις φωλιασμένα **Αν**, ένα για κάθε συνθήκη. Εάν το επιτρέπει η λογική του προγράμματος, μετάτρεψέ τα σε απλά **ΑΝ**.
- Μην "κάνεις εξυπνάδες", ακόμη και αν νομίζεις ότι έχεις τον απόλυτο έλεγχο. Εκτός του ότι δεν υπάρχει απόλυτος έλεγχος, σκέψου μήπως δεν υπάρχει και λόγος.
- Μεταξύ των νέων προγραμματιστών κυκλοφορεί η άποψη ότι το πρόγραμμα είναι σαν το χταπόδι. "'Οσο το κτυπάς (με τον compiler), τόσο μαλακώνει, στο τέλος θα τρώγεται".

4. Δραστηριότητες

Στην Τάξη

ΔΤ1. Τι ημέρα ήταν πριν ή μετά από χ μέρες. Διερεύνησε ποια είναι τα απαραίτητα δεδομένα.

Υπόδειξη: Χρησιμοποίησε ένα πίνακα με τις ημέρες της εβδομάδος. Αν γνωρίζεις ότι η σήμερα είναι Τετάρτη, και το χ είναι +10. Τότε επειδή 10mod7=3, σημαίνει ότι μετά από δέκα μέρες θα είναι Τετάρτη και 3 ημέρες, δηλαδή Σάββατο.

ΔΤ2. Ο πίνακας αποφάσεων στο 3ο παράδειγμα περιέχει οκτώ στήλες όσοι και οι δυνατοί συνδυασμοί των τριών προτάσεων. Αφού τον μελετήσεις, να αποκλείσεις τις περιπτώσεις που δεν μπορούν να συμβούν (αλληλοαναιρούνται). Στην συνέχεια με τη βοήθεια του πίνακα αποφάσεων πρότεινε εναλλακτικές λύσεις με τμήμα κώδικα. Δεδομένα ελέγχου: τα 1996, 1600, 2000 είναι δίσεκτα, τα 1500, 1998, 1900, 2006 δεν είναι δίσεκτα.

ΔΤ3. Να γραφεί το τμήμα του προγράμματος που πρέπει να προστεθεί στη λύση του 3ου παραδείγματος του βιβλίου, ώστε να εμφανίζονται έξη αριθμοί α, β, γ, δ, ε, ζ σε αύξουσα σειρά.

ΔΤ4. Δίνεται από το πληκτρολόγιο σειρά λέξεων. Να σχεδιαστεί αλγόριθμος που θα τις κατατάσσει με λεξικογραφική σειρά και θα τις εμφανίζει στην οθόνη σε τρεις στήλες.

Υπόδειξη: 'Ορισε τα όρια του προγράμματος σου και δώσε τη σχετική πληροφορία στο χρήστη.

Στο σπίτι

ΔΣ1. Να κατασκευαστεί πρόγραμμα που θα δέχεται μία ημερομηνία με τη μορφή ηη/μμ/εεεε και θα δίνει την ημέρα της εβδομάδος που αντιστοιχεί. Να γίνει εφαρμογή στην ημέρα γέννησής σου.

Υπόδειξη: Βασίσου στη λύση της ΔΤ1

ΔΣ2. Να μετατραπεί ένας αριθμός του δεκαδικού συστήματος στον ίσο του ελληνικού αριθμητικού συστήματος.

α, β, γ, δ, ε, στίγμα, ζ, η, θ
ι, κ, λ, μ, ν, ξ, ο, π, κόππα
π, ρ, σ, τ, υ, φ, χ, ψ, ω, σαμπί

ϛ = στίγμα
ϟ = κόππα
ϡ = σαμπί

ΔΣ3. Μια αεροπορική εταιρία έχει δύο κατηγορίες θέσεων, Α' θέση και Τουριστική. Να κατασκευαστεί πρόγραμμα που θα εκδίδει εισιτήριο ανάλογα με την επιθυμία του πελάτη. Οι επιθυμίες του πελάτη μετά από έρευνα βρέθηκε ότι είναι:

Ζητά εισιτήριο Α' θέσης αποκλειστικά.
Ζητά εισιτήριο Τουριστικής θέσης αποκλειστικά.
Θέλει να ταξιδέψει οπωσδήποτε.
Να κατασκευαστεί μόνο ο πίνακας αποφάσεων για το συγκεκριμένο πρόγραμμα.
Να συνταχθεί ο φάκελος τεκμηρίωσης του προγράμματος.

Υπόδειξη: Πρόσεξε ότι η έκδοση εισιτηρίου είναι και συνάρτηση της ύπαρξης ή όχι διαθέσιμου εισιτηρίου.

Ο πίνακας αποφάσεων θα έχει πολλές στήλες / περιπτώσεις. Μετά το αρχικό ανάπτυγμα του πίνακα εξέτασε ποιες περιπτώσεις δεν είναι δυνατόν να συμβούν και σύμπτυξε τον πίνακα.

ΔΣ4. Να κατασκευάσετε πρόγραμμα που θα διαβάζει δυάδες αριθμών από το πληκτρολόγιο και θα τους εμφανίζει με αύξουσα σειρά στην οθόνη.

Το ίδιο για τριάδες αριθμών. Τέλος να συνδυάσεις τα παραπάνω σε ένα πρόγραμμα που θα εκτελεί και τις δύο λειτουργίες ανάλογα με την επιθυμία του χρήστη.

Να συνταχθεί ο φάκελος τεκμηρίωσης του προγράμματος.

Υπόδειξη: Συνδύασε το 2ο και 3ο παράδειγμα του βιβλίου.

Στο εργαστήριο

ΔΕ1. Δίνεται ταξινομημένος πίνακας με στοιχεία, 2, 2, 3, 3, 3, 3, 5, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9. Ζητείται να κατασκευαστεί πρόγραμμα θα εμφανίζει την συχνότητα του αριθμού και την θέση του. Σε περίπτωση που ο αριθμός βρίσκεται πε-

ρισσότερες από μία φορά, η θέση να είναι η πρώτη που παρουσιάζεται.

Για τα δεδομένα που είδη δώσαμε το πρόγραμμα σου θα πρέπει να δίνει τα παρακάτω αποτελέσματα:

Αριθμός	Συχνότητα	Θέση
2	2	1
3	4	3
5	1	7
6	5	8
7	1	13
8	6	14
9	6	20

Υπόδειξη: ακολούθησε τα βήματα του 1ου παραδείγματος του τετραδίου.

ΔΕ2. Τι αλλαγή θα κάνεις στο πρόγραμμα της ΔΕ1 ώστε να εμφανίζεται η τελευταία θέση που βρέθηκε ο ίσος αριθμός;

Για τα δεδομένα σου τα αποτελέσματα θα είναι:

Αριθμός	Συχνότητα	Θέση
2	2	2
3	4	6
5	1	7
6	5	12
7	1	13
8	6	19
9	6	25

ΔΕ3. Στην αρχαία Ελλάδα η μετάδοση των μηνυμάτων γινόταν με φρυκτωρίες (από φρυκτός = πυρσός και ώρα = φροντίδα). Αναφορές του Όμηρου το επιβεβαιώνουν. Η μέθοδος περιγράφεται από τον ιστορικό Πολύβιο και επινοήθηκε το 350 π.χ. από τους Αλεξανδρινούς τεχνικούς Κλεοξένη και Δημόκλειτο.
Η μέθοδος απαιτούσε δύο ή περισσότερους σταθμούς σε απόσταση που σήμερα υπολογίζεται σε 30 km. Κάθε σταθμός είχε δύο ομάδες των πέντε δαυλών με επάλειψη ρετσίνι ή ακάθαρτο πετρέλαιο από την περιοχή Κερί της Ζακύνθου, από τότε μέχρι σήμερα) το άγγαρο (άσβεστο) πυρ που αναφέρει ο Αισχύλος. Εχοντας χωρίσει την αλφάβητο σε πέντε πεντάδες, η πρώτη ομάδα των πέντε πυρσών έδειχνε τη στήλη ενώ η δεύτερη ομάδα τη γραμμή. Ετσι ο συνδυασμός των αναμμένων πυρσών αντιστοιχούσε σε συγκεκριμένο γράμμα, άρα μπορούσε να μεταδοθεί ένα μήνυμα

γράμμα-γράμμα.
Να σχεδιαστεί αλγόριθμος που να διαβάζει μία λέξη και να εμφανίζει για κάθε γράμμα το αντίστοιχο ζεύγος αριθμών (πυρσών).
Σημείωση: Ο πίνακας των γραμμάτων είναι:

		ΣΤΗΛΗ				
		1	2	3	4	5
ΓΡΑΜΜΗ	1	Α	Β	Γ	Δ	Ε
	2	Ζ	Η	Θ	Ι	Κ
	3	Λ	Μ	Ν	Ξ	Ο
	4	Π	Ρ	Σ	Τ	Υ
	5	Φ	Χ	Ψ	Ω	

ΔΕ4. Στο βιβλίο αναφέρονται διάφοροι τρόποι δημιουργίας ψηφίου ελέγχου. Κατασκεύασε πρόγραμμα που θα παράγει το ψηφίο ελέγχου ενός οκταψήφιου αριθμού. Δοκίμασε διαφορετικές μεθόδους. Δοκίμασε και διαλέξτε αυτήν τη λύση που παράγει αξιόπιστο ψηφίο ελέγχου.

ΔΕ5. Το ρωμαϊκό σύστημα αρίθμησης χρησιμοποιεί όπως είναι γνωστό χαρακτήρες που αντιστοιχούν σε αριθμούς. Οι χαρακτήρες του ρωμαϊκού συστήματος και η αντίστοιχη αξία τους είναι η εξής:

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Ζητείται να κατασκευαστεί πρόγραμμα το οποίο, θα δέχεται έναν ρωμαϊκό αριθμό, και θα εμφανίζει τον αντίστοιχό του αραβικό στην οθόνη.

Διάβασε τον ρωμαϊκό αριθμό μέσα σε ένα πίνακα, κάθε ψηφίο σε μία θέση πίνακα, από τα αριστερά προς τα δεξιά, αφήνοντας τις τελευταίες θέσεις του πίνακα κενές.

R(1)	R(2)	R(3)	R(4)	R(5)	R(6)	R(7)	R(8)	R(9)	R(10)	R(11)	R(12)	
M	M	D	C	C	C	I	I	I	NULL	NULL	NULL	=2803
C	M	X	X	X	V	NULL	NULL	NULL	NULL	NULL	NULL	=935
M	C	D	L	X	X	X	I	I	NULL	NULL	NULL	=1482
M	C	D	X	L	I	I	NULL	NULL	NULL	NULL	NULL	=1442

Κατασκεύασε ένα VTOC.

Οι κανόνες που υπολογίζουν την αξία του ρωμαϊκού αριθμού είναι:

I. Αν το γράμμα που προηγείται έχει μικρότερη αξία από το επόμενο του τότε η αξία του αφαιρείται. Πχ. CD=500-100=400, IX=10-1=9, XLV=+5+50-10=45

II. Αν το γράμμα που προηγείται έχει μεγαλύτερη αξία από το επόμενο του τότε η αξία του προστίθεται. Πχ. DC=500+100=600, LXV =+5+10+50=65.

Προσοχή: Οι παρακάτω έλεγχοι θα εξασφαλίσουν, την αξιόπιστη λειτουργία του προγράμματος.

I. Δεν επιτρέπονται άλλοι χαρακτήρες εκτός από τους: I, V, X, L, C, D, M.

II. Δεν επιτρέπονται κενά ανάμεσα στους χαρακτήρες.

III. Δεν επιτρέπεται να υπάρχουν 2 συνεχόμενα σύμβολα ίσα, εκτός αν είναι τα I, X, C, M, οπότε δεν μπορεί να είναι παραπάνω από 3. Πχ. όχι LLV=+5+50+50=105 αλλά CV=+5+100=105, όχι XXXX=10+10+10+10=40 αλλά XL=+50-10=40, σωστό XXX=+10+10+10.

IV. Δεν επιτρέπεται να υπάρχουν 2 συνεχόμενα ίσα σύμβολα, αν ακολουθεί μεγαλύτερης αξίας σύμβολο. Πχ. IIX=+10-1-1=8 λάθος, VIII =1+1+1+5=8 σωστό, IM=999 σωστό. IIC=+100-1-1=98 λάθος, XCVIII=+1+1+1+5+100-10=98 σωστό.

V. Δεν επιτρέπεται να υπάρχει δύο φορές το ίδιο σύμβολο και να χωρίζεται από ένα μόνο άλλο σύμβολο.
Πχ. VLVM=1000-5+50-5=1040 λάθος, MXL=+50-10+1000=1040 σωστό, CMXCVIII=+1+1+1+5+100-10+1000-100=998 σωστό.

VI. Δεν επιτρέπεται να υπάρχουν συνεχόμενα, 2 διαφορετικά σύμβολα μικρότερα από το επόμενό τους.
Πχ. LVM=1000-5+50=1045 λάθος, MXLV=5+50-10+1000=1045 σωστό, XLMVC=100-5+1000-50-10=945 λάθος, VLM=1000-50-5=945 λάθος, CMVL=50-5+1000-100=945 σωστό.

Δεδομένα ελέγχου: MMDCCCIII = 2803, CMXXXV = 935, MCDLXXXII = 1482, MCDXLII = 1442, και όλα τα παραπάνω.

14.5. Τεστ Αυτοαξιολόγησης

1. Η μετατροπή μιας δραχμικής αξίας σε αξία euro που γίνεται στις Τράπεζες είναι πρόγραμμα ή εφαρμογή.

2. Η διάρκεια ζωής ενός προγράμματος αυξάνεται όταν, διαθέσεις περισσότερο χρόνο στη φάση:

   Α) της κωδικοποίησης

   Β) των ελέγχων

   Γ) της αρχικής σχεδίασης

3. Με ποιους τρόπους μπορείτε να αυξήσεις την ταχύτητα του προγράμματός σου:

   Α) με έναν ταχύτερο υπολογιστή
   Β) με έρευνα επί των δεδομένων
   Γ) με ένα γρήγορο πρόγραμμα
   Δ) με επινόηση άλλης τεχνικής

4. 'Οταν επιλέγεις ευφυή λύση στο πρόβλημά σου, τι πρέπει να προσέχεις:

   Α) Την πολύ καλή τεκμηρίωση.
   Β) Την πολύ καλή κωδικοποίηση.
   Γ) Τους εξονυχιστικούς ελέγχους.
   Δ) Να μην σου κλέψουν την ιδέα.

5. 'Οταν σχεδιάζεις τη λύση ενός προβλήματος, τι πρέπει να προσέχεις περισσότερο:

   Α) Την ταχύτητα.
   Β) Την αξιοπιστία.
   Γ) Τη φιλικότητα.
   Δ) Την ευελιξία.
   Ε) 'Ολα τα παραπάνω.

6. Δίνεται ο εξαψήφιος κωδικός 123456, ποιο θα είναι το έβδομο ψηφίο ελέγχου που παράγεται με τη μέθοδο των συνεχών προσθέσεων (διέγραψε το σωστό):

   1, 2, 3, 4, 5, 6, 7, 8, 9, 0.

7. Στο 1ο παράδειγμα " Σύνολα_Είδους" ποιες εντολές περιέχει το καθένα από τα δύο τμήματα της αλλαγής αριθμού (τμήμα κύριας επεξεργασίας –Ε-, τμήμα αρχικών τιμών –Α-). Σημείωσε ένα Ε ή ένα Α αριστερά από την εντολή.

__ `ΓΡΑΨΕ Προηγ_Α, S`
__ `GS <- GS+S`
__ `Προηγ_Α <-Α (1,i)`
__ `S <- 0`

8. Ποια χαρακτηριστικά πρέπει να έχει μια ενότητα. (Σωστό, Λάθος)

   Α) Ο βαθμός ανεξαρτησίας καθορίζει το αν και πόσο οι αλλαγές σε μία ενότητα, επιβάλλουν αλλαγές σε άλλες ενότητες.

   Β) Κάθε ενότητα έχει μία είσοδο, μία έξοδο και εκτελεί μία καθορισμένη επεξεργασία (κανόνας 3ε, είσοδος, επεξεργασία, έξοδος).

   Γ) Κάθε ενότητα καλείται από μία άλλη ενότητα και επιστρέφει σ' αυτήν όταν τελειώσει.

   Δ) Κάθε ενότητα αποτελείται από τις βασικές δομές, ακολουθία, επιλογή, επανάληψη.

   Ε) Σε κάθε ενότητα δεν πρέπει να υπάρχει απότομη διακοπή ή άπειρη επανάληψη.

9. Στα προγράμματα ελέγχου ημερομηνιών πρέπει να προσέχεις ώστε:

   Α) Να δίνεται η πληροφορία αν πρόκειται για ελληνική ημερομηνία ή ημερομηνία ευρωπαϊκού κράτους.

   Β) Το έτος να δίνεται πάντα διψήφιο.

Γ) Να ελέγχεται, αν το έτος είναι δίσεκτο.

Δ) Αν δίνονται αρνητικοί αριθμοί.

10. Συντήρηση ενός προγράμματος λέμε:

Α) Την αναζήτηση λαθών.

Β) Την προσθήκη νέων λειτουργιών.

Γ) Τη σύνταξη και ολοκλήρωση της τεκμηρίωσης.

11. Η τεκμηρίωση ενός προγράμματος γίνεται γιατί:

Α) Πρέπει.

Β) Χωρίς αυτήν η συντήρηση του προγράμματος θα είναι μια περιπέτεια με άγνωστο τέλος.

Γ) Χωρίς αυτήν δεν "τρέχει" το πρόγραμμα.

Δ) Χωρίς αυτήν ο εντοπισμός της αιτίας των λαθών είναι δύσκολος.

Ε) Ξεχνάς.

1. ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ

1.1. ΣΥΝΟΛΟ ΧΑΡΑΚΤΗΡΩΝ

Το σύνολο των χαρακτήρων της Turbo Pascal απαρτίζεται από τους αριθμητικούς χαρακτήρες (0-9), τους αλφαβητικούς κεφαλαία (A-Z) και πεζά (a-z), καθώς και τους περισσότερους ειδικούς χαρακτήρες.

1.2. ΣΤΑΘΕΡΕΣ

1.2.1. Αριθμητικές σταθερές

Υπάρχουν οι εξής τύποι:

- "μικροί" ακέραιοι (short integers) σε 1 byte με τιμές από -128 μέχρι 127
- ακέραιοι σε 2 bytes με τιμές από -32768 μέχρι 32767
- "μακροί" ακέραιοι (long integers) σε 4 bytes από -2147483648 μέχρι 2147483647
- byte σε 1 byte με τιμές από 0 μέχρι 255
- word σε 2 bytes με τιμές από 0 μέχρι 65535
- πραγματικούς σε 6 bytes με τιμές από 2.9e^-39 μέχρι 1.7e^38 και ακρίβεια 11-12 δεκαδικών ψηφίων
- πραγματικούς απλής ακρίβειας (real single precision) σε 4 bytes με τιμές από 1.5e^-45 μέχρι 3.4e^38 και ακρίβεια 7-8 δεκαδικών ψηφίων
- πραγματικούς διπλής ακρίβειας (real double precision) σε 8 bytes με τιμές από 5.0e^-324 μέχρι 1.7e^308 και ακρίβεια 15-16 δεκαδικών ψηφίων
- εκτεταμένοι πραγματικοί αριθμοί (extended) σε 10 bytes με τιμές από 3.4e^-4932 μέχρι 1.1e^4932 και ακρίβεια 19-20 δεκαδικών ψηφίων
- πραγματικοί αριθμοί τύπου comp σε 8 bytes με τιμές από -9.2e^18 μέχρι 9.2e^18 και ακρίβεια 19-20 δεκαδικών ψηφίων

1.2.2. Αλφαριθμητικές σταθερές

Μια αλφαριθμητική σταθερά είναι μια συμβολοσειρά (string) με μήκος μέχρι 32767 χαρακτήρες περικλειόμενη σε αγκύλες.

1.2.3. Συμβολικές σταθερές

Οι συμβολικές σταθερές ορίζονται με την εντολή CONST και μπορούν να χρησιμοποιηθούν αντί των αριθμητικών ή αλφαριθμητικών σταθερών, π.χ.

```
CONST max_Item = 255;
    Matr = array[1..max_Item] of integer;
```

Ορίζεται η συμβολική σταθερά max_Item ίση με 255, η οποία στη συνέχεια χρησιμοποιείται για να ορίσει τη διάσταση ενός πίνακα. Οι συμβολικές σταθερές μπορεί να είναι οποιουδήποτε τύπου και το όνομά τους σχηματίζεται με βάση τους κανόνες δημιουργίας μεταβλητών της Turbo Pascal.

1.2.4 Λογικές σταθερές.

Οι λογικές σταθερές (Boolean) μπορούν να πάρουν μόνο δύο τιμές TRUE ή FALSE.

1.3. ΜΕΤΑΒΛΗΤΕΣ

Μια μεταβλητή είναι ένα όνομα το οποίο αναφέρεται σε ένα αντικείμενο (έναν αριθμό, μια συμβολοσειρά ή μια εγγραφή).

Το όνομα μιας μεταβλητής σχηματίζεται από αριθμητικούς και αλφαριθμητικούς χαρακτήρες με τον πρώτο υποχρεωτικά αλφαβητικό.

Στους αλφαβητικούς χαρακτήρες των ονομάτων μεταβλητών δεν έχει σημασία αν είναι κεφαλαία ή πεζά γράμματα. Η Turbo Pascal δεν ξεχωρίζει τα κεφαλαία από τα πεζά γράμματα. Έτσι τα ονόματα LETTER, Letter ή LeTTer αναφέρονται στην ίδια θέση μνήμης, πρόκειται δηλαδή για τις ίδιες μεταβλητές. Ο χρήστης μπορεί να χρησιμοποιήσει κεφαλαία ή πεζά γράμματα κατά την κρίση του. Η Turbo Pascal δεν μετατρέπει τα πεζά γράμματα των μεταβλητών σε κεφαλαία και αντίστροφα. Στη συνέχεια, για λόγους καλύτερης αναγνωσιμότητας όλες οι μεταβλητές γράφονται με πεζά. Σε πολλές περιπτώσεις όμως όταν οι μεταβλητές ονοματίζονται, έτσι ώστε το όνομα να προσδιορίζει και το περιεχόμενο, τότε χρησιμοποιούνται και κεφαλαία γράμματα για έμφαση π.χ. MatrixIndex, ArrayPointer, κλπ.

Μια μεταβλητή μπορεί να είναι αριθμητική, αλφαριθμητική ή εγγραφή. Ο προσδιορισμός τους γίνεται δηλώνοντας τον τύπο της μεταβλητής σε δηλωτικές εντολές του τύπου:

Όνομα-μεταβλητής : τύπος

όπου τύπος μπορεί να είναι ένας από τους INTEGER, LONGINT, SINGLE, DOUBLE, REAL, STRING, CHAR, ARRAY, POINTER, RECORD, BOOLEAN ή τύπος δεδομένων του χρήστη.

π.χ. με την εντολή:

x : real;

η μεταβλητή x ορίζεται πραγματική.

Στον ορισμό αλφαριθμητικών μεταβλητών υπάρχουν δύο δυνατότητες, μεταβλητές σταθερού ή μεταβλητού μήκους. Π.χ. με την εντολή:

Mat1 : string;

Ορίζεται μια μεταβλητή με μήκος που μπορεί να εκτείνεται από 0 μέχρι 32767 χαρακτήρες.

Με την εντολή:

Mat1 : string[20];

Ορίζεται μια μεταβλητή με μήκος ακριβώς 20 χαρακτήρες.

Επίσης η Pascal επιτρέπει τη δημιουργία νέων τύπων δεδομένων από το χρήστη. Οι τύποι αυτοί ορίζονται με απλή απαρίθμηση των στοιχείων τους στο τμήμα δήλωσης ορισμού τύπων που τοποθετείται πριν από τη δήλωση των μεταβλητών και ορίζεται από τη δεσμευμένη λέξη TYPE. Στο τμήμα δήλωσης των μεταβλητών **μπορούν να οριστούν** μεταβλητές αυτού του τύπου.

Παράδειγμα.

```
TYPE
    Month={Jan, Feb, Mar, Apr, Mai, Jun,
    Jul, Aug, Sep, Oct, Nov, Dec}
VAR
    A, B:Month
```

Με τον ίδιο τρόπο μπορεί να δηλωθεί μια μεταβλητή εγγραφής, αρκεί προηγούμενα να έχει ορισθεί η εγγραφή με την εντολή TYPE. Π.χ.

```
TYPE
    StockItem = Record
        Code        : String[4];
        Description: String[20];
        UnitPrice   : Single;
        Quantity    : Long;
    End;

VAR
    CurrentItem : StockItem;
```

Μετά τον ορισμό μιας εγγραφής, η αναφορά σε ένα πεδίο της μπορεί να γίνει χρησιμοποιώντας τον τύπο όνομα-εγγραφής.όνομα-πεδίου, π.χ. η μεταβλητή CurrentItem.Code αναφέρεται στο πεδίο κωδικός της εγγραφής με το όνομα CurrentItem.

1.4. ΠΙΝΑΚΕΣ

Ένας πίνακας είναι ένα σύνολο αντικειμένων επί των οποίων η αναφορά γίνεται με το ίδιο όνομα μεταβλητής. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχεία γίνεται με το όνομα του πίνακα ακολουθούμενο από έναν ή περισσότερους δείκτες μέσα σε τετραγωνικές αγκύλες. Οι δείκτες προσδιορίζουν την τάξη του στοιχείου μέσα στον πίνακα.

Π.χ. Mat[2,4], Table[j-1]

Σε έναν πίνακα, ο αριθμός των δεικτών προσδιορίζει το πλήθος των διαστάσεων, ενώ η μέγιστη δυνατή τιμή κάθε δείκτη προσδιορίζει το πλήθος των στοιχείων του πίνακα ανά διάσταση. Πριν χρησιμοποιηθεί ένας πίνακας, πρέπει να οριστούν οι διαστάσεις του. Με τον όρο αυτό αναφερόμαστε τόσο στον ορισμό του πλήθους των διαστάσεων, όσο και των μέγιστων τιμών κάθε μιας. Ο ορισμός των διαστάσεων ενός πίνακα επιτυγχάνεται αν δηλώσουμε το όνομα του πίνακα τύπου ARRAY. Οι διαστάσεις του πίνακα διαχωρίζονται μεταξύ τους με κόμμα (,).

Π.χ. Mat1 : array[1..50] of integer;

Με αυτή την εντολή ορίζουμε έναν πίνακα με όνομα Mat1, που θα καταλαμβάνει 50 θέσεις μνήμης, και τα στοιχεία που θα αποθηκεύονται θα είναι τύπου ακέραιοι (integer).

Να σημειωθεί εδώ η δυνατότητα να ορίζονται με τον τύπο αυτό και αρνητικές τιμές δεικτών

π.χ. array[-5..5] of integer; Έτσι, μπορεί να ορισθεί οποιαδήποτε περιοχή τιμών των δεικτών από -32768 έως 32767.

Ένας πίνακας που μόλις έχει οριστεί, πρέπει στο κυρίως πρόγραμμα να αρχικοποιηθεί (initiallze). Για ονόματα πινάκων μπορούν να χρησιμοποιηθούν οποιεσδήποτε μεταβλητές οποιουδήποτε τύπου. Ως δείκτες χρησιμοποιούνται ακέραιοι ή ακέραιες εκφράσεις (μπορούν να χρησιμοποιηθούν και μη ακέραιοι αλλά στρογγυλοποιούνται πριν από τη χρήση).

Για τον ορισμό ενός πίνακα με στοιχεία εγγραφές αρχείου πρέπει πρώτα να δηλωθεί ο τύπος της εγγραφής και μετά οι διαστάσεις του πίνακα.

```
TYPE
    QueueNode = Record
        Data : String[20];
        QPtr : Integer;
    End;
VAR
    Qnode : array[1..100] of QueueNode;
```

Κάθε στοιχείο του πίνακα Qnode είναι μια εγγραφή τύπου QueueNode. Η αναφορά σε ένα πεδίο της εγγραφής σαν στοιχείο πίνακα γίνεται με τη χρήση του τύπου όνομα-πίνακα.πεδίο-εγγραφής,

π.χ. Writeln(Qnode.Data);

1.5. ΕΚΦΡΑΣΕΙΣ

Σε ένα πρόγραμμα Pascal μπορεί να συνυπάρχουν σταθερές, λογικοί (boolean) ή αριθμητικοί τελεστές καθώς και συναρτήσεις. Υπάρχουν από τη Γλώσσα οι σημαντικότεροι λογικοί τελεστές που είναι οι AND, OR, NOT και XOR. ενώ οι αριθμητικοί τελεστές είναι + (πρόσθεση),- (αφαίρεση),/ (πολλαπλασιασμός),* (διαίρεση), div που είναι η ακέραια διαίρεση και το mod που εξάγει το υπόλοιπο της διαίρεσης. Στη συνέχεια παρατίθεται σε πίνακα η προτεραιότητα τελεστών:

Προτεραιότητα	**Τελεστές**
1 (υψηλή)	NOT, μοναδιαίοι +,-
2	*,/,DIV,MOD,AND
3	δυαδικά +,-,OR,XOR
4 (χαμηλή)	=,<>,<,>,<=,>=

Οι λογικοί τελεστές εκτελούν λογικές πράξεις στους τελεστέους καθένας από τους οποίους θωρεί-

ται λογικά "αληθής" ή "ψευδής". Η Turbo Pascal αναγνωρίζει τη διαφορά μεταξύ των δύο παραπάνω κατηγοριών από τα συμφραζόμενα της έκφρασης. Οι λογικοί τελεστές φαίνονται στον παρακάτω πίνακα:

Τελεστής	Σύνταξη	Έννοια
NOT	NOT <έκφραση>	Αρνηση
AND	<έκφραση1> AND <έκφραση2>	Ένωση
OR	<έκφραση1> OR <έκφραση2>	Διάζευξη
XOR	<έκφραση1> XOR <έκφραση2>	Αποκλειστική Διάζευξη

Τελεστές Σύγκρισης

Οι τελεστές σύγκρισης συγκρίνουν δύο εκφράσεις και επιστρέφουν "αληθές" εάν η συνθήκη που ορίζεται από τον τελεστή ικανοποιείται ή "ψευδές" εάν όχι. Οι τελεστές σύγκρισης φαίνονται στον κατωτέρω πίνακα.

Τελεστής	Σύνταξη	Έννοια
=	<έκφραση1> = <έκφραση2>	Οι εκφράσεις είναι ίσες
<>	<έκφραση1> <> <έκφραση2>	Οι εκφράσεις δεν είναι ίσες
<	<έκφραση1> < <έκφραση2>	Η <έκφραση1> είναι μικρότερη από την <έκφραση2>
<=	<έκφραση1> <= <έκφραση2>	Η <έκφραση1> είναι μικρότερη ή ίση από την <έκφραση2>
>	<έκφραση1> > <έκφραση2>	Η <έκφραση1> είναι μεγαλύτερη από την <έκφραση2>
>=	<έκφραση1> >= <έκφραση2>	Η <έκφραση1> είναι μεγαλύτερη ή ίση από την <έκφραση2>

Η Pascal διαθέτει επίσης ένα πλούσιο ρεπερτόριο ενσωματωμένων μαθηματικών συναρτήσεων οι σπουδαιότερες των οποίων είναι:

Συνάρτηση	Λειτουργία
ABS	Εξάγει το απόλυτο μέρος ενός αριθμού
INT	Εξάγει το ακέραιο μέρος ενός δεκαδικού αριθμού
SQR	Υψώνει έναν αριθμό στο τετράγωνο
SQRT	Βρίσκει την τετραγωνική ρίζα
SIN	Βρίσκει το ημίτονο μιας γωνίας
COS	Βρίσκει το συνημίτονο μιας γωνίας
ARC	Βρίσκει το τόξο μιας γωνίας
ARCTAN	Βρίσκει το τόξο εφαπτομένης μιας γωνίας
LN	Βρίσκει τον λογάριθμο ενός αριθμού

1.6. ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Η γενική μορφή ενός προγράμματος Pascal είναι η εξής:

```
Program Ονομα_προγράμματος;
        (Δηλώσεις σταθερών)
        (Δηλώσεις τύπου μεταβλητών)
        (Δηλώσεις μεταβλητών)
BEGIN
        (Εντολές)
END.
```

Το όνομα του προγράμματος μπορεί να είναι ένα οποιοδήποτε αλφαριθμητικό όνομα. Μετά το όνομα του προγράμματος πρέπει να προστίθεται πάντα ένα ελληνικό ερωτηματικό. Το ελληνικό ερωτηματικό (;) στην Pascal γενικότερα παίζει μεγάλο ρόλο και η ύπαρξή του είναι καθοριστική. Υποδηλώνει το τέλος μιας εντολής ή δήλωσης και την αρχή μιας νέας. Η τελευταία εντολή του προγράμματος πρέπει να είναι η END και ακολουθείται υποχρεωτικά από την τελεία '.'.

Στην περιοχή δηλώσεων δεδομένων, καταγράφονται όλες οι σταθερές, οι τύποι δεδομένων και οι μεταβλητές που θα χρησιμοποιηθούν από το πρόγραμμα.Φυσικά δεν είναι υποχρεωτικό να υπάρχουν όλα αυτά τα τμήματα, για παράδειγμα αν το πρόγραμμα δεν χρησιμοποιεί σταθερές το αντίστοιχο τμήμα θα απουσιάζει.

Οι σταθερές δηλώνονται κάτω από την ετικέτα CONST με το όνομά τους και την τιμή που έχουν. π.χ. CONST Max = 100;

Στο παράδειγμα αυτό προσδιορίζεται μια σταθερά με όνομα Max που έχει την τιμή 100. Την τιμή αυτή θα την έχει έως το πέρας του προγράμματος. Το περιεχόμενο των σταθερών δεν επιτρέπεται αλλαχθεί μέσα στο πρόγραμμα.

Οι τύποι που θα χρησιμοποιούνται μέσα σε ένα πρόγραμμα ορίζονται στην περιοχή TYPE όνομα_τύπου = Τύπος_Δεδομένου;

π.χ. TYPE MyArray = array[1..10] of integer;

Στο παράδειγμα αυτό ορίζεται ένας τύπος πίνακα MyArray που θα καταλαμβάνει 10 θέσεις στη μνήμη του υπολογιστή και η κάθε θέση θα έχει έναν ακέραιο αριθμό. (Βλέπε πίνακες). Οι μεταβλητές του προγράμματος ορίζονται στην περιοχή

```
VAR όνομα_μεταβλητής : τύπος_δεδομένου;
```

π.χ. VAR number : integer;

Στο παράδειγμα αυτό ορίστηκε η μεταβλητή number που είναι τύπου ακεραίου.

```
VAR Arr1 : MyArray;
```

Στο παράδειγμα αυτό ορίστηκε η μεταβλητή πίνακα Arr1, σαν τύπου MyArray. Ο τύπος αυτός δεν είναι τύπος της Turbo Pascal αλλά έχει ορισθεί στο προηγούμενο παράδειγμα που αφορούσε την περιοχή δηλώσεων TYPE. Έτσι, δίνεται η δυνατότητα ορισμού τύπων δεδομένων του χρήστη και μέσω της περιοχής δηλώσεων VAR ορίζεται μια μεταβλητή του αυτού τύπου και η οποία θα χρησιμοποιηθεί τελικά μέσα στο πρόγραμμα. Ετσι, το παράδειγμα συνολικά θα δείχνει ως εξής:

```
TYPE
    MyArray = array[1..10] of integer;
VAR
    Arr1 : MyArray;
```

Μετά τις δηλώσεις δεδομένων ακολουθεί ο ορισμός τυχόν διαδικασιών ή συναρτήσεων που υπάρχουν στο πρόγραμμα. Αναφορά στις διαδικασίες και τις συναρτήσεις θα γίνει στη συνέχεια.

Ένα πρόγραμμα αποτελείται από γραμμές προγράμματος. Κάθε γραμμή προγράμματος περιέχει μια ή περισσότερες εντολές προγράμματος. Αν υπάρχουν περισσότερες από μια εντολές σε μία γραμμή, τότε πρέπει να χωρίζονται με το χαρακτήρα ελληνικό ερωτηματικό (;). Μια γραμμή προγράμματος μπορεί να εκτείνεται σε περισσότερες από μία φυσικές γραμμές.

2. ΕΚΧΩΡΗΣΗ ΤΙΜΩΝ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ

Εκχώρηση τιμών

Η πιο θεμελιώδης δήλωση στην Pascal είναι η δήλωση εκχώρησης τιμής η οποία εκχωρεί μια τιμή σε μια μεταβλητή. Η ταυτότητα (όνομα) της μεταβλητής που πρόκειται να αλλάξει δηλώνεται ακολουθούμενη από τον τελεστή εκχώρησης (:=) και την νέα τιμή.

π.χ.
```
quantity := 40;
message := 'File not found';
answer := Y;
```

Στην πρώτη γραμμή εκχωρείται μια ακέραια τιμή στην μεταβλητή quantity, στην δεύτερη γραμμή εκχωρείται ένα αλφαριθμητικό στην μεταβλητή message ενώ στην Τρίτη μεταβλητή εκχωρείται το περιεχόμενο της μεταβλητής Y στην μεταβλητή answer.

Επίσης αντί για συγκεκριμένες τιμές, σε μια μεταβλητή μπορεί να εκχωρείται το αποτέλεσμα μιας έκφρασης ή συνάρτησης.

π.χ.
```
x:=(3*y)/6;
z:=Sqrt(Sqr(x)+Sqr(x));
```

Είσοδος

Η είσοδος τιμών ή δεδομένων γίνεται με τις εντολές readln και read. Η σύνταξή τους είναι: readln(όνομα_μεταβλητής);

Εφόσον δεν ορίζεται συσκευή εισόδου θεωρείται ότι η είσοδος γίνεται από το πληκτρολόγιο.

Η διαφορά της readln από την απλή read είναι ότι στην πρώτη περίπτωση η είσοδος γίνεται στην αρχή μιας νέας σειράς σε αντίθεση με την read, στην οποία η είσοδος γίνεται στο σημείο που βρίσκεται την τρέχουσα στιγμή ο δρομέας. π.χ. readln(var1);

Έξοδος

Αντίστοιχα, η έξοδος δεδομένων γίνεται με τις εντολές writeln και write. Η γενική της σύνταξη είναι:

writeln(όνομα_μεταβλητής);

Η έξοδος αν δεν οριστεί κάποια άλλη συσκευή εξόδου γίνεται στην οθόνη.

Και σε αυτή την περίπτωση, η διαφορά μεταξύ των δύο εντολών είναι ότι στη μεν writeln, η έξοδος των δεδομένων πραγματοποιείται σε μια νέα γραμμή ενώ στην write, η έξοδος γίνεται στην τρέχουσα θέση του δρομέα.

