

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِیْمِ

برنامه‌سازی (۳)

رشته کامپیوتر

گروه تحصیلی کامپیوتر

زمینه خدمات

شاخه آموزش فنی و حرفه‌ای

جباریه، علیرضا	۰۰۵
برنامه‌سازی (۳) / مؤلف: علیرضا جباریه. - تهران: شرکت چاپ و نشر کتاب‌های درسی ایران، ۱۳۹۱.	ب ۲۷۹ ج / ۱۳۹۱
۱۷۱ص. :مصور. - (آموزش فنی و حرفه‌ای) متون درسی رشته کامپیوتر گروه تحصیلی کامپیوتر، زمینه خدمات. برنامه‌ریزی و نظارت، بررسی و تصویب محتوا: دفتر برنامه‌ریزی و تألیف آموزش‌های فنی و حرفه‌ای و کاردانش وزارت آموزش و پرورش. ۱. برنامه نویسی - کتاب‌های درسی (متوسطه). ۲. الگوریتم‌های کامپیوتری - کتاب‌های درسی (متوسطه). ۳. کامپیوتر - کتاب‌های درسی (متوسطه). الف. ایران. وزارت آموزش و پرورش. دفتر برنامه‌ریزی و تألیف آموزش‌های فنی و حرفه‌ای و کاردانش. ب. عنوان. ج. فروست.	

همکاران محترم و دانش‌آموزان عزیز :

پیشنهادات و نظرات خود را درباره محتوای این کتاب به نشانی
تهران - صندوق پستی شماره ۴۸۷۴/۱۵ دفتر برنامه‌ریزی و تألیف آموزش‌های
فنی و حرفه‌ای و کاردانش، ارسال فرمایند.

پیام‌نگار (ایمیل) info@tvoccd.sch.ir
وب‌گاه (وب‌سایت) www.tvoccd.sch.ir

محتوای این کتاب بر اساس تغییرات حوزه فن‌آوری و نظرات هنرآموزان و گروه‌های
آموزشی استانها توسط سرکار خانم مهندس زهرا عسگری رکن‌آبادی و سرکار خانم مهندس
مریم حسکوثیان زیر نظر کمیسیون تخصصی برنامه‌ریزی و تألیف کتابهای درسی رشته کامپیوتر
در سال ۱۳۹۰ مورد بازبینی و اصلاح قرار گرفته است.

وزارت آموزش و پرورش

سازمان پژوهش و برنامه‌ریزی آموزشی

برنامه‌ریزی محتوا و نظارت بر تألیف : دفتر برنامه‌ریزی و تألیف آموزش‌های فنی و حرفه‌ای و کاردانش

نام کتاب : برنامه سازی (۳) - ۴۵۱/۵

مؤلف : مهندس علیرضا جباریه

اعضای کمیسیون تخصصی : بتول عطاران، محمدرضا شکرریز، محمدرضا یمقانی، افشین اکبری، سیدسعید رضاسعادت یزدی،

شهناز علیزاده و ملیحه طبری

آماده‌سازی و نظارت بر چاپ و توزیع : اداره کل چاپ و توزیع کتاب‌های درسی

تهران : خیابان ایرانشهر شمالی - ساختمان شماره ۴ آموزش و پرورش (شهید موسوی)

تلفن : ۹-۸۸۸۳۱۱۶۱، دورنگار : ۰۹۲۶۶-۸۸۳، کد پستی : ۱۵۸۴۷۴۷۳۵۹.

وب‌سایت : www.chap.sch.ir

مدیر امور فنی و چاپ : سید احمد حسینی

طراح جلد : مریم کیوان

صفحه‌آرا : سمیه قنبری

حروفچین : زهرا ایمانی نصر

مصحح : شاداب ارشادی، معصومه صابری

امور آماده‌سازی خبر : فاطمه گیتی جبین

امور فنی رایانه‌ای : حمید نابت کلاچاهی، مریم دهقان‌زاده

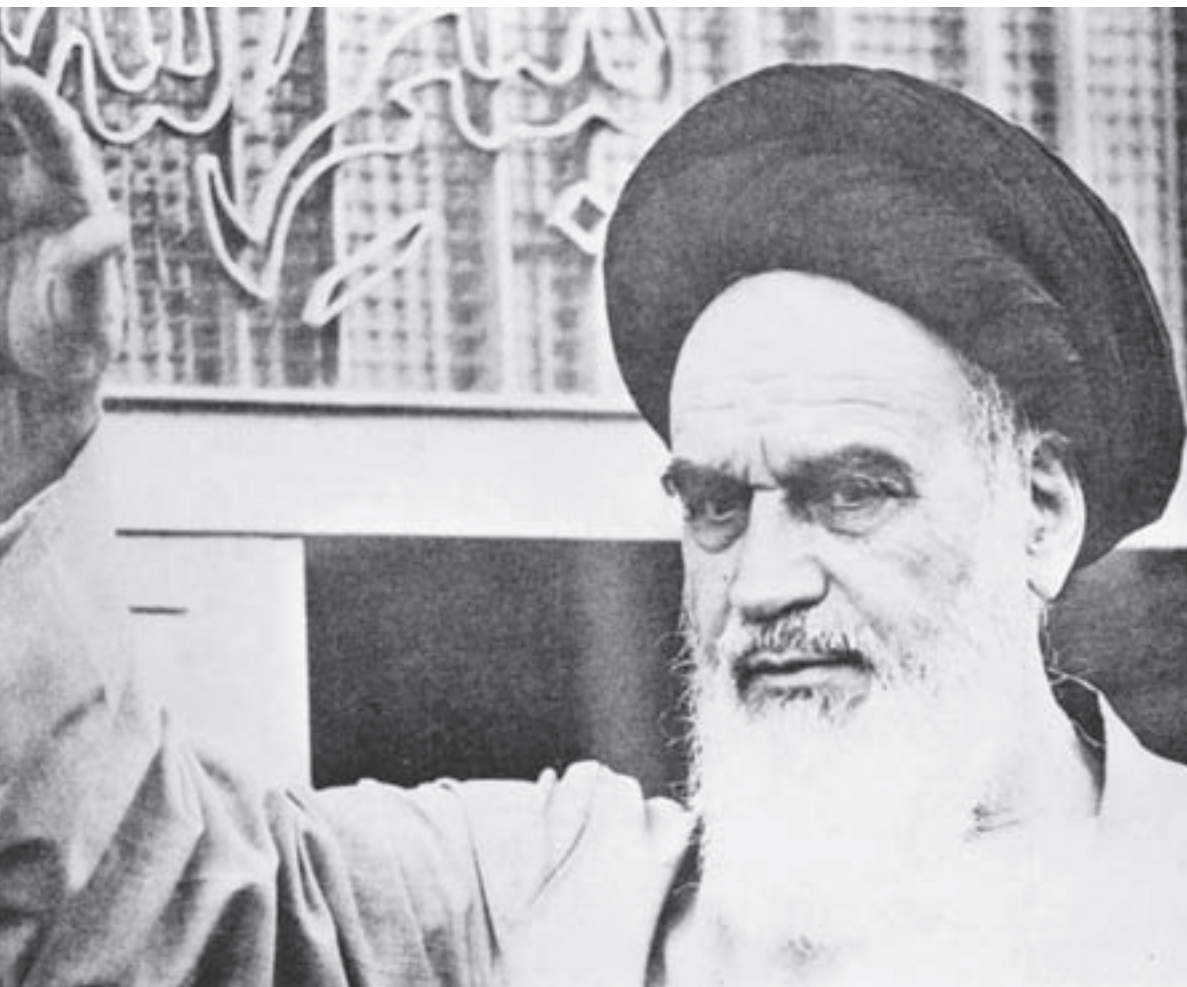
ناشر : شرکت چاپ و نشر کتاب‌های درسی ایران - تهران - کیلومتر ۱۷ جاده مخصوص کرج - خیابان ۶۱ (داروپخش)

تلفن : ۵-۴۴۹۸۵۱۶۱، دورنگار : ۴۴۹۸۵۱۶۰، صندوق پستی : ۳۷۵۱۵-۱۳۹

چاپخانه : شرکت چاپ و نشر کتاب‌های درسی ایران «سهامی خاص»

سال انتشار و نوبت چاپ : چاپ هفتم ۱۳۹۱

حق چاپ محفوظ است.



شما عزیزان کوشش کنید که از این وابستگی بیرون آید و احتیاجات کشور خودتان را برآورده سازید، از نیروی انسانی ایمانی خودتان غافل نباشید و از اتکای به اجانب بپرهیزید.

امام خمینی «قدس سره الشریف»

مقدمه

درس برنامه‌سازی (۳) را شاید بتوان محل جمع‌بندی آموخته‌های هنرجویان از درس‌های مختلف رشته کامپیوتر نامید. در این درس، سعی بر این است که هنرجو از آموخته‌های سایر درس‌ها مانند بسته‌های نرم‌افزاری، چند رسانه‌ای، بانک اطلاعاتی و به‌خصوص برنامه‌سازی ۱ و ۲ بهره‌برداری کند تا بتواند پروژه‌های مناسبی را با زبان ویژوال بیسیک تولید و اجرا کند.

شاید یکی از کاربردی‌ترین مواردی که هنرجو انتظار دارد در درس برنامه‌سازی انجام دهد، تولید یک برنامه کاربردی بانک اطلاعاتی است که در این کتاب پوشش داده می‌شود.

در فصل دوم کتاب، مفهوم پرونده و پیاده‌سازی آن به کمک زبان برنامه‌نویسی ویژوال بیسیک مطرح شده است تا هنرجو بتواند آن را با مطالبی که در کتاب بانک اطلاعاتی مطالعه می‌کند، مطابقت دهد و مزایا و معایب بیان شده در مبحث فایلینگ، در آن کتاب را لمس کند.

در فصل سوم سعی شده است که مقدمات روش برنامه‌نویسی شیء‌گرا مطرح شود تا هنرجویان علاقمند بتوانند با آشنایی ابتدایی با این روش برنامه‌نویسی، آن را در دوره کاردانی دنبال کنند و در کنار آموخته‌های خود در روش ساخت یافته از این روش نیز برای تولید برنامه‌های کاربردی موردنیاز بهره‌گیرند.

در فصل‌های چهارم و پنجم، چگونگی تولید برنامه‌های کاربردی بانک اطلاعاتی مطرح شده است که هنرجویان با فراگیری این مطالب و با کمک هنرآموزان محترم خواهند توانست پروژه‌های کاربردی مناسبی را تولید و ارائه کنند.

در فصل ششم، چگونگی ارتباط ویژوال بیسیک با نرم‌افزارهای مجموعه آفیس مطرح شده است تا هنرجویان بتوانند با آموخته‌های خود در کتاب‌های بسته‌های نرم‌افزاری ۱ و ۲ ارتباط برقرار کنند.

در فصل هفتم، برنامه‌نویسی سیستمی با توابع API مطرح شده است که به دلیل تنوع زیاد این توابع، ذکر همه آن‌ها در این کتاب مقدور نبود و هنرجویان علاقمند می‌توانند به کتاب‌های مرجع مراجعه کنند.

شایسته است از کلیه استادان محترم دانشگاه‌ها و هنرآموزان گرامی سطح کشور که از راهنمایی‌های خود ما را بی‌نصیب نگذاشتند، سپاسگزاریم. این کتاب با نظرات این بزرگواران اصلاح گردیده و آماده شده است که امیدوارم این اثر مورد قبول واقع شود.

مؤلف

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.

۱-۱ - 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 برنامه 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 از نوع 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 استفاده مجدد را دارند فراهم می‌کند.

روال‌ها امکان تغییر ساده کد را فراهم می‌کنند. اگر نیاز به استفاده 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.

۱-۲- ایجاد و فراخوانی یک Sub ساده

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

```
[private|public] sub SubName()
```

خطوطی از کد ...

```
End Sub
```

- [private|public] کلید واژه‌های اختیاری هستند که حوزه 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 رشته 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 کلیدواژه 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
```

۱- حوزه عمل در ادامه همین فصل توضیح داده شده است.

۱-۳- 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 گزینه Add Procedure منوی Tools.

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

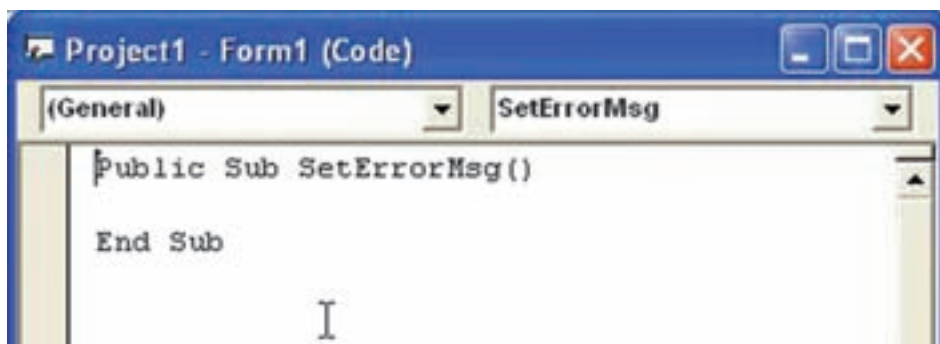
- ۱- مراحل اضافه کردن sub به پروژه با روش دوم، به صورت زیر است :
- ۱- از منوی Tools گزینه Add Procedure را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود.
- ۲- نامی را برای sub وارد کنید (شکل ۱-۱).
- ۳- Sub را از گزینه‌های Type انتخاب کنید.
- ۴- از گزینه‌های Scope نوع حوزه عمل sub را انتخاب کنید.



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

- ۵- روی Ok کلیک کرده و بلاک کد را به فرم یا مدول اضافه کنید (شکل ۱-۲).

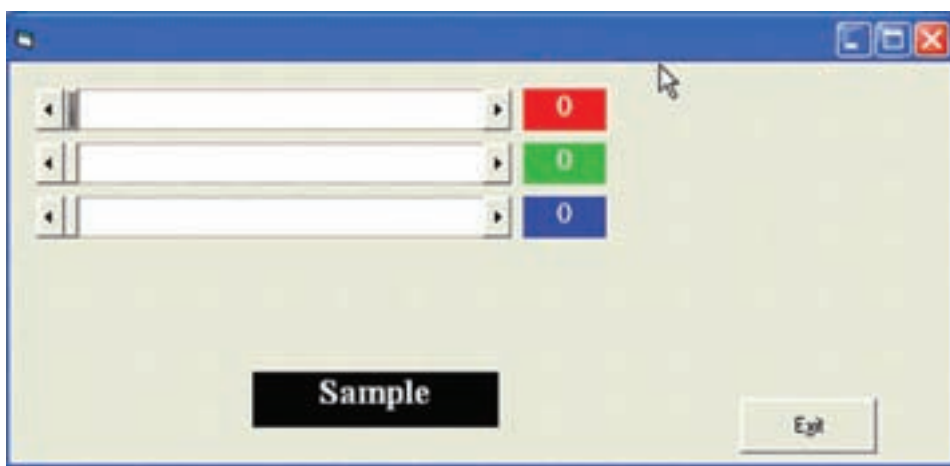
بعد از این که بلاک کد را با کادر محاوره‌ای Add Procedure ایجاد کردید، کد روال را بین اعلان sub و کلید واژه End sub اضافه کنید. بعد از End Sub کدی را وارد نکنید، انجام این کار سبب بروز خطا در زمان کامپایل خواهد شد.



شکل ۱-۲ sub جدیدی را در بخش General فرم یا مدول به دست خواهید آورد.



فرمی به صورت شکل ۱-۳ ایجاد نمایید که با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی، سبز بتوان رنگ برجسب را تغییر داد.



شکل ۱-۳


```
Private Sub setcolor()  
    r = HScroll1.Value  
    g = HScroll2.Value  
    b = HScroll3.Value  
    Lblr.Caption = r  
    Lbly.Caption = g  
    Lblb.Caption = b  
    Labell.BackColor = RGB(r,g,b)  
End Sub  
  
Private Sub Command1_Click()  
    End  
End Sub  
  
Private Sub Form_Load()  
    Labell.BackColor = RGB(0,0,0)  
End Sub  
  
Private Sub HScroll1_Change()  
    Call setcolor  
End Sub  
  
Private Sub HScroll2_Change()  
    Call setcolor  
End Sub  
  
Private Sub HScroll3_Change()  
    Call setcolor  
End Sub
```

۴-۱- 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 حوزه 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 را نشان 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
```

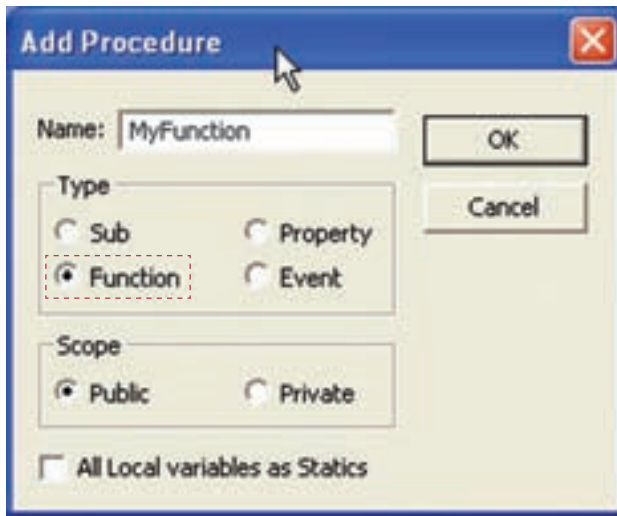
```
b = 12
```

```
c = a+b
```

```
GetNumber = c
```

```
End Function
```

تابع را نیز می‌توان مانند Sub با همان دو روش به فرم یا مدول اضافه کرد (شکل ۴-۱).



شکل ۱-۴- تابع را در کادر محاوره‌ای Add Procedure اضافه کنید.

۱-۵- 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 پراتزهای 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 در یک برنامه نیاز دارید که بزرگ‌ترین مقدار بین دو عدد را به دست 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 می دهد :

```

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. به عنوان مثال، اگر تابعی به نام EndDay() دارید که به صورت زیر اعلان می شود :

```

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

```

و تابع را با استفاده از کد زیر فراخوانی می کنید،

```

dMyResult = EndDay (6,"056R")

```

این فراخوانی، خطایی را تولید می کند. "056R" از نوع رشته ای است ولی تابع برای آرگومان دوم انتظار داده ای از نوع Double را دارد.

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

به صورت 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 می کنید :

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

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

می توان آرگومانی را به این منظور از کلید واژه Optional، در اعلان تابع قبل از آرگومان مورد نظر استفاده کرد. آرگومان های اختیاری باید از نوع Variant باشند.

۱-۵-۱- کاربرد آرگومان های نام دار: هنگام فراخوانی روال ها می توان از آرگومان های نام دار

برای ارسال ساده تر مقادیر به آنها استفاده کرد. به عنوان مثال، اگر تابعی به نام GetGreaterNum دارای دو آرگومان از نوع Integer باشد و به صورت زیر تعریف شود :

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

هنگام فراخوانی این تابع بعد از اسامی آرگومان ها از نویسه های = : استفاده کرده و مقداری را برای آنها به صورت زیر تعیین کنید.

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

۱-۶- خروج از روال ها

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

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

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.

۱-۷- آشنایی با حوزه عمل

حوزه عمل (میدان دید)، قابلیت است که دو متغیر مختلف با اسامی یکسان می‌توانند مقادیر مختلفی را نگهداری کنند و دارای دوره حیات متفاوتی هستند. کد زیر، دو تابع `GetNumber()` و `Bar()` را نشان می‌دهد:

حوزه عمل (میدان دید)، محدوده 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 همان جایی که ایجاد شده‌اند به کار 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 حوزه 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.



```
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.

۸-۱- مستندسازی روال‌ها

مستندسازی روال‌ها به سایر برنامه‌نویسان امکان می‌دهد که به‌طور کامل از برنامه شما استفاده کرده و در صورت 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 ارائه می‌دهد.

۹-۱- تعیین نقطه ورودی با SubMain()

به طور پیش فرض، 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.

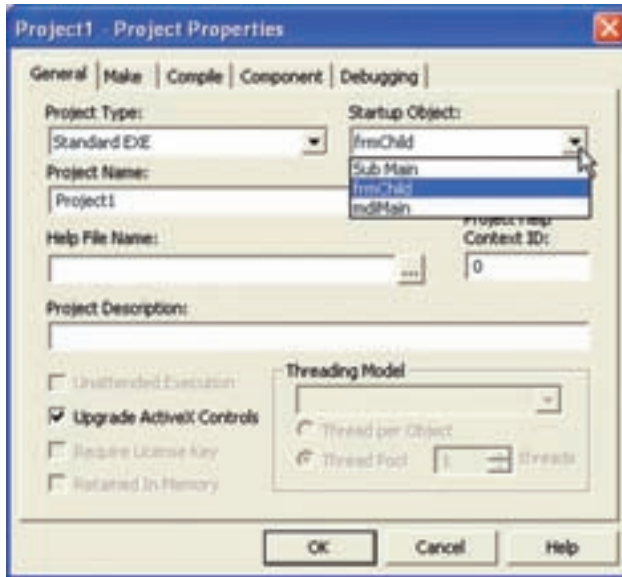
در پروژه‌هایی که دارای هیچ فرمی نیستند (مثل برنامه‌های اینترنتی که در روی سرور کار می‌کنند) و چیزی برای بارگذاری وجود ندارد چه کاری باید برای نقطه شروع (نقطه ورودی) برنامه انجام دهید؟

ویژوال بیسیک، یک نقطه شروع غیر مبتنی بر فرم را برای برنامه ارایه می‌کند (روال Sub Main()). Sub Main() روال خاصی است که به وسیله 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() در نظر گرفت. مراحل زیر، چگونگی تعیین این روال به عنوان نقطه شروع را نشان می‌دهند :

۱- از منوی Project گزینه Properties مربوط به پروژه جاری را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود.

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

۳- روی Ok کلیک کنید.



شکل ۱-۵-۱ 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 این که همه فرم‌ها ظاهر شدند، کادر پیامی را نشان می‌دهد.

Sub Main()

frmMain.Show

frmOther.Show

MsgBox "Everything shown"

End Sub

۱-۱۰-۱- توابع تشخیص نوع داده

این توابع نوع داده 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 آن که محاسبه‌ای انجام دهد باید مطمئن شود که داده‌ها از نوع عددی هستند. در جدول ۱-۱ توابع Is....() را مشاهده می‌کنید؛ آرگومان این توابع همگی از نوع Variant است.

