# فصل ۲

# پرونده‌ها

**هدف‌های رفتاری:** پس از آموزش این فصل هنرجو می‌تواند :

- انواع پرونده‌های ترتیبی و تصادفی را شرح دهد و با انواع پرونده‌های ترتیبی و تصادفی کار کند.
- چگونگی استفاده از شمارهٔ پرونده را شرح دهد و به کار ببرد.
- از دستورهای#Get، #Read، #Write، #Print و #Put در پرونده‌ها استفاده کند.
- کنترل‌های مربوط به پرونده را در برنامه‌های خود به‌کار گیرد.
- اهمیت کاربرد پرونده‌ها را بیان کند.
- چگونگی باز کردن پرونده‌ها را بیان کرده و انجام دهد.
- پرونده‌ها را در سه حالت مختلف باز کند.

## ۱ـ۲ـ آشنایی با مفهوم پایداری

برای اینکه برنامه‌ای بتواند اطلاعات را از جلسه‌ای به جلسهٔ کاری دیگری نگه دارد، باید توانایی ذخیره‌سازی داده‌ها روی دیسک سخت را داشته باشد. در غیر این صورت، هنگامی که برنامهٔ کاربردی بسته شود، تمام داده‌های برنامه که در حافظه اصلی هستند از بین می‌روند.

بنابراین، برای اینکه داده‌ها پایدار باشند، باید برنامه توانایی ذخیره و بازیابی داده‌ها در دیسک سخت را داشته باشد.

می‌توان با چندین روش، عملیات ذخیره و بازیابی داده‌ها را انجام داد. از پرونده دودویی یا متنی می‌توان برای ذخیرهٔ اطلاعات با اندازه و قالب متفاوت استفاده کرد.

## ۲ـ۲ـ کارکردن با پرونده‌ها برای ذخیره و بازیابی داده‌ها

داده‌ها در حافظهٔ اصلی و پرونده‌ها روی حافظه جانبی (دیسک‌ها) ذخیره می‌شوند. هرگز برنامه‌ها به‌طور مستقیم با یک پرونده روی دیسک کار نمی‌کنند.

برنامه از سیستم عامل می‌خواهد که واسطی بین دیسک و برنامه ایجاد کند (به عبارت دیگر، داده‌های مورد نیاز برنامه به کمک پردازنده و سیستم عامل به حافظهٔ اصلی بارگذاری می‌شوند و سپس برنامه اجرا می‌شود).

موقعیت (محل) یک پرونده روی دیسک را می‌توان با به دست آوردن شماره پرونده (file handle) از طریق سیستم عامل، پیدا کرد. از تابع FreeFile برای به‌دست آوردن شمارهٔ پرونده از طریق سیستم عامل استفاده می‌شود. بعد از به‌دست آوردن این شماره، دستور Open برای دسترسی به پرونده برای عملیات خواندن و نوشتن به کار می‌رود. برای نوشتن در یک پرونده، می‌توان از دستور Print (یا Write) و برای خواندن خطوطی از داده‌ها از پرونده روی دیسک، از دستور Line Input استفاده می‌شود. شکل ۱ـ۲ این مفهوم را شرح می‌دهد.


**شکل ۱ـ۲ـ نوشتن در یک پرونده، برعکس خواندن از پرونده است. برای این کار نیاز به دسترسی به File handle و استفاده از دستور Open دارید.**

پرونده‌ها سه نوع هستند : ترتیبی (sequential)، تصادفی (random) و دودویی (binary)[1].
در فایل‌های ترتیبی اطلاعات به همان ترتیبی که در فایل ذخیره می‌شوند قابل بازیابی هستند ولی در فایل‌های تصادفی نوشتن و بازیابی اطلاعات از هر نقطهٔ فایل امکان‌پذیر است. پرونده ترتیبی ساده‌ترین نوع پرونده است اما دارای معایبی است که یکی از آن‌ها کند بودن کار با این نوع پرونده‌ها است. کار با پرونده‌های تصادفی سرعت بیشتری دارد اما پیچیدگی بیشتری را به برنامه تحمیل می‌کند. پرونده‌های دودویی نوع خاصی از پرونده‌های تصادفی هستند.

## ۳ـ۲ـ دستور Open

از دستور Open می‌توان برای ذخیره و بازیابی داده‌ها از یک پرونده روی دیسک استفاده کرد.
شکل کلی دستور Open به صورت زیر است :

Open *FilePath* For *Mode* As [#]*FileNumber* [Len *CharInBuffer*٪]

در این دستور :

- Open نام دستور است.
- *FilePath* محل دقیق پرونده برای خواندن یا ذخیره کردن که شامل درایو و مسیر است.
- For کلید واژه‌ای است که حالت پرونده به دنبال آن ارایه می‌شود.
- *Mode* حالت دسترسی به پرونده است (جدول ۱ـ۲).

**جدول ۱ـ۲ـ حالت‌های دسترسی به پرونده**

| نوع پرونده | حالت | شرح |
|---|---|---|
| ترتیبی | Append | اضافه کردن داده‌ها به انتهای یک پرونده ترتیبی موجود در صورتی که پرونده موجود نباشد، آن را ایجاد خواهد کرد |
| ترتیبی | Input | پرونده ترتیبی را برای خواندن باز می‌کند |
| ترتیبی | Output | پرونده ترتیبی را برای نوشتن باز می‌کند در صورتی که پرونده وجود نداشته باشد، آن را ایجاد خواهد کرد و در صورت وجود پرونده محتوای آن را پاک می‌کند |

۱ـ مبحث فایل‌های دودویی خارج از محدودهٔ این کتاب است. برای اطلاعات بیشتر به MSDN مراجعه کنید.

| | | |
|---|---|---|
| تصادفی | Random | پرونده را برای دسترسی تصادفی باز می‌کند برای ذخیره‌سازی رکوردی مورد استفاده قرار می‌گیرد در صورتی که پرونده وجود نداشته باشد، آن را ایجاد خواهد کرد (پیش‌فرض حالت دسترسی، این گزینه است) |
| دودویی | Binary | پرونده را به صورت دودویی باز می‌کند (بیت‌ها و بایت‌ها) در صورتی که پرونده وجود نداشته باشد، آن را ایجاد خواهد کرد، (مخصوص فایل‌های دودویی است) |

- As کلید واژه‌ای که تعیین می‌کند شمارهٔ پرونده بعد از آن ارایه می‌شود.
- *FileNumber* شمارهٔ پرونده (file handle) است.
- Len کلید واژهٔ اختیاری است که قبل از پارامتر طول رکورد قرار می‌گیرد.
- *CharInBuffer*٪ یک گزینهٔ اختیاری است که طول رکورد پرونده‌ای که برای دسترسی تصادفی باز شده است را تعیین می‌کند.

**۱ـ۳ـ۲ـتخصیص شمارهٔ پرونده:** در Visual Basic این امکان وجود دارد که به‌طور همزمان چندین پرونده باز داشته باشید، مشروط بر اینکه هر پرونده شمارهٔ خاص خود را داشته باشد. به این منظور همیشه باید شمارهٔ پرونده‌های باز را بدانید و این کار بسیار مشکلی است. اما Visual Basic برای این مشکل راه حلی دارد : تابع ()FreeFile. این تابع، اولین شمارهٔ پرونده آزاد را برمی‌گرداند. با استفاده از این تابع همواره اطمینان خواهید داشت که شمارهٔ پرونده‌ها تکراری نخواهند بود. شکل استفاده از تابع ()FreeFile چنین است :

```
FreeFile [ (intRangeNumber) ]
```

پارامتر اختیاری intRangeNumber اجازه می‌دهد تا محدودهٔ شمارهٔ پرونده را (255 1 یا 511 256) مشخص کنید. محدودهٔ پیش‌فرض 255 1 است. اغلب برنامه‌نویسان ویژوال بیسیک از همین محدوده استفاده می‌کنند چون به ندرت پیش می‌آید که یک برنامه در آن واحد بیش از ۲۵۶ پرونده باز داشته باشد. در این حالت حتی نوشتن پرانتزها هم ضروری نیست. در دستورهای زیر، ابتدا یک شمارهٔ پرونده مشخص شده و سپس پرونده موردنظر با این شماره باز می‌شود :

```
intFileNumber  FreeFile()
Open "AccPay.Dat" For Output As intFileNumber
```

با استفاده از()FreeFile می‌توانید مطمئن شوید که حتی در برنامه‌های کوچک، دو پرونده با

شمارهٔ مشابه وجود نخواهند داشت. دو دستور فوق را می‌توان در یک دستور نیز خلاصه کرد :

```
Open "AccPay.Dat" For Output As FreeFile()
```

با این روش، امکان اینکه شمارهٔ پرونده باز شده را بدانید، وجود ندارد.

**۲ـ۳ـ۲ـ طول رکورد :** آرگومان CharInBuffer Len هنگام کار با پرونده‌های تصادفی اهمیت می‌یابد. هنگامی که ویژوال بیسیک از پرونده‌ای که برای دسترسی تصادفی باز شده است، اطلاعات را می‌خواند (یا در آن می‌نویسد)، باید طول رکورد (Record) را بداند. هر پرونده تصادفی در واقع مجموعه‌ای از N قسمت است که هر قسمت معادل CharInBuffer طول دارد. کار با پرونده‌های تصادفی بسیار شبیه کار با آرایه‌هاست و اگر از دستور Option Base 1 استفاده کنید بین آن‌ها یک تناظر یک به یک ایجاد خواهد شد، چون شمارهٔ رکوردها هم از ۱ شروع خواهد شد.

## ۲ـ۴ـ دستور Close

هر پرونده‌ای که باز شده باید بعد از استفاده بسته شود. دستور بستن پرونده ساده است :

```
Close# intFileNumber1[,intFileNumber2] [,... intFileNumberX]
```

تعداد پرونده‌هایی که می‌توانید در این دستور قید کنید محدودیتی ندارد و اگر هیچ شماره‌ای قید نشود تمام پرونده‌های باز، بسته خواهند شد. در کد زیر، ابتدا دو پرونده (یکی برای خواندن و دیگری برای نوشتن) باز شده و سپس بسته می‌شوند.

```
Dim intReadFile As Integer, intWriteFile As Integer
intReadFile    FreeFile     'Get first file #
Open "AccPay.Dat" For Input As intReadFile
IntWriteFile    Freefile     'Get next file #
Open "AccPayOut.Dat" For Output As intWriteFile
Close intReadFile
Close intWriteFile
```

اگر پرونده‌ای بسته نشود، احتمال صدمه به آن وجود دارد. بنابراین به محض اینکه کارتان با یک پرونده تمام شد، در اولین فرصت آن را ببندید. بهترین روش (علاوه بر بستن تک‌تک پرونده‌ها) آن است که در پایان برنامه با دستور سادهٔ زیر تمام پرونده‌های باز را به یکباره ببندید :

```
Close
```

## ۵ـ۲ـ کار با پرونده‌های ترتیبی

حال که با کلیات پرونده (مانند باز کردن، بستن) آشنا شدید، بهتر است با چند مثال خواندن ــ نوشتن پرونده‌های ترتیبی را بیشتر تمرین کنید.

