Σημειώσεις:

Μάθημα 20

Πίνακες αντικειμένων και ιδιοτήτων Συλλογές

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να ορίζουν πίνακες αντικειμένων.
- Να χρησιμοποιούν τις συλλογές αντικειμένων που έχει προκαθορίσει η VB.

Η έννοια του πίνακα μπορεί να επεκταθεί και στα αντικείμενα ελέγχου που τοποθετούνται πάνω σε μια φόρμα. Ένας ***πίνακας αντικειμένων*** είναι ένα σύνολο αντικειμένων του ιδίου τύπου που μοιράζονται το ίδιο όνομα και τις ίδιες διαδικασίες συμβάντων. Ωστόσο, κάθε αντικείμενο στοιχείο του πίνακα είναι διαφορετικό και έχει τις δικές του τιμές ιδιοτήτων.

Οι πίνακες αντικειμένων έχουν το πλεονέκτημα ότι πολλά αντικείμενα μοιράζονται τον ίδιο κώδικα, άρα μπορούμε να τα χειριστούμε μέσα από ενοποιημένο κώδικα. Όμως το μεγαλύτερο πλεονέκτημα των εφαρμογών στις οποίες γίνεται χρήση πινάκων αντικειμένων είναι ότι μπορούμε να προσθέσουμε ή να αφαιρέσουμε αντικείμενα από έναν πίνακα δυναμικά, κατά την εκτέλεση της εφαρμογής. Αυτό είναι πολύ σημαντικό στις περιπτώσεις που δεν γνωρίζουμε το πλήθος των αντικειμένων που θα απαιτήσει η εφαρμογή κατά την εκτέλεσή της.

Δημιουργία πίνακα αντικειμένων

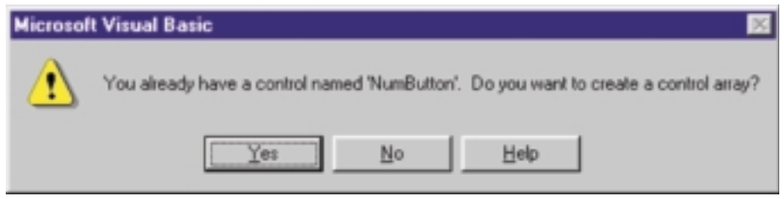
Κατά το σχεδιασμό της διεπαφής μπορούμε να δημιουργήσουμε έναν πίνακα αντικειμένων με τρεις τρόπους:

- Δίνοντας σε δύο αντικείμενα ελέγχου του ίδιου τύπου, το ίδιο όνομα. Το περιβάλλον της VB δημιουργεί αυτόματα έναν πίνακα και χαρακτηρίζει τα δύο αντικείμενα αποδίδοντας στην ιδιότητα **`Index`** *(δείκτης)* τις τιμές 0 και 1. Για κάθε νέο στοιχείο του πίνακα η ιδιότητα **`Index`** έχει και διαφορετική τιμή (2, 3, ... κλπ).
- Δημιουργώντας ένα αντικείμενο από ένα προϋπάρχον αντικείμενο ελέγχου με τη διαδικασία αντιγραφής/επικόλλησης (copy/paste). Τότε, το περιβάλλον της VB μας πληροφορεί ότι υπάρχει ήδη ένα αντικείμενο ελέγχου με το ίδιο όνομα και μας ρωτά αν πράγματι επιθυμούμε να δημιουργήσουμε έναν πίνακα αντικειμένων.
- Δίνοντας τιμή στην ιδιότητα **`Index`** ενός αντικειμένου ελέγχου ενημερώνοντας το αντίστοιχο πεδίο του παραθύρου ιδιοτήτων.

Άσκηση 20-1.

Σε μια εφαρμογή μας ζητείται να δημιουργήσουμε πάνω σε μια φόρμα ένα αριθμητικό πληκτρολόγιο. Το αριθμητικό πληκτρολόγιο μπορεί να θεωρηθεί σαν μια συστοιχία (πίνακας) πλήκτρων διαταγής.

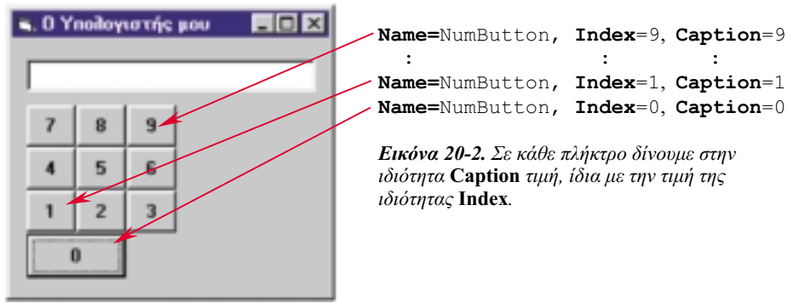
1. Σχεδιάζουμε το πρώτο πλήκτρο διαταγής και δίνουμε τιμές στις ιδιότητες του.
2. Στο πλήκτρο διαταγής δίνουμε το όνομα `NumButton`.
3. Αντιγράφουμε το πλήκτρο και ζητάμε την επικόλλησή του στη φόρμα, ή σχεδιάζουμε ένα νέο πλήκτρο διαταγής και του δίνουμε το ίδιο όνομα με το πρώτο (`NumButton`). Και στις δυο περιπτώσεις, η VB μας ρωτάει αν επιθυμούμε τη δημιουργία πίνακα αντικειμένων.



Εικόνα 20-1.

4. Απαντάμε πατώντας το **Yes**. Αυτόματα, η ιδιότητα **Index** του πρώτου πλήκτρου παίρνει τιμή 0 και του νέου πλήκτρου παίρνει την τιμή 1.
5. Επαναλαμβάνουμε το βήμα 3 μέχρι να προσθέσουμε και τα 10 πλήκτρα. Κάθε ένα πλήκτρο με τη σειρά που προστίθεται, παίρνει τον αμέσως επόμενο δείκτη.

Δεν υπάρχουν δισδιάστατοι ή πολυδιάστατοι πίνακες αντικειμένων. Τα πλήκτρα της αριθμομηχανής μπορεί να έχουν τοποθετηθεί σε γραμμές και σε στήλες, ο δείκτης όμως του πίνακα που ανήκουν είναι ένας.



Εικόνα 20-2. *Σε κάθε πλήκτρο δίνουμε στην ιδιότητα* **Caption** *τιμή, ίδια με την τιμή της ιδιότητας* **Index**.

6. Από τη στιγμή που έχουμε δημιουργήσει το πληκτρολόγιο, για κάθε πλήκτρο δίνουμε κατάλληλη τιμή στην ιδιότητα **Caption**, ώστε το πληκτρολόγιό μας να πάρει τη μορφή που έχει σε μια αριθμομηχανή.

Οι υπορουτίνες διαχείρισης συμβάντων ενός πίνακα αντικειμένων ελέγχου μοιάζουν με τις αντίστοιχες των απλών αντικειμένων ελέγχου, τόσο ως προς τη σύνταξη όσο και ως προς τη συμπεριφορά. Η μόνη διαφορά που μπορούμε να παρατηρήσουμε είναι η ύπαρξη μιας επιπλέον παραμέτρου με όνομα Index. Για παράδειγμα, η υπορουτίνα διαχείρισης συμβάντος **Click** για πίνακα αντικειμένων έχει τη μορφή:

```
Private Sub αντικείμενο_Click(Index As Integer)
        :
End Sub
```

Επίσης, η υπορουτίνα διαχείρισης συμβάντος MouseDown για πίνακα αντικειμένων έχει τη μορφή:

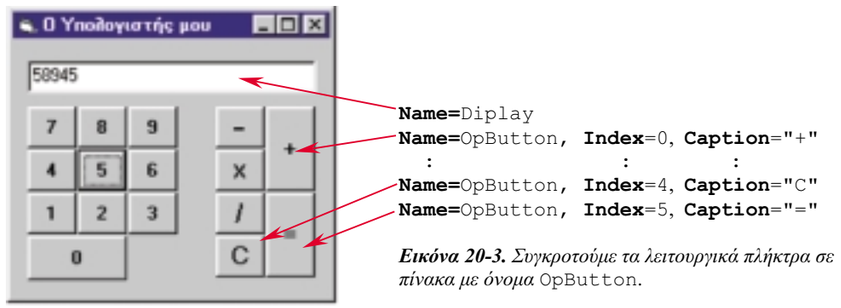
```
Private Sub αντικείμενο_MouseDown(Index As Integer, Button As Integer, _
                                  Shift As Integer, _
                                  X As Single, Y As Single)
        :
End Sub
```

Η τιμή της παραμέτρου Index προσδιορίζει το στοιχείο του πίνακα που προκάλεσε το συμβάν. Με τη βοήθεια αυτού του δείκτη και χρησιμοποιώντας συνήθως τη δομή Select Case … End Select χειριζόμαστε κάθε ένα από τα αντικείμενα.

Τέλος, η αναφορά σε ένα στοιχείο του πίνακα αντικειμένων γίνεται με τη βοήθεια αριθμητικού δείκτη. Ο δείκτης ενός πίνακα γράφεται αμέσως μετά το όνομα του και μέσα σε παρενθέσεις.

Άσκηση 20-2.

Συμπληρώστε την άσκηση του πληκτρολογίου. Τοποθετήστε πάνω στη φόρμα τα λειτουργικά πλήκτρα και αποδώστε τους λειτουργίες.



Εικόνα 20-3. *Συγκροτούμε τα λειτουργικά πλήκτρα σε πίνακα με όνομα* OpButton.

Αφού τοποθετήσουμε τα λειτουργικά πλήκτρα όπως φαίνεται στην εικόνα 20-3, γράφουμε το πρόγραμμα:

```
Option Explicit
Dim Buffer         As Long      'á' üñïò ôçò ðñÜîçò
Dim Old_OpIndex    As Integer   'Ç ðñÜîç ðïõ åß÷å åðéëåãåß
Dim OpPressed      As Boolean   'Å÷åé åðéëåãåß ðñÜîç
Private Sub Form_Load()
    Display.Text = 0            'ÌçäÝíéóå ôçí ïèüíç
    OpPressed = False           'Äåí Ý÷åé åðéëåãåß ðñÜîç
End Sub

Private Sub NumButton_Click(Index As Integer)
    If OpPressed = False Then
        'Áí äåí Ý÷åé ðáôçèåß ðëÞêôñï ðñÜîçò, ðñüóèåóá øçößï óôï ôÝëïò
         Display.Text = Val(Display.Text & Index)
    Else
        'Áí Ý÷åé ðáôçèåß ðëÞêôñï ðñÜîçò, óõãêåñÜóçóå ìüíï áõôü ôï øçößï
         Display.Text = Index
         OpPressed = False      'Áêýñùóå ôçí ðñÜîç
    End If
End Sub

Private Sub OpButton_Click(Index As Integer)
    Select Case Index
        Case 0, 1, 2, 3         '+, -, *, /
            Buffer = Display.Text
        Case 4                  'C. ÐëÞêôñï äéáãñáöÞò
            Buffer = 0
            Display.Text = 0
        Case 5                  '=  Åîáãùãþ áðïôåëÝóìáôïò
            Select Case Old_OpIndex
                Case 0
                    Display.Text = Buffer + Display.Text
                Case 1
                    Display.Text = Buffer - Display.Text
                Case 2
                    Display.Text = Buffer * Display.Text
                Case 3
                    Display.Text = Buffer / Display.Text
            End Select
    End Select
    Old_OpIndex = Index         'ÓõãêñÜôçóå ôçí ðñÜîç
    OpPressed = True            'Ðáôþèçêå ðëÞêôñï ðñÜîçò
End Sub
```

Κάθε φορά που πατάμε ένα από τα αριθμητικά πλήκτρα, εκτελείται η υπορουτίνα NumButton_Click. Η υπορουτίνα αυτή προσθέτει στο τέλος του αριθμού, που υπάρχει ήδη στην οθόνη, το ψηφίο που αντιστοιχεί στο πλήκτρο που μόλις πατήθηκε. Αν όμως έχει προηγηθεί πάτημα πλήκτρου πράξης, διαγράφει την οθόνη και τοποθετεί σε αυτήν μόνο του το ψηφίο που μόλις πατήσαμε.

Επίσης, κάθε φορά που πατάμε ένα από τα λειτουργικά πλήκτρα, εκτελείται η υπορουτίνα OpButton_Click. Αν το πλήκτρο είναι πλήκτρο πράξης, μεταφέρεται ο αριθμός που υπάρχει στην οθόνη, σε έναν προσωρινό αποθηκευτή Buffer και ταυτόχρονα συγκρατείται και η πράξη. Ο χρήστης μπορεί να πληκτρολογήσει τον δεύτερο αριθμό. Μόλις ο χρήστης πατήσει το πλήκτρο "=" καλείται πάλι η υπορουτίνα OpButton_Click, που αυτή τη φορά εκτελεί την πράξη που είχε επιλεγεί μεταξύ του περιεχομένου του προσωρινού αποθηκευτή και του αριθμού που υπάρχει στην οθόνη.

Δυναμική δημιουργία πίνακα αντικειμένων

Από τη στιγμή που έχουμε δημιουργήσει έναν πίνακα αντικειμένων μπορούμε, κατά τη διάρκεια εκτέλεσης του προγράμματος, να του επισυνάψουμε νέα στοιχεία. Η επέκταση ενός πίνακα με νέα στοιχεία γίνεται με τη μέθοδο Load.

Η γενική μορφή της μεθόδου είναι:

Load *αντικείμενο(δείκτης)*

Η μέθοδος αυτή κληροδοτεί στο νέο στοιχείο, όλες τις ιδιότητες που έχει το στοιχείο με τον μεγαλύτερο δείκτη εκτός των ιδιοτήτων **Visible**, **Index** και **TabIndex**. Όταν θέλουμε λοιπόν να εμφανίσουμε το νέο αντικείμενο πρέπει να θέσουμε την ιδιότητά του **Visible** σε True. Όμως, ακόμα και τότε δε θα διακρίνουμε το αντικείμενο, διότι έχει τις ιδιότητες θέσης **Top**, **Left**, **Height** και **Width** ίσες με του πατρικού του και κατά συνέπεια το καλύπτει. Πρέπει λοιπόν να δοθούν διαφορετικές τιμές και σε αυτές τις ιδιότητες.

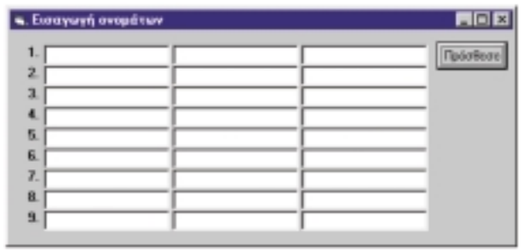
Ο πρώτος δείκτης ενός πίνακα αντικειμένων ελέγχου έχει τιμή 0, ανεξάρτητα από το τι ορίζει η εντολή Option Base.

Άσκηση 20-3.

Να δημιουργηθεί φόρμα με τρία πλαίσια κειμένου στη σειρά, στα οποία να καταχωρίζουμε το όνομα, το πατρώνυμο και το επώνυμο ατόμων. Αριστερά των πλαισίων κειμένου να τοποθετηθεί μια ετικέτα, στην οποία να εμφανίζεται ένας αύξοντας αριθμός. Να γραφτεί πρόγραμμα που με το πάτημα του πλήκτρου να προσθέτει μια σειρά ετικετών.

1. Δημιουργούμε φόρμα και τοποθετούμε μια ετικέτα με όνομα RowLabel και πλαίσια κειμένου με ονόματα FirstName, FatherName, SurName.
2. Χαρακτηρίζουμε τα αντικείμενα ως στοιχεία πίνακα, καταχωρίζοντας στην ιδιότητα **Index** την τιμή 0.
3. Φροντίζουμε να στοιχιστούν τα αντικείμενα και να αποκτήσουν το ίδιο ύψος.
4. Τέλος, τοποθετούμε ένα πλήκτρο διαταγής με όνομα AddRow και του επισυνάπτουμε τον κώδικα:

```
Private Sub AddRow_Click()
    Dim Row         As Integer
   'Áðáñßèçóå ôç óåéñÜ
    Row = RowLabel.UBound + 1
   'Äçìéïýñãçóå ôá óôïé÷åßá ôçò íÝáò óåéñÜò
    Load RowLabel(Row)
    Load FirstName(Row)
    Load FatherName(Row)
    Load SurName(Row)
   'Áñßèìiçóå & êáèÜñéóå ôï ðåñéå÷üìåíï ôùí óôïé÷åßùí
    RowLabel(Row).Caption = (Row + 1) & "."
    FirstName(Row).Text = ""
    FatherName(Row).Text = ""
    SurName(Row).Text = ""
   'Ìåôáêßíçóå ôç óåéñÜ óå íÝá èÝóç
    RowLabel(Row).Top = RowLabel(Row - 1).Top + RowLabel(0).Height
    FirstName(Row).Top = FirstName(Row - 1).Top + FirstName(0).Height
    FatherName(Row).Top = FatherName(Row - 1).Top + FatherName(0).Height
    SurName(Row).Top = SurName(Row - 1).Top + SurName(0).Height
   'Áìöüíéóå ôç óåéñÜ
    RowLabel(Row).Visible = True
    FirstName(Row).Visible = True
    FatherName(Row).Visible = True
    SurName(Row).Visible = True
End Sub
```



Εικόνα 20-4. *Το παράθυρο της εφαρμογής μετά από την εισαγωγή 8 γραμμών.*

Για να δημιουργήσουμε με τη μέθοδο Load το επόμενο στοιχείο ενός πίνακα χρειαζόμαστε τον επόμενο δείκτη. Επειδή, όπως προαναφέραμε, η Load δεν αντιγράφει ούτε και υπολογίζει αυτόματα την τιμή της ιδιότητας **Index**, για να προσδιορίσουμε την τιμή του επόμενου δείκτη πρέπει να συμβουλευτούμε την ιδιότητα **Ubound** του πίνακα, που έχει τιμή ίση με την τιμή του δείκτη του τελευταίου στοιχείου.

Ας σημειωθεί ότι αν προσπαθήσουμε να δημιουργήσουμε ένα αντικείμενο που υπάρχει ήδη, προκύπτει λάθος εκτέλεσης.

Για να απομακρύνουμε ένα αντικείμενο από τον πίνακα αντικειμένων χρησιμοποιούμε τη μέθοδο Unload. Η γενική μορφή της μεθόδου είναι:

UnLoad *αντικείμενο(δείκτης)*

Η μέθοδος μπορεί να δράσει μόνο στα αντικείμενα που έχουμε δημιουργήσει με τη μέθοδο Load κατά την εκτέλεση της εφαρμογής. Αν προσπαθήσουμε να τη χρησιμοποιήσουμε για να απομακρύνουμε αντικείμενα που έχουμε δημιουργήσει κατά τη διάρκεια της σχεδίασης της διεπαφής, προκύπτει λάθος εκτέλεσης.

Άσκηση 20-4.

Να προστεθεί πλήκτρο διαταγής στη φόρμα της άσκησης 20-3, του οποίου το πάτημα να προκαλεί την αφαίρεση της τελευταίας γραμμής. Αφού προσθέσουμε το πλήκτρο διαταγής του επισυνάπτουμε τον κώδικα:

```
Private Sub SubRow_Click()
    Dim Row     As Integer
    Row = RowLabel.UBound - 1
    If Row <0 Then
       Unload RowLabel(Row)
       Unload FirstName(Row)
       Unload FatherName(Row)
       Unload SurName(Row)
       Row = Row - 1
    End If
End Sub
```

Παρατηρήστε ότι φροντίσαμε να μην αφαιρείται ποτέ η πρώτη γραμμή.

Πίνακες ιδιοτήτων

Σε μερικά αντικείμενα, μια ιδιότητα μπορεί να έχει ταυτόχρονα πολλές διαφορετικές τιμές. Σε αυτήν την περίπτωση, οι τιμές της ιδιότητας είναι οργανωμένες σε πίνακα και γι' αυτό λέμε ότι η ιδιότητα αποτελεί ιδιότητα πίνακα. Η αναφορά σε ένα συγκεκριμένο στοιχείο της ιδιότητας πίνακα γίνεται με τη χρήση δείκτη.

Παράδειγμα 20-1.

Οι γραμματοσειρές της οθόνης ενός υπολογιστικού συστήματος περιέχονται στον πίνακα ιδιοτήτων **Fonts** του αντικειμένου Screen. Το πλήθος των στοιχείων του πίνακα αναφέρεται από την ιδιότητα **FontCount**. Η παρουσίαση των εγκατεστημένων γραμματοσειρών της οθόνης δίνεται από τον κώδικα:

Οι γραμματοσειρές της οθόνης

```
Dim Message As String
Dim i       As Integer
Message = ""
For i = 0 To Screen.FontCount
Message = Message & Screen.Fonts(i) & vbCrLF
Next i
MsgBox Message
```

Παράδειγμα 20-2.

Οι γραμματοσειρές του εκτυπωτή

Οι γραμματοσειρές του εκτυπωτή περιέχονται στον πίνακα ιδιοτήτων **Fonts** του αντικειμένου `Printer`. Το πλήθος των στοιχείων του πίνακα το δίνει η ιδιότητα **FontCount**. Η παρουσίαση των κοινών για την οθόνη και τον εκτυπωτή γραμματοσειρών γίνεται με το τμήμα του κώδικα:

```
Dim i  As Integer
Dim j   As Integer
'ÓÜñùóå üëåò ôéò ãñáììáôïóåéñÝò ôçò ïèüíçò
For i = 0 To Screen.FontCount - 1
   'Äéá êÜèå ãñáììáôïóåéñÜ ôçò ïèüíçò âñåò áíôßóôïé÷ç óôïí åêôõðùôÞ
    For j = 0 To Printer.FontCount - 1
        If Screen.Fonts(i) = Printer.Fonts(j) Then
            Print Screen.Fonts(i)
            Exit For
        End If
    Next j
Next i
```

Συλλογές

Η ***συλλογή (collection)*** είναι ένα αντικείμενο που περιέχει άλλα αντικείμενα που έχουν όλα τον ίδιο τύπο. Η συλλογή μοιάζει με πίνακα αντικειμένων. Ενώ όμως, έναν πίνακα αντικειμένων δεν μπορούμε να τον διαχειριστούμε σαν ολότητα, μια συλλογή μπορούμε. Για παράδειγμα, μέσα από κώδικα μπορούμε να αναφερθούμε στη συλλογή ή και τα στοιχεία της, ενώ δεν μπορούμε να αναφερθούμε στον πίνακα αυτόν καθεαυτόν αν και μπορούμε να αναφερθούμε σε συγκεκριμένα στοιχεία του. Για να καταλάβουμε αυτή τη διαφορά, ας φανταστούμε ένα σύνολο γραμματοσήμων, τα οποία βρίσκονται τοποθετημένα μέσα σε ένα άλμπουμ. Κάθε γραμματόσημο αποτελεί ένα στοιχείο του άλμπουμ, το ίδιο το άλμπουμ αποτελεί τη συλλογή. Αν στην VB θεωρούσαμε το άλμπουμ σαν πίνακα δεν θα μπορούσαμε να αναφερθούμε σε αυτό (π.χ. το άλμπουμ μου: `MyAlbum`, το άλμπουμ του Νίκου: `NicAlbum`), αλλά μόνο σε κάθε ένα στοιχείο του (π.χ. το πρώτο γραμματόσημο: `MyAlbumStamp(1)` κλπ).

Σε κάθε στοιχείο της συλλογής μπορούμε να αναφερθούμε αναφέροντας το όνομα της συλλογής και έναν δείκτη. Ακόμα, σε ένα στοιχείο της συλλογής μπορούμε να αναφερθούμε αναφέροντας το όνομα της συλλογής και το όνομα του αντικειμένου. Δηλαδή, τα στοιχεία της συλλογής είναι δυνατόν να έχουν ξεχωριστά ονόματα σε αντίθεση με τα στοιχεία ενός πίνακα τα οποία έχουν ένα κοινό όνομα. Μια συλλογή μπορεί να συμπεριφερθεί και σαν λίστα. Όπως η λίστα έχει ιδιότητες, έτσι και η συλλογή έχει ιδιότητες. Για παράδειγμα η ιδιότητα **Count** δείχνει πόσα στοιχεία περιέχει.

Στην VB προϋπάρχουν συλλογές αντικειμένων. Για παράδειγμα, σε κάθε έργο (project) υπάρχει μια συλλογή από φόρμες, η συλλογή `Forms`. Επίσης, όλοι οι διαθέσιμοι εκτυπωτές ενός υπολογιστικού συστήματος περιέχονται στη ***συλλογή αντικειμένων*** εκτυπωτών ***(Printers Collection)***. Οι εκτυπωτές αυτοί είναι όσοι εμφανίζονται στο διαλογικό παράθυρο **Printers** του πίνακα ελέγχου (Control Panel) των Windows.

Παραδείγματα 20-3.

Συλλογή οθονών

Όταν θέλουμε να αναφερθούμε σε μια από τις φόρμες χρησιμοποιούμε έναν δείκτη. Ο δείκτης μπορεί να πάρει τιμές από `0` έως `Forms.Count-1`. Το τμήμα προγράμματος:

```
Dim Message As String
Dim i        As Integer
Message = ""
For i = 0 To Forms.Count - 1
    Message = Message & Forms(i).Name & vbCrLf
Next i
MsgBox Message
```

σαρώνει όλα τα μέλη της συλλογής `Forms` και αναφέρει τα ονόματα των φορμών που υπάρχουν μέσα στο τρέχον πρόγραμμα.

Παραδείγματα 20-4.

Συλλογή εκτυπωτών

α) Όταν θέλουμε να αναφερθούμε σε έναν από τους εκτυπωτές του συστήματος χρησιμοποιούμε έναν δείκτη, ο οποίος προσδιορίζει τον συγκεκριμένο εκτυπωτή μέσα στη συλλογή `Printers` των εκτυπωτών. Ο δείκτης μπορεί να πάρει τιμές από `0` έως `Printers.Count-1`. Για να εκχωρήσουμε στο αντικείμενο `Printer` έναν συγκεκριμένο εκτυπωτή, μέσα από τη συλλογή των εκτυπωτών, χρησιμοποιούμε την εντολή:

`Set Print = Printers(`*δείκτης*`)`

β) Η ιδιότητα **DeviceName** αναφέρεται στο όνομα του εκτυπωτή και η ιδιότητα **Port** περιέχει το όνομα της θύρας στην οποία είναι συνδεδεμένος ο εκτυπωτής. Το τμήμα προγράμματος:

```
Dim Message As String
Dim i       As Integer
Message = ""
For i = 0 To Printers.Count - 1
    Message = Message & Printers(i).DeviceName & " óôç èýñá " & _
              Printers(i).Port & vbCrLf
Next i
MsgBox Message
```

αναφέρει τους εκτυπωτές και τις πόρτες στις οποίες είναι συνδεδεμένο.

Ανακεφαλαίωση

Ένας πίνακας αντικειμένων είναι ένα σύνολο αντικειμένων του ιδίου τύπου που μοιράζονται το ίδιο όνομα και τις ίδιες διαδικασίες συμβάντων. Κατά το σχεδιασμό της διεπαφής μπορούμε να δημιουργήσουμε έναν πίνακα αντικειμένων με τρεις τρόπους. Να δώσουμε σε δύο αντικείμενα ελέγχου του ίδιου τύπου, το ίδιο όνομα, να δημιουργούμε ένα αντικείμενο από ένα προϋπάρχον αντικείμενο ελέγχου με τη διαδικασία αντιγραφής/επικόλλησης, να δώσουμε τιμή στην ιδιότητα **Index** ενός αντικειμένου ελέγχου.

Σε μερικά αντικείμενα, μια ιδιότητα μπορεί να έχει ταυτόχρονα πολλές διαφορετικές τιμές. Σε αυτήν την περίπτωση, οι τιμές της ιδιότητας είναι οργανωμένες σε πίνακα και αναφερόμαστε σε ιδιότητα πίνακα. Η συλλογή είναι ένα αντικείμενο που περιέχει άλλα αντικείμενα που έχουν όλα τον ίδιο τύπο. Στην VB προϋπάρχουν συλλογές αντικειμένων, όπως η συλλογή με τις φόρμες του έργου, η συλλογή με τους εκτυπωτές κ.ά.

Εργαστηριακές ασκήσεις

1. Να εκτελεστεί η άσκηση 20-1.
2. Να εκτελεστεί η άσκηση 20-2. Να χρησιμοποιηθεί το πληκτρολόγιο για να γίνουν αριθμητικές πράξεις.
3. Να βελτιωθεί το απλό αριθμητικό πληκτρολόγιο και να εξελιχθεί σε ένα που να μπορεί να υπολογίζει τριγωνομετρικές συναρτήσεις, υψώσεις σε εκθέτη, λογαρίθμους κ.ά.
4. Να εκτελεστεί η άσκηση 20-3 και στη συνέχεια η 20-4.
5. Στον φάκελο `"C:\Program Files\DevStudio\VB\Samples\Elements"` υπάρχουν τα αρχεία `Moon1.ico`, `Moon2.ico`, . . . , `Moon8.ico` με τις φάσεις της σελήνης. Να μεταφερθούν οι εικόνες σε πίνακα αντικειμένων εικόνων και να γραφεί κώδικας που να δημιουργεί κινούμενο σχέδιο με τις φάσεις της σελήνης.
   Υπόδειξη: Πρόκειται για παραλλαγή της άσκησης 16-2.
6. Να γραφεί πρόγραμμα που να εμφανίζει σε λίστα και μάλιστα ταξινομημένες τις γραμματοσειρές της οθόνης.
7. Να επαναληφθεί η άσκηση 5 για τις γραμματοσειρές του εκτυπωτή.
8. Η συλλογή `Controls` είναι η συλλογή των αντικειμένων ελέγχου κάθε φόρμας. Να γραφτεί τμήμα προγράμματος, το οποίο να σαρώνει όλες τις φόρμες ενός έργου και να τυπώνει τα αντικείμενα ελέγχου κάθε μιας.
9. Να δημιουργηθεί φόρμα από την οποία να είναι δυνατή η επιλογή της γραμματοσειράς, του μεγέθους, του τύπου και του χρώματος των γραμμάτων των πεδίων κειμένου μιας άλλης φόρμας.

Σημειώσεις:

Μάθημα 21
Υπορουτίνες και Συναρτήσεις

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να επιχειρηματολογούν για τις δυνατότητες του αρθρωτού προγραμματισμού.
- Να συντάσσουν προγράμματα δομημένα από συναρτήσεις και υποροτίνες.
- Να περιγράφουν τις διαφορές μεταξύ υπορουτινών και συναρτήσεων.
- Να αναγνωρίζουν τις διαφορές του τρόπου περάσματος δεδομένων σε υπορουτίνες και συναρτήσεις

·Η περιγραφή και η ανάλυση ενός σύνθετου έργου μπορεί να γίνει εύκολα και απλά, αν το τεμαχίσουμε σε μικρότερα ***τμήματα (modules)***, περιγράψουμε καθένα από αυτά χωριστά με άλλα μικρότερα τμήματα κ.ο.κ. Αυτή η λογική μεταφέρεται και στην κωδικοποίηση. Αν έχουμε να περιγράψουμε μια σύνθετη και μεγάλη 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 για κάθε υποδιαδικασία γράφεται μόνο μια φορά και δεν απαιτείται η γραφή του σε όλα τα σημεία, στα οποία χρειάζεται να εκτελεστεί η υποδιαδικασία. Σε αυτά τα σημεία αρκεί να γίνει η κλήση του.
- Εύκολη γραφή, γρήγορος έλεγχος και απλή συντήρηση. Αν χρειαστεί να γίνουν τροποποιήσεις στον τρόπο λειτουργίας μιας υποδιαδικασίας, αυτό θα γίνει σε ένα μόνο σημείο, επομένως αποφεύγεται η πολλαπλή αναζήτηση και διόρθωση. Αν δεν ακολουθήσουμε αυτόν τον τρόπο προγραμματισμού και ο κώδικας της υποδιαδικασίας επαναλαμβάνεται σε πολλά σημεία, υπάρχει περίπτωση να μας ξεφύγουν κάποιες από τις διορθώσεις με αποτέλεσμα να παρουσιάζει το πρόγραμμα διαφορές σε σημεία που θα έπρεπε να έχει την ίδια συμπεριφορά.
- Παραστατική παρουσίαση και γρήγορη κατανόηση από τρίτους. Ο κώδικας γίνεται μικρότερος και αποκτά μεγαλύτερη συνοχή, είναι λοιπόν δυνατή η εύκολη συντήρησή του. Οι σημερινές εφαρμογές γράφονται από περισσότερους από έναν προγραμματιστές. Ο κάθε προγραμματιστής αναλαμβάνει την κωδικοποίηση κάποιων τμημάτων. Αν η ομάδα ακολουθήσει αυτήν την τυποποίηση κάθε μέλος μπορεί να εκμεταλλευτεί τη δουλειά των άλλων μελών της ομάδας. · Ταχύτατη και εύκολη ανάπτυξη νέων εφαρμογών. Την ίδια δουλειά μπορούμε να την ενσωματώσουμε σε πολλές διαφορετικές εφαρμογές. Πολλές μάλιστα εφαρμογές αποτελούν συρραφή τμημάτων εφαρμογών που έχουμε αναπτύξει στο παρελθόν. Αυτό γίνεται συνήθως σε διαδικασίες και υποδιαδικασίες γενικού σκοπού. Η ενσωμάτωση συμβάλλει επιπλέον και στην ομοιομορφία συμπεριφοράς των εφαρμογών

Στη VB η κωδικοποίηση των υποδιαδικασιών γίνεται σε ανεξάρτητα τμήματα κώδικα σαφώς ορισμένα, τις ***υπορουτίνες (subroutines)*** και τις ***συναρτήσεις (functions)***, που τοποθετούνται μέσα σε προγραμματιστικές μονάδες. Με αυτήν τη λογική είμαστε εξοικειωμένοι σε κάποιο βαθμό, αφού τα προγράμματα που έχουμε αναπτύξει μέχρι τώρα δεν είναι μονολιθικά, αλλά αποτελούνται από υπορουτίνες διαχείρισης συμβάντων, που βρίσκονται σε προγραμματιστικές μονάδες φόρμας.

Υπορουτίνες γενικού σκοπού

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 όταν γίνεται κλήση τους από άλλα σημεία του κώδικα.

Οι υπορουτίνες γενικού σκοπού μπορούν να τοποθετηθούν σε οποιοδήποτε σημείο μιας προγραμματιστικής μονάδας. Συντάσσονται όπως ακριβώς και οι υπορουτίνες διαχείρισης συμβάντων. Αρχίζουν με την δηλωτική εντολή Sub, στην οποία δηλώνεται το όνομά τους οι παραμέτροι τους και οι τύποι των παραμέτρων τους. Αμέσως μετά γράφονται οι εντολές που κωδικοποιούν τη διαδικασία, που επιτελεί η υπορουτίνα. Το πέρας της υπορουτίνας δηλώνεται με την εντολή End Sub. Η γενική μορφή των υπορουτινών γενικού σκοπού είναι:

```
Sub όνομα_υπορουτίνας(παράμετρος1 As τύπος1, παράμετρος2 As τύπος2, ...)
    εντολή_1
    εντολή_2
      :
End Sub
```

Υπάρχουν δύο τάσεις για τη διάταξη των υπορουτινών μέσα στην προγραμματιστική μονάδα. Η μια θέλει τις υπορουτίνες σε αλφαβητική διάταξη και η άλλη θέλει τις υπορουτίνες σε ομάδες ανάλογα με τη χρήση τους και την κλήση τους. Στόχος και των δύο τάσεων είναι η δημιουργία ενός κομψού και τακτικού προγράμματος με απώτερο σκοπό την εύκολη ανεύρεσή τους από τον προγραμματιστή.

Μια υπορουτίνα μπορεί να μην έχει παραμέτρους. Σε αυτήν την περίπτωση ανταλλάσει δεδομένα με το υπόλοιπο πρόγραμμα μέσω καθολικών μεταβλητών ή με μεταβλητές της προγραμματιστικής μονάδας που ανήκει. Οι υπορουτίνες εκτελούνται όταν καλούνται. Η ***κλήση (call)*** μιας υπορουτίνας γίνεται με αναφορά του ονόματός της και παράθεση των παραμέτρων της (αν υπάρχουν) σε μια γραμμή κώδικα.

Παράδειγμα 21-1.

Έστω, ότι σε δύο - τρία σημεία ενός προγράμματος, απαιτείται να μηδενίζουμε τα στοιχεία ενός πίνακα Α, που το πρώτο του στοιχείο είναι το Α(1) και το τελευταίο του το στοιχείο Α(100). Στο παράθυρο κώδικα της προγραμματιστικής μονάδας θα μπορούσαμε να γράψουμε:

```
    :
'ÌçäÝíéóå  ôï Á
 For  i = 1 To 100
      A(i)  = 0
 Next  i
    :
'ÌçäÝíéóå ôï Á
 For  i = 1 To 100
      A(i)  = 0
 Next  i
    :
'ÌçäÝíéóå  ôï Á
 For  i = 1 To 100
      A(i)  = 0
 Next  i
    :
```

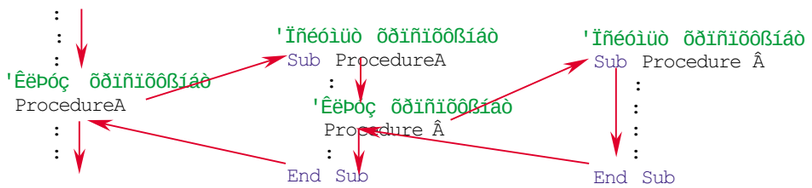
Επαναλήψεις του ίδιου κώδικα

```
'Ïñéóå ôçí ðñïñïõðßíá
 Sub ClearArrayA
     For  i = 1 To 100
         A(i) = 0
     Next i
 End Sub
    :
'ÌçäÝíéóå ôï Á
 ClearArrayA
    :
'ÌçäÝíéóå ôï Á
 ClearArrayA
    :
'ÌçäÝíéóå ôï Á
 ClearArrayA
    :
```

Κλήσεις υπορουτίνας

Σχήμα 21-1. *α) Κώδικας χωρίς υπορουτίνα. β) Κώδικας με υπορουτίνα*

Στο δεξί παράθυρο κώδικα, στα σημεία στα οποία γίνεται κλήση της υπορουτίνας υποδεικνύεται η μεταφορά του ελέγχου του προγράμματος στην περιοχή ορισμού της. Ακολουθεί η εκτέλεση του κώδικα στο εσωτερικό της υπορουτίνας και με τον τερματισμό της η εκτέλεση του προγράμματος συνεχίζεται στην εντολή που βρίσκεται αμέσως μετά το σημείο που έκανε την κλήση της. Στο εσωτερικό μιας υπορουτίνας γενικού σκοπού μπορεί να πραγματοποιηθεί η κλήση άλλης υπορουτίνας κ.ο.κ. (ένθεση κλήσης). Αν γράφαμε τις διαδοχικά καλούμενες υπορουτίνες τη μια δίπλα στην άλλη, θα μπορούσαμε να παραστήσουμε τη ροή του προγράμματος όπως στο σχήμα 21-2:



Σχήμα 21-2. *Κλήση υπορουτίνας από υπορουτίνα*

Η έξοδος από μια υπορουτίνα δεν είναι υποχρεωτικό να γίνει από το τέλος της. Μπορεί να γίνει και ενδιάμεσα αν υπάρχει εντολή Exit Sub. Η εντολή End Sub δηλώνει το φυσικό τέλος της υπορουτίνας, ενώ η εντολή Exit Sub το λογικό της τέλος. Μια υπορουτίνα μπορεί να έχει πολλά σημεία εξόδου από πολλές εντολές Exit Sub αλλά μόνο ένα φυσικό τέλος.

Οι παράμετροι

Με την έννοια της παραμέτρου είμαστε λίγο πολύ εξοικειωμένοι, αφού έχουμε χρησιμοποιήσει παραμέτρους στις ενσωματωμένες συναρτήσεις και στις μεθόδους των αντικειμένων. Επίσης, έχουμε παρατηρήσει, ότι σε πολλές υπορουτίνες διαχείρισης συμβάντων, όπως στις υπορουτίνες **MouseMove**, **DragOver**, **DragDrop**, γίνεται δήλωση παραμέτρων. Τις παραμέτρους αυτές τις χρησιμοποιήσαμε μηχανικά, χωρίς να μπούμε σε λεπτομέρειες επί του τρόπου λειτουργίας τους. Τώρα θα ξεκινήσουμε από την αρχή εκθέτοντας πρώτα απ' όλα την ανάγκη εισαγωγής και χρήσης παραμέτρων.

Παράδειγμα 21-2.