جدول ۱-۱- 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 هر یک از توابع مثالی ذکر شده است.
در برنامه 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(var1) 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

```

برنامه 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 راه حل 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 این جا برنامه قبل از ادامه کار، خالی نبودن یکی از فیلدهای برنامه (txtHoursWorked)

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 ۱-۲

مقادیر برگشتی این تابع را نشان می‌دهد.

جدول ۱-۲- مقادیر برگشتی تابع VarType()

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

در برنامه زیر، با دستور Select Case نوع داده 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

۱-۱-۱- توابع تبدیل نوع


در جدول زیر، توابع تبدیل نوع را مشاهده می‌کنید؛ به حرف C (سرنام کلمه 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 ۱-۳- 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.

۱-۱۲- تابع Array

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

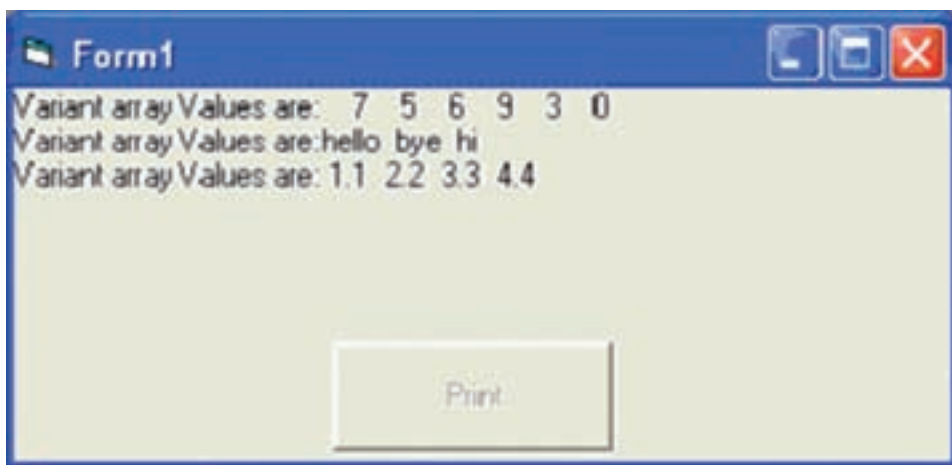
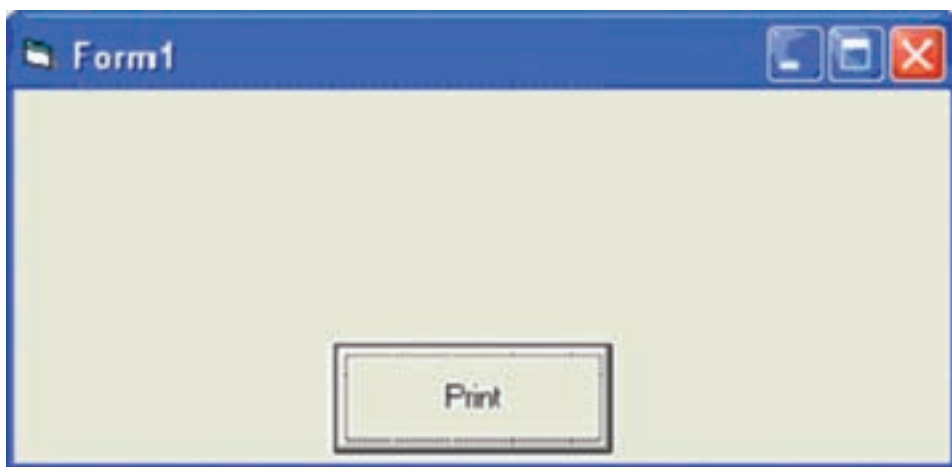
1. 'Demonstrating function Array
2. Option Explicit 'General declaration
3. Option Base 1 'General declaration


```

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 برنامه فوق آرایه V با سه نوع داده 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 ۶-۱- استفاده از تابع Array()

خودآزمایی

- ۱- برنامه‌ای بنویسید که دو عدد را دریافت کند و با استفاده از یک تابع، عدد کوچک را به توان عدد بزرگ برساند.
- ۲- برنامه‌ای بنویسید که عددی را دریافت کند و فاکتوریل آن را به کمک یک تابع محاسبه کند و نمایش دهد.
- ۳- برنامه‌ای بنویسید که عددی چندرقمی را دریافت کند و با استفاده از یک روال، مجموع ارقام آن عدد را محاسبه کند و نمایش دهد.
- ۴- برنامه‌ای بنویسید که مضارب معادله درجه ۲ را دریافت کند و با استفاده از یک روال، ریشه‌های آن را به دست آورده و نمایش دهد.
- ۵- برنامه‌ای بنویسید که عددی را دریافت کند و با استفاده از یک روال، مقسوم‌علیه‌های آن را به دست آورد و نمایش دهد.
- ۶- برنامه‌ای بنویسید که رشته‌ای را دریافت کند و تعداد فضاهای خالی آن را به کمک یک روال، شمارش کند.

۲

فصل

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 شماره 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 به کار 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.

۲-۱- آشنایی با مفهوم پایداری

برای اینکه برنامه‌ای بتواند اطلاعات را از جلسه‌ای به جلسه‌ای دیگری نگه دارد، باید توانایی ذخیره‌سازی داده‌ها روی دیسک سخت را داشته باشد. در غیر این صورت، هنگامی که برنامه 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.

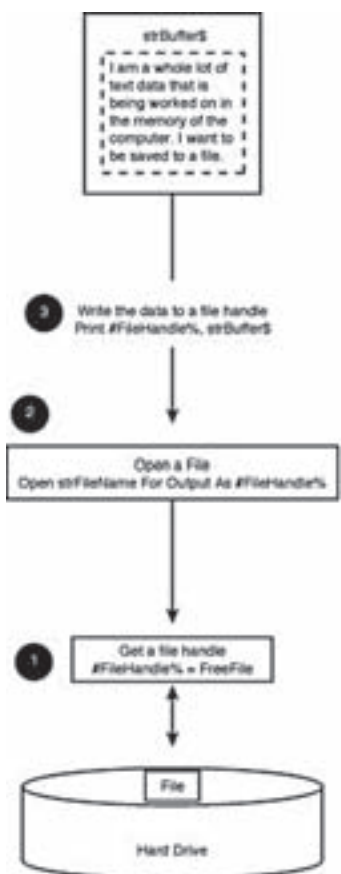
می توان با چندین روش، عملیات ذخیره و بازیابی داده ها را انجام داد. از پرونده دودویی یا متنی می توان برای ذخیره اطلاعات با اندازه و قالب متفاوت استفاده کرد.

۲-۲- کار کردن با پرونده ها برای ذخیره و بازیابی داده ها

داده ها در حافظه اصلی و پرونده ها روی حافظه جانبی (دیسک ها) ذخیره می شوند. هرگز برنامه ها به طور مستقیم با یک پرونده روی دیسک کار نمی کنند.

برنامه از سیستم عامل می خواهد که واسطی بین دیسک و برنامه ایجاد کند (به عبارت دیگر، داده های مورد نیاز برنامه به کمک پردازنده و سیستم عامل به حافظه اصلی بارگذاری می شوند و سپس برنامه اجرا می شود).

موقعیت (محل) یک پرونده روی دیسک را می توان با به دست آوردن شماره پرونده (file handle) از طریق سیستم عامل، پیدا کرد. از تابع FreeFile برای به دست آوردن شماره پرونده از طریق سیستم عامل استفاده می شود. بعد از به دست آوردن این شماره، دستور Open برای دسترسی به پرونده برای عملیات خواندن و نوشتن به کار می رود. برای نوشتن در یک پرونده، می توان از دستور Print (یا Write) و برای خواندن خطوطی از داده ها از پرونده روی دیسک، از دستور Line Input استفاده می شود. شکل ۲-۱ این مفهوم را شرح می دهد.



شکل ۲-۱- 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)^۱. 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 نقطه 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.

۲-۳- 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 ۲-۱).

جدول ۲-۱- 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.

۱- مبحث فایل‌های دودویی خارج از محدوده 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 (پیش فرض 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 شماره پرونده بعد از آن ارایه می‌شود.
- *FileNumber* شماره پرونده (file handle) است.
- Len کلید واژه اختیاری است که قبل از پارامتر طول رکورد قرار می‌گیرد.
- *%CharInBuffer* یک گزینه اختیاری است که طول رکورد پرونده‌ای که برای دسترسی تصادفی باز شده است را تعیین می‌کند.

۱-۳-۲- تخصیص شماره پرونده: در Visual Basic این امکان وجود دارد که به طور همزمان چندین پرونده باز داشته باشید، مشروط بر اینکه هر پرونده شماره خاص خود را داشته باشد. به این منظور همیشه باید شماره پرونده‌های باز را بدانید و این کار بسیار مشکلی است. اما Visual Basic برای این مشکل راه حلی دارد: تابع *FreeFile()*. این تابع، اولین شماره پرونده آزاد را برمی‌گرداند. با استفاده از این تابع همواره اطمینان خواهید داشت که شماره پرونده‌ها تکراری نخواهند بود. شکل استفاده از تابع *FreeFile()* چنین است:

`FreeFile [(intRangeNumber)]`

پارامتر اختیاری *intRangeNumber* اجازه می‌دهد تا محدوده شماره پرونده را (1-255 یا 256-511) مشخص کنید. محدوده پیش فرض 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 پراترها هم ضروری نیست. در دستوره‌های زیر، ابتدا یک شماره 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

شماره 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 شماره پرونده باز شده را بدانید، وجود ندارد.

۲-۳-۲- طول رکورد: آرگومان `Len = CharInBuffer` هنگام کار با پرونده‌های تصادفی

اهمیت می‌یابد. هنگامی که ویژگی 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 شماره رکوردها هم از ۱ شروع خواهد شد.

۲-۴- 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 تک تک پرونده‌ها) آن است که در پایان برنامه با دستور ساده زیر تمام پرونده‌های باز را به یکباره ببندید :

```
Close
```


۲-۵- کار با پرونده‌های ترتیبی

حال که با کلیات پرونده (مانند باز کردن، بستن) آشنا شدید، بهتر است با چند مثال خواندن - 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. مثلاً اگر 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.

۲-۵-۱- 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.

۲-۵-۱-۱- 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` شماره 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 ۲-۲ توضیحات بیشتری درباره عناصر `OutputList` ملاحظه می‌کنید.

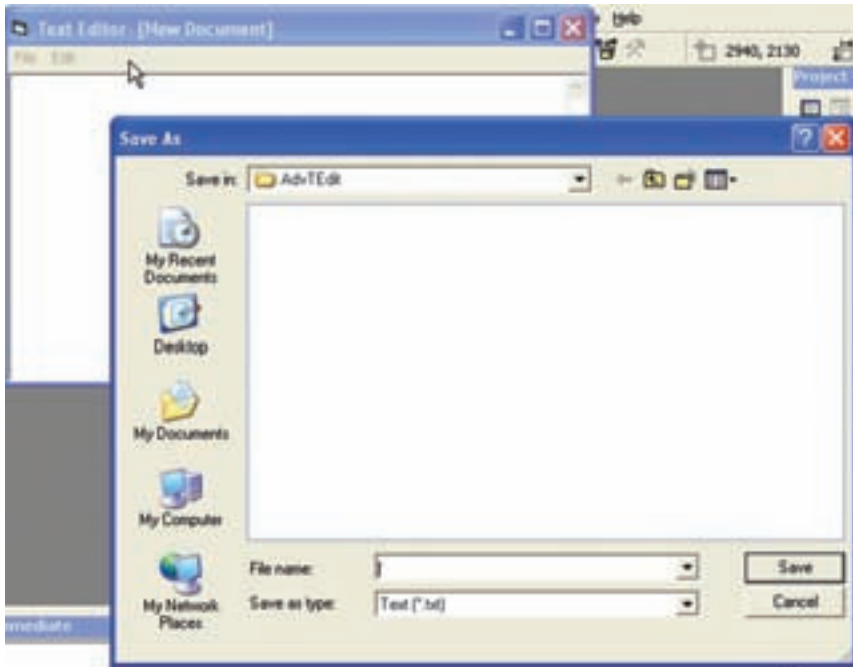
(تابع `Tab(intN۲)` مکان‌نما را به ستون `intN۲` منتقل می‌کند)

جدول ۲-۲. عناصر لیست خروج دستور # Print

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



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 باز کردن پرونده خواهد بود. برای استفاده از برنامه، هنگام ذخیره داده‌ها، روی گزینه 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 دکمه Save در کادر محاوره‌ای کلیک کنید تا داده‌ها ذخیره شوند (شکل ۲-۲). 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 TextBox ی با نام textMain استفاده کنید و مشخصه MultiLine آن را مساوی True قرار دهید.



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

کد زیر، روال رویداد Click() گزینه 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 شماره 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 Common Dialog تنظیم مشخصه filter برای کنترل

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 در صورت خالی نبودن filename strfilename قرار می‌دهیم

18 strFileName = cdMain.filename

19

20 قرار دادن محتوای Main txt در strText

21 strText = txtMain.Text

22

23

به دست آوردن شماره پرونده برای فایل

24

25 FileHandle% = FreeFile

26

27 باز کردن فایل برای نوشتن

28 Open strFileName For Output As # FileHandle%

29

30

تغییر اشاره‌گر ماوس به صورت ساعت شنی

31 MousePointer = vbHourglass

32

33 نوشتن در فایل

34 Print # FileHandle%, strText

35

36 برگرداندن اشاره گر ماوس به حالت عادی

37 MousePointer = vbDefault

38


39 بستن فایل

40 Close # FileHandle%

41 End If

42

43 End Sub

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

 مثال ۲-۲

در کد زیر، روالی را مشاهده می کنید که پرونده Print.txt را باز کرده، اعداد ۱ تا ۶ را در آن پرونده می نویسد و سپس پرونده را می بندد.

```
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

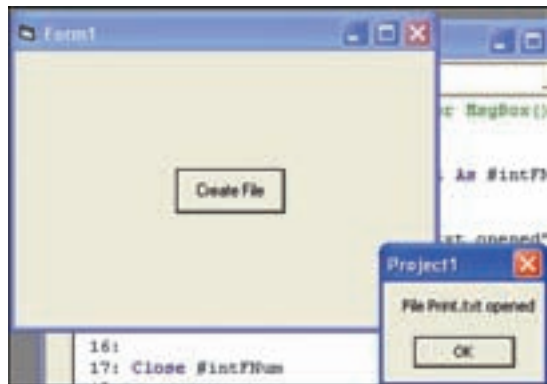
Close #intFNum

MsgBox "File Print.txt closed"

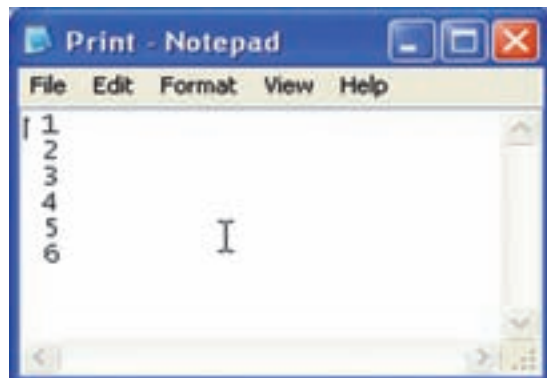
End Sub

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 ۲-۳



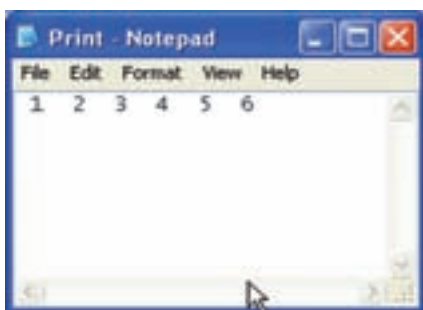
شکل ۲-۴

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; '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



شکل ۲-۵

با توجه به مثال ۲-۲ و ۲-۳ مشاهده می‌شود که اگر در انتهای دستورات Print # از علامت ; (سمیکolon) استفاده کنیم عملیات نوشتن در همان خط جاری ادامه می‌یابد و در غیر این صورت مکان نما به خط بعد منتقل می‌شود.

۲-۵-۱-۲- 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 وسیله خطای رخ داده است.

۲-۵-۲- خواندن پرونده‌های ترتیبی: بازبایی داده‌ها در دیسک خیلی شبیه نوشتن

داده‌ها 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.

۲-۵-۲-۱- دستور **Input#**: بعد از نوشتن اطلاعات در پرونده، باید بتوانید آن‌ها را دوباره

بازبایی 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 شماره 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 داده 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 وسیله 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 دکمه CmdFileOut برای عملیات نوشتن و از روال رویداد Click دکمه CmdFileIn برای عملیات خواندن پرونده استفاده شده است.

```
1: Private Sub cmdFileOut_Click()  
2:     'Create the sequential file  
3:     Dim intCtr As Integer    'Loop counter  
4:     Dim intFNum As Integer  'File number  
5:     intFNum = FreeFile     `به دست 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     `به دست آوردن شماره پرونده`  
23: Open "Print.txt" For Input As # intFNum آن باز کردن پرونده به منظور 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 Immediate 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 در نظر 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.)

۲-۵-۲- بازبازی 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 شماره پرونده باز است.
 - StrBuffer رشته‌ای است که دستور، داده‌های بازبازی 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 دهید، یک خط وارد کرده‌اید. هر بار که کلید Enter را فشار دهید، ویژوال بیسیک رشته 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 حلقه Do While استفاده کنید. از تابع EOF() و ویژوال بیسیک برای تعیین اینکه آیا به انتهای پرونده رسیده‌ایم، استفاده کنید. این تابع، شماره پرونده را به عنوان آرگومان دریافت می‌کند. تا زمانی که به انتهای پرونده نرسیده‌اید، دستور Line Input به خواندن خطوط پرونده در داخل حلقه Do While ادامه می‌دهد.



کد زیر، روال رویداد مربوط به گزینه Open از منوی File پرونده مثال ۲-۱ 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 مشخصه filename در کادر محاوره‌ای  
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 آزاد شماره 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 اشاره گر ماوس به ساعت شنی

29 MousePointer = vbHourglass

30

31 `Traverse the lines of the file خواندن فایل به صورت 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 ذخیره آن در strBuffer

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 رشته strText

39 Loop

40

41 `Change the mousepointer back to the arrow

تغییر اشاره گر 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 &"]"

در عنوان فرم

52 End If

53 End Sub

۲-۶- کار با پرونده‌های تصادفی

با آن که درباره 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 اولیه اهمیت پرونده‌های تصادفی، معرفی نوع جدیدی از انواع داده است: نوع داده تعریف شده به وسیله کاربر (user-defined data type). این نوع داده، همانطور که از نام آن پیداست، برخلاف انواع داده ذاتی Visual Basic به وسیله کاربر تعریف می‌شود.

۱-۲-۶- نوع داده تعریف شده به وسیله کاربر: در مطالبی که تاکنون بیان کرده‌ایم از انواع

داده‌های ذاتی ویزوال بیسیک مثل Integer، String و Double استفاده کرده‌ایم. یکی از ویژگی‌های ویزوال بیسیک، فراهم نمودن امکان ایجاد نوع داده سفارشی است که به نام نوع داده تعریف شده به وسیله 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

05 Public g_PieceTwo As String

اعلان متغیر برای نام قطعه ۲

06

07 Public g_ComposerThree As String

اعلان متغیر برای آهنگساز قطعه ۳

08 Public g_PieceThree As String

اعلان متغیر برای نام قطعه ۳

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 راه حل ساده برای این مسأله، تعریف یک بسته متغیری و نامگذاری آن است. به بیان دیگر با ترکیب داده‌های ذاتی Visual Basic می‌توانید انواع داده‌های جدیدی ایجاد کنید. این بسته را نوع داده تعریف شده به وسیله 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 داده کاربر، از دستور Type استفاده می‌شود. هر نوع داده 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 واژه Type در بخش General مدول ایجاد کرد:

```
Type TypeName
```

```
    Elements as DataType
```

```
    .....
```

```
End type
```

در این دستور:

- **Type**: کلید واژه ویزوال 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 به صورت 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 داده تعریف شده به وسیله 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 داده Music که از اجزای قطعه 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 داده جدیدی ایجاد کنیم و از آن به بعد با این نوع داده کار کنیم.

از طریق کد زیر، می‌توانید مطالب بیشتری درباره 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 داده جدید دارای دو متغیر از نوع رشته‌ای (str FName و str LName) 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 داده کاربر یک فیلد (field) گفته می‌شود. متغیرهای رشته‌ای درون نوع داده کاربر را می‌توان با گزینه *StringLenght بعد از String As محدود کرد. بدین ترتیب طول رشته با مقدار ثابت StringLenght مقداردهی خواهد شد. هنگام نوشتن نوع داده کاربر در یک فایل (به دلیل این که اندازه هر رکورد باید مشخص باشد) فیلدهای رشته‌ای باید اندازه ثابت داشته باشند. در کد زیر، شکل تغییر یافته 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 رشته 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 بقیه رشته را با فضای خالی پر خواهد کرد.

۲-۶-۲- باز کردن پرونده‌های تصادفی: قبلاً دیدیم که در حالت پیش فرض، 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.


۳-۶-۲- دستوره‌های Put و Get: 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 دستوره‌های Input# و Print# راهی وجود ندارد تا نقطه‌ای از پرونده برای خواندن یا نوشتن مشخص شود، در حالی که دستوره‌های Get# و Put# دارای چنین امکانی هستند :

Put [#]intFileNum, [intRecNum,] Variable

Get [#]intFileNum, [intRecNum,] Variable

● intFileNum شماره پرونده مورد نظر است.

● intRecNum شماره رکوردی است که می‌خواهید با آن کار کنید.

 **نکته:** در صورتی که شماره رکورد در این دستورها ذکر نشود، عملیات روی رکورد بعد از رکورد جاری انجام می‌شود.

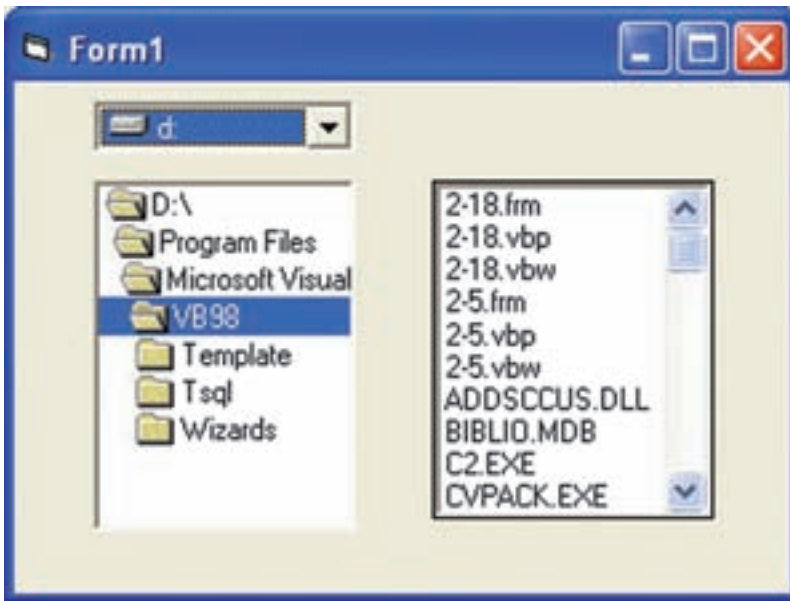
● Variable متغیری است که مقدار آن در پرونده نوشته می‌شود و یا مقداری که از پرونده خوانده شده، در آن ذخیره می‌شود.

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

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

۲-۷- کنترل‌های FileListBox، DirectoryListBox و DriveListBox

- ویژوال بیسیک دارای سه کنترل خاص برای مدیریت پوشه‌ها، درایوها و پرونده‌هاست:
- کادر لیست دایرکتوری - به کاربر امکان باز کردن پوشه (دایرکتوری)ها را می‌دهد.
 - کادر لیست درایو - به کاربر امکان انتخاب یک درایو دیسک را می‌دهد.
 - کادر لیست پرونده - به کاربر امکان انتخاب پرونده را می‌دهد.
- شکل ۲-۶ فرمی را با سه کنترل مزبور نشان می‌دهد.



