ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ
με VISUAL BASIC

ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
Β΄ ΚΟΙΝΟΤΙΚΟ ΠΛΑΙΣΙΟ ΣΤΗΡΙΞΗΣ

ΕΥΡΩΠΑΪΚΗ ΕΠΙΤΡΟΠΗ

ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ

ISBN 960-06-1042-8

ΟΕΔΒ

ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ
με VISUAL BASIC

ΒΙΒΛΙΟ ΜΑΘΗΤΗ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΩΝ
2ος ΚΥΚΛΟΣ ΣΠΟΥΔΩΝ

ΕΙΔΙΚΟΤΗΤΑ:
ΗΛΕΚΤΡΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ & ΔΙΚΤΥΩΝ

ΟΡΓΑΝΙΣΜΟΣ ΕΚΔΟΣΕΩΣ ΔΙΔΑΚΤΙΚΩΝ ΒΙΒΛΙΩΝ
ΑΘΗΝΑ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΟΜΑΔΑ ΣΥΓΓΡΑΦΗΣ
Βουτυράς Γεώργιος, Σύμβουλος Πληροφορικής, Msc Τηλεπικοινωνίες, Msc Πληροφορική
Βιδιαδάκης Ανδρέας, Φυσικός Ρ/Η και Η/Α, καθηγητής Δ/βάθμιας εκπαίδευσης ΠΕ-12
Ματζάκος Πέτρος, Φυσικός Ρ/Η και Η/Α, καθηγητής Δ/βάθμιας εκπαίδευσης ΠΕ-19
Σκουρλάς Χρήστος, καθηγητής ΤΕΙ Αθήνας

ΟΜΑΔΑ ΚΡΙΣΗΣ
Βλάχος Κωνσταντίνος, Ηλεκτρονικός, Δρ. Πανεπιστημίου Liverpool
Χάλαρης Ιωάννης, Δρ. Πληροφορικής, καθηγητής ΤΕΙ
Χελιώτης Παντελής, Ηλεκτρολόγος Μηχανικός, καθηγητής εφαρμογών ΤΕΙ Πειραιά

ΣΥΝΤΟΝΙΣΤΗΣ
Βιδιαδάκης Ανδρέας

ΕΠΙΜΕΛΕΙΑ ΕΚΔΟΣΗΣ
Βουτυράς Γεώργιος

ΗΛΕΚΤΡΟΝΙΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΚΕΙΜΕΝΟΥ
Αγραφιώτης Δημήτρης

ΓΛΩΣΣΙΚΗ ΕΠΙΜΕΛΕΙΑ
Κόκιου Ροδάνθη, καθηγήτρια Δ/θμιας εκπαίδευσης ΠΕ- 2

ΣΧΕΔΙΑΣΜΟΣ ΕΞΩΦΥΛΛΟΥ & ΠΡΟΕΚΤΥΠΩΣΗ
dimiourgies
Λ. ΠΕΝΤΕΛΗΣ 73, ΧΑΛΑΝΔΡΙ, ΤΗΛ. 6834738
www.dimiourgies.gr, e-mail:gmcz@aias.gr

ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ
Επιστημονικός Υπεύθυνος του Τομέα "ΗΛΕΚΤΡΟΝΙΚΩΝ",
Δρ ΧΑΡΑΛΑΜΠΟΣ ΔΗΜ. ΚΑΝΕΛΛΟΠΟΥΛΟΣ (PH.D)
(Σύμβουλος του Παιδαγωγικού Ινστιτούτου)

Με απόφαση της Ελληνικής Κυβερνήσεως τα διδακτικά βιβλία του Δημοτικού του Γυμνασίου και του Λυκείου τυπώνονται από τον Οργανισμό Εκδόσεων Διδακτικών Βιβλίων και διανέμονται δωρεάν.

ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ

Γ. Βουτυράς, Α. Βιδιαδάκης, Π. Ματζάκος, Χ. Σκουρλάς

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ
με VISUAL BASIC

ΒΙΒΛΙΟ ΜΑΘΗΤΗ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΩΝ
2ος ΚΥΚΛΟΣ ΣΠΟΥΔΩΝ

ΕΙΔΙΚΟΤΗΤΑ: ΗΛΕΚΤΡΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ & ΔΙΚΤΥΩΝ

ΟΡΓΑΝΙΣΜΟΣ ΕΚΔΟΣΕΩΣ ΔΙΔΑΚΤΙΚΩΝ ΒΙΒΛΙΩΝ
ΑΘΗΝΑ

Ευχαριστούμε την ομάδα κρίσης, τη Βίκυ Οικονόμου, την Άννα-Μαρία Σκουρτσή και το Νίκο Ευθυμίου που διάβασαν επιμελώς τα κείμενα και μας έκαναν εύστοχες παρατηρήσεις για τη βελτίωση του βιβλίου.

Πρόλογος

Το βιβλίο Προγραμματισμός Υπολογιστών με Visual Basic γράφτηκε για τους μαθητές της ειδικότητας, "Ηλεκτρονικός Υπολογιστικών Συστημάτων και Δικτύων" του 2ου κύκλου σπουδών του Τομέα Ηλεκτρονικών των Τεχνικών Επαγγελματικών Εκπαιδευτηρίων (ΤΕΕ).

Το βιβλίο είναι σύμφωνο με το προφίλ της ειδικότητας αυτής και το πρόγραμμα σπουδών του αντίστοιχου μαθήματος.

Ο βασικός σκοπός του βιβλίου είναι να καταστήσει το μαθητή ικανό, να δημιουργεί απλά προγράμματα. Το βιβλίο αυτό χρησιμοποιεί τη Visual Basic ως μέσο διδασκαλίας του προγραμματισμού. Δεν αποτελεί εγχειρίδιο αναφοράς της Visual Basic. Επεξεγεί εμμέσως τα πιο σημαντικά θέματα του προγραμματισμού χρησιμοποιώντας την πιο απλή και διαδεδομένη γλώσσα προγραμματισμού. Είναι με τέτοιο τρόπο γραμμένο, ώστε ο μαθητής να μπορεί να συνδυάσει τη θεωρία με την πράξη και από τα πρώτα κιόλας μαθήματα να μπορεί να γράφει προγράμματα.

Το μάθημα "Προγραμματισμός Υπολογιστών (Visual Basic)" είναι εργαστηριακό και διδάσκεται τρεις συνεχόμενες ώρες την εβδομάδα. Για το λόγο αυτό η ύλη του βιβλίου είναι οργανωμένη σε 28 μαθήματα. Προτείνεται η ολοκλήρωση ενός μαθήματος σε κάθε τρίωρο.

Μεταξύ των θεμάτων που περιγράφονται στο βιβλίο είναι: εισαγωγή στον προγραμματισμό, γνωριμία με το περιβάλλον της Visual Basic, προγραμματισμός με συμβάντα, δομημένος προγραμματισμός, γραφικά, πολυμέσα, πίνακες, υπορουτίνες και συναρτήσεις, αρχεία, βάσεις δεδομένων, παγίδευση λαθών.

Το βιβλίο αυτό, ως πρώτη προσπάθεια διδασκαλίας του προγραμματισμού μέσω της Visual Basic στη μέση εκπαίδευση, σίγουρα θα έχει δυσκολίες. Η κριτική από κάθε ενδιαφερόμενο είναι ευπρόσδεκτη και θα βοηθήσει στη βελτίωση του. Μπορείτε να στέλνετε τις παρατηρήσεις σας στο Παιδαγωγικό Ινστιτούτο (Μεσογείων 396, Αγία Παρασκευή, ΤΚ 15 341, για τον Τομέα Ηλεκτρονικών), στα τηλέφωνα ή φαξ 60 14 213 και 60 14 215 και στο e-mail: avid@pi-schools.gr

Οι συγγραφείς

Περιεχόμενα

Μάθημα 4
Η δημιουργία της διεπαφής

Μάθημα 5
Προγραμματισμός με συμβάντα

Μάθημα 6
Σταθερές, μεταβλητές και παραστάσεις

Μάθημα 6

Μαθημα 7
Συναρτήσεις

Μάθημα 8
Λογικές παραστάσεις-Δομές Επιλογής

Μάθημα 9
Δομές Επανάληψης

Μάθημα 10
Δομημένος προγραμματισμός-Εφαρμογές

Μάθημα 11
Εκσφαλμάτωση

Μάθημα 12
Δημιουργία Μενού

Μάθημα 13
Διαλογικά παράθυρα-Εφαρμογές με πολλά παράθυρα

Μάθημα 14
Πρόσθετα αντικείμενα ελέγχου-Εκτυπώσεις

Μάθημα 15
Γραφικά

Μάθημα 16
Κινούμενο σχέδιο-Τεχνική σύρε κι άσε

Μάθημα 17
Πολυμέσα

Μάθημα 18
Πίνακες (Συστοιχίες)

Βιβλιογραφία

1. Aitken P., Visual Basic Εξερευνώντας τον προγραμματισμό, Εκδόσεις ΙΩΝ, 1997.
2. Bloom S., Kiely D., Visual Basic Database How-to, Wait Group, 1995.
3. Champeaux D., Lea D., Faure P., Object-Oriented System Development, Hewlett-Packard Company, 1993.
4. Cormen T., Leiserson C., Rivest R., Introduction to Algorithms, Mc Graw-Hill, 1990.
5. Date C., An Introduction to Database Systems, Addison-Wesley, 1986.
6. Entsminger G., Secrets of the Visual Basic, Sams Publishing, 1994.
7. Garland S., Introduction to Computer Science with Applications, Addison-Wesley, 1986.
8. Hatmaker M., Butler W., κ.ά., Visual Basic OLE, Database and Controls, Waite Group Press, 1995.
9. Knuth, D., The art of computer programming, Addison Wesley Publishing Company, 1973.
10. Tremblay J., Richard B., An Introduction to Computer Science an algorithmic approach, McGraw-Hill Kogakusha, Ltd, 1979.
11. Miller T., Powell D., Using Delphi 3, Que, 1997.
12. Microsoft Access User's Guide DocNo. Microsoft Corporation DB53728-0294.
13. Morris S., Visual Basic, ο εύκολος τρόπος, Εκδόσεις Δίαυλος, 1998.
14. Neill G., Introduction to computer science, West Publishing Company, 1985.
15. Ross N., Visual Basic, Εκδόσεις Κλειδάριθμος, 1993.
16. Βουτυράς Γ., Basic, Αλγόριθμοι και εφαρμογές, Αθήνα 1992.
17. Βουτυράς Γ., Basic, Ειδικά θέματα και τεχνικές, Αθήνα 1991.
18. Γιαννακουδάκης Ε., Η αρχιτεκτονική των συστημάτων βάσεων δεδομένων, 1994.
19. Πολίτης Π., Γιαννόπουλος Η., Προγραμματισμός με τη Visual Basic 4.0 για Windows 95, Εκδόσεις Νέων Τεχνολογιών, 1997.

Μάθημα 1
Η διαδικασία ανάπτυξης λογισμικού

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να κατονομάζουν και να περιγράφουν τις φάσεις ανάπτυξης του λογισμικού.
- Να επιχειρηματολογούν για την αναγκαιότητα της ανάλυσης του προβλήματος.
- Να αναφέρουν τις ενέργειες που γίνονται σε κάθε φάση.
- Να περιγράφουν τους τρόπους διασύνδεσης των φάσεων.
- Να περιγράφουν τον αναβαθμισμένο ρόλο του προγραμματιστή.

Στα ***υπολογιστικά συστήματα*** (***computer systems***) υπάρχει ένας αρμονικός συνδυασμός της υλικής τους φύσης (το ***υλικό - hardware***) με την άυλη φύση τους (το ***λογισμικό - software***). Το λογισμικό έχει την ανάγκη του υλικού για να αποκτήσει υπόσταση αλλά, από την άλλη πλευρά, είναι αυτό που δίνει πνοή στο υλικό και το καθιστά εκμεταλλεύσιμο. Όμως το κύριο στοιχείο που χαρακτηρίζει αυτήν την αρμονική σχέση είναι η μεγάλη ευελιξία με την οποία γίνεται η αλλαγή χρήσης ενός υπολογιστικού συστήματος. Με διαφορετικό λογισμικό εφαρμογών το ίδιο υπολογιστικό σύστημα χρησιμοποιείται για την εκτέλεση διαφορετικών μορφών εργασίας. Ήδη, έχουμε δει από το γυμνάσιο ότι μπορούμε να γράψουμε κείμενο με έναν επεξεργαστή κειμένου, να κάνουμε υπολογισμούς με ένα φύλλο εργασίας, να οργανώνουμε στοιχεία σε μια βάση δεδομένων, να αποκτήσουμε γνώσεις πλοηγώντας μέσα σε εκπαιδευτικό λογισμικό πολυμέσων, να συνδεθούμε στο Internet με σκοπό να αντλήσουμε πληροφορίες από τον Παγκόσμιο Ιστό, να επικοινωνήσουμε με άλλα άτομα χρησιμοποιώντας το Ηλεκτρονικό Ταχυδρομείο κ.ά. χρησιμοποιώντας τον ίδιο ακριβώς υπολογιστή.

Ο όρος υπολογιστικό σύστημα είναι πολύ γενικότερος του όρου υπολογιστής. Ο όρος υπολογιστής αναφέρεται σε συγκεκριμένο υλικό των υπολογιστικών συστημάτων.

Τα τελευταία χρόνια τόσο η εξέλιξη του υλικού όσο και η εξέλιξη του λογισμικού έχει πάρει εκρηκτικές διαστάσεις. Το υλικό γίνεται όλο και ταχύτερο στην εκτέλεση λειτουργιών, όλο και μικρότερο σε ότι αφορά τον όγκο, και αποκτά όλο και μεγαλύτερες δυνατότητες αποθήκευσης στοιχείων. Το λογισμικό έχει επεκταθεί σε κάθε κλάδο των ανθρώπινων δραστηριοτήτων και γίνεται όλο και πιο φιλικό προς το χρήστη. Όμως η έρευνα και η παραγωγή νέων προϊόντων υλικού παρουσιάζει διαφορετικό βαθμό δυσκολίας από αυτήν του λογισμικού. Η ανάπτυξη του υλικού απαιτεί πολύ εξειδικευμένο και πανάκριβο εξοπλισμό και γίνεται σήμερα από επιστήμονες σε ερευνητικά κέντρα εταιρειών και πανεπιστημιακών ιδρυμάτων, που διαθέτουν υψηλή τεχνογνωσία. Αντίθετα, η ανάπτυξη του λογισμικού μπορεί να γίνει με συμβατικά μέσα και χαμηλό κόστος, ακόμη και από ένα μόνο προγραμματιστή, απαιτεί όμως μεθοδικότητα, προσήλωση σε προκαθορισμένη λογική, συνεχή ενημέρωση για τις δυνατότητες των σύγχρονων προγραμματιστικών εργαλείων ανάπτυξης και παρακολούθηση των σύγχρονων τάσεων σχεδίασης.

Η συναρμολόγηση δεν αποτελεί διαδικασία ανάπτυξης υλικού.

Οι φάσεις ανάπτυξης λογισμικού

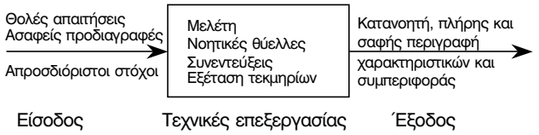
Η διαδικασία ανάπτυξης λογισμικού δεν είναι μια μονολιθική διαδικασία κατά την οποία σε μια μόνο φάση δημιουργούνται ως δια μαγείας τα προγράμματα. Είναι μια διαδικασία στην οποία διακρίνουμε ξεχωριστές φάσεις, καθεμιά από τις οποίες παίζει το δικό της σημαντικό ρόλο, εμφανίζει τις δικές της ιδιαιτερότητες και απαιτεί τη δική της μεθοδολογία υλοποίησης. Συγκεκριμένα, οι φάσεις αυτές είναι οι εξής:

- Ανάλυση του προβλήματος.
- Σχεδίαση της λύσης.
- Υλοποίηση των προγραμμάτων.
- Έλεγχος συμμόρφωσης στις προδιαγραφές και διόρθωση λαθών.
- Λειτουργία και Υποστήριξη.

Ο χρόνος, οι πόροι (όχι μόνο οι οικονομικοί αλλά και οι πόροι ανθρώπινου δυναμικού και υπολογιστικών συστημάτων) καθώς και η βαρύτητα που δίνεται σε κάθε φάση εξαρτάται από το μέγεθος και το είδος του προβλήματος.

Ανάλυση του προβλήματος

Πρωταρχικός στόχος της ***ανάλυσης*** (***analysis***) είναι να διαπιστωθεί κατά πόσον το πρόβλημα, που ζητείται να λυθεί, είναι επιλύσιμο με υπολογιστή. Η ***εφικτότητα*** (***feasibility***) υλοποίησης είναι το πρώτο πράγμα που πρέπει να ελέγχεται πριν γίνει οποιαδήποτε άλλη ενέργεια. Στη φάση της ανάλυσης γίνεται προσπάθεια καταγραφής κάθε υπάρχουσας πληροφορίας για το πρόβλημα. Περιγράφονται οι ***απαιτήσεις*** (***requirements***) της εφαρμογής, επισημαίνονται οι ιδιαιτερότητες, απομακρύνονται οι επουσιώδεις λεπτομέρειες που περιπλέκουν τη λύση, σκιαγραφείται ο τρόπος λύσης, υποδεικνύονται τα σημεία στα οποία απαιτείται ιδιαίτερη προσοχή και περιγράφονται οι συνθήκες κάτω από τις οποίες είναι δυνατή η υλοποίηση. Ακόμα αναλύονται στοιχεία που αφορούν το κόστος, τρόποι διείσδυσης του τελικού προϊόντος στην αγορά, μέθοδοι διανομής του κ.ά.



Σχήμα 1-1. *Δεδομένα, τρόπος επεξεργασίας και αποτελέσματα της ανάλυσης*

Η ανάλυση είναι μια καθαρά νοητική διαδικασία, που είναι δύσκολο να περιγραφεί θεωρητικά και πολύ δύσκολο να τυποποιηθεί σε μεγάλο βαθμό. Θεωρείται η πιο σημαντική φάση μιας και θέτει τα θεμέλια για όλες τις επόμενες φάσεις. Αν το έργο είναι μεγάλο πραγματοποιείται από εξειδικευμένους ***αναλυτές συστημάτων*** (***system analysts***). Οι αναλυτές συστημάτων είναι άτομα που διαθέτουν μεγάλη εμπειρία από προηγούμενα παρόμοια έργα και διορατικότητα, μιας και πρέπει να προβλέπουν και να περιγράφουν καταστάσεις και προβλήματα, που μπορεί να παρουσιαστούν σε έργα τα οποία αντιμετωπίζονται για πρώτη φορά. Όμως, αν το έργο είναι μικρό, έχει προφανείς λύσεις και δεν απαιτεί οικονομικές αναλύσεις, τη δουλειά του αναλυτή μπορεί να την κάνει και ένας έμπειρος προγραμματιστής.

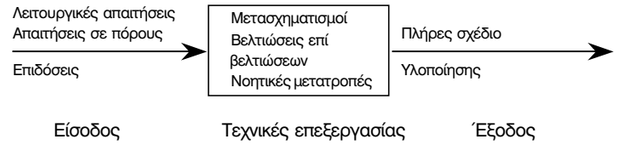
Το μεγαλύτερο πρόβλημα στη φάση της ανάλυσης είναι η εσφαλμένη εκτίμηση του μεγέθους του έργου και του προβλήματος. Πολλοί βλέποντας εκ των υστέρων τις δυσκολίες που παρουσιάζει μια διεξοδική ανάλυση και φοβούμενοι ότι θα χάσουν πολύ χρόνο, την προσπερνούν χωρίς να δώσουν την απαιτούμενη προσοχή, με αποτέλεσμα να μην τίθενται οι σωστές προδιαγραφές. Οι αβλεψίες και τα λάθη που πραγματοποιούνται σε αυτή τη φάση, ανακαλύπτονται στη συνέχεια με πολύ μεγάλη δυσκολία και προκαλούν αναθεωρήσεις και μακρές καθυστερήσεις στις υπόλοιπες φάσεις.

Σχεδίαση της λύσης

Η φάση της ***σχεδίασης και ανάπτυξης της λύσης*** (***solution development***) μπορεί να χαρακτηριστεί ως η κατ' εξοχήν δημιουργική φάση. Είναι η φάση που απαιτεί όχι μόνο τεχνικές γνώσεις σε θέματα υπολογιστικών συστημάτων, αλλά και γνώσεις πάνω στον τομέα του προβλήματος καθώς και γνώσεις διοικητικοοικονομικής φύσης. Στα μεγάλα έργα, οι ***σχεδιαστές*** (***designers***) πρέπει να σχεδιάσουν όχι μόνο τη λύση του προβλήματος αλλά και τον τρόπο με τον οποίο θα δουλέψουν οι ομάδες εργασίας, να κάνουν χρονοπρογραμματισμό των επιμέρους εργασιών, κατανομή των πόρων και οικονομικό προγραμματισμό.

Σ' αυτήν τη φάση το πρόβλημα χωρίζεται σε επιμέρους προβλήματα, που είναι δυνατόν να επιλυθούν πιο εύκολα και καθορίζεται ο τρόπος της μεταξύ τους συσχέτισης. Κάθε πρόβλημα χωρίζεται πάλι με τη σειρά του σε άλλα κ.ο.κ. μέχρι του βαθμού που να μην μπορεί να γίνει περαιτέρω διαίρεση του. Ταυτόχρονα με τη διαδοχική κατάτμηση, δημιουργείται ένα διάγραμμα στο οποίο γίνεται η περιγραφή της σύνδεσης των προβλημάτων. Το διάγραμμα αυτό μας βοηθά να δούμε την όλη εφαρμογή σφαιρικά, να διακρίνουμε όλα της τα επίπεδα και να καταστρώσουμε πλάνο λύσης. Στο πλάνο λύσης για κάθε τελικό πρόβλημα ή επινοείται, αν δεν υπάρχει, ή επιλέγεται ένας από τους γνωστούς ***αλγορίθμους*** (***algorithms***) που το επιλύει. Επίσης, οργανώνονται τα δεδομένα σε ***δομές δεδομένων*** (***data structures***), ώστε να αποτελούν λογικές και εύληπτες ενότητες.

Ο αλγόριθμος είναι μια σειρά ενεργειών, που περιγράφουν τη διαδικασία εκτέλεσης μιας εργασίας ή τον τρόπο επίλυσης ενός προβλήματος.

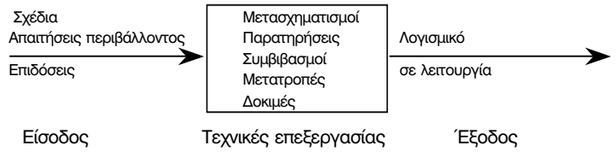


Σχήμα 1-2. *Δεδομένα, τρόπος επεξεργασίας και αποτελέσματα της σχεδίασης*

Η δεξιότητα διάκρισης των επιμέρους προβλημάτων, η εύρεση λύσεων, η δημιουργία ή η επιλογή αλγορίθμων, καθώς και η δημιουργία δομών δεδομένων δεν είναι κάτι που διδάσκεται από τη μια στιγμή στην άλλη, αλλά κάτι που αποκτάται με τον καιρό μετά από συμμετοχή σε ομάδες σχεδίασης και μετά από μελέτη πολλών ειδικών περιπτώσεων.

Υλοποίηση προγραμμάτων

Στην τρίτη φάση, την ***υλοποίηση*** (***implementation***) της λύσης, γίνεται η δημιουργία και η σχεδίαση των φορμών επικοινωνίας χρήστη - υπολογιστή, η διάταξή τους στην οθόνη και η ***κωδικοποίηση*** (***coding***) των αλγορίθμων και των δομών δεδομένων σε προγράμματα σύμφωνα με τους κανόνες μιας γλώσσας προγραμματισμού. Από πολλούς θεωρείται ότι αυτή η διαδικασία είναι μια καθαρά μηχανική δουλειά, που διέπεται από αυστηρούς τεχνικούς κανόνες, συγκεκριμένους κανόνες αισθητικής και κανόνες που διέπουν το συντακτικό της γλώσσας προγραμματισμού. Τα πράγματα όμως δεν είναι ακριβώς έτσι. Σημαντικό ρόλο στη σχεδίαση των οθονών και την κωδικοποίηση παίζει το στυλ προγραμματισμού. Ο στόχος δεν είναι να δημιουργηθεί μόνο λογισμικό το οποίο να πραγματοποιεί αυτά που έχουν προδιαγραφεί, αλλά λογισμικό που να μπορεί εύκολα να επεκταθεί και να συντηρηθεί, λογισμικό που να είναι λειτουργικό από την πλευρά του χρήστη, λογισμικό στο οποίο να μπορούν εύκολα να βρεθούν τα πιθανά σφάλματα, με άλλα λόγια ***ποιοτικό λογισμικό***. Οι καλοί προγραμματιστές έχουν πολλά να προσφέρουν σε αυτή τη φάση, που πολλές φορές απαιτεί αναθεώρηση στοιχείων τόσο της φάσης της σχεδίασης όσο και της φάσης της ανάλυσης.



Σχήμα 1-3. *Δεδομένα, τρόπος επεξεργασίας και αποτελέσματατης υλοποίησης*

Στη φάση της υλοποίησης αρχίζουν να φαίνονται όλες οι αδυναμίες των προηγουμένων φάσεων. Είναι η φάση που χαρακτηρίζεται από τη φράση "Ιδού η Ρόδος, ιδού και το πήδημα". Αποτελεί τη φάση στην οποία τα λόγια που έχουν καταγραφεί στην ανάλυση και στη σχεδίαση πρέπει να γίνουν πράξη. Πολλές φορές ένα τμήμα της υλοποίησης γίνεται παράλληλα με τις δύο προηγούμενες φάσεις, σε μορφή ***πρωτοτύπου***, για να διαπιστωθεί κατά πόσο είναι εφικτές οι λύσεις που προτείνονται.

Στο μύθο του Αισώπου "Ανήρ κομπαστής", προκαλείται κάποιος που καυχόταν ότι είχε πραγματοποιήσει ένα μεγάλο άλμα στη νήσο Ρόδο, να το ξαναπραγματοποιήσει.

Έλεγχος προδιαγραφών και διόρθωση λαθών

Στη φάση των ***δοκιμών*** (***testing***) γίνονται έλεγχοι κατά πόσο το λογισμικό που έχει παραχθεί βρίσκεται μέσα στις προδιαγραφές της ανάλυσης (***αξιολόγηση*** του προϊόντος) και ανιχνεύονται κατά το δυνατόν τα υπάρχοντα λάθη. Αυτή η φάση μπορεί να εκτελείται παράλληλα με την κωδικοποίηση. Οι έμπειροι προγραμματιστές συνηθίζουν να κάνουν ελέγχους του κώδικα που παράγουν, όταν τελειώσουν μια μικρή ενότητα, και δεν αφήνουν όλους τους ελέγχους για το τέλος. Όμως, αν και ο επιμέρους έλεγχος μπορεί να εξασφαλίσει την ορθότητα κάποιων λεπτομερών προδιαγραφών και την απομάκρυνση των επιμέρους λαθών, δε συνεπάγεται και την κάλυψη γενικότερων προδιαγραφών ή την απομάκρυνση των λαθών που μπορεί να προκύψουν από το συνδυασμό των ενοτήτων σε μεγαλύτερες ενότητες.

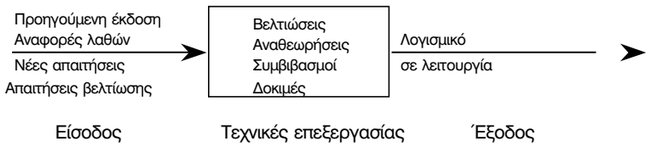
Οι δοκιμές που γίνονται στο λογισμικό δεν αφορούν τις εσωτερικές του μόνο λειτουργίες. Το πρόγραμμα συνεργάζεται τόσο με το λογισμικό συστήματος όσο και με το υλικό. Επιβάλλεται λοιπόν να γίνουν έλεγχοι του κατά πόσον το λογισμικό μπορεί να λειτουργήσει το ίδιο ικανοποιητικά κάτω από διαφορετικές ρυθμίσεις λογισμικού και συνθέσεις υλικού (***έλεγχος μεταφερσιμότητας - portability test***). Ακόμα, το λογισμικό το χειρίζονται άνθρωποι, οι οποίοι μπορεί να υποπέσουν σε λάθος χειρισμούς εξαιτίας της μη λειτουργικής σχεδίασής του (***δοκιμές ανθρωπίνων παραγόντων - human factors tests***). Για παράδειγμα, μια διαδικασία που διαγράφει στοιχεία πρέπει πάντα να ρωτά με ένα διαλογικό παράθυρο, αν ο χρήστης πράγματι επιθυμεί τη διαγραφή. Η έλλειψη μιας τέτοιας ερώτησης αποτελεί σφάλμα αφού ένας χρήστης μπορεί να καλέσει τη διαδικασία από απροσεξία.

Πάντως αν και ένα πρόγραμμα μπορεί να ελεγχθεί για το αν καλύπτει κάποιες προδιαγραφές που έχουν τεθεί, δεν μπορεί να ελεγχθεί πλήρως για το αν περιέχει λάθη. Για την ανίχνευση των λαθών οι προγραμματιστές χρησιμοποιούν ξεχωριστό όρο, τον όρο ***εκσφαλμάτωση*** (***debugging***). Κατά την εκσφαλμάτωση γίνεται προσπάθεια να ανιχνευθούν όσο το δυνατόν περισσότερα από τα υπάρχοντα λάθη. Ένας επιτυχής έλεγχος μπορεί να δείξει ότι δεν υπάρχουν λάθη μέσα στο πρόγραμμα για τις συνθήκες κάτω από τις οποίες αυτός γίνεται, όχι όμως ότι δεν υπάρχουν καθόλου λάθη. Μόνο με ***εξαντλητικό έλεγχο*** (***exhaustive test***) είναι δυνατή η απομάκρυνση των λαθών και κάτι τέτοιο δεν είναι δυνατόν να γίνει ούτε στο πιο μικρό πρόγραμμα, μιας και δεν μπορούν να προβλεφθούν όλες οι καταστάσεις κάτω από τις οποίες θα εκτελείται ένα πρόγραμμα και όλες οι τιμές δεδομένων που είναι πιθανό να δοθούν. Το μόνο λοιπόν που μπορεί να γίνει είναι να ανιχνευθούν όσο το δυνατόν περισσότερα λάθη.

Το ότι ένα πρόγραμμα πέρασε όλες τις δοκιμές χωρίς λάθη δε σημαίνει ότι είναι απαλλαγμένο λαθών. Με άλλα λόγια η απουσία ανωμαλιών στην εκτέλεση του προγράμματος δε συνεπάγεται τη μη ύπαρξή λαθών.

Λειτουργία και Υποστήριξη

Η τελευταία φάση έχει και τη μεγαλύτερη διάρκεια. Δεν περιλαμβάνει μόνο την έναρξη της λειτουργίας του λογισμικού αλλά και την ***υποστήριξή*** (***maintenance***) του. Η υποστήριξη πρέπει να γίνεται σε όλη τη διάρκεια ζωής του λογισμικού. Συνεχώς, νέες απαιτήσεις εισάγονται, βελτιώσεις απαιτούνται και λάθη που δεν είχαν ανιχνευθεί επιβάλλεται να διορθωθούν.



Σχήμα 1-4. *Δεδομένα, τρόπος επεξεργασίας και αποτελέσματα της υποστήριξης*

Η μορφή μιας εργασίας δεν παραμένει κατά κανόνα η ίδια, αλλά συνεχώς εξελίσσεται. Το λογισμικό λοιπόν που εξυπηρετεί αυτή την εργασία ή θα πρέπει να αντικατασταθεί με άλλο που να ικανοποιεί τις σύγχρονες απαιτήσεις ή να προσαρμοστεί στις νέες καταστάσεις. Επίσης, δεν είναι λίγες οι φορές που η κάλυψη των αναγκών μιας εφαρμογής γίνεται σταδιακά. Εγκαθίσταται στην αρχή ένα πρόγραμμα που επιτελεί ένα τμήμα των όλων λειτουργιών μιας διαδικασίας και στη συνέχεια γίνεται η βελτίωσή του για να καλύψει και τις υπόλοιπες. Τέλος, όπως ήδη έχουμε αναφέρει, δεν είναι δυνατόν να εξασφαλιστεί ότι μέσα σε ένα πρόγραμμα δεν θα υπάρχουν λάθη. Έτσι, παρόλο που το πρόγραμμα έχει παραδοθεί απαιτείται η διόρθωση των λαθών που εμφανίζονται κατά τη λειτουργία του.

Τεκμηρίωση

Στη διαδικασία ανάπτυξης του λογισμικού ανήκει και η ***τεκμηρίωση*** (***documentation***). Η τεκμηρίωση δεν αποτελεί ξεχωριστή φάση ανάπτυξης αλλά διαδικασία που πραγματοποιείται σε όλο τον κύκλο της ζωής του λογισμικού. Στην τεκμηρίωση γίνεται η συγκέντρωση, η επεξεργασία και η κατάταξη όλων των σχετικών στοιχείων και σχολίων που αφορούν το πρόβλημα. Η λεπτομερής περιγραφή του προβλήματος, οι αλγόριθμοι που επιλέχθηκαν, οι τρόποι οργάνωσης των δεδομένων σε δομές, οι αδυναμίες ή τα προτερήματα της

ακολουθούμενης μεθοδολογίας, τα τεχνάσματα που χρησιμοποιήθηκαν κατά την κωδικοποίηση καταγράφονται και τοποθετούνται μέσα στον ***κατασκευαστικό φάκελο*** της εφαρμογής (***εξωτερική τεκμηρίωση - external documentation***) ή μέσα στον κώδικα του προγράμματος σε μορφή σχολίων (***εσωτερική τεκμηρίωση - internal documentation***).

Πρέπει να γίνει συνείδηση σε όλους όσοι εμπλέκονται σε ένα έργο ότι η τεκμηρίωση δεν είναι πάρεργο, που εκτελείται στο περιθώριο της διαδικασίας ανάπτυξης του λογισμικού, αλλά η ραχοκοκκαλιά πάνω στην οποία κτίζεται και στηρίζεται το λογισμικό. Από τη φάση της ανάλυσης πρέπει να αρχίζει η λεπτομερής και γραπτή περιγραφή. Η προφορική περιγραφή δεν είναι μόνο νεφελώδης, αλλά λησμονείται και εύκολα. Η τεκμηρίωση δε βοηθά μόνο τους συνεργάτες μας να κατανοήσουν μια διαδικασία για να μπορέσουν να την εκτελέσουν και να τη συντηρήσουν αλλά και εμάς τους ίδιους για να θυμηθούμε τι ακριβώς είχε ζητηθεί να γίνει, πώς ακριβώς το πραγματοποιήσαμε και τι προβλήματα συναντήσαμε. Η τεκμηρίωση δε βοηθά μόνο στην υποστήριξη του υπάρχοντος λογισμικού αλλά και στην ανάπτυξη νέου παρεμφερούς λογισμικού.

Η σειρά εκτέλεσης των φάσεων ανάπτυξης λογισμικού

Κάποιες από τις φάσεις ανάπτυξης πρέπει να ολοκληρωθούν πλήρως, πριν γίνει οτιδήποτε στις επόμενες, ενώ κάποιες άλλες μπορούν να συνδυαστούν ή και να ενοποιηθούν. Όταν το έργο είναι πολύ απλό, όλη η ανάλυση, η σχεδίαση, η ανάπτυξη και οι δοκιμές μπορούν να γίνουν σε μία μόνο προσπάθεια. Για παράδειγμα, η δημιουργία ενός προγράμματος, το οποίο υπολογίζει το ρεύμα που διαρρέει μιαν αντίσταση από την τάση στα άκρα της, μπορεί να γίνει χωρίς μεγάλη προετοιμασία και με πολύ μικρή προσπάθεια. Αντίθετα, σε ένα σύνθετο έργο, για παράδειγμα στη διαχείριση μιας αποθήκης ηλεκτρονικού υλικού, απαιτείται κατακερματισμός των φάσεων ανάπτυξης σε ακόμη μικρότερα τμήματα, τα οποία υλοποιούνται, ή το ένα μετά το άλλο, ή παράλληλα ανάλογα με τη μεταξύ τους εξάρτηση.

Μεταβαίνοντας από μια φάση της διαδικασίας ανάπτυξης σε μια άλλη, δε σημαίνει ότι δε θα χρειαστεί να επιστρέψουμε πάλι στην προηγούμενη. Για παράδειγμα, πολλές φορές απαιτείται η σχεδίαση και η υλοποίηση ενός πειραματικού μοντέλου στη φάση της ανάλυσης πολύ πριν αρχίσει η σχεδίαση του κανονικού προϊόντος. Ακόμα, σε κάποια έργα με πολλά στάδια, όταν τελειώνει η υλοποίηση του ενός σταδίου αρχίζει η ανάλυση του άλλου μιας και το δεύτερο εξαρτάται ισχυρά από τον τρόπο συμπεριφοράς και την επιτυχία του πρώτου.

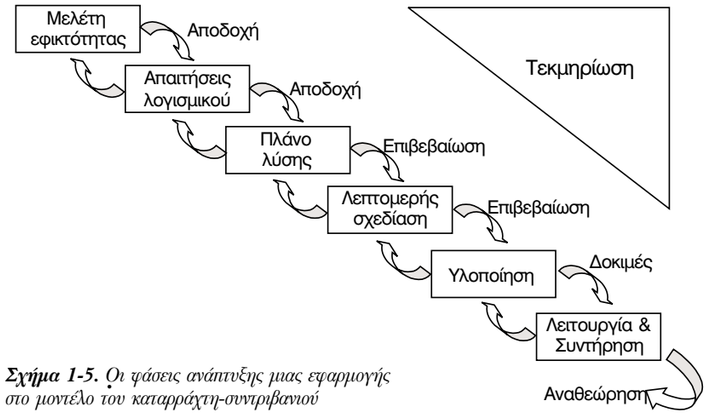
Πολύ συχνά, οι φάσεις και τα επιμέρους τμήματά τους επαναλαμβάνονται ξανά και ξανά, πριν θεωρηθεί ότι προέκυψε κάποιο ποιοτικό αποτέλεσμα. Δεν είναι λίγες οι φορές που πρέπει να γίνει αναθεώρηση της εργασίας που έχει πραγματοποιηθεί σε μια φάση, γιατί έχουμε βρεθεί σε αδιέξοδο σε μια επόμενή της. Όσο προσεκτικοί κι αν είμαστε δεν είναι δυνατόν να εντοπιστούν όλες οι παράμετροι ενός πολύπλοκου προβλήματος και δεν είναι δυνατή η πλήρης πρόβλεψη της εξέλιξης της τεχνολογίας. Επίσης, σύγχρονες τάσεις, στις οποίες γίνεται προσπάθεια μείωσης του κόστους και του χρόνου ανάπτυξης, προτείνουν τη συνδυασμένη εκτέλεση κάποιων φάσεων. Σήμερα κατά τη σχεδίαση, δε δημιουργούνται λεπτομερειακά διαγράμματα ροής ούτε γράφεται ψευδοκώδικας. Ο προγραμματιστής χρησιμοποιεί την οθόνη του υπολογιστή του σαν χαρτί για να σχεδιάζει λύσεις και αλγορίθμους εκφράζοντας τις ιδέες του απευθείας σε γλώσσα προγραμματισμού.

Για τη σύνδεση των φάσεων ανάπτυξης λογισμικού έχουν προταθεί κατά καιρούς πολλά μοντέλα. Δύο από τα πιο κλασικά είναι το ***μοντέλο του καταρράχτη-συντριβανιού*** (***waterfall-fountain model***) και το ***μοντέλο του σαλίγκαρου*** ή ***σπειροειδές μοντέλο*** (***spiral model***).

Το μοντέλο του καταρράχτη - συντριβανιού

Το απλό μοντέλο του καταρράχτη στηρίζεται στην υπόθεση, ότι η σχεδίαση και η ανάπτυξη του λογισμικού μπορεί να συντίθενται από τόσο ακριβείς, σαφείς και περιγράψιμες λειτουργικές διαδικασίες και αρχές όσο και η σχεδίαση και η ανάπτυξη ενός μηχανολογικού έργου (π.χ. η κατασκευή ενός σπιτιού). Με βάση αυτή την υπόθεση, οι φάσεις μπορούν να τοποθετηθούν η μία μετά την άλλη.

Η πληροφορική δανείστηκε το μοντέλο καταρράχτη από άλλους τεχνικούς χώρους με μεγαλύτερη ιστορία. Όμως τα πρότυπα αυτών των χώρων δεν μπορούν να εφαρμοστούν επακριβώς στο χώρο του λογισμικού, που είναι χώρος στον οποίο υπεισέρχονται και άϋλοι παράγοντες. Για παράδειγμα, η κατασκευή ενός σπιτιού δε μοιάζει με την παραγωγή ενός προγράμματος. Η κατασκευή του σπιτιού στηρίζεται σε σαφώς ορισμένες αρχές, κανόνες και νόμους της στατικής και της δυναμικής, που εφαρμόζονται στερεότυπα και χωρίς παρεκκλίσεις. Για το λογισμικό όμως, δεν υπάρχουν κάποιοι αυστηροί και υποχρεωτικοί από τη φύση νόμοι και κανόνες. Κυρίαρχο ρόλο παίζει η ικανοποίηση του τελικού χρήστη, του οποίου οι απαιτήσεις αλλάζουν πολύ γρήγορα. Επίσης, η επιτυχία μετριέται σε μεγάλο βαθμό από τη λειτουργικότητα της εφαρμογής, που δεν είναι σταθερός και μετρήσιμος παράγοντας. Το μεγάλο πρόβλημα είναι ότι το αντικείμενο δεν είναι σαφώς καθορισμένο και μεταβάλλεται κατά τη φάση της ανάπτυξης και της υλοποίησής του ώστε να ικανοποιήσει τις τρέχουσες ανάγκες και να προφτάσει τον καλπάζοντα ανταγωνισμό. Κατεβαίνοντας λοιπόν την κλίμακα των φάσεων εμφανίζονται νέα δεδομένα, τα οποία μας υποχρεώνουν να αλλάξουμε γνώμη και να μεταβάλλουμε αποφάσεις που είχουμε πάρει σε προγενέστερες φάσεις. Η συμπεριφορά αυτή μετασχηματίζει το απλό μοντέλο σε μοντέλο καταρράκτη - συντριβανιού.



Σχήμα 1-5. *Οι φάσεις ανάπτυξης μιας εφαρμογής στο μοντέλο του καταρράχτη-συντριβανιού*

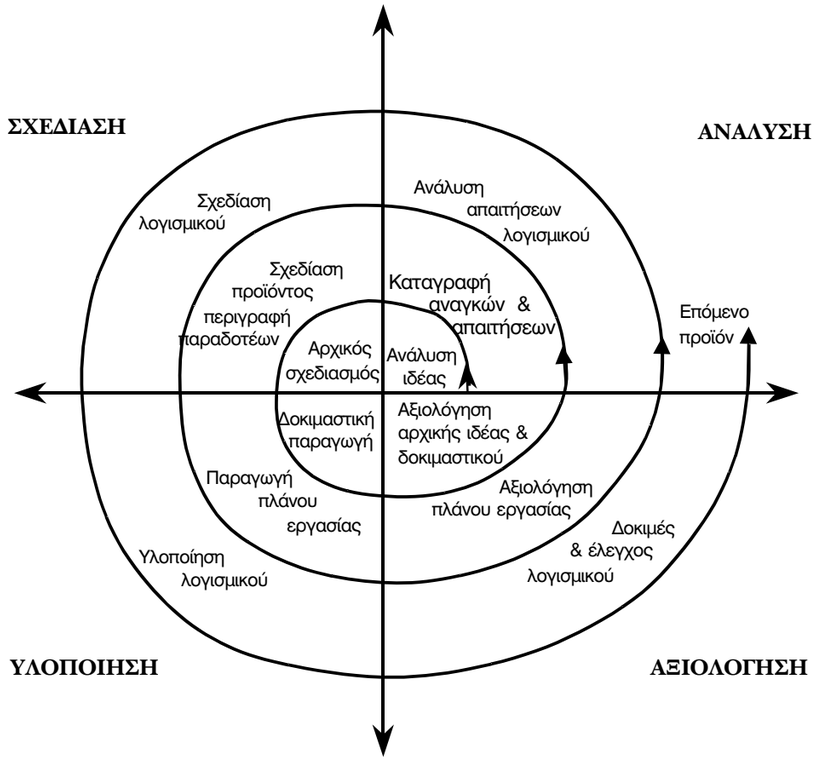
Στο σχήμα 1-5, τα προς τα πάνω βέλη δηλώνουν την αλληλεπίδραση μεταξύ των γειτονικών φάσεων και την περίπτωση επιστροφής σε μια προηγούμενη φάση για αναθεώρησή της μετά από την εμφάνιση προβλημάτων, που δεν είχαν προβλεφθεί. Συχνά, οι αλλαγές σε κάποια φάση με στόχο την προσαρμογή του προϊόντος σε νέες απαιτήσεις, επηρεάζουν όχι μόνο τις φάσεις που ακολουθούν, αλλά και τις φάσεις που προηγούνται.

Το μοντέλο του σαλίγκαρου

Το μοντέλο αυτό βασίζεται στην παρατήρηση, ότι οι φάσεις περιέχουν διαδικασίες που απέχουν χρονικά και στο μεταξύ μεσολαβούν διαδικασίες άλλων φάσεων. Η μη χρονική γειτνίαση των διαδικασιών των φάσεων μας αναγκάζει να επανερχόμαστε ξανά και ξανά στην ίδια φάση, αφού φυσικά ασχοληθούμε ενδιάμεσα με τις διαδικασίες άλλων φάσεων.

Στο μοντέλο του σαλίγκαρου οι φάσεις και οι διαδικασίες τοποθετούνται πάνω σε σπείρες. Στο πρώτο τεταρτημόριο κάθε σπείρας γίνεται η ανάλυση που αφορά τον ορισμό του αντικειμένου, την αναγνώριση των περιορισμών και την ανίχνευση των εναλλακτικών λύσεων για ό,τι πρόκειται να υλοποιηθεί και παραδοθεί στο κλείσιμο της σπείρας. Στο δεύτερο τεταρτημόριο γίνεται η σχεδίαση των διαδικασιών που έχουν αναλυθεί και περιγραφεί στο πρώτο τεταρτημόριο, στο τρίτο τεταρτημόριο γίνεται η υλοποίηση και στο τέταρτο η αξιολόγηση του έργου που έχει παραχθεί. Μετά ακολουθεί νέα σπείρα που ξεκινά πάλι με ανάλυση, συνεχίζει με σχεδίαση, υλοποίηση, αξιολόγηση κ.ο.κ.

Το πλήθος των σπειρών είναι απεριόριστο. Στο κλείσιμο κάθε σπείρας γίνεται η παράδοση ενός προϊόντος, που θα αποτελέσει το σημείο εκκίνησης και το σκελετό του προϊόντος της επόμενης σπείρας. Η διαδικασία επαναλαμβάνεται μέχρι να προκύψει το επιθυμητό



Σχήμα 1-6. *Οι φάσεις ανάπτυξης μιας εφαρμογής στο μοντέλο του σαλίγκαρου. Στον κύκλο ανάπτυξης της εφαρμογής μια διαδικασία μπορεί να εκτελεστεί ξανά και ξανά και σε μη γειτονικές χρονικές στιγμές.*

κάτι το οποίο μπορεί να μην γίνει ποτέ, μιας και τα προϊόντα λογισμικού δεν είναι στατικά αλλά βελτιώνονται και εξελίσσονται συνεχώς. Εννοείται φυσικά, ότι η προσθήκη κάθε νέας σπείρας αυξάνει το συνολικό κόστος και το χρόνο παράδοσης.

Ο ρόλος του προγραμματιστή

Σήμερα, τα μέσα ανάπτυξης λογισμικού έχουν εξελιχθεί σε τέτοιο βαθμό, ώστε να είναι δυνατή η χρήση του υπολογιστή και στη φάση της σχεδίασης και στη φάση της ανάλυσης (π.χ. αντικειμενοστρεφής προγραμματισμός και εργαλεία CASE). Απαιτούνται λοιπόν ιδιαίτερα προσόντα από τους σημερινούς προγραμματιστές. Ο ρόλος του προγραμματιστή έχει αναβαθμιστεί και δε θεωρείται ένας απλός "κωδικοποιητής εντολών", όπως συνέβαινε πριν από χρόνια. Ο προγραμματιστής συμμετέχει σε όλες τις φάσεις που περιγράψαμε. Αναλύει τα στοιχεία που του περιγράφει ο αναλυτής συστήματος, σχεδιάζει από κοινού με τον αναλυτή τρόπους λύσης, υλοποιεί το λογισμικό, δοκιμάζει και αξιολογεί τα προγράμματά του, αναλαμβάνει τη συντήρησή τους.

Στη φάση της ανάλυσης του προβλήματος, ο προγραμματιστής προσπαθεί να κατανοήσει το πρόβλημα και την εφαρμογή. Τα προβλήματα για τα οποία μας ζητείται να δημιουργηθεί λογισμικό μπορεί να προέρχονται από οποιονδήποτε τομέα, τα μαθηματικά, τη φυσική, τα οικονομικά, τη διοίκηση, την οργάνωση των επιχειρήσεων και των οργανισμών. Προτού γίνει προσπάθεια αντιμετώπισής τους πρέπει να γίνουν κατανοητά, να ταυτοποιηθούν, να οριοθετηθούν και να διατυπωθούν σωστά, ώστε να μην επιδέχονται καμιά άλλη ερμηνεία. Για να διαπιστωθεί κατά πόσο έχουν γίνει όλα αυτα κατανοητά οι προγραμματιστές επαναλαμβάνουν στον αναλυτή με το δικό τους τρόπο τις περιγραφές που τους έχουν γίνει. Μόνο έτσι αντιμετωπίζονται οι παρεμηνείες και αποφεύγονται οι παρανοήσεις.

Στη φάση της σχεδίασης ο προγραμματιστής προσφέρει αναφέροντας τις εμπειρίες του από λύσεις που έχουν δοθεί σε παρόμοια προβλήματα, καταθέτοντας έτοιμα τμήματα λογισμικού από παλαιότερες εργασίες, περιγράφοντας τις δυνατότητες που έχει η γλώσσα προγραμματισμού και το περιβάλλον στο οποίο θα γίνει η υλοποίηση. Στα σύγχρονα περιβάλλοντα πολλές διαδικασίες είναι προτυποποιημένες και δεν απαιτείται η λεπτομερής σχεδίασή τους. Για παράδειγμα, το πάτημα ενός πλήκτρου στην οθόνη, που δημιουργεί την ψευδαίσθηση ότι το πλήκτρο οπισθοχωρεί και επανέρχεται στη θέση του, είναι προτυποποιημένη συμπεριφορά που δε χρειάζεται να ξανασχεδιαστεί. Επίσης, πολλά σύγχρονα εργαλεία ανάπτυξης λογισμικού επιτρέπουν τη δημιουργία σχεδιασμού απευθείας στον υπολογιστή. Για παράδειγμα, τα παράθυρα επικοινωνίας ανθρώπου - υπολογιστή δε σχεδιάζονται πλέον σε χαρτί, αλλά απευθείας στην οθόνη.

Πολλοί προγραμματιστές περνούν στην τρίτη φάση, τη φάση της υλοποίησης πολύ γρηγορότερα από ότι θα έπρεπε. Αυτή η βιασύνη να προχωρήσουν σε κωδικοποίηση, πριν δοθούν τελικές λύσεις, οδηγεί σίγουρα σε προβληματικές καταστάσεις στο μέλλον. Στη φάση της υλοποίησης, το στυλ προγραμματισμού αποτελεί τον πιο σημαντικό παράγοντα δημιουργίας ποιοτικού λογισμικού. Το ποιοτικό πρόγραμμα ταυτίζεται με το απλό μιας και κάνει ευκολότερη την υλοποίηση και τη συντήρηση. Συνήθως, τα απλά προγράμματα είναι και ***εύκαμπτα*** (***flexible***). Σε ένα εύκαμπτο πρόγραμμα μπορεί να γίνουν εύκολα αλλαγές και επεκτάσεις. Για να γράφουμε εύκαμπτα προγράμματα πρέπει να αναρωτιόμαστε κάθε στιγμή τι θα συνέβαινε αν μας ζητούσαν να αλλάξουμε το τμήμα που κωδικοποιούμε τη συγκεκριμένη στιγμή. Οι νέοι προγραμματιστές πιεζόμενοι πολλές φορές από το χρόνο παράδοσης γράφουν συχνά πρόχειρα, γιατί νομίζουν ότι μπορούν να επανέλθουν αργότερα και να κάνουν διορθώσεις. Αυτό σπάνια γίνεται και ένα κακογραμμένο πρόγραμμα παραμένει για χρόνια σ' αυτήν την κατάσταση.

Η δουλειά του προγραμματιστή είναι να προσφέρει ένα πρόγραμμα με όσο το δυνατόν λιγότερα σφάλματα. Ο προγραμματιστής μπορεί να κάνει πολλά για να ελαττώσει τον αριθμό των ελέγχων και να εξασφαλίσει τη συμμόρφωση σε αυτή τη βασική προδιαγραφή. Παρατηρήσεις έχουν δείξει, ότι οι χρόνοι υλοποίησης και οι χρόνοι δοκιμών παρουσιάζουν μεγάλες αποκλίσεις μεταξύ προγραμματιστών.

Η φάση της υποστήριξης είναι η φάση την οποία αποφεύγουν κατά κανόνα οι προγραμματιστές. Οι έμπειροι προγραμματιστές προτιμούν να παράγουν πάντα κάτι καινούργιο και να ασχοληθούν με νέες τεχνολογίες. Οι νέοι προγραμματιστές δείχνουν απροθυμία να ασχοληθούν με κάτι που έχουν γράψει άλλοι, το οποίο μπορεί και να μη συνοδεύεται από την κατάλληλη τεκμηρίωση. Το κλειδί λοιπόν στη λύση του προβλήματος είναι η τεκμηρίωση που βοηθά τον έμπειρο προγραμματιστή να κάνει γρήγορα αλλαγές και να ασχοληθεί με άλλες εργασίες που προτιμά περισσότερο, αλλά και το νέο προγραμματιστή να μάθει τεχνικές που χρησιμοποιούν οι πιο έμπειροι.

Καλός προγραμματιστής είναι αυτός ο οποίος γράφει προγράμματα αποδοτικά, τα οποία συνοδεύονται από καλή τεκμηρίωση και ο οποίος βρίσκει εύκολα τα λάθη του.

Ανακεφαλαίωση

Η διαδικασία ανάπτυξης λογισμικού χωρίζεται σε φάσεις. Αυτές είναι η ανάλυση του προβλήματος, η σχεδίαση της λύσης, η υλοποίηση των προγραμμάτων, ο έλεγχος προδιαγραφών και η διόρθωση λαθών, η λειτουργία και η υποστήριξη.

Στόχος της ανάλυσης είναι να γίνει μια κατανοητή, πλήρης και σαφής περιγραφή των απαιτήσεων, των χαρακτηριστικών και της συμπεριφοράς του λογισμικού. Στη φάση της σχεδίασης το πρόβλημα χωρίζεται σε επιμέρους προβλήματα και επιλέγονται οι αλγόριθμοι που επιλύουν καθένα από αυτά. Όλα τα στοιχεία συνδέονται σε ένα πλήρες σχέδιο υλοποίησης.

Η εκτέλεση των φάσεων ανάπτυξης λογισμικού δεν είναι σειριακή. Κάποιες από τις φάσεις ανάπτυξης πρέπει να ολοκληρωθούν πλήρως, πριν γίνει οτιδήποτε στις επόμενες, ενώ κάποιες άλλες μπορούν να συνδυαστούν ή και να ενοποιηθούν και να γίνονται ταυτόχρονα με κάποιες άλλες. Για τη σύνδεση των φάσεων ανάπτυξης λογισμικού έχουν προταθεί κατά καιρούς πολλά μοντέλα. Δύο από τα πιο κλασικά είναι το μοντέλο του καταρράκτη - συντριβανιού και το μοντέλο του σαλίγκαρου.

Θέματα προς Συζήτηση

1. Η ανάλυση σκοπεύει στην παράδοση μιας σαφούς περιγραφής του τι θα μπορεί να κάνει το λογισμικό. Υπάρχουν διαφορές μεταξύ της ανάλυσης λογισμικού και της ανάλυσης που θα γινόταν για την κατασκευή ενός σχολείου σε μια περιοχή;

2. Σχολιάστε την πρόταση: "Ένα μικρό πρότυπο, αν μεγεθυνθεί, μπορεί να αποτελέσει το λογισμικό για τη λύση ενός γενικότερου προβλήματος".

3. Στη φάση της σχεδίασης διαιρούμε ένα πρόβλημα σε άλλα μικρότερα. Για παράδειγμα, το πρόβλημα δανεισμού ενός βιβλίου από τη βιβλιοθήκη του σχολείου, το χωρίζουμε σε πρόβλημα παράδοσης του βιβλίου, σε πρόβλημα τήρησης των ημερομηνιών επιστροφής του, σε πρόβλημα παραλαβής κατά την επιστροφή του. Σχεδιάστε τον τρόπο αντιμετώπισης καθενός από αυτά τα προβλήματα.

4. Το λογισμικό κατά την υλοποίηση και τον έλεγχο περνά από διαδοχικές αναθεωρήσεις, βελτιώσεις και εκδόσεις. Μια από αυτές είναι η έκδοση βήτα. Πότε ακριβώς γίνεται αυτή η έκδοση και τι εξυπηρετεί;

5. Πιεσμένη από το χρόνο παράδοσης ενός έργου μια ομάδα δεν κάνει καθόλου ή κάνει ελλειπή τεκμηρίωση. Νομίζετε ότι θα καταφέρει να παραδώσει το έργο της στον προκαθορισμένο χρόνο; Τι προβλήματα θα αντιμετωπίσει;

6. Στο σπίτι σας κάνει κάποιος τεκμηρίωση; Για ποια συγκεκριμένη δουλειά; Με ποιό τρόπο;

7. Οι συνεχείς αναθεωρήσεις και τα αιτήματα για βελτιώσεις μπορεί να οδηγήσουν σε συνεχείς παλινδρομήσεις μεταξύ φάσεων με αποτέλεσμα την αέναη ανάπτυξη ενός
8. προγράμματος. Ποια λύση δίνει το μοντέλο του σαλίγκαρου σε αυτό το πρόβλημα;
   <u>Υπόδειξη:</u> Παρατηρήστε ότι στο μοντέλο του σαλίγκαρου μπαίνει στόχος σε κάθε σπείρα.

9. Σε μια εφαρμογή βρίσκεστε στη φάση της υποστήριξης. Σε περίπτωση που ανακαλυφθεί κάποιο σφάλμα ή ακόμα σε περίπτωση εμφάνισης νέων απαιτήσεων, μέχρι ποιο στάδιο της ανάπτυξης της εφαρμογής είναι δυνατόν να αναθεωρήσετε;

10. Εφαρμόζεται το μοντέλο του σαλίγκαρου σε προϊόντα βιομηχανικής παραγωγής;
    <u>Υπόδειξη:</u> Εξετάστε την περίπτωση παραγωγής ενός αυτοκινήτου.

    Στα σχήματα 1-1 έως 1-4 αναφέρονται κάποιας μορφής επεξεργασίες που γίνονται στις φάσεις ανάπτυξης λογισμικού, π.χ. μελέτη, νοητική θύελλα κ.ά. Αναπτύξτε πώς φαντάζεστε ότι γίνονται κάποιες από αυτές.

Σημειώσεις:

Μάθημα 2

Αλγόριθμοι + δομές δεδομένων = προγράμματα

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να διακρίνουν αν μια διαδικασία, μέθοδος, ή συνταγή είναι αλγόριθμος.
- Να περιγράφουν έναν αλγόριθμο με λεκτικό τρόπο.
- Να αναγνωρίζουν τη σημασία των δομών δεδομένων.
- Να περιγράφουν τις κατηγορίες των γλωσσών προγραμματισμού.
- Να απαριθμούν τα χαρακτηριστικά της Visual Basic.

Οι άνθρωποι για να εκτελέσουν μια εργασία ακολουθούν οδηγίες, διαδικασίες και μεθόδους. Πολλές από αυτές είναι αρκετά σαφείς και οδηγούν στο ίδιο πάντα αποτέλεσμα, άλλες πάλι αφήνουν περιθώρια αυτενέργειας, ενώ άλλες δεν είναι καθόλου προσδιορισμένες και χρειάζονται φαντασία και νοητικές δεξιότητες για να παράγουν ένα αποτέλεσμα. Για παράδειγμα, η εκτέλεση της πρόσθεσης μεταξύ δύο ακέραιων αριθμών είναι πλήρως καθορισμένη πράξη και δίνει για όλους το ίδιο αποτέλεσμα. Η παρασκευή ενός γλυκού δεν είναι σαφώς προσδιορισμένη γι' αυτό και υπάρχουν διαφορές στη μορφή και τη γεύση του ακόμη κι αν η παραγωγή του γίνεται από το ίδιο άτομο. Τέλος, η δημιουργία ενός πίνακα ζωγραφικής δε διέπεται ούτε καν από οδηγίες και το αποτέλεσμα χαρακτηρίζει το δημιουργό.

Στους υπολογιστές τα πράγματα, τουλάχιστον προς το παρόν, είναι διαφορετικά. Οι λειτουργίες τους πρέπει να διέπονται από σαφείς κανόνες και να τους έχει υποδειχθεί με μεγάλη ακρίβεια το τι ακριβώς θα εκτελέσουν. Ακόμα κι αν φαίνεται ότι ο υπολογιστής παρουσιάζει διαφορετική συμπεριφορά, θα δούμε ότι στην ουσία δεν αυτενεργεί, αλλά επιλέγει μέσα από ένα μεγάλο πλήθος διαφορετικών ενεργειών, που όλες τους του έχουν περιγραφεί εκ των προτέρων. Στα μαθήματα που ακολουθούν, θα δούμε ότι ο τρόπος λειτουργίας των υπολογιστών καθορίζεται από αλγορίθμους, η περιγραφή των οποίων γίνεται στον υπολογιστή με τα προγράμματα σε μια γλώσσα προγραμματισμού.

Τι είναι Αλγόριθμος

Τη λέξη αλγόριθμο πριν λίγα χρόνια τη χρησιμοποιούσαν μόνο οι μαθηματικοί για να χαρακτηρίσουν μια αριθμητική διαδικασία (π.χ. ο αλγόριθμος του Ευκλείδη, το κόσκινο του Ερατοσθένη κ.ά.). Σήμερα, ο όρος αλγόριθμος χρησιμοποιείται εκτενώς και αδιακρίτως για να δηλώσει μεθόδους, διαδικασίες, τεχνικές, ακολουθίες εντολών και συνταγές οποιασδήποτε μορφής. Είναι όμως ό,τι εμπεριέχεται σε αυτές τις έννοιες αλγόριθμος; Για λόγους απλότητας μπορούμε να δώσουμε τον ορισμό του αλγορίθμου σαν μια σειρά από ενέργειες που περιγράφουν τη διαδικασία εκτέλεσης μιας εργασίας ή τον τρόπο επίλυσης ενός προβλήματος. Όμως αν θέλουμε να ορίσουμε την έννοια πιο αυστηρά πρέπει να πούμε ότι για να είναι μια διαδικασία αλγόριθμος πρέπει να έχει:

1. ***Είσοδο*** (***input***). Ο αλγόριθμος μπορεί να δέχεται καμία, μία ή πολλές τιμές δεδομένων. Στην περίπτωση που δε δέχεται δεδομένα ο αλγόριθμος επεξεργάζεται δεδομένα που παράγει ο ίδιος.
2. ***Έξοδο*** (***output***). Ο αλγόριθμος επιβάλλεται να δημιουργεί αποτελέσματα. Σε αντίθετη περίπτωση δεν έχουμε αλγόριθμο αλλά ***μαύρη τρύπα***.

και να είναι:

3. ***Περατή*** (***finite***). Ο αλγόριθμος πρέπει να τελειώνει σε πεπερασμένο χρονικό διάστημα. Αν μια διαδικασία δε χαρακτηρίζεται από τερματισμό, τότε δεν έχουμε αλγόριθμο αλλά ***υπολογιστική διαδικασία*** (***computational procedure***).
4. ***Πλήρως καθορισμένη*** (***definite***). Κάθε ενέργεια της διαδικασίας πρέπει να είναι σαφώς καθορισμένη και μάλιστα για κάθε μορφή δεδομένων. Δεν πρέπει να αφήνονται περιθώρια παρεμηνείας ούτε και να υπάρχει ελευθερία σε επιλογές. Για παράδειγμα, σε έναν

Η λέξη αλγόριθμος (algorithmi) προέρχεται από την παράφραση του ονόματος του Πέρση συγγραφέα Abu Mohammed ibn Musa al-Khowarizmi (του Ja'far που πατέρας του ήταν ο Mohammed, γιός του Musa από την Khowarizm). Ο al-Khowarizmi είχε γράψει το βιβλίο Kitab al jabr w'al-muqabala (Κανόνες αποκατάστασης και απλοποίησης) από την παράφραση του οποίου προέρχεται η λέξη Άλγεβρα).

Έστω ότι μας ζητείται να γράψουμε την τετραγωνική ρίζα του 2 με απόλυτη ακρίβεια. Αυτό δεν είναι δυνατόν να γίνει από έναν αλγόριθμο, ούτε στο χαρτί, ούτε στον υπολογιστή, αφού η τετραγωνική ρίζα του 2 έχει άπειρα ψηφία.

αλγόριθμο που σε ένα βήμα του εκτελείται η πράξη της διαίρεσης πρέπει να είναι σαφώς καθορισμένος ο τρόπος διαχείρισης των δεκαδικώ ψηφίων (πόσα θα λαμβάνονται υπ' όψιν και πώς θα γίνεται η στρογγυλοποίηση).

5. ***Αποτελεσματική*** (***Effective***). Κάθε βήμα της διαδικασίας πρέπει να είναι εφικτό να εκτελεστεί από αυτόν στον οποίο θα ανατεθεί η εκτέλεσή του. Για παράδειγμα η ενέργεια "τετραγώνισε τον κύκλο" αν δοθεί σαν βήμα μέσα σε μια διαδικασία δεν είναι κατορθωτή από κανέναν, όπως δεν είναι κατορθωτός και ο ακριβής υπολογισμός του αριθμού π ή της τετραγωνικής ρίζας του 2.

Ο τετραγωνισμός του κύκλου είναι άλυτο πρόβλημα και το π υπολογίζεται μόνο προσεγγιστικά.

Παράδειγμα 2-1.

Η σειρά των ενεργειών με την οποία υπολογίζεται το εμβαδόν του κύκλου είναι:

1. Διάβασε την ακτίνα του κύκλου.
2. ΄Υψωσε την ακτίνα στο τετράγωνο.
3. Πολλαπλασίασε το 3.14159 με την ακτίνα.
4. Τύπωσε το αποτέλεσμα σαν εμβαδόν του κύκλου.

Η διαδικασία έχει όλα τα χαρακτηριστικά ενός αλγορίθμου. Η πρώτη ενέργεια κάνει είσοδο δεδομένων και η τελευταία έξοδο. Ο χρόνος που απαιτείται για την εκτέλεσή της είναι πεπερασμένος (από την εμπειρία μας ξέρουμε ότι αρκούν μερικά δευτερόλεπτα για να γίνουν οι υπολογισμοί από άνθρωπο) και οι πράξεις που ζητείται να εκτελεστούν είναι πλήρως καθορισμένες και δίνουν αποτέλεσμα.

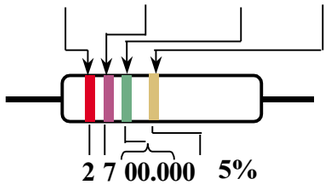
Παράδειγμα 2-2.

Η τιμή μιας ηλεκτρικής αντίστασης σε Ωμ γράφεται με κώδικα χρωμάτων. Στις ηλεκτρικές αντιστάσεις που έχουν τέσσερα χρώματα τα δύο πρώτα αποτελούν τα αρχικά ψηφία της τιμής της αντίστασης, το τρίτο δηλώνει το πλήθος των μηδενικών που ακολουθούν και το τέταρτο την επί τοις εκατό ανοχή, όπως φαίνεται και στο παράδειγμα του πίνακα 2-1.

Η ανάγνωση λοιπόν της τιμής μιας αντίστασης γίνεται με τον αλγόριθμο:

1. Διαβάζουμε το πρώτο χρώμα (π.χ. Κόκκινο)
2. Από τον πίνακα βρίσκουμε σε πιο Α'ψηφίο αντιστοιχεί. (Κόκκινο= 2)
3. Διαβάζουμε το δεύτερο χρώμα (π.χ. Μωβ)
4. Από τον πίνακα βρίσκουμε σε πιο Β'ψηφίο αντιστοιχεί. (Μωβ= 7)
5. Διαβάζουμε το τρίτο χρώμα (π.χ. Πράσινο)
6. Από τον πίνακα βρίσκουμε σε πόσα μηδενικά αντιστοιχεί. (Πράσινο = 00.000)
7. Διαβάζουμε το τέταρτο χρώμα (π.χ. Χρυσό)
8. Από τον πίνακα βρίσκουμε την ανοχή ±5%.
9. Συνθέτουμε την τιμή: (2.700.000 ±5% Ωμ)

Χρώμα	Α' ψηφίο	Β' ψηφίο	Μηδενικά	Ανοχή
Μαύρο	0	0	-	-
Καφέ	1	1	0	-
Κόκκινο	2	2	00	-
Πορτοκαλί	3	3	000	-
Κίτρινο	4	4	0.000	-
Πράσινο	5	5	00.000	-
Κυανούν	6	6	000.000	-
Ιώδες	7	7	0.000.000	-
Γκρίζο	8	8	-	-
Λευκό	9	9	-	-
Χρυσό	-	-	-	±5%
Ασημί	-	-	-	±10%



Πίνακας 2-1. *Κώδικας χρωμάτων ηλεκτρικών αντιστάσεων*

Η είσοδος του αλγορίθμου είναι τα τέσσερα χρώματα και η έξοδος είναι η τιμή της αντίστασης. Τα βήματα στα οποία εκτελείται η διαδικασία είναι πεπερασμένα και κάθε ενέργεια είναι σαφής και καθορισμένη.

Παράδειγμα 2-3.

Η διαδικασία:

1. Μετρητής ← 1
2. Μετρητής ← Μετρητής +1
3. Επανέλαβε το βήμα 2

δεν αποτελεί αλγόριθμο αλλά υπολογιστική διαδικασία. Στο πρώτο βήμα ο μετρητής έχει την τιμή 1, στο δεύτερο βήμα ο μετρητής παίρνει την τιμή 1+1=2, το βήμα 3 υποδεικνύει την επανάληψη του βήματος 2, οπότε ο μετρητής θα πάρει την τιμή 2+1=3 κ.ο.κ. Η διαδικασία εκτελείται επάπειρον κάνοντας απαρίθμηση των ακεραίων αριθμών, δεν έχει λοιπόν την ιδιότητα της περατότητας. Μόνο κάποιο εξωτερικό συμβάν μπορεί να τη διακόψει.

Παράδειγμα 2-4.

Μια από τις μεθόδους επίλυσης κυκλωμάτων είναι η μέθοδος ρευμάτων του Kirchhoff. Η σειρά εργασίας κατά την εφαρμογή της μεθόδου είναι:

1. Αναγνωρίζουμε τους κόμβους κ, τους κλάδους λ και τους βρόχους του κυκλώματος.
2. Καθορίζουμε αυθαιρέτως τις φορές των αγνώστων ρευμάτων στους διάφορους κλάδους.
3. Καθορίζουμε αυθαιρέτως θετική φορά διαγραφής των βρόχων.
4. Εφαρμόζουμε τη σχέση $\Sigma i = 0$ σε κ-1 κόμβους και λαμβάνουμε ισάριθμες γραμμικώς ανεξάρτητες εξισώσεις, όπου κ το πλήθος των κόμβων.
5. Εφαρμόζουμε τη σχέση $\Sigma E = \Sigma iR$ σε λ-κ+1 βρόχους και λαμβάνουμε ισάριθμες γραμμικώς ανεξάρτητες εξισώσεις.
6. Επιλύουμε το προκύπτον σύστημα.

Ο τρόπος με τον οποίο περιγράφθηκε εδώ η μέθοδος δεν αποτελεί αλγόριθμο. Σε κάποια βήματα ζητείται να γίνει αυθαίρετη επιλογή (π.χ. αυθαίρετη επιλογή αγνώστων ρευμάτων) και σε κάποια άλλα δεν περιγράφονται οι τρόποι με τους οποίους θα κάνουμε κάποιες ενέργειες αλλά απλώς υποδεικνύονται οι ενέργειες (π.χ. λαμβάνουμε ισάριθμες γραμμικώς ανεξάρτητες εξισώσεις). Σε έναν αλγόριθμο δεν πρέπει να υπάρχουν αυθαιρεσίες, ούτε υποδείξεις και απαιτείται λεπτομερής περιγραφή των ενεργειών.

Ανάλυση αλγορίθμων

Αρκετά προβλήματα επιλύονται με περισσότερους από έναν αλγορίθμους. Σ' αυτήν την περίπτωση μας ενδιαφέρει ποιοι από αυτούς είναι οι καλύτεροι λαμβάνοντας υπ'όψιν κάποια κριτήρια. Το βασικό κριτήριο χαρακτηρισμού ενός αλγόριθμου ως καλού ή κακού είναι ο χρόνος που απαιτεί για να φέρει σε πέρας την όλη διαδικασία σε σχέση με άλλους αλγόριθμους. Άλλα κριτήρια είναι η απαίτηση του σε πόρους (π.χ. πόση μνήμη χρησιμοποιεί, αν κάνει χρήση ειδικού υλικού), η γενικότητα (αν επιλύει ένα σύνολο συναφών προβλημάτων), η προσαρμοστικότητα, η απλότητα.

Ένα πρόβλημα μπορεί να λύνεται με περισσότερους από έναν αλγόριθμους.

Παράδειγμα 2-5.

΄Εστω ότι θέλουμε να αναζητήσουμε το όνομα κάποιου μέσα σε έναν τηλεφωνικό κατάλογο που περιέχει ονόματα ταξινομημένα αλφαβητικά. ΄Ενας από τους αλγόριθμους αναζήτησης, γνωστός ως αλγόριθμος σειριακής αναζήτησης, ορίζει τα εξής:

1. Διαβάζουμε το όνομα που θα αναζητήσουμε.
2. Ξεκινάμε από την αρχή του καταλόγου.
3. Διαβάζουμε το όνομα που έχει σειρά.
4. Αν είναι ίδιο με το όνομα που ψάχνουμε, διακόπτουμε.
5. Προχωράμε στην επόμενη γραμμή του καταλόγου.
6. Επανερχόμαστε στο βήμα 3.

Από την εμπειρία μας γνωρίζουμε ότι η σειριακή αναζήτηση μπορεί να δώσει αποτέλεσμα αλλά είναι πολύ χρονοβόρα, ιδιαίτερα αν το όνομα που ψάχνουμε βρίσκεται στις τελευταίες θέσεις του καταλόγου. Αντί αυτής μπορούμε να χρησιμοποιήσουμε μια πιο σύντομη που είναι γνωστή ως μέθοδος της δυαδικής αναζήτησης.

Ο γκουρού των αλγορίθμων Donald Knuth γράφει: Τους αλγόριθμους δεν τους διαβάζουμε σαν νουβέλες, χρησιμοποιούμε μολύβι και χαρτί για να ακολουθήσουμε τα βήματά τους.

1. Διαβάζουμε το όνομα που θα αναζητήσουμε.
2. Ξεκινάμε από τη μέση του καταλόγου
3. Διαβάζουμε το όνομα που έχει σειρά.
4. Αν είναι ίδιο με το όνομα που ψάχνουμε, διακόπτουμε.
5. Αν προηγείται αλφαβητικά θεωρούμε νέα μέση το κάτω μισό του καταλόγου.
6. Αν έπεται αλφαβητικά θεωρούμε νέα μέση το πάνω μισό του καταλόγου.
7. Επανερχόμαστε στο βήμα 3.

Έστω, ότι ο τηλεφωνικός κατάλογος περιέχει 1.000.000 ονόματα. Κατά μέσο όρο, ο αλγόριθμος της σειριακής αναζήτησης θα κάνει 500.000 αναζητήσεις. Αντίθετα, ο αλγόριθμος της δυαδικής αναζήτησης θα κάνει μόνο 20 αναζητήσεις.

Αναπαράσταση αλγορίθμων

Διαγράμματα ροής, δομοδιαγράμματα και ψευδοκώδικες. Παλιές ιστορίες ...

Ο λεκτικός τρόπος περιγραφής των αλγορίθμων που χρησιμοποιήθηκε στα παραδείγματα δεν είναι και ο πιο ενδεδειγμένος. Όταν οι αλγόριθμοι είναι μεγάλοι και χρησιμοποιούν πολύπλοκη λογική υποθέσεων και επαναλήψεων η περιγραφή τους γίνεται δυσνόητη. Κατά καιρούς χρησιμοποιήθηκαν πιο εποπτικοί τρόποι περιγραφής των αλγορίθμων.

Στην αρχή για την αναπαράσταση των αλγορίθμων χρησιμοποιήθηκαν διαγραμματικοί τρόποι αναπαράστασης όπως τα ***διαγράμματα ροής*** (***flow charts***) και τα ***δομοδιαγράμματα*** (***structure grams***). Ο διαγραμματικός τρόπος παρουσίασης είχε σαν στόχο την όσο το δυνατόν πιο γενική περιγραφή, ώστε να είναι ανεξάρτητη γλώσσας προγραμματισμού, μιας και οι γλώσσες προγραμματισμού παρουσίαζαν τεράστιες διαφορές στη σύνταξη και τη μορφή. Επίσης, ο τρόπος αυτός αναπαράστασης ήταν πιο οικείος στους μηχανικούς του παρελθόντος, που είχαν συνηθίσει να περιγράφουν ένα έργο πρώτα με σχέδια.

Καθώς όμως τα προγράμματα γίνονταν όλο και μεγαλύτερα η περιγραφή με σχέδια καταλάμβανε όλο και μεγαλύτερο χώρο, γινόταν όλο και πιο δύσκολη, απαιτούσε πολύ χρόνο, ενώ παράλληλα αύξανε το κόστος. Παράλληλα, οι γλώσσες προγραμματισμού σύγκλιναν σε ότι αφορά τον τρόπο σύνταξης και τις προγραμματιστικές δομές. Τότε επινοήθηκε ένας γενικός λεκτικός τρόπος περιγραφής αλγορίθμων που έμοιαζε με γλώσσα προγραμματισμού, ο οποίος ονομάστηκε ***ψευδοκώδικας*** (***pseudocode***). Ο ψευδοκώδικας χρησιμοποιούσε τα κοινά στοιχεία σύνταξης των γλωσσών προγραμματισμού.

Με την εξέλιξη όμως των επεξεργαστών κειμένου, οι προγραμματιστές εκμεταλλευόμενοι τις δυνατότητες εισαγωγής, διαγραφής και μεταφοράς κώδικα στην οθόνη του υπολογιστή προτιμούσαν να γράφουν απευθείας σε μια γλώσσα προγραμματισμού παρά να γράφουν δυο φορές το ίδια πράγματα (και σε ψευδοκώδικα και σε γλώσσα προγραμματισμού). Επιπλέον, τα τελευταία χρόνια με την εισαγωγή νέων τεχνικών οπτικού προγραμματισμού, αντικειμενοστρεφούς προγραμματισμού και προγραμματισμού καθοδηγούμενου από συμβάντα, η κωδικοποίηση μετεξελίχθηκε σε συγγραφή συμπλεγμάτων κώδικα, με παράλληλο σχεδιασμό αντικειμένων, περιγραφή των ιδιοτήτων τους, των μεταξύ τους σχέσεων και της αλληλεπίδρασής τους. Ο ψευδοκώδικας έχασε και τα τελευταία ερείσματα του μιας και ήταν κατάλληλος μόνο για ***διαδικασιακές γλώσσες*** (***procedural languages***).

Οι διαδικασιακές ή αλγοριθμικές γλώσσες στηρίζονταν στη λογική ότι οι εντολές ενός προγράμματος εκτελούνται η μια μετά την άλλη.

Δομές δεδομένων

Εκτός από την επιλογή του αλγορίθμου, μια άλλη διαδικασία που πρέπει να πραγματοποιηθεί είναι ο καθορισμός των δομών των δεδομένων. Τα δεδομένα που χρησιμοποιούνται από το πρόγραμμα πρέπει να είναι έτσι οργανωμένα ώστε να συνθέτουν λογικές ενότητες με στόχο να γίνεται ευκολότερα και γρηγορότερα η επεξεργασία τους από τον υπολογιστή και να είναι πιο κατανοητά από αυτούς που θα αναλάβουν τη συντήρηση.

Κατά τη σύνθεση μιας δομής δεδομένων λαμβάνουμε υπ' όψιν μας τα είδη των δεδομένων που θα περιλάβει, τη θέση τους, τον τρόπο αποθήκευσής τους, τη μεταξύ τους σχέση.

Παραδείγματα 2-6.

α) Τα στοιχεία του παραλήπτη που γράφονται πάνω σε μια επιστολή αποτελούν μια δομή δεδομένων. Τα δεδομένα που περιλαμβάνουμε στη δομή "παραλήπτης" είναι το ονοματεπώνυμο του παραλήπτη, η οδός και ο αριθμός, ο ταχυδρομικός κώδικας και η περιοχή, η χώρα, που θα πάει η επιστολή. Η δομή είναι:

Ονοματεπώνυμο
Οδός, αριθμός
Τ.Κ., Περιοχή
Χώρα

και μπορεί να δεχτεί δεδομένα της μορφής:

Οικονόμου Γιάννης
Ολυμπίας 22
176 73, Καλλιθέα
Ελλάδα

β) Η ημερομηνία αποτελεί μια δομή δεδομένων, στην οποία περιλαμβάνονται η ημέρα, ο μήνας και το έτος σε μια συγκεκριμένη σειρά και με συγκεκριμένη μορφή. Π.χ. η μορφή ηη/μμ/εε επιβάλλει να γράψουμε την 5η Μαρτίου του 2000 ως 05/03/00.

γ) Σε ένα σύστημα ορθογωνίων αξόνων μπορούμε να παραστήσουμε κάθε σημείο του επιπέδου, που έχει τετμημένη x και τεταγμένη y, με ένα ζεύγος (x,y).

δ) Ο τρόπος γραφής της τιμής μιας αντίστασης που περιγράψαμε στο παράδειγμα 2-2 αποτελεί μια δομή δεδομένων:

α' χρώμα
β' χρώμα
μηδενικά
ανοχή

στην οποία έχει συμφωνηθεί το πλήθος των σημαντικών ψηφίων που θα χρησιμοποιείται (τα δύο πρώτα χρώματα), ο τρόπος που θα περιγράφεται το μέγεθος της τιμής (το τρίτο χρώμα) και η ανοχή της (τό τέταρτο χρώμα).

Οι αλγόριθμοι και οι δομές δεδομένων είναι δύο στοιχεία άρρηκτα συνδεδεμένα. Η σύνθεση της κατάλληλης δομής δεδομένων και η επιλογή του κατάλληλου αλγορίθμου απαιτεί ιδιαίτερη προσοχή γιατί, αν ο αλγόριθμος ή η δομή είναι είτε ακατάλληλα είτε αταίριαστα μεταξύ τους, μπορεί να οδηγήσουν στη σύνθεση προγράμματος υπερβολικά σύνθετου, που να οδηγήσει την εφαρμογή σε πλήρη αποτυχία.

Παραδείγματα 2-7.

α) Τόσο το είδος των στοιχείων του παραλήπτη όσο και η σειρά που έχουν τοποθετηθεί είναι προσυμφωνημένα και εξυπηρετούν το σύστημα ταξινόμησης και διανομής των ταχυδρομείων. Ας δούμε πώς δουλεύει ένας αλγόριθμος παράδοσης.

1. Οι ταξινομητές στα ταχυδρομικά γραφεία κοιτούν πάντα τον ταχυδρομικό κώδικα που βρίσκεται στην αρχή της τελευταίας γραμμής για να κατευθύνουν τις επιστολές στο κατάλληλο ταχυδρομικό γραφείο.
2. Εκεί, οι διανομείς συμβουλεύονται την οδό και τον αριθμό που βρίσκεται στη μεσαία γραμμή και αφήνουν τις επιστολές έξω από τα σπίτια.
3. Οι παραλήπτες αναγνωρίζουν το όνομά τους στις επιστολές και τις παραλαμβάνουν.

"Αλγόριθμοι+ δομές δεδομένων= προγράμματα" αποτελεί τον τίτλο βιβλίου του Niclaus Wirth, πατέρα της γλώσσας προγραμματισμού Pascal.

Σκεφτείτε όμως, αν θα μπορούσε να δουλέψει ένα σύστημα διανομής με δεδομένα της μορφής:

Στον μπατζανάκη μου το Γιάννη
που τρώει κοτόπουλα
στη Νέα Ιωνία
Χτυπήστε το τζάμι για να ανοίξει

β) Από ένα φίλο μας ελληνοαμερικάνο παίρνουμε ένα γράμμα που γράφει: "Θα έρθω στη Θεσσαλονίκη την 05/03/00, με την πρωϊνή πτήση".
Στις 5 Μαρτίου του 2000 πάμε στο αεροδρόμειο και περιμένουμε αλλά ο φίλος μας δεν έρχεται. Τον καλούμε στο τηλέφωνο για να τον ρωτήσουμε αν ανέβαλε το ταξίδι του. Μας απαντά, πως δε σκοπεύει να κάνει κάτι τέτοιο και ότι θα έρθει στις 3 Μαΐου, όπως μας είχε γράψει στο γράμμα.
Η παρεξήγηση δημιουργήθηκε γιατί στις Η.Π.Α. χρησιμοποιούν τη μορφή μμ/ηη/εε όταν γράφουν τις ημερομηνίες και όχι τη μορφή ηη/μμ/εε που συνηθίζεται σε μερικές ευρωπαϊκές χώρες.

Γλώσσες προγραμματισμού

Αποτελεί πια κοινό μυστικό, ότι η περιγραφή ενός αλγορίθμου στον υπολογιστή δεν είναι δυνατόν να γίνει απευθείας σε γλώσσα που χρησιμοποιούμε εμείς οι άνθρωποι. Θα ήταν ευχής έργο να καταλαβαίνει ένα υπολογιστικό σύστημα τη γλώσσα που μιλάμε, αλλά η σύνταξη και η δομή των ανθρωπίνων γλωσσών είναι τόσο πολύπλοκες, ώστε να μη θεωρείται προς το παρόν πραγματοποιήσιμο κάτι τέτοιο.

Αυτή η δυσκολία έχει ξεπεραστεί με τη δημιουργία τεχνητών γλωσσών, που διαθέτουν μικρό και απλό λεξιλόγιο, εύκολη και στερεότυπη σύνταξη και συμβολισμούς, που δε διαφέρουν από τους συμβολισμούς που χρησιμοποιούμε στα μαθηματικά και στις καθημερινές δραστηριότητες. Σήμερα υπάρχει σε χρήση μια πληθώρα τέτοιων γλωσσών (πάνω από 2500 !) με ποικίλες δυνατότητες και χρήσεις. Οι λόγοι που οδήγησαν σε αυτή τη βαβέλ των γλωσσών ήταν, και είναι, οι διαφορετικές ανάγκες που παρουσιάζονται σε κάθε κλάδο δραστηριοτήτων του ανθρώπου, η διαφορετική θεώρηση των προβλημάτων ακόμα και στον ίδιο χώρο εφαρμογών, οι ιδέες πάνω στον προγραμματισμό.

Οι γλώσσες ως προς το χώρο που απευθύνονται ταξινομούνται σε:

- ***Γενικής χρήσης*** (***general purpose***) που θεωρητικά μπορούν να χρησιμοποιηθούν για προβλήματα οποιασδήποτε μορφής, αλλά στην πράξη είναι σχεδιασμένες για να μπορούν να ανταποκριθούν σε ορισμένη κατηγορία προβλημάτων, οπότε τις διακρίνουμε σε:
  - ***Εμπορικής κατεύθυνσης*** (***business oriented languages***), όπως η Cobol και η RPG για τα εμπορικά, οικονομικά και διοικητικά προβλήματα, στα οποία γίνεται διακίνηση μεγάλου όγκου δεδομένων (π.χ. μητρώα πελατών, καταστάσεις τιμολογίων και άλλων παραστατικών).
  - ***Επιστημονικής κατεύθυνσης*** (***science oriented languages***), όπως η FORTRAN, η ADA, η ALGOL, η PASCAL, για τα επιστημονικά και τεχικά προβλήματα, στα οποία απαιτούνται πολλοί αριθμητικοί υπολογισμοί (πράξεις με μιγαδικούς αριθμούς, διανύσματα και μαθηματικές συναρτήσεις).
  - ***Προγραμματισμού συστήματος*** (***system programming languages***), όπως η C, για τη δημιουργία λογισμικού συστήματος (λειτουργικά συστήματα, βιβλιοθήκες ρουτινών γενικής εξυπηρέτησης συστήματος και περιφερειακών συσκευών).
  - ***Τεχνητής νοημοσύνης*** (***artificial intelligence languages***), όπως η PROLOG και η LISP για δημιουργία έμπειρων συστημάτων, απόδειξη θεωρημάτων.
- ***Ειδικού σκοπού*** (***special purpose languages***), που έχουν αναπτυχθεί για ορισμένη κατηγορία εφαρμογών και διαθέτουν λειτουργίες ανάλογα με την περιοχή των εφαρμογών, για παράδειγμα:
  - ***Ρομποτική***, ***αυτοματισμούς και μηχανές ελέγχου***, όπως είναι η FORTH, η LADER η APT.
  - ***Διαχείριση βάσεων δεδομένων***, όπως είναι οι γλώσσες ερωταπαντήσεων (Query languages) SQL, για την άντληση πληροφοριών από μια βάση δεδομένων και τη δημιουργία, διαγραφή και ενημέρωση των δεδομένων.
  - ***Χειρισμό κειμένων***, όπως είναι η SNOBOL, για την εύκολη αναζήτηση τμημάτων κειμένων και λεκτικών σε κείμενα.
  - ***Παρουσίαση Ιστοσελίδων*** στον Παγκόσμιο Ιστό, όπως είναι η HTML, η XML, η VRML.
  - Γραφικά και εκδόσεις, για τη δημιουργία δισδιάστατων και τρισδιάστατων γραφικών κ.ά.

Οι γλώσσες ως προς τις ιδέες πάνω στις οποίες βασίζονται ταξινομούνται σε:

- ***Διαδικασιακές*** (***procedural languages***) ή αλγοριθμικές γλώσσες, που έχουν τέτοια δομή ώστε να είναι κατάλληλες για την έκφραση ενός αλγορίθμου. Τα προγράμματα δομούνται πάνω σε έναν σκελετό ο οποίος έχει τον έλεγχο του υπολογιστικού συστήματος τη στιγμή της εκτέλεσης ρυθμίζοντας το πότε θα πάρει δεδομένα, το πώς θα τα επεξεργαστεί και το πότε θα δώσει αποτελέσματα.
- ***Αντικειμενοστρεφείς*** (***object oriented languages***), στις οποίες το πρόγραμμα δομείται από αντικείμενα, που έχουν ιδιότητες και συγκεκριμένο τρόπο συμπεριφοράς όταν αλληλεπιδρούν ανταλλάσσοντας μηνύματα.
- ***Συναρτησιακές*** (***functional languages***), στις οποίες τα προγράμματα συντίθενται αποκλειστικά και μόνο από συναρτήσεις.
- ***Μη διαδικασιακές*** (***non procedural languages***) στις οποίες περιγράφονται τα συστατικά ενός προβλήματος και ο τρόπος σύνδεσής τους.
- ***Ροής δεδομένων*** (***data flow languages***)

κ.ά.

Το ερώτημα που εύλογα τίθεται είναι, σε ποια από όλες τις υπάρχουσες γλώσσες συμφέρει να γίνει η υλοποίηση μιας εφαρμογής. Η επιλογή αυτή καθορίζεται από τους εξής παράγοντες:

- Τη φύση του προβλήματος (διοικητικοοικονομικό, τεχνικό κ.λπ.).
- Τις γλώσσες που γνωρίζουν οι διαθέσιμοι προγραμματιστές.
- Τα δικαιώματα χρήσης για κάθε γλώσσα.
- Τις δυνατότητες συντήρησης του λογισμικού.
- Τη φιλικότητα του περιβάλλοντος ανάπτυξης.
- Τη βιωσιμότητα της κάθε γλώσσας.

Κατά κανόνα αποφασίζεται η ανάπτυξη σε μια γλώσσα γενικής χρήσης και οι σύγχρονες αντιλήψεις υπαγορεύουν ως πρόσθετο κριτήριο να έχει γραφικό περιβάλλον ανάπτυξης και να είναι αντικειμενοστρεφής.

Η γλώσσα Basic

Η γλώσσα Basic είναι η πιο διαδεδομένη γλώσσα στο χώρο των μικροϋπολογιστών. Το όνομά της προέρχεται από τα αρχικά των λέξεων ***Beginner's All purpose Symbolic Instruction Code***, που σε ελεύθερη απόδοση σημαίνει ***Συμβολικός Κώδικας Εντολών κάθε χρήσης για τους αρχάριους***. Η γλώσσα άρχισε να αναπτύσσεται στο Dartmouth College για καθαρά εκπαιδευτικούς σκοπούς. Οι δημιουργοί της, ο John Kemeny και ο Thomas Kurtz, ήθελαν να βρουν έναν τρόπο να μαθαίνουν εύκολα οι σπουδαστές τους προγραμματισμό και να χρησιμοποιούν αμέσως τον υπολογιστή για να εισάγουν τα προγράμματά τους, χωρίς να είναι υποχρεωμένοι να αποστηθίζουν τις πολύπλοκες διαδικασίες γραφής, μετάφρασης και τρεξίματος του προγράμματος που απαιτούσαν οι τότε γλώσσες προγραμματισμού.

Με τα χρόνια, η γλώσσα αγαπήθηκε λόγω της απλότητάς της. Η συγκυρία κατασκευής των μικροϋπολογιστών τη βοήθησε να εξαπλωθεί ευρέως, όταν υιοθετήθηκε ως η κατ' εξοχήν γλώσσα προγραμματισμού στους μικροϋπολογιστές τη δεκαετία του '70. Παράλληλα, επεκτάθηκε και συμπληρώθηκε για να καλύψει κι άλλες ανάγκες, με αποτέλεσμα να μπορεί να ικανοποιήσει τις απαιτήσεις κάθε είδους εφαρμογών και να μην υστερεί έναντι των άλλων γλωσσών προγραμματισμού. Όταν άρχισαν να επικρατούν οι νέες ιδέες παραθυρικής επικοινωνίας χρήστη - υπολογιστή δημιουργήθηκε νέας μορφής ενοποιημένο περιβάλλον ανάπτυξης εφαρμογών για την Basic από την εταιρεία Microsoft. Μπρος από το όνομά της μπήκε ο προσδιορισμός Visual (οπτική).

Η Visual Basic

Με την επικράτηση των Windows η γλώσσα Visual Basic ακολούθησε τις τάσεις των καιρών και μετεξελίχθηκε σε ένα ακόμα πιο ισχυρό εργαλείο προγραμματισμού. Σήμερα μάλιστα προσφέρει τον απλούστερο, τον ευκολότερο και το γρηγορότερο τρόπο ανάπτυξης λογισμικού

Η Visual Basic θα αναφέρεται στο εξής για λόγους συντομίας ως VB.

στο παραθυρικό περιβάλλον των Windows. Επίσης, αν και η γλώσσα περιέχει το συνθετικό Basic στο όνομά της, τα πράγματα που μπορούμε να κάνουμε, όταν χρησιμοποιούμε αυτή τη γλώσσα, δεν είναι και τόσο βασικά.

Η σημερινή μορφή της γλώσσας απέχει κατά πολύ από την αρχική της μορφή και είναι εμπλουτισμένη με πλήθος σύγχρονων προγραμματιστικών δομών καθώς και συναρτήσεων και διαδικασιών που διαχειρίζονται απευθείας το γραφικό περιβάλλον των Windows. Τόσο τα παλιά χαρακτηριστικά της γλώσσας όσο και τα νέα βοηθούν όχι μόνο τον αρχάριο προγραμματιστή να δημιουργήσει γρήγορα αξιόλογες εφαρμογές, αλλά και τον έμπειρο προγραμματιστή να εκμεταλλευτεί κάθε δυνατότητα που προσφέρουν οι σημερινοί υπολογιστές και τα σύγχρονα Λειτουργικά Συστήματα. Έτσι, μαθαίνοντας ένας νέος προγραμματιστής αυτή τη γλώσσα επενδύει στο μέλλον μιας και αποκτά ένα εργαλείο για την υλοποίηση και επαγγελματικών εφαρμογών. Τα βασικά χαρακτηριστικά της VB είναι:

- Απλή σύνταξή που δεν απαιτεί πολλές γνώσεις πληροφορικής και αυστηρούς ορισμούς, η οποία είναι πιο κοντά στην αγγλική γλώσσα από ότι η σύνταξη άλλων γλωσσών προγραμματισμού. Επίσης, ως προς τη σύνταξη, οι εντολές σχηματίζουν δομές που ακολουθούν τους κανόνες του ***δομημένου προγραμματισμού*** (***structured programming***).
- Το περιβάλλον, στο οποίο υλοποιούνται οι εφαρμογές, πραγματοποιεί αυτόματα συντακτικό έλεγχο τη στιγμή εισαγωγής των εντολών και προσφέρει μεγάλες ευκολίες για την ανίχνευση και τη διόρθωση λαθών.
- ***Οπτικός προγραμματισμός*** (***visual programming***). Ο προγραμματιστής δίνει πολύ λίγο από το χρόνο του για τη δημιουργία της διεπαφής χρήστη - υπολογιστή, μιας και την πραγματοποιεί σχεδιάζοντας παρά κωδικοποιώντας. Αντί δηλαδή να γράφει μεγάλους κώδικες κειμένου για την περιγραφή των αντικειμένων που τη συνθέτουν, δεν έχει παρά να επιλέξει με το ποντίκι προκατασκευασμένα αντικείμενα από μια εργαλειοθήκη, να τα τοποθετήσει στην κατάλληλη θέση σε μια φόρμα σχεδίασης και να καθορίσει το μέγεθός τους, όπως ακριβώς θα έκανε και σε ένα σχεδιαστικό πρόγραμμα (π.χ. το Paint).

Μέσω της διεπαφής γίνεται η αμφίδρομη επικοινωνία του χρήστη με το λογισμικό και τον υπολογιστή.

- Εκμετάλλευση σύνθετων λειτουργιών και εννοιών με απλό τρόπο. Παράθυρα, μενού, πεδία κειμένου, διαλογικά παράθυρα, ράβδοι κύλισης, γραμματοσειρές κ.ά. σχεδιάζονται χωρίς κώδικα, με απλές κινήσεις και ελέγχονται μέσα από το πρόγραμμα με μερικές μόνο γραμμές.
- ***Αντικειμενοστρεφής προγραμματισμός*** (***object oriented programming***). Τα παράθυρα, τα αντικείμενα που τοποθετούνται πάνω στα παράθυρα και άλλα αντικείμενα του περιβάλλοντος εργασίας έχουν ιδιότητες και συγκεκριμένους τρόπους συμπεριφοράς, λειτουργούν δηλαδή σαν αντικείμενα του φυσικού κόσμου.
- ***Προγραμματισμός οδηγούμενος από συμβάντα*** (***event driven programming***). Σε ένα αντικειμενοστρεφές περιβάλλον το πρόγραμμα δεν ελέγχει τις συσκευές αλληλεπίδρασης (π.χ. το ποντίκι, το πληκτρολόγιο) με το χρήστη για να διαπιστώσει αν έχουν πατηθεί τα πλήκτρα τους, ούτε και το ρολόι του υπολογιστή για να διαπιστώσει, αν έχει φτάσει κάποια στιγμή διέγερσης. Όταν προκύψει κάποιο συμβάν από αυτές τις συσκευές, στέλνεται ένα μήνυμα που διεγείρει το αντικείμενο που έχει υποδειχθεί από τη συσκευή.
- Πλήρης εκμετάλλευση του μηχανισμού ActiveX που επιτρέπει τη δημιουργία αντικειμένων και την διαχείρισή τους από άλλες εφαρμογές.
- Διαχείριση αρχείων βάσεων δεδομένων. Είναι δυνατή η εκμετάλλευση αρχείων Access μέσω του μηχανισμού ***Jet Database Engine*** και αρχείων βάσεων δεδομένων Oracle και SQL Server μέσω του μηχανισμού ***Open DataBase Connectivity - ODBC***.
- Εκτέλεση διαδικασιών ***API*** (***Application Programming Interface***) των Windows ή διαδικασιών που βρίσκονται σε ***δυναμικής σύνδεσης βιβλιοθήκες*** (***Dynamic Link Libraries - DLLs***) για εκμετάλλευση των δυνατοτήτων άλλων εφαρμογών μέσω του μηχανισμού ***σύνδεσης και ενσωμάτωσης αντικειμένων*** (***Object Linking & Embedding - OLE***).
- Παραλλαγές της, γνωστές με το όνομα Visual Basic for Applications, χρησιμοποιούνται για την πραγματοποίηση αυτοματοματοποιημένων διαδικασιών σε τυποποιημένα προγράμματα αυτοματισμού γραφείου, όπως είναι οι επεξεργαστές κειμένου (π.χ. Word), τα φύλλα εργασίας (π.χ. Excel), τα προγράμματα παρουσίασης, βάσεων δεδομένων (π.χ. Access), τα προγράμματα παρουσίασης (π.χ. Power Point).
- Υπερσύνολο της VBScript που χρησιμοποιείται σήμερα για προγραμματισμό στο Internet. Η VBScript έχει δυνατότητες δημιουργίας και εύκολης προσπέλασης σε εφαρμογές και έγγραφα στο Internet καθώς και δημιουργίας εφαρμογών που να προσφέρουν υπηρεσίες στους χρήστες του Internet.

Ανακεφαλαίωση

Αλγόριθμος είναι μια σειρά από ενέργειες που περιγράφουν τη διαδικασία εκτέλεσης μιας εργασίας ή τον τρόπο επίλυσης ενός προβλήματος που έχει είσοδο και έξοδο και είναι περατή, πλήρως καθορισμένη και κατορθωτή. Το βασικό κριτήριο χαρακτηρισμού ενός αλγόριθμου ως καλού ή κακού είναι ο χρόνος που απαιτεί για να φέρει σε πέρας την όλη διαδικασία σε σχέση με άλλους αλγόριθμους. Άλλα κριτήρια είναι η απαίτησή του σε πόρους, η γενικότητα, η προσαρμοστικότητα, η απλότητα. Οι αλγόριθμοι περιγράφονται λεκτικά, με διαγράμματα ροής, με δομοδιαγράμματα, με ψευδοκώδικα, με κώδικα σε γλώσσα προγραμματισμού. Η σύνθεση της κατάλληλης δομής δεδομένων και η επιλογή του κατάλληλου αλγορίθμου απαιτεί ιδιαίτερη προσοχή.

Οι γλώσσες προγραμματισμού ως προς το χώρο που απευθύνονται ταξινομούνται σε εμπορικής κατεύθυνσης, επιστημονικής κατεύθυνσης, προγραμματισμού συστήματος, τεχνητής νοημοσύνης, ρομποτικής, διαχείρισης βάσεων δεδομένων, χειρισμού κειμένων, παρουσίασης ιστοσελίδων, γραφικών κ.ά. Οι γλώσσες, ως προς τις ιδέες πάνω στις οποίες βασίζονται, ταξινομούνται σε διαδικασιακές, αντικειμενοστρεφείς, συναρτησιακές, μη διαδικασιακές, ροής δεδομένων.

Η γλώσσα Basic είναι η πιο διαδεδομένη γλώσσα προγραμματισμού στους μικροϋπολογιστές. Όταν άρχισαν να επικρατούν οι νέες ιδέες παραθυρικής επικοινωνίας χρήστη - υπολογιστή δημιουργήθηκε νέας μορφής ενοποιημένο περιβάλλον ανάπτυξης εφαρμογών για την Basic, που ονομάστηκε Visual Basic. Με την επικράτηση των Windows η γλώσσα Visual Basic ακολούθησε τις τάσεις των καιρών και μετεξελίχθηκε σε ένα ακόμα πιο ισχυρό εργαλείο προγραμματισμού.

Ασκήσεις

1. Ποια είναι τα δεδομένα εισόδου σε έναν αλγόριθμο υπολογισμού της χωρητικότητας ενός επίπεδου πυκνωτή;
2. Περιγράψτε λεκτικά τον αλγόριθμο που υπολογίζει το ρεύμα, το οποίο διαρρέει μιαν αντίσταση, όταν δίδεται η τιμή της και η τάση στα άκρα της.
3. Γράψτε αλγόριθμο που υπολογίζει τη συνολική τιμή δύο αντιστάσεων, οι οποίες είναι συνδεδεμένες παράλληλα.
4. Πάνω σε μιαν αντίσταση υπάρχουν τα χρώματα: πορτοκαλί, πορτοκαλί, πορτοκαλί, χρυσό. Ακολουθήστε τον αλγόριθμο του παραδείγματος 2-2 βήμα-βήμα για να υπολογίσετε την τιμή της.
5. Φτιάξτε μια λίστα με 16 ονόματα ταξινομημένα αλφαβητικά. Διαλέξτε ένα όνομα και εφαρμόζοντας τον αλγόριθμο της δυαδικής αναζήτησης του παραδείγματος 2-5 αναζητήστε το μέσα στη λίστα.
6. Εξακριβώστε, αν πράγματι ο αλγόριθμος της δυαδικής αναζήτησης κάνει 20 μόνο αναζητήσεις για να ψάξει μέσα σε έναν κατάλογο με 1.000.000 άτομα.
   Υπόδειξη: Την πρώτη φορά αποκλείονται 500.000 ονόματα, τη δεύτερη άλλα 250.000 κ.ο.κ.
7. Περιγράψτε τη δομή δεδομένων με βάση την οποία γράφετε τις ετικέτες που τοποθετείτε στα τετράδιά σας.
8. Περιγράψτε δομές δεδομένων από τις καθημερινές σας εμπειρίες.

Σημιώσεις:

Μάθημα 3
Το Περιβάλλον Εργασίας

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να καλούν το περιβάλλον εργασίας.
- Να αναγνωρίζουν τα συστατικά του περιβάλλοντος εργασίας.
- Να χρησιμοποιούν την άμεση βοήθεια.
- Να τερματίζουν το περιβάλλον εργασίας.

Σε προηγούμενο μάθημα, είδαμε ότι ο προγραμματιστής πρέπει να σχεδιάσει τα παράθυρα, από τα οποία θα επικοινωνεί ο χρήστης με την εφαρμογή, να γράψει και να διορθώσει τον κώδικα του προγράμματος, να ζητήσει τη μετάφραση σε γλώσσα μηχανής και να κάνει τις απαραίτητες δοκιμαστικές εκτελέσεις του προγράμματος. Όλες αυτές οι εργασίες γίνονται σήμερα με τη βοήθεια ενοποιημένου λογισμικού, που αποτελεί ένα ***ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών*** (***Integrated Development Environment - IDE***) ή για συντομία ***περιβάλλον εργασίας***.

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών χαρακτηρίζεται από αλληλεπιδραστικότητα με τον προγραμματιστή για την υλοποίηση, την εκσφαλμάτωση και την υποστήριξη λογισμικού.

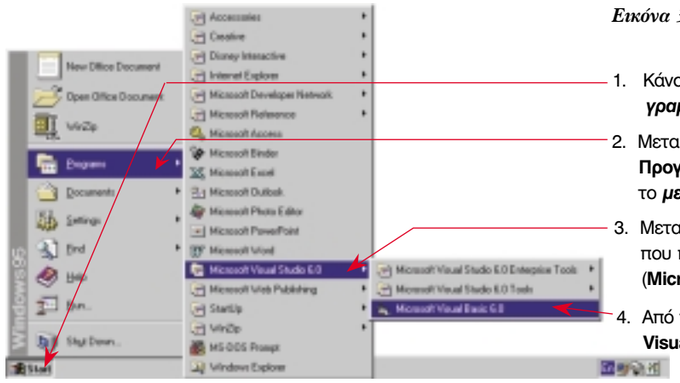
Σε ένα περιβάλλον εργασίας ο προγραμματιστής μπορεί να καλέσει παλαιότερα προγράμματά του, να αποθηκεύσει στο δίσκο τη δουλειά του, να συνδυάσει κομμάτια κώδικα από διάφορα προγράμματά του. Επίσης, με τον ενσωματωμένο επεξεργαστή κειμένου μπορεί, όχι μόνο να γράψει το πρόγραμμά του, αλλά και να το ελέγξει ορθογραφικά και συντακτικά μιας και τη στιγμή που γράφεται μια εντολή, γίνεται ο έλεγχος της ορθότητάς της και η υπόδειξη των σημείων που υπάρχουν λάθη.

Όταν τελειώσει ένα πρόγραμμα ο προγραμματιστής ζητά τη μετάφραση και την εκτέλεσή του. Μέσα από το περιβάλλον εργασίας είναι δυνατή η βηματική εκτέλεσή του προγράμματος (εντολή προς εντολή), η ανίχνευση των σημείων στα οποία δε δίνονται σωστά αποτελέσματα και η εισαγωγή σημείων διακοπής, για να διαπιστωθεί κατά πόσο το πρόγραμμα διέρχεται από συγκεκριμένα σημεία τη στιγμή της εκτέλεσής του, ενώ είναι δυνατή και η παρουσίαση ενδιάμεσων αποτελεσμάτων.

Τέλος, το περιβάλλον εργασίας είναι εφοδιασμένο με ***άμεση βοήθεια*** (***on-line help***), ώστε ο προγραμματιστής να συμβουλεύεται τα εγχειρίδια της γλώσσας προγραμματισμού σε μορφή υπερκειμένου και να εκμεταλλεύεται έτοιμα κομμάτια κώδικα που βρίσκονται γραμμένα μέσα σε παραδείγματα.

Η κλήση του περιβάλλοντος εργασίας της VB

Το περιβάλλον εργασίας της VB καλείται ακριβώς όπως και όλα τα άλλα προγράμματα στο περιβάλλον των Windows.

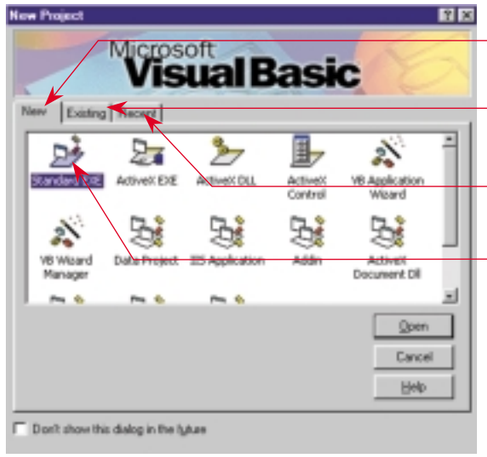


Εικόνα 3-1. *Διαδοχικά μενού για την κλήση της VB*

1. Κάνουμε κλικ στο πλήκτρο ΄**Εναρξη** (**Start**) της ***γραμμής εργασιών*** (***task bar***).
2. Μετακινούμε το ποντίκι στην επιλογή **Προγράμματα** (**Programs**), οπότε εμφανίζεται το ***μενού προγραμμάτων*** (***programs menu***).
3. Μετακινούμε το ποντίκι στην επιλογή που περιέχει την ομάδα προγραμμάτων της VB (**Microsoft Visual xxxxx**).
4. Από τα προγράμματα επιλέγουμε τη **Microsoft Visual Basic x.x**

Επιλογή έργου

Ένα περιβάλλον εργασίας το καλούμε για να δημιουργήσουμε ένα νέο πρόγραμμα ή για να συμπληρώσουμε και να αλλάξουμε ένα πρόγραμμα που έχουμε δημιουργήσει στο παρελθόν. Γι' αυτό και το πρώτο παράθυρο, που εμφανίζει το περιβάλλον εργασίας, τη στιγμή της κλήσης του, είναι ένα διαλογικό παράθυρο με το οποίο μπορούμε να κάνουμε αυτή την επιλογή. Το παράθυρο αυτό ονομάζεται **New Project** (**Νέο έργο**).



Δημιουργία νέου έργου (είναι ήδη επιλεγμένο μιας και ο καρτελοδείκτης **New** προεξέχει).

Επιλογή ενός έργου από τά ήδη υπάρχοντα.

Επιλογή ενός έργου από αυτά που επεξεργαστήκαμε πρόσφατα.

Προτεινόμενο είδος νέου έργου.

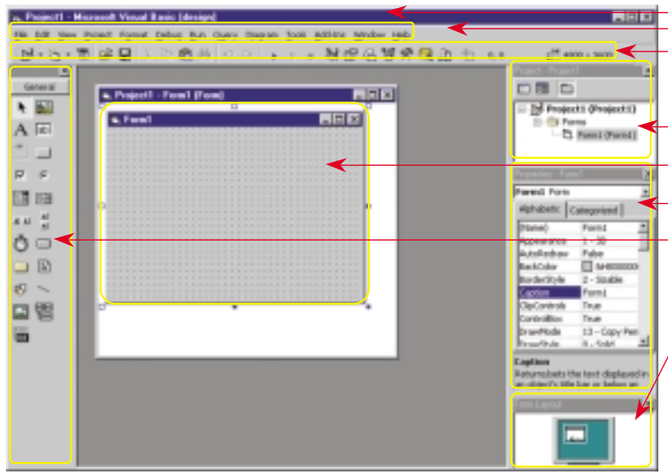
Εικόνα 3-2. *Το παράθυρο New Project για την επιλογή του έργου και του είδους του*

Δημιουργία νέου προγράμματος

Στην περίπτωση που θέλουμε να δημιουργήσουμε ένα νέο έργο πρέπει να υποδείξουμε το είδος του, ώστε να βασιστεί πάνω σε ένα συγκεκριμένο ***υπόδειγμα*** (***template***). Στο κέντρο του παραθύρου New Project υπάρχει ένα δειγματολόγιο υποδειγμάτων. Η επιλογή του υποδείγματος γίνεται κάνοντας κλικ πάνω στο αντίστοιχο εικονίδιο. Κάθε υπόδειγμα περιέχει τα βασικά δομικά στοιχεία πάνω στα οποία θα δημιουργηθεί το πρόγραμμα και τις βασικές παραμέτρους που θα ρυθμίζουν τη λειτουργία του. Για απλά αυτοδύναμα προγράμματα (stand-alone), όπως αυτά που θα κάνουμε στη διάρκεια της χρονιάς, επιλέγουμε αυτό που προτείνεται, δηλαδή το **Standard.EXE** (**Τυπικό εκτελέσιμο**).

Τα συστατικά του περιβάλλοντος εργασίας

Μετά την επιλογή έργου η οθόνη καλύπτεται με το περιβάλλον εργασίας της VB. Η πρώτη εντύπωση που προκαλεί η θέα του στο νέο προγραμματιστή είναι η αμηχανία. Μπροστά του βλέπει ένα αρκετά πολύπλοκο περιβάλλον που φαντάζεται ότι δε θα μπορέσει να το διαχειριστεί.



Γραμμή Τίτλου

Γραμμή μενού

Γραμμή εργαλείων

Παράθυρο έργου

Σχεδιαστής φόρμας

Παράθυρο ιδιοτήτων

Εργαλειοθήκη

Σκαρίφημα εμφάνισης παραθύρων στην οθόνη

Εικόνα 3-3. *Το περιβάλλον εργασίας της VB όπως φαίνεται στην οθόνη αμέσως μετά την κλήση του*

Μετά όμως από λίγα λεπτά αναγνώρισης των παραθύρων που το αποτελούν και μετά από κάποια σύντομη εξερεύνηση, το περιβάλλον αρχίζει να του φαίνεται αρκετά εργονομικό και λειτουργικό. Το περιβάλλον εργασίας της VB αποτελείται από τα εξής δομικά στοιχεία:

Γραμμή τίτλου (Title bar)

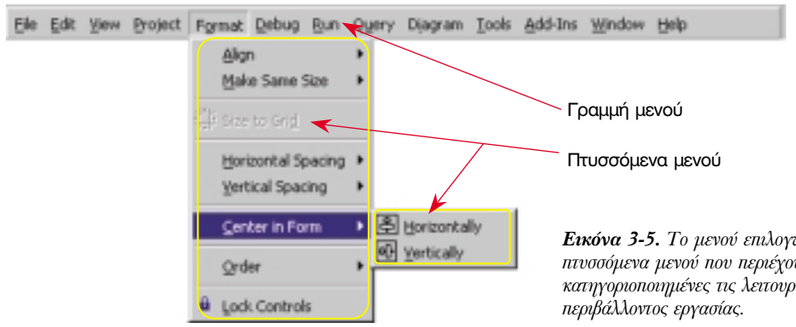
Περιέχει τον τίτλο του παραθύρου του περιβάλλοντος εργασίας και τα πλήκτρα διαχείρισης της μορφής του παραθύρου. Ο τίτλος αποτελείται από τρία συνθετικά. Το πρώτο συνθετικό είναι το όνομα του έργου (στο σχήμα 3-4 το όνομα του έργου είναι `Project1`). Ακολουθεί το όνομα του περιβάλλοντος εργασίας (Microsoft Visual Basic). Στο τέλος δηλώνεται η κατάσταση, στην οποία βρίσκεται το περιβάλλον εργασίας (`[design]` - υπό σχεδιασμό).

Εικόνα 3-4. *Γραμμή τίτλου*

Τη στιγμή της σχεδίασης και της γραφής του κώδικα το περιβάλλον εργασίας βρίσκεται σε κατάσταση ***σχεδιασμού*** (***design***). Όπως θα δούμε αργότερα, εκτός από αυτή την κατάσταση το περιβάλλον εργασίας μπορεί να βρίσκεται σε κατάσταση ***εκτέλεσης*** (***run***), και σε ***κατάσταση διακοπής*** (***break***).

Γραμμή μενού (Menu bar)

Η γραμμή μενού βρίσκεται κάτω από τη γραμμή τίτλου. Από τη γραμμή μενού εκτελούνται όλες οι λειτουργίες οι σχετικές με την ανάπτυξη των εφαρμογών και καθορίζονται οι τιμές των παραμέτρων λειτουργίας της VB. Το μενού του περιβάλλοντος εργασίας έχει ακριβώς τα ίδια χαρακτηριστικά με όλα τα μενού στα Windows. Ο χειρισμός του γίνεται είτε με το ποντίκι είτε από το πληκτρολόγιο με το πάτημα του πλήκτρου `Alt` και του αντίστοιχου υπογραμμισμένου γράμματος της επιλογής (π.χ. `Alt + F` για την επιλογή **File**, `Alt + V` για την επιλογή **View**



Γραμμή μενού

Πτυσσόμενα μενού

Εικόνα 3-5. *Το μενού επιλογών εμφανίζει πτυσσόμενα μενού που περιέχουν κατηγοριοποιημένες τις λειτουργίες του περιβάλλοντος εργασίας.*

Η γραμμή μενού περιέχει τις βασικές επιλογές από τις οποίες "ξετυλίγονται" πτυσσόμενα μενού με πρόσθετες σχετικές επιλογές. Κάποιες μάλιστα από τις επιλογές της (π.χ. οι **File, Edit, View, Window, Help**) είναι οι ίδιες με αυτές άλλων προγραμμάτων λογισμικού που έχετε δουλέψει. Οι πιο σημαντικές επιλογές που περιέχει η γραμμή μενού είναι οι εξής:

File: Περιέχει τις υποεπιλογές για τη διαχείριση των αρχείων του έργου. Από αυτές τις υποεπιλογές είναι δυνατή η δημιουργία νέων έργων (**New**), η κλήση έργων που βρίσκονται αποθηκευμένα στους δίσκους (**Open**), η διαγραφή έργων (**Remove**), η αποθήκευση των έργων σε αρχεία (**Save**), οι ρυθμίσεις των παραμέτρων εκτύπωσης (**Print Setup**), οι εκτυπώσεις (**Print**), η μετάφραση και η σύνδεση όλων των αρχείων του έργου σε ένα εκτελέσιμο αρχείο (**Make exe**).

Edit: Περιέχει τις υποεπιλογές για την επιμέλεια και επεξεργασία του κώδικα του έργου και των αντικειμένων που υπάρχουν πάνω στη φόρμα εργασίας. Οι υποεπιλογές ανάκλησης (**Undo**), αντιγραφής (**Copy**), αποκοπής (**Cut**), επικόλλησης (**Paste**), διαγραφής (**Delete** και **Remove**), επιλογής κειμένου και αντικειμένων (**Select**), αναζήτησης (**Find**), αντικατάστασης (**Replace**), εσοχής (**Indent**) και προεξοχής κειμένου (**Outdent**), εισαγωγής κειμένου από αρχεία (**Insert**), τοποθέτησης σελιδοδεικτών (***Bookmarks***), παρουσίασης παραμέτρων (**List**), πινάκων κ.ά. βρίσκονται στο πτυσσόμενο μενού αυτής της επιλογής.

Μην προσπαθήσετε να αποστηθίσετε τις επιλογές και τις υποεπιλογές τους. Αυτό θα γίνει σιγά-σιγά με τη χρήση του περιβάλλοντος εργασίας. Πρέπει όμως να γνωρίζετε σε ποια επιλογή θα ψάξετε αρχικά για να εκτελέσετε κάποιο είδος εργασίας.

View: Η εμφάνιση και η εξαφάνιση των παραθύρων και των εργαλειοθηκών (**Form, Code, Project Explorer, Watch**) του περιβάλλοντος εργασίας καθορίζεται από τις υποεπιλογές του πτυσσόμενου μενού της.

Project: Περιέχει υποεπιλογές για την προσθήκη νέων στοιχείων στο έργο (**Add**), όπως φορμών και αντικειμένων, καθώς και υποεπιλογές για επισκόπηση των παραμέτρων των έργων και των αναφορών σε βιβλιοθήκες λογισμικού (**References**) και αντικειμένων (**Components**) που μπορούν να τοποθετηθούν πάνω σε φόρμες.

Format: Από αυτή την επιλογή καλούνται υποεπιλογές που ευθυγραμμίζουν (**Align**) τα αντικείμενα πάνω στις φόρμες, ρυθμίζουν το μέγεθος τους (**Size**), καθορίζουν τη σειρά τοποθέτησής τους (**Order**) και τη μεταξύ τους απόσταση (**Spacing**).

Debug: Χρησιμοποιείται κατά την εκσφαλμάτωση του προγράμματος. Περιέχει υποεπιλογές για βηματική εκτέλεση εντολή προς εντολή (**Step**), τοποθέτηση σημείων διακοπής (**Breakpoint**), παρουσίαση παραθύρων από τα οποία γίνεται έλεγχος μεταβλητών (**Watch**), δημιουργία αλμάτων μέσα στον κώδικα (**Next Statement**).

Run: Περιέχει τις υποεπιλογές εκκίνησης (**Start**), διακοπής (**Break**), τερματισμού (**End**) και επανεκκίνησης (**Restart**) του προγράμματος.

Tools: από τις υποεπιλογές που περιέχει ξεχωρίζουν η υποεπιλογή για τη δημιουργία μενού (**Menu editor**) στις φόρμες των εφαρμογών και η υποεπιλογή για τον καθορισμό των παραμέτρων λειτουργίας του περιβάλλοντος εργασίας (**Options**).

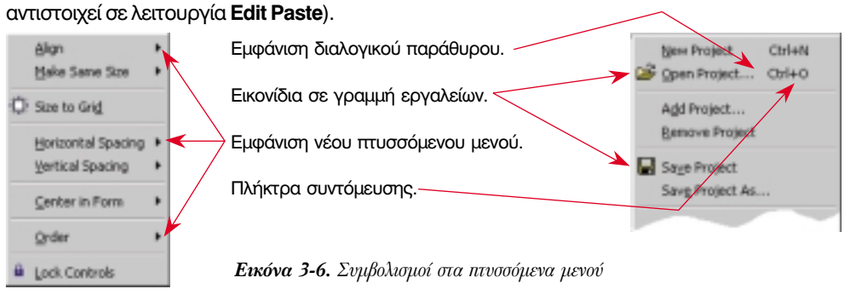
Add-ins: Περιέχει μια από τις πιο βασικές υποεπιλογές (**Add-in Manager**) με την οποία είναι δυνατή η προσθήκη πρόσθετων εργαλείων στο περιβάλλον εργασίας. Με τα επιπλέον εργαλεία είναι δυνατή η διαχείριση βάσεων δεδομένων, η εκσφαλμάτωση των διαδικασιών ερωτήσεων προς βάσεις δεδομένων, η παροχή πληροφοριών για τις βιβλιοθήκες του λογισμικού συστήματος (Application Programming Interface - API). Τα πρόσθετα εργαλεία που θα εισαχθούν με τον Add-in Manager εισάγονται αυτόματα σαν υποεπιλογές σε αυτό το πτυσσόμενο μενού.

Window: Οι υποεπιλογές που υπάρχουν αναδιατάσσουν (**Tile, Arrange**) και υποδιαιρούν (**Split**) τα ανοιχτά παράθυρα του περιβάλλοντος εργασίας.

Help: Περιέχει υποεπιλογές με τις οποίες γίνεται η κλήση των παραθύρων της άμεσης βοήθειας. Ο προγραμματιστής μπορεί να ψάξει ως προς τα περιεχόμενα (**Contents**), τα ευρετήρια όρων (**Index**), τις λέξεις κλειδιά (**Search**). Ακόμα περιέχει υποεπιλογές για τη σύνδεση με ιστοσελίδες του Παγκόσμιου Ιστού (**Microsoft on the Web**) και για επιπλέον αναζήτηση θεμάτων και ενημέρωση με τις πιο πρόσφατες εξελίξεις στο χώρο του προγραμματισμού και της VB.

Αν μια υποεπιλογή μέσα στο πτυσσόμενο μενού έχει στο τέλος της τρεις τελείες, η επιλογή της εμφανίζει ένα διαλογικό παράθυρο (π.χ. η υποεπιλογή **Save As ...** εμφανίζει διαλογικό παράθυρο για τον καθορισμό από το χρήστη του αρχείου στο οποίο θα γίνει αποθήκευση). Επίσης, αν η υποεπιλογή έχει στο τέλος της το σύμβολο u εμφανίζει στο πλάι της ένα νέο πτυσσόμενο μενού με υπο-υποεπιλογές.

Μπροστά από κάποιες υποεπιλογές υπάρχει το εικονίδιο που τις αντιπροσωπεύει στις γραμμές εργαλείων και στο τέλος κάποιων υποεπιλογών αναφέρονται οι συνδυασμοί πλήκτρων, ***πλήκτρα συντόμευσης*** (***sortcut keys***) που μπορούν να χρησιμοποιηθούν εναλλακτικά για την εκτέλεση των λειτουργιών που τους αντιστοιχούν (π.χ. ο συνδυασμός πλήκτρων `Ctrl + C` αντιστοιχεί σε λειτουργία **Edit Copy** και ο συνδυασμός `Ctrl + V` αντιστοιχεί σε λειτουργία **Edit Paste**).

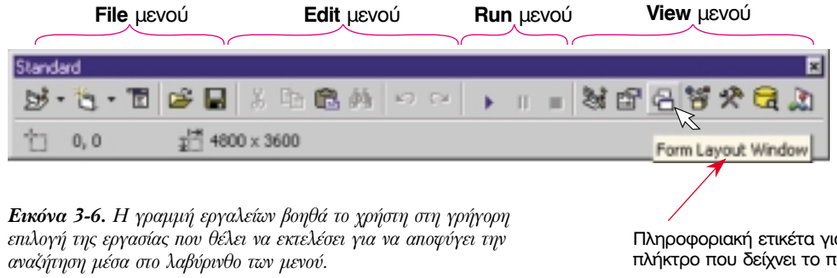


Εικόνα 3-6. *Συμβολισμοί στα πτυσσόμενα μενού*

Όλες οι υποεπιλογές έχουν ένα γράμμα τους υπογραμμισμένο. Όταν το πτυσσόμενο μενού στο οποίο ανήκουν είναι ανοιχτό, το πάτημα του πλήκτρου που αντιστοιχεί στο υπογραμμισμένο γράμμα, προκαλεί την εκτέλεση της λειτουργίας που αντιστοιχεί στην επιλογή. Έτσι πατώντας τα πλήκτρα `Alt + O` και στη συνέχεια `A` και `L`, επιλέγουμε **Format**, οπότε ανοίγει το πτυσσόμενο μενού, στη συνέχεια επιλέγουμε **Align** και **Left**.

Γραμμές Εργαλείων (Toolbars)

Στο περιβάλλον εργασίας μπορεί να υπάρχει μια ή περισσότερες γραμμές εργαλείων. Οι γραμμές εργαλείων λειτουργούν παρόμοια με τις αντίστοιχες άλλων εφαρμογών των Windows, όπως για παράδειγμα των επεξεργαστών κειμένου (π.χ. Word) και των υπολογιστικών φύλλων (π.χ. Excel). Αποτελούνται από πλήκτρα-εικονίδια των οποίων το πάτημα προκαλεί την εκτέλεση των λειτουργιών που συμβολίζει το εικονίδιο τους. Οι λειτουργίες είναι ομαδοποιημένες σε κατηγορίες και αντιστοιχούν στις πιο κοινές της γραμμής μενού.



Εικόνα 3-6. *Η γραμμή εργαλείων βοηθά το χρήστη στη γρήγορη επιλογή της εργασίας που θέλει να εκτελέσει για να αποφύγει την αναζήτηση μέσα στο λαβύρινθο των μενού.*

Ας σημειωθεί ότι, αν τοποθετήσουμε το δείκτη του ποντικιού πάνω σε ένα από τα εργαλεία της γραμμής εργαλείων και τον αφήσουμε για λίγο ακίνητο, χωρίς να πατήσουμε κάποιο από τα πλήκτρα του ποντικιού, θα εμφανιστεί μια μικρή ***πληροφοριακή ετικέτα*** (***tool tip***) που περιγράφει τη λειτουργία του πλήκτρου.

Για να εμφανίζονται οι πληροφοριακές ετικέτες, επιλέγουμε ***Tools*** ¦ ***Options***, τον καρτελοδείκτη **Environment** και μαρκάρουμε το πλαίσιο σημείωσης **Show Tooltips**.

Σχεδιαστής Φόρμας (Form Designer)

Στο σχεδιαστή φόρμας ο χρήστης δημιουργεί τις φόρμες των παραθύρων επικοινωνίας του χρήστη με την εφαρμοφή. Όπως θα δούμε, οι φόρμες αποτελούν το σημείο από το οποίο αρχίζει ο σχεδιασμός και η υλοποίηση κάθε εφαρμογής.

Όταν ξεκινάμε μια νέα εργασία, το περιβάλλον παρουσιάζει μια κενή φόρμα σχεδίασης (με το όνομα Form1), η οποία, όπως και κάθε νέα φόρμα, περιέχει ***γραμμή τίτλου*** (***Title Bar***), ***πλαίσιο ελέγχου*** (***Control Box***) ***πλήκτρα επαναφοράς/μεγιστοποίησης*** (***Restore/Maximize***), ***ελαχιστοποίησης*** (***Minimize***) και ***κλεισίματος*** (***Close***).

Εικόνα 3-7. *Η φόρμα αποτελεί το υπόβαθρο πάνω στο οποίο χτίζεται η επικοινωνία χρήστη - εφαρμογής.*

Το πλέγμα των σημείων που παρουσιάζονται στο εσωτερικό της φόρμας αποτελεί βοήθημα για το σχεδιασμό των αντικειμένων. Το πλέγμα εμφανίζεται μόνο τη στιγμή που το περιβάλλον εργασίας βρίσκεται σε κατάσταση σχεδιασμού. Κατά την εκτέλεση του προγράμματος το πλέγμα γίνεται αόρατο.

Στη φόρμα σχεδίασης επισυνάπτονται από τον προγραμματιστή τα αντικείμενα ελέγχου που παρουσιάζουν δεδομένα στο χρήστη (π.χ. πλαίσια κειμένου) και αντικείμενα ελέγχου χειρισμών (π.χ. πλήκτρα). Ο προγραμματιστής δημιουργεί όσες φόρμες σχεδίασης απαιτεί το πρόγραμμά του, καθορίζει τις διαστάσεις τους και τις τοποθετεί στην κατάλληλη θέση. Η θέση και οι διαστάσεις που δίνονται στη φόρμα κατά το σχεδιασμό της είναι αυτές που θα έχει και κατά την εκτέλεση του προγράμματος.

Εργαλειοθήκη (ToolBox)

Το πιο σημαντικό χαρακτηριστικό της VB είναι ότι ο προγραμματιστής δεν είναι αναγκασμένος να δημιουργήσει δικά του αντικείμενα, αλλά μπορεί να χρησιμοποιήσει κάποια από τα διαθέσιμα. Η εργαλειοθήκη περιέχει όλα τα εργαλεία για το σχεδιασμό και την τοποθέτηση ***αντικειμένων ελέγχου*** (***controls***) πάνω στις φόρμες. Με πολύ απλές κινήσεις ο προγραμματιστής-σχεδιαστής μπορεί να δημιουργήσει σύνθετες αλλά εργονομικές και λειτουργικές φόρμες και να συνθέσει το κατάλληλο περιβάλλον επικοινωνίας χρήστη - εφαρμογής.

Κάποια από τα αντικείμενα της εργαλειοθήκης είναι ήδη γνωστά σε ένα χρήστη των Windows, μιας και είναι τα ίδια με αυτά που εμφανίζονται στα διαλογικά παράθυρα των εφαρμογών. Για παράδειγμα, τα πεδία κειμένου, οι συνδυασμένες και οι απλές λίστες, τα πλήκτρα, τα κουτάκια μαρκαρίσματος, οι ράβδοι κύλισης, τα πλαίσια επιλογής δίσκων, καταλόγων, αρχείων.

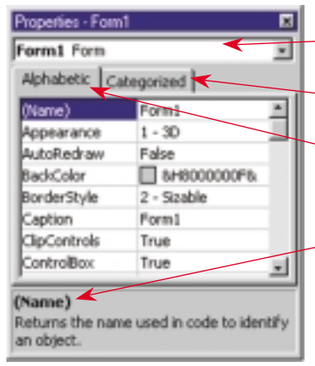
Για να τοποθετήσει ο προγραμματιστής ένα αντικείμενο ελέγχου στη φόρμα σχεδίασης, επιλέγει πρώτα το εικονίδιο, που το αναπαριστά στην εργαλειοθήκη, και στη συνέχεια το σχεδιάζει στις κατάλληλες διαστάσεις πάνω στη φόρμα. Μετά, όπως θα δούμε παρακάτω, του δίνει ιδιότητες και καθορίζει τον ειδικό τρόπο συμπεριφοράς του γράφοντας κώδικα στη γλώσσα Basic. Περισσότερα για τα εργαλεία σχεδιασμού θα δούμε σε προσεχές μάθημα.

Εικόνα 3-8. *Η εργαλειοθήκη*

Παράθυρο Ιδιοτήτων (Properties window)

Οι ιδιότητες είναι τα ιδιαίτερα χαρακτηριστικά των αντικειμένων. Όπως θα δούμε, κάποιες από τις ιδιότητες καθορίζουν τα χαρακτηριστικά της εμφάνισης των αντικειμένων, όπως τη θέση, το μέγεθος, το χρώμα, το ίχνος της μύτης του "μολυβιού" που χρησιμοποιείται για τη σχεδίαση τους. Επίσης, κάποιες άλλες ιδιότητες καθορίζουν τον τρόπο αντίδρασης και συμπεριφοράς των αντικειμένων, όπως κλείδωμα, ενεργοποίηση, είδος επικοινωνίας για ανταλλαγές τιμών κ.ά. Κάθε είδος αντικειμένου έχει διαφορετικό σύνολο ιδιοτήτων.

Οι φόρμες και τα αντικείμενα ελέγχου είναι είδη αντικειμένων που χαρακτηρίζονται από ιδιότητες.



Το αντικείμενο το οποίο χαρακτηρίζουν οι ιδιότητες στη λίστα που ακολουθεί

Ταξινόμηση των ιδιοτήτων κατά κατηγορία

Αλφαβητική ταξινόμηση όλων των ιδιοτήτων

Σύντομη περιγραφή της ιδιότητας που έχει επιλεγεί

Εικόνα 3-9. *Στο παράθυρο ιδιοτήτων μπορούμε να δούμε και να αλλάξουμε τις τιμές κάποιων ιδιοτήτων των αντικειμένων.*

Το παράθυρο ιδιοτήτων περιέχει μια λίστα με τις ιδιότητες και τις τιμές των ιδιοτήτων του αντικειμένου που έχουμε επιλέξει. Η επιλογή του αντικειμένου μπορεί να γίνει είτε από τη

φόρμα σχεδίασης, είτε από την πτυσσόμενη λίστα που βρίσκεται στο πάνω μέρος του παραθύρου. Στην περίπτωση που έχουν επιλεγεί πάνω από ένα αντικείμενα, προβάλλονται μόνο οι κοινές ιδιότητες όλων των αντικειμένων. Η παρουσίαση των ιδιοτήτων μέσα στο παράθυρο μπορεί να γίνει ή αλφαβητικά ή κατά κατηγορίες. Τέλος, ας σημειωθεί ότι υπάρχουν και ιδιότητες, οι οποίες δεν αναφέρονται στο παράθυρο ιδιοτήτων.

Παράθυρο ΄Εργου (Project Explorer window)

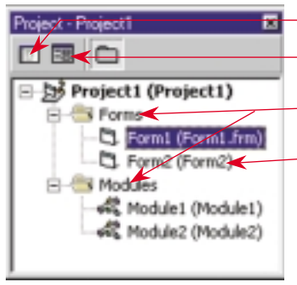
Είδαμε ότι κάθε εφαρμογή της VB αποτελεί ένα έργο. Επίσης, είδαμε ότι σε ένα έργο μπορεί να έχουμε μια ή περισσότερες φόρμες. Σε προσεχή μαθήματα θα δούμε ότι ο κώδικας σε γλώσσα Basic ενός έργου, δε γράφεται κάπου μονοκόμματος, αλλά σε τμήματα στις ***προγραμματιστικές μονάδες φόρμας*** (***form modules***), που καθεμιά τους συνοδεύει μια φόρμα, και στις ***βασικές προγραμματιστικές μονάδες*** (***standard modules*** ή απλά ***modules***) που δεν ανήκουν σε καμιά φόρμα.

Για κάθε έργο δημιουργείται ένα αρχείο, που περιγράφει τον τρόπο δόμησής του και περιέχει και άλλες λειτουργικές του πληροφορίες. Τα αρχεία τύπου έργου έχουν κατάληξη `.VBP`. Για κάθε φόρμα δημιουργείται και ένα αρχείο, που περιγράφει τον τρόπο δόμησής της και περιέχει τα αντικείμενα ελέγχου της και τον κώδικα σε γλώσσα Basic που συνοδεύει τη φόρμα. Κάθε αρχείο φόρμας έχει κατάληξη `.FRM`. Τέλος, οι βασικές προγραμματιστικές μονάδες αποθηκεύονται και αυτές σε αρχεία. Τα αρχεία των βασικών προγραμματιστικών μονάδων έχουν κατάληξη `.BAS`. Το περιεχόμενο και των τριών ειδών αρχείων μπορούμε να το δούμε και με απλό επεξεργαστή κειμένου.


Εικονίδιο για τα αρχεία .VBP


Εικονίδιο για τα αρχεία .FRM


Εικονίδιο για τα αρχεία .BAS



Πλήκτρο για εμφάνιση φόρμας

Πλήκτρο για εμφάνιση κώδικα

Διπλό κλικ για να αναδιπλωθεί ή να ξεδιπλωθεί η λίστα

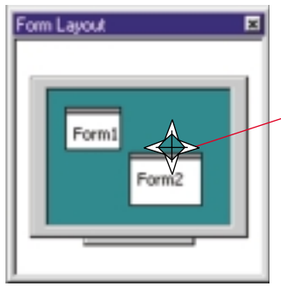
Διπλό κλικ για να εμφανιστεί η φόρμα

Εικόνα 3-10. *Στο παράθυρο έργου τα δομικά στοιχεία του έργου παρουσιάζονται όπως και στον εξερευνητή.*

Τα δομικά στοιχεία του έργου, η ιεραρχική σύνδεσή τους και τα αρχεία στα οποία έχουν αποθηκευτεί, παρουσιάζονται σε μορφή παρόμοια με αυτή του εξερευνητή αρχείων των Windows. Αυτοί οι τύποι δομικών στοιχείων ενός έργου δεν είναι και οι μοναδικοί. Για πιο σύνθετες εφαρμογές χρησιμοποιούνται και άλλες μορφές φορμών και προγραμματιστικών μονάδων. Τέλος, ας σημειωθεί, ότι μια και κάθε δομικό στοιχείο του έργου αποθηκεύεται και σε ξεχωριστό αρχείο, είναι δυνατόν ένα δομικό στοιχείο να ανήκει σε περισσότερα από ένα έργα.

Σκαρίφημα Εμφάνισης Παραθύρων (Form Layout Window)

Πρόκειται για ένα παράθυρο που παρουσιάζει σε μικρογραφία την οθόνη και τη σχετική θέση κάθε παραθύρου ανοιχτής φόρμας στο εσωτερικό της οθόνης. Χρησιμεύει στην προεπισκόπηση των σχετικών διαστάσεων του παραθύρου κάθε φόρμας και την τοποθέτησή του μέσα στο χώρο της οθόνης σε σχέση με τα άλλα παράθυρα που τυχαίνει να είναι ανοιχτά.



Όταν ο δείκτης του ποντικιού πάρει τη μορφή σταυρονήματος μπορούμε να σύρουμε το παράθυρο και να το τοποθετήσουμε σε νέα θέση.

Εικόνα 3-11. *Η οθόνη και η σχετική θέση δύο παραθύρων που αντιστοιχούν σε φόρμες.*

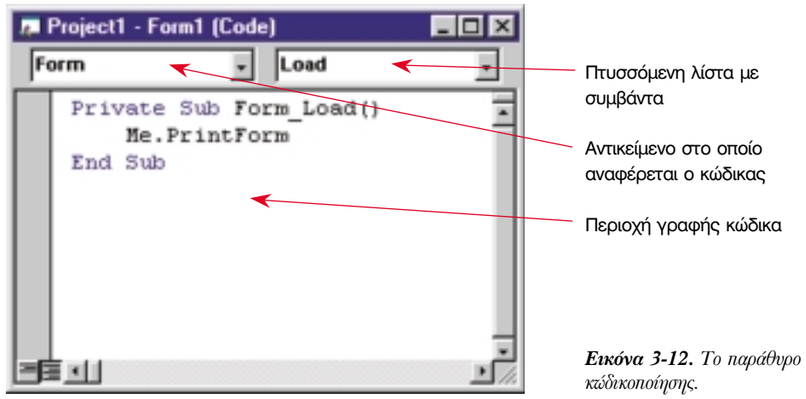
Άλλα παράθυρα του περιβάλλοντος εργασίας

Εκτός από τα παράθυρα που περιγράψαμε πιο πάνω, το περιβάλλον εργασίας διαθέτει και άλλα παράθυρα, τα οποία αρχικά βρίσκονται αναδιπλωμένα. Η εμφάνιση και η απόκρυψη των πρόσθετων παραθύρων, καθώς και κάθε παραθύρου, μπορεί να γίνει από τις αντίστοιχες υποεπιλογές της επιλογής **View** του μενού.

Από την επιλογή **View** γίνεται η εμφάνιση και η απόκρυψη των παραθύρων του περιβάλλοντος εργασίας.

Παράθυρο Κώδικα (Code window)

Για κάθε προγραμματιστική μονάδα φόρμας και για κάθε βασική προγραμματιστική μονάδα υπάρχει και ένα παράθυρο κώδικα, ώστε να γίνεται καλύτερα η οργάνωση και η διαχείριση του κώδικα. Στο παράθυρο κώδικα γίνεται η εισαγωγή και η διαμόρφωση του πηγαίου κώδικα της εφαρμογής. Στο παράθυρο αυτό γράφουμε, όπως ακριβώς γράφουμε και σε έναν επεξεργαστή κειμένου. Μπορούμε να πληκτρολογήσουμε κώδικα, να σβήσουμε τμήματά του, να τα μετακινήσουμε σε άλλη θέση, να αντιγράψουμε κείμενο από άλλο παράθυρο κώδικα ή από άλλο επεξεργαστή κειμένου.



Εικόνα 3-12. *Το παράθυρο κώδικοποίησης.*

Κάθε γραμμή που εισάγεται στο παράθυρο κώδικα ελέγχεται για συντακτικά λάθη. Σε περίπτωση που βρεθεί ένα τέτοιο λάθος γίνεται αυτόματα η αναφορά του και η γραμμή χρωματίζεται κόκκινη. Αν η γραμμή δεν περιέχει λάθη, χρωματίζονται οι κωδικές λέξεις μπλε και γίνεται κατάλληλος διαχωρισμός όλων των λέξεων με κενά.

Παράθυρο Άμεσης Εκτέλεσης Εντολών (Immediate window)

Σε αυτό το παράθυρο ο προγραμματιστής μπορεί να ζητήσει την άμεση εκτέλεση κάποιων εντολών, ακόμη κι αν αυτές δεν ανήκουν στον κώδικα του προγράμματος. Σκοπός του προγραμματιστή είναι να ελέγξει τις τιμές των μεταβλητών του προγράμματός του κατά την εκσφαλμάτωση, ή να αλλάξει τις τιμές κάποιων μεταβλητών με στόχο να συντομεύσει την εκτέλεση κάποιων λειτουργιών, ή να κάνει κάποιους πρόχειρους υπολογισμούς παραστάσεων.

Παράθυρο Τοπικών Μεταβλητών (Locals window)

Πρόκειται για ένα παράθυρο που σε μορφή πίνακα παρουσιάζει τις τιμές των εσωτερικών (τοπικών) μεταβλητών μιας υπορουτίνας. Χρησιμοποιείται συνήθως κατά την εκσφαλμάτωση. Από αυτό το παράθυρο ο προγραμματιστής μπορεί να έχει μια ολοκληρωμένη εικόνα των μεταβλητών που χρησιμοποιεί για να δει, πώς μεταβάλλονται και αλληλεπιδρούν μεταξύ τους.

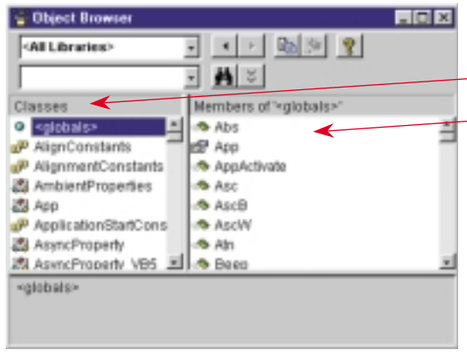
Υπορουτίνα είναι ένα τμήμα προγράμματος.

Παράθυρο Παρακολούθησης (Watch window)

Και αυτό είναι ένα παράθυρο που χρησιμοποιείται συνήθως κατά την εκσφαλμάτωση. Προβάλλει επιλεκτικά τις τιμές κάποιων μεταβλητών και παραστάσεων που έχει υποδείξει ο προγραμματιστής.

Παράθυρο Επισκόπησης Αντικειμένων (Object Browser)

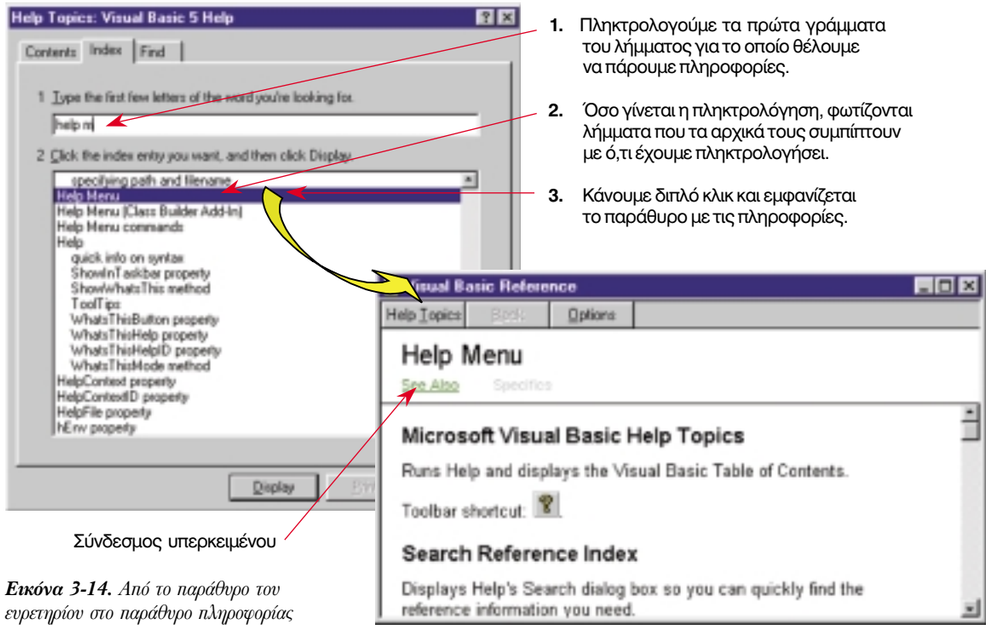
Στο παράθυρο Επισκόπησης Αντικειμένων υπάρχει μια λίστα με όλα τα αντικείμενα του έργου ταξινομημένα. Δείχνοντας τα αντικείμενα στο παράθυρο επισκόπησης γίνεται και ταχεία πλοήγηση στα παράθυρα κώδικα που τα περιέχουν.



Εικόνα 3-13. *Παράθυρο επισκόπησης αντικειμένων*

Άμεση Βοήθεια

Ένα από τα βασικά χαρακτηριστικά του περιβάλλοντος εργασίας είναι η δυνατότητα παροχής άμεσης βοήθειας στον προγραμματιστή. Ο τρόπος χρήσης του περιβάλλοντος εργασίας, οι τρόποι σύνταξης των εντολών της γλώσσας Basic, παραδείγματα και άλλα θέματα σχετικά με τον προγραμματισμό είναι οργανωμένα σε δομή ***υπερκειμένου*** (***hypertext***). Μέσα στο κείμενο υπάρχουν ειδικά τονισμένες λέξεις (πράσινες στο χρώμα), πάνω από τις οποίες όταν περάσει ο δρομέας του ποντικιού αποκτά μορφή χεριού που δείχνει. Αν ο προγραμματιστής κάνει κλικ επάνω τους, ανοίγει νέο παράθυρο με πρόσθετη πληροφορία γι' αυτές τις λέξεις. Αυτός ο τρόπος οργάνωσης του κειμένου έχει αποδειχθεί ο πιο ενδεδειγμένος για εγχειρίδια αναφοράς μιας και ο προγραμματιστής μπορεί να ξεκινήσει από ένα σημείο, να κινηθεί από πληροφορία σε πληροφορία και να βρει την πληροφορία που τον ενδιαφέρει συσχετιστικά.



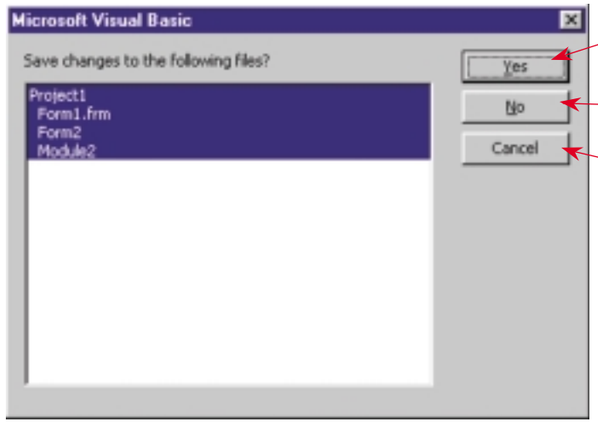
Εικόνα 3-14. *Από το παράθυρο του ευρετηρίου στο παράθυρο πληροφορίας*

Για να καλέσει την άμεση βοήθεια ο προγραμματιστής έχει δύο δυνατότητες:

- Να κάνει κλικ στην επιλογή **Help** της γραμμής μενού και να ψάξει ως προς τα περιεχόμενα (υποεπιλογή **Contents**), ως προς τα ευρετήρια όρων (υποεπιλογή **Index**), ή ως προς κωδικές λέξεις (υποεπιλογή **Search**).
- Να εστιάσει στην οθόνη στο αντικείμενο, για το οποίο χρειάζεται να πάρει πληροφορίες κάνοντας κλικ επάνω του και μετά να πατήσει το πλήκτρο `F1`. Για παράδειγμα, αν θέλουμε να ζητήσουμε πληροφορίες για ένα παράθυρο κάνουμε κλικ κάποιου μέσα στο παράθυρο και πατάμε το πλήκτρο `F1`. Αν πάλι θέλουμε πληροφορίες για ένα εικονίδιο της εργαλειοθήκης, κάνουμε κλικ πάνω στο εικονίδιο και πατάμε το πλήκτρο `F1`. Τέλος, αν τη στιγμή που γράφουμε κώδικα, θέλουμε πληροφορίες για μιαν εντολή, κάνουμε κλικ στην κωδική της λέξη και πατάμε το πλήκτρο `F1`.

Η έξοδος από το περιβάλλον εργασίας

Για να τερματίσουμε το περιβάλλον εργασίας της VB επιλέγουμε την επιλογή **File** από το μενού και στη συνέχεια την υποεπιλογή **Exit**. Αν δεν έχουμε αποθηκεύσει τις αλλαγές που έχουμε κάνει στο τρέχον έργο, πριν τον τερματισμό εμφανίζεται ένα διαλογικό παράθυρο από το οποίο ερωτώμαστε αν επιθυμούμε, να τις αποθηκεύσουμε.



Αποθήκευση των αλλαγών και μετά τερματισμός. Ισοδυναμεί και με το πάτημα του πλήκτρου `Enter`.

Τερματισμός χωρίς αποθήκευση των αλλαγών.

Αναστολή του τερματισμού. Ισοδυναμεί και με το πάτημα του πλήκτρου `Esc`.

Εικόνα 3-15. *Το διαλογικό παράθυρο με το οποίο η VB μας ερωτά για την αποθήκευση των αρχείων τη στιγμή του τερματισμού της.*

Ανακεφαλαίωση

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών χαρακτηρίζεται από αλληλεπιδραστικότητα με τον προγραμματιστή για τη υλοποίηση, την εκσφαλμάτωση και την υποστήριξη λογισμικού. Τα συστατικά του περιβάλλοντος εργασίας είναι:

- η γραμμή τίτλου που περιέχει και τα πλήκτρα διαχείρισης της μορφής του παραθύρου,
- η γραμμή μενού, από την οποία εκτελούνται όλες οι λειτουργίες οι σχετικές με την ανάπτυξη των εφαρμογών και γίνεται ο καθορισμός των παραμέτρων της VB,
- οι γραμμές εργαλείων που περιέχουν πλήκτρα συντόμευσης των λειτουργιών της γραμμής μενού,
- οι φόρμες, στις οποίες σχεδιάζεται ο τρόπος επικοινωνίας χρήστη - εφαρμογής,
- η εργαλειοθήκη των αντικειμένων που τοποθετούνται πάνω στις φόρμες,
- το παράθυρο ιδιοτήτων των αντικειμένων,
- το παράθυρο έργου,
- το σκαρίφημα εμφάνισης παραθύρων.

Ακόμη στο περιβάλλον εργασίας μπορούν να εμφανιστούν τα παράθυρα κώδικα, άμεσης εκτέλεσης εντολών, μεταβλητών, επισκόπησης αντικειμένων.

Εργαστηριακές Ασκήσεις

1. Καλέστε το περιβάλλον εργασίας εκτελώντας βήμα βήμα, όσα περιγράψαμε σε αυτό το μάθημα.
2. Βρείτε άλλους τρόπους κλήσης του περιβάλλοντος της VB.
   Υπόδειξη:
   α) Από τον Explorer των Windows Αναζητήστε τον φάκελο στον οποίο υπάρχει το εκτελέσιμο αρχείο του περιβάλλοντος εργασίας. Το όνομα του αρχείου είναι συνήθως `VBx.EXE`, όπου `x` η έκδοση της VB. Με διπλό κλικ επάνω στο όνομα του αρχείου καλείται η VB.
   β) Δημιουργήστε εικονίδιο συντόμευσης για την VB.
3. Αναγνωρίστε τα παράθυρα που εμφανίζονται και όσα από τα στοιχεία τους έχουμε περιγράψει.
4. Κάντε μια σύντομη επισκόπηση στις επιλογές της γραμμής μενού και προσπαθήστε να αναγνωρίσετε υποεπιλογές που είναι κοινές με υποεπιλογές μενού άλλων προγραμμάτων που γνωρίζετε (π.χ. Word ή Excel).
5. Εκτός από τη γραμμή εργαλείων είναι δυνατόν να εμφανίσουμε και αναδυόμενα περιεκτικού περιεχομένου ***μενού συντόμευσης*** (***shortcut menus***). Αυτά τα μενού περιέχουν από τις επιλογές της γραμμής μενού μόνο τις πιο βασικές σε σχέση με το αντικείμενο που δείχνει το ποντίκι. Τοποθετήστε το δείκτη του ποντικιού σε διάφορα παράθυρα, εργαλειοθήκες και γραμμές και κάντε δεξί κλικ, για να εμφανιστεί το αντίστοιχο μενού.
6. Η γραμμή εργαλείων που περιγράψαμε είναι η ***τυπική γραμμή εργαλείων*** (***standard toolbar***). Εκτός όμως από αυτή, υπάρχουν και άλλες γραμμές εργαλείων. Από ποια επιλογή του μενού μπορείτε να τις εμφανίσετε; Ερευνήστε, μήπως μπορείτε να συνθέσετε γραμμή εργαλείων με τις δικές σας προτιμήσεις από τις επιλογές της γραμμής μενού.
7. Τα παράθυρα του περιβάλλοντος εργασίας λειτουργούν όπως ακριβώς και κάθε παράθυρο του περιβάλλοντος των Windows. Μπορούμε να τα μετακινήσουμε σύροντάς τα από τον τίτλο τους, να αλλάξουμε τις διαστάσεις τους και να τα κλείσουμε. Για να τα επανεμφανίσουμε, χρησιμοποιούμε την επιλογή **View** του μενού. Εξοικειωθείτε με αυτές τις λειτουργίες αναδιατάσσοντας τα παράθυρα και διαμορφώνοντας διαφορετικά το χώρο εργασίας. Στο τέλος να επαναφέρετε τα παράθυρα στη θέση τους, όπως ακριβώς δείχνει η εικόνα 3-3.
   Προσοχή: Κάποια από τα παράθυρα (π.χ. το παράθυρο έργου και το παράθυρο ιδιοτήτων) έχουν την ιδιότητα της ***συσσώρευσης*** (***Stacked***) ενεργοποιημένη. Αυτό σημαίνει, ότι μπορούν να συμπεριληφθούν μέσα σε ένα κοινό πλαίσιο με άλλα παράθυρα. Κινώντας ένα παράθυρο και φέρνοντάς το σε επαφή με ένα άλλο, με το οποίο μπορεί να γίνει κοινή συσσώρευση, το ένα παράθυρο απορροφάται στο εσωτερικό του άλλου με αποτέλεσμα να δημιουργείται ένας σωρός παραθύρων.

Εικόνα 3-16. *Διαδοχικά στιγμιότυπα κατά τη μετακίνηση του παραθύρου σκαριφήματος προς το παράθυρο ιδιοτήτων. Στο δεύτερο στιγμιότυπο το παράθυρο σκαριφήματος δεν έχει μπει στα "σύνορα" της περιοχής του παραθύρου έργου, το πλαίσιο μετακίνησής του έχει χοντρό περίγραμμα. Στο τρίτο στιγμιότυπο έχει περάσει τα σύνορα και το περίγραμμά του γίνεται πιο λεπτό. Αν αυτή τη στιγμή ο προγραμματιστής αφήσει το πλήκτρο του ποντικιού, θα γίνει η συσσώρευση (στιγμιότυπο 4).*

8. Ανοίξτε ένα αρχείο `.VBP` από έναν επεξεργαστή κειμένου. Αναγνωρίζετε κάποια στοιχεία μέσα του; Επαναλάβετε το ίδιο με αρχεία τύπου `.FRM` και `.BAS`.
   Υπόδειξη: Αναζητήστε στο δίσκο του υπολογιστή τέτοιας μορφής αρχεία. Χρησιμοποιήστε τις δυνατότητες του λειτουργικού συστήματος.
9. Ανοίξτε το παράθυρο άμεσης εκτέλεσης εντολών. Πληκτρολογήστε στο εσωτερικό του την εντολή: `Print 2+2` και πατήστε το πλήκτρο `Enter`. Τι παρατηρείτε; Συντάξτε μόνοι σας μια πιο σύνθετη αριθμητική παράσταση μετά την εντολή `Print` και ζητήστε τον υπολογισμό της. Σχολιάστε ποια είναι τα πλεονεκτήματα του παραθύρου άμεσης εκτέλεσης εντολών σε σχέση με την αριθμομηχανή.
   Υπόδειξη: Για συντομία μπορεί να χρησιμοποιηθεί και το σύμβολο `?`, αντί της κωδικής λέξης `Print`, π.χ. `? 2+2`.
10. Ανοίξτε το παράθυρο τοπικών μεταβλητών και το παράθυρο παρακολούθησης.
11. Εστιάστε στο παράθυρο έργου και πατήστε το πλήκτρο `F1` για να δείτε πρόσθετες πληροφορίες. Επαναλάβετε τη διαδικασία με ένα εργαλείο από την εργαλειοθήκη του περιβάλλοντος εργασίας.

Σημειώσεις:

Μάθημα 4
Η δημιουργία της διεπαφής

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να σχεδιάζουν φόρμες για τη διεπαφή χρήστη.
- Να αναγνωρίζουν και να χρησιμοποιούν τα βασικά αντικείμενα ελέγχου.
- Να εκμεταλλεύονται τις ιδιότητες των φορμών και των αντικειμένων ελέγχου.
- Να ζητούν την εκτέλεση των προγραμμάτων τους από το περιβάλλον εργασίας.

Έχουμε αναφέρει ότι ένα από τα πιο σημαντικά χαρακτηριστικά της VB είναι ο οπτικός τρόπος προγραμματισμού. Σύμφωνα με αυτόν τον τρόπο, ο προγραμματιστής δε γράφει κείμενο κωδικοποιημένο σε γλώσσα προγραμματισμού, αλλά σχεδιάζει επάνω στην οθόνη και τα σχέδιά του μετατρέπονται αυτόματα σε πρόγραμμα. Ο οπτικός προγραμματισμός διευκολύνει τους προγραμματιστές και τους κάνει πιο παραγωγικούς μιας και ο άνθρωπος από τη φύση του διαχειρίζεται πιο εύκολα την εικόνα από το κείμενο. Δυστυχώς όμως, ο οπτικός τρόπος προγραμματισμού δεν είναι δυνατόν να εφαρμοστεί σε όλα τα κομμάτια ενός έργου. Γι' αυτό και αρκετά κομμάτια ενός προγράμματος πρέπει να υλοποιηθούν προγραμματίζοντας με τον παραδοσιακό τρόπο σε γλώσσα προγραμματισμού.

Αυτός ο ανάμεικτος τρόπος προγραμματισμού δεν προκαλεί προβλήματα στους προγραμματιστές γιατί το τι θα υλοποιηθεί σχεδίαζοντας και το τι θα υλοποιηθεί κωδικοποιώντας είναι σαφώς καθορισμένο. Ο προγραμματιστής έχει:

1. Να δημιουργήσει σχεδιάζοντας τη ***διεπαφή χρήστη*** (***user interface***), δηλαδή τον τρόπο επικοινωνίας της εφαρμογής με το χρήστη για την είσοδο δεδομένων και την εμφάνιση αποτελεσμάτων.
2. Να γράψει τον κατάλληλο κώδικα για τα ***συμβάντα*** (***events***) για να δημιουργήσει τις λειτουργίες που θέλει να επιτελεί το πρόγραμμά του.

Τον όρο συμβάν θα τον αναλύσουμε σε προσεχές μάθημα. Εδώ τον αναφέραμε για να έχει πληρότητα το κείμενο.

Η διεπαφή παίζει πρωταρχικό ρόλο στην αποδοχή ή την απόρριψη ενός προγράμματος. Δεν αρκεί να εκτελεί τις βασικές της ενέργειες σωστά και να είναι οπτικά ελκυστική, ικανοποιώντας κάποια κριτήρια αισθητικής, αλλά πρέπει να καθοδηγεί το χρήστη εκμεταλλευόμενη τους κανόνες της ανθρώπινης συμπεριφοράς. Το βασικό στοιχείο της διεπαφής στο περιβάλλον των Windows είναι το ***παράθυρο*** (***window***), με αποτέλεσμα να έχει σχεδόν ταυτιστεί με αυτό. Μια εφαρμογή μπορεί να παρουσιάζει κατά την επικοινωνία της με το χρήστη ένα η περισσότερα παράθυρα, ανάλογα με το πόσο σύνθετη είναι η είσοδος των στοιχείων και η παρουσίαση των αποτελεσμάτων.

Η δημιουργία της διεπαφής αποτελεί παιχνίδι στο περιβάλλον εργασίας της VB. Χρησιμοποιώντας τις αυξημένες δυνατότητες του περιβάλλοντος εργασίας ο προγραμματιστής μπορεί να σχεδιάσει ποιοτικές διεπαφές σε σύντομο χρονικό διάστημα, χωρίς να χρειαστεί να γράψει ούτε μια γραμμή κώδικα.

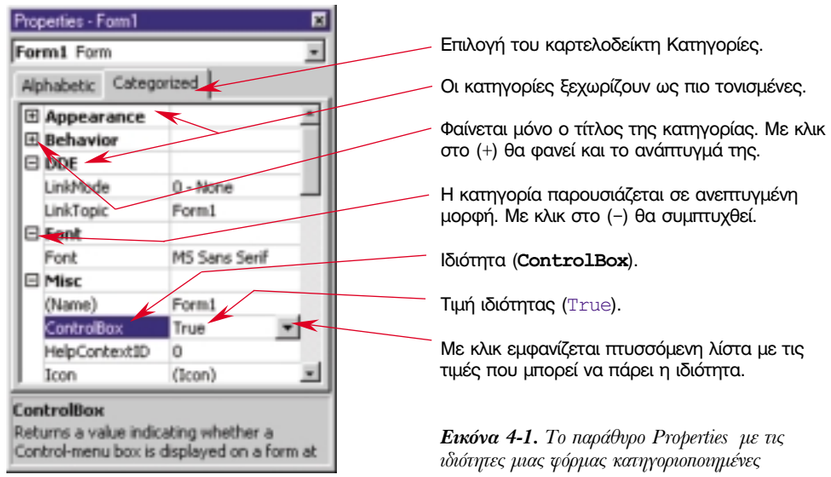
Φόρμες

Όπως στην κατασκευή ενός σπιτιού αρχίζουμε από τα θεμέλια, έτσι και στη δημιουργία της διεπαφής αρχίζουμε από το θεμελιακό στοιχείο, που είναι η φόρμα του παραθύρου. Έχουμε δει ότι, όταν ζητάμε τη δημιουργία ενός νέου έργου, το περιβάλλον της VB δημιουργεί με δική του πρωτοβουλία μια φόρμα, την `Form1`. Αυτή τη φόρμα μπορούμε να τη χρησιμοποιήσουμε για το σχεδιασμό του κύριου παράθυρου της εφαρμογής.

Οι ιδιότητες της φόρμας

Κάθε φόρμα είναι ένα αντικείμενο και σαν τέτοιο χαρακτηρίζεται από τις ιδιότητές του. Κάποιες από τις ιδιότητες της φόρμας αφορούν τη μορφή της, ενώ κάποιες άλλες τη συμπεριφορά της. Για να δούμε τις ιδιότητες μιας φόρμας στο παράθυρο ιδιοτήτων, κάνουμε κλικ επάνω στη φόρμα. Στη συνέχεια για να αλλάξουμε μια ιδιότητα, την επιλέγουμε στο παράθυρο ιδιοτήτων.

Τις τιμές κάποιων ιδιοτήτων τις πληκτρολογούμε, ενώ τις τιμές κάποιων άλλων τις επιλέγουμε μέσα από πτυσσόμενες λίστες.



Επιλογή του καρτελοδείκτη Κατηγορίες.

Οι κατηγορίες ξεχωρίζουν ως πιο τονισμένες.

Φαίνεται μόνο ο τίτλος της κατηγορίας. Με κλικ στο (+) θα φανεί και το ανάπτυγμά της.

Η κατηγορία παρουσιάζεται σε ανεπτυγμένη μορφή. Με κλικ στο (−) θα συμπτυχθεί.

Ιδιότητα (**ControlBox**).

Τιμή ιδιότητας (`True`).

Με κλικ εμφανίζεται πτυσσόμενη λίστα με τις τιμές που μπορεί να πάρει η ιδιότητα.

Εικόνα 4-1. *Το παράθυρο Properties με τις ιδιότητες μιας φόρμας κατηγοριοποιημένες*

Το όνομα της φόρμας, ο τίτλος του παραθύρου, το όνομα αρχείου της φόρμας

Το όνομα της φόρμας, ο τίτλος του παραθύρου, το όνομα αρχείου της φόρμας αντιπροσωπεύουν διαφορετικές έννοιες.

Οι φόρμες, αλλά όπως θα δούμε και όλα τα αντικείμενα, έχουν ένα ***όνομα***. Το όνομα αυτό χρησιμοποιείται για την ταυτοποίησή τους και τη μεταξύ τους διάκριση, γι' αυτό και δεν επιτρέπεται δύο φόρμες να έχουν το ίδιο όνομα. Η ιδιότητα που καθορίζει το όνομα μιας φόρμας είναι η **`Name`**. Το όνομα της φόρμας είναι αυτό που εμφανίζεται στο παράθυρο έργου. Όταν δημιουργείται μια φόρμα, το περιβάλλον εργασίας της προσδίδει αυτόματα ένα όνομα `Form1`, `Form2`, ... Συνήθως, αλλάζουμε αυτό το όνομα για να μας θυμίζει κάτι πιο ουσιαστικό π.χ. τη λειτουργία ή την ιεραρχία της φόρμας μέσα σε μια κατάταξη. Η ονοματοδοσία των φορμών (όπως και των αντικειμένου ελέγχου που θα συναντήσουμε στη συνέχεια) καθορίζεται από τους εξής κανόνες:

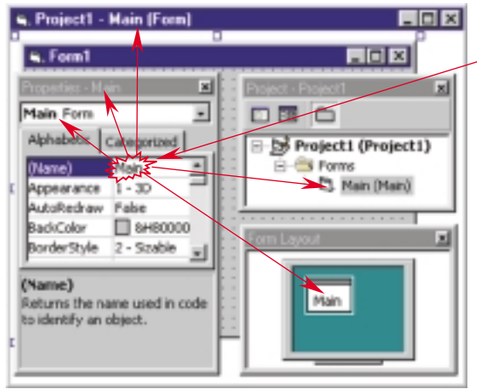
- Τα ονόματα πρέπει να αρχίζουν με γράμμα.
- Τα ονόματα αποτελούνται από γράμματα, αριθμούς και χαρακτήρες υπογράμμισης (_). Δεν επιτρέπονται άλλα σύμβολα ή κενά μέσα σε ένα όνομα.
- Τα ονόματα, καλό είναι, να αποτελούνται από λιγότερους των 40 χαρακτήρων. Περισσότεροι χαρακτήρες τα κάνουν δυσδιάκριτα.
- Δε γίνεται διάκριση μεταξύ των πεζών και των κεφαλαίων γραμμάτων οπότε μπορούν να χρησιμοποιούνται εναλλακτικά. Για παράδειγμα, τα ονόματα `lightemitting` και `LightEmitting` είναι ταυτόσημα, το δεύτερο όμως είναι πιο ευανάγνωστο και
- Χωρίς να είναι υποχρεωτικό, καλό είναι να χρησιμοποιούμε τρία γράμματα στην αρχή ή στο τέλος του ονόματος που να χαρακτηρίζουν το αντικείμενο. Για παράδειγμα, `frm` για τις φόρμες, `cmd` για τα πλήκτρα ... .
- Το όνομα μπορεί να περιέχει ελληνικούς χαρακτήρες.

Κάθε παράθυρο μπορεί να έχει μια γραμμή τίτλου. Ο τίτλος του παραθύρου, που φαίνεται στη γραμμή τίτλου, καθορίζεται από την ιδιότητα **`Caption`** της φόρμας του. Αν το παράθυρο αποτελεί το κύριο παράθυρο της εφαρμογής συνηθίζεται, να δίνουμε ως τίτλο το όνομα της εφαρμογής. Ο τίτλος του παραθύρου δεν ακολουθεί τους περιορισμούς του ονόματος της φόρμας.

Όπως έχουμε αναφέρει στο προηγούμενο μάθημα, κάθε φόρμα αποθηκεύεται και σε διαφορετικό αρχείο. Το όνομα του αρχείου μπορεί να είναι το ίδιο με το όνομα της φόρμας, αλλά αυτό δεν αποτελεί κανόνα. Το όνομα του αρχείου κάθε φόρμας φαίνεται μέσα σε παρενθέσεις δίπλα ακριβώς από το όνομά της στο παράθυρο έργου. Οι περιορισμοί που υπάρχουν για τα ονόματα των αρχείων των φορμών (όπως και κάθε αρχείου) είναι αυτοί που υπαγορεύονται από το λειτουργικό σύστημα.

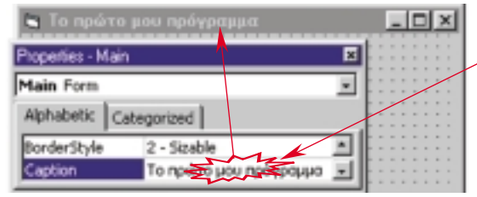
Άσκηση 4-1.

Να δημιουργηθεί μια φόρμα με όνομα `Main`, τίτλο παραθύρου `"Ôï ðñþôï ìïõ ðñüãñáììá"` και στη συνέχεια να αποθηκευθεί σε αρχείο με όνομα `MainForm.frm`.



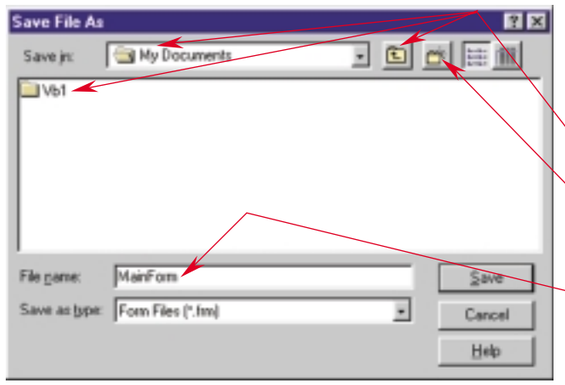
Εικόνα 4-2. *Αλλαγή ονόματος φόρμας*

Καλούμε το περιβάλλον της VB, όπως μάθαμε στο μάθημα 3.
Στο παράθυρο ιδιοτήτων αλλάζουμε την ιδιότητα (**Name**) από `Form1` σε `Main`. Παρατηρούμε, ότι η αυτή αλλαγή γίνεται αυτόματα και στο παράθυρο έργου, όπου η γραμμή `Form1 (Form1)` γίνεται `Main (Main)` και στα άλλα παράθυρα, όπου αναφέρεται.



Εικόνα 4-3. *Αλλαγή τίτλου του παραθύρου*

Στο παράθυρο ιδιοτήτων αλλάζουμε την ιδιότητα **`Caption`** από `Form1` σε `"Ôï ðñþôï ìïõ ðñüãñáììá"`. Παρατηρούμε ότι η αλλαγή αυτή γίνεται αυτόματα και στον τίτλο της φόρμας.

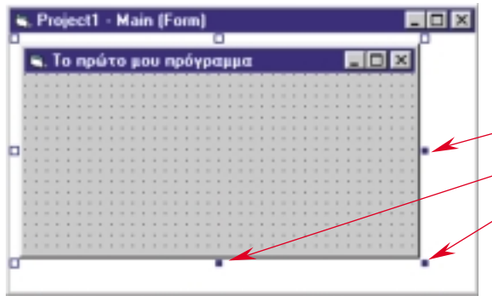


Εικόνα 4-4. *Αλλαγή ονόματος αρχείου*

1. Επιλέγουμε τη φόρμα **Main** στο παράθυρο έργου.
2. Από τη γραμμή μενού επιλέγουμε **File** και την υποεπιλογή **Save Main As** ... (Η VB προσαρμόζει και την υποεπιλογή ανάλογα με τη φόρμα).
3. Διαλέγουμε το φάκελο στον οποίο θα τοποθετήσουμε την εργασίεα μας.
4. Δημιουργούμε ένα νέο υποφάκελο. Καλό είναι κάθε έργο να το αποθηκεύουμε και σε έναν υποφάκελο.
5. Δίνουμε όνομα στο αρχείο.

Οι διαστάσεις της φόρμας

Το μέγεθος της φόρμας μπορεί να αλλάξει σύροντας τα σκουρόχρωμα τετραγωνάκια (***λαβές***) που υπάρχουν στο περίγραμμα φόρμας.



Η θέση της πάνω αριστερής κορυφής και οι διαστάσεις της φόρμας αναφέρονται στη γραμμή εργαλείων.

Αλλαγή του μεγέθους οριζόντια.

Αλλαγή του μεγέθους κατακόρυφα.

Αλλαγή του μεγέθους οριζόντια και κατακόρυφα.

Εικόνα 4-5. *Το μέγεθος της φόρμας μπορεί να αλλάξει από τις λαβές.*

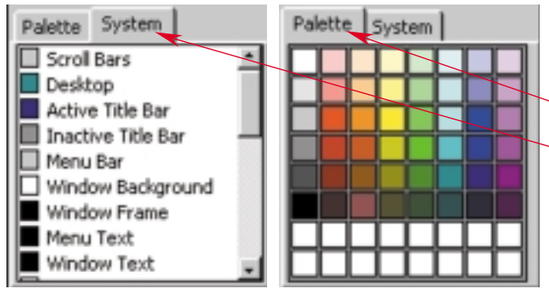
Η θέση της φόρμας στην οθόνη μπορεί να αλλάξει σέρνοντας το σκαρίφημά της μέσα στο παράθυρο εμφάνισης σκαριφημάτων.

Ακόμα η θέση της φόρμας μπορεί να αλλάξει από τις ιδιότητες **Left** (αριστερά), **Top** (πάνω), **Width** (πλάτος), **Height** (ύψος) με αλλαγή των τιμών τους στο παράθυρο ιδιοτήτων. Οι τιμές πρέπει να δοθούν σε twips. Ας σημειωθεί, ότι 20 twips = 1στιγμή και αφού 1 ίντζα = 72 στιγμές υπάρχουν 1440 twips ανά ίντσα. Η μονάδα αυτή έχει επιλεγεί, ώστε να είναι δυνατόν να καλυφθούν οι ανάγκες και των οθονών πολύ υψηλής ανάλυσης που θα κατασκευαστούν στο μέλλον (όσο μικρότερη είναι η μονάδα μέτρησης, τόσο καλύτερη διακριτότητα πετυχαίνεται).

1ίντσα=2.54εκατοστά

Οπτικές ιδιότητες

Το χρώμα του ***υπόβαθρου*** (***background***) της φόρμας καθορίζεται από την ιδιότητα **BackColor**. Το χρώμα μπορεί να επιλεγεί από μια παλέτα χρωμάτων ή από μια λίστα χρωμάτων που χρησιμοποιούν τα Windows ανάλογα με τον καρτελοδείκτη, που έχει επιλεγεί. Ακόμα, το χρώμα μπορεί να δοθεί σαν ένας δεκαεξαδικός αριθμός.



Εικόνα 4-6. *Η επιλογή χρώματος μπορεί να γίνει από:*

Η ιδιότητα **BorderStyle** καθορίζει τον τύπο του πλαισίου και των στοιχείων που θα εμφανίζονται στη γραμμή τίτλου του παραθύρου. Οι τιμές που μπορεί να πάρει είναι:

0 - None Παράθυρο χωρίς πλαίσιο και γραμμή τίτλου.

1 - Fixed Single Παράθυρο με πλαίσιο και γραμμή τίτλου. Ο χρήστης δεν μπορεί να αλλάξει το μέγεθος του παραθύρου.

2 - Sizeable Παράθυρο με πλαίσιο και γραμμή τίτλου. Ο χρήστης μπορεί να αλλάξει το μέγεθος του παραθύρου. Αποτελεί την προτεινόμενη τιμή του περιβάλλοντος εργασίας για κάθε νέο παράθυρο.

3 -Fixed Dialog Παράθυρο με πλαίσιο και γραμμή τίτλου, χωρίς πλήκτρα επαναφοράς /μεγιστοποίησης και ελαχιστοποίησης. Ο χρήστης δεν μπορεί να αλλάξει το μέγεθος του παραθύρου.

4 - Fixed ToolWindow Παράθυρο παλέτας χωρίς πλήκτρα ελαχιστοποίησης και επαναφοράς /μεγιστοποίησης. Ο χρήστης δεν μπορεί να αλλάξει το μέγεθος του παραθύρου. Η γραμμή τίτλου γίνεται πιο στενή.

5 - Sizeable ToolWindow Παράθυρο παλέτας χωρίς πλήκτρα ελαχιστοποίησης και επαναφοράς /μεγιστοποίησης. Ο χρήστης μπορεί να αλλάξει το μέγεθος του παραθύρου. Η γραμμή τίτλου γίνεται πιο στενή.

Ας σημειωθεί ότι η τιμή που δίνεται στην ιδιότητα **BorderStyle** δεν είναι δυνατόν να αλλάξει κατά τη στιγμή της εκτέλεσης του προγράμματος.

Τα πλήκτρα της γραμμής τίτλου

Δίνοντας την τιμή `True` (αληθές) στην ιδιότητα **MinButton** (αν το επιτρέπει η τιμή της ιδιότητας **BorderStyle** και της ιδιότητας **ControlBox**), στη γραμμή τίτλου εμφανίζεται το πλήκτρο ελαχιστοποίησης. Διαφορετικά, αν δοθεί η τιμή `False` (ψευδές) το πλήκτρο ελαχιστοποίσης δεν εμφανίζεται.

Παραμοίως, δίνοντας την τιμή `True` στην ιδιότητα **MaxButton** (αν το επιτρέπει η τιμή της ιδιότητας **BorderStyle** και της ιδιότητας **ControlBox**), στη γραμμή τίτλου εμφανίζεται το πλήκτρο επαναφοράς/μεγιστοποίησης.

Η ιδιότητα **ControlBox** καθορίζει, αν στη γραμμή τίτλου θα εμφανίζεται το πλήκτρο ελέγχου και τα πλήκτρα ελαχιστοποίησης, επαναφοράς/μεγιστοποίησης και κλεισίματος.

Ιδιότητες συμπεριφοράς

Μια εφαρμογή μπορεί να έχει περισσότερα από ένα παράθυρα. Όταν ένα παράθυρο εκπληρώσει το σκοπό του και δε χρειάζεται άλλο, για να αποσυμφορήσουμε την οθόνη μπορούμε να το κάνουμε αόρατο, δίνοντας την τιμή `False` στην ιδιότητα **Visible** (ορατό).

Τα αποτελέσματα από την αλλαγή των τιμών των ιδιοτήτων αυτής της κατηγορίας φαίνονται μόνο κατά τη διάρκεια εκτέλεσης της εφαρμογής.

Επίσης, πολλές φορές χρειάζεται να είναι εμφανές το περιεχόμενο ενός παραθύρου αλλά δε θέλουμε να έχει ο χρήστης τη δυνατότητα να μεταβεί σε αυτό για να πραγματοποιήσει λειτουργίες που μπορεί να εκτελέσει. Αυτό γίνεται δίνοντας την τιμή `False` στην ιδιότητα **Enabled** (ενεργοποιημένο).

Κατά κανόνα τα παράθυρα μπορούμε να τα μετακινούμε στην οθόνη. Αν όμως θέλουμε ένα παράθυρο να μένει ακίνητο και να μην έχει δυνατότητα ο χρήστης να το σύρει σε μιαν άλλη θέση στην οθόνη, δίνουμε την τιμή `False` στην ιδιότητα **Moveable** (μετακινήσιμο).

Θα έχετε παρατηρήσει,ότι σε πολλά προγράμματα, ο δείκτης του ποντικιού αλλάζει μορφή τη στιγμή που διέρχεται πάνω από μια φόρμα. Αυτό μπορούμε να το κάνουμε και σε κάποιες φόρμες του προγράμματός μας για να ειδοποιήσουμε το χρήστη να δείξει ιδιαίτερη προσοχή. Η αλλαγή του δείκτη του ποντικιού γίνεται δίνοντας κατάλληλη τιμή στην ιδιότητα **MousePointer**. Η τιμή επιλέγεται μέσα από μια προτυποποιημένη λίστα τιμών.

Θα έχετε παρατηρήσει ότι, όταν ένα πρόγραμμα εκτελείται, ο τίτλος του παραθύρου του εμφανίζεται στη γραμμή εργασιών. Για να κάνουμε το ίδιο με ένα παράθυρο του προγράμματός μας δίνουμε την τιμή `True` στην ιδιότητα **ShowInTaskbar** της φόρμας του. Έτσι, θα είναι εύκολη η ανάκλησή του μέσα από τη στοίβα των παραθύρων που δημιουργεί κατά την εργασία του ο χρήστης.

Η εκτέλεση του προγράμματος

Για να ζητήσουμε την εκτέλεση του προγράμματός μας επιλέγουμε **Run** από τη γραμμή μενού και στη συνέχεια **Start**. Το περιβάλλον εργασίας αλλάζει, από τη φόρμα εξαφανίζεται το πλέγμα. Το παράθυρο συμπεριφέρεται σύμφωνα με τις τιμές που έχουμε δώσει στις ιδιότητες. Μπορεί να αλλάξει μέγεθος και θέση, αν το έχουμε προδιαγράψει, να μικρύνει σε εικονίδιο, να επανέλθει στο μέγεθός του και να καταλάβει όλη την οθόνη, αν του έχουμε βάλει τα κατάλληλα πλήκτρα κ.ά.

Το αντίστοιχο πλήκτρο της γραμμής εργαλείων για την εκτέλεση του προγράμματος.

Για να διακόψουμε την εκτέλεση του προγράμματος μας επιλέγουμε **Run** από το μενού και στη συνέχεια **End**. Το περιβάλλον εργασίας επανέρχεται σε κατάσταση σχεδίασης.

Το αντίστοιχο πλήκτρο της γραμμής εργαλείων για τον τερματισμό του προγράμματος.

Αντικείμενα ελέγχου

Τοποθετώντας τα αντικείμενα ελέγχου πάνω σε μια φόρμα, αισθανόμαστε περισσότερο σαν σχεδιαστές που χειρίζονται ένα σχεδιαστικό πρόγραμμα παρά σαν προγραμματιστές. Με απλές κινήσεις επιλέγουμε το κατάλληλο εργαλείο από την εργαλειοθήκη και πάνω στη φόρμα, σχεδιάζουμε το αντίστοιχο αντικείμενο, διαμορφώνουμε τις διαστάσεις του, τα ευθυγραμμίζουμε με άλλα αντικείμενα και αλλάζουμε τις άλλες ιδιότητές τους.

Εικόνα 4-7. *Η εργαλειοθήκη*

Η εργαλειοθήκη περιέχει τα εργαλεία, με τα οποία μπορούμε να τοποθετήσουμε αντικείμενα ελέγχου επάνω στις φόρμες, για να συμπληρώσουμε τη διεπαφή χρήστη. Ο ***δείκτης*** (***Pointer***) είναι το μοναδικό εργαλείο που δεν αντιστοιχεί σε αντικείμενο ελέγχου. Τα αντικείμενα ελέγχου της εργαλειοθήκης είναι:

Ετικέτα (Label): Παρουσιάζει κείμενο που δεν μπορεί να αλλάξει ο χρήστης. Κατά κανόνα και ο προγραμματιστής δεν αλλάζει την τιμή της.

Πλαίσιο κειμένου (Text box): Δέχεται δεδομένα από το χρήστη ή προβάλλει αποτελέσματα στο χρήστη. Είναι το αντικείμενο ελέγχου που χρησιμοποιείται περισσότερο.

Πλαίσιο (Frame): Ομαδοποιεί λειτουργικά ή διακοσμητικά τα αντικείμενα ελέγχου (π.χ. πλήκτρα επιλογής), που είναι τοποθετημένα στο εσωτερικό του.

Πλήκτρο διαταγής (Command button): Απεικονίζει ένα πλήκτρο, το οποίο, όταν το πατά ο χρήστης εκτελούνται οι λειτουργίες που έχει επισυνάψει ο προγραμματιστής.

Πλήκτρο σημείωσης (Check box): Απεικονίζει ένα κουτάκι σαν αυτά που μαρκάρουμε (Αληθές) ή που αφήνουμε αμαρκάριστα (Ψευδές).

Πλήκτρο επιλογής (Option button): Απεικονίζει ένα κουτάκι σαν αυτά που μαρκάρουμε (Αληθές) ή που αφήνουμε αμαρκάριστα (Ψευδές). Όταν αποτελεί ομάδα με άλλα πλήκτρα επιλογής μέσα σε ένα πλαίσιο, μόνο ένα πλήκτρο επιλογής επιτρέπεται να είναι επιλεγμένο.

Λίστα (List box): Παρουσιάζει μια λίστα επιλογών. Ο χρήστης είναι υποχρεωμένος να επιλέξει μια ή περισσότερες τιμές μέσα από αυτή τη λίστα χωρίς να έχει δυνατότητα να εισάγει μια δική του τιμή.

Συνδυασμένη λίστα (Combo box): Συνδυάζει τα χαρακτηριστικά του πλαίσιου κειμένου και της λίστας. Ο χρήστης μπορεί να εισάγει δεδομένα ή να επιλέξει μια τιμή μέσα από την πτυσσόμενη λίστα.

Οριζόντια ράβδος κύλισης (Horizontal scroll bar): Χρησιμοποιείται για γρήγορη μεταφορά μέσα σε ένα εύρος τιμών.

Κατακόρυφος ράβδος κύλισης (Vertical scroll bar): Χρησιμοποιείται για γρήγορη μεταφορά μέσα σε ένα εύρος τιμών.

Χρονόμετρο (Timer): Πρόκειται για μη ορατό αντικείμενο ελέγχου. Χρησιμοποιείται σαν χρονοδιακόπτης για την εκτέλεση λειτουργιών σε τακτά χρονικά διαστήματα.

Πτυσσόμενη λίστα δίσκων (Drive list box): Εμφανίζει σε πτυσσόμενη λίστα τα ονόματα όλων των αποθηκευτικών μέσων του υπολογιστή. Ο χρήστης μπορεί να επιλέξει έναν από αυτούς.

Πτυσσόμενη λίστα καταλόγων (Directory list box): Εμφανίζει σε δενδροϊδή μορφή έναν κλάδο με τα ονόματα καταλόγων. Ο χρήστης μπορεί να επιλέξει έναν από αυτούς μέσα από την πτυσσόμενη λίστα.

Λίστα αρχείων (File list box): Εμφανίζει τα ονόματα των αρχείων ενός καταλόγου. Ο χρήστης μπορεί να επιλέξει ένα από αυτά.

Γεωμετρικό σχήμα (Shape): Με το εργαλείο αυτό μπορούμε να σχεδιάσουμε ένα τετράγωνο, ένα ορθογώνιο, έναν κύκλο, μια έλλειψη.

Γραμμή (Line): Ευθύγραμμο τμήμα.

Πλαίσιο Εικόνας (Picture box): Προβάλλει εικόνες ή γραφικά.

Εικόνα (Image): Προβάλλει εικόνες. Έχει μειωμένες δυνατότητες σε σχέση με το πλαίσιο εικόνας.

Χειρισμός δεδομένων (Data): Προσφέρει προσπέλαση στα δεδομένα μιας βάσης δεδομένων (π.χ. της Access).

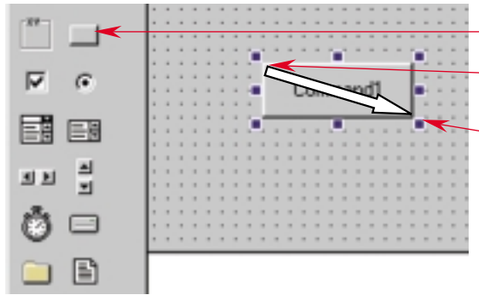
Προς το παρόν δε θα χρησιμοποιούμε όλα τα αντικείμενα ελέγχου παρά μόνο τα πιο βασικά από αυτά. Τα κάπως πιο εξειδικευμένα θα τα αναπτύξουμε σταδιακά σε προσεχή μαθήματα. Επίσης, τα αντικείμενα ελέγχου που υπάρχουν σε μια εργαλειοθήκη ενός περιβάλλοντος εργασίας, δεν είναι τα μοναδικά. Η εργαλειοθήκη μπορεί να εμπλουτιστεί με πολύ απλό τρόπο και με αντικείμενα ελέγχου τρίτων κατασκευαστών και με άλλα που ονομάζονται ***ActiveX***.

Η εισαγωγή των ActiveX γίνεται από την επιλογή του **Project** του μενού, υποεπιλογή **Components**.

Διαχείριση αντικειμένων ελέγχου

Εισαγωγή αντικειμένου ελέγχου

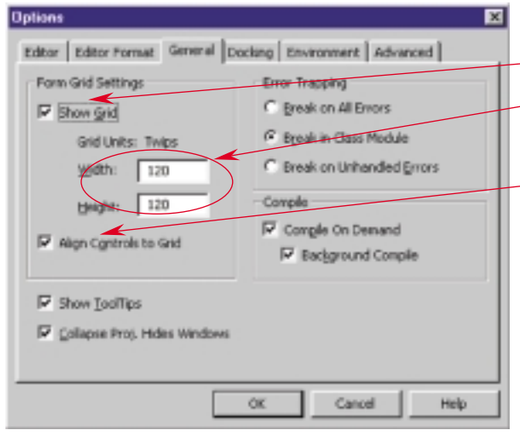
Η τοποθέτηση των αντικειμένων επάνω σε μια φόρμα γίνεται πάρα πολύ απλά σαν να σχεδιάζουμε σε ένα από τα προγράμματα σχεδίασης. Επιλέγουμε με το ποντίκι το κατάλληλο εργαλείο από την εργαλειοθήκη, δείχνουμε στη φόρμα το σημείο που θέλουμε να τοποθετήσουμε το αντικείμενο ελέγχου και ορίζουμε το μέγεθος του αντικειμένου σέρνοντας το ποντίκι. Όταν αφήσουμε το πλήκτρο του ποντικιού, θα εμφανιστεί το αντικείμενο.



Εικόνα 4-8. *Εισαγωγή πλήκτρου διαταγής*

Αν σχεδιάσουμε και δεύτερο αντικείμενο ελέγχου, οι λαβές στο περίγραμμα του πρώτου αντικειμένου ελέγχου εξαφανίζονται. Μπορούμε να επιλέξουμε πάλι το πρώτο αντικείμενο κάνοντας κλικ επάνω του, οπότε και επανεμφανίζονται οι λαβές του. Πάνω σε μια φόρμα μπορούμε να έχουμε οποιοδήποτε συνδυασμό αντικειμένων ελέγχου.

Στην περίπτωση που υπάρχει πλέγμα πάνω στη φόρμα, τα άκρα των αντικειμένων παγιδεύονται στα πλησιέστερα σημεία του πλέγματος. Η εμφάνιση του πλέγματος σε κατάσταση σχεδίασης του περιβάλλοντος εργασίας, η πυκνότητά του και η δυνατότητα ***παγίδευσης*** (***snap***) των άκρων των αντικειμένων ελέγχου μπορεί να ρυθμιστεί επιλέγοντας **Tools** από τη γραμμή μενού, **Options** στη συνέχεια και τον καρτελοδείκτη **General** του διαλογικού παραθύρου που θα εμφανιστεί.



Εικόνα 4-9. *Αλλαγές στα χαρακτηριστικά του πλέγματος σχεδίασης*

Αλλαγή θέσης αντικειμένου ελέγχου

Για να αλλάξουμε τη θέση ενός αντικειμένου ελέγχου, το επιλέγουμε και το σέρνουμε επάνω στη φόρμα. Το αντικείμενο θα τοποθετηθεί στο σημείο που θα αφήσουμε το πλήκτρο του ποντικιού. Αλλαγή της θέσης ενός αντικειμένου ελέγχου γίνεται και από το πληκτρολόγιο. Αφού επιλέξουμε το αντικείμενο, κρατάμε πατημένο το πλήκτρο **`Control`** και το αντίστοιχο από τα βελάκια πλήκτρο.

Πολλαπλή επιλογή

Μπορούμε να μετακινήσουμε ταυτόχρονα περισσότερα από ένα αντικείμενα, αν τα επιλέξουμε όλα μαζί. Για πολλαπλή επιλογή κάνουμε κλικ στο πρώτο αντικείμενο και ενώ κρατάμε πατημένο το πλήκτρο `Shift`, κάνουμε διαδοχικά κλικ πάνω σε αυτά που θέλουμε να επιλέξουμε επιπλέον. Η πολλαπλή επιλογή γίνεται επίσης εύκολα σέρνοντας το ποντίκι σε μια ορθογώνια περιοχή που περιβάλλει ή τέμνει τα αντικείμενα. Για να αφαιρέσουμε ένα αντικείμενο από την ομάδα κάνουμε κλικ πάλι επάνω του, ενώ κρατάμε πατημένο το πλήκτρο `Shift`.

Για να αλλάξουμε το μέγεθος ενός αντικειμένου ελέγχου, το επιλέγουμε και σέρνουμε προς τη νέα του θέση ένα από τα σκουρόχρωμα τετραγωνάκια (λαβές), που υπάρχουν στις πλευρές του. Αλλαγή της διάστασης του αντικειμένου ελέγχου γίνεται και από το πληκτρολόγιο. Αφού επιλέξουμε το αντικείμενο, κρατάμε πατημένο το πλήκτρο `Shift` και το αντίστοιχο από τα βελάκια πλήκτρο.

Αλλαγή διάστασης αντικειμένου ελέγχου

Για να αφαιρέσουμε ένα αντικείμενο ελέγχου από τη φόρμα, το επιλέγουμε και πατάμε το πλήκτρο `Del`.

Διαγραφή αντικειμένου ελέγχου

Το πλέγμα βοηθά στη σωστή ευθυγράμμιση των αντικειμένων ελέγχου. Υπάρχει όμως και μια σειρά επιλογών του μενού **Format** με τις οποίες είναι δυνατή η γρήγορη ευθυγράμμιση, ρύθμιση του μεγέθους και κατανομή μιας ομάδας αντικειμένων. Συγκεκριμένα η υποεπιλογή:

Align: Ευθυγραμμίζει τα αντικείμενα ελέγχου στα αριστερά, στα δεξιά, στα πάνω, στα κάτω άκρα

Make Same Size: Δίνει στα επιλεγμένα αντικείμενα ελέγχου το ίδιο πλάτος ή και ύψος.

Horizontal Spacing: Ρυθμίζει την απόσταση των επιλεγμένων αντικειμένων ελέγχου, ώστε να ισαπέχουν, να πυκνώσουν ή να αραιώσουν κατά την οριζόντιο διεύθυνση.

Vertical Spacing: Ρυθμίζει την απόσταση των επιλεγμένων αντικειμένων ελέγχου, ώστε να ισαπέχουν, να πυκνώσουν ή να αραιώσουν κατά την κατακόρυφη διεύθυνση.

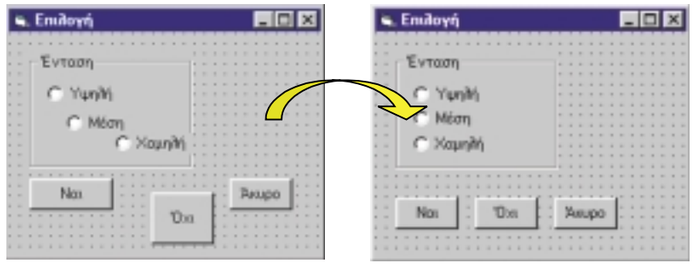
Center in Form: Ευθυγραμμίζει τα επιλεγμένα αντικείμενα στο μέσο της φόρμας.

Order: Τοποθετεί τα αντικείμενα σε σειρά βάθους. Κάθε νέο αντικείμενο ελέγχου που τοποθετείται πάνω στην οθόνη, τοποθετείται πάνω από τα ήδη υπάρχοντα αντικείμενα. Η σειρά αυτή μπορεί να αλλάξει με τις υποεπιλογές **Send to Back** (τελευταίο στη σειρά) **Bring to Front** (πρώτο στη σειρά).

Στην πραγματικότητα δεν υπάρχει μια μόνο σειρά αλλά τρεις. Δείτε άσκηση 7.

Άσκηση 4-2.

Σε μια φόρμα να τοποθετηθούν τρία πλήκτρα επιλογής μέσα σε ένα πλαίσιο και τρία πλήκτρα διαταγής, όπως φαίνεται στην αριστερή εικόνα. Στη συνέχεια να διαμορφωθεί η ιδιότητα **Caption** αυτών των αντικειμένων ελέγχου, ώστε να παρουσιάζουν ελληνικά λεκτικά.



Εικόνα 4-10. *Διαμόρφωση και στοίχιση αντικειμένων.*

Για να τακτοποιήσουμε λίγο τα αντικείμενα ελέγχου μέσα στη φόρμα:

1. Επιλέγουμε τα πλήκτρα επιλογής φροντίζοντας να επιλεγεί τελευταίο το πλήκτρο του οποίου τα άκρα καθορίζουν τη γραμμή στοίχισης (το πλήκτρο με **Caption** `"Õøçëþ"`),
   - στη συνέχεια κάνουμε τις επιλογές **Format ¦ Align ¦ Left** και για
   - ισοκατανομή της απόστασής τους **Format ¦ Vertical Spacing ¦ Equal**
2. Επιλέγουμε τα πλήκτρα διαταγής φροντίζοντας να επιλεγεί τελευταίο το πλήκτρο του οποίου τα άκρα καθορίζουν τη γραμμή στοίχισης (το πλήκτρο με **Caption** `"Íáé"`).
   - Στη συνέχεια κάνουμε τις επιλογές **Format ¦ Align ¦ Tops** και για να
   - αποκτήσουν τις ίδιες διαστάσεις **Format ¦ Make Same Size ¦ Both**, επίσης για
   - ισοκατανομή της απόστασής τους **Format ¦ Horizontal Spacing ¦ Equal** και για
   - πύκνωση **Format ¦ Horizontal Spacing ¦ Decrease**
   - μετακινούμε όλα μαζί τα πλήκτρα λίγο προς τα κάτω.

Στη συνέχεια του βιβλίου θα χρησιμοποιούμε το συμβολισμό επιλογή ¦ υποεπιλογή ¦ υπο-υποεπιλογή για να συμβολίζουμε διαδοχικές επιλογές μέσα σε ένα μενού.

Η εστίαση

Όταν ο χρήστης επιλέξει ένα παράθυρο, το παράθυρο αποκτά την εστίαση. Τονίζεται ο τίτλος του και ότι πληκτρολογείται από αυτή τη στιγμή και μετά αφορά αυτό το παράθυρο. Μόνο ένα παράθυρο μπορεί να έχει κάθε στιγμή την εστίαση. Κάτι ανάλογο γίνεται και με τα αντικείμενα του παραθύρου. Σε ένα παράθυρο, ένα και μόνο ένα αντικείμενο ελέγχου μπορεί να έχει την εστίαση. Το εστιασμένο αντικείμενο ελέγχου διακρίνεται με τονισμό κάποιου οπτικού χαρακτηριστικού του. Για παράδειγμα, αν το αντικείμενο ελέγχου είναι πλήκτρο (διαταγής, επιλογής, σημείωσης) γύρω από τον τίτλο του εμφανίζεται ένα διακεκομμένο τετράγωνο, αν είναι αντικείμενο κειμένου (πλαίσιο κειμένου, συνδυασμένη λίστα) αναβοσβήνει ο δρομέας κειμένου στο εσωτερικό του κ.ά.
Μπορούμε να μεταφέρουμε δυναμικά την εστίαση σε ένα αντικείμενο ελέγχου:

- κάνοντας κλικ επάνω του,
- πατώντας μερικές φορές το πλήκτρο `Tab` που μεταφέρει την εστίαση από αντικείμενο
- ελέγχου σε αντικείμενο ελέγχου
- πατώντας το πλήκτρο `Alt` και το κατάλληλο ***πλήκτρο άμεσης πρόσβασης***.

Πλήκτρο άμεσης πρόσβασης μπορεί να οριστεί για ένα αντικείμενο ελέγχου, αν το αντικείμενο ελέγχου έχει ιδιότητα **Caption** (που τη γνωρίσαμε ήδη στις φόρμες, αλλά την έχουν και μερικά αντικείμενα ελέγχου). Η τιμή της ιδιότητας **Caption** εμφανίζεται σαν τίτλος στην επιφάνεια του αντικειμένου ελέγχου και το πλήκτρο άμεσης πρόσβασης σαν ένα υπογραμμισμένο γράμμα του. Ο χαρακτηρισμός του πλήκτρου άμεσης πρόσβασης γίνεται τοποθετώντας το σύμβολο & μπρος από το γράμμα, που αντιστοιχεί στο πλήκτρο. Για παράδειγμα, γράφουμε `&Open` στην τιμή της ιδιότητας **Caption**, για εμφάνιση του τίτλου σαν `Open` και πλήκτρο άμεσης πρόσβασης το "O". Συνήθως σαν πλήκτρο άμεσης πρόσβασης επιλέγεται το πρώτο γράμμα του τίτλου. Άλλο γράμμα επιλέγεται όταν περισσότερα από ένα αντικείμενα ελέγχου αρχίζουν με το ίδιο γράμμα, που αποτελεί ήδη πλήκτρο άμεσης πρόσβασης.

Μόνο για λατινικά γράμματα και αριθμούς έχει δράση το πλήκτρο άμεσης πρόσβασης.

Τα πλαίσια κειμένου δεν έχουν ιδιότητα **Caption**. Για να έχουμε όμως τη δυνατότητα να μεταφέρουμε την εστίαση απευθείας σε αυτά με πλήκτρο άμεσης πρόσβασης, τοποθετούμε δίπλα τους μια ετικέτα. Μιας και δεν είναι δυνατή η εστίαση στην ετικέτα, η εστίαση μεταφέρεται στο επόμενο αντικείμενο ελέγχου που φροντίζουμε να είναι το πλαίσιο κειμένου που θέλουμε.

Ετικέτα.
Caption: `Address`
Πλήκτρο άμεσης πρόσβασης: A

Πλαίσιο κειμένου που αφορά το πλήκτρο άμεσης πρόσβασης.

Εικόνα 4-11. *Όταν ένα αντικείμενο ελέγχου δεν έχει ιδιότητα* **Caption**, *βάζουμε μπροστά του μια ετικέτα.*

Πατώντας το πλήκτρο `Tab`, η εστίαση μεταφέρεται από αντικείμενο ελέγχου σε αντικείμενο ελέγχου. Η σειρά με την οποία γίνεται αυτή η μεταφορά καθορίζεται από τις τιμές των ιδιοτήτων **TabStop** και **TabIndex**. Η ιδιότητα **TabStop** καθορίζει αν ένα αντικείμενο ελέγχου μπορεί να δεχτεί την εστίαση μετά το πάτημα του πλήκτρου `Tab`. Για τιμή `True` μπορεί να δεχτεί την εστίαση μετά από `Tab` και για τιμή `False` δεν μπορεί. Η ιδιότητα **TabIndex** καθορίζει τη σειρά στην οποία θεωρούνται τα αντικείμενα ελέγχου. Σε κάθε φόρμα η τιμή της **TabIndex** για κάθε αντικείμενο ελέγχου είναι μοναδική. Δυνατές τιμές είναι 0, 1, 2, ... Αν αλλάξουμε την τιμή **TabIndex** για ένα αντικείμενο ελέγχου αυτόματα γίνεται επαναρίθμηση στη σειρά.

Τέλος, ας ληφθεί υπ' όψιν ότι η μετάβαση σε ένα αντικείμενο ελέγχου γίνεται μόνο αν η τιμή της ιδιότητας **Enabled** είναι `True`. Αν η τιμή της ιδιότητας **Enabled** είναι `False`, το αντικείμενο ελέγχου φαίνεται αχνό.

Οι ιδιότητες των αντικειμένων ελέγχου

Όταν δημιουργούμε ένα αντικείμενο ελέγχου πάνω σε μια φόρμα το περιβάλλον εργασίας αποδίδει κάποιες εξορισμού τιμές στις ιδιότητές του, διαμορφώνοντας έτσι τα αρχικά χαρακτηριστικά του. Αυτές τις τιμές (ρυθμίσεις) μπορούμε να τις αλλάξουμε κατά το σχεδιασμό

μέσω του παραθύρου ιδιοτήτων ή/και κατά την εκτέλεση της εφαρμογής μέσω του κώδικα. Είναι σημαντικό να σημειώσουμε ότι οι ιδιότητες που εμφανίζονται στο παράθυρο ιδιοτήτων, και είναι διαθέσιμες κατά το χρόνο σχεδιασμού, δεν είναι το πλήρες σύνολο των ιδιοτήτων που χαρακτηρίζουν ένα αντικείμενο. Κάποιες ιδιότητες δεν έχουν υπόσταση κατά το σχεδιασμό αλλά μόνο κατά την εκτέλεση της εφαρμογής. Όλο το σύνολο των ιδιοτήτων ενός αντικειμένου ελέγχου μπορούμε να το δούμε σε λίστα από το σύστημα βοήθειας του περιβάλλοντος εργασίας.

Πολλές ιδιότητες είναι κοινές σε όλα τα αντικείμενα ελέγχου (π.χ. **`Name`, `Left`, `Top`**). Επίσης, πολλές ιδιότητες συναντώνται σε κάποιες ομάδες αντικειμένων ελέγχου (π.χ. **`BackColor`, `BorderStyle`, `Visible`** κ.ά. για τα αντικείμενα που παρουσιάζουν κείμενο). Υπάρχουν όμως και ιδιότητες που είναι μοναδικές για ένα συγκεκριμένο είδος αντικειμένου ελέγχου (π.χ. **`Interval`** για το χρονοδιακόπτη, **`PasswordChar`** για το πλαίσιο κειμένου κ.ά.).

Μα είναι αυτό προγραμματισμός;

Αφού τοποθετήσουμε αντικείμενα ελέγχου επάνω στη φόρμα μπορούμε να ζητήσουμε και πάλι την εκτέλεση του προγράμματος. Θα δούμε ότι όλα λειτουργούν κανονικά. Τα πλήκρα διαταγών, όταν "πιέζονται" από το ποντίκι υποχωρούν και επανέρχονται, τα πλήκτρα σημείωσης και επιλογής μαρκάρονται, στα πλαίσια κειμένου γράφεται κείμενο από πληκτρολόγιο, οι ράβδοι κύλισης συμπεριφέρονται σαν να υπάρχει κάτι να κυλίσουν.

Προκύπτει όμως το ερώτημα. Για να γίνονται κάποιες λειτουργίες στον υπολογιστή πρέπει να τον προγραμματίσουμε και εμείς δε γράψαμε ούτε μια γραμμή κώδικα. Πώς έγιναν όλα αυτά;

Σε οποιοδήποτε κλάδο είναι ιδιαίτερα αποδοτικό, κατά την ανάπτυξη μιας εφαρμογής, να αξιοποιούνται έτοιμα ή ημιέτοιμα κομμάτια και να μη δημιουργούνται τα πάντα από την αρχή. Κάτω από αυτή τη λογική και οι προγραμματιστές κοιτούν να χρησιμοποιήσουν έτοιμα ή ημιέτοιμα κομμάτια από δική τους δουλειά είτε από δουλειά που έχουν προετοιμάσει τρίτοι. Οι συνειδητοποιημένοι μάλιστα προγραμματιστές όταν χρειαστεί να δημιουργήσουν κάτι καινούργιο, προσπαθούν να το φτιάξουν με τέτοιο τρόπο, ώστε να μπορέσουν να το εκμεταλλευτούν και στο μέλλον.

Παλαιότερα, στις διαδικασιακές γλώσσες προγραμματισμού, που ο προγραμματισμός θεωρούταν σχεδόν ταυτόσημος με την κωδικοποίηση, οι προγραμματιστές δημιουργούσαν ή χρησιμοποιούσαν έτοιμες "βιβλιοθήκες" από διαδικασίες. Τα δεδομένα πάνω στα οποία δρούσαν αυτές οι διαδικασίες θεωρούνταν ότι ήταν κάτι το οποίο έπρεπε να επιμεληθεί ο προγραμματιστής. Οι σύγχρονες τάσεις του αντικειμενοστρεφούς προγραμματισμού υπαγορεύουν τη δημιουργία "αποθηκών" αντικειμένων που να μπορούν να τα χρησιμοποιήσουν αυτούσια άλλοι προγραμματιστές. Έτσι, οι πληροφορίες που αφορούν χαρακτηριστικά και τρόπους συμπεριφοράς μιας οντότητας ομαδοποιούνται και τοποθετούνται σε μια κοινή συσκευασία. Αυτή η λογική οδήγησε στον ορισμό προγραμματιστικών οντοτήτων παρόμοιων με τα αντικείμενα του φυσικού κόσμου.

Το αντικείμενο αποτελεί ένα συνδυασμό κώδικα και δεδομένων που αντιμετωπίζεται και ελέγχεται ως μονάδα. Κάθε αντικείμενο διαθέτει μια δική του, συγκεκριμένη ομάδα από ***ιδιότητες, μεθόδους*** και ***συμβάντα***. Οι ιδιότητες αποτελούν τα χαρακτηριστικά των αντικειμένων, οι μέθοδοι τον τρόπο συμπεριφοράς τους και τα συμβάντα τα αίτια που μπορούν να προκαλέσουν αλλαγές στις ιδιότητές τους ή ενεργοποίηση των μεθόδων. Μέσω αυτών των στοιχείων επιτυγχάνεται η διαχείριση ενός αντικειμένου και η σύνδεση-συνεργασία του με άλλα αντικείμενα προκειμένου να χτιστεί μια εφαρμογή. Όπως είδαμε, τα γραφικά αντικείμενα που διαθέτει η VB είναι οι φόρμες και τα αντικείμενα ελέγχου που δημιουργούμε με τα προσφερόμενα εργαλεία. Υπάρχουν, ωστόσο, και αντικείμενα που δεν είναι ορατά αλλά εκτελούν ειδικές εργασίες. Πρόκειται για τα ειδικά αντικείμενα `Clipboard`, `Debug`, `Printer`, `Screen`, `App`, και `Err`.

Στο επόμενο μάθημα θα ασχοληθούμε με τις μεθόδους και τα συμβάντα.

Ανακεφαλαίωση

Στον οπτικό, αντικειμενοστραφή και οδηγούμενο από συμβάντα προγραμματισμό, ο προγραμματιστής δημιουργεί σχεδιάζοντας τη διεπαφή χρήστη για την είσοδο δεδομένων και την εμφάνιση αποτελεσμάτων και γράφει τον κώδικα για τα συμβάντα για να υλοποιήσει τις λειτουργίες που θέλει να επιτελεί το πρόγραμμά του.

Οι φόρμες αποτελούν τα θεμελιώδη αντικείμενα της διεπαφής. Οι τιμές των ιδιοτήτων τους καθορίζουν τόσο τα οπτικά χαρακτηριστικά του παραθύρου του οποίου αποτελούν το πρότυπο (π.χ. ιδιότητες **`BorderStyle`**, **`BackColor`**, **`Left`**, **`Top`** κ.ά.), όσο και τον τρόπο συμπεριφοράς τους (π.χ. ιδιότητες **`Movable`**, **`Enabled`**, **`ShowInTaskbar`** κ.ά.). Πάνω στις φόρμες τοποθετούνται τα κατάλληλα αντικείμενα ελέγχου και ρυθμίζονται οι ιδιότητές τους.

Η εκτέλεση του προγράμματος γίνεται με την επιλογή **Run** του μενού, υποεπιλογή **Start** και η διακοπή από την ίδια επιλογή, υποεπιλογή **End**.

Εργαστηριακές Ασκήσεις

1. Δημιουργήστε ένα νέο έργο και:
   α) Πραγματοποιήστε τις αλλαγές ιδιοτήτων της άσκησης 4-1.
   β) Αλλάξτε το μέγεθος της φόρμας "τραβώντας" τις λαβές. Παρατηρήστε την αλλαγή των διαστάσεων στη γραμμή εργαλείων και την αλλαγή του μεγέθους του σκαριφήματός της στο παράθυρο εμφάνισης σκαριφημάτων.
   γ) Αλλάξτε το χρώμα του υποβάθρου.
   δ) Κάντε δοκιμές με όλες τις δυνατές τιμές της ιδιότητας **`BorderStyle`**.
   ε) Τοποθετήστε πλήκτρα γραμμής τίτλου. Πραγματοποιήστε δοκιμές σε συνδυασμό με την ιδιότητα **`BorderStyle`**.
   στ) Ζητήστε την εκτέλεση του προγράμματος για διάφορες τιμές του **`BorderStyle`** και διαπιστώστε την πραγματοποίηση των λειτουργιών του παραθύρου. Κάντε το ίδιο αλλάζοντας τις τιμές από τις ιδιότητες **`MinButton`**, **`MaxButton`**, **`ControlBox`**.
   ζ) Ζητήστε την εκτέλεση του προγράμματος για διάφορες τιμές των ιδιοτήτων **`Visible`**, **`Enabled`**, **`Moveable`**, **`ShowInTaskbar`**. Ελέγξτε τις λειτουργίες του παραθύρου για κάθε τιμή ιδιότητας.
   η) Θα έχετε παρατηρήσει ότι τα προγράμματα των Windows έχουν διαφορετικά εικονίδια στο πάνω αριστερό άκρο του παραθύρου τους δίπλα από το πλήκτρο ελέγχου. Για να αλλάξουμε το εικονίδιο που τοποθετεί από μόνο του το περιβάλλον εργασίας με τη δημιουργία κάθε νέας φόρμας, δίνουμε κατάλληλη τιμή στην ιδιότητα **`Icon`** της φόρμας, καθορίζοντας το αρχείο που περιέχει το εικονίδιο. Στη φόρμα του έργου της άσκησης 1 αλλάξτε το εικονίδιο.

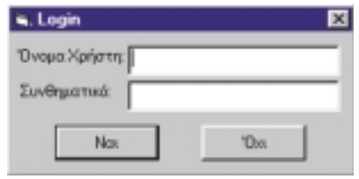
   Υπόδειξη: Το αρχείο πρέπει να είναι κατάλληλης μορφής 16x16 bits και τύπου `.ico`. Κατά την αλλαγή της τιμής της ιδιότητας εμφανίζεται διαλογικό παράθυρο, στο οποίο προσδιορίζουμε το δίσκο, την ιεραρχία των καταλόγων και το αρχείο που βρίσκεται το εικονίδιο. Φάκελοι με τέτοια εικονίδια υπάρχουν π.χ. στον `\Program Files\Microsoft Visual Studio\Common\Graphics\Icons`.

2. Τοποθετήστε πάνω σε μια φόρμα αντικείμενα ελέγχου. Αλλάξτε τις διαστάσεις τους, την αρχική θέση τους, τη στοίχισή τους.

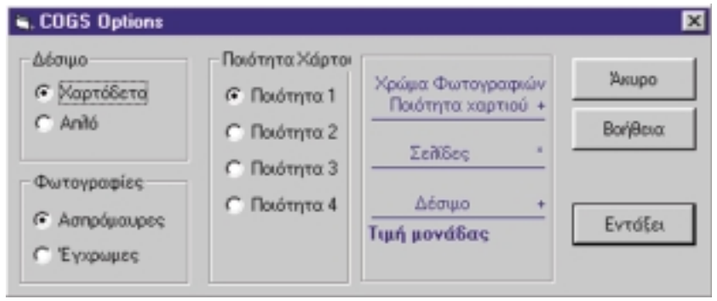
3. Ζητήστε την εκτέλεση του προγράμματος της προηγούμενης άσκησης και κάντε παρατηρήσεις στον τρόπο εστίασης των αντικειμένων ελέγχου πατώντας το πλήκτρο Tab και τα πλήκτρα `Shift + Tab`. Αλλάξτε τη σειρά επιλογής.

4. Πραγματοποιήστε την άσκηση 4-2.

5. Σχεδιάστε μια φόρμα για εισαγωγή συνθηματικού:

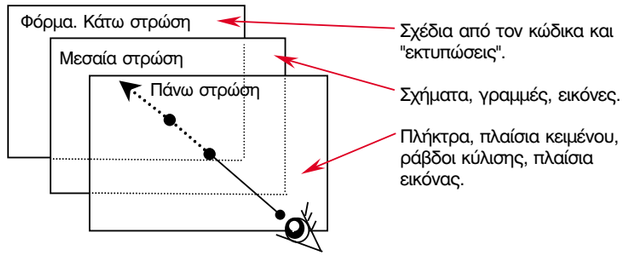


6. Σχεδιάστε τη φόρμα:



7. Σε μια φόρμα να σχεδιάσετε ένα ευθύγραμμο τμήμα, έναν κύκλο, ένα πλήκτρο διαταγής, ένα πλαίσιο κειμένου. Αναδιατάξτε τα αντικείμενα με την **Format ¦ Order** ως προς το βάθος. Τι παρατηρείτε;

   Υπόδειξη: Τα αντικείμενα ελέγχου δεν τοποθετούνται απευθείας πάνω στην επιφάνεια της φόρμας, αλλά σε δύο στρώσεις που βρίσκονται πάνω από αυτή. Και η επιφάνεια της φόρμας αποτελεί στρώση, γι' αυτό συνολικά έχουμε τρεις στρώσεις.



Στην πάνω στρώση τοποθετούνται τα πλαίσια κειμένου, τα πλήκτρα γενικά, οι ράβδοι κύλισης, τα πλαίσια εικόνων. Στη μεσαία στρώση τοποθετούνται τα ευθύγραμμα τμήματα, τα σχήματα και οι εικόνες. Στην κάτω στρώση θα δούμε ότι γίνεται η σχεδίαση μέσα από το πρόγραμμα με εντολές σχεδίασης και παρουσίασης κειμένου. Το μάτι βλέπει την προβολή των αντικειμένων, γι' αυτό και δε διακρίνονται απευθείας οι στρώσεις.

Σημειώσεις:

Μάθημα 5
Προγραμματισμός με συμβάντα

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να περιγράφουν τη φιλοσοφία του οδηγούμενου από συμβάντα προγραμματισμού.
- Να απαριθμούν τα πιο κοινά από τα συμβάντα.
- Να συντάσσουν απλά προγράμματα, στα οποία τροποποιούνται συγκεκριμένες ιδιότητες των αντικειμένων και να καλούν μεθόδους που δρουν πάνω στα αντικείμενα.

Από την κατασκευή των πρώτων σύγχρονων υπολογιστών μέχρι την εποχή που εμφανίστηκαν τα Windows, οι προγραμματιστές δημιουργούσαν λογισμικό εφαρμογών χρησιμοποιώντας διαδικασιακές γλώσσες προγραμματισμού, οι οποίες θεωρούν ότι το πρόγραμμα εκτελείται από μια ***μηχανή Von Neumann***. Σύμφωνα με τις απαιτήσεις των μηχανών αυτού του τύπου, το πρόγραμμα αρχίζει να εκτελείται από την πρώτη του εντολή. Οι εντολές του προγράμματος τοποθετούνται η μια μετά την άλλη με τη σειρά που θα κληθούν από την κεντρική μονάδα επεξεργασίας. Οι παντός είδους έλεγχοι που πραγματοποιούνται πριν από διακλαδώσεις και παρακάμψεις σε διαδικασίες ανάλογα με τις τρέχουσες συνθήκες περιέχονται μέσα στο πρόγραμμα.

Οι μηχανές Von Neumann στηρίζονται στην έννοια του απαριθμητή προγράμματος, ο οποίος καθορίζει τη σειρά ανάκλησης των εντολών από την κύρια μνήμη, προσδιορίζει δηλαδή τη σειριακή ροή ελέγχου του προγράμματος.

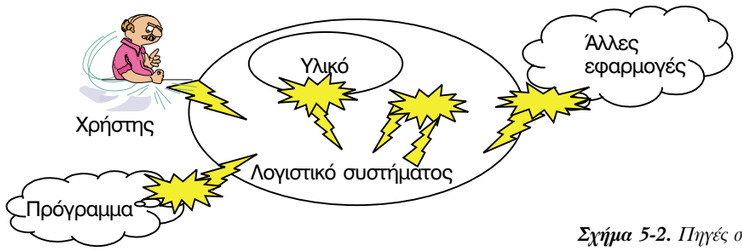
Συμβάντα

Με τα Windows, το ***λογισμικό συστήματος*** (λειτουργικό σύστημα, βιβλιοθήκες ρουτινών, βοηθητικά προγράμματα, οδηγοί συσκευών) κάλυψε τόσο πολύ το υλικό με το μανδύα του, ώστε το λογισμικό εφαρμογών να θεωρεί το υπολογιστικό σύστημα σαν μια άλλου τύπου μηχανή, διαφορετική από τις μηχανές Von Neumann, της οποίας η λειτουργία καθοδηγείται (άγεται και φέρεται καλύτερα) από ***συμβάντα*** (***events***).

Σχήμα 5-1. *Το λογισμικό εφαρμογής μπορεί να εκμεταλλευτεί το υλικό μόνο μέσω του λογισμικού συστήματος.*

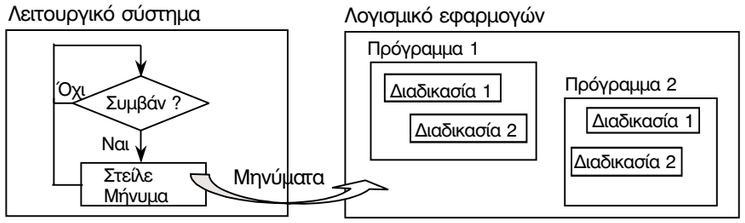
Συμβάν είναι κάτι που προκύπτει από διαφορετικές αιτίες κατά τη διάρκεια λειτουργίας ενός προγράμματος. Συγκεκριμένα,, τα συμβάντα προκαλούνται από:

- το χρήστη, π.χ. όταν κάνει κλικ του ποντικιού, ή πραγματοποιεί μετακίνησή του, όταν κάνει πληκτρολόγηση μέσα σε ένα πλαίσιο κειμένου κ.ά.,
- το ίδιο το πρόγραμμα, ως αποτέλεσμα άλλου συμβάντος, π.χ. ανοίγει ή κλείνει ένα παράθυρο, αποκτά την εστίαση ένα αντικείμενο ελέγχου κ.ά.,
- άλλη εφαρμογή, π.χ. αίτηση για λήψη ή αποστολή μηνύματος,
- το ίδιο το υπολογιστικό σύστημα, π.χ. διακοπή σε τακτά χρονικά διαστήματα που ορίζει ένας χρονοδιακόπτης.



Σχήμα 5-2. *Πηγές συμβάντων*

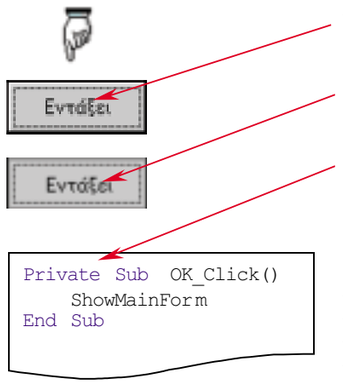
Στον καθοδηγούμενο από συμβάντα προγραμματισμό, ο προγραμματιστής επισυνάπτει στα συμβάντα διαδικασίες, οι οποίες εκτελούνται, όταν προκαλείται το αντίστοιχο συμβάν. Η σειρά με την οποία προκαλούνται τα συμβάντα, επομένως και η σειρά με την οποία θα εκτελεστούν οι διαδικασίες, μπορεί να είναι διαφορετική κάθε φορά και κατά συνέπεια μπορεί να θεωρηθεί χρονικώς μη προβλέψιμη. Για παράδειγμα, το ποιο πλήκτρο θα πατήσει ένας χρήστης από μια σειρά πλήκτρων για να επιλέξει μια διαδικασία, και ποια ακριβώς χρονική στιγμή θα το πράξει, είναι θέμα του χρήστη, και στην περίπτωση που θα θέλαμε ένα πρόγραμμα να περιέχει τέτοιας μορφής ελέγχους, ώστε να αποκρίνεται άμεσα, θα πρέπει να τους επιτελεί σε οποιοδήποτε σημείο του.



Σχήμα 5-3. *Το λειτουργικό σύστημα συνεχώς ελέγχει για συμβάντα και δρομολογεί μηνύματα στις εφαρμογές και τις διαδικασίες τους.*

Τον έλεγχο διαχείρισης των συμβάντων τον αναλαμβάνει το λειτουργικό σύστημα, το οποίο με κατάλληλο ***μηχανισμό αποστολής μηνυμάτων*** (***messaging***) προκαλεί τη διέγερση και την εκτέλεση της κατάλληλης διαδικασίας. Κάθε συμβάν το επεξεργάζεται πρώτα το λειτουργικό σύστημα και αν χρειαστεί μεταβιβάζει μήνυμα στις εφαρμογές. Σε περίπτωση που η εφαρμογή είναι εφαρμογή VB το μήνυμα μεταβιβάζεται στην κατάλληλη ***υπορουτίνα διαχείρισης συμβάντων***, που έχουμε συγγράψει, ώστε να γίνουν οι ενέργειες που έχουμε προγραμματίσει.

Παράδειγμα 5-1.



1. Ο χρήστης κάνει κλικ στο πλήκτρο.
2. Τα Windows παρουσιάζουν το πλήκτρο πατημένο.
3. Στέλνεται μήνυμα να εκτελεστεί η υπορουτίνα διαχείρισης συμβάντος που έχει γράψει ο προγραμματιστής.

Εικόνα 5-1. *Διαδοχή των λειτουργιών κατά το κλικ πάνω σε ένα πλήκτρο. Σε διαδικασιακό προγραμματισμό το πρόγραμμα θα χρειαζόταν πάρα πολλές γραμμές κώδικα για την ίδια λειτουργία.*

Για κάποια συμβάντα το λειτουργικό σύστημα των Windows εκτελεί και κάποια πρόσθετη λειτουργία, π.χ. το κλικ πάνω σε ένα πλήκτρο διαταγής προκαλεί οπισθοχώρηση του πλήκτρου για όσο διάστημα ο χρήστης κρατά πατημένο το πλήκτρο του ποντικιού, το πάτημα ενός πλήκτρου ελαχιστοποίησης στη γραμμή τίτλου ενός παραθύρου προκαλεί ελαχιστοποίηση του παραθύρου κ.ά. Αυτές τις λειτουργίες δεν μπορούμε να τις παρακάμψουμε ή να τις τροποποιήσουμε. Θα εκτελεστούν οπωσδήποτε και μάλιστα ως έχουν. Μπορούμε όμως να τις επαυξήσουμε περιγράφοντας τις επιπλέον λειτουργίες που θα θέλαμε να εκτελεστούν στην αντίστοιχη υπορουτίνα διαχείρισης συμβάντος. Ας σημειωθεί ότι ο προγραμματιστής δεν είναι υποχρεωμένος να δημιουργήσει διαδικασίες για όλα τα συμβάντα που είναι δυνατόν να προκύψουν παρά μόνο για όσα επιθυμεί πρόσθετες λειτουργίες.

Υπορουτίνες διαχείρισης συμβάντων

Έχουμε δει σε προηγούμενο μάθημα, ότι σε κάθε φόρμα επισυνάπτεται και ένα παράθυρο κώδικα. Στο παράθυρο αυτό γράφονται οι υπορουτίνες διαχείρισης συμβάντων, που αφορούν τη φόρμα και τα αντικείμενα ελέγχου της. Ό,τι υπάρχει γραμμένο στα παράθυρα κώδικα όλων των φορμών και στα παράθυρα κώδικα των βασικών προγραμματιστικών μονάδων (modules) αποτελεί το πρόγραμμα της εφαρμογής.

Για να εμφανιστεί το παράθυρο κώδικα μιας συγκεκριμένης φόρμας δείχνουμε τη φόρμα στη λίστα του παραθύρου έργου και πατάμε το πλήκτρο εμφάνισης κώδικα (Code View), ή πιο απλά, κάνουμε δεξί κλικ πάνω στη φόρμα και επιλέγουμε **Code View** από το αναδυόμενο μενού που θα εμφανιστεί.

Μέσα σε μια προγραμματιστική μονάδα φόρμας, κάθε υπορουτίνα διαχείρισης συμβάντος έχει μοναδικό όνομα. Τα ονόματα αυτά δίδονται αυτόματα από το περιβάλλον εργασίας και αποτελούνται από δύο συνθετικά: το όνομα του αντικειμένου, το σύμβολο της υπογράμμισης (_) και το όνομα του *συμβάντος*. Η υπορουτίνα διαχείρισης συμβάντος αρχίζει με τις κωδικές λέξεις `Private Sub` και το όνομά της και τερματίζει με την εντολή `End Sub`. Η γενική μορφή των διαδικασιών διαχείρισης συμβάντων είναι:

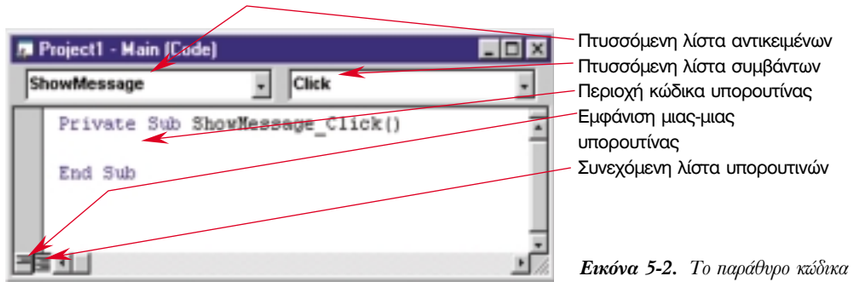
```
Private Sub αντικείμενο_συμβάν()
    εντολή_1
    εντολή_2
      :
      :
End Sub
```

Όνομα υπορουτίνας

Περιοχή όπου περιγράφεται με κώδικα η διαδικασία της υπορουτίνας.

Πριν δημιουργήσουμε υπορουτίνες διαχείρισης συμβάντων για ένα αντικείμενο, πρέπει να έχουμε αποφασίσει για το όνομά του. Αν αλλάξουμε το όνομα ενός αντικειμένου, δεν αλλάζουν αυτόματα και τα ονόματα των υπορουτινών διαχείρισης των συμβάντων του.

Για να δημιουργήσουμε μια υπορουτίνα που διαχειρίζεται ένα συμβάν για ένα αντικείμενο, επιλέγουμε το αντικείμενο από την αριστερή πτυσσόμενη λίστα του παραθύρου κώδικα και το συμβάν από την πτυσσόμενη λίστα στα δεξιά.



Εικόνα 5-2. *Το παράθυρο κώδικα*

Τα συμβάντα δεν είναι κοινά για όλα τα αντικείμενα. Η πτυσσόμενη λίστα συμβάντων ενημερώνεται δυναμικά ανάλογα με το είδος του αντικειμένου που επιλέγουμε στην πτυσσόμενη λίστα αντικειμένων. Μερικά από τα πιο συνηθισμένα συμβάντα είναι:

`Activate:` δημιουργείται από φόρμες τη στιγμή που από ανενεργές γίνονται ενεργές. Κάθε χρονική στιγμή μόνο μια φόρμα είναι ενεργή.

`Change:` προκαλείται με την αλλαγή των δεδομένων που περιέχει το αντικείμενο είτε με ενέργειες του χρήστη από το πληκτρολόγιο ή το ποντίκι είτε από κώδικα.

`Click:` προκύπτει όταν ο χρήστης πατά και αφήνει το πλήκτρο του ποντικιού, ενώ ο δείκτης βρίσκεται πάνω από το αντικείμενο. Όλα σχεδόν τα αντικείμενα δέχονται αυτό το συμβάν. Για μερικά μάλιστα αντικείμενα το συμβάν προκύπτει και με αλλαγή της τιμής τους.

`DoubleClick:` προκύπτει όταν ο χρήστης κάνει διπλό κλικ πάνω στο αντικείμενο. Τα πλήκτρα διαταγής δεν υποστηρίζουν αυτό το συμβάν.

`DragDrop:` προκύπτει στο τέλος της διαδικασίας "**σύρε κι άσε**" (**drag & drop**), που εφαρμόζεται πάνω σε ένα αντικείμενο. Π.χ. κατά τη μεταφορά για αντιγραφή ενός αρχείου από μια θέση αρχείου από μια θέση σε μιαν άλλη.

DragOver: προκύπτει κατά το πέρασμα ενός αντικειμένου στο οποίο εφαρμόζεται η διαδικασία "σύρε κι άσε" (drag & drop) πάνω από ένα άλλο αντικείμενο.

GotFocus: προκαλείται όταν ένα αντικείμενο αποκτά την εστίαση. Συνήθως κατά την εξυπηρέτηση ενός τέτοιου συμβάντος δίνουμε αρχικές τιμές σε κάποιες ιδιότητες του αντικειμένου.

KeyPress: προκύπτει όταν ο χρήστης πατά και αφήνει ένα πλήκτρο του πληκτρολογίου. Το συμβάν το δέχεται το αντικείμενο που έχει την εστίαση. Μια φόρμα μπορεί να δεχτεί το συμβάν μόνο αν περιέχει αόρατα και μη ενεργοποιημένα αντικείμενα ελέγχου ή την ιδιότητα **KeyPreview** με τιμή True.

KeyDown: προκύπτει όταν ο χρήστης πατά ένα πλήκτρο του πληκτρολογίου.

KeyUp: προκύπτει όταν ο χρήστης αφήνει το πλήκτρο του πληκτρολογίου.

Load: προκαλείται όταν φορτώνεται μια φόρμα. Συνηθίζεται τότε να δίνουμε αρχικές τιμές σε κάποιες ιδιότητες τν αντικειμένων της φόρμας.

LostFocus: προκαλείται όταν ένα αντικείμενο χάνει την εστίαση. Συνήθως τότε ελέγχουμε τις τιμές που έχουν δοθεί από ένα χρήστη ή άλλα σημεία του προγράμματος.

MouseMove: προκαλείται όταν ο δρομέας του ποντικιού κινείται πάνω από το αντικείμενο.

MouseDown: προκύπτει όταν ο χρήστης πατά το πλήκτρο του ποντικιού.

MouseUp: προκύπτει όταν ο χρήστης αφήνει το πλήκτρο του ποντικιού.

Τερματισμός του προγράμματος. Η εντολή End.

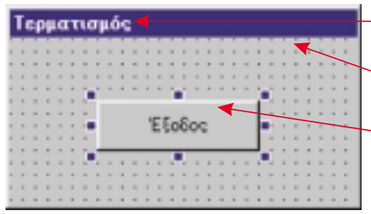
Μάθημα Πρώτο. Πώς σταματάμε!

Το πρόγραμμά μας μπορεί να τερματιστεί αν κάνουμε κλικ στο πλήκτρο κλεισίματος που βρίσκεται στα δεξιά της γραμμής τίτλου του παραθύρου του. Όμως σε πολλά προγράμματα δεν πρέπει ο χρήστης να έχει μια τέτοια δυνατότητα και πρέπει να του επιβάλλεται να ακολουθήσει μια προκαθορισμένη διαδικασία, που να τακτοποιεί τα δεδομένα πριν γίνει ο τερματισμός του προγράμματος. Σε αυτή την περίπτωση δεν εφοδιάζουμε τη γραμμή τίτλου με πλήκτρο τερματισμού. Ο τερματισμός του προγράμματος θα γίνεται τότε με την εκτέλεση της εντολής End.

Άσκηση 5-1.

Να δημιουργηθεί μια φόρμα με ένα πλήκτρο διαταγής, με το όνομα Exit και **Caption** "¸ îïäïò". Όταν πατάμε το πλήκτρο **Exit** θέλουμε να διακόπτεται η εκτέλεση του προγράμματος. Εκτελούμε τις εξής ενέργειες:

- Ζητάμε τη δημιουργία ενός νέου έργου από το περιβάλλον εργασίας και στη φόρμα τοποθετούμε το πλήκτρο διαταγής.
- Δίνουμε τιμές στις ιδιότητες των αντικειμένων.



1. Αλλάζουμε την τιμή της ιδιότητας **Caption** της φόρμας σε "Ôåñìáôéóìüò".
2. Αλλάζουμε την τιμή της ιδιότητας **ControlBox** της φόρμας σε False.
3. Αλλάζουμε την τιμή της ιδιότητας **Caption** του πλήκτρου σε "¸ îïäïò".

Εικόνα 5-3. *Η φόρμα τερματισμού του προγράμματος*

- Κάνουμε διπλό κλικ στο πλήκτρο Exit, οπότε εμφανίζεται το παράθυρο κώδικα, με έτοιμες την τελευταία και την πρώτη γραμμή της υπορουτίνας συμβάντος **Click**.
- Πληκτρολογούμε την εντολή End μέσα στην υπορουτίνα.

```
Private Sub Exit_Click()
    End
End Sub
```


- Ζητάμε την εκτέλεση του προγράμματος από την επιλογή **Run | Start** και ελέγχουμε, αν λειτουργεί σωστά το πλήκτρο.

Ποια όμως είναι η διαφορά μεταξύ της εντολής End Sub και της εντολής End; Η εντολή End Sub σηματοδοτεί, στην ουσία, το τέλος της περιοχής που γράφονται οι εντολές της υπορουτίνας και τερματίζει την εκτέλεση της υπορουτίνας. Όπως θα δούμε στη συνέχεια μετά την εκτέλεσή της (σε άλλες περιπτώσεις πλην αυτού του παραδείγματος) το πρόγραμμα εξακολουθεί την εκτέλεσή του. Κάθε υπορουτίνα διαχείρισης συμβάντος, αλλά και κάθε άλλης μορφής υπορουτίνα έχει για τελευταία εντολή, την εντολή End Sub. Αντίθετα, η εντολή End τερματίζει το πρόγραμμα και προκαλεί τη διακοπή του.

Το πρόγραμμα που μόλις γράψαμε είναι πολύ απλό. Σε άλλα προγράμματα όμως, στα οποία η διαδικασία τερματισμού είναι πιο σύνθετη (απαιτείται να ενημερωθούν αρχεία και βάσεις δεδομένων και να κλείσει η διεπαφή με άλλες εφαρμογές), πριν από την εντολή End μπορεί να υπάρχουν πάρα πολλές εντολές. Αυτό δε σημαίνει ότι την εντολή End τη γράφουμε πάντα στο τέλος μιας υπορουτίνας. Μπορούμε να τη γράψουμε σε οποιοδήποτε σημείο. Επίσης, μέσα στο ίδιο πρόγραμμα μπορούμε να χρησιμοποιήσουμε περισσότερες από μία εντολές τερματισμού.

Εκχώρηση τιμής σε ιδιότητες

Η πιο απλή λειτουργία που μπορούμε να ζητήσουμε να εκτελεστεί μέσα σε μια υπορουτίνα διαχείρισης συμβάντος είναι η αλλαγή της τιμής μιας ιδιότητας ενός αντικειμένου. Η εντολή με την οποία γίνεται αυτή η αλλαγή, είναι μια απλή ***εντολή εκχώρησης τιμής*** (***assignment statement***), η οποία έχει τη μορφή:

αντικείμενο.ιδιότητα = παράσταση

Τον τρόπο σύνταξης των παραστάσεων θα δούμε στο επόμενο μάθημα. Σε αυτό το μάθημα θα χρησιμοποιήσουμε πολύ απλές παραστάσεις.

όπου, *αντικείμενο* το όνομα που έχουμε δώσει στο αντικείμενο, *ιδιότητα* το όνομα της ιδιότητας, της οποίας θέλουμε να αλλάξουμε την τιμή και *παράσταση* μια αριθμητική, λογική ή αλφαριθμητική σταθερά, μεταβλητή ή παράσταση της οποίας η τιμή θα αποδοθεί για τιμή στην ιδιότητα.

Όπως έχουμε δει, μια ιδιότητα μπορεί να πάρει συγκεκριμένου είδους τιμές. Άλλες ιδιότητες μπορούν να πάρουν αριθμητικές τιμές, άλλες μόνο τις τιμές True και False και άλλες αλφαριθμητικές τιμές.

Παραδείγματα 5-2.

α) Έστω ετικέτα, με όνομα Message. Η εντολή:

```
Message.Caption = "Ðñïóï÷Þ! Ìç áðïäåêôÞ ôéìÞ"
```

εμφανίζει στην ετικέτα το κείμενο, που φαίνεται στο δεξί σκέλος, μέσα στα ***διπλά εισαγωγικά*** (***quotes***).

Τα κείμενα γράφονται πάντα μεταξύ διπλών εισαγωγικών.

β) Έστω πλήκτρο διαταγής, με όνομα OK. Η εντολή:

```
OK.Enabled = False
```

απενεργοποιεί το πλήκτρο, ώστε να μη δέχεται την εστίαση και να μην είναι δυνατή η πρόκληση του συμβάντος **Click**, όταν πατιέται το πλήκτρο του ποντικιού πάνω σε αυτό. Τα απενεργοποιημένα πλήκτρα διακρίνονται από την ετικέτα τους που γίνεται γκρι.

γ) Έστω πλαίσιο κειμένου, με όνομα Product. Οι εντολές:

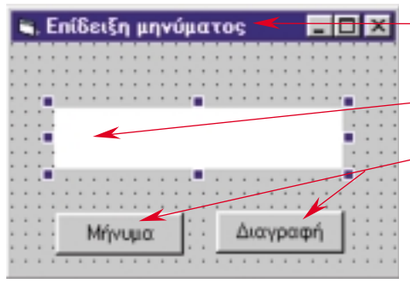
```
Product.Height = 200
Product.Width = 1600
```

του αλλάζουν τις διαστάσεις και τις διαμορφώνουν σε ύψος 200 twips και πλάτος 1600 twips, αντίστοιχα.

Άσκηση 5-2.

Να δημιουργηθεί μια φόρμα με μια ετικέτα, με όνομα `Message`, και δύο πλήκτρα διαταγής με ονόματα `Show` και `Clear`. Όταν πατάμε το πλήκτρο `Show` θέλουμε να εμφανίζεται ένα μήνυμα στην ετικέτα και όταν πατάμε το πλήκτρο `Clear` το μήνυμα να διαγράφεται. Εκτελούμε τις εξής ενέργειες:

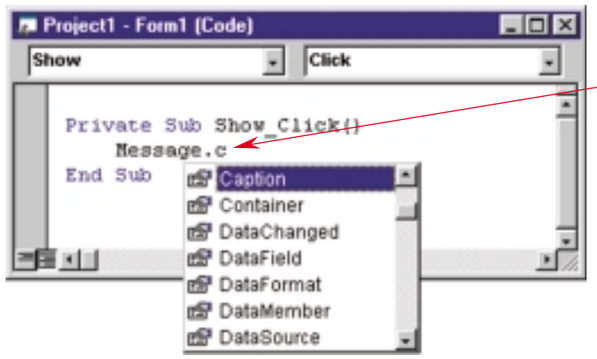
- Ζητάμε τη δημιουργία ενός νέου έργου από το περιβάλλον εργασίας και στη φόρμα τοποθετούμε τα αντικείμενα ελέγχου.
- Δίνουμε τιμές στις ιδιότητες των αντικειμένων.



1. Αλλάζουμε την τιμή της ιδιότητας **`Caption`** της φόρμας σε "Άðßäåéîç ìçíýìáôïò".
2. Διαγράφουμε την τιμή της ιδιότητας **`Caption`** της ετικέτας.
3. Δίνουμε ονόματα στα πλήκτρα.
4. Αλλάζουμε τις τιμές των ιδιοτήτων **`Caption`** των πλήκτρων `Show` και `Clear` σε "Ìçíýìá" και "Äéáãñáöþ" αντίστοιχα.

Εικόνα 5-4. *Παράθυρο μηνύματος*

- Κάνουμε διπλό κλικ στο πλήκτρο `Show`, οπότε εμφανίζεται το παράθυρο κώδικα, με έτοιμες την τελευταία και την πρώτη γραμμή της υπορουτίνας συμβάντος **`Click`**.
- Πληκτρολογούμε την εντολή που θα αλλάζει την ιδιότητα **`Caption`** της ετικέτας `Message`.



Τη στιγμή που γράφουμε την τελεία, που χωρίζει το όνομα της ετικέτας με την ιδιότητα, εμφανίζεται ένα αναδυόμενο μενού με όλες τις διαθέσιμες ιδιότητες τις ετικέτας.Τη στιγμή μάλιστα που πατάμε το πλήκτρο `C` εστιάζεται η ιδιότητα **`Caption`**. Μπορούμε να αποδεχτούμε την ιδιότητα και να αποφύγουμε την πληκτρολόγηση όλου του ονόματος της πατώντας το πλήκτρο `Tab`.

Εικόνα 5-5. *Διαδικασία κωδικοποίησης*

- Κάνουμε διπλό κλικ στο πλήκτρο `Clear` και συμπληρώνουμε τον κώδικα. Οι υπορουτίνες τοποθετούνται μέσα στο παράθυρο κώδικα αλφαβητικά.

```
Private Sub Clear_Click()
    Message.Caption = ""
End Sub

Private Sub Show_Click()
    Message.Caption = "Ãåéü óïõ ößëå!"
End Sub
```

- Ας σημειωθεί ότι τα δύο συνεχόμενα διπλά εισαγωγικά "" αποδίδουν την τιμή ***μηδενική συμβολοσειρά*** (***null string***). Η μηδενική συμβολοσειρά στην ουσία εκκενώνει το περιεχόμενο της ετικέτας.

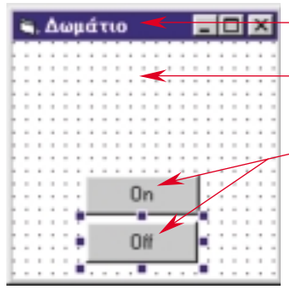
Το αντικείμενο `Me` Σε περίπτωση που το αντικείμενο, στο οποίο θέλουμε να αναφερθούμε, είναι η ίδια η φόρμα του παραθύρου κώδικα, μπορούμε αντί για το όνομά της (που μπορεί να αλλάξει στο μέλλον) να χρησιμοποιούμε την κωδική λέξη `Me`.

Άσκηση 5-3.

Να δημιουργηθεί μια μικρή εφαρμογή στην οποία η φόρμα θα παριστάνει το χώρο ενός δωματίου, στο οποίο υπάρχει μια λάμπα που την ανάβει και τη σβήνει ένας διακόπτης. Αρχικά, η λάμπα είναι αναμμένη, το δωμάτιο φωτίζεται και ο διακόπτης γράφει επάνω του `On`. Όταν πατηθεί ο διακόπτης και σβήνει η λάμπα, το δωμάτιο σκοτεινιάζει και ο διακόπτης γράφει επάνω του `Off`.

Η λάμπα είναι ένα αντικείμενο ελέγχου εικόνας με όνομα `Lamp` και ο διακόπτης δεν είναι ένα μόνο πλήκτρο διαταγής, αλλά δύο, το ένα τοποθετημένο πάνω στο άλλο, με ονόματα `OnButton` και `OffButton`. Εκτελούμε τις εξής ενέργειες:

- Ζητάμε τη δημιουργία ενός νέου έργου από το περιβάλλον εργασίας και στη φόρμα τοποθετούμε τα πλήκτρα διαταγής.
- Δίνουμε τιμές στις ιδιότητες των αντικειμένων.

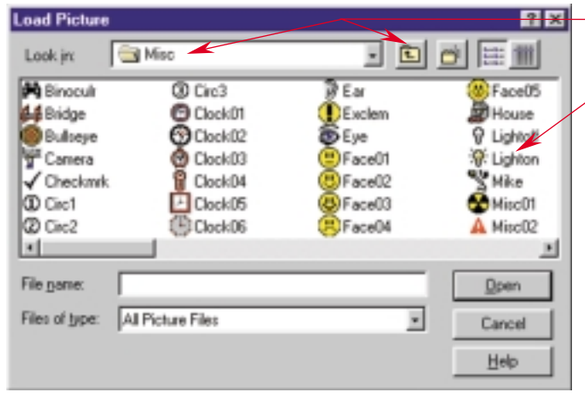


1. Αλλάζουμε την τιμή της ιδιότητας **`Caption`** της φόρμας σε "Äùìüôéï".
2. Αλλάζουμε την ιδιότητα **`BackColor`** σε λευκό (`&H00FFFFFF&`).
3. Αλλάζουμε τις τιμές των ιδιοτήτων **`Caption`** των πλήκτρων `OnButton` και `OffButton` σε "On" και "Off" αντίστοιχα.

Τα χρώματα περιγράφονται στο δεκαεξαδικό σύστημα αρίθμησης.

Εικόνα 5-6. *Αλλάζουμε τιμές στις ιδιότητες*

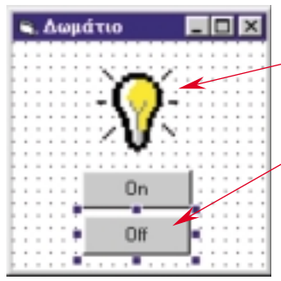
- Τοποθετούμε ένα αντικείμενο ελέγχου εικόνας πάνω στη φόρμα και ζητάμε αλλαγή της τιμής της ιδιότητας **`Picture`**. Εμφανίζεται το διαλογικό παράθυρο:



1. Επιλέγουμε κατάλογο με εικονίδια. `C:\....\VB\Graphics\Icons\Misc`
2. Επιλέγουμε εικονίδιο.

Εικόνα 5-7. *Αλλαγή της ιδιότητας* **`Picture`**

- Διαμορφώνουμε τις υπόλοιπες ιδιότητες των αντικειμένων ελέγχου.



1. Αλλάζουμε την τιμή της ιδιότητας **`Stretch`** σε `True` και αλλάζουμε το μέγεθος της εικόνας.
2. Δίνουμε στην ιδιότητα **`Visible`** του `OffButton` την τιμή `False`.

Η ιδιότητα **`Stretch`** (τέντωμα) ανάλογα με την τιμή της επιτρέπει τη μεγέθυνση ή τη σμίκρυνση της εικόνας που περιέχεται μέσα σε ένα αντικείμενο εικόνας, ταυτόχρονα με τη μεγέθυνση ή τη σμίκρυνση του αντικειμένου.

Εικόνα 5-8. *Συμπληρώνουμε τη φόρμα*

- Κάνουμε διπλό κλικ στο πλήκτρο `Off`, οπότε εμφανίζεται το παράθυρο κώδικα, με έτοιμες την τελευταία και την πρώτη γραμμή της υπορουτίνας συμβάντος **`Click`**.

- Γράφουμε τον κώδικα:

```
Private Sub OffButton_Click()
    OffButton.Visible=False
    OnButton.Visible=True
    Lamp.Visible=False
    Me.BackColor=0
End Sub
```

Όταν πατηθεί το πλήτρο "σβήσε":
1. Κρύψε το πλήκτρο "σβήσε".
2. Εμφάνισε το πλήκτρο "άναψε".
3. Κρύψε τη λάμπα.
4. Σκοτείνιασε το δωμάτιο.

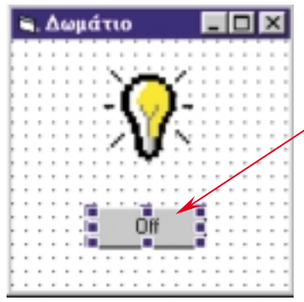
Η κωδική λέξη `Me` υποδηλώνει την ίδια τη φόρμα

- Κάνουμε διπλό κλικ στο πλήκτρο `On` και γράφουμε τον κώδικα:

```
Private Sub OnButton_Click()
    OnButton.Visible=False
    OffButton.Visible=True
    Lamp.Visible=True
    Me.BackColor=&HFFFFFF
End Sub
```

Όταν πατηθεί το πλήτρο "άναψε":
1. Κρύψε το πλήκτρο "άναψε".
2. Εμφάνισε το πλήκτρο "σβήσε".
3. Εμφάνισε τη λάμπα.
4. Φώτισε το δωμάτιο.

- Τελειοποιούμε τη διεπαφή χρήστη.



Με την επιλογή ***Format ¦ Align*** ευθυγραμμίζουμε το ένα πλήκτρο πάνω στο άλλο.

Εικόνα 5-9. *Η τελική διεπαφή*

Μέσα από το πρόγραμμά μας μπορούμε να αναφερόμαστε σε ιδιότητες και ειδικών αντικειμένων, όπως είναι το αντικείμενο `Screen`, το οποίο είναι η ίδια η οθόνη.

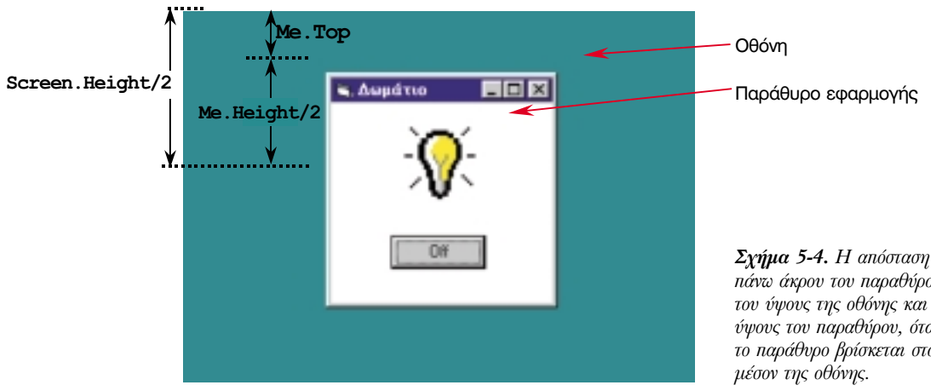
Το αντικείμενο `Screen`

Άσκηση 5-4.

Να συμπληρωθεί η προηγούμενη άσκηση, ώστε όταν τρέχει το πρόγραμμα, το παράθυρο του να εμφανίζεται στο κέντρο της οθόνης.

Με το συμβάν **`Load`**, που προκύπτει όταν φορτώνεται η φόρμα, θα πρέπει η πάνω αριστερή κορφή του παραθύρου να τοποθετείται στην κατάλληλη θέση. Θα υπολογίσουμε με απλή Γεωμετρία τις τιμές που πρέπει να έχουν οι ιδιότητες **`Left`** και **`Top`**. Από ότι φαίνεται στο σχήμα:

```
Me.Top = Screen.Height/2 - Me.Height/2
Me.Left = Screen.Width/2 - Me.Width/2
```



Σχήμα 5-4. *Η απόσταση του πάνω άκρου του παραθύρου, του ύψους της οθόνης και του ύψους του παραθύρου, όταν το παράθυρο βρίσκεται στο μέσον της οθόνης.*

Η υπορουτίνα που θα τοποθετεί το παράθυρο σε αυτή τη θέση είναι:

```
Private Sub Form_Load()
    Me.Left = (Screen.Width - Me.Width) / 2
    Me.Top = (Screen.Height - Me.Height) / 2
End Sub
```

Μέθοδοι

Έχουμε δει ότι τα αντικείμενα αναγνωρίζουν κάποια είδη συμβάντων και ότι μπορούμε να γράψουμε κώδικα γι' αυτά τα συμβάντα, ο οποίος να αλλοιώνει τις ιδιότητές τους ή τις ιδιότητες άλλων αντικειμένων και να τους προσδίδει μια δυναμική συμπεριφορά. Τελειώνοντας με τα χαρακτηριστικά των αντικειμένων έχουμε να αναφέρουμε ότι τα αντικείμενα από κατασκευής τους συνοδεύονται και από ***μεθόδους*** (***methods***). Οι μέθοδοι είναι προτυποποιημένες διαδικασίες, που έχουν κωδικοποιηθεί από τον κατασκευαστή του αντικειμένου σε έτοιμες υπορουτίνες, οι οποίες μπορούν να κληθούν κατά την εκτέλεση του προγράμματος και να προκαλέσουν μια δράση επάνω στο αντικείμενο.

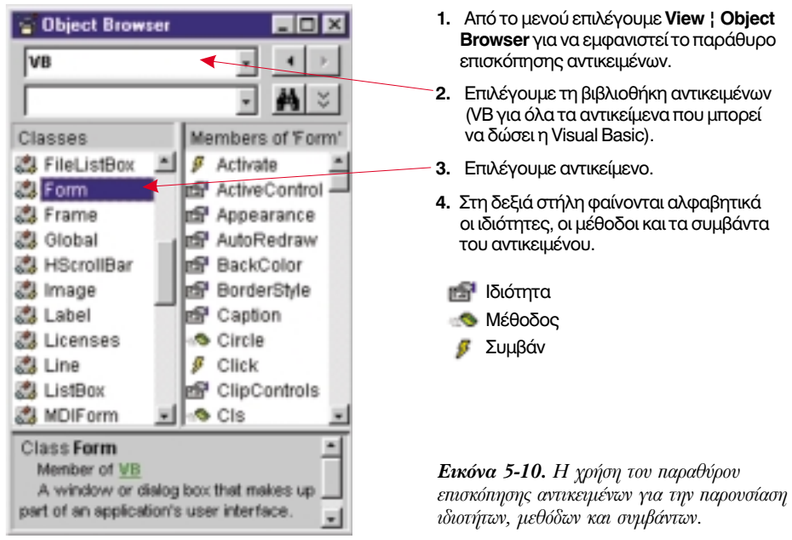
Οι ιδιότητες είναι τα χαρακτηριστικά των αντικειμένων.
Οι μέθοδοι είναι προτυποποιημένες διαδικασίες που συνοδεύουν τα αντικείμενα και οι οποίες όταν ζητηθεί να δράσουν επιδρούν στα αντικείμενα.
Τα συμβάντα είναι οι αφορμές για να αλλαχτούν τα χαρακτηριστικά των αντικειμένων, να ενεργοποιηθούν οι μέθοδοι και να εκτελεστούν και πρόσθετες διαδικασίες.

Η κλήση μιας μεθόδου γίνεται με σαφή προσδιορισμό του αντικειμένου πάνω στο οποίο δρα. Συγκεκριμένα, ισχύει η σύνταξη:

αντικείμενο.μέθοδος παράμετρος1, παράμετρος2,...

όπου, *αντικείμενο* το όνομα που έχουμε δώσει στο αντικείμενο, *μέθοδος* το όνομα της μεθόδου που θέλουμε να ενεργοποιήσουμε και *παράμετρος1, παράμετρος2, ...* μια σειρά από παραμέτρους, που το πλήθος τους εξαρτάται από τη μέθοδο. Συγκρίνοντας τον τρόπο αναφοράς σε μια μέθοδο βλέπουμε ότι αυτός είναι παρόμοιος με αναφορά σε ιδιότητα. Γι' αυτό και κατά τη στιγμή της κωδικοποίησης, η πτυσσόμενη λίστα που εμφανίζεται μετά την πληκτρολόγηση της τελείας περιέχει και μεθόδους.

Οι μέθοδοι δεν είναι κοινές για όλα τα αντικείμενα. Μια λίστα με όλες τις μεθόδους, τις ιδιότητες και τα συμβάντα που αφορούν ένα συγκεκριμένο είδος αντικειμένου μπορούμε να δούμε στο παράθυρο επισκόπησης αντικειμένων.



1. Από το μενού επιλέγουμε **View ¦ Object Browser** για να εμφανιστεί το παράθυρο επισκόπησης αντικειμένων.
2. Επιλέγουμε τη βιβλιοθήκη αντικειμένων (VB για όλα τα αντικείμενα που μπορεί να δώσει η Visual Basic).
3. Επιλέγουμε αντικείμενο.
4. Στη δεξιά στήλη φαίνονται αλφαβητικά οι ιδιότητες, οι μέθοδοι και τα συμβάντα του αντικειμένου.

Εικόνα 5-10. *Η χρήση του παραθύρου επισκόπησης αντικειμένων για την παρουσίαση ιδιοτήτων, μεθόδων και συμβάντων.*

Μερικές από τις πιο συνηθισμένες μεθόδους είναι:

`SetFocus`: δίνει την εστίαση σε ένα αντικείμενο.

`Move` *Left, Top, Width, Height:* μετακινεί ένα αντικείμενο σε μια νέα θέση, ώστε η αριστερή πάνω κορφή του να έχει συντεταγμένες (*Left, Top*). Προαιρετικά, αν υπάρχουν και οι παράμετροι *Width* και *Height*, αλλάζει και τις διαστάσεις του κατά πλάτος και ύψος.

Refresh: σχεδιάζει ξανά το αντικείμενο στην οθόνη.
Show: εμφανίζει μια φόρμα στην οθόνη.
Hide: κρύβει μια φόρμα.

Print *λίστα_ορισμάτων:* τυπώνει τις τιμές της λίστας ορισμάτων στην επιφάνεια του αντικειμένου. Εφαρμόζεται σε φόρμες, στο παράθυρο `Debug` και στο ειδικό αντικείμενο `Printer`.

Cls: καθαρίζει την επιφάνεια του αντικειμένου από ό,τι έχει τυπωθεί ή σχεδιαστεί μέχρι εκείνη τη στιγμή.

Παραδείγματα 5-3.

α) Έστω αντικείμενο με όνομα `Resolution`. Η εντολή:

```
Resolution.SetFocus
```

δίνει την εστίαση στο αντικείμενο. Η μέθοδος συντάσσεται χωρίς παραμέτρους.

β) Η μέθοδος **Move** μπορεί να δεχτεί μέχρι τέσσερεις παραμέτρους. Οι παράμετροι καθορίζουν με τη σειρά την οριζόντια και την κατακόρυφη θέση του πάνω άκρου ενός αντικειμένου καθώς και το πλάτος και το ύψος του. Η θέση του αντικειμένου είναι σχετική ως προς το αντικείμενο, μέσα στο οποίο περιέχεται. Αν `FirstName` είναι ένα πλαίσιο κειμένου, η εντολή:

```
FirstName.Move 0, 0
```

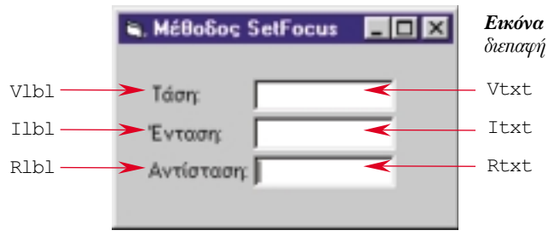
τοποθετεί το πλαίσιο κειμένου στο πάνω αριστερό μέρος του παραθύρου χωρίς να επηρεάζει το ύψος ή το πλάτος - συντεταγμένες (0,0). Επίσης, αν `Main` είναι η φόρμα που περιέχει το αντικείμενο πλαίσιο `FirstName`, η εντολή:

```
Main.Move 0, 0
```

τοποθετεί τη φόρμα στο πάνω αριστερό μέρος της οθόνης.

Άσκηση 5-5.

Σε μια εφαρμογή υπάρχει μια φόρμα με τρεις ετικέτες με **Caption** "ÔÜός:", "¸ íôáός:" και "Áíôßôôáός:" και ονόματα `Vlbl`, `Ilbl` και `Rlbl`, αντίστοιχα. Δίπλα στις ετικέτες υπάρχουν τρία πλαίσια κειμένου με ονόματα `Vtxt`, `Itxt` και `Rtxt`. Μας ζητείται να γράψουμε τις κατάλληλες ρουτίνες, ώστε όταν το ποντίκι περνά πάνω από τις ετικέτες, να δίδεται η εστίαση στο αντίστοιχο πλαίσιο κειμένου.



Εικόνα 5-11. *Το παράθυρο διεπαφής χρήστη*

Εκμεταλλευόμαστε το συμβάν **MouseMove** και τη μέθοδο **SetFocus** και γράφουμε:

```
Private Sub Ilbl_MouseMove(Button As Integer, _
                           Shift As Integer, _
                           X As Single, Y As Single)
    Itxt.SetFocus
End Sub
Private Sub Rlbl_MouseMove(Button As Integer, _
                           Shift As Integer, _
                           X As Single, Y As Single)
    Rtxt.SetFocus
End Sub
```

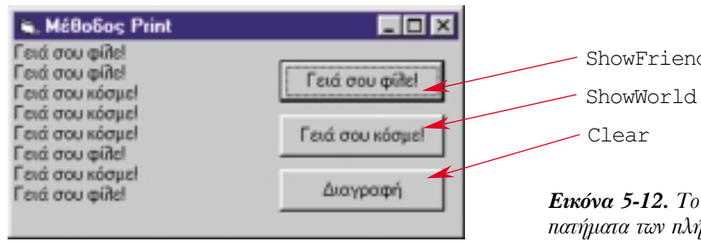
Στο παράθυρο κώδικα διαλέγουμε από το αριστερό πτυσσόμενο μενού το κατάλληλο αντικείμενο και από το δεξί πτυσσόμενο μενού το συμβάν **MouseMove**.

Το σύμβολο της υπογράμμισης (_) χρησιμοποιείται για να δηλώσουμε ότι μια εντολή συνεχίζει στη γραμμή που ακολουθεί.


```
Private Sub Vlbl_MouseMove(Button As Integer, _
                           Shift As Integer, _
                           X As Single, Y As Single)
    Vtxt.SetFocus
End Sub
```

Άσκηση 5-6.

Να δημιουργηθεί μια φόρμα με τρία πλήκτρα διαταγής με ονόματα `ShowFriend`, `ShowWorld` και `Clear`. Όταν πατάμε το πλήκτρο `ShowFriend` να γράφεται ένα μήνυμα στην επιφάνεια της οθόνης, όταν πατάμε το πλήκτρο `ShowWorld` να γράφεται ένα διαφορετικό μήνυμα στην επιφάνεια της οθόνης και όταν πατάμε το πλήκτρο `Clear` να καθαρίζει από τα μηνύματα η επιφάνεια της οθόνης.



Εικόνα 5-12. *Το παράθυρο μετά από μερικά πατήματα των πλήκτρων*

Γράφουμε τον κώδικα:

```
Private lSub Clear_Click()
    Me.Cls
End Sub

Private Sub ShowFriend_Click()
    Me.Print "Ãåéü óïõ ößëå!"
End Sub

Private Sub ShowWorld_Click()
    Me.Print "Ãåéü óïõ êüóìå!"
End Sub
```

Η αναγνωσιμότητα του προγράμματος

Η αναγνωσιμότητα του προγράμματος μπορεί να βελτιωθεί κόβοντας τις μεγάλες εντολές σε περισσότερες από μια γραμμές, συμπτύσσοντας μικρές εντολές σε περισσότερες από μια γραμμές και γράφοντας σχόλια στο εσωτερικό του (εσωτερική τεκμηρίωση).

Συνέχεια εντολής σε περισσότερες από μια γραμμές

Μια εντολή μπορεί να καταλαμβάνει περισσότερες από μια γραμμές. Για να δηλωθεί η συνέχεια μιας εντολής στην επόμενη γραμμή, στο τέλος της τρέχουσας γραμμής τοποθετείται ένα κενό και ο ***χαρακτήρας συνέχειας-γραμμής*** (***line-continuation character***) που είναι η υπογράμμιση (_):

Παραδείγματα 5-4.

α) Ήδη στην άσκηση 5-5 εφαρμόσαμε αυτή την τεχνική:

```
Private Sub Vlbl_MouseMove(Button As Integer, _
                           Shift As Integer, _
                           X As Single, _Y As Single)
```

β) Σε περίπτωση που απαιτείται να "τεμαχίσουμε" κάποιο κείμενο επιβάλλεται να δημιουργήσουμε περισσότερα από ένα κείμενα που τα ενώνουμε με την πράξη &:

```
Message.Text = "Ôï áñ÷åßïï:" _
             & "C:\My Documents\MyFile.txt" _
             & "äåí õðÜñ÷åé óôï öÜêåëï C:\My Documents\"
```

Ο τεμαχισμός μιας εντολής σε περισσότερες από μια γραμμές γίνεται για να παρουσιάζεται όλη η εντολή στην οθόνη και να μη χρειάζεται μετατόπιση του παραθύρου κώδικα δεξιά-αριστερά για να διαβάσουμε την κρυμμένη πληροφορία.

Συνδυασμός εντολών σε μια γραμμή

Σε μια γραμμή μπορούν να τοποθετηθούν περισσότερες από μια εντολές. Σε αυτή την περίπτωση, μεταξύ των εντολών πρέπει να υπάρχει το ***σύμβολο διαχωρισμού*** (***statement separator***) που είναι η άνω και κάτω τελεία (**:**)

Παράδειγμα 5-5.

Οι εντολές:

```
OnButton.Visible = False
OffButton.Visible = True
Lamp.Visible = True
```

Μπορούν να γραφούν σε μια γραμμή:

```
OnButton.Visible = False: OffButton.Visible = True: Lamp.Visible = True
```

Τεκμηρίωση

Αν ξανακοιτάξουμε τον κώδικα ενός προγράμματος που γράψαμε πριν από λίγο καιρό και προσπαθήσουμε να καταλάβουμε τι ακριβώς κάνει ή τι συμβολίζουν οι διάφορες ποσότητες, σίγουρα θα δυσκολευτούμε. Εκεί που τα πράγματα γίνονται ακόμα δυσκολότερα είναι όταν επανεξετάζουμε μεγάλα και σύνθετα προγράμματα, που περιέχουν εκτός από πολύπλοκες διαδικασίες και προγραμματιστικά τεχνάσματα.

Για λόγους εσωτερικής τεκμηρίωσης επιβάλλεται να τοποθετούνται μέσα στα προγράμματά μας σχόλια που να επεξηγούν, τι ακριβώς κάνει το πρόγραμμα, το ρόλο κάθε υπορουτίνας, τα βήματα των αλγορίθμων που χρησιμοποιούνται, τα τεχνάσματα που εφαρμόζονται. Σκοπός της τεκμηρίωσης είναι να κάνει το πρόγραμμα ένα "ομιλόν" κείμενο. Το σχόλιο προλογίζεται πάντα με τον χαρακτήρα (').

Παράδειγμα 5-6.

Οι υπορουτίνες διαχείρισης συμβάντων της άσκησης 5-3 μπορούν να γίνουν πιο αναγνώσιμες με την προσθήκη σχολίων στο εσωτερικό τους.

```
Private Sub OffButton_Click()
    '¼ôáí ðáôçèåß ôï ðëÞñôñ "óâÞóå":
    OffButton.Visible = False       'Êñýøå ôï ðëÞñôñ "óâÞóå"
    OnButton.Visible = True         'ÅìöÜíéóå ôï ðëÞêôñ "Üíáøå"
    Lamp.Visible = False            'Êñýøå ôç ëÜìðá
    Me.BackColor = 0                'Óêïôåßíéáóå ôï äùìÜôéï
End Sub
```

Τα σχόλια τα παρακάμπτει η VB κατά τη διάρκεια μετάφρασης του προγράμματος. Τα σχόλια δεν μπορούν να τοποθετηθούν μαζί με μια εντολή που συνεχίζεται σε περισσότερες από μια γραμμές, ούτε και να γραφούν ενδιάμεσα σε εντολές που υπάρχουν σε μια γραμμή.

Δημιουργία Εκτελέσιμου Αρχείου

Μέχρι τώρα εκτελούμε τα προγράμματα μέσα από το περιβάλλον εργασίας της VB. Για να εκτελούμε το πρόγραμμα από το περιβάλλον των Windows, σαν εκτελέσιμο αρχείο, χωρίς να χρειάζεται να καλούμε το περιβάλλον εργασίας πριν από κάθε εκτέλεσή του, πρέπει να μεταφράσουμε το έργο και να το αποθηκεύσουμε σε ένα αρχείο μορφής .EXE.

Αφού αποθηκεύσουμε ό,τι έχουμε κάνει σε ένα έργο επιλέγουμε από το μενού **File ¦ Make** *όνομα_έργου* **.exe**. Στο παράθυρο που θα εμφανιστεί δίνουμε το όνομα του αρχείου στο οποίο θέλουμε να αποθηκευτεί ο εκτελέσιμος κώδικας. Αυτόματα, το περιβάλλον εργασίας μεταφράζει τις φόρμες τις διαδικασίες και τα άλλα συστατικά σε εκτελέσιμο αρχείο.

Πρέπει να σημειωθεί ότι το πρόγραμμα .EXE που δημιουργείται δεν μπορεί να τρέξει ως έχει σε άλλο υπολογιστή, στον οποίο δεν έχει γίνει εγκατάσταση του περιβάλλοντος εργασίας. Αυτή η μορφή εκτελέσιμου προγράμματος έχει την ανάγκη διαδικασιών που βρίσκονται μέσα σε βιβλιοθήκες. Για να μεταφερθούν όλες οι βιβλιοθήκες που χρειάζονται πρέπει να εκτελέσουμε ειδική διαδικασία δημιουργίας δισκετών εγκατάστασης.

Ανακεφαλαίωση

Στον καθοδηγούμενο από συμβάντα προγραμματισμό, ο προγραμματιστής επισυνάπτει στα συμβάντα διαδικασίες, οι οποίες εκτελούνται, όταν προκαλείται το αντίστοιχο συμβάν. Τον έλεγχο διαχείρισης των συμβάντων τον αναλαμβάνει το λειτουργικό σύστημα, το οποίο με κατάλληλο μηχανισμό αποστολής μηνυμάτων προκαλεί τη διέγερση και την εκτέλεση της κατάλληλης υπορουτίνας διαχείρισης συμβάντων της εφαρμογής.

Η πιο απλή λειτουργία που μπορούμε να ζητήσουμε να εκτελεστεί μέσα σε μια υπορουτίνα διαχείρισης συμβάντος είναι η αλλαγή της τιμής μιας ιδιότητας ενός αντικειμένου. Η εντολή με την οποία γίνεται αυτή η αλλαγή είναι μια απλή εντολή εκχώρησης τιμής, η οποία έχει τη μορφή:

αντικείμενο.ιδιότητα = παράσταση

Τα αντικείμενα από κατασκευής τους συνοδεύονται και από μεθόδους. Οι μέθοδοι είναι προτυποποιημένες διαδικασίες, που έχουν κωδικοποιηθεί από τον κατασκευαστή του αντικειμένου σε έτοιμες υπορουτίνες, οι οποίες μπορούν να κληθούν κατά την εκτέλεση του προγράμματος και να προκαλέσουν μια δράση επάνω στο αντικείμενο.
Η κλήση μιας μεθόδου γίνεται με σαφή προσδιορισμό του αντικειμένου πάνω στο οποίο δρα. Συγκεκριμένα, ισχύει η σύνταξη:

αντικείμενο.μέθοδος παράμετρος1, παράμετρος2,...

Εργαστηριακές Ασκήσεις

1. Δημιουργήστε ένα έργο για την άσκηση 5-1 και εκτελέστε την βήμα προς βήμα.
2. Δημιουργήστε ένα έργο για την άσκηση 5-2 και εκτελέστε την βήμα προς βήμα. Κάντε στη συνέχεια τις κατάλληλες αλλαγές, ώστε αντί για ετικέτα να χρησιμοποιείται πλαίσιο κειμένου.

   Υπόδειξη: Η ιδιότητα που αλλάζει το περιεχόμενο του πλαισίου κειμένου είναι η **Text**.
3. Δημιουργήστε ένα έργο για την άσκηση 5-3 και εκτελέστε την βήμα προς βήμα. Τι πρόβλημα θα δημιουργηθεί αν αντί για αντικείμενο ελέγχου εικόνας χρησιμοποιηθεί πλαίσιο εικόνας;
4. Σε μια φόρμα να τοποθετήσετε τέσσερα πλήκτρα διαταγής. Γράψτε κώδικα, ώστε το πάτημα κάθε πλήκτρου να εμφανίζει και ένα από τα πρόσωπα:

   
5. Δημιουργήστε ένα έργο για την άσκηση 5-4 και εκτελέστε την βήμα προς βήμα.
   Τι αλλαγές χρειάζεται να γίνουν για να τοποθετηθεί το πάνω άκρο του παραθύρου στο 1/3 της οθόνης;
6. Δημιουργήστε ένα έργο για την άσκηση 5-5 και εκτελέστε την βήμα προς βήμα.
7. Δημιουργήστε ένα έργο για την άσκηση 5-6 και εκτελέστε την βήμα προς βήμα.
8. Δημιουργήστε εφαρμογή που να μας αναφέρει τις διαστάσεις της οθόνης σε twips.
9. Δημιουργήστε εκτελέσιμο αρχείο για την άσκηση 5-3.

Σημειώσεις:

Μάθημα 6
Σταθερές, μεταβλητές και παραστάσεις

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να απαριθμούν τα είδη των σταθερών και να περιγράφουν τον τρόπο γραφής τους.
- Να περιγράφουν τα είδη των μεταβλητών και τον τρόπο αποθήκευσής τους στη μνήμη.
- Να συντάσσουν αριθμητικές και αλφαριθμητικές παραστάσεις.

Στο προηγούμενο μάθημα είδαμε ότι μπορούμε να χρησιμοποιούμε μέσα σε ένα πρόγραμμα αριθμούς, λεκτικά, σειρές συμβόλων για να αποδίδουμε τιμές σε ιδιότητες. Γενικά, σε ένα πρόγραμμα μπορούμε να χρησιμοποιούμε ***σταθερές*** (***constants***), δηλαδή αριθμούς, λεκτικά, σειρές συμβόλων, ημερομηνίες που δεν αλλάζουν κατά τη διάρκεια εκτέλεσής του. Επίσης, μπορούμε να ορίσουμε ***μεταβλητές*** (***variables***), δηλαδή ποσότητες που η τιμή τους μπορεί να μεταβληθεί από τις εντολές του προγράμματος, με στόχο να αποθηκεύουμε προσωρινά κάποιες τιμές και να τις χρησιμοποιήσουμε σε άλλα σημεία του προγράμματος ή σε άλλες χρονικές στιγμές. Σε αυτό το μάθημα θα δούμε πώς μπορούμε να συνθέτουμε ***παραστάσεις*** (***expressions***) που βασίζονται πάνω σε τύπους για να κάνουμε υπολογισμούς.

Σταθερές

Η VB είναι εφοδιασμένη με τεσσάρων ειδών σταθερές. Συγκεκριμένα τις:

Αριθμητικές σταθερές (numeric constants)
Λογικές σταθερές (boolean)
Αλφαριθμητικές σταθερές ή συμβολοσειρές (strings)
Ημερομηνίες (dates)

Αριθμητικές σταθερές

Αριθμητικές σταθερές είναι οι κάθε μορφής αριθμοί. Διακρίνουμε δύο ειδών αριθμητικές σταθερές. Τις ***ακέραιες*** (***integer***) και τις ***πραγματικές*** (***real***). Οι ακέραιες σταθερές παριστάνουν ορισμένους από τους ακέραιους αριθμούς, ενώ οι πραγματικές ορισμένους από τους πραγματικούς αριθμούς.

Μπροστά από τις αρνητικές αριθμητικές σταθερές τοποθετείται υποχρεωτικά το πρόσημο (-), ενώ δε απαιτείται το πρόσημο (+) μπροστά από τις θετικές σταθερές.

Το σύμβολο της υποδιαστολής είναι η τελεία (.) και όχι το κόμμα (,) που συνηθίζουμε στην Ελλάδα. Επίσης, δε χρησιμοποιείται κανένα σύμβολο για το χωρισμό των ψηφίων του ακεραίου μέρους σε τριάδες.

Υπάρχει περίπτωση, η VB να αλλάξει τον τρόπο γραφής μιας αριθμητικής σταθεράς σε κάποια δικά της πρότυπα γραφής, αμέσως μετά τη συμπλήρωση της εντολής που την περιέχει. Για παράδειγμα, απομακρύνει τα μηδενικά από την αρχή του ακέραιου μέρους του αριθμού και από το τέλος του μετά την υποδιαστολή. Επίσης, τους πολύ μεγάλους ακέραιους αριθμούς τους αποθηκεύει σαν πραγματικούς και για να μας δηλώσει αυτή την αλλαγή τοποθετεί στο τέλος τους το σύμβολο (#).

Ο υπολογιστής έχει πεπερασμένη μνήμη. Οι αριθμοί αποθηκεύονται σε μικρά τμήματά της και γι' αυτό δεν είναι δυνατή η αποθήκευση οποιουδήποτε αριθμού.

Αν κάποιοι λόγοι μας επιβάλλουν να χρησιμοποιήσουμε μηδενικά στην αρχή του ακεραίου μέρους των αριθμών (π.χ. κωδικός υλικού σε αποθήκη), οι αριθμοί πρέπει να αποθηκευτούν σαν συμβολοσειρές και όχι σαν αριθμητικές σταθερές.

Παραδείγματα 6-1.

Αριθμός	Γραφή VB
-11	`-11`
-1028	`-1028`
0	`0`
0.0	`0#`
7.0	`7#`
-9,83491	`-9.83491`
76.869.211,04	`76869211.04`
250.098.500	`2500985000#`

Στις θετικές επιστήμες, όπως στη Φυσική, στα Μαθηματικά, στην Αστρονομία κ.ά. συναντάμε πολύ μεγάλους και πολύ μικρούς αριθμούς. Αυτούς τους συμβολίζουμε συνήθως με έναν τρόπο που είναι γνωστός ως ***επιστημονικός συμβολισμός*** (***scientific notation***). Στον επιστημονικό συμβολισμό χρησιμοποιούμε ένα μόνο ακέραιο ψηφίο, κάποια ψηφία μετά την υποδιαστολή και την τάξη μεγέθους του αριθμού τη δηλώνουμε με μια δύναμη του 10. Τον ίδιο τρόπο συμβολισμού μπορούμε να χρησιμοποιήσουμε και στην VB, με μόνη τη διαφορά ότι στη θέση της βάσης 10 χρησιμοποιείται το λατινικό γράμμα E.

Παραδείγματα 6-2.

Ποσότητα	Τιμή	Γραφή VB
Όγκος της γής	1.087.000.000.000.000.000.000 m^3 ή 1,087x10^{21} m^3	1.087Å+21
Φορτίο ηλεκτρονίου	0,00000000000000000001602 Cb ή 1,602x10^{-19} Cb	1.602E-19
Γκούγκολ	10^{100}	1E101
	10^{-31}	1Å-30

Πολλές φορές η κωδικοποίηση των αριθμητικών σταθερών γίνεται με λάθος τρόπο. Στην περίπτωση που το λάθος είναι ορθογραφικό, η VB μπορεί να το ανακαλύψει και να το υποδείξει. Υπάρχουν όμως δικά μας λάθη που μπορεί η VB να τα εκλάβει σαν κάτι το διαφορετικό και να τα αποδεχτεί. Σε αυτή την περίπτωση, όπως θα δούμε, η ανακάλυψή τους είναι πολύ δύσκολη.

Παραδείγματα 6-3.

Αριθμός	Γραφή	Λάθος
107,35	107,35	Χρησιμοποιήθηκε το κόμμα για υποδιαστολή.
2,93x$10^{7,6}$	2.93E7.6	Ο εκθέτης δεν είναι ακέραιος.
e^2	1E2	Η βάση δεν είναι το 10.
6^3	6E3	Η βάση δεν είναι το 10.
10^{23}	E23	Δηλώθηκε σαν μεταβλητή.
318x10^0	318E	Λείπει ο εκθέτης.

Η γραφή που είναι σημειωμένη με κόκκινο ανακαλύπτεται ως λάθος από την VB

Οι τιμές των αριθμητικών σταθερών δεν μπορεί να είναι όσο μεγάλες ή όσο μικρές θέλουμε. Τα όρια τους είναι τα ίδια με των μεταβλητών και όπως θα δούμε, είναι μέσα σε πλαίσια που δεν προκαλούν προβλήματα στους κοινούς υπολογισμούς.

Μόνο ρητοί αριθμοί μπορούν να αποθηκευτούν στον υπολογιστή.

Κατά κανόνα η παράσταση των αριθμών στα προγράμματά μας γίνεται στο δεκαδικό σύστημα αρίθμησης. Όμως οι υπολογιστές βασίζουν τη λογική τους στο δυαδικό σύστημα. Έτσι, επιβάλλεται μερικές φορές η χρήση αυτού του συστήματος. Όμως η αποστήθιση και η διαχείριση των δυαδικών αριθμών από τον άνθρωπο είναι δύσκολη, αφού ακόμα και μικροί στο μέγεθος αριθμοί έχουν πάρα πολλά ψηφία στη δυαδική τους παράσταση. Αντί λοιπόν της δυαδικής παράστασης των αριθμών είναι προτιμότερο να χρησιμοποιείται μια ευκολομνημόνευτη παράσταση, η οποία να έχει και το πρόσθετο πλεονέκτημα της εύκολης μετατροπής από και προς το δυαδικό σύστημα. Τις ιδιότητες αυτές τις έχει το οκταδικό και το δεκαεξαδικό σύστημα. Στη VB η δεκαεξαδική παράσταση των αριθμών αρχίζει με το πρόθεμα &H και η οκταδική με το πρόθεμα &O (όμικρον).

Στο προηγούμενο μάθημα είδαμε ότι τα χρώματα της οθόνης για τα αντικείμενα παριστάνονται στο δεκαεξαδικό σύστημα αρίθμησης.

Παραδείγματα 6-4.

Δεκαδικός	Οκταδικός	Δεκαεξαδικός
10	&Ï12	&ÇÁ
16	&O20	&H10
100	&O144	&H64
255	&O377	&HFF
42484	&O122764	&HA5F4
43981	&O125715	&HABCD
16777215	&O77777777	&HFFFFFF

Λογικές σταθερές

Οι λογικές σταθερές είναι οι True (Αληθές) και False (Ψευδές), που ήδη χρησιμοποιήσαμε στην απόδοση τιμών σε ιδιότητες. Αποτελούν κωδικές λέξεις που μπορούν να θεωρηθούν ως συμβολισμοί σταθερών. Όπως θα δούμε, οι λογικές σταθερές χρησιμοποιούνται, όπου απαιτείται να ληφθεί απόφαση.

Αλφαριθμητικές σταθερές ή συμβολοσειρές

Αλφαριθμητικές σταθερές ή συμβολοσειρές είναι οι "λέξεις", οι "φράσεις" και γενικά οι σειρές συμβόλων, που αποτελούνται από γράμματα, ψηφία και ειδικά σύμβολα. Περικλείονται μεταξύ ***διπλών εισαγωγικών*** (") (***quotes***) και συνήθως χρησιμοποιούνται για την παράσταση μη αριθμητικών ποσοτήτων, όπως ονομάτων, διευθύνσεων, μηνυμάτων κ.ά. Από τον ορισμό τους όμως δεν αποκλείεται και η χρήση τους για την παράσταση αριθμητικών στοιχείων, κάτι το οποίο γίνεται σε ειδικές περιπτώσεις.

Το πλήθος των χαρακτήρων από τους οποίους αποτελείται μια συμβολοσειρά ονομάζεται ***μήκος της συμβολοσειράς***. Η μικρότερη σε πλήθος χαρακτήρων συμβολοσειρά είναι η "", που έχει μήκος μηδέν, δεν περιέχει κανένα χαρακτήρα και ονομάζεται ***μηδενική συμβολοσειρά*** (***null string***).

Παραδείγματα 6-5.

Δίδονται μερικές συμβολοσειρές που είναι δυνατό να εμφανιστούν σε μια εντολή της VB.

```
"Ðñüãñáììá ÌáèçìÜôùí"
"ÐëçêôñïëïãÞóôå ôï ÂÜñïò:"
"ÄËÁ×ÏFRAGIKA"
"ÌÝóïò üñïò = 17.1"
"  2004"
"x*/,.F"
"."
"174,45"
"001300889700"
" "
""
```

Σημειώνουμε, ότι μια συμβολοσειρά δεν είναι υποχρεωτικό να είναι κάποια λέξη ή κάποια πρόταση ομιλούμενης γλώσσας. Μπορεί να αποτελείται από οποιοδήποτε συνδυασμό χαρακτήρων του χρησιμοποιούμενου κώδικα. Παρατηρούμε επίσης, ότι η σταθερά "174,45" είναι αποδεκτή παρ' όλο που περιέχει κόμμα μιας και δε λαμβάνεται σαν αριθμός αλλά σαν κείμενο. Ακόμη, στη σταθερά "001300889700" διατηρούνται τα μηδενικά στην αρχή για τον ίδιο ακριβώς λόγο. Τέλος, ας σημειωθεί ότι η σταθερά " " διαφέρει από τη σταθερά "". Η πρώτη έχει μήκος 1, αφού περιέχει ένα χαρακτήρα (το ***διάστημα***) ενώ η δεύτερη είναι η μηδενική συμβολοσειρά.

Ημερομηνίες

Οι ημερομηνίες αντιμετωπίζονται από την VB με έναν ειδικό τρόπο. Δε γράφονται σαν συμβολοσειρές όπως γίνεται με άλλες γλώσσες. Περικλείονται μεταξύ συμβόλων (#) και ακολουθούν τον αμερικανικό τρόπο συμβολισμού. Η γενική τους μορφή είναι:

#μμ/ηη/εεεε#

Παρατηρούμε, ότι πρώτα γράφεται ο μήνας και μετά η ημέρα. Το έτος το γράφουμε πάντα τετραψήφιο. Η διαχείριση των ημερομηνιών είναι αριθμητική. Έτσι, με αριθμητικές πράξεις είναι δυνατόν να βρεθεί η διαφορά σε μέρες μεταξύ δύο ημερομηνιών ή η ημερομηνία μετά από ένα χρονικό διάστημα.

Η συνήθεια γραφής μόνο των δύο τελευταίων ψηφίων του έτους προκάλεσε μεγάλη αναστάτωση τον περασμένο αιώνα. Πολλοί δε γνώριζαν αν τα προγράμματά τους θα θεωρούσαν το 00 σαν 2000 ή σαν 1900.

Παραδείγματα 6-6.

Ημερομηνία	Τρόπος γραφής	Σχόλιο
28η Οκτωβρίου 2004	#10/28/2004#	
15η Ιουνίου 2010	#6/15/2010#	Μονοψήφιος μήνας.
5η Αυγούστου 1992	#8/5/1992#	Μονοψήφιος μήνας και μονοψήφια μέρα.
3η Μαρτίου 2001	#3/3/2001#	Προσοχή! Δυσδιάκριτη ημερομηνία. Να γράφεται σχόλιο στον κώδικα.
29η Φεβρουαρίου 2005	#2/29/2005#	Κατά την πληκτρολόγηση το περιβάλλον εργασίας δεν κάνει αποδεκτή την ημερομηνία.

Συμβολική παράσταση των σταθερών

Ο συμβολισμός μιας σταθεράς με ένα συμβολικό όνομα, όχι μόνο βελτιώνει την αναγνωσιμότητα του προγράμματος αλλά και δημιουργεί τις κατάλληλες προϋποθέσεις για να γίνει πιο ευέλικτο στις αλλαγές και λιγότερο επιρρεπές σε λάθη.

Για να δηλώσουμε μια ***συμβολική σταθερά*** (***symbolic constant***) πρέπει να χρησιμοποιήσουμε τη δηλωτική εντολή σταθεράς, της οποίας μια μορφή είναι η:

```
Const συμβολική_σταθερά = τιμή
```

Η *τιμή* είναι συνήθως μια αριθμητική ή αλφαριθμητική σταθερά, ή σταθερά ημερομηνίας. Μπορεί όμως να είναι και μια παράσταση που δεν περιέχει συναρτήσεις.

Παραδείγματα 6-7.

α) Η σταθερά 3,141592 μπορεί να συμβολιστεί σαν π. Ο ορισμός του συμβολισμού γίνεται με την δήλωση:

```
Const ð = 3.141592
```

Μπορούμε να ορίσουμε ονόματα χρησιμοποιώντας και ελληνικά γράμματα. Κάτι τέτοιο όμως σπάνια το συνηθίζουν οι προγραμματιστές.

β) Το διπλάσιο του αριθμού 3,141592 μπορεί να δηλωθεί :

```
Const ð2 = 2 * ð
```

Δεν μπορεί να δηλωθεί σαν 2π, επειδή οι σταθερές δεν επιτρέπεται να αρχίζουν από ψηφίο.

γ) Το πλήθος των εργάσιμων ημερών μιας εβδομάδας μπορεί να συμβολιστεί σαν `NumOfWorkDays`. Ο ορισμός του συμβολισμού γίνεται με τη δήλωση:

```
Const NumOfWorkDays = 5
```

Μια μελλοντική αλλαγή του πλήθους των εργάσιμων ημερών από 5 σε 4, δε θα απαιτήσει την αναζήτηση του αριθμού 5 μέσα σε όλο το πρόγραμμα και την αντικατάστασή του με το 4, κάτι το οποίο μπορεί να προκαλέσει λάθη, μιας και ο αριθμός 5 μπορεί να μην αντιστοιχεί παντού στο πλήθος των εργάσιμων ημερών. Η αντικατάσταση θα γίνει μόνο στο σημείο που γίνεται ο ορισμός του συμβολισμού.

δ) Η ημερομηνία #1/1/2000# αν θεωρηθεί σαν ημερομηνία αναφοράς μπορεί να συμβολιστεί σαν `StartDate`. Ο ορισμός του συμβολισμού γίνεται με τη δήλωση:

```
Const StartDate = #1/1/2000#
```

Η VB, για πολλές ποσότητες και ιδιαίτερα τιμές ιδιοτήτων, έχει ορίσει εσωτερικά και αναγνωρίζει κάποιο συμβολισμό για να μας απαλλάξει από τον κόπο αναζήτησης συμβολικών ονομάτων και για να μας βοηθήσει να αποφύγουμε τη δημιουργία μη συμβατών μεταξύ των προγραμμάτων ορισμών. Οι σταθερές αυτές ονομάζονται ***εσωτερικές σταθερές*** (***intrinsic constants***).

Παραδείγματα 6-8.

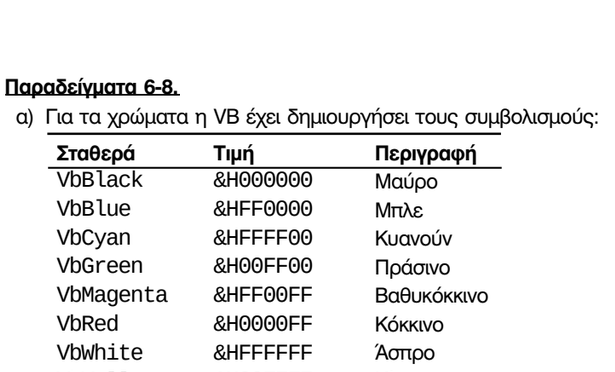
α) Για τα χρώματα η VB έχει δημιουργήσει τους συμβολισμούς:

Σταθερά	Τιμή	Περιγραφή
VbBlack	&H000000	Μαύρο
VbBlue	&HFF0000	Μπλε
VbCyan	&HFFFF00	Κυανούν
VbGreen	&H00FF00	Πράσινο
VbMagenta	&HFF00FF	Βαθυκόκκινο
VbRed	&H0000FF	Κόκκινο
VbWhite	&HFFFFFF	Άσπρο
VbYellow	&H00FFFF	Κίτρινο

β) Στις οπτικές ιδιότητες των παραθύρων είδαμε, ότι η ιδιότητα **`BorderStyle`** καθορίζει τον τύπο πλαισίου των παραθύρων και δώσαμε τις τιμές που μπορεί να πάρει σε λίστα. Γι' αυτές τις τιμές έχουν οριστεί οι σταθερές:

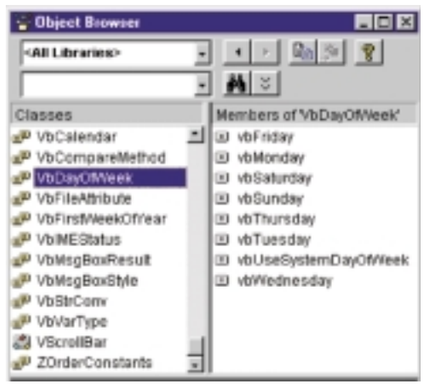
Σταθερά	Τιμή	Περιγραφή
vbBSNone	0	Χωρίς πλαίσιο
vbFixedSingle	1	Σταθερό πλαίσιο, απλό
vbSizable	2	Μεταβλητού μεγέθους πλαίσιο
vbFixedDialog	3	Διαλογικό πλαίσιο
vbFixedToolWindow	4	Σταθερό πλαίσιο εργαλειοθήκης
vbSizableToolWindow	5	Μεταβλητού μεγέθους πλαίσιο εργαλειοθήκης

Τους συμβολισμούς των σταθερών που έχουν οριστεί στη VB, μπορούμε να τους βρούμε χρησιμοποιώντας είτε την άμεση βοήθεια, είτε το παράθυρο επισκόπησης αντικειμένων. Επίσης, σε πολλές περιπτώσεις τη στιγμή της πληκτρολόγησης κώδικα εμφανίζεται αναδυόμενο μενού με σταθερές που μπορούμε να επιλέξουμε.



1. Καλούμε την άμεση βοήθεια από την επιλογή **Help ¦ Index**.
2. Δίνουμε σαν κωδική λέξη, τη λέξη Constants και ψάχνουμε στη λίστα το είδος της σταθεράς.

Εικόνα 6-1. *Σταθερές για την ευθυγράμμιση αντικειμένων ελέγχου μέσα στο παράθυρο*



Εικόνα 6-2. *Σταθερές για τις ημέρες της εβδομάδας*

Μεταβλητές

Πολύ συχνά προκύπτει η ανάγκη να παραστήσουμε με συμβολικό τρόπο διάφορα μεγέθη, των οποίων οι τιμές μεταβάλλονται κατά τη διάρκεια της εκτέλεσης του προγράμματος. Η συμβολική παράσταση γίνεται με λέξεις που ονομάζονται ***μεταβλητές*** (***variables***). Για κάθε μεταβλητή ο υπολογιστής δεσμεύει ένα τμήμα της κεντρικής μνήμης του, στο οποίο καταχωρίζει την τιμή που δίνεται στη μεταβλητή από τις εντολές του προγράμματος.

Τα ονόματα των μεταβλητών (και των σταθερών) πρέπει να υπακούουν στους εξής κανόνες:

- Το όνομα δεν πρέπει να συμπίπτει με δεσμευμένες λέξεις (keywords), δηλαδή λέξεις που έχουν κάποιο ιδιαίτερο νόημα για την VB (π.χ. `False`, `Const`, `If`, `Loop`, `Case` κ.ά.).
- Τα ονόματα πρέπει να αρχίζουν με γράμμα.
- Τα ονόματα αποτελούνται από γράμματα, αριθμούς και χαρακτήρες υπογράμμισης (_). Δεν επιτρέπονται άλλα σύμβολα (π.χ. σύμβολα στίξης, παρενθέσεις κ.λπ.) ή κενά μέσα σε ένα όνομα.
- Τα ονόματα, δεν πρέπει να ξεπερνούν τους 255 χαρακτήρες. Όμως, όπως έχουμε τονίσει, και για τα ονόματα των αντικειμένων πρέπει να φροντίζουμε να χρησιμοποιούμε όσο το δυνατόν μικρότερα ονόματα.
- Το όνομα μπορεί να περιέχει ελληνικούς χαρακτήρες.

Στις μεταβλητές δίνουμε συνήθως τέτοια ονόματα, ώστε να μας θυμίζουν τα μεγέθη και τις ποσότητες που αντιπροσωπεύουν.

Παραδείγματα 6-9.

Μέγεθος	Μεταβλητή
Αντίσταση	`Resistance`
Χωρητικότητα	`Capacitance`
Επιτάχυνση βαρύτητας	`g`
Γενικό σύνολο	`GrandTotal`
Μέγιστος	`Max`
Πάνω άκρο	`TopCorner`
Μετρητής	`Counter`
Τρέχουσα ημερομηνία	`CurrentDate`
Προηγούμενη χρονιά	`LastYear`
Προμηθευτής	`Supplier`
Περιγραφή	`Description`
Διεύθυνση	`Address`
Πόλη-Χώρα	`City_Country`

Παραδείγματα 6-10.

Οι επιλογές ονομάτων που έχουμε κάνει πιο κάτω είτε δεν είναι αποδεκτές από την VB (σημειώνονται με κόκκινο) είτε δεν είναι επιτυχημένες.

Μέγεθος	Μεταβλητή	
Μήνας	`Month`	Συμπίπτει με κωδική λέξη.
Υποτείνουσα	`A`	Δε θυμίζει τίποτε μετά από λίγο καιρό.
Εμβαδόν	`Volume`	Volume=όγκος. Παραπλανητικό όνομα.
Από	`From`	Συμπίπτει με κωδική λέξη.
Μεταβλητή ΠΡΟΠΟ	`12X`	Αρχίζει από ψηφίο.
Ποσοστιαία τιμή	`percentvalue`	Δυσανάγνωστη μεταβλητή
Γενικό σύνολο	`Grand.Total`	Περιέχει σύμβολο στίξης.
Μετρητής χρόνου	`Time Counter`	Περιέχει κενό.

Τύποι μεταβλητών

Οι μεταβλητές, όπως και οι σταθερές, διακρίνονται σε αριθμητικές μεταβλητές, σε λογικές μεταβλητές (οι οποίες ονομάζονται και ***δίτιμες***), σε αλφαριθμητικές μεταβλητές ή μεταβλητές συμβολοσειρών και σε μεταβλητές ημερομηνιών. Επίσης, οι αριθμητικές μεταβλητές διακρίνονται σε ακέραιες και σε δεκαδικές.

Οι λογικές μεταβλητές ονομάζονται και δίτιμες μια και μπορούν να πάρουν μόνο μια από δύο τιμές.

Το μέγεθος της περιοχής της κεντρικής μνήμης του υπολογιστή, που δεσμεύεται για κάθε είδος μεταβλητής, εξαρτάται από το είδος της μεταβλητής. Στην περίπτωση που η μεταβλητή είναι αριθμητική εξαρτάται και από το πόσο μεγάλες τιμές μπορεί να πάρει καθώς και από την ακρίβεια με την οποία μπορεί να τις παραστήσει. Ο τρόπος με τον οποίο αποθηκεύονται οι τιμές των μεταβλητών στη μνήμη, σε ό,τι αφορά το πλήθος των bytes και την παράσταση κάθε bit, καθορίζεται από τον ***τύπο δεδομένων*** (***data type***) της μεταβλητής.

Τύπος δεδομένων	Μνήμη	Αποδεκτές τιμές
`Byte` (μικρός ακέραιος)	1 byte	Ακέραιοι από 0 έως 255
`Boolean`	2 bytes	`True` ή `False`
`Integer`	2 bytes	Ακέραιοι από -32.768 έως 32.767
`Long` (μεγάλος ακέραιος)	4 bytes	Ακέραιοι από -2.147.483.648 έως 2.147.483.647
`Currency` (νόμισμα)	8 bytes	Από -922.337.203.685.477,5808 έως 922.337.203.685.477,5807
`Single` (απλής ακριβείας)	4 bytes	Το 0 και πραγματικοί με απόλυτη τιμή μεταξύ1,401298x10-45 και 3,402823x1038
`Double` (διπλής ακριβείας)	8 bytes	Το 0 και πραγματικοί με απόλυτη τιμή μεταξύ 4,94065645841247x10-324 και 1,79769313486232x10308
`Date`	8 bytes	Από 1η Ιανουαρίου 100 έως 31 η Δεκεμβρίου 9999
`String` (μεταβλητού μεγέθους)	10 bytes + πλήθος χαρακτήρων	Από 0 έως 2.000.000.000 χαρακτήρες περίπου
`String *` (σταθερού μεγέθους)	Πλήθος χαρακτήρων	Από 1 χαρακτήρα έως 65.400 χαρακτήρες περίπου
`Variant` (με αριθμούς)	16 bytes	Κάθε αριθμητική τιμή που μπορεί να δεχτεί ο τύπος `Double`
`Variant` (με χαρακτήρες)	22 bytes + πλήθος χαρακτήρων	Κάθε συμβολοσειρά που μπορεί να δεχτεί ο τύπος String μεταβλητού μεγέθους

Πίνακας 6-1. *Οι τύποι μεταβλητών σε σχέση με το μέγεθος μνήμης που απαιτούν και την περιοχή από την οποία μπορούν να δεχθούν τιμές.*

Για να εκμεταλλευτούμε όσο το δυνατόν ορθολογιστικότερα τη μνήμη του υπολογιστή και για να πετύχουμε όσο το δυνατόν μεγαλύτερες ταχύτητες εκτέλεσης του προγράμματός μας, φροντίζουμε να επιλέξουμε όχι μόνο τύπους δεδομένων που να μπορούν να δεχτούν τα είδη των δεδομένων που θα χρησιμοποιήσουμε, αλλά και όσο το δυνατόν απλούστερους τύπους δεδομένων.

Δήλωση μεταβλητής. Επιλογή τύπου μεταβλητής.

Πριν αναφέρουμε το όνομα μιας μεταβλητής μέσα στο πρόγραμμα καλό είναι να την δηλώσουμε, ώστε η VB να δεσμεύσει τον κατάλληλο χώρο στη μνήμη και να καθορίσει τον τρόπο με τον οποίο θα τη χειριστεί. Η δήλωση μιας μεταβλητής γίνεται με μια δηλωτική εντολή μεταβλητής, η οποία αρχίζει, συνήθως με την κωδική λέξη `Dim`:

Οι δηλωτικές εντολές μεταβλητών είναι δυνατόν να αρχίζουν και με τις κωδικές λέξεις, `Private`, `Static` και `Public`.

`Dim` όνομα_*μεταβλητής* `As` *τύπος*

όπου όνομα_*μεταβλητής*, το όνομα της μεταβλητής και *τύπος*, ένας από τους τύπους δεδομένων.

Κάθε τύπος χαρακτηρίζεται από το είδος των δεδομένων που μπορεί να δεχτεί. Η επιλογή του τύπου της μεταβλητής εξαρτάται από το είδος των δεδομένων που πρόκειται να αποθηκευτούν σε αυτήν κατά τη διάρκεια εκτέλεσης του προγράμματος.

Αν τα δεδομένα που πρόκειται να δεχτεί μια μεταβλητή υποστηρίζονται από περισσότερους του ενός τύπους, πρέπει να γίνεται προσπάθεια χρησιμοποίησης τύπων, όσο το δυνατόν ψηλότερα στον πίνακα 6-1. Ο λόγος είναι διπλός. Εξοικονόμηση μνήμης και επίτευξη ταχύτερης εκτέλεσης του προγράμματος. Παρατηρήστε, ότι οι τύποι που βρίσκονται στις πρώτες θέσεις του πίνακα δεσμεύουν λιγότερη μνήμη από τους τύπους που βρίσκονται σε χαμηλότερες θέσεις. Επίσης, η ταχύτητα εκτέλεσης των πράξεων είναι μεγαλύτερη όσο ψηλότερα βρίσκεται ο τύπος των δεδομένων στον πίνακα.

Παραδείγματα 6-11.

α) Μια μεταβλητή, στην οποία θα δώσουμε το όνομα `Counter` πρόκειται να πάρει ακέραιες τιμές μέσα από το διάστημα [1, 10000]. Συγκρίνοντας τα όρια του διαστήματος, με τα όρια των τύπων του πίνακα καταλήγουμε ότι η καταλληλότερη δήλωση είναι η:

```
Dim Counter      As Integer        'ÁðáñéèìçôÞò
```

β) Μια μεταβλητή, στην οποία θα δώσουμε το όνομα `Max` προβλέπεται, ότι πρόκειται να πάρει ακέραιες τιμές μέσα από το διάστημα [-20000, 100000]. Συγκρίνοντας τα όρια του διαστήματος με τα όρια των τύπων του πίνακα καταλήγουμε, ότι η καταλληλότερη δήλωση είναι η:

```
Dim Max          As Long           'ÌÝãéóôïò ìåôáîý áñéèìþí
```

γ) Μια μεταβλητή, στην οποία θα δώσουμε το όνομα `GrandTotal` προβλέπεται ότι πρόκειται να πάρει ακέραιες τιμές μέσα από το διάστημα [0, 4.000.000.000]. Από ότι φαίνεται, παρ' όλο που οι τιμές της είναι ακέραιες, δεν είναι δυνατή η χρήση ενός ακεραίου τύπου. Ο τύπος `Currency` είναι ο κατάλληλος, συνηθίζεται όμως να χρησιμοποιείται ο τύπος `Double` για να υπάρχει συμβατότητα στην ανταλλαγή δεδομένων με άλλες γλώσσες προγραμματισμού:

```
Dim GrandTotal   As Double         'Ãåíéêü óýíïëï
```

Ο τύπος `Single` δεν προτιμήθηκε γιατί, παρ' όλο που ικανοποιεί το εύρος τιμών, δεν παρέχει την απαιτούμενη ακρίβεια μιας και μπορεί να παραστήσει αριθμούς με ακρίβεια 7 ψηφίων.

δ) Σε ένα πρόγραμμα γίνονται υπολογισμοί για αντιστάσεις και πυκνωτές που η ακρίβειά τους είναι της τάξης του 5%. Για τις μεταβλητές `Resistance` και `Capacitance` οι καταλληλότερες δηλώσεις είναι:

```
Dim Resistance   As Single         'Áíôßóôáóç
Dim Capacitance  As Single         '×ùñçôéêüôçôá
```

ε) Γνωρίζουμε ότι η μεταβλητή για την επωνυμία ενός προμηθευτή αποτελείται από 20 το πολύ χαρακτήρες. Αντίθετα, αγνοούμε το μέγεθος της μεταβλητής που χρησιμοποιείται για την εισαγωγή σχολίων σχετικών με τον προμηθευτή. Οι πιο κατάλληλες δηλώσεις είναι:

```
Dim Supplier     As String * 20    'Âðùíõìßá ðñïìçèåõôÞ
Dim Comments     As String         'Ó÷ïëéá          "
```

Η μεταβλητή `Comments` είναι μεταβλητού μεγέθους. Το μέγεθός της καθορίζεται από την τιμή της. Αντίθετα, η μεταβλητή Supplier είναι σταθερού μεγέθους. Στην περίπτωση που της δοθεί τιμή με μέγεθος μικρότερο από το μήκος της, ο υπόλοιπος χώρος συμπληρώνεται με κενά. Αν όμως της δοθεί τιμή μεγαλύτερου μήκους γίνεται αποκοπή στους 20 χαρακτήρες.

Αν είναι άγνωστος ο τύπος των δεδομένων που πρόκειται να δεχτεί μια μεταβλητή μπορεί να χρησιμοποιηθεί ο τύπος `Variant` (***παραλλαγή***). Ο τύπος αυτός, όπως δηλώνει και το όνομά του, μπορεί να παραλλάσσεται σαν χαμαιλέων και να δέχεται διαφορετικού τύπου δεδομένα κατά περίπτωση, χωρίς να είναι αναγκαίο να γίνει μετατροπή τους. Ας σημειωθεί επίσης, ότι στην περίπτωση που δε δηλώσουμε τον τύπο δεδομένων για μια μεταβλητή, η μεταβλητή θεωρείται του τύπου `Variant`.

Παράδειγμα 6-12.

Στα εμπορικά και τα τεχνικά προβλήματα η ταυτοποίηση προϊόντων δε γίνεται λεκτικά με μια περιγραφή, αλλά με έναν κωδικό. Για παράδειγμα, οι δίοδοι, τα τρανζίστορ και τα ολοκληρωμένα κυκλώματα ταυτοποιούνται από τον κωδικό τους. Επίσης, τα τυποποιημένα προϊόντα στα ράφια των καταστημάτων ταυτοποιούνται με τον κωδικό που περιγράφει ένας γραμμωτός κωδικός (bar code).

Οι δίοδοι, τα τρανζίστορ και τα ολοκληρωμένα κυκλώματα ταυτοποιούνται από τον κωδικό τους.

Ας υποθέσουμε ότι δε γνωρίζουμε, αν ο κωδικός είναι αριθμός ή συμβολοσειρά. Ο πιο κατάλληλος τρόπος δήλωσης της μεταβλητής για την αποθήκευση του κωδικού είναι:

```
Dim ProductCode  As Variant   'Êùäéêüò ðñïúüíôïò
```

ή αλλιώς:

```
Dim ProductCode               'Êùäéêüò ðñïúüíôïò
```

Η εμβέλεια των σταθερών και των μεταβλητών

Ως ***εμβέλεια*** (***scope***) μιας συμβολικής σταθεράς ή μιας μεταβλητής ορίζουμε την περιοχή του κώδικα μέσα στην οποία είναι δυνατόν να χρησιμοποιηθεί η τιμή της. Η εμβέλεια καθορίζεται από το σημείο, στο οποίο θα γίνει η δήλωση και από την κωδική λέξη της εντολής που θα χρησιμοποιηθεί για τη δήλωση. Ανάλογα με την εμβέλεια διακρίνουμε τριών ειδών συμβολικές σταθερές και μεταβλητές:

- Τοπικές (local)
- Προγραμματιστικής μονάδας (private)
- Καθολικές (public)

Τοπικές συμβολικές σταθερές και μεταβλητές

Οι δηλωτικές εντολές σταθερών και μεταβλητών είναι δυνατόν να γραφούν στην αρχή μιας υπορουτίνας, πριν από τις άλλες εντολές της. Οι συμβολικές σταθερές και οι μεταβλητές που δηλώνονται μέσα σε μια υπορουτίνα μπορούν να χρησιμοποιηθούν μόνο μέσα στην περιοχή της υπορουτίνας, γι' αυτό και χαρακτηρίζονται ως ***τοπικές*** (***local***).

Η δήλωση των συμβολικών σταθερών γίνεται με τη δηλωτική εντολή σταθερών, που αρχίζει με την κωδική λέξη `Const`.

Η δήλωση των μεταβλητών γίνεται με τη δηλωτική εντολή μεταβλητών, που αρχίζει με την κωδική λέξη `Dim`.

Σε διαφορετικές υπορουτίνες είναι δυνατόν να χρησιμοποιούμε τοπικές συμβολικές σταθερές ή μεταβλητές με το ίδιο όνομα. Η VB δεν μπερδεύει τις τιμές τους, αφού κάθε μια από αυτές έχει τοπική εμβέλεια και δεν είναι δυνατή η αναφορά της σε άλλη υπορουτίνα.

Παράδειγμα 6-13.

```
Private Sub Form_Load()
Const ð = 3.141592

Dim Total      As Double      'Óýíïëï  } ÔïðéêÝò äçëþóåéò
Dim   i        As Integer     'Äåßêôçò } υπορουτίνας Form_Load
      :
      :
End Sub

Private Sub OK_Click()
Dim Stock      As Long        'Áðüèåìá } Τοπικές δηλώσεις
Dim   i        As Integer     'Äåßêôçò } υπορουτίνας OK_Click
      :
      :
End Sub
```

Αν και έχουν το ίδιο όνομα είναι διαφορετικές τοπικές μεταβλητές.

Σχήμα 6-1. *Τοπικές δηλώσεις*

Ο κύκλος ζωής μιας τοπικής μεταβλητής αρχίζει με την αρχικοποίησή της, κατά την εκτέλεση της δηλωτικής της εντολής μέσα στην υπορουτίνα, και τερματίζεται με τον τερματισμό της υπορουτίνας.

Κατά την αρχικοποίηση μιας μεταβλητής, αν είναι αριθμητική της αποδίδεται η τιμή 0, αν είναι δίτιμος, η τιμή `False` και αν είναι αλφαριθμητική, η τιμή "".

Σε περίπτωση που δε θέλουμε να αρχικοποιείται μια μεταβλητή κάθε φορά που εκτελείται η υπορουτίνα, τη δηλώνουμε ως ***στατική*** (***Static***), χρησιμοποιώντας την κωδική λέξη `Static` στη δηλωτική της εντολή.

Παράδειγμα 6-14.

```
Static Total  As Double     'Óýíïëï
Dim    i      As Integer    'Äåßêôçò
```

Η μεταβλητή `Total` αρχικοποιείται κατά την πρώτη εκτέλεση της υπορουτίνας, αλλά διατηρεί την τιμή της μεταξύ δύο διαδοχικών εκτελέσεων. Αντίθετα, η μεταβλητή i αρχικοποιείται κάθε φορά που εκτελείται η υπορουτίνα.

Συμβολικές σταθερές και μεταβλητές προγραμματιστικής μονάδας

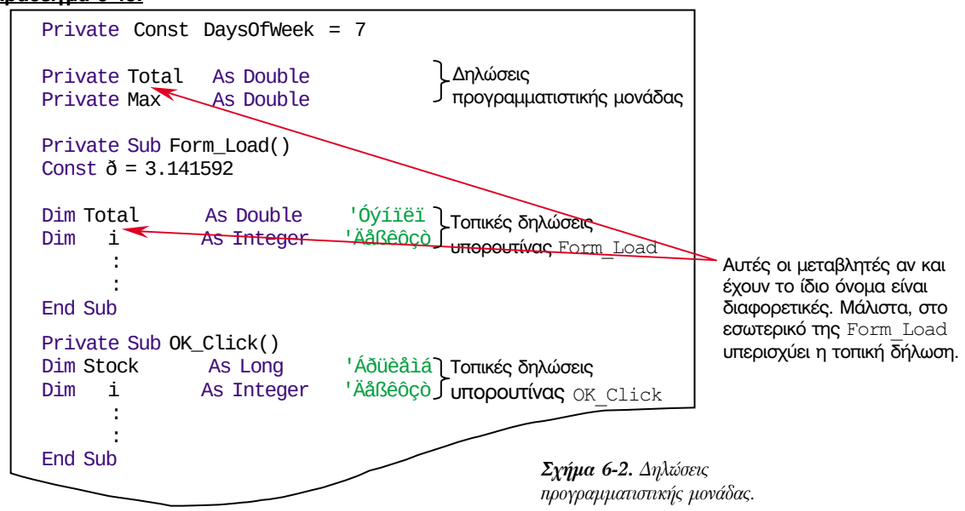
Πολλές φορές χρειάζεται να χρησιμοποιηθεί η ίδια μεταβλητή ή η ίδια συμβολική σταθερά σε περισσότερες από μια υπορουτίνες μιας προγραμματιστικής μονάδας (βασικής ή φόρμας). Σ' αυτήν την περίπτωση η δήλωσή τους πρέπει να γίνει στο ***τμήμα δηλώσεων*** (***declaration section***) της προγραμματιστικής μονάδας. Το τμήμα δηλώσεων βρίσκεται πάνω από όλες τις υπορουτίνες στην αρχή κάθε προγραμματιστικής μονάδας.

Για να μεταβούμε στο τμήμα δηλώσεων, επιλέγουμε το αντικείμενο **General** από την αριστερή πτυσσόμενη λίστα του παραθύρου κώδικα.

Για τη δήλωση των συμβολικών σταθερών, πριν από την κωδική λέξη `Const` μπορεί να τοποθετηθεί προαιρετικά η κωδική λέξη `Private`.

Η δήλωση των μεταβλητών μπορεί να αρχίζει με την κωδική λέξη `Dim` ή με την κωδική λέξη `Private`. Σ' αυτήν την περίπτωση οι δύο κωδικές λέξεις είναι ισοδύναμες. Συνήθως, για τις μεταβλητές προγραμματιστικής μονάδας χρησιμοποιείται η κωδική λέξη `Private`.

Παράδειγμα 6-15.



Σχήμα 6-2. *Δηλώσεις προγραμματιστικής μονάδας.*

Καθολικές συμβολικές σταθερές και μεταβλητές

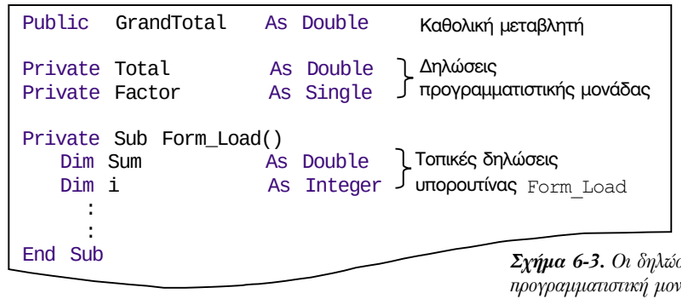
Για να είναι διαθέσιμη μια συμβολική σταθερά ή μια μεταβλητή σε όλη την έκταση του προγράμματος, πρέπει να δηλωθεί ως ***καθολική*** (***Public***).

Η δήλωση των καθολικών συμβολικών σταθερών γίνεται μόνο στο τμήμα δηλώσεων βασικών προγραμματιστικών μονάδων και ποτέ σε προγραμματιστικές μονάδες φόρμας. Η δηλωτική εντολή μιας καθολικής συμβολικής σταθεράς αρχίζει με την κωδική λέξη `Public`.

Κατά τη δήλωση των μεταβλητών καθορίζεται το όνομά τους, ο τύπος της μεταβλητής και η εμβέλειά τους.

Η δήλωση των καθολικών μεταβλητών μπορεί να γίνει στο τμήμα δηλώσεων, οποιασδήποτε προγραμματιστικής μονάδας (βασικής ή φόρμας) με μια δηλωτική εντολή μεταβλητής που αρχίζει με την κωδική λέξη `Public`.

Παράδειγμα 6-16.



Σχήμα 6-3. *Οι δηλώσεις σε μια προγραμματιστική μονάδα*

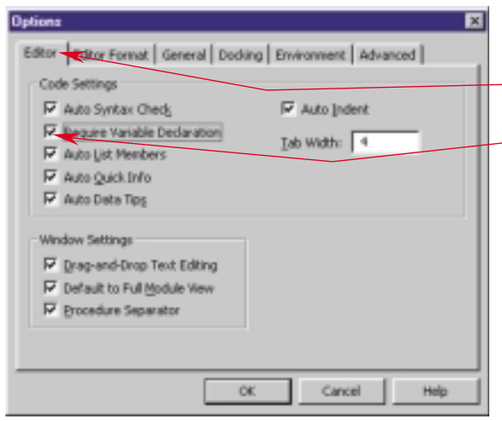
Η διαχείριση των καθολικών μεταβλητών απαιτεί ιδιαίτερη προσοχή. Στον οδηγούμενο από συμβάντα προγραμματισμό, η τιμή των καθολικών μεταβλητών μπορεί να αλλάζει σε οποιαδήποτε προγραμματιστική μονάδα, με σειρά που δεν μπορεί να προβλέψει ο προγραμματιστής. Οι καθολικές μεταβλητές χρησιμοποιούνται αν δεν υπάρχει κανένας άλλος τρόπος μεταφοράς τιμών μεταξύ φορμών. Συνήθως για λόγους ευκολίας τεκμηρίωσης, όλες οι καθολικές μεταβλητές τοποθετούνται σε μια προγραμματιστική μονάδα, στην οποία μάλιστα δίνεται και κατάλληλο όνομα π.χ. `Global`.

Υποχρεωτική δήλωση μεταβλητής ή συμβολικής σταθεράς

Η δήλωση μιας μεταβλητής πριν τη χρήση της δεν είναι υποχρεωτική. Όταν γράφουμε μια μεταβλητή χωρίς να την έχουμε ορίσει, η VB την κάνει αποδεκτή και την ορίζει ως τύπου `Variant`. Αυτή η ευκολία του αυτόματου ορισμού μπορεί να μας δημιουργήσει προβλήματα. Αν γράψουμε σε κάποιο σημείο του προγράμματος το όνομα μιας μεταβλητής λάθος (π.χ. αντί `GrandTotal` γράψουμε `GranTotal`), η VB θα εκλάβει τη λάθος γραφή ως νέα μεταβλητή. Έτσι, ενώ εμείς θα νομίζουμε ότι έχουμε μια μεταβλητή, η VB στην πραγματικότητα θα διαχειρίζεται δύο, δίνοντας απρόβλεπτα αποτελέσματα.

Αυτό το πρόβλημα μπορεί να παρακαμφθεί με τον υποχρεωτικό ορισμό κάθε μεταβλητής και κάθε συμβολικής σταθεράς. Για να μας επιβάλλεται ο υποχρεωτικός ορισμός πρέπει να τοποθετήσουμε την εντολή `Option Explicit`, στην περιοχή δηλώσεων κάθε προγραμματιστικής μονάδας πριν οποιοδήποτε ορισμό μεταβλητής ή συμβολικής σταθεράς. Τότε, στην περίπτωση που η VB αναγνωρίσει ένα νέο όνομα, το οποίο δεν έχει οριστεί, θα εμφανίσει μήνυμα λάθους.

Μπορούμε να ρυθμίσουμε το περιβάλλον εργασίας, ώστε να τοποθετεί αυτόματα την εντολή `Option Explicit` σε κάθε καινούργια προγραμματιστική μονάδα.



1. Από το μενού επιλέγουμε **Tools ¦ Options**.
2. Διαλέγουμε τον καρτελοδείκτη **Editor**.
3. Μαρκάρουμε το πλαίσιο σημείωσης **Require Variable Declaration**.

Εικόνα 6-3. *Ρύθμιση για αυτόματη εισαγωγή της εντολής* `Option Explicit`.

Πράξεις και παραστάσεις

Στη VB είναι δυνατόν να έχουμε πράξεις μεταξύ σταθερών και μεταβλητών ενός συγκεκριμένου τύπου (για παράδειγμα, πράξεις μεταξύ αριθμητικών μεταβλητών και σταθερών ή πράξεις μεταξύ αλφαριθμητικών σταθερών και μεταβλητών), αλλά και πράξεις μεταξύ σταθερών και μεταβλητών διαφορετικών τύπων (π.χ. πράξεις μεταξύ αριθμητικών μεταβλητών και σταθερών και ημερομηνιών). Με μεταβλητές, σταθερές και πράξεις σχηματίζονται παραστάσεις.

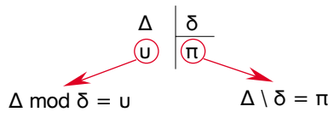
Αριθμητικές πράξεις και παραστάσεις

Οι αριθμητικές πράξεις που μπορεί να εκτελέσει η VB είναι οι τέσσερεις βασικές πράξεις της πρόσθεσης, της αφαίρεσης, του πολλαπλασιαμού και της διαίρεσης, η ύψωση στη δύναμη και άλλες δύο πράξεις που έχουν να κάνουν με τη διαίρεση μεταξύ ακεραίων αριθμών, η ***ακέραια διαίρεση*** και η πράξη ***modulo***:

Πράξη	Σύμβολο
Πρόσθεση	+
Αφαίρεση	-
Πολλαπλασιασμός	*
Διαίρεση	/
Ακεραία διαίρεση	\
Modulo	mod
Ύψωση σε δύναμη	^

Παρατηρήστε, ότι το σύμβολο του πολλαπλασιασμού είναι ο αστερίσκος (*) και όχι το (x) ή η τελεία (.). Ο λόγος είναι η αποφυγή σύγχυσης με το γράμμα x, που υπάρχει στα ονόματα των μεταβλητών και των αντικειμένων, και την τελεία, που είναι το διαχωριστικό σύμβολο μεταξύ αντικειμένου και ιδιότητας ή αντικειμένου και μεθόδου.

Η ακέραια διαίρεση (σύμβολό της η ανάποδη κάθετος \) γίνεται συνήθως μεταξύ σταθερών και μεταβλητών ακέραιου τύπου. Το αποτέλεσμά της είναι το ***ακέραιο τμήμα του πηλίκου*** που προκύπτει. Η πράξη modulo είναι συμπληρωματική της ακέραιας διαίρεσης, αφού μας δίνει το ***υπόλοιπο*** της διαίρεσης. Δηλαδή:



Η ακέραια διαίρεση και η πράξη modulo είναι αρκετά χρήσιμες και εφαρμόζονται κυρίως στις μετατροπές συμμιγών αριθμών.

Παράδειγμα 6-17.

Έστω ότι η μεταβλητή `Minutes` παριστάνει χρόνο σε λεπτά και ότι η τιμή της είναι 124. Για τη μετατροπή του χρόνου σε ώρες και λεπτά πρέπει να γίνουν οι πράξεις:

```
Minutes \ 60          με αποτέλεσμα 2 ώρες
Minutes mod 60              με αποτέλεσμα 4 λεπτά
```

Οι αριθμητικές παραστάσεις στη VB σχηματίζονται από αριθμητικές σταθερές και αριθμητικές μεταβλητές που μεταξύ τους υπάρχουν τα σύμβολα των αριθμητικών πράξεων. Πολλές φορές, για να είναι σαφής η σειρά με την οποία θα εκτελεστούν οι πράξεις, τοποθετούνται κατάλληλα και παρενθέσεις.

Παραδείγματα 6-18.

Για να υπολογιστούν σωστά οι τιμές των αλγεβρικών παραστάσεων που βρίσκονται στην αριστερή στήλη, πρέπει να τις κωδικοποιήσουμε με τον τρόπο που υποδεικνύεται στη δεξιά στήλη.

Αλγεβρική παράσταση	Παράσταση στη VB
πr^2	`3.141592 * r^2 * õ`
$\frac{(b \quad B)}{2}$	`(B1 + B2) * õ/2`
$3 \quad ^{2} \quad 4 \quad 9$	`3*X^2 + 4*X + 9`
$\frac{1}{2}m \quad ^{2}$	`1/2 * m * õ^2`
$k\frac{mM}{r}$	`k * Maza1 * Maza2 / r^2`
$\sqrt{x}$	`X^0.5`
$\frac{\sqrt{\quad ^{2} \quad 4}}{2}$	`(-b + (b^2 - 4*a*c)^0.5) / (2 * a)`
$\sqrt[3]{\quad ^{2}}$	`x^(2/3)`
$((a \quad b) \quad)$	`(a + b)^N^M`
$\frac{1}{\sqrt{(x \quad 1)^3}}$	`(x + 1)^(-3/2)`

Παραδείγματα 6-19.

Κατά την κωδικοποίηση αλγεβρικών παραστάσεων απαιτείται ιδιαίτερη προσοχή, επειδή συχνά γίνονται λάθη από αβλεψία.

Αλγεβρική παράσταση	Λανθασμένη κωδικοποίηση
$(a \quad b)(a \quad b)$	`(A+B)(A-B)` Λείπει το σύμβολο του πολλαπλασιασμού.
$p \quad q$	`P.Q` Λάθος σύμβολο πολλαπλασιασμού.
$\frac{k}{5[a(\quad z)^2 \quad (\quad)^2}$	`(K+L) / (5*(A*(B-Z)^2+(C-W)^2)` Λείπει μια παρένθεση.
$\{ \quad ^{2}[\quad ^{2} \quad (1 \quad)^2]\}$	`Á*{×^2*[õ^2-(1+Æ)^2]}` Χρήση αγκυλών και αγκίστρων.
$\frac{\sqrt{3v}}{4}$	`(3/4*ð*V)^0.5` Υπολογίζεται άλλη παράσταση.

Πράξεις και παραστάσεις συμβολοσειρών

Η μοναδική πράξη μεταξύ αλφαριθμητικών σταθερών και μεταβλητών είναι η πράξη της ***συνένωσης*** (***concatanation***). Με τη συνένωση δύο όρων προκύπτει μια τιμή που αποτελείται από τους χαρακτήρες του πρώτου όρου ακολουθούμενους από τους χαρακτήρες του δεύτερου όρου. Το σύμβολο της πράξης της συνένωσης είναι το (&) ή το σύμβολο (+).

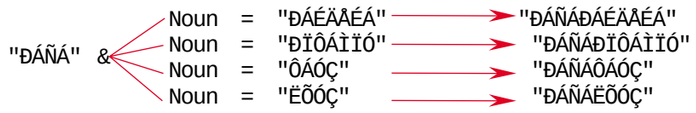
Παραδείγματα 6-20.

Πράξη	Αποτέλεσμα
"ÊÁËÇ" & "ÌÅÑÁ"	"ÊÁËÇÌÅÑÁ"
"" & "Êåíü"	"Êåíü"
"ÓÔÑÁÔÏ" & "ÓÖÁÉÑÁ"	"ÓÑÁÔÏÓÖÁÉÑÁ"
"12" & "3"	"123"
"17.5" + "19.5"	"17.519.5"Ô

Από τα δύο τελευταία παραδείγματα φαίνεται καθαρά, ότι η πράξη της συνένωσης δεν έχει τίποτα το κοινό με την πράξη της πρόσθεσης, παρ' όλο που και στις δύο χρησιμοποιείται το ίδιο σύμβολο.

Παράδειγμα 6-21.

Συνδυάζοντας την συμβολοσειρά "ÐÁÑÁ" με διάφορες τιμές της μεταβλητής Noun σχηματίζουμε παράγωγα:



Πράξεις μεταξύ ημερομηνιών

Τις ημερομηνίες μπορούμε να τις διαχειριστούμε σαν αριθμούς. Αν αφαιρέσουμε δύο ημερομηνίες, θα βρούμε πόσες μέρες μεσολαβούν μεταξύ τους, αν προσθέσουμε σε μια ημερομηνία έναν αριθμό, θα βρούμε μια ημερομηνία κάποιες μέρες μετά.

Παραδείγματα 6-22.

α) Στη μεταβλητή `ThisDay` είναι καταχωρισμένη η σημερινή ημερομηνία, έστω #28/10/2002#. Στη συμβολική σταθερά `StartDate` είναι καταχωρισμένη μια ημερομηνία που τη θεωρούμε σαν εναρκτήρια μιας σειράς γεγονότων, η #1/1/2000#. Η διαφορά:

```
ThisDay - StartDate
```

θα δώσει το πλήθος των ημερών που μεσολαβούν μεταξύ των δύο ημερομηνιών, δηλαδή 1031.

β) Στη μεταβλητή `ThisDay` είναι καταχωρισμένη η σημερινή ημερομηνία. Η VB με την παράσταση:

```
ThisDay + 1
```

υπολογίζει την επόμενη μέρα, λαμβάνοντας υπ' όψιν της όλες τις ιδιομορφίες του ημερολογίου μας (μήνες με 30 ή 31 μέρες, δίσεκτα έτη, αλλαγή χρονιάς μετά την 31η Δεκεμβρίου) και μας απαλλάσσει από πολύπλοκους υπολογισμούς.

Οι ημερομηνίες είναι δυνατόν να δεχτούν και αριθμητικές τιμές. Σ' αυτήν την περίπτωση το ακέραιο μέρος μετατρέπεται σε ημέρα και το δεκαδικό μέρος σε ώρα. Η τιμή 0 αντιστοιχεί στην 30/12/1899, άρα οι θετικές τιμές είναι μετέπειτα ημερομηνίες και οι αρνητικές τιμές είναι προγενέστερες ημερομηνίες. Τα μεσάνυχτα έχουν δεκαδικό μέρος μηδέν και το μεσημέρι τιμή 0.5.

Εντολή Εκχώρησης

Ήδη έχουμε αναφέρει, ότι με την εντολή εκχώρησης δίνονται τιμές στις ιδιότητες αντικειμένων. Η ίδια εντολή μπορεί να χρησιμοποιηθεί για να δοθούν τιμές σε μεταβλητές. Οι τιμές μπορεί να προέρχονται από σταθερές, από μεταβλητές, από ιδιότητες αντικειμένων, από παραστάσεις. Άρα, μια άλλη μορφή της εντολής είναι:

μεταβλητή = παράσταση

Με την εκτέλεση της εντολής χάνεται η προηγούμενη τιμή της μεταβλητής του αριστερού μέλους και γίνεται αντικατάστασή της με την τιμή του δεξιού μέλους της.

Παραδείγματα 6-23.

α) Έστω ότι θέλουμε να εκχωρήσουμε σε μεταβλητή την τιμή του πρώτου παραδείγματος 6-18. Πρέπει να προηγηθούν οι δηλώσεις:

```
Const ð = 3.141592
Dim CylinderVolume   As Double
Dim r                As Double
Dim õ                As Double
```

και να ακολουθήσει η εντολή εκχώρησης:

```
CylinderVolume = ð * r^2 * õ
```

β) Έστω ότι θέλουμε να εκχωρήσουμε σε μεταβλητή την τιμή του τέταρτου παραδείγματος 6-18. Πρέπει να προηγηθούν οι δηλώσεις:

```
Dim KineticEnergy   As Double
Dim m               As Double
Dim õ               As Double
```

και να ακολουθήσει η εντολή εκχώρησης:

```
KineticEnergy = 1/2 * m * õ^2
```

γ) Έστω ότι θέλουμε να εκχωρήσουμε σε μεταβλητή την τιμή του πρώτου παραδείγματος 6-20. Πρέπει να προηγηθεί η δήλωση:

```
Dim ComplexNoun      As String
```

και να ακολουθήσει η εντολή εκχώρησης:

```
ComplexNoun = "ÊÁËÇ" & "ÌÅÑÁ"
```

δ) Έστω ότι θέλουμε να εκχωρήσουμε σε μεταβλητές τις τιμές του τρίτου παραδείγματος 6-21. Πρέπει να προηγηθούν οι δηλώσεις:

```
Dim ComplexNoun      As String
Dim Noun             As String
```

και να ακολουθήσουν οι εντολές εκχώρησης:

```
Noun = "ÔÁÓÇ"
ComplexNoun = "ÐÁÑÁ" & Noun
```

ε) Έστω ότι θέλουμε να εκχωρήσουμε σε μεταβλητή την τιμή του δεύτερου παραδείγματος 6-22. Πρέπει να προηγηθούν οι δηλώσεις:

```
Dim ThisDay          As Date
Dim NextDay          As Date
```

και να ακολουθήσει η εντολή εκχώρησης:

```
NextDay = ThisDay + 1
```

Το σύμβολο ίσον (=) μεταξύ των δύο μελών χρησιμοποιείται ως σύμβολο εκχώρησης και όχι ως σύμβολο ισότητας. Γι' αυτό και η εντολή δεν πρέπει να θεωρείται ποτέ ως έκφραση μαθηματικής εξίσωσης. Έτσι η εντολή:

```
counter = counter + 1
```

εκχωρεί ως καινούργια τιμή στη μεταβλητή counter την προηγούμενη της αυξημένη κατά 1. Δηλαδή, αν η προηγούμενη τιμή της μεταβλητής ήταν 10, η νέα της τιμή θα είναι 10+1=11.

Παράδειγμα 6-24.

Με τις μεταβλητές του τύπου `Variant` είναι δυνατόν να κάνουμε πράξεις με δεδομένα διαφορετικών τύπων, χωρίς να απαιτείται πάντα η μετατροπή τους σε έναν τύπο:

```
Dim TotalTime                'ÄÞëùóç Variant

TotalTime = "20"             'ÐåñéÝ÷åé ôç óõìâïëïóåéñÜ "20"
TotalTime = TotalTime + 15   'ÐåñéÝ÷åé ôïí áñéèìü 35
TotalTime = "Óõíïëéêüò ÷ñüíïò " & TotalTime & " ëåðôÜ."
```

Πρέπει να τονιστεί, ότι δεν επιτρέπεται σε μια αριθμητική μεταβλητή να αντιστοιχίσουμε την τιμή μιας αλφαριθμητικής ποσότητας αλλά ούτε και σε μια αλφαριθμητική μεταβλητή να αντιστοιχίσουμε την τιμή μιας αριθμητικής ποσότητας. Αν προσπαθήσουμε να κάνουμε κάτι τέτοιο, η VB θα το αντιχνεύσει και θα εμφανίσει το διαγνωστικό μήνυμα λάθους **"Type mismatch".**

Εφαρμογές

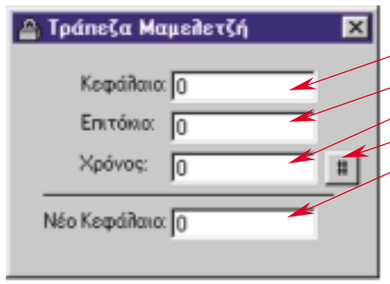
Υπολογισμός τόκου και νέου κεφαλαίου

Οι τράπεζες για τα ποσά που κατατίθενται υπολογίζουν τόκους ανάλογα με το προσυμφωνημένο επιτόκιο και το χρόνο κατάθεσης. Αν,

- Κ είναι το κεφάλαιο που έχει κατατεθεί
- Ε το ετήσιο επιτόκιο
- Χ ο χρόνος κατάθεσης σε μέρες

ο τόκος είναι $T = \frac{K \cdot E \cdot X}{360 \cdot 100}$, αφού το οικονομικό έτος έχει 360 μέρες.

Θα δημιουργήσουμε πρόγραμμα που θα δέχεται το κεφάλαιο, το επιτόκιο και το χρόνο κατάθεσης και θα εμφανίζει το νέο κεφάλαιο. Αρχικά, σχεδιάζουμε τη φόρμα για το παράθυρο της εφαρμογής.



Name	Text	Enabled	TabIndex
Capital	0	True	0
CompInterest	0	True	1
Duration	0	True	2
Evaluate	0	True	3
NewCapital	0	False	0

Εικόνα 6-4. *Η φόρμα και οι ιδιότητες των αντικειμένων*

Καλό είναι να δίνεται πάντα αρχική τιμή στην ιδιότητα **Text** των πλαισίων κειμένου. Στην περίπτωση που αντιπροσωπεύουν αριθμητικές ποσότητες, δίνεται συνήθως η τιμή 0. Στην περίπτωση που αντιπροσωπεύουν αλφαριθμητικές ποσότητες, δίνεται συνήθως η τιμή "".

Ο χρήστης θα ενημερώνει τα πλαίσια κειμένου `Capital`, `CompInterest`, `Duration` και θα κάνει κλικ στο πλήκτρο διαταγής `Evaluate` για τον υπολογισμό του τόκου και του νέου κεφαλαίου. Ο κώδικας του προγράμματος είναι:

```
Option Explicit
Private Sub Evaluate_Click()
    Dim Interest    As Double
    Interest = (Capital * CompInterest * Duration) / 36000
    NewCapital = Capital + Interest
End Sub
```

Η αναφορά της ιδιότητας **Text** είναι προαιρετική, δηλαδή οι εκφράσεις `Capital.Text` και `Capital` είναι ισοδύναμες, η δεύτερη μάλιστα προτιμάται μιας και είναι πιο σύντομη. Γι' αυτό από εδώ και πέρα δε θα αναφέρουμε αυτήν την ιδιότητα μετά το αντικείμενο.

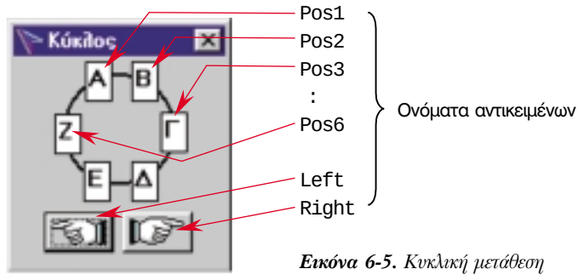
Κυκλική μετάθεση ατόμων γύρω από τραπέζι

Σε ένα κυκλικό τραπέζι κάθονται έξι άτομα. Κάποιες χρονικές στιγμές ζητείται από τα άτομα να σηκωθούν από το τραπέζι και να αλλάξουν θέση καταλαμβάνοντας τη θέση του διπλανού τους. Η κίνηση γίνεται ή αριστερόστροφα ή δεξιόστροφα ανάλογα με την εντολή που θα δοθεί. Να γίνει προσομοίωση της κίνησης.

Αρχικά δημιουργούμε τη φόρμα διεπαφής:

1. Με το εργαλείο γεωμετρικό σχήμα, σχεδιάζουμε ένα σχήμα και του αλλάζουμε την ιδιότητα **Shape** (μορφή) σε ***3-Circle*** (κύκλος) και την ιδιότητα **BorderWidth** (πάχος ορίου, δηλαδή της περιφερείας του κύκλου) σε 2.

2. Σχεδιάζουμε έξι πλαίσια κειμένου, με ονόματα `Pos1`, `Pos2`, ... , `Pos6` και τα τοποθετούμε πάνω στον κύκλο κατά τη φορά των δεικτών του ρολογιού. Στην ιδιότητα **Text** κάθε ενός δίνουμε την τιμή A, B, ... , Z αντίστοιχα, όπως φαίνεται στην εικόνα 6.5.
3. Σχεδιάζουμε και δύο πλήκτρα διαταγής με ονόματα `Left` και `Right` που θα εντολοδοτούν την κίνηση. Διαγράφουμε την ιδιότητα τους **Caption** και ενεργοποιούμε την ιδιότητα **Style** σε ***1-Graphical***. Στη συνέχεια από την ιδιότητα **Picture** επιλέγουμε από τον φάκελο `...\Common\Graphics\Icons\Arrows` τα εικονίδια που φαίνονται στην εικόνα.



Εικόνα 6-5. *Κυκλική μετάθεση*

Στη συνέχεια σχεδιάζουμε και γράφουμε τον κώδικα. Τη στιγμή που θα γίνει κλικ στο πλήκτρο `Left` για μετάθεση στα αριστερά (φορά δεικτών του ρολογιού):

1. Στο "πρώτο" πλαίσιο κειμένου στον κύκλο (`Pos1`), πρέπει να δοθεί η τιμή του "τελευταίου" (δηλαδή η τιμή Z του `Pos6`). Αυτή η εκχώρηση πρέπει να γίνει, χωρίς να χαθεί η τιμή του `Pos1`. Επομένως, απαιτείται να προηγηθεί προσωρινή αποθήκευση του `Pos1` σε μια μεταβλητή, έστω την `PosTemp`.
2. Στο "τελευταίο" πλαίσιο κειμένου στον κύκλο (`Pos6`), πρέπει να δοθεί η τιμή του "προτελευταίου" (δηλαδή η τιμή Å του `Pos5`). Αυτή η εκχώρηση δεν απαιτεί προστασία του περιεχομένου του `Pos6`, επειδή ήδη έχει μεταφερθεί η παλιά του τιμή στο `Pos1`.

:
:

6. Κλείνοντας τον κύκλο πρέπει να δοθεί και τιμή στο "δεύτερο" πλαίσιο κειμένου `Pos2`. Αυτή η ενημέρωση δεν πρέπει να γίνει απευθείας από το προηγούμενό του (`Pos1`), όπως συνέβαινε με όλα τα άλλα πλαίσια, αλλά από την `PosTemp` που έχει συγκρατήσει την προηγούμενη τιμή του `Pos1`.

Το πρόγραμμα για δεξιόστροφη και αριστερόστροφη μετάθεση είναι το:

```
Option Explicit
Private PosTemp     As String
Private Sub Left_Click()  'ÖïñÜ äåéêôþí ôïõ ñïëïãéïý
    PosTemp = Pos1        'ÓõãêñÜôçóå ôçí áñ÷éêÞ ôéìÞ ôïõ Pos1
    Pos1 = Pos6
    Pos6 = Pos5
    Pos5 = Pos4
    Pos4 = Pos3
    Pos3 = Pos2
    Pos2 = PosTemp        'Ç áñ÷éêÞ ôéìÞ ôïõ Pos1 óôï Pos2
End Sub
Private Sub Right_Click() 'Áíôßèåôç öïñÜ ôùí äåéêôþí ôïõ ñïëïãéïý
    PosTemp = Pos1        'ÓõãêñÜôçóå ôçí áñ÷éêÞ ôéìÞ ôïõ Pos1
    Pos1 = Pos2
    Pos2 = Pos3
    Pos3 = Pos4
    Pos4 = Pos5
    Pos5 = Pos6
    Pos6 = PosTemp        'Ç áñ÷éêÞ ôéìÞ ôïõ Pos1 óôï Pos6
End Sub
```

Ανακεφαλαίωση

Σε ένα πρόγραμμα μπορούμε να χρησιμοποιούμε σταθερές, δηλαδή αριθμούς, λεκτικά, σειρές συμβόλων, ημερομηνίες που δεν αλλάζουν κατά τη διάρκεια εκτέλεσης του. Τις σταθερές μπορούμε να τις παραστήσουμε με συμβολικό τρόπο, ώστε το πρόγραμμα να είναι πιο ευέλικτο σε μελλοντικές αλλαγές και ευκολότερα αναγνώσιμο. Η VB, για πολλές ποσότητες και ιδιαίτερα τιμές ιδιοτήτων, έχει ορίσει εσωτερικές σταθερές.

Για να παραστήσουμε με συμβολικό τρόπο διάφορα μεταβλητά μεγέθη χρησιμοποιούμε μεταβλητές. Κατά τη δήλωση των μεταβλητών καθορίζεται το όνομά τους, ο τύπος τους και η εμβέλειά τους. Η δήλωση των τοπικών μεταβλητών γίνεται με εντολές `Dim` και `Static`, η δήλωση μεταβλητών προγραμματιστικής μονάδας γίνεται με εντολές `Private` και η δήλωση καθολικών μεταβλητών με εντολές `Public`.

Στη VB είναι δυνατόν να έχουμε πράξεις μεταξύ σταθερών και μεταβλητών ενός συγκεκριμένου τύπου, αλλά και πράξεις μεταξύ σταθερών και μεταβλητών διαφορετικών τύπων. Με μεταβλητές, σταθερές και πράξεις σχηματίζονται παραστάσεις.

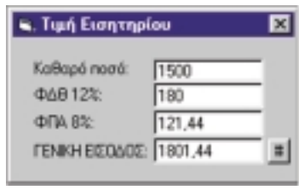
Η εντολή εκχώρησης μπορεί να χρησιμοποιηθεί για να δοθούν τιμές σε μεταβλητές ή ιδιότητες αντικειμένων. Οι τιμές μπορεί να προέρχονται από σταθερές, από μεταβλητές, από ιδιότητες αντικειμένων, από παραστάσεις. Οι γενικές μορφές της εντολής είναι:

αντικείμενο.ιδιότητα = παράσταση
μεταβλητή = παράσταση

Εργαστηριακές Ασκήσεις

1. Υλοποιήστε το πρόγραμμα που δίνεται ως εφαρμογή για τον υπολογισμό του τόκου και του νέου κεφαλαίου.
   Προσοχή !!! Σε αυτήν την άσκηση, καθώς και σε όλες τις ασκήσεις που θα ακολουθήσουν, να δίνεται αρχική τιμή στην ιδιότητα **`Text`**. Διαφορετικά, θα εμφανιστούν λάθη κατά την εκτέλεση του προγράμματος.
2. Στα εισιτήρια των θεάτρων και των κινηματογράφων, γίνεται ανάλυση, που αναγράφεται ο καταμερισμός των ποσών σε δικαιώματα αίθουσας και φόρους. Οι φόροι είναι:
   - ο ***φόρος δημοσίων θεαμάτων*** (***ΦΔΘ***) - 12% επί του καθαρού ποσού που εισπράττει ο αιθουσάρχης- ,
   - ο ***φόρος προστιθέμενης αξίας*** (***ΦΠΑ***) - 8% επί του αθροίσματος του καθαρού ποσού του αιθουσάρχη και του ΦΔΘ.

   Δημιουργήστε πρόγραμμα, στο οποίο να δίνεται η γενική τιμή εισόδου και να επιμερίζονται τα δικαιώματα και οι φόροι.



Εικόνα 6-6. *Δείγμα της οθόνης υπολογισμού. Από το καθαρό ποσό προκύπτουν τα υπόλοιπα.*

3. Η περίοδος ταλάντωσης ενός μαθηματικού εκκρεμούς δίνεται από τον τύπο:

$$T \quad 2 \sqrt{\frac{L}{g}}$$

   με L το μήκος του εκκρεμούς και g την επιτάχυνση της βαρύτητας. Δημιουργήστε πρόγραμμα, που να υπολογίζει την περίοδο.
4. Η θερμότητα Q, που εκλύεται στα άκρα ενός αγωγού (θερμότητα Joule), αντίστασης μέσα σε χρόνο t, δίνεται από τον τύπο:

$$Q \quad i^2 \quad R \quad t$$

   με i το ρεύμα που διαρρέει τον αγωγό. Δημιουργήστε πρόγραμμα που να υπολογίζει το Q από τα i, R και t.

5. Δημιουργήστε πρόγραμμα που να υπολογίζει τον όγκο, το εμβαδόν και την περίμετρο του μέγιστου κύκλου σφαίρας, όταν δίνεται η ακτίνα.
6. Υλοποιήστε το πρόγραμμα που δίνεται ως εφαρμογή στην κυκλική μετάθεση ατόμων γύρω από το τραπέζι.
   Κάντε δοκιμή να τρέξετε το πρόγραμμα χωρίς τη χρήση της βοηθητικής μεταβλητής. Παρατηρήστε τι θα συμβεί.

Σημειώσεις:

Μαθημα 7
Συναρτήσεις

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να χρησιμοποιούν αριθμητικές και αλφαριθμητικές συναρτήσεις.
- Να χρησιμοποιούν τις συναρτήσεις μετατροπής τύπων δεδομένων.
- Να χρησιμοποιούν τις συναρτήσεις χρόνου.

Οι τύποι και οι σχέσεις που χρησιμοποιούνται στην επίλυση τεχνολογικών προβλημάτων, προβλημάτων της Φυσικής και προβλημάτων των Μαθηματικών, περιέχουν συνήθως ***τριγωνομετρικές συναρτήσεις*** (ημίτονο, συνημίτονο, εφαπτομένη κ.ά.) και ***αλγεβρικές συναρτήσεις*** (λογάριθμος, απόλυτη τιμή κ.ά.). Για την πιο εύκολη κωδικοποίηση των τύπων και των σχέσεων που περιέχουν συναρτήσεις, η VB μας παρέχει τη δυνατότητα της συμβολικής γραφής των πιο κοινών ***μαθηματικών συναρτήσεων***. Εκτός όμως από μαθηματικές συναρτήσεις, η VB αναγνωρίζει και άλλου τύπου συναρτήσεις, ***ενσωματωμένες συναρτήσεις*** ή ***συναρτήσεις βιβλιοθήκης***, για τη διαχείριση όλων των τύπων δεδομένων που υποστηρίζει. Συγκεκριμένα, η VB διαθέτει:

- Μαθηματικές συναρτήσεις.
- Αλφαριθμητικές συναρτήσεις.
- Συναρτήσεις μετατροπής τύπων δεδομένων (π.χ. από δεδομένα ακεραίου τύπου σε δεδομένα διπλής ακριβείας).
- Συναρτήσεις ελέγχου τύπου και τιμής δεδομένων.
- Συναρτήσεις ημερομηνιών.

Οι συναρτήσεις έχουν μετά το όνομά τους και μέσα σε παρένθεση μια ή περισσότερες μεταβλητές, τις οποίες στο εξής θα ονομάζουμε ***παραμέτρους***. Στις παραμέτρους μπορούμε να γράψουμε σταθερές, μεταβλητές, παραστάσεις. Ανάλογα με τη συνάρτηση, οι τιμές των παραμέτρων καθορίζουν την τιμή της.

Στον προγραμματισμό ο όρος μεταβλητή έχει καθιερωθεί να υπονοεί μεταβλητή προγράμματος. Για να μη δημιουργείται σύγχυση χρησιμοποιούμε τον όρο παράμετρος συνάρτησης αντί του όρου μεταβλητή συνάρτησης.

Μαθηματικές Συναρτήσεις

Οι μαθηματικές συναρτήσεις της VB δέχονται στις παραμέτρους τους αριθμητικές τιμές οποιουδήποτε τύπου (μικρούς ακέραιους, ακέραιους, μεγάλους ακέραιους, πραγματικούς απλής και διπλής ακρίβειας) και είναι:

Συνάρτηση	Παράσταση	Λειτουργία
`Abs(x)`	x	Απόλυτη τιμή του x
`Átn(x)`	τοξεφx	Τόξο εφαπτομένης x
`Cos(x)`	συνx	Συνημίτονο x
`Åxp(x)`	e^x	Δύναμη με βάση τον αριθμό e
`Fix(x)`		Αποκοπή των ψηφίων μετά την υποδιαστολή
`Ént(x)`	[x]	Ακέραιο μέρος του x
`Log(x)`	lηx	Νεπέριος λογάριθμος του x
`Rnd(x)`		Τυχαίος αριθμός από το διάστημα (0,1)
`Sgn(x)`		Πρόσημο του x
`Sin(x)`	ημx	Ημίτονο x
`Sqr(x)`	$\sqrt{x}$	Τετραγωνική ρίζα του x
`ðan(x)`	εφx	Εφαπτομένη x

Πίνακας 7-1. *Οι ενσωματωμένες μαθηματικές συναρτήσεις.*

Η τετραγωνική ρίζα `Sqr(x)`

Η τιμή της παραμέτρου της τετραγωνικής ρίζας απαγορεύεται να είναι αρνητικός αριθμός. Σε διαφορετική περίπτωση διακόπτεται η εκτέλεση του προγράμματος και στην οθόνη εμφανίζεται το μήνυμα "**Illegal function call**". Η τετραγωνική ρίζα μιας ποσότητας x μπορεί να υπολογιστεί και με την ύψωσή της στη δύναμη 0,5.

Παραδείγματα 7-1.

α) Η εντολή που υπολογίζει την υποτείνουσα `Hypotenuse` ενός ορθογωνίου τριγώνου από τις κάθετες πλευρές του `Beta` και `Gamma` είναι η:

```
Hypotenuse = Sqr(Beta^2 + Gamma^2)
```

β) Η διακρίνουσα `Descriminant` του τριωνύμου Ay^2+By+C υπολογίζεται από την:

```
Descriminant = Sqr(B^2 - 4*A*C)
```

γ) Ο υπολογισμός της έντασης του ρεύματος `Amperes`, από την ισχύ `Watts` και την αντίσταση `Ohms` του αγωγού γίνεται με την:

```
Amperes = Sqr(Watts/Ohms)
```

Η απόλυτη τιμή Abs(x)

Παραδείγματα 7-2.

α) Η παράσταση:

$$z \quad |x \quad y| \quad |x \quad y|$$

κωδικοποιείται στη VB:

```
z = Abs(x + y) + Abs(x - y)
```

β) Η παράσταση:

$$z \quad \left\| a \quad b^2 \right| \quad \sqrt{|k \quad |}$$

κωδικοποιείται:

```
z = Abs(Abs(a - b^2) - Sqr(Abs(k-ë)) )
```

Από ό,τι φαίνεται, στην παράμετρο μιας συνάρτησης είναι δυνατόν να υπάρχει παράσταση που περιέχει συναρτήσεις. Κάτι τέτοιο όμως, όπου είναι δυνατόν, προσπαθούμε να το αποφεύγουμε διασπώντας την παράσταση σε απλούστερες. Κι αυτό γιατί στις πολύπλοκες παραστάσεις μπορεί να κάνουμε λάθη, ενώ ταυτόχρονα δημιουργούνται και δυσκολίες στην κατανόηση της κωδικοποίησης από έναν τρίτο.

Το πρόσημο Sgn(x)

Η τιμή αυτής της συνάρτησης είναι:

Sgn(x) → 1 αν x > 0
Sgn(x) → 0 αν x = 0
Sgn(x) → -1 αν x < 0

Παράδειγμα 7-3.

Η παράσταση:

```
Max = (Sgn(A-B)*(A-B) + A+B)/2
```

παίρνει ως τιμή, το μέγιστο μεταξύ των `A` και `B`. Πράγματι:

	A>B	Sgn(A-B) = 1	(A-B+A+B)/2 = 2A/2 = A
αν	A=B	Sgn(A-B) = 0 και η παράσταση:	(A+B)/2 = 2A/2 = A
	A<B	Sgn(A-B) = -1	(B-A+A+B)/2 = 2B/2 = B

Οι συναρτήσεις Int(x) και Fix(x)

Οι συναρτήσεις αυτές χρησιμοποιούνται για την εύρεση ενός ακέραιου αριθμού, που βρίσκεται κοντά στην τιμή της ποσότητας x. Οι διαφορές τους είναι λεπτές, γι' αυτό χρειάζεται να δοθεί ιδιαίτερη προσοχή κατά τη μελέτη τους.

Η τιμή της συνάρτησης `Int` είναι ο μεγαλύτερος ακέραιος που δεν ξεπερνά την τιμή του `x`. Αυτός συμβολίζεται στα μαθηματικά με [x] και ικανοποιεί τη σχέση:

$$[x] \quad x \quad [x] + 1$$

Παράδειγμα 7-4.

Με αυτή τη συνάρτηση μπορούμε να στρογγυλοποίησουμε έναν αριθμό σε οποιοδήποτε πλήθος ψηφίων του, είτε σε δεκαδικό ψηφίο είτε σε ακέραιο.

Αν `X=32.9854` και `N=2`, η παράσταση:

```
Y = Int( ×*10^N + 0.5 )/10^N
```

καταχωρίζει στο Õ την τιμή `32.99`.

Πράγματι:

```
32,9854 * 102 = 3298,54
3298,54 + 0.5 = 3299,04
[3299,04] = 3299
3299 / 10^2 = 32,99
```

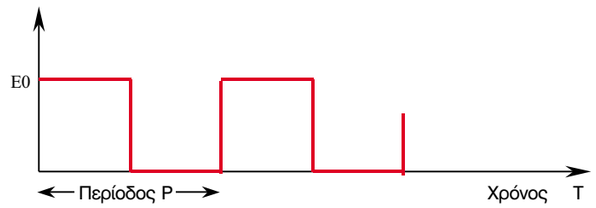
Αν `N=-1`, το Y γίνεται ίσο με `30`.

Δηλαδή η τιμή του N καθορίζει το πλήθος των ψηφίων που θεωρούμε σημαντικά. Πάντως, η παράσταση αυτή δε δίνει τη σωστή στρογγυλοποίηση αρνητικού αριθμού, αν το αποβαλλόμενο τμήμα είναι 5. Η στρογγυλοποίηση σε ένα ψηφίο του -27.14 δίνει -27.1.

Παράδειγμα 7-5.

Η σειρά των ορθογώνιων παλμών μπορεί να περιγραφεί από την παράσταση:

```
E = EÏ - EÏ*Int(2*T/P) + 2*EÏ*Int(T/P)
```



Σχήμα 7-1. Τετραγωνική κυματομορφή

Πράγματι:

για 0 < T < P/2 και οι δύο συναρτήσεις `Int` δίνουν αποτέλεσμα ίσο με 0 και E = E0
για P/2 < T < P η πρώτη γίνεται 1 και E = E0 - E0 = 0
για P < T < 3P/2 η πρώτη γίνεται -2E0 και η δεύτερη +2E0, άρα E = E0, κ.ο.κ.

Η τιμή της συνάρτησης `Fix` είναι ο ακέραιος, ο οποίος προκύπτει από την τιμή του x μετά την αποκοπή των ψηφίων, που βρίσκονται στα δεξιά της υποδιαστολής. Η διαφορά μεταξύ `Fix` και `Int` είναι ότι η `Fix` δε δίνει τον πλησιέστερο ακέραιο, αν το x είναι αρνητικό.

Παράδειγμα 7-6.

Η παράσταση:

```
Y = Fix( X*10^N + Sgn(X)*0.5 )/10^N
```

στρογγυλοποιεί σωστά τόσο τους θετικούς όσο και τους αρνητικούς αριθμούς.

Οι τριγωνομετρικές συναρτήσεις Ημίτονο Sin(x), συνημίτονο Cos(x), εφαπτομένη Tan(x)

Από τις τριγωνομετρικές συναρτήσεις, μόνο οι τρεις θεμελιώδεις περιέχονται στη βιβλιοθήκη των συναρτήσεων. Από αυτές όμως μπορούμε να υπολογίσουμε όλους τους τριγωνομετρικούς αριθμούς του τόξου x, χρησιμοποιώντας τους γνωστούς τύπους της τριγωνομετρίας. Ας σημειωθεί ότι σε όλες τις τριγωνομετρικές συναρτήσεις, η τιμή της παραμέτρου x πρέπει να εκφράζεται σε ακτίνια.

Παραδείγματα 7-7.

α) Η κωδικοποίηση της παράστασης:

$$y = \cos^2(1 + \tan^2)$$

είναι:

```
Y = Cos(Angle)^2 * ( 1 + Tan(Angle)^2 )
```

β) Η κωδικοποίηση της παράστασης:

$$y = \frac{x}{\frac{x}{x} - \frac{x}{x}}$$

είναι:

```
Y = (Sin(w) - Cos(x))/(Tan(x)/Sin(w) - 1/(Tan(w)*Cos(x)))
```

Η αντίστροφη τριγωνομετρική συνάρτηση τόξο εφαπτομένης Atn(x)

Από τις αντίστροφες τριγωνομετρικές συναρτήσεις, μόνο η συνάρτηση που δίνει το τόξο εφαπτομένης περιέχεται στις ενσωματωμένες συναρτήσεις. Με τη βοήθεια όμως αυτής είναι δυνατόν να υπολογιστούν οι τιμές όλων των άλλων αντίστροφων τριγωνομετρικών συναρτήσεων.

Παραδείγματα 7-8.

```
ArcTan = Atn(x)                      Τόξο εφαπτομένης
ArcSin = Atn( x/( 1-x*x )^0.5 )      Τόξο ημιτόνου (-1< x < 1)
```

Οι μεταβλητές ArcTan και ArcSin παριστάνουν τόξα που οι τιμές τους εκφράζονται σε ακτίνια μέσα από το διάστημα [-π/2, π/2]. Επίσης:

```
ArcCot =-Atn(x) + 3.141593/2          Τόξο συνεφαπτομένης
ArcCos =-Atn(x/(1-x*x))+3.141593/2    Τόξο συνημιτόνου (-1< x < 1)
```

με τιμές των μεταβλητών ArcCot και ArcCos μέσα από το διάστημα [0, π].

Εκμεταλλευόμενοι το γεγονός ότι η τιμή της συνάρτησης Atn είναι σε ακτίνια, μπορούμε να υπολογίζουμε τον αριθμό π με την εντολή:

```
ð = 4*Atn(1)
```

αφού τοξεφ 1 = π/4.

Η εκθετική συνάρτηση Exp(x)

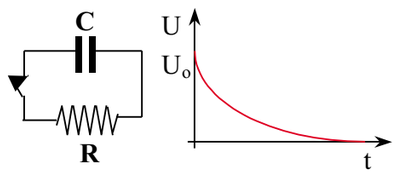
Η τιμή της συνάρτησης Exp είναι το αποτέλεσμα της ύψωσης του αριθμού e στη δύναμη x, όπου e, η βάση των φυσικών ή νεπερίων λογαρίθμων, με προσεγγιστική τιμή 2,718282. Ας σημειωθεί ότι η εκθετική συνάρτηση είναι μια συνάρτηση που συναντιέται σε πολλούς τύπους της Φυσικής και της Στατιστικής.

Παράδειγμα 7-9.

Αποδεικνύεται ότι η καμπύλη εκφόρτισης του πυκνωτή του σχήματος 7-2 περιγράφεται από τον τύπο:

$$U = U_o e^{\frac{t}{RC}}$$

με Uo την τάση στα άκρα του πυκνωτή κατά τη χρονική στιγμή 0, R την αντίσταση εκφόρτισης και C τη χωρητικότητα του πυκνωτή. Η κωδικοποίηση αυτού του τύπου στη VB είναι:



Σχήμα 7-2. Εκφόρτιση πυκνωτή

```
U = U0 * Exp(-t/(R*C))
```


Η λογαριθμική συνάρτηση Log(x)

Η συνάρτηση Log, που είναι η αντίστροφη της εκθετικής συνάρτησης με βάση το e, υπολογίζει το νεπέριο λογάριθμο του x και ορίζεται μόνο για θετικές τιμές της παραμέτρου.

Παράδειγμα 7-10.

α) Για να υπολογίσουμε τον δεκαδικό λογάριθμο ενός αριθμού χρησιμοποιούμε τον τύπο:

$$\log_{10} x = \frac{\log_e x}{\log_e 10}$$

Με την εντολή:

```
TenBaseLog  = Log(x)/Log(10)
```

υπολογίζεται και καταχωρίζεται στη μεταβλητή TenBaseLog ο δεκαδικός λογάριθμος της τιμής της μεταβλητής x.

β) Στη μέτρηση καλωδίων, για να βρεθεί η εξασθένιση που υφίσταται ένα σήμα από το ένα τους άκρο στο άλλο σε Decibel, εφαρμόζεται ο τύπος:

$$A = 10 \log_{10} \frac{P_{εισ}}{P_{εξ}}$$

με $P_{εισ}$ την ισχύ του σήματος στην είσοδο και $P_{εξ}$ την ισχύ του σήματος στην έξοδο.
Στη VB γράφουμε:

```
Attenuation = 10*Log(Pin/Pout)/Log(10)
```

Παραγωγή τυχαίων αριθμών

Η συνάρτηση RND(x) και η εντολή RANDOMIZE

Η τιμή της συνάρτησης Rnd είναι ένας τυχαίος αριθμός μέσα από το διάστημα (0,1). Οι αριθμοί, που προκύπτουν από διαδοχικές κλήσεις αυτής της συνάρτησης, δεν έχουν καμιά σχέση μεταξύ τους και γι' αυτό θεωρούμε ότι αποτελούν μια ακολουθία τυχαίων αριθμών.

Η παράμετρος της συνάρτησης δρα κάπως διαφορετικά από τις παραμέτρους των συναρτήσεων που μελετήσαμε μέχρι τώρα. Δεν καθορίζει άμεσα την τιμή της. Αν x > 0 ή αν δεν την γράψουμε καθόλου, μια και είναι προαιρετική, η συνάρτηση παράγει τον επόμενο όρο της ακολουθίας των τυχαίων αριθμών.

Η συνάρτηση είναι χρήσιμη στην προσομοίωση στατιστικών φαινομένων, στα πειράματα τύχης, στη δημιουργία τυχαίων κωδίκων, στα παιχνίδια, στα οποία θέλουμε να έχει τυχαία συμπεριφορά ο υπολογιστής.
Στα δίκτυα υπολογιστών, στα οποία χρησιμοποιείται η τεχνική πολλαπλής προσπέλασης με ανίχνευση φέροντος και ανίχνευση σύγκρουσης CSMA/CD, αν ανιχνευτεί σύγκρουση μηνυμάτων μεταξύ δύο σταθμών, οι σταθμοί επανεκπέμπουν σε χρονική στιγμή που υπολογίζεται από γεννήτρια ψευδοτυχαίων.

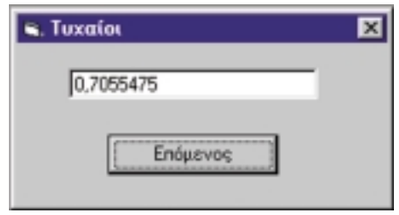
Άσκηση 7-1.

Να δημιουργηθεί η φόρμα της εικόνας 7-1, με το πλαίσιο κειμένου RandomNumber και το πλήκτρο διαταγής Generator (με **Caption** "Åðüìåíïò"). Γράφουμε τον κώδικα:

```
Private Sub Generator_Click()
    RandomNumber = Rnd
End Sub
```

Κάθε φορά που πατάμε το πλήκτρο Generator εμφανίζεται και ένας τυχαίος αριθμός. Διαδοχικά εμφανίζονται οι αριθμοί:

```
0,7055475     0,533424     0,5795186
0,2895625     0,301948     0,7747401
1,401764E-02  0,7607236    0,81449
```



Εικόνα 7-1.

Οι αριθμοί αυτοί, από ό,τι μπορείτε να διαπιστώσετε, δεν έχουν καμιά σχέση μεταξύ τους και γι' αυτό θεωρούνται τυχαίοι.

Πρέπει να τονίσουμε ότι κάθε φορά που ζητάμε την εκτέλεση του προγράμματος, παράγεται πάντα η ίδια ακολουθία αριθμών. Αυτή η επαναληψιμότητα είναι χρήσιμη στη φάση που ελέγχουμε, αν το πρόγραμμά μας εργάζεται σωστά. Είναι όμως ενοχλητική και ανεπιθύμητη,

όταν το πρόγραμμα έχει παραδοθεί για χρήση. Πρέπει λοιπόν με κάποιο τρόπο να δημιουργούμε τις κατάλληλες αρχικές συνθήκες, ώστε το πρόγραμμά μας, κάθε φορά που καλείται, να παράγει διαφορετικές ακολουθίες τυχαίων αριθμών. Η δημιουργία αυτών των συνθηκών γίνεται με την εισαγωγή κατάλληλου ***σπόρου*** (***seed***), όπως συνηθίζεται να λέγεται, στον αλγόριθμο υπολογισμού των όρων της ακολουθίας. Η "σπορά" γίνεται με ειδική κλήση της συνάρτησης Rnd δίνοντας στην παράμετρο x τιμή μικρότερη του μηδενός. Κάθε διαφορετική αρνητική τιμή αποτελεί το σπόρο μιας διαφορετικής ακολουθίας.

Άσκηση 7-2.

Στην προγραμματιστική μονάδα της φόρμας της άσκησης 7-1 να προστεθεί η υπορουτίνα διαχείρισης συμβάντος:

```
Private Sub Form_Load()
    RandomNumber = Rnd(-5)       'Ôðüñïò -5
End Sub
```

Έτσι, με το φόρτωμα της φόρμας, εισάγεται ο σπόρος -5 και παίρνουμε μια ακολουθία τυχαίων την:

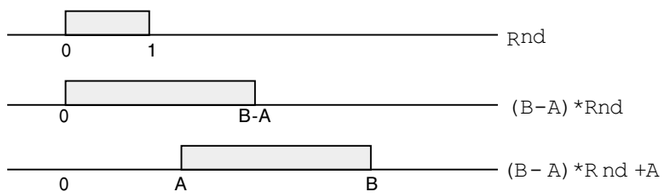
0,8383257 0,2874333 0,8604596 0,8178332 0,6038546 0,2088641

Επίσης, με σπόρο -6 παίρνουμε την ακολουθία:

0,4633257 0,4124333 0,4854596 0,9428332 0,2288546 0,3338641

Αν θέλουμε να παράγεται εντελώς τυχαία ακολουθία, που να μην εξαρτάται ούτε από την τιμή του σπόρου, χρησιμοποιούμε την εντολή Randomize. Η εντολή Randomize πραγματοποιεί "σπόρα" που εξαρτάται από την ώρα του ρολογιού του υπολογιστή. Έτσι είναι δύσκολο να καταφέρει ο χρήστης να "διαλέξει" ακολουθία, γιατί, ακόμα και αν παρακολουθεί το χρόνο του υπολογιστή, είναι αδύνατο να συντονιστεί και να ζητήσει εκτέλεση του προγράμματος ακριβώς τη στιγμή που έχει υπολογίσει ότι θα προκύψει η συγκεκριμένη ακολουθία.

Για να παράγονται τυχαίοι αριθμοί από ένα οποιοδήποτε διάστημα (A, B) και όχι αναγκαστικά από το (0, 1), πρέπει πρώτα να κάνουμε αναγωγή των αριθμών του διαστήματος (0, 1) σε διάστημα πλάτους B-A και ύστερα μεταφορά τους κατά A. Τα βήματα αυτής της διαδικασίας φαίνονται στο σχήμα 7-3.



Σχήμα 7-3. *Αναγωγή του διαστήματος (0, 1) στο διάστημα (A, B)*

Άσκηση 7-3.

Σε ένα πείραμα τύχης με ζάρια, θέλουμε να παίρνουμε ομοιόμορφα κατανεμημένους ακέραιους αριθμούς στο διάστημα [1, 6]. Στην προγραμματιστική μονάδα της άσκησης 7-1 να γραφούν οι υπορουτίνες διαχείρισης συμβάντων:

```
Option Explicit
Private Sub Form_Load()
    Randomize
End Sub
Private Sub Generator_Click()
    RandomNumber = Int((6 * Rnd) + 1)
End Sub
```

Πράγματι, η παράσταση 6*Rnd δίνει αριθμό του διαστήματος (0, 6), η συνάρτηση Int τον μετατρέπει σε ακέραιο του διαστήματος [0, 5] και η προσθήκη της μονάδας δίνει τη ζαριά.

Αλφαριθμητικές συναρτήσεις

Ως αλφαριθμητικές συναρτήσεις ορίζουμε τις συναρτήσεις εκείνες, που είτε η τιμή τους είναι αλφαριθμητική είτε περιλαμβάνουν στις παραμέτρους τους μία τουλάχιστον αλφαριθμητική ποσότητα. Οι αλφαριθμητικές συναρτήσεις είναι κάτι το ξεχωριστό στη VB. Χρησιμεύουν στη διαμόρφωση και τη δημιουργία νέων αλφαριθμητικών ποσοτήτων από τον κατακερματισμό άλλων, στη διαχείριση λέξεων, προτάσεων και κειμένων, καθώς και στη μετατροπή και τη διαχείριση κωδίκων. Οι αλφαριθμητικές συναρτήσεις είναι:

Συνάρτηση	Λειτουργία
Asc(a)	Ο κωδικός αριθμός ASCII του πρώτου χαρακτήρα του a
Chr(x)	Ο χαρακτήρας ASCII του αριθμού x
Insrt(x,a,b)	Η θέση του b μέσα στο a, αρχίζοντας από τη θέση x
LCase(a)	Μετατροπή κεφαλαίων σε πεζά
Len(x)	Το πλήθος των χαρακτήρων του x
Left(a,x)	Οι x πρώτοι χαρακτήρες του a
LTrim(a)	Αφαιρεί τα διαστήματα από την αρχή του a
Mid(a,x,y)	y χαρακτήρες του a, αρχίζοντας από τον x-οστό
Right(a,x)	Οι x τελευταίοι χαρακτήρες του a
RTrim(a)	Αφαιρεί τα διαστήματα από το τέλος του a
Space(x)	Η συμβολοσειρά που αποτελείται από x διαστήματα
String(x,a)	Μια συμβολοσειρά από x όμοιους χαρακτήρες a
Trim(a)	Αφαιρεί τα διαστήματα από την αρχή και το τέλος
UCase(a)	Μετατροπή πεζών σε κεφαλαία

Πίνακας 7-2. *Οι ενσωματωμένες αλφαριθμητικές συναρτήσεις. Οι παράμετροι a, b δηλώνουν αλφαριθμητικές ποσότητες και οι παράμετροι x, y δηλώνουν αριθμητικές.*

Η συνάρτηση Len(a)

Η τιμή της συνάρτησης Len ισούται με το πλήθος των χαρακτήρων της τιμής της αλφαριθμητικής ποσότητας a. Τα διαστήματα και οι μη εκτυπούμενοι χαρακτήρες, που τυχόν περιέχονται στην τιμή της a, λαμβάνονται υπ' όψιν στην απαρίθμηση των χαρακτήρων.

Παράδειγμα 7-11.

```
Length = Len(Sentence)
Message= "ÕðÜñ÷ïõí " & Length & " ÷áñáêôÞñåò óôçí ðñüôáóç " & Sentence
```

Αν στη μεταβλητή Sentence δοθεί η τιμή "Áõôïêéíçôüäçìïò", η μεταβλητή Message θα πάρει την τιμή "ÕðÜñ÷ïõí 16 ÷áñáêôÞñåò óôçí ðñüôáóç Áõôïêéíçôüäçìïò"

Οι συναρτήσεις Left(a,x) και Right(a,x)

Η τιμή της συνάρτησης Left είναι μια συμβολοσειρά που αποτελείται από τους x πρώτους χαρακτήρες της τιμής της αλφαριθμητικής ποσότητας a. Αν x>Len(a), η συνάρτηση παίρνει την τιμή του x, ενώ αν x = 0, η συνάρτηση παίρνει τιμή το "".

Παράδειγμα 7-12.

```
FirstName = "Áñéóôïôýëçò"
SurName = "Êéïýóçò"
FatherName = "Ðáýëïò"
InitialsAndName = Left(FirstName,1) & "." & _
                  Left(FatherName,1) & "." & _
                  SurName
```

Το όνομα, το επώνυμο και το πατρώνυμο ενός ατόμου έχουν καταχωριστεί στις μεταβλητές, FirstName, SurName και FatherName αντίστοιχα, και η τιμή της μεταβλητής InitialsAndName είναι "Á.Ð.Êéïýóçò".

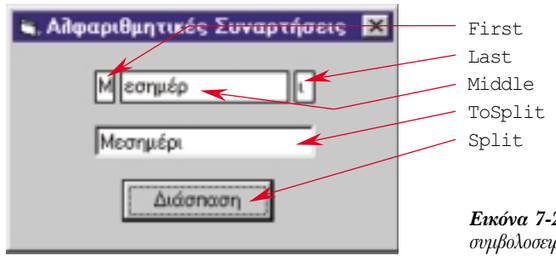
Η τιμή της συνάρτησης `Right` είναι μια συμβολοσειρά που αποτελείται από τους `x` τελευταίους χαρακτήρες της αλφαριθμητικής ποσότητας `a`. Αν `x>Len(a)`, η συνάρτηση παίρνει την τιμή του `x`, ενώ αν `x = 0`, η συνάρτηση παίρνει την τιμή `""`.

Η συνάρτηση `Mid(a,x,y)` και η εντολή `Mid`

Η τιμή της συνάρτησης `Mid` (από το middle) είναι μια συμβολοσειρά που αποτελείται από `y` χαρακτήρες της `a`, αυτούς που βρίσκονται από τον χιοστό χαρακτήρα και μετά. Η χρήση της παραμέτρου `y` είναι προαιρετική. Αν δε χρησιμοποιηθεί η παράμετρος, η συμβολοσειρά που θα προκύψει θα περιλαμβάνει όλους τους χαρακτήρες της `a`, από τη θέση `x` και μετά. Αν `y=0` ή `x>Len(a)`, η συνάρτηση εξισώνεται με τη μηδενική συμβολοσειρά `""`.

Άσκηση 7-4.

Να δημιουργηθεί η φόρμα της εικόνας 7-2 και να δοθούν ονόματα στα αντικείμενα πλαίσια κειμένου.



Εικόνα 7-2. *Αρχή, μέσο, τέλος συμβολοσειράς*

Πριν πατήσετε το πλήκτρο `"ÄéÜóðáóç"`, βεβαιωθείτε ότι έχετε πληκτρολογήσει τρεις χαρακτήρες στο πλαίσιο κειμένου `ToSplit`. Διαφορετικά, θα προκύψει λάθος, επειδή η συνάρτηση `Mid` δε δέχεται αρνητικές τιμές στις παραμέτρους της.

Στη συνέχεια γράφουμε το πρόγραμμα:

```
Private Sub Split_Click()
    Dim Distance     As Integer
    First = Left(ToSplit, 1)              'Ðñþôïò ÷áñáêôÞñáò
    Distance = Len(ToSplit) - 2
    Middle = Mid(ToSplit, 2, Distance)    'Ìåóáßïé ÷áñáêôÞñáò
    Last = Right(ToSplit, 1)              'Ôåëåõôáßïò ÷áñáêôÞñáò
End Sub
```

Όταν τρέχει το πρόγραμμα, στο πλαίσιο κειμένου `ToSplit` γράφουμε μια λέξη (με περισσότερους από 2 χαρακτήρες) ή μια πρόταση και πατάμε το πλήκτρο διαταγής `Split` για να γίνει διάσπασή της στα πλαίσια κειμένου `First`, `Middle` και `Last`.

Τη στιγμή που αρχίζουμε να πληκτρολογούμε τις παραμέτρους μιας συνάρτησης εμφανίζεται πληροφοριακή ετικέτα που μας θυμίζει τον τρόπο σύνταξη της.

Η εντολή `Mid` ανήκει στην κατηγορία των εντολών εκχώρησης τιμής και η γενική της μορφή είναι:

`Mid(a,x,y)` = *παράσταση*

Με αυτή την εντολή αντικαθίστανται `y` χαρακτήρες της μεταβλητής `a`, από τους πρώτους `y` χαρακτήρες της τιμής της *παράσταση*, αρχίζοντας από τον χιοστό χαρακτήρα της.

Παράδειγμα 7-13.

Το παιχνίδι "κρεμάλα" είναι λίγο πολύ γνωστό σε όλους. Στην αρχή ο ένας παίκτης γράφει το πρώτο και το τελευταίο γράμμα μιας λέξης, ενώ στη μέση όλων των άλλων γραμμάτων τοποθετεί το σύμβολο πλην (-). Το τμήμα προγράμματος, που ακολουθεί, βοηθά τον παίκτη στη σωστή γραφή του σκελετού της λέξης.

```
Dim Crypto          As String
Dim LengthChange    As Integer

LengthChange = Len(Crypto) - 2
Mid(Crypto, 2, LengthChange) = "----------------"
```


Αν η μεταβλητή `Crypto` έχει πάρει την τιμή `"ÖõëëïîÞñá"`, η τιμή της μετά την εκτέλεση του τμήματος προγράμματος θα είναι: `"Ö-------Á"`, αφού από τη συμβολοσειρά με τα πλην θα ληφθεί ένα τμήμα μήκους `Len(Crypto)-2` και θα αντικαταστήσει τα γράμματα από το δεύτερο και μετά γράμμα.

Η συνάρτηση `Instr(x,a,b)`

Αυτή η συνάρτηση ερευνά, αν υπάρχει η συμβολοσειρά `b` μέσα στην `a` και μας δίνει τη θέση στην οποία βρέθηκε. Η τιμή της παράστασης `x` υποδεικνύει τη θέση από την οποία θα αρχίσει η έρευνα. Αν το `x` παραλειφθεί, η αναζήτηση αρχίζει από τον πρώτο χαρακτήρα του `a`. Αν το `b` δεν υπάρχει μέσα στο `a` ή `x>Len(a)` ή `a=""` η συνάρτηση παίρνει την τιμή μηδέν. Αν `b=""` η συνάρτηση παίρνει την τιμή `x`. Στην περίπτωση όμως που δεν έχουμε γράψει το `x`, παίρνει την τιμή `1`.

Παράδειγμα 7-14.

Στο τμήμα προγράμματος:

```
Sentence  = "Íá ìéá ðÜðéá ìá ðéá ðÜðéá"
Word = "ðéá"
Position = Instr(5, Sentence, Word)
```

η μεταβλητή `Position` θα πάρει την τιμή 10.

Οι συναρτήσεις `String(y,a ή x)` και `Space(x)`

Η τιμή της συνάρτησης `String` είναι η συμβολοσειρά που προκύπτει από την επανάληψη, `y` φορές, του πρώτου χαρακτήρα της τιμής της ποσότητας `a` ή του χαρακτήρα που αντιστοιχεί στον αριθμό ASCII `x`. Δηλαδή, η δεύτερη παράμετρος της συνάρτησης `String` μπορεί να είναι αλφαριθμητική ή αριθμητική ποσότητα.

Παραδείγματα 7-15.

α) Η εντολή:

```
HidePassword = String(20, "*" )
```

δίνει στη μεταβλητή `HidePassword` την τιμή:

```
********************
```

β) Η εντολή:

```
LineFeeds = String(25, vbLf )
```

δημιουργεί μια συμβολοσειρά με 25 χαρακτήρες αλλαγής γραμμής αφού η εσωτερική σταθερά `vbLf` αντιστοιχεί στον κωδικό ASCII 10 (Line Feed).

Η συνάρτηση `Space(x)` είναι ειδική περίπτωση της συνάρτησης `String`. Είναι ισοδύναμη με την `String(x," ")`, αφού δημιουργεί μια συμβολοσειρά που αποτελείται από `x` διαστήματα.

Απομάκρυνση διαστημάτων από την αρχή και το τέλος

Οι συναρτήσεις `LTrim(a)`, `RTrim(a)` και `Trim(a)`

Πολλές φορές, οι χρήστες πληκτρολογούν από απροσεξία διαστήματα στην αρχή ή στο τέλος συμβολοσειρών μέσα σε πλαίσια κειμένου. Τα διαστήματα, παρ' όλο που δε φαίνονται, αποτελούν μέρος της συμβολοσειράς, με αποτέλεσμα να εκχωρείται λάθος τιμή στο πλαίσιο κειμένου. Για παράδειγμα οι τιμές:

```
" ×ùñßò äéáóôÞìáôá"
"×ùñßò äéáóôÞìáôá"
"×ùñßò äéáóôÞìáôá "
```

αν και ένας χρήστης μπορεί να τις θεωρεί ταυτόσημες είναι διαφορετικές. Οι συναρτήσεις `LTrim` (Left Trim), `RTrim` (Right Trim) και `Trim` αφαιρούν τα διαστήματα από την αρχή (αριστερά), το τέλος (δεξιά) και την αρχή και το τέλος, αντίστοιχα.

Μετατροπή πεζών γραμμάτων σε κεφαλαία και αντίστροφα

Οι συναρτήσεις UCase(a), LCase(a)

Η συνάρτηση `UCase` (Upper Case) μετατρέπει όλα τα γράμματα της τιμής της παραμέτρου της σε κεφαλαία. Αντίστοιχα, η συνάρτηση `LCase` (Lower Case) μετατρέπει όλα τα γράμματα της τιμής της παραμέτρου της σε πεζά.

Παράδειγμα 7-16.

Η εντολή:

```
Sentence = UCase(Left(Sentence,1)) & LCase(Mid(Sentence,2))
```

μας εξασφαλίζει ότι η τιμή της μεταβλητής `Sentence` θα έχει το πρώτο γράμμα πάντα κεφαλαίο και τα υπόλοιπα μικρά.

Οι συναρτήσεις Asc(a) και Chr(x)

Η συνάρτηση `Asc` μας δίνει τον αριθμό ASCII που αντιστοιχεί στον πρώτο χαρακτήρα της τιμής της ποσότητας `a`.

Παράδειγμα 7-17.

Με τις εντολές:

```
AscValue = Asc( "914" )
AscNumber = Asc( "TWO" )
AscCode = Asc( "öéëßá" )
```

οι μεταβλητές `AscValue`, `AscNumber` και `AscCode` θα πάρουν τις τιμές `57`, `84` και `246` αντίστοιχα.

Ας σημειωθεί ότι για `a=""`, η κλήση της συνάρτησης προκαλεί διακοπή της εκτέλεσης του προγράμματος και εμφάνιση του μηνύματος "**Illegal function call**".

Η συνάρτηση `Chr` (από τη λέξη Character) είναι η αντίστροφη της συνάρτησης Asc. Μας δίνει το χαρακτήρα ASCII, που αντιστοιχεί στην τιμή της αριθμητικής ποσότητας `x`. Η τιμή του `x` πρέπει να είναι μέσα στο διάστημα [0, 255], αλλιώς στην οθόνη εμφανίζεται το μήνυμα "**Illegal function call**".

Στον κώδικα ASCII οι 128 πρώτοι χαρακτήρες είναι ακριβώς αυτοί που έχουν καθοριστεί από το Αμερικανικό Ινστιτούτο προτύπων (ANSI). Οι χαρακτήρες που υπάρχουν μετά τη θέση 128, δεν έχουν καθιερωθεί διεθνώς, γι' αυτό και οι οργανισμοί τυποποίησης κάθε χώρας μπορούν να τους καθορίσουν ανάλογα με τα εθνικά αλφάβητα. Στη χώρα μας έχει καθιερωθεί πλέον για το περιβάλλον των Windows ένα πρότυπο αντιστοιχίας των ελληνικών γραμμάτων (το πρότυπο 928).

Παράδειγμα 7-18.

```
Delta = Chr(196)
HamletSaid = "Êáé ï Áìëåô åßðå: " & Chr(34) & _
             "To be or not to be" & Chr(34)
LetterA = Chr(&Ç61)
```

Με την πρώτη εντολή, η μεταβλητή `Delta` παίρνει για τιμή το γράμμα Ä , αφού αυτό το γράμμα αντιστοιχεί στον αριθμό 196 σύμφωνα με τον κώδικα 928. Επίσης, η μεταβλητή `HamletSaid` γίνεται `Êáé ï Áìëåô åßðå: "To be or not to be"` αφού ο αριθμός 34 αντιστοιχεί στα διπλά εισαγωγικά (") σύμφωνα με τον κώδικα ASCII. Τέλος, η τελευταία εντολή αντιστοιχεί στη μεταβλητή `LetterA` το λατινικό γράμμα `a`.

Στην τελευταία εντολή χρησιμοποιήσαμε τη δεκαεξαδική παράσταση του αριθμού. Αυτό είναι σύνηθες κατά τη χρήση της συνάρτησης `Chr`, γιατί ο πίνακας ASCII που δίνει την αντιστοιχία χαρακτήρων - δεκαεξαδικών αριθμών είναι πιο εύχρηστος από τον πίνακα αντιστοιχίας χαρακτήρων δεκαδικών αριθμών.

Συνήθως, χρησιμοποιούμε τη συνάρτηση `Chr` για να τοποθετήσουμε σε μια αλφαριθμητική ποσότητα ειδικούς χαρακτήρες. Για παράδειγμα, για την παρεμβολή των εισαγωγικών (") σε ένα μήνυμα. Επίσης, η συνάρτηση χρησιμοποιείται για την αποστολή ειδικών χαρακτήρων που προκαλούν ειδικές λειτουργίες στις περιφερειακές συσκευές. Για παράδειγμα, ο χαρακτήρας με κωδικό 07 (BEL), όταν σταλεί προς ένα τερματικό, το αναγκάζει να εκπέμψει έναν χαρακτηριστικό ήχο (μπιπ).

Συναρτήσεις μετατροπής τύπων δεδομένων

Οι συναρτήσεις αυτές είναι είτε αριθμητικές είτε αλφαριθμητικές συναρτήσεις που πραγματοποιούν μετατροπές από έναν τύπο μεταβλητής σε άλλο. Οι συναρτήσεις μετατροπής δεδομένων της VB είναι:

Συνάρτηση	Τύπος	Περιοχή τιμών / Λειτουργία
`CBool(v)`	Boolean	Αριθμητική ή αποδεκτή αλφαριθμητική τιμή (π.χ. "True")
`CByte(v)`	Byte	0 έως 255
`CCur(v)`	Currency	Τύπος νομίσματος (πίνακας 6-1)
`CDbl(v)`	Double	Τύπος διπλής ακρίβειας (πίνακας 6-1)
`CInt(v)`	Integer	-32768 έως 32767
`CLng(v)`	Long	-2.147.483.648 έως 2.147.483.647
`CSng(v)`	Single	Τύπος απλής ακρίβειας (πίνακας 6-1)
`CVar(v)`	Variant	Περιοχή διπλής ακριβείας για αριθμητικές τιμές Τύπος
		String για μη αριθμητικές
`CStr(v)`	String	Οποιαδήποτε αριθμητική ή αλφαριθμητική τιμή
`IsNumeric(v)`	Boolean	Ελέγχει αν η παράμετρος αποτελεί αριθμό
`Hex(x)`	String	-2.147.483.648 έως 2.147.483.647
`Oct(x)`	String	-2.147.483.648 έως 2.147.483.647

Πίνακας 7-3. *Οι ενσωματωμένες συναρτήσεις μετατροπής τύπων δεδομένων. Η παράμετρος v μπορεί να είναι αριθμητικού ή αλφαριθμητικού τύπου και η παράμετρος x αριθμητικού.*

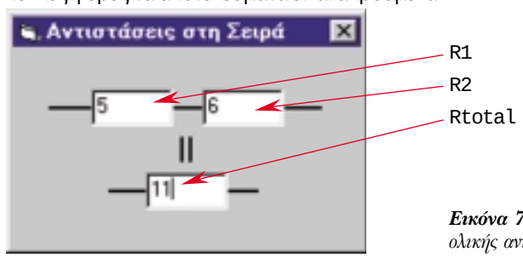
Οι συναρτήσεις Cxxx(v)

Πρόκειται για τις συναρτήσεις `CBool`, `CByte`, `CCur`, `CDbl`, `CInt`, `CLng`, `CSng`, `CVar`, `CStr`. Από το όνομά τους μπορούμε να προσδιορίσουμε την τιμή τους και τον τύπο δεδομένων που επιστρέφουν ως τιμή. Βασικό χαρακτηριστικό αυτών των συναρτήσεων είναι ότι η παράμετρος v μπορεί να είναι αριθμητικού ή αλφαριθμητικού τύπου.

Οι τιμές των παραμέτρων των συναρτήσεων μετατροπής τύπων δεδομένων πρέπει να είναι μέσα στα αποδεκτά όρια που αναφέρονται στον πίνακα 7-3. Για παράδειγμα, αν πρόκειται να μετατρέψουμε μια μεγάλου ακέραιου (`Long`) μεταβλητή σε μικρό ακέραιο (`Integer`), η τιμή της πρέπει να είναι μεταξύ -32768 και 32767.

Άσκηση 7-5.

Μέχρι τώρα προσπαθήσαμε να δώσουμε παραδείγματα που δεν περιείχαν την επεξεργασία αριθμητικών δεδομένων. Ο λόγος είναι ότι τα πλαίσια κειμένου, όπως δηλώνει και το όνομά τους, διαχειρίζονται τις τιμές που εισάγει ο χρήστης σαν συμβολοσειρές. Γι' αυτό το λόγο πολλές φορές τα αποτελέσματα είναι απρόσμενα.



Εικόνα 7-4. *Υπολογισμός ολικής αντίστασης*

Σε αυτή την άσκηση θα δημιουργήσουμε μια φόρμα με την οποία θα υπολογίζουμε την ολική αντίσταση δύο αντιστάσεων, οι οποίες βρίσκονται στη σειρά. Θέλουμε τη στιγμή που πληκτρολογούμε μια τιμή, να εμφανίζεται αυτόματα το αποτέλεσμα στο πλαίσιο κειμένου `Rtotal`, χωρίς να απαιτείται το πάτημα κάποιου πλήκτρου. Το κατάλληλο συμβάν για την περίπτωση είναι το συμβάν **Change**. Για να εξασφαλίσουμε τη σωστή λειτουργία του προγράμματος, πρέπει από την αρχή να υπάρχουν αριθμητικές τιμές στα πλαίσια κειμένου R1 και R2. Γι' αυτό και με το φόρτωμα της φόρμας (συμβάν **Load**) δίνουμε αρχική τιμή 0 στα R1 και R2.

Γράφουμε τον κώδικα:

```
Private Sub Form_Load()
    R1 = 0
    R2 = 0
    Rtotal = 0
End Sub

Private Sub R1_Change()
    Rtotal = CDbl(R1) + CDbl(R2)
End Sub

Private Sub R2_Change()
    Rtotal = CDbl(R1) + CDbl(R2)
End Sub
```

Αν η παράμετρος μιας συνάρτησης Cxxx είναι αλφαριθμητική ποσότητα, η τιμή της πρέπει να αντιστοιχεί σε αριθμό. Διαφορετικά, γίνεται διακοπή της εκτέλεσης του προγράμματος και εμφανίζεται το μήνυμα λάθους "**Type mismatch**". Για να αποφύγουμε αυτό το λάθος, επιβάλλεται να ελέγχουμε πρώτα το περιεχόμενο της παραμέτρου. Αυτό γίνεται με τη συνάρτηση `IsNumeric`, που παίρνει την τιμή `True`, αν η τιμή της παραμέτρου αντιστοιχεί σε αριθμό, και την τιμή `False`, αν η τιμή δεν αντιστοιχεί σε αριθμό.

Άσκηση 7-6.

Η βελτιωμένη μορφή της υπορουτίνας εξυπηρέτησης συμβάντος **`Change`** της άσκησης 7-5

```
Private Sub R1_Change()
   If IsNumeric(R1) And IsNumeric(R2) Then
        Rtotal = CDbl(R1) + CDbl(R2)
   End If
End Sub
```

Την εντολή `If` θα τη μελετήσουμε στο επόμενο μάθημα.

δηλαδή, αν και το R1 και το R2 περιέχουν τιμές που αντιστοιχούν σε αριθμούς τότε μόνο υπολογίζεται η ολική αντίσταση.

Οι συναρτήσεις `Hex(x)` και `Oct(x)`

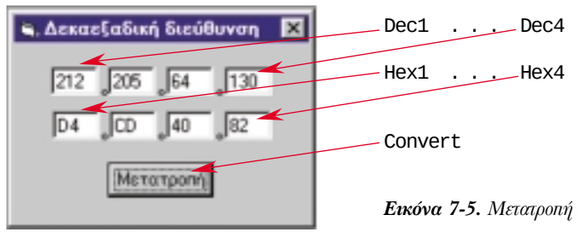
Συχνά, οι μηχανικοί υπολογιστών χρησιμοποιούν το δεκαεξαδικό σύστημα, και κάπως σπανιότερα το οκταδικό, για να παραστήσουν αριθμούς, που αφορούν τεχνικά στοιχεία του υπολογιστή (π.χ. μνήμες, καταχωρητές επικοινωνιών, σειρές από μικροδιακόπτες κ.ά.). Η τιμή της συνάρτησης `Hex` είναι μια συμβολοσειρά, που παριστά τη δεκαεξαδική μορφή της τιμής της παράστασης x. Επίσης, η τιμή της συνάρτησης `Oct` παριστά την οκταδική μορφή της τιμής της παράστασης x. Η τιμή της x πρέπει να είναι ακέραιος, διαφορετικά γίνεται στρογγυλοποίηση της τιμής. Τέλος, αν `x < 0`, η συμβολοσειρά που δίνει η συνάρτηση αντιστοιχεί στην παράσταση των αρνητικών αριθμών σε ***συμπλήρωμα ως προς 2*** (***2's***

Ήδη στην άσκηση 5-3 είδαμε ότι το δεκαεξαδικό σύστημα χρησιμοποιείται για την παράσταση των χρωμάτων.

Η λογική του συμπληρώματος ως προς 2 καθορίζει τον τρόπο της αναπαράστασης των αρνητικών αριθμών σε μορφή δυαδικών ψηφίων στη μνήμη του υπολογιστή.

Άσκηση 7-7.

Οι IP διευθύνσεις στο Internet (Internet Protocol Addresses) αποτελούνται από τέσσερεις ακεραίους θετικούς αριθμούς μικρότερους του 256, χωρισμένους με τελείες, δηλαδή αποτελούνται από τέσσερα bytes. Να δημιουργηθεί φόρμα που να μετατρέπει τις διευθύνσεις από το δεκαδικό σύστημα αρίθμησης στο δεκαεξαδικό:



Εικόνα 7-5. *Μετατροπή διεύθυνσης IP*

και γράφουμε τον κώδικα:

```
Private Sub Form_Load()
    Dec1 = 0:   Hex1 = 0
    Dec2 = 0:   Hex2 = 0
    Dec3 = 0:   Hex3 = 0
    Dec4 = 0:   Hex4 = 0
End Sub

Private Sub Convert_Click()
    Hex1 = Hex(Dec1)
    Hex2 = Hex(Dec2)
    Hex3 = Hex(Dec3)
    Hex4 = Hex(Dec4)
End Sub
```

Ημερομηνίες και ώρα

Πρόκειται για μια πολύ σημαντική κατηγορία συναρτήσεων που μας απαλλάσσουν από πολλούς και συχνά πολύπλοκους υπολογισμούς.

Συνάρτηση	Τιμή/Λειτουργία
`Date()`	Τρέχουσα ημερομηνία
`DateSerial(y,m,d)`	Ημερομηνία από έτος, μήνα, ημέρα
`DateValue(v)`	Μετατροπή σε ημερομηνία
`Day(d)`	Ημέρα χρονικής στιγμής
`Hour(d)`	Ώρα χρονικής στιγμής
`Minute(d)`	Λεπτά χρονικής στιγμής
`Month(d)`	Μήνας χρονικής στιγμής
`Now()`	Τρέχουσα χρονική στιγμή
`Second(d)`	Δευτερόλεπτα χρονικής στιγμής
`Timer`	Τα δευτερόλεπτα που πέρασαν από τα μεσάνυχτα
`TimeSerial(h,m,s)`	Ώρα από ώρες, λεπτά, δευτερόλεπτα
`TimeValue(v)`	Μετατροπή σε ώρα
`Year(d)`	Έτος χρονικής στιγμής
`WeekDay(d)`	Ημέρα της εβδομάδας
`IsDate(v)`	Ελέγχει, αν η παράμετρος αποτελεί ημερομηνία

Πίνακας 7-4. *Οι ενσωματωμένες συναρτήσεις ημερομηνίας.*

Οι συναρτήσεις `Day`, `Month` και `Year` δίνουν την ημέρα, το μήνα και το έτος, αντίστοιχα, μιας ημερομηνίας. Π.χ.

```
οι    Day("13/01/2000")                13
      Month("13/01/2000") παίρνουν τιμές  1
      Year("13/01/2000")              2000 αντίστοιχα
```

Οι συναρτήσεις `Hour`, `Minute` και `Second` δίνουν την ώρα, τα λεπτά και τα δευτερόλεπτα, αντίστοιχα, μιας χρονικής στιγμής. Π.χ.

```
οι    Hour("08:38:09")                 8
      Minute("08:38:09") παίρνουν τιμές  38
      Second("08:38:09")               9 αντίστοιχα
```

Η συνάρτηση `Weekday` δίνει την ημέρα της εβδομάδας μιας ημερομηνίας, π.χ. 1 για την Κυριακή, 2 για τη Δευτέρα κ.ο.κ.

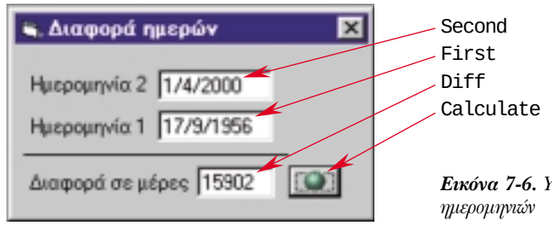
Η συνάρτηση `DateValue` μετατρέπει μια συμβολοσειρά σε τύπο δεδομένων ημερομηνίας (`Date`). Υπενθυμίζουμε ότι μόνο με τον τύπο δεδομένων `Date` είναι δυνατή η πράξη μεταξύ ημερομηνιών. Ανάλογα γίνεται και η μετατροπή μιας συμβολοσειράς σε ώρα από την `TimeValue`.

Στην VB υπάρχουν και οι εξής εσωτερικές σταθερές:

```
vbSunday      1
vbMonday      2
vbTuesday     3
vbWednesday   4
vbThursday    5
vbFriday      6
vbSaturday    7
```

Άσκηση 7-8.

Να δημιουργηθεί φόρμα και να γραφεί κώδικας που να υπολογίζει τον αριθμό των ημερών που μεσολαβούν μεταξύ δύο ημερομηνιών:



Εικόνα 7-6. *Υπολογισμός διαφοράς ημερομηνιών*

και γράφουμε την υπορουτίνα:

```
Private Sub Calculate_Click()
    Diff = Abs(DateValue(Second) - DateValue(First))
End Sub
```

Η συνάρτηση `DateSerial` επιστρέφει αντίστοιχη τιμή με την `DateValue` αλλά οι παράμετροί της είναι το έτος, ο μήνας και η ημέρα. Αντίστοιχα, η συνάρτηση `TimeSerial` δέχεται ως παραμέτρους την ώρα, τα λεπτά, τα δευτερόλεπτα.

Παράδειγμα 7-18.

Στο ερώτημα τι ημέρα "πέφτει" η 1/8/2004 απαντά η παράσταση:

```
Weekday(DateSerial(2004,8,1))
```

αφού η συνάρτηση `DateSerial` δίνει μια ημερομηνία τύπου δεδομένων `Date` και στη συνέχεια η `Weekday` υπολογίζει τον αύξοντα αριθμό της ημέρας.

Οι παράμετροι των συναρτήσεων ημερομηνίας μπορεί να είναι αριθμητικές ποσότητες, π.χ. 32456,25, ή αλφαριθμητικές ποσότητες που παριστάνουν ημερομηνίες. Αν η τιμή μιας παραμέτρου δεν αντιστοιχεί σε ημερομηνία διακόπτεται η εκτέλεση του προγράμματος και εμφανίζεται το μήνυμα λάθους "**Type mismatch**". Για να αποφύγουμε αυτό το λάθος, επιβάλλεται να ελέγχουμε πρώτα το περιεχόμενο της παραμέτρου. Αυτό γίνεται με τη συνάρτηση `IsDate`, που παίρνει την τιμή `True`, αν η τιμή της παραμέτρου αντιστοιχεί σε ημερομηνία, και την τιμή `False`, αν η τιμή δεν αντιστοιχεί σε ημερομηνία.

Ημερομηνία και ώρα συστήματος

Η συνάρτηση `Now` δίνει την τρέχουσα χρονική στιγμή σε μορφή ημερομηνίας και ώρας. Η συνάρτηση `Date` δίνει την τρέχουσα ημερομηνία και η συνάρτηση `Time` την τρέχουσα ώρα. Οι συναρτήσεις αυτές δεν έχουν παραμέτρους.

Επίσης, υπάρχει η εντολή εκχώρησης `Date` που αλλάζει την ημερομηνία του υπολογιστικού συστήματος και η εντολή `Time` που αλλάζει την ώρα. Οι μορφές των εντολών είναι:

`Date` = *παράσταση*
`Time` = *παράσταση*

Εφαρμογή

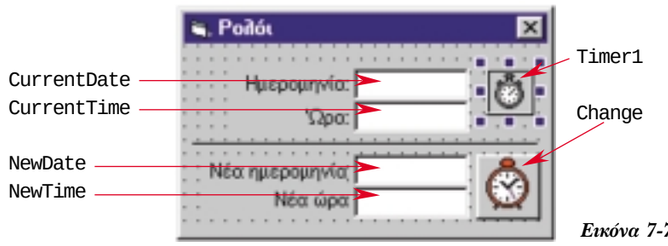
Ημερολόγιο - Χρονόμετρο

Οι χρήστες προγραμμάτων εκφράζουν συχνά την επιθυμία να μπορούν να εκτελέσουν κάποιες λειτουργίες χωρίς να απαιτείται να χρησιμοποιούν βοηθητικά προγράμματα του λειτουργικού συστήματος. Στη συνέχεια θα δημιουργήσουμε μια φόρμα, που θα μπορεί να χρησιμοποιηθεί από οποιοδήποτε έργο και από την οποία ο χρήστης θα βλέπει την τρέχουσα ημερομηνία και ώρα. Η ενημέρωση της φόρμας θα γίνεται κάθε δευτερόλεπτο και ο χρήστης θα έχει τη δυνατότητα να αλλάζει την ημερομηνία και ώρα.

Στη φόρμα διεπαφής τοποθετούμε:

1. Τα πλαίσια κειμένου `CurrentDate` και `CurrentTime` που θα δείχνουν την τρέχουσα ημερομηνία και ώρα και σύμφωνα με την προδιαγραφή θα ενημερώνονται κάθε δευτερόλεπτο.
2. Τα πλαίσια κειμένου `NewDate` και `NewTime`, από τα οποία θα εισάγεται η νέα ημερομηνία και ώρα. Για να είναι κενού περιεχομένου αυτά τα πεδία κατά την έναρξη του προγράμματος διαγράφουμε την τιμή της ιδιότητας **`Text`** (τιμή "").
3. Το πλήκτρο διαταγής `Change`, του οποίου το πάτημα θα αλλάζει την ημερομηνία και την ώρα του υπολογιστικού συστήματος. Στο πλήκτρο αυτό αλλάζουμε κατάλληλα τις ιδιότητες **`Caption`**, **`Style`** και **`Picture`**, ώστε στο εσωτερικό του να υπάρχει το εικονίδιο ενός ρολογιού.
4. Το χρονόμετρο `Timer1`. Αυτό το αντικείμενο ελέγχου αποτελεί και το κλειδί της εφαρμογής. Η ιδιότητα του χρονομέτρου **`Interval`** καθορίζει ανά πόσα msecs (χιλιοστά του δευτερολέπτου) θα προκύπτει, το σχετικό με αυτό, συμβάν **`Timer`**. Δίνουμε στην ιδιότητα **`Interval`** την τιμή 1000 και δηλώνουμε, κατ' αυτόν τον τρόπο, ότι ανά 1sec (1000 x 1/1000 sec) θα προκύπτει συμβάν ρολογιού.

Το αντικείμενο χρονόμετρο το χρησιμοποιούμε για να εισάγουμε καθυστέρηση στην εκτέλεση του κώδικα ή για να εκτελέσουμε κάποιον κώδικα σε προκαθορισμένες χρονικές στιγμές.



Εικόνα 7-7. *Χρονόμετρο*

Γράφουμε το πρόγραμμα :

```
Private Sub Timer1_Timer()
     'Óå êÜèå ÷ôýðï ôïõ ñïëïãéïý (áíÜ 1 sec)
      CurrentDate = Date            'ÅíçìÝñùóå ôçí çìåñïìçíßá
      CurrentTime = Time            'ÅíçìÝñùóå ôçí þñá
End Sub
Private Sub Change_Click()
     '¼ôáí ï ÷ñÞóôçò æçôÞóåé áëëáãÞ
      Date = NewDate
      Time = NewTime
End Sub
```

Ας σημειωθεί ότι κατά την εκτέλεση του προγράμματος το εικονίδιο του χρονομέτρου δεν εμφανίζεται μέσα στο παράθυρο.

Ανακεφαλαίωση

Για την πιο εύκολη κωδικοποίηση των τύπων, η VB περιέχει ενσωματωμένες συναρτήσεις ή συναρτήσεις βιβλιοθήκης. Στις ενσωματωμένες συναρτήσεις υπάρχουν μαθηματικές συναρτήσεις, αλφαριθμητικές συναρτήσεις, συναρτήσεις μετατροπής τύπων δεδομένων, συναρτήσεις ελέγχου τύπου και τιμής δεδομένων και συναρτήσεις ημερομηνιών. Στις παραμέτρους μπορούμε να γράψουμε σταθερές, μεταβλητές, παραστάσεις. Ανάλογα με τη συνάρτηση, οι τιμές των παραμέτρων καθορίζουν την τιμή της.

Στις μαθηματικές συναρτήσεις περιλαμβάνονται η τετραγωνική ρίζα, η απόλυτη τιμή, το πρόσημο, οι συναρτήσεις αποκοπής του ακεραίου μέρους, οι τριγωνομετρικές συναρτήσεις, η εκθετική συνάρτηση, η λογαριθμική συνάρτηση και η συνάρτηση παραγωγής τυχαίων αριθμών.

Στις αλφαριθμητικές συναρτήσεις περιλαβάνονται συναρτήσεις λήψης τμήματος μιας συμβολοσειράς (από την αρχή το μέσον και το τέλος), η συνάρτηση αναζήτησης μιας συμβολοσειράς μέσα σε μιαν άλλη, οι συναρτήσεις αποκοπής διαστημάτων από την αρχή και το τέλος συμβολοσειράς, συναρτήσεις μετατροπής γραμμάτων σε κεφαλαία ή πεζά.

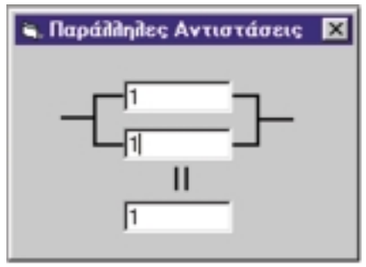
Οι συναρτήσεις μετατροπής τύπων δεδομένων είναι είτε αριθμητικές είτε αλφαριθμητικές συναρτήσεις, που πραγματοποιούν μετατροπές από έναν τύπο μεταβλητής σε άλλο. Ειδικά οι συναρτήσεις μετατροπής των συμβολοσειρών σε αριθμητικές τιμές είναι ιδιαίτερα χρήσιμες για την εκτέλεση πράξεων μεταξύ των τιμών πλαισίων κειμένου.

Οι συναρτήσεις ημερομηνίας και ώρας απομονώνουν τμήμα μιας ημερομηνίας ή ώρας, συνθέτουν μια ημερομηνία ή μια ώρα και χρησιμοποιούνται για την αναφορά της τρέχουσας ημερομηνίας και ώρας.

Η εντολή εκχώρησης `Mid` αντικαθιστά μέρος μιας συμβολοσειράς με μιαν άλλη. Οι εντολές εκχώρησης `Date` και `Time` αλλάζουν την τρέχουσα ημερομηνία και ώρα του υπολογιστικού συστήματος.

Εργαστηριακές Ασκήσεις

1. Γράψτε παράσταση που να υπολογίζει τον ελάχιστο μεταξύ δύο αριθμών Α και Β.
2. Γράψτε πρόγραμμα, που να παρουσιάζει φόρμα με πλαίσιο κειμένου, στο οποίο ο χρήστης θα εισάγει μια γωνία σε μοίρες. Σε άλλα πλαίσια κειμένου να εμφανίζονται οι τριγωνομετρικοί αριθμοί της γωνίας (ημίτονο, συνημίτονο, εφαπτομένη, τέμνουσα, συντέμνουσα, συνεφαπτομένη), όταν ο χρήστης πατά ένα πλήκτρο.
3. Γράψτε πρόγραμμα, που να υπολογίζει τη συνισταμένη δύο δυνάμεων.
4. Δημιουργήστε ένα έργο για την άσκηση 7-1. Στη συνέχεια μετασχηματίστε τον κώδικα και εκτελέστε την άσκηση 7-2.
5. Δημιουργήστε ένα έργο για την άσκηση 7-3, στην οποία γίνεται εξομοίωση ζαριού. Διαμορφώστε το πρόγραμμα, ώστε να εξομοιώνονται ταυτόχρονα δύο ζάρια.
6. Υλοποιήστε την άσκηση 7-4. Φροντίστε να απομακρύνονται τα διαστήματα από την αρχή και το τέλος της πρότασης.
7. Υλοποιήστε την άσκηση 7-5. Στη συνέχεια διαμορφώστε το πρόγραμμα σύμφωνα με τις υποδείξεις της άσκησης 7-6.
8. Δημιουργήστε πρόγραμμα, που να υπολογίζει την ολική αντίσταση δύο παράλληλων αντιστάσεων. Θέστε αρχική τιμή στις αντιστάσεις, την τιμή 1Ω.



9. Δημιουργήστε έργο για την άσκηση 7-7.
10. Δημιουργήστε έργο για την άσκηση 7-8. Διαμορφώστε την άσκηση, ώστε να δίνει την ημερομηνία μετά από x μέρες μιας αρχικής ημερομηνίας.
11. Υλοποιήστε το πρόγραμμα που δίνεται ως εφαρμογή για το ημερολόγιο-χρονόμετρο.

Μάθημα 8

Λογικές παραστάσεις Δομές Επιλογής

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να συντάσσουν απλές και σύνθετες λογικές παραστάσεις.
- Να χρησιμοποιούν τις δομές επιλογής.
- Να επεξηγούν τη διαφορά της δομής `If … Then` από τη δομή `Select Case`.
- Να υιοθετούν κατά περίπτωση την κατάλληλη δομή επιλογής.
- Να δημιουργούν προγράμματα με ένθετες προγραμματιστικές δομές.

Μέχρι τώρα έχουμε γράψει κώδικες που είναι πολύ απλοί, ως προς τον τρόπο εκτέλεσης τους και ως προς τη δομή τους, στους οποίους οι εντολές εκτελούνται η μια μετά την άλλη, ακριβώς με τη σειρά που τις έχουμε γράψει μέσα στις υπορουτίνες συμβάντων. Όμως δεν είναι λίγες οι φορές που απαιτείται να γίνει κάποιος έλεγχος και ανάλογα με το αποτέλεσμά του να επιλέγεται και κάποια αντίστοιχη λειτουργία. Για παράδειγμα, αν μας δώσουν έναν αριθμό και μας ζητήσουν την απόλυτη τιμή του, πρέπει να ελέγξουμε, αν ο αριθμός είναι θετικός, μηδέν ή αρνητικός και στη συνέχεια αν είναι θετικός ή μηδέν αποφαινόμαστε ότι η απόλυτη τιμή του είναι ο ίδιος ο αριθμός, ενώ αν είναι αρνητικός, αποφαινόμαστε ότι η απόλυτη τιμή του είναι ο αντίθετός του.

Στον προγραμματισμό οι έλεγχοι γίνονται σε λογικές παραστάσεις και οι λειτουργίες επιλέγονται από σύνολα γραμμένα μέσα σε προγραμματιστικές δομές επιλογής. Στο εσωτερικό των προγραμματιστικών δομών επιλογής γράφονται κωδικοποιημένες όλες οι λειτουργίες που είναι δυνατόν να ζητηθεί να εκτελεστούν. Κατά την εκτέλεση του προγράμματος, ανάλογα με τα αποτελέσματα των ελέγχων, υποδεικνύεται η εκτέλεση της κατάλληλης λειτουργίας. Η VB διαθέτει δύο βασικές δομές επιλογής, την `If…Then…Else` και τη `Select Case.` Και οι δύο δομές είναι αρκετά ευέλικτες στη σύνταξη και μπορούν να καλύψουν κάθε περίπτωση κωδικοποίησης.

Λογικές Παραστάσεις

Όπως αναφέραμε, η πραγματοποίηση των ελέγχων γίνεται πάνω στην αλήθεια λογικών παραστάσεων. Οι λογικές παραστάσεις είναι παραστάσεις για τις οποίες μπορούμε να αποφανθούμε αν είναι ***αληθείς*** (***true***) ή ***ψευδείς*** (***false***). Συντίθενται από σταθερές, μεταβλητές, αριθμητικές ή αλφαριθμητικές παραστάσεις, ***σύμβολα σύγκρισης*** (=, >, <, <>, >= κ.ά.) και ***λογικούς συνδέσμους*** ΚΑΙ (`And`), Ή (`Or`), ΟΧΙ (`Not`).

Οι λογικές πράξεις μας είναι γνωστές από το μάθημα των ψηφιακών ηλεκτρονικών.

Παραδείγματα 8-1.

Οι παραστάσεις:

```
Total > 172348.45
TypeOfError = 2
A^2 + B^2 >= C^2
Voltage.Enabled = True
Not ReadedValue = False
(LimitDown < X) And (X < LimitDown)
(Professor = "Ìáèçìáôéêüò") Or (Professor ="Öõóéêüò")
```

είναι λογικές παραστάσεις.

Απλές λογικές παραστάσεις

Οι λογικές παραστάσεις μπορεί να είναι απλές ή σύνθετες. Στις απλές λογικές παραστάσεις γίνεται σύγκριση της τιμής της παράστασης που βρίσκεται αριστερά του συμβόλου σύγκρισης με την τιμή της παράστασης που βρίσκεται δεξιά του. Αν το αποτέλεσμα της