پرونده ترتیبی یعنی پرونده‌ای که فقط به صورت متوالی می‌توان با آن کار کرد؛ پرونده‌های ترتیبی را باید از اول تا آخر پرونده خواند یا نوشت. خواندن/ نوشتن بزرگ‌ترین نقطه ضعف پرونده‌های ترتیبی است. برای استفاده از این قبیل پرونده‌ها باید تمام پرونده را خواند. مثلاًاگر پرونده‌ای ۱۰۰۰ بایت باشد و فقط بخواهید ۱ بایت آن را تغییر دهید، باید تمام بایت‌های دیگر را هم بخوانید و دوباره بنویسید. پرونده‌های ترتیبی برای ذخیره کردن پرونده‌های متنی کوچک که در آن‌ها سرعت چندان مهم نیست، مناسب هستند.

**۱ـ۵ـ۲ـ نوشتن پرونده‌های ترتیبی:** برای نوشتن پرونده‌های ترتیبی آن را در حالت دسترسی output یا Append باز کنید و از دستورهای Print# و Write# برای نوشتن پرونده‌های ترتیبی استفاده کنید.

***۱ـ۱ـ۵ـ۲ـ دستور Print#:*** قبل از استفاده از هر پرونده‌ای باید آن را باز کنید. بعد از باز کردن پرونده، می‌توانید اطلاعات را در آن بنویسید. یکی از روش‌های نوشتن در پرونده استفاده از دستور Print# است. با این دستور فقط در پرونده‌های ترتیبی می‌توان نوشت :

```
Print# intFileNumber, [OutputList]
```

در این دستور، intFileNumber شمارهٔ پرونده موردنظر و OutputList یکی از اقلام زیر است :

```
[Spc(intN1)  Tab (intN2)] [Expression] [charPos]
```

طرز کار تابع Spc() در دستور Print# مشابه دستور Print (در برنامه‌سازی ۱ با این دستور و تابع آشنا شده‌اید) است. در جدول ۲ـ۲ توضیحات بیشتری دربارهٔ عناصر OutputList ملاحظه می‌کنید.

(تابع Tab(intN۲) مکان‌نما را به ستون intN۲ منتقل می‌کند)

**جدول ۲ــ۲ــ عناصر لیست خروج دستور Print #**

| عنصر | مفهوم |
|---|---|
| Spc(intN1) | به مقدار intN1 بین خروجی‌ها فاصله می‌اندازد |
| Tab (intN2) | مکان ظاهر شدن خروجی را تعیین می‌کند intN2 شمارهٔ ستون رامشخص می‌کند برای نوشتن در مناطق چاپی (به فواصل ۱۴ کاراکتر) از دستور Tab بدون آرگومان استفاده کنید |
| Expression | عبارت عددی یا رشته‌ای |
| Charpos | محل ظاهر شدن دستور Print بعدی ، با ؛ دستور Print بعدی از ادامهٔ همین خط، نوشتن خروجی را ادامه خواهد داد اگر این عنصر حذف شود، دستور print بعدی از ابتدای خط بعد شروع خواهد شد |

**مثال ۱ـ۲**

**ایجاد یک ویراستار متن :** پروژه‌ای ایجاد کنید که دارای دو منوی File و Edit باشد. منوی File دارای گزینه‌های Save برای ذخیره و ایجاد پرونده متنی و Open برای بازکردن پرونده خواهد بود. برای استفاده از برنامه، هنگام ذخیرهٔ داده‌ها، روی گزینهٔ Save از منوی File کلیک کنید. در کادر محاوره‌ای که ظاهر می‌شود، نام پرونده را وارد کرده و مسیر آن را وارد کنید. سپس روی دکمهٔ Save در کادر محاوره‌ای کلیک کنید تا داده‌ها ذخیره شوند (شکل **۲ــ۲**). برای کار با کادر محاوره‌ای از کنترل Common Dialog روی فرم استفاده کنید و نام آن را CdMain قرار دهید.

برای نوشتن متنی که می‌خواهید ذخیره کنید از کنترل TextBoxای با نام textMain استفاده کنید و مشخصه MultiLine آن را مساوی True قرار دهید.


**شکل ۲ــ۲ــ استفاده از کادر محاوره‌ای، روش ساده‌ای برای ایجاد نام پرونده و محلی برای داده‌هاست.**

کد زیر، روال رویداد ()Click گزینهٔ save را نشان می‌دهد. این روال یک پرونده را با دستور Open (خط ۲۸) باز می‌کند و محتوای کادر متن را با استفاده از متد Print ذخیره می‌کند (خط ۳۴). روال رویداد، پرونده را با استفاده از دستور Close می‌بندد (خط ۴۰). دستور Close شمارهٔ file handle را به عنوان یک آرگومان می‌گیرد.

```
01 Private  Sub itmSave  Click()
02 Dim strFileName As String `String of file to open
03 Dim strText As String `Contents of file
04 Dim strFilter As String `Common Dialog filter string
05 Dim strBuffer As String `String buffer variable
06 Dim FileHandle% `Variable to hold file handle
07
08 تنظیم مشخصه filterبرای کنترل Common Dialog
```

```
09 strFilter    "Text (*.txt) *.txt  All Files (* .*) *.*"
10 cdMain.Filter    strFilter
11
12    باز کردن کادر محاوره‌ای save
13 cdMain.ShowSave
14
15
اطمینان از خالی نبودن مشخصه filename در کنترل Common Dialog
16 If cdMain.filename <>"" Then
17    درصورت خالی نبودن strfilename filename قرار می‌دهیم
18 strFileName    cdMain.filename
19
20    قراردادن محتوای txt Main در strText
21 strText    txtMain.Text
22
23
                                        به‌دست آوردن شماره پرونده برای فایل
24
25 FileHandle%    FreeFile
26
27 باز کردن فایل برای نوشتن
28 Open  strFileName For Output As # FileHandle%
29
30
تغییر اشاره‌گر ماوس به صورت ساعت شنی
31 MousePointer    vbHourglass
32
```

```
33 نوشتن در فایل
34 Print # FileHandle%, strText
35
36 برگرداندن اشاره‌گر ماوس به حالت عادی
37 MousePointer    vbDefault
38
39 بستن فایل
40 Close # FileHandle%
41 End If
42
43 End Sub
```

**نکته:** به خاطر داشته باشید که بعد از پایان کار با پرونده، آن را ببندید (دستور Close). این دستور شمارهٔ File handle را از حافظه خارج می‌کند.

**مثال ۲-۲**

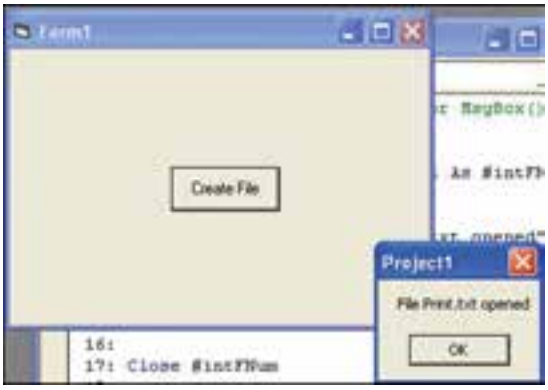
در کد زیر، روالی را مشاهده می‌کنید که پرونده Print.txt را باز کرده، اعداد ۱ تا ۶ را در آن پرونده می‌نویسد و سپس پرونده را می‌بندد.

```
Private Sub cmdFile_Click()
    Dim intCtr As Integer    'Loop counter
    Dim intFNum As Integer    'File number
    intFNum    FreeFile
    Open "C: \Print.txt" For Output As #intFNum
    MsgBox "File Print.txt opened"
    For intCtr    1 To 6
        Print #intFNum, intCtr    'Write the loop counter
        MsgBox "Writing a "& cstr(intCtr) & "to Print.txt"
```

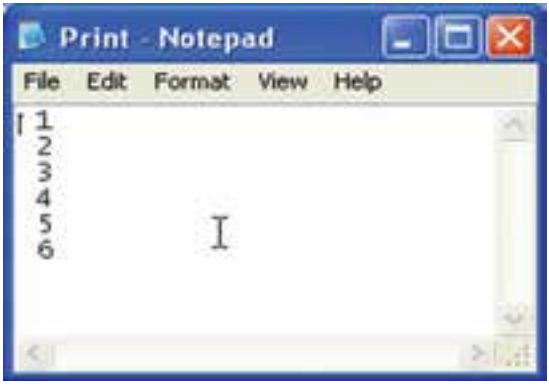
```
    Next intCtr
    Close #intFNum
    MsgBox "File Print.txt closed"
End Sub
```

برنامه را اجرا کنید. این برنامه در هر مرحله یک کادر پیام نمایش خواهد داد : هنگام باز شدن پرونده، هنگام نوشتن اعداد در پرونده و هنگام بسته شدن پرونده. برای اینکه مطمئن شوید این روال، کار خود را به درستی انجام داده است، در برنامهٔ NotePad ویندوز پرونده Print.txt را باز کنید. اعداد ۱ تا ۶ را باید در این پرونده مشاهده کنید :

1
2
3
4
5
6




شکل ۳ــ۲




شکل ۴ــ۲

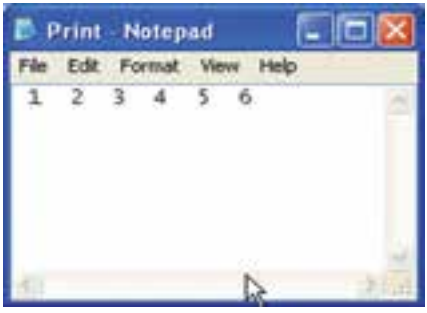
**مثال۳ـ۲**

به کد زیر توجه کنید. در این برنامه بعد از دستور #Print از ; استفاده شده است، بنابراین اعداد پشت سر هم در پرونده نوشته خواهند شد.

```
Private Sub cmdFile  Click()
   Dim intCtr As Integer   'Loop counter
   Dim intFNum As Integer   'File number
   intFNum    FreeFile
   Open "C: \Print.txt" For Output As #intFNum
   MsgBox "File Print.txt opened"
   For intCtr    1 To 6
         Print #intFNum, intCtr;      به علامت ; توجه کنید
         MsgBox "Writing a "& cstr(intCtr) & "to Print.txt"
   Next intCtr
   Close #intFNum
   MsgBox "File Print.txt closed"
End Sub
```

خروجی این پرونده چنین خواهد بود :

1 2 3 4 5 6




شکل۵ـ۲

فاصله‌ای که بین اعداد ملاحظه می‌کنید علامت مثبت اعداد مزبوراست که به قرینه حذف شده است. دستور # Print علامت اعداد را هم در پرونده چاپ خواهد کرد.

با توجه به مثال ۲ـ۲ و ۳ـ۲ مشاهده می‌شود که اگر در انتهای دستور # Print از علامت ; (سمیکلون) استفاده کنیم عملیات نوشتن در همان خط جاری ادامه می‌یابد و در غیر این صورت مکان‌نما به خط بعد منتقل می‌شود.