Έστω ότι μας ζητούν να γράψουμε μια υπορουτίνα, που να υπολογίζει τον μέσο όρο των στοιχείων ενός πίνακα Α, που έχει Ν στοιχεία. Βασιζόμενοι, σε ότι γνωρίζουμε μέχρι τώρα, θα γράφαμε:

```
Sub CalcMean
    Dim Sum       As Double
    Dim MeanValue As Double
    Dim i          As Long
    Sum = 0
    For i = 1 To N
        Sum = Sum + A(i) 'Ìåãéêü Üèñïéóìá
    Next I
    MeanValue = Sum / N
End Sub
```

Η υπορουτίνα CalcMean επιτελεί μια συγκεκριμένη λειτουργία, πάνω σε συγκεκριμένα δεδομένα, τα στοιχεία του πίνακα Α. Πώς θα αντιμετωπίζαμε όμως το πρόβλημα, σε περίπτωση που θα θέλαμε η υπορουτίνα να βρίσκει το μέσο όρο των στοιχείων ενός οποιουδήποτε πίνακα, έστω του Β, του C, κλπ; Φυσικά, πριν από την κλήση της υπορουτίνας, θα έπρεπε να έχουμε μεταφέρει τα στοιχεία του προς επεξεργασία πίνακα, στα στοιχεια του πίνακα Α. Σε ένα πρόγραμμα με πολλές κλήσεις της υπορουτίνας θα γράφαμε:

```
    :
'ÌÝóïò üñïò ôùí óôïé÷åßùí ôïõ Â
 For i = 1 To 100:  A(i) = B(i):  Next i
 CalcMean
    :
'ÌÝóïò üñïò ôùí óôïé÷åßùí ôïõ C
 For i = 1 To 100:  A(i) = C(i):  Next i
 CalcMean
    :
'ÌÝóïò üñïò ôùí óôïé÷åßùí ôïõ D
 For i = 1 To 100:  A(i) = D(i):  Next i
 CalcMean
    :
```

Αυτός ο τρόπος κωδικοποίησης κάνει το πρόγραμμα περισσότερο σύνθετο και δεν είναι εξυπηρετικός αφού του προσθέτει επιπλέον κώδικα. Για τις ανάγκες τις υπορουτίνας χρειάζεται να ορίσουμε ένα σύνολο μεταβλητών κατάλληλης εμβέλειας, που να τις χρησιμοποιούμε ως μεταβλητές επικοινωνίας με το υπόλοιπο πρόγραμμα. Πριν από την κλήση της υπορουτίνας πρέπει να γίνεται αρχικοποίηση των μεταβλητών και μετά την κλήση της, αντιγραφή για λήψη των αποτελεσμάτων. Επιπλέον, η διαχείριση αυτών των μεταβλητών χρειάζεται προσοχή, αφού είναι δυνατόν να αλλοιώνεται η τιμή τους, χωρίς αυτό να είναι επιθυμητό, από άλλες υπορουτίνες που καλούνται στο εσωτερικό της υπορουτίνας.

Χρησιμοποιώντας παραμέτρους τα πράγματα γίνονται πολύ πιο απλά. Οι παράμετροιαναγράφονται και στις εντολές ορισμού των υπορουτινών (και όπως θα δούμε σε προσεχή παράγραφο και των συναρτήσεων), όπου ονομάζονται ***τυπικές μεταβλητές (formal variables)***. Μέσω των τυπικών μεταβλητών μεταβιβάζονται (εισάγονται) δεδομένα στις υπορουτίνες, αλλά και λαμβάνονται (εξάγονται) οι επεξεργασμένες τιμές. Οι μεταβλητές που τίθενται ως παράμετροι στην εντολή κλήσης μιας υπορουτίνας χαρακτηρίζονται ως ***πραγματικές μεταβλητές (actual variables)***. Η αντιστοίχιση των πραγματικών μεταβλητών και των τυπικών μεταβλητών γίνεται μία προς μία. Στην πρώτη πραγματική μεταβλητή αντιστοιχίζεται η πρώτη τυπική μεταβλητή, στη δεύτερη πραγματική μεταβλητή η δεύτερη τυπική μεταβλητή κ.ο.κ.

Παράδειγμα 21-3.

Η υπορουτίνα που ανταλλάσσει (swap) το περιεχόμενο δύο μεταβλητών ορίζεται ως:

```
Sub Swap (A As String, B As String)
    Dim Temp   As String  'ÐñïóùñéíÞ ìåôáâëçôÞ
    Temp = A
    A = B
    B = Temp
End Sub
Ç êëÞóç ôçò õðïñïõôßíáò ìÝóá áðü ôï ðñüãñáììá ãßíåôáé ùò:
  :
Swap Ê, Z
  :
```

Οι μεταβλητές A και B που αναγράφονται στον ορισμό της υπορουτίνας είναι τυπικές μεταβλητές, ενώ οι μεταβλητές K και Z, που αναγράφονται στην κλήση της υπορουτίνας είναι πραγματικές μεταβλητές.

Για τη μεταβίβαση και την ανταλλαγή δεδομένων από και προς τις υπορουτίνες και τις συναρτήσεις έχουν αναπτυχθεί κατά καιρούς διάφορες τεχνικές. Από αυτές έχουν επικρατήσει δύο. Η ***τεχνική περάσματος τιμής (pass by value)*** και η τεχνική ***περάσματος με αναφορά (pass by reference)***. Στην τεχνική περάσματος τιμής, η τιμή της σταθεράς, της μεταβλητής ή της παράστασης, που γράφεται στη θέση της αντίστοιχης παραμέτρου στο σημείο κλήσης της υπορουτίνας, μεταβιβάζεται στην υπορουτίνα μέσω της αντίστοιχης τυπικής μεταβλητής. Για να γίνει πέρασμα τιμής σε μια παράμετρο, πρέπει να υπάρχει η κωδική λέξη `ByVal` μπρος από την τυπική μεταβλητή στην εντολή δήλωσης της υπορουτίνας. Αλλαγές στην τιμή της τυπικής μεταβλητής, δεν προκαλούν αλλαγές στην τιμή της πραγματικής μεταβλητής, που χρησιμοποιήθηκε σαν παράμετρος. Αυτή η τεχνική χρησιμοποιείται μόνον για την εισαγωγή δεδομένων σε μια υπορουτίνα.

Το πλήρες όνομα της τεχνικής περάσματος με αναφορά είναι τεχνική περάσματος με αναφορά διεύθυνσης. Επίσης οι όροι pass by reference, ***pass by address*** και ***pass by location*** είναι

Η τεχνική περάσματος με αναφορά λειτουργεί, σα να περνά στην υπορουτίνα η μεταβλητή που αναγράφεται στην παράμετρο της εντολής κλήσης. Αυτό σημαίνει ότι αλλαγές στην τιμή της τυπικής μεταβλητής μέσα στην υπορουτίνα θα προκαλέσουν και αλλαγή της τιμής της αντίστοιχης πραγματικής μεταβλητής. Αυτή η τεχνική λοιπόν, μπορεί να χρησιμοποιηθεί και για την εξαγωγή τιμών από μια υπορουτίνα. Αν μπρος από μια τυπική μεταβλητή υπάρχει η κωδική λέξη `ByRef` ή δεν υπάρχει κωδική λέξη η μεταβλητή χρησιμοποιείται για πέρασμα με αναφορά.

Οι τυπικές μεταβλητές μπορεί να είναι οποιουδήποτε τύπου. Αν δε δηλωθεί ο τύπος τους θεωρείται ότι είναι του τύπου `Variant`. Ως τυπικές μεταβλητές μπορούμε να έχουμε και πίνακες. Για τους πίνακες όμως χρησιμοποιείται πάντα πέρασμα με αναφορά.

Παράδειγμα 21-4.

Ας δούμε πώς λειτουργεί ο μηχανισμός κλήσης της υπορουτίνας `Swap` στο παράδειγμα 21-3. Η ρουτίνα `Swap` καλείται με παραμέτρους τις μεταβλητές K και Æ. Ο τρόπος ορισμού της υπορουτίνας δηλώνει ότι θα γίνει πέρασμα με αναφορά, τόσο από την πρώτη όσο και από τη δεύτερη παράμετρο. Η μεταβλητή K περνά λοιπόν στην τυπική μεταβλητή A και η μεταβλητή Z περνά στην τυπική μεταβλητή A. Οι μεταβλητές K και A, όπως και οι

μεταβλητές Z και B θεωρούνται ταυτόσημες. Οι αλλαγές στις τυπικές μεταβλητές A και B, στο σώμα της υπορουτίνας, αντικατοπτρίζονται στις πραγματικές μεταβλητές K και Z μετά την εκτέλεση της υπορουτίνας. Επομένως, οι μεταβλητές K και Z μετά το πέρας της υπορουτίνας, θα έχουν ανταλλάξει αμοιβαία τιμές. Άρα το τμήμα προγράμματος:

```
  :
K = 10:     Z = 20
Print "Âêôýðùóç Ðñéí: ", K, Z
Swap Ê, Z
Print "Âêôýðùóç ÌåôÜ: ",K, Z
  :
```

θα εμφανίζει στη φόρμα το αποτέλεσμα:

```
Âêôýðùóç Ðñéí: 10            20
Âêôýðùóç ÌåôÜ: 20            10
```

Παράδειγμα 21-5.

Ένας καλύτερος ορισμός της υπορουτίνας, που υπολογίζει τον μέσο όρο, είναι ο:

Δήλωση πίνακα στις παραμέτρους

```
Sub Mean (Á() As Double, ByVal N, MeanValue As Double)
    Dim Sum    As Double
    Dim i      As Long
    Sum = 0      For i = 1 To N
       Sum = Sum + A(i)     'Ìåãéêü Üèñïéóìá
    Next I
    MeanValue = Sum / N
End Sub
```

Με τη μεταβλητή A γίνεται μεταβίβαση των τιμών του πίνακα στην υπορουτίνα υποχρεωτικά με πέρασμα αναφοράς. Το μέγεθος του πίνακα μεταβιβάζεται στην υπορουτίνα, μέσω της μεταβλητής N, με πέρασμα τιμής. Η μεταβλητή `MeanValue` χρησιμοποιείται για την εξαγωγή του μέσου όρου που υπολογίζεται μέσα στην υπορουτίνα και γι' αυτό δηλώνεται ως περάσματος αναφοράς. Παρατηρήστε ότι η δήλωση του πίνακα στην περιοχή των παραμέτρων της υπορουτίνας έγινε με τέτοιο τρόπο, ώστε να μπορεί να χρησιμοποιηθεί πίνακας με οποιοδήποτε μέγεθος. Το μέγεθος του πίνακα δε δηλώθηκε. Αμέσως μετά το όνομά του γράφηκαν παρενθέσεις κενού περιεχομένου.

Σε ένα πρόγραμμα με πολλές κλήσεις της υπορουτίνας `Mean` θα γράφαμε:

```
      :
'ÌÝóïò üñïò ôùí óôïé÷åßùí ôïõ Â
 Mean Â, Ubound(B), MeanB
      :
'ÌÝóïò üñïò ôùí óôïé÷åßùí ôïõ C
 Mean C, Ubound(C), MeanC
      :
'ÌÝóïò üñïò ôùí óôïé÷åßùí ôïõ D
 Mean D, Ubound(D), MeanD
      :
```

Παρατηρήστε ότι η δήλωση του πίνακα στην περιοχή των παραμέτρων, κατά την κλήση της υπορουτίνας, δεν είναι υποχρεωτικό να γίνει με κενού περιεχομένου παρενθέσεις.

Τα ονόματα των τυπικών μεταβλητών ακολουθούν τους κανόνες ονοματολογίας των μεταβλητών. Η εμβέλεια των τυπικών μεταβλητών είναι τοπική μέσα στην υπορουτίνα. Αυτό σημαίνει ότι τα ίδια ονόματα μεταβλητών μπορούν να χρησιμοποιηθούν και σε άλλες υπορουτίνες μέσα στο πρόγραμμα, χωρίς να υπάρχει πρόβλημα σύγχυσής τους.

Συναρτήσεις

Οι συναρτήσεις που ορίζει ο προγραμματιστής ονομάζονται οριζόμενες συναρτήσεις καιμοιάζουν με τις υπορουτίνες. Η διαφορά μεταξύ υπορουτινών και συναρτήσεων βρίσκεται στο ότι οι συναρτήσεις παίρνουν τιμή (όπως ακριβώς παίρνουν τιμή και οι ενσωματωμένες συναρτήσεις), ενώ οι υπορουτίνες δεν παίρνουν τιμή (όπως ακριβώς δεν παίρνουν τιμή και οι

```
Function όνομα_συνάρτησης(παράμετρος1 As τύπος1, ... ) As τύπος
    εντολή_1
    εντολή_2
      :
    όνομα_συνάρτησης = τελική_τιμή
End Sub
```

Όπως φαίνεται, ο ορισμός μιας συνάρτησης αρχίζει με την εντολή Function. Στην εντολή αυτή δίνεται όνομα στη συνάρτηση και αναφέρονται οι παράμετροί της και ο τύπος τους. Επίσης, 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, η οποία να εκχωρεί τιμή στο *όνομα_συνάρτησης*. Το τέλος της συνάρτησης δηλώνεται με την εντολή End Function. Η έξοδος από μια συνάρτηση δεν είναι υποχρεωτικό να γίνει από το τέλος της. Μπορεί να γίνει και ενδιάμεσα με εντολή Exit Function. Μια συνάρτηση μπορεί να περιέχει πολλές εντολές Exit Function αλλά μόνον μια εντολή End Function.

Η αντιστοίχιση πραγματικών μεταβλητών στα σημεία κλήσης της συνάρτησης, με τις τυπικές μεταβλητές είναι μία προς μία. Στην πρώτη πραγματική μεταβλητή αντιστοιχίζεται η πρώτη τυπική μεταβλητή, στη δεύτερη πραγματική μεταβλητή αντιστοιχίζεται η δεύτερη τυπική μεταβλητή κ.ο.κ.

Παράδειγμα 21-5.

Στο σκαρίφημα του προγράμματος:

```
Function CylinderArea(ByVal r As Double, ByVal h As Double) As Double
    Const ð = 3.141593
    CylinderArea = 2 * ð * r * (r + h)
End Function
          :
          :
Sub ...
          :
    TotalArea = Cube + CylinderArea(a, b)
          :
End Sub
```

φαίνεται ο τρόπος αντιστοίχισης των πραγματικών μεταβλητών a και b κατά την κλήση της συνάρτησης CylinderArea με τις τυπικές μεταβλητές r και h αντίστοιχα.

Παράδειγμα 21-6.

Ο μεγαλύτερος μεταξύ δύο αριθμός υπολογίζεται από τη συνάρτηση:

```
Function Max(ByVal a As Double, ByVal b As Double) As Double
    If a>b Then
        Max = a
    Else
        Max = b
    End If
End Function
```

Οι παράμετροι των συναρτήσεων μπορεί να είναι οποιουδήποτε τύπου. Αν δε δηλωθεί ο τύπος μιας παραμέτρου, θεωρείται ότι η παράμετρος είναι του τύπου Variant. Επίσης, ως τύπου Variant θεωρείται και η συνάρτηση στην περίπτωση που δε δηλωθεί ο τύπος της. Δεν είναι όμως δυνατόν, η τιμή μιας συνάρτησης να είναι πίνακας.

Παράδειγμα 21-7.

Η συνάρτηση που υπολογίζει τον μεγαλύτερο όρο από τα στοιχεία ενός πίνακα είναι η:

```
Function ArrayMax(A() As Double) As Double
    Dim i       As Long    'ÁðáñéèìçôÞò
    Dim First              'Ärßêôçò ðñþôïõ óôïé÷åßïõ
    Dim Last               'Ärßêôçò ôåëåõôáßïõ óôïé÷åßïõ
    Dim M                  'Ðñïóùñéíü ìÝãéóôï
        First = LBound(A):     Last = UBound(a)
        M = A(First)                    'Áóôù ìÝãéóôï ôï ðñþôï óôïé÷åßï
    For i = First To Last
        If A(i) > M Then M = A(i) 'Áí äåí éó÷ýåé ç ðñüèåóç áíôéêáôÜóôçóå
    Next i
    ArrayMax = M           'Äþóå ôéìÞ óôç óõíÜñôçóç
End Function
```

Με τη μεταβλητή A γίνεται μεταβίβαση των τιμών του πίνακα στη συνάρτηση υποχρεωτικά με πέρασμα αναφοράς.

Οι οριζόμενες συναρτήσεις καλούνται μέσα στο πρόγραμμα, όπως ακριβώς καλούνται και οι ενσωματωμένες συναρτήσεις. Γράφονται στο εσωτερικό παραστάσεων, όπου απαιτείται να γίνουν υπολογισμοί με τις τιμές τους.

Παραδείγματα 21-8.

α) Αν έχουμε ορίσει τη συνάρτηση Min, για τον υπολογισμό του μικρότερου μεταξύ δύο αριθμών, παρόμοια με τη συνάρτηση Max του παραδείγματος 21-6, η εντολη:

```
Length = Max(a, b) - Min(a, b)
```

υπολογίζει την απόσταση μεταξύ των a και b, και η εντολή:

```
Middle = (Max(a, b) - Min(a, b)) /2
```

υπολογίζει το μέσον της απόστασής τους.

β) Αν έχουμε ορίσει τη συνάρτηση MaxArray, του παραδείγματος 21-7, το τμήμα προγράμματος:

```
If MaxArray(A()) < 0 Then
    MsgBox "Ïëïé ïé áñéèìïß åßíáé áñíçôéêïß"
End if
```

μας πληροφορεί, αν όλοι οι αριθμοί ενός πίνακα είναι αρνητικοί.

Παράδειγμα 21-9.

Έστω ότι σε ένα πρόγραμμα ορίζεται μια συνάρτηση, που παίρνει την τιμή 1, αν το έτος που της δίνεται ως παράμετρος είναι δίσεκτο και την τιμή 0, αν το έτος δεν είναι δίσεκτο. Μια κωδικοποίηση αυτής της συνάρτησης είναι:

```
Function LeapYear(YearToCheck As Integer) As Integer
    If (YearToCheck mod 400 = 0) Then
        LeapYear = 1    'Äßóåêôï Ýôïò áöïý äéáéñåßôáé ìå 400
        Exit Function
    End If
    If (YearToCheck mod 100 = 0) Then
        LeapYear = 0    'Ìç äßóåêôï Ýôïò áöïý äéáéñåßôáé ìå 100
        Exit Function
    End If
    If (YearToCheck mod 4 = 0) Then
        LeapYear = 1    'Äßóåêôï Ýôïò áöïý äéáéñåßôáé ìå 4
        Exit Function
    End If
End Function
```

Η συνάρτηση μπορεί να χρησιμοποιηθεί για την εύκολη κωδικοποίηση σε πολλά σημεία του προγράμματος, όπως:

```
If LeapYear(Now()) Then Msgbox "Åôïò Äßóåêôï"
```

ή

```
DaysOfFebruary = 28 + LeapYear(Ôhis_year)
```

Εφαρμογή

Ταξινόμηση με απευθείας επιλογή

Στο μάθημα 19 αναλύσαμε τον αλγόριθμο της ταξινόμησης με απευθείας επιλογή. Σε αυτήν την παράγραφο θα δούμε πώς μπορούμε να συντάξουμε το ίδιο ακριβώς πρόγραμμα χρησιμοποιώντας υπορουτίνες και συναρτήσεις.

Σε κάθε αλγόριθμο, υπάρχουν τρία τμήματα, τα οποία μπορούμε να τα διακρίνουμε πολύ εύκολα. Στο πρώτο τμήμα γίνεται η είσοδος των δεδομένων, στο δεύτερο τμήμα γίνεται η επεξεργασία τους και στο τρίτο η παρουσίαση των αποτελεσμάτων. Για κάθε ένα από αυτά τα τμήματα μπορούμε να συντάξουμε και μια υπορουτίνα. Το κύριο τμήμα του προγράμματος θα έχει λοιπόν τη μορφή:

```
    :
Dim A()                    'ÄÞëùóç ôïõ ðßíáêá
Dim N        As Integer    'Ôï ðëÞèïò ôùí óôïé÷åßùí
'1. Åßóïäïò äåäïìÝíùí
InputData A(), N
'2. Åðåîåñãáóßá äåäïìÝíùí
SortBySelection A(), N
'3. ¸îïäïò áðïôåëåóìÜôùí
OutputData A(), N
    :
```

Η υπορουτίνα `InputData` δέχεται τα δεδομένα, η υπορουτίνα `SortBySelection` τα ταξινομεί και η υπορουτίνα `OutputData` τα εμφανίζει. Μια κωδικοποίηση αυτών των υπορουτινών θα μπορούσε να είναι η εξής:

```
Sub InputData(A(), N As Integer)
    Dim j     As Integer   'ÌåôñçôÞò âñü÷ïõ
   'ÄéÜâáóå ôï ðëÞèïò ôùí óôïé÷åßùí ôïõ ðßíáêá
    N = InputBox("ÐëÞèïò óôïé÷åßùí", "Åéóáãùãþ")
    ReDim A(N)              'Êáèïñéóìüò ìåãÝèïõò ôïõ ðßíáêá
   'Åéóáãùãþ óôïé÷åßùí
    For j = 1 To N
        A(j) = InputBox("Óôïé÷åßï " & j, "Åéóáãùãþ")
    Next j
End Sub

Sub SortBySelection(A(), N As Integer)
    Dim i, j    As Integer   'Ìåôñçôýò âñü÷ùí
    Dim Min                  'Ðñïóùñéíüò åëÜ÷éóôïò
    Dim k       As Integer   'ÈÝóç ðñïóùñéíïý åëÜ÷éóôïõ
'   Ðñáãìáôïðïßçóå Í-1 ðáñÜóìáôá
    For j = 1 To N - 1
       'Áýñåóç åëÜ÷éóôïõ óôï ðçãáßï ôìÞìá
        Min = A(j):   k = j     'Áóôù üôé ï åëÜ÷éóôïò åßíáé ï ðñþôïò
        For i = j + 1 To N
            If A(i) < Min Then 'Ái äåí éó÷ýåé ç õðüèåóç áíôéêåßùóçóå
                Min = A(i)
                k = i
            End If
        Next i
       'ÁíôéìåôÜèåóç á' óôïé÷åßïõ ôïõ ðçãáßïõ ìå åëÜ÷éóôï
       A(k) = A(j)          'Ôï á' ôïõ ðçãáßïõ óôç èÝóç ôïõ åëÜ÷éóôïõ
       A(j) = Min           'Ôï åëÜ÷éóôï óôç èÝóç ôïõ á' ôïõ ðçãáßïõ
    Next j
End Sub
```



```
Sub OutputData(A(), N As Integer)
    Dim j        As Integer   'Ìåôñçôþò âñü÷ïõ
    Dim MsgStr   As String    'Áíáêïßíùóç äéÜôáîçò
    For j = 1 To N
        MsgStr = MsgStr & A(j) & ", "
    Next j
    MsgBox MsgStr, , "ÄéáôåôáãìÝíá äåäïìÝíá"
End Sub
```

Η κωδικοποίηση των υπορουτινών αυτών θα μπορούσε να γίνει με διαφορετικό τρόπο, σε περίπτωση που θέλουμε να αποκτήσει το πρόγραμμα άλλες δυνατότητες. Για παράδειγμα, αλλάζοντας την υπορουτίνα `InputData`, θα μπορούσαμε να αντλήσουμε τα δεδομένα από ένα αρχείο ή από έναν πίνακα μιας βάσης δεδομένων (όπως θα δούμε στα επόμενα μαθήματα). Αλλάζοντας την υπορουτίνα `OutputData`, θα μπορούσαμε να εξάγουμε τα αποτελέσματα στον εκτυπωτή. Τέλος, αν τα δεδομένα ήταν πάρα πολλά και ο χρόνος εξαγωγής του αποτελέσματος μη ικανοποιητικός, θα μπορούσαμε να εφαρμόσουμε έναν άλλο ταχύτερο αλγόριθμο ταξινόμησης. Σε κάθε περίπτωση, οι αλλαγές που θα χρειαζόταν να γίνουν, θα αφορούσαν στην αντίστοιχη υπορουτίνα. Το κύριο σώμα του προγράμματος θα παρέμενε αναλλοίωτο.

Έχουμε αναφέρει ότι μια υπορουτίνα μπορεί να καλεί άλλες υπορουτίνες και αυτές με τη σειρά τους άλλες κ.ο.κ. Επίσης σε μια υπορουτίνα είναι δυνατόν να καλούνται και συναρτήσεις, ή μια συνάρτηση να καλεί υπορουτίνες κλπ. Η υπορουτίνα `SortBySelection` θα μπορούσε να αναλυθεί σε απλούστερες υπορουτίνες και να γραφεί:

```
Sub SortBySelection(A(), N As Integer)
    Dim i        As Integer   'Áñ÷Þ ðçãáßïõ ôìÞìáôïò
    Dim k        As Integer   'ÈÝóç åëÜ÷éóôïõ

   'Ðñáãìáôïðïßçóå Í-1 ðáñÜóìáôá
    For j = 1 To N - 1
   'Åýñåóç ôçò èÝóçò ôïõ åëÜ÷éóôïõ óôï ðçãáßï ôìÞìá
        k = Position_ofMin_InTarget(A(), j, N)
       'ÁíôéìåôÜèåóç á' óôïé÷åßïõ ôïõ ðçãáßïõ ìå åëÜ÷éóôï
        Swap A(k), A(j)
    Next j
End Sub
```

Η συνάρτηση `Position_ofMin_InTarget` βρίσκει τη θέση του ελαχίστου όρου μέσα στο πηγαίο τμήμα. Δέχεται ως παράμετρο όλον τον πίνακα, την αρχή του πηγαίου τμήματος και το μέγεθος του πίνακα.

```
Function Position_ofMin_InTarget(A(), Start As Integer, N As Integer) As Integer

    Dim p      As Integer    'ÈÝóç ðñïóùñéíïý åëÜ÷éóôïõ
    Dim i      As Integer    'Ìåôñçôþò âñü÷ïõ
    Dim Min                  'Ðñïóùñéíüò åëÜ÷éóôïò

    Min = A(Start):   p = Start 'Áóôù üôé ï åëÜ÷éóôïò åßíáé ï ðñþôïò
    For i = Start + 1 To N
        If A(i) < Min Then  'Ái äåí éó÷ýåé ç õðüèåóç áíôéêåßùóçóå
            Min = A(i)
            p = i
        End If
    Next i
    Position_ofMin_InTarget = p

End Function
```

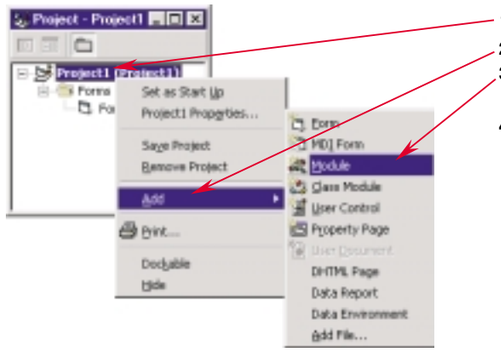
Η υπορουτίνα `Swap` αντιμεταθέτει τις τιμές δύο μεταβλητών. Στη συγκεκριμένη περίπτωση δέχεται ως παραμέτρους τα στοιχεία του πίνακα που θέλουμε να αντιμεταθούν.

```
Sub Swap(a, b)
    Dim temp
    temp = a
    a = b
    b = temp
End Sub
```

Η εμβέλεια των υπορουτινών και των συναρτήσεων

Οι υπορουτίνες και οι συναρτήσεις, που ορίζονται από τον προγραμματιστή, υπακούουν στους ίδιους κανόνες εμβέλειας με τις σταθερές και τις μεταβλητές. Το όνομά τους θεωρείται γνωστό και μπορεί να χρησιμοποιηθεί ανάλογα με την περιοχή του κώδικα που έχουν οριστεί και τον τρόπο σύνταξης της εντολής ορισμού τους.

Αν μια υπορουτίνα ή συνάρτηση έχει οριστεί σε μια βασική προγραμματιστική μονάδα και μπρος από την κωδική λέξη `Sub` ή `Function` υπάρχει η κωδική λέξη `Public` (ή δεν υπάρχει καμία κωδική λέξη) η υπορουτίνα ή η συνάρτηση θεωρείται καθολική και μπορεί να χρησιμοποιηθεί σε οποιοδήποτε σημείο του προγράμματος. Αντίθετα, αν μπρος από την κωδική λέξη `Sub` ή `Function` υπάρχει η κωδική λέξη `Private` η υπορουτίνα ή η συνάρτηση έχει εμβέλεια προγραμματιστής μονάδας και μόνον στον εσωτερικό της προγραμματιστικής μονάδας μπορεί να χρησιμοποιηθεί.



1. Κάνουμε κλικ στο επίπεδο **Project** του παραθύρου έργου.
2. Από το αναδυόμενο μενού επιλέγουμε **Add.**
3. Επιλέγουμε **Module** από το δεύτερο αναδυόμενο μενού που εμφανίζεται.
4. Στο παράθυρο έργου παρουσιάζεται η νέα προγραμματιστική μονάδα στο επίπεδο **Module.**

Εικόνα 21-1. *Διαδοχικά μενού για τη δημιουργία βασικής προγραμματιστικής μονάδας.*

Οι υπορουτίνες και οι διαδικασίες γράφονται στο παράθυρο κώδικα των βασικών προγραμματιστικών μονάδων, όπως ακριβώς γράφονται και στο παράθυρο κώδικα των προγραμματιστικών μονάδων φόρμας.

Ανακεφαλαίωση

Στη VB η κωδικοποίηση των υποδιαδικασιών γίνεται σε ανεξάρτητα τμήματα κώδικα, τις υπορουτίνες γενικού σκοπού και τις οριζόμενες]συναρτήσεις. Οι υπορουτίνες γενικού σκοπού είναι τμήματα κώδικα τα οποία δεν εκτελούνται όταν προκαλούνται κάποια συμβάντα, αλλά όταν γίνεται κλήση τους από άλλα σημεία του κώδικα. Αρχίζουν με την δηλωτική εντολή `Sub`, στην οποία δηλώνεται το όνομά τους οι παραμέτροι τους και οι τύποι των παραμέτρων τους. Αμέσως μετά γράφονται οι εντολές που κωδικοποιούν τη διαδικασία, που επιτελεί η υπορουτίνα. Το πέρας της υπορουτίνας γίνεται με την εντολή `End Sub`. Οι οριζόμενες συναρτήσεις μοιάζουν στον ορισμό με τις υπορουτίνες διαφέρουν όμως από αυτές στο ότι παίρνουν τιμή, όπως ακριβώς παίρνουν τιμή και οι ενσωματωμένες συναρτήσεις.

Οι παράμετροι που αναγράφονται στις εντολές ορισμού των υπορουτινών και των συναρτήσεων ονομάζονται τυπικές μεταβλητές. Μέσω των τυπικών μεταβλητών μεταβιβάζονται δεδομένα στις υπορουτίνες, αλλά και λαμβάνονται οι επεξεργασμένες τιμές. Οι μεταβλητές που τίθενται ως παράμετροι στην εντολή κλήσης μιας υπορουτίνας χαρακτηρίζονται ως πραγματικές μεταβλητές. Η μεταβίβαση και η ανταλλαγή δεδομένων από και προς τις υπορουτίνες και τις συναρτήσεις γίνεται με δύο κυρίως τεχνικές. Την τεχνική περάσματος τιμής και την τεχνική περάσματος με αναφορά.

Στην τεχνική περάσματος τιμής, η τιμή της σταθεράς, της μεταβλητής ή της παράστασης, που γράφεται στη θέση της αντίστοιχης παραμέτρου στο σημείο κλήσης της υπορουτίνας, μεταβιβάζεται μέσω της αντίστοιχης τυπικής μεταβλητής. Η τεχνική περάσματος με αναφορά λειτουργεί σα να περνά η μεταβλητή που αναγράφεται στην παράμετρο της εντολής κλήσης.

Εργαστηριακές Ασκήσεις

1. Να γραφτεί υπορουτίνα μηδενισμού των στοιχείων ενός πίνακα.
2. Να εκτελεστεί η εφαρμογή ταξινόμησης των στοιχείων ενός πίνακα.
3. Να γραφτεί συνάρτηση υπολογισμού του μέσου όρου ενός πίνακα.
4. Να γραφτεί υπορουτίνα που να δέχεται σε παράμετρο μια συμβολοσειρά που περιέχει ένα κείμενο και να επιστρέφει μέσω παραμέτρων το πλήθος των γραμμάτων, των λέξεων και των προτάσεων της συμβολοσειράς.
5. Να γραφεί υπορουτίνα που να δέχεται το ποσό μιας επιταγής (με πέντε το πολύ ψηφία) και να το τυπώνει ολογράφως. Π.χ. αν δεχτεί το ποσό 93642 να τυπώνει:

   ΕΝΕΝΗΝΤΑ ΤΡΕΙΣ ΧΙΛΙΑΔΕΣ ΕΞΑΚΟΣΙΕΣ ΣΑΡΑΝΤΑ ΔΥΟ ΔΡΑΧΜΕΣ
6. Να γραφτεί συνάρτηση με όνομα `FtoC`, η οποία να δέχεται ως τιμή τη θερμοκρασία σε βαθμούς Φαρενάιτ και να επιστρέφει τη θερμοκρασία σε βαθμούς Κελσίου.
7. Να γραφτεί συνάρτηση που να δέχεται ως παράμετρο έναν ακέραιο αριθμό και να επιστρέφει ως τιμή την παράσταση του αριθμού στο δυαδικό σύστημα.
8. Υπό τον φόβο της ιεράς εξέτασης, ο Λεονάρντο ντα Βίντσι έγραφε τα κείμενά του ανάποδα,δηλαδή από τα δεξιά προς τα αριστερά. Γράψτε συνάρτηση που να δέχεται μια πρόταση και να την κωδικοποιεί κατά ντα Βίντσι. Σε τι θα διαφέρει αυτή η συνάρτηση από μιαν άλλη που επαναφέρει την πρόταση στην αρχική της μορφή;
9. Να γραφτεί συνάρτηση που να δέχεται ως παραμέτρους έναν πίνακα, το μέγεθός του και ένα στοιχείο που θέλουμε να αναζητήσουμε μέσα στον πίνακα. Η συνάρτηση να πραγματοποιεί δυαδική αναζήτηση μέσα στον πίνακα και να επιστρέφει ως τιμή τη θέση που βρέθηκε το αναζητούμενο στοιχείο ή την τιμή -1 σε περίπτωση που το στοιχείο δεν υπάρχει μέσα στον πίνακα.
10. Να δημιουργηθεί υπορουτίνα που να πραγματοποιεί τη συγχώνευση (merge) των στοιχείων δύο πινάκων σε έναν τρίτο πίνακα.

Σημειώσεις:

Μάθημα 22
Φάκελοι & Αρχεία

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να διαχειρίζονται τους φακέλους του συστήμαος αρχείων μέσα από τα προγράμματά τους.
- Να συντάσσουν διαδικασίες αναζήτησης και επιλογής αρχείων.
- Να αντιγράφουν, να μετονομάζουν, να διαγράφουν και γενικά να διαχειρίζονται αρχεία μέσα από το πρόγραμμά τους.
- Να αναγνωρίζουν και να περιγράφουν τα είδη προσπέλασης των αρχείων.

Μέχρι τώρα έχουμε γράψει προγράμματα, τα οποία χρησιμοποιούν αποκλειστικά το πληκτρολόγιο για είσοδο δεδομένων, και την οθόνη για έξοδο των αποτελεσμάτων. Όμως, οι υπολογιστές χρησιμοποιούνται συνήθως για τη διαχείριση μεγάλου όγκου δεδομένων, τα οποία συγκεντρώνονται από διαφορετικά σημεία, αποστέλλονται στο σημείο επεξεργασίας "συσκευασμένα" σε αρχεία και διατηρούνται στα αρχεία για μεγάλο χρονικό διάστημα. Η είσοδος λοιπόν των δεδομένων στα προγράμματα και η έξοδος των αποτελεσμάτων γίνονται πολύ τακτικά από, ή σε αρχεία των αποθηκευτικών μονάδων του υπολογιστικού συστήματος (τους δίσκους, τις δισκέτες, τους οπτικούς δίσκους, τους δικτυακούς δίσκους κ.ά.).

Εφαρμογές που έχουν τέτοιες απαιτήσεις συναντάμε στις τράπεζες π.χ. στην ενημέρωση λογαριασμών, το ταμιευτήριο, τις επιταγές κ.ά., στις αεροπορικές εταιρείες π.χ. για κρατήσεις θέσεων, αναζήτηση αποσκευών κ.ά., στους οργανισμούς για δημιουργία τραπεζών πληροφοριών, στις επιχειρήσεις π.χ. για τη διαχείριση του λογιστηρίου, των πελατών, των αγορών, των πωλήσεων κ.ά.

Διαχείριση φακέλων

Τα αρχεία που δημιουργούνται με κάθε τρόπο στον υπολογιστή αποθηκεύονται στους δίσκους του υπολογιστή και οργανώνονται σε ***φακέλους (folders)***. Η δημιουργία, η μετονομασία, η διαγραφή των φακέλων και γενικά κάθε μορφής διαχείριση, γίνεται από το βοηθητικό πρόγραμμα του λειτουργικού συστήματος (utility), τον Explorer (Εξερευνητή). Πολλές φορές όμως, θέλουμε να διαχειριστούμε ένα φάκελο μέσα από ένα πρόγραμμα. Η VB μας προσφέρει και αυτή την ευκολία με τις εντολές και τις συναρτήσεις που δίδονται στον πίνακα:

Παλαιότερα οι φάκελοι ονομάζονταν ***κατάλογοι (directories)***

Κωδ. λέξη	Τύπος	Λειτουργία
`ChDrive`	Εντολή	Αλλαγή τρέχοντος δίσκου
`ChDir`	Εντολή	Αλλαγή τρέχοντος φακέλου
`CurDir`	Συνάρτηση	Αναφέρει τον τρέχοντα φάκελο
`MkDir`	Εντολή	Δημιουργία φακέλου
`RmDir`	Εντολή	Διαγραφή φακέλου

Πίνακας 22-1. *Εντολές και συναρτήσεις για φακέλους*

Στον Explorer, αλλά και σε άλλα προγράμματα, ο δίσκος στον οποίο γίνονται οι εργασίες, ονομάζεται ***τρέχων δίσκος (current drive)*** ή ***δίσκος αναφοράς***. Επίσης, ο φάκελος στον οποίο γίνονται οι εργασίες, ονομάζεται ***τρέχων φάκελος (current folder)*** ή ***φάκελος αναφοράς***. Η αλλαγή του τρέχοντος δίσκου (Change Drive) γίνεται με την εντολή:

Αλλαγή τρέχοντος δίσκου

`ChDrive` *όνομα_δίσκου*

Η παράμετρος *όνομα_δίσκου* πρέπει να έχει αλφαριθμητική τιμή και καθορίζει τον δίσκο, που θα θεωρείται στο εξής ως τρέχων δίσκος. Από όλους τους χαρακτήρες της αλφαριθμητικής τιμής μόνο ο πρώτος χαρακτήρας λαμβάνεται υπ' όψιν.

Η αλλαγή του τρέχοντος φακέλου (Change Directrory) γίνεται με την εντολή:

Αλλαγή τρέχοντος φακέλου

`ChDir` *όνομα_φακέλου*

Η παράμετρος όνομα_φακέλου είναι αλφαριθμητική σταθερά, μεταβλητή ή παράσταση, η οποία προσδιορίζει τον φάκελο, που θα θεωρείται στο εξής ως τρέχων φάκελος. Σε κάθε

δίσκο του υπολογιστικού συστήματος ένας μόνο φάκελος θεωρείται ως τρέχων. Στο *όνομα_φακέλου* μπορεί να περιλαμβάνεται και το όνομα του δίσκου. Αν δεν περιλαμβάνεται το όνομα του δίσκου αλλάζει ο τρέχων φάκελος του τρέχοντα δίσκου.

Ο τρέχων φάκελος σημειώνεται στον Explorer με έναν ανοιχτό φάκελο και τονισμένο πλαίσιο γύρω από το όνομά του.

Παραδείγματα 22-1.

α) Η εντολή:

```
ChDrive "F"
```

καθορίζει ως τρέχοντα δίσκο, το δίσκο F (στα υπολογιστικά συστήματα ο δίσκος F είναι συνήθως δίσκος δικτύου).

β) Η εντολή:

```
ChDir "C:\My Documents\Faxes"
```

γ) καθορίζει ως τρέχοντα φάκελο στο δίσκο C, το φάκελο `"\My Documents\Faxes"`.

Η εντολή:

```
ChDir "\Physics\Optics"
```

καθορίζει ως τρέχοντα φάκελο, το φάκελο `"\Physics\Optics"`, στον τρέχοντα δίσκο.

Παραδείγματα 22-2.

α) Έστω οι εντολές:

```
NewPath = "C:\My Documents\Letters"
ChDrive NewPath
ChDir NewPath
```

Η δεύτερη εντολή καθορίζει ως τρέχοντα δίσκο το δίσκο C, αφού μόνο το πρώτο γράμμα της τιμής της μεταβλητής `NewPath` λαμβάνεται υπ' όψιν. Η τρίτη εντολή καθορίζει το φάκελο `"\My Documents\Letters"` ως τρέχοντα φάκελο του δίσκου C.

β) Αν ο τρέχων δίσκος είναι ο C, η εντολή:

```
ChDir "D:\BOOKS\VB"
```

αλλάζει τον τρέχοντα φάκελο του δίσκου D, αλλά ο δίσκος C εξακολουθεί να παραμένει ο τρέχων δίσκος, με δικό του τρέχοντα κατάλογο. Η εντολή `ChDir` αλλάζει τον τρέχοντα φάκελο αλλά όχι τον τρέχοντα δίσκο.

Για να διαπιστώσουμε ποιος είναι ο τρέχων φάκελος (Current Directory) σε ένα δίσκο χρησιμοποιούμε την εντολή:

`CurDir`(*δίσκος*)

Το όνομα του φακέλου στον οποίο βρίσκεται το πρόγραμμα που εκτελείται δίνεται από την ιδιότητα **`Path`** του αντικειμένου App. Π.χ. `App.Path`

Η παράμετρος δίσκος είναι προαιρετική. Σε περίπτωση που δεν αναγράφεται, ή σε περίπτωση που η τιμή της είναι η μηδενική συμβολοσειρά (""), η συνάρτηση παίρνει για τιμή τον τρέχοντα φάκελο του τρέχοντα δίσκου.

Παραδείγματα 22-3.

