فصل ۱

روال‌ها و توابع

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- مفهوم روال، تابع و تفاوت‌های آن‌ها را بیان کند؛
- برنامه‌ها را در صورت لزوم به کمک توابع و روال‌ها به قسمت‌های کوچک تقسیم و برنامه‌نویسی کند.

۱ـ۱ ـ استفاده از روال‌ها در ویژوال بیسیک

ویژوال بیسیک، یک زبان برنامه‌نویسی روالی است. بعد از نامگذاری یک بلاک کد، می‌توان آن را فراخوانی و اجرا کرد. به عبارت دیگر، می‌توان چند خط کد نوشت و آن را در یک بلاک قرار داده و نامی به آن اختصاص داد. سپس بلاک کد را هنگام نیاز فراخوانی کرد. این بلاک کد تقریباً شبیه برنامه‌ای در داخل برنامهٔ دیگر است. این برنامه‌های کوچک را که داخل برنامه‌های بزرگ هستند در صورتی که مقدار برگردانند، " Function " و در غیر این صورت "Sub " (در حقیقت sub مخفف subroutine است.) می‌نامند. روال‌های رویدادی مثل Click() و Load ازنوع Sub هستند و LoadPicture() و Len() توابعی هستند که قبلاً با آن‌ها کار کرده‌اید.

برنامه‌نویسی با این مفاهیم، سال‌هاست که رواج دارد این روش کدنویسی را ساده‌تر، سریع‌تر و کارآمدتر می‌کند. همچنین استفاده از این مفاهیم، امکان نوشتن کدهایی که قابلیت استفادهٔ مجدد را دارند فراهم می‌کند.

روالها امکان تغییر سادهٔ کد را فراهم می‌کنند. اگر نیاز به استفادهٔ مکرر از کدی را دارید، آن را در یک روال قرار دهید. در این صورت اگر نیاز به تغییر کد داشته باشید، به سادگی می‌توانید به آن

رجوع کرده و تغییرات را اعمال کنید. اگر کد را در یک روال قرار ندهید، مجبور خواهید بود که به هر نمونه‌ای از کد در برنامه رجوع کرده و تغییرات مورد نیاز را اعمال کنید؛ البته انجام تغییرات مؤثر و کارآمد با این روش، مشکل خواهد بود.

۲ـ۱ـ ایجاد و فراخوانی یک Sub ساده

یک sub روالی است که خطوطی از کد داخل بلاک را اجرا می‌کند ولی مقداری را برنمی‌گرداند. شکل کلی یک sub ساده به صورت زیر است :

```
[private|public] sub     SubName()
  خطوطی از کد ...
End Sub
```

- [private|public][1] کلید واژه‌های اختیاری هستند که حوزهٔ عمل sub را تعریف می‌کنند.
- Sub کلید واژه‌ای است که نوع روال را تعیین می‌کند.
- SubName نامی است که برای روال تعیین می‌شود.
- End Sub کلید واژه‌هایی هستند که پایان بلاک کد را مشخص می‌کنند.

کد زیر، مثالی از یک sub ساده است :

```
Public Sub DataFinding ( )
   MgBox“Data Not Found”, vbInformation
End Sub
```

هنگامی که این sub را از سایر نواحی کد، فراخوانی می‌کنید، sub کادر پیغامی را با رشتهٔ Data Not Found نمایش می‌دهد.

کد زیر نشان می‌دهد که یک sub با دستور Call فراخوانی شده است. استفاده از دستور Call اختیاری است. اگرچه می‌توان یک sub را بدون کلیدواژهٔ Call فراخوانی کرد (با نوشتن نام آن) ولی استفاده از این کلید واژه، خوانایی کد را افزایش می‌دهد :

```
Private Sub itmOpen _Click( )
   Call DataFinding
End Sub
```

۱ــ حوزهٔ عمل در ادامه همین فصل توضیح داده شده است.

۳ـ۱ـ ایجاد یک Sub ساده

می‌توان یک sub را به دو روش به پروژه اضافه کرد :

- با نوشتن مستقیم کد در بخش General Declarations یک فرم یا مدول.
- با استفاده از گزینهٔ Add Procedure منوی Tools.

نکته : برای فعال کردن گزینه Add Procedure باید در پنجرهٔ code فرم یا مدول موردنظر باشید.

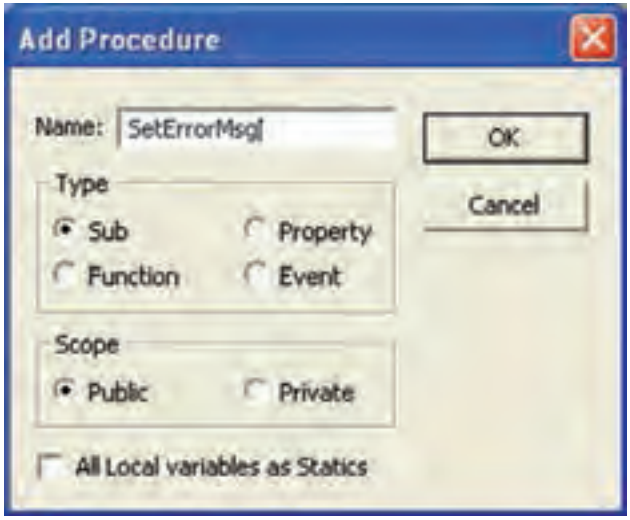
مراحل اضافه کردن sub به پروژه با روش دوم، به صورت زیر است :

۱ــ از منوی Tools گزینهٔ Add Procedure را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود.

۲ــ نامی را برای sub وارد کنید (شکل ۱ــ۱).

۳ــ Sub را از گزینه‌های Type انتخاب کنید.

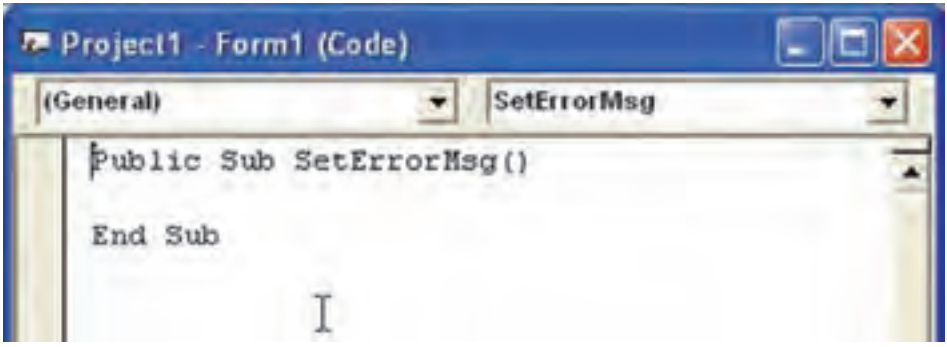
۴ــ از گزینه‌های Scope نوع حوزهٔ عمل sub را انتخاب کنید.



شکل ۱ــ۱ــ کادر محاوره‌ای Add Procedure امکان ایجاد Subs و توابع برای انواع پروژه‌های VB را فراهم می‌کند.

۵ــ روی Ok کلیک کرده و بلاک کد را به فرم یا مدول اضافه کنید (شکل ۲ــ۱).

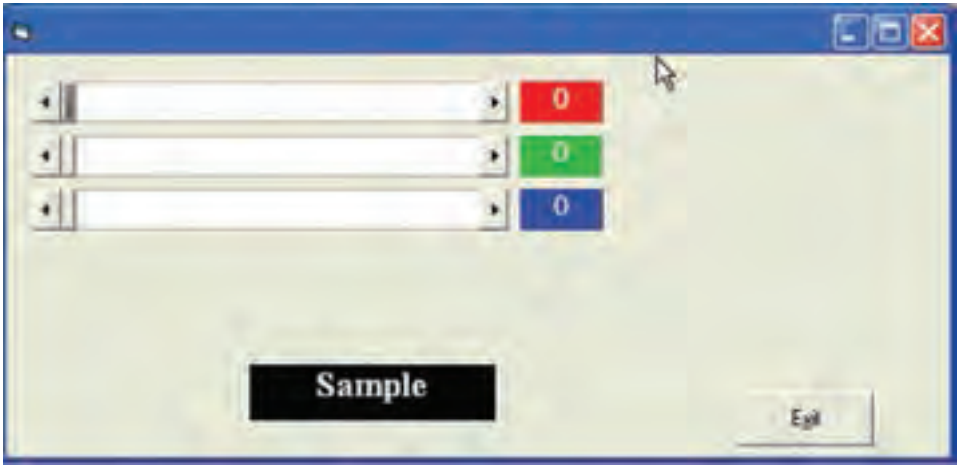
بعد از این که بلاک کد را با کادر محاوره‌ای Add Procedure ایجاد کردید، کد روال را بین اعلان sub و کلید واژهٔ End sub اضافه کنید. بعد از End Sub کدی را وارد نکنید، انجام این کار سبب بروز خطا در زمان کامپایل خواهد شد.



شکل ۲ــ۱ــ sub جدیدی را در بخش General فرم یا مدول به‌دست خواهید آورد.

مثال ۱ـ۱

فرمی به‌صورت شکل ۳ــ۱ ایجاد نمایید که با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی، سبز بتوان رنگ برچسب را تغییر داد.



شکل ۳ــ۱

```
Private Sub setcolor()
    r = HScroll1 . Value
    g = HScroll2 . Value
    b = HScroll3 . Value
    Lblr . Caption = r
    Lbly . Caption = g
    Lblb .  Caption = b
    Label1 .  BackColor = RGB(r,g,b)
End Sub
Private Sub Command1 _Click()
    End
End Sub


Private Sub Form _Load()
    Label1 .  BackColor = RGB(0,0,0)
End Sub
Private Sub HScroll 1_Change()
    Call setcolor
End Sub
Private Sub HScroll2_Change()
    Call setcolor
End Sub
Private Sub HScroll3_Change()
    Call setcolor
End Sub
```

۴ـ۱ـ ایجاد یک تابع ساده

تابع روالی است که خطوطی از کد را اجرا می‌کند و مقداری را برمی‌گرداند. شکل کلی اعلان یک تابع ساده به صورت زیر است :

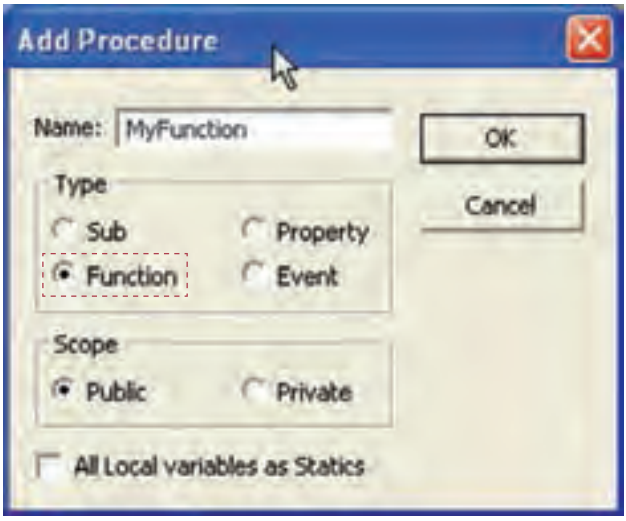
```
[private|Public] Function FunctionName() As DataType
خطوطی از کد ...
    Function Name = ReturnValue
End Function
```

- Private|Public کلید واژه‌های اختیاری هستند که حوزهٔ عمل تابع را تعریف می‌کنند.
- Function کلید واژه‌ای است که مشخص می‌کند، روال از نوع تابع است.
- FunctionName نام تابع است.
- As کلید واژه‌ای برای تعیین نوع داده‌ای است که تابع برمی‌گرداند.
- DataType نوع داده‌ای است که تابع برمی‌گرداند.
- ReturnValue مقداری است که به وسیلهٔ تابع برگردانده می‌شود.
- End Function کلید واژه‌هایی هستند که پایان بلاک کد را مشخص می‌کنند.

کد زیر، تابعی رانشان می‌دهد که مجموع دو عدد تعریف شده در داخل خود تابع را برمی‌گرداند :

```
Public Function GetNumber() As Integer
    Dim a As Integer
    Dim b As Integer
    Dim c As Integer
    a = 7
    b =12
    c = a+b
    GetNumber = c
End Function
```

تابع را نیز می‌توان مانند Sub با همان دو روش به فرم یا مدول اضافه کرد (شکل ۴ـ۱).



شکل ۴ــ۱ــ تابع را در کادر محاوره‌ای **Add Procedure** اضافه کنید.

۵ـ۱ـ ارسال آرگومان‌ها به روال‌ها

آرگومان (پارامتر)، متغیری است که به عنوان جانگهدار برای مقادیری که به تابع یا Sub ارسال می‌شوند، عمل می‌کند. می‌توان قدرت و همه منظوره بودن روال‌ها را با استفاده از آرگومان‌ها افزایش داد. می‌توان آرگومان‌ها را با قرار دادن آن‌ها در داخل پرانتزهای دستور اعلان روال‌ها، ایجاد کرد. کد زیر، اعلان تابع GetGreaterNum که دو آرگومان می‌گیرد را نشان می‌دهد :

```
Public Function GetGreaterNum (NumOne As Integer, NumTwo As Integer) As Integer
```

استفاده از آرگومان‌ها کارایی کد را افزایش می‌دهد. به عنوان مثال، فرض کنید که چندین بار دریک برنامه نیاز دارید که بزرگ‌ترین مقدار بین دو عدد را به‌دست آورید. مناسب‌ترین روش این است که این کد را در یک تابع بنویسید و هر جایی که می‌خواهید آن را فراخوانی کنید.

کد زیر، تابع GetGreaterNum را نشان می‌دهد که بزرگ‌ترین مقدار بین دو عدد دریافتی را محاسبه می‌کند و برمی‌گرداند.

```
Public Function GetGreaterNum(NumOne As Integer,_
NumTwo As Integer) As Integer
    If NumOne> NumTwo Then
        GetGreaterNum=NumOne
```

```
    Else
        GetGreaterNum=NumTwo
    End If
End Function
```

کد زیر چگونگی فراخوانی تابع فوق را از داخل یک روال رویداد Click نشان می‌دهد :

```
Private Sub cmdGreaterNum_Click()
    Dim i As Integer
    Dim j As Integer
    Dim RetVal As Integer
    i= CInt(txtNumOne.Text)
    j= CInt(txtNumTwo.Text)
    RetVal=GetGreaterNum(i,j)
    cmdGreaterNum.Caption=CStr(RetVal)
End Sub
```

هنگام استفاده از آرگومان‌ها یکسان بودن نوع و ترتیب آن‌ها خیلی مهم است. اگر روالی دارید که سه آرگومان از نوع Integer دارد، باید سه عدد صحیح ارسال کنید. در صورتی که دو عدد صحیح و یک رشته ارسال کنید، کامپایلر یک خطا تولید خواهد کرد. به‌عنوان مثال، اگر تابعی به نام EndDay() دارید که به صورت زیر اعلان می‌شود :

```
Public Function EndDay(iNum As Integer, dAccount As Double) As Double
```

و تابع را با استفاده از کد زیر فراخوانی می‌کنید،

```
dMyResult = EndDay (6,"056R")
```

این فراخوانی، خطایی را تولید می‌کند. "056R" از نوع رشته‌ای است ولی تابع برای آرگومان دوم انتظار داده‌ای از نوع Double را دارد.

همچنین تعداد آرگومان‌ها نیز باید یکسان باشند. به عنوان مثال، فرض کنید تابعی دارید که

به‌صورت زیر تعریف شده است :

```
Public Function Bar(iNum As Integer, dNum As double, strName As string)
As Integer
```

و با استفاده از کد زیر، آن را فراخوانی می‌کنید :

```
iMyResult = Bar(6,7)
```

این نیز سبب بروز خطا شود. تابع انتظار سه آرگومان را دارد ولی فقط دو آرگومان ارسال شده است.

می‌توان آرگومانی را به‌این منظور از کلید واژهٔ Optional، در اعلان تابع قبل از آرگومان موردنظر استفاده کرد. آرگومان‌های اختیاری باید از نوع Variant باشند.

۱ـ۵ـ۱ـ کاربرد آرگومان‌های نام‌دار: هنگام فراخوانی روال‌ها می‌توان از آرگومان‌های نام‌دار برای ارسال ساده‌تر مقادیر به آن‌ها استفاده کرد. به‌عنوان مثال، اگر تابعی به نام GetGreaterNum دارای دو آرگومان از نوع Integer باشد و به‌صورت زیر تعریف شود :

```
GetGreaterNum (NumOne As Integer, NumTwo as Integer) As Integer
```

هنگام فراخوانی این تابع بعد از اسامی آرگومان‌ها از نویسه‌های = : استفاده کرده و مقداری را برای آن‌ها به‌صورت زیر تعیین کنید.

```
X=GetGreaterNum(NumOne: =3, NumTwo: =4)
```

۶ـ۱ـ خروج از روال‌ها

بعضی مواقع قبل از پایان روال، نیاز به خروج از آن دارید. می‌توانید این کار را با کلید واژه‌های Exit Function و Exit sub به ترتیب برای خروج از روال‌های از نوع Function و sub انجام دهید. کد زیر مربوط به تابع TestExit()است که دو آرگومان X و Y را دریافت می‌کند و حاصل عبارت زیر را نمایش می‌دهد :

$$F(x,y) = \frac{x^2y + 2y + 4x}{x}$$

اگر مقدار آرگومان X صفر بود، عبارت جواب ندارد و باید از تابع خارج شود. (خطای تقسیم بر صفر)

```
Public Function TestExit(x As Integer, y As Integer) As Integer
```

```
    If x = 0 then
        MsgBox("Division By Zero")
        Exit Function
    Else
        TestExit = (x^2*y+2*y+4*x)/x
    End If
End Function
```

یک مثال برای Exitsub ذکر شود.

۷ـ۱ـ آشنایی با حوزهٔ عمل

حوزهٔ عمل (میدان دید)، قابلیتی است که دو متغیر مختلف با اسامی یکسان می‌توانند مقادیر مختلفی را نگهداری کنند و دارای دورهٔ حیات متفاوتی هستند. کد زیر، دوتابع GetNumber() و Bar() را نشان می‌دهد :

حوزهٔ عمل (میدان دید)، محدودهٔ اعتبار متغیرها را تعیین می‌کند. به مثال زیر توجه کنید :

```
01 Public Function GetNumber() as Integer
02 Dim x as Integer
03 Dim y as Integer
04
05 x = 2
06 y = 7
07 GetNumber = x+y
08 End Function
09
10 Public Function Bar() as Integer
11 Dim x as Integer
12 Dim y as Integer
13
```

```
14 x = 12
15 y = 34
16 Bar = x*y
17 End Function
```

توجه کنید که هر تابع، متغیرهای x و y را اعلان می‌کند. همچنین این متغیرها در هر تابع، مقادیر مختلفی را می‌گیرند. انجام این کار، بدین دلیل است که هر مجموعه‌ای از متغیرها فقط در همان‌جایی که ایجاد شده‌اند به‌کار برده می‌شوند. در تابع GetNumber()، متغیرهای x و y در خطوط ۲ و ۳ ایجاد می‌شوند. هنگامی که تابع خاتمه می‌یابد، متغیرها از حافظه حذف می‌شوند (این را خروج از حوزهٔ عمل می‌نامند). این مطلب دربارهٔ متغیرهای x و y در تابع Bar() نیز صدق می‌کند. برای این که میدان دید متغیرها محدود به روال نباشد، آن‌ها را در بخش General Declarations فرم یا مدول و با استفاده از کلید واژه‌های Public یا Private اعلان کنید.

مثال ۲ـ۱

```
Public Dim x,y As Integer
Public Function GetNumber() As Integer
GetNumber = x+y
End Function
Public Function Bar() As Integer
Bar = x*y
End Function
```

میدان دید متغیرهای x و y در این مثال فرم یا مدولی است که این کد در آن نوشته شده است.

۸ـ۱ـ مستندسازی روال‌ها

مستند سازی روال‌ها به سایر برنامه‌نویسان امکان می‌دهد که به‌طور کامل از برنامهٔ شما استفاده کرده و درصورت لزوم آن را تغییر دهد. زیرا قبل از هر عبارتی، توضیحی برای کاربرد آن نوشته شده است.

تمام روال‌ها دارای یک سرآیند (header) خواهند بود. سرآیند بخشی است که در ابتدای بلاک کد آورده شده و توضیحاتی را دربارهٔ روال ارایه می‌دهد.

۹ـ۱ـ تعیین نقطهٔ ورودی با ()SubMain

به‌طور پیش‌فرض، هنگامی که پروژه‌ای را در VB شروع می‌کنید، پروژه در آغاز اولین فرم ایجاد شده را فراخوانی می‌کند. اگر دارای پروژه‌ای با چندین فرم هستید، برای دسترسی به فرم‌های دیگر می‌توانید آن‌ها را از داخل اولین فرم، بارگذاری (فراخوانی) کنید :

```
Private Sub Form_Load()
    frmAnotherForm. Show
End Sub
```

این روش زمانی مفید است که تعداد فرم‌ها محدود باشد.

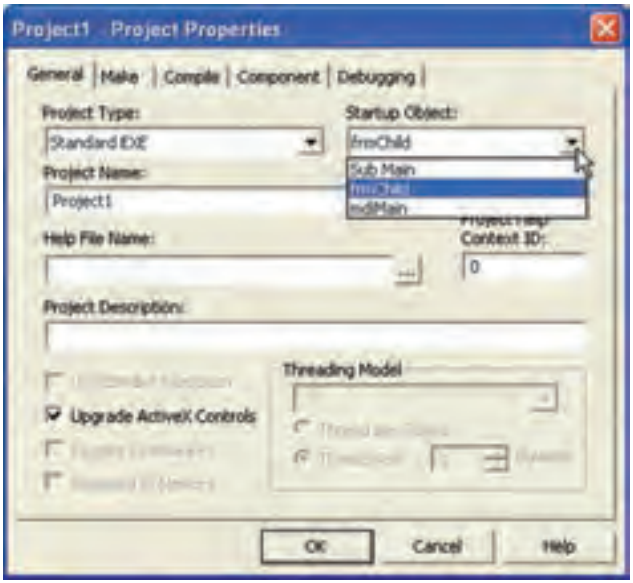
در پروژه‌هایی که دارای هیچ فرمی نیستند (مثل برنامه‌های اینترنتی که در روی سِرورِ کار می‌کنند) و چیزی برای بارگذاری وجود ندارد چه کاری باید برای نقطهٔ شروع (نقطهٔ ورودی) برنامه انجام دهید؟

ویژوال بیسیک، یک نقطهٔ شروع غیر مبتنی بـر فـرم را بـرای برنامه ارایه مـی‌کند (روال Sub Main()). Sub Main() روال خاصی است که به وسیلهٔ ویژوال بیسیک به عنوان روال شروع هر پروژه‌ای رزرو شده است. Sub Main() باید در یک مدول اعلان شود و برای هر پروژه فقط می‌توان یک Sub Main() درنظر گرفت. مراحل زیر، چگونگی تعیین این روال به‌عنوان نقطهٔ شروع را نشان می‌دهند :

۱ـ از منوی Project گزینهٔ Properties مربـوط بـه پروژهٔ جاری را انتخاب کنید تـا کـادر محاوره‌ای مربوطه باز شود.

۲ـ در سر برگ General از لیست بازشوی Startup Object گزینهٔ Sub Main را انتخاب کنید.

۳ـ روی Ok کلیک کنید.



شکل ۵ــ۱ــ می‌توان Sub Main() یا هر فرم دیگری را به‌عنوان شیء شروع در پروژه انتخاب کرد.

بعد از تعیین Sub Main() به عنوان شیء شروع پروژه، باید Sub Main() را در مدول ایجاد کنید. می‌توان از کادر محاوره‌ای Add Procedure برای ایجاد روال‌ها استفاده کرد یا اعلان را در بخش General مدول انتخابی وارد کرد. به خاطر داشته باشید که یک پروژه فقط می‌تواند یک Sub Main() داشته باشد. بعد از ایجاد Sub Main()، نیاز به نوشتن کد Startup دارید. کد زیر یک Sub Main() را نشان می‌دهد که دو فرم را با استفاده از متد Show نمایش می‌دهد و سپس بعد از این‌که همهٔ فرم‌ها ظاهر شدند، کادر پیامی را نشان می‌دهد.

```
Sub Main()
   frmMain.Show
   frmOther.Show
   MsgBox "Everything shown"
End Sub
```

۱۰ـ۱ـ توابع تشخیص نوع داده

این توابع نوع دادهٔ آرگومان ورودی را مشخص می‌کنند. برنامه‌ها معمولاً با داده‌های مختلفی سروکار دارند، ولی گاهی برنامه‌نویس از قبل نمی‌تواند حدس بزند که با چه نوع داده‌ای سروکار خواهد

داشت. مثلاً، قبل از آن‌که محاسبه‌ای انجام دهد باید مطمئن شود که داده‌ها از نوع عددی هستند. در جدول ۱ــ۱ توابع ()Is.... را مشاهده می‌کنید؛ آرگومان این توابع همگی از نوع Variant است.

جدول ۱ــ۱ــ توابع تشخیص نوع داده

تابع	مفهوم
IsDate()	آیا آرگومان تاریخ (یا قابل تبدیل به تاریخ) است؟
IsEmpty()	آیا آرگومان مقدار گرفته؟
IsNull()	آیا آرگومان مقدار Null دارد؟
IsNumeric()	آیا آرگومان یک عدد است (یا می‌تواند به عدد تبدیل شود)؟

در ادامه برای هریک از توابع مثالی ذکر شده است.
در برنامهٔ زیر، چگونگی استفاده از تابع ()IsEmpty نشان داده شده است.

```
‘Code that tests the Is() functions
Dim var1 As Variant, var2 As Variant
Dim var3 As Variant, var4 As Variant
Dim intMsg As Integer      ‘MsgBox return
‘Fill variables with sample values to test
var1 = 0        ‘Zero value
var2 = Null     ‘Null value
var3 = “”       ‘Null string
‘Call each Is() function
If IsEmpty(varl) Then
   intMsg = MsgBox(“var1 is empty”, vbOKOnly)
End If
```

```
If IsEmpty(var2)Then
        intMsg = MsgBox(“var2 is empty”, vbOKOnly)
End If
If IsEmpty(var3)Then
        intMsg = MsgBox(“var3 is empty”, vbOKOnly)
End If
If IsEmpty(var4)Then
        intMsg = MsgBox(“var4 is empty”, vbOKOnly)
End If
```

برنامهٔ فوق پس از اجرا شدن، خروجی زیر را نمایش خواهد داد :

var4 is empty

چون تمام متغیرهای دیگر مقدار گرفته‌اند (حتی Null هم یک مقدار محسوب می‌شود). برای تست کردن Null می‌توانید از ()IsNull استفاده کنید.

نکته : توجه داشته باشید که شرط زیر :

If(varA = Null) Then ....

حتی اگر متغیر varA واقعاً Null باشد، True نخواهد شد. در این موارد تنها راه‌حل استفاده از تابع ()IsNull است.

به قطعه کد زیر توجه کنید :

```
If IsNull(txtHoursWorked) Then
   intMsg = MsgBox (“You didn’t enter hours worked!”,vbOKOnly)
Else                    ‘Thank them for the good hours
   intMsg = MsgBox(“Thanks for entering hours worked!”,vbOKOnly)
End If
```

در این‌جا برنامه قبل از ادامهٔ کار، خالی نبودن یکی از فیلدهای برنامه (textHoursWorked) را بررسی می‌کند.

تابع IsNumeric() با آرگومان‌های عددی (یا هر چیزی که قابل تبدیل به یک عدد باشد) مقدار True را برخواهد گرداند. مقادیر عددی عبارت‌اند از :

- Empty (به صفر تبدیل می‌شود)
- اعداد صحیح (Integer)
- اعداد صحیح بلند (Long)
- اعداد اعشاری (Single)
- اعداد اعشاری با دقت مضاعف (Double)
- واحد پول (Currency)
- تاریخ
- رشته (اگر شبیه یک عدد باشد)

قطعه کد زیر، سن کاربر را (در یک متغیر (Variant) گرفته و در صورت پاسخ اشتباه کاربر، به وی اخطار می‌دهد :

```
Dim varAge As Variant
Dim intMsg As Integer
varAge = InputBox("How old are you?", "Get Your Age")
If IsNumeric(varAge) Then
   intMsg = MsgBox("Thanks!",vbOKOnly)
Else
   intMsg = MsgBox("What are you trying to hide?", _
   vbOKOnly+vbQuestion)
End If
```

درستی پاسخ کاربر در خط ۴ بررسی می‌شود.

اگر می‌خواهید نوع یک متغیر را بدانید، باید از تابع VarType() استفاده کنید. جدول ۲ـ۱ مقادیر برگشتی این تابع را نشان می‌دهد.

جدول ۲ــ۱ــ مقادیر برگشتی تابع VarType()

مقدار برگشتی	ثابت نام دار	نوع داده
0	vbEmpty	Empty
1	vbNull	Null
2	vbInteger	Integer
3	vbLong	Long
4	vbSingle	Single
5	vbDouble	Double
6	vbCurrency	Currency
7	vbDate	Date
8	vbString	String
9	vbObject	Object
10	vbError	یک مقدار خطا
11	VbBoolean	Boolean
12	vbVariant	Variant
13	vbDataObject	یک شیء دسترسی داده
14	vbDecimal	Decimal
17	vbByte	Byte
8194	vbArray	یک آرایه

در برنامهٔ زیر، با دستور Select Case نوع دادهٔ ارسال شده به تابع مشخص شده است.

```
Private Sub PrntType(varA)
Dim intMsg As Integer
Select Case VarType(varA)
```

```
Case 0
        intMsg = MsgBox("The argument is Empty")
Case 1
        intMsg = MsgBox("The argument is null")
Case 2
        intMsg = MsgBox("The argument is Integer")
Case 3
        intMsg = MsgBox("The argument is Long")
Case 4
        intMsg = MsgBox("The argument is Single")
Case 5
        intMsg = MsgBox("The argument is Double")
Case 6
        intMsg = MsgBox("The argument is Currency")
Case 7
        intMsg = MsgBox("The argument is Date")
Case 8
        intMsg = MsgBox("The argument is String")
Case 9
        intMsg = MsgBox("The argument is Object")
Case 10
        intMsg = MsgBox("The argument is Error")
Case 11
        intMsg = MsgBox("The argument is Boolean")
Case 12
        intMsg = MsgBox("The argument is a Variant Array")
Case 13
        intMsg = MsgBox("The argument is a Data Access Object")
```

```
        Case 14
            intMsg = MsgBox("The argument is Decimal")
        Case 17
            intMsg = MsgBox("The argument is Byte")
        Case 8194
            intMsg = MsgBox("The argument is Array")
    End Select
    End Sub
```

۱۱-۱- توابع تبدیل نوع

در جدول زیر، توابع تبدیل نوع را مشاهده می‌کنید؛ به حرف C (سرنام کلمهٔ Convert) در اول نام این توابع دقت کنید. هر تابع آرگومان خود را از نوعی به نوع دیگر تبدیل می‌کند.