**۲-۱-۵-۲- دستور #Write:** دستور #Write فرمان دیگری است برای نوشتن در پرونده‌های ترتیبی. دستورات #Write و #Print تفاوت کمی با هم دارند : داده‌هایی که با #Write در پرونده نوشته می‌شوند با کاما (,) از هم جدا خواهند شد. این دستور هنگام نوشتن **رشته آن را درون زوج گیومه (") قرار خواهد داد و برای نوشتن تاریخ از # استفاده می‌کند. مقادیر Boolean را به صورت #TRUE# و #FALSE# می‌نویسد. داده‌های null و خطا را هم به صورت #NULL# و #Error *errorcode*# خواهد نوشت**. *errorcode* کد تولید شده به وسیلهٔ خطای رخ داده است.

## ۲-۵-۲- خواندن پرونده‌های ترتیبی:

بازیابی داده‌ها در دیسک خیلی شبیه نوشتن داده ها در آن است با این تفاوت که باید پرونده‌ها را در حالت دسترسی Input باز کنید و از دستورات #Input یا line Input برای خواندن پرونده استفاده کنید.

**۲-۱-۵-۲- دستور #Input:** بعد از نوشتن اطلاعات در پرونده، باید بتوانید آن‌ها را دوباره بازیابی کنید. دستور خواندن پرونده‌های ترتیبی، دستور #Input است. خواندن یک پرونده ترتیبی باید دقیقاً به همان ترتیب نوشتن آن صورت گیرد :

```
Input #intFileNumber, Variable1 [, Variable2] [,... VariableN]
```

دستور #Input هم به یک شمارهٔ پرونده نیاز دارد. این دستور باید دقیقاً متناظر با دستور #Print همان پرونده باشد. دستور زیر پنج مقدار را از پرونده باز شده می‌خواند و آن‌ها را در متغیرهای V1 تا V5 قرار می‌دهد :

```
Input #intFileNumber, V1, V2, V3, V4, V5
```

نوع دادهٔ این پنج متغیر باید با دستور #Print که این مقادیر را در پرونده نوشته است، مطابقت داشته باشد. در غیر این صورت، دستور #Input قادر به خواندن داده‌ها نخواهد بود.

از آن‌جایی که داده‌ها با #Write به وسیلهٔ کاما از هم جدا خواهند شد، انطباق #Input و #Write ضروری نیست. بنابراین برای سهولت کار، سعی کنید از #Write به جای #Print استفاده کنید. شکل کلی دستور #Write چنین است :

```
Write #intFileNumber , [OutputList]
```

که در آن OutputList فهرست متغیرهایی است که باید در پرونده intFileNumber نوشته شوند.

مشکلاتی که در انطباق دستورات #Print و #Input وجود دارد، ما را به یک دستور کلی‌تر و قابل انعطاف‌تر می‌رساند : دستور Write.

نوشتن و خواندن پرونده‌ها به یکدیگر وابسته‌اند و هر دو عمل می‌توانند در یک برنامه انجام شوند.

در کد زیر، برنامه‌ای را مشاهده می‌کنید که دو روال جداگانه عملیات نوشتن و خواندن پرونده را نشان داده است. در این برنامه از روال رویداد Click دکمهٔ CmdFileout برای عملیات نوشتن و از روال رویداد Click دکمهٔ CmdFileIn برای عملیات خواندن پرونده استفاده شده است.

```
1:  Private Sub cmdFileOut  Click()
2:          'Create the sequential file
3:          Dim intCtr As Integr        'Loop counter
4:          Dim intFNum As Integer   'File number
5:          intFNum    FreeFile      `به‌دست آوردن شماره پرونده
6:                  باز کردن پرونده Print.txt برای نوشتن
7:          Open "Print.txt" For Output As #intFNum
8:          `نوشتن اعداد ۱ تا ۶ در پرونده
9:          For intCtr    1 To 6
10:         Print # intFNum, intCtr;    'Write the loop counter
11: Next intCtr
12:         `بستن پرونده
13: Close # intFNum
14: End Sub
15:
16: Private Sub cmdFileIn  Click()
17: 'Read the sequential file
18: Dim intCtr As Integer       'Loop counter
19: Dim intVal As Integer       'Read value
20: Dim intFNum As Integer   'File number
21: Dim intMsg As Integer     'MsgBox()
22: intFNum    FreeFile    `به‌دست آوردن شماره پرونده
23: Open "Print.txt" For Input As # intFNumباز کردن پرونده به‌منظور خواندن آن
24:
```

```
25: For intCtr    1 To 6      خواندن ۶ مقدار از نوع Integer از پرونده
26:         Input # intFNum, intVal
27:         'Display the results in the lmmediate windows
28:         intMsg    MsgBox("Retrieved a "& intVal &"from Print.txt")
29: Next intCtr
30:
31: Close # intFNum         بستن پرونده
32: intMsg    MsBox("The Print.txt file is now closed.")
33: End Sub
```

اکنون کد زیر را درنظر بگیرید. در این برنامه برای نوشتن در پرونده از دستور #Write استفاده شده است.

```
1: Private cmdFile  Click()
2:         Dim intCtr As Integer       'Loop counter
3:         Dim intFNum As Integer    'File number
4:         intFNum    FreeFile
5:
6:         Open "c: \Write.txt" For Output As #intFNum
7:
8:         For intCtr    1 To 6
9:                 Write # intFNum, intCtr;     'Write the loop counter
10: Next intCtr
11:
12: Close # intFNum
13: End Sub
```

برنامه را اجرا کرده و سپس پرونده ایجاد شده (Write.txt) را باز کنید. به تفاوت خروجی دستورات #Write و #Print دقت کنید :

```
1,2,3,4,5,6
```

**نکته :** اگر در پایان دستور #Write از; استفاده نکنیم، هر عدد در یک خط و بدون کاما نوشته خواهد شد و دیگر از کاما هم خبری نخواهد بود. چون در این حالت دیگر وجود کاما برای جدا کردن داده‌ها ضرورتی ندارد. (در این حالت دستورات #Write و #Print شبیه یکدیگر خواهند شد.)

**۲ـ۲ـ۵ـ۲ـ بازیابی داده‌ها با دستور LineInput :** با استفاده از دستور Line Input می‌توانید داده‌های پرونده را خط به خط بخوانید. شکل کلی دستور Line Input به صورت زیر است :

```
Line Input #FileHandle, strBuffer
```

در این دستور :

- Line Input کلید واژه‌های دستور هستند.
- FileHandle شمارهٔ پرونده باز است.
- StrBuffer رشته‌ای است که دستور، داده‌های بازیابی شده را در آن قرار می‌دهد.

پرونده‌های متنی ساده به صورت خط به خط در دیسک ذخیره می‌شوند. اگر متنی را در NotePad وارد کرده و کلید Enter را فشار ندهید، یک خط وارد کرده‌اید. هر بار که کلید Enter را فشار دهید، ویژوال بیسیک رشتهٔ chr(13)&chr(10) را به کادر متن اضافه می‌کند تا پایان خط را مشخص کند. هنگامی که این خط را ذخیره می‌کنید، این نویسه‌ها نیز در پرونده نوشته می‌شوند. ویژوال بیسیک دارای ثابت vbCrLF برای این رشته است. دستور Line Input نویسه‌های داخل پرونده را تا زمانی که به vbCrLF برسد می‌خواند. در پایان خط، دستور نویسه‌ها را به آرگومان بافر (strBuffer) ارسال می‌کند و از vbCrLF صرف نظر می‌کند.

برای پیمایش تمام خطوط پرونده، از حلقهٔ Do While استفاده کنید. از تابع EOF() ویژوال بیسیک برای تعیین اینکه آیا به انتهای پرونده رسیده‌ایم، استفاده کنید. این تابع، شمارهٔ پرونده را به عنوان آرگومان دریافت می‌کند. تا زمانی که به انتهای پرونده نرسیده‌اید، دستور Line Input به خواندن خطوط پرونده در داخل حلقهٔ Do While ادامه می‌دهد.

**مثال ۲ـ۴**

کد زیر، روال رویداد مربوط به گزینهٔ Open از منوی File پروژهٔ مثال ۱ـ۲ را نشان می‌دهد. کاربر برای بازکردن پرونده در داخل ویراستار متن، روی این گزینه کلیک می‌کند. روال

رویداد از یک کادر محاوره‌ای برای فراهم کردن امکان شناسایی پرونده‌ای که باید باز شود، استفاده می‌کند. (کنترل CommonDialog با نام CdMain)

```
01 Private  Sub itmOpen  Click()
02 Dim strFileName As String `String of file to open
03 Dim strText As String `Contents of file
04 Dim strFilter As String `Common Dialog filter string
05 Dim strBuffer As String `String buffer variable
06 Dim FileHandle% `Variable to hold file handle
07
08 `Set the Common Dialog filter
09 strFilter   "Text (*.txt) *.txt  All Files (*.*) *.*"
10 cdMain.Filter   strFilter
11
12 `Open the common dialog
13 cdMain.ShowOpen
14
15 `Make sure the retrieved filename is not a blank string
16 If cdMain.filename <>"" Then
```

اطمینان از خالی نبودن مشخصهٔ filename در کادر محاوره‌ای Open

```
17
18 `If it is not blank, open the file
19 strFileName   cdMain.filename
20
21 `Get a free file handle and assign it to the file
22 `handle variable
23 FileHandle%   FreeFile
```

به‌دست آوردن شمارهٔ فایل آزاد

```
24
25 `Open the file
```

```
26 Open strFileName For Input As #FileHandle%
```
باز کردن فایل برای خواندن به حالت دسترسی INput توجه کنید
```
27
28 `Make the mouse pointer an hourglass
```
تغییر اشاره‌گر ماوس به ساعت شنی
```
29 MousePointer    vbHourglass
30
31 `Traverse the lines of the file
```
خواندن فایل به‌صورت خط به خط تا رسیدن به انتهای فایل
```
32 Do While Not EOF(FileHandle%) `Check for end of file
33
34 `Read a line of the file
35 Line Input #FileHandle%, strBuffer . strBuffer
```
خواندن یک خط فایل و ذخیرهٔ آن در
```
36
37 `Add the line from the Output buffer
38 strText    strText & strBuffer & vbCrLf
```
اضافه کردن خط خوانده شده از فایل به متن موجود در رشتهٔ strText
```
39 Loop
40
41 `Change the mousepointer back to the arrow
```
تغییر اشاره‌گر ماوس به حالت عادی
```
42 MousePointer    vbDefault
43
44 `Close the file when completed
45 Close #FileHandle%
46
47 `Assign the retrieved text to the text box
48 txtMain.Text    strText
```
قرار دادن محتوای فایل (که اکنون در StrText است) در کنترل textbox

```
49
50 `Put the filename in the form caption
51 frmMain.Caption    "Text Editor  ["& strFileName &"]"
```
قرار دادن نام فایل در عنوان فرم
```
52 End If
53 End Sub
```

## ۶ـ۲ـ کار با پرونده‌های تصادفی

با آن که دربارهٔ پرونده‌های ترتیبی بسیار صحبت کردیم، ولی باید بگوییم که برنامه‌نویسان Visual Basic به ندرت از این نوع پرونده‌ها استفاده می‌کنند و بیشتر تمایل دارند با پرونده‌های تصادفی (random access) کار کنند. پرونده تصادفی، پرونده‌ای است که در هر نقطه از آن (بدون رعایت ترتیب) می‌توان نوشت یا از آن خواند. دلیل اولیهٔ اهمیت پرونده‌های تصادفی، معرفی نوع جدیدی از انواع داده است : نوع دادهٔ تعریف شده به وسیلهٔ کاربر (user defined data type). این نوع داده، همانطور که از نام آن پیداست، برخلاف انواع دادهٔ ذاتی Visual Basic به وسیلهٔ کاربر تعریف می‌شود.

