

# Tutorial: Working with Files



**The objective of this tutorial is to learn what the differences between the different file types are and how to use files. At the end of this tutorial you should be able to:**

* Select an appropriate file type for any situation.
* Open a file for input or output.
* Read or write to a file.
* Close a file correctly.

## ***File Types***

Visual Basic has three built-in ways of accessing files (excluding the Data control which accesses databases):

1. **Sequential access** to files reads and writes data line by line. Storage is efficient but accessing data at different sections of the file is harder because of its sequential access mechanisms.
2. **Random access** reads and writes data record by record to any location within the file. This type of access is convenient when the file contains homogeneous type data. However, storage is less efficient because padding is inserted to make up field lengths.
3. **Binary access** to files reads and writes data byte by byte to any location within the file. This type of access if efficient in terms of storage, but is the hardest to program.

## ***Task 1 (Sequential Access)***

Often in an application there are various features of the software which can be altered and set by the user. Because these alterations are made a run-time, and not design-time, they will be lost when the application is terminated. Thus, to make these changes persist the application must store the information about these alteration somewhere outside itself. One way is to use a sequential access file which is created when the application terminates, and is read in when the application starts the next time. Into this file can be put all information relating to font sizes, colours, window positions, states (maximised, minimised, etc.).

1. Add the ability to store settings to the ‘Language Tutor’ application created as part of the [Menus and List Manipulation](http://docs.google.com/tutorial_5.html) tutorial.
2. Using the ‘Form\_Unload’ event write a routine to open a suitably named file and store all the current settings.  
    **Dim intFileNum As Integer  
      
    intFileNum = FreeFile  
    Open "A:\SETTINGS.DAT" For Output As intFileNum  
    Print #intFileNum, lstEnglish.FontName  
    Print #intFileNum, lstEnglish.BackColor  
    ...  
    Close intFileNum ' Important to close the file**
3. Add to the ‘Form\_Load’ event a routine to open the same file created in 2, read the contents, then reset the values of different controls where necessary to restore the previous state of the application.  
    **Dim intFileNum As Integer  
    Dim vntFileLine ' declare as a variant to handle different types of data  
      
    FileNum = FreeFile  
    Open "A:\SETTINGS.DAT" For Input As intFileNum  
    Input #FileNum vntFileLine ' read first line in file  
    lstEnglish.FontName = vntFileLine  
    Input #FileNum vntFileLine ' read next line in file  
    lstEnglish.BackColor = vntFileLine  
    ...  
    Close intFileNum ' Important to close the file**  
   **Note:** the variable ‘temp’ is used because control properties cannot be used directly with an input statement. Thus, each line from the file is read into ‘temp’ and then ‘temp’ is assigned in the usual way to the control property.

## ***Task 2 (Random Access)***

The disadvantage with sequential access files is to get to line 20, for example, the program has to read lines 1 to 19 first. This problem does not occur with random access files. The program can read or write directly to any part of the file at random. However, creating a file the programmer needs to do a bit more work. The structure of the data must be explicitly specified before using a random access file. In this second task we will create a small database of information holding information about names, phone numbers, ages, and space for comments.

1. Define the structure of the random access file. Place the following code in a code module (MODULE1.BAS):  
    **Option Explicit  
     
    Type PersonData  
    FirstName As String \* 20  
    LastName As String \* 25  
    PhoneNo As String \* 14  
    Age As Integer  
    Comments As String \* 100  
    End Type**  
   **Note:** The variables of type String are followed by ‘\*’ and a number to show their assign length measured in characters.
2. The above code defines a new data type called ‘PersonData’ which can then be used to declare instances of that type. To do this place the following code in the general declarations section of the main form:  
    **Option Explicit  
    Dim udtPerson As PersonData**  
   This creates a new variable called ‘Person’. However, it is not of the usual types, such as String, Integer, Long, etc, it is of a new type called PersonData.
3. Create suitable label and text controls on the main form so that the user can edit information about each person in the database.
4. To assign the data held in the newly created controls to the variable Person the following lines can be used:  
    **udtPerson.FirstName = txtFirstName.Text  
    udtPerson.LastName = txtLastName.Text  
    udtPerson.PhoneNo = txtPhoneNo.Text**  
   **Note:** the variable Person is followed by a ‘.’ and then a field name which matches the same names that were used in the Type command in MODULE1.BAS. It should be remembered that Person is not a simple variable type like a String, it is a composite variable made up of 5 different basic data types.
5. To save the data a file must first be opened. However, unlike sequential files, we now need to tell VB the length in characters of each record. Errors made here will have peculiar consequences! For the personal details database the length is (20 + 25 + 14 + 2 + 100). This is the length of the strings added together plus 2 which is the number of characters an Integer requires. If you are unsure about the exact length of a record type, the command ‘Len’ can be used to automatically do the calculation (see code below):  
    **Dim intFileNum As Integer  
    Dim intPosition As Integer  
      
    intFileNum = FreeFile  
    intPosition = 1 ' Determines which record will be accessed  
      
    Open "A:\PERSONAL.DAT" For Random As intFileNum Len = Len(udtPerson)  
    Put #intFileNum, intPosition, udtPerson  
    Close #intFileNum**  
   To write data to the other records in the file change the value of the variable ‘Position’. If a file contains 5 records and then a new one is added at position 7 garbage data will pad out record 6!
6. To read in data from a random access file and assign it to various controls use the following lines after having opened the file:  
    **Open "A:\PERSONAL.DAT" For Random As intFileNum Len = Len(udtPerson) Get #intFileNum, intPosition, udtPerson  
    txtFirstName.Text = udtPerson.FirstName  
    txtLastName.Text = udtPerson.LastName  
    ...  
    Close #intFileNum**  
   To determine how many records there are in a file the following line can be used:  
     
    **intRecordNumber = FileLen("A:\PERSONAL.DAT") / Len(udtPerson)**  
     
   This code works by dividing the total size of the file in bytes by the record size.

 [Tutorial 8](http://docs.google.com/tut8.html)(Working with Multiple Forms)

 [Menu](http://docs.google.com/dvbhome.html)

 [Tutorial 10](http://docs.google.com/tut8b.html) (Animation and Graphics)