بسم الله الرحمن الرحیم

برنامه‌نویسی ویژوال‌بیسیک پیشرفته

(جلد اول)

شاخه: کاردانش

زمینه: خدمات

گروه تحصیلی: کامپیوتر

زیرگروه: کامپیوتر

رشتهٔ مهارتی: برنامه‌نویسی پایگاه داده

شمارهٔ رشته مهارتی: ۳۱۵-۱۰۱-۱۷-۳

کد رایانه‌ای رشتهٔ مهارتی: ۶۱۴۰

نام استاندارد مهارت مبنا: برنامه‌نویسی، (VB، DELPHI)

کد استاندارد متولی: ۰-۸۴/۸۰/۱/۳/۳

شمارهٔ درس: نظری: ۰۴۹۴ **عملی:** ۰۴۹۵

عنوان و نام پدیدآور: برنامه‌نویسی ویژوال بیسیک پیشرفته [کتابهای درسی] ... کد استاندارد متولی ۰-۸۴/۸۰/۱/۳/۳ مؤلف منصور ولی‌نژاد؛ وزارت آموزش و پرورش، سازمان پژوهش و برنامه‌ریزی آموزشی؛ برنامه‌ریزی محتوا و نظارت بر تألیف، دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش.
مشخصات نشر: تهران: شرکت چاپ و نشر کتاب‌های درسی ایران، ۱۳۹۴.
مشخصات ظاهری: ۲ج.: مصور، جدول.
شابک: 978-964-05-2160-1
وضعیت فهرست‌نویسی: فیپا
یادداشت: ج.۱ (چاپ اول: ۱۳۸۹) (فیپا).
یادداشت: ج.۱ (چاپ دوم: ۱۳۹۰) (فیپا).
موضوع: ویژوال بیسیک (زبان برنامه‌نویسی کامپیوتر)
موضوع: بیسیک (زبان برنامه‌نویسی کامپیوتر)
شناسهٔ افزوده: ولی‌نژاد، منصور، ۱۳۴۵-
شناسهٔ افزوده: سازمان پژوهش و برنامه‌ریزی آموزشی. دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش.
شناسهٔ افزوده: سازمان پژوهش و برنامه‌ریزی آموزشی
رده‌بندی کنگره: ۱۳۸۹ ۴۶ب۹و/QA۷۶/۷۳
رده‌بندی دیویی: ۰۸۴/۸۰/۱۳/۳/۳ک/۳۷۳
شماره کتابشناسی ملی: ۲۰۷۰۵۶۲

۱۳۹۴

وزارت آموزش و پرورش

سازمان پژوهش و برنامه‌ریزی آموزشی

همکاران محترم و دانش‌آموزان عزیز:

پیشنهادها و نظرهای خود را درباره محتوای این کتاب به نشانی: تهران- صندوق پستی شماره ۴۸۷۴/۱۵ دفتر تألیف کتاب‌های درسی فنی‌وحرفه‌ای و کاردانش، ارسال فرمایند.

پیام‌نگار (ایمیل) tvoccd@roshd.ir وب‌گاه (وب‌سایت) www.tvoccd.medu.ir

برنامه‌ریزی محتوا و نظارت بر تألیف: دفتر تألیف کتاب‌های درسی فنی‌وحرفه‌ای و کاردانش

عنوان و شماره کتاب: برنامه‌نویسی ویژوال‌بیسیک پیشرفته (جلد اول)- ۶۱۲/۸

شمارهٔ درس: ۰۴۹۴ ، ۰۴۹۵

مؤلف: منصور ولی‌نژاد

ویراستار ادبی: شیوا غمگسار

صفحه‌آرا: احمد یوسفی دلچه، آرزو مهدوی

طراح جلد: بیتا اشرفی مقدم

محتوای این کتاب در نوزدهمین جلسه مورخ ۸۹/۳/۱۸ کمیسیون تخصصی رشته کامپیوتر دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش با عضویت: بتول عطاران، محمدرضا شکرریز، سید حمیدرضا ضیایی، افشین اکبری، فرنگیس شاکری و حسن رحیمی مقدم تأیید شده است.

نوبت و سال چاپ: چاپ ششم ۱۳۹۴

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

نشانی ناشر: شرکت چاپ و نشر کتاب‌های درسی ایران: تهران ـ کیلومتر ۱۷ جادهٔ مخصوص کرج ـ خیابان ۶۱ (داروپخش)

تلفن: ۵-۴۴۹۸۵۱۶۱ **دورنگار:** ۴۴۹۸۵۱۶۰ **صندوق پستی:** ۱۳۹ـ ۳۷۵۱۵

نظارت بر چاپ و توزیع: اداره کل نظارت بر نشر و توزیع مواد آموزشی، سازمان پژوهش و برنامه‌ریزی آموزشی

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

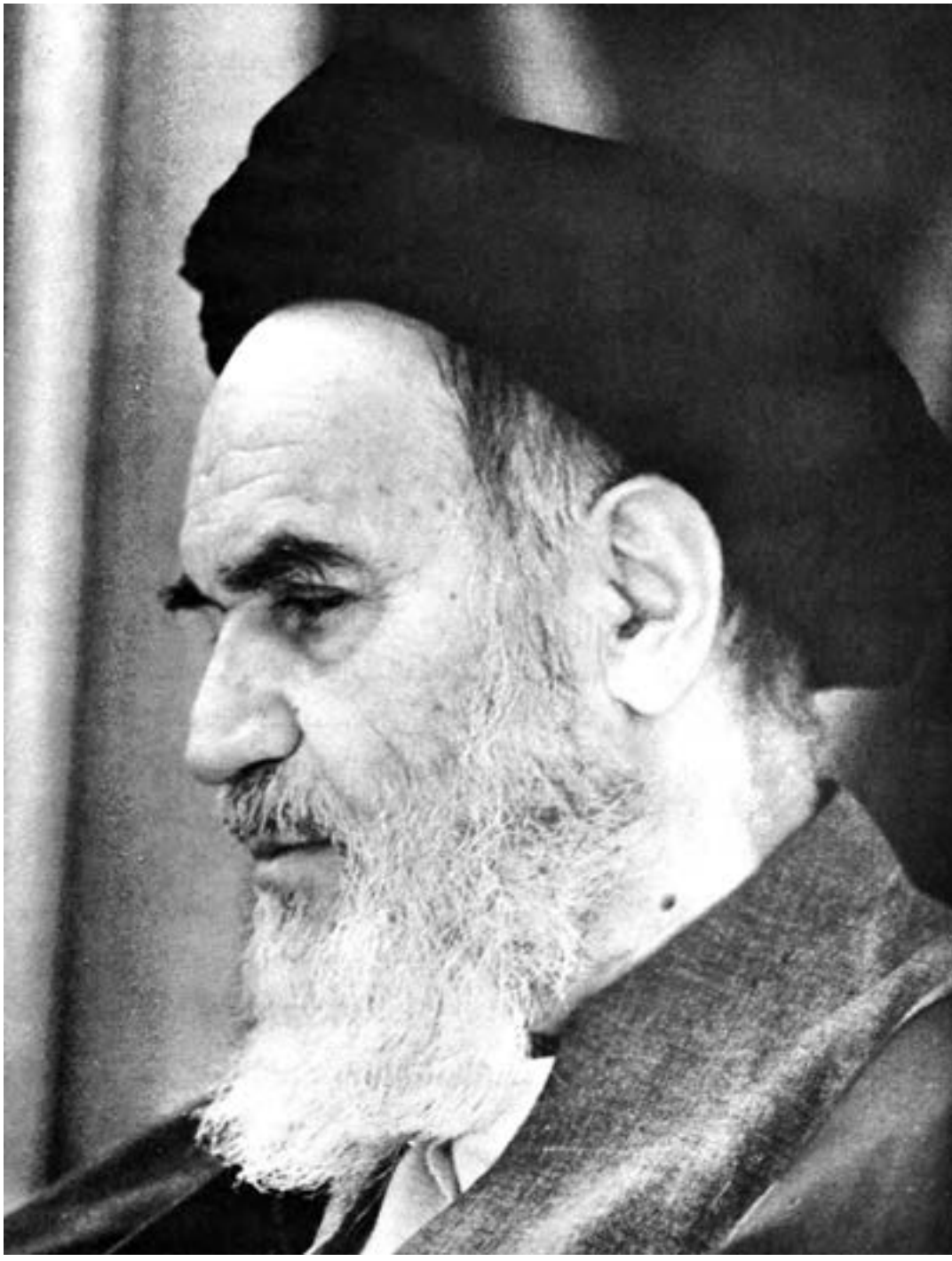
تلفن: ۹-۸۸۸۳۱۱۶۱ **دورنگار:** ۸۸۳۰۹۲۶۶ **کدپستی:** ۱۵۸۴۷۴۷۳۵۹

آدرس الکترونیکی: www.chap.sch.ir

شابک: ۱-۲۱۶۰-۰۵-۹۶۴-۹۷۸

ISBN **978-964-05-2160-1**

(حق چاپ محفوظ است)



بدانید مادام که در احتیاجات صنایع پیشرفته، دست خود را پیش دیگران دراز کنید و به دریوزگی عمر را بگذرانید، قدرت ابتکار و پیشرفت در اختراعات در شما شکوفا نخواهد شد.

امام خمینی « قدّس سره الشّریف»

مجموعه کتاب‌های درسی رشته کامپیوتر شاخه کاردانش

(استاندارد وزارت فرهنگ و ارشاد اسلامی)

رشته تصویرسازی	رشته طراحی صفحات وب	رشته تولید چندرسانه‌ای
مفاهیم پایه فناوری اطلاعات	مفاهیم پایه فناوری اطلاعات	مفاهیم پایه فناوری اطلاعات
سیستم‌عامل مقدماتی	سیستم‌عامل مقدماتی	سیستم‌عامل مقدماتی
اطلاعات و ارتباطات	اطلاعات و ارتباطات	اطلاعات و ارتباطات
واژه‌پرداز Word 2007	واژه‌پرداز Word 2007	واژه‌پرداز Word 2007
صفحه گسترده Excel 2007	صفحه گسترده Excel 2007	صفحه گسترده Excel 2007
ارایه مطالب PowerPoint 2007	ارایه مطالب PowerPoint 2007	ارایه مطالب PowerPoint 2007
نرم‌افزارهای اداری تکمیلی	نرم‌افزارهای اداری تکمیلی	نرم‌افزارهای اداری تکمیلی
بانک اطلاعاتی Access 2007	بانک اطلاعاتی Access 2007	بانک اطلاعاتی Access 2007
سیستم‌عامل پیشرفته	سیستم‌عامل پیشرفته	سیستم‌عامل پیشرفته
برنامه‌نویسی مقدماتی	برنامه‌نویسی مقدماتی	برنامه‌نویسی مقدماتی
طراحی امور گرافیکی با رایانه	طراحی امور گرافیکی با رایانه	طراحی امور گرافیکی با رایانه
کاربر FreeHand	کاربر Flash	کاربر Flash
کاربر CorelDraw	طراحی صفحات وب مقدماتی	کاربر Director
	طراحی صفحات وب پیشرفته	میکس رایانه‌ای

مجموعه کتاب‌های درسی رشته کامپیوتر شاخه کاردانش

(استاندارد وزارت کار و امور اجتماعی)

رشته تصویرسازی	رشته طراحی صفحات وب	رشته تولید چندرسانه‌ای	رشته برنامه‌نویسی پایگاه داده
مفاهیم پایه فناوری اطلاعات	مفاهیم پایه فناوری اطلاعات	مفاهیم پایه فناوری اطلاعات	مفاهیم پایه فناوری اطلاعات
سیستم‌عامل مقدماتی	سیستم‌عامل مقدماتی	سیستم‌عامل مقدماتی	سیستم‌عامل مقدماتی
اطلاعات و ارتباطات	اطلاعات و ارتباطات	اطلاعات و ارتباطات	اطلاعات و ارتباطات
سیستم‌عامل پیشرفته	سیستم‌عامل پیشرفته	سیستم‌عامل پیشرفته	سیستم‌عامل پیشرفته
واژه‌پرداز Word 2007	واژه‌پرداز Word 2007	واژه‌پرداز Word 2007	واژه‌پرداز Word 2007
صفحه گسترده Excel 2007	صفحه گسترده Excel 2007	صفحه گسترده Excel 2007	صفحه گسترده Excel 2007
ارایه مطالب PowerPoint 2007	ارایه مطالب PowerPoint 2007	ارایه مطالب PowerPoint 2007	ارایه مطالب PowerPoint 2007
نرم‌افزارهای اداری تکمیلی	نرم‌افزارهای اداری تکمیلی	نرم‌افزارهای اداری تکمیلی	نرم‌افزارهای اداری تکمیلی
طراح گرافیک رایانه‌ای	طراح گرافیک رایانه‌ای	طراح گرافیک رایانه‌ای	برنامه‌نویسی مقدماتی
شهروندالکترونیکی	نرم‌افزار گرافیکی Flash Mx	شهروند الکترونیکی	برنامه‌نویسی ویژوال بیسیک پیشرفته (جلد اول)
نرم‌افزار گرافیکی FreeHand	طراحی مقدماتی صفحات وب	نرم‌افزار گرافیکی Director	برنامه‌نویسی ویژوال بیسیک پیشرفته (جلد دوم)
نرم‌افزار گرافیکی CorelDraw	رایانه کار Interdev	تدوین فیلم و صدا SSP	مدیریت پایگاه داده
نرم‌افزار گرافیکی Flash Mx	رایانه کار DreamWeaver	نرم‌افزار گرافیکی Flash Mx	مهارت عمومی برنامه‌نویسی
	رایانه کار CIW	نرم‌افزار گرافیکی Authorware	

فهرست مطالب

واحدکار سوم: توانایی برنامه‌نویسی به روش شئ‌گرا

واحدکار چهارم: توانایی خطایابی، خطازدایی و رفع اشکال برنامه‌ها

واحدکار پنجم: توانایی استفاده از کنترل‌های پیشرفته

مقدمه

کتاب حاضر برای رفع نیازهای آموزشی پیمانه مهارتی برنامه‌نویسی پیشرفته، تألیف و ارائه شده است که لازم است پس از پیمانه مهارتی برنامه‌نویسی مقدماتی مطالعه شود. کتاب از ۹ واحد کار تشکیل شده است که واحد کار اول تا هفتم به بررسی مباحث پیشرفته در زبان برنامه‌نویسی ویژوال‌بیسیک ۶، واحد کار هشتم به زبان برنامه‌نویسی ویژوال‌بیسیک Net. و واحد کار آخر به زبان برنامه‌نویسی دلفی اختصاص داده شده‌اند. علاوه بر تمرین‌های موجود در هر واحد کار، در پایان هر واحد کار نیز یک آزمون نظری و عملی ارائه شده است تا مطالب مجدداً مرور شوند. در پایان از کلیه همکاران و دانش‌آموزان تقاضا دارم اینجانب را از پیشنهادات و انتقادات ارزشمند خود بهره‌مند نمایند.

مؤلف

vbbook6@yahoo.com

واحد کار اول

هدف جزیی

توانایی پردازش درایوها، پوشه‌ها، فایل‌ها و فایل‌های ترتیبی

زمان (ساعت)	
نظری	عملی
۷	۱۴

هدف‌های رفتاری

پس از مطالعه این واحد کار از فراگیر انتظار می‌رود که:

۱- مدل شئ File System Object را توضیح دهد.

۲- اشیای Drive،Folder و File را توضیح دهد.

۳- بتواند با متدهای CopyFile،CopyFolder،CreatFolder،
DeleteFile،DeleteFolder،DriveExists،FileExists،
GetDriveName، GetDrive، FolderExists،
GetFolder، GetParent FolderName، GetAbsolutePathName، MoveFolder، GetFile،
MoveFile، CreateTextFile،
GetTempName، OpenTextFile و OpenAsTextStream کار کند.

۴- شئ TextStream را توضیح دهد و بتواند با متدهای Write، WriteLine، Read، ReadLine و ReadAll کار کند.

۵- بتواند با کنترل‌های DirListBox، DriveListBox و FileListBox کار کند.

۶- مفهوم دسترسی ترتیبی به فایل‌ها را بیان کند.

۷- بتواند با فایل‌ها به‌صورت ترتیبی کار کند و عملیات خواندن، نوشتن و اضافه کردن داده را انجام دهد.

کلیات

یکی از مسائل مهم در برنامه‌نویسی، ذخیره‌سازی و نگهداری داده‌ها و اطلاعات در حافظه دائم کامپیوتر می‌باشد زیرا داده‌ها در حافظه اصلی کامپیوتر (RAM) به‌صورت موقت ذخیره و نگهداری می‌شوند و با خارج شدن از برنامه داده‌های آن‌ها نیز از حافظه خارج شده و از بین می‌روند. به‌عنوان مثال فرض کنید می‌خواهید بخشی از یک متن را امروز و ادامه آن را در روزهای بعد تایپ کنید، در این صورت نمی‌توان بخش تایپ شده در کنترل کادر متن را مدت زیادی نگهداری نمود زیرا با خارج شدن از برنامه، متن تایپ شده در کادرمتن نیز از بین می‌رود. به‌علاوه گاهی اوقات لازم است در بعضی از برنامه‌ها عملیاتی چون نسخه‌برداری، ایجاد و انتقال فایل‌ها، پوشه‌ها و مدیریت درایوها را انجام داد. ویژوال‌بیسیک به این منظور امکانات مختلفی را ارائه می‌دهد.

۱-۱ مدل شئ FSO (File System Object)

در ویژوال‌بیسیک ابزار جدید و قدرتمندی به نام FSO برای انجام هر گونه عملیات روی فایل‌ها، پوشه‌ها و درایوها ارائه شده است. FSO به شما اجازه می‌دهد تا با استفاده از خواص و متدهای موجود در آن، فایل‌ها، پوشه‌ها و درایوها را مورد پردازش قرار دهید، به عبارت دیگر با استفاده از FSO شما قادر خواهید بود تا فایل‌ها و پوشه‌ها را ایجاد کرده، انتقال داده یا حذف نمایید و در صورت نیاز اطلاعات موردنظر را درباره آن‌ها مانند تاریخ تشکیل یا ویرایش فایل‌ها و پوشه‌ها، ظرفیت درایوها و غیره به‌دست آورید. مدل شئ FSO شامل اشیای زیر می‌باشد:

۱-۱-۱ شئ Drive

این شئ می‌تواند اطلاعات زیادی در مورد درایوهای موجود در سیستم در اختیار شما قرار دهد. این اطلاعات می‌توانند شامل فضای قابل دسترس، فضای کل، شماره سریال، نوع درایو (درایو شبکه، CD ROM، پارتیشن‌های هارددیسک، فلاپی‌دیسک) و نظایر آن باشند.

۱-۱-۲ شئ Folder

این شئ می‌تواند به شما اجازه دهد که پوشه‌ها را ایجاد کرده، حذف کنید یا انتقال دهید. به‌علاوه می‌توانید اطلاعاتی در خصوص نام، مسیر و سایر ویژگی‌های پوشه‌ها به‌دست آورید.

۱-۱-۳ شئ File

توسط این شئ می‌توان فایل‌ها را ایجاد کنید، انتقال دهید یا حذف نمایید؛ در ضمن امکان کسب اطلاعاتی در رابطه با نام، مسیر و سایر ویژگی‌های فایل‌ها نیز به‌دست آورید.

۱-۱-۴ شئ File System Object

شئ اصلی گروه است. تمام متدهایی که به شما اجازه می‌دهند فایل‌ها و پوشه‌ها را ایجاد یا حذف کنید و اطلاعات مورد نیاز را در رابطه با آن‌ها به‌دست آورید در این شئ قرار دارند.

۱-۱-۵ شئ TextStream

این شئ امکان ایجاد فایل‌های متنی و نوشتن و خواندن آن‌ها را فراهم می‌کند.

۱-۲ نحوه ایجاد شئ از مدل شئ FSO

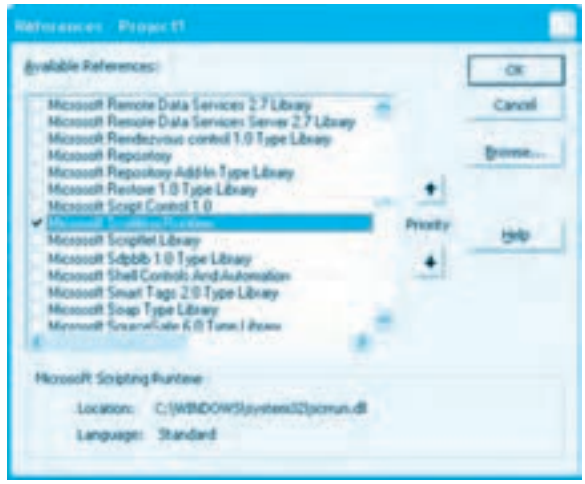
برای استفاده از امکانات موجود در مدل شئ FSO لازم است ابتدا یک نمونه شئ از آن ایجاد کرد، به این منظور می‌توان از دو روش مختلف استفاده نمود:

۱- می‌توانید یک متغیر از نوع FSO ایجاد کنید.

نحوه انجام این کار به‌صورت زیر است:

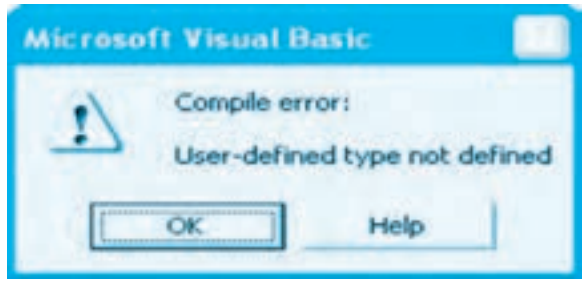
Dim (شئ) نام متغیر As New FileSystemObject

توجه داشته باشید در صورتی‌که از این روش استفاده می‌کنید قبل از اجرای پروژه گزینه Microsoft Scripting Runtime را در کادرمحاوره References انتخاب کنید، به این منظور در پنجره ویژوال‌بیسیک روی منوی Project و سپس روی گزینه References... کلیک کنید تا کادرمحاوره References مطابق شکل ۱-۱ نمایش داده شود.



شکل ۱-۱

در این کادرمحاوره در کادر لیست Available References گزینه Microsoft Scripting Runtime را انتخاب نمایید و در پایان روی دکمه OK کلیک کنید. در صورت عدم انتخاب این گزینه این دستور سبب ایجاد خطا خواهد شد (شکل ۲-۱).



شکل ۲-۱

به‌عنوان مثال به این دستور توجه کنید:

```
Dim myfso As New FileSystemObject
```

۲- در این روش از تابع CreateObject می‌توان استفاده نمود. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

Set نام متغیر (شئ) = CreateObject ("Scripting.FileSystemObject")

به‌عنوان مثال به این دستور توجه کنید:

```
Set  myfso = CreateObject ("Scripting.FileSystemObject")
```

۲-۱ متدهای شئ FSO

مدل شئ FSO متدهای متنوعی را برای انجام عملیات مختلف روی فایل‌ها، پوشه‌ها و درایوها ارائه می‌کند در اینجا به توضیح هر یک از آن‌ها می‌پردازیم.

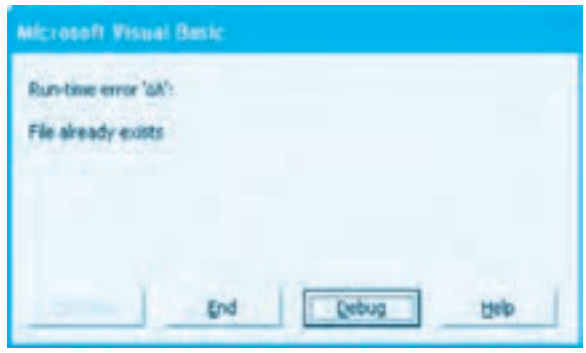
۲-۱-۱ متد CopyFile

به‌وسیله این متد می‌توان یک یا چند فایل را از یک محل به محل دیگر کپی کرد. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

نام شئ از نوع FSO . CopyFile source, destination [, overwrite]

این متد دارای دو آرگومان اجباری و یک آرگومان اختیاری است. آرگومان source از نوع رشته‌ای می‌باشد و مسیر و نام فایل یا فایل‌هایی را که کپی می‌شوند، معین می‌کند. آرگومان destination نیز از نوع رشته‌ای

است و مسیری را که فایل‌ها در آن کپی می‌شوند، تعیین می‌کند. آرگومان overwrite اختیاری بوده و از نوع منطقی می‌باشد. اگر مقدار آن True (مقدار پیش‌فرض) باشد و فایل یا فایل‌هایی که در مقصد کپی می‌شوند در آنجا وجود داشته باشند رونویسی خواهند شد اما اگر مقدار آن False باشد در صورت اجرای متد، پیام خطایی مطابق شکل ۳-۱ نمایش داده می‌شود.



شکل ۳-۱

به‌عنوان مثال به دستورات زیر توجه کنید:

```
Dim myfso As New FileSystemObject
myfso.CopyFile "C:\ Readme.txt", " D:\ Test\"
```

در این دستورات ابتدا شئ myfso از مدل شئ FSO ایجاد می‌شود سپس با استفاده از متد CopyFile فایل Readme.txt از ریشه درایو C: در پوشه Test در ریشه درایو D: کپی می‌شود. در صورتی‌که فایل Readme.txt در مسیر D:\ Test وجود داشته باشد این دستورات فایل Readme.txt را از ریشه درایو C: برروی این فایل در مسیر D:\ Test رونویسی می‌کند.

```
Private Sub txtdrive_KeyDown(KeyCode As Integer, Shift As Integer)
    Dim myfso As New FileSystemObject
  Dim myfolder As Folder
  If (KeyCode = 13) Then
    Set myfolder = myfso.GetFolder(txtdrive.Text)
    lblts.Caption = Str(myfolder.Size) + "  Byte"
    lblfs.Caption = myfolder.DateCreated
```

```
    lblas.Caption = myfolder.DateLastAccessed
    lblfsm.Caption = myfolder.ParentFolder
  End If
End Sub
```

در آرگومان source این متد می‌توانید از کاراکترهای جانشین ? و * برای تعیین گروهی از فایل‌ها استفاده کنید. به‌عنوان مثال این دستورات، تمام فایل‌های متنی موجود در پوشه C:\Tools را به فهرست ریشه درایو D: کپی می‌کند.

```
Dim myfso As New FileSystemObject
myfso.CopyFile  "C:\Tools\ *.txt", "D:\"
```

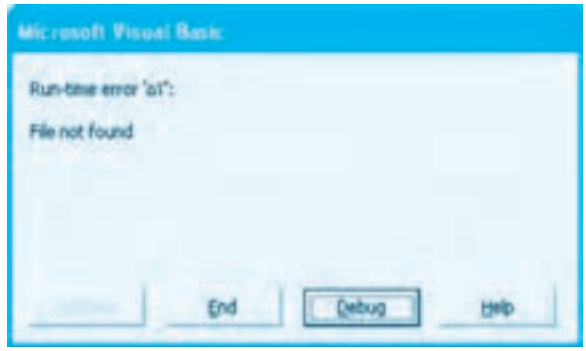
اگر در آرگومان destination علاوه بر مسیر مقصد، نام جدیدی برای فایل تعیین شود فایل مبدأ با این نام در مسیر مقصد کپی خواهد شد.

به عنوان مثال

```
Dim myfso As New FileSystemObject
myfso.CopyFile  "C:\Tools\ readme.txt", "D:\game\ letter.txt"
```

در صورت اجرای این دستورات فایل readme.txt با نام letter.txt در پوشه game کپی خواهد شد.

نکته: در صورتی‌که فایل یا فایل‌های تعیین شده در آرگومان source وجود نداشته باشند پیام خطایی مانند شکل ۴-۱ نمایش داده می‌شود.



شکل ۴-۱

نکته: در صورتی‌که پوشه مبدأ یا مقصد موجود نباشد پیام خطایی مانند شکل ۵-۱ نمایش داده می‌شود.

شکل ۵-۱

تمرین: دستوراتی بنویسید که فایل‌های با پسوند BMP را از پوشه سیستم عامل ویندوز به فهرست ریشه درایو C: کپی کند.

تمرین: رویه فرعی با نام CopyFiles بنویسید که بتواند نام یک یا گروهی از فایل‌ها همراه با مسیر مبدأ و مقصد را دریافت کند و فایل‌های مورد نظر را از مسیر مبدأ به مسیر مقصد کپی کند را از یک مسیر در مسیر دیگری کپی کند.

۱-۳-۲ متد CopyFolder

به‌وسیله این متد می‌توان یک پوشه را از یک محل به محل دیگر کپی کرد. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

نام شئ از نوع FSO . CopyFolder Source, destination [, overwrite]

این متد دارای دو آرگومان اجباری و یک آرگومان اختیاری است. آرگومان Source از نوع رشته‌ای می‌باشد و مسیر و نام پوشه‌ای را که کپی می‌شود، معین می‌کند. آرگومان destination نیز از نوع رشته‌ای است و مسیری را که فایل‌ها در آن کپی می‌شوند تعیین می‌کند.

آرگومان overwrite اختیاری بوده و از نوع منطقی می‌باشد. اگر مقدار آن True باشد (مقدار پیش‌فرض) و پوشه‌ای که در مقصد کپی می‌شود در آنجا وجود داشته باشد رونویسی انجام خواهد شد اما اگر مقدار آن False باشد، در صورت اجرای متد پیام خطایی مطابق شکل ۳-۱ نمایش داده

می‌شود.

به عنوان مثال دستورات زیر، پوشه C:\ Tools را همراه محتویات آن در پوشه D:\ Test کپی می‌کند.

```
Dim myfso As New FileSystemObject
myfso. CopyFolder  "C:\ Tools", "D:\ Test\"
```

به عنوان مثال دستورات بعدی تمام پوشه‌های موجود در پوشه C:\Tools (بجز خود پوشه Tools) را به پوشه D:\Test کپی می‌کند.

```
Dim myfso As New FileSystemObject
myfso. CopyFolder "C:\ Tools\*" , "D:\ Test\"
```

اگر پس از اجرای این دستور نتیجه را در برنامه My Computer مشاهده کنید خواهید دید که تمام محتویات پوشه Tools در پوشه Test کپی می‌شود اما پوشه‌ای با نام Tools در مقصد به‌وجود نمی‌آید.

نکته: در صورتی‌که آرگومان destination به کاراکتر \ ختم نشود، محتویات پوشه ذکر شده در آرگومان source در مقصد کپی می‌شوند. به عنوان مثال دستور زیر به جای پوشه Tools محتویات آن را در پوشه Test کپی می‌کند.

```
myfso. CopyFolder  "C:\ Tools" , "D:\ Test"
```

نکته: در صورتی‌که پوشه مبدأ موجود نباشد پیام خطایی مانند شکل ۵-۱ نمایش داده می‌شود.

نکته: در صورتی‌که پوشه مقصد موجود نباشد در زمان اجرای متد به‌طور خودکار ایجاد می‌شود.

تمرین: دستوراتی بنویسید که پوشه C:\ game را در مسیر D:\ my Folder کپی کند.

تمرین: رویه فرعی با نام CopyFolders بنویسید که بتواند یک پوشه را از یک مسیر در مسیر دیگر کپی کند.

۳-۳-۱ متد CreatFolder

به‌وسیله این متد می‌توان یک پوشه جدید ایجاد نمود. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

CreateFolder Foldername. نام متغیر از نوع FSO

این متد یک آرگومان اجباری دارد. آرگومان Foldername از نوع رشته‌ای می‌باشد و نام و مسیر پوشه‌ای را که ایجاد می‌شود، مشخص می‌کند.

به عنوان مثال دستورات زیر، پوشه جدیدی به نام game را در پوشه C:\Tools ایجاد می‌کند.

```
Dim myfso As New FileSystemObject

Myfso. CreateFolder "C:\Tools\game
```

نکته: در صورتی‌که پوشه جدید در مسیر تعیین شده وجود داشته باشد پیام خطایی مطابق شکل ۳-۱ نمایش داده می‌شود.

نکته: در صورتی‌که مسیر ایجاد پوشه جدید اشتباه باشد پیام خطایی مطابق شکل ۵-۱ نمایش داده می‌شود.

تمرین: یک رویه تابعی بنویسید که نام مسیر هر پوشه دلخواهی را دریافت نماید و آن‌را ایجاد کند.

۴-۳-۱ متد DeleteFile

به‌وسیله این متد می‌توان یک فایل یا گروهی از فایل‌ها را حذف کرد. این متد به‌صورت زیر است:

DeleteFile filename [, force]. نام شئ از نوع FSO

این متد دارای یک آرگومان اجباری و یک آرگومان اختیاری است. آرگومان filename از نوع رشته‌ای می‌باشد و نام و مسیر فایل (های) موردنظر را برای حذف مشخص می‌کند. آرگومان force اختیاری است و از نوع منطقی می‌باشد و در صورتی‌که مقدار آن True باشد فایل‌ها با صفت فقط خواندنی نیز حذف می‌شوند و اگر مقدار آن برابر False باشد فایل‌ها با صفت فقط خواندنی حذف نخواهند شد و پیام خطایی مطابق شکل ۶-۱ نمایش داده می‌شود. مقدار این آرگومان به‌طور پیش‌فرض False است.