**۱ـ۶ـ۲ـ نوع دادهٔ تعریف شده به‌وسیلهٔ کاربر:** در مطالبی که تاکنون بیان کرده‌ایم از انواع داده‌های ذاتی ویژوال بیسیک مثل String، Integer و Double استفاده کرده‌ایم. یکی از ویژگی‌های ویژوال بیسیک، فراهم نمودن امکان ایجاد نوع دادهٔ سفارشی است که به نام نوع داده تعریف شده به وسیلهٔ کاربر (user defined Data Type یا به اختصار UDT) معروف است. می‌توان UDT را به عنوان متغیری فرض کرد که از چندین بخش ایجاد شده است و هر بخش را می‌توان چندین بار در برنامه به کار برد.

برنامه‌ای را در نظر بگیرید که نام قطعات موسیقی و آهنگساز هر قطعه را ذخیره می‌کند. اگر بخواهید برای هر قطعه و آهنگساز، متغیر خاصی را تعریف کنید، بخش اعلان متغیرها به صورت زیر خواهد بود :

```
01 Public g  ComposerOne As String        اعلان متغیر برای آهنگساز قطعه ۱
02 Public g  PieceOne As String           اعلان متغیر برای نام قطعه ۱
03
04 Public g  Composer Two As String       اعلان متغیر برای آهنگساز ۲
```

اعلان متغیر برای نام قطعه ۲ **05** Public g PieceTwo As String

**06**

اعلان متغیر برای آهنگساز قطعه ۳ **07** Public g ComposerThree As String

اعلان متغیر برای نام قطعه ۳ **08** Public g PieceThree As String

این قبیل کدنویسی ممکن است برای برنامه کوتاه مناسب باشد ولی اگر تعداد قطعات موسیقی که می‌خواهیم ذخیره کنیم، بیشتر شود، این روش، مناسب نخواهد بود. یک راه‌حل ساده برای این مسأله، تعریف یک بستهٔ متغیری و نامگذاری آن است. به بیان دیگر با ترکیب داده‌های ذاتی Visual Basic می‌توانید انواع داده‌های جدیدی ایجاد کنید. این بسته را نوع دادهٔ تعریف شده به‌وسیلهٔ کاربر (UDT) می‌نامند. که به آن گاهی ساختار (structure) یا رکورد (Record) نیز گفته می‌شود. متغیرهای UDT را نیز با استفاده از کلید واژه‌های Dim، Public یا Private می‌توان اعلان کرد.

برای تعریف نوع دادهٔ کاربر، از دستور Type استفاده می‌شود. هر نوع دادهٔ کاربر باید دارای یک نام باشد (*TypeName*). این نام باید در کل برنامه منحصر به فرد باشد. تمام انواع داده کاربر باید در سطح مدول تعریف شوند و تعریف آن‌ها در داخل روال‌ها مجاز نیست. اگر یک نوع داده کاربر در مدول فرم تعریف شود، حتماً باید به صورت Private تعریف شده باشد.

یک UDT را می‌توان با استفاده از کلید واژهٔ Type در بخش General مدول ایجاد کرد :

```
Type TypeName
        Elements as DataType
        ……
End type
```

در این دستور :

- Type : کلید واژهٔ ویژوال بیسیک است که شروع یک بلاک نوع داده را مشخص می‌کند.
- *TypeName* : نام نوع داده است.
- Elements as DataType : هر عضوی از نوع داده است.
- End type : پایان دستور Type را مشخص می‌کند.

برای مثال آهنگساز و قطعه موسیقی، می‌توان نوع داده‌ای به نام Music به‌صورت زیر تعریف کرد.

```
Type Music
        Composer As String
```

```
    Piece As String
End Type
```

پس از تعریف نوع داده توسط کاربر ویژوال بیسیک این نوع داده را مانند انواع داده‌های ذاتی خود (Integer، String ،...) می‌شناسد. حال برای استفاده از این نوع داده مانند استفاده از انواع داده‌های ذاتی در ویژوال بیسیک عمل کنید لذا باید متغیری از این نوع را اعلان کنید.

```
Dim MyMusic As Music
```

متغیر MyMusic از نوع داده Music اعلان شده است.

شکل کلی رجوع به عناصر نوع دادهٔ تعریف شده به وسیلهٔ کاربر به صورت زیر است :

```
VarName.ElementName
```

که در این شکل کلی :

- VarName : نام متغیر است (نمونه‌ای از نوع داده).
- ElementName : نام عنصر خاصی از نوع داده است.

در مثال بالا از MyMusic.Composer و Mymusic.Piece به ترتیب برای دسترسی به نام آهنگساز و نام قطعه موسیقی استفاده می‌شود.

کد زیر، چگونگی ایجاد نوع دادهٔ Music که از اجزای قطعهٔ موسیقی و آهنگساز تشکیل شده است را نشان می‌دهد. همچنین در این کد، متغیری به نام MyMusic از نوع داده تعریف شده است. با استفاده از دو کنترل textbox با نام‌های txtcomposer و txtpiece نام آهنگساز و نام قطعه از کاربر دریافت می‌شود.

```
01 `Make a user  defined type that has
02 `elements for the composer and the piece
03 Type Music
04 Composer As String
05 Piece As  String
06 End Type

01 Dim MyMusic As Music
02 Dim Msg $
```

```
03 `Assign values to each element in the user
04 `defined type. Values come from TextBoxes on a form.
05 MyMusic. Composer    txtComposer. Text
06 MyMusic.Piece    txtPiece. Text
07
```

در کادر پیام نام آهنگساز و نام قطعه در دو خط مجزا به شکل زیر

نام آهنگساز : Composer
نام قطعه : **Piece**

نمایش داده می‌شود. برای انتقال مکان نما به خط جدید از ثابت Vberlf استفاده شده است.

```
08 `Create a string that displays all of the
09 `values in the type
10 Msg$   "Composer: "&MyMusic.Composer & vbCrLf
11 Msg$    Msg$ & "Piece: "& MyMusic. Piece
12
13 `Display the string
14 MsgBox Msg$
```

فرض کنید برنامه‌ای دارید که نیاز به یک دفترچه آدرس دارد. هر آدرس تشکیل می‌شود از نام، نام خانوادگی، آدرس، شماره تلفن و اطلاعاتی از این قبیل. برای کار با این اطلاعات می‌توانید از متغیرهای جداگانه استفاده کنید ولی در این صورت برنامه‌نویسی بسیار خسته کننده و پیچیده خواهد شد. بسیار ساده‌تر خواهد بود تا بتوانیم با ترکیب این متغیرها نوع دادهٔ جدیدی ایجاد کنیم و از آن به بعد با این نوع داده کار کنیم.

از طریق کد زیر، می‌توانید مطالب بیشتری دربارهٔ دستور Type یاد بگیرید.

```
1: 'Module Page of the Project
2: Type UserType
3: strFName As String
4: strLName As String
5: End Type
```

```
6:  Public Names As UserType
```

در این کد یک نوع داده کاربر به نام UserType تعریف شده است. این نوع دادهٔ جدید دارای دو متغیر از نوع رشته‌ای (strFName و strLName) است. در خط ۶ هم یک متغیر از این نوع جدید تعریف شده است. دقت کنید که UserType یک متغیر نیست بلکه یک نوع داده است و این Names است که متغیری از نوع UserType می‌باشد.

برای دسترسی به اجزای درونی نوع داده کاربر باید از عملگر نقطه (.) استفاده کنیم. به کد زیر توجه کنید :

```
Names.strFName    "Mohammad"
Names.strLNames   "Yamaghani"
lblFName.Caption  "First Name: "&Names.strFName
lblLName.Caption  "Last Name: "&Names.strLName
```

به هر عضو درون نوع دادهٔ کاربر یک فیلد (field) گفته می‌شود. متغیرهای رشته‌ای درون نوع دادهٔ کاربر را می‌توان با گزینهٔ *StringLenght بعد از As String محدود کرد. بدین ترتیب طول رشته با مقدار ثابت StringLenght مقداردهی خواهد شد. هنگام نوشتن نوع دادهٔ کاربر در یک فایل (به دلیل این‌که اندازهٔ هر رکورد باید مشخص باشد) فیلدهای رشته‌ای باید اندازهٔ ثابت داشته باشند. در کد زیر، شکل تغییریافته کد قبل را مشاهده می‌کنید.

```
1:  'Module Page of the Project
2:  Type UserType2
3:          strFName As String * 8
4:          strLName As String * 20
5:  End Type
6:  Public Names As UserType2
```

یک رشتهٔ ثابت نمی‌تواند بیش از حداکثر تعیین شده مقدار بگیرد و اگر مقداری که به آن می‌دهیم کمتر از حداکثر طول تعریف شده باشد، Visual Basic بقیهٔ رشته را با فضای خالی پر خواهد کرد.

**۲-۶-۲- باز کردن پرونده‌های تصادفی :** قبلاً دیدید که در حالت پیش‌فرض، دستور Open پرونده‌ها را در حالت تصادفی باز می‌کند :

```
Open "Random.txt" As#1
```

که معادل دستور زیر است :

```
Open "random.txt" For random As#1
```

(البته می‌توان یک پرونده را در حالت تصادفی باز کرد ولی با آن به صورت ترتیبی کار کرد.) برای درک بهتر تفاوت پرونده‌های ترتیبی و تصادفی به یک مثال توجه کنید. فرض کنید پرونده‌ای دارید که در ۱۰ خط آن مقدار کل کالاهای یک انبار را نوشته‌اید. حال اگر این پرونده را در حالت ترتیبی باز کرده باشید و بخواهید خط ششم (رکورد ششم) را بخوانید باید پنج رکورد قبل را هم بخوانید تا بتوانید به رکورد ششم دسترسی پیدا کنید. ولی در حالت تصادفی، می‌توانید به‌طور مستقیم به سراغ رکورد ششم رفته و آن را بخوانید (در نوشتن پرونده هم همین مطلب صادق است). وقتی پرونده‌ای دارای ۱۰ رکورد باشد، تفاوت چندان محسوس نخواهد بود ولی می‌توانید تصور کنید که برای پرونده‌ای با ۱۰۰۰۰ رکورد چه تفاوتی بین این دو روش وجود خواهد داشت.