توجه دارید که این توابع در صورتی می‌توانند به درستی عمل کنند که امکان تبدیل نوع وجود داشته باشد. مثلاً، عدد ۱۲۳۴۵۶۷۸۹ اساساً امکان تبدیل به نوع Byte را ندارد چون بزرگ‌ترین عددی که یک متغیر Byte می‌تواند در خود ذخیره کند ۲۵۵ است.

برخلاف Int() و Fix()، تابع CInt آرگومان خود را به نزدیک‌ترین عدد صحیح گرد می‌کند. به مثال‌های زیر توجه کنید :

```
intA1 = Cint(8.5)          'Stores an 8 in intA1
intA2 = Cint (8.5001)      'Stores an 9 in intA2
intA3 = Cint (9.5)          Stores an 10 in intA3
```

چون توابع تبدیل نوع می‌توانند روی عبارات هم عمل کنند، می‌توانید حاصل محاسبات را قبل از ذخیره در متغیرها به نوع مناسب تبدیل کنید.

نکته: اگر آرگومان تابع CInt دارای مقدار اعشار 0.5 باشد آن را به نزدیک‌ترین عدد زوج گرد می‌کند.

جدول ۳ــ۱ــ توابع تبدیل نوع

تابع	مفهوم
CBool()	آرگومان خود را به نوع Boolean تبدیل می‌کند.
CByte()	آرگومان خود را به نوع Byte تبدیل می‌کند.
CCur()	آرگومان خود را به نوع Currency تبدیل می‌کند.
CDate()	آرگومان خود را به نوع Date تبدیل می‌کند.
CDbl()	آرگومان خود را به نوع Double تبدیل می‌کند.
CDec()	آرگومان خود را به نوع Decimal تبدیل می‌کند.
CInt()	آرگومان خود را به نوع Integer تبدیل می‌کند.
CLng()	آرگومان خود را به نوع Long تبدیل می‌کند.
CSng()	آرگومان خود را به نوع Single تبدیل می‌کند.
CStr()	آرگومان خود را به نوع String تبدیل می‌کند.
CVar()	آرگومان خود را به نوع Variant تبدیل می‌کند.

۱۲ـ۱ـ تابع Array

تابع Array یک آرایه از نوع Variant را در زمان اجرا، ایجاد کرده و بر می‌گرداند. مرز پایین آرایهٔ بازگشت داده شده، بستگی به Option Base با مقدار ۰ یا ۱ دارد. برنامهٔ زیر چگونگی استفاده از تابع Array را نشان می‌دهد.

```
1. 'Demonstrating function Array
2. Option Explicit 'General declaration
3. Option Base 1   'General declaration
```

```
4. Private sub cmdPrint_Click()
5. Dim v As Variant, x As Integer
6. V = Array (7,5,6,9,3,0)
7. Print "Variant array Values are: ";
8. For x = LBound(v) To UBound(v)
9. Print Format$(v(x),"@@@");
10. Next x
11. Print
12. V = Array("hello","bye","hi")
13. Print"Variant array Values are.";
14. For x = LBound(v) To UBound(v)
15. Print v(x) Space$(2);
16. Next x
17. Print
18. V = Array (1.1,2.2,3.3,4.4)
19. Print"Variant array Values are: ";
20. For x = LBound(v) To UBound(v)
21. Print v(x) & Space$(2);
22. Next x
23. cmdPrint.Enabled = False
24. End Sub
```

در برنامهٔ فوق آرایهٔ V با سه نوع دادهٔ متفاوت (Double، String، Integer) ایجاد و چاپ شده است و برای جلوگیری از امکان چاپ مجدد در انتها کنترل CmdPrint را غیرفعال نموده است.

شکل ۶ ــ۱ــ استفاده از تابع Array()

خودآزمایی

۱ــ برنامه‌ای بنویسید که دو عدد را دریافت کند و با استفاده از یک تابع، عدد کوچک را به توان عدد بزرگ برساند.

۲ــ برنامه‌ای بنویسید که عددی را دریافت کند و فاکتوریل آن را به کمک یک تابع محاسبه کند و نمایش دهد.

۳ــ برنامه‌ای بنویسید که عددی چندرقمی را دریافت کند و با استفاده از یک روال، مجموع ارقام آن عدد را محاسبه کند و نمایش دهد.

۴ــ برنامه‌ای بنویسید که مضارب معادلهٔ درجه ۲ را دریافت کند و با استفاده از یک روال، ریشه‌های آن را به‌دست آورده و نمایش دهد.

۵ــ برنامه‌ای بنویسید که عددی را دریافت کند و با استفاده از یک روال، مقسوم‌علیه‌های آن را به‌دست آورد و نمایش دهد.

۶ــ برنامه‌ای بنویسید که رشته‌ای را دریافت کند و تعداد فضاهای خالی آن را به کمک یک روال، شمارش کند.

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.

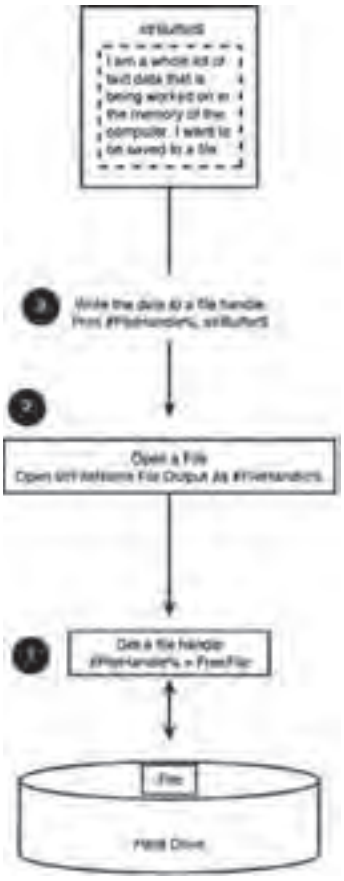
می‌توان با چندین روش، عملیات ذخیره و بازیابی داده‌ها را انجام داد. از پرونده دودویی یا متنی می‌توان برای ذخیرهٔ اطلاعات با اندازه و قالب متفاوت استفاده کرد.

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 (1–255 یا 256–511) مشخص کنید. محدودهٔ پیش‌فرض 1–255 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 Len = CharInBuffer 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)

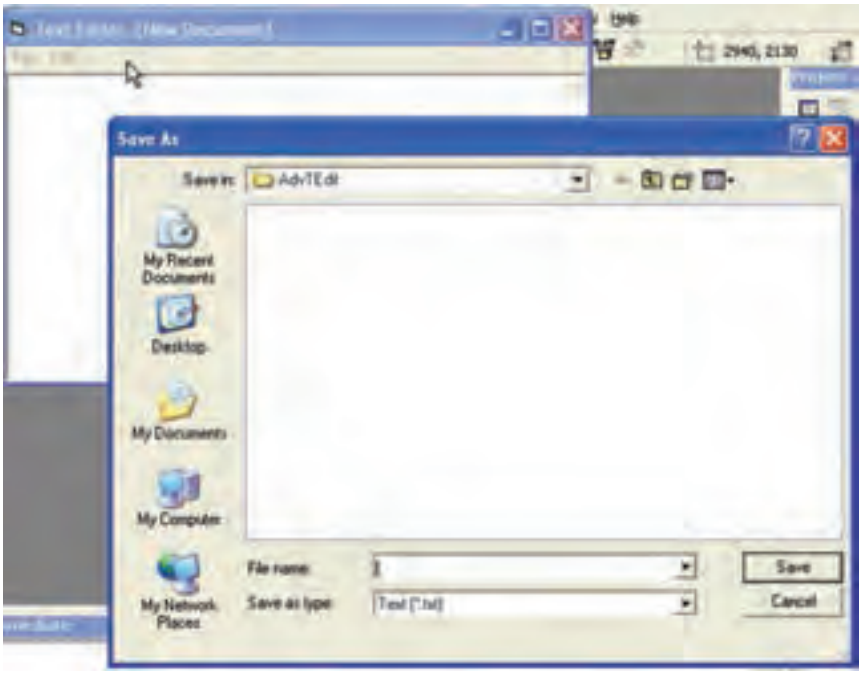
جدول ۲ــ۲ــ عناصر لیست خروج دستور Print #

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 TextBoxی 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
```

```
نوشتن در فایل
Print # FileHandle%, strText

برگرداندن اشاره‌گر ماوس به حالت عادی
MousePointer = vbDefault

بستن فایل
Close # FileHandle%
End If

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.

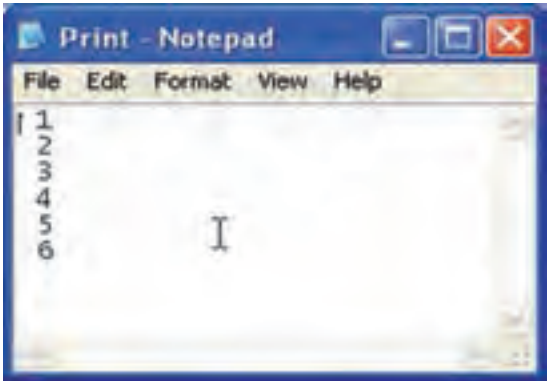
```
Private Sub cmdFile_Click()
    Dim intCtr As Integer   'Loop counter
    Dim intFNum As Integer   'File number
    intFNum = FreeFile
    Open "C: \Print.txt" For Output As #intFNum
    MsgBox "File Print.txt opened"
    For intCtr = 1 To 6
            Print #intFNum, intCtr     'Write the loop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

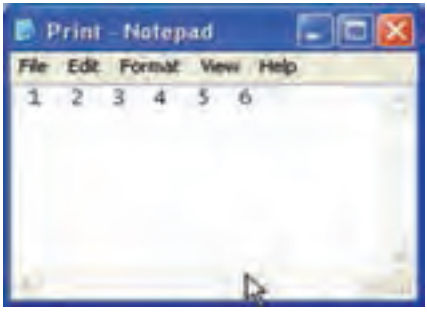
مثال ۳ـ۲

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 &"]"
52 End If
53 End Sub
```

قرار دادن نام فایل در عنوان فرم

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.

۷ـ۲ـ 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.

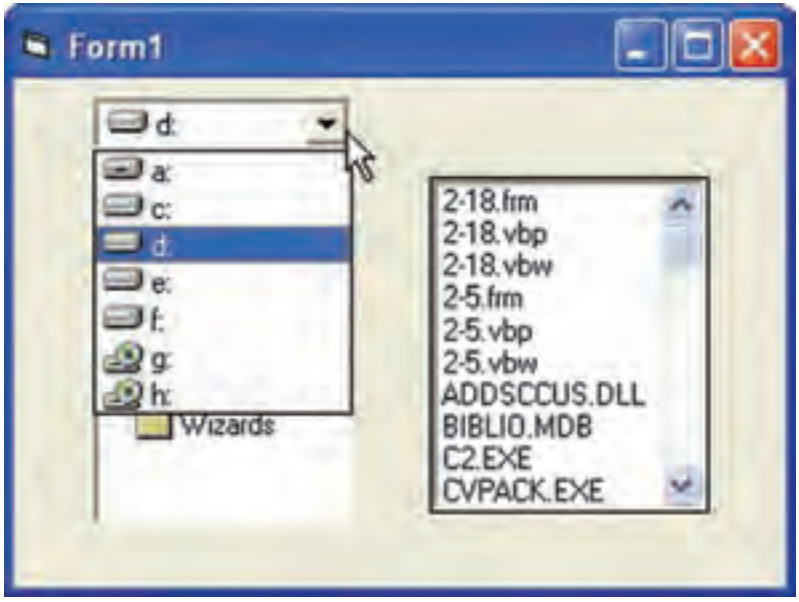
شکل ۶ــ۲ فرمی را با سه کنترل مزبور نشان می‌دهد.

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.

۹-۲- 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 listBox هستند. لذا هر سه دارای مشخصهٔ List و ListIndex 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.

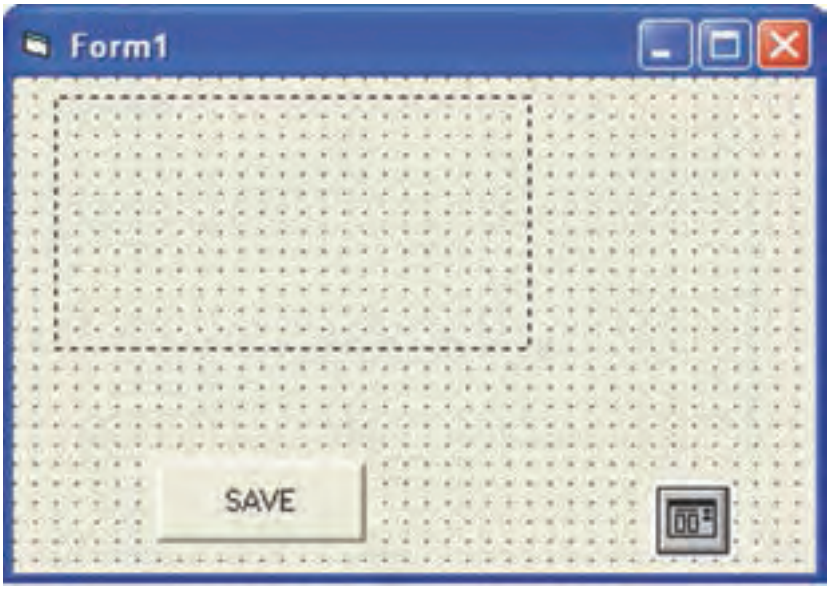
مثال۵ـ۲

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 ۶ـ۲

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    'بررسی وجود فایل StrFs
      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 = 1 مقدار مشخصه
    ⋮              ⋮
.n مشخصه = n مقدار مشخصه
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.

فصل ۳

مفاهیم شیءگرایی و ماژول کلاس

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- مفاهیم اصلی شیءگرایی را توضیح دهد.
- Component Object Model را تعریف کند.
- تفاوت بین ماژول کلاس و کد را شرح دهد.
- یک مؤلفهٔ COM در ویژوال بیسیک ایجاد کند.
- نمونه‌ای از یک کلاس را ایجاد کند.
- مشخصه‌ها و متدهای شیء را به‌کار گیرد.
- روال رویدادی برای مدیریت رویدادهای شیء بنویسد.

۱ـ۳ـ برنامه‌نویسی شیءگرا

برنامه‌نویسی شیءگرا دارای اصول و ویژگی‌هایی است که با پیدایش ویندوز و مفاهیمی مانند «چندوظیفه‌ای[۱]» که در سیستم عامل وجود دارد، مطرح شد. در این روش، برخلاف روش قبلی، به‌جای استفاده از تابع اصلی که وظیفهٔ کنترل تمام برنامه را بر عهده دارد از مفاهیمی هم‌چون کلاس، مشخصه و شیء استفاده می‌شود. بنابراین برای یادگیری بهتر این روش، ابتدا باید با مفاهیم گفته شده آشنا شویم.

۱ــ Multi Tasking

در محیط زندگی با مفهوم شیء آشنا هستیم. همان‌طور که می‌دانید یک شیء چیزی است مادی که دارای مشخصه‌هایی است، از جمله این‌که می‌تواند در مقابل برخی رویدادها که ممکن است برایش رخ دهد، از خود واکنش‌هایی نشان دهد. پس با این حال می‌توان گفت که هر شیء به همراه سه جنبهٔ زیر شناخته می‌شود :

- مشخصه‌ها
- رفتار یا متد
- روابط

مشخصه‌ها، ویژگی‌هایی هستند که مشخص‌کنندهٔ حالت فعلی شیء است. به‌عنوان مثال، می‌توان گفت رنگ یک میز قهوه‌ای است یا قد یک شخص ۱۷۰ سانتی‌متر است. در این مثال، میز و انسان شیء هستند و قهوه‌ای بودن و ۱۷۰ سانتی‌متر بودن قد، مشخصهٔ آن‌هاست.

رفتار یک شیء، نحوهٔ پاسخ آن شیء در مقابل رویدادهایی است که ممکن است برایش رخ دهد. به‌عنوان مثال، شیء میز در مقابل رویداد وارد آمدن نیروی بیش از حد، می‌شکند (رفتار).

تمرین: حال خودتان مثال‌هایی را برای شیء انسان و رویدادهایی که می‌تواند برای وی اتفاق افتد و رفتارهای متقابل او بیابید.

توجه داشته باشید که ممکن است شیء خاصی در مقابل بعضی رویدادها هیچ رفتاری از خود نشان ندهد. این در صورتی است که برای رویداد موردنظر هیچ رفتاری تعریف نشده باشد. روابط هر شیء نیز نشان‌دهندهٔ ارتباط آن شیء با شیءهای دیگر است. به‌عنوان مثال، یک شخص می‌تواند مالک یک شیء مانند میز باشد که در این‌صورت رابطهٔ مالکیت بین شیء و شخص برقرار است.

شیءهایی که در دنیای واقعی وجود دارند، از انواع متفاوت هستند. حتی شیءهای هم‌نوع ممکن است مشخصه‌ها، رفتارها و روابط متفاوتی داشته باشند. با توجه به این نکته، برای شیءها تقسیم‌بندی خاصی را درنظر می‌گیریم و اصطلاح کلاس را تعریف می‌کنیم. کلاس مجموعهٔ تمام شیءهای هم‌نوع است. هر چند این شیءها، مشخصه‌ها، رفتار و روابط متفاوتی داشته باشند.

به‌عنوان مثال، انسان یک کلاس است و هر شخص به خصوصی از این مجموعه، شیئی از کلاس مذکور محسوب می‌شود. با توجه به تعاریف و مفاهیمی که ذکر شد، می‌توان روش برنامه‌نویسی

شیءگرا را به‌صورت زیر بیان کرد :

هر برنامهٔ شیءگرا شامل تعدادی شیء با مشخصه‌ها و متدهای متفاوت است به‌نحوی که روابط خاصی بین آن‌ها برقرار می‌باشد.

متدها، مجموعه‌ای از دستورالعمل‌های برنامه‌نویسی هستند که باید در هنگام بروز رویدادهایی آشکار شوند. مجموعهٔ این دستورالعمل‌ها، رفتار آن شیء را در برابر رویداد به‌خصوصی نشان می‌دهند.

شیءگرایی از بهترین مفاهیم برنامه‌نویسی ساخت یافته به‌وجود آمده و با چندین مفهوم قوی ترکیب شده تا امکان سازماندهی برنامه‌ها به‌طور کارآمد را فراهم کند. به‌طورکلی، هنگامی که با روش شیءگرا برنامه می‌نویسید، مسأله را به بخش‌های تشکیل‌دهندهٔ آن تجزیه می‌کنید. هر مؤلفه‌ای شامل دستورالعمل‌ها و داده‌های مرتبط با خودش است. از طریق این عملیات، پیچیدگی کاهش یافته و می‌توان برنامه‌های بزرگ‌تر را مدیریت کرد. همهٔ زبان‌های برنامه‌نویسی شیءگرا در سه چیز مشترک هستند : کپسوله‌سازی، چندریختی و وراثت.

۱ـ۱ـ۳ـ کپسوله‌سازی (Encapsulation): همان‌طوری که می‌دانید تمام برنامه‌ها از دو عنصر اصلی تشکیل می‌شوند : عبارات برنامه (کد) و داده‌ها. کد بخشی از برنامه است که عملیات را اجرا می‌کند و داده‌ها اطلاعاتی است که به‌وسیلهٔ این عملیات تحت تأثیر قرار می‌گیرند. کپسوله‌سازی، مکانیزم برنامه‌نویسی است که کد و داده‌ها را با هم در یک جا قرار داده و هر دو را از استفاده نادرست و تداخل خارجی ایمن نگه می‌دارد.

در یک زبان شیءگرا، کد و داده‌ها ممکن است با هم در چنین روشی محدود شوند که یک جعبه را ایجاد می‌کنند. درون جعبه تمام داده‌های موردنیاز و کد قرار دارد. هنگامی که در این روش، کد و داده‌ها با هم پیوند برقرار می‌کنند، یک شیء به‌وجود می‌آید. به عبارت دیگر، یک شیء ابزاری است که از کپسوله‌سازی پشتیبانی می‌کند.

اتومبیل دارای چهار چرخ، فرمان، ترمز، بدنه، در، موتور، شمع‌ها، ... و متدهای حرکت چرخ‌ها، روشن و خاموش کردن، عقب و جلو بردن، ... است و تمام اجزای اتومبیل با هم در ارتباطند. راننده از برخی از متدها مثل عقب و جلو بردن، روشن و خاموش کردن استفاده می‌کند و نیاز ندارد که از چگونگی عملکرد موتور یا شمع‌ها (کد متدها) برای حرکت اتومبیل آگاه باشد. لذا اگر عملکرد موتور اتومبیل تغییر کند در رفتار راننده برای استفاده از اتومبیل تأثیری ندارد.

نکته: کپسوله‌سازی سبب می‌شود که اشیاء بدون آنکه از چگونگی عملکرد یکدیگر اطلاع داشته باشند با هم ارتباط برقرار کنند.

۲ـ۱ـ۳ـ چندریختی (Polymorphism) : چندریختی، کمیتی است که به یک رابط امکان می‌دهد تا برای یک کلاس عمومی (مفهوم کلاس را در همین فصل توضیح خواهیم داد)، از عملیات یکسانی استفاده کند. عمل خاص، توسط ذات حقیقی شیء تعیین می‌شود. یک مثال ساده از چندریختی در فرمان اتومبیل است. فرمان اتومبیل برای تمام اتومبیل‌ها بدون توجه به روشی که مورد استفاده قرار می‌دهند، یکسان است. فرمان برای اتومبیلی که به‌طور دستی کار می‌کند یا با نیروی برق یا هر چیز دیگری، عمل یکسانی را انجام می‌دهد. بنابراین، بعد از این‌که شما چگونگی عمل کردن را یاد گرفتید، می‌توانید فرمان هر نوع اتومبیلی را کنترل کنید. همین هدف می‌تواند در برنامه‌نویسی نیز اعمال شود.

در برنامه نویسی شیء گرا می‌توان کلاس‌های عمومی را تعریف کرد مانند کلاس انسان که شامل تمام انسان‌ها (مرد و زن) می‌شود. انسان‌ها دارای مشخصه‌هایی مثل دست، پا، سر، ...، پوشش، احساسات، ... هستند. پوشش و احساسات و بسیاری از ویژگی‌های انسان‌ها در مرد و زن با هم تفاوت دارد در حالی‌که هر دو دارای این ویژگی‌ها هستند لذا کلاس مرد و کلاس زن زیر کلاس‌های[1] کلاس انسان هستند. در مثال دیگر کلاس میوه‌ها را در نظر بگیرید که شامل همهٔ میوه‌ها می‌شود و طعم از مشخصه‌های آن است. کلاس میوه‌ها شامل زیر کلاس‌های سیب، گلابی، موز، ... است که هر کدام طعم مخصوص به خود دارند.

نکته: با استفاده از چند ریختی می‌توان برای متد تعریف شده در کلاس عمومی، پیاده‌سازی متفاوت در زیر کلاس‌ها داشت.

۳ـ۱ـ۳ـ وراثت: وراثت، عملی است که یک شیء می‌تواند مشخصه‌های شیء دیگری را به‌دست آورد. به‌همین دلیل، از مفهوم دسته‌بندی سلسله مراتبی پشتیبانی می‌کند. اگر دربارهٔ وراثت بیشتر فکر کنید، اطلاعات بیشتری راجع به دسته‌بندی سلسله‌مراتبی (از بالا به پایین) به‌دست خواهید آورد. به‌عنوان مثال، سیب قرمز بخشی از دسته‌بندی سیب است که آن هم بخشی از کلاس میوه‌هاست. میوه‌ها هم در کلاس بزرگ‌تری به‌نام غذا قرار دارند. کلاس غذا دارای مشخصات اصلی (خوراکی، پروتئین

۱ــ Subclass

و غیره) است که به‌طور منطقی به زیر کلاس‌های غذا اعمال می‌شود. علاوه بر این مشخصات، کلاس میوه دارای مشخصه‌های (آبدار، شیرین و غیره) است که آن را از سایر غذاها متمایز می‌کند. کلاس سیب نیز مشخصه‌های خاصی را برای یک سیب تعریف می‌کند که عبارتند از : رسیدن روی درخت و غیره. یک سیب قرمز، تمام خصوصیات کلاس‌های بالاتر را به ارث می‌برد و فقط مشخصه‌هایی که منحصر به فرد هستند را تعریف خواهد کرد.

بدون استفاده از وراثت، هر شیء به‌طور مجزا بایستی تمام مشخصه‌های خودش را تعریف کند. با استفاده از وراثت، شیء فقط نیاز به تعریف مشخصه‌هایی دارد که در داخل آن کلاس منحصر به‌فرد هستند. این سبب می‌شود که صفات عمومی را از پدرشان به ارث ببرند.

بنابراین، مکانیزم وراثت به یک شیء امکان می‌دهد تا نمونهٔ خاصی از یک حالت عمومی‌تر باشد.

۲-۳- ماژول‌های کلاس

برنامه‌نویسی شیءگرا برای مدل‌سازی مفاهیم و مسأله‌ها در دنیای واقعی به‌کار می‌رود. از اجزای اصلی برنامه‌نویسی شیءگرا کلاس است. تاکنون با کلاس‌های زیادی در ویژوال بیسیک آشنا شده‌اید. کلاس‌هایی مانند Form، Command Button، Text Box، List Box که هر کدام دارای مشخصه‌ها، متدها و رویدادهای خاص خود هستند و می‌توان به هر تعدادی که لازم است نمونه‌هایی از این کلاس‌ها را در برنامه استفاده کرد. به‌عنوان مثال می‌توان روی فرم، دو کادر متن و یک دکمهٔ فرمان قرار داد که نمونه‌هایی از کلاس‌های کادر متن و دکمهٔ فرمان هستند. نمونه‌های یک کلاس را شیء می‌گویند همان‌طور که به نمونه‌های انواع داده‌ها مثل Integer، String متغیر می‌گویند. به عبارت دیگر کلاس مجموعهٔ تمام شیء‌های هم نوع است ولی شیء نمونه‌ای از یک کلاس است مانند سیب که نمونه‌ای از کلاس میوه‌ها است.

در فصل ۲ با نوع دادهٔ تعریف شده به‌وسیلهٔ کاربر (UDT) آشنا شدید. کلاس، شبیه یک نوع داده تعریف شده به‌وسیلهٔ کاربر است با این تفاوت که علاوه بر داشتن داده‌ها، کلاس دارای زیرروال‌هایی نیز می‌باشد.

کلاس‌ها شامل متغیرها و زیرروال‌ها هستند. متغیرها را اعضای داده‌ای (مشخصه) و زیرروال‌های کلاس را که می‌توانند از نوع زیر برنامه (SUB) یا تابع (Function) باشند، متد می‌گویند. بنابراین، کلاس، مجموعه‌ای از مشخصه‌ها و متدهاست.

به عنوان مثال به نوع دادهٔ Music که به صورت زیر تعریف شده است توجّه کنید.

```
Type Music
    Composer As String
    Piece As String
End type
```

در برنامه روالی به نام Report نوشته شده که مقادیر عناصر نوع دادهٔ Music را در کادر پیام نمایش می‌دهد.

```
Dim my Music As Music
    Private Sub Report ( )
    Dim Msg As String
    Msg = "Composer: " & my Music.Composer & vbcrLF
    Msg =  Msg & "Piece: " & my Music.Piece
    Msg Box Msg                                    'نمایش مقادیر در کادر پیام
End sub
```

اگر بخواهید روال Report به عنوان عنصری از نوع داده Music باشد باید آن را به‌صورت کلاس Music تعریف کنید.

۱ـ۲ـ۳ـ ایجاد ماژول کلاس: در ویژوال بیسیک، کلاس‌ها به‌صورت یک ماژول پیاده‌سازی می‌شوند و تمام اجزای کلاس در داخل ماژول تعریف می‌شوند و ماژول کلاس به پروژه اضافه می‌شود.

ماژول‌های کلاس (پرونده‌های cls.) اصول و اساس برنامه‌نویسی شیءگرا در ویژوال بیسیک است. ماژول‌های کلاس را می‌توان با یک نقشهٔ ساخت خانه مقایسه کرد. دقیقاً مانند یک خانه که از طریق نقشه ساخته می‌شود، شیءهای جدید نیز از طریق ماژول‌های کلاس ایجاد می‌شوند. مشخصه‌ها و متدهای این شیءهای جدید را می‌توانید تغییر دهید. یک ماژول کلاس شبیه یک ماژول کد استاندارد (پروندهٔ bas.) است، به همین دلیل هر دو دارای عملیاتی هستند که می‌توانند به‌وسیلهٔ ماژول‌های دیگری در داخل برنامهٔ کاربردی مورد استفاده قرار گیرند.

- میدان دید داده‌های یک ماژول استاندارد تمام برنامه است. این بدین معنی است که این داده‌ها تا پایان دورهٔ حیات برنامه وجود دارند (معتبر هستند).

- برای هر شیئی که از کلاس ایجاد می‌شود، داده‌های ماژول کلاس به‌وجود می‌آیند. هر دادهٔ شیء فقط در دورهٔ حیات شیء وجود دارد (با ایجاد شیء، به‌وجود می‌آید و با از بین رفتن شیء، از بین می‌رود).
- متغیرهایی که به‌صورت Public در ماژول استاندارد تعریف می‌شوند، در هر جایی از پروژه قابل رؤیت هستند. ولی متغیرهایی که به‌صورت public در ماژول کلاس تعریف می‌شوند فقط با مراجعه به نمونهٔ خاصی از کلاس یا شیء قابل دسترس هستند.