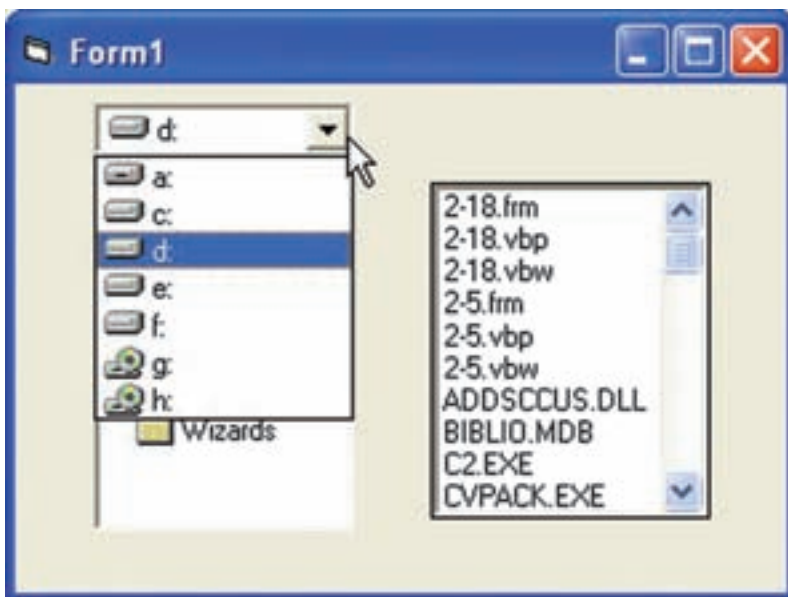
شکل ۲-۶- کنترل‌های پرونده Visual Basic

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 وظیفه برنامه‌نویس است که با نوشتن کد بین آن‌ها ارتباط برقرار کند.

۱-۲-۷- 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 ۲-۷ یک کادر لیست درایو باز را نشان می‌دهد.



شکل ۲-۷- 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.

مشخصه 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 پیش‌فرض کنترل drvDisk را به درایو C تغییر می‌دهد.

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

در صورتی 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 مشخصه درایو کنترل قرار دارد (drvDisk.drive) شکل کلی دستور chDrive در جدول ۲-۴ آمده است.

۲-۷-۲- کادر لیست دایرکتوری : با کادر لیست دایرکتوری (DirectoryListBox)، کاربر

امکان انتخاب پوشه مورد نظر 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 پیش فرض کادر لیست دایرکتوری، پوشه‌ای است که برنامه از آنجا اجرا شده است، ولی این وضعیت را می‌توان تغییر داد.

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

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

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

در صورتی که کاربر مسیر دیگری را از کنترل انتخاب کند رویداد Change رخ میدهد. مسیر انتخاب شده توسط کاربر در مشخصه path قرار دارد (dirDirect.path). برای تغییر مسیر از دستور ChDir استفاده کنید (شکل کلی این دستور در جدول ۲-۴ آمده است).

```
private sub dirDirect-Change ()
```

```
ChDir dirDirect.path
```

```
End sub
```

۲-۷-۳- کادر لیست پرونده : به کمک کادر لیست پرونده (FileListBox) کاربر می‌تواند

پرونده مورد نظرش 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 خودی خود قادر به تشخیص درایو و پوشه انتخاب شده نیست. کادر لیست پرونده

به طور پیش فرض پرونده‌های موجود در پوشه‌ای 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 ۲-۳ ارایه شده‌اند.

جدول ۲-۳

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 مشخصه 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.)

۲-۸-۲- دستورهای پرونده

ویژوال بیسیک دارای چندین دستور برای کار با پرونده‌ها، پوشه‌ها و درایوها است. این دستورات را در جدول ۲-۴ ملاحظه می‌کنید.

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

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

علاوه بر دستورهای جدول ۲-۴، Visual Basic از توابع Dir() (برای تعیین وجود یا عدم وجود پرونده‌ها) و CurDir() (برای تعیین نام دایرکتوری جاری) نیز پشتیبانی می‌کند. تابع Dir() به توضیح بیشتری نیاز دارد. فرض کنید می‌خواهید بدانید آیا پرونده‌ای به نام SALES98.DAT در درایو C وجود دارد یا خیر؟ برای این کار می‌توانید از کد زیر استفاده کنید:

جدول ۲-۴- دستورات پرونده Visual Basic

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

```
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 در صورتی که پرونده موجود نباشد، تابع Dir() چیزی برنمی‌گرداند. آرگومان تابع Dir() (مانند اکثر توابع پرونده) می‌تواند از کاراکترهای عمومی (Wildcard) استفاده کند:

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


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 رشته null ("") برگشت داده شود، نخواهد 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
```

در صورتی 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 Change() درایو 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 به کار 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 مشخصه جالبی به نام ListIndex 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 مشخصه 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 برجسب مشاهده کرد.

تذکره: 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 مشخصه Picture کنترل کادر تصویر یا تصویر قرار دهید.

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

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

۱- کنترل لیست درایو از نوع Combo Box و دو کنترل دیگر از نوع listBox هستند. لذا هر سه دارای مشخصه List و

ListIndex می‌باشند.

در این شکل کلی تابع :

● ImageCtrl : کنترل کادر تصویر، تصویر یا فرم است.

● Picture : مشخصه Picture شیء است.

● LoadPicture : نام تابع است.

● FilePath : محل دقیق پرونده روی دیسک است.

برای ذخیره 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 مشخصه 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.



در این مثال از دستور SavePicture برای ذخیره 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 ذخیره آن استفاده می‌کند.

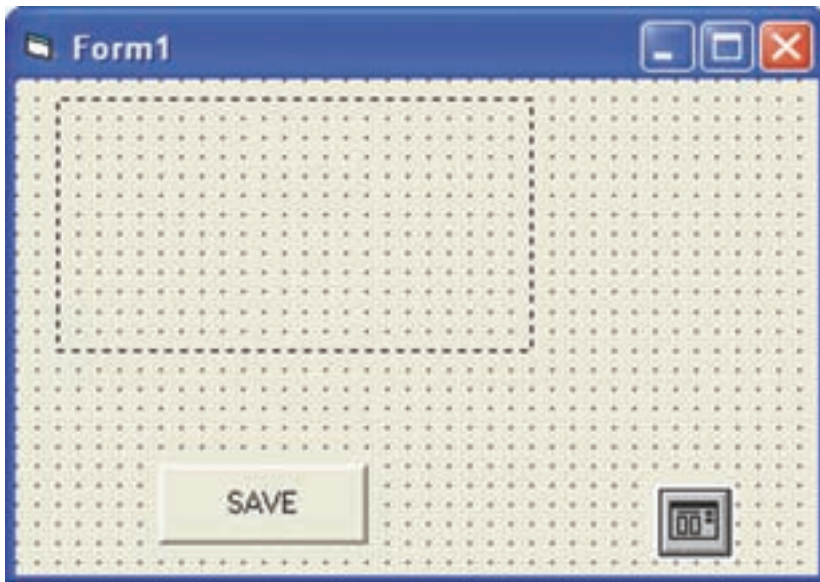
۱- پروژه جدیدی ایجاد کنید.

۲- روی Form1 کنترل‌های زیر را اضافه کنید :

● کنترل Image به نام imgMain

● دکمه فرمان به نام CmdImageSave

● CommandDialog به نام cdMain



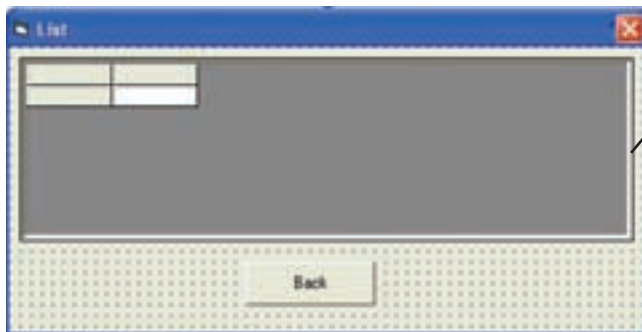
شکل ۲-۸

```
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
```

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



شکل ۲-۹

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 داده 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 فایل‌ها با نام tel.dat در مسیر Application در نظر 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 'در صورت 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 ذخیره رکورد جدید'
```

```

StrPerson.name1 = 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 دکمه 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 رشته دریافت شده حذف می‌شود، سپس مقادیر دریافت شده به ترتیب در عناصر Famil 1, name1 و tel 1 رکورد Strperson قرار داده شده و در فایل StrFs ذخیره می‌شود. هنگام ذخیره رکورد در فایل ابتدا تعداد رکوردهای موجود در فایل را محاسبه می‌کند، برای محاسبه تعداد رکوردها به کمک تابع Dir وجود فایل StrFs بررسی می‌شود در صورت وجود فایل از تقسیم طول فایل بر طول یک رکورد تعداد رکوردها محاسبه می‌شود و در صورتی که فایل StrFs وجود نداشته باشد تعداد رکوردها را صفر قرار می‌دهد. از تعداد رکوردهای ذخیره شده در فایل برای تعیین محل ذخیره رکورد جدید استفاده می‌شود. برای انتقال اشاره گر فایل به محل ذخیره 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")))
```

```
StrOldFamill = 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 + ".tmp"
```

```
        Open StrFs For Random As #1 Len = Len(StrPerson)
```

```
        If Dir (StrFt) <> "" Then Kill StrFt     ' حذف فایل موقت در صورت 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) = StrOldFamill And
```

```
                    Trim(StrPerson.namel) = StrOldName) Then
```

```
                        Put #2,, StrPerson     ' نوشتن رکورد به فایل موقتی
```

```
                        Else
```

```
                            BlnResult = True     ' نشانه 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 "Erasing Information Successfully"
```

```
        Else
```



```

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

```

رویداد Click دکمه 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 Trim و Ucase به ترتیب به حروف بزرگ تبدیل شده و فضای خالی ابتدا و انتهای رشته دریافتی حذف می‌شود. در صورت تأکید کاربر به حذف رکورد (کلیک دکمه 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) 'محاسبه 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.name1 = StrNewname
      StrPerson.tel1 = StrNewTel
      Put#1, LngI, StrPerson جایگزین کردن رکورد تغییر کرده '
      lblfamil.Caption = StrPerson.famill' نمایش عناصر رکورد تغییر کرده روی label ها
      lblname.Caption = StrPerson.name1
      lbltel.Caption = Strperson.tel
      BlnResult = True نشانه 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 If

End Sub

رویداد Click دکمه 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 همان محل رکورد قبلی نوشته می‌شود لذا رکورد قبلی حذف شده و رکورد جدید جایگزین می‌شود Put #1 و Lng I و Strperson) در این دستور Lng شماره 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 'LngI خواندن رکورد شماره'

مقداردهی مشخصه Caption برچسبها در صورتی که رکورد خوانده شده رکورد مورد جستجو باشد.

If Trim(StrPerson.famil) = StrSearchFamil Then

 lblfamil.Caption = StrPerson.famil

 lblname.Caption = StrPerson.name1

 lbltel.Caption = StrPerson.tell

End If

Next

Close #1

End Sub

در رویداد Click دکمه CmdSearch نام خانوادگی فردی که می‌خواهد اطلاعات آن را در دفترچه تلفن جستجو کند از کاربر دریافت کرده و آن را به کمک توابع Trim و Ucase 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 در صورت مساوی بودن محتوای فیلدهای آن رکورد را به وسیله 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 msflexgrid کنترل مشخصه‌های

.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 (۱ و ۰)'

.Col = 2

.Text = "Famil" نوشتن عبارت Famil در خانه شماره (۲ و ۰)'

.Col = 3

.Text = "Tel" نوشتن عبارت Tel در خانه شماره (۳ و 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 (LngI و 1) Col = 1
.Text = Trim(StrPerson . famill)
نوشتن محتوای فیلد name1 رکورد خوانده شده در خانه (LngI و 2) Col = 2
.Text = Trim (StrPerson . name1)
نوشتن محتوای فیلد tell از رکورد خوانده شده در خانه (LngI و 3) (Col = 1)
.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 می‌توان از دستور With استفاده کرد. شکل کلی With به صورت زیر است

```

With      نام کنترل
مقدار مشخصه 1 = مشخصه 1
      ⋮           ⋮
مقدار مشخصه n = n مشخصه n

```

End With

در رویداد Activate فرم دوم، از کنترل Msflexgrid برای نمایش اطلاعات دفترچه تلفن استفاده شده است و برای مقداردهی مشخصه‌های این کنترل از دستور With استفاده شده است. برخی از مشخصه‌های کنترل MsFlexgrid در جدول ۲-۵ آمده است.

جدول ۵-۲- مشخصه های کنترل MsFlexgrid

شرح	مشخصه
تعداد سطرها	Rows
تعداد ستون ها	Cols
پهنای جدول	Width
شماره ردیفی که مکان نما روی آن قرار دارد.	Row
شماره ستونی که مکان نما روی آن قرار دارد.	Col
محتوای سلولی از جدول که مکان نما روی آن قرار دارد.	Text
پهنای ستون شماره n (شماره ستون ها و سطرها از صفر شروع می شود)	Colwidth(n)

متد AddItem کنترل MsFlexgrid یک سطر به جدول اضافه می کند و متد Clear محتوای جدول را پاک می کند، (جدول را خالی می کند) برای مقداردهی هر سلول جدول ابتدا شماره 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 می شود.

خودآزمایی

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

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

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

Open #99 For Random Access Append Len = 140 (ب)

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

'Open a file for reading and Writing (د)

Open "c: \customer.rnd" Access Read+

۳- 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.

۴- 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 اجازه 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 شماره 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

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

۶- چه تابعی اولین شماره پرونده آزاد را برمی‌گرداند؟

۷- اگر یک پرونده ترتیبی را برای خروجی باز کنید و آن پرونده موجود باشد، چه اتفاقی

خواهد افتاد؟

۸- اگر یک پرونده ترتیبی را برای افزودن باز کنید و آن پرونده موجود باشد، چه اتفاقی

خواهد افتد؟

۹- دستور زیر چه نوع پرونده‌ای را باز می‌کند؟

Open "TestFile.dat" For Append As #1

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

۱۱- چرا برای نوشتن نوع داده کاربر در پرونده، طول رشته‌ها باید مشخص باشد؟

۱۲- با کدام دستور Visual Basic می‌توانید نوع داده کاربر را تعریف کنید؟

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

۱۴- هنجویی 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 پوشه Game در درایو C وجود دارد.)

۱۵- 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.

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



فصل

مفاهیم شیء گرای و ماژول کلاس

اهداف رفتار ی: پس از آموزش این فصل هنرجو می تواند :

- مفاهیم اصلی شیء گرای را توضیح دهد.
- Component Object Model را تعریف کند.
- تفاوت بین ماژول کلاس و کد را شرح دهد.
- یک مؤلفه COM در ویژوال بیسیک ایجاد کند.
- نمونه ای از یک کلاس را ایجاد کند.
- مشخصه ها و متدهای شیء را به کار گیرد.
- روال رویدادی برای مدیریت رویدادهای شیء بنویسد.

۱-۳- برنامه نویسی شیء گرا

برنامه نویسی شیء گرا دارای اصول و ویژگی هایی است که با پیدایش ویندوز و مفاهیمی مانند «چند وظیفه ای» که در سیستم عامل وجود دارد، مطرح شد. در این روش، برخلاف روش قبلی، به جای استفاده از تابع اصلی که وظیفه کنترل تمام برنامه را بر عهده دارد از مفاهیمی هم چون کلاس، مشخصه و شیء استفاده می شود. بنابراین برای یادگیری بهتر این روش، ابتدا باید با مفاهیم گفته شده آشنا شویم.

در محیط زندگی با مفهوم شیء آشنا هستیم. همان طور که می دانید یک شیء چیزی است مادی که دارای مشخصه‌هایی است، از جمله این که می تواند در مقابل برخی رویدادها که ممکن است برایش رخ دهد، از خود واکنش‌هایی نشان دهد. پس با این حال می توان گفت که هر شیء به همراه سه جنبه زیر شناخته می شود:

- مشخصه‌ها
- رفتار یا متد
- روابط

مشخصه‌ها، ویژگی‌هایی هستند که مشخص کننده حالت فعلی شیء است. به عنوان مثال، می توان گفت رنگ یک میز قهوه‌ای است یا قد یک شخص ۱۷۰ سانتی متر است. در این مثال، میز و انسان شیء هستند و قهوه‌ای بودن و ۱۷۰ سانتی متر بودن قد، مشخصه آن‌هاست. رفتار یک شیء، نحوه پاسخ آن شیء در مقابل رویدادهایی است که ممکن است برایش رخ دهد. به عنوان مثال، شیء میز در مقابل رویداد وارد آمدن نیروی بیش از حد، می شکند (رفتار).

تمرین: حال خودتان مثال‌هایی را برای شیء انسان و رویدادهایی که می تواند



برای وی اتفاق افتد و رفتارهای متقابل او بیاید.

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 مورد نظر هیچ رفتاری تعریف نشده باشد. روابط هر شیء نیز نشان دهنده ارتباط آن شیء با شیء‌های دیگر است. به عنوان مثال، یک شخص می تواند مالک یک شیء مانند میز باشد که در این صورت رابطه مالکیت بین شیء و شخص برقرار است.

شیء‌هایی که در دنیای واقعی وجود دارند، از انواع متفاوت هستند. حتی شیء‌های هم نوع ممکن است مشخصه‌ها، رفتارها و روابط متفاوتی داشته باشند. با توجه به این نکته، برای شیء‌ها تقسیم بندی خاصی را در نظر می گیریم و اصطلاح کلاس را تعریف می کنیم. کلاس مجموعه تمام شیء‌های هم نوع است. هر چند این شیء‌ها، مشخصه‌ها، رفتار و روابط متفاوتی داشته باشند.

به عنوان مثال، انسان یک کلاس است و هر شخص به خصوصی از این مجموعه، شیئی از کلاس مذکور محسوب می شود. با توجه به تعاریف و مفاهیمی که ذکر شد، می توان روش برنامه نویسی

شیء‌گرا را به صورت زیر بیان کرد :

هر برنامه شیء‌گرا شامل تعدادی شیء با مشخصه‌ها و متدهای متفاوت است به نحوی که روابط خاصی بین آن‌ها برقرار می‌باشد.

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


شیء‌گرایی از بهترین مفاهیم برنامه‌نویسی ساخت یافته به وجود آمده و با چندین مفهوم قوی ترکیب شده تا امکان سازماندهی برنامه‌ها به طور کارآمد را فراهم کند. به طور کلی، هنگامی که با روش شیء‌گرا برنامه می‌نویسید، مسأله را به بخش‌های تشکیل دهنده 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. همه زبان‌های برنامه‌نویسی شیء‌گرا در سه چیز مشترک هستند : کپسوله‌سازی، چندریختی و وراثت.

۱-۱-۳- کپسوله‌سازی (Encapsulation): همان طوری 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 به وسیله 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 به وجود 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.

۲-۱-۳- چندریختی (Polymorphism): 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 این که شما چگونگی عمل کردن را یاد گرفتید، می‌توانید فرمان هر نوع اتومبیلی را کنترل کنید. همین هدف می‌تواند در برنامه‌نویسی نیز اعمال شود. در برنامه‌نویسی 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 حالی که هر دو دارای این ویژگی‌ها هستند لذا کلاس مرد و کلاس زن زیر کلاس‌های کلاس انسان هستند. در مثال دیگر کلاس میوه‌ها را در نظر بگیرید که شامل همه 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.

۳-۱-۳- وراثت: وراثت، عملی است که یک شیء می‌تواند مشخصه‌های شیء دیگری را

به‌دست آورد. به‌همین دلیل، از مفهوم دسته‌بندی سلسله‌مراتبی پشتیبانی می‌کند. اگر درباره 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

و غیره) است که به طور 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 به فرد هستند. این سبب می‌شود که صفات عمومی را از پدرشان به ارث ببرند. بنابراین، مکانیزم وراثت به یک شیء امکان می‌دهد تا نمونه‌ خاصی از یک حالت عمومی تر باشد.

۲-۳- مازول‌های کلاس

برنامه‌نویسی شیء‌گرا برای مدل‌سازی مفاهیم و مسأله‌ها در دنیای واقعی به کار 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. به عنوان مثال می‌توان روی فرم، دو کادر متن و یک دکمه فرمان قرار داد که نمونه‌هایی از کلاس‌های کادر متن و دکمه فرمان هستند. نمونه‌های یک کلاس را شیء می‌گویند همان‌طور که به نمونه‌های انواع داده‌ها مثل Integer، String متغیر می‌گویند. به عبارت دیگر کلاس مجموعه 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 داده تعریف شده به وسیله کاربر (UDT) آشنا شدید. کلاس، شبیه یک نوع داده تعریف شده به وسیله 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 داده Music که به صورت زیر تعریف شده است توجه کنید.

Type Music

Composer As String

Piece As String

End type

در برنامه روالی به نام Report نوشته شده که مقادیر عناصر نوع داده Music را در کادر پیام نمایش می دهد.

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 به صورت کلاس Music تعریف کنید.

۱-۲-۳- ایجاد ماژول کلاس: در ویژوال بیسیک، کلاس ها به صورت یک ماژول پیاده سازی می شوند و تمام اجزای کلاس در داخل ماژول تعریف می شوند و ماژول کلاس به پروژه اضافه می شود.

ماژول های کلاس (پرونده های .cls) اصول و اساس برنامه نویسی شیء گرا در ویژوال بیسیک است. ماژول های کلاس را می توان با یک نقشه ساخت خانه مقایسه کرد. دقیقاً مانند یک خانه که از طریق نقشه ساخته می شود، شیء های جدید نیز از طریق ماژول های کلاس ایجاد می شوند. مشخصه ها و متدهای این شیء های جدید را می توانید تغییر دهید. یک ماژول کلاس شبیه یک ماژول کد استاندارد (پرونده .bas) است، به همین دلیل هر دو دارای عملیاتی هستند که می توانند به وسیله ماژول های دیگری در داخل برنامه کاربردی مورد استفاده قرار گیرند.

● میدان دید داده های یک ماژول استاندارد تمام برنامه است. این بدین معنی است که این داده ها تا پایان دوره حیات برنامه وجود دارند (معتبر هستند).

● برای هر شیئی که از کلاس ایجاد می‌شود، داده‌های ماژول کلاس به وجود می‌آیند. هر داده‌ی شیء فقط در دوره‌ی حیات شیء وجود دارد (با ایجاد شیء، به وجود می‌آید و با از بین رفتن شیء، از بین می‌رود).

● متغیرهایی که به صورت Public در ماژول استاندارد تعریف می‌شوند، در هر جایی از پروژه قابل رؤیت هستند. ولی متغیرهایی که به صورت public در ماژول کلاس تعریف می‌شوند فقط با مراجعه به نمونه‌ی خاصی از کلاس یا شیء قابل دسترس هستند.

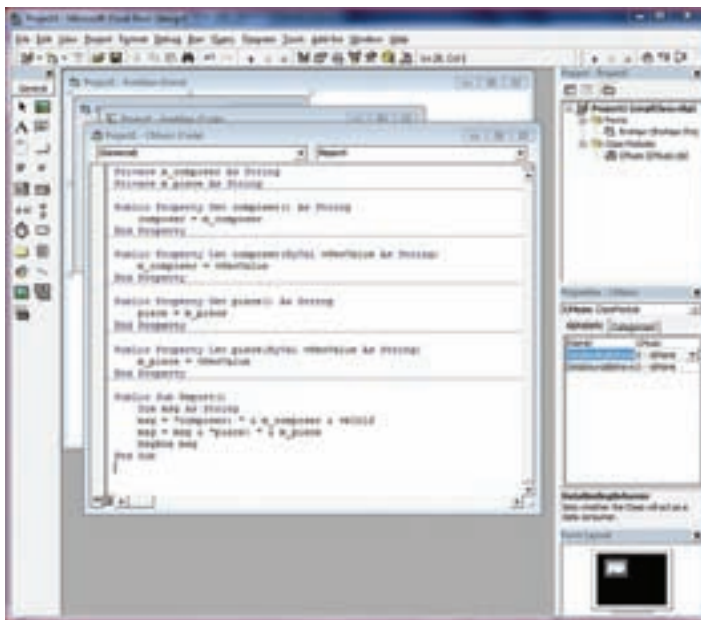


۱- پروژه جدیدی را ایجاد کرده، نام آن را SmpI Class. vbp قرار دهید. فرم پیش فرض را به FrmMain تغییر دهید.

۲- از منوی Project گزینه Add Class Module را انتخاب کنید.

۳- Class Module را انتخاب کرده، دکمه Open را کلیک کنید.

۴- در پنجره Properties مشخصه Name ماژول کلاس را CMusic قرار دهید و سپس کلاس را در فایل به نام CMusic.cls ذخیره کنید (شکل ۳-۱).



شکل ۳-۱- ایجاد مدول کلاس

۲-۲-۳- اضافه کردن مشخصه‌ها به کلاس: کلاس CMusic دارای دو مشخصه

Composer و Piece از نوع String می‌باشد. برای این مشخصه‌ها در کلاس متغیرهایی در نظر بگیرید.

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

اجزای کلاس (مشخصه‌ها و متدها) می‌توانند به صورت Public یا Private تعریف شوند متغیرهایی که به صورت Private تعریف می‌شوند فقط به وسیله متدهای کلاس قابل استفاده هستند ولی متغیرهایی که به صورت Public تعریف می‌شوند علاوه بر متدهای کلاس، در سایر زیر برنامه‌های موجود در برنامه قابل دسترسی هستند. توصیه می‌شود که متغیرهای کلاس به صورت 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 به صورت Private تعریف شده‌اند برای دسترسی به آن‌ها (خواندن و تغییر مقدار آن‌ها) روال‌های Get و Let به کلاس اضافه می‌شود.

برای اضافه کردن این روال‌ها مراحل زیر را انجام دهید.

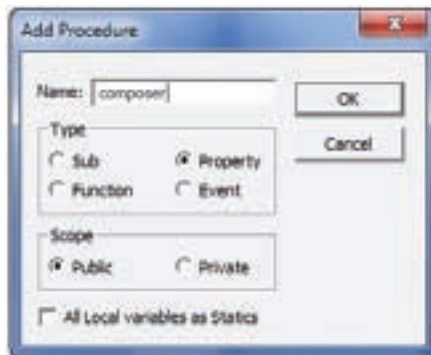
۱- پنجره Code مازول کلاس را باز کنید.

۲- از منوی Tools گزینه Add Procedure را انتخاب کنید.

۳- گزینه Property را در کادر محاوره‌ای Add Procedure انتخاب کنید و مشخصه

Composer را در کادر متن Name وارد کنید. روی Ok کلیک کنید (شکل ۲-۳)

Get Composer و Let Composer به پنجره کد اضافه می‌شود.



شکل ۲-۳- 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 بدنه روال دستور زیر را

بنویسید.