**۳ـ۶ـ۲ـ دستورهای Get و Put :** برای خواندن و نوشتن پرونده‌های تصادفی از دو دستور Get# و Put# استفاده می‌کنیم. این دو دستور معادل دستورات Input# و Print# در پرونده ترتیبی هستند. ولی تفاوت کوچکی بین این دستورها وجود دارد. در دستورهای Input# و Print#راهی وجود ندارد تا نقطه‌ای از پرونده برای خواندن یا نوشتن مشخص شود، در حالی که دستورهای Get# و Put# دارای چنین امکانی هستند :

```
Put [#]intFileNum, [intRecNum,] Variable
Get [#]intFileNum, [intRecNum,] Variable
```

- intFileNum شمارهٔ پرونده موردنظر است.
- intRecNum شمارهٔ رکوردی است که می‌خواهید با آن کار کنید.

> **نکته :** درصورتی که شمارهٔ رکورد در این دستورها ذکر نشود، عملیات روی رکورد بعد از رکورد جاری انجام می‌شود.

- Variable متغیری است که مقدار آن در پرونده نوشته می‌شود و یا مقداری که از پرونده خوانده شده ، در آن ذخیره می‌شود.

همان‌طور که مشاهده می‌کنید در این دستورها از آرگومانی به نام شمارهٔ رکورد استفاده می‌شود. با تعیین شمارهٔ رکورد، می‌توانید فقط آن رکورد را خوانده یا در آن بنویسید. شمارهٔ رکوردها از ۱

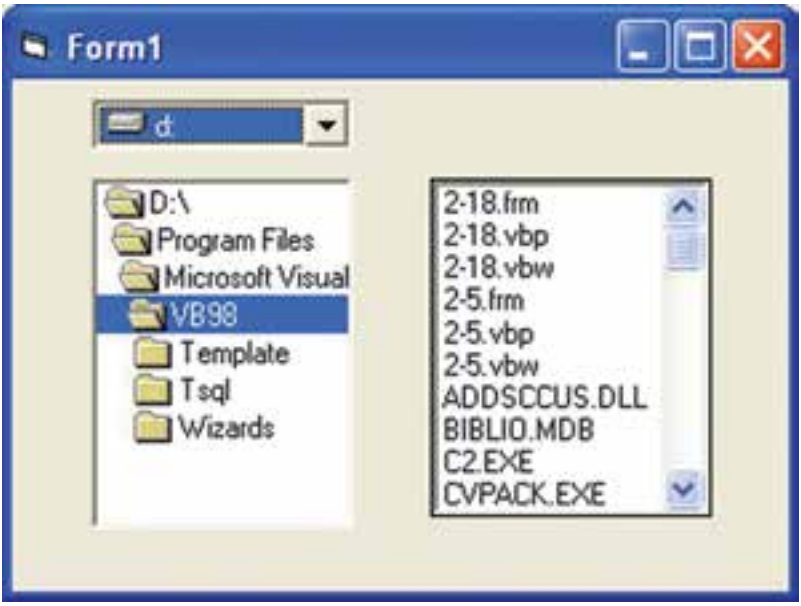
شروع می‌شود. این دستورها می‌توانند هر نوع داده‌ای (حتی آرایه یا نوع دادهٔ کاربر) را بخوانند و این قوی‌ترین ویژگی پرونده‌های تصادفی است. در مطالب بعد طی چند مثال با روش کار این‌ها بیشتر آشنا خواهید شد.

## ۶ـ۲ـ۷ـ کنترل‌های FileListBox، DirectoryListBox و DriveListBox

ویژوال بیسیک دارای سه کنترل خاص برای مدیریت پوشه‌ها، درایوها و پرونده‌هاست :

- کادر لیست دایرکتوری ــ به کاربر امکان باز کردن پوشه (دایرکتوری)ها را می‌دهد.
- کادر لیست درایو ــ به کاربر امکان انتخاب یک درایو دیسک را می‌دهد.
- کادر لیست پرونده ــ به کاربر امکان انتخاب پرونده را می‌دهد.

شکل ۶ــ۲ فرمی را با سه کنترل مزبور نشان می‌دهد.

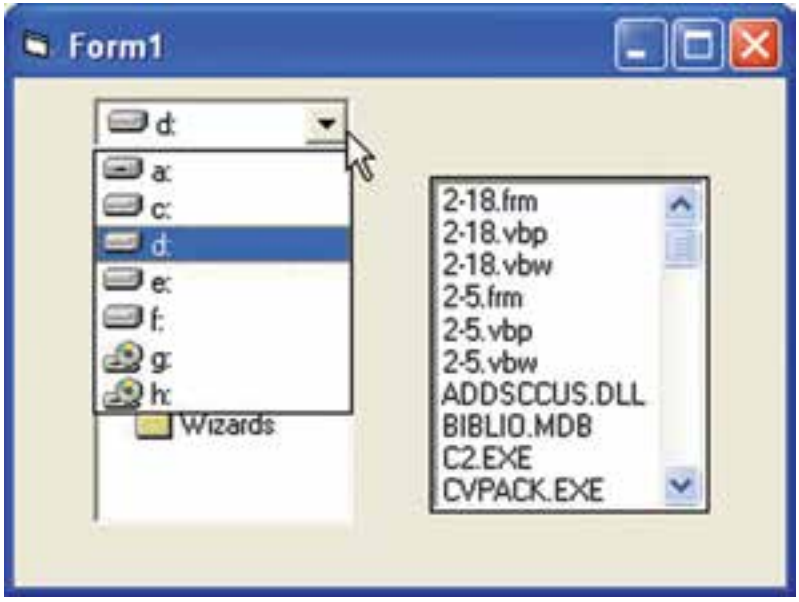



شکل ۶ــ۲ــ کنترل های پرونده Visual Basic

شاید تعجب کنید که با وجود کنترل‌های کادر محاوره‌ای، دیگر چه نیازی به این سه کنترل داریم. مواقعی پیش می‌آید که فقط به یک یا دو جنبه از این کنترل‌ها نیاز داریم. به عنوان مثال، نوشتن پرونده‌هایی که پوشه یا درایو آن از قبل مشخص شده‌است (در شبکه‌ها این وضعیت اغلب اتفاق می‌افتد). کنترل‌های

پرونده در Visual Basic با هم ارتباط ساختاری ندارند، یعنی اگر در کادر لیست درایو، یک درایو را انتخاب کنید، دو کنترل دیگر به طور خودکار متوجه این تغییر نخواهند شد. این وظیفهٔ برنامه‌نویس است که با نوشتن کد بین آن‌ها ارتباط برقرار کند.

**۱ـ۷ـ۲ـ کادر لیست درایو :** کادر لیست درایو (DriveListBox) به کاربران امکان انتخاب یک درایو را می‌دهد. این کنترل می‌تواند تمام درایوهای موجود در سیستم (دیسک‌های سخت محلی و شبکه، درایوهای فلاپی و CD ROM) را شناسایی کند. شکل ۷ـ۲ یک کادر لیست درایو باز را نشان می‌دهد.




شکل ۷ـ۲ـ کادر لیست درایو

درایو پیش‌فرض در کادر لیست درایو، درایوی است که برنامه از آن‌جا اجرا شده است، اما با نوشتن کد می‌توان این حالت را تغییر داد.

مشخصهٔ متداول این کنترل، Drive است که نام درایو انتخاب شده در آن قرار می‌گیرد. به کمک این مشخصه می‌توان نام درایو پیش‌فرض را نیز تغییر داد.

دستور زیر درایو پیش فرض کنترل drvDisk را به درایو C تغییر می‌دهد.

```
drvDisk.drive  "C:\".
```

درصورتی که کاربر درایو دیگری را از این کنترل انتخاب کند رویداد Change رخ می‌دهد.

برای تغییر درایو از دستور ChDrive استفاده کنید.

```
Private sub drvDisk  Change ( )
ChDrive drVDisk.drive
End sub
```

درایو انتخاب شده توسط کاربر در مشخصهٔ درایو کنترل قرار دارد (drvDisk.drive) شکل کلی دستور chDrive در جدول ۴ــ۲ آمده است.

**۲ـ۷ـ۲ـ کادر لیست دایرکتوری :** با کادر لیست دایرکتوری (DirectoryListBox)، کاربر امکان انتخاب پوشهٔ موردنظر را خواهد داشت. این کنترل تمام پوشه‌های موجود در سیستم را شناسایی می‌کند. کادر لیست دایرکتوری برای نمایش پوشه‌ها از استانداردهای ویندوز استفاده می‌کند. به یاد داشته باشید که این کنترل نمی‌تواند به طور خودکار درایو انتخاب شده را تشخیص دهد. در این فصل، مشاهده خواهید کرد که چگونه می‌توان بین این کنترل‌ها ارتباط برقرار کرد.

دایرکتوری پیش‌فرض کادر لیست دایرکتوری، پوشه‌ای است که برنامه از آن‌جا اجرا شده است، ولی این وضعیت را می‌توان تغییر داد.

مشخصهٔ متداول این کنترل، Path است که نام درایو انتخاب شده در آن قرار می‌گیرد. به کمک این مشخصه می‌توان نام پوشهٔ پیش‌فرض را نیز تغییر داد.

دستور زیر مسیر پیش فرض کنترل dirDirect را مساوی "C: \MyFiles" قرار می‌دهد.

```
dirDirect.path    "C: \MyFiles"
```

درصورتی‌که کاربر مسیر دیگری را از کنترل انتخاب کند رویداد Change رخ میدهد. مسیر انتخاب شده توسط کاربرد در مشخصه path قرار دارد (dirDirect.path). برای تغییر مسیر از دستور ChDir استفاده کنید (شکل کلی این دستور در جدول ۴ــ۲ آمده است).

```
private sub dirDirect  Change ()
   ChDir dirDirect.path
End sub
```

**۳ـ۷ـ۲ـ کادر لیست پرونده :** به کمک کادر لیست پرونده (FileListBox) کاربر می‌تواند پرونده موردنظرش را انتخاب کند. این کنترل قادر است تمام فایل‌های موجود در سیستم را شناسایی کند. این کنترل برای نمایش پرونده‌ها از استاندارد گرافیکی ویندوز استفاده می‌کند. به یاد داشته باشید که این کنترل به خودی‌خود قادر به تشخیص درایو و پوشهٔ انتخاب شده نیست. کادر لیست پرونده

به‌طور پیش‌فرض پرونده‌های موجود درپوشه‌ای که برنامه از آن‌جا اجرا شده را نمایش خواهد داد. این وضعیت با نوشتن کد مناسب قابل تغییر است.

مشخصه‌های متداول این کنترل در جدول ۳ـ۲ ارایه شده‌اند.