π.χ. writeln(var1); εμφάνιση του περιεχομένου της μεταβλητής var1 στην οθόνη

write('Turbo Pascal'); εμφάνιση του λεκτικού "Turbo Pascal" στην οθόνη.

Για να καθοριστεί ο ακριβής τρόπος με τον οποίο θα εμφανίζονται τα δεδομένα χρησιμοποιείται η παράμετρος πλάτους πεδίου. Η τιμή του πλάτους πεδίου ορίζεται μετά τον χαρακτήρα άνω και κάτω τελεία ':'. Η τιμή κάθε παράστασης τυπώνεται σε σε τόσες θέσεις όσες είναι η τιμή του πλάτους του πεδίου. Για παράδειγμα η εντολή write A:5, θα τυπώσει το περιεχόμενο της μεταβλητής A σε 5 θέσεις.

Αν η μεταβλητή A περιέχει τον αριθμό 10 τότε το αποτέλεσμα της εντολής write A:5 είναι να αφήσει τρία κενά και μετά να εκτυπώσει τον αριθμό 10.

Σε περίπτωση πραγματικών αριθμών χρησιμοποιούνται δύο τιμές, η πρώτη καθορίζει το συνολικό πλάτος του αριθμού ενώ ο δεύτερος τα δεκαδικά στοιχεία που θα εκτυπωθούν.

Αν το A έχει την τιμή 3.14156 τότε η εντολή write A:4:2 θα εκτυπώσει 3.14

3. ΔΟΜΕΣ ΕΛΕΓΧΟΥ

Οι δομές ελέγχου είναι ένα εργαλείο που προσφέρουν οι διαδικαστικές γλώσσες προγραμματισμού για τον έλεγχο της σειράς εκτέλεσης των εντολών. Οι δομές ελέγχου είναι βασικά τρεις: η ακολουθία (sequence), η επιλογή (selection) και η ανακύκλωση ή βρόχος (loop). Οι δύο τελευταίες έχουν διάφορες παραλλαγές.

3.1. ΑΚΟΛΟΥΘΙΑ

Πρόκειται για την απλούστερη δομή κατά την οποία οι εντολές του προγράμματος εκτελούνται η μία μετά την άλλη, με τη σειρά που γράφονται στο πρόγραμμα, δηλ. Από την αρχή του κειμένου προς το τέλος και στην ίδια γραμμή από αριστερά προς τα δεξιά. Μεταξύ δύο συνεχόμενων εντολών που βρίσκονται στην ίδια γραμμή απαιτείται η παρουσία ενός διαχωριστή εντολών, που εδώ είναι ο χαρακτήρας ελληνικό ερωτηματικό ';'.

3.2. ΕΠΙΛΟΓΗ

3.2.1. Επιλογή

Κατά την εκτέλεση ενός προγράμματος είναι δυνατό να επιλεγεί η εκτέλεση ορισμένων εντολών και να αποφευχθεί η εκτέλεση άλλων. Η πιο απλή δομή για κάτι τέτοιο είναι η IF..THEN..ELSE. Η σύνταξη της εντολής είναι:

IF B THEN E1 ELSE E2

Η B είναι μια λογική παράσταση (μια συνθήκη) και η οποία έχει τιμές: αληθής (true) αν η συνθήκη ισχύει και ψευδής (false) αν η συνθήκη δεν ισχύει. Οι E1 και E2 είναι εντολές ή ομάδες εντολών. Η λειτουργία της εντολής είναι η εξής: υπολογίζεται πρώτα η λογική τιμή της παράστασης B. Αν η τιμή της B είναι αληθής τότε εκτελείται η εντολή E1, αλλιώς εκτελείται η E2.

Οι εντολές E1 και E2 μπορούν να είναι ατομικές εντολές ή σύνολο εντολών χωρισμένες με ";". Όλες σχεδόν οι εντολές της γλώσσας μπορούν να υπάρχουν μέσα στις E1 και E2, ακόμη και εντολές IF..THEN..ELSE, οπότε δημιουργούνται τα λεγόμενα εμφωλευμένα (nested) IF. Επίσης είναι δυνατόν το τμήμα ELSE να απουσιάζει, οπότε αν η συνθήκη δεν ισχύει, η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί την IF..THEN.

Παράδειγμα. Ένας άνδρας χαρακτηρίζεται ελαφρύς (βαρύς) αν είναι κάτω (πάνω) από 80 εκ. Επίσης χαρακτηρίζεται κοντός (ψηλός) αν είναι κάτω (πάνω) από 1.75 εκ. Στο επόμενο τμήμα προγράμματος εισάγονται το βάρος και το ύψος ενός ατόμου και εξάγεται ο χαρακτηρισμός του με σύνθετο IF.

```
Write('Εισάγετε το βάρος : '); readln(b);
Write('Εισάγετε το ύψος : '); readln(y);
If (b<80) then
    if (y<1.75) then
        writeln('Κοντός και ελαφρύς')
    Else
        writeln('ψηλός και ελαφρύς');
Else
    if (y<75) then
        Writeln('κοντός και βαρύς')
    Else
writeln('ψηλός και βαρύς');
```

Ένας άλλος τύπος εντολής IF είναι η εντολή IF.. ELSE IF..ELSE, με την οποία παρέχεται μεγαλύτερη ευελιξία στη σύνταξη πολύπλοκων δομών.

```
Writeln('Δώσε έναν χαρακτήρα : ');
Readln(a);
If (a>'0') and (a<='9') then
    writeln('Ψηφίο')
Else if  (a>='A') and (a<='z') then
    writeln('Λατινικό γράμμα')
Else if (a>='A') and (a<='ω') then
    writeln('Ελληνικό γράμμα')
Else
    writeln('μη αλφαριθμητικός χαρακτήρας');
```

Η Turbo Pascal εξετάζει κάθε μια από τις συνθήκες της IF και των ELSE IF δηλώσεων από πάνω προς τα κάτω υπερπηδώντας τις ομάδες των εντολών που ακολουθούν μέχρι να βρεθεί η πρώτη αληθής συνθήκη. Τότε εκτελούνται οι εντολές που αντιστοιχούν και μετά γίνεται διακλάδωση στην εντολή που ακολουθεί το μετά το τέλος της IF.

3.2.2. Πολλαπλή επιλογή

Με τον όρο αυτό αναφερόμαστε στη δυνατότητα μιας δομής να επιτυγχάνει την εκτέλεση μιας ομάδας εντολών ανάμεσα σε πολλές ή τη διακλάδωση του προγράμματος σε περισσότερα από ένα σημεία του προγράμματος ανάλογα με την τιμή της εξεταζόμενης συνθήκης, η σχετική εντολή είναι η CASE..END. Πρόκειται για εντολή πολλαπλών επιλογών απόφασης ανάλογη,από πλευράς τελικού αποτελέσματος, με το ομαδοποιημένο IF..THEN..ELSE. Η βασική διαφορά των δύο εντολών είναι ότι η πρώτη εξετάζει μια απλή έκφραση και ανάλογα με το αποτέλεσμα εκτελεί διαφορετικές εντολές ή προκαλεί διακλάδωση σε διαφορετικά σημεία, ενώ η δεύτερη εξετάζει απλές ή σύνθετες λογικές εκφράσεις. Αν και η λειτουργία της CASE..END μπορεί να πραγματοποιηθεί και με την IF..THEN..ELSE, λόγω της συμπαγούς δομής της η χρήση της προσφέρει σημαντικά πλεονεκτήματα στον προγραμματιστή.

Η γενική μορφή της εντολής είναι:

```
CASE όνομα_μεταβλητής OF
    Λίστα_Τιμών_1 : εντολές1;
    Λίστα_Τιμών_2 : εντολές2;
    ............
ELSE
    Εντολές_ν;
END;
```

Το όνομα μεταβλητής μπορεί να είναι αριθμητική ή αλφαριθμητική. Οι λίστες τιμών μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές, περιοχή τιμών από-έως, ή τιμές που υπακούουν σε μια λογική συνθήκη. Η εντολή εξετάζει την έκφραση και ανάλογα με την τιμή της εκτελεί τις εντολές που βρίσκονται μετά το διαχωριστικό ":". Αν η τιμή της μεταβλητής δεν αντιστοιχεί σε καμία από τις λίστες τιμών, τότε εκτελούνται οι εντολές που ακολουθούν την ELSE. Μετά την εκτέλεση των εντολών κάποιοιας CASE, η ροή του προγράμματος συνεχίζεται μετά το END της εντολής.

Η λειτουργία της εντολής καθώς και οι δυνατότητές της φαίνονται στα επόμενα παραδείγματα:

Παράδειγμα 1.

```
Writeln('Δώσε έναν ακέραιο από 1 ως 3 : ');
Readln(number);
CASE number OF
    1 : writeln('Ο αριθμός είναι 1');
    2 : writeln('Ο αριθμός είναι 2');
    3 : writeln('Ο αριθμός είναι 3');
ELSE
    Writeln('Λάθος εισαγωγή');
END;
```

Παράδειγμα 2.

```
Writeln('Δώσε έναν ακέραιο από 1 ως 20: ');
Readln(number);
CASE number OF
    1..10 : writeln('1η δεκάδα');
    11..20 : writeln('2η δεκάδα');
ELSE
    Writeln('Λάθος εισαγωγή');
END;
```

Υπάρχει επιπλέον η δυνατότητα ορισμού περιοχής τιμών από-έως για τις λίστες_τιμών. Η περιοχή τιμών δίνεται με το συνδετικό δύο τελειών "..", ανάμεσα από την αρχική και την τελική τιμή.

3.3. ΑΝΑΚΥΚΛΩΣΗ

Η δομή της ανακύκλωσης ή του βρόχου αναφέρεται στην επαναληπτική εκτέλεση μιας ομάδας εντολών προγράμματος κατά ένα προκαθορισμένο αριθμό φορών ή μέχρι να ισχύσει κάποια συνθήκη. Η δομή της ανακύκλωσης υλοποιείται με εντολές FOR..DO, WHILE..DO, REPEAT..UNTIL.

3.3.1. FOR..DO

Η εντολή αυτή εκτελεί τις εντολές προγράμματος που βρίσκονται μετά το FOR και ανάμεσα στο BEGIN..END που ακολουθεί και εκτελείται για όλες τις τιμές της μεταβλητής ελέγχου. Η μεταβλητή ελέγχου που αναφέρεται στη FOR, συνοδεύεται από την αρχική και τελική τιμή και σε κάθε επανάληψη αυξάνεται κατά μία μονάδα. Στο επόμενο τμήμα προγράμματος χρησιμοποιείται η εντολή FOR..DO για να διατρέξει διαδοχικά όλα τα στοιχεία του πίνακα K, και αν κάποιο στοιχείο είναι ίσο με τον αριθμό τότε τυπώνεται η θέση του στοιχείου.

```
Write('Εισάγετε τον κωδικό :'); readln(le);
Index:=0; ex:=0;
For i:=1 to 50 do
    If cle=k[i] then
        Writeln (I);
```

Υπάρχει και μια άλλη έκδοση αυτής της εντολής η FOR..DOWNTO..DO. Εδώ η μεταβλητή ελέγχου που συνοδεύει το βρόχο περιέχει αρχική τιμή μεγαλύτερη από την τελική τιμή του βρόχου. Έτσι, η επανάληψη γίνεται από την μεγαλύτερη στην μικρότερη τιμή. Σε κάθε επανάληψη η μεταβλητή μειώνεται κατά μία μονάδα.

3.3.2. WHILE..DO

Με την εντολή αυτή εκτελούνται όλες οι εντολές προγράμματος που υπάρχουν κάτω από την WHILE και ανάμεσα στο BEGIN..END, όσο ισχύει η συνθήκη που συνοδεύει το WHILE. Το επόμενο τμήμα προγράμματος κάνει ότι και το προηγούμενο αλλά πλέον η επαναληπτική εντολή είναι η WHILE..DO. Αν κατά την εξέταση όλων των στοιχείων ενός πίνακα k βρεθεί κάποιο στοιχείο που να ισούται με τη τιμή της μεταβλητής cle που έδωσε ο χρήστης από το πληκτρολόγιο, τυπώνεται η θέση του πίνακα.

```
Write('Δώσε τον κωδικό :'); readln(cle);
i:=1;
While i<=50 do
Begin
    If cle=k[i] then
        Writeln (I);
    i:=i+1;
End;
```

3.3.3. REPEAT..UNTIL

Παρόμοια στη λογική με τη WHILE..DO είναι και η REPEAT..UNTIL. Με την εντολή αυτή θα εκτελεστούν σίγουρα μία φορά όλες οι εντολές που περιλαμβάνονται μέσα στο REPEAT..UNTIL και αυτό γιατί η συνθήκη ελέγχου βρίσκεται στο τέλος του βρόχου και όχι στην αρχή (περίπτωση WHILE..DO). Έως ότου η συνθήκη αυτή γίνει ψευδής, η εκτέλεση των εντολών θα επαναλαμβάνεται συνεχώς..

Σε περίπτωση που δεν υπάρχει καθόλου συνθήκη ελέγχου, ο βρόχος θα εκτελείται συνεχώς.

Το επόμενο τμήμα προγράμματος κάνει ότι και το προηγούμενο αλλά πλέον η επαναληπτική εντολή είναι η REPEAT.. UNTIL. Αν κατά την εξέταση όλων των στοιχείων ενός πίνακα k βρεθεί κάποιο στοιχείο που να ισούται με τη τιμή της μεταβλητής cle που έδωσε ο χρήστης από το πληκτρολόγιο, τυπώνεται η θέση του πίνακα.

```
Write('Δώσε τον κωδικό :'); readln(cle);
i:=1;
repeat
    If cle=k[i] then
        Writeln (I);
    i:=i+1;
until I>50
```

Μία συχνή χρήση της εντολής REPEAT...UNTIL παρουσιάζεται στο επόμενο τμήμα προγράμματος

```
Write('Εντάξει ; (N/O) : ');
REPEAT
    Readln(a);
UNTIL (UPCASE(a) = 'N')OR (UPCASE(a)='O');
```

Σε αυτό το παράδειγμα ζητείται η εισαγωγή των χαρακτήρων Ν για Ναι και Ο για Οχι. Εστω κι αν δοθεί αντίστοιχος πεζός χαρακτήρας η συνάρτηση UPCASE αναλαμβάνει να τον μετατρέψει σε κεφαλαίο. Μόνο όταν δοθεί Ν (Ναι) ή Ο (Όχι) η ροή του προγράμματος θα συνεχίσει κανονικά. Σε αντίθετη περίπτωση ο βρόχος επαναλαμβάνεται.

4. ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ: ΔΙΑΔΙΚΑΣΙΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Η κυριότερη ίσως φροντίδα του προγραμματιστή εφαρμογών στην ανάπτυξη των προγραμμάτων είναι να επιτύχει τον καλύτερο χωρισμό ενός προγράμματος σε μικρότερες ενότητες. Κάθε ενότητα αφού πρώτα ελεγχθεί χωριστά και έπειτα σε συνεργασία με άλλες, θεωρείται ότι έχει ολοκληρωθεί και δεν απομένει παρά η σύνδεσή της με τις υπόλοιπες προκειμένου να αποτελέσουν μαζί ένα ενιαίο λειτουργικό σύνολο, το ζητούμενο πρόγραμμα. Η Turbo Pascal παρέχει έναν τρόπο κατάτμησης ενός μεγάλου προγράμματος σε μικρότερα με τις διαδικασίες (procedures) και τις συναρτήσεις (functions). Η χρήση τους αποφέρει σοβαρά πλεονεκτήματα στην καλή λειτουργία και συντήρηση των εφαρμογών του χρήστη, τα σπουδαιότερα των οποίων είναι:

1. Οι διαδικασίες επιτρέπουν τη διάσπαση ενός προγράμματος σε μικρότερες λογικές ενότητες, η κάθε μια των οποίων είναι ευκολότερο να ελεγχθεί και να διορθωθεί παρά ένα ολόκληρο πρόγραμμα χωρίς διαδικασίες.
2. Διαδικασίες που χρησιμοποιήθηκαν από ένα πρόγραμμα, μπορούν να χρησιμοποιηθούν και από άλλα με ελάχιστες ή καθόλου μετατροπές.

ΛΕΙΤΟΥΡΓΙΚΕΣ ΜΟΝΑΔΕΣ (UNITS)

Είναι δυνατόν οι διαδικασίες να ενταχθούν σε Λειτουργικές μονάδες (Units). Κάθε λειτουργική μονάδα, αποτελεί ένα ειδικό αρχείο το οποίο μπορεί να περιέχει μεταβλητές, τύπους δεδομένων, σταθερές, διαδικασίες ή συναρτήσεις και από την οποία είναι εφικτή η κλήση των διαδικασιών ή συναρτήσεων της Λειτουργικής μονάδας μέσα στο κυρίως πρόγραμμά. Κάθε Λειτουργική μονάδα, μεταφράζεται (compilation) κατά την μετάφραση του κυρίως προγράμματος. Οι Λειτουργικές μονάδες ορίζονται στην περιοχή κάτω από την δήλωση PROGRAM, με την εντολή

USES Λειτουργική_Μονάδα1, Λειτουργική_Μονάδα2 κ.ο.κ.

Η Turbo Pascal περιέχει λειτουργικές μονάδες που περιέχουν πολύ χρήσιμες έτοιμες διαδικασίες και συναρτήσεις τις οποίες ο προγραμματιστής τις καλεί με απλή αναφορά του ονόματος τους όπως τις κοινές εντολές (εφόσον βέβαια έχει δηλώσει τη χρήση της αντίστοιχης λειτουργικής μονάδας). Οι πλέον χρησιμοποιούμενες λειτουργικές μονάδες είναι η CRT και η GRAPH. Η λειτουργική μονάδα CRT περιέχει εντολές (διαδικασίες) διαχείρισης της οθόνης όπως την εντολή clrscr αποτέλεσμα της οποίας είναι ο καθαρισμός της οθόνης ή την εντολή Gotoxy() που μετακινεί το δείκτη σε συγκεκριμένη θέση στην οθόνη ενώ η λειτουργική μονάδα GRAPH περιέχει εντολές σχεδίασης γραφικών.

Παράδειγμα

```
Program TEST
USES crt;
Begin
    Clrscr;
    gotoxy(10,10);
    write ('δώσε έναν αριθμό')
    ........
end.
```

4.1. ΔΙΑΔΙΚΑΣΙΕΣ

Διαδικασία είναι ένα αυτόνομα τμήμα προγράμματος το οποίο πραγματοποιεί μία ή περισσότερες ανεξάρτητες λειτουργίες. Μια διαδικασία αποτελείται από το όνομά της, το Begin για να δηλώσει την έναρξη της διαδικασίας και το End για να δηλώσει το τέλος της διαδικασίας συνοδευόμενο από ένα ελληνικό ερωτηματικό (;). Μέσα στην διαδικασία μπορεί να υπάρχει οποιαδήποτε εντολή της Turbo Pascal. Μια διαδικασία μπορεί να κληθεί από το κυρίως πρόγραμμα ή από άλλη διαδικασία ή συνάρτηση, προκειμένου να εκτελέσει τη λειτουργία για την οποία γράφτηκε και στη συνέχεια ο έλεγχος του προγράμματος επιστρέφει στην επόμενη εντολή απ' αυτή που κάλεσε την διαδικασία.

Οι διαδικασίες μπορούν να χρησιμοποιούν τοπικές (local) ή καθολικές (global) μεταβλητές. Τοπικές μεταβλητές είναι αυτές που ορίζονται από την διαδικασία και διατηρούν την τιμή τους καθ' όλη τη διάρκειά της διαδικασίας και αποτελούν ξεχωριστές οντότητες από τυχόν άλλες συνώνυμες μεταβλητές που υπάρχουν στο κυρίως πρόγραμμα ή σε άλλες διαδικασίες ή συναρτήσεις. Αντίθετα, οι καθολικές μεταβλητές είναι μεταβλητές που διατηρούν την τιμή τους καθ' όλη τη διάρκεια του προγράμματος και είναι προσπελάσιμες από οποιοδήποτε μέρος του προγράμματος (διαδικασία ή συνάρτηση). Φυσικά, τυχόν συνωνυμία τους με άλλες μεταβλητές δεν θα γίνει δεκτή από τον μεταφραστή (compiler).

Οι μεταβλητές που ορίζονται στην περιοχή VAR μέσα σε μια διαδικασία ή συνάρτηση ορίζονται εξ' ορισμού ως τοπικές μεταβλητές.

4.1.2. Ορισμός Διαδικασιών

Μια διαδικασία ορίζεται με την εντολή PROCEDURE. Η σύνταξη της εντολής είναι

```
PROCEDURE Όνομα_Διαδικασίας(Λίστα_Τυπικών_Παραμέτρων);
CONST
    Όνομα_Σταθεράς1 = Τιμή1;
    Όνομα_Σταθεράς2 = Τιμή2;
VAR
    Ονομα_Μεταβλητής1 : Τύπος_Δεδομένων1;
    Ονομα_Μεταβλητής2 : Τύπος_Δεδομένων2;
BEGIN
    Εντολές;
    ..............
END;
```

Το όνομα της διαδικασίας πρέπει να είναι μοναδικό σε όλο το πρόγραμμα και δεν πρέπει να εμφανίζεται σε άλλο σημείο του προγράμματος.

Η Λίστα_Τυπικών_Παραμέτρων περιέχει σταθερές, μεταβλητές και πίνακες ή γενικότερα τύπους δεδομένων που υποστηρίζονται από την Turbo Pascal. Οι παράμετροι χωρίζονται με κόμμα. Επίσης, η κάθε μεταβλητή που ορίζεται για πρώτη φορά στην Λίστα_Τυπικών_Παραμέτρων πρέπει να δηλώνεται και ο τύπος της.

Π.χ. PROCEDURE Display_Name(Title : String, TLng : Integer);

Στην περιοχή CONST, ορίζονται οι τυχόν σταθερές που μπορεί να χρησιμοποιούνται μέσα στη διαδικασία. Στην περιοχή VAR, ορίζονται οι τοπικές μεταβλητές της διαδικασίας και οι οποίες είναι ορατές μόνο μέσα στη διαδικασία αυτή. Ανάμεσα στο BEGIN και το END, μπορεί να υπάρχει οποιοσδήποτε αριθμός εντολών. Μέσα σε μια διαδικασία δεν μπορεί να έχει οριστεί μια άλλη διαδικασία. Η κάθε διαδικασία είναι αυτόνομη και διατηρεί την ακεραιότητά της.

Παράδειγμα. Η επόμενη διαδικασία πραγματοποιεί τη λύση των συντελεστών a και b και επιστρέφει ως αποτέλεσμα την τιμή του x εφόσον η εξίσωση έχει λύση, καθώς και την τιμή μιας μεταβλητής flag, η οποία είναι 1 αν υπάρχει λύση, διαφορετικά είναι 0. Η λειτουργία του Var πριν από τις παραμέτρους θα εξηγηθεί στη συνέχεια.

```
PROCEDURE EqSolve1(VAR a,b,x,flag :
Integer);
BEGIN
    flag:=1;x:=0;
    IF a=0 THEN flag=0
    ELSE x:=-b/a;
END;
```

Μια διαδικασία μπορεί να κληθεί από το κυρίως πρόγραμμα ή από άλλες διαδικασίες με το όνομά της.

Γενική κλήση μιας διαδικασίας είναι:

Όνομα_Διαδικασίας (Λίστα_Πραγματικών_Παραμέτρων);

π.χ. EqSolve1(c, d, z, myflag);

Όταν καλείται μια διαδικασία, οι πραγματικές παράμετροι που στέλνονται σε αυτήν τοποθετούνται σε σειρά σε μια ειδική περιοχή στην μνήμη που καλείται σωρός (stack). Η διαδικασίες και οι συναρτήσεις διαβάζουν τα δεδομένα αυτά με την σειρά που εξάγονται από τον σωρό.

Ο αριθμός και ο τύπος των ορισμάτων κατά την κλήση μιας διαδικασίας πρέπει να είναι ίδιος με τον αριθμό και τον τύπο της λίστας παραμέτρων της διαδικασίας. Οι παράμετροι και τα ορίσματα πρέπει να δίνονται με την ίδια σειρά και χωρίζονται με κόμμα.

Παράδειγμα.

```
TYPE
    ArrType : array[1..10] of Integer;

PROCEDURE VectorSum( Var ArrName : ArrType;
                         n,s : Integer );
BEGIN
    s:=0;
    FOR i:=1 to n DO
        s := s + ArrName[i];
END;
```

Σε μια διαδικασία ο ορισμός ενός πίνακα στην λίστα τυπικών παραμέτρων γίνεται με τον ακόλουθο τρόπο:

PROCEDURE όνομα_Διαδικασίας(Όνομα_Πίνακα : Τύπος_Πίνακα);

π.χ. PROCEDURE VectorSum(VAR ArrName : ArrType);

Αν σε μια διαδικασία δεν απαιτείται να περαστεί ως παράμετρος ολόκληρος ο πίνακας, μπορεί να μεταβιβαστούν μόνο τα στοιχεία του πίνακα που χρειάζονται. Για να μεταβιβαστεί ένα μόνο στοιχείο του πίνακα αρκεί να προστεθούν και οι τιμές των δεικτών του στοιχείου μέσα στις πραγματικές παραμέτρους κλήσης της διαδικασίας.

π.χ. Vector(ArrName[3]);

Εγγραφές και πεδία. Σε μια διαδικασία μπορούν να μεταβιβαστούν και ολόκληρες εγγραφές ή ατομικά πεδία εγγραφής. Στο επόμενο παράδειγμα αφού ορισθεί ο τύπος της εγγραφής και δηλωθεί μια μεταβλητή εγγραφής μεταβιβάζεται ολόκληρο το περιεχόμενό της στη διαδικασία PrintRec.

```
TYPE StockElem = RECORD
    Code : String[4];
    Description : String[20];
    UnitPrice : Single;
    Quantity : Integer;
END;

VAR
    StockRec : StockElem;

PROCEDURE PrintRec(VAR RecVar : StockElem);
BEGIN
    ............
END;

BEGIN
    ...........
    PrintRec(StockRec);
    ...........
END.
```

Ατομικά πεδία μπορούν να μεταβιβαστούν αν στη λίστα πραγματικών παραμέτρων αναφερθούν τα ονόματά τους. Στο ίδιο παράδειγμα μεταβιβάζεται στη διαδικασία SearchKey μόνο το πεδίο Code.

```
PROCEDURE SearchKey(VAR CodVar:String[4]);
```

```
BEGIN
    ........
END;

BEGIN
    ..............
SearchKey(StockElem.Code);
..............
END.
```

Το StockElem είναι η μεταβλητή εγγραφής που χρησιμοποιείται για την διαχείριση των εγγραφών. Κάθε φορά που γίνεται μια αναφορά στα πεδία της εγγραφής StockElem, πρέπει να γίνεται ως εξής:

StockElem.Code, για τον κωδικό,

StockElem.Description για την περιγραφή, κ.ο.κ.

Η Turbo Pascal περιλαμβάνει μια εντολή με την οποία ο χρήστης διευκολύνεται αρκετά κάθε φορά που απαιτείται αναφορά σε εγγραφές. Η εντολή αυτή είναι WITH Μεταβλητή_Εγγραφής DO. Οι εντολές περικλείονται σε BEGIN και END και ολοκληρώνεται με το ελληνικό ερωτηματικό π.χ.

```
WITH StockElem DO
BEGIN
    Readln(Code);
    Readln(Description);
    Readln(UnitPrice),
    Readln(Quantity);
END;
```

Οι παράμετροι στη Pascal μπορούν να περάσουν με δύο τρόπους με τιμή ή με αναφορά.

Πέρασμα παραμέτρων με αναφορά

Οι παράμετροι που περνούν με αναφορά έχουν υποχρεωτικά πριν από τη δήλωση τους στη λίστα των τυπικών παραμέτρων τη λέξη VAR. Μία παράμετρος η οποία περνάει με αναφορά σημαίνει απλά ότι θα επιστρέψει τη νέα της τιμή μετά το τέλος της διαδικασίας.

Οι πραγματικές παράμετροι στον ορισμό μιας διαδικασίας ή συνάρτησης περνούν με αναφορά όταν πρέπει τα δεδομένα να προσπελαστούν απευθείας. Αυτό σημαίνει ότι αντί να χρησιμοποιείται η στοίβα για την αποθήκευση των πραγματικών παραμέτρων, η Turbo Pascal αποθηκεύει έναν δείκτη που δείχνει στη διεύθυνση μνήμης όπου βρίσκεται η πραγματική μεταβλητή. Καθώς η διαδικασία εκτελείται, κάθε αναφορά στις τυπικές παραμέτρους, περνιούνται μέσω του δείκτη στις πραγματικές.

Έτσι, όταν αλλάζει το περιεχόμενο μιας τυπικής παραμέτρου στην ουσία αλλάζει και της πραγματικής.

Πέρασμα παραμέτρων με τιμή

Σε αυτή την περίπτωση, οι τιμές των πραγματικών παραμέτρων μιας διαδικασίας ή συνάρτησης, αντιγράφονται στη στοίβα. Οι τυπικές παράμετροι χρησιμοποιούν απλά ένα αντίγραφο της τιμής που βρίσκεται στη στοίβα. Οι αλλαγές στις τυπικές παραμέτρους δεν αλλάζουν την τιμή των πραγματικών παραμέτρων.

Τα δύο είδη περάσματος των παραμέτρων φαίνονται στο παρακάτω τμήμα προγράμματος όπου η μεταβλητή A περνάει με αναφορά ενώ η B με τιμή. Έτσι η μεταβλητή D όταν μεταβάλλεται από τη διαδικασία test αλλάζει και την τιμή της μεταβλητής B ενώ η μεταβλητή A μένει αναλλοίωτη.

```
.....
Procedure test(C:integer, Var D:Integer);
Begin
    C:=C*C;
    D:=D*D;
End;

......
A:=10;
B:=20;
test (A,B)
Write (A,B)
....
```

Το πρόγραμμα θα τυπώσει τους αριθμούς: 10 400

4.2. ΣΥΝΑΡΤΗΣΕΙΣ

Μια συνάρτηση είναι μια διαδικασία που επιστρέφει μια τιμή με το όνομά της. Η συνάρτηση διαφέρει από τη διαδικασία γιατί ενώ μπορεί να δεχθεί πολλές παραμέτρους, επιστρέφει πάντα μια μόνο τιμή. Οι συναρτήσεις μπορούν να εκτελούν οποιαδήποτε λειτουργία επιθυμεί ο χρήστης γι' αυτό και καλούνται συναρτήσεις ορισμένες από τον χρήστη (user-defined functions) σε αντιδιαστολή με τις ενσωματωμένες συναρτήσεις της γλώσσας. Για να ορισθεί μια συνάρτηση χρησιμοποιείται η εντολή FUNCTION.

Μία συνάρτηση όπως έχει ήδη αναφερθεί, μπορεί να κληθεί μέσα από το κυρίως πρόγραμμα, άλλες διαδικασίες ή συναρτήσεις του προγράμματος. Οι συναρτήσεις όπως και οι διαδικασίες στην Turbo Pascal έχουν την δυνατότητα της αναδρομικής κλήσης τους (recursive call) - η ιδιότητα της διαδικασίας ή της συνάρτησης να καλεί τον εαυτό της μέσα από την ίδια.

Για το όνομα μιας συνάρτησης ισχύουν ότι και για τις διαδικασίες. Στη συνάρτηση υπάρχει η δυνατότητα όπως περιγράφεται παρακάτω, χρησιμοποίησης τοπικών αλλά και καθολικών μεταβλητών.

4.2.2. Ορισμός Συναρτήσεων

Μια συνάρτηση ορίζεται με την εντολή FUNCTION. Η σύνταξη της εντολής είναι:

FUNCTION όνομα_συνάρτησης (Λίστα_Τυπικών_Παραμέτρων) : Τύπος_Δεδομένου

Το όνομα_συνάρτησης δεν μπορεί να αποδοθεί σε άλλη διαδικασία ή συνάρτηση. Για τη λίστα τυπικών παραμέτρων ισχύουν όσα αναφέρθηκαν για τις διαδικασίες. Μετά τη Λίστα τυπικών παραμέτρων ακολουθεί η άνω και κάτω τελεία και συνεχίζεται ο προσδιορι-

σμός του τύπου δεδομένου που θα επιστρέψει η συνάρτηση. Άρα, η κάθε συνάρτηση επιστρέφει πάντοτε μια τιμή. Έτσι, η κλήση της συνάρτησης δεν γίνεται μόνο καλώντας το όνομά της, αλλά εκχωρώντας την σε μια μεταβλητή ίδιου τύπου με αυτόν που θα εξάγει η συνάρτηση.

π.χ. αν υφίσταται μια συνάρτηση FUNCTION Func1(a : integer) : Longint;

η κλήση της θα γίνει ως εξής:

a := Func1(variable1);

όπου η μεταβλητή a πρέπει να είναι τύπου Longint σύμφωνα με τον τύπο δεδομένου που επιστρέφει η συνάρτηση.

Οι εντολές μιας συνάρτησης περικλείονται σε BEGIN και END. Μεταξύ των εντολών αυτών μπορούν να υπάρχουν οποιεσδήποτε εντολές εκτός από αυτές που ορίζουν συναρτήσεις ή άλλες διαδικασίες.

Παράδειγμα. Μέγιστος Κοινός Διαιρέτης

```
FUNCTION MKD(a,b : longint) : longint;
BEGIN
    writeln(a:10, b:10);
    while b<>0 do
    begin
        c := a mod b;
        a := b;
        b := c;
        writeln(a:10, b:10);
    end;
    MKD := a;
END;
```

Η συνάρτηση του παραδείγματος, βρίσκει το μέγιστο κοινό διαιρέτη. Η διαφορά των συναρτήσεων με τις διαδικασίες είναι ότι οι συναρτήσεις επιστρέφουν πάντα μια τιμή, που εκχωρείται συνήθως σε μια μεταβλητή. Στις εντολές της συνάρτησης του παραδείγματος υπάρχει η εντολή MKD:=a. Με τον τρόπο αυτό, επιτυγχάνεται επιστροφή της τιμής στο κυρίως πρόγραμμα. Αν η εντολή αυτή δεν υπήρχε, η συνάρτηση δεν θα είχε επιστρέψει κανένα δεδομένο στο κυρίως πρόγραμμα.

4.3. ΑΝΑΔΡΟΜΗ

Ο όρος αναδρομή αναφέρεται στη δυνατότητα μιας διαδικασίας να χρησιμοποιεί την ίδια την διαδικασία δηλαδή να μπορεί να καλεί τον εαυτό της. Οι διαδικασίες και οι συναρτήσεις μπορεί να είναι αναδρομικές (recursive). Κλασικό παράδειγμα αναδρομής είναι ο υπολογισμός του n! (παραγοντικού). Το παραγοντικό ενός μη αρνητικού αριθμού n ορίζεται ως εξής:

n! = n*(n-1)! και

0! = 1

π.χ. για n=5 ⇒ 5!=5*4*3*2*1=120

Η αναδρομική συνάρτηση Fact υπολογισμού του n! υλοποιείται στο επόμενο πρόγραμμα:

```
PROGRAM Fact1;
VAR
    number : longint;
FUNCTION Fact(n : longint) : longint;
BEGIN
    IF n=1 THEN
        Fact := 1
    ELSE
        Fact := n * Fact(n-1);
END;
BEGIN
    write('Εισάγετε έναν αριθμό από το 1
                        έως το 20 : ');
    readln(number);
    writeln('Το παραγοντικό του αριθμού
    ',number:2,' είναι ',Fact(number));
END.
```

1. ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ

1.1. ΣΥΝΟΛΟ ΧΑΡΑΚΤΗΡΩΝ

Το σύνολο των χαρακτήρων της QuickBASIC (QB) απαρτίζεται από τους αριθμητικούς χαρακτήρες (0-9), τους αλφαβητικούς, κεφαλαία (A-Z) και πεζά (a-z), καθώς και τους περισσότερους ειδικούς.

1.2. ΣΤΑΘΕΡΕΣ

1.2.1 Αριθμητικές σταθερές.

Υπάρχουν τέσσερις τύποι:

- ακέραιοι σε 2 bytes με τιμές από -32768 μέχρι 32767.
- "μακρυοί" ακέραιοι (long integers) σε 4 bytes με τιμές από -2.147.483.648 μέχρι 2.147.483.647.
- πραγματικοί απλής ακρίβειας (real single precision) σε 4 bytes με τιμές τάξης μεγέθους 10^{38} και ακρίβεια 6-7 δεκαδικών ψηφίων.
- πραγματικοί διπλής ακρίβειας (real double precision) σε 8 bytes με τιμές τάξης μεγέθους 10^{308} και ακρίβεια 15-16 δεκαδικών ψηφίων.

1.2.2 Αλφαριθμητικές σταθερές.

Μια αλφαριθμητική σταθερά είναι μια στοιχειοσειρά (string) με μήκος μέχρι 32767 χαρακτήρες περικλειόμενη σε εισαγωγικά.

1.2.3 Συμβολικές σταθερές.

Οι συμβολικές σταθερές ορίζονται με την εντολή CONST και μπορούν να χρησιμοποιηθούν αντί των αριθμητικών ή αλφαριθμητικών σταθερών. Π.χ.

```
CONST MaxElem% = 255
```

Ορίζεται η συμβολική σταθερά MaxElem ίση με 255.

Οι συμβολικές σταθερές μπορεί να είναι οποιουδήποτε τύπου και το όνομά τους σχηματίζεται με βάση τους κανόνες δημιουργίας μεταβλητών της QB. Αν χρησιμοποιηθούν οι χαρακτήρες %, &, !, # και $ για τον προσδιορισμό του τύπου της σταθεράς, τότε οι χαρακτήρες αυτοί **δεν** αποτελούν μέρος του ονόματος. Οι συμβολικές σταθερές δεν επηρεάζονται από τις δηλωτικές εντολές DEF.

1.3. ΜΕΤΑΒΛΗΤΕΣ

Μια μεταβλητή είναι ένα όνομα το οποίο αναφέρεται σε ένα αντικείμενο (έναν αριθμό, μία στοιχειοσειρά ή μια εγγραφή). Το όνομα μιας μεταβλητής σχηματίζεται από 40 το πολύ αριθμητικούς και αλφαβητικούς χαρακτήρες με τον πρώτο υποχρεωτικά αλφαβητικό. Στους αλφαβητικούς χαρακτήρες των ονομάτων μεταβλητών δεν έχει σημασία, αν είναι κεφαλαία ή πεζά γράμματα. Η QB δεν ξεχωρίζει τα κεφαλαία από τα πεζά γράμματα. Ετσι τα ονόματα NUMBER, number και NumBeR αναφέρονται στην ίδια θέση μνήμης, πρόκειται δηλ. για ίδιες μεταβλητές. Ο χρήστης μπορεί να χρησιμοποιεί κεφαλαία ή πεζά γράμματα κατά την κρίση του. Συνήθως οι μεταβλητές ονοματίζονται έτσι, ώστε το όνομα να προσδιορίζει και το περιεχόμενο. Στην περίπτωση αυτή χρησιμοποιούνται και κεφαλαία γράμματα για έμφαση π.χ. InventoryItem, StockElement, κ.λπ.