```
Composer = m_Composer
```

این دستور مقدار واقعی مشخصه Composer را که در متغیر m_Composer قرار دارد در مشخصه Composer قرار می دهد.

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

```
m_Composer = vnewValue
```

توجه کنید که ویژگی‌های بیسیک تمام مشخصه‌ها را از نوع Variant در نظر می گیرد. با توجه به این که مشخصه Composer از نوع String است کد را اصلاح کنید.
کد برای مشخصه Composer به صورت زیر تکمیل می شود:

```
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
```

۴- مشخصه 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

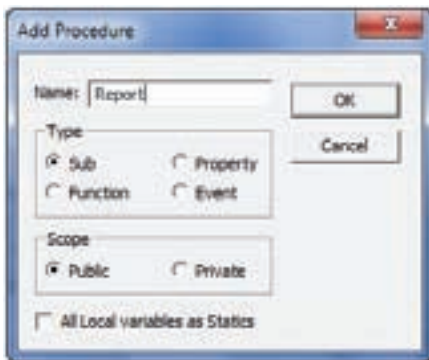
۳-۲-۳- اضافه کردن متدها به کلاس : کلاس مجموعه‌ای از مشخصه‌ها و متدها می‌باشد.

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.

۱- پنجره کد ماژول کلاس را باز کنید.

۲- از منوی Tools گزینه Add Procedure را انتخاب کنید.



۳- 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 ۳-۳)

شکل ۳-۳- اضافه کردن متد به کلاس

کد زیر به پنجره اضافه می‌شود.

Public Sub Report ()

End Sub

۴- 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

۴-۲-۳- 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:

۱- استفاده از AsNew:

```
Public object_name As New Class_name
```

● As New: کلمه 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.

۲- استفاده از دستور انتساب Set:

```
Dim object_name As Class_name
```

```
Set object_name = New Class_name
```

در خط اول همانند اعلان متغیر، شیء به نام object_name از نوع کلاس Class_name

را تعریف می‌کند.

خط دوم مانند دستور مقداردهی به متغیر می‌باشد.

● Set: کلمه کلیدی برای مقداردهی به اشیاء

● New: کلمه کلیدی برای ایجاد یک نمونه از کلاس

● object_name: نام شیء

● Class_name: نام کلاس

```
Dim MyMusic As CMusic
```

```
Set MyMusic = New CMusic
```



روی فرم دکمه 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 به گونه‌ای تغییر دهید که مشخصات قطعه موزیک را با استفاده از دو کادر متن به وسیلهٔ 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 استفاده می‌کند.

تمرین: با قرار دادن breakpoint روی روال Let در مثال فوق بررسی کنید



این روال در چه زمانی فراخوانی می‌شود؟

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

Set MyMusic = Nothing

۵-۲-۳- ایجاد رویدادها: یک شیء اعلان می‌کند که بعضی از عملیات از طریق به کارگیری

رویدادها ارائه می‌شوند. مشخصه‌ها و متدها، رابط‌های درونی هستند زیرا آن‌ها به خارج شیء ارتباطی ندارند. به طور واضح، رویدادها رابط‌های بیرونی هستند، زیرا آن‌ها درون شیء مقداردهی اولیه و خارج از شیء مدیریت می‌شوند. به عنوان مثال، هنگامی که مشخصه 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 این که مشخصه‌ای مقداردهی شود، رخ می‌دهد. هنگام نوشتن یک ماژول کلاس، رویداد Initialize را برای مقداردهی اولیه هر داده‌ای که به وسیله کلاس مورد استفاده قرار می‌گیرد، به کار گیرید. هم چنین می‌توان از این رویداد برای بارگذاری فرم‌های مورد استفاده به وسیله 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 ذخیره اطلاعات، unload کردن فرم‌ها یا اجرای وظایفی که هنگام خاتمه کلاس رخ خواهند داد، استفاده کنید.



کلاس CSpecialDate برای نمایش تاریخ تعریف شده است این کلاس دارای مشخصه Date

Format برای تعیین قالب نمایش تاریخ می‌باشد. کد ماژول کلاس به صورت زیر است.

Option Explicit

Private m_Format As String

اعلان مشخصه کلاس

متد Get برای خواندن مشخصه DateFormat

Public Property Get DateFormat () As String

DateFormat = m_Format

End Property

متد Let برای نوشتن (مقداردهی کردن) مشخصه 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 مشخصه DateFormat،

دارای متد GetSpecialDate برای نمایش تاریخ با فرمت تعیین شده در مشخصه DateFormat

است. این متد به صورت Function نوشته شده است.

روال رویداد Initialize برای این کلاس نوشته شده است و در آن مشخصه DateFormat با

مقدار اولیه «yy-mm-dd» برای فرمت نمایش تاریخ مقداردهی شده است.

توجه کنید که مقدار واقعی مشخصه DateFormat در متغیر m_Format قرار دارد. این

متغیر به صورت 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 مشخصه DateFormat استفاده می‌کنند که سبب می‌شود کلاس از متدهای Let و Get این مشخصه برای تغییر دادن و خواندن مشخصه استفاده کند.

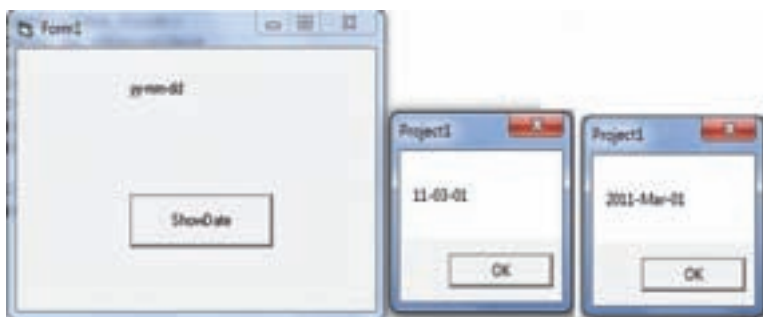
برای استفاده از کلاس CSpecialDate روی فرم برچسب Lable1 و دکمه فرمان CmdShowDate را قرار داده و کد زیر نوشته شده است؛

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 مشخصه'
MsgBox oSpecDate1.GetSpecialDate      'نمایش تاریخ در کادر پیام'
oSpecDate2.DateFormat = "yyyy-mm-dd" 'تغییر مقدار مشخصه DateFormat'
MsgBox oSpecDate2.GetSpecialDate      'نمایش تاریخ در کادر پیام'
```

End Sub

در رویداد Click دکمه 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 مشخصه DateFormat را تغییر داده سپس تاریخ را با قالب جدید نمایش می‌دهد (شکل ۳-۴).



شکل ۳-۴- فرم و کادرهای پیام مثال ۳-۳

خودآزمایی

- ۱- مفاهیم زیر را توضیح دهید :
کپسوله‌سازی - چندریختی - وراثت
- ۲- هدف از ماژول کلاس در ویژوال بیسیک، چیست؟
- ۳- اگر یک مدل شیء مبتنی بر دوچرخه ایجاد کرده باشید، چگونه آن را پیاده‌سازی خواهید کرد؟
- ۴- ماژولی برای کلاس مستطیل بنویسید که دارای متدهایی برای محاسبه محیط و مساحت مستطیل باشد؟
- ۵- 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.

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 ساده 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 برنامه کاربردی به کار 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 به کار 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.

۱-۴- 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 به داده‌ها وجود دارد که عبارتند از: Remote Data Objects (RDO)، Activex Data (ADO) 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^۱ 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 - رکورد - فیلد - کلید

تمرین: ۱ - هدف زبان SQL را بیان کنید.



۲- عبارت SELECT و اجزای آن را شرح دهید.

۴-۲- کنترل ADO Data

اگر چه می توان از ADO به طور مستقیم در برنامه های کاربردی استفاده کرد، ولی ADO Data دارای مزیت کنترل گرافیکی به همراه دکمه های پیمایش رکوردی است. همچنین این رابط ساده، امکان می دهد تا برنامه های کاربردی بانک اطلاعاتی را با حداقل کدنویسی ایجاد کنید.

کنترل ADO Data از Microsoft ADO برای ایجاد اتصال سریع بین کنترل های مقید به داده ها (data-bound) و ارایه کننده داده استفاده می کند. کنترل های Data-bound (مقید به داده ها)، کنترل هایی هستند که دارای مشخصه DataSource می باشند و شامل Image, ComboBox, CheckBox و PictureBox, Label, ListBox, و TextBox است. این کنترل ها بانکی از فیلدهای جداول بانک اطلاعاتی متصل شده و مقادیر آن فیلد را نمایش می دهند.

به علاوه ویژوال بیسیک دارای چندین کنترل Activex^۱ مقید به داده ها مثل DataGrid, DataCombo و Datalist نیز است. هنگامی که از کنترل ADO Data استفاده می کنید، می توانید هر فیلدی را به یک کنترل مقید کنید و هنگام پیمایش رکوردها، به طور خودکار محتوای فیلدها نمایش داده می شوند. این کار به طور داخلی به وسیله ویژوال بیسیک انجام می شود و نوشتن هیچ کدی نیاز نیست.

۴-۳- ایجاد فرم های مقید به بانک اطلاعاتی با Data Form Wizard

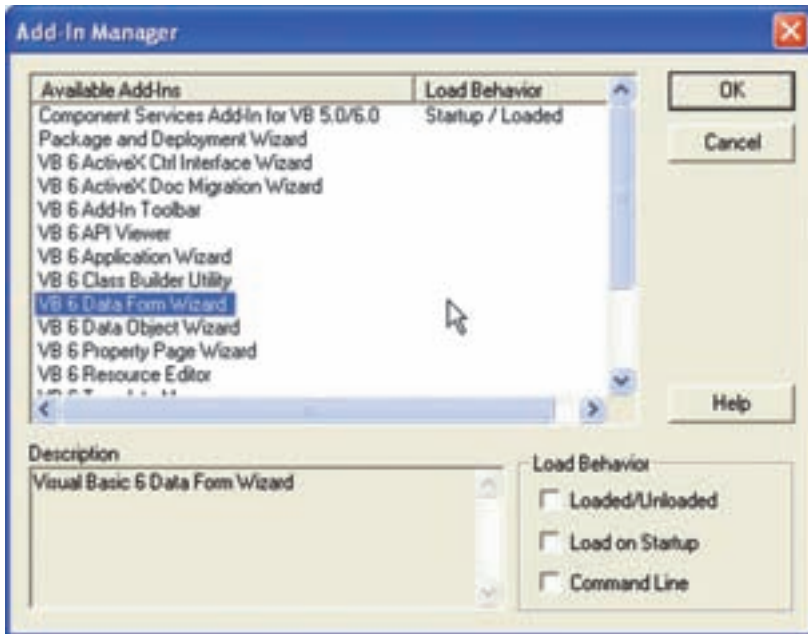
ویژوال بیسیک ابزاری ارایه می کند که امکان ایجاد فرم هایی که دارای کنترل های مقید به بانک اطلاعاتی هستند را فراهم می کند. می توان این ویزارد را از منوی Add-Ins فعال کرد. ویزارد Data Form فرمی را ایجاد می کند که امکان مرور بانک اطلاعاتی، کامل کردن کادرهای متن، برچسب ها و کنترل ADO Data را فراهم می کند.

هنگامی که ویژوال بیسیک را نصب می کنید، ویزارد Data Form و ویژوال بیسیک نصب نمی شود و باید آن را به منوی Add-Ins اضافه کنید.

۱- کنترل Activex یک مؤلفه نرم افزاری است که می تواند در صفحات وب، آفیس یا هر میزبانی که از کنترل های Activex پشتیبانی می کند، قرار داده شود و دارای متدهایی است که می تواند از طریق سایر برنامه های کاربردی، کتابخانه های پیوند پویا (ppl)، صفحات وب یا کنترل های دیگر فراخوانی شود.

۱-۳-۴- نصب Data Form Wizard :

- ۱- از منوی Add-Ins گزینه Add-In Manager را انتخاب کنید.
- ۲- در کادر محاوره‌ای Add-In Manager، گزینه Data Form Wizard، VB6 را از لیست انتخاب کرده و کادر علامت Loaded/Unloaded را فعال کنید (شکل ۴-۱) روی OK کلیک کنید.



شکل ۴-۱- اگر گزینه 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 ویزوال بیسیک اضافه کردید، می‌توانید آن را در تمام پروژه‌ها به کار

برید.

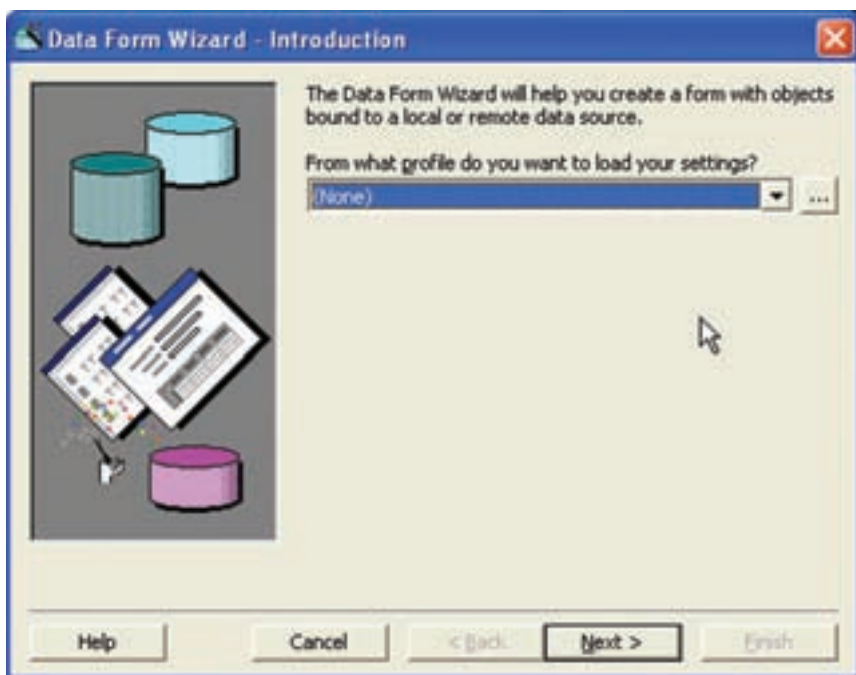


ایجاد فرم مقید به جدول بانک اطلاعاتی: از ویزارد VB Data Form به منظور ایجاد فرم

برای جدول tblWorks از بانک اطلاعاتی composer.mdb استفاده می‌کنیم.

۱- از منوی Add-Ins گزینه DataForm Wizard را انتخاب کنید.

۲- در کادر محاوره‌ای Introduction روی Next کلیک کنید (شکل ۴-۲).



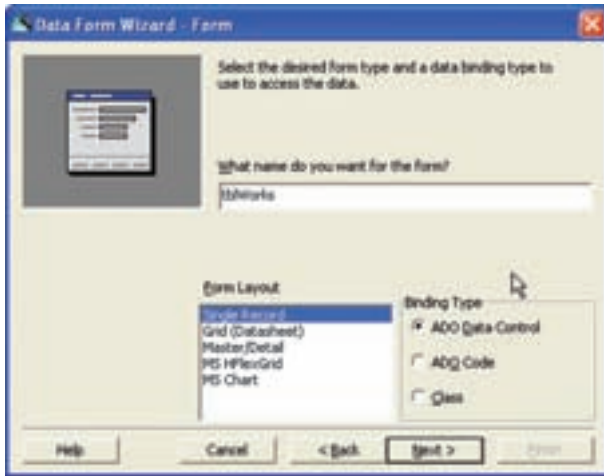
شکل ۲-۴- کادر محاوره‌ای Introduction امکان بارگذاری پروفایل تنظیمات Data Form Wizard را فراهم می‌کند.

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

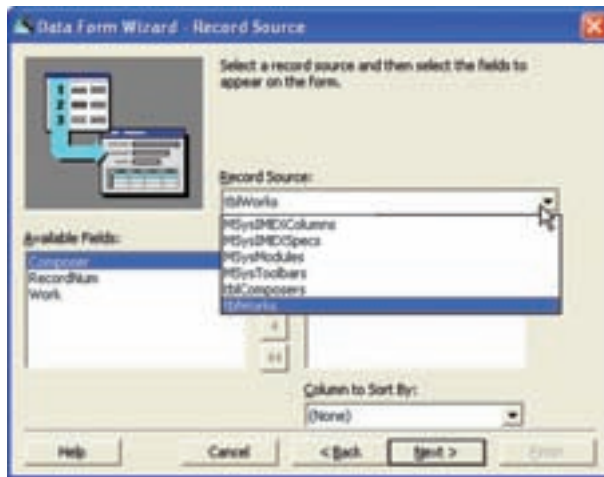
۴- در کادر محاوره‌ای Database روی دکمه Browse کلیک کنید تا کادر محاوره‌ای Access Database را مشاهده کنید. اکنون می‌توانید بانک اطلاعاتی مورد نظر برای ایجاد فرم را انتخاب کنید. Composer.mdb را پیدا کرده و روی Next کلیک کنید.

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

۵- نام فرم را frmWorks قرار داده و از لیست Form Layout گزینه Single Record را انتخاب کنید. از گزینه‌های Binding Type گزینه ADO Data Control را انتخاب کرده و روی Next کلیک کنید (شکل ۳-۴).



شکل ۳-۴ طرح کلی فرم را با انتخاب گزینه‌ای از لیست Form Layout انتخاب کنید.

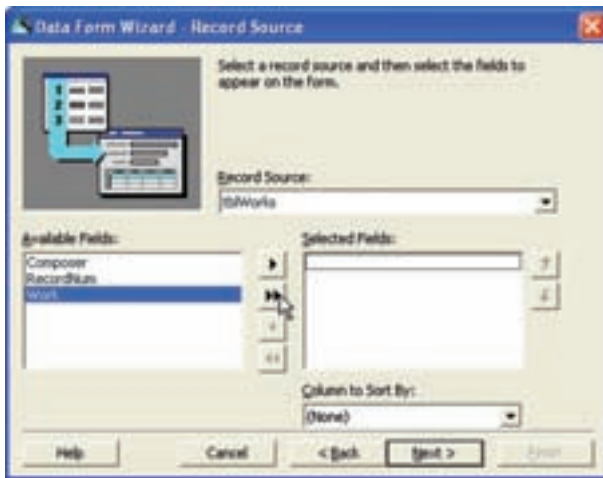


شکل ۴-۴ تمام جدول‌ها و پرس و جوهای بانک اطلاعاتی انتخاب شده در لیست بازشوی Record Source فهرست می‌شوند. بعد از انتخاب Record Source، فیلدهای آن در لیست Available fields ظاهر خواهند شد.

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

۷- 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 دکمه «» و سپس Next کلیک کنید تا تمام فیلدها انتخاب شود. (شکل ۵-۴) در صورتی که

می خواهید رکوردها براساس فیلد خاصی مرتب شوند، از کادر Column to sort by نام فیلد مورد نظر را انتخاب کنید. روی Next کلیک کنید.



شکل ۴-۵ 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 دکمه ▶ کلیک کنید. برای حذف فیلدها از فرم داده‌ها، روی دکمه‌های ◀ و ◀◀ کلیک کنید.

۸- 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 ۴-۶) روی Next کلیک کنید.



شکل ۴-۶ می‌توان تعداد محدودی از دکمه‌ها را با انتخاب یا پاک کردن کادر علامت مربوطه روی فرم قرار داد.

۹- در کادر محاوره‌ای پایانی تنظیمات ایجاد شده را با کلیک کردن روی دکمه سه نقطه، ذخیره کنید (شکل ۷-۴). این تنظیمات در فایل‌ی با پسوند .twp ذخیره می‌شوند.

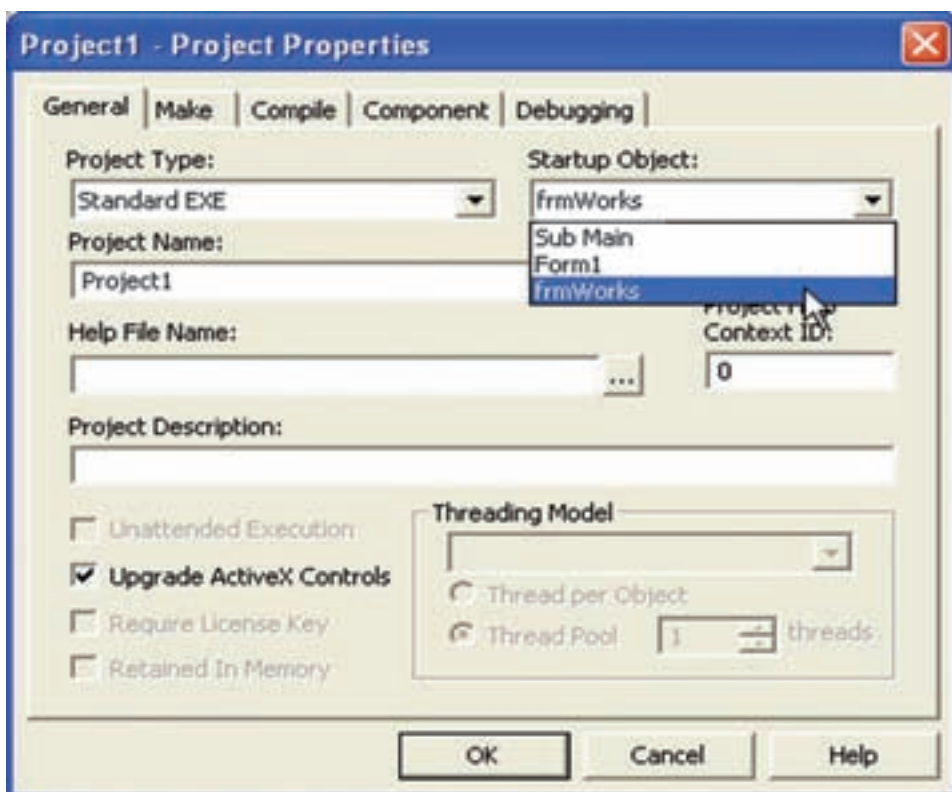


شکل ۷-۴- ذخیره تنظیمات در یک پروفایل می‌تواند سبب صرفه‌جویی در وقت برنامه بانک اطلاعاتی شود.