**جدول ۳ـ۲**

| مشخصه | توضیح |
|---|---|
| FileName | نام پرونده انتخاب شده |
| Path | مسیر پرونده انتخاب شده |
| Pattern | تعیین نوع پرونده‌های قابل مشاهده در این کنترل |
| Multi Select | امکان انتخاب چندین پرونده به صورت همزمان |

هنگامی که کاربر پرونده‌ای را انتخاب کند رویداد Change رخ می‌دهد. نام پرونده در مشخصه Filename و مسیر آن در مشخصهٔ path قرار دارد. کد زیر نام فایل انتخابی را عنوان فرم قرار می‌دهد.

```
Private sub filFiles  Change ()
form1.Caption     filFiles. File Name
End sub
```

(نام فرم form1 و نام کنترل لیست پرونده filFiles است.)

## ۸ـ۲ـ دستورهای پرونده

ویژوال بیسیک دارای چندین دستور برای کار با پرونده‌ها، پوشه‌ها و درایوها است. این دستورات را در جدول ۴ـ۲ ملاحظه می‌کنید.

فرض کنید می‌خواهید در شروع برنامه کنترل‌های لیست درایو و دایرکتوری به مسیر C: \MyFiles اشاره کنند. برای این کار باید از کد زیر در روال Form  Load() استفاده کنید :

```
ChDrive"C: "
ChDir"\MyFiles"
```

علاوه بر دستورهای جدول ۴ــ۲، Visual Basic از توابع ()Dir (برای تعیین وجود یاعدم وجود پرونده‌ها) و ()CurDir (برای تعیین نام دایرکتوری جاری) نیز پشتیبانی می‌کند.

تابع ()Dir به توضیح بیشتری نیاز دارد. فرض کنید می خواهید بدانید آیا پرونده ای به نام SALES98.DAT در درایو C وجود دارد یا خیر؟ برای این کار می توانید از کد زیر استفاده کنید :

**جدول ۴ــ۲ــ دستورات پرونده Visual Basic**

| دستور | مفهوم |
|---|---|
| ChDrive strDrive | به درایو strDrive تغییر درایو می‌دهد |
| ChDir strDirectory | به دایرکتوری Strdirectory تغییر دایرکتوری می‌دهد اگر درایو مشخص نشده باشد، Visual Basic از درایو جاری استفاده خواهد کرد |
| Kill strFileSpec | پرونده (یا پرونده‌های) تعیین شده را پاک می‌کند |
| MkDir strDirectory | دایرکتوری strDirectory را می‌سازد |
| RmDir strDirectory | دایرکتوری strDirctory را حذف می‌کند اگر در دایرکتوری هنوز پرونده‌ای وجود داشته باشد، این دستور تولید خطا خواهد کرد |

```
If (Dir("C: \SALES98.DAT"))    "SALES98.DAT" Then
        IntMsg    MsgBox ("The file exist")
Else
        IntMsg    MsgBox ("The file does not exist")
End If
```

اگر پرونده خواسته شده موجود باشد، این تابع نام آن را برمی‌گرداند و درصورتی که پرونده موجود نباشد، تابع ()Dir چیزی برنمی‌گرداند. آرگومان تابع ()Dir (مانند اکثر توابع پرونده) می‌تواند از کاراکترهای عمومی (Wildcard)[1] استفاده کند :

```
Dir("C: \Sales*.DAT")
```

---

۱ــ در کتاب سیستم عامل با این نویسه‌ها آشنا شده‌اید.

این دستور اولین پرونده‌ای را که با آرگومان داده شده مطابقت داشته باشد، برمی‌گرداند. بعد از اجرای اولین دستور Dir می‌توانید از آن به بعد تابع ()Dir بدون آرگومان (یا حتی بدون پرانتز) استفاده کنید. در این حالت Visual Basic پرونده‌های بعدی که با معیار بالا مطابقت داشته باشند را تا زمانی که رشتهٔ null ("") برگشت داده شود، برخواهد گرداند.

اگر بخواهید درایو پیش‌فرض کادر لیست درایو را تغییر دهید باید از خاصیت Drive استفاده کنید :

```
drvDisk.Drive   "d:\"
```

با این کار، هنگامی که این کنترل فعال شود، درایو D : در بالای آن مشاهده خواهد شد. هنگامی که در این کنترل، کاربر درایو دیگری را انتخاب کند، رویداد ()Change رخ خواهد داد که می‌توانید در روال ()drvDisk_Change درایو پیش‌فرض را تنظیم کنید :

```
ChDrive drvDisk.Drive
```

درصورتی که می‌خواهید بین کنترل‌های کادر لیست دایرکتوری و کادر لیست درایو ارتباط برقرار کنید، باید به صورت زیر عمل کنید :

```
dirDirect.path   drvDisk.Drive
```

با این کار، درایو تعیین شده در کادر لیست درایو تبدیل به درایو پیش‌فرض کادر لیست دایرکتوری خواهد شد. دستور فوق را می‌توانید در روال رویداد ()drvDisk_Change قرار دهید.

برای برقراری ارتباط بین کادر لیست دایرکتوری و کادر لیست پرونده می‌توانید دستور زیر را در روال رویداد Change کادر لیست دایرکتوری قرار دهید :

```
chDir dirDirect.Path
```

روش دیگر ایجاد ارتباط بین کادر لیست دایرکتوری و کادر لیست پرونده استفاده از خاصیت Path است :

```
filFiles.Path   dirDirect.Path
```

قطعه کد زیر برای برقراری ارتباط بین سه کنترل drvDisk و dirDirect و filFiles به‌کار می‌رود.

```
private sub drvDisk_Change ()
        dirDirect.Path   drvDisk.Drive
End sub
```

```
private sub dirDirect.Change ()
        filFiles.path    dirDirect.Path
End sub
```

کادر لیست دایرکتوری دارای مشخصهٔ جالبی به نام ListIndex[1] است : مقدار این مشخصه به اولین زیر دایرکتوری تحت دایرکتوری انتخاب شده اشاره می‌کند. ۱ به خود دایرکتوری اشاره می‌کند. ۲ به یک دایرکتوری بالاتر از آن و الی آخر. اعداد مثبت هم به ترتیب به دایرکتوری‌های پایین‌تر اشاره خواهند کرد.

اگر می‌خواهید کادر لیست پرونده فقط انواع خاصی از پرونده‌ها را نمایش دهد، می‌توانید از خاصیت Pattern این کنترل استفاده کنید :

```
filFiles.Pattern    "*.vbp; *.frm"
```

هنگامی که کاربر پرونده‌ای را انتخاب می‌کند، رویداد Change رخ داده و نام پرونده انتخاب شده در خاصیت FileName قرار داده می‌شود. این کنترل هم (مانند کادر لیست دایرکتوری) دارای مشخصهٔ ListIndex است و پرونده انتخاب شده مقدار ۱ دارد.

**تمرین :** پروژه‌ای ایجاد کنید که بتوان با انتخاب یک پرونده در کادر لیست پرونده، نام پرونده و مسیر کامل آن را در یک کنترل برچسب مشاهده کرد.

**تذکر:** در کادر لیست پرونده، فقط پرونده‌های از نوع TXT و DOC قابل رؤیت باشند.

## ۹ـ۲ـ ذخیره و بازیابی تصویرها

به همان ترتیبی که متن را در یک پرونده ذخیره یا از آن بازیابی کردید، می‌توانید تصویرهای طرح بیتی یا پرونده نشانه را به کمک تابع LoadPicture() (که قبلاً با آن آشنا شده‌اید) از روی دیسک خوانده و در مشخصهٔ Picture کنترل کادر تصویر یا تصویر قرار دهید.

همان‌طور که می‌دانید شکل کلی تابع LoadPicture() به صورت زیر است :

```
ImageCtrl.Picture    LoadPicture (FilePath)
```

---

۱ــ کنترل لیست درایو از نوع Combo Box و دو کنترل دیگر از نوع stBox هستند. لذا هر سه دارای مشخصهٔ L st و L stIndex می‌باشند.

در این شکل کلی تابع :

- ImageCtrl : کنترل کادر تصویر، تصویر یا فرم است.
- Picture : مشخصهٔ Picture شیء است.
- LoadPicture : نام تابع است.
- FilePath : محل دقیق پرونده روی دیسک است.

برای ذخیرهٔ تصویری که در کنترل‌های کادر تصویر، تصویر یا فرم است، از دستور SavePicture استفاده کنید. شکل کلی این دستور به صورت زیر است :

```
SavePicture Picture, strFilePath
```

در این دستور :

- SavePicture نام دستور است.
- Picture تصویری است که در مشخصهٔ Picture کنترل‌های کادر تصویر، تصویر یا فرم قرار داده شده است.
- StrFilePath مسیر و نام پرونده‌ای است که می‌خواهید تصویر را در آن ذخیره کنید.

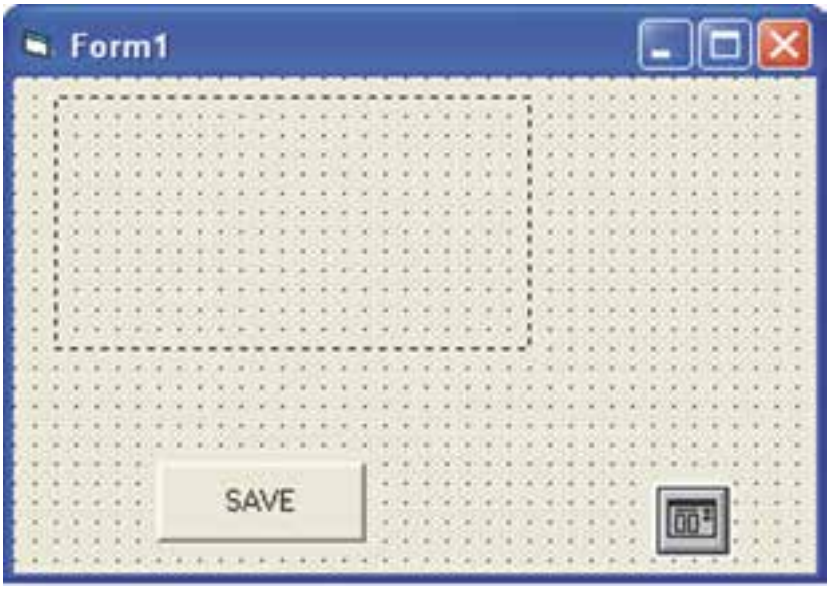
**مثال۵ـ۲**

در این مثال از دستور SavePicture برای ذخیرهٔ تصویر موجود در یک کنترل تصویر (Image) و در محل خاصی روی دیسک، استفاده شده است. این برنامه از یک کادر محاوره‌ای برای تعیین نام پرونده و محل ذخیرهٔ آن استفاده می‌کند.

۱ـ پروژهٔ جدیدی ایجاد کنید.