Μια μεταβλητή μπορεί να είναι αριθμητική, αλφαριθμητική ή τύπου εγγραφής. Ο προσδιορισμός του τύπου μιας μεταβλητής μπορεί να γίνει με έναν από τους επόμενους τρόπους.

1. Χρησιμοποιώντας τους παρακάτω ειδικούς χαρακτήρες στο τέλος του ονόματος της μεταβλητής.

% για ακέραιες π.χ. a%, Delta%

& για μακρυές ακέραιες π.χ. a&, Value&

! για πραγματικές απλής ακρίβειας π.χ. a!, Sum!

για πραγματικές διπλής ακρίβειας π.χ. a#, Total#

$ για αλφαριθμητικές π.χ. a$, Nam$

Οι χαρακτήρες αυτοί αποτελούν μέρος του ονόματος της μεταβλητής, γι' αυτό και οι μεταβλητές a%, a&, a!, a# και a$ είναι όλες διαφορετικές μεταξύ τους. Είναι διαφορετικές επίσης και από την a η οποία ορίζεται κατ' αρχήν (by default) ως μεταβλητή απλής ακρίβειας.

2. Δηλώνοντας τον τύπο της μεταβλητής σε δηλωτικές εντολές του τύπου:

```
εντολή όνομα-μεταβλητής AS τύπος
```

όπου η εντολή μπορεί να είναι μία από τις DIM, COMMON, REDIM, SHARED ή STATIC και ο τύπος ένας από τους INTEGER, LONG, SINGLE, DOUBLE, STRING ή τύπος δεδομένων από τον χρήστη.

Π.χ. με την εντολή

```
DIM x AS LONG
```

η μεταβλητή x ορίζεται μακρύς ακέραιος.

Στον ορισμό αλφαριθμητικών μεταβλητών υπάρχουν δύο δυνατότητες, μεταβλητές σταθερού ή μεταβλητού μήκους. Π.χ. με την εντολή

```
DIM Nam1 AS STRING
```

ορίζεται μια μεταβλητή με μήκος που μπορεί να εκτείνεται από 0 μέχρι 32767 χαρακτήρες.

Με την εντολή

```
DIM Nam2 AS STRING*20
```

ορίζεται μια μεταβλητή με μήκος ακριβώς 20 χαρακτήρες.

Με τον ίδιο τρόπο μπορεί να δηλωθεί μια μεταβλητή εγγραφής, αρκεί προηγούμενα να έχει ορισθεί η εγγραφή με την εντολή TYPE. Π.χ.

```
TYPE StockItem
    Code            AS STRING * 4
    Description     AS STRING * 20
    UnitPrice       AS SINGLE
    Quantity        AS LONG
END TYPE
DIM CurrentItem AS StockItem
```

Μετά τον ορισμό μιας εγγραφής η αναφορά σε ένα πεδίο της μπορεί να γίνει χρησιμοποιώντας τον τύπο *όνομα-εγγραφής.όνομα-πεδίου,* π.χ. η μεταβλητή StockItem.Code αναφέρεται στο πεδίο κωδικός της εγγραφής με τό όνομα StockItem.

Ας σημειωθεί ότι ο χαρακτήρας τελεία (.) μπορεί να χρησιμοποιηθεί μόνο σε μεταβλητές τύπου εγγραφής.

3. Με τη χρήση των δηλωτικών εντολών DEFINT, DEFLNG, DEFSNG, DEFDBL και DEFSTR για ακέραιους, μακρυούς ακέραιους, πραγματικούς απλής ακρίβειας, πραγματικούς διπλής ακρίβειας και αλφαριθμητικές μεταβλητές αντίστοιχα. Με τις εντολές αυτές μεταβλητές που αρχίζουν από ένα γράμμα ή περιοχή γραμμάτων είναι συγκεκριμένου τύπου. Οι εντολές DEF επηρεάζουν μόνο μεταβλητές που ανήκουν στην ενότητα που αναφέρονται.

Π.χ. με την εντολή

```
DEFINT A-Z
```

ορίζονται όλες οι μεταβλητές ως ακέραιες.

1.4. ΠΙΝΑΚΕΣ

Ενας πίνακας είναι ένα σύνολο αντικειμένων επί των οποίων η αναφορά γίνεται με το ίδιο όνομα μεταβλητής. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχεία γίνεται με το όνομα του πίνακα ακολουθούμενο από έναν ή περισσότερους δείκτες μέσα σε παρενθέσεις. Οι δείκτες προσδιορίζουν την τάξη του στοιχείου μέσα στον πίνακα. Π.χ.

```
A(3), Array(5,8), Nam$(20)
```

Σε ένα πίνακα ο αριθμός των δεικτών προσδιορίζει το πλήθος των διαστάσεων, ενώ η μέγιστη δυνατή τιμή κάθε δείκτη προσδιορίζει το πλήθος των στοιχείων του πίνακα ανά διάσταση. Πριν χρησιμοποιηθεί ένας πίνακας πρέπει να ορισθούν οι διαστάσεις του. Με τον όρο αυτό αναφερόμαστε τόσο στον ορισμό του πλήθους των διαστάσεων, όσο και των μέγιστων τιμών κάθε μιάς. Ο ορισμός των διαστάσεων ενός πίνακα επιτυγχάνεται με την εντολή DIM. Ο μέγιστος αριθμός των διαστάσεων ενός πίνακα είναι 60 και η μέγιστη τιμή μιας διάστασης 32767. Είναι δυνατόν ένας πίνακας να χρησιμοποιηθεί χωρίς να ορισθούν οι διαστάσεις με την DIM, αρκεί ο μέγιστος αριθμός στοιχείων ανά διάσταση να μην ξεπεράσει το 10. Με μία εντολή DIM δηλώνεται η μέγιστη τιμή κάθε δείκτη, οπότε ελάχιστη θεωρείται η μηδενική π.χ. DIM A(100). Μπορεί να χρησιμοποιηθεί όμως και ο τύπος από-έως, οπότε προσδιορίζονται μαζί τα άνω και κάτω όρια π.χ. DIM A(1 TO 100). Να σημειωθεί η πολύ σπουδαία δυνατότητα να ορίζονται με τον τύπο αυτό και αρνητικές τιμές δεικτών π.χ. DIM A(-5 TO 5). Ετσι μπορεί να ορισθεί οποιαδήποτε περιοχή τιμών των δεικτών από -32768 έως 32767.

Τέλος με την DIM όλα τα στοιχεία του πίνακα λαμβάνουν αρχική τιμή μηδέν ή null.

Για ονόματα πινάκων μπορούν να χρησιμοποιηθούν οποιεσδήποτε μεταβλητές οποιουδήποτε τύπου. Ως δείκτες χρησιμοποιούνται ακέραιοι ή ακέραιες εκφράσεις (μπορούν να χρησιμοποιηθούν και μη ακέραιοι αλλά στρογγυλοποιούνται πριν από τη χρήση). Για το ορισμό ενός πίνακα με στοιχεία εγγραφές αρχείου πρέπει πρώτα να δηλωθεί ο τύπος της εγγραφής και μετά οι διαστάσεις του πίνακα. Π.χ.

```
TYPE QueueNode
    Data    AS STRING * 20
    Pointer AS INTEGER
END TYPE
DIM Q(100) AS QueueNode
```

Κάθε στοιχείο του πίνακα Q είναι μία εγγραφή τύπου QueueNode. Η αναφορά σε ένα πεδίο της εγγραφής σαν στοιχείο πίνακα γίνεται με τη χρήση του τύπου *ονομα - πίνακα.πεδίο - εγγραφής,* π.χ.

```
PRINT Q(i).Data
```

1.5 ΕΚΦΡΑΣΕΙΣ

Για τη διαμόρφωση των εκφράσεων χρησιμοποιούνται σταθερές, μεταβλητές, τελεστές, συναρτήσεις και παρενθέσεις με τη συνήθη ιεραρχία. Υπάρχουν οι γνωστοί από τη ΓΛΩΣΣΑ αριθμητικοί και συγκριτικοί τελεστές με τη διαφορά ότι ως τελεστής ακέραιας διαίρεσης χρησιμοποιείται η ανάποδη κάθετος "\".

Οι σπουδαιότεροι λογικοί τελεστές είναι οι AND, OR και NOT. Ακόμη ο τελεστής "+" χρησιμοποιείται και για τη συνένωση δύο στοιχειοσειρών (strings) σε μία.

Τέλος η QB διαθέτει ένα πλουσιότατο ρεπερτόριο ενσωματωμένων συναρτήσεων μαθηματικών, αλφαριθμητικών, κ.α. (βλ. βιβλιογραφία).

Κυριότερες μαθηματικές συναρτήσεις	
ABS	Απόλυτη τιμή
ATN	Τόξο εφαπτομένης
COS	Συνημίτονο
EXP	e^x
FIX	Ακέραιο μέρος
INT	Μικρότερος ακέραιος
LOG	Φυσικός λογάριθμος
RND	Τυχαίος αριθμός
SIN	Ημίτονο
SQR	Τετραγωνική ρίζα
TAN	Εφαπτομένη

1.6 ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Ενα πρόγραμμα QB αποτελείται από μία ή περισσότερες **ενότητες** (modules). Μια ενότητα είναι ένα αρχείο πηγαίου κώδικα (source file), το οποίο μπορεί να μεταγλωττισθεί ξεχωριστά. Ολες οι εντολές της γλώσσας μπορούν να υπάρχουν σε μία ενότητα.

Ενα πρόγραμμα αποτελείται από γραμμές προγράμματος. Κάθε γραμμή προγράμματος περιέχει μια ή περισσότερες εντολές προγράμματος. Προαιρετικά μπορεί να υπάρχουν στο τέλος της γραμμής σχόλια. Οπως και στον ορισμό μεταβλητών τα κεφαλαία γράμματα δεν ξεχωρίζουν από τα πεζά.

Οι εντολές διακρίνονται σε εκτελέσιμες και μη. Ολες οι εντολές είναι εκτελέσιμες εκτός από τις:

REM ή ‘, COMMON, CONST, DATA, DECLARE, DEF τύπος, DIM, OPTION BASE, SHARED, STATIC, TYPE...END TYPE.

Αν υπάρχουν περισσότερες από μία εντολές σε μια γραμμή, τότε πρέπει να χωρίζονται με το χαρακτήρα άνω-κάτω τελεία “:”. Μια γραμμή προγράμματος μπορεί να εκτείνεται σε περισσότερες από μία φυσικές γραμμές. Το μέγιστο μήκος για μία γραμμή προγράμματος είναι 256 χαρακτήρες.

2. ΕΚΧΩΡΗΣΗ ΤΙΜΩΝ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ

Η εντολή εκχώρησης είναι

```
Μεταβλητή = έκφραση
```

Ας σημειωθεί ότι στην QB ο τελεστής εκχώρησης είναι ο χαρακτήρας ίσον “=”, που χρησιμοποιείται και ως (συγκριτικός) τελεστής ισότητας. Στην εντολή εκχώρησης το ίσον αντιστοιχεί στο αριστερό βέλος που χρησιμοποιείται στην αλγοριθμική ψευδογλώσσα για τον ίδιο σκοπό.

Η εντολή SWAP a,b ανταλλάσει τα περιεχόμενα των δύο μεταβλητών.

Για την είσοδο δεδομένων από το πληκτρολόγιο χρησιμοποιούνται οι εντολές:

```
INPUT ["προτροπή"{;|,}] λίστα_μεταβλητών
```

Η προτροπή, αν υπάρχει, εμφανίζεται στην οθόνη και μετά ζητείται η εισαγωγή τιμών. Π.χ.

```
INPUT "Τιμή=",Value
```

Εάν μετά προτροπή υπάρχει ο χαρακτήρας “;”, τότε εμφανίζεται και ένα αγγλικό ερωτηματικό, ενώ αν υπάρχει ο χαρακτήρας “,”, τότε δεν εμφανίζεται τίποτα. Η ολοκλήρωση της εισαγωγής τιμών επισημαίνεται με την πληκτρολόγηση του Enter. Αν υπάρχουν πάνω από μια μεταβλητές στη *λίστα_μεταβλητών*, τότε οι εισαγόμενες τιμές χωρίζονται με κόμα.

Η εντολή LINE INPUT a$ πραγματοποιεί την εισαγωγή μιας στοιχειοσειράς, που μπορεί να αποτελείται από οποιουσδήποτε χαρακτήρες. Το τέλος του κειμένου πιστοποιείται με το Enter.

Η συνάρτηση INPUT$(n) δέχεται από το πληκτρολόγιο ακριβώς n χαρακτήρες, χωρίς να απαιτείται η πίεση του Enter. Π.χ.

```
z$=INPUT$(1)
```

Αντίθετα η συνάρτηση INKEY$ επιστρέφει ένα χαρακτήρα που διαβάστηκε από το πληκτρολόγιο ή την κενή στοιχειοσειρά, αν δεν περιμένει κανένας χαρακτήρας εκεί (βλ. 3.3.3.).

Εναλλακτική εντολή εισόδου είναι η READ *λίστα_μεταβλητών*, η οποία εκχωρεί τιμές στις μεταβλητές που βρίσκονται σε εντολές DATA. Π.χ.

```
READ a,b,c
DATA 10,-5,3000
```

Η εντολή READ βρίσκει χρήση στην ανάγνωση σταθερών τιμών σε ένα πρόγραμμα και κατά τη φάση ανάπτυξης του προγράμματος. Στη δεύτερη περίπτωση η είσοδος των τιμών στις μεταβλητές του προγράμματος γίνεται αρχικά με εντολές READ-DATA, προκειμένου ν’ αποφεύγεται η επαναπληκτρολόγηση σε κάθε εκτέλεσή του. Στο τέλος του ελέγχου οι εντολές αυτές αντικαθιστώνται από εντολές INPUT.

Εντολή εξόδου είναι η

```
PRINT λίστα_μεταβλητών
```

η οποία εμφανίζει τις τιμές των μεταβλητών στην οθόνη. Η ακριβής εμφάνιση των τιμών εξαρτάται από τους διαχωριστικούς χαρακτήρες στη *λίστα_μεταβλητών*. Αν υπάρχουν κόματα, τότε η εμφάνιση γίνεται σε σταθερές ζώνες των 14 χαρακτήρων, Αν υπάρχουν ερωτηματικά, τότε η εμφάνιση γίνεται σε συνεχόμενες θέσεις.

Μορφοποιημένη εμφάνιση τιμών επιτυγχάνεται με την εντολή

```
PRINT USING μάσκα; λίστα_μεταβλητών
```

Η *μάσκα* είναι μια στοιχειοσειρά από ειδικούς χαρακτήρες, που καθορίζει τον τρόπο εμφάνισης των τιμών των μεταβλητών (βλ. βιβλιογραφία).

Οι εντολές LPRINT και LPRINT USING πραγματοποιούν με τους ίδιους κανόνες εκτύπωση τιμών στον εκτυπωτή.

3. ΔΟΜΕΣ ΕΛΕΓΧΟΥ

Οι δομές ελέγχου είναι ένα εργαλείο που προσφέρουν οι διαδικασιακές γλώσσες προγραμματισμού για τον έλεγχο της σειράς εκτέλεσης των εντολών. Οι δομές ελέγχου είναι βασικά τρείς: η ακολουθία, η επιλογή και η επανάληψη ή ανακύκλωση. Οι δύο τελευταίες έχουν διάφορες παραλλαγές.

3.1 ΑΚΟΛΟΥΘΙΑ

Πρόκειται για την απλούστερη δομή κατά την οποία οι εντολές του προγράμματος εκτελούνται η μία μετά την άλλη με τη σειρά που γράφονται στο πρό-

γραμμα, δηλ. από την αρχή του κειμένου προς το τέλος και πάνω στην ίδια γραμμή από τ' αριστερά προς τα δεξιά. Μεταξύ δύο συνεχόμενων εντολών που βρίσκονται στην ίδια γραμμή απαιτείται η παρουσία ενός διαχωριστή εντολών, που εδώ είναι ο χαρακτήρας "άνω-κάτω τελεία" (:).

3.2 ΕΠΙΛΟΓΗ

Σε όλα τα επίπεδα του προγραμματισμού η απλούστερη απόκλιση από την ακολουθιακή δομή είναι το αλμα σε άλλο σημείο του προγράμματος με την εντολή GOTO. Οταν συναντάται η εντολή GOTO διακόπτεται η σειριακή εκτέλεση των εντολών και ο έλεγχος μεταφέρεται στο σημείο προορισμού της εντολής, γι' αυτό και το GOTO αναφέρεται πολλές φορές ως διακλάδωση άνευ όρων. Η εντολή GOTO ακολουθείται από μια διεύθυνση προορισμού που είναι μια αλφαριθμητική ετικέτα (label).

Η χρήση της GOTO στα προγράμματα παρουσιάζει αρκετά προβλήματα, γι' αυτό και οι πεπειραμένοι προγραμματιστές την αποφεύγουν. Αλλωστε η χρήση της δεν είναι απαραίτητη στον προγραμματισμό. Η QB διαθέτει την GOTO, κύρια για λόγους συμβατότητας με προηγούμενες εκδόσεις.

3.2.1. Σύνθετη επιλογή

Κατά την εκτέλεση ενός προγράμματος είναι δυνατό να επιλεγεί η εκτέλεση ορισμένων εντολών και να αποφευχθεί η εκτέλεση άλλων. Η πιο απλή δομή για κάτι τέτοιο είναι η IF..THEN..ELSE. Η σύνταξη της εντολής είναι:

```
IF B THEN E1 ELSE E2
```

Η B είναι μια λογική έκφραση (μια συνθήκη) και η οποία έχει τιμές: αληθής (true), αν η συνθήκη ισχύει και ψευδής (false), αν η συνθήκη δεν ισχύει. Οι E1 και E2 είναι εντολές ή ομάδες εντολών. Η λειτουργία της εντολής είναι η εξής: υπολογίζεται πρώτα η λογική τιμή της παράστασης B. Αν η τιμή της B είναι αληθής, τότε εκτελείται η εντολή E1, αλλιώς εκτελείται η E2.

Οι E1 και E2 μπορούν να είναι ατομικές εντολές ή σύνολο εντολών χωρισμένες με ":". Ολες σχεδόν οι εντολές της γλώσσας μπορούν να υπάρχουν μέσα στις E1 και E2, ακόμη και εντολές IF..THEN..ELSE, οπότε δημιουργούνται τα λεγόμενα εμφωλευμένα IF. Επίσης είναι δυνατόν το τμήμα ELSE να απουσιάζει, οπότε αν η συνθήκη δεν ισχύει, η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί την IF..THEN.

Μια εντολή IF..THEN..ELSE μπορεί να εκτείνεται σε περισσότερες από μία φυσικές γραμμές μέχρι ένα μέγιστο 255 χαρακτήρων.

Παράδειγμα. Ενας άνδρας χαρακτηρίζεται ελαφρύς (βαρύς) αν είναι κάτω (πάνω) από 80 κ. Επίσης χαρακτηρίζεται κοντός (ψηλός) αν είναι κάτω (πάνω) από 1.75. Στο επόμενο τμήμα προγράμματος εισάγονται το βάρος και ύψος ενός ατόμου και εξάγεται ο χαρακτηρισμός του με ένα σύνθετο IF.

```
INPUT "Βάρος =",B
INPUT "Υψος =",Y
IF B<80 THEN IF Y<1.75 THEN PRINT "κοντός
και ελαφρύς" ELSE PRINT "ψηλός και ελαφρύς"
ELSE IF Y<1.75 THEN PRINT "κοντός και βα-
ρύς" ELSE PRINT "ψηλός και βαρύς"
```

Για να αποφεύγονται αυτού του είδους πολύπλοκες εντολές, η QB διαθέτει έναν άλλο τύπο εντολής IF..THEN..ELSE, η οποία δεν περιορίζεται πλέον σε μία φυσική γραμμή προγράμματος. Η εντολή αυτή καλείται **ομαδοποιημένο** (block) IF, με την έννοια ότι οι ομάδες εντολών E1 και E2 μπορούν να εκτείνονται σε οποιονδήποτε αριθμό γραμμών προγράμματος. Επειδή τώρα το μήκος της εντολής δεν προκαθορίζεται, μία ειδική εντολή η END IF προσδιορίζει το τέλος της εντολής. Με τη νέα μορφή η εντολή IF στο προηγούμενο παράδειγμα γράφεται:

```
IF B<80 THEN
    IF Y<1.75 THEN
        PRINT "κοντός και ελαφρύς"
    ELSE
        PRINT "ψηλός και ελαφρύς"
    END IF
ELSE
    IF Y<1.75 THEN
        PRINT "κοντός και βαρύς"
    ELSE
        PRINT "ψηλός και βαρύς"
    END IF
END IF
```

Από το παράδειγμα αυτό γίνεται αμέσως φανερή η ευελιξία που προσφέρει στον προγραμματισμό αυτό ο τύπος της εντολής. Μάλιστα αυτό δεν επιδρά μόνο μόνο στη δημιουργία πολύπλοκων δομών ελέγχου, αλλά διευκολύνεται σημαντικά και η ανάγνωση και κατανόηση προγραμμάτων από κάθε τρίτον.

Ο τύπος αυτός της εντολής IF..THEN..ELSE διαθέτει και μια άλλη προαιρετική δήλωση την ELSEIF, με την οποία παρέχεται ακόμα μεγαλύτερη ευελιξία στη σύνταξη πολύπλοκων δομών. Π.χ.

```
INPUT "Δώσε έναν χαρακτήρα :",a$
IF a$>= "0" AND a$<="9" THEN
    PRINT "ψηφίο"
ELSEIF a$>= "A" AND a$<="z" THEN
    PRINT "λατινικό γράμμα"
ELSEIF a$>= "A" AND a$<="ω" THEN
    PRINT "ελληνικό γράμμα"
ELSE
    PRINT "μη αλφαριθμητικός χαρακτήρας"
END IF
```

Κάθε μία από τις δηλώσεις IF, ELSEIF και ELSE ακολουθείται από μια ομάδα εντολών, οι οποίες δεν πρέπει να βρίσκονται στην ίδια γραμμή με τις IF, ELSEIF και ELSE. Αλλιώς η QB θεωρεί την εντολή αυτή σαν μιας γραμμής.

Η QB εξετάζει κάθε μία από τις συνθήκες της IF και των ELSEIF δηλώσεων από πάνω προς τα κάτω υπερπηδώντας τις ομάδες των εντολών που ακολουθούν μέχρι να βρεθεί η πρώτη αληθής συνθήκη. Τότε εκτελούνται οι εντολές που αντιστοιχούν και μετά γίνεται διακλάδωση στην εντολή που ακολουθεί το END IF.

3.2.2 Πολλαπλή επιλογή

Με τον όρο αυτό αναφερόμαστε στη δυνατότητα μιας δομής να επιτυγχάνει την εκτέλεση μιας ομάδας εντολών ανάμεσα σε πολλές ανάλογα με την τιμή της εξεταζόμενης συνθήκης. Η σχετική εντολή είναι η SELECT..CASE. Πρόκειται για εντολή πολλαπλών επιλογών απόφασης ανάλογη, από πλευράς τελικού αποτελέσματος, με το ομαδοποιημένο IF..THEN..ELSE. Η βασική διαφορά της SELECT..CASE από το IF..THEN..ELSE είναι, ότι η πρώτη εξετάζει μια απλή έκφραση και ανάλογα με το αποτέλεσμα εκτελεί διαφορετικές εντολές, ενώ η δεύτερη εξετάζει απλές ή σύνθετες λογικές εκφράσεις. Αν και η λειτουργία της SELECT..CASE μπορεί να πραγματοποιηθεί και με την IF..THEN..ELSE, λόγω της συμπαγούς δομής της η χρήση της προσφέρει σημαντικά πλεονεκτήματα στον προγραμματιστή.

Η γενική μορφή της εντολής είναι:

```
SELECT CASE έκφραση
    CASE λίστα-τιμών-1
        εντολές-1
    CASE λίστα-τιμών-2
        εντολές-2
    ..............
    CASE ELSE
        εντολές-ν
    END SELECT
```

Η έκφραση μπορεί να είναι αριθμητική ή αλφαριθμητική. Οι *λίστες-τιμών* μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές, περιοχή τιμών από-έως ή τιμές που υπακούουν σε μια λογική συνθήκη. Η εντολή εξετάζει την έκφραση και ανάλογα με την τιμή της εκτελεί τις εντολές μετά την CASE που αντιστοιχεί στην τιμή αυτή. Αν η τιμή της έκφρασης δεν αντιστοιχεί σε καμμία από τις λίστες τιμών, τότε εκτελούνται οι εντολές που ακολουθούν την CASE ELSE. Μετά την εκτέλεση των εντολών κάποιας CASE, το πρόγραμμα συνεχίζεται με την εντολή που ακολουθεί την END SELECT.

Η λειτουργία της εντολής καθώς και οι δυνατότητές της αναδεικνύονται με τα επόμενα παραδείγματα.

Παράδειγμα 1.

```
INPUT "Δώσε ακέραιο από 1 έως 3 :" , i
SELECT CASE i
    CASE 1
        PRINT "Περίπτωση 1"
    CASE 2
        PRINT "Περίπτωση 2"
    CASE 3
        PRINT "Περίπτωση 3"
    CASE ELSE
        PRINT "Λάθος"
END SELECT
```

Παράδειγμα 2.

```
INPUT "Δώσε ακέραιο μεταξύ 0 και 9 :" , i
SELECT CASE i
    CASE 0
        PRINT "Μηδέν"
    CASE 1,3,5,7,9
        PRINT "Μονός"
    CASE 2,4,6,8
        PRINT "Ζυγός"
    CASE ELSE
        PRINT " ή 9 ή μη ακέραιος"
END SELECT
```

Εδώ στις περιπτώσεις μονού ή ζυγού η *λίστα-τιμών* περιέχει όλες τις δυνατές τιμές χωρισμένες με κόμματα.

Παράδειγμα 3.

```
INPUT "Δώσε αριθμό μεταξύ 1 και 20 :" , i
SELECT CASE i
CASE 1 TO 10
    PRINT "1η δεκάδα"
CASE 11 TO 20
    PRINT "2η δεκάδα"
    CASE ELSE
    PRINT "Λάθος"
END SELECT
```

Εδώ στη *λίστα-τιμών* χρησιμοποιείται περιοχή τιμών από-έως, όπου μεταξύ αρχικής και τελικής τιμής τίθεται η λέξη TO.

Παράδειγμα 4.

```
INPUT "Σε ποιά ηλικία άρχισες να μαθαίνεις
                                BASIC ; ",age
SELECT CASE age
    CASE IS<0
        PRINT "Είπαμε ηλικία"
    CASE IS<5
        PRINT "Μάλλον τα παραλές! "
    CASE 5 TO 80
        PRINT "Εντάξει"
    CASE IS>80
        PRINT "Κάλλιο αργά παρά ποτέ."
    CASE ELSE
        PRINT "Αδύνατον"
END SELECT
```

Εδώ χρησιμοποιείται η λέξη IS ακολουθούμενη από ένα συγκριτικό τελεστή και μια τιμή. Ετσι η περίπτωση CASE IS<5 πληρείται εφ' όσον age<5. Ως συγκριτικοί τελεστές μπορούν να χρησιμοποιηθούν όλοι οι γνωστοί τελεστές της BASIC δηλ. οι <, <=, >, >=, <> και =. Να σημειωθεί ότι αν μια τιμή της έκφρασης αντιστοιχεί σε περισσότερες από μία περιπτώσεις, τότε εκτελείται η πρώτη. Ετσι αν εισαχθεί αρνητικός αριθμός που είναι μικρότερος από το 0 και το 5, εκτελούνται οι εντολές που ακολουθούν την CASE IS<0.

Περισσότερες από μία συνθήκες ή περιοχές τιμών μπορούν να υπάρχουν σε μία CASE, όπως στα επόμενα παραδείγματα.

```
CASE -1, 0, 10 TO 20, 50 TO 90, IS>99
CASE i, j, k, -1 TO 1
CASE IS<"A", IS>"z"
CASE "α" TO "ω", "ά", "έ", "ή", "ί", "ό",
                                "ύ", "ώ"
```

3.3 ΕΠΑΝΑΛΗΨΗ 'Η ΑΝΑΚΥΚΛΩΣΗ

Η δομή της επανάληψης ή ανακύκλωσης αναφέρεται στη επαναληπτική εκτέλεση μιας ομάδας εντολών προγράμματος κατά ένα προκαθορισμένο αριθμό φορών ή μέχρι να ισχύσει κάποια συνθήκη. Η δομή της ανακύκλωσης υλοποιείται με εντολές FOR-NEXT, WHILE-WEND και DO-LOOP.

3.3.1 FOR-NEXT

Η εντολή αυτή εκτελεί τις εντολές προγράμματος που βρίσκονται μεταξύ του FOR και του NEXT για όλες τις τιμές της μεταβλητής ελέγχου. Η μεταβλητή ελέγχου που αναφέρεται στη FOR, συνοδεύεται από την αρχική και τελική τιμή, καθώς και το βήμα μεταβολής της. Ως γνωστόν αν το βήμα είναι 1 τότε παραλείπεται. Στην υλοποίηση της εντολής από την QB έχει προστεθεί μια νέα δυνατότητα, ότι η έξοδος από το βρόχο μπορεί να γίνει σε οποιοδήποτε σημείο με τη χρήση της εντολής EXIT FOR. Στο επόμενο παράδειγμα χρησιμοποιείται η εντολή FOR-NEXT για να διατρέξει διαδοχικά όλα τα στοιχεία του πίνακα k$(). Αν σημειωθεί ισότητα στον έλεγχο κάποιου στοιχείου του πίνακα με το ζητούμενο cle$, τότε η διαδικασία διακόπτεται με την EXIT FOR.

```
INPUT "Κωδικός : ",cle$
index = 0 : ex = 0
FOR i = 1 TO n
    IF cle$ = k$(i) THEN
        index = i : ex = 1 : EXIT FOR
    END IF
NEXT i
```

Στο τέλος λαμβάνεται η θέση του στοιχείου index, καθώς και μία μεταβλητή ex η οποία γίνεται 1 σε επιτυχημένη αναζήτηση.

3.3.2 WHILE-WEND

Με την εντολή αυτή εκτελούνται όλες οι εντολές προγράμματος που υπάρχουν μεταξύ WHILE και WEND όσο ισχύει η συνθήκη που συνοδεύει το WHILE. Στο επόμενο τμήμα προγράμματος η WHILE ελέγχει τη συνθήκη "οχι τέλος αρχείου" και εφόσον ισχύει δηλ. δεν είναι στο τέλος του αρχείου, τότε γίνεται ανάγνωση μιας εγγραφής.

```
OPEN "I",#1,"SEQFILE.DAT"
WHILE NOT EOF(1)
    LINE INPUT#1, Rec$
    PRINT Rec$
WEND
CLOSE
```

3.3.3. DO-LOOP

Παρόμοια στη λογική με τη WHILE-WEND αλλά με περισσότερες δυνατότητες είναι και η εντολή DO-LOOP. Με την εντολή αυτή εκτελούνται όλες οι εντολές προγράμματος που βρίσκονται μεταξύ του DO και του LOOP. Η εκτέλεση των εντολών επαναλαμβάνεται συνέχεια εφόσον μια συνθήκη ελέγχου παραμένει αληθής (WHILE) ή μέχρις ότου γίνει ψευδής (UNTIL). Ο έλεγχος της συνθήκης εξόδου μπορεί να γίνεται στην αρχή ή στο τέλος του βρόχου. Η έξοδος από το βρόχο μπορεί να γίνει σε οποιοδήποτε σημείο με τη χρήση της εντολής EXIT DO. Είναι δυνατόν επίσης να μην υπάρχει καθόλου συνθήκη ελέγχου, οπότε ο βρόχος εκτελείται επ' άπειρον. Στην περίπτωση αυτή η μοναδική δυνατότητα εξόδου παρέχεται με την EXIT DO.

Η DO-LOOP είναι μια εντολή ανακύκλωσης που μπορεί να χρησιμοποιηθεί σε κάθε περίπτωση. Η αξία της αναδεικνύεται με τα επόμενα παραδείγματα.

Παράδειγμα 1.

```
z$=""
DO UNTIL LEN(z$)<>0
    z$ = INKEY$
LOOP
```

Στο παράδειγμα αυτό ο βρόχος εκτελείται μέχρι να πληκτρολογηθεί κάποιος χαρακτήρας. Η συνθήκη ελέγχεται στην αρχή του βρόχου.

Παράδειγμα 2.

```
z$ = ""
DO WHILE LEN(Z$) = 0
    z$ = INKEY$
LOOP
```

Ακριβώς η ίδια λειτουργία με το προηγούμενο γίνεται και στο παράδειγμα αυτό. Ο βρόχος εκτελείται εφόσον κανείς χαρακτήρας δεν εισάγεται. Και εδώ ο έλεγχος γίνεται στην αρχή του βρόχου.

Παράδειγμα 3.

```
DO
    INPUT "Εντάξει (N/O) : "; z$
    z = INSTR("NOνο",z$)
LOOP UNTIL z<>0
```

Στο παράδειγμα αυτό ζητείται η εισαγωγή των χαρακτήρων Ν για ΝΑΙ και Ο για ΟΧΙ. Αν δεν δοθούν οι χαρακτήρες αυτοί και μόνο, καθώς και οι αντίστοιχοι πεζοί, τότε ο βρόχος επαναλαμβάνεται. Εδώ η συνθήκη εξετάζεται στο τέλος του βρόχου.

Παράδειγμα 4.

```
i% = INT(RND(1)*10)
j% = INT(RND(1)*10)
DO
```

```
    PRINT i% ; "*" ; j% ; "=";
    INPUT "",k%
LOOP WHILE k%<>i% * j%
```

Στο παράδειγμα αυτό ζητείται η απάντηση του χρήστη στην ερώτηση ποιο είναι το γινόμενο των αριθμών i% επί j%. Εφόσον η απάντηση δεν είναι σωστή, ο βρόχος επαναλαμβάνεται. Και εδώ ο έλεγχος της συνθήκης γίνεται στο τέλος.

Παράδειγμα 5.

```
s = 0
DO
    INPUT x
    IF x = 0 THEN EXIT DO
    s = s + x
LOOP
```

Στο παράδειγμα αυτό αθροίζονται τιμές που εισάγει ο χρήστης σε έναν αθροιστή. Το πλήθος των τιμών δεν είναι γνωστό εκ των προτέρων, γι' αυτό η έξοδος από το βρόχο γίνεται όταν πληκτρολογηθεί το μηδέν. Στην περίπτωση αυτή δεν υπάρχει καθόλου συνθήκη εξόδου στην αρχή ή το τέλος του βρόχου.

Από τα προηγούμενα γίνεται φανερό ότι ένας βρόχος μπορεί να υλοποιηθεί είτε με τη συνθήκη WHILE, είτε με την UNTIL. Να σημειωθεί τέλος ότι όταν ο έλεγχος της συνθήκης γίνεται στο τέλος του βρόχου, αυτός εκτελείται τουλάχιστον μία φορά.

4. ΔΙΑΔΙΚΑΣΙΕΣ : ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Η κυριότερη ίσως φροντίδα του προγραμματιστή εφαρμογών στην ανάπτυξη των προγραμμάτων είναι να επιτύχει τον καλύτερο χωρισμό ενός προγράμματος σε μικρότερες ενότητες. Κάθε ενότητα αφού πρώτα ελεχθεί χωριστά και έπειτα σε συνεργασία με άλλες, θεωρείται ότι έχει ολοκληρωθεί και δεν απομένει παρά η σύνδεσή της με τις υπόλοιπες προκειμένου να αποτελέσουν μαζί ένα ενιαίο λειτουργικό σύνολο, το ζητούμενο πρόγραμμα. Η QB παρέχει ένα μέσο κατάτμησης ενος μεγάλου προγράμματος σε μικρότερα, τις διαδικασίες (procedures). Ο όρος διαδικασία αναφέρεται στα **υποπρογράμματα** (subprograms) και τις **συναρτήσεις** (functions).

4.1 ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Υποπρόγραμμα είναι ένα αυτόνομο τμήμα προγράμματος το οποίο πραγματοποιεί μία ή περισσότερες ανεξάρτητες λειτουργίες. Ενα υποπρόγραμμα έχει ένα όνομα, μία αρχή και ένα τέλος και μπορεί να περιέχει οποιοδήποτε αριθμό εντολών. Ενα υποπρόγραμμα μπορεί να κληθεί από ένα πρόγραμμα, το κύριο πρόγραμμα ή άλλο υποπρόγραμμα προκειμένου να επιτελέσει τη λειτουργία για την οποία γράφτηκε. Σχετική με την έννοια του υποπρογράμματος είναι και η **υπορουτίνα** (subroutine). Υπορουτίνα είναι τμήμα προγράμματος στο οποίο μπορεί να πραγματοποιηθεί διακλάδωση με τη χρήση της εντολής GOSUB. Η επιστροφή από την υπορουτίνα γίνεται με την εντολή RETURN. Παρόλο που τα υποπρογράμματα και οι υπορουτίνες επιτελούν ουσιαστικά παρόμοιες λειτουργίες, διαφέρουν σε πολλά σημεία και για να αποφευχθεί οποιαδήποτε σύγχυση στο κεφάλαιο αυτό ο όρος υποπρόγραμμα αναφέρεται στο τμήμα προγράμματος που περικλείεται ανάμεσα στις εντολές SUB και END SUB.

Οι κυριότερες διαφορές μεταξύ υποπρογραμμάτων και υπορουτινών είναι οι εξής:

1. Τα υποπρογράμματα μπορούν να χρησιμοποιούν **τοπικές** (local) ή **καθολικές** (global) μεταβλητές. Τοπικές μεταβλητές είναι αυτές που διατηρούν την τιμή τους μέσα στα όρια του υποπρογράμματος και δεν έχουν καμμία σχέση με συνώνυμες μεταβλητές του κύριου προγράμματος ή άλλων υποπρογραμμάτων. Αντίθετα οι καθολικές μεταβλητές είναι κοινές για όλα τα υποπρογράμματα της ίδιας ενότητας. Σε ένα υποπρόγραμμα μία ή περισσότερες μεταβλητές αν δεν δηλωθούν ρητά σαν καθολικές, τότε είναι τοπικές (by default). Στις υπορουτίνες όλες οι μεταβλητές είναι καθολικές και δεν υπάρχει η δυνατότητα τοπικών μεταβλητών. Το γεγονός αυτό ενέχει σοβαρούς κινδύνους για τον προγραμματιστή και δεν είναι σπάνιες οι φορές που σε μια υπορουτίνα επηρεάζεται κατά λάθος μια μεταβλητή βασικής σημασίας άλλης υπορουτίνας.