Έστω, ότι:

- ο τρέχων δίσκος είναι ο `C`.
- ο τρέχων κατάλογος στον δίσκο C είναι ο `"C:\My Documents\Faxes"`
- ο τρέχων κατάλογος στον δίσκο D είναι ο `"D:\BOOKS\VB"`

η εντολή:

```
DocsPath = CurDir
```

δίνει στη μεταβλητή `DocsPath` την τιμή `"C:\My Documents\Faxes"` και η εντολή:

```
ÂïïêsPath = CurDir("D")
```

δίνει στη μεταβλητή `BooksPath` την τιμή `"D:\Books\VB"`.

Παραδείγματα 22-4.

Σε αναλογία με τη συνάρτηση `CurDir` ορίζουμε τη συνάρτηση `CurDrive`, που μας επιστρέφει τον τρέχοντα δίσκο:

```
Function CurDrive() As String
    CurDrive = Left(CurDir, 1)
End Function
```

Δημιουργία φακέλου

Για να δημιουργήσουμε έναν φάκελο χρησιμοποιούμε την εντολή `MkDir`, η οποία έχει τη γενική μορφή:

`MkDir` *όνομα_φακέλου*

Η παράμετρος *όνομα_φακέλου* είναι αλφαριθμητική σταθερά, μεταβλητή ή παράσταση, της οποίας η τιμή καθορίζει το όνομα του νέου φακέλου. Στην τιμή της παραμέτρου μπορεί να περιέχεται το όνομα του δίσκου και οι ανώτεροι ιεραρχικά φάκελοι του νέου φακέλου. Αν δεν καθορίζεται ο δίσκος ή ο ανώτερος ιεραρχικά φάκελος, στον οποίο θα δημιουργηθεί ο νέος φάκελος, ο νέος φάκελος δημιουργείται στον τρέχοντα δίσκο και φάκελο.

Παραδείγματα 22-5.

α) Η εντολή:

```
MkDir "C:\My Documents\Faxes\Inbox"
```

δημιουργεί το φάκελο `Inbox` μέσα στον φάκελο `"C:\My Documents\Faxes"`.

β) Η εντολή:

```
MkDir "Outbox"
```

Διαγραφή φακέλου

Για να διαγράψουμε ένα φάκελο, χρησιμοποιούμε την εντολή `RmDir`, η οποία έχει τη γενική μορφή:

`RmDir` *όνομα_φακέλου*

Η παράμετρος *όνομα_φακέλου* είναι αλφαριθμητική σταθερά, μεταβλητή ή παράσταση, της οποίας η τιμή καθορίζει το όνομα του φακέλου που θα διαγραφεί. Στην τιμή της παραμέτρου μπορεί να περιέχεται το όνομα του δίσκου και οι ανώτεροι ιεραρχικά φάκελοι του προς διαγραφή φακέλου. Αν δεν καθορίζεται ο φάκελος ή ο δίσκος στον οποίο ανήκει ο φάκελος, διαγράφεται ο φάκελος του τρέχοντα δίσκου και φακέλου.

Για να γίνει η διαγραφή του φακέλου, πρέπει ο φάκελος να μην περιέχει αρχεία, διαφορετικά γίνεται διακοπή της εκτέλεσης του προγράμματος και εμφάνιση του μηνύματος "**Path/File access error**".

Διαχείριση αρχείων

Δεν είναι λίγες οι φορές που μέσα σε ένα πρόγραμμα θέλουμε να μετονομάσουμε, να αντιγράψουμε, να διαγράψουμε από το δίσκο ένα αρχείο. Η VB μας προσφέρει και αυτή την ευκολία.

Κωδ. λέξη	Τύπος	Λειτουργία
`Name`	Εντολή	Μετονομασία ή μετακίνηση αρχείου
`FileCopy`	Εντολή	Αντιγραφή αρχείου
`Kill`	Εντολή	Διαγραφή αρχείου
`FileDateTime`	Συνάρτηση	Ημερομηνία και ώρα αρχείου ή φακέλου
`FileLen`	Συνάρτηση	Μέγεθος αρχείου
`GetAttr`	Συνάρτηση	Αναφορά των χαρακτηριστικών αρχείου ή φακέλου
`SetAttr`	Εντολή	Απόδοση χαρακτηριστικών σε αρχείο ή σε φάκελο

Πίνακας 22-2. *Εντολές και συναρτήσεις για αρχεία*

Για να αλλάξουμε όνομα σε ένα αρχείο χρησιμοποιούμε την εντολή `Name`, η οποία έχει τη γενική μορφή:

Μετονομασία αρχείου και μεταφορά αρχείου

`Name` *όνομα_αρχείου_*πριν `As` *όνομα_αρχείου_μετά*

όπου το όνομα_αρχείου_πριν το παλαιό όνομα του αρχείου και *όνομα_αρχείου_μετά* το όνομα που θα αποκτήσει το αρχείο. Τόσο το *όνομα_αρχείου_πριν* όσο και το *όνομα_αρχείου_μετά* μπορούν να είναι αλφαριθμητικές σταθερές, μεταβλητές ή παραστάσεις. Ας σημειωθεί, ότι μόνο ένα αρχείο τη φορά μπορεί να μετονομαστεί. Επίσης, αν το *όνομα_αρχείου_μετά* αναφέρεται σε φάκελο διαφορετικό του φακέλου του αρχικού ονόματος, γίνεται μεταφορά του αρχείου στον νέο φάκελο.

Το όνομα ενός αρχείου δεν μπορεί να είναι μεγαλύτερο από 255 χαρακτήρες. Επίσης, δεν επιτρέπεται να περιέχει τα ειδικά σύμβολα `\ / : * ? "`

Παραδείγματα 22-6.

α) Η εντολή:

```
Name "C:\My Documents\Euro.Doc" As "Euro_Old.Doc"
```

μετονομάζει το αρχείο `"Euro.Doc"` του φακέλου `"C:\My Documents"` και του δίνει το όνομα `"Euro_Old.Doc"`.

β) Η εντολή:

```
Name "C:\My Documents\Euro.Doc" As "C:\BackUp\Euro.Doc"
```

μεταφέρει το αρχείο `"Euro.Doc"` του φακέλου `"C:\My Documents"` στο φάκελο `"C:\BackUp"`.

Αντιγραφή αρχείου

Για να αντιγράψουμε το περιεχόμενο ενός αρχείου σε ένα άλλο αρχείο, χρησιμοποιούμε την εντολή `FileCopy`, η οποία έχει τη γενική μορφή:

`FileCopy` *όνομα_αρχείου_πρωτοτύπου όνομα_αρχείου_αντιγράφου*

Απ' ότι φαίνεται, η πρώτη παράμετρος δηλώνει το όνομα του αρχείου πρωτοτύπου, ενώ η δεύτερη παράμετρος καθορίζει το όνομα του αρχείου αντιγράφου.

Παραδείγματα 22-7.

Οι εντολές:

```
SourceFile = "C:\My Documents\MyApplication.exe"
DestinationFile = "NewApplication.exe"
FileCopy SourceFile, DestinationFile
```

αντιγράφουν το αρχείο `"MyApplication.exe"` του φακέλου `"C:\My Documents"` στο αρχείο `"NewApplication.exe"` στον ίδιο φάκελο.

Διαγραφή αρχείου

Για να διαγράψουμε ένα αρχείο χρησιμοποιούμε την εντολή `Kill`, η οποία έχει τη γενική μορφή:

`Kill` *όνομα_αρχείου*

όπου το *όνομα_αρχείου* αλφαριθμητική σταθερά, μεταβλητή ή παράσταση που προσδιορίζει το αρχείο ή τα αρχεία που θα διαγραφούν. Μέσα στο όνομα αρχείου είναι δυνατόν να χρησιμοποιηθούν οι χαρακτήρες μπαλαντέρ, αστερίσκος (*) και ερωτηματικό (?). Ο αστερίσκος χρησιμοποιείται σαν υποκατάστατο οποιουδήποτε πλήθους χαρακτήρων, ενώ το ερωτηματικό, για έναν μόνο χαρακτήρα.

Παραδείγματα 22-7.

α) Η διαγραφή των αρχείων με επέκταση `"tmp"` από τον τρέχοντα κατάλογο, γίνεται με την εντολή:

```
Kill "*.tmp"
```

β) Η διαγραφή των αρχείων των οποίων το όνομά αρχίζει από `"Lib"` και έχει επέκταση που αρχίζει από `"b"` από τον τρέχοντα κατάλογο, γίνεται με την εντολή:

```
Kill "Lib*.b*"
```


γ) Η διαγραφή όλων των αρχείων του καταλόγου `"C:\Temp"` γίνεται με την εντολή:

```
Kill "C:\Temp\*.*"
```

δ) Η προσπάθεια διαγραφής αρχείου, το οποίο δεν υπάρχει, προκαλεί την εμφάνιση διαγνωστικού μηνύματος λάθους **"File not found"** και τη διακοπή του προγράμματος. Για να αποφευχθεί η διακοπή του προγράμματος, αν στη μεταβλητή `FileToKill` υπάρχει το προς διαγραφή αρχείο, γράφουμε:

```
If Dir(FileToKill)<> "" then Kill FileToKill
```

Οι συναρτήσεις `FileDateTime`, `FileLen`, `GetAttr` δίνουν πληροφορίες σχετικές με τις παραμέτρους που χαρακτηρίζουν τα αρχεία. Συγκεκριμένα, η συνάρτηση `FileDateTime` δέχεται ως παράμετρο το όνομα ενός αρχείου και παίρνει ως τιμή την ημερομηνία που το αρχείο δημιουργήθηκε ή ενημερώθηκε τελευταία φορά. Η συνάρτηση `FileLen` δέχεται ως παράμετρο το όνομα ενός αρχείου και παίρνει ως τιμή το μέγεθος του αρχείου σε byte. Τέλος, η συνάρτηση `GetAttr` δέχεται ως παράμετρο το όνομα ενός αρχείου ή ενός φακέλου και παίρνει ως τιμή έναν αριθμό, ο οποίος δίνει πληροφορίες για το αρχείο. Ο αριθμός αυτός προκύπτει ως συνδυασμός των σταθερών:

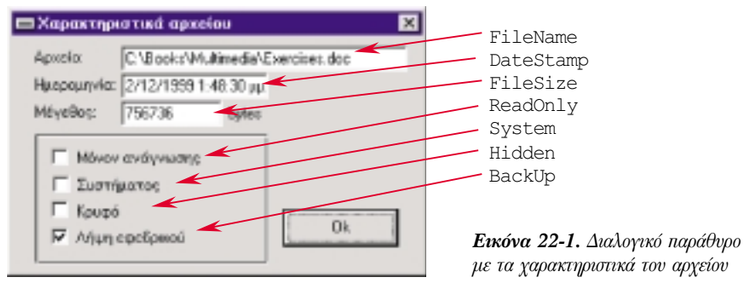
Σταθερά	Τιμή	Περιγραφή
`vbNormal`	0	Κανονικό αρχείο
`vbReadOnly`	1	Αρχείο μόνο για ανάγνωση
`vbHidden`	2	Αόρατο αρχείο (κρυφό)
`vbSystem`	4	Αρχείο συστήματος
`vbDirectory`	16	Φάκελος
`vbArchive`	32	Το αρχείο έχει υποστεί αλλαγές μετά τη λήψη του τελευταίου εφεδρικού αντιγράφου

Οι τιμές των σταθερών αθροίζονται, όπως γίνεται και με τις τιμές των σταθερών της συνάρτησης `MsgBox`.

Άσκηση 22-1.

Να δημιουργηθεί μια φόρμα που θα χρησιμοποιηθεί σαν πλαίσιο διαλόγου με σκοπό να εμφανίζει τις παραμέτρους που χαρακτηρίζουν ένα αρχείο, του οποίου το όνομα βρίσκεται αποθηκευμένο στην καθολική μεταβλητή `FileToTest`.

1. Δίνουμε στην ιδιότητα **`Border`** της φόρμας την τιμή **Fixed Dialog** (τα πλαίσια διαλόγου δεν πρέπει να αλλάζουν μέγεθος). Ας σημειωθεί, ότι δίνοντας αυτήν την τιμή στην ιδιότητα **`Border`** δε λαμβάνονται υπ' όψιν οι τιμές των ιδιοτήτων **`MaxButton`** και **`MinButton`** και το παράθυρο θα εμφανιστεί χωρίς πλήκτρα μεγιστοποίησης και ελαχιστοποίησης.
2. Τοποθετούμε πάνω στη φόρμα τα αντικείμενα ελέγχου.



Εικόνα 22-1. *Διαλογικό παράθυρο με τα χαρακτηριστικά του αρχείου*

3. Γράφουμε τον κώδικα:

```
Private Sub Form_Load()
    FileName = FileToTest
    DateStamp = FileDateTime(FileToTest)
    FileSize = FileLen(FileToTest)
    ReadOnly = Sgn(GetAttr(FileToTest) And vbReadOnly)
    System = Sgn(GetAttr(FileToTest) And vbSystem)
    Hidden = Sgn(GetAttr(FileToTest) And vbHidden)
    BackUp = Sgn(GetAttr(FileToTest) And vbArchive)
End Sub
```

Με τον τελεστή `And` απομονώνεται μια τιμή και με τη συνάρτηση `Sgn` η τιμή μετατρέπεται σε 0 ή 1.

Η εντολή `SetAttr` είναι συζυγής της συνάρτησης `GetAttr`. Δίνει τιμές στα χαρακτηριστικά (attributes) ενός αρχείου ή φακέλου. Η σύνταξή της είναι:

`SetAttr` *όνομα, χαρακτηριστικά*

Η παράμετρος όνομα μπορεί να είναι όνομα αρχείου ή φακέλου και η παράμετρος χαρακτηριστικά παίρνει τις ίδιες τιμές με αυτές που δίνει η συνάρτηση `GetAttr`.

Αναζήτηση και επιλογή αρχείου

Η VB διαθέτει τρεις τρόπους για αναζήτηση και επιλογή αρχείων. Συγκεκριμένα, ο προγραμματιστής κατά περίπτωση μπορεί να χρησιμοποιήσει:

- Τη συνάρτηση `Dir`.
- Τα αντικείμενα ελέγχου πτυσσόμενη λίστα δίσκων, λίστα φακέλων και λίστα αρχείων.
- Τα προτυποιημένα κοινά διαλογικά παράθυρα.

Την αναζήτηση και επιλογή αρχείων από προτυποποιημένα κοινά διαλογικά παράθυρα τη μελετήσαμε στο παράδειγμα 13-4 και στην άσκηση 17-3.

Η συνάρτηση Dir

Η συνάρτηση Dir επιστρέφει το όνομα του αρχείου ή του φακέλου που ικανοποιεί τα κριτήρια που τίθενται μέσω των παραμέτρων της. Αν δε βρεθεί κανένα αρχείο ή φάκελος η τιμή της συνάρτησης είναι η μηδενική συμβολοσειρά (`""`). Η γενική μορφή της συνάρτησης είναι:

`Dir` (*φίλτρο, χαρακτηριστικά*)

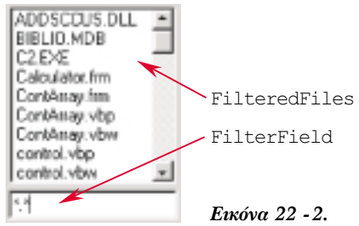
όπου το φίλτρο αλφαριθμητική σταθερά, μεταβλητή ή παράσταση που προσδιορίζει το αρχείο ή το φάκελο που αναζητείται. Μέσα στο φίλτρο είναι δυνατόν να χρησιμοποιηθούν οι χαρακτήρες μπαλαντέρ, αστερίσκος (*) και ερωτηματικό (?). Η τιμή της παραμέτρου χαρακτηριστικά συντίθεται από το άθροισμα των εσωτερικών σταθερών που περιγράψαμε στη συνάρτηση `GetAttr`.

Η χρήση και των δύο παραμέτρων είναι προαιρετική. Όμως, την πρώτη φορά που καλείται η συνάρτηση `Dir`, πρέπει να χρησιμοποιηθεί οπωσδήποτε η παράμετρος φίλτρο. Τότε η συνάρτηση επιστρέφει το όνομα του πρώτου αρχείου, που ικανοποιεί το κριτήριο που θέτει το *φίλτρο*. Αν θέλουμε να λάβουμε και τα υπόλοιπα αρχεία καλούμε τη συνάρτηση `Dir` χωρίς παραμέτρους. Ας σημειωθεί, ότι η σειρά με την οποία λαμβάνονται τα ονόματα των αρχείων είναι τυχαία. Αν θέλουμε να ταξινομήσουμε τα αρχεία με κάποιον τρόπο, πρέπει είτε τα αποθηκεύσουμε σε πίνακα και μετά να τα ταξινομήσουμε, είτε να εκμεταλλευτούμε την αυτόματη ταξινόμηση που προσφέρει το αντικείμενο ελέγχου λίστα.

Άσκηση 22-2.

Σε μια φόρμα να τοποθετηθεί λίστα με όνομα `FilteredFiles` και πλαίσιο κειμένου με όνομα `FilterField`. Στη λίστα να φαίνονται ταξινομημένα τα ονόματα των αρχείων που ικανοποιούν το φίλτρο που γράφεται στο πλαίσιο κειμένου `FilterField`.

1. Τοποθετούμε στη φόρμα τα αντικείμενα ελέγχου.
2. Για αυτόματη ταξινόμηση δίνουμε στην ιδιότητα **`Sorted`** της λίστας `FilteredFiles` την τιμή `True`.
3. Γράφουμε τον κώδικα:



Εικόνα 22 -2.

```
Private Sub FilterField_Change()
    Dim FileName     As String
   'Êáèüñéóå ôç ëßóôá
    FilteredFiles.Clear
   'Ðñþôç êëþóç ôçò Dir. Êáèüñéóå ôï ößëôñï
    FileName = Dir(FilterField)
   'Ïóï õðÜñ÷ïõí áñ÷åßá ðïõ éêáíïðïéïýí ôï êñéôÞñéï ôïõ ößëôñïõ
    Do While FileName <> ""
        FilteredFiles.AddItem FileName   'Ðñüóèåóå óôç ëßóôá
        FileName = Dir()                 'ÐÜñå ôï åðüìåíï áñ÷åßï
    Loop
End Sub
```


Πτυσσόμενη λίστα δίσκων, λίστα φακέλων και λίστα αρχείων

Τα αντικείμενα ελέγχου, πτυσσόμενη λίστα δίσκων, λίστα φακέλων και λίστα αρχείων μπορούμε να τα τοποθετήσουμε πάνω σε φόρμες και γράφοντας πολύ λίγο κώδικα, που εκμεταλλεύεται τις ιδιότητές τους, προσφέρουμε στους χρήστες τη δυνατότητα να κινούνται με μεγάλη ευκολία μέσα στην ιεραρχία των φακέλων των δίσκων.

Η πτυσσόμενη λίστα δίσκων εμφανίζει μια λίστα με τις διαθέσιμες μονάδες δίσκων. Την πληροφορία αυτή την αντλεί από το λειτουργικό σύστημα του υπολογιστή. Η ιδιότητα **`Drive`** του αντικειμένου λαμβάνει ως τιμή την επιλεγμένη, μέσα από τη λίστα, μονάδα δίσκου. Η αρχική τιμή της ιδιότητας **`Drive`** είναι ο τρέχων δίσκος. Αλλαγή τιμής σε αυτήν την ιδιότητα δε συνεπάγεται και αλλαγή του τρέχοντα δίσκου.

Η λίστα φακέλων εμφανίζει μια λίστα με φακέλους σε μορφή ιεραρχίας. Η ιδιότητα **`Path`** του αντικειμένου λαμβάνει ως τιμή τον επιλεγμένο, μέσα από τη λίστα, φάκελο. Στη συμβολοσειρά που αποτελεί την τιμή της ιδιότητας **`Path`** περιλαμβάνεται το όνομα του δίσκου και, στη σειρά, τα ονόματα όλων των ανώτερων ιεραρχικά φακέλων. Η αρχική τιμή της ιδιότητας **`Path`** είναι ο τρέχων φάκελος στον τρέχοντα δίσκο. Αλλαγή τιμής σε αυτήν την ιδιότητα δε συνεπάγεται και αλλαγή του τρέχοντα φακέλου.

Η λίστα αρχείων εμφανίζει μια λίστα με αρχεία. Η τιμή της ιδιότητας **`Pattern`** αποτελεί φίλτρο και περιορίζει τη λίστα στα αρχεία που το ικανοποιούν. Επίσης, υπάρχουν και ιδιότητες, που έχουν σχέση με τα χαρακτηριστικά των αρχείων, όπως η ιδιότητα **`ReadOnly`**, **`System`** κ.ά. Στις ιδιότητες αυτές μπορούν να δοθούν οι τιμές `True`, `False` ανάλογα με το αν θέλουμε να θέσουμε πρόσθετα κριτήρια στην επιλογή των αρχείων που θα εμφανιστούν. Αρχικά, στη λίστα παρουσιάζονται όλα τα αρχεία του τρέχοντος δίσκου και φακέλου. Η τιμή της ιδιότητας **`Filename`** είναι το όνομα του αρχείου που έχει επιλεγεί μέσα από τη λίστα.

Άσκηση 22-3.

Να δημιουργηθεί φόρμα με πτυσσόμενη λίστα δίσκων με όνομα `DriveList`, λίστα φακέλων με όνομα `DirList` και λίστα αρχείων με όνομα `FileList`. Η επιλογή δίσκου στην πτυσσόμενη λίστα δίσκων να προκαλεί αυτόματη προσαρμογή του περιεχομένου της λίστας φακέλων ώστε να παρουσιάζονται οι φάκελοι του δίσκου που έχει επιλεγεί και η επιλογή του φακέλου στη λίστα φακέλων να προκαλεί αυτόματη προσαρμογή του περιεχομένου της λίστας αρχείων, ώστε να παρουσιάζονται τα αρχεία του φακέλου που έχει επιλεγεί.

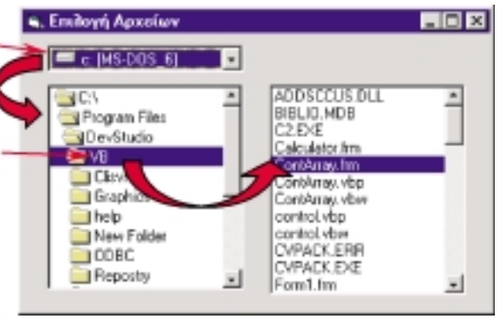
Αφού τοποθετήσουμε τα αντικείμενα ελέγχου γράφουμε τον κώδικα:

```
Private Sub DriveList_Change()
    DirList.Path = DriveList.Drive
End Sub

Private Sub DirList_Change()
    FileList.Path = DirList.Path
End Sub
```

Το συμβάν **`Change`** της πτυσσόμενης λίστας `DriveList` ενημερώνει τη λίστα `DirList`.

Η ενημέρωση της λίστας `DirList` προκαλεί το συμβάν **`Change`** της λίστας και ενημέρωση της λίστας `FileList`.



Εικόνα 22-3. *Ο κώδικας που συνοδεύει τη φόρμα προκαλεί αυτόματη διαδοχή των συμβάντων* ***`Change`*** *.*

Η προσπέλαση στα αρχεία

Το σύνολο των ενεργειών που κάνει ο υπολογιστής για να βρεί, να διαβάσει, ή να γράψει δεδομένα σε ένα αρχείο ονομάζεται ***προσπέλαση (access)***. Η προσπέλαση μπορεί να γίνει με δύο τρόπους ***σειριακά (sequential)*** και ***τυχαία (random)***. Όταν γράφουμε σε αρχείο με σειριακό τρόπο τα δεδομένα αποθηκεύονται το ένα μετά το άλλο, ακριβώς με τη σειρά με την οποία γίνεται η εισαγωγή τους. Η σειριακή γραφή είναι ίδια με τη γραφή στις σελίδες ενός τετραδίου. Συμπληρώνουμε πρώτα τη μια γραμμή, μετά την επομένη κ.ο.κ. Επίσης, κατά την σειριακή ανάγνωση, τα δεδομένα διαβάζονται ακριβώς με την ίδια σειρά που γράφτηκαν. Αν θέλουμε να διαβάσουμε μια συγκεκριμένη πληροφορία και δεν είναι η σειρά της να διαβαστεί, είμαστε υποχρεωμένοι να διαβάσουμε όλες τις προηγούμενες πληροφορίες. Επίσης, αν κατά την ανάγνωση, χρειαστεί να επιστρέψουμε σε πληροφορίες, που έχουμε ήδη προσπεράσει, πρέπει να γυρίσουμε πάλι στην αρχή του αρχείου και να διαβάσουμε όλες τις πληροφορίες που προηγούνται. Η σειριακή προσπέλαση γίνεται συνήθως σε αρχεία που περιέχουν κείμενα ή σε αρχεία που περιέχουν μεταβλητού μεγέθους ομάδες πληροφοριών, ή σε αρχεία τα οποία επεξεργαζόμαστε πάντα αρχίζοντας από την αρχή τους.

Αντί του όρου τυχαία προασπέλαση χρησιμοποιείται και ο όρος ***άμεση προσπέλαση (direct access)***.

Πολλές φορές στις εφαρμογές μας, δε θέλουμε να διαβάζουμε όλα τα στοιχεία από την αρχή του αρχείου, έως το σημείο που περιέχει τις ενδιαφέρουσες πληροφορίες, αλλά μόνον τις πληροφορίες που μας ενδιαφέρουν. Η ανάγνωση στην τυχαία προσπέλαση μοιάζει με την ανάγνωση στοιχείων από έναν τηλεφωνικό κατάλογο. Δε διαβάζουμε όλα τα ονόματα, ξεκινώντας πάντα από το άλφα, αλλά πάμε απευθείας στο γράμμα με το οποίο αρχίζει το όνομα του ατόμου που ψάχνουμε. Ακόμη, κατά την καταχώριση πληροφοριών, γράφουμε σε ένα συγκεκριμένο σημείο, χωρίς να απαιτείται να μεταβάλλουμε ή να επαναγράψουμε όλη την προγενέστερη πληροφορία. Στην τυχαία προσπέλαση, η εκλογή της περιοχής που θα διαβαστεί ή θα γραφτεί, γίνεται είτε με βάση την πληροφορία που μας δίνουν κάποιοι δείκτες (π.χ. γράμματα του ευρετηρίου) είτε με βάση κάποια μέθοδο (πχ. μέθοδος δυαδικής αναζήτησης). Η τυχαία προσπέλαση προτιμάται, αν πιο σύνθετη, αφού προσφέρει μεγαλύτερη

Ο επιθετικός προσδιορισμός "τυχαία" δεν έχει δοθεί με την έννοια του ότι κάτι γίνεται στην τύχη, αλλά με την έννοια του ότι αν κάποιος παρακολουθεί "εκ του μακρόθεν" τις ενέργειές μας χωρίς να είναι ενημερωμένος για το τι κάνουμε νομίζει ότι εκτελούμε προσπελάσεις σε τυχαία σημεία του αρχείου.

1. Το άνοιγμα του αρχείου.,
2. Την ανάγνωση ή και την εγγραφή των δεδομένων.
3. Το κλείσιμο του αρχείου.

Ανακεφαλαίωση

Η αλλαγή του τρέχοντος φακέλου γίνεται με την εντολή `ChDir`, η δημιουργία ενός φακέλου γίνεται με την εντολή `MkDir` και η διαγραφή με την εντολή `RmDir`. Για να διαπιστώσουμε ποιος είναι ο τρέχων φάκελος χρησιμοποιούμε την εντολή `CurDir`.

Για να αλλάξουμε όνομα σε ένα αρχείο χρησιμοποιούμε την εντολή `Name`, για να αντιγράψουμε το περιεχόμενο ενός αρχείου σε ένα άλλο, χρησιμοποιούμε την εντολή `FileCopy` και για να διαγράψουμε ένα αρχείο την εντολή `Kill`. Οι συναρτήσεις `FileDateTime`, `FileLen`, `GetAttr` δίνουν πληροφορίες σχετικές με τις παραμέτρους που χαρακτηρίζουν τα αρχεία. Η αναζήτηση και η επιλογή αρχείων γίνεται με τη συνάρτηση `Dir`, με τα αντικείμενα ελέγχου πτυσσόμενη λίστα δίσκων, λίστα φακέλων και λίστα αρχείων, με προτυποιημένα κοινά διαλογικά παράθυρα.

Εργαστηριακές Ασκήσεις

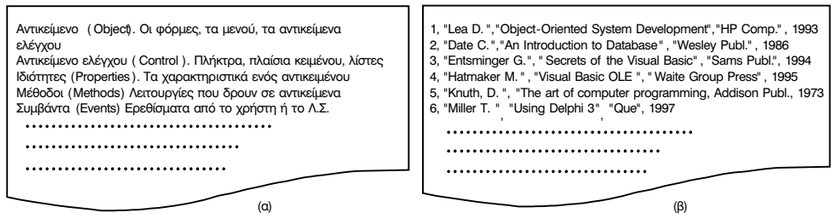
1. Να γραφεί υποροutίνα που να διαγράφει ένα φάκελο. Αν ο φάκελος περιέχει αρχεία να πραγματοποιεί πρώτα τη διαγραφή τους και μετά να γίνεται η διαγραφή του φακέλου.
2. Να εκτελεστεί η άσκηση 22-1. Να προσαρμοστεί η άσκηση, ώστε το όνομα του αρχείου να δίνεται από κοινό διαλογικό παράθυρο.
3. Να εκτελεστεί η άσκηση 22-2.
4. Να εκτελεστεί η άσκηση 22-3. Στη φόρμα να προστεθεί και πλαίσιο κειμένου από το οποίο να καθορίζεται το φίλτρο των αρχείων.
5. Να γίνει εφαρμογή από την οποία να είναι δυνατή η γενική διαχείριση αρχείων, όπως η αναζήτηση, η αντιγραφή, η μετονομασία, η διαγραφή.

Μάθημα 23
Σειριακά αρχεία

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να περιγράφουν τη δομή των σειριακών αρχείων.
- Να ανοίγουν αρχεία για ανάγνωση ή εγγραφή και να τα κλείνουν μετά το πέρας κάθε διαδικασίας. Να συντάσσουν απλές εντολές `Open`, `Close`.
- Να δημιουργούν απλά προγράμματα εισαγωγής στοιχείων σε σειριακά αρχεία.
- Να δημιουργούν απλά προγράμματα ανάγνωσης στοιχείων από σειριακά αρχεία.

Η αποθήκευση των δεδομένων στα σειριακά αρχεία δε γίνεται με κάποια ιδιαίτερη μέθοδο κωδικοποίησης. Το περιεχόμενο των σειριακών αρχείων μπορεί να διαβαστεί ακόμα και από απλούς επεξεργαστές κειμένου, όπως είναι το Σημειωματάριο (Notepad) των Windows.
Τα σειριακά αρχεία αποτελούνται από μια σειρά γραμμών κειμένου και γι' αυτό συχνά ονομάζονται και ***αρχεία κειμένου (text files)***. Κάθε γραμμή του αρχείου τερματίζεται με χαρακτήρες αλλαγής γραμμής, που ισοδυναμούν με την τιμή της σταθεράς `vbCrLf`. Τις γραμμές του αρχείου τις αποκαλούμε συχνά ***εγγραφές (records)***. Μια μορφή σειριακών αρχείων είναι και τα ***οριοθετημένα αρχεία (delimited files)***. Σε ένα οριοθετημένο αρχείο κάθε γραμμή περιέχει δεδομένα, τα οποία χωρίζονται μεταξύ τους με κόμματα (,). Τα δεδομένα μπορεί να είναι αριθμοί ή αλφαριθμητικά στοιχεία. Τα αλφαριθμητικά στοιχεία που περιέχουν κενά διαστήματα περικλείονται μέσα σε εισαγωγικά.



Σχήμα 23-1. *α) Δείγμα αρχείου κειμένου. β) Δείγμα οριοθετημένου αρχείου*

Κατά την προσπέλαση σε ένα σειριακό αρχείο, είτε διαβάζουμε εγγραφές από αυτό είτε γράφουμε. Δεν είναι δυνατή η παράλληλη ανάγνωση και εγγραφή δεδομένων από τα σειριακά αρχεία. Οι εντολές που χρησιμοποιούνται στα τρία στάδια προσπέλασης ενός σειριακού αρχείου είναι:

1. Η εντολή `Open` για το άνοιγμα του αρχείου.
2. Οι εντολές `Print #` και `Write #` για την εγγραφή δεδομένων και οι εντολές `Input #` και `Line Input #` για την ανάγνωση των δεδομένων.
3. Η εντολή `Close` για το κλείσιμο του αρχείου.

Επίσης, κατά τη σειριακή προσπέλαση μπορούμε να χρησιμοποιήσουμε και τις συναρτήσεις LOF, LOC και EOF για να ελέγξουμε το μέγεθος του αρχείου, για να ελέγξουμε τη θέση που διαβάζουμε ή γράφουμε, για να ελέγξουμε το τέλος του αρχείου, αντίστοιχα.

Άνοιγμα και κλείσιμο αρχείου

Η εντολή με την οποία ανοίγουμε ένα αρχείο είναι η εντολή `Open`. Η εντολή `Open` προετοιμάζει το αρχείο ανάλογα με το είδος της επεξεργασίας (ανάγνωση ή εγγραφή) και συντάσσεται ανάλογα κατά περίπτωση. Η γενική σύνταξη της εντολής είναι η:

`Open` *όνομα_αρχείου* `For` *τύπος_πρόσβασης* `As` *αριθμός_αρχείου*

Η εντολή `Open` χρησιμοποιείται και για τα σειριακά και για τα τυχαίας προσπέλασης αρχεία.

Το όνομα_αρχείου είναι αλφαριθμητική σταθερά, μεταβλητή ή παράσταση, που καθορίζει το όνομα του αρχείου στο υπολογιστικό σύστημα. Η τιμή της πρέπει να είναι σύμφωνη με τους γενικούς κανόνες ονοματολογίας των αρχείων. Ο τύπος_πρόσβασης, στην περίπτωση

των σειριακών αρχείων είναι μια από τις κωδικές λέξεις `Input`, `Output` και `Append`. Ο *αριθμός_αρχείου* είναι ένας ακέραιος αριθμός, μεταξύ 1 και 511, ο οποίος χρησιμοποιείται για την ταυτοποίηση του αρχείου από τις εντολές ανάγνωσης, εγγραφής ή κλεισίματος του αρχείου.

Ο *τύπος_πρόσβασης* δεν είναι παράμετρος αλλά κωδική λέξη.

Για να δημιουργήσουμε ένα σειριακό αρχείο, το ανοίγουμε με τύπο πρόσβασης `Output` (έξοδο). Αν το αρχείο δεν υπάρχει, δημιουργείται ένα κενό αρχείο. Αν το αρχείο υπάρχει, διαγράφονται όλα τα ήδη υπάρχοντα δεδομένα. Οι διαδοχικές εντολές `Print #`, που θα ακολουθήσουν, θα δημιουργήσουν το περιεχόμενό του.

Δημιουργία σειριακού αρχείου

Για να μη χαθούν οι εγγραφές που υπάρχουν σε ένα αρχείο και να γίνει επαύξηση του περιεχομένου του με νέες εγγραφές, χρησιμοποιείται ο τύπος πρόσβασης `Append` (προσθήκη). Σ' αυτήν την περίπτωση οι εντολές `Print #`, που θα ακολουθήσουν, τοποθετούν νέες εγγραφές στο τέλος του αρχείου. Ας σημειωθεί ότι δεν υπάρχει τρόπος να εισαχθούν εγγραφές στο μέσον σειριακού αρχείου.

Προσθήκη σε σειριακό αρχείο

Για να διαβάσουμε δεδομένα από ένα σειριακό αρχείο πρέπει να το ανοίξουμε με τύπο πρόσβασης `Input` (είσοδος). Στην περίπτωση αυτή η ανάγνωση αρχίζει από την πρώτη εγγραφή του αρχείου και γίνεται εγγραφή προς εγγραφή. Αν το αρχείο δεν υπάρχει γίνεται διακοπή της εκτέλεσης του προγράμματος και στην οθόνη εμφανίζεται το μήνυμα "**File not Found**".

Ανάγνωση από σειριακό αρχείο

Παραδείγματα 23-1.

α) Η εντολή:

```
Open "C:\School\Students.txt" For Input As 1
```

ανοίγει το αρχείο `"C:\School\Students.txt"` για ανάγνωση δεδομένων. Αριθμός αρχείου ορίζεται το 1.

β) Το τμήμα προγράμματος:

```
FileName = "C:\School\Lessons.txt"
FileNo = 3
Open FileName For Append As FileNo
```

ανοίγει το αρχείο `"C:\School\Lessons.txt"` για προσθήκη δεδομένων και ορίζει τον αριθμό 3 ως αριθμό αρχείου.

Στο πρόγραμμά μας μπορούμε να έχουμε πολλά αρχεία ταυτόχρονα ανοιχτά, αρκεί σε κάθε αρχείο να εκχωρούμε ένα διαφορετικό *αριθμό_αρχείου*. Ο *αριθμός_αρχείου* επιβάλλεται να είναι μοναδικός. Αν προσπαθήσουμε να ανοίξουμε ένα αρχείο με αριθμό, που ήδη έχει δεσμευτεί από άλλο αρχείο, γίνεται διακοπή της εκτέλεσης του προγράμματος και στην οθόνη προβάλλεται το διαγνωστικό μήνυμα "**File already open**".

Για να είναι το πρόγραμμά μας όσο γίνεται πιο παραμετρικό, μπορούμε να ζητάμε από την VB να μας αναφέρει τον επόμενο διαθέσιμο αριθμό αρχείου. Χρησιμοποιώντας τη συνάρτηση `FreeFile` είναι εξασφαλισμένο ότι ο αριθμός αρχείου που θα επιστρέψει ως τιμή, δε χρησιμοποιείται από άλλο αρχείο.

Παράδειγμα 23-2.

Με τις εντολές:

```
FileNumber = FreeFile
Open "C:\Accounts\Pay.Dat" For Output As FileNumber
```

ο αριθμός αρχείου, για το αρχείο `"C:\Accounts\Pay.Dat"` δίνεται αυτόματα από την VB.

Είναι απαραίτητο να κλείνουμε τα αρχεία που ανοίγουμε με εντολή `Open`, όταν πλέον η προσπέλαση σταματά. Μόνο έτσι είναι δυνατή η απελευθέρωση πόρων του υπολογιστικού συστήματος και η διάθεσή τους σε άλλα προγράμματα. Η εντολή με την οποία γίνεται το κλείσιμο των ανοιχτών αρχείων είναι η εντολή `Close`. Η γενική μορφή της εντολής είναι:

`Close` *# αριθμός_αρχείου1, αριθμός_αρχείου2, . . .*

Η αναφορά των *αριθμών_αρχείων* είναι προαιρετική. Αν συντάξουμε την εντολή `Close` χωρίς παραμέτρους, τότε κλείνουν όλα τα ανοιχτά αρχεία.

Παράδειγμα 23-3.

Ο παρακάτω κώδικας ανοίγει δύο σειριακά αρχεία, ένα για ανάγνωση και ένα για καταχώριση στοιχείων, και στη συνέχεια τα κλείνει.

```
Dim ReadFileNo       As Integer
Dim WriteFileNo      As Integer
'Áíïéãìá áñ÷åßùí
 ReadFileNo = FreeFile     'Ï ðñþôïò áñéèìüò áñ÷åßïõ
 Open "C:\Warehouse\PartsIn.Dat" For Input As ReadFileNo
 WriteFileNo = FreeFile    'Ï äåýôåñïò áñéèìüò áñ÷åßïõ
Open "C:\Warehouse\PartsOut.Dat" For Output As WriteFileNo
'
'Åðåîåñãáóßá äåäïìÝíùí
'
        . . .

'Êëåßóéìï áñ÷åßùí
 Close ReadFileNo, WriteFileNo
```

Αν δεν κλείσουμε κάποια από τα αρχεία που έχουμε ανοίξει κατά τη διάρκεια εκτέλεσης του προγράμματος, η VB τα κλείνει αυτόματα και απελευθερώνει τους πόρους τους με τον τερματισμό του προγράμματος.

Η εγγραφή στα σειριακά αρχεία

Αφού ανοίξουμε ένα αρχείο για σειριακή προσπέλαση, μπορούμε να γράψουμε δεδομένα σ' αυτό χρησιμοποιώντας την εντολή `Print #`. Η σύνταξη της εντολής είναι:

`Print` *#αριθμός_αρχείου, παράμετρος1, παράμετρος2, . . .*

Ο *αριθμός_αρχείου* είναι ο αριθμός του αρχείου με τον οποίο έχουμε συνδέσει το αρχείο και *παράμετρος1, παράμετρος2, ...* μια σειρά σταθερών, μεταβλητών ή παραστάσεων των οποίων τις τιμές θέλουμε να αποθηκεύσουμε στο αρχείο. Η εντολή μοιάζει με τη μέθοδο `Print`. Όπως και στη μέθοδο `Print` το πλήθος των παραμέτρων δεν είναι καθορισμένο. Δεξιά από την κωδική λέξη `Print` είναι δυνατόν να θέσουμε όσες παραμέτρους θέλουμε. Ο διαχωρισμός τους δεν είναι υποχρεωτικό να γίνει με κόμματα, είναι δυνατόν να γίνει με κενά διαστήματα ή με το σύμβολο semicolon (;). Τα σύμβολα αυτά δεν παίζουν μόνο διαχωριστικό ρόλο μεταξύ των παραμέτρων, αλλά επιδρούν στη μορφή της εγγραφής.

Παραδείγματα 23-4.