۲ـ روی Forml کنترل‌های زیر را اضافه کنید :

- کنترل Image به‌نام imgMain
- دکمهٔ فرمان به‌نام CmdImageSave
- Commondialog به‌نام cdMain




شکل ۸ــ۲

```
Private Sub cmdImgSave  Click()
Dim strFilter As String `common dialog filter
Dim strFileName As String `Filename variable
strFilter    "Bitmaps (*.bmp) *.bmp"
cdMain.Filter    strFilter
cdMain.ShowSave
```

اطمینان از خالی نبودن مشخصه filename کادر محاوره‌ای

```
If cdMain.filename<> ""Then
strFileName    cdMain.filename
SavePicture imgMain.Picture, strFileName
MsgBox strFileName & "saved."
End If
End Sub
```

**مثال ۶-۲**

**دفترچه تلفن:** این مثال، پروژهٔ کوچکی است که از پرونده تصادفی استفاده می‌کند. فرم‌های مربوط به این پروژه را به صورت زیر ایجاد کنید.



شکل ۹ــ۲

**توضیح** : در این مثال، مواردی (کنترل‌ها و دستورات) را مشاهده خواهید کرد که قبلاً بیان نشده‌اند و این موارد به دلیل این‌که از اهمیت کمتری برخوردار هستند، به عهده هنرجویان واگذار شده است که در صورت علاقمند بودن می‌توانند به کمک هنرآموز یا راهنمای MSDN اطلاعاتی راجع به آن‌ها کسب کنند.

کد ماژول به صورت زیر است :

```
Public Type TypInfo
```

```
        namel As String * 20
        famil1 As String * 30
        tell As String * 15
    End Type
    Public StrPerson As TypInfo, StrFs As String
    Public StrPath As String
```

نوع دادهٔ TypInfo برای ذخیره‌سازی نام و نام خانوادگی و شماره تلفن افراد تعریف شده است و متغیر StrPerson از این نوع می‌باشد.

کد فرم اول به صورت زیر است :

```
    Private Sub Form  Load()
    StrPath    App.Path          'مسیر فایل را مساوی مسیر Application قرار می‌دهد
        قرار دادن علامت "\" در انتهای مسیر اگر انتهای مسیر علامت \ وجود ندارد.
    If Right (StrPath, 1) <>"\" Then StrPath    StrPath    "\"
    StrFs    StrPath    "tel.dat"  'اضافه کردن نام فایل (tel.dat) به انتهای مسیر فایل
    End Sub
```

در رویداد Form  Loud فایلی با نام tel.dat در مسیر Application درنظر گرفته و نام فایل که شامل مسیر فایل است را در متغیر عمومی StrFs و مسیر فایل را در متغیر عمومی StrPath قرار می‌دهد.

```
    Private Sub cmdadd  Click()
    Dim LngTotalRec As Long          'متغیری برای نگهداری تعداد رکوردها در فایل
    If Dir (StrFs)<> "" Then    'StrFs بررسی وجود فایل
      LngTotalRec    FileLen (StrFs) / Len(StrPerson)
     Else
      LngTotalRec    0          'درصورت موجود نبودن فایل تعداد رکوردها صفر است
    End If
      Open StrFs For Random As #1 Len    Len(StrPerson)
      Seek #1, LngTotalRec    1  'انتقال مکان نما به انتهای فایل برای ذخیرهٔ رکورد جدید
```

```
StrPerson.namel   Trim(UCase(InputBox("Enter name: ","Add Data")))
StrPerson . famil1   Trim (UCase ( InputBox ("Enter famil : ", "Add
Data")))
StrPerson.tell   Trim(UCase(InputBox("Enter Tel: ", "Add Data")))
Put # 1, , StrPerson        ذخیره رکورد جدید در فایل
Close # 1
End Sub
```

در رویداد Click دکمهٔ Cmd Add، نام و نام خانوادگی و شماره تلفن با استفاده از تابع Input Box از کاربر دریافت می‌شود . به کمک تابع UCase تبدیل به حروف بزرگ شده و با استفاده از تابع Trim فضای خالی سمت چپ و راست رشتهٔ دریافت شده حذف می‌شود، سپس مقادیر دریافت شده به‌ترتیب در عناصر name1، Famil 1 و tel 1 رکورد Strperson قرار داده شده و در فایل StrFs ذخیره می‌شود. هنگام ذخیرهٔ رکورد در فایل ابتدا تعداد رکوردهای موجود در فایل را محاسبه می‌کند، برای محاسبهٔ تعداد رکوردها به کمک تابع Dir وجود فایل StrFs بررسی می‌شود درصورت وجود فایل از تقسیم طول فایل بر طول یک رکورد تعداد رکوردها محاسبه می‌شود و درصورتی که فایل StrFs وجود نداشته باشد تعداد رکوردها را صفر قرار می‌دهد. از تعداد رکوردهای ذخیره شده در فایل برای تعیین محل ذخیرهٔ رکورد جدید استفاده می‌شود. برای انتقال اشاره‌گر فایل به محل ذخیرهٔ رکورد در فایل از دستور Seek# استفاده می‌شود.

```
Private Sub cmddelete  Click()
Dim StrPerson As TypInfo
Dim StrOldName As String, StrOldFamil As String, StrFt As String
Dim IntYn, BlnResult As Boolean
Dim LngTotalRes As Long, LngI As Long
  LngTotalRec   FileLen(StrFs)/Len(StrPerson)
  If LngTotalRec   0 Then
        MsgBox "No Existing Data"
        Exit Sub
  End If
```

```
StrOldName    Trim (UCase ( InputBox ("Enter Name for Delete: ", "Delete
Data")))
StrOldFamil    Trim(UCase(InputBox("Enter Famil for Delete: ", "Delete
Data")))
   IntYn    MsgBox("Are You Sure Erasing Data?", vbYesNo, "Erase
   Data")
      If IntYn    vbYes Then
      StrFt    StrPath    "tel.tmp"
      Open StrFs For Random As #1 Len    Len(StrPerson)
      If Dir (StrFt)<> ""Then Kill StrFt       'حذف فایل موقت درصورت وجود
      Open StrFt For Random Access Write As #2 Len    Len(StrPerson)
      BlnResult    False
         For LngI    1 To LngTotalRec    'حلقه برای انتقال رکوردها به فایل موقتی
         Get # 1,, StrPerson          'خواندن رکورد از فایل اصلی
       If Not (Trim(StrPerson.famill)    StrOldFamil And
       Trim(StrPerson.namel)    StrOldName) Then
                                  Put #2, ,StrPerson  'نوشتن رکورد به فایل موقتی
                                     Else
                               BlnResult    True  'نشانهٔ پیدا کردن رکوردی که باید حذف شود
          End If
Next
Close #1, #2
If BlnResult    True Then
   Kill StrFs          'حذف فایل اصلی
   Name StrFt As StrFs      'جایگزین کردن فایل موقتی به جای فایل اصلی
   MsgBox "Eraseing Information Successfuly"
Else
```

```
        Kill StrFt
        MsgBox "Can Not Erase Information"
            End If
    End If
    End Sub
```

رویداد Click دکمهٔ Cmddelete برای حذف رکورد نوشته شده است. در این روال عنصر نام و نام خانوادگی رکوردی که باید حذف شود با استفاده از تابع Input Box از کاربر دریافت شده و توسط توابع Ucase و Trim به ترتیب به حروف بزرگ تبدیل شده و فضای خالی ابتدا و انتهای رشتهٔ دریافتی حذف می‌شود. درصورت تأکید کاربر به حذف رکورد (کلیک دکمهٔ yes کادر پیام) فایل رکوردها (StrFs) و فایل موقتی (StrFt) را باز کرده و تمام رکوردهای موجود در فایل به جز رکوردی که باید حذف شود را در فایل StrFt می‌نویسد در انتها فایل اصلی را حذف کرده (تابع Kill) و فایل StrFt را جایگزین آن می‌کند (به کمک تابع Name نام StrFt را به StrFs تغییر می‌دهد.)

نام فایل موقتی tel.tmp و نام فایل اصلی tel.dat است و هر دو در مسیر Application (App.path) قرار دارند.

```
Private Sub cmdedit  Click()
    Dim StrOldName As String, StrOldFamil As String
    Dim StrNewName, StrNewFamil, StrNewTel As String
    Dim LngTotalRec As Long, LngI As Long, BlnResult As Boolean
        If Dir(StrFs)  ""Then   'tel.dat                    بررسی وجود فایل
          MsgBox "Data File Not Found!"
          Exit Sub
    End If
    LngTotalRec   FileLen(StrFs)/ Len(StrPerson)   'محاسبهٔ تعداد رکوردها
    If LngTotalRec   0 Then
      MsgBox "No Existing Data"
      Exit Sub
```

```
    End If
    StrOldName    UCase (InputBox("Enter Old Name: ", "Edit"))
    StrOldFamil    UCase (InputBox("Enter Old Famil: ", "Edit"))
    StrNewName    UCase(InputBox("Enter New Name: ", "Edit"))
    StrNew Famil    UCase (InputBox("Enter New Famil: "))
    StrNewTel    UCase (InputBox("Enter New Tel: "))
    Open StrFs For Random As #1 Len    Len(StrPerson)
    BlnResult    False
    For LngI    1 To LngTotalRec    حلقه برای پیمایش رکوردها
    Get #1,, StrPerson              خواندن یک رکورد از فایل
        If Trim(StrPerson.famill)  Trim(StrOldFamil)And  Tim(StrPerson.name  1)
Trim(StrOldName) Then
            StrPerson.famill    StrNewFamil
            StrPerson.namel    StrNewname
            StrPerson.tel1    StrNewTel
            Put#1, LngI, StrPerson      'جایگزین کردن رکورد تغییر کرده
            lblfamil.Caption    StrPerson.famill`ها label نمایش عناصر رکورد تغییر کرده روی
            lblname.Caption    StrPerson.namel
            lbltel.Caption    Strperson.tell
            BlnResult    True  'نشانهٔ موفق بودن عملیات تغییر رکورد
    End If
  Next
  Close #1
        If BlnResult    True Then
        MsgBox "Editing Successfully"
  Else
    MsgBox "Not Found Data For Editing"
```

```
End If
End Sub
```

