ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΜΕ VISUAL BASIC & C++ BUILDER

ΧΡΗΣΤΟΣ Γ. ΤΡΙΑΝΤΑΦΥΛΛΟΥ

ctriantafy@sch.gr

www.christriantafyllou.eu

ΙΑΝΟΥΑΡΙΟΣ 2020

Π Ρ Ο Λ Ο Γ Ο Σ

Το σύγγραμμα με τίτλο “ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΓΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ VISUAL BASIC & C++ BUILDER” αποτελεί μία εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό, σ' ένα δηλαδή από τα βασικότερα και πιο σύγχρονα πεδία της Πληροφορικής επιστήμης. Παρουσιάζονται βασικές έννοιες, όπως: η έννοια του αντικειμένου (object), ιδιότητες (properties) και μέθοδοι (methods) αντικειμένου, μηνύματα (messages), η έννοια της κλάσης (class), η έννοια της αφαίρεσης (abstraction), γενίκευση- εξειδίκευση (generalization-specialization), κληρονομικότητα (inheritance).

Ο σκοπός αυτού του συγγράμματος είναι να διδάξει σε μαθητές ή σε προπτυχιακούς φοιτητές, συναφών σχολών της Πληροφορικής, τις βασικές αρχές του προγραμματισμού με γλώσσες υψηλού επιπέδου, χρησιμοποιώντας τη γλώσσα Visual Basic & C++ Builder.

Το βιβλίο αυτό έχει διαρθρωθεί σε δύο βασικά μέρη. Στο Α’ μέρος, καλύπτονται οι βασικές αρχές αντικειμενοστραφούς προγραμματισμού και το περιβάλλον της Visual Basic, ενώ στο Β’ μέρος καλύπτεται το περιβάλλον της C++ Builder.

Ο συγγραφέας

Χρήστος Γ. Τριανταφύλλου

Λέξεις κλειδιά : αντικειμενοστραφής προγραμματισμός, αντικειμενοστρεφής προγραμματισμός, κλάσεις, αντικείμενα, ενθυλάκωση, αφαίρεση, σύνθεση, ιεραρχίες, κληρονομικότητα, αφηρημένοι τύποι δεδομένων, Visual Basic, Visual Studio, C++ Builder.

Ευχαριστίες

Στο σημείο αυτό, θα ήθελα να κάνω μία μικρή αναφορά σε όλους οι οποίοι συνέβαλαν με οποιονδήποτε τρόπο στη συγγραφή αυτής της διπλωματικής εργασίας.

Θέλω να ευχαριστήσω όλη την οικογένειά μου για την αμέριστη συμπαράσταση-στήριξη και ανοχή τους, αφού ήταν αναγκαίο να με στερηθούν, ώστε να καταφέρω τον στόχο μου. Ιδιαίτερα ευχαριστώ θερμά την σύζυγο μου Ελένη και την κόρη μου Μαρία-Ιωάννα για την ανεκτίμητη Φιλολογική ματιά τους σε όλο το κείμενο του εν λόγω βιβλίου, καθώς και τον γιό μου Γιώργο για την τεχνική υποστήριξη και το άριστο αποτέλεσμα της σχεδίασης των εξωφύλλων των βιβλίων μου.

Επίσης, ιδιαίτερα ευχαριστώ τους γονείς μου Γιώργο & Γιαννούλα για τις αρχές και τις αξίες της ζωής με τις οποίες με γαλούχησαν.

Χρήστος Γ. Τριανταφύλλου

Περίληψη

Ο σκοπός της συγγραφής του συγκεκριμένου βιβλίου είναι η εισαγωγή του αναγνώστη στον αντικειμενοστραφή προγραμματισμό και στη χρήση των γλωσσών Visual Basic & C++ (περιβάλλον Builder) για την ανάπτυξη ολοκληρωμένων εφαρμογών δίνοντας έμφαση στη διαφοροποίηση του προγραμματισμού αυτού σε σχέση με τον παλαιότερο δομημένο προγραμματισμό.

Το βιβλίο αναπτύσσει τις βασικές αρχές του αντικειμενοστραφούς προγραμματισμού: δηλαδή, αφαιρετικοί τύποι δεδομένων, ενθυλάκωση, κληρονομικότητα, πολυμορφισμός, καθώς και τα βασικά στοιχεία των δύο προαναφερθέντων γλωσσών προγραμματισμού όπως αντικείμενα, κλάσεις, διεπαφές αντικειμένων και μεθόδους. Τέλος, χρησιμοποιώντας τα εργαλεία του ολοκληρωμένου περιβάλλοντος ανάπτυξης λογισμικού, της Basic & Builder, αναπτύσσονται εφαρμογές-παραδείγματα για περισσότερη κατανόηση και τέλος παραθέτονται ερωτήσεις με τις απαντήσεις τους, πάνω σε θεωρητικό γνώσεις καθώς και στην ανάπτυξη μικροεφαρμογών.

Ο Αντικειμενοστραφής Προγραμματισμός συνδέεται άμεσα με τα αντικείμενα και κατά συνέπεια, όπως θα δούμε και στη συνέχεια, και με τις κλάσεις. Στην ουσία, πρόκειται για έναν τρόπο οργάνωσης των προγραμμάτων που γράφουμε και αναπτύσσουμε. Ο τρόπος αυτός οργάνωσης, δεν είναι φυσικά ούτε μοναδικός, ούτε καν ο βέλτιστος. Υπάρχουν κι άλλοι τρόποι οργάνωσης όπως ο διαδικαστικός (procedural) και ο συναρτησιακός (functional). Συνηθίζεται, αυτοί οι τρόποι οργάνωσης ή αν θέλετε τεχνικές οργάνωσης των προγραμμάτων να ονομάζονται προγραμματιστικά παραδείγματα (programming paradigms).

Η επιλογή του προγραμματιστικού παραδείγματος το οποίο θα χρησιμοποιήσουμε εξαρτάται από το πρόβλημα το οποίο καλούμαστε να επιλύσουμε, αλλά και από τη γλώσσα προγραμματισμού την οποία διαθέτουμε και χρησιμοποιούμε.

Η Visual Basic ήταν ένα από τα πρώτα συστήματα που κατέστησαν πρακτική τη σύνταξη προγραμμάτων για το λειτουργικό σύστημα των Windows. Αυτό ήταν δυνατό επειδή η VB περιελάμβανε εργαλεία λογισμικού για τη δημιουργία αυτόματου λεπτομερούς προγραμματισμού που απαιτείται από τα Windows. Αυτά τα εργαλεία λογισμικού όχι μόνο δημιουργούν προγράμματα Windows, αλλά εκμεταλλεύονται επίσης πλήρως τον γραφικό τρόπο λειτουργίας των Windows, επιτρέποντας στους προγραμματιστές να «σχεδιάσουν» τα συστήματά τους με ένα ποντίκι στον υπολογιστή. Αυτός είναι ο λόγος που ονομάζεται "Visual" Basic.

Η τελική έκδοση 6 βγήκε το 1998. Η εκτεταμένη υποστήριξη της Microsoft έληξε το Μάρτιο του 2008 και ορίστηκε διάδοχος της η Visual Basic.NET (γνωστή απλά ως Visual Basic).

Όσον αφορά την C++ Builder είναι ένα περιβάλλον ταχείας ανάπτυξης εφαρμογών, που αναπτύχθηκε αρχικά από την Borland, για τη σύνταξη προγραμμάτων στη γλώσσα προγραμματισμού C++ που στοχεύει τα Windows iOS, αλλά και για αρκετές εκδόσεις macOS & Android.

Ξεκίνησε από την έκδοση Borland C++Builder1 το 1997, πέρασε στην έκδοση 5 το 2000, στην 6 το 2002, το 2005 εκδόθηκε η Borland C++Builder2006, από το 2009 ανήκει στην Embarcadero Technologies (θυγατρική της Idera) με την έκδοση Borland C++Builder2006 και έφτασε τον Αύγουστο του 2009 στην έκδοση Borland C++Builder2010 και συνέχισε με υποεκδόσεις της.

Ο συγγραφέας

Πίνακας περιεχομένων

1. Εισαγωγή στη **Visual Basic -** Α' Μέρος

Η **Visual Basic** (**VB**), μ έλος της ομάδας προγραμμάτων του Microsoft Visual Studio, αποτελεί την 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».

Εκτός από την GW Basic υπήρξαν και άλλες εκδόσεις όπως η Quick Basic της Microsoft και η Turbo Basic της Borland.

Με την VB η αρχή λειτουργίας έχει αλλάξει οριστικά πλέον. Δηλαδή, δημιουργώντας και εισάγοντας «αντικείμενα» οδηγηθήκαμε σε μια λειτουργία του τύπου «κάνω κάτι όταν με καλέσεις», συμπεριλαμβανομένου και του τερματισμού. Ο προγραμματισμός που ακολουθεί αυτή τη γενική λογική ονομάζεται «αντικειμενοστραφής» ή «αντικειμενοστρεφής» και έχει ως προτεραιότητα την επέμβαση του χρήστη για την εκτέλεση οποιασδήποτε λειτουργίας.

Η Visual Basic δημιουργήθηκε αρχικά για να διευκολύνει τη σύνταξη προγραμμάτων για το λειτουργικό σύστημα υπολογιστών με Windows, είναι ένα σύστημα προγραμματισμού υπολογιστών που αναπτύχθηκε και ανήκει στη Microsoft. Η Visual Basic αναφέρεται συχνά χρησιμοποιώντας μόνο τα αρχικά, VB. Και είναι ίσως το πιο διαδεδομένο σύστημα προγραμματισμού υπολογιστών στην ιστορία του λογισμικού.

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, κ.ά.).

Από το 1991, όταν κυκλοφόρησε για πρώτη φορά από τη Microsoft, υπήρξαν εννέα εκδόσεις της Visual Basic έως το VB.NET 2005. Οι πρώτες έξι εκδόσεις ονομάστηκαν όλες Visual Basic. Το 2002, η Microsoft παρουσίασε τη Visual Basic .NET 1.0, μια πλήρως επανασχεδιασμένη και επανεγγραφόμενη έκδοση που ήταν βασικό μέρος μιας πολύ μεγαλύτερης αρχιτεκτονικής υπολογιστών. Οι πρώτες έξι εκδόσεις ήταν όλες «συμβατές προς τα πίσω». Αυτό σημαίνει ότι οι νεότερες εκδόσεις του VB θα μπορούσαν να χειριστούν προγράμματα γραμμένα με παλαιότερη έκδοση. Επειδή η αρχιτεκτονική .NET ήταν μια τόσο ριζική αλλαγή, οι παλαιότερες εκδόσεις της Visual Basic πρέπει να ξαναγραφούν πριν μπορούν να χρησιμοποιηθούν με το .NET. Πολλοί προγραμματιστές προτιμούν ακόμη τη Visual Basic 6.0 και μερικοί χρησιμοποιούν ακόμη και παλαιότερες εκδόσεις. Η τελευταία έκδοση της είναι η Microsoft Visual Basic for Windows, V2019.16.9 και λειτουργεί σε λειτουργικά συστήματα: Windows XP; Windows 2000; Windows ME; Windows NT; Windows Server 2003 και νεότερα έως τα Windows 10.

Η VB.NET είναι σίγουρα "αντικειμενοστραφής". Μία από τις μεγάλες αλλαγές που εισήγαγε το .NET ήταν η πλήρης αντικειμενοστραφής αρχιτεκτονική.

Η γλώσσα **Visual Basic** λοιπόν είναι μια γλώσσα **event-driven** (καθοδηγούμενη από γεγονότα) και **object-oriented** (αντικειμενοστραφής).

1.1 Event Programming-Προγραμματισμός Συμβάντων

Όπως αναφέραμε ήδη, στον παραδοσιακό Διαδικαστικό και Δομημένο προγραμματισμό η εφαρμογή η ίδια καθορίζει ποια τμήματα κώδικα θα εκτελεστούν και με ποια σειρά. Η εκτέλεση ακολουθεί μια προκαθορισμένη διαδρομή μέσα στην εφαρμογή, καλώντας όπου χρειάζεται διαδικασίες.

Αντίθετα, σε μια εφαρμογή **event-driven** ο κώδικας δεν ακολουθεί προκαθορισμένη διαδρομή αλλά τα διάφορα τμήματα κώδικα εκτελούνται ανάλογα με τα γεγονότα (events) που συμβαίνουν, τα οποία γεγονότα προκαλούνται: από ενέργειες του χρήστη, από το σύστημα ή από την ίδια την εφαρμογή.

Η σειρά με την οποία συμβαίνουν αυτά τα γεγονότα καθορίζει και τη σειρά εκτέλεσης του κώδικα, έτσι κάθε φορά που εκτελείται η εφαρμογή, ο κώδικας εκτελείται με διαφορετική σειρά.

Η Visual Basic χρησιμοποιείται για την δημιουργία εφαρμογών (προγραμμάτων) για Microsoft Windows. Περιλαμβάνει εργαλεία (tools) που βοηθούν τον προγραμματιστή να δημιουργήσει προγράμματα που να έχουν χαρακτηριστικά όμοια με αυτά των Windows, χωρίς να πρέπει να γράψει πολλές γραμμές κωδικοποίησης (codes).

Η Visual Basic είναι γλώσσα οπτικού προγραμματισμού, για να το κατανοήσουμε θα μπορούσαμε να συγκρίνουμε το περιβάλλον της Visual Basic με ένα θέατρο, όπου ο προγραμματιστής είναι ο σκηνοθέτης του έργου.

Η κάθε σκηνή, του έργου, είναι μία φόρμα και οι ηθοποιοί είναι τα «αντικείμενα» που έχει πάνω της αυτή η φόρμα, τα οποία, μέσω κώδικα που γράφουμε εμείς ως σκηνοθέτες, εκτελούν-παίζουν τον ρόλο που τους αναθέσαμε.

Αυτοί οι ηθοποιοί, που λέγονται «αντικείμενα» στην Visual Basic, μπορεί να είναι ένα κουμπί (button), μια εικόνα, ένα πτυσσόμενο μενού κ.ά.. Κάθε ηθοποιός-αντικείμενο ξέρει τι πρέπει να κάνει, μέσω των εντολών που του έχει δώσει ο σκηνοθέτης-προγραμματιστής.

Σε μια γλώσσα οπτικού προγραμματισμού ο προγραμματιστής ξεκινά την ανάπτυξη της εφαρμογής του δημιουργώντας πρώτα απ' όλα τις φόρμες της. Αυτό σημαίνει ότι τοποθετεί πάνω σε μια κενή φόρμα, με τη βοήθεια του πληκτρολογίου και του ποντικιού ή άλλης παρόμοιας συσκευής, όλα τα αντικείμενα που χρειάζεται. Η φόρμα και ότι βρίσκεται πάνω σε αυτήν έχουν συγκεκριμένες ιδιότητες με προκαθορισμένες τιμές. Ο προγραμματιστής μπορεί να αλλάξει όσες από τις τιμές των ιδιοτήτων κρίνει σκόπιμο.

Όταν μία event-driven εφαρμογή ξεκινά, αρχικά εμφανίζεται μια φόρμα, η οποία απλά περιμένει ένα γεγονός από τον χρήστη, από το σύστημα ή από την εφαρμογή.

Βέβαια, για να δούμε κάτι να εκτελείται, πρέπει να υπάρχει γραμμένος κώδικας, αλλιώς απλά το γεγονός αγνοείται. Οπότε η εφαρμογή τίθεται σε αναμονή εκ νέου περιμένοντας το επόμενο γεγονός.

1.2 Object-oriented Programming-Αντικειμενοστραφής Προγραμματισμός

Ο Αντικειμενοστραφής Προγραμματισμός αποτελεί εξέλιξη του Δομημένου Προγραμματισμού, τεχνικής που υλοποιούν οι γλώσσες προγραμματισμού 3ης γενιάς, και είναι η κυρίαρχη τεχνική στον προγραμματισμό από τις αρχές του '90. Κατ' αναλογία με τον φυσικό κόσμο, εισάγει στον προγραμματισμό την έννοια του Αντικειμένου (Object).

Ο Αντικειμενοστραφής Προγραμματισμός (Object-Oriented Programming ή OOP για συντομία), όπως λέει και το όνομά του συνδέεται άμεσα με τα αντικείμενα και κατά συνέπεια, όπως θα δούμε και στη συνέχεια, και με τις κλάσεις. Κατ' ουσίαν, ο OOP είναι ένας τρόπος οργάνωσης των προγραμμάτων που γράφουμε. Ο τρόπος αυτός οργάνωσης, δεν είναι φυσικά ούτε μοναδικός, ούτε καν ο βέλτιστος. Άλλοι τρόποι οργάνωσης είναι ο διαδικαστικός (procedural ή imperative) και ο συναρτησιακός (functional).

Βασικότατο λοιπόν στον αντικειμενοστραφή προγραμματισμό είναι η κλάση-class, η οποία είναι μία αυτοτελής και αφαιρετική αναπαράσταση κάποιας κατηγορίας αντικειμένων (είτε φυσικών αντικειμένων του πραγματικού κόσμου είτε νοητών) σε ένα περιβάλλον προγραμματισμού. Πρακτικά δηλαδή, είναι ένας νέος τύπος δεδομένων ή αλλιώς το προσχέδιο μίας δομής δεδομένων με δικά της περιεχόμενα (μεταβλητές και διαδικασίες).

Τα περιεχόμενα αυτά δηλώνονται είτε ως δημόσια-public, είτε ως ιδιωτικά-private (τα ιδιωτικά βέβαια δεν είναι προσπελάσιμα από κώδικα εκτός της κλάσης).

Σε ένα πρόγραμμα έχουμε πολλές κλάσεις και πολλά αντικείμενα, τα οποία αντικείμενα όταν πλέον δεν χρειάζονται στο πρόγραμμα αποδεσμεύουν τις πηγές και τους πόρους που χρησιμοποίησαν ώστε να είναι διαθέσιμες για χρήση από άλλα αντικείμενα.

Τώρα, οι διαδικασίες των κλάσεων καλούνται μέθοδοι-methods και οι μεταβλητές τους γνωρίσματα-attributes ή πεδία-fields.

Το αντικείμενο τώρα είναι το στιγμιότυπο μιάς κλάσης, είναι δηλαδή ένα αντίγραφο από το ίδιο "καλούπι" της κλάσης.

Πληρέστερη ανάλυση του Αντικειμενοστραφούς προγραμματισμού ξεφεύγει από τον σκοπό συγγραφής του εν λόγω βιβλίου, για αυτό θα περιοριστούμε στην παράθεση ορισμένων από τα βασικά χαρακτηριστικά του:

- Η εφαρμογή κατακερματίζεται σε Αντικείμενα **,** καθένα από τα οποία αποτελεί ξεχωριστή οντότητα, με δικό του κώδικα και δεδομένα και ανεξάρτητη συμπεριφορά μέσα στην εφαρμογή. Το τμήμα που αντικειμένου που επικοινωνεί με τον έξω κόσμο αποτελεί το interface (διεπαφή) του και υλοποιείται με ιδιότητες (properties) και μεθόδους (methods).
- Κάθε αντικείμενο προέρχεται από μια Κλάση (Class) και δημιουργείται σαν Στιγμιότυπο (Instance) της κλάσης του.
- Θεμελιώδεις έννοιες του Αντικειμενοστραφούς Προγραμματισμού είναι ο Πολυμορφισμός (Polymorphism), η Κληρονομικότητα (Inheritance) και η Απόκρυψη (Encapsulation).
- Πολυμορφισμός είναι η δυνατότητα να έχουμε πολλές κλάσεις με διαφορετικό κώδικα αλλά με το ίδιο interface. Έτσι μπορούμε να γράφουμε προγράμματα που χρησιμοποιούν το interface αυτό χωρίς να μας ενδιαφέρει ποιος από τους τύπος αντικειμένου χρησιμοποιείται κατά την εκτέλεση της εφαρμογής. Για παράδειγμα, μία μέθοδος Draw μπορεί να είναι κοινή στις κλάσεις

Circle, Square & Rectangle, αλλά εσωτερικά λειτουργεί διαφορετικά σε κάθε κλάση. Ωστόσο, ο προγραμματιστής και στα 3 αντικείμενα την καλεί με το ίδιο όνομα, απλοποιώντας την σύνταξη του προγράμματος.

- Κληρονομικότητα είναι η δυνατότητα μια κλάση να βασίζεται σε μια άλλη, κληρονομώντας το interface και τις λειτουργίες της αρχικής κλάσης. Κατόπιν, μπορούν να προστεθούν στη νέα κλάση δικές τις ιδιότητες και μέθοδοι ή να τροποποιηθούν οι υπάρχουσες. Για παράδειγμα, αν έχουμε μια κλάση που λέγεται Autokinhto με ιδιότητες και μεθόδους κοινές για όλα τα αυτοκίνητα, με χρήση της κληρονομικότητας μπορούμε εύκολα να δημιουργήσουμε τις κλάσεις BMW, Toyota, Citroen κ.α, στις οποίες δεν χρειάζεται να ορίσουμε ξανά μεθόδους και ιδιότητες κοινές σε όλα τα αυτοκίνητα αλλά μόνο τις ιδιαίτερες για το κάθε αυτοκίνητο.

- Απόκρυψη είναι η δυνατότητα ενός αντικειμένου να λειτουργεί σαν 'black box'. Το κάθε αντικείμενο περιέχει όλα τα δεδομένα και τον κώδικα που χρειάζεται για να λειτουργήσει, αθέατα στο πρόγραμμα και τα άλλα αντικείμενα. Η επικοινωνία του αντικειμένου με τον 'έξω κόσμο' γίνεται μόνο με τις ιδιότητες και τις μεθόδους του, οι οποίες αποτελούν το interface του. Το πρόγραμμα επικοινωνεί με τα αντικείμενα στέλνοντας τους 'μηνύματα' (messages). Και τα αντικείμενα αντιδρούν στα μηνύματα αυτά μέσω των μεθόδων τους.

Κάθε αντικείμενο υποστηρίζει τις δικές του Ιδιότητες και Μεθόδους:

- Ιδιότητες (Properties). Προσδιορίζουν διάφορες ιδιότητες του αντικειμένου, π.χ. το αντικείμενο Form έχει τις ιδιότητες form.BackColor, form.Caption, form.BorderStyle κ.α.

- Μεθόδους (Methods). Είναι οι ενέργειες που μπορεί να εκτελέσει το αντικείμενο, Π.χ. Clipboard.SetText

1.3 Αντικείμενα στην **VB**

Όλες οι αντικειμενοστραφής γλώσσες, έτσι και η Visual Basic, υποστηρίζει αντικείμενα, όπως:

- Φόρμες . Αποτελούν το βασικό στοιχείο επικοινωνίας της εφαρμογής με το χρήστη.
- Κλάσεις (Classes). Για τη δημιουργία αντικειμένων οριζόμενων από το χρήστη
- Διάφορα Εργαλεία (Controls). Αποτελούν, μαζί με τις φόρμες, τα υλικά σχεδιασμού της εφαρμογής και μπορεί να είναι : Εσωτερικά (Intrinsic), όπως τα πλήκτρα, οι ετικέτες, τα πλαίσια κειμένου κ.α. ActiveX ή Ενθετα (Insertable).
- Ειδικά αντικείμενα . Αντικείμενα που δεν έχουν οπτική υπόσταση όπως το Clipboard, ο Debugger, Timer κ.α.
- Αντικείμενα Δεδομένων (Database objects), για διαχείριση βάσεων δεδομένων.

1.4 Ιδιότητες αντικειμένων στη **Visual Basic**

Οι ιδιότητες των αντικειμένων στη Visual Basic καθορίζουν διάφορα χαρακτηριστικά τους. Η αναφορά σε μια ιδιότητα ενός αντικειμένου γίνεται γράφοντας το όνομα του αντικειμένου, την τελεία "." και το όνομα της ιδιότητας. Για παράδειγμα: η δήλωση Form1.BackColor αναφέρεται στην ιδιότητα BackColor του αντικειμένου Form1.

Κάθε αντικείμενο της Visual Basic έχει τις δικές του ιδιότητες αλλά υπάρχουν και μερικές ιδιότητες που είναι κοινές σε όλα τα αντικείμενα. Οι κοινές αυτές ιδιότητες είναι οι παρακάτω:

Name:	Το όνομα του αντικειμένου. Αυτό χρησιμοποιούμε για να αναφερθούμε στο αντικείμενο όταν γράφουμε κώδικα.
Caption & **Text**:	Τα περισσότερα αντικείμενα έχουν μια ιδιότητα CAPTION ή TEXT όπου είναι η περιγραφή τους, αυτό δηλαδή που θα φαίνεται στην οθόνη μας. Προσοχή! Μην μπερδεύετε το name με τα caption και text, το name είναι το χαρακτηριστικό όνομα του αντικειμένου όταν θέλουμε να αναφερθούμε σε αυτό μέσα από τον κώδικα μας, ενώ τα caption και text αλλάζουν το κείμενο της περιγραφής/ετικέτας του αντικειμένου ή το αρχικό κείμενο τους (text).

Appearance:	Με την ιδιότητα αυτή μπορούμε να δώσουμε μια ψευδοτριδιάστατη υφή σε ένα αντικείμενο (3D) ή όχι (Flat).
Autoredraw:	Αν θα φρεσκάρετε αυτόματα το περιεχόμενο μιας φόρμας ή όχι
Border Style form:	Ουσιαστικά από εδώ ορίζουμε αν η φόρμα μπορεί να μετακινηθεί, αν είναι πάντα πάνω από τις υπόλοιπες (modal), αν μπορεί να αλλάξει μέγεθος κλπ. Στα υπόλοιπα αντικείμενα που έχουν αυτήν την ιδιότητα καθορίζει αν θα υπάρχει περίγραμμα ή όχι γύρω από το αντικείμενο. Αντίστοιχο είναι το BORDERSTYLE των Textboxes όπου δηλώνει αν θα υπάρχει περιθώριο ή όχι.
Bordercolor:	Το χρώμα του περιγράμματος, π.χ. αν πρόκειται για ένα ορθογώνιο shape, αν θέλουμε να έχει γύρω γύρω μια κόκκινη γραμμή πάχους 2 pixels, τότε διαλέγουμε το κόκκινο χρώμα από το BorderColor και το BorderWidth = 2 για το πάχος της γραμμής.
Control box :	Αν θα εμφανίζεται ή όχι το κουμπί ελέγχου πάνω αριστερά στο παράθυρο της φόρμας.
Left:	Η απόσταση ενός αντικειμένου από το τέρμα αριστερά της φόρμας, ή της ίδιας της φόρμας από την επιφάνεια εργασίας.
Top:	Η απόσταση ενός αντικειμένου από το πάνω αριστερό άκρο (κορυφή) της φόρμας, ή της ίδιας της φόρμας από την επιφάνεια εργασίας. Ουσιαστικά το LEFT καθορίζει το [X] οριζόντιο ενός αντικειμένου ενώ το TOP το [Y] δηλαδή το κάθετο. Αλλάζοντας όσο εκτελείται το πρόγραμμα την τιμή των Left και Top δημιουργούμε το εφέ της κίνηση ενός αντικειμένου. Έτσι σε μια φόρμα το πάνω αριστερά σημείο της είναι το 0,0. Αν το συνολικό της πλάτος (width) είναι 6855 twips και εμείς θέλουμε να βάλουμε ένα κουμπί στο κέντρο της, θα πρέπει να το τοποθετήσουμε στα μισά του συνολικού πλάτους της δηλαδή το left του αντικειμένου περίπου στα 3427.
Width/height:	Πλάτος και ύψος του αντικειμένου σε μονάδες μέτρησης που έχουμε ορίσει από την φόρμα. (π.χ. twips, pixels, centimeters, inches κλπ). Μπορούμε να αλλάξουμε την μονάδα μέτρησης αν αλλάξουμε την ιδιότητα SCALEMODE της φόρμας μας.)
Key preview:	Ορίζεται από τις ιδιότητες της φόρμας και ουσιαστικά επιτρέπει ή όχι την χρήση του πληκτρολογίου και τον συνδυασμών πλήκτρων από αυτό.
Min/Max	Αν θα εμφανίζονται τα πλήκτρα ελαχιστοποίησης, μεγιστοποίησης σε μια φόρμα.
Picture Image Icon:	Αλλάζουν / εμφανίζουν (φορτώνουν) μια εικόνα ή ένα γραφικό ή ένα εικονίδιο (icon)
Enabled:	Αν το αντικείμενο θα είναι ενεργό ή όχι (παίρνει τιμές TRUE ή FALSE). Δηλαδή αν σε ένα command button του δώσουμε αρχική τιμή ENABLED=false τότε αυτό το κουμπί ΔΕΝ μπορεί να πατηθεί από τον χρήστη μέχρι να αλλάξει η τιμή του ENABLED σε TRUE.
Visible:	Αν θα εμφανίζεται ή όχι (παίρνει τιμές TRUE ή FALSE) το αντικείμενο όταν τρέξει το πρόγραμμα. Αλλάζοντας την ιδιότητα αυτή σε false κρύβουμε κάποιο αντικείμενο.
Font:	Ορίζει τους χαρακτήρες γραφής (γράμματα) του κειμένου ενός αντικειμένου. Αυτή η ιδιότητα μας επιτρέπει να αλλάξουμε την γραμματοσειρά του αντικειμένου καθώς και αν θα είναι έντονο, πλάγιο ή υπογραμμισμένο, το μέγεθος της κλπ
Forecolor/ Backcolor:	Καθορίζουν το χρώμα των γραμμάτων και του φόντου του αντικειμένου.
FillColor:	Το χρώμα γεμίσματος π.χ. σε ένα αντικείμενο shape κύκλο, αν διαλέξουμε κίτρινο χρώμα ως fillcolor και το fillstyle του ως solid, θα δούμε τον κύκλο να γεμίζει με κίτρινο χρώμα.

FillStyle:	Με ποιόν τρόπο θα γεμίσει με χρώμα ένα αντικείμενο. Solid σημαίνει πως θα γεμίσει ολόκληρο με το επιλεγμένο χρώμα, ενώ transparent σημαίνει πως είναι διάφανο (αυτό είναι και το default).
Mouseicon/ Mousepointe:	Καθορίζουν το σχήμα του δείκτη όταν περνάει πάνω από το αντικείμενο μας. Εκτός από τα προκαθορισμένα σχήματα δείκτη, μπορούμε να φορτώσουμε και δικά μας αλλάζοντας την τιμή του Mousepointer σε custom και φορτώνοντας τον δείκτη που θέλουμε από το mouseicon.
T**ooltip**:	ράφοντας ένα περιγραφικό κείμενο στην ιδιότητα tooltip ενός αντικειμένου εμφανίζεται μια επεξήγηση όταν περνάει ο δείκτης από πάνω του όσο εκτελείτε το πρόγραμμα έτσι ώστε να μπορεί να καταλάβει ο χρήστης περί τίνος πρόκειται και γενικότερα τι περιμένουμε να κάνει στο συγκεκριμένο αντικείμενο.

1.5 Μέθοδοι της **Visual Basic**

Μία μέθοδος είναι μια διαδικασία (υποπρόγραμμα) που εκτελεί μια συγκεκριμένη λειτουργία για κάποιο συγκεκριμένο αντικείμενο. Η διαδικασία αυτή είναι προκαθορισμένη, παρέχεται από τη Visual Basic έτοιμη για χρήση από μας και αφορά και ενεργεί πάνω σε κάποιο συγκεκριμένο αντικείμενο. Την εκτελούμε γράφοντας το όνομα του αντικειμένου για το οποίο θα εκτελεστεί αυτή η διαδικασία, ακολούθως η τελεία "." και έπειτα το όνομα της μεθόδου. Π.χ ο κώδικας–εντολή Formbasic.Refresh θα εκτελέσει πάνω στο αντικείμενο με όνομα Formbasic (που είναι μια φόρμα) την μέθοδο Refresh. Κάθε μέθοδος εκτελεί μια συγκεκριμένη εργασία. Στην προκειμένη περίπτωση η μέθοδος Refresh ανανεώνει στη οθόνη, επανασχεδιάζει την φόρμα Formbasic.

1.6 Συμβάντα στη **Visual Basic**

Συμβάν είναι κάθε γεγονός που προκαλείται κυρίως από τον χρήστη αλλά και από την λειτουργία και κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Από τον χρήστη συμβάν προκαλείται όταν πατήσει οποιοδήποτε πλήκτρο, κάνει κλικ, διπλό κλικ με το ποντίκι κ.α. Κατά την εκτέλεση-λειτουργία ενός προγράμματος συμβάν προκαλείται όταν π.. φορτώνεται στην μνήμη ή ανοίγει μια φόρμα, όταν ενεργοποιείται επιλέγοντάς την, όταν κλείνει κ.α.

Ένα συμβάν συμβαίνει σε κάποιο συγκεκριμένο αντικείμενο και όχι αόριστα, κυρίως όταν η εστίαση των Windows βρίσκεται σ' αυτό. Άρα προσδιορίζεται, όπως και μια ιδιότητα από το όνομα του αντικειμένου όπου σχετίζεται το συμβάν και από το προκαθορισμένο όνομα του συμβάντος. Π.χ κάθε φορά που ένας χρήστης πιέζει οποιοδήποτε πλήκτρο, ενώ βρίσκεται η εστίαση του προγράμματος στο αντικείμενο με όνομα Textsave, συμβαίνει το συμβάν Textsave_Keypress. Παρατηρούμε ότι ο χαρακτήρας (_) ανάλογα της (.) χρησιμοποιείται ως σύμβολο για την ένωση του ονόματος ενός αντικειμένου και του συμβάντος του. Το τμήμα κώδικα που επεξεργάζεται το κάθε συμβάν ονομάζεται event procedure.

Μερικά από τα συνηθισμένα συμβάντα της Visual Basic είναι τα Click, DblClick, Load, KeyPress, MouseDown, MouseUp, MouseMove κ.α.

2. Ανάπτυξη Εφαρμογών στη **Visual Basic**

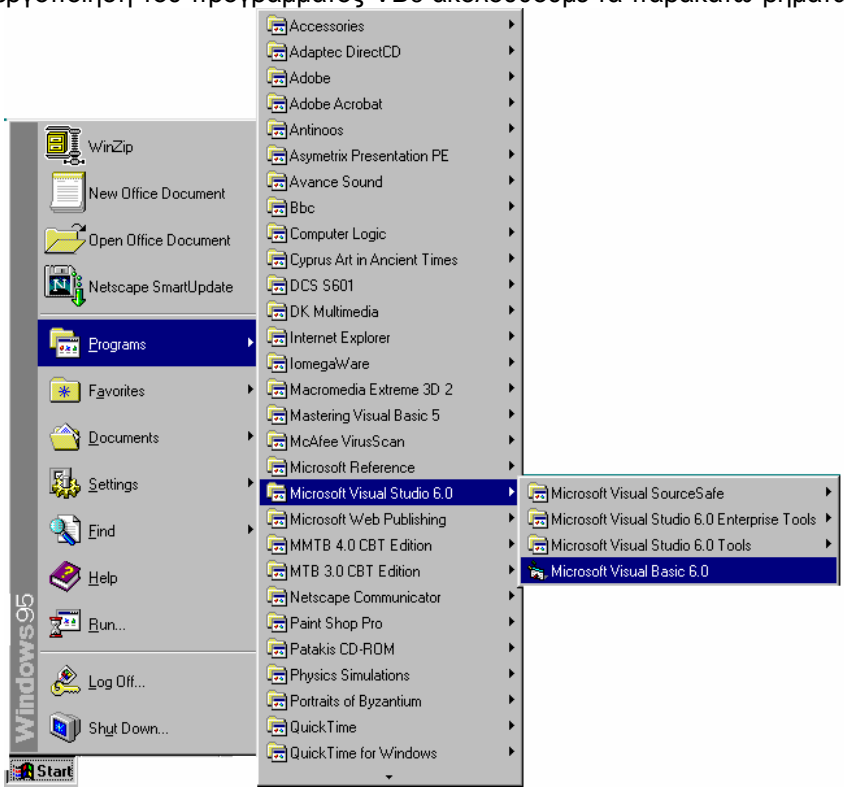
Η φιλοσοφία ανάπτυξης μιας εφαρμογής διαφοροποιείται ανάλογα με το περιβάλλον μέσα στο οποίο πρόκειται να εργαστεί. Στο περιβάλλον των Windows ο προγραμματιστής πρέπει να έχει πάντοτε υπόψη του ότι πρόκειται να εργαστεί μέσα σε ένα γραφικό περιβάλλον, άρα θα σχεδιάσει την εφαρμογή του βλέποντας την πάντα από τη μεριά του χρήστη. Επίσης, εκμεταλλευόμενος τα πλεονεκτήματα σχεδιασμού που προσφέρει το γραφικό περιβάλλον εργασίας των Windows πρέπει να δημιουργήσει ένα φιλικό και λειτουργικό τρόπο επικοινωνίας χρήστη-εφαρμογής.

Ο σχεδιασμός της εφαρμογής ξεκινάει πάντα από τη δημιουργία του τρόπου χρήστη-εφαρμογής, αφού η ροή εκτέλεσης εξαρτάται από τις αντιδράσεις του χρήστη. Ο τρόπος επικοινωνίας χρήστη-εφαρμογής αποτελεί το **user interface** της εφαρμογής. Γενικά, η ανάπτυξη μιας εφαρμογής με τη Visual Basic αποτελεί μια διαδικασία 3 βημάτων :

1. Σχεδιασμός του τρόπου επικοινωνίας χρήστη-εφαρμογής (διεπαφή, user interface).
2. Καθορισμός της αρχικής συμπεριφοράς των αντικειμένων της εφαρμογής μέσω των ιδιοτήτων τους.

3. Δημιουργία και αποσφαλμάτωση των διαφόρων ρουτινών κώδικα.

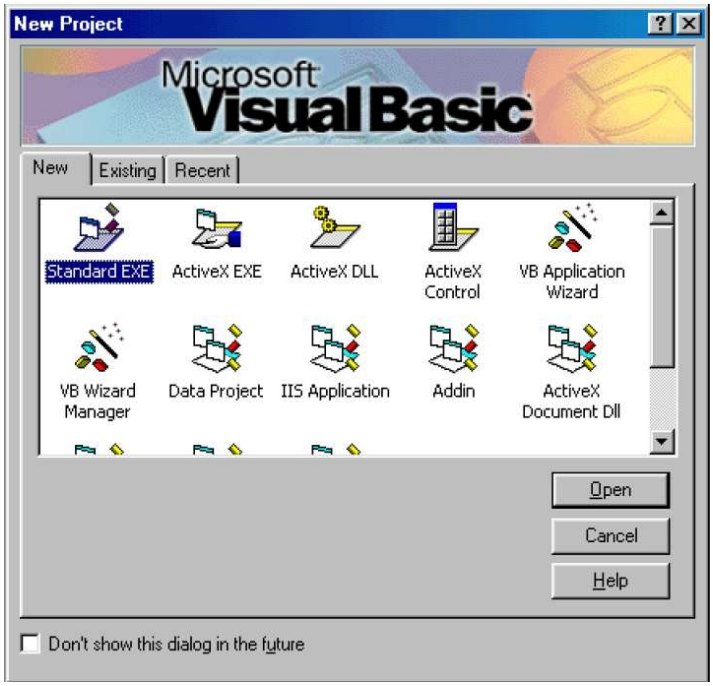
Για την ενεργοποίηση του προγράμματος VB6 ακολουθούμε τα παρακάτω βήματα:



Σχήμα: 1

2.1 Το Περιβάλλον Ανάπτυξης της **Visual Basic**

Όταν εκκινούμε την Visual Basic για πρώτη φορά, ανοίγει ο οδηγός Project Wizard και θα δούμε το παράθυρο διαλόγου νέου έργου New Project, όπως φαίνεται παρακάτω.



Σχήμα: 2

Το παράθυρο αυτό θα εμφανίζεται κάθε φορά που θα εκκινούμε την Visual Basic.