α) Το τμήμα προγράμματος:

```
Factor = 0.7
Profit = 5.8
Cost = -3.5
SName = "ÐÁÐÁÈÁÍÏÕ"
FName = "ÉÙÁÍÍÇÓ"
Print #1, SName, FName, Factor, Profit, Cost
```

κάνει την εγγραφή:

```
ÐÁÐÁÈÁÍÏÕ      ÉÙÁÍÍÇÓ         0.7             5.8            -3.5
```

β) Αν αντικαταστήσουμε τα διαχωριστικά των παραμέτρων και αντί των κομμάτων χρησιμοποιήσουμε σύμβολα semicolon (;) και γράψουμε:

```
Print #1, SName; FName; Factor; Profit; Cost
```

θα γίνει η εγγραφή:

```
ÐÁÐÁÈÁÍÏÕÉÙÁÍÍÇÓ 0.7  5.8 -3.5
```

γ) Στην πιο πάνω εγγραφή, τα δεδομένα δεν είναι οριοθετημένα, αφού δεν χωρίζονται με κόμματα. Για να υπάρχει μεταξύ τους οριοθέτηση, ώστε να είναι δυνατή η διακριτή τους ανάγνωση από τις εντολές ανάγνωσης (π.χ. την εντολή `Input #` που θα μελετήσουμε στη

συνέχεια), πρέπει να παρεμβάλλουμε κόμματα και μεταξύ τωνδεδομένων. Διαμορφώνουμε λοιπόν την εντολή σε:

```
Print #1, SName; ","; FName; ","; Factor; ","; Profit; ","; Cost
```

που κάνει την εγγραφή:

```
ÐÁÐÁÈÁÍÏÕ,ÉÙÁÍÍÇÓ, 0.7 , 5.8 ,-3.5
```

δ) Αν επιπλέον θέλουμε τα αλφαριθμητικά δεδομένα να περιέχονται μέσα σε εισαγωγικά, πρέπει να γράψουμε:

```
Dim Quote       As String
  . . . .
Quote = Chr(34)     'Ï êùäéêüò ASCII ðïõ áíôéóôïé÷åß óôï "
  . . . .
Print #1, Quote; SName; Quote; ","; Quote; FName; Quote ","; _
         Factor; ","; Profit; ","; Cost
```

για να γίνει η εγγραφή:

```
"ÐÁÐÁÈÁÍÏÕ","ÉÙÁÍÍÇÓ", 0.7 , 5.8 ,-3.5
```

Παρατηρούμε ότι η εντολή `Print #` γίνεται αρκετά σύνθετη, όταν πρόκειται να γράψουμε σε οριοθετημένο αρχείο. Αντί αυτής χρησιμοποιείται η εντολή `Write #` που έχει παρόμοια σύνταξη και καταχωρίζει τα δεδομένα διαχωρισμένα με κόμμα, περικλείει τα αλφαριθμητικά δεδομένα σε εισαγωγικά και γράφει τις ημερομηνίες μεταξύ συμβόλων (#). Η σύνταξη της εντολής `Write #` είναι:

`Write` *#αριθμός_αρχείου, παράμετρος1, παράμετρος2, . . .*

Άσκηση 23-1.

Σε ένα αρχείο, θέλουμε να καταγράφονται τα μηνύματα κατάστασης λειτουργίας ενός προγράμματος (αρχείο log). Κάθε φορά που αλλάζει η κατάσταση να προστίθεται στο τέλος του αρχείου μια εγγραφή που να περιλαμβάνει το όνομα του χειριστή, το σημείο του προγράμματος που προκαλεί την εγγραφή, τον κωδικό κατάστασης και την ώρα.
Δημιουργούμε μια υπορουτίνα, που δέχεται ως παραμέτρους τα πιο πάνω στοιχεία στις τυπικές μεταβλητές `User`, `Segment`, `Status` αντίστοιχα και τα αποθηκεύει στο αρχείο `"C:\Logs\MyProg.Txt"`. Συγκεκριμένα γράφουμε:

```
Sub WriteToLog(ByVal User As String, _
               ByVal Segment As String, _
               ByVal Status As Integer)
    Dim FileNo As Integer
   'Áíïéíå ôï áñ÷åßï
    FileNo = FreeFile
    Open "C:\Logs\MyProg.Txt" For Append As FileNo
   'Åããñáöþ êáôÜóôáóçò
    Write #FileNo, User, Segment, Status, Now()
   'Êëåßóå ôï áñ÷åßï
    Close FileNo
End Sub
```

Παρατηρήστε ότι το αρχείο έχει ανοιχτεί ως τύπου Append, ώστε να γίνεται επέκτασή του με κάθε κλήση της ρουτίνας. Η γραμμή που ακολουθεί είναι δείγμα γραφής στο αρχείο.

```
"ÓêåíóÝñïãëïõ Ößëéðäïò","Äéáãñáöþ êéíÞóåùí",15,#2000-11-02 03:03:26#
"Ìé÷áëüðïõëïò Íßêïò","Áëëáãþ öüñìáò",15,#2000-11-02 03:03:28#
"Ìðßæìò ÁíÝóôçò","Êáô÷þñéóç áðïèÞêçò",15,#2000-11-02 03:04:32#
```

Η οριοθέτηση των δεδομένων μέσα σε μια εγγραφή βοηθά σημαντικά την ανάγνωση τους. Οι περιοχές που καταλαμβάνουν τα δεδομένα διακρίνονται με ευκολία. Την περιοχή που καταλαμβάνει κάθε δεδομένο την ονομάζουμε ***πεδίο (field)***. Το μεγάλο πλεονέκτημα της

εντολής `Write,` έναντι της εντολής `Print,` είναι ότι δημιουργεί σαφώς διακρινόμενα πεδία. Κάθε παράμετρος της εντολής `Write` αντιστοιχεί σε ένα μόνο πεδίο της εγγραφής.

Η ανάγνωση από τα σειριακά αρχεία

Η πιο απλή εντολή, τόσο ως προς τη σύνταξη όσο και ως προς τη λειτουργία, για τη σειριακή ανάγνωση από ένα αρχείο, είναι η εντολή `Line Input #`. Η γενική μορφή της εντολής είναι η:

`Line Input` *#αριθμός_αρχείου, μεταβλητή*

Η *μεταβλητή* πρέπει να είναι αλφαριθμητικού τύπου και η τιμή της, μετά την εκτέλεση της εντολής, είναι το περιεχόμενο της εγγραφής που διαβάστηκε. Η εντολή δηλαδή, δεν κάνει διάκριση των δεδομένων που περιέχονται μέσα στην εγγραφή. Έτσι, είναι δυνατή η ανάγνωση από οποιοδήποτε αρχείο, ακόμα και αν δεν γνωρίζουμε τη δομή των δεδομένων που περιέχει η κάθε εγγραφή του.

Μετά το άνοιγμα του αρχείου, η πρώτη εντολή `Line Input #`, διαβάζει την πρώτη εγγραφή του αρχείου, η δεύτερη εντολή τη δεύτερη εγγραφή κ.ο.κ. Η προσπάθεια να διαβάσουμε έξω από το αρχείο, προκαλεί διακοπή του προγράμματος και εμφάνιση του μηνύματος **"Input past end of file"**. Για να αποφύγουμε τη διακοπή του προγράμματος από μια τέτοια αιτία, πρέπει να κάνουμε έλεγχο πριν από κάθε εντολή ανάγνωσης, μήπως και έχουμε φτάσει στο τέλος του αρχείου. Ο έλεγχος αυτός γίνεται με τη συνάρτηση EOF (End Of File) που δέχεται ως παράμετρο τον *αριθμό_αρχείου*.

Άσκηση 23-2.

Στην προγραμματιστική μονάδα μιας φόρμας να γραφτεί υπορουτίνα, η οποία να διαβάζει το περιεχόμενο οποιουδήποτε αρχείου κειμένου και να το παρουσιάζει στο εσωτερικό της φόρμας.
Στη φόρμα τοποθετούμε ένα πλήκτρο με όνομα `Display` και ένα αντικείμενο ελέγχου κοινό διαλογικό παράθυρο με όνομα `Dialog`. Στη ρουτίνα διαχείρισης συμβάντος **`Click`** του πλήκτρου, γράφουμε τον κώδικα:

```
Private Sub Display_Click()
    Dim FileName  As String          'Ôï üíïìá ôïõ áñ÷åßïõ
    Dim FileNo    As Integer         'Ï áñéèìüò áñ÷åßïõ
    Dim FileRec   As String          'Ïëç ç åããñáöÞ
   'Åðýëåîå ôï áñ÷åßï
    Dialog.Filter = "all files (*.*)|*.*"
    Dialog.ShowOpen                  'Áéáëïãáéêü ðáñÜèõñï Open
    Filename = Dialog.Filename
   'Áíïéíå ôï áñ÷åßï
    FileNo = FreeFile
    Open Filename For Input As FileNo
   'ÄéÜâáóå ãñáììÞ ãñáììÞ êáé ðáñïõóßáóÝ ôçí óôç öüñìá
    Do While Not EOF(FileNo)
        Line Input #FileNo, FileRec
        Print FileRec
    Loop
   'Êëåßóå ôï áñ÷åßï
    Close FileNo
End Sub
```

Όταν πρόκειται να διαβάσουμε δεδομένα από ένα οριοθετημένο αρχείο, του οποίου γνωρίζουμε τη δομή, χρησιμοποιούμε την εντολή `Input #` που έχει τη μορφή:

`Input` *#αριθμός_αρχείου, μεταβλητή1, μεταβλητή2, . . .*

Οι παράμετροι *μεταβλητή1, μεταβλητή2, . . .* είναι αριθμητικές ή και αλφαριθμητικές μεταβλητές που κατά την εκτέλεση της εντολής παίρνουν ως τιμές τις τιμές των αντίστοιχων πεδίων της εγγραφής. Τα όρια των πεδίων πρέπει να είναι διακριτά και σύμφωνα με τον τρόπο που τα

δημιουργεί η εντολή `Write #`. Μεταξύ των δεδομένων πρέπει να υπάρχουν κόμματα, οι αλφαριθμητικές τιμές να περικλείονται σε εισαγωγικά (") και οι ημερομηνίες να περικλείονται μεταξύ συμβόλων (#).

Παράδειγμα 23-5.

Έστω ότι σε ένα αρχείο περιέχονται τα δεδομένα:

```
7.1, -48, Êñçôéêü, " ÈÜëáóóá ÊáñðÜèïõ"
-8907, 3258, Éüíéï, ËåõêÜää
296, 4912, "Ìõñôþï,Áéãáßï", "ÖáëêîíÝñá"
```

μετά την εκτέλεση του τμήματος προγράμματος:

```
Print "Ðñþôï Ðåäßï", "Äåýôåñï Ðåäßï", "Ôñßôï Ðåäßï", "ÔÝôáñôï Ðåäßï"
For RecNo = 1 To 3
    Input #1, Field1, Field2, Field3, Field4
    Print Field1, Field2, Field3, Field4
Next RecNo
```

στο παράθυρο της φόρμας θα παίρνουμε την εκτύπωση:

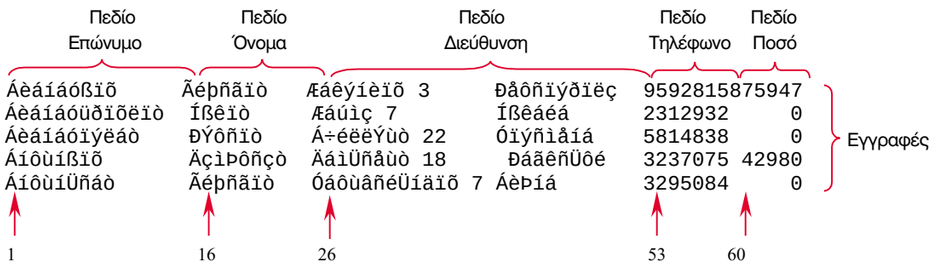
```
Ðñþôï Ðåäßï   Äåýôåñï Ðåäßï Ôñßôï Ðåäßï    ÔÝôáñôï Ðåäßï
7.1            -48           Êñçôéêü          ÈÜëáóóá ÊáñðÜèïõ
-8907          3258          Éüíéï            ËåõêÜää
296            4912          Ìõñôþï,Áéãáßï    ÖáëêîíÝñá"
```

Δομημένες εγγραφές σειριακών αρχείων

Από όσα έχουμε αναφέρει μέχρι τώρα είναι φανερό ότι με την VB μπορούμε να δημιουργήσουμε πολύ εύκολα σειριακά αρχεία, των οποίων τα πεδία έχουν μεταβλητό μήκος και οριοθετούνται από ειδικά σύμβολα. Αυτά τα αρχεία τα δημιουργούμε, συνήθως, όταν θέλουμε να γίνεται μεταβίβαση δεδομένων από ένα πρόγραμμα VB σε ένα άλλο πρόγραμμα της VB. Υπάρχουν όμως περιπτώσεις που θέλουμε να δημιουργήσουμε αρχεία, στα οποία τα δεδομένα θα είναι στηλοθετημένα για να διακρίνονται και να διαβάζονται εύκολα και από προγράμματα άλλων γλωσσών προγραμματισμού ή να εκτυπώνονται σε στήλες στον εκτυπωτή. Τα πεδία αυτών των αρχείων πρέπει να είναι ***σταθερού πλάτους (fixed width)***.

Παράδειγμα 23-6

Έστω ότι στο αρχείο `Customer.dat` υπάρχουν αποθηκευμένα τα στοιχεία των πελατών μιας εταιρείας. Για κάθε πελάτη έχει καταγραφεί το επώνυμο, το όνομα, διεύθυνση, ο αριθμός τηλεφώνου, το ποσό που οφείλει. Το περιεχόμενο του αρχείου έχει τη μορφή:



όπου οι αριθμοί δηλώνουν τη θέση, στην οποία αρχίζει το κάθε πεδίο μέσα στην εγγραφή. Το αρχείο αυτό μπορεί να τυπωθεί ως έχει στον εκτυπωτή.

Συνήθως, την περιγραφή της εγγραφής ενός αρχείου την κάνουμε με πίνακα που περιέχει τον αύξοντα αριθμό του πεδίου, την περιγραφή του περιεχομένου του, την περιοχή του, το μέγεθός του, τον τύπο των δεδομένων που θα περιέχει. Ο πίνακας αυτός ονομάζεται πίνακας ***γραμμογράφησης.***

Παράδειγμα 23-7.

Ο πίνακας γραμμογράφησης του παραδείγματος 23-6 είναι ο:

Πεδίο	Περιγραφή	Περιοχή	Εύρος	Τύπος
1	Επώνυμο	1-15	15	Λεκτικό
2	Όνομα	16-25	10	Λεκτικό
3	Διεύθυνση	26-52	27	Λεκτικό
4	Αριθμός τηλεφώνου	53-59	7	Αριθμός
5	Υπόλοιπο	60-65	6	Αριθμός

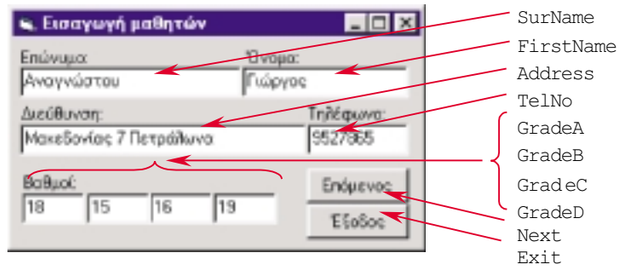
Άσκηση 23-3.

Σε ένα σχολείο ο καθηγητής των Φυσικών αποφάσισε να αποθηκεύσει σε ένα αρχείο τους βαθμούς των μαθητών του. Να δημιουργηθεί φόρμα με την οποία θα γίνεται η καταχώριση των στοιχείων των μαθητών στο αρχείο `"C:\School\Students.dat"`.

Έστω, ότι η γραμμογράφηση του αρχείου των μαθητών είναι η:

Πεδίο	Περιγραφή	Περιοχή	Εύρος	Τύπος
1	Επώνυμο	1-15	15	Λεκτικό
2	Όνομα	16-25	10	Λεκτικό
3	Διεύθυνση	26-52	27	Λεκτικό
4	Αριθμός τηλεφώνου	53-59	7	Αριθμός
5	Βαθμός α'	60-62	3	Αριθμός
6	Βαθμός β'	63-65	3	Αριθμός
7	Βαθμός γ'	66-68	3	Αριθμός
8	Βαθμός δ'	69-71	3	Αριθμός

Δημιουργούμε τη φόρμα της εικόνας 23-1. Ο χρήστης θα καταχωρίσει τα στοιχεία του μαθητή στη φόρμα. Για να μεταφερθούν τα στοιχεία από τη φόρμα στο αρχείο πρέπει να πατά το πλήτρο `Next`. Όταν τελειώσει τις καταχωρίσεις, θα πατήσει το πλήκτρο `Exit`.



Εικόνα 23-1*. Φόρμα καταχώρισης βαθμολογιών*

Γράφουμε τον κώδικα:

```
Option Explicit
   'ÊáèïëéêÝò ìåôáâëçôÝò
    Dim FileNo      As Integer
    Dim FileRec     As String

Private Sub Form_Load()
   'Áí äåí õðÜñ÷åé ï öÜêåëïò, äçìéïýñãçóÝ ôïí
    If Dir("C:\School", vbDirectory) = "" Then MkDir "C:\School"

   'Áíïéêôü ôï áñ÷åßï ãéá ðñïóèÞêç óôïé÷åßùí
    FileNo = FreeFile
    Open "C:\School\Students.dat" For Append As FileNo
End Sub
```

```
Private Sub Next_Click()
    'Äçìéïýñãçóå ôçí åããñáöÞ ðåäßï ðåäßï
     FileRec = Left(SurName & Space(15), 15)
     FileRec = FileRec & Left(FirstName & Space(10), 10)
     FileRec = FileRec & Left(Address & Space(27), 27)
     FileRec = FileRec & Left(TelNo & Space(7), 7)
     FileRec = FileRec & Right(Space(3) & GradeA, 3)
     FileRec = FileRec & Right(Space(3) & GradeB, 3)
     FileRec = FileRec & Right(Space(3) & GradeC, 3)
     FileRec = FileRec & Right(Space(3) & GradeD, 3)
     Print #FileNo, FileRec
    'ÊáèÜñéóå ôá ðåäßá
     SurName = ""
     FirstName = ""
     Address = ""
     TelNo = ""
     GradeA = ""
     GradeB = ""
     GradeC = ""
     GradeD = ""
End Sub
Private Sub Exit_Click()
    'Êëåßóå ôï áñ÷åßï
     Close
    'Îåöüñôùóå ôç öüñìá
    Unload Me
End Sub
```

Όταν φορτώνεται η φόρμα, ανιχνεύεται αν υπάρχει ο φάκελος του αρχείου και αν ο φάκελος δεν υπάρχει, δημιουργείται. Στη συνέχεια ανοίγεται και το αρχείο που παραμένει ανοιχτό συνεχώς, μέχρι ο χρήστης να πατήσει το πλήκτρο `Exit`.

Παρατηρήστε τον τρόπο σύνθεσης της εγγραφής, πεδίο προς πεδίο, στη ρουτίνα διαχείρισης συμβάντος `Next_Click`. Για τα στοιχισμένα στα αριστερά πεδία (π.χ. αλφαριθμητικά πεδία) τοποθετούνται κενά διαστήματα στα δεξιά και λαμβάνεται το αριστερό τμήμα με μέγεθος ίσο με το μέγεθος του πεδίου. Το αντίθετο συμβαίνει για τα στοιχισμένα στα δεξιά πεδία (π.χ. αριθμητικά πεδία).

Άσκηση 23-4.

Ανάγνωση από αρχείο με σταθερό πλάτος πεδίων

Μπορούμε να χρησιμοποιήσουμε τη φόρμα της εικόνας 23-1 για να παρουσιάσουμε τα στοιχεία που περιέχονται στο αρχείο μαθητών. Όταν φορτώνεται η φόρμα θα ελέγχει αν υπάρχει το αρχείο με τα στοιχεία των μαθητών και αν δεν υπάρχει,, θα γίνεται διακοπή του προγράμματος. Κάθε φορά που ο χρήστης πατά το πλήκτρο `Next` θα εμφανίζεται στη φόρμα και το περιεχόμενο της επόμενης εγγραφής.

Δημιουργούμε μια νέα φόρμα (μπορεί να προκύψει από αντιγραφή της παλαιάς) και γράφουμε τον κώδικα:

```
Option Explicit
   'Êáèïëéêôýò ìåôáâëçôýò
    Dim FileNo      As Integer
    Dim FileRec     As String

Private Sub Form_Load()
    'Áí äåí õðÜñ÷åé ôï áñ÷åßï äéÜêïøå
     If Dir("C:\School\Students.dat") = "" Then
        MsgBox "Äåí õðÜñ÷åé ôï áñ÷åßï", vbCritical, "ÁíáöïñÜ ËÜèïõò"
        Unload Me
     End If
    'Áíïßîå ôï áñ÷åßï ãéá áíÜãíùóç óôïé÷åßùí
     FileNo = FreeFile
     Open "C:\School\Students.dat" For Input As FileNo
End Sub
```



```
Private Sub Next_Click()
   'ÄéÜâáóå ôçí åããñáöÞ êáé áðïìüíùóå ôá ðåäßá
    If Not EOF(FileNo) Then
        Line Input #FileNo, FileRec
        SurName = Mid(FileRec, 1, 15)
        FirstName = Mid(FileRec, 16, 10)
        Address = Mid(FileRec, 26, 27)TelNo = Mid(FileRec, 53, 7)
        GradeA = Mid(FileRec, 60, 3)
        GradeB = Mid(FileRec, 63, 3)
        GradeC = Mid(FileRec, 66, 3)
        GradeD = Mid(FileRec, 69, 3)
    Else
        MsgBox "ÔÝëïò ôïõ áñ÷åßïõ", vbExclamation, "ÁíáöïñÜ ËÜèïõò"
    End If
End Sub

Private Sub Exit_Click()
   'Êëåßóå ôï áñ÷åßï
    Close
   'Îåöüñôùóå ôç öüñìá
    Unload Me
End Sub
```

Παρατηρήστε τον τρόπο με τον οποίο γίνεται η απομόνωση του περιεχομένου κάθε πεδίου της εγγραφής. Χρησιμοποιείται η συνάρτηση `Mid`.

Ανακεφαλαίωση

Τα σειριακά αρχεία αποτελούνται από μια σειρά γραμμών κειμένου και γι' αυτό συχνά ονομάζονται και αρχεία κειμένου. Τις γραμμές του αρχείου τις αποκαλούμε συχνά εγγραφές. Τα σειριακά αρχεία μπορούμε να τα ανοίξουμε ή για να τα διαβάσουμε ή για να τα γράψουμε. Δεν είναι δυνατή η ταυτόχρονη ανάγνωση και εγγραφή.

Οι εντολές που χρησιμοποιούνται για την προσπέλαση των σειριακών αρχείων είναι η εντολή `Open` για το άνοιγμα του αρχείου, οι εντολές `Print #` και `Write #` για την εγγραφή δεδομένων, οι εντολές `Input #` και `Line Input #` για την ανάγνωση και η εντολή `Close` για το κλείσιμο του αρχείου. Με τη συνάρτηση `EOF` μπορούμε να ανιχνεύουμε το τέλος του αρχείου.

Εργαστηριακές Ασκήσεις

1. Να προσομοιωθεί η άσκηση 23-1. Να δημιουργηθεί φόρμα και επάνω της να τοποθετηθούν τρία πλήκτρα που το πάτημά τους υποτίθεται ότι καλεί διαφορετικές φόρμες. Πριν τη μετάβαση σε κάθε φόρμα, να γίνεται εγγραφή στο αρχείο log.
   Υπόδειξη: Για να επαληθεύσετε ότι η διαδικασία δούλεψε σωστά ανοίξτε τον επεξεργαστή κειμένου Notepad και δείτε το περιεχόμενο του αρχείου.
2. Να εκτελεστεί η άσκηση 23-2. Αναζητήστε αρχεία με όνομα `Readme.txt` στους φακέλους του υπολογιστικού σας συστήματος και διαβάστε το περιεχόμενό τους.
3. Αντιπαραβολή του περιεχομένου αρχείων
   Δύο αρχεία περιέχουν περίπου τα ίδια δεδομένα. Να γραφτεί πρόγραμμα που να διαβάζει μια εγγραφή από κάθε αρχείο και να κάνει τη μεταξύ τους σύγκριση. Αν βρεθούν διαφορετικές εγγραφές να εμφανίζεται στην οθόνη μήνυμα με τη γραμμή που βρέθηκε η διαφορά. Η διαδικασία να συνεχίζεται μέχρι να ανιχνευθεί το τέλος των αρχείων. Αν ένα από τα αρχεία περιέχει επιπλέον στοιχεία να εμφανίζεται κατάλληλο μήνυμα.
4. Γράψτε πρόγραμμα που να δημιουργεί ένα αρχείο και να το γεμίζει με το περιεχόμενο δύο άλλων αρχείων, παίρνοντας εναλλάξ μια εγγραφή από το πρώτο αρχείο και μια εγγραφή από το δεύτερο. Να καλύψετε και την περίπτωση που τα αρχεία δεν θα έχουν το ίδιο πλήθος εγγραφών.

5. Αναζήτηση συμβολοσειράς σε αρχείο
   Γράψτε πρόγραμμα που να αναζητά μια συμβολοσειρά μέσα σε ένα αρχείο. Όταν το πρόγραμμα βρίσκει τη συμβολοσειρά, να εμφανίζει στην οθόνη διαλογικό παράθυρο, στο οποίο να αναφέρει τον αύξοντα αριθμό της εγγραφής. Η αναζήτηση να συνεχίζεται για επόμενη εμφάνιση στην περίπτωση που ο χρήστης κάνει μια τέτοια επιλογή.
6. Να εκτελεστεί η άσκηση 23-3. Να εισαχθούν πραγματικά δεδομένα στο αρχείο.
7. Να εκτελεστεί η άσκηση 23-4.
8. Να ενωποιηθούν οι φόρμες των ασκήσεων 23-3 και 23-4 σε μία. Όταν φορτώνεται η φόρμα να γίνεται ερώτηση στο χρήστη, αν θέλει να εισάγει στοιχεία στο αρχείο ή να διαβάσει τα στοιχεία του αρχείου. Ανάλογα με την απάντηση να ανοίγει το αρχείο για ανάγνωση ή για εγγραφή.
   Υπόδειξη: Δημιουργήστε μια υπορουτίνα ανάγνωσης και μια υπορουτίνα εγγραφής στο αρχείο. Η υπορουτίνα διαχείρισης συμβάντος `Next_Click` να καλεί κατά περίπτωση τη μια ή την άλλη υπορουτίνα.
9. Να δημιουργηθεί πρόγραμμα που να διαβάζει ένα κείμενο από ένα αρχείο και να μετατρέπει όλα τα γράμματα του σε κεφαλαία. Το αποτέλεσμα να αντικαθιστά το περιεχόμενο του αρχικού αρχείου.
10. Να γραφτεί πρόγραμμα που να δημιουργεί ένα αρχείο με ποδοσφαιρικές ομάδες. Το αρχείο να έχει τη γραμμογράφηση:

Πεδίο	Περιγραφή	Περιοχή	Εύρος	Τύπος
1	Όνομα ομάδας	1-15	15	Λεκτικό
2	Αγώνες	16-18	3	Αριθμός
3	Βαθμοί	19-22	4	Αριθμός
4	Νίκες	23-25	3	Αριθμός
5	Ήττες	26-28	3	Αριθμός
6	Ισοπαλίες	29-31	3	Αριθμός
7	Γκολ που πέτυχε	32-35	4	Αριθμός
8	Γκολ που δέχτηκε	36-39	4	Αριθμός

11. Να γραφτεί πρόγραμμα που να διαβάζει τα αποτελέσματα των αγώνων μιας αγωνιστικής ημέρας και συμβουλευόμενο το πιο πάνω αρχείο να δημιουργεί ένα άλλο, ενημερωμένο με τα νέα στοιχεία.

Μάθημα 24
Αρχεία τυχαίας προσπέλασης
Δυαδικά αρχεία

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να περιγράφουν τη δομή των αρχείων τυχαίας προσπέλασης.
- Να δημιουργούν απλά προγράμματα εισαγωγής και ανάγνωσης στοιχείων σε και από αρχεία τυχαίας προσπέλασης.
- Να δημιουργούν απλά προγράμματα εισαγωγής και ανάγνωσης στοιχείων σε και από δυαδικά αρχεία.

Η σειριακή προσπέλαση των αρχείων θέτει αρκετούς περιορισμούς και δημιουργεί δυσκολίες στην επεξεργασία και τη διαχείριση των δεδομένων. Δεν επιτρέπει την ταυτόχρονη ανάγνωση και εγγραφή στις εγγραφές του αρχείου, καθιστά αδύνατη την αλλαγή του περιεχομένου μιας συγκεκριμένης εγγραφής, δεν υποστηρίζει την άμεση προσπέλαση της πληροφορίας κ.ά.

Στα αρχεία τυχαίας προσπέλασης δεν υπάρχουν τέτοιας μορφής περιορισμοί. Οι εγγραφές εντοπίζονται και προσπελαύνονται σύμφωνα με τον αριθμό της εγγραφής που υποδεικνύουμε. Η διαχείριση των δεδομένων είναι πιο άμεση και τόσο η εγγραφή όσο και η ανάγνωση μπορεί να γίνει με οποιαδήποτε σειρά. Όμως τα αρχεία τυχαίας προσπέλασης πρέπει να έχουν για όλες τις εγγραφές τους στο ίδιο μέγεθος, η γραμμογράφηση των εγγραφών τους να είναι αυστηρά τυποποιημένη και να προδηλώνεται μέσα στο πρόγραμμα, με τύπο δεδομένων οριζόμενο από τον χρήστη.

Δομή εγγραφής

Όπως έχουμε δει, με τη γραμμογράφηση γίνεται η περιγραφή των πεδίων μιας εγγραφής. Η γραμμογράφηση μπορεί να κωδικοποιηθεί σε γλώσσα VB και να δημιουργήσει έναν ***τύπο δεδομένων ορισμένο από το χρήστη (user-defined data type)***. Γίνεται με τις δηλωτικές εντολές `Type` ... `End Type`:

```
Type όνομα_τύπου
    πεδίο1   As   τύπος_δεδομένων
    πεδίο2   As   τύπος_δεδομένων
                :
End Type
```

Το *όνομα_τύπου* αποτελεί το βασικό χαρακτηριστικό του οριζόμενου από τον χρήστη τύπου και μπορεί να είναι οποιαδήποτε μη δεσμευμένη λέξη. Οι τύποι_δεδομένων μπορεί να είναι ή ενσωματωμένοι στη VB τύποι (π.χ. `Long`, `Double`, `String*` κ.ά.) ή άλλοι τύποι που έχουν ήδη οριστεί από το χρήστη ή και τα δύο. Όταν έχουμε ορίσει έναν τύπο μπορούμε να δηλώσουμε και μεταβλητές ή και πίνακες μεταβλητών αυτού του τύπου. Η δήλωση γίνεται με τις δηλωτικές εντολές `Dim`, `Private`, `Public`.

Οι τύποι δεδομένων που έχουν ορστείί από το χρήστη μπορούν να χρησιμοποιηθούν και σε άλλες εφαρμογές που δεν έχουν σχέση με την επεξεργασία αρχείων τυχαίας προσπέλασης.

Παράδειγμα 24-1.

Οι εντολές:

```
Type CustomerType
    FName       As String*15   'Ïíïìá
    SName       As String*10   'Åðþíõìï
    Address     As String*27   'Äéåýèõíóç
    TelNo       As String*7    'Áñéèìüò ôçëåöþíïõ
    Rest        As long        'Õðüëïéðï
End Type
Dim CustomerRec  As CustomerType
```

ορίζουν έναν τύπο δεδομένων με τα στοιχεία των πελατών μιας εταιρείας (δείτε και παράδειγμα 23-6). Για τον πελάτη κρατούνται τα στοιχεία επώνυμο, όνομα, διεύθυνση,

αριθμός τηλεφώνου, ποσό οφειλής. Η τελευταία εντολή πραγματοποιεί τη δήλωση τηςμεταβλητής CustomerRec, ως μεταβλητής του τύπου CustomerType.

Η αναφορά σε ένα πεδίο μιας μεταβλητής, της οποίας ο τύπος είναι ορισμένος από το χρήστη, γίνεται συνδυάζοντας το όνομα της μεταβλητής, μία τελεία (.), και το όνομα του πεδίου.

Παράδειγμα 24-2.

Οι εντολές:

```
CustomerRec.FName = "ÃéÜííçò"
CustomerRec.SName = "ÌáñáìÝíïò"
CustomerRec.Rest = CustomerRec.Rest - Pay
```

δίνουν τιμές στα πεδία της μεταβλητής CustomerRec. Ας σημειωθεί ότι στα πεδία τύπου String* , όπως είναι τα πεδία Fname και Sname, η τιμή αποθηκεύεται με αριστερή στοίχιση και το υπόλοιπο του πεδίου γεμίζει με κενά διαστήματα.

Οι ορισμοί των τύπων δεδομένων του χρήστη πρέπει να γίνονται στην περιοχή δηλώσεων των προγραμματιστικών μονάδων και ποτέ στο εσωτερικό υπορουτινών ή συναρτήσεων. Αντίθετα, οι δηλώσεις των μεταβλητών ως τύπου οριζόμενου από τον χρήστη μπορεί να γίνουν σε οποιοδήποτε σημείο. Αν και ο τύπος String μπορεί να χρησιμοποιηθεί ως τύπος πεδίου, δεν πρέπει να γίνεται κάτι τέτοιο στην περίπτωση που ο οριζόμενος από τον χρήστη τύπος θα χρησιμοποιηθεί για τη δήλωση μεταβλητής, που θα αποθηκεύει το περιεχόμενό της σε αρχείο τυχαίας προσπέλασης. Όπως προαναφέραμε, οι εγγραφές στα αρχεία τυχαίας προσπέλασης πρέπει να έχουν όλες το ίδιο μέγεθος.

Οι εντολές στην τυχαία προσπέλαση

Οι εντολές που χρησιμοποιούνται για την επεξεργασία των αρχείων τυχαίας προσπέλασης είναι:

1. Η εντολή Open για το άνοιμα του αρχείου.
2. Η εντολή Put # για την εγγραφή δεδομένων και η εντολή Get # για την ανάγνωση των δεδομένων.
3. Η εντολή Close για το κλείσιμο του αρχείου.

Η εντολή με την οποία ανοίγουμε ένα αρχείο, είναι η εντολή Open, που μελετήσαμε στο προηγούμενο μάθημα. Η σύνταξη της εντολής για τυχαία προσπέλαση είναι:

Open *όνομα_αρχείου* For Random As *αριθμός_αρχείου* Len = *μέγεθος*

Στην τυχαία προσπέλαση η παράμετρος Len, καθορίζει το μήκος (Length) σε bytes κάθε εγγραφής του αρχείου. Αν παραληφθεί, το μήκος της εγγραφής θεωρείται 128 bytes.

Παραδείγματα 24-3.

Σε συνέχεια του παραδείγματος 24-1, η εντολή:

```
Open "C:\Customers.dat" For Random As 1 Len = Len(CustomerRec)
```

ανοίγει το αρχείο "C:\Customers.dat" για τυχαία προσπέλαση. Το μέγεθος της εγγραφής ορίζεται με την παράμετρο Len. Για να είναι σύμφωνο το μέγεθος των εγγραφών του αρχείου με το μέγεθος του ορισμένου από τον χρήστη τύπου των δεδομένων, έγινε υπολογισμός του από το μέγεθος της μεταβλητής. Χρησιμοποιήθηκε η συνάρτηση Len(CustomerRec).

Για τον υπολογισμό του μεγέθους της εγγραφής πρέπει να συμβουλευόμαστε τον πίνακα 6-1, στον οποίο αναφέρεται το πλήθος των byte που καταλαμβάνει κάθε τύπος μεταβλητής.

Αφού ανοίξουμε ένα αρχείο για τυχαία προσπέλαση, μπορούμε να γράψουμε δεδομένα σε αυτό ή να διαβάσουμε δεδομένα από αυτό, χρησιμοποιώντας τις εντολές Put # και Get # αντίστοιχα.

Η σύνταξη των εντολών Put # και Get # είναι παρόμοια. Οι γενικές τους μορφές είναι:

Put *#αριθμός_αρχείου, αριθμός_εγγραφής, μεταβλητή*
Get *#αριθμός_αρχείου, αριθμός_εγγραφής, μεταβλητή*

όπου *αριθμός_αρχείου,* ο αριθμός του αρχείου με τον οποίο έχουμε συνδέσει το αρχείο. Επίσης, ο *αριθμός_εγγραφής* είναι ο αριθμός της εγγραφής του αρχείου, στην οποία θα γίνει εγγραφή ή από την οποία θα γίνει ανάγνωση. Οι τιμές που μπορεί να πάρει ο αριθμός_εγγραφής είναι από 1 έως 2^{31} = 2.147.483.647. Τέλος, η μεταβλητή είναι μια *μεταβλητή,* της οποίας ο τύπος είναι ορισμένος από το χρήστη, με τον οποίο περιγράφεται η γραμμογράφηση του αρχείου. Η μεταβλητή αυτή λειτουργεί ως ενδιάμεσος. Πριν από τη διαδικασία της εγγραφής στο αρχείο, στα πεδία αυτής της μεταβλητής γίνονται οι καταχωρίσεις των τιμών των πεδίων. Με την εντολή Put, το περιεχόμενο όλων των πεδίων της μεταβλητής αντιγράφεται στην εγγραφή του αρχείου, που έχει επιλεγεί με τον *αριθμό_εγγραφής*. Επίσης, σ' αυτήν τη μεταβλητή αντιγράφεται το περιεχόμενο της εγγραφής του αρχείου, που διαβάζεται από μια εντολή Get.

Παράδειγμα 24-4.

Σε μια εφαρμογή βιβλιοθήκης, σε κάθε βιβλίο έχει αντιστοιχιστεί ένας κωδικός αριθμός, ο οποίος συμπίπτει με τον αριθμό εγγραφής στο αρχείο που χρησιμοποιείται .

1. Η γραμμογράφηση των εγγραφών περιγράφεται από τις εντολές:

```
Type BookDefinition
    Code    As Long        'Êùäéêüò âéâëßïõ
    Title   As String*30   'Ôßôëïò
    Author  As String*30   'ÓõããñáöÝáò
    ISBN    As String*14   'ÄéåèíÞò ôáîéíïìéêüò áñéèìüò
End Type
```

2. Η δήλωση της μεταβλητής που θα χρησιμοποιηθεί ως ενδιάμεσος γίνεται με την εντολή:

```
Dim BookRec   As BookDefinition
```

3. Το άνοιγμα του αρχείου γίνεται με την εντολή:

```
Open "C:\Books.dat" For Random As 1 Len = Len(BookRec)
```

4. Η ενημέρωση της ενδιάμεσης μεταβλητής και η εγγραφή στο αρχείο γίνεται με τις εντολές:

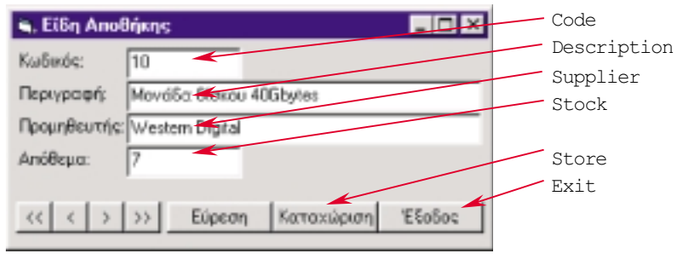
```
BookRec.Code = 45
BookRec.Title = "Ðñïãñáììáôéóìüò ìå VB"
BookRec.Author = "ÌìÜáá óõããñáöÝùí
BookRec.ISBN = "960-531-049-4"
Put #1, BookRec.Code, BookRec
```

Η παράμετρος *αριθμός_εγγραφής* είναι προαιρετική. Αν η παράμετρος παραληφθεί, η ανάγνωση (ή η εγγραφή) θα γίνει από (ή στην) επόμενη εγγραφή από αυτήν που προσπελάσαμε τελευταία. Η VB συγκρατεί έναν εσωτερικό δείκτη (pointer) για κάθε αρχείο, ώστε να γνωρίζει την εγγραφή, που πρέπει να προσπελάσει στη συνέχεια. Αν δεν έχει προηγηθεί καμία προσπέλαση στο αρχείο, ο δείκτης δείχνει την πρώτη του εγγραφή.

Άσκηση 24-1.

Σε μια εφαρμογή ζητείται να αποθηκεύσουμε τα στοιχεία των ανταλλακτικών υπολογιστών σε ένα αρχείο. Να δημιουργηθεί φόρμα και να της επισυναφθεί κώδικας, με τον οποίο θα πραγματοποιείται λειτουργία αποθήκευσης.

1. Έστω, ότι πρόκειται να κρατήσουμε στοιχεία για την περιγραφή του ανταλλακτικού, τον προμηθευτή του και το απόθεμα που διαθέτουμε. Δημιουργούμε μια φόρμα με τα αντίστοιχα πλαίσια κειμένου και ένα επιπλέον πλαίσιο με τον κωδικό του είδους που συμπίπτει με τον αριθμό της εγγραφής που θα γίνει η αποθήκευση.
2. Στη φόρμα τοποθετούμε μια σειρά από πλήκτρα για διάφορες λειτουργίες που πρόκειται να ενεργοποιήσουμε σε επόμενες ασκήσεις.