رویداد Click دکمهٔ Cmdedit برای تغییر دادن یکی از رکوردها استفاده می‌شود. در این روال ابتدا نام و نام خانوادگی فعلی، سپس نام و نام خانوادگی و شماره تلفن جدید با استفاده از Input Box دریافت می‌شود سپس تمام رکوردها برای پیدا کردن رکورد موردنظر بررسی می‌شود در صورت پیدا کردن رکورد موردنظر محتوای آن را به مقادیر جدید تغییر می‌دهد. توجه کنید که رکورد جدید در همان محل رکورد قبلی نوشته می‌شود لذا رکورد قبلی حذف شده و رکورد جدید جایگزین می‌شود (Strperson و Lng I و Put #1 در این دستور Lng I شمارهٔ رکوردی است که باید تغییر کند).

```
Private Sub CmdList  Click()
        FrmList.Show 1          'نمایش فرم دوم
End Sub

Private Sub cmdsearch  Click()
Dim StrsearchFamil As String
Dim StrPerson As TypInfo
Dim LngTotalRec As Long, LngI As Long
If Dir(StrFs)   "" Then      'اگر فایل موجود نباشد
        MsgBox "Data File Not Found!"
        Exit Sub
End If
LngTotalRec   FileLen(StrFs) / Len(StrPerson)'بدست آوردن تعداد رکوردها در فایل
If LngTotalRec   0 Then'در صورت خالی بودن فایل
        MsgBox "No Existing Data"
        Exit Sub
```

```
        End If
StrSearchFamil    Trim(UCase(InputBox("Enter Famil For search: ")))
```
دریافت نام خانوادگی برای جستجو
```
        Open StrFs For Random As #1 Len    Len(StrPerson)
        For LngI    1 To LngTotalRec        'پیمایش تمام رکوردهای فایل
                    Get # 1,, StrPerson     'خواندن رکورد شمارهٔ LngI
```
مقداردهی مشخصهٔ Caption برچسب‌ها در صورتی‌که رکورد خوانده شده رکورد مورد جستجو باشد.
```
                    If Trim(StrPerson.famill)    StrSearchFamil Then
                       lblfamil.Caption    StrPerson. famill
                       lblname. Caption    StrPerson.namel
                       lbltel.Caption    StrPerson.tell
                    End If
            Next
            Close #1
        End Sub
```
در رویداد Click دکمهٔ CmdSearch نام خانوادگی فردی که می‌خواهد اطلاعات آن را در دفترچه تلفن جستجو کند از کاربر دریافت کرده و آن را به کمک توابع Ucase و Trim تبدیل به حروف بزرگ نموده و فاصله‌های ابتدا و انتهای آن را حذف می‌کند سپس در صورت وجود فایل و خالی نبودن آن فیلد نام خانوادگی تمام رکوردهای فایل را با نام خانوادگی دریافت شده مقایسه کرده درصورت مساوی بودن محتوای فیلدهای آن رکورد را به وسیلهٔ برچسب‌ها نمایش می‌دهد.

کد فرم دوم به صورت زیر است :
```
        Private Sub CmdBack  Click()
                    Me.Hide         'مخفی کردن فرم دوم
        End Sub


        Private Sub Form  Activate()
```

```
Dim LngI As Long
With msfl                    مقداردهی مشخصه‌های کنترل msflexgrid
        .Rows   1            'تعیین تعداد سطرها
        .Cols   4            'تعیین تعداد ستون‌ها
        .Clear               'خالی کردن جدول
        .ColWidth(0)   500   'تعیین پهنای ستون شماره صفر
        .ColWidth (1)   2000
        .ColWidth (2)   2000
        .ColWidth (3)   2000
    .Width  .ColWidth(0)  .ColWidth(1)  .ColWidth(2)  .ColWidth(3)  50
'تعیین پهنای کل جدول
        .Row   0
    .Col   1
    .Text   "Name"          'نوشتن عبارت Name در خانه شماره (۱و ۰)
    .Col   2
    .Text   "Famil"         'نوشتن عبارت Famil در خانه شماره (۲و ۰)
    .Col   3
    .Text   "Tel"           'نوشتن عبارت Tel در خانه شماره (۳و ۰)
If Dir(StrFs)   "" Then     'در صورت وجود نداشتن فایل
    MsgBox "Data File Not Found!"
    Exit Sub
End If
Open StrFs For Random Access Read As #1 Len   Len(StrPerson)
For LngI   1 To FileLen (StrFs) \ Len(StrPerson)   'پیمایش رکوردهای فایل
    Get #1,, StrPerson      'خواندن رکورد شماره LngI
    .AddItem LngI           'اضافه کردن سطر به جدول
    .Row   LngI
```

```
                نوشتن محتوای فیلد famil رکورد خوانده شده در خانه (1وLngI) 1 .Col
                .Text  Trim(StrPerson.famill)
                نوشتن محتوای فیلد namel رکورد خوانده شده در خانه (2وLngI) 2 .Col
                .Text  Trim (StrPerson.namel)
                نوشتن محتوای فیلد tell از رکورد خوانده شده در خانه(LngI3) (1 .(Col
                .Text  Trim(StrPerson.tell)
            Next
            Close#1
            End With
            End Sub
```

گاهی لازم است که برنامه‌نویس چندین مشخصه از یک کنترل را مقداردهی کند برای مقداردهی مشخصه‌های آن کنترل باید نام کنترل را ذکر کند به مثال زیر توجه کنید.

```
CmdBack.left  200
CmdBack. Top  1200
CmdBack.Width  500
CmdBack.Height  300
```

در این موارد می توان از دستور With استفاده کرد. شکل کلی With به صورت زیر است

```
With          نام کنترل
.1 مشخصه   مقدار مشخصه 1
    ⋮           ⋮
.n مشخصه   مقدار مشخصه n
End With
```

در رویداد Activate فرم دوم، از کنترل Msflexgrid برای نمایش اطلاعات دفترچه تلفن استفاده شده است و برای مقدار دهی مشخصه‌های این کنترل از دستور With استفاده شده است.

برخی از مشخصه‌های کنترل MsFlexgrid در جدول ۵ــ۲ آمده است.

**جدول ۵ــ۲ــ مشخصه‌های کنترل MsFlexgrid**

| مشخصه | شرح |
|---|---|
| Rows | تعداد سطرها |
| Cols | تعداد ستون‌ها |
| Width | پهنای جدول |
| Row | شماره ردیفی که مکان‌نما روی آن قرار دارد |
| Col | شماره ستونی که مکان‌نما روی آن قرار دارد |
| Text | محتوای سلولی از جدول که مکان‌نما روی آن قرار دارد |
| Colwidth(n) | پهنای ستون شماره n (شماره ستون‌ها و سطرها از صفر شروع می‌شود) |

متد AddItem کنترل MsFlexgrid یک سطر به جدول اضافه می‌کند و متد Clear محتوای جدول را پاک می‌کند، (جدول را خالی می‌کند) برای مقداردهی هر سلول جدول ابتدا شمارهٔ سطر و ستون آن سلول (Row و Col) را تعیین کرده سپس مشخصه Text را مقداردهی کنید.

```
msfl. Row   5
msfl.Col    2
msfl.Text    "Cell (5 , 2)"
```

در سلولی که در سطر ۵ و ستون ۲ قرار دارد عبارت Cell (5 , 2) نوشته می‌شود.

## خودآزمایی

۱ــ برنامه‌ای بنویسید که مجموعه‌ای از اعداد صحیح را از پرونده data.dat بخواند و به صورت صعودی آن‌ها را مرتب کند.

۲ــ خطاهای عبارت‌های زیر را پیدا کرده و اصلاح کنید.

الف)
```
Get#6, udtCarInformation   'Store data in record
```

ب)
```
Open #99 For Random Access Append Len   140
```

ج)
```
Put #33, 15, inventory%   'inventory is an array name
```

د)
```
'Open a file for reading and Writing
Open "c: \customer.rnd" Access Read
```

۳ــ مجموعه‌ای از دستورها برای فراهم کردن خواسته‌های زیر بنویسید فرض کنید رکورد

```
Type Person
        lastName As String *15
        firstName As String *15
        age As String *3
End Type
```

از قبل تعریف شده است و پرونده با دسترسی تصادفی به درستی باز شده باشد.

الف) برای ۱۰ رکورد داده وارد کرده و آن‌ها را در پرونده بنویسید.

ب) اطلاعاتی از رکوردها را به هنگام کنید.

ج) یکی از رکوردها را حذف کنید.

۴ــ فرض کنید شما صاحب یک فروشگاه ابزارآلات هستید و نیاز دارید همواره لیستی از موجودی ابزارهای متفاوت، تعداد، هزینه‌ها و قیمت در اختیار داشته باشید. برنامه‌ای بنویسید که با استفاده از یک پرونده تصادفی به نام hardward.dat که دارای صد رکورد خالی است به شما امکان دسترسی به اطلاعات موردنیاز هر کدام از ابزارها را فراهم آورد. این برنامه باید به شما اجازهٔ لیست گرفتن از تمامی ابزارها، حذف ابزاری که مدت طولانی در اختیار نداشته‌اید و همچنین اجازه به هنگام کردن تمام اطلاعات پرونده را بدهد. شماره شناسایی

ابزار باید شمارهٔ رکورد باشد. از اطلاعاتی که در زیر آورده شده است، می‌توانید در پرونده خود استفاده کنید.

| شماره رکورد | نام ابزار | تعداد | قیمت به هزار ریال |
|---|---|---|---|
| 3 | Electric sander | 7 | 57 98 |
| 17 | Hammer | 76 | 11 98 |
| 24 | Jigsaw | 21 | 11 00 |
| 39 | Lawn mower | 3 | 79 50 |
| 56 | Power saw | 18 | 99 99 |
| 68 | Sledgehammer | 11 | 21 50 |
| 77 | Screwdriver | 106 | 6 99 |
| 83 | Wrench | 34 | 7 50 |

۵ــ با یک دستور Close چند پرونده را می‌توان بست؟

۶ــ چه تابعی اولین شمارهٔ پرونده آزاد را برمی‌گرداند؟

۷ــ اگر یک پرونده ترتیبی را برای خروجی باز کنید و آن پرونده موجود باشد، چه اتفاقی خواهد افتاد؟

۸ــ اگر یک پرونده ترتیبی را برای افزودن باز کنید و آن پرونده موجود باشد، چه اتفاقی خواهد افتد؟

۹ــ دستور زیر چه نوع پرونده‌ای را باز می‌کند؟

```
Open "TestFile.dat" For Append As #1
```

۱۰ــ چرا برای باز کردن پرونده‌های تصادفی باید طول هر رکورد معلوم باشد؟

۱۱ــ چرا برای نوشتن نوع دادهٔ کاربر در پرونده، طول رشته‌ها باید مشخص باشد؟

۱۲ــ با کدام دستور Visual Basic می‌توانید نوع دادهٔ کاربر را تعریف کنید؟

۱۳ــ تفاوت تابع Dir با آرگومان و تابع Dir بدون آرگومان چیست؟

۱۴ــ هنرجویی برنامه‌ای نوشته و در آن از دستور زیر استفاده کرده است :

RmDir"c: \Game"

اما اجرای این برنامه با خطا متوقف می‌شود. آیا می‌توانید محتمل‌ترین علت این مشکل را معلوم کنید؟ (فرض کنید پوشهٔ Game در درایو C وجود دارد.)

۱۵ــ روالی بنویسید که یک پرونده ترتیبی ایجاد کرده و اطلاعات زیر را در آن بنویسید : نام، سن، رنگ مورد علاقه. پنج رکورد در این پرونده قرار دهید (هر رکورد باید دارای یک نام، یک سن و یک رنگ باشد). برای نوشتن در پرونده از حلقه‌های For استفاده کنید. راهنمایی : برای هر یک از این مقادیر یک آرایه ایجاد کنید.

۱۶ــ یک کادر محاوره‌ای ایجاد کنید که کادر محاوره‌ای Open ویندوز را شبیه‌سازی کند. فقط از کنترل‌های کادر لیست درایو، دایرکتوری، پرونده و دو دکمهٔ OK و Cancel استفاده کنید. بین سه کنترل درایو، پوشه و پرونده ارتباط برقرار کنید.