2. Ενα υποπρόγραμμα είναι ένα ανεξάρτητο τμήμα προγράμματος, σε αντίθεση με την υπορουτίνα που είναι συνδεδεμένη με το πρόγραμμα από το οποίο χρησιμοποιείται. Το υποπρόγραμμα μπορεί να ορισθεί σε μια ενότητα και να χρησιμοποιηθεί από άλλη. Αυτό επιφέρει σημαντική οικονομία στον απαιτούμενο κώδικα για την υλοποίηση εφαρμογών.

4.1.2. Ορισμός υποπρογραμμάτων

Ενα υποπρόγραμμα ορίζεται με την εντολή SUB. Η σύνταξη της εντολής είναι:

```
SUB όνομα-υποπρογράμματος [(λιστα
                παραμετρων)] STATIC
```

Οπου:

- το *όνομα_υποπρογράμματος* έχει μήκος έως 40 χαρακτήρες και δεν πρέπει να εμφανίζεται σε άλλη εντολή SUB στο ίδιο πρόγραμμα.
- η *λίστα_παραμέτρων* περιέχει σταθερές, μεταβλητές και πίνακες που περνούν απο το κύριο πρόγραμμα στο υποπρόγραμμα. Οι παράμετροι χωρίζονται με κόμα.
- η δήλωση STATIC είναι προαιρετική και προσδιορίζει τον τρόπο με τον οποίο χειρίζονται οι τοπικές μεταβλητές. Οταν υπάρχει, οι τοπικές μεταβλητές ορίζονται ως **στατικές** δηλ. διατηρούν την τιμή τους μεταξύ διαδοχικών κλήσεων του υποπρογράμματος. Αν παραληφθεί, όλες οι τοπικές μετα-

βλητές παίρνουν αρχική τιμή το μηδέν ή null κατά την κλήση του υποπρογράμματος. Οι τελευταίες καλούνται **αυτόματες** (automatic).

Το τέλος του υποπρογράμματος προσδιορίζεται με την εντολή:

```
END SUB
```

Μεταξύ των εντολών SUB και END SUB μπορεί να υπάρχει οποιοσδήποτε αριθμός εντολών. Όλες οι εκφράσεις της QuickBASIC επιτρέπονται μέσα σε ένα υποπρόγραμμα εκτός από:

- συναρτήσεις ορισμένες από το χρήστη (user-defined functions) και οι εντολές ορισμού τους (DEF FN...END DEF, FUNCTION...END FUNCTION).
- ένα μπλοκ SUB ... END SUB. Υποπρογράμματα δεν μπορούν να περιέχονται μέσα σε άλλα υποπρογράμματα (nested subprograms). Πάντως ένα υποπρόγραμμα μπορεί να καλεί ένα άλλο υποπρόγραμμα.

Παράδειγμα. Το επόμενο υποπρόγραμμα πραγματοποιεί τη λύση μιας πρωτοβάθμιας εξίσωσης ax+b=0. Δέχεται ως είσοδο τις τιμές των συντελεστών a και b και επιστρέφει ως αποτελέσματα την τιμή του x, εφόσον η εξίσωση έχει λύση, καθώς και την τιμή μιας μεταβλητής flag, η οποία είναι 1, αν υπάρχει λύση, αλλιώς 0.

```
SUB EqSolve1 (a,b,x,flag) STATIC
    x=0 : flag=1
    IF a=0 THEN flag=0 ELSE x=-b/a
END SUB
```

4.1.3. Κλήση υποπρογραμμάτων.

Η εκτέλεση μεταφέρεται από το κύριο πρόγραμμα στο υποπρόγραμμα με την εντολή CALL. Η εντολή CALL μπορεί να καλέσει υποπρογράμματα της QB, υποπρογράμματα σε άλλες γλώσσες ή υπορουτίνες σε γλώσσα ASSEMBLY.

Η σύνταξη της εντολής CALL είναι η εξής:

```
CALL όνομα-υποπρογράμματος [(λίστα
                        ορισμάτων)]
```

Οπου:

- *όνομα_υποπρογράμματος* είναι το όνομα του καλούμενου υποπρογράμματος.
- η *λίστα_ορισμάτων* περιλαμβάνει μεταβλητές,πίνακες και στοιχεία πινάκων που περνούν στο καλούμενο υποπρόγραμμα.

Ο αριθμός και ο τύπος των ορισμάτων στη εντολή CALL πρέπει να είναι ίδια με τον αριθμό και τον τύπο των παραμέτρων στην εντολή SUB. Οι παράμετροι και τα ορίσματα πρέπει να δίνονται με την ίδια σειρά και χωρίζονται με κόμα.

Με την CALL ο έλεγχος ματαβιβάζεται στο υποπρόγραμμα. Εκτελούνται όλες οι εντολές του και όταν συναντηθεί η END SUB, τότε γίνεται επιστροφή στο κύριο πρόγραμμα στην εντολή που ακολουθεί την CALL. Επιστροφή στο κύριο πρόγραμμα μπορεί να γίνει και από οποιοδήποτε σημείο του υποπρογράμματος με την εντολή EXIT SUB.

Για παράδειγμα η κλήση του προηγούμενου υποπρογράμματος EqSolve1 για την εξίσωση 3X+2=0 είναι:

```
CALL EqSolve1 (3,2,x,flag)
```

Ας σημειωθεί ότι η κλήση ενός υποπρογράμματος μπορεί να γίνει και χωρίς τη χρήση της CALL. Προς τούτο αρκεί να τεθεί μόνο το όνομα του υποπρογράμματος, αλλά τώρα τα ορίσματα δεν περικλείονται σε παρενθέσεις. Ετσι στο ίδιο παράδειγμα η κλήση γίνεται:

```
EqSolve1 3,2,x,flag
```

4.1.4. Μεταβίβαση τιμών με την εντολή CALL.

Σταθερές, μεταβλητές, πίνακες, εγγραφές και πεδία εγγραφών μπορούν να μεταβιβαστούν από το κύριο πρόγραμμα στο υποπρόγραμμα μέσω της λίστας ορισμάτων της εντολής CALL.

Απλές μεταβλητές. Το παράδειγμα που ακολουθεί δείχνει πως μια εντολή CALL καλεί ενα υποπρόγραμμα και περνά σ'αυτά τις μεταβλητές a και b.

```
' Κύριο πρόγραμμα
a = 5 : b = 0
PRINT "Κύριο πρόγραμμα, πριν την κλήση:";
PRINT "a ="; a, "b ="; b
CALL Square (a,b)
PRINT "Κύριο πρόγραμμα, μετά την κλήση:";
PRINT "a ="; a, "b ="; b
END

SUB Square (x,y) STATIC
y = x * x
PRINT "Στο υποπρόγραμμα, x =";x;" και y =";y
END SUB
```

Το παραπάνω πρόγραμμα εκτυπώνει τα εξής:

```
Κύριο πρόγραμμα, πριν την κλήση:
a = 5       b = 0
Στο υποπρόγραμμα, x = 5 και y = 25
Κύριο πρόγραμμα, μετά την κλήση:
a = 5       b = 25
```

Το υποπρόγραμμα αλλάζει την τιμή της μεταβλητής b δίνοντας μια νέα τιμή στην αντίστοιχη παράμετρο y της εντολής SUB. Οταν αρχίζει η εκτέλεση του υποπρογράμματος Square, η αρχική τιμή του y ειναι 0. Το υποπρόγραμμα υπολογίζει μια νέα τιμή για το y (σ'αυτή την περίπτωση το 25) και την αποθηκεύει στην παράμετρο που πέρασε στο πρόγραμμα.

Οι τιμές των μεταβλητών που μεταβιβάζονται με τον παραπάνω τρόπο λέμε ότι περνούν **με αναφορά** (by reference). Αυτό σημαίνει ότι στο υποπρόγραμμα περνά η διεύθυνσή τους στη μνήμη. Το υποπρόγραμμα επενεργεί στην τιμή σ'αυτή τη διεύθυνση και όταν ο έ-

λεγχος επιστρέφει στο κύριο πρόγραμμα το περιεχόμενο της διεύθυνσης αυτής μπορεί να έχει αλλάξει. Αν δεν επιθυμούμε το υποπρόγραμμα να μεταβάλλει την τιμή της μεταβλητής, τότε πρέπει να μεταβιβάζεται μόνο η τιμή της μεταβλητής και όχι η διεύθυνσή της. Οταν περνά η πραγματική τιμή της μεταβλητής, το υποπρόγραμμα αντιγράφει την τιμή αυτή σε μια προσωρινή διεύθυνση για δική του χρήση, την οποία και ακυρώνει πριν ο έλεγχος γυρίσει στο κύριο πρόγραμμα. Η αρχική τιμή τότε δεν αλλάζει. Για να περάσουμε μια μεταβλητή **με τιμή** (by value) την περιβάλουμε με παρενθέσεις. Για παράδειγμα, αν αλλάξουμε τη δήλωση CALL Square (a,b) στο προηγούμενο παράδειγμα σε CALL Square(a,(b)) το b περνά "με τιμή" η οποία είναι 0. Στο παράδειγμα το υποπρόγραμμα Square δεν μπορεί να αλλάξει την τιμή της μεταβλητής b. Το υποπρόγραμμα μπορεί ν'αλλάξει την τιμή του y αλλά αυτή η τιμή είναι τοπική στο υποπρόγραμμα και δεν μπορεί να γυρίσει πίσω στο κύριο πρόγραμμα μέσω της μεταβλητής b.

Το αναθεωρημένο παράδειγμα τυπώνει:

```
Κύριο πρόγραμμα, πριν την κλήση:
a = 5   b = 0
Στο υποπρόγραμμα x = 5 και y = 25
Κύριο πρόγραμμα, μετά την κλήση:
a = 5   b = 5
```

Ας σημειωθεί οτι μπορούμε ακόμα να περάσουμε εκφράσεις ως παραμέτρους όπως παρακάτω:

```
CALL Prog2(x,(y+1))
```

Οι εκφράσεις περνούν "με τιμή".

Πίνακες. Με τον ίδιο τρόπο μπορούν να μεταβιβαστούν και πίνακες σε ένα υποπρόγραμμα.

Παράδειγμα. Το ακόλουθο υποπρόγραμμα υπολογίζει το άθροισμα των στοιχείων του μονοδιάστατου πίνακα X. Το πλήθος των στοιχείων του πίνακα μεταβιβάζεται στο υποπρόγραμμα με τη μεταβλητή N, ενώ το αποτέλεσμα επιστρέφεται στη μεταβλητή S.

```
SUB VectorSum (x(1),n,s) STATIC
    s=0
    FOR i=1 TO n
        s=s+x(i)
    NEXT i
END SUB
```

Η εντολή CALL με όρισμα έναν πίνακα έχει την ακόλουθη μορφή:

```
CALL όνομα-υποπρογράμματος (όνομα πίνακα())
```

π.χ. CALL VectorSum (x(),n,s)

Αν ένα υποπρόγραμμα δεν χρειάζεται έναν ολόκληρο πίνακα,μπορεί να μεταβιβαστούν μόνο όσα στοιχεία του πίνακα απαιτούνται. Για να μεταβιβαστεί ένα ατομικό στοιχείο ενός πίνακα αρκεί να προστεθούν και οι τιμές των δεικτών του στοιχείου μέσα στις παραμέτρους της CALL.

π.χ. CALL Prog1(Array(3,9))

4.2 ΣΥΝΑΡΤΗΣΕΙΣ

Μία συνάρτηση είναι μια διαδικασία που επιστρέφει μια τιμή με το όνομά της. Η συνάρτηση διαφέρει από το υποπρόγραμμα, γιατί ενώ μπορεί να δεχτεί πολλές παραμέτρους, επιστρέφει πάντα μια μόνο τιμή. Οι συναρτήσεις μπορούν να εκτελούν οποιαδήποτε λειτουργία που επιθυμεί ο χρήστης, γι' αυτό και καλούνται συναρτήσεις ορισμένες από τον χρήστη (user defined functions) σε αντιδιαστολή με τις ενσωματωμένες συναρτήσεις της γλώσσας.

4.2.2. Ορισμός συναρτήσεων

Μία συνάρτηση ορίζεται με την εντολή FUNCTION. Η σύνταξη της εντολής είναι:

```
FUNCTION όνομα_συνάρτησης (λίστα_παραμέτρων) STATIC
```

Το όνομα_συνάρτησης έχει μήκος έως 40 χαρακτήρες και δεν μπορεί να έχει αποδοθεί σε άλλη συνάρτηση ή/και υποπρόγραμμα. Για τη λίστα παραμέτρων και τη δήλωση STATIC ισχύουν όσα αναφέρθηκαν για τα υποπρογράμματα. Το τέλος τη συνάρτησης προσδιορίζεται από την εντολή END SUB. Μεταξύ της FUNCTION και END SUB μπορούν να υπάρχουν οποιεσδήποτε εντολές εκτός απο αυτές που ορίζουν συναρτήσεις ή υποπρογράμματα. Η έξοδος από μία συνάρτηση επιτυγχάνεται με την εντολή EXIT SUB.

Παράδειγμα. Η επόμενη συνάρτηση ορίζει τη μαθηματική συνάρτηση y=ημ(x)/x.

```
FUNCTION y (x) STATIC
    IF x<>0 THEN
        y=SIN(x)/x
    ELSE
        y=0
    END IF
END FUNCTION
```

4.2.3 Κλήση συναρτήσεων και πέρασμα μεταβλητών

Η κλήση μιας FUNCTION γίνεται με τον ίδιο τρόπο που καλείται οποιαδήποτε ενσωματωμένη συνάρτηση της BASIC, δηλ. με απλή αναφορά του ονόματός της σε μία έκφραση. Ετσι η κλήση της συνάρτησης του προηγούμενου παραδείγματος μπορεί να γίνει ως εξής:

```
INPUT "X=",x
PRINT y(x)
```

Να σημειωθεί ότι επειδή μια συνάρτηση επιστρέφει μια τιμή με το όνομά της, αυτό πρέπει να συμφωνεί με τον τύπο του αποτελέσματος. Αν για παράδειγμα μια συνάρτηση επιστρέφει ως αποτέλεσμα μια στοιχειοσειρά, τότε το όνομα της συνάρτησης πρέπει να είναι αλφαριθμητική μεταβλητή.

Η μεταβίβαση τιμών σε μία συνάρτηση γίνεται μέσω της λίστας ορισμάτων της καλούμενης συνάρτησης. Η σειρά των ορισμάτων και ο τύπος τους πρέπει να συμφωνεί με τις παραμέτρους στον ορισμό της συ-

νάρτησης. Σε μία συνάρτηση μπορούν να περνούν σταθερές, εκφράσεις, μεταβλητές, πίνακες και εγγραφές με το ίδιο τρόπο που γίνεται και στα υποπρογράμματα.

4.3 ΠΕΡΑΣΜΑ ΜΕΤΑΒΛΗΤΩΝ ΜΕ ΤΗ ΒΟΗΘΕΙΑ ΤΗΣ SHARED.

Πέρα από τη δυνατότητα να μεταβιβάζονται μεταβλητές μέσα από την εντολή CALL, η QuickBASIC παρέχει εναλλακτικούς τρόπους με τους οποίους περνούν μεταβλητές κατά την κλήση διαδικασιών:

1. Η ιδιότητα SHARED σε εντολές COMMON και DIM στο κύριο πρόγραμμα επιτρέπει τη χρήση των ορισμένων εκεί μεταβλητών από όλα τα υποπρογράμματα, δηλ. οι μεταβλητές αυτές ορίζονται ως καθολικές. Αυτή η μέθοδος χρησιμοποιείται όταν όλες οι μεταβλητές χρησιμοποιούνται από όλα τα υποπρογράμματα. Για να χρησιμοποιηθεί η ιδιότητα SHARED, πρέπει να τοποθετηθεί η λέξη SHARED αμέσως μετά τις εντολές COMMON ή DIM όπως στα παρακάτω παραδείγματα:

```
COMMON SHARED a,b,c
DIM SHARED Array(10,10)
```

Αν απαιτείται για μία ή περισσότερες μεταβλητές να δηλωθεί και ο τύπος τους, τότε οι προηγούμενες εντολές επεκτείνονται και με τη δήλωση AS πχ.

```
DIM SHARED i AS INTEGER
```

Δεν επιτρέπεται η διπλή δήλωση για μία μεταβλητή, δηλ αν η i έχει δηλωθεί ως ακέραια μεταβλητή με την DEFINT, τότε θα σημειωθεί λάθος στην εντολή COMMON [SHARED] i που τυχόν ακολουθεί.

2. Η εντολή SHARED επιτρέπει σ'ενα υποπρόγραμμα να χρησιμοποιήσει μεταβλητές οι οποίες έχουν δηλωθεί στο κύριο πρόγραμμα χωρίς την ιδιότητα SHARED. Αυτή η μέθοδος είναι πιο χρήσιμη όταν διαφορετικά υποπρογράμματα χρησιμοποιούν διαφορετικούς συνδυασμούς μεταβλητών του κύριου προγράμματος. Η εντολή SHARED έχει την ακόλουθη μορφή:

```
SHARED λίστα_μεταβλητών
```

Μια απλή χρήση της SHARED φαίνεται στο παρακάτω παράδειγμα:

```
a = 1 : b = 2
CALL Prog2

SUB Prog2 STATIC
    SHARED a,b
    c = a + b
    PRINT c          ' εκτυπώνεται 3
END SUB
```

4.4 ΑΝΑΔΡΟΜΗ

Ο όρος αναδρομή αναφέρεται στη δυνατότητα μιας διαδικασίας να χρησιμοποιεί την ίδια τη διαδικασία στον ορισμό της δηλ. να μπορεί να καλεί τον εαυτό της. Οι διαδικασίες της QB μπορούν να είναι αναδρομικές. Κλασικό παράδειγμα αναδρομικής διαδικασίας είναι ο υπολογισμός του n! (n παραγοντικό). Το παραγοντικό ενός μη αρνητικού αριθμού n ορίζεται ως εξής:

n! = n . (n-1)! και

0! = 1

Π.χ. διά n=5 => 5!=5.4.3.2.1=120

Η αναδρομική συνάρτηση Fact υπολογισμού του n! υλοποιείται άμεσα στις επόμενες γραμμές.

```
DECLARE FUNCTION Fact# (n%)
INPUT "Δώσε αριθμό από 1 έως 20 :", n%
PRINT n%,Fact#(n%)
END
'
FUNCTION Fact#(n%) STATIC
    IF n%>0 THEN
        Fact# = n% * Fact#(n%-1)
    ELSE
        Fact# = 1
    END IF
END FUNCTION
```

Στην αναδρομική κλήση διαδικασιών πρέπει να λαμβάνονται ορισμένες προφυλάξεις. Πριν από κάθε κλήση μιας διαδικασίας φυλάσσονται σε μια στοίβα (stack) οι τιμές όλων των τοπικών μεταβλητών, ώστε μετά την επιστροφή να ανακληθούν και η διαδικασία να συνεχίσει από εκεί που σταμάτησε με τη σωστή "εικόνα". Το γεγονός αυτό μπορεί να εξαντλήσει τα περιθώρια της στοίβας, ιδιαίτερα αν γίνεται μεγάλος αριθμός κλήσεων ή χρησιμοποιούνται πολλές μεταβλητές. Στην περίπτωση αυτή προκαλείται μήνυμα Out of stack space. Το πρόβλημα αυτό μπορεί να διορθωθεί με την αύξηση του μεγέθους της στοίβας με τη χρήση της CLEAR,,μσ. Οπου μσ είναι το μέγεθος στοίβας, το οποίο προεκτιμάται ανάλογα με την πολυπλοκότητα της διαδικασίας.

4.5 ΕΛΕΓΧΟΣ ΜΕΤΑΒΛΗΤΩΝ ΜΕ ΤΗΝ ΕΝΤΟΛΗ DECLARE

Κατά την αποθήκευση επί του δίσκου προγραμμάτων που περιέχουν διαδικασίες, αυτόματα από το συντάκτη της QB παρεμβάλονται στην αρχή ορισμένες εντολές DECLARE μία για κάθε χρησιμοποιούμενη διαδικασία. Κάθε τέτοια εντολή αποτελείται από τη λέξη DECLARE, ακολουθούμενη από τη SUB ή FUNCTION, το όνομα της διαδικασίας και ένα ζεύγος παρενθέσεων. Αν η διαδικασία έχει παραμέτρους, τότε η λίστα παραμέτρων παρεμβάλλεται μεταξύ των παρενθέσεων, αλλιώς παραμένουν κενές. Η λίστα παραμέτρων της DECLARE είναι ακριβώς ίδια με αυτή που χρησιμοποιείται στη γραμμή ορισμού της διαδικασίας SUB ή FUNCTION.

Visual Basic

1. ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΤΗΣ VISUAL BASIC 201
1.1. Το κεντρικό παράθυρο 202
1.2. Η φόρμα εργασίας 202
1.3. Η εργαλειοθήκη 204
1.4. Προσθήκη εργαλείων 205
1.5. Ο εξερευνητής εργασίας 206
1.6.Το παράθυρο ιδιοτήτων 207
1.7. Το παράθυρο μορφής φόρμας 208
2. ΑΡΧΕΣ ΣΥΓΧΡΟΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 208
2.1. Εισαγωγή στον προγραμματισμό μιας εφαρμογής 208
2.2. Αντικειμενοστραφής προγραμματισμός 209
2.3. Προγραμματισμός οδηγούμενος από τα γεγονότα 210
3. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 212
3.1. Δομή μίας Visual Basic εφαρμογής 212
3.2.Τα τμήματα κώδικα..... 213
3.3. Σταθερές 214
3.4. Μεταβλητές 214
3.5. Ορισμός μεταβλητών.....215
3.6. Δομές Απόφασης ή Επιλογής 216
3.7. Δομές Ανακύκλωσης 216
4. ΤΕΧΝΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 217
4.1. Μενού επιλογών 217
4.2. Πλαίσια διαλόγου 218
4.3. Εκτυπώσεις 218
4.4. Επικοινωνία με το Clipboard 218
5. ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ 219
5.1. Εργαλείο δεδομένων 219
5.2. Ιδιότητες βάσεων δεδομένων 219
5.3. Τα εργαλεία εμφάνισης δεδομένων 220
6. ΓΡΑΦΙΚΑ 220
6.1. Το σύστημα συντεταγμένων.....220
6.2. Μονάδες μέτρησης 221
6.3. Αλλαγή κλίμακας συστήματος συντεταγμένων 221
6.4. Χρώματα 221
6.5. Εργαλεία γραφικών.....222
6.6. Μέθοδοι γραφικών.....222
6.7. Ιδιότητες γραφικών 223
7. ΠΟΛΥΜΕΣΑ (MULTIMEDIA) 225
7.1. Multimedia εργαλεία της Visual Basic 225
7.2. Ηχος 226
7.3. Προσομοίωση κίνησης 226
7.4. Video.....226

ΕΙΣΑΓΩΓΗ

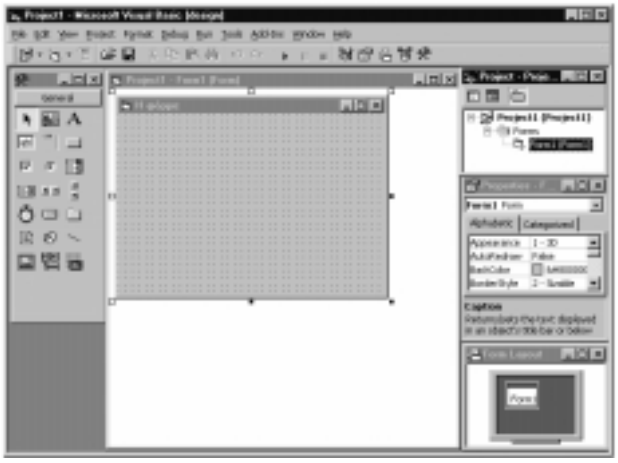
Η Visual Basic είναι το πρώτο ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών που δημιούργησε τις προϋποθέσεις για εύκολο και δημιουργικό προγραμματισμό στο περιβάλλον των Windows. Οι δυνατότητες και τα εργαλεία προγραμματισμού που προσφέρει, επιτρέπουν σε ένα προγραμματιστή να δημιουργήσει πολύ εύκολα και σε σύντομο χρονικό διάστημα, δυναμικές εφαρμογές ανάλογες με αυτές που χρησιμοποιούνται στο περιβάλλον των Windows. Οι εφαρμογές της Visual Basic χαρακτηρίζονται από τη φιλικότητα που διακρίνει το περιβάλλον εργασίας τους, αφού είναι δυνατό να περιέχουν όλα τα γνωστά στοιχεία που αποτελούν το γραφικό περιβάλλον επικοινωνίας των Windows. Όμως το στοιχείο που σίγουρα θα καταπλήξει έναν προγραμματιστή της Visual Basic είναι το εύχρηστο περιβάλλον προγραμματισμού της.

1. ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΤΗΣ VISUAL BASIC

Ας ξεκινήσουμε όμως τη γνωριμία μας με το περιβάλλον ανάπτυξης εφαρμογών της Visual Basic. Μόλις εκκινήσουμε τη Visual Basic στον υπολογιστή εμφανίζεται στην οθόνη το περιβάλλον ανάπτυξης εφαρμογών της Visual Basic (Σχήμα 1).

Η αρχική εικόνα της Visual Basic θυμίζει περισσότερο μία εφαρμογή των Windows παρά ένα προγραμματιστικό εργαλείο.

Αρχικά το περιβάλλον της Visual Basic φαίνεται να είναι αρκετά πολύπλοκο για ένα καινούριο χρήστη, όμως πολύ σύντομα θα εξοικειωθεί με αυτό και θα ανακαλύψει τα πλεονεκτήματα και την απλότητα του. Το περιβάλλον ανάπτυξης εφαρμογών της Visual Basic αποτελείται από τα παρακάτω στοιχεία :



Σχ.1. Το περιβάλλον ανάπτυξης εφαρμογών της Visual Basic.

- Το κεντρικό παράθυρο της Visual Basic.
- Τη φόρμα εργασίας (Form)
- Την εργαλειοθήκη (Toolbox).
- Τον εξερευνητή εργασίας (Project Explorer).
- Το παράθυρο ιδιοτήτων (Properties Window).
- Το παράθυρο μορφής φόρμας (Form Layout Window).

1.1. Το κεντρικό παράθυρο

Στο επάνω τμήμα της οθόνης εμφανίζεται το κεντρικό παράθυρο της Visual Basic (Σχήμα 2), το οποίο αποτελείται από τη γραμμή τίτλου (Title bar), το κεντρικό μενού επιλογών (Menu bar) και τη γραμμή εργαλείων (Toolbar).

Η αρχική θέση του κεντρικού παραθύρου, είναι στο πάνω τμήμα του περιβάλλοντος προγραμματι-

Σχ.2. Το κεντρικό παράθυρο της Visual Basic.

σμού της Visual Basic. Ωστόσο αν επιθυμούμε μπορούμε πολύ εύκολα να το μετακινήσουμε ή να αλλάξουμε τις διαστάσεις του όπως ένα οποιοδήποτε άλλο παράθυρο των Windows. Προσοχή όμως, μόλις κλείσουμε το κεντρικό παράθυρο τερματίζει αυτόματα και η Visual Basic.

Η **γραμμή τίτλου**, αποτελείται από τα βασικά στοιχεία ενός κλασικού παραθύρου των Windows. Επάνω της εμφανίζονται τα στοιχεία ταυτότητας του προγράμματος, το Πλαίσιο Μενού Ελέγχου (Control Menu Box) και τα πλήκτρα μεγιστοποίησης, ελαχιστοποίησης και κλεισίματος του παραθύρου (Minimize, Maximize buttons & Close buttons).

Το **κεντρικό μενού**, περιέχει όλες τις εντολές που χρειάζεται ο προγραμματιστής για τη δημιουργία μίας εφαρμογής. Για να ανοίξει ένα μενού επιλογών, πατάμε το αριστερό πλήκτρο του ποντικιού, όταν ο δρομέας του βρίσκεται επάνω στο όνομα του ή χρησιμοποιούμε από το πληκτρολόγιο κλειδιά πρόσβασης (Access Keys), πατάμε δηλαδή συνδυαστικά το αριστερό πλήκτρο Alt και το υπογραμμισμένο γράμμα του ονόματος της επιθυμητής επιλογής. Για να επιλέξουμε μια εντολή από ένα ήδη ανοιγμένο μενού επιλογών χρησιμοποιούμε επίσης το ποντίκι ή από το πληκτρολόγιο πατάμε το υπογραμμισμένο γράμμα του ονόματος της επιλογής. Στις εργασίες των μενού επιλογών που εκτελούνται συχνά, μπορούμε να έχουμε άμεση πρόσβαση χωρίς να ανοίξουμε ένα μενού επιλογών με τη χρήση πλήκτρων συντομίας (shortcut keys), τα οποία εμφανίζονται δίπλα από το όνομα της επιλογής.

Στη **γραμμή εργαλείων,** υπάρχουν οι πιο συχνά εκτελέσιμες επιλογές του κεντρικού μενού επιλογών σε μορφή εικονιδίων (Σχήμα 3). Τα εικονίδια είναι χωρισμένα σε ομάδες (groups) ανάλογα με τις εργασίες που εκτελούν.

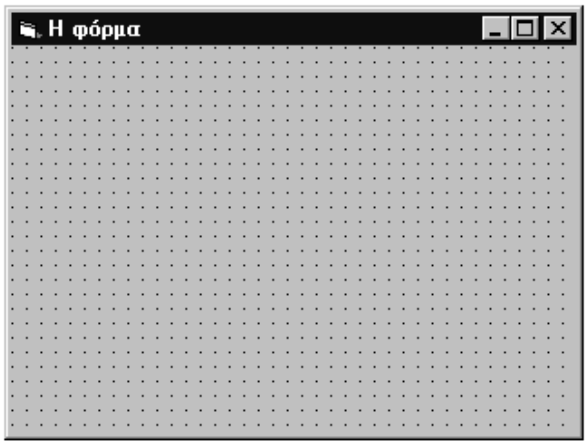
Σχ.3. Η γραμμή εργαλείων.

Στο δεξί τμήμα της γραμμής εργαλείων (Toolbar) υπάρχουν δύο ενδεικτικά πλαίσια, το πλαίσιο θέσης και το πλαίσιο μεγέθους. Στο πλαίσιο θέσης εμφανίζονται κάθε φορά οι συντεταγμένες της θέσης του επιλεγμένου γραφικού αντικειμένου, ενώ στο πλαίσιο μεγέθους εμφανίζονται οι διαστάσεις του. Όταν μεταφέρουμε το ποντίκι επάνω σε κάποιο εικονίδιο της γραμμής εργαλείων, να εμφανίζονται επεξηγηματικά πλαίσια (Tooltips) που περιγράφουν τη λειτουργία του.

1.2. Η φόρμα εργασίας

Στο κέντρο της αρχικής οθόνης της Visual Basic εμφανίζεται ένα κενό παράθυρο το οποίο ονομάζεται φόρμα εργασίας (form). Η φόρμα εργασίας είναι το βασικότερο γραφικό αντικείμενο της Visual Basic. Κατά την εκτέλεση μιας εφαρμογής, κάθε φόρμα της Visual Basic αποτελεί ένα ανεξάρτητο παράθυρο επικοινωνίας της εφαρμογής με το χρήστη του συστήματος (Σχήμα 4).

Η φόρμα εργασίας αποτελεί το σημείο εκκίνησης του σχεδιασμού κάθε Visual Basic εφαρμογής. Ο σχεδιασμός της διεπαφής χρήστη (user interface) ξεκινά πάντοτε από τη δημιουργία μίας φόρμας εργασίας. Στη συνέχεια επάνω στη φόρμα θα "χτιστούν" σιγά-σιγά τα υπόλοιπα τμήματα της εφαρμογής. Αφού σχεδιάσουμε τη φόρμα θα τοποθετήσουμε επάνω σε αυτή τα υπόλοιπα γραφικά αντικείμενα της Visual Basic, μέσω των οποίων θα γίνεται η επικοινωνία του χρήστη με την εφαρμογή κατά την εκτέλεση της.



***Σχ.4** Η φόρμα εργασίας.*

Εξετάζοντας προσεχτικά μια φόρμα της Visual Basic, θα παρατηρήσουμε ότι περιέχει όλα τα γνωστά στοιχεία ενός παραθύρου, παρόμοιο με αυτά που χρησιμοποιούνται σε όλες τις κλασικές εφαρμογές των Windows. Το μέγεθος και τη θέση της φόρμας επάνω στην οθόνη, είναι δυνατό να τα μεταβάλλουμε με μεγάλη ευκολία κατά το σχεδιασμό της εφαρμογής. Αν μεταφέρουμε το ποντίκι επάνω σε κάποιο περίγραμμα της φόρμας, ο δείκτης του θα μετατραπεί σε διπλό βέλος. Κρατώντας πατημένο συνεχώς το αριστερό πλήκτρο του ποντικιού μπορούμε να το μετακινήσουμε και ταυτόχρονα παρατηρούμε ότι μεταφέρεται προς την ίδια κατεύθυνση και το περίγραμμα της φόρμας. Μόλις απελευθερώσουμε το πλήκτρο του ποντικιού η φόρμα θα αποκτήσει τις νέες διαστάσεις που της αποδώσαμε. Για να μεταφέρουμε τη φόρμα σε μια καινούρια θέση, πηγαίνουμε το ποντίκι επάνω στη γραμμή τίτλου της φόρμας και κρατώντας πατημένο συνεχώς το αριστερό του πλήκτρο το μετακινούμε και ταυτόχρονα μεταφέρεται η φόρμα εργασίας. Μόλις απελευθερώσουμε το πλήκτρο του ποντικιού η φόρμα καταλαμβάνει τη καινούρια θέση. Οι διαστάσεις και η θέση που θα αποδώσουμε στη φόρμα κατά το σχεδιασμό της, θα είναι αυτές με τις οποίες θα εμφανίζεται στην οθόνη του υπολογιστή και κατά την εκτέλεση της εφαρμογής.

Κατά τη διάρκεια του σχεδιασμού μιας φόρμας εμφανίζεται επάνω στην επιφάνειά της, ένα πλέγμα (Grid) το οποίο αποτελείται από μικρές μαύρες τελείες που ισαπέχουν μεταξύ τους. Οι τελείες αυτές δεν εμφανίζονται κατά την εκτέλεση της εφαρμογής, αλλά αποτελούν βοήθημα του προγραμματιστή για την τοποθέτηση και το σχεδιασμό των υπόλοιπων γραφικών αντικειμένων της Visual Basic επάνω στην επιφάνεια της φόρμας. Αν θέλουμε, κατά το σχεδιασμό της εφαρμογής μπορούμε να εξαφανίσουμε το πλέγμα από την επιφάνεια της φόρμας ή να αλλάξουμε την απόσταση των τελειών που το αποτελούν, θέτοντας την τιμή Show Grid=No και μετατρέποντας τις αριθμητικές τιμές στα πλαίσια Width και Height στην κατηγορία General της επιλογής Options στο μενού Tools της Visual Basic. Ακόμη στην ίδια κατηγορία, αν ενεργοποιήσουμε την επιλογή Align Controls To Grid οι εξωτερικές άκρες κάθε εργαλείου θα στοιχίζονται στις γραμμές πλέγματος.

Μία εφαρμογή της Visual Basic μπορεί να περιέχει μία ή περισσότερες φόρμες εργασίας. Ο αριθμός των φορμών που θα σχεδιάσουμε, συνήθως είναι ανάλογος με το μέγεθος της εφαρμογής. Κάθε φόρμα μέσα σε μία εφαρμογή αποτελεί ένα ξεχωριστό παράθυρο και μπορεί να έχει διαφορετικό ρόλο. Μπορεί για παράδειγμα μια φόρμα να χρησιμοποιείται για εισαγωγή δεδομένων ή την εμφάνιση ενημερωτικών στοιχείων κ.ο.κ. Σε μια εφαρμογή είναι προτιμότερο, αν είναι δυνατό να αποφεύγουμε τη χρήση πολλών φορμών εργασίας, γιατί συνήθως ο μεγάλος αριθμός τους επηρεάζει αρνητικά τη λειτουργικότητα της εφαρμογής.

Το κυριότερο χαρακτηριστικό μίας φόρμας της Visual Basic, είναι ότι περιέχει τα δικά της εσωτερικά στοιχεία και γνωρίζει κάθε φόρα πως πρέπει να συμπεριφερθεί στις απαιτήσεις του χρήστη. Για να γίνει κατανοητό αυτό μπορούμε να υποθέσουμε ότι η φόρμα που εμφανίζεται στην οθόνη εκκίνησης της Visual Basic αποτελεί από μόνη της την πρώτη μας εφαρμογή και μάλιστα χωρίς να χρειαστεί να γράψουμε καθόλου εντολές κώδικα. Αν θέλουμε να εκκινήσουμε την εκτέλεση της, επιλέγουμε από το μενού επιλογών Run της Visual Basic την επιλογή Start ή το πλήκτρο συντομίας F5 ή εναλλακτικά επιλέγουμε με το ποντίκι του υπολογιστή το δέκατο εικονίδιο της γραμμής εργαλείων.

Αμέσως η Visual Basic περνάει σε κατάσταση εκτέλεσης (run mode), όπως δηλώνει το λεκτικό [run] επάνω στη γραμμή τίτλου. Η φόρμα μας εμφανίζεται στην οθόνη στο σημείο που την είχαμε τοποθετήσει κατά το σχεδιασμό της και είναι δυνατό να εκτελέσουμε με αυτή όλες τις γνωστές εργασίες των παραθύρων των Windows. Μπορούμε δηλαδή, να μετακινήσουμε τη φόρμα και να μεταβάλλουμε το μέγεθος και τη θέση της επάνω στην οθόνη χρησιμοποιώντας το ποντίκι ή τα πλήκτρα μεγιστοποίησης και ελαχιστοποίησης και τέλος μπορούμε να κλείσουμε το παράθυρο μέσω των επιλογών του Πλαισίου Μενού Ελέγχου ή με το πλήκτρο κλεισίματος. Αν κλείσουμε τη φόρμα, επειδή αποτελεί το μοναδικό ανοικτό παράθυρο της εφαρμογής τερματίζει αυτόματα και η εκτέλεση της. Μπορούμε επίσης να σταματήσουμε την εκτέλεση της εφαρμογής και με την επιλογή End από το μενού επιλογών Run ή επιλέγοντας το δωδέκατο εικονίδιο της γραμμής εργαλείων της Visual Basic. Μετά το κλείσιμο της εφαρμογής επιστρέφουμε πάλι στην κατάσταση σχεδιασμού της Visual Basic.