Εικόνα 24-1. *Φόρμα καταχώρισης στοιχείων ανταλλακτικών*

3. Στην προγραμματιστική μονάδα της φόρμας γράφουμε τον κώδικα:

```
Option Explicit
'ÃñáììïãñÜöçóç åããñáöÞò áñ÷åßïõ áíôáëëáêôéêþí
 Private Type PartDefinition
    Code          As Long             'Êùäéêüò
    Description   As String * 30      'ÐåñéãñáöÞ
    Supplier      As String * 30      'ÐñïìçèåõôÞò
    Stock         As Long             'Áðüèåìá
 End Type

'Êáèïëéêþò ìåôáâëçôÝò
    Dim FileNo    As Integer          'Áñéèìüò áñ÷åßïõ
    Dim RecLen    As Integer          'ÌÞêïò åããñáöÞò
    Dim PartRec   As PartDefinition   'ÁíáéÜìåóç ìåôáâëçôÞ

Private Sub Form_Load()
   'Áíïéîå ôï áñ÷åßï ãéá ôõ÷áßá ðñüóâáóç
    FileNo = FreeFile
    RecLen = Len(PartRec)
    Open "C:\Warehouse\Parts.dat" For Random As FileNo Len = RecLen
End Sub

Private Sub Store_Click()
   'ÁíçìÝñùóå ôá ðåäßá ôçò åíäéÜìåóçò ìåôáâëçôÞò êáé êÜíå ôçí åããñáöÞ
    PartRec.Code = Code
    PartRec.Description = Description
    PartRec. Supplier = Supplier
    PartRec.Stock = Stock
    Put #FileNo, Code, PartRec
   'ÊáèÜñéóå ôá ðëáßóéá êåéìÝíïõ
    Code = ""
    Description = ""
    Supplier = ""
    Stock = ""
End Sub

Private Sub Exit_Click()
   'Êëåßóå ôï áñ÷åßï
    Close
   'Îåöüñôùóå ôç öüñìá
    Unload Me
End Sub
```

Παρατηρούμε ότι στο εσωτερικό μιας προγραμματιστικής μονάδας φόρμας, η δήλωση του τύπου πρέπει να προλογίζεται από την κωδική λέξη `Private`. Στις προγραμματιστικές μονάδες φόρμας δεν είναι δυνατός ο ορισμός καθολικών τύπων.

Με τα αρχεία τυχαίας προσπέλασης μπορούμε να χρησιμοποιήσουμε και τις συναρτήσεις `EOF`, `LOC` και `LOF`. Για να ελέγξουμε σε ποια εγγραφή έγινε η τελευταία προσπέλαση σε ένα ανοιχτό αρχείο χρησιμοποιούμε τη συνάρτηση `LOC`, που δέχεται ως παράμετρο τον αριθμό του αρχείου και επιστρέφει τον αριθμό της εγγραφής που προσπελάστηκε τελευταία. Επίσης, για να ελέγξουμε το μέγεθος του αρχείου χρησιμοποιούμε τη συνάρτηση `LOF`, που δέχεται ως παράμετρο τον αριθμό του αρχείου και επιστρέφει το πλήθος των byte του αρχείου.

Η τιμή που δίνει η συνάρτηση `FileLen` είναι το μέγεθος του αρχείου πριν ανοίξει το αρχείο. Για τα ανοιχτά αρχεία πρέπει να χρησιμοποιούμε τη συνάρτηση `LOF`.

Άσκηση 24-2.

Να γραφεί κώδικας για τα συμβάντα **`Click`** των τεσσάρων πρώτων πλήκτρων της φόρμας (`First`, `Previous`, `Next`, `Last`), της άσκησης 24-1. Το πρώτο πλήκτρο να μας οδηγεί στην πρώτη εγγραφή του αρχείου, το δεύτερο πλήκτρο να μας οδηγεί στην προηγούμενη από αυτήν που βρισκόμαστε, το τρίτο να μας οδηγεί στην επόμενη και το τέταρτο να μας οδηγεί στην τελευταία.

Συμπληρώνουμε την προγραμματιστική μονάδα της φόρμας με τον κώδικα.

```
Private Sub First_Click()
   'ÌåôÜâáóç óôçí ðñþôç åããñáöÞ
    Get #FileNo, 1, PartRec
   'ÁíçìÝñùóç ôçò öüñìáò
    UpDateFormWithRecordFields
End Sub

Private Sub Previous_Click()
   'ÌåôÜâáóç óôçí ðñïçãïýìåíç åããñáöÞ
    If Loc(FileNo) > 1 Then
        Get #FileNo, Loc(FileNo) - 1, PartRec
       'ÁíçìÝñùóç ôçò öüñìáò
        UpDateFormWithRecordFields
    Else
        MsgBox "Âñßóêåóôå óôçí ðñþôç åããñáöÞ", vbExclamation, "Ðñïåéäïðïßçóç"
    End If
End Sub

Private Sub Next_Click()
   'Ìåôáâáóç óôçí åðüìåíç åããñáöÞ
    If Loc(FileNo) < LOF(FileNo) / RecLen Then
        Get #FileNo, , PartRec
           'ÁíçìÝñùóç ôçò öüñìáò
        UpDateFormWithRecordFields
    Else
        MsgBox "Âñßóêåóôå óôçí ôåëåõôáßá åããñáöÞ", vbExclamation, "Ðñïåéäïðïßçóç"
    End If
End Sub

Private Sub Last_Click()
   'Ìåôáâáóç óôçí ôåëåõôáßá åããñáöÞ
    Get #FileNo, LOF(FileNo) / RecLen, PartRec
   'ÁíçìÝñùóç ôçò öüñìáò
    UpDateFormWithRecordFields
End Sub

Private Sub UpDateFormWithRecordFields()
   'ÁíçìÝñùóå ôç öüñìá
    Code = PartRec.Code
    Description = PartRec.Description
    Supplier = PartRec.Supplier
    Stock = PartRec.Stock
End Sub
```

Όποτε γίνεται αλλαγή εγγραφής ενημερώνεται η φόρμα με το περιεχόμενο της εγγραφής. Αν γίνουν αλλαγές στα πλαίσια κειμένου στην οθόνη και πατηθεί το πλήκτρο καταχώριση, οι αλλαγές αυτές θα περάσουν και στην αντίστοιχη εγγραφή του αρχείου.

Άσκηση 24-3.

Να γραφεί κώδικας που να ενεργοποιείται με το συμβάν **`Click`** του πλήκτρου `"Åýñåóç"` (`Find`). Όταν γράφουμε έναν κωδικό στο πλαίσιο κειμένου `Code` και πατάμε το πλήκτρο να αρχίζει η αναζήτηση μέσα στις εγγραφές του αρχείου. Η αναζήτηση να γίνεται σύμφωνα με τον αλγόριθμο της δυαδικής αναζήτησης.

Συμπληρώνουμε την προγραμματιστική μονάδα με τον κώδικα:

```
Private Sub Find_Click()
    Dim Top      As Long
    Dim Bottom   As Long
    Dim Middle   As Long
   'Áñ÷éêÜ üñéá
    Top = 1                             'Ðñþôç åããñáöÞ
    Bottom = LOF(FileNo) / RecLen       'Ôåëåõôáßá åããñáöÞ
   'Ïóï õðÜñ÷åé ðåñéï÷Þ áíáæÞôçóçò
    Do While Top <= Bottom
       Middle = (Top + Bottom) \ 2      'Âñåò ôï ìÝóï ôïõ äéáóôÞìáôïò
       Get #FileNo, Middle, PartRec
       Select Case Val(Code)
            Case Is < PartRec.Code
                 Bottom = Middle - 1    'Ç ôéìÞ óôï êÜôù ìÝñïò ôïõ áñ÷åßïõ
            Case Is > PartRec.Code
                 Top = Middle + 1       'Ç ôéìÞ óôï ðÜíù ìÝñïò ôïõ áñ÷åßïõ
            Case Else
                 Exit Do                'Ç ôéìÞ âñÝèçêå
       End Select
    Loop
   'ÅìöÜíéóç áðïôåëåóìÜôùí
    If Top <= Bottom Then
       UpDateFormWithRecordFields
    Else
       MsgBox "Ç ôéìÞ äå âñÝèçêå", vbExclamation, "ÁíáæÞôçóç"
    End If
End Sub
```

Δυαδικά αρχεία

Στα ***δυαδικά αρχεία (binary files)*** τα δεδομένα αποθηκεύονται όπως ακριβώς αναπαριστόνται στη μνήμη του υπολογιστή. Οι εγγραφές δεν ακολουθούν κάποια συγκεκριμένη δομή και τα πεδία δεν αποτελούν τα θεμελιώδη στοιχεία του αρχείου. Στα δυαδικά αρχεία μπορούμε να αποθηκεύσουμε μεταβλητές τύπων ορισμένων από το χρήστη, οι οποίες έχουν μεταβλητό μέγεθος ή και απλές μεταβλητές. Επίσης, μπορούμε να προσπελάσουμε το περιεχόμενο των δυαδικών αρχείων σε οποιοδήποτε σημείο και μάλιστα byte προς byte. Θα μπορούσε να πει κανείς ότι τα δυαδικά αρχεία δεν έχουν εγγραφές και πεδία, και ότι ο καθένας μπορεί να δει τα δεδομένα από οποιαδήποτε σκοπιά το επιθυμεί. Πρόκειται λοιπόν για τον πιο ευέλικτο τρόπο προσπέλασης, που, όμως, απαιτεί ιδιαίτερη προσοχή από τον προγραμματιστή.
Για να ανοίξουμε ένα αρχείο για δυαδική προσπέλαση χρησιμοποιούμε την εξής σύνταξη της εντολής Open:

```
Open όνομα_αρχείου For Binary As αριθμός_αρχείου
```

Η ανάγνωση και η εγγραφή από τα αρχεία δυαδικής προσπέλασης γίνεται με τις εντολές Get και Put αντίστοιχα.

Παράδειγμα 24-5.

Έστω, ότι στην άσκηση 24-1είχαμε κάνει τις δηλώσεις:

```
Private Type PartDefinition
     Code          As Long         'Êùäéêüò
     Description   As String       'ÐåñéãñáöÞ
     Supplier      As String       'Ðñïìçèåõôþò
     Stock         As Long         'Áðüèåìá
End Type
Dim PartRec  As PartDefinition     'ÅíäéÜìåóç ìåôáâëçôÞ
```

Με αυτόν τον τύπο εγγραφών το αρχείο δεν θα μπορούσε να ανοιχτεί για να προσπελαστεί τυχαία, θα μπορούσε όμως να ανοιχτεί για δυαδική προσπέλαση, με την εντολή:

```
Open "C:\Warehouse\Parts.dat" For Binary As FileNo
```


Η ενημέρωση της ενδιάμεσης μεταβλητής και η εγγραφή στο δυαδικό αρχείο γίνεται με τις εντολές:

```
PartRec.Code = Code
PartRec.Description = Description
PartRec. Supplier = Supplier
PartRec.Stock = Stock
Put #FileNo, , PartRec
```

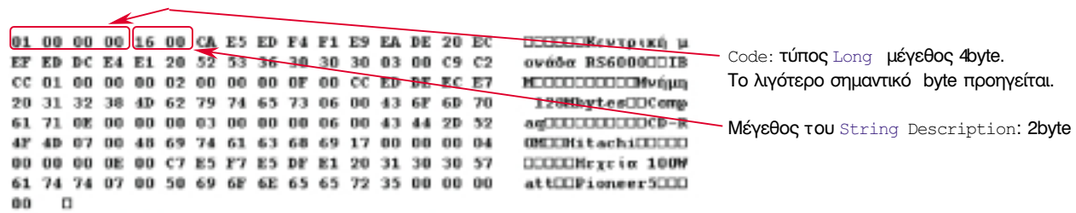
Σε κάθε εγγραφή με την εντολή Put, τα πεδία Description και Supplier θα είχαν διαφορετικό μήκος, ανάλογο του πλήθους των χαρακτήρων που πληκτρολογεί ο χρήστης. Εννοείται, ότι κανένα από τα πλήκτρα της φόρμας που πραγματοποιούν την κίνηση μέσα στις εγγραφές του αρχείου δε θα είχε υπόσταση.

Άσκηση 24-4.

Μια τακτικά επιτελούμενη διαδικασία από τους μηχανικούς συστημάτων είναι η ***αποτύπωση (dump)*** του περιεχομένου ενός αρχείου. Η αποτύπωση είναι μια διαδικασία που αποκαλύπτει όλους τους χαρακτήρες που περιέχει το αρχείο, ορατούς και μη. Κατά την εκτύπωση στην οθόνη ή στον εκτυπωτή, αντί των χαρακτήρων γράφονται οι δεκαεξαδικοί αριθμοί που τους αντιστοιχούν, σύμφωνα με τον κώδικα ASCII. Μια πολύ απλή υπορουτίνα αποτύπωσης είναι η εξής:

```
Sub FileDump()
    Dim b           As Byte     'ÅíäéÜìåóç ìåôáâëçôÞ
    Dim HexPart     As String   'ÄåêáåîáäéêÞ ðåñéï÷Þ
    Dim AsciiPart   As String   'Ðåñéï÷Þ ASCII
   'Áíïéíå ôï áñ÷åßï
    Open "C:\Warehouse\Parts.dat" For Binary As 1
   'Êáèüñéóå ôç ãñáììáôïóåéñÜ ãéá óôïé÷éóìÝíç âêôýðùóç
    Me.Font = "Courier New":   Me.FontSize = 10:   Me.Font.Bold = True
   'Áñ÷éêïðïßçóç ìåôáâëçôþí
    HexPart = "":    AsciiPart = ""
   'ÌÝ÷ñé íá ôåëåéþóåé ôï áñ÷åßï
    Do Until EOF(1)
       'ÄéÜâáóå Ýíá byte
        Get #1, , b
       'Óýíèåóç äåêáåîáäéêïý ôìÞìáôïò êáé ôìÞìáôïò ASCII âêôýðùóçò
        HexPart = HexPart & Right("00" & Hex(b), 2) & " "
        AsciiPart = AsciiPart & Chr(b)
       'ÁíÜ 16 ÷áñáêÞñåò åìÜíéóå ôï ðåñéå÷üìåíï óôçí ïèüíç
        If (Loc(1) Mod 16 = 0) Or (EOF(1)) Then
            Print HexPart; "  "; AsciiPart
            HexPart = "":   AsciiPart = ""
        End If
    Loop
   'Êëåßóå ôï áñ÷åßï
    Close
End Sub
```

Η ενδιάμεση μεταβλητή είναι του τύπου Byte. Έτσι είναι δυνατή η ανάγνωση ενός ενός byte από το αρχείο. Η ανάγνωση των byte γίνεται μέχρι να βρεθεί το τέλος του αρχείου. Ανά 16 byte το περιεχόμενο του αρχείου παρουσιάζεται σε μια γραμμή της οθόνης.



Εικόνα 24-2*. Δείγμα αποτύπωσης περιεχομένου αρχείου*

Ανακεφαλαίωση

Στα αρχεία τυχαίας προσπέλασης οι εγγραφές εντοπίζονται και προσπελαύνονται σύμφωνα με τον αριθμό της εγγραφής που υποδεικνύεται. Η περιγραφή της γραμμογράφησης στην VB γίνεται με τις δηλωτικές εντολές `Type ... End Type` που περιγράφουν τύπους δεδομένων ορισμένους από τον χρήστη. Στα δυαδικά αρχεία τα δεδομένα εισάγονται ακριβώς όπως παριστάνονται στη μνήμη του υπολογιστή. Τα δυαδικά αρχεία αποτελούν τον πιο ευέλικτο τρόπο αποθήκευσης.

Οι εντολές που χρησιμοποιούνται για την προσπέλαση των αρχείων τυχαίας προσπέλασης και των δυαδικών αρχείων είναι η εντολή `Open` για το άνοιμα του αρχείου, η εντολή `Get #` για την εγγραφή δεδομένων, η εντολή `Put #` για την ανάγνωση και η εντολή `Close` για το κλείσιμο του αρχείου. Με τη συνάρτηση `LOF` μπορούμε να ελέγξουμε το μέγεθος του αρχείου και με τη συνάρτηση `LOC` τη θέση στην οποία έχει γίνει η τελευταία ανάγνωση ή εγγραφή.

Εργαστηριακές Ασκήσεις

1. Να εκτελεστεί η άσκηση 24-1. Να εισαχθούν 10 με 15 ανταλλακτικά στο αρχείο, ώστε να μπορούμε να δουλεύουμε με πραγματικά στοιχεία στις επόμενες ασκήσεις.
2. Να εκτελεστεί η άσκηση 24-2. Να σαρωθούν τα στοιχεία μπρος-πίσω. Να γίνουν αλλαγές τιμών στα πλαίσια κειμένου της περιγραφής, του προμηθευτή και του αποθέματος και εκ νέου καταχώριση με αλλαγμένες τιμές.
3. Να εκτελεστεί η άσκηση 24-3. Να γίνουν αναζητήσεις υπαρκτών και ανύπαρκτων ανταλλακτικών.
4. Να εκτελεστεί η άσκηση 24-4.
5. Στη φόρμα της άσκησης 24-4, να προστεθεί αντικείμενο κοινού διαλογικού παραθύρου, ώστε ο χρήστης να έχει τη δυνατότητα να επιλέξει και να παρουσιάσει το περιεχόμενο οπουδήποτε αρχείου

Σημειώσεις:

Μάθημα 25

Η έννοια της βάσης δεδομένων

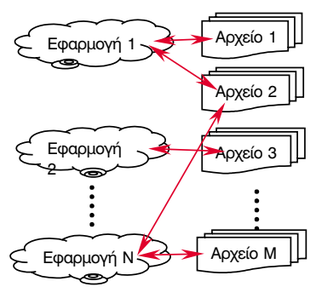
Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να περιγράφουν την έννοια της βάσης δεδομένων.
- Να σχεδιάζουν και να οργανώνουν απλές βάσεις δεδομένων.
- Να επιχειρηματολογούν για τα πλεονεκτήματα των βάσεων δεδομένων.
- Να κατασκευάζουν βάσεις δεδομένων μέσα από το περιβάλλον της VB χρησιμοποιώντας τον οπτικό διαχειριστή δεδομένων.

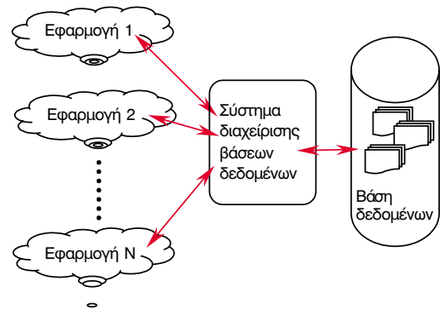
Στα προηγούμενα μαθήματα είδαμε πώς αποθηκεύονται τα δεδομένα σε αρχεία και πώς οι εφαρμογές διαχειρίζονται τα αρχεία για να αντλήσουν τα στοιχεία που είναι αποθηκευμένα σε αυτά. Για αρκετά χρόνια τα πληροφοριακά συστήματα βασίζονταν στην οργάνωση των δεδομένων σε απλά αρχεία. Κατά κανόνα, τα αρχεία δημιουργούνται για συγκεκριμένες εφαρμογές και περιέχουν κοινές πληροφορίες με άλλα αρχεία της ίδιας ή άλλης εφαρμογής. Όμως, η επανάληψη των δεδομένων σε διαφορετικά αρχεία και η διαχείρισή τους από διαφορετικές εφαρμογές προκαλεί προβλήματα. Επανάληψη σημαίνει πλεονασμός και πλεονασμός σημαίνει σπατάλη του πολύτιμου και πεπερασμένου αποθηκευτικού χώρου των δίσκων. Επίσης, η ανεξαρτησία των αρχείων και η μη ύπαρξη κάποιας σύνδεσης μεταξύ των πληροφοριών από μια "εποπτεύουσα" αρχή είναι αιτία πρόκλησης ασυμφωνίας, μη δυνατότητας ορθού συσχετισμού των δεδομένων και εξαγωγής εσφαλμένων πληροφοριών. Ένας χρήστης μιας εφαρμογής μπορεί να ενημερώνει μια πληροφορία σε ένα αρχείο, ενώ ταυτόχρονα ένας άλλος χρήστης άλλης εφαρμογής, να αλλοιώνει "εν αγνοία του" την πληροφορία που εισάγει ο πρώτος ή να εισάγει αντιφατικές πληροφορίες σε σχέση με τις πληροφορίες που εισάγει ένας τρίτος.

Σήμερα, οι περισσότερες εφαρμογές των υπολογιστών στις επιχειρήσεις και στους οργανισμούς επεξεργάζονται δεδομένα, τα οποία είναι αποθηκευμένα σε ***βάσεις δεδομένων (data bases)***. Η διαχείριση των δεδομένων δε γίνεται απ' ευθείας από τις εφαρμογές, αλλά με τη μεσολάβηση ενός ***συστήματος διαχείρισης βάσεων δεδομένων (database management system)***. Το σύστημα διαχείρισης των βάσεων δεδομένων είναι ένα πρόγραμμα που μεσολαβεί μεταξύ των εφαρμογών και των δεδομένων, το οποίο αποτελεί την εποπτεύουσα αρχή, που αναφέραμε πιο πάνω. Το πρόγραμμα αυτό δέχεται τις αιτήσεις των εφαρμογών για τη διαθεσιμότητα και ενημέρωση στοιχείων και φροντίζει, με ενιαίο τρόπο, για την ορθή εισαγωγή τους στις βάσεις δεδομένων, για την ασφάλειά τους, για την ταυτόχρονη διάθεσή τους σε διαφορετικές εφαρμογές κ.ά.

Υπάρχουν πολλά συστήματα διαχείρισης βάσεων δεδομένων, που διαφέρουν ως προς τις επιδόσεις, τους μηχανισμούς διαχείρισης των δεδομένων, τον τρόπο επικοινωνίας με τις εφαρμογές κ.ά. Τα πιο γνωστά είναι η Access, ο SQL Server, η Oracle, η Sybase κ.ά. Στη γλώσσα VB, με καλό σχεδιασμό και μικρή σχετικά προσπάθεια μπορούμε να δημιουργήσουμε προγράμματα που να επικοινωνούν με συστήματα διαχείρισης βάσεων δεδομένων.



Σχήμα 25-1α. *Η διαχείριση των δεδομένων στα αρχεία γίνεται απευθείας από τις εφαρμογές*



Σχήμα 25-1β. *Η διαχείριση των δεδομένων της βάσης δεδομένων γίνεται από το σύστημα διαχείρισης βάσεων*

Για να διαχειριστούμε σε ένα σύστημα μια βάση δεδομένων χρειάζεται να υπάρχει και το αντίστοιχο σύστημα διαχείρισης της βάσης. Στην ειδική περίπτωση που το πρόγραμμά μας είναι γραμμένο σε VB και η βάση δεδομένων είναι του τύπου Access, δε χρειάζεται το σύστημα διαχείρισης της Access. Στην VB υπάρχει ενσωματωμένος ο ***μηχανισμός (engine)*** διαχείρισης των βάσεων αυτού του τύπου και έχει το όνομα ***Jet***.

Τι είναι βάση δεδομένων

Πριν αρχίσουμε να αναπτύσσουμε τη θεωρία των βάσεων δεδομένων θα περιγράψουμε μέσα από ένα παράδειγμα τις χαρακτηριστικές καταστάσεις που επιβάλλουν τη δημιουργία και τη χρήση βάσεων δεδομένων.

Παράδειγμα 25-1.

Έστω, ότι ο καθηγητής των Μαθηματικών συντηρεί ένα αρχείο, στο κεντρικό σύστημα εξυπηρέτησης (server) του σχολείου, με τα στοιχεία και τις βαθμολογίες των μαθητών του. Το αρχείο έχει τη μορφή του πίνακα 25-1.

ΑΜ	Ονοματεπώνυμο	Διεύθυνση	Τηλέφωνο	Α'	Β'	Γ'	Δ'
127	Γαλιώτος Νίκος	Κορίνθου 12, Μπραχάμι 17562	9566485	18	17	18	19
128	Ευθυμίου Γιάννης	Ολυμπίας 22, Μπραχάμι 17562	9563785	17	18	16	18
129	Κούτρας Μάρκος	Σώρου 15, Νέα Σμύρνη 17574	0977 824059	15	14	15	15,
130	Νίκας Κώστας	Γκύζη 2, Νέα Σμύρνη 17574	9563700	16	18	16	16
:	:	:	:	:	:	:	:
144	Σταύρου Άγγελος	. . .	. . .	. . .	. . .	. . .	. . .
145	Σκουρλάς Γιώργος	Θησέως 7, Καλλιθέα 176 73	8912458	13	14	13	12

Πίνακας 25-1.

Έστω, ότι και ο καθηγητής των Φυσικών συντηρεί ένα παρόμοιο αρχείο, στο κεντρικό σύστημα εξυπηρέτησης του σχολείου, με τα στοιχεία και τις βαθμολογίες των μαθητών του, οι οποίοι είναι οι ίδιοι με του καθηγητή των Μαθηματικών. Το περιεχόμενο του αρχείου έχει τη μορφή του πίνακα 25-2.

ΑΜ	Ονοματεπώνυμο	Διεύθυνση	Τηλέφωνο	Α'	Β'	Γ'	Δ'
127	Γαλιώτος Νίκος	Κορίνθου 12, Μπραχάμι 17562	9566485	15	12	14	16
128	Ευθυμίου Γιάννης	Ολυμπίας 22, Μπραχάμι 17562	9563785	17	19	17	16
129	Κούτρας Μάρκος	Σώρου 15, Νέα Σμύρνη 17574	0932 002532	15	16	15	17
130	Νίκας Κώστας	Γκύζη 2, Νέα Σμύρνη 17574	9563700	17	19	17	16
:	:	:	:	:	:	:	:
144	Σταύρου Άγγελος	. . .	. . .	. . .	. . .	. . .	. . .
145	Σκουρλάς Γιώργος	Θησέως 7, Καλλιθέα 176 73	8912458	15	16	17	16

Πίνακας 25-2.

Παρατηρούμε ότι τα τέσσερα πρώτα πεδία, που είναι αυτά με τα περισσότερα δεδομένα, περιέχουν και στα δύο αρχεία ακριβώς τα ίδια στοιχεία. Υπάρχει σπατάλη χώρου από αυτήν την επανάληψη, αν όχι και σπατάλη χρόνου κατά την διπλοεισαγωγή των στοιχείων.

Παρατηρούμε επίσης, ότι ο μαθητής Κούτρας Μάρκος έχει διαφορετικό αριθμό τηλεφώνου στα δύο αρχεία. Αυτό ίσως οφείλεται στο ότι έχασε κάποια στιγμή το κινητό του τηλέφωνο και αγόρασε καινούργιο, αλλά δεν ενημέρωσε έναν από τους δύο καθηγητές. Αν ο διευθυντής του σχολείου ζητήσει από τους καθηγητές να του πουν το τηλέφωνο αυτού του μαθητή, θα πάρει αντιφατικές απαντήσεις.

Αν όμως τα γενικά στοιχεία του μαθητή υπήρχαν σε μια περιοχή (π.χ. αρχείο ή πίνακα, όπως θα αναφέρουμε αργότερα) δεν θα υπήρχε πλεονασμός από την επανάληψή τους σε δύο αρχεία και δεν θα υπήρχε αντίφαση στις εξαγόμενες πληροφορίες, αφού η ενημέρωσή τους θα γινόταν σε μια μόνον περιοχή. Προκύπτει όμως το ερώτημα, πώς θα γινόταν η συσχέτιση μεταξύ των γενικών στοιχείων του μαθητή, τα οποία θα βρίσκονταν σε μια περιοχή, με τους βαθμούς του σε ένα μάθημα που θα βρίσκονταν σε μιαν άλλη;

Η απάντηση είναι σχετικά εύκολη. Πρέπει να χρησιμοποιηθεί ένα μικρού μεγέθους στοιχείο, το οποίο να ***ταυτοποιεί*** τον μαθητή, δηλαδή να τον προσδιορίζει μοναδικά, και το οποίο να χρησιμοποιείται σαν δείκτης μεταξύ των περιοχών.

Στη συγκεκριμένη περίπτωση αυτό το στοιχείο είναι ο αριθμός μητρώου (ΑΜ) του μαθητή που αναφέρεται στην πρώτη στήλη.

Η ταυτοποίηση δεν αποτελεί κάτι καινούργιο και αποκλειστικό του χώρου των υπολογιστών. Για παράδειγμα η ταυτοποίηση ενός ατόμου γίνεται με τον αριθμό ταυτότητάς του. Η ταυτοποίηση ενός φορολογούμενου γίνεται με τον αριθμό του φορολογικού του μητρώου, η ταυτοποίηση ενός ηλεκτρονικού υλικού με τον κωδικό του κ.ά.

Έχουμε λοιπόν τη δομή:

Γενικά στοιχεία μαθητών

ΑΜ	Ονοματεπώνυμο	Διεύθυνση	Τηλέφωνο
127	Γαλιώτος Νίκος	Κορίνθου 12, Μπραχάμι 175 62	9566485
128	Ευθυμίου Γιάννης	Ολυμπίας 22, Μπραχάμι 17562	9563785
:	:	:	:
145	Σκουρλάς Γιώργος	Θησέως 7, Καλλιθέα 176 73	8912458

Βαθμοί στα Μαθηματικά

ΑΜ	Α'	Β'	Γ'	Δ'
127	18	17	18	19
128	17	18	16	18
:	:	:	:	:
145	13	14	13	12

Βαθμοί στα Φυσικά

ΑΜ	Α'	Β'	Γ'	Δ'
127	15	12	14	16
128	17	19	17	16
:	:	:	:	:
145	15	16	17	16

Σχήμα 25-2. *Σύνδεση των βαθμολογιών στα Μαθηματικά και στα Φυσικά με τα στοιχείου των μαθητών με τη βοήθεια του αριθμού μητρώου.*

Στο πρόγραμμα, που παρουσιάζει τις βαθμολογίες των μαθητών σε ένα από τα μαθήματα, θα χρησιμοποιηθεί ως δείκτης ο αριθμός μητρώου του μαθητή για να εντοπιστεί η γραμμή του πίνακα περιέχει τις βαθμολογίες του συγκεκριμένου. Οι δείκτες των πινάκων χρησιμεύουν στην οργάνωση και στον συνταιριασμό των πληροφοριών. Στην περίπτωση που ένας δείκτης λειτουργεί σαν κλειδί για να ανοίγει μια μοναδική περιοχή που βρίσκεται μια πληροφορία, ονομάζεται και ***κλειδί*** (***key***).

Δεν είναι όλοι οι δείκτες πάντα κλειδιά.

Ο τρόπος αποθήκευσης και αναζήτησης των στοιχείων που περιγράψαμε πιο πάνω τυποποιήθηκε και αποτέλεσε τα θεμέλια πάνω στα οποία βασίστηκε η θεωρία των βάσεων δεδομένων. Φυσικά, από τότε που δημιουργήθηκε η πρώτη, απλοϊκή στην αρχή, βάση δεδομένων μέχρι σήμερα έχουν τυποποιηθεί και προστεθεί πολύ περισσότεροι τρόποι δόμησης των δεδομένων και ακόμη περισσότερες λειτουργικές δυνατότητες.

Οι βάσεις δεδομένων που χρησιμοποιούνται κυρίως σήμερα είναι οι ***σχεσιακές βάσεις δεδομένων*** (***relational databases***). Βασικά στοιχεία μιας σχεσιακής βάσης δεδομένων είναι οι ***οντότητες*** (***entities***) και οι ***συσχετίσεις*** (***relationships***) μεταξύ των οντοτήτων. Ως οντότητες θεωρούμε πρόσωπα, αντικείμενα, γεγονότα, πράξεις ή και αφηρημένες έννοιες για τις οποίες θα συγκεντρωθεί και θα αποθηκευθεί πληροφορία.

Έχουν αναπτυχθεί και άλλα μοντέλα βάσεων δεδομένων όπως είναι το ιεραρχικό και το δικτυακό. Αυτά τα μοντέλα χρησιμοποιούνται σήμερα σε μικρότερη έκταση.

Για παράδειγμα, σε ένα σχολείο οι οντότητες είναι οι καθηγητές, οι μαθητές, τα μαθήματα, οι αίθουσες κ.ά. Μεταξύ των μαθημάτων και των καθηγητών υπάρχει συσχέτιση (κάποιοι καθηγητές διδάσκουν κάποια μαθήματα, δε διδάσκουν όλοι οι καθηγητές όλα τα μαθήματα), μεταξύ των μαθημάτων και των αιθουσών υπάρχει συσχέτιση (κάποια μαθήματα διδάσκονται σε συγκεκριμένες αίθουσες, δε διδάσκονται όλα τα μαθήματα σε όλους τους χώρους) μεταξύ των μαθητών και των μαθημάτων υπάρχει συσχέτιση (σε μια σχολική χρονιά δε διδάσκονται σε όλους τους μαθητές όλα τα μαθήματα). Ως δεύτερο παράδειγμα σχεσιακής βάσης δεδομένων αναφέρουμε τη σχολική βιβλιοθήκη. Σε μια σχολική βιβλιοθήκη υπάρχουν οντότητες, όπως είναι οι τίτλοι των βιβλίων, οι συγγραφείς των βιβλίων, τα δανειζόμενα μέλη (μαθητές και καθηγητές). Μεταξύ των συγγραφέων και των τίτλων των βιβλίων υπάρχει συσχέτιση (ένας συγγραφέας μπορεί να έχει γράψει πολλά βιβλία και ένα βιβλίο να έχει γραφεί από πολλούς συγγραφείς), όπως επίσης υπάρχει συσχέτιση μεταξύ βιβλίων και δανειζομένων μελών (ένα μέλος έχει δανειστεί κάποια βιβλία).

Στο εξής, για λόγους συντομίας, θα αναφέρουμε τη σχεσιακή βάση δεδομένων σαν βάση δεδομένων.

Τα μέλη μιας οντότητας μπορούν να αποθηκευτούν σε έναν πίνακα βάσης δεδομένων. Σε αναλογία με τα records και τα πεδία των αρχείων οι γραμμές του πίνακα καλούνται ***εγγραφές*** (***records***) και οι στήλες καλούνται ***πεδία*** (***fields***),. Τις συσχετίσεις τις παριστάνουμε είτε με σχέσεις ανάμεσα στους πίνακες ή και με πίνακες.

Παράδειγμα 25-2.

Στο σχήμα 25-3 δίνονται οι πίνακες και οι σχέσεις μιας βάσης δεδομένων με τα στοιχεία των μαθητών, των μαθημάτων και των βαθμών.

Πίνακας μαθητών

Πεδίο

ΑΜ	Ονοματεπώνυμο	Διεύθυνση	Τηλέφωνο
127	Γαλιώτος Νίκος	Κορίνθου 12, Μπραχάμι 175 62	9566485
128	Κούτρας Μάρκος	Ολυμπίας 22, Νέα Σμύρνη 175 74	0937 004059
:	:	:	:
145	Σκουρλάς Γιώργος	Θησέως 7, Καλλιθέα 176 73	8912458

Εγγραφή

Πίνακας μαθητών

Πεδίο

Κωδικός	Μάθημα	Τάξη
Α1	Μαθηματικά	Α
Α2	Φυσικά	Α
:	:	:
Γ12	Προγραμματισμός	Γ

Πίνακας βαθμών

Πεδίο

Μάθημα	ΑΜ	Α'	Β'	Γ'	Δ'
Α1	127	18	17	18	19
Α2	127	15	12	14	16
Α3	127	19	18	14	18
	:	:	:	:	:
Γ11	145	15	13	14	15
Γ12	145	13	14	13	12

Εγγραφή

Σχήμα 25-3. *Ο πίνακας των βαθμολογιών έχει συσχέτιση με τους μαθητές και τα μαθήματα.*

Στο παράδειγμά μας, κάθε εγγραφή μαθητή περιλαμβάνει τα στοιχεία για έναν μόνο μαθητή. Επίσης, στον πίνακα βαθμών, κάθε εγγραφή περιλαμβάνει τους βαθμούς ενός μόνο μαθητή σε ένα μόνο μάθημα.

Κάθε εγγραφή αποτελείται από πολλά πεδία. Κάθε πεδίο περιέχει ένα τμήμα πληροφορίας για την εγγραφή. Για παράδειγμα, η εγγραφή του μαθητή περιλαμβάνει τα πεδία: αριθμός μητρώου, ονοματεπώνυμο, διεύθυνση, τηλέφωνο. Η εγγραφή του πίνακα βαθμών περιλαμβάνει τα πεδία: κωδικός μαθήματος, αριθμός μητρώου μαθητή, βαθμός α' στο μάθημα, βαθμός β' ...

Το κλειδί ενός πίνακα μπορεί να είναι ένα πεδίο ή ένας συνδυασμός πεδίων που έχει ***μοναδική τιμή***. Ο αριθμός μητρώου του μαθητή είναι μοναδικός για κάθε μαθητή και αποτελεί κλειδί στον πίνακα των μαθητών. Επίσης, ο κωδικός μαθήματος είναι μοναδικός για κάθε μάθημα στον πίνακα των μαθημάτων. Τέλος, ο συνδυασμός κωδικού μαθήματος και αριθμού μητρώου μαθητή είναι κλειδί στον πίνακα των βαθμών.

Η οργάνωση της βάσης δεδομένων είναι πιο γενική από την οργάνωση που είχαμε προτείνει στο σχήμα 25-2. Δεν χρειάζεται ένας πίνακας για τη βαθμολογία σε κάθε μάθημα. Εισαγωγές νέων μαθημάτων υλοποιούνται με την εισαγωγή μιας γραμμής στον πίνακα των μαθημάτων και όχι με τη δημιουργία ενός πίνακα. Αποτελεί λοιπόν σημαντικό μέρος του σχεδιασμού, ο τρόπος με τον οποίο θα δομηθούν τα δεδομένα. Μια πρόχειρα σχεδιασμένη βάση δεδομένων μπορεί να δημιουργήσει πολλά προβλήματα στην κατασκευή ενός αξιόπιστου προγράμματος, ενώ αντίθετα η καλή σχεδίαση διευκολύνει τη ζωή του προγραμματιστή.

Η σχεδίαση μιας βάσης δεδομένων

Η δημιουργία της βάσης δεδομένων αρχίζει με την καλή σχεδίαση. Αυτό αποτελεί μια βασική αρχή σε όλες τις περιπτώσεις τεχνολογικών εφαρμογών. Ακόμη και απο την καθημερινή μας εμπειρία γνωρίζουμε ότι δεν είναι δυνατόν να ξεκινήσουμε την κατασκευή ενός οικοδομήματος χωρίς σχέδιο. Η σχεδίαση μιας βάσης δεδομένων γίνεται απλά και γρήγορα αν ακολουθήσουμε κάποια τυποποιημένα βήματα.

Συγκεκριμένα:

1. ***Καθορίζουμε το σκοπό της βάσης δεδομένων***. Είναι σημαντικό να ξέρουμε από την αρχή ποιος είναι ο τελικός στόχος, ώστε να εντοπίσουμε σωστά τα συστατικά της βάσης και να τους ενσωματώσουμε τις κατάλληλες δυνατότητες. Για παράδειγμα, ο σκοπός της βάσης που περιγράψαμε πιο πάνω είναι η οργάνωση του μαθητολογίου του σχολείου.
   Εργαζόμενοι για τον προσδιορισμό και τον καθορισμό των εργασιών που θα πρέπει να διεκπεραιώνει η εφαρμογή δημιουργούμε ένα έγγραφο ***λειτουργικών προδιαγραφών*** (***functional specifications***) της εφαρμογής . Το έγγραφο αυτό είναι χρήσιμο όχι μόνο σε μας, ως έγγραφο αναφοράς, αλλά και απαραίτητο στοιχείο της εγγράφου συμφωνίας με το πρόσωπο που μας έχει ζητήσει να δημιουργήσουμε την εφαρμογή.
2. ***Εντοπίζουμε τις οντότητες***. Ο εντοπισμός των οντοτήτων αποτελεί τη βάση του οικοδομήματος μιας βάσης δεδομένων. Από τις οντότητες, όπως έχουμε δει, θα δημιουργηθούν οι πίνακες μέσα στους οποίους θα τοποθετηθούν τα δεδομένα. Για παράδειγμα, τα μαθήματα, οι μαθητές, οι βαθμολογίες, οι καθηγητές, οι αίθουσες, οι σχολικές περίοδοι, τα εποπτικά όργανα κλπ.
   Μερικές φορές οι οντότητες είναι εμφανείς, όπως στο πιο πάνω παράδειγμα. Υπάρχουν όμως περιπτώσεις που είναι δυσδιάκριτες, και αυτό ειδικά στις περιπτώσεις που οι οντότητες αποτελούν υλοποιήσεις συσχετίσεων.
3. ***Καθορίζουμε τα χαρακτηριστικά των οντοτήτων***. Τα χαρακτηριστικά των οντοτήτων θα μας καθορίσουν τα πεδία και τους τύπους των δεδομένων που θα αποθηκευθούν σε κάθε πίνακα. Για παράδειγμα, στον πίνακα των μαθητών θα αποθηκευθεί το ονοματεπώνυμο, η διεύθυνση, το τηλέφωνο κ.λ.π., στον πίνακα των βαθμών θα αποθηκευτούν οι βαθμοί. Το ονοματεπώνυμο είναι αλφαβητικό στοιχείο, ενώ ο βαθμός είναι αριθμητικό στοιχείο.
   Σ' αυτό το σημείο πρέπει να τονίσουμε, ότι ή επιλογή του σωστού τύπου των δεδομένων παίζει σημαντικό ρόλο στη βιωσιμότητα της εφαρμογής. Κλασικά παραδείγματα έχουμε στην περίπτωση του προβλήματος του 2000 και του Ευρώ. Το πρόβλημα του 2000 προέκυψε διότι οι προγραμματιστές χρησιμοποιούσαν δύο μόνο ψηφία για να παραστήσουν το έτος. Έτσι, υπήρχε περίπτωση κάποιες βάσεις δεδομένων να εκλάβουν την παράσταση 00 σαν έτος 1900 και όχι ως 2000, την παράσταση 01 σαν έτος 1901 και όχι σαν 2001 κοκ. Επίσης, το πρόβλημα του Ευρώ προέκυψε διότι κάποιοι προγραμματιστές είχαν ορίσει ακεραίου τύπου πεδία για τις τιμές των προϊόντων και των υπηρεσιών, αφού δεν υπήρχαν κλάσματα της δραχμής. Με την εισαγωγή του Ευρώ χρειάστηκε να γίνουν αλλαγές στους χρησιμοποιούμενους τύπους δεδομένων.
4. ***Καθορίζουμε τις συσχετίσεις***. Προσδιορίζουμε τα πεδία κλειδιά σε κάποιους βασικούς πίνακες και τα πεδία που θα συσχετιστούν με τα πεδία κλειδιά στους συσχετιζόμενους με τους βασικούς πίνακες. Υπάρχουν περιπτώσεις που μπορεί να χρειαστεί να προστεθούν πεδία σε έναν πίνακα για να είναι δυνατή η συσχέτισή του με έναν άλλο. Για παράδειγμα, το πεδίο κωδικός μαθήματος στον πίνακα των βαθμών. Ακόμη, υπάρχει περίπτωση να χρειαστεί να δημιουργηθούν πίνακες που να λειτουργούν σαν συνδετικοί κρίκοι (πίνακες συσχέτισης) μεταξύ δύο άλλων πινάκων.
5. ***Αναθεωρούμε και βελτιώνουμε κάθε προηγούμενο σχεδιασμό***. Αφού δημιουργήσουμε τους πίνακες εισάγουμε δοκιμαστικά δεδομένα προς έλεγχο. Μόνον κατ' αυτόν τον τρόπο είναι δυνατόν να ανιχνευθούν τα πιθανά λάθη. Αν οι εγγραφές του πίνακα αφήνουν κάποια πεδία χωρίς τιμή υπάρχει περίπτωση να χρειάζεται να χωριστεί ο πίνακας σε δύο συσχετιζόμενους πίνακες. Επίσης, αν υπάρχουν επαναλαμβανόμενες τιμές σε πεδία, επιβάλλεται να χωριστεί ο πίνακας σε δύο συσχετιζόμενους πίνακες κ.ά.