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

1. New: Επιλέγουμε τον τύπο του προγράμματος που θα δημιουργήσουμε «από το μηδέν».

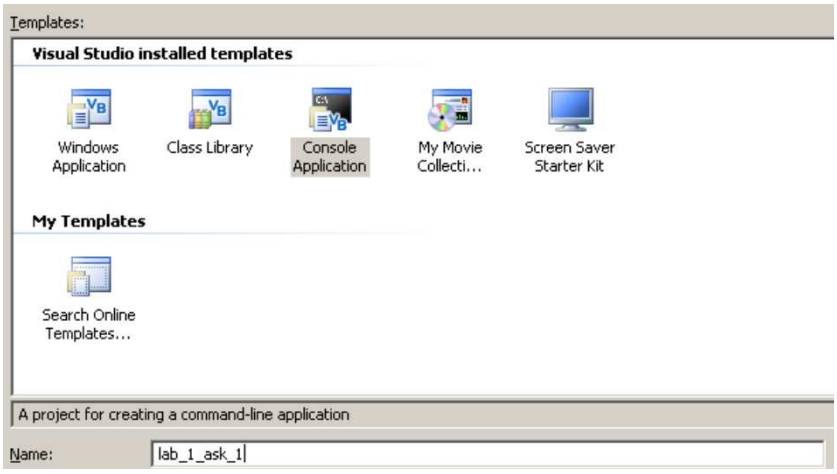
2. Existing: Ψάχνουμε στον δίσκο να εντοπίσουμε και να ανοίξουμε ένα ήδη δημιουργημένο και αποθηκευμένο πρόγραμμα (πρόσφατο/recent, ή παλαιότερο).

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.

Επιλέγοντας να δουλέψουμε σε Standard περιβάλλον δουλεύουμε πάντα κάποιο Project μέσα στο οποίο υπάρχουν μία ή και περισσότερες φόρμες, ή Modules που είναι λειτουργικές μονάδες που περιέχουν δηλώσεις μεταβλητών κοινά χρησιμοποιούμενες συναρτήσεις, διαδικασίες κλπ. Πάνω δεξιά στο παράθυρο βλέπουμε τα στοιχεία του Project το οποίο δουλεύουμε σε δενδροειδή μορφή. Στην συγκεκριμένη περίπτωση στον φάκελο των φορμών υπάρχει μία φόρμα, η Form1, που μπορούμε να δούμε–επιλέξουμε κάνοντας κλικ στο μεσαίο κουμπί εντολής (View Object). Μπορούμε να δούμε όλες τις διαδικασίες που έχουν γραφτεί (κώδικας) κάνοντας κλικ στο αριστερό κουμπί (View Code).

Όσον αφορά το νεότερο περιβάλλον της Visual Basic 2005 Express Edition, αυτό διατίθεται ελεύθερα στο διαδίκτυο στην ηλεκτρονική διεύθυνση: http://msdn.microsoft.com/vstudio/express/vb/download/

Σε αυτή την περίπτωση η εκκίνηση της Visual Basic γίνεται από το εικονίδιο «Microsoft Visual Basic 2005 Express Edition» (αν υπάρχει στην επιφάνεια εργασίας ή από την επιλογή του μενού "Έναρξη/Όλα τα προγράμματα/Microsoft Visual Basic 2005 Express Edition)".



Σχήμα: 3 Καθορισμός ονόματος και τύπου εφαρμογής

Η δημιουργία μια νέας εφαρμογής της Visual Basic, γίνεται με την επιλογή «New Project» του μενού «File». Εναλλακτικά μπορούμε να χρησιμοποιηθεί η συντόμευση Ctrl-N.

Στην συνέχεια εμφανίζεται το πλαίσιο διαλόγου (Σχήμα 3) με τον Τίτλο «New Project». Με την επιλογή «Console Application» στο πάνω παράθυρο του πλαισίου διαλόγου με τον τίτλο «Template» δημιουργείται μια νέα εφαρμογή τύπου «Console Application». Ο τύπος της εφαρμογής αυτής είναι ο απλούστερος τύπος εφαρμογής και χρησιμοποιείται όταν δεν χρειάζεται το γραφικό περιβάλλον και οι φόρμες των Windows για την επικοινωνία (διεπαφή) με τον χρήστη.

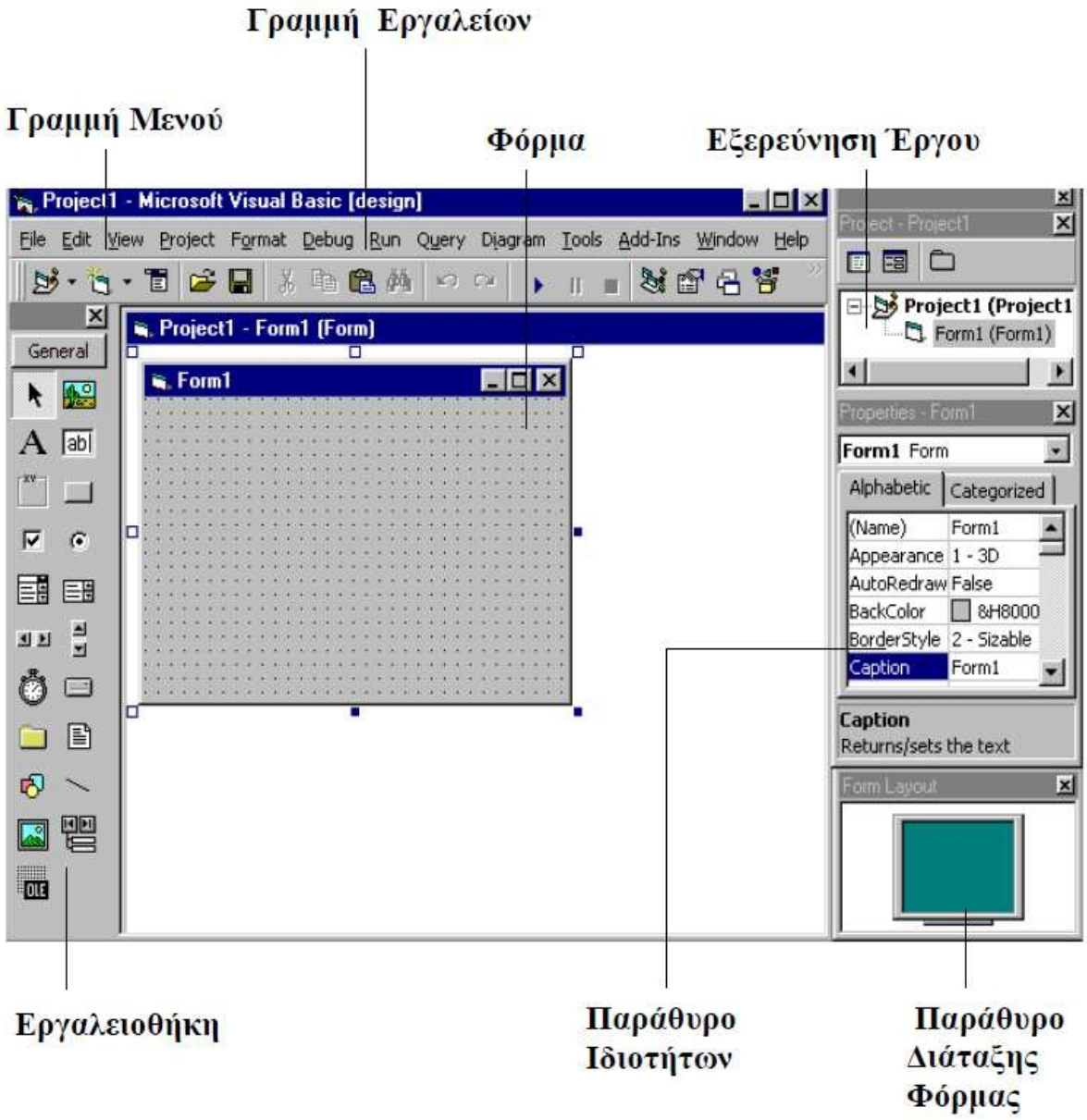
Εμείς τώρα θα συνεχίσουμε να εργαζόμαστε στο Standard περιβάλλον και ξεκινώντας ένα νέο project, αυτόματα δημιουργείται ένα αντικείμενο φόρμας με όνομα Form1, και παρακάτω, στο σχήμα 4, βλέπουμε το παράθυρο ιδιοτήτων της φόρμας (Properties – Form1), όπως και κάθε αντικειμένου που έχουμε επιλέξει. Κάτω από τον τίτλο του παραθύρου ιδιοτήτων βλέπουμε το όνομα του αντικειμένου και το τι τύπος αντικειμένου είναι (Form1 Form) ή θα μπορούσε να είναι (Text1 text).

Επίσης, βλέπουμε όλες τις ιδιότητες του αντικειμένου αλφαβητικά ή ομαδοποιημένες ανάλογα με την λειτουργία τους. Δηλαδή όλες όσες αφορούν την συμπεριφορά (Behavior), Θέση (Position) κ.α. Μπορούμε να αλλάξουμε μία ιδιότητα από το παράθυρο ιδιοτήτων ή όπως προείπαμε κατά τον χρόνο εκτέλεσης με κώδικα.

Κάτω δεξιά βλέπουμε την θέση της φόρμας σε σχέση με την οθόνη στο παράθυρο Form Layout.

Αριστερά υπάρχει το παράθυρο toolbox (ενεργοποιείται από το μενού Προβολή), το οποίο περιέχει τα αντικείμενα που μπορούμε να εισάγουμε σε φόρμες.

Αφού επιλέξουμε με το ποντίκι κάποιο αντικείμενο το εισάγουμε σχεδιάζοντάς το πάνω σε μια φόρμα.



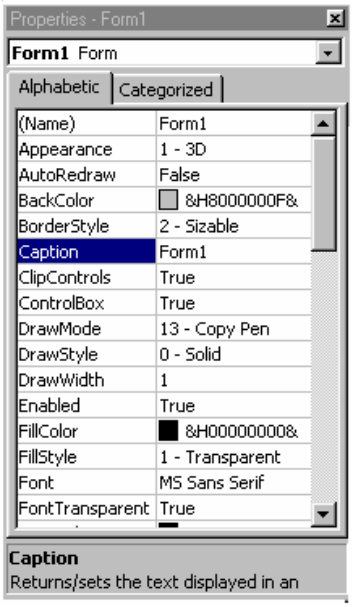
Σχήμα: 4

2.2 Η εργαλειοθήκη **Toolbox** της **Visual Basic**

Σχήμα: 5

Η εργαλειοθήκη της Visual Basic, όπως φαίνεται παρακάτω στο σχήμα 5, περιέχει μια συλλογή αντικειμένων-εργαλείων τα οποία χρησιμοποιούνται κατά κύριο λόγο στις εφαρμογές.

Ακολουθεί και το πλαίσιο των ιδιοτήτων (σχήμα 6) , το οποίο αλλάζει ανάλογα με το αντικείμενο που επιλέγουμε:



Σχήμα: 6

Τα εργαλεία αυτά περιγράφονται παρακάτω :

Pointer/δείκτης	Το μοναδικό αντικείμενο από την εργαλειοθήκη της VB που δεν εισάγει κάποιο αντικείμενο στην φόρμα. Όταν το επιλέγετε μπορείτε να αλλάξετε το μέγεθος ή να μετακινήσετε ένα αντικείμενο πάνω στην φόρμα.
Labels/ετικέτες	Καθαρά περιγραφικά αντικείμενα όπου τα χρησιμοποιούμε συνήθως δίπλα σε κάποια άλλα αντικείμενα με ένα επεξηγηματικό κείμενο, έτσι ώστε ο χρήστης να καταλαβαίνει τι πρέπει να κάνει. Δηλαδή ένα label δίπλα σε ένα text box για παράδειγμα που έχει ως caption « Παρακαλώ γράψτε το όνομα σας» δηλώνει στον χρήστη πως πρέπει να συμπληρώσει το όνομα του μέσα στο text box. Αλλάζοντας την ιδιότητα autosize σε true αυτόματα το μέγεθος της ετικέτας γίνετε ίσο με το μέγεθος του κειμένου που περιέχει. Αν αλλάξουμε το BackStyle της ετικέτας σε Opaque την κάνουμε αυτόματα "διάφανη", δηλαδή δεν είναι βαμμένη με κάποιο χρώμα αλλά φαίνεται από πίσω το χρώμα ή η εικόνα της φόρμας. Με το WordWarp = true αν δεν χωράει το κείμενο να εμφανιστεί στην ετικέτα γίνετε αυτόματη αλλαγή σειράς.
Text Boxes /κουτιά κειμένου	Τα text boxes δεν είναι τίποτε περισσότερο από αυτό που δηλώνει και το όνομα τους, δηλαδή απλά «κουτιά κειμένου». Πεδία δηλαδή όπου ο χρήστης μπορεί να γράψει μέσα τους κείμενο. Ιδιότητες άξιες προσοχής εκτός από τις συνηθισμένες που έχουμε περιγράψει ήδη είναι : **Aligment**: (στοίχιση κειμένου) όπου μπορεί να είναι αριστερή, κέντρο ή δεξιά. **Locked**: Όπου καθορίζει αν είναι κλειδωμένο ή όχι το αντικείμενο (δηλαδή αν μπορούμε να γράψουμε ή όχι σε αυτό… κάτι αντίστοιχο με το ENABLED) **MaxLength**: Όπου καθορίζει το μέγιστο αριθμό χαρακτήρων που μπορεί να καταχωρήσει (γράψει) ο χρήστης μέσα σε αυτό το αντικείμενο. **MultiLine**: (πολλές γραμμές). Ορίζοντας αυτήν την ιδιότητα ως αληθές (true) δηλώνουμε πως το κείμενο μπορεί να περιέχει πολλές γραμμές (να είναι

	δηλαδή ακόμα και ολόκληρο κείμενο και όχι μια απλή γραμμή κειμένου) Αν ορίσουμε το MultiLine ως true καλό είναι να μεγαλώσουμε και το ύψος του και να ορίσουμε και το Scrollbars σε Horizontal – vertical ή both για να εμφανίζεται μια οριζόντια, κάθετη ή και οι δύο γραμμές ολίσθησης αντίστοιχα, όταν ΔΕΝ χωράει (δεν εμφανίζεται) ολόκληρο το κείμενο μέσα στο text box. **Passwordchar**: μετατρέπει το text box σε πεδίο κειμένου όπου εμφανίζει αστεράκια όσο γράφουμε σε αυτό για λόγους ασφαλείας (έτσι ώστε αυτός που κάθεται για παράδειγμα από πίσω μας και μας βλέπει, να μην μπορεί να δει στην οθόνη τον κωδικό που γράφουμε)
 **Buttons /**Κουμπιά	Ένα από τα πιο «κοινά» αντικείμενα είναι τα κουμπιά η command buttons. Εκτός των κλασικών ιδιοτήτων περιγραφής, χρωμάτων και θέσης του αντικειμένου στην φόρμα, μπορούμε αν θέλουμε να του δώσουμε και την λεγόμενη «υφή» ή με απλά λόγια ένα κουμπί να έχει ως φόντο μια εικόνα. Για να το πετύχουμε αυτό αλλάζουμε την ιδιότητα **Style** από Standard σε Graphical και ορίζουμε ένα αρχείο εικόνας στην ιδιότητα **Picture**. Τον κώδικα για αυτά τα κουμπιά τον γράφουμε τις περισσότερες φορές στο **onclick** τους, δηλαδή περιγράφουμε με κώδικα, τι θα γίνει όταν κάποιος χρήστης κάνει κλικ σε κάποιο κουμπί.
 **Option Buttons /** κουμπιά επιλογών	Χρησιμοποιούμε τα option buttons όταν θέλουμε ο χρήστης να επιλέξει μια τιμή ανάμεσα σε Χ επιλογές. π.χ. Τοποθετώντας τρία option buttons σε μια φόρμα και με περιγραφή Πρώτη, δεύτερη, τρίτη στα caption τους αντίστοιχα βλέπουμε πως όταν εκτελέσουμε το πρόγραμμα αν πάμε να κάνουμε κλικ π.χ. στην επιλογή «δεύτερη» και μετά κλικ στην «τρίτη» μόνο μια επιλογή παραμένει επιλεγμένη. Αυτό σημαίνει πως δεν μπορούμε να κάνουμε πολλαπλή επιλογή όταν χρησιμοποιούμε τα option buttons, αν θέλουμε κάτι τέτοιο χρησιμοποιούμε τα CHECK BOXES.
 **Check boxes /** κουτιά ελέγχου	Κάτι αντίστοιχο με τα option buttons είναι και τα check boxes με την διαφορά ότι αυτά μας επιτρέπουν πολλαπλή επιλογή. Μπορούμε δηλαδή να τσεκάρουμε και να αποεπιλέξουμε όποια από αυτά θέλουμε. Για να δούμε με κώδικα αν είναι επιλεγμένο ή όχι σε κάποιο από αυτά τα αντικείμενα, πρέπει να ελέγξουμε την τιμή του (value) Έτσι αν π.χ. πούμε msgbox(Check1.Value) θα μας εμφανίσει 1 ή true (αληθές) αν είναι τσεκαρισμένο και 0 αν είναι false (ψευδές) – ξετσεκαρισμένο. Και αυτά όπως και τα command buttons και τα checkboxes αν αλλάξουμε την ιδιότητα τους **style** σε graphical μπορούμε να τα δώσουμε ως «φόντο» μια εικόνα την οποία την ορίζουμε από την ιδιότητα picture.
 **Frames/**πλαίσια	Τα frames τα χρησιμοποιούμε συνήθως για να διαχωρίζουμε αντικείμενα στην οθόνη του προγράμματος ή και να τα ομαδοποιούμε. Είναι θα λέγαμε περισσότερο για «ομορφιά» και διακόσμηση της φόρμας μας, και όχι για κάτι ουσιαστικό ή σημαντικό. Πρέπει πρώτα να τοποθετήσετε το frame στην φόρμα και μετά μέσα σε αυτό τα υπόλοιπα αντικείμενα για να πετύχετε την ομαδοποίηση.
 **Combo Boxes/** κουτιά επιλογών	Χρησιμοποιούμε τα combo boxes για εύκολη επιλογή συγκεκριμένων τιμών (λίστας τιμών) που τα έχουμε ορίσει ή τα ορίζουμε (προσθέτουμε) μέσω κώδικα. Δηλαδή, αν στην σχεδίαση φόρμας, δώσουμε σε ένα combo box στην ιδιότητα **LIST** της τιμές Γιώργος και Νίκος μόλις εκτελέσουμε το πρόγραμμα με κλικ στο βελάκι του combo box θα μας εμφανίσει αυτές τις δυο τιμές και εμείς μπορούμε με κλικ να επιλέξουμε μια από τις δυο. Η εντολή **selectedindex** μας επιστρέφει τον αριθμό σειράς (index) της επιλογής μας σε ένα combobox. Για να προσθέσουμε με κώδικα σε combo box χρησιμοποιούμε την μέθοδο **ADDITEM** και για να αφαιρέσουμε την **REMOVEITEM**. Η σύνταξη τους έχει ως εξής: [combo box name].additem ([μεταβλητή ή αντικείμενο]) π.χ. Combo2.AddItem x και η [combo box name].removeitem ([index]) όπου index ένας αριθμός που αντιστοιχεί στις τιμές από το 0 μέχρι Χ που περιέχει το combo box. Δηλαδή αν περιέχει όπως στο παράδειγμα μας τις τιμές Γιώργος και Νίκος, το index μηδέν είναι το Γιώργος και το index ένα είναι το Νίκος… μια αρίθμηση δηλαδή από το μηδέν μέχρι να

	μετρήσει όλες τις τιμές που περιέχονται στο combo box. Αν αλλάξουμε την ιδιότητα **SORTED** σε true τότε τα περιεχόμενα του Combo box θα ταξινομούνται αυτόματα. Τα combo boxes έχουν ύψος ΜΟΝΟ μια γραμμή και για να εμφανίσουν τα περιεχόμενα τους χρησιμοποιούμε το βελάκι που δείχνει προς τα κάτω που έχουν στην δεξιά γωνία τους. Αλλάζοντας το **STYLE** του combo box σε SIMPLE COMBO μπορούμε να αυξήσουμε το ύψος του combo και πάλι μας δίνεται η δυνατότητα να γράφουμε όπως γίνετε και στο dropdown combo σε αντίθεση με την 3η επιλογή την Dropdown list όπου μπορούμε ΜΟΝΟ να επιλέξουμε από την λίστα επιλογών του combo και όχι να γράψουμε (ή αναζητήσουμε) τιμές.
List boxes/ λίστες	Τα list boxes είναι σχεδόν όμοια με τα combo boxes με την διαφορά ότι δεν έχουν βελάκι που δείχνει προς τα κάτω για την εμφάνιση των περιεχομένων του αντικειμένου, αλλά μπορούν να έχουν συγκεκριμένο ύψος και να βγάζουν γραμμές ολίσθησης όταν χρειάζεται για να εμφανίσουν τα περιεχόμενα τους. Τα list boxes επίσης μας δίνουν την δυνατότητα να έχουμε περισσότερες του ενός στήλες αλλάζοντας την ιδιότητα τους με όνομα **COLUMNS**. Μπορούμε αν θέλουμε να αλλάξουμε την ιδιότητα **STYLE** σε checkboxes επιτρέποντας μας έτσι να τσεκάρουμε τις επιλογές που θέλουμε να επιλέξουμε από την λίστα, πράγμα που σημαίνει πως μπορούμε να κάνουμε πολλαπλή επιλογή.
VSrollBars	(Vertical scrollbars - Οριζόντιες γραμμές ολίσθησης)
HSrollBars	(Horizontal scrollbars - Κάθετες γραμμές ολίσθησης) Πρόκειται δηλαδή για τις γνωστές γραμμές ολίσθησης που μεταβάλουν την τιμή τους ανάμεσα στα όρια που μπορούμε να τους ορίσουμε εμείς, αλλάζοντας τις ιδιότητες: **SmallChange** (το βήμα με το οποίο αυξάνεται η τιμή της γραμμής ολίσθησης όταν ο χρήστης κάνει κλικ στα βελάκια της) **LargeChange** (το βήμα με το οποίο αυξάνεται η τιμή της γραμμής ολίσθησης όταν ο χρήστης κάνει κλικ μέσα στην γραμμή ολίσθησης ή κρατώντας το «κουτάκι» και μετακινώντας το.) **Min** (η τιμή από την οποία ξεκινάει το μέτρημα – αρχική μικρότερη τιμή δηλαδή) **Max** (η τιμή από στην οποία τελειώνει το μέτρημα – τελική μέγιστη τιμή)
Timers / χρονόμετρα	Με την χρήση αυτού του αντικειμένου μπορούμε να εκτελούμε συγκεκριμένο κώδικα με το πέρασμα ορισμένου χρονικού ορίου. Δηλαδή μπορούμε παράδειγμα να εμφανίζουμε ένα μήνυμα κάθε ένα δευτερόλεπτο ή να κινούμε ένα αντικείμενο αλλάζοντας τις ιδιότητες **left** και **top** του κάθε [Χ] δευτερόλεπτα. Για να καθορίσουμε κάθε πότε θα επαναλαμβάνεται ο κώδικας Timer() που θα γράψουμε στο function πρέπει να έχουμε δώσει μια τιμή στην ιδιότητα **INTERVAL** του που αντιστοιχεί σε χιλιοστά του δευτερολέπτου (milliseconds) . Δηλαδή αν δώσουμε interval 1000 ο κώδικας που θα έχουμε γράψει π.χ. στο Timer1_Timer() θα επαναλαμβάνεται κάθε 1 δευτερόλεπτο. Αν λοιπόν αυξάνουμε την ιδιότητα left π.χ. ενός αντικειμένου κατά 5 παράδειγμα twips ανά 1000 interval (1 δευτ.) τότε θα το αντικείμενο αυτό θα φαίνεται πως κινείτε μόνο του προς τα δεξιά. Το αντικείμενο αυτό ΔΕΝ εμφανίζεται όσο τρέχει το πρόγραμμα, φαίνεται μόνο στην σχεδίαση της φόρμας
Shapes / σχήματα	Χρησιμοποιούμε τα σχήματα για να «ομορφύνουμε» την εμφάνιση μια φόρμας ή ακόμα και για να δημιουργήσουμε εικόνες ή άλλα σχέδια. Η ιδιότητα που μας ενδιαφέρει περισσότερο είναι η **SHAPE** όπου καθορίζουμε αν το σχήμα είναι rectangle (παραλληλόγραμμο), square (τετράγωνο), oval (οβάλ), circle (κύκλος), rounder rectangle ή rounded square (δηλαδή τετράγωνο ή παραλληλόγραμμο με στρογγυλεμένες άκρες)
Lines/γραμμές	Χρησιμοποιούμε το συγκεκριμένο αντικείμενο για να εμφανίζουμε οριζόντιες, κάθετες ή διαγώνιες γραμμές στην φόρμα μας (οθόνη μας)
File List Boxes	Με την χρήση αυτών των τριών αντικειμένων μπορούμε ουσιαστικά να δούμε τα περιεχόμενα ενός αποθηκευτικού μέσου (δίσκου / δισκέτα / CD-ROM κλπ)

/λίστες αρχείων **Drive List boxes/**λίστες δίσκων **Dir List Boxes /** λίστες καταλόγων	ακόμα και να επιλέξουμε κάποιο από αυτά. Το Drive list box μας δείχνει τα ονόματα των αποθηκευτικών μέσων (π.χ. Δίσκος C:, Δίσκος D: CD-ROM g: κλπ) Το Dir List Box μας δείχνει και μας αφήνει να κινηθούμε και να επιλέξουμε κάποιον κατάλογο από κάποιο αποθηκευτικό μέσω. Και τέλος το File List Box εμφανίζει και μας επιτρέπει να επιλέξουμε ένα αρχείο από κάποιον φάκελο. Tip: Αντί την χρήση των παραπάνω αντικειμένων μπορούμε να χρησιμοποιήσουμε και τα components common dialog της Microsoft ορίζοντας ως method fileopen ή filesave.
Picture Boxes /φωτογραφίες	Εμφανίζουν μια εικόνα / ζωγραφιά, φωτογραφία. Ορίζουμε ποια εικόνα θα εμφανίζουν από την ιδιότητα PICTURE. Στα picture boxes υπάρχει η ιδιότητα : **autosize** όπου αυτόματα αλλάζει το μέγεθος του αντικειμένου και το κάνει ίσο με την εικόνα που εμφανίζει. Τα picture boxes σε αντίθεση με τα images δουλεύουν ως containers (δοχεία) όπου μέσα σε αυτά μπορούμε να τοποθετήσουμε και να ομαδοποιήσουμε άλλα αντικείμενα. (όπως γίνετε και στο frame)
Data/Βάσεις δεδομένων	Ένα από τα «δυνατά» αντικείμενα της Visual basic είναι το data object που μας επιτρέπει να μετακινηθούμε στις εγγραφές μια βάσης δεδομένων χρησιμοποιώντας τα βελάκια που διαθέτει. Για να καταφέρουμε να φτάσουμε στο σημείο να εμφανιστούν οι εγγραφές μια βάσης δεδομένων στην οθόνη μας πρέπει να κάνουμε τα παρακάτω βήματα: 1. Τοποθετούμε ένα data object στην φόρμα μας 2. Από την ιδιότητα databasename του αντικειμένου βρίσκουμε και επιλέγουμε την βάση δεδομένων από τον δίσκο μας (access mdb file). Αν θέλετε μπορείτε να δημιουργήσετε την βάση σας στην Access 97 και να την προσπελάσετε από την VB. Καλό είναι να υπάρχει τουλάχιστον μια εγγραφή στην βάση γιατί αλλιώς μπορεί να σας βγάλει λάθος ο κώδικας 3. Διαλέγεται ποιο table της βάση δεδομένων θέλετε να εμφανίσετε στην φόρμα από την ιδιότητα RecordSource 4. Τοποθετείτε αντίστοιχα text boxes όσα και τα πεδία κειμένου που έχετε στο συγκεκριμένο table της βάσης δεδομένων, πάνω στην φόρμα και μετά ένα ένα αντιστοιχείτε το data source τους στο όνομα (name) του data object που έχετε στην φόρμα. (π.χ. data1) και σαν datafield το όνομα του πεδίου που θέλετε να εμφανίζει. Με αυτά τα τέσσερα απλά βήματα όταν τρέξετε το πρόγραμμα σας, θα μπορείτε με τα βελάκια του data να βλέπετε στα text boxes τις εγγραφές της βάσης δεδομένων σας. Γενικότερα όσα αντικείμενα έχουν datasource και datafield επιλογές, σημαίνει πως μπορούμε να τα αντιστοιχίσουμε με κάποια πεδία μια βάσης δεδομένων.
OLE	Μας επιτρέπει να συνδέσουμε ή ενσωματώσουμε ξένα αντικείμενα και άλλες εφαρμογές που έχουμε εγκατεστημένες στον υπολογιστή μας, στο πρόγραμμα μας. Έτσι για παράδειγμα μπορείτε να εισάγετε την ζωγραφική των windows μέσα στο πρόγραμμα σας. Σας εμφανίζει λίστα με τα αντικείμενα που μπορείτε να εισάγετε η οποία έχει να κάνει με τα εγκατεστημένα στον υπολογιστή σας προγράμματα.

Μετά την ανάπτυξη της εφαρμογής, τις δοκιμαστικές εκτελέσεις και την αποσφαλμάτωσή της, τελευταία βήμα πριν την διάθεσή της είναι η δημιουργία εκτελέσιμου αρχείου. Το δυαδικό αυτό αρχείο προέρχεται από τη μεταγλώττιση (compilation) του πηγαίου κώδικα που έχουμε συγγράψει και των διάφορων βιβλιοθηκών της γλώσσας και είναι τελείως ανεξάρτητο από το περιβάλλον ανάπτυξης που χρησιμοποιήσαμε.

Η δημιουργία του εκτελέσιμου αρχείου γίνεται από το μενού “File” και την επιλογή “Make projectNo.exe”. Κατόπιν ζητείται ο φάκελος και το όνομα του εκτελέσιμου αρχείου και μετά τη συμπλήρωση από το χρήστη των στοιχείων αυτών δημιουργείται το αρχείο.

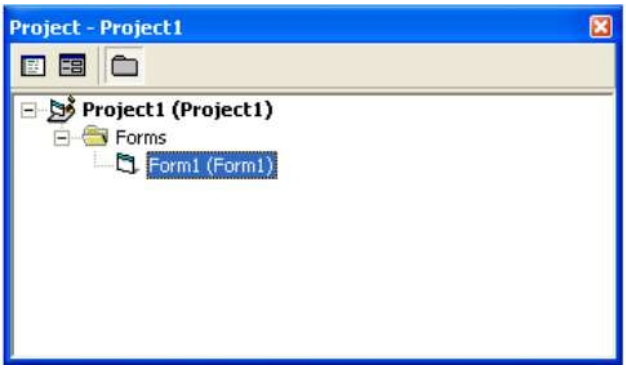
2.3 Το **Project** της **Visual Basic**

Μια εφαρμογή της Visual Basic απαρτίζεται από μια συλλογή φορμών, προγραμματιστικών μονάδων, αντικειμένων και ορισμένων ακόμη στοιχείων. Για να βοηθηθεί ο προγραμματιστής στη διαχείριση όλων αυτών των συστατικών που αποτελούν την εφαρμογή του, η Visual Basic εισήγαγε την έννοια του **Project**, το οποίο είναι ένα αρχείο κειμένου, σε ρόλου εργαλείου διαχείρισης, με επέκταμα .**VBP**, που περιέχει μια λίστα όλων των συστατικών μερών της εφαρμογής, χωρίς να περιέχει τα ίδια τα συστατικά αυτά. Το αρχείο αυτό ενημερώνεται αυτόματα κάθε φορά που αποθηκεύουμε το project. Κατά την μεταγλώττιση, όλα τα περιγραφόμενα αρχεία στο project δίνουν το τελικό δυαδικό αρχείο, τύπου .exe, .dll ή .ocx.

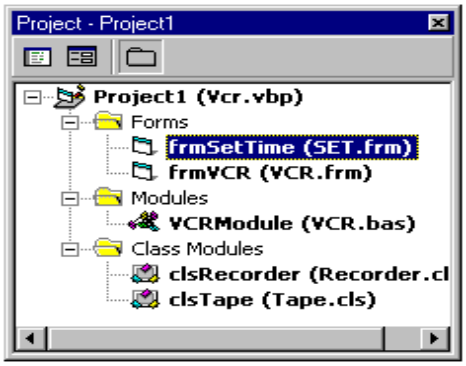
Τα περιεχόμενα ενός project οπτικοποιούνται στο περιβάλλον ανάπτυξης μέσω του παραθύρου του **Project Explorer**. Στο παράθυρο αυτό παρουσιάζονται με οπτικό τρόπο, ιεραρχημένα και με ξεχωριστό εικονίδιο τα διάφορα συστατικά του project.

Ο Project Explorer μας δείχνει από ποια συστατικά αποτελείται το project που φτιάχνουμε και μας επιτρέπει την μετάβαση από το ένα στο άλλο.

Όπως βλέπουμε στην επόμενη εικόνα, τα περιεχόμενα του εξερευνητή είναι οι φόρμες, τα modules, τα class modules κ.ά.



Σχήμα: 7



Σχήμα: 8

Οι βασικοί τύποι αρχείων που μπορεί να περιέχει ένα project (στο αρχείο .vbp) είναι :

- Αρχεία φόρμας (Form modules, .FRM). Περιλαμβάνουν τα οπτικά στοιχεία και τις ιδιότητες των φορμών και των εργαλείων που περιέχουν καθώς και τις σταθερές, μεταβλητές και ρουτίνες κώδικα που συνοδεύουν τη φόρμα.
- Δυαδικά Αρχεία , με επέκταμα .FRX, που δημιουργούνται αυτόματα για κάθε φόρμα που περιέχει γραφικά, κυρίως μέσω των ιδιοτήτων Picture και Icon. Τα αρχεία αυτά δεν είναι άμεσα επεξεργάσιμα από το χρήστη.
- Αρχεία προγραμματιστικών μονάδων (Standard Modules, .BAS). Περιέχουν καθαρές εντολές προγραμματισμού και χρησιμοποιούνται για δήλωση μεταβλητών, καθορισμό σταθερών και δημιουργία συναρτήσεων και υπο-ρουτίνων που καθορίζονται από το χρήστη.
- Αρχεία κλάσεων (Class Modules, .CLS). Είναι αρχεία κώδικα που χρησιμοποιούνται για τη δημιουργία καινούργιων αντικειμένων.
- Αρχεία πόρων (Resource file, .RES).
- Αρχεία **ActiveX** (.OCX).
- Αντικείμενα εγγράφων (Document Objects, .DOB)
- Σελίδες ιδοτήτων (Property Pages, .PAG)
- **ActiveX Designers** (.DSR)

Όλα τα περιεχόμενα ενός project μπορούν να χρησιμοποιηθούν και σε άλλα projects.

2.4 Προσθήκη και διαγραφή αρχείων στο **project**

Σε ένα project μπορούμε να προσθέτουμε και να διαγράφουμε διάφορα αρχεία. Για να προσθέσουμε ένα υπάρχον αρχείο (π.χ. μια φόρμα που έχουμε ήδη σχεδιάσει σε άλλο project) σε ένα project :

- Από το μενού **Project** επιλέγουμε μια από τις επιλογές **Add …….**, ανάλογα με το αρχείο που θέλουμε να εισάγουμε.
- Στο παράθυρο που θα εμφανιστεί επιλέγουμε το αρχείο που θέλουμε να εισάγουμε. Το αρχείο που επιλέξαμε θα εμφανιστεί στο παράθυρο του Project Explorer.

Για να διαγράψουμε ένα αρχείο από ένα project :

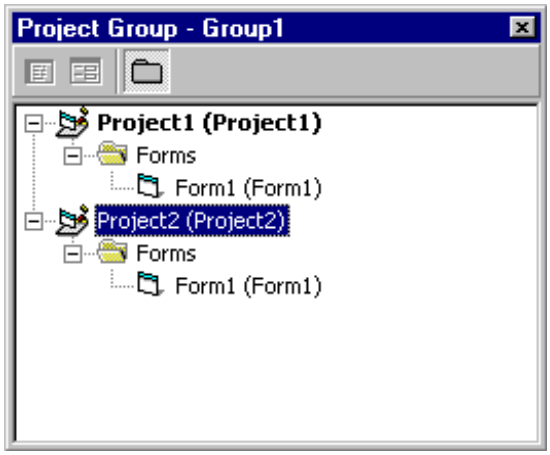
- Στο παράθυρο του Project Explorer επιλέγουμε το αρχείο που θέλουμε να διαγράψουμε.
- Από το μενού **Project** επιλέγουμε την επιλογή **Remove …….**

Το αρχείο διαγράφεται από το project που εργαζόμαστε. Δεν διαγράφεται όμως σαν αρχείο και μπορούμε να το χρησιμοποιήσουμε σε άλλα projects.

2.5 Προσθήκη και διαγραφή **projects**

Σε περιπτώσεις σύνθετων εφαρμογών, μπορούμε να χρησιμοποιούμε πολλά projects ταυτόχρονα, δημιουργώντας ένα **Project Group**.

Για να προσθέσουμε ένα νέο project στο project group που εργαζόμαστε:



Σχήμα: 9

- Από το μενού **File** επιλέγουμε **Add Project**.
- Στο παράθυρο που εμφανίζεται επιλέγουμε ένα υπάρχων project ή ένα καινούργιο.
- Το project που επιλέξαμε εμφανίζεται στο παράθυρο του Project Explorer.

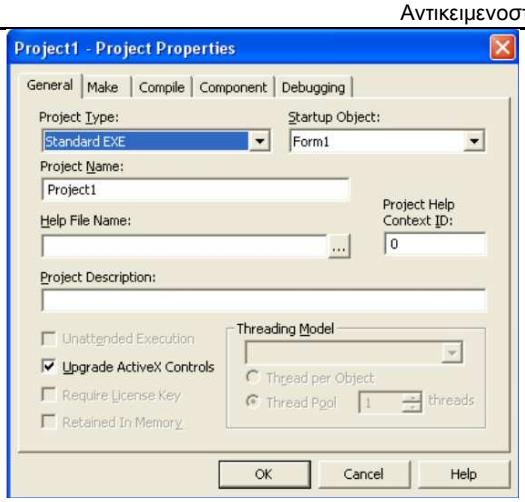
Για να διαγράψουμε ένα project :

- Στο παράθυρο του Project Explorer επιλέγουμε το project που θέλουμε να διαγράψουμε.
- Από το μενού **File** επιλέγουμε **Remove Project**

Το project διαγράφεται από το project που εργαζόμαστε. Δεν διαγράφεται όμως σαν αρχείο και μπορούμε να το χρησιμοποιήσουμε σε άλλα project groups.

2.6 Ιδιότητες του **project**

Η βασική και πιο σημαντική ιδιότητα ενός project είναι αυτή που προσδιορίζει το όνομα του, δηλαδή η Name. Εκτός από αυτή υπάρχουν και άλλες, όπως ο τίτλος της εφαρμογής, ο αριθμός έκδοσης μετά από κάθε μεταγλώττιση, το αρχείο βοηθείας που σχετίζεται με την εφαρμογή, το εικονίδιο που συμβολίζει την εφαρμογή στον Windows Explorer, μια περιγραφή της εφαρμογής, ρυθμίσεις σχετικά με την μεταγλώττιση κ.α.



Σχήμα: 10 Γενικές ιδιότητες ενός project

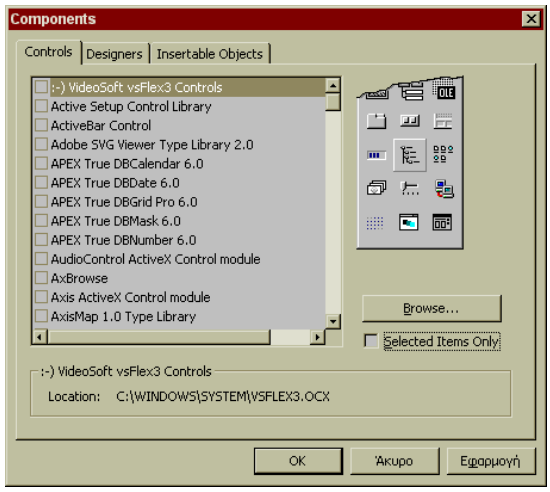
Όλες οι ιδιότητες ενός project μπορούν να προσπελαστούν κάνοντας δεξί κλικ πάνω στο όνομα του project.