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

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

۱۲- فرم frmWorks را از لیست بازسوی Startup Object انتخاب کنید (شکل ۸-۴).



شکل ۸-۴ در صورتی که می‌خواهید فرم جدید را به صورت فرم آغازین در نظر بگیرید، باید شیء Startup را دوباره مقاردهی کنید.

۱۳- پروژه را ذخیره کرده و اجرا کنید (شکل ۹-۴).



شکل ۹-۴ فرم داده‌های ایجاد شده با ویزارد، امکان اضافه، به‌هنگام، حذف و نو کردن (refresh) 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 مشخصه Data BaseName آن را مساوی بانک اطلاعاتی موردنظر قرار دارد و در مشخصه 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.

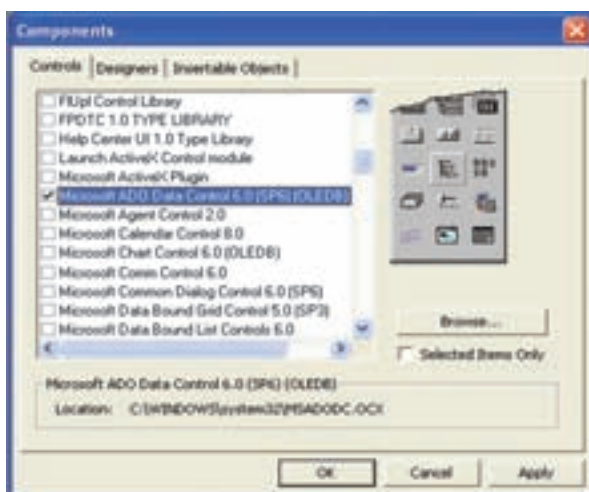
۴-۴- اضافه کردن کنترل ADO Data به جعبه ابزار

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

۱- از منوی Project گزینه Components را انتخاب کنید.

۲- در کادر محاوره‌ای Components، روی Microsoft ADO Data Control 6.0 (OLE DB) کلیک کنید (شکل ۱-۴).

۳- روی OK کلیک کنید.



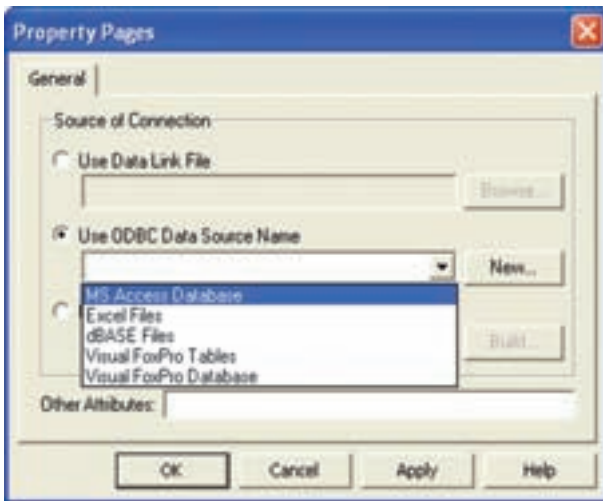
شکل ۱-۴- کادر محاوره‌ای Components

۴-۵- اتصال به منبع داده‌ها

در زمان طراحی، می‌توان با مقدار دهی مشخصه `ConnectionString` کنترل `ADO Data` با یک رشته اتصال معتبر، اتصال به منبع داده‌ها را برقرار کرد. برای مقداردهی این مشخصه روی شیء راست کلیک کرده و گزینه `ADODC Propertic` را انتخاب کنید (یا در پنجره `Properties` روی علامت سه نقطه روی مشخصه `ConnectionString` کلیک کنید) کادر محاوره‌ای `property Page` باز می‌شود (شکل ۴-۱۱). هنگامی که مشخصه `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` 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 پرونده `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 رشته اتصال استفاده کنید. می‌توان به فهرستی از `DSN`های تعریف شده سیستم از طریق کادر ترکیبی دسترسی پیدا کرد و `DSN` موردنظر را انتخاب نمود و یا (شکل ۴-۱۱). روی `New` کلیک کرد و از طریق کادر محاوره‌ای `Create New Data Source Wizard` برای ایجاد `DSN` جدید و یا اصلاح `DSN`های موجود، استفاده کرد.



شکل ۴-۱۱- به‌کارگیری
گزینه نام منبع داده‌های ODBC

● Use Connection String: این گزینه تعیین می‌کند که شما می‌خواهید از یک رشته

اتصال برای دسترسی به داده‌ها استفاده کنید. روی Build کلیک کنید تا به کادر 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 آرایه‌کننده OLE DB تعیین کرد.

۴-۶- تعیین رشته اتصال

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

مقداردهی مشخصه Connection String را به صورت زیر انجام دهید :

۱- یک کنترل ADO Data روی فرم قرار دهید.

۲- در پنجره Properties مربوط به این کنترل، مقابل مشخصه Connection String

روی ... کلیک کنید.

۳- گزینه Use Connection String را انتخاب و روی Build کلیک کنید.

۴- در کادر محاوره‌ای Data Link Properties و زبانه Provider به دلیل این که بانک

اطلاعاتی که ایجاد کرده‌ایم، در Access بوده است، گزینه Microsoft jet 4.0 OLE DB Provider را انتخاب می‌کنیم.

۵- در زبانه Connection با کلیک روی ... شماره ۱ نام بانک اطلاعاتی مورد نظر (در این

مثال، Nwind.mdb) را انتخاب کنید. در شماره ۲ این کادر محاوره‌ای می‌توانید نام کاربری و گذر واژه خاصی را برای دسترسی به بانک اطلاعاتی تعیین کنید.

۶- 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؟

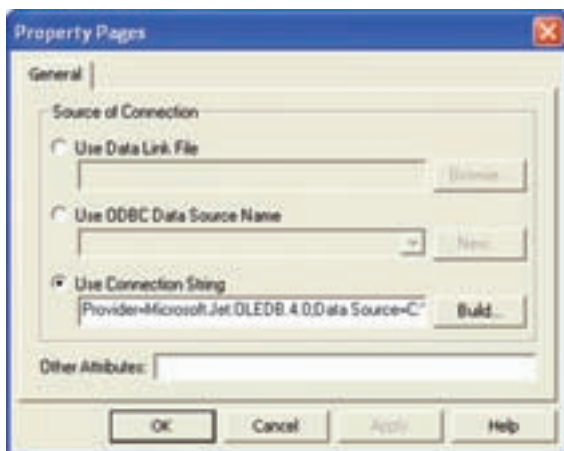
۷- در صورتی که پیام موفقیت‌آمیز بودن ایجاد اتصال را دریافت کردید، روی OK کلیک کنید

و در کادر محاوره‌ای Data Link Properties نیز روی OK کلیک کنید.

یک مقدار رشته‌ای به‌طور خودکار برای Connection String تولید خواهد شد

(شکل ۱۲-۴).

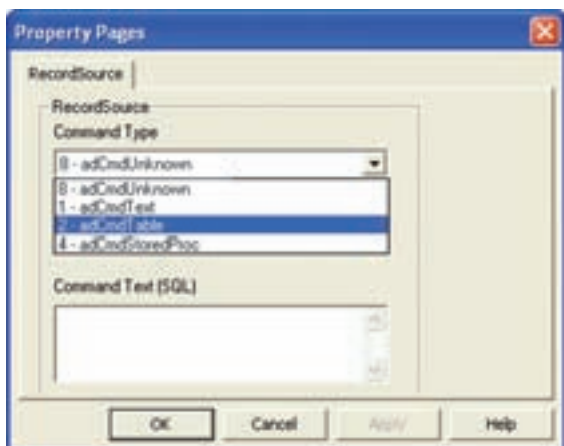
۸- روی OK کلیک کنید تا کادر محاوره‌ای Property Pages بسته شود.



شکل ۱۲-۴- یک مقدار رشته‌ای اتصال

۱-۶-۴- تعیین مشخصه RecordSource: بعد از مقداردهی مشخصه ConnectionString

برای اتصال به بانک اطلاعاتی، می‌توانید مشخصه RecordSource را برای به دست آوردن رکوردها، مقداردهی کنید. مشخصه 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 رشته 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. مشخصه Record Source می‌تواند در زمان طراحی و با استفاده از Property Pages مقداردهی شود (شکل ۱۳-۴).



شکل ۱۳-۴- صفحه مشخصه

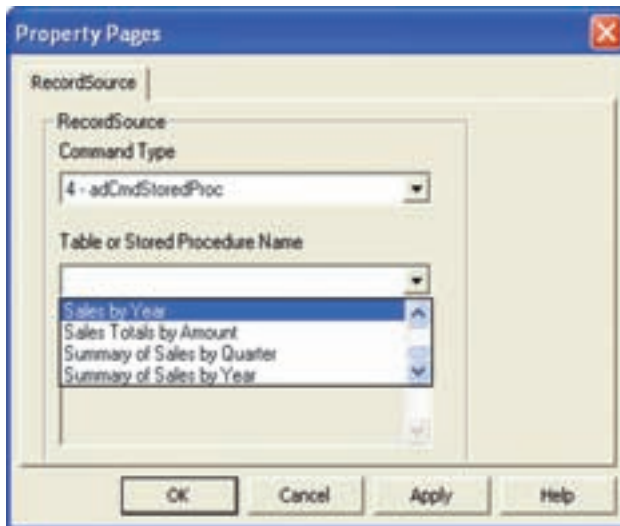
Record Source

در کادر محاوره‌ای فوق، پارامتر نوع فرمان را تعیین کنید که به ADO بیان می‌کند که نوع شیء فرمان مورد استفاده کدام است. جدول ۴-۱، گزینه‌های نوع فرمان مختلف را شرح می‌دهد.

جدول ۴-۱- انواع فرمان ADO

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

اگر adCmd Table یا adCmd StoredProc را انتخاب کنید، از کادر لیست زیرین، می‌توانید نام جدول یا روال ذخیره شده را انتخاب کنید (شکل ۴-۱۴).



شکل ۴-۱۴- انتخاب یک روال ذخیره شده به عنوان RecordSource

اتصال کنترل ADO Data به منبع داده ها: در این تمرین، کنترل ADO Data را به فرم اضافه خواهید کرد و سپس آن را به منبع داده ها متصل می کنید. منبع داده ها، بانک اطلاعاتی نمونه Northwind (Nwind.mdb) است که درون ویژوال بیسیک قرار دارد. با چگونگی اضافه کردن کنترل ADO Data به جعبه ابزار قبلاً آشنا شدید.

۱- کنترل ADO Data را به Form1 اضافه کنید.

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

کادر محاوره ای Property Pages ظاهر می شود. در زبانه General، مطمئن شوید که گزینه Connection String انتخاب شده باشد.

۳- برای تعیین رشته اتصال، روی Build کلیک کنید.

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

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

۵- از زبانه Connection برای انتخاب یا تایپ نام بانک اطلاعاتی استفاده کنید. روی ... کلیک کنید.

۶- در کادر محاوره ای SelectAccessDatabase، روی Nwind.mdb و سپس open کلیک کنید.

۷- در کادر محاوره ای Data Link Properties روی Test Connection کلیک کنید.

یک کادر پیام ظاهر شده و بیان می کند که اتصال موفقیت آمیز بوده است یا نه؟

۸- روی OK کلیک کنید.

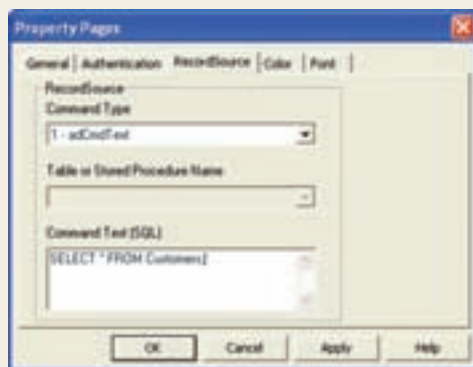
۹- تا این جا، مقدار رشته اتصال ایجاد شده است، روی زبانه RecordSource از کادر محاوره ای Property Page کلیک کنید.

۱۰- برای Command Type گزینه adCmd Text را انتخاب کنید.

۱۱- برای مقدار Command Text (SQL)، عبارت SQL زیر را تایپ کنید (شکل ۱۵-۴).

SELECT * FROM Customers

- ۱۲- 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.
- ۱۳- از منوی File گزینه Save Project را انتخاب کنید و پروژه را با نام PrjADO.vbp ذخیره کنید تا در تمرین بعدی نیز از آن استفاده کنید.



شکل ۱۵-۴- کادر محاوره‌ای
 Property Pages

۴-۷- مقیدسازی کنترل‌ها

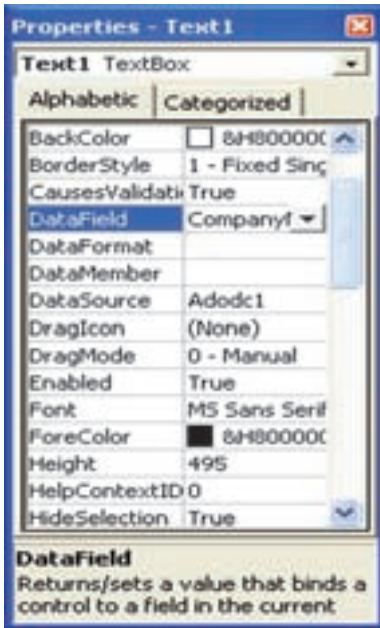
بعد از تعیین مشخصه‌های ConnectionString و RecordSource برای کنترل ADO Data، که به ترتیب بانک اطلاعاتی و یکی از جداول یا Queryهای بانک اطلاعاتی را مشخص می‌کند، می‌توانید یک کنترل مقید برای نمایش داده‌ها روی فرم اضافه کنید.

هنگامی که 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 این که مقدار کنترل مقید به طور خودکار از بانک اطلاعاتی بازیابی و در آن نوشته می‌شود، به برنامه‌نویسی کمی نیاز دارد.

۴-۷-۱- مقداردهی مشخصه‌های DataSource و DataField: برای این که کنترلی را به

کنترل ADO Data مقید کنید، باید این دو مشخصه را تنظیم نمایید. مشخصه 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 به وسیله مشخصه

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



مشخصه DataField نام فیلد معتبری که در شیء Recordset به وسیله 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 پنجره Properties مقاداردهی کرد (شکل ۱۶-۴).

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

شکل ۱۶-۴ - تنظیم مشخصه‌های DataSource و Datafield

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

تمرین:

مقید سازی کنترل‌ها

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

۱- پروژه PrjADO.vbp را باز کنید.

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

۳- مشخصه DataSource هر دو کادر متن را Adodc1 قرار دهید.

۴- مشخصه DataField کادر متن اول (Text1) را برابر با CompanyName

قرار دهید.

۵- مشخصه DataField کادر متن دوم (Text2) را برابر با Phone قرار دهید.

۶- 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 ۱۷-۴ خواهد بود.

۷- روی End کلیک کنید.

۸- پروژه را ذخیره کنید.



شکل ۱۷-۴- نام شرکت و شماره تلفن در کنترل‌های مقید نمایش داده می‌شوند.

۸-۴- 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.

۱-۸-۴- کدنویسی کنترل ADO Data: بعد از این که چگونگی تنظیم مشخصه‌های کنترل

ADO Data را برای اتصال به بانک اطلاعاتی و مقیدسازی کنترل‌ها را آموختید. در این قسمت، درباره کار کردن 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 پیش‌تری روی چگونگی تعامل با داده‌های خارجی اعمال کرد.

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

مشخصه 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 ارائه 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، مشخصه RecordSource کنترل ADO Data را مقدار دهی کنید. مشخصه RecordSource یک مقدار رشته‌ای است که می‌تواند نام جدول یا پرس‌وجو در یک بانک اطلاعاتی باشد.

همچنین می‌توان مشخصه Filter از شیء Recordset را برای انتخاب رکوردهای خاصی به کار 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 این که مشخصه RecordSource مقداردهی شد، می‌توانید مشخصه‌های EOF و BOF شیء Recordset را بررسی کنید. این مشخصه‌ها تعیین می‌کنند که در ابتدا یا انتهای Recordset هستید. اگر رکورد جاری اولین رکورد باشد، مقدار BOF برابر با True خواهد بود و اگر اشاره‌گر در انتهای رکوردها باشد، EOF برابر True است. در صورتی که هیچ رکوردی در Recordset نباشد، مقدار هر دو مشخصه 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 به‌هنگام کند، ولی می‌توان متد Update را برای ویرایش داده‌ها به جای استفاده از فلش‌های کنترل ADO، به کار گرفت.

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

- ۱- جابه‌جایی به رکوردی که می‌خواهید ویرایش کنید.
- ۲- تغییر هر نوع اطلاعاتی که در کنترل‌های مقید نمایش داده می‌شوند.
- ۳- کلیک کردن روی هر کدام از فلش‌های کنترل ADO Data برای جابه‌جایی به رکورد دیگر. با این وجود، به‌کارگیری دکمه فرمان امکان اضافه کردن کد به رویداد Click برای انجام کارهایی مثل بررسی صحت فیلدها روی فرم، قبل از به‌هنگام سازی را فراهم می‌کند. برای انجام عمل به‌هنگام سازی 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

۳-۸-۴- 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 ذخیره تغییرات در رکورد جدید، می‌توان متد Update شیء Recordset را فراخوانی کرد یا می‌توان روی یکی از دکمه‌های پیمایش کنترل ADO Data کلیک کرد.

۴-۸-۴- 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 ۲-۴- مقادیر پارامتر Affect Records

مقدار AffectRecords	شرح
adAffectCurrent	این گزینه فقط رکورد جاری را حذف خواهد کرد و گزینه پیش‌فرض است.
adAffectGroup	این گزینه می‌تواند حذف تمام رکوردهایی که از مشخصه Filter عبور کرده‌اند را دربرگیرد. برای استفاده از این گزینه باید ابتدا مشخصه 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 این که از رکورد حذف شده جابه‌جا شدید، در دسترس نخواهد بود. برای به‌کارگیری متد Delete از شکل کلی زیر استفاده کنید :


Recordset.Delete AffectRecords

برای مشاهده این که آخرین رکورد حذف شده است یا نه، مشخصه 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 برنامه 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 این صورت 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 : SearchDirection, SkipRows 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 می‌کند، 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 (به عنوان مثال، "state='WA'"). و مقادیر تاریخ با علامت # مشخص می‌شوند (به عنوان مثال، #7/2/93#)

برای جستجو براساس بخشی از یک رشته، می‌توان از کلید واژه Like موجود در SQL استفاده کرد. به خاطر 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 به کارگیری این عملگر را نشان می‌دهد.

```
adostates.Recordset.Find "state LIKE 'M%'"
```

 **نکته:** متد Find می‌تواند کند باشد، مگر این که تعداد کل رکوردها در مجموعه رکورد را محدود کنید. روش دیگر برای پیدا کردن رکوردها، استفاده از عبارت SQL هنگام مقداردهی مشخصه RecordSource است. همچنین می‌توانید از مشخصه 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 داده Long است که مقدار پیش فرض آن صفر است. از این پارامتر برای تعیین فاصله از سطر جاری یا نشانه آغازین برای شروع جستجو استفاده می‌شود.

• **SearchDirection:** این پارامتر تعیین می‌کند 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:

– adsearch Forward(0) – جستجو به سمت جلو از رکورد جاری در صورتی که جستجو

با موفقیت انجام نشود، EOF برمی‌گرداند.

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

• **Start:** این پارامتر، یک نشانه Variant است که به عنوان محل شروع جستجو مورد استفاده قرار می گیرد. مقدار این پارامتر می تواند یکی از مقادیر عددی زیر باشد :

– adBookmarkCurrent (0) – رکورد جاری

– adBookmark First (1) – اولین رکورد

– adBook mark Last (2) – آخرین رکورد

هنگامی که رکوردها را با استفاده از متد Find جستجو می کنید، می توانید مشخصه های EOF و BOF از شیء Recordset را به کار 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 مشخصه ها را نشان می دهد :

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

End If

مشخصه های BOF، EOF و Bookmark از مشخصه های Recordset می باشند.
لذا این قطعه کد باید داخل بلوک

```
With ADO Data Recordset
```

```
·
·
·
```

```
End With
```

قرار گیرد.

مشخصه Bookmark برای علامت گذاری رکورد معینی در شیء Recordset استفاده می شود.
برای این منظور با استفاده از متد Move روی رکورد رفته از دستور زیر استفاده می کنیم؛

```
Dim VarBookmark As Variant
```

VarBookmark = ADodc . Recordset . Bookmark

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

ADodc . Recordset . Bookmark = VarBook Mark

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

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، پروژه ای 

که در دو تمرین قبلی این فصل ایجاد کردید را ادامه خواهیم داد. عملیاتی را به برنامه کاربردی اضافه خواهیم کرد تا اسامی شرکت ها را جستجو کند.

۱- پروژه prjADO.vbp که در تمرین قبل ذخیره کردید را باز کنید.

۲- بین دو کنترل Text2 و Adodc1 یک دکمه فرمان اضافه کنید.

۳- در پنجره Properties، مشخصه Name کنترل Command1 را به

CmdSearch تغییر دهید.

۴- مشخصه Caption کنترل CmdSearch را به search & تغییر دهید.

۵- در رویداد Click دکمه فرمان Cmdsearch، کد زیر را تایپ کنید.

```
Adodc1.Recordset.MoveFirst
```

```
Adodc1.Recordset.Find "CompanyName='FrankenVersand'"
```

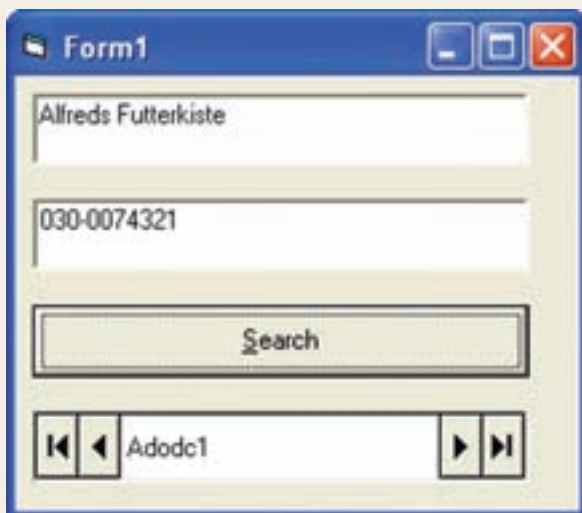
```
If Adodc1.Recordset.EOF Then MsgBox "Record not found."
```

۶- از منوی Run گزینه Start را انتخاب کنید.

۷- روی دکمه Search کلیک کنید.

توجه کنید که رکورد حاوی نام شرکت Frankenversand رکورد جاری خواهد


شد. برنامه کاربردی شبیه شکل ۱۸-۴ خواهد بود.



شکل ۱۸-۴- استفاده از متد Find برای جستجو

۸- روی End کلیک کنید.

۹- 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 شده 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 ۲-۴ - 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	جابجایی در Recordset
MoveFirst	جابجایی به اولین رکورد Recordset
MoveLast	جابجایی به آخرین رکورد Recordset
MoveNext	جابجایی به رکورد بعدی در Recordset
MovePrevious	جابجایی به رکورد قبلی در Recordset
Update	به روز کردن تغییرات انجام شده در Recordset
find	به منظور پیدا کردن رکوردهایی با ویژگی‌های معین در Recordset (جستجو در مجموعه رکوردها)

جدول ۳-۴ - 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 SQL یا 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

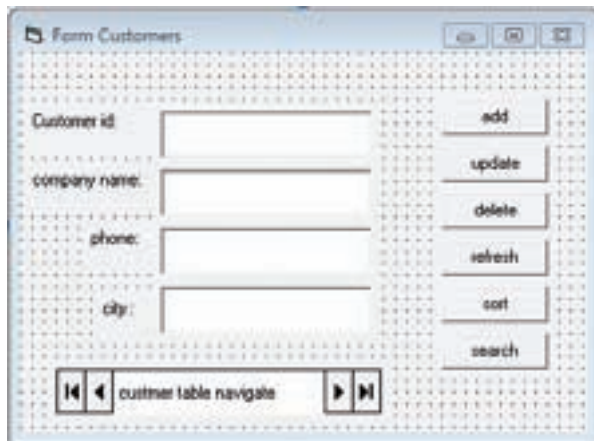
varBookmark =adodc.Recordset.bookmark

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

adodc.Recordset.bookmark =varBookmark



در این مثال فرمی مانند شکل ۴-۱۹ ایجاد کنید .



شکل ۴-۱۹- فرم نمایش رکوردهای جدول Customers

ابتدا نام کنترل adoCustomers را 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 می شود که تمام فیلدهای جدول customers از طریق کنترل adoCustomers قابل دسترسی باشند.

نام کنترل های TextBox را با توجه به فیلدی که نمایش می دهند و نام دکمه های فرمان را با توجه به عملیاتی که انجام می دهند، 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()  
    AdoCustomers.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 دکمه 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 دکمه 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 دکمه 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.

۴-۹- کاربرد کنترل DataGrid

اولین کنترلی که به کار خواهیم برد، کنترل DataGrid است که امکان مشاهده و ویرایش چندین سطر از داده‌ها را به طور 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:

۱- از منوی Project گزینه Components را انتخاب کنید.

۲- 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 آیکن 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 ۳-۴

پیوند Data Grid به کنترل ADO Data :

۱- روی کنترل DataGrid در روی فرم کلیک کنید. مشخصه Name را با dgrdCustomers

مقداردهی کنید، زیرا اطلاعات مشتری را نمایش خواهد داد.

۲- adoCustomers را از لیست بازشوی مشخصه DataSource انتخاب کنید (این همان

کنترل ADO Data است که قبلاً ایجاد شده است).

بعد از تعیین DataSource برای DataGrid، کنترل DataGrid به طور خودکار

تعداد سطرها و ستون‌ها را پیکربندی می‌کند. برای مشاهده فرم در عمل، از منوی Run

گزینه Start را انتخاب کنید. پنجره، داده‌های جدول Customers را نشان خواهد داد

(شکل ۴-۲۰).

به دلیل این که تنظیمات پیش فرض همیشه بهترین نیستند، DataGrid به طور 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 هر کدام به طور مستقل از دیگری قابل مشاهده و مرور

باشند. انجام این کار در Excel و سایر صفحات گسترده متداول است.

CustomerID	CompanyName	ContactName
ALFKJ	Alfreds Futterkiste	Maria Anders
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
ANTON	Antonio Moreno Taquería	Antonio Moreno
AROUT	Around the Horn	Thomas Hardy
BERGS	Berglunds snabbköp	Christina Berglund
BLAUS	Blauer See Delikatessen	Hanna Moos
BLONP	Blondel père et fils	Frédérique Citeaux
BOLID	Bólido Comidas preparadas	Martín Sommer
BONAP	Bon app'	Laurence Leblond
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln
BSBEV	B's Beverages	Victoria Ashworth

شکل ۲۰-۴- DataGrid می تواند به طور خودکار خودش را بیکر بندی کند تا داده های منبع تعیین شده را نمایش دهد.

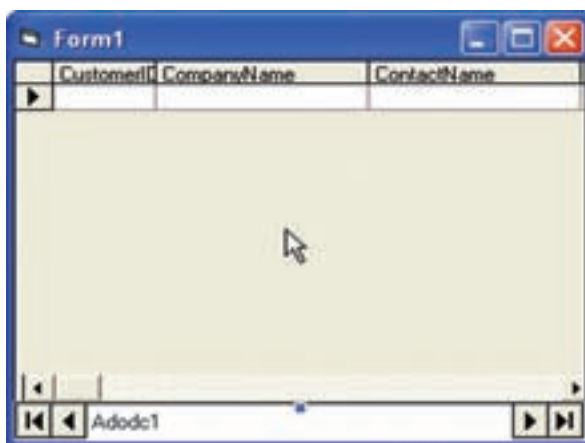
۱-۹-۴- انتخاب ستون ها :

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

مثال ۴-۴

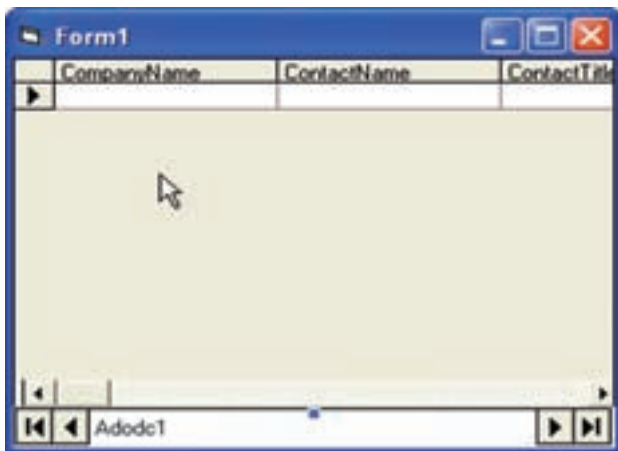
تعیین ستون ها در DataGrid:

۱- روی DataGrid کلیک راست کرده و از منوی میان بر، Retrieve Fields را انتخاب کنید.



هنگامی که ویزوال بیسیک درباره جایگزینی طرح کلی سؤال می کند، Yes را انتخاب کنید. تمام فیلدها از Recordsource انتخاب شده (در این حالت، جدول Customers) در کنترل DataGrid بارگذاری خواهند شد (شکل ۲۱-۴).

شکل ۲۱-۴- کنترل DataGrid تمام فیلدها را از جدول بارگذاری می کند بنابراین می توان یکی از آن ها را انتخاب کرده و مشاهده کرد.



۲- روی DataGrid کلیک

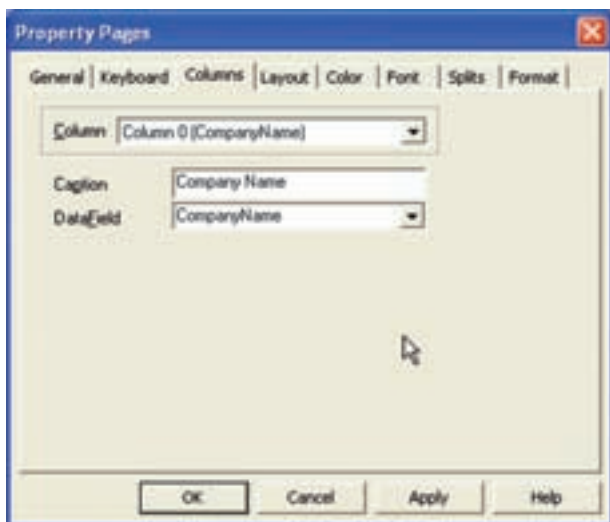
راست کرده و گزینه Edit را از منوی میان بر انتخاب کنید. اکنون می توانید روی ستون های خاصی کلیک کرده و آن ها را ویرایش کنید.

• برای حذف ستون ها از لیست، روی فیلد CustomerID کلیک راست کرده و Delete را انتخاب کنید (شکل ۲۲-۴).

شکل ۲۲-۴- هنگامی که DataGrid در حالت Edit است، می توان ستون ها را اضافه و حذف کرد.

۳- ستون های Contact Title، Address، City، Region، PostalCode و Fax را حذف کنید.

• در این کنترل نمی توان ستون ها را با استفاده از کشیدن و رها کردن دوباره مرتب کرد. به عنوان مثال، اگر می خواهید ستون های ContactName و CompanyName را جابه جا 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.

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

شکل ۲۳-۴- مشخصه Caption صفحه Columns کنترل می کند که چه چیزی در عنوان هر ستون نمایش داده خواهد شد.

۵- مرحله ۶ را برای تغییر عنوان هر ستون مورد نظر، تکرار کنید.

۲-۹-۴- پر کردن فرم: می‌توان مشخصه Visible کنترل ADO Data را با False

مقدار دهی کرد تا کنترل ADOData برای کاربران قابل مشاهده نباشد. هم چنین می‌توان DataGrid را به اندازه‌ای بزرگ کرد که فرم را پر کند. ابتدا مشخصه Align را با vbAlignTop مقداردهی کنید تا DataGrid در بالای فرم قرار گرفته 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 Form_Resize()
```

```
    dgrdCustomers.Height=Me.scaleHeight
```

```
End sub
```



Company Name	ContactName	ContactTitle
Alfreds Futterkiste	Maria Anders	Sales Representative
Ana Trujillo Emparedados y heladerias	Ana Trujillo	Owner
Antonio Moreno Taqueria	Antonio Moreno	Owner
Around the Horn	Thomas Hardy	Sales Representative
Berglunds snabbkop	Christina Berglund	Order Administrator
Blauer See Delikatessen	Hanna Moos	Sales Representative
Blondel père et fils	Frédérique Citeaux	Marketing Assistant
Bólido Comidas preparadas	Martín Sommer	Owner
Bon app'	Laurence Leblond	Owner
Bottom-Dollar Markets	Elizabeth Lincoln	Accountant
B's Beverages	Victoria Ashworth	Sales Representative
Cactus Comidas para llevar	Patricio Simpson	Sales Agent
Centro comercial Moctezuma	Francisco Chang	Marketing Assistant
Chop-suey Chinese	Yang Wang	Owner
Comércio Mineiro	Pedro Afonso	Sales Assistant
Consolidated Holdings	Elizabeth Brown	Sales Representative
Drachenblut Delikatessen	Sven Ottlieb	Order Administrator
Du monde entier	Janine Labrousse	Owner
Eastern Connection	Ann Devon	Sales Agent
Ernst Handel	Roland Mendel	Sales Manager
Familia Arquibaldo	Aria Cruz	Marketing Assistant
FISSA Fabrica Inter. de Salami	Diego Roel	Accountant
Foiles gourmandes	Martine Rancé	Assistant
Folk och färgvarer AB	Maria Larsson	Owner


شکل ۲۴-۴ DataGrid کامل شده است.


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

۳-۹-۴- پیکربندی سایر مشخصه‌های DataGrid: می‌توان چندین مشخصه دیگر را روی DataGrid پیکربندی کرد که آن‌ها نیز به سادگی می‌توانند مورد استفاده قرار گیرند. جدول ۴-۵ هدف هر مشخصه را شرح می‌دهد.

جدول ۴-۵. 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 از کلیدهای 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 ارائه 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 جدول های دیگر مرتبط است، حذف رکورد ممکن نخواهد بود و سبب بروز پیام خطا می شود. حذف کردن فقط هنگامی ممکن است که تمام رکوردهای مرتبط حذف شده باشند. (Refrential Integrity)

 **نکته:** هنگامی که یک برنامه کاربردی را طراحی و درباره نمایش داده های بانک اطلاعاتی آن فکر می کنید، باید بدانید که اگر داده ها را در یک جدول ذخیره کنید می توانید آن ها را در DataGrid ویرایش کنید. در صورتی که نیاز به قراردادن داده ها در بیش از یک جدول دارید، باید نوع دیگری از فرم را به کار ببرید. به عنوان مثال، یک سفارش هم اطلاعاتی درباره سفارش و هم کالاهای سفارش شده خواهد داشت. سفارش را نمی توانید در یک grid ویرایش کنید. زیرا داده ها حداقل در دو جدول متفاوت ذخیره شده اند. با این وجود، DataGrid می تواند لیستی از کالاها را نشان دهد. در صورتی که فقط می خواهید داده های سایر جدول ها را مشاهده کنید، DataGrid هنوز هم گزینه مناسبی است.

۴-۱۰- کاربرد کنترل DataList

کنترل DataList خیلی ساده تر از کنترل DataGrid است. از این کنترل برای نمایش لیست یک ستونی از داده های جدول استفاده می شود. به دلیل این که این کنترل مقید به داده هاست، 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.

 مثال ۴-۵

ایجاد و استفاده از کنترل DataList:

- ۱- روی فرم جدید، یک کنترل ADO Data اضافه کنید که به جدول Customers از بانک اطلاعاتی Nwind.mdb دسترسی داشته باشد.
- ۲- کنترل DataList را از طریق کادر محاوره ای Components به جعبه ابزار اضافه کنید (در

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

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

۴- نام کنترل DataList را dlstCustomers قرار دهید.

۵- مشخصهٔ RowSource را با adoCustomers مقداردهی کنید. این مشخصه منبع داده‌هایی

که لیستی از داده‌ها را برای نمایش ارائه خواهد کرد، مشخص می‌کند. برای تعیین فیلد موردنظر،

مشخصهٔ ListField را با CompanyName مقداردهی کنید.

۶- فرم را ذخیره کرده و پروژه را اجرا کنید (شکل ۲۵-۴).



شکل ۲۵-۴- 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 به کار ببرید.

برای اینکه DataList به‌طور خودکار کل فرم را پر کند، کد زیر را 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
```

۱-۱۰-۴- پیکربندی مشخصه‌های کنترل DataList: همان‌طور که بیان شد، کنترل

DataList دارای مشخصه‌های کمتری نسبت به DataGridView است ولی چند ویژگی دارد که کنترل DataGridView ندارد (جدول ۴-۶).

جدول ۴-۶- 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 اندازه کافی بزرگ نباشد، نشان دهد یا نه؟
MatchEntry	همزمان با تایپ کاربر، DataList براساس نویسه وارد شده به عنصر موردنظر رجوع می‌کند.

۱-۱۱-۴- کاربرد کنترل DataCombo

کنترل DataCombo دقیقاً شبیه کنترل DataList است و فقط ظاهر آن‌ها متفاوت است. این کنترل به جای این که 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.



ایجاد و استفاده از کنترل DataCombo:

- ۱- روی فرم جدید، کنترل ADO Data جدیدی را اضافه کنید که به جدول Customers از بانک اطلاعاتی Nwind.mdb دسترسی داشته باشد. به دلیل این که کنترل‌های DataList و DataCombo با هم هستند، پس کنترل DataCombo روی جعبه ابزار وجود خواهد داشت.
- ۲- روی کنترل DataCombo دابل کلیک کنید. نام آن را به دلیل این که برای نگهداری اسامی مشتریان مورد استفاده قرار می‌گیرد، dcboCustomers قرار دهید.
- ۳- مشخصه RowsSource کنترل DataCombo را با adoCustomers و مشخصه 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 ۴-۲۶).



شکل ۴-۲۶- کنترل DataCombo محتوای فیلد CompanyName را از جدول customers بارگذاری می‌کند.

مشخصه‌های قابل بیکربندی برای کنترل DataCombo همان مشخصه‌های DataList هستند. تنها تفاوت مشخصه Style است که شبیه مشخصه 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 این که DataCombo اغلب برای انتخاب مقادیر از لیست، مورد استفاده قرار می‌گیرد، معمولاً مقدار مشخصه Style آن را 2-DropDown List انتخاب می‌کنیم.

خودآزمایی

- ۱- سه رابط دسترسی به داده‌ها در ویژوال بیسیک، کدامند؟
 - ۲- دلیل استفاده از ADO را توضیح دهید.
 - ۳- از SQL به چه منظوری استفاده می‌شود؟
 - ۴- در SQL از چه عملگری برای جستجوی مقادیر براساس یک الگو استفاده می‌شود؟
 - ۵- دو مشخصه‌ای که برای اتصال کنترل ADO Data به بانک اطلاعاتی باید مقداردهی شوند، کدامند؟
 - ۶- دو روش برای به‌هنگام کردن رکوردها را شرح دهید.
 - ۷- پروژه‌ای طراحی کنید که 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.

کاربرد نرم افزار Crystal Report

اهداف رفتاری: پس از آموزش این فصل هنرجو می تواند:

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

هر نرم افزاری که داده ها را ذخیره می کند نیاز به بازیابی این داده ها و ایجاد گزارشات مناسب است. گزارش را می توان به عنوان پاسخی به پرسش کاربر از داده های خود در نظر گرفت. نرم افزار Crystal Report یکی از ابزارهای قدرتمند تهیه گزارش است و با این نرم افزار می توان گزارشات پیشرفته و پیچیده ای را از بانک های اطلاعاتی Access – SQL – Olap – Oracle – Excel – Server و ... تهیه نمود. طراحی خروجی گزارشات می تواند به راحتی به وسیله Crystal Report صورت گیرد به طوری که نظر و علاقه هر کاربری را تأمین نماید. این نرم افزار می تواند به طور مستقل و یا از طریق نرم افزار Visual Basic گزارش گیری را ایجاد، ویرایش و نمایش دهد.


۵-۱- مراحل ایجاد یک گزارش

برای اجرای نرم افزار از طریق مسیر زیر اقدام می شود.

Start | All Program | Crystal Reports9 Tools| Crystal Reports9

و برای اجرای نرم افزار از درون VisualBasic به صورت زیر اقدام می شود.

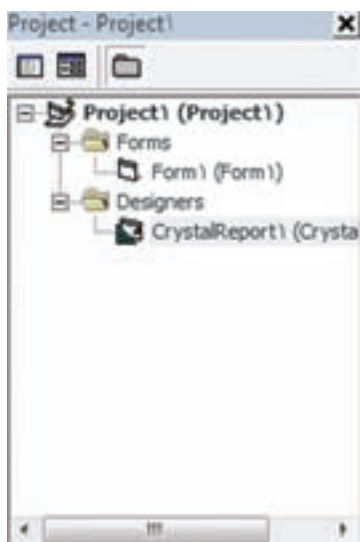
Project Menu | Add Crystal Report 9

 **نکته:** در صورتی که بعد از نصب کریستال گزینه Add Crystal Report 9 مشاهده نشود از طریق

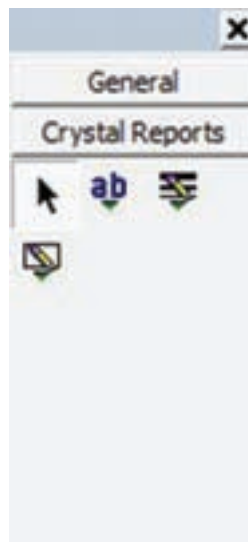
Project Menu | Components | Designers Tab

گزینه Crystal Report 9 را تیک دار نمایید.

که در این روش در کنار فرم های برنامه یک فرم گزارش گیری در زیرشاخه Designers برای Crystal Report (شکل ۵-۱) و در Tool Box، ابزار این نرم افزار نمایش داده می شود (شکل ۵-۲). این نوع فایل ها با پسوند rpt . ذخیره می شود.



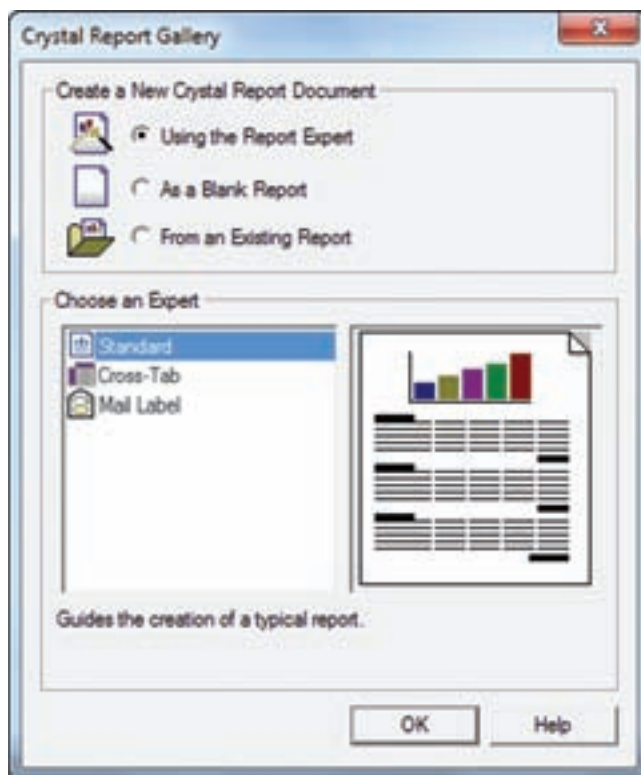
شکل ۵-۱



شکل ۵-۲

با انتخاب CrystalReport در Visual Basic مراحل ایجاد گزارش گیری آغاز می شود. اولین کادر محاوره ای Crystal Report Gallery می باشد که نوع سند گزارش گیری توسط کاربر تعیین می شود (شکل ۳-۵).

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

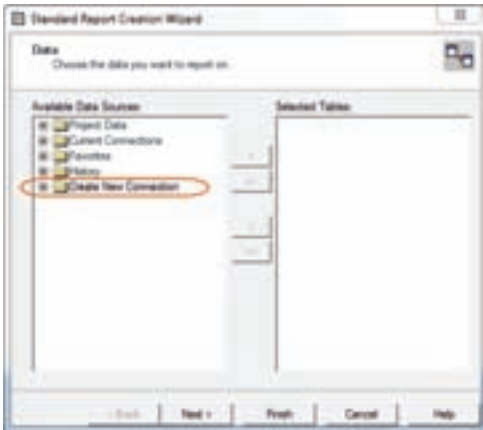


شکل ۳-۵

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

۵-۲- انتخاب یک منبع داده و ارتباط با آن

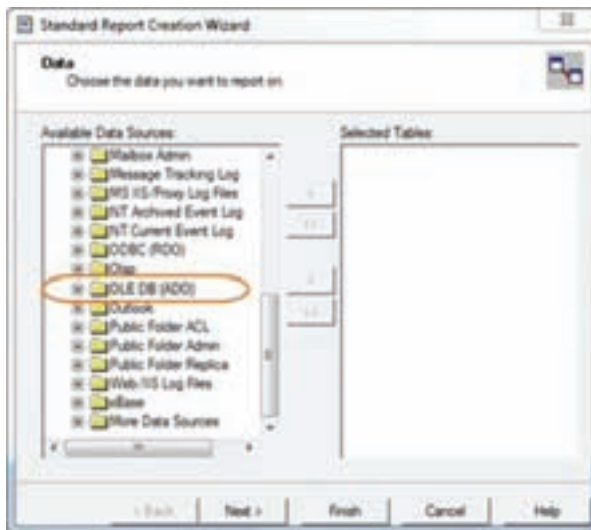
گام بعدی در ایجاد گزارش، انتخاب داده‌هایی است که در گزارش به کار خواهید برد. کادر محاوره‌ای Standard Report Creation Wizard امکان انتخاب منبع داده را فراهم می‌نماید (شکل ۵-۴).



شکل ۵-۴

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

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



شکل ۵-۵

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 ۵-۶).



شکل ۵-۶

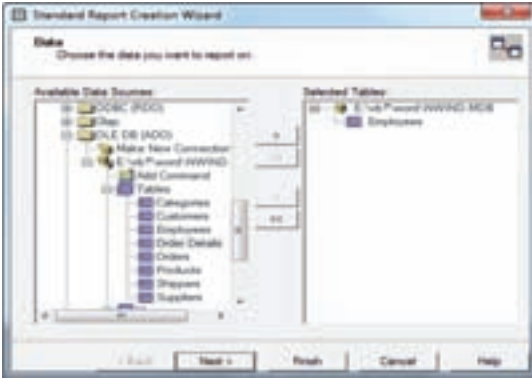
در این مرحله محل بانک اطلاعاتی (بانک Access با پسوند .mdb) را با کلیک بر روی ... تعیین نمایید و در صورتیکه بانک اطلاعاتی با نام کاربر و رمز عبور خاصی وجود دارد، آن‌ها را تعیین و در غیر اینصورت همان اطلاعات پیش فرض را تغییر نداده و Finish را کلیک کنید (شکل ۵-۷).



شکل ۵-۷

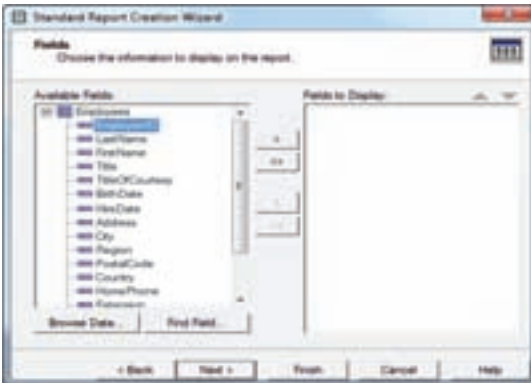
۵-۳- اضافه کردن جدول و فیلدها

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 ۵-۸).



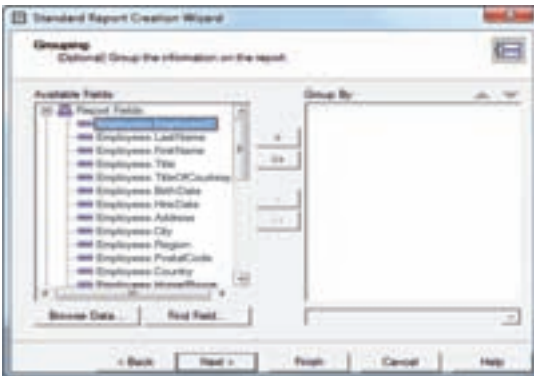
شکل ۵-۸

در این مرحله فیلدهایی که می‌خواهید در گزارش مشاهده شود را انتخاب و به لیست سمت راست اضافه نمایید و کلید Next را فشار دهید (شکل ۵-۹).



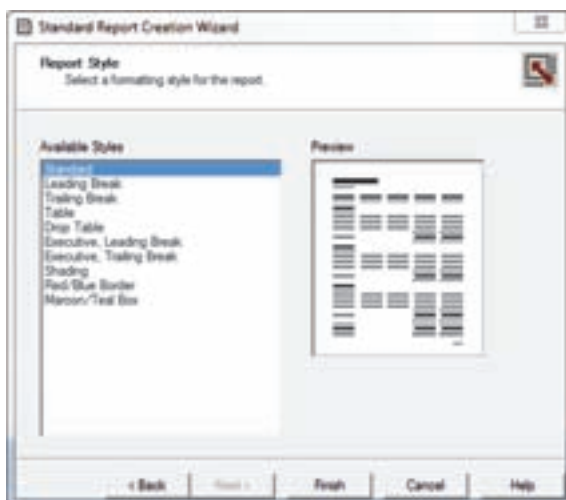
شکل ۵-۹

در مرحله دسته‌بندی (Grouping) و مرتب‌سازی اطلاعات، می‌توان بدون انتخاب گزینه‌ای به مرحله بعد (Next) رفت (شکل ۵-۱۰).



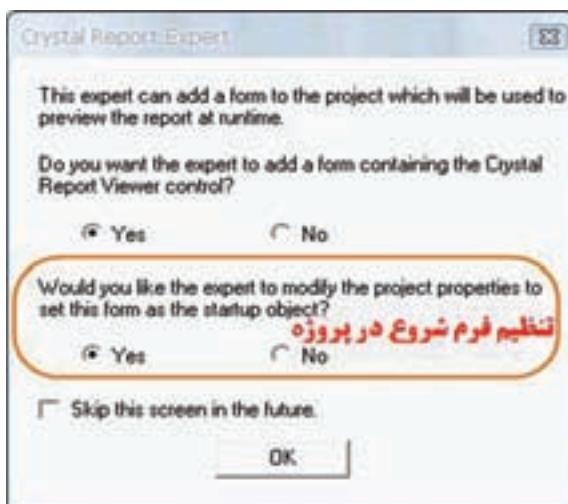
شکل ۵-۱۰

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 ۵-۱۱).



شکل ۵-۱۱

برای اتمام کار شکل ۵-۱۲ نمایش داده می‌شود و اعلام می‌کند، فرمی برای خروجی گزارش به پروژه اضافه می‌شود.



شکل ۵-۱۲

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

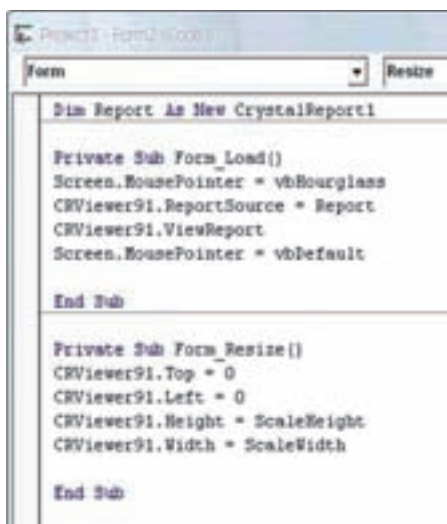
در صورت انتخاب Yes: به صورت اتوماتیک یک فرم در کنار فرم‌های دیگر در محیط برنامه‌نویسی Visual Basic طراحی شده و پیش‌نمایش (Preview) گزارش مشاهده می‌شود و کدهای ارتباط با گزارش کریستال تولید می‌شود (شکل ۱۳-۵).

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

در صورت انتخاب Yes: به صورت 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

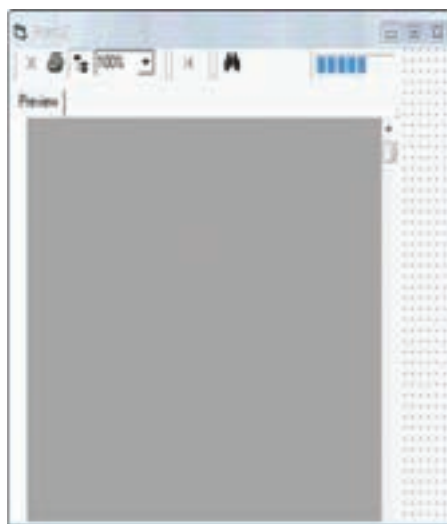
Startup object | Project Properties | Project | منوی



```
Form
Dim Report As New CrystalReport1

Private Sub Form_Load()
    Screen.MousePointer = vbHourglass
    CRViewer91.ReportSource = Report
    CRViewer91.ViewReport
    Screen.MousePointer = vbDefault
End Sub

Private Sub Form_Resize()
    CRViewer91.Top = 0
    CRViewer91.Left = 0
    CRViewer91.Height = ScaleHeight
    CRViewer91.Width = ScaleWidth
End Sub
```



شکل ۱۳-۵

۴-۵- مد طراحی

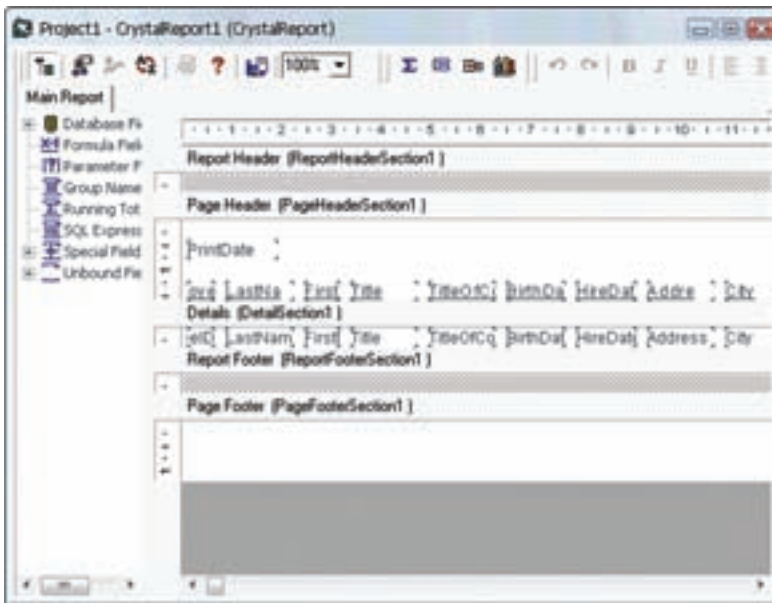
در صورتی که بخواهید ظاهر صفحه و فیلدهای گزارش را تغییر دهید می‌توانید از این قسمت استفاده نموده و تغییرات لازم را اعمال نمایید. مانند تغییر سایز و نوع قلم، محل نمایش و چیدمان فیلد، حذف فیلد (انتخاب نام فیلد از Detail و فشردن کلید Delete) و اضافه کردن فیلد (انتخاب و درک کردن فیلد به Detail) و تغییر عنوان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

برای تغییرات طراحی گزارش بر روی فایل CrystalRepoet1 کلیک نماید (شکل ۱۴-۵).
• 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:

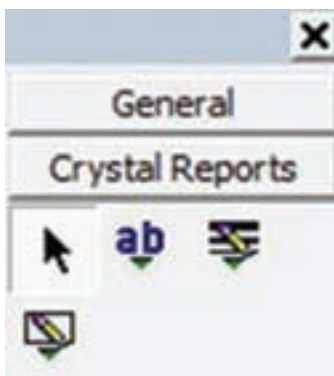
Report Header, Report Footer, Details, Page Header, Page Footer

که در درس بانک اطلاعاتی به‌طور کامل تشریح شده است.



شکل ۱۴-۵

توسط ابزار موجود در Toolbox می‌توانید اشیا، متن، خط، شکل و... را بر روی گزارش خود درج نمایید تا ظاهری زیباتر و کارآمدتر داشته باشید (شکل ۵-۱۵).



شکل ۵-۱۵

۱-۴-۵- 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.

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...

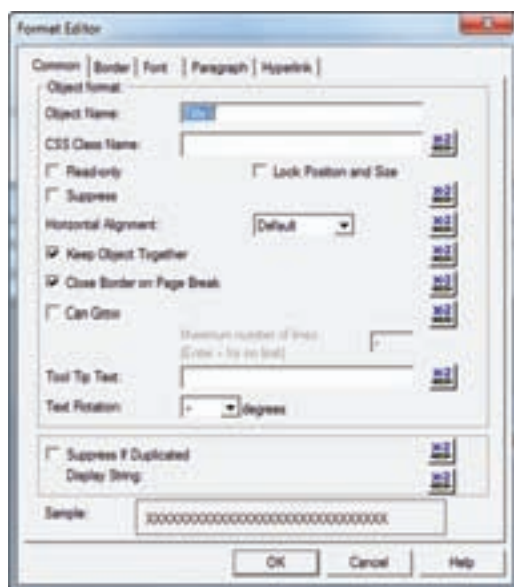
در صورتی که در کنار فیلدی تیک سبز باشد یعنی از آن در طراحی گزارش استفاده شده است.



شکل ۵-۱۶

۲-۴-۵- 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 همه دستورات شرطی در زبان VB در این بخش از برنامه قابل اجرا است. برای مشاهده این امکانات فیلدی از بخش Detail را انتخاب و کلیک راست نموده و گزینه Format Field را انتخاب نمایید (شکل ۱۷-۵).



شکل ۱۷-۵

۵-۵- خصوصیات مشترک در قالب‌بندی انواع فیلدها

۱-۵-۵-۱- 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 بازگرداند.

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 شیء می‌تواند داشته باشد را نیز تعیین نمایید. به صورت 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 ۵-۱

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.

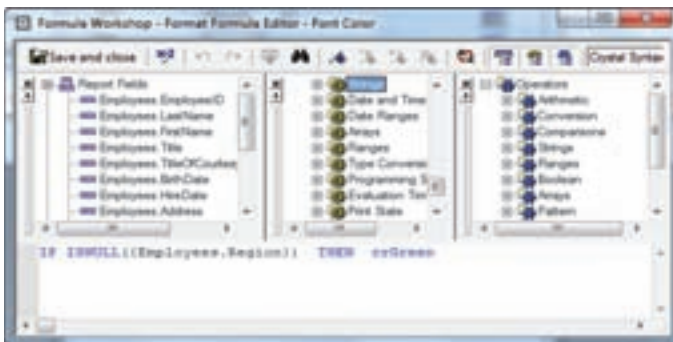
برای انجام این کار گزارشی که از جدول Employees تهیه شده و فیلدهای آن بر روی فرم

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 ۵-۱۸).




شکل ۵-۱۸

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 صورتی که پیغام No Error Founds داده شود، درست بودن دستورات تأیید می‌شود. 

گزینه Save and close را انتخاب تا تغییرات ذخیره شود.

مثال ۲-۵

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
```

در جدول Orders تمام رکوردهایی که فیلد کرایه (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 ۳-۵

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



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.

شکل ۱۹-۵

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.UnitsOnOrder} * {Products.UnitPrice}

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



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

در فرمی که به صورت اتوماتیک توسط کریستال ریپورت برای جدول Products تولید شده است کنترل‌های 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 Bottom 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

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 $\text{Len}()$, $\text{Abs}()$, $\text{Sin}()$, ... و توابع رشته‌ای $\text{Len}()$, $\text{Mid}()$, $\text{Trim}()$, ... و غیره.

یک برنامه کاربردی وظایف خودش را از طریق شیء‌ها 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 به صورت زیر مجموعه و `Weight` خصوصیت می‌باشد؛ همانند شیء فرم که کنترل‌های روی فرم‌های دیگری هستند و هر یک از شیء‌ها 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 شیء‌ها را مشاهده و در هنگام نیاز از آنها استفاده کرد یا آنها را تغییر داد.

۱-۶- 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 و آن را بر روی یک فایل دیگر و یا هر جایی 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 و آنها را در قالب یک Sub مشاهده و در صورت نیاز تغییر یا حذف نمود.

مثال ۱-۶

ایجاد یک ماکرو ساده: یک ماکرو با نام Macro1 در محیط Excel تولید نمایید


(شکل ۱-۶) که فعالیت‌های زیر در آن انجام شود:

- در ردیف اول تولید اعداد اتوماتیک از ۳۰ تا ۳۳
- در ردیف ۳ ستون A، فرمول = Len("Visual")
- در ردیف ۴ ستون A، فرمول = ABS - 2
- در 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 ۱-۶

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

 **نکته:** مسیر مشاهده محیط IDE و ژوال در آفیس (فشار کلیدهای (Alt + F11) : **OFFICE 2003** از منوی Tools زیر منوی Macro و انتخاب گزینه

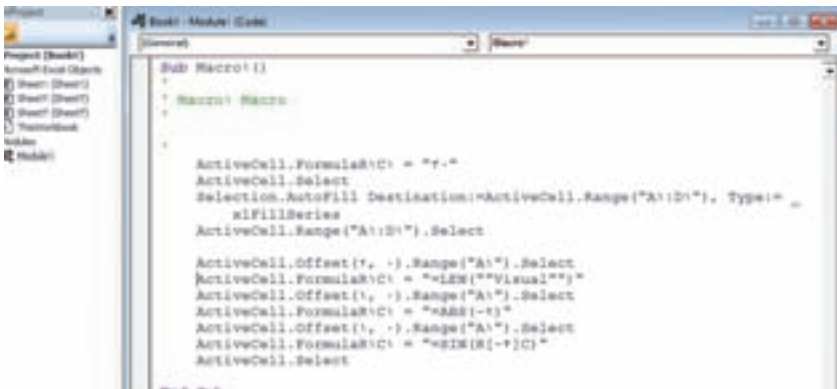
Visual Basic Editor

OFFICE 2007 : از زبانه Developer انتخاب گزینه Visual Basic

(مشاهده Developer با انتخاب گزینه Show Developer tab in the Ribbon

از (Excel Option

شکل ۲-۶ کدهای ماکرو ضبط شده در Moudul را نشان می دهد.



```
Sub Macro1 ()  
Macro Macro1  
  
ActiveCell.FormulaR1C1 = "=r."  
ActiveCell.Select  
Selection.AutoFill Destination:=ActiveCell.Range("A1:D1"), Type:= _  
xlFillSeries  
ActiveCell.Range("A1:D1").Select  
  
ActiveCell.Offset(1, 0).Range("A1").Select  
ActiveCell.FormulaR1C1 = "=LEN("Visual")"  
ActiveCell.Offset(1, 0).Range("A1").Select  
ActiveCell.FormulaR1C1 = "=ABS(-1)"  
ActiveCell.Offset(1, 0).Range("A1").Select  
ActiveCell.FormulaR1C1 = "=SIN(PI/2)"  
ActiveCell.Select
```

شکل ۲-۶

 مثال ۲-۶

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 :

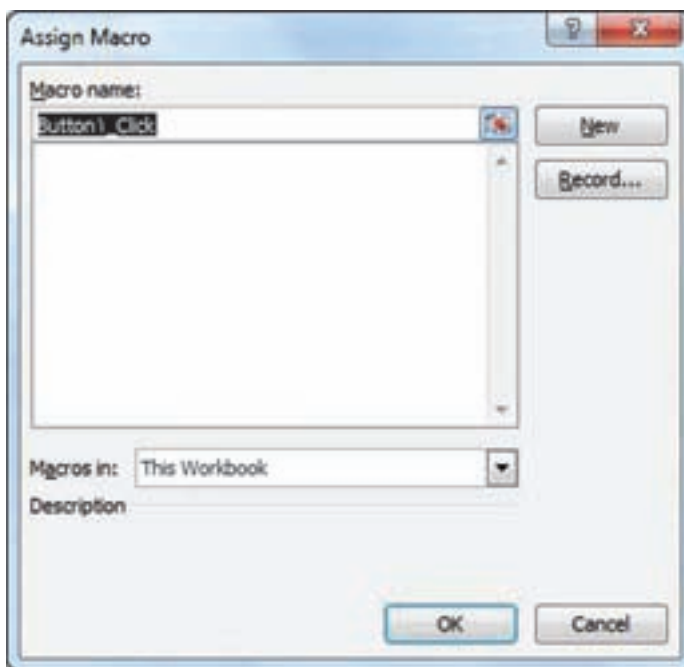
```
ActiveCell.Range("A1").Borders.Weight = xlThin
```

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

 مثال ۳-۶

ایجاد Sheet جدید با نام دلخواه بر روی Work Sheet به وسیله 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 ۶-۳).



شکل ۶-۳

در Modual برای رویداد Button1_Click تکه کد زیر را تایپ نمایید (شکل ۶-۴).



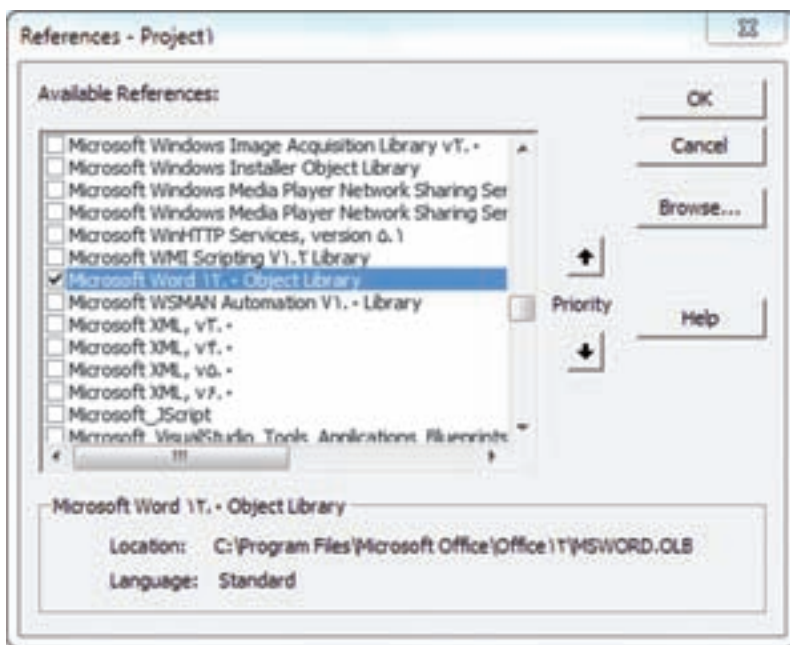
شکل ۶-۴

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

۶-۲- ارتباط Visual Basic با Microsoft Office

نرم افزار، Visual Basic می تواند تمام مؤلفه های ActiveX، فایل کتابخانه ای (olb) نرم افزارهای نصب شده در سیستم عامل را مشاهده و در صورت نیاز به پروژه اضافه و از آن استفاده نماید. وقتی یک فایل کتابخانه ای نرم افزاری انتخاب شود می توان از آن به عنوان یک شیء در محیط برنامه نویسی استفاده نمود. که این شیء شامل خصوصیت ها (properties) و متدها (methods) می باشد. یک فایل کتابخانه ای مانند WORD را از گزینه References در منوی Project انتخاب نمایید (شکل ۶-۵).

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



شکل ۶-۵ - انتخاب فایل کتابخانه ای WORD

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

۳-۶- بعضی از شیء‌های مورد استفاده در نرم‌افزار Word

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

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



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 ۶-۶).



شکل ۶-۶



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 فصل ۱ دو Command Button با نام‌های 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 می توان به یک فایل اکسل موجود اطلاعات دیگری اضافه نمود.

خودآزمایی

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



API 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.
- نحوه اتصال ویزوال بیسیک به روال‌های API ویندوز از طریق دستور Declare را توضیح دهد.

در این فصل، نحوه دسترسی به روال‌های داخلی ویندوز شرح داده می‌شود. هرچند ویزوال بیسیک می‌تواند تقریباً همه کارهای مورد نیاز شما را انجام دهد، ولی در برخی از برنامه‌ها اگر بخواهید از ویزوال 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 ویزوال 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 نحوه دسترسی به این روال‌های ویندوز توضیح داده می‌شود بلکه تعدادی از این روال‌ها که می‌توانید با آن‌ها کار کنید، بررسی می‌شوند.

۱-۷-API ویندوز

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

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 ویرال 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\System و Windows\Windows 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 ویرال 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 کاربری 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 رابط نرم 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

ویرال بیسیک خودتان فراخوانی کنید. با یک نگاه کوتاه به پوشه‌های Windows\System و Windows\Windows 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 برنامه جدیدی را در سیستم خودتان نصب می‌کنید، این برنامه DLL خاص خودش را ارائه می‌کند. بنابراین، در طول زمان تعداد زیادی پرونده DLL روی سیستم خواهید داشت.

۲-۷- 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. برنامه 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.

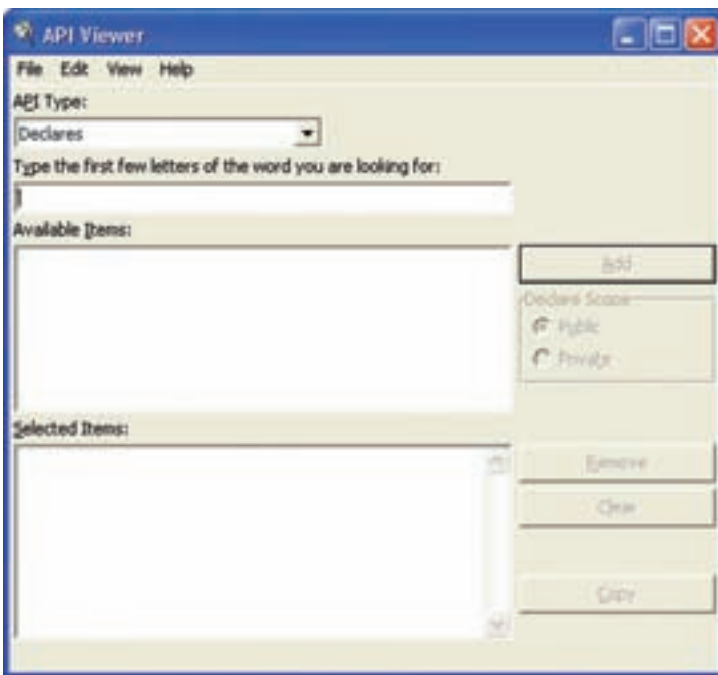
۳-۷- دستور Declare

برای فراخوانی روال‌های API ویندوز باید از یک دستور خاص با نام Declare استفاده کنید. در مورد توابع داخلی و 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.

۱-۳-۷- 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.

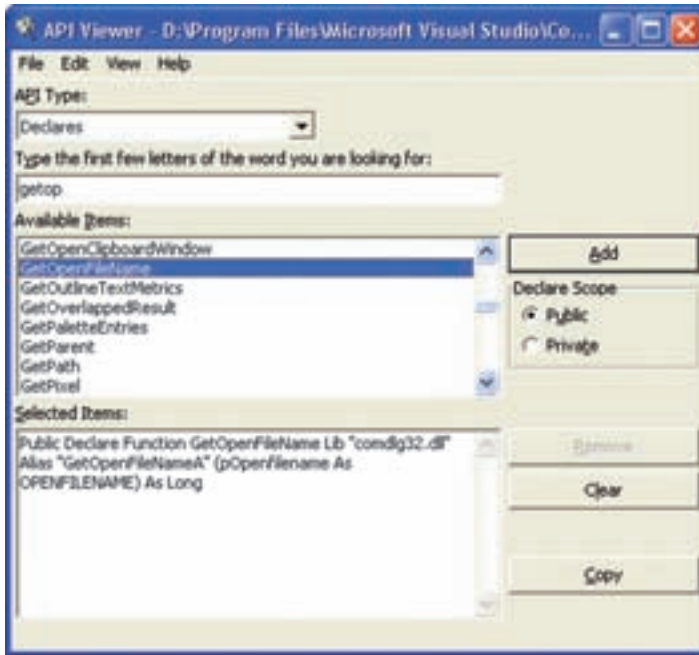
ابزار API Viewer، روال‌های API را نمایش می‌دهد و آن‌ها را از نظر موضوع دسته‌بندی می‌کند طوری که می‌توانید روال‌های مورد نیاز خود را به سادگی پیدا کنید.

دکمه Copy روی API Viewer اطلاعات مربوط به اعلان انتخاب شده را درون Clipboard ویندوز کپی می‌کند. همچنین اگر قبل از کلیک کردن روی دکمه 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 Tools و سپس گزینه Microsoft Visual Studio 6.0 در پایان API Text Viewer را انتخاب کنید.



شکل ۱-۷- می‌توانید از طریق APIViewer به سادگی قالب روال‌های API را مشاهده کنید.

API Viewer اطلاعات اساسی خود را از پرونده‌های متنی MAPI32.txt، APILOAD.txt و WIN32API.txt پیدا می‌کند. این پرونده‌ها همراه API Viewer روی سیستم نصب می‌شوند. از آنجایی که اکثر روال‌های API که مورد نظر شما هستند، در پرونده WIN32API.txt قرار دارند. گزینه Load Text File از منوی File پنجره API Viewer انتخاب و سپس پرونده WIN32API.txt را انتخاب کنید.



شکل ۲-۷

دقت کنید کادر لیست موجود در بالای پنجره دارای عنوان API Type است. با باز کردن این کادر لیست، سه مقدار زیر را مشاهده می کنید :

● **Constants:** همه ثابت‌های نامگذاری شده که پرونده API بارگذاری شده تشخیص می‌دهد را فهرست می‌کند.

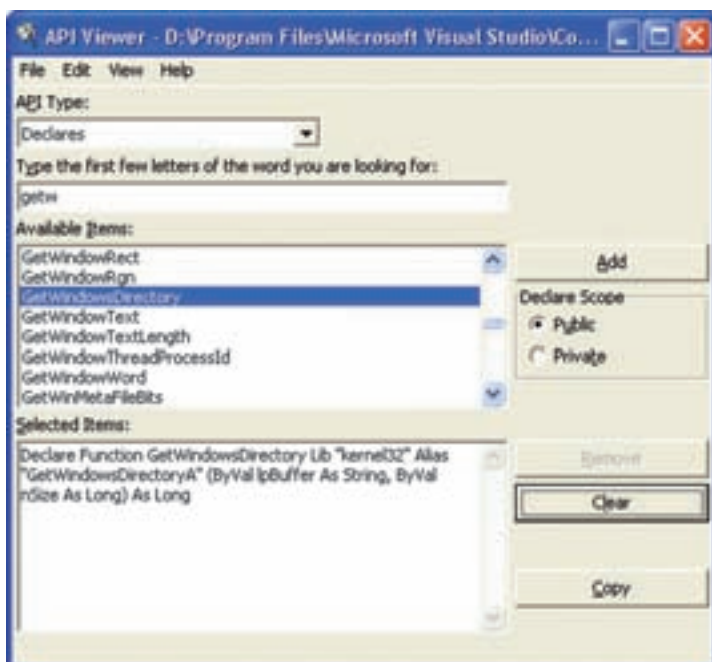
● **Declares:** همه اعلان‌هایی که درون پرونده API بارگذاری شده ظاهر می‌شوند را فهرست می‌کند.

● **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 :

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

- ۲- 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.
- ۳- این فهرست را مرور کنید تا گزینه GetWindowsDirectory را پیدا کنید.
- ۴- روی گزینه GetWindowsDirectory دابل کلیک کنید تا دستور Declare مورد نیاز تابع مطابق شکل ۳-۷ نمایش داده شود.



شکل ۳-۷- 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.

۴-۷- تعریف تابع API

قبل از این که 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 به کار 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 Dll ای 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 ByRef، ByVal، 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 Reference) 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.

Data – type نوع داده‌ای است که تابع برمی‌گرداند.

۷-۴-۱- 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_beep'
```

```
(byval wtype as long) as long
```


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

مثال ۷-۱

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 ۷-۲

نمایش نام کاربر ویندوز : فرمی به همراه یک کادر متن ایجاد کنید و کد صفحه 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
```

رشته‌ای به طول ۲۵۵ ایجاد کرده و آن را با کاراکتری با کد صفر پر می‌کنیم (255,0) String Buffer =

```

    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.



تشخیص بزرگی و کوچکی و نوع (عددی - 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 = "alphanumeric"

```

```

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 IsCharAlpha مساوی True است و اگر حرفی یا عددی باشد خروجی تابع IsCharAlpha Numeric مساوی True است و اگر کلید حرفی فشرده شده از حروف بزرگ باشد (مثل A) خروجی تابع IsCharUpper مساوی True و اگر از حروف کوچک باشد مثل a خروجی تابع IsCharLower مساوی True است.



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 hwndInsertAfter 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

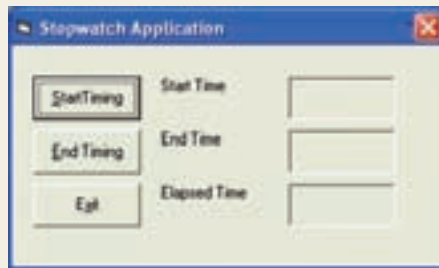
(توضیح: تمام تمرین‌های این فصل را با استفاده از توابع API بنویسید.)

۱- API چیست؟ کاربرد آن را شرح دهید.

۲- انواع پرونده‌های DLL را نام ببرید و کاربرد هر کدام را توضیح دهید.

۳- فرمی به شکل زیر طراحی کرده و مشخصه‌های آن را تنظیم کنید. با کلیک روی

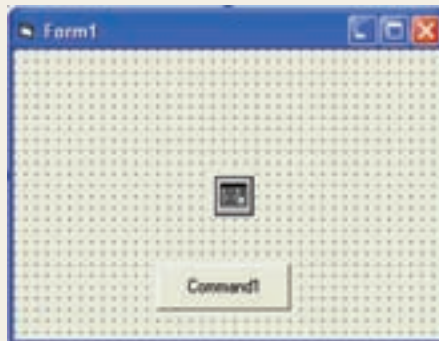
دکمه‌های Start Timing و End Timing زمان سیستم خوانده شده و در برچسب‌های مقابل آن‌ها نمایش داده می‌شود. سپس اختلاف آن‌ها محاسبه و در برچسب سوم نمایش داده می‌شود.



شکل ۴-۷

۴- به وسیله شیء CommonDialog یک پرونده صوتی از نوع WAVE را

انتخاب و به وسیله تابع SndPlaySound آن را پخش کنید.



شکل ۵-۷

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



شکل ۶-۷

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

GetDesktopWindow() (۱)

GetDC() (۲)

PaintDesktop() (۳)

ReleaseDC() (۴)

SearchBlit() (۵)

انجام پروژه پایانی

مهارت در برنامه‌نویسی با تکرار و تمرین برنامه‌نویسی به صورت عملی امکان‌پذیر است، بدین منظور دو هفته از زمان آموزش این درس اختصاص به انجام یک پروژه داده شده است. در این فصل، پروژه‌های برنامه‌نویسی نمونه ارائه می‌شوند. برای پرهیز از افزایش حجم کتاب، خلاصه‌ای از آن‌ها در زیر شرح داده می‌شود. این پروژه‌ها به عنوان نمونه بوده و پس از تجزیه و تحلیل این پروژه در کلاس، هنرجویان می‌توانند پروژه مورد نظر 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 به کار گیرند.

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

- ۱- بررسی اولیه و شناخت از مسأله و راه‌حل‌های آن (تجزیه و تحلیل پروژه)
- ۲- رعایت اصول و قوانین یک برنامه‌نویسی خوب
- ۳- کنترل دستگاه‌های ورودی و خروجی (صفحه کلید - ماوس و...) در هنگام Data Entry
- ۴- مدیریت فرم‌ها (صفحات ورود اطلاعات) و گزارشات خروجی
- ۵- طراحی یک رابط کاربرپسند (User Friendly Interface) با گرافیک مناسب
- ۶- مدیریت عناصر چندرسانه‌ای (صوت، تصویر، فیلم، پویانمایی) در پروژه (کنترل بخش و قطع آن‌ها)
- ۷- به کارگیری ماژول‌های مناسب اکتیوایکس و API در صورت نیاز در پروژه
- ۸- مدیریت کاربران پروژه
- ۹- تهیه و ایجاد برنامه نصب برای پروژه

- ۱۰- تهیه مستندات داخلی و خارجی برای پروژه و ارائه User Manual
- ۱۱- مدیریت کامل بانک اطلاعاتی در پروژه (درج - حذف - ویرایش - جستجو)
- ۱۲- ارائه پروژه در کلاس به صورت 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:

۱- مدیر

- قابلیت ویرایش نام و رمز مدیر

(توجه: پیش فرض 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

۲- دفتردار

- قابلیت افزایش دانش‌آموز جدید

- قابلیت ویرایش دانش‌آموز

- قابلیت جستجوی دانش‌آموز بر اساس نام و نام خانوادگی 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.

۳- معاون

- قابلیت مشاهده مشخصات دانش‌آموز

- قابلیت ویرایش وضعیت تحصیلی

- قابلیت جستجوی دانش‌آموز بر اساس نام و نام خانوادگی 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 به‌وسیله دو تابع DeCode و EnCode رمزگشایی و رمزگذاری می‌شوند. در ضمن رمز بانک ۱۲۳۴ است.

(ب) مدیریت کتابخانه مدرسه

● کتاب ● اعضا ● کارمندان

برای هر کدام 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 ● مشتریان

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 به‌عهده هنرجو واگذار می‌کنیم.

آشنایی با MSDN

(Microsoft Developer Network)

MSDN راهنمای کامل برنامه های میکروسافت می باشد و می توانید 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. نسخه های بالاتر از آن با شاخه .NET. ارتباط دارد.

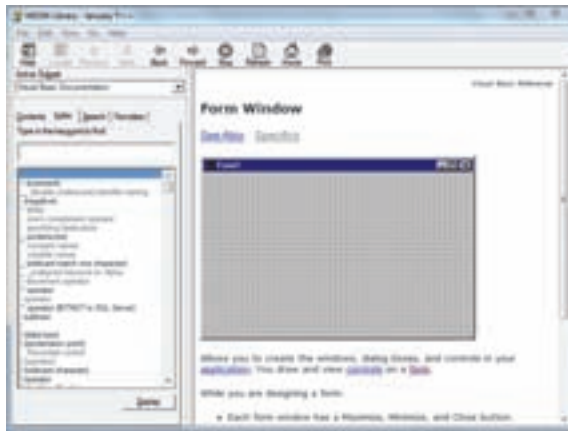
این بسته در قالب یک DVD یا ۳ CD ارائه می شود و در صورت نیاز می توانید آنرا از سایت [Microsoft](http://msdn.microsoft.com/subscriptions) دانلود نمایید.



در هنگام نصب حتماً گزینه Full را انتخاب نمایید تا تمام قسمت های موجود روی ۳ CD بر روی دستگاه مقصد نصب شود در غیر این صورت برای جستجوی هر مطلبی به CD های نصب نیازمندید (شکل ۹-۱)

شکل ۹-۱

بعد از نصب کامل MSDN می‌توان با زدن کلید F1 در محیط Visual Basic محیط راهنما را مشاهده نمود (شکل ۹-۲).



شکل ۹-۲

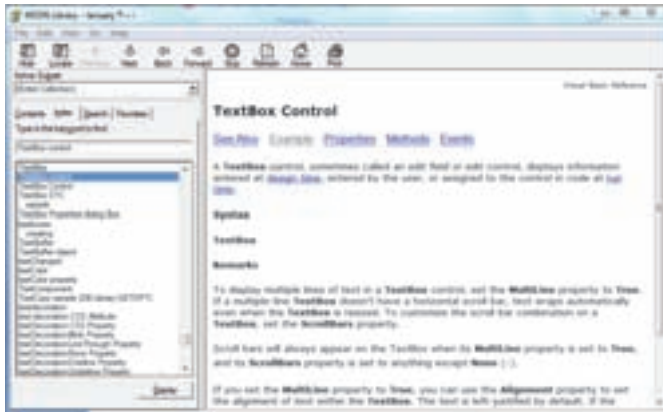
گزینه های موجود در MSDN

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



شکل ۹-۳

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

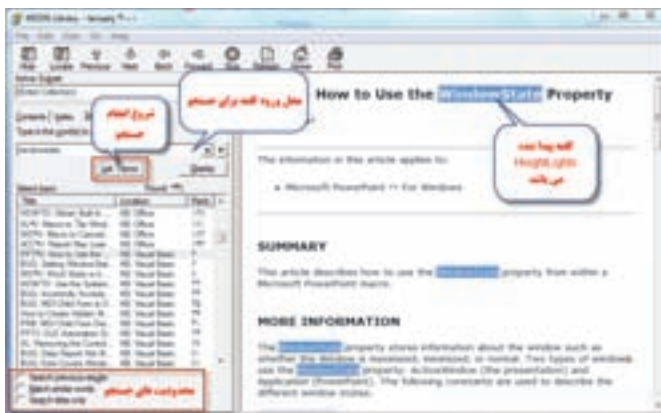


شکل ۹-۴

زبان Search: برای جستجوی کلمه ای در Msdn می توان آنرا تایپ نمود و حداکثر تا ۵۰۰ مورد، نمایش داده می شود.

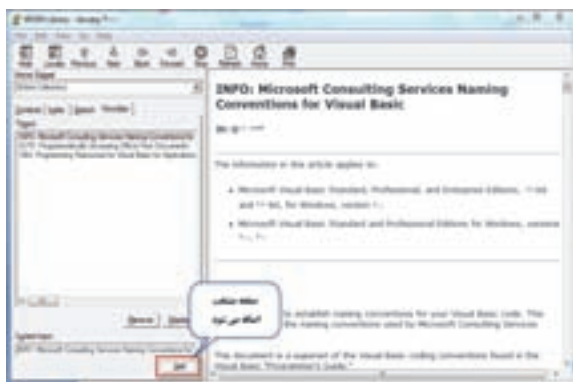
مرتب سازی مطالب یافت شده بر اساس سه آیتام می باشد: عنوان صفحه (Title) و محل صفحه (Location) و رتبه صفحه (Rank)

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)



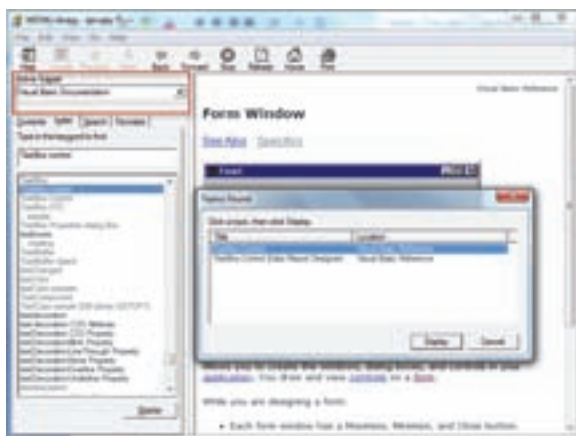
شکل ۹-۵

زبانۀ 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 ۹-۶

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

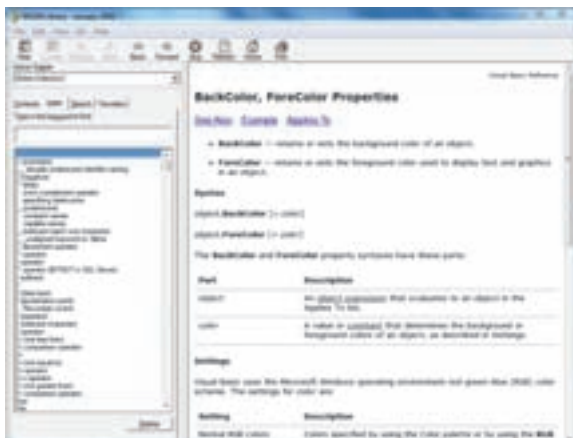


شکل ۹-۷

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 می‌نماییم. به‌طور مثال در کد نویسی یک دکمه فرمان دستور زیر تایپ می‌شود :

cmdRun.BackColor

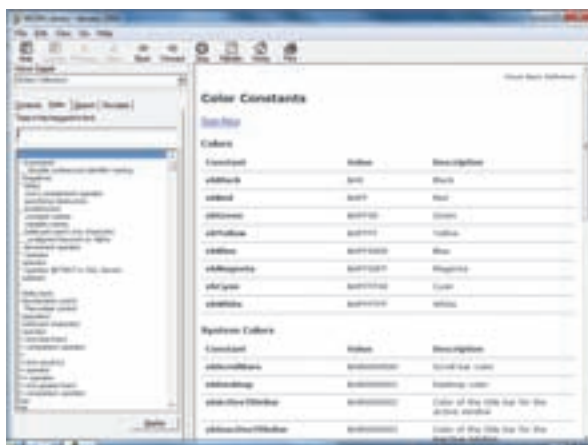
می‌توان برای اطلاعات بیشتر از MSDN استفاده نمود. مکان‌نمای چشم‌کزن را بر روی خصوصیت BackColor انتقال و کلید F1 را فشار دهیم. راهنمای MSDN باز می‌شود و توضیحات لازم و حتی مثال درباره این خصوصیت مشاهده می‌شود.



شکل ۸-۹

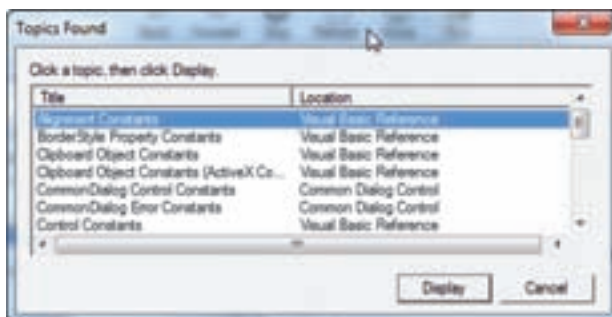
کد نویسی را ادامه داده و رنگ مورد نظر را تایپ می‌نماییم، بر روی رنگ کلید F1 را فشار داده و تمام ثابت‌های رنگ در راهنما مشاهده می‌شود.

cmdRun.BackColor=vbRed



شکل ۹-۹

برای مشاهده دیگر نابتها در خصوصیت های دیگر، بر روی See Also کلیک نمایید.



شکل ۱۰-۹

منابع

- 1_ Sams Teach yourself Visual Basic 6 in 21 Days, Greg Perry .
- 2_ Visual Basic 6 How To Program, Deitel & Deitel.
- 3_ Learn Visual Basic 6, Lou Tylee.
- 4_ Using Visual Basic 6, Bob Resel man, . . .
- 5_ MCSD Visual Basic 6, Microsoft.
- 6_ Desktop Applications With Visual Basic 6.0, Microsoft press.

فهرست کتاب‌های آموزشی مناسب دوره متوسطه حوزه کامپیوتر

- ۱- آموزش گام به گام Word 2007، آرزو خسروپور، ابوالفضل طاهریان، ۱۳۸۶، ناشر طاهریان.
- ۲- آموزش گام به گام Excell 2007، آرزو خسروپور، ابوالفضل طاهریان، ۱۳۸۶، ناشر طاهریان.
- ۳- آموزش کامل اکسل 2007، داریوش فرسای، ۱۳۸۷، نشر علوم و فنون.
- ۴- آموزش تضمینی Photoshop cs3، حسین چناری، سیدجواد سیدمحسنی، ۱۳۸۷، نشر راه نوین.
- ۵- طراحی شبکه‌های کامپیوتری، محمد هادی امامی شوشتری، ۱۳۸۶، نشر جلوه.
- ۶- برنامه‌سازی جلد ۱ و ۲، محمدرضا عابدینی، ۱۳۸۶، نشر جنگل.
- ۷- کتاب کار و تمرین زبان تخصصی کامپیوتر، رمضانعلی یاسکی، مهدی احمدی، ۱۳۸۷، نشر پیام دانش روز.
- ۸- آموزش گام به گام کامپیوتر و ویندوز XP، آرزو خسروپور، ابوالفضل طاهریان، ۱۳۸۷، ناشر طاهریان.
- ۹- کلاس درس A dobe Flash cs3، در یک کتاب، پیمان دانش اشراقی، ۱۳۸۷، نشر صفار- اشراقی.
- ۱۰- آموزش تصویری طراحی صفحات وب با PHP، رضا سرابی میانجی، ۱۳۸۶، نشر تمثیل.
- ۱۱- آموزش تصویری Photoshop cs3، حبیب فروزنده، عابدکفاش، ۱۳۸۴، نشر عابد.
- ۱۲- آموزش تصویری Windows vista، حسین جوهرچی، ۱۳۸۶، نشر عابد.
- ۱۳- آموزش Access 2002، ربابه واثق رحیم‌پور، ۱۳۸۵، نشر کتابخانه فرهنگ.
- ۱۴- آموزش اصول عیب‌یابی رایانه‌های شخصی ۱ و ۲، فاطمه بخشیان و دیگران، ۱۳۸۶، نشر گستره علم و فن.



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