Πλεονεκτήματα των βάσεων δεδομένων

Ο πιο βασικός λόγος χρησιμοποίησης βάσεων δεδομένων σε μία επιχείρηση ή ένα οργανισμό είναι το γεγονός ότι ένα τέτοιο σύστημα οργάνωσης και ανάκλησης δεδομένων, "οικοδομείται" με την εποπτεία και την καθοδήγηση του οργανισμού, ώστε να καλύψει τις ανάγκες του συνόλου των τμημάτων και να προσφέρει στον οργανισμό ή στην επιχείρηση τα εξής πλεονεκτήματα:

Κεντρικό έλεγχο (***Centralized control***). Ο σχεδιασμός των εφαρμογών γίνεται κεντρικά απο ειδικούς. Ο κάθε προγραμματιστής δε δημιουργεί τα δικά του αρχεία, τα οποία μόνον αυτός γνωρίζει που βρίσκονται και τι ακριβώς περιέχουν τα πεδία τους.

Μικρό πλεονασμό. Κύριο χαρακτηριστικό μιας βάσης είναι ότι τα δεδομένα της είναι οργανωμένα και αποθηκευμένα σε συσχετιζόμενους πίνακες χωρίς περιττές επαναλήψεις.

Συμβατότητα (***Consistency***). Η ελάττωση του πλεονασμού μειώνει και την πιθανότητα δημιουργίας ασυμβίβαστων δεδομένων και την εξαγωγή αντιφατικών πληροφοριών. Για παράδειγμα, αν τα γενικά στοιχεία ενός προσώπου υπάρχουν καταχωρισμένα μια μόνο φορά μέσα σε έναν πίνακα γενικών στοιχείων προσώπων, η αλλαγή ενός στοιχείου του συγκεκριμένου προσώπου, π.χ. της διεύθυνσής του, πρέπει να γίνει σε ένα μόνο πίνακα. Αν η διεύθυνση υπήρχε σε περισσότερους από έναν πίνακες, θα έπρεπε να γνωρίζουμε σε ποιους και σε ποια πεδία θα γίνει η αλλαγή. Στην αντίθετη περίπτωση, αν παραλείπαμε την ενημέρωση σε έναν πίνακα, η εξαγωγή της διεύθυνσης από αυτόν θα έδινε αντιφατική πληροφορία από την εξαγωγή της διεύθυνσης από τους υπόλοιπους.

Ανεξαρτησία (***Independence***). Ο όρος σχετίζεται με τα προγράμματα που χρησιμοποιούν τη βάση. Σε περίπτωση που χρειαστεί να τροποιηθεί ο τρόπος οργάνωσης, προσπέλασης και περιγραφής των δεδομένων δεν απαιτείται να γίνουν αλλαγές στα προγράμματα των εφαρμογών. Για παράδειγμα, η αλλαγή του μεγέθους ενός πεδίου δεν επιβάλλει την αλλαγή των προγραμμάτων που το χρησιμοποιούν.

Ακεραιότητα (***Integrity***). Ο όρος σχετίζεται με τα δεδομένα της βάσης. Ακέραια θεωρούνται τα δεδομένα των οποίων δεν έχουν χαθεί οι αναφορές. Τα συστήματα διαχείρισης βάσεων δεδομένων περιέχουν εσωτερικό μηχανισμό, ο οποίος ελέγχει και αποτρέπει τέτοιας μορφής "παράλογες" απαιτήσεις, όταν μια εφαρμογή ζητήσει να διαγράψει κάποια στοιχεία τα οποία αναφέρονται σε άλλα σχετικά στοιχεία. Για παράδειγμα, στην περίπτωση του μαθητολογίου δεν επιτρέπεται να διαγραφτεί ένας μαθητής από τον πίνακα των μαθητών, αν υπάρχουν βαθμοί στους πίνακες βαθμολογίας.

Ασφάλεια (***Security***). Το σύστημα διαχείρισης των βάσεων δεδομένων είναι επιφορτισμένο και με τον έλεγχο των εξουσιοδοτήσεων των χρηστών. Ένας χρήστης δεν μπορεί να προσπελάσει κάποια μορφής πληροφορία, αν δεν του επιτρέπεται.

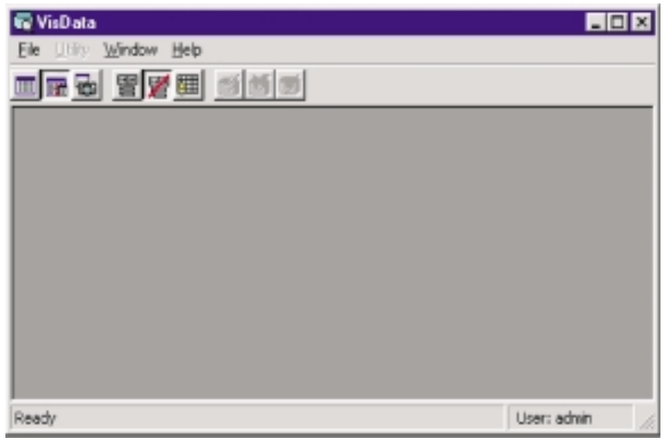
Υιοθέτηση προτύπων (***Standarization***). Τα πρότυπα οργάνωσης είναι ήδη προκαθορισμένα από τους κανόνες δημιουργίας και διαχείρισης της βάσης δεδομένων. Επίσης, οι κατασκευαστές συστημάτων διαχείρισης βάσεων δεδομένων έχουν αναπτύξει και μηχανισμούς, ώστε να υπάρχει η δυνατότητα μετατροπής ή χρήσης δεδομένων από άλλου τύπου βάσεις. Έτσι, η μεταφορά στοιχείων γίνεται ευκολότερα μεταξύ εταιρειών ή οργανισμών, που χρησιμοποιούν του ίδιου τύπου βάσεις ή ακόμη και εταιρειών που χρησιμοποιούν βάσεις διαφορετικού τύπου.

Ευκολία ανάπτυξης εφαρμογών. Κάθε νέα εφαρμογή μπορεί να βασιστεί πάνω σε ήδη υπάρχουσες δομές δεδομένων μέσα στις βάσεις και να εκμεταλλευτεί όλες τις δυνατότητες των συστημάτων διαχείρισης βάσεων δεδομένων (π.χ. ασφάλεια, ακεραιότητα δεδομένων, συμβιβαστότητα) χωρίς να απαιτείται ιδιαίτερη πρόνοια από τον προγραμματιστή.

Η υλοποίηση μιας βάσης δεδομένων από το περιβάλλον της VB

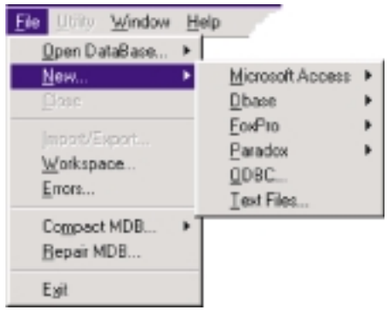
Η VB προσφέρει έναν πολύ απλό εργαλείο δημιουργίας και διαχείρισης βάσεων δεδομένων. Μέσα από το περιβάλλον εργασίας της VB μπορούμε να καλέσουμε μια εφαρμογή, τον ***οπτικό διαχειριστή δεδομένων*** (***visual data manager***), που με διαλογικό τρόπο μας βοηθά στη δημιουργία πινάκων, τον ορισμό πεδίων και γενικά στη διαμόρφωση βάσεων δεδομένων.

Από τη γραμμή μενού επιλέγουμε την επιλογή **Add-Ins** και στη συνέχεια την υποεπιλογή **Visual Data Manager**. Στην οθόνη εμφανίζεται το παράθυρο της εφαρμογής με τίτλο VisData. Ο οπτικός διαχειριστής δεδομένων συνεργάζεται με διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων. Για παράδειγμα, με την Access, την Dbase, την FoxPro, την Paradox και μπορεί να επικοινωνήσει μέσω του μηχανισμού ***ODBC*** (***Open Data Base Connectivity***) με τον SQL Server, την Oracle, τη Sybase κά.



Εικόνα 25-1. *Το παράθυρο του οπτικού διαχειριστή δεδομένων αμέσως μετά την κλήση του.*

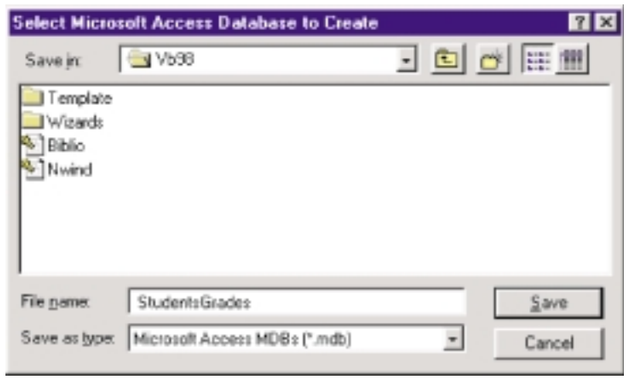
Αρχικά, πρέπει να καθορίσουμε το σύστημα διαχείρισης της βάσης δεδομένων που θα χρησιμοποιήσουμε καθώς και τον αποθηκευτικό χώρο στον οποίο θα τοποθετηθεί η βάση. Στο παράθυρο του οπτικού διαχειριστή δεδομένων επιλέγουμε το μενού **File**, μετά την επιλογή **New** και από το πτυσσόμενο μενού το σύστημα διαχείρισης βάσης δεδομένων.



Εικόνα 25 -2. *Διαδοχικές επιλογές στο μενού του οπτικού διαχειριστή δεδομένων και η λίστα με τους τύπους των συστημάτων βάσεων δεδομένων που υποστηρίζει.*

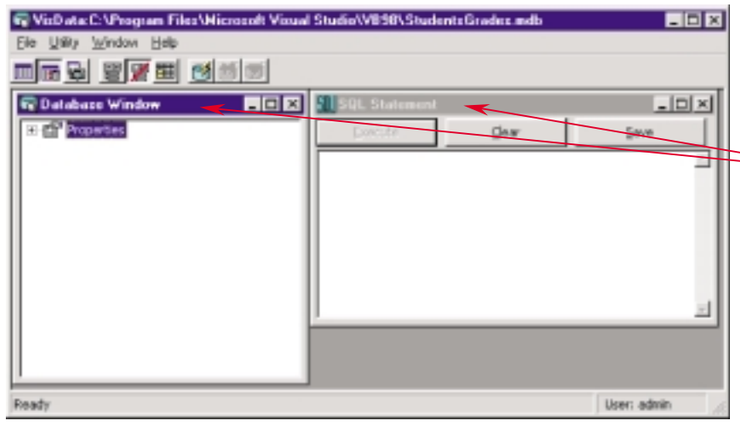
Δημιουργία του αρχείου της βάσης δεδομένων

Στην περίπτωση που θέλουμε να δημιουργήσουμε μια βάση δεδομένων τύπου Access, επιλέγουμε την πρώτη υποεπιλογή, οπότε εμφανίζεται το διαλογικό παράθυρο της εικόνας 25-3. Από αυτό το παράθυρο καθορίζουμε το φάκελο, στον οποίο θα τοποθετηθεί το αρχείο που θα περιέχει τη βάση δεδομένων. Ας σημειωθεί, ότι όλοι οι πίνακες μιας βάσης δεδομένων Access τοποθετούνται σε ένα μόνο αρχείο. Στη συγκεκριμένη περίπτωση στο αρχείο δίνουμε το όνομα `StudentsGrades.mdb`.



Εικόνα 25-3. *Διαλογικό παράθυρο για τη δημιουργία και την τοποθέτηση του αρχείου της βάσης δεδομένων στο φάκελο της επιλογής μας.*

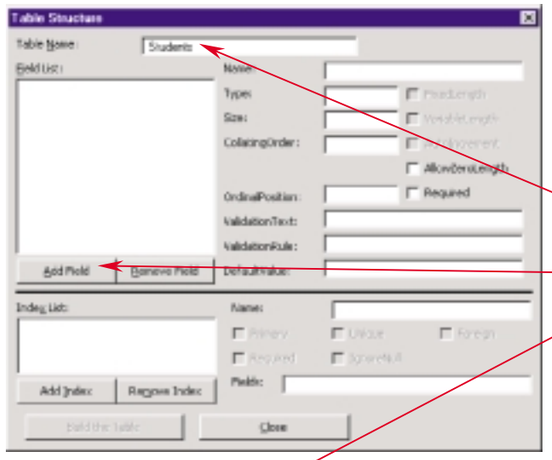
Το παράθυρο του οπτικού διαχειριστή δεδομένων διαμορφώνεται, όπως φαίνεται στην εικόνα 25-4. Το αριστερό παράθυρο, το **Database Window**, είναι το παράθυρο στο οποίο θα εμφανίζονται σε μορφή δένδρου τα στοιχεία της βάσης δεδομένων.



Εικόνα 25-4. *Στο εσωτερικό του οπτικού διαχειριστή δεδομένων μετά τη δημιουργία ή την επιλογή μιας βάσης δεδομένων εμφανίζονται δύο παράθυρα.*

Προσθήκη νέου πίνακα

Για να προσθέσουμε έναν πίνακα στη βάση δεδομένων κάνουμε δεξί κλικ οπουδήποτε πάνω στο παράθυρο **Database Window** και από το αναδυόμενο μενού που θα παρουσιαστεί επιλέγουμε **New Table**. Τότε εμφανίζεται το διαλογικό παράθυρο **Table Structure**, οπότε:

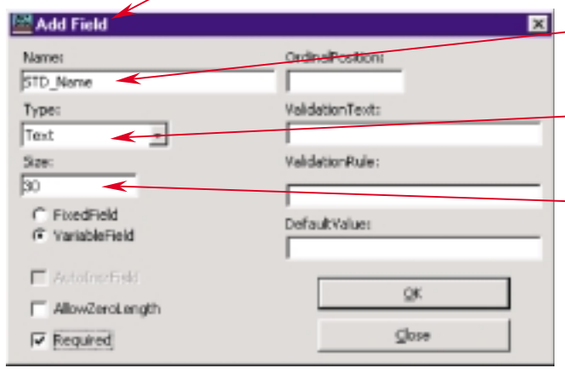


1. Στο πλαίσιο κειμένου **Table Name** πληκτρολογούμε το όνομα του πίνακα.
2. Πατάμε το πλήκτρο **Add Fields** για να προσθέσουμε νέα πεδία. Εμφανίζεται το διαλογικό παράθυρο **Add Fields**.

Εικόνα 25-5. *Το διαλογικό παράθυρο* ***Table Structure***

Προσθήκη νέου πεδίου

Συνεχίζουμε την περιγραφή του πίνακα περιγράφοντας τις χαρακτηριστικές ιδιότητες του πεδίου.



3. Στο πλαίσιο κειμένου **Name** πληκτρολογούμε το όνομα του πρώτου πεδίου, έστω STD_Name.
4. Επιλέγουμε τον τύπο δεδομένων του πεδίου απο τη πτυσσόμενη λίστα **Type**.
5. Πληκτρολογούμε το μέγεθος του πεδίου, αν έχουμε δηλώσει ότι ο τύπος του είναι αλφαριθμητικός (Text).

Εικόνα 25-6. *Το διαλογικό παράθυρο* ***Add Fields***

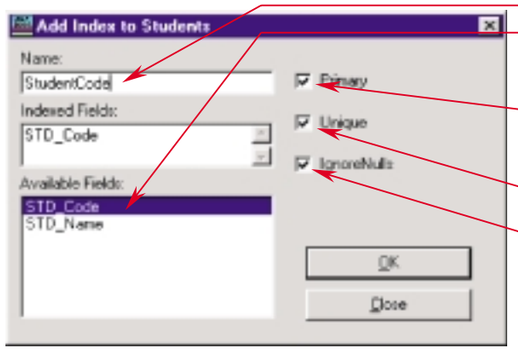
Κάποια από τα χαρακτηριστικά των πεδίων δεν είναι δυνατόν να αλλάξουν. Υπάρχει περίπτωση να χρειαστεί να διαγραφεί ένα πεδίο και να ξαναδημιουργηθεί.

Με παρόμοιο τρόπο προσθέτουμε και τα υπόλοιπα πεδία του πίνακα. Κάθε φορά που δημιουργείται ένα πεδίο, προστίθεται το όνομά του στη λίστα **Field List** του διαλογικού

παραθύρου **Table Structure**. Όταν περιγράψουμε όλα τα πεδία πατάμε το πλήκτρο **Build the Table** του διαλογικού παραθύρου **Table Structure** για να δημιουργηθεί ο πίνακας.

Για να αλλάξουμε τα χαρακτηριστικά ενός πεδίου κάνουμε διπλό κλικ πάνω στο όνομά του στο παράθυρο **Database Window**. Τότε μόνο παρουσιάζονται τα χαρακτηριστικά του και μπορούμε να επιλέξουμε με διπλό κλικ το χαρακτηριστικό, που θέλουμε να μεταβάλλουμε. Στο διαλογικό παράθυρο που θα εμφανιστεί, κάνουμε τις αλλαγές. Για να διαγράψουμε ένα πεδίο κάνουμε δεξί κλικ πάνω στο όνομά του στο παράθυρο **Database Window** και από το αναδυόμενο μενού επιλέγουμε **Delete**.

Ο ορισμός κάποιων πεδίων ως δεικτών ή ακόμη καλύτερα ως κλειδιών γίνεται από το παράθυρο **Field List** πατώντας το πλήκτρο **Add Index**. Τότε εμφανίζεται το διαλογικό παράθυρο **Add Index**.



1. Πληκτρολογούμε το όνομα του δείκτη.
2. Με διπλό κλικ επιλέγουμε τα πεδία του πίνακα από τα οποία θα συντεθεί ο δείκτης, με τη σειρά που θέλουμε να διαταχθούν.
3. Καθορίζουμε αν ο δείκτης θα είναι ***πρωτεύον κλειδί***. Κάθε πίνακας επιτρέπεται να έχει μόνον ένα πρωτεύον κλειδί.
4. Καθορίζουμε αν κάθε τιμή του δείκτη θα μοναδική.
5. Καθορίζουμε αν επιτρέπεται να υπάρχουν εγγραφές που έχουν κενό το συγκεκριμένο πεδίο.

Εικόνα 25-7. *Διαδικασία ορισμού δείκτη*

Άσκηση 25-1.

Με τη βοήθεια του οπτικού διαχειριστή δεδομένων να δημιουργηθεί η βάση δεδομένων StudentsGrades, στην οποία να τοποθετηθούν τα στοιχεία των μαθητών, των μαθημάτων και των βαθμών σύμφωνα με όσα αναλύονται στο παράδειγμα 25-2.

1. Δημιουργούμε τους πίνακες Students, Lessons, Grades για τους μαθητές, τα μαθήματα και τους βαθμούς αντίστοιχα.

Students

Πεδίο	Τύπος
STD_Code	Integer
STD_Name	Text 30 χαρ.
STD_Address	Text 60 χαρ.
STD_Phone	Text 30 χαρ.

Lessons

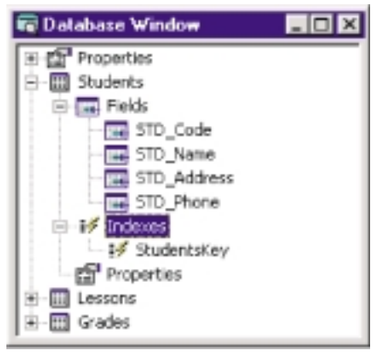
Πεδίο	Τύπος
LES_Code	Text 2 χαρ.
LES_Name	Text 30 χαρ.
LES_Class	Text 1 χαρ.

Grades

Πεδίο	Τύπος
GRD_STD_Code	Integer
GRD_LES_Code	Text 2 χαρ.
GRD_AGrade	Single
GRD_BGrade	Single
GRD_CGrade	Single
GRD_DGrade	Single

Πίνακες 25-3. *Τα ονόματα των πεδίων, για λόγους εύκολης απομνημόνευσης και για λόγους διάκρισης από τα ονόματα άλλων ποσοτήτων, τα σχηματίζουμε από τρία κεφαλαία γράμματα, αντιπροσωπευτικά του πίνακα, το χαρακτήρα της υπογράμμισης και το λεκτικό που διακρίνει το πεδίο από τα υπόλοιπα πεδία.*

2. Ορίζουμε ως πρωτεύοντα κλειδιά: το πεδίο STD_Code του πίνακα Students, το πεδίο LES_Code του πίνακα Lessons και τον συνδυασμό πεδίων GRD_STD_Code και GRD_LES_Code του πίνακα Grades, δίνοντας τα ονόματα StudentsKey, LessonsKey και GradeKey αντίστοιχα.



Εικόνα 25-8. *Το παράθυρο Database Window μετά τη δημιουργία των πινάκων και με αναπτυγμένη την ιεραρχία των πεδίων και των δεικτών*

Ανακεφαλαίωση

Είναι προτιμότερο, τα δεδομένα μιας εφαρμογής αντί να αποθηκεύονται σε αρχεία σειριακά ή τυχαίας προσπέλασης να αποθηκεύονται σε πίνακες βάσης δεδομένων, οι οποίοι σχετίζονται μεταξύ τους. Η οργάνωση των δεδομένων σε βάση δεδομένων πρέπει να γίνεται κατά τρόπο που να διευκολύνει την αναζήτηση και την τροποποίηση της αποθηκευμένης πληροφορίας.
Αφού σχεδιάσουμε μια βάση δεδομένων μπορούμε να την κατασκευάσουμε στο περιβάλλον εργασίας χρησιμοποιώντας την εφαρμογή **Visual Data Manager**, που συνοδεύει το περιβάλλον της VB. Με τη βοήθεια αυτής της εφαρμογής γίνεται η περιγραφή της βάσης πίνακα προς πίνακα, πεδίο προς πεδίο και δείκτη προς δείκτη.

Εργαστηριακές ασκήσεις

1. Να σχεδιάσετε τη δομή μιας βάσης δεδομένων που να έχει τη δυνατότητα να εξυπηρετήσει μια σχολική βιβλιοθήκη. Οι οντότητες που μετέχουν είναι: συγγραφέας, εκδοτικός οίκος, τίτλος βιβλίου, αντίτυπο, δανειζόμενος. Ποιους δείκτες θα ορίσετε;
2. Σχεδιάστε μια βάση δεδομένων με τις των οδούς της Ελλάδας. Η βάση δεδομένων να είναι οργανωμένη σε γεωγραφικά διαμερίσματα, νομούς, πόλεις, συνοικίες, οδούς.
3. Να εκτελέσετε την άσκηση 25-1. Στη συνέχεια να συμπληρώσετε τους πίνακες με στοιχεία.
   Σημείωση: Η βάση δεδομένων StudentsGrades θα αποτελέσει την πηγή άντλησης δεδομένων στα επόμενα δύο μαθήματα. Συμπληρώστε λοιπόν τους πίνακες Students και Lessons με περισσότερες από 5 εγγραφές και τον πίνακα Grades με 15 τουλάχιστον εγγραφές. Ειδικά ο πίνακας Grades να περιέχει εγγραφές, στις οποίες να συνδυάζονται πολλοί μαθητές και πολλά μαθήματα.

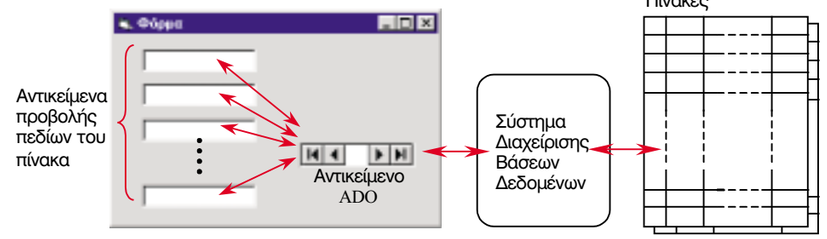
Μάθημα 26
Σύνδεση φόρμας με βάση δεδομένων

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να περιγράφουν τον τρόπο σύνδεσης μιας φόρμας με μια βάση δεδομένων.
- Να χρησιμοποιούν το μηχανισμό του αντικειμένου ελέγχου ADO για να προσπελαύνουν και να εμφανίζουν το περιεχόμενο των εγγραφών μιας βάσης δεδομένων.
- Να χρησιμοποιούν την τεχνολογία ADO για να προσθέτουν, να αλλάζουν, να διαγράφουν και να αναζητούν εγγραφές.

Στο προηγούμενο μάθημα είδαμε πώς μέσα από το περιβάλλον της VB, με τη βοήθεια του οπτικού διαχειριστή δεδομένων, μπορούμε να δημιουργήσουμε βάσεις δεδομένων. Σε αυτό το μάθημα θα δούμε πώς μπορούμε να προσπελάσουμε και να εκμεταλλευτούμε τα στοιχεία των βάσεων δεδομένων μέσα από φόρμες και προγράμματα VB και θα διαπιστώσουμε πώς με πολύ μικρή προγραμματιστική προσπάθεια είναι δυνατόν να δημιουργηθούν φόρμες εισαγωγής και διαχείρισης εγγραφών σε μια βάση δεδομένων.

Η κατασκευή και διαχείριση φορμών που εκμεταλλεύονται το περιεχόμενο των βάσεων δεδομένων βασίζεται σε ένα αντικείμενο ελέγχου τύπου ActiveX, το οποίο χρησιμοποιεί μια τεχνολογία, που είναι γνωστή ως ***ADO*** (***ActiveX Data Object***). Το αντικείμενο αυτό τοποθετείται πάνω στη φόρμα και προσφέρει έναν οπτικό μηχανισμό στο χρήστη για την προσπέλαση των εγγραφών ενός πίνακα της βάσης. Επίσης, παίζει το ρόλο του συνδετικού κρίκου μεταξύ των πεδίων του πίνακα, που θέλουμε να παρουσιάζονται στο χρήστη, και των αντικειμένων της φόρμας (πλαισίων κειμένου, πλήκτρων σημείωσης, συνδυασμένων λιστών κ.ά.) από τα οποία γίνεται η προβολή τους.



Σχήμα 26-1. *Το αντικείμενο ADO μεσολαβεί στην επικοινωνία με το σύστημα διαχείρισης βάσεων δεδομένων και μεταφέρει εγγραφές από έναν πίνακα στη φόρμα. Τα περιεχόμενα των πεδίων των εγγραφών μπορούν να εμφανιστούν σε αντικείμενα ελέγχου της οθόνης.*

Το αντικείμενο ελέγχου ADO

Είδαμε, ότι η σύνδεση μιας φόρμας, μέσω του μηχανισμού ADO, με μια βάση δεδομένων γίνεται με τη βοήθεια ενός αντικειμένου ελέγχου τύπου ActiveX. Το αντικείμενο αυτό είναι το ***ADO Data control***. Το εργαλείο με το οποίο δημιουργούνται τα αντικείμενα τύπου ADO δεν υπάρχει εξ αρχής στην εργαλειοθήκη του περιβάλλοντος εργασίας. Για να το προσθέσουμε στην εργαλειοθήκη κάνουμε τα εξής:

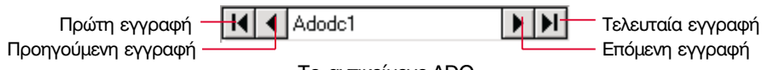
1. Επιλέγουμε **Project ¦ Components** από τη γραμμή μενού.
2. Επιλέγουμε τον καρτελοδείκτη **Controls**.
3. Από το πινάκιο με τις ομάδες των εργαλείων (δείτε εικόνα 13-7) επιλέγουμε το εργαλείο ***Microsoft ADO Data Control***.

Το αντικείμενο ADO τοποθετείται πάνω στις φόρμες, με τον ίδιο ακριβώς τρόπο που τοποθετούνται όλα τα αντικείμενα και είναι ορατό. Αποτελείται από μια περιοχή κειμένου, στην οποία ο προγραμματιστής μπορεί να εμφανίσει οποιοδήποτε λεκτικό, και μια σειρά πλήκτρα με τα οποία ο χρήστης μπορεί να κινηθεί στην επόμενη ή την προηγούμενη εγγραφή της

Το εργαλείο Data που υπάρχει στην εργαλειοθήκη είναι εργαλείο σύνδεσης με βάσεις δεδομένων παλαιότερης τεχνολογίας, της τεχνολογίας DAO.

Το εργαλείο ADO

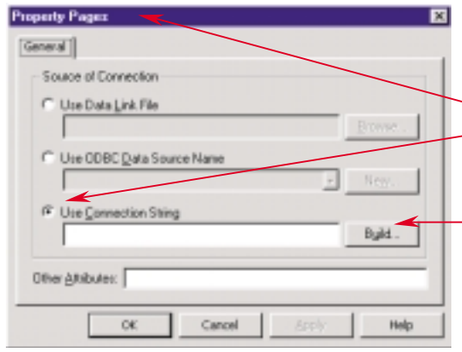
τρέχουσας εγγραφής του πίνακα, ή ακόμη και να κάνει άλμα στην πρώτη ή στην τελευταία εγγραφή.



Το αντικείμενο ADO

Για να λειτουργήσει η σύνδεση του αντικειμένου ADO με μια βάση δεδομένων, πρέπει να δοθούν τιμές στις ιδιότητες **`ConnectionString`** (συμβολοσειρά σύνδεσης) και **`RecordSource`** (πηγή εγγραφών).

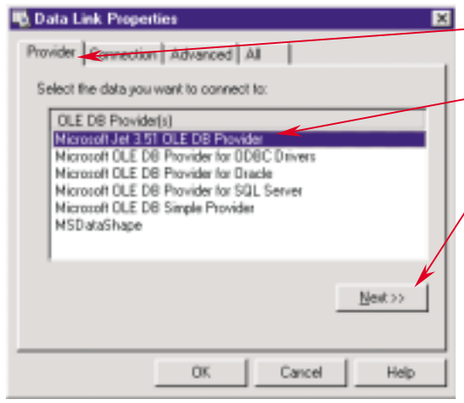
Η ιδιότητα **`ConnectionString`** παίρνει ένα σύνολο τιμών και καθορίζει το σύστημα διαχείρισης βάσεων δεδομένων που θα χρησιμοποιηθεί και τον τρόπο επικοινωνίας του αντικειμένου ADO με αυτό. Για να καθορίσουμε τις τιμές της ιδιότητας:



1. Κάνουμε διπλό κλικ πάνω στο όνομά της ιδιότητας,στο παράθυρο ιδιοτήτων του αντικειμένου ADO, οπότε εμφανίζεται το διαλογικό παράθυρο **Property Pages**.
2. Επιλέγουμε **Use Connection String**, για να συνδέσουμε το αντικείμενο με έναν πίνακα βάσης δεδομένων Access.
3. Πατάμε το πλήκτρο **Build**.

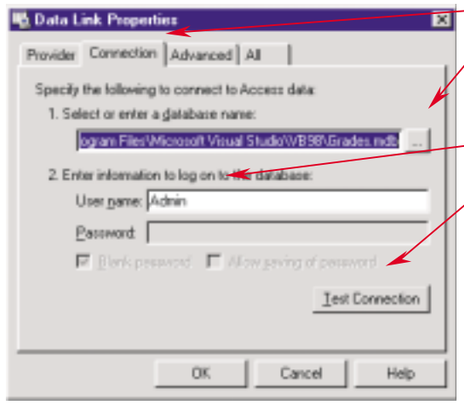
Εικόνα 26-1. *Το διαλογικό παράθυρο* ***Property Pages****, στην καρτέλα* ***General***

Από την πρώτη επιλογή **Use Data Link File** καθορίζουμε το αρχείο από το οποίο θα αντλούνται τα χαρακτηριστικά της σύνδεσης. Κάτι τέτοιο απαιτεί ειδικές γνώσεις. Με τη δεύτερη επιλογή **Use ODBC Data Source Name** μπορούμε να χρησιμοποιήσουμε ένα Data Source Name (**DSN**) όπως αυτό φαίνεται στο Control Panel των Window.



4. Εμφανίζεται το διαλογικό παράθυρο **Data Link Properties** με ενεργή την καρτέλα **Provider** (παροχέας).
5. Για σύνδεση με βάση δεδομένων της Access επιλέγουμε ως παροχέα τη μηχανή **Microsoft Jet 3.5.1 OLE DB**.
6. Πατάμε το πλήκτρο **Next**> >.

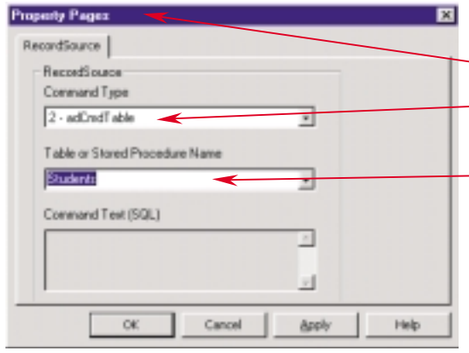
Εικόνα 26-2. *Το διαλογικό παράθυρο* ***Data Link Properties****, καρτέλα* ***Provider***



7. Ενεργοποιείται η καρτέλα **Connection**.
8. Πατάμε το πλήκτρο . . .
9. Εμφανίζεται διαλογικό παράθυρο από το οποίο μπορούμε να επιλέξουμε το όνομα του αρχείου που περιέχει τη βάση δεδομένων π.χ. το αρχείο `StudentsGrades.mdb`.
10. Παρατηρούμε ότι στην ιδιότητα **User Name** έχει δημιουργηθεί ο κωδικός χρήστη Admin.
11. Πατάμε το πλήκτρο **Test Connection** για να διαπιστώσουμε κατά πόσο είναι εφικτή η σύνδεση με τη βάση δεδομένων που επιλέξαμε.

Εικόνα 26-3. *Το διαλογικό παράθυρο* ***Data Link Properties****, καρτέλα* ***Connection***

Όταν φορτώνεται η φόρμα που περιέχει το αντικείμενο ADO, δημιουργείται ένα ***σύνολο εγγραφών*** (***recordset***) με στοιχεία από έναν ή περισσότερους πίνακες της βάσης δεδομένων. Το σύνολο των εγγραφών παραμένει ενεργό για όση ώρα παραμένει ενεργή η φόρμα. Η πηγή (source) των δεδομένων της βάσης, από την οποία αντλήθηκε το σύνολο των εγγραφών του αντικειμένου ADO, καθορίζεται από την ιδιότητα `RecordSource`.



1. Κάνουμε διπλό κλικ πάνω στο όνομα της ιδιότητας,στο παράθυρο ιδιοτήτων του αντικειμένου ADO, οπότε εμφανίζεται το διαλογικό παράθυρο **Property Pages**.
2. Από τη συνδυασμένη λίστα **Command Type** επιλέγουμε την τιμή **adCmdTable**, για να συνδέσουμε το αντικείμενο με πίνακα τη Access.
3. Επιλέγουμε έναν πίνακα από την πτυσσόμενη λίστα που είναι ενημερωμένη πλέον με τα ονόματα των υπαρχόντων πινάκων στη βάση δεδομένων που έχουμε επιλέξει.

Εικόνα 26-4. *Το διαλογικό παράθυρο* ***Property Pages****, στην καρτέλα* ***RecordSource***

Υπάρχει περίπτωση να μη θέλουμε να υπάρχουν στο σύνολο των εγγραφών του αντικειμένου ADO όλες οι εγγραφές του πίνακα. Όπως θα δούμε αργότερα, μπορούμε να περιγράψουμε τον τρόπο απομόνωσης κάποιων εγγραφών ή/και τον τρόπο απομόνωσης κάποιων πεδίων των εγγραφών. Επίσης, το σύνολο των εγγραφών μπορεί να περιέχει και πεδία από διαφορετικούς πίνακες. Στις δύο τελευταίες περιπτώσεις πρέπει, από τη συνδυασμένη λίστα **Command Type**, της καρτέλας **RecordSource**, να επιλέξουμε την τιμή **adCmdText**, αντί της τιμής **adCmdTable**, και να περιγράψουμε με μια ειδική γλώσσα (τη γλώσσα ερωτοαπαντήσεων SQL) τον τρόπο συνδυασμού των πεδίων των πινάκων και τα κριτήρια επιλογής των εγγραφών τους που θα μετάσχουν στο σύνολο εγγραφών. Επίσης, μπορούμε να αφήσουμε τη VB να αποφασίσει κατά περίπτωση και να δώσουμε την τιμή **adCmdUnknown**.

Μέχρι αυτό το σημείο έχουμε πετύχει το πρώτο τμήμα του σκοπού μας. Να συνδεθεί η βάση δεδομένων με τη φόρμα. Αν σταματήσουμε σε αυτό το σημείο ο χρήστης θα πατά τα πλήκτρα του αντικειμένου ADO, θα μεταβαίνει από εγγραφή σε εγγραφή του συνόλου εγγραφών, αλλά δεν θα παρατηρεί καμιά αλλαγή στα πεδία της φόρμας.

Πρόσδεση αντικειμένων στο αντικείμενο ελέγχου ADO

Αν κάνουμε ***πρόσδεση*** (***bound***) ενός αντικειμένου ελέγχου της φόρμας σε ένα πεδίο του συνόλου εγγραφών του αντικειμένου ADO, μπορούμε να κρατάμε συνεχώς ενημερωμένη την τιμή του αντικειμένου ελέγχου με την τιμή του πεδίου της τρέχουσας εγγραφής του συνόλου εγγραφών. Καθώς ο χρήστης μεταβαίνει από εγγραφή σε εγγραφή, με τη βοήθεια του αντικειμένου ADO, είτε πατώντας τα πλήκτρα του είτε χρησιμοποιώντας τις μεθόδους του μέσω κώδικα VB, η τιμή του αντικειμένου ελέγχου ενημερώνεται αυτόματα και στη φόρμα προβάλλεται η τιμή του πεδίου της τρέχουσας εγγραφής. Επίσης, αν ο χρήστης αλλάξει την τιμή του αντικειμένου ελέγχου και μεταβεί σε άλλη εγγραφή ή αν κλείσει τη φόρμα, η βάση δεδομένων ενημερώνεται, ώστε να αντανακλά όλες τις αλλαγές στοιχείων. Με αυτόν τον τρόπο περιορίζονται σημαντικά οι ανάγκες συγγραφής κώδικα.

Όλα τα γνωστά αντικείμενα ελέγχου, δηλαδή τα πλαίσια κειμένου, τα πλήκτρα σημείωσης, οι λίστες, οι συνδυασμένες λίστες κλπ. μπορούν να προσδεθούν σε ένα αντικείμενο ADO. Για να προσδέσουμε ένα αντικείμενο ελέγχου σε ένα αντικείμενο ADO πρέπει να δώσουμε τις κατάλληλες τιμές στις ιδιότητες **`DataSource`** (πηγή δεδομένων) και **`DataField`** (πεδίο δεδομένων).

Ως τιμή στην ιδιότητα **`DataSource`** δίνουμε το όνομα του αντικειμένου ADO, με το οποίο θέλουμε να προσδέσουμε το αντικείμενο ελέγχου. Ας σημειωθεί, ότι κατά την ενημέρωση αυτής της ιδιότητας στο παράθυρο ιδιοτήτων εμφανίζεται πτυσσόμενη λίστα ενημερωμένη με τα ονόματα των αντικειμένων ADO της φόρμας.