Με το παραπάνω παράδειγμα έγινε αντιληπτό ότι η δημιουργία ενός παραθύρου με τη Visual Basic γίνεται πολύ απλά και γρήγορα, ενώ είναι σημαντικό ότι μια φόρμα κατά την εκτέλεση της εφαρμογής αποτελεί ένα πραγματικό αντικείμενο που αναγνωρίζει αυτόματα τις ενέργειες του χρήστη και αντιδρά ανάλογα, χωρίς να χρησιμοποιήσουμε ούτε μία εντολή κώδικα.

Ακόμη ο προγραμματιστής είναι δυνατό μέσω εντολών κώδικα να μεταβάλλει την εμφάνιση και να αναλάβει το χειρισμό μιας φόρμας. Μέσω εντολών κώδικα Basic έχουμε τη δυνατότητα μετατρέποντας την τιμή μιας ιδιότητας της φόρμας να αλλάξουμε τα χαρακτηριστικά της (π.χ. αλλαγή της ιδιότητας BackColor σημαίνει μεταβολή στο χρώμα φόντου), αλλά και με τη χρήση μεθόδων που υποστηρίζει η Visual να εκτελέσουμε μια ενέργεια (π.χ. με τη μέθοδο Move μπορούμε να μεταβάλουμε τη θέση της).

1.3. Η εργαλειοθήκη

Η εργαλειοθήκη (Toolbox) εμφανίζεται στο αριστερό άκρο της οθόνης και περιέχει τα βασικά εργαλεία σχεδιασμού της Visual Basic (Σχήμα 5). Τα εργαλεία σχεδιασμού βρίσκονται επάνω στην εργαλειοθήκη σε τρεις στήλες με μορφή εικονιδίου (icons) και ονομάζονται controls.

***Σχ.5** Η εργαλειοθήκη.*

Τα εργαλεία που περιέχει η εργαλειοθήκη αποτελούν μαζί με τις φόρμες εργασίας τα γραφικά αντικείμενα της Visual Basic. Με τα γραφικά αντικείμενα σχεδιάζουμε το τρόπο επικοινωνίας χρήστη-εφαρμογής (user interface). Η δημιουργία της διεπαφής χρήστη ολοκληρώνεται με την τοποθέτηση των κατάλληλων εργαλείων επάνω στη φόρμα εργασίας. Κατά την εκτέλεση της εφαρμογής μπορεί κάθε στιγμή ο χρήστης να επικοινωνεί μέσω των γραφικών αντικειμένων με την εφαρμογή και να κατευθύνει τη ροή εκτέλεσής της.

Ένας προγραμματιστής πριν ξεκινήσει το σχεδιασμό μίας εφαρμογής στη Visual Basic, πρέπει να γνωρίζει ακριβώς πως μπορεί να χρησιμοποιήσει κάθε εργαλείο που περιέχει η εργαλειοθήκη και ποια η λειτουργία καθενός από αυτά. Η σωστή επιλογή των κατάλληλων εργαλείων κατά τη δημιουργία μιας εφαρμογής εξασφαλίζει την επιτυχία της εκτέλεσης της.

Όπως είδαμε ήδη με τις φόρμες εργασίας, έτσι και κάθε εργαλείο της Visual Basic περιέχει τα δικά του χαρακτηριστικά και κυρίως έχει τη δική του εσωτερική συμπεριφορά. Ο χειρισμός των περισσοτέρων εργαλείων είναι ήδη γνωστός σε ένα πεπειραμένο χρήστη των Windows αφού θα τα έχει ήδη χρησιμοποιήσει σε κάποια άλλη εφαρμογή.

Η Visual Basic υποστηρίζει τρεις τύπους εργαλείων τα **εσωτερικά** (intrinsic controls) τα **ActiveX** και τα ***ένθετα*** (insertable objects). Τα εσωτερικά εργαλεία είναι τα βασικά εργαλεία, εμπεριέχονται στο εκτελέσιμο αρχείο της Visual Basic και εμφανίζονται πάντοτε στην εργαλειοθήκη. Τέτοια εργαλεία είναι τα πλήκτρα εντολής, οι εικόνες και οι ετικέτες. Στη συνέχεια θα κάνουμε μία περιγραφή των εσωτερικών εργαλείων που περιέχει η εργαλειοθήκη της Visual Basic.

- ✓ **Δείκτης (Pointer):** Ο δείκτης είναι το πρώτο εικονίδιο που εμφανίζεται στο πάνω αριστερό άκρο της εργαλειοθήκης και είναι το μοναδικό εργαλείο με το οποίο δεν μπορούμε να σχεδιάσουμε κάποιο γραφικό αντικείμενο. Ο δείκτης απεικονίζει το δρομέα (cursor) του ποντικιού και χρησιμοποιείται για την επιλογή και το χειρισμό των υπόλοιπων εργαλείων της Visual Basic. Κάθε φορά που ολοκληρώνουμε το σχεδιασμό ενός αντικειμένου με κάποιο από τα υπόλοιπα εργαλεία της εργαλειοθήκης αυτόματα επανεπιλέγεται ο δείκτης.
- ✓ **Πλαίσιο εικόνας (Picture box):** Είναι το πρώτο εργαλείο της εργαλειοθήκης και χρησιμοποιείται για την εμφάνιση εικόνων και γραφικών. Ένα πλαίσιο εικόνας είναι δυνατό να περιέχει αρχεία εικονιδίων (icons) με επέκταση ονόματος .ICO, αρχεία χαρτογράφησης (bitmaps) με επέκταση ονόματος.BMP ή τέλος μεταρχεία (metafiles) των Windows με επέκταση ονόματος .WMF ή .EMF.
- ✓ **Ετικέτα (Label):** Με τη χρήση των ετικετών μπορούμε να εμφανίσουμε κείμενο επάνω σε κάποιο σημείο της φόρμας. Το κείμενο που περιέχει μια ετικέτα δεν μπορεί να το μεταβάλλει ο χρήστης κατά την εκτέλεση της εφαρμογής. Συνήθως οι ετικέτες σε μία εφαρμογή χρησιμοποιούνται δίπλα από ένα άλλο γραφικό αντικείμενο της Visual Basic και περιγράφουν τα περιεχόμενα ή τη λειτουργία του.
- ✓ **Πλαίσιο κειμένου (Text box):** Χρησιμοποιείται κατά την εκτέλεση της εφαρμογής σαν πλαίσιο εισαγωγής, εμφάνισης και τροποποίησης κειμένου.
- ✓ **Πλαίσιο (Frame):** Το χρησιμοποιούμε αν θέλουμε να ομαδοποιήσουμε λειτουργικά ή διακοσμητικά διάφορα εργαλεία επάνω σε μία φόρμα. Για να ομαδοποιήσουμε ένα αριθμό εργαλείων πρέπει απαραίτητα να σχεδιάσουμε πρώτα το πλαίσιο και στη συνέχεια θα τοποθετήσουμε τα εργαλεία στο εσωτερικό του.

- ✓ **Πλήκτρο εντολής (Command button):** Απεικονίζει γραφικά ένα πλήκτρο, το οποίο όταν επιλεγεί από το χρήστη εκτελεί μία σειρά εντολών κώδικα που έχουμε συμπεριλάβει στην εφαρμογή μας.
- ✓ **Πλαίσιο ελέγχου (Check box):** Εμφανίζει ένα διακόπτη επιλογής στον οποίο ο χρήστης επιλέγει ανάμεσα σε δύο καταστάσεις (Αληθής-Ψευδής ή On-Off). Συνήθως σε μία φόρμα τοποθετούμε μία ομάδα πλαισίων ελέγχου για την εμφάνιση όλων των δυνατών τιμών μίας επιλογής. Κατά την εκτέλεση της εφαρμογής την ίδια χρονική στιγμή μπορούν να είναι επιλεγμένα δύο ή περισσότερα πλαίσια ελέγχου ταυτόχρονα.
- ✓ **Πλήκτρο επιλογής (Option button):** Δημιουργεί όπως και το πλαίσιο ελέγχου ένα διακόπτη επιλογής. Η λειτουργική διαφορά τους, είναι ότι κάθε φορά σε μία ομάδα πλήκτρων επιλογής μπορεί να είναι επιλεγμένο μόνο το ένα από αυτά, ενώ τα υπόλοιπα θα βρίσκονται σε κατάσταση μη επιλογής.
- ✓ **Πλαίσιο σύνθετης λίστας (Combo box):** Είναι ο συνδυασμός ενός πλαισίου εισαγωγής κειμένου και μίας λίστας επιλογών. Σε ένα πλαίσιο σύνθετης λίστας ο χρήστης έχει τη δυνατότητα να επιλέξει μία επιλογή της λίστας ή αν δεν τον ικανοποιεί καμία να πληκτρολογήσει μια νέα εισαγωγή στο πλαίσιο εισαγωγής κειμένου.
- ✓ **Λίστα (List):** Εμφανίζει μία λίστα επιλογών, στην οποία ο χρήστης έχει τη δυνατότητα να επιλέξει μία επιλογή μέσα από τα περιεχόμενα της. Η απλή λίστα δεσμεύει το χρήστη και δεν του επιτρέπει να εισάγει μια καινούργια επιλογή στα περιεχόμενα της.
- ✓ **Οριζόντια και κατακόρυφη γραμμή κύλισης (Horizontal and Vertical scroll bars):** Είναι δύο γραφικά εργαλεία χρήσιμα για γρήγορη μεταφορά μέσα σε ένα συνεχές πεδίο τιμών. Επιτρέπουν στο χρήστη μετακινώντας τα βέλη της γραμμής να μεταφερθεί γραφικά στην επιθυμητή τιμή και να παρουσιάσει τη θέση της ανάμεσα σε ένα πλήθος επιλογών.
- ✓ **Χρονομέτρης (Timer):** Εκτελεί εντολές κώδικα σε συγκεκριμένα χρονικά διαστήματα. Το εργαλείο του χρόνου δεν εμφανίζεται επάνω στη φόρμα κατά την εκτέλεση της εφαρμογής αλλά λειτουργεί πάντοτε στο παρασκήνιο. Για το λόγο αυτό δεν έχει σημασία σε ποιο σημείο της φόρμας θα το τοποθετήσουμε κατά το σχεδιασμό του.
- ✓ **Πλαίσιο λίστας επιλογής δίσκου (Drive list box):** Εμφανίζει σε μία λίστα όλους τους οδηγούς δίσκου του συστήματος και επιτρέπει στον χρήστη την ενεργοποίηση τους. Ο χρήστης δε χρειάζεται να ορίσει στο εργαλείο τους οδηγούς δίσκου του συστήματος, αλλά τους ανιχνεύει αυτόματα η Visual Basic.
- ✓ **Πλαίσιο λίστας επιλογής καταλόγων (Directory list box):** Εμφανίζει σε ιεραρχική δομή καταλόγους προγραμμάτων.
- ✓ **Πλαίσιο λίστας επιλογής αρχείων (File list box):** Χρησιμοποιείται για τη εμφάνιση λίστας αρχείων.
- ✓ **Γεωμετρικό Σχήμα (Shape):** Επιτρέπει κατά το σχεδιασμό της εφαρμογής τη τοποθέτηση ενός γεωμετρικού σχήματος επάνω στη φόρμα. Το εργαλείο αυτό δίνει τη δυνατότητα στο χρήστη να επιλέξει από ένα πλήθος γεωμετρικών σχημάτων όπως τετράγωνο, ορθογώνιο, κύκλο, κ.α.
- ✓ **Γραμμή (Line):** Επιτρέπει κατά το σχεδιασμό της εφαρμογής τη τοποθέτηση μίας ευθείας γραμμής επάνω στη φόρμα. Ο χρήστης μπορεί να επιλέξει την εμφάνιση της γραμμής μέσα από μία ποικιλία σχεδίων.
- ✓ **Εικόνα (Image):** Εμφανίζει εικόνες και γραφικά όπως και το πλαίσιο εικόνας, αξιοποιώντας όμως καλύτερα τη μνήμη και της πηγές του συστήματος. Το μειονέκτημα των εικόνων είναι ότι υποστηρίζουν λιγότερες ιδιότητες, γεγονότα και μεθόδους από τα πλαίσια εικόνας.
- ✓ **Εργαλείο Δεδομένων (Data):** Επιτρέπει τη πρόσβαση μίας εφαρμογής της Visual Basic σε μια βάση δεδομένων. Η πρόσβαση στα δεδομένα της βάσης γίνεται μέσω άλλων εργαλείων της Visual Basic που έχουν δυνατότητα σύνδεσης με τα πεδία της (bound controls).
- ✓ **Εργαλείο πλέγματος (Grid):** Η εμφάνισή του θυμίζει ένα φύλλο εργασίας του Excel. Δημιουργεί πίνακες μέσα στους οποίους μπορούμε να εμφανίσουμε ή να εισάγουμε πληροφορίες και να τις επεξεργαστούμε κατά την εκτέλεση της εφαρμογής.
- ✓ **Εργαλείο OLE (OLE Container):** Χρησιμοποιείται για την πραγματική διασύνδεση ή ενσωμάτωση σε μία Visual Basic εφαρμογή, αντικειμένων που έχουν δημιουργηθεί σε άλλες εφαρμογές μέσω του μηχανισμού OLE των Windows.

1.4. Προσθήκη εργαλείων

Η εργαλειοθήκη της Visual Basic εκτός από τα παραπάνω εσωτερικά εργαλεία όπως αναφέραμε μπορεί να εμπλουτιστεί και με επιπλέον εργαλεία σχεδιασμού τα ActiveX εργαλεία ή με ένθετα αντικείμενα (insertable objects) που έχουν δημιουργηθεί σε κάποια άλλη εφαρμογή.

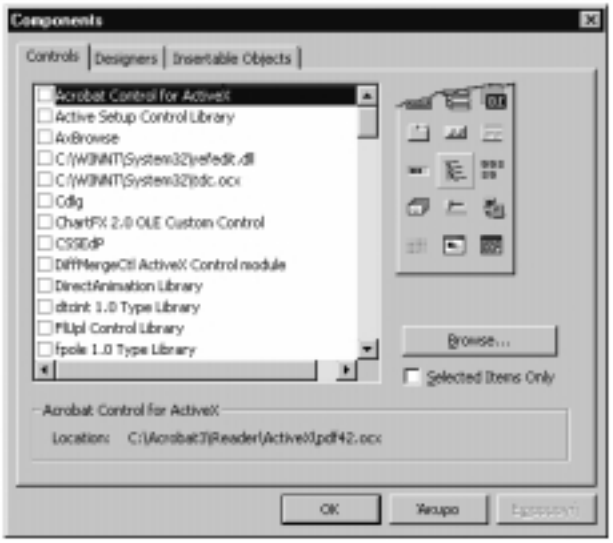
Η Visual Basic υποστηρίζει δύο τύπους ActiveX εργαλείων με επέκταση ονόματος .OCX. ή .VBX. Τα OCX ActiveX εργαλεία εκμεταλλεύονται την σύγχρονη OLE (Object Linking and Embedding) τεχνολογία και χρησιμοποιούνται στην 32-μπιτη αλλά και στη 16-μπιτη έκδοση της Visual Basic. Τα VBX εργαλεία είναι εργαλεία παλαιότερης τεχνολογίας, χρησιμοποιούνται μόνο στη 16-μπιτη έκδοση της Visual Basic και παραμένουν κυρίως για συμβατότητα με προηγούμενες εκδόσεις της.

Όταν ανοίξουμε μια εργασία που έχει δημιουργηθεί σε παλιότερη έκδοση της Visual Basic και περιέχει κάποιο VBX εργαλείο, η Visual Basic εξ΄ ορισμού αναβαθμίζει αυτόματα το συγκεκριμένου εργαλείο με την OCX έκδοση του αν φυσικά υπάρχει.

Τα ένθετα αντικείμενα, όπως αναφέραμε είναι αντικείμενα που έχουν κατασκευαστεί σε άλλη εφαρμογή (π.χ. ένα λογιστικό φύλλο του Excel) και μπορούμε να τα προσθέσουμε στην εργαλειοθήκη της Visual Basic. Τα εργαλεία αυτής της κατηγορίας αφού είναι δυνατό να προστεθούν στην εργαλειοθήκη μπορούμε να τα προσομοιώσουμε με τα ActiveX εργαλεία. Ορισμένα από αυτά τα εργαλεία επιπλέον υποστηρίζουν την τεχνολογία OLE Automation και έχουμε τη δυνατότητα να επέμβουμε προγραμματιστικά στα αντικείμενα άλλης εφαρμογής μέσα από μια Visual Basic εφαρμογή.

Από τη στιγμή που θα προσθέσουμε ένα ActiveX εργαλείο ή ένα ένθετο αντικείμενο στην εργαλειοθήκη της Visual Basic μπορούμε να το χειριστούμε σαν ένα εσωτερικό εργαλείο της γλώσσας.

Για να προσθέσουμε ένα ActiveX εργαλείο ή ένα ένθετο αντικείμενο στην εργαλειοθήκη χρησιμοποιούμε την επιλογή Components του μενού επιλογών Project της Visual Basic. Αμέσως θα εμφανιστεί στη οθόνη μας το πλαίσιο διαλόγου Components (σχήμα 6).



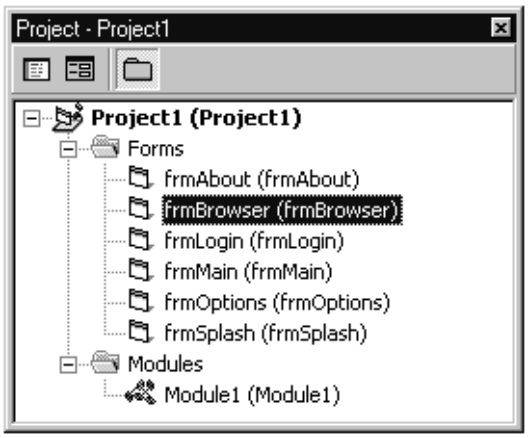
Σχ.6 Το πλαίσιο διαλόγου Components.

Στη λίστα των διαθέσιμων εργαλείων, επιλέγουμε το πλαίσιο ελέγχου που βρίσκεται αριστερά από το όνομά του εργαλείου που θέλουμε να προσθέσουμε και πατάμε το πλήκτρο εντολής OK. Το εργαλείο θα προστεθεί στην εργαλειοθήκη και μπορούμε να το χρησιμοποιήσουμε όπως ακριβώς κάνουμε με κάποιο από τα βασικά εργαλεία της Visual Basic.

Για να αφαιρέσουμε ένα ActiveX εργαλείο ή ένα ένθετο αντικείμενο πρέπει αρχικά να αφαιρέσουμε από την εργασία όλα τα αντίγραφα του (instances) και τις εντολές κώδικα που αναφέρονται σε αυτά. Στη συνέχεια εμφανίζουμε πάλι το πλαίσιο διαλόγου Components, αποεπιλέγουμε το πλαίσιο ελέγχου που βρίσκεται αριστερά από το όνομά του και πατάμε το πλήκτρο εντολής OK. Προσοχή, αν δεν έχουμε αφαιρέσει από την εργασία μας τα αντίγραφά του αντικειμένου και τις αναφορές κώδικα που γίνονται σε αυτό, θα εμφανιστεί μήνυμα λάθους.

1.5. Ο εξερευνητής εργασίας

Κάθε εφαρμογή που δημιουργείται στο περιβάλλον της Visual Basic είναι δυνατό να περιέχει διαφορετικούς τύπους αρχείων. Το σύνολο όλων των αρχείων που αποτελούν μία εφαρμογή στη Visual Basic, κατά το σχεδιασμό της περιέχονται σε μια εργασία (Project). Ο εξερευνητής εργασίας (Σχήμα 7) περιέχει κάθε φορά τα ονόματα όλων των αρχείων της τρέχουσας εργασίας.



Σχ.7 Ο εξερευνητής εργασίας.

Αν δεν εμφανίζεται το παράθυρο εργασίας στην οθόνη, πρέπει να επιλέξουμε την επιλογή Project Explorer του μενού View.

Τα επιμέρους αρχεία που αποτελούν μία εργασία της Visual Basic αποθηκεύονται ξεχωριστά στο δίσκο του συστήματος. Η λίστα όλων των αρχείων που αποτελούν την εργασία καθώς και οι πληροφορίες για τη διεύθυνση τους επάνω στο δίσκο του συστήματος αποθηκεύονται στο **αρχείο εργασίας** (Project file). Ακόμη ένα αρχείο εργασίας είναι δυνατό να περιέχει τις αναφορές σύνδεσης (references) και πληροφορίες ή ειδικές ρυθμίσεις που τυχόν έχουμε κάνει για τη συγκεκριμένη εργασία, τα αντικείμενα που περιέχει, το περιβάλλον και το εκτελέσιμο αρχείο της.

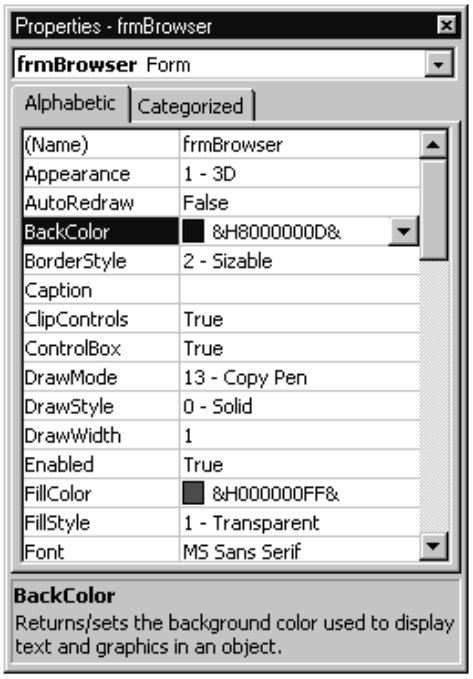
Οι βασικοί τύποι αρχείων που συνήθως περιέχει μια εργασία και συνεπώς το παράθυρο εργασίας της Visual Basic είναι:

- ✓ **Αρχεία φόρμας (Form modules):** Περιλαμβάνουν τα οπτικά στοιχεία και τις ιδιότητες μίας φόρμας και των εργαλείων που περιέχει, καθώς και τις μεταβλητές, τις σταθερές και τις ρουτίνες κώδικα που συνοδεύουν τη φόρμα. Κάθε αρχείο φόρμας που δημιουργείται σε μία εφαρμογή της Visual Basic αποθηκεύεται ανεξάρτητα στο σκληρό δίσκο με επέκταση ονόματος .FRM.
- ✓ **Αρχεία βασικής προγραμματιστικές μονάδες (Standard modules):** Είναι αρχεία κώδικα, τα οποία περιέχουν τις ρουτίνες, τις συναρτήσεις και τις δηλώσεις και τους ορισμούς των γενικών μεταβλητών, των δεδομένων και των τύπων δεδομένων κάθε προγράμματος της Visual Basic. Τα αρχεία Βασικής προγραμματιστικής μονάδας έχουν επέκταση ονόματος .BAS.
- ✓ **Αρχεία προγραμματιστικής μονάδας κλάσης (Class modules):** Είναι επίσης αρχεία κώδικα και τα χρησιμοποιούμε για να ορίσουμε καινούργια αντικείμενα. Τα αρχεία κλάσης είναι παρόμοια με τα αρχεία φόρμας, αλλά δεν περιέχουν οπτικά στοιχεία. Ένα αρχείο κλάσης είναι δυνατό να περιέχει ορισμούς ιδιοτήτων και μεθόδων του αντικειμένου που ορίζουμε, όχι όμως κάποια γεγονότα. Τα αρχεία κλάσης έχουν επέκταση ονόματος .CLS.
- ✓ **Αρχείο πηγή (Resource file):** Στη Visual Basic 4.0 έχουμε τη δυνατότητα όταν γράφουμε κώδικα να χρησιμοποιήσουμε δεδομένα από κάποιο αρχείο Πηγή. Τα αρχεία Πηγή είναι ανεξάρτητα αρχεία με επέκταση ονόματος .RES και περιέχουν διάφορους τύπους δεδομένων (π.χ. αρχεία bitmap ή τμήματα κειμένου), τα οποία μπορούμε να χρησιμοποιήσουμε χωρίς να χρειάζεται συνεχή επεξεργασία στον κώδικα τους. Με το τρόπο αυτό αυξάνονται οι δυνατότητες εκτέλεσης της εφαρμογής, γιατί φορτώνουμε κατά απαίτηση στη μνήμη του συστήματος τα δεδομένα που πραγματικά χρειαζόμαστε και όχι αναγκαστικά όλα όσα περιέχει μια φόρμα ή μια προγραμματιστική μονάδα

1.6.Το παράθυρο ιδιοτήτων

Το παράθυρο ιδιοτήτων (Properties window), εμφανίζεται στην αρχική οθόνη της Visual Basic (Σχήμα 8) και περιέχει κάθε φορά μία λίστα με τις ιδιότητες του επιλεγμένου αντικειμένου (εργαλείο, φόρμα, ή κλάση), προγραμματιστικής μονάδας ή μενού επιλογών. Οι ιδιότητες είναι τα δεδομένα κάθε αντικειμένου και καθορίζουν τα χαρακτηριστικά της εμφάνισης του (π.χ. τη θέση ή το χρώμα) και τη συμπεριφορά του (π.χ. πως θα αντιδράσει στην εισαγωγή στοιχείων του χρήστη ή αν είναι ενεργό).

Οι τιμές των περισσότερων ιδιοτήτων ενός γραφικού αντικειμένου μπορούν να μεταβληθούν κατά το σχεδιασμό της εφαρμογής μέσω του παραθύρου ιδιοτήτων. Το παράθυρο ιδιοτήτων συνήθως βρίσκεται στο δεξί τμήμα της αρχικής οθόνης της Visual Basic και μπορούμε να το μετακινήσουμε ή να μεταβάλλουμε τις διαστάσεις του με τη χρήση του ποντικιού όπως ένα οποιοδήποτε άλλο παράθυρο των Windows.



Σχ.8 Το παράθυρο ιδιοτήτων.

Αν το παράθυρο ιδιοτήτων δεν εμφανίζεται στην οθόνη, επιλέγουμε από το μενού View την επιλογή Properties Window ή πατάμε το πλήκτρο συντομίας F4 ή εναλλακτικά επιλέγουμε το αντίστοιχο εικονίδιο της γραμμής εργαλείων. Ακόμη το παράθυρο ιδιοτήτων μπορούμε να το εμφανίσουμε από το μενού συντομίας, που εμφανίζεται όταν πατήσουμε το δεξί πλήκτρο του ποντικιού ενώ βρισκόμαστε επάνω σε κάποιο γραφικό αντικείμενο της Visual Basic.

Στο επάνω τμήμα του παραθύρου ιδιοτήτων εμφανίζονται τα κλασικά στοιχεία ενός παραθύρου των Windows και στη συνέχεια υπάρχει το πλαίσιο των αντικειμένων (Object box) μέσα στο οποίο εμφανίζεται κάθε φορά το όνομα του ενεργού αντικειμένου της εργασίας. Πατώντας το βέλος που εικονίζεται στο δεξί τμήμα του πλαισίου αντικειμένων, εμφανίζεται μια λίστα που περιέχει την ενεργή φόρμα και όλα τα αντικείμενά της. Επιλέγοντας από τη λίστα ένα όνομα, το αντίστοιχο αντικείμενο έρχεται στο προσκήνιο και μετατρέπεται αυτόματα σε ενεργό αντικείμενο της εργασίας.

Στο κάτω τμήμα του παραθύρου εμφανίζονται δύο καρτέλες από τις οποίες επιλέγουμε τον τρόπο εμφάνισης των ιδιοτήτων, αλφαβητικά (Alphabetic) ή κατηγοριοποιημένες (Categorized). Αμέσως μετά, υπάρχει η λίστα ιδιοτήτων (Properties list), η οποία αποτελείται από δύο στήλες και περιέχει όλες τις ιδιότητες του ενεργού αντικειμένου και τις επιλεγμένες τιμές τους. Για

να μεταβάλουμε την τιμή μιας ιδιότητας, επιλέγουμε την ιδιότητα στη λίστα ιδιοτήτων και πληκτρολογούμε τη νέα τιμή στη πλαίσιο της δεξιάς στήλης που ονομάζεται πλαίσιο τιμών (Settings box). Σε ορισμένες ιδιότητες υπάρχουν προκαθορισμένες τιμές, οι οποίες εμφανίζονται με μορφή λίστας και μπορούμε να τις εμφανίσουμε και να επιλέξουμε κάποια τιμή αναδιπλώνοντας τα περιεχόμενά της λίστας, με το βελάκι που εμφανίζεται δεξιά από το πλαίσιο τιμών.

Στο σημείο αυτό πρέπει να σημειώσουμε ότι ορισμένα αντικείμενα τις Visual Basic περιέχουν και ιδιότητες που δεν είναι διαθέσιμες κατά τη διάρκεια του σχεδιασμού αλλά παίρνουν τιμές μόνο μέσω εντολών κώδικα κατά τη διάρκεια της εκτέλεσης της εφαρμογής.

1.7. Το παράθυρο μορφής φόρμας

Το παράθυρο μορφής φόρμας (form layout window) μας επιτρέπει κατά το σχεδιασμό της εφαρμογής να ρυθμίσουμε οπτικά τη θέση της φόρμας (Σχήμα 9).

Στο παράθυρο μορφής φόρμας, κατά το χρόνο σχεδιασμού εμφανίζονται όλες οι φόρμες που είναι ορατές στο περιβάλλον εργασίας στην σωστή θέση και με τις αντίστοιχες διαστάσεις.

***Σχ.9** Το παράθυρο μορφής φόρμας.*

2. ΑΡΧΕΣ ΣΥΓΧΡΟΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Ως τώρα παρουσιάσαμε τα στοιχεία που αποτελούν το περιβάλλον ανάπτυξης εφαρμογών της Visual Basic. Το βασικότερο συμπέρασμα, είναι ότι στο περιβάλλον της Visual Basic τα εργαλεία προγραμματισμού διαφοροποιούνται από τα μέχρι σήμερα γνωστά εργαλεία προγραμματισμού του DOS και ακολουθούν τη φιλοσοφία του γραφικού περιβάλλοντος των Windows.

Στο περιβάλλον ανάπτυξης εφαρμογών της Visual Basic, βλέπουμε ότι εγκαταλείπεται ο ανιαρός τρόπος χειρισμού των εργαλείων μιας γλώσσας προγραμματισμού μέσω εντολών κώδικα. Χαρακτηριστικά μπορούμε να αναφέρουμε ότι δεν υπάρχει ένας απλός επεξεργαστής κειμένου μέσα στον οποίο αναπτύσσουμε την εφαρμογή, αλλά ένα πλήρες γραφικό περιβάλλον ανάπτυξης, μέσα στο οποίο χρησιμοποιούμε πρακτικά και εύκολα τα εργαλεία που προσφέρονται για το σχεδιασμό μιας εφαρμογής.

2.1. Εισαγωγή στον προγραμματισμό μιας εφαρμογής

Οι περισσότεροι από εμάς γνωρίζουμε ότι η φιλοσοφία ανάπτυξης μίας εφαρμογής διαφοροποιείται ανάλογα με το περιβάλλον μέσα στο οποίο πρόκειται να εργαστεί. Στο περιβάλλον των Windows ο προγραμματιστής πρέπει να έχει πάντοτε υπόψη, ότι πρόκειται να εργαστεί μέσα σε ένα γραφικό περιβάλλον. Άρα θα σχεδιάσει την εφαρμογή του βλέποντας την πάντοτε από την άποψη του χρήστη. Επίσης εκμεταλλευόμενος τα πλεονεκτήματα σχεδιασμού που προσφέρει το γραφικό περιβάλλον εργασίας των Windows πρέπει να δημιουργήσει ένα φιλικό και λειτουργικό τρόπο επικοινωνίας χρήστη-εφαρμογής.

Ο σχεδιασμός της εφαρμογής ξεκινά πάντοτε από τη δημιουργία του τρόπου επικοινωνίας χρήστη-εφαρμογής. Ο λόγος είναι προφανής. Σε μία εφαρμογή που εργάζεται στο περιβάλλον των Windows και συνεπώς και της Visual Basic, η ροή εκτέλεσής της καθορίζεται κάθε φορά από τις αντιδράσεις του χρήστη και του συστήματος. Ο χρήστης κάθε χρονική στιγμή είναι ο κυρίαρχος της εφαρμογής και μπορεί να επεμβαίνει και να καθοδηγεί τη ροή εκτέλεσής της, στέλνοντας σε αυτή ένα μήνυμα (message) ή προκαλώντας ένα γεγονός (event), μέσω των γραφικών αντικειμένων που αποτελούν το μέσο επικοινωνίας εφαρμογής-χρήστη. Επίσης στη ροή εκτέλεσης της εφαρμογής είναι πιθανό να επιδράσει και το ίδιο το σύστημα, αφού έχει τη δυνατότητα να προκαλέσει ένα εσωτερικό μήνυμα και να μεταβάλλει τις συνθήκες μέσα στο περιβάλλον εργασίας. Αφού η ροή εκτέλεσης μίας εφαρμογής εξαρτάται από την επικοινωνία του χρήστη και του συστήματος με τα αντικείμενα της εφαρμογής, εύκολα γίνεται κατανοητό ότι δε μπορούμε να ξεκινήσουμε το γράψιμο του κώδικα αν δε ξέρουμε πρώτα ποιες θα είναι οι πιθανές παρεμβάσεις τους κατά την εκτέλεση της εφαρμογής.

Σύμφωνα με τα παραπάνω μπορούμε να περιγράψουμε την ανάπτυξη μίας εφαρμογής με τη Visual Basic σαν διαδικασία τριών βημάτων:

- Σχεδιασμός του τρόπου επικοινωνίας χρήστη-εφαρμογής.
- Καθορισμός της αρχικής συμπεριφοράς των αντικειμένων της εφαρμογής μέσω των ιδιοτήτων που τα χαρακτηρίζουν.
- Δημιουργία και εκσφαλμάτωση των ρουτινών κώδικα.

Με τα δύο πρώτα βήματα δημιουργούμε τον τρόπο επικοινωνίας χρήστη-εφαρμογής και εισάγουμε τα αρχικά στοιχεία της εφαρμογής. Με το τρίτο βήμα γράφουμε τον κώδικα, ο οποίος θα καθοδηγεί την εφαρμογή και θα εκτελεί ακριβώς τις εργασίες που εμείς επιθυμούμε, μετά από κάποια αντίδραση του χρήστη ή του συστήματος. Για τη δημιουργία του κώδικα μίας εφαρμογής με τη Visual Basic συνδυάζονται οι τεχνικές του δομημένου (structural) αλλά και του τμη-

ματικού προγραμματισμού (modular). Ο δομημένος προγραμματισμός είναι λίγο πολύ γνωστός στους περισσότερους προγραμματιστές, αφού χρησιμοποιείται στην ανάπτυξη εφαρμογών στο περιβάλλον του DOS, με τις γλώσσες προγραμματισμού τρίτης γενιάς. Ο τμηματικός προγραμματισμός επιτυγχάνεται με τη χρήση δύο τεχνικών προγραμματισμού, του αντικειμενοστραφή (object-oriented) και του οδηγούμενου από τα γεγονότα (event-driven) προγραμματισμού. Οι δύο αυτές τεχνικές καθορίζουν και τον τρόπο με τον οποίο σχεδιάζεται μία εφαρμογή στο περιβάλλον της Visual Basic.

2.2. Αντικειμενοστραφής προγραμματισμός

Σε ένα πραγματικά αντικειμενοστραφές περιβάλλον, η εφαρμογή του αντικειμενοστραφή προγραμματισμού (Object-Oriented programming) γίνεται με τον κερματισμό της εφαρμογής σε αντικείμενα, καθένα από τα οποία αποτελεί ξεχωριστή οντότητα. Κάθε αντικείμενο περιέχει το δικό του κώδικα και δεδομένα και έχει ανεξάρτητη συμπεριφορά μέσα στο περιβάλλον εργασίας. Ο σωστός χειρισμός και η κατάλληλη συνεργασία των αντικειμένων του προγράμματος, θα επιφέρουν την επιτυχή ολοκλήρωση της εφαρμογής.

Η Visual Basic για να απλοποιήσει τον προγραμματισμό στο περιβάλλον των Windows περιέχει όπως είδαμε τα δικά της εσωτερικά αντικείμενα, μέσω των οποίων επιτυγχάνεται η εφαρμογή του αντικειμενοστραφή προγραμματισμού στο περιβάλλον της. Ας περιγράψουμε όμως τον κόσμο των αντικειμένων της Visual Basic. Στην καθημερινή μας ζωή όταν βλέπουμε ένα φυτό το αντιμετωπίζουμε σαν ένα απλό αντικείμενο. Αν όμως το προσέξουμε καλύτερα θα παρατηρήσουμε ότι το φυτό αποτελείται από περισσότερα αντικείμενα όπως τα φύλλα του, τη ρίζα του, τα λουλούδια του κ.ά. Στη συνέχεια μπορούμε να παρατηρήσουμε ότι το κάθε επιμέρους αντικείμενο περιέχει τα δικά του δεδομένα, δηλαδή τις δικές του ιδιότητες (π.χ. τα φύλλα έχουν πράσινο χρώμα) και τη δική του συμπεριφορά (π.χ. τα λουλούδια ανθίζουν). Ακόμη κάθε αντικείμενο έχει το δικό του τρόπο επικοινωνίας με το περιβάλλον (π.χ. τα φύλλα αναπνέουν μέσα από τους πόρους τους).

Με τον ίδιο τρόπο μπορούμε να περιγράψουμε μία εφαρμογή της Visual Basic. Εξετάζοντας προσεκτικά τη διαχείριση των προκαθορισμένων γραφικών αντικειμένων στο περιβάλλον της εφαρμογής θα παρατηρήσουμε ότι γίνεται περίπου με τον ίδιο τρόπο που διαχειριζόμαστε τα αντικείμενα του φυσικού περιβάλλοντος.

Πρώτα από όλα η ίδια η εφαρμογή μπορεί να χαρακτηριστεί σαν ένα ανεξάρτητο αντικείμενο. Ένα αντικείμενο περιέχει τα δικά του δεδομένα και το τρόπο επικοινωνίας για να επικοινωνεί με το περιβάλλον του. Τα στοιχεία αυτά είναι χαρακτηριστικά κάθε εφαρμογής της Visual Basic. Στη συνέχεια μπορούμε να παρατηρήσουμε ότι η εφαρμογή αποτελείται από πολλά γραφικά αντικείμενα όπως φόρμες, πλήκτρα επιλογής κ.α. Αναλύοντας τα επιμέρους στοιχεία της εφαρμογής θα δούμε ότι και αυτά με τη σειρά τους μπορούν να χαρακτηριστούν αυτόνομα αντικείμενα, τα οποία περιέχουν τις δικές τους ιδιότητες και μεθόδους που καθορίζουν την εμφάνιση και τη συμπεριφορά τους μέσα στο περιβάλλον εργασίας και φυσικά το δικό τους μέσο επικοινωνίας, με το οποίο αναγνωρίζουν τα μηνύματα του χρήστη και του συστήματος.