Το ίδιο μπορεί να γίνει από το μενού **Project** και την επιλογή **Project Properties…** (κάνοντας τις απαραίτητες ρυθμίσεις στο παράθυρο που εμφανίζεται).

2.7 Προσθήκη εργαλείων σε **project**

Εκτός από τα εργαλεία της εργαλειοθήκης που υπάρχουν σε κάθε project, μπορούν να προστεθούν και άλλα, περισσότερο εξειδικευμένα και ανάλογα με την εφαρμογή ActiveX στοιχεία ελέγχου.

Για να προσθέσουμε ActiveX στοιχεία ελέγχου σε ένα project :



Σχήμα: 11

- ✓ Από το μενού **Project** επιλέγουμε **Components**…
- ✓ Στο παράθυρο που εμφανίζεται επιλέγουμε την καρτέλα **Controls**
- ✓ Εαν το πλαίσιο ελέγχου **Selected Items Only** είναι τσεκαρισμένο, αφαιρούμε το σημάδι τσεκαρίσματος από αυτό
- ✓ Τσεκάρουμε το στοιχείο ελέγχου που επιθυμούμε να προσθέσουμε και πατάμε το πλήκτρο OK.
- ✓ Το στοιχείο ελέγχου που επιλέξαμε εμφανίζεται στην εργαλειοθήκη στο project μας.

Αν θέλουμε να αφαιρέσουμε ένα στοιχείο ελέγχου, ακολουθούμε την ίδια διαδικασία και αποεπιλέγουμε το στοιχείο που θέλουμε να αφαιρέσουμε

3. Πλαίσια Διαλόγου **-Dialog Boxes**

Τα πλαίσια διαλόγου (**Dialog Boxes**) αποτελούν το σύντομο τρόπο επικοινωνίας ανάμεσα στα προγράμματα και το χρήστη σχετικά με διάφορα θέματα. Για παράδειγμα, μπορεί να θέλουμε από το πρόγραμμά μας να πληροφορεί το χρήστη σχετικά με κάποιο σφάλμα το οποίο έχει κάνει ή να παρουσιάζει στο χρήστη πολλές δυνατότητες επιλογών για μια συγκεκριμένη λειτουργία. Σε όλες αυτές τις περιπτώσεις μπορούμε να χρησιμοποιήσουμε πλαίσια διαλόγου.

Τα πλαίσια διαλόγου μπορούν να δημιουργηθούν :

- Από το χρήστη μέσω φορμών
- Με χρήση συγκεκριμένων πλαισίων διαλόγου ActiveX, όπως το CommonDialog
- Με τη χρήση των προκαθορισμένων πλαισίων διαλόγου μέσω των συναρτήσεων **InputBox()** και **MsgBox(),** τα οποία αναφέρονται πιο κάτω.

Τα πλαίσια διαλόγου μπορεί να είναι :

- Υποχρεωτικά (Modal). Εδώ ο χρήστης πρέπει να πατήσει κάποιο πλήκτρο του πλαισίου, πριν μπορέσει να κάνει οποιαδήποτε άλλη εργασία στην εφαρμογή ή στον υπολογιστή του.
- Μη υποχρεωτικά (Modeless). Ο χρήστης μπορεί να αγνοήσει το πλαίσιο και να συνεχίσει κανονικά την εργασία του.

3.1 Το πλαίσιο διαλόγου **InputBox**

Το πλαίσιο αυτό αποτελεί επίσης ένα τυποποιημένο πλαίσιο διαλόγου που ζητά από τον χρήστη να πληκτρολογήσει κάποιο κείμενο.

Σχήμα: 12

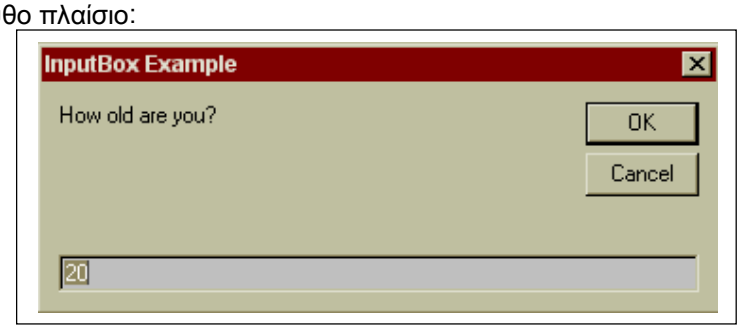
Το κείμενο αυτό συντάσσεται ως εξής :

Κείμενο επιστροφής **= InputBox(**Κείμενο **[,**τίτλος **] [,default] [,x][,y])**

Όπου οι παράμετροι έχουν την εξής σημασία :

- Κείμενο : Το κείμενο που θα εμφανίζει στο χρήστη
- Τίτλος : Προαιρετικό. Το κείμενο στη γραμμή τίτλου του πλαισίου
- Default : Προαιρετικό. Το προεπιλεγμένο κείμενο
- X,y : Προαιρετικά, οι αποστάσεις του πλαισίου από το πάνω αριστερό άκρο της οθόνης

Ετσι, η δήλωση **a = InputBox("How old are you?", "InputBox Example", "20", 200, 200)** δημιουργεί το ακόλουθο πλαίσιο:



Σχήμα: 13

3.2 Το πλαίσιο διαλόγου **MsgBox**

Το πλαίσιο αυτό αποτελεί ένα τυποποιημένο πλαίσιο διαλόγου που προβάλει πληροφορίες στο χρήστη, περιμένοντας την απόκρισή του. Συντάσσεται ως εξής :

Τιμή επιστροφής **= MsgBox(**Κείμενο **[,group** τιμών **] [,**τίτλος **])**

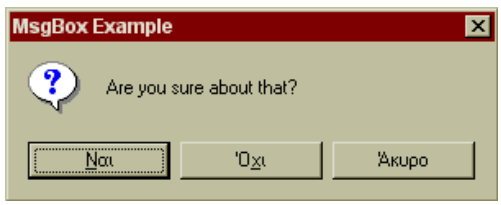
Όπου οι παράμετροι έχουν την εξής σημασία :

- Κείμενο : Το κείμενο που θα εμφανίζει στο χρήστη
- Group τιμών : Αναλύεται πιο κάτω
- Τίτλος : Προαιρετικό. Το κείμενο στη γραμμή τίτλου του πλαισίου

Οι τιμές που παίρνει το group τιμών δίνονται στον πιο κάτω πίνακα :

Σταθερά	Τιμή	Περιγραφή
vbOKOnly	**0**	Εμφάνιση πλήκτρου **OK** μόνο
vbOKCancel	**1**	Εμφάνιση πλήκτρων **OK** και **Cancel**
vbAbortRetryIgnore	**2**	Εμφάνιση πλήκτρων **Abort**, **Retry**, και **Ignore**
vbYesNoCancel	**3**	Εμφάνιση πλήκτρων **Yes**, **No**, και **Cancel**
vbYesNo	**4**	Εμφάνιση πλήκτρων **Yes** και **No**
vbRetryCancel	**5**	Εμφάνιση πλήκτρων **Retry** και **Cancel**
vbCritical	16	Εμφάνιση εικονιδίου **Critical Message**
vbQuestion	32	Εμφάνιση εικονιδίου **Warning Query**
vbExclamation	48	Εμφάνιση εικονιδίου **Warning Message**
vbInformation	64	Εμφάνιση εικονιδίου **Information Message**
vbDefaultButton1	**0**	Προεπιλεγμένο το 1^{ο} πλήκτρο
vbDefaultButton2	**256**	Προεπιλεγμένο το 2^{ο} πλήκτρο
vbDefaultButton3	**512**	Προεπιλεγμένο το 3^{ο} πλήκτρο
vbDefaultButton4	**768**	Προεπιλεγμένο το 4^{ο} πλήκτρο
vbApplicationModal	0	Application modal; Ο χρήστης πρέπει να πατήσει κάποιο πλήκτρο πριν μπορέσει να συνεχίσει να εργάζεται στην εφαρμογή του
vbSystemModal	4096	System modal; Ολες οι εφαρμογές διακόπτονται μέχρι ο χρήστης να πατήσει κάποιο πλήκτρο.

Οι πρώτες 6 τιμές (0-5) αναφέρονται στον αριθμό και τον τύπο των πλήκτρων που εμφανίζονται στο πλαίσιο διαλόγου. Οι επόμενες 4 τιμές (16, 32, 48, 64) αναφέρονται στο εικονίδιο που θα εμφανίζεται. Οι επόμενες 3 τιμές (0, 256, 512) καθορίζουν ποιο πλήκτρο θα είναι το εξ΄ορισμού επιλεγμένο.Τέλος, οι επόμενες 2 τιμές (0, 4096) αναφέρονται στο αν θα είναι υποχρεωτικό ή όχι. Ετσι, για παράδειγμα, η δήλωση α **= MsgBox("Are you sure about that?", 3 + 32, "MsgBox Example")** εμφανίζει το ακόλουθο πλαίσιο**.**



Σχήμα: 14

Ανάλογα με το πλήκτρο που πατήθηκε, η τιμή που επιστρέφεται δίνεται από τον πίνακα :

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

Σχήμα: 15

return_value = MsgBox("Hi all", vbInformation, "MsgBox")

Σχήμα: 16

return_value = MsgBox("Hi all", vbYesNoCancel, "MsgBox")

3.3 Το στοιχείο ελέγχου **CommonDialog**

Το στοιχείο ελέγχου common dialog παρέχει τυποποιημένα πλαίσια διαλόγου για τις λειτουργίες:

- Open
- Save As
- Color
- Font
- Print

Για να χρησιμοιήσουμε τα πλαίσια αυτά στο project που δουλεύουμε θα πρέπει κατ' αρχάς να προσθέσουμε το στοιχείο ελέγχου **Microsoft Common Dialog Control 6** από το μενού **Project** και την επιλογή **Components…** όπως είδαμε στο προηγούμενο κεφάλαιο. Αν δεν εμφανίζεται η επιλογή αυτή στη λίστα, θα πρέπει να προσθέσουμε το αρχείο **Commdlg.dll** στο φάκελο System των Windows. Κατόπιν, μπορούμε να εμφανίσουμε τα σχετικά πλαίσια διαλόγου μέσω των μεθόδων ShowOpen, ShowSave, ShowColor, ShowFont, ShowPrinter και ShowHelp. Την απόκριση του χρήστη μπορούμε να τη διαβάσουμε μέσω των διαφόρων ιδιοτήτων του στοιχείου ελέγχου common dialog. Για παράδειγμα, ο κώδικας :

CommonDialog1.ShowColor

Form1.BackColor=CommonDialog1.color

Εμφανίζει ένα τυποποιημένο πλαίσιο επιλογής χρωμάτων και το χρώμα που επιλέγει ο χρήστης γίνεται χρώμα υποβάθρου στη φόρμα.

Σχήμα: 17 Το τυποποιημένο παράθυρο επιλογής χρωμάτων του CommonDialog

3.4 Απλές εφαρμογές σε **Visual Basic**

Ακολουθούν απλές εφαρμογές για κατανόηση της λογικής, για την ανάπτυξη προγραμμάτων.

Εφαρμογή **1**η**.** Απλή αριθμομηχανή **4** πράξεων

Στην επόμενη εφαρμογή θα υλοποιήσουμε έναν απλό υπολογιστή 4 πράξεων, όπως στο παρακάτω πίνακα. Αφού τρέξουμε την Visual Basic και διαλέξουμε το είδος της εφαρμογής που θα φτιάξουμε, με drag & drop μεταφέρουμε τα αντικείμενα που μας ενδιαφέρουν από την μπάρα (toolbar) της Visual Basic, στην φόρμα μας, στις θέσεις και στα μεγέθη που θέλουμε, και ορίζουμε τις αρχικές τους ιδιότητες. Κατόπιν ξεκινάμε να γράφουμε τον κώδικα που θα εκτελείται κατά την χρήση του προγράμματος.

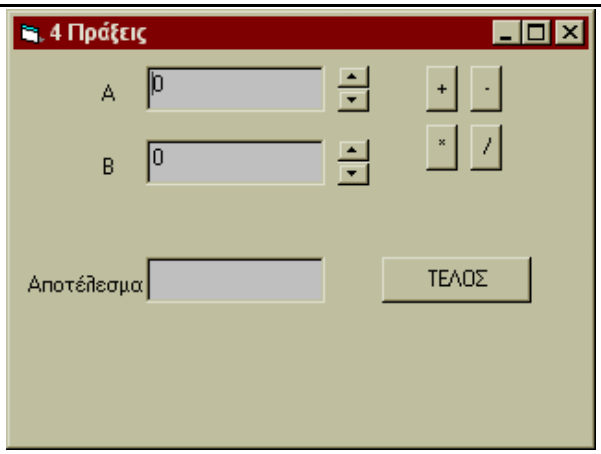
Για παράδειγμα, κρατήστε πατημένο το αριστερό κουμπί του mouse στο εικονίδιο που γράφει Command Button όταν περνάμε τον δείκτη από πάνω του. Μετακινείστε το ποντίκι μέσα στην φόρμα (form1) και αφήστε το κουμπί. Με αυτό τον τρόπο έχουμε τοποθετήσει ένα κουμπί πάνω στην φόρμα. Η Visual Basic by default θα δώσει κάποια αρχικά ονόματα στα αντικείμενα μας έτσι βλέπουμε πως ονομάζει το Command Button μας, “Command1” και την φόρμα μας “Form1”.
Αν θέλουμε να αλλάξουμε το κείμενο πάνω στο κουμπί που γράφει Command1 τότε κάνουμε ένα κλικ πάνω του και πηγαίνουμε και βρίσκουμε από τις ιδιότητες του (δεξιά ένα παραθυράκι που γράφει properties) την επιλογή caption και την αλλάζουμε.

Αν θέλουμε να αλλάξουμε μέγεθος στο κουμπί μας μπορούμε είτε χειροκίνητα από τις ιδιότητες να αλλάξουμε τα width και height, δηλαδή πλάτος και ύψος αντίστοιχα, είτε με οπτικό τρόπο, κάνοντας ένα κλικ πάνω στο κουμπί και μετά από τα τετραγωνάκια που εμφανίζονται να του αλλάξουμε το μέγεθος. (κρατάμε πατημένο το αριστερό σε κάποιο τετραγωνάκι και μετακινούμε το ποντίκι).

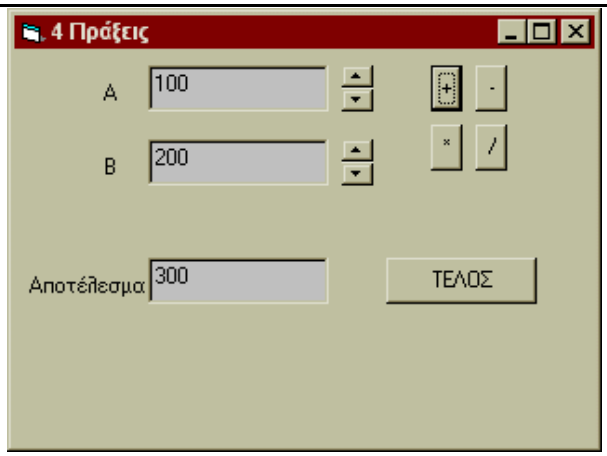
Αν εκτελέσουμε (τρέξουμε) το πρώτο μας πρόγραμμα από το εικονίδιο στην πάνω γραμμή εργαλείων της VB:  θα δούμε πως ήδη μας δίνετε η δυνατότητα να κάνουμε κλικ πάνω στο κουμπί, το οποίο δείχνει να «πατιέται»…

Το σημαντικότερο όμως κομμάτι του προγράμματος, είναι ο κώδικας που πρέπει να γράψουμε για τα αντικείμενα που μας ενδιαφέρουν. Ουσιαστικά πρέπει να πούμε στον υπολογιστή, τι να κάνει, όταν κάποιος

π.χ. πατήσει στο κουμπί μας μια φορά (όταν κάνει κλικ δηλαδή). Αν δεν του πούμε τι να κάνει με κώδικα, δηλαδή με εντολές, τότε φυσιολογικό είναι να μην έχουμε και κανένα αποτέλεσμα. Για να γράψουμε λοιπόν κώδικα ο οποίος θα εκτελείται αναλόγως με κάποιο συμβάν που σχετίζεται με το αντικείμενο, στην περίπτωση μας στο click πάνω στο κουμπί, αρκεί να κάνουμε ένα διπλό κλικ στο αντικείμενο μας και να περάσουμε στο editor όπου πλέον γράφουμε τις εντολές μας (κώδικα). Αυτόματα η Visual Basic δημιουργεί την πιο συνηθισμένη διαδικασία (procedure) για το αντικείμενο στο οποίο κάναμε διπλό κλικ, έτσι λοιπόν για το command button μας γράφει: Private Sub Command1_Click() End Sub και περιμένει από εμάς να γράψουμε κάτι ανάμεσα στα Private Sub και στο End Sub τα οποία καθορίζουν και μια διαδικασία (procedure), δηλαδή μια συγκεκριμένη ενέργεια που συνήθως αποτελείτε από ένα σύνολο εντολών.



Σχήμα: 18 Αρχική μορφή



Σχήμα: 19 Η πράξη της πρόσθεσης

Τα βήματα που ακολουθούμε στο σχεδιασμό της εφαρμογής είναι τα ακόλουθα **:**

- Στη φόρμα εργασίας form1 τοποθετούμε και στοιχίζουμε τα πλαίσια κειμένου text1, text2, text3, τα πλήκτρα Command1, Command2, Command3, Command4 και Command5, τις ετικέτες Label1, Label2 και Label3 και τα scroll bar Vscroll1 και Vscroll2.
- Δίνουμε αρχικές τιμές στις ιδιότητες των αντικειμένων, μέσω του Παράθυρου Ιδιοτήτων :
  - Form1.Caption=4 Πράξεις
  - Command1.Caption= +
  - Command2.Caption=-
  - Command3.Caption=*
  - Command4.Caption=/
  - Command5.Caption=ΤΕΛΟΣ
  - Label1.Caption=A
  - Label2.Caption=B
  - Label3.Caption=Αποτέλεσμα
  - VScroll1.min=0
  - VScroll1.max=100
  - Vscroll2.min=0
  - VScroll1.max=100

Στη συνέχεια γράφουμε τις διάφορες event procedures ανάλογα με την ανταπόκριση κάθε στοιχείου στα διάφορα συμβάντα :

Υπορουτίνα για αρχικοποίηση των τιμών κατά το φόρτωμα της φόρμας **:**

```
Private Sub Form_Load()
    Text1.Text = 0
    Text2.Text = 0
```

```
    Text3.Text = ""
End Sub
```

Αλλαγή στα **VScroll1** και **VScroll2,** για να αλλάζει η τιμή στο **text1, text2** αντίστοιχα **:**

```
Private Sub VScroll1_Change()
    Text1.Text = VScroll1.Value
End Sub

Private Sub VScroll2_Change()
    Text2.Text = VScroll2.Value
End Sub
```

Υπορουτίνες όταν συμβεί το **«**γεγονός **»** πίεσης στα πλήκτρα **Command1, Command2, Command3, Command4,** να γίνεται η αντίστοιχη πράξη στις τιμές των **text1, text2** και το αποτέλεσμα να εμφανίζεται στο **text3 :**

```
Private Sub Command1_Click()
    Text3.Text = CInt(Text1.Text) + CInt(Text2.Text)
End Sub

Private Sub Command2_Click()
    Text3.Text = CInt(Text1.Text) - CInt(Text2.Text)
End Sub

Private Sub Command3_Click()
    Text3.Text = CInt(Text1.Text) * CInt(Text2.Text)
End Sub

Private Sub Command4_Click()
    Text3.Text = CStr(CInt(Text1.Text) / CInt(Text2.Text))
End Sub
```

Υπορουτίνα για να τερματίζει η εφαρμογή όταν συμβεί πίεση στο πλήκτρο **Command5,:**

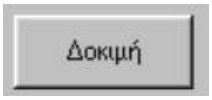
```
Private Sub Command5_Click()
    End
End Sub
```

Εφαρμογή **2**η**.**Πλήκτρο χαιρετισμού

Σε αυτή την εφαρμογή θα εμφανίζεται στη φόρμα ένα πλήκτρο και όταν ο χρήστης πιέσει το πλήκτρο θα εμφανίζεται ένα νέο παράθυρο με ένα μήνυμα χαιρετισμού.

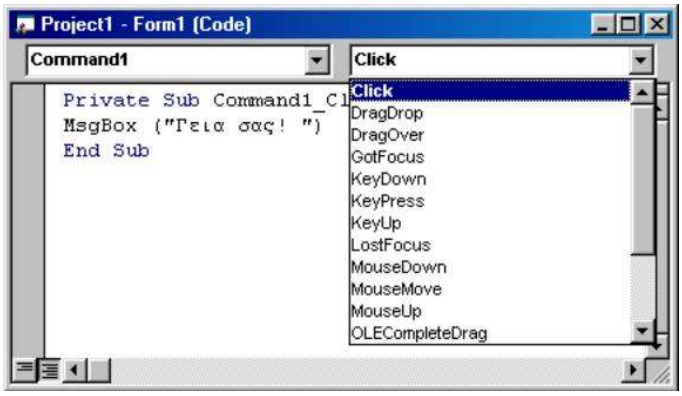
Αφού τρέξουμε πάλι την Visual Basic, με drag & drop ρίχνουμε τα αντικείμενα που μας ενδιαφέρουν από την μπάρα (toolbar) της VB, στην φόρμα μας, στις θέσεις που θέλουμε και ορίζουμε τις αρχικές τους ιδιότητες.

Αν θέλουμε αλλάζουμε το κείμενο πάνω στο κουμπί που γράφει Command1 με την επιλογή caption στις ιδιότητες του / properties.



Για να γράψουμε κώδικα ο οποίος θα εκτελείται με το συμβάν click πάνω στο κουμπί, αρκεί να κάνουμε ένα διπλό κλικ στο αντικείμενο μας και να περάσουμε στο editor όπου πλέον γράφουμε τον κώδικα μας ανάμεσα στο Private Sub Command1_Click() και την εντολή τέλους End Sub.

Ας δοκιμάσουμε να γράψουμε εκεί ανάμεσα το μήνυμα: Msgbox("Γεια σας***!*** ").



Σχήμα: 20

Στη συνέχεια πατώντας το play  θα τρέξει το πρόγραμμα μας, και αν κάνουμε κλικ στο κουμπί θα εμφανιστεί ένα μήνυμα σε ένα popup παραθυράκι που θα γράφει: Γεια σας***!***

Βέβαια το σύνολο των εντολών τις VB είναι αρκετά μεγάλο και θέλει πολλές δοκιμές και εξάσκηση μέχρι να τις μάθουμε όλες, αλλά αν μάθετε αρκετές από αυτές, θα ξέρετε πότε να τις χρησιμοποιήσετε και με ποιόν τρόπο.

Οι διαδικασίες συνήθως σχετίζονται και με κάποιο συμβάν του αντικειμένου, στην περίπτωση μας με το συμβάν click, όταν δηλαδή κάποιος που δουλεύει το πρόγραμμα μας κάνει κλικ πάνω σε αυτό. Μπορούμε να αλλάξουμε και να δούμε και τα υπόλοιπα συμβάντα κάποιου αντικειμένου, όσο βρισκόμαστε στον editor όπου γράφουμε κώδικα, με κλικ στην λίστα των συμβάντων όπως βλέπετε στην παρακάτω εικόνα.
Έτσι αν θέλουμε ο ίδιος κώδικας που εμφάνιζε το μήνυμα Γεια σας*!* Να εμφανίζεται όταν ο χρήστης απλώς περνάει τον δείκτη πάνω από το κουμπί μας, τότε επιλέγουμε από τα συμβάντα το MouseMove, και αυτόματα εμφανίζονται στον editor μας τα παρακάτω: Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single) End Sub Όπου ξανά ανάμεσα γράφουμε το MsgBox ("Γεια σας*!* "). Αν τρέξετε το πρόγραμμα και δοκιμάσετε να περάσετε τον δείκτη από πάνω από το κουμπί, θα δείτε πως κατευθείαν εμφανίζεται και το μήνυμα Γεια σας*!*

Εφαρμογή **3**η**.** Εκτύπωση μηνύματος στην οθόνη

Και αυτό το μικρό προγραμματάκι έχει ως αποτέλεσμα να εκτυπωθεί στην οθόνη το μήνυμα που γράψαμε δηλαδή "Welcome".

Σχήμα : 21

Εδώ όμως, κάνουμε χρήση της εντολής 'WriteLine' που ανήκει στην κλάση Console για να έχουμε την εκτύπωση της λέξης στην οθόνη μας.

4. Τελεστές και μεταβλητές της γλώσσας **Visual Basic**

4.1 Αλφαριθμητικά **&** σχόλια στη **VB**

Μια έκφραση (Expression) της Visual Basic είναι ένας συνδυασμός λέξεων-κλειδιών της γλώσσας, παρενθέσεων, τελεστών, μεταβλητών και σταθερών που επιστρέφει ένα string ή έναν αριθμό ή ένα αντικείμενο. Μια έκφραση χρησιμοποιείται για να εκτελέσει έναν υπολογισμό ή να επεξεργαστεί δεδομένα.

Παραδείγματα εκφράσεων :

A=(1+2) / (3^5)
B=”Hello Mr “ & text1.text & vbCrLf

Για να γίνεται πιο κατανοητός ο κώδικας του προγράμματος συνίσταται η χρήση σχολίων (Comments). Τα σχόλια δηλώνονται με το σύμβολο **‘** (απόστροφος). Για παράδειγμα :

MyValue = 2 ^ 2 **'** Αυτό είναι σχόλιο, η πράξη επιστρέφει 4,

4.2 Αριθμητικοί τελεστές

Οι αριθμητικοί τελεστές που υποστηρίζει η Visual Basic δίνονται στον ακόλουθο πίνακα :

Τελεστής	Χρήση	Παράδειγμα
^	Υψώνει έναν αριθμό σε μία δύναμη	MyValue = 2 **^** 2 ' Returns 4
*****	Πολλαπλασιάζει δύο αριθμούς	MyValue = 2 ***** 2 ' Returns 4
/	Διαιρεί 2 αριθμούς και επιστρέφει το αποτέλεσμα σε μορφή δεκαδικού αριθμού	MyValue = 10 **/** 4 ' Returns 2.5 MyValue = 10 **/** 3 ' Returns 3.333333
****	Διαιρεί 2 αριθμούς και επιστρέφει το αποτέλεσμα σε μορφή ακέραιου αριθμού	MyValue = 11 **** 4 ' Returns 2 MyValue = 9 **** 3 ' Returns 3
Mod	Διαιρεί 2 αριθμούς και επιστρέφει το υπόλοιπο της διαίρεσης	MyResult = 10 **Mod** 5 ' Returns 0 MyResult = 10 **Mod** 3 ' Returns 1
+	Προσθέτει 2 αριθμούς	MyNumber = 2 **+** 2 ' Returns 4
-	Αφαιρεί 2 αριθμούς	MyNumber = 2 **-** 2 ' Returns 0

Οι τελεστές είναι γραμμένοι με σειρά προτεραιότητας, δηλαδή για τον υπολογισμό μιας έκφρασης πρώτα υπολογίζονται οι τελεστές **^**, μετά οι ***** και **/**, κατόπιν οι **** και **Mod** και τελευταίοι οι **+** και **-**. Έτσι, για παράδειγμα, το αποτέλεσμα της έκφρασης **2 + 12^4/3** είναι 3, καθώς πρώτα εκτελείται η ύψωση σε δύναμη, μετά η διαίρεση και τελευταία η πρόσθεση.

Η σειρά προτεραιότητας των τελεστών μπορεί να παρακαμφθεί με τη χρήση των παρενθέσεων. Στην περίπτωση αυτή, πρώτα υπολογίζονται οι όροι που περικλείονται σε παρενθέσεις, ανεξαρτήτως τελεστή. Έτσι, η έκφραση **(2+12)^4/3** δίνει αποτέλεσμα 12805,3, καθώς πρώτα εκτελείται η πρόσθεση μέσα στις παρενθέσεις, μετά η ύψωση σε δύναμη και τέλος η διαίρεση.

4.3 Τελεστές σύγκρισης

Οι τελεστές σύγκρισης που υποστηρίζει η Visual Basic είναι χαρακτήρες ή σύμβολα τα οποία αποδίδουν τη σχέση που υπάρχει μεταξύ δύο ή περισσότερων μεταβλητών ή εκφράσεων και δίνονται στον ακόλουθο πίνακα :

Τελεστής	Χρήση	Παράδειγμα
<	Μικρότερο από	MyResult = (45 **<** 35) ' Returns False
<=	Μικρότερο από ή ίσο με	MyResult = (45 **<=** 35) ' Returns False
>	Μεγαλύτερο από	MyResult = (45 **>** 35) ' Returns True
>=	Μεγαλύτερο από ή ίσο με	MyResult = (45 **>=** 45) ' Returns True
=	Ισο με	MyResult = (45 **=** 45) ' Returns True
<>	Διάφορο από	MyResult = (4 **<>** 3) ' Returns True
Is	Συγκρίνει δύο αντικείμενα	MyCheck = MyObject **Is** ThatObject

Like	Συγκρίνει string, χρησιμοποιούνται χαρακτήρες μπαλαντέρ (!,*,#,[])	MyCheck = "aBBBa" **Like** "a*a" ' Returns True MyCheck = "F" **Like** "[A-Z]" ' Returns True MyCheck = "F" **Like** "[!A-Z]" ' Returns False MyCheck = "a2a" **Like** "a#a" ' Returns True MyCheck = "aM5b" **Like** "a[L-P]#[!c-e]" ' Returns True MyCheck = "BAT123khg" **Like** "B?T*" ' Returns True MyCheck = "CAT123khg" **Like** "B?T*" ' Returns False

4.4 Λογικοί τελεστές

Οι λογικοί τελεστές εκτελούν λογικές πράξεις μεταξύ μεταβλητών ή εκφράσεων της Visual Basic και δίνονται στον ακόλουθο πίνακα :

Τελεστής	Χρήση	Παράδειγμα
AND	Λογική σύζευξη	A = 10: B = 8: C = 6: D = Null ' Initialize variables MyCheck = A > B **And** B > C ' Returns True
EQV	Λογική ισοδυναμία	MyCheck = A > B **Eqv** B > C ' Returns True
IMP	Λογική σύμπτωση	MyCheck = A > B **Imp** B > C ' Returns True
NOT	Λογική άρνηση	MyCheck = **Not**(A > B) ' Returns False
OR	Λογική διάζευξη	MyCheck = A > B **Or** B > C ' Returns True
XOR	Αποκλειστική διάζευξη	MyCheck = A > B **Xor** B > C ' Returns False

4.5 Ονοματολογία σταθερών και μεταβλητών

Κατά τον ορισμό σταθερών και μεταβλητών, αποδίδουμε σε αυτές συμβολικά ονόματα. Τα ονόματα αυτά πρέπει να πληρούν τους εξής κανόνες για να είναι έγκυρα :

- Να μην είναι δεσμευμένες λέξεις
- Να περιέχουν μόνον αλφαριθμητικούς χαρακτήρες (0..9, a..z) και το _ (underscore)
- Να αρχίζουν από κάποιο γράμμα και όχι αριθμό ή σύμβολο
- Το μήκος τους να μην ξεπερνά τους 255 χαρακτήρες

Μερικά αποδεκτά ονόματα σταθερών και μεταβλητών είναι τα Resistance, address, city, people, Grand_Total, TimeCounter, quantity κ.α. Μη αποδεκτά ονόματα είναι τα Month (είναι δεσμευμένη λέξη), 12X (αρχίζει από αριθμό), Grand.Total (περιέχει σημείο στίξης), Time counter (περιέχει κενό).

4.6 Σταθερές

Οι σταθερές χρησιμοποιούνται για την αναπαράσταση ποσοτήτων που παραμένουν αμετάβλητες καθ' όλη τη διάρκεια εκτέλεσης του προγράμματος. Η απόδοση ενός συμβολικού ονόματος στις ποσότητες αυτές διευκολύνει τη διαχείρισή τους από τον προγραμματιστή και προσφέρει καλύτερη διαχείριση της μνήμης του Η/Υ.

Οι σταθερές της Visual Basic, ανάλογα με τον τύπο των δεδομένων τους, μπορεί να ανήκουν σε μια από τις παρακάτω κατηγορίες :

- Αριθμητικές σταθερές (Numeric)
- Λογικές σταθερές (Boolean)
- Αλφαριθμητικές σταθερές (Strings)
- Ημερομηνίες (Dates)

Οι σταθερές της Visual Basic, επίσης, ανάλογα με τον τρόπο ορισμού τους μπορεί να είναι :

- Ετοιμες σταθερές της VisualBasic, π.χ. vBOKOnly

- Σταθερές οριζόμενες από το χρήστη

Οι σταθερές ορίζονται στο τμήμα δηλώσεων μιας προγραμματιστικής μονάδας ή σε μια ρουτίνα ως εξής :

[Private | Public] Const όνομα _σταθεράς **[As** τύπος **]=**Τιμή

Για παράδειγμα:
Const PI = 3.1415, Const fortio = 1.602E-19, Const acolor = &HFF00FF, Const avalue = &O12
Const aflag = True, Const flag1 = False
Const dealer = "Αποστόλου"
Const HireDate="#1/3/99#" 'Μήνας, μέρα, έτος

Τέλος, υπάρχουν 3 ειδών σταθερές ανάλογα με την εμβέλεια (scope) :

- Τοπικές σταθερές . Δηλώνονται με Const μέσα σε μια ρουτίνα. Είναι ορατές μόνο μέσα στη ρουτίνα που δηλώνονται
- Σταθερές επιπέδου προγραμματιστικής μονάδας . Δηλώνονται με Const στο Τμήμα Δηλώσεων της προγραμματιστικής μονάδας. Είναι ορατές μόνο μέσα στην προγραμματιστική μονάδα που δηλώνονται
- Καθολικές σταθερές . Δηλώνονται με Public Const στο Τμήμα Δηλώσεων μιας προγραμματιστικής μονάδας. Είναι ορατές σε όλη την εφαρμογή

4.7 Μεταβλητές

Οι μεταβλητές χρησιμοποιούνται για την αναπαράσταση ποσοτήτων που η τιμή τους μεταβάλλεται κατά τη διάρκεια εκτέλεσης του προγράμματος.
Οι μεταβλητές ορίζονται στο τμήμα δηλώσεων μιας προγραμματιστικής μονάδας ή σε μια ρουτίνα ως εξής :
[Dim | Static | Private | Public] όνομα_μεταβλητής [As τύπος]
Για παράδειγμα: Dim a As Integer ή Dim a%

Τύπος	Αρ. Bytes	Περιοχή Τιμών	Επίθεμα
Integer	2	-32.768 ως +32.767	%
Long	4	-2.147.483.648 ως +2.147.483.647	&
Byte	1	0 ως 255	
Single	4	-3.4E38 ως –1.4E-45 ή 1.4E-45 ως 3.4E38	!
Double	8	-1.7E308 ως –4.9E-324 ή 4.9E-324 ως 1.7E308	#
Decimal	14	Πολύ μεγάλοι και πολύ μικροί αριθμοί	
Currency	8	-922.337.203.685.477,5808 ως 922.337.203.685.477,5807	@
String	1 ανά χαρακτήρα	Χαρακτήρες	$
Boolean	2	True, False	
Date	8	1/1/100 ως 31/12/9999	
Object	4	Οποιοδήποτε αντικείμενο	

Dim lexi As String ή Dim lexi$

Οι τύποι μεταβλητών που υποστηρίζει η Visual Basic είναι οι εξής :

Η δήλωση μεταβλητών μπορεί να είναι :

- Υποχρεωτική , Στην περίπτωση αυτή κάθε μεταβλητή πριν χρησιμοποιηθεί πρέπει απαραιτήτως να δηλωθεί με τη δήλωση **Option Explicit** στο τμήμα δηλώσεων της

προγραμματιστικής μονάδος. Ο τρόπος αυτός δήλωσης είναι προγραμματιστικά ο πιο σωστός και διευκολύνει τη διαδικασία συγγραφής του κώδικα.

- Μη Υποχρεωτική , όπου οι μεταβλητές μπορεί να χρησιμοποιούνται χωρίς να δηλώνονται πρώτα και υλοποιείται χωρίς καμία δήλωση στο τμήμα δηλώσεων.

Υπάρχουν 4 ειδών μεταβλητές ανάλογα με την εμβέλεια (scope) :

- Τοπικές μεταβλητές . Δηλώνονται με **Dim** μέσα σε μια διαδικασία. Είναι ορατές μόνο εντός της διαδικασίας.
- Στατικές μεταβλητές **.** Σαν τις τοπικές αλλά διατηρούν την τιμή τους μεταξύ διαδοχικών κλήσεων της διαδικασίας **.** Δηλώνονται με το **Static.** Είναι ορατές μόνο εντός της διαδικασίας **.**
-
- Μεταβλητές προγραμματιστικής μονάδας **.** Δηλώνονται με **Dim** ή **Private** στο τμήμα δηλώσεων της προγραμματιστικής μονάδας **.** Είναι ορατές μόνο εντός της προγραμματιστικής μονάδας **.**
- Καθολικές μεταβλητές **.** Δηλώνονται με **Public** στο Τμήμα Δηλώσεων οποιασδήποτε προγραμματιστικής μονάδας **.** Είναι ορατές σε όλη την εφαρμογή **.**

4.8 Συναρτήσεις μετατροπής τύπων μεταβλητών

Πολλές φορές υπάρχει η ανάγκη από μια μεταβλητή ενός συγκεκριμένου τύπου να δημιουργούμε μια άλλη μεταβλητή, με ίδια τιμή αλλά διαφορετικό τύπο. Για παράδειγμα, το περιεχόμενο ενός text box, το οποίο είναι πάντα κείμενο, μπορούμε να το μετατρέψουμε σε αριθμητικό τύπο, όταν φυσικά αυτό είναι δυνατό, για να εκτελέσουμε αριθμητικές πράξεις, όπως για παράδειγμα το string "123" μπορεί να μετατραπεί σε μια μεταβλητή τύπου byte ή integer με τιμή 123.