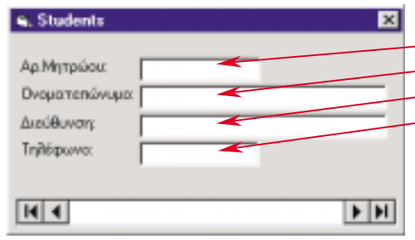
Στις φόρμες, που παρουσιάζονται δεδομένα από βάσεις δεδομένων, δεν είναι υποχρεωτικό να είναι όλα τα αντικείμενα ελέγχου προσδεδεμένα με πεδία του συνόλου εγγραφών. Για παράδειγμα, σε μια φόρμα μπορεί να υπάρχει ένα πεδίο που να δείχνει την τρέχουσα ημερομηνία και ώρα ή ένα πεδίο που να αναφέρει μια κατάσταση. Τα πεδία αυτά τα χαρακτηρίζουμε ως ***μη προσδεδεμένα*** (***unbounded***).

Ως τιμή στην ιδιότητα **DataField** δίνουμε το όνομα του πεδίου του πίνακα από τον οποίο δημιουργείται το σύνολο εγγραφών. Ας σημειωθεί, ότι κατά την ενημέρωση αυτής της ιδιότητας, στο παράθυρο ιδιοτήτων, εμφανίζεται πτυσσόμενη λίστα ενημερωμένη με τα ονόματα των πεδίων του συνόλου εγγραφών.

Άσκηση 26-1.

Να δημιουργηθεί μια φόρμα από την οποία να ενημερώνονται τα γενικά στοιχεία των μαθητών της βάσης δεδομένων που αναλύσαμε στο προηγούμενο μάθημα.

1. Σε μια φόρμα τοποθετούμε αντικείμενο ADO, το ονομάζουμε ADOStudents και ορίζουμε τις ιδιότητες:
   ConnectionString ως PROVIDER=Microsoft.Jet.OLEDB.3.51;
   Data Source=C:\Program Files\...\StudentsGrades.mdb;
   RecordSource ως Students
   CommandType ως 8-adCmdUnknown
2. Τοποθετούμε πλαίσια κειμένου, στα οποία δίνουμε ονόματα και καταχωρίζουμε τις ιδιότητες **DataSource** και **DataField**, που φαίνονται στην εικόνα 26-5. Τα ονόματα των πεδίων STD_Code, STD_Name κτλ. τα παίρνουμε από τον πίνακα 25-3.



Name	DataSource	DataField
Code	ADOStudents	STD_Code
Name	ADOStudents	STD_Name
Address	ADOStudents	STD_Address
Phone	ADOStudents	STD_Phone

Εικόνα 26-5. *Η φόρμα και οι ιδιότητες των αντικειμένων.*

Με τον τρόπο αυτό "δένουμε" κάθε αντικείμενο πλαισίου κειμένου με ένα συγκεκριμένο πεδίο του πίνακα Students.

Διαχείριση εγγραφών

Το σύνολο των εγγραφών ενός αντικειμένου ADO συνθέτει ένα αντικείμενο, το αντικείμενο Recordset. Για το αντικείμενο αυτό έχουν δημιουργηθεί μέθοδοι με τις οποίες γίνεται η διαχείριση των μελών του συνόλου εγγραφών. Από τις ιδιότητες αυτού του αντικειμένου παίρνουμε επίσης και άλλες πληροφορίες που έχουν σχέση με το σύνολο εγγραφών. Μερικές από τις πιο κοινές μεθόδους του Recordset είναι οι:

Μέθοδοι διαμόρφωσης

AddNew. Προετοιμάζει μια εγγραφή για εισαγωγή στο σύνολο εγγραφών και στον αντίστοιχο πίνακα της βάσης δεδομένων.

Update. Μεταφέρει τις αλλαγές που έχουν γίνει στα πεδία μιας εγγραφής του συνόλου εγγραφών στα αντίσοτιχα πεδία του πίνακα της βάσης δεδομένων.

Delete. Διαγράφει την τρέχουσα εγγραφή του συνόλου εγγραφών από το σύνολο εγγραφών και από τον πίνακα της βάσης δεδομένων.

Refresh. Ενημερώνει ξανά το σύνολο εγγραφών με τις αλλαγές που πιθανόν να έχουν κάνει στον πίνακα, άλλοι χρήστες της βάσης δεδομένων.

Μέθοδοι μετακίνησης

MoveFirst. Μετακινεί στην πρώτη εγγραφή του συνόλου εγγραφών.

MoveNext. Μετακινεί στην επόμενη εγγραφή του συνόλου εγγραφών.

MovePrevious. Μετακινεί στην προηγούμενη εγγραφή του συνόλου εγγραφών.

MoveLast. Μετακινεί στην τελευταία εγγραφή του συνόλου εγγραφών.

Μέθοδοι αναζήτησης

FindFirst *κριτήρια*. Αναζητά την πρώτη εγγραφή που ικανοποιεί τα *κριτήρια*.

FindNext *κριτήρια*. Αναζητά την επόμενη εγγραφή που ικανοποιεί τα *κριτήρια*.

FindPrevious *κριτήρια*. Αναζητά την προηγούμενη εγγραφή που ικανοποιεί τα *κριτήρια*.

FindLast *κριτήρια*. Αναζητά την τελευταία εγγραφή που ικανοποιεί τα κριτήρια.

Μερικές από τις πιο συχνά χρησιμοποιούμενες ιδιότητες του Recordset είναι:

AbsolutePosition: Η θέση της τρέχουσας εγγραφής μέσα στο σύνολο εγγραφών.

RecordCount: Το πλήθος των εγγραφών που έχουν μεταφερθεί μέχρι αυτή τη στιγμή στο σύνολο εγγραφών.

BOF: Η αρχή του συνόλου εγγραφών. Παίρνει την τιμή True αν τη στιγμή του ελέγχου βρισκόμαστε ήδη στην αρχή του συνόλου εγγραφών (Begin Of File).

EOF: Το τέλος του συνόλου εγγραφών. Παίρνει την τιμή True αν τη στιγμή του ελέγχου βρισκόμαστε ήδη στο τέλος του συνόλου εγγραφών (End Of File).

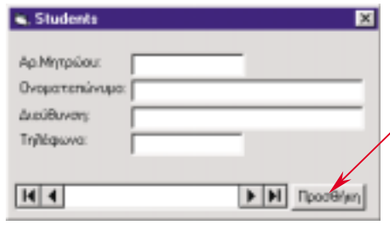
NoMatch: Μη επιτυχής αναζήτηση. Μετά από αναζήτηση παίρνει την τιμή True αν δεν έχει βρεθεί εγγραφή που να ικανοποιεί τα κριτήρια της αναζήτησης.

Άσκηση 26-2.

Να τοποθετηθεί, πάνω στη φόρμα της άσκησης 26-1, πλήκτρο με όνομα Add και **Caption** "ÐñïόèÞêç", του οποίου το πάτημα να προκαλεί την προετοιμασία εισαγωγής εγγραφής μέσα στο σύνολο εγγραφών.

Δημιουργούμε το πλήκτρο Add και του επισυνάπτουμε τον κώδικα:

```
Private Sub Add_Click()
    ADOStudents.Recordset.AddNew
    Code.Setfocus
End Sub
```



Εικόνα 26-6. *Η φόρμα με το πλήκτρο Add*

Για να εισάγουμε εγγραφές πατάμε το πλήκτρο Add και συμπληρώνουμε τα πλαίσια κειμένου της φόρμας. Όταν τελειώσουμε με όλα τα στοιχεία πατάμε πάλι το πλήκτρο Add και επαναλαμβάνουμε την διαδικασία. Αυτή η διαδικασία γίνεται για όσες εγγραφές θέλουμε να εισάγουμε.

Χρησιμοποιώντας τα πλήκτρα κίνησης του αντικειμένου ADO, μπορούμε να κινηθούμε μέσα στις εγγραφές του συνόλου εγγραφών.

Με αυτό το απλό πρόγραμμα μπορούμε ακόμη και να ενημερώσουμε τη βάση. Για παράδειγμα, αν στην τρέχουσα εγγραφή πληκτρολογήσουμε μιαν άλλη τιμή στο πλαίσιο κειμένου αριθμός τηλεφώνου, η αλλαγή αυτή μεταφέρεται στη βάση δεδομένων, μόλις κινηθούμε σε μία άλλη εγγραφή.

Άσκηση 26-3.

Να τοποθετηθεί, πάνω στη φόρμα, πλήκτρο με όνομα Delete και **Caption** "ÄéáãñáöÞ", του οποίου το πάτημα να προκαλεί τη διαγραφή της εγγραφής. Επίσης, να τοποθετηθεί πλήκτρο με όνομα Update και **Caption** "ÅíçìÝñùóç".

Δημιουργούμε τα δύο πλήκτρα και στο παράθυρο κώδικα της φόρμας γράφουμε:

```
Private Sub Delete_Click()
    ADOStudents.Recordset.Delete
    ADOStudents.Recordset.MoveNext
    If ADOStudents.Recordset.EOF = True Then
        ADOStudents.Recordset.MoveLast
    End If
End Sub
Private Sub Update_Click()
    ADOStudents.Recordset.Update
End Sub
```

Παρατηρήστε ότι μετά τη διαγραφή μεταβαίνουμε στην επόμενη εγγραφή του συνόλου εγγραφών και αν αυτή δεν υπάρχει, στην τελευταία εγγραφή της.

Άσκηση 26-4.

Να τοποθετηθεί, πάνω στη φόρμα, πλήκτρο με όνομα `Find` και **`Caption`** "`ÁíáæÞôçóς`". Το πάτημα του πλήκτρου να παρουσιάζει διαλογικό παράθυρο, στο οποίο ο χρήστης να πληκτρολογεί τμήμα του ονόματος του μαθητή. Στη φόρμα να παρουσιάζονται, διαδοχικά, τα στοιχεία των μαθητών των οποίων το όνομα περιέχει το τμήμα που πληκτρολόγησε ο χρήστης.

Δημιουργούμε το πλήκτρο και γράφουμε τον κώδικα:

```
Private Sub Find_Click()

    Dim Criteria     As String
    Dim PartName     As String
    Dim Response     As String

   'ÐÜñå ôï ôìÞìá ôïõ ïíüìáôïò êáé óýíèåóå ôï êñéôÞñéï
    PartName = InputBox("Äþóôå ôìÞìá ôïõ ïíüìáôïò:", "ÁíáæÞôçóç")
    Criteria = "STD_Name Like '*" & PartName & "*'"
   'ÁðáíÜëåâå ôçí áíáæÞôçóç ìÝ÷ñé íá âéáêüøåé ï ÷ñÞóôçò
    ADOStudents.Recordset.FindFirst Criteria
    Do While Not ADOStudents.Recordset.NoMatch
        Response = MsgBox("Íá óõíå÷ßóù ìå ôïí ðñùìåíï", _
                          VbQuestion, "ÁíáæÞôçóç")
        If Response = vbNo Then Exit Do
        ADOStudents.Recordset.FindNext Criteria
    Loop

End Sub
```

Αν η λειτουργία αναζήτησης αποτύχει, η τρέχουσα εγγραφή δεν αλλάζει.

Τα κριτήρια των μεθόδων `Findxxxx` συντάσσονται όπως ακριβώς και τα κριτήρια του τμήματος `WHERE` των εντολών `SELECT`. Για τις εντολές `SELECT` περισσότερα στο επόμενο μάθημα.

Παρατηρήστε τον τρόπο σύνταξης του κριτηρίου στην πιο πάνω άσκηση. Η κωδική λέξη `Like` (μοιάζει, σαν) καταλαμβάνει τη θέση συμβόλου σύγκρισης. Φυσικά, στη σύνταξη των κριτηρίων μπορούμε να χρησιμοποιήσουμε και τα κλασικά σύμβολα σύγκρισης. Για παράδειγμα, κατά την αναζήτηση του μαθητή με αριθμό μητρώου 189 θα γράφαμε:

```
Criteria = "STD_Code = 189"
```

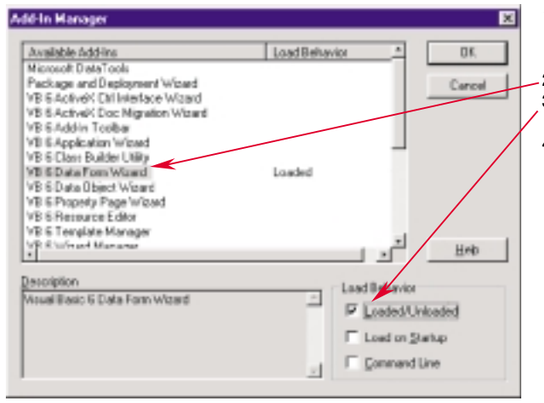
και για την αναζήτηση των μαθητών που έχουν αριθμό μητρώου μεγαλύτερο ή ίσο του 100 και το όνομά τους αρχίζει από Β θα γράφαμε:

```
Criteria = "STD_Code >= 100 AND STD_Name >'B'"
```

Στην αναζήτηση αλφαριθμητικού, το λεκτικό που αναζητείται πρέπει να περικλείεται μέσα σε μονά εισαγωγικά. Στην αναζήτηση αριθμών, ο αριθμός δεν πρέπει να περικλείεται μέσα σε εισαγωγικά.

Αυτόματη δημιουργία φόρμας

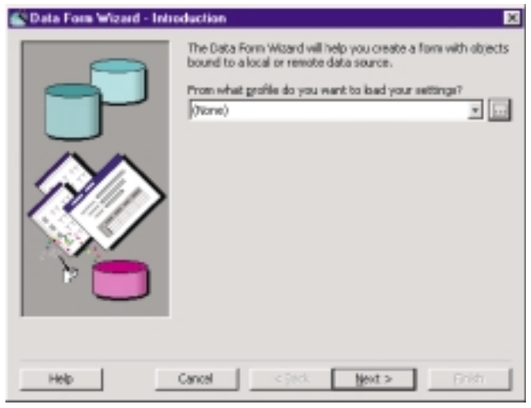
Η δημιουργία μιας φόρμας, από την οποία διαχειριζόμαστε τα στοιχεία ενός συνόλου εγγραφών, είναι μια αρκετά συνήθης και τυποποιημένη διαδικασία. Για να μειωθεί ο χρόνος ανάπτυξης των εφαρμογών, στο περιβάλλον εργασίας μπορεί να τοποθετηθεί ένα πρόσθετο εργαλείο, ο "μάγος των φορμών δεδομένων" ***Data Form Wizard*** (***DFW***), με το οποίο είναι δυνατόν να γίνεται η αυτόματη δημιουργία φορμών. Για να προστεθεί το εργαλείο Data Form Wizard στο περιβάλλον εργασίας της VB:



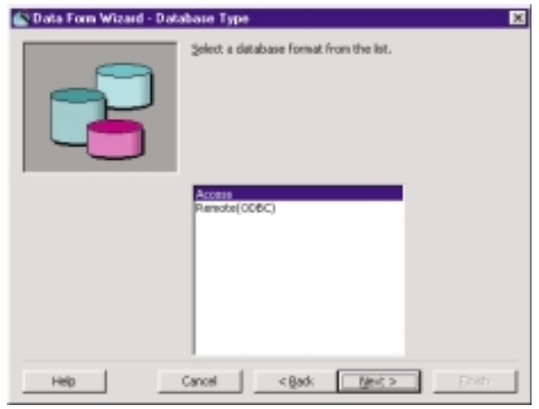
1. Από το μενού **Add-Ins** επιλέγουμε **Add-In Manager**, οπότε εμφανίζεται το διαλογικό παράθυρο εισαγωγής πρόσθετων (Add-In).
2. Επιλέγουμε τον Data Form Wizard.
3 Καθορίζουμε αν θα τον φορτώνουμε κάθε φορά ή θα φορτώνεται αυτόματα.
4. Πατώντας το πλήκτρο Ok το μενού **Add-Ins** προσαυξάνεται με την επιλογή:

Εικόνα 26-7. *Διαλογικό παράθυρο για την εισαγωγή πρόσθετων στο περιβάλλον εργασίας της VB.*

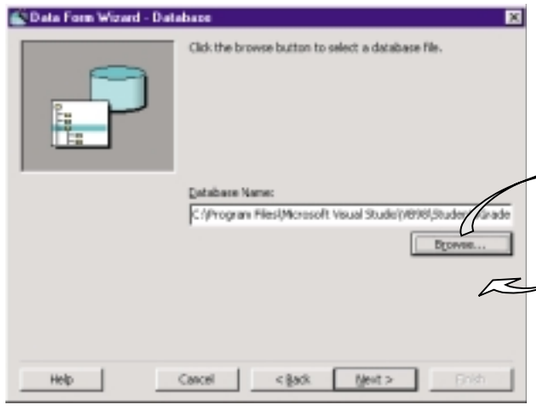
Από τη στιγμή που έχει προστεθεί ο Data Form Wizard μπορεί να κληθεί από το μενού **Add-Ins** και με διαδοχικά βήματα να μας οδηγήσει στη δημιουργία της επιθυμητής φόρμας.

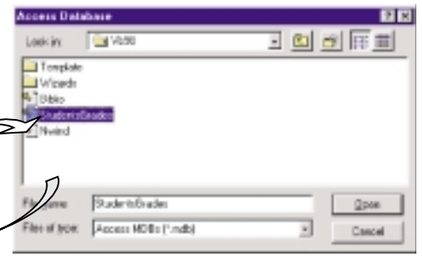


Εικόνα 26-8. *Εισαγωγικό σχόλιο. Δυνατότητα επιλογής κάποιου προτύπου*

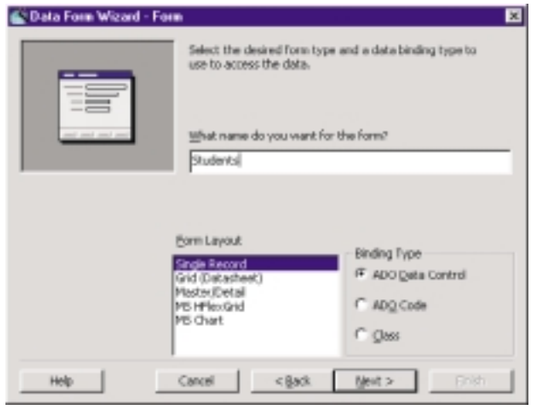


Εικόνα 26-9. *Επιλογή του είδους της βάσης δεδομένων*

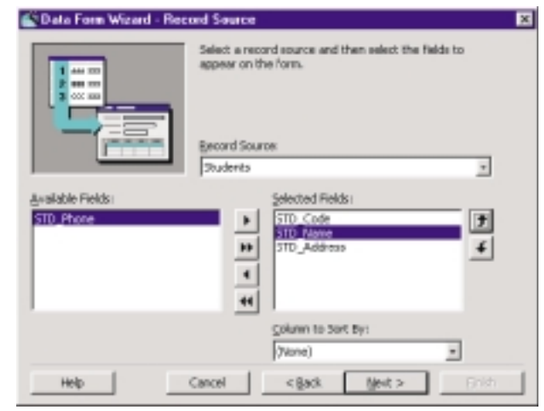




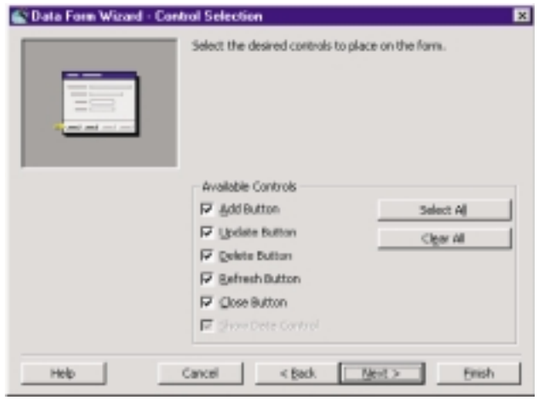
Εικόνα 26-10. *Επιλογή του αρχείου που περιέχει τη βάση δεδομένων*



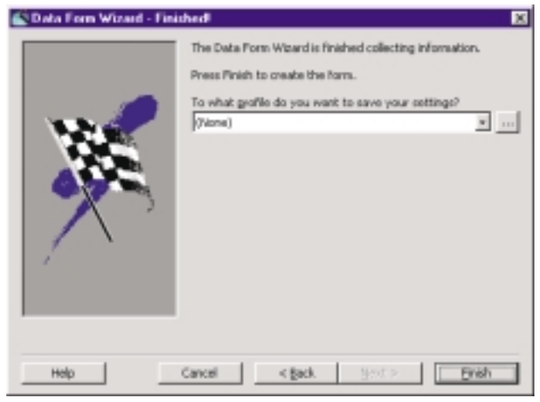
Εικόνα 26-11. *Επιλογή του είδους της φόρμας και του τρόπου σύνδεσης με τη βάση δεδομένων*



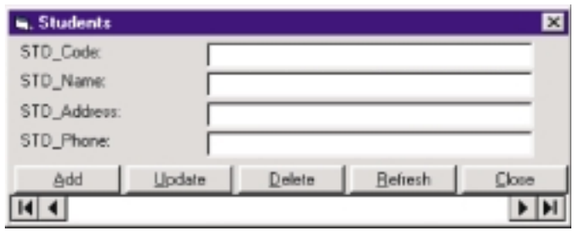
Εικόνα 26-12. *Επιλογή του πίνακα και των πεδίων του που θα εμφανίζονται στη φόρμα*



Εικόνα 26-13. *Επιλογή των πλήκτρων και των λειτουργιών που θα επισυναφθούν στη φόρμα*



Εικόνα 26-14. *Τερματισμός*



Εικόνα 26-15. *Το αποτέλεσμα*

Παρατηρούμε, ότι στη φόρμα που παράγεται έχουν προστεθεί όλα τα απαιτούμενα αντικείμενα ελέγχου ακόμα και ετικέτες. Χρειάζονται όμως κάποιες αλλαγές στην αισθητική της φόρμας. Τα **`Caption`** των ετικετών έχουν προκύψει από τα ονόματα των πεδίων του πίνακα και τα μεγέθη των πλαισίων κειμένου δεν είναι ανάλογα με το μέγεθος των στοιχείων που θα υποδεχθούν. Πάντως, έχει παραχθεί κώδικας για τα πιο σημαντικά συμβάντα.

Ανακεφαλαίωση

Η κατασκευή και διαχείριση των φορμών βασίζεται σε ένα αντικείμενο ελέγχου τύπου ActiveX, το οποίο χρησιμοποιεί μια τεχνολογία η οποία είναι γνωστή ως ADO (ActiveX Data Object). Το αντικείμενο αυτό τοποθετείται πάνω σε μια φόρμα και προσφέρει έναν οπτικό μηχανισμό στο χρήστη για την προσπέλαση των εγγραφών ενός πίνακα μιας βάσης δεδομένων. Για να λειτουργήσει η σύνδεσή του αντικειμένου ADO με μια βάση δεδομένων πρέπει να δοθούν τιμές στις ιδιότητες **`ConnectionString`** και **`RecordSource`**.

Επίσης, αν κάνουμε πρόσδεση ενός αντικειμένου ελέγχου της φόρμας σε ένα πεδίο του συνόλου εγγραφών του αντικειμένου ADO, μπορούμε να κρατάμε συνεχώς ενημερωμένη την τιμή του αντικειμένου ελέγχου, με την τιμή του πεδίου της τρέχουσας εγγραφής του συνόλου εγγραφών. Για να προσδέσουμε ένα αντικείμενο ελέγχου σε ένα αντικείμενο ADO πρέπει να δώσουμε τις κατάλληλες τιμές στις ιδιότητες **`DataSource`** και **`DataField`**.

Εργαστηριακές ασκήσεις

1. Να εκτελέσετε την άσκηση 26-1 και αμέσως μετά την άσκηση 26-2. Στη συνέχεια να συμπληρώσετε τον πίνακα της βάσης δεδομένων με στοιχεία.
2. Να εκτελέσετε την άσκηση 26-3.
3. Δημιουργήστε φόρμα για την εισαγωγή των μαθημάτων στον αντίστοιχο πίνακα.
4. Να εκτελέσετε την άσκηση 26-4.
5. Χρησιμοποιώντας το πρόσθετο εργαλείο Data Form Wizard, να δημιουργήσετε μια φόρμα για την εισαγωγή των στοιχείων των μαθητών στον αντίστοιχο πίνακα.

Μάθημα 27
Η γλώσσα ερωτοαπαντήσεων SQL
Πρόσθετα αντικείμενα προβολής στοιχείων πινάκων

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν :

- Να συντάσσουν απλές εντολές της γλώσσας ερωτοαπαντήσεων SQL.
- Να εμφανίζουν σε φόρμες τα αποτελέσματα των εντολών SQL.
- Να χρησιμοποιούν τα ειδικά αντικείμενα ελέγχου για την παρουσίαση σε φόρμες των περιεχομένων πινάκων.

Στο προηγούμενο μάθημα είδαμε πώς μπορούμε να προβάλλουμε το περιεχόμενο πινάκων σε φόρμες στην οθόνη. Όμως, δεν είναι λίγες οι φορές που θέλουμε να παρουσιάσουμε μερικά και όχι όλα τα πεδία και μερικές από τις εγγραφές ενός πίνακα ή να συνδυάσουμε τα πεδία από δύο ή περισσότερους πίνακες. Επίσης, πολλές φορές θέλουμε να δούμε τα πεδία ενός συνόλου εγγραφών σε μια λίστα ή σε ένα φύλλο εργασίας και όχι σε μορφή καρτέλας όπως γίνεται στις φόρμες.

Προφέρεται σίκιουελ.

Σε αυτό το μάθημα θα μελετήσουμε πώς, με τη βοήθεια της γλώσσας ***SQL (Structured Query Language)***, μπορούμε να απομονώσουμε και να μεταφέρουμε σε μια φόρμα μόνο τα στοιχεία του πίνακα που μας ενδιαφέρουν. Επίσης, θα μελετήσουμε και κάποια ειδικά αντικείμενα ελέγχου, τα οποία χρησιμοποιούνται ως υποκατάστατα μιας φόρμας για την καλύτερη προβολή του περιεχομένου ενός συνόλου εγγραφών.

Η γλώσσα SQL

Το σημείο στο οποίο πρέπει να επέμβουμε, ώστε να καθορίσουμε τα στοιχεία που θα απαρτίσουν ένα σύνολο εγγραφών, είναι το σημείο στο οποίο καθορίζουμε την τιμή της ιδιότητας **`RecordSource`** του αντικειμένου ADO. Με την προϋπόθεση ότι έχουμε δώσει στην ιδιότητα **`CommandType`** την τιμή `adCmdUnknown` ή την τιμή `adCmdText`, μπορούμε να δώσουμε ως τιμή της ιδιότητας **`RecordSource`** μια εντολή προς το σύστημα διαχείρισης της βάσης δεδομένων. Η εντολή αυτή είναι μια εντολή SELECT της γλώσσας ***SQL***, η οποία καθορίζει τον τρόπο με τον οποίο θα συντεθεί το σύνολο των εγγραφών.

Η γλώσσα SQL είναι μία γλώσσα με την οποία είναι δυνατόν να γίνει αναζήτηση, εισαγωγή, και ενημέρωση των στοιχείων μιας βάσης δεδομένων. Σήμερα, έχει υιοθετηθεί από όλα τα γνωστά συστήματα διαχείρισης βάσεων δεδομένων. Με τη βασική εντολή της γλώσσας, την εντολή SELECT, μπορούμε να υποβάλλουμε ερωτήσεις στο σύστημα διαχείρισης της βάσης δεδομένων. Η εντολή αυτή έχει συνήθως τη μορφή:

```
SELECT πεδίο1, πεδίο2, . . .
FROM πίνακα
WHERE κριτήρια
ORDER BY πεδίο
```

Με την εντολή SELECT, μπορούμε να επιλέξουμε τα πεδία του πίνακα, που θα συνθέσουν κάθε εγγραφή στοιχείο του συνόλου των εγγραφών. Αμέσως μετά την κωδική λέξη SELECT (επέλεξε), αναγράφουμε ένα ένα τα ονόματα των πεδίων που θέλουμε να επιλεγούν από τον πίνακα. Επίσης, μετά την κωδική λέξη `FROM` (από), αναγράφουμε το όνομα του πίνακα από τον οποίο θα γίνει η άντληση των στοιχείων.

Στην περίπτωση που χρησιμοποιείται και η κωδική λέξη `WHERE` (όπου), καθορίζονται τα κριτήρια με τα οποία θα γίνει μερική επιλογή εγγραφών από το σύνολο των εγγραφών. Μετά την κωδική λέξη `WHERE` γράφουμε λογικές παραστάσεις, στις οποίες μετέχουν πεδία του πίνακα. Από όλες τις εγγραφές του πίνακα επιλέγονται μόνον αυτές για τις οποίες η λογική παράσταση θα είναι αληθής.

Τέλος, στην περίπτωση που θέλουμε να είναι ταξινομημένες οι εγγραφές του συνόλου εγγραφών, ως προς ένα ή περισσότερα πεδία, γράφουμε μετά από τις κωδικές λέξεις ORDER BY τα ονόματα των πεδίων, ως προς τα οποία θέλουμε να γίνει η ταξινόμηση.

Παράδειγμα 27-1.

Έστω, ότι από τον πίνακα Students της βάσης δεδομένων StudentsGrades, θέλουμε να αντλήσουμε από όλες τις εγγραφές, τα πεδία που περιέχουν τον αριθμό μητρώου και το ονοματεπώνυμο, δηλαδή τα πεδία STD_Code και STD_Name. Η εντολή SELECT, που πραγματοποιεί αυτήν την επιλογή είναι:

```
SELECT STD_Code, STD_Name FROM Students
```

Σε περίπτωση που θέλουμε να αντλήσουμε όλα τα πεδία ενός πίνακα δεν είναι υποχρεωτικό να αναφέρουμε ένα ένα τα πεδία του, αλλά μπορούμε να γράψουμε τον αστερίσκο (*) στη θέση τους. Για παράδειγμα:

```
SELECT * FROM Students
```

Άσκηση 27-1.

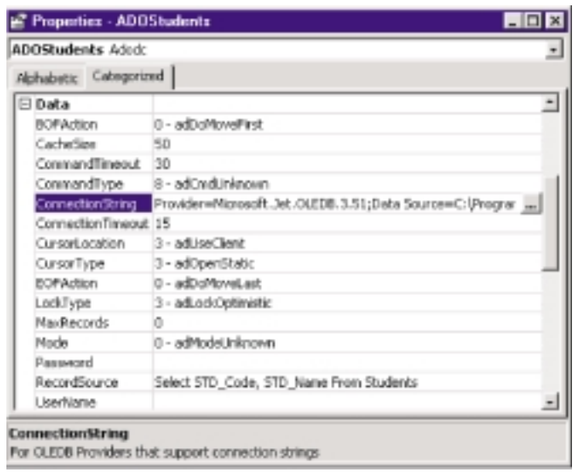
Να δημιουργηθεί φόρμα που να εμφανίζει μόνο τον αριθμό μητρώου και το επώνυμο των μαθητών.

1. Σε ένα νέο έργο δημιουργούμε μια φόρμα και πάνω της τοποθετούμε ένα αντικείμενο ADO, το οποίο ονομάζουμε ADOStudents.
2. Από το παράθυρο ιδιοτήτων του αντικειμένου ADO ορίζουμε τις ιδιότητες:

   ConnectionString ως PROVIDER=Microsoft.Jet.OLEDB.3.51;
   Data Source=C:\Program Files\…\StudentsGrades.mdb;

   CommandType ως 8-adCmdUnknown.
3. Πατάμε το πλήκτρο . . . της ιδιότητας **RecordSource**. Στο παράθυρο που θα εμφανιστεί, στο πλαίσιο κειμένου **Command Text**, γράφουμε την εντολή:

   ```
   SELECT STD_Code, STD_Name FROM Students
   ```



Εικόνα 27-1. *Οι ιδιότητες του αντικειμένου ADO, που σχετίζονται με τη βάση δεδομένων.*

4. Τοποθετούμε στη φόρμα δύο πλαίσια κειμένου, στα οποία δίνουμε τα ονόματα Code και Name και καταχωρίζουμε στην ιδιότητα τους **DataSource** την τιμή ADOStudents.
5. Στην ιδιότητα **DataField**, του πλαισίου κειμένου Code, δίνουμε την τιμή STD_Code. Επίσης, στην ιδιότητα **DataField**, του πλαισίου κειμένου Name, δίνουμε την τιμή STD_Name.

Ιδιότητες **DataSource** και **DataField** έχουν και οι ετικέτες. Χρησιμοποιούμε ετικέτες όταν δε θέλουμε να δώσουμε στο χρήστη τη δυνατότητα να αλλάζει το περιεχόμενο των πεδίων.

Παραδείγματα 27-2.

α) Έστω, ότι θέλουμε να επιλέξουμε το μαθητή, του οποίουν το όνομα είναι "Ãéþñãïò Ïéêïíüìïõ". Συντάσσουμε την εντολή:

```
SELECT STD_Code, STD_Name
FROM Students
WHERE STD_Name='Ãéþñãïò Ïéêïíüìïõ'
```

β) Έστω, ότι θέλουμε να επιλέξουμε τους μαθητές, των οποίων το όνομα αρχίζει από "ÌÐÁ". Συντάσσουμε την εντολή:

```
SELECT STD_Code, STD_Name
FROM Students
WHERE STD_Name Like 'ÌÐÁ*'
```

γ) Ο αστερίσκος παίζει το ρόλο μπαλαντέρ ενός ή περισσότερων χαρακτήρων. Όποιοι και αν είναι οι χαρακτήρες που ακολουθούν το φθόγγο "ÌÐÁ" είναι αποδεκτοί.
Έστω, ότι στη βάση δεδομένων του μαθητολογίου, οι μαθητές που έχουν αριθμό μητρώου μεταξύ 100 και 199, είναι μαθητές της πρώτης τάξης, οι μαθητές που έχουν αριθμό μητρώου μεταξύ 200 και 299, είναι μαθητές της δευτέρας τάξης κλπ. Θέλουμε να επιλέξουμε τους μαθητές, της δευτέρας τάξης. Συντάσσουμε την εντολή:

```
SELECT STD_Code, STD_Name
FROM Students
WHERE STD_Code>=200 AND STD_Code<300
```

δ) Αντί των τελεστών >= και <= μπορεί να χρησιμοποιηθεί και η κωδική λέξη BETWEEN. Έστω, ότι ο πίνακας των μαθητών έχει και ένα πρόσθετο πεδίο, το πεδίο STD_BirthDate. Η εντολή:

```
SELECT * FROM Students
WHERE  Birthdate BETWEEN #01/01/87# AND #12/31/87#
```

δημιουργεί ένα σύνολο εγγραφών με όλα τα στοιχεία όλων των μαθητών που γεννήθηκαν το 1987. Παρατηρήστε ότι οι ημερομηνίες πρέπει να περικλείονται μεταξύ συμβόλων (#). Επίσης, παρατηρήστε ότι στις ημερομηνίες, πρώτα γράφεται ο μήνας και μετά η μέρα

Άσκηση 27-2.

Να τοποθετηθεί, πάνω στη φόρμα, πλήκτρο με όνομα Selection και **Caption** "ÅðéëïãÞ". Το πάτημα του πλήκτρου να παρουσιάζει διαλογικό παράθυρο, στο οποίο ο χρήστης να πληκτρολογεί τμήμα του ονόματος του μαθητή. Κλείνοντας το διαλογικό παράθυρο να ενημερώνεται το σύνολο εγγραφών του αντικειμένου ADO, ώστε στη φόρμα να εμφανίζονται μόνον οι μαθητές των οποίων το όνομα περιέχει το τμήμα του ονόματος που πληκτρολόγησε ο χρήστης.

Δημιουργούμε το πλήκτρο και γράφουμε τον κώδικα:

```
Private Sub Selection_Click()
    Dim PartName    As String
    Dim SQLstr      As String
   'ÐÜñå ôï ôìÞìá ôïõ ïíüìáôïò êáé óýíèåóå ôï query
    PartName = InputBox("Äþóôå ôìÞìá ôïõ ïíüìáôïò:", "ÅðéëïãÞ")
    SQLstr = "SELECT * FROM Students " & _
             "WHERE STD_Name Like '*" & PartName & "*'"
    ADOStudents.RecordSource = SQLstr
    ADOStudents.Refresh
End Sub
```

Κάθε φορά που δίνεται ένα άλλο τμήμα ονόματος, αλλάζουμε την ιδιότητα **RecordSource** του αντικειμένου ADO και ζητάμε να γίνει φρεσκάρισμα του συνόλου εγγραφών.

Για να είναι δυνατή η σωστή ταξινόμηση των ελληνικών χαρακτήρων πρέπει να έχει γίνει κατάλληλη δήλωση στο σύστημα διαχείρισης των βάσεων δεδομένων που χρησιμοποιούμε.

Παραδείγματα 27-3.

α) Οι εγγραφές εμφανίζονται στη φόρμα σας με τη σειρά της εισαγωγής τους στον πίνακα της βάσης. Οι μαθητές μπορεί να μην έχουν εισαχθεί κατά αλφαβητική σειρά. Για να φαίνονται σε αλφαβητική σειρά ως προς το όνομά τους γράφουμε:

```
SELECT * FROM Students ORDER BY STD_Name
```

β) Έστω, ότι θέλουμε να παρουσιάζονται οι μαθητές από το μεγαλύτερο σε ηλικία προς το μικρότερο και σε περίπτωση που δύο μαθητές έχουν την ίδια ηλικία να ταξινομούνται και αλφαβητικά ως προς το ονοματεπώνυμό τους. Συντάσσουμε την εντολή:

```
SELECT * FROM Students ORDER BY STD_BirthDate DESC, STD_Name
```

Η κωδική λέξη DESC (Descending) δηλώνει την ταξινόμηση κατά φθίνουσα σειρά. Για την ταξινόμηση σε αύξουσα σειρά δεν απαιτείται η γραφή κωδικής λέξης.

Όπως έχουμε ήδη αναφέρει, τα πεδία που συνθέτουν το σύνολο των εγγραφών μπορεί να είναι παρμένα από διαφορετικούς πίνακες. Σε αυτήν την περίπτωση, μετά την κωδική λέξη FROM αναγράφουμε όλους τους πίνακες από τους οποίους λαμβάνονται πεδία και μετά την κωδική λέξη WHERE, τις σχέσεις μεταξύ των πεδίων των διαφορετικών πινάκων.

Παραδείγματα 27-4.

Η εντολή:

```
SELECT LES_Name, STD_Name,
       GRD_AGrade, GRD_BGrade, GRD_CGrade, GRD_DGrade
FROM Grades, Lessons, Students
WHERE LES_Code=GRD_LES_Code AND STD_Code=GRD_STD_Code
```

Επιλέγει τα ονόματα των μαθημάτων και των μαθητών από τους πίνακες `Lessons` και `Students` αντίστοιχα και τους βαθμούς από τον πίνακα `Grades`. Η σύνδεση μεταξύ βαθμών και μαθημάτων γίνεται με τη σχέση `LES_Code=GRD_LES_Code` και η σύνδεση μεταξύ αριθμών μητρώου και μαθημάτων με τη σχέση `STD_Code=GRD_STD_Code`.

Άσκηση 27-3.

Να δημιουργηθεί φόρμα η οποία να ζητά τον αριθμό μητρώου ενός μαθητή και να εμφανίζει τους βαθμούς του σε όλα τα μαθήματα.

1. Σε μια φόρμα τοποθετούμε αντικείμενο τύπου ADO και πέντε πλαίσια κειμένου, ένα για την ονομασία του μαθήματος και τέσσερα για τους βαθμούς του.
2. Συνδέουμε το αντικείμενο ADO με τη βάση δεδομένων χωρίς να καθορίσουμε την τιμή της ιδιότητας **RecordSource**.
3. Προσδένουμε τα πλαίσια κειμένου στο αντικείμενο ADO και στις ιδιότητες **DataField** δίνουμε τις τιμές `LES_Name`, `GRD_AGrade`, `GRD_BGrade`, `GRD_CGrade`, `GRD_DGrade` αντίστοιχα.
4. Γράφουμε τον κώδικα:

```
Private Sub Form_Load()
    Dim StudCode    As String
    Dim SQLstr      As String
   'ÐÜñå ôï üíïìá ôïõ ìáèçôÞ
    StudCode = InputBox("Äþóå ôïí áñéèìü ìçôñþïõ:", "Åðéëïãþ")
    SQLstr = "SELECT LES_Name, " & _
             "GRD_AGrade, GRD_BGrade, GRD_CGrade, GRD_DGrade " & _
             "FROM Grades, Lessons " & _
             "WHERE LES_Code=GRD_LES_Code AND STD_Code=" & StudCode
    ADOStudents.RecordSource = SQLstr
    ADOStudents.Refresh
    Me.Caption="Áñéèìüò ìçôñþïõ:" & StudCode
End Sub
```


Πρόσθετα αντικείμενα ελέγχου

Για την προβολή του περιεχομένου βάσεων δεδομένων, εκτός από τα τυπικά αντικείμενα ελέγχου μπορούμε να χρησιμοποιήσουμε και πρόσθετα αντικείμενα του τύπου ActiveX. Αυτά τα αντικείμενα έχουν σχεδιαστεί ειδικά για να καλύπτουν τις ιδιαιτερότητες μιας δομής συνόλου δεδομένων. Για να προσθέσουμε κάποια από αυτά τα εργαλεία στην εργαλειοθήκη:

1. Επιλέγουμε **Project ¦ Components** από τη γραμμή μενού.
2. Επιλέγουμε τον καρτελοδείκτη **Controls.**
3. Από το πινάκιο με τις ομάδες των εργαλείων (δείτε εικόνα 13-7) επιλέγουμε αυτές των οποίων τα εργαλεία θέλουμε να προσθέσουμε στην εργαλειοθήκη.