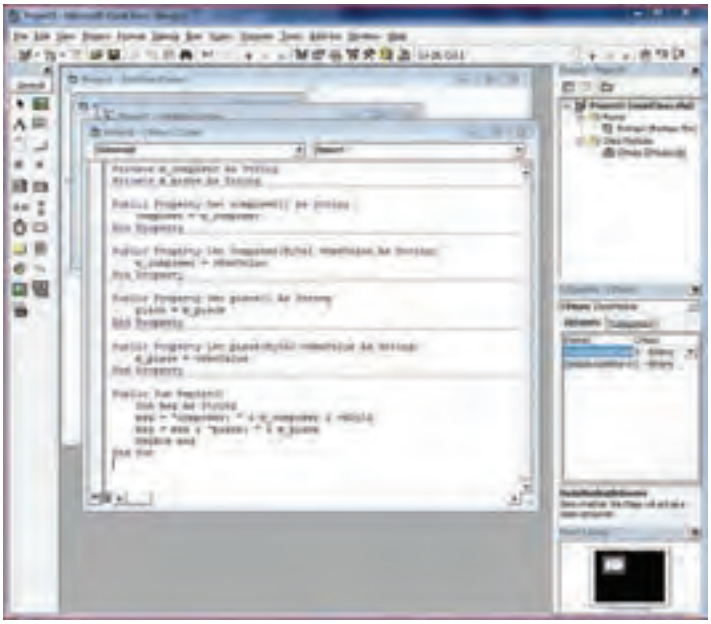
مثال ۱ـ۳

۱ـ پروژه جدیدی را ایجاد کرده، نام آن را Smpl Class.vbp قرار دهید. فرم پیش فرض را به FrmMain تغییر دهید.

۲ـ از منوی Project گزینهٔ Add Class Module را انتخاب کنید.

۳ـ Class Module را انتخاب کرده، دکمهٔ Open را کلیک کنید.

۴ـ در پنجرهٔ Properties مشخصهٔ Name ماژول کلاس را CMusic قرار دهید و سپس کلاس را در فایلی به نام CMusic.cls ذخیره کنید (شکل ۱ـ۳).



شکل ۱ـ۳ـ ایجاد مدول کلاس

۲ـ۲ـ۳ـ اضافه کـردن مشخصه‌هـا بـه کلاس: کلاس CMusic دارای دو مشخصهٔ Composer و Piece از نوع String می‌باشد. برای این مشخصه‌ها در کلاس متغیرهایی در نظربگیرید.

```
Private    m_Composer    AsString
Private    m_Piece       AsString
```

اجـزای کـلاس (مشخصه‌ها و متدها) مـی‌توانند بـه‌صورت Private یا Public تعریف شوند متغیرهایی که به‌صورت Private تعریف می‌شوند فقط به‌وسیلهٔ متدهای کلاس قابل استفاده هستند ولی متغیرهایی که به‌صورت Public تعریف می‌شوند علاوه بر متدهای کلاس، در سایر زیر برنامه‌های موجود در برنامه قابل دسترسی هستند. توصیه می‌شود که متغیرهای کلاس به‌صورت Private تعریف شوند تا فقط از طریق متدهای کلاس قابل دسترسی باشند.

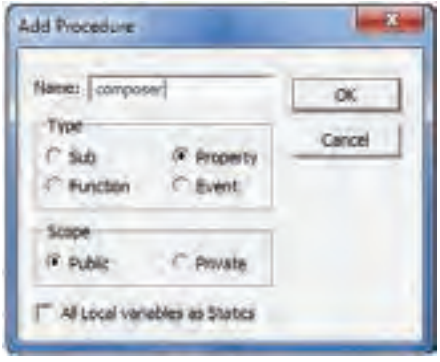
در کلاس CMusic مقادیر واقعی مشخصه‌های Composer و Piece در متغیرهای m_Composer و m_Piece (متغیرهای عضو کلاس) قرار دارند و از آن‌جا که این متغیرها به‌صورت Private تعریف شده‌اند برای دسترسی به آن‌ها (خواندن و تغییر مقدار آن‌ها) روال‌های Let و Get به کلاس اضافه می‌شود.

برای اضافه کردن این روال‌ها مراحل زیر را انجام دهید.

۱ــ پنجرهٔ Code ماژول کلاس را باز کنید.

۲ــ از منوی Tools گزینهٔ Add Procedure را انتخاب کنید.

۳ــ گزینهٔ Property را در کادر محاوره‌ای Add Procedure انتخاب کنید و مشخصهٔ Composer را در کـادر متن Name وارد کنید. روی Ok کلیک کنید (شکـل ۲ــ۳) متدهـای Get Composer و Let Composer به پنجرهٔ کد اضافه می‌شود.



شکل ۲ــ۳ــ اضافه کردن مشخصه به کلاس

```
Public Property Get Composer ( ) As Variant
End Property
Public Property let Composer (Byval vNew Value As Variant)
End Property
```

روال Get برای خواندن مقدار مشخصه Composer است لذا در بدنهٔ روال دستور زیر را بنویسید.

```
Composer = m_Composer
```

این دستور مقدار واقعی مشخصهٔ Composer را که در متغیر m_Composer قرار دارد در مشخصهٔ Composer قرار می‌دهد.

روال Let برای مقداردهی کردن مشخصهٔ Composer است لذا متغیر m_Composer را با آرگومان ورودی این روال مقداردهی کنید.

```
m _Composer = vnewValue
```

توجه کنید که ویژوال بیسیک تمام مشخصه‌ها را از نوع Variant درنظر می‌گیرد. با توجه به این‌که مشخصه Composer از نوع String است کد را اصلاح کنید.

کد برای مشخصهٔ Composer به‌صورت زیر تکمیل می‌شود :

```
Public Property Get Composer ( ) As String
    Composer = m_Composer
End Property

Public   Property    Let Compose (By Val Vnew Vlaue As String)
    m_Composer = Vnew Value
End Property
```

۴ــ مشخصهٔ Piece را مانند مشخصه Composer به کلاس اضافه کنید.

```
Public property Get Piece ( ) As String
    Piece = m_piece
End Property
```

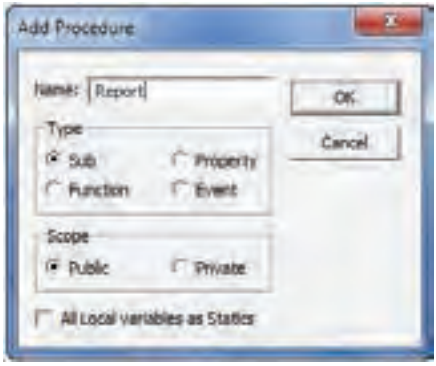
```
Public Property Let Piece (By Val vNewValue As String)
    m_piece = piece
End property
```

۳ـ۲ـ۳ـ اضافه کردن متدها به کلاس : کلاس مجموعه‌ای از مشخصه‌ها و متدها می‌باشد. کلاس CMusic دارای متد Report برای نمایش مشخصه‌های کلاس در کادر پیام است. برای اضافه کردن این متد به کلاس CMusic مراحل زیر را انجام دهید.

۱ـ پنجرهٔ کد ماژول کلاس را باز کنید.

۲ـ از منوی Tools گزینهٔ Add Procedure را انتخاب کنید.

۳ـ در کادر محاوره‌ای Add Procedure نوع Sub را انتخاب کرده و Report را در کادر متن Name وارد کرده، سپس Ok را کلیک کنید. (شکل ۳ـ۳)



شکل ۳ـ۳ـ اضافه کردن متد به کلاس

کد زیر به پنجره اضافه می‌شود.

```
Public Sub Report ( )
End Sub
```

۴ـ متد Report را تکمیل کنید.

```
Public Sub Report ( )
    Dim msg As String
    msg = "Composer: " m_composer & vbCrLf
    msg = msg & "Piece: " & m_piece
    Msg Box msg
End Sub
```

۴ـ۲ـ۳ـ ایجاد شیء از یک کلاس : پس از ایجاد کلاس، می‌توان نمونه‌ای از آن را به‌عنوان شیء در برنامه استفاده کرد. از دو روش برای ایجاد شیء می‌توان استفاده کرد :

۱ــ استفاده از AsNew :

```
Public obgect_name As New Class_name
```

- As New : کلمهٔ کلیدی برای ایجاد شیء
- Object_name : نام شیء که مشابه نام متغیر در اعلان متغیر می‌باشد.
- Class_name : نام کلاسی که شیء از نوع آن کلاس ایجاد می‌شود (مانند نوع داده در اعلان متغیر می‌باشد)

```
Public MyMusic As New CMusic
```

شیء MyMusic را از نوع کلاس CMusic ایجاد می‌کند.

۲ــ استفاده از دستور انتساب Set :

```
Dim object_name As Class_name
Set object_name = New Class_name
```

در خط اول همانند اعلان متغییر، شی‌ای به نام object_name از نوع کلاس Class_name را تعریف می‌کند.

خط دوم مانند دستور مقداردهی به متغیر می‌باشد.

- Set : کلمهٔ کلیدی برای مقداردهی به اشیاء
- New : کلمهٔ کلیدی برای ایجاد یک نمونه از کلاس
- object_name : نام شیء
- Class_name : نام کلاس

```
Dim MyMusic As CMusic
Set MyMusic = New CMusic
```

مثال ۲ـ۳

روی فرم دکمهٔ فرمان گذاشته و Caption آن را Report قرار دهید. رویداد Click دکمه سبب نمایش مشخصات قطعه موزیک در کادر محاوره‌ای می‌شود.

در پنجره کد فرم برنامه زیر را بنویسید.

```
Public MyMusic As New CMusic
```

```
Private Sub Form_Load ( )
    MyMusic.Composer ="Banan"
    MyMusic.Piece = "Elahenaz"
End Sub

Private Sub CmdReport_Click ( )
    MyMusic.Report ( )
End Sub
```

کد زیر مثال بالا را با استفاده از دستور انتساب Set نشان می‌دهد.

```
Public Mymusic As CMusic
Private Sub Form_Load ( )
    Set MyMusic = New CMusic
    MyMusic.Composer = "Banan"
    MyMusic.Piece = "Elahenaz"
End Sub

Private Sub CmdReport_Click ( )
    MyMusic.Report ( )
End Sub
```

تمرین : مثال فوق را به‌گونه‌ای تغییر دهید که مشخصات قطعه موزیک را با استفاده از دو کادر متن به‌وسیلهٔ کاربر دریافت کند.

روال Let که برای تغییر مشخصه‌های کلاس نوشته می‌شود در هنگام استفاده از کلاس به کار نمی‌رود و از خود مشخصه استفاده می‌شود.

```
MyMusic.Composer = "Banan"
```

شیء MyMusic برای مقداردهی مشخصهٔ Composer در دستور بالا از روال Let Composer استفاده می‌کند.

تمرین : با قرار دادن breakpoint روی روال Let در مثال فوق بررسی کنید این روال درچه زمانی فراخوانی می‌شود؟

هنگام نوشتن کدی که از شیءها استفاده می‌کند، بهتر است حافظه‌ای مورد استفاده به‌وسیلهٔ شیءها را پس از پایان کار با آن‌ها آزاد سازید. بعد از این‌که کارتان با یک شیء به پایان رسید از عبارت Set برای مقداردهی Nothing به متغیر شیء استفاده کنید. مثال زیر، حافظهٔ مورداستفاده به‌وسیلهٔ شیء MyMusic را آزاد می‌کند :

```
Set MyMusic = Nothing
```

۵ـ۲ـ۳ـ ایجاد رویدادها : یک شیء اعلان می‌کند که بعضی از عملیات از طریق به‌کارگیری رویدادها ارائه می‌شوند. مشخصه‌ها و متدها، رابط‌های درونی هستند زیرا آن‌ها به خارج شیء ارتباطی ندارند. به‌طور واضح، رویدادها رابط‌های بیرونی هستند، زیرا آن‌ها درون شیء مقداردهی اولیه و خارج از شیء مدیریت می‌شوند. به‌عنوان مثال، هنگامی که مشخصهٔ Caption یک فرم را تغییر می‌دهید، شیء فرم شامل کدی است که این تغییر را اعمال می‌کند و شما این کد را مشاهده نمی‌کنید. ولی روال‌های رویداد را خودتان کدنویسی می‌کنید.

ماژول‌های کلاس شامل دو رویداد درونی هستند : Initialize و Terminate.

- رویداد Initialize هنگامی که نمونه‌ای از کلاس ایجاد و قبل از این‌که مشخصه‌ای مقداردهی شود، رخ می‌دهد. هنگام نوشتن یک ماژول کلاس، رویداد Initialize را برای مقداردهی اولیه هر داده‌ای که به‌وسیلهٔ کلاس مورد استفاده قرار می‌گیرد، به‌کار گیرید.

هم‌چنین می‌توان از این رویداد برای بارگذاری فرم‌های مورد استفاده به‌وسیلهٔ کلاس نیز استفاده کرد.

- هنگامی که متغیر شیء خارج از میدان دید باشد یا با مقدار Nothing مقداردهی شود، رویداد Terminate رخ می‌دهد. هنگامی که ماژول کلاس را می‌نویسید، از این رویداد برای ذخیرهٔ اطلاعات، unload کردن فرم‌ها یا اجرای وظایفی که هنگام خاتمهٔ کلاس رخ خواهند داد، استفاده کنید.

مثال ۲ـ۳

کلاس CSpecialDate برای نمایش تاریخ تعریف شده است این کلاس دارای مشخصهٔ Date Format برای تعیین قالب نمایش تاریخ می‌باشد. کد ماژول کلاس به‌صورت زیر است.

```
Option Explicit
Private m_Format As String                    اعلان مشخصهٔ کلاس
                                    متد Get برای خواندن مشخصهٔ DateFormat
Public Property Get DateFormat ( ) As String
    DateFormat = m_Format
End Propety

                                    متد Let برای نوشتن (مقداردهی کردن) مشخصهٔ DateFormat
Public Property let DateFormat (Byval vNew Value As String)
    m_Format = vNewValue
End Property

                                    متد GetSpecialDate برای نمایش تاریخ با فرمت m–Format
Public Function GetSpecialDate ( ) As String
    GetSpecialDate = Format (Now, m_Format)
End Function

                                    روال رویداد Initialize
Private Sub Class–Initialize ( )
    m–Format = "yy_mm_dd"
End Sub
```

با توجه به کد ماژول کلاس CSpecialDate، این کلاس علاوه بر مشخصهٔ DateFormat، دارای متد GetSpecialDate برای نمایش تاریخ با فرمت تعیین شده در مشخصهٔ DateFormat است. این متد به‌صورت Function نوشته شده است.

روال رویداد Initialize برای این کلاس نوشته شده است و در آن مشخصهٔ DateFormat با مقدار اولیهٔ «yy_mm_dd» برای فرمت نمایش تاریخ مقداردهی شده است.

توجه کنید که مقدار واقعی مشخصهٔ DateFormat در متغیر m_Format قرار دارد. این متغیر به‌صورت Private تعریف شده است، لذا تنها روال‌های موجود در ماژول کلاس به آن دسترسی

دارند و نمونه‌هایی که از کلاس ایجاد می‌شوند نمی‌توانند مستقیماً از این متغیر استفاده کنند و به جای آن از خود مشخصهٔ DateFormat استفاده می‌کنند که سبب می‌شود کلاس از متدهای Let و Get این مشخصه برای تغییر دادن و خواندن مشخصه استفاده کند.

برای استفاده از کلاس CSpeciaLDate روی فرم برچسب Lable1 و دکمهٔ فرمان CmdShowDate را قرار داده و کد زیر نوشته شده است؛

```
Privat Sub CmdShowDate_Click ( )
    Dim oSpecDate1 As cSpeciaLDate          'اعلان متغیری از نوع کلاس
    Dim oSpecDate2 As cSpeciaLDate          'اعلان متغیری از نوع کلاس
    Set oSpecDate1 = New cSpeciaLDate       'ایجاد شیء اول (ospecDate1)
    Set oSpecDate2 = New cSpeciaLDate       'ایجاد شیء دوم (ospecDate2)
    Label1.Caption = oSpecDate1.DateFormat /DateFormat نمایش مقدار مشخصهٔ
    MsgBox oSpecDate1.Get SpeciaLDate       'نمایش تاریخ در کادر پیام
    oSpecDate2.DateFormat = "yyyy-mmm-dd" 'DateFormat تغییر مقدار مشخصهٔ
    MsgBox ospecDate2.GetSpeciaLDate        'نمایش تاریخ در کادر پیام
End Sub
```

در رویداد Click دکمهٔ فرمان دو شیء به نام‌های ospecDate1 و ospecDate2 از نوع کلاس CspecialDate ایجاد شده است. برای شیء ospecDate1 مقدار مشخصه DateFormat را تغییر نداده و تاریخ را با همان قالب تعیین شده در روال رویداد Initialize کلاس نمایش می‌دهد، ولی برای شیء ospecDate2 مقدار مشخصهٔ DateFormat را تغییر داده سپس تاریخ را با قالب جدید نمایش می‌دهد (شکل ۴ــ۳).

شکل ۴ــ۳ــ فرم و کادرهای پیام مثال۳ــ ۳

خودآزمایی

۱ــ مفاهیم زیر را توضیح دهید :

کپسوله‌سازی ــ چندریختی ــ وراثت

۲ــ هدف از ماژول کلاس در ویژوال بیسیک، چیست؟

۳ــ اگر یک مدل شیء مبتنی بر دوچرخه ایجاد کرده باشید، چگونه آن را پیاده‌سازی خواهید کرد؟

۴ــ ماژولی برای کلاس مستطیل بنویسید که دارای متدهایی برای محاسبهٔ محیط و مساحت مستطیل باشد؟

۵ــ کلاس CTime را ایجاد کنید که دارای متغیرهایی برای ذخیره ساعت، دقیقه و ثانیه است و متد ShowTime که زمان را با توجه به مقدار متغیرهایش نمایش می‌دهد.

فصل ۴

آشنایی با

Activex Data Objects (ADO)

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- دلیل استفاده از ADO به عنوان متد دسترسی به داده‌ها را شرح دهد.
- هدف از SQL را شرح دهد.
- شکل کلی عبارات SQL ساده را تعریف کند (عبارت Select).
- رکوردها را در یک پرس‌وجو با استفاده از عبارات سادهٔ SQL فیلتر کند.
- چگونگی اتصال کنترل ADO Data به منبع داده‌ها را شرح دهد.
- کنترل ADO Data را به جعبه ابزار ویژوال بیسیک اضافه کند.
- کنترل ADO Data را در یک برنامهٔ کاربردی به‌کار گیرد.
- چگونگی ارتباط شیء Recordset به کنترل ADO Data را شرح دهد.
- رکوردهای جدیدی را به Recordset اضافه کند.
- رکوردهای موجود را به هنگام کند.
- رکوردها را از Recordset حذف کند.
- رکوردهای درون Recordset را جست‌وجو کند.

تقریباً تمام برنامه‌های کاربردی نیاز به دسترسی به داده‌ها دارند. برای برنامه‌های کاربردی، که در یک کامپیوتر اجرا می‌شوند، دسترسی به داده‌ها و پیاده‌سازی آن ساده است و نیاز به برنامه‌نویسی کمی دارد. مکانیزم دسترسی به داده‌ها تحت شبکه متفاوت است.

به عنوان یک برنامه‌نویس، مجبور خواهید بود که فناوری دسترسی به داده‌هایی که برای ایجاد برنامه به‌کار می‌برید را انتخاب کنید. فناوری‌های دسترسی به داده‌هایی که در ویژوال بیسیک ارایه شده‌اند، زمان ایجاد برنامه را کاهش داده و کد نویسی را ساده می‌کنند و کارایی بالایی را ارایه می‌دهند.

۱ـ۴ـ رابط‌های دسترسی به داده‌ها

یک رابط دسترسی به داده‌ها یک مدل شیء است که شکل‌های مختلف دسترسی به داده‌ها را ارایه می‌کند. در ویژوال بیسیک، سه نوع رابط دسترسی به‌داده‌ها وجود دارد که عبارتند از : Activex Data (ADO)، Remote Data Objects(RDO) و Data Access Objects (DAO).

در ویژوال بیسیک می‌توان با برنامه‌نویسی اتصال به بانک اطلاعاتی، بازیابی رکوردها و تغییر مقدار رکوردها را کنترل کرد.

از هر کدام از سه فناوری دسترسی به داده‌ها برای تعامل با بانک اطلاعاتی می‌توان استفاده کرد، ولی ADO جدیدترین و قوی‌ترین آن‌ها و رابطی برای OLE DB است. OLE DB راهکار رابط سطح پایین برای تمام انواع داده‌هاست. این مفهوم را UDA[1] نیز می‌نامند.

به عنوان مثال، OLE DB و ADO رابط یکسانی را نه تنها برای دسترسی به داده‌های بانک‌های اطلاعاتی رابطه‌ای و غیررابطه‌ای فراهم می‌کنند، بلکه به سایر منابع مثل پست الکترونیکی، سیستم‌های فایل، ابزارهای مدیریت پروژه، صفحه گسترده‌ها و شیء‌های کاری خاص را نیز ارایه می‌کنند.

در کتاب بانک اطلاعاتی با مفاهیم بانک‌های اطلاعاتی رابطه‌ای و عبارت‌های SQL آشنا شده‌اید.

تمرین : مفاهیم زیر را توضیح دهید :

جدول ــ رکورد ــ فیلد ــ کلید

۱ــ Universal Data Access

تمرین: ۱ــ هدف زبان SQL را بیان کنید.

۲ــ عبارت SELECT و اجزای آن را شرح دهید.

۲ـ۴ـ کنترل ADO Data

اگر چه می‌توان از ADO به طور مستقیم در برنامه‌های کاربردی استفاده کرد، ولی ADO Data دارای مزیت کنترل گرافیکی به همراه دکمه‌های پیمایش رکوردی است. همچنین این رابط ساده، امکان می‌دهد تا برنامه‌های کاربردی بانک اطلاعاتی را با حداقل کدنویسی ایجاد کنید.

کنترل ADO Data از Microsoft ADO برای ایجاد اتصال سریع بین کنترل‌های مقید به داده‌ها (data–bound) و ارایه‌کنندهٔ داده استفاده می‌کند. کنترل‌های Data–bound (مقید به داده‌ها)، کنترل‌هایی هستند که دارای مشخصهٔ DataSource می‌باشند و شامل Image, ComboBox, CheckBox و Label, PictureBox, ListBox و TextBoxاست. این کنترل‌ها بانکی از فیلدهای جداول بانک اطلاعاتی متصل شده و مقادیر آن فیلد را نمایش می‌دهند.

به علاوه ویژوال بیسیک دارای چندین کنترل Activex[1] مقید به داده‌ها مثل DataGrid، DataCombo وDatalist نیز است. هنگامی که از کنترل ADO Data استفاده می‌کنید، می‌توانید هر فیلدی را به یک کنترل مقید کنید و هنگام پیمایش رکوردها، به طور خودکار محتوای فیلدها نمایش داده می‌شوند. این‌کار به‌طور داخلی به‌وسیلهٔ ویژوال بیسیک انجام می‌شود و نوشتن هیچ کدی نیاز نیست.

۳ـ۴ـ ایجاد فرم‌های مقید به بانک اطلاعاتی با Data Form Wizard

ویژوال بیسیک ابزاری ارایه می‌کند که امکان ایجاد فرم‌هایی که دارای کنترل‌های مقید به بانک اطلاعاتی هستند را فـراهم می‌کند. می‌توان این ویزارد را از منوی Add–Ins فعال کرد. ویزارد Data Form فرمی را ایجاد می‌کند که امکان مرور بانک اطلاعاتی، کامل کردن کادرهای متن، برچسب‌ها و کنترل ADO Data را فراهم می‌کند.

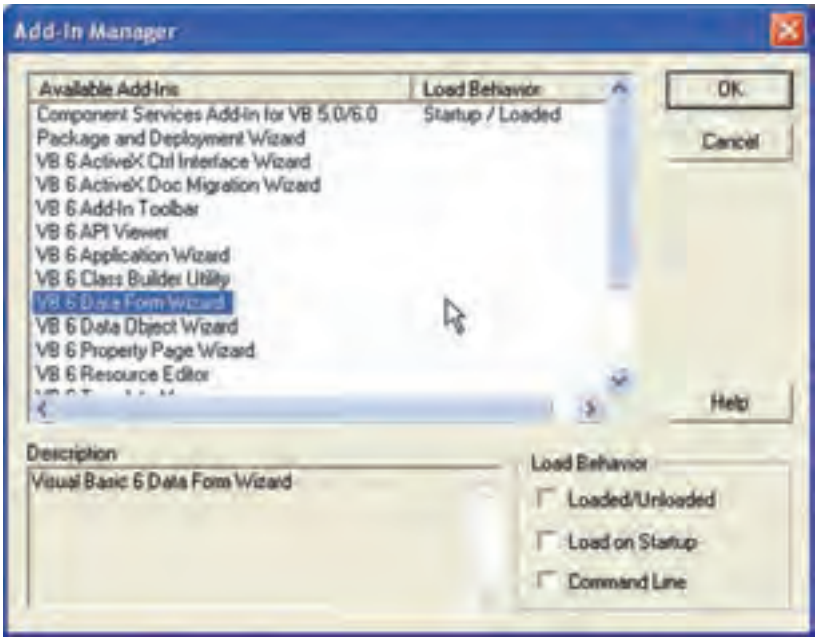
هنگامی که ویژوال بیسیک را نصب می‌کنید، ویزارد Data Form ویژوال بیسیک نصب نمی‌شود و باید آن را به منوی Add–Ins اضافه کنید.

۱ــ کنترل Activex یک مؤلفهٔ نرم افزاری است که می‌تواند در صفحات وب، آفیس یا هر میزبانی که از کنترل‌های Activex پشتیبانی می‌کند، قرار داده شود و دارای متدهایی است که می‌تواند از طریق سایر برنامه‌های کاربردی، کتابخانه‌های پیوند پویا (pll)،صفحات وب یا کنترل‌های دیگر فراخوانی شود.

۱ـ۳ـ۴ـ نصب Data Form Wizard :

۱ــ از منوی Add–Ins گزینهٔ Add–In Manager را انتخاب کنید.

۲ــ در کادر محاوره‌ای Add–In Manager، گزینهٔ VB6،Data Form Wizard را از لیست انتخاب کرده و کادر علامت Loaded/Unloaded را فعال کنید (شکل ۱ــ۴) روی OK کلیک کنید.



شکل ۱ــ۴ــ اگر گزینهٔ Load on Startup را انتخاب کنید، ویزارد Data Form بعد از عملیات نصب به منوی Add–Ins اضافه می‌شود.

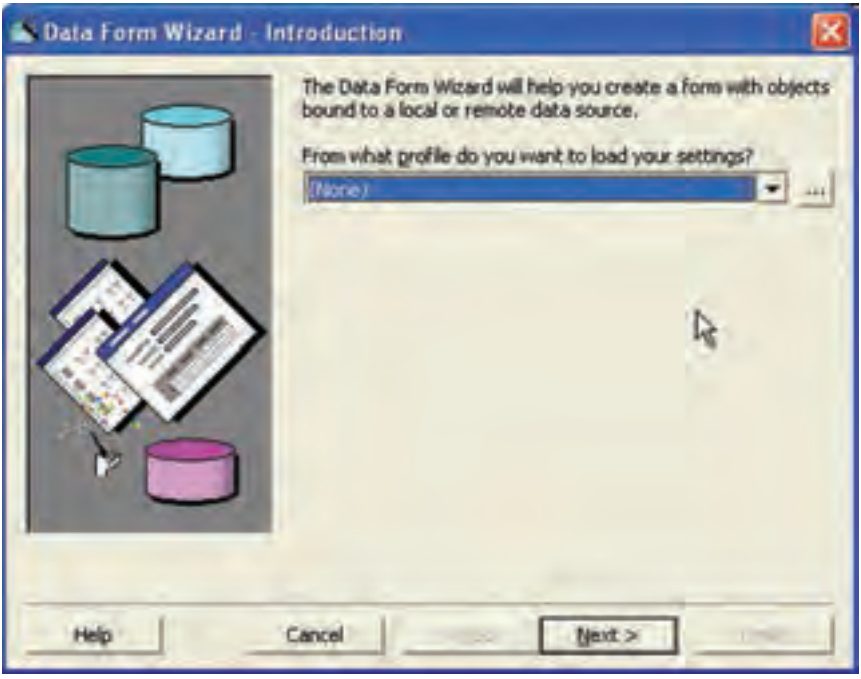
اکنون که ویزارد را به IDE ویژوال بیسیک اضافه کردید، می‌توانید آن را در تمام پروژه‌ها به کار ببرید.

مثال ۱ـ۴

ایجاد فرم مقید به جدول بانک اطلاعاتی : از ویزارد VB Data Form به‌منظور ایجاد فرم برای جدول tblWorks از بانک اطلاعاتی Composer.mdb استفاده می‌کنیم.

۱ــ از منوی Add–Ins گزینهٔ DataForm Wizard را انتخاب کنید.

۲ــ در کادر محاوره‌ای Introduction روی Next کلیک کنید (شکل ۲ــ۴).



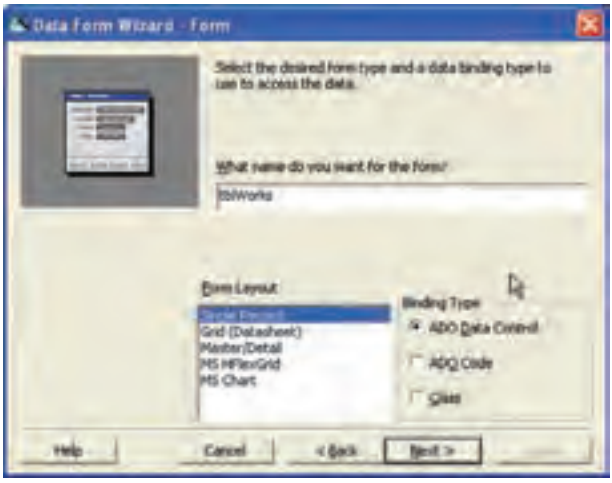
شکل ۲ــ۴ــ کادر محاوره‌ای **Introduction** امکان بارگذاری پروفایل تنظیمات **Data Form Wizard** را فراهم می‌کند.

۳ــ در کادر محاوره‌ای Database Type، انتخاب Access را برگزیده و روی Next کلیک کنید.

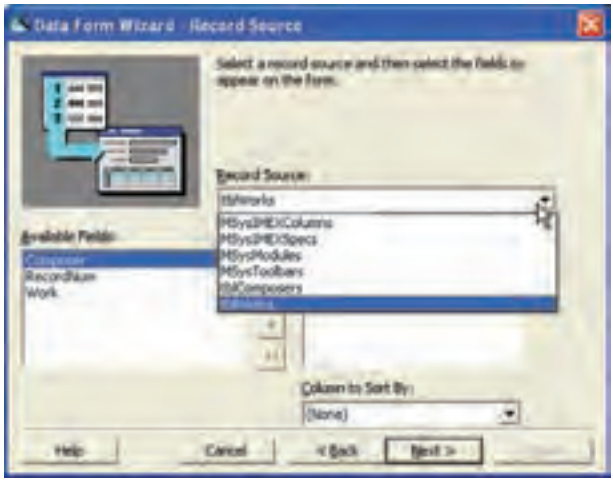
۴ــ در کادر محاوره‌ای Database روی دکمهٔ Browse کلیک کنید تا کادر محاوره‌ای Access Database را مشاهده کنید. اکنون می‌توانید بانک اطلاعاتی مورد نظر برای ایجاد فرم را انتخاب کنید. Composer.mdb را پیدا کرده و روی Next کلیک کنید.