Για την μετατροπή τύπου υπάρχουν οι ακόλουθες συναρτήσεις της Visual Basic :

Συνάρτηση	Τύπος επιστροφής	Δέχεται σαν είσοδο
Cbool	Boolean	Οποιαδήποτε αριθμητική ή string έκφραση
Cbyte	Byte	0 έως 255
Ccur	Currency	-922,337,203,685,477.5808έως 922,337,203,685,477.5807
Cdate	Date	Οποιαδήποτε έγκυρη έκφραση ημερομηνίας
CDbl	Double	-1.79769313486232E308έως -4.94065645841247E-324για αρνητικούς. 4.94065645841247E-324έως 1.79769313486232E308για θετικούς
Cdec	Decimal	+/-79,228,162,514,264,337,593,543,950,335για αριθμούς χωρίς δεκαδικά. Για αριθμούς με έως 28 δεκαδικά ψηφία +/-7.9228162514264337593543950335. Ο μικρότερος μη μηδενικός αριθμός είναι ο 0.0000000000000000000000000001.
Cint	Integer	-32,768 έως 32,767. Αν υπάρχουν δεκαδικά ψηφία στρογγυλοποιούνται
CLng	Long	-2,147,483,648 έως 2,147,483,647. Αν υπάρχουν δεκαδικά ψηφία στρογγυλοποιούνται
CSng	Single	-3.402823E38 έως -1.401298E-45 για αρνητικές τιμές. 1.401298E-45 έως 3.402823E38 για θετικές τιμές
CStr	String	Οτιδήποτε
Cvar	Variant	Αριθμούς ή string

Για παράδειγμα:

```
DIM numero AS Integer
DIM lexi AS String
Number1=100
Word1="100"
Number1=Cint(lexi)
Word1=CStr(numero)
```

```
DIM numero AS Integer
DIM lexi AS String
Number1=Cint(text1.text)
Text1.text=CStr(numero)
```

4.9 Η εντολή **if**

Στις περιπτώσεις που το πρόγραμμα μας θα πρέπει να εξετάσει κάποιες συνθήκες, και ανάλογα να εκτελέσει διαφορετικές εντολές για κάθε περίπτωση, χρησιμοποιούμε την δομή **if…then…else**. Οι τρόποι που συντάσσουμε την if έχουν ως ακολούθως :

If...Then … Else

If συνθήκη **Then** εντολή

Ή

If συνθήκη **Then**
  Εντολή 1
  Εντολή 2
  ……
End If

Παράδειγμα **:**

```
If anyDate < Now Then
  anyDate = Now
  Timer1.Enabled = False
End If
```

If συνθήκη1 **Then**
  Εντολή 1α
  Εντολή 1β
  ……
ElseIf συνθήκη2 **Then**
  Εντολή 2α
  Εντολή 2β
  ……
Else
  Εντολή Να
  Εντολή Νβ
  ……
End If

Παράδειγμα **:**

```
Private Sub mnuCut_Click (Index As Integer)
  If Index = 0 Then
      CopyActiveControl
        ClearActiveControl
  ElseIf Index = 1 Then
    CopyActiveControl
  ElseIf Index = 2 Then
    ClearActiveControl
  Else
    PasteActiveControl
  End If
End Sub
```

(Β) Select Case

Στις περιπτώσεις που υπάρχουν πολλές συνθήκες προς εξέταση μπορούμε να χρησιμοποιήσουμε την **Select … Case** η οποία δίνει πιο κατανοητό κώδικα. Η σύνταξη έχει ως ακολούθως :

Select Case έκφραση
Case τιμή1
  Εντολές
Case τιμή2
  Εντολές
……
Case IS ανίσωση
  Εντολές
Case τιμή1 TO τιμή2
  Εντολές
Case Else
  Εντολές

End Select

Παράδειγμα **:**

```
Select Case Index
  Case 0
      CopyActiveControl
      ClearActiveControl
   Case 1
      CopyActiveControl
   Case 2
      ClearActiveControl
   Case 3
      PasteActiveControl
   Case 4 TO 10
      MsgBox("Μεγάλος Αριθμός")
   Case IS >100
      MsgBox("ΛΑΘΟΣ")
   Case Else
      frmFind.Show

End Select
```

(Α) FOR…NEXT

Οι δομές επανάληψης (Loops) επιτρέπουν, μέσα από την ικανοποίηση ή μη μιας συνθήκης, την επαναληπτική εκτέλεση μιας ομάδας εντολών. Η Visual Basic υποστηρίζει 3 γενικούς τύπους δομών επανάληψης :

FOR…NEXT. Τη δομή αυτή χρησιμοποιούμε όταν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων. Συντάσσεται ως εξής :

For μεταβλητή **=** αρχικήΤιμη **To** τελικήΤιμή [**Step** βήμα]

εντολές

……

[**Exit For**]

εντολές

……

Next [μεταβλητή]

(Β1) DO While…LOOP

Τη χρησιμοποιούμε όταν δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων.

Ο έλεγχος της συνθήκης γίνεται στην αρχή του βρόχου και η επόμενη επανάληψη εκτελείται αν ισχύει η συνθήκη (DO While…). Οι επαναλήψεις που μπορούν να εκτελεστούν είναι 0 έως όσο ισχύει η συνθήκη. Συντάσσεται ως εξής:

Do While συνθήκη

εντολές

……

[**Exit Do**]

εντολές

……

Loop

(Β2) DO Until…LOOP

Τη χρησιμοποιούμε όταν δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων.

Ο έλεγχος της συνθήκης γίνεται στην αρχή του βρόχου και η επόμενη επανάληψη εκτελείται αν δεν ισχύει η συνθήκη (DO Until...). Οι επαναλήψεις που μπορούν να εκτελεστούν είναι 0 έως όσο ισχύει η συνθήκη. Συντάσσεται ως εξής :

Do Until συνθήκη

εντολές

……

[**Exit Do**]

εντολές

……

Loop

(Γ1) DO … Loop Until

Τη δομή αυτή χρησιμοποιούμε επίσης όταν δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων. Ο έλεγχος της συνθήκης γίνεται στο τέλος του βρόχου και η επόμενη επανάληψη εκτελείται αν δεν ισχύει η συνθήκη (Loop Until). Οι επαναλήψεις που μπορούν να εκτελεστούν είναι 1 έως όσο δεν ισχύει η συνθήκη, δηλαδή είναι συνθήκη τέλους. Συντάσσεται ως εξής :

Do

εντολές

……

[**Exit Do**]

εντολές

……

Loop Until συνθήκη

(Γ2) DO … Loop While

Τη δομή αυτή χρησιμοποιούμε επίσης όταν δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων. Ο έλεγχος της συνθήκης γίνεται στο τέλος του βρόχου και η επόμενη επανάληψη εκτελείται αν ισχύει η συνθήκη (Loop While). Οι επαναλήψεις που μπορούν να εκτελεστούν είναι 1 έως όσο ισχύει η συνθήκη, δηλαδή είναι συνθήκη τέλους. Συντάσσεται ως εξής :

Do

εντολές

……

[**Exit Do**]

εντολές

……

Loop While συνθήκη

Επάνω στην δομή επιλογής if μπορούμε να υλοποιήσουμε μία μικρή εφαρμογή: ένα χρονόμετρο. Το χρονόμετρο αυτό θα έχει 3 κουμπιά, το START για να ξεκινάει η μέτρηση του χρόνου, το STOP για να σταματά και το RESET για να μηδενίζει. Σαν οθόνη χρησιμοποιούμε ένα text box. Για τη μέτρηση του χρόνου χρησιμοποιούμε το εργαλείο timer του οποίου θέτουμε την ιδιότητα **timer1.interval=1000** ώστε να δημιουργεί ένα γεγονός timer κάθε 1.000 msec. Χρησιμοποιούμε 3 μεταβλητές, την **sec** για τα δευτερόλεπτα, την **min** για τα λεπτά και την **hour** για τις ώρες. Σε κάθε συμβάν timer αυξάνουμε τη sec κατά 1 και εξετάζουμε αν έχει πάρει την τιμή 60. Αν ναι, την μηδενίζουμε και αυξάνουμε την min κατά 1. Κατόπιν εξετάζουμε την min αν έχει πάρει την τιμή 60. Αν ναι, την μηδενίζουμε και αυξάνουμε την hour κατά 1.

Το πλήκτρο START ενεργοποιεί τον timer θέτοντας την ιδιότητα **timer1.enabled=true** και το πλήκτρο STOP τον απενεργοποιεί θέτοντας την ιδιότητα **timer1.enabled=false**.

Ακολουθεί ο κώδικας του χρονομέτρου :

```
Option Explicit
Dim sec%, min%, hour%

Private Sub Form_Load()
    Text1.Text = "00:00:00"
    sec% = 0
    min% = 0
```

```
Private Sub Timer1_Timer()
        sec% = sec% + 1
        If sec% = 60 Then
           sec% = 0
           min% = min% + 1
        End If
        If min% = 60 Then
```

```
    hour% = 0
    Timer1.Enabled = False
End Sub

Private Sub Command1_Click()
    Timer1.Enabled = True
End Sub

Private Sub Command2_Click()
    Timer1.Enabled = False
End Sub

Private Sub Command3_Click()
    Text1.Text = "00:00:00"
    sec% = 0
    min% = 0
    hour% = 0
End Sub
```

```
            min% = 0
            hour% = hour% + 1
        End If
        If hour% = 99 Then
            hour% = 0
        End If
        Text1.Text = hour% & ":" & min% & ":" &
sec%
End Sub
```

5. Πίνακες στη **Visual Basic**

5.1 Δήλωση πίνακα

Ο προγραμματισμός μεγάλου όγκου δεδομένων και όταν μάλιστα αυτά έχουν διαιρεθεί σε πολλές κατηγορίες συνεπάγεται αναγκαστικά τη χρήση των πινάκων (Arrays). Οι πίνακες αποτελούν πολύ σημαντικά εργαλεία σε οποιαδήποτε γλώσσα προγραμματισμού καθώς επιτρέπουν εύκολη απόδοση ονομάτων σε μεγάλο πλήθος μεταβλητών ενώ η εγγραφή και ανάγνωση στους πίνακες γίνεται εύκολα, είτε τυχαία είτε με την βοήθεια επαναλήψεων.

Ένας πίνακας είναι μια συλλογή μεταβλητών του ιδίου τύπου και ονόματος. Για τον προσδιορισμό ενός συγκεκριμένου στοιχείου σε έναν πίνακα χρησιμοποιούμε το όνομα του πίνακα και έναν αριθμητικό δείκτη (Index) που προσδιορίζει μονοσήμαντα το στοιχείο αυτό, π.χ. η έκφραση Students(5) αναφέρεται στο $5^{ο}$ στοιχείο του πίνακα Students.

Οι πίνακες μπορούν να έχουν διάφορα μεγέθη. Ένας πίνακας μπορεί να έχει δύο ή τέσσερα στοιχεία, ένας άλλος μπορεί να έχει 1200 στοιχεία και μπορεί, επίσης, ένας πίνακας να μην έχει καθόλου στοιχεία, απλά να έχει τη δυνατότητα να "κρατήσει" κάποια στοιχεία τα οποία θα εισαχθούν αργότερα.

Καθώς θα εξοικειώνεστε με τους πίνακες, θα διαπιστώσετε ότι συμβαδίζουν με τα βασικά στοιχεία ελέγχου ListBox (πλαίσιο λίστας) και ComboBox (σύνθετο πλαίσιο) της Visual Basic

Οι πίνακες ορίζονται στο τμήμα δηλώσεων μιας προγραμματιστικής μονάδας ή σε μια ρουτίνα, με τρόπο παρόμοιο με τις μεταβλητές, ως εξής :

Dim | Public | Private ΟνομαΠίνακα **(**Μέγιστος _Δείκτης **) As** τύπος

όπου εδώ ο Μέγιστος_Δείκτης αναφέρεται στο υψηλότερο στοιχείο του πίνακα και ξεκινά εξ ορισμού από το 0. Επειδή όμως αυτό προκαλεί σύγχυση, μπορούμε να κάνουμε το πρώτο στοιχείο του πίνακα να έχει δείκτη 1 εισάγοντας την πρόταση **Options Base 1** στην ενότητα General μιας λειτουργικής μονάδας.

Για παράδειγμα:

Δήλωση πίνακα πέντε ακεραίων αριθμών : Dim x(4) as integer

Εισαγωγή τιμών στον ανωτέρω πίνακα : X(0)=4, X(1)=31, X(2)=14, X(3)=18, X(4)=46

Δήλωση πίνακα εννέα καθολικών μεταβλητών τύπου χαρακτήρων : Public X(8) as string

Δήλωση μονοδιάστατου πίνακα χωρίς στοιχεία : Dim X() as string

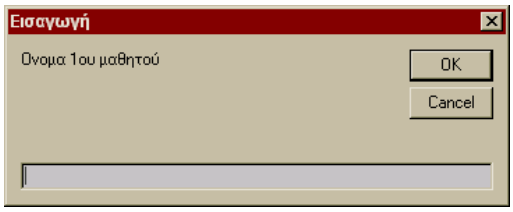
Μπορούμε επίσης να δηλώσουμε έναν πίνακα χρησιμοποιώντας τη λέξη κλειδί "To" μέσα στη σύνταξη του δείκτη. Για παράδειγμα, αν θέλουμε να δημιουργήσουμε έναν πίνακα πέντε μεταβλητών τύπου Integer και με δείκτες από 1 έως 5, τότε μπορούμε να γράψουμε : Dim X(1 To 5) as Integer

5.2 Εισαγωγή , εκτύπωση , επεξεργασία στοιχείων πίνακα

Για να εισάγουμε στοιχεία σε έναν πίνακα χρησιμοποιούμε κυρίως δομές επανάληψης, ειδικά αν ο πίνακας είναι μεγάλος σε μέγεθος. Τα στοιχεία μπορούν να εισαχθούν είτε από αρχεία είτε από παράθυρα. Στο παρακάτω παράδειγμα χρησιμοποιούμε έναν πίνακα 40 θέσεων για την εισαγωγή των ονομάτων των μαθητών μιας τάξης. Για την εισαγωγή από τον χρήστη των ονομάτων χρησιμοποιούμε ένα InputBox και για την προβολή των ονομάτων ένα TextBox.

```
Option Explicit
Dim students(40) As String  'Πίνακας μαθητών
Dim No As Integer           'Πλήθος μαθητών τάξης
Dim i As Integer            'Μετρητής βρόγχου

Private Sub Command1_Click()
'Εισαγωγή πλήθους μαθητών
No = InputBox("Πλήθος μαθητών ?", "Εισαγωγή")
If No > 40 Then End
'Εισαγωγή ονομάτων μαθητών
For i = 1 To No
   students(i) = InputBox("Ονομα " & i & "ου μαθητού", "Εισαγωγή")
Next i
End Sub
```



Σχήμα: 22

```
Private Sub Command2_Click()
For i = 1 To No
   Text1.Text = Text1.Text & i & ". " & students(i) & vbCrLf
Next i
End Sub
```

Στην δήλωση ενός πίνακα φροντίζουμε να του δίνουμε το μέγιστο μέγεθος που μπορεί να χρησιμοποιήσουμε αλλά όχι παραπάνω, γιατί τότε δεσμεύουμε άσκοπα χώρο στη μνήμη. Επίσης, όταν διαβάζουμε ή γράφουμε το στοιχείο ενός πίνακα προσέχουμε ο δείκτης να μην πάρει τιμή μεγαλύτερη από τον μέγιστο δείκτη, διότι αυτό θα οδηγήσει σε σοβαρό σφάλμα εκτέλεσης και 'κρέμασμα' της εφαρμογής μας.

Ένα σημαντικό στοιχείο που θα πρέπει να έχει υπόψη του ο προγραμματιστής είναι ότι οι πίνακες διατηρούνται στη μνήμη RAM και με την έξοδο από την εφαρμογή μας, αυτοί διαγράφονται οριστικά. Συνεπώς, για να διατηρήσουμε τις καταχωρήσεις μας σε έναν πίνακα θα πρέπει να τον αντιγράψουμε σε ένα αρχείο, και την επόμενη φορά που θα εκτελέσουμε την εφαρμογή μας να διαβάσουμε τις εγγραφές από εκεί.

5.3 Δισδιάστατοι και πολυδιάστατοι πίνακες

Οι πίνακες που είδαμε μέχρι τώρα είναι ανεπτυγμένοι σε μια διάσταση και λέγονται μονοδιάστατοι . Σε πολλά προβλήματα, όμως, υπάρχει η ανάγκη χρήσης δισδιάστατων , τρισδιάστατων και γενικά πολυδιάστατων πινάκων. Αρκετά χρήσιμοι είναι οι πίνακες 2 διαστάσεων, όπως για παράδειγμα στην περίπτωση που θέλουμε να αποθηκεύσουμε τους βαθμούς Χ μαθητών σε Υ μαθήματα. Επίσης, όλες οι εικόνες στον υπολογιστή είναι πίνακες 2 διαστάσεων, όπου κάθε στοιχείο του πίνακα αντιστοιχεί σε ένα pixel της εικόνας. Πίνακα 3 διαστάσεων θα μπορούσαμε να χρησιμοποιήσουμε για να αποθηκεύσουμε τη θέση ενός αντικειμένου που κινείται στο χώρο, π.χ. ενός αεροπλάνου. Πίνακες μεγαλυτέρων διαστάσεων χρησιμοποιούνται πιο σπάνια, καθώς είναι δύσκολη η κατανόηση και ο χειρισμός τους.

Η Visual Basic υποστηρίζει πίνακες μέχρι 60 διαστάσεων.

Για τη δήλωση ενός Ν-διάστατου πίνακα, βάζουμε όχι έναν αλλά τόσους μέγιστους δείκτες όσες είναι οι διαστάσεις του πίνακα :

Dim | Public | Private ΟνομαΠίνακα **(**Μ_Δείκτης **1,** Μ_Δείκτης **2,…,** Μ_Δείκτης **Ν) As** τύπος

Για παράδειγμα:

Δήλωση 2-διάστατου πίνακα 5Χ4 ακεραίων αριθμών : Dim x(5,4) as integer

Εισαγωγή τιμών στον ανωτέρω πίνακα : X(0,0)=4, X(1,0)=31, X(2,0)=14, X(3,2)=18, X(4,3)=46

Η διαχείριση των πολυδιάστατων πινάκων γίνεται συνήθως με ένθετους βρόγχους, τόσους όσες και οι διαστάσεις του πίνακα. Για παράδειγμα, η εισαγωγή τιμών σε έναν δισδιάστατο πίνακα 30Χ15 θα γίνεται με 2 βρόγχους ως εξής :

```
Dim pinakas(30,15) As String  'Πίνακας μαθητών
Dim Message As String
For i=1 to 30
  For j=1 to 15
    Message="Τιμή στη " & i & "η σειρά " & j & "η θέση"
    Pinakas(I,j)=InputBox(Message, "Εισαγωγή")
  Next i
Next j
```

5.4 Στατικοί και Δυναμικοί πίνακες

Ο τρόπος δήλωσης των πινάκων που είδαμε μέχρι τώρα θέτει τον περιορισμό ότι πρέπει να γνωρίζουμε εκ των προτέρων την μέγιστη τιμή που μπορεί να πάρει ένας δείκτης κατά την εκτέλεση της εφαρμογής. Επίσης, ο πίνακας καταλαμβάνει συνεχώς χώρο στη μνήμη, ακόμη και όταν δεν χρησιμοποιείται. Οι πίνακες αυτοί ονομάζονται Στατικοί .

Στη Visual Basic υπάρχει η δυνατότητα ορισμού και Δυναμικών πινάκων, πινάκων δηλαδή που τα όρια τους, και κατά συνέπεια το μέγεθος τους στη μνήμη, μπορεί να αλλάζει δυναμικά κατά τη διάρκεια εκτέλεσης της εφαρμογής. Οι πίνακες αυτοί είναι ιδιαίτερα χρήσιμοι, καθώς σπάνια γνωρίζουμε εκ των προτέρων τα ακριβή όρια ενός πίνακα, ενώ, παράλληλα, επιτυγχάνουμε αποδοτικότερη χρήση της μνήμης, χρησιμοποιώντας ακριβώς όσο χώρο χρειαζόμαστε.

Ο ορισμός ενός δυναμικού πίνακα γίνεται όπως και ενός στατικού, με μόνη διαφορά ότι δε δηλώνουμε μέγιστο δείκτη :

Dim | Public | Private ΟνομαΠίνακα **() As** τύπος

Για παράδειγμα:

Δήλωση δυναμικού πίνακα ακεραίων αριθμών : Dim x() as integer

Όταν χρειαστεί να οριστεί ή να αλλάξει το μέγεθος του πίνακα χρησιμοποιείται η δήλωση **Redim** ΟνομαΠίνακα **(**τρέχον _μέγιστος _δείκτης **)**. Προφανώς η Redim μπορεί να χρησιμοποιηθεί πολλές φορές αυξομειώνοντας το μέγεθος του πίνακα. Η συνάρτηση **Ubound**() επιστρέφει την τρέχουσα τιμή του μέγιστου δείκτη.

Όταν εκτελείται η Redim όλα τα στοιχεία του πίνακα διαγράφονται και τίθεται η τιμή 0 στους αριθμητικούς πίνακες και "" στους πίνακες string. Αν υπάρχουν ήδη καταχωρημένα στοιχεία και θέλουμε να τα διατηρήσουμε, τότε προσθέτουμε την παράμετρο **Preserve** στην Redim, π.χ. **Redim Preserve** ΟνομαΠίνακα **(**τρέχον _μέγιστος _δείκτης **)**. Φυσικά, όταν μικραίνουμε το μέγεθος του πίνακα, τα στοιχεία που βρίσκονται στις θέσεις που καταργούμε θα χαθούν. Τέλος, ο μέγιστος δείκτης που δίνουμε με την Redim μπορεί να είναι όχι μόνο σταθερά, όπως στους στατικούς πίνακες, αλλά και μια έκφραση, δίνοντας έτσι μεγαλύτερη ευελιξία στον προγραμματιστή.

Στο επόμενο παράδειγμα ορίζουμε έναν μονοδιάστατο δυναμικό πίνακα και αλλάζουμε το μέγεθός του :

```
Dim pinakas() As Integer   ' Δυναμικός Πίνακας ακεραίων
Redim pinakas(5)           ' Δημιουργία 5 θέσεων
Pinakas(1)=3               '  Καταχωρήσεις τιμών
Pinakas(2)=4
Redim Preserve pinakas(10) ' Αύξηση μεγέθους με διατήρηση περιεχομένων
Pinakas(9)=100
Redim pinakas(100)         ' Αύξηση μεγέθους χωρίς διατήρηση περιεχομένων
```

6. Υπορουτίνες και Συναρτήσεις

6.1 Υπορουτίνες

Η περιγραφή και η ανάλυση ενός σύνθετου έργου μπορεί να γίνει εύκολα και απλά αν το διασπάσουμε σε μικρότερα τμήματα (**modules**), τα οποία διασπούμε σε ακόμη πιο μικρά τμήματα κ.ο.κ. Η λογική αυτή μεταφέρεται και στην κωδικοποίηση. Έτσι, αν έχουμε να περιγράψουμε μια μεγάλη και σύνθετη διαδικασία, τη χωρίζουμε σε μικρότερες και απλούστερες υποδιαδικασίες, κωδικοποιούμε κάθε μία ξεχωριστά, αν χρειαστεί τη διασπάμε σε άλλες απλούστερες κ.ο.κ Αυτός ο τρόπος προγραμματισμού είναι γνωστός ως Αρθρωτός ή Τμηματικός Προγραμματισμός (Modular Programming) και έχει επιπλέον τα εξής πλεονεκτήματα :

- Αποφυγή της επαναληπτικής γραφής του ίδιου συνόλου εντολών σε πολλά διαφορετικά σημεία. Ο κώδικας κάθε διαδικασίας γράφεται μόνο μια φορά και καλείται όσες φορές χρειάζεται, από διάφορα σημεία του προγράμματος.
- Ευκολία συγγραφής κώδικα, ελέγχου και συντήρησης. Επειδή, οι διάφορες διαδικασίες του γράφονται μια φορά και σε ένα σημείο, είναι εύκολο να γίνουν διορθώσεις ή βελτιώσεις σε κάθε μία ξεχωριστά, χωρίς να επηρεάζονται άλλα σημεία της εφαρμογής.
- Ευκολότερη κατανόηση του κώδικα από τρίτους, καθώς γίνεται πιο μικρός και συνεκτικός.

Στη Visual Basic η κωδικοποίηση των διαδικασιών γίνεται σε ανεξάρτητα τμήματα κώδικα, τις υπορουτίνες (subroutines) και τις συναρτήσεις (functions), οι οποίες τοποθετούνται μέσα σε προγραμματιστικές μονάδες.

Στον κώδικα μιας προγραμματιστικής μονάδας εκτός από τις υπορουτίνες διαχείρισης συμβάντων μπορούμε να συμπεριλάβουμε και υπορουτίνες γενικού σκοπού. Οι υπορουτίνες γενικού σκοπού είναι τμήματα κώδικα, τα οποία δεν εκτελούνται όταν προκαλούνται κάποια συμβάντα αλλά μόνον όταν γίνεται η κλήση τους από άλλα σημεία του κώδικα.

Οι υπορουτίνες αυτές μπορούν να οριστούν σε οποιοδήποτε σημείο μιας προγραμματιστικής μονάδος και συντάσσονται ως εξής :

[Public | Private | Static] Sub όνομα _υπορουτίνας **(**παράμετρος **1 As** τύπος **1,** παράμετρος **2 As** τύπος **2, …)**
Εντολή **1**
Εντολή **2**
[exit sub]
……..
End Sub

Η προαιρετική δήλωση **Exit Sub** χρησιμοποιείται για την έξοδο από μια υπορουτίνα πριν ολοκληρωθεί η εκτέλεσή της. Οι υπορουτίνες εκτελούνται μόνον όταν καλούνται και η κλήση (call) τους γίνεται με αναφορά του ονόματος τους και παράθεση παραμέτρων (αν υπάρχουν) σε μια γραμμή κώδικα :

όνομα _υπορουτίνας μεταβλητή **1,** μεταβλητή **2, ….**

Οι παράμετροι της υπορουτίνας χρησιμοποιούνται για την ανταλλαγή πληροφοριών με το υπόλοιπο πρόγραμμα. Μια υπορουτίνα μπορεί να μην έχει παραμέτρους και στην περίπτωση αυτή ανταλλάσσει δεδομένα με το υπόλοιπο πρόγραμμα μέσω καθολικών μεταβλητών ή με τις μεταβλητές της προγραμματιστικής μονάδας που ανήκει

Στο επόμενο παράδειγμα έχουμε ένα πρόγραμμα στο οποίο θα πρέπει σε 3 σημεία του να μηδενίζουμε έναν μονοδιάστατο πίνακα μεγέθους 100 θέσεων. Στην αριστερή στήλη φαίνεται ο κώδικας χωρίς χρήση υπορουτίνας και στη δεξιά στήλη με χρήση υπορουτίνας.

…… 'Μηδένισε τον πίνακα A For i=1 to 100 A(i)=0 Next I ……	'Ορισμός υπορουτίνας **Sub ClearArrayA** For I=1 to 100 A(i)=0 Next I **End Sub**

……	…….
'Μηδένισε τον πίνακα A For i=1 to 100 A(i)=0 Next I	…… 'Μηδένισε τον πίνακα A ClearArrayA
…… ……	…… ……
'Μηδένισε τον πίνακα A For i=1 to 100 A(i)=0 Next I	'Μηδένισε τον πίνακα A ClearArrayA
…… ……	…… ……
	'Μηδένισε τον πίνακα A ClearArrayA

6.2 Συναρτήσεις

Οι συναρτήσεις μοιάζουνε με τις υπορουτίνες με τη διαφορά ότι, αντίθετα με τις υπορουτίνες, επιστρέφουν μια τιμή στο πρόγραμμα. Για το λόγο αυτό, όταν καλούνται δεν μπορούν να κληθούν μόνο με το όνομά τους αλλά πάντα μπαίνουν στο δεξιό μέρος μια ισότητας ή μέσα σε μια εντολή εκτύπωσης. Επίσης, μια συνάρτηση μπορεί να καλέσει τον εαυτό της. Η τεχνική αυτή ονομάζεται Αναδρομή (Recursion) και χρησιμοποιείται σε ορισμένους αλγορίθμους, όπως η εύρεση του παραγοντικού ενός αριθμού, αλλά θέλει προσοχή στη χρήση της γιατί μπορεί να προκαλέσει ένα κρίσιμο σφάλμα συστήματος, που ονομάζεται **stack overflow,** με αποτέλεσμα το 'κρέμασμα' του προγράμματος.

Οι συναρτήσεις ορίζονται ως εξής :

[Public | Private | Static] Function όνομα _συνάρτησης **(**παράμετρος **1 As** τύπος **1,** παράμετρος **2 As** τύπος **2, …) AS** τύπος _συνάρτησης

Εντολή **1**

Εντολή **2**

[exit function]

όνομα _συνάρτησης **=**τιμή

……..

End Function

Όπως φαίνεται, ο ορισμός μιας συνάρτησης αρχίζει με την εντολή Function και εν συνεχεία δηλώνονται οι παράμετροι της συνάρτησης και οι τύποι τους. Επίσης, σε αντίθεση με τις υπορουτίνες, δηλώνεται και ο τύπος της συνάρτησης, δηλαδή ο τύπος της τιμής που επιστρέφει η συνάρτηση. Στο εσωτερικό της συνάρτησης πρέπει να υπάρχει τουλάχιστον μια εντολή εκχώρησης τιμής στο όνομα της συνάρτησης. Η έξοδος από μια συνάρτηση γίνεται με τη δήλωση **End Function** ή, αν είναι επιθυμητή η έξοδος πριν την πλήρη εκτέλεσή της, με τη δήλωση **exit function**.

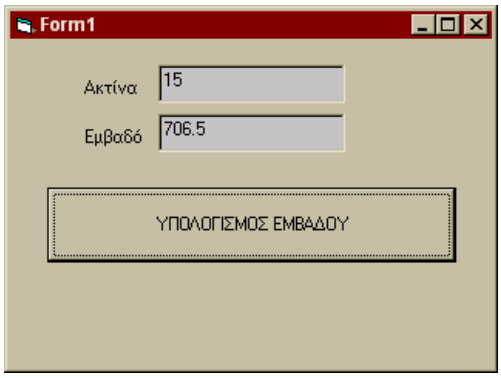
Οι συναρτήσεις καλούνται με το όνομα τους, στο δεξιό μέρος εντολών εκχώρησης ή μέσα σε παραστάσεις όπου χρειάζεται να γίνουν υπολογισμοί τιμών, η σε εντολές εκτύπωσης κ.λ.π.

Το πέρασμα τιμών στις συναρτήσεις μπορεί να γίνει είτε με αναφορά είτε με τιμή, όπως ακριβώς και στις υπορουτίνες.

Στο επόμενο παράδειγμα παρουσιάζεται μια απλή συνάρτηση που υπολογίζει το εμβαδόν ενός κύκλου δεχόμενη σαν παράμετρο την ακτίνα του :

```
Option Explicit
Function CalcArea(r As Integer) As Double
Const pi = 3.14
CalcArea = pi * r ^ 2
End Function

Private Sub Command1_Click()
Dim r As Integer
r = Text1.Text
Text2.Text = CalcArea(r)
End Sub
```



Σχήμα :23

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

- Μαθηματικές συναρτήσεις
- Αλφαριθμητικές συναρτήσεις
- Συναρτήσεις μετατροπής τύπων δεδομένων (δες σελ. 27, Παρ. 5.8)
- Συναρτήσεις ελέγχου τύπου και τιμής δεδομένων
- Συναρτήσεις ημερομηνιών

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

Συνάρτηση	Λειτουργία	Παράδειγμα
Abs(x)	Απόλυτη Τιμή	MyNumber = Abs(50.3) 'Επιστρέφει 50.3 MyNumber = Abs(-50.3) 'Επιστρέφει 50.3
Atn(x)	Τόξο εφαπτομένης	pi = 4 * Atn(1) 'Υπολογισμός της τιμής του π
Cos(x)	Συνημίτονο	MyAngle = 1.3 'Η γωνίες δίνονται σε ακτίνια MySecant = 1 / Cos(MyAngle) 'Υπολογισμός τέμνουσας
Exp(x)	Εκθετικό e^x	MyAngle = 1.3 ' Υπολογισμός υπερβολικού ημιτόνου MyHSin = (Exp(MyAngle) - Exp(-1 * MyAngle)) / 2
Fix(x)	Ακέραιο μέρος	MyNumber = Fix(99.2) 'Επιστρέφει 99 MyNumber = Fix(-99.8) 'Επιστρέφει -99
Int(x)	Ακέραιο μέρος	MyNumber = Int(99.8) 'Επιστρέφει 99 MyNumber = Int(-99.8) 'Επιστρέφει -100
Log(x)	Λογάριθμος νεπέριος lnx	MyAngle = 1.3 ' Υπολογισμός αντιστρόφου υπερβολικού ημιτόνου MyLog = Log(MyAngle + Sqr(MyAngle * MyAngle + 1))
Rnd(x)	Τυχαίος αριθμός, από 0 ως 1	Randomize 'Ενεργοποίηση γεννήτριας τυχαίων αριθμών MyValue = Int((6 * Rnd) + 1) 'Δημιουργία τυχαίων αριθμών από 1 ως 6
Sqn(x)	Πρόσημο αριθμού	MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0 MySign = Sgn(MyVar1) 'Επιστρέφει 1 MySign = Sgn(MyVar2) 'Επιστρέφει -1 MySign = Sgn(MyVar3) 'Επιστρέφει 0
Sin(x)	Ημίτονο	MyAngle = 1.3

		MyCosecant = 1 / Sin(MyAngle) 'Υπολογισμός συντέμνουσας
Sqr(x)	Τετραγωνική ρίζα ($\sqrt{x}$)	MySqr = Sqr(4) 'Επιστρέφει 2 MySqr = Sqr(23) 'Επιστρέφει 4.79583152331272 MySqr = Sqr(0) 'Επιστρέφει0 MySqr = Sqr(-4) 'Προκαλεί run-time error
Tan(x)	Εφαπτομένη	MyAngle = 1.3 MyCotangent = 1 / Tan(MyAngle) 'Υπολογισμός συνεφαπτομένης

Θα πρέπει να σημειωθεί ότι οι γωνίες σε όλες τις τριγωνομετρικές συναρτήσεις δίνονται σε ακτίνια και όχι σε μοίρες. Οι τύποι που συνδέουν μοίρες και ακτίνια είναι :

Μοίρες = Ακτίνια * 180 / π & Ακτίνια = Μοίρες * π / 180

Από τις μαθηματικές αυτές συναρτήσεις μπορεί να δημιουργηθούν πολλές άλλες που να υπολογίζουν διάφορα άλλα μαθηματικά μεγέθη. Σαν παράδειγμα, ακολουθούν μερικές συναρτήσεις οριζόμενες από το χρήστη που εκτελούν τέτοιους υπολογισμούς :

ΜΕΓΙΣΤΟΣ ΚΟΙΝΟΣ ΔΙΑΙΡΕΤΗΣ **(**ΜΚΔ**)** ΔΥΟ ΑΚΕΡΑΙΩΝ

Η συνάρτηση MD βασίζεται στον Ευκλείδειο αλγόριθμο εύρεσης του ΜΚΔ.

```
Public Function MD(x, y As Integer) As Integer
Dim z As Integer
z = y
Do While (z <> 0)
   z = x Mod y
   x = y
   y = z
Loop
MD = x
End Function
```

ΕΛΑΧΙΣΤΟ ΚΟΙΝΟ ΠΟΛΛΑΠΛΑΣΙΟ **(**ΕΚΠ**)** ΔΥΟ ΑΚΕΡΑΙΩΝ

Η συνάρτηση MM έχει άμεση συνάφεια με την αμέσως προηγούμενη συνάρτηση και βασίζεται στη σχέση ΕΚΠ **x** ΜΚΔ **=** ΓΙΝΟΜΕΝΟ ΤΩΝ ΑΚΕΡΑΙΩΝ**.**

```
Public Function MM(x, y As Integer) As Integer
If MD(x, y) <> 0 Then
   MM = x * y / MD(x, y)
Else
   MM = 0
End If
End Function
```

ΠΑΡΑΓΟΝΤΙΚΟ

```
Public Function Factorial(x As Integer) As Long
Dim n As Integer
Factorial = 1
For n = 2 To x
   Factorial = Factorial * n
Next n
End Function
```

Γνωρίζοντας ότι $n! = 1 \times 2 \times \ldots \times n$ μπορούμε να κατανοήσουμε πολύ εύκολα τον τρόπο λειτουργίας της συνάρτησης Factorial. Βασίζεται σε μια πάρα πολύ απλή επαναληπτική δομή και είναι ιδιαίτερα χρήσιμη για εφαρμογές που σχετίζονται με συγκεκριμένα μαθηματικά πεδία.

ΜΟΙΡΕΣ ΣΕ ΑΚΤΙΝΙΑ **/** ΑΚΤΙΝΙΑ ΣΕ ΜΟΙΡΕΣ

```
Public Function Degree(Radian As Currency) As
Double
Degree = Radian * 180 / pi
End Function

Public Function Radian(Degree As Currency) As
Double
Radian = Degree * pi / 180
End Function
```

Δύο συναρτήσεις με άμεση συνάφεια. Η Radian που δέχεται μέτρο μιας γωνίας σε μοίρες και το αποδίδει σε ακτίνια και η Degree που εκτελεί το ακριβώς αντίστροφο.

Και οι δύο βασίζονται στον τύπο $\frac{rad}{\deg} = \frac{\pi}{180^\circ}$.

Αλφαριθμητικές συναρτήσεις

Συνάρτηση	Λειτουργία	Παράδειγμα
Asc(a)	Ο Κωδικός ASCII ενός χαρακτήρα	MyNumber = Asc("A") 'Επιστρέφει 65 MyNumber = Asc("a") 'Επιστρέφει 97
Chr(a)	Ο χαρακτήρας που αντιστοιχεί σε έναν κωδικό ASCII	MyChar = Chr(97) 'Επιστρέφει a MyChar = Chr(62) 'Επιστρέφει >
Instr(x,a,b)	Η θέση του string b μέσα στο string a, αρχίζοντας από τη θέση x	MyPos = Instr(1, "F-16", "16") 'Επιστρέφει 3 MyPos = Instr(1, "Mirage", "age") 'Επιστρέφει 4
LCase(a)	Μετατροπή των χαρακτήρων του string a σε πεζά	Lowercase = Lcase("Hello World 1234") 'Επιστρέφει "hello world 1234"
Len(x)	Το πλήθος των χαρακτήρων του	MyLen = Len("Hello World") 'Επιστρέφει 11

	string x	
Left(a,x)	Οι x πρώτοι χαρακτήρες του string a	MyStr = Left("Hello World",7) 'Επιστρέφει "Hello W"
LTrim(a)	Αφαιρεί τα διαστήματα από την αρχή του string a	TrimString = LTrim(" <-Trim-> ") 'Επιστρέφει = "<-Trim-> "
Mid(a,x,y)	y χαρακτήρες του string a, αρχίζοντας από τον Χιοστό	FirstWord = Mid("Mid Function Demo", 1, 3) 'Επιστρέφει "Mid" LastWord = Mid("Mid Function Demo", 14, 4) 'Επιστρέφει "Demo"
Right(a,x)	Οι x τελευταίοι χαρακτήρες του string a	MyStr = Right("Hello World" , 1) 'Επιστρέφει "d" MyStr = Right("Hello World" , 6) 'Επιστρέφει " World"
RTrim(a)	Αφαιρεί τα διαστήματα από το τέλος του string a	TrimString = RTrim(" <-Trim-> ") 'Επιστρέφει = " <-Trim->"
Space(x)	Δημιουργεί ένα string που αποτελείται από x διαστήματα	MyString = Space(10) 'Επιστρέφει ένα string με 10 διαστήματα
String(x,a)	Δημιουργεί ένα string που αποτελείται από x όμοιους χαρακτήρες a	MyString = String(5, "*") 'Επιστρέφει "*****"
Trim(a)	Αφαιρεί τα κενά διαστήματα από την αρχή και το τέλος του string a	TrimString = Trim(" <-Trim-> ") 'Επιστρέφει = "<-Trim->"
Ucase(a)	Μετατροπή των χαρακτήρων του string a σε κεφαλαία	UpperCase = UCase("Hello World 1234") 'Επιστρέφει "HELLO WORLD 1234"

Ημερολογιακές συναρτήσεις

Συνάρτηση	Λειτουργία	Παράδειγμα
Date()	Η τρέχουσα ημερομηνία	Dim MyDate MyDate = Date ' MyDateέχει την ημερομηνία συστήματος
DateSerial(y,m,d)	Ημερομηνία από έτος, μήνα, μέρα	MyDate = DateSerial(1969, 2, 12) 'Επιστρέφει την ημερομηνία 12 Φεβρουαρίου 1969
DateValue(v)	Μετατροπή σε ημερομηνία	MyDate = DateValue("February 12, 1969") Επιστρέφει ημερομηνία

Day(d)	Ημέρα χρονικής στιγμής	MyDate = #February 12, 1969# 'Δημιουργία ημερομηνίας
		MyDay = Day(MyDate) ' MyDayείναι 12

Hour(d)	Ωρα χρονικής στιγμής	MyTime = #4:35:17 PM# 'Δημιουργία ώρας MyHour = Hour(MyTime) ' MyHourείναι 16
Minute(d)	Λεπτά χρονικής στιγμής	MyTime = #4:35:17 PM# 'Δημιουργία ώρας MyMinute = Minute(MyTime) ' MyMinuteείναι 35

Month(d)	Μήνας χρονικής στιγμής	MyDate = #February 12, 1969# 'Δημιουργία ημερομηνίας
		MyMonth = Month(MyDate) ' MyMontheίναι 2

Now()	Τρέχουσα χρονική στιγμή	Dim Today Today = Now() 'Τρέχουσα ημερομηνία και ώρα συστήματος
Second(d)	Δευτερόλεπτα χρονικής στιγμής	MyTime = #4:35:17 PM# 'Δημιουργία ώρας MySecond = Second(MyTime) ' MySecondείναι 17
Timer	Τα δευτερόλεπτα που πέρασαν από τα μεσάνυχτα	Start = Timer ' Set start time. Do While Timer < Start + PauseTime DoEvents ' Yield to other processes Loop Finish = Timer ' Set end time TotalTime = Finish - Start ' Calculate total time
TimeSerial (h,m,s)	Ωρα από ώρες, λεπτά, δευτερόλεπτα	MyTime = TimeSerial(16, 35, 17) Η MyTime περιέχει την ώρα 4:35:17 PM
TimeValue(v)	Μετατροπή σε ώρα	MyTime = TimeValue("4:35:17 PM") Επιστρέφει μια ώρα

Year(d)	Ετος χρονικής στιγμής	MyDate = #February 12, 1969# 'Δημιουργία ημερομηνίας
		MyYear = Year(MyDate) ' MyYearείναι 1969

WeekDay(d)	Ημέρα της εβδομάδος που αντιστοιχεί σε μια ημερομηνία	MyDate = #February 12, 1969# ' Assign a date.
		MyWeekDay = Weekday(MyDate) ' MyWeekDayπεριέχει 4 δηλαδή Τετάρτη

IsDate(v)	Ελέγχει αν η παράμετρος αποτελεί ημερομηνία	MyDate = "February 12, 1969": YourDate = #2/12/69# NoDate = "Hello" MyCheck = IsDate(MyDate) 'Επιστρέφει True. MyCheck = IsDate(YourDate) 'Επιστρέφει True. MyCheck = IsDate(NoDate) 'Επιστρέφει False

7. Συμβάντα πληκτρολογίου & ποντικιού

7.1 Συμβάντα πληκτρολογίου

Όταν ο χρήστης πιέσει ένα πλήκτρο του πληκτρολογίου σε μια εφαρμογή της Visual Basic, είτε πάνω σε μια φόρμα είτε πάνω σε κάποιο στοιχείο ελέγχου, ενεργοποιούνται 3 ειδών συμβάντα, το **KeyPress**, το **KeyDown** και το **KeyUp**. Οι event procedures των συμβάντων αυτών έχουν ως εξής :

- **Private Sub ControlName_KeyPress(KeyAscii As Integer)** ή
- **Private Sub Form_KeyPress(KeyAscii As Integer)**

  Η παράμετρος **KeyAscii** περιέχει τον κωδικό ASCII του πλήκτρου που πατήθηκε. Το συμβάν αυτό δεν ενεργοποιείται όταν πατηθούν τα πλήκτρα Shift, Alt, Ctrl και τα Function Keys.