به عنوان مثال دستورات زیر فایل متنی letter.txt را حذف می‌کنند.

```
Dim myfso As New FileSystemObject

myfso. DeleteFile "D:\letter.txt"
```

به عنوان مثال این دستورات تمام فایل‌های متنی در ریشه درایو D: را حذف می‌کند.

```
Dim myfso As New FileSystemObject
myfso. DeleteFile "D:\*.txt"
```

به عنوان مثالی دیگر این دستورات کلیه فایل‌های متنی با صفت فقط خواندنی را از ریشه درایو :D حذف می‌کند.

```
Dim myfso As New FileSystemObject
myfso. DeleteFile "D:\*.txt"
```

شکل ۶-۱

نکته: در صورتی‌که مسیر فایل‌هایی که حذف می‌شوند یا فایل(های) تعیین شده برای حذف وجود نداشته باشد، پیام خطایی مطابق شکل ۴-۱ نمایش داده می‌شود.

تمرین: یک رویه فرعی بنویسید که بتواند یک یا گروهی از فایل‌ها را حذف کند و در هنگام حذف فایل‌ها به‌صورت گروهی پیامی مبنی بر تأیید حذف آن‌ها نمایش دهد.

۵-۳-۱ متد DeleteFolder

به وسیله این متد می‌توان یک پوشه را همراه با محتویات آن حذف کرد. شکل کلی این متد به‌صورت زیر است:

نام شئ از نوع FSO.DeleteFolder Foldername [, force]

این متد دارای یک آرگومان اجباری و یک آرگومان اختیاری است. آرگومان foldername از نوع رشته‌ای می‌باشد و نام و مسیر پوشه‌ای را که حذف می‌شود، تعیین می‌کند.

به عنوان مثال این دستورات، پوشه Program را از ریشه درایو :D حذف می‌کند.

```
Set  myfso= Createobject ("Scripting. FileSystemObject")

myfso. DeleteFolder  "D:\Program"
```

تمرین: رویه‌ای بنویسید که نام و مسیر یک پوشه فقط خواندنی را دریافت کرده و آن را حذف کند.

در این دستورات ابتدا با استفاده از تابع CreateObject یک شئ (myfso) ایجاد می‌شود، سپس متد DeleteFolder پوشه Program را حذف می‌کند.

به عنوان مثال دستور زیر تمام پوشه‌های موجود در پوشه Tools را حذف می‌کند.

```
Dim myfso As New FileSystemObject

myfso. DeleteFolder "C:\Tools\*"
```

نکته: در صورتی‌که مسیر پوشه‌ای که حذف می‌شود اشتباه باشد یا پوشه موردنظر برای حذف وجود نداشته باشد پیام خطایی مطابق شکل ۵-۱ نمایش داده می‌شود.

تمرین: یک رویه فرعی بنویسید که بتواند نام و مسیر هر پوشه دلخواهی را دریافت کند و آن‌را حذف کند.

۶-۳-۱ متد DriveExists

این متد وجود یا عدم وجود یک درایو را بررسی می‌کند، در صورتی‌که درایو تعیین شده موجود باشد مقدار True و در غیر این صورت مقدار False را بازگشت می‌دهد. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

نام شئ از نوع FSO.DriveExists (drivename)

آرگومان drivename در این متد از نوع رشته‌ای است که نام درایو را تعیین می‌کند. به عنوان مثال به دستورات زیر توجه کنید:

```
Dim myfso As New FileSystemObject

If  myfso. DriveExists ("D:")  Then

   Print  "The Drive Already Exists."

Else

   Print  "The Drive Doesn't Exist."

End If
```

تمرین: رویه‌ای بنویسید که نام و مسیر یک پوشه فقط خواندنی را دریافت کرده و آن را حذف کند.

نکته: در توابعی که مقدار منطقی بازگشت می‌دهند می‌توان مستقیماً از مقدار بازگشتی در دستور شرطی IF استفاده کرد و نیازی به مقایسه مقدار بازگشتی با مقدار True یا False نمی‌باشد. به عنوان مثال در دستورات قبل می‌توان دستور If را به این صورت نیز نوشت: If my fso Drive Exists (D:ً) = True Then

نکته: در صورتی‌که از این متد برای درایوهای دیسکت، CD ROM یا DVD ROM استفاده شود، متد وجود دیسک در درایو را تشخیص نمی‌دهد.

در این دستورات ابتدا یک شئ از نوع FSO ایجاد می‌شود سپس با استفاده از یک دستور If و متد DriveExists وجود درایو D: بررسی می‌شود. در صورتی‌که این درایو روی سیستم وجود داشته باشد پیام موجود بودن درایو و در غیر این صورت پیام عدم وجود درایو نمایش داده می‌شود.

تمرین: تابعی بنویسید که نام یک درایو را دریافت نموده و وجود یا عدم وجود آن‌را مشخص کند.

۱-۳-۷ متد FileExists

این متد وجود یا عدم وجود یک فایل را در مسیر تعیین شده بررسی می‌کند، در صورتی‌که فایل تعیین شده موجود باشد مقدار True و در غیر این‌صورت مقدار False را بازگشت می‌دهد. شکل کلی این متد به صورت زیر است:

نام شئ از نوع FSO.FileExists (filename)

تمرین: رویه فرعی CopyFiles را به‌گونه‌ای تغییر دهید که ابتدا با استفاده از متد FileExists وجود فایل بررسی شود و در صورت عدم وجود فایل پیام "File Not Found" نمایش داده شده و در غیر این‌صورت عملیات کپی انجام شود.

۱-۳-۸ متد FolderExists

این متد وجود یا عدم وجود یک پوشه را در مسیر تعیین شده بررسی می‌کند، در صورتی‌که پوشه تعیین شده موجود باشد مقدار True و در غیر این‌صورت مقدار False را بازگشت می‌دهد. شکل کلی نحوه استفاده از این متد به صورت زیر است:

نام شئ از نوع FSO.FolderExists (foldername)

تمرین: رویه فرعی CopyFolders را به‌گونه‌ای تغییر دهید که ابتدا با استفاده از متد FolderExists وجود پوشه بررسی شود و در صورت عدم وجود پوشه پیام "Path Not Found" نمایش داده شده و در غیر این‌صورت عملیات کپی انجام شود.

۹-۳-۱ متد GetDrive

به وسیله این متد می‌توانید اطلاعات مورد نیاز خود را درباره یک درایو یا پارتیشن‌های دیسک‌سخت، دیسکت یا انواع دیسک‌های فشرده CD ROM و DVD ROM به‌دست آورید. شکل کلی این متد به‌صورت زیر می‌باشد:

GetDrive (drivename).نام شئ از نوع FSO

آرگومان drivename از نوع رشته‌ای است و می‌تواند شامل نام درایو به تنهایی (مانند "C" برای درایو C:)، نام درایو همراه با کاراکتر : (مانند "C:")یا نام درایو همراه با کاراکتر \ (مانند "C:\") باشد. به منظور ذخیره‌سازی اطلاعات مربوط به درایو که به‌وسیله این متد به‌دست می‌آید می‌توان از دستور Set استفاده نمود و اطلاعات به‌دست آمده را در یک متغیر از نوع Variant یا از نوع شئ Drive ذخیره کرد سپس با استفاده از خاصیت‌های ارائه شده در جدول ۱-۱ و ۱-۲ به این اطلاعات دسترسی پیدا نمود.

نکته: در صورتی‌که درایو تعیین شده در سیستم وجود نداشته باشد پیام خطایی مشابه شکل ۱-۷ نمایش داده می‌شود.

شکل ۱-۷

جدول ۱-۱ خواص مربوط به شئ Drive

نام خاصیت	توضیح
AvailableSpace	فضای قابل دسترسی درایو را براساس بایت مشخص می‌کند.
DriveLetter	نام درایو فیزیکی یا درایو شبکه را مشخص می‌کند.
DriveType	نوع درایو را با توجه به جدول ۲-۱ مشخص می‌کند.
FileSystem	نوع سیستم آدرس‌دهی را در درایو مشخص می‌کند. (FAT16، FAT32، NTFS و ...)
FreeSpace	فضای خالی درایو را براساس بایت مشخص می‌کند.
IsReady	درصورتی‌که مقدار این خاصیت True باشد درایو آماده است و در غیر این صورت درایو آماده نمی‌باشد.
SerialNumber	شماره سریال دیسک را مشخص می‌کند.
TotalSize	فضای کل درایو را براساس بایت مشخص می‌کند.
VolumeName	برچسب (Label) درایو را مشخص می‌کند.

مثال ۱: یک پروژه از نوع Standard EXE به همراه یک فرم مطابق شکل ۸-۱ طراحی کنید که با تعیین نام یک درایو بتوان مشخصات آن‌را مشاهده نمود. برای این‌کار عملیات بعدی را به ترتیب انجام دهید.

شکل ۸-۱

جدول ۲-۱ مقادیر مربوط به خاصیت DriveType

مقدار	توضیح
۰	نامعین
۱	قابل جابه‌جایی (Removable)
۲	ثابت (Fixed)
۳	درایو شبکه
۴	درایو CD ROM و DVD ROM
۵	دیسک RAM (RAM Disk)

۱- برنامه ویژوال‌بیسیک را اجرا کنید و یک پروژه از نوع Standard EXE به همراه یک فرم با عرض ۵۵۰۰ و ارتفاع ۴۵۰۰ ایجاد کنید.

۲- رویداد KeyDown کنترل کادرمتن txtdrive را به این صورت تنظیم کنید:

```
Private Sub txtdrive_KeyDown (KeyCode As Integer, Shift As Integer)
  Dim myfso As New FileSystemObject
  Dim mydrv As Drive
  If (KeyCode = 13) Then
    Set mydrv = myfso.GetDrive (txtdrive.Text)
    lblts.Caption = Str (mydrv.TotalSize) + "  Byte"
    lblfs.Caption = Str (mydrv.FreeSpace) + "  Byte"
    lblvn.Caption = mydrv.VolumeName
    lblfsm.Caption = mydrv.FileSystem
  End If
```

```
End Sub
```

در این رویداد ابتدا یک شئ (myfso) از نوع FSO ایجاد می‌شود، سپس یک متغیر (mydrv) از نوع Drive برای ذخیره کردن اطلاعات مربوط به درایو تعریف می‌شود. در مرحله بعد با استفاده از یک دستور If کد کلید فشرده شده در کادرمتن txtdrive بررسی می‌گردد، اگر کلید Enter فشرده شده باشد متد GetDrive شئ myfso نام درایو را از کادرمتن دریافت نموده و اطلاعات درایو را در متغیر mydrv ذخیره می‌نماید سپس فضای درایو، فضای خالی، برچسب و نوع سیستم فایلی درایو تعیین شده به ترتیب در کنترل‌های برچسب نمایش داده می‌شوند.

۳- پروژه و فرم را با نام DriveInformation ذخیره کنید، سپس پروژه را اجرا نموده و نام یک درایو را در کادرمتن txtdrive تایپ کنید و کلید Enter را فشار دهید و نتیجه اجرای دستورات را در برنامه مشاهده نمایید.

۴- اجرای پروژه را متوقف کرده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه مثال ۱ را به گونه‌ای تغییر دهید تا ابتدا وجود درایو تعیین شده بررسی شود و در صورت عدم وجود آن پیام خطای مناسب نمایش داده شود.

۱۰-۳-۱ کنترل کادرلیست درایو (DriveListBox)

به وسیله این کنترل می‌توانید لیستی از درایوهای موجود در سیستم را به منظور انتخاب کاربر نمایش دهید. این کنترل تمام پارتیشن‌های دیسک‌سخت، دیسکت، CD ROM، DVD ROM و درایوهای شبکه را به طور خودکار شناسایی می‌کند و در اختیار کاربر قرار می‌دهد (شکل ۹-۱).

شکل ۹-۱

این کنترل علاوه بر ویژگی‌های مشترکی که با سایر کنترل‌ها دارد دارای خاصیت‌ها و رویدادهای ویژه‌ای نیز می‌باشد.

خاصیت Drive

این خاصیت از نوع رشته‌ای می‌باشد و نام درایوی را که کاربر از لیست انتخاب کرده است مشخص می‌کند، به علاوه به‌وسیله این خاصیت می‌توانید درایو موردنظر را به عنوان درایو انتخاب شده در کنترل تعیین کنید؛ در غیر این صورت درایو جاری (درایوی که پروژه از روی آن اجرا می‌شود) در کنترل انتخاب خواهد شد.

شکل کلی این خاصیت به صورت زیر است:

نام کنترل DriveListBox. Drive = drive

drive یک عبارت رشته‌ای است که نام درایو موردنظر را تعیین می‌کند. در صورتی که از این بخش (=drive) استفاده نشود، نام درایو انتخاب شده را در کنترل مشخص می‌کند. به عنوان مثال دستور زیر سبب می‌شود در هنگام نمایش، درایو D: به عنوان درایو انتخاب شده در نظر گرفته شود.

```
DriveListBox1. Drive= "D:"
```

دستور زیر نیز درایو انتخاب شده در کنترل را روی فرم نمایش می‌دهد.

```
Print  DriveListBox1. Drive
```

نکته: در صورتی‌که درایوی دارای برچسب باشد، برچسب آن نیز همراه با نام درایو در خاصیت Drive کنترل DriveListBox ارائه می‌شود.

خاصیت ListCount

این خاصیت تعداد درایوهای موجود در کادرلیست کنترل را مشخص می‌کند. به عنوان مثال دستور زیر تعداد درایوها را روی فرم نمایش می‌دهد.

```
Print  DriveListBox1. ListCount
```

خاصیت ListIndex

در کنترل DriveListBox، هر درایو دارای یک شماره منحصر به فرد است. شماره‌ها از صفر برای اولین درایو آغاز شده و به ترتیب تا آخرین درایو ادامه می‌یابند، به وسیله این خاصیت می‌توان شماره درایو انتخاب شده

در کنترل را به‌دست آورد. به‌عنوان مثال در کنترل شکل ۹-۱ اگر کاربر روی درایو :a کلیک کند مقدار این خاصیت برابر با صفر و اگر روی درایو :f کلیک کند مقدار آن برابر با ۴ خواهد بود. شکل کلی این خاصیت به این صورت می‌باشد:

DriveListBox نام کنترل.ListIndex = index

index یک مقدار عددی صحیح مثبت است که از صفر شروع می‌شود و درایو انتخاب شده در کنترل را تعیین می‌کند. در صورت عدم استفاده از index، شماره درایوی که در کنترل انتخاب شده است مشخص می‌شود. به عنوان مثال دستورات زیر ابتدا درایو :C را به عنوان درایو انتخاب شده تعیین می‌کند سپس درایو انتخاب شده روی فرم نمایش داده می‌شود.

```
DriveListBox1. ListIndex= 1
Print  DriveListBox1. ListIndex
```

رویداد Change

مهم‌ترین رویداد این کنترل رویداد Change است. این رویداد زمانی اجرا می‌شود که کاربر درایوی را از کادرلیست کنترل انتخاب کند.

نکته: تغییر درایو جاری در این کنترل، درایو جاری را از نظر سیستم عامل و سایر دستورات در برنامه تغییر نمی‌دهد.

نکته: در کنترل کادر لیست درایو به طور پیش‌فرض درایوی که برنامه در روی آن ذخیره و اجرا شده است انتخاب می‌شود.

مثال ۲: می‌خواهیم پروژه DriveInformation را به گونه‌ای تنظیم کنیم تا کاربر به جای تایپ نام درایو، لیست تمام درایوهای موجود روی سیستم را مشاهده نموده و یک درایو را انتخاب کند. به این منظور عملیات زیر را به ترتیب انجام دهید:

۱- پروژه DriveInformation را باز کنید و کنترل کادرمتن را از روی آن حذف نمایید.

۲- در جعبه ابزار روی آیکن کنترل DriveListBox دابل‌کلیک کنید و کنترل جدید را در محل کنترل کادرمتن قرار دهید (شکل ۱۰-۱). سپس خاصیت نام آن‌را روی drvdrive تنظیم کنید.

شکل ۱۰-۱

۳- دستورات موجود در رویداد txtdrive_KeyDown را به رویداد Change کنترل drvdrive منتقل نمایید، سپس دستورات را به این صورت تغییر دهید:

```
Private Sub drvdrive_Change()
    Dim myfso As New FileSystemObject
    Dim mydrv As Drive
    Dim strdrive As String
    strdrive = Left (drvdrive.Drive, 2)
    Set mydrv = myfso.GetDrive (strdrive)
    lblts.Caption = Str (mydrv.TotalSize) + "  Byte"
    lblfs.Caption = Str (mydrv.FreeSpace) + "  Byte"
    lblvn.Caption = mydrv.VolumeName
    lblfsm.Caption = mydrv.FileSystem
```

```
End Sub
```

در این رویداد یک متغیر جدید (strdrive) برای نگهداری نام درایو تعریف شده است، سپس با استفاده از تابع Left دو کاراکتر اول در رشته‌ای که به‌وسیله خاصیت Drive کنترل drvdrive مشخص می‌شود جدا می‌شود و در متغیر strdrive ذخیره می‌گردد، چون ممکن است درایو انتخاب شده دارای برچسب باشد که همراه نام آن در خاصیت Drive کنترل drvdrive قرار می‌گیرد. اما متد GetDrive فقط به نام درایو نیاز دارد، دستور If هم در این رویداد کاربردی ندارد بنابراین حذف شده است زیرا با انتخاب نام یک درایو در کنترل drvdrive رویداد Change آن اجرا شده و به‌طور خودکار اطلاعات درایو انتخاب شده نمایش داده می‌شود.

۴- تغییرات را ذخیره کرده و پروژه را اجرا کنید، سپس نام یک درایو را از کادرلیست کنترل انتخاب نمایید تا مشخصات آن روی فرم نمایش داده شود.

۵- اجرای برنامه را متوقف کرده و به پنجره ویژوال‌بیسیک بازگردید.

۱-۳-۱۱ متد GetDriveName

این متد می‌تواند یک مسیر دلخواه را دریافت نماید و نام درایو آن‌را به‌صورت یک رشته بازگشت دهد. شکل کلی این متد به صورت زیر است:

نام شئ از نوع FSO.GetDriveName (path)

آرگومان path از نوع رشته‌ای است و یک مسیر دلخواه را مشخص می‌کند.

به عنوان مثال دستورات زیر عبارت C: را روی فرم نمایش می‌دهد.

```
Dim myfso As New FileSystemObject

Print  myfso. GetDriveName ("C:\Windows")
```

نکته: در صورتی‌که در آرگومان path نام درایو ذکر نشود یک رشته با طول صفر بازگشت داده می‌شود.

نکته: در صورتی‌که مسیر تعیین شده وجود نداشته باشد در عملکرد این تابع هیچ تغییری ایجاد نمی‌شود و نام درایو بازگشت داده می‌شود.

۱-۳-۱۲ متد GetFolder

به‌وسیله این متد می‌توانید اطلاعات مورد نیاز خود را در رابطه با یک پوشه مانند تاریخ تشکیل، پوشه والد، حجم محتویات آن و غیره به دست آورید. شکل کلی نحوه استفاده از این متد به این صورت است:

نام شئ از نوع FSO.GetFolder (foldername)

آرگومان foldername از نوع رشته‌ای است و نام و مسیر پوشه موردنظر را تعیین می‌کند. به منظور ذخیره‌سازی اطلاعات مربوط به پوشه که به‌وسیله این متد به‌دست می‌آید می‌توان از دستور Set استفاده نمود و اطلاعات به دست آمده را در یک متغیر از نوع Variant یا از نوع شئ Folder ذخیره کرد، سپس با استفاده از خاصیت‌های ارائه شده در جدول ۳-۱ و ۴-۱ به این اطلاعات دسترسی پیدا نمود.

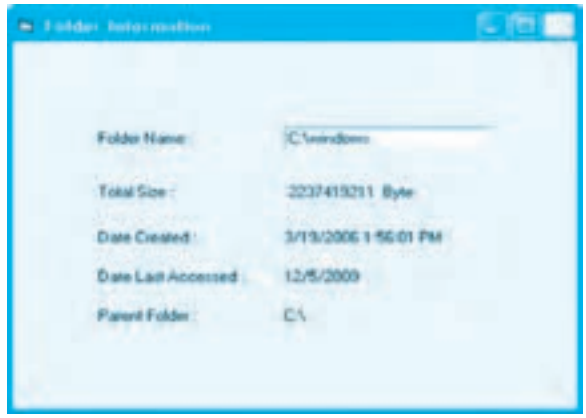
جدول ۳-۱ خواص مربوط به شئ Folder

نام خاصیت	توضیح
Attributes	صفت‌های پوشه را با توجه به جدول ۴-۱ تنظیم کرده یا مشخص می‌کند.
Date Created	تاریخ تشکیل پوشه را مشخص می‌کند.
DateLastAccessed	تاریخ آخرین دسترسی به پوشه را مشخص می‌کند.
Drive	نام درایوی را که پوشه در آن قرار گرفته است مشخص می‌کند.
IsRootFolder	در صورتی‌که مقدار آن برابر با True باشد پوشه فهرست ریشه است.
Name	نام پوشه را مشخص می‌کند.
ParentFolder	نام پوشه والد را مشخص می‌کند.
Path	مسیر پوشه را مشخص می‌کند.
Size	حجم محتویات پوشه را مشخص می‌کند.
DateLastModified	تاریخ آخرین ویرایش پوشه را مشخص می‌کند.

جدول ۴-۱ مقادیر مربوط به خاصیت Attribute

توضیح	ثابت عددی	ثابت رشته‌ای
بدون صفت	۰	Normal
صفت فقط‌خواندنی	۱	ReadOnly
صفت مخفی	۲	Hidden
صفت سیستمی	۴	System
برچسب دیسک درایو	۸	Volume
پوشه	۱۶	Directory
صفت بایگانی	۳۲	Archive
میانبر (Shortcut)	۶۴	Alias
فایل فشرده	۱۲۸	Compressed

مثال ۳: یک پروژه از نوع Standard EXE به همراه یک فرم مانند شکل ۱۱-۱ طراحی کنید که با تعیین نام و مسیر یک پوشه دلخواه بتوان مشخصات آن‌را مشاهده نمود. برای این کار عملیات بعد را به ترتیب انجام دهید.



شکل ۱۱-۱

۱- برنامه ویژوال‌بیسیک را اجرا کنید و یک پروژه از نوع Standard EXE به همراه یک فرم با عرض ۶۰۰۰ و ارتفاع ۵۰۰۰ ایجاد کنید.

۲- رویداد KeyDown کنترل کادرمتن txtfolder را به این صورت تنظیم کنید:

```
Private Sub txtfolder_KeyDown (KeyCode As Integer, Shift As Integer)
  Dim myfso As New FileSystemObject
  Dim myfolder As Folder
  If (KeyCode = 13) Then
    Set myfolder = myfso.GetFolder (txtfolder.Text)
    lblts.Caption = Str (myfolder.Size) + "  Byte"
    lbldc.Caption = myfolder.DateCreated
    lblla.Caption = myfolder.DateLastAccessed
    lblpf.Caption = myfolder.ParentFolder
  End If
End Sub
```

در این رویداد ابتدا یک شئ (myfso) از نوع FSO ایجاد می‌شود سپس یک متغیر myfolder از نوع Folder برای ذخیره‌سازی اطلاعات مربوط به پوشه موردنظر تعریف می‌شود؛ در مرحله بعد با استفاده از یک دستور شرطی کد کلید فشرده شده در کادرمتن txtfolder بررسی می‌گردد، اگر کلید Enter فشرده شده باشد متد GetFolder شئ myfso نام و مسیر پوشه را از کادرمتن دریافت نموده و اطلاعات پوشه را در متغیر myfolder ذخیره می‌نماید سپس حجم محتویات، تاریخ تشکیل، تاریخ آخرین دسترسی و پوشه والد آن نمایش داده می‌شود.

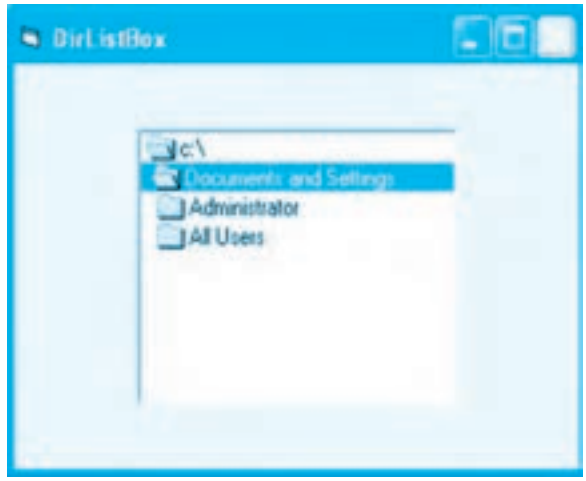
۳- پروژه و فرم را با نام FolderInformation ذخیره کنید، سپس پروژه را اجرا نموده و نام و مسیر یک پوشه دلخواه را در کادرمتن txtfolder تایپ کنید و کلید Enter را فشار دهید، سپس نتیجه اجرای دستورات را روی فرم مشاهده نمایید.

۴- اجرای پروژه را متوقف کرده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه مثال ۳ را به‌گونه‌ای تغییر دهید تا ابتدا وجود پوشه روی سیستم بررسی شود و در صورت عدم وجود آن پیام خطای مناسبی نمایش داده شود.

۱۳-۳-۱ کنترل کادر لیست پوشه‌ها (DirListBox)

به وسیله این کنترل می‌توانید لیستی از پوشه‌های موجود در یک مسیر روی پارتیشن‌های دیسک سخت، دیسکت، CDROM، DVDROM و مانند آن‌ها را مشاهده کنید و در صورت نیاز وارد پوشه موردنظر شوید، این کنترل به طور خودکار زیر پوشه‌های پوشه انتخاب شده را نمایش می‌دهد. برای باز کردن هر پوشه و ورود به آن باید روی آیکن پوشه در کنترل دابل‌کلیک کنید، در شکل ۱۲-۱ نمونه‌ای از این کنترل را مشاهده می‌کنید.



شکل ۱۲-۱

این کنترل علاوه بر خاصیت‌های مشترکی که با سایر کنترل‌ها دارد، دارای خاصیت‌های مخصوص به خود نیز می‌باشد که به این شرح هستند:

خاصیت ListCount

این خاصیت از نوع عددی می‌باشد و تعداد زیرپوشه‌هایی را که در پوشه باز شده در کنترل وجود دارند مشخص می‌کند. به عنوان مثال دستور زیر تعداد زیرپوشه‌های موجود در فهرست ریشه درایو C: را (مقدار ۵) در کنترل DirListBox شکل ۱۲-۱ نمایش می‌دهد.

```
Print  DirListBox1. ListCount
```

خاصیت ListIndex

در کنترل DirListBox، هر پوشه دارای یک شماره منحصر به فرد می‌باشد که مقدار آن با توجه به ترتیب قرارگرفتن پوشه‌ها نسبت به پوشه‌ای که باز شده است تنظیم می‌شود. به عنوان مثال مقدار خاصیت

ListIndex برای پوشه Administrator در شکل ۱۳-۱، ۱- می‌باشد و برای پوشه Documents And Settings، ۲- و برای فهرست ریشه یعنی C:\، ۳- است. به علاوه مقدار این خاصیت برای پوشه‌هایی که در پوشه Administrator قرار گرفته‌اند به ترتیب برای پوشه Cookies، صفر برای پوشه Desktop، ۱ و به همین ترتیب تا آخرین پوشه تنظیم می‌شود.

شکل ۱۳-۱

توجه داشته باشید که این خاصیت در هر لحظه شماره پوشه‌ای را که در کنترل انتخاب شده است مشخص می‌کند و شماره‌ها همواره با توجه به پوشه‌ای که باز می‌شود (روی آن دابل‌کلیک می‌شود) برای پوشه‌های والد و زیرپوشه‌ها مجدداً تنظیم می‌گردد.

نکته: مقدار خاصیت ListIndex همواره برای پوشه‌ای که باز شده است ۱- می‌باشد.

شکل کلی این خاصیت به این صورت است:

نام کنترل.ListIndex = index

index یک مقدار از نوع عددی صحیح است و پوشه جاری (پوشه‌ای که باز شده است) را در کنترل تعیین می‌کند. در صورت عدم استفاده از index، شماره پوشه انتخاب شده در کنترل مشخص می‌شود. به عنوان مثال این دستورات با توجه به شکل ۱۳-۱ ابتدا پوشه Desktop را در کنترل انتخاب نموده سپس شماره آن را (مقدار ۱) روی فرم نمایش می‌دهند.

```
DirListBox1. ListIndex = 1

Print  DirListBox1. ListIndex
```

خاصیت Path

این خاصیت از نوع رشته‌ای می‌باشد و مسیر پوشه‌ای را که در حال حاضر در کنترل DirListBox باز شده است مشخص می‌کند، به علاوه به وسیله این خاصیت می‌توانید پوشه جاری (پوشه باز) را نیز در کنترل انتخاب کنید.

شکل کلی این خاصیت به صورت زیر است:

DirListBox نام کنترل.Path = pathname

Pathname از نوع رشته‌ای است که پوشه جاری را در کنترل تعیین می‌کند، در صورتی که از این بخش (=pathname) استفاده نشود خاصیت Path مسیر پوشه جاری در کنترل را مشخص می‌کند.

به عنوان مثال این دستورات ابتدا پوشه C:\Windows را به عنوان پوشه جاری در کنترل تعیین نموده و سپس نام و مسیر آن را روی فرم نمایش می‌دهند.

```
DirListBox1. Path ="C:\Windows"

Print  DirListBox1. Path
```

رویداد Change

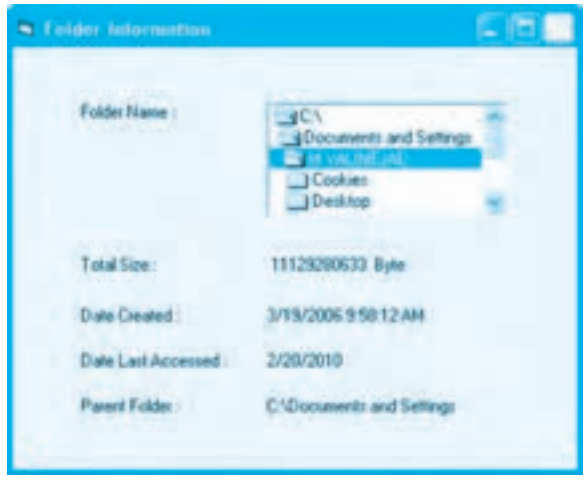
این رویداد زمانی اجرا می‌شود که کاربر در کنترل روی یک پوشه دابل‌کلیک کند.

نکته: با تغییر پوشه جاری در کنترل، پوشه جاری از نظر سیستم‌عامل و سایر دستورات در برنامه تغییر نمی‌کند.

مثال ۴: می‌خواهیم پروژه FolderInformation را به گونه‌ای تنظیم کنیم تا کاربر به جای تایپ نام و مسیر پوشه‌ها، لیست تمام پوشه‌ها را در یک مسیر مشاهده نموده و پوشه موردنظر خود را انتخاب کند. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- پروژه FolderInformation را باز کنید و کنترل کادرمتن را از روی آن حذف نمایید.

۲- در جعبه ابزار روی آیکن کنترل DirListBox دابل‌کلیک کنید و کنترل جدید را در محل کنترل کادرمتن قرار دهید (شکل ۱۴-۱)، سپس خاصیت نام آن را روی dirpath تنظیم کنید.



شکل ۱۴-۱

۳- دستورات موجود در رویداد txtfolder_KeyDown را به رویداد Change کنترل dirpath منتقل نمایید سپس دستورات را به این صورت تغییر دهید:

```
Private Sub dirpath_Change()
    Dim myfso As New FileSystemObject
    Dim myfolder As Folder
    Set myfolder = myfso.GetFolder (dirpath.Path)
    lblts.Caption = Str (myfolder.Size) + "  Byte"
    lbldc.Caption = myfolder.DateCreated
    lblla.Caption = myfolder.DateLastAccessed
    lblpf.Caption = myfolder.ParentFolder
End Sub
```

۴- تغییرات را ذخیره کرده و پروژه را اجرا کنید، سپس یک پوشه را از کادرلیست کنترل dirpath انتخاب نمایید و آن را باز کنید تا مشخصات آن روی فرم نمایش داده شود.

۵- اجرای برنامه را متوقف کنید و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه FolderInformation را به گونه‌ای تنظیم کنید تا کاربر بتواند نام درایو را با استفاده

از یک کنترل DriveListBox انتخاب کند، سپس در یک کنترل DirListBox پوشه موردنظر خود را از درایو انتخاب شده تعیین نمایید.

۱-۳-۱۴ متد GetParentFolderName

این متد می‌تواند با دریافت نام و مسیر یک پوشه، مسیر پوشه والد آن‌را بازگشت دهد. نوع داده بازگشتی در این متد از نوع رشته‌ای می‌باشد. شکل کلی این متد به‌صورت زیر است:

نام شئ از نوع FSO.GetParentFolderName (path)

آرگومان Path یک عبارت رشته‌ای است که نام و مسیر پوشه موردنظر را تعیین می‌کند. به عنوان مثال این دستورات عبارت رشته‌ای D:\Program را نمایش می‌دهند.

```
Dim myfso As New FileSystemObject
Print  myfso. GetParentFolderName ("D:\Program\Vbasic6")
```

۱-۳-۱۵ متد GetAbsolutePathName

این متد نام یک پوشه را دریافت می‌کند و مسیر کامل آن‌را به‌صورت یک عبارت رشته‌ای بازگشت می‌دهد. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

نام شئ از نوع FSO.GetAbsolutePathName (foldername)

آرگومان foldername از نوع رشته‌ای است که مسیر یا نام یک پوشه یا نام یک درایو را مشخص می‌کند. به عنوان مثال فرض کنید که مسیر جاری در هنگام اجرای یک برنامه، D:\Program\Vbasic باشد در این صورت عملکرد این متد در حالت‌های مختلف مطابق جدول ۵-۱ خواهد بود.

جدول ۵-۱

مقدار بازگشتی	مقدار آرگومان foldername
«D:\Program\Vbasic»	«D:»
«D:\Program»	«D:..»
«D:\»	«D:\»
«D:\Program\Vbasic»	«Vbasic»
«D:\Program\Vbasic»	« «

۱-۳-۱۶ متد MoveFolder

با استفاده از این متد می‌توانید یک یا چند پوشه را به همراه محتویات آن‌ها از محلی به محل دیگر منتقل کنید. شکل کلی نحوه استفاده از این متد به این صورت است:

نام شئ از نوع FSO.MoveFolder source, destination

آرگومان source از نوع رشته‌ای است و نام و مسیر پوشه یا پوشه‌هایی را که منتقل می‌شوند معین می‌کند. آرگومان destination نیز از نوع رشته‌ای است و نام و مسیر پوشه مقصد را مشخص می‌کند به عنوان مثال این دستورات پوشه C:\game را با تمام محتویاتش به پوشه D:\test منتقل می‌کند.

```
Dim myfso As New FileSystemObject
myfso. MoveFolder  "C:\game" , "D:\test"
```

به عنوان مثال این دستورات تمام پوشه‌های موجود در پوشه C:\game را (بجز خود پوشه game) به پوشه D:\test منتقل‌می‌کند.

```
Dim myfso As New FileSystemObject
myfso. MoveFolder  "C:\game\*", "D:\test"
```

نکته: در صورتی‌که پوشه مبدأ موجود نباشد پیام خطایی مانند شکل ۵-۱ نمایش داده می‌شود.

نکته: در صورتی‌که پوشه مقصد موجود نباشد، در زمان اجرای متد به‌طور خودکار ایجاد می‌شود. به عنوان مثال این دستورات پوشه letter را با نام Document به پوشه D:\test منتقل می‌نمایند.

```
Dim myfso As New FileSystemObject
myfso. MoveFolder "C:\letter" , "D:\test\Document"
```

تمرین: پروژه‌ای طراحی کنید که با استفاده از کنترل‌های DirListBox و DriveListBox بتوان هر پوشه‌ای را از یک مسیر به مسیر دلخواه منتقل نمود.

۱-۳-۱۷ متد MoveFile

با استفاده از این متد می‌توان یک یا گروهی از فایل‌ها را از یک مسیر به مسیر دیگر منتقل نمود. شکل کلی نحوه استفاده از این متد به این صورت است:

FSO نام شئ از نوع MoveFile source, destination.

آرگومان source از نوع رشته‌ای است و نام و مسیر فایل یا فایل‌هایی را که منتقل می‌شوند معین می‌کند. آرگومان destination نیز از نوع رشته‌ای است و مسیری را که فایل یا فایل‌ها به آنجا منتقل می‌شوند، مشخص می‌کند. به عنوان مثال این دستورات فایل text.txt را از ریشه درایو C: به پوشه d:\Program منتقل می‌کنند.

```
Dim myfso As New FileSystemObject

myfso. MoveFile  "C:\text.txt", "D:\program"
```

به عنوان مثال این دستورات تمام فایل‌های گرافیکی از نوع BMP را از پوشه D:\graphic به پوشه C:\ Pictures منتقل می‌نمایند.

```
Dim myfso As New FileSystemObject

myfso. MoveFile "D:\graphic\*.*", "C:\pictures"
```

اگر در آرگومان destination علاوه بر مسیر مقصد نام جدیدی برای فایل تعیین شود، فایل مبدأ با این نام در مسیر مقصد کپی خواهد شد. به عنوان مثال این دستورات فایل letter.txt را با نام جدید .mydocument txt به پوشه game منتقل می‌نماید.

```
Dim myfso As New FileSystemObject

myfso. MoveFile  "C:\tools\letter.txt", "D:\game\mydocument.txt"
```

نکته: در صورتی که فایل یا فایل‌های تعیین شده در آرگومان source وجود نداشته باشند، پیام خطای File not found مانند شکل ۴-۱ نمایش داده می‌شود.

نکته: در صورتی که پوشه مبدأ یا مقصد موجود نباشد پیام خطای Path not found مانند شکل ۵-۱ نمایش داده می‌شود.

تمرین: یک رویه فرعی بنویسید که به وسیله آن بتوان هر فایل دلخواهی را با دریافت نام و مسیر آن به مسیر دیگری منتقل نمود.

۱۸-۳-۱ کنترل کادرلیست فایل (FileListBox)

به وسیله این کنترل می‌توانید لیستی از فایل‌های موجود در یک پوشه را مشاهده کنید و در صورت نیاز یک فایل را

انتخاب نمایید. در شکل ۱۵-۱ نمونه‌ای از این کنترل را مشاهده می‌کنید.

شکل ۱۵-۱

این کنترل علاوه بر خاصیت‌های مشترکی که با سایر کنترل‌ها دارد دارای خاصیت‌ها و رویدادهای مخصوص به خود می‌باشد:

خاصیت FileNeme

این خاصیت از نوع رشته‌ای می‌باشد و نام فایلی را که در کنترل FileListBox انتخاب شده است، مشخص می‌کند. شکل کلی نحوه استفاده از این خاصیت به‌صورت زیر است:

نام کنترل FileListBox.FileName = filename

filename از نوع رشته‌ای است و مسیر و نام فایل‌هایی را که در کنترل نمایش داده می‌شوند، تعیین می‌کند و در صورت عدم استفاده از آن نام فایل انتخاب شده در کنترل قابل دسترسی می‌باشد. به عنوان مثال در این دستورات ابتدا مسیر C:\ pictures برای نمایش فایل‌ها در کنترل تنظیم می‌شوند سپس نام فایل انتخاب شده (coffee Bean.bmp) روی فرم نمایش داده می‌شود (شکل ۱۵-۱).

```
FileListBox1.FileName = "C:\ pictures"

Print  FileListBox1.FileName
```

در عبارت رشته‌ای filename می‌توان از کاراکترهای ? و * استفاده نمود، به عنوان مثال این دستور سبب نمایش فایل‌های bmp در کنترل می‌شود (شکل ۱۶-۱).

```
FileListBox1.FileName = "C:\ pictures\*.bmp"
```

شکل ۱۶-۱

خاصیت ListCount

این خاصیت از نوع عددی بوده و تعداد فایل‌های موجود در پوشه را مشخص می‌کند. شکل کلی نحوه استفاده از این خاصیت به‌صورت زیر می‌باشد:

FileListBox نام کنترل.ListCount

به عنوان مثال این دستور با توجه به شکل ۱۵-۱ مقدار ۵ را نمایش می‌دهد.

```
Print FileListBox1.ListCount
```

خاصیت ListIndex

در کنترل FileListBox، هر فایل دارای یک شماره منحصر به فرد است که مقدار آن برای اولین فایل، صفر، برای دومین فایل ۱ و به همین شکل تا آخرین فایل موجود در کادرلیست کنترل ادامه می‌یابد. وقتی یک فایل در کنترل انتخاب می‌شود شماره آن به وسیله این خاصیت قابل دسترسی می‌باشد. شکل کلی این خاصیت به این صورت می‌باشد:

FileListBox نام کنترل.ListIndex

به عنوان مثال با توجه به شکل ۱۵-۱ اگر دومین فایل انتخاب شده باشد این دستور شماره ۱ را روی فرم نمایش می‌دهد.

```
Print FileListBox1.ListIndex
```

و این دستور مسیری را که خاصیت Path کنترل کادر لیست فایل روی آن تنظیم شده است روی فرم نمایش می‌دهد.

```
Print File list Box1. Path
```

خاصیت Path

این خاصیت از نوع رشته‌ای می‌باشد و مسیری را که اسامی فایل‌های آن در کنترل FileListBox نمایش داده می‌شوند، مشخص می‌کند. شکل کلی این خاصیت به این صورت می‌باشد:

FileListBox نام کنترل.Path = Pathname

Pathname از نوع رشته‌ای است و مسیر موردنظر را برای نمایش فایل‌ها در کنترل تعیین می‌کند، در صورت عدم استفاده از آن مسیری که کنترل فایل‌های آن را نمایش می‌دهد، مشخص می‌شود.

به عنوان مثال این دستور سبب می‌شود فایل‌های موجود در پوشه C:\ Windows در کنترل نمایش داده شوند.

```
FileListBox1.Path = "C:\ Windows"
```

و این دستور مسیری را که خاصیت Path کنترل کادر لیست فایل روی آن تنظیم شده است روی فرم نمایش می‌دهد.

```
Print File list Box1. Path
```

خاصیت Pattern

این خاصیت از نوع رشته‌ای می‌باشد و به وسیله آن می‌توان نوع فایل‌هایی را که در کنترل نمایش داده می‌شوند، تعیین نمود. شکل کلی نحوه استفاده از این خاصیت به‌صورت زیر است:

FileListBox نام کنترل.Pattern = strpattern

strpattern یک عبارت رشته‌ای است که نوع فایل‌ها را برای نمایش در کنترل تعیین می‌کند. به عنوان مثال این دستور سبب می‌شود فقط فایل‌های متنی در کنترل نمایش داده شوند.

```
FileListBox1.Pattern = "*.TXT"
```

خاصیت Hidden

این خاصیت از نوع منطقی می‌باشد و وقتی مقدار آن روی True تنظیم شود فایل‌ها با صفت مخفی در کنترل نمایش داده می‌شوند.

خاصیت ReadOnly

این خاصیت نیز از نوع منطقی می‌باشد و وقتی مقدار آن روی True تنظیم شود فایل‌ها با صفت فقط خواندنی در کنترل نمایش داده می‌شوند.

خاصیت List

این خاصیت یک آرایه یک بعدی رشته‌ای می‌باشد که هر عضو آن نام یکی از فایل‌های موجود در کنترل FileListBox را در خود نگهداری می‌کند. شکل کلی نحوه استفاده از این خاصیت به‌صورت زیر است:

FileListBox نام کنترل.List (index)

index یک مقدار عددی بزرگ‌تر یا مساوی صفر است و شماره فایلی را که در کنترل نمایش داده شده است، مشخص می‌کند. به عنوان مثال به این دستورات توجه کنید:

```
Print  FileListBox1.List (0)

Print  FileListBox1.List (4)

Print  FileListBox1.List (FlieListBox1.ListCount-1)
```

دستور اول نام اولین فایل و دستور دوم نام پنجمین فایل موجود در کنترل را روی فرم نمایش می‌دهد. در دستور سوم ابتدا تعداد فایل‌های موجود در کنترل با استفاده از خاصیت ListCount به‌دست آمده است، سپس یک واحد از آن کم می‌شود در نتیجه شماره آخرین فایل در کنترل به‌دست می‌آید زیرا شماره فایل‌ها از مقدار صفر شروع می‌شود، بنابراین دستور سوم، نام آخرین فایل را نمایش می‌دهد.

تمرین: رویه‌ای بنویسید که با استفاده از خاصیت List نام تمام فایل‌ها را در کنترل FileListBox روی فرم نمایش دهد.

خاصیت MultiSelect

به‌وسیله این خاصیت می‌توان روش‌های مختلفی را برای انتخاب گروهی از فایل‌ها در کنترل FileListBox تعیین نمود. اگر این خاصیت روی مقدار 0-None تنظیم شود، فقط امکان انتخاب یک فایل در کنترل وجود خواهد داشت. در صورت تنظیم این خاصیت روی مقدار 1-Simple می‌توانید چند فایل را با عمل کلیک یا فشردن کلید Space انتخاب کنید و در صورتی‌که این خاصیت روی مقدار 2-Extended تنظیم شود انتخاب چند فایل با عمل کلیک همراه با فشردن کلید Ctrl و عمل کلیک همراه با فشردن کلید Shift یا فشردن کلیدهای جهت‌دار همراه با کلید Shift امکان‌پذیر می‌شود.

خاصیت Selected

برای شناسایی فایل‌های انتخاب شده در حالت Simple و Extended می‌توانید از خاصیت Selected استفاده کنید. این خاصیت یک آرایه از نوع منطقی می‌باشد و با دریافت شماره اندیس هر فایل در کنترل FileListBox در صورتی که فایل انتخاب شده باشد مقدار آن True و در غیر این‌صورت مقدار آن False خواهد بود.

به عنوان مثال این دستور مشخص می‌کند که آیا دومین فایل در کنترل FileListBox انتخاب شده است یا خیر.

```
Print  FileListBox1.Selected (1)
```

اگر اجرای این دستور عبارت True را روی فرم نمایش دهد نشان‌دهنده آن است که دومین فایل انتخاب شده است اما اگر عبارت False نمایش داده شود به معنی عدم انتخاب فایل می‌باشد.

تمرین: رویه‌ای بنویسید که اسامی فایل‌های انتخاب شده در کنترل FileListBox را نمایش دهد.

مثال ۵: پروژه‌ای مطابق شکل ۱۷-۱ طراحی کنید که به‌وسیله آن بتوان یک فایل را به هر مسیر دلخواهی منتقل نمود. برای این کار عملیات زیر را به ترتیب انجام دهید.

شکل ۱۷-۱

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE به همراه یک فرم مانند شکل ۱۷-۱ ایجاد کنید.

۲- دو کنترل قاب (Frame) روی فرم قرار داده و روی هریک از آن‌ها یک کنترل کادرمتن برای دریافت نام درایو و مسیر فایل‌ها، یک کنترل FileListBox برای مشاهده اسامی فایل‌ها و دو کنترل برچسب قرار دهید و در پایان یک کنترل دکمه فرمان در قسمت پایین فرم بگذارید سپس عنوان کنترل‌های قاب و برچسب دکمه فرمان را مانند شکل ۱۷-۱ تنظیم کنید.

۳- رویداد کنترل‌های کادرمتن را به این صورت تنظیم کنید:

```
Private Sub txtdespath_KeyDown(KeyCode As Integer, Shift As Integer)

If (KeyCode = 13) Then flbdestination.Path = txtdespath.Text
```

```
End Sub

Private Sub txtsourcepath_KeyDown(KeyCode As Integer, Shift As Integer)

If (KeyCode = 13) Then flbsource.Path = txtsourcepath.Text

End Sub
```

در هر یک از این رویدادها یک دستور If برای شناسایی کلید فشرده شده استفاده می‌شود تا در صورت فشرده شدن کلید Enter، مسیر تایپ شده در هریک از کادرهای متن در خاصیت Path کنترل FileListBox مربوط به آن قرار گیرد تا کنترل‌های FileListBox بتوانند فایل‌های مسیرهای تعیین شده را به کاربر نشان دهند.

۴- رویداد Click دکمه فرمان Move را به این صورت تنظیم کنید:

```
Private Sub cmdmovefile_Click()

  Dim myfso As New FileSystemObject

  myfso.MoveFile flbsource.Path + "\" + _

    flbsource.FileName, flbdestination.Path + "\"

End Sub
```

در این رویداد مسیر و نام فایل مبدأ به ترتیب با استفاده از خاصیت Path و FileName کنترل کادرلیست فایل flbsource به‌دست می‌آید و در اختیار متد MoveFile قرار می‌گیرد. به‌علاوه مسیری که فایل انتخاب شده در کادرلیست فایل flbsource به آنجا منتقل می‌شود نیز با استفاده از خاصیت Path کنترل کادرلیست فایل flbdestination به‌دست آمده و در اختیار متد MoveFile قرار می‌گیرد تا فایل را به محل تعیین شده انتقال دهد.

۵- پروژه و فرم را با نام MoveFile ذخیره کنید و سپس پروژه را اجرا نمایید.

۶- در کادرمتن موجود در کنترل قاب Source، مسیری را که فایل شما در آنجا قرار دارد (مبدأ)، تایپ کنید و کلید Enter را بفشارید، در این لحظه کادرلیست فایل flbsource نام و پسوند فایل‌های موجود در مسیر تعیین شده را نشان می‌دهد. یک فایل را در کنترل کادرلیست فایل انتخاب کنید.

۷- در کادرمتن موجود در کنترل قاب Destination نیز درایو مسیر مقصد را تعیین نموده و کلید Enter را بفشارید، کنترل کادرلیست فایل flbdistination اسامی فایل‌های موجود در مسیر تعیین شده را نشان می‌دهد. بررسی کنید آیا فایل مبدأ در مسیر مقصد موجود می‌باشد یا خیر.

۸- روی دکمه فرمان Move کلیک کنید تا فایل تعیین شده به مسیر موردنظر منتقل شود، نتیجه را به‌وسیله برنامه My Computer بررسی کنید.

۹- اجرای برنامه را متوقف کرده و به پنجره ویژوال‌بیسیک بازگردید.

۱-۳-۱۹ متد GetFile

به وسیله این متد می‌توانید اطلاعات مورد نیاز خود را در رابطه با یک فایل مانند تاریخ تشکیل، تاریخ آخرین ویرایش، تاریخ آخرین دستیابی، حجم فایل و غیره به‌دست آورید. شکل کلی نحوه استفاده از این متد به این صورت است:

نام شئ از نوع FSO.GetFile (filename)

آرگومان filename از نوع رشته‌ای است و نام و مسیر فایل موردنظر را تعیین می‌کند. به منظور ذخیره‌سازی اطلاعات مربوط به فایل که به‌وسیله این متد به‌دست می‌آید می‌توان از دستور Set استفاده نمود و اطلاعات به‌دست آمده را در یک متغیر از نوع Variant یا از نوع شئ File ذخیره کرد، سپس با استفاده از خاصیت‌های ارائه شده در جدول ۶-۱ به این اطلاعات دسترسی پیدا نمود.

جدول ۶-۱ خاصیت‌های مربوط به شئ File

نام خاصیت	توضیح
Attributes	صفت‌های فایل را با توجه به جدول ۴-۱ تنظیم کرده یا مشخص می‌کند.
DateCreated	تاریخ تشکیل فایل را مشخص می‌کند.
DateLastAccessed	تاریخ آخرین دسترسی به فایل را مشخص می‌کند.
Drive	نام درایوی را که فایل در آن قرار گرفته است مشخص می‌کند.
DateLastModified	تاریخ آخرین ویرایش فایل را مشخص می‌کند.
Name	نام پوشه را تنظیم کرده یا مشخص می‌کند.
ParentFolder	نام پوشه‌ای را که فایل در آن قرار دارد مشخص می‌کند.
Path	مسیری را که فایل در آن قرار دارد مشخص می‌کند.
Size	اندازه فایل را مشخص می‌کند.
Type	نوع فایل را مشخص می‌کند.
Shortname	نام فایل را به‌صورت خلاصه ارائه می‌کند (۸ کاراکتر برای نام و ۳ کاراکتر برای پسوند).
ShortPath	مسیر فایل را به‌صورت خلاصه ارائه می‌کند (۸ کاراکتر برای نام و ۳ کاراکتر برای پسوند).

مثال ۶: می‌خواهیم یک پروژه از نوع Standard EXE همراه با یک فرم مانند شکل ۱۸-۱ طراحی کنیم که با انتخاب فایل موردنظر بتوان مشخصات آن را مشاهده نمود. برای این کار عملیات زیر را به ترتیب انجام دهید.

شکل ۱۸-۱

۱- برنامه ویژوال‌بیسیک را اجرا کنید و یک پروژه از نوع Standard EXE به همراه یک فرم و کنترل‌های آن مانند شکل ۱۸-۱ ایجاد کنید.

۲- رویداد Change کنترل‌های drvdrive و dirpath را به این صورت تنظیم کنید:

```
Private Sub drvdrive_Change()
   dirpath.Path = drvdrive.Drive
   filfile.Path = dirpath.Path
End Sub
Private Sub dirpath_Change()
      filfile.Path = dirpath.Path
End Sub
```

در این رویدادها با استفاده از خاصیت Drive در کنترل drvdrive و خاصیت Path در کنترل‌های dirpath و filfile به گونه‌ای عمل می‌شود تا با تغییر درایو بلافاصله کنترل‌های DirListBox و FileListBox تغییرات را دریافت کرده و پوشه‌ها و فایل‌های درایو انتخاب شده را نمایش دهند، به‌علاوه با تغییر پوشه در کنترل DirListBox و باز کردن یک پوشه جدید نیز کنترل FileListBox تغییرات را دریافت کرده و فایل‌ها را روی مسیر جدید نمایش دهد.

۳- رویداد Click کنترل filfile را به این صورت تنظیم کنید:

```
Private Sub filfile_Click()
    Dim myfso As New FileSystemObject
    Dim myfile As File
    Set myfile = myfso.GetFile (filfile.Path + "\" + filfile.FileName)
    lblname.Caption = myfile.Name
    lbldatecreated.Caption = myfile.DateCreated
    lblparentfolder.Caption = myfile.ParentFolder
    lbldrive.Caption = myfile.Drive
    lblreadonly.Caption = myfile.Attributes
    lbltype.Caption = myfile.Type
    lblsize.Caption = myfile.Size
End Sub
```

در این رویداد ابتدا یک شئ از نوع FileSystemObject و یک شئ از نوع File ایجاد می‌شوند، سپس با استفاده از متد GetFile شئ myfso اطلاعات فایل انتخاب شده در کنترل filfile در شئ myfile قرار می‌گیرند و بعد از آن با استفاده از خاصیت‌های شئ myfile اطلاعات مربوط به فایل، روی فرم نمایش داده می‌شوند.

۴- فرم و پروژه را با نام Fileinformation ذخیره کنید و پروژه را اجرا نمایید سپس در درایو و پوشه‌های سیستم خود فایل موردنظر را انتخاب کنید و نتیجه به‌دست آمده را بررسی نمایید.

۵- اجرای برنامه را متوقف کرده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: مثال MoveFile را به گونه‌ای تغییر دهید تا برای تعیین درایو و مسیر مبدأ و مقصد از کنترل‌های کادرلیست درایو و کادرلیست پوشه استفاده شود.

۲۰-۳-۱ متد CreateTextFile

به وسیله این متد می‌توانید یک فایل متنی در مسیر موردنظر خود ایجاد کنید. شکل کلی نحوه استفاده از این متد به این صورت می‌باشد:

نام شئ از نوع FSO.CreateTextFile filename,overwrite

این متد دارای یک آرگومان اجباری و یک آرگومان اختیاری است. آرگومان filename از نوع رشته‌ای است

و نام و مسیر فایل متنی را که ایجاد می‌شود تعیین می‌کند. آرگومان overwrite اختیاری است و از نوع منطقی می‌باشد، در صورتی‌که مقدار آن True باشد و فایل در مسیر تعیین شده وجود داشته باشد عمل رونویسی فایل انجام می‌شود و فایل قبلی از بین می‌رود اما در صورتی‌که مقدار آن False باشد و فایل در مسیر تعیین شده وجود داشته باشد اجرای متد سبب ایجاد خطا شده و پیام خطایی مانند شکل ۳-۱ نمایش داده می‌شود. در صورت عدم استفاده از این آرگومان مقدار پیش‌فرض آن True می‌باشد.

به عنوان مثال این دستورات یک فایل متنی با نام Letter در فهرست ریشه درایو :D ایجاد می‌کنند.

```
Dim myfso As New FileSystemObject

myfso.CreateTextFile  "D:\Letter.txt"
```

نکته: این متد فقط یک فایل خالی را روی مسیر تعیین شده ایجاد می‌کند و هیچ محتویاتی را در آن ذخیره نمی‌کند.

نکته: در صورتی‌که مسیر تعیین شده اشتباه باشد پیام خطای Path not found مانند شکل ۵-۱ نمایش داده می‌شود.

تمرین: تابعی بنویسید که مسیر و نام یک فایل متنی را دریافت کند و در صورت موجود بودن فایل از ایجاد آن خودداری کند و مقدار False را بازگشت دهد، اما اگر فایل در مسیر تعیین شده وجود نداشته باشد آن را ایجاد کند و مقدار True را بازگشت دهد.

۲۱-۳-۱ متد GetTempName

این متد یک نام فایل موقت به‌صورت تصادفی و با پسوند tmp ایجاد می‌کند. باید توجه داشت که این متد به هیچ عنوان فایلی به وجود نمی‌آورد بلکه یک نام فایل تولید می‌کند که می‌توان از آن در متد CreateTextFile یا سایر متدهای مرتبط برای ایجاد یک فایل موقت در هنگام اجرای برنامه‌ها استفاده کرد. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

نام شئ از نوع FSO.GetTempName

مقدار بازگشتی این متد یک داده رشته‌ای است که نام یک فایل موقت را دربرمی‌گیرد. به عنوان مثال در این دستورات ابتدا یک نام فایل موقت با استفاده از متد GetTempName تولید می‌شود، سپس به‌وسیله متد CreateTextFile یک فایل متنی با نام فایل موقت ایجاد می‌گردد.

```
Dim myfso As New FileSystemObject
```

```
Dim tempfile As String
tempfile = myfso.GetTempName
myfso.CreatTextFile ("D:\" + tempfile)
```

نکته: پسوند نام فایلی که به وسیله متد **GetTempName** تولید می‌شود، **tmp** می‌باشد.

۲۲-۳-۱ متد OpenTextFile

به وسیله این متد می‌توانید یک فایل متنی با قالب ASCII یا Unicode ایجاد کنید یا در یک فایل متنی اطلاعاتی را نوشته یا از آن اطلاعات ذخیره شده را بازیابی کنید. شکل کلی نحوه استفاده از این متد به‌صورت زیر است:

نام شئ از نوع FSO.openTextFile filename,iomode,create,format

این متد دارای یک آرگومان اجباری filename و سه آرگومان اختیاری iomode، create و format می‌باشد. آرگومان filename از نوع رشته‌ای می‌باشد و نام و مسیر فایل را معین می‌کند. آرگومان iomode حالت و نحوه استفاده از فایل را مشخص می‌کند و می‌تواند یکی از مقادیر ارائه شده در جدول ۷-۱ را کسب کند. در صورت عدم استفاده از این آرگومان فایل به منظور خواندن اطلاعات باز می‌شود.

جدول ۷-۱ مقادیر مربوط به آرگومان iomode

ثابت رشته‌ای	ثابت عددی	توضیح
ForWriting	۱	فایل برای نوشتن اطلاعات باز می‌شود.
ForReading	۲	فایل برای خواندن اطلاعات باز می‌شود.
ForAppending	۸	فایل برای اضافه کردن اطلاعات به محتویات قبلی باز می‌شود.

آرگومان create از نوع منطقی می‌باشد، اگر مقدار آن True باشد یک فایل جدید ایجاد می‌شود و اگر فایل در مسیر تعیین شده وجود داشته باشد، رونویسی خواهد شد اما اگر مقدار این آرگومان False باشد در صورت وجود فایل در مسیر تعیین شده فایل جدید ایجاد می‌شود و فایل قبلی را رونویسی می‌کند ولی اگر فایل در مسیر تعیین شده موجود نباشد پیام خطایی مانند شکل ۴-۱ نمایش داده خواهد شد، در صورت عدم

استفاده از این آرگومان مقدار False مورد استفاده قرار می‌گیرد.

آخرین آرگومان یعنی format، استاندارد ذخیره‌سازی اطلاعات در فایل را تعیین می‌کند، اگر استاندارد ASCII انتخاب شود یک بایت برای ذخیره‌سازی هر کاراکتر استفاده می‌شود در صورتی‌که از استاندارد Unicode استفاده شود دو بایت برای ذخیره‌سازی هر کاراکتر استفاده می‌شود (جدول ۸-۱).

جدول ۸-۱ مقادیر مربوط به آرگومان format

توضیح	ثابت عددی	ثابت رشته‌ای
فرمت Unicode	-۱	TristateTrue
فرمت ASCII	۰	TristateFalse

در صورت عدم استفاده از این آرگومان استاندارد ASCII مورد استفاده قرار می‌گیرد.

نکته: متد OpenTextFile به تنهایی هیچ اطلاعاتی را در فایل ذخیره نمی‌کند بلکه فقط فایل را برای خواندن و نوشتن اطلاعات آماده می‌کند.

به عنوان مثال این دستورات یک فایل متنی با استاندارد ASCII به منظور نوشتن اطلاعات با نام Letter در فهرست ریشه درایو D: باز می‌کنند.

```
Dim myfso As New FileSystemObject
myfso.OpenTextFile  "D:\ Letter.txt", ForWriting, True, ۰
```

به عنوان مثالی دیگر، به این دستورات توجه کنید:

```
Dim myfso As New FileSystemObject
myfso. OpenTextFile "D:\ Letter.txt", ForAppending
```

این دستورات فایل متنی موجود در ریشه درایو D: را برای افزودن اطلاعات به محتویات آن باز می‌کنند.

نکته: در صورتی‌که متد OpenTextFile با مقدار ForAppending برای اضافه کردن اطلاعات به فایل اجرا شود اما فایل در مسیر تعیین شده موجود نباشد، فایل جدید ایجاد می‌شود.

۲۳-۳-۱ شئ TextStream

متدهای OpenTextFile و CreateTextFile، یک شئ از نوع TextStream را به محل فراخوانی خود بازگشت می‌دهند. این شئ امکان انجام عملیات مختلف مانند خواندن و نوشتن به روش ترتیبی را روی فایل‌ها فراهم می‌کند. در روش دسترسی ترتیبی، داده و اطلاعات به همان ترتیبی که روی فایل در یک رسانه ذخیره‌سازی مانند دیسک ذخیره می‌شوند بازیابی شده و مورد دسترسی قرار می‌گیرند. این شئ متدهای مختلفی را برای خواندن و نوشتن داده‌ها و اطلاعات در فایل ارائه می‌کند. شکل کلی نحوه تعریف یک شئ از نوع TextStream به‌صورت زیر می‌باشد:

Dim نام شئ As TextStream

متد Write

به وسیله این متد می‌توان یک رشته را در فایل موردنظر ذخیره کرد. متد Write رشته موردنظر را پشت‌سر رشته‌ای که متد Write قبل از آن نوشته است، ذخیره می‌کند. شکل کلی نحوه استفاده از این متد به‌صورت زیر می‌باشد:

نام شئ از نوع TextStream.Write (string)

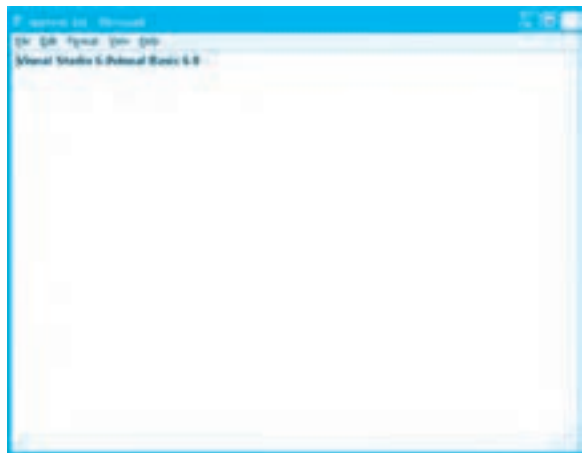
آرگومان string از نوع رشته‌ای است و رشته موردنظر برای نوشته شدن در فایل را مشخص می‌کند.

به عنوان مثال این دستورات یک فایل متنی را ایجاد کرده و عبارت‌های Visual Basic 6.0 و Visual Studio 6.0 را در آن ذخیره می‌کند.

```
Dim myfso As New FileSystemObject
Dim myfile As TextStream
Set myfile = myfso. OpenTextFile ("D:\ mytext.txt", For Writing, True)
myfile.Write ("Visual Studio 6.0")
myfile.Write ("Visual Basic 6.0")
myfile.Close
```

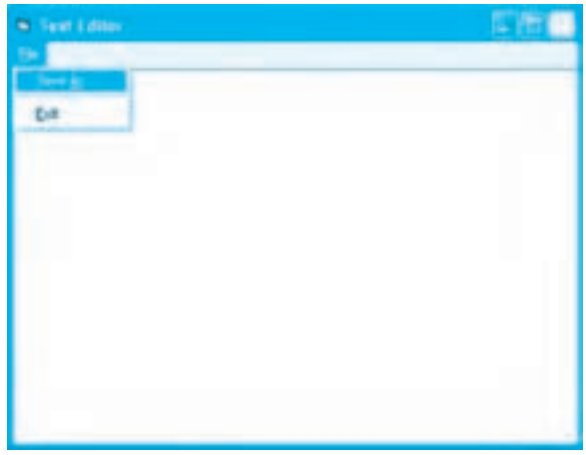
در این دستورات ابتدا یک شئ از نوع FSO تعریف می‌شود تا به‌وسیله آن بتوان متد OpenTextFile را اجرا کرد و فایل را برای ذخیره کردن اطلاعات باز نمود، سپس یک شئ (myfile) از نوع TextStream تعریف می‌شود و از این شئ برای اجرای متد Write که عملیات نوشتن در فایل را انجام می‌دهد، استفاده می‌شود. در مرحله بعد با استفاده از متد OpenTextFile فایل mytext.txt روی درایو D:\ به منظور انجام عملیات نوشتن (ForWriting) باز می‌شود. آرگومان سوم در این متد True می‌باشد تا در صورت عدم وجود فایل در مسیر تعیین شده خطایی به وجود نیاید.

پس از باز شدن فایل به‌وسیله متد OpenTextFile این متد یک شئ از نوع TextStream را بازگشت می‌دهد. دستور Set این شئ را در اختیار شئ myfile که خود نیز از نوع TextStream است قرار می‌دهد. با ایجاد این ارتباط بین فایل باز شده و شئ myfile می‌توان عملیات نوشتن اطلاعات در فایل را انجام داد. سپس با استفاده از متدهای Write در شئ myfile عبارت‌های Visula Studio 6.0 و Visual Basic 6.0 در فایل mytext نوشته می‌شوند و در مرحله آخر متد Close اجرا می‌گردد تا فایل بسته شود و انجام عملیات روی فایل خاتمه پیدا کند و در صورت لزوم سایر برنامه‌ها بتوانند از این فایل استفاده کنند. اگر فایل mytext.txt را با برنامه NotePad ویندوز باز کنید عبارت‌های ذخیره شده را مانند شکل ۱۹-۱ پشت سر هم مشاهده خواهید نمود.



شکل ۱۹-۱

مثال ۷: می‌خواهیم پروژه‌ای مطابق شکل ۲۰-۱ طراحی کنیم تا به وسیله آن بتوان یک فایل متنی دلخواه را ایجاد نمود، محتویات آن را تایپ و در صورت نیاز آن را ذخیره نمود. برای این کار عملیات بعد را به ترتیب انجام دهید:



شکل ۲۰-۱

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه جدید از نوع Standard EXE همراه با یک فرم و یک منو مانند شکل ۲۰-۱ ایجاد کنید.

۲- یک کنترل کادرمتن با نام txttext مانند شکل ۲۰-۱ روی فرم قرار دهید و خاصیت Text آن را خالی

۳- کنید و سپس خاصیت multiLine آن را روی مقدار True تنظیم نمایید.

رویداد munsaveas_Click مربوط به گزینه Save As ... در منوی File را به این صورت تنظیم کنید:

```
Private Sub mnusaveas_Click()
   Dim strfilename As String
   Dim myfso As New FileSystemObject
   Dim myfile As TextStream
   strfilename = InputBox("Enter Path and Filename :", "Save As")
   If strfilename <> "" Then
      Set myfile = myfso.OpenTextFile(strfilename, ForWriting, True)
      myfile.Write (txttext.Text)
      myfile.Close
   End If
```

End Sub

در این رویداد ابتدا متغیرها و اشیای مورد نیاز تعریف می‌شوند، سپس به وسیله یک کادر ورود داده (InputBox)، نام و مسیر فایل از کاربر دریافت می‌شود. در مرحله بعد با استفاده از دستور شرطی If پاسخ کاربر بررسی می‌شود تا در صورتی‌که نام و مسیر فایل تعیین و در کادر ورود داده روی دکمه OK کلیک شود متد OpenTextFile اجرا شود و سپس متد Write محتویات کادرمتن txt text را در فایل ذخیره کند، اما اگر کاربر در کادر ورود داده روی دکمه Cancel کلیک کند یا نام و مسیر فایل را تعیین نکند هیچ عملی انجام نخواهد شد.

۴- پروژه و فرم را با نام texteditor ذخیره کنید، سپس پروژه را اجرا کنید.

۵- یک متن دلخواه را در کادرمتن برنامه تایپ کنید، سپس روی گزینه Save As در منوی File کلیک کنید. مسیر و نام فایل را در کادر ورود داده تایپ نمایید و روی دکمه OK کلیک کنید.

۶- اجرای برنامه را خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

۷- با استفاده از برنامه NotePad ویندوز، فایل متنی ذخیره شده را باز کنید و محتویات آن را بررسی نمایید.

تمرین: با استفاده از متد Write چند عبارت دیگر را به محتویات فایل mytext.txt اضافه کنید و پس از اجرای دستورات محتویات آن را با برنامه NotePad مشاهده کرده و نتیجه را بررسی نمایید.

متد WriteLine

به وسیله این متد نیز می‌توان یک رشته را در فایل موردنظر ذخیره نمود. تفاوت این متد با متد Write این است که در این متد پس از نوشته شدن رشته تعیین شده، کاراکتر خط جدید (New Line Character) را در انتهای آن ذخیره می‌کند. این کاراکتر امکان ذخیره‌سازی رشته‌ها را به‌صورت سطرهای جداگانه در داخل فایل متنی فراهم می‌نماید. شکل کلی نحوه استفاده از این متد به‌صورت زیر می‌باشد:

نام شئ از نوع TextStream.WriteLine (string)

آرگومان string از نوع رشته‌ای است و رشته موردنظر برای نوشته شدن در فایل را مشخص می‌کند.

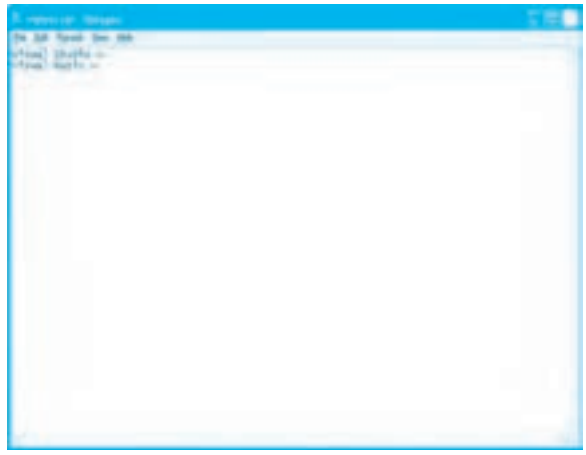
به عنوان مثال این دستورات فایل متنی mytext.txt را به منظور اضافه کردن اطلاعات باز کرده و عبارت‌های Visual Studio 6.0 و Visual Basic 6.0 را در آن ذخیره می‌کند.

```
Dim myfso As New FileSystemObject
Dim myfile As  TextStream
```

```
Set myfile = myfso.OpenTextFile ("D:\ my text.txt" , ForWriting, True)
myfile.WriteLine ("Visual Studio 6.0")
myfile.Write ("Visual Basic 6,0")
myfile.Close
```

در این دستورات ابتدا یک شئ از نوع FSO و myfile تعریف می‌شوند، سپس متد OpenTextFile فایل را برای ذخیره کردن اطلاعات در آن باز می‌کند و دستورات بعدی با اجرای متدهای WriteLine عبارت Visual Studio 6.0 و متد Write عبارت Visual Basic 6.0 را در فایل می‌نویسند و آخرین دستور با اجرای متد Close فایل را می‌بندد.

اگر فایل mytext.txt را با برنامه NotePad ویندوز باز کنید، عبارت‌های ذخیره شده را مانند شکل ۱-۲۱ مشاهده خواهید کرد. عبارت Visual Basic 6.0 در زیر عبارت Visual Studio 6.0 قرار دارد زیرا عبارت اول با استفاده از متد WriteLine ذخیره شده است که در انتهای عبارت کاراکتر خط جدید را نیز ذخیره می‌کند و این امر سبب می‌شود عبارت دوم در خط بعدی نمایش داده شود.



شکل ۱-۲۱

تمرین: با استفاده از متد WriteLine و Write چند عبارت دیگر را به محتویات فایل mytext.txt اضافه کنید و پس از اجرای دستورات محتویات آن را با برنامه NotePad مشاهده کرده و نتیجه را بررسی نمایید.

مثال ۸: می‌خواهیم پروژه‌ای مطابق شکل ۱-۲۲ طراحی کنیم تا به وسیله آن بتوان نام و نام‌خانوادگی تعدادی از دانش‌آموزان را در یک فایل متنی ذخیره نمود. برای این کار عملیات بعد را به ترتیب انجام دهید:

شکل ۲۲-۱

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه با یک فرم و کنترل‌های آن مانند شکل ۲۲-۱ ایجاد کنید.

۲- در بخش تعاریف ماژول فرم این دستورات را بنویسید:

```
Dim myfso As New FileSystemObject
Dim myfile As textStream
```

شئ myfso و myfile در بخش تعاریف فرم تعریف شده‌اند تا در تمام رویدادها قابل شناسایی باشند.

۳- در رویداد Load فرم با استفاده از متد OpenTextFile فایل را باز کنید:

```
Set myfile = myfso.OpenTextfile ("D:\ information.txt", For Appending, True)
```

۴- این دستورات را در رویداد Click دکمه Save تایپ کنید:

```
myfile.WriteLine (txtname.Text)
myfile.WriteLine (txtfamily.Text)
```

در این رویداد با استفاده از متد WriteLine نام و نام‌خانوادگی دانش‌آموزان که در کادرهای متن txtname و txtfamily تایپ شده‌اند همراه با کاراکتر خط جدید در فایل ذخیره می‌شوند تا نام و نام‌خانوادگی هر شخص به‌طور جداگانه قابل دسترسی باشد.

۵- در رویداد unload فرم فایل را با استفاده از متد Close ببندید تا در هنگام بسته شدن فرم و خروج از برنامه فایل نیز بسته شود.

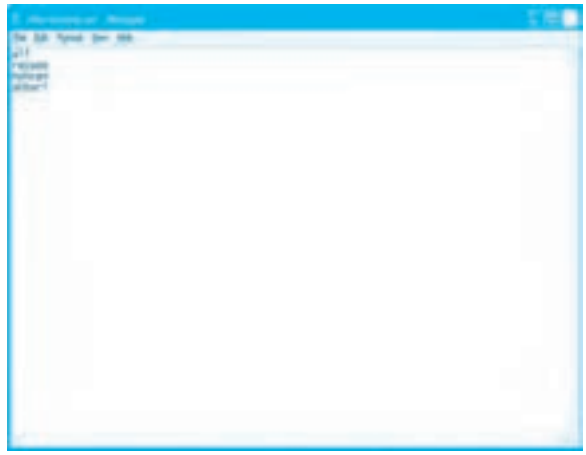
```
myfile.Close
```

۶- پروژه و فرم را با نام Students ذخیره کنید، سپس پروژه را اجرا نمایید.

۷- نام و نام‌خانوادگی یک دانش‌آموز را در کادرهای متن تایپ کنید و روی دکمه Save کلیک کنید به همین ترتیب نام و نام خانوادگی چند دانش‌آموز دیگر را در فایل ذخیره نمایید.

۸- اجرای برنامه را متوقف کنید و به پنجره ویژوال‌بیسیک بازگردید.

۹- فایل information.txt را با برنامه NotePad باز کنید و محتویات آن را مشاهده نمایید. (شکل ۲۳-۱)



شکل ۲۳-۱

تمرین: پروژه Students را به شکلی تغییر دهید تا علاوه بر نام و نام‌خانوادگی دانش‌آموز بتوان کد دانش‌آموزی وی را نیز ذخیره کرد.

متد Read

این متد می‌تواند یک یا تعدادی کاراکتر را از فایل متنی به همان ترتیبی که نوشته شده‌اند بخواند و به‌صورت یک رشته بازگشت دهد. شکل کلی نحوه استفاده از این متد به این صورت می‌باشد:

نام شئ از نوع TextStream.Read (characters)

آرگومان characters از نوع عددی صحیح و مثبت است و تعداد کاراکترهایی را که در هر بار اجرای متد از فایل خوانده می‌شوند، مشخص می‌کند. به عنوان مثال این دستورات فایل متنی mytext.txt را باز کرده و محتویات آن را با استفاده از متد Read می‌خواند.

```
Dim myfso As New FileSystemObject

Dim myfile As textStream

Set myfile = myfso.OpenTextFile ("d:\ mytext.txt" , ForReading)

Do While (myfile AtEndOfStream <> True)

    Print  myfile. Read (1)

Loop

myfile.Close
```

در این دستورات ابتدا اشیای مورد نیاز برای کار با فایل ساخته شده‌اند، سپس متد OpenTextFile فایل mytext.txt را که در مثال‌های قبل ایجاد شده است برای خواندن باز می‌کند. در مرحله بعد متد Read با استفاده از یک حلقه خواندن محتویات فایل را به همان ترتیبی که داده‌ها ذخیره شده‌اند بازیابی می‌کند.

دلیل استفاده از حلقه این است که چون تعداد کاراکترهای ذخیره شده در فایل مشخص نیست در نتیجه باید از یک حلقه با تکرار نامعین مثل حلقه‌های while یا until استفاده کرد، در ضمن برای کنترل عملیات خواندن از خاصیت AtEndOfStream در شرط حلقه استفاده شده است.

این خاصیت در صورتی‌که به انتهای فایل رسیده باشیم مقدار True و در غیر این صورت مقدار False را ارائه می‌کند، با این روش وقتی تمام محتویات فایل خوانده شود اجرای حلقه نیز خاتمه می‌یابد.

متد Read نیز در حلقه هر بار یک کاراکتر را از فایل خوانده و آن را به‌صورت یک رشته در اختیار دستور Print قرار می‌دهد تا نمایش دهد. در خاتمه نیز فایل با متد Close بسته می‌شود. نتیجه اجرای این دستورات مانند شکل ۲۴-۱ می‌باشد.

نکته: در صورتی‌که بدون استفاده از خاصیت AtEndOfStream عملیات خواندن اطلاعات انجام شود با رسیدن به انتهای فایل پیام خطایی مشابه شکل ۲۴-۱ نمایش داده می‌شود.

شکل ۲۴-۱

تمرین: پروژه texteditor را به گونه‌ای تنظیم کنید که به وسیله آن بتوان علاوه بر ذخیره‌سازی متن روی دیسک، فایل‌های موجود را خوانده و محتویات آن‌ها را داخل کادرمتن مشاهده کرد.

متد ReadLine

عملکرد این متد شبیه به متد Read می‌باشد با این تفاوت که این متد تمام کاراکترها را تا رسیدن به کاراکتر خط جدید در یک مرحله خوانده و نتیجه را به‌صورت یک رشته بازمی‌گرداند. شکل کلی این متد به این صورت می‌باشد:

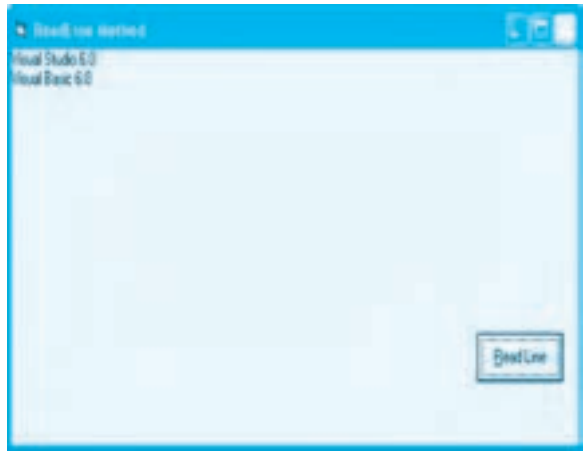
نام شئ از نوع TextStream.ReadLine

به عنوان مثال این دستورات فایل متنی mytext.txt را باز کرده و محتویات آن را با استفاده از متد ReadLine می‌خواند.

```
Dim myfso As New FileSystemObject
Dim myfile As TextStream
Set myfile = myfso.OpenTextFile ("d:\ mytext.txt" , ForReading)
Do While (myfile.AtEndOfStream <> True)
    Print myfile.ReadLine
Loop
myfile.Close
```

در این دستورات نیز ابتدا اشیای مورد نیاز برای انجام عملیات خواندن روی فایل ایجاد می‌شوند سپس

محتویات فایل با متد ReadLine بازیابی شده و مانند شکل ۲۵-۱ نمایش داده می‌شوند.



شکل ۲۵-۱

تمرین: پروژه students را به شکلی تنظیم کنید تا بتوان اطلاعات تمام دانش‌آموزان را روی یک فرم مشاهده نمود.

متد ReadAll

عملکرد این متد شبیه به متد ReadLine است با این تفاوت که این متد تمام محتویات فایل متنی را به‌طور همزمان و در یک مرحله تا رسیدن به انتهای فایل می‌خواند و به‌صورت یک رشته واحد بازمی‌گرداند. شکل کلی این متد به این صورت می‌باشد:

نام شئ از نوع TextStream.ReadAll

به عنوان مثال این دستورات فایل متنی mytext.txt را باز کرده و محتویات آن را با استفاده از یک متد ReadAll و بدون استفاده از حلقه می‌خواند:

```
Dim myfso As New FileStreamObject

Dim myfile As TextStream

Set myfile = myfso.OpenTextFile ("d:\ mytext.txt" , ForReading)

Print  myfile.ReadAll

myfile.Close
```

تمرین: پروژه texteditor را طوری تنظیم کنید تا برای خواندن فایل‌های متنی به جای متد Read از متد ReadAll استفاده شود.

۲۴-۳-۱ متد OpenAsTextStream

عملکرد این متد مشابه متد OpenTextFile است و به وسیله آن می‌توان یک فایل متنی ایجاد نمود و داده‌ها را در آن ذخیره کرد، اطلاعاتی را به یک فایل متنی موجود اضافه کرد یا محتویات یک فایل را خواند. شکل کلی این متد به‌صورت زیر است:

نام شئ از نوع File.OpenAsTextStream (iomode,format)

این متد دارای دو آرگومان اختیاری است. آرگومان iomode حالت استفاده از فایل را مشخص می‌کند و می‌تواند یکی از مقادیر ارائه شده در جدول ۷-۱ را کسب کند. در صورت عدم استفاده از این آرگومان فایل به منظور خواندن اطلاعات باز می‌شود.

آرگومان format نیز استاندارد ذخیره‌سازی اطلاعات در فایل را تعیین می‌کند و می‌تواند یکی از مقادیر ارائه شده در جدول ۷-۱ را کسب کند، در صورتی‌که این آرگومان حذف شود از استاندارد ASCII استفاده می‌شود.

این متد پس از باز کردن فایل یک شئ از نوع TextStream را بازمی‌گرداند.

به عنوان مثال این دستورات فایل color.txt را برای ذخیره‌سازی اطلاعات باز می‌کنند و اسامی رنگ‌ها را در آن می‌نویسند:

```
Dim myfso As New FileSystemObject
Dim myfile As TextStream
Dim textfile As File
myfso.CreateTextFile "d:\ color.txt"
Set textfile = myfso.GetFile ("d:\ color.txt")
Set myfile = textfile.OpenAsTextStream (ForWriting)
myfile.WriteLine ("Red")
myfile.WriteLine ("Green")
myfile.WriteLine ("Yellow")
myfile.Close
```

در این دستورات پس از تعریف اشیا، ابتدا یک فایل متنی جدید با استفاده از متد CreateTextFile ایجاد می‌شود سپس با استفاده از متد GetFile یک شئ از فایل color.txt در اختیار شئ textfile که از نوع شئ File می‌باشد، قرار می‌گیرد. در مرحله بعد متد OpenAsTextStream شئ textfile فایل color را برای نوشتن داده‌ها در آن باز می‌کند و یک شئ از نوع TextStream را در اختیار شئ myfile قرار می‌دهد و بعد از اجرای این متد، سه متد WriteLine اسامی رنگ‌ها را در فایل می‌نویسند.

Learn in English

CreateTextFile Method

Description

Creates a specified file name and returns a **TextStream** object that can be used to read from or write to the file.

Syntax

object.**CreateTextFile**(*filename[, overwrite[, unicode]]*)

The **CreateTextFile** method has these parts:

Part	Description
object	Required. Always the name of a FileSystemObject or Folder object.
filename	Required. String expression that identifies the file to create.
overwrite	Optional. Boolean value that indicates if an existing file can be overwritten. The value is True if the file can be overwritten; False if it can't be overwritten. If omitted, existing files are not overwritten.
unicode	Optional. Boolean value that indicates whether the file is created as a Unicode or ASCII file. The value is True if the file is created as a Unicode file; False if it,s created as an ASCII file. If omitted, an ASCII file is assumed.

واژه‌نامه

مقصد	Destination
رونویسی کردن	Overwrite
والد	Parent
مسیر	Path
طرح، الگو	Pattern
مبدأ، منبع	Source
موقت	Temporary

خلاصه مطالب

- مدل شئ FSO امکان انجام عملیات روی درایوها، پوشه‌ها و فایل‌ها را فراهم می‌کند.
- مدل شئ FSO شامل اشیایی مانند Drive، Folder، File، FileSystemObject و TextStream است.
- شئ Drive می‌تواند اطلاعات ارزشمندی در رابطه با درایوهای موجود در سیستم ارائه کند.
- شئ Folder می‌تواند اطلاعات ارزشمندی در رابطه با پوشه‌ها ارائه کند.
- شئ File می‌تواند اطلاعات ارزشمندی در رابطه با فایل‌ها ارائه کند.
- شئ FileSystemObject متدهای مختلفی برای انجام عملیات روی فایل‌ها و پوشه‌ها ارائه می‌کند.
- شئ TextStream امکان ایجاد فایل‌های متنی و عملیات خواندن و نوشتن به روش ترتیبی را روی فایل‌ها فراهم می‌کند.
- متد CopyFile برای نسخه‌برداری از یک یا گروهی از فایل‌ها استفاده می‌شود.
- متد CopyFolder برای نسخه‌برداری از یک یا چند پوشه استفاده می‌شود.
- متد CreatFolder، یک پوشه در مسیر تعیین شده ایجاد می‌کند.
- متد CreatTextFile یک فایل متنی در مسیر تعیین شده ایجاد می‌کند.
- متد DeleteFile برای حذف یک یا گروهی از فایل‌ها استفاده می‌شود.
- متد DeleteFolder برای حذف یک یا گروهی از پوشه‌ها استفاده می‌شود.
- متد DriveExists وجود یا عدم وجود یک درایو را روی سیستم مشخص می‌کند.

- متد FolderExists وجود یا عدم وجود یک پوشه را در مسیر تعیین شده مشخص می‌کند.
- متد FileExists وجود یا عدم وجود یک فایل را در مسیر تعیین شده مشخص می‌کند.
- متد GetAbsolutePathName مسیر کامل یک مسیر دلخواه را مشخص می‌کند.
- متد GetDrive اطلاعاتی در رابطه با درایو موردنظر ارائه می‌کند.
- متد GetDriveName یک مسیر را دریافت کرده و نام درایو آن را مشخص می‌کند.
- متد GetFolder اطلاعاتی در رابطه با پوشه موردنظر ارائه می‌کند.
- متد GetParentFolderName مسیر پوشه والد هر پوشه دلخواهی را مشخص می‌کند.
- متد GetTempName یک نام فایل موقت به‌صورت تصادفی و با پسوند tmp تولید می‌کند.
- متد GetFile اطلاعاتی در رابطه با فایل موردنظر ارائه می‌کند.
- متد MoveFile یک یا گروهی از فایل‌ها را از یک مسیر به مسیر دیگر منتقل می‌کند.
- متد MoveFolder یک یا چند پوشه را از یک مسیر به مسیر دیگر منتقل می‌کند.
- متد OpentTextFile می‌تواند یک فایل متنی با استاندارد ASCII یا Unicode ایجاد کند.
- متد OpenTextFile می‌تواند یک فایل متنی را برای خواندن یا نوشتن داده‌ها و اطلاعات باز کند.
- متد Write داده‌ها را به‌صورت متوالی داخل فایل می‌نویسد.
- متد WriteLine هر داده را همراه با کاراکتر خط جدید (NL) در داخل فایل متنی می‌نویسد.
- متد Read می‌تواند یک یا مجموعه‌ای از کاراکترها را از یک فایل متنی بخواند و به‌صورت یک رشته بازگشت دهد.
- متد ReadLine می‌تواند داده‌ها را تا رسیدن به کاراکتر خط جدید (NL) از یک فایل متنی بخواند و به‌صورت یک رشته بازگشت دهد.
- متد ReadAll می‌تواند تمام محتویات فایل متنی را در یک مرحله از فایل متنی بخواند و به‌صورت یک رشته بازگشت دهد.
- متد Close فایل را پس از خاتمه عملیات می‌بندد.
- متد OpenAsTextStream یک فایل متنی را برای خواندن و نوشتن یا اضافه کردن داده و اطلاعات باز می‌کند.
- از کنترل DriveListBox برای نمایش و انتخاب درایوهای موجود در یک سیستم استفاده می‌شود.
- از کنترل DirListBox برای نمایش و انتخاب پوشه‌ها استفاده می‌شود.
- از کنترل FileListBox برای نمایش و انتخاب فایل‌ها استفاده می‌شود.

آزمون نظری

۱- کدام شئ امکان انجام عملیات روی فایل‌های متنی را فراهم می‌کند؟

الف- Drive **ب-** Folder **ج-** Text **د-** TextStream

۲- کدام خاصیت نوع و روش آدرس‌دهی فایل‌ها و پوشه‌ها در یک درایو را مشخص می‌کند؟

الف- DriveType **ب-** FileSystem **ج-** DriveLetter **د-** FolderType

۳- کدام متد می‌تواند تمام کاراکترها را به‌طور همزمان از یک فایل متنی بخواند؟

الف- Read **ب-** ReadAll **ج-** ReadLine **د-** ReadEOF

۴- کدام گزینه در رابطه با متد GetTempName صحیح است؟

الف- یک فایل تصادفی ایجاد می‌کند. **ب-** یک فایل متنی ایجاد می‌کند.

ج- یک فایل موقت ایجاد می‌کند. **د-** به‌صورت تصادفی یک نام فایل موقت تولید می‌کند.

۵- کدام خاصیت از شئ Drive نوع درایو را مشخص می‌کند؟

الف- DriveType **ب-** DriveLetter **ج-** VolumeName **د-** SerialNumber

۶- اگر فایل‌های phone.txt و text.txt در مسیرهای تعیین شده موجود باشند نتیجه اجرای این دستورات چیست؟

```
Dim mycopy As New FileSystemObject
mycopy.CopyFile "c:\ text.txt", "D:\ phone.txt"
```

الف- عملیات کپی انجام نمی‌شود.

ب- پیام خطا نمایش داده می‌شود و عملیات کپی متوقف می‌شود.

ج- عملیات کپی انجام شده و فایل phone.txt رونویسی می‌شود.

د- عملیات کپی انجام شده و فایل text.txt رونویسی می‌شود.

۷- نتیجه اجرای دستور زیر چیست؟

```
Dim mydelete As New FileSystemObject
\mydelete.DeleteFolder "D:\ Documents\"
```

الف- پوشه Documents با محتویاتش حذف می‌شود.

ب- فقط محتویات پوشه Documents حذف می‌شوند.

ج- پیام خطای Path not found نمایش داده می‌شود.

د- فقط پوشه‌های موجود در پوشه Documents حذف می‌شوند.

۸- خاصیت ListIndex در کنترل DriveListBox ..... .

الف- اسامی درایوهای موجود روی سیستم را نگهداری می‌کند.

ب- نام درایو انتخاب شده در کنترل را مشخص می‌کند.
ج- شماره درایو انتخاب شده در کنترل را مشخص می‌کند.
د- نام و شماره درایو انتخاب شده در کنترل را مشخص می‌کند.

۹- کدام خاصیت تعداد زیرپوشه‌ها در یک پوشه را مشخص می‌کند؟

الف- ListIndex **ب-** Count **ج-** ListCount **د-** ListName

۱۰- کدام متد می‌تواند پوشه والد یک پوشه دلخواه را مشخص کند؟

الف- GetParent **ب-** GetParentFolder **ج-** GetParentFolderName **د-** GetParentName

۱۱- با توجه به این‌که مسیر D:\Winxp\System32\drivers مسیر جاری باشد نتیجه اجرای این دستورات چیست؟

```
Dim myfso As New FileSystemObject
Print myfso.GetAbsolutePathName ("D: ..\..)
```

الف- عبارت D: نمایش داده می‌شود. **ب-** عبارت D:\ نمایش داده می‌شود.
ج- عبارت D:\winxp نمایش داده می‌شود. **د-** عبارت D:\winxp\System32 نمایش داده می‌شود.

۱۲- با توجه به این‌که پوشه Tools وجود ندارد نتیجه اجرای این دستورات چیست؟

```
Dim myfso As New FileSystemObject
myfso.MoveFolder  "D:\VB6" , "C:\ Programs\ tools"
```

الف- پوشه VB6 با نام tools به پوشه Programs منتقل می‌شود.
ب- پیام خطا نمایش داده می‌شود.
ج- محتویات پوشه VB6 با نام Tools به پوشه Programs منتقل می‌شود.
د- پوشه VB6 بدون محتویاتش به پوشه Programs منتقل می‌شود.

۱۳- به وسیله کدام خاصیت در کنترل FileListBox می‌توان فایل‌ها را فیلتر کرد؟

الف- FileName **ب-** List **ج-** ListIndex **د-** Pattern

۱۴- نتیجه اجرای این دستورات چیست؟

```
Set myfso = CreateObject ("Scripting.FileSystemObject)
myfso.OpenTextFile "C:\drivers.txt" , ForWriting, True, TristateFalse
```

الف- یک فایل متنی جدید را با استاندارد ASCII برای ذخیره‌سازی داده‌ها باز می‌کنند.
ب- یک فایل متنی جدید را با استاندارد Unicode برای ذخیره‌سازی داده‌ها باز می‌کنند.
ج- یک فایل متنی جدید را با استاندارد ASCII برای خواندن اطلاعات باز می‌کنند.
د- یک فایل متنی جدید را با استاندارد ASCII برای خواندن و نوشتن اطلاعات باز می‌کنند.

۱۵- کدام متد می‌تواند شبیه به متد OpenTextFile از مدل شئ FSO عمل کند؟

الف- متد OpenTextFile از شئ File
ب- متد OpenAsTextStream از شئ File
ج- متد OpenAsTextStream از شئ FSO
د- متد OpenTextStream از شئ FSO

16- The Creat TextFile method Creates a text File and returns a ..... object.

a- Text **b-** Text Stream **c-** Text File **d-** Fso

۱۷- مدل شئ FileSystemObject و اشیای تشکیل‌دهنده آن را توضیح دهید.
۱۸- روش‌های تعریف شئ از نوع FSO را با ذکر مثال بیان کنید.
۱۹- کنترل‌های FileListBox، DirListBox و DriveListBox را همراه با خاصیت‌ها و رویدادهای هریک توضیح دهید.
۲۰- نحوه ایجاد فایل‌های متنی و نوشتن، خواندن و اضافه کردن داده‌ها و اطلاعات به آن‌ها را توضیح دهید.
۲۱- شئ TextStream را به همراه متدهای آن به‌طور کامل توضیح دهید.
۲۲- متدهای CopyFile، DeleteFile، FileExists، GetFile، MoveFile، CreateTextFile و GetTempName را همراه با کاربرد هریک توضیح دهید.
۲۳- متدهای CopyFolder، CreateFolder، DeleteFolder، MoveFolder، FolderExists، GetFolder، GetParentFolderName و GetAbsolutePathName را همراه با کاربرد هریک توضیح دهید.
۲۴- متدهای DriveExists، GetDrive و GetDriveName را همراه با کاربرد هریک توضیح دهید.
۲۵- نحوه ایجاد فایل متنی را با متد OpenAsTextStream توضیح دهید.

آزمون عملی

۱-پروژه‌ای طراحی کنید که با استفاده از کنترل‌های DirListBox، DriveListBox و FileListBox امکان انجام عملیات مختلف روی درایوها، پوشه‌ها و فایل‌ها مانند کپی کردن، انتقال، حذف کردن، ایجاد فایل‌ها و پوشه‌ها و دریافت اطلاعات در رابطه با یک درایو، فایل یا پوشه وجود داشته باشد.
۲-پروژه‌ای طراحی کنید که به وسیله آن بتوان نام، نام خانوادگی، آدرس و شماره تلفن افراد را در یک فایل ذخیره نمود، به‌علاوه امکان جستجوی تلفن افراد با استفاده از نام و نام خانوادگی آن‌ها نیز وجود داشته باشد.
۳-پروژه‌ای طراحی کنید که به وسیله آن بتوان نمرات هر درس دانش‌آموزان را به‌صورت دائم ذخیره نمود و در صورت نیاز کارنامه هر دانش‌آموز همراه با جمع نمرات، معدل و قبول یا رد شدن وی در هر درس نمایش داده شود (اطلاعاتی که ذخیره می‌شوند عبارتند از شماره دانش‌آموزی، نام درس، نمره درس).

واحد کار دوم

هدف جزیی

توانایی دسترسی به فایل‌ها
و خواندن و نوشتن در آن‌ها با روش تصادفی

زمان (ساعت)	
نظری	عملی
۸	۱۶

هدف‌های رفتاری

پس از مطالعه این واحد کار از فراگیر انتظار می‌رود که:

۱- روش دسترسی تصادفی به فایل‌ها را بداند.

۲- بتواند فایل را با دسترسی تصادفی باز کرده و ببندد و عملیات خواندن و نوشتن داده‌ها را روی آن انجام دهد و با دستورات Open،Put و Get کار کند.

۳- نحوه تعریف یک نوع داده جدید را با دستور Type بداند.

۴- بتواند از نوع داده جدید تعریف شده یک رکورد ایجاد نماید.

۵- نحوه نوشتن و خواندن داده‌ها را به صورت رکورد بداند.

کلیات

قبلاً نحوه ذخیره‌سازی و بازیابی داده‌ها و اطلاعات به‌وسیله فایل‌های متنی و با دسترسی ترتیبی را فراگرفته‌اید. در این واحدکار می‌خواهیم این کار را با روش دستیابی تصادفی و روی فایل‌هایی با ساختار غیرمتنی انجام دهیم. در این روش داده‌ها و اطلاعات به همان صورت که در حافظه اصلی ذخیره می‌شوند در دیسک قرار می‌گیرند، به عبارت دیگر نوع داده‌ها در نحوه ذخیره‌سازی آن‌ها و حجم اشغالی داده در دیسک دخالت می‌کند. به عنوان مثال اگر یک عدد از نوع Integer در چنین فایلی ذخیره شود ۲ بایت، یک عدد از نوع صحیح بلند (Long)، ۴ بایت و یک کاراکتر اسکی ۱ بایت را اشغال می‌کند. بنابراین در این نوع از فایل‌ها می‌توان ترکیبی از چند نوع داده را نیز در کنار هم روی دیسک ذخیره نمود.

در ضمن در این روش لازم نیست داده‌ها به‌صورت ترتیبی در فایل ذخیره یا از آن بازیابی شوند بلکه علاوه بر این روش می‌توان به‌صورت مستقیم به هر بخش از داده‌ها و اطلاعات موردنظر دسترسی پیدا کرد یا در محل خاصی از فایل داده‌ها را ذخیره نمود.

به عبارت دیگر در روش ترتیبی اگر لازم باشد تا بخشی از داده‌ها خوانده، ذخیره یا ویرایش شود باید داده‌هایی که قبل از آن بخش قرار دارند خوانده شوند و نمی‌توان به‌صورت مستقیم به آن بخش دسترسی پیدا کرد، ولی در روش تصادفی می‌توان به‌طور مستقیم به هر بخش از داده‌ها دسترسی پیدا کرد، آن‌ها را خواند، ویرایش نمود یا در محلی از فایل داده‌هایی را ذخیره کرد، بدون آن‌که به سایر داده‌ها مراجعه شود.

۱-۲ نحوه باز کردن فایل‌ها با روش دسترسی تصادفی (مستقیم)

برای باز کردن فایل‌ها به روش دسترسی تصادفی می‌توان از دستور Open به این صورت استفاده کرد:

```
Open pathname For Random [Access access] As [#] filenumber Len= reclength
```

در این دستور pathname از نوع رشته‌ای است که نام و مسیر فایل را تعیین می‌کند. بخش Access اختیاری است که حالت فایل را برای انجام عملیات مشخص می‌کند. access می‌تواند یکی از مقادیر Read، Write یا Read Write را کسب کند. مقدار Read فایل را برای خواندن، مقدار Write، فایل را برای نوشتن و مقدار Read Write فایل را برای خواندن و نوشتن به‌طور همزمان باز می‌کند. بخش filenumber یک مقدار عددی بین یک تا ۵۱۲ است، از این مقدار عددی برای دسترسی به فایل جهت انجام عملیات خواندن یا نوشتن استفاده می‌شود. استفاده از کاراکتر # قبل از این بخش نیز اختیاری است.

بخش Len مقدار بایتی را که در هر بار، عملیات خواندن یا نوشتن براساس آن انجام می‌شود، تعیین می‌کند. reclength یک عدد صحیح بین ۱ تا ۳۲۷۶۷ می‌باشد.

به عنوان مثال این دستور فایل Data.dat را به منظور نوشتن داده‌ها با طول ۴ بایت و با شماره دسترسی ۲ باز می‌کند.

```
Open "D:\ Data. dat" For Random Access Write As#2 Len=4
```

به عنوان مثال دیگر این دستور فایل Data.dat را به منظور خواندن داده‌ها با طول ۴ بایت و با شماره دسترسی ۳ باز می‌کند.

```
Open "D:\ Data. dat" For Random Access Read As 3 Len=4
```

نکته: اگر فایل تعیین شده در pathname موجود نباشد فایل جدیدی ایجاد می‌شود.

نکته: هنگام باز کردن فایل‌ها باید دقت شود تا به‌طور همزمان از یک شماره دسترسی استفاده نگردد.

نکته: یک فایل را می‌توانید با استفاده از شماره‌های دسترسی متفاوت به‌طور همزمان بیش از یک بار باز کنید.

۲-۲ نحوه نوشتن داده‌ها در یک فایل با روش دستیابی تصادفی

پس از باز کردن فایل با دستور Open می‌توان با دستور Put داده‌ها را در آن ذخیره نمود. شکل کلی نحوه استفاده از این دستور به‌صورت زیر است:

```
Put # filenumber, [recnumber] , varname
```

filenumber شماره دسترسی فایل است که در دستور Open مورد استفاده قرار گرفته است. استفاده از کاراکتر # قبل از شماره دسترسی اختیاری است.

recnumber شماره داده‌ای را که در فایل نوشته می‌شود مشخص می‌کند، این شماره برای اولین داده مقدار ۱، برای دومین داده مقدار ۲ و به همین ترتیب تا آخرین داده ادامه می‌یابد و از نوع عددی صحیح و مثبت (Long) می‌باشد استفاده از این بخش نیز اختیاری است؛ در صورتی‌که از recnumber استفاده نشود عملیات نوشتن رکورد بعد، از موقعیت فعلی در فایل (رکورد جاری) انجام می‌شود.

نکته: در پایان عملیات ذخیره‌سازی اطلاعات در فایل، فایل را با استفاده از دستور Close#filenumber ببندید.

filenumber شماره دسترسی فایل است که در دستور Open مورد استفاده قرار گرفته است.

به عنوان مثال این دستورات فایل mydata.dat را با شماره دسترسی ۳ و طول رکورد ۲ بایت با روش دسترسی تصادفی برای نوشتن و ذخیره‌سازی داده‌ها باز می‌کنند و پس از نوشتن سه عدد صحیح در آن فایل بسته می‌شود.

```
Open "D:\ mydata.dat" For Random Access
    Write As 3 Len= 2
Put # 3,1,10
Put # 3,2,20
Put # 3,3,30
Close # 3
```

در این دستورات پس از باز شدن فایل به‌وسیله دستور Open، سه دستور Put به ترتیب هریک اعداد ۱۰، ۲۰ و ۳۰ را در فایل می‌نویسند. شماره رکورد در دستور Put اول عدد ۱ است زیرا عدد ۱۰ اولین داده‌ای است که در فایل ذخیره می‌شود و شماره رکورد برای عدد ۲۰، مقدار ۲ خواهد بود و به همین ترتیب الی آخر (شکل ۱-۲) و در پایان عملیات نیز دستور Close فایل را می‌بندند.

شکل ۱-۲

البته باید توجه داشت که دستورات Put را در این مثال می‌توان بدون شماره رکورد نیز استفاده کرد بنابراین دستورات Put را در این مثال می‌توان به این صورت نیز استفاده کرد:

```
Put # 3,,10
Put # 3,,20
Put # 3,,30
```

اگر در چند دستور Put از یک شماره رکورد ثابت استفاده شود اعداد روی هم ذخیره می‌شوند، به عبارت دیگر اعداد در یک موقعیت ثابت در فایل روی هم قرار گرفته و در نتیجه عدد قبلی در آن موقعیت رونویسی می‌شود. به عنوان مثال اگر سه دستور Put را در مثال قبل به این صورت تغییر دهید در فایل فقط عدد ۳۰ ذخیره می‌شود.

```
Put # 3,1,10

Put # 3,1,20

Put # 3,1,30
```

تمرین: یک پروژه با نام RandomNumbers بنویسید که ۱۰ عدد دو رقمی را به‌صورت تصادفی تولید کرده و در یک فایل با روش تصادفی ذخیره کند.

۳-۲ نحوه خواندن داده‌ها از فایل با روش دستیابی تصادفی

با استفاده از دستور Get می‌توان داده‌ها را از فایل‌ها خواند و مورد دسترسی قرار دارد، شکل کلی این دستور به این صورت است:

```
Get [#] filenumber , [recnumber] , Varname
```

عملکرد بخش‌های filenumber و recnumber مشابه همین قسمت‌ها در دستور Put می‌باشد و Varname نام متغیری است که داده‌های خوانده شده از فایل در آن ذخیره می‌شود. استفاده از کاراکتر # در این دستور اختیاری است.

نکته: در پایان عملیات خواندن داده و اطلاعات از فایل، آن را با استفاده از دستور# Close filenumber ببندید.

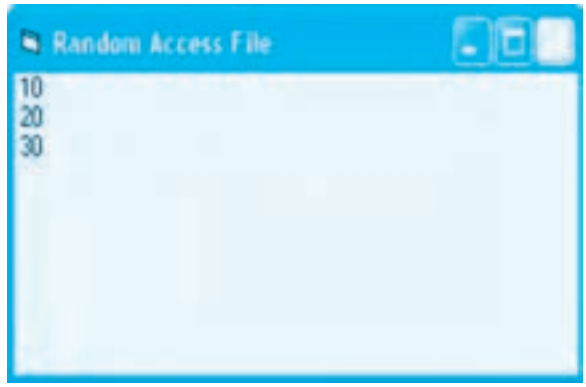
filenumber شماره دسترسی فایل است که در دستور Open مورد استفاده قرار گرفته است.

به عنوان مثال این دستورات فایل mydata.dat را با روش دسترسی تصادفی برای خواندن باز می‌کنند سپس داده‌ها را از آن می‌خوانند و روی فرم نمایش می‌دهند.

```
Open "D:\ mydata.dat" For Random Access Read As 3 Len=2

Dim intno As Integer

Get # 3,1, intno

Print intno

Get # 3,2, intno

Print intno
```

```
Get # 3,3, intno

Print intno

Close # 3
```

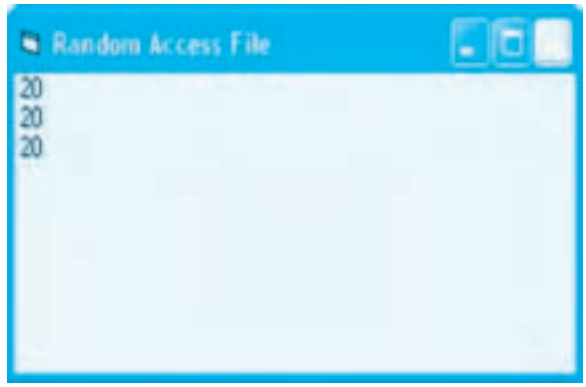
در این دستورات پس از باز شدن فایل برای عملیات خواندن داده‌ها یک متغیر از نوع صحیح تعریف می‌شود تا اعدادی که به‌وسیله متد Get از فایل خوانده می‌شوند در آن ذخیره شده و سپس نمایش داده شوند. سپس اولین متد Get اجرا می‌شود که رکورد شماره ۱ را از فایل mydata.dat می‌خواند و در متغیر intno ذخیره می‌کند و در مرحله بعد دستور Print مقدار متغیر intno را روی فرم نمایش می‌دهد و به همین شکل دستور Get دوم و سوم، دومین و سومین عدد را به ترتیب می‌خوانند و نمایش می‌دهند و در پایان متد Close فایل را می‌بندد. نتیجه اجرای این دستورات مانند شکل ۲-۲ می‌باشد.



شکل ۲-۲

در این مثال اگر دستورات Get را به این صورت تغییر دهید عدد ۲۰ سه بار خوانده شده و نمایش داده می‌شود (شکل ۳-۲).

```
Get # 3,2, intno

Print  intno

Get # 3,2, intno

Print intno

Get # 3,2, intno

Print intno
```



شکل ۳-۲

با توجه به مثال‌های مطرح شده می‌توان روش ذخیره‌سازی و بازیابی داده‌ها را به روش مستقیم یا تصادفی به وضوح مشاهده نمود. در این مثال‌ها به‌وسیله شماره رکورد و طول هر رکورد بدون توجه به سایر داده‌ها می‌توانید به هر داده به‌طور مستقل دسترسی پیدا کرده یا در هر مکان از فایل داده‌ای را ذخیره کنید.

تمرین: پروژه RandomNumber را به شکلی تنظیم کنید تا اعداد تصادفی ذخیره شده در فایل را بخواند و روی فرم نمایش دهد.

برای جلوگیری از تکرار دستور Get می‌توانید از حلقه‌هایی مانند For به این صورت استفاده کنید:

```
Open "D:\ mydata.dat" For Random Access Read As 3 Len=2
Dim intno As Integer
Dim inti As Integer
For inti=1 To 3
    Get # 3,inti, intno
    Print intno
Next inti
Close # 3
```

در این دستورات پس از باز شدن فایل برای خواندن، یک حلقه For سه بار دستور Get را اجرا می‌کند تا داده‌های ذخیره شده به ترتیب خوانده شوند و شماره رکورد نیز به‌وسیله شمارنده حلقه یعنی inti تعیین می‌شود که در هر بار اجرای حلقه For یک واحد افزایش می‌یابد.

مثال ۱: می‌خواهیم یک پروژه از نوع Standard EXE ایجاد کنیم تا تعدادی عدد اعشاری را از کاربر دریافت کرده و در یک فایل با روش دسترسی تصادفی بنویسد، به علاوه بتوان اعداد ذخیره شده را از فایل خوانده و مجموع آن‌ها را محاسبه نمود. برای انجام این کار عملیات بعد را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه یک فرم و کنترل‌های آن مانند شکل ۴-۲ ایجاد کنید.

شکل ۴-۲

۲- در بخش تعاریف ماژول دستور زیر را تایپ کنید. متغیر intrecno شماره رکوردی را که در فایل نوشته می‌شود، نگهداری می‌کند.

```
Dim  intrecno As Integer
```

۳- رویداد Click دکمه فرمان Save را به این صورت تنظیم کنید.

```
Private Sub cmdsave_Click()
    intrecno = intrecno + 1
    Open "D:\numbers.dat" For Random Access Write As 1 Len = 4
    Dim sngno As Single
    sngno = Val(txtno.Text)
    Put #1, intrecno, sngno
    Close #1
```

```
End Sub
```

در این رویداد ابتدا متغیر intrecno یک واحد افزایش می‌یابد تا در هر بار اجرای رویداد بتواند شماره رکورد جدید را محاسبه و برای استفاده در دستور Put نگهداری کند سپس متغیر sngno از نوع اعشاری تعریف می‌شود تا بتوان اعداد وارد شده در کادر متن را پس از تبدیل با تابع Val در آن ذخیره نمود. در مرحله بعد دستور Put با توجه به مقدار متغیر intrecno عدد اعشاری ذخیره شده در متغیر sngno را در فایل numbers.dat می‌نویسد.

در پایان نیز فایل با متد Close بسته می‌شود.

۴- رویداد Click دکمه فرمان Sum را به این صورت تنظیم کنید:

```
Private Sub cmdsum_Click()
   Open "D:\numbers.dat" For Random Access Read As 1 Len = 4
   Dim sngno As Single
   Dim sum As Single
   Do While (Not EOF(1))
      Get #1, , sngno
      sum = sum + sngno
   Loop
   txtsum.Text = sum
   Close #1
End Sub
```

در این رویداد پس از آنکه فایل numbers.dat با شماره دسترسی ۱ و طول رکورد ۴ بایت برای خواندن داده‌ها باز می‌شود، چون هر عدد اعشاری از نوع single،۴ بایت را در حافظه اشغال می‌کند. بنابراین طول رکورد در فایل ۴ بایت می‌باشد، سپس با یک حلقه Do While دستور Get هر بار یک عدد اعشاری را از فایل خوانده و در متغیر sngno ذخیره می‌کند و در مرحله بعد مقدار متغیر sngno به متغیر sum اضافه می‌شود تا مجموع اعداد خوانده شده محاسبه گردد. در این حلقه برای تشخیص رسیدن به انتهای فایل از تابع EOF استفاده شده است. در صورتی‌که هنگام اجرای دستور Get به انتهای فایل برسیم تابع EOF مقدار درست True را بازگشت می‌دهد و به دلیل اینکه عملکرد منطقی NOT مقدار True را به False تبدیل

می‌کند اجرای حلقه خاتمه می‌یابد. پس از خاتمه اجرای حلقه مجموع اعداد خوانده شده به‌وسیله متغیر sum در کنترل کادر متن txtsum نمایش داده می‌شود و در مرحله آخر نیز فایل به‌وسیله دستور Close () بسته می‌شود.

۵- پروژه و فرم را با نام Numbers ذخیره کنید سپس برنامه را اجرا کرده و چند عدد را با استفاده از کادرمتن txtno و دکمه فرمان Save در فایل ذخیره کنید.

۶- پس از خاتمه ذخیره‌سازی اعداد روی دکمه فرمان sum کلیک کنید تا مجموع اعداد در کنترل کادرمتن txtsum نمایش داده شود و نتیجه را بررسی نمایید. مجدداً چند عدد دیگر را وارد نموده و مجموع آن‌ها را به‌دست آورده و نتیجه را بررسی کنید.

۷- اجرای برنامه را خاتمه داده و به پنجره برنامه ویژوال‌بیسیک بازگردید.

مثال ۲: می‌خواهیم پروژه Numbers را به شکلی تغییر دهیم تا بتوانیم یکی از اعداد را با عدد جدیدی در فایل تعویض کنیم. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- پروژه Numbers را باز کنید و روی فرم آن کنترل دکمه فرمان مانند شکل ۵-۲ با نام cmdchange را اضافه کنید. این دکمه برای تغییر یک عدد با عدد جدید استفاده می‌شود.

شکل ۵-۲

۲- رویداد Click دکمه فرمان cmdchange را به این صورت تنظیم کنید:

```
Private Sub cmdchange_Click()
    Dim sngoldno As Single
```

```
    Dim sngnewno As Single
    Dim stroldno As String
    Dim strnewno As String
    stroldno = InputBox("Enter Your Number :", "Input Number")
    strnewno = InputBox("Enter New Number :", "Input Number")
    sngoldno = Val(stroldno)
    sngnewno = Val(strnewno)
    Open "D:\numbers.dat" For Random Access Read Write As 1 Len = 4
    Dim sngno As Single
    Dim intrecno As Integer
    intrecno = 1
    Do While (Not EOF(1))
        Get #1, , sngno
        If (sngno = sngoldno) Then
            Put #1, intrecno, sngnewno
            Exit Do
        End If
        intrecno = intrecno + 1
    Loop
    Close #1
End Sub
```

در این رویداد پس از تعریف متغیرهای مورد نیاز ابتدا با دو کادر ورود داده، عدد موردنظر و عدد جدیدی که جای آن در فایل قرار می‌گیرد دریافت شده و به ترتیب در متغیرهای رشته‌ای stroldno و strnewno ذخیره می‌شوند، سپس با استفاده از تابع Val این دو رشته عددی به اعداد اعشاری تبدیل و در متغیرهای sngoldno و sngnewno ذخیره می‌شوند. پس از اجرای این دستورات فایل numbers.dat با دستور Open برای عملیات

خواندن و نوشتن با یک شماره دسترسی باز می‌شود، در مرحله بعد متغیرهای sngno و intrecno تعریف می‌شوند. متغیر sngno برای نگهداری اعدادی که از فایل خوانده می‌شوند و متغیر intrecno برای مشخص کردن شماره رکوردی که خوانده شده است، استفاده می‌شوند. پس از این مرحله یک حلقه Do While به‌وسیله دستور Get اعداد را یکی‌یکی از فایل می‌خواند و در متغیر sngno ذخیره می‌کند. سپس یک دستور If عدد خوانده شده با عدد موردنظر کاربر که در متغیر sngoldno نگهداری می‌شود بررسی شده، در صورتی‌که نتیجه این بررسی درست (True) باشد یک دستور Put با توجه به شماره رکوردی که در متغیر intrecno ذخیره شده است، عدد جدید را روی عدد پیدا شده در فایل رونویسی می‌کند. بعد از اجرای این دستور، دستور Exit Do نیز اجرای برنامه را از حلقه خارج می‌کند زیرا دیگر نیاز به ادامه جستجوی عدد وجود ندارد، اما اگر نتیجه بررسی شرط در دستور If نادرست (False) باشد دستورات موجود در دستور If اجرا نمی‌شوند و یک واحد به مقدار شمارنده رکوردها یعنی متغیر intrecno اضافه می‌شود و با رسیدن به دستور Loop اجرای برنامه به حلقه Do While منتقل می‌گردد تا عدد بعدی خوانده شود. به این صورت اگر عدد مورد نظر در فایل پیدا شود با عدد جدید عوض می‌شود در غیر این صورت پس از خوانده شدن تمام اعداد از فایل و رسیدن به انتهای فایل تابع EOF مقدار False را بازگشت می‌دهد و اجرای حلقه خاتمه می‌یابد، بنابراین در هر دو صورت در خاتمه اجرای دستورات فایل با دستور Close بسته می‌شود.

۳- تغییرات را ذخیره کنید و برنامه را اجرا نمایید، سپس روی دکمه Change Number کلیک کنید و یک عدد را که در فایل موجود است، تایپ کرده و در کادر ورود داده دوم یک عدد جدید وارد کنید. در این مرحله عدد دوم به جای عدد اول در فایل قرار می‌گیرد. با کلیک روی دکمه Sum نتیجه را بررسی کنید.

۴- برنامه را با چند عدد دیگر نیز آزمایش نمایید و در پایان به اجرای برنامه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

۲-۴ نحوه ذخیره‌سازی و بازیابی داده‌ها به‌صورت رکورد

گاهی اوقات لازم است داده‌ها با انواع مختلفی به‌طور همزمان در فایل ذخیره شوند در این حالت خواندن و نوشتن هریک از داده‌ها به‌طور جداگانه کار دشوار و زمان‌بری خواهد بود. به‌علاوه در هنگام خواندن یا نوشتن داده‌ها باید کاملاً دقت شود تا نوع و ترتیب داده در هنگام خواندن آن‌ها دقیقاً مطابق با نوع و ترتیب داده در هنگام نوشتن آن‌ها در فایل باشد در غیر این‌صورت مشکلات جدی در کار با داده‌ها به‌وجود می‌آید. برای حل این مشکل می‌توانید از واحدهای اطلاعاتی با طول ثابت که به آن‌ها رکورد (Record) می‌گویند، استفاده کنید.

یک رکورد خود شامل دو یا چند واحد اطلاعاتی کوچک‌تر است که به هریک از آن‌ها یک فیلد (Field) گفته می‌شود. هر فیلد می‌تواند یک ویژگی و مشخصه از مجموعه ویژگی‌های یک شئ را توصیف کند، در واقع

فیلدها عملکردی شبیه به متغیرها دارند.

به عنوان مثال اگر مشخصات تحصیلی یک دانش‌آموز را در نظر بگیرید داده‌هایی مانند نام، نام‌خانوادگی، شماره شناسنامه، رشته تحصیلی، تاریخ تولد، کد دانش‌آموزی و رشته تحصیلی مورد استفاده قرار می‌گیرند. به هریک از این داده‌ها فیلد می‌گویند و مجموع این فیلدها تشکیل یک رکورد را می‌دهند که برای دانشجو نیز یک رکورد مورد استفاده قرار می‌گیرد.

به عنوان مثال دیگر می‌توان به مشخصات شناسنامه‌ای هر فرد در کشور اشاره کرد. برای نگهداری این مشخصات به چند فیلد مختلف احتیاج دارید و هر فرد نیز به یک رکورد که شامل تمام این فیلدها می‌شود احتیاج خواهد داشت. اطلاعاتی که در هر رکورد و فیلدهای آن ذخیره می‌شود برای هر فرد مجزا از سایر رکوردها خواهد بود، دقیقاً مانند شناسنامه‌های هر فرد که قالب و شکل نگهداری اطلاعات در همه آن‌ها یکی است ولی هریک نام و نام‌خانوادگی و شماره شناسنامه متفاوتی را نگهداری می‌کنند. در یک فایل با دسترسی تصادفی رکوردها به‌صورت پشت سر هم روی دیسک ذخیره می‌شوند. شکل ۶-۲ نحوه ذخیره‌سازی اطلاعات در یک فایل با دسترسی تصادفی با رکوردی به طول ۵۸ بایت را نشان می‌دهد.

شکل ۶-۲

۲-۴-۱ نحوه تعریف یک نوع داده جدید

برای ایجاد یک نوع داده جدید از ترکیب انواع داده‌های معرفی شده در زبان برنامه‌نویسی ویژوال‌بیسیک از دستور Type استفاده می‌شود. شکل کلی نحوه استفاده از این دستور به این صورت است:

```
نام نوع داده جدید Type
    نوع داده  As  نام داده اول
    نوع داده  As نام داده دوم
```

. . . . .

. . . . .

. . . . .

. . . . .

End Type

به عنوان مثال برای ایجاد یک نوع داده جدید که به‌وسیله آن بتوان مشخصات یک دانش‌آموز را ذخیره کرد، از دستور Type به این صورت استفاده می‌شود:

```
Type information
    firstname As String * 20
    lastname As String * 30
    studentid As Long
    b_date As Date
End Type
```

در این دستورات با استفاده از دستور Type یک نوع داده جدید به نام information تعریف شده است که خود شامل چهار متغیر یا به عبارت دیگر چهار فیلد می‌باشد. فیلدهای firstname و lastname از نوع رشته‌ای، فیلد studentid از نوع عدد صحیح بلند و فیلد b_date از نوع تاریخ برای تاریخ تولد در نظر گرفته شده‌اند. در واقع اکنون می‌توان با استفاده از نوع داده جدید information یک رکورد برای نگهداری اطلاعات یک دانش‌آموز تعریف کرد.

برای تعریف یک متغیر از نوع داده جدید یا به عبارت بهتر یک رکورد از نوع داده information از همان شیوه معرفی شده برای سایر انواع داده‌ها استفاده کنید. به عنوان مثال این دستور، یک متغیر از نوع داده information به نام student ایجاد می‌کند.

```
Dim student As information
```

برای دسترسی به اعضای این متغیر که از نوع داده جدید information تعریف شده است می‌توانید به این صورت عمل کنید:

```
student.firstname = "Ali"
student.lastname = "Rezaee"
```

student.studentid = 8821437

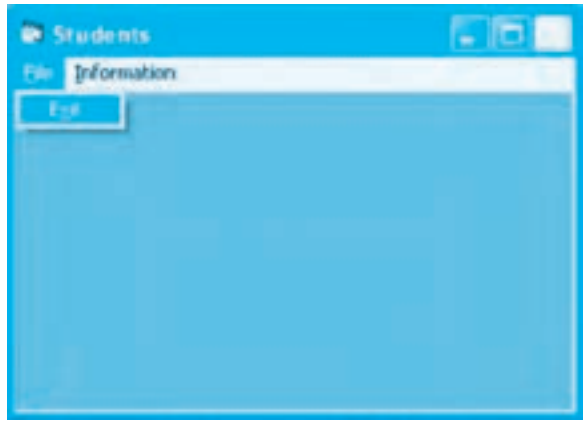
student.b_date = "11.241995"

در این دستورات رکورد student تمام اطلاعات مورد نیاز برای یک دانش‌آموز را در خود نگهداری کرده است.

نکته: با استفاده از کلمات کلیدی Private و Public قبل از دستور Type می‌توانید نوع داده جدید را به‌صورت محلی یا عمومی تعریف کنید.

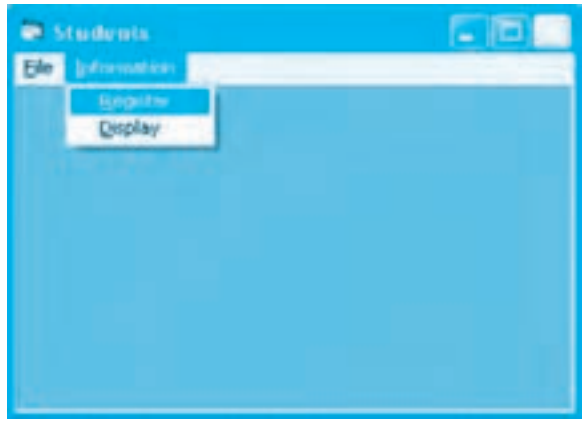
نکته: برای تعریف نوع داده جدید با استفاده از دستور Type می‌توانید از بخش تعاریف در ماژول فرم یا ماژول کد استفاده کنید.

مثال ۳: می‌خواهیم یک پروژه از نوع Standard EXE مانند شکل ۷-۲ طراحی کنیم که به‌وسیله آن بتوان اطلاعات دانش‌آموزان یک مدرسه را در یک فایل با روش دستیابی تصادفی ذخیره نمود. برای این کار عملیات بعد را به ترتیب انجام دهید:



شکل ۷-۲

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه یک فرم با نام frmmain و از نوع MDI مانند شکل ۷-۲ ایجاد کنید، سپس یک کنترل منوی اصلی همراه با گزینه‌های آن مانند شکل‌های ۷-۲ و ۸-۲ به فرم اضافه کنید.



شکل ۸-۲

۲- یک ماژول کد به پروژه اضافه کنید و نوع داده جدیدی با نام information به این صورت در آن تعریف کنید:

```
PublicType information
    firstname As String * 20
    lastname As String * 30
     stdentid As Long
     b_date As Date
End Type
```

شکل ۹-۲

۳- یک فرم جدید با نام frmregister به پروژه اضافه کنید و کنترل‌های آن را مانند شکل ۹-۲ روی فرم قرار دهید، از این فرم برای اضافه کردن اطلاعات دانش‌آموزان استفاده می‌شود.

۴- در بخش تعاریف ماژول فرم frmregister این دستور را تایپ کنید تا یک متغیر از نوع داده جدید information تعریف شود:

```
Dim student As information
```

۵- رویداد Load فرم frmregister را به این صورت تنظیم کنید:

```
Private Sub Form_Load()
    Open "D:\ students.dat" For Random Access Write As # 1 Len=62
End sub
```

در این رویداد فایل students.dat با شماره دسترسی ۱ برای دریافت داده‌ها با طول رکورد ۶۲ بایت و روش دسترسی تصادفی باز می‌شود. برای محاسبه طول رکورد ۲۰ بایت برای فیلد firstname، ۳۰ بایت برای فیلد lastname، ۴ بایت برای فیلد studentid و ۸ بایت برای فیلد b_date در نظر گرفته شده است.

۶- رویداد Click دکمه فرمان Save را نیز به این صورت تنظیم کنید:

```
Private Sub cmdsave_Click()
  student.firstname = txtfirstname.Text
  student.lastname = txtlastname.Text
  student.studentid = Val(txtid.Text)
  student.b_date = CDate(txtbdate.Text)
    Put #1, , student
End Sub
```

در این رویداد ابتدا داده‌های وارد شده در کنترل‌های کادر متن در فیلدهای متناظر خود در متغیر student ذخیره می‌شوند. با توجه به اینکه فیلد studentid از نوع عدد صحیح بلند (Long) و فیلد b_date از نوع تاریخ می‌باشد، با استفاده از توابع Val و CDate محتویات کادرهای متن مربوط به این دو فیلد ابتدا به نوع داده عددی و تاریخ تبدیل شده سپس در فیلدهای مربوطه ذخیره می‌شوند و در مرحله بعد با استفاده از دستور Put متغیر student به صورت یک رکورد کامل در فایل students ذخیره می‌گردد. به همین ترتیب در هر بار کلیک روی دکمه فرمان Save یک رکورد جدید در فایل ذخیره می‌شود.

۷- رویداد Click دکمه فرمان Close را به این صورت تنظیم کنید:

```
Private Sub cmdclose_Click()
    Close # 1
    Unload Me
End Sub
```

۸- فرم frmregister را با نام register ذخیره کنید.

۹- به فرم frmmain بروید و رویداد Click گزینه Register از منوی Information را به صورت زیر تنظیم کنید تا با کلیک روی این گزینه فرم ورود و ثبت اطلاعات دانش‌آموزان نمایش داده شود:

```
Private Sub mnuregister_Click()
    frmregister.Show()
    mnudisplay.Enabled=False
End Sub
```

۱۰- برای بازیابی و خواندن داده‌های ذخیره شده در فایل لازم است یک فرم جدید با نام frmdisplay به پروژه اضافه کنید و کنترل‌های آن را مانند شکل ۱۰-۲ روی فرم قرار دهید.

شکل ۱۰-۲

۱۱- رویداد Load فرم frmread را به این صورت تنظیم کنید:

```
Private Sub Form_Load()
    Open "D:\ students.dat" For Random Access Write As # 1 Len=62
```

```
End Sub
```

در این رویداد فایل students.dat با شماره دسترسی ۱ برای خواندن داده‌ها با طول رکورد ۶۲ بایت و روش دسترسی تصادفی باز می‌شود.

۱۲- در بخش تعاریف ماژول فرم frmdisplay این دستور را تایپ کنید تا یک متغیر از نوع داده جدید information تعریف شود.

```
Dim student As information
```

۱۳- برای نمایش اطلاعات دانش‌آموزان روی فرم می‌توان از دکمه فرمان Display استفاده کرد. برای این کار رویداد Click این دکمه فرمان را به این صورت تنظیم کنید:

```
Private Sub cmddisplay_Click()
  Print
  Print
  Print Tab(5); "Student ID"; Tab(25); "Name"; Tab(50); "Family"; Tab(90); "Birth Date"
  Print Tab(3); String(165, "-")
  Do While (Not EOF(1))
    Get #1, , student
    If (Not EOF(1)) Then
      Print Tab(5); student.studentid; Tab(25); _
        student.firstname; Tab(50); student.lastname; Tab(90); student.b_date
    End If
  Loop
End Sub
```

در این رویداد ابتدا دستورات Print عناوین مربوط به اطلاعات را در محل مناسب روی فرم نمایش می‌دهند، برای این کار از تابع Tab استفاده شده است تا عنوان در موقعیت تعیین شده روی فرم نمایش داده شود، سپس با استفاده از یک حلقه Do While و متد Get یک رکورد از نوع داده information فایل خوانده شده و در متغیر student ذخیره می‌گردد. سپس با دستور Print هر یک از فیلدهای رکورد student به ترتیب در موقعیت مناسب روی فرم نمایش داده می‌شود. دستور Print در داخل حلقه در یک دستور If قرار داده

شده است.

بنابراین دستور Print تا زمانی اجرا می‌شود که رکوردی از فایل خوانده شود. با رسیدن به انتهای فایل اجرای دستور Get سبب می‌شود تابع EOF مقدار True را بازگشت دهد بنابراین بررسی شرط موجود در دستور If، نتیجه نادرست False خواهد داشت و دستور Print اجرا نمی‌شود.

۱۴- رویداد Click دکمه فرمان Close را به این صورت تنظیم کنید:

```
Private Sub cmdclose_Click()
    Close # 1
    Unload Me
    Frmmain_mnuregister Enabled=True
End Sub
```

۱۵- فرم frmdisplay را با نام display ذخیره کنید.

۱۶- به فرم frmmain بروید و رویداد Click گزینه Display را از منوی Information به این صورت تنظیم کنید:

```
Private Sub mnudisplay_Click()
    frmdisplay.Show()
    mnuregister.Enabled=False
Ens Sub
```

۱۷- به فرم مادر frmmain بروید و رویداد Click گزینه Exit از منوی File را به این صورت تنظیم کنید:

```
Private Sub mnuexit_Click()
    Unload Me
End Sub
```

۱۸- فرم frmmain و پروژه را با نام students ذخیره نمایید، سپس برنامه را اجرا کرده و با استفاده از گزینه Register اطلاعات چند دانش‌آموز را در فایل ذخیره کنید.

۱۹- پس از خاتمه ورود و ثبت داده‌ها، به‌وسیله گزینه Display از منوی Information داده‌ها را بازیابی کرده و روی فرم نمایش دهید.

۲۰- اجرای پروژه را خاتمه داده و به پنجره برنامه ویژوال‌بیسیک بازگردید.

نکته: در صورتی‌که بخواهید اطلاعات رکورد معینی را از فایل با دسترسی تصادفی بخوانید یا بنویسید شماره رکورد مورد نظر را در دستور Get و Put وارد کنید. به عنوان مثال دستور Get # 1,2 , students دومین رکورد را می‌خواند.

تمرین: پروژه students را به گونه‌ای تنظیم کنید تا بتوان اطلاعات هر دانشجوی دلخواهی را با وارد کردن شماره دانشجویی وی مشاهده نمود. به علاوه امکان ویرایش و تغییر اطلاعات یک دانشجو نیز وجود داشته باشد.

Learn in English

Using Random File Access

The File System Object model does not provide random file creation or access methods. If you need to create or read random files, this information will help you do so.

The bytes in random-access files form identical records, each containing one or more fields. A record with one field corresponds to any standard type, such as an integer or fixed-length string. A record with more than one field corresponds to a user-defined type. For example, the Worker Type defined below creates 19-byte records that consist of three fields:

```
Type Worker
  LastName As String * 10
  Title    As String * 7
  Rank     As String * 2
End Type
```

Declaring Variables

Before your application opens a file for random access, it should declare all variables required to handle data from the file. This includes user-defined types, which correspond to records in the file, as well as standard types for other variables that hold data related to processing a file

opened for random access.

Defining Record Types

Before opening a file for random access, define a type that corresponds to the records the file does or will contain. For example, an Employee Records file could declare a user-defined data type called Person as follows:

```
Type Person
  ID              As Integer
  MonthlySalary   As Currency
  LastReviewDate  As Long
  FirstName       As String * 15
  LastName        As String * 15
  Title           As String * 15
  ReviewComments  As String * 150
End Type
```

Declaring Field Variables in a Type Definition

Because all records in a random-access file must have the same length, it is often useful for string elements in a user-defined type to have a fixed length, as shown in the Person type declaration above, where, for instance, FirstName and LastName have a fixed length of 15 characters.

If the actual string contains fewer characters than the fixed length of the string element to which it is written, Visual Basic fills the trailing spaces in the record with blanks (character code 32). Also, if the string is longer than the field size, it is truncated. If you use variable-length strings, the total size of any record stored with Put or retrieved with Get must not exceed the record length specified in the Open statement's Len clause.

Opening Files for Random Access

To open a file for random access, use the following syntax for the Open statement:

```
Open pathname [For Random] As filenumber Len = reclength
```

Because Random is the default access type, the For Random keywords are optional.

The expression Len = reclength specifies the size of each record in bytes. Note that every string variable in Visual Basic stores a Unicode string and that you must specify the byte length of that Unicode string. If reclength is less than the actual length of the record written to the file, an error is generated. If reclength is greater than the actual length of the record, the record is written, although some disk space may be wasted.

You could use the following code to open a file:

```
Dim FileNum As Integer, RecLength As Long, Employee As Person

RecLength = LenB(Employee)
FileNum = FreeFile
Open «MYFILE.FIL» For Random As FileNum Len = RecLength
```

واژه‌نامه

دسترسی	Access
رابطه داشتن، برابر بودن، مشابه بودن	Correspond
تعریف	Definition
جزء، عنصر	Element
میدان	Field
به‌دست آوردن	Get
معنی، مقصود، تصور	Identical
حالت	Mode
اختیاری	Optional
ثبت کردن، قرار دادن	Put
تصادفی	Random
سابقه	Record
ثبت کردن	Register
نیاز داشتن، لازم داشتن	Require
نوع	Type
درست، صحیح	Valid

خلاصه مطالب

- روش‌های دسترسی به فایل‌ها به دو گروه روش دسترسی ترتیبی و روش دسترسی تصادفی تقسیم می‌شوند.
- خواندن و نوشتن داده‌ها در یک فایل با دسترسی تصادفی بر اساس رکورد با طول ثابت انجام می‌شود.
- به هر ویژگی و مشخصه از مجموعه ویژگی‌های یک شئ، شخص یا فیلد می‌گویند.
- مجموع چند فیلد تشکیل یک رکورد می‌دهند.
- از دستور Type برای تعریف یک نوع داده جدید که شامل چند فیلد است، استفاده می‌شود.

- از یک نوع داده جدید که به‌وسیله دستور Type تعریف می‌شود، می‌توان یک رکورد ایجاد نمود.
- با استفاده از دستور Open می‌توان یک فایل را با دسترسی تصادفی برای عملیات خواندن یا نوشتن داده‌ها باز کرد.
- برای نوشتن داده‌ها در فایل با دسترسی تصادفی از دستور Put استفاده می‌شود.
- برای خواندن داده‌ها از فایل با دسترسی تصادفی از دستور Get استفاده می‌شود.
- با استفاده از دستور Close می‌توان یک فایل با دسترسی تصادفی را در پایان عملیات خواندن و نوشتن داده‌ها بست.
- تابع EOF با رسیدن به انتهای فایل مقدار True و در غیر این‌صورت مقدار False را بازمی‌گرداند.

آزمون نظری

۱- کدام دستور امکان نوشتن داده‌ها در یک فایل با دسترسی تصادفی را فراهم می‌کند؟

الف- Put ب- Get ج- Open د- Write

۲- وظیفه آرگومان Len در دستور Open چیست؟

الف- تعیین شماره دسترسی فایل ب- تعیین شماره رکورد جاری

ج- تعیین شماره رکورد برای خواندن یا نوشتن داده‌ها د- تعیین اندازه فایل

۳- در صورت رسیدن به انتهای یک فایل با دسترسی تصادفی تابع EOF چه مقداری را بازمی‌گرداند؟

الف- True ب- False ج- یک رشته خالی د- رشته Null

۴- کدام دستور برای تعریف یک نوع داده جدید استفاده می‌شود؟

الف- Dim ب- Record ج- Type د- Field

۵- در کدام نوع از انواع فایل‌ها استفاده از روش دسترسی ترتیبی مناسب‌تر است؟

الف- فایل‌های باینری ب- فایل‌های با دسترسی تصادفی

ج- فایل‌های متنی د- فایل‌های داده

۶- کدام دستور برای خواندن داده‌ها از فایل با دسترسی تصادفی مناسب است؟

الف- Put ب- Get ج- Read د- ReadLine

۷- حداکثر مقدار filenumber در دستور Open ..... می‌باشد.

الف- ۱۲۸ ب- ۲۵۶ ج- ۵۱۲ د- ۱۰۲۴

8- Which of the following answer can define a eype of record for Read and Write information in the files?

a- Put b- Get c- Type d- Records

۹- تفاوت روش دسترسی ترتیبی و تصادفی را در فایل‌ها بیان کنید.

۱۰- نحوه باز کردن فایل‌ها با روش دسترسی تصادفی را توضیح دهید.

۱۱- نحوه نوشتن و خواندن داده‌ها و اطلاعات با روش دسترسی تصادفی را با ذکر مثال به‌طور کامل توضیح دهید.

۱۲- نحوه تعریف یک نوع داده جدید به‌وسیله دستور Type را با ذکر مثال توضیح دهید.

۱۳- نحوه تعریف یک رکورد از نوع داده جدید را که به‌وسیله دستور Type تعریف می‌شود، بیان کنید و روش نوشتن و خواندن یک رکورد را در فایل‌ها با دسترسی تصادفی توضیح دهید.

۱۴- روش خواندن و نوشتن یک رکورد خاص در فایل‌ها با دسترسی تصادفی را با ذکر مثال توضیح دهید.

آزمون عملی

۱- یک پروژه طراحی کنید که با استفاده از فایل با دسترسی تصادفی اطلاعات مربوط به کارمندان یک اداره را براساس جدول ۱-۲ دریافت کند و امکان انجام این عملیات را داشته باشد:

الف- ثبت اطلاعات یک کارمند

ب- ویرایش و اصلاح اطلاعات وارد شده

ج- جستجو و مشاهده اطلاعات یک کارمند

د- مشاهده اطلاعات تمام کارمندان

جدول ۱-۲ اطلاعات کارمندان

کدپرسنلی
نام
نام خانوادگی
تاریخ تولد
جنسیت
حقوق دریافتی
تاریخ استخدام

۲- پروژه‌ای طراحی کنید که بتواند اطلاعات و ورود و خروج هزینه توقف خودروها را در یک پارکینگ با استفاده از فایل با دسترسی تصادفی ذخیره، نگهداری و مدیریت نماید.

واحد کار سوم

هدف جزیی

توانایی برنامه‌نویسی به روش شئ گرا

زمان (ساعت)	
نظری	عملی
۱۰	۳۰

هدف‌های رفتاری

پس از مطالعه این واحد کار از فراگیر انتظار می‌رود که:

۱- ویژگی‌های برنامه‌نویسی به روش شئ‌گرا را بداند و تفاوت آن را با روش برنامه‌نویسی ساخت یافته بیان کند.

۲- مفاهیم کلاس، شیء و کپسوله کردن را بداند.

۳- توانایی ایجاد یک Class را داشته باشد.

۴- عضو فیلد و خاصیت را در یک کلاس ایجاد کند و تفاوت آن‌ها را بداند.

۵- عضو متد را در یک کلاس ایجاد کند و انواع آن‌ها را توضیح دهد.

۶- عضو رویداد را در یک کلاس ایجاد کند.

۷- رویدادهای کلاس را توضیح داده و کاربرد آن‌ها را بیان کند.

۸- کاربرد Class Builder و Object Browser را بداند و بتواند با آن‌ها کار کند.

۹- مفهوم کلکسیون را بداند و بتواند کلکسیونی از اشیا را ایجاد کرده و استفاده کند و متدها و خاصیت‌های آن‌ها را به کار گیرد.

۱۰- کاربرد دستور For Each را بداند و بتواند از آن استفاده کند.

۱۱- نحوه ساخت پروژه‌ها از نوع ActiveX DLL را بداند.

۱۲- بتواند پروژه‌ها از نوع ActiveX DLL Control را ایجاد کند.

کلیات

در این واحدکار یکی از روش‌های جدید برنامه‌نویسی به نام برنامه‌نویسی به روش شئ‌گرا یا Object Oriented Programming را که به اصطلاح به آن OOP می‌گویند فرامی‌گیرد. در روش‌های قدیمی‌تر برنامه‌نویسی مانند برنامه‌نویسی به روش ساخت یافته (Structural) برنامه به بخش‌های کوچک‌تر تقسیم می‌شود و این کار تا رسیدن به کوچک‌ترین اجزا ادامه می‌یابد، سپس با استفاده از مفاهیمی مانند توابع و رویه‌ها این اجزا طراحی و ساخته می‌شوند و با به کار گرفتن اجزای کوچک اجزای بزرگ‌تر و هدف‌های اصلی برنامه دنبال می‌شوند. یکی از بزرگ‌ترین مشکلات این روش این است که هنگام فراخوانی توابع و رویه‌های فرعی همواره لازم است داده‌های مورد نیاز از محل فراخوانی به داخل تابع یا رویه فرعی انتقال یابد، این مسأله هنگامی که داده‌ها زیاد باشند این کار را سخت و دشوار می‌کند. خصوصاً اگر لازم باشد این فراخوانی‌ها بیش از یک بار و به دفعات زیاد در طول برنامه انجام شود. در برنامه‌نویسی به روش شئ‌گرا از مفهوم شئ به همان معنی که در محیط پیرامون ما وجود دارد استفاده می‌شود. در این روش داده‌ها، توابع و رویه‌ها داخل یک بدنه واحد به نام شئ قرار می‌گیرد و وظایف موجود در برنامه به صورت بخش‌های جداگانه‌ای که هر یک با استفاده از یک یا مجموعه‌ای از اشیا تأمین می‌شوند، باعث آسان‌تر و سریع‌تر انجام شدن برنامه‌نویسی شده و این روش به سرعت جایگزین روش‌های قدیمی می‌شود.

۳-۱ مفاهیم بنیادی در برنامه‌نویسی شئ‌گرا

یکی از مهم‌ترین و اصلی‌ترین مفاهیم در روش برنامه‌نویسی شئ‌گرا، کلاس است به طور کلی یک کلاس قالب و الگویی است که می‌توان به وسیله آن یک نمونه واقعی به نام شئ را به وجود آورد. به عبارت دیگر یک کلاس، قالب و الگویی است که می‌تواند شکل ظاهری و عملکرد یک شئ را مشخص کند.

به عنوان مثال فرض کنید که می‌خواهیم در یک کارخانه شیرینی‌پزی، کیک و کلوچه تولید کرده و به بازار عرضه کنیم، به این منظور می‌توانیم از روش‌های مختلفی استفاده کنیم. در روش اول می‌توان پس از آماده کردن خمیر کیک، هر کارگر با جدا کردن یک تکه از خمیر یک کلوچه را شکل داده و آماده پختن کند که این روش نه تنها برای فرم و شکل‌دهی به هر کیک به زمان زیادی نیاز دارد بلکه سبب می‌شود هیچ یک از کیک‌های تولید شده از نظر وزن و شکل ظاهر شبیه به هم نباشد. در روش دیگر می‌توان یک قالب فلزی یا پلاستیکی از کیک موردنظر برای هر کارگر به وجود آورد و تمام ویژگی‌های کیک از نظر وزن و شکل ظاهری را در آن ایجاد کرد. سپس کارگر با استفاده از این قالب، کیک‌ها را بدون زحمت و صرف زمان زیادی شکل دهد. در این حالت کلیه کیک‌هایی که توسط تمام کارگران به وجود می‌آید دارای وزن، اندازه و شکل ظاهری مشابه خواهند شد، حتی می‌توان کارگران را از سیستم تولید حذف و از دستگاه‌های خودکار استفاده

کرد. در این روش قالب فلزی یا پلاستیکی کیک در واقع خود کیک نیست اما ویژگی‌های کیک تولید شده را توصیف می‌کند. این قالب همان کلاس است و به خمیری که داخل قالب کیک قرار می‌گیرد و پس از پخته شدن قابل خوردن است، شئ می‌گویند. اگر به محیط اطرافتان توجه کنید می‌بینید که برای تولید اشیا از این روش استفاده می‌شود.

به عنوان مثالی دیگر وقتی شما در محیط ویژوال‌بیسیک یک کنترل کادر متن را از جعبه ابزار انتخاب می‌کنید و آن را روی فرم قرار می‌دهید، ویژوال‌بیسیک به طور خودکار از یک قالب آماده که از قبل تعریف و ایجاد شده است یک کنترل جدید و واقعی ایجاد می‌کند و روی فرم قرار می‌دهد. به قالب و الگوی کنترل، کلاس و به نمونه‌ای که از آن ساخته شده و روی فرم قرار می‌گیرد شئ می‌گویند. بنابراین یک کلاس را می‌توان چنین تعریف کرد: «به مجموعه‌ای از داده‌ها و دستورالعمل‌ها که می‌توان به وسیله آن یک شئ را به صورت کامل ایجاد نمود کلاس می‌گویند.» برای نگهداری داده‌ها از مفهوم فیلد و خاصیت و برای تعریف و اجرای دستورالعمل‌ها از مفهوم متد و رویداد استفاده می‌شود، به عبارت دیگر یک کلاس مجموعه‌ای از فیلدها، خاصیت‌ها، متدها و رویدادها می‌باشد که فیلدها و خاصیت‌ها وظیفه نگهداری داده‌ها و متدها و رویدادها وظیفه اجرای دستورالعمل‌ها را به عهده دارند. به هر یک از اجزای تشکیل‌دهنده یک کلاس مانند یک فیلد، خاصیت، متد یا رویداد، یک عضو (Member) کلاس می‌گویند.

۳-۲ مفهوم کپسوله کردن (Encapsulation)

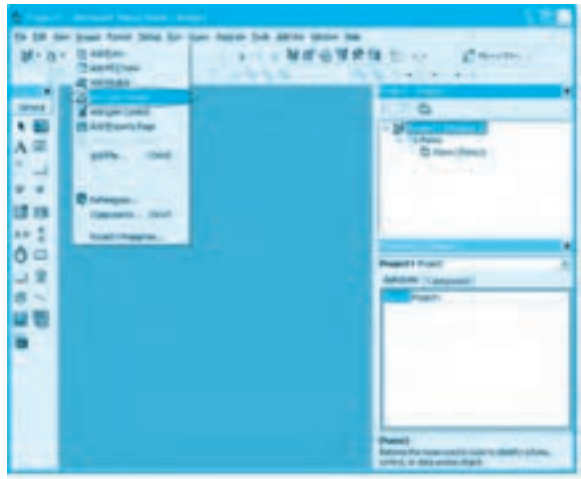
کپسوله کردن یعنی گردآوری و ایجاد مجموعه‌ای از اعضا در یک بخش واحد به نام کلاس، به شکلی که وقتی نمونه‌ای از کلاس (شئ) ایجاد می‌شود بتوان به راحتی از اعضا استفاده نمود، بدون آن‌که لازم باشد از جزییاتی که در رابطه با آن عضو وجود دارد اطلاع و آگاهی داشت. به این عمل مخفی کردن اعضا نیز می‌گویند. به عنوان مثال وقتی شما خاصیت Width را در یک فرم یا کنترل مقداردهی می‌کنید لازم نیست در رابطه با چگونگی تنظیم عرض فرم اطلاعاتی داشته باشید، بلکه فقط باید مقدار موردنظر خود را در خاصیت مربوطه ذخیره کنید یا وقتی که با استفاده از متد SetFocus، فوکوس به یک کنترل انتقال داده می‌شود لازم نیست در رابطه با چگونگی عملکرد این متد اطلاعاتی داشته باشید بلکه متد را فراخوانی کرده و از امکانات آن استفاده کنید.

۳-۳ نحوه ایجاد یک کلاس

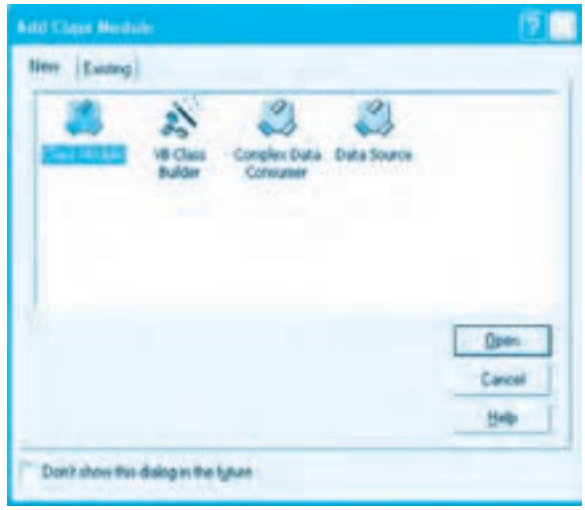
برای ایجاد یک کلاس از ماژول کلاس (Class Module) استفاده می‌شود. ماژول کلاس شبیه به ماژول کد است و شکل ظاهری مانند فرم‌ها ندارد برای اضافه کردن یک ماژول کلاس به برنامه می‌توانید یکی از این روش‌ها را استفاده کنید:

۱- در پنجره ویژوال‌بیسیک روی منوی Project کلیک کنید و سپس روی گزینه Add Class Module کلیک

نمایید (شکل ۱-۳)، سپس در کادرمحاوره Add Class Module روی زبانه New کلیک کنید و بعد آیکن Class Module را انتخاب کنید (شکل ۲-۳).

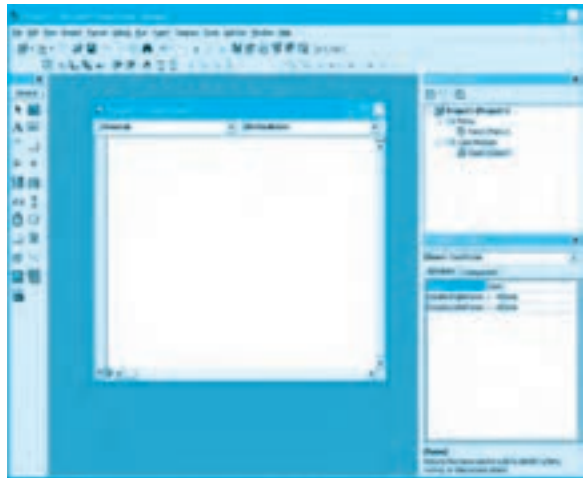


شکل ۱-۳



شکل ۲-۳

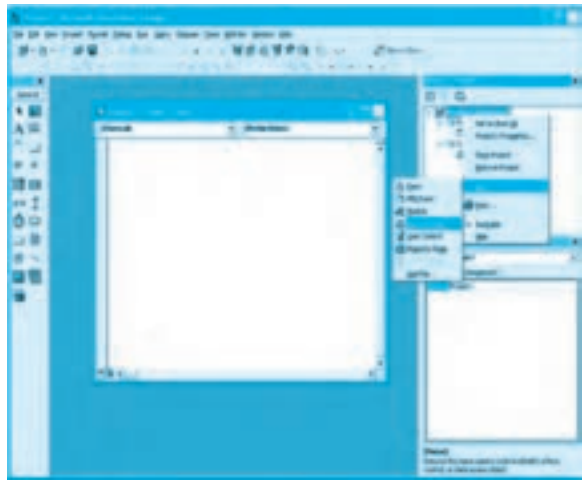
در پایان روی دکمه فرمان Open کلیک کنید در این مرحله یک پنجره شبیه به پنجره ماژول کد در پنجره برنامه ویژوال‌بیسیک باز می‌شود.



شکل ۳-۳

به طور پیش‌فرض برای نام ماژول کلاس از نام Class1 استفاده می‌شود که در پنجره مرورگر پروژه و در نوار عنوان پنجره ماژول کلاس قابل مشاهده است (شکل ۳-۳). ماژول کلاس روی دیسک با نام ماژول کلاس و پسوند cls به صورت یک فایل مستقل روی دیسک ذخیره می‌شود، به عنوان مثال برای یک ماژول کلاس به نام Class1 نام فایل آن Class1.cls می‌باشد.

۲- در پنجره مرورگر پروژه کلیک راست کنید. در منویی که نمایش داده می‌شود اشاره‌گر ماوس را روی گزینه Add نگه دارید سپس در زیرمنویی که باز می‌شود روی گزینه Class Module کلیک کنید (شکل ۴-۳).



شکل ۴-۳

پس از مشاهده کادرمحاوره Add Class Module می‌توانید مانند روش اول عمل نمایید و یک ماژول کلاس را به پروژه اضافه کنید.

۱-۳-۳ نحوه ایجاد عضو فیلد

فیلد یکی از اعضای تشکیل‌دهنده کلاس‌هاست که داده‌ها و اطلاعات را می‌توان در آن ذخیره کرد. در واقع فیلدها در کلاس‌ها عملکردی شبیه به متغیرها دارند، بنابراین از همان روش‌هایی که برای تعریف متغیرها به کار می‌رود برای تعریف فیلدها می‌توان استفاده کرد.

فیلدها را می‌توان با دستور Dim یا Private به صورت خصوصی تعریف نمود، در این حالت از فیلد فقط در همان ماژول کلاس می‌توان استفاده کرد اما اگر فیلد با دستور Public تعریف شود هم در ماژول کلاسی که در آن تعریف شده است و هم در سایر بخش‌های برنامه می‌توان از آن استفاده نمود.

مثال ۱: می‌خواهیم یک پروژه طراحی کنیم که با استفاده از یک کلاس بتوان اطلاعات مربوط به یک کارمند را در یک مؤسسه ذخیره، نگهداری و بازیابی نمود. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE ایجاد کنید.

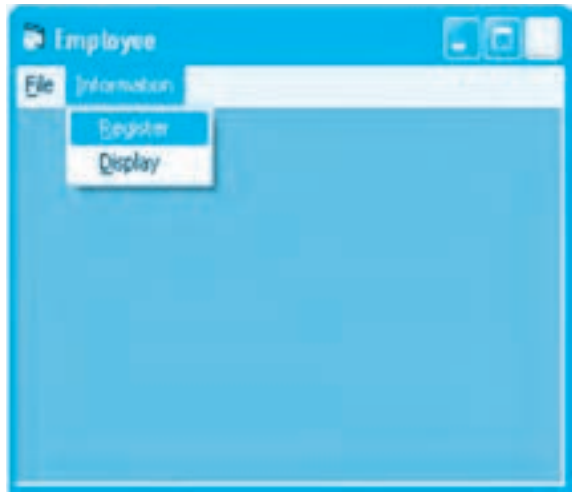
۲- یک ماژول کلاس به پروژه اضافه کنید، سپس در پنجره خواص خاصیت Name آن را روی مقدار clsemployee تنظیم کنید.

۳- در این مرحله می‌خواهیم اعضایی در کلاس تعریف کنیم تا بتوانند داده‌ها و اطلاعات یک کارمند را دریافت و نگهداری کنند. برای این کار می‌توان از عضو فیلد استفاده کرد برای این کار در پنجره ماژول کلاس کلیک کنید و این دستورات را در بخش تعاریف آن تایپ کنید:

```
Public firstname As String

Public lastname As String

Public employeeid As Long

Public b_date As Date
```

در این دستورات متغیرهای firstname برای نام، lastname برای نام خانوادگی، studentid برای کد دانش‌آموز و b_date برای تاریخ تولد به عنوان فیلد در کلاس clsemplayee به صورت عمومی تعریف شده‌اند. ماژول کلاس را با نام clsemployee ذخیره نمایید.

۴- به پنجره فرم پروژه بروید و خاصیت نام آن را به frmmain تغییر دهید، سپس فرم را به صورت یک فرم MDI. تنظیم کنید و مانند شکل ۵-۳ یک کنترل منو روی فرم قرار داده و گزینه‌های آن را تنظیم کنید. سپس آن را با نام main ذخیره کنید.



شکل ۵-۳

۵- در این مرحله لازم است تا از کلاس clsemployee یک شئ ایجاد شود تا در تمام برنامه بتوان از آن استفاده نمود. به دلیل این‌که از خود کلاس به طور مستقیم نمی‌توان استفاده کرد بنابراین ابتدا یک ماژول کد به پروژه اضافه کرده و این دستورات را در آن تایپ کنید تا یک شئ از کلاس clsemployee به صورت عمومی معرفی شود تا در تمام برنامه قابل شناسایی باشد، البته باید توجه داشته باشید که این دستور شئ را ایجاد نمی‌کند بلکه فقط فضایی در حافظه برای آن اختصاص می‌دهد که توانایی دریافت و نگهداری شئ مشابه کلاس clsemployee را داشته باشد.

```
Public employee As clsemployee
```

شکل ۶-۳

ماژول کد را با نام mdlobject ذخیره کنید.

۶- در این مرحله لازم است برای ذخیره‌سازی داده در شئ employee از یک فرم استفاده کنید، بنابراین به پنجره مرورگر پروژه رفته و یک فرم جدید مانند شکل ۶-۳ به پروژه اضافه کنید و نام آن را روی frmregister تنظیم نمایید.

۷- در این مرحله برای ذخیره‌سازی هر یک از داده‌های مربوط به یک کارمند دلخواه، لازم است ابتدا یک شئ از کلاس clsemployee ایجاد کرده و به هر یک از فیلدهای آن به صورت مستقل دسترسی پیدا کنید.

برای ایجاد شئ از یک کلاس از دستور Set به این صورت می‌توان استفاده کرد:

Set نام شئ = New نام کلاس

بنابراین برای ایجاد شئ از کلاس clsemployee می‌توانید این دستور را اجرا کنید:

```
Set employee = New clsemployee
```

به علاوه برای دسترسی به هر یک از اعضای یک شئ می‌توان به این صورت عمل نمود:

نام شئ . نام عضو

پس برای دسترسی به فیلد firstname از شئ employee و مقداردهی به آن می‌توان به این صورت عمل کرد:

```
employee.firstname = txtfirstname.Text
```

اکنون رویداد Click دکمه فرمان Save را می‌توانید به این صورت تنظیم کنید:

```
Private Sub cmdsave_Click()
    Set mdlobject.employee = New clsemployee
    mdlobject.employee.firstname = txtfirstname.Text
    mdlobject.employee.lastname = txtlastname.Text
    mdlobject.employee.employeeid = Val(txtid.Text)
    mdlobject.employee.b_date = CDate(txtbdate.Text)
End Sub
```

در این رویداد ابتدا با استفاده از دستور Set شئ employee از کلاس clsemployee ایجاد می‌شود و چون شئ employee در ماژول کد mdlobject معرفی شده است، لازم است قبل از نام شئ نام ماژول کد ذکر شود سپس اطلاعات مربوط به کارمند از طریق کنترل‌های کادر متن در فیلدهای متناظر با آن‌ها ذخیره می‌شود.

۸- فرم frmregister را با نام register ذخیره کنید.

نکته: وقتی در پنجره کد نام شئ و پس از آن کاراکتر نقطه را تایپ می‌کنید به طور خودکار لیستی از اعضای شئ، مشابه اعضایی که در کلاس به صورت عمومی تعریف شده‌اند نمایش داده می‌شود (شکل ۷-۳).

شکل ۷-۳

۹- برای نمایش اطلاعات ذخیره شده کارمند مربوطه یک فرم جدید دیگر با نام frmdisplay مانند شکل ۸-۳ به پروژه اضافه کنید.

شکل ۸-۳

سپس رویداد Click دکمه فرمان Display را در آن به این صورت تنظیم کنید:

```
Private Sub cmddisplay_Click()
  txtfirstname.Text = mdlobject.employee.firstname
  txtlastname.Text = mdlobject.employee.lastname
  txtid.Text = Str(mdlobject.employee.employeeid)
  txtbdate.Text = Str(mdlobject.employee.b_date)
End Sub
```

در این رویداد نیز با استفاده از نام شئ به هر یک از اعضای آن دسترسی پیدا کرده و محتویات فیلد مربوطه را در کنترل کادر متن متناظر با آن قرار داده می‌شود.

این فرم را نیز با نام display روی دیسک ذخیره کنید.

۱۰- به فرم frmmain رفته و رویداد Click گزینه‌های Register و Display را به این صورت تنظیم کنید:

```
Private Sub mnuregister_Click()
  frmregister.Show
End Sub

Private Sub mnudisplay_Click()
  frmdisplay.Show
End Sub
```

۱۱- پروژه را با نام employee ذخیره نمایید سپس آن را اجرا کنید.

۱۲- اطلاعات یک کارمند دلخواه را ذخیره نمایید و سپس نمایش داده و مشاهده و بررسی کنید.

۱۳- از برنامه خارج شده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: یک پروژه با نام library از نوع MDI طراحی کنید که به وسیله آن بتوان اطلاعات یک کتاب را با استفاده از یک کلاس ذخیره و بازیابی نمود (اطلاعات کتاب می‌تواند شامل: کد کتاب، عنوان کتاب، موضوع کتاب، مؤلف، ناشر و بهای کتاب باشد).

۲-۳-۳ نحوه ایجاد عضو خاصیت

خاصیت نیز مانند فیلد وظیفه نگهداری داده و اطلاعات را دارد به دلیل این‌که فیلدها امکان بررسی و مدیریت داده‌هایی را هنگام دریافت و ذخیره‌سازی ندارند مشکلاتی را به وجود می‌آورند. به عنوان نمونه در مثال ۱ اگر شما یک عدد منفی را به عنوان کد کارمندی در فیلد employeeid ذخیره کنید بدون هیچ اشکالی این

مقدار پذیرفته می‌شود و برای جلوگیری از چنین مشکلی باید دستورات شرطی مناسبی در خارج از کلاس کارمندان مورد استفاده قرار گیرد. این نوع راه‌حل‌ها برخلاف قوانین برنامه‌نویسی به روش شئ‌گرا می‌باشد عضو خاصیت، امکاناتی را در اختیار برنامه‌نویس قرار می‌دهد که سبب می‌شود اجرای هرگونه عملیات مربوط به ذخیره‌سازی و بازیابی داده‌ها داخل کلاس و شیئی که از آن به وجود می‌آید انجام شود.

برای ایجاد عضو خاصیت می‌توانید به این ترتیب عمل کنید:

۱- تعریف یک فیلد (متغیر) خصوصی داخل کلاس تا به وسیله آن بتوان داده موردنظر را ذخیره و نگهداری کرد به دلیل خصوصی بودن این فیلد، امکان دسترسی به آن از بیرون کلاس و شئ وجود ندارد.

۲- برای آن‌که بتوان پس از تعریف شئ از کلاس، خاصیت موردنظر را مقداردهی نمود از یک رویه به نام Property Let استفاده می‌شود این رویه به طور خودکار در زمان مقداردهی به خاصیت اجرا می‌شود. شکل کلی این رویه به این صورت است:

Public Property Let نام خاصیت (ByVal نام آرگومان As نوع داده)

دستورات

End Property

این رویه هنگام مقداردهی به خاصیت شئ به طور خودکار اجرا شده و مقدار داده در آرگومان رویه کپی می‌شود، بنابراین می‌توان داخل رویه Let هرگونه عملیاتی که لازم است روی مقدار ارسالی انجام داد و در متغیر خصوصی که در مرحله ۱ تعریف شده است داخل شئ ذخیره کرد.

۳- برای آن‌که بتوان مقدار یک خاصیت را مورد دسترسی قرار داد و خواند از یک رویه به نام Property Get استفاده می‌شود. شکل کلی این رویه به این صورت است:

Public Property Get نام خاصیت ( ) As نوع داده

دستورات

End Property

این رویه نیز هنگام خواندن خاصیت شئ به طور خودکار اجرا می‌شود، بنابراین می‌توان داخل رویه Get مقدار ذخیره شده در فیلد خصوصی را به دست آورد و به محل موردنظر بازگشت داد.

به عنوان مثال برای تعریف خاصیت کد کارمندی می‌توان به این صورت عمل کرد:

```
Private lngid As Long

Public Property Let employeeid(ByVal id As Long)

    lngid = id
```

```
End Property

Public Property Get employeeid() As Long

    employeeid = lngid

End Property
```

در این دستورات ابتدا یک فیلد محلی به نام lngid از نوع Long تعریف می‌شود این فیلد محلی برای ذخیره و نگهداری مقدار خاصیت employeeid استفاده می‌شود و چون با دستور Private تعریف می‌شود فقط داخل کلاس و شئ قابل دستیابی می‌باشد و نوع داده آن هم با توجه به این‌که خاصیت employeeid از نوع Long است از این نوع انتخاب شده است.

در مرحله دوم رویه Property Let تعریف شده است نام این رویه هم نام خاصیت employeeid می‌باشد و شامل یک آرگومان است آرگومان id مقداری را که به خاصیت نسبت داده می‌شود دریافت می‌کند تا داخل رویه Let بتوان آن عملیات مورد نیاز را انجام داد نوع این آرگومان نیز از نوع خاصیت employeeid است.

داخل رویه Let نیز مقدار آرگومان id در فیلد خصوصی lngid ذخیره می‌شود با این روش داده موردنظر داخل شئ نگهداری خواهد شد.

در مرحله سوم رویه Property Get تعریف شده است نام این رویه نیز هم نام خاصیت employeeid می‌باشد.

در این رویه لازم است نوع داده‌ای که بازگشت داده می‌شود مشخص شود چون فیلد خصوصی lngid، از نوع Long می‌باشد، بنابراین نوع داده بازگشتی در رویه Get نیز از نوع Long خواهد بود. داخل این رویه نیز یک دستور وجود دارد که مانند توابع مقدار فیلد خصوصی lngid به محل فراخوانی یعنی جایی که خاصیت خوانده می‌شود بازگردانده خواهد شد.

بنابراین وقتی یک شئ به نام employee از کلاس clsemployee تعریف شود این دستور رویه Property Let را اجرا می‌کند:

```
employee.employeeid = 8821345
```

در این حالت عدد ۸۸۲۱۳۴۵ به رویه Property Let ارسال شده و در آرگومان id قرار می‌گیرد سپس داخل رویه مقدار آرگومان id در فیلد خصوصی lngid ذخیره می‌شود. حال اگر بخواهیم مقدار خاصیت employeeid را خوانده و نمایش دهیم این دستور را اجرا کرده تا رویه Property Get اجرا شود.

```
txtid.Text = Str(employee.employeeid)
```

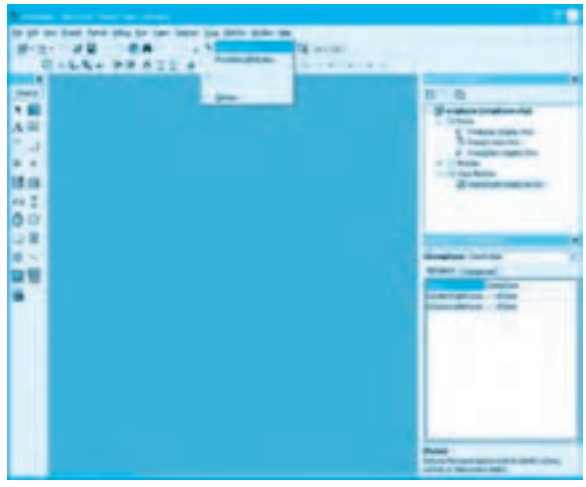
در این حالت رویه Get اجرا شده و دستور employeeid=lngid مقدار ذخیره شده در فیلد خصوصی lngid را به محل فراخوانی بازگشت می‌دهد تا در کادر متن txtid نمایش داده شود.

اکنون می‌توان رویه Property Let را به شکلی تنظیم کرد تا خاصیت employeeid از دریافت اعداد کوچک‌تر یا مساوی صفر خودداری کند. برای این کار می‌توان رویه Property Let را به این صورت تنظیم کرد:

```
Public Property Let employeeid(ByVal id As Long)
  If (id > 0) Then
    lngid = id
  Else
    lngid = 0
    MsgBox "Invalid Your EmployeeID!", vbOKOnly + vbCritical, "Error"
  End If
End Property
```

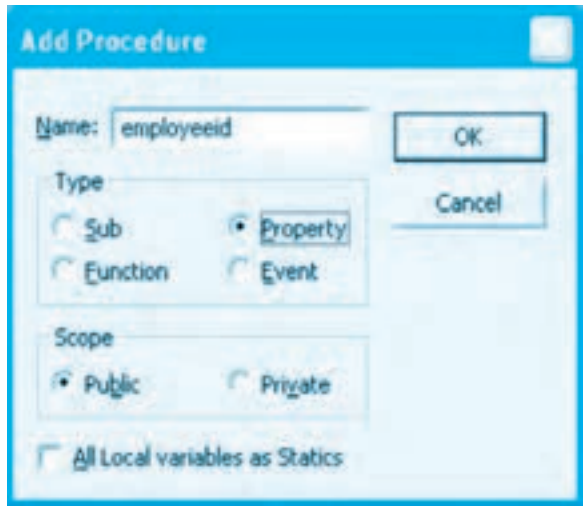
در این رویه با استفاده از دستور شرطی If مقدار آرگومان id بررسی می‌شود. در صورتی که مقدار آن بزرگ‌تر از صفر باشد در فیلد خصوصی lngid ذخیره می‌شود، در غیر این صورت مقدار صفر در فیلد lngid ذخیره شده و یک پیام خطا نیز نمایش داده می‌شود. بنابراین وقتی یک شئ از کلاس clsemployee ایجاد می‌شود لازم نیست نگران مقداردهی به خاصیت employeeid باشید چون به هیچ روشی نمی‌توان یک مقدار نادرست را در این خاصیت ذخیره نمود.

برای ایجاد عضو خاصیت در یک کلاس می‌توانید از گزینه Add Procedure در منوی Tools نوار منوی ویژوال‌بیسیک استفاده کنید (شکل ۹-۳).



شکل ۹-۳

پس از کلیک روی این گزینه کادرمحاوره Add Procedure مانند شکل ۱۰-۳ نمایش داده می‌شود.



شکل ۱۰-۳

در این کادرمحاوره و کادر متن موجود در آن نام خاصیت را تایپ کنید، سپس روی دکمه انتخاب Public Property کلیک کنید تا انتخاب شود و در پایان روی دکمه Add کلیک کنید. در این مرحله یک رویه Property Let و Property Get مانند شکل ۱۱-۳ به کلاس اضافه می‌شوند، در این روش نوع داده بازگشتی در رویه Property Get و نوع داده برای آرگومان رویه Property Let از نوع Variant استفاده می‌شود. اکنون می‌توانید نوع خاصیت و آرگومان آن را به دلخواه تنظیم کنید در ضمن لازم است یک فیلد محلی مناسب نیز برای خاصیت خود در کلاس تعریف نمایید.

شکل ۱۱-۳

مثال ۲: می‌خواهیم کلیه اعضای کلاس clsemployee را با استفاده از رویه‌های Property Let و Property Get تعریف کنیم. برای انجام این کار عملیات زیر را به ترتیب انجام دهید:

۱- پروژه employee را باز کنید.

۲- به ماژول کلاس clsemployee بروید و دستورات آن را به این صورت تغییر دهید:

```
Private strfname As String
Private strlname As String
Private lngid As Long
Private dtmbdate As Date
Public Property Let employeeid(ByVal id As Long)
   If (id > 0) Then
      lngid = id
   Else
      lngid = 0
      MsgBox "Invalid Your EmployeeID!", vbOKOnly + vbCritical, "Error"
   End If
End Property
Public Property Get employeeid() As Long
   employeeid = lngid
End Property

Public Property Let firstname(ByVal fname As String)
   strfname = fname
End Property
Public Property Get firstname() As String
```

```
  firstname = strfname
End Property

Public Property Let lastname(ByVal lname As String)
   strlname = lname
End Property
Public Property Get lastname() As String
  lastname = strlname
End Property

Public Property Let b_date(ByVal bdate As Date)
   dtmbdate = bdate
End Property

Public Property Get b_date() As Date
  b_date = dtmbdate
End Property
```

۳- تغییرات را ذخیره کرده و پروژه را اجرا و آزمایش کنید.
۴- اجرای برنامه را خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: اعضای فیلد در کلاس پروژه Library را به عضو خاصیت تبدیل کنید. سپس برنامه را اجرا کرده و نتیجه را با حالت قبل مقایسه کنید.

۳-۳-۳ نحوه ایجاد عضو متد

متدها اعضایی از یک کلاس هستند که وظیفه اجرای دستورالعمل‌ها را به عهده دارند و نمونه شئ که از هر کلاس ایجاد می‌شود یک عضو متد را به همان شکل که سایر نمونه‌ها از آن کلاس اجرا می‌کنند، مورد

استفاده قرار می‌دهد.

مانند متد Show در فرم‌ها، دستوراتی که در متد Show یک فرم وجود دارند و اجرا می‌شوند دقیقاً به همان صورتی است که در سایر فرم وجود داشته و اجرا می‌شوند و هیچ تفاوتی بین متد Show در دو فرم دیده نمی‌شود.

روش تعریف یک متد در کلاس بسیار ساده است با توجه به این‌که متد موردنظر مقداری را به محل فراخوانی بازگشت می‌دهد یا خیر می‌توان آن را به صورت یک رویه تابعی یا رویه فرعی تعریف نمود.

بنابراین بهتر است اگر متد مقداری را بازگشت می‌دهد آن را به صورت یک تابع و در غیر این صورت به صورت یک رویه فرعی تعریف کرد.

مثال ۳: می‌خواهیم یک پروژه طراحی کنیم که با استفاده از یک کلاس بتوان مساحت یک مربع دلخواه را محاسبه نمود. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE ایجاد کنید.

۲- یک ماژول کلاس به پروژه اضافه کنید، سپس در پنجره خواص خاصیت Name آن را روی مقدار clssquare تنظیم کنید.

۳- یک عضو خاصیت به نام Length برای نگهداری طول ضلع مربع در کلاس clssquare به این صورت ایجاد کنید:

```
Private snglength As Single

Public Property Let Length(ByVal l As Single)
   If (l > 0) Then
      snglength = l
   Else
      snglength = 0
      MsgBox "Invalid Your Length!", vbOKOnly + vbCritical, "Error"
   End If
End Property


Public Property Get Length() As Single
```

```
    Length= snglength
End Property
```

۴- برای محاسبه مساحت مربع با توجه به دستورالعملی که مساحت مربع را محاسبه می‌کند می‌توان یک عضو متد را تعریف کرد. برای انجام این کار در نوار منوی ویژوال‌بیسیک روی منوی Tools و سپس گزینه Add Procedure کلیک کنید و در کادرمحاوره Add Procedure ابتدا در کادر متن Name نام متد (Area) را تایپ کنید، سپس در کنترل قاب Type دکمه انتخاب Function را انتخاب نمایید و بعد در کنترل قاب Scope دکمه انتخاب Public را انتخاب کنید.

در پایان روی دکمه فرمان OK کلیک کنید تا متد به صورت یک رویه تابعی به کلاس اضافه شود (شکل ۱۲-۳).

شکل ۱۲-۳

۵- متد Area را به این صورت تنظیم کنید:

```
Public Function Area() As Single
    Area = snglength * snglength
End Function
```

در این متد مساحت مربع از حاصلضرب یک ضلع در خودش محاسبه شده و بازگشت داده می‌شود.

۶- کلاس clssquare را با نام square ذخیره کنید.

۷- به پنجره طراحی فرم بروید خاصیت و نام آن را روی frmsquare و عنوان آن را روی عبارت Area تنظیم کنید، سپس مانند شکل ۱۳-۳ کنترل‌های برچسب، متن و دکمه فرمان را روی فرم قرار دهید. سپس

خاصیت Enable دومین کادر متن را روی مقدار false تنظیم کنید.

شکل ۱۳-۳

۸- رویداد Click دکمه فرمان Area را به این صورت تنظیم کنید:

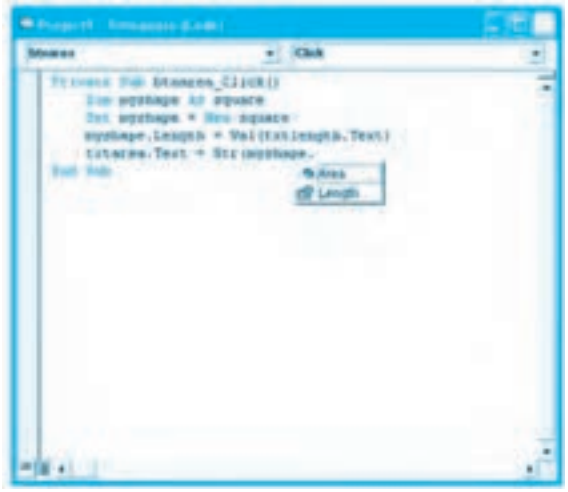
```
Private Sub cmdarea_Click()
   Dim myshape As  clssquare
   Set myshape = New  clssquare
   myshape.Length = Val(txtlength.Text)
   txtarea.Text = Str(myshape.Area)
End Sub
```

در این رویداد ابتدا با دستورات Dim و Set یک نمونه شئ از کلاس clssquare با نام myshape ایجاد می‌شود، سپس خاصیت Length این شئ مقداردهی شده و پس از آن با فراخوانی متد Area مساحت مستطیل موردنظر محاسبه و نتیجه در کادر متن مربوطه نمایش داده می‌شود.

نکته: پس از تایپ نام شئ myshape خاصیت‌ها و متدهای تعریف شده در کلاس مربوطه در یک لیست بازشو نمایش داده می‌شوند (شکل ۱۴-۳).

۹- پروژه و فرم را با نام Square ذخیره و سپس آن را اجرا کنید.

۱۰- طول و عرض چند مستطیل دلخواه را وارد کرده و روی دکمه Area کلیک نمایید و نتایج به دست آمده را بررسی کنید.



شکل ۱۴-۳

۱۱- از برنامه خارج شده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: یک متد به کلاس مربع اضافه کنید تا به وسیله آن بتوان محیط مربع را نیز محاسبه کرد.

البته باید توجه داشت که متدها را می‌توان با توجه به کاربرد آن‌ها به صورت‌های مختلفی طراحی نمود مثلاً در مثال ۳ کلاس مربع را می‌توان به این صورت نیز طراحی کرد:

```
Public Function Area(length As Single) As Single
    Area = length * length
End Function
```

در این حالت خاصیت length حذف شده و فقط متد Area همراه با یک آرگومان که طول ضلع مربع را دریافت می‌کند در کلاس تعریف می‌شود. بنابراین اگر بخواهیم با استفاده از این کلاس جدید مساحت یک مربع دلخواه را محاسبه کنیم لازم است رویداد Click دکمه فرمان Area به این صورت تنظیم شود:

```
Private Sub cmdarea_Click()
    Dim myshape As clssquare
    Set myshape = New clssquare
    txtarea.Text = Str(myshape.Area(Val(txtlength.Text)))
End Sub
```

این دستورات از کلاس جدید مربع یک شئ ایجاد و مساحت آن را محاسبه می‌کند با این حال نتیجه محاسبات با حالت قبل یکسان خواهد بود.

مثال ۴: می‌خواهیم کلاس مربع را به گونه‌ای تنظیم کنیم تا مساحت مربع را محاسبه کرده و نمایش دهد.

برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه square را باز کنید.

۲- کلاس clssquare را به این صورت تغییر دهید:

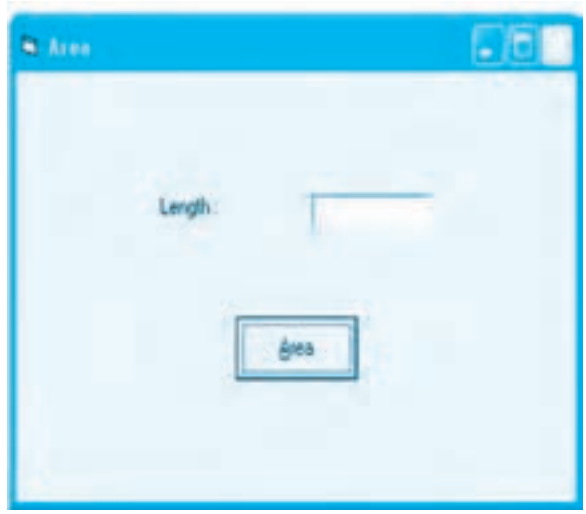
```
Public Sub Area(length As Single)
    Dim sngarea As Single
    sngarea = length * length
    MsgBox ("Area I Equal :" + Str(sngarea))
End Sub
```

در این دستورات کلاس clssquare با یک عضو متد و متد Area نیز به صورت یک رویه فرعی تعریف شده‌اند متد Area طول ضلع مربع را به صورت یک آرگومان از نوع Single دریافت می‌کند و پس از محاسبه مقدار مساحت، نتیجه را در یک کادر پیغام نمایش می‌دهد.

در این حالت برای ایجاد شئ و استفاده از متد Area می‌توان این دستورات را در رویداد Click دکمه فرمان Area قرار داد.

```
Private Sub cmdarea_Click()
    Dim myshape As clssquare
    Set myshape = New clssquare
    myshape.Area (Val(txtlength.Text))
End Sub
```

۳- فرم frmsquare را نیز در پنجره طراحی باز کرده و آن را مانند شکل ۱۵-۳ تنظیم کنید.



شکل ۱۵-۳

۴- پروژه، فرم و کلاس مربع را با نام جدید و در یک پوشه مستقل ذخیره نمایید، سپس پروژه را اجرا کرده و نتیجه را بررسی و با حالت قبل مقایسه کنید.

۵- از برنامه خارج شده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: در مثال ۴ یک رویه فرعی در کلاس مربع تعریف کنید تا محیط مربع را محاسبه کرده و در یک کادر پیغام نمایش دهد.

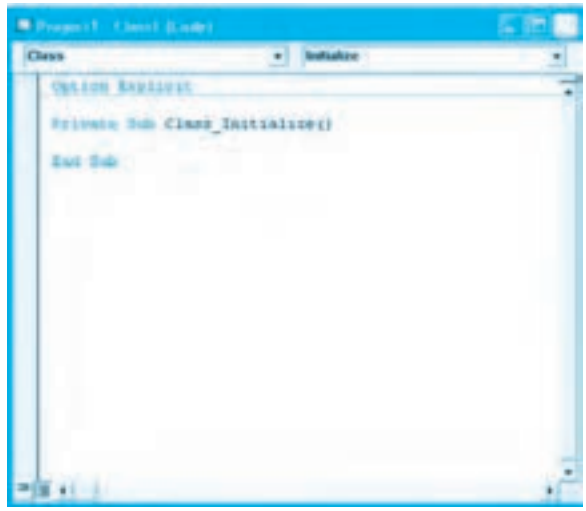
۴-۳-۳ رویدادهای کلاس

به طور کلی در هر کلاس دو رویداد وجود دارد که عبارتند از: رویداد Initialize و رویداد Terminate.
با استفاده از این رویدادها می‌توان مجموعه‌ای از دستورات را در زمان مناسب به طور خودکار اجرا کرد.

رویداد Initialize

این رویداد زمانی اجرا می‌شود که یک نمونه شئ از کلاس به وجود می‌آید.
یکی از کاربردهای مهم این رویداد تنظیم مقدار اولیه فیلدها و خاصیت‌های شئ در زمان ایجاد شئ از کلاس می‌باشد. برای استفاده از این رویداد می‌توانید پس از ایجاد ماژول کلاس، در پنجره‌ای که ماژول کلاس نمایش داده می‌شود کادر لیست ترکیبی بازشو سمت چپ را باز کرده و گزینه Class را انتخاب کنید در این مرحله در کادر لیست ترکیبی بازشو سمت راست به طور خودکار گزینه Initialize انتخاب

می‌شود (شکل ۱۶-۳).



شکل ۱۶-۳

مثال ۵: می‌خواهیم کلاس clssquare در پروژه square را به شکلی تنظیم کنیم تا در زمان ایجاد شئ از کلاس خاصیت length مربع به طور دلخواه روی مقدار ۵ تنظیم شود. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه square را باز کنید.

۲- ماژول کلاس clssquare را باز کرده و در پنجره کدنویسی آن کادر لیست ترکیبی بازشو سمت چپ را باز کنید. سپس گزینه Class را انتخاب کنید در این مرحله رویداد Initialize به ماژول کلاس اضافه می‌شود. رویداد را به این صورت تنظیم کنید:

```
Private Sub Class_Initialize()
    snglength = 5
End Sub
```

در این رویداد فیلد محلی snglength روی مقدار ۵ تنظیم شده است، بنابراین وقتی یک شئ از کلاس clssquare ایجاد می‌شود رویداد اجرا شده و این فیلد به طور خودکار روی مقدار تعیین شده تنظیم می‌شود.

۳- برای آن‌که نحوه عملکرد رویداد Initialize بهتر مشاهده شود رویداد Click دکمه فرمان Area را در فرم به این صورت تنظیم کنید:

```
Private Sub cmdarea_Click()
    Dim myshape As clssquare
```

```
    Set myshape = New clssquare
    txtarea.Text = Str(myshape.Area())
End Sub
```

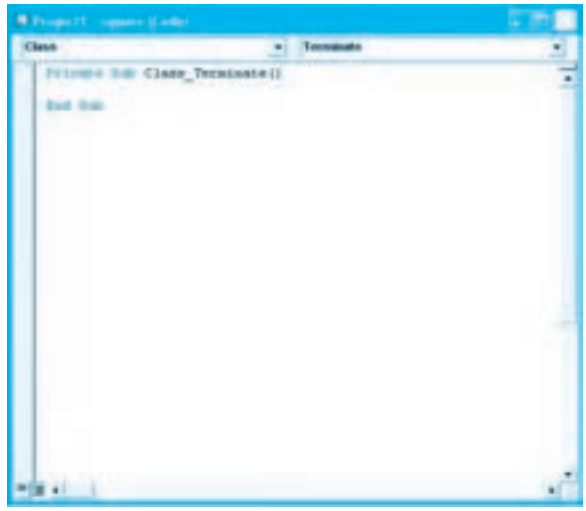
در این رویداد بدون آن‌که خاصیت Length مقداردهی شود مقدار مساحت محاسبه می‌شود. در این حالت پس از اجرای دستور دوم رویداد و ایجاد شئ، رویداد Initialize نیز به طور خودکار اجرا شده و مقدار خاصیت Length را روی ۵ تنظیم می‌کند، بنابراین متد Area نیز مقدار ۲۵ را محاسبه کرده و بازگشت می‌دهد.

۴- تغییرات را ذخیره کرده، پروژه را اجرا نمایید و پس از کلیک روی دکمه فرمان Area نتیجه را مشاهده و بررسی کنید.

۵- به اجرای برنامه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

رویداد Terminate

این رویداد زمانی اجرا می‌شود که شئ ایجاد شده از حافظه خارج می‌شود زمان از بین رفتن شئ به نحوه تعریف، میدان و حوزه‌ای که برای آن تعیین شده است بستگی دارد. به عنوان مثال وقتی یک شئ داخل یک رویه رویداد تعریف و ایجاد می‌شود، با پایان اجرای رویداد و رسیدن به دستور End Sub شئ نیز از بین می‌رود. این رویداد محل مناسبی برای اجرای دستوراتی است که می‌خواهید در زمان از بین رفتن یک شئ اجرا شوند. برای استفاده از این رویداد، پس از ایجاد ماژول کلاس در پنجره‌ای که ماژول کلاس نمایش داده می‌شود می‌توانید کادر لیست ترکیبی بازشو سمت چپ را باز کرده و گزینه Class را انتخاب کنید. سپس کادر لیست ترکیبی بازشو سمت راست را باز کرده و رویداد Terminate را انتخاب کنید (شکل ۱۷-۳).



شکل ۱۷-۳

مثال ۶: می‌خواهیم کلاس clssquare در پروژه square را به شکلی تنظیم کنیم تا در زمانی که محاسبات پایان یافته و شئ از بین می‌رود یک پیام به کاربر نشان داده شود. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه square را باز کنید.

۲- ماژول کلاس clssquare را باز کرده و رویداد Terminate آن را به این صورت تنظیم کنید:

```
Private Sub Class_Terminate()
    MsgBox "Calculation Is Terminated"
End Sub
```

در این رویداد یک پیام با استفاده از تابع MsgBox در زمان از بین رفتن شئ به طور خودکار نمایش داده می‌شود.

۳- تغییرات را ذخیره کنید، سپس برنامه را اجرا کرده و روی دکمه فرمان Area کلیک کنید. یک کادر پیغام بلافاصله پس از نمایش مقدار مساحت مربع به طور خودکار نمایش داده می‌شود.

۴- از برنامه خارج شده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: یک پروژه از نوع Standard EXE با نام rectangle طراحی کنید که با استفاده از یک کلاس بتواند محیط و مساحت هر مستطیل دلخواهی را محاسبه کند. کلاس ایجاد شده باید این شرایط را داشته باشد:

الف- برای ذخیره و نگهداری طول و عرض مستطیل از عضو خاصیت استفاده شود.

ب- امکان ذخیره‌سازی مقادیر عددی کوچک‌تر از صفر در خاصیت‌های طول و عرض وجود نداشته باشد.

ج- برای محاسبه مساحت و محیط از عضو متد استفاده شود.

د- هنگام ایجاد شئ از کلاس، خاصیت‌های طول و عرض مستطیل به طور خودکار روی مقادیر مشخصی تنظیم شوند.

هـ- هنگام ازبین‌رفتن شئ که از کلاس مستطیل ایجاد می‌شود پیام مناسبی به کاربر نمایش داده شود.

۳-۳-۵ نحوه ایجاد عضو رویداد

رویدادها اعضایی از یک کلاس هستند که دستورات خود را هنگام رخ دادن یک اتفاق به‌طور خودکار اجرا می‌کنند. برخلاف متدها دستورات یک رویداد برای شئ که از یک کلاس به وجود می‌آیند می‌تواند با سایر اشیا که از همان کلاس ایجاد شده‌اند متفاوت باشد مانند رویداد Click در دکمه‌های فرمان. دستوراتی که در رویداد Click یک دکمه فرمان نوشته می‌شود زمانی اجرا می‌شوند که عمل کلیک روی دکمه فرمان انجام شود. اما رویداد Click هر دکمه فرمان می‌تواند دستورات متفاوتی با رویداد Click سایر دکمه‌های فرمان داشته باشد.

روش تعریف و استفاده از رویدادها کمی پیچیده‌تر از متدها است.

برای ایجاد یک عضو رویداد در ماژول کلاس می‌توان به ترتیب زیر عمل کرد:

۱- **تعریف رویداد:** لازم است رویداد نیز مانند سایر اعضا در ماژول کلاس تعریف شود. برای این کار می‌توان

از دستور Event استفاده کرد به عنوان مثال این دستور یک رویداد را با نام Warning تعریف می‌کند:

```
Public Event Warning()
```

۲- فراخوانی رویداد: لازم است رویداد را پس از تعریف در یک یا چند موقعیت مناسب فراخوانی نمود. برای این کار می‌توان از دستور RaiseEvent داخل یک یا چند متد استفاده کرد. به عنوان مثال این دستور، رویداد Warning را که قبلاً تعریف شده است فراخوانی می‌کند.

```
RaiseEvent Warning
```

۳- تعریف شئ همراه با رویدادهای آن: برای آن‌که بتوان از رویدادهای یک شئ استفاده کرد لازم است شئ همراه با رویدادهایش تعریف شود. برای این‌کار از کلمه کلید WithEvents در زمان تعریف شئ از کلاس استفاده می‌شود. به عنوان مثال این دستور یک شئ myshape را از کلاس square با رویدادهای آن تعریف می‌کند.

```
Dim WithEvents myshape As clssquare
```

مثال ۷: می‌خواهیم پروژه square را به شکلی تنظیم کنیم تا وقتی مقدار مساحت مربع کمتر از ۱۰ واحد شود یک رویداد به‌طور خودکار اجرا شده و عملیات تعیین شده انجام شود. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه square را باز کنید.

۲- ماژول کلاس clssquare را باز کرده و در بخش تعاریف آن رویداد را به این صورت تعریف کنید:

```
Public Event Warning()
```

۳- متد مساحت را به صورت زیر تنظیم کنید و تغییرات را ذخیره کنید.

```
Public Function Area() As Single
    Area = snglength * snglength
    If (snglength * snglength < 10) Then RaiseEvent Warning
End Function
```

در این متد علاوه بر دستورات قبلی از یک دستور شرطی If نیز استفاده شده است تا مقدار مساحت را بررسی کند اگر این مقدار کوچک‌تر از ۱۰ باشد رویداد Warning، با استفاده از دستور RaiseEvent فراخوانی می‌شود.

۴- در این مرحله می‌توان از عضو رویداد ایجاد شده در کلاس clssquare برای اشیایی که به‌وجود می‌آیند استفاده کرد. برای این کار این دستور را در بخش تعاریف ماژول فرم تایپ کنید.

```
Dim WithEvents myshape As clssquare
```

این دستور شئ myshape را به وسیله کلمه کلیدی WithEvents در دستور Dim همراه با رویداد آن معرفی می‌کند.

نکته: در رویه‌های رویداد نمی‌توان اشیا را همراه با رویدادهایشان تعریف کرد.

۵- رویداد Click دکمه فرمان Area را به این صورت تنظیم کنید:

```
Private Sub cmdarea_Click()
   Set myshape = New clssquare
   myshape.Length = Val(txtlength.Text)
   txtarea.Text = Str(myshape.Area())
End Sub
```

۶- در ماژول فرم نام شئ (myshape) ایجاد شده را از کادر لیست ترکیبی بازشو سمت چپ و از کادر لیست ترکیبی بازشو سمت راست نام رویداد (Warning) را انتخاب کنید در این مرحله یک رویداد جدید با نام myshape_warning در ماژول فرم قابل مشاهده و استفاده می‌باشد (شکل ۱۸-۳).

شکل ۱۸-۳

۷- رویداد اضافه شده را به این صورت تنظیم کنید:

```
Private Sub myshape_warning()
   MsgBox "Area < 10 "
End Sub
```

در صورتی که این رویداد اجرا شود یک کادر پیغام عبارت Area < 10 را نمایش می‌دهد.

۸- تغییرات را ذخیره کرده و برنامه را اجرا کنید. برنامه را یک‌بار با مربعی به طول ۲ و بار دیگر با مربعی به

طول ۸ آزمایش کنید و نتایج حاصل را بررسی و مقایسه کنید.

سپس برنامه را برای مربعی با طول ۲ آزمایش کنید در این صورت رویداد Warning در متد Area فراخوانی می‌شود و در نتیجه کادر پیغام پیام تعیین شده را نمایش می‌دهد. اما برای مربع با طول ۸ چون مقدار مساحت از ۱۰ واحد بیشتر می‌شود، رویداد Warning فراخوانی نمی‌شود.

۹- از برنامه خارج شده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: کلاس clssquare را به شکلی تنظیم کنید تا اگر مقدار محیط آن بزرگ‌تر از ۲۰ واحد شود با استفاده از یک رویداد بتوان پیام مناسبی را نمایش داد.

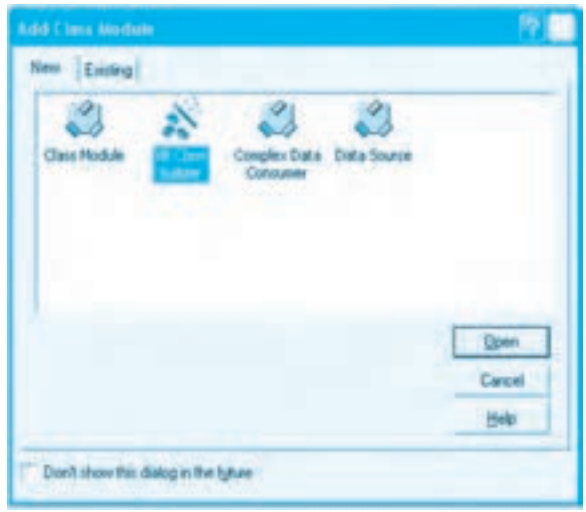
تمرین: در پروژه rectangle کلاس مستطیل را به شکلی تنظیم کنید تا اگر مقدار مساحت آن بین ۲۰ تا ۵۰ واحد باشد با استفاده از یک رویداد بتوان پیام مناسبی را نمایش داد.

۳-۳-۶ ابزار Class Builder

در ویژوال‌بیسیک روش ساده‌تر و سریع‌تری نیز برای ایجاد کلاس وجود دارد. با استفاده از Class Builder می‌توان عملیات تعریف اعضایی مانند فیلد، خاصیت، متد و رویداد را انجام داد. برای استفاده از این ابزار به این ترتیب عمل کنید:

۱- در نوار منوی ویژوال‌بیسیک روی منوی Project و سپس گزینه Add Class Module کلیک کنید تا کادرمحاوره مربوطه نمایش داده شود.

۲- در کادرمحاوره Add Class Module، آیکن VB Class Builder را انتخاب کنید و بعد روی دکمه فرمان Open کلیک کنید (شکل ۱۹-۳).



شکل ۱۹-۳

۳- در این مرحله پنجره Class Builder مانند شکل ۲۰-۳ نمایش داده می‌شود.

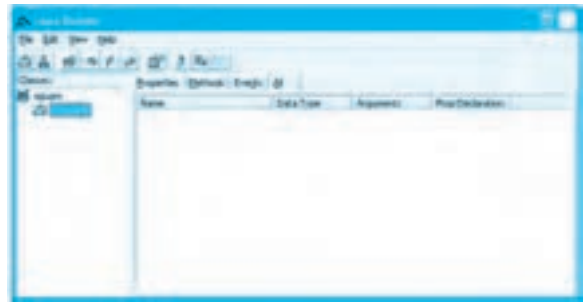
شکل ۲۰-۳

۴- در پنجره Class Builder بجز نوار منو و نوار ابزار دو بخش مجزا در زیر نوار ابزار پنجره وجود دارند. بخش سمت چپ نام کلاس‌های موجود در پروژه را نمایش می‌دهد و بخش سمت راست اعضای تشکیل‌دهنده کلاسی را که در بخش سمت چپ انتخاب شده است نشان می‌دهد.

نحوه ایجاد یک کلاس جدید با استفاده از Class Builder

برای ایجاد یک کلاس جدید در پروژه می‌توانید به این ترتیب عمل کنید:

۱- در نوار منوی پنجره Class Builder روی منوی File کلیک کنید و سپس روی گزینه‌های New و ... Class کلیک کنید (شکل ۲۱-۳).

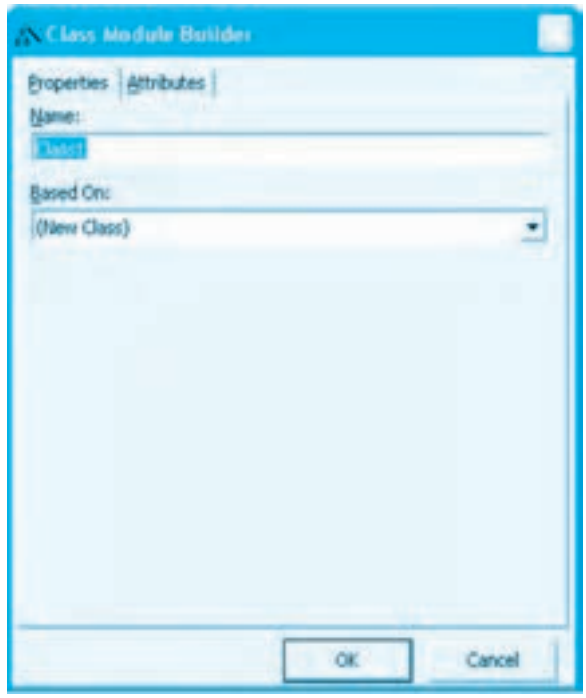


شکل ۲۱-۳

نکته: برای ایجاد یک کلاس جدید می‌توانید در نوار ابزار پنجره Class Builder روی دکمه New Class کلیک کنید.

۲- کادرمحاوره Class Module Builder مانند شکل ۲۲-۳ نمایش داده می‌شود. در کادر متن

Name نام کلاس و در کادر لیست ترکیبی بازشو Based On گزینه New Class را انتخاب کنید (این گزینه به‌طور پیش‌فرض انتخاب شده است)، سپس روی دکمه فرمان OK کلیک کنید. در این مرحله کلاس شما ایجاد می‌شود و نام آن در بخش Classes پنجره Class Builder نمایش داده می‌شود.



شکل ۲۲-۳

مثال ۸: می‌خواهیم کلاس clssquare را در پروژه Square به وسیله ابزار Class Builder ایجاد کنیم. برای این‌کار عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه Square را باز کنید.

۲- در پنجره پروژه روی آیکن کلاس clssquare کلیک‌راست کنید، سپس گزینه Remove clssquare را انتخاب کنید تا کلاس clssquare از پروژه خارج شود.

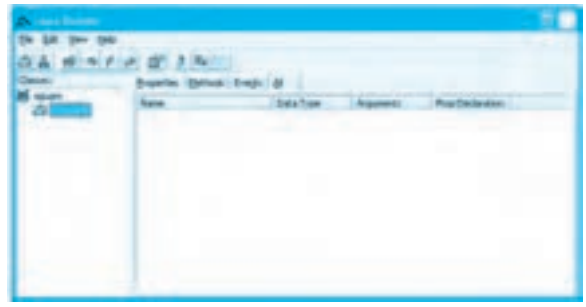
۳- در پنجره ویژوال‌بیسیک روی منوی Project کلیک کنید و گزینه Add Class Module را انتخاب کنید تا کادرمحاوره مربوطه نمایش داده شود.

۴- در کادرمحاوره Add Class Module آیکن VB Class Builder را انتخاب کنید سپس روی دکمه فرمان

Open کلیک کنید.

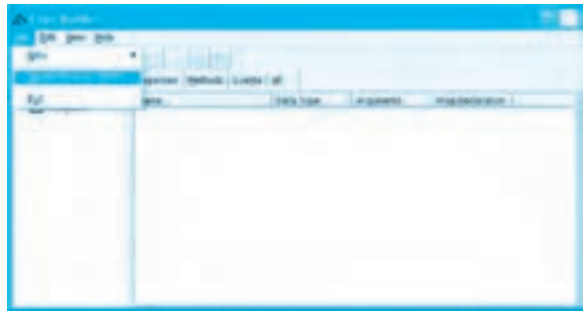
۵- در پنجره Class Builder با استفاده از گزینه New Class در منوی File یا دکمه  در نوار ابزار، یک کلاس جدید ایجاد کنید.

۶- در کادرمحاوره Class Module Builder و در کادر متن Name نام کلاس را روی مقدار clssquare تنظیم کرده و روی دکمه فرمان OK کلیک کنید. پس از این مرحله کلاس جدید، در بخش Classes پنجره Class Builder نمایش داده می‌شود (شکل ۲۳-۳).



شکل ۲۳-۳

۷- در پنجره Class Builder ابتدا روی منوی File و سپس روی گزینه Update Project کلیک کنید تا ماژول کلاس ایجاد شده به پروژه اضافه شود (شکل ۲۴-۳).



شکل ۲۴-۳

۸- پنجره Class Builder را بسته و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: در پروژه rectangle کلاس clsrectangle را حذف کرده و مجدداً آن را با استفاده از برنامه Class Builder ایجاد کنید.

نکته: در صورتی‌که در پروژه‌ای یک یا مجموعه‌ای از کلاس‌ها بدون استفاده از Class Builder ایجاد شده باشند پس از اجرای برنامه Class Builder پیامی مانند شکل ۲۵-۳ نمایش داده می‌شود مبنی بر این‌که کلاس‌های ایجاد شده با Class Builder ایجاد نشده‌اند روی دکمه فرمان OK کلیک کنید تا کادرمحاوره بسته شود و به پنجره Class Builder وارد شوید در این مرحله کلیه کلاس‌ها در پنجره Class Builder قابل مشاهده می‌باشند.

شکل ۲۵-۳

نحوه ایجاد عضو فیلد و خاصیت Class Builder

برای ایجاد عضو فیلد در یک کلاس می‌توانید به این ترتیب عمل کنید:

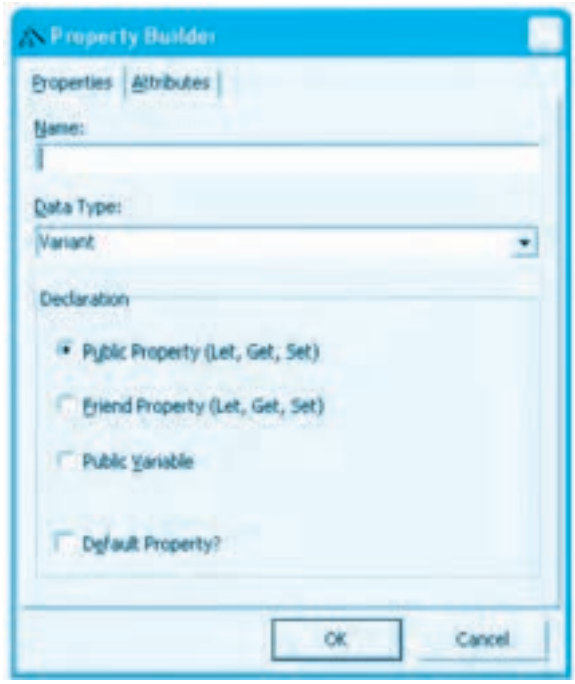
۱- در نوار منوی پنجره Class Builder روی منوی File و سپس روی گزینه‌های New و ... Property کلیک نمایید (شکل ۲۶-۳).

شکل ۲۶-۳

نکته: برای ایجاد عضو فیلد یا عضو خاصیت می‌توانید در نوار ابزار پنجره Class Builder روی دکمه کلیک کنید.

۲- کادرمحاوره Property Builder مانند شکل ۲۷-۳ نمایش داده می‌شود. در زبانه Properties و کادر متنی Name نام عضو فیلد یا خاصیت موردنظر را تایپ کنید، در کادر لیست ترکیبی بازشو Data Type نوع داده را برای فیلد یا خاصیت جدید انتخاب کنید. هم‌چنین می‌توانید با استفاده از دکمه‌های انتخاب موجود در کنترل قابل Declaration نیز میدان شناسایی فیلد یا خاصیت موردنظر را انتخاب کنید. دکمه انتخاب (Set و Get و Let) Public Property یک خاصیت را به‌صورت عمومی با رویه‌های Property ایجاد می‌کند و دکمه انتخاب Public Variable یک فیلد را به‌صورت عمومی تعریف می‌کند. به‌علاوه با استفاده از کادر علامت Default Property? می‌توان یک فیلد یا خاصیت را به عنوان فیلد یا خاصیت پیش‌فرض تعیین کرد.

۳- در پایان روی دکمه فرمان OK کلیک کنید تا عضو جدید ایجاد شود.



شکل ۲۷-۳

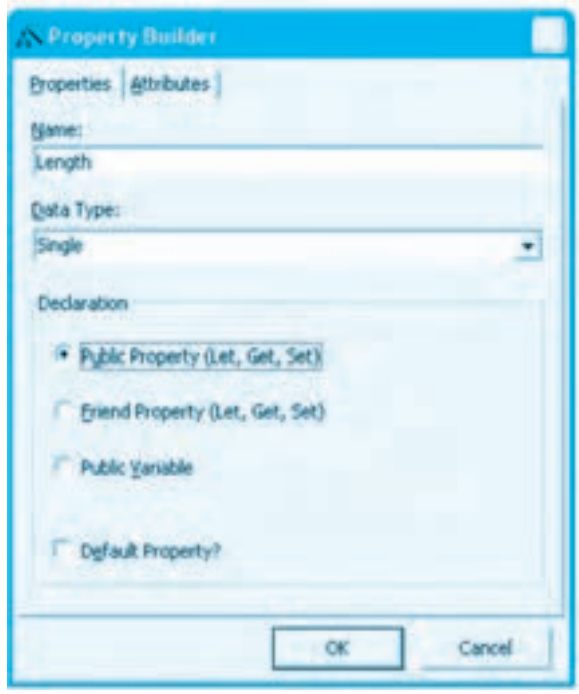
مثال ۹: می‌خواهیم در کلاس clssquare یک عضو فیلد برای ذخیره‌سازی و نگهداری طول ضلع مربع ایجاد کنیم. برای این کار عملیات بعد را به ترتیب انجام دهید:

۱- به پنجره ویژوال‌بیسیک و پروژه square بازگردید.

۲- به پنجره Class Builder بروید و در بخش Classes روی آیکن کلاس clssqare کلیک کنید.

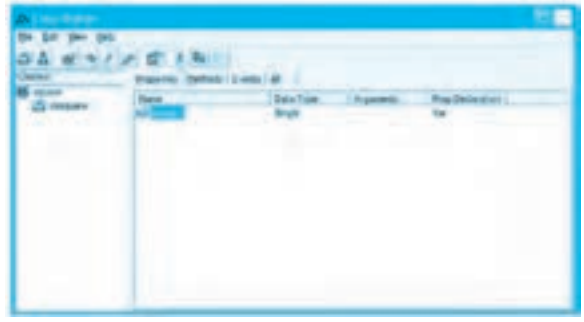
۳- با استفاده از نوار منو یا نوار ابزار پنجره Class Builder به کادرمحاوره Property Builder بروید و بعد

در کادر متن Name نام فیلد عبارت Length را تایپ کنید، سپس کادر لیست ترکیبی بازشوی Data Type را باز کرده و نوع داده را روی Single تنظیم کنید و در بخش Declaration دکمه انتخاب Public Variable را انتخاب کنید و در پایان روی دکمه فرمان OK کلیک کنید (شکل ۲۸-۳).



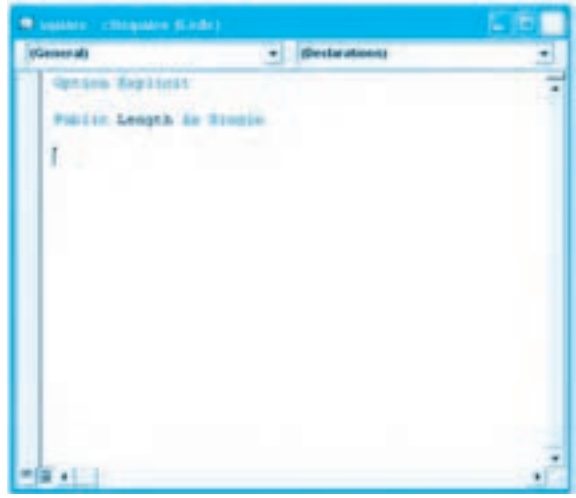
شکل ۲۸-۳

در این مرحله عضو فیلد ایجاد شده و نام و مشخصات آن در بخش سمت راست پنجره Class Builder نمایش داده می‌شود (شکل ۲۹-۳).

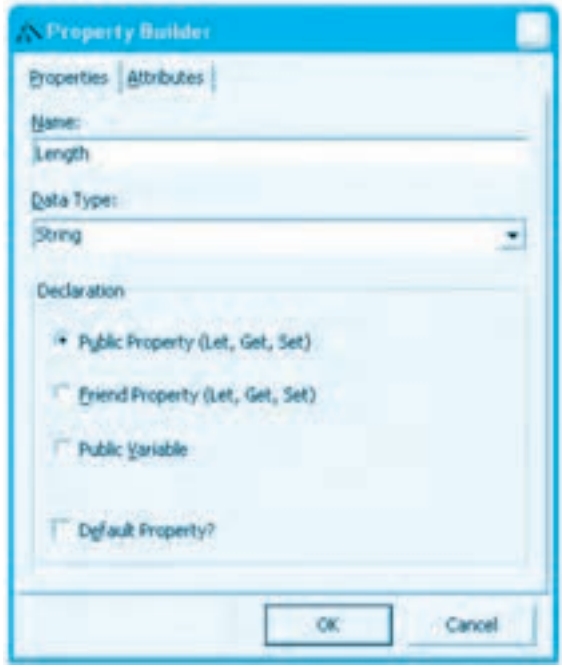


شکل ۲۹-۳

۴- در نوار منوی پنجره Class Builder روی منوی File و سپس روی گزینه Update Project کلیک کنید تا عضو فیلد به ماژول کلاس اضافه شود. سپس پنجره Class Builder را ببندید و به ماژول کلاس clssquare بروید و کد تولید شده را مشاهده و بررسی کنید. ماژول کلاس مانند شکل ۳۰-۳ می‌باشد.



شکل ۳۰-۳



شکل ۳۱-۳

۵- به ماژول کلاس clssquare بروید و عضو ایجاد شده را حذف کنید.

۶- با استفاده از نوار منو یا نوار ابزار پنجره Class Builder به کادرمحاوره Property Builder بروید، سپس در کادر متن Name نام خاصیت عبارت Length را تایپ کنید و در کادر لیست ترکیبی بازشو Data Type نوع خاصیت را Single انتخاب کنید. در مرحله بعد به بخش Declaration بروید و روی دکمه انتخاب Public Property کلیک کنید تا حوزه شناسایی عضو خاصیت تعیین شود (شکل ۳۱-۳).

روی دکمه فرمان OK کلیک کنید در این مرحله عضو خاصیت ایجاد شده و نام و مشخصات آن در بخش سمت راست پنجره Class Builder نمایش داده می‌شود (شکل ۳۲-۳).

شکل ۳۲-۳

۷- در نوار منوی پنجره Class Builder ابتدا روی منوی File و سپس گزینه Update Project کلیک کنید تا عضو جدید به ماژول کلاس اضافه شود.

شکل ۳۳-۳

۸- پنجره Class Builder را بسته و به پنجره ویژوال‌بیسیک بازگردید، سپس به پنجره ماژول کلاس clssquare بروید و کد تولید شده را مشاهده و بررسی کنید (شکل ۳۳-۳).

۹- تغییرات را ذخیره کرده و سپس برنامه را اجرا، آزمایش و بررسی کنید.

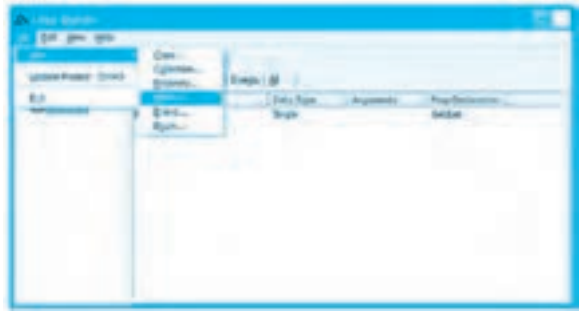
۱۰- به اجرای برنامه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: در پروژه rectangle و کلاس مستطیل با استفاده از Class Builder یک عضو فیلد برای طول و یک عضو خاصیت برای عرض ایجاد کنید.

نحوه ایجاد عضو متد با Class Builder

برای ایجاد عضو متد در یک کلاس می‌توانید به این ترتیب عمل کنید:

۱- در نوار منوی پنجره Class Builder روی منوی File و سپس روی گزینه‌های New و ... Method کلیک کنید (شکل ۳۴-۳).



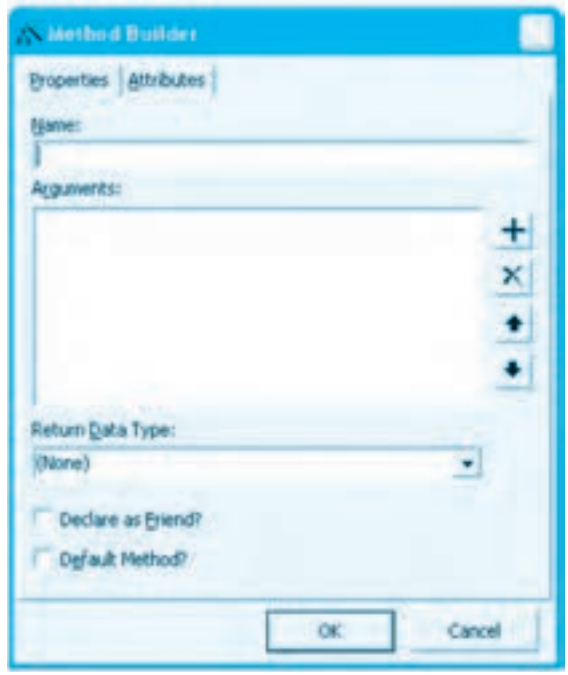
شکل ۳۴-۳

نکته: برای ایجاد عضو متد می‌توانید در نوار ابزار Class Builder روی دکمه کلیک کنید.

۲- در این مرحله کادرمحاوره Method Builder مانند شکل ۳۵-۳ نمایش داده می‌شود. در زبانه Properties و کادر متن Name نام عضو متد را تایپ کنید.

۳- در این مرحله می‌توانید آرگومان متد را در کادرمحاوره Method Builder با استفاده از دکمه تعریف کنید. یا آرگومان‌های تعریف شده را با دکمه حذف کنید یا با استفاده از دکمه‌های و ترتیب قرار گرفتن آن‌ها را در تعریف عضو متد مشخص کنید.

۴- اگر روی دکمه کلیک کنید، کادرمحاوره دیگری به نام Add Argument نمایش داده می‌شود (شکل ۳۶-۳).



شکل ۳-۳۵

شکل ۳-۳۶

در این کادرمحاوره بخش‌های مختلفی وجود دارند، در کادر متن Name می‌توانید نام آرگومان را تایپ کنید و با انتخاب کادر علامت ByVal آرگومان را به‌صورت ارسال با مقدار معرفی نمایید.

برای تعیین نوع آرگومان نیز می‌توانید از کادر لیست ترکیبی بازشو Data Type نوع داده موردنظر را انتخاب کنید و اگر آرگومان یک آرایه باشد کادر علامت Array را انتخاب نمایید.

برای آرگومان‌های اختیاری نیز می‌توانید روی کادر علامت Optional کلیک کنید و مقدار پیش‌فرض آرگومان اختیاری را نیز در کادر متن Default Value تایپ کنید.

۵- اگر عضو متد مقداری را بازگشت دهد، می‌توانید در کادرمحاوره Method Builder نوع داده بازگشتی را از کادر لیست ترکیبی بازشو Return Data Type انتخاب کنید.

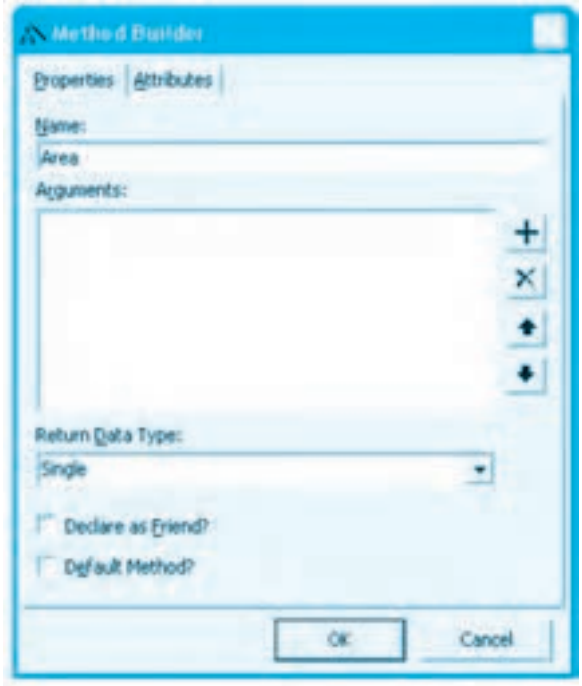
۶- در پایان روی دکمه فرمان OK کلیک کنید تا عضو جدید ایجاد شود.

مثال ۱۰: می‌خواهیم در کلاس clssquare یک عضو متد برای محاسبه مساحت مربع ایجاد کنیم، برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- به پنجره ویژوال‌بیسیک و پروژه Square بازگردید.

۲- به پنجره Class Builder بروید و در بخش Classes روی آیکن کلاس clssquare کلیک کنید.

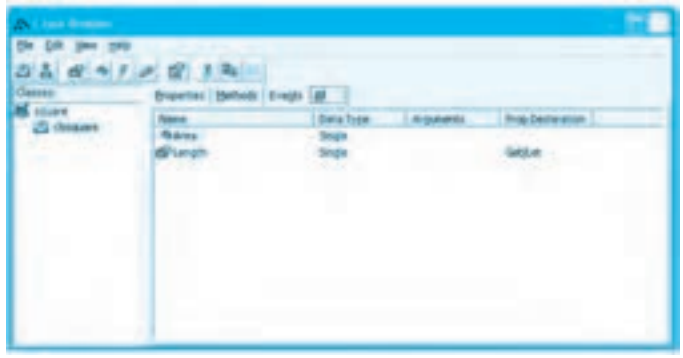
۳- با استفاده از نوار منو یا نوار ابزار پنجره Class Builder به کادرمحاوره Method Builder بروید و در کادر متن Name نام متد عبارت Area را تایپ کنید و در کادر لیست ترکیبی بازشو Return Data Type نوع داده بازگشتی را Single انتخاب کنید (شکل ۳۷-۳).



شکل ۳۷-۳

در پایان روی دکمه فرمان OK کلیک کنید.

در این مرحله عضو متد ایجاد شده و نام و مشخصات آن در بخش سمت راست پنجره Class Builder نمایش داده می‌شود (شکل ۳۸-۳).



شکل ۳۸-۳

۴- در نوار منوی پنجره Class Builder ابتدا روی منوی File و سپس گزینه Update Project کلیک کنید تا عضو جدید به ماژول کلاس اضافه شود.

۵- پنجره Class Builder را ببندید و به پنجره ویژوال‌بیسیک بازگردید.

سپس به پنجره ماژول کلاس clssquare بروید و کد تولید شده را مشاهده و بررسی کنید (شکل ۳۹-۳).

شکل ۳۹-۳

۶- در این مرحله باید دستورالعمل‌های متد Area را داخل آن تایپ کنید، بنابراین به ماژول کلاس و داخل

متد Area بروید و دستور زیر را تایپ کنید:

```
Area = snglength * snglength
```

۷- تغییرات را ذخیره کرده و سپس برنامه را اجرا، آزمایش و بررسی کنید.

۸- به اجرای برنامه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

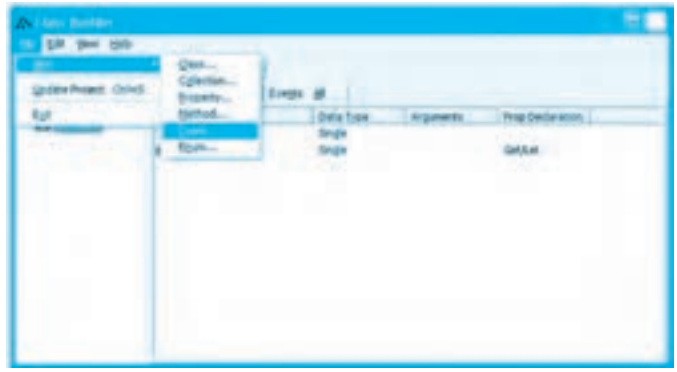
تمرین: در پروژه square و کلاس clssquare با استفاده از Class Builder یک عضو متد برای محاسبه محیط ایجاد کنید.

تمرین: در پروژه rectangle و کلاس مستطیل با استفاده از Class Builder یک عضو متد برای محاسبه مساحت و یک عضو متد برای محیط ایجاد کنید.

نحوه ایجاد عضو رویداد با Class Builder

برای ایجاد عضو رویداد در یک کلاس می‌توانید به این ترتیب عمل کنید:

۱- در نوار منوی پنجره Class Builder روی منوی File و سپس روی گزینه‌های New و ... Event کلیک کنید (شکل ۴۰-۳).

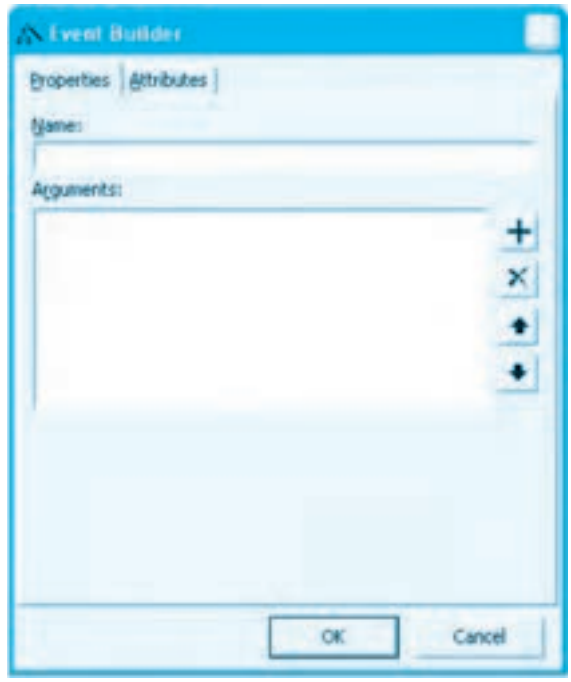


شکل ۴۰-۳

نکته: برای ایجاد عضو رویداد می‌توانید در نوار ابزار Class Builder روی دکمه کلیک کنید.

۲- در این مرحله کادرمحاوره Event Builder مانند شکل ۴۱-۳ نمایش داده می‌شود. در زبانه Properties و کادر متن Name نام عضو رویداد را تایپ کنید.

۳- در پایان روی دکمه فرمان OK کلیک کنید تا عضو جدید ایجاد شود.



شکل ۴۱-۳

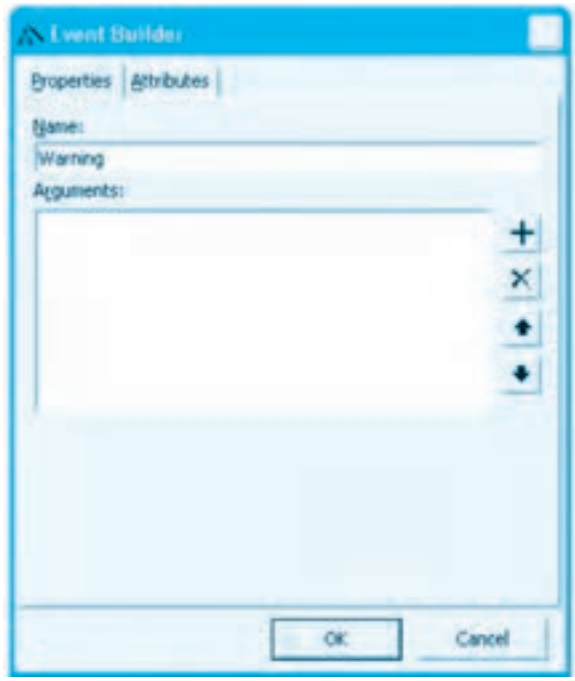
مثال ۱۱: می‌خواهیم در کلاس clssquare یک عضو رویداد ایجاد کنیم تا وقتی مقدار مساحت مربع کمتر از ۱۰ واحد شود رویداد به‌طور خودکار اجرا شود. برای این کار عملیات زیر را به ترتیب انجام دهید:

۱- به پنجره ویژوال‌بیسیک و پروژه square بازگردید.

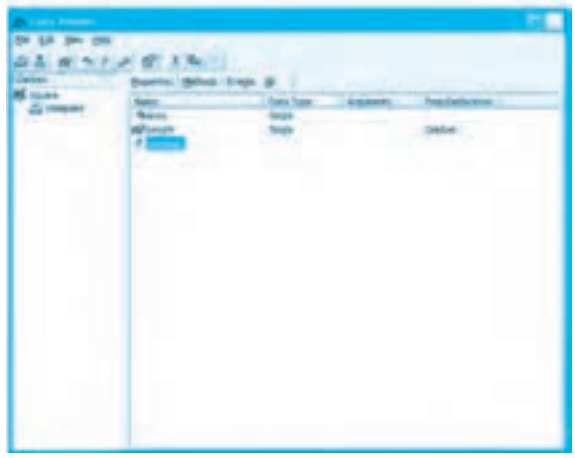
۲- به پنجره Class Builder بروید و در بخش Classes روی آیکن کلاس clssquare کلیک کنید.

۳- با استفاده از نوار منو یا نوار ابزار پنجره Class Builder به کادرمحاوره Event Builder بروید و بعد در کادر متن Name برای نام رویداد عبارت Warning را تایپ کنید (شکل ۴۲-۳).

سپس روی دکمه فرمان OK کلیک کنید، در این مرحله عضو متد ایجاد شده و نام و مشخصات آن در بخش سمت راست پنجره Class Builder نمایش داده می‌شود (شکل ۴۳-۳).



شکل ۳-۴۲



شکل ۳-۴۳

۴- در نوار منوی پنجره Class Builder ابتدا روی منوی File و سپس گزینه Update Project کلیک کنید

تا عضو جدید به ماژول کلاس اضافه شود.

۵- پنجره Class Builder را ببندید و به پنجره ویژوال‌بیسیک بازگردید. سپس به پنجره ماژول کلاس clssquare بروید و کد تولید شده را مشاهده و بررسی کنید (شکل ۴۴-۳).

شکل ۴۴-۳

۶- در این مرحله لازم است رویداد در متد مساحت فراخوانی شود، بنابراین متد مساحت مربع را به این صورت تنظیم کنید:

```
Public Function Area() As Single
    Area = snglength * snglength
    If (snglength * snglength < 10) Then RaiseEvent Warning
End Function
```

۷- تغییرات را ذخیره کرده و سپس برنامه را با ایجاد یک شئ از کلاس مربع همراه با رویدادهای آن آزمایش و بررسی نمایید.

۸- به اجرای برنامه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: در پروژه rectangle و کلاس مستطیل با استفاده از Class Builder یک عضو رویداد ایجاد کنید تا در زمانی که مقدار مساحت و محیط مستطیل از ۱۵ واحد کوچک‌تر باشد به‌طور خودکار اجرا

شود، سپس با ایجاد شیئی از کلاس مستطیل مساحت و محیط یک مستطیل دلخواه را محاسبه نموده و از رویدادهای آن نیز استفاده کنید.

۷-۳-۳ ایجاد مجموعه‌ای از اشیا

تاکنون نحوه ایجاد یک شئ به‌صورت مستقل و مجزا از سایر اشیا را آموختید، اما گاهی اوقات ممکن است لازم باشد مجموعه‌ای از دو یا چند شئ را به‌طور همزمان ایجاد کرده و مورد استفاده قرار دهید. در این صورت روش‌های قبلی برای نگهداری و مدیریت گروهی از اشیای کاری مشکل و در بعضی موارد غیرممکن خواهد بود برای رفع این مشکل می‌توان یکی از این روش‌ها را استفاده کرد:

آرایه از اشیا

در کتاب برنامه‌نویسی ویژوال‌بیسیک مقدماتی نحوه