تحقیق: اگر فایل بانک اطلاعاتی در ۲۰۰۳ ,Access XP ایجاد شده باشد، خطا رخ می‌دهد. برای رفع آن چه کاری باید انجام داد؟

۵ــ نام فرم را frmWorks قرار داده و از لیست Form Layout گزینهٔ Single Record را انتخاب کنید. از گزینه‌های Binding Type گزینهٔ ADO Data Control را انتخاب کرده و روی Next کلیک کنید (شکل ۳ــ۴).



شکل ۳ــ۴ــ طرح کلی فرم را با انتخاب گزینه‌ای از لیست Form Layout انتخاب کنید.



شکل ۴ــ۴ــ تمام جدول‌ها و پرس‌وجوهای بانک اطلاعاتی انتخاب شده در لیست بازشوی Record Source فهرست می‌شوند. بعد از انتخاب Record Source، فیلدهای آن در لیست Available fields ظاهر خواهند شد.

۶ــ جدول tblworks را از لیست بازشوی Record Source انتخاب کنید. لیست بازشوی Record Source شامل تمام جداول و پرس‌وجوهای بانک اطلاعاتی انتخاب شده است.

۷ــ تمام فیلدهای جدول انتخاب شده از لیست Record Source در لیست Available Fields ظاهر می‌شود از دکمه‌های ▶ و ▶▶ می‌توان به ترتیب برای انتخاب برخی از فیلدها یا همه فیلدها استفاده کرد. روی دکمهٔ ▶▶ و سپس Next کلیک کنید تا تمام فیلدها انتخاب شود. (شکل ۵ــ۴) در صورتی که

می‌خواهید رکوردها براساس فیلد خاصی مرتب شوند، از کادر Column to sort by نام فیلد موردنظر را انتخاب کنید. روی Next کلیک کنید.

شکل ۵ــ۴ــ هر فیلدی که انتخاب کنید روی فرم داده‌ها ظاهر خواهد شد. در صورتی که بعضی از فیلدها را می‌خواهید، آن‌ها را انتخاب کرده و روی دکمهٔ ▶ کلیک کنید. برای حذف فیلدها از فرم داده‌ها، روی دکمه‌های ◀ و ◀◀ کلیک کنید.

۸ــ در کادر محاوره‌ای Control Selection کادرهای علامت را از حالت انتخاب خارج نکنید، بنابراین تمام کنترل‌های کار کردن با داده‌ها روی فرم قابل دسترس خواهند بود (شکل ۶ــ۴) روی Next کلیک کنید.

شکل ۶ــ۴ــ می‌توان تعداد محدودی از دکمه‌ها را با انتخاب یا پاک کردن کادر علامت مربوطه روی فرم قرار داد.

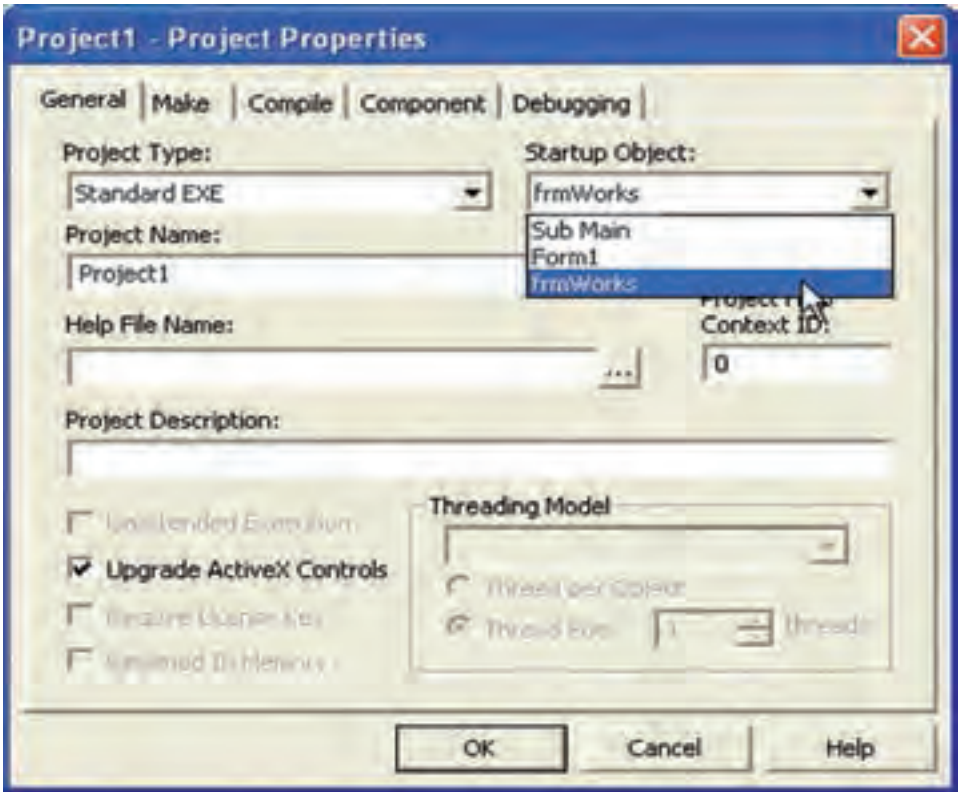
۹ــ در کادر محاوره‌ای پایانی تنظیمات ایجاد شده را با کلیک کردن روی دکمهٔ سه نقطه، ذخیره کنید (شکل ۷ــ۴). این تنظیمات در فایلی با پسوند rwp. ذخیره می‌شوند.

شکل ۷ــ۴ــ ذخیرهٔ تنظیمات در یک پروفایل می‌تواند سبب صرفه‌جویی در وقت برنامهٔ بانک اطلاعاتی شود.

۱۰ــ روی Finish کلیک کنید. کادر محاوره‌ای نهایی Data Form Created ظاهر خواهد شد. درصورتی که نمی‌خواهید این پیام تأیید دفعهٔ بعد ظاهر شود، روی کادر علامت Don't show this Dialog in the Future کلیک کنید. روی OK کلیک کنید.

۱۱ــ کادر محاوره‌ای Project Properties را با انتخاب Project1 Properties از منوی Project باز کنید.

۱۲ــ فرم frmWorks را از لیست بازشوی Startup Object انتخاب کنید (شکل ۸ــ۴).



شکل ۸ــ۴ــ در صورتی که می‌خواهید فرم جدید را به صورت فرم آغازین درنظر بگیرید، باید شیء Startup را دوباره مقداردهی کنید.

۱۳ــ پروژه را ذخیره کرده و اجرا کنید (شکل ۹ــ۴).

شکل ۹ــ۴ــ فرم داده‌های ایجاد شده با ویزارد، امکان اضافه، به هنگام، حذف و نو کردن (refreash) داده‌های جدول را فراهم می‌کند.

کنترل دیگری به نام کنترل Data در جعبه ابزار ویژوال بیسیک وجود دارد که با کنترل ADO Data متفاوت است. برای کار با کنترل Data باید مشخصهٔ Data BaseName آن را مساوی بانک اطلاعاتی موردنظر قرار دارد و در مشخصهٔ Record Source آن یکی از جداول بانک اطلاعاتی را انتخاب نمود.

می‌توان از چندین کنترل Data روی فرم استفاده کرد. هر کنترل Data می‌تواند RecordSource متفاوتی از یک بانک اطلاعاتی یا از بانک‌های اطلاعاتی مختلف داشته باشد. همچنین می‌توان مشخصه‌های کنترل Data را در زمان اجرا تغییر داد.

کنترل Data امکان کار کردن با بانک اطلاعاتی ساده را فراهم می‌کند ولی دارای محدودیت است. برای برنامه‌نویسی حرفه‌ای با بانک اطلاعاتی از فناوری پیشرفته‌ای به نام ActiveX Data Objects (ADO) استفاده کنید. کنترل Data برای برنامه‌های کاربردی ساده‌تر، مناسب است.

۴ـ۴ـ اضافه کردن کنترل ADO Data به جعبه ابزار

برای اضافه کردن کنترل ADO Data به جعبه ابزار، مراحل زیر را انجام دهید :

۱ــ از منوی Project گزینهٔ Components را انتخاب کنید.

۲ــ در کادر محاوره‌ای Components، روی Microsoft ADO Data Control 6.0 (OLE DB) کلیک کنید (شکل ۱۰ــ۴).

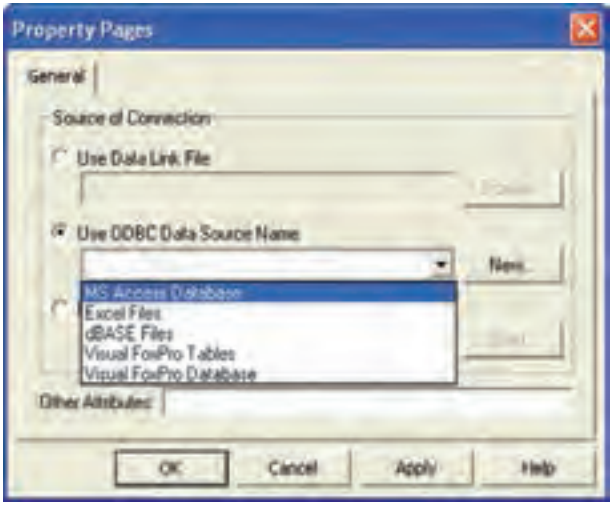
۳ــ روی OK کلیک کنید.

شکل ۱۰ــ۴ــ کادر محاوره‌ای Components

۵ـ۴ـ اتصال به منبع داده‌ها

در زمان طراحی، می‌توان با مقداردهی مشخصهٔ ConnectionString کنترل ADO Data با یک رشتهٔ اتصال معتبر، اتصال به منبع داده‌ها را برقرار کرد. برای مقداردهی این مشخصه روی شیء راست کلیک کرده و گزینهٔ ADODC Propertic را انتخاب کنید (یا در پنجرهٔ Properties روی علامت سه نقطهٔ روبه‌روی مشخصهٔ Connection String کلیک کنید) کادر محاوره‌ای property Page باز می‌شود (شکل ۱۱ـ۴). هنگامی که مشخصهٔ ConnectionString از کنترل ADO Data را می‌خواهید مقداردهی کنید، سه انتخاب خواهید داشت :

- **Use Data Link File:** این گزینه تعیین می‌کند که شما می‌خواهید از یک فایل با پسوند UDL[1] خاص استفاده کنید. هنگامی که این گزینه انتخاب شود، می‌توانید روی Browse کلیک کنید تا به کادر محاوره‌ای Select Data Link File دسترسی پیدا کنید و از آن‌جا پروندهٔ Data Link را انتخاب نمایید.
- **Use ODBC Data Source Name:** این گزینه تعیین می‌کند که شما می‌خواهید از نام منبع داده‌های تعریف شده (DSN) برای رشتهٔ اتصال استفاده کنید. می‌توان به فهرستی از DSNهای تعریف شدهٔ سیستم از طریق کادر ترکیبی دسترسی پیدا کردو DSN موردنظر را انتخاب نمود و یا (شکل ۱۱ـ۴). روی New کلیک کرد و از طریق کادر محاوره‌ای Create New Data Source Wizard برای ایجاد DSN جدید و یا اصلاح DSNهای موجود، استفاده کرد.



شکل ۱۱ـ۴ـ به‌کارگیری گزینهٔ نام منبع داده‌های ODBC

۱ـ Use Data Link

• **Use Connection String:** این گزینه تعیین می‌کند که شما می‌خواهید از یک رشتهٔ اتصال بـرای دسترسی بـه داده‌ها استفاده کنید. روی Build کلیک کنید تـا بـه کـادر محاوره‌ای Data Link Properties دسترسی پیدا کنید. با استفاده از این کادر محاوره‌ای می‌توان اتصال، مجوزهای دسترسی و اطلاعات اضافی مورد نیاز برای دسترسی به داده‌ها را با استفاده از ارایه‌کننده OLE DB تعیین کرد.

۶ـ۴ـ تعیین رشتهٔ اتصال

در روال زیر، ما روی استفاده از رشتهٔ اتصال به منبع داده‌ها تمرکز می‌کنیم. در این فرآیند، یک ارایه‌کنندهٔ OLE DB را انتخاب، نام و محل بانک اطلاعاتی را تعیین و اتصال را آزمایش خواهید کرد.

مقداردهی مشخصهٔ ConnectionString را به صورت زیر انجام دهید :

۱ــ یک کنترل ADO Data روی فرم قرار دهید.

۲ــ در پنجرهٔ Properties مربوط به این کنترل، مقابل مشخصهٔ ConnectionString روی ... کلیک کنید.

۳ــ گزینهٔ Use Connection Stringرا انتخاب و روی Build کلیک کنید.

۴ــ در کادر محاوره‌ای Data Link Properties و زبانهٔ Provider به دلیل این‌که بانک اطلاعاتی که ایجاد کرده‌ایم، در Access بوده است، گزینهٔ Microsoft jet 4.0 OLE DB Provider را انتخاب می‌کنیم.

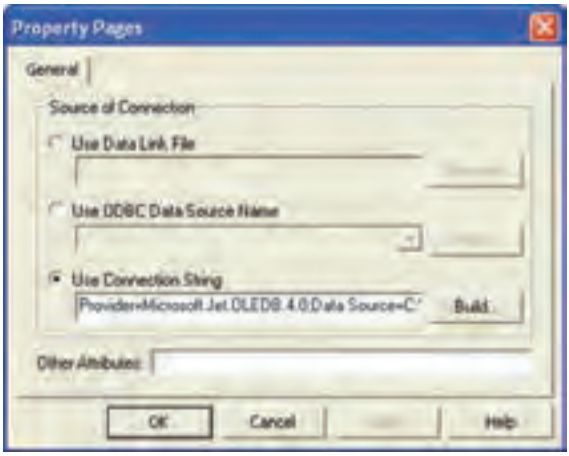
۵ــ در زبانهٔ Connection با کلیک روی ... شمارهٔ ۱ نام بانک اطلاعاتی مورد نظر (در این مثال، Nwind.mdb) را انتخاب کنید. در شمارهٔ ۲ این کادر محاوره‌ای می‌توانید نام کاربری و گذر واژهٔ خاصی را برای دسترسی به بانک اطلاعاتی تعیین کنید.

۶ــ روی Test Connection کلیک کنید تا مطمئن شوید که اتصال برقرار شده است. یک کادر پیام ظاهر می‌شود و اعلان می‌کند که اتصال موفقیت‌آمیز بوده است یا نه؟

۷ــ در صورتی که پیام موفقیت‌آمیز بودن ایجاد اتصال را دریافت کردید، روی OK کلیک کنید و در کادر محاوره‌ای Data Link Properties نیز روی OK کلیک کنید.

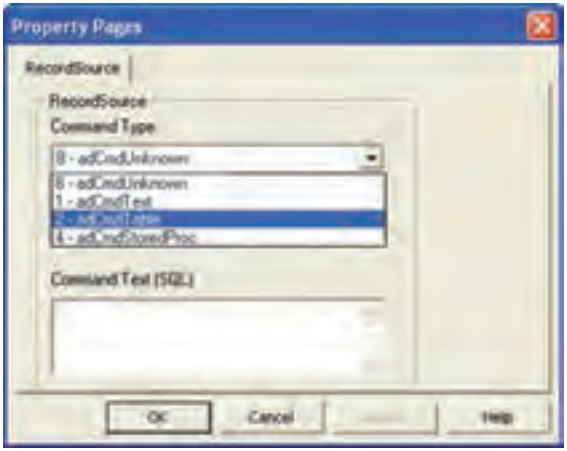
یک مقدار رشته‌ای بـه‌طـور خـودکـار بـــرای Connection String تـولید خـواهد شد (شکل ۱۲ــ۴).

۸ ــ روی OK کلیک کنید تا کادر محاوره‌ای Property Pages بسته شود.



شکل ۱۲ــ۴ــ یک مقدار رشته‌ای اتصال

۱ـ۶ـ۴ـ تعیین مشخصهٔ RecordSource : بعد از مقداردهی مشخصهٔ ConnectionString برای اتصال به بانک اطلاعاتی، می‌توانید مشخصهٔ RecordSource را برای به‌دست آوردن رکوردها، مقداردهی کنید. مشخصهٔ RecordSource می‌تواند با نام جدول، پرس‌وجوی ذخیره شده یا یک عبارت SQL مقداردهی شود. برای بهبود کارایی، از مقداردهی این مشخصه با یک جدول کامل پرهیز کنید. مقدار این مشخصه را برابر با یک رشتهٔ SQL قرار دهید تا فقط رکوردهای مورد نیاز را بازیابی کند. با توجه به اینکه شکل دستورهای SQL در برنامه‌های Microsoft Access و Microsoft SQL Server متفاوت است. بنابراین باید برای هر بانک اطلاعاتی خاص، از شکل دستور مناسبی استفاده کرد. مشخصهٔ Record Source می‌تواند در زمان طراحی و با استفاده از Property Pages مقداردهی شود (شکل ۱۳ــ۴).



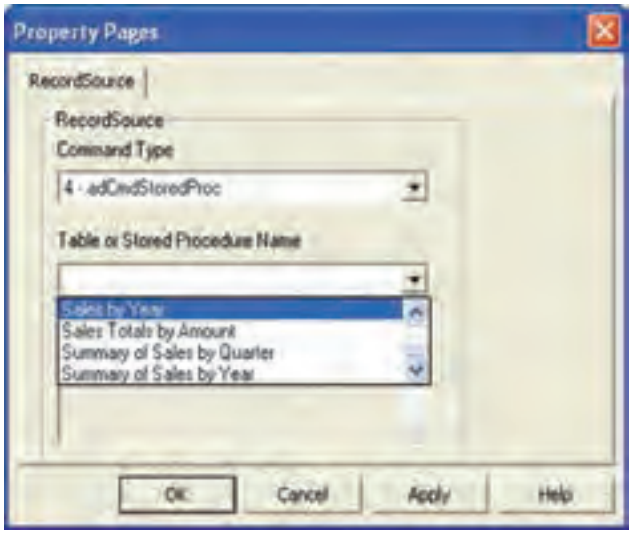
شکل ۱۳ــ۴ــ صفحهٔ مشخصهٔ Record Source

در کادر محاوره‌ای فوق ، پارامتر نوع فرمان را تعیین کنید که به ADO بیان می کند که نوع شیء فرمان مورد استفاده کدام است. جدول ۱ــ۴، گزینه‌های نوع فرمان مختلف را شرح می‌دهد.

جدول ۱ــ۴ــ انواع فرمان ADO

مقدار	شرح
adCmdUnknown	نوع فرمان در مشخصهٔ Command Text ناشناخته است. مقدار پیش فرض است.
adCmd Text	Command Text را به عنوان یک تعریف متنی از یک فرمان یا فراخوانی روال ذخیره شده در نظر می گیرد.
adCmd Table	Command Text را به عنوان نام یک جدول در نظر می گیرد که تمام ستون‌های آن به وسیلهٔ یک پرس‌وجوی SQL که به طور داخلی تولید می‌شود، برگردانده شده است.
adCmd StoredProc	متن فرمان را بنام یک روال ذخیره شده در نظر می گیرد. این می‌تواند یک روال ذخیره شده در بانک اطلاعاتی SQL Server یا یک پرس‌وجو در Access باشد.

اگر adCmd Table یا adCmd StoredProc را انتخاب کنید، از کادر لیست زیرین، می توانید نام جدول یا روال ذخیره شده را انتخاب کنید(شکل ۱۴ــ۴).



شکل ۱۴ــ۴ــ انتخاب یک روال ذخیره شده به عنوان RecordSource

تمرین :

اتصال کنترل ADO Data به منبع داده‌ها : در این تمرین، کنترل ADO Data را به فرم اضافه خواهید کرد و سپس آن را به منبع داده‌ها متصل می‌کنید. منبع داده‌ها، بانک اطلاعاتی نمونهٔ Northwind (Nwind.mdb) است که درون ویژوال بیسیک قرار دارد.

با چگونگی اضافه کردن کنترل ADO Data به جعبه ابزار قبلاً آشنا شدید.

۱ــ کنترل ADO Data را به Form1 اضافه کنید.

۲ــ روی کنترل ADO Data کلیک راست و ADODC Properties را انتخاب کنید.

کادر محاوره‌ای Property Pages ظاهر می‌شود. در زبانهٔ General، مطمئن شوید که گزینهٔ Connection String انتخاب شده باشد.

۳ــ برای تعیین رشتهٔ اتصال، روی Build کلیک کنید.

۴ــ Microsoft jet OLE DB Provider را انتخاب و روی Next کلیک کنید.

به دلیل این‌که بانک اطلاعاتی از نوع Access است این گزینه انتخاب شده است.

۵ــ از زبانهٔ Connection برای انتخاب یا تایپ نام بانک اطلاعاتی استفاده کنید. روی ... کلیک کنید.

۶ــ در کادر محاوره‌ای SelectAccessDatabase، روی Nwind.mdb و سپس open کلیک کنید.

۷ــ در کادر محاوره‌ای Data Link Properties روی Test Connection کلیک کنید.

یک کادر پیام ظاهر شده و بیان می‌کند که اتصال موفقیت‌آمیز بوده است یا نه؟

۸ــ روی OK کلیک کنید.

۹ــ تا این‌جا، مقدار رشتهٔ اتصال ایجاد شده است، روی زبانهٔ RecordSource از کادر محاوره‌ای Property Page کلیک کنید.

۱۰ــ برای Command Type گزینهٔ adCmd Text ــ1 را انتخاب کنید.

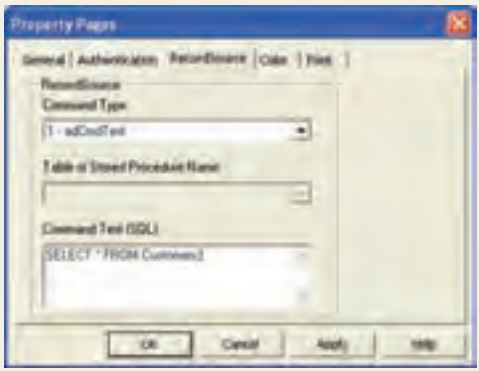
۱۱ــ برای مقدار Command Text (SQL)، عبارت SQL زیر را تایپ کنید (شکل ۱۵ــ۴).

```
SELECT * FROM Customers
```

۱۲ــ روی OK کلیک کنید تا کادر محاوره‌ای Property Pages بسته شود.

اکنون کنترل ADO Data مقید به منبع داده‌هاست و می‌توانید اطلاعات را از جدول Customers در بانک اطلاعاتی Nwind.mdb بازیابی کنید.

۱۳ــ از منوی File گزینهٔ Save Project را انتخاب کنید و پروژه را با نام PrjADO.vbp ذخیره کنید تا در تمرین بعدی نیز از آن استفاده کنید.



شکل ۱۵ـ۴ـ کادر محاوره‌ای Property Pages

۷ـ۴ـ مقیدسازی کنترل‌ها

بعد از تعیین مشخصه‌های ConnectionString و RecordSource برای کنترل ADO Data، که به ترتیب بانک اطلاعاتی و یکی از جداول یا Query‌های بانک اطلاعاتی را مشخص می‌کند، می‌توانید یک کنترل مقید برای نمایش داده‌ها روی فرم اضافه کنید.

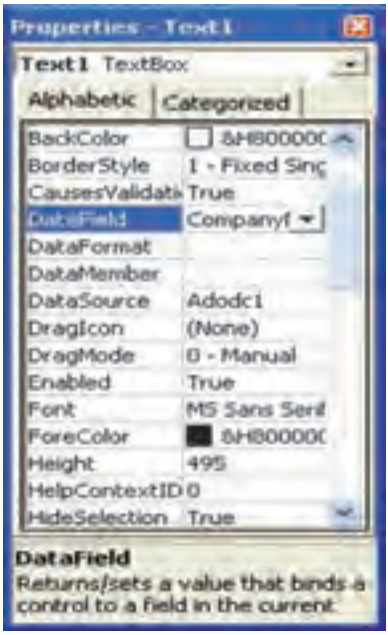
هنگامی‌که با کنترل ADO Data بین رکوردها جابه‌جا شوید، کنترل‌های مقید نیز داده‌های فیلدها را در رکوردها نمایش می‌دهند. با تغییر مقدار یک فیلد و جابه‌جایی بین رکوردها، این تغییر به طور خودکار به بانک اطلاعاتی اعمال می‌شود. مزیت استفاده از کنترل‌های مقید، به حداقل رساندن مقدار کدنویسی است. به دلیل این‌که مقدار کنترل مقید به طور خودکار از بانک اطلاعاتی بازیابی و در آن نوشته می‌شود، به برنامه‌نویسی کمی نیاز دارد.

۱ـ۷ـ۴ـ مقداردهی مشخصه‌های DataSource و DataField : برای این‌که کنترلی را به کنترل ADO Data مقید کنید، باید این دو مشخصه را تنظیم نمایید. مشخصهٔ DataSource منبع داده را از طریق کنترلی که به بانک اطلاعاتی مقید است، تعیین می‌کند (در این مثال، کنترل ADO Data).

شیء RecordSet شامل مجموعه‌ای از رکوردهای جدول یا Query است که به‌وسیلهٔ مشخصهٔ

RecordSource تعیین شده است. برای نمایش مقادیر فیلدهای این رکوردها در کنترل‌های مقید از مشخصهٔ DataField کنترل‌های مقید استفاده می‌کنیم.

مشخصهٔ DataField نام فیلد معتبری که در شیء Recordset به‌وسیلهٔ منبع داده‌ها ایجاد شده است را تعیین می‌کند. این مقدار تعیین می‌کند که کدام فیلد در کنترل مقید، نمایش داده شود. این دو مشخصه را می‌توان در زمان طراحی و از طریق پنجرهٔ Properties مقداردهی کرد (شکل ۱۶ــ۴).



شکل ۱۶ــ۴ــ تنظیم مشخصه‌های Datasource و Datafield

هم‌چنین می‌توان این دو مشخصه را در زمان اجرا نیز مقداردهی کرد. اگر می‌خواهید مشخصهٔ DataSource را در زمان اجرا با استفاده از کد مقداردهی کنید، باید از کلید واژهٔ set استفاده کنید، زیرا این مشخصه یک شیء است. مثال زیر، این دو مشخصه را برای کنترل Text box مقداردهی می‌کند.

```
set txt1.DataSource = Adodc1
txt1.DataField = "CompanyName"
```

تمرین:

مقید سازی کنترل‌ها

در این تمرین از پروژهٔ ایجاد شده در تمرین قبلی استفاده خواهیم کرد :

۱ــ پروژهٔ PrjADO.vbp را باز کنید.

۲ــ دو کادر متن در بالای کنترل ADO Data اضافه کنید.

۳ــ مشخصهٔ DataSource هر دو کادر متن را Adodc1 قرار دهید.

۴ــ مشخصهٔ DataField کادر متن اول (Text1) را برابر با CompanyName قرار دهید.

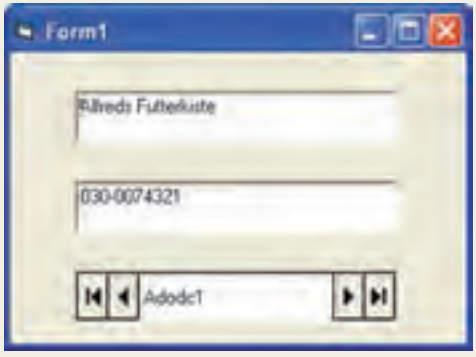
۵ــ مشخصهٔ DataField کادر متن دوم (Text2) را برابر با Phone قرار دهید.

۶ــ پروژه را اجرا کنید.

شما می‌توانید از طریق کلیک کردن روی فلش‌های کنترل ADO Data، اسامی شرکت‌ها و شماره تلفن‌های آن‌ها را مشاهده کنید. ظاهر فرم شبیه شکل ۱۷ــ۴ خواهد بود.

۷ــ روی End کلیک کنید.

۸ــ پروژه را ذخیره کنید.



شکل ۱۷ــ۴ــ نام شرکت و شماره تلفن در کنترل‌های مقید نمایش داده می‌شوند.

۸ـ۴ـ برنامه‌نویسی بانک اطلاعاتی با کنترل‌های ADO

همان‌طور که می‌دانید از کنترل ADO Data می‌توان برای پیوند خودکار با یک بانک اطلاعاتی استفاده کرد. کنترل ADO Data روش خوبی برای مرور رکوردها و ویرایش یک رکورد در هر لحظه است. علاوه بر مقیدسازی داده‌ها با کنترل ADO Data، می‌توان یک فرم ورود داده‌های ساده را به سرعت ایجاد کرد.

۱ـ۸ـ۴ـ کدنویسی کنترل ADO Data : بعد از این‌که چگونگی تنظیم مشخصه‌های کنترل ADO Data را برای اتصال به بانک اطلاعاتی و مقید سازی کنترل‌ها را آموختید. در این قسمت، دربارهٔ کارکردن با داده‌ها از طریق این کنترل، آشنا خواهید شد. این کنترل، امکان مشاهده و ویرایش رکوردها بدون نوشتن کد را فراهم می‌کند. با این وجود، برای پشتیبانی ویژگی‌های پیشرفته‌تر، نیاز به نوشتن کد خواهید داشت. با استفاده از این کنترل و مشخصه‌ها، متدها و رویدادهای کنترل مقید، می‌توان کنترل بیش‌تری روی چگونگی تعامل با داده‌های خارجی اعمال کرد.

کنترل ADO Data دارای مشخصه‌ای به نام Recordset است که گروهی از رکوردها می‌باشد.

مشخصهٔ Recordset این کنترل، خودش یک شیء است و دارای مشخصه‌ها و متدهای خاص خودش می‌باشد. شیء Recordset رکوردهایی را از جدول پایه یا حاصل اجرای یک پرس‌وجو ارایه می‌دهد. شیءهای Recordset با استفاده از رکوردها (سطرها) و فیلدها (ستون‌ها) ساخته می‌شوند. می‌توان از این شیءها برای کار کردن با داده‌های بانک اطلاعاتی استفاده کرد. در هر لحظه، شیء Recordset فقط به یک رکورد اشاره می‌کند که رکورد جاری نامیده می‌شود.