- **Private Sub ControlName_KeyDown(KeyCode As Integer, Shift As Integer)** ή
- **Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)**

Πλήκτρο	Τιμή της Shift
Shift	1
Ctrl	2
Alt	4
Shift + Ctrl	3
Shift + Alt	5
Ctrl + Alt	6
Shift + Ctrl + Alt	7

Το συμβάν ενεργοποιείται όταν το πλήκτρο παραμένει πατημένο. Εδώ η παράμετρος **Shift** δηλώνει αν πατήθηκαν τα πλήκτρα Alt, Shift, Ctrl ή συνδυασμός τους (δες πίνακα) και η παράμετρος **KeyCode** δηλώνει ποιο πλήκτρο πατήθηκε.

- **Private Sub ControlName_KeyUp(KeyCode As Integer, Shift As Integer)** ή
- **Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)**

  Το συμβάν ενεργοποιείται όταν το πλήκτρο που πατήθηκε αφεθεί. Η σημασία των παραμέτρων **Shift** και **KeyCode** είναι η ίδια με το προηγούμενο συμβάν.

Τα συμβάντα αυτά μοιάζουν αρκετά μεταξύ τους. Στην πράξη όταν μας ενδιαφέρει το ποιος χαρακτήρας πατήθηκε χρησιμοποιούμε το KeyPress ενώ αν μας ενδιαφέρει το ποιο πλήκτρο πατήθηκε χρησιμοποιούμε την KeyDown.

7.2. Συμβάντα **mouse**

Τα σπουδαιότερα συμβάντα που συνδέονται με το ποντίκι είναι τα **Click**, **DblClick**, **MouseDown**, **MouseUp** και **MouseMove**. Η περιγραφή και οι event procedures των συμβάντων αυτών έχουν ως εξής :

Click & DBlClick: Τα συμβάντα αυτά συμβαίνουν όταν ο χρήστης πιέζει και ελευθερώνει ένα πλήκτρο του ποντικιού, ή κάνει double click, αντίστοιχα, πάνω σε κάποιο στοιχείο ελέγχου ή πάνω σε μια φόρμα, σε σημείο που δεν υπάρχουν ενεργά στοιχεία ελέγχου. Επίσης, μπορεί να δημιουργηθεί και τεχνητά αλλάζοντας την τιμή κάποιου στοιχείου ελέγχου, για παράδειγμα θέτοντας τιμή true στην ιδιότητα Value ενός CommandButton. Η σύνταξη των event procedures για φόρμα και στοιχείο ελέγχου είναι αντίστοιχα :

- **Private Sub Form_Click( ) & Private Sub Form_DblClick( )**
- **Private Sub ControlName _Click(**[index As Integer]**)** & **Private Sub ControlName _DblClick(**[index As Integer]**)**
  όπου το index αναγνωρίζει το στοιχείο ελέγχου όταν αυτό είναι σε Control Array.

MouseDown & MouseUp : Τα συμβάντα αυτά συμβαίνουν όταν ο χρήστης πιέσει (**MouseDown**) ή ελευθερώσει (**MouseUp**) ένα πλήκτρο ποντικιού πάνω σε φόρμα ή στοιχείο ελέγχου. Οι αντίστοιχες event procedures είναι :

- **Private Sub Form_MouseDown(button As Integer, shift As Integer, x As Single, y As Single)**
- **Private Sub ControlName_MouseDown([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)**

Επίσης:

- **Private Sub Form_MouseUp(button As Integer, shift As Integer, x As Single, y As Single)**
- **Private Sub ControlName_MouseUp([index As Integer,]button As Integer, shift As Integer, x As Single, y As Single)**

Όπου οι παράμετροι έχουν τις εξής σημασίες :

index : αναγνωρίζει το στοιχείο ελέγχου όταν αυτό είναι σε Control Array.

button : επιστρέφει τις τιμές 1,2,4 ανάλογα αν πατήθηκε το αριστερό, δεξί ή μεσαίο πλήκτρο του ποντικιού αντίστοιχα.

Shift : επιστρέφει έναν ακέραιο που δείχνει αν ήταν πατημένα και τα πλήκτρα ALT, CTRL, SHIFT ή οποιοσδήποτε συνδυασμός τους.

X,Y : οι συντεταγμένες του ποντικιού την στιγμή που πατήθηκε το πλήκτρο.

MouseMove : Το συμβάν αυτό συμβαίνει συνεχώς όταν και για όσο χρόνο ο χρήστης κινεί το ποντίκι πάνω σε μια φόρμα ή στοιχείο ελέγχου. Οι αντίστοιχες event procedures είναι :

- **Private Sub Form_MouseMove(button As Integer, shift As Integer, x As Single, y As Single)**
- **Private Sub ControlName_MouseMove([index As Integer,] button As Integer, shift As Integer, x As Single, y As Single)**

Όπου οι παράμετροι έχουν την ίδια σημασία με το προηγούμενο συμβάν.

Όταν μας ενδιαφέρει μόνο το αν πατήθηκε το ποντίκι, τότε συνήθως χρησιμοποιούμε το συμβάν Click. Αν μας ενδιαφέρει η ανίχνευση του ποιο πλήκτρο του ποντικιού πατήθηκε, τότε χρησιμοποιούμε το συμβάν MouseDown. Αν μας ενδιαφέρει και το αν το ποντίκι κινείται, τότε χρησιμοποιούμε το συμβάν MouseMove.

8. Διαχείριση Βάσεων Δεδομένων στη **VB**

Στις εφαρμογές της Visual Basic μπορούν να χρησιμοποιηθούν έτοιμα δεδομένα που προέρχονται από συστήματα διαχείρισης βάσεων δεδομένων. Η γλώσσα παρέχει δυνατότητα πρόσβασης σε αρχεία βάσεων δεδομένων διαφόρων μορφών, όπως Access, Excel, dBase, FoxPro, Paradox, text κ.α.

Ο απλούστερος τρόπος προσπέλασης μιας βάσης δεδομένων είναι μέσω του εργαλείου Δεδομένων (Data), το οποίο θα αναλύσουμε εδώ. Για πιο σύνθετες εφαρμογές η Visual Basic μπορεί να συνδεθεί με ισχυρές μηχανές διαχείρισης δεδομένων όπως ένας SQL Server ή να χρησιμοποιηθούν οι τεχνολογίες ADO και DAO.

Το εργαλείο Δεδομένων επιτρέπει την σύνδεση της Visual Basic με μια βάση δεδομένων, χρησιμοποιώντας την μηχανή **Microsoft Jet Database engine** – ίδια με αυτή της Microsoft Access. Με το εργαλείο αυτό έχουμε πρόσβαση σε διάφορες βάσεις δεδομένων (MS Access, dBase, MS FoxPro, Paradox, Btrieve, MS Excel, Lotus 1-2-3, ASCII αρχεία, MS SQL & Oracle μέσω ODBC) χωρίς να χρειάζεται να γράψουμε κώδικα.

Σχήμα :24

Οι εργασίες που μπορούμε να πραγματοποιήσουμε με το εργαλείο **DATA** είναι η εμφάνιση, προσθήκη και διόρθωση εγγραφών μιας βάσης δεδομένων. Η μορφή του εργαλείου είναι 4 κουμπιά που μας μετακινούν στην πρώτη, τελευταία, επόμενη και προηγούμενη εγγραφή μιας βάσης και ένα text box που αναγράφει τον αριθμό της τρέχουσας εγγραφής.

Οι πιο χρήσιμες ιδιότητες του εργαλείου Data είναι οι ακόλουθες :

Ιδιότητα	Περιγραφή
Connect	Προσδιορίζει τον τύπο της Β. Δ. και ίσως αν χρειαστεί το συνθηματικό
Databasename	Το όνομα και η διαδρομή του αρχείου Β.Δ. που θα χρησιμοποιήσουμε, πχ. **C:\Chris\EEF\documents\old_files\access\BD_eef_ChrisTriantafyllou.mdb**
Exclusive	Με την ιδιότητα αυτή, στην επιλογή true, απαιτούμε αποκλειστικό έλεγχο στη Β. Δ.
Options	Ορίζοντας τιμές στην ιδιότητα Options, μπορούμε να επιτύχουμε έλεγχο σε ότι αφορά στην πρόσβαση στους πίνακες της βάσης δεδομένων. Μέσω αυτών των επιλογών μπορούμε να ορίσουμε απαγόρευση πρόσβασης σε κάποιους πίνακες οι οποίοι περιέχουν τις εγγραφές εκείνες που θέλουμε να χαρακτηρίσουμε σαν μη προσβάσιμες
Readonly	Αν μέσω μίας εφαρμογής της Visual Basic θέλουμε να ανοίξουμε μία βάση δεδομένων, την οποία δεν πρόκειται να τροποποιήσουμε σε ότι αφορά την δομή ή τα δεδομένα της, τότε μπορούμε να θέσουμε στην ιδιότητα Readonly την τιμή True\
RecordSource	Η ιδιότητα RecordSource προσδιορίζει πού θα βρεθούν τα δεδομένα της βάσης δεδομένων που θα χρησιμοποιηθεί στην εφαρμογή της Visual Basic. Η ιδιότητα αυτή δεν έχει νόημα παρά μόνο αφού ανοιχτεί η βάση δεδομένων

Τα βήματα για την ενσωμάτωση ενός εργαλείου δεδομένων σε μια φόρμα είναι τα ακόλουθα :

1. Σχεδιάζουμε οπτικά το εργαλείο πάνω στη φόρμα.
2. Στο παράθυρο ιδιοτήτων θέτουμε την ιδιότητα **Connect** στον τύπο της βάσης δεδομένων που θα χρησιμοποιήσουμε (Access, Excel, dBase κλπ).
3. Θέτουμε στην ιδιότητα **DatabaseName** το όνομα του αρχείου της βάσης δεδομένων. Η βάση μπορεί να είναι τοπική ή να βρίσκεται σε κάποιο server.
4. Θέτουμε στην ιδιότητα **RecordSource** το όνομα του πίνακα της βάσης δεδομένων που θα χρησιμοποιήσουμε.
5. Προαιρετικά, μπορούμε να ορίσουμε τις ιδιότητες **Exclusive**, που καθορίζει αν θα έχουμε την αποκλειστική χρήση μιας βάσης ή όχι, και **ReadOnly**, που καθορίζει αν η βάση θα ανοιχτεί για ανάγνωση μόνο ή όχι.

Εν συνεχεία, σχεδιάζουμε τα εργαλεία που θα εμφανίζουν τα περιεχόμενα του πίνακα και τα συνδέουμε με το εργαλείο δεδομένων :

6. Σχεδιάζουμε ένα πλαίσιο κειμένου στη φόρμα.
7. Θέτουμε στην ιδιότητα **DataSource** το όνομα του εργαλείου δεδομένων που ορίσαμε πιο πριν.
8. Θέτουμε στην ιδιότητα **DataField** το όνομα του πεδίου (ή της στήλης) που θέλουμε να δούμε ή να τροποποιήσουμε.
9. Επαναλαμβάνουμε τα βήματα 5,6 και 7 για κάθε άλλο πεδίο που θέλουμε να εμφανίσουμε. Εναλλακτικά, εκτός από πλαίσια κειμένου μπορούν να χρησιμοποιηθούν και άλλα εργαλεία για την εμφάνιση των πεδίων.

Για να δείξουμε τη δυνατότητα χρήσης διαφόρων εργαλείων για την απεικόνιση των δεδομένων, χρησιμοποιούμε ένα scroll bar, το **Hscroll1** το οποίο μας παρέχει οπτικά μια ένδειξη των ωρών ανάθεσης κάθε εκπαιδευτή σε σχέση με μια μέγιστη τιμή (15 ώρες). Για την ενημέρωσή του το έχουμε συνδέσει με τις αλλαγές στο περιεχόμενο του text7 το οποίο δίνει το σύνολο ωρών ανάθεσης μέσω της event procedure

```
Private Sub Text7_Change()

    HScroll1.Value = Text7.Text

End Sub
```

Εναλλακτικά, μπορεί να χρησιμοποιηθεί και το συμβάν Data_Validate του εργαλείου Data1. Η αντίστοιχη event procedure θα είναι :

```
Private Sub Data1_Validate(Action As Integer, Save As Integer)

    HScroll1.Value = Text7.Text

End Sub
```

Όπου η παράμετρος action καθορίζει με ακρίβεια το συμβάν που προκάλεσε το Data1_Validate, όπως η εκτέλεση των μεθόδων MoveFirst, MoveNext, MovePrevious, MoveLast, AddNew, Delete, Close κλπ.

9. Γραφικά και Πολυμέσα στη **Visual Basic**

9.1 Γενικά για τα γραφικά

Η Visual Basic διαθέτει δύο διαφορετικούς τρόπους για τη δημιουργία γραφικών σε μία εφαρμογή. Μπορούμε να χρησιμοποιήσουμε είτε τα εργαλεία γραφικών είτε μεθόδους γραφικών. Σχεδιάζουμε γραμμές, πλαίσια, κύκλους, ορθογώνια και άλλα γεωμετρικά σχήματα και μορφοποιούμε την εμφάνιση τους επάνω στην φόρμα.

Κάθε project με γραφικά χρησιμοποιεί ένα σύστημα συντεταγμένων. Με το σύστημα αυτό συντεταγμένων μπορούμε να προσδιορίσουμε σημεία στην οθόνη, σε μία φόρμα ή οπουδήποτε. Η μορφή που χρησιμοποιείται για να οριστεί ένα σημείο είναι η (x,y). Το χ είναι η τιμή της προβολής του σημείου στον οριζόντιο άξονα των x και το y η αντίστοιχη τιμή της προβολής του στον κατακόρυφο άξονα των y.

Κανόνες στο σύστημα συντεταγμένων της Visual Basic :

- Αν σχεδιάζουμε ένα πλαίσιο εικόνας πάνω σε μία φόρμα, το σύστημα συντεταγμένων της φόρμας, είναι αυτό που θα καθορίζει οτιδήποτε αφορά τη μετακίνηση ή τις αλλαγές του μεγέθους του πλαισίου εικόνας.
- Οι εντολές που χρησιμοποιούνται για τον επαναπροσδιορισμό του μεγέθους ή τη μετακίνηση μιας φόρμας, εκφράζουν πάντοτε τη θέση και το μέγεθος της φόρμας σε twips.
- Η επάνω αριστερή γωνία της οθόνης έχει τιμές συντεταγμένων (0,0).
- Η επάνω αριστερή γωνία κάθε φόρμας, πλαισίου εικόνας ή πλαισίου, έχει εξ ορισμού τιμές (0, 0).
- Έχουμε διάφορες κλίμακες για τη μέτρηση των σημείων κατά μήκος των δύο αξόνων. Δηλαδή η Visual Basic μπορεί να υποστηρίξει στο σύστημα συντεταγμένων διαφορετική κλίμακα για κάθε άξονα.

Όπως αναφέραμε πριν, η Visual Basic εξ ορισμού χρησιμοποιεί, στις μετακινήσεις και στα μεγέθη γραφικών, σαν μονάδα μέτρησης το **twip**. Ένα twip είναι το 1/20 ενός σημείου εκτύπωσης ή διαφορετικά 1440 twips ισούνται με μία ίντσα. Αυτές οι αντιστοιχίες αναφέρονται στο μέγεθος του αντικειμένου κατά την εκτύπωση. Σε ότι αφορά την εμφάνιση του αντικειμένου στην οθόνη, οι φυσικές αποστάσεις ποικίλουν ανάλογα με το μέγεθος της οθόνης.

9.2 Κλίμακες μετρήσεων

Ο χρήστης σχετικά με το σύστημα συντεταγμένων μπορεί είτε να χρησιμοποιήσει την εξ ορισμού κλίμακα μέτρησης, δηλαδή τα twips, είτε να χρησιμοποιήσει μία από τις υπόλοιπες έτοιμες κλίμακες που παρέχει η Visual Basic είτε να δημιουργήσει μία δική του κλίμακα.

Ο απλούστερος τρόπος να ορίσουμε το σύστημα συντεταγμένων είναι να αποδώσουμε μία τιμή στην ιδιότητα **ScaleMode**. Οι δυνατές τιμές της ιδιότητας, η οποία αναφέρεται στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής, φαίνονται στον παρακάτω πίνακα .

Τιμή της ScaleMode	Περιγραφή
0-User	Δηλώνει πως μία ή περισσότερες από τις τιμές των ιδιοτήτων ScaleHeight, ScaleWidth, ScaleLeft και ScaleTop ορίζονται από το χρήστη
1 - Twip	Η εξ ορισμού κλίμακα, όπου 567 twips = 1 cm
2 - Point	Όπου 1 inch = 72 σημεία (points)
3 - Pixel	Όπου σαν pixel ορίζεται η μικρότερη μονάδα ανάλυσης της οθόνης.
4 - Character	Όπου 1 χαρακτήρας (char) αναπαρίσταται μέσα σε ορθογώνιο πλαίσιο, οριζόντιας πλευράς 120 twips και κατακόρυφης πλευράς 240 twips
5 - Inch	Ίντσα, όπου 1 inch = 2,54 cm.
6 -Millimeter	Χιλιοστό του μέτρου
7- Centimeter	Εκατοστό του μέτρου

Όταν αποδίδουμε μία από τις δυνατές τιμές (πλην της 0) στην ιδιότητα ScaleMode, η Visual Basic επαναπροσδιορίζει τις τιμές των ιδιοτήτων ScaleWidth και ScaleHeight, αποδίδοντας τους τιμές που να αντιστοιχούν στη νέα κλίμακα συντεταγμένων. Ταυτόχρονα μηδενίζονται οι τιμές των ιδιοτήτων ScaleTop και ScaleLeft.

Επίσης αλλάζουν οι τιμές των ιδιοτήτων CurrentX και CurrentY, έτσι ώστε να εκφράζουν τις συντεταγμένες του τρέχοντος σημείου στο καινούριο σύστημα συντεταγμένων.

Ένας άλλος τρόπος ορισμού του συστήματος συντεταγμένων είναι χρησιμοποιώντας τις συσχετιζόμενες ιδιότητες ScaleHeight, ScaleWidth, ScaleLeft καιScaleTop. Αποδίδοντας τιμή σε τουλάχιστον μία από αυτές, μπορούμε να δημιουργήσουμε ένα δικό μας σύστημα συντεταγμένων για το αντικείμενο στο οποίο αναφερόμαστε, θέτοντας όμως τιμή σε οποιαδήποτε από τις παραπάνω ιδιότητες, αυτόματα τίθεται η τιμή 0 στην ιδιότητα ScaleMode.

Στην κλίμακα που δημιουργεί ο χρήστης, μπορεί ν' αποδώσει στην επάνω αριστερή γωνία τόσο της φόρμας όσο και του αντικειμένου που περιέχεται σ' αυτήν, οποιοδήποτε ζεύγος τιμών θέλει, και όχι υποχρεωτικά το ζεύγος τιμών (0,0).

Τέλος σημειώνουμε πως με τη χρήση της μεθόδου Scale μπορούμε επίσης να αποδώσουμε τιμές στις ιδιότητες ScaleHeight, ScaleWidth, ScaleLeft και ScaleTop, πράγμα που σημαίνει, όπως ήδη εξηγήθηκε, τον ορισμό μιας καινούριας κλίμακας συντεταγμένων.

9.3 Εργαλεία γραφικών **& Χ**ρώματα

Η Visual Basic υποστηρίζει 256 χρώματα, με την προϋπόθεση βέβαια πως το σύστημα απεικόνισης του υπολογιστή μπορεί να τα αποδώσει. Η ανάγκη για ταυτόχρονη απεικόνιση 256 διαφορετικών χρωμάτων, εμφανίζεται κύρια σε Multimedia εφαρμογές και στις εφαρμογές εκείνες που χειρίζονται εικόνες με μεγάλες απαιτήσεις στη χρωματική απόδοση.

Μπορούμε να χρησιμοποιήσουμε 256 χρώματα σε μεθόδους γραφικών που επιδρούν σε φόρμες, αντικείμενα πλαισίου εικόνας και αντικείμενα εικόνας, θα πρέπει όμως να σημειωθεί πως για αρχεία εικόνας τύπου Metafile, η Visual Basic δεν υποστηρίζει 256 χρώματα, αλλά μόνο τα 16 της QuickBasic.

Κάθε χρώμα στη Visual Basic αντιπροσωπεύεται από ένα long integer. Η γλώσσα διαθέτει δύο διαφορετικούς τρόπους για τον προσδιορισμό των απαιτούμενων χρωμάτων μιας εφαρμογής:

- Με χρήση της συνάρτησης **RGB**, π.χ. Form1.BackColor=RGB(0,255,0) για το πράσινο
- Με χρήση της συνάρτησης **QBColor** που παρέχει τη δυνατότητα επιλογής ενός από τα 16 χρώματα που υποστηρίζει η QuickBasic. Π.χ. Form1.BackColor=QBColor(10) για το ανοικτό πράσινο

Στον πίνακα που ακολουθεί δίνονται οι δυνατές τιμές της QBColor και τα αντίστοιχα χρώματα.

Αριθμός - Χρώμα	Αριθμός - Χρώμα	Αριθμός - Χρώμα
0=Μαύρο	6=Κίτρινο	12=Ανοικτό κόκκινο
1=Μπλε	7=Άσπρο	13=Ανοικτό μοβ
2=Πράσινο	8=Γκρι	14=Ανοικτό κίτρινο
3=Κυανό	9=Ανοικτό μπλε	15=Εντονο άσπρο
4=Κόκκινο	10=Ανοικτό πράσινο	
5=Μωβ	11=Ανοικτό κυανό	

Η Visual Basic διαθέτει τρία εργαλεία που μπορούν να χρησιμοποιηθούν για τη δημιουργία γραφικών σε εφαρμογές. Τα εργαλεία αυτά είναι το εργαλείο **Image**, το εργαλείο **Line** και το εργαλείο **Shape**. Τα τρία αυτά εργαλεία γραφικών αποδεικνύονται πολύ χρήσιμα για τη δημιουργία γραφικών κατά τη σχεδίαση της εφαρμογής.

Ένα πλεονέκτημα τους είναι ότι απαιτούν την εγγραφή πολύ λιγότερου κώδικα απ' ότι απαιτείται αν χρησιμοποιηθούν οι μέθοδοι γραφικών. Για παράδειγμα, μπορούμε να σχεδιάσουμε έναν κύκλο χρησιμοποιώντας είτε το εργαλείο γεωμετρικού σχήματος, είτε τη μέθοδο Circle. Η μέθοδος Circle βέβαια απαιτεί την εγγραφή κώδικα, ενώ με τη χρησιμοποίηση του εργαλείου γεωμετρικό σχήμα δεν έχουμε παρά να "ζωγραφίσουμε" τον ζητούμενο κύκλο ορίζοντας την κατάλληλη τιμή στην ιδιότητα Shape του αντικειμένου.

Βέβαια από την άλλη πλευρά παρουσιάζουν και κάποια μειονεκτήματα :

- Τα αντικείμενα γραφικών δεν μπορούν να αναδυθούν μέσα από άλλα αντικείμενα, εκτός κι αν βρίσκονται μέσα σε ένα τρίτο που έχει αυτή την ικανότητα.
- Δεν μπορούν να συμπεριλάβουν μέσα τους άλλα αντικείμενα.
- Δεν μπορούν να υποστηρίξουν διαδικασίες εστίασης πάνω στο γραφικό κατά τη διάρκεια της εκτέλεσης της εφαρμογής.

9.4 Μέθοδοι γραφικών

Η Visual Basic παρέχει συμπληρωματικά προς τα εργαλεία γραφικών, μία σειρά από μεθόδους γραφικών οι οποίες είναι σχεδιασμένες ειδικά για τη δημιουργία γραφικών στις εφαρμογές της. Οι μέθοδοι γραφικών μπορούν να προσφέρουν μερικά ενδιαφέροντα οπτικά εφέ, που δεν μπορούν να δημιουργηθούν με τη χρήση των εργαλείων γραφικών. Γενικά, οι μέθοδοι γραφικών δεν ενδείκνυνται για τις περιπτώσεις εκείνες, που θέλουμε να δημιουργήσουμε πολύ απλά σχέδια στην παρουσίαση της εφαρμογής μας. Στις

περιπτώσεις δημιουργίας γραφικών με μεθόδους γραφικών, θα πρέπει να εκτελέσουμε την εφαρμογή, για να διαπιστώσουμε το οπτικό αποτέλεσμα.

Οι μέθοδοι γραφικών μπορούν να εφαρμοστούν στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής και είναι οι ακόλουθοι:

Cls	Καθαρίζει όλα τα γραφικά και κείμενα που δημιουργούνται κατά τη διάρκεια εκτέλεσης της εφαρμογής σε μία φόρμα ή σ' ένα πλαίσιο εικόνας. Μετά την εκτέλεση της, οι ιδιότητες CurrentX και CurrentY παίρνουν την τιμή 0. Σύνταξη : **[**Αντικείμενο **].Cls**
PSet	Προσδίδει το καθοριζόμενο χρώμα σ' ένα σημείο του αντικειμένου. Σύνταξη : **[**Αντικείμενο **].PSet [Step] (x, y), [color]** Το ζεύγος (x,y) προσδιορίζει τις συντεταγμένες του σημείου. Η προαιρετική παράμετρος step σημαίνει ότι το ζεύγος (x,y) αναφέρεται σε μετατόπιση από την τρέχουσα θέση και όχι σε απόλυτες συντεταγμένες. Αν παραληφθεί τέλος η παράμετρος Color, το σημείο παίρνει το χρώμα της ιδιότητας ForeColor του αντικειμένου.
Point	Επιστρέφει, έναν long integer που αναφέρεται στην RGB τιμή του χρώματος ενός συγκεκριμένου σημείου μιας φόρμας ή ενός πλαισίου εικόνας. Σύνταξη : **[**Αντικείμενο **].Point(x, y)** Το ζεύγος (x,y) προσδιορίζει τις συντεταγμένες του σημείου.
Line	Χρησιμοποιείται για τη σχεδίαση γραμμών και ορθογωνίων σχημάτων. Σύνταξη : **[**Αντικείμενο **].Line [Step] (x1, y1) [Step] - (x2, y2), [color], [B][F]** Τα ζεύγη τιμών (x1,y1) και (x2,y2) αναφέρονται στις συντεταγμένες του αρχικού και τελικού σημείου της γραμμής ή του επάνω-αριστερού και κάτω-δεξιού σημείου του ορθογωνίου. Οι προαιρετικές παράμετροι Step σημαίνουν ότι τα αντίστοιχα ζεύγη δεν αναφέρονται σε απόλυτες συντεταγμένες αλλά σε μετατοπίσεις από τις τρέχουσες θέσεις. Αν παραληφθεί η παράμετρος Color, η ευθεία ή το ορθογώνιο παίρνει το χρώμα της ιδιότητας ForeColor του αντικειμένου. Η προαιρετική παράμετρος B σημαίνει ότι θα σχεδιαστεί ορθογώνιο και όχι γραμμή. Η προαιρετική παράμετρος F χρησιμοποιείται μόνο με το B και δηλώνει ότι το ορθογώνιο θα είναι γεμάτο με χρώμα, ίδιο με αυτό της σχεδίασης του ορθογωνίου. Αν το B χρησιμοποιηθεί χωρίς το F, ο τύπος και το χρώμα γεμίσματος λαμβάνεται από τις ιδιότητες FillColor και FillStyle, με εξ' ορισμού τιμή το διαφανές χρώμα.
Circle	Χρησιμοποιείται για τη σχεδίαση καμπυλόγραμμων σχημάτων (κύκλου, έλλειψης ή τόξου). Σύνταξη : **[**Αντικείμενο **].Circle [Step] (x, y), radius, [color, start, end, aspect]** Το ζεύγος (x,y) αναφέρεται στις συντεταγμένες του κέντρου. Η προαιρετική παράμετρος step έχει ίδια σημασία με τις προηγούμενες μεθόδους. Η radius είναι η ακτίνα του σχήματος. Το color αναφέρεται στο χρώμα σχεδίασης. Οι παράμετροι start και end εκφράζονται σε ακτίνια, παίρνουν τιμές από –2π έως 2π, και σε περίπτωση σχεδίασης τμήματος κύκλου ή έλλειψης αναφέρονται στο σημείο έναρξης και λήξης της σχεδίασης. Η παράμετρος aspect καθορίζει την ελλειπτικότητα του σχήματος. Με τιμή 1 σχεδιάζεται κύκλος, ενώ με άλλες τιμές σχεδιάζεται έλλειψη.
Print	Αν και αναφέρεται στο αντικείμενο Debug μπορεί να θεωρηθεί σαν μέθοδος γραφικών. Η μέθοδος αυτή μπορεί να εμφανίσει στο παράθυρο Debug κείμενο που έχει αποδοθεί σαν τιμή σε μία μεταβλητή
PaintPicture	Σχεδιάζει τα περιεχόμενα ενός αρχείου γραφικών τύπου .ico, .wmf, .bmp και .dib. Εφαρμόζεται στα αντικείμενα φόρμα, πλαίσιο εικόνας και printer. Σύνταξη :

	[Αντικείμενο **].PaintPicture picture, x1, y1, [width1], [height1], [x2], [y2], [width2], [height2], [opcode]** Η παράμετρος picture αναφέρεται στο όνομα του αρχείου γραφικών, οι παράμετροι x1 και y1 στις συντεταγμένες του πάνω δεξιού σημείου της εικόνας, οι προαιρετικές παράμετροι width1 και height1 στο ύψος και πλάτος της εμφανιζόμενης εικόνας (με αρνητικές τιμές η εικόνα εμφανίζεται ανεστραμμένη), οι προαιρετικές παράμετροι x2, y2, width2, height2 αναφέρονται στον ορισμό clipping περιοχών, μια τεχνική που έχει να κάνει με τη βελτιστοποίηση της ταχύτητας εμφάνισης των γραφικών. Τέλος, η προαιρετική παράμετρος opcode ισχύει μόνο για Bitmap αρχεία γραφικών και εκτελεί μια επεξεργασία σε επίπεδο bit πάνω στην εικόνα.

9.5 Ιδιότητες γραφικών

Οι φόρμες και αρκετά άλλα αντικείμενα, χαρακτηρίζονται από μία σειρά από ιδιότητες γραφικών, οι ονομασίες και οι κατηγορίες των οποίων παρουσιάζονται στον παρακάτω πίνακα :

Ιδιότητες γραφικών ανά κατηγορία

Κατηγορία	Ιδιότητες
Τρεχουσών συντεταγμένων	CurrentX, CurrentY
Επεξεργασίας εμφάνισης	AutoRedraw, ClipControls
Τεχνικών σχεδίασης	DrawMode, DrawStyle, DrawWidth, BorderStyle, BorderWidth
Τεχνικών γεμίσματος	FillColor, FillStyle
Χρωματισμού	BackColor, ForeColor, BorderColor,FillColor
Ορισμού κλίμακας	ScaleLeft, ScaleTop, ScaleHeight.ScaleWidth, ScaleMode

Ιδιότητες τρεχουσών συντεταγμένων

Οι ιδιότητες **CurrentX** και **CurrentY** επιστρέφουν ή θέτουν τιμή στην οριζόντια και κατακόρυφη συντεταγμένη αντίστοιχα, του σημείου από το οποίο θα αρχίσει να εφαρμόζεται στη συνέχεια μία μέθοδος γραφικών ή μία διαδικασία εκτύπωσης. Οι ιδιότητες αυτές συναντώνται στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής. Δεν είναι όμως διαθέσιμες στη φάση της σχεδίασης, παρά μόνο στη φάση εκτέλεσης της εφαρμογής.

Η σύνταξη της εντολής είναι :

όνομα αντικειμένου **. CurrentX [=** χ**]** & όνομα αντικειμένου **.CurrentY [= y]**

όπου χ και y είναι αριθμοί που προσδιορίζουν τις συντεταγμένες θέσης. Οι αριθμοί αυτοί αντιστοιχούν σε twips, ή στην οποιαδήποτε άλλη κλίμακα μέτρησης χρησιμοποιείται.

Ιδιότητες επεξεργασίας εμφάνισης

Η ιδιότητα **AutoRedraw** (Αυτόματη Επανασχεδίαση) παίρνει τιμές boolean και χαρακτηρίζει κάθε αντικείμενο φόρμα και πλαίσιο εικόνας. Η AutoRedraw χρησιμοποιείται για την αυτόματη επανασχεδίαση επί της οθόνης γραφικών, τα οποία έχουν δημιουργηθεί με χρήση μεθόδων γραφικών, όταν στη φάση εκτέλεσης της εφαρμογής ολόκληρα γραφικά ή μέρος τους, αποκαλύπτονται μετά από μετακίνηση άλλων αντικειμένων που τα κάλυπταν γραφικά αλλάζουν το μέγεθος τους.

Η AutoRedraw έχει εξ ορισμού τιμή False. Αυτό σημαίνει πως κάθε γραφικό που σχηματίστηκε με τη χρήση μεθόδων γραφικών και εμφανίζεται πάνω στη φόρμα, θα χαθεί αν καλυφτεί προσωρινά από ένα άλλο αντικείμενο. Επίσης το ίδιο αποτέλεσμα απώλειας των γραφικών θα συμβεί αν μικρύνει τη φόρμα που τα περιλαμβάνει και στη συνέχεια την επανέλθει στις προηγούμενες διαστάσεις της. Όταν όμως η τιμή της AutoRedraw είναι True, τότε η Visual Basic αναλαμβάνει τη διαχείριση της οθόνης και την επανεμφάνιση των γραφικών όποτε χρειάζεται.

Ιδιότητες τεχνικών σχεδίασης

Η ιδιότητα **DrawWidth** (Πλάτος Σχεδίασης) προσδιορίζει το πλάτος της γραμμής σχεδίασης για τη μεθόδους γραφικών Circle, Cls, Line, PaintPicture, Point, Print και PSet. Εφαρμόζεται στα αντικείμενα φόρμα, πλαίσιο εικόνας, εκτυπωτής και OLE. Όταν η τιμή της DrawWidth είναι μεγαλύτερη από 1, τότε οι τιμές 1 έως 4 της ιδιότητας DrawStyle, δεν επιφέρουν το αποτέλεσμα που περιγράφουν.

Η ιδιότητα **BorderWidth** (Πλάτος Περιγράμματος) προσδιορίζει το πάχος του περιγράμματος των αντικειμένων γραμμή και γεωμετρικό σχήμα..

Η ιδιότητα **DrawStyle** (Είδος Σχεδίασης) εφαρμόζεται επί των αντικειμένων φόρμα, πλαίσιο εικόνας και εκτυπωτής και προσδιορίζει το τρόπο εμφάνισης των σχεδιαζόμενων γραμμών. Σημειώνεται πως αν η ιδιότητα DrawWidth πάρει τιμές μεγαλύτερες του Λ, τότε οι επιλογές 1 -4 της ιδιότητας DrawStyle παρέχουν το ίδιο αποτέλεσμα.

Η ιδιότητα **BorderStyle** (Είδος Περιγράμματος) έχει διαφορετικό σκοπό και χρήση στα ποικίλα αντικείμενα που απαντάται. Στα αντικείμενα γραμμή και γεωμετρικό σχήμα, η BorderStyle έχει την ίδια χρήση με αυτή που έχει η ιδιότητα DrawStyle, δηλαδή περιγράφει το είδος της σχεδιαζόμενης γραμμής. Στα αντικείμενα DBList και DBCombo, η BorderStyle προσδιορίζει κατά πόσο το αντικείμενο θα έχει απλό περίγραμμα (border), και αν ναι, αν θα εμφανίζονται διάφορα στοιχεία ενός παραθύρου, όπως τίτλος, πλήκτρα μεγιστοποίησης και ελαχιστοποίησης και αν το παράθυρο θα είναι σταθερού ή μεταβλητού μεγέθους. Στα αντικείμενα φόρμα, πλαίσιο εικόνας, εικόνα, πλαίσιο κειμένου, ετικέτα, πλέγμα και OLE η BorderStyle επιστρέφει ή θέτει τιμή που προσδιορίζει κατά πόσο το αντικείμενο θα έχει απλό περίγραμμα ή δεν θα έχει καθόλου.

Η ιδιότητα **DrawMode** (Τρόπος Σχεδίασης) προσδιορίζει την εμφάνιση των γραφικών που προέρχονται από τη χρησιμοποίηση μεθόδων γραφικών στα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής, καθώς επίσης και την εμφάνιση των αντικειμένων γραμμή και γεωμετρικό σχήμα. Μπορούμε να θεωρήσουμε ότι για τη σχεδίαση κάθε γραφικού μπορεί να χρησιμοποιηθεί ένας μεγάλος αριθμός από πένες σχεδίασης. Η χρήση κάθε μιας από αυτές επιφέρει και διαφορετικά αποτελέσματα στη σχεδίαση. Η τιμή της ιδιότητας DrawMode, προσδιορίζει ποιο είναι το οπτικό αποτέλεσμα αν συμβεί ένα γραφικό να δημιουργείται πάνω από ένα άλλο στη φάση της εκτέλεσης της εφαρμογής.