Μερικά από τα πιο κοινά εργαλεία είναι:

Data Bound ListBox (Προσδεδεμένη λίστα): Δημιουργεί λίστες παρόμοιες με τις απλές λίστες. Όμως, το περιεχόμενο της προσδεδεμένης λίστας γεμίζει αυτόματα με τις τιμές ενός πεδίου του συνόλου εγγραφών του αντικειμένου ADO, στο οποίο έχει γίνει η πρόσδεσή του. Δεν απαιτούνται διαδοχικές κλήσεις της μεθόδου `AddItem` για την εισαγωγή στη λίστα της τιμής του πεδίου από κάθε εγγραφή.

Data Bound ComboBox (Προσδεδεμένη συνδυασμένη λίστα): Δημιουργεί λίστες παρόμοιες με τις απλές συνδυασμένες λίστες. Όμως, το περιεχόμενο της προσδεδεμένης συνδυασμένης λίστας γεμίζει αυτόματα με τις τιμές ενός πεδίου του συνόλου εγγραφών του αντικειμένου ADO, στο οποίο έχει γίνει η πρόσδεσή του. Δεν απαιτούνται διαδοχικές κλήσεις της μεθόδου `AddItem` για την εισαγωγή στη συνδυασμένη λίστα της τιμής του πεδίου από κάθε εγγραφή.

Άσκηση 27-4.

Σε μια φόρμα να τοποθετηθεί μια προσδεδεμένη λίστα, στην οποία να περιέχονται τα ονόματα των μαθητών που είναι καταχωρισμένα στον πίνακα `Students` της βάσης δεδομένων `StudentsGrades`.

1. Τοποθετούμε στη φόρμα ένα αντικείμενο ADO, το οποίο ονομάζουμε `ADOStudents`, και ενημερώνουμε τις ιδιότητές του, ώστε να αντλεί στοιχεία από τη βάση `StudentsGrades`.
2. Αφού εισάγουμε το εργαλείο προσδεδεμένη λίστα στην εργαλειοθήκη, το χρησιμοποιούμε για να δημιουργήσουμε αντικείμενο ελέγχου, το οποίο ονομάζουμε `StudList`.
3. Στην ιδιότητα **RowSource** του αντικειμένου `StudList` δίνουμε την τιμή `ADOStudents` για να προσδέσουμε τη λίστα στο αντικείμενο ADO. Επίσης, στην ιδιότητα **ListField** δίνουμε την τιμή `STD_Name` για να καθορίσουμε το πεδίο που θα δίνει τις τιμές του στη λίστα.

Εικόνα 27-2. *Η προσδεδεμένη λίστα ενημερώνεται αυτόματα, χωρίς να γραφεί ούτε μια γραμμή κώδικα.*

Αφού εισάγουμε το εργαλείο προσδεδεμένη λίστα στην εργαλειοθήκη, το χρησιμοποιούμε για να δημιουργήσουμε αντικείμενο ελέγχου, το οποίο ονομάζουμε `StudList`.

Στην ιδιότητα **RowSource** του αντικειμένου `StudList` δίνουμε την τιμή `ADOStudents` για να προσδέσουμε τη λίστα στο αντικείμενο ADO. Επίσης, στην ιδιότητα **ListField** δίνουμε την τιμή `STD_Name` για να καθορίσουμε το πεδίο που θα δίνει τις τιμές του στη λίστα.

Data Bound Grid (Προσδεδεμένο πλέγμα): Χρησιμοποιείται για τη δημιουργία πινάκων, στους οποίους προβάλλονται τα δεδομένα του συνόλου εγγραφών του αντικειμένου ADO. Η πληροφορία που περιέχεται στο σύνολο εγγραφών, μεταφέρεται αυτόματα στο πλέγμα. Η εμφάνιση του προσδεδεμένου πλέγματος είναι ίδια με αυτήν των υπολογιστικών φύλλων. Τα πλάτη των στηλών και τα ύψη των γραμμών μεταβάλλονται εύκολα με τη βοήθεια του ποντικιού, όπως και στα φύλλα εργασίας, ενώ με τη βοήθεια αναδυόμενου μενού είναι δυνατές οι λειτουργίες επιλογής, αντιγραφής, διαγραφής,

στηβάση δεδομένων το περιεχόμενό της, ως νέα εγγραφή. Επίσης, αν η ιδιότητα **AllowDelete** έχει την τιμή True και ο χρήστης κάνει κλικ πάνω στο τετραγωνίδιο που βρίσκεται μπρος από την πρώτη στήλη του πλέγματος, επιλέγεται όλη η γραμμή, και αν στη συνέχεια πατήσει το πλήκτρο Delete, η εγγραφή θα διαγραφεί.

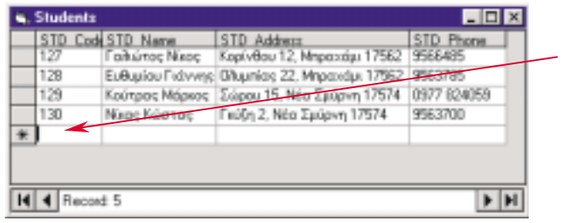
Άσκηση 27-5.

Σε μια φόρμα να τοποθετηθεί ένα προσδεδεμένο πλέγμα, στο οποίο να παρουσιάζονται όλα τα στοιχεία των μαθητών, που είναι καταχωρισμένα στον πίνακα Students της βάσης δεδομένων StudentsGrades.

1. Τοποθετούμε στη φόρμα αντικείμενο ADO, με όνομα ADOStudents, και δίνουμε τιμές στις ιδιότητές του, ώστε να αντλεί στοιχεία από τη βάση StudentsGrades. Για να ενημερώνεται η περιοχή **Caption** με τον αύξοντα αριθμό της τρέχουσας εγγραφής του συνόλου εγγραφών γράφουμε τον κώδικα:

```
Private Sub ADOStudents_MoveComplete(ByVal Reason As ADODB.EventReasonEnum, _
            ByVal pErr As ADODB.Error, Status As ADODB.EventStatusEnum, _
            ByVal pRecordset As ADODB.Recordset)
    ADOStudents.Caption = "Record: " & CStr(ADOStudents.Recordset.AbsolutePosition)
End Sub
```

2. Αφού εισάγουμε το εργαλείο προσδεδεμένο πλέγμα στην εργαλειοθήκη, το χρησιμοποιούμε για να δημιουργήσουμε αντικείμενο ελέγχου, το οποίο ονομάζουμε StudGrid. Στην ιδιότητα **DataSource** του αντικειμένου StudGrid δίνουμε την τιμή ADOStudents.



Εικόνα 27-3. *Τα στοιχεία ενός συνόλου εγγραφών ψαίνονται στο εσωτερικό ενός προσδεδεμένου πλέγματος όπως ακριβώς στα*

Ανακεφαλαίωση

Η γλώσσα ***SQL*** είναι μία γλώσσα με την οποία είναι δυνατόν να γίνει αναζήτηση, εισαγωγή, και ενημέρωση των στοιχείων μιας βάσης δεδομένων. Η βασική εντολή της γλώσσας, η εντολή SELECT, η οποία δίνεται σαν τιμή της ιδιότητας **RecordSource** των αντικειμένων ADO, έχει συνήθως τη μορφή:

SELECT *πεδίο1, πεδίο2, . . .* FROM *πίνακα* WHERE *κριτήρια* ORDER BY *πεδίο*

Για την προβολή δεδομένων από βάσεις δεδομένων, μπορούν να χρησιμοποιηθούν και τα ειδικά αντικείμενα ελέγχου προσδεδεμένη λίστα, προσδεδεμένη συνδυασμένη λίστα και προσδεδεμένο πλέγμα.

Εργαστηριακές ασκήσεις

1. Να εκτελεστεί η άσκηση 27-1.
2. Να εκτελεστεί η άσκηση 27-2 για διαφορετικούς τρόπους σύνταξης της υποεντολής WHERE.
3. Να εκτελεστεί η άσκηση 27-3.
4. Να κατασκευαστεί φόρμα η οποία να περιέχει πλήκτρο του οποίου το πάτημα να παρουσιάζει διαλογικά παράθυρα, που να δέχονται τον αριθμό μητρώου του μαθητή και τον κωδικό μαθήματος. Στη φόρμα να φαίνεται η βαθμολογία του μαθητή στο μάθημα.
5. Να εκτελεστεί η άσκηση 27-4.
6. Να εκτελεστεί η άσκηση 27-5. Να επαναληφθεί η άσκηση κάνοντας χρήση του Data Form Wizard.

Μάθημα 28
Παγίδευση λαθών

Τελειώνοντας αυτό το μάθημα οι μαθητές θα μπορούν:

- Να παγιδεύουν λάθη εκτέλεσης.
- Να δημιουργούν υποροutίνες διαχείρισης λαθών.
- Να καθορίζουν το σημείο επιστροφής στο πρόγραμμα μετά από ένα λάθος εκτέλεσης.

Ένας καλός προγραμματιστής παραδίδει τα προγράμματά του για χρήση, μόνο αφού τα έχει ελέγξει σχολαστικά και είναι σίγουρος ότι είναι απαλλαγμένα από λάθη. Τα δοκιμάζει για διάφορες τιμές και τύπους δεδομένων και ελέγχει όλες τις πιθανές διαδρομές που είναι δυνατόν να ακολουθήσουν οι χρήστες. Όταν όμως ένα πρόγραμμα μπει στην παραγωγή, μπορούν να παρουσιαστούν διάφορα προβλήματα που οφείλονται σε παραλείψεις ή κακούς χειρισμούς του χρήστη, σε αστοχίες των συσκευών του υλικού του υπολογιστικού συστήματος κ.ά. Όλα αυτά μπορεί να διακόψουν την ομαλή εκτέλεση του προγράμματος ως λάθη εκτέλεσης.

Τα λάθη εκτέλεσης μπορούμε να τα παγιδεύσουμε, δηλαδή να τα ανιχνεύσουμε προγραμματιστικά και να τα χειριστούμε κατάλληλα, ώστε, αν είναι δυνατόν, να τα διορθώσουμε ή έστω να σταματήσουμε το πρόγραμμα με τρόπο φιλικό προς το χρήστη. Αυτό το πετυχαίνουμε χρησιμοποιώντας μεθόδους που είναι γνωστές ως μέθοδοι ***διαχείρισης λαθών (error-handling)*** ή ***παγίδευσης λαθών (error trapping)***. Με αυτές τις μεθόδους μπορούμε να εξασφαλίσουμε την ομαλή εκτέλεση ενός προγράμματος, ώστε να μη χάσουμε τα στοιχεία που έχουμε εισάγει ή τους υπολογισμούς που έχουν γίνει πριν την εμφάνιση του λάθους. Μπορούμε, για παράδειγμα, να ανιχνεύσουμε το λάθος που προκαλείται, όταν προσπαθούμε να ανοίξουμε ένα αρχείο σε έναν φάκελο που δεν υπάρχει ή το λάθος που προκαλείται, όταν προσπαθούμε να εισάγουμε συμβολοσειρές σε μια συνάρτηση, που δέχεται αριθμητικές παραμέτρους, και να προχωρήσουμε σε διορθωτικές ενέργειες.

Τις διορθωτικές αυτές ενέργειες τις πραγματοποιούμε στη ρουτίνα ***διαχείρισης λαθών (error-handling routine)***. Στη ρουτίνα αυτή μπορούμε να αντιμετωπίζουμε ένα ή περισσότερα λάθη και να πραγματοποιούμε την κατάλληλη κάθε φορά ενέργεια. Η ενέργεια αυτή δεν είναι κατ' ανάγκη διορθωτική. Υπάρχουν περιπτώσεις που δεν μπορούμε να διορθώσουμε ένα λάθος, μπορούμε όμως να το παρακάμψουμε και να επιτρέψουμε στο πρόγραμμα να συνεχίσει την εκτέλεσή του. Γράφοντας δηλαδή κατάλληλα τη ρουτίνα διαχείρισης λαθών μπορούμε να κάνουμε την εκτέλεση ενός προγράμματος ανεξάρτητη από την ύπαρξη λαθών. Σε άλλες περιπτώσεις πάλι η ρουτίνα διαχείρισης λαθών μπορεί να εντοπίζει απλά το είδος και τη θέση του λάθους και να διακόπτει το πρόγραμμα αφήνοντας στον προγραμματιστή τη φροντίδα για τη διόρθωσή του.

Υπάρχουν δύο τεχνικές με τις οποίες μπορεί να γίνει η διαχείριση των λαθών:

- Η ***επιτόπια διαχείριση λαθών (inline error handling)***
- Η ***διαχείριση λαθών σε περιοχή παραπομπής (exception error handling)***

Η πρώτη τεχνική επιτρέπει στο πρόγραμμα να συνεχίσει με την εντολή που ακολουθεί το σημείο που προκάλεσε το λάθος σα να μη συμβαίνει τίποτε. Είναι θέμα του προγραμματιστή να έχει φροντίσει να επισυνάψει κώδικα διαχείρισης λαθών αμέσως μετά τα σημεία που είναι επιρρεπή σε λάθη. Αντίθετα, στη δεύτερη τεχνική η εμφάνιση του λάθους προκαλεί ένα άλμα σε μια ειδική περιοχή του κώδικα που είναι επιφορτισμένη με τη διαχείριση των λαθών.

Η επιτόπια διαχείριση λαθών

Υπάρχουν περιπτώσεις κατά τις οποίες θέλουμε να διαχειριστούμε ένα λάθος αμέσως μετά την εμφάνισή του. Σε αυτές τις περιπτώσεις πρέπει να φροντίσουμε ώστε:

1. Το πρόγραμμα να συνεχίσει την εκτέλεσή του μετά την εμφάνιση του λάθους
2. Να διαγνωστεί το είδος του λάθους και να αντιμετωπιστεί
3. Να αρθεί η κατάσταση εκτέλεσης υπό συνθήκη λάθους.

Για να δηλώσουμε στο πρόγραμμα ότι πρέπει να συνεχίσει την εκτέλεσή του ακόμη και μετά την εμφάνιση ενός λάθους χρησιμοποιούμε την εντολή `On Error Resume Next`. Η γενική μορφή της εντολής είναι:

```
On Error Resume Next
```

Η δήλωση αυτή πρέπει να υπάρχει σε κατάλληλο σημείο πριν το τμήμα κώδικα που μπορεί να προκαλέσει λάθη. Η πρόταση `On Error Resume Next` δηλώνει ότι μετά την εμφάνιση ενός λάθους ο έλεγχος πηγαίνει στην πρόταση που βρίσκεται αμέσως μετά την εντολή που προκάλεσε το λάθος. Για τη διάγνωση των λαθών η VB διαθέτει το ειδικό αντικείμενο `Err`. Από τις ιδιότητες του αντικειμένου αυτού μπορούμε να διακρίνουμε την αιτία που προκάλεσε ένα λάθος. Όταν προκύπτει ένα λάθος, η ιδιότητα **`Number`** του αντικειμένου `Err` αποκτά ως τιμή τον ***κωδικό του λάθους (error code)***, που χαρακτηρίζει μοναδικά αυτό το λάθος. Παράλληλα η ιδιότητα **`Description`** αποκτά ως τιμή τη συμβολοσειρά που αποτελεί τη σύντομη λεκτική περιγραφή του λάθους. Ας σημειωθεί ότι σε κανονικές συνθήκες η ιδιότητα **`Number`** έχει την τιμή 0. Η άρση της κατάστασης λάθους γίνεται με μηδενισμό του κωδικού λάθους στην ιδιότητα **`Number`**. Γι' αυτό χρησιμοποιείται η μέθοδος **`Clear`** του αντικειμένου `Err`.

Παράδειγμα 28-1.

Θα δημιουργήσουμε μια υπορουτίνα που πραγματοποιεί αντιγραφή του περιεχομένου ενός αρχείου σε ένα άλλο. Η υπορουτίνα δέχεται ως παραμέτρους τα ονόματα των δύο αρχείων. Αν για κάποιο λόγο η αντιγραφή δεν είναι δυνατή - για παράδειγμα λάθος ονομασία των αρχείων, οι περιφερειακές συσκευές δεν είναι σε λειτουργία ή δεν περιέχουν μονάδες αποθήκευσης κ.ά. - δε θα διακόπτεται η λειτουργία του προγράμματος, αλλά θα παρουσιάζεται σε πλαίσιο διαλόγου η αιτία του λάθους και το πρόγραμμα θα συνεχίζει την εκτέλεσή του. Ο κώδικας της υπορουτίνας είναι:

```
Sub MyFileCopy(ByVal SourceFileName, ByVal DestinationFileName)
    On Error Resume Next        'Óå ðåñßðôùóç ëÜèïõò óõíÝ÷éóå
    FileCopy SourceFileName, DestinationFileName
    If Err.Number <> 0 Then      'Áí Ý÷åé ðñïêýøåé ëÜèïò ôüôå ...
        MsgBox Err.Description, vbCritical, "ÁíôéãñáöÞ Áñ÷åßùí"
        Err.Clear                'ÊáèÜñéóå ôïí êùäéêü ëÜèïõò
    End If
End Sub
```

Με την είσοδο στην υπορουτίνα ενεργοποιείται η παγίδευση λαθών. Αν η υπορουτίνα `FileCopy` της VB που αντιγράφει το περιεχόμενο του αρχείου `SourceFileName` στο `DestinationFileName` υποπέσει για κάποια αιτία σε λάθος, διακόπτεται η εκτέλεσή της και η εκτέλεση του προγράμματος συνεχίζει με την αμέσως επόμενη εντολή. Όμως η επόμενη εντολή ελέγχει αν υπάρχει εκκρεμής κωδικός λάθους και αν υπάρχει παρουσιάζει πλαίσιο διαλόγου με την περιγραφή του και καθαρίζει τον κωδικό του λάθους.

Παράδειγμα 28-2.

Θα δημιουργήσουμε μια υπορουτίνα για διαγραφή των αρχείων ενός φακέλου. Η υπορουτίνα δέχεται ως παράμετρο το όνομα του φακέλου. Αν για κάποιο λόγο η διαγραφή ενός αρχείου δεν είναι δυνατή, να εμφανίζεται κατάλληλο μήνυμα λάθους.

```
Sub DeleteAllFiles(ByVal PathName)
    Dim CurrentFile              As String
    On Error Resume Next
    CurrentFile = Dir(PathName & "*.*", vbNormal)  'Ðñþôï áñ÷åßï
    If Err.Number <> 0 Then              'ËÜèïò óôïí öÜêåëï
        MsgBox Err.Description, vbCritical, "ÄéáãñáöÞ Áñ÷åßùí"
        Err.Clear
        Exit Sub
    End If
```



```
    Do While CurrentFile <> ""
        CurrentFile = PathName & CurrentFile
        Kill CurrentFile
        If Err.Number <> 0 Then       'Áäýíáìßá äéáãñáöÞò
            MsgBox Err.Description, vbCritical, "Äéáãñáöþ Áñ÷åßùí"
            Err.Clear
        End If
        CurrentFile = Dir()           'Åðüìåíï áñ÷åßï
    Loop
End Sub
```

Σ' αυτό το παράδειγμα είναι πιο εμφανής η αναγκαιότητα του μηδενισμού του κωδικού λάθους με την εντολή `Err.Clear`. Αν μετά από ένα λάθος δε γίνει μηδενισμός, στην επόμενη ανακύκλωση του βρόχου θα βρεθεί η `Err.Number <> 0` και θα παρουσιάζεται το ίδιο πλαίσιο διαλόγου ξανά και ξανά χωρίς λόγο.

Παρατηρούμε ότι η τεχνική της επιτόπιας διαχείρισης λαθών μας υποχρεώνει να δημιουργούμε ελέγχους σε κάθε σημείο του κώδικα. Αυτό έχει ως αποτέλεσμα να δημιουργούνται μεγάλοι κώδικες. Υπάρχουν όμως περιπτώσεις που μπορούμε να αγνοήσουμε τα επιμέρους λάθη και να τα διαχειριστούμε σαν ένα λάθος. Η ιδιότητα **`Number`**, αν δεν έχει μηδενιστεί, περιέχει το πιο πρόσφατο λάθος.

Παράδειγμα 28-3.

Συμπτύσουμε τον κώδικα του πιο πάνω παραδείγματος σε:

```
Sub DeleteAllFiles(ByVal PathName)
    Dim CurrentFile              As String
    On Error Resume Next
    CurrentFile = Dir(PathName & "*.*", vbNormal)  'Ðñþôï áñ÷åßï
    Do While CurrentFile <> ""
        CurrentFile = PathName & CurrentFile
        Kill CurrentFile
        CurrentFile = Dir()          'Åðüìåíï áñ÷åßï
    Loop
'Âê ôùí ðóôÝñùí ïëéêÞ äéá÷åßñéóç ëáèþí
    If Err.Number <> 0 Then                  'ËÜèïò ïðïéïäþðïôå ôýðïõ
        MsgBox "ËÜèïò êáôÜ ôç äéáãñáöÞ", vbCritical, "Äéáãñáöþ Áñ÷åßùí"
        Err.Clear
    End If
End Sub
```

Η τεχνική αυτή είναι γνωστή ως εκ των ***υστέρων διαχείριση λαθών*** ή ως ***διαχείριση λαθών με καθυστέρηση (Delayed error handling)***.

Ας σημειωθεί ότι η εμβέλεια της εντολής `On Error Resume Next` περιορίζεται μέσα στα όρια της υπορουτίνας ή της συνάρτησης που έχει γραφεί. Αυτό σημαίνει ότι γίνεται ανενεργή όταν καλείται μια άλλη διαδικασία. Τέλος, αναφέρουμε ότι η ιδιότητα **`Number`** μηδενίζεται και χωρίς τη μεσολάβηση της μεθόδου **`Clear`**, αν εκτελεστεί εντολή `Exit Sub` ή `Exit Function`.

Η διαχείριση λαθών σε περιοχή παραπομπής

Σε μια υπορουτίνα ή συνάρτηση που χρησιμοποιεί την τεχνική της παγίδευσης λαθών και διαχείρισής τους σε περιοχή παραπομπής πρέπει να περιέχονται:

- Εντολή που ενεργοποιεί την παγίδευση λαθών και υποδεικνύει τη ρουτίνα διαχείρισης λαθών (εντολή `On Error GoTo` *ετικέτα*).
- Προαιρετικά, εντολή που απενεργοποιεί την παγίδευση λαθών (εντολή `On Error GoTo 0`).
- Ρουτίνα στην οποία θα μεταβιβάζεται ο έλεγχος μόλις ανιχνευθεί ένα λάθος - πρόκειται για τη ***ρουτίνα διαχείρισης λαθών***.

Η ρουτίνα διαχείρισης λαθών πρέπει:

- Να αρχίζει με μια ετικέτα που συμφωνεί με εκείνην της πρότασης `On Error GoTo`.
- Να περιέχει διαδικασίες που να λαμβάνουν τον κωδικό του λάθους από την ιδιότητα **`Number`** του αντικειμένου `Err`, διαδικασίες που να πραγματοποιούν διάγνωσή του και διαδικασίες που να εκτελούν κατά περίπτωση την κατάλληλη διορθωτική ενέργεια.
- Να περιέχει τουλάχιστον μια εντολή `Resume` για επιστροφή στο κύριο μέρος του κώδικα.

Η ρουτίνα διαχείρισης λαθών δεν είναι μια υποροutίνα `Sub` ή μια συνάρτηση `Function`, αλλά ένα τμήμα κώδικα μέσα σε μια `Sub` ή `Function`, που χαρακτηρίζεται από μια ετικέτα. Αυτό το τμήμα κώδικα πρέπει να τοποθετείται σε τέτοια θέση μέσα στην υποροutίνα ή τη συνάρτηση ώστε να μην εκτελείται κατά την κανονική ροή εκτέλεσής της. Μια κοινή σύμβαση είναι να τοποθετείται στο τέλος της διαδικασίας, αμέσως μετά από μια πρόταση `Exit Sub` ή `Exit Function`.

Η γενική δομή μιας υποροutίνας που χρησιμοποιεί παγίδευση λαθών είναι η εξής:

```
Sub όνομα_υποροutίνας
        εντολή_1
        εντολή_2
        :
        On Error GoTo ετικέτα      <-- Ενεργοποίηση παγίδευσης λαθών
        εντολή_ν
        εντολή_ν+1                 <-- Άλμα στην περίπτωση που προκύψει κάποιο λάθος
        :
        Exit Sub

    ετικέτα:                       <-- Επιστροφή μετά από την αντιμετώπιση του λάθους
        εντολή_μ
        εντολή_μ+1
        :
        Resume ...
End Sub
```

Ενεργοποίηση, απενεργοποίηση παγίδευσης λαθών

Για να ενεργοποιήσουμε την παγίδευση λαθών και να καθορίσουμε τη ρουτίνα διαχείρισής τους χρησιμοποιούμε την εντολή `On Error GoTo`. Η γενική μορφή της είναι:

```
On Error GoTo ετικέτα
```

Η ενεργοποίηση παγίδευσης λαθών πρέπει να γίνεται σε κατάλληλο σημείο πριν το τμήμα κώδικα που μπορεί να προκαλέσει λάθη. Από τη στιγμή που ενεργοποιείται η παγίδευση, κάθε λάθος που προκύπτει προκαλεί τη μεταφορά του ελέγχου στην ετικέτα που καθορίζεται με την εντολή. Το όνομα της ετικέτας πρέπει να επιλέγεται όσο το δυνατόν πιο περιγραφικό. Ακόμα, η ετικέτα πρέπει να βρίσκεται μέσα στην υποροutίνα ή τη συνάρτηση που αφορά (δεν μπορούμε να χρησιμοποιήσουμε την πρόταση `On error GoTo` *ετικέτα* για να μεταφέρουμε τον έλεγχο σε άλλη υποροutίνα ή συνάρτηση). Η παγίδευση παραμένει ενεργοποιημένη όσο η διαδικασία που την περιέχει είναι ενεργή.

Αν θέλουμε να απενεργοποιήσουμε την παγίδευση σε συγκεκριμένο σημείο της διαδικασίας (πριν να τελειώσει η εκτέλεσή της) χρησιμοποιούμε την πρόταση:

```
On error GoTo 0
```

Αυτήν τη δυνατότητα ενεργοποίησης-απενεργοποίησης της παγίδευσης λαθών τη χρησιμοποιούμε στην περίπτωση που μόνο ένα τμήμα του κώδικα της υποροutίνας ή της συνάρτησης είναι πιθανόν να προκαλέσει λάθη κατά το χρόνο εκτέλεσης.

Μέσα στην ίδια υποροutίνα μπορεί να υπάρχουν διαφορετικές εντολές `On Error GoTo`, οι οποίες, κατά περίπτωση, να μεταφέρουν τον έλεγχο σε διαφορετικές ρουτίνες. Διαφορετικά είδη λαθών χρειάζονται και διαφορετικό τρόπο αντιμετώπισης.

Διάγνωση λάθους

Η ρουτίνα διαχείρισης λαθών πρέπει να μπορεί να αναγνωρίζει τα λάθη για να τα χειρίζεται ανάλογα. Η διάγνωση των λαθών γίνεται με τη βοήθεια των ιδιοτήτων του αντικειμένου `Err`.

Επιστροφή από τη ρουτίνα διαχείρισης λαθών

Η επιστροφή από μια ρουτίνα διαχείρισης λαθών γίνεται με μια εντολή `Resume`. Η εντολή αυτή μηδενίζει αυτόματα την ιδιότητα **`Number`** του αντικειμένου `Err`, οπότε δε χρειάζεται να εκτελεστεί και μέθοδος `Clear`. Η εντολή `Resume` μπορεί να έχει μια από τις μορφές:

```
Resume
Resume Next
Resume ετικέτα
```

ανάλογα με το σημείο στο οποίο θέλουμε να μεταφερθεί ο έλεγχος.

Με την πρώτη μορφή (απλή `Resume`) ο έλεγχος επιστρέφει ακριβώς στην εντολή που προκάλεσε το λάθος. Αυτήν τη μορφή την χρησιμοποιούμε όταν η ρουτίνα διαχείρισης λαθών μπορεί να άρει την αιτία που προκάλεσε το λάθος. Έτσι, η εκτέλεση συνεχίζεται κανονικά σα να μη μεσολάβησε κανένα λάθος.

Με την εντολή `Resume Next` ο έλεγχος επιστρέφει στην εντολή που βρίσκεται αμέσως μετά από αυτήν που προκάλεσε το λάθος. Αυτήν τη μορφή τη χρησιμοποιούμε όταν η ρουτίνα διαχείρισης λαθών δεν μπορεί ή δεν επιθυμούμε να άρει το λάθος. Στην ουσία λοιπόν ζητάμε την παράκαμψή του.

Τέλος, με την εντολή `Resume` *ετικέτα* ο έλεγχος μεταφέρεται σε άλλα σημεία της υποροutίνας ή της συνάρτησης. Αυτήν τη μορφή τη χρησιμοποιούμε όταν θέλουμε η εμφάνιση ενός συγκεκριμένου τύπου λάθους να αλλάξει ριζικά τη ροή εκτέλεσης της διαδικασίας.

Παράδειγμα 28-4.

Θα δημιουργήσουμε μια υποροutίνα που πραγματοποιεί αντιγραφή των αρχείων ενός φακέλου σε έναν άλλο φάκελο. Η υποροutίνα δέχεται ως παραμέτρους τα ονόματα των δύο φακέλων. Αν για κάποιο λόγο η αντιγραφή δεν είναι δυνατή, για παράδειγμα λάθος ονομασία φακέλων, οι περιφερειακές συσκευές δεν είναι σε λειτουργία ή δεν περιέχουν μονάδες αποθήκευσης κ.ά. δε θα διακόπτεται η λειτουργία του προγράμματος αλλά θα παρουσιάζεται σε πλαίσιο διαλόγου η αιτία του λάθους και το πρόγραμμα θα συνεχίζει την εκτέλεσή του ανάλογα με την επιλογή του χρήστη. Ο χρήστης πρέπει να έχει τη δυνατότητα να ζητήσει να επαναληφθεί η προσπάθεια (Retry), να μη γίνει η αντιγραφή του συγκεκριμένου αρχείου (Ignore), να διακοπεί η εκτέλεση του προγράμματος. Ο κώδικας της υποροutίνας είναι:

```
Sub CopyAllFiles(ByVal SourcePathName, ByVal DestinationPathName)
   'Êïëíïß êùäéêïß ëáèþí ãéá áñ÷åßá
    Const Err_BadFileNameOrNumber = 57
    Const Err_DeviceIO = 57
    Const Err_DiskFull = 61
    Const Err_DeviceUnavailable = 68
    Const Err_DiskNotReady = 71
    Const Err_PathDoesNotExist = 76
    Const Err_PermissionDenied = 70
    Dim CurrentFile     As String   'Ïíïìá áñ÷åßïõ
    Dim MsgStr          As String   'Ìçíýìá óôïí ÷ñþóôç
    Dim Response        As Long     'Áðüíôçóç ÷ñþóôç óå ðëáßóéï äéáëüãïõ
    On Error GoTo Err_Handler       'Åíåñãïðïßçóç ñïõôßíáò äéá÷åßñéóçò ëáèþí
    CurrentFile = Dir(SourcePathName & "*.*", vbNormal)
    Do While CurrentFile <> ""
        FileCopy SourcePathName & CurrentFile, _
                 DestinationPathName & CurrentFile
        CurrentFile = Dir()
    Loop
Exit_CopyAllFiles:
Exit Sub
```

```
'Ñïõôßíá äéá÷åßñéóçò ëάèþí
Err_Handler:
   'ÁíÙëïãά lå ôï åßäïò ôïõ ëÙèïõò åìöÙíéóå êáé êáôÙëëçëï ìÞíõìá
    Select Case Err.Number
         Case Err_BadFileNameOrNumber
              MsgStr = "Åóöáëèýíï üíïìá áñ÷åßïõ"
         Case Err_DeviceIO
              MsgStr = "ÓöÙëìá Åéóüäïõ/Âîüäïõ"
         Case Err_DiskFull
              MsgStr = "Ï äßóêïò åßíáé ðëÞñçò"
         Case Err_DeviceUnavailable
              MsgStr = "Ç óõóêåõÞ äåí åßíáé äéáèýóéìç"
         Case Err_DiskNotReady
              MsgStr = "Ï äßóêïò äåí åßíáé Ýôïéìïò"
         Case Err_PathDoesNotExist
              MsgStr = "Ï öÙêåëïò äåí õðÙñ÷åé"
         Case Err_PermissionDenied
              MsgStr = "Äåí õðÙñ÷åé Üäåéá ÷ñÞóçò"
         Case Else
              MsgStr = Err.Description
    End Select
   'ÓôïìÞíìá ðñüèåóå ôï üíïìá ôïõ áñ÷åßïõ
    MsgStr = "Áñ÷åßï: " & CurrentFile & vbCrLf & MsgStr
    Response = MsgBox(MsgStr, vbExclamation + vbAbortRetryIgnore, _
                                                        "ÁíôéãñáöÞ ÖáêÝëïõ")
   'ÁíÙëïãά lå ôçí áðÙíôçóç ôïõ ÷ñÞóôç åðÝóôñåøå
    Select Case Response
         Case vbRetry
              Resume                    'ÅðáíÙëçøç ôçò åíôïëÞò
         Case vbIgnore
              Resume Next               'ÐáñÙêáìøç ôçò åíôïëÞò
         Case vbAbort
              Resume Exit_CopyAllFiles  'Âîïäïò áðü ôç ñïõôßíá
    End Select
End Sub
```

Μέσα σε μια ρουτίνα διαχείρισης λαθών δεν έχει ισχύ η εντολή `On Error GoTo` για παραπομπή σε άλλη ρουτίνα διαχείρισης λαθών. Αν προκληθεί λάθος μέσα στη ρουτίνα διαχείρισης λαθών, το λάθος μεταβιβάζεται για διαχείριση στην καλούσα υπορουτίνα ή τη συνάρτηση. Στην περίπτωση που δεν υπάρχει καλούσα υπορουτίνα εμφανίζεται πλαίσιο διαλόγου με το λάθος και προκαλείται διακοπή του προγράμματος. Γι' αυτό το λόγο και οι ρουτίνες διαχείρισης λαθών πρέπει να είναι γραμμένες κατά τον απλούστερο δυνατό τρόπο και με εντολές που δεν προκαλούν λάθη.

Πρόκληση τεχνητού λάθους

Η VB μας δίνει τη δυνατότητα να προσομοιώσουμε την κατάσταση που προκαλεί ένα λάθος. Ακόμα, μας επιτρέπει να ορίσουμε δικούς μας κωδικούς λάθους για προγραμματιστικά και αλγοριθμικά λάθη, τα οποία δεν είναι της δικαιοδοσίας της και που εξαρτώνται από τη λογική του προγράμματος και την εφαρμογή. Η πρόκληση του τεχνητού λάθους γίνεται με τη μέθοδο **`Raise`** του αντικειμένου `Err`, η οποία συντάσσεται ως εξής:

`Err.Raise` *κωδικός_λάθους*

Αν στον κωδικό_λάθους δώσουμε μια τιμή που συμπίπτει με κάποιον από αυτούς που ήδη χρησιμοποιεί η VB, η μέθοδος `Raise` προσομοιώνει την εμφάνιση του συγκεκριμένου λάθους (προκαλούμε στο σύστημα την εντύπωση ότι προέκυψε το λάθος χωρίς αυτό να ανταποκρίνεται στην πραγματικότητα). Με αυτόν τον τρόπο, κατά τη διαδικασία εκσφαλμάτωσης, προκαλούμε την κλήση της ρουτίνας διαχείρισης λαθών για το συγκεκριμένο λάθος, με σκοπό να ελέγξουμε την ορθότητά της.

Για να ορίσουμε τους δικούς μας κωδικούς λάθους δεν έχουμε παρά να δώσουμε στον *κωδικό_λάθους* μια τιμή διάφορη από όλους τους κωδικούς που χρησιμοποιεί η VB. Προτείνεται, οι νέοι κωδικοί λαθών να έχουν αρκετά μεγάλη τιμή, γύρω στο 32000.

Παράδειγμα 28-5.

Ο κωδικός λάθους 6 αντιστοιχεί σε υπερχείλιση (overflow) ενός τύπου δεδομένων (π.χ. σε ακέραιο τύπο τιμή πάνω από 32767). Αυτόν τον κωδικό μπορούμε να χρησιμοποιήσουμε και στην περίπτωση που ένας χρήστης δίνει υπερβολικά μεγάλες τιμές για μια ποσότητα π.χ. το βάρος ή το ύψος ενός ανθρώπου. Οι εντολές:

```
If  Age < 0 Or Age > 120 Then Err.Raise 6
If  Weight < 0 Or Weight > 180 Then Err.Raise 6
```

προκαλούν λάθος υπερχείλισης το οποίο το διαχειρίζεται η ρουτίνα διαχείρισης λαθών σαν να πρόκειται για πραγματική υπερχείλιση μιας μεταβλητής.

Ανακεφαλαίωση

Τα λάθη εκτέλεσης, μπορούμε να τα ανιχνεύσουμε προγραμματιστικά και να τα χειριστούμε κατάλληλα, ώστε, αν είναι δυνατόν, να τα διορθώσουμε ή έστω να σταματήσουμε το πρόγραμμα με τρόπο φιλικό προς το χρήστη. Αυτό το πετυχαίνουμε χρησιμοποιώντας μεθόδους που είναι γνωστές ως ***διαχείρισης λαθών*** ή ***παγίδευσης λαθών***. Υπάρχουν δύο τεχνικές με τις οποίες μπορεί να γίνει η διαχείριση των λαθών.

Στην τεχνική της ***επιπόπλας διαχείρισης λαθών*** φροντίζουμε ώστε, χρησιμοποιώντας την εντολή `On Error Resume Next`, το πρόγραμμα να συνεχίσει την εκτέλεσή του μετά την εμφάνιση του λάθους. Επίσης, με την ιδιότητα **`Err.Number`** ανιχνεύουμε το είδος του λάθους και προκαλούμε την κατάλληλη διορθωτική ενέργεια και με τη μέθοδο **`Err.Clear`** αίρουμε την κατάσταση εκτέλεσης υπό συνθήκη λάθους.

Αντίθετα, στην τεχνική διαχείρισης λαθών σε περιοχή παραπομπής χρησιμοποιείται η εντολή `On Error GoTo` *ετικέτα* για να ενεργοποιηθεί η παγίδευση λαθών και να υποδειχθεί η ρουτίνα διαχείρισης λαθών. Στη ρουτίνα διαχείρισης λαθών με την ιδιότητα **`Err.Number`** ανιχνεύεται το είδος του λάθους και γίνονται οι διορθωτικές ενέργειες. Η επιστροφή στον κώδικα που προκάλεσε το λάθος γίνεται με την εντολή `Resume`. Χρησιμοποιώντας τη μέθοδο **`Err.Raise`** μπορούμε να προκαλέσουμε τεχνητή κλήση της ρουτίνας διαχείρισης λαθών.

Εργαστηριακές Ασκήσεις

1. Σε μια φόρμα τοποθετήστε δύο πλαίσια κειμένου. Στο ένα ο χρήστης θα γράφει το πλήρες όνομα ενός αρχείου που θέλουμε να αντιγράψουμε και στο άλλο το πλήρες όνομα του αρχείου στο οποίο θέλουμε να γίνει η αντιγραφή. Πατώντας ένα πλήκτρο διαταγής να καλείται η υπορουτίνα του παραδείγματος 28-1. Προκαλέστε διαφόρους τύπους λαθών, όπως κακή ονομασία αρχείων, λάθος φάκελο, ανύπαρκτο αποθηκευτικό μέσο.
   Υπόδειξη: Η πλήρης μορφή των ονομάτων των αρχείων είναι (δίσκος:\φάκελος\υποφάκελος\αρχείο.επέκταση).
2. Σε μια φόρμα τοποθετήστε ένα πλαίσιο κειμένου, στο οποίο ο χρήστης θα γράφει το πλήρες όνομα ενός φακέλου (το όνομα να τερματίζεται με (\)).Πατώντας ένα πλήκτρο διαταγής να καλείται η υπορουτίνα του παραδείγματος 28-2. Προκαλέστε διαφόρους τύπους λαθών. Δοκιμάστε να εφαρμόσετε τη διαδικασία στο αρχείο `C:\Windows\Temp`.
3. Γράψτε συνάρτηση τύπου `Long`, η οποία να υπολογίζει το τετράγωνο ενός αριθμού τύπου `Long`. Στην περίπτωση που ο αριθμός είναι τόσο μεγάλος, ώστε να προκαλεί λάθος υπερχείλισης (κωδικός λάθους 6) να εμφανίζεται πλαίσιο διαλόγου με κατάλληλο μήνυμα.
4. Γράψτε συνάρτηση, που να υπολογίζει την τετραγωνική ρίζα ενός αριθμού. Σε περίπτωση που ο αριθμός είναι αρνητικός να εμφανίζει πλαίσιο διαλόγου με κατάλληλο μήνυμα.
5. Η συνάρτηση `Shell` ζητά την εκτέλεση ενός εκτελέσιμου αρχείου. Π.χ. η εντολή:

   ```
   AppID=Shell("Pbrush", vbNormalFocus)
   ```

   ζητά την εκτέλεση του προγράμματος της Ζωγραφικής (Paint ή Paint Brush).
   Γράψτε πρόγραμμα που να ζητά το όνομα ενός εκτελέσιμου αρχείου και να εντολοδοτεί την εκτέλεσή του. Αν το αρχείο δεν υπάρχει ή αν προκαλείται οποιασδήποτε μορφής λάθος κατά την κλήση του, να εμφανίζεται διαγνωστικό μήνυμα.
6. Ενσωματώστε την υπορουτίνα 28-4 σε πρόγραμμα και πραγματοποιήστε δοκιμές.