Η Visual Basic περιέχει αρκετά ενσωματωμένα αντικείμενα. Κάθε αντικείμενο της Visual Basic, προέρχεται από μια κλάση. Μια κλάση, αντιπροσωπεύεται από ένα εργαλείο της εργαλειοθήκης ή δημιουργείται μέσω κώδικα. Όλα τα αντικείμενα, δημιουργούνται σαν ακριβή αντίγραφα ή στιγμιότυπα (instances) των κλάσεων τους. Όταν τα αντικείμενα αποκτήσουν ανεξάρτητη υπόσταση, μπορούμε να διαφοροποιήσουμε κάποια από τα χαρακτηριστικά τους. Έτσι τις περισσότερες φορές σε μια εφαρμογή συναντάμε αντικείμενα που προέρχονται από την ίδια κλάση αντικειμένων (π.χ. εργαλεία πλαισίου κειμένου), τα οποία έχουν διαφορετική εμφάνιση.

Κάθε αντικείμενο υποστηρίζει τις δικές του ιδιότητες (properties) και μεθόδους (methods). Ο προγραμματιστής χρειάζεται να διαχειριστεί αυτά τα χαρακτηριστικά για να συνδέσει τα αντικείμενα μεταξύ τους και να επιτύχει την αρμονική συνεργασία τους μέσα στην εφαρμογή. Οι ιδιότητες προσδιορίζουν την εμφάνιση των αντικειμένων της Visual Basic. Κάθε αντικείμενο περιέχει τις δικές του ιδιότητες. Για παράδειγμα μια φόρμα (Form) χαρακτηρίζεται από τις ιδιότητες AutoRedraw, BackColor, BorderStyle, Caption, Font, Name κ.α. Για να ορίσουμε τα χαρακτηριστικά ενός αντικειμένου πρέπει πρώτα φυσικά να δημιουργήσουμε το ίδιο. Στη συνέχεια αντιστοιχούμε σε αυτό τις ιδιότητες που θέλουμε να το περιγράφουν. Αν για παράδειγμα επιλέξουμε το εργαλείο πλήκτρο εντολής (command button) και το τοποθετήσουμε επάνω στη φόρμα, στη συνέχεια μπορούμε να ορίσουμε τον τίτλο του, τη θέση του μέσα στη φόρμα, τις διαστάσεις του κλπ.

Η τιμή μιας ιδιότητας ορίζεται κατά το σχεδιασμό της εφαρμογής μέσω του παραθύρου ιδιοτήτων (Properties window) της Visual Basic. Η μετατροπή της τιμής της όμως κατά την εκτέλεση της εφαρμογής μπορεί να γίνει μόνο μέσω εντολών κώδικα. Για να αναφερθούμε σε μια ιδιότητα ενός γραφικού αντικειμένου πρέπει πρώτα να αναφερθούμε στο ίδιο το αντικείμενο, γιατί όπως είναι φανερό δύο διαφορετικά αντικείμενα είναι δυνατό να περιέχουν την ίδια ιδιότητα. Για παράδειγμα, τα εργαλεία φόρμα και πλήκτρο εντολής υποστηρίζουν και τα δύο την ιδιότητα Caption (Επικεφαλίδα). Αν θέλουμε να αναφερθούμε στην ιδιότητα Caption της φόρμας χρησιμοποιούμε την εντολή:

```
Form1.Caption = "Πρώτη Φόρμα"
```

Όταν θέλουμε να αναφερθούμε στην ιδιότητα ενός αντικειμένου που βρίσκεται σε διαφορετική φόρμα εργασίας, πρέπει να αναφέρουμε και το όνομα της φόρμας με την παρακάτω σύνταξη:

Όνομα_φόρμας! Όνομα_αντικειμένου. Ιδιότητα

Ένα αντικείμενο εκτός από ιδιότητες υποστηρίζει και μεθόδους (methods). Οι μέθοδοι είναι στην πραγματικότητα ενέργειες που μπορούν να επιδράσουν στα αντικείμενα της εφαρμογής, μέσω μηνυμάτων που στέλνουν σε αυτά. Για να στείλει ο χρήστης ένα μήνυμα σε ένα αντικείμενο προσδιορίζει το όνομα του και τη μέθοδο μέσω της οποίας θα το στείλει. Για παράδειγμα αν θέλουμε να εισάγουμε σε ένα εργαλείο Λίστας (List), μια καινούρια επιλογή μέσω της μεθόδου AddItem χρησιμοποιούμε την εντολή:

List1.AddItem "Πρώτη επιλογή"

Μέσα στο περιβάλλον της Visual Basic υπάρχει η δυνατότητα να ανταλλάσσουν μηνύματα και δυο διαφορετικά αντικείμενα μιας εφαρμογής. Η ανταλλαγή των μηνυμάτων ανάμεσα στα δύο αντικείμενα επιτυγχάνεται επίσης μέσω των μεθόδων της Visual Basic. Για να αντιγράψουμε τα περιεχόμενα ενός Πλαισίου κειμένου (Text box) στο ειδικό αντικείμενο Clipboard χρησιμοποιούμε τη μέθοδο SetText:

Clipboard.SetText Text1.Text

Τέλος, μηνύματα μπορούν να ανταλλάσσουν και δύο διαφορετικές εφαρμογές μέσα στο περιβάλλον των Windows. Η εφαρμογή κατά την εκτέλεση της μπορεί να ανταλλάσσει μηνύματα με άλλες εφαρμογές μέσω του συστήματος επικοινωνίας API των Windows. Αν χρειαστεί για παράδειγμα μια εφαρμογή της Visual Basic, τη συνεργασία μίας άλλης εφαρμογής για την εκτέλεση μίας εργασίας, στέλνει ένα μήνυμα στην αντίστοιχη εφαρμογή, αυτή με τη σειρά της αναλαμβάνει τον έλεγχο του συστήματος και μόλις ολοκληρώσει την εργασία, απαντάει με τα αποτελέσματα της. Για να στείλουμε το περιεχόμενο ενός πλαισίου κειμένου, σε μια άλλη εφαρμογή των Windows (π.χ. Word) δημιουργούμε μια σύνδεση (DDE Link) ανάμεσα στις δύο εφαρμογές και χρησιμοποιούμε τη μέθοδο LinkSend:

Picture1.LinkSend

Η δημιουργία της διεπαφής χρήστη Visual Basic, χαρακτηρίζεται από τα αντικείμενα που το αποτελούν. Κατά το σχεδιασμό του θα πρέπει ο προγραμματιστής να έχει τη σκέψη του κυρίως στα αντικείμενα που θα συμπεριλάβει σε αυτό και στον τρόπο που θα μπορεί να μεταβάλλει τη συμπεριφορά τους.

Τα αντικείμενα που υποστηρίζει η Visual Basic είναι:

- ✓ Οι **φόρμες** αποτελούν τα παράθυρα επικοινωνίας της εφαρμογής με το χρήστη. Ένα παράθυρο μπορεί για παράδειγμα να είναι μία οθόνη εμφάνισης στοιχείων ή ένα πλαίσιο διαλόγου του χρήστη με την εφαρμογή.
- ✓ Τα **εργαλεία** (controls), αποτελούν μαζί με τις φόρμες τα "υλικά" σχεδιασμού της εφαρμογής. Μια εφαρμογή της Visual Basic είναι δυνατό να περιέχει και άλλα αντικείμενα:
- ✓ Τα **ειδικά αντικείμενα** της Visual Basic δεν έχουν οπτική εμφάνιση αλλά χρησιμοποιούνται για την εκτέλεση ειδικών εργασιών. Τέτοια αντικείμενα είναι το Clipboard, ο Debugger, ο Εκτυπωτής και η Οθόνη. Τα αντικείμενα αυτά Η Visual Basic διαχειρίζεται τα ειδικά αντικείμενα μέσω των μεθόδων που υποστηρίζουν. Τα ειδικά αντικείμενα μπορεί να υποστηρίζουν μεθόδους, αλλά δεν έχουν δικές τους ιδιότητες. Για παράδειγμα έχουμε τη δυνατότητα να αντιγράφουμε δεδομένα στο αντικείμενο Clipboard, αλλά δεν μπορούμε να μετατρέψουμε τα χαρακτηριστικά του, γιατί δεν έχει οπτική υπόσταση.
- ✓ Τα **αντικείμενα δεδομένων** (Database objects). Κάθε βάση δεδομένων υποστηρίζει ένα αριθμό ειδικών αντικειμένων με το χειρισμό των οποίων μπορούμε να έχουμε πρόσβαση στα δεδομένα της.
- ✓ **Κλάσεις** (Classes). Η Visual Basic ως την τρίτη έκδοσή της δε μπορούσε να χαρακτηριστεί σαν μία πλήρη αντικειμενοστραφή γλώσσα προγραμματισμού, γιατί ο προγραμματιστής δεν ήταν δυνατό να δημιουργήσει στο περιβάλλον της τα δικά του αντικείμενα, ούτε μπορούσε να αντιστοιχήσει καινούριες ιδιότητες και μεθόδους στα ήδη υπάρχοντα. Από την τέταρτη έκδοση της Visual Basic, έχουμε τη δυνατότητα σε μια εφαρμογή να δημιουργήσουμε δικά μας αντικείμενα προσαρμοσμένα σε συγκεκριμένες ανάγκες, τα οποία θα περιέχουν τις δικές τους ιδιότητες και μεθόδους. Ο ορισμός ενός τέτοιου αντικειμένου, καλείται κλάση και πραγματοποιείται μέσα από αρχεία κώδικα.

Μία αντικειμενοστραφή εφαρμογή επιτρέπει στον προγραμματιστή της να χρησιμοποιεί αρκετές φορές τα ίδια αντικείμενα. Με την τεχνική αυτή γλιτώνει αρκετό χρόνο και απαλλάσσεται από την επανάληψη του κώδικα. Με την Visual Basic έχουμε τη δυνατότητα να επαναχρησιμοποιήσουμε όλα τα αντικείμενα της, ακόμη και αλλάζοντας μερικές μόνο από τις ιδιότητες τους.

2.3. Προγραμματισμός οδηγούμενος από τα γεγονότα

Στην καθημερινή μας ζωή, κάθε χρονική στιγμή είναι πιθανό να συμβεί ένα γεγονός το οποίο μπορεί να μεταβάλλει την κατάσταση των αντικειμένων που μας περιβάλουν. Για παράδειγμα, το άνοιγμα ενός παραθύρου θα επιτρέψει την είσοδο του αέρα μέσα στο δωμάτιο και θα μετακινηθούν αρκετά αντικείμενα.

Με τον ίδιο τρόπο σε ένα αντικειμενοστραφές περιβάλλον όπως το περιβάλλον της Visual Basic, ένα γεγονός μπορεί να προκαλέσει τη μεταβολή της κατάστασης των γραφικών αντικειμένων και να επιδράσει στη ροή εκτέλεσης της εφαρμογής. Κατά την εκτέλεση της εφαρμογής ένα γεγονός μπορεί να προκληθεί από το χρήστη, όπως το πάτημα ενός πλήκτρου από το πληκτρολόγιο προκαλεί το Keypress γεγονός, το σύστημα, όπως ένα χρονικό μήνυμα του εσωτερικού ρολογιού προκαλεί το Timer γεγονός, ή έμμεσα από το κώδικα όπως η εντολή με την οποία "φορτώνουμε" μια φόρμα προκαλεί το γεγονός Load.

Ο τρόπος για να καθοδηγούμε τη ροή εκτέλεσης μίας εφαρμογής της Visual Basic, είναι να εφαρμόσουμε τις αρχές του οδηγούμενου από τα γεγονότα προγραμματισμού. Ουσιαστικά πρέπει να γράψουμε τον κώδικα που θα κάνει τα γραφικά αντικείμενα να αντιδρούν στα γεγονότα, που πιθανών να συμβούν μέσα στο περιβάλλον εργασίας τους.

Ας δούμε όμως, πως εφαρμόζεται η τεχνική του οδηγούμενου από τα γεγονότα προγραμματισμού σε μία εφαρμογή της Visual Basic. Σε κάθε γραφικό αντικείμενο της Visual Basic μπορούμε να συμπεριλάβουμε εντολές κώδικα, οι οποίες θα εκτελούνται μετά από την επίδραση ενός γεγονότος. Ο κώδικας για κάθε γεγονός που πιθανών πρόκειται να συμβεί βρίσκεται σε ανεξάρτητες ρουτίνες, οι οποίες ονομάζονται ρουτίνες γεγονότων (event procedures).

Κάθε ρουτίνα γεγονότος χαρακτηρίζεται από το αντικείμενο και το γεγονός που θα την ενεργοποιήσει. Για παράδειγμα για να γράψουμε κώδικα για την περίπτωση που θέλουμε να εκτελεστεί μία εργασία όταν "φορτώνεται" μία φόρμα της εφαρμογής δημιουργούμε μία ρουτίνα η οποία θα τρέχει όταν αναγνωρίσει το γεγονός "φόρτωμα" (Load) της φόρμας. Η σύνταξη της ρουτίνας είναι ως εξής:

Sub Form_Load().

Το μεγάλο πλεονέκτημα της Visual Basic είναι ότι τα γεγονότα που συμβαίνουν μέσα στο περιβάλλον της, αναγνωρίζονται αυτόματα από τα γραφικά αντικείμενα που αποτελούν το μέσο επικοινωνίας της εφαρμογής και δε χρειάζεται να γράψουμε επιπλέον κώδικα για το σκοπό αυτό. Τα γραφικά αντικείμενα της Visual Basic δεν αντιδρούν σε κάθε γεγονός που προκαλείται στο περιβάλλον εργασίας τους, παρά μόνο σε αυτά που τα αφορούν, αφού κάθε γραφικό αντικείμενο της Visual Basic υποστηρίζει ένα προκαθορισμένο αριθμό γεγονότων (Events).

Το εργαλείο Πλαίσιο Ελέγχου (Check box) για παράδειγμα υποστηρίζει τα γεγονότα:

Click	KeyDown	KeyUp	KeyPress
DragOver	DragDrop	GotFocus	LostFocus

Κατά την ανάπτυξη μίας εφαρμογής μόλις σχεδιάσουμε ένα γραφικό αντικείμενο, η Visual Basic δημιουργεί αυτόματα στο παράθυρο κώδικα (code window) τα πρότυπα των ρουτινών γεγονότος για κάθε γεγονός που υποστηρίζει. Ο προγραμματιστής στη συνέχεια, επιλέγει τα γεγονότα που θέλει να συμπεριλάβει στην εφαρμογή και συμπληρώνει τις γραμμές κώδικα. Σε μία εφαρμογή δεν είναι απαραίτητο για κάθε γραφικό αντικείμενο να συμπεριλάβουμε κώδικα σε όλες τις ρουτίνες γεγονότων, αλλά επιλέγουμε και γράφουμε κώδικα μόνο για τα γεγονότα τα οποία μας ενδιαφέρουν και θέλουμε να αντιδρά η εφαρμογή.

Ας δούμε όμως πως αντιδρούν τα γραφικά αντικείμενα της Visual Basic όταν συμβεί ένα γεγονός στο περιβάλλον τους. Κατά την εκτέλεση της εφαρμογής τα γραφικά αντικείμενά που περιέχει βρίσκονται σε κατάσταση αναμονής. Μόλις συμβεί στο περιβάλλον ένα γεγονός, αναγνωρίζεται αυτόματα από το γραφικό αντικείμενο που το υποστηρίζει και η Visual Basic ελέγχει αν έχουμε γράψει κώδικα στην αντίστοιχη ρουτίνα γεγονότος. Αν υπάρχουν γραμμένες κάποιες εντολές κώδικα προχωράει στην εκτέλεση τους, διαφορετικά αγνοεί το γεγονός. Μετά την ολοκλήρωση της εκτέλεσης του κώδικα, τα γραφικά αντικείμενα επιστρέφουν πάλι σε κατάσταση αναμονής και η εφαρμογή περιμένει το επόμενο γεγονός να συμβεί.

Ακόμη πρέπει να τονίσουμε, ότι ο προγραμματιστής μπορεί να προκαλεί γεγονότα μέσω εντολών κώδικα. Για παράδειγμα όταν "φορτώνεται" μια φόρμα, μπορούμε να συμπεριλάβουμε στην αντίστοιχη ρουτίνα γεγονότος, μια εντολή κώδικα με την οποία θα μετατρέπουμε την τιμή της ιδιότητας Text και θα προκαλείται η αλλαγή της κατάστασης ενός πλαισίου κειμένου. Τα γεγονότα που προκαλούνται μέσα από κώδικα καλούνται trigger events.

Ένα εξωτερικό μήνυμα, μπορεί να προκαλέσει δύο ή περισσότερα διαφορετικά γεγονότα στην εφαρμογή. Για παράδειγμα όταν συμβεί το γεγονός Dblclick (διπλό πάτημα) του αριστερού πλήκτρου του ποντικιού, συγχρόνως συμβαίνουν και τα γεγονότα Click, MouseDown και MouseUp. Έτσι η Visual Basic θα ανιχνεύσει για εντολές κώδικα σε περισσότερες από μία ρουτίνες γεγονότων.

Θα μπορούσαμε λοιπόν να παρουσιάσουμε μία εφαρμογή που ακολουθεί την τεχνική του οδηγούμενου από τα γεγονότα προγραμματισμού, με τα παρακάτω βήματα:

1. Η εκτέλεση της εφαρμογής ξεκινά και "φορτώνεται" η πρώτη φόρμα της.
2. Τα γραφικά αντικείμενα της εφαρμογής βρίσκονται σε κατάσταση αναμονής.
3. 'Οταν ένα γραφικό αντικείμενο αναγνωρίσει ένα γεγονός που το αφορά, καλεί την ανάλογη ρουτίνα γεγονότος.
4. Αν η ρουτίνα γεγονότος περιέχει κώδικα εκτελείται, διαφορετικά η εφαρμογή αγνοεί το γεγονός.
5. Τα αντικείμενα της εφαρμογής, περνούν πάλι σε κατάσταση αναμονής.

Ένα σημείο, που πρέπει να προσέξουμε ιδιαίτερα στην εφαρμογή του οδηγούμενου από τα γεγονότα προγραμματισμού, είναι η αποφυγή της εκτέλεσης μιας ενέργειας μέσα από μια ρουτίνα γεγονότος που θα προκαλεί το ίδιο γεγονός. Αυτό το γεγονός ονομάζεται cascading event και θα έχει ως συνέπεια τη συνεχή εκτέλεση της ίδιας ρουτίνας, μέχρι να προκληθεί πρόβλημα μνήμης στη Visual Basic (Out of stack space error). Για παράδειγμα, όταν συμπεριλάβουμε στην ρουτίνα γεγονότος αλλαγή κατάστασης ενός πλαισίου κειμένου, μια εντολή που θα μετατρέπει την ιδιότητά του Text, θα έχουμε επανάληψη του ιδίου γεγονότος και θα καλείται συνεχώς η ίδια ρουτίνα.

3. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αν συγκρίνουμε μία παραδοσιακή Basic που εργάζεται στο περιβάλλον του DOS με τη Visual Basic, η βασικότερη διαφορά στην οποία πρέπει να σταθούμε βρίσκεται στη φιλοσοφία προγραμματισμού. Όπως είδαμε, ο σχεδιασμός μίας εφαρμογής με τη Visual Basic στηρίζεται στην τεχνική του αντικειμενοστραφή και του οδηγούμενου από τα γεγονότα προγραμματισμού. Ο λόγος είναι ότι η εφαρμογή πρόκειται να εργαστεί μέσα σε ένα αντικειμενοστραφές περιβάλλον, όπως των Windows όπου η ροή εκτέλεσης κάθε εφαρμογής προσδιορίζεται από τα γεγονότα που συμβαίνουν κάθε χρονική στιγμή και τα μηνύματα που δέχεται ο τρόπος διασύνδεσης και επικοινωνίας της, μέσα στο περιβάλλον εργασίας. Αντιθέτως μία εφαρμογή που έχει δημιουργηθεί σε μια κλασική Basic και εργάζεται στο περιβάλλον του DOS, ακολουθεί πάντοτε ένα ιεραρχικά δομημένο σχεδιασμό και η ροή εκτέλεσης της εξαρτάται αποκλειστικά και μόνο από τις κλήσεις των ρουτινών κώδικα του προγράμματος.

Όμως τα δύο περιβάλλοντα προγραμματισμού, παρά τις διαφορές που παρουσιάζουν στη φιλοσοφία σχεδιασμού μιας εφαρμογής, όπως προδίδει το όνομα τους έχουν και πολλά κοινά χαρακτηριστικά. Το πιο σημαντικό κοινό σημείο τους, είναι ο κώδικας που δημιουργεί ένας προγραμματιστής για τη κατασκευή ενός προγράμματος. Η φιλοσοφία δημιουργίας των ρουτινών ενός προγράμματος παραμένει σταθερή, αφού η ανάπτυξη του κώδικα της Visual Basic ακολουθεί επίσης τις τεχνικές του δομημένου προγραμματισμού. Οι εντολές κώδικα που χρησιμοποιεί η Visual Basic, είναι σχεδόν όλες ίδιες με τις εντολές μίας παραδοσιακής Basic (π.χ QuickBasic) και φυσικά γνώριμες στους περισσότερους προγραμματιστές. Ακόμη η διαχείριση και ο ορισμός των μεταβλητών και των δομών δεδομένων που δημιουργούμε μέσα σε ένα πρόγραμμα, γίνεται με παρόμοιο τρόπο και στα δύο περιβάλλοντα προγραμματισμού. Τέλος αρκετές τεχνικές που εφαρμόζονται κατά την ανάπτυξη των εφαρμογών είναι ίδιες, όπως για παράδειγμα η διαχείριση των αρχείων και οι δομές ελέγχου του προγράμματος.

Στη συνέχεια του κεφαλαίου θα παρουσιάσουμε τη δομή που πρέπει να έχει μια εφαρμογή που σχεδιάζουμε με τη Visual Basic και θα αναφέρουμε τα βασικά σημεία του κώδικα, που πρέπει να έχει σαν εφόδια ένας προγραμματιστής για τη δημιουργία ενός προγράμματος.

3.1. Δομή μίας Visual Basic εφαρμογής

Οι προγραμματιστές που εργάζονται στο περιβάλλον της Visual Basic πρέπει να σχεδιάζουν την εφαρμογή τους, εφαρμόζοντας τις κατάλληλες τεχνικές που θα της δίνουν τη δυνατότητα να εργάζεται μέσα σε ένα παράθυρο των Windows. Κατά την ανάπτυξη μιας εφαρμογής με τη Visual Basic τα πιο σημαντικά στοιχεία επεξεργασίας είναι, η επικοινωνία της με το χρήστη και το περιβάλλον, ο χειρισμός και η αποτελεσματική επεξεργασία των στοιχείων εισόδου της (δεδομένα) και τέλος η μορφοποίηση των στοιχείων εξόδου.

Ένας προγραμματιστής για να επιτύχει την επίλυση ενός προβλήματος με τη Visual Basic, πρέπει να δημιουργήσει μέσα σε μια εφαρμογή τα παρακάτω τμήματα:

- ✓ Το τρόπο επικοινωνίας της εφαρμογής με το περιβάλλον εργασίας και το χρήστη
- ✓ Το χειρισμό των στοιχείων εισόδου (Data Input)
- ✓ Τα τμήματα κώδικα (ρουτίνες προγράμματος), που θα επεξεργαστούν τα στοιχεία εισόδου και θα δημιουργήσουν τα αποτελέσματα της εφαρμογής
- ✓ Τη μορφοποίηση και εξαγωγή των αποτελεσμάτων (Data Output)

Το τμήμα που είναι καινούριο για ένα προγραμματιστή, είναι η διασύνδεση της εφαρμογής με το σύστημα και το χρήστη. Όπως είδαμε, με τη Visual Basic η εργασία αυτή έχει απλοποιηθεί και στη πραγματικότητα θυμίζει τη χρήση ενός σχεδιαστικού προγράμματος. Η δημιουργία της διεπαφής χρήστη δεν ξεκινάει από το μηδέν, αλλά ο προγραμματιστής χρησιμοποιεί τα προκαθορισμένα γραφικά εργαλεία της (controls) και σχεδιάζει πολύ εύκολα και γρήγορα το περιβάλλον εργασίας της εφαρμογής.

Τα τρία τελευταία στοιχεία της εφαρμογής, είναι ήδη γνωστά στους περισσότερους προγραμματιστές από το περιβάλλον του DOS και η δημιουργία τους γίνεται μέσω εντολών κώδικα της Visual Basic. Το μοναδικό σημείο που πρέπει να προσέξει ιδιαίτερα ένας προγραμματιστής της Visual Basic κατά τη δημιουργία του κώδικα του προγράμματος, είναι ότι χρειάζεται να χωρίσει την εφαρμογή σε διακεκριμένα τμήματα, για να μπορεί να εκμεταλλευτεί τις αντιδράσεις των γραφικών αντικειμένων της εφαρμογής στα γεγονότα και τα μηνύματα που δέχεται κάθε χρονική στιγμή το περιβάλλον επικοινωνίας της.

3.2.Τα τμήματα κώδικα

Όπως περιγράψαμε, μία εφαρμογή της Visual Basic αποτελείται από το τρόπο επικοινωνίας του χρήστη με την εφαρμογή και το κώδικα που περιέχει. Τα γραφικά αντικείμενα διασύνδεσης που περιέχει η εφαρμογή, είναι τα στοιχεία που προσδιορίζουν τα αρχικά χαρακτηριστικά του περιβάλλοντος επικοινωνίας της. Η ουσία όμως, κάθε εφαρμογής στο περιβάλλον της Visual Basic είναι πάντοτε ο κώδικας που περιέχει. Οι εντολές κώδικα που περιέχουν οι ρουτίνες μιας εφαρμογής, καθορίζουν τη ροή εκτέλεσής της και προσδιορίζουν ακριβώς τις εργασίες που θα εκτελεί.

Σε μία εφαρμογή της Visual Basic μπορούμε να χρησιμοποιήσουμε μόνο μία φόρμα και όλες οι εντολές κώδικα να αφορούν εργασίες της συγκεκριμένης φόρμας. Σχεδιάζοντας όμως, μία μεγαλύτερη εφαρμογή, για να αποφύγουμε τη δημιουργία ενός πολύπλοκου και δύσχρηστου περιβάλλοντος επικοινωνίας χρήστη-εφαρμογής, σίγουρα θα προσθέσουμε και άλλες φόρμες εργασίας και συνεπώς θα χρειαστούμε περισσότερες εντολές κώδικα για το χειρισμό τους. Επίσης, όσο μεγαλώνει η εφαρμογή εμφανίζονται σημεία, που πρέπει να διαχειριστούν κοινά τμήματα κώδικα και παρουσιάζεται η ανάγκη για γενικές μεταβλητές που θα ανταλλάσσουν τα δεδομένα τους μέσα σε διαφορετικές ρουτίνες της εφαρμογής.

Για το σκοπό αυτό η Visual Basic μας δίνει τη δυνατότητα να δημιουργήσουμε μέσα σε ένα πρόγραμμα ανεξάρτητα τμήματα κώδικα, τα οποία ονομάζονται προγραμματιστικές μονάδες (modules). Ο διαχωρισμός των προγραμματιστικών μονάδων γίνεται με την εμβέλεια και τη χρήση τους μέσα σε μια εφαρμογή. Σε ένα πρόγραμμα είναι δυνατό να υπάρχουν βασικές προγραμματιστικές μονάδες (standard modules) και προγραμματιστικές μονάδες φόρμας (form modules) και κλάσης (class modules). Οι προγραμματιστικές μονάδες φόρμας, στη πραγματικότητα είναι προγραμματιστικές μονάδες κλάσης, αλλά περιέχουν και οπτικά στοιχεία, όπως τις γραφικές περιγραφές του παραθύρου φόρμας και των αντικειμένων που υπάρχουν στην επιφάνειά του. Μια προγραμματιστική μονάδα είναι δυνατό να περιέχει ένα τμήμα δηλώσεων μεταβλητών και ρουτίνες κώδικα.

3.2.1. Τμήματα Δηλώσεων μεταβλητών

Κάθε προγραμματιστική μονάδα (βασική, φόρμας ή κλάσης), περιέχει ένα τμήμα δηλώσεων μεταβλητών (Declarations section). Στο τμήμα αυτό μπορούμε να συμπεριλάβουμε γενικές δηλώσεις μεταβλητών (Declarations). Οι δηλώσεις μεταβλητών είναι μη εκτελέσιμες εντολές κώδικα και τις χρησιμοποιούμε για να αποδώσουμε κάποιο όνομα και να ορίσουμε τα χαρακτηριστικά (π.χ. τύπο δεδομένων) στις εξωτερικές ρουτίνες, στις σταθερές, στις δομές δεδομένων και στα ορίσματα κλήσης DLL αρχείων του προγράμματος.

3.2.2. Ρουτίνες κώδικα

Το μεγάλο πλεονέκτημα του κώδικα της Visual Basic, είναι ότι μπορούμε να διασπάσουμε ένα πρόγραμμα, σε μικρότερα λογικά τμήματα που καλούνται ρουτίνες (procedures). Οι ρουτίνες περιέχουν εντολές κώδικα, που εκτελούν συγκεκριμένες εργασίες οι οποίες επαναλαμβάνονται μέσα σε μια εφαρμογή, όπως αριθμητικούς υπολογισμούς ή την διαχείριση των δεδομένων εισόδου.

Οι ρουτίνες, μας επιτρέπουν να απλοποιήσουμε τις προγραμματιστικές εργασίες, αφού είναι ευκολότερη η τμηματική δημιουργία και πιο αξιόπιστη η αποσφαλμάτωση του προγράμματος. Επίσης είναι δυνατό να τις χρησιμοποιήσουμε αυτούσιες ή με μικρές αλλαγές και σε άλλα προγράμματα με παρόμοιες εργασίες.

Σε ένα πρόγραμμα της Visual Basic, είναι δυνατό να συμπεριλάβουμε τους παρακάτω τύπους ρουτινών :

- ***Sub ρουτίνες***

Οι Sub ρουτίνες, είναι τμήματα κώδικα που εκτελούν συγκεκριμένη εργασία μέσα σε ένα πρόγραμμα. Μια Sub ρουτίνα δεν επιστρέφει κάποια τιμή στο σημείο κλήσης της και ο κώδικάς της ξεκινά πάντοτε με την εντολή Sub και τελειώνει με την εντολή End Sub. Σε μια εφαρμογή είναι δυνατό να έχουμε γενικές ρουτίνες (general procedures) και ρουτίνες γεγονότων (event procedures).

Οι γενικές ρουτίνες, δεν έχουν σχέση με τα γεγονότα που συμβαίνουν στο περιβάλλον εργασίας της εφαρμογής, αλλά αποτελούν ανεξάρτητα τμήματα του κώδικά της. Στις γενικές ρουτίνες συμπεριλαμβάνουμε εργασίες οι οποίες είναι δυνατό να εκτελούνται από διαφορετικά σημεία του προγράμματος. Με τη χρήση των γενικών ρουτινών έχουμε τη δυνατότητα να τοποθετούμε τις κοινές εντολές κώδικα σε ένα σημείο, αποφεύγοντας την επανάληψή τους σε διαφορετικά σημεία του προγράμματος.

Οι ρουτίνες γεγονότων, περιέχουν εντολές κώδικα οι οποίες εκτελούνται αυτόματα, μόλις ένα γραφικό αντικείμενο της διεπαφής χρήστη αναγνωρίσει ένα γεγονός που το αφορά μέσα στο περιβάλλον εργασίας. Οι ρουτίνες γεγονότων δεν αποτελούν ανεξάρτητο τμήμα κώδικα, αλλά ανήκουν απαραίτητα στο κώδικα που περιέχει μία προγραμματιστική μονάδα φόρμας της εφαρμογής.

- ***Ρουτίνες συνάρτησης***

Η Visual Basic μας παρέχει ένα αριθμό έτοιμων ρουτινών συνάρτησης (Function procedures), που μπορούμε να χρησιμοποιήσουμε μέσα σε ένα πρόγραμμά μας, όπως τις συναρτήσεις Sqr ή Chr. Ένα από τα πλεονέκτημα της όμως, είναι ότι μας δίνει τη δυνατότητα να χρησιμοποιήσουμε επιπλέον ρουτίνες συνάρτησης που θα προέρχονται από άλλες εφαρμογές ή θα τις δημιουργήσουμε μόνοι μας σύμφωνα με τις ανάγκες των προγραμμάτων μας.

Ρουτίνες ιδιοτήτων

Οι ρουτίνες ιδιοτήτων (property procedures), χρησιμοποιούνται για τη δημιουργία και το χειρισμό νέων εξειδικευμένων ιδιοτήτων. Όταν δημιουργούμε μια ιδιότητα, αυτή είναι χαρακτηριστικό της προγραμματιστικής μονάδας που περιέχει τη ρουτίνα δημιουργίας της. Με τις ρουτίνες ιδιοτήτων μπορούμε να δημιουργήσουμε ιδιότητες, για κάθε τύπο προγραμματιστικής μονάδας. Στις ιδιότητες αυτές, ο χρήστης θα έχει δικαίωμα μόνο ανάγνωσης (read-only) και έχουμε τη δυνατότητα να ορίσουμε εντολές κώδικα που θα εκτελούνται αυτόματα μόλις οριστεί η τιμή της ιδιότητας.

3.3. Σταθερές

Σε πολλά σημεία ενός προγράμματος της Visual Basic, είναι δυνατό να συναντάμε μια σταθερή τιμή η οποία θα επαναλαμβάνεται συνεχώς. Αρκετές φορές οι τιμές αυτές είναι πολύπλοκες για να απομνημονευτούν ή δε δείχνουν αντιπροσωπευτικά τη σημασία τους μέσα στην εφαρμογή. Για το σκοπό αυτό η Visual Basic επιτρέπει στον προγραμματιστή τη χρήση συμβολικών Σταθερών (Constants) μέσα σε ένα πρόγραμμά της.

Μια συμβολική σταθερά πρέπει να έχει πάντοτε ένα αντιπροσωπευτικό όνομα για να είναι εύκολα αναγνώσιμος ο κώδικας του προγράμματος. Η τιμή μιας συμβολικής σταθεράς δε μπορεί να αλλάξει κατά την εκτέλεση ενός προγράμματος.

Μέσα σε ένα πρόγραμμα της Visual Basic, μπορούμε να χρησιμοποιήσουμε έτοιμες συμβολικές σταθερές του συστήματος που παρέχονται από τη Visual Basic ή άλλες εφαρμογές και εργαλεία. Οι συμβολικές σταθερές συστήματος, βρίσκονται στις βιβλιοθήκες αντικειμένων του παράθυρου Επισκόπησης Αντικειμένων (Object Browser) και μπορούμε εφόσον είναι ενεργό το παράθυρο κώδικα να τις επικολλήσουμε και να τις χρησιμοποιήσουμε με το πλήκτρο Paste, σε οποιοδήποτε τμήμα του προγράμματός μας.

Ακόμη η Visual Basic μας δίνει τη δυνατότητα να δημιουργήσουμε τις δικές μας συμβολικές σταθερές, σύμφωνα με τις ανάγκες του προγράμματός μας. Αν θέλουμε να ορίσουμε μόνοι μας μία συμβολική σταθερά, πρέπει να χρησιμοποιήσουμε την εντολή Const:

[Private¦Public]Const όνομα_σταθεράς[As τύπος_δεδομένων]=έκφραση

Στον τύπο ορισμού μιας συμβολικής σταθεράς η τιμή της έκφρασης μπορεί να είναι αριθμητική, αλφαριθμητική ή τύπου Ημερομηνίας:

```
Const PI = 3.1415
Const Dealer = "Αποστόλου"
Const HireDate = #1/3/92#
```

Η εμβέλεια μιας συμβολικής σταθεράς μέσα σε ένα πρόγραμμα της Visual Basic καθορίζεται από τον τύπο της εντολής και το τμήμα κώδικα στο οποίο δηλώθηκε:

- ✓ Για να ορίσουμε μια τοπική συμβολική σταθερά, απλά χρησιμοποιούμε την εντολή Const μέσα σε μια τοπική ρουτίνα.
- ✓ Για να ορίσουμε μια συμβολική σταθερά επιπέδου προγραμματιστικής μονάδας, ορίζουμε τη σταθερά με την εντολή Const στο τμήμα δηλώσεων (Declarations section) της προγραμματιστικής μονάδας.
- ✓ Τέλος για να ορίσουμε μια καθολική συμβολική σταθερά, στην τιμή της οποίας θα έχουν πρόσβαση όλα τα τμήματα κώδικα της εφαρμογής, χρησιμοποιούμε την εντολή Const με το δηλωτικό Public και ορίζουμε τη σταθερά στο τμήμα δηλώσεων (Declarations section) μιας βασικής προγραμματιστικής μονάδας (standard module):

```
Public Const Pi=3.14
```

Οι σταθερές που εμφανίζονται στο παράθυρο Επισκόπησης Αντικειμένων, δε χρειάζεται να οριστούν πάλι στο πρόγραμμά μας με την εντολή Const.

3.4. Μεταβλητές

Οι μεταβλητές είναι θέσεις μνήμης του συστήματος στις οποίες αποθηκεύονται προσωρινά δεδομένα κατά την εκτέλεση μίας εφαρμογής. Στη Visual Basic κάθε μεταβλητή χαρακτηρίζεται από το όνομά της και τον τύπο δεδομένων που υποστηρίζει. Το όνομα μίας μεταβλητής πρέπει να ξεκινά πάντοτε με αλφαβητικό στοιχείο, δεν πρέπει να είναι κάποια από της δεσμευμένες λέξεις της Visual Basic (π.χ Sub ή End) και το μήκος του δεν πρέπει να ξεπερνά τους 255 χαρακτήρες. Σε μια εφαρμογή της Visual Basic είναι δυνατό να έχουμε Αλφαριθμητικές μεταβλητές (String) με περιεχόμενα γράμματα και αριθμούς, Αριθμητικές μεταβλητές (Numeric) με περιεχόμενα αριθμητικές τιμές, Λογικές μεταβλητές (Logical) οι οποίες μπορούν να περιέχουν τις τιμές Αληθής (True) ή Ψευδής (False), μεταβλητές Ημερομηνίας (Date) οι οποίες μπορούν να περιέχουν δεδομένα ημερομηνίας ή ώρας και τέλος μεταβλητές Αντικειμένων (Object) οι οποίες αναφέρονται σε αντικείμενα της εφαρμογής.