برای بازیابی مجموعه‌ای از رکوردها، مشخصهٔ RecordSource کنترل ADO Data را مقدار دهی کنید. مشخصهٔ RecordSource یک مقدار رشته‌ای است که می‌تواند نام جدول یا پرس‌وجو در یک بانک اطلاعاتی باشد.

هم‌چنین می‌توان مشخصهٔ Filter از شیء Recordset را برای انتخاب رکوردهای خاصی به‌کار برد. به عنوان مثال، می‌توان تعیین کرد که مجموعه رکورد شامل فقط رکوردهایی باشد که CustomerID آن‌ها بزرگتر از ده است.

```
adoCustomers.Recordset.Filter = "CustomerID>10"
```

بعد از این‌که مشخصهٔ RecordSource مقداردهی شد، می‌توانید مشخصه‌های BOF و EOF شیء Recordsetرا بررسی کنید. این مشخصه‌ها تعیین می‌کنند که در ابتدا یا انتهای Recordset هستید. اگر رکورد جاری اولین رکورد باشد، مقدارBOF برابر با Trueخواهد بود و اگر اشاره‌گر در انتهای رکوردها باشد، EOF برابر True است. در صورتی که هیچ رکوردی درRecordset نباشد، مقدار هر دو مشخصهٔ EOF و BOF برابر با True خواهد بود.

۲-۸-۴- ویرایش رکوردها در Recordset: اگر چه کنترل ADO Data می‌تواند به طور خودکار رکوردها را بدون نیاز به اضافه کردن کد، ویرایش و به هنگام کند، ولی می‌توان متد Update را برای ویرایش داده‌ها به جای استفاده از فلش‌های کنترل ADO ، به کارگرفت.

کنترل ADO Data به‌طور خودکار رکوردهای بانک اطلاعاتی را در مواقع زیر تغییر می‌دهد :

۱ــ جابه‌جایی به رکوردی که می‌خواهید ویرایش کنید.

۲ــ تغییر هر نوع اطلاعاتی که در کنترل‌های مقید نمایش داده می‌شوند.

۳ــ کلیک کردن روی هر کدام از فلش‌های کنترل ADO Data برای جابه‌جایی به رکورد دیگر.

با این وجود، به‌کارگیری دکمهٔ فرمان امکان اضافه کردن کد به رویداد Click برای انجام کارهایی مثل بررسی صحت فیلدها روی فرم، قبل از به هنگام سازی را فراهم می‌کند. برای انجام عمل به‌هنگام‌سازی رکورد جاری، از متد Update استفاده کنید. به عنوان مثال، در رویداد Click دکمه‌ای

مانند Update، می‌توان کد زیر را اضافه کرد :

```
adoCustomers.Recordset.Update
```

در صورتی که می‌خواهید از تغییرات اعمال شده در رکورد جاری؛ صرف‌نظر کنید یا رکورد جدیدی را قبل از فراخوانی متد Update درج کنید، می‌توانید از متد Cancel Update استفاده کنید. شکل کلی متد Cancel Update در مثال زیر نشان داده شده است.

```
adoEmployees.Recordset.CancelUpdate
```

۳ـ۸ـ۴ـ اضافه کردن رکوردها به Recordset: رکوردهای جدید را با فراخوانی متد AddNew می‌توان به Recordset اضافه کرد. متد AddNew کنترل‌های مقید را مقداردهی اولیه می‌کند و رکورد جدید، رکورد جاری می‌شود. اگر هم‌زمان با ویرایش رکورد دیگری، AddNew را فراخوانی کنید، ADO به‌طور خودکار متد Update را فراخوانی می‌کند تا تغییرات را ذخیره کند و سپس یک رکورد جدید ایجاد می‌کند. کد مثال زیر، رکورد جدیدی را به مجموعه رکورد adoCustomers اضافه خواهد کرد :

```
adoCustomers.Recordset.AddNew
```

برای ذخیرهٔ تغییرات در رکورد جدید، می‌توان متد Update شیء Recordset را فراخوانی کرد یا می‌توان روی یکی از دکمه‌های پیمایش کنترل ADO Data کلیک کرد.

۴ـ۸ـ۴ـ حذف رکوردها از Recordset: با استفاده از متد Delete می‌توان رکورد جاری یا گروهی از رکوردها را در مجموعه رکورد حذف کرد. متد Delete دارای یک پارامتر AffectRecords است که برای تعیین تعداد رکوردهایی که این متد تحت تأثیر قرار خواهد داد، مورد استفاده قرار می‌گیرد.

جدول ۲ــ۴ــ مقادیر پارامتر Affect Records

مقدار AffectRecords	شرح
adAffectCurrent	این گزینه فقط رکورد جاری را حذف خواهد کرد و گزینهٔ پیش‌فرض است.
adAffectGroup	این گزینه می‌تواند حذف تمام رکوردهایی که از مشخصهٔ Filter عبور کرده‌اند را دربرگیرد. برای استفاده از این گزینه باید ابتدا مشخصهٔ Filter را مقداردهی کرد.

بازیابی مقادیر فیلد از رکوردهای حذف شده، خطایی را تولید می‌کند. بعد از حذف رکورد جاری، رکورد حذف شده جاری باقی می‌ماند تا زمانی که به رکورد دیگری جابه‌جا شوید. بعد از این‌که از رکورد حذف شده جابه‌جا شدید، در دسترس نخواهد بود. برای به‌کارگیری متد Delete از شکل کلی زیر استفاده کنید :

```
Recordset.Delete AffectRecords
```

برای مشاهدهٔ این‌که آخرین رکورد حذف شده است یا نه، مشخصهٔ EOF را بررسی کنید. اگر EOF برابر با True بود، به آخرین رکورد منتقل شوید.

```
حذف رکورد جاری /  adoCustomers.Recordset.Delete
جابه‌جایی به رکورد بعدی /  adoCustomers.Recordset.MoveNext
آیا آخرین رکورد حذف شده است؟/ If adoCustomers.Recordset.EOF=True Then
انتقال به آخرین رکورد جدید/  adoCustomers.Recordset.MoveLast
End IF
```

نکته : بانک اطلاعاتی Northwind دارای قواعد جامعیت ارجاعی (referential integrity) تعریف شده است که از حذف رکوردهای Recordset جلوگیری می‌کند. به عنوان مثال، نمی‌توان یک رکورد از جدول Customer را حذف کرد، در صورتی که رکوردهای مرتبطی در جدول orders داشته باشد.

۵-۸-۴- جستجوی رکوردها : برای افزودن ویژگی جستجو به برنامهٔ کاربردی‌تان، از متد Find مربوط به مجموعه رکورد (Recordset) کنترل ADO Data استفاده کنید. متد Find مجموعه رکورد موجود را برای رکوردی که دارای شرط خاصی است، جستجو می‌کند. اگر شرط با رکوردی مطابقت داشته باشد، اشاره‌گر روی آن رکورد قرار می‌گیرد، در غیر این‌صورت به انتهای مجموعه رکورد (EOF) منتقل می‌شود. شکل کلی متد Find به صورت زیر است.

```
Recordset.Find Criteria, [SkipRows],[Search Direction],[Start]
```

متد Find دارای پارامتر ضروری شرط (Criteria) و سه پارامتر اختیاری است که عبارتند از : SkipRows، SearchDirection و Start.

پارامتر شرط (Criteria) رشته‌ای است که شامل عبارتی است که نام ستون، عملگر مقایسه و

مقدار مورد استفاده در جستجو را شامل می‌شود. عملگر مقایسه در شرط ممکن است، بزرگتر(>) ، کوچکتر (<)، مساوی (=) یا like (مطابقت الگو) باشد.

در مثال زیر، اولین مشتری که در ایالت واشنگتن زندگی می کند، برگردانده خواهد شد :

adoCustomers.Recordset.Find "state='WA'"

مقدار شرط ممکن است یک رشته، عدد ممیز شناور یا تاریخ باشد. مقادیر رشته‌ای با تک کوتیشن مشخص می‌شوند (به‌عنوان مثال،"state='WA'"). و مقادیر تاریخ با علامت # مشخص می‌شوند (به‌عنوان مثال، #BirthDate=#7/2/93)

برای جستجو براساس بخشی از یک رشته، می‌توان از کلید واژهٔ Like موجود در SQL استفاده کرد. به‌خاطر داشته باشید که باید تک کوتیشن در شروع و پایان رشته قرار دهید. اگر عملگر مقایسه Like باشد می‌توانید به عنوان جایگزین یک یا چند نویسه از علامت ٪ و جایگزین یک کاراکتر از علامت Dash استفاده کنید. کد زیر چگونگی به‌کارگیری این عملگر را نشان می‌دهد.

adostates.Recordest. Find "state LIKE 'M٪"

نکته : متد Find می‌تواند کند باشد، مگر این‌که تعداد کل رکوردها در مجموعه رکورد را محدود کنید. روش دیگر برای پیدا کردن رکوردها، استفاده از عبارت SQL هنگام مقداردهی مشخصهٔ RecordSource است. همچنین می‌توانید از مشخصهٔ Filter در کنترل ADO برای محدود کردن تعداد رکوردهای مجموعه رکورد استفاده کنید.

سه پارامتر اختیاری برای متد Find شیء Recordset وجود دارد :

- **SkipRows:** این یک نوع دادهٔ Long است که مقدار پیش‌فرض آن صفر است. از این پارامتر برای تعیین فاصله از سطر جاری یا نشانهٔ آغازین برای شروع جستجو استفاده می‌شود.
- **SearchDirection:** این پارامتر تعیین می کند جستجو باید از سطر جاری تا آخرین رکورد یا سطر جاری تا اولین رکورد انجام شود. جستجو در شروع یا پایان مجموعه رکورد خاتمه می‌یابد که بستگی به مقدار SearchDirection دارد. این پارامتر می‌تواند یکی از مقادیر عددی زیر باشد :

ـ adsearch Forward(0) ـ جستجو به سمت جلو از رکورد جاری در صورتی که جستجو با موفقیت انجام نشود، EOF برمی‌گرداند.

ـ adsearch Backward(1) ـ جستجو به سمت عقب از رکورد جاری در صورتی که جستجو با موفقیت انجام نشود، BOF برمی‌گرداند.

• **Start:** این پارامتر، یک نشانهٔ Variant است که به عنوان محل شروع جستجو مورد استفاده قرار می‌گیرد. مقدار این پارامتر می‌تواند یکی از مقادیر عددی زیر باشد :

ـ adBookmarkCurrent (0) ـ رکورد جاری

ـ adBookmark First (1) ـ اولین رکورد

ـ adBook mark Last (2) ـ آخرین رکورد

هنگامی که رکوردها را با استفاده از متد Find جستجو می‌کنید، می‌توانید مشخصه‌های EOF و BOF از شیء Recordset را به‌کار بگیرید تا تعیین کنید که آیا رکورد خاصی پیدا شده است یا نه؟ اگر جستجو ناموفق باشد، باید به رکوردی که جستجو شروع شده است، برگردید. کد زیر چگونگی استفاده از این مشخصه‌ها را نشان می‌دهد :

```
'If the record isn't found
If .EOF or .BOF Then
        'Return to the starting record
        .Bookmark=varBookmark
        MsgBox "Record not found."
End If
```

مشخصه‌های BOF، EOF و Bookmark از مشخصه‌های Recordset می‌باشند.

لذا این قطعه کد باید داخل بلوک

```
With ADO Data نام کنترل .Recordset
.
.
.
End With
```

قرار گیرد.

مشخصهٔ Bookmark برای علامت‌گذاری رکورد معینی در شیء Recordset استفاده می‌شود.

برای این منظور با استفاده از متد Move روی رکورد رفته از دستور زیر استفاده می‌کنیم؛

```
Dim VarBookmark As Variant
```

VarBookmark = ADodc نام کنترل . Recordset.Bookmark

و هرگاه بخواهیم به رکورد علامت‌گذاری شده برگردیم از دستور زیر استفاده می‌کنیم؛

ADodc نام کنترل . Recordset.Bookmark= VarBook Mark

مثال زیر، متد Find را با استفاده از هر چهار پارامتر پیاده سازی می‌کند :

```
Dim VarBookmark As Variant
With adoFood.Recordset
    'mark the current record
    varBookmark=.Bookmark
    'specify the search criteria, start and direction
    .Find "CategoryName='Condiments'",0, adsearchForward,_
adBookmarkCurrent
'If the record isn't found
If .EoF or .BOF then
    'Return to the starting record
    .Bookmark=varBookmark
    MsgBox "Record not found."
  End If
End with
```

تمرین: جستجوی رکوردها در یک Recordset : در این تمرین، پروژه‌ای که در دو تمرین قبلی این فصل ایجاد کردید را ادامه خواهیم داد. عملیاتی را به برنامهٔ کاربردی اضافه خواهید کرد تا اسامی شرکت‌ها را جستجو کند.

۱ــ پروژهٔ prjADO.vbp که در تمرین قبل ذخیره کردید را باز کنید.

۲ــ بین دو کنترل Text2 و Adodc1 یک دکمهٔ فرمان اضافه کنید.

۳ــ درپنجرهٔ Properties، مشخصهٔ Name کنترل Command1 را به CmdSearch تغییر دهید.

۴ــ مشخصهٔ Caption کنترل CmdSearch را به search& تغییر دهید.

۵ــ در رویداد Click دکمهٔ فرمان Cmdsearch، کد زیر را تایپ کنید.

```
Adodc1.Recordset.MoveFirst
Adodc1.Recordset.Find "CompanyName= 'FrankenVersand' "
If Adodc1.Recordset.EOF Then MsgBox "Record not found."
```

۶ــ از منوی Run گزینهٔ Start را انتخاب کنید.

۷ــ روی دکمهٔ Search کلیک کنید.

توجه کنید که رکورد حاوی نام شرکت Frankenversand رکورد جاری خواهد شد. برنامهٔ کاربردی شبیه شکل ۱۸ــ۴ خواهد بود.

شکل ۱۸ـ۴ـ استفاده از متد Find برای جستجو

۸ــ روی End کلیک کنید.

۹ــ پروژه را ذخیره کنید.

تمرین: روی خطی که متد Find قرار دارد Break Point گذاشته و مشخصه‌های کنترل ADodc1 را با کمک پنجره Watch مشاهده کنید. در فرم‌های طراحی شدهٔ مثال‌های قبل امکان نمایش تنها یک رکورد وجود داشت. حال می‌خواهیم از کنترل‌هایی استفاده کنیم که به ما امکان نمایش چندین رکورد را به‌طور همزمان می‌دهد.

جدول ۲ــ۴ــ متدهای شیء Recordset

متد	شرح
AddNew	اضافه کردن رکورد جدید به Recordset
CancelUpdate	انصراف از تغییرات انجام شده که در حافظه ذخیره شده است.
Close	بستن Recordset
Delete	حذف کردن رکورد از Recordset
Move	جابه‌جایی در Recordset
MoveFirst	جابه‌جایی به اولین رکورد Recordset
MoveLast	جابه‌جایی به آخرین رکورد Recordset
MoveNext	جابه‌جایی به رکورد بعدی در Recordset
MovePrevious	جابه‌جایی به رکورد قبلی در Recordset
Update	به روزکردن تغییرات انجام شده در Recordset
find	به‌منظور پیدا کردن رکوردهایی با ویژگی‌های معین در Recordset (جستجودر مجموعه رکوردها)

جدول ۳ــ۴ــ مشخصه‌های شیء Recordset

مشخصه	شرح
BOF	نشان می‌دهد که رکورد جاری قبل از اولین رکورد جدول می‌باشد (اشاره گر رکورد از جدول خارج شده است و در بالای جدول قرار دارد.)
Bookmark	این مشخصه برای علامت گذاری یک رکورد معین استفاده می‌شود.
EOF	نشان می‌دهد که رکورد جاری بعد از آخرین رکورد جدول می‌باشد (اشاره گر رکورد از جدول خارج شده است و در پایین جدول قرار دارد.)
RecordCount	از نوع long است و تعداد رکوردهای موجود در Recordset را نشان می‌دهد.
Source	منبع داده در Recordset را نشان می‌دهد که می‌تواند عبارت SQLیا نام جدول باشد.

برای علامت گذاری رکورد ،به کمک متدهای move روی رکورد رفته و از دستور زیر استفاده می کنیم؛

```
Dim varBookmark as variant
varBookmark =adodc نام کنترل. Recordset. bookmark
```

و هرگاه بخواهیم به رکورد علامت گذاری شده برگردیم از دستور زیر استفاده می کنیم؛

```
adodc نام کنترل. Recordset. bookmark =varBookmark
```

مثال ۲ـ۴

در این مثال فرمی مانند شکل ۱۹ــ۴ ایجاد کنید .

شکل ۱۹ــ۴ــ فرم نمایش رکوردهای جدول Customers

ابتدا نام کنترل ado را adoCustomers قرار داده و آن را به بانک NWind.MDB متصل کرده، نوع فرمان را در بخش Command Type مساوی adCmdText قرار داده و در بخش command Text(SQL) دستور زیر را وارد کنید.

```
Select * from customers
```

این دستور سبب می‌شود که تمام فیلدهای جدول customers از طریق کنترل adoCustomers قابل دسترسی باشند.

نام کنترل‌های TextBox را با توجه به فیلدی که نمایش می‌دهند و نام دکمه‌های فرمان را با توجه به عملیاتی که انجام می‌دهند، تغییر دهید. (txtCustomerId, txtCompanyName, txtPhone, cmdAdd, cmdUpdate,cmdDelete , cmdRefresh, cmdSort, cmdSearch,txtCity)

اضافه کردن رکورد به جدول :

برای اضافه کردن رکورد به جدول، قطعه کد زیر را در رویداد کلیک دکمه فرمان ADD اضافه کنید.

```
Private Sub CmdAdd _Click()
        Ado Customers . Recordset . AddNew
End Sub
```

پس از اجرای فرمان فوق در انتهای جدول یک سطر خالی ایجاد می‌شود تا بتوان رکورد جدیدی از اطلاعات را به انتهای جدول اضافه کرد.

ثبت اطلاعات در جدول :

بعد از اجرای فرمان فوق برای ثبت اطلاعات وارد شده و یا برای ثبت تغییرات در برنامه باید فرمان زیر را اجرا کنیم. قطعه کد زیر را در بخش کد دکمه فرمان Update وارد نمایید.

```
Private Sub cmdUpdate_Click()
        AdoCustomers . Recordset . Update
End Sub
```

حذف اطلاعات در جدول :

برای حذف اطلاعات از جدول باید از فرمان Delete استفاده کنیم. قطعه کد زیر را در بخش کد دکمه فرمان Delete وارد نمایید.

```
Private Sub cmdDelete _Click()
        AdoCustomers . Recordset . Delete
End Sub
```

لغو عملیات‌های روی جدول :

اگر بخواهیم عملیات انجام شده روی جدول را لغو کنیم تا از ثبت تغییرات جلوگیری شود،باید Refresh از فرمان استفاده کنیم. قطعه کد زیر را در بخش کد دکمه فرمان Refresh وارد کنید. یکی دیگر از کاربردهای این دستور برای فراخوانی مجدد اطلاعات از بانک اطلاعاتی می‌باشد.

```
Private Sub cmd_Refresh _Click()
        AdoCustomers . Refresh
End Sub
```

استفاده از دستورات SQL :

برای استفاده از دستورات SQL در VB باید کدهای مورد نظر را در بخش RecordSource کنترل ADO وارد و سپس فرمان Refresh را اجرا کرد.

می‌خواهیم در رویداد کلیک دکمهٔ Sort اطلاعات جدول براساس فیلد CompanyName مرتب شود. کد زیر را در رویداد کلیک این دکمه بنویسید.

```
Private Sub cmdSort_Click()
    AdoCustomers.RecordSource = "select * from customers order by
    CompanyName"
    AdoCustomers.Refresh
End Sub
```

می‌خواهیم رکوردهایی را جستجو کنیم که فیلد City آن‌ها مساوی محتوای txtCity باشد لذا در رویداد کلیک دکمهٔ Search کد زیر را بنویسید.

```
Private Sub cmdSearch_Click()
    AdoCustomers.RecordSource = "SELECT * FROM customers
    WHERE City=" '–& txtCity.Text &" '
    AdoCustomers.Refresh
End Sub
```

اگر بخواهید جستجو براساس نام کمپانی و نام شهر انجام شود در رویداد کلیک دکمهٔ Search و کد زیر را بنویسید.

```
Private Sub cmdSearch_Click()
AdoCustomers.RecordSource = "SELECT * FROM customers WHERE
    CompanyName ='" & txtCompanyName.Text & "" & "AND City
    ='" & txtCity.Text & "' "
AdoCustomers.Refresh
End Sub
```

توجه کنید که مقادیر رشته‌ای و با تک کوتیشن مشخص می‌شوند.

۹ـ۴ـ کاربرد کنترل DataGrid

اولین کنترلی که به‌کار خواهیم برد، کنترل DataGrid است که امکان مشاهده و ویرایش چندین سطر از داده‌ها را به‌طور هم‌زمان فراهم می‌کند. هم‌چنین DataGrid برای ورود سریع مقادیر زیادی از داده‌ها مفید است. برای استفاده از این کنترل، باید آن را به جعبه ابزار ویژوال بیسیک اضافه کنید. مراحل زیر، چگونگی اضافه کردن این کنترل را نشان می‌دهند :

۱ــ از منوی Project گزینهٔ Components را انتخاب کنید.

۲ــ Microsoft DataGrid Control 6.0 را از لیست انتخاب کرده و سپس روی OK کلیک کنید. کنترل DataGrid در جعبه ابزار نشان داده خواهد شد.

اکنون می‌توانید DataGrid را به فرم اضافه کنید. روی آیکون DataGrid در جعبه ابزار دابل کلیک کنید. کنترل را بزرگ کنید تا کل فرم را دربر بگیرد.

در صورتی که برنامه را اجرا کنید، DataGrid به طور کامل خالی خواهد بود زیرا هنوز آن را به کنترل ADO Data پیوند نکرده‌اید.

مثال ۴ـ۳

پیوند Data Grid به کنترل ADO Data :

۱ــ روی کنترل DataGrid در روی فرم کلیک کنید. مشخصهٔ Name را با dgrdCustomers مقداردهی کنید، زیرا اطلاعات مشتری را نمایش خواهد داد.

۲ــ adoCustomers را از لیست بازشوی مشخصهٔ DataSource انتخاب کنید (این همان کنترل ADO Data است که قبلاً ایجاد شده است).

بعد از تعیین DataSource برای DataGrid، کنترل DataGrid به‌طور خودکار تعداد سطرها و ستون‌ها را پیکربندی می‌کند. برای مشاهدهٔ فرم در عمل، از منوی Run گزینهٔ Start را انتخاب کنید. پنجره، داده‌های جدول Customers را نشان خواهد داد (شکل ۲۰ـ۴).

به‌دلیل این‌که تنظیمات پیش‌فرض همیشه بهترین نیستند، DataGrid به‌طور کامل قابل پیکربندی بوده و امکان نمایش ستون‌های خاص، قالب‌بندی و رنگ هر ستون را فراهم می‌کند. هم‌چنین می‌توان DataGrid را به چندین بخش تقسیم کرد که هرکدام به‌طور مستقل از دیگری قابل مشاهده و مرور باشند. انجام این کار در Excel و سایر صفحات گسترده متداول است.

شکل ۲۰ــ۴ــ DataGrid می‌تواند به‌طور خودکار خودش را پیکربندی کند تا داده‌های منبع تعیین شده را نمایش دهد.

۱ـ۹ـ۴ـ انتخاب ستون‌ها :

به دلیل اینکه جدول‌های بانک اطلاعاتی با گذشت زمان، بزرگ می‌شوند، همیشه نمی‌خواهید که تمام ستون‌های جدول را به کاربران نشان دهید. دراین قسمت، چگونگی تعیین ستون‌ها برای نمایش در کنترل DataGrid را نشان خواهیم داد. قبل از شروع، باید تصمیم بگیرید که کدام ستون‌ها در DataGrid نمایش داده شوند. برای این مثال، DataGrid ستون‌های Company Name، Contact Name، Phone Number و Country را نشان خواهد داد.

مثال ۴ـ۴

تعیین ستون‌ها در DataGrid:

۱ــ روی DataGrid کلیک راست کرده و از منوی میان‌بر، Retrieve Fields را انتخاب کنید.

هنگامی که ویژوال بیسیک دربارهٔ جایگزینی طرح کلی سؤال می‌کند، Yes را انتخاب کنید. تمام فیلدها از Recordsource انتخاب شده (در این حالت، جدول Customers) در کنترل DataGrid بارگذاری خواهند شد (شکل ۲۱ــ۴).

شکل ۲۱ــ۴ــ کنترل DataGrid تمام فیلدها را از جدول بارگذاری می‌کند بنابراین می‌توان یکی از آن‌ها را انتخاب کرده و مشاهده کرد.

۲ــ روی DataGrid کلیک راست کرده و گزینهٔ Edit را از منوی میان‌بر انتخاب کنید. اکنون می‌توانید روی ستون‌های خاصی کلیک کرده و آن‌ها را ویرایش کنید.

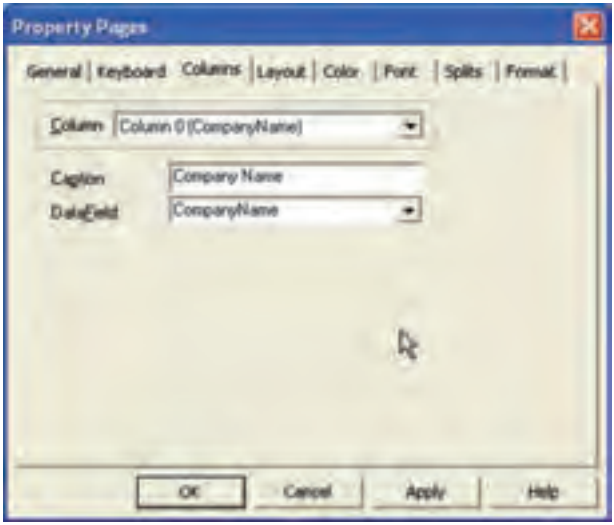
- برای حذف ستون‌ها از لیست، روی فیلد CustomerID کلیک راست کرده و Delete را انتخاب کنید (شکل ۲۲ــ۴).

شکل ۲۲ــ۴ــ هنگامی که DataGrid در حالت Edit است، می‌توان ستون‌ها را اضافه و حذف کرد.

۳ــ ستون‌های Contact Title، Address، City، Region، PostalCode و Fax را حذف کنید.

- در این کنترل نمی‌توان ستون‌ها را با استفاده از کشیدن و رها کردن دوباره مرتب کرد. به عنوان مثال، اگر می‌خواهید ستون‌های ContactName و CompanyName را جابه‌جا کنید، ابتدا باید ستون ContactName را Cut کرده و سپس روی CompanyName کلیک راست کرده و Paste را انتخاب کنید. DataGrid تمام تنظیمات مربوط به یک ستون را نگه خواهد داشت.

۴ــ هنگام نمایش داده‌ها در DataGrid، ضروری نیست که عنوان ستون‌ها همان نام فیلدها باشد. برای تغییر عنوان ستون‌ها، روی ستون کلیک راست کرده و Properties را انتخاب کنید. روی صفحه Columns نام جدیدی را در کادر متن Caption وارد کنید (شکل ۲۳ــ۴).



شکل ۲۳ــ۴ــ مشخصهٔ Caption صفحهٔ Columns کنترل می کند که چه چیزی در عنوان هر ستون نمایش داده خواهد شد.

۵ــ مرحلهٔ ۶ را برای تغییر عنوان هر ستون موردنظر، تکرار کنید.

۲ـ ۹ـ۴ـ پر کردن فـرم : مـی‌تـوان مشخصهٔ Visible کنترل ADO Data را بـا False مقدار دهی کرد تا کنترل ADOData برای کاربران قابل مشاهده نباشد. هم‌چنین می‌توان DataGrid را به اندازه‌ای بزرگ کرد که فرم را پر کند. ابتدا مشخصهٔ Align را با vbAlignTop ــ1 مقداردهی کنید تا DataGrid در بالای فرم قرارگرفته و از چپ به راست فرم را پر کند. سپس کد زیر را اضافه کنید تا از بالا به پایین فرم را پر کند :

```
Private Sub From_Resize()
    dgrdCustomers.Height=Me.scaleHeight
End sub
```

شکل ۲۴ــ۴ــ DataGrid کامل شده است.

به عنوان یک کار پایانی، عنوانی مثل Customer Viewer را در مشخصهٔ Caption فرم قرار دهید تا فرم حرفه‌ای‌تر به نظر برسد.

۳ـ۹ـ۴ـ پیکربندی سایر مشخصه‌های DataGrid : می‌توان چندین مشخصهٔ دیگر را روی DataGrid پیکربندی کرد که آن‌ها نیز به سادگی می‌توانند مورد استفاده قرار گیرند. جدول ۵ــ۴ هدف هر مشخصه را شرح می‌دهد.

جدول ۵ــ۴ــ مشخصه‌های DataGrid

مشخصه	شرح
Align	تعیین می‌کند که DataGrid نسبت به بالا، چپ، راست یا پایین تنظیم شود.
AllowAddNew	اگر با True مقداردهی شود، رکوردهای جدید را می‌توان در پایین DataGrid اضافه کرد.
AllowArrows	اگر با True مقداردهی شود، با استفاده ازکلیدهای جهت‌دار می‌توان پیمایش را انجام داد.
AllowDelete	اگر با True مقداردهی شود، با فشار دادن کلید Delete می‌توان سطرها را از DataGrid حذف کرد.
AllowUpdate	اگر با True مقداردهی شود، می‌توان سطرها را ویرایش کرد و تغییرات به‌طور خودکار ذخیره می‌شوند.
CausesValidation	اگر با True مقداردهی شود، رویداد Validate هنگامی که فوکوس به کنترل دیگری منتقل می‌شود، فعال خواهد شد. .
ColumnHeaders	عنوان ستون را فعال یا غیرفعال می‌کند.
HeadLines	تعداد خطوط عمودی برای عنوان‌های ستون را تعیین می‌کند.
RowDivider Style	گزینه‌های مختلفی را برای قالب‌بندی جدا کننده‌های سطرها ارایه می‌کند.
TabAcrossSplits	کلید Tab را برای انتقال بین بخش‌های DataGrid فعال می‌کند.
TabAction	پاسخ به فشار دادن کلید Tab را تعیین می‌کند.
WrapcellPointer	تعیین می‌کند که آیا مکان‌نما باید هنگام رسیدن به انتهای یک سطر به سطر بعدی و اولین ستون منتقل شود؟

نکته: در صورتی که رکورد با جدول‌های دیگر مرتبط است، حذف رکورد ممکن نخواهد بود و سبب بروز پیام خطا می‌شود. حذف کردن فقط هنگامی ممکن است که تمام رکوردهای مرتبط حذف شده باشند. (Refrential Integrity)

نکته: هنگامی که یک برنامه کاربردی را طراحی و دربارهٔ نمایش داده‌های بانک اطلاعاتی آن فکر می‌کنید، باید بدانید که اگر داده‌ها را در یک جدول ذخیره کنید می‌توانید آن‌ها را در DataGrid ویرایش کنید. در صورتی که نیاز به قراردادن داده‌ها در بیش‌از یک جدول دارید، باید نوع دیگری از فرم را به‌کار ببرید. به عنوان مثال، یک سفارش هم اطلاعاتی دربارهٔ سفارش و هم کالاهای سفارش شده خواهد داشت. سفارش را نمی‌توانید در یک grid ویرایش کنید. زیرا داده‌ها حداقل در دو جدول متفاوت ذخیره شده‌اند. با این وجود، DataGrid می‌تواند لیستی از کالاها را نشان دهد. در صورتی که فقط می‌خواهید داده‌های سایر جدول‌ها را مشاهده کنید، DataGrid هنوز هم گزینهٔ مناسبی است.

۱۰-۴- کاربرد کنترل DataList

کنترل DataList خیلی ساده‌تر از کنترل DataGrid است. از این کنترل برای نمایش لیست یک ستونی از داده‌های جدول استفاده می‌شود. به دلیل این‌که این کنترل مقید به داده‌هاست، هیچ محدودیت حافظه برای نمایش تعداد عناصر ندارد. ولی اضافه کردن عناصر خیلی زیاد سبب می شود که پیدا کردن عنصر خاصی برای کاربران مشکل باشد. شبیه DataGrid کنترل DataList نیز از کنترل ADO Data استفاده می کند.

مثال۵-۴

ایجاد و استفاده از کنترل DataList :

۱ــ روی فرم جدید، یک کنترل ADO Data اضافه کنید که به جدول Customers از بانک اطلاعاتی Nwind.mdb دسترسی داشته باشد.

۲ــ کنترل DataList را از طریق کادر محاوره‌ای Components به جعبه ابزار اضافه کنید (در

کادر محاوره‌ای، گزینهٔ Microsoft DataList Control 6.0 را انتخاب کنید).

۳ــ در جعبه ابزار روی کنترل DataList دابل کلیک کنید تا به فرم اضافه شود.

۴ــ نام کنترل DataList را dlstCustomers قرار دهید.

۵ــ مشخصهٔ RowSource را با adoCustomers مقداردهی کنید. این مشخصه منبع داده‌هایی که لیستی از داده‌ها را برای نمایش ارایه خواهد کرد، مشخص می‌کند. برای تعیین فیلد موردنظر، مشخصهٔ ListField را با CompanyName مقداردهی کنید.

۶ــ فرم را ذخیره کرده و پروژه را اجرا کنید (شکل ۲۵ــ۴).

شکل ۲۵ــ۴ــ کنترل DataList ستونی از داده‌های منبع داده را نشان می‌دهد.

هم‌چنین می‌توان روش پرکردن فرم که در قسمت قبلی آموختید را برای این فرم نیز به‌کار ببرید. برای اینکه DataList به‌طور خودکار کل فرم را پر کند، کد زیررا به فرم اضافه کنید

```
Private sub Form_Resize( )
        dlstCustomers.Height = Me.scaleHeight
                        ارتفاع کنترل را مساوی ارتفاع فرم قرار می‌دهیم'
        dlstCustomers.Width = Me.scaleWidth
                        پهنای کنترل را مساوی پهنای فرم قرار می‌دهیم'
End sub
```

۱ـ۱۰ـ۴ـ پیکربندی مشخصه‌های کنترل DataList : همان‌طور که بیان شد، کنترل DataList دارای مشخصه‌های کمتری نسبت به DataGrid است ولی چند ویژگی دارد که کنترل DataGrid ندارد (جدول۶ـ۴).

جدول ۶ـ۴ـ برخی از مشخصه‌های کنترل DataList

مشخصه	شرح
CausesValidation	در صورتی که با True مقداردهی شود، رویداد Validate پس از انتقال فوکوس به کنترل دیگری، فعال می‌شود.
IntegralHeight	تعیین می‌کند که آیا کنترل DataList بخش‌هایی از عناصر را در صورتی که به اندازهٔ کافی بزرگ نباشد، نشان دهد یا نه؟
MatchEntry	همزمان باتایپ کاربر، DataList براساس نویسهٔ وارد شده به عنصر موردنظر رجوع می‌کند.

۱۱ـ۴ـ کاربرد کنترل DataCombo

کنترل DataCombo دقیقاً شبیه کنترل DataList است و فقط ظاهر آن‌ها متفاوت است. این کنترل به جای این‌که فضای عمودی زیادی را اشغال کند، در صورت نیاز بخش لیست را باز می‌کند. این کار هنگامی مفید است که نیاز به فرم کوچک داشته باشیم. کنترل DataCombo شبیه کنترل ComboBox استاندارد است و برای تعداد کم (کمتر از ۱۰۰) مورد استفاده قرار خواهد گرفت.

مثال۶ـ۴

ایجاد و استفاده از کنترل DataCombo :

۱ـ روی فرم جدید، کنترل ADO Data جدیدی را اضافه کنید که به جدول Customers از بانک اطلاعاتی Nwind.mdb دسترسی داشته باشد. به دلیل این‌که کنترل‌های DataList و DataCombo با هم هستند، پس کنترل DataCombo روی جعبه ابزار وجود خواهد داشت.

۲ـ روی کنترل DataCombo دابل کلیک کنید. نام آن را به دلیل این‌که برای نگهداری اسامی مشتریان مورد استفاده قرار می‌گیرد، dcboCustomers قرار دهید.

۳ـ مشخصهٔ Rowsource کنترل DataCombo را با adoCustomers و مشخصهٔ ListField را با CompanyName مقداردهی کنید (شبیه کنترل DataList عمل کنید).

هنگامی که برنامه را اجرا کنید، لیست مشتریان که در کنترل DataCombo لیست شده‌اند را مشاهده خواهید کرد (شکل ۲۶ــ۴).

شکل ۲۶ــ۴ــ کنترل DataCombo محتوای فیلد CompanyName را از جدول customers بارگذاری می‌کند.

مشخصه‌های قابل پیکربندی برای کنترل DataCombo همان مشخصه‌های DataList هستند. تنها تفاوت مشخصهٔ Style است که شبیه مشخصهٔ Style کنترل ComboBox عمل می‌کند. DataCombo می‌تواند فقط خواندنی یا قابل ویرایش بوده و به کاربران امکان اضافه کردن رکوردی به لیست موجود را فراهم کند. به دلیل این‌که DataCombo اغلب برای انتخاب مقادیر از لیست، مورد استفاده قـرار مـی‌گیرد، معمولاً مقدار مشخصهٔ Style آن را 2–DropDown List انتخاب می‌کنیم.

خودآزمایی

۱ــ سه رابط دسترسی به داده‌ها در ویژوال بیسیک، کدامند؟

۲ــ دلیل استفاده از ADO را توضیح دهید.

۳ــ از SQL به چه منظوری استفاده می‌شود؟

۴ــ در SQL از چه عملگری برای جستجوی مقادیر براساس یک الگو استفاده می‌شود؟

۵ــ دو مشخصه‌ای که برای اتصال کنترل ADO Data به بانک اطلاعاتی باید مقداردهی شوند، کدامند؟

۶ــ دو روش برای به هنگام‌کردن رکوردها را شرح دهید.

۷ــ پروژه‌ای طراحی کنیدکه از طریق آن بتوانید مشخصات CDهای موجود در هنرستان را در آن ذخیره کنید.

توضیح : ابتدا بانک اطلاعاتی مربوطه را به کمک مربی خودتان طراحی و پیاده‌سازی کنید و سپس به کمک کنترل ADO Data رابط کاربری مناسبی برای انجام عملیات درج، حذف، ویرایش و جستجوی رکوردها در ویژوال بیسیک ایجاد کنید.

فصل ۵

کاربرد نرم‌افزار Crystal Report

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- تنظیمات اولیه در ویژوال بیسیک برای برقراری ارتباط با کریستال ریپورت انجام دهد.
- با استفاده از کریستال ریپورت با بانک اطلاعاتی در برنامهٔ اکسس ارتباط برقرار کند.
- گزارش‌های سفارشی در محیط کریستال ریپورت را انجام دهد.
- گزارش‌های سفارشی را در ویژوال بیسیک انجام دهد.
- گزارش‌های ایجاد شده را چاپ کند.

هر نرم افزاری که داده‌ها را ذخیره می‌کند نیاز به بازیابی این داده‌ها و ایجاد گزارشات مناسب است. گزارش را می‌توان به عنوان پاسخی به پرسش کاربر از داده‌های خود در نظر گرفت.

نرم‌افزار Crystal Report یکی از ابزارهای قدرتمند تهیه گزارش است و با این نرم‌افزار می‌توان گزارشات پیشرفته و پیچیده‌ای را از بانک‌های اطلاعاتی Access ـ SQL ـ Olap ـ Oracle ـ Server ـ Excel و ... تهیه نمود. طراحی خروجی گزارشات می‌تواند به راحتی به‌وسیله Crystal Report صورت گیرد به‌طوری که نظر و علاقه هر کاربری را تأمین نماید.

این نرم‌افزار می‌تواند به‌طور مستقل و یا از طریق نرم‌افزار Visual Basic گزارش‌گیری را ایجاد، ویرایش و نمایش دهد.

۱ـ۵ـ مراحل ایجاد یک گزارش

برای اجرای نرم‌افزار از طریق مسیر زیر اقدام می‌شود.

Start | All Program | Crystal Reports9 Tools| Crystal Reports9

و برای اجرای نرم‌افزار از درون VisualBasic به صورت زیر اقدام می‌شود.

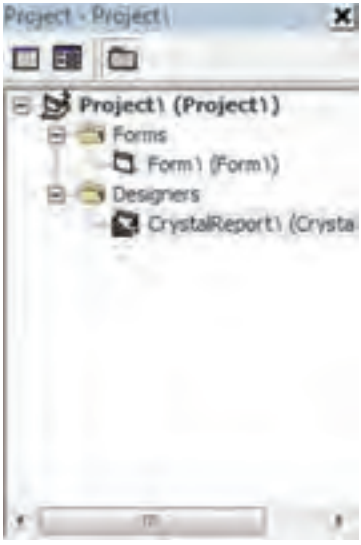
Project Menu | Add Crystal Report 9

نکته: در صورتی‌که بعد از نصب کریستال گزینه Add Crystal Report 9 مشاهده نشود از طریق

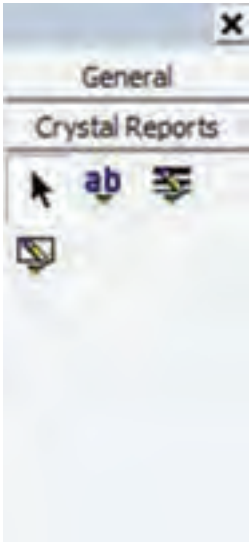
Project Menu | Components | Designers Tab

گزینه Crystal Report 9 را تیک‌دار نمائید.

که در این روش در کنار فرم‌های برنامه یک فرم گزارش‌گیری در زیرشاخه Designers برای Crystal Report (شکل ۱ــ۵) و در Tool Box، ابزار این نرم‌افزار نمایش داده می‌شود (شکل ۲ــ۵). این نوع فایل‌ها با پسوند rpt . ذخیره می‌شود.



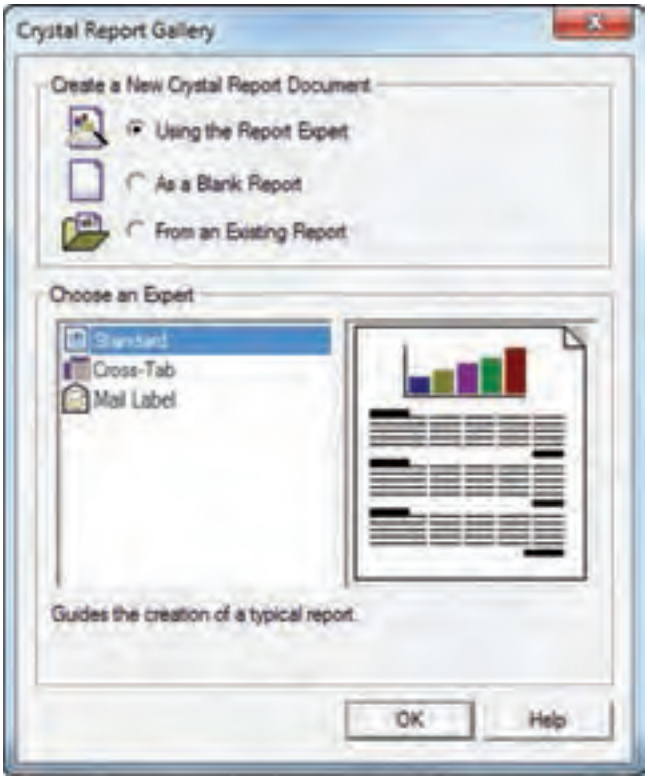
شکل ۱ــ۵



شکل ۲ــ۵

با انتخاب CrystalReport در Visual Basic مراحل ایجاد گزارش‌گیری آغاز می‌شود. اولین کادر محاوره‌ای Crystal Report Gallery می‌باشد که نوع سند گزارش‌گیری توسط کاربر تعیین می‌شود (شکل ۳ــ۵).

تولید گزارش‌گیری ماهر (ویزارد)	● Using the Report Expert
تولید یک گزارش خالی	● As a Blank Report
تولید گزارش از گزارش‌های موجود	● From an Existing Report



شکل۳ــ ۵

گزینه Using the Report Expert و Standard (متداول‌ترین نوع گزارش) را انتخاب نمایید.

۲ـ۵ـ انتخاب یک منبع داده و ارتباط با آن

گام بعدی در ایجاد گزارش، انتخاب داده‌هایی است که در گزارش به‌کار خواهید برد. کادر محاوره‌ای Standard Report Creation Wizard امکان انتخاب منبع داده را فراهم می‌نماید (شکل ۴ــ۵).

شکل۴ــ ۵

برای دسترسی به منبع داده گزینه Create New Connection و زیر گزینه OLE DB(ADO) را انتخاب و دابل کلیک نمایید (شکل ۵ــ۵).

توجه : از آنجا که هنرجو با مفهوم ADO و بانک اطلاعاتی Access آشنا می‌باشد گزینه OLE DB انتخاب می‌شود.

شکل۵ــ ۵

در کادر محاوره‌ای OLE DB(ADO) مراحل ارتباط با بانک انجام می‌گردد. نوع ارتباط به بانک با انتخاب Microsoft Jet 4.0 OLE DB Provider و Next ادامه می‌یابد(شکل ۶ــ۵).

شکل۶ــ۵

در این مرحله محل بانک اطلاعاتی (بانک Access با پسوندmdb.) را با کلیک بر روی ... تعیین نمایید و در صورتیکه بانک اطلاعاتی با نام کاربر و رمز عبور خاصی وجود دارد، آن‌ها را تعیین و در غیر اینصورت همان اطلاعات پیش فرض را تغییر نداده و Finish را کلیک کنید (شکل ۷ــ۵).

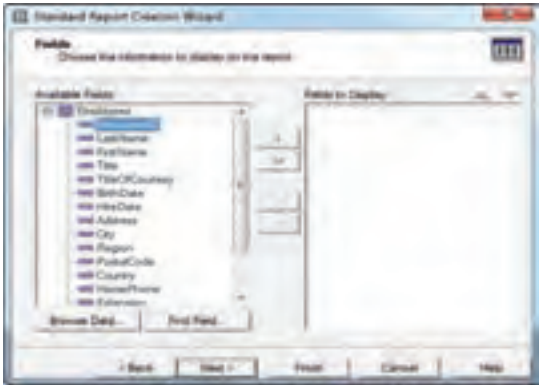
شکل۷ــ۵

۳ـ۵ـ اضافه کردن جدول و فیلدها

ارتباط به بانک برقرار شده است و شامل مسیر بانک و جدول (Tables) و Views می‌باشد. جدولی را از زیر شاخه Tables برای مشاهده داده‌های آن در گزارش انتخاب و دابل کلیک نمایید (شکل ۸ــ۵).

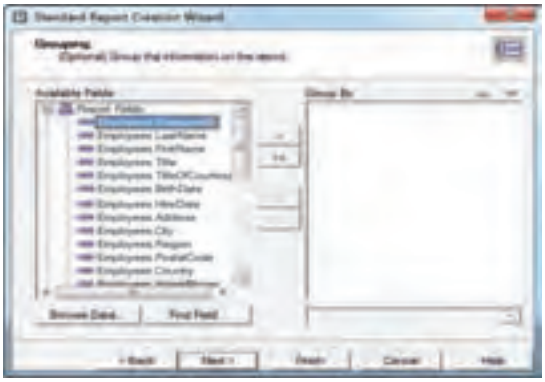
شکل۸ــ ۵

در این مرحله فیلدهایی که می‌خواهید در گزارش مشاهده شود را انتخاب و به لیست سمت راست اضافه نمایید و کلید Next را فشاردهید (شکل ۹ــ۵).



شکل ۹ــ ۵

در مرحله دسته‌بندی (Grouping) و مرتب‌سازی اطلاعات، می‌توان بدون انتخاب گزینه‌ای به‌مرحله بعد (Next) رفت (شکل ۱۰ــ۵).

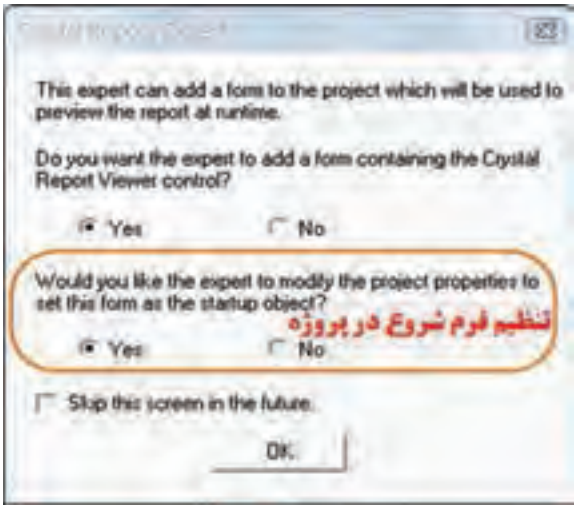


شکل ۱۰ــ ۵

مشاهده گزارش: در مرحله آخر مدل تولید گزارش (Style) بر اساس فیلدهای انتخابی مشخص می‌شود و با انتخاب Finish مراحل تولید گزارش به پایان می‌رسد (شکل ۱۱ــ۵).

شکل۱۱ــ۵

برای اتمام کار شکل ۱۲ــ۵ نمایش داده می‌شود و اعلام می‌کند، فرمی برای خروجی گزارش به‌پروژه اضافه می‌شود.



شکل۱۲ــ۵

Do you want the expert to add a form containing the Crystal Report Viewer Control?

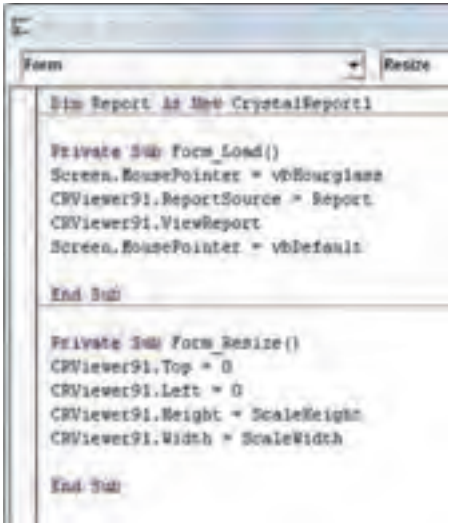
در صورت انتخاب Yes : به‌صورت اتوماتیک یک فرم در کنار فرم‌های دیگر در محیط برنامه‌نویسی Visual Basic طراحی شده و پیش نمایش (Preview) گزارش مشاهده می‌شود و کدهای ارتباط با گزارش کریستال تولید می‌شود (شکل۱۳ــ۵).

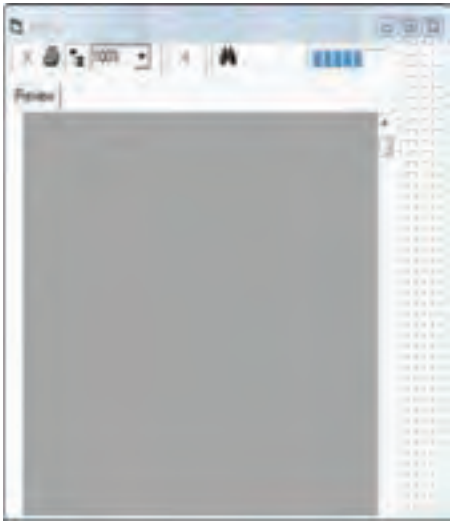
Would you like the expert to modify the project properties to set this form as the startup object?

در صورت انتخاب Yes : به‌صورت اتوماتیک فرم شروع پروژه در Visual Basic، همان فرمی خواهد شد که برای نمایش گزارش در گزینه قبل تولید شده است.

یادآوری: تنظیم فرم شروع پروژه

Project | Project Properties| گزینه Start up objectمنوی





شکل۱۳ــ۵

۴ـ۵ـ مد طراحی

در صورتی‌که بخواهید ظاهر صفحه و فیلدهای گزارش را تغییر دهید می‌توانید از این قسمت استفاده نموده و تغییرات لازم را اعمال نمایید. مانند تغییر سایز و نوع قلم، محل نمایش و چیدمان فیلد، حذف فیلد (انتخاب نام فیلد از Detail و فشردن کلید Delete) و اضافه کردن فیلد (انتخاب و درک کردن فیلد به Detail) و تغییر عنوان فیلد (انتخاب عنوان فیلد از Page Header و کلیک راست، گزینه Edit Text) و...

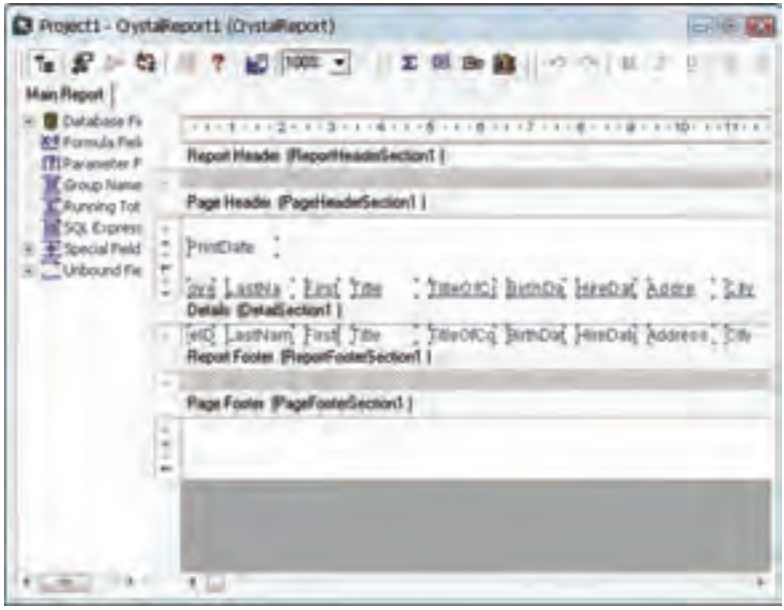
برای تغییرات طراحی گزارش بر روی فایل CrystalRepoet1 کلیک نماید (شکل۱۴ـ۵).

- برای مشاهده پیش نمایش می‌توانید از کلید F5 استفاده نمایید. تا وضعیت چیدمان فیلدها، عنوان‌ها و تعداد صفحه‌های خروجی را نمایش دهد.

توجه: صفحه طراحی گزارش دارای پنج بخش اصلی می‌باشد :

Report Header, Report Footer, Details, Page Header, Page Footer

که در درس بانک اطلاعاتی به‌طور کامل تشریح شده است.



شکل۱۴ـ۵

توسط ابزار موجود در Toolbox می‌توانید اشیا، متن، خط، شکل و... را بر روی گزارش خود درج نمایید تا ظاهری زیباتر و کارآمدتر داشته باشد (شکل ۱۵ــ۵).

۱ـ۴ـ۵ـ پنجره Field Explorer :

توسط این پنجره جدول و فیلدهای انتخابی آن و فیلدهای ویژه (Special Fields)مشاهده می‌شود. که با درگ کردن هر کدام از فیلدهای مورد نظر بر روی صفحه طراحی گزارش می‌توان از آن‌ها استفاده نمود.

شکل۱۵ــ۵

نکته : فیلدهای ویژه : در برنامه Crystal Report فیلدهایی از پیش تعریف شده وجود دارد که در هنگام طراحی گزارش امکانات ویژه‌ای را برای کاربر فراهم می‌نماید. مانند : درج خودکار تاریخ و ساعت، شماره صفحه، شماره رکورد و...

در صورتی که در کنار فیلدی تیک سبز باشد یعنی از آن در طراحی گزارش استفاده شده است.

شکل۱۶ــ۵

۲ـ۴ـ۵ـ گزینه Format Field: یکی از قوی‌ترین امکاناتی که در این نرم‌افزار در اختیار برنامه‌نویس قرار می‌دهد، نوشتن کدهای Script در داخل هر شیء می‌باشد. لازم به ذکر است که همهٔ دستورات شرطی در زبان VB در این بخش از برنامه قابل اجرا است.

برای مشاهده این امکانات فیلدی ازبخش Detail را انتخاب و کلیک راست نموده و گزینه Format Field را انتخاب نمایید (شکل ۱۷ــ۵).

شکل۱۷ــ۵

۵ـ۵ـ خصوصیات مشترک در قالب‌بندی انواع فیلدها

۱ـ۵ـ۵ـ زبانه Common

Suppress: در صورت فعال بودن این گزینه فیلد مورد نظر در زمان چاپ گزارش و نیز در صفحه نمایش اطلاعات دیده نخواهد شد. ولی در گزارش باقی می‌ماند و با غیر فعال نمودن این گزینه می‌توان آن را به حالت عادی خود باز گرداند.

Keep Object Together: برای کنار هم نگهداشتن اطلاعات مربوط به فیلدهای بزرگ در یک صفحه باید این گزینه را فعال نمود.

Can Grow: این گزینه را باید برای شیء که دارای چندین خط اطلاعات است فعال نمود تا از نمایش محتویات آن به صورت کامل مطمئن گردید. همچنین در صورت فعال بودن این گزینه

می‌توانید بیشترین تعداد خطی که یک شئ می‌تواند داشته باشد را نیز تعیین نمایید. به‌صورت پیش فرض این مقدار برابر صفر و به معنای بدون محدودیت می‌باشد.

Tool Tip Text : با استفاده از این گزینه می‌توان توضیح مختصری را به یک شیء اختصاص داد.

Text Rotation: با استفاده از این گزینه می‌توان شیء را با زاویه ۹۰ و یا ۲۷۰ درجه چرخش به نمایش گذارد.

Suppress If Duplicated : این گزینه زمانی مورد استفاده قرار می‌گیرد که قصد ممانعت از به نمایش درآمدن مقادیر تکراری برای یک فیلد را داشته باشید.

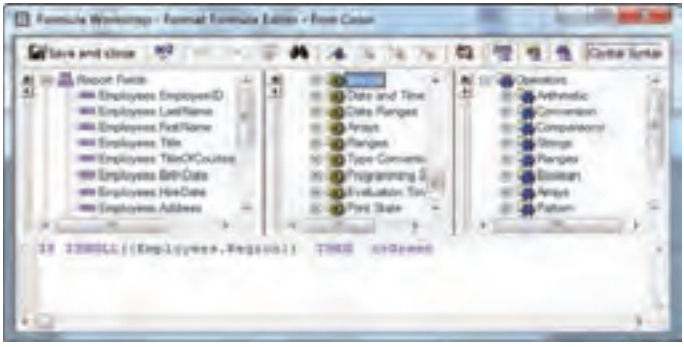
: با این دکمه می‌توان برای نمایش قالب فیلد شرایطی را تعیین نمود تا اطلاعات در هنگام گزارش به صورتی متمایز نمایش داده شود.

مثال ۱ـ۵

می‌خواهیم گزارشی از جدول Employees بانک اطلاعاتی Nwind.mdb تهیه کنیم که اگر فیلد ناحیه (Region) رکوردی خالی بود آن رکورد قرمز نمایش داده شود.

برای انجام این کار گزارشی که از جدول Employees تهیه شده و فیلدهای آن بر روی فرم گزارش‌گیری درج شده را باز نموده و تمام فیلدهای موجود در Detail را، با درگ کردن انتخاب نمایید.

کلیک راست کرده و گزینه Format Multiple Objects را انتخاب نمایید. سپس زبانه Font در قسمت Color نشانه انتخاب شود (شکل ۱۸ـ۵).



شکل۱۸ـ۵

پنجره Formula Workshop به چهار بخش تقسیم شده است : بخش اول اسامی فیلدها،

دوم توابع داخلی، بخش سوم عملوندهای ریاضی و رشته‌ای، بخش چهارم محلی برای نوشتن Script در بخش چهارم کد زیر را بنویسید :

```
IF ISNULL({Employees.Region}) THEN  crGreen
```

: برای صحت کدهای نوشته شده کلیک می‌شود در صورتی‌که پیغام No Error Founds داده شود، درست بودن دستورات تأیید می‌شود.

گزینه Save and close را انتخاب تا تغییرات ذخیره شود.

مثال ۵-۲

گزارشی از جدول Orders ساخته شود که تمام رکوردهایی که فیلد کرایه (Freight) کمتر از ۲۰ ریال است به رنگ قرمز، بیشتر از ۴۰ ریال به رنگ سبز و بقیه به رنگ مشکی نمایش داده شود.

کد نوشته شده در بخش چهارم :

```
IF {Orders.Freight}<20 THEN CRRED
ELSE IF {Orders.Freight}>40 THEN CRGREEN
```

مثال ۵-۳

در جدول Orders تمام رکوردهایی که فیلدکرایه (Freight) در محدوده ۲۰ تا ۴۰ ریال است به رنگ ارغوانی نمایش داده شود.

```
IF {Orders.Freight}>20 And {Orders.Freight}<40 THEN crPurple
```

مثال ۵-۴

در جدول Products قیمت کل از قیمت واحد کالا و تعداد آن به‌دست آمده و در گزارش نمایش داده شود.

شکل ۱۹-۵

در ابتدا باید یک فیلد قیمت کل برای محاسبات ایجاد شود. البته این فیلد در جدول تولید نمی‌شود و فقط توسط کریستال ریپورت قابل استفاده و مشاهده است. گزینه Formula Field را انتخاب نمایید.

گزینه New را انتخاب نموده و پنجره Formula Name مشاهده می‌شود. در محل Name نام فیلد مجازی برای قیمت کل را وارد نموده و دکمه فرمان Use Editor را انتخاب نمایید.

در پنجره Formula Workshop اسم فیلدهای مورد نظر در قسمت Report Fields انتخاب می‌شود و علامت * را بین دو فیلد می‌گذاریم. که کد زیر تولید می‌شود.

{Products.UnitPrice} * {Products.UnitsOnOrder}

کد را ذخیره نمایید و برنامه را اجرا کنید تا خروجی گزارش را مشاهده نمایید.

مثال۵-۵

می‌خواهیم در زمان اجرای برنامه فقط شماره کالای خاصی را وارد نموده و آن را در گزارش مشاهده نماییم.

در فرمی که به‌صورت اتوماتیک توسط کریستال ریپورت برای جدول Products تولید شده است کنترل های زیر را بر روی فرم ایجاد و تنظیمات مربوط به آنها را اعمال می‌نماییم.

Crviewer1: خصوصیت EnableRefreshButton را True تا در نوار ابزار آن نمایش داده شود.

Picture Box: خصوصیت Align به Align Bottom تغییر یابد. و کنترل‌های زیر در آن قرار دهید.

TextBox: خصوصیت Text خالی شود.

Lable: خصوصیت Caption به شماره مشتری تغییر یابد.

ADO: ارتباط به بانک برقرار شود.

Command Button: با نام CmdSearch و عنوان Search

و کدهای زیر در CmdSearch_Click نوشته می‌شود :

```
Me.Adodc1.CommandType = adCmdText
Me.Adodc1.RecordSource = "select * from Products where Productid=" &
Text1
Me.Adodc1.Refresh
Report.Database.SetDataSource Adodc1.Recordset
CRViewer91.ReportSource = Report
CRViewer91.ViewReport
Screen.MousePointer = vbDefault
```

برنامه را اجرا و کد محصول را وارد کرده و بر روی دکمه فرمان Search کلیک نمائید. سپس روی دکمه Refresh کریستال ریپورت کلیک نمائید. رکورد کالای خاص مشاهده می‌شود.

فصل ۶

کاربرد VBA برای ارتباط برنامه‌های ویژوال بیسیک با Microsoft Office

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- کاربرد VBA برای ارتباط برنامه‌های ویژوال بیسیک با برنامه‌های آفیس را شرح دهد.
- برنامه‌های ساده‌ای برای ارتباط برنامه‌های ویژوال بیسیک با برنامه‌های آفیس را ایجاد کند.

Visual Basic for Applications (VBA) از زبان ویژوال بیسیک مشتق گرفته شده است که ارتباط زبان ویژوال بیسیک با بسته نرم افزاری Microsoft Office می‌باشد. توابع در VBA از نظر ساختار شبیه به توابع دیگر زبان‌های برنامه نویسی است. اغلب توابع VBA مشابه با توابعی است که در زبان ویژوال بیسیک با آنها آشنا شده‌اید مانند : توابع ریاضی ...,Sin(),Abs(),Len() و توابع رشته ای ...,Trim(),Mid(),Len() و غیره .

یک برنامه کاربردی وظایف خودش را از طریق شیء ها انجام می‌دهد. برنامه‌های کاربردی مانند Word و Excel دارای صدها شیء هستند که می‌توانند از طریق VBA کار کنند. مانند :

```
ActiveCell.Range("A1").Borders.Weight = xlThin
```

شیء ActiveCell و Range و Borders که به‌صورت زیر مجموعه و Weight خصوصیت می‌باشد؛ همانند شئ فرم که کنترل های روی فرم شی‌های دیگری هستند و هر یک از شیء ها خصوصیات مربوط به خود را دارند.

```
Form1.Textbox1.text="VBA"
```

به دلیل اینکه Office از صدها شیء تشکیل شده است و ارتباط بین این شیء‌ها پیچیده است،

مایکروسافت، ابزاری را فراهم کرده است که بتوان با این شی‌ها به سادگی کار کرد. به کمک این ابزار می‌توان شیء ها را مشاهده و درهنگام نیاز از آنها استفاده کرد یا آنها را تغییر داد.

۱ـ۶ـ ارتباط Microsoft Office با Visual Basic

همه فایل‌های آفیس در درون خودشان دارای برنامه کاربردی ویژوال بیسیک هستند و می‌توان همانند VB از طریق IDE فایل، دستوراتی را تایپ نمود تا در فایل اجرا شوند. در برنامه‌های کاربردی آفیس ابزاری به نام Macro وجود دارد که به کاربر در انجام بعضی از فعالیت‌های تکراری کمک می‌کند. برای مثال کاربر می‌تواند یک ماکرو برای تولید یک جدول و رنگ متن و چیدمان اطلاعات و... ضبط نموده وآن را بر روی یک فایل دیگر و یا هرجایی که نیاز دارد، اجرا نماید. البته باید به این نکته توجه کرد که عمل ضبط ماکرو همان کدهای VBA است که تولید و در قالب یک Sub در یک Module ذخیره شده است.

نکته : توسط Macro می‌توان دستوراتی را به صورت اتوماتیک تولید وآنها را در قالب یک Sub مشاهده و در صورت نیاز تغییر یا حذف نمود.

مثال ۱ـ۶

ایجاد یک ماکرو ساده: یک ماکرو با نام Macro1 در محیط Excel تولید نمایید (شکل ۱ـ۶)که فعالیت‌های زیر در آن انجام شود :

- در ردیف اول تولید اعداد اتوماتیک از ۳۰ تا ۳۳
- ردیف ۳ ستون A، فرمول =Len("Visual")
- ردیف ۴ ستون A، فرمول =ABS(-2)
- ردیف ۵ ستون A، فرمول =Sin(A1)

	A	B	C	D	E
۱	۳۰	۳۱	۳۲	۳۳	
۲					
۳	۶				
۴	۲				
۵	-۰.۹۸۸				
۶					

شکل ۱ـ۶

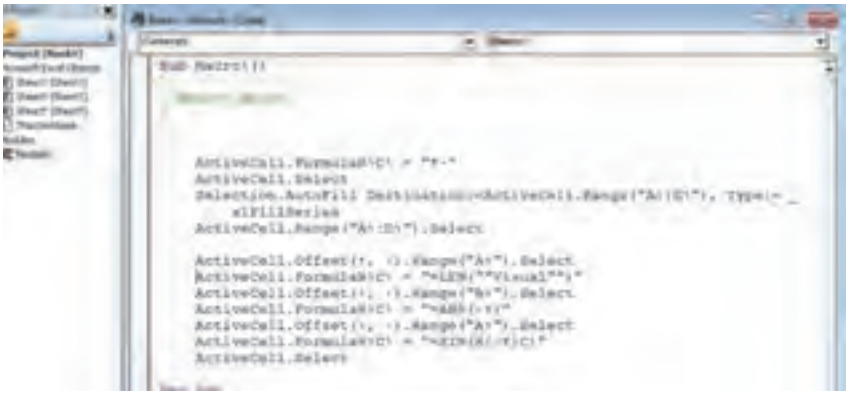
بعد از اتمام ضبط ماکرو، کدهای تولید شده را مشاهده نمایید.

نکته : مسیر مشاهده محیط IDE ویژوال در آفیس (فشار کلیدهای Alt + F11)

OFFICE 2003 : از منوی Tools زیر منوی Macro و انتخاب گزینه Visual Basic Editor

OFFICE 2007 : از زبانه Developer انتخاب گزینه Visual Basic (مشاهده Developer با انتخاب گزینه Show Developer tab in the Ribbon از Excel Option)

شکل ۲ــ۶ کدهای ماکرو ضبط شده در Moudul را نشان می‌دهد.



شکل۲ــ۶

مثال۲ـ۶

ویرایش ماکرو از طریق کد نویسی برای خانه فرمول Sin، کادر ایجاد کرده و رنگ سلول را به رنگ قرمز تغییر دهید :

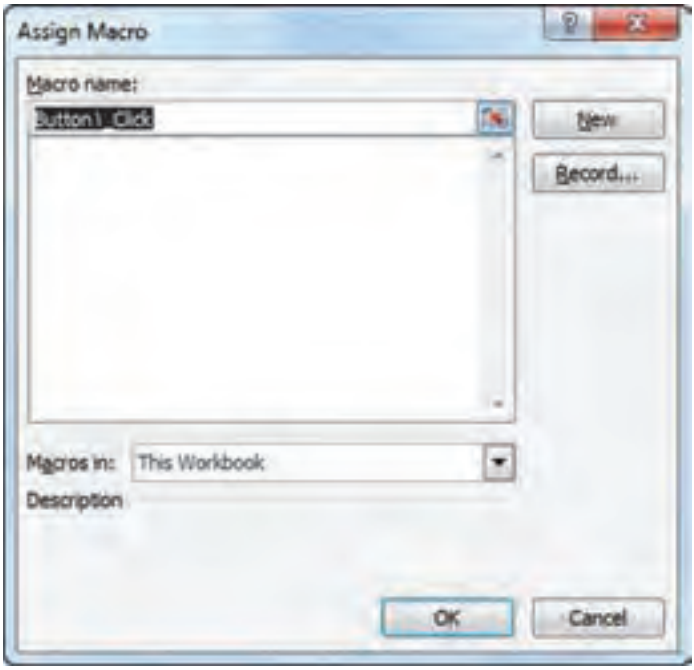
بعد از آخرین خط دستور Macro1 کد زیر را بنویسید :

```
ActiveCell.Range("A1").Borders.Weight = xlThin
ActiveCell.Range("A1").Interior.Color = vbRed
```

مثال۳ـ۶

ایجاد Sheet جدید با نام دلخواه بر روی Work Sheet به‌وسیلهٔ یک Command Button یک Command Button را انتخاب و بر روی Sheet اضافه کنید. روی کنترل اضافه شده

کلیک راست کرده و گزینه Assign Macro و بعد New را انتخاب نمایید (شکل ۳ــ۶).



شکل ۳ــ ۶

در Modual برای رویداد Button1_ Click تکه کد زیر را تایپ نمایید (شکل ۴ــ۶).

شکل ۴ــ ۶

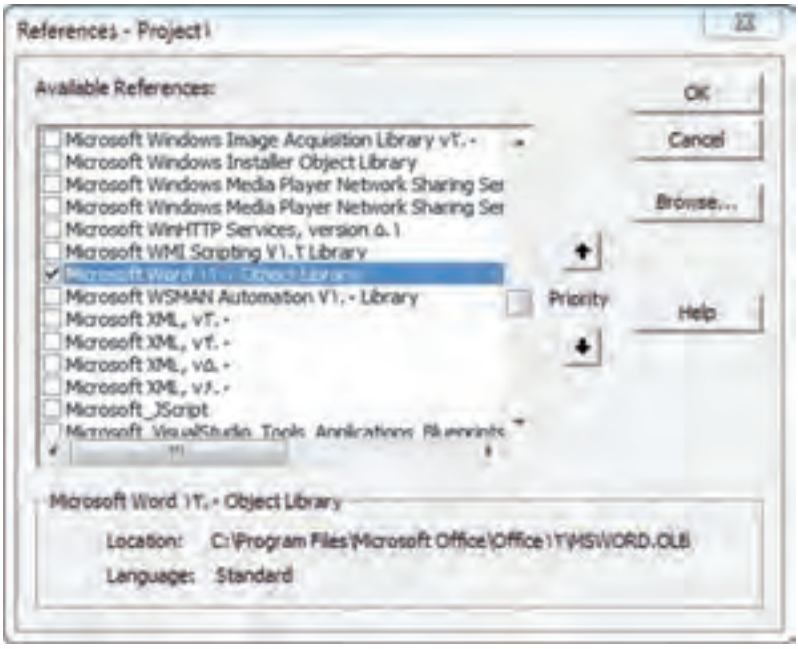
محیط باز شده را بسته و در Sheet مورد نظر دوباره Command Button را انتخاب کرده و اجرای برنامه را مشاهده نمایید.

۲ـ۶ـ ارتباط Visual Basic با Microsoft Office

نرم‌افزار Visual Basic می‌تواند تمام مؤلفه‌های ActiveX، فایل کتابخانه‌ای (olb) نرم‌افزارهای نصب شده در سیستم عامل را مشاهده و در صورت نیاز به پروژه اضافه و از آن استفاده نماید. وقتی یک فایل کتابخانه‌ای نرم افزاری انتخاب شود می‌توان از آن به عنوان یک شیء در محیط برنامه‌نویسی استفاده نمود. که این شیء شامل خصوصیت‌ها (properties) و متدها (methods) می‌باشد.

یک فایل کتابخانه‌ای مانند WORD را از گزینه References در منوی Project انتخاب نمایید (شکل ۵ـ۶).

توجه: حتماً نرم‌افزار word نصب باشد. و شماره ۱۲ یا ۱۱ یا ۱۰ در این گزینه بستگی به نسخهٔ Office نصب شده دارد.



شکل ۵ــ ۶ ــ انتخاب فایل کتابخانه‌ای WORD

در برنامه یک متغیر برای شیء Word ایجاد نمایید، تا بتوانید به دیگر شیء‌ها و خصوصیات Word دسترسی پیدا کنید.

۳ـ۶ـ بعضی از شیءهای مورد استفاده در نرم‌افزار Word

• تعریف یک متغیر از شیء کتابخانه‌ای	Dim objWord As New Word.Application
این متغیر می‌تواند تمام وظایف Word را در قالب یک شیء ارائه نماید	
• مشاهده نرم‌افزار مربوطه • مشاهده وضعیت پنجره نرم‌افزار باز شده • بستن نرم‌افزار باز شده	objWord.Visible objWord.WindowState objWord.Quit
• با استفاده از شیء Document از word، می‌توان یک صفحه ایجاد، باز، ذخیره و یا سندی بسته شود	objWord.Documents.Add objWord.Documents.Open objWord.Documents.Close objWord.Documents.Save
• برای درج متن به سند word • برای درج یک پاراگراف خالی	objWord.Selection.TypeText objWord.Selection.TypeParagraph
• برای تعیین مشخصات شیء پاراگراف در سند • برای تعیین مشخصات صفحه سند	objWord.Selection.Paragraphs objWord.Selection.PageSetup
• سند Word را می‌تواند تحت نام جدید ذخیره نماید • برای تعیین مشخصات شیء جدول	objWord.ActiveDocument.SaveAs objWord.ActiveDocument.Tables
• پاک کردن متغیر شیء از حافظه	Set objWord = Nothing

بعضی از شیءهای نرم‌افزارها پیچیده می‌باشد که می‌توانید برای اطلاعات بیشتر به MSDN مراجعه نمایید.

مثال ۴ـ۶

برنامه‌ای بنویسید که جملهٔ درون کادر متنی را در یک سند Word نمایش دهد و آنرا با اسم Word App.Doc ذخیره نماید.

تکه کد برنامه به صورت زیر می‌باشد :

```
Public objWord As New Word.Application ' General VARIABLE
Private Sub cmdCreat_Click()
```

```
    If txtWord.Text = vbNullString Then
        MsgBox "Insert Text In Textbox", vbInformation, "Warning"
        txtWord.SetFocus
        Exit Sub
    End If
    objWord.Visible = True

    objWord.Documents.Add
    objWord.Selection.Paragraphs.Alignment = wdAlignParagraphCenter
    objWord.Selection.TypeText txtWord.Text
    objWord.Selection.TypeParagraph
    objWord.Selection.TypeText txtWord.Text
End Sub
Private Sub cmdQuit_Click()
    On Error Resume Next
    objWord.Visible = False
    objWord.Quit
    Set objWord = Nothing
End Sub
Private Sub cmdSave_Click()
  objWord.ActiveDocument.SaveAs FileName: =App.Path & "\WordApp.doc"
    MsgBox " Word File Save in Path " & App.Path & "\WordApp.doc"
End Sub
```

کد را ذخیره و برنامه را اجرا نمایید (شکل ۶ــ۶).

شکل ۶ـ۶

مثال ۵ـ۶

در فصل ۱ برنامه دفترچه تلفن را با استفاده از توابع کار با فایل‌ها نوشتیم. در اینجا می‌خواهیم از آن فایل در محیط Excel گزارش تهیه کنیم.

ابتدا گزینه Excel Microsoft 12.0 Object Library را انتخاب نمایید و در فرم طراحی شده در فصل۱ دو Command Button با نام های cmdReport , cmdClose را اضافه نمایید و برای رویداد کلیک آنها کدهای زیر را بنویسید.

```
Dim objExcel As New Excel.Application
Private Sub cmdClose_Click()
  objExcel.Visible = False
```

```
  objExcel.Quit
  Set objExcel = Nothing
End Sub

Private Sub cmdReport_Click()
  Dim iRow As Byte
  Dim objBook As New Excel.Workbook
  Dim objSheet As New Excel.Worksheet

  objExcel.Visible = True
  Set objBook = objExcel.Workbooks.Add
  Set objSheet = objBook.Worksheets("sheet1")
  objSheet.Range("A1:B1").Merge
  objSheet.Range("A1:B1").Value = " Telephone Book "
  objSheet.Range("A1:B1").Font.Bold = True
  objSheet.Range("A1:B1").Font.Color = vbBlue
  objSheet.Range("A1:B1").VerticalAlignment = xlTop
  objSheet.Range("A1:B1").HorizontalAlignment = xlCenter
  iRow = 2
  With objSheet
    .Cells(iRow, 1).Value = "Name"
    .Cells(iRow, 1).BorderAround LineStyle:=3, Weight:=xlThin
    .Cells(iRow, 2).Value = "Number"
    .Cells(iRow, 2).BorderAround LineStyle:=3, Weight:=xlThin
  End With
  For i = 1 To 10
    iRow = iRow + 1
```

```
        With objSheet
            . Cells(iRow, 1). Value = recperson. Name
            . Cells(iRow, 1). BorderAround LineStyle: =1, Weight: =xlThin
            . Cells(iRow, 2). Value = recperson. Tel
            . Cells(iRow, 2). BorderAround LineStyle: =1, Weight: =xlThin
        End With
    Next
    objSheet. Columns. AutoFit
    objSheet. Name = "Telephone"
    If Dir(App. Path & " \tel. xls") = "tel. xls" Then
        Kill App. Path & " \tel. xls"
        objBook. SaveAs App. Path & "\tel. xls"
    Else
        objBook. SaveAs App. Path & "\tel. xls"
    End If
    Set objSheet = Nothing
    Set objBook = Nothing
End Sub
```

برنامه را ذخیره و اجرا نمایید.

تحقیق: چگونه می‌توان به یک فایل اکسل موجود اطلاعات دیگری اضافه نمود.

خودآزمایی

پروژه‌ای را ایجاد کنید که عمل بررسی املای کلمات را در سندهای word انجام دهد (spelling checker)

فصل ۷

API[۱] ویندوز

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- مفهوم API ویندوز را شرح دهد.
- دلیل نیاز برنامه‌های شما به روال‌های ویندوز را بیان کنید.
- کتابخانه‌های پیوند دینامیکی (پرونده‌های DLL) را شرح دهد.
- نحوهٔ اتصال ویژوال بیسیک به روال‌های API ویندوز از طریق دستور Declare را توضیح دهد.

در این فصل، نحوهٔ دسترسی به روال‌های داخلی ویندوز شرح داده می‌شود. هرچند ویژوال بیسیک می‌تواند تقریباً همه کارهای مورد نیاز شما را انجام دهد، ولی در برخی از برنامه‌ها اگر بخواهید از ویژوال بیسیک برای انجام بعضی قابلیت‌ها استفاده کنید، برنامه‌نویسی مشکل می‌شود. خوشبختانه در چنین شرایطی روال‌هایی در ویندوز وجود دارند که درون پرونده‌های DLL ذخیره شده‌اند.

با استفاده از روال‌های ویندوز می‌توانید قدرت ویژوال بیسیک را در برنامه‌های خودتان افزایش دهید و از آن بخواهید کارهایی را انجام دهد که فقط ویندوز حق اجرای آن‌ها را دارد. در این فصل، نه تنها نحوهٔ دسترسی به این روال‌های ویندوز توضیح داده می‌شود بلکه تعدادی از این روال‌ها که می‌توانید با آن‌ها کار کنید، بررسی می‌شوند.

۱ـ۷ـ API ویندوز

API ویندوز مجموعه‌ای از روال‌هاست که در دسترس برنامه‌نویس می‌باشد. به عبارت دیگر، این روال‌های API دقیقاً مانند توابع داخلی خود ویژوال بیسیک عمل می‌کنند. وقتی لازم است تا از کد یک روال API استفاده کنید، برنامه ویژوال بیسیک آن روال را فراخوانی می‌کند. زمانی که API

۱ـ Application Programming Interface

ویندوز به پایان رسید، کنترل به برنامه برمی‌گردد و اجرای آن ادامه پیدا می‌کند.

API سرنام Application Programming Interface به معنای رابط برنامه‌نویسی کاربردی است و به مجموعه‌ای از روال‌های داخلی ویندوز اطلاق می‌شود که می‌توانید آن‌ها را از ویژوال بیسیک فراخوانی کنید.

تمام روال‌های API ویندوز در پرونده‌های خاصی با نام DLL ذخیره می‌شوند. چند هزار روال API وجود دارد که می‌توانید آن‌ها را به کار ببرید. این روال‌های API درون پرونده‌هایی وجود دارند که در پوشه‌های Windows و Windows\System ذخیره شده‌اند. هنگام نصب ویندوز، پرونده‌های DLL، هم نصب می‌شوند. بنابراین، به‌طور خودکار به این کتابخانه‌ها دسترسی دارید.

DLL سرنام Dynamic Link Library به معنای کتابخانه پیوند پویاست. پرونده‌های DLL در دسترس برنامه‌هایی که به زبان ویژوال بیسیک و زبان‌های دیگری (که از DLL پشتیبانی می‌کنند) نوشته شده‌اند، قرار دارد.

اکثر پرونده‌های DLL دارای پسوند DLL. یا EXE. هستند. هر برنامه‌ای که می‌نویسید، به پرونده‌های DLL ویندوز دسترسی دارد.

سه پرونده DLL که معمولاً به کار می‌روند، عبارتند از :

- **USER32.DLL:** شامل توابعی است که محیط ویندوز و رابط کاربر مثل مکان‌نماها، منوها و پنجره‌ها را کنترل می‌کنند.
- **GDI32. DLL:** شامل توابعی است که خروجی برنامه به صفحه نمایش و ابزارهای دیگر را کنترل می‌کنند.
- **KERNEL32. DLL:** شامل توابعی است که سخت‌افزار و رابط‌نرم افزار داخلی ویندوز را کنترل می‌کنند. اکثر روال‌های مربوط به حافظه، پرونده و پوشه درون KERNEL32.DLL قرار دارند.

این سه پرونده، اغلب روال‌ها یا توابع API را نگه می‌دارند. شما می‌توانید این توابع را در برنامه‌های ویژوال بیسیک خودتان فراخوانی کنید. با یک نگاه کوتاه به پوشه‌های Windows و Windows\System چند کتابخانه پیوند دینامیکی دیگر را نیز می‌بینید مثل

COMDLG.DLL و MAPI32.DLL و NETAPI32 و WINMM.DLL.

همچنان‌که مایکروسافت قابلیت‌هایی را به سیستم عامل اضافه می‌کند، پرونده‌های جدید DLL هم ظاهر می‌شوند.

هنگامی که برنامهٔ جدیدی را در سیستم خودتان نصب می‌کنید، این برنامه DLL خاص خودش را ارایه می‌کند. بنابراین، در طول زمان تعداد زیادی پرونده DLL روی سیستم خواهید داشت.

۷ـ۲ـ پرونده‌های DLL

اصطلاح پیوند پویا معنای خاصی برای برنامه‌نویسان دارد. وقتی گفته می‌شود که یک روال با یک برنامه پیوند دینامیکی دارد، بدین معناست که این روال (سابروتین یا تابع) تا قبل از کامپایل برنامه، به آن متصل نمی‌شود. این تابع فقط در زمان اجرا در دسترس می‌باشد. توابعی که درون پنجره کد می‌نویسید دارای پیوند استاتیکی می‌باشند یعنی هر وقت برنامه را کامپایل می‌کنید، این توابع با بقیه کد اصلی ترکیب می‌شوند. اما پرونده‌های DLL با برنامه ترکیب نمی‌شوند. برنامهٔ شما به این روال‌ها در زمان اجرا دسترسی دارد اما پرونده EXE برنامه شامل روال‌های واقعی DLL نمی‌باشد.

نکته: مزیت استفاده از روال‌های DLL آن است که چند برنامه اجرایی تحت ویندوز می‌توانند به یک روال از پرونده DLL دسترسی داشته باشند. علاوه بر این همه کاربران باید روال‌های استاندارد DLL را داشته باشند. از آن‌جایی که برای اجرای یک برنامه ویژوال بیسیک، وجود ویندوز ضروری است، پس پرونده‌های DLL در دسترس می‌باشند.

۷ـ۳ـ دستور Declare

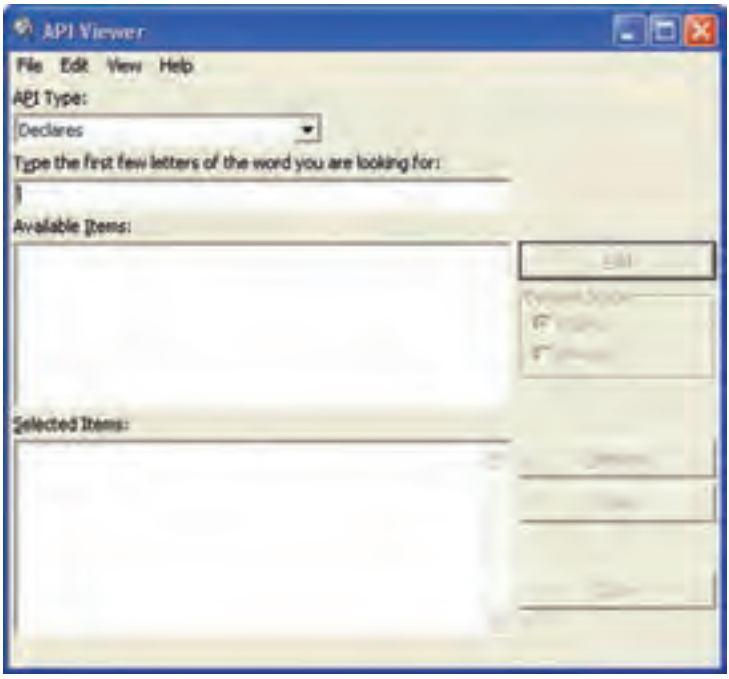
برای فراخوانی روال‌های API ویندوز باید از یک دستور خاص با نام Declare استفاده کنید. در مورد توابع داخلی ویژوال بیسیک به دستور Declare نیاز ندارید چون ویژوال بیسیک طرز کار توابع خودش و آرگومان‌های این توابع را می‌شناسد. اما روال‌های API خارج از میدان دید ویژوال بیسیک می‌باشند. بنابراین باید از این دستور استفاده کنید.

۱ـ۳ـ۷ـ ابزار API Viewer : ویندوز شامل هزاران روال API است که می‌توانید آن‌ها را فراخوانی کنید. حتی دانستن قالب تعداد کمی از این روال‌ها هم کار مشکلی است. در این رابطه، ویژوال بیسیک یک ابزار خاص به نام API Viewer دارد که برای راهنمایی در مورد قالب روال‌های API می‌توانید از این ابزار استفاده کنید.

ابزار API Viewer، روال‌های API را نمایش می‌دهد و آن‌ها را از نظر موضوع دسته‌بندی می‌کند طوری که می‌توانید روال‌های مورد نیاز خود را به سادگی پیدا کنید.

دکمهٔ Copy روی API Viewer اطلاعات مربوط به اعلان انتخاب شده را درون Clipboard ویندوز کپی می‌کند. همچنین اگر قبل از کلیک‌کردن روی دکمهٔ Copy، گزینه‌های Public یا Private این ابزار را کلیک کنید دیگر مجبور نیستید شناسه‌های اعلان را به صورت دستی تغییر دهید. زیرا API Viewer کلمات کلیدی مناسب را در دستور Declare مشخص می‌کند.

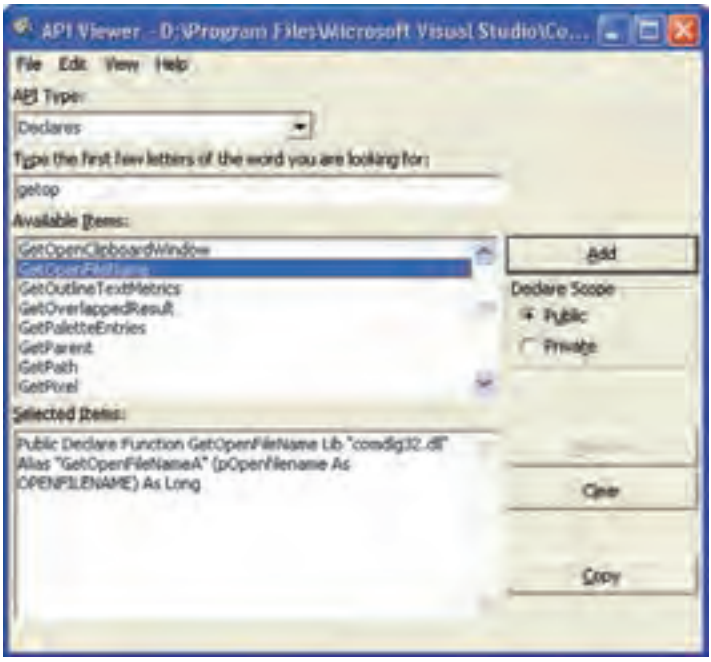
برای دسترسی به ابزار API Text Viewer ، از Start گزینه Programs، سپس گزینه Microsoft Visual Studio 6.0 و سپس گزینهٔ Microsoft Visual Studio 6.0 Tools و در پایان API Text Viewer را انتخاب کنید.



شکل ۱ــ۷ــ می‌توانید از طریق APIViewer به سادگی قالب روال‌های API را مشاهده کنید.

API Viewer اطلاعات اساسی خود را از پرونده‌های متنی APILOAD.txt، MAPI32.txt و WIN32API.txt پیدا می‌کند. این پرونده‌ها همراه API Viewer روی سیستم نصب می‌شوند.

از آن‌جایی که اکثر روال‌های API که موردنظر شما هستند، در پرونده WIN32API.txt قرار دارند. گزینه Load Text File را از منوی File پنجره API Viewer انتخاب و سپس پرونده WIN32API.txt را انتخاب کنید.



شکل ۲ــ۷

دقت کنید کادر لیست موجود در بالای پنجره دارای عنوان API Type است. با بازکردن این کادرلیست، سه مقدار زیر را مشاهده می‌کنید :

- **Constants:** همه ثابت‌های نامگذاری شده که پرونده API بارگذاری شده تشخیص می‌دهد را فهرست می‌کند.
- **Declares:** همه اعلان‌هایی که درون پرونده API بارگذاری شده ظاهر می‌شوند را فهرست می‌کند.
- **Types:** همه انواع داده‌ها که پرونده API بارگذاری شده تشخیص می‌دهد را فهرست می‌کند.

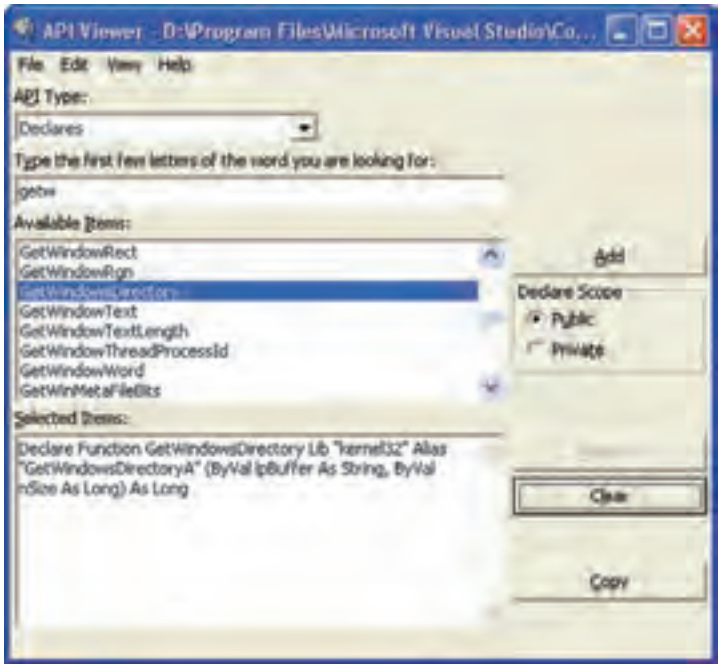
کادر لیست Available Items شامل کلیه روال‌های API ویندوز (مربوط به پرونده که بارگذاری کرده‌اید) و نوع مقداری است که می‌خواهید مشاهده کنید. مثلاً اگر می‌خواهید دستور Declare مورد نیاز روال GetWindowsDirectory را پیدا کنید، مراحل زیر را دنبال نمایید :

۱ــ از لیست API Type گزینه Declares را انتخاب کنید. چند دستور Declare درون لیست Available Items ظاهر می‌شوند.

۲ــ می‌توانید چند حرف اول یک دستور Declare خاص را درون کادر متن تایپ کنید تا این دستور به سرعت پیدا شود. بدین منظور Getw را تایپ کنید. همه گزینه‌هایی که با این حروف شروع می‌شوند درون کادر لیست Available Items ظاهر می‌شوند.

۳ــ این فهرست را مرور کنید تا گزینه GetWindowsDirectory را پیدا کنید.

۴ــ روی گزینه GetWindowsDirectory دابل کلیک کنید تا دستور Declare مورد نیاز تابع مطابق شکل ۳ــ۷ نمایش داده شود.



شکل ۳ــ۷ــ API Viewer دستور Declare مورد نیاز برای دستوری که انتخاب کرده‌اید را نمایش می‌دهد.

اکنون می‌توانید تمام دستور Declare را کپی کنید و آن را درون پنجره کد برنامه خودتان قرار دهید.

۴ـ۷ـ تعریف تابع API

قبل از این‌که کار با توابع را شروع کنیم بهتر است با فرم کلی توابع API آشنا شویم :

[Public| Private] Declare Function Function _ name Lib "DllFilename"_
[alias "Function alias"] (argument _ List) as data_type

دستور Declare برای تعریف تابع استفاده می‌شود. این دستور می‌تواند داخل یک ماژول یا فرم به کار رود. اگر داخل فرم استفاده شود کلید واژه Private برای آن به‌کار می‌رود و اگر داخل ماژول به کار رود، می‌تواند با یکی از کلید واژه‌های Public یا Private استفاده شود.

Function name، نام تابع API است که برای فراخوانی تابع مورد استفاده قرار می‌گیرد.

Dll–Filename، نام پرونده Dllای است که این تابع داخل آن قرار دارد. این نام نباید شامل مسیر باشد زیرا مسیر این پرونده‌ها ثابت است. نوشتن پسوند این پرونده الزامی نیست.

Function alias، این پارامتر اختیاری است. نامی است که داخل پرونده dll قرار دارد.

Argument List، لیستی از آرگومان‌های تابع می‌باشد. این لیست نشان می‌دهد چه تعداد متغیر و از چه نوعی باید به تابع فرستاده شود. نحوه تعریف آرگومان‌ها به شکل زیر است :

[byval¦ byRef] argument – Name as data – type

که در آن argument – Name نام آرگومان مورد نیاز است. انتخاب این نام اختیاری است و data – type نوع آرگومان را مشخص می‌کند.

کلید واژه ByVal، ByRef روش‌های خاص ارسال پارامتر به تابع API است. این دو روش کاملاً متمایز هستند. روش و متد ByRef به طور معمول استفاده می‌شود مگر اینکه به صراحت از ByVal استفاده شود. به همین دلیل شما در موقع تعریف API کلمه ByRef را نمی‌بینید و فقط به ByVal نیاز است.

ByVal به این مفهوم است که تابع API نتواند مقدار این متغیر را تغییر دهد. ولی ByRef به معنی «با مرجع» می‌باشد (By Refrence) و به این مفهوم است که تابع API می‌تواند مقدار این متغیر را تغییر دهد. با این روش نمی‌توان یک مقدار ثابت را ارسال کرد.

type ـ Data نوع داده‌ای است که تابع برمی‌گرداند.

۱ـ۴ـ۷ـ تابع Messagebeep : یکی از ساده‌ترین روال‌های API است. این تابع یکی از دو کار زیر را انجام می‌دهد :

اگر آرگومان ارسالی به این تابع مثبت باشد، صدای بوق از طریق کارت صوتی رایانه به گوش می‌رسد.

شکل کلی این تابع به صورت زیر است :

```
Private Declare Function messagebeep Lib 'user32' alias 'message ـ beep'
(byval wtype as long) as long
```

دستور Declare دقیقاً به ویژوال بیسیک اعلام می‌کند که چگونه تابع messagebeep را پیدا کند و چگونه مقادیر را منتقل نماید. این تابع درون پرونده user32.dll است.

مثال ۱-۷

باز و بسته شدن درب CD ROM : بر روی فرم دو دکمه فرمان برای OPEN و CLOSE قرار دهید. و تابع mciSendString را توسط APIViewer اضافه نمایید.

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA"
  (ByVal lpstrCommand As String, ByVal lpstrReturnString As String, _
  ByVal uReturnLength As Long, ByVal hwndCallback As Long) As
 LongPrivate Sub CMDOPEN_Click()
   mciSendString "Set CDAudio Door Open Wait", 0&, 0&, 0&
End Sub
Private Sub CMDCLOSE_Click()
   mciSendString "Set CDAudio Door Closed Wait", 0&, 0&, 0&
End Sub
```

تحقیق : برنامه قبل را طوری تغییر دهید که بر اساس زمان بندی مشخصی درب CD ROM باز و بسته شود.

مثال ۲-۷

نمایش نام کاربر ویندوز : فرمی به همراه یک کادر متن ایجاد کنید و کد صفحهٔ بعد را بنویسید :

```
Private Declare Function GetUsername Lib "advapi32.dll" Alias
"GetUsernameA" (ByVal lpBuffer As String, nSize As Long) As Long

Private sub Form_Load()
   Dim Buffer As String

Buffer = String (255,0)
```

رشته‌ای به طول ۲۵۵ ایجاد کرده و آن را با کاراکتری با کد صفر پر می‌کنیم

```
    GetUsername Buffer, 255          ' نام کاربر ویندوز را دریافت می‌کنیم
Buffer = Left$ (Buffer, InStr(Buffer, Chr$(0))-1)'
                                     قسمتی از رشته را جدا می‌کنیم که نام کاربر در آن قرار دارد
    Text1 . Text = Buffer'           نام کاربر را در text box نمایش می‌دهیم
End Sub
```

در این کد به کمک تابع .Instr شمارهٔ اولین کاراکتر با کد صفر در رشتهٔ Buffer را پیدا کرده و به کمک تابع left$ قسمتی از رشته را که قبل از این کاراکتر قرار دارد (نام کاربر) جدا می‌کند.

مثال ۳-۷

تشخیص بزرگی و کوچکی و نوع (عددی ـ حرفی) نویسه‌های وارد شده از صفحه کلید:

یک فرم ایجاد کنید و کد زیر را در آن بنویسید :

```
Private Declare Function IsCharAlpha Lib "user32" Alias "IsCharAlphaA"
(ByVal cChar As Byte) As Long

Private Declare Function IsCharAlphaNumeric Lib "user32" Alias
"IsCharAlphaNumericA" (ByVal cChar As Byte) As Long

Private Declare Function IsCharLower Lib "user32" Alias "IsCharLowerA"
(ByVal cChar As Byte) As Long

Private Declare Function IsCharUpper Lib "user32" Alias "IsCharUpperA"
(ByVal cChar As Byte) As Long

Private Sub Form_KeyPress(KeyAscii As Integer)
    Dim Str As String
   Me.Cls
   If IsCharAlphaNumeric(KeyAscii) Then Str = "alphabetic Numeric"
```

```
If IsCharAlpha(KeyAscii) Then Str = “alphabetic”
If IsCharLower(KeyAscii) Then Str = Str + “Lower”
If IsCharUpper(KeyAscii) Then Str = Str + “Upper”
Me.Print “You pressed: ” + Chr$(KeyAscii)
Me.Print “This is: ” + Str
End Sub
```

در صورتی که کلید فشرده شده حرفی باشد خروجی تابع Is charAlpha مساوی True است و اگر حرفی یا عددی باشد خروجی تابع Is char Alpha Numeric مساوی True است و اگر کلید حرفی فشرده شده از حروف بزرگ باشد (مثل A) خروجی تابع Is char Upper مساوی True و اگر از حروف کوچک باشد مثل a خروجی تابع Is charlower مساوی True است.

مثال ۴-۷

مخفی یا ظاهر کردن Taskbar ویندوز:

بر روی فرم دو دکمه فرمان برای ظاهر و مخفی شدن نوار ایجاد نمایید و کدهای زیر را بنویسید(با استفاده از توابع SetWindowPos و FindWindow)

```
Private Hwnd1 As Long
Private Const SWP_HIDEWINDOW = &H80
Private Const SWP_SHOWWINDOW = &H40
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long,
 ByVal hwndlnsertAfter As Long, ByVal x As Long, ByVal y As Long,
 ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA"
(ByVal lpClassname As String,ByVal lpWindowName As String) As Long
Private Sub HideTask_Click()
   Hwnd1 = FindWindow("Shell_Traywnd", "")
   Call SetWindowPos(Hwnd1, 0, 0, 0, 0, 0, SWP_HIDEWINDOW)
End Sub
```

```
Private Sub ShowTask_Click()
   Call SetWindowPos(Hwnd1, 0, 0, 0, 0, 0, SWP_SHOWWINDOW)
End Sub
```

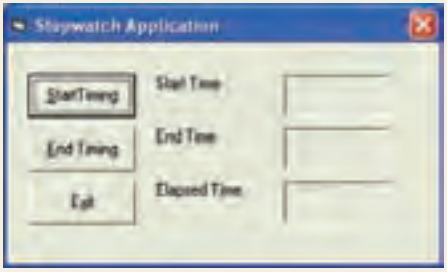
خودآزمایی

(**توضیح** :تمام تمرین‌های این فصل را با استفاده از توابع API بنویسید.)

۱ــ API چیست؟ کاربرد آن را شرح دهید.

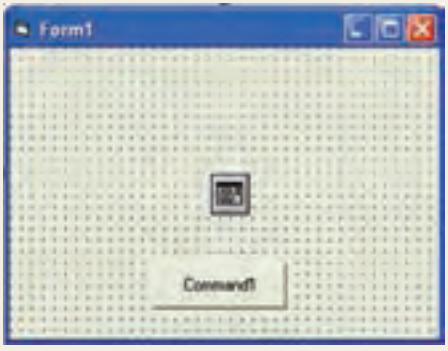
۲ــ انواع پرونده‌های DLL را نام ببرید و کاربرد هرکدام را توضیح دهید.

۳ــ فرمی به شکل زیر طراحی کرده و مشخصه‌های آن را تنظیم کنید. با کلیک روی دکمه‌های Start Timing و End Timing زمان سیستم خوانده شده و در برچسب‌های مقابل آن‌ها نمایش داده می‌شود. سپس اختلاف آن‌ها محاسبه و در برچسب سوم نمایش داده می‌شود.



شکل ۴ــ۷

۴ــ به وسیلهٔ شیء CommonDialog یک پرونده صوتی از نوع WAVE را انتخاب و به وسیله تابع SndPlaySound آن را پخش کنید.



شکل ۵ــ۷

۵ــ این مثال، تصویر زمینه دسک‌تاپ و همچنین تصویر کل صفحه نمایش را در یک شیء تصویر کپی می‌کند.

شکل ۶ــ۷

توضیح : در این تمرین از توابع API زیر استفاده کنید :

۱) GetDesktopWindow()

۲) GetDC()

۳) PaintDesktop()

۴) ReleaseDC()

۵) SearchBlt()

فصل ۸

انجام پروژه پایانی

مهارت در برنامه‌نویسی با تکرار و تمرین برنامه‌نویسی به‌صورت عملی امکان‌پذیر است، بدین‌منظور دو هفته از زمان آموزش این درس اختصاص به انجام یک پروژه داده شده است. در این فصل، پروژه‌های برنامه‌نویسی نمونه ارایه می‌شوند. برای پرهیز از افزایش حجم کتاب، خلاصه‌ای از آن‌ها در زیر شرح داده می‌شود.

این پروژه‌ها به‌عنوان نمونه بوده و پس از تجزیه و تحلیل این پروژه در کلاس، هنرجویان می‌توانند پروژهٔ موردنظر خود را با نظر هنرآموز درس انتخاب و اجرا نمایند. بدیهی است که هنرآموز محترم قسمتی از نمره عملی را به پروژه اختصاص خواهد داد.

هنرجویان عزیز توجه داشته باشند در انجام پروژه از مطالبی که در درس‌های بانک اطلاعاتی، چندرسانه‌ای، شبکه، بسته‌های نرم‌افزاری و برنامه‌سازی ۱، ۲ و ۳ آموخته‌اند، استفاده کنند و آن‌ها را در پروژه به‌کار گیرند.

مواردی که رعایت آن‌ها در انجام پروژه ضروری به‌نظر می‌رسد را به‌صورت خلاصه در زیر ذکر می‌کنیم تا هنرجویان مدنظر قرار دهند :

۱ــ بررسی اولیه و شناخت از مسأله و راه‌حل‌های آن (تجزیه و تحلیل پروژه)

۲ــ رعایت اصول و قوانین یک برنامه‌نویسی خوب

۳ــ کنترل دستگاه‌های ورودی و خروجی (صفحه کلید ــ ماوس و...) در هنگام Data Entry

۴ــ مدیریت فرم‌ها (صفحات ورود اطلاعات) و گزارشات خروجی

۵ــ طراحی یک رابط کاربرپسند (User Friendly Interface) با گرافیک مناسب

۶ــ مدیریت عناصر چندرسانه‌ای (صوت، تصویر، فیلم، پویانمایی) در پروژه (کنترل پخش و قطع آن‌ها)

۷ــ به‌کارگیری ماژول‌های مناسب اکتیوایکس و API در صورت نیاز در پروژه

۸ــ مدیریت کاربران پروژه

۹ــ تهیه و ایجاد برنامهٔ نصب برای پروژه

۱۰ــ تهیه مستندات داخلی و خارجی برای پروژه و ارائه User Manual

۱۱ــ مدیریت کامل بانک اطلاعاتی در پروژه (درج ــ حذف ــ ویرایش ــ جستجو)

۱۲ــ ارائه پروژه در کلاس به‌صورت یک نمایش PowerPoint برای آگاهی بیشتر همکلاسی‌ها و آشنایی هنرجو با چگونگی ارائه محتوا در یک محیط جمعی

الف) پروژه وضعیت تحصیلی دانش‌آموزان

فرم Login

در این فرم، کاربر نام کاربری و رمز خود را وارد می‌کند. انواع کاربر عبارتند از :

۱ــ مدیر

- قابلیت ویرایش نام و رمز مدیر

(توجه : پیش‌فرض Username = admin Password = 0000)

- قابلیت تعریف کاربر جدید از دو نوع معاون و دفتردار
- قابلیت ویرایش کاربر موجود
- قابلیت حذف کاربر موجود
- قابلیت حرکت بین رکوردها

۲ــ دفتردار

- قابلیت افزایش دانش‌آموز جدید
- قابلیت ویرایش دانش‌آموز
- قابلیت جستجوی دانش‌آموز براساس نام و نام‌خانوادگی یا شماره دانش‌آموزی
- قابلیت حذف دانش‌آموز
- قابلیت حرکت بین رکوردها

توجه : دفتردار فقط می‌تواند با مشخصات دانش‌آموز کار کند و با بخش وضعیت تحصیلی ارتباط ندارد.

۳ــ معاون

- قابلیت مشاهده مشخصات دانش‌آموز
- قابلیت ویرایش وضعیت تحصیلی
- قابلیت جستجوی دانش‌آموز براساس نام و نام‌خانوادگی یا شماره دانش‌آموزی

- قابلیت نمایش لیست دانش‌آموزان

توجه : معاون نمی‌تواند با مشخصات دانش‌آموز کار کند و فقط با بخش وضعیت تحصیلی ارتباط دارد.

توجه : تمام اطلاعات به‌وسیلهٔ دو تابع DeCode و EnCode رمزگشایی و رمزگذاری می‌شوند. در ضمن رمز بانک ۱۲۳۴ است.

ب) مدیریت کتابخانه مدرسه

- کتاب
- اعضاء
- کارمندان

برای هرکدام از موجودیت‌های فوق عملیات درج، حذف، ویرایش و جستجو فراهم شود.
کتاب‌های امانت داده شده از لیست کتاب‌های موجود کسر خواهد شد.

گزارشاتی مبنی بر کتاب‌های موجود، کتاب‌های امانت داده شده، کل کتاب‌های ثبت شده در کتابخانه ارائه شود.

مدیریت امانت کتاب به‌نحوی که در کتابخانه‌ها مرسوم است، انجام شود.

پ) مدیریت تاکسی تلفنی

موجودیت‌ها :

- راننده
- کارمند
- مشترکان

عملیات اصلی بانک اطلاعاتی روی این موجودیت‌ها اعمال شود.

صدور فاکتور برای مشترک، صورت‌حساب راننده، گزارشات روزانه، ماهیانه و سالیانه ارائه شود.

مدیریت کاربران سیستم طراحی و ارائه شود.

ت) مهدکودک

پروژه‌ای طراحی کنید که بتواند مطالعه موردی که در فصل هشتم کتاب بانک اطلاعاتی موجود است را پیاده‌سازی و مدیریت کند.

ث) دفترچه تلفن

از توضیح این پروژه به‌دلیل سادگی صرف‌نظر می‌کنیم و تجزیه و تحلیل و شناسایی موجودیت‌های آن را به‌عهدهٔ هنرجو واگذار می‌کنیم.

فصل ۹

آشنایی با MSDN (Microsoft Developer Network)

MSDN راهنمای کامل برنامه های مایکروسافت می‌باشد و می‌توانید از راهنمای کامل برنامه ها در قالب مقاله، سند و مثال استفاده نمایید. اطلاعاتی از بسته های نرم افزاری شامل Visual Studio Office, SQL ,Html,Web,Windows و... گردآوری شده است.

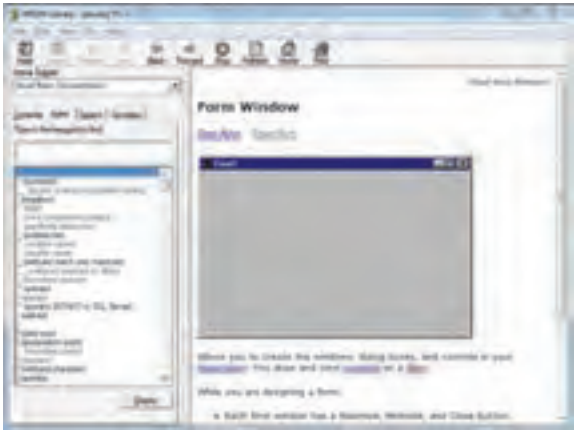
از نسخه January 2000 استفاده نمایید تا با Visual Basic 6 ارتباط داشته باشد. نسخه‌های بالاتر از آن با شاخه .NET ارتباط دارد.

این بسته در قالب یک DVD یا۳ CD ارائه می‌شود و در صورت نیاز می‌توانید آنرا از سایت Microsoft دانلود نمایید. http: //msdn.microsoft.com/subscriptions

در هنگام نصب حتماً گزینه Full را انتخاب نمایید تا تمام قسمتهای موجود روی ۳ CD بر روی دستگاه مقصد نصب شود در غیر اینصورت برای جستجوی هر مطلبی به CD های نصب نیازمندید (شکل ۱ــ۹)

شکل ۱ــ ۹

بعد از نصب کامل MSDN می‌توان با زدن کلید F1 در محیط Visual Basic محیط راهنما را مشاهده نمود (شکل ۲ــ۹).



شکل ۲ــ ۹

گزینه های موجود در MSDN

زبانه Contents : اطلاعات دسته‌بندی شده است. برای یافتن اطلاعات مورد نیاز زیر شاخه مورد نظر را کلیک نمایید.

شکل ۳ــ۹

زبانه Index : اطلاعات بر اساس حروف الفبا مرتب شده است و می‌توانید کلمه مورد نظر را تایپ نمایید.

شکل ۴ــ۹

زبانه Search : برای جستجوی کلمه ای در Msdn می‌توان آنرا تایپ نمود و حداکثر تا ۵۰۰ مورد، نمایش داده می‌شود.

مرتب‌سازی مطالب یافت شده بر اساس سه آیتم می‌باشد : عنوان صفحه (Title) و محل صفحه(Location) و رتبه صفحه (Rank)

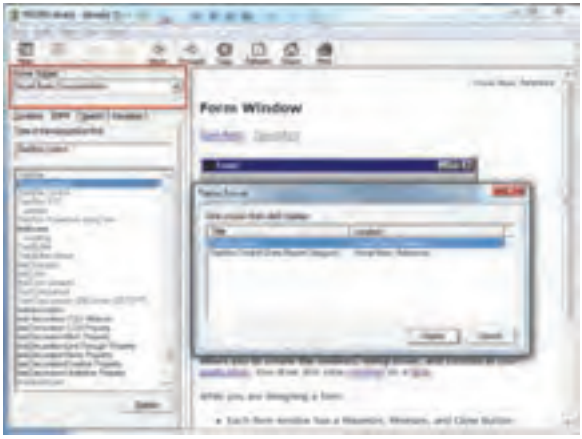
محدوده جستجو تعیین شود : جستجو فقط در عنوان سند انجام شود (Search Titles Only) جستجو به حروف کوچک و بزرگ حساس باشد(Math Similar Words) و جستجوی مجدد در همین تعداد صفحات یافت شده صورت پذیرد (Results Search Previous)

شکل ۵ ــ ۹

زبانه Favorites : در صورتی که می‌خواهید صفحه‌ای را که باز نموده‌اید در دفعات دیگر بدون جستجو به آن مراجعه کنید آنرا به این قسمت اضافه نمایید.

شکل ۶ــ ۹

می‌توان برای دسترسی به اطلاعات Visual Basic گزینه Visual Basic Documentation را از قسمت Active Subset انتخاب نمود.



شکل ۷ــ ۹

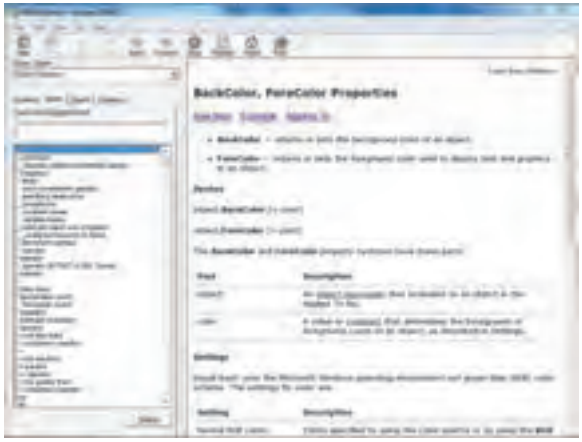
بعد از انتخاب و دابل کلیک کردن آن، صفحه درخواستی در سمت راست نمایش داده می‌شود می‌شود و یا در صورتیکه دارای چندین زیرمجموعه باشد ،زیرمجموعه‌های آن نمایش داده می‌شود که بر اساس نیاز خود یکی از آنها را انتخاب می نماییم.

به‌طور مثال در کد نویسی یک دکمه فرمان دستور زیر تایپ می شود :

```
cmdRun.Backcolor
```

می‌توان برای اطلاعات بیشتر از MSDN استفاده نمود. مکان‌نمای چشمک‌زن را بر روی خصوصیت BackColor انتقال و کلید F1 را فشار دهیم.

راهنمای MSDN باز می‌شود و توضیحات لازم و حتی مثال درباره این خصوصیت مشاهده می‌شود.



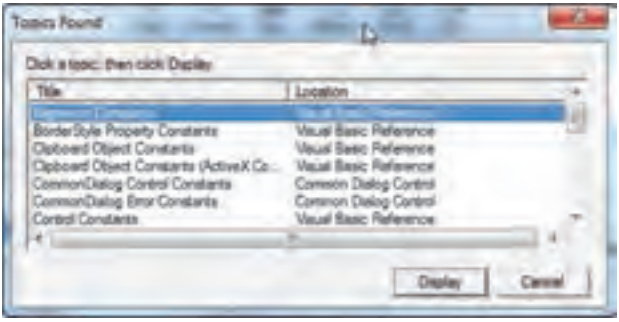
شکل ۸ــ۹

کد نویسی را ادامه داده و رنگ مورد نظر را تایپ می نماییم، بر روی رنگ کلید F1 را فشار داده و تمام ثابت های رنگ در راهنما مشاهده می‌شود.

```
cmdRun.Backcolor=vbRed
```

شکل ۹ ــ ۹

برای مشاهده دیگر ثابتها در خصوصیت های دیگر، بر روی See Also کلیک نمائید.



شکل ۱۰ ــ ۹

منابع

1ــ Sams Teach yourself Visual Basic 6 in 21 Days, Greg Perry.

2ــ Visual Basic 6 How To Program, Deitel & Deitel.

3ــ Learn Visual Basic 6, Lou Tylee.

4ــ Using Visual Basic 6, Bob Resel man,...

5ــ MCSD Visual Basic 6, Microsoft.

6ــ Desktop Applications With Visual Basic 6.0, Microsoft press.

فهرست کتاب‌های آموزشی مناسب دوره متوسطه حوزه کامپیوتر

۱ــ آموزش گام به گام Word 2007، آرزو خسروپور، ابوالفضل طاهریان، ۱۳۸۶، ناشر طاهریان.

۲ــ آموزش گام به گام Excell 2007، آرزو خسروپور، ابوالفضل طاهریان، ۱۳۸۶، ناشر طاهریان.

۳ــ آموزش کامل اکسل 2007، داریوش فرسایی، ۱۳۸۷، نشر علوم و فنون.

۴ــ آموزش تضمینی Photoshop cs3، حسین چناری، سیدجواد سیدمحسنی، ۱۳۸۷، نشر راه نوین.

۵ــ طراحی شبکه‌های کامپیوتری، محمد هادی امامی شوشتری، ۱۳۸۶، نشر جلوه.

۶ــ برنامه‌سازی جلد ۱ و ۲، محمدرضا عابدینی، ۱۳۸۶، نشر جنگل.

۷ــ کتاب کار و تمرین زبان تخصصی کامپیوتر، رمضانعلی یاسکی، مهدی احمدی، ۱۳۸۷، نشر پیام دانش روز.

۸ــ آموزش گام به گام کامپیوتر و ویندوز XP، آرزو خسروپور، ابوالفضل طاهریان، ۱۳۸۷، ناشر طاهریان.

۹ــ کلاس درس A dobe Flash cs3، در یک کتاب، پیمان دانش اشراقی، ۱۳۸۷، نشر صفارــ اشراقی.

۱۰ــ آموزش تصویری طراحی صفحات وب با PHP، رضا سرابی میانجی، ۱۳۸۶، نشر تمثیل.

۱۱ــ آموزش تصویری Photoshop cs3 ــcs، حبیب فروزنده، عابدکفاش، ۱۳۸۴، نشر عابد.

۱۲ــ آموزش تصویری Windows vista، حسین جوهرچی،۱۳۸۶،نشر عابد.

۱۳ــ آموزش Access 2002، ربابه واثق رحیم‌پور، ۱۳۸۵، نشر کتابخانه فرهنگ.

۱۴ــ آموزش اصول عیب‌یابی رایانه‌های شخصی ۱ و۲، فاطمه بخشیان و دیگران، ۱۳۸۶، نشر گستره علم و فن.

کدنویسی به زبان ساده

فهرست

فصل ۱ : روال‌ها و توابع ۱

فصل ۲ : پرونده‌ها ۲۴

فصل ۳ : مفاهیم شیءگرایی و ماژول کلاس ۷۱

فصل ۴ : آشنایی با Activex Data Objects (ADO) ۸۷

فصل ۵ : کاربرد نرم‌افزار Crystal Report ۱۲۶

فصل ۶ : کاربرد VBA برای ارتباط برنامه‌های ویژوال بیسیک با Microsoft Office ۱۴۰

فصل ۷ : API ویندوز ۱۵۰

فصل ۸ : انجام پروژه پایانی ۱۶۲

فصل ۹ : آشنایی با MSDN(Microsoft Developer Network) ۱۶۵

منابع ۱۷۰