Ιδιότητες τεχνικών γεμίσματος

Η ιδιότητα **FillStyle** (Είδος Γεμίσματος) προσδιορίζει το εσωτερικό σχέδιο ενός αντικειμένου γεωμετρικό σχήμα ή ενός γραφικού που δημιουργείται με τη χρήση μεθόδων γραφικών πάνω σε ένα από τα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής.

Η ιδιότητα **FillColor** (Χρώμα Γεμίσματος) προσδιορίζει το εσωτερικό χρώμα ενός αντικειμένου γεωμετρικό σχήμα ή ενός γραφικού που δημιουργείται με τη χρήση των μεθόδων γραφικών Line και Circle πάνω σε ένα από τα αντικείμενα φόρμα, πλαίσιο εικόνας και εκτυπωτής. Η εξ ορισμού τιμή της ιδιότητας FillColor είναι 0 (Μαύρο), η δε επιλογή των χρωμάτων γίνεται με τη βοήθεια της παλέτας χρωμάτων. Όταν η ιδιότητα FillStyle έχει τιμή Transparent (Διαφανής), η τιμή της ιδιότητας FillColor αγνοείται.

Ιδιότητες χρωματισμού

Οι ιδιότητες **BackColor**(Χρώμα Παρασκηνίου) και **ForeColor**(Χρώμα Προσκήνιου) ορίζουν το χρώμα του παρασκηνίου και του προσκήνιου αντίστοιχα για το επιλεγμένο αντικείμενο. Οι δύο αυτές ιδιότητες αναφέρονται σε πάνω από 20 αντικείμενα, μεταξύ των οποίων η φόρμα, η ετικέτα, το πλαίσιο εικόνας, το πλαίσιο κειμένου κλπ.

Η ιδιότητα **BorderColor** (Χρώμα Περιγράμματος), αποδίδει χρώμα στο περίγραμμα ενός αντικειμένου. Η ιδιότητα αυτή χαρακτηρίζει τα αντικείμενα γεωμετρικό σχήμα και γραμμή.

Η απόδοση τιμής στις τρεις αυτές ιδιότητες μπορεί να γίνει στη φάση σχεδίασης της εφαρμογής μέσω του παράθυρου των ιδιοτήτων. Στη φάση της εκτέλεσης της εφαρμογής η απόδοση των τιμών γίνεται με τον εξής τρόπο σύνταξης :

Form1.BackColor = color Text1.ForeColor = color Shape1.BorderColor = color

όπου color είναι μία τιμή ή μία σταθερά που προσδιορίζει το χρώμα που αποδίδουμε στα εν λόγω αντικείμενα. Αν πρόκειται για τιμή, αυτή εκφράζεται είτε χρησιμοποιώντας τις χρωματικές συναρτήσεις RGB και QBColor, είτε χρησιμοποιώντας τη δεκαεξαδική έκφραση των χρωματικών αποχρώσεων της παλέτας χρωμάτων.

Ιδιότητες ορισμού κλίμακας

Οι ιδιότητες **ScaleWidth** (Πλάτος Κλίμακας) και **ScaleHeight** (Ύψος Κλίμακας) επιστρέφουν ή θέτουν τιμή στην οριζόντια και κατακόρυφη αντίστοιχα εσωτερική διάσταση ενός αντικειμένου όταν χρησιμοποιούνται σε αυτό μέθοδοι γραφικών. Λέγοντας εσωτερική διάσταση του αντικειμένου, εννοούμε ότι δεν συμπεριλαμβάνεται το πάχος της γραμμής πλαισίου. Οι ιδιότητες ScaleWidth και ScaleHeight εφαρμόζονται στα αντικείμενα φόρμα και πλαίσιο εικόνας και στο ειδικό αντικείμενο εκτυπωτής.

Οι ιδιότητες ScaleWidth και ScaleHeight διαφέρουν από τις ιδιότητες Width και Height, αφού οι δεύτερες αναφέρονται στις εξωτερικές διαστάσεις του αντικειμένου, συμπεριλαμβανομένου δηλαδή και του πάχους της γραμμής του πλαισίου του.

Οι ιδιότητες **ScaleLeft** (Αριστερό Κλίμακας) και **ScaleTop** (Κορυφή Κλίμακας) αποδίδουν τιμές στις, οριζόντια και κατακόρυφη αντίστοιχα, συντεταγμένες της επάνω αριστερής γωνίας ενός αντικειμένου όταν χρησιμοποιούνται μέθοδοι γραφικών. Οι ιδιότητες αυτές χαρακτηρίζουν τα αντικείμενα φόρμα, πλαίσιο εικόνας και το ειδικό αντικείμενο εκτυπωτής.

Οι ιδιότητες ScaleLeft και ScaleTop δεν είναι ταυτόσημες με τις ιδιότητες Left και Top οι οποίες απαντώνται σε περισσότερα από 20 αντικείμενα. Η ιδιότητα left δηλώνει την απόσταση μεταξύ της αριστερής εσωτερικής ακμής ενός αντικειμένου και της αριστερής ακμής του αντικειμένου που το περιέχει, συνηθέστερα της φόρμας. Αντίστοιχα, η ιδιότητα Top δηλώνει την απόσταση μεταξύ της επάνω εσωτερικής ακμής του αντικειμένου και της επάνω ακμής του αντικείμενου που το περιέχει. Για τη φόρμα οι τιμές των ιδιοτήτων left και Top εκφράζονται πάντα σε twips, ενώ για τα άλλα αντικείμενα εκφράζονται σύμφωνα με το σύστημα συντεταγμένων που ισχύει για το αντικείμενο στο οποίο περιέχονται.

10. Πολυμέσα **(MULTIMEDIA)**

Η Visual Basic μπορεί να συμπεριλάβει στις εφαρμογές της τόσο δεδομένα κειμένου, όσο και δεδομένα ήχου, εικόνας, προσομοίωσης κίνησης και video. Κύρια χάρη στις διαδικασίες **OLE** και **DDE** που υποστηρίζει, η Visual Basic μπορεί να επιλεγεί σαν εργαλείο συγγραφής (authoring tool), Interactive Multimedia και Hypermedia εφαρμογών. Η χρησιμοποίηση Multimedia δεδομένων στις εφαρμογές, συνεπάγεται μεγαλύτερες απαιτήσεις σε υλικό, τόσο σε ταχύτητα επεξεργασίας, όσο και σε μνήμη RAM, αλλά και σε αποθηκευτικές δυνατότητες. Τα αρχεία κυρίως video (πχ. ,avi) και προσομοίωσης κίνησης (πχ. .flc), αλλά και αυτά ήχου (πχ. wav) και στη συνέχεια εικόνας (πχ. .bmp) είναι πολύ μεγάλου μεγέθους. Κατά συνέπεια η χρησιμοποίηση τους χωρίς να έχουν εξασφαλιστεί οι απαιτούμενοι υλικοί πόροι του συστήματος, θα επιφέρει αρνητικά αποτελέσματα στην εφαρμογή. Ένα video που συμπεριλάβαμε σε μία εφαρμογή, αλλά στη φάση της εκτέλεσης εμφανίζεται με μία τρεμώδη και σπαστή κίνηση, σίγουρα δεν μπορεί να καταχωρηθεί στα θετικά στοιχεία της εφαρμογής.

Τα εργαλεία που παρέχει η Visual Basic προς χρήση για δημιουργία Multimedia εφαρμογών είναι αυτά τα οποία μπορούν να δεχτούν δεδομένα κειμένου, εικόνας, ήχου, animation και video. Αυτά τα εργαλεία είναι :

Εργαλείο πλαίσιο εικόνας (Picture Box): Το εργαλείο αυτό, μέσω της ιδιότητας του Picture, μπορεί να χρησιμοποιηθεί για την εμφάνιση εικόνων διαφορετικής μορφοποίησης (πχ. bmp, dib, wmf, ico κλπ). Αν η εικόνα είναι μεγαλύτερη από το αντικείμενο πλαίσιο εικόνας, τότε ένα τμήμα μόνο της εικόνας εμφανίζεται. Αν

η εικόνα είναι μικρότερη από το αντικείμενο πλαίσιο εικόνας, τότε προβάλλεται ολόκληρη, ενώ μένει κενός ο υπόλοιπος χώρος του πλαισίου εικόνας. Το αντικείμενο πλαίσιο εικόνας επαναπροσδιορίζει τις διαστάσεις του σύμφωνα με τις διαστάσεις της εικόνας, όταν στην ιδιότητα του AutoSize έχει τεθεί τιμή True.

Εργαλείο εικόνα (Image) : Η χρήση του εργαλείου εικόνα είναι παρόμοια με αυτή του εργαλείου πλαίσιο εικόνας. Εμφανίζει της ίδιας μορφοποίησης αρχεία εικόνας και οι διαστάσεις του προσαρμόζονται στις διαστάσεις της εικόνας όταν στην ιδιότητα Stretch τεθεί τιμή True.

Εργαλείο ετικέτα (Label): Από την άποψη πως όταν στην ιδιότητα BackStyle αποδώσουμε τιμή Transparent, το κείμενο που έχει γραφτεί στο αντικείμενο ετικέτα, μπορεί να προβληθεί πάνω από κάποιο άλλο αντικείμενο, δημιουργώντας την εντύπωση των τίτλων ή των υπότιτλων πάνω από γραφικά ή video, το εργαλείο ετικέτα μπορεί να θεωρηθεί σαν ένα Multimedia εργαλείο.

Εργαλείο πολυμέσα (MMControl) : Το εργαλείο αυτό παρέχει το βασικό interface για το χειρισμό όλων των Multimedia συσκευών (κασετόφωνο, CD-player, video κλπ), θέτοντας στις αντίστοιχες ιδιότητες τις κατάλληλες τιμές, μπορούμε να επιλέξουμε εκείνα τα πλήκτρα που θέλουμε να είναι ορατά στη φάση της εκτέλεσης και να ανταποκρίνονται σε λειτουργίες. Για να το συμπεριλάβουμε σε ένα project θα πρέπει να το εισάγουμε από το μενού Project/Components…/Microsoft Multimedia Control X.

Εργαλείο **OLE** : Σαν multimedia εργαλείο μπορεί βέβαια να καταχωρηθεί και το εργαλείο OLE, ανεξάρτητα από το γεγονός πως η χρήση του δεν στοχεύει αποκλειστικά στην πρόσβαση σε multimedia τύπου εφαρμογές. Το εργαλείο OLE χρησιμοποιείται ευρέως για την δημιουργία απλών και εύκολων multimedia εφαρμογών σε περιβάλλον Visual Basic.

Όπως όλα τα αντικείμενα της Visual Basic, έτσι και το αντικείμενο OLE χαρακτηρίζεται από μία σειρά από ιδιότητες. Από αυτές στη συνέχεια παρουσιάζεται αναλυτικά η ιδιαίτερα σημαντική ιδιότητα **Action**, η οποία είναι διαθέσιμη μόνο κατά τη φάση της εκτέλεσης της εφαρμογής.

Κατά τη διάρκεια εκτέλεσης μιας εφαρμογής μπορούμε με ένα αντικείμενο OLE να πραγματοποιήσουμε μία σειρά από διαφορετικές ενέργειες, ανάλογα με την τιμή που αποδίδουμε στη ιδιότητα **Action**. Οι σημαντικότερες τιμές που μπορεί να πάρει η ιδιότητα Action είναι οι παρακάτω :

7 : Αυτή η τιμή ενεργοποιεί το αντικείμενο OLE. Το τι ακριβώς θα συμβεί όταν το αντικείμενο θα ενεργοποιηθεί εξαρτάται από τη τιμή της ιδιότητας του **Verb**. Κάθε τύπος αντικείμενου μπορεί να υποστηρίζει το δικό του σύνολο από verbs, που ουσιαστικά δεν είναι τίποτε άλλο από το σύνολο των ενεργειών που μπορεί να πραγματοποιήσει το αντικείμενο. Για να δούμε την λίστα αυτών των ενεργειών, θα πρέπει στην ιδιότητα AutoVerbMenu να έχουμε θέσει τιμή True και στη φάση της εκτέλεσης της εφαρμογής να πατήσουμε το δεξί πλήκτρο του ποντικιού όταν βρισκόμαστε πάνω από αντικείμενο OLE. Αν λοιπόν στην ιδιότητα Verb έχουμε αποδώσει τιμή 2, αυτό σημαίνει πως όταν ενεργοποιήσουμε το αντικείμενο μέσω κώδικα (ΌνομαOLE. Action = 7), τότε αυτό που θα συμβεί είναι η δεύτερη ενέργεια που παρατίθεται στη λίστα των ενεργειών. Η εξ ορισμού τιμή της ιδιότητας Verb είναι Ο, πράγμα που σημαίνει πως αυτό το οποίο συμβαίνει όταν το αντικείμενο ενεργοποιείται, είναι η πιο συνηθισμένη ενέργεια, η οποία βέβαια εξαρτάται από το είδος της διασυνδεδεμένης εφαρμογής (π.χ. edit για κείμενο, play για ήχο και video).

9 : Αυτή η τιμή σταματάει τη σύνδεση, εφ' όσον πρόκειται για ένα ενσωματωμένο (embedded) αντικείμενο OLE, με την διασυνδεδεμένη εξωτερική εφαρμογή την οποία και κλείνει, ενώ δεν επιφέρει κανένα ουσιαστικό αποτέλεσμα στην περίπτωση του συνδεδεμένου (linked).

11. Διαχείριση αρχείων στη Visual Basic

Η Visual Basic, πέρα από την υποστήριξη σε τύπους αρχείων άλλων εφαρμογών, όπως είδαμε στο προηγούμενο κεφάλαιο, μπορεί να διαχειριστεί και αρχεία που δημιουργεί η ίδια. Υποστηρίζονται 3 τύποι τέτοιων αρχείων :

- Σειριακά Αρχεία (Sequential)
- Αρχεία Τυχαίας Προσπέλασης (Random)
- Δυαδικά Αρχεία (Binary)

11.1 Σειριακά αρχεία

Τα σειριακά αρχεία αποτελούνται από μια σειρά γραμμών κειμένου, και για αυτό ονομάζονται και αρχεία κειμένου (Text Files). Κάθε γραμμή του αρχείου ονομάζεται εγγραφή (record). Ειδική περίπτωση σειριακών αρχείων είναι τα οριοθετημένα αρχεία (delimited) όπου τα δεδομένα κάθε εγγραφής χωρίζονται μεταξύ τους με κόμματα (,). Κατά την προσπέλαση μπορούμε είτε να διαβάσουμε το αρχείο είτε να γράψουμε σ' αυτό, όχι όμως ταυτόχρονα. Για την εγγραφή δεδομένων στα σειριακά αρχεία χρησιμοποιούνται οι εντολές **Print** και **Write** και για την ανάγνωση οι εντολές **Input** και **Line Input**.

Τυποποιημένες ενέργειες στα αρχεία αυτά είναι :

- Ανοιγμα αρχείου . Πριν χρησιμοποιηθεί ένα αρχείο πρέπει να ανοιχτεί για εγγραφή, ανάγνωση ή προσθήκη δεδομένων στο τέλος του. Οι τρόποι αυτοί ανοίγματος δηλώνονται ως εξής :

Open ονομα _αρχείου **For** τύπος _προσπέλασης **As** αριθμός _αρχείου

Όπου ο τύπος_προσπέλασης μπορεί να έχει τιμές input (ανάγνωση), output (εγγραφή), append (προσθήκη) και ο αριθμός_αρχείου είναι ένας αριθμός από 1 ως το 511 που αντιστοιχίζεται στο αρχείο αυτό. Μπορούν να είναι πολλά αρχεία ανοικτά ταυτόχρονα, καθ' ένα με διαφορετικό αριθμό. Για μεγαλύτερη παραμετροποίηση της εφαρμογής, μπορούμε να παίρνουμε τον αριθμό του αρχείου από την συνάρτηση **FreeFile**, η οποία δίνει τον επόμενο ελεύθερο αριθμό αρχείου.

- Κλείσιμο αρχείου . Όταν ολοκληρωθεί η προσπέλαση του αρχείου θα πρέπει να κλείσουμε το αρχείο, ώστε και να διασφαλιστούν τα δεδομένα μας και να επιστραφούν πόροι στο λειτουργικό σύστημα του Η/Υ. Το κλείσιμο γίνεται με τη δήλωση :

Close αριθμός _αρχείου **1,** αριθμός _αρχείου **2,…..**

- Εγγραφή δεδομένων . Η εγγραφή δεδομένων στα σειριακά αρχεία γίνεται με τις εντολές Write ή Print. Η σύνταξη τους είναι η ακόλουθη :

Print #αριθμός _αρχείου **,** παράμετρος **1,** παράμετρος **2,……**

Write #αριθμός _αρχείου **,** παράμετρος **1,** παράμετρος **2,……**

Η διαφορά μεταξύ τους είναι στον τρόπο αποθήκευσης των δεδομένων στο αρχείο. Η Print γράφει απλά τα δεδομένα στο αρχείο χωρίς να τα διαχωρίζει με κόμμα ή να περικλείει σε εισαγωγικά τα strings. Η Write καταχωρεί τα δεδομένα οριοθετημένα , δηλαδή διαχωρισμένα με κόμμα (,), περικλείει τα αλφαριθμητικά δεδομένα σε εισαγωγικά και γράφει τις ημερομηνίες μεταξύ συμβόλων (#). Δημιουργεί δηλαδή πεδία (fields) σε κάθε εγγραφή. Έτσι, είναι εύκολη η διάκριση τους κατά την ανάγνωση του αρχείου.

- Ανάγνωση δεδομένων **.** Η ανάγνωση δεδομένων γίνεται με τις εντολές Line Input και Input, οι οποίες συντάσσονται ως εξής :

Line Input #αριθμός _αρχείου **,** μεταβλητή

Input #αριθμός _αρχείου **,** μεταβλητή **1,** μεταβλητή **2,…**

Η Line Input διαβάζει μια σειρά (=εγγραφή) από το αρχείο και την αντιστοιχεί στην μεταβλητή, η οποία πρέπει να είναι τύπου string. Η Input χρησιμοποιείται για οριοθετημένα αρχεία και διαβάζει κάθε πεδίο της εγγραφής στην αντίστοιχη μεταβλητή. Οι μεταβλητές αυτές μπορεί να είναι διαφόρων τύπων, ανάλογα με τα δεδομένα που έχουν γραφεί.

Κατά την ανάγνωση των αρχείων, για να μην υπάρχει περίπτωση να επιχειρήσουμε προσπέλαση πέραν του τέλους του αρχείου, οπότε θα 'κρεμάσει' η εφαρμογή μας, χρησιμοποιούμε έναν βρόχο και την συνάρτηση **EOF()** η οποία γίνεται true όταν φθάσουμε στο τέλος του αρχείου.

11.2 Αρχεία Τυχαίας Προσπέλασης

Τα αρχεία τυχαίας προσπέλασης (Random) δεν έχουν τους περιορισμούς των σειριακών, καθώς είναι δυνατή η ταυτόχρονη ανάγνωση και εγγραφή στο αρχείο, σε οποιαδήποτε θέση του. Το τίμημα γι' αυτό είναι ότι όλες οι εγγραφές πρέπει να έχουν το ίδιο μέγεθος και η γραμμογράφηση των εγγραφών τους να είναι αυστηρά τυποποιημένη και να προδηλώνεται στο πρόγραμμα με τύπο δεδομένων οριζόμενο από το χρήστη. Για την ανάγνωση και εγγραφή στα αρχεία αυτά χρησιμοποιούνται οι εντολές **Get** και **Put** αντίστοιχα.

Τυποποιημένες ενέργειες στα αρχεία αυτά είναι :

- Άνοιγμα αρχείου . Πριν χρησιμοποιηθεί ένα αρχείο τυχαίας προσπέλασης, είτε για εγγραφή είτε για ανάγνωση, πρέπει να ανοιχτεί ως εξής :

Open ονομα _αρχείου **[For Random] As** αριθμός _αρχείου **Len=**μέγεθος _εγγραφής

Το μήκος εγγραφής, αν παραλειφθεί παίρνει αυτόματα την τιμή 128 bytes. Επειδή στα αρχεία αυτά αποθηκεύονται εγγραφές που είναι τύποι δεδομένων οριζόμενοι από το χρήστη, το μέγεθος_εγγραφής μπορεί να υπολογιστεί από την συνάρτηση Len(μεταβλητή_τύπου_οριζόμενου_από_το_χρήστη)

- Κλείσιμο αρχείου . Όταν ολοκληρωθεί η προσπέλαση του αρχείου θα πρέπει να κλείσουμε το αρχείο, ώστε και να διασφαλιστούν τα δεδομένα μας και να επιστραφούν πόροι στο λειτουργικό σύστημα του Η/Υ. Το κλείσιμο γίνεται με τη δήλωση :

Close αριθμός _αρχείου **1,** αριθμός _αρχείου **2,…..**

- Εγγραφή δεδομένων . Η εγγραφή δεδομένων στα αρχεία τυχαίας προσπέλασης γίνεται με την εντολή Put. Η σύνταξη της είναι η ακόλουθη :

Put #αριθμός _αρχείου **,** αριθμός _εγγραφής **,** μεταβλητή

Οι τιμές που μπορεί να πάρει ο αριθμός εγγραφής είναι από 1 έως 2^{31} (2.147.483.647).

- Ανάγνωση δεδομένων **.** Η ανάγνωση δεδομένων γίνεται με την εντολή Get η οποία συντάσσεται ως εξής:

Get #αριθμός _αρχείου **,** αριθμός _εγγραφής **,** μεταβλητή

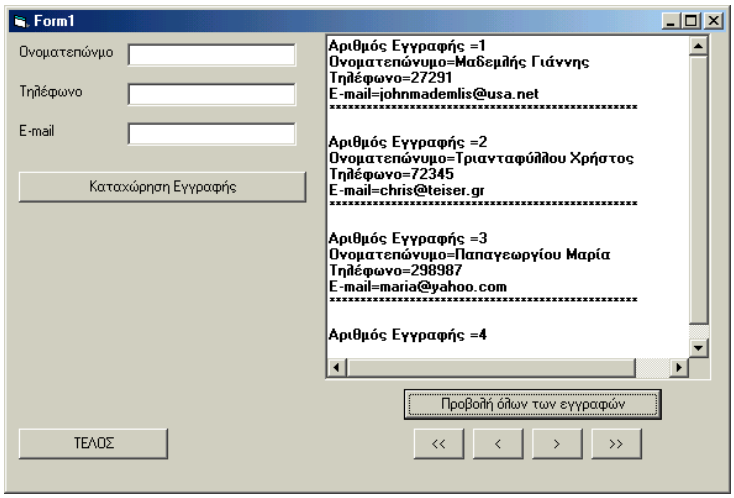
Κατά την ανάγνωση των αρχείων τυχαίας προσπέλασης, για να μην υπάρχει περίπτωση να επιχειρήσουμε προσπέλαση πέραν του τέλους του αρχείου, οπότε θα 'κρεμάσει' η εφαρμογή μας, χρησιμοποιούμε έναν βρόχο και την συνάρτηση **EOF()** η οποία γίνεται true όταν φθάσουμε στο τέλος του αρχείου.

Άλλες χρήσιμες συναρτήσεις στα αρχεία τυχαίας προσπέλασης είναι :

- **FileLen(**όνομα _αρχείου). Επιστρέφει το μέγεθος σε bytes ενός κλειστού αρχείου, π.χ FileLen("TEST.TXT")
- **LOF(**αριθμός _αρχείου **).** Επιστρέφει το μέγεθος σε bytes ενός ανοικτού αρχείου, π.χ LOF(4).
- **LOC(**αριθμός _αρχείου **).** Επιστρέφει τον αριθμό της εγγραφής που προσπελάστηκε τελευταία.

11.3 Εφαρμογή στα αρχεία τυχαίας προσπέλασης

Στο παράδειγμα που ακολουθεί φαίνεται ο τρόπος εισαγωγής και ανάγνωσης αρχείων τυχαίας προσπέλασης, δημιουργώντας μια ατζέντα.



Σχήμα :25

Δημιουργούμε έναν τύπο οριζόμενο από τον χρήστη, και δηλώνουμε μια μεταβλητή στον τύπο αυτό.

Κατόπιν, ενημερώνουμε τα πεδία της μεταβλητής αυτής και καταχωρούμε τη μεταβλητή στο αρχείο ή διαβάζουμε διαδοχικά τις εγγραφές από το αρχείο.

Επίσης, με τα ανάλογα κουμπάκια μπορούμε να μετακινηθούμε στις εγγραφές του αρχείου.

```
Option Explicit
Const filename = "c:\FRIENDS.DAT"

Private Type MyFriends
   name As String * 40
   telephone As String * 12
   email As String * 20
End Type

Private varMyfriends As MyFriends
Dim provoli As String
Dim Filenum As Integer

Private Sub Command1_Click()
'Add new record
'Go to last record
If LOF(Filenum) / Len(varMyfriends) = 0 Then 'empty file
   Get #Filenum, , varMyfriends
Else
   Get #Filenum, LOF(Filenum) / Len(varMyfriends), varMyfriends
End If
varMyfriends.name = Text1.Text
varMyfriends.telephone = Text2.Text
varMyfriends.email = Text3.Text
Put #Filenum, , varMyfriends
Call ClearTextBoxes
Call MsgBox("Η καταχώρηση έγινε", 48)
End Sub
```

```
Private Sub Command2_Click(Index As Integer)
'Show First, Last, Previous, Next record
Text4.Text = " "
provoli = ""
Select Case Index
   Case 0 'first record
      Get #Filenum, 1, varMyfriends
   Case 1 'previous record
      If Loc(Filenum) > 1 Then
         Get #Filenum, Loc(Filenum) - 1, varMyfriends
      Else
         Call MsgBox("Βρίσκεστε στην πρώτη εγγραφή", 48)
      End If
   Case 2 'next record
      If Loc(Filenum) < LOF(Filenum) / Len(varMyfriends) Then
            Get #Filenum, , varMyfriends
      Else
         Call MsgBox("Βρίσκεστε στην τελευταία εγγραφή", 48)
      End If
   Case 3 'Last Record
       Get #Filenum, LOF(Filenum) / Len(varMyfriends), varMyfriends
End Select
provoli = provoli & "Αριθμός Εγγραφής =" & Loc(Filenum) & vbCrLf
provoli = provoli & "Ονοματεπώνυμο=" & varMyfriends.name & vbCrLf
provoli = provoli & "Τηλέφωνο=" & varMyfriends.telephone & vbCrLf
provoli = provoli & "E-mail=" & varMyfriends.email & vbCrLf
provoli = provoli & String(40, "*") & vbCrLf & vbCrLf
Text4.Text = provoli
End Sub

Private Sub Command6_Click()
'Show all records
provoli = ""
Filenum = FreeFile
Open filename For Random As Filenum Len = Len(varMyfriends)
Do While Not EOF(Filenum)
   Get #Filenum, , varMyfriends
   provoli = provoli & "Αριθμός Εγγραφής =" & Loc(Filenum) & vbCrLf
   provoli = provoli & "Ονοματεπώνυμο=" & varMyfriends.name & vbCrLf
   provoli = provoli & "Τηλέφωνο=" & varMyfriends.telephone & vbCrLf
   provoli = provoli & "E-mail=" & varMyfriends.email & vbCrLf
   provoli = provoli & String(50, "*") & vbCrLf & vbCrLf
Loop
Text4.Text = provoli
```

```
End Sub

Private Sub Command7_Click()
   Close #Filenum
   End
End Sub

Private Sub Form_Load()
   Text1.Font.Bold = True
   Text4.Font.Bold = True
   Call ClearTextBoxes
   Filenum = FreeFile
   Open filename For Random As Filenum Len = Len(varMyfriends)
End Sub

Sub ClearTextBoxes()
   Text1.Text = ""
   Text2.Text = ""
   Text3.Text = ""
   Text4.Text = ""
End Sub
```

11.4 Δυαδικά αρχεία

Στα δυαδικά αρχεία (Binary) τα δεδομένα αποθηκεύονται όπως ακριβώς αναπαρίστανται στην μνήμη του Η/Υ. Οι εγγραφές δεν ακολουθούν κάποια συγκεκριμένη δομή και τα πεδία δεν αποτελούν τα θεμελιώδη στοιχεία του αρχείου. Μπορούμε να αποθηκεύσουμε σε αυτά μεταβλητές τύπου οριζόμενου από το χρήστη ή και απλές μεταβλητές. Επίσης, μπορούμε να προσπελάσουμε το περιεχόμενο τους σε οποιοδήποτε σημείο και μάλιστα byte-byte. Πρόκειται για τον πιο ευέλικτο τρόπο προσπέλασης που όμως απαιτεί ιδιαίτερη προσοχή από τον προγραμματιστή. Για την ανάγνωση και εγγραφή στα αρχεία αυτά χρησιμοποιούνται οι εντολές **Get** και **Put** αντίστοιχα.

Τυποποιημένες ενέργειες στα αρχεία αυτά είναι :

- Ανοιγμα αρχείου . Πριν χρησιμοποιηθεί ένα δυαδικό αρχείο, είτε για εγγραφή είτε για ανάγνωση, πρέπει να ανοιχτεί ως εξής :

  Open ονομα _αρχείου **For Binary As** αριθμός _αρχείου

- Κλείσιμο αρχείου . Όταν ολοκληρωθεί η προσπέλαση του αρχείου θα πρέπει να κλείσουμε το αρχείο, ώστε και να διασφαλιστούν τα δεδομένα μας και να επιστραφούν πόροι στο λειτουργικό σύστημα του Η/Υ. Το κλείσιμο γίνεται με τη δήλωση :

  Close αριθμός _αρχείου **1,** αριθμός _αρχείου **2,…..**

- Εγγραφή δεδομένων . Η εγγραφή δεδομένων στα δυαδικά αρχεία γίνεται με την εντολή Put. Η σύνταξη της είναι η ακόλουθη :

  Put #αριθμός _αρχείου **,** αριθμός _εγγραφής **,** μεταβλητή

Οι τιμές που μπορεί να πάρει ο αριθμός εγγραφής είναι από 1 έως 2^{31} (2.147.483.647).

- Ανάγνωση δεδομένων **.** Η ανάγνωση δεδομένων γίνεται με την εντολή Get η οποία συντάσσεται ως εξής:

Get #αριθμός _αρχείου **,** αριθμός _εγγραφής **,** μεταβλητή

Η μεταβλητή που χρησιμοποιείται στην ανάγνωση μπορεί να είναι είτε τύπου οριζόμενου από το χρήστη, οπότε από το αρχείο διαβάζεται μια εγγραφή είτε τύπου byte, οπότε το αρχείο διαβάζεται byte-byte και ανεξάρτητα από τη γραμμογράφησή του.

Κατά την ανάγνωση των δυαδικών αρχείων, για να μην υπάρχει περίπτωση να επιχειρήσουμε προσπέλαση πέραν του τέλους του αρχείου, οπότε θα 'κρεμάσει' η εφαρμογή μας, χρησιμοποιούμε έναν βρόχο και την συνάρτηση **EOF()** η οποία γίνεται true όταν φθάσουμε στο τέλος του αρχείου. Επίσης, ισχύουν και οι συναρτήσεις LOC() και LOF().

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

- Από την χρήση της Visual Basic 6.0, Visual Studio 2015 (Α' Μέρος)
- Ιστότοπος: https://msdn.microsoft.com/en-us/library/windows/desktop/ms788229.aspx
- Από την χρήση της Borland C++Builder 4 & 5 (Β' Μέρος)

Β’ Μέρος
Οπτικός Προγραμματισμός με C++ Builder

ΠΕΡΙΕΧΟΜΕΝΑ Β’ ΜΕΡΟΥΣ

Οπτικός Προγραμματισμός με τον C++Builder 5.0

Είναι ένα εργαλείο προγραμματισμού, το οποίο μοιάζει πολύ στη Visual Basic. και στο Delphi.

Η ανάπτυξη εφαρμογών γίνεται μέσω του ολοκληρωμένου περιβάλλοντος (IDE) που διαθέτει. Βασίζεται στην αντικειμενοστρεφή γλώσσα προγραμματισμού C++. Ο C++Builder είναι ευέλικτος και έχει πολλά πλεονεκτήματα σε όλα τα στάδια ανάπτυξης εφαρμογών, καθώς και στην εγκατάσταση και εκτέλεση των εφαρμογών από τους χρήστες, αφού η εφαρμογή μπορεί να αποτελείται από ένα αυτόνομο εκτελέσιμο αρχείο (.EXE), χωρίς να βασίζεται στην ύπαρξη άλλων αρχείων. Με τον C++Builder μπορούμε να ξαναχρησιμοποιήσουμε τα στοιχεία και τον κώδικα καθώς και να ενσωματώσουμε βιβλιοθήκες. Έτσι να κάνουμε πολύ γρήγορη και εύκολη τη δουλειά μας σαν προγραμματιστές.

Υπάρχουν πολλές έτοιμες σπουδαίες βιβλιοθήκες που μπορούμε να αγοράσουμε ή να «κατεβάσουμε» από το Internet, που θα μας λύσουν τα χέρια στη δουλειά μας. Επίσης, είναι πολύ ισχυρός στην ανάπτυξη εφαρμογών με βάσεις δεδομένων

Γενικά, ο προγραμματισμός σε windows βασίζεται κυρίως σε φόρμες. Οι φόρμες φιλοξενούν μηχανισμούς-αντικείμενα. Όποιος αναπτύσει μια εφαρμογή είναι υπεύθυνος να χρησιμοποιήσει πάνω σε κάθε φόρμα τους κατάλληλους μηχανισμούς-αντικείμενα. Οι πληροφορίες για τα στοιχεία μιας φόρμας καθώς και των μηχανισμών που περιέχει, αποθηκεύονται σε ένα αρχείο φόρμας (.dfm) και σε ένα αρχείο μονάδας (.cpp) με το ίδιο όνομα.

Ο C-Builder περιλαμβάνει δικούς του μηχανισμούς που αποκαλούνται **οπτικά συστατικά** (Visual Components) και περιέχονται στην βιβλιοθήκη VCL (Visual Component Library). Ολα αυτά τα συστατικά εμφανίζονται στη γραμμή εργαλείων ώστε να μπορούν εύκολα να χρησιμοποιηθούν. Ο χρήστης βέβαια μπορεί να φτιάξει και δικούς του μηχανισμούς και να τους ενσωματώσει σε αυτή τη βιβλιοθήκη και στη γραμμή εργαλείων.

Κάθε συστατικό-αντικείμενο του C-Builder έχει τις **ιδιότητες** του, στις οποίες βασίζεται η συμπεριφορά του, όταν χρησιμοποιείται σε ένα έργο ή μία Φόρμα. Για να δείτε ή να αλλάξετε τις ιδιότητες κάποιου συστατικού πατήστε το πλήκτρο F11 ή δώστε την εντολή «Edit/Object Inspector». Το Object Inspector θα εμφανιστεί:

Αποτελείται από τις καρτέλες Properties (ιδιότητες) και Events (συμβάντα).

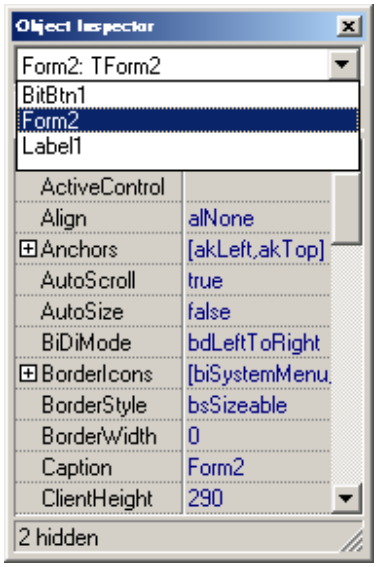
Καρτέλα **Properties**:

Ανάλογα με το συστατικό-αντικείμενο υπάρχουν πληροφορίες για το μέγεθος, το σχήμα, την θέση του και άλλα στοιχεία.

Καρτέλα **Events**:

Ανάλογα με το συστατικό-αντικείμενο υπάρχουν χειριστές συμβάντων για πολλές διαφορετικές περιπτώσεις. **Συμβάν** μπορεί να θεωρηθεί το πάτημα ενός.κουμπιού μιας φόρμας, η μετακίνηση του ποντικιού, η εισαγωγή κειμένου από το πληκτρολόγιο και άλλα.

Κάνοντας κλικ στο πτυσόμενο μενού του Object Inspector, όπως φαίνεται δίπλα, μπορούμε να επιλέξουμε τίνος αντικειμένου τις ιδιότητες θα δούμε.



ΤΑ ΑΡΧΕΙΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Κάθε εφαρμογή είναι ένα **έργο** (project). Κάθε έργο αποτελείται από διάφορα αρχεία που δημιουργεί ο χρήστης και άλλα αρχεία που δημιουργεί κατά την μεταγλώτιση ο C++Builder.

Παρακάτω φαίνονται τα βασικότερα είδη των αρχείων που εμφανίζονται κατά την ανάπτυξη μιας εφαρμογής:

Τύπος αρχείου	Ορος	Κατάληξη	Περιγραφή
Αρχείο έργου	Project	bpr	Περιέχει πληροφορίες για τις φόρμες και τις μονάδες που αποτελούν το έργο και τον κώδικα που ξεκινάει την εκτέλεση της εφαρμογής
Αρχεία μονάδας	Units	cpp	Περιέχουν κώδικα (δηλώσεις μεταβλητών και σταθερών συναρτήσεις, διαδικασίες χειρισμό συμβάντων'). Συνήθως μία μονάδα αντιστοιχεί σε μία φόρμα
Αρχεία φόρμας	Forms	dfm	Κάθε αρχείο περιέχει πληροφορίες σχετικά με μια φόρμα, όπως από ποια αντικείμενα αποτελείται, σε ποια θέση είναι το καθένα κ.ο.κ
Αρχείο επιλογών έργου	Project options	dof	Περιέχει τις ρυθμίσεις του έργου ως προς το περιβάλλον, το μεταγλωττιστή κ.ο.κ.
Αρχείο πόρων	Resources	res	Ενημερώνεται από το C++Builder και περιέχει πληροφορίες για τους πόρους του έργου.
Αντίγραφα αρχεία	Backup files	~bp , ~df ,	Αντίγραφα για αρχεία έργου, φορμών και μονάδων αντίστοιχα.

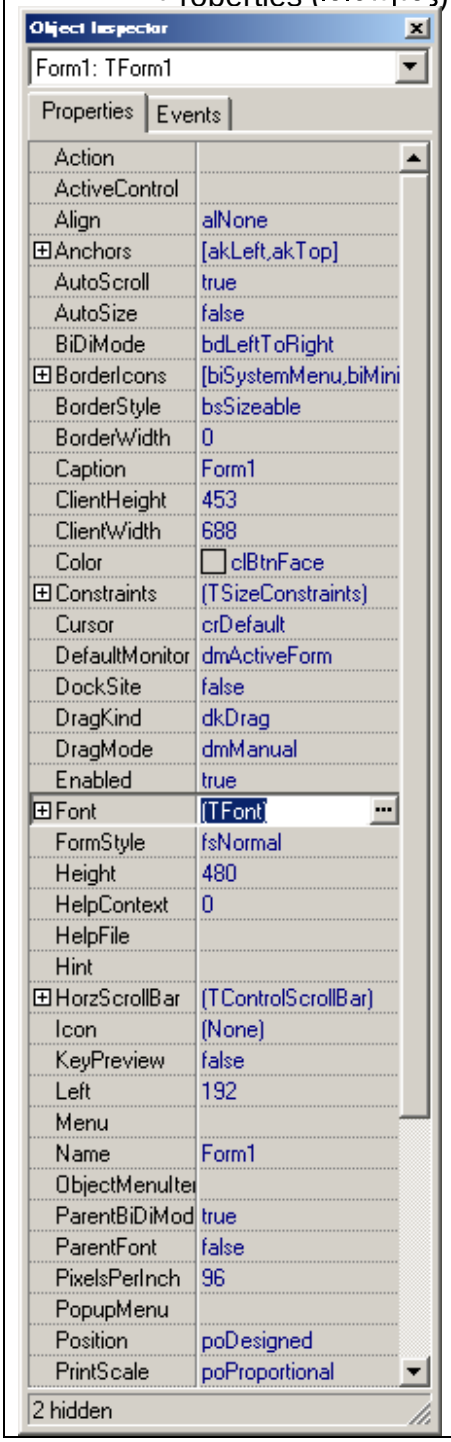
Εκτελέσιμο αρχείο	Application executable	Exe	το αυτόνομο εκτελέσιμο αρχείο του έργου.

ΒΑΣΙΚΕΣ ΙΔΙΟΤΗΤΕΣ

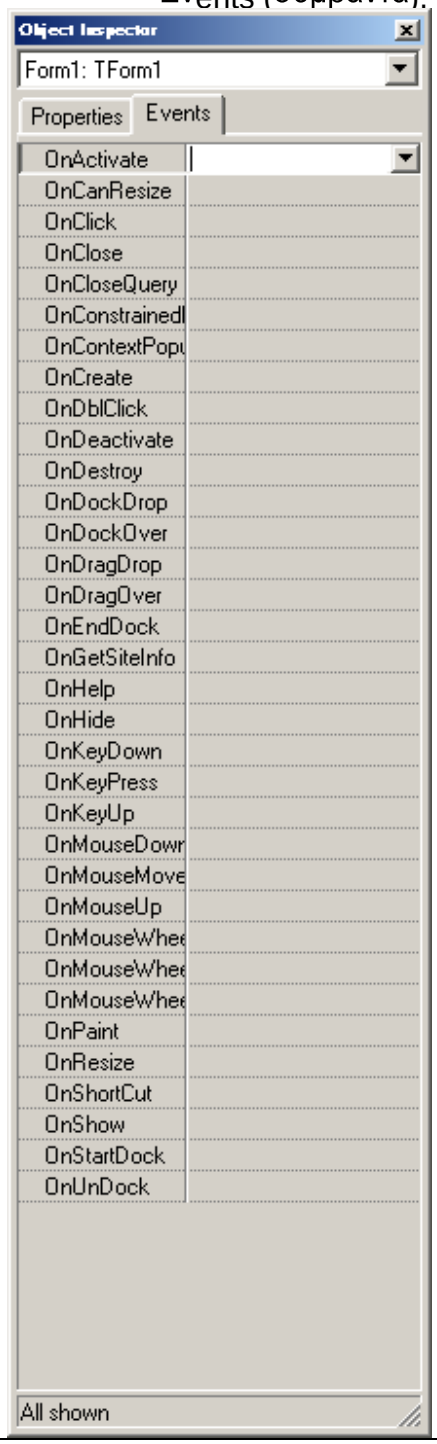
Κάθε συστατικό-αντικείμενο έχει τις δικές του ιδιότητες αλλά μερικές ιδιότητες υπάρχουν στα περισσότερα είδη συστατικών-αντικειμένων. Οι πιο σημαντικές από αυτές είναι οι ακόλουθες:

Ιδιότητες	Περιγραφή
Align	Ιδιότητα που ελέγχει τη στοίχιση (αριστερά, δεξιά, κέντρο) του συστατικού ή του κειμένου του συστατικού
Caption	Λεκτικό που θα εμφανίζεται στο συστατικό
Color	Χρώμα του συστατικού
Ctrl3D	Ιδιότητα που επιτρέπει στο συστατικό να εμφανίζεται τρισδιάστατο
Enabled	Δείχνει αν το συστατικό θα είναι ενεργοποιημένο ή απενεργοποιημένο
Font	Γραμματοσειρά που θα χρησιμοποιηθεί κατά την εμφάνιση κειμένου
Height	Υψος του συστατικού
HelpContext	Αριθμός που παραπέμπει στο αρχείο βοήθειας ώστε να εμφανιστεί το κατάλληλο κείμενο βοήθειας
Hint	Κείμενο που εμφανίζεται όταν το ποντίκι μεταφέρεται πάνω στο συστατικό
Left	Απόσταση του συστατικού από το αριστερό άκρο της φόρμας
Name	Ονομα με το οποίο χαρακτηρίζεται το συστατικό
ShowHint	Δείχνει αν το Hint του συστατικού θα εμφανίζεται όταν το ποντίκι μεταφέρεται πάνω στο συστατικό
TabOrder	Σειρά προτεραιότητας στη χρήση του Tab
Tag	Βοηθητική ιδιότητα για να καταχωρεί ο προγραμματιστής δικές του τιμές
Top	Απόσταση του συστατικού από το άνω άκρο της φόρμας
Visible	Δείχνει αν το συστατικό θα εμφανιστεί ή όχι πάνω στη φόρμα κατά την εκτέλεση
Width	Οριζόντιο μέγεθος του συστατικού

Properties (ιδιότητες)



Events (συμβάντα).



ΤΑ ΚΥΡΙΟΤΕΡΑ ΣΥΜΒΑΝΤΑ

Κάθε συστατικό-αντικείμενο έχει τα δικά του συμβάντα, αλλά μερικά συμβάντα υπάρχουν στα περισσότερα είδη συστατικών-αντικειμένων.
Τα πιο σημαντικά από αυτά είναι τα ακόλουθα :

Συμβάντα	Το συμβάν αυτό ενεργοποιείται όταν……
OnChange	Αλλάξει το συστατικό ή τα δεδομένα του
OnClick	Πατηθεί το αριστερό κουμπί του ποντικιού πάνω στο συστατικό
OnDblClick	Γίνει διπλό συνεχόμενο κλικ πάνω στο συστατικό
OnEnter	Το συστατικό παίρνει τον έλεγχο
OnExit	Το συστατικό χάνει τον έλεγχο
OnKeyDown	Οταν πατιέται ένα πλήκτρο στο πληκτρολόγιο
OnKeyPress	Οταν πατιέται ένα πλήκτρο στο πληκτρολόγιο
OnKeyUp	Οταν απελευθερώνεται ένα πατημένο πλήκτρο στο πληκτρολόγιο
OnMouseDown	Οταν πατιέται ένα κουμπί στο ποντίκι
OnMouseMove	Οταν κινείται το ποντίκι
OnMouseUp	Οταν απελευθερώνεται ένα πατημένο κουμπί στο ποντίκι

ΠΡΟΓΡΑΜΜΑΤΙΖΟΝΤΑΣ με τον C++BUILDER

Κατά την ανάπτυξη μιας εφαρμογής σε C++Builder ο χρήστης δημιουργεί φόρμες, οι οποίες αποτελούνται από μηχανισμούς-αντικείμενα και σε κάθε μηχανισμό αντιστοιχίζει τον κώδικα που χρειάζεται για να λειτουργεί η εφαρμογή όπως πρέπει.

Βέβαια υπάρχουν έτοιμες φόρμες που παρέχονται μαζί με το εργαλείο, οι οποίες μπορούν να καλύψουν πολλές από τις ανάγκες του προγραμματιστή. Επίσης υπάρχουν βιβλιοθήκες με πάρα πολλές φόρμες έτοιμες για χρήση που μπορεί κανείς να προμηθευτεί από τρίτους, όπως προαναφέραμε.

Συνήθως, ο προγραμματιστής καλείται να πρέπει να δημιουργήσει από μόνος του φόρμες που ανταποκρίνονται σε συγκεκριμένες απαιτήσεις της εφαρμογής. Σε κάθε φόρμα που δημιουργεί τοποθετεί ένα πλήθος μηχανισμών-αντικειμένων.

Μπορεί να επιλέξει μέσα από τα διαθέσιμα συστατικά-αντικείμενα ότι χρειάζεται.

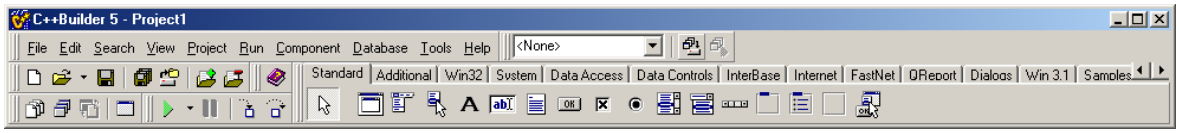
Παρακάτω περιγράφονται οι πιο χρήσιμες εργαλειομπάρες με τα σημαντικότερα αντικείμενα-μηχανισμούς τους.

Συνήθως μία φόρμα ανοίγει κάποια άλλη φόρμα. Για να γίνει αυτό, θα πρέπει στην πρώτη φόρμα να προστεθεί το όνομα του αρχείου μονάδας (unit) της δεύτερης φόρμας (δηλ. #include Form2.h) και στο κουμπί (button), που πατώντας το θα ανοίγει η δεύτερη φόρμα, να υπάρχει μία από τις εντολές Form2.ShowModal ή Form2.Show, αν θεωρήσουμε ότι το όνομα της δεύτερης φόρμας είναι Form2.

Το ίδιο γίνεται ως εξείς:

1. Το F12 μας εναλλάσει μεταξύ Form & Code Editor, έτσι στο Code Editor πατάμε και επιλέγουμε το Main.cpp ή Unit1.cpp ή όπως έχουμε ονομάσει την $1^{η}$ μονάδα.
2. Εκεί δίνουμε την εντολή «File/Include Unit Hdr»

ΕΡΓΑΛΕΙΟΜΠΑΡΑ STANDARD

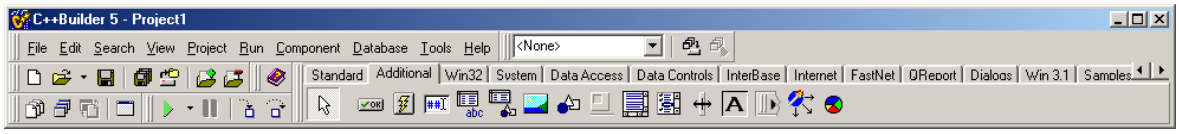


Περιέχει τα ακόλουθα συστατικά-αντικείμενα:

Συστατικό	Περιγραφή	Επεξήγηση
MainMenu	Κύριο μενού	Δημιουργία κύριου μενού
PopupMenu	Αναδυόμενο μενού	Δημιουργία αναδυόμενων μενού τα οποία ενεργοποιούνται πατώντας το δεξί κουμπί του ποντικιού
Label	Ετικέτα	Εμφάνιση κειμένου
Edit	Πεδίο επεξεργασίας	Εισαγωγή κειμένου και εμφάνιση κειμένου σε μια γραμμή
Memo	Πεδίο κειμένου	Εισαγωγή κειμένου και εμφάνιση κειμένου πολλών γραμμών
Button	Πλήκτρο	Ενεργοποίηση κάποιας λειτουργίας
Checkbox	Πλαίσιο ελέγχου	Ενεργοποίηση / Απενεργοποίηση κάποιας ένδειξης

RadioButton	Στρογγυλό πλήκτρο	Ενεργοποίηση / Απενεργοποίηση κάποιας ένδειξης και επιλογή, λειτουργιών
ListBox	Πλαίσιο λίστας	Επιλογή στοιχείων μέσα από
Combobox	Αναδιπλούμενη λίστα	Επιλογή στοιχείων μέσα από μία αναδιπλούμενη λίστα αλλά με δυνατότητα εισαγωγής κειμένου
ScrollBar	Πλαίσιο ολίσθησης	Γραμμή ολίσθησης
GroupBox	Πλαίσιο ομαδοποίησης	Χώρος ομαδοποίησης άλλων \ μηχανισμών
RadioGroup	Ομάδα στρογγυλών πλήκτρων	Επιλογή μιας λειτουργίας πλήκτρων μέσα από μια ομάδα στρογγυλών πλήκτρων
Panel	Πλαίσιο	Δημιουργία γραμμών καταστάσεων, παλετών. Επίσης χρησιμοποιείται για εμφάνιση κειμένου σε πλαίσιο

ΕΡΓΑΛΕΙΟΜΠΑΡΑ ADDITIONAL

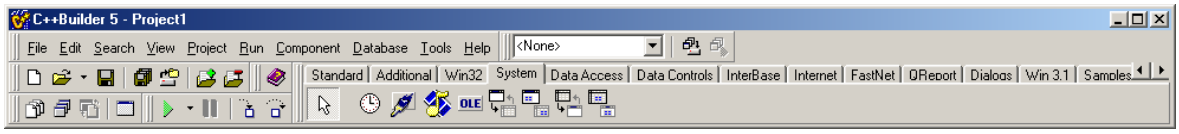


Περιέχει τα ακόλουθα συστατικά-αντικείμενα:

Συστατικό	**Περιγραφή**	**Επεξήγηση**
Bit Bin	Πλήκτρο με bitmap	Ενεργοποίηση κάποιας λειτουργίας όπως και το Button. Μπορεί να περιέχει bitmap
SpeedButton	Πλήκτρο γρήγορης επιλογής	Χρησιμοποιείται στις γραμμές εργαλείων. Μπορεί να εμφανίζεται ως πατημένο και να περιέχει bitmap
MaskEdit	Πεδίο κειμένου με μάσκα	Ελεγχόμενη εισαγωγή δεδομένων ή εμφάνιση μορφοποιημένου κειμένου σε μια γραμμή
StringGrid	Πίνακας αλφαριθμητικών	Εμφάνιση και εισαγωγή αλφαριθμητικών τιμών σε μορφή πίνακα με γραμμές και στήλες
DrawGrid	Πίνακας σχεδίων	Εμφάνιση γραφικών σε μορφή πίνακα με γραμμές

		και στήλες
Image	Εικόνα	Χώρος εμφάνισης γραφικών (εικονίδια, bitmaps και αρχεία μορφής metafile)
Shape	Σχήμα	Σχεδιασμός σχημάτων όπως τα τετράγωνα, οι κύκλοι κ. ο. κ.
Bevel	Ανάγλυφο	Σχεδιασμός ανάγλυφου ορθογωνίου
ScrollBox	Περιοχή ολίσθησης	Δημιουργία περιοχής με δυνατότητα ολίσθησης
CheckList Box	Πλαίσιο λίστας με πλαίσια ελέγχου	Συνδυασμός του ListBox και του Checkbox
Splitter	Χώρισμα	Δίνει τη δυνατότητα στο χρήστη να ορίζει το μέγεθος των περιοχών μιας εφαρμογής κατά την εκτέλεση
StaticText	Στατική ετικέτα	Παρόμοιο με το Label. Εμφανίζει κείμενο αλλά έχει περισσότερες οπτικές δυνατότητες όπως ο ορισμός περιγράμματος
Chart	Διάγραμμα	Εμφανίζει γραφήματα και διαγράμματα

ΕΡΓΑΛΕΙΟΜΠΑΡΑ SYSTEM

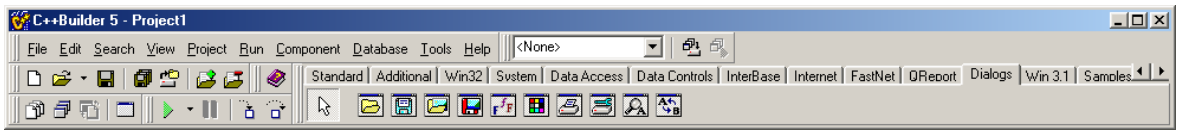


Περιέχει τα ακόλουθα συστατικά-αντικείμενα:

Συστατικό	**Περιγραφή**	**Επεξήγηση**
Timer	Ξυπνητήρι	Ενεργοποίηση κάποιας λειτουργίας σε συγκεκριμένα ή τακτά χρονικά διαστήματα
PaintBox	Πλαίσιο ζωγραφικής	Περιοχή η οποία έχει τη δυνατότητα σχεδίασης
MediaPlayer	Αντικείμενο πολυμέσων	Αναπαραγωγή αρχείων ήχου και βίντεο. Εμφανίζεται ένας πίνακα. ελέγχου της

		αναπαραγωγής
OLEContainer	Αντικείμενο OLE	Περιοχή που χρησιμοποιείται για ένα αντικείμενο OLE
DDEClientConv, DDEClientItem	DDE συνομιλία Client	Συστατικά του Client για συνομιλία DDE ενός server και ενός client
DDEServerConv, DDEServerItem	DDE συνομιλία Server	Συστατικά του Server για συνομιλία DDE ενός server και ενός client

ΕΡΓΑΛΕΙΟΜΠΑΡΑ DIALOGS

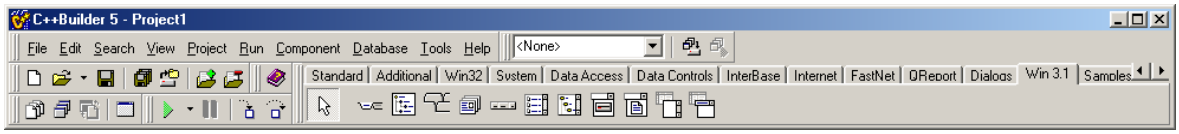


Περιέχει τα ακόλουθα συστατικά-αντικείμενα:

Συστατικό	**Περιγραφή**	**Επεξήγηση**
OpenDialog	Παράθυρο επιλογής αρχείου	Επιλογή ενός ή περισσότερων αρχείων άνοιγμα
SaveDialog	Παράθυρο αποθήκευσης αρχείου	Επιλογή ή καταχώρηση του ονόματος ενός αρχείου για αποθήκευση
OpenPictureDialog	Παράθυρο επιλογής αρχείου εικόνας	Επιλογή ενός ή περισσότερων αρχείων εικόνας για άνοιγμα
SavePictureDialog	Παράθυρο αποθήκευσης αρχείου εικόνας	Επιλογή ή καταχώρηση ονόματος ενός αρχείου εικόνας για αποθήκευση
FontDialog	Παράθυρο επιλογής	Επιλογή γραμματοσειράς

	γραμματο-σειράς	
ColorDialog	Παράθυρο επιλογής χρώματος	Επιλογή χρώματος
PrintDialog	Παράθυρο εκτύπωσης	Εμφανίζει το πλαίσιο διαλόγου εκτύπωσης
PrinterSetupDialog	Παράθυρο ρύθμισης εκτυπωτή	Διαμόρφωση του εκτυπωτή
FindDialog	Παράθυρο αναζήτησης κειμένου	Εμφανίζει το πλαίσιο διαλόγου αναζήτησης
ReplaceDialog	Παράθυρο αντικατάστασης κειμένου	Εμφανίζει το πλαίσιο διαλόγου αντικατάστασης

ΕΡΓΑΛΕΙΟΜΠΑΡΑ WIN 3.1



Περιέχει τα ακόλουθα συστατικά-αντικείμενα:

Συστατικό	**Περιγραφή**	**Επεξήγηση**
TabSet	Καρτέλες	Καρτέλες σε μορφή σημειωματάριου
OutLine	Δένδρο δεδομένων	Εμφάνιση δεδομένων ιεραρχικά με δενδροειδή μορφή
Header	Επικεφαλίδα με ενότητες	Εμφάνιση κειμένου σε ενότητες με δυνατότητες αλλαγής μεγέθους στις ενότητες

FileListBox	Λίστα αρχείων	Εμφανίζει τα αρχεία που υπάρχουν στον επιλεγμένο κατάλογο και ικανοποιούν κάποιο φίλτρο
DirectoryListBox	Λίστα καταλόγων	Εμφανίζει τους καταλόγους που περιέχονται σε κάποιο επιλεγμένο drive ή δίσκο
DriveComboBox	Επιλογή οδηγού ή δίσκου	Αναδιπλούμενη λίστα με όλα τα διαθέσιμα drives και δίσκους η οποία χρησιμοποιείται για επιλογή ενός drive ή δίσκου
FilterComboBox	Επιλογή τύπου αρχείων	Αναδιπλούμενη λίστα η οποία χρησιμοποιείται για δήλωση του φίλτρου επιλογής αρχείων

ΕΡΓΑΛΕΙΟΜΠΑΡΑ SAMPLES

Περιέχει τα ακόλουθα συστατικά-αντικείμενα:

Συστατικό	Περιγραφή	Επεξήγηση
Gauge	Δείκτης προόδου	Ενδειξη προόδου μιας διαδικασίας σε μορφή κειμένου , μπάρας ή κυκλικού γραφήματος
ColorGrid	Πίνακας χρωμάτων	Επιλογή χρώματος
Spin Button	Πλήκτρα αυξομείωσης	Πλήκτρα μείωσης και αύξησης τιμής
SpinEdit	Πεδίο με πλήκτρα αυξομείωσης	Εισαγωγή μιας αριθμητικής τιμής με τη βοήθεια πλήκτρων μείωσης και αύξησης
DirectoryOutline	Δενδροειδή λίστα καταλόγων	Εμφάνιση των καταλόγων μιας επιλεγμένης μονάδας δίσκου σε δενδροειδή μορφή
Calendar	Ημερολόγιο	Εμφάνιση μηνιαίου ημερολογίου όπου μπορεί να

		γίνει επιλογή ημερομηνίας

Ιδιότητες Φορμών

Η τάξη των φορμών (Tform class) έχει πολλές ιδιότητες, όπως όλα τα αντικείμενα των διαφόρων τάξεων. Άλλες είναι πάρα πολύ χρησιμοποιημένες, και άλλες σπάνια χρησιμοποιημένες και δυσνόητες. Παρακάτω θα αναπτύξω μερικές χρήσιμες ιδιότητες, εκτός βέβαια από τις φανερές και εύκολα κατανοήσιμες, όπως Color, Left, Top, Width, Height…

Published properties

Οι Published properties εμφανίζονται στο Object Inspector. Αυτές μπορούν να τοποθετηθούν στο design time διαμέσου του Object Inspector και επίσης στο runtime (Όταν το πρόγραμμα εκτελείται) διαμέσου κώδικα. Ας πάρουμε το Top property, ως παράδειγμα. Ο παρακάτω κώδικας τοποθετεί στο Top property την τιμή 20 «at runtime» :

```
Top = 20;
```

Όταν αυτος ο κώδικας εκτελείται, η φόρμα θα μετακινηθεί στην οθόνη εξαιτίας της αλλαγής στο value από το Top property.
Οι properties μπορούν να διαβαστούν στο runtime επίσης. Ας πούμε ότι θέλεις να παρουσιάσεις μερικούς υπολογισμούς βασισμένους στη θέση της φόρμας. Σε αυτή την περίπτωση θα πρέπει πρώτα να διαβάσεις το Top property για να αποκαταστήσεις την αξία του (την τιμή του):

```
int t = Top;  // Η μεταβλητή t τώρα περιέχει την τιμή της ιδιότητας Top
```

Όταν λέμε " Διάβασε το Top property " Εννοούμε ότι η αξία της propertys αποκαθιστάται και αποθηκεύεται σε μία μεταβλητή.

ActiveControl ⇨ Αυτή η ιδιότητα χρησιμοποιείται για να θέσει το control το οποίο θα έχει κεντράρει όταν η φόρμα είναι ενεργή. Μπορεί να θέλεις να κεντράρεις ένα ιδιαίτερο edit control όταν ένα dialog box form εμφανίζεται. Στο design time το Value column για το ActiveControl property περιέχει μία λίστα εξαρτημάτων στη φόρμα. Μπορείς να επιλέξεις ένα από τη λίστα για να το κάνεις active control όταν η φόρμα εμφανίζεται πρώτη.

AutoScroll, HorzScrollBar, and VertScrollBar ⇨ Μαζί ελέγχουν τις scrollbars για μία φόρμα Αν το AutoScroll είναι true οι scrollbars αυτόματα εμφανίζονται όταν η φόρμα είναι μικρή για να εμφανίσει όλα τα εξαρτήματα. Το HorzScrollBar και το VertScrollBar properties έχουν διάφορες ιδιότητες οι οποίες ελέγχουν τις scrollbar

operations.

BorderStyle ⇨ υποδεικνύει τι τύπο border θα έχει η φόρμα. Το default value είναι bsSizeable, το οποίο δημιουργεί ένα παράθυρο που μπορεί να αλλάξει το μέγεθος του.

ClientWidth και ClientHeigth ⇨ Αυτά οριοθετούν την client area width και heigth αντι να γεμίζουν το βάθος και το ύψος όταν χρησιμοποιούνται. Χρησιμοποίησε αυτές τις ιδιότητες όταν θέλεις να είναι η client area specific size και το υπόλοιπο του παράθυρο ριθμισμενη ως απαραίτητη. Θέτοντας τα ClientWidth και ClientHeigth properties γίνονται αλλαγές στο Width και στο Heigth properties.

Constraits ⇨ Αυτή η χρησιμοποιείται για να θέσει το μεγαλύτερο και το μικρότερο Width και Heigth
της φόρμας. Χε "forms : properties . Απλά θέσε τα MaxHeigth, MaxWidth, MinHeigth, MinWidth values όπως επιθυμούν

DefaultMonitor ⇨ Αυτή η ιδιότητα καθορίζει ποιο μονιτορ της φόρμας θα εμφανιστεί σε ένα multi-monitor environment (όπως τα Windows 98).

Font ⇨ Αυτή η ιδιότητα καθορίζει το χρώμα των γραμμάτων που χρησιμοποιεί η φόρμα. Το σημαντικό είναι ότι το χρώμα των γραμμάτων της φόρμας κληρονομείται από τα εξαρτήματα της φόρμας. Αυτό σημαίνει ότι μπορείς να αλλάξεις το χρώμα από όλα τα εξαρτήματα απλά αλλάζοντας τα γράμματα της φόρμας. Αν ένα ατομικό χρώμα Font αλλάξει χειροκίνητα, το χρώμα αυτού του Font δεν θα αλλάξει όταν αλλάξει το φόντο της κύριας φόρμας.

FormStyle ⇨ Αυτή η ιδιότητα είναι συχνά στο fsNormal. Αν θέλεις μία φόρμα που να είναι πάντα on top, χρησιμοποίησε το fsStayOnTop style. Οι φόρμες MDI πρέπει να χρησιμοποιούν το fsMDIForm style και οι MDI θυγατρικές φόρμες το fsMDIChild style. Οι MDI φόρμες και τα MDI «θυγατρικά παράθυρα» αναπτύσονται παρακάτω στην παράγραφο «MDI (Multple Document Interface model)»

HelpContext and Helpfile ⇨ Το Helpfile χρησιμοποιήται για να θέσει το help context για μία φόρμα. Αν το context help είναι ικανό για μία φόρμα, το Windows Help system θα ενεργοποιηθεί όταν το F1 είναι πατημένο. Το context ID χρησιμοποιήται για να λέει στο Help system ποιο …… να εμφανίσει στο help file. Το HelpFile property είναι το όνομα του help file που χρησιμοποιήταιόταν είναι πατημένο το F1.

Icon ⇨ Αυτή η ιδιότητα θέτει το εικονίδιο στο title bar για τη φόρμα όταν η φόρμα εμφανίζεται στο runtime, και επίσης όταν η φόρμα είναι ελαχιστοποιημένη. Σε μερικές περιπτώσεις αυτή η ιδιότητα δεν έχει αποτέλεσμα. Για παράδειγμα όταν το FormStyle είναι στο fsDialog, η ιδιότητα icon αγνοείται.

KeyPreview ⇨ Όταν το KeyPreview είναι στο True, τα γεγονότα OnKeyPress και OnKeyDown της φόρμας θα παραχθούν όταν ένα κλειδί είναι πατημένο σε

οποιοδήποτε εξάρτημα της φόρμας. Οι φόρμες δεν λαμβάνουν γεγονότα πληκτρολογίου όταν ένα εξάρτημα στη φόρμα έχει κεντράρισμα.

Menu ⇨ Χρησιμοποίησε αυτή την ιδιότητα για να καθορίσεις το κυρίως menu για τη φόρμα. Πρώτα θα πρέπει να έχεις …… ένα MainMenu εξάρτημα στη φόρμα.

PopupMenu ⇨ Αυτή η ιδιότητα χρησιμοποιείται για να οριοθετήση ένα PopupMenu το οποίο θα εμφανίζεται όταν ο χρήστης κάνει δεξί κλικ στη φόρμα. Εσείς θα πρέπει πρώτα να έχετε τοποθετείσει ένα PopupMenu εξάρτημα στη φόρμα.

Position ⇨ Αυτή η ιδιότητα καθορίζει το μέγεθος και τη θέση της φόρμας όταν η φόρμα εμφανίζεται. Οι τρείς βασικές επιλογές είναι poDesigned, poDefault & poScreenCenter

- PoDesigned Κάνει τη φόρμα να εμφανίζεται, στην ακριβή θέση που αυτό ήταν, όταν σχεδιάστηκε
- PoDefault Επιτρέπει στο παράθυρο να θέση το μέγεθος και τη θέση, σε σχέση με το συνηθησμένο (δηλ. αυτό που το παράθυρο χρησιμοποιεί για να αποφασίση που αυτό θα εμφανίσει ένα καινούργιο παράθυρο στη οθόνη. Αν το νέο παράθυρο δεν έχει καθορισμένες πληροφορίες θα εμφανίζεται ακριβώς κάτω και στα δεξιά από το τελευταίο παράθυρο που εμφνίστηκε στην οθόνη.)
- PoScreenCenter Κάνει τη φόρμα να εμφανίζεται στο κέντρο της οθόνης, κάθε φορά που αυτό εμφανίζεται.

Visible ⇨ Αυτή η ιδιότητα ελέγχει αν η φόρμα είναι αρχικά ορατή. Αυτή η ιδιότητα δεν είναι ιδιαίτερα χρήσιμη στην ώρα του Design. Αλλά στο runtime αυτό μπορεί να καθορίζει αν η φόρμα (η γενικά ένα αντικείμενο) είναι ορατό. Έτσι μπορώ να προσομοιώσω κίνηση στις εφαρμογές μου.

WindowState ⇨ Αυτή η ιδιότητα μπορεί να καθορίσει την τρέχουσα κατάσταση της φόρμας (Μεγιστοποίηση, Ελαχιστοποίηση ή Κανονικό). Αυτό επίσης μπορεί να χρησιμοποιειθεί για να υποδεικνύει πώς η φόρμα θα πρέπει να εμφανίζεται αρχικά. Οι επιλογές είναι wsMinimized, wsMaximized & wsNormal.

Form Methods

Οι φόρμες είναι επίσης εξαρτήματα, γι΄αυτό αυτές έχουν πολλές μεθόδους κοινές με τα άλλα εξαρτήματα.Show(), ShowModal() & Invalidate()

…ingToFront Αυτή η μέθοδος κάνει την φόρμα να είναι τοπ από όλες τις άλλες φόρμες, στην εφαρμογή

Close() Η μέθοδος κλέινει μία φόρμα αφού πρώτα καλέσει το CloseQuery() να συγουρέυσει ότι είναι εντάξει για να κλείσει την φόρμα.

CloseQuery() Η συνάρτηση αυτή κλείνει το On… με το χειριστή (χέρι), αν η Bool μεταβλητή περάσει στο OnCloseQuery χειροκίνητα τοποθετείται στο False, η φόρμα δεν είναι κλειστή. Αν αυτό είναι τοποθετειμένο στο True, η φόρμα κλείνει κανονικά.

Μπορούμε να χρησιμοποιήσουμε το OnCloseQuery χειροκίνητα για να προτρέψεις

το χρήστη να σώσει ένα αρχείο το οποίο σεδς και να ελέγξεις αν η φόρμα μπορεί να κλείσει.

Print() Αυτή η μέθοδος τυπώνει τα περιεχόμενα της φόρμας, μόνο η περιοχή client της φόρμας τυπώνεται, όχι το Caption, Title ή Borders.

ScrollInView() Αυτή η μέθοδος εμφανίζει περιθώρια στη φόρμα, ώστε να μπορούν να εμφανιστούν τα εικονίδια-εξαρτήματα πάνω σ'αυτήν.

GetFocus() Αυτή η μέθοδος ενεργοποιεί τη φόρμα και την φέρνει στο προσκύνιο. Αν η φόρμα έχει εξαρτήματα, το εξάρτημα που είναι καθορισμένο στην ιδιότητα Activecontrol θα λάβει είσοδο Focus. (δες την ιδιότητα Activecontrol)

Show() & ShowModal() Και οι δύο αυτές μέθοδοι εμφανίζουν την φόρμα. Η Show() μέθοδος εμφανίζει την φόρμα χωρίς Mode (as modeless), έτσι οι άλλες φόρμες μπορούν να ενεργοποιηθούν, ενώ η φόρμα είναι ορατή. Η μέθοδος ShowModal() εκτελεί την φόρμα με Mode (modally). Δηλ. μία Modally φόρμα πρέπει να διωχθεί πριν ο χρήστης μπορέσει να συνεχίση να χρησιμοποιεί την εφαρμογή.

Form Events

On Activate ⇨ Αυτό το γεγονός λαμβάνει χώρα όταν η φόρμα είναι αρχικά ενεργεί. Η φόρμα μπορεί να είναι ενεργεί, σαν αποτέλεσμα της αρχικής της δημιουργίας ή όταν ο χρήστης γυρίσει από τη μία φόρμα στην άλλη. Η εφαρμογή επίσης έχει ένα το οποίο είναι γεννημένο όταν ο χρήστης μεταπηδά από άλλη εφαρμογή στη δική του εφαρμογή.

OnClose & OnCloseQuery ⇨ Όταν μία εφαρμογή είναι κλειστή το συμβάν στέλνετε. Το OnClose καλεί το OnCloseQuery για να ελέγξει αν αυτό είναι εντάξει, ώστε νβα κλείσει την φόρμα. Αν του OnCloseQuery συμβάντος η παράμετρος CanClose είναι False η φόρμα δεν κλείνει.

OnCreate ⇨ Αυτό το γεγονός γίνεται όταν η φόρμα είναι αρχικά δημιουργημένη. Μόνο ένα OnCreate συμβάν θα γίνει για κάθε στιγμή μίας ιδιαίτερης φόρμας. Χρησιμοποιήστε το OnCreate χειροκίνητα για να παρουσιάσετε οποιοδήποτε εργασία την οποία η φόρμα χρειάζεται να χρησιμοποιήσει.

OnDestroy ⇨ Αυτό το γεγονός είναι το αντίθετο από το OnCreate. Χρησιμοποιήστε αυτό το συμβάν για να καθαρίσετε οποιαδήποτε περιοχή μνήμης διαχειρίζεται η φόρμα ή για να καθαρίσει άλλη αχρείαστη εργασία

OnDragDrop ⇨ Αυτό το γεγονός συμβαίνει όταν ένα αντικείμενο κάνει Drop στη φόρμα. Απάντηση σ'αυτό το συμβάν είναι αν η φόρμα σας υποστηρίζει το Drag-and-Drop

OnMouseDown, OnMouseMove, OnMouseUp ⇨ Αυτά τα γεγονότα απαντούν στις αντίστοιχες κινήσεις του ποντικιού.

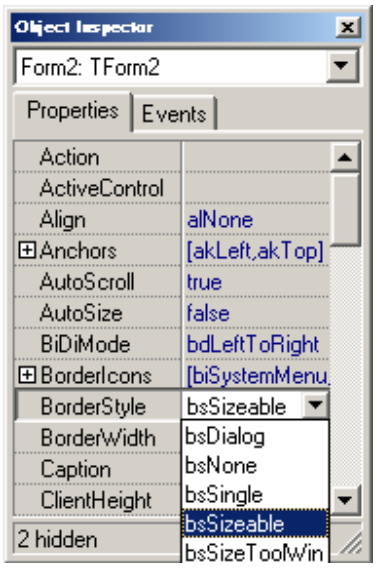
OnPaint ⇨ Αυτό το γεγονός γίνεται όποτε η φόρμα χρειάζεται ξαναζωγράφισμα. Το οποίο μπορεί να συμβεί για μία ποικιλία από αιτίες.

OnResize ⇨ Αυτό το συμβάν στέλνεται κάθε φορά που ξαναορίζεται το μέγεθος της φόρμας. Χρειάζεται να ανταποκριθείτε σ'αυτό το συμβάν για να ρυθμίσετε τα εξαρτήματα στη φόρμα ή να ξανασχεδιάσετε τη φόρμα.

OnShow ⇨ Αυτό το συμβάν λαμβάνει χώρα ακριβώς πριν γίνει ορατή η φόρμα. Μπορείται να χρησιμοποιήσετε αυτό το συμβάν για να παρουσιάσετε οποιαδήποτε διεργασία χρειάζεται η εφαρμογή σας να κάνει ακριβώς πριν γίνει ορατή η φόρμα.

Διαλογικά κουτιά (Dialog Boxes)

Τα διαλογικά κουτιά στον C++Builder δεν είναι τίποτε άλλο από απλές φόρμες. Δηλ. τα διαλογικά κουτιά δημιουργούνται όπως όλες οι άλλες φόρμες. Η βασική ιδιότητα που πρέπει να αλλάξει σ' αυτές τις φόρμες είναι το BorderStyle.



BsDialog
Οχι αλλαγή μεγέθους
Και μόνο το ✕ κουμπί

BsNone
Τίποτα από κουμπιά και αλλάγες μεγέθους

BsSingle
Οχι αλλαγή μεγέθους
Όλα τα χειριστήρια

BsSizeable
Αλλαγή μεγέθους
Με scroll bar
Όλα τα χειριστήρια

BsSizeToolWin
Αλλαγή μεγέθους
Με scroll bar
Και μόνο το ✕ κουμπί

BsToolWindow

1η Εφαρμογή

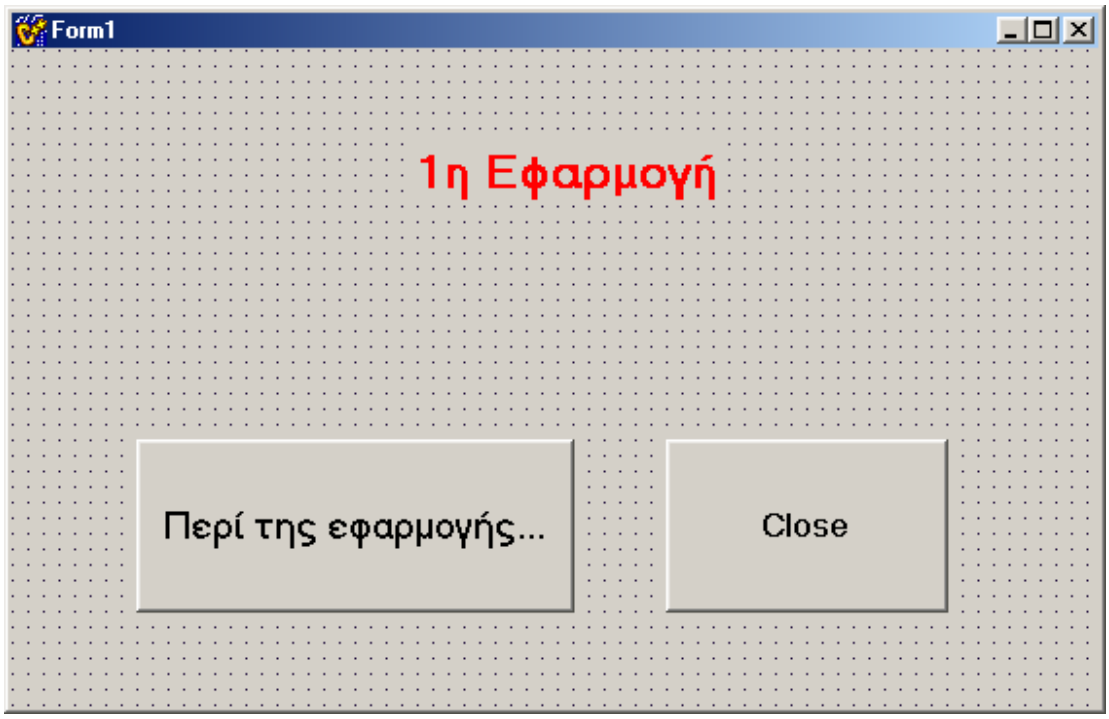
Θα δημιουργήσουμε μία φόρμα, στην οποία θα σχεδιάσουμε δύο κουμπιά. Το ένα απλά θα κλείνει την εφαρμογή και το δεύτερο θα εμφανίζει μία νέα φόρμα. Η νέα δεύτερη φόρμα θα περιέχει ετικέτες, μία εικόνα και ένα πλήκτρο «OK».

Λύση:

Δημιουργία 1ης φόρμας

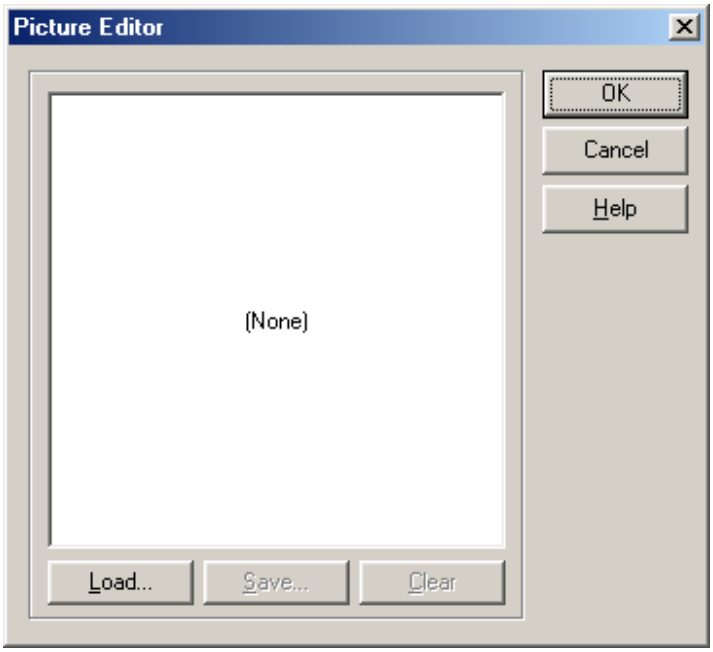
- Στην 1η φόρμα που δημιουργείται αυτόματα κατά την εκκίνηση του C++Builder, από την εργαλειομπάρα Standard, επιλέγουμε το συστατικό-αντικείμενο Button (OK), και σχεδιάζουμε δύο κουμπιά, σε θέσεις που επιλέγουμε.
- Αλλάζουμε την ιδιότητα Name του Button1 πχ. σε «Aboutbutton» και την Caption σε «Περί της εφαρμογής…». Ομοίως την ιδιότητα Name του Button2 πχ. σε «Closebutton» και την Caption σε «Close».
- Κάνουμε διπλό κλικ στο Closebutton συστατικό και στον CodeEditor που εμφανίζεται προσθέτουμε την γραμμή
- «Closebutton->close();»

- Και με διπλό κλικ στο Aboutbutton εικονίδιο γράφουμε
- Aboutbutton->ShowModal();



Δημιουργία 2ης φόρμας

- Κάνουμε κλικ στο εικονίδιο της εργαλειομπάρας (Toolbar) New Form Button
- Καθορίζουμε το μέγεθος που θέλουμε να έχει η 2η μας φόρμα
- Αλλάζουμε την ιδιότητα Name της 2ης φόρμας (Form2) πχ. σε «AboutForm» και την Caption σε «Περί της εφαρμογής XXX» ή «About program Chris».
- Αλλάζουμε την ιδιότητα BorderStyle της 2ης φόρμας (AboutForm) σε bsDialog
- Προσθέτουμε τις ετικέτες που χρειαζόμαστε να φαίνονται σ' αυτή τη φόρμα (Toolbar Standard εργαλείο Label (A))
- Εισάγουμε ένα εικονίδιο
- Από την εργαλειομπάρα (Toolbar) Additional επιλέγουμε το εργαλείο-συστατικό-αντικείμενο Image, σχεδιάζουμε ένα πλαίσιο όπου θα εισάγουμε την εικόνα
- Αλλάζουμε την ιδιότητα AutoSize (του Image) σε true
- Κάνουμε κλίκ δεξιά από την ιδιότητα picture, στο None, και πατούμε στις τρείς τελείες που εμφανίζονται. Στον Picture Editor που εμφανίζεται όπως παρακάτω, επιλέγουμε Load… και διαλέγουμε μία εικόνα.

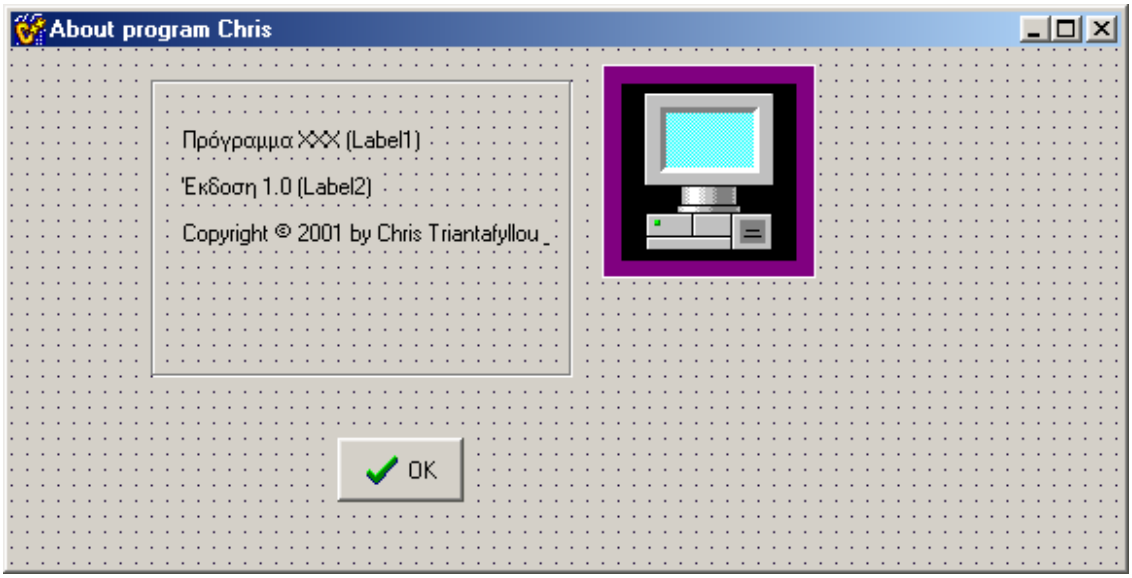


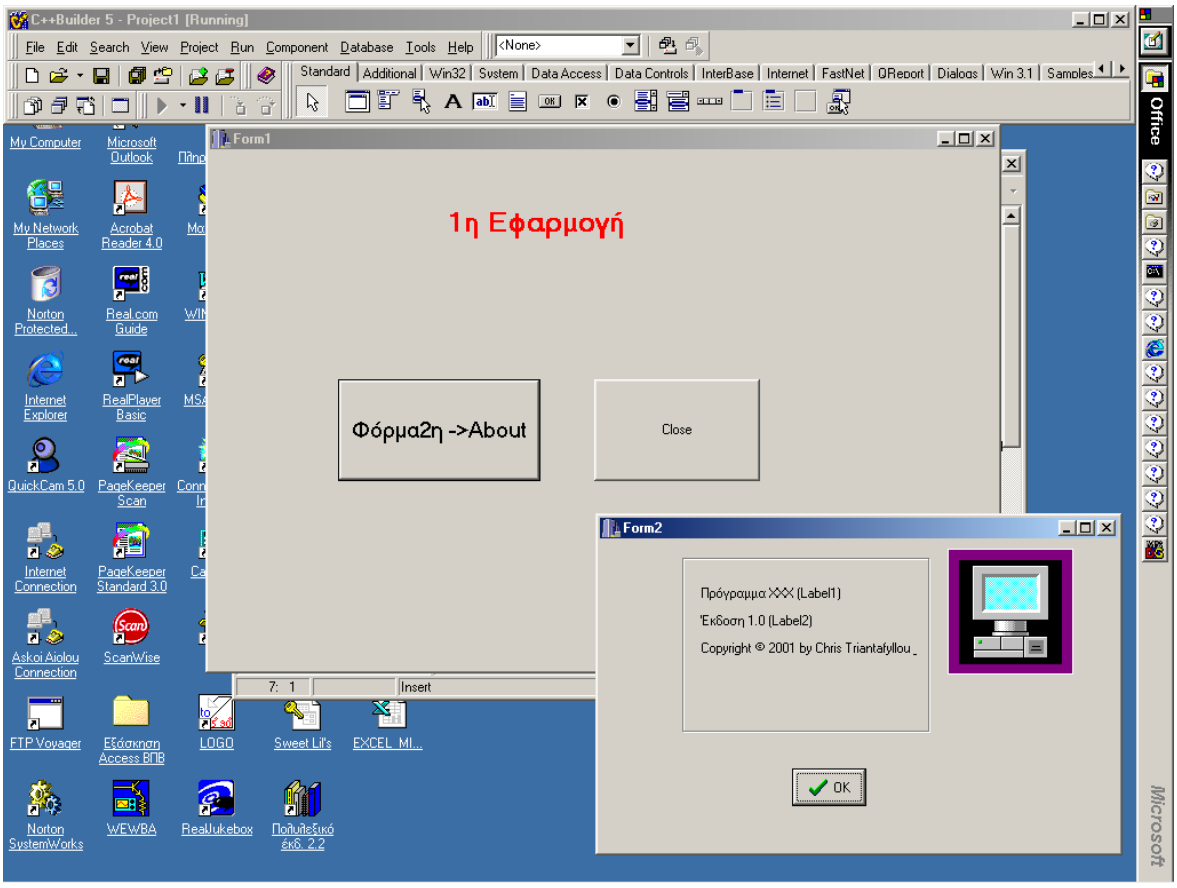
- Τώρα θα εισάγουμε και το πλήκτρο ✓ OK στη 2η φόρμα:
- Πάλι από την εργαλειομπάρα (Toolbar) Additional επιλέγουμε το εργαλείο BitBtn () και σχεδιάζουμε το κουμπί
  - Αλλάζουμε την ιδιότητα του Kind σε bkOK (το εργαλείο BitBtn περιλαμβάνει έτοιμο κ΄ώδικα για κλείσιμο της φόρμας όταν το κουμπί OK πατηθεί.
- Μπορούμε αν θέλουμε να εισάγουμε ένα τρεισδυάστατο πλαίσιο (3D Frame) το οποίο θα περικλείει τις ετικέτες μας.
  - Από την Toolbar Additional επιλέγουμε το εργαλείο Bevel () και σχεδιάζουμε ένα ορθογώνιο γύρο από τις ετικέτες μας
  - Αλλάζουμε την ιδιότητα του Shape σε bsFrame

- Τελειώνοντας πρέπει να σώσουμε την νέα μας φόρμα πχ. με το όνομα Aboutform

και το σπουδαιότερο σημείο !!!

- Με F12 γυρίζουμε στον Code Editor και στην 1η μονάδα (είχαμε προαναφέρει ονόματα όπως: το Main.cpp ή Unit1.cpp cpp ή όπως έχουμε ονομάσει την 1η μονάδα. Και δίνουμε την εντολή «File/Include Unit Hdr»





Είναι μία χαρακτηριστική SDI εφαρμογή δηλ. η $2^{η}$ φόρμα μπορεί να κινείται οπουδεί μέσα στην οθόνη.

MDI (Multple Document Interface model)

Αρκετά γρήγορα έχουμε «χτίσει» μόνο single document interface (SDI) applications.Ένα (SDI) applications έχει ένα απλό κυρίως παράθυρο και τυπικά εμφανίζει dialog boxes όταν το χρειάζεται, αλλά δεν εμφανίζει «θυγατρικό παράθυρο». (Ένα κυρίως παράθυρο το οποίο εμφανίζει secondary windows λέγεται ότι είναι ένα «πατρικό παράθυρο». Αν το «πατρικό παράθυρο» είναι κατεστραμένο το «θυγατρικό παράθυρο» είναι επίσης κατεστραμένο.) Μερικά προγράμματα ακολουθούν multiple document interface (MDI). Το mdi application περιλαμβάνει από ένα κυρίως παράθυρο (το MDI parent) και «θυγατρικό παράθυρο» (το MDI children) .

Παραδείγματα από προγράμματα τα οποία χρησιμοποιούν το Mdi model είναι ο Windows System Configuration Editor (SYSEDIT) και ο Program Manager των Wndows 3.1

Ένα από τα πιο φανερά χαρακτηριστικά του MDI model είναι ότι το MDI «θυγατρικό παράθυρο» είναι εγκλεισμένο (περιορισμένο) στο πατρικό.

Μπορούμε να σύρουμε το «θυγατρικό παράθυρο» μέσα στο «πατρικό παράθυρο» , αλλά δεν μπορούμε να το σύρουμε έξω από το «πατρικό παράθυρο».

MDI applications σχεδόν πάντα έχουν ένα Window Item Μενού που περιλαμβάνει Cascade & Tile επιλογή, κι έτσι σου επιτρέπεται να εμφανίσης τα MDI child Windows σε Cascaded & Tiled τακτοποίηση. Όταν ένα child Window ελαχιστοποιείται, το εικονίδιο του τοποθετείται στο Windows Desktop.

Δημιουργία MDI applications (Δημιουργία Parent & Child Window)

Για να δημιουργήσεις ένα MDI applications με τον C++Builder, πρέπει να τοποθετήσεις στην ιδιότητα FormStyle της φόρμας σου την τιμή FsMDIForm. Και καθένα από τα MDI παιδιά (child Window) πρέπει να έχουν στην ίδια ιδιότητα την τιμή FsMDIChild.

MDI μέθοδοι

Διάφορες μέθοδοι φόρμας έχουν σχέση καθορισμένη. με MDI λειτουργίες. Οι μέθοδοι MDI εφαρμόζονται μόνο στις MDI πατρικές φόρμες.

Η μέθοδος **ArrangeIcons()** τοποθετεί τα εικονίδια οποιουδήποτε ελαχιστοποιημένου MDI «θυγατρικό πράθυρο» σε ένα MDI «πατρικό παράθυρο».

Η μέθοδος **Cascade()** …όλα τα μη ελαχιστοποιημένα MDI «θυγατρικά παράθυρα».
Η μέθοδος **Tile()** … όλα τα ανοιχτά MDI «θυγατρικά παράθυρα».

Η μέθοδος **Next()** ενεργοποιεί το επόμενο MDI «θυγατρικό παράθυρο» στη λίστα των «θυγατρικών παραθύρων».

Η μέθοδος **Previous()** «θυγατρικά παράθυρα» ενεργοποιεί το προηγούμενο MDI «θυγατρικό παράθυρο» στη λίστα των «θυγατρικών παραθύρων».

2η Εφαρμογή

Θα τροποποιήσουμε την προηγούμενη φόρμα, στην οποία σχεδιάσαμε τα δύο κουμπιά. (Το ένα απλά κλείνει την εφαρμογή και το δεύτερο εμφανίζει μία νέα φόρμα με ετικέτες, μία εικόνα και ένα πλήκτρο «OK».) Έτσι ώστε η $2^{η}$ φόρμα να είναι «θυγατρικό παράθυρο» (chlid window) της $1^{ης}$ φόρμας, που θα είναι το «πατρικό παράθυρο» (parent window).

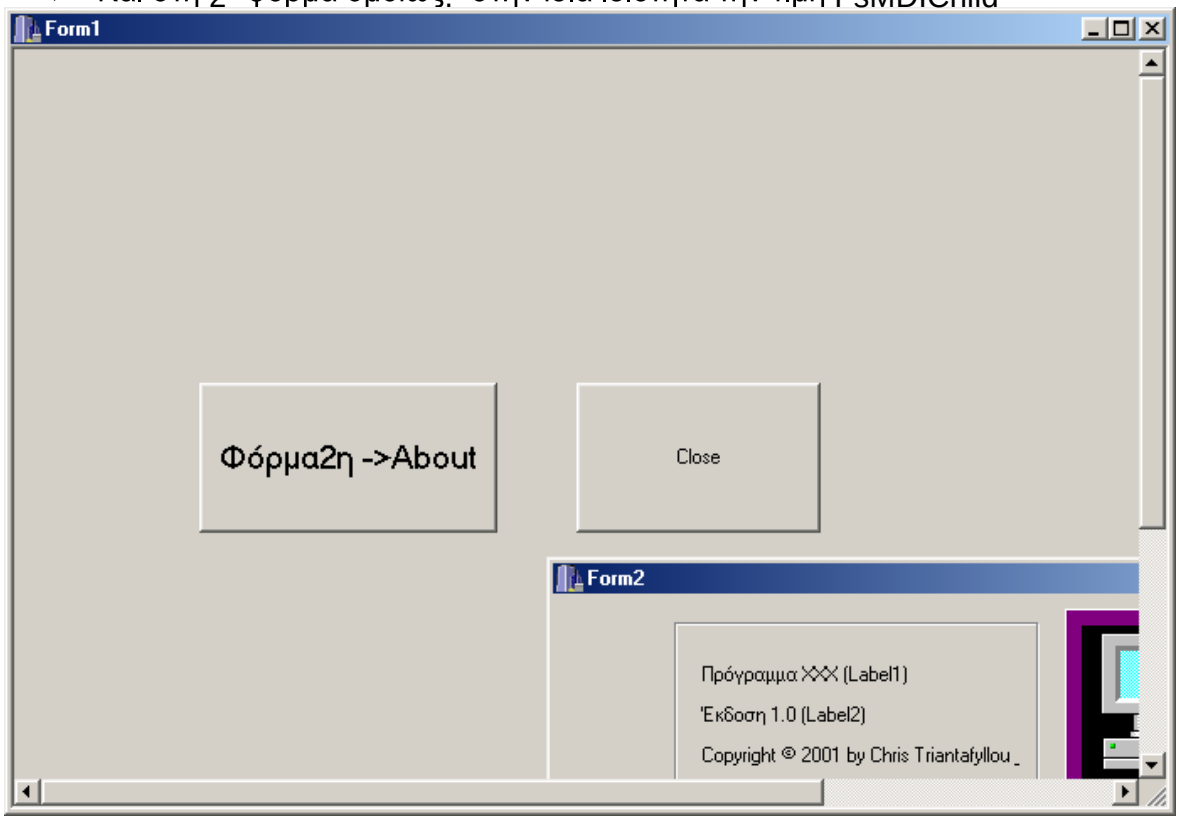
Λύση:

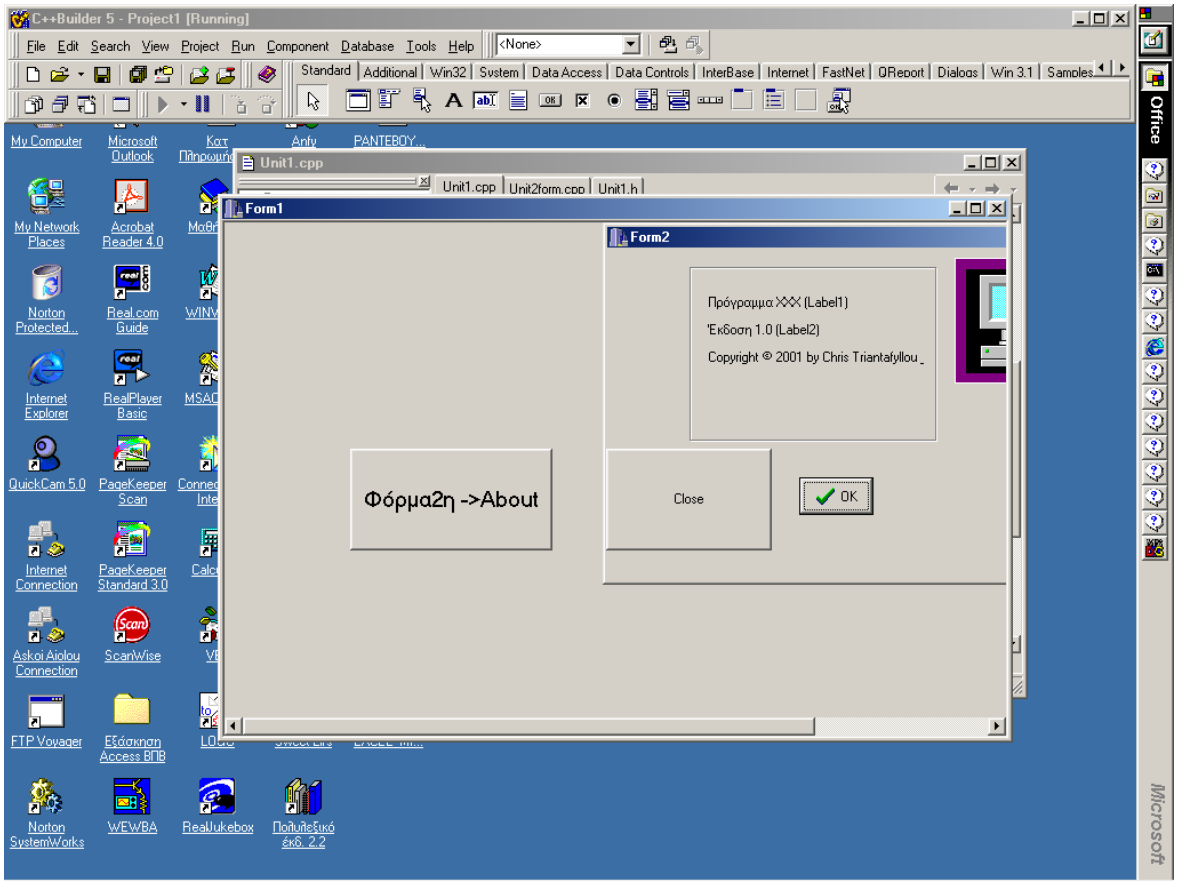
- Δημιουργία $1^{ης}$ φόρμας και $2^{ης}$ φόρμας
  Βλέπε ανωτέρω $1^{η}$ εφαρμογή

Έτσι οι αλλάγες που χρειαζόμαστε , είναι μόνο οι παρακάτω:

MDI applications

- Πρέπει να τοποθετήσουμε στην ιδιότητα FormStyle της $1^{ης}$ φόρμας την τιμή FsMDIForm.
- Και στη $2^{η}$ φόρμα ομοίως, στην ίδια ιδιότητα την τιμή FsMDIChild





Στα δύο παραπάνω σχήματα προσπάθησα να δείξω τη διαφορά της MDI εφαρμογής από την SDI εφαρμογή.
Όπως βλέπετε το χαρακτηριστικό του MDI model είναι ότι το MDI «θυγατρικό παράθυρο» είναι εγκλεισμένο (περιορισμένο) στο πατρικό.
Μπορούμε να σύρουμε το «θυγατρικό παράθυρο» μέσα στο «πατρικό παράθυρο» , αλλά δεν μπορούμε να το σύρουμε έξω από το «πατρικό παράθυρο».

3η Εφαρμογή
Δημιουργία μιας MDI εφαρμογής

Εκκινούμε τον C++Builder και δίνουμε την εντολή File/New Application

1. Αλλάζουμε την ιδιότητα Name σε MainForm
2. Αλλάζουμε την ιδιότητα Caption σε ΕΦΑΡΜΟΓΗ: PICTURE VIEWER
3. Αλλάζουμε την ιδιότητα Height πχ. σε 500 & την ιδιότητα Width σε 600 και αν θέλουμε το χρώμα της φόρμας
4. Αλλάζουμε την ιδιότητα FormStyle σε fsMDIForm

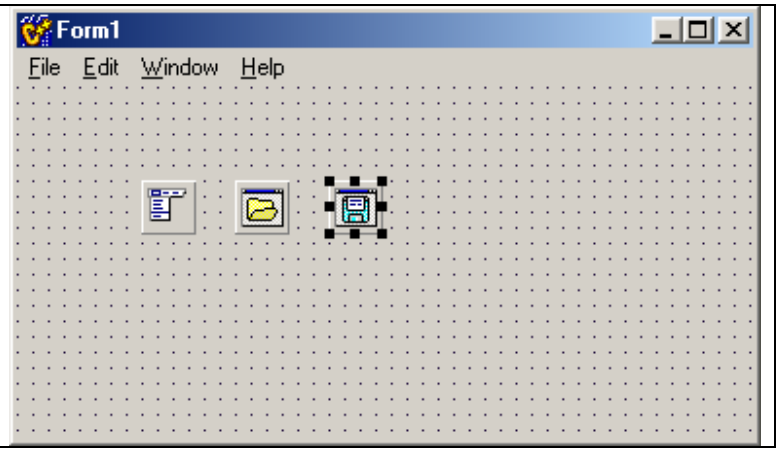
Έτσι, έχουμε φτιάξει το κυρίως μέρος άλλος φόρμας. Μετά θα προσθέσουμε ένα μενού στη φόρμα. Θα ακολουθήσουμε τον εύκολο δρόμο για τη δημιουργία του (ο

άλλος δρόμος είναι το Menu Designer).

5. Από την Standard εργαλειομπάρα επιλέγουμε το εικονίδιο-κουμπί MainMenu
6. Κάνουμε Drop στην φόρμα οπουδήποτε, γιατί δεν έχει σημασία η θέση που θα τοποθετήσουμε το εικονίδιο, αφού δεν θα φαίνεται κατά το runtime.
5. Στην ιδιότητα Name γράφουμε MainMenu
6. Κάνουμε διπλό κλικ στο MainMenu εικονίδιο, κι έτσι εμφανίζεται το Menu Designer
7. Με δεξί κλικ τώρα πάνω στο εικονίδιο, επιλέγουμε «Insert From Template»
8. Επιλέγουμε MDI Frame και OK (Εμφανίζεται το συγκεκριμένο μενού στη φόρμα μας, και στη συνέχεια κλείνουμε το παράθυρο)

Φυσικά τα υπομενού που εμφανίζονται δεν είναι ακόμη έτοιμα. Αυτό θα το κάνουμε στη συνέχεια. Ας ξεκινήσουμε από το File/Open & File/Save μενού.

9. Από την παλέττα (εργαλειομπάρα) Dialogs διαλέγουμε το εργαλείο OpenDialog. Και κάνοντας Drop στη φόρμα το τοποθετούμε κι αυτό οπουδήποτε.
10. Στην ιδιότητα Name γράφουμε OpenDialog & στην ιδιότητα Title γράφουμε πχ. Open an Image
11. Στη συνέχεια προσθέτουμε κι ένα SaveDialog εργαλείο
12. Στην ιδιότητα Name γράφουμε SaveDialog & στην ιδιότητα Title γράφουμε πχ. Save an Image

Η φόρμα μας θα μοιάζει κάπως έτσι	

Τώρα θα πρέπει να γράψουμε και τον κώδικα γι' αυτά τα μενού.

13. Από την κεντρική μας φόρμα επιλέγουμε: File/Open, κι εμφανίζεται ο Code Editor, σ' αυτόν γράφουμε τον πιο κάτω κώδικα:

```
14. void __fastcall TForm1::Open1Click(TObject *Sender)
    {
    if (OpenDialog->Execute())
      {
       TChild* child = new TChild(this);
       child->Image->Picure->LoadFromFile(OpenDialog->FileName );
       child->ClientWidth = child->Image->Pictue->Width;
       child->ClientHeight = child->Image->Pictue->Height;
       child->Caption = ExtactFileName(OpenDialog->FileName );
       child->Show();
      }
```

}

(Η θεωρία περί της τάξης MDI child, αναπτύχθηκε παραπάνω, στην παράγραφο MDI (Multple Document Interface model))

Τώρα θα πρέπει να γράψουμε και τον κώδικα γιά το μενού File/Save. Είναι γνωστό ότι μεταξύ φόρμας και κώδικα εναλλασόμαστε με το πλήκτρο F12.

15. Από την κεντρική μας φόρμα επιλέγουμε: File/Save as, κι εμφανίζεται ο Code Editor, σ' αυτόν γράφουμε τον πιο κάτω κώδικα:

```
16. void __fastcall TForm1::SaveAs1Click(TObject *Sender)
    {
     Tchild* child = dynamic_cast<Tchild*>(ActiveMDIChild);
     if "(!child) return;
     if (SaveDialog->Execute())
     {
      child->Image->Pictue->SaveToFile(SaveDialog->FileName);
     }
    }
```

Τώρα φτάσαμε να γράψουμε κώδικα γιά το μενού Window/Tile. Οι εντολές είναι απλές και φαίνονται παρακάτω:

```
17. void __fastcall TForm1::Tile1Click(TObject *Sender)
    {
    Tile();
    }
    //---------------------------------------------------------------------------

    void __fastcall TForm1::Cascade1Click(TObject *Sender)
    {
    Cascade();
    }
    //---------------------------------------------------------------------------

    void __fastcall TForm1::ArrangeAll1Click(TObject *Sender)
    {
    ArrangeIcons();
    }
```

Δημιουργία της MDI child φόρμας

18. File/New Formή από το εικονίδιο New Form, δημιουργούμε την νέα φόρμα
19. Στην ιδιότητα Name γράφουμε Child, την ιδιότητα Caption δεν είναι απαραίτητο να την αλλάξουμε, αφού την αλλάζουμε κατά το runtime

```
    ……..
    child->Caption = ExtactFileName(OpenDialog->FileName );
```

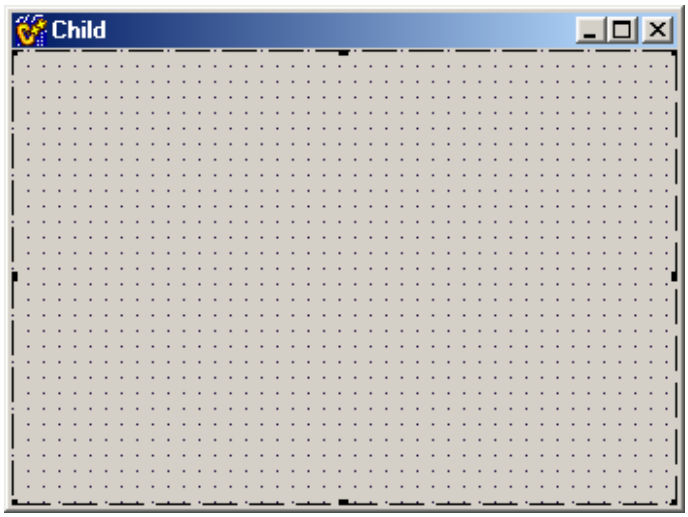
20. Αλλάζουμε την ιδιότητα FormStyle σε fsMDIChild, ώστε η δεύτερη αυτή φόρμα να είναι «θυγατρική φόρμα»

Τώρα για να εισάγουμε Image εργαλείο στην φόρμα

21. Από την Additional εργαλειομπάρα επιλέγουμε το κουμπί Image
22. Κάνουμε Drop στην φόρμα οπουδήποτε, γιατί δεν έχει σημασία η θέση που θα τοποθετήσουμε το εικονίδιο.

23. Στην ιδιότητα Name γράφουμε Image
24. Στην ιδιότητα Stretch επιλέγουμε true
25. Αλλάζουμε την ιδιότητα Align σε alClient, ώστε το Image να περιλάμβάνει όλη την περιοχή της φόρμας

Σώζουμε την φόρμα με όνομα πχ. PrVieChild



26. Γυρίζουμε στον Code Editor με το πλήκτρο F12, σ' αυτόν επιλέγουμε την κεντρική μονάδα πχ. Unit_MDI.cpp και δίνουμε την εντολή File/Include Unit Hdr επιλέγουμε την δεύτερη φόρμα που σώσαμε με όνομα PrVieChild, και δίνουμε OK.

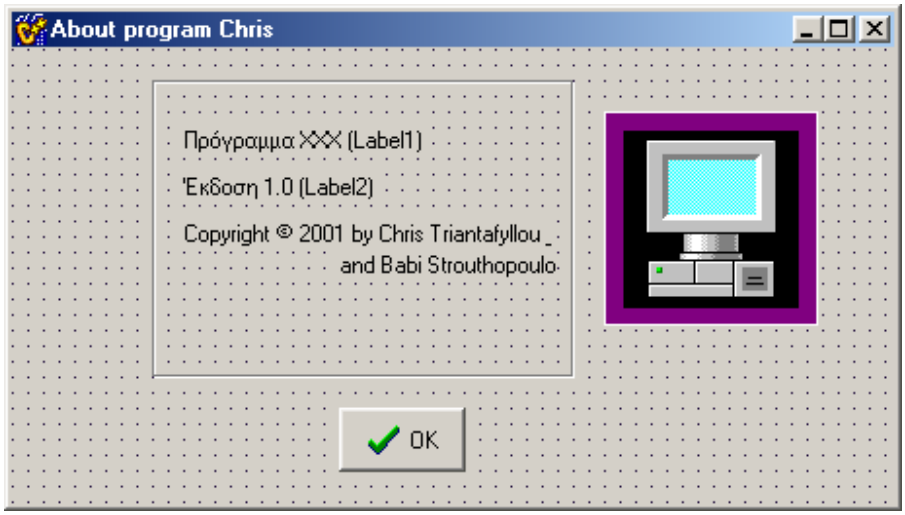
Με αυτό το τελευταίο βήμα προστίθετε στην Unit_MDI.cpp η γραμμή
include "PrVieChild.h"
με την οποία συνδέθηκε η δεύτερη φόρμα PrVieChild με την μονάδα Unit_MDI.cpp.

Δημιουργία νέας φόρμας Dialog Box

27. Μπορούμε να επαναλάβουμε τα τελευταία βήματα, μια και τώρα γνωρίζουμε αρκετά για τον C++Builder, ώστε να δημιουργήσουμε ένα κουτί διαλόγου (Dialog Box) το οποίο να εμφανίζεται όταν κάνουμε κλικ στο μενού Help και την επιλογή About.

Απλά αφού φτιάξουμε την νέα φόρμα και έστω ότι στο Name δώσουμε AboutBox, χρειάζεται να επιλέξουμε από την κεντρική μας φόρμα Help/About και στον Code Editor να γράψουμε μόνο την γραμμή:

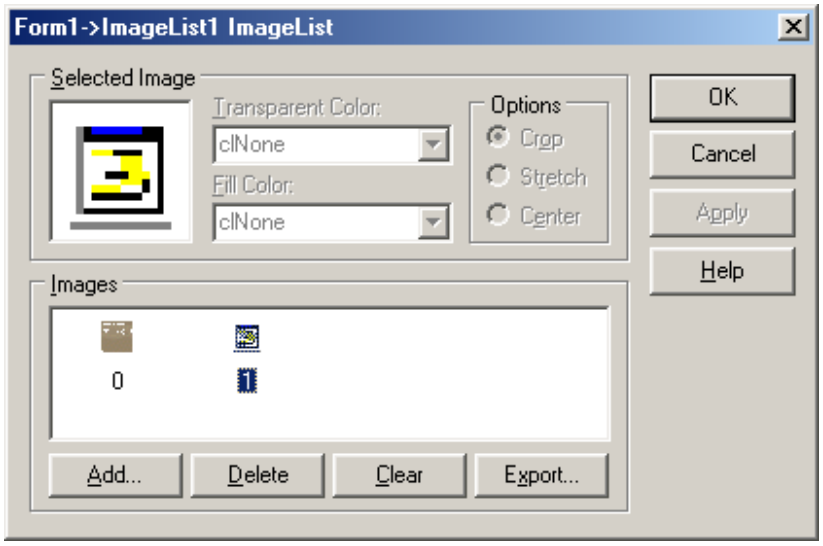
```
AboutBox->ShowModal();
```



Προσθήκη εργαλειομπάρας (Toolbar) στην εφαρμογή μας

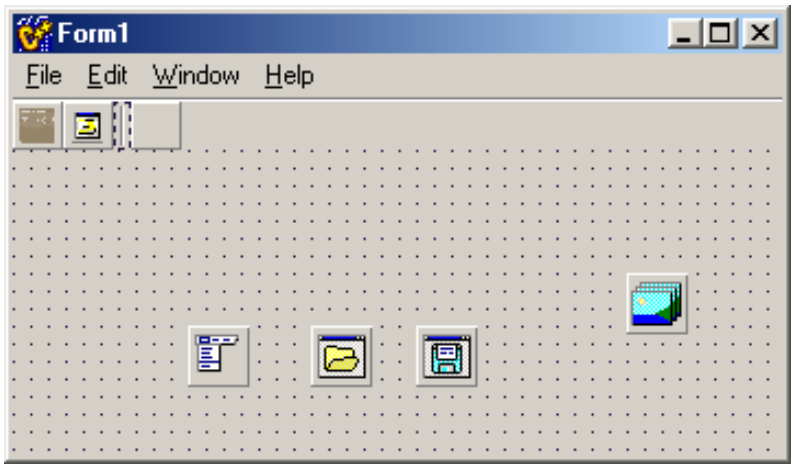
28. Από την Win32 εργαλειομπάρα επιλέγουμε το κουμπί ToolBar
29. Κάνουμε Drop στην φόρμα, και η εργαλειομπάρα αυτόματα τοποθετείται στην κορυφή της φόρμας μας
30. Στην ιδιότητα Name γράφουμε ToolBar
31. Αλλάζουμε την ιδιότητα Flat σε true
32. Αλλάζουμε την ιδιότητα AutoSize σε true, αλλάζοντας την ιδιότητα σε true δίνουμε την δυνατότητα στην εργαλειομπάρα να προσαρμόζει το μέγεθος αυτόματα για κάθε εικονίδιο-εργαλείο που προσθέτουμε σ' αυτή.
33. Για να προσθέσουμε τώρα πλήκτρα-εργαλεία στην εργαλειομπάρα, θα πρέπει με δεξί κλικ σ' αυτή να επιλέξουμε New Button, θα δούμε ένα πλήκτρο να εμφανίζεται σ' αυτή. Μπορούμε να αλλάξουμε αν θέλουμε και την Name του πχ. σε FileOpenButton & την Hint σε Open
34. Επαναλαμβάνουμε το βήμα 33 για όσα πλήκτρα θέλουμε στην Toolbar πχ. ακόμη μία φορά ορίζοντας το πλήκτρο για το Save as…, με Name FileSaveAsButton & Hint Save As
35. Όταν θέλουμε ένα κενό ανάμεσα στα πλήκτρα μας με δεξί κλικ επιλέγουμε New Separator
36. Επαναλαμβάνουμε το βήμα 33 για ένα πλήκτρο Help, με Name πχ. HelpAboutButton & Hint PICTURE VIEWER
37. Επιλέγουμε όλα τα πλήκτρα της εργαλειομπάρας με Shift + click πάνω στο πλήκτρο και αλλάζουμε την ShowHint σε true.
38. Στην καρτέλα Events του Object Inspector του 1ου κουμπιού FileOpenButton, στο OnClick event δίνουμε Open1Click. Έτσι συνδέσαμε το πλήκτρο με το συγκεκριμένο συμβάν.
39. Επαναλαμβάνουμε το ίδιο για τα άλλα δύο πλήκτρα δίνοντας στο OnClick SaveAs1Click & About!Click αντίστοιχα.
40. Από την Win32 εργαλειομπάρα επιλέγουμε το κουμπί ImageList, κάνουμε Drop στην φόρμα.
41. Στην ιδιότητα Name γράφουμε ImageList
42. Με δεξί κλικ στο ImageList εικονίδιο επιλέγουμε ImageList Editor (ή διπλοπατόντας μέσα του). Και προσθέτουμε μία εικόνα (που να δείχνει βέβαια το άνοιγμα αρχείου, μπορούμε να το αντιγράψουμε με το Pbrush (Paint) από άλλη εφαρμογή) ….bmp. Αν το εικονίδιο που επιλέξαμε είναι μεγάλο σε μέγεθος σπάει σε κομμάτια, μπορούμε να επιλέξουμε πιό θέλουμε.

Επαναλαμβάνουμε το ίδιο για τα άλλα δύο πλήκτρα.



43. Στο τελευταίο βήμα, κάνουμε κλικ στην νέα μας εργαλειομπάρα και από τον Object Inspector και την ιδιότητα Images επιλέγουμε από το πτυσόμενο μενού ImageList.

Αν όλα έχουν πάει καλά, τα εικονίδια θα εμφανιστούν μέσα στα π\νέα πλήκρα μας και θα δούμε την παρακάτω φόρμα.



6. ΒΙΒΛΙΟΓΡΑΦΙΑ

- Από την χρήση της Visual Basic 6.0, Visual Studio 2015 (Α' Μέρος)
- Ιστότοπος: https://msdn.microsoft.com/en-us/library/windows/desktop/ms788229.aspx
- Από την χρήση της Borland C++Builder 4 & 5 (Β' Μέρος)

ISBN:

"ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
με Visual Basic & C++ Builder"

Χρήστος Γ. Τριανταφύλλου