Όταν δηλώνουμε μια μεταβλητή μέσα σε ένα πρόγραμμα της Visual Basic έχουμε τη δυνατότητα να ορίσουμε τον τύπο των δεδομένων (Πίνακας 1), που πρόκειται να αποθηκεύσει, όταν θα εκτελείται η εφαρμογή. Με τον τρόπο αυτό μπορούμε να επιτύχουμε καλύτερη διαχείριση της μνήμης του συστήματος και συνεπώς τις βέλτιστες επιδόσεις για τις εφαρμογές που δημιουργούμε.

Τύπος μεταβλητής	Αρ.bytes	Εύρος τιμών
Ακέραιος (Integer)	2	-32768 έως +32767
Μεγάλος Ακέραιος (Long)	4	-2,147,483,648 έως 2,147,483,647
Απλής Ακρίβειας (Single)	4	(αρνητικές τιμές) -3.402823E38 έως -1.401298E-45 (θετικές τιμές) 1.401298E-45 έως 3.402823E38
Διπλής Ακρίβειας (Double)	8	(αρνητικές τιμές) -1.79769313486231E308 έως -4.94065645851247E-324 (θετικές τιμές) 4.94065645851247E-324 έως1.79769313486231E308
Νομίσματος (Currency)	8	-9.22337203685477.5808 έως 9.22337203685477.5807
Αλφαριθμητική (String)	1byte/ χαρακτ.	0 έως 65000 χαρακτήρες 0 έως 2E32 σε 32-bit συστήματα
Ψηφιολέξη (Byte)	1	0 έως 255
Λογική (Boolean)	2	True ή False
Ημερομηνίας (Date)	8	January 1, 100 to December 31, 9999
Αντικειμένου (Object)	4	Οποιαδήποτε αναφορά αντικειμένου

Πίνακας 1: Οι τύποι μεταβλητών της Visual Basic.

Εκτός από τους παραπάνω τύπους μεταβλητών, η Visual Basic υποστηρίζει ακόμη και ένα ειδικό τύπο μεταβλητής τη Variant. Σε μία μεταβλητή τύπου Variant, μπορούμε να αποθηκεύσουμε διαφορετικούς τύπους δεδομένων.

Το μεγαλύτερο πλεονέκτημα μιας μεταβλητής τύπου Variant είναι ότι, μας επιτρέπει να κάνουμε πράξεις με δεδομένα διαφορετικού τύπου (format), χωρίς καμία μετατροπή. Μια Variant μεταβλητή δεσμεύει 16 bytes για αριθμητικές και 1 byte ανά χαρακτήρα για αλφαριθμητικές τιμές. Η Visual Basic όταν συναντά δεδομένα διαφορετικού τύπου, κάνει αυτόματη μετατροπή και εκτελεί την πράξη. Φυσικά όταν κάνουμε πράξεις με μία Variant μεταβλητή πρέπει να είμαστε ιδιαίτερα προσεχτικοί, γιατί οι πράξεις ορίζονται διαφορετικά για κάθε τύπο δεδομένων που χρησιμοποιούμε. Για παράδειγμα ο χαρακτήρας της πρόσθεσης (+), σε αλφαριθμητικές μεταβλητές σημαίνει ένωση των μεταβλητών, ενώ σε αριθμητικές σημαίνει το άθροισμά τους.

Αν μέσα σε ένα πρόγραμμα δεν ορίσουμε τον τύπο μίας μεταβλητής, η Visual Basic την ορίζει αυτόματα σαν τύπου Variant. Επειδή όμως ο συγκεκριμένος τύπος μεταβλητής δεσμεύει αρκετή μνήμη του συστήματος, είναι προτιμότερο αν θέλουμε να έχουμε ευέλικτες και γρήγορες εφαρμογές, να ορίζουμε τον τύπο κάθε μεταβλητής ανάλογα με τα δεδομένα που πρόκειται να αποθηκεύει. Για παράδειγμα, αν είμαστε σίγουροι ότι μια μεταβλητή σε όλη τη διάρκεια εκτέλεσης της εφαρμογής θα περιέχει μικρές αριθμητικές τιμές, είναι σκόπιμο να την ορίσουμε σαν τύπου Integer οπότε θα δεσμεύει κάθε φορά μόνο δύο bytes μνήμης.

Αν θέλουμε υποχρεωτικά να ορίζουμε κάθε μεταβλητή και τον τύπο δεδομένων της, πριν τη χρησιμοποιήσουμε σε μια προγραμματιστική μονάδα της Visual Basic πρέπει να συμπεριλάβουμε στο κώδικά της την εντολή Option Explicit. Η εντολή Option Explicit, πρέπει να τοποθετηθεί στο τμήμα δηλώσεων της προγραμματιστικής μονάδας, πριν από οποιαδήποτε δηλωτική εντολή μεταβλητής ή συμβολικής σταθεράς. Όταν έχουμε συμπεριλάβει στο κώδικα της προγραμματιστικής μονάδας την εντολή Option Explicit, αν προσπαθήσουμε να χρησιμοποιήσουμε μια μεταβλητή χωρίς πρώτα να την έχουμε δηλώσει θα εμφανιστεί μήνυμα λάθους. Με τη τεχνική αυτή, ανιχνεύουμε τυχόν λάθη στη πληκτρολόγηση των ονομάτων των μεταβλητών κατά τη συγγραφή του κώδικα.

3.5. Ορισμός μεταβλητών

Σε μία εφαρμογή της Visual Basic έχουμε τη δυνατότητα να δημιουργήσουμε μεταβλητές με διαφορετική εμβέλεια (scope). Η εμβέλεια μιας μεταβλητής, ορίζει ποια σημεία της εφαρμογής θα έχουν τη δυνατότητα πρόσβασης στα περιεχόμενα της. Ο ορισμός μιας μεταβλητής στη Visual Basic εξαρτάται κάθε φορά από την εμβέλεια που πρόκειται να έχει μέσα στην εφαρμογή που θα χρησιμοποιηθεί αλλά και από τον τύπο των δεδομένων που πρόκειται να αποθηκευτούν σε αυτή.

Στη Visual Basic η δήλωση μιας μεταβλητής γίνεται με δηλωτικές εντολές της μορφής:

Εντολή 'Ονομα_μεταβλητής As τύπος_δεδομένων.

Η δηλωτική Εντολή που χρησιμοποιούμε κάθε φορά, εξαρτάται από το σημείο κώδικα στο οποίο θα δηλώσουμε τη μεταβλητή και καθορίζει την εμβέλεια της μέσα στην εφαρμογή. Ο τύπος δεδομένων μπορεί είναι κάποια από τις τιμές του Πίνακα 1. της προηγούμενης παραγράφου ή η τιμή Variant.

Οι μεταβλητές της Visual Basic χωρίζονται ανάλογα με την εμβέλεια τους μέσα στην εφαρμογή σε μεταβλητές τοπικής εμβέλειας, προγραμματιστικής μονάδας και καθολικού επιπέδου.

⇨ *Μεταβλητές τοπικής εμβέλειας*

Μία μεταβλητή τοπικής εμβέλειας(local variable), περιέχει δεδομένα τα οποία είναι ορατά μόνο μέσα

από τη διαδικασία (ρουτίνα ή συνάρτηση) στην οποία έχει δηλωθεί. Η διάρκεια της τοπικής μεταβλητής εξαρτάται από το χρονικό διάστημα το οποίο είναι ανοικτή η διαδικασία. Η ζωή της τοπικής μεταβλητής ξεκινά όταν αρχίζει η εκτέλεση της διαδικασίας και τελειώνει μόλις ολοκληρωθεί η εκτέλεση. Σε δύο διαφορετικές διαδικασίες έχουμε τη δυνατότητα να χρησιμοποιούμε διαφορετικές τοπικές μεταβλητές με το ίδιο όνομα. Αυτό δεν επηρεάζει καθόλου την εφαρμογή αφού κάθε φορά που ολοκληρώνουμε την εκτέλεση μίας διαδικασίας η τιμή των τοπικών της μεταβλητών μηδενίζεται και η θέση μνήμης που καταλαμβάνουν στο σύστημα απελευθερώνεται.

Αν θέλουμε να ορίσουμε μέσα σε μία διαδικασία μία μεταβλητή σαν μεταβλητή τοπικής εμβέλειας χρησιμοποιούμε τη δήλωση Dim. Με μία δήλωση Dim μπορούμε να ορίσουμε μια ή περισσότερες μεταβλητές:

```
Dim Total As Long
Dim Test As Integer, Name As String
```

Η τιμή μιας τοπικής μεταβλητής όπως ήδη αναφέραμε, χάνεται μόλις κλείσουμε τη διαδικασία μέσα στην οποία την ορίσαμε. Αν θέλουμε να διατηρήσουμε την τιμή μίας τοπικής μεταβλητής μεταξύ διαδοχικών κλήσεων μιας ρουτίνας, πρέπει να την ορίσουμε με τον ειδικό τύπο Static ως εξής:

```
Static Flag As Integer
```

Οι μεταβλητές τύπου Static έχουν το πλεονέκτημα ότι διατηρούν την τιμή τους σε όλη τη διάρκεια εκτέλεσης της εφαρμογής, ακόμη και αν κλείσει η διαδικασία που τις χρησιμοποιεί.

Μεταβλητές προγραμματιστικής μονάδας

Μία προγραμματιστική μονάδα περιγράψαμε, αποτελεί ανεξάρτητο τμήμα κώδικα του μια κώδικα μιας εφαρμογής. Κάθε μεταβλητή που δηλώνουμε μέσα στο τμήμα δηλώσεων μίας προγραμματιστικής μονάδας (module-level variable), περιέχει δεδομένα που είναι προσπελάσιμα μόνο από τα τμήματα κώδικα της συγκεκριμένης προγραμματιστικής μονάδας. Η διάρκεια ζωής των μεταβλητών της προγραμματιστικής μονάδας διαρκεί για όλο το διάστημα της εκτέλεσης της εφαρμογής, ακόμη και αν σταματήσει να εκτελείται ο κώδικας της συγκεκριμένης προγραμματιστικής μονάδας ή αν κατεβάσουμε τη φόρμα από τη μνήμη του συστήματος.

Για να ορίσουμε μία μεταβλητή επιπέδου προγραμματιστικής μονάδας, χρησιμοποιούμε μια από τις δηλωτικές εντολές Dim ή Private. Η δήλωση της μεταβλητής όμως, πρέπει να γίνει απαραίτητα στο τμήμα δηλώσεων (Declarations section) της προγραμματιστικής μονάδας :

```
Private Counter As Integer
```

ή

```
Dim Counter As Integer
```

Για τον ορισμό των μεταβλητών επιπέδου προγραμματιστικής μονάδας, προτείνεται να χρησιμοποιούμε την δηλωτική εντολή Private, για να είναι ευδιάκριτος ο διαχωρισμός των μεταβλητών προγραμματιστικής μονάδας από τις μεταβλητές καθολικής εμβέλειας που θα γνωρίσουμε στη συνέχεια.

Μεταβλητές καθολικής εμβέλειας

Αποτελούν μέσα σε ένα πρόγραμμα της Visual Basic τις μεταβλητές με τη μεγαλύτερη εμβέλεια. Στα δεδομένα μίας καθολικής μεταβλητής (public variable), έχουν πρόσβαση όλα τα τμήματα κώδικα που περιέχει η εφαρμογή.

Η δήλωση μίας μεταβλητής καθολικής εμβέλειας γίνεται στο τμήμα δηλώσεων (Declarations section), οποιασδήποτε προγραμματιστικής μονάδας (επιπέδου κώδικα ή φόρμας) με τη δηλωτική εντολή Public:

```
Public LastName As String
```

3.6. Δομές Απόφασης ή Επιλογής

Τις δομές απόφασης (Decision structures), ή δομές επιλογής μπορούμε να τις περιγράψουμε σαν τις μεθόδους λήψης αποφάσεων μέσα στον κώδικα της Visual Basic. Σε μία εφαρμογή της Visual Basic μία ρουτίνα μπορεί να αποτελείται από διαφορετικές εντολές κώδικα. Ο προγραμματιστής έχει τη δυνατότητα να συμπεριλάβει μία συνθήκη μέσα στην ρουτίνα, η οποία κάθε φορά θα καθορίζει ανάλογα με τα αποτελέσματα της ποιες από τις εντολές κώδικα θα εκτελούνται και ποιες θα αγνοούνται. Μία συνθήκη συνήθως είναι μία σύγκριση αριθμητικών ή αλφαριθμητικών τιμών, αλλά μπορεί να είναι και μια επαλήθευση των λογικών τιμών Αληθής ή Ψευδής.

Η Visual Basic υποστηρίζει τους παρακάτω τύπους δομών απόφασης :

- ✓ **If...Then**
- ✓ **If...Then...Else**
- ✓ **Select Case**

Η σύνταξη των δομών απόφασης της Visual Basic είναι ίδια με την σύνταξη των αντίστοιχων δομών απόφασης της QuickBasic.

3.7. Δομές Ανακύκλωσης

Οι δομές ανακύκλωσης (Loops), επιτρέπουν μέσα από την ικανοποίηση μίας συνθήκης την επαναληπτική εκτέλεση μίας ομάδας εντολών κώδικα. Η Visual Basic υποστηρίζει τους παρακάτω τύπους δομών ανακύκλωσης:

- ✓ **For...Next**
- ✓ **For Each...Next**
- ✓ **Do...Loop**

Η σύνταξη των δομών ανακύκλωσης της Visual Basic είναι ίδια με την σύνταξη των αντίστοιχων δομών ανακύκλωσης της QuickBasic.

4. ΤΕΧΝΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Όπως αναφέραμε, η ανάπτυξη μιας εφαρμογής στο περιβάλλον προγραμματισμού της Visual Basic γίνεται με μεγάλη ευκολία και ταχύτητα. Η ευκολία προγραμματισμού της εφαρμογής, στηρίζεται στην απλότητα σχεδίασης του γραφικού μέσου διασύνδεσης χρήστη-εφαρμογής (graphical user interface) και στον τρόπο δημιουργίας του κώδικα των προγραμμάτων με τη χρήση των γνωστών εντολών της Basic του DOS. Αρκετοί από εσάς σκέφτηκαν ίσως, ότι αφού η δημιουργία μιας εφαρμογής με τη Visual Basic γίνεται τόσο απλά και γρήγορα, άρα το περιβάλλον της δεν θα προσφέρεται για την ανάπτυξη επαγγελματικών εφαρμογών.

Η Visual Basic όμως θα σας διαψεύσει πολύ σύντομα. Μια εφαρμογή που δημιουργείται μέσα στο γραφικό περιβάλλον προγραμματισμού της, έχει τη δυνατότητα υποστήριξης ανεπτυγμένων τεχνικών προγραμματισμού, όπως χρήση μενού επιλογών, διαχείριση λαθών (error handling), υποστήριξη γραφικών (graphics), επικοινωνία με άλλες εφαρμογές (μηχανισμοί DDE και OLE), κ.α. Αλλά το γεγονός που σίγουρα θα σας εντυπωσιάσει είναι ότι όσο μεγάλη και πολύπλοκη και αν είναι μια εφαρμογή της Visual Basic, το περιβάλλον εργασίας της διατηρεί τη λειτουργικότητά του και τη φιλικότητα επικοινωνίας με το χρήστη.

4.1. Μενού επιλογών

Όλοι μας θα έχουμε σίγουρα χρησιμοποιήσει μέσα από τις εφαρμογές μας κάποιο μενού επιλογών (menu). Μέσα από τις επιλογές ενός μενού, έχουμε τη δυνατότητα κατά την εκτέλεση μιας εφαρμογής, να παρέχουμε στους χρήστες μια εύκολα προσπελάσιμη λίστα εργασιών. Σε μια εφαρμογή της Visual Basic, μπορούμε να δημιουργήσουμε πολύ εύκολα μενού επιλογών και να εισάγουμε σε κάθε επιλογή του εντολές κώδικα για την εκτέλεση συγκεκριμένων εργασιών.

Σε κάθε φόρμα που περιέχει μια εφαρμογή της Visual Basic, μπορούμε να δημιουργήσουμε ένα μενού επιλογών. Τα μενού επιλογών που υποστηρίζει η Visual Basic, συνδυάζουν όλα τα πλεονεκτήματα των μενού επιλογών που έχουμε συναντήσει σε άλλες Windows-based εφαρμογές, όπως δυνατότητα υποστήριξης υπομενού επιλογών, πλήκτρων άμεσης πρόσβασης (access keys), πλήκτρων συντομίας (shortcut keys), κ.α.

Το εργαλείο που χρησιμοποιούμε για να σχεδιάσουμε ένα μενού επιλογών, είναι ο Επεξεργαστής Μενού (Menu Editor) της Visual Basic (Σχήμα 10). Για να εμφανίσουμε τον Επεξεργαστή Μενού, επιλέγουμε από το μενού Tools την επιλογή Menu Editor ή το πλήκτρο Menu Editor της γραμμής εργαλείων.

Σχ.10 Ο Επεξεργαστής Μενού της Visual Basic.

Μέσα από τον Επεξεργαστή Μενού δημιουργούμε τις επιλογές ενός μενού, οι οποίες ονομάζονται αντικείμενα μενού (menu items). Κάθε επιλογή που εισάγουμε σε ένα μενού της Visual Basic, αποτελεί ανεξάρτητο αντικείμενο της εφαρμογής και χαρακτηρίζεται όπως και τα υπόλοιπα αντικείμενα της Visual Basic, από ένα προκαθορισμένο αριθμό ιδιοτήτων και επιπλέον το γεγονός click. Τις τιμές των ιδιοτήτων ενός αντικειμένου μενού, μπορούμε να τις μετατρέψουμε κατά το σχεδιασμό της εφαρμογής μέσω του Επεξεργαστή Μενού ή μέσω του παραθύρου ιδιοτήτων . Οι εργασίες που πρέπει να εκτελούν οι επιλογές ενός μενού, ορίζονται μέσα από εντολές κώδικα που θα συμπεριλάβουμε στις ρουτίνες γεγονότων click κάθε αντικειμένου μενού.

Ο Επεξεργαστής Μενού αποτελείται από δύο τμήματα. Στο επάνω τμήμα του, παρατηρούμε ότι εμφανίζεται μια ομάδα ιδιοτήτων, που ονομάζονται ιδιότητες εργαλείου μενού (menu control properties). Μέσω των ιδιοτήτων κάθε εργαλείου, καθορίζουμε τα χαρακτηριστικά εμφάνισης και την αρχική συμπεριφορά των επιλογών του μενού. Οι δύο πιο βασικές ιδιότητες μιας επιλογής ενός μενού, είναι η Name που δηλώνει το όνομα και η Caption που περιγράφει την επικεφαλίδα της επιλογής. Η ιδιότητα Name είναι το όνομα με το οποίο διαχειριζόμαστε το αντικείμενο μενού μέσω εντολών κώδικα της Visual Basic και η ιδιότητα Caption είναι το λεκτικό με το οποίο θα εμφανίζεται η επιλογή επάνω στο μενού κατά την εκτέλεση της εφαρμογής.

Το κάτω τμήμα του Επεξεργαστή Μενού, καλείται πλαίσιο λίστας εργαλείων μενού (menu control list box)

και κατά το σχεδιασμό ενός μενού περιέχει όλες τις επιλογές που έχουμε ήδη δημιουργήσει και το επίπεδό τους.

4.2. Πλαίσια διαλόγου

Κατά την εκτέλεση μιας εφαρμογής χρειάζεται πολλές φορές ο χρήστης να ανταλλάσσει μηνύματα με την εφαρμογή. Για να επιτύχουμε την επικοινωνία του χρήστη με την εφαρμογή, έχουμε τη δυνατότητα να δημιουργήσουμε με τη Visual Basic πλαίσια διαλόγου (dialog boxes). Τα πλαίσια διαλόγου μπορούμε να τα χρησιμοποιήσουμε για να εισάγει ο χρήστης κάποια στοιχεία στην εφαρμογή, αλλά και προς την αντίθετη κατεύθυνση για την εμφάνιση μηνυμάτων της εφαρμογής προς το χρήστη.

Σε μια εφαρμογή της Visual Basic, υπάρχουν τρεις τρόποι να προσθέσουμε πλαίσια διαλόγου:

- με τη δημιουργία καινούριων πλαισίων σύμφωνα με τις ανάγκες μιας εφαρμογής, τοποθετώντας εργαλεία επάνω σε μια φόρμα ή εισάγοντας και μορφοποιώντας στην εφαρμογή κάποιο υπάρχον πλαίσιο διαλόγου.
- με τη χρήση συγκεκριμένων πλαισίων διαλόγου (standard dialog boxes), μέσω του εργαλείου Common dialog ActiveX control της Visual Basic, όπως το πλαίσιο αποθήκευσης αρχείων.
- με τη χρήση προκαθορισμένων πλαισίων διαλόγου μέσω των συναρτήσεων Input() και MsgBox().

Από τους παραπάνω τύπους πλαισίων διαλόγου, καινούριο στοιχείο αποτελούν τα προκαθορισμένα πλαίσια της Visual Basic. Η Visual Basic υποστηρίζει δύο συναρτήσεις τη **MsgBox()** και την **InputBox()** για τη δημιουργία προκαθορισμένων πλαισίων διαλόγου. Χρησιμοποιώντας τις δύο παραπάνω συναρτήσεις δε χρειάζεται να σχεδιάσουμε από την αρχή πλαίσια διαλόγου αλλά μπορούμε να εμφανίσουμε με τις δικές μας παραμέτρους τα εσωτερικά πλαίσια της Visual Basic. Ωστόσο με τη χρήση των δύο αυτών συναρτήσεων, δεν έχουμε ιδιαίτερες δυνατότητες μορφοποίησης της εμφάνισης των πλαισίων διαλόγου.

Τα πλαίσια διαλόγου της Visual Basic χωρίζονται σε δύο κατηγορίες:

- Τα υποχρεωτικά (modal), τα οποία παραμένουν τα ενεργά αντικείμενα της εφαρμογής μέχρι να τα κλείσει ο χρήστης. Με την τεχνική αυτή είναι σίγουρο ότι ο χρήστης θα δει το μήνυμα του πλαισίου, αφού είναι υποχρεωμένος πρώτα να το κλείσει για να προχωρήσει στη συνέχεια της εφαρμογής.
- Τα μη υποχρεωτικά (modeless), τα οποία δεν είναι απαραίτητο να κλείσει ο χρήστης για να συνεχίσει την εκτέλεση της εφαρμογής. Όταν εμφανίζεται ένα μη υποχρεωτικό πλαίσιο διαλόγου, μπορούμε να φέρουμε στο προσκήνιο της εφαρμογής κάποια άλλη φόρμα. Τα μη υποχρεωτικά πλαίσια διαλόγου δεν εμφανίζονται συχνά μέσα σε εφαρμογές και χρησιμοποιούνται κυρίως για την εμφάνιση βοηθητικών μηνυμάτων προς το χρήστη και όχι για την εκτέλεση κρίσιμων εργασιών της εφαρμογής.

4.3. Εκτυπώσεις

Μια εφαρμογή της Visual Basic μας δίνει τη δυνατότητα να εκμεταλλευτούμε όλα τα πλεονεκτήματα εκτύπωσης που προσφέρει το περιβάλλον γραφικών των Windows, όπως συνδυασμένη εκτύπωση γραφικών και κειμένου, χρήση Truetype γραμματοσειρών, επιλογή εκτυπωτή, κ.α.

Δυστυχώς όμως, η εργασία των εκτυπώσεων μέσα από το περιβάλλον των Windows δεν είναι ιδιαίτερα φιλική, αφού το αποτέλεσμα μιας εκτύπωσης εξαρτάται από πολλές παραμέτρους. Σε μια διαδικασία εκτύπωσης είναι δυνατό να μην επιτύχουμε καλή ποιότητα, όχι λόγω προβλημάτων της εφαρμογής αλλά από την επίδραση εξωτερικών παραγόντων όπως η χρήση διαφορετικού οδηγού εκτυπωτή (printer driver) ή ενός εκτυπωτή με περιορισμένες δυνατότητες. Τέλος η ποιότητα εκτύπωσης μέσα από μια εφαρμογή της Visual Basic, είναι συνάρτηση και της μεθόδου που εκτελείς την εκτύπωση.

Για να εκτυπώσουμε δεδομένα, μέσα από μια εφαρμογή της Visual Basic σε ένα εκτυπωτή, χρησιμοποιούμε τρεις τεχνικές:

- ✓ Δημιουργούμε την αναφορά εκτύπωσης επάνω σε μια φόρμα εργασίας και την τυπώνουμε με τη μέθοδο PrintForm.
- ✓ Στέλνουμε τα στοιχεία εκτύπωσης απ ευθείας στον εκτυπωτή, θέτοντας τον εξ ορισμού εκτυπωτή σαν στοιχείο του αντικειμένου συλλογής εκτυπωτών (Printers Collection).
- ✓ Στέλνουμε τα στοιχεία εκτύπωσης στο ειδικό αντικείμενο Printer της Visual Basic με τη μέθοδο Print και τα εκτυπώνουμε με τις μεθόδους NewPage και EndDoc που υποστηρίζει.

4.4. Επικοινωνία με το Clipboard

Μια εφαρμογή της Visual Basic, έχει τη δυνατότητα εκμετάλλευσης των εργαλείων του λειτουργικού συστήματος, όπως το Clipboard των Windows. Το Clipboard είναι ένα προσωρινό μέσο αποθήκευσης, που χρησιμοποιείται για τη μεταφορά γραφικών και κειμένου μεταξύ εφαρμογών. Στο χώρο μνήμης του Clipboard, έχουν πρόσβαση όλες οι εφαρμογές των Windows.

Οι περισσότεροι χρήστες των Windows έχουν χρησιμοποιήσει το χώρο μνήμης του Clipboard, ακόμη και αν δε το έχουν ανοίξει ποτέ. Κάθε φορά που αντιγράφουν ένα κομμάτι κειμένου σε κάποιο από τους γνωστούς επεξεργαστές κειμένου των Windows, τα δεδομένα περνάνε απαραίτητα μέσα από το χώρο μνήμης του Clipboard.

Για να ανταλλάξουμε δεδομένα μεταξύ μιας εφαρμογής και του Clipboard, χρησιμοποιούμε το ειδικό αντικείμενο Clipboard που υποστηρίζει η Visual Basic. Το ειδικό αντικείμενο Clipboard, χρησιμοποιείται για το χειρισμό κειμένου και γραφικών μέσα στο χώρο μνήμης του Clipboard. Η ανταλλαγή των δεδομένων, γίνεται μέσω των μεθόδων SetText και GetText όταν πρόκειται για χαρακτήρες κειμένου ή των μεθόδων SetData και GetData όταν πρόκειται για γραφικά.

Στη συνέχεια, θα παρουσιάσουμε το τρόπο με τον οποίο αντιγράφουμε ένα τμήμα κειμένου από ένα πλαίσιο κειμένου μιας εφαρμογής προς το Clipboard και το τρόπο με τον οποίο διαβάζουμε τα περιεχόμενά του. Πριν όμως, αντιγράψουμε ένα τμήμα κειμένου από το πλαίσιο κειμένου, πρέπει πρώτα να το επιλέξουμε. Ο πιο απλός τρόπος για να επιλέξουμε ένα τμήμα κειμένου, είναι να τοποθετήσουμε το δρομέα μπροστά από το πρώτο χαρακτήρα, να κρατήσουμε το αριστερό πλήκτρο του ποντικιού συνεχώς πατημένο και να μεταφερθούμε στο τελευταίο χαρακτήρα του τμήματος κειμένου. Το επιλεγμένο κείμενο, μπορούμε να το χειριστούμε μέσω της ιδιότητας SelText του πλαισίου κειμένου. Μια ακόμη χρήσιμη ιδιότητα των πλαισίων κειμένου για το χειρισμό κειμένου, είναι η SelLength που δηλώνει το μήκος του επιλεγμένου κειμένου. Όταν δεν έχουμε επιλέξει κανένα τμήμα του κειμένου, η τιμή της SelLength είναι ίση με το μηδέν .

Για να αντιγράψουμε το επιλεγμένο κείμενο από το πλαίσιο κειμένου TxtEntry στο Clipboard, πρέπει να χρησιμοποιήσουμε τη μέθοδο SetText του ειδικού αντικειμένου Clipboard:

```
Clipboard.SetText TxtEd.it.SelText
```

Η επαναφορά του επιλεγμένου κειμένου από το Clipboard στην εφαρμογή, μπορεί να γίνει μέσω της μεθόδου GetText του ειδικού αντικειμένου Clipboard:

```
.TxtEdit.SelText = Clipboard.GetText
```

Όταν χρησιμοποιούμε γραφικά αντί για κείμενο οι τεχνικές ανταλλαγής δεδομένων μεταξύ του Clipboard και της εφαρμογής, είναι ίδιες. Το μόνο που αλλάζει είναι οι μέθοδοι (GetData, SetData) που θα χρησιμοποιήσουμε.

Πριν αντιγράψουμε το επιλεγμένο κείμενο, έχουμε τη δυνατότητα να καθαρίσουμε τα περιεχόμενα του Clipboard με την εντολή :

```
Clipboard.Clear
```

Τα περιεχόμενα του Clipboard, πρέπει να τα καθαρίζουμε μετά από την αντιγραφή μεγάλου όγκου δεδομένων, για να μη δεσμεύουν χώρο στη μνήμη του συστήματος.

5. ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Στις εφαρμογές της Visual Basic μπορούν να χρησιμοποιηθούν έτοιμα δεδομένα που προέρχονται από συστήματα διαχείρισης βάσεων δεδομένων. Η Visual Basic παρέχει δυνατότητα πρόσβασης σε αρχεία βάσεων δεδομένων διαφορετικών μορφοποιήσεων. Επιτυγχάνεται πρόσβαση σε αρχεία δεδομένων των Access, Excel, Btrieve, dBase, FoxPro, Paradox κ.ά. Όλες αυτές οι βάσεις δεδομένων, από τις οποίες μπορεί να αντλήσει στοιχεία η Visual Basic, ονομάζονται εξωτερικές βάσεις δεδομένων.

5.1. Εργαλείο δεδομένων

Για να μπορέσουμε να επιτύχουμε πρόσβαση της εφαρμογής μας σε μία βάση δεδομένων, θα πρέπει να χρησιμοποιήσουμε το εργαλείο δεδομένων (data control). (σχήμα 11).

Σχ.11 Το εργαλείο δεδομένων.

Χρησιμοποιώντας το εργαλείο δεδομένων μπορούμε να συνδέσουμε μια εφαρμογή της Visual Basic με μία βάση δεδομένων και να ανακτήσουμε ένα σύνολο εγγραφών (recordset).

Με τον όρο recordset object αναφέρεται ένα αντικείμενο που αυτόματα δημιουργείται όταν ανοίγουμε μία βάση δεδομένων. Το recordset αποτελεί ένα σύνολο από εγγραφές.

5.2. Ιδιότητες βάσεων δεδομένων

5.2.1. Η ιδιότητα Connect

Η ιδιότητα Connect προσδιορίζει τον τύπο της βάσης δεδομένων που χρησιμοποιείται και σε ορισμένες περιπτώσεις κάποιες επιπρόσθετες παραμέτρους (πχ. password).

5.2.2. Η ιδιότητα DatabaseName

Η ιδιότητα DatabaseName χρησιμοποιείται για τον προσδιορισμό της βάσης δεδομένων που θα χρησιμοποιηθεί. Οι βάσεις δεδομένων που μπορούν να χρησιμοποιηθούν σε μία εφαρμογή της Visual Basic μπορούν να βρίσκονται σε τοπικό ή σε remote επίπεδο, δηλαδή στο server ενός δικτύου. Ακόμα η χρήση μιας βάσης δεδομένων μπορεί να γίνεται ταυτόχρονα από πολλούς χρήστες.

5.2.3. Η ιδιότητα Exclusive

Από τη στιγμή που θα επιτύχουμε πρόσβαση σε μία βάση δεδομένων, μπορούμε να απαιτήσουμε τον αποκλειστικό έλεγχό της, ορίζοντας την τιμή της ιδιότητας Exclusive σαν True. Σε κάποιες περιπτώσεις μία τέτοια απαίτηση μπορεί πραγματικά να είναι απαραίτητη. Χρησιμοποιώντας αποκλειστικά μία βάση δεδο-

μένων, επιτυγχάνεται γρηγορότερη πρόσβαση στα δεδομένα, άρα και ταχύτερη εκτέλεση της εφαρμογής.

5.2.4. Η ιδιότητα Options

Ορίζοντας τιμές στην ιδιότητα Options, μπορούμε να επιτύχουμε έλεγχο σε ότι αφορά στην πρόσβαση στους πίνακες της βάσης δεδομένων. Μέσω αυτών των επιλογών μπορούμε να ορίσουμε απαγόρευση πρόσβασης σε κάποιους πίνακες οι οποίοι περιέχουν τις εγγραφές εκείνες που θέλουμε να χαρακτηρίσουμε σαν μη προσβάσιμες.

5.2.5. Η ιδιότητα ReadOnly

Αν μέσω μίας εφαρμογής της Visual Basic θέλουμε να ανοίξουμε μία βάση δεδομένων, την οποία δεν πρόκειται να τροποποιήσουμε σε ότι αφορά την δομή ή τα δεδομένα της, τότε μπορούμε να θέσουμε στην ιδιότητα ReadOnly την τιμή True.

5.2.6. Η ιδιότητα RecordSource

Η ιδιότητα RecordSource προσδιορίζει πού θα βρεθούν τα δεδομένα της βάσης δεδομένων που θα χρησιμοποιηθεί στην εφαρμογή της Visual Basic. Η ιδιότητα αυτή δεν έχει νόημα παρά μόνο αφού ανοιχτεί η βάση δεδομένων.

5.3. Τα εργαλεία εμφάνισης δεδομένων

Σαν εργαλεία εμφάνισης (Bound Controls) χαρακτηρίζονται όλα εκείνα τα εργαλεία με τα οποία δημιουργούνται τα αντίστοιχα αντικείμενα, μέσω των οποίων επιτυγχάνεται η εμφάνιση των πληροφοριών που βρίσκονται σε μια βάση δεδομένων που έχει συνδεθεί με την εφαρμογή μας. Όταν ένα αντικείμενο εμφάνισης συνδέεται με το αντικείμενο δεδομένων, τότε η Visual Basic αποθέτει στο συγκεκριμένο αντικείμενο εμφάνισης, τις τιμές ενός πεδίου από την επιλεγμένη βάση δεδομένων. Οι τιμές των πεδίων μιας εγγραφής μπορούν να αλλαχτούν μέσω της εφαρμογής της Visual Basic, οι δε αλλαγές που πραγματοποιούνται στη συγκεκριμένη εγγραφή, σώζονται αυτόματα μόλις ο έλεγχος του προγράμματος περάσει σε μία άλλη εγγραφή. Η Visual Basic υποστηρίζει αρκετά εργαλεία (εικόνα (image) ετικέτα (label) πλαίσιο εικόνας (picture box) πλαίσιο κειμένου (text box) πλαίσιο ελέγχου (check box) λίστα εμφάνισης δεδομένων (DBList) συνδυασμένη λίστα εμφάνισης δεδομένων (DBCombo) κλπ) τα οποία μπορούν δημιουργήσουν τα αντίστοιχα αντικείμενα τα οποία θα συνδεθούν με το αντικείμενο δεδομένων και θα χρησιμοποιηθούν για την εμφάνιση των πληροφοριών της βάσης δεδομένων.

Τα αντικείμενα εμφάνισης παρουσιάζουν δύο χαρακτηριστικές ιδιότητες :

- DataField: προσδιορίζει το όνομα του πεδίου οι τιμές του οποίου εμφανίζονται στο συγκεκριμένο αντικείμενο εμφάνισης.
- DataSource: προσδιορίζει το όνομα του αντικείμενου δεδομένων με το οποίο συνδέεται το συγκεκριμένο αντικείμενο εμφάνισης.

6. ΓΡΑΦΙΚΑ

Η Visual Basic διαθέτει δύο διαφορετικούς τρόπους για τη δημιουργία γραφικών σε μία εφαρμογή. Μπορούμε να χρησιμοποιήσουμε είτε τα εργαλεία γραφικών είτε μεθόδους γραφικών. Σχεδιάζουμε σημεία, γραμμές, πλαίσια, κύκλους και άλλα γεωμετρικά σχήματα και τροποποιούμε τη μορφή και τη θέση εμφάνισής τους επάνω στην φόρμα.

6.1. Το σύστημα συντεταγμένων

Κάθε διαδικασία με γραφικά χρησιμοποιεί ένα σύστημα συντεταγμένων που αφορά την περιοχή σχεδίασης. Χρησιμοποιώντας το σύστημα συντεταγμένων μπορούμε να προσδιορίσουμε σημεία στην οθόνη, σε μία φόρμα, σ' ένα πλαίσιο εικόνας. Η μορφή που χρησιμοποιείται για να οριστεί ένα σημείο είναι η (x,y). Το x είναι η τιμή της προβολής του σημείου στον οριζόντιο άξονα των x και το y η αντίστοιχη τιμή της προβολής του στον κατακόρυφο άξονα των y.

Το σύστημα συντεταγμένων της Visual Basic υπόκειται σε κάποιους κανόνες:

- Η αλλαγή μεγέθους ή η μετακίνηση ενός αντικειμένου καθορίζεται από το σύστημα συντεταγμένων που χρησιμοποιεί η οντότητα που περιέχει αυτό το αντικείμενο. Για παράδειγμα εάν σχεδιάζουμε ένα πλαίσιο εικόνας πάνω σε μία φόρμα, το σύστημα συντεταγμένων της φόρμας, είναι αυτό που θα καθορίζει οτιδήποτε αφορά τη μετακίνηση ή τις αλλαγές του μεγέθους του πλαισίου εικόνας.
- Όλες οι μέθοδοι γραφικών χρησιμοποιούν το σύστημα συντεταγμένων της φόρμας ή του αντικειμένου στο οποίο αναφέρονται. Αν για παράδειγμα χρησιμοποιούμε μία μέθοδο σχεδίασης για ένα πλαίσιο εικόνας, το σύστημα συντεταγμένων που χρησιμοποιείται είναι αυτό του πλαισίου εικόνας.
- Οι εντολές που χρησιμοποιούνται για τον επαναπροσδιορισμό του μεγέθους ή τη μετακίνηση μιας φόρμας, εκφράζουν πάντοτε τη θέση και το μέγεθος της φόρμας σε twips.
- Η επάνω αριστερή γωνία της οθόνης έχει τιμές συντεταγμένων (0,0). Επίσης η επάνω αριστερή γωνία κάθε φόρμας, πλαισίου εικόνας ή πλαισίου, έχει - στο εξ ορισμού σύστημα συντεταγμένων - τιμές (0,0).
- Οι μονάδες μέτρησης που χρησιμοποιούνται για τον προσδιορισμό των σημείων κατά μήκος των δύο αξόνων ονομάζονται κλίμακα. Η Visual Basic μπορεί να υποστηρίξει στο σύστημα συντεταγμένων διαφορετική κλίμακα για κάθε άξονα. Επίσης μπορεί ν' αλλαχτεί τόσο η διεύθυνση και τα σημεία

εκκίνησης των αξόνων, όσο και η κλίμακα του συστήματος συντεταγμένων.

6.2. Μονάδες μέτρησης

Εξ ορισμού στην Visual Basic όλες οι εντολές σχεδίασης, μετακίνησης και καθορισμού μεγέθους γραφικών, χρησιμοποιούν σαν μονάδα μέτρησης το twip. Ένα twip είναι το 1/20 του ενός σημείου εκτύπωσης ή διαφορετικά 1440 twips ισούνται με μία ίντσα. Αυτές οι αντιστοιχίες αναφέρονται στο μέγεθος του αντικειμένου κατά την εκτύπωση. Σε ότι αφορά την εμφάνιση του αντικειμένου στην οθόνη, οι φυσικές αποστάσεις ποικίλουν ανάλογα με το μέγεθος της οθόνης.

6.3. Αλλαγή κλίμακας συστήματος συντεταγμένων

Ο χρήστης σε ότι αφορά το σύστημα των συντεταγμένων έχει τρεις επιλογές:

- ✓ να χρησιμοποιήσει την εξ ορισμού κλίμακα μέτρησης, δηλαδή τα twips
- ✓ να χρησιμοποιήσει μία από τις υπόλοιπες έτοιμες κλίμακες που παρέχει η Visual Basic
- ✓ να δημιουργήσει μία δική του κλίμακα.

Ένας τρόπος, ο απλούστερος, ορισμού του συστήματος συντεταγμένων είναι να αποδοθεί μία τιμή στην ιδιότητα ScaleMode. Οι δυνατές τιμές της ιδιότητας, η οποία αναφέρεται στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής, φαίνονται στον πίνακα 2.

Τιμή	Περιγραφή
0 - User	Δηλώνει πως μία ή περισσότερες από τις τιμές των ιδιοτήτων ScaleHeight, ScaleWidth, ScaleLeft και ScaleTop ορίζονται από το χρήστη.
1 - Twip	Η εξ ορισμού κλίμακα, όπου 567 twips = 1 cm.
2 - Point	Όπου 1 inch = 72 σημεία (points).
3 - Pixel	Όπου σαν pixel ορίζεται η μικρότερη μονάδα ανάλυσης της οθόνης.
4 - Character	Όπου 1 χαρακτήρας (character) αναπαρίσταται μέσα σε ορθογώνιο πλαίσιο, οριζόντιας πλευράς 120 twips και κατακόρυφης πλευράς 240 twips
5 - Inch	Ίντσα, όπου 1 inch = 2,53 cm.
6 - Milimeter	Χιλιοστό του μέτρου.
7 - Centimeter	Εκατοστό του μέτρου.

Πίνακας 2: Οι δυνατές τιμές της ιδιότητας ScaleMode.

Όταν αποδίδουμε μία από τις δυνατές τιμές (πλην της 0) στην ιδιότητα ScaleMode, η Visual Basic επαναπροσδιορίζει τις τιμές των ιδιοτήτων ScaleWidth και ScaleHeight, αποδίδοντάς τους τιμές που να αντιστοιχούν στη νέα κλίμακα συντεταγμένων. Ταυτόχρονα μηδενίζονται οι τιμές των ιδιοτήτων ScaleTop και ScaleLeft. Επίσης αλλάζουν οι τιμές των ιδιοτήτων CurrentX και CurrentY, έτσι ώστε να εκφράζουν τις συντεταγμένες του τρέχοντος σημείου στο καινούριο σύστημα συντεταγμένων.

Ένας άλλος τρόπος ορισμού του συστήματος συντεταγμένων είναι χρησιμοποιώντας τις συσχετιζόμενες ιδιότητες ScaleHeight, ScaleWidth, ScaleLeft και ScaleTop. Αποδίδοντας τιμή σε τουλάχιστον μία από αυτές, μπορούμε να δημιουργήσουμε ένα δικό μας σύστημα συντεταγμένων για το αντικείμενο στο οποίο αναφερόμαστε. Θέτοντας όμως τιμή σε οποιαδήποτε από τις παραπάνω ιδιότητες, αυτόματα τίθεται η τιμή 0 στην ιδιότητα ScaleMode.

Στην κλίμακα που δημιουργεί ο χρήστης, μπορεί ν' αποδώσει στην επάνω αριστερή γωνία τόσο της φόρμας όσο και του αντικειμένου που περιέχεται σ' αυτήν, οποιοδήποτε ζεύγος τιμών θέλει, και όχι υποχρεωτικά το ζεύγος τιμών (0,0).

Τέλος σημειώνουμε πως με τη χρήση της μεθόδου Scale μπορούμε επίσης να αποδώσουμε τιμές στις ιδιότητες ScaleHeight, ScaleWidth, ScaleLeft και ScaleTop, πράγμα που σημαίνει, όπως ήδη εξηγήθηκε, τον ορισμό μιας καινούριας κλίμακας συντεταγμένων.

6.4. Χρώματα

Η Visual Basic υποστηρίζει 256 χρώματα, με την προϋπόθεση βέβαια πως το σύστημα απεικόνισης του υπολογιστή μπορεί να τα αποδώσει. Η ανάγκη για ταυτόχρονη απεικόνιση 256 διαφορετικών χρωμάτων, εμφανίζεται κύρια σε Multimedia εφαρμογές και στις εφαρμογές εκείνες που χειρίζονται εικόνες με μεγάλες απαιτήσεις στη χρωματική απόδοση.

Μπορούμε να χρησιμοποιήσουμε 256 χρώματα σε μεθόδους γραφικών που επιδρούν σε φόρμες, αντικείμενα πλαισίου εικόνας και αντικείμενα εικόνας. Θα πρέπει όμως να σημειωθεί πως για αρχεία εικόνας τύπου Metafile, η Visual Basic δεν υποστηρίζει 256 χρώματα, αλλά μόνο τα 16 της QuickBasic.

Κάθε χρώμα στη Visual Basic αντιπροσωπεύεται από ένα long integer. Η γλώσσα διαθέτει δύο διαφορετικούς τρόπους για τον προσδιορισμό των απαιτούμενων χρωμάτων μιας εφαρμογής:

- ✓ Με χρήση της συνάρτησης RGB.
- ✓ Με χρήση της συνάρτησης QBColor που παρέχει τη δυνατότητα επιλογής ενός από τα 16 χρώματα που υποστηρίζει η QuickBasic.

6.4.1. Η συνάρτηση RGB

Η συνάρτηση RGB επιστρέφει σαν τιμή ένα long integer ο οποίος αντιπροσωπεύει την τιμή ενός χρώματος. Η τιμή αυτή κατ' ουσία αντιστοιχεί στην σχετική ένταση συμμετοχής των χρωμάτων κόκκινο, πράσινο και μπλε, ώστε να δημιουργηθεί το επιθυμητό χρώμα ή χρωματική απόχρωση. Ο τρόπος σύνταξης της συνάρτησης περιλαμβάνει τη χρήση τριών ακέραιων - μετά τη δήλωση του ονόματός της - οι οποίοι μπορούν να παίρνουν τιμές από 0 μέχρι 255 και καθένας από αυτούς αντιπροσωπεύει κατά σειρά, τη συμμετοχή των τριών βασικών χρωμάτων κόκκινου, πράσινου και μπλε.

```
Form1.BackColor = RGB(0,255,0)
Form2.ForeColor = RGB(0,0,128)
```

Κάθε χρώμα ή χρωματική απόχρωση μπορεί να παρασταθεί σαν συνδυασμός διαφορετικών τιμών συμμετοχής των τριών αυτών βασικών χρωμάτων, η παράθεση των τιμών των οποίων πρέπει να γίνεται αποκλειστικά με την προαναφερθείσα σειρά. Στον πίνακα 3 απεικονίζεται η δεκαεξαδική μορφή μερικών συνηθισμένων χρωμάτων και η τιμή της συμμετοχής των τριών βασικών χρωμάτων κόκκινου, πράσινου και μπλε στη δημιουργία καθενός από αυτά.

Χρώμα	Τιμή RGB	Τιμή κόκκινου	Τιμή πράσινου	Τιμή μπλε
Μαύρο	&H00	0	0	0
Μπλε	&HFF0000	0	0	255
Πράσινο	&HFF00	0	255	0
Κόκκινο	&HFF	255	0	0
Κυανό	&HFFFF00	0	255	255
Κίτρινο	&HFFFF	255	255	0
Μοβ	&HFF00FF	255	0	255
Άσπρο	&HFFFFFF	255	255	255

Πίνακας 3: Τιμές RGB μερικών συνηθισμένων χρωμάτων.

6.4.2. Η συνάρτηση QBColor

Η συνάρτηση QBColor δέχεται έναν ακέραιο αριθμό από 0 έως 15, ο οποίος αντιστοιχεί σ' ένα από τα 16 χρώματα που χρησιμοποιούνται από την QuickBasic (πίνακας 4) και επιστρέφει μία τιμή η οποία αντιστοιχίζει το επιλεγμένο χρώμα της QuickBasic, στο αντίστοιχό του στο χρωματικό σύστημα RGB που χρησιμοποιεί η Visual Basic. Για παράδειγμα:

```
TestForm.BackColor = QBColor(4)
```

Με την παραπάνω εντολή, αποδίδεται στην ιδιότητα BackColor της φόρμας TestForm το κόκκινο χρώμα, όπως ορίζεται στην QuickBasic με τη σταθερά 4.

Αριθμός - Χρώμα	Αριθμός - Χρώμα
0 - Μαύρο	8 - Γκρι
1 - Μπλε	9 - Ανοικτό Μπλε
2 - Πράσινο	10 - Ανοικτό Πράσινο
3 - Κυανό	11 - Ανοικτό Κυανό
4 - Κόκκινο	12 - Ανοικτό Κόκκινο
5 - Μοβ	13 - Ανοικτό Μοβ
6 - Κίτρινο	14 - Ανοικτό Κίτρινο
7 - Άσπρο	15 - Έντονο Άσπρο

Πίνακας 4: Τα 16 χρώματα της συνάρτησης QBColor και οι τιμές τους.

6.5. Εργαλεία γραφικών

Η Visual Basic διαθέτει τρία εργαλεία που μπορούν να χρησιμοποιηθούν για τη δημιουργία γραφικών εφέ σε εφαρμογές. Τα εργαλεία αυτά είναι το εργαλείο εικόνα (Image control), το εργαλείο γραμμή (Line control) και το εργαλείο γεωμετρικό σχήμα (Shape control). Τα τρία αυτά εργαλεία γραφικών αποδεικνύονται πολύ χρήσιμα για τη δημιουργία γραφικών κατά τη σχεδίαση της εφαρμογής.

Ένα πλεονέκτημά τους είναι ότι απαιτούν την εγγραφή πολύ λιγότερου κώδικα απ' ότι απαιτείται αν χρησιμοποιηθούν οι μέθοδοι γραφικών. Για παράδειγμα, μπορούμε να σχεδιάσουμε έναν κύκλο χρησιμοποιώντας είτε το εργαλείο γεωμετρικού σχήματος, είτε τη μέθοδο Circle. Η μέθοδος Circle βέβαια απαιτεί την εγγραφή κώδικα, ενώ με τη χρησιμοποίηση του εργαλείου γεωμετρικό σχήμα δεν έχουμε παρά να "ζωγραφίσουμε" τον ζητούμενο κύκλο ορίζοντας την κατάλληλη τιμή στην ιδιότητα Shape του αντικειμένου.

Βέβαια από την άλλη πλευρά παρουσιάζουν και κάποια μειονεκτήματα:

- ✓ Τα αντικείμενα γραφικών δεν μπορούν να αναδυθούν μέσα από άλλα αντικείμενα, εκτός κι αν βρίσκονται μέσα σε ένα τρίτο που έχει αυτή την ικανότητα.
- ✓ Δεν μπορούν να συμπεριλάβουν μέσα τους άλλα αντικείμενα.
- ✓ Δεν μπορούν να υποστηρίξουν διαδικασίες εστίασης πάνω στο γραφικό κατά τη διάρκεια της εκτέλεσης της εφαρμογής.

6.6. Μέθοδοι γραφικών

Η Visual Basic παρέχει συμπληρωματικά προς τα εργαλεία γραφικών, μία σειρά από μεθόδους γραφικών οι οποίες είναι σχεδιασμένες ειδικά για τη δημιουργία γραφικών στις εφαρμογές της. Οι μέθοδοι γραφικών μπορούν να προσφέρουν μερικά ενδιαφέροντα οπτικά εφέ, που δεν μπορούν να δημιουργηθούν με τη χρήση των εργαλείων γραφικών. Γενικά, οι μέθοδοι

γραφικών δεν ενδείκνυνται για τις περιπτώσεις εκείνες, που θέλουμε να δημιουργήσουμε πολύ απλά σχέδια στην παρουσίαση της εφαρμογής μας. Στις περιπτώσεις δημιουργίας γραφικών με μεθόδους γραφικών, θα πρέπει να εκτελέσουμε την εφαρμογή, για να διαπιστώσουμε το οπτικό αποτέλεσμα.

Οι μέθοδοι γραφικών μπορούν να εφαρμοστούν στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής και είναι οι ακόλουθοι:

- Cls, η οποία καθαρίζει όλα τα γραφικά που δημιουργούνται κατά τη διάρκεια εκτέλεσης της εφαρμογής σε μία φόρμα ή σ' ένα πλαίσιο εικόνας.
- PSet, η οποία προσδίδει το καθοριζόμενο χρώμα σ' ένα σημείο του αντικειμένου.
- Point, η οποία επιστρέφει, σε κλίμακα RGB, την τιμή του χρώματος ενός συγκεκριμένου σημείου μιας φόρμας ή ενός πλαισίου εικόνας.
- Line, η οποία χρησιμοποιείται για τη σχεδίαση γραμμών και ορθογωνίων σχημάτων.
- Circle, η οποία χρησιμοποιείται για τη σχεδίαση καμπυλόγραμμων σχημάτων (κύκλου, έλλειψης ή τόξου).
- Print, η οποία αναφέρεται στο αντικείμενο Debug μπορεί να θεωρηθεί σαν μέθοδος γραφικών. Η μέθοδος αυτή μπορεί να εμφανίσει στο παράθυρο Debug κείμενο που έχει αποδοθεί σαν τιμή σε μία μεταβλητή.
- PaintPicture, η οποία σχεδιάζει τα περιεχόμενα ενός αρχείου γραφικών μορφοποίησης .ico, .wmf, .bmp και .dib. Εφαρμόζεται στα αντικείμενα φόρμα, πλαίσιο εικόνας.

6.7. Ιδιότητες γραφικών

Οι φόρμες και αρκετά άλλα αντικείμενα, χαρακτηρίζονται από μία σειρά από ιδιότητες γραφικών, οι ονομασίες και οι κατηγορίες των οποίων παρουσιάζονται στον παρακάτω πίνακα.

Κατηγορία	Ιδιότητες
Τρεχουσών συντεταγμένων	CurrentX, CurrentY
Επεξεργασίας εμφάνισης	AutoRedraw, ClipControls
Τεχνικών σχεδίασης	DrawMode, DrawStyle, DrawWidth, BorderStyle, BorderWidth
Τεχνικών γεμίσματος	FillColor, FillStyle
Χρωματισμού	BackColor, ForeColor, BorderColor,FillColor
Ορισμού κλίμακας	ScaleLeft, ScaleTop, ScaleHeight,ScaleWidth, ScaleMode

Πίνακας 5: Ιδιότητες γραφικών ανά κατηγορία.

6.7.1. Ιδιότητες τρεχουσών συντεταγμένων

Οι ιδιότητες CurrentX και CurrentY επιστρέφουν ή θέτουν τιμή στην οριζόντια και κατακόρυφη συντεταγμένη αντίστοιχα, του σημείου από το οποίο θα αρχίσει να εφαρμόζεται στη συνέχεια μία μέθοδος γραφικών ή μία διαδικασία εκτύπωσης. Οι ιδιότητες αυτές συναντώνται στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής. Δεν είναι όμως διαθέσιμες στη φάση της σχεδίασης, παρά μόνο στη φάση εκτέλεσης της εφαρμογής.

Η σύνταξη της εντολής είναι :

όνομα αντικειμένου. CurrentX [= x]

όνομα αντικειμένου.CurrentY [= y]

όπου x και y είναι αριθμοί που προσδιορίζουν τις συντεταγμένες θέσης. Οι αριθμοί αυτοί αντιστοιχούν σε twips, ή στην οποιαδήποτε άλλη κλίμακα μέτρησης χρησιμοποιείται.

6.7.2. Ιδιότητες επεξεργασίας εμφάνισης

Η ιδιότητα AutoRedraw (Αυτόματη Επανασχεδίαση) παίρνει τιμές boolean και χαρακτηρίζει κάθε αντικείμενο φόρμα και πλαίσιο εικόνας. Η AutoRedraw χρησιμοποιείται για την αυτόματη επανασχεδίαση επί της οθόνης γραφικών, τα οποία έχουν δημιουργηθεί με χρήση μεθόδων γραφικών, όταν στη φάση εκτέλεσης της εφαρμογής

- ✓ ολόκληρα γραφικά ή μέρος τους, αποκαλύπτονται μετά από μετακίνηση άλλων αντικειμένων που τα κάλυπταν
- ✓ γραφικά αλλάζουν το μέγεθος τους.

Η AutoRedraw έχει εξ ορισμού τιμή False. Αυτό σημαίνει πως κάθε γραφικό που σχηματίστηκε με τη χρήση μεθόδων γραφικών και εμφανίζεται πάνω στη φόρμα, θα χαθεί αν καλυφτεί προσωρινά από ένα άλλο αντικείμενο. Επίσης το ίδιο αποτέλεσμα απώλειας των γραφικών θα συμβεί αν μικρύνει τη φόρμα που τα περιλαμβάνει και στη συνέχεια την επανέλθει στις προηγούμενες διαστάσεις της. Όταν όμως η τιμή της AutoRedraw είναι True, τότε η Visual Basic αναλαμβάνει τη διαχείριση της οθόνης και την επανεμφάνιση των γραφικών όποτε χρειάζεται.

6.7.3. Ιδιότητες τεχνικών σχεδίασης

Η ιδιότητα DrawWidth (Πλάτος Σχεδίασης) προσδιορίζει το πλάτος της γραμμής σχεδίασης για τη μεθόδους γραφικών Circle, Cls, Line, PaintPicture, Point, Print και PSet. Εφαρμόζεται στα αντικείμενα φόρμα, πλαίσιο εικόνας, εκτυπωτής και OLE. Όταν η τιμή της DrawWidth είναι μεγαλύτερη από 1, τότε οι τιμές 1 έως 4 της ιδιότητας DrawStyle, δεν επιφέρουν το αποτέλεσμα που περιγράφουν.

Η ιδιότητα BorderWidth (Πλάτος Περιγράμματος) προσδιορίζει το πάχος του περιγράμματος των αντικειμένων γραμμή και γεωμετρικό σχήμα..

Η ιδιότητα DrawStyle (Είδος Σχεδίασης) εφαρμόζεται επί των αντικειμένων φόρμα, πλαίσιο εικόνας και εκτυπωτής και προσδιορίζει το τρόπο εμφάνισης των σχεδιαζόμενων γραμμών. Σημειώνεται πως αν η ιδιότητα DrawWidth πάρει τιμές μεγαλύτερες του 1, τότε οι επιλογές 1-4 της ιδιότητας DrawStyle παρέχουν το ίδιο αποτέλεσμα.

Η ιδιότητα BorderStyle (Είδος Περιγράμματος) έχει διαφορετικό σκοπό και χρήση στα ποικίλα αντικείμενα που απαντάται. Στα αντικείμενα γραμμή και γεωμετρικό σχήμα, η BorderStyle έχει την ίδια χρήση με αυτή που έχει η ιδιότητα DrawStyle, δηλαδή περιγράφει το είδος της σχεδιαζόμενης γραμμής. Στα αντικείμενα DBList και DBCombo, η BorderStyle προσδιορίζει κατά πόσο το αντικείμενο θα έχει απλό περίγραμμα (border), και αν ναι, αν θα εμφανίζονται διάφορα στοιχεία ενός παραθύρου, όπως τίτλος, πλήκτρα μεγιστοποίησης και ελαχιστοποίησης και αν το παράθυρο θα είναι σταθερού ή μεταβλητού μεγέθους. Στα αντικείμενα φόρμα, πλαίσιο εικόνας, εικόνα, πλαίσιο κειμένου, ετικέτα, πλέγμα και OLE η BorderStyle επιστρέφει ή θέτει τιμή που προσδιορίζει κατά πόσο το αντικείμενο θα έχει απλό περίγραμμα ή δεν θα έχει καθόλου.

Η ιδιότητα DrawMode (Τρόπος Σχεδίασης) προσδιορίζει την εμφάνιση των γραφικών που προέρχονται από τη χρησιμοποίηση μεθόδων γραφικών στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής, καθώς επίσης και την εμφάνιση των αντικειμένων γραμμή και γεωμετρικό σχήμα. Μπορούμε να θεωρήσουμε ότι για τη σχεδίαση κάθε γραφικού μπορεί να χρησιμοποιηθεί ένας μεγάλος αριθμός από πένες σχεδίασης. Η χρήση κάθε μιας από αυτές επιφέρει και διαφορετικά αποτελέσματα στη σχεδίαση. Η τιμή της ιδιότητας DrawMode, προσδιορίζει ποιο είναι το οπτικό αποτέλεσμα αν συμβεί ένα γραφικό να δημιουργείται πάνω από ένα άλλο στη φάση της εκτέλεσης της εφαρμογής.

6.7.4. Ιδιότητες τεχνικών γεμίσματος

Η ιδιότητα FillStyle (Είδος Γεμίσματος) προσδιορίζει το εσωτερικό σχέδιο ενός αντικειμένου γεωμετρικό σχήμα ή ενός γραφικού που δημιουργείται με τη χρήση μεθόδων γραφικών πάνω σε ένα από τα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής.

Η ιδιότητα FillColor (Χρώμα Γεμίσματος) προσδιορίζει το εσωτερικό χρώμα ενός αντικειμένου γεωμετρικό σχήμα ή ενός γραφικού που δημιουργείται με τη χρήση των μεθόδων γραφικών Line και Circle πάνω σε ένα από τα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής. Η εξ ορισμού τιμή της ιδιότητας FillColor είναι 0 (Μαύρο), η δε επιλογή των χρωμάτων γίνεται με τη βοήθεια της παλέτας χρωμάτων. Όταν η ιδιότητα FillStyle έχει τιμή Διαφανής, η τιμή της ιδιότητας FillColor αγνοείται.

6.7.5. Ιδιότητες χρωματισμού

Οι ιδιότητες BackColor (Χρώμα Παρασκηνίου) και ForeColor (Χρώμα Προσκήνιου) ορίζουν το χρώμα του παρασκηνίου και του προσκήνιου αντίστοιχα για το επιλεγμένο αντικείμενο. Οι δύο αυτές ιδιότητες αναφέρονται σε πάνω από 20 αντικείμενα, μεταξύ των οποίων η φόρμα, η ετικέτα, το πλαίσιο εικόνας, το πλαίσιο κειμένου κλπ.

Η ιδιότητα BorderColor (Χρώμα Περιγράμματος), αποδίδει χρώμα στο περίγραμμα ενός αντικειμένου. Η ιδιότητα αυτή χαρακτηρίζει τα αντικείμενα γεωμετρικό σχήμα και γραμμή.

Η απόδοση τιμής στις τρεις αυτές ιδιότητες μπορεί να γίνει στη φάση σχεδίασης της εφαρμογής μέσω του παράθυρου των ιδιοτήτων. Στη φάση της εκτέλεσης της εφαρμογής η απόδοση των τιμών γίνεται με τον εξής τρόπο σύνταξης :

```
Form1.BackColor = color
Text1.ForeColor = color
Shape1.BorderColor = color
```

όπου color είναι μία τιμή ή μία σταθερά που προσδιορίζει το χρώμα που αποδίδουμε στα εν λόγω αντικείμενα. Αν πρόκειται για τιμή, αυτή εκφράζεται είτε χρησιμοποιώντας τις χρωματικές συναρτήσεις RGB και QBColor, είτε χρησιμοποιώντας τη δεκαεξαδική έκφραση των χρωματικών αποχρώσεων της παλέτας χρωμάτων.

6.7.6. Ιδιότητες ορισμού κλίμακας

Οι ιδιότητες ScaleWidth (Πλάτος Κλίμακας) και ScaleHeight (Ύψος Κλίμακας) επιστρέφουν ή θέτουν τιμή στην οριζόντια και κατακόρυφη αντίστοιχα εσωτερική διάσταση ενός αντικειμένου όταν χρησιμοποιούνται σε αυτό μέθοδοι γραφικών. Λέγοντας εσωτερική διάσταση του αντικειμένου, εννοούμε ότι δεν συμπεριλαμβάνεται το πάχος της γραμμής πλαισίου. Οι ιδιότητες ScaleWidth και ScaleHeight εφαρμόζονται στα αντικείμενα φόρμα και πλαίσιο εικόνας και στο ειδικό αντικείμενο εκτυπωτής.

Οι ιδιότητες ScaleWidth και ScaleHeight διαφέρουν από τις ιδιότητες Width και Height, αφού οι δεύτερες αναφέρονται στις εξωτερικές διαστάσεις του αντικειμένου, συμπεριλαμβανομένου δηλαδή και του πάχους της γραμμής του πλαισίου του.

Οι ιδιότητες ScaleLeft (Αριστερό Κλίμακας) και ScaleTop (Κορυφή Κλίμακας) αποδίδουν τιμές στις, οριζόντια και κατακόρυφη αντίστοιχα, συντεταγμένες της επάνω αριστερής γωνίας ενός αντικειμένου όταν χρησιμοποιούνται μέθοδοι γραφικών. Οι ιδιότητες αυτές χαρακτηρίζουν τα αντικείμενα φόρμα, πλαίσιο εικόνας και το ειδικό αντικείμενο εκτυπωτής.

Οι ιδιότητες ScaleLeft και ScaleTop δεν είναι ταυτόσημες με τις ιδιότητες Left και Top οι οποίες απαντώνται σε περισσότερα από 20 αντικείμενα. Η ιδιότητα

Left δηλώνει την απόσταση μεταξύ της αριστερής εσωτερικής ακμής ενός αντικειμένου και της αριστερής ακμής του αντικειμένου που το περιέχει, συνηθέστερα της φόρμας. Αντίστοιχα η ιδιότητα Top δηλώνει την απόσταση μεταξύ της επάνω εσωτερικής ακμής του αντικειμένου και της επάνω ακμής του αντικείμενου που το περιέχει. Για τη φόρμα οι τιμές των ιδιοτήτων Left και Top εκφράζονται πάντα σε twips, ενώ για τα άλλα αντικείμενα εκφράζονται σύμφωνα με το σύστημα συντεταγμένων που ισχύει για το αντικείμενο στο οποίο περιέχονται.

7. ΠΟΛΥΜΕΣΑ (MULTIMEDIA)

Η Visual Basic μπορεί να συμπεριλάβει στις εφαρμογές της τόσο δεδομένα κειμένου, όσο και δεδομένα ήχου, εικόνας, προσομοίωσης κίνησης και video. Κύρια χάρη στις διαδικασίες OLE και DDE που υποστηρίζει, η Visual Basic μπορεί να επιλεγεί σαν εργαλείο συγγραφής (authoring tool), Interactive Multimedia και Hypermedia εφαρμογών. Η χρησιμοποίηση Multimedia δεδομένων στις εφαρμογές, συνεπάγεται μεγαλύτερες απαιτήσεις σε υλικό, τόσο σε ταχύτητα επεξεργασίας, όσο και σε μνήμη RAM, αλλά και σε αποθηκευτικές δυνατότητες. Τα αρχεία κυρίως video (πχ. .avi) και προσομοίωσης κίνησης (πχ. .flc), αλλά και αυτά ήχου (πχ. .wav) και στη συνέχεια εικόνας (πχ. .bmp) είναι πολύ μεγάλου μεγέθους. Κατά συνέπεια η χρησιμοποίηση τους χωρίς να έχουν εξασφαλιστεί οι απαιτούμενοι υλικοί πόροι του συστήματος, θα επιφέρει αρνητικά αποτελέσματα στην εφαρμογή. Ένα video που συμπεριλάβαμε σε μία εφαρμογή, αλλά στη φάση της εκτέλεσης εμφανίζεται με μία τρεμώδη και σπαστή κίνηση, σίγουρα δεν μπορεί να καταχωρηθεί στα θετικά στοιχεία της εφαρμογής.

7.1. Multimedia εργαλεία της Visual Basic

Τα εργαλεία (controls) που παρέχει η Visual Basic προς χρήση για δημιουργία Multimedia εφαρμογών είναι αυτά τα οποία μπορούν να δεχτούν δεδομένα κειμένου, εικόνας, ήχου, animation και video. Αυτά τα εργαλεία είναι :

- Εργαλείο πλαίσιο εικόνας (PictureBox) : Το εργαλείο αυτό, μέσω της ιδιότητάς του Picture, μπορεί να χρησιμοποιηθεί για την εμφάνιση εικόνων διαφορετικής μορφοποίησης (πχ. bmp, dib, wmf, ico κλπ). Αν η εικόνα είναι μεγαλύτερη από το αντικείμενο πλαίσιο εικόνας, τότε ένα τμήμα μόνο της εικόνας εμφανίζεται. Αν η εικόνα είναι μικρότερη από το αντικείμενο πλαίσιο εικόνας, τότε προβάλλεται ολόκληρη, ενώ μένει κενός ο υπόλοιπος χώρος του πλαισίου εικόνας. Το αντικείμενο πλαίσιο εικόνας επαναπροσδιορίζει τις διαστάσεις του σύμφωνα με τις διαστάσεις της εικόνας, όταν στην ιδιότητά του AutoSize έχει τεθεί τιμή True.
- Εργαλείο εικόνα (Image) : Η χρήση του εργαλείου εικόνα είναι παρόμοια με αυτή του εργαλείου πλαίσιο εικόνας. Εμφανίζει της ίδιας μορφοποίησης αρχεία εικόνας και οι διαστάσεις του προσαρμόζονται στις διαστάσεις της εικόνας όταν στην ιδιότητα Stretch τεθεί τιμή True.
- Εργαλείο ετικέτα (Label): Από την άποψη πως όταν στην ιδιότητα BackStyle αποδώσουμε τιμή Transparent, το κείμενο που έχει γραφτεί στο αντικείμενο ετικέτα, μπορεί να προβληθεί πάνω από κάποιο άλλο αντικείμενο, δημιουργώντας την εντύπωση των τίτλων ή των υπότιτλων πάνω από γραφικά ή video, το εργαλείο ετικέτα μπορεί να θεωρηθεί σαν ένα Multimedia εργαλείο.
- Εργαλείο πολυμέσα (MMControl) : Το εργαλείο αυτό παρέχει το βασικό interface για το χειρισμό όλων των Multimedia συσκευών (κασετόφωνο, CD-player, video κλπ). Θέτοντας στις αντίστοιχες ιδιότητες τις κατάλληλες τιμές, μπορούμε να επιλέξουμε εκείνα τα πλήκτρα που θέλουμε να είναι ορατά στη φάση της εκτέλεσης και να ανταποκρίνονται σε λειτουργίες.
- Εργαλείο OLE : Σαν multimedia εργαλείο μπορεί βέβαια να καταχωρηθεί και το εργαλείο OLE, ανεξάρτητα από το γεγονός πως η χρήση του δεν στοχεύει αποκλειστικά στην πρόσβαση σε multimedia τύπου εφαρμογές. Το εργαλείο OLE χρησιμοποιείται ευρέως για την δημιουργία απλών και εύκολων multimedia εφαρμογών σε περιβάλλον Visual Basic.

Όπως όλα τα αντικείμενα της Visual Basic, έτσι και το αντικείμενο OLE χαρακτηρίζεται από μία σειρά από ιδιότητες. Από αυτές στη συνέχεια παρουσιάζεται αναλυτικά η ιδιαίτερα σημαντική ιδιότητα Action, η οποία είναι διαθέσιμη μόνο κατά τη φάση της εκτέλεσης της εφαρμογής.

Κατά τη διάρκεια εκτέλεσης μιας εφαρμογής μπορούμε με ένα αντικείμενο OLE να πραγματοποιήσουμε μία σειρά από διαφορετικές ενέργειες, ανάλογα με την τιμή που αποδίδουμε στη ιδιότητα Action. Οι σημαντικότερες τιμές που μπορεί να πάρει η ιδιότητα Action είναι οι παρακάτω :

- ✓ 7 Αυτή η τιμή ενεργοποιεί το αντικείμενο OLE. Το τι ακριβώς θα συμβεί όταν το αντικείμενο θα ενεργοποιηθεί εξαρτάται από τη τιμή της ιδιότητάς του Verb. Κάθε τύπος αντικείμενου μπορεί να υποστηρίζει το δικό του σύνολο από verbs, που ουσιαστικά δεν είναι τίποτε άλλο από το σύνολο των ενεργειών που μπορεί να πραγματοποιήσει το αντικείμενο. Για να δούμε την λίστα αυτών των ενεργειών, θα πρέπει στην ιδιότητα AutoVerbMenu να έχουμε θέσει τιμή True και στη φάση της εκτέλεσης της εφαρμογής να πατήσουμε το δεξί πλήκτρο του ποντικιού όταν βρισκόμαστε πάνω από αντικείμενο OLE. Αν λοιπόν στην ιδιότητα Verb έχουμε αποδώσει τιμή 2, αυτό σημαίνει πως όταν ενεργοποιήσουμε το αντικείμενο μέσω κώδικα (ΌνομαOLE. Action = 7), τότε αυτό

που θα συμβεί είναι η δεύτερη ενέργεια που παρατίθεται στη λίστα των ενεργειών. Η εξ ορισμού τιμή της ιδιότητας Verb είναι 0, πράγμα που σημαίνει πως αυτό το οποίο συμβαίνει όταν το αντικείμενο ενεργοποιείται, είναι η πιο συνηθισμένη ενέργεια, η οποία βέβαια εξαρτάται από το είδος της διασυνδεδεμένης εφαρμογής (π.χ. edit για κείμενο, play για ήχο και video).

- ✓ 9 Αυτή η τιμή σταματάει τη σύνδεση, εφ' όσον πρόκειται για ένα ενσωματωμένο (embedded) αντικείμενο OLE, με την διασυνδεδεμένη εξωτερική εφαρμογή την οποία και κλείνει, ενώ δεν επιφέρει κανένα ουσιαστικό αποτέλεσμα στην περίπτωση του συνδεδεμένου (linked).

7.2. Ήχος

Παρ' ότι η Visual Basic προσφέρει έναν πραγματικά μεγάλο αριθμό συναρτήσεων, που προσθέτουν στην εφαρμογή πολλά multimedia χαρακτηριστικά και καθιστούν ευκολότερο τον τρόπο επικοινωνίας με τον χρήστη, εντούτοις φαίνεται να υστερεί σε επίπεδο διαχείρισης ήχου. Η QuickBasic με τις συναρτήσεις της Sound () και Play (), παρέχει τη δυνατότητα για τη δημιουργία ενδιαφερόντων ηχητικών εφέ. Αντίθετα, η Visual Basic φαίνεται να παρέχει πολύ λιγότερες δυνατότητες σε ότι αφορά τον ήχο, αφού δεν έχει να παρουσιάσει παρά μία μόνο εντολή, την Beep ().

Στην πραγματικότητα όμως εφαρμογές της Visual Basic μπορούν να εμπλουτιστούν με κάθε είδους ηχητικά εφέ. Υπάρχουν δύο τρόποι για την προσθήκη ηχητικών στοιχείων στις εφαρμογές. Ο πρώτος είναι με τη χρήση DLLs και ο δεύτερος με τη δημιουργία OLE αντικειμένων. Ο δεύτερος είναι οπωσδήποτε απλούστερος και κατά συνέπεια το εύρος δράσης του χρήστη σε ηχητικό επίπεδο είναι περιορισμένο.

7.3. Προσομοίωση κίνησης (animation)

Ανάμεσα στα πολλά ενδιαφέροντα χαρακτηριστικά που μπορεί να παρουσιάσει η Visual Basic στις εφαρμογές της, είναι η δυνατότητα δημιουργίας προσομοίωσης κίνησης (animation). Με τον όρο προσομοίωση κίνησης, εννοούμε την προβολή μιας σειράς από εικόνες, που προκαλούν στο χρήστη την αίσθηση της κίνησης. Υπάρχουν τρεις βασικοί τρόποι για τη δημιουργία προσομοίωσης κίνησης : frame-based, object και palette animation.

- Προσομοίωση κίνησης βασισμένη σε πλαίσια (frame-based animation), είναι εκείνη η τεχνική που στηρίζεται στην γρήγορη, συνεχή προβολή εικόνων που παρουσιάζουν μικρές διαφορές μεταξύ τους, δημιουργώντας κατά αυτό το τρόπο την αίσθηση της κίνησης.
- Προσομοίωση κίνησης με αντικείμενα (object animation), είναι εκείνη η τεχνική που χρησιμοποιούνται διαφορετικά, ανεξάρτητα μεταξύ τους, γραφικά αντικείμενα για τη δημιουργία της παρουσίασης. Αυτό το είδος προσομοίωσης κίνησης στηρίζεται αυστηρά στο χρόνο (time-based).
- Προσομοίωση κίνησης παλέτας (palette animation) είναι εκείνη η τεχνική που στηρίζεται στην αλλαγή των χρωμάτων μιας εικόνας, δημιουργώντας περισσότερο την αίσθηση της χρονικής κίνησης. Για παράδειγμα, αλλάζουν τα χρώματα μιας εικόνας, δίνοντας την εντύπωση πως πέρασε η μέρα και νύχτωσε.

Με τη Visual Basic, μπορούμε να δημιουργήσουμε και να χειριστούμε και τα τρία είδη προσομοίωσης κίνησης. Η ίδια η Visual Basic διαθέτει μία σειρά από εικόνες και εικονίδια, πολλά από τα οποία εμφανίζονται ανά ζευγάρια, και τα οποία μπορούν να χρησιμοποιηθούν σε απλές εφαρμογές προσομοίωσης κίνησης.

7.4. Video

Η Visual Basic μπορεί να συμπεριλάβει video στις εφαρμογές της, εμπλουτίζοντας τες ακόμη περισσότερο. Η χρήση video μπορεί να γίνει με τη χρησιμοποίηση του αντικειμένου OLE ή με το ActiveX εργαλείο MCI.