ΟΔΗΓΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
στο περιβάλλον της
VISUAL BASIC 6

Θεοφάνης Γεωργάκης
Κωνσταντίνος Κατσάμπαλος

Έκδοση 1.3
e-τοπο, ΕΠΕΑΕΚ-2, 2003-2006
http://web.auth.gr/e-topo

Θεσσαλονίκη
Μάρτιος 2006

Περιεχόμενα

Κεφάλαιο 1

Γνωριμία με το περιβάλλον της Visual Basic

Εισαγωγή

Η Visual Basic (VB), μέλος της ομάδας προγραμμάτων του Microsoft® Visual Studio, αποτελεί τη μετεξέλιξη της παλαιότερης έκδοσής της με το όνομα GW Basic.

Η GW Basic «έτρεχε» σε περιβάλλον MS DOS και η αρχή λειτουργίας της βασίζονταν στη σχεδόν σειριακή εκτέλεση του κώδικα. Η εκτέλεση του κώδικα ξεκινούσε από την πρώτη εντολή και τερματίζονταν στην τελευταία. Η σύνταξη προγραμμάτων με αυτόν τον τρόπο οδηγούσε στην δημιουργία των λεγόμενων «Console Applications», δηλαδή προγραμμάτων τα οποία, εάν δεν υπήρχε αναμονή δεδομένων, συνήθως τερμάτιζαν τη λειτουργία τους. Αυτή είναι και η μεγάλη διαφορά των παλαιότερων εκδόσεων με την VB. Η αρχή λειτουργίας έχει αλλάξει οριστικά. Με την εισαγωγή «αντικειμένων» οδηγηθήκαμε σε μια λειτουργία του τύπου «μην με καλέσεις αν δεν σε καλέσω», δηλαδή κοινώς «μην κάνεις τίποτα αν δεν σου πω», συμπεριλαμβανομένου και του τερματισμού. Ο προγραμματισμός που ακολουθεί αυτή τη γενική λογική ονομάζεται «αντικειμενοστραφής» και έχει ως προτεραιότητα την επέμβαση του χρήστη για την εκτέλεση οποιασδήποτε λειτουργίας. Παρακάτω θα δούμε αναλυτικά πώς επιτυγχάνεται αυτό.

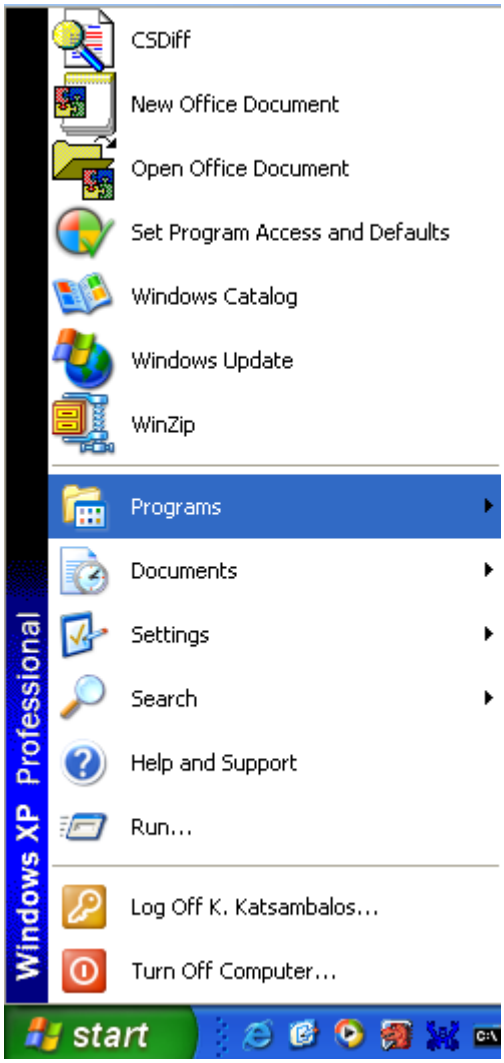
Ένα βασικό χαρακτηριστικό της VB είναι η καθιέρωση της χρήσης της (από τη Microsoft) ως την κατ εξοχήν γλώσσα επικοινωνίας μεταξύ διαφορετικών προγραμμάτων - εφαρμογών (Excel, Word, AutoCAD, Access, κα).

Σημείωση

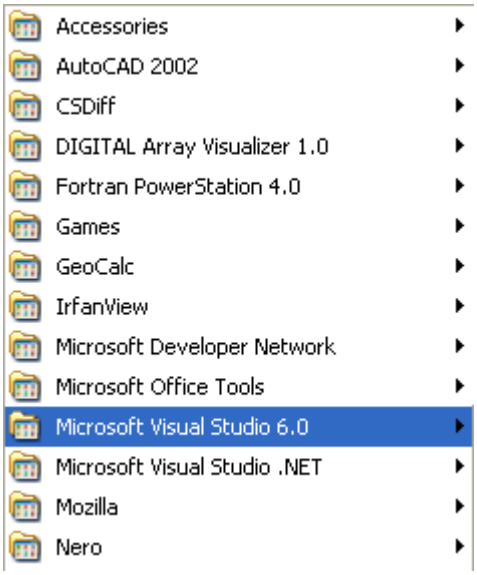
Πέραν από την προαναφερθείσα παλαιότερη έκδοση με το όνομα GW Basic υπήρχαν και άλλες εκδόσεις όπως η Basic A (IBM), η Quick Basic (Microsoft) και η Turbo Basic (Borland).

Εκκίνηση της Visual Basic

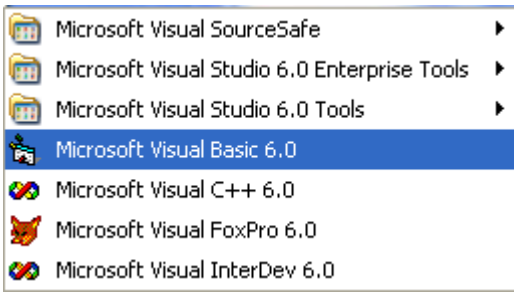
Θεωρώντας ότι έχουμε ήδη εγκατεστημένη τη VB στον υπολογιστή μας, πάμε να εντοπίσουμε το εικονίδιο εκτέλεσής της. Πατώντας πάνω στο "start" ή εναλλακτικά «Έναρξη» (για όσους έχουν ελληνικά windows) θα εμφανιστεί το παρακάτω αναδυόμενο μενού επιλογών.



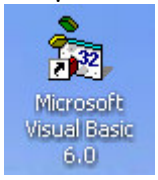
Από το μενού αυτό επιλέγουμε το υπο-μενού «Programs» ή «Προγράμματα», το οποίο μας ανοίγει μια νέα στήλη με τα ονόματα των προγραμμάτων που είναι εγκατεστημένα στον υπολογιστή μας:



Από αυτή τη λίστα, επιλέγουμε την κατηγορία «Microsoft Visual Studio 6.0» από όπου οδηγούμαστε στην τελική επιλογή για τη MS VB6:



Πατώντας στον τελευταίο τίτλο ανοίγει το παράθυρο της Visual Basic. Εναλλακτικά, και εφόσον έχουμε δημιουργήσει κάποια συντόμευση στην επιφάνεια εργασίας, θα μπορούσαμε απλά να κάνουμε διπλό κλικ στο εικονίδιο



της VB.

Περιβάλλον της Visual Basic

Με την εκκίνηση της VB θα εμφανιστεί αμέσως ένας οδηγός επιλογής του τύπου του προγράμματος ή στοιχείου που επιθυμούμε να δημιουργήσουμε.

Στην διπλανή εικόνα φαίνονται οι επιλογές μας. Θα ασχοληθούμε μόνο με το «Standard EXE», δηλαδή με τη δημιουργία ενός ολοκληρωμένου προγράμματος, και όχι με την δημιουργία κάποιου συστατικού.

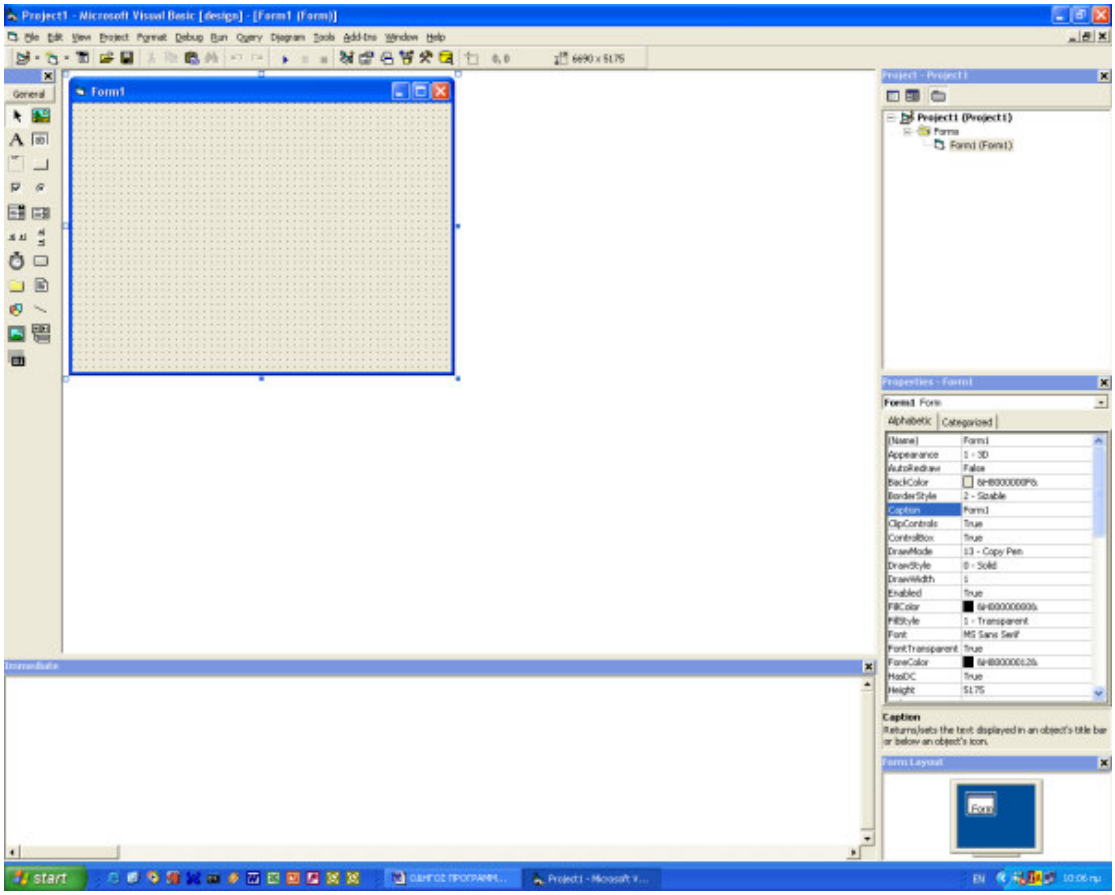
Στην παραπάνω εικόνα, όπως βλέπουμε, υπάρχουν τρεις βασικές επιλογές:

1. New: Επιλέγουμε τον τύπο του προγράμματος που θα δημιουργήσουμε «από το μηδέν».
2. Existing: Ψάχνουμε στον δίσκο να εντοπίσουμε και να ανοίξουμε ένα ήδη δημιουργημένο και αποθηκευμένο πρόγραμμα (πρόσφατο/recent, ή παλαιότερο).
3. Recent: Μας προτείνει μια λίστα που περιέχει τα προσφάτως ανοιγμένα προγράμματα. Μας απαλλάσσει από τον κόπο να ψάχνουμε διαρκώς ένα project το οποίο επεξεργαζόμαστε συχνά.

Σημείωση

1. Συστατικό είναι κάποιο τμήμα ενός project, (module/class module, εργαλείο [ActiveX], βιβλιοθήκη [dll] κτλ).
2. Από εδώ και πέρα, όταν αναφερόμαστε σε κάποιο πρόγραμμα που δημιουργούμε, θα αναφέρουμε τον όρο project που περιλαμβάνει, τόσο το ζητούμενο, όσο και τη λύση, το αποτέλεσμα, τα συστατικά κτλ.

Μόλις επιλέξουμε «Standard EXE» εισερχόμαστε στο κυρίως περιβάλλον της VB:



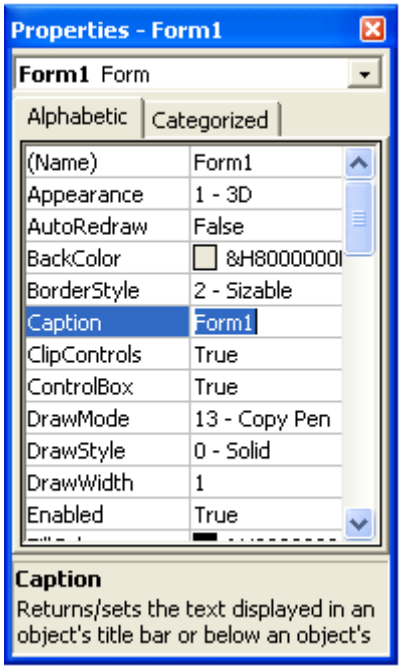
Όπως παρατηρούμε, το περιβάλλον είναι διαιρεμένο σε υπο-περιοχές. Κάθε μια απ' αυτές, έχει έναν συγκεκριμένο ρόλο στην υποβοήθηση του προγραμματιστή για τη σύνταξη ενός ολοκληρωμένου project. Ας τις δούμε αναλυτικά:

ToolBox (Γραμμή εργαλείων)

Βρίσκεται στα αριστερά της κεντρικής οθόνης. Περιλαμβάνει τα «στοιχεία ελέγχου» πάνω στα οποία βασίζεται το «χτίσιμο» ενός project. Τα αντικείμενα αυτά καθοδηγούν τον χρήστη στη σύνταξη ενός λογισμικού. Σε αυτά οφείλεται ο όρος «αντικειμενοστραφής προγραμματισμός».

Η γραμμή εργαλείων περιλαμβάνει τα βασικά στοιχεία ελέγχου. Ωστόσο, ο χρήστης μπορεί εύκολα να προσθέσει επιπλέον όσα προαιρετικά εργαλεία επιθυμεί. Θα δείξουμε πώς γίνεται αυτό παρακάτω.

Properties Window (Παράθυρο ιδιοτήτων)

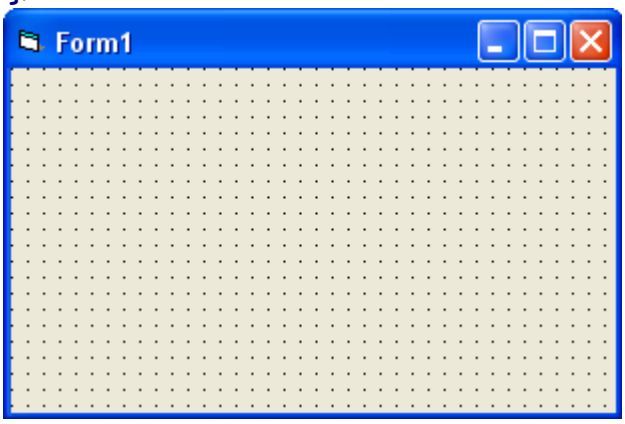


Όλα τα στοιχεία ελέγχου έχουν ιδιότητες. Αυτές είναι παράμετροι που σχετίζονται με την εμφάνιση ενός αντικειμένου (χρώμα, γραμματοσειρά, τίτλος κτλ), με τη θέση του αντικειμένου σε μια φόρμα (απόσταση από το αριστερό και το πάνω άκρο της φόρμας), με το αν είναι προσβάσιμα από το χρήστη, αν είναι ορατά, με τον τύπο εμφάνισής τους και πολλά άλλα.

Τα στοιχεία ελέγχου, εκτός από τις ιδιότητες, έχουν συμβάντα (events) και μεθόδους (methods).

Σημείωση

1. Φόρμα (Form) είναι εκείνο το αντικείμενο πάνω στο οποίο «χτίζουμε» το project μας, το «χαρτί» πάνω στο οποίο θα σχεδιάσουμε τη μορφή του προγράμματός μας.

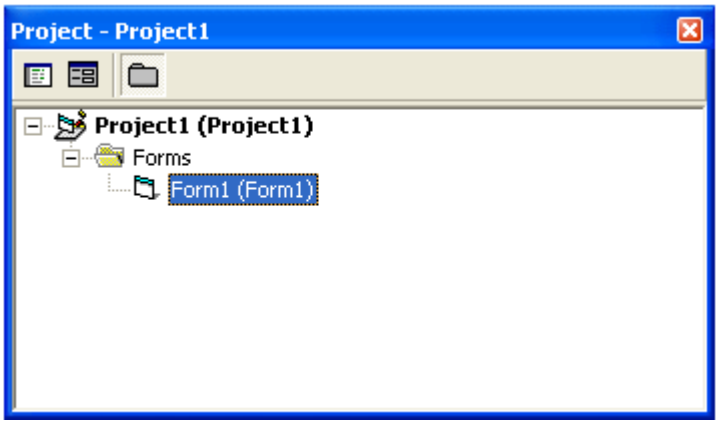


Πρόκειται για την κεντρική μονάδα εργασίας την οποία διαμορφώνουμε οπτικά με τα αντικείμενα που προσθέτουμε επάνω της. Χωρίς τη φόρμα δεν εννοείται αντικειμενοστραφής προγραμματισμός. Πρόγραμμα (με τη στενή έννοια του όρου) χωρίς φόρμα δεν υφίσταται. Μόνο τα ειδικά (καθαρά υπολογιστικά) συστατικά (π.χ. DLL) δεν απαιτούν τη χρήση μιας φόρμας.

2. Συμβάντα ονομάζονται όλες εκείνες οι ενέργειες οι οποίες γίνονται από τον χρήστη κατά τη χρήση του προγράμματος. Για παράδειγμα, συμβάν για ένα κουμπί (button) το οποίο προσθέσαμε σε μια φόρμα, είναι το «κλικ» του ποντικιού, το «δεξί κλικ», το πέρασμα του ποντικιού από πάνω κτλ. Είναι λοιπόν ενέργειες του χρήστη στις οποίες εμείς προσθέτουμε κάποιες λειτουργίες.

3. Μέθοδοι ονομάζονται κάποιες λειτουργίες οι οποίες είναι κατά το πλείστον προαποφασισμένου αποτελέσματος, ενώ εμείς προσθέτουμε απλά μια παράμετρο. Για παράδειγμα, μια ενδεικτική μέθοδος για το button είναι η «move». Το τι θα κάνει είναι προαποφασισμένο. Θα μετακινήσει το κουμπί. Το πού όμως θα μετακινηθεί, το δίνει ο συντάκτης ή ο χρήστης του προγράμματος.

Project Explorer (Εξερευνητής)

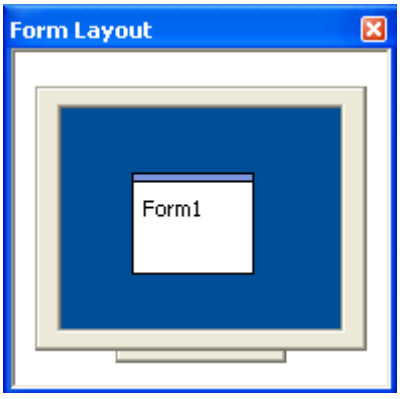


Ο Project Explorer μας δείχνει από ποια συστατικά αποτελείται το project που φτιάχνουμε και μας επιτρέπει την μετάβαση από το ένα στο άλλο.
Περιεχόμενα του εξερευνητή είναι οι φόρμες, τα modules, τα class modules κτλ.

Σημείωση

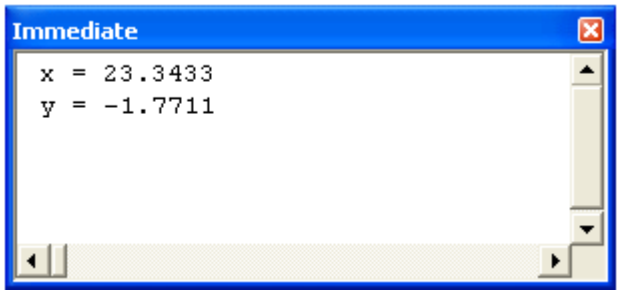
Module ονομάζεται ένα αυτόνομο τμήμα κώδικα το οποίο δεν περιέχει γραφικό περιβάλλον. Δεν ανήκει σε μια φόρμα, έχει δικό του όνομα, αποθηκεύεται ξεχωριστά και είναι κατάλληλο για χρήση σε πολλά προγράμματα. Ένα module περιέχει συνήθως δηλώσεις τύπων και μεταβλητών, συναρτήσεις και υπορουτίνες.
Class Module είναι ένα προωθημένο χαρακτηριστικό της VB. Όταν ένα class module «μεταγλωττίζεται» (compilation & build), δημιουργείται μια βιβλιοθήκη DLL. Αποτελεί μια αυτόνομη ομάδα διαδικασιών και συναρτήσεων η οποία μπορεί να δουλέψει ξεχωριστά από το υπόλοιπο project. *Στο **παράρτημα-Γ** θα δούμε τον τρόπο σύνταξη, κατασκευής και χρήσης μιας βιβλιοθήκης DLL (Dynamically Linked Library).*

Form Layout (Επισκόπηση φορμών)



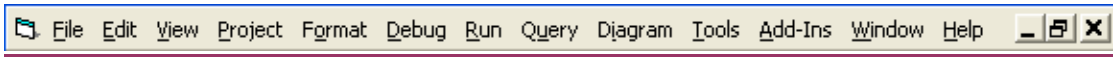
Το παράθυρο αυτό βρίσκεται κάτω δεξιά της κύριας οθόνης. Μας παρουσιάζει μια επισκόπηση για τη θέση στην οποία θα εμφανιστεί η κάθε φόρμα κατά την εκτέλεση του προγράμματος. Μπορούμε να μετακινήσουμε κάθε φόρμα μέσα στην οθόνη, έτσι ώστε να εμφανίζεται εκεί που θέλουμε.

Immediate Window (Παράθυρο άμεσης εκτύπωσης)

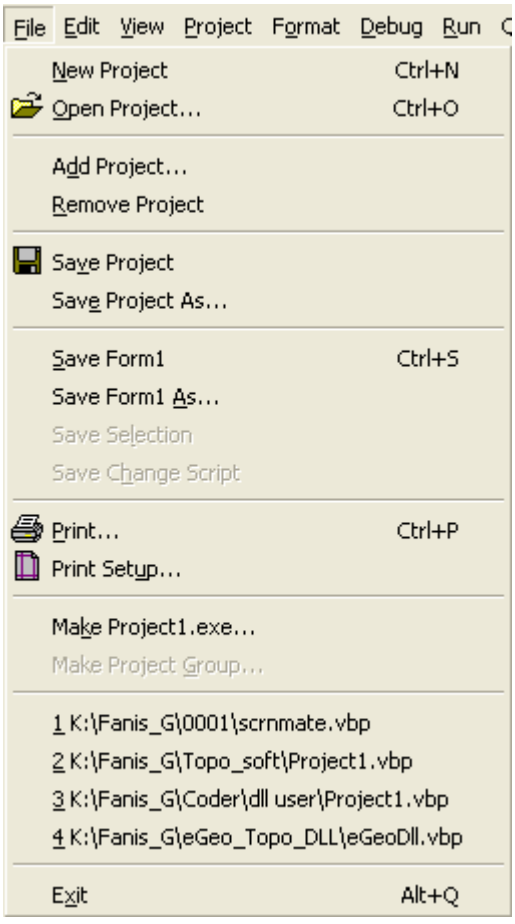


Το παράθυρο αυτό βρίσκεται στο κάτω και μέσο του περιβάλλοντος εργασίας. Είναι πολύ χρήσιμο για να αποσφαλματοποιούμε (debuging) ένα project ή να εκτυπώνουμε πρόχειρα αποτελέσματα για έλεγχο.

Menu (Μενού επιλογών)



Το μενού επιλογών βρίσκεται στην κορυφή του περιβάλλοντος εργασίας και περιλαμβάνει ταξινομημένες όλες τις λειτουργίες της VB. Ας δούμε τις πιο ενδεικτικές κατά κατηγορία:



File

Δημιουργία νέου Project
Άνοιγμα υπάρχοντος Project

Αποθήκευση project
Αποθήκευση project ως...(όνομα)

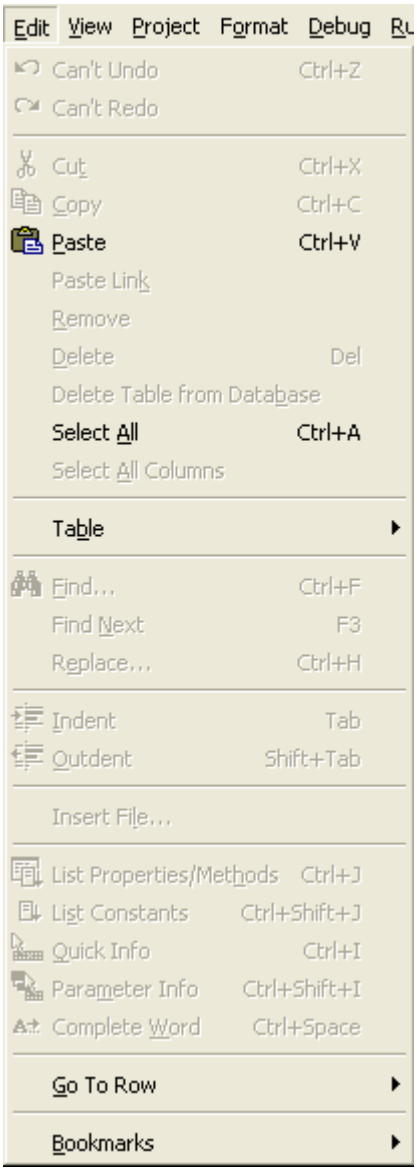
Αποθήκευση φόρμας

Εκτύπωση

Δημιουργία EXE οπό project (δημιουργία αυτοτελώς εκτελέσιμου αρχείου)

Άνοιγμα πρόσφατων project

Έξοδος από τη VB



Edit

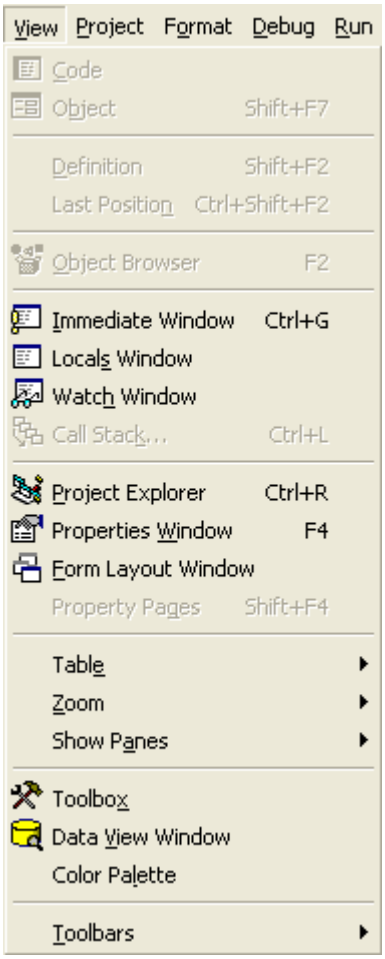
Undo / Redo τελευταίας λειτουργίας

Αποκοπή
Αντιγραφή
Επικόλληση

Διαγραφή

Επιλογή όλων

Εύρεση
Εύρεση επόμενου
Αντικατάσταση



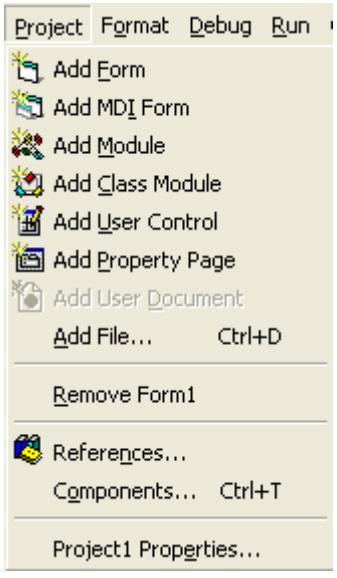
View
Κώδικα
Αντικείμενο

Εμφάνιση του Immediate Window

Εμφάνιση του Project Explorer
Εμφάνιση του Properties Window

Εμφάνιση του ToolBox

Εμφάνιση επιλεγμένων ToolBars



Project
Προσθήκη φόρμας

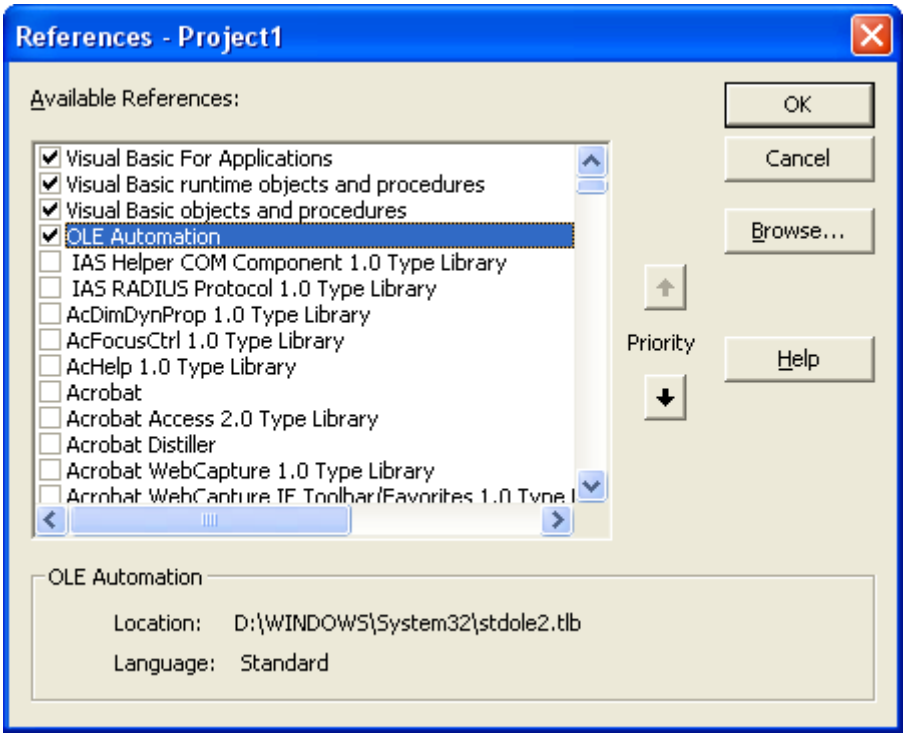
Προσθήκη Module
Προσθήκη Class Module

Αφαίρεση φόρμας
Αναφορές
Στοιχεία ελέγχου

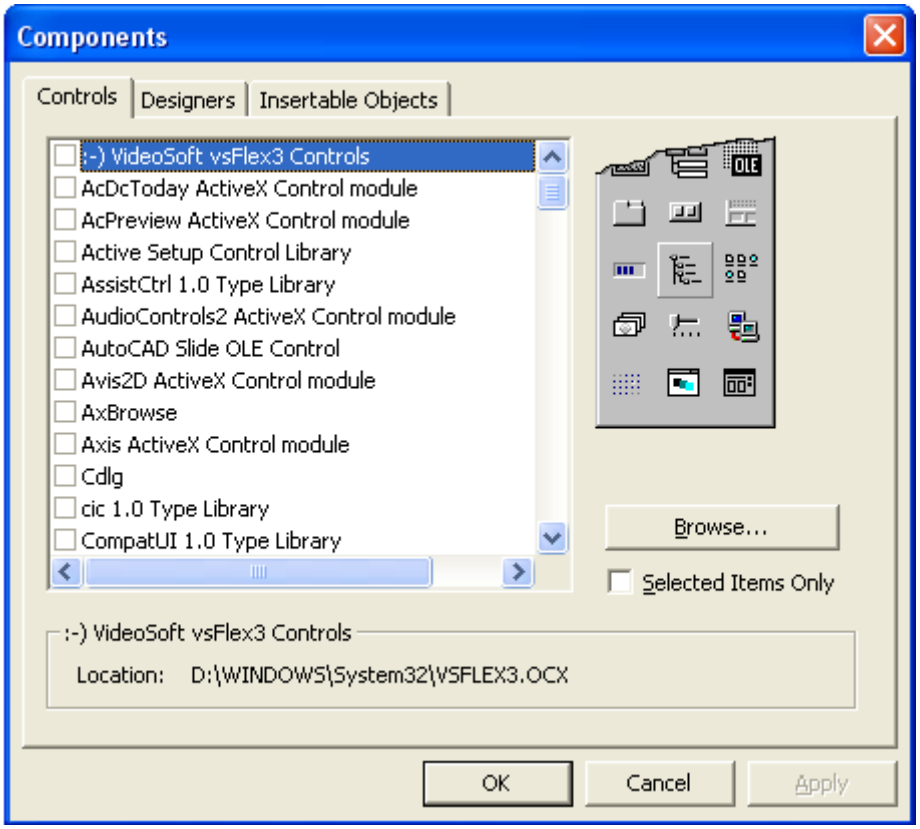
Ιδιότητες του Project μας

Σημείωση

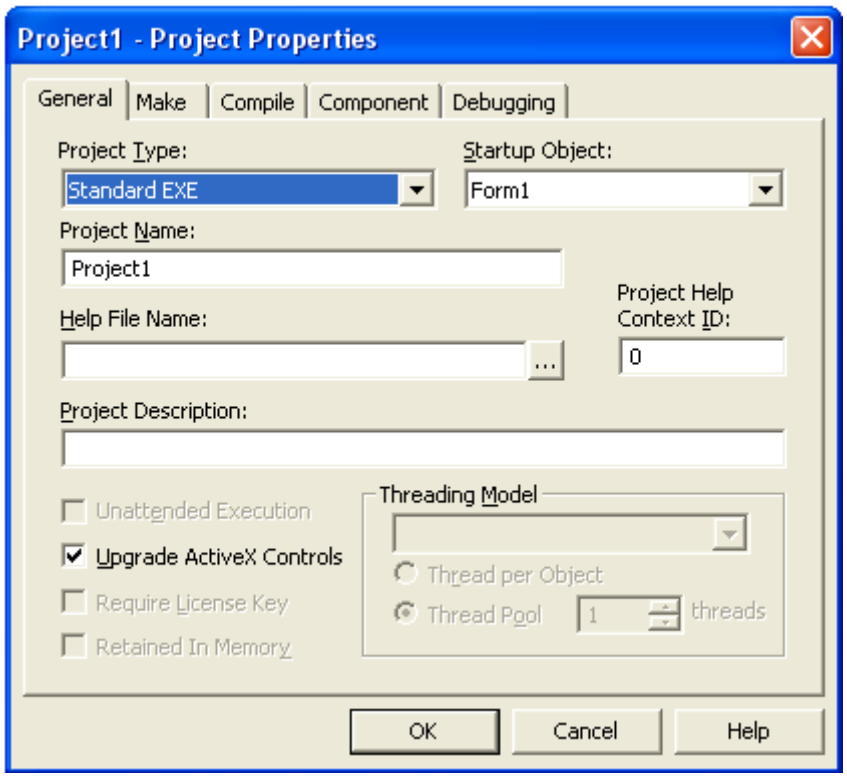
Αναφορές (References): Μας επιτρέπει να συνδέσουμε το project που δημιουργούμε με μια εξωτερική βιβλιοθήκη ή κάποιο άλλο λογισμικό. Θα αναφερθούμε λεπτομερέστερα στο παράρτημα-Γ.

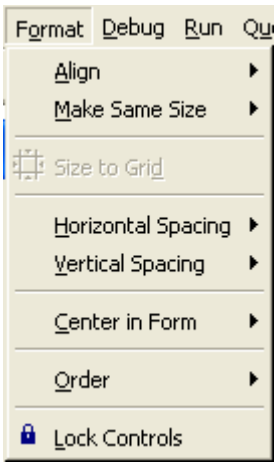


Στοιχεία ελέγχου (Components): Πέρα από τα στοιχεία ελέγχου που υπάρχουν στο ToolBox όταν ανοίγουμε την VB, υπάρχουν πάρα πολλά ακόμα τα οποία, είτε έχει φτιάξει η Microsoft, είτε τρίτοι κατασκευαστές. Κάποια από αυτά θα μας είναι απαραίτητα στην πορεία όπου και θα αναλύσουμε τον τρόπο εισαγωγής τους.



Ιδιότητες του Project (Project Properties): Είναι γενικές ιδιότητες που αφορούν το project μας όπως το όνομα, η έκδοση, το όνομα του αρχείου βοηθείας, τα σχόλια του κατασκευαστή κτλ. Προτείνεται να συμπληρώνονται εξ αρχής, διότι μας προσφέρουν πολύ χρήσιμες πληροφορίες κατά την ανάπτυξη του προγράμματος.



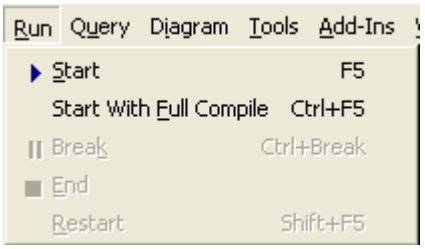


Format

Τακτοποίηση αντικειμένου στη φόρμα
Ίδιο μέγεθος σε 2 αντικείμενα

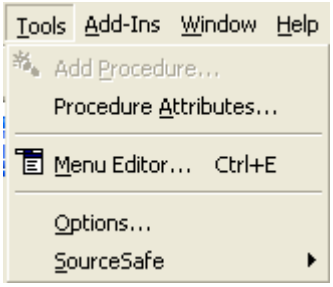
Πύκνωση/αραίωση οριζόντια ή κάθετα διαστημάτων.

Κεντράρισμα στην φόρμα
Τακτοποίηση στοιχείων οπτικά
Κλείδωμα εργαλείων



Run
Εκτέλεση / συνέχεια

Παύση
Τερματισμός

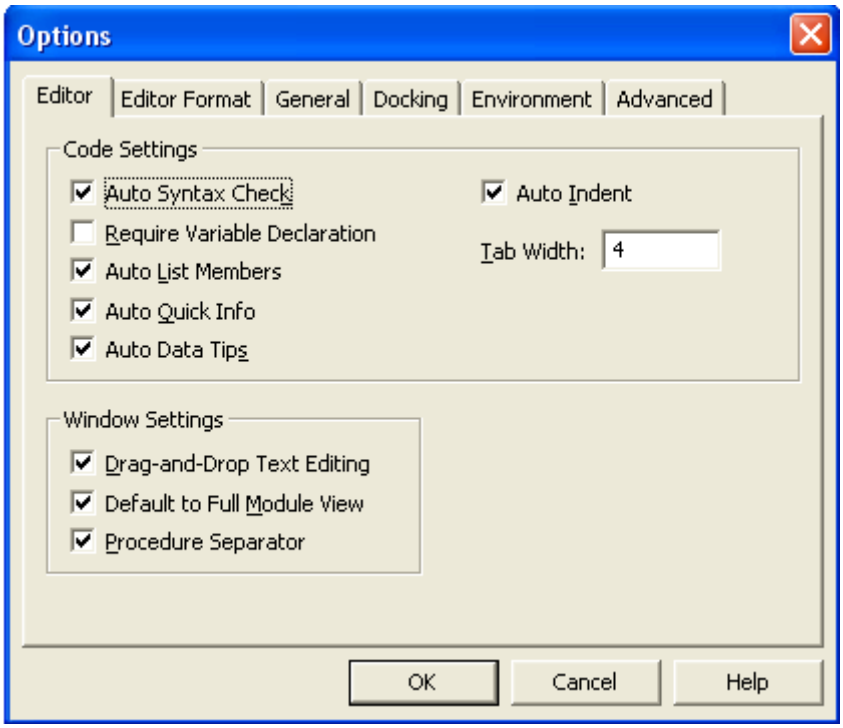


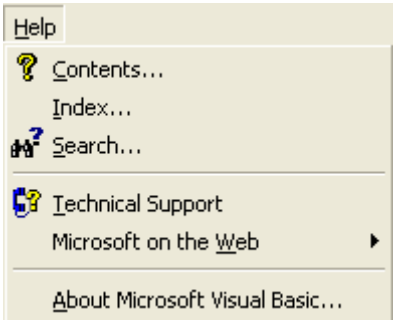
Tools

Δημιουργία/επεξεργασία μενού
Επιλογές

Σημείωση

Επιλογές (Options): Πρόκειται για έναν πίνακα γενικών επιλογών που αφορούν όλο το studio της Visual Basic. Από εκεί ρυθμίζεται ο κάνναβος, οι γραμματοσειρές, τα χρώματα, η διαδικασία της αυτόματης συμπλήρωσης κώδικα, η διαδικασία αυτόματης αποθήκευσης και πολλά άλλα.





Help
Περιεχόμενα
Εύρεση με βάση τον όρο
Εύρεση

Η Microsoft στο διαδίκτυο

Η βοήθεια (Help) της Visual Basic είναι πραγματικά πολύ ισχυρή. Διατίθεται και ξεχωριστά από το Visual Studio με το όνομα MSDN. Αποτελεί το καλύτερο βοήθημα, προσφέροντας συμβουλές, δείγματα κώδικα, σύνταξη εντολών κλπ.

Αυτό είναι συνοπτικά το περιβάλλον της Visual Basic. Αναφέραμε παραπάνω τις επιλογές που θεωρούμε ότι είναι πιο χρήσιμες κατά τη δημιουργία του project.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Ο αντικειμενοστραφής προγραμματισμός στηρίζεται στην ύπαρξη αντικειμένων (στοιχείων ελέγχου) τα οποία υπάρχουν στο ToolBox. Αυτά τα τοποθετούμε πάνω στη φόρμα η οποία είναι ο μοναδικός αποδέκτης τέτοιων στοιχείων. Όλα τα στοιχεία ελέγχου έχουν ιδιότητες, συμβάντα και μεθόδους.
    a. Οι ιδιότητες προσδιορίζουν κάποια φυσικά χαρακτηριστικά.
    b. Τα συμβάντα προσομοιώνουν τις ενέργειες του χρήστη.
    c. Οι μέθοδοι είναι εσωτερικά δημιουργημένες εντολές που περιμένουν από εμάς κάποιο όρισμα.
2. Η φόρμα είναι ένα στοιχείο ελέγχου. Είναι το βασικό στοιχείο και χωρίς τουλάχιστον μια φόρμα δεν δημιουργείται πρόγραμμα (EXE).
3. Μέσα από το Properties Window μπορούμε να αλλάξουμε τις ιδιότητες κάποιου στοιχείου ελέγχου.

ΕΡΩΤΗΣΕΙΣ

1. Πώς μπορώ να εισάγω μια δεύτερη φόρμα στο project μου;
2. Από πού μπορώ να αλλάξω τη θέση εμφάνισής της;
3. Από πού προσθέτω περισσότερα στοιχεία ελέγχου στο ToolBox;
4. Από πού μπορώ να αλλάξω γραμματοσειρά;
5. Πώς αλλάζω το όνομα στο project μου;
6. Από πού αποθηκεύω μια φόρμα;
7. Πώς μπορώ να μετακινηθώ από μια φόρμα σε μια άλλη;
8. Τι είναι το module;
9. Πώς μπορώ να «φορτώσω» ένα project που αποθήκευσα πρόσφατα;
10. Πού μπορώ να βρω πληροφορίες για τη σύνταξη μιας εντολής;
11. Πώς «τρέχω» ένα πρόγραμμα;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

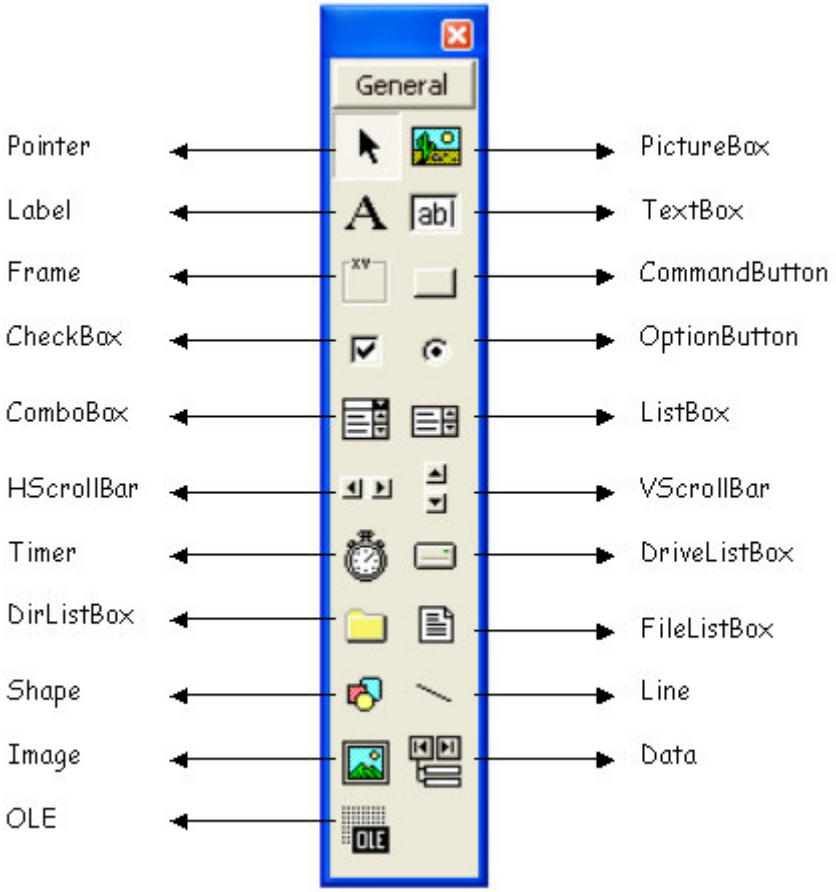
1. Αλλάξτε το φόντο (χρώμα) της φόρμας. Γράψτε σαν τίτλο τη φράση «Εισαγωγική φόρμα». Δώστε στη φόρμα διαστάσεις: πλάτος 2000, ύψος 1500. Προσθέστε ένα κουμπί στο κέντρο της φόρμας. Δώστε του τον τίτλο «Press me!». Εκτελέστε το πρόγραμμα.

Κεφάλαιο 2

Στοιχεία ελέγχου, χρήση, ιδιότητες, συμβάντα, μέθοδοι

Έχουμε ήδη αναφέρει γενικά τι κάνουν τα στοιχεία ελέγχου (Controls). Είναι τα αντικείμενα εκείνα με τα οποία ο χρήστης εκτελεί κάποιες διαδικασίες και λειτουργεί το πρόγραμμα. Για παράδειγμα, προσθέτουμε εμείς ένα κουμπί (Command Button) στη φόρμα μας και στο συμβάν «κλικ» γράφουμε κάποιο κώδικα (πχ να εμφανίζει ένα μήνυμα χαιρετισμού). Ο χρήστης γνωρίζει όταν βλέπει ένα κουμπί ότι πρέπει να το πατήσει για να κάνει κάτι. Όταν όμως υπάρχουν πολλά κουμπιά; Το πρόβλημα λύνεται δίνοντας έναν περιγραφικό τίτλο στο κουμπί που λέει στο χρήστη τι κάνει. Για παράδειγμα, θα μπορούσε να γράφει «Έξοδος από το πρόγραμμα». Ο χρήστης καταλαβαίνει ότι αν το πατήσει, θα τερματιστεί το πρόγραμμα. Αυτός ο τίτλος είναι μια ιδιότητα!

Παρακάτω θα δούμε αναλυτικά κάποια στοιχεία ελέγχου συνοδευόμενα από ορισμένες βασικές ιδιότητές τους, συμβάντα και μεθόδους.



Πριν προχωρήσουμε στην ανάλυση των στοιχείων ελέγχου, θα πούμε πρώτα τους δυνατούς τρόπους με τους οποίους μπορούμε να εισάγουμε οποιοδήποτε στοιχείο ελέγχου επάνω σε μια φόρμα.

α) Με διπλό κλικ επάνω στο αντίστοιχο εικονίδιο στη γραμμή εργαλείων.

β) Επιλέγοντας το αντίστοιχο εικονίδιο (με απλό κλικ) και σχεδιάζοντάς το επάνω στη φόρμα.

γ) Με προγραμματιστικές τεχνικές (προχωρημένη τεχνική, εκτός των στόχων του παρόντος οδηγού).

Pointer (Δείκτης): Δεν πρόκειται για στοιχείο ελέγχου, αλλά για δείκτη επιλογής. Δεν έχει ιδιότητες, συμβάντα και μεθόδους. Είναι το προεπιλεγμένο εργαλείο μετά από κάθε ενέργεια.

Label (Ετικέτα):

Πρόκειται για ένα απλό εργαλείο το οποίο χρησιμοποιούμε για να γράψουμε κάτι (μια λεζάντα / caption). Κατά την εκτέλεση του προγράμματος δεν μπορεί ο χρήστης να επέμβει στο κείμενό της.

Οι ιδιότητες (οι οποίες είναι προσβάσιμες από το Properties Window) παρουσιάζονται ταξινομημένες στον παρακάτω πίνακα.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name)	Προσδιορίζει το πραγματικό όνομα αναφοράς της Label.	Οποιοδήποτε όνομα με λατινικούς χαρακτήρες (μπορεί να περιλαμβάνει και αριθμό στο τέλος) το οποίο δεν είναι **δεσμευμένο** όνομα της Visual Basic.	Όλα τα συμβάντα που προγραμματίζουμε, αναφέρονται σε αυτό το όνομα.
Alignment	Προσδιορίζει τη θέση της λεζάντας μέσα στην Label	0 ή 1 ή 2	0: Η λεζάντα αριστερά 1: Η λεζάντα δεξιά 2: Η λεζάντα στη μέση
AutoSize	Προσδιορίζει αν το μέγεθος της Label είναι ακριβώς όσο της αναγραφόμενης λεζάντας	True ή False	True: Μέγεθος Label = μέγεθος λεζάντας False: Μέγεθος Label = όσο το ορίζουμε εμείς.
BackColor	Προσδιορίζει το χρώμα-φόντο της	Οποιαδήποτε έγκυρη τιμή χρώματος (από	Αλλάζει το χρώμα-φόντο της Label

	Label.	την αναδιπλούμενη λίστα επιλογών) ή εναλλακτικά: VbRed, vbGreen, vbCyan, vbYellow, vbWhite κτλ	ανάλογα με το όρισμα που του δίνουμε.
BackStyle	Προσδιορίζει αν η Label είναι διαφανής (και επιτρέπει την ανάδειξη της φόρμας) ή όχι	0 ή 1	0: (Διαφανής) 1: (Συμπαγής)
BorderStyle	Προσδιορίζει αν θα υπάρχει περίγραμμα στην Label ή όχι.	0 ή 1	0: Χωρίς περίγραμμα 1: Με περίγραμμα
Caption	Η λεζάντα που θα δείχνει η Label	Οποιοδήποτε κείμενο, αριθμός, σύμβολο κτλ και οποιοσδήποτε συνδυασμός των παραπάνω.	Μέσα στην Label βλέπουμε ότι έχουμε δώσει ως Caption.
Enabled	Προσδιορίζει αν η Label θα είναι αποδέκτης συμβάντων.	True ή False	True: Θα είναι αποδέκτης συμβάντων (π.χ. θα μπορούμε να κάνουμε «κλικ» πάνω της για να ενεργοποιήσουμε κάποιο συμβάν) False: Δεν θα είναι αποδέκτης συμβάντων. Η Label θα παρουσιάζεται «γκριζαρισμένη» δηλαδή απενεργοποιημένη.
Font	Προσδιορίζει τον τύπο της γραμματοσειράς που θα χρησιμοποιηθεί για την παρουσίαση του Caption (λεζάντα)	Οποιαδήποτε έγκυρη γραμματοσειρά από την σχετική αναδιπλούμενη λίστα επιλογών. Επίσης από τη σχετική λίστα καθορίζουμε στυλ (πλάγια, έντονα κτλ) καθώς και μέγεθος γραμμάτων.	Η εμφανιζόμενη λεζάντα θα εμφανίζεται στο στυλ της επιλεγμένης γραμματοσειράς.
ForeColor	Προσδιορίζει το χρώμα των γραμμάτων της λεζάντας.	Οποιαδήποτε έγκυρη τιμή χρώματος (από την αναδιπλούμενη λίστα επιλογών) ή εναλλακτικά: VbRed, vbGreen, vbCyan, vbYellow κτλ	Αλλάζει το χρώμα της λεζάντας σε αυτό που επιλέξαμε.
Height	Το ύψος του πλαισίου της Label σε **twip**. Δεν έχει νόημα αν δώσουμε προηγουμένως στην ιδιότητα AutoSize την τιμή True.	Οποιαδήποτε θετική ακέραια τιμή. Αρκεί να χωράει στην φόρμα.	Αλλάζει το ύψος της Label (όχι και της περιεχόμενης λεζάντας) στην τιμή που δώσαμε.

Left	Προσδιορίζει την απόσταση της Label οπό το αριστερό άκρο της φόρμας σε twip.	Οποιαδήποτε θετική ακέραια τιμή.	Τοποθετεί την Label σε όση απόσταση του ορίσουμε από το αριστερό άκρο της φόρμας.
MousePointer	Προσδιορίζει τον τύπο του «βέλους» του ποντικιού όταν αυτό θα περνάει πάνω από την Label.	Οποιαδήποτε έγκυρη τιμή Pointer (από την αναδιπλούμενη λίστα επιλογών)	Όταν ο δείκτης βρίσκεται πάνω από τη Label αλλάζει στην συγκεκριμένη μας επιλογή.
ToolTipText	Ορίζουμε ένα κείμενο το οποίο θέλουμε να εμφανίζεται όταν ο δείκτης του ποντικιού «στέκεται» πάνω από τη Label.	Οποιοδήποτε κείμενο, αριθμός, σύμβολο κτλ και οποιοσδήποτε συνδυασμός των παραπάνω.	Το κείμενο που ορίζουμε εμφανίζεται μέσα σε ένα κίτρινο πλαίσιο (σαν σημείωση / συμβουλή)
Top	Προσδιορίζει την απόσταση της Label οπό το πάνω άκρο της φόρμας σε twip.	Οποιαδήποτε θετική ακέραια τιμή.	Τοποθετεί την Label σε όση απόσταση του ορίσουμε από το πάνω άκρο της φόρμας.
Visible	Προσδιορίζει αν η Label θα είναι ορατή ή αόρατη στο χρήστη.	True ή False	True: Είναι ορατή False: Δεν είναι ορατή
Width	Το πλάτος του πλαισίου της Label σε twip. Δεν έχει νόημα αν δώσουμε προηγουμένως στην ιδιότητα AutoSize την τιμή True.	Οποιαδήποτε θετική ακέραια τιμή. Αρκεί να χωράει στην φόρμα.	Αλλάζει το πλάτος της Label (όχι και της περιεχόμενης λεζάντας) στην τιμή που δώσαμε.
WordWrap	Προσδιορίζει αν η Label θα αναδιπλώνει το κείμενο σε περισσότερες σειρές αν δεν χωράει σε μια.	True ή False	True: Αναδιπλώνει το κείμενο. False: Εμφανίζει όσο κείμενο χωράει μέσα στην Label και δεν το αναδιπλώνει.

Σημείωση

Δεσμευμένο όνομα (reserved word) στην Visual Basic είναι κάθε λέξη, όρος, εντολή, τελεστής, έκφραση ή κατοχυρωμένη συμβολοσειρά η οποία χρησιμοποιείται από την VB για κάποια εσωτερική εργασία. Όλες οι εντολές, οι τελεστές κτλ που θα παρουσιάσουμε σε επόμενα κεφάλαια είναι δεσμευμένες και δεν μπορούμε να χρησιμοποιήσουμε κάποια από αυτές για διαφορετική χρήση (όπως πχ για την ιδιότητα (Name) σε κάποιο στοιχείο ελέγχου).

Twip είναι μια μονάδα μέτρησης μήκους που χρησιμοποιεί η Visual Basic (text width in pixels). Είναι η προεπιλεγμένη μονάδα, ωστόσο μπορεί να αλλάξει από μια σχετική ιδιότητα της φόρμας που τη φέρει, όπως θα δούμε παρακάτω. Η μονάδα που θα επιλέξουμε στην φόρμα μεταφέρεται σε όλα τα στοιχεία ελέγχου που περιλαμβάνει.

Οι παραπάνω ιδιότητες που αναφέρθηκαν είναι ενδεικτικές (οι πιο σημαντικές). Πολλές από αυτές είναι κοινές για τα περισσότερα στοιχεία ελέγχου, οπότε θα αναφέρονται σχετικά, χωρίς περαιτέρω επεξήγηση.

Πέραν από τις ιδιότητες των στοιχείων ελέγχου έχουμε και τα συμβάντα:

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω **υπορουτίνας συμβάντος (Sub)**. (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «Label1»)
Αλλαγή Change	Όταν αλλάζει το περιεχόμενο της Label (δηλαδή η Caption)	Private Sub Label1_Change() End Sub
Απλό «κλικ» Click	Όταν κάνουμε κλικ επάνω στην Label.	Private Sub Label1_Click() End Sub
Διπλό «κλικ» DblClick	Όταν κάνουμε διπλό κλικ πάνω στην Label	Private Sub Label1_DblClick() End Sub
Πάτημα και κράτημα πλήκτρου ποντικιού MouseDown	Όταν κάνουμε κλικ και κρατάμε πατημένο το πλήκτρο.	Private Sub Label1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single) End Sub
Απελευθέρωση πλήκτρου ποντικιού (μετά την παραπάνω ενέργεια) MouseUp	Όταν αφού έχουμε κρατήσει πατημένο το πλήκτρο του ποντικιού, το ελευθερώνουμε.	Private Sub Label1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single) End Sub
Κίνηση του ποντικιού από πάνω MouseMove	Όταν απλά περνάμε το δείκτη του ποντικιού πάνω από την Label.	Private Sub Label1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single) End Sub

Σημείωση

Υπορουτίνα συμβάντος (sub) : Υπορουτίνα (subroutine) είναι ένα τμήμα κώδικα το οποίο εκτελείται όταν εμείς με κάποιο τρόπο καλέσουμε την υπορουτίνα. Όλα τα συμβάντα συνοδεύονται από τέτοιες υπορουτίνες. Για παράδειγμα, όταν κάνουμε «κλικ» σε ένα κουμπί που γράφει «Έξοδος από το πρόγραμμα» (ναι!, είναι η ιδιότητα Caption), εκτελείται αυτόματα το τμήμα του κώδικα που έχουμε ορίσει στο συμβάν του κουμπιού Click, το οποίο «λέει» στο πρόγραμμα να τερματίσει την λειτουργία του. Θα δούμε αναλυτικά τις υπορουτίνες σε επόμενο κεφάλαιο, όπου θα αρχίσουμε να συντάσσουμε τα πρώτα απλά προγράμματα.

Για την ώρα κρατήστε στο μυαλό σας ότι κάθε συμβάν ενεργοποιεί την αποκλειστική υπορουτίνα που συνοδεύει την εκάστοτε ενέργεια, και ότι το όνομα κάθε στοιχείου ελέγχου περιλαμβάνεται στο όνομα της υπορουτίνας (όπως π.χ. βλέπετε στον παραπάνω πίνακα των συμβάντων, όπου το όνομα είναι «Label1»).

Τέλος, θα δούμε τρεις από τις μεθόδους που συνοδεύουν το Label, παρέχοντας οδηγίες για τη χρήση κάθε μιας.

Μέθοδος	Επεξήγηση	Ορίσματα	Σύνταξη (Θεωρώντας ότι το όνομα της Label είναι Label1)
Move	Μετακινεί (και προαιρετικά επαναδιαστασιολογεί) την Label σε μια νέα θέση.	Left, Top, (Width, Height) τα οποία είναι οι παραπάνω ιδιότητες.	Label1.Move 100,200 ή Label1.Move 100,200,400,400
Refresh	Κάνει «ανανέωση» στην Label		Label1.Refresh
ZOrder	Προσδιορίζει το αντικείμενο που θα είναι οπτικά «από πάνω»	0: (μπροστά από όλα) 1: (πίσω από όλα)	Label1.Zorder = 0 Label1.Zorder = 1

ΠΡΟΣΟΧΗ ! ! !

Τώρα που έχουμε μια εποπτική άποψη για το τί είναι μια ιδιότητα, ένα συμβάν και μια μέθοδος, και έχοντας μελετήσει το στοιχείο ελέγχου «Label», θα δούμε πως συντάσσονται μέσα από ένα παράδειγμα.

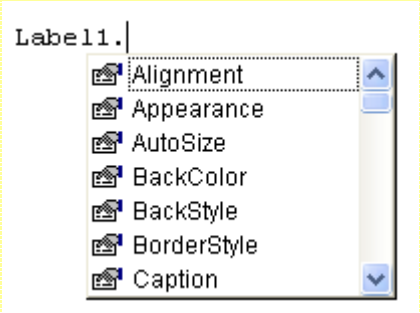
Παράδειγμα

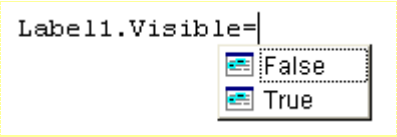
Θα αλλάξουμε τη θέση και το μέγεθος του Label1. Η διαδικασία αυτή θα είναι μια υπορουτίνα η οποία θα περιέχεται στο συμβάν "Click" της ίδιας της Label.

```
Private Sub Label1_Click()
'Δίνω την τιμή "Hallo" στην ιδιότητα Caption
        Label1.Caption = "Hallo"
'Θεση (από αριστερά και πάνω)
        Label1.Left = 500
        Label1.Top = 500
'Μέγεθος (πλάτος και ύψος)
        Label1.Width = 250
        Label1.Height = 200
End Sub
```

Το σύμβολο « ' » (Remark) «λέει» στην Visual Basic να μην εκτελέσει τις συγκεκριμένες σειρές. Οι γραμμές που αρχίζουν με αυτό το σύμβολο ονομάζονται «ΣΧΟΛΙΑ». Σ' αυτές μπορούμε να γράψουμε οτιδήποτε. Για τους καλούς προγραμματιστές τα REMARKS που χρησιμοποιούν για να περιγράψουν με λόγια αυτά που γράφουν με κώδικα είναι σήμα κατατεθέν.

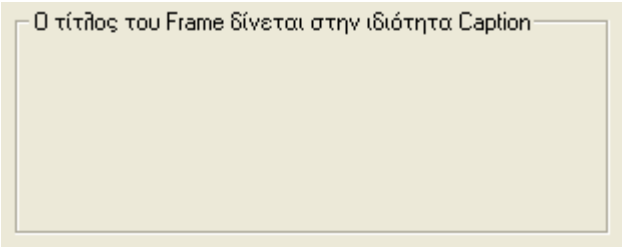
Η τελεία « . » που χρησιμοποιούμε ανάμεσα στο όνομα στου στοιχείου (εδώ Label1) «λέει» στην Visual Basic ότι θα ακολουθήσει ιδιότητα ή μέθοδος. Δεν είναι απαραίτητο να γνωρίζετε τις ιδιότητες και τις μεθόδους. Καθώς πληκτρολογείτε Label1 και πατάτε την « . » εμφανίζεται ένας οδηγός που θα σας βοηθήσει τόσο στην επιλογή της μεθόδου, όσο και στον τύπο του ορίσματος που θα πρέπει να εισάγετε:





Η δυνατότητα επιλογής από αναδιπλούμενες λίστες, είναι ένα πολύ δυνατό χαρακτηριστικό της VB!

Frame (πλαίσιο):

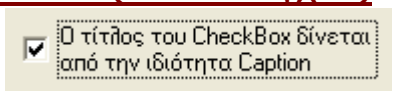


Πρόκειται για ένα εργαλείο ομαδοποίησης στοιχείων ελέγχου επάνω στη φόρμα. Είναι εξαιρετικά χρήσιμο γιατί παρέχει ευελιξία και διευκολύνει την οργάνωση των εργαλείων επάνω στη φόρμα.

Οι ιδιότητές του, τα συμβάντα και οι μέθοδοι δεν διαφέρουν από αυτές της Label (κάποιες ωστόσο δεν υπάρχουν) και έτσι θα αναφερθούν επιγραμματικά:

Ιδιότητα	Συμβάν	Μέθοδος
(Name)	Απλό «κλικ» Click	Move
AutoSize	Διπλό «κλικ» DblClick	Refresh
BackColor	Πάτημα και κράτημα πλήκτρου ποντικιού MouseDown	ZOrder
BorderStyle	Απελευθέρωση πλήκτρου ποντικιού (μετά την παραπάνω ενέργεια) MouseUp	
Caption	Κίνηση του ποντικιού από πάνω MouseMove	
Enabled		
Font		
ForeColor		
Height		
Left		
MousePointer		
ToolTipText		
Top		
Visible		
Width		

CheckBox (κουτί ελέγχου):



Ένα πολύ σημαντικό εργαλείο το οποίο χρησιμοποιείται για να δώσει στον χρήστη την ευχέρεια να εκτελέσει μια διαδικασία με πολλαπλές επιλογές.

Στις ήδη γνωστές ιδιότητες που έχουμε δει παραπάνω, θα προσθέσουμε κάποιες ακόμα οι οποίες αποτελούν το κλειδί της διαχείρισης ενός CheckBox.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
Style	Προσδιορίζει το οπτικό αποτέλεσμα του CheckBox.	0: (Standard) 1: (Graphical)	0: Όπως φαίνεται στην παραπάνω εικόνα. 1: Παρουσιάζεται ως κουμπί το οποίο παραμένει πατημένο όταν είναι τσεκαρισμένο
Value	Προσδιορίζει αν το CheckBox είναι τσεκαρισμένο ή όχι.	0: UnChecked 1: Checked 2: Grayed	0: Μη τσεκαρισμένη επιλογή. 1: Τσεκαρισμένη επιλογή. 2: Τσεκαρισμένη επιλογή ως προεπιλογή (default option)

Τα συμβάντα που έχει το CheckBox παρουσιάζονται στον παρακάτω πίνακα.

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «Check1»)
Απλό «κλικ» Click	Όταν κάνουμε κλικ επάνω στο CheckBox. (Παράλληλα αυτό σημαίνει ότι αν είναι επιλεγμένο, αποεπιλέγεται και αντίστροφα)	Private Sub Check1_Click() End Sub
Πάτημα και κράτημα πλήκτρου ποντικιού MouseDown	Όταν κάνουμε κλικ και κρατάμε πατημένο το πλήκτρο.	Private Sub Check1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single) End Sub
Απελευθέρωση πλήκτρου ποντικιού (μετά από την παραπάνω ενέργεια) MouseUp	Όταν, αφού έχουμε κρατήσει πατημένο το πλήκτρο του ποντικιού, το ελευθερώνουμε.	Private Sub Check1_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)

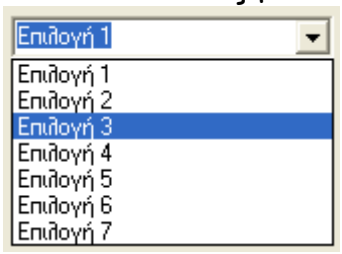
		End Sub
Κίνηση του ποντικιού από πάνω MouseMove	Όταν απλά περνάμε το δείκτη του ποντικιού πάνω από την Label.	Private Sub Check1_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single) End Sub
Σε εστίαση GotFocus	Όταν θέτουμε «προσοχή» σε ένα εργαλείο (πχ όταν μεταφερόμαστε σε αυτό με το Tab Key)	Private Sub Check1_GotFocus() End Sub
Εκτός εστίασης LostFocus	Όταν φεύγουμε από ένα εργαλείο.	Private Sub Check1_LostFocus() End Sub

Οι μέθοδοι που μας ενδιαφέρουν είναι οι ίδιες όπως και στα προηγούμενα στοιχεία ελέγχου (label, frame) και για αυτό δεν θα επαναληφθούν.

ComboBox (Πλαίσιο αναδιπλούμενης λίστας):

«Διπλωμένο»

«Σε ανάπτυξη»



Το ComboBox είναι ένα πλαίσιο το οποίο χρησιμοποιούμε για οικονομία χώρου, όταν θέλουμε να διαθέσουμε ένα πλήθος επιλογών στον χρήστη. Μας δίνει την δυνατότητα της απλής ή πολλαπλής επιλογής (ανάλογα με την τιμή μιας συγκεκριμένης ιδιότητας).

Οι ιδιότητες που έχει είναι αρκετές. Ωστόσο, θα εξετάσουμε μόνο ορισμένες από αυτές. Οι ιδιότητες που έχουν αναλυθεί σε προηγούμενα στοιχεία ελέγχου θα παραληφθούν για οικονομία χώρου.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), BackColor, Enabled, Font, ForeColor, Height, Left, MousePointer, Style, ToolTipText, Top, Visible, Width	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Locked	Κλειδώνει το ComboBox ώστε να μην μπορεί να γίνει επιλογή.	True ή False	True: Κλειδωμένο False: Ξεκλείδωτο (ελεύθερο)
Sorted	Προσδιορίζει αν τα περιεχόμενα του ComboBox θα είναι ταξινομημένα ή όχι.	True ή False	True: Ταξινομημένα False: Ως εισήχθησαν
Text	Προσδιορίζει την τρέχουσα επιλογή (αυτή δηλαδή που εμφανίζεται και όταν η λίστα είναι «διπλωμένη»	Κάποιο από τα ορίσματα που περιέχει μέσα στη λίστα.	Τρέχων επιλογή
List	**Επιστρέφει** το αντικείμενο που επιλέξαμε	Index: Ένας αριθμός που δείχνει τον αύξοντα αριθμό του αντικειμένου στη λίστα	**vName** = Combo1.List(1)
ListCount	**Επιστρέφει** το πλήθος των καταχωρίσεων στη λίστα του ComboBox.	Κανένα	**iLength** = Combo1.ListCount

Σημείωση

Τα ονόματα vName και iLength είναι τυχαία. Είναι ονόματα μεταβλητών τα οποία επιλέχθηκαν αυθαίρετα, με μοναδικό περιορισμό να αρχίζουν από γράμμα και να μην είναι δεσμευμένες λέξεις της Visual Basic. Το ότι η μια αρχίζει από το γράμμα «v» και η άλλη από το γράμμα «i» είναι μια παραδοχή που γίνεται για να περιγραφούν δυο διαφορετικοί τύποι μεταβλητών: Οι Variant και οι Integer. Οι πρώτες δέχονται όλων των ειδών τις τιμές ενώ οι δεύτερες μόνο ακέραιους. Ο λόγος που επιλέχθηκε στην πρώτη περίπτωση μια Variant είναι γιατί δεν γνωρίζουμε τι είδους τιμή θα μας επιστραφεί. Μπορεί να είναι λέξη, αριθμός, σύμβολα, οτιδήποτε. Στην δεύτερη περίπτωση επιλέχθηκε ένας Integer γιατί το πλήθος των καταχωρίσεων είναι ακέραιος θετικός αριθμός >=0. Θα περιγράψουμε τους τύπους και τον τρόπο δήλωσης των μεταβλητών σε επόμενο κεφάλαιο.

Τα συμβάντα που ενεργοποιούνται είναι τα παρακάτω.

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (υποτίθεται ότι η ιδιότητα (Name) έχει ως όρισμα το «Combo1»)
Αλλαγή Change	Όταν αλλάζει το περιεχόμενο του Combo (δηλαδή η Text)	Private Sub Combo1_Change() End Sub
Απλό «κλικ» Click	Όταν κάνουμε κλικ επάνω στο Combo.	Private Sub Combo1_Click() End Sub
Διπλό «κλικ» DblClick	Όταν κάνουμε διπλό κλικ πάνω στο Combo	Private Sub Combo1_DblClick() End Sub
Σε εστίαση GotFocus	Όταν θέτουμε «προσοχή» σε ένα εργαλείο (πχ όταν μεταφερόμαστε σε αυτό με το Tab Key)	Private Sub Combo1_GotFocus() End Sub
Εκτός εστίασης LostFocus	Όταν φεύγουμε από ένα εργαλείο.	Private Sub Combo1_LostFocus() End Sub
Κύλιση Scroll	Όταν «κυλάμε» την αναδιπλούμενη λίστα	Private Sub Combo1_Scroll() End Sub
Αναδίπλωση DropDown	Όταν πατάμε το βελάκι (στα δεξιά) που δείχνει κάτω.	Private Sub Combo1_DropDown() End Sub

Οι διαθέσιμες μέθοδοι είναι αυτές που κάνουν το ComboBox να λειτουργεί. Εισάγονται όπως όλες οι μέθοδοι σε περιβάλλον σύνταξης κώδικα. Θα αναπτύξουμε ορισμένες από τις βασικές μεθόδους στη συνέχεια.

Μέθοδος	Επεξήγηση	Ορίσματα	Σύνταξη (Θεωρώντας ότι το όνομα του ComboBox είναι Combo1)
Clear	Διαγράφει τα περιεχόμενα του ComboBox	Κανένα	Combo1.Clear
AddItem	Προσθέτει μια εγγραφή (επιλογή) στο ComboBox	Οτιδήποτε	Combo1.AddItem "Επιλογή 1" Combo1.AddItem "12.45" Combo1.AddItem "mail@mail.com"
RemoveItem	Διαγράφει ένα αντικείμενο από τη λίστα του ComboBox	Index: Ένας αριθμός που δείχνει τον αύξοντα αριθμό του αντικειμένου στη λίστα	Combo1.RemoveItem(1) Combo1.RemoveItem(2) κτλ
Refresh	Κάνει «ανανέωση» στην λίστα.	Κανένα	Combo1.Refresh
SetFocus	Θέτει τη λίστα «υπό εστίαση»	Κανένα	Combo1.SetFocus

HScrollBar (Οριζόντια μπάρα κύλισης), VScrollBar (Κατακόρυφη μπάρα κύλισης) :

Κατακόρυφη

Οριζόντια

Τόσο η οριζόντια όσο και η κατακόρυφη μπάρα κύλισης είναι στοιχεία ελέγχου «πλοηγητικού» χαρακτήρα. Μπορούμε να μετακινηθούμε μεταξύ μιας ελάχιστης και μιας μέγιστης τιμής, εύκολα και γρήγορα. Οι ιδιότητες, τα συμβάντα και οι μέθοδοι είναι κοινές και για τα δύο. Το μόνο που πρακτικά τα κάνει να διαφέρουν είναι ο προσανατολισμός του εργαλείου. Ας τα δούμε αναλυτικότερα.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Enabled, Height, Left, MousePointer, Top, Visible, Width	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Min	Προσδιορίζει το κάτω όριο του πεδίου τιμών	Ακέραιος αριθμός [-32768, 32767]	Θέτουμε το κάτω όριο
Max	Προσδιορίζει το άνω όριο του πεδίου τιμών	Ακέραιος αριθμός [-32768, 32767] μεγαλύτερος του Min	Θέτουμε το άνω όριο
Value	Προσδιορίζει την τρέχουσα τιμή	Ακέραιος αριθμός [-32768, 32767] ανάμεσα στα Min και Max	Θέτουμε / παίρνουμε την τρέχουσα τιμή
SmallChange	Προσδιορίζει την αλλαγή της τρέχουσας τιμής (χρησιμοποιώντας τα βελάκια κατεύθυνσης)	Ακέραιος αριθμός [-32768, 32767]	Πλοήγηση (όπως στην επεξήγηση με αλλαγή τιμής ανά τον δοσμένο αριθμό)
LargeChange	Προσδιορίζει την αλλαγή της τρέχουσας τιμής (κάνοντας «κλικ» στο λευκό «σώμα» του εργαλείου, ανάμεσα στα βελάκια κατεύθυνσης)	Ακέραιος αριθμός [-32768, 32767]	Πλοήγηση (όπως στην επεξήγηση με αλλαγή τιμής ανά τον δοσμένο αριθμό)

Τα συμβάντα της μπάρας κύλισης είναι (τα περισσότερα) λίγο – πολύ γνωστά. Εμείς θα παραθέσουμε επιγραμματικά τα γνωστά και θα επεξηγήσουμε τα συμβάντα τα οποία έχουν να κάνουν με τον τρόπο διαχείρισης της μπάρας κύλισης.

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «Hscroll1»)
GotFocus, LostFocus	Όπως προαναφέρθηκαν	Private Sub HScroll1_GotFocus() End Sub Private Sub HScroll1_LostFocus() End Sub
Αλλαγή Change	Όταν χρησιμοποιούμε τα βελάκια κατεύθυνσης της μπάρας κύλισης για να αλλάξουμε την τιμή της.	Private Sub HScroll1_Change() End Sub
Κύλιση Scroll	Όταν χρησιμοποιούμε τον δείκτη θέσης της μπάρας κύλισης για να αλλάξουμε την τιμής της.	Private Sub HScroll1_Scroll() End Sub

Οι μέθοδοι είναι κοινές με τα περισσότερα ως τώρα στοιχεία ελέγχου

Μέθοδος	Επεξήγηση	Ορίσματα	Σύνταξη (Θεωρώντας ότι το όνομα του scrollbar είναι Hscroll1)
Move, Refresh, Zorder, SetFocus	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν (με όνομα αναφοράς βέβαια το Hscroll1)

Timer (Χρονομετρητής):

Είναι ένα εργαλείο το οποίο μας επιτρέπει να θέσουμε μια έγκυρη τιμή χρονικής διάρκειας (θα δούμε παρακάτω τη μονάδα), που, με τη λήξη αυτού «πυροδοτεί» κάποιο συμβάν. Είναι αόρατο στον τελικό χρήστη.

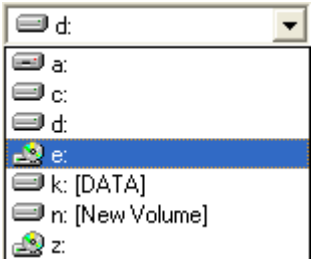
Οι ιδιότητες και τα συμβάντα που συνοδεύουν τον χρονομετρητή είναι ελάχιστα. Δεν έχει μεθόδους. Ας τα δούμε αναλυτικά.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
Left, Top	Όπως προαναφέρθηκαν (χωρίς ωστόσο ουσιαστικό αποτέλεσμα, αφού το εργαλείο είναι αόρατο κατά τη φάση εκτέλεσης)	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Enabled	Το τι κάνει η ιδιότητα Enabled το είδαμε παραπάνω. Σε αυτό το εργαλείο και μόνο, η στιγμή που γίνεται Enabled είναι η στιγμή εκκίνησης της αντίστροφης μέτρησης (Ενεργοποίηση).	True ή False	Όταν γίνει True αρχίζει να μετράει ο χρόνος που έχουμε ορίσει για την έναρξη της εκτέλεσης.
Interval	Είναι η ιδιότητα που δέχεται τον χρόνο, μετά το πέρας του οποίου θα εκτελέσει την διαδικασία που ορίζεται σε αυτό. Η μονάδα που χρησιμοποιεί είναι τα msec (1000msec=1sec)	Από 0 έως 65535	Δίνει την αντίστοιχη τιμή.

Το συμβάν του Control Timer είναι ένα:

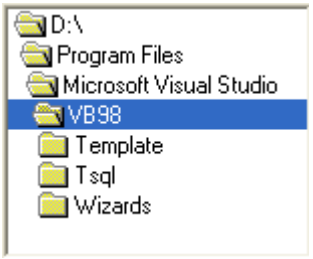
Συμβάν	Ενεργοποίηση	Αποτέλεσμα
		Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «Timer1»)
Timer	Όταν το εργαλείο Timer είναι σε κατάσταση Enabled = True και έχει ολοκληρωθεί το χρονικό διάστημα που θέσαμε στο Interval.	Private Sub Timer1_Timer() End Sub

DriveListBox (Λίστα οδηγών δίσκων), DirListBox (Λίστα φακέλων), FileListBox (Λίστα αρχείων):



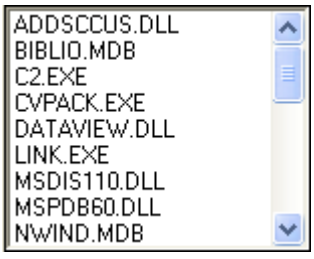
DriveListBox

Το εργαλείο αυτό μας παρέχει τη δυνατότητα πλοήγησης ανάμεσα στους εγκατεστημένους δίσκους στο σύστημά μας. Είναι το πρώτο από μια σειρά συνεργαζόμενων εργαλείων. Δεν επιτρέπει την πλοήγηση στο περιεχόμενο των δίσκων, παρά μόνο στην επιλογή κάποιου.



DirListBox

Είναι το δεύτερο κατά σειρά χρήσης εργαλείο, λειτουργεί σε δεύτερο επίπεδο, και μας επιτρέπει την επιλογή φακέλου στον δίσκο που ήδη έχουμε προεπιλέξει παραπάνω. Μπορούμε να πλοηγηθούμε στα περιεχόμενα κάποιου φακέλου με το συγκεκριμένο εργαλείο, αλλά το μόνο που μπορούμε να δούμε είναι οι φάκελοι.



FileListBox

Το τρίτο κατά σειρά εργαλείο (από άποψη χρήσης). Αφού επιλέξουμε δίσκο στο DriveListBox και πλοηγηθούμε στον φάκελο που μας ενδιαφέρει από το DirListBox, επιλέγουμε τελικώς το αρχείο που μας ενδιαφέρει. Ωστόσο αν θέλουμε να δημιουργήσουμε κάποιο αρχείο σε κάποιο φάκελο, τότε μπορούμε να παραλείψουμε το FileListBox.

Τα παραπάνω εργαλεία, αν και είναι διαφορετικά, τα εξετάζουμε σαν μια ομάδα, επειδή ο τρόπος λειτουργίας τους είναι παραπλήσιος και συνδέονται όπως είδαμε άμεσα.

Θα παρουσιάσουμε κάποιες ιδιότητες των παραπάνω εργαλείων, οι οποίες είναι απαραίτητες για τη χρήση τους.

DriveListBox

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), BackColor, Enabled, ForeColor, Font, Height, Width, Left, Top, MousePointer, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Drive	Είναι η βασική ιδιότητα του σχετικού εργαλείου. Μας «επιστρέφει» την επιλογή που κάναμε, ή θέτουμε προορισμό σε κάποιο δίσκο.	Όλοι οι υπάρχοντες δίσκοι, όχι ως ονόματα αλλά ως καταχωρισμένα «γράμματα». Για παράδειγμα, μπορούμε να θέσουμε: Drive1.Drive = "C" Παρόλο που μπορεί εμείς να ονομάζουμε τον δίσκο «C» για παράδειγμα «SYSTEM»	Ανάλογα με τη χρήση είτε παίρνουμε την τιμή του δίσκου που επιλέγει ο χρήστης, είτε θέτουμε εμείς κάποια τιμή.

DirListBox

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), BackColor, Enabled, ForeColor, Font, Height, Width, Left, Top, MousePointer, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Path	Είναι η βασική ιδιότητα του σχετικού εργαλείου. Μας «επιστρέφει» την επιλογή που κάναμε, ή θέτουμε προορισμό σε κάποιο φάκελο.	Όλοι οι υπάρχοντες φάκελοι, στον σχετικό δίσκο.	Ανάλογα με τη χρήση είτε παίρνουμε την τιμή του φακέλου που επιλέγει ο χρήστης, είτε θέτουμε εμείς κάποια τιμή.

FileListBox

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), BackColor, Enabled, ForeColor, Font, Height, Width, Left, Top, MousePointer, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Path	Μας «επιστρέφει» την επιλογή που κάναμε, ή θέτουμε προορισμό σε κάποιο φάκελο που περιέχει αρχεία.	Όλοι οι υπάρχοντες φάκελοι, στον σχετικό δίσκο.	Ανάλογα με τη χρήση, είτε παίρνουμε την τιμή του φακέλου που επιλέγει ο χρήστης, είτε θέτουμε εμείς κάποια τιμή.
FileName	Ορίζει το αρχείο που επιλέξαμε από το	Όλα τα υπάρχοντα αρχεία στον επιλεγμένο	Επιλέγουμε κάποιο αρχείο.

	σχετικό εργαλείο.	φάκελο.	

Όλα τα παραπάνω γίνονται πιο εύκολα αντιληπτά με το παρακάτω παράδειγμα. Θεωρούμε ότι τα ονόματα των εργαλείων είναι: Drive1, Dir1 και File1:

```
Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
File1.Path = Dir1.Path
End Sub

Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub

Private Sub File1_Click()
Dim SingleName As Variant
Dim TotalPath As Variant
SingleName = File1.FileName
TotalPath = Dir1.Path + "\" + File1.FileName
End Sub
```

Ας εξηγήσουμε τι ακριβώς συμβαίνει στον παραπάνω κώδικα.

Βλέπουμε 3 blocks κώδικα τα οποία αρχίζουν με τις φράσεις «Private Sub» και τελειώνουν με τις φράσεις «End Sub». Πρόκειται για 3 υπορουτίνες εκτέλεσης συμβάντος.

Η πρώτη αφορά στην αλλαγή του στοιχείου ελέγχου DriveListBox με το όνομα Drive1. Τι σημαίνει αλλαγή; Όταν στο Drive1 επιλέξουμε ένα δίσκο, αυτό σημαίνει αλλαγή και ενεργοποιεί την εκτέλεση της υπορουτίνας που αφορά σε αυτό το συμβάν. Όπως διαπιστώνουμε, χρησιμοποιούμε τις ιδιότητες που αναφέραμε παραπάνω. Δηλαδή αν για παράδειγμα επιλέξουμε τον δίσκο «D», τότε αυτόματα, στο επόμενο στοιχείο ελέγχου, το DirListBox, θα παρουσιάσει τα περιεχόμενα του δίσκου που επιλέξαμε. Επίσης, το ίδιο θα κάνει και στο FileListBox. Ενώ δηλαδή ενεργοποιείται μια κλήση συμβάντος προς ένα εργαλείο, παράλληλα αποστέλλεται η αλλαγή και στα επόμενα, δηλαδή επιτυγχάνεται ένας συγχρονισμός των εργαλείων.

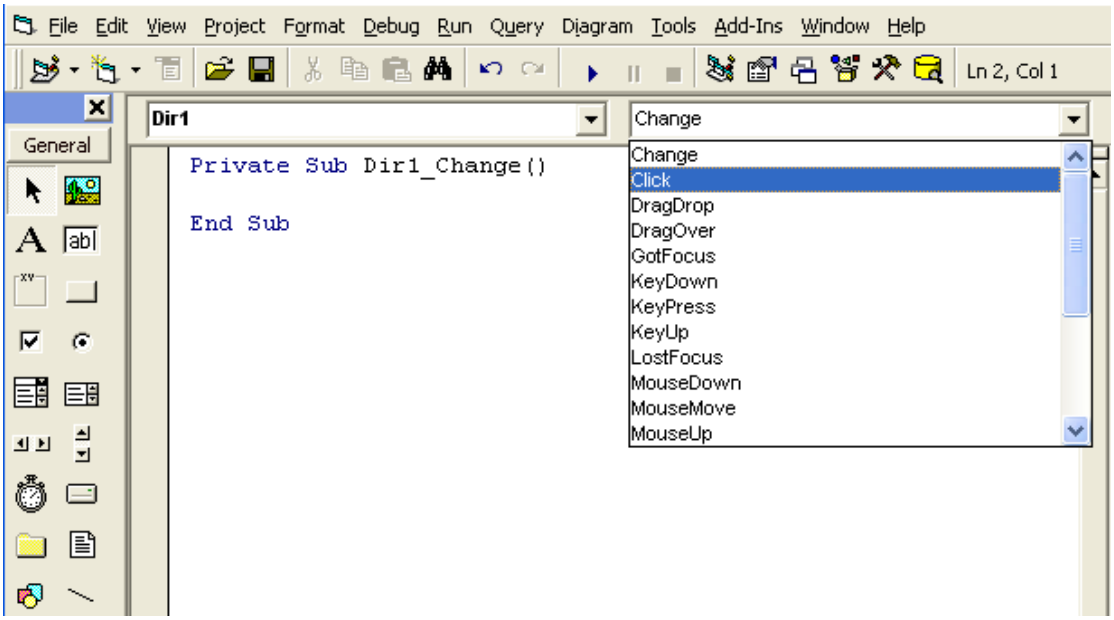
Στην δεύτερη υπορουτίνα η οποία αφορά στην αλλαγή (επιλογή) κάποιου φακέλου, συγχρονίζεται μόνο το επόμενο στοιχείο ελέγχου που αφορά στην επιλογή του αρχείου (δεν υπάρχει άλλωστε λόγος να αλλάξει η διαδρομή του δίσκου).

Στην τρίτη υπορουτίνα πλέον, μας ενδιαφέρει να επιλέξουμε το αρχείο που μας ενδιαφέρει. Όπως βλέπουμε, υπάρχουν δύο δηλώσεις μεταβλητών. Όπως προαναφέρθηκε, η μεταβλητή είναι μια μη-δεσμευμένη λέξη, η οποία χρησιμοποιείται για να καταχωριστεί μια πληροφορία. Η δήλωσή της, όπως φαίνεται παραπάνω, ακολουθεί ένα τύπο σύνταξης. Αρχίζει με τη δεσμευμένη

εντολή Dim, και μετά το όνομα που θέτουμε, ορίζουμε το τι θα περιέχει η μεταβλητή. Στο παρόν παράδειγμα επιλέξαμε τον τύπο Variant ο οποίος δέχεται οτιδήποτε για καταχώριση . Υπάρχουν πολλοί τύποι μεταβλητών, όπως και κάποιοι ακόμα εναλλακτικοί τύποι δηλώσεων, τους οποίους θα δούμε σε παρακάτω κεφάλαιο. Στην πρώτη μεταβλητή που χρησιμοποιούμε, δίνουμε ως όρισμα το όνομα του αρχείου (π.χ. MyText.doc). Στην δεύτερη μεταβλητή όμως δίνουμε την ολική διαδρομή του αρχείου στον Η/Υ (π.χ. C:\Program Files\MyProgram\Backup\MyText.doc). Ο δεύτερος τρόπος είναι πιο σωστός, διότι γίνεται αποφυγή λαθών προορισμού.

Τα συμβάντα και οι μέθοδοι των παραπάνω εργαλείων, είναι ίδια με τα περισσότερα εργαλεία που έχουμε γνωρίσει ως τώρα, και δεν θα τα αναλύσουμε περαιτέρω. Αυτά όμως που πρέπει να γνωρίζουμε είναι τα εξής, και ισχύουν για όλα τα εργαλεία:

1. Που υπάρχουν τα συμβάντα; Μπορώ να τα βρω κάπου όταν συντάσσω κώδικα ή πρέπει απλά να τα γνωρίζω;



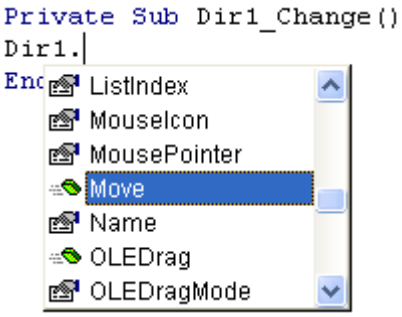
Όταν είμαστε σε κατάσταση σύνταξης κώδικα, στο πάνω δεξιά μέρος του παράθυρου υπάρχει ένα πλαίσιο αναδιπλούμενης λίστας το οποίο περιέχει όλα τα συμβάντα για το στοιχείο ελέγχου που επιλέξαμε. Όταν στην κατάσταση σχεδίασης (τοποθέτησης δηλαδή των εργαλείων σε μια φόρμα) κάνουμε διπλό κλικ πάνω σε ένα εργαλείο, τότε αυτομάτως θα μεταφερθούμε στην κατάσταση σύνταξης κώδικα με το εργαλείο επιλεγμένο, και προτεινόμενο συμβάν το πιο σημαντικό του εργαλείου. Παραπάνω κάναμε διπλό κλικ στο DirListBox που τοποθετήσαμε (με το όνομα Dir1).

2. Που υπάρχουν οι μέθοδοι;

Τόσο οι μέθοδοι, όσο και οι ιδιότητες (για τις οποίες υπάρχει και το γνωστό πλέον Properties Window) είναι προσβάσιμες κατά την ώρα της σύνταξης κώδικα. Για να εμφανιστούν, πρέπει να πληκτρολογήσουμε το όνομα του εργαλείου στο οποίο αναφερόμαστε. Παράδειγμα:

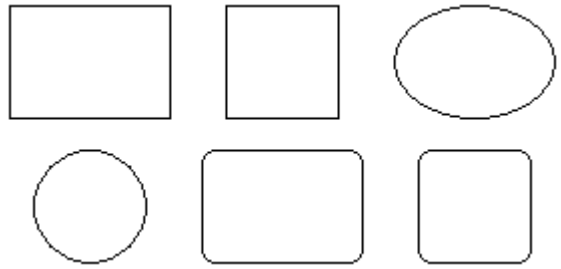
```
Private Sub Dir1_Change()
Dir1.
End Sub
```

Αφότου γράψουμε δηλαδή το όνομα του εργαλείου για το οποίο θέλουμε να χρησιμοποιήσουμε μια ιδιότητα ή μια μέθοδος και πατήσουμε την τελεία « . » θα εμφανιστεί ένα πλαίσιο λίστας



το οποίο περιλαμβάνει όλες τις ιδιότητες και τις μεθόδους, έτσι ώστε να επιλέξουμε. Αξίζει να σημειωθεί ότι με αυτόν τον τρόπο έχουμε πολύ περισσότερες ιδιότητες στην διάθεσή μας για κάθε στοιχείο ελέγχου. (Οι μέθοδοι ξεχωρίζουν από τις ιδιότητες διότι έχουν ένα πράσινο σχεδιάκι αριστερά, ενώ οι ιδιότητες παρουσιάζουν ένα χέρι να κρατάει μια καρτέλα).

Shape (Σχήμα):



Το στοιχείο ελέγχου Shape μας επιτρέπει να προσθέτουμε απλά σχήματα πάνω στις φόρμες μας. Τα σχέδια που παρουσιάζονται δίπλα είναι οι έξι επιλογές που διατίθενται.

Το εργαλείο Shape δεν έχει συμβάντα, δηλαδή δεν μπορούμε με κλικ να ενεργοποιήσουμε κάποια διαδικασία. Προσφέρει μόνο διακοσμητικό χαρακτήρα στα προγράμματά μας.

Οι ιδιότητες του shape προφανώς έχουν να κάνουν με το στυλ εμφάνισής του.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Left,Top, Width, Height, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
BackColor	Θέτει το χρώμα του φόντου του αντικειμένου.	Έγκυρο τύπο χρώματος (όπως προαναφέρθηκαν)	Όπως προαναφέρθηκαν
BackStyle	Θέτει τον τύπο εμφάνισης (συμπαγής ή διαφανής)	0 ή 1	0: Διαφανής 1: Συμπαγής
BorderColor	Θέτει το χρώμα του περιγράμματος του αντικειμένου.	Έγκυρο τύπο χρώματος (όπως προαναφέρθηκαν)	Όπως προαναφέρθηκαν
BorderStyle	Θέτει τον τύπο γραμμής του περιγράμματος.	0, 1, 2, 3, 4, 5, 6	0: Χωρίς περίγραμμα 1: Συνεχής 2: Παύλες 3: Τελείες 4: Παύλα-τελεία 5: Παύλα-τελεία-τελεία 6: Τύπου συνεχής
BorderWidth	Θέτει το πάχος του περιγράμματος	1 έως 8192	Αλλάζει το πάχος του περιγράμματος
FillColor	Θέτει το χρώμα του αντικειμένου.	Έγκυρο τύπο χρώματος (όπως προαναφέρθηκαν)	Όπως προαναφέρθηκαν
FillStyle	Προσδιορίζει τον τρόπο «γεμίσματος» του αντικειμένου.	0, 1, 2, 3, 4, 5, 6, 7	0: Συμπαγής 1: Χωρίς γέμισμα 2: Οριζόντιες γραμμές 3: Κάθετες γραμμές 4: Διαγώνιες 5: Διαγώνιες 6: Πλέγμα 7: Διαγώνιο πλέγμα
Shape	Προσδιορίζει τον τύπο του σχήματος	0, 1, 2, 3, 4, 5	0: Ορθογώνιο 1: Τετράγωνο 2: Έλλειψη

			3: Κύκλος 4: Ορθογώνιο με στρογγυλεμένες γωνίες 5: Τετράγωνο με στρογγυλεμένες γωνίες

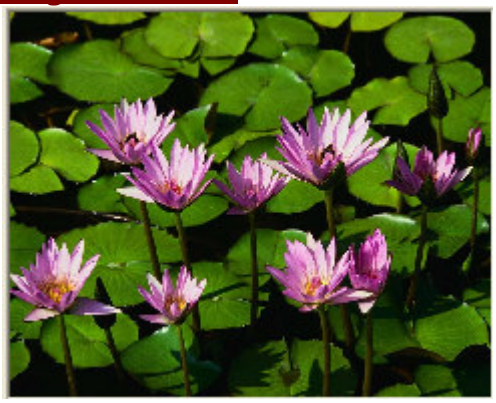
Οι μέθοδοι του shape είναι ίδιες με αυτές των περισσότερων στοιχείων ελέγχου και δεν θα αναφερθούν ξεχωριστά.

Line (Γραμμή):

Το εργαλείο Line ανήκει ουσιαστικά στην κατηγορία του εργαλείου Shape. Δεν έχει συμβάντα. Έχει μόνο 2 μεθόδους (Refresh και ZOrder) τις οποίες έχουμε ξαναδεί και απλές ιδιότητες.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), BorderColor, BorderStyle, BorderWidth, Visible	*Όπως προαναφέρθηκαν*	*Όπως προαναφέρθηκαν*	*Όπως προαναφέρθηκαν*
X1	Η τετμημένη του πρώτου άκρου.	Πραγματικός αριθμός	Προσδιορισμός συντεταγμένων πρώτου σημείου.
Y1	Η τεταγμένη του πρώτου άκρου.	Πραγματικός αριθμός	
X2	Η τετμημένη του δεύτερου άκρου.	Πραγματικός αριθμός	Προσδιορισμός συντεταγμένων δεύτερου σημείου.
Y2	Η τεταγμένη του δεύτερου άκρου.	Πραγματικός αριθμός	

Image (Εικόνα):



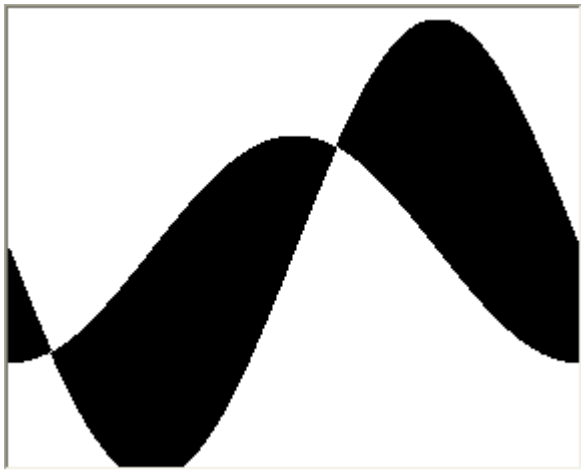
Το εργαλείο αυτό μας παρέχει τη δυνατότητα να προσθέτουμε εικόνες σε μια φόρμα.

Έχει απλές ιδιότητες, μεθόδους και συμβάντα τα οποία θα τα δούμε συνοπτικά:

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Appearance, BorderStyle, Enabled, Height, Width, Left, Top, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Picture	Προσδιορίζουμε το όνομα της εικόνας που θα εμφανίσει (το όνομα του αρχείου)	Οποιοδήποτε έγκυρο όνομα αρχείου εικόνας που έχουμε σε κάποιο δίσκο.	Εμφανίζει την εικόνα.
Stretch	Προσδιορίζουμε αν η εικόνα θα συμπιεστεί (οπτικά) ώστε να χωρέσει στο Image που δημιουργήσαμε.	True ή False	True: Συμπιέζει την εικόνα. False: Παρουσιάζει την εικόνα στο πραγματικό της μέγεθος. Αν το εργαλείο Image είναι πιο μικρό από την εικόνα, την κόβει.

Οι μέθοδοι και τα συμβάντα είναι ήδη γνωστά και δεν θα σχολιαστούν. Γνωρίζουμε ήδη πού θα τα βρούμε και πώς αυτά ανταποκρίνονται. Το συγκεκριμένο στοιχείο ελέγχου είναι, όπως θα δούμε παρακάτω, μια ελαφριά έκδοση του PictureBox.

PictureBox (Πλαίσιο Απεικόνισης):



Το PictureBox, σε αντίθεση με το Image, μπορεί να χρησιμοποιηθεί, πέρα από την απλή εμφάνιση μιας εικόνας και ως χώρος σχεδίασης. Μπορούμε να προβάλουμε και να δημιουργήσουμε οτιδήποτε επιθυμούμε πάνω σε ένα PictureBox, από ελεύθερο ή γραμμικό σχέδιο, μέχρι και γραφικές παραστάσεις, να προσθέσουμε εργαλεία και να το χρησιμοποιήσουμε ως ομαδοποιητή κλπ. Είναι λοιπόν ένα πανίσχυρο εργαλείο, αλλά και δύσκολο για πλήρη αξιοποίηση.

Θα προσπαθήσουμε να αναφερθούμε σε λίγα χαρακτηριστικά του πλαισίου απεικόνισης ώστε να γίνουν αντιληπτά.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Appearance, BackColor, BorderStyle, DrawStyle, DrawWidth, Enabled, FillColor, FillStyle, Font, ForeColor, MousePointer, Height, Width, Left, Top, Picture, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Align	Προσδιορίζει αν το πλαίσιο απεικόνισης θα παρουσιάζεται προσαρμοσμένο σε κάποια πλευρά της φόρμας.	0, 1, 2, 3, 4	0: Χωρίς προσαρμογή 1: Πάνω 2: Κάτω 3: Αριστερά 4: Δεξιά
AutoRedraw	Χρησιμοποιείται για να κρατάει στη μνήμη την εικόνα που απεικονίζει (γραφικά)	True ή False	True: Κρατάει στη μνήμη τα γραφικά. False: Δεν τα κρατάει στη μνήμη. Ένα κρύψιμο της φόρμας διαγράφει τα περιεχόμενα του πλαισίου απεικόνισης.
AutoSize	Προσδιορίζει αν το μέγεθος του πλαισίου θα προσαρμόζεται αυτόματα στο μέγεθος	True ή False	True: Αυτόματη προσαρμογή. False: Καμιά προσαρμογή.

	της προβαλλόμενης εικόνας.		
FontTransparent	Προσδιορίζει αν το προβαλλόμενο κείμενο θα αποκρύβει ή όχι τα γραφικά που βρίσκονται από πίσω του.	True ή False	True: Τα κείμενα παρουσιάζονται διαφανή False: Τα κείμενα παρουσιάζονται συμπαγή.
ScaleMode	Προσδιορίζει την μονάδα μέτρησης αποστάσεων του πλαισίου απεικόνισης.	0, 1, 2, 3, 4, 5, 6, 7	0: Χρήστη 1: Twip 2: Point 3: Pixel 4: Χαρακτήρας 5: Ίντσα 6: Χιλιοστό 7: Εκατοστό
ScaleLeft	Προσδιορίζει την τετμημένη του αριστερού άκρου.	Οποιοδήποτε έγκυρο πραγματικό αριθμό.	Διαμορφώνουν το χώρο προβολής του πλαισίου απεικόνισης, και δημιουργούν **νέο σύστημα συντεταγμένων**.
ScaleTop	Προσδιορίζει την τεταγμένη του πάνω άκρου.	Οποιοδήποτε έγκυρο πραγματικό αριθμό.	
ScaleHeight	Προσδιορίζει το διάστημα που είναι ορατό κατά τον άξονα των Υ.	Οποιοδήποτε έγκυρο πραγματικό αριθμό.	
ScaleWidth	Προσδιορίζει το διάστημα που είναι ορατό κατά τον άξονα των Χ.	Οποιοδήποτε έγκυρο πραγματικό αριθμό.	

Σημείωση

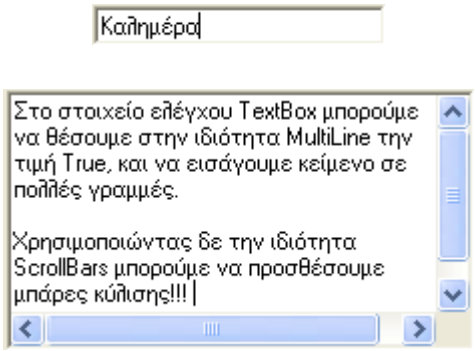
Γενικότερα, όπως ήδη έχετε παρατηρήσει, όλα τα μεγέθη των αντικειμένων έχουν ως αρχή μέτρησης την πάνω αριστερή άκρη της οθόνης. Αυτή η άκρη έχει τοπικές συντεταγμένες (0,0). Το ίδιο συμβαίνει και για τις συντεταγμένες της φόρμας. Αυτό όμως δεν είναι ένα βολικό σύστημα συντεταγμένων, διότι αν και τα Χ αυξάνουν από αριστερά προς τα δεξιά, τα Υ αυξάνουν από πάνω προς τα κάτω. Με τις ιδιότητες της κατηγορίας Scale μπορείτε να διαμορφώσετε ένα κανονικό σύστημα συντεταγμένων. Αξίζει να σημειωθεί ότι ιδιότητες τύπου Scale έχει και η φόρμα.

Οι μέθοδοι που χρησιμοποιεί το PictureBox είναι περισσότερες από κάθε άλλο στοιχείο ελέγχου και χρησιμοποιούνται κυρίως για τις γραφικές απεικονίσεις.

Μέθοδος	Επεξήγηση	Ορίσματα	Σύνταξη (Θεωρώντας ότι το όνομα του PictureBox είναι Picture1)
Move, SetFocus, ZOrder, Refresh	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν (με όνομα αναφοράς βέβαια το Picure1)
Circle	Σχεδιάζει ένα κύκλο στο PictureBox.	Κέντρο κύκλου X,Y, ακτίνα R, χρώμα, τμήμα (σε ακτίνια)	Picture1.Cirlce (X,Y), R, Color, Start, End
Cls	Καθαρίζει το PictureBox από όλα τα γραφικά	Κανένα	Picture1.Cls
Line	Σχεδιάζει μια γραμμή στο PictureBox.	X1,Y1 και X2,Y2, χρώμα	Picture1.Line (X1,Y1)-(X2,Y2), color
Point	Επιστρέφει την τιμή του χρώματος σε ένα συγκεκριμένο σημείο.	Θέση σημείου (X,Y)	TheColor = Picture1.Point (X,Y)
Pset	Σχεδιάζει ένα σημείο σε μια συγκεκριμένη θέση.	Θέση σημείου (X,Y), χρώμα	Picture1.Pset (X,Y), color
Scale	Δημιουργεί ένα πλήρες σύστημα συντεταγμένων	X1: X αριστερά Y1: Y πάνω X2: X δεξιά Y2: Y κάτω	Picture1. Scale (X1 , Y1)- (X2 , Y2)
TextHeight	Επιστρέφει το ύψος ενός κειμένου στην μονάδα μέτρησης που ορίσαμε.	Κάποιο κείμενο	MyH = Picture1.TextHeight(TheText)
TextWidth	Επιστρέφει το μήκος ενός κειμένου στην μονάδα μέτρησης που ορίσαμε.	Κάποιο κείμενο	MyW = Picture1.TextWidth(TheText)

Τα συμβάντα είναι ήδη γνωστά, και σίγουρα έχετε ήδη μάθει πως να τα καλείτε και να τα χρησιμοποιείτε.

TextBox (Πλαίσιο κειμένου):



Το πλαίσιο κειμένου είναι ένα πολύ χρήσιμο εργαλείο το οποίο μας επιτρέπει να εισάγουμε δεδομένα στο πρόγραμμα πριν και κατά τη φάση της εκτέλεσης.

Θα δούμε αναλυτικά κάποιες από τις βασικές ιδιότητές του.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Alignment, Appearance, BackColor, BorderStyle, Enabled, Font, ForeColor, MousePointer, Height, Width, Left, Top, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Locked	Κλειδώνει το εργαλείο και δεν επιτρέπει πρόσβαση σε αυτό.	True ή False	True: Κλειδωμένο False: Ελεύθερο
MaxLength	Επιτρέπει ένα μέγιστο πλήθος χαρακτήρων να γραφούν σε αυτό από το χρήστη.	Ένας έγκυρος ακέραιος αριθμός.	Δεν επιτρέπει εισαγωγή δεδομένων με πλήθος χαρακτήρων μεγαλύτερο από αυτό που θέσαμε.
MultiLine	Επιτρέπει στο πλαίσιο κειμένου την εισαγωγή πολλών σειρών δεδομένων (κειμένου)	True ή False	True: Πολλές σειρές False: Μια σειρά
PasswordChar	Θέτουμε ένα σύμβολο, αριθμό ή γράμμα ο οποίος θα εμφανίζεται στο TextBox ότι και αν εισάγει ο χρήστης. Χρησιμοποιείται για προστασία καταχωρίσεων.	Οτιδήποτε (1 χαρακτήρα)	Αποκρύπτονται τα πραγματικά δεδομένα που εισάγει ο χρήστης.
ScrollBars	Αν έχουμε επιλέξει τη χρήση πολλαπλών σειρών στο TextBox, μπορούμε να προσθέσουμε μπάρες κύλισης οι οποίες κάνουν πιο εύκολη την πρόσβαση στα δεδομένα.	0, 1, 2, 3	0: Χωρίς μπάρες κύλισης 1: Μόνο οριζόντια 2: Μόνο κατακόρυφη 3: Οριζόντια και κατακόρυφη

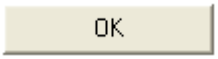
Text	Η βασικότερη ιδιότητα του πλαισίου κειμένου. Με αυτή μπορούμε να γράψουμε ένα κείμενο ή να λάβουμε αυτό που έγραψε ο χρήστης.	Οτιδήποτε	Θέτουμε ή λαμβάνουμε το περιεχόμενο του πλαισίου κειμένου.

Οι μέθοδοι που χρησιμοποιεί το πλαίσιο κειμένου (Move, Refresh, Zorder κ.α.) είναι όλες γνωστές και δεν θα αναφερθούν αναλυτικά.

Όσο αφορά τα συμβάντα, πέρα από τα είδη γνωστά (Click, DoubleClick, Change, MouseMove κτλ), το textbox περιλαμβάνει κάποια επιπλέον, που αφορούν στην πληκτρολόγηση.

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «Text1»)
KeyPress	Όταν πατάμε κάποιο πλήκτρο (γράφουμε) μέσα σε ένα πλαίσιο κειμένου.	Private Sub Text1_KeyPress(KeyAscii As Integer) End Sub
KeyDown	Όταν πατάμε κάποιο πλήκτρο (γράφουμε) μέσα σε ένα πλαίσιο κειμένου και το κρατάμε πατημένο.	Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer) End Sub
KeyUp	Όταν αφού έχουμε πατήσει (και κρατήσει πατημένο κάποιο πλήκτρο), το «ελευθερώνουμε»	Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer) End Sub

CommandButton (Κουμπί ή πλήκτρο):



Το «κουμπί» είναι το πιο άμεσο εργαλείο που χρησιμοποιείται για να «πυροδοτήσει» την έναρξη κάποιας διαδικασίας. Χρησιμοποιείται σε κάθε εφαρμογή λόγω της απλότητας χειρισμού του.

Οι ιδιότητες του κουμπιού αναφέρονται παρακάτω:

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Appearance, BackColor, Enabled, Font, Height, Width, Left, Top, ToolTipText, MousePointer, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Caption	Όπως και στο εργαλείο ετικέτα, ορίζουμε ένα κείμενο ή λέξη το οποίο θα αναγράφεται πάνω στο κουμπί.	Οτιδήποτε	Το κουμπί φέρει πάνω τη σχετική λεζάντα που συνήθως περιγράφει την αρμοδιότητα του κουμπιού.
Cancel	Προσδιορίζει αν ως συντόμευση του κουμπιού θα οριστεί το πλήκτρο [Esc] του πληκτρολογίου. Επιτρέπεται μόνο για ένα κουμπί πάνω σε κάθε φόρμα.	True ή False	True: Πάτημα του [Esc] ισοδυναμεί με πάτημα του συγκεκριμένου κουμπιού False: Προεπιλεγμένα, δεν αντιστοιχεί το [Esc]
Default	Προσδιορίζει αν ως συντόμευση του κουμπιού θα οριστεί το πλήκτρο [Enter] του πληκτρολογίου. Επιτρέπεται μόνο για ένα κουμπί πάνω σε κάθε φόρμα.	True ή False	True: Πάτημα του [Enter] ισοδυναμεί με πάτημα του συγκεκριμένου κουμπιού False: Προεπιλεγμένα, δεν αντιστοιχεί το [Enter]
Style	Ορίζει το στυλ εμφάνισης του κουμπιού.	0 ή 1	0: (Standard): Το πλήκτρο έχει κάποιες προεπιλεγμένες δυνατότητες εμφάνισης. Δεν μπορεί για παράδειγμα να έχει χρώμα ως υπόβαθρο. 1: (Graphical): Το κουμπί αποκτά εξελιγμένη γραφική συμπεριφορά. Μπορούμε να βάλουμε ως υπόβαθρο χρώμα,

			εικόνα κτλ
DisabledPicture	Προσδιορίζει την εικόνα που θα εμφανίζεται όταν θέτουμε στην ιδιότητα Enabled την τιμή False (δηλαδή όταν απενεργοποιούμε το πλήκτρο)	Οποιοδήποτε έγκυρο αρχείο εικόνας που υπάρχει σε κάποιο δίσκο.	Η εικόνα εμφανίζεται στο κουμπί, μόλις το απενεργοποιήσουμε.
DownPicture	Προσδιορίζει την εικόνα που θα εμφανίζεται όταν πατάμε (ή και κρατάμε πατημένο το κουμπί).	Οποιοδήποτε έγκυρο αρχείο εικόνας που υπάρχει σε κάποιο δίσκο.	Η εικόνα εμφανίζεται όταν πατήσουμε το κουμπί, ως φόντο.
Picture	Προσδιορίζει την εικόνα που θα εμφανίζεται ως φόντο στο κουμπί.	Οποιοδήποτε έγκυρο αρχείο εικόνας που υπάρχει σε κάποιο δίσκο.	Η εικόνα εμφανίζεται στο κουμπί, ως φόντο.

Οι μέθοδοι (Move, Refresh, SetFocus, Zorder) δεν παρουσιάζουν καμία διαφορά με τα υπόλοιπα στοιχεία ελέγχου. Τα συμβάντα μάλιστα, είναι τα απλά συμβάντα των υπόλοιπων εργαλείων.

OptionButton (Κουμπί επιλογής ή πλήκτρο επιλογής):

Το OptionButton είναι ένα στοιχείο ελέγχου το οποίο μας επιτρέπει να πραγματοποιήσουμε μία και μόνο μία επιλογή από αυτές που μας δίνονται. Μοιάζει με το εργαλείο CheckBox που είδαμε παραπάνω, αλλά η μεγάλη διαφορά έγκειται στο γεγονός ότι στο checkbox μπορούσαμε να επιλέξουμε όσα θέλουμε ταυτόχρονα ή να μην επιλέξουμε κανένα. Στο optionbutton η επιλογή είναι μια και αποκλειστική. Σκεφτείτε για παράδειγμα την ερώτηση «τι χρώμα επιθυμείτε για φόντο;». Προφανώς δεν θα μπορούσατε να επιλέξετε πάνω από ένα, αλλά ούτε και κανένα διότι δεν υπάρχει «κανένα» χρώμα!

Ας περάσουμε στις ιδιότητες του πλήκτρου επιλογής.

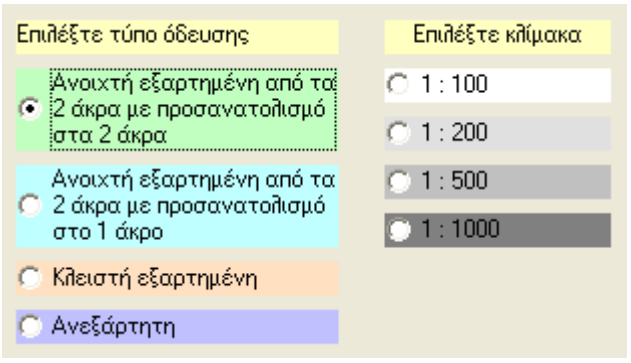
Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Alignment, Appearance, BackColor, DisabledPicture, DownPicture, Enabled, Font, ForeColor, MousePointer, Left, Top, Height, Width, ToolTipText, Visible	*Όπως προαναφέρθηκαν*	*Όπως προαναφέρθηκαν*	*Όπως προαναφέρθηκαν*
Caption	*Όπως και στο εργαλείο κουμπί, ορίζουμε ένα κείμενο ή λέξη το οποίο θα αναγράφεται πάνω στο κουμπί επιλογής.*	Οτιδήποτε	Το κουμπί επιλογής φέρει πάνω τη σχετική λεζάντα που συνήθως περιγράφει την αρμοδιότητά του.
Style	*Ορίζει το στυλ εμφάνιστης του κουμπιού επιλογής.*	0 ή 1	0: (Standard): Το πλήκτρο επιλογής έχει κάποιες προεπιλεγμένες δυνατότητες εμφάνισης. Σε αντίθεση με το CommandButton μπορεί να έχει χρώμα ως υπόβαθρο, αλλά όχι εικόνα. Παράλληλα εμφανίζεται οπτικά ως

			«ετικέτα» με ένα «κυκλάκι» το οποίο ανάλογα με την επιλογή μας είτε είναι άδειο είτε εμφανίζει εσωτερικά μια μαύρη «μπίλια». 1: (Graphical): Το κουμπί επιλογής αποκτά εμφάνιση παρόμοια με του CommandButton. Η διαφορά έγκειται στο γεγονός ότι το επιλεγμένο κουμπί επιλογής εμφανίζεται πατημένο κάτω ενώ τα υπόλοιπα σηκωμένα. Στην συγκεκριμένη περίπτωση μπορούμε να τοποθετήσουμε ως φόντο εικόνα.
Picture	Προσδιορίζει την εικόνα που θα εμφανίζεται ως φόντο στο κουμπί επιλογής. Προϋπόθεση να έχουμε ορίσει στην ιδιότητα Style την τιμή Graphical.	Οποιοδήποτε έγκυρο αρχείο εικόνας που υπάρχει σε κάποιο δίσκο.	Η εικόνα εμφανίζεται στο κουμπί επιλογής, ως φόντο.
Value	Είναι η κύρια ιδιότητα η οποία σηματοδοτεί αν ένα πλήκτρο επιλογής είναι «επιλεγμένο»	True ή False	True: Είναι το επιλεγμένο πλήκτρο. False: Δεν είναι το επιλεγμένο.

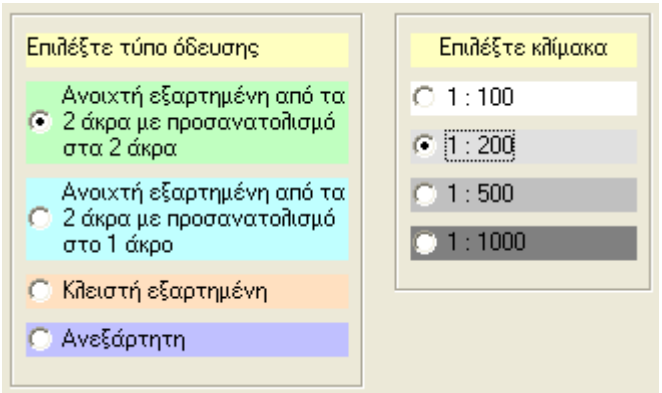
Δεν θα ασχοληθούμε καθόλου με συμβάντα και μεθόδους διότι τα έχουμε ήδη δει πολλές φορές.

Παρατήρηση:

Στην περίπτωση που έχουμε παραπάνω από μία θεματικές ενότητες προς επιλογή (συνήθης περίπτωση), αντιμετωπίζουμε το εξής πρόβλημα: Λόγω του ότι μόνο ένα πλήκτρο επιλογής μπορεί να είναι «πατημένο» (όταν επιλέγουμε κάποιο, αυτόματα όλα τα υπόλοιπα «απο-επιλέγονται») πώς θα μπορέσουμε να διαχωρίσουμε τις ενότητες; Το παραπάνω πρόβλημα παρουσιάζεται ως εξής:

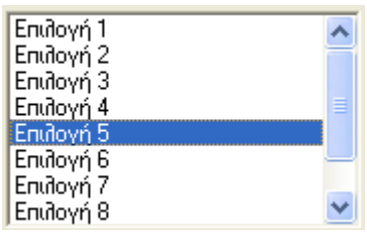


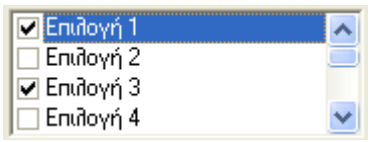
Στο διπλανό σχήμα έχουμε το προαναφερθέν πρόβλημα. Η VB δεν γνωρίζει ότι εμείς έχουμε 2 διαφορετικές κατηγορίες για πλήκτρα επιλογής. Έτσι, μας επιτρέπει να κάνουμε μόνο μία επιλογή (πράγμα καθόλου βολικό, όπως βλέπετε). Πώς επιλύεται αυτό το πρόβλημα; Απλά! Έχουμε ήδη μιλήσει για το στοιχείο ελέγχου Frame (Πλαίσιο). Ο ρόλος του είναι ακριβώς αυτός! Ομαδοποίηση στοιχείων ελέγχου. Σχεδιάζοντας στην φόρμα μας 2 Frames (ή και PictureBoxes, όπως έχουμε ήδη πει) και τοποθετώντας μέσα σε κάθε Frame μια ομάδα, έχουμε το εξής αποτέλεσμα:



Το πρόβλημά μας δεν υφίσταται πλέον. Ομαδοποίησαμε τα OptionButtons της εφαρμογής μας και έτσι μπορούμε να πραγματοποιήσουμε μια επιλογή από κάθε κατηγορία.

ListBox (Πλαίσιο λίστας):





Το πλαίσιο λίστας μοιάζει πάρα πολύ με το πλαίσιο αναδιπλούμενης λίστας (ComboBox), τόσο στον τρόπο της λειτουργίας και της διαχείρισης, όσο και στην εμφάνιση.

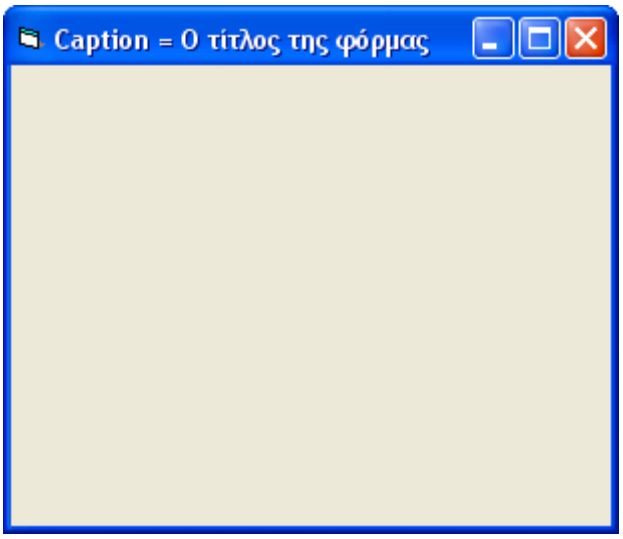
Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), BackColor, Enabled, Font, ForeColor, Height, Left, MousePointer, ToolTipText, Top, Visible, Width	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Locked	Κλειδώνει το ListBox ώστε να μην μπορεί να γίνει επιλογή.	True ή False	True: Κλειδωμένο False: Ξεκλείδωτο (ελεύθερο)
Sorted	Προσδιορίζει αν τα περιεχόμενα του ListBox θα είναι ταξινομημένα ή όχι.	True ή False	True: Ταξινομημένα False: Ως εισήχθησαν
MultiSelect	Προσδιορίζει τον τρόπο επιλογής των στοιχείων της λίστας	0 ή 1 ή 2	0: Δεν επιτρέπεται η πολλαπλή επιλογή 1: Επιτρέπεται η πολλαπλή επιλογή με απλό κλικ. 2: Επιτρέπεται η πολλαπλή επιλογή με χρήση του πλήκτρο [Shift] ή [Ctrl].
Style	Προσδιορίζει τον τρόπο εμφάνισης.	0 ή 1	0: Standard: (Όπως εμφανίζεται στο αριστερό σχήμα) 1: CheckBox: (Όπως εμφανίζεται στο δεξιό σχήμα)
ListIndex	Προσδιορίζει τον αυτόματο κωδικό καταχώριση ς	Δεδομένου ότι οι N καταχωρίσεις αριθμούνται από 0...N-1	Σύνταξη (έστω ότι το εργαλείο ονομάζεται List1): AnInteger = List1.Listindex
ListCount	Επιστρέφει το πλήθος των καταχωρίσεων N.	Είναι μόνο για ανάγνωση.	Σύνταξη (έστω ότι το εργαλείο ονομάζεται List1): AllRegs = List1.ListCount

List	Επιστρέφει την ίδια την καταχώριση που επιλέγουμε βάση του ListIndex	Ως όρισμα δέχεται το όρισμα του ListIndex.	Σύνταξη (έστω ότι το εργαλείο ονομάζεται List1): MyChoice = List1.List(List1.ListIndex)

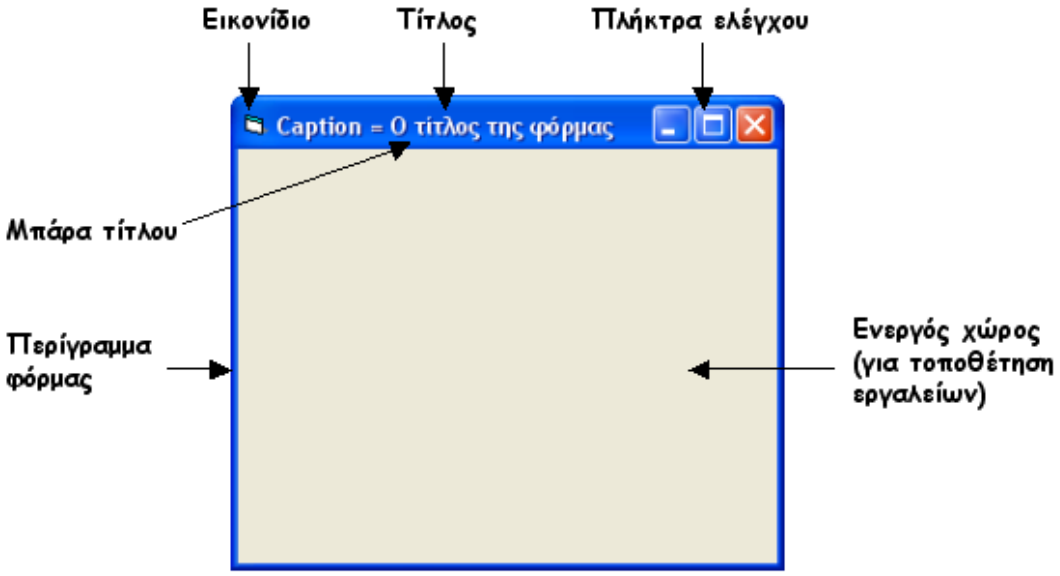
Οι μέθοδοι και τα συμβάντα είναι ίδια με αυτές του ComboBox. Ωστόσο, υπάρχει ένα συμβάν που χρησιμοποιεί το ListBox όταν ως Style έχει οριστεί η τιμή CheckBox:

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «List1»)
ItemCheck	Όταν τσεκάρουμε μια καταχώριση στο πλαίσιο λίστας. Δεξιά η τιμή του «Item» είναι ο αριθμός καταχώριση ς της επιλογής που τσεκάρουμε.	Private Sub List1_ItemCheck(Item As Integer) End Sub

Form (Φόρμα):



Η φόρμα αποτελεί το πιο σημαντικό στοιχείο ελέγχου. Πάνω της τοποθετούνται όλα τα εργαλεία τα οποία συνθέτουν το τελικό αποτέλεσμα (Project). Έχει πολλές ιδιότητες, συμβάντα και μεθόδους. Θα παρατηρήσατε ήδη ότι η φόρμα περιλαμβάνει εκ φύσεως ένα μικρό εικονίδιο (image), έναν τίτλο (Label), 3 CommandButtons καθώς και κάποια «αόρατα» στοιχεία χειρισμού.



Στο παραπάνω σχήμα αναγράφονται οι ονομασίες των στοιχείων που βλέπετε. Η παραπάνω μορφή της φόρμας (απλή μορφή) ενσωματώνει τα στοιχεία ελέγχου που αναφέραμε προηγουμένως ως ιδιότητες. Πριν ξεκινήσουμε να τις βλέπουμε θα παραθέσουμε κάποια μικρά σχόλια για αυτές.

Εικονίδιο: Οι περισσότερες, αν όχι όλες οι εφαρμογές, τα προγράμματα και τα παιχνίδια που έχετε εγκατεστημένα στον προσωπικό σας Η/Υ, παρουσιάζονται στην επιφάνεια εργασίας των Windows με ένα χαρακτηριστικό εικονίδιο. Το εικονίδιο αυτό δεν είναι τίποτα περισσότερο από αυτό που θα δώσετε εσείς στην φόρμα, κατά τη σύνταξη ενός προγράμματος.

Τίτλος: Ο τίτλος (ο οποίος δίνεται από την ιδιότητα Caption) πρέπει να περιγράφει συνοπτικά τη γενική λειτουργία του προγράμματος, ή όταν το πρόγραμμα που γράφουμε αποτελείται από περισσότερες της μιας φόρμες, να περιγράφει συνοπτικά το ρόλο της συγκεκριμένης φόρμας στο συνολικό project.

Πλήκτρα ελέγχου: Τα πλήκτρα ελέγχου (τα οποία, αν επιθυμούμε, μπορούμε να απενεργοποιήσουμε) είναι διαχειριστικά εργαλεία της φόρμας. Με τη χρήση τους μπορούμε να «σμικρύνουμε» τη φόρμα, να τη μεγιστοποιήσουμε ώστε να καλύπτει όλη την οθόνη, ή τέλος να την κλείσουμε.

Μπάρα τίτλου: Περιλαμβάνει τα παραπάνω στοιχεία. Ο ρόλος της επίσης είναι να επιτρέπει την μετακίνηση της φόρμας πάνω στην οθόνη.

Περίγραμμα φόρμας: Έχει το ίδιο στυλ με τον τίτλο και χρησιμοποιείται για να μπορούμε να αλλάζουμε δυναμικά το μέγεθος της φόρμας.

Ενεργός χώρος: Είναι ο υπόλοιπος (και μεγαλύτερος) χώρος της φόρμας πάνω στον οποίο μπορούμε να σχεδιάσουμε, να τοποθετήσουμε εργαλεία κ.τ.λ.

Ιδιότητα	Επεξήγηση	Όρισμα	Αποτέλεσμα
(Name), Appearance, AutoRedraw, BackColor, Caption, Enabled, FillColor, FillStyle, Font, FontTransparent, ForeColor, Left, Top, Height, Width, MousePointer, Picture, ScaleMode, ScaleLeft, ScaleHeight, ScaleWidth, Visible	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
BorderStyle	Καθορίζει ουσιαστικά το στυλ εμφάνισης της φόρμας, αν περιλαμβάνει τα πλήκτρα ελέγχου, αν επιτρέπεται η μετακίνησή της κτλ	0, 1, 2, 3, 4, 5	0: None: Η φόρμα εμφανίζεται μόνο ως ενεργός χώρος. 1: Fixed Single: Η φόρμα εμφανίζεται με τίτλο, εικονίδιο, ενώ από τα πλήκτρα ελέγχου υπάρχει μόνο το πλήκτρο κλεισίματος. Δεν επιτρέπει αλλαγή μεγέθους της φόρμας. 2: Sizable: Η κανονική μορφή της φόρμας

			όπως την είδαμε παραπάνω. 3: Fixed Dialog: Μοιάζει ιδιαίτερα με την Fixed Single. 4: Fixed Tool Window: Η μορφή της φόρμας είναι κάπως «Light», μοιάζει με την Fixed Single, δεν επιτρέπει αλλαγή μεγέθους. 5: Sizable ToolWindow: Είναι σαν την προηγούμενη μορφή αλλά επιτρέπει την αλλαγή μεγέθους της φόρμας.
ControlBox	Εμφανίζει ή αποκρύβει τα πλήκτρα ελέγχου της φόρμας.	True ή False	True: Εμφανίζει τα πλήκτρα ελέγχου. False: Αποκρύβει τα πλήκτρα ελέγχου.
Icon	Θέτει το εικονίδιο της φόρμας.	Ico, cur τύπο αρχείων εικόνας.	Εμφανίζει το εικνίδιο.
MaxButton	Ενεργοποιεί ή απενεργοποεί το πλήκτρο μεγιστοποίησης της φόρμας (από τα πλήκτρα ελέγχου)	True ή False	True: Ενεργοποιεί False: Απενεργοποιεί
MinButton	Ενεργοποιεί ή απενεργοποεί το πλήκτρο ελαχιστοποίησης της φόρμας (από τα πλήκτρα ελέγχου)	True ή False	True: Ενεργοποιεί False: Απενεργοποιεί
MDIChild	Προσδιορίζει αν μια φόρμα υπόκειται μιας άλλης φόρμας. Η φόρμα στην οποία μπορεί να υπόκεινται άλλες φόρμες δεν είναι η απλή φόρμα που εξετάζουμε. Ονομάζεται MDI Form και πρέπει να την προσθέσουμε ξεχωριστά.	True ή False	True: Την υποτάσσει στην MDI Form False: Η φόρμα παραμένει στην ελεύθερη μορφή της
Moveable	Προσδιορίζει αν η φόρμα θα μπορεί να μετακινηθεί από τον χρήστη στην οθόνη.	True ή False	True: Επιτρέπει την μετακίνηση. False: Απαγορεύει την μετακίνηση.
StartUpPosition	Προσδιορίζει την θέση εμφάνισης της φόρμας στην οθόνη κατά την εκτέλεση.	0, 1, 2, 3	0: Manual: Όπου έχει προσδιορίσει ο χρήστης. 1: CenterOwner: Στο

			κέντρο της φόρμας που την περιέχει (Για MDI φόρμες κυρίως) 2: CenterScreen: Στο κέντρο της οθόνης. 3: WindowsDefault: Στην προεπιλεγμένη από τα Windows θέση.
WindowState	Προσδιορίζει το μέγεθος της φόρμας κατά την εκκίνηση.	0, 1, 2	0: Το μέγεθος που έχει δώσει ο χρήστης 1: Ελαχιστοποιημένη 2: Μεγιστοποιημένη

Οι μέθοδοι της φόρμας παρουσιάζονται παρακάτω

Μέθοδος	Επεξήγηση	Ορίσματα	Σύνταξη (Θεωρώντας ότι το όνομα της φόρμας είναι Form1)
Circle, Cls, Line, Move, Point, Pset, Refresh, Scale, ScaleX, ScaleY, SetFocus, TextHeight, TextWidth, ZOrder	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν	Όπως προαναφέρθηκαν
Hide	Αποκρύβει οπτικά τη φόρμα.	Κανένα	Form1.Hide
PopUpMenu	Χρησιμοποιεί κάποιο MENU ως αναδυόμενο MENU	Το όνομα του MENU	Form1.PopUpMenu [Ονομα]
PrintForm	Εκτυπώνει τη φόρμα	Κανένα	Form1.PrintForm
Show	Εμφανίζει τη φόρμα	Κανένα	Form1.Show

Σημείωση

MENU: Δεν θεωρείται στοιχείο ελέγχου αλλά στην πραγματικότητα δεν διαφέρει καθόλου από αυτά. Το έχουμε δει όλοι σε προγράμματα που δουλεύουμε. Έχουμε δει και το Menu επιλογών της Visual Basic σε προηγούμενο κεφάλαιο. Το πώς δημιουργούμε και χειριζόμαστε ένα Menu θα το δούμε στο επόμενο κεφάλαιο.

Η φόρμα διαχειρίζεται πολλά συμβάντα. Τα συμβάντα γνωρίζετε ήδη πού θα τα βρείτε, οπότε τα γνωστά δεν θα αναφερθούν.

Συμβάν	Ενεργοποίηση	Αποτέλεσμα Εκτελεί τον κώδικα που έχουμε γράψει εντός της παρακάτω υπορουτίνας συμβάντος (Sub). (Θεωρώ ότι η ιδιότητα (Name) έχει ως όρισμα το «Form1»)
Activate	Ενεργοποιείται όταν η φόρμα γίνεται ενεργή, δηλαδή για παράδειγμα όταν πατήσουμε για πρώτη φορά πάνω της (εστιάζουμε πάνω της).	Private Sub Form_Activate() End Sub
DeActivate	Ενεργοποιείται όταν η φόρμα γίνεται ανενεργή, δηλαδή για παράδειγμα όταν εστιάσουμε την προσοχή μας σε άλλη φόρμα.	Private Sub Form_Deactivate() End Sub
Initialize	Ενεργοποιείται όταν η φόρμα «ξεκινάει» για πρώτη φορά.	Private Sub Form_Initialize() End Sub
Load	Ενεργοποιείται όταν η φόρμα «φορτώνει» για πρώτη φορά.	Private Sub Form_Load() End Sub
QueryUnload	Ενεργοποιείται όταν κλείνουμε την φόρμα από το πλήκτρο [x] των πλήκτρων ελέγχου.	Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer) End Sub
Resize	Ενεργοποιείται όταν μεταβάλλουμε το μέγεθος της φόρμας.	Private Sub Form_Resize() End Sub
Terminate	Ενεργοποιείται όταν κλείσουμε όλες τις φόρμες ενός project. Ουσιαστικά θα έπρεπε να είναι τερματισμός προγράμματος και όχι τερματισμός φόρμας.	Private Sub Form_Terminate() End Sub
Unload	Ενεργοποιείται όταν η φόρμα «αποφορτώνεται».	Private Sub Form_Unload(Cancel As Integer) End Sub

Παρατήρηση:

Κατά τη σύνταξη κώδικα σε μια φόρμα, και όταν αναφερόμαστε στην τρέχουσα φόρμα, μπορούμε, αντί να πληκτρολογούμε (πχ)

Form1.PrintForm

να χρησιμοποιούμε την ισοδύναμη σύνταξη

Me.PrintForm

Η Visual Basic καταλαβαίνει οτι η δεσμευμένη λέξη «Me» αναφέρεται στην τρέχουσα φόρμα.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Τα στοιχεία ελέγχου έχουν ιδιότητες, μεθόδους και συμβάντα.
2. Οι ιδιότητες βρίσκονται στο Properties Window. Οι μέθοδοι (και οι ιδιότητες) εμφανίζονται κατά τη σύνταξη κώδικα όταν, αφού γράψουμε το όνομα του στοιχείου ελέγχου, πατήσουμε την τελεία « . ».
3. Τα συμβάντα, κατά τη σύνταξη κώδικα, βρίσκονται σε μια αναδιπλούμενη λίστα πάνω δεξιά από το παράθυρο σύνταξης κώδικα.
4. Τα στοιχεία ελέγχου βρίσκονται στο ToolBox το οποίο βρίσκεται προσαρμοσμένο στην αριστερή πλευρά της οθόνης.
5. Όταν αναφερόμαστε σε ένα εργαλείο πρέπει πάντα να ακολουθούμε πιστά τον τρόπο σύνταξης: ΟνομαΕργαλείου.Ιδιότητα = τιμή

ΕΡΩΤΗΣΕΙΣ

1. Πώς μπορώ να εισάγω ένα εργαλείο στο project μου;
2. Από πού μπορώ να αλλάξω τη θέση και το μέγεθός τους;
3. Η ιδιότητα Caption υπάρχει σε όλα τα εργαλεία;
4. Πως μπορώ να ενεργοποιήσω μια διαδικασία απλά περνώντας το δείκτη του mouse από πάνω του;
5. Σε ποιο τμήμα κώδικα και ποιου εργαλείου μπορώ να προσθέσω κώδικα ο οποίος να εκτελείτε αυτόματα κατά την εκκίνηση;
6. Πως μπορώ να αποκρύψω τα δεδομένα εισαγωγής σε ένα πλαίσιο κειμένου για προστασία;
7. Πώς μπορώ να απαγορεύσω σε κάποιο χρήστη να εισάγει τον αριθμό 3 σε ένα πλαίσιο κειμένου; Από ποιο συμβάν;
8. Πως μπορώ να εντοπίσω ένα αρχείο μουσικής στο δίσκο μέσα από μια εφαρμογή που φτιάχνω; Ποια εργαλεία είναι απαραίτητα;
9. Πώς μπορώ να κεντράρω μια φόρμα στην οθόνη;
10. Ποιες οι οπτικές διαφορές μεταξύ ComboBox και ListBox;
11. Ποιες είναι οι διαφορές μεταξύ CheckBox και OptionButton;
12. Πού μπορώ να προβάλλω ένα γράφημα;
13. Πώς μπορώ να ομαδοποιήσω κάποια στοιχεία ελέγχου;
14. Πώς μπορώ να εκτελέσω αυτόματα μια διαδικασία 12 λεπτά μετά την έναρξη του προγράμματος;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα project με 1 πλαίσιο κειμένου, 1 πλήκτρο και ένα ComboBox. Ο χρήστης θα γράφει το όνομά του και πατώντας το πλήκτρο θα το προσθέτει στο ComboBox. Αντίστροφα όταν ο χρήστης επιλέξει ένα όνομα από το ComboBox αυτό θα εμφανίζεται αυτόματα και στο TextBox.

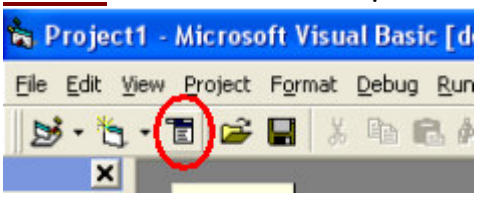
Κεφάλαιο 3

Περισσότερα στοιχεία ελέγχου

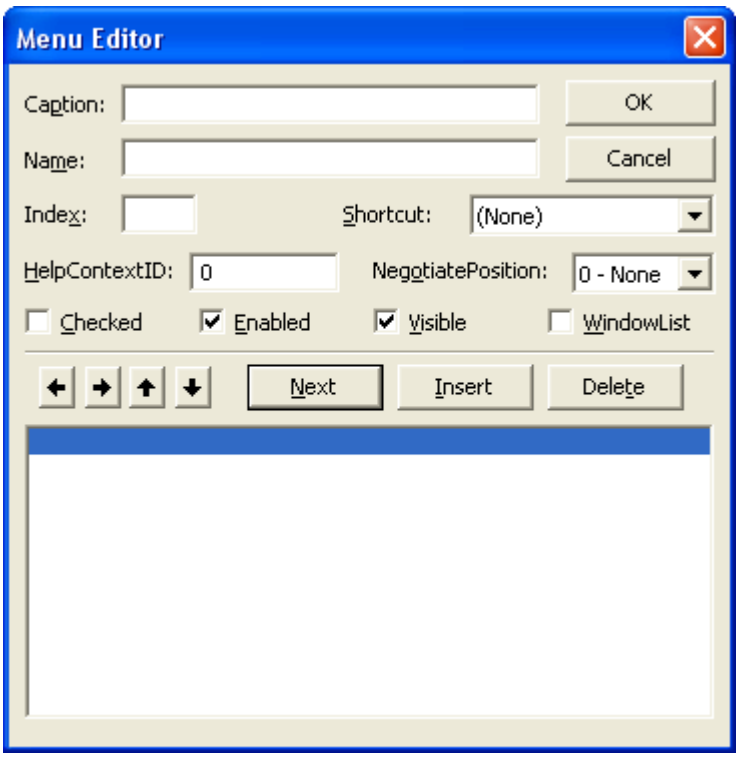
Πέρα από τα στοιχεία ελέγχου που συναντήσαμε στην μπάρα ToolBox υπάρχουν εκατοντάδες ακόμα που δημιουργήθηκαν είτε από την MicroSoft είτε από ανεξάρτητους (τρίτους) κατασκευαστές, από τα οποία άλλα πωλούνται και άλλα διατίθενται δωρεάν.

Σε αυτό το κεφάλαιο, θα αναλύσουμε το εργαλείο Menu το οποίο είναι ένα standard εργαλείο της Visual Basic (αν και δεν βρίσκεται στο ToolBox), και κάποια ακόμα, πολύ χρήσιμα όπως θα δούμε στην συνέχεια.

Menu : Μενού επιλογών



Το εικονίδιό του, όπως φαίνεται στην μπάρα εργαλείων της VB, κάτω ακριβώς από το κεντρικό menu επιλογών



Αριστερά βλέπουμε το παράθυρο δημιουργίας του Menu επάνω στην τρέχουσα φόρμα. Ο Editor του Menu ανοίγει όταν εμείς πατήσουμε το παραπάνω εικονίδιο. Θα αναλύσουμε στην συνέχεια με ένα απλό παράδειγμα τις λειτουργίες που είναι διαθέσιμες για τη δημιουργία ενός menu επιλογών.

Caption: Γράφουμε τον τίτλο που θα φαίνεται στο menu (π.χ. File, Open, Print κτλ).

Name: Είναι η γνωστή μας ιδιότητα (Name) η οποία θα χρησιμοποιείται για κλήση συμβάντων.

Shortcut: Μπορούμε να ορίσουμε ένα συνδυασμό πλήκτρων για συντόμευση κλήσης της επιλογής που επιθυμούμε στο menu (π.χ. Για την εντολή Open μπορούμε να θέσουμε Ctrl + O).

Checked: Προσδιορίζουμε αν επιθυμούμε την ένδειξη τσεκαρίσματος στην επιλογή που κάνουμε. Σε αρκετές περιπτώσεις μπορεί να φανεί πολύ λειτουργικό.

Enabled: Όταν αποεπιλέγουμε την παραπάνω επιλογή, αυτή θα παρουσιάζεται απενεργοποιημένη. Χρησιμοποιείται όταν θέλουμε για κάποιο λόγο να αποκόψουμε από το χρήστη κάποιες λειτουργίες.

Visible: Η γνωστή ιδιότητα, προσδιορίζει αν η επιλογή θα είναι ορατή από το χρήστη.

Next: Καταχωρίζει ότι προσθέσαμε και ετοιμάζει την φόρμα για νέα καταχώριση.

Insert: Παρεμβάλλει μια καταχώριση επιλογής ανάμεσα σε άλλες.

Delete: Διαγράφει μια καταχωρισμένη επιλογή.

Αριστερό βελάκι: Μας μεταφέρει ένα επίπεδο πίσω *

Δεξί βελάκι: Μας μεταφέρει ένα επίπεδο εμπρός *

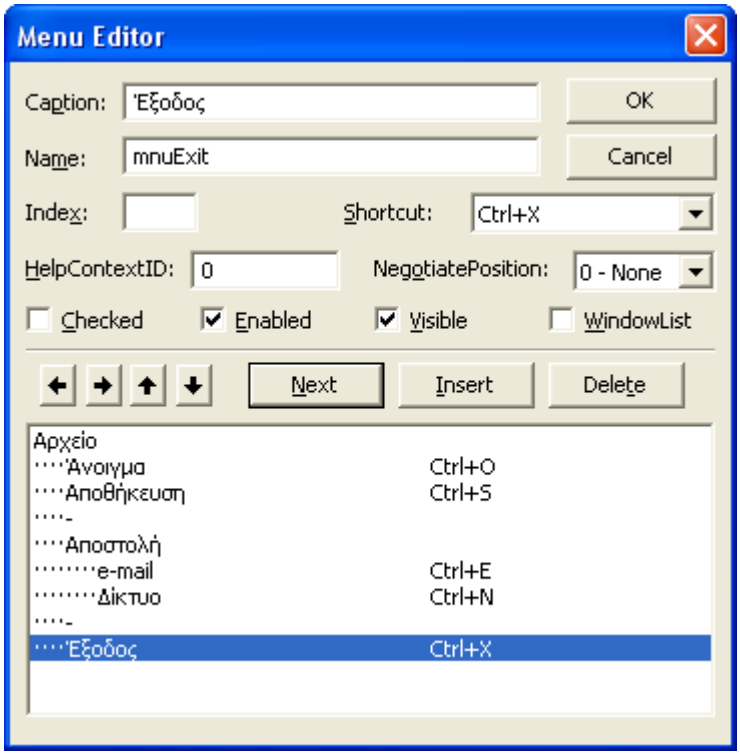
Πάνω βελάκι: Μας επιτρέπει την πλοήγηση ανάμεσα στις καταχωρίσεις.

Κάτω βελάκι: Μας επιτρέπει την πλοήγηση ανάμεσα στις καταχωρίσεις.

OK: Κατοχυρώνει όλα όσα προσθέσαμε και κλείνει την φόρμα.

Cancel: Κλείνει την φόρμα χωρίς να αποδεχτεί τις καταχωρίσεις.

Το Menu είναι ένα ιδιόρρυθμο εργαλείο. Λειτουργεί με το σκεπτικό του δενδροειδούς διαγράμματος. Για αυτό, οι καταχωρίσεις γίνονται σε επίπεδα. Δείτε τα παρακάτω σχήματα που παρουσιάζουν ένα Menu στην φάση της δημιουργίας και στην φάση της εκτέλεσης.



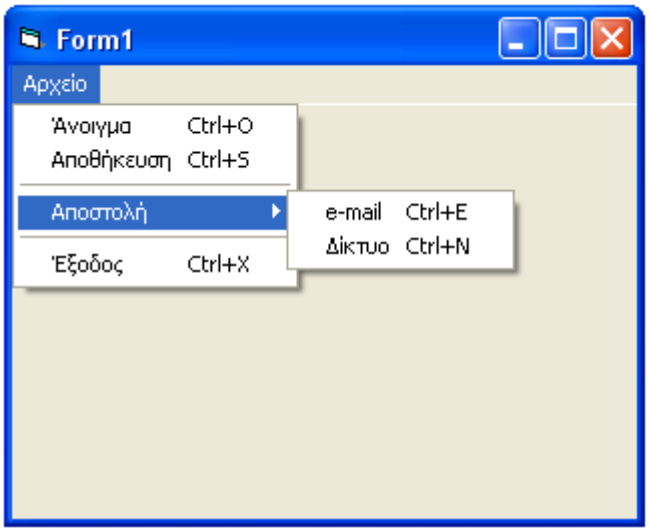
(*) Προσέξτε ιδιαίτερα το λευκό πλαίσιο κειμένου. Εκεί μπορείτε να παρατηρήσετε τα επίπεδα που προαναφέραμε. Το [Αρχείο] βρίσκεται στο επίπεδο «Μηδέν». Αυτό σημαίνει ότι θα εμφανιστεί ως γενικός τίτλος κατηγορίας στο Menu, και δεν θα περιλαμβάνει κώδικα εκτέλεσης.

Αντίθετα οι επιλογές [Άνοιγμα] και [Αποθήκευση] «κρέμονται» από την επιλογή-τίτλο [Αρχείο] και βρίσκονται στο επίπεδο «1». Γενικά όσες επιλογές του Menu «κρέμονται» από κάποια άλλη, τότε αυτές περιλαμβάνουν κώδικα σε αντίθεση με αυτή από την οποία εξαρτώνται.

Έτσι, όπως φαίνεται στο παραπάνω σχήμα, οι επιλογές [e-mail] και [Δίκτυο] περιλαμβάνουν κώδικα (εκτελούνται) σε αντίθεση με αυτή που τις φέρει, δηλαδή την [Αποστολή], η οποία εμφανίζεται απλά ως γενικότερος τίτλος αυτών. Οι επιλογές [e-mail] και [Δίκτυο] βρίσκονται στο επίπεδο «2». Μπορούμε να έχουμε πάρα πολλά επίπεδα ανά κατηγορία.

Θα παρατηρήσατε επίσης τις συντομεύσεις του τύπου [Ctrl+O] που χρησιμοποιήσαμε. Πέρα από το πλήκτρο [Ctrl] μπορείτε να χρησιμοποιήσετε τα [Shift], [Alt], {F keys} σε πολλούς συνδυασμούς.

Παρακάτω, στο σχήμα που ακολουθεί, βλέπετε το οπτικό αποτέλεσμα των παραπάνω και τη σχέση των επιπέδων που αναλύσαμε. Παρατηρήστε επίσης ότι εκεί που εισάγαμε ως Caption την « - », εμφανίζεται ένα διαχωριστικό (το οποίο δεν περιλαμβάνει κώδικα, δηλαδή δεν είναι επιλέξιμο) που χρησιμοποιείται για ομαδοποίηση λειτουργιών (οπτικά καλύτερο αποτέλεσμα).



Πατώντας την επιλογή [Αρχείο] εμφανίζεται το υπομενού του. Το ίδιο συμβαίνει και στην επιλογή [Αποστολή].

Τα ονόματα που δίνουμε στην ιδιότητα Caption πρέπει να είναι η περιγραφή της λειτουργίας που πρόκειται να εκτελεστεί ώστε να διευκολύνεται ο χρήστης.

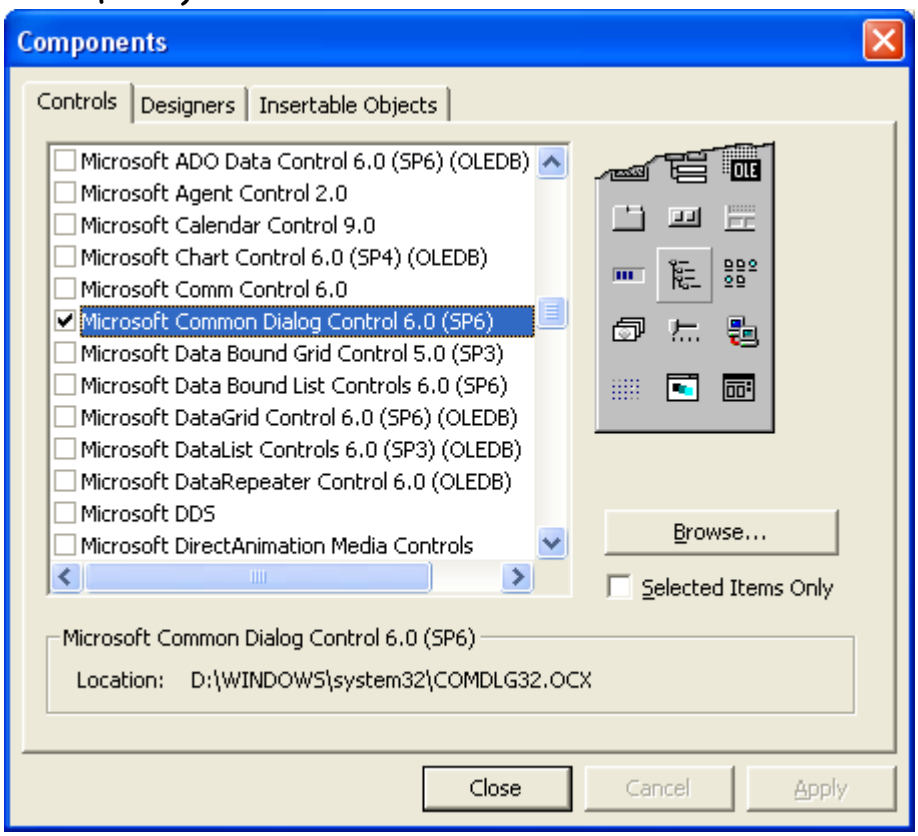
Πώς όμως εισάγουμε εκτελέσιμο κώδικα στο συμβάν κλήσης κάποιας επιλογής; Απλά, κατά την φάση σχεδίασης, κάνουμε κλικ στην επιλογή που επιθυμούμε να προσθέσουμε κώδικα, από όπου και εμφανίζεται η γνωστή πλέον υπορουτίνα συμβάντος:

```
Private Sub mnuOpen_Click()

End Sub
```

Στην παραπάνω υπορουτίνα κάναμε κλικ στην επιλογή [Άνοιγμα] όπου είχαμε θέσει στην ιδιότητα «Name» την τιμή «mnuOpen».

Το στοιχείο ελέγχου Menu δεν περιλαμβάνει άλλα συμβάντα κλήσης, δεν έχει καμία μέθοδο, ενώ οι ιδιότητές του είναι αυτές που περιγράψαμε παραπάνω.

Microsoft Common Dialog Control : (Εργαλείο «κοινών διαλόγων»)



Όπως βλέπετε στην παραπάνω εικόνα «ψάξαμε» στο παράθυρο διαλόγου με το όνομα «Components» και τσεκάραμε την επιλογή «Microsoft Common Dialog Control». Αλλά πώς εμφανίστηκε το παραπάνω παράθυρο;

1ος τρόπος εμφάνισης: Στο ToolBox κάνουμε δεξί κλικ και εμφανίζεται ένα «αναδυόμενο» (pop-up) menu από όπου επιλέγουμε την επιλογή [Components…].
2ος τρόπος: Από το βασικό menu της Visual Basic (το οποίο είδαμε στο 1ο κεφάλαιο), από την βασική κατηγορία [Project] επιλέγουμε [Components…] ή επιλεκτικά πατάμε Ctrl+T.

Επειδή ο ελληνικός όρος μπορεί να αποδειχθεί ατυχής, από εδώ και πέρα θα αναφερόμαστε στο συγκεκριμένο εργαλείο με το όνομα «Common Dialog».

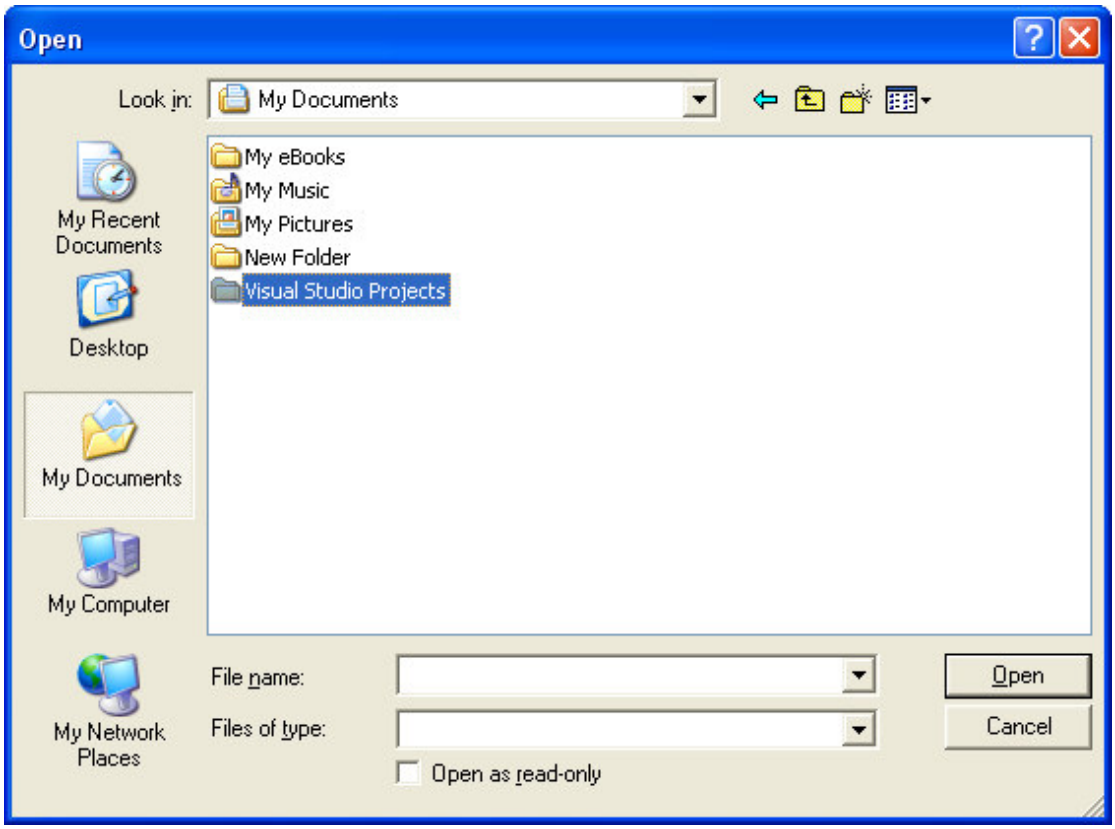
Όμως, τι ακριβώς μας παρέχει το παραπάνω εργαλείο; Η απάντηση είναι «εργονομία στον τελικό χρήστη». Το Common Dialog μας επιτρέπει εύκολα και γρήγορα να εντοπίσουμε κάποιο αρχείο στον δίσκο για εγγραφή ή για ανάγνωση, να επιλέξουμε το χρώμα ή / και την γραμματοσειρά της

αρεσκείας μας, να προσδιορίσουμε τον εκτυπωτή στον οποίο θέλουμε να εκτυπώσουμε κ.α.

Έχοντας ήδη αναφέρει πώς μπορούμε να εισάγουμε το εργαλείο στο ToolBox, επιλέγοντάς το, το προσθέτουμε στην φόρμα (ή κάνοντας διπλό κλικ πάνω του).

Στην συνέχεια θα αναλύσουμε τις λειτουργίες του οι οποίες στην ουσία δεν είναι τίποτα άλλο παρά οι μέθοδοι του εργαλείου. Θα θεωρήσουμε ότι ως όνομα (ιδιότητα «Name») του εργαλείου έχουμε θέσει την «CD1».

ShowOpen : Το χρησιμοποιούμε για ανεύρεση κάποιου αρχείου στο δίσκο για άνοιγμα.
Σύνταξη: `CD1.ShowOpen`
Αποτέλεσμα: Εμφάνιση του παρακάτω πλαισίου διαλόγου «Άνοιγμα αρχείου».



Τα στοιχεία (ιδιότητες) που μπορούμε να αλλάξουμε είναι ο εμφανιζόμενος τίτλος, το προεπιλεγμένο «File name», και το «Files of type». Με το τελευταίο μπορούμε να προσδιορίσουμε ποιοι τύποι αρχείου θα προβάλλονται ως διαθέσιμοι για άνοιγμα.

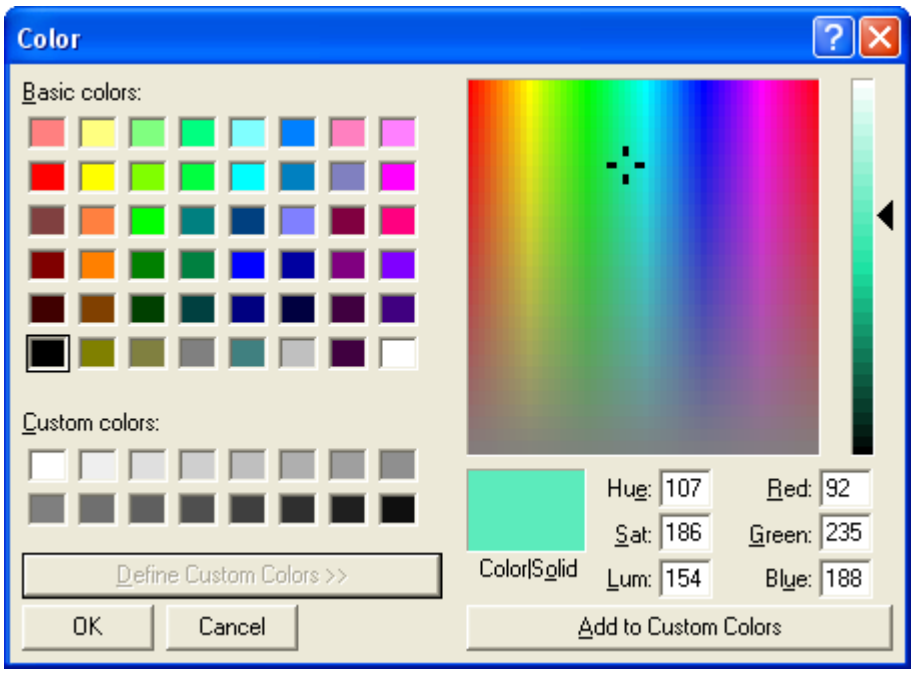
Ιδιότητες:

DialogTitle: Κείμενο το οποίο θα αναγράφεται ως τίτλος.
Παράδειγμα σύνταξης:
```
CD1.DialogTitle = "Άνοιγμα αρχείου συντεταγμένων"
```
FileName: Το όνομα του προεπιλεγμένου αρχείου (ή του αρχείου που επιλέγουμε).
Παράδειγμα σύνταξης:
```
CD1.FileName = "MyCords.txt"
```
Filter: Προσδιορίζει τον/τους τύπους των αρχείων που θα εμφανίζονται.
Παράδειγμα σύνταξης:
```
CD1.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία συντεταγμένων *.seq (*.seq)|*.seq"
```

Η δήλωση CD1.ShowOpen πρέπει να έπεται των παραπάνω δηλώσεων ιδιοτήτων.

ShowSave : Το χρησιμοποιούμε για να βρούμε κάποιο σημείο στο δίσκο για αποθήκευση αρχείου.
Σύνταξη: `CD1.ShowSave`
Επειδή οπτικά είναι πανομοιότυπο με το παραπάνω (αλλά και λειτουργικά) δεν θα αναλύσουμε τις ιδιότητές του.

ShowColor : Το χρησιμοποιούμε για να επιλέξουμε κάποιο χρώμα.



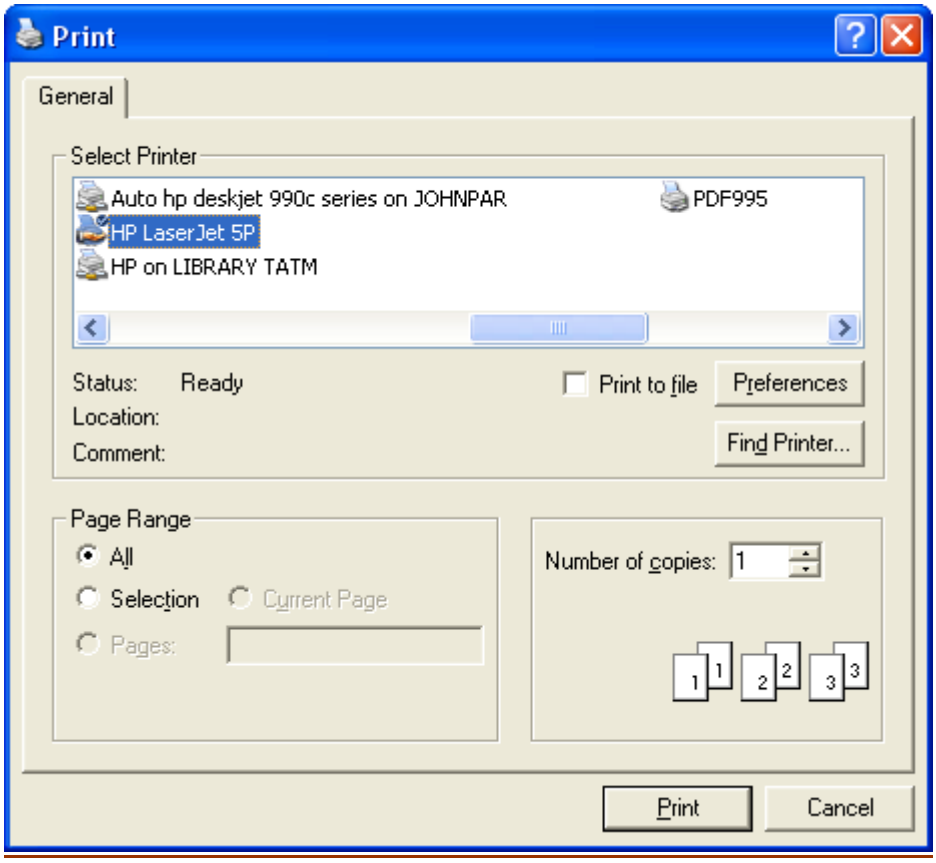
Σύνταξη: `CD1.ShowColor`

Να επισημάνουμε εδώ ότι όλες οι επιλογές που κάνουμε πρέπει να καταχωριστούν σε κάποια μεταβλητή για να τις χρησιμοποιήσουμε (όπως θα δούμε και σε παρακάτω κεφάλαιο).

Ιδιότητες:
Color: Η τιμή του χρώματος που θα μας επιστραφεί μετά την επιλογή που θα κάνουμε.

```
MyColor = CD1.Color
```

ShowPrinter : Μας επιτρέπει να επιλέξουμε έναν από τους εγκατεστημένους εκτυπωτές.



Σύνταξη: CD1.ShowPrinter

Ιδιότητες:
Copies: Θέτει το πλήθος των αντιγράφων που επιθυμούμε.

```
CD1.Copies = 5
```

FromPage: Θέτει την σελίδα εκκίνησης της εκτύπωσης.

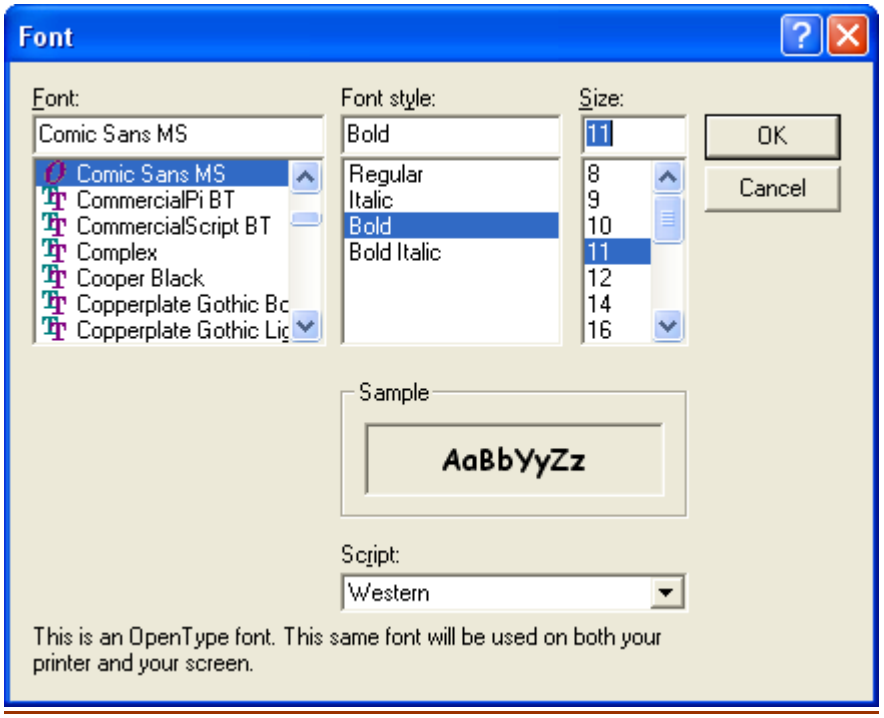
```
CD1.FromPage = 1
```

ToPage: Θέτει την σελίδα λήξης της εκτύπωσης.

```
CD1.ToPage = 12
```

Σημείωση: Για να λειτουργήσουν οι τελευταίες δύο ιδιότητες που αναφέραμε θα πρέπει να θέσουμε προηγουμένως στην ιδιότητα Flags την παρακάτω τιμή.
CD1.Flags = cdlPDPageNums

ShowFont : Μας επιτρέπει να επιλέξουμε την γραμματοσειρά που επιθυμούμε να χρησιμοποιήσουμε.



Προσοχή: Για να γίνουν διαθέσιμες οι γραμματοσειρές του συστήματος πρέπει οπωσδήποτε προηγουμένως να θέσουμε στην ιδιότητα Flags την παρακάτω τιμή:
CD1.Flags = cdlCFBoth

Σύνταξη: `CD1.ShowFont`

Ιδιότητες:
FontName: Μας επιστρέφει το όνομα της γραμματοσειράς που επιλέξαμε.
`MyFont = CD1.FontName`

Υπάρχουν βέβαια και άλλες ιδιότητες για το Common Dialog αλλά εμείς παρουσιάσαμε τις πιο λειτουργικές.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Τα στοιχεία ελέγχου Menu και Common Dialog δεν βρίσκονται στο ToolBox. Το Menu ωστόσο υπάρχει ως έτοιμο εργαλείο μέσα στην VB σε αντίθεση με το Common Dialog το οποίο πρέπει να προσθέσουμε εμείς.
2. Σε κάθε γενική κατηγορία Menu μπορούμε να «αναρτήσουμε» πολλές υποκατηγορίες. Μόνο όμως οι τελικές επιλογές εκτελούνται.
3. Η ιδιότητα «Name» του Menu περιέχει το όνομα κάθε επιλογής όπου μπορούμε στο συμβάν κλικ να προσθέσουμε κώδικα.
4. Το Common Dialog διαιρείται σε υποκατηγορίες. Τα ShowOpen και ShowSave επειδή προσφέρουν ευχρηστία μπορούν να χρησιμοποιηθούν αντί των DriveListBox, DirListBox και FileListBox.
5. Η ιδιότητα Flags (την οποία δεν αναλύσαμε) είναι απαραίτητη μόνο με τη σχετική αναφερόμενη δήλωση στην υποκατηγορία ShowFont. Στις υπόλοιπες είναι προαιρετική ανάλογα με τον τρόπο χρήσης του εργαλείου.

ΕΡΩΤΗΣΕΙΣ

1. Μπορώ να δημιουργήσω Menu με μη προσβάσιμες επιλογές;
2. Μπορώ να χρησιμοποιήσω το Common Dialog για να επιλέξω κάποιο Scanner το οποίο έχω εγκατεστημένο;
3. Πώς μπορώ να εκτυπώσω επιλεκτικά ένα σύνολο σελίδων από ένα κείμενο;
4. Μπορώ να καλέσω την επιλογή ShowColor μέσα από κάποια επιλογή ενός Menu;
5. Ποιες μεθόδους έχει το εργαλείο Menu;
6. Μπορώ το ίδιο Common Dialog να το χρησιμοποιήσω μια φορά για την μέθοδο ShowSave και μια φορά για την μέθοδο ShowPrint, ή πρέπει να προσθέσω περισσότερα εργαλεία Common Dialog στην φόρμα μου;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα νέο project και προσθέστε όλα τα εργαλεία που βρίσκονται στο ToolBox (από ένα). Ονομάστε τα όπως θέλετε. Δημιουργήστε ένα Menu με 2 κύριες κατηγορίες, την [Show] και την [Hide]. Σε κάθε μια από τις παραπάνω δημιουργήστε από μια επιλογή που να αναφέρεται σε κάθε εργαλείο (π.χ. Label, CommandButton κτλ). Στο Menu [Show], σε κάθε επιλογή προσθέστε κώδικα που να εμφανίζει το εργαλείο στο οποίο αναφέρεται, ενώ στο [Hide] να το εξαφανίζει.

Κεφάλαιο 4

1) Μεταβλητές (Τύποι, χρήση, δήλωση, σύνταξη)
2) Πίνακες μεταβλητών

1) Μεταβλητές

Μεταβλητές ονομάζονται εκείνες οι παράμετροι τις οποίες χρησιμοποιούμε για να αποθηκεύσουμε κάποια τιμή. Υπάρχουν εξ ορισμού διάφοροι τύποι μεταβλητών οι οποίοι χρησιμοποιούνται ανάλογα με τον τύπο των δεδομένων των οποίων επιθυμούμε να «αποθηκεύσουμε» την τιμή.

Τι σημαίνουν όμως τα παραπάνω, και γιατί έχουμε τύπους μεταβλητών; Η απάντηση είναι απλή. Πρώτον διότι μπορούμε εύκολα να κατηγοριοποιήσουμε τα δεδομένα μας και δεύτερον για οικονομία μνήμης, άρα και για την αύξηση της ταχύτητας επεξεργασίας.

Η σύνταξη των μεταβλητών είναι πολύ απλή υπόθεση. Φανταστείτε την εξής μαθηματική πράξη:

Result = 10 + 15

Η λέξη «Result» είναι μια μεταβλητή την οποία χρησιμοποιούμε για να αποθηκεύσουμε το αποτέλεσμα της παραπάνω πράξης.

Result = X1 + X4

Σε αυτή την δήλωση (η οποία είναι στην ουσία μια εντολή αντικατάστασης) τόσο η Result όσο και οι X1, X4 είναι μεταβλητές. Ας υποθέσουμε ότι η τιμή της X1 ισούται με 12 (X1 = 12) ενώ η X4 ισούται με 5. Τότε η Result παίρνει αυτόματα την τιμή 17. Η παραπάνω δηλαδή έκφραση μπορεί να καταχωρίζει στην τιμή της Result οποιοδήποτε αποτέλεσμα της παραπάνω μαθηματικής πράξης. Άρα η τιμή που φέρει δεν είναι σταθερή αλλά μεταβλητή (ανάλογα με τα δεδομένα των X1 και X4). Για αυτό το λόγο οι Result, X1, X4 ονομάζονται μεταβλητές.

Οι μεταβλητές μπορούν να καταχωρίσουν οποιαδήποτε τιμή (όχι μόνο αριθμητική). Για να καταλάβουμε ακριβώς τον τρόπο που δουλεύουν, θα

δούμε τους τύπους των μεταβλητών που μπορούμε να χρησιμοποιήσουμε, καθώς και τα ορίσματα (τιμές) τα οποία δέχονται ανάλογα με τον τύπο.

Τύπος	Μέγεθος	Επεξήγηση	Όρισμα που δέχεται
Byte	1 byte	Μικρός ακέραιος	ακέραιο αριθμό στο [0…255]
Integer	2 bytes	Ακέραιος	ακέραιο αριθμό στο [-32768…32767]
Long	4 bytes	Μεγάλος ακέραιος	ακέραιο αριθμό στο [-2147483648…2147483647]
Single	4 bytes	Πραγματικός αριθμός απλής ακρίβειας	Πραγματικό αριθμό στο [– $3.402823*10^{38}$… – $1.401298*10^{-45}$] για αρνητικούς και [$1.401298*10^{-45}$… $3.402823*10^{38}$] για θετικούς.
Double	8 bytes	Πραγματικός αριθμός διπλής ακρίβειας	Πραγματικό αριθμό στο [– $1.79769313486232*10^{308}$ … – $4.94065645841247*10^{-324}$] για αρνητικούς και [$4.94065645841247*10^{-324}$ …$1.79769313486232*10^{308}$] για θετικούς.
String	1 byte / χαρακτήρα	Κείμενο	Οτιδήποτε, αλλά το αναγνωρίζει ως κείμενο
Variant	16 bytes	Διάφοροι	Όλους τους τύπους δεδομένων.
Boolean	1 byte	Λογικός	True ή False

Πώς όμως δηλώνουμε σε κάποια μεταβλητή τον τύπο τον οποίο θα ακολουθήσει;

Η δήλωση κάθε μεταβλητής γίνεται με μια κωδική (δεσμευμένη από την Visual Basic) έκφραση. Ο τρόπος δήλωσης διαφέρει ανάλογα με το αν επιθυμούμε τη χρήση της τιμής κάποιας μεταβλητής «δημόσια» στο πρόγραμμά μας, ή μόνο «τοπικά». Στην πραγματικότητα υπάρχουν περισσότεροι τρόποι δήλωσης μεταβλητών από τις δύο βασικές που θα αναλύσουμε στην συνέχεια.

Τοπική δήλωση: Dim

Σύνταξη:

Dim MyVariable As Τύπος

Παραδείγματα:

```
Dim MyName As Variant
Dim MyCounter As Integer
Dim Ok_Pressed As Boolean
Dim X3 As Double , Y3 As Double , X4 As Double , Y4 As Double
```

Παρατήρηση: Η Dim χρησιμοποιείται μέσα σε οποιαδήποτε κλήση συμβάντος, αλλά ποτέ έξω από αυτές. Η τιμή της μεταβλητής που δηλώνεται κατά αυτόν τον τρόπο κρατάει την τιμή της μόνο μέσα στην υπορουτίνα στην οποία δηλώνεται.

```
Private Sub Command1_Click()
Dim MyName As Variant
MyName = "George"
Print MyName
End Sub
```

Επιχειρώντας να εκτυπώσουμε την τιμή της παραπάνω μεταβλητής (MyName) από μια άλλη υπορουτίνα, διαπιστώνουμε ότι θα εκτυπωθεί ένα κενό, δηλαδή τίποτα. Αυτό συμβαίνει γιατί η μεταβλητή MyName δεν μπορεί να «κρατήσει» την τιμή που της δώσαμε έξω από την υπορουτίνα στην οποία δηλώθηκε.

```
Private Sub Command2_Click()
Print MyName
End Sub
```

Δημόσια δήλωση: Public

Σύνταξη:

Public MyVariable As Τύπος

Παραδείγματα:

Public MyName As Variant
Public MyCounter As Integer
Public Ok_Pressed As Boolean
Public X3 As Double, Y3 As Double, X4 As Double, Y4 As Double

Παρατήρηση: Η Public χρησιμοποιείται μόνο στο τμήμα Declarations του κώδικα (Δηλώσεις). Δεν πρέπει δηλαδή να βρίσκεται μέσα σε κάποια υπορουτίνα. Η τιμή της μεταβλητής που δηλώνεται κατά αυτόν τον τρόπο κρατάει την τιμή της αναλλοίωτη σε όλο το πρόγραμμα, και εξαρτάται μόνο από την φόρμα που τη φέρει. Αν χρησιμοποιηθεί από οποιαδήποτε υπορουτίνα ή συνάρτηση του προγράμματος θα επιστρέψει την πραγματική τιμή που της δόθηκε.

Γιατί όμως μια public μεταβλητή εξαρτάται από τη φόρμα που τη φέρει;

Η μεταβλητή που δηλώνεται ως Public μπορεί πρακτικά να χρησιμοποιηθεί από οποιαδήποτε φόρμα του προγράμματος. Απλά για να

χρησιμοποιηθεί από μια δεύτερη φόρμα δεν αρκεί το όνομά της, αλλά όπως είδαμε και με τις ιδιότητες, θα πρέπει να κληθεί με τον παρακάτω τρόπο (Έστω από τη Form2, ενώ η μεταβλητή ανήκει στην Form1) :

```
Print Form1.MyName
```

Για την ακρίβεια, όταν μια μεταβλητή δηλωθεί ως Public, τότε αυτομάτως δηλώνεται η ύπαρξή της και στην αναδιπλούμενη λίστα ιδιοτήτων της φόρμας (η οποία όπως είδαμε καλείται με το όνομα της φόρμας και έπειτα πατώντας την τελεία).

Ας δούμε στη συνέχεια κάποιες προτάσεις δηλώσεων, συνοδευόμενες πλέον με σύνταξη εκχώρησης τιμής σε κάθε μεταβλητή. Θα θεωρήσουμε ότι οι μεταβλητές δηλώνονται ως τοπικές στο συμβάν κλικ ενός κουμπιού:

```
Private Sub Command1_Click()

Dim MyName As Variant
Dim MyCounter As Integer
Dim Pressed As Boolean
Dim X3 As Double, Y3 As Double, X4 As Double, Y4 As Double

MyName = "George"
MyCounter = 12
Pressed = True
X3 = 15.3456
Y3 = -24.9912
X4 = (X3 + Y3) / 2
Y4 = X3 - X4

End Sub
```

Στο παραπάνω τμήμα κώδικα βλέπουμε ότι και η λέξη «True» (όπως και η «False») είναι επίσης δεσμευμένες από τη VB.

Στην μεταβλητή MyName χρησιμοποιήσαμε εισαγωγικά για να εισάγουμε το κείμενο "George". Αν δεν χρησιμοποιούσαμε εισαγωγικά, τότε η VB θα καταλάβαινε (εσφαλμένα) ότι η λέξη George δεν είναι τίποτα άλλο από μια μεταβλητή.

Η MyCounter, επειδή είναι δηλωμένη ως Integer αν έπαιρνε την τιμή «George» προφανώς θα προκαλούσε διακοπή της εκτέλεσης του προγράμματος λόγω λάθους (με ένα ενοχλητικό μήνυμα σφάλματος!). Γιατί; Διότι περιμένει ακέραια τιμή ως όρισμα, και μάλιστα από το πεδίο τιμών που προαναφέραμε. Αν ωστόσο της δίναμε την τιμή 12.87 δεν θα επέστρεφε

λάθος. Απλά θα στρογγυλοποιούσε τον αριθμό στον πλησιέστερο ακέραιο (στο 13).

Η μεταβλητή Pressed, λόγω του ότι γνωρίζει επακριβώς τις τιμές που περιμένει (True ή False), μας καθοδηγεί έτσι ώστε να μην κάνουμε λάθος. Γράφοντας δηλαδή Pressed, πατώντας το « = » μας εμφανίζει δίπλα μια λίστα με τις επιλογές True και False μόνο, για να επιλέξουμε.

Λάθη θα προέκυπταν και στις μεταβλητές που είναι δηλωμένες ως πραγματικές (απλής ή διπλής ακρίβειας) δηλαδή είτε Single είτε Double, εάν επιχειρούσαμε να καταχωρίσουμε σε αυτές κάτι πέρα από μια αριθμητική τιμή (που ανήκει στα προαναφερθέντα διαστήματα).

Στις μεταβλητές X4 και Y4 δεν δίνουμε απευθείας τιμές. Απλά συντάσσουμε μια μαθηματική παράσταση η οποία περιέχει μεταβλητές και αριθμούς σε οποιονδήποτε συνδυασμό, και την «αφήνουμε» να μας υπολογίσει το αποτέλεσμα.

Πιθανές απορίες:

1. Πως επιλέγω κατάλληλο όνομα για μια μεταβλητή;
2. Είναι υποχρεωτική η δήλωση των μεταβλητών;

(1). Οι μόνες αληθινές δεσμεύσεις που υπάρχουν για την επιλογή του ονόματος είναι:

α) Να μην ξεκινάει το όνομα από σύμβολο ή αριθμό.

β) Να μην είναι δεσμευμένη λέξη από την Visual Basic.

Ωστόσο καλό είναι να υιοθετήσουμε κάποιο σκεπτικό επιλογής ονόματος, ώστε όταν συναντάμε την μεταβλητή να καταλαβαίνουμε και τον τύπο της. Για παράδειγμα μπορούμε να χρησιμοποιούμε πεζούς χαρακτήρες πριν το όνομα:

```
Dim vMyName As Variant
Dim iMyCounter As Integer
Dim boPressed As Boolean
Dim dX3 As Double, dY3 As Double, dX4 As Double, dY4 As Double
Dim siNumber As Single
Dim sNameOfFile As String.
Dim lCounter As Long
```

(2). Για να καταστήσουμε κάποια μεταβλητή «δημόσια» πρέπει οπωσδήποτε να τη δηλώσουμε. Για τις τοπικές μπορούμε να αποφύγουμε την δήλωση. Τότε όμως θεωρούνται ως Variant (γενικού τύπου), καταλαμβάνουν αρκετή μνήμη, και δεν θυμόμαστε εύκολα πού τις χρησιμοποιούμε. Αν όμως έχουμε εισάγει

(ή έχει εισαχθεί αυτόματα) την πρόταση Option Explicit στο τμήμα Declarations του κώδικα, τότε η απουσία δήλωσης οδηγεί σε σφάλμα.

Γενικά θα πρέπει να θυμόμαστε τους εξής κανόνες κατά τη χρήση μεταβλητών:

Α) Η μεταβλητή η οποία βρίσκεται αριστερά της «ισότητας» στην ουσία είναι αυτή στην οποία εκχωρείται το αποτέλεσμα (ή οτιδήποτε δώσουμε) το οποίο βρίσκεται στα δεξιά του « = ».

X1 = X2 * 1.00023 + Y2 * X3 / X4 + 55.12

Στην παραπάνω παράσταση το αποτέλεσμα της πράξης εκχωρείτε στην μεταβλητή X1.

Β) Δεν ισχύει στις μεταβλητές η αλγεβρική ισότητα (κατά την καταχώριση τιμής), δηλαδή αν και η παρακάτω παράσταση αλγεβρικά είναι εσφαλμένη,

X3 = X3 + 50.412

στην Visual Basic σημαίνει: Αύξησε την τιμή της μεταβλητής X3 κατά 50.412 (το ίδιο ισχύει για οποιαδήποτε έκφραση όσο πολύπλοκη και αν είναι). Δηλαδή όπου το όνομα της μεταβλητής βρίσκεται και αριστερά και δεξιά της ισότητας, σημαίνει ότι η τιμή που θα καταχωρισθεί τελικώς στην μεταβλητή είναι συνάρτηση της προηγούμενης τιμής που κρατούσε.

X1 = 1647.2791
X1 = X2 * 1.00023 + (Y2 * X1) / (X4 + 55.12 + X1)

2) Πίνακες Μεταβλητών

Σε κάποιες διαδικασίες, χρειαζόμαστε ένα πλήθος μεταβλητών οι οποίες θα δέχονται ως όρισμα ιδίου τύπου δεδομένα. Σε κάποιες άλλες περιπτώσεις, μας είναι άγνωστο το πλήθος των μεταβλητών που θα απαιτηθούν. Υποθέστε ότι έχουμε κάποιο αρχείο το οποίο περιέχει συντεταγμένες, και θέλουμε να το διαβάσουμε και να καταχωρίσουμε κάθε τιμή από αυτό το αρχείο σε μια ξεχωριστή μεταβλητή.

Όταν ο χρήστης «τρέχει» το πρόγραμμα, δεν γνωρίζει πόσο μεγάλο είναι το αρχείο που ανοίγει (πόσες γραμμές / records περιέχει), αλλά τον ενδιαφέρει να διαβάσει όλα τα δεδομένα. Προφανώς δεν θα μπορούσαμε να γράψουμε δηλώσεις για κάθε μεταβλητή (μπορεί να χρειαζόταν εκατομμύρια τέτοιες δηλώσεις!!!). Σε αυτές τις περιπτώσεις η λύση δίνεται με τη χρήση πινάκων μεταβλητών.

Οι τύποι των πινάκων μεταβλητών, καθώς και ο τρόπος δήλωσής τους, είναι ο ίδιος με τις απλές μεταβλητές. Αυτό που αλλάζει είναι η σύνταξη και η χρήση της ίδιας της μεταβλητής – πίνακα.

Μοναδική εξαίρεση αποτελεί το γεγονός ότι η δήλωση πίνακα ως Public είναι εφικτή μόνο μέσα σε ένα Module. Επειδή όμως προς το παρόν δεν θα ασχοληθούμε με τη δημόσια δήλωση πίνακα, το ζήτημα αυτό θα το αντιμετωπιστεί σε επόμενο κεφάλαιο.

Παράδειγμα δήλωσης πίνακα με 65 στοιχεία:

```
Dim MyAr(65) As Single
```

Αυτό που κάνουμε εδώ είναι να προσδιορίσουμε μέσα σε παρένθεση το πλήθος των στοιχείων που επιθυμούμε. Ο MyAr είναι ένας μονοδιάστατος πίνακας στοιχείων. Μπορούμε να έχουμε n-διάστατους πίνακες, π.χ.

```
Dim MyAr(65, 5, 4) As Single
```

Στον προγραμματισμό απλών προβλημάτων θα δυσκολευόμασταν να αντιληφθούμε τι σημαίνει ένας πίνακας με γραμμές, στήλες και βάθος. Στην Visual Basic όμως φανταστείτε ότι μπορείτε να έχετε πολλές «διαστάσεις» ακόμα!

```
Dim MyAr(65, 5, 4, 2, 12) As Single
```

Πώς όμως χρησιμοποιούμε αυτά τα στοιχεία; Πώς γίνεται η σύνταξη αυτού του είδους των μεταβλητών; Όπως και στις απλές μεταβλητές, με μόνη

διαφορά ότι θα πρέπει να «δείχνουμε» τη μεταβλητή στην οποία αναφερόμαστε.

Παράδειγμα:
```
Dim dX(50, 3) As Double

dX(1, 2) = 14.455
dX(10, 1) = 15.12
dX(50, 3) = 4.488
```

Δεν μπορούμε να ξεπεράσουμε τα όρια καμίας διάστασης από αυτές που έχουμε δηλώσει στην πρόταση Dim. Οι «δείκτες» μπορεί να είναι μόνο ακέραιοι αριθμοί.

Τι γίνεται όμως αν δεν γνωρίζουμε το μέγεθος του πίνακα που θα απαιτηθεί από το πρόγραμμά μας; Σε αυτή την περίπτωση ο πίνακας δηλώνεται «σκέτος», δηλαδή χωρίς ορίσματα:

```
Dim MyArray( ) As Single
```

Αργότερα ο χρήστης μπορεί να επαναδιαστασιολογήσει (ReDim) τον πίνακα MyArray προσδιορίζοντας επακριβώς τις διαστάσεις του πίνακα με την βοήθεια μεταβλητών.

```
ReDim MyArray(iNumber1, iNumber2) [As Single]
```

Χρησιμοποιούμε αγκύλες ([ ]) στον παραπάνω κώδικα για να επισημάνουμε το γεγονός ότι η δήλωση εντός των αγκυλών ΔΕΝ είναι πλέον υποχρεωτική. Αν όμως χρησιμοποιηθεί, δεν επιτρέπεται να επιφέρει την αλλαγή του τύπου του πίνακα διότι αυτή η ενέργεια θα οδηγούσε σε σφάλμα.

Παρατήρηση: Χρησιμοποιούμε κάποιες μεταβλητές τις οποίες ονομάζουμε iNumber1 και iNumber2. Αυτό γίνεται για να καταδειχθεί ότι ως διάσταση ενός πίνακα δεν μπορεί να χρησιμοποιηθεί τίποτα πέρα από έναν ακέραιο αριθμό (Byte, Integer, Long).

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Στη μεταβλητή η οποία βρίσκεται αριστερά του συμβόλου « = » εκχωρείται η τιμή (ή το αποτέλεσμα της παράστασης) που αναγράφεται δεξιά αυτού.
2. Κατά την εκχώρηση τιμής σε μια μεταβλητή δεν ισχύει η αλγεβρική ισότητα (met1 = met1 + 1).
3. Τα ονόματα των μεταβλητών πρέπει να είναι γραμμένα πάντα με λατινικούς χαρακτήρες, να μην ξεκινάνε με σύμβολο ή αριθμό, και να μην είναι αυτούσιες δεσμευμένες λέξεις της Visual Basic.
4. Πρέπει να δηλώνουμε πάντα τις μεταβλητές που θα χρησιμοποιήσουμε, καθώς και να επιλέγουμε συνετά τον τύπο κάθε μεταβλητής που δηλώνουμε.
5. Όταν δηλώνουμε μια μεταβλητή με την έκφραση Dim εσωτερικά σε μια υπορουτίνα, η τιμή της μεταβλητής δεν είναι διαθέσιμη έξω από αυτή. Για να είναι διαθέσιμη η μεταβλητή σε όλο το project πρέπει να την δηλώσουμε ως Public στο τμήμα Declarations του παραθύρου σύνταξης του κώδικα.
6. Οι πίνακες πρέπει να δηλώνονται, είτε με τον ακριβή αριθμό των στοιχείων τους, είτε ως κενοί και έπειτα να επαναπροσδιορίζουμε τις «διαστάσεις» τους με την εντολή ReDim.

ΕΡΩΤΗΣΕΙΣ

1. Μπορώ να δώσω την τιμή 13.176 σε μια μεταβλητή που έχω δηλώσει ως String; Αν ναι, τι αποτέλεσμα έχει αυτή η αντικατάσταση;
2. Σε μια μεταβλητή που είναι δηλωμένη ως Variant μπορώ να δώσω την τιμή «Γιώργος»;
3. Μπορώ να γράψω μια μαθηματική παράσταση χρησιμοποιώντας ως ορίσματα ακέραιους και πραγματικούς; Αν ναι, σε τι τύπο μεταβλητής μπορώ να εκχωρήσω το αποτέλεσμα;
4. Σε ποιες περιπτώσεις πιστεύετε ότι θα μπορούσα να χρησιμοποιήσω μεταβλητή τύπου Boolean;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα project και σχεδιάστε 2 κουμπιά και 2 ετικέτες. Δηλώστε 2 μεταβλητές οι οποίες, όταν κάνουμε κλικ στο πρώτο κουμπί, θα παίρνουν ως όρισμα το όνομα η πρώτη και το επώνυμό σας η δεύτερη αντίστοιχα. Όταν κάνουμε κλικ στο δεύτερο κουμπί θα εκτυπώνουν τις τιμές τους αντίστοιχα στην πρώτη και δεύτερη Label.

Κεφάλαιο 5
Εσωτερικές συναρτήσεις

Η βιβλιοθήκη της Visual Basic περιλαμβάνει ένα μεγάλο φάσμα συναρτήσεων για την διευκόλυνση του προγραμματιστή. Το γεγονός αυτό πλέον το θεωρούμε αυτονόητο αφού και τα κομπιουτεράκια τσέπης παρέχουν έναν ικανό αριθμό συναρτήσεων.

Οι εσωτερικές συναρτήσεις (intrinsic functions) που μας παρέχονται στη VB δεν αφορούν μόνο σε κλασικούς μαθηματικούς υπολογισμούς, αλλά μας προσφέρουν μεγάλη ευχέρεια στον χειρισμό και την αποτελεσματική διαχείριση δεδομένων παντός τύπου. Χωρίζονται γενικά στις παρακάτω κατηγορίες:

α) Μαθηματικές συναρτήσεις:
Σε αυτή την κατηγορία ανήκουν συναρτήσεις όπως η Sin (ημίτονο), Cos (συνημίτονο), Abs (απόλυτη τιμή) κτλ. Ας τις δούμε αναλυτικότερα:

Συνάρτηση	Επεξήγηση	Όρισμα	Αποτέλεσμα
Sin(a)	Ημίτονο του α	Το α σε ακτίνια	Πραγματικός
Cos(a)	Συνημίτονο του α	Το α σε ακτίνια	Πραγματικός
Tan(a)	Εφαπτομένη του α	Το α σε ακτίνια	Πραγματικός
Atn(a)	Τόξο εφαπτομένης του α	Πραγματικός αριθμός	Πραγματικός
Log(a)	Λογάριθμος του α	Πραγματικός αριθμός >0	Πραγματικός
Sqr(a)	Τετραγωνική ρίζα του α	Πραγματικός αριθμός >=0	Πραγματικός
Abs(a)	Απόλυτη τιμή του α	Πραγματικός αριθμός	Πραγματικός
Int(a)	Ακέραιος του α	Πραγματικός αριθμός	Ακέραιος
Rnd * a	Τυχαίος αριθμός από 0 έως α	Πραγματικός αριθμός	Πραγματικός
Sgn(a)	Επιστρέφει το πρόσημο του α.	Πραγματικός αριθμός	Αν α>0 επιστρέφει 1 Αν α=0 επιστρέφει 0 Αν α<0 επιστρέφει -1
Exp(a)	Επιστρέφει την τιμή e^a	Πραγματικός αριθμός	Πραγματικός

Στις παραπάνω συναρτήσεις δεν περιλαμβάνονται οι αντίστοιχες για τα τόξα ημιτόνου και συνημιτόνου, νεπέριου λογαρίθμου κτλ. Αυτό συμβαίνει γιατί η Microsoft επέλεξε (στο περιβάλλον της VB) να συστήσει τον υπολογισμό αυτών των συναρτήσεων με έμμεσο τρόπο, ως «Παράγωγες Μαθηματικές Συναρτήσεις» :

Συνάρτηση	Επεξήγηση	Όρισμα
Atn(a / Sqr(-a * a + 1))	Τόξο ημιτόνου (Arcsin)	a = Sin(b)
Atn(-a/Sqr(-a * a + 1)) + 2 * Atn(1)	Τόξο συνημιτόνου (Arccos)	a = Cos(b)
Log(a) / Log(N)	Λογάριθμος με βάση N	
a mod b a mod b = a – int(a/b) * b	Modulo (υπόλοιπο διαίρεσης)	a,b ακέραιοι

Σχόλιο:

Άλλες «γλώσσες» και περιβάλλοντα προγραμματισμού διαθέτουν τις «ειδικές» συναρτήσεις arcsin, arccos, arctan2 (τόξο εφαπτωμένης με δύο ορίσματα) ως intrinsic functions στις βασικές τους βιβλιοθήκες. Αυτό είναι πολύ «βολικό» στον Τοπογράφο ο οποίος απαιτείται να αντιμετωπίσει, π.χ., το 2ο θεμελιώδες πρόβλημα στον κώδικά του (να υπολογίσει δηλαδή την γωνία διεύθυνσης στο «σωστό» τεταρτημόριο από τις διαφορές των συντεταγμένων δύο σημείων: $\tan G_{12}$ = DX/DY , G_{12} = arctan [DX/DY]). Επίσης, η απουσία της συνάρτησης Arccos προκαλεί αδυναμία στον υπολογισμό της γωνίας από τον «νόμο του συνημιτόνου» και ο προγραμματιστής αναγκάζεται να παρακάμψει το πρόβλημα, με έμμεσο υπολογισμό της Arccos (μέσω της Atn του παραπάνω πίνακα).

Προκειμένου ο (τοπογράφος) προγραμματιστής να αξιολογήσει τις δυνατότητες και τους περιορισμούς άλλων «γλωσσών», αξίζει να αναφέρουμε ότι

(α) στη Fortran και στην Pascal υπάρχουν οι συναρτήσεις TAN, ATAN, ATAN2, SIN, ASIN, COS, ACOS και όλες οι αντίστοιχες σε διπλή (και σε μερικές εκδόσεις τετραπλή) ακρίβεια: DTAN, DATAN, DATAN2, κτλ

(β) στο Excel του MS Office υπάρχουν οι συναρτήσεις TAN, ATAN, ATAN2, SIN, ASIN, COS, ACOS διπλής ακρίβειας. Άρα, ο (τοπογράφος) προγραμματιστής διαθέτει στο Excel και στις γλώσσες Fortran και Pascal άμεση δυνατότητα στον υπολογισμό θεμελιωδών προβλημάτων, και πάντως καλύτερη από αυτήν της VB.

Στην κατάσταση «Βοήθειας» (Help - F1) της Visual Basic μπορείτε να δείτε όλες τις διαθέσιμες παράγωγες συναρτήσεις (Derived Math Functions) όπως παρουσιάζονται αυτούσιες παρακάτω:

Function	Derived equivalents
Secant	Sec(X) = 1 / Cos(X)
Cosecant	Cosec(X) = 1 / Sin(X)
Cotangent	Cotan(X) = 1 / Tan(X)
Inverse Sine	Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Inverse Cosine	Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Inverse Secant	Arcsec(X) = Atn(X / Sqr(X * X – 1)) + Sgn((X) – 1) * (2 * Atn(1))
Inverse Cosecant	Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) – 1) * (2 * Atn(1))
Inverse Cotangent	Arccotan(X) = Atn(X) + 2 * Atn(1)
Hyperbolic Sine	HSin(X) = (Exp(X) – Exp(-X)) / 2
Hyperbolic Cosine	HCos(X) = (Exp(X) + Exp(-X)) / 2
Hyperbolic Tangent	HTan(X) = (Exp(X) – Exp(-X)) / (Exp(X) + Exp(-X))
Hyperbolic Secant	HSec(X) = 2 / (Exp(X) + Exp(-X))
Hyperbolic Cosecant	HCosec(X) = 2 / (Exp(X) – Exp(-X))
Hyperbolic Cotangent	HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) – Exp(-X))
Inverse Hyperbolic Sine	HArcsin(X) = Log(X + Sqr(X * X + 1))
Inverse Hyperbolic Cosine	HArccos(X) = Log(X + Sqr(X * X – 1))
Inverse Hyperbolic Tangent	HArctan(X) = Log((1 + X) / (1 – X)) / 2
Inverse Hyperbolic Secant	HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Inverse Hyperbolic Cosecant	HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) + 1) / X)
Inverse Hyperbolic Cotangent	HArccotan(X) = Log((X + 1) / (X – 1)) / 2
Logarithm to base N	LogN(X) = Log(X) / Log(N)

Προσοχή: Τα παραπάνω ονόματα συναρτήσεων (όπως π.χ. το Arcsin) δεν υπάρχουν. Είναι απλά ονόματα που προτείνονται για τη δημιουργία της εκάστοτε συνάρτησης.

Παραδείγματα χρήσης μαθηματικών συναρτήσεων:

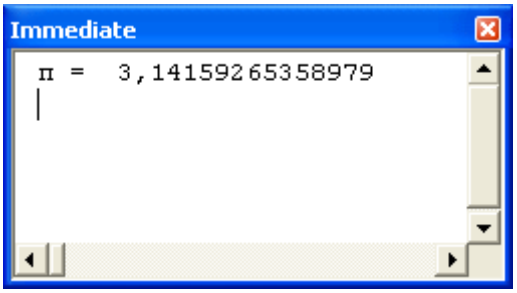
```
Private Sub Command1_Click()
Dim dCos As Double, dSin As Double, dAtn As Double
Dim siRand As Single, dMyNum As Double

dMyNum = 1.3466
dCos = Cos(dMyNum)
dSin = Sin(dMyNum)
dAtn = Atn(dMyNum)

Randomize Timer
siRand = Rnd * dMyNum

Debug.Print dMyNum
Debug.Print dCos, dSin, dAtn
Debug.Print siRand
End Sub
```

Στο παραπάνω παράδειγμα προσθέτουμε κώδικα στο συμβάν κλικ του κουμπιού με όνομα «Command1».
Δηλώσαμε κάποιες μεταβλητές τοπικά.
Θέσαμε τιμή στην μεταβλητή dMyNum.
Δώσαμε στις μεταβλητές dCos, dSin, dAtn το αποτέλεσμα της συνάρτησης Cos, Sin και Atn (όπως φαίνεται παραπάνω), με όρισμα συνάρτησης την τιμή της μεταβλητής dMyNum.
Randomize Timer: Πρόκειται για μια έκφραση της VB η οποία χρησιμοποιείται «όπως είναι», προαιρετικά, πριν από τη χρήση της Rnd. Αυτό που κάνει είναι να βοηθήσει την Rnd να επιστρέψει όσο το δυνατόν πιο τυχαία νούμερα.
Στην συνέχεια αφήνουμε να επιλεγεί μια τυχαία τιμή στο [0...dMyNum), και να καταχωριστεί η τιμή στη μεταβλητή siRand.
Debug.Print: Η εντολή αυτή εκτυπώνει αποτελέσματα ή ό,τι άλλο επιθυμούμε στο Immediate Window (Παράθυρο άμεσης εκτύπωσης). Πρόκειται για εξαιρετικό βοήθημα στην αποσφαλμάτωση των προγραμμάτων μας.

Το παρακάτω παράδειγμα υπολογίζει την τιμή του « π » μέσω της συνάρτησης Atn.

```
Private Sub Command1_Click()
Dim Pi As Double

Pi = 4 * Atn(1)
Debug.Print "π = "; Pi

End Sub
```

Το αποτέλεσμα που βλέπουμε στο Immediate Window είναι το παρακάτω:



Η χρήση των μαθηματικών συναρτήσεων είναι πολύ απλή υπόθεση όπως είδατε. Μπορούν να χρησιμοποιηθούν σε οποιονδήποτε συνδυασμό, αρκεί τα ορίσματα να είναι σωστά.

β) Συναρτήσεις κειμένου:

Εδώ θα συναντήσουμε συναρτήσεις οι οποίες χρησιμοποιούνται για τη διαμόρφωση του περιεχομένου μιας μεταβλητής, η οποία πλέον δεν «κρατάει» αριθμητική τιμή, αλλά κείμενο κτλ. Τέτοιου τύπου είναι οι μεταβλητές που δηλώνονται ως String ή ως Variant.

Συνάρτηση	Επεξήγηση	Όρισμα	Αποτέλεσμα
Len(aString)	Επιστρέφει το μήκος της μεταβλητής aString σε χαρακτήρες.	Οποιαδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο).	Ακέραιος
Mid(aString, iStartP, iEndP)	Επιστρέφει τμήμα της μεταβλητής aString από τη θέση iStartP εως τη θέσης iEndP.	Οποιοδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο), αρχική θέση και τελική θέση ως ακέραιοι.	Variant ή String

Left(aString, iLength)	Επιστρέφει τμήμα της μεταβλητής aString από την αρχή της έως το μήκος (σε χαρακτήρες) που θα ορίσουμε.	Οποιοδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο), διάστημα (μήκος)	Variant ή String
Right(aString, iLength)	Επιστρέφει τμήμα της μεταβλητής aString από το τέλος της έως το μήκος (σε χαρακτήρες) που θα ορίσουμε προς την αρχή.	Οποιαδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο), διάστημα (μήκος)	Variant ή String
Trim(aString)	Διαγράφει τα κενά διαστήματα που προηγούνται ή έπονται του κειμένου.	Οποιαδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο).	
Ltrim(aString)	Διαγράφει τα κενά διαστήματα που προηγούνται του κειμένου.	Οποιαδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο).	
Rtrim(aString)	Διαγράφει τα κενά διαστήματα που έπονται του κειμένου.	Οποιαδήποτε μεταβλητή που της έχει εκχωρηθεί αλφαριθμητική τιμή (κείμενο).	

Παράδειγμα χρήσης συναρτήσεων κειμένου:

```
Private Sub Command1_Click()

Dim vMyName As Variant
Dim iStPos As Integer, iEndPos As Integer
Dim iLength As Integer, vNameAfter As Variant

vMyName = "    Fanis  "
iLength = Len(vMyName)
Debug.Print "Αρχικό όνομα = "; vMyName
Debug.Print "Αρχικό μήκος = "; iLength

vMyName = Trim(vMyName)
iLength = Len(vMyName)
Debug.Print "Τελικό όνομα = "; vMyName
Debug.Print "Τελικό μήκος = "; iLength

iStPos = 1
iEndPos = 3
vNameAfter = Mid(vMyName, iStPos, iEndPos)
Debug.Print "Αποκομένο όνομα = "; vNameAfter

End Sub
```

Σχολιασμός του παραπάνω κώδικα:
Δήλωση κάποιων μεταβλητών. Απόδοση στην vMyName μιας τιμής.
Υπολογισμός του μήκους της vMyName και αποθήκευση στην iLength.
Εκτύπωση των παραπάνω μεταβλητών.
Αποκοπή των κενών διαστημάτων πριν και μετά από το αληθινό κείμενο της μεταβλητής.
Εκ νέου υπολογισμός του μήκους της **vMyName** και εκτύπωση.
Απόδοση στις 2 μεταβλητές ακεραίων τιμών, ώς τιμών εκκίνησης και λήξης (1 και 3 αντίστοιχα).
Καταχώριση στην vNameAfter των 3 πρώτων χαρακτήρων της **vMyName** (από τη θέση 1 έως και την 3). Εκτύπωση.

Αυτή ήταν μια σύντομη περιγραφή των συναρτήσεων επεξεργασίας μεταβλητών που τους έχει καταχωριθεί κείμενο.

γ) Συναρτήσεις Ημερομηνίας και Ώρας:

Οι συναρτήσεις αυτές είναι κάπως εξειδικευμένες, αλλά θα τις δούμε κάπως συνοπτικά για να πάρουμε μια ιδέα για τη χρήση τους.

Συνάρτηση	Επεξήγηση	Όρισμα	Αποτέλεσμα
Now	«Επιστρέφει» την τρέχουσα ημερομηνία και ώρα.	Κανένα	
Date	«Επιστρέφει» την τρέχουσα ημερομηνία.	Κανένα	
Time	«Επιστρέφει» την τρέχουσα ώρα.	Κανένα	
Year(aDate)	«Επιστρέφει» το έτος της ημερομηνίας που δίνουμε.	Ημερομηνία	Ακέραιος
Month(aDate)	«Επιστρέφει» το μήνα της ημερομηνίας που δίνουμε.	Ημερομηνία	Ακέραιος
Day(aDate)	«Επιστρέφει» την ημέρα της ημερομηνίας που δίνουμε.	Ημερομηνία	Ακέραιος
Hour(aTime)	«Επιστρέφει» τις ώρες της «Ώρας» που δίνουμε.	Ώρα	Ακέραιος
Minute(aTime)	«Επιστρέφει» τα λεπτά της «Ώρας» που δίνουμε.	Ώρα	Ακέραιος
Second(aTime)	«Επιστρέφει» τα δευτερόλεπτα της «Ώρας» που δίνουμε.	Ώρα	Ακέραιος

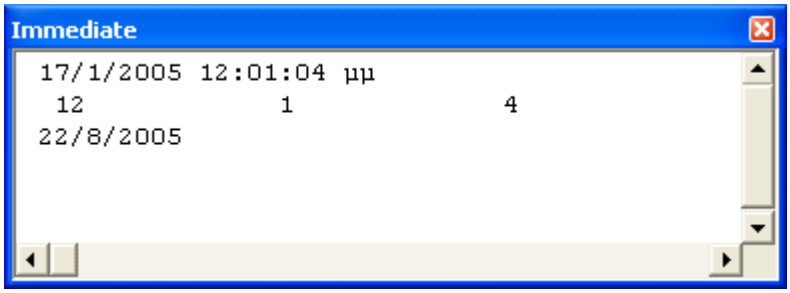
Να σημειώσουμε εδώ ότι μπορούμε να δηλώσουμε μια μεταβλητή με τύπο ημερομηνίας ως εξής:

```
Dim MyDate As Date
```

το οποίο ωστόσο δεν κρίνεται απαραίτητο, αφού η δήλωση ως Variant υπερκαλύπτει την παραπάνω δήλωση.

Παράδειγμα χρήσης συναρτήσεων ημερομηνίας και ώρας:

```
Private Sub Command1_Click()
Dim vMyDate As Variant, AnotherDate As Date

vMyDate = Now
Debug.Print vMyDate

Debug.Print Hour(vMyDate), Minute(vMyDate), Second(vMyDate)

AnotherDate = #8/22/2005#
Debug.Print AnotherDate

End Sub
```



Όπως είπαμε δεν θα ασχοληθούμε ιδιαίτερα με αυτές τις συναρτήσεις, παρόλα αυτά έχετε πάρει πλέον μια ιδέα για τη χρήση τους.

δ) Συναρτήσεις εισόδου, εξόδου και παρουσίασης:

Οι συναρτήσεις αυτές επιτρέπουν την εύκολη εισαγωγή και παρουσίαση των δεδομένων και των αποτελεσμάτων.

Συνάρτηση MsgBox (πλαίσιο εκτύπωσης ή πλαίσιο εξόδου)

Σύνταξη :

```
msgbox
MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
```

Prompt : Το σύνολο των τιμών (αριθμητικών ή / και αλφαριθμητικών που θέλουμε να εκτυπώσουμε.

Buttons As VbMsgBoxStyle = vbOKOnly : Διαλέγουμε τα πλήκτρα τα οποία θα εμφανιστούν πάνω στο MsgBox καθώς και το οπτικό στυλ.

Title : Ο τίτλος που θα αναγράφεται πάνω στην μπλε μπάρα του MsgBox.

HelpFile : Στην περίπτωση που χρησιμοποιήσουμε κάποιο εξωτερικό αρχείο θα δηλώσουμε εδώ το όνομά του.

Context : Τα αρχεία βοηθείας της Visual Basic παίρνουν ένα όρισμα – αναφορά (για να γνωρίζουν ποιο τμήμα θα καλέσουν). Εδώ γράφεται το συγκεκριμένο όρισμα.

Παράδειγμα :

```
myname = "Λουδίας"
msgbox ("Το όνομά μου είναι " & myname ,
```

MsgBox(*Prompt*, [***Buttons As*** **VbMsgBoxStyle = vbOKOnly**], [*Title*], [*HelpFile*], [*Context*]) As VbMsgBoxResult

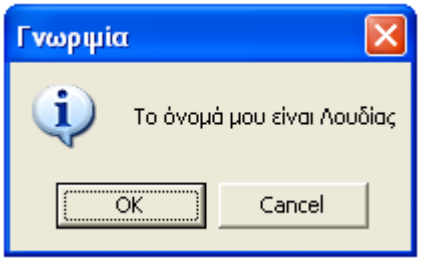
- vbApplicationModal
- vbCritical
- vbDefaultButton1
- vbDefaultButton2
- vbDefaultButton3
- vbDefaultButton4

επιλέγω ως στυλ και πλήκτρα τα vbInformation + vbOKCancel …

```
myname = "Λουδίας"
MsgBox "Το όνομά μου είναι " & myname, vbInformation + vbOKCancel, "Γνωριμία"
```

και ως τίτλο επέλεξα να γράφει «Γνωριμία»

Εκτελώντας το πρόγραμμα θα λάβω το εξής αποτέλεσμα:



Μπορείτε να παρατηρήσετε την αντιστοιχία μεταξύ κώδικα και αποτελέσματος. Παρέλειψα να χρησιμοποιήσω το HelpFile και κατά συνέπεια το Context διότι δεν έχω δημιουργήσει κάποιο αρχείο βοηθείας που να περιλαμβάνει οδηγίες για το συγκεκριμένο πρόγραμμα.

Στα παραπάνω ορίσματα της συνάρτησης μόνο το Prompt είναι υποχρεωτικό όρισμα. Τα υπόλοιπα είναι προαιρετικά και χρησιμοποιούνται για την καλύτερη εμφάνιση και λειτουργία της φόρμας μηνύματος (Active Interface).

Συνάρτηση InputBox (πλαίσιο εισόδου)

Σύνταξη :

```
Answer = InputBox (
         InputBox(Prompt, [Title], [Default], [XPos], [YPos], [HelpFile], [Context]) As String
```

Όπως παρατηρούμε ΒΑΣΙΚΟ στοιχείο σύνταξης της InputBox είναι η ύπαρξη μιας μεταβλητής η οποία θα πάρει την τιμή που θα πληκτρολογήσουμε στο InputBox. Ονομάσαμε αυτή την μεταβλητή Answer.

Prompt : Το μήνυμα το οποίο θα εκτυπωθεί πάνω στη φόρμα του InputBox και το οποίο θα περιγράφει το δεδομένο που ζητάμε.

Title : Ο τίτλος που θα αναγράφεται πάνω στην μπλε μπάρα του InputBox.
Default : Μια προκαθορισμένη τιμή που προτείνουμε εμείς (τρέχουσα τιμή).

XPos : Η οριζόντια συντεταγμένη της οθόνης όπου θα εμφανιστεί το InputBox.

YPos : Η κάθετη συντεταγμένη της οθόνης όπου θα εμφανιστεί το InputBox.

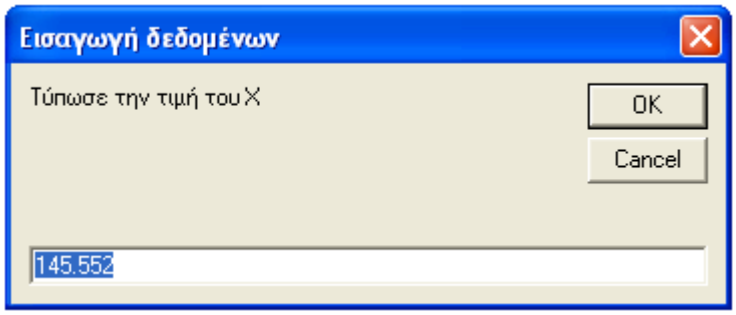
HelpFile : Στην περίπτωση που χρησιμοποιήσουμε κάποιο εξωτερικό αρχείο θα δηλώσουμε εδώ το όνομά του.

Context : Τα αρχεία βοηθείας της Visual Basic παίρνουν ένα όρισμα – αναφορά (για να γνωρίζουν ποιο τμήμα θα καλέσουν). Εδώ γράφεται το συγκεκριμένο όρισμα

```
Answer = InputBox("Τύπωσε την τιμή του Χ", "Εισαγωγή δεδομένων", "145.552")
```

Επιλέξαμε ως μήνυμα το «Τύπωσε την τιμή του Χ», ως τίτλο «Εισαγωγή δεδομένων», ως προκαθορισμένη τιμή το «145.522».

Παραλείψαμε τα XPos, YPos και όπως παραπάνω τα HelpFile και Context. Λόγω έλλειψης των XPos, YPos το πλαίσιο εισόδου θα εμφανιστεί κεντραρισμένο στην οθόνη.



Η τιμή που θα πληκτρολογήσουμε θα καταχωριστεί στην μεταβλητή Answer.

Απαραίτητα ορίσματα της συνάρτησης InputBox είναι μόνο το Prompt.

Συνάρτηση Format

Σύνταξη :

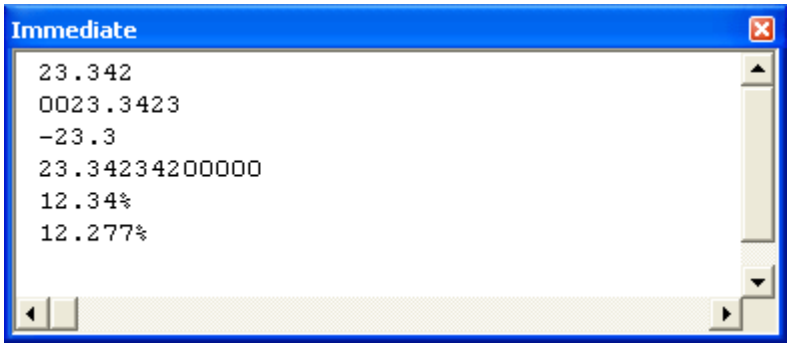
```
format (
Format(Expression, [Format], [FirstDayOfWeek As VbDayOfWeek = vbSunday], [FirstWeekOfYear As VbFirstWeekOfYear = vbFirstJan1])
```

ή απλούστερα:
Format(Μεταβλητή ή παράσταση, "τύπος")
Π.χ. Format(MyNumber, "0,000")

Η Format παρουσιάζει με συγκεκριμένο τρόπο μια μεταβλητή, είτε έχει αριθμητική τιμή, είτε κείμενο είτε ημερομηνία. Στην ανάλυσή μας θα χρησιμοποιήσουμε την Format για παρουσίαση αριθμητικών δεδομένων, μέσα από κάποια παραδείγματα.

```
Private Sub Command1_Click()
Dim dNum1 As Double, dNum2 As Double
dNum1 = 23.342342
dNum2 = 0.12342342

Debug.Print Format(dNum1, "0.000")
Debug.Print Format(dNum1, "0000.0000")
Debug.Print Format(dNum1, "-0.0")
Debug.Print Format(dNum1, "0.00000000000")

Debug.Print Format(dNum2, "0.00%")
Debug.Print Format(dNum2 * dNum1 / (dNum2 + dNum1), ".000%")

End Sub
```

Τα αποτελέσματα που θα πάρουμε είναι τα εξής:

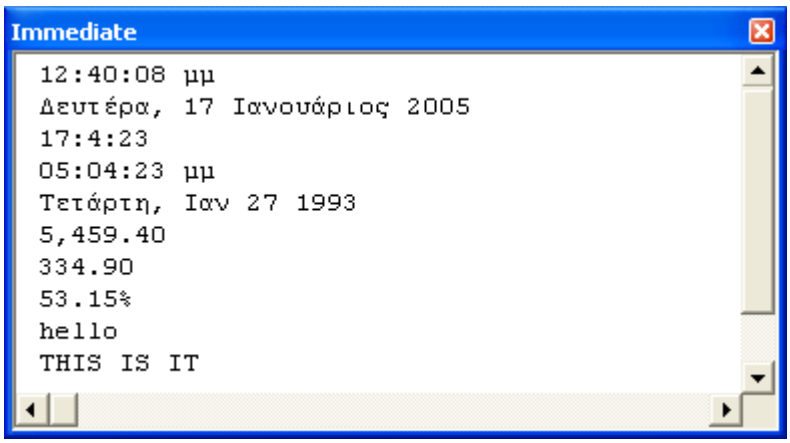


Παρατηρήστε τη συσχέτιση του τύπου του Format και των εκτυπωμένων αποτελεσμάτων.

Παρακάτω παρουσιάζεται ένα ακόμα παράδειγμα από τη βιβλιοθήκη βοηθείας της Visual Basic:

```
Private Sub Form_Load()
Dim MyTime As Variant, MyDate As Variant, MyStr As Variant
MyTime = #5:04:23 PM#
MyDate = #1/27/1993#

' Επιστρέφει την τρέχουσα ώρα σε «μεγάλο» format.
MyStr = Format(Time, "Long Time")
Debug.Print MyStr
' Επιστρέφει την τρέχουσα ημερομηνία σε «μεγάλο» format.
MyStr = Format(Date, "Long Date")
Debug.Print MyStr
MyStr = Format(MyTime, "h:m:s")   ' Επιστρέφει "17:4:23".
Debug.Print MyStr
MyStr = Format(MyTime, "hh:mm:ss AMPM")   ' Επιστρέφει "05:04:23 μμ".
Debug.Print MyStr
MyStr = Format(MyDate, "dddd, mmm d yyyy")   ' Επιστρέφει "Τετάρτη, Ιαν 27
1993".
Debug.Print MyStr

' τύποι format του χρήστη
MyStr = Format(5459.4, "##,##0.00")   ' Επιστρέφει "5,459.40".
Debug.Print MyStr
MyStr = Format(334.9, "###0.00")   ' Επιστρέφει "334.90".
Debug.Print MyStr
MyStr = Format(0.5315, "0.00%")   ' Επιστρέφει "53.15%".
Debug.Print MyStr
MyStr = Format("HELLO", "<")   ' Επιστρέφει "hello".
Debug.Print MyStr
MyStr = Format("This is it", ">")   ' Επιστρέφει "THIS IS IT".
Debug.Print MyStr

End Sub
```

Τα αποτελέσματα όπως εκτυπώθηκαν με τη σειρά:



Παρατηρήστε ότι σε ορισμένες σειρές εμφανίζεται το σύμβολο « ' », είτε από την αρχή, είτε από κάποιο σημείο και έπειτα. Το παραπάνω σύμβολο ονομάζεται «ReMark», *και όταν το συναντήσει ο Compiler της Visual Basic, αγνοεί ο,τιδήποτε γράφεται από εκεί και πέρα. Χρησιμοποιείται λοιπόν για προσθήκη σημειώσεων και σχολίων κατά τη σύνταξη του κώδικα.*

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Οι εσωτερικές συναρτήσεις της Visual Basic αποτελούν δεσμευμένες λέξεις. Πρέπει να χρησιμοποιούνται πάντοτε με κάποια μεταβλητή στην οποία θα εκχωρήσουν κάποια τιμή – αποτέλεσμα.
2. Οι παράγωγες συναρτήσεις δεν είναι απ' ευθείας διαθέσιμες στον χρήστη. Πρέπει να συνταχθούν όπως περιγράφεται στο παρόν τεύχος ή στο αρχείο βοηθείας της Visual Basic.
3. Τα αποτελέσματα που επιστρέφουν οι συναρτήσεις πρέπει να καταχωρίζονται σε συμβατού τύπου μεταβλητές.
4. Η MsgBox χρησιμοποιείται για την παρουσίαση των αποτελεσμάτων / μηνυμάτων με συγκεκριμένη από τον χρήστη εμφάνιση.
5. Η InputBox χρησιμοποιείται για εισαγωγή δεδομένων από τον χρήστη. Πρέπει πάντα να καταχωρίζεται σε κάποια μεταβλητή κάθε εισαγωγή που κάνει ο χρήστης.
6. Η Format διαμορφώνει οπτικά το προς εκτύπωση κείμενο / αποτέλεσμα κτλ. Δεν επηρεάζει την τιμή ή το περιεχόμενο της μεταβλητής, σε αντίθεση με τις συναρτήσεις χειρισμού των συμβολοσειρών (κειμένων).

ΕΡΩΤΗΣΕΙΣ

1. Πώς μπορώ να υπολογίσω το τόξο ημιτόνου μιας γωνίας;
2. Πώς μπορώ να αποκόψω τα κενά διαστήματα σε μια μεταβλητή που «κρατάει» κάποιο κείμενο;
3. Πώς μπορώ να πάρω τον τέταρτο χαρακτήρα ενός ονόματος που είναι καταχωρισμένο σε μια μεταβλητή;
4. Πώς μπορώ να υπολογίσω σε πραγματικό χρόνο τη διάρκεια χρήσης ενός προγράμματός μου από τον χρήστη;
5. Μπορεί μια ημερομηνία να καταχωρισθεί σε μεταβλητή τύπου String; Σε μεταβλητή τύπου Variant;
6. Μπορώ να δώσω στον χρήστη την ευχέρεια να εισάγει τιμές μέσα από ένα MsgBox;
7. Από ποια επιμέρους στοιχεία ελέγχου αποτελείται ένα InputBox;
8. Τι τύπου μεταβλητή καταχώρισης πρέπει να χρησιμοποιώ ώστε ο χρήστης να μπορεί να εισάγει δεδομένα μέσω ενός InputBox με ασφάλεια; Έστω ότι περιμένω εισαγωγή πραγματικού αριθμού.

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα project στο οποίο ο χρήστης θα εισάγει 3 αριθμούς και θα λαμβάνει μια εκτύπωση με τίτλο «Μέσος όρος» και αποτέλεσμα το μέσο όρο των 3 αριθμών με 3 δεκαδικούς.

Κεφάλαιο 6

Εντολές και τελεστές σύγκρισης

Η Visual Basic χρησιμοποιεί ορισμένες εντολές και λογικούς τελεστές για τη σύγκριση των επεξεργαζομένων στοιχείων. Για να αντιληφθούμε άνετα το νόημα μιας τέτοιας σύγκρισης αρκεί να «αντιμετωπίσουμε» το εξής ερώτημα:

Ζητούμε από το χρήστη να εισάγει 2 αριθμούς. Αν ο πρώτος αριθμός είναι μεγαλύτερος ή ίσος του δευτέρου, τότε τον ειδοποιούμε ότι πρέπει να επανεισάγει αριθμό. Διαφορετικά το πρόγραμμα συνεχίζει την εκτέλεσή του.

Άλλο παράδειγμα αποτελεί η εισαγωγή κωδικού πρόσβασης σε έναν Η/Υ. Αν δεν εισαχθεί ο σωστός κωδικός, τότε το πρόγραμμα «Ταυτοπροσωπίας» απορρίπτει τον χρήστη.

Πώς επιτυγχάνεται όμως πρακτικά ο έλεγχος που δίνει τη λύση σε προβλήματα αυτού του είδους; Η Visual Basic αντιμετωπίζει τα σχετικά προβλήματα βάσει της ακόλουθης «λογικής» :

Αν συμβαίνει αυτό τότε κάνε εκείνο , αλλιώς κάνε το άλλο.

Εμείς θα αναλύσουμε δύο διαφορετικούς τρόπους σύγκρισης, ή αλλιώς δύο διαφορετικά set εντολών.

If…Then…Else…(End If)

Σύνταξη:
If Συνθήκη Then Αυτό Else Άλλο
Παράδειγμα:

```
MyNum1 = 12
MyNum2 = 23
If MyNum1 > MyNum2 Then MsgBox "Ο πρώτος είναι μεγαλύτερος!" Else
MsgBox "Ο δεύτερος είναι μεγαλύτερος ή ίσος!"
```

Τι πρόκειται να συμβεί όταν εκτελέσουμε τον παραπάνω κώδικα; Ποιο από τα δύο μηνύματα θα εμφανιστεί;

Η παραπάνω σύνταξη των εντολών If...Then...Else αποτελεί την πιο απλή αλλά παράλληλα και την πλέον «αδύναμη» μορφή. Στην πραγματικότητα οι εντολές αυτές μπορούν να χρησιμοποιηθούν ποικιλοτρόπως και με χειρισμό περισσότερων συνθηκών. Πριν προχωρήσουμε όμως, ας δούμε τους τελεστές σύγκρισης.

Τελεστής	Επεξήγηση
=	Ίσο με
>	Μεγαλύτερο από
<	Μικρότερο από
>=	Μεγαλύτερο ή ίσο από
=<	Μικρότερο ή ίσο από
<>	Διαφορά από

Στον παραπάνω πίνακα αναφερόμαστε στη σύγκριση του πρώτου ορίσματος σε σχέση με το δεύτερο (δηλαδή τη σύγκριση του ορίσματος που βρίσκεται αριστερά του τελεστή, σε σχέση με το όρισμα που βρίσκεται στα δεξιά).

Σημείωση

Όρισμα (argument): η τιμή που απαιτείται από μια συνάρτηση, εντολή, υπορουτίνα, κα, προκειμένου να γίνει κάποιος (υπο)λογισμός, π.χ.

- *Cos(3.14), η τιμή 3.14 είναι το όρισμα της εσωτερικής συνάρτησης «συνημίτονο»*
- *MsgBox "Ο πρώτος είναι μεγαλύτερος!", όπου το κείμενο που περιέχεται στα εισαγωγικά αποτελεί το όρισμα για τη συνάρτηση MsgBox και το οποίο θα εκτυπωθεί στην οθόνη ως αποτέλεσμα της εκτέλεσης της εντολής.*

Υπάρχουν κάποιες σχέσεις οι οποίες μας επιτρέπουν να εξετάσουμε μια πιο πολύπλοκη συνθήκη. Ας υποθέσουμε ότι θέλουμε στο παραπάνω παράδειγμα να ελέγξουμε αν ο πρώτος αριθμός είναι μικρότερος ή μεγαλύτερος του δεύτερου και παράλληλα, αν ο πρώτος αριθμός είναι μικρότερος ή μεγαλύτερος του 10. Για να το πετύχουμε, θα πρέπει πιθανόν να χρησιμοποιήσουμε κάποια συνδετική λογική σχέση:

Σχέση	Επεξήγηση
And	Να ισχύει η πρώτη και η δεύτερη (κτλ) συνθήκη
Not	Να μην ισχύει η συνθήκη
Or	Να ισχύει είτε η πρώτη είτε η δεύτερη (κτλ) συνθήκη

Xor	Να ισχύει ή η πρώτη ή η δεύτερη (κτλ) αλλά μόνο η μία

Προτού προχωρήσουμε στην επεξήγηση των παραπάνω, ας δούμε την αναλυτική σύνταξη της If…Then στην μορφή που ονομάζεται «Block If…End If»

```
If συνθήκη [And Συνθήκη] [Or Συνθήκη] Then
'Κώδικας προς εκτέλεση
Else
'Κώδικας προς εκτέλεση
End If
```

Ό,τι βρίσκεται εντός των αγκυλών μπορεί φυσικά να παραλειφθεί. Ας δούμε μερικά παραδείγματα:

```
X1 = 30.45 : Y1 = 23.11
X2 = -49.77 : Y2 = 16.78
Z1 = 100 : Z2 = 0

If X1 > X2 And Y1 < Y2 Then
Z1 = Z2
Else
Z2 = Z1
End If
Debug.Print Z1 , Z2
```

Τι θα εμφανιστεί στην εκτύπωση; Προφανώς : 100 100

```
If X1 < Abs(X2) And (Y1 - Y2) > 0 Then
Z1 = Z2
Else
Z2 = Z1
End If
Debug.Print Z1 , Z2
```

Τι θα εμφανιστεί στην εκτύπωση; Προφανώς : 0 0

```
If X1 < X2 Or Y1 = Y2 Then
Z1 = Z2
Else
Z2 = Z1
End If
Debug.Print Z1 , Z2
```

Τι θα εμφανιστεί στην εκτύπωση; Προφανώς : 100 100

```
If X1 > X2 Xor Y1 > Y2 Then
Z1 = Z2
Else
Z2 = Z1
End If
Debug.Print Z1 , Z2
```

Τι θα εμφανιστεί στην εκτύπωση; Προφανώς : 100 100

Μπορούμε φυσικά να χρησιμοποιήσουμε τους παραπάνω τελεστές (And, Or, Xor, Not) σε οποιονδήποτε συνδυασμό. Μια άλλη εντολή που χρησιμοποιείται με το Block If…End If είναι η ElseIf, που συνδυάζει την κατάσταση Else με ένα νέο έλεγχο συνθήκης If.

```
If X1 > X2 And Y1 < Y2 Then
Z1 = Z2
Debug.Print Z1, Z2

ElseIf X1 > X2 Then
MsgBox "Ισχύει η πρώτη συνθήκη"
ElseIf Y1 < Y2 Then
MsgBox "Ισχύει η δεύτερη συνθήκη"

End If
```

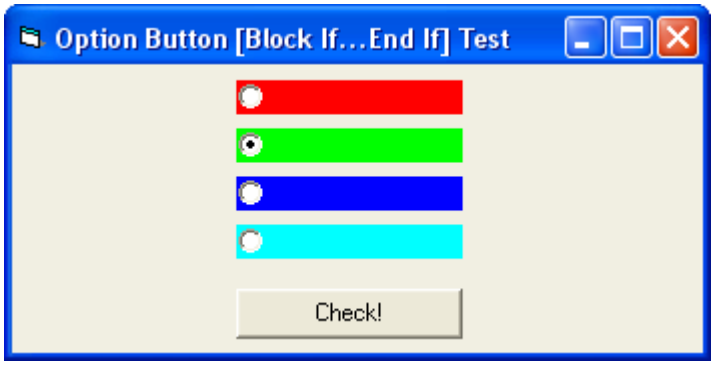
Τι αποτέλεσμα θα πάρουμε; Ας το δούμε πως εκτελείται σειρά προς σειρά. (Θεωρούμε βέβαια ότι οι τιμές των X1,Y1 κτλ είναι αυτές που φαίνονται στην προηγούμενη σελίδα).

```
If X1 > X2 And Y1 < Y2 Then     ' Αν X1 >X2 και Y1 < Y2 τότε:
Z1 = Z2                         ' Δώσε την τιμή Z2 στο Z1
Debug.Print Z1, Z2              ' Εκτύπωσε την τιμή Z1 και Z2
ElseIf X1 > X2 Then             'Αλλιώς αν X1 > του X2 τότε:
MsgBox "Ισχύει η πρώτη συνθήκη" ' Παρουσίασε αυτό το μήνυμα
ElseIf Y1 < Y2 Then             'Αλλιώς αν Y1 < Y2 τότε:
MsgBox "Ισχύει η δεύτερη συνθήκη" 'Παρουσίασε αυτό το μήνυμα
End If                          ' Κλείσιμο του Block If
```

Το αποτέλεσμα θα είναι η εμφάνιση ενός πλαισίου εξόδου μηνύματος που θα αναγράφει ότι «Ισχύει η πρώτη συνθήκη».

Θυμάστε το στοιχείο ελέγχου OptionButton; Ας δώσουμε ένα παράδειγμα της χρήσης του σε σχέση με το Block If.

Σχεδιάζουμε 4 Option Buttons πάνω σε μια φόρμα, αλλάζουμε το χρώμα φόντου σε κόκκινο, πράσινο, μπλε και κυανό και τα ονόματά τους σε OptRed, OptGreen, OptBlue, OptCyan, αντίστοιχα. Σβήνουμε εντελώς τις λεζάντες των πλήκτρων επιλογής. Σε όποιο από τα τέσσερα Option Buttons θέλουμε, θέτουμε την ιδιότητα Value = True, για να εμφανίζεται κάποιο προεπιλεγμένο (στην εικόνα είναι το πράσινο). Τα τακτοποιούμε πάνω στη φόρμα και προσθέτουμε ένα κουμπί με λεζάντα «Check!». Η φόρμα μας θα μοιάζει με την παρακάτω εικόνα:



Μέσα στο συμβάν κλικ του CommandButton γράφουμε τον παρακάτω κώδικα:

```
Private Sub Command1_Click()
If OptRed.Value = True Then
MsgBox "Επιλέξατε το κόκκινο!"

    ElseIf OptGreen.Value = True Then
    MsgBox " Επιλέξατε το πράσινο!"

    ElseIf OptBlue.Value = True Then
    MsgBox " Επιλέξατε το μπλε!"

    ElseIf OptCyan.Value = True Then
    MsgBox " Επιλέξατε το γαλάζιο!"

End If
End Sub
```

Εκτελέστε τώρα το παραπάνω πρόγραμμα (Menu [Run] -> [Start] ή πατώντας F5). Κάντε κάποια επιλογή και πατήστε το κουμπί «Check!». Προφανώς «βλέπετε» αυτό που επιλέξατε !

Η εντολή ElseIf είναι χρήσιμη όταν τα πιθανά «σενάρια» επιλογών είναι πάνω από ένα. Αν είναι μόνο ένα, τότε η εντολή Else είναι υπέρ αρκετή.

Έχετε ήδη αντιληφθεί τη χρήση του Block If…End If, και μάλιστα μόλις τώρα γράψατε και το πρώτο σας αντικειμενοστραφές πρόγραμμα στην Visual Basic!

Η If…End If όμως δεν είναι ο μοναδικός τρόπος για να κάνουμε κάποιες συγκρίσεις συνθηκών και επιλογές σε ένα πρόγραμμα. Θα δούμε αμέσως τώρα ένα άλλο Block εντολών το οποίο θα αποδειχθεί πολύ χρήσιμο.

Select Case

Σύνταξη:

```
Select Case όρισμα

    Case α
    ...
    Case β
    ...
    Case γ
    ...
End Select
```

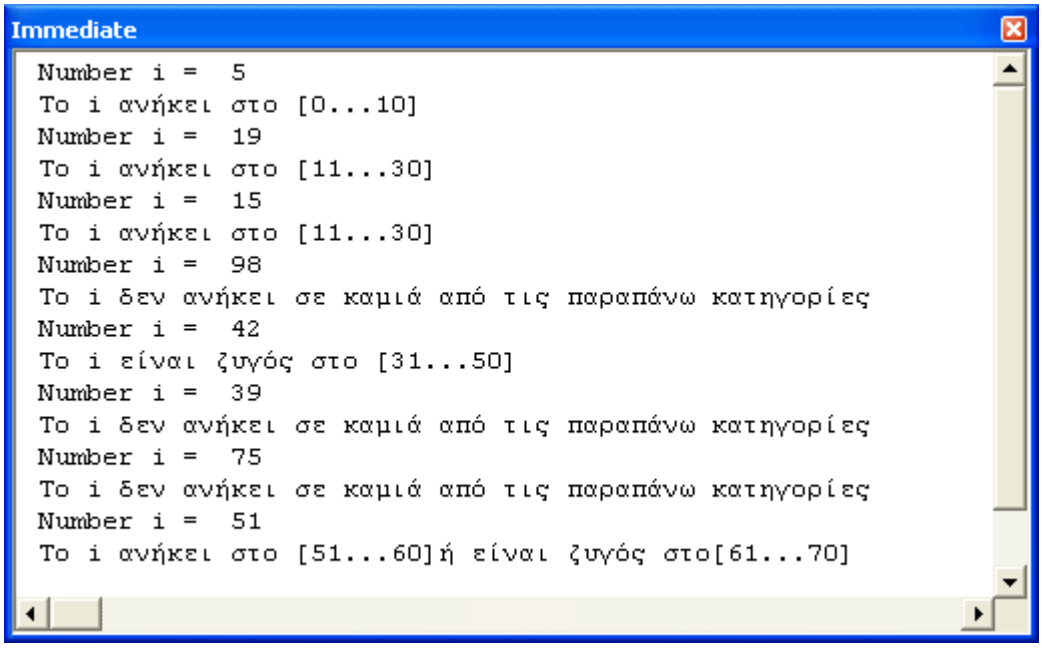
Η Select Case είναι εξαιρετικά χρήσιμη όταν θέλουμε, ανάλογα με τη συνθήκη που ελέγχουμε, να εκτελέσουμε ένα Block κώδικα. Χρησιμοποιεί κάποιους δικούς της συνδέσμους, όπως το κόμμα « , », τη λέξη «To», τη λέξη «Is» και τη λέξη «Else».

Σχέση	Επεξήγηση
To	Από την πρώτη έκφραση έως και τη δεύτερη (num1 To num2)
Is	+ τελεστή/ές σύγκρισης: (π.χ. Is > 23 , Is<=12 κτλ).
,	Είναι το αντίστοιχο And του Block If. (πχ. 12, 23, 17, 9 κτλ)
Else	Όταν δεν ισχύει οτιδήποτε άλλο. Ανάλογο του Else του Block If.

Ας δούμε πως χρησιμοποιούνται στο παρακάτω παράδειγμα. Γράψτε τον παρακάτω κώδικα στο συμβάν κλικ ενός κουμπιού.

```
Private Sub Command1_Click()
Dim i As Integer
Randomize Timer
i = Int(Rnd * 100)                    'Τυχαίος ακέραιος στο [0,100)
Debug.Print "Number i = "; i

Select Case i

    Case 0 To 10
    Debug.Print "Το i ανήκει στο [0...10]"
    Case 11 To 30
    Debug.Print " Το i ανήκει στο [11...30]"
    Case 32, 34, 36, 38, 40, 42, 44, 46, 48, 50
    Debug.Print " Το i είναι ζυγός στο [31...50]"
    Case 51 To 60, 62, 64, 66, 68, 70
    Debug.Print " Το i ανήκει στο [51...60] ή είναι ζυγός στο [61...70]"
    Case Else
    Debug.Print " Το i δεν ανήκει σε καμιά από τις παραπάνω κατηγορίες "

End Select

End Sub
```

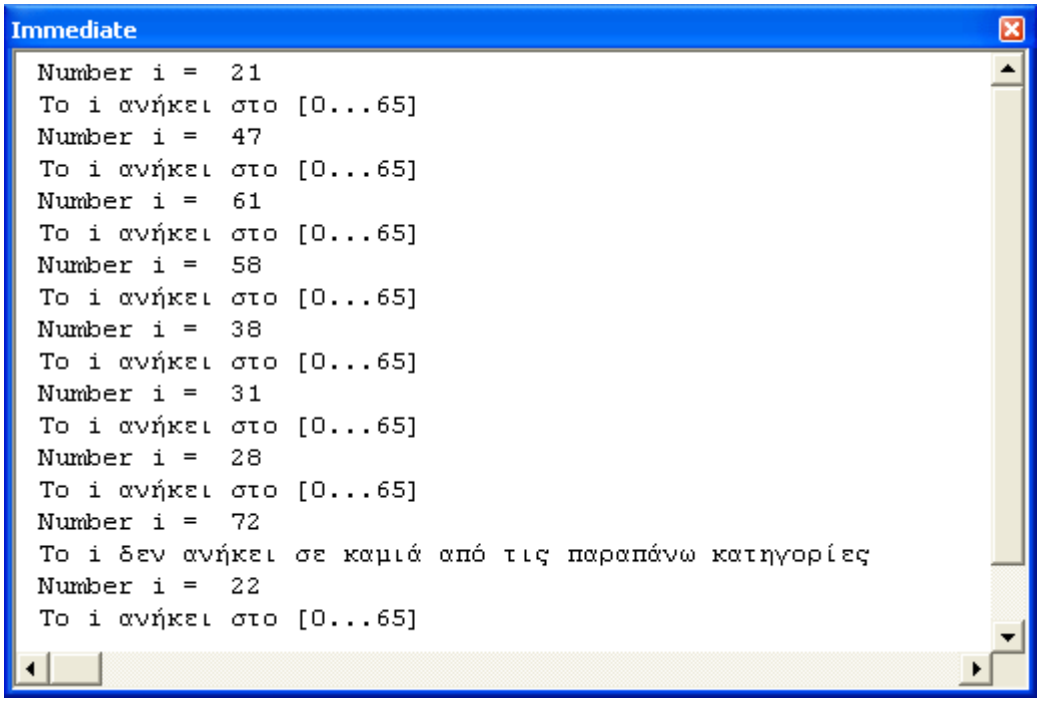
Δείτε τα παρακάτω αποτελέσματα όπως προέκυψαν από μερικά «πατήματα» του κουμπιού:



Πρέπει εδώ να σημειώσουμε ότι σε αντίθεση με το Block If, στο Select Case «μετράει» η σειρά τοποθέτησης των Case στον κώδικα:

```
Private Sub Command2_Click()
Dim i As Integer
Randomize Timer
i = Int(Rnd * 100)                    'Τυχαίος ακέραιος στο [0,100)
Debug.Print "Number i = "; i

Select Case i

    Case 0 To 65
    Debug.Print " Το i ανήκει στο [0...65]"
    Case 10 To 30
    Debug.Print " Το i ανήκει στο [11...30]"
    Case 32, 34, 36, 38, 40, 42, 44, 46, 48, 50
    Debug.Print " Το i είναι ζυγός στο [31...50]"
    Case Is > 66
    Debug.Print " Το i δεν ανήκει σε καμιά από τις παραπάνω κατηγορίες "

End Select

End Sub
```

Δείτε τα παρακάτω αποτελέσματα όπως προέκυψαν από μερικά «πατήματα» του κουμπιού. Τι έπρεπε λογικά να γίνει;



Το i = 21 ανήκει τόσο στην πρώτη όσο και στην δεύτερη Case του Block Select Case. Ωστόσο περιήλθε στην πρώτη κατά σειρά γραφής κατηγορία και φυσικά αγνοήθηκε στην επόμενη! Το ίδιο συνέβη και για τα i= 38, 28, 22.

Μόνο μια περίπτωση Case εκτελείται σε κάθε Select Case Block.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Ένα Block If…End If πρέπει πάντοτε να «ανοίγει» με μια πρόταση «If» και να κλείνει με μια πρόταση «End If».
2. Μπορούμε να χρησιμοποιήσουμε πολλά Block If, το ένα μέσα στο άλλο.
3. Το Select Case Block πρέπει πάντοτε να «ανοίγει» με μια πρόταση «Select Case» και να κλείνει με μια πρόταση «End Select».
4. Στην Select Case γράφουμε τις περιπτώσεις Case με σειρά προτεραιότητας εκτέλεσης.
5. Οι εντολές σύγκρισης προσδιορίζουν την ροή ενός προγράμματος.
6. Η εντολή Else χρησιμοποιείται για να συμπεριλάβει αποτελέσματα σύγκρισης για τα οποία δεν υπάρχει άλλη πρόβλεψη χειρισμού.
7. Η εντολή ElseIf χρησιμοποιείται για να διαφοροποίησει μια νέα συνθήκη από την ήδη υπάρχουσα.

ΕΡΩΤΗΣΕΙΣ

1. Ποια πιστεύετε ότι είναι η καλύτερη μέθοδος σύγκρισης (Block If ή Select Case); Ποια είναι καταλληλότερη για σύγκριση αριθμών;
2. Με ποια μέθοδο μπορώ να συγκρίνω ευκολότερα λογικές συνθήκες;
3. Πώς συντάσσεται η Block If…End If;
4. Τι παρέχει ο τελεστής Xor;
5. Πού χρησιμοποιείται ο τελεστής Is;
6. Πόσες γραμμές κώδικα μπορεί να περιέχει η Case;
7. Μπορεί μια Case να περιέχει ένα Block If…End If; Μπορεί μέσα σε ένα Block If…End If να υπάρχει ένα Block Select Case;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Τροποποιήστε το project με τα OptionButtons και αντί για έλεγχο με Block If…End If, κάντε τον έλεγχο με Block Select Case. Προσθέστε μεταβλητές, αν το κρίνετε απαραίτητο.
2. Δημιουργήστε ένα project με πλαίσιο αναδιπλούμενης λίστας (ComboBox), εισάγετε 6 ονόματα, προσθέστε 2 κουμπιά και κάντε έλεγχο με Block If…End If και Block Select Case για να εκτυπώσετε την επιλογή του χρήστη.
3. Δημιουργήστε ένα project στο οποίο ο χρήστης θα εισάγει έναν ακέραιο αριθμό από το 10 έως το 30. Ελέγξτε αν ο αριθμός ανήκει στο παραπάνω διάστημα. Δημιουργήστε έναν τυχαίο αριθμό στο ίδιο διάστημα και συγκρίνετέ τον με τον αριθμό που εισήγαγε ο χρήστης. Αν ο τυχαίος είναι μικρότερος ή μεγαλύτερος, ο χρήστης να λαμβάνει ένα σχετικό μήνυμα. Αν είναι ίσος, τότε να εκτυπώνετε τον αριθμό που σας έδωσε και να ζητάτε το διπλάσιό του. Αν ο χρήστης τον εισάγει σωστά, τότε να το επιβεβαιώνετε με ένα σχετικό μήνυμα.

Κεφάλαιο 7

Βρόγχοι (επαναληπτικές διαδικασίες)

Βρόγχοι στη Visual Basic ονομάζονται οι επαναληπτικές διαδικασίες που δημιουργούμε με σκοπό την εκτέλεση κάποιων τμημάτων κώδικα περισσότερες από μία φορές.

Όλες οι επαναληπτικές διαδικασίες χαρακτηρίζονται από κάποια συνθήκη επανάληψης. Έτσι, υπάρχουν οι σταθερές επαναληπτικές διαδικασίες και οι μεταβλητές. Ας παρουσιάσουμε το εξής παράδειγμα για να κατανοήσετε εύκολα την διαφορά των παραπάνω:

Σε έναν αγώνα μπάσκετ οι ομάδες αγωνίζονται για 40 λεπτά. Με το πέρας των 40 λεπτών η διαδικασία τερματίζει. Σε έναν αγώνα σκάκι όμως, οι παίκτες αγωνίζονται μέχρι κάποιος να κερδίσει. Ο χρόνος για τους παίκτες δεν έχει σημασία. Θα λέγαμε λοιπόν ότι όσον αφορά τη διάρκεια, το πρώτο είναι σταθερό ενώ το δεύτερο μεταβλητό.

Κάπως έτσι διαχωρίζονται και οι βρόγχοι στην Visual Basic. Για κάποιους γνωρίζουμε και θέτουμε τον αριθμό επαναλήψεων, ενώ για κάποιους άλλους η συνθήκη λήξης της επανάληψης δεν είναι σταθερή, αλλά εξαρτάται από κάποιον παράγοντα.

Λογικά, διαφορετικό κώδικα θα γράψουμε σε ένα project όπου θα ζητήσουμε από τον χρήστη να εισάγει 10 αριθμούς και διαφορετικό αν του ζητήσουμε να εισάγει αριθμούς μέχρι να πετύχει τον αριθμό που επιθυμούμε (άγνωστος για τον χρήστη). Στην πρώτη περίπτωση απλά θα εισάγει 10 αριθμούς και η εκτέλεση θα τερματιστεί. Στην δεύτερη, μπορεί να πετύχει τον αριθμό με την πρώτη εισαγωγή (και να τελειώσει), με την 100^{η} ή...ποτέ!

Στη συνέχεια θα αναλύσουμε τους παραπάνω βρόγχους, ξεκινώντας από αυτόν που χρησιμοποιεί προκαθορισμένο κύκλο επαναλήψεων.

For … [Step] … Next

Χρησιμοποιείται για σταθερό κύκλο επαναλήψεων.

Σύνταξη:

```
For μεταβλητή = αρχή To τέλος [Step βήμα]
'
'Κώδικας προς εκτέλεση
'
Next μεταβλητή
```

Για να εξηγήσουμε τον παραπάνω κώδικα θα δούμε ορισμένα απλά παραδείγματα:

```
Private Sub Command1_Click()
Dim i As Integer

For i = 1 To 10        'Αρχή βρόγχου από 1 μέχρι το i να φτάσει την τιμή 10.
Debug.Print i * 10
Next i                 'Το i παίρνει την τιμή (i+1)

End Sub
```

Ας δούμε τον παραπάνω κώδικα σειρά προς σειρά

```
For i = 1 To 10        'Ξεκινάει ο βρόγχος για 10 επαναλήψεις. Στην ουσία χρειαζόμαστε μια μεταβλητή (εδώ η i), η οποία αυξάνεται κατά την τιμή του βήματος που αναγράφεται στην Step (μόλις φτάσει στην Next). Εδώ που η Step παραλείπεται, αυτομάτως θεωρείται ίση με τη μονάδα.
Debug.Print i * 10     'Εκτυπώνει την τιμή του i πολλαπλασιασμένη με 10.
Next i                 'Κάθε φορά που φτάνει η εκτέλεση στην εντολή Next η τιμή της μεταβλητής i αυξάνεται κατά την τιμή του βήματος (εδώ που δεν θέσαμε τιμή θεωρήθηκε = 1)
```

Αν το εκτελέσουμε, θα μας εκτυπώσει με την σειρά το 10,20,…,90,100. Μόλις το i γίνει 11 και η εκτέλεση επιστρέψει στη For, ο βρόγχος δεν εκτελείται πλέον, διότι το i έχει περάσει τα όρια, και παραβλέπεται. Η εκτέλεση του προγράμματος συνεχίζεται αμέσως μετά την εντολή Next.

Παρατηρήστε στον παρακάτω πίνακα, τί συμβαίνει κατά την εκτέλεση του προγράμματος:

Επανάληψη	i	Εκτύπωση
1η	1	10
2η	2	20
3η	3	30
4η	4	40
5η	5	50
6η	6	60
7η	7	70
8η	8	80
9η	9	90
10η	10	100
	11	(Τίποτα) Αφού το i έγινε 11 δεν ξαναεκτελέστηκε ο βρόγχος)

Ας δούμε το επόμενο παράδειγμα:

```
Private Sub Command1_Click()
Dim i As Integer

For i = 1 To 10 Step 2 'Αρχή βρόγχου από 1 έως να φτάσει το i την τιμή 10.
Debug.Print i * 10
Next i                  'Το i παίρνει την τιμή (i+2)

End Sub
```

Επανάληψη	i	Εκτύπωση
1η	1	10
2η	3	30
3η	5	50
4η	7	70
5η	9	90
	11	(Τίποτα) Αφού το i έγινε 11 δεν ξαναεκτελέστηκε ο βρόγχος)

```
Private Sub Command3_Click()
For i = 10 To 3 Step -2
Debug.Print i * 10
Next i
End Sub
```

Στο παραπάνω παράδειγμα ξεκινάμε από αρχική τιμή 10 και επιθυμούμε ο βρόγχος να εκτελείται, ώσπου το i να γίνει ίσο με 3. Η μείωση του i γίνεται με

βήμα = -2 (στην εντολή Next). Όταν το i γίνει μικρότερο του 3, ο βρόγχος For…Next δεν θα ξαναεκτελεστεί.

Επανάληψη	i	Εκτύπωση
1η	10	100
2η	8	80
3η	6	60
4η	4	40
	2	(Τίποτα) Αφού το i έγινε 2 (<3) δεν ξαναεκτελέστηκε ο βρόγχος)

Ας μελετήσουμε τώρα ένα λίγο πιο δύσκολο παράδειγμα στο οποίο θα χρησιμοποιήσουμε βρόγχο μέσα σε βρόγχο (embedded loop):

```
Private Sub Command1_Click()
For i = 1 To 4

    For j = 3 To 1 Step -1
    Debug.Print i, j, i * j * 10
    Next j

Next i

End Sub
```

Θα περιγράψουμε λίγο πρώτα με λόγια τι συμβαίνει παραπάνω και έπειτα θα δείτε τον πίνακα των εκτυπώσεων.

Α) For i = 1 To 4 Ξεκινάει η διαδικασία για τον βρόγχο του i για 4 επαναλήψεις (από 1 έως 4), με αρχική τιμή του i = 1 και βήμα = 1.

Β) For j = 3 To 1 Step -1 Ξεκινάει η διαδικασία για τον βρόγχο του j για 3 επαναλήψεις (από 3 έως 1), με αρχική τιμή του j = 3 και βήμα = -1.

Γ) Debug.Print i, j, i * j * 10 Εκτυπώνει τις τιμές του i, j, i * j * 10

Δ) Next j Μειώνει την τιμή του j κατά 1 και επανέρχεται στο στάδιο **Β**.

Όταν η τιμή του j γίνει 0 (<1) τότε προχωράει στο βήμα **Ε**

Ε) Next i Αυξάνει την τιμή του i κατά 1 και επανέρχεται στο βήμα **Α**. Όταν η τιμή του i γίνει 5 (>4) τότε τερματίζεται ο βρόγχος και συνεχίζει με τις επόμενες εντολές (αν υπάρχουν) του προγράμματος.

i	j	Εκτύπωση i , j, i * j * 10
1	3	1 3 30
	2	1 2 20
	1	1 1 10
2	3	2 3 60
	2	2 2 40
	1	2 1 20
3	3	3 3 90
	2	3 2 60
	1	3 1 30
4	3	4 3 120
	2	4 2 80
	1	4 1 40

Ο εσωτερικός βρόγχος (του j) εκτελείται κάθε φορά από την αρχή για κάθε επανάληψη του βρόγχου i, διότι «ανήκει» σε αυτόν.

Καταλάβαμε πλέον πώς λειτουργεί ένας βρόγχος For…Next. Μέσα σε ένα βρόγχο φυσικά μπορούμε να συμπεριλάβουμε κάθε λογής κώδικα όπως Block If…End If κτλ. Πολλές φορές όμως παρουσιάζεται η ανάγκη να τερματίσουμε «βίαια» έναν βρόγχο, ίσως διότι καταλήξαμε σε αυτό που ψάχναμε μέσω επαναλήψεων νωρίτερα από το αναμενόμενο.

Υποθέστε για παράδειγμα ότι έχουμε 300 σημεία με συντεταγμένες X, Y, Z και μέσω ενός βρόγχου ψάχνουμε να βρούμε κάποιο σημείο που γνωρίζουμε ότι έχει X = 100, Y=200 και Z = 45. Αν το σημείο αυτό είναι το 25ο στη σειρά και το βρούμε, δεν υπάρχει λόγος πρακτικά να συνεχίσουμε το ψάξιμο. Σε αυτή την περίπτωση, εισάγουμε μέσα σε ένα βρόγχο την πρόταση Exit For.

Πριν σας παρουσιάσουμε ένα σχετικό παράδειγμα, νομίζουμε ότι είναι σκόπιμο να δούμε αρχικά μια χρήση του βρόγχου For…Next σε συνδυασμό με έναν πίνακα μεταβλητών. Αυτό που θα κάνουμε είναι να δημιουργήσουμε έναν δισδιάστατο πίνακα μεταβλητών, στον οποίο θα δώσουμε τυχαίες τιμές.

```
Private Sub Command1_Click()
Dim Cords(100, 2) As Single
Dim j As Integer, k As Integer

For j = 1 To 100
    For k = 1 To 2
    Randomize Timer
    Cords(j, k) = Rnd * 200 - 100
    Next k
Next j
End Sub
```

Στον παραπάνω κώδικα δηλώσαμε έναν πίνακα με 100 γραμμές και 2 στήλες. Σε κάθε στοιχείο του πίνακα δώσαμε τυχαίες τιμές στο διάστημα (-100...100). Είδατε πως λειτουργεί ο βρόγχος; Για κάθε γραμμή του πίνακα ανοίγει ένας νέος βρόγχος ο οποίος εκτελείται για κάθε στήλη του πίνακα. Διαφορετικά θα έπρεπε, για να επιτύχουμε το ίδιο αποτέλεσμα, να γράψουμε 200 δηλώσεις και 200 εκχωρήσεις μεταβλητών!!!

Τώρα θα επεκτείνουμε λίγο το πρόγραμμα που γράψαμε και θα ψάξουμε να βρούμε τη γραμμή εκείνη του πίνακα της οποίας και οι 2 τιμές (δηλαδή τα στοιχεία (n,1) και (n,2) έχουν τιμές από 0 εως 20. Επειδή όμως δώσαμε τιμές με τυχαίο τρόπο είναι πιθανόν να υπάρχουν περισσότερα από 1 σημεία στο παραπάνω διάστημα ή και κανένα. Εμείς θα θεωρήσουμε αυθαίρετα ότι μόνο ένα (του οποίου όμως δεν γνωρίζουμε την θέση στον πίνακα) πληροί τις παραπάνω προϋποθέσεις.

Ο κώδικας θα θεωρηθεί ότι γράφεται κάτω από το Next j του παραπάνω παραθέματος.

```
For j = 1 To 100

    If Cords(j, 1) >= 0 And Cords(j, 1) <= 20 Then

        If Cords(j, 2) >= 0 And Cords(j, 2) <= 20 Then
            MsgBox "Το βρήκα! Είναι το " & j
            Exit For
        End If

    End If

Next j
```

Τι συμβαίνει εδώ;

>Ξεκινάει μια επανάληψη για τα 100 στοιχεία – γραμμές.

>>Ελέγχω το πρώτο υποστοιχείο – στήλη. ΑΝ πληροί τη συνθήκη (μεγαλύτερο ή ίσο του μηδέν ΚΑΙ μικρότερο ή ίσο το 20) ΤΟΤΕ

>>>Ελέγχω το δεύτερο υποστοιχείο – στήλη. ΑΝ και αυτό πληροί την ανάλογη συνθήκη ΤΟΤΕ

α)Εμφανίζω ένα μήνυμα ότι το στοιχείο βρέθηκε στη θέση j.

β)Σπάω «βίαια» το βρόγχο, διότι δεν έχει νόημα το επιπλέον ψάξιμο.

<<<Τέλος του Block του δεύτερου ελέγχου.

<< Τέλος του Block του πρώτου ελέγχου.

<Τέλος επανάληψης βρόγχου, αύξηση του μετρητή j και επιστροφή στην αρχή.

Σπάζοντας το βρόγχο μεταφέρουμε την εκτέλεση του προγράμματος στις εντολές που ακολουθούν το Next j (αν υπάρχουν).

Ένα τελευταίο χαρακτηριστικό που θα αναφέρουμε για τον βρόγχο For…Next είναι ότι κατά την εκτέλεσή του δεν μπορούμε να αλλάξουμε τα όριά του ή το βήμα του.

```
Private Sub Command1_Click()
Dim Gend As Integer, Gstep As Integer, Counter As Integer

Gend = 20
Gstep = 1

For Counter = 1 To Gend Step Gstep
Gend = 5
Gstep = 2
Debug.Print Counter
Next Counter

End Sub
```

Στην αρχή θέτουμε ως όριο στον βρόγχο την τιμή του Gend = 20 και βήμα αύξησης την τιμή του Gstep = 1. Μέσα στον βρόγχο αλλάζουμε τις τιμές των Gend και Gstep όπως φαίνεται. Ο βρόγχος όμως αγνοεί αυτές τις αλλαγές και εκτελείται κανονικά ως σαν να μην γράφτηκαν ποτέ. Ως εκτύπωση θα πάρουμε τις τιμές του Counter: 1, 2, … ,19, 20.

Θα συνεχίσουμε με τους βρόγχους μεταβλητής επαναληπτικότητας (επανάληψη βάση δυναμικής συνθήκης). Ουσιαστικά πρόκειται για ένα βρόγχο αλλά με παραλλαγές. Ας δούμε πρώτα την απλούστερη και πιο διαδεδομένη του μορφή.

While…Wend

Χρησιμοποιείται για μεταβλητό κύκλο επαναλήψεων.

Σύνταξη:

```
While συνθήκη
'
'Κώδικας προς εκτέλεση
'
Wend
```

Εκτελεί ένα μεταβλητό κύκλο επαναλήψεων ανάλογα με τη συνθήκη.
Παράδειγμα:

```
Private Sub Command1_Click()
Dim x As Single
x = 10

While x >= 0 'Για όσο το x είναι μεγαλύτερο ή ίσο του 0
Debug.Print x 'Εκτύπωσε το x
x = x - 1      'Ελλάτωσε την τιμή του x κατά 1
Wend           'Επέστρεψε για επανέλεγχο της συνθήκης

End Sub
```

Η παραπάνω διαδικασία θα μπορούσε να εκτελείται επ' αόριστον ή να μην εκτελεστεί ποτέ (ούτε μια φορά). Εμείς βάλαμε την x να μειώνεται κατά 1 έτσι ώστε να σταματήσει ο βρόγχος.

Δείτε το παρακάτω παράδειγμα, όπου το αν θα σταματήσει ο βρόγχος αφήνεται στην τύχη(!) του χρήστη:

```
Private Sub Command1_Click()
Dim iRand As Integer, iUser As Integer
Randomize Timer
iRand = Int(Rnd * 20) + 1
iUser = 0

While iRand <> iUser
iUser = InputBox("Δώσε έναν αριθμό από το [1...20]", "Εισαγωγή δεδομένων")
Wend

MsgBox "Συγχαρητήρια! Ήταν το " & iUser, vbExclamation, "Αποτέλεσμα!"
End Sub
```

Επιλέγεται ένας τυχαίος ακέραιος στο διάστημα [1...20].
Θέτουμε τον αριθμό του χρήστη στο 0, δηλαδή εκτός του διαστήματος.

Εισερχόμαστε σε μια επαναληπτική διαδικασία η οποία θα εκτελείται όσο ο τυχαίος αριθμός που έχει επιλεγεί θα είναι διαφορετικός από αυτόν που εισάγει ο χρήστης. Αν ο χρήστης πετύχει τον αριθμό, η επαναληπτική διαδικασία τερματίζεται και η εκτέλεση του προγράμματος συνεχίζεται μετά την πρόταση Wend. Εκεί εκτελείται και μια εντολή πλαισίου εξόδου μηνύματος προς επιβράβευση του χρήστη!

Ας δούμε το παράδειγμα με τον έλεγχο των στοιχείων του πίνακα (όπου είχαμε χρησιμοποιήσει βρόγχο For…Next σε συνδυασμό με Exit For). Θεωρούμε ότι ήδη έχουμε δώσει τιμές (όπως παραπάνω) στον πίνακα και πάμε κατευθείαν να δούμε το τμήμα του ελέγχου. Επίσης πρέπει να πούμε ότι αν σε έναν πραγματικό έλεγχο δεν ικανοποιηθεί ποτέ η συνθήκη μας, θα παγιδευτούμε μέσα στο βρόγχο, και το πρόγραμμα θα κολλήσει.

```
j = 1
While Cords(j, 1) < 0 Or Cords(j, 1) > 20 Or Cords(j, 2) < 0 Or Cords(j, 2) > 20
j = j + 1
Wend
MsgBox "Βρέθηκε, είναι το " & j
```

Αυτό που συμβαίνει είναι ότι ο βρόγχος εκτελείται όσο ικανοποιείται η συνθήκη, η οποία είναι αντίθετη αυτής που επιθυμούμε! (Δηλαδή η επανάληψη συνεχίζεται για όσο τα στοιχεία είναι εκτός των δικών μας ορίων). Η εκτέλεση του βρόγχου έχει ως αποτέλεσμα της αύξησης του δείκτη j του πίνακά μας. Όταν η συνθήκη ικανοποιηθεί τότε τερματίζεται η επαναληπτική διαδικασία και λαμβάνουμε το αποτέλεσμα.

Αυτή που είδαμε είναι και η απλούστερη μορφή σύνταξης της While…Wend. Με το πέρασμα του χρόνου, η Microsoft άλλαξε τον τρόπο σύνταξης της εντολής και συμπεριέλαβε νέα στοιχεία.

Do While / Until … Loop

Χρησιμοποιείται για μεταβλητό κύκλο επαναλήψεων.

Σύνταξη:

Do While συνθήκη ' 'Κώδικας προς εκτέλεση [Exit Do] ' Loop	Σε αυτή την περίπτωση το μόνο που αλλάζει είναι ο τρόπος σύνταξης (Do…Loop). Σαν αποτέλεσμα και τρόπο λειτουργίας είναι ακριβώς το ίδιο με το απλό While…Wend. Έχει προστεθεί όμως δυνατότητα Exit Do για βίαιο τερματισμό του βρόγχου (κάτι αντίστοιχο με το Exit For).
Do Until συνθήκη ' 'Κώδικας προς εκτέλεση [Exit Do] ' Loop	Στη σύνταξη του βρόγχου με αντικατάσταση του While με το Until, ο βρόγχος εκτελείται μέχρι να ικανοποιηθεί η συνθήκη.

Παράδειγμα κώδικα με χρήση Do Until…Loop

```
Private Sub Command1_Click()
q = 10

Do Until q >= 30
Debug.Print q
q = q + 1
Loop

End Sub
```

Η διαδικασία σταματάει όταν το q φτάσει την τιμή 30. Αυτό που κάνει ο βρόγχος είναι να εκτελείται Ώσπου το q να φτάσει την επιθυμητή τιμή.

Στους βρόγχους μεταβλητής επαναληπτικότητας, η τιμή της μεταβλητής που ελέγχουμε (ή οποιασδήποτε συνθήκης) μένει αναλλοίωτη. Δεν ισχύει αυτό που ισχύει για το βρόγχο For…Next όπου το Next αλλάζει την τιμή της μεταβλητής που ελέγχει προσθέτοντάς της το βήμα. Στους βρόγχους με Loop ή Wend πρέπει ο χρήστης (ή κάποια άλλη διαδικασία) να ικανοποιήσει τη συνθήκη ελέγχου από τον βρόγχο.

Οι παραπάνω Do…Loop έχουν και εναλλακτικό τρόπο σύνταξης:

Do ' 'Κώδικας προς εκτέλεση [Exit Do] ' Loop While συνθήκη	Η συνθήκη πλέον ελέγχεται στο τέλος του βρόγχου και όχι στην αρχή. Αυτό το χαρακτηριστικό δίνει μεγάλη ευελιξία στην χρήση των Do…Loop βρόγχων, αφού επιτρέπει τον κώδικα εντός του βρόγχου να εκτελεστεί τουλάχιστον μια φορά, και έπειτα να γίνει ο έλεγχος κατά πόσον ικανοποιείται η συνθήκη. Υποθέστε ότι σας ζητείται κάποιος κωδικός ασφαλείας για να εισέλθετε σε ένα σύστημα. Με τον τρόπο αυτό, σας δίνεται η ευκαιρία να εισάγεται έναν κωδικό και εάν δεν είναι ο σωστός, τότε να σας ξαναζητηθεί.
Do ' 'Κώδικας προς εκτέλεση [Exit Do] ' Loop Until συνθήκη	

```
Private Sub Command1_Click()
Do
MyAnswer = InputBox("Πληκτρολόγησε τον κωδικό εισόδου", "Έλεγχος ασφαλείας")
Loop Until MyAnswer = "LetMeIn!"

MsgBox "Καλώς ήρθατε!"

End Sub
```

```
Private Sub Command1_Click()
Do
MyAnswer = InputBox("Πληκτρολόγησε τον κωδικό εισόδου", "Έλεγχος ασφαλείας")
Loop While MyAnswer <> "LetMeIn!"

MsgBox "Καλώς ήρθατε!"

End Sub
```

Τα παραπάνω προγράμματα κάνουν ακριβώς το ίδιο πράγμα!!!

Θα δούμε ένα τελευταίο παράδειγμα για το πως μπορούμε να συμπληρώσουμε στοιχεία σε ένα πίνακα μεταβλητών στον οποίο δεν γνωρίζουμε το πλήθος των στοιχείων που μπορεί να δεχτεί:

```
Private Sub Command1_Click()
Dim Ar(15) As Single, i As Integer

Do
i = i + 1
Randomize Timer
Ar(i) = Rnd * 100
Debug.Print i, Ar(i)
Loop Until i = UBound(Ar, 1)

End Sub
```

Στο παραπάνω παράδειγμα έστω ότι δεν γνωρίζουμε ότι ο πίνακας Ar έχει δηλωθεί με πλήθος στοιχείων = 15. Αγνοώντας λοιπόν αυτό, προσπαθούμε να «γεμίσουμε» πλήρως τον πίνακα Ar με τυχαίες τιμές.

Ξεκινάμε ένα βρόγχο Do…Loop Until, στον οποίο μέσα χρησιμοποιούμε ένα μετρητή του οποίου αυξάνουμε την τιμή κατά ένα σε κάθε επανάληψη. Η συνθήκη σύγκρισης που χρησιμοποιούμε περιέχει μια άγνωστη έως τώρα πρόταση, την UBound. Η Ubound μας δίνει το πάνω όριο στοιχείων του πίνακα (δηλαδή εδώ το 15). Η σύνταξή της περιλαμβάνει το όνομα του πίνακα στον οποίο αναφερόμαστε (Ar) καθώς και στην στήλη η οποία μας ενδιαφέρει. Εδώ προφανώς αφού ο πίνακας είναι μονοδιάστατος, η τιμή της στήλης ισούται με 1.

Όταν λοιπόν η εκτέλεση φτάσει στην Loop, και εφόσον ο μετρητής δείχνει μικρότερη τιμή από την επιθυμητή, η επανάληψη θα συνεχιστεί (η εκτέλεση του κώδικα θα μεταφερθεί πάλι στην γραμμή του Do.

Σημείωση

Παράλληλα με την Ubound, την οποία εξηγήσαμε, υπάρχει και η εντολή LBound η οποία μας δίνει το κάτω όριο των στοιχείων που μπορεί να δεχτεί ένας πίνακας. Η σύνταξή της είναι ίδια με της UBound. Αλλά ποια είναι η χρησιμότητά της; Δεν ξεκινάνε όλοι οι πίνακες με πρώτο στοιχείο το 1, δεύτερο το 2 κτλ;
Η απάντηση είναι όχι.
Ο πίνακας τον οποίο δηλώνουμε μπορεί να πάρει «περίεργες» για τα μαθηματικά διαστάσεις. Δείτε λίγο της παρακάτω δηλώσεις, καθώς και τις τιμές που παίρνουν οι UBound, LBound.

Dim …	UBound		LBound		Πλήθος στοιχείων	
MyAr(-6 To 22)	22		-6		22-(-6)=28	
MyAr(-6 To 22, -3 To 8)	22	8	-6	-3	28	11

Η εντολή To στις παραπάνω δηλώσεις δείχνει το εύρος του πίνακα: -6 έως 22, -3 έως 8.

Όπως ίσως θα παρατηρήσατε, πρακτικά δεν υπάρχει τίποτα που να μπορεί να γίνει με τη χρήση ενός βρόγχου σταθερών επαναλήψεων και να μην μπορεί να γίνει με έναν βρόγχο μεταβλητών επαναλήψεων. Η επιλογή του κατάλληλου βρόγχου δείχνει την ικανότητα του προγραμματιστή και απαλλάσσει τον αλγόριθμο από αρκετούς περιττούς ελέγχους.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Ένας For…Next βρόγχος είναι βρόγχος σταθερών επαναλήψεων. Βίαιη έξοδος από τον βρόγχο επιτυγχάνεται με την χρήση της Exit For.
2. Βρόγχοι μεταβλητών επαναλήψεων είναι οι While…Wend, Do While…Loop, Do Until…Loop και οι διαφορετικής σύνταξης Do…Loop While, Do…Loop Until. Βίαιη έξοδος από ένα βρόγχο Do…Loop επιτυγχάνεται με την χρήση της εντολής Exit Do.
3. Ο βρόγχος For…Next απαιτεί την ύπαρξη ενός «μετρητή» του οποίου η τιμή μεταβάλλεται σε κάθε επανάληψη κατά την εκτέλεση της εντολής Next, σύμφωνα με την τιμή που δίνουμε ως βήμα (Step).
4. Οι βρόγχοι While…Wend, Do While…Loop, Do Until…Loop μπορεί να εκτελούνται επ' άπειρο ή να μην εκτελεστούν ποτέ, ανάλογα με τη συνθήκη που περιμένουν να ικανοποιηθεί. Ωστόσο, οι Do…Loop While, Do…Loop Until θα εκτελεστούν σίγουρα τουλάχιστον μια φορά, διότι ο έλεγχος γίνεται στο τέλος του βρόγχου και όχι στην αρχή.
5. Αν ένας βρόγχος For…Next ο οποίος πρόκειται να εκτελεστεί m φορές, περιέχει έναν άλλο βρόγχο For…Next ο οποίος θα εκτελεστεί k φορές, οι συνολικές «εκτελέσεις» του κώδικα στον εσωτερικό βρόγχο είναι m * k.

ΕΡΩΤΗΣΕΙΣ

1. Με ποια διάταξη βρόγχου είναι καλύτερο να διαβάσουμε έναν πίνακα γνωστών διαστάσεων και να συγκρίνουμε κάθε στοιχείο με έναν προεπιλεγμένο αριθμό;
2. Με ποιόν βρόγχο θα μπορούσαμε να διαβάσουμε ένα αρχείο δεδομένων αγνώστου μεγέθους; Ποια συνθήκη ελέγχου θα προτείνατε;
3. Με ποια διάταξη βρόγχου θα μπορούσαμε να κατασκευάσουμε ένα πρόγραμμα το οποίο θα καταχωρίζει 10 τυχαίους αριθμούς σε έναν πίνακα, θα ζητάει έναν αριθμό από τον χρήστη, και εάν αυτός ο αριθμός δεν συμπεριλαμβάνεται στους 10 τυχαίους, αυτόματα θα ξαναδιάλεγε 10 νέους τυχαίους αριθμούς και θα ζητούσε νέο αριθμό από τον χρήστη;
4. Μπορούμε να χρησιμοποιήσουμε ένα βρόγχο For…Next για να εκτελέσουμε m επαναλήψεις με φθίνοντα μετρητή;
5. Πώς θα συντάσσαμε ένα πρόγραμμα ώστε να θέσει τυχαίες τιμές σε έναν πίνακα ο οποίος είναι δηλωμένος : Dim MyAr(-10 To 10) ;
6. Πώς θα συντάσσαμε ένα πρόγραμμα ώστε να επιτρέπει σε ένα χρήστη να εισάγει κωδικό εισόδου το πολύ 3 φορές;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα project στο οποίο θα επιλέγονται 10 τυχαίοι ακέραιοι αριθμοί στο διάστημα [20...40], με την προϋπόθεση ότι όλοι οι αριθμοί είναι διαφορετικοί μεταξύ τους. Εκτυπώστε τους αριθμούς.
2. Δημιουργήστε ένα project στο οποίο θα ορίσετε 2 πίνακες 10 στοιχείων ο καθένας, και θα τους δώσετε τυχαίες ακέραιες τιμές στο [0...25]. Έπειτα θα ελέγξετε κάθε στοιχείο του ενός με κάθε στοιχείο του άλλου. Αν κάποιο στοιχείο του πρώτου πίνακα ταυτίζεται με κάποιο στοιχείο του δεύτερου (ανεξαρτήτου θέσης) τότε θα εκτυπώσετε τον συγκεκριμένο αριθμό, καθώς και τη θέση που βρέθηκε τόσο στον πρώτο όσο και στον δεύτερο πίνακα. Η διαδικασία ολοκληρώνεται εφόσον ελεγχθούν πλήρως οι 2 πίνακες (μπορεί να βρεθούν περισσότερα του ενός κοινά στοιχεία).
3. Δημιουργήστε ένα project στο οποίο θα καταχωρίσετε 20 τυχαίες ακέραιες τιμές στο [0...20]. Έπειτα θα κάνετε έλεγχο για να δείτε πόσα 1, 2, 3,…,19,20 έχετε. Θα εκτυπώσετε στο τέλος συγκεντρωτική αναφορά.
4. Δημιουργήστε ένα project το οποίο θα κάνει πρόσθεση πινάκων A(m,n) + B(m,n) = C(m,n). Δώστε στους πίνακες A και B τυχαίες ακέραιες τιμές στο [1...10].
5. Δημιουργήστε ένα project το οποίο θα κάνει πολλαπλασιασμό πινάκων A(m,n) * B(n,k) = C(m,k). Δώστε στους πίνακες A και B τυχαίες ακέραιες τιμές στο [1...10].

Κεφάλαιο 8

Αρχεία σειριακής και άμεσης (τυχαίας) προσπέλασης

Τα αρχεία είναι η βασική μορφή αποθήκευσης πληροφορίας και δεδομένων στον Η/Υ. Γράφοντας ένα κείμενο π.χ. στο Microsoft Word έπειτα το αποθηκεύουμε για να μπορούμε οποιαδήποτε στιγμή επιθυμούμε, να το διορθώσουμε, να το συμπληρώσουμε, να το εκτυπώσουμε κτλ. Άρα η ψηφιακή αποθήκευση των δεδομένων είναι απαραίτητη.

Τα αρχεία μπορούμε να τα δημιουργήσουμε, να τα διαβάσουμε, να τα διαγράψουμε και να τα συμπληρώσουμε. Ο τρόπος εγγραφής και επεξεργασίας των αρχείων (προσπέλαση) ωστόσο δεν είναι μοναδικός. Έχουμε διαφορετικούς τύπους αρχείων (διαφορετικού χειρισμού) ανάλογα με το αποτέλεσμα που επιθυμούμε. Εμείς στην συγκεκριμένη ενότητα θα ασχοληθούμε με 2 τύπους αρχείων: Σειριακής (κατ' άλλους σειραϊκής / sequential) και Άμεσης (direct, κατ' άλλους τυχαίας / random) προσπέλασης.

Αρχεία σειριακής προσπέλασης (Σειριακά αρχεία / Sequential files):

Έτσι ονομάζονται τα αρχεία στα οποία η εγγραφή των δεδομένων και η ανάγνωση γίνονται με σειριακό τρόπο, δηλαδή πρώτα γράφουμε την πρώτη εγγραφή, μετά τη δεύτερη, έπειτα την τρίτη κτλ. Το ίδιο συμβαίνει και με την ανάγνωση. Πρώτα διαβάζουμε την πρώτη εγγραφή, έπειτα τη δεύτερη κτλ. Ακόμα και αν εμάς μας ενδιαφέρει η 30η εγγραφή, πρέπει πρώτα υποχρεωτικά να διαβάσουμε τις προηγούμενες 29. Από εκεί και πέρα όμως, αν δεν επιθυμούμε να πάρουμε από το αρχείο άλλη εγγραφή, μπορούμε να τερματίσουμε την επικοινωνία με το αρχείο, κλείνοντάς το.

Που χρειάζεται ένα σειριακό αρχείο;

Τα σειριακά σε αντίθεση με τα άμεσα, μπορούν εύκολα να διαβαστούν και να υποστούν επεξεργασία μέσω ενός απλού Text Editor, όπως λόγω χάρη το NotePad (Σημειωματάριο) των Windows. Επίσης, ο τρόπος ανάγνωσής τους είναι πολύ βολικός όταν μας ενδιαφέρει η ανάγνωση όλου του αρχείου. Έστω για παράδειγμα ότι έχουμε ένα αρχείο συντεταγμένων. Μπορούμε να το διαβάσουμε πολύ εύκολα παίρνοντας τα δεδομένα με τη σειρά.

Ας δούμε τις βασικότερες εντολές διαχείρισης ενός σειριακού αρχείου:

Open (Άνοιγμα αρχείου)

Σύνταξη: (1) Open TheFileName For Input As #n
(2) Open TheFileName For Output As #n
(3) Open TheFileName For Append As #n

Παράδειγμα:

```
Open "D:\MyProgs\Texts\MyText.txt" For Input As #1
```

Η μεταβλητή TheFileName μπορεί κάλλιστα να είναι δηλωμένη ως Variant ή ως String, και περιέχει το όνομα του αρχείου (με την πλήρη διαδρομή εύρεσής του στο δίσκο – full path name).

Η μεταβλητή "n" παίρνει την τιμή της στο [1...255] (ακέραιες τιμές) και προσδιορίζει μια χαρακτηριστική θέση κόμβου (κανάλι) την οποία θα χρησιμοποιήσει η Visual Basic για επικοινωνία με το αρχείο. Μόνο ένα αρχείο μπορεί να είναι ανοικτό σε κάθε θέση μια δεδομένη χρονική στιγμή.

For Input: Προσδιορίζει ότι το αρχείο θα ανοιχτεί για ανάγνωση και μόνο. Αν το αρχείο δεν υπάρχει, ή το κανάλι που επιλέξαμε είναι κατειλημμένο από άλλο αρχείο, η διαδικασία διακόπτεται με το σχετικό μήνυμα λάθους.
For Output: Προσδιορίζει ότι το αρχείο θα ανοιχτεί για εγγραφή και μόνο. Αν το αρχείο προϋπάρχει, διαγράφονται όλα τα περιεχόμενά του αυτομάτως. Αν

δεν υπάρχει, το δημιουργεί. Αν το κανάλι επικοινωνίας που επιλέξαμε είναι κατειλημμένο από άλλο αρχείο η διαδικασία διακόπτεται με το σχετικό μήνυμα λάθους.
For Append: Προσδιορίζει ότι το αρχείο θα ανοιχτεί για συμπλήρωμα εγγραφών και μόνο. Αν το αρχείο προϋπάρχει, τότε το ανοίγει και μεταφέρει τη σειρά εγγραφής στο τέλος του αρχείου. Τα δεδομένα δηλαδή που ήδη περιέχει το αρχείο παραμένουν (σε αντίθεση με ό,τι συμβαίνει με τη δήλωση For Output). Αν το αρχείο δεν υπάρχει, το δημιουργεί. Αν το κανάλι επικοινωνίας που επιλέξαμε είναι κατειλημμένο από άλλο αρχείο, η διαδικασία διακόπτεται με το σχετικό μήνυμα λάθους.

Πολλές φορές λόγω του ότι χρησιμοποιούμε πολλά αρχεία ταυτόχρονα, είναι δύσκολο να θυμόμαστε τα κανάλια στα οποία έχουμε ήδη ανοικτά αρχεία. Για να αποφύγουμε το λάθος που θα προκύψει από διπλή δήλωση του ίδιου καναλιού, χρησιμοποιούμε την πρόταση FreeFile. Η FreeFile μας προτείνει ένα τυχαίο κανάλι το οποίο είναι διαθέσιμο. Έτσι μπορούμε να ανοίξουμε ένα αρχείο (έστω για ανάγνωση) ως εξής:

```
Dim FF As Byte        '(ή ως Integer)
Dim FilNam As Variant
FilNam = "D:\MyProgs\Texts\MyText.txt"
FF = FreeFile
Open FilNam For Input As #FF
```

Πριν προχωρήσουμε σε περισσότερες εντολές που αφορούν τα αρχεία, ας δούμε μερικές προτάσεις ανοίγματος αρχείου.

Open "MyFile.txt" For Output As #1
Ανοίγει ή δημιουργεί το αρχείο "MyFile.txt" για εγγραφή στο κανάλι 1
Περίπτωση λάθους:

- Το κανάλι 1 χρησιμοποιείται από άλλο αρχείο το οποίο είναι ήδη ανοικτό.

Open "MyFile.txt" For Input As #1
Ανοίγει ή δημιουργεί το αρχείο "MyFile.txt" για ανάγνωση στο κανάλι 1
Περίπτωση λάθους:

- Το κανάλι 1 χρησιμοποιείται από άλλο αρχείο το οποίο είναι ήδη ανοικτό.
- Το αρχείο "MyFile.txt" δεν υπάρχει.

Open "MyFile.txt" For Append As #1
Ανοίγει ή δημιουργεί το αρχείο "MyFile.txt" για συμπλήρωση στο κανάλι 1

Περίπτωση λάθους:

- Το κανάλι 1 χρησιμοποιείται από άλλο αρχείο το οποίο είναι ήδη ανοικτό.

Τα αρχεία όπως βλέπουμε ανοίγουν με διαφορετικό τρόπο. Ο τρόπος κλεισίματος όμως της επικοινωνίας είναι κοινός για όλους τους τύπους των αρχείων.

Close (Κλείσιμο αρχείου)
Σύνταξη: (1) Close #n
(2) Close
Παράδειγμα:
Close #1

Παρατηρήσεις:

1. Αν χρησιμοποιήσουμε το δεύτερο τρόπο σύνταξης της Close, τότε αυτομάτως κλείνουν όλα τα αρχεία τα οποία είναι ανοικτά. Για να αποφύγουμε κάτι τέτοιο χρησιμοποιούμε τον πρώτο τρόπο σύνταξης όπου προσδιορίζουμε ποιο ακριβώς κανάλι επιθυμούμε να κλείσουμε.
2. Μετά το κλείσιμο κάποιου αρχείου, το αντίστοιχο κανάλι είναι και πάλι διαθέσιμο και μπορούμε να το χρησιμοποιήσουμε για ένα άλλο αρχείο.

Ας δούμε τώρα κάποιες εντολές εγγραφής και ανάγνωσης δεδομένων.
Print (Εγγραφή δεδομένων στο αρχείο)
Σύνταξη: Print #n, μεταβλητή1, μεταβλητή2 , ...
Παράδειγμα:
Print #1, "Αυτό το μήνυμα θα είναι μια εγγραφή στο αρχείο"
Print #1, "Γιώργος", MyAr(1,5), MyCounter, j, "Θεσσαλονίκη"

Για να μπορέσουμε να γράψουμε δεδομένα σε ένα αρχείο θα πρέπει οπωσδήποτε πρώτα να το έχουμε ανοίξει ως Output ή ως Append. Μπορούμε να γράψουμε οτιδήποτε σε ένα αρχείο. Η βασική σύνταξη είναι:
Print #n,
όπου [n] είναι το κανάλι που ανοίξαμε το αρχείο. Όλες οι λειτουργίες διαχείρισης ενός αρχείου αναφέρονται με βάση αυτό τον αριθμό [n].
Όταν επιθυμούμε να γράψουμε πολλά δεδομένα στην ίδια σειρά, χωρίζουμε τα δεδομένα με κόμμα « , ».

Παράδειγμα:

```
Open "MyFile.txt" For Output As #6
Print #6, "Πρώτη Εγγραφή"
Print #6, "Δεύτερη Εγγραφή"
Close #6
```

Input (Ανάγνωση δεδομένων από αρχείο)

Σύνταξη: Input #n, μεταβλητή1, μεταβλητή2, ...

Παράδειγμα:

```
Input #1, Rec1
Input #1, Rec1, MyVariable, TheDate
```

Για να μπορέσουμε να διαβάσουμε κάποια εγγραφή από ένα αρχείο, θα πρέπει προηγουμένως να το έχουμε ανοίξει ως Input. Επίσης, όπως έχουμε ήδη αναφέρει, δεν μπορούμε να διαβάσουμε μια εγγραφή, αν προηγουμένως δεν έχουμε διαβάσει όλες τις προηγούμενες.

Προσοχή!!!: Αν επιχειρήσουμε να διαβάσουμε περισσότερες εγγραφές από αυτές που περιέχει το αρχείο, η διαδικασία θα τερματιστεί βίαια με ένα μήνυμα λάθους «Input past end of file», δηλαδή «η ανάγνωση πέρασε το τέλος του αρχείου». Μεγάλη προσοχή στο πλήθος των εγγραφών !

Παράδειγμα ανάγνωσης του αρχείου που δημιουργήσαμε παραπάνω:

```
Open "MyFile.txt" For Input As #6
Input #6, vA1
Input #6, vA2
Close #6
```

Σημείωση:

Το MyFile.txt θα μπορούσαμε να το «ανοίξουμε» και σε άλλο κανάλι (διάφορο του «6»). Αυτό που έχει σημασία είναι το όνομα του συγκεκριμένου αρχείου να είναι σωστό, ενώ ο αριθμός του καναλιού να μην χρησιμοποιείται παράλληλα από άλλο αρχείο.

Στην πραγματικότητα κάθε εγγραφή που κάνουμε στο αρχείο είναι και μια διαφορετική οντότητα, ανεξαρτήτως αν γίνεται στην ίδια σειρά ή σε διαφορετική. Με άλλα λόγια, η παραπάνω ανάγνωση μπορεί να γίνει και ως εξής:

```
Input #6, vA1, vA2
```

Όταν κατά την εγγραφή διαχωρίζουμε τις καταχωρίσεις με κόμμα, τότε αυτές οι εγγραφές γράφονται στην ίδια σειρά, ανεξαρτήτως αν πρόκειται για διαφορετικά δεδομένα.

Υπάρχει όμως τρόπος να διαβάσουμε όλη την σειρά ενός αρχείου και να την εκχωρήσουμε σε μια μεταβλητή, ανεξάρτητα αν η εγγραφή του αρχείου ανά σειρά περιείχε περισσότερες μεταβλητές. Αυτό επιτυγχάνεται με χρήση της εντολής Line Input.

Line Input (Ανάγνωση ολόκληρης σειράς δεδομένων από το αρχείο)
Σύνταξη: Line Input #n, μεταβλητή1
Παράδειγμα:
Line Input #1, Rec1

Η Line Input δέχεται μόνο μια μεταβλητή για εκχώρηση όλων των περιεχομένων μιας σειράς ενός αρχείου.

Προηγουμένως, όταν εξετάζαμε την Input (κάτι που φυσικά ισχύει και για την Line Input), παρουσιάσαμε ένα πρόβλημα το οποίο είχε να κάνει με το πώς καταλαβαίνουμε ότι φτάσαμε στο τέλος του αρχείου. Αν δεν το καταλάβουμε και πάμε να συνεχίσουμε την ανάγνωση, τότε η διαδικασία θα τερματιστεί βίαια. Τώρα θα χρησιμοποιήσουμε έναν απαραίτητο τελεστή ελέγχου για ανάγνωση αρχείου με ασφάλεια.

EOF (End Of File – Τέλος αρχείου)
Σύνταξη: EOF(n)
όπου [n] το κανάλι που ανοίξαμε το αρχείο.
Η EOF δεν χρησιμοποιείται «σκέτη». Θα δούμε κάποιο παράδειγμα όπου θα καταλάβουμε τη χρήση του.

```
Private Sub Command1_Click()
Dim MyVar As Variant
Open "Demo.txt" For Input As #3

    While Not EOF(3)
    Line Input #3, MyVar
    Wend

Close #3
End Sub
```

Προσέξτε την σύνταξη της While. Αυτό πού κάνει είναι *«εφόσον δεν έχει φτάσει το τέλος του αρχείου που άνοιξε στο κανάλι 3»* να επιτρέπει την είσοδο στο βρόγχο και προφανώς να επιτρέπει την ανάγνωση από το αρχείο. Όταν φτάσει στο τέλος το αρχείο, δεν εκτελείται πλέον ο βρόγχος, συνεχίζει την εκτέλεση των εντολών και κλείνει το αρχείο.

Γνωρίζουμε ήδη πως δουλεύει η While…Wend. Αυτό που προσθέσαμε είναι ένας τελεστής «άρνησης ή αντίθεσης», τον τελεστή Not. Σημειώστε την παραπάνω σύνταξη της While [**While Not EOF(n) /…. / Wend**] γιατί θα την χρησιμοποιούμε πάντοτε αυτούσια κατά την ανάγνωση ενός σειριακού αρχείου.

Μια άλλη εντολή που επιτρέπει να εκχωρήσουμε το μέγεθος του αρχείου (σε bytes) σε μια μεταβλητή είναι η LOF(n).

LOF (Length Of File – Μέγεθος αρχείου)
Σύνταξη: LOF(n)
Παράδειγμα:

```
LengthOfMyFile = LOF(1)
```

Η παραπάνω πρόταση εκχωρεί το μέγεθος του αρχείου (σε Bytes) που ανοίξαμε στο κανάλι 1 στην μεταβλητή LengthOfMyFile.

Θεωρητικά, μάθαμε πλέον να ανοίγουμε κάποιο αρχείο, είτε για εγγραφή είτε για ανάγνωση, ή για συμπλήρωση. Στη συνέχεια θα ακολουθήσουν ορισμένα παραδείγματα με σχόλια.

Παραδείγματα δημιουργίας αρχείου (Output)

Παράδειγμα 1ο

```
Private Sub Command1_Click()
Dim Myar(20) As Single, i As Integer

        For i = 1 To 20
        Myar(i) = i * 5
        Next i

Open "demo1.txt" For Output As #1

        For i = 1 To 20
        Print #1, i, Myar(i)
        Next i

Close #1

End Sub
```

Δημιουργούμε έναν πίνακα (Myar) 20 στοιχείων, στα οποία δίνουμε τιμές (ανάλογες της θέσης στον πίνακα). Ανοίγουμε ένα αρχείο με όνομα «demo1.txt» για εγγραφή στο κανάλι 1. Σε ένα νέο βρόγχο αποθηκεύουμε στο αρχείο τη θέση κάθε στοιχείου καθώς και τις τιμές του πίνακα. Η αποθήκευση κάθε ζευγαριού γίνεται σε ξεχωριστή σειρά. Τέλος κλείνουμε το αρχείο.

Παράδειγμα 2ο

```
Private Sub Command1_Click()
Dim Pin(10, 10) As Single, i As Integer, j As Integer
For i = 1 To 10
    For j = 1 To 10
    Pin(i, j) = i * j * 3
    Next j
Next i

Open "demo2.txt" For Output As #15
For i = 1 To 10
    For j = 1 To 10
    Print #15, Pin(i, j),
    Next j
    Print #15, ""
Next i
Close #15
End Sub
```

Δημιουργούμε έναν δισδιάστατο πίνακα (10 χ 10) στον οποίο δίνουμε μέσα σε έναν διπλό βρόγχο For…Next τιμή σε κάθε στοιχείο. Έπειτα ανοίγουμε ένα αρχείο εγγραφής και με χρήση ξανά διπλού βρόγχου γράφουμε τα στοιχεία στο αρχείο. Προσέξτε όμως καλύτερα! Στην εντολή εγγραφής

```
Print #15, Pin(i, j),
```

έχουμε αφήσει ένα κόμμα « , » μετά την μεταβλητή. Αυτό το κάνουμε έτσι ώστε όλες οι υπόλοιπες εγγραφές της ίδιας σειράς του πίνακα να γραφτούν στην ίδια σειρά του αρχείου. Ο βρόγχος For j…Next ο οποίος είναι εσωτερικός του For i…Next αναφέρεται στην j στήλη της i σειράς. Για να αλλάξει σειρά εγγραφής στο αρχείο, αμέσως μετά την εγγραφή όλων των στηλών της κάθε σειράς, στέλνει μια κενή εγγραφή στο αρχείο, χωρίς κόμμα στο τέλος!

```
Print #15, ""
```

Έτσι αλλάζει αυτόματα σειρά στο αρχείο. Τα δεδομένα στο αρχείο είναι πλέον γραμμένα όπως φαίνεται παρακάτω.

3	6	9	12	15	18	21	24	27	30
6	12	18	24	30	36	42	48	54	60
9	18	27	36	45	54	63	72	81	90
12	24	36	48	60	72	84	96	108	120
15	30	45	60	75	90	105	120	135	150
18	36	54	72	90	108	126	144	162	180
21	42	63	84	105	126	147	168	189	210
24	48	72	96	120	144	168	192	216	240
27	54	81	108	135	162	189	216	243	270
30	60	90	120	150	180	210	240	270	300

Όταν τελειώσουν όλες οι εγγραφές, κλείνουμε το αρχείο.

Παράδειγμα 3ο

Στο παρακάτω παράδειγμα θα πάμε ένα βήμα παραπέρα. Έχουμε ήδη γνωρίσει το Microsoft Common Dialog Control (Εργαλείο κοινών διαλόγων) στο 3ο κεφάλαιο. Θα προσθέσουμε λοιπόν το παραπάνω εργαλείο στο ToolBox και έπειτα στην φόρμα μας. Θα το ονομάσουμε CD1. Όπως παραπάνω, προσθέτουμε και ένα κουμπί (CommandButton) του οποίου θα χρησιμοποιήσουμε το προεπιλεγμένο όνομα Command1.

```
Private Sub Command1_Click()
Dim MyFile As Variant, Pinakas(5, 5) As Single, i As Integer, j As Integer

With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowSave
End With

MyFile = CD1.FileName
Open MyFile For Output As #1

For i = 1 To 5
    For j = 1 To 5
    Pinakas(i, j) = i / j
    Print #1, Format(Pinakas(i, j), "0.0000"),
    Next j
    Print #1, ""
Next i

Close #1
End Sub
```

Περισσότερη δουλειά, λιγότερος κόπος! Χρησιμοποιήσαμε το Common Dialog (με τη μέθοδο ShowSave) για να προσδιορίσουμε τον τύπο (επέκταμα) των αρχείων και να προτείνουμε ένα αρχικό όνομα αρχείου. Στο παραπάνω παράδειγμα βλέπουμε ένα Block εντολών With…End With. Το Block αυτό χρησιμοποιείται προκειμένου να μην γράφουμε συνέχεια το αντικείμενο στο οποίο αναφερόμαστε, αλλά να το ορίσουμε μόνο μια φορά.

With Command1 .Default = True .SetFocus .Caption = "Press Me!" .Left = 0 .Top = 100 End With	Command1.Default = True Command1.SetFocus Command1.Caption = "Press Me!" Command1.Left = 0 Command1.Top = 100

Στον πίνακα επάνω φαίνεται ξανά η χρήση του. Οι εντολές που παρουσιάζονται στην πρώτη και στην δεύτερη στήλη είναι ισοδύναμες. Ωστόσο στην πρώτη στήλη δηλώνουμε μόνο μια φορά το αντικείμενο στο οποίο αναφερόμαστε και γλιτώνουμε περιττό κόπο και πληκτρολόγηση. Στο παράδειγμά μας χρησιμοποιήσαμε 2 βρόγχους For…Next λιγότερους από ότι στα προηγούμενα παραδείγματα και επιτύχαμε το ίδιο αποτέλεσμα. Αυτό έγινε διότι ακολουθήσαμε καλύτερη δόμηση, κάτι πολύ βασικό για τον προγραμματιστή. Τέλος, κατά την αποθήκευση των δεδομένων στο αρχείο, ζητήσαμε Format αριθμού με τέσσερα δεκαδικά. Ιδού το αποτέλεσμα:

```
1.0000          0.5000          0.3333          0.2500          0.2000
2.0000          1.0000          0.6667          0.5000          0.4000
3.0000          1.5000          1.0000          0.7500          0.6000
4.0000          2.0000          1.3333          1.0000          0.8000
5.0000          2.5000          1.6667          1.2500          1.0000
```

Από τώρα και στο εξής όταν επιθυμούμε να ανοίξουμε ή να δημιουργήσουμε αρχείο, θα χρησιμοποιούμε πάντοτε Common Dialog Control για να προσδιορίζουμε το όνομα και τη θέση του αρχείου στο δίσκο.

Παραδείγματα ανάγνωσης αρχείου (Input)

Παράδειγμα 1ο

Στο παράδειγμα αυτό θα ανοίξουμε το αρχείο που δημιουργήσαμε στο Παράδειγμα 3 της δημιουργίας αρχείων.

```
Dim MyFile As Variant, Pinak(50) As Single, i As Integer, j As Integer
With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
End With

MyFile = CD1.FileName
Open MyFile For Input As #1
j = 0
While Not EOF(1)
j = j +1
Input #1, Pinak(j)
Wend
Close #1
```

Επειδή επιθυμούμε να ανοίξουμε αρχείο και όχι να αποθηκεύσουμε, καλούμε τη μέθοδο ShowOpen του Common Dialog.

Στον παραπάνω κώδικα θα μπορούσαν να δημιουργηθούν προβλήματα διότι δηλώθηκε αυθαίρετα ένας πίνακας με 50 στοιχεία. Αν οι εγγραφές του αρχείου ήταν περισσότερες, τότε θα δημιουργούνταν κάποιο λάθος και θα διακόπτονταν η ομαλή εξέλιξη της εκτέλεσης του κώδικα.

Με τη χρήση του While Not EOF…Wend επετεύχθει η επιτυχής ανάγνωση του αρχείου χωρίς απρόσμενα προβλήματα.

Τα στοιχεία του πίνακα που πήραν τιμές είναι 25 (5 * 5), και αποθηκεύτηκαν ως στήλη. Για να τα αποθηκεύσουμε σε αντίστοιχο πίνακα με εκείνον που χρησιμοποιήσαμε στην εγγραφή, θα πρέπει να γνωρίζουμε εξ αρχής το format (τη διάταξη των δεδομένων) του αρχείου. Εδώ φυσικά τη γνωρίζαμε και έτσι θα μπορούσαμε να χρησιμοποιήσουμε πίνακα (5 χ 5), και βρόγχους For…Next :

```
Dim MyFile As Variant, Pinak(5,5) As Single, i As Integer, j As Integer
With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
End With

MyFile = CD1.FileName
Open MyFile For Input As #1

For i = 1 To 5
    For j = 1 To 5
    Input #1, Pinak(i, j)
    Next j
Next i

Close #1
```

Παράδειγμα 2ο

Θα διαβάσουμε και πάλι τα δεδομένα του παραπάνω παραδείγματος, αλλά αυτή τη φορά με χρήση της εντολής Line Input.

```
Dim MyFile As Variant, Pinak(50) As Single, i As Integer, j As Integer
With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
End With

MyFile = CD1.FileName
Open MyFile For Input As #1
j = 0

While Not EOF(1)
j = j +1
Line Input #1, Pinak(j)
Wend
Close #1
```

Εδώ το μόνο που αλλάζει είναι η μέθοδος ανάγνωσης. Το κάθε στοιχείο του πίνακα Pinak περιέχει όλες τις στήλες των δεδομένων ανά σειρά. Προφανώς μόλις 5 στοιχεία του πίνακα χρειάστηκαν για να διαβάσουμε όλο το αρχείο!

Παράδειγμα 3ο

Θα χρησιμοποιήσουμε πάλι το αρχείο που είχαμε δημιουργήσει παραπάνω, θα το διαβάσουμε ως αγνώστου Format αρχείο και θα αριθμήσουμε πόσα στοιχεία ανήκουν σε κάθε μια από τις παρακάτω κατηγορίες:
Από [0...1], (1...2], (2...3], (3...5]

```
Private Sub Command1_Click()

Dim ByCase(4) As Integer
Dim MyFile As Variant, Pinak(50) As Single, i As Integer, j As Integer

With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
End With

MyFile = CD1.FileName
Open MyFile For Input As #1

j = 0

While Not EOF(1)
j = j + 1
Input #1, Pinak(j)

    Select Case Pinak(j)

        Case 0 To 1
        ByCase(1) = ByCase(1) + 1
        Case 1 To 2
        ByCase(2) = ByCase(2) + 1
        Case 2 To 3
        ByCase(3) = ByCase(3) + 1
        Case 3 To 5
        ByCase(4) = ByCase(4) + 1
    End Select

Wend

Debug.Print ByCase(1), ByCase(2), ByCase(3), ByCase(4)

Close #1
End Sub
```

Το παράδειγμα αυτό είναι συνδυαστικό. Περιλαμβάνει μια έξυπνη χρήση της Select Case. Όμως, αν την παρατηρήσουμε καλύτερα, θα δούμε ότι υπάρχει κάτι περίεργο. Εμείς θέλουμε να δούμε πόσοι αριθμοί ανήκουν στα [0...1], (1...2], (2...3], (3...5]. Το «1» όμως πού ανήκει τελικά; (και το 2 και το 3); (αφού το 1 υπάρχει τόσο στην πρώτη δήλωση Case όσο και στην δεύτερη). Μήπως θα το μετρήσουμε 2 φορές; Όχι βέβαια! Έχουμε αναλύσει σχολαστικά την Select Case, και έχουμε πει ότι λειτουργεί με σειρά προτεραιότητας, και εκτελεί μόνο μια περίπτωση Case κάθε φορά. Άρα αφού θα εκτελέσει την πρώτη Case, η εκτέλεση θα μεταφερθεί στην End Select και θα συνεχιστεί κανονικά η εκτέλεση.

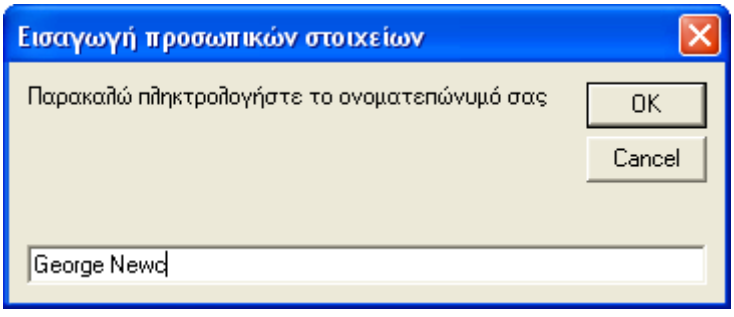
Παραδείγματα συμπλήρωσης αρχείου (Append)

Παράδειγμα 1ο

Θα ανοίξουμε ένα ήδη υπάρχον αρχείο, και θα συμπληρώσουμε την τρέχουσα ημερομηνία και ώρα, καθώς και το ονοματεπώνυμο του χρήστη.

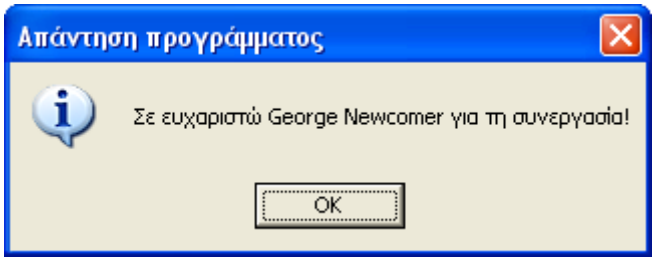
```
Private Sub Command1_Click()
Dim MyFile As Variant
Dim NameAndLastName As Variant

With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
End With

MyFile = CD1.FileName
Open MyFile For Append As #4

Do
NameAndLastName = InputBox("Παρακαλώ πληκτρολογήστε το ονοματεπώνυμό σας", _
"Εισαγωγή προσωπικών στοιχείων")
Loop Until NameAndLastName <> ""

Print #4, Now
Print #4, NameAndLastName
Close #4

MsgBox "Σε ευχαριστώ " & NameAndLastName & " για τη συνεργασία!",
vbInformation, "Απάντηση προγράμματος"

End Sub
```

Ρίξτε μια ματιά στον παραπάνω κώδικα. Χρησιμοποιείται ένας βρόγχος Do…Loop Until για να βεβαιωθεί ο χρήστης ότι έχει εισάγει ονοματεπώνυμο και δεν έχει αφήσει κανένα κενό πεδίο.



Προσέξτε επίσης και την κάτω παύλα « _ » (UnderScore). Η κάτω παύλα χρησιμοποιείται για την αναδίπλωση του κώδικα. Επειδή πολλές φορές κάποια εντολή που γράφουμε είναι πολύ μεγάλη, και εμείς θέλουμε να εμφανίζεται στην οθόνη ολόκληρη, χρησιμοποιούμε την « _ » και πατάμε [Enter]. Η Visual Basic καταλαβαίνει ότι η ίδια εντολή συνεχίζει στην κάτω σειρά. Σημαντικό είναι η κάτω παύλα να μην ακουμπάει πουθενά, δηλαδή να παρεμβάλουμε ένα κενό πριν την χρησιμοποίησή της.

Μετά την επιτυχή εισαγωγή δεδομένων από τον χρήστη, το πρόγραμμα αποθηκεύει στο αρχείο την τρέχουσα ημερομηνία και ώρα, και κλείνει την επικοινωνία με αυτό.
Ακολουθεί ένα πλαίσιο εκτύπωσης μηνύματος του τύπου:



όπου «George Newcomer» ήταν το ονοματεπώνυμο που έδωσε ο χρήστης. Τα σύμβολα « & » χρησιμοποιούνται στο MsgBox ως συνδέσεις μεταξύ μηνυμάτων, μεταβλητών και γενικώς οτιδήποτε επιθυμούμε να εμφανίσουμε ως ενιαίο.

Αυτά για την ώρα όσον αφορά τα σειριακά αρχεία. Ας περάσουμε τώρα να εξετάσουμε τα αρχεία άμεσης προσπέλασης (Random Access Files).

Αρχεία άμεσης προσπέλασης (Άμεσα αρχεία / Direct, τυχαίας προσπέλασης / Random access files):

Ονομάζονται τα αρχεία στα οποία η εγγραφή και ανάγνωση των δεδομένων γίνεται με «τυχαία προσπέλαση», δηλαδή για να διαβάσουμε για παράδειγμα την $5^η$ εγγραφή δεν χρειάζεται προηγουμένως να διαβάσουμε τις προηγούμενες 4 όπως συνέβαινε με τα σειριακά αρχεία.

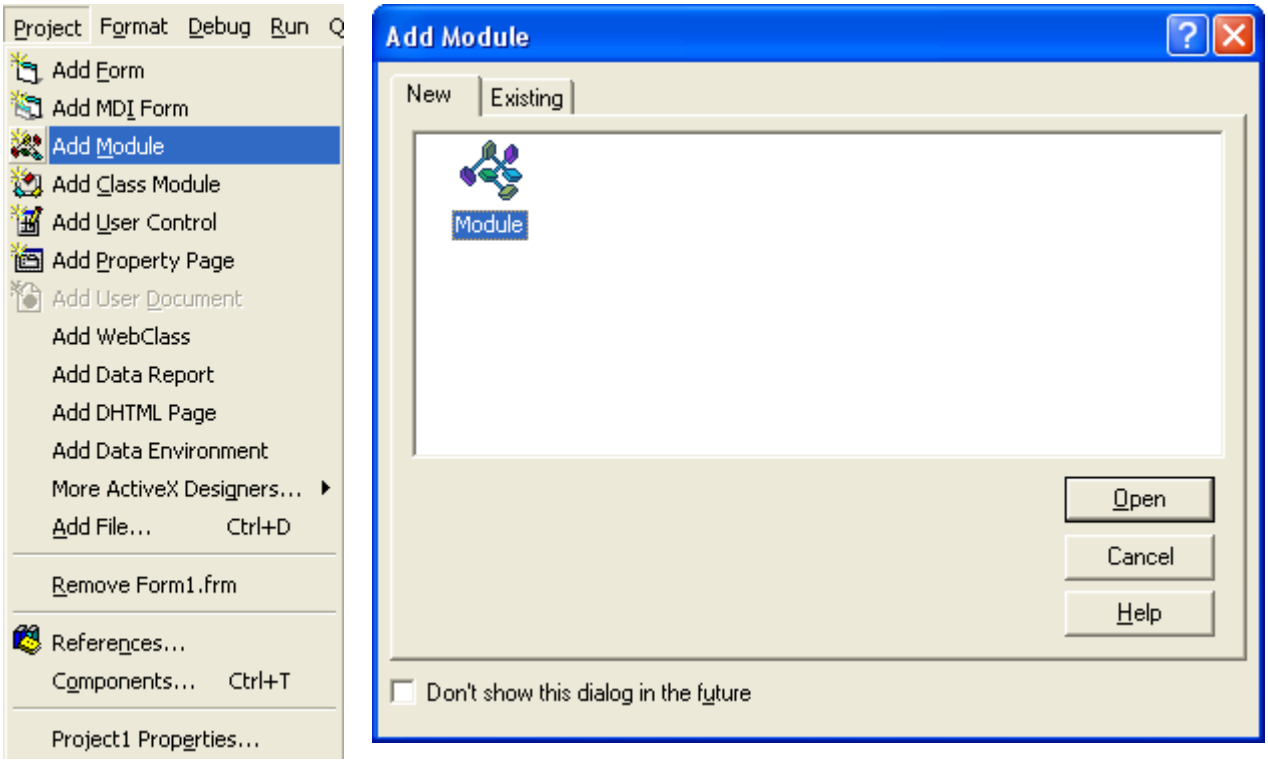
Υπάρχουν σημαντικές διαφορές μεταξύ των «τυχαίων» και των «σειριακών» αρχείων. Ας δούμε τις σημαντικότερες από αυτές:

1. Στα σειριακά αρχεία έχουμε διαφορετικό τρόπο ανοίγματος για εγγραφή, ανάγνωση και συμπλήρωση. Στα αρχεία τυχαίας προσπέλασης ο τρόπος είναι ένας. Το αρχείο ανοίγει και εμείς έπειτα μπορούμε να γράψουμε, να διαβάσουμε ή να συμπληρώσουμε εγγραφές στο αρχείο.
2. Η εγγραφή, ανάγνωση και γενικότερα η επεξεργασία των δεδομένων ενός αρχείου τυχαίας προσπέλασης, γίνεται με βάση κάποιο «κλειδί», το οποίο χαρακτηρίζει μοναδικά την κάθε εγγραφή. Είναι κάτι σαν το ΑΦΜ ή τον αριθμό ταυτότητάς μας.
3. Λόγω του ειδικού τρόπου εγγραφής των αρχείων τυχαίας προσπέλασης, είναι πολύ πιο γρήγορη και η ανάγνωση/επανεγγραφή τους σε σχέση με τα σειριακά.
4. Ένα σειριακό αρχείο μπορεί να ανοιχτεί ή να δημιουργηθεί από έναν απλό Text Editor. Αυτό δεν ισχύει για τα άμεσα αρχεία. Η κωδικοποίηση των αποθηκευμένων δεδομένων είναι «ακαταλαβίστικη» από τον μέσο χρήστη.
5. Στα άμεσα αρχεία δηλώνουμε εκ των προτέρων το μήκος της εγγραφής που θέλουμε να αναγνώσουμε. Άρα είναι αναγκαίο να γνωρίζουμε το Format του αρχείου για να μπορέσουμε να το επεξεργαστούμε.

Για να μπορέσουμε να διαχειριστούμε ένα άμεσο αρχείο, θα πρέπει όπως είπαμε να γνωρίζουμε ακριβώς το Format του. Επίσης θα πρέπει να δηλώσουμε το μήκος της εγγραφής και της ανάγνωσης. Πώς μπορούμε να επιτύχουμε τα παραπάνω με ασφάλεια;

Θα χρειαστεί να δούμε μια νέα δήλωση της Visual Basic, τη δήλωση Type. Προτού όμως την αναλύσουμε και δούμε τι μας προσφέρει, θα πρέπει να σημειώσουμε ότι η δήλωση της Type επιτρέπεται μόνο μέσα σε ένα Module (κάτι το οποίο είχαμε αναφέρει και για την δημόσια δήλωση ενός πίνακα στοιχείων) .

Ας πάμε λίγο στο περιβάλλον της Visual Basic για να δούμε τον τρόπο με τον οποίο μπορούμε να προσθέσουμε ένα Module στο πρόγραμμά μας:
Από το κεντρικό Menu το οποίο εμφανίζεται σαν μπάρα στην κορυφή της οθόνης μας, πηγαίνουμε στην κεντρική επιλογή (κατηγορία) [Project] και από εκεί επιλέγουμε την επιλογή [Add Module].



Βλέπουμε την επιλογή που κάνουμε από το Menu [Project] καθώς και το παράθυρο δεξιά που εμφανίζεται. Από το παράθυρο πατάμε του κουμπί με την λεζάντα «Open». Ως αποτέλεσμα ένα Module με αρχικό όνομα Module1 εισάγεται στο project μας, και εμφανίζεται (κενό) μπροστά μας.

Τι είναι όμως το Module;
Το Module είναι μια μονάδα για τη σύνταξη κώδικα. Δεν διαθέτει γραφικό περιβάλλον (όπως η φόρμα). Ο ρόλος του είναι πραγματικά πολύ σπουδαίος. Μέσα σε ένα Module μπορούμε να δηλώσουμε δημόσιες μεταβλητές και πίνακες (που θέλουμε να είναι διαθέσιμοι σε όλο το project μας), να γράψουμε υπορουτίνες και συναρτήσεις κτλ. Η ευελιξία του παραπάνω έγκειται στο γεγονός ότι ο κώδικας ή οι δηλώσεις (τα οποία χρησιμοποιούμε συχνά σε διάφορα projects), μπορούν να γραφτούν μόνο μια φορά και να αποθηκευτούν. Οποτεδήποτε τα χρειαστούμε, απλά τα ανοίγουμε και τα εισάγουμε στα προγράμματά μας. Δεν θα πούμε περισσότερα αυτή τη στιγμή, αλλά σιγά-σιγά θα αρχίσουμε να τα χρησιμοποιούμε και θα καταλάβουμε τη δύναμή τους.

Επιστρέφοντας στην περιγραφή μας, έχουμε εισάγει επιτυχώς ένα Module (με όνομα Module1). Αλλάξτε το όνομα του Module1 σε modDiloseis (ή άλλο όνομα τις επιλογής σας).

Θα δούμε τώρα αναλυτικά τι προσφέρει η Type:
Σύνταξη:
[Public] Type μεταβλητή
Μεταβλητή1 As τύπος
Μεταβλητή2 As τύπος
....
ΜεταβλητήΝ As τύπος
End Type

Με την Type δημιουργούμε ένα νέο «τύπο» μεταβλητής (ή κατηγορίας καλύτερα) όπως οι γνωστοί τύποι (Integer, Single, String κτλ). Στην κατηγορία που δημιουργούμε, προσθέτουμε κάποιες μεταβλητές τις οποίες θα διαχειρίζεται. Επειδή νομίζω ότι σας μπέρδεψα αρκετά δείτε το παρακάτω παράδειγμα:

```
Public Type Points
ID As Variant
X As Double
Y As Double
Z As Double
End Type
```

Δημιουργήσαμε στο modDiloseis μια «κατηγορία» με το όνομα Points. Αυτή η κατηγορία πλέον αναγνωρίζεται ως τύπος μεταβλητής. Στο συμβάν κλικ του Command6 δηλώνουμε μια μεταβλητή ως Points!

```
Private Sub Command6_Click()
Dim MyPoints As Points

MyPoints.
```


```
End Sub
```

Δηλώνουμε μια μεταβλητή, την MyPoints ως Points. Αυτομάτως, η μεταβλητή περιέχει τις υποκατηγορίες ID, X, Y, Z. Μόλις γράψουμε το όνομά της και πατήσουμε την τελεία « . », εμφανίζεται μια αναδιπλούμενη λίστα για να επιλέξουμε σε ποια από τα παραπάνω επιθυμούμε να εκχωρήσουμε τιμή! Άρα στην ουσία η MyPoints (η οποία δηλώθηκε ως Points) μπορεί να κρατήσει για κάθε υποκατηγορία μια τιμή!

Ας δούμε το παραπάνω παράδειγμα με πίνακα στοιχείων.

```
Private Sub Command6_Click()
Dim MyPointAr(50) As Points

For i = 1 To 50
Randomize Timer
MyPointAr(i).ID = i
MyPointAr(i).X = Rnd * 500
MyPointAr(i).Y = Rnd * 850
MyPointAr(i).Z = 150 + Rnd * 50
Next i

End Sub
```

Τι κάναμε στον παραπάνω κώδικα; Απλά δηλώσαμε έναν πίνακα στήλη ως Points, και έπειτα σε ένα βρόγχο For…Next δώσαμε

1. ως κωδικό σημείου τα 1,2,3,….,49,50
2. τυχαίες τιμές στα X, Y, Z κάθε σημείου

Η χρηστικότητα της Type είναι πλέον εμφανής. Ομαδοποιούμε πολύ εύκολα κάποια στοιχεία, και έτσι επιτυγχάνεται αμεσότερη πρόσβαση σε αυτά.

Σημείωση:
Μια που είδαμε το Module, να σας θυμίσουμε απλά ότι για να δηλώσετε έναν πίνακα μεταβλητών για δημόσια χρήση, αρκεί να γράψετε:
Public πίνακας(n,m…) As τύπος
στο Module.

Ας επιστρέψουμε και πάλι στα άμεσα αρχεία, μετά από την αρκετά χρήσιμη παρένθεση που κάναμε.
Αναλύσαμε την Type παραπάνω, διότι μιλούσαμε για το Format του αρχείου, καθώς και για το μήκος εγγραφής που θα πρέπει να δηλώσουμε.
Η γενική δήλωση ανοίγματος ενός τυχαίας προσπέλασης αρχείου είναι η εξής:

Open όνομα_αρχείου For Random As #1 Len = μήκος_εγγραφής
Όπου «όνομα_αρχείου» το όνομα του αρχείου που επιθυμούμε να ανοίξουμε και «μήκος_εγγραφής» το μήκος της εγγραφής που θα γράψουμε ή θα διαβάσουμε.
Οι εντολές που χρησιμοποιούμε για εγγραφή και για ανάγνωση είναι οι παρακάτω:
Εγγραφή: Put

Σύνταξη: Put #n, {κλειδί}, μεταβλητή
Όπου «n» είναι το κανάλι στο οποίο ανοίξαμε το αρχείο, {κλειδί} είναι το χαρακτηριστικό κλειδί αναζήτησης μιας εγγραφής. Αν το αρχείο αφορούσε καταχώριση στοιχείων ανθρώπων, τότε το κλειδί θα μπορούσε να είναι το ΑΦΜ. Προσοχή: Το κλειδί πρέπει να είναι μοναδικό σε κάθε αρχείο. **Αν όχι τότε γίνεται επικάλυψη των προηγούμενων δεδομένων από τα νέα.**

Ανάγνωση: Get
Σύνταξη: Get #n, {κλειδί}, μεταβλητή
Όπου «n» είναι το κανάλι στο οποίο ανοίξαμε το αρχείο, {κλειδί} είναι το χαρακτηριστικό κλειδί αναζήτησης μιας εγγραφής.

Ας δούμε το παρακάτω παράδειγμα εγγραφής ενός άμεσου αρχείου: (θα χρησιμοποιήσω την Type…End Type στο Module που έχουμε εισάγει)

```
Public Type Cords
PointID As Integer
X As Double
Y As Double
Z As Double
End Type
```

```
Private Sub Command6_Click()
Dim NewPnt As Cords, i As Integer
With CD1
.DialogTitle = "Άνοιγμα αρχείου τυχαίας προσπέλασης"
.Filter = "Όλα τα αρχεία *.* (*.*)|*.*| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.dat"
.ShowOpen
End With

MyFile = CD1.FileName
Open MyFile For Random As #1 Len = Len(NewPnt)

    For i = 1 To 50
    Randomize Timer
    NewPnt.PointID = i
    NewPnt.X = Rnd * 1000
    NewPnt.Y = Rnd * 1000 - 500
    NewPnt.Z = 245 + Rnd * 25
    Put #1, NewPnt.PointID, NewPnt
    Next i

Close #1
End Sub
```

Στο παράδειγμα που προηγήθηκε, δηλώθηκε μια μεταβλητή NewPnt με τύπο Cords (τον οποίο δημιουργήσαμε μέσα στο Module) και σε ένα βρόγχο For…Next δόθηκε ως κωδικός σημείου ο αύξων αριθμός « i » του μετρητή, καθώς και τυχαίες τιμές στα X, Y, Z της NewPnt. Αυτά καταχωρίστηκαν σε ένα άμεσο αρχείο, όπου δηλώθηκε ως μήκος εγγραφής τα bytes τα οποία καταλαμβάνουν όλες οι μεταβλητές που διαχειρίζεται η NewPnt. Αυτό ακριβώς είναι που μας προστατεύει από σφάλμα.

Στην καταχώριση των δεδομένων με την εντολή Put, δόθηκε ως κλειδί διαχείρισης ο αριθμός του κάθε σημείου (NewPnt.PointID), και καταχωρίστηκαν όλα τα στοιχεία της NewPnt. Όταν θα θελήσουμε να διαβάσουμε το αρχείο θα ζητήσουμε κάποια εγγραφή με βάση αυτό το κλειδί. Δείτε το παρακάτω παράδειγμα:

```
Private Sub Command7_Click()
Dim GivePoint As Cords, TheCode As Integer

With CD1
.DialogTitle = "Άνοιγμα αρχείου τυχαίας προσπέλασης"
.Filter = "Όλα τα αρχεία *.* (*.*)|*.*| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.dat"
.ShowOpen
End With

MyFile = CD1.FileName

Open MyFile For Random As #1 Len = Len(GivePoint)

TheCode = InputBox("Ποιά εγγραφή επιθυμείτε; [1...50]", _
"Εισαγωγή κωδικού")
Get #1, TheCode, GivePoint

MsgBox "X = " & GivePoint.X & " Y = " & GivePoint.Y & _
" Z = " & GivePoint.Z, vbInformation, "Ανάγνωση " & TheCode & "ης εγγραφής"

Close #1

End Sub
```

Ο χρήστης εισάγει τον κωδικό της εγγραφής που επιθυμεί να δει και τα περιεχόμενα εμφανίζονται μπροστά του με ένα MsgBox.

Ως τελική παρατήρηση θα αναφέρουμε ότι μπορούμε, καθώς προσθέτουμε εγγραφές σε ένα άμεσο αρχείο, να διαβάσουμε κάποιες άλλες ή καθώς διαβάζουμε εγγραφές να προσθέσουμε άλλες. Δεν απαιτείται κλείσιμο και άνοιγμα του αρχείου (όπως συνέβαινε με τα σειριακά).

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Τα σειριακά αρχεία έχουν διαφορετικό τρόπο ανοίγματος για εγγραφή, ανάγνωση ή συμπλήρωση. Τα τυχαίας προσπέλασης έχουν ένα τρόπο ανοίγματος, και δέχονται οποιαδήποτε ενέργεια.
2. Στα σειριακά αρχεία, η εγγραφή και η ανάγνωση γίνονται έτσι ώστε για να διαβάσουμε ή να γράψουμε την n-οστή εγγραφή, πρέπει πρώτα να έχουμε διαβάσει ή γράψει όλες τις προηγούμενες.
3. Το άνοιγμα ενός σειριακού αρχείου ως αρχείου εγγραφής διαγράφει τα περιεχόμενα που είχε αν προϋπήρχε. Στα αρχεία άμεσης προσπέλασης δεν συμβαίνει κάτι τέτοιο.
4. Τα σειριακά αρχεία μπορούμε να τα διαβάσουμε ακόμα και αν δεν γνωρίζουμε το format εγγραφής τους, κάτι που δεν γίνεται στα άμεσης προσπέλασης, όπου πρέπει να γνωρίζουμε τόσο το format εγγραφής όσο και το κλειδί αναφοράς κάθε εγγραφής.
5. Τα σειριακά γράφουν με την Print # και διαβάζουν με την Input # ενώ τα τυχαίας προσπέλασης γράφουν Put # με την και διαβάζουν με την Get #.
6. Η δήλωση κατηγορίας [Public] Type…End Type καθώς και η δημόσια δήλωση πινάκων γίνεται μέσα σε ένα Module.

ΕΡΩΤΗΣΕΙΣ

1. Ποιος τύπος αρχείου είναι προτιμότερος όταν επιθυμούμε να επεξεργαστούμε όλα τα δεδομένα ενός αρχείου;
2. Τι είναι το κλειδί στα αρχεία τυχαίας προσπέλασης;
3. Πώς βεβαιωνόμαστε ότι κάποιο κανάλι είναι διαθέσιμο για το άνοιγμα ενός αρχείου;
4. Πόσα αρχεία μπορούμε να έχουμε ανοιχτά ταυτόχρονα;
5. Ποιος τύπος αρχείου είναι γρηγορότερος;
6. Πώς γίνεται η προσπέλαση σε ένα σειριακό αρχείο όταν θέλουμε να συμπληρώσουμε εγγραφές;
7. Ποιος είναι ο ρόλος της δήλωσης Type…End Type;
8. Ποια η χρησιμότητα των Modules;
9. Τι σημαίνει «μήκος εγγραφής» σε ένα αρχείο τυχαίας προσπέλασης;
10. Μπορούμε να επέμβουμε σε ένα τυχαίας προσπέλασης αρχείο μέσω ενός Text Editor;
11. Πού χρησιμοποιείται η κάτω παύλα « _ » και που το « & »;
12. Τι ρόλο έχει η EOF(n) και τι η LOF(n);
13. Πότε χρησιμοποιούμε την Line Input # ; Σε τι είδους αρχείο;
14. Τι συμβαίνει αν γράψουμε Close σε ένα πρόγραμμα που διαχειρίζεται αρχεία;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα project στο οποίο θα αποθηκεύονται 30 τυχαίοι ακέραιοι αριθμοί στο [0...100] σε σειριακό αρχείο. Κλείστε το αρχείο. Δημιουργήστε ένα δεύτερο αρχείο, στο οποίο θα γράφονται οι αριθμοί του πρώτου αρχείου που είναι μικρότεροι του 50.
2. Δημιουργήστε project στο οποίο θα αποθηκεύονται 30 τυχαίοι ακέραιοι αριθμοί στο [0...100] σε σειριακό αρχείο. Κλείστε το αρχείο. Δημιουργήστε ένα δεύτερο σειριακό αρχείο στο οποίο, αφού ταξινομηθούν κατά φθίνουσα σειρά τα δεδομένα του πρώτου, θα γράφονται ταξινομημένα.
3. Αποθηκεύστε 40 τυχαίους ακέραιους αριθμούς στο [1...40] σε ένα σειριακό αρχείο, και 25 τυχαίους ακέραιους αριθμούς στο [1...25] σε άλλο σειριακό αρχείο. Κλείστε τα αρχεία. Σε τρίτο σειριακό αρχείο προσθέστε τόσο τις εγγραφές του πρώτου όσο και του δεύτερου. Συμπληρώστε στο τέλος άλλες 10 εγγραφές (τυχαίοι ακέραιοι στο [1...20].
4. Δημιουργήστε σειριακό αρχείο με 50 τυχαία σημεία τύπου (κωδικός, X, Y, Z). Κλείστε το αρχείο. Αντιγράψτε τα δεδομένα του σειριακού σε άμεσο με κλειδί τον κωδικό του σημείου. Διαβάστε από το άμεσο αρχείο τις 20 τελευταίες εγγραφές και εκτυπώστε τις.
5. Δημιουργήστε τύπο «Customer» με υποκατηγορίες τις «ID, Name, LastName, Tel». Επιτρέψτε στο χρήστη να εισάγει 5 καταχωρίσεις και αποθηκεύστε τις σε τυχαίας προσπέλασης. Έπειτα επιτρέψτε του να καλέσει από το αρχείο και να δεί σε διαφορετικά Label (στοιχεία ελέγχου) όποια καταχώριση θέλει βάσει του ID που θα εισάγει.
6. Δημιουργήστε ένα τυχαίας προσπέλασης αρχείο που θα περιέχει 100 τυχαίες εγγραφές με τύπο «Human» και υποκατηγορίες τις «ID, Age, Weight, Height, Sex». Κλείστε το αρχείο. Ανοίξτε ένα σειριακό αρχείο στο οποίο θα μεταφερθούν όλες οι εγγραφές του άμεσου, ταξινομημένες κατά ύψος.

Κεφάλαιο 9

Modules, υπορουτίνες (sub-routines), συναρτήσεις (functions)

Έχετε ήδη μάθει να προγραμματίζετε! Ξέρετε να κάνετε ελέγχους, να εκτελείτε επαναληπτικές διαδικασίες, να δηλώνετε μεταβλητές, να δημιουργείτε δικούς σας τύπους, να χρησιμοποιείτε πίνακες μεταβλητών, να δημιουργείτε αρχεία, να επεξεργάζεστε στοιχεία ελέγχου (ιδιότητες, μεθόδους, συμβάντα) και τόσα άλλα. Πρέπει όμως σιγά-σιγά να μάθετε να συντάσσετε τον κώδικά σας με στόχο να είναι άρτια «δομημένο» το πρόγραμμά σας.
Δείτε το παρακάτω τμήμα κώδικα:

```
With CD1
.DialogTitle = "Αποθήκευση αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
End With

MyFile = CD1.FileName
```

Άραγε, πόσες φορές γράψαμε και ξαναγράψαμε τον ίδιο κώδικα σε διάφορα σημεία του προγράμματός μας; Μεγάλο χάσιμο χρόνου! Τι θα ήταν σωστό να κάναμε; Σίγουρα να μπορούσαμε να γράψουμε τον κώδικα μια φορά μόνο, και να τον καλούσαμε όσες φορές θέλουμε. Αυτό θα ήταν και μεγάλη ευκολία, και τεράστια εξοικονόμηση χρόνου.

Στο κεφάλαιο αυτό θα δούμε πως μπορούμε να γλιτώσουμε από τέτοιους πλεονασμούς στους οποίους οδηγούμαστε με την επανάληψη της σύνταξης του ίδιου κώδικα σε διαφορετικά τμήματα ενός προγράμματος. Θα δούμε επίσης, τι μπορούμε να κάνουμε έτσι ώστε να γίνεται πιο εύκολα αναγνώσιμος και συντηρήσιμος ο κώδικάς μας.

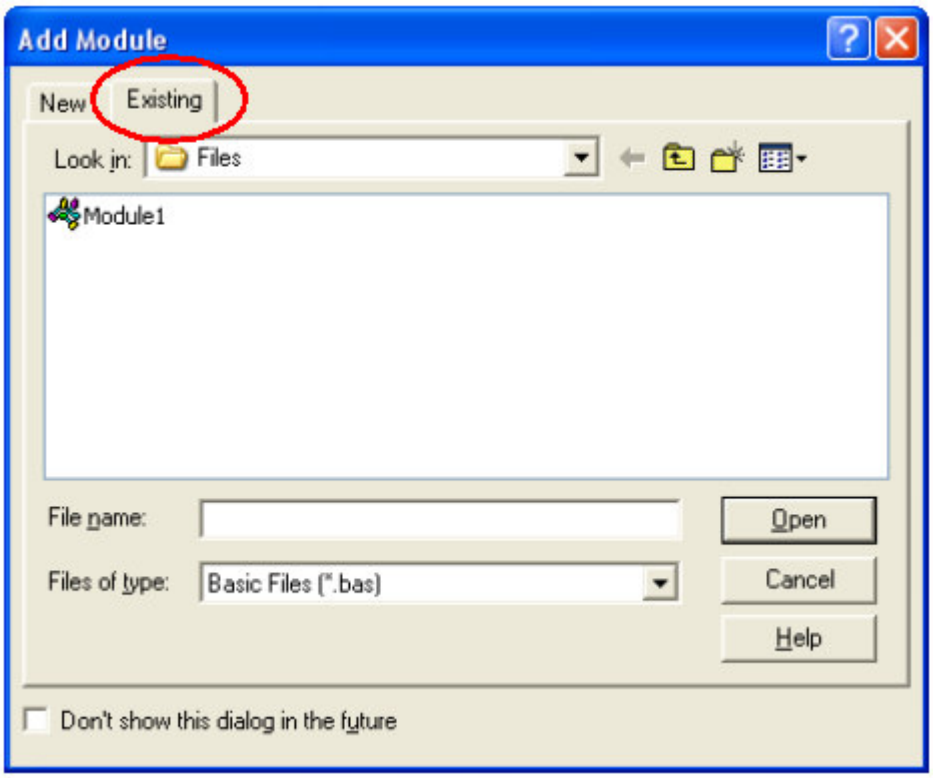
Υπορουτίνες (Sub-Routines)

Είναι τμήματα κώδικα τα οποία τα απομονώνουμε μέσα σε μια υπορουτίνα και τα οποία εκτελούν μια συγκεκριμένη διαδικασία την οποία χρειαζόμαστε συχνά.

Μια υπορουτίνα μπορεί να βρίσκεται στο τμήμα σύνταξης κώδικα της φόρμας ή μέσα σε ένα Module. Αν για παράδειγμα σκοπεύουμε να εκτελέσουμε το συγκεκριμένο τμήμα κώδικα μόνο μέσα από μια φόρμα, τότε μπορούμε να δηλώσουμε την υπορουτίνα ως «Ιδιωτική» και να την γράψουμε στο τμήμα σύνταξης κώδικα της φόρμας. Αν από την άλλη επιθυμούμε να έχουμε πρόσβαση στην συγκεκριμένη υπορουτίνα και από άλλες φόρμες, τότε είναι προτιμότερο να τη δηλώσουμε ως «Δημόσια» και να την εντάξουμε μέσα σε ένα Module.

Ωστόσο καμία από τις παραπάνω προτάσεις δεν είναι δεσμευτική. Μπορούμε δηλαδή να έχουμε και δημόσιες υπορουτίνες σε μια φόρμα, και ιδιωτικές σε ένα Module. Ποια θα πρέπει να προτιμήσουμε;

Είπαμε σε προηγούμενο κεφάλαιο ότι ο κώδικας που γράφουμε σε ένα Module, μπορεί εύκολα να προστεθεί και σε ένα διαφορετικό project. Απλά από το Menu [Project] >> [Add Module] επιλέγουμε [Existing]



και από εκεί βρίσκουμε ένα Module το οποίο είχαμε δημιουργήσει σε ένα άλλο project και το προσθέτουμε στο project που δουλεύουμε.

Έτσι δεν θα χρειαστεί να επαναλάβουμε τη σύνταξη μεγάλων τμημάτων κώδικα, τυπικών δηλώσεων τύπων και μεταβλητών κτλ.

Από εδώ και στο εξής θα θεωρούμε ότι τόσο τις υπορουτίνες όσο και τις συναρτήσεις τις γράφουμε σε Module και όχι στη φόρμα.

Τρόπος σύνταξης υπορουτίνας:

```
Private Sub DoThis()
'
End Sub
```

```
Public Sub DoThat()
'
End Sub
```

Η υπορουτίνα που ονομάσαμε DoThis είναι δηλωμένη ως ιδιωτική ενώ η DoThat ως δημόσια. Έτσι η DoThis μπορεί να κληθεί μόνο μέσα από το ίδιο το Module ενώ η DoThat από οποιοδήποτε σημείο του προγράμματος (από φόρμα, από άλλο Module κτλ).

Θα θεωρήσουμε ότι έχουμε ένα Module με το όνομα Module1, και θα γράψουμε μια υπορουτίνα δημόσιας κλήσης από όπου θα επιλέγουμε ένα αρχείο για άνοιγμα. Θα δηλώσουμε πρώτα μέσα στο Module1 την μεταβλητή MyFile ως δημόσια.

```
Public MyFile As Variant
```

```
Public Sub OpFile()

With Form1.CD1
.DialogTitle = "Άνοιγμα αρχείου"
.Filter = "Αρχεία κειμένου *.txt (*.txt)|*.txt| Αρχεία δεδομένων *.dat (*.dat)|*.dat"
.FileName = "Myfile.txt"
.ShowOpen
MyFile = .FileName
End With

End Sub
```

Προκειμένου να καλέσουμε το Common Dialog με το όνομα CD1, αναφερόμαστε πρώτα στην φόρμα που το έχουμε τοποθετήσει. Αφού το

Module δεν έχει γραφικό περιβάλλον, είναι λογικό να μην μπορούμε να προσθέσουμε στοιχεία ελέγχου σε αυτό.

Πάμε να δούμε ένα πρόγραμμα στο οποίο χρησιμοποιούσαμε τον παραπάνω κώδικα:

Πριν	Μετά
Private Sub Command6_Click() Dim NewPnt As Cords, i As Integer With CD1 .DialogTitle = "Άνοιγμα αρχείου τυχαίας προσπέλασης" .Filter = "Όλα τα αρχεία *.* (*.*)|*.*|Αρχεία δεδομένων *.dat (*.dat)|*.dat" .FileName = "Myfile.dat" .ShowOpen End With MyFile = CD1.FileName Open MyFile For Random As #1 Len = Len(NewPnt) For i = 1 To 50 Randomize Timer NewPnt.PointID = i NewPnt.X = Rnd * 1000 NewPnt.Y = Rnd * 1000 - 500 NewPnt.Z = 245 + Rnd * 25 Put #1, NewPnt.PointID, NewPnt Next i Close #1 End Sub	Private Sub Command6_Click() Dim NewPnt As Cords, i As Integer Module1.OpFile Open Module1.MyFile For Random As #1 Len = Len(NewPnt) For i = 1 To 50 Randomize Timer NewPnt.PointID = i NewPnt.X = Rnd * 1000 NewPnt.Y = Rnd * 1000 - 500 NewPnt.Z = 245 + Rnd * 25 Put #1, NewPnt.PointID, NewPnt Next i Close #1 End Sub

Δείτε πόσο ποιο δομημένος είναι ο κώδικας στη δεύτερη περίπτωση, στην οποία μάλιστα θα μπορούσαμε να επαναλάβουμε την κλήση της Module1.OpFile από οπουδήποτε και οσεσδήποτε φορές.

Αυτό που πρέπει να θυμάστε είναι ότι όταν καλούμε υπορουτίνα (μεταβλητή ή συνάρτηση κτλ) η οποία δεν ανήκει στην φόρμα ή στο Module από όπου την καλούμε, πρέπει πάντοτε να γράφουμε πρώτα το αντικείμενο το οποίο την περιέχει.

Έτσι στο πάνω παράδειγμα για να καλέσουμε την υπορουτίνα OpFile αναφερόμαστε στο Module1, και για να χρησιμοποιήσουμε την μεταβλητή

MyFile, αναφερόμαστε πάλι στο Module1 (Όπως αναφερόμαστε και στην φόρμα που περιέχει το Common Dialog προκειμένου να το χρησιμοποιήσουμε στο Module1).

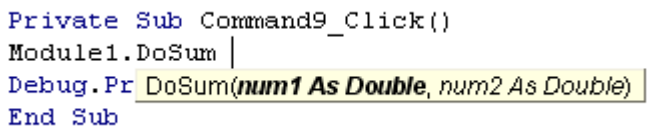
Ο παραπάνω τρόπος σύνταξης μιας υπορουτίνας είναι και ο πιο απλός. Η υπορουτίνα δέχεται ορίσματα τα οποία μεταφέρουν τιμές, για να τις χρησιμοποιήσουν σε κάποια διαδικασία. Θα συντάξουμε μια υπορουτίνα η οποία προσθέτει 2 αριθμούς. Επίσης θα δηλώσουμε και μια δημόσια μεταβλητή στο Module1.

```
Public Sum As Double
Public Sub DoSum(num1 As Double, num2 As Double)
Sum = num1 + num2
End Sub
```

Όλα τα παραπάνω είναι γραμμένα στο Module1. Δηλώνουμε μια δημόσια υπορουτίνα, και μέσα σε παρένθεση δηλώνουμε δυο τυπικές μεταβλητές στις οποίες περιμένουμε να εισάγει τιμές ο χρήστης όταν την καλέσει. Στο συμβάν κλικ του κουμπιού Command9 προσθέτουμε τον παρακάτω κώδικα:

```
Private Sub Command9_Click()
Module1.DoSum 23.346, 66.55
Debug.Print Sum
End Sub
```

Όπως βλέπετε παρακάτω, μόλις πάμε να καλέσουμε την υπορουτίνα εμφανίζεται μια κίτρινη λεζάντα βοηθείας η οποία μας ενημερώνει ότι πρέπει να εισάγουμε ως ορίσματα 2 αριθμούς (διπλής ακρίβειας).



Ωστόσο δεν συνηθίζεται να χρησιμοποιούμε υπορουτίνες για να πάρουμε άμεσα κάποιο αποτέλεσμα. Περισσότερο χρησιμοποιούνται για εκτέλεση διαδικασιών (όπως για παράδειγμα η ταξινόμηση ενός πίνακα στοιχείων).

Όταν όμως πραγματικά αυτό που μας ενδιαφέρει είναι ένα αποτέλεσμα το οποίο προκύπτει από δεδομένα που εισάγουμε, τότε χρησιμοποιούμε τη Συνάρτηση.

Συνάρτηση (Function)

Είναι κατανοητό σε όλους τι είναι μια συνάρτηση. Αν πάρουμε ως παράδειγμα την

$y = F(x) = x^2 - 6*x + 23$

καταλαβαίνουμε ότι πρέπει να θέσουμε μια τιμή στο « x » για να πάρουμε την τιμή του y. Ο τρόπος σύνταξης είναι παρόμοιος με τον τρόπο σύνταξης των υπορουτινών. Όσον αφορά τον τρόπο δήλωσης των συναρτήσεων (ιδιωτική ή δημόσια, στη φόρμα ή σε Module) ισχύει ό,τι και στις υπορουτίνες.

Μια μεγάλη όμως διαφορά είναι η εξής: Επειδή η Function μας επιστρέφει τιμή, αυτή πρέπει οπωσδήποτε να καταχωριστεί σε μια μεταβλητή (ή να χρησιμοποιηθεί απ' ευθείας σε κάποιο υπολογισμό). Αν καταχωριστεί σε μεταβλητή, ο τύπος της μεταβλητής προκαθορίζεται από τη δήλωση της συνάρτησης.

Πρέπει ακόμα να σημειωθεί ότι ενώ στις υπορουτίνες τα ορίσματα είναι προαιρετικά, στις συναρτήσεις είναι απαραίτητα.

Ας δημιουργήσουμε μια συνάρτηση η οποία θα πολλαπλασιάζει 3 αριθμούς: (Την συντάσσουμε στο Module1)

```
Public Function Multiply(num1 As Double, num2 As Double, num3 As Integer) _
As Double
Multiply = num1 * num2 * num3
End Function
```

Παρατηρούμε ότι το όνομα της συνάρτησης είναι και αυτό που «φέρει» το αποτέλεσμα των υπολογισμών. Ας την καλέσουμε από το συμβάν κλικ του κουμπιού Command10.

```
Private Sub Command10_Click()
Dim MyNum As Double
MyNum = Multiply(23.4545, 12.66, 3)
Debug.Print MyNum
End Sub
```

Προσέξτε ότι

Α) καταχωρίζουμε την τιμή της συνάρτησης στη μεταβλητή MyNum η οποία είναι δηλωμένη ως Double, όπως ακριβώς συνέβη κατά τη σύνταξη της συνάρτησης την οποία δηλώσαμε επίσης ως Double.

Β) Κατά την κλήση της δεν είναι υποχρεωτικό (εφόσον είναι δηλωμένη ως δημόσια συνάρτηση) να αναφέρουμε το όνομα του Module που τη φέρει. Ωστόσο αντί για:

```
MyNum = Multiply(23.4545, 12.66, 3)
```

Θα μπορούσαμε να γράψουμε

```
MyNum = Module1.Multiply(23.4545, 12.66, 3)
```

χωρίς καμία διαφορά.

Θα μπορούσαμε επίσης να χρησιμοποιήσουμε την τιμή της απ'ευθείας σε μια άλλη επεξεργασία:

```
Dim vResult As Double

VResult = 45 * Log(Multiply(12.11, 3.6543, 7))
```

Μια συνάρτηση μπορεί να είναι δηλωμένη ως τύπος μεταβλητής (κατηγορία) την οποία έχει δημιουργήσει ο χρήστης:

```
Public Type Cords2D
X As Double
Y As Double
End Type
```

```
Public Function AThem(BasePoint As Cords2D, Distance As Double, _
G_Rad As Double) As Cords2D
AThem.X = BasePoint.X + Distance * Sin(G_Rad)
AThem.Y = BasePoint.Y + Distance * Cos(G_Rad)
End Function
```

Η συνάρτηση παραπάνω υπολογίζει με τη χρήση του Α' Θεμελιώδους προβλήματος, τις συντεταγμένες ενός σημείου, από τις συντεταγμένες ενός άλλου από όπου μετρήθηκε η οριζόντια απόσταση και η γωνία διεύθυνσης. Θεωρούμε ότι η γωνία διεύθυνσης θα εισαχθεί σε ακτίνια (όχι σε βαθμούς).

Ας δούμε την κλήση της παραπάνω συνάρτησης από το συμβάν κλικ του Command11, με τυχαίες τιμές:

```
Private Sub Command11_Click()
Dim BaseP As Cords2D, PntFound As Cords2D
Dim Dis As Double, Gonia As Double

BaseP.X = 234.777
BaseP.Y = -129.016
Dis = 36.813
Gonia = 0.23455343

PntFound = AThem(BaseP, Dis, Gonia)

Debug.Print PntFound.X, PntFound.Y

End Sub
```

Η συνάρτηση θα υπολογίσει τις συντεταγμένες του νέου σημείου βάσει των τιμών που δώσαμε στα ορίσματα. Τέλος εκτυπώνουμε τα αποτελέσματα.

Μπορούμε να δημιουργήσουμε διάφορες συναρτήσεις σε ένα Module, να το ονομάσουμε λόγω χάρη modTopoFunctions και να το αποθηκεύσουμε. Αυτό το Module θα μπορούμε να το αξιοποιήσουμε σε οποιοδήποτε άλλο πρόγραμμα συντάξουμε, χωρίς να χρειάζεται να επαναλαμβάνουμε τα ίδια και τα ίδια.

Γενικά σε ένα Module θα πρέπει να:

1. δηλώνουμε τις μεταβλητές και τους πίνακες που θέλουμε να είναι δημόσιας χρήσης.
2. δημιουργούμε δικούς μας τύπους (κατηγορίες) μεταβλητών.
3. γράφουμε συναρτήσεις και υπορουτίνες που θα χρειαστεί να καλέσουμε από τα προγράμματά μας.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΘΥΜΑΜΑΙ;

1. Τα Modules θα πρέπει να χρησιμοποιούνται για δηλώσεις μεταβλητών, πινάκων, υπορουτίνων και συναρτήσεων.
2. Η συνάρτηση πάντα παίρνει ορίσματα και πάντοτε πρέπει να είναι δηλωμένος ο τύπος της (είτε δικός μας, είτε Integer, Double κτλ).
3. Η τιμή της συνάρτησης πρέπει να καταχωρίζεται σε μια μεταβλητή ίδιου τύπου ή να χρησιμοποιείται άμεσα.
4. Η υπορουτίνα θα πρέπει να χρησιμοποιείται για δόμηση του κυρίως προγράμματος. Κάθε υπορουτίνα θα πρέπει να εκτελεί μόνο μια ολοκληρωμένη διαδικασία, έτσι ώστε να είναι ευέλικτη η χρήση της.
5. Τα Modules δεν έχουν γραφικό περιβάλλον, είναι μόνο για συγγραφή κώδικα. Δεν μπορούμε να προσθέσουμε στοιχεία ελέγχου σε ένα Module.

ΕΡΩΤΗΣΕΙΣ

1. Έστω ότι έχω ένα πρόγραμμα το οποίο ανοίγει για ανάγνωση ένα αρχείο από το δίσκο, διαβάζει τα δεδομένα και τα ταξινομεί. Σε πόσες υπορουτίνες (και ποιες) θα μπορούσα να «σπάσω» τον παραπάνω κώδικα;
2. Ποια είναι τα πλεονεκτήματα ενός Module;
3. Αν δηλώσω μια συνάρτηση ως ιδιωτική σε ένα Module, από πού μπορώ να την καλέσω;
4. Ποιες διαφορές υπάρχουν μεταξύ της υπορουτίνας και της συνάρτησης;

ΘΕΜΑΤΑ ΔΙΕΡΕΥΝΗΣΗΣ

1. Δημιουργήστε ένα project, προσθέστε ένα νέο Module και συντάξτε με τον βέλτιστο τρόπο συναρτήσεις για την εύρεση του εμβαδού ενός τριγώνου, την εύρεση της απόστασης μεταξύ 2 σημείων και την εύρεση του εμβαδού ενός τετραπλεύρου.
2. Δημιουργήστε ένα project, προσθέστε ένα νέο Module και συντάξτε μία συνάρτηση που να επιλύει το δεύτερο θεμελιώδες. Καλέστε την από ένα κουμπί.
3. Δημιουργήστε ένα project, προσθέστε ένα νέο Module και συντάξτε μια συνάρτηση η οποία να δέχεται ως όρισμα πίνακα στήλη Ν στοιχείων και να βρίσκει τον μικρότερο και τον μεγαλύτερο αριθμό.
4. Δημιουργήστε ένα project, προσθέστε ένα νέο Module και συντάξτε μια υπορουτίνα η οποία θα δέχεται ως όρισμα τον τύπο του αρχείου (σειριακό/άμεσο), το κανάλι στο οποίο έχετε ήδη ανοίξει το αρχείο για εγγραφή και 5 μεταβλητές. Η υπορουτίνα θα αποθηκεύει τα δεδομένα στο αρχείο.

ΠΑΡΑΡΤΗΜΑ Α

ΕΙΣΑΓΩΓΗ ΣΤΗΝ VISUAL BASIC for APPLICATIONS

Η Visual Basic for Application (VBA) είναι μια εξειδικευμένη μορφή της Microsoft Visual Basic®. Δημιουργήθηκε με βάση την γνωστή Basic και χρησιμοποιεί πληθώρα ίδιων ή παρόμοιων χαρακτηριστικών και παρόμοιο σχεδιαστικό περιβάλλον.

Πρόκειται βασικά για μια «Light» έκδοση όσον αφορά τα χαρακτηριστικά λειτουργίας, τον τρόπο διαχείρισης, τις εντολές και τις συναρτήσεις. Σκοπός της δεν είναι η αντικατάσταση της Ms VB αλλά η δυνατότητα χρήσης της με σκοπό τον προγραμματισμό των λειτουργιών ενός λογισμικού ξενιστή. Η VBA δεν υπάρχει ως ξεχωριστό λογισμικό. Είναι όμως ενσωματωμένη σε άλλα προγράμματα όπως MsOffice, StarOffice, AutoCad, MicroStation κτλ. Η χρήση της συνίσταται στην αυτοματοποίηση κάποιων διαδικασιών (ή στην προσθήκη κάποιων νέων) τις οποίες, είτε δεν προέβλεψε ο κατασκευαστής, είτε πρόκειται για λειτουργίες που χρησιμεύουν σε μας, αλλά όχι και σε άλλους χρήστες. Είναι λοιπόν το εργαλείο εκείνο το οποίο θα μας βοηθήσει να προσαρμόσουμε το υπάρχον λογισμικό στις ανάγκες μας.

Ο προγραμματισμός της VBA διαφέρει από τον αντίστοιχο της Ms VB σε αρκετά σημεία. Για παράδειγμα τα εργαλεία που χρησιμοποιούνται στις περισσότερες περιπτώσεις έχουν παραπλήσια ονόματα αλλά όχι ίδια. Επίσης κάθε λογισμικό ξενιστής, πχ το Excel εισάγει τις δικές του εντολές και συναρτήσεις στη διάθεση του χρήστη. Με αυτό τον τρόπο δίνεται η δυνατότητα στο χρήστη να υπολογίσει μια συνάρτηση «τραβώντας» δεδομένα από ορισμένα κελιά του Excel.

1.Περιβάλλον VBA στο MsExcel

Για να εισέλθετε στο περιβάλλον της VBA από το Excel πατήστε Alt+F11 ή εναλλακτικά από το Menu [Tools] → [Macro] → [Visual Basic Editor]

Η οθόνη που εμφανίζεται μπροστά σας αποτελείται από τον Project Explorer ο οποίος σας εμφανίζει τα «φύλλα» τα οποία δουλεύετε καθώς και την εργασία σας, και το Properties Window το οποίο σας δείχνει τις ιδιότητες κάποιου επιλεγμένου αντικειμένου. Σε αυτή τη φάση δεν έχετε κάποιο δικό σας project σε λειτουργία.

Μπορείτε να ενεργοποιήσετε και το Immediate Window πατώντας Ctrl+G ή από το menu [View] → [Immediate Window].

Εισάγετε μια φόρμα: Από το menu [Insert] → [UserForm]
Μόλις το κάνετε αυτό τότε εμφανίζεται μια φόρμα με το όνομα UserForm1 καθώς και το ToolBox. Γιατί συμβαίνει αυτό; Στην VBA μπορείτε να δημιουργήσετε 3 πράγματα. Το πρώτο είναι ένα project με interface (buttons, textboxes κτλ). Το δεύτερο είναι ένα module. Αυτό είναι ένα τμήμα κώδικα το οποίο καλείται από ένα project. Δεν περιέχει interface και μπορεί να κληθεί από πολλά διαφορετικά προγράμματα. Το τρίτο είναι ένα Class module το οποίο είναι πάλι ένα τμήμα κώδικα που ενσωματώνει κατηγορίες με κοινά χαρακτηριστικά.
Οι πρώτες διαφορές που εντοπίζετε είναι το όνομα της φόρμας: Userform1 αντί για form1. Αυτό σημαίνει ότι εφόσον δεν αλλάξουμε το όνομα της φόρμας (προσοχή: ιδιότητα 'name' και όχι 'caption'), η «κλήση» της θα γίνεται ως Userform1. Με διπλό κλικ πάνω στην φόρμα μεταφερόμαστε στο παράθυρο κώδικα όπου βλέπουμε:

```
Private Sub UserForm_Click()

End Sub
```

Πάνω δεξιά υπάρχει ένα παράθυρο αναδιπλούμενης λίστας το οποίο γράφει Click. Αν εξετάσουμε τα υπόλοιπα συμβάντα θα δούμε ότι είναι παρόμοια ή ίδια με αυτά της Visual Basic.

Επιστρέψτε στο σχεδιαστικό περιβάλλον (project explorer → View object).

Προσθέστε στην φόρμα ένα πλαίσιο κειμένου, ένα κουμπί, ένα πλαίσιο επιλογής, ένα πλαίσιο λεζάντας και ένα κουμπί ραδιοφώνου (option button).

Θα παρατηρήσετε ότι υπάρχουν διαφοροποιήσεις όσον αφορά τα ονόματα αλλά όχι και όσον αφορά τα συμβάντα και τις ιδιότητες. Η κοινή αυτή αντιμετώπιση των συμβάντων και των διαδικασιών είναι το χαρακτηριστικό που δίνει την ευελιξία στον προγραμματιστή.

Παρακάτω θα παρατεθούν ορισμένες από τις διαφοροποιήσεις που απαιτούν προσοχή.

Εργαλεία	Microsoft Visual Basic	Visual Basic for Applications (VBA)
Πλαίσιο κειμένου	Text1	TextBox1
Λεζάντα	Label1	Label1
Κουμπί	Command1	CommandButton1
Κουμπί επιλογής	Check1	CheckBox1
Κουμπί ραδιοφώνου	Option1	OptionButton1

2.Παραδείγματα χρήσης VBA στο MsExcel & AutoCad

1. Επιλογή ενός κελιού που περιέχει αριθμητικό δεδομένο, πολλαπλασιασμό της τιμής του με μια σταθερά και εγγραφή της νέας τιμής σε επιλεγμένη θέση.

```
Private Sub CommandButton1_Click()
'Λεζάντα κουμπιού: Επιλογή δεδομένων

'Τα q1, q2 είναι η στήλη και η γραμμή που βρίσκεται το κελί μας.
q1 = InputBox("Στήλη", "Επιλογή στήλης δεδομένου", "A")
q2 = InputBox("Γραμμή", "Επιλογή γραμμής δεδομένου", "1")

'Δημιουργώ ένα string που περιέχει μια διεύθυνση κελιού π.χ. "A1"
CelName = q1 + q2

'Τραβάω την τιμή σε μια μεταβλητή CelV
CelV = Worksheets("Sheet1").Range(CelName).Value


'Αν το κελί είναι κενό τότε να μου επιστρέψει την τιμή "Blank cell"
If CelV = "" Then CelV = "Blank cell"

'Αναγραφή του αποτελέσματος στο Textbox1
TextBox1.Text = CelV

End Sub
```

```
Private Sub CommandButton2_Click()
'Λεζάντα κουμπιού: Πολλαπλασιασμός και εγγραφή

'Τα q1, q2 είναι η στήλη και η γραμμή που βρίσκεται το κελί στο οποίο θα αναγραφεί
το αποτέλεσμα
q1 = InputBox("Στήλη", "Επιλογή στήλης αναγραφής αποτελέσματος", "C")
q2 = InputBox("Γραμμή", "Επιλογή γραμμής αναγραφής αποτελέσματος", "1")

'Ζητείται η καταχώριση  του πολλαπλασιαστή
Mul = InputBox("Πολλαπλασιαστής", "Επιλογή πολλαπλασιαστή", "2")

'Δημιουργώ ένα string που περιέχει μια διεύθυνση κελιού π.χ. "C1"
CelName = q1 + q2

'Δίνω στο επιλεγμένο κελί την τιμή CelV * Mul
Worksheets("Sheet1").Range(CelName).Value = Val(TextBox1.Text) * Mul

End Sub
```

---

2. Ανίχνευση ενός πλήθους κελιών για ύπαρξη αλφαριθμητικής τιμής. Όπου η τιμή βρεθεί τα γράμματα γίνονται έντονα κόκκινα (bold & red)

```
Private Sub CommandButton1_Click()

'c Είναι μια μεταβλητή η οποία ουσιαστικά έχει τύπο κελιού
'Στη συγκεκριμένη περίπτωση η έρευνα γίνεται από το κελί A1 έως και το D10

For Each c In Worksheets("Sheet1").Range("A1:D10")

    'Αν η τιμή είναι αλφαριθμητική...
    If c.Value <> Val(c.Value) Then
    'το στυλ των γραμμάτων να γίνει έντονο
    c.Font.Bold = True
    'το χρώμα να γίνει κόκκινο
    c.Font.Color = vbRed
    End If

Next c

End Sub
```

3. Άνοιγμα ενός σειριακού αρχείου συντεταγμένων τύπου seq και δημιουργία πίνακα στο Excel. Ο τύπος του αρχείου είναι : Κωδικός, Χ , Υ , Η

```
Private Sub CommandButton1_Click()
'Λεζάντα : Επιλογή αρχείου

'Επιλογή αρχείου.
'Στο Textbox1 αναγράφεται η πλήρης διαδρομή του δίσκου και το όνομα του αρχείου
NamOfFile = TextBox1.Text

'Δημιουργείται μια επικεφαλίδα στη γραμμή Α και στις στήλες Α,Β,C,D με τα παρακάτω
' στοιχεία
Worksheets("Sheet1").Range("A1").Value = "Κωδικός"
Worksheets("Sheet1").Range("B1").Value = "X (m)"
Worksheets("Sheet1").Range("C1").Value = "Y (m)"
Worksheets("Sheet1").Range("D1").Value = "H (m)"

'Μετρητής ο οποίος κρατάει την τρέχουσα γραμμή αναγραφής του Excel
MyLine = 1

'Ανοιγμα σειριακού αρχείου στο 1ο buffer
Open NamOfFile For Input As #1

'Δήλωση : Το Loop να διαρκέσει τόσο όσο υπάρχουν δεδομένα στο αρχείο
While Not EOF(1)

'Ανάγνωση πεδίων κωδικού,Χ,Υ,Η
Input #1, COD, XX, YY, HH

'Πρόσθεση στο μετρητή. Αλλάζουμε σειρά εκτύπωσης
MyLine = MyLine + 1

'Δήλωση ακριβούς θέσης εγγραφής.
'Περιλαμβάνει την κωδικοποίηση της στήλης η οποία είναι
'σταθερή για κάθε τύπο δεδομένων, και την αρίθμηση της γραμμής
'η οποία αυξάνει προσθετικά.
POS1 = "A" & MyLine
POS2 = "B" & MyLineCreated by Information Technology Center (ΚΥΤΠ)
POS3 = "C" & MyLine
POS4 = "D" & MyLine

'Δήλωση εκτυπώσεων. Στην στήλη Α ο κωδικός, στην Β το Χ στην C το Υ και στην D '
το Η
Worksheets("Sheet1").Range(POS1).Value = COD
Worksheets("Sheet1").Range(POS2).Value = XX
Worksheets("Sheet1").Range(POS3).Value = YY
Worksheets("Sheet1").Range(POS4).Value = HH
```

```
'Κλείσιμο Loop - επανάληψη έως τέλους συνθήκης
Wend

'Κλείσιμο καναλιού ανάγνωσης (κλείσιμο αρχείου)
Close #1

End Sub
```

4. Δημιουργία ενός πλήθους τυχαίων σημείων και ραπορτάρισμα αυτών στο AutoCad. Θα μπορούσαμε όπως παραπάνω να καλέσουμε με τον ίδιο τρόπο ένα σειριακό αρχείο και να χρησιμοποιήσουμε τα δικά του δεδομένα.

```
Private Sub CommandButton1_Click()

'Δηλώσεις μεταβλητών οι οποίες θα περιέχουν τις συντεταγμένες των σημείων
Dim pcords(0 To 2) As Double
Dim tcords(0 To 2) As Double

'Δηλώσεις 2 μεταβλητών οι οποίες θα είναι οι μεταφορείς των δεδομένων
Dim apo As AcadPoint
Dim ate As AcadText

'Σταθερές δηλώσεις AutoCad ώστε να επιτρέπεται η είσοδος σε αυτό
    Dim activeDoc As AcadDocument
    Dim mspace As AcadModelSpace

'Δηλώσεις οι οποίες μας δείχνουν που θα επέμβουμε
'Συγκεκριμένα : Σε_αυτή_την_εργασία.Ως_πρόγραμμα.Στο_ενεργό_έγγραφο
'Σε_αυτή_την_εργασία.Στο_παράθυρο_σχεδίασης
    Set activeDoc = ThisDrawing.Application.ActiveDocument
    Set mspace = ThisDrawing.ModelSpace

'Δημιουργία 500 τυχαίων σημείων
For i = 1 To 500

Randomize Timer
'Ο δείκτης 0 αντιστοιχεί στο X
pcords(0) = Rnd * 1000 + 1000

Randomize Timer
'Ο δείκτης 1 αντιστοιχεί στο Y
pcords(1) = Rnd * 1000 + 1000

Randomize Timer
'Ο δείκτης 2 αντιστοιχεί στο Z
pcords(2) = Rnd * 100 + 100

'Ορίζω κάποιες άλλες "κοντινές" συντεταγμένες
'όπου θα εγγραφεί ο κωδικός σημείου
tcords(0) = pcords(0) + 0.2
tcords(1) = pcords(1) + 0.2
'Το Z να είναι 0.000μ
tcords(2) = 0

'Αποστολή σημείου στο παράθυρο σχεδίασης στις συντεταγμένες pcords
    Set apo = mspace.AddPoint(pcords)
```

```
'Αποστολή κειμένου (αρ.σημείου = i) στη θέση tcords με ύψος γραμμάτων = 0.9
    Set ate = mspace.AddText(i, tcords, 0.9)
'Zoom στα σημεία μου
ZoomAll
'Τέλος επανάληψης
Next i
End Sub
```

ΠΑΡΑΡΤΗΜΑ Β

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΣΚΗΣΕΩΝ

1° παράδειγμα: Χρήση δισδιάστατου πίνακα (n x m) και σειριακού αρχείου.

```
Private Sub Form_Click()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης κάνει "κλικ" πάνω στην φόρμα.
'Σ: Όχι σε κάποιο κουμπί ή σε κάποιο άλλο στοιχείο ελέγχου.

'Δήλωση μεταβλητών x,y,z ως μεταβλητές διπλής ακρίβειας
Dim x As Double, y As Double, z As Double

'Δήλωση πίνακα ar διαστάσεων 3 x 3 , του οποίου τα περιεχόμενα θα είναι μεταβλητές διπλής ακρίβειας
Dim ar(3, 3) As Double

'Άνοιγμα αρχείου με όνομα "inputfile1.txt" για ΕΙΣΑΓΩΓΗ δεδομένων στο "κανάλι" 1
'Σ: Αν το αρχείο δεν υπάρχει, το πρόγραμμα ανταποκρίνεται με σχετικό μήνυμα λάθους
Open "inputfile1.txt" For Input As #1

'Άνοιγμα αρχείου με όνομα "outfile1.txt" για ΕΞΑΓΩΓΗ δεδομένων στο "κανάλι" 2
'Σ: Αν το αρχείο δεν υπάρχει το δημιουργεί. Αν υπάρχει τότε ΤΟ ΔΙΑΓΡΑΦΕΙ και το ΞΑΝΑΔΗΜΙΟΥΡΓΕΙ
Open "outfile1.txt" For Output As #2

'Διάβασμα από το αρχείο που ανοίχτηκε στο κανάλι 1, τριών μεταβλητών
'Σ: Οι μεταβλητές πρέπει να είναι σαφώς διαχωρισμένες μεταξύ τους. Συνήθης διαχωρισμός είναι το "," ή η αλλαγή σειράς.
Input #1, x, y, z

'Εκτύπωση των ορισμάτων που διαβάσαμε πάνω στην φόρμα μας.
'Σ: Form1 είναι η τρέχουσα ιδιότητα "ονομα" της φόρμας μας. Το "Form1" είναι ένα αντικείμενο διεύθυνσης. Τέτοια αντικείμενα είναι ΟΛΑ τα ορατά
'εργαλεία (πλήκτρα, πλαίσια κειμένου κ.τ.λ) που χρησιμοποιεί η Visual Basic. Όλα αυτά τα αντικείμενα έχουν μεθόδους, συμβάντα και ιδιότητες.
'Συνεπώς το αντικείμενο Form1 καλεί την μέθοδο Print για να εκτυπώσει τα x,y,z πάνω στην φόρμα.
Form1.Print x, y, z
```

```
'Δίνω τις τιμές x,y,z στον πίνακα ar, και συγκεκριμένα στην στήλη 1 και στις γραμμές 1,2,3 αντίστοιχα
ar(1, 1) = x
ar(1, 2) = y
ar(1, 3) = z

'Εκτύπωση στην φόρμα Form1 των τιμών του πίνακα (όπως παραπάνω)
Form1.Print ar(1, 1), ar(1, 2), ar(1, 3)

'Εγγραφή στο αρχείο που είναι ανοιχτό ως αρχείο εξόδου στο κανάλι 2, των στοιχείων (1,1) (1,2) και (1,3) του πίνακα ar
Print #2, ar(1, 1), ar(1, 2), ar(1, 3)

'Εκτύπωση ενός κενού στην φόρμα. Ουσιαστικά αφήνουμε κενή σειρά στην εκτύπωση
Form1.Print " "

'Κλείνουμε τα αρχεία που είχαμε ανοίξει στα κανάλια 1,2
Close #1
Close #2

'Τέλος προγράμματος ---> Κατάσταση προγράμματος: ΑΝΑΜΟΝΗ
End Sub
```

Ανάλυση 1ου προγράμματος: Σημεία προσοχής

1. Form1 είναι το όνομα της φόρμας που χρησιμοποιούμε. Είναι αντικείμενο και έχει μεθόδους συμβάντα και ιδιότητες. Για παράδειγμα , μέθοδος της φόρμας είναι το Print που εκτυπώνει, το Move που μετακινεί τη φόρμα, το Cls που καθαρίζει την φόρμα από γραφικά κ.α. Συμβάντα είναι αυτά με τα οποία μπορούμε να ενεργοποιήσουμε μια λειτουργία στην φόρμα. Έτσι για παράδειγμα συμβάν είναι το Click , DblClick (διπλό κλικ) , Resize (αλλαγή μεγέθους) , Unload (κατάργηση) κ.α. Οι Ιδιότητες έχουν να κάνουν με χαρακτηριστικά που αφορούν πιο άμεσα την εμφάνιση και την συμπεριφορά της φόρμας. Ιδιότητες για παράδειγμα είναι η Caption (τίτλος) , Font (τύπος γραμματοσειράς) , ForeColor (χρώμα γραφής) , BackColor (χρώμα υπόβαθρου) , Left (αριστερή θέση στην οθόνη) , Top (πάνω θέση στην οθόνη) , Width (πλάτος) , Height(ύψος) κ.α.
2. Όταν καλούμε κάποιες μεθόδους μέσα από την ίδια την φόρμα μπορούμε να αντικαταστήσουμε το όνομά της με την κωδική ονομασία "Me" . Θα μπορούσαμε δηλαδή στο παραπάνω πρόγραμμα να γράψουμε Me.Print αντί για Form1.Print. Αυτό σημαίνει ότι αναφερόμαστε στην τοπική φόρμα.

3. Δεν είναι δυνατόν να χρησιμοποιήσουμε το ίδιο κανάλι για να ανοίξουμε περισσότερα από 1 αρχεία. Όταν δεν είμαστε σίγουροι για τον αριθμό καναλιού που είναι ελεύθερο τότε καλύτερα να χρησιμοποιούμε την εντολή FreeFile με την βοήθεια μιας μεταβλητής, έστω π.χ. της ff :

```
ff = FreeFile
Open "aaa.txt" For Input As ff
```

Το πλήθος των διαθέσιμων καναλιών διαφέρει με την σύνταξη του FreeFile. Με τον τρόπο που χρησιμοποιήθηκε εδώ γίνεται επιλογή καναλιού από το εύρος [1 – 255]

2ο παράδειγμα: Χρήση συναρτήσεων τετραγωνικής ρίζας (Sqr) , ημίτονου (Sin), συνημίτονου (Cos), και εφαπτομένης (Tan). Εφαρμογή των τριγωνομετρικών συναρτήσεων για τον υπολογισμό των ορισμάτων τους στη γωνία των 45ο. Δήλωση του «π» (=3,14159…)

```
Private Sub Form_Click()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης κάνει "κλικ" πάνω στην φόρμα.
'Σ: Όχι σε κάποιο κουμπί ή σε κάποιο άλλο στοιχείο ελέγχου.

'Δήλωση μεταβλητών p, sin45, cos45, tan45 ως μεταβλητές διπλής ακρίβειας
Dim p As Double
Dim sin45 As Double, cos45 As Double , tan45 As Double

'Υπολογισμός του π
p = 4# * Atn(1#)

'Εκτύπωση του μηνύματος "Greek pi" πάνω στην φόρμα μας.
Form2.Print "Greek pi"
'Με τον ίδιο τρόπο εκτύπωση της τιμής του π
Form2.Print p

'Εκτύπωση του μηνύματος "(Τετραγωνική ρίζα του 2) / 2 " πάνω στην φόρμα μας.
Form2.Print "(Τετραγωνική ρίζα του 2) / 2 "
'Εκτύπωση του αποτελέσματος του υπολογισμού
Form2.Print Sqr(2) / 2

'Εκτύπωση του μηνύματος "sin cos tan of 45 degrees" πάνω στην φόρμα μας.
Form2.Print "sin, cos, tan of 45 degrees"

'Υπολογισμός του ημίτονου, συνημίτονου και εφαπτομένης των 45 μοιρών
'Σ: Οι τριγωνομετρικές συναρτήσεις δέχονται ορίσματα σε rad (ακτίνια). Επειδή εμείς
' θέλουμε να υπολογίσουμε τα ορίσματα σε μοίρες,
'πρέπει να τα μετατρέψουμε. Για αυτό το λόγο πολλαπλασιάζουμε με το "π" και
' διαιρούμε με το 180.
sin45 = Sin(45 * p / 180)
cos45 = Cos(45 * p / 180)
tan45 = Tan(45 * p / 180)

'Εκτύπωση των αποτελεσμάτων πάνω στην φόρμα. Τα αποτελέσματα θα εμφανιστούν με
' μέγιστο πλήθος δεκαδικών, διαδοχικά το ένα
'δίπλα στο άλλο σε απόσταση ενός Tab
Form2.Print sin45, cos45, tan45

'Τέλος προγράμματος ---> Κατάσταση προγράμματος: ΑΝΑΜΟΝΗ
End Sub
```

Ανάλυση 2ου προγράμματος: Σημεία προσοχής

1. Η Visual Basic παρέχει κάποιες τριγωνομετρικές συναρτήσεις έτοιμες για χρήση. Πρέπει πάντοτε όμως να προσέχουμε το ΟΡΙΣΜΑ που περιμένουν μέσα στην παρένθεση για να εκτελέσουν την συνάρτηση. Εξ' ορισμού περιμένουν κάποιον αριθμό ή μεταβλητή τύπου Double, σε μονάδα γωνιών ακτίνια. Έτσι θα πρέπει να είμαστε πολύ προσεκτικοί στις μετατροπές των βαθμών και των μοιρών σε ακτίνια.
2. Ιδιαίτερη έμφαση πρέπει να δοθεί στην σειρά εκτέλεσης των αριθμητικών πράξεων. Πρώτα εκτελείται η διαίρεση και ο πολλαπλασιασμός και έπειτα η αφαίρεση και η πρόσθεση. Η ομαδοποίηση των πράξεων γίνεται με χρήση παρενθέσεων ( ). Προσοχή επίσης στα σύμβολα διαίρεσης. Το «/» εκτελεί μια κανονική διαίρεση μη αφήνοντας υπόλοιπο, ενώ το «\» εκτελεί μια ακέραια διαίρεση αφήνοντας υπόλοιπο. Έτσι 8/3 = 2.6666667 ενώ 8\3 = 2 .
3. Προσοχή στην δήλωση του «π». Η παραπάνω δήλωση είναι η σωστή διότι παρέχει μέγιστη ακρίβεια υπολογισμών.

3ο παράδειγμα: Χρήση στοιχείων Label και TextBox, καθώς και βρόγχο επανάληψης For…Next

```
Private Sub Form_Click()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης κάνει "κλικ" πάνω στην φόρμα.
'Σ: Όχι σε κάποιο κουμπί ή σε κάποιο άλλο στοιχείο ελέγχου.

        'Στην φόρμα μας έχουμε προσθέσει και τακτοποιήσει χωρικά ένα πλαίσιο
        ετικέτας (Label)
        'και δυο πλαίσια κειμένου (TextBox).
        'Το όνομα του στοιχείου ελέγχου Label είναι Label1 ενώ των TextBox είναι
        'Text1 και Text2 αντίστοιχα

'Δήλωση μεταβλητής p διπλής ακρίβειας
Dim p As Double

'Δίνω στην ιδιότητα Caption (= λεζάντα) του στοιχείου Label1 την συμβολοσειρά
'                                                              "greek p"
label1 = "greek p"

'Υπολογίζω κατά τα γνωστά την τιμή του "π"
p = 4 * Atn(1)

'Δίνω στην ιδιότητα Text (= κείμενο) του στοιχείου Text1 την τιμή του π
Text1 = p

'Δημιουργώ ένα βρόγχο 10 επαναλήψεων
For i = 1 To 10

'Εκτυπώνω πρώτα στην φόρμα μου και έπειτα στο Text2 την τιμή του "i" καθώς αυτό
' λόγω του βρόγχου παίρνει τιμές απο 1 εως 10
Form3.Print i
Text2 = i

'Κλείνω τον βρόγχο
Next i

'Τέλος προγράμματος ---> Κατάσταση προγράμματος: ΑΝΑΜΟΝΗ
End Sub
```

Ανάλυση 3ου προγράμματος: Σημεία προσοχής

1. Τα στοιχεία ελέγχου Label και Textbox έχουν όπως και οι φόρμες μεθόδους, συμβάντα και ιδιότητες. Ωστόσο το καθένα από αυτά έχει μια βασική ιδιότητα. Έτσι το Label ως βασική ιδιότητα έχει την Caption, για αυτό και μπορεί να παραληφθεί στην δήλωσή της. Οι παρακάτω γραμμές κώδικα έχουν ακριβώς το ίδιο αποτέλεσμα:

```
Label1 = "My name is Q"
Label1.Caption = "My name is Q"
```

Ωστόσο όταν γράφετε πολύπλοκο κώδικα είναι προτιμότερο να χρησιμοποιείτε τον δεύτερο τρόπο σύνταξης.

Ανάλογα και για τα TextBox. Βασική ιδιότητα ενός TextBox είναι η ιδιότητα Text. Μπορούμε δηλαδή να γράψουμε

```
Text1 = "Hi!"
Text1.Text = "Hi!"
```

Και να πάρουμε ακριβώς το ίδιο αποτέλεσμα. Για τους ίδιους λόγους προτιμότερο είναι να χρησιμοποιείτε τον δεύτερο τρόπο σύνταξης.

2. Η χρήση του βρόγχου For…Next [Step] χρησιμοποιείται όταν γνωρίζουμε εξ' αρχής το πλήθος των επαναλήψεων. Η γενική του σύνταξη είναι :

```
For μεταβλητή = (αρχικός αριθμός) To (τελικός αριθμός) [Step βήμα]
…
…
Next μεταβλητή
```

Παράδειγμα:

```
For i = 10 To 100 Step 10
…
…
Next i
```

Η πρόταση Step μπορεί να παραληφθεί και τότε ισούται με την μονάδα. Διαφορετικά δίνει το βήμα προσαύξησης του i. Στο παραπάνω

παράδειγμα το i παίρνει τις τιμές 10, 20, 30, … , 100 οπότε και τελειώνουν οι επαναλήψεις.

Παρόλο που οι επαναλήψεις όπως είπαμε είναι προκαθορισμένες, μπορούμε να διακόψουμε βίαια τον βρόγχο εισάγοντας (π.χ. κάτω από μια ορισμένη συνθήκη) την πρόταση Exit For.

4ο παράδειγμα: Χρήση βρόγχων Do/While, For/Next, If/Then/Else, InputBox και TextBox.

```
Private Sub Command1_Click()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης κάνει "κλικ" πάνω στο κουμπί Command1
'Σ: Δεν ενεργοποιείται αν γίνει κλικ σε κάποιο άλλο στοιχείο ελέγχου.

'Δήλωση δισδιάστατου πίνακα με ορίσματα διπλής ακρίβειας
Dim a(1, 3) As Double

'Δήλωση μεταβλητών ως Long (μεγάλος ακέραιος): Ο Long είναι σαν τον Integer με την διαφορά ότι διαθέτει πολύ πιο εκτεταμένο εύρος τιμών
Dim repetitions As Long, initial_value As Long

'Δήλωση μεταβλητής ως ακέραιας
Dim reply As Integer

'Θέτω αρχική τιμή στην μεταβλητή initial_value μηδέν
initial_value = 0

'Αρχή βρόγχου συνθήκης, αόριστων επαναλήψεων
'Σ: Ο βρόγχος που δημιουργείται από μια αντίστοιχη δήλωση μπορεί να εκτελείται επ'
' άπειρον ή ποτέ!!
'Ο μόνος τρόπος εξόδου από το βρόγχο είναι η μη περαιτέρω ικανοποίηση της
' περιγραφόμενης συνθήκης.
'Συγκεκριμένα ο παρακάτω βρόγχος θα εκτελείται ΓΙΑ ΟΣΟ Η ΜΕΤΑΒΛΗΤΗ
' initial_value < 10000
Do While (initial_value < 10000)

'Βρόγχος For/Next συγκεκριμένου αριθμού επαναλήψεων
For i = 1 To 3

        'Δίνω ορίσματα - τιμές στον πίνακα: Η στήλη (=1) είναι σταθερή. Το στοιχείο
        ' της 1ης γραμμής παίρνει την τιμή
        'initial_value +1 , της 2ης initial_value +2, και της 3ης initial_value +3
        a(1, i) = initial_value + i

' τέλος βρόγχου
Next i


'Εκτύπωση του αποτελέσματος στα TextBox1, TextBox2, TextBox3 αντίστοιχα
' (χρησιμοποιώντας και εδώ πρακτικά την ιδιότητα Text)
Text1 = a(1, 1)
Text2 = a(1, 2)
Text3 = a(1, 3)

'Συνάρτηση InputBox
```

```
'Πρόκειται για εσωτερική συνάρτηση η οποία περιμένει κάποια εισαγωγή.
'Η γενική σύνταξη είναι:
'Inputbox(Μήνυμα επικοινωνίας, Τίτλος , Προεπιλεγμένη τιμή)
'Στην προκειμένη περίπτωση το μήνυμα επικοινωνίας (το οποίο πρέπει να κατευθύνει
' τον χρήστη ως προς τον τύπο του δεδομένου που θα εισάγει) είναι το: "Να συνεχιστεί
' η διαδικασία; (1-Ναι 0-Οχι)"
'Ο τίτλος είναι το κείμενο που θα εμφανιστεί στην μπλε μπάρα των Windows πάνω από
' το InputBox
'Η προεπιλεγμένη τιμή (που εδώ είναι το "1") αποτελεί μια επιλογή του προγραμματιστή
' και συνήθως πρόκειται για την πιθανότερη απάντηση
reply = InputBox("Να συνεχιστεί η διαδικασία; (1-Ναι 0-Οχι)","Επιλογή","1")

''Έλεγχος για το αν θα συνεχιστεί η διαδικασία
If (reply = 1) Then             'Θετική απάντηση χρήστη


'Προσθέτουμε στην τιμή initial_value τον αριθμό 10
initial_value = initial_value + 10

'διαφορετικά
Else

'βίαιη εγκατάλειψη προγράμματος
Exit Sub

'τέλος ελέγχου για το εάν θα συνεχιστεί η διαδικασία
End If


'Κλείσιμο το Do While / Loop
Loop

'Τέλος προγράμματος ---> Κατάσταση προγράμματος: ΑΝΑΜΟΝΗ
End Sub
```

```
Private Sub Command2_Click()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης κάνει "κλικ" πάνω στο κουμπί Command2

'Τερματισμός προγράμματος - Επιστροφή στο σχεδιαστικό περιβάλλον της Visual Basic
End

'Δήλωση τέλους υπορουτίνας
End Sub
```

Ανάλυση 4ου προγράμματος: Σημεία προσοχής

1. Η χρήση του Do While / Loop βρόγχου ή αλλιώς While / Wend προτείνεται όταν ο αριθμός των επαναλήψεων εξαρτάται από κάποια συνθήκη. Στο τμήμα του προγράμματος όπου γίνεται ο έλεγχος με If..Then..Else..End If η πρόταση Exit Sub θα μπορούσε απλά να αντικατασταθεί ως εξής:

```
initial_value = 10000
```

Η πρόταση αυτή δίνοντας την τιμή 10000 στην μεταβλητή initial_value δεν περνάει τον έλεγχο του While και έτσι έχουμε έξοδο από την ρουτίνα.

2. Ο κώδικας του Do While / Loop θα μπορούσε επίσης να γραφτεί ως εξής:

```
While initial_value < 10000
...
...
Wend
```

Ο κώδικας του προγράμματος και ο παραπάνω είναι ίδιοι σε λειτουργία και θα μπορούσαν να χρησιμοποιηθούν εναλλακτικά.

3. Το If/Then//Else/End If ονομάζεται If Block. Χρησιμοποιείται για έλεγχο συνθηκών μέσα στο πρόγραμμα. Η γενική σύνταξη είναι:

```
If μια συνθήκη ισχύει Then     'τότε
...
'Εκτελεί αυτό το τμήμα του κώδικα
...
Else
...
'Εκτελεί αυτό το τμήμα του κώδικα
...
End If                          'Τέλος ελέγχου
```

Η συνθήκη μπορεί να είναι αριθμητική

If val1 <= 30 then

Λογική

If Answer = True then

ή μπορεί να αναφέρεται σε σύγκριση προτάσεων κ.τ.λ.

4. Η συνάρτηση InputBox όπως περιγράφηκε πιο πάνω, αντικατέστησε το απλό Input των παλαιότερων εκδόσεων της Basic με μια πιο ολοκληρωμένη συνάρτηση. Χρήσιμο αν όχι αναγκαίο να γίνεται έλεγχος των τιμών ή συμβολοσειρών που δίνει ο χρήστης προτού αυτή η τιμή χρησιμοποιηθεί από το πρόγραμμα.

5° παράδειγμα: Χρήση σειριακών αρχείων. Εφαρμογή στο 1° θεμελιώδες πρόβλημα της Τοπογραφίας.

```
Private Sub Form_Load()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης τρέξει το πρόγραμμα.
'Σ: Η ρουτίνα αυτή λόγω του ότι ανήκει στο συμβάν LOAD εκτελείται μόνο μια φορά
' κατά την εκκίνηση

'Δήλωση των παρακάτω μεταβλητών ως διπλής ακρίβειας
Dim x0 As Double, y0 As Double, s As Double, g As Double
Dim dx As Double, dy As Double

'Άνοιγμα αρχείου με όνομα "inputfile2.txt" για ΕΙΣΑΓΩΓΗ δεδομένων στο "κανάλι" 1
'Σ: Αν το αρχείο δεν υπάρχει, το πρόγραμμα ανταποκρίνεται με σχετικό μήνυμα λάθους
Open "inputfile2.txt" For Input As #1

'Άνοιγμα αρχείου με όνομα "outfile2.txt" για ΕΞΑΓΩΓΗ δεδομένων στο "κανάλι" 2
'Σ: Αν το αρχείο δεν υπάρχει το δημιουργεί. Αν υπάρχει τότε ΤΟ ΔΙΑΓΡΑΦΕΙ και το
' ΞΑΝΑΔΗΜΙΟΥΡΓΕΙ
Open "outfile2.txt" For Output As #2

'Υπολογισμός του π = 3,14159...
pi = 4 * Atn(1)

'Ανάγνωση από το αρχείο εισόδου (κανάλι 1) δυο εγγραφών, τα x0 και y0
Input #1, x0, y0

'Βρόγχος Do While / Loop (ή While/Wend )
'Χρησιμοποιεί τον παράγοντα EOF το οποίο είναι αρκτικόλεξο του END OF FILE.
'Η παρακάτω πρόταση δημιουργεί μια επανάληψη άγνωστων κύκλων με συνθήκη
' τερματισμού το τέλος των δεδομένων του αρχείου.
'Συνοπτικά θα λέγαμε ότι εκτελείται "Για όσο το αρχείο στο κανάλι 1 ΔΕΝ έχει
' τελειώσει".
'Συνεπώς ο βρόγχος καταλύεται όταν το πρόγραμμα "διαβάσει" όλες τις εγγραφές.
Do While Not EOF(1)

'Ανάγνωση από το αρχείο εισόδου δυο εγγραφών. Η τιμή τους εισάγεται στις
' μεταβλητές s,g
Input #1, s, g

'Υπολογισμός των dx,dy. Μαθηματική συνάρτηση χρησιμοποιώντας γνωστά ορίσματα
'Δ: Τα x0,y0,s,g,pi είναι γνωστά. Καμιά γλώσσα προγραμματισμού δεν μπορεί να
' επιλύσει μια εξίσωση με άγνωστο στην δεξιά πλευρά του "=" (όπως θα την λύναμε
' ίσως εμείς "στο χέρι"). Έστω ότι εμείς δεν είχαμε δώσει τιμές στα x0,y0.
' Αυτά θα θεωρούνταν αυτομάτως = 0. Δεν είναι δηλαδή δυνατόν για παράδειγμα να μας
' επιστρέψει τιμή του dx = x0 + 13.434
```

```
dx = x0 + s * Sin(g * pi / 200)

dy = y0 + s * Cos(g * pi / 200)

'Δίνω στα Text1 και Text2 (με χρήση της ιδιότητας Text) τις τιμές των dx,dy
Text1.Text = dx
Text2.Text = dy

'Εγγράφω τα dx,dy στο αρχείο εξόδου (κανάλι 2)
Print #2, dx, dy

'Επανάληψη βρόγχου
Loop


'Κλείσιμο αρχείων
Close #1
Close #2

'Τέλος υπορουτίνας. Αυτή η ρουτίνα δεν είναι πλέον σε αναμονή. Πρέπει μέσα από το
πρόγραμμα να την "ξεφορτώσουμε" και να την "ξαναφορτώσουμε" για να τρέξει ξανά.
End Sub
```

Ανάλυση 5ου προγράμματος: Σημεία προσοχής

1. Τα σειριακά αρχεία είναι αρχεία ascii και μπορούμε να τα ανοίξουμε από οποιονδήποτε Editor διαθέτουμε (π.χ. Notepad). Ονομάζονται σειριακά διότι δεν μπορούμε να διαβάσουμε την 5η εγγραφή για παράδειγμα αν προηγουμένως δεν έχουμε διαβάσει τις τέσσερις πρώτες. Η ανάγνωση του αρχείου γίνεται πάντα ξεκινώντας από την αρχή προς το τέλος.
2. Η πρόταση EOF όπως προαναφέρθηκε χρησιμοποιείται για να ελέγξουμε πότε έχουν τελειώσει οι εγγραφές μας. Έχει προφανώς νόημα μόνο κατά την ανάγνωση του αρχείου και όχι κατά την εγγραφή του. Παρόμοια χρησιμοποιείται η πρόταση LOF (Length Of File) η οποία μας παρέχει το μέγεθος των εγγραφών σε bytes. Αν LOF(κανάλι)= 0 τότε το αρχείο είναι κενό. Επίσης υπάρχει και η LOC η οποία ομολογουμένως δεν έχει ιδιαίτερη χρησιμότητα στα σειριακά αρχεία και μας δείχνει το σημείο όπου βρισκόμαστε κατά την ανάγνωση σε bytes. Πρέπει όμως να διαιρείται πάντα με το 128.
3. Γενική παρατήρηση: Όταν βλέπουμε μια παράσταση με μια ισότητα π.χ.

```
Pl1 = K2
Text1.Text = 57.983
K3 = Text2.Text
```

Πάντα ότι βρίσκεται αριστερά του «=» παίρνει τιμή από αυτό που βρίσκεται δεξιά. ΔΕΝ είναι αλγεβρική ισότητα:

```
Mul1 = Mul1 * 2
Sum = Sum + Cop1
```

Στην άλγεβρα οι παραπάνω προτάσεις προφανώς δεν ισχύουν. Στον προγραμματισμό το αριστερό σκέλος είναι ο καταχωριτής και το δεξιό η τιμή για καταχώριση.

6ο παράδειγμα: Χρήση αρχείων τυχαίας προσπέλασης (Random Access File). Παράλληλη χρήση της εντολής Type (ομαδοποίηση αναφορών μεταβλητής)

```
'-----------------------------------------------------
'-----------------------------------------------------
'Το παρακάτω τμήμα κώδικα δεν ανήκει σε κάποια ρουτίνα.
'Εισάγεται στο τμήμα General [Declarations] στο πάνω
'μέρος του παράθυρου Code. (Εκεί δηλαδή που γράφεται τον κώδικα)

'Δημιουργία τύπου μεταβλητής. Τον νέο αυτό τύπο τον ονομάζουμε "xyz" και του
' δίνουμε 3 νέα υποστοιχεία: τα x,y,z
'Τα υποστοιχεία αυτά τα δηλώνουμε ως μεταβλητές διπλής ακρίβειας
Private Type xyz
x As Double
y As Double
z As Double
End Type
'Τέλος δήλωσης τύπου


'Δήλωση μεταβλητής xyz1 ως xyz
'Σ: Ο xyz είναι ο νέος τύπος που χρησιμοποιούμε. Με την παρακάτω δήλωση δίνουμε
' στην μεταβλητή xyz1 τα χαρακτηριστικά της xyz
Private xyz1 As xyz
'-----------------------------------------------------
'-----------------------------------------------------


Private Sub Command1_Click()
'Αυτή η ρουτίνα καλείται όταν ο χρήστης κάνει "κλικ" πάνω στο κουμπί Command1

'Δήλωση μεταβλητής irec ως ακέραια
Dim irec As Integer

'Δίνουμε στην μεταβλητή length το συνολικό μέγεθος της μεταβλητής xyz1
'χρησιμοποιώντας την συνάρτηση Len.
'Η συνάρτηση Len μας δίνει το μέγεθος της μεταβλητής που δέχεται ως όρισμα σε
'Bytes.
'Στην προκειμένη περίπτωση επειδή η xyz1 περιλαμβάνει 3 υπομεταβλητές, τις x,y,z
'και η κάθε μία από αυτές είναι δηλωμένη ως
'μεταβλητή διπλής ακρίβειας, τότε συνολικά το "μήκος" (= μέγεθος) της xyz1
' θα είναι 3 * 8 = 24 bytes (διότι 8 bytes προκαταλαμβάνει
'κάθε μεταβλητή που είναι δηλωμένη ως double.
Length = Len(xyz1)


'Άνοιγμα αρχείου με όνομα "randomfile7" για χρήση του ως αρχείο τυχαίας
```

```
' προσπέλασης στο κανάλι 1 με μήκος (ή μέγεθος) εγγραφής  24 (bytes)
Open "randomfile7" For Random As #1 Len = length

'Δημιουργία βρόγχου προκαθορισμένων επαναλήψεων ( = 100)
For k = 1 To 100

'Δίνω τις τιμές k+1, k+2 , k+3 με την παρακάτω σειρά στις υπομεταβλητές x,y,z της
' xyz1
xyz1.x = k + 1
xyz1.y = k + 2
xyz1.z = k + 3

'Γράφω τα στοιχεία στον αρχείο. Η μεταβλητή k στο For/Next χρησιμοποιείται ως
' δείκτης εγγραφής
'Με χρήση αυτού του δείκτη θα μπορέσουμε στην συνέχεια να "καλέσουμε" κάποια
' εγγραφή
Put #1, k, xyz1

'Τέλος βρόγχου
Next k

'Κλείνουμε το αρχείο
Close #1

'Ανοίγουμε ξανά το αρχείο όπως προηγουμένως με τη διαφορά ότι το ανοίγουμε στο
' κανάλι 11
Open "randomfile7" For Random As #11 Len = length

'Ζητάμε από τον χρήση με την βοήθεια της συνάρτησης InputBox να μας δώσει τον
'αριθμό εγγραφής από τον οποίο θα αναγνώσουμε τις εγγραφές.
'Το "irec" δηλαδή είναι το "k" που χρησιμοποιήσαμε κατά την διάρκεια της δημιουργίας
'του αρχείου.
'Υπάρχει πλήρης αντιστοιχία των εγγραφών που δημιουργούμε με βάση κάποιο κλειδί με
'τις εγγραφές που διαβάζουμε με χρήση του ίδιου κλειδιού.
'Σ: Κλειδί είναι η επίσημη ονομασία του δείκτη
irec = InputBox("Record number ")

'Ανάγνωση της μεταβλητής xyz1 και των υποστοιχείων της που αντιστοιχούν στο κλειδί
irec
Get #11, irec, xyz1

'Εκτύπωση των αποτελεσμάτων στα Text1 , Text2 , Text3 (που υπάρχουν στην φόρμα
μας) με χρήση της βασικής τους ιδιότητας Text       (π.χ. Text1.Text = xyz1.x) η
οποία εδώ παραλείπεται.
Text1 = xyz1.x
Text2 = xyz1.y
Text3 = xyz1.z

'Κλείνουμε το αρχείο
```

```
Close #11

'Τέλος εκτέλεσης κώδικα - Κατάσταση ΑΝΑΜΟΝΗ
End Sub
```

Ανάλυση 6ου προγράμματος: Σημεία προσοχής

1. Τα αρχεία τυχαίας προσπέλασης διαφέρουν σε αρκετά σημεία από τα σειριακά αρχεία. Συγκεκριμένα:
   a. Δεν είναι επεξεργάσιμα ή δημιουργήσιμα από κάποιον editor όπως γίνεται με τα σειριακά.
   b. Οι εγγραφές κατανέμονται με τυχαίο τρόπο μέσα στο αρχείο και δεν εγγράφονται η μια διαδοχικά με την προηγούμενη.
   c. Η ανάκτηση των δεδομένων γίνεται με χρήση κάποιου αριθμητικού κλειδιού.
   d. Τόσο η εγγραφή όσο και η ανάγνωση των αρχείων άμεσης προσπέλασης γίνονται με κοινή εντολή προσπέλασης. Η σύνταξή της είναι:

Open όνομα αρχείου For Random As κανάλι Len = μήκος κάθε εγγραφής

Π.χ.:

Open "DATA1.dat" For Random As #3 Len = 36

Πρακτικά το αρχείο που ανοίξαμε είναι έτοιμο να δεχτεί και να δώσει δεδομένα.

   e. Οι εντολές των σειριακών αρχείων Input και Print έχουν αντικατασταθεί με τις Get και Put αντίστοιχα. Η διαφορά όμως έγκειται στο γεγονός ότι αν αποθηκευτεί κάτι χωρίς σχετικό κλειδί δεν θα είναι δυνατή η ανάκτησή του. Η σύνταξη των Put και Get είναι η εξής:

Put κανάλι , κλειδί , σύνολο μεταβλητών
Get κανάλι , κλειδί , σύνολο μεταβλητών

Π.χ.:

Put #12, iKey , X , Y

Get #12, iKey , X , Y

Προφανώς η τιμή του κλειδιού (εδώ iKey) δίνεται από τον χρήστη ως κριτήριο αναζήτησης.

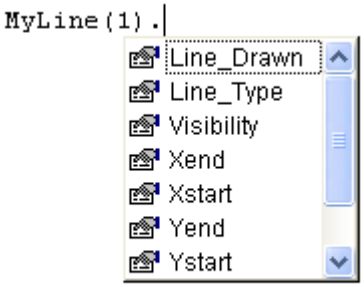
2. Η πρόταση Type/End Type μας δίνει την δυνατότητα να ορίσουμε εμείς ένα νέο τύπο μεταβλητών. Είναι πολύ χρήσιμη στην συλλογική διαχείριση δεδομένων. Με την δημιουργία κάποιου νέου τύπου, ορίζουμε υπομεταβλητές – χαρακτηριστικά μιας μεταβλητής. Για παράδειγμα:

```
Private Type New_Line

Xstart As Double
Ystart As Double
Zstart As Double
Xend As Double
Yend As Double
Zend As Double
Line_Type As Variant
Visibility As Boolean
Line_Drawn As Boolean

End Type
```

Δημιουργήσαμε έναν τύπο μεταβλητής στον οποίο δώσαμε κάποιες μεταβλητές - χαρακτηριστικά. Έτσι πετύχαμε άμεση παραπομπή στο αντικείμενό μας. Για παράδειγμα ο παραπάνω κώδικας αναφέρεται σε ένα αντικείμενο το οποίο ονομάσαμε «Νέα γραμμή». Έστω ότι γράφουμε κάποιο υποτυπώδες σχεδιαστικό πρόγραμμα, και θέλουμε για κάθε γραμμή που σχεδιάζουμε να γνωρίζουμε τα χαρακτηριστικά της.

Παράδειγμα:

```
Dim MyLine(3000) As New_Line          'Δημιουργούμε ένα πίνακα στήλη με
'3000 στοιχεία ως τύπος New_Line
'Θα δώσουμε στην 1η γραμμή κάποια χαρακτηριστικά:
```



Παρατηρούμε ότι με το που αναφερόμαστε στην μεταβλητή MyLine την οποία ορίσαμε ως τύπο New_Line αυτόματα με την εισαγωγή της « . » η οποία παραπέμπει στις ιδιότητες του αντικειμένου, εμφανίζεται ένα παράθυρο με αλφαβητικά ταξινομημένες τις «ιδιότητες» της μεταβλητής. Ας δώσουμε ορισμένες τιμές:

```
MyLine(1).LineDrawn = False
MyLine(1).Xstart = 500.45
MyLine(1).Ystart = 300.16
MyLine(1).Zstart = 800.99
```

Παραπάνω ορίσαμε την αρχή σε ένα 3Δ σύστημα συντεταγμένων και μια ακόμα ιδιότητα που μας λέει για παράδειγμα ότι η γραμμή 1 δεν έχει σχεδιαστεί ακόμα. Είναι φανερό πόσο εύκολο είναι να δημιουργούμε αντικείμενα και να τα παραμετροποιούμε με ιδιότητες.

Αναφέροντας τον τύπο μεταβλητής Boolean θα πρέπει να πούμε ότι χρησιμοποιείται κυρίως ως «σημαία» ή δείκτης, διότι παίρνει για τιμές μόνο το True και το False. Παράδειγμα σε κάποιο έλεγχο συνθήκης, αν η μεταβλητή π.χ. Pos1 > 0 τότε MyLine(1).Visibility = True

3. Χρήση συνάρτησης Len: Η συνάρτηση αυτή μας επιστρέφει το μέγεθος ή μήκος του ορίσματος που εισάγουμε σε χαρακτήρες (ή bytes). Για παράδειγμα Len("Fanis") θα επιστρέψει την τιμή 5 (5 γράμματα)

ΒΟΗΘΗΜΑ ΓΙΑ ΑΣΚΗΣΕΙΣ

Κατάστρωση συναρτήσεων σε γνωστά προβλήματα Τοπογραφίας

1. Β' Θεμελιώδες: Από 2 σημεία γνωστών συντεταγμένων x, y μας ζητείται η γωνία διεύθυνσης και η μεταξύ τους απόσταση.
    a. Θεωρώ Α το πρώτο σημείο και Β το δεύτερο. Θα υπολογίσω το G(A→B) και το S(A<>B)
    b. Οι τριγωνομετρικές σταθερές και συναρτήσεις που χρειάζομαι είναι: π = 3,14159… ,ATN( ) (: τόξο εφαπτομένης)
    c. Τα βήματα που θα ακολουθήσω είναι:
        i. Δηλώσεις μεταβλητών
        ii. Ανάγνωση τιμών (Αρχείο ή InputBox)
        iii. Κατάστρωση – Υπολογισμός συνάρτησης για απόσταση – γωνία.
        iv. Περιορισμοί τεταρτημορίων για έλεγχο και εύρεση της σωστής γωνίας διεύθυνσης.
        v. Εξαγωγή του αποτελέσματος.

i)
Δηλώσεις μεταβλητών:

```
Dim Xa As Double, Ya As Double, Xb As Double
Dim Yb As Double, Sab As Double, Gab As Double
Dim Pi As Double
Pi = 4 * Atn(1)
```

ii)
Ανάγνωση τιμών από αρχείο:

```
Open "MyFile.txt" For Input As #1
Input #1, Xa, Ya, Xb, Yb
Close #1
```

iii)
Κατάστρωση – Υπολογισμός συνάρτησης για απόσταση – γωνία:

```
Dx = Xb – Xa
Dy = Yb - Ya
```

```
If Dx = 0 And Dy = 0 Then Exit Sub 'Τα σημεία ταυτίζονται
Sab = Sqr((Dx)^2 + (Dy)^2)

If Dy<> 0 Then
Gab = Atn(Dx/Dy)
End If

Gab = Abs(Gab * 200 / pi)                    'μετατροπή από ακτίνια σε βαθμούς
```

iv)

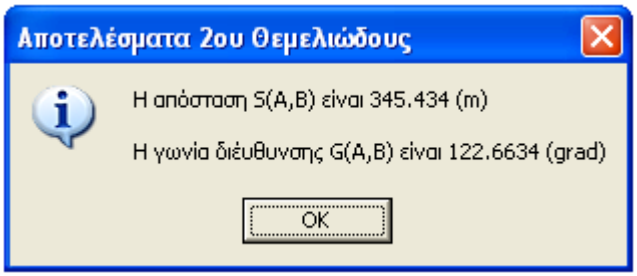
Περιορισμοί τεταρτημορίων για έλεγχο και εύρεση της σωστής γωνίας διεύθυνσης:

```
If Dx > 0 And Dy > 0 Then G = G
If Dx > 0 And Dy < 0 Then G = 200 - G
If Dx < 0 And Dy < 0 Then G = 200 + G
If Dx < 0 And Dy > 0 Then G = 400 - G

' Περιπτώσεις Dx ή/και Dy = 0
If Dy > 0 And Dx = 0 Then
G = 0
End If

If Dy < 0 And Dx = 0 Then
G = 200
End If

If Dx > 0 And Dy = 0 Then
G = 100
End If

If Dx < 0 And Dy = 0 Then
G = 300
End If
```

v)

Εξαγωγή του αποτελέσματος:

```
Msgbox "Η απόσταση S(A,B) είναι " & Format(Sab,"0.000") & " (m) " &
VbCrLf & VbCrLf & "Η γωνία διεύθυνσης G(A,B) είναι "&
Format(Gab,"0.0000") & " (grad) " , vbInformation, "Αποτελέσματα 2ου
Θεμελιώδους"
```



Προσοχή στη χρήση της εντολής MsgBox. Μια καλή παρουσίαση του αποτελέσματος είναι τόσο σημαντική, όσο η ίδια η επεξεργασία του προβλήματος.

Σύνταξη της εντολής *Format:*
Format (μεταβλητή , τύπος Format)

Π.χ.
Met1 = 345.212345
Met2 = Format (Met1 , "0.00")

Η Met2 έχει την τιμή 345.21

Επεξήγηση της δήλωσης VbCrLf
Η VbCrLf χρησιμοποιείται όπως παραπάνω (2 φορές) για την αλλαγή σειράς στην εκτύπωση κάποιου μηνύματος – αποτελέσματος.

Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί η παρακάτω παράσταση:
Chr(13) + Chr(10)
Η οποία θα μπορούσε να καταχωριστεί σε μια μεταβλητή:
NeLi = Chr(13) + Chr(10)
Έτσι η παραπάνω εκτύπωση μέσω του MsgBox θα δώσει το ίδιο αποτέλεσμα με το παρακάτω:
Msgbox "Η απόσταση S(A,B) είναι " & Format(Sab,"0.000") & " (m) " & NeLi & NeLi & "Η γωνία διεύθυνσης G(A,B) είναι "& Format(Gab,"0.0000") & " (grad) " , vbInformation, "Αποτελέσματα 2ου Θεμελιώδους"

2. Υπολογισμός Εμβαδού
    a. Τριγώνου
    b. Κύκλου / κυκλικού τομέα
    c. Πολυγώνου

(a)

Εμβαδόν τριγώνου: Κατάστρωση εναλλακτικών συναρτήσεων

1) Με τον τύπο του Ήρωνα

```
T = (a + b + c) / 2          ' ημιπερίμετρος τριγώνου
E = Sqr( T * (T-a) * (T-b) * (T-c) )
```

2) Με τον νόμο των ημιτόνων

```
E = 0.5 * b * c * Sin(A)    'Η γωνία A σε ακτίνια
```

3) Με χρήση της βάσης και του ύψους του τριγώνου

```
E = 0.5 * base * height
```

Όπου base , height μεταβλητές που μας δίνουν την βάση και το ύψος αντίστοιχα.

(b)

Εμβαδόν κύκλου:

```
E = pi * R^2                 'pi = 3,14159... → pi = 4 * Atn(1)
```

R : ακτίνα του κύκλου

Εμβαδόν κυκλικού τομέα:

```
E = pi * R^2 * m / 400
```

m : Το «άνοιγμα» του κυκλικού τομέα σε βαθμούς

(c)

Εμβαδόν πολυγώνου:

```
For I = 1 To Vertices – 1        'Vertices ο αριθμός των κορυφών
                                 'X(I) ,Y(I) οι συντεταγμένες κάθε κορυφής
    adxi = X(I) + X(I + 1)       'adxi: Μεταβλητή για το άθροισμα των X
    adyi = Y(I) - Y(I + 1)       'adyi: Μεταβλητή για τη διαφορά των Y
    Emb_plus = 0.5 * (adxi * adyi)   'Υπολογισμός αθροιστέου εμβαδού
    E = E + Emb_plus                 'Υπολογισμός συνολικού εμβαδού

Next I

E = Abs (E)                      'Διότι θα μπορούσε να έχει αρνητική τιμή
```

ΠΑΡΑΡΤΗΜΑ Γ

Δημιουργία και χρήση DLL

Τα αρχεία DLL (Dynamic Link Library / βιβλιοθήκη δυναμικής διασύνδεσης) ήταν πολύ διαδεδομένα όταν η υπολογιστική ισχύς των προσωπικών Η/Υ ήταν κατά πολύ μικρότερη, και η μνήμη σχεδόν ανεπαρκής. Τα DLL περιέχουν συναρτήσεις τις οποίες δημιουργεί ο χρήστης και αποθηκεύονται σε ξεχωριστό αρχείο.

Το ίδιο όμως δεν κάνει και το Module; Ναι, αλλά υπάρχουν συγκεκριμένες διαφορές μεταξύ των δύο.

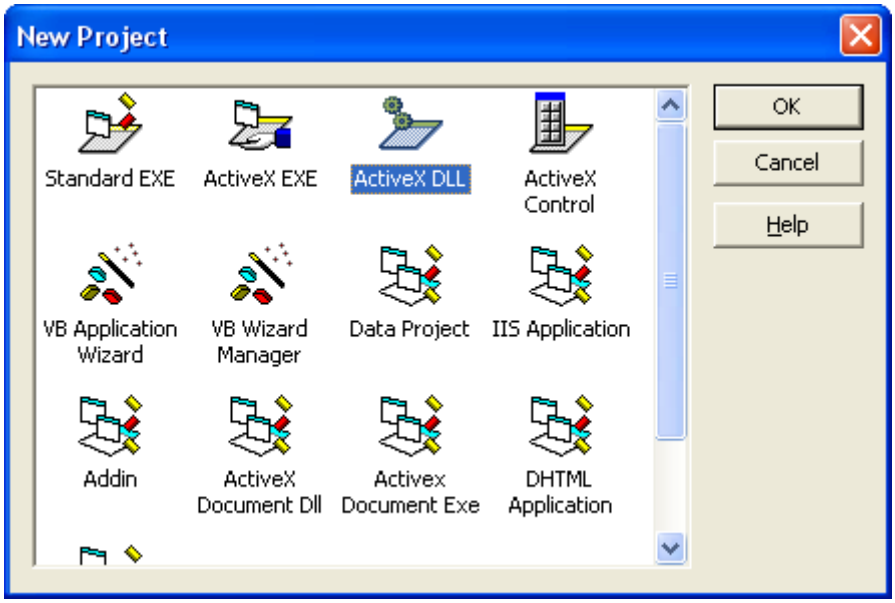
1. Το Module περιέχει όντως συναρτήσεις, αλλά για να χρησιμοποιηθούν αυτές από κάποιο πρόγραμμα, πρέπει να προστεθεί το Module (άνοιγμα του αρχείου) και να ενσωματωθεί σε αυτό. Η παραπάνω ενέργεια μεγαλώνει τον όγκο του προγράμματος και συνεπώς αυξάνει τις απαιτήσεις σε μνήμη κατά την εκτέλεση. Σε αντίθεση, το DLL δεν προστίθεται στο Project μας, αλλά καλείται ως εξωτερικό αρχείο, φορτώνεται στη μνήμη, εκτελείται και στο τέλος αποδεσμεύεται απελευθερώνοντας τη μνήμη.
2. Στην περίπτωση που επιθυμούμε να εκμεταλλευτούμε εμπορικά κάποιες συναρτήσεις που συντάξαμε, το Module δεν αποτελεί καλό τρόπο διανομής. Αυτό συμβαίνει διότι το Module περιέχει κώδικα τον οποίο μπορεί να αντιγράψει οποιοσδήποτε. Σε αντίθεση με αυτό, το DLL περιέχει compiled κώδικα, δηλαδή «μεταγλωττισμένο» τον οποίο δεν μπορεί να «υφαρπάξει» κανείς.
3. Τα DLLs χρησιμοποιούνται από όλες τις σύγχρονες γλώσσες προγραμματισμού, διότι είναι σε εκτελέσιμη μορφή. Έτσι, παρόλο ότι συντάσσουμε και δημιουργούμε ένα DLL στην Visual Basic, μπορούμε να το χρησιμοποιήσουμε στην C, C++, Delphi, Fortran κτλ.

Προσέξτε λίγο την τελευταία παρατήρηση. Τα DLL αποτελούν τον σύγχρονο τρόπο προγραμματισμού. Πλέον, πολλές εταιρίες που

δραστηριοποιούνται στον χώρο αυτό, προτιμούν να δημιουργούν και να εμπορεύονται DLL παρά ολοκληρωμένα προγράμματα.

Ας δούμε όμως τον τρόπο με τον οποίο δημιουργούμε ένα DLL.

Ανοίγουμε την Visual Basic και επιλέγουμε ως project αντί για το «Standard EXE» που επιλέγαμε ως τώρα, το «ActiveX DLL».



Αυτόματα μας δημιουργεί ένα «Class Module» (και όχι κάποια φόρμα), το οποίο όπως και το απλό Module, μπορεί να περιέχει μόνο κώδικα και όχι γραφικά.

Το προεπιλεγμένο όνομα για το Class Module είναι το Class1. Εμείς θα το τροποποιήσουμε στη συνέχεια, μέσω ενός παραδείγματος, και θα δούμε βήμα προς βήμα όλα τα στάδια τα οποία θα πρέπει να ακολουθήσουμε.

Βήμα 1°: Αλλαγή ονόματος κλάσης (Class Module). Αλλάζουμε το όνομα σε ένα περιγραφικό για τη χρήση μας όνομα. Εδώ ας ονομάσουμε την κλάση «clMat» (από την ιδιότητα Name).

Βήμα 2ο: Θα συντάξουμε 2 συναρτήσεις. Ο τρόπος γραφής τους δεν διαφέρει από τον τρόπο γραφής τους σε ένα απλό Module.

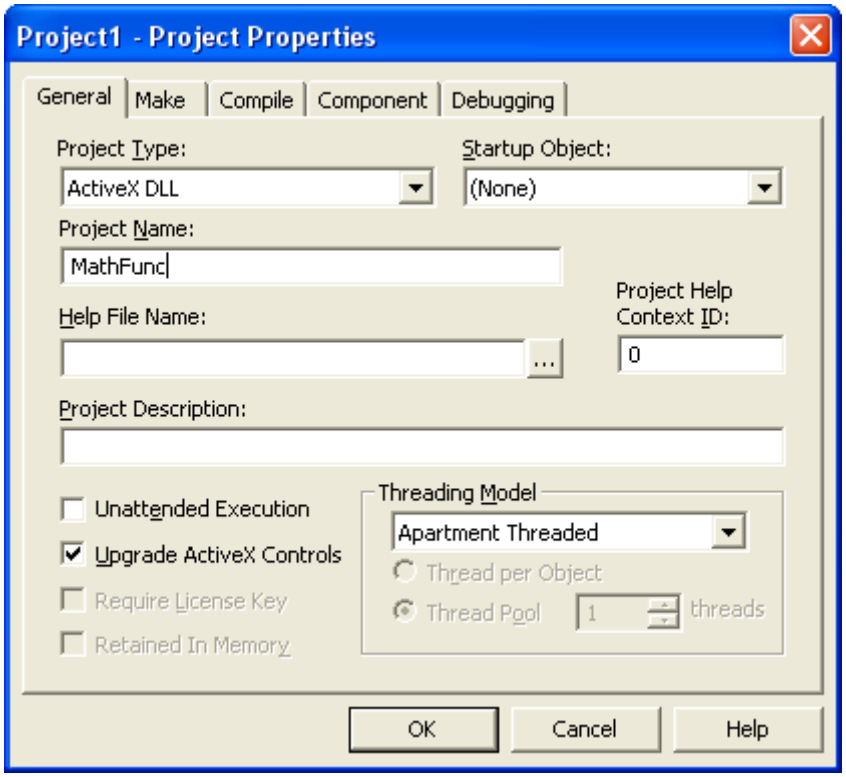
```
Public Function Rand(FromNumber As Double, ToNumber As Double, IntegerType _
As Boolean) As Double
Dim TheSpace As Double
TheSpace = ToNumber - FromNumber
If TheSpace <= 0 Then
MsgBox "Λάθος δεδομένα!", vbCritical, "Προσοχή!"
Exit Function
Else
        Randomize Timer
        If IntegerType = True Then
        Rand = Int((Rnd * TheSpace) + FromNumber)
        Else
        Rand = Rnd * TheSpace + FromNumber
        End If

End If
End Function
```

Η παραπάνω συνάρτηση δέχεται τρία ορίσματα, τα οποία επεξεργάζεται και υπολογίζει έναν τυχαίο αριθμό. Ο τυχαίος αριθμός βρίσκεται ανάμεσα στον πρώτο και δεύτερο αριθμό που εισάγουμε, ενώ το λογικό όρισμα «IntegerType» που θέσαμε, προσδιορίζει αν επιθυμούμε ο τυχαίος να είναι ακέραιος ή όχι.
Θα προσθέσουμε και μια ακόμα συνάρτηση όπως παρακάτω:

```
Public Function Add2(num1 As Double, num2 As Double) As Double
Add2 = num1 + num2
End Function
```

Η συνάρτηση Add2 που δημιουργήσαμε δέχεται ως ορίσματα δυο μεταβλητές διπλής ακρίβειας και υπολογίζει το άθροισμά τους.
Τελειώσαμε με την σύνταξη των συναρτήσεων.

Βήμα 3ο: Από το βασικό Menu της Visual Basic και την βασική κατηγορία [Project] επιλέγουμε την επιλογή [Project1 Properties…]. Εκεί μας εμφανίζεται το παρακάτω παράθυρο, και επιλέγουμε ένα όνομα για το αρχείο μας (εμείς επιλέξαμε το «MathFunc»). Έπειτα πατάμε το κουμπί «OK».



Θα παρατηρήσετε ότι αν ξαναπάτε στο Menu > [Project] η επιλογή [Project1 Properties…] έχει μετατραπεί σε [MathFunc Properties…]. Αυτό σημαίνει ότι η αλλαγή που κάνατε έχει καταχωριστεί.

Βήμα 4°: Αποθηκεύουμε σε κάποια τοποθεσία στο δίσκο το πρόγραμμά μας. Από το βασικό Menu, πηγαίνουμε στην επιλογή κατηγορίας [File] και από εκεί επιλέγουμε την κατηγορία [Save Project As…]. Η επιλογή αυτή θα μας ανοίξει ένα Common Dialog το οποίο χρησιμοποιεί η Visual Basic (όπως και πολλά άλλα προγράμματα). Διατηρούμε τα ονόματα των αρχείων που μας προτείνει διότι εμείς τα επιλέξαμε προηγουμένως. Δημιουργούμε ένα φάκελο στο δίσκο (κάπου που να θυμόμαστε που βρίσκεται) και πατάμε το κουμπί «Save» όσες φορές μας προτείνει για αποθήκευση (εδώ δύο: μια για όλο το project και μια για το Class Module). Έχουμε επιτυχώς αποθηκεύσει το πρόγραμμα, αλλά ακόμα δεν έχουμε δημιουργήσει το DLL.

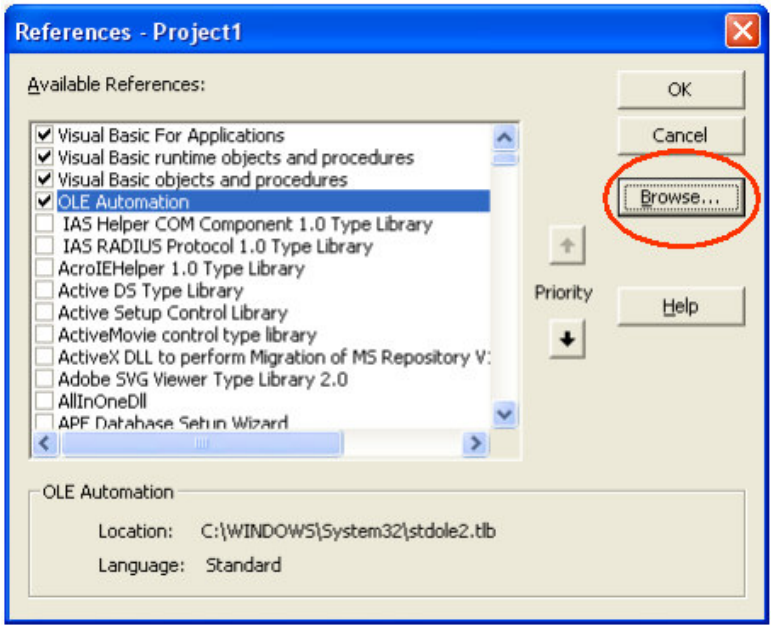
Βήμα 5°: Δημιουργούμε το DLL: Από το Menu > [File] και επιλέγουμε την επιλογή που πλέον γράφει [Make MathFunc.dll…]. (Όπως έχετε προσέξει, οι επιλογές διαχείρισης του project μας προσαρμόζονται αυτόματα στα ονόματα που έχουμε δώσει στα επιμέρους στοιχεία που χρησιμοποιούμε). Η επιλογή αυτή μας εμφανίζει ένα παράθυρο διαλόγου αποθήκευσης. Κρατάμε το όνομα που μας προτείνει και πατάμε το κουμπί «OK». Τη στιγμή της δημιουργίας του DLL βλέπουμε μια μπάρα προόδου της διεργασίας στο πάνω δεξιά μέρος

της οθόνης. Έχουμε πλέον ολοκληρώσει τη δημιουργία μιας βιβλιοθήκης (DLL). Κλείνουμε τη Visual Basic.

Θα δούμε στη συνέχεια πώς μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη που δημιουργήσαμε μέσα από ένα πρόγραμμα το οποίο θα συντάξουμε.

Βήμα 1°: Ανοίξτε και πάλι τη Visual Basic, και δημιουργήστε ένα νέο «Standard EXE» project. Προσθέστε δυο κουμπιά στη φόρμα σας.

Βήμα 2°: Από το βασικό Menu επιλέξτε την βασική κατηγορία [Project] και έπειτα επιλέξτε την κατηγορία [References…]. Από εκεί, πατήστε το κουμπί που γράφει «Browse» και αναζητήστε το φάκελο στον οποίο δημιουργήσατε το «MathFunc.dll».



Μόλις το εντοπίσετε, επιλέξτε το και πατήστε το κουμπί «Open». Το DLL αυτόματα θα προστεθεί στη λίστα, και μάλιστα τσεκαρισμένο (επιλεγμένο). Πατήστε το κουμπί «OK» για να επιστρέψετε στο project σας. Έχετε ήδη δημιουργήσει την πρώτη σύνδεση με το εξωτερικό αρχείο βιβλιοθήκης!

Βήμα 3°: Στα κουμπιά που προσθέσατε στη φόρμα, γράψτε τον ακόλουθο κώδικα:

```
Private Sub Command1_Click()

Dim MyRand As clMat, Anum As Double, Bnum As Double

Set MyRand = New clMat

Anum = MyRand.Rand(7, 23, True)
MsgBox Anum, vbInformation, "Integer"

Bnum = MyRand.Rand(5, 59, False)
MsgBox Bnum, vbInformation, "Real"

End Sub
```

Ας δούμε τι γράψαμε στον παραπάνω κώδικα.

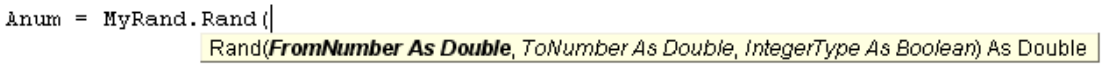
Αρχικά δηλώσαμε μια μεταβλητή, την MyRand ως τύπο clMat ο οποίος δεν είναι άλλος παρά το όνομα της κλάσης που είχαμε δημιουργήσει όταν φτιάχναμε την βιβλιοθήκη μας. Δηλώσαμε επίσης και δυο ακόμα μεταβλητές στις οποίες θα καταχωριστεί το αποτέλεσμα της συνάρτησης.

Δώστε ιδιαίτερη προσοχή στον τρόπο σύνταξης καθώς και στη λειτουργία της Set MyRand = New clMat. *Αυτή η εντολή είναι ουσιαστικά που λέει στην Visual Basic ότι θα χρησιμοποιήσουμε την μεταβλητή MyRand (την οποία έχουμε δηλώσει ως τύπο clMat) για να καλέσουμε ένα νέο στιγμιότυπο της βιβλιοθήκης που περιέχει την κλάση clMat στο πρόγραμμά μας. Μπορούμε να χρησιμοποιήσουμε όσα «στιγμιότυπα» της ίδιας κλάσης επιθυμούμε.*

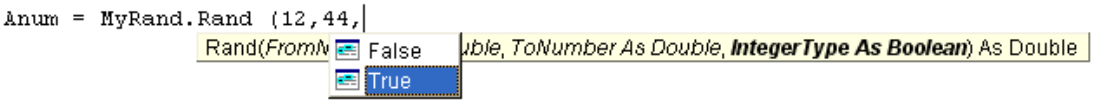
Στις παρακάτω προτάσεις χρησιμοποιούμε τη συνάρτηση Rand την οποία είχαμε δημιουργήσει στο DLL μας, με αναφορά στην MyRand η οποία αντιπροσωπεύει πλέον την clMat.

Κατά την σύνταξη της κλήσης της συνάρτησης, η VB μας ενημερώνει αρχικά για τις συναρτήσεις οι οποίες είναι διαθέσιμες

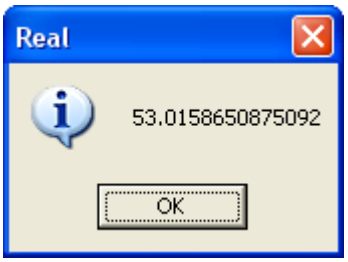
και στη συνέχεια για τα ορίσματα τα οποία είναι αποδεκτά, καθώς και για τους αποδεκτούς τύπους μεταβλητών.



Εισάγοντας τα δεδομένα, προβάλλεται όποια βοήθεια είναι διαθέσιμη:



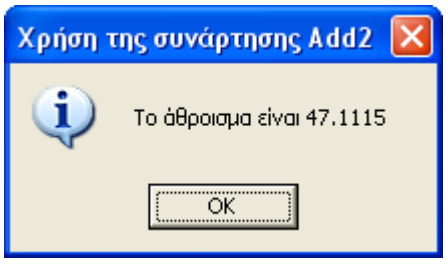
Στο παραπάνω συμβάν πατήματος του κουμπιού Command1, καλέσαμε δυο φορές τη συνάρτηση, μια για ακέραια τιμή και μια για πραγματική. Δείτε δυο παραδείγματα εκτύπωσης των αποτελεσμάτων από το MsgBox:



Ας δούμε τον κώδικα στην υπορουτίνα συμβάντος «κλικ» του Command2:

```
Private Sub Command2_Click()
Dim MyAdded As clMat, Sum As Double
Set MyAdded = New clMat

Sum = MyAdded.Add2(12.455, 34.6565)
MsgBox "Το άθροισμα είναι " & Sum, vbInformation, "Χρήση της συνάρτησης Add2"

End Sub
```

Όμοιος τρόπος χρήσης της βιβλιοθήκης με την υπορουτίνα της Command1. Εδώ τροφοδοτούμε τη συνάρτηση με 2 αριθμούς, καλούμε τη συνάρτηση και αποθηκεύουμε το αποτέλεσμα της συνάρτησης σε μια μεταβλητή, την Sum. Παράδειγμα εκτύπωσης του αποτελέσματος:



Η Dll ωστόσο, παρόλες τις ευκολίες που προσφέρει, έχει και κάποια αρνητικά σημεία. Για παράδειγμα, αν διαθέσετε κάποιο DLL με ένα πρόγραμμά σας, για να ενεργοποιηθεί θα πρέπει ο χρήστης να το καταχωρίσει στο μητρώο των Windows. Αυτό επιτυγχάνεται ως εξής: Από το φάκελο System32 των Windows, επιλέγετε το Regsvr32.exe και «τρέχετε» την βιβλιοθήκη σας μέσω αυτού (Regsvr32.exe MyDll.DLL).

Επίσης θα πρέπει οπωσδήποτε να διατηρείτε τον αρχικό κώδικα του DLL ως project.

ΘΕΜΑΤΑ ΕΡΓΑΣΙΑΣ

1. Δημιουργήστε ένα DLL το οποίο να περιέχει συναρτήσεις επίλυσης των 3 Θεμελιωδών προβλημάτων της Τοπογραφίας.
2. Δημιουργήστε ένα DLL το οποίο θα επιλύει μια όδευση ανοιχτή, εξαρτημένη και με προσανατολισμό στα 2 άκρα.
3. Δημιουργήστε ένα DLL το οποίο θα περιέχει τις παράγωγες τριγωνομετρικές συναρτήσεις.

ΠΑΡΑΡΤΗΜΑ Δ

Ευελιξία στον προγραμματισμό

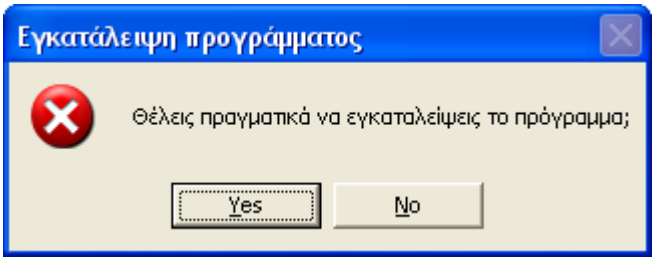
Στο παράρτημα αυτό θα μάθουμε για τη χρήση κάποιων επιπλέον εντολών, τη δημιουργία νέων τύπων μεταβλητών κ.α. με τα οποία θα προσπαθήσουμε να γίνουμε ακόμα καλύτεροι προγραμματιστές. Μπορεί να φανούν ιδιαίτερα απλά, ωστόσο ο τρόπος χρήσης τους είναι αυτός που καθιστά τα συγκεκριμένα στοιχεία (και τις λειτουργίες τους) εξαιρετικά χρήσιμα.

Η εντολή End:

Όταν κατά την εκτέλεση κάποιου κώδικα βρεθεί η εντολή End τότε η λειτουργία του προγράμματος τερματίζεται. Για τον τερματισμό ενός προγράμματος πολλές φορές αρκεί να πατήσουμε το [X] το οποίο βρίσκεται στο πάνω δεξιά τμήμα μιας φόρμας (ControlBox). Ωστόσο το πάτημα αυτού του κουμπιού χωρίς ανάλογο προγραμματισμό θα οδηγούσε σε άμεσο τερματισμό του προγράμματος, χωρίς να μας δοθεί η ευκαιρία να σώσουμε για παράδειγμα κάποια αποτελέσματα. Όμως το παραπάνω «πάτημα» έχει ένα δικό του συμβάν: Το *QueryUnload*. (Θα το βρείτε στην αναδιπλούμενη λίστα συμβάντων της φόρμας). Ας δούμε τον παρακάτω κώδικα:

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
If MsgBox("Θέλεις πραγματικά να εγκαταλείψεις το πρόγραμμα;", _
vbCritical + vbYesNo, "Εγκατάλειψη προγράμματος") = vbNo Then
Cancel = 1
End If
End Sub
```

Το πάτημα του [X] στη φόρμα εμφανίζει πλέον το παρακάτω μήνυμα:



Πατώντας το «Yes» το πρόγραμμα τερματίζεται. Πατώντας «No» το πρόγραμμα επιστρέφει σε κανονική λειτουργία.

Σημειώστε ότι χρησιμοποιήσαμε στο παραπάνω παράδειγμα τον τελεστή Cancel (Boolean) και όχι την εντολή End. Ωστόσο πετύχαμε το επιθυμητό αποτέλεσμα.

Επίσης, επειδή η MsgBox είναι συνάρτηση, χρησιμοποιήσαμε το αποτέλεσμά της σε μια συνθήκη If…End If, χρησιμοποιώντας ως δεδομένο το πλήκτρο που πατήθηκε.

Η εντολή End θα μπορούσε να χρησιμοποιηθεί στο παραπάνω παράδειγμα ως εξής:

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
If MsgBox("Θέλετε πραγματικά να εγκαταλείψετε το πρόγραμμα;", _
vbCritical + vbYesNo, "Εγκατάλειψη προγράμματος") = vbYes Then
End
Else
Cancel = 1
End If
End Sub
```

Επίσης η End μπορεί να χρησιμοποιηθεί σε ένα Menu στην επιλογή «έξοδος από το πρόγραμμα» όπου δεν υπάρχουν τελεστές Cancel κτλ.

Η εντολή DoEvents:

Αυτή η εντολή επιτρέπει στο χρήστη την πλήρη κλήση συμβάντων, παράλληλα με διαδικασίες που εκτελούνται. Κανονικά, όταν η Visual Basic εκτελεί κάποια διαδικασία (για παράδειγμα αποθήκευση δεδομένων σε αρχείο) δεσμεύει όλο το Interface έτσι ώστε ο χρήστης να μην μπορεί να επέμβει. Αν όμως ο χρήστης επιθυμεί να διακόψει μια διαδικασία δεν μπορεί. Αυτό συμβαίνει διότι για να δοθεί προτεραιότητα στο χρήστη, θα πρέπει προηγουμένως να έχει ολοκληρωθεί η διαδικασία που εκτελείται.

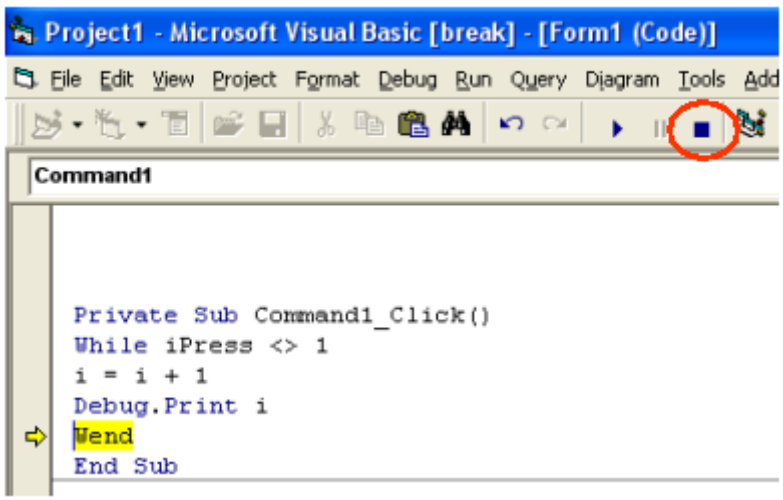
Η DoEvents αλλάζει τα δεδομένα. Επιτρέπει στο χρήστη τη χρήση του Interface ταυτόχρονα με την εκτέλεση κάποιας διαδικασίας. Όλοι μας έχουμε «ακυρώσει» μια λάθος εκτύπωση τη στιγμή που ο εκτυπωτής δουλεύει. Αυτό το επιτυγχάνει η σωστή χρήση της DoEvents. Δείτε το παρακάτω παράδειγμα: (Δημιουργήστε μια φόρμα με δυο κουμπιά και προσθέστε και ένα Module)

```
Public iPress As Integer   'Αυτή η δήλωση βρίσκεται στο Module
```

```
Private Sub Command1_Click()
iPress = 0
While iPress <> 1
i = i + 1
Debug.Print i
Wend
End Sub
```

```
Private Sub Command2_Click()
iPress = 1
End Sub
```

Εκτελέστε το πρόγραμμα και πατήστε το πρώτο κουμπί (Command1). Στο Immediate Window θα εκτυπώνεται διαρκώς η τιμή του i (καθώς αυξάνει διαρκώς μέσα στο βρόγχο While…Wend. Για να σταματήσει ο βρόγχος θα πρέπει η τιμή του iPress να γίνει 1. Αυτό το κάνει το Command2. Πατήστε το! Τι γίνεται; Το Command2 δεν μπορεί να πατηθεί και εμείς είμαστε παγιδευμένοι μέσα σε ένα ατέρμονα βρόγχο!
Σταματήστε το πρόγραμμα πατώντας [Ctrl] + [Scroll Lock] στο πληκτρολόγιο και έπειτα πατήστε το {Stop} της Visual Basic



Αλλάξτε το συμβάν του Command1 (στην ουσία προσθέτοντας την DoEvents) όπως φαίνεται στη συνέχεια:

```
Private Sub Command1_Click()
iPress = 0
While iPress <> 1
DoEvents
i = i + 1
Debug.Print i
Wend
End Sub
```

Εκτελέστε ξανά το πρόγραμμα και πατήστε το Command1. Έπειτα από λίγο πατήστε το Command2. Τι συμβαίνει τώρα; Το Command2 τώρα λειτουργεί κανονικά και η εκτέλεση του βρόγχου τερματίζεται! Αυτό συμβαίνει διότι η DoEvents έδωσε προτεραιότητα στο χρήστη, χωρίς όμως να εμποδίσει την εκτέλεση του προγράμματος.

Η εντολή Stop:

Η Stop χρησιμοποιείται για να σταματήσει την εκτέλεση του κώδικα σε κάποιο συγκεκριμένο σημείο, ώστε να μπορέσουμε να κάνουμε κάποιους ελέγχους. Χρησιμοποιείται αρκετά κατά τη σύνταξη του κώδικα. Είναι αντίστοιχη του [Ctrl] + [Scroll Lock] με τη διαφορά ότι ο προηγούμενος συνδυασμός πλήκτρων θα σταματήσει την εκτέλεση σε άγνωστο σημείο. Για να συνεχίσουμε την εκτέλεση του προγράμματος μετά από κάποιο Stop πατάμε το [F5] ή επιλέγουμε [Continue] από το Menu [Run] της Visual Basic.

Στο διάστημα κατά το οποίο έχουμε σταματημένη την εκτέλεση, μπορούμε με τον κέρσορα να πάμε πάνω από κάποια μεταβλητή και να δούμε την τιμή της.

Δείτε το παρακάτω απόσπασμα:

```
Private Sub Command1_Click()
iPress = 0
While iPress <> 1
Stop
DoEvents
i = i + 1
Deb i = 2 Print i
Wend
End Sub
```

Με κίτρινο χρώμα μας δείχνει το σημείο που σταμάτησε η εκτέλεση. Πήγαμε με τον κέρσορα πάνω από τη μεταβλητή i και μας έδειξε την τρέχουσα τιμή της.

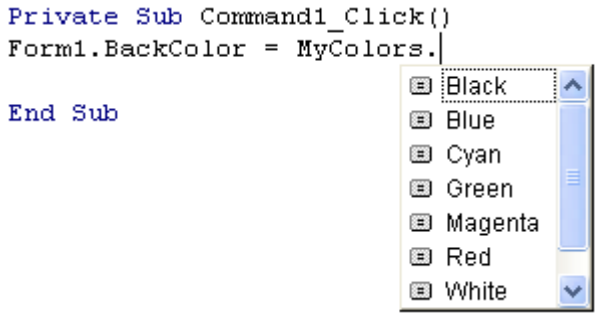
Η δήλωση τύπου Enum:

Η παραπάνω δήλωση (enumerate) μοιάζει πολύ με την Type…End Type και συντάσσεται με τον ίδιο τρόπο. Με μια διαφορά: Κατά τη χρήση της Enum δίνουμε τιμές στις μεταβλητές που περιέχει. Η δήλωσή της γίνεται μόνο μέσα σε κάποιο Module. Ας δούμε το παρακάτω παράδειγμα:

```
Public Enum MyColors
Red = vbRed
Green = vbGreen
Yellow = vbYellow
Black = vbBlack
White = vbWhite
Cyan = vbCyan
Blue = vbBlue
Magenta = vbMagenta
End Enum
```

```
Private Sub Command1_Click()
Form1.BackColor = MyColors.Yellow
End Sub
```

Δήλωσα μια Enum μεταβλητή, την MyColors και έδωσα στις μεταβλητές Red, Green κτλ τιμές χρωμάτων που χρησιμοποιεί η Visual Basic (δεσμευμένες λέξεις, με χαρακτηριστικό το πρόθεμα vb). Έπειτα στο συμβάν κλικ ενός κουμπιού ζήτησα να μπεί ως φόντο στη φόρμα μου το χρώμα που ήταν ορισμένο στην MyColors.Yellow:



Γενικότερα αυτό που πρέπει να ξέρετε είναι ότι οι τιμές που τίθενται στις μεταβλητές που ορίζονται από την Enum δεν μπορούν να αλλάξουν από το χρήστη. Η ευελιξία προσφέρεται μόνο στον προγραμματιστή.

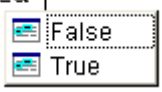
Η δήλωση Def…:

Η παραπάνω δήλωση δεν υπάρχει όπως αναγράφεται. Αποτελεί πρόθεμα άλλου τύπου συντάξεων (για αυτό βάλαμε «...» να ακολουθεί το πρόθεμα). Θα δούμε αναλυτικά τον τρόπο σύνταξης των τύπων που μας ενδιαφέρουν, καθώς και το τι κάνει πραγματικά. Οι δηλώσεις αυτές γίνονται μόνο μέσα σε Module.

Δήλωση	Σύνταξη	Παράδειγμα	Αποτέλεσμα
DefBool (τύπου Boolean)	DefBool LetterRange [,LetterRange]	DefBool B-C	Όλες οι μεταβλητές που αρχίζουν από τα σχετικά περιγραφόμενα γράμματα (ή αρχίζουν από γράμμα το οποίο βρίσκεται ανάμεσα στα Γράμμα1 – Γράμμα2), αυτομάτως δηλώνονται ανάλογα με την πρόταση που χρησιμοποιείται. Για παράδειγμα: DefInt I – K Όσες μεταβλητές αρχίζουν από i, j, k θα θεωρούνται αυτομάτως ακέραιοι.
DefByte (τύπου Byte)	DefByte LetterRange [,LetterRange]	DefByte A, Y - Z	
DefInt (τύπου Integer)	DefInt LetterRange [,LetterRange]	DefInt I – K	
DefLng (τύπου Long)	DefLng LetterRange [,LetterRange]	DefLng L – N	
DefSng (τύπου Single)	DefSng LetterRange [,LetterRange]	DefSng S - U	
DefDbl (τύπου Double)	DefDbl LetterRange [,LetterRange]	DefDbl D - H	
DefDate (τύπου Date)	DefDate LetterRange [,LetterRange]	DefDate W	
DefStr (τύπου String)	DefStr LetterRange [,LetterRange]	DefStr O - R	
DefVar (τύπου Variant)	DefVar LetterRange [,LetterRange]	DefVar V	

Παράδειγμα:

```
DefBool B-C
Dim bAsked

basked=
```



χωρίς να δηλώσουμε τύπο στην Dim, κατευθείαν όταν πήγαμε να θέσουμε τιμή στην μεταβλητή bAsked μας πρότεινε τις True/False διότι η μεταβλητή αρχίζει από b, και το b είναι δηλωμένο ως Boolean (λογικό).

Ανίχνευση λαθών (error trapping):

Η ανίχνευση των λαθών (bugs) σε ένα πρόγραμμα είναι δύσκολη δουλειά αλλά αναγκαία. Είναι το σημείο που κάνει να ξεχωρίζει ο καλός από τον κακό προγραμματιστή. Όλοι μας έχουμε πέσει πάνω σε λάθη που έχουν ξεφύγει από κάποιον προγραμματιστή, με αποτέλεσμα το πρόγραμμα να «σπάσει» και εμείς να χάσουμε όλη τη δουλειά μας.

Λάθη μπορεί πάντοτε να ξεφεύγουν, ακόμα και από τους πιο έμπειρους προγραμματιστές. Δυστυχώς, με μαθηματική ακρίβεια, ο χρήστης ο οποίος θα «ανακαλύψει» το λάθος (χάνοντας όλη την μη αποθηκευμένη δουλειά του) είναι αυτός που θα έχει κάνει την περισσότερη δουλειά και θα έχει ξεχάσει να την «σώσει»! (Νόμος του Murphy!).

Οι μεγάλες εταιρείες που δραστηριοποιούνται στο χώρο του προγραμματισμού, αναθέτουν σε επαγγελματίες «Testers» να δοκιμάσουν το πρόγραμμα για να δουν αν «σπάει», πολύ πριν αυτό κυκλοφορήσει στην αγορά. Εσείς από την πλευρά σας φροντίστε να ακολουθήστε ορισμένους βασικούς κανόνες:

1. Ελέγχετε πάντοτε τα δεδομένα που εισάγει ο χρήστης. Αν το πρόγραμμά σας περιμένει αριθμητική τιμή και κατά λάθος ο χρήστης εισάγει γράμματα ή σύμβολα, το πρόγραμμα θα «σπάσει».
2. Προσέξτε μήπως ο χρήστης από εισαγωγή περίεργων δεδομένων παγιδευτεί σε κάποιο βρόγχο χωρίς διέξοδο.
3. Προσέξτε τον τρόπο με τον οποίο ανοίγετε τα αρχεία. Το κανάλι να είναι πάντα μοναδικό, και εάν πρόκειται για σειριακό αρχείο, στην περίπτωση που θέλουμε να το ανοίξουμε αυτό πρέπει να προϋπάρχει. Προσοχή: ελέγχετε πάντοτε κατά την εισαγωγή δεδομένων από ένα σειριακό αρχείο, το που τελειώνουν τα περιεχόμενά του.
4. Αποφεύγετε τη διαίρεση με το μηδέν. Όταν χρησιμοποιείτε κάποια συνάρτηση η οποία εκτελεί κάποια διαίρεση, ελέγχετε τον διαιρέτη. Το ίδιο ισχύει και για οποιαδήποτε μαθηματική πράξη, η οποία εμπλέκει δεδομένα που εισάγει ο χρήστης (π.χ. τετραγωνική ρίζα αρνητικού αριθμού).

Θα μπορούσαμε να επισημάνουμε πολλά περισσότερα, αλλά η «γλύκα» του προγραμματισμού είναι «να ανακαλύπτεις μόνος σου τη σωστή λύση και να μαθαίνεις μέσα από τα λάθη σου».

Συμβουλή: Οποτεδήποτε εμφανίζεται ένα MsgBox μηνύματος λάθους που σας ειδοποιεί για το λάθος που κάνατε, μην το αγνοείτε. Ανατρέξτε στην βοήθεια που σας προτείνει για σχετικές πληροφορίες. Με αυτό τον τρόπο μαθαίνετε!

Ας δούμε ένα παράδειγμα αντιμετώπισης λάθους εισαγωγής δεδομένων από το χρήστη. Εισάγετε σε μια φόρμα ένα κουμπί και ένα TextBox.

```
Private Sub Command1_Click()
Dim anum As Double, bnum As Variant
bnum = Text1.Text
On Error GoTo ErrorTrap
anum = bnum * 1
MsgBox "Πληκτρολογήσατε αριθμό", vbInformation, "Απόκριση"

Exit Sub
ErrorTrap:
Err.Clear
MsgBox "Παρακαλώ πληκτρολογήστε αριθμό", vbExclamation, "Λάθος δεδομένα"
```

```
Text1.SetFocus

End Sub
```

Πολλά άγνωστα σημεία στο παραπάνω παράδειγμα. Ας δούμε τι συμβαίνει:

On Error GoTo ErrorTrap: Αν βρεθεί λάθος πήγαινε στην εσωτερική υπορουτίνα με το όνομα ErrorTrap. Είναι ουσιαστικά ένα set εντολών ελέγχου, το οποίο χρησιμοποιούνταν σε παλαιότερες εκδόσεις της Basic (GwBasic κ.α.).

anum = bnum * 1 : Τι γίνεται εδώ; Έχοντας δηλώσει έναν αριθμό ως πραγματικό διπλής ακρίβειας, πάω να του δώσω την τιμή που διάβασε από το TextBox πολλαπλασιασμένη με το 1. Αν είναι αριθμός το περιεχόμενο της μεταβλητής bnum, τότε προφανώς δεν υπάρχει πρόβλημα. Αν δεν είναι όμως και είναι οτιδήποτε άλλο, δημιουργείται ένα λάθος! Διότι δεν μπορούμε να πολλαπλασιάσουμε λέξη με αριθμό! Αυτό το λάθος πυροδοτεί την On Error GoTo και η εκτέλεση μεταφέρεται στην εσωτερική υπορουτίνα (την οποία δηλώνουμε ως ΟνομαΥπορουτίνας:). Εκεί σβήνουμε το λάθος με την Err.Clear ώστε να μην πυροδοτηθεί ξανά κατά λάθος το ίδιο συμβάν, εμφανίζεται ένα σχετικό προειδοποιητικό μήνυμα, και η VB εστιάζεται ξανά στο TextBox.

Όταν χρησιμοποιούμε τέτοιου είδους εσωτερικές υπορουτίνες οι οποίες θέλουμε να εκτελούνται κάτω από ορισμένες προϋποθέσεις και μόνο, τότε χρησιμοποιούμε πριν από αυτές την εντολή Exit Sub η οποία ενεργεί ακριβώς σαν την End Sub. Η μόνη διαφορά είναι ότι μπορούμε να την χρησιμοποιήσουμε όσες φορές επιθυμούμε μέσα σε μια υπορουτίνα.

Αν δεν μας καλύπτει μόνο η Err.Clear, η οποία όπως είπαμε «καθαρίζει» το λάθος, μπορούμε για περισσότερες πληροφορίες να επικαλεστούμε τις παρακάτω σχετικές εντολές:

Err.Description :
Πιθανή χρήση : Debug.Print Err.Description
Λειτουργία : Μας δίνει περιγραφικά το λάθος που δημιουργήθηκε

Err.Number :
Πιθανή χρήση : Debug.Print Err.Number
Λειτουργία : Μας δίνει τον αριθμό του λάθους που δημιουργήθηκε, όπως έχει καταχωριστεί από τη Microsoft.

Err.Source :
Πιθανή χρήση : Debug.Print Err.Source

Λειτουργία : Μας επιστρέφει το όνομα του γενικού αντικειμένου το οποίο προκάλεσε το λάθος.

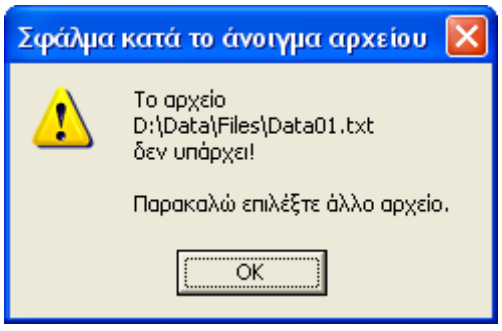
Err.Raise αριθμητικό όρισμα :
Πιθανή χρήση : Err.Raise Err.Number
Λειτουργία : Μας εμφανίζει το κανονικό παράθυρο λάθους της Visual Basic. Μπορούμε να καλέσουμε το παραπάνω και χωρίς την ύπαρξη λάθους.

Όπως είδατε η αντιμετώπιση των λαθών θέλει στρατηγική. Εμείς περιγράψαμε την αντιμετώπιση μόνο ενός προβλήματος. Τι θα μπορούσε να γίνει με τα αρχεία; Όσον αφορά την επιλογή καναλιού, τα πράγματα είναι εύκολα με την χρήση της FreeFile. Πώς όμως μπορούμε να ανιχνεύσουμε την ύπαρξη ενός αρχείου που προσπαθεί να ανοίξει ο χρήστης για ανάγνωση;

Σκεφτείτε όλα όσα είπαμε για τα αρχεία, τόσο τα άμεσα όσο και τα σειριακά. Το προγραμματιστικό μέρος όμως, σας το αφήνουμε για άσκηση.

Άσκηση:

Δημιουργήστε ένα project στο οποίο ο χρήστης με χρήση του Common Dialog θα επιλέγει ένα αρχείο σειριακό για ανάγνωση. Αυτό που πρέπει να κάνετε είναι να ανιχνεύσετε αν το αρχείο, το όνομα του οποίου επιλέγει ο χρήστης, υπάρχει στον δίσκο. Χειριστείτε το λάθος του χρήστη με επαγγελματικότητα. Ας λαμβάνει ένα μήνυμα του τύπου:



Καλή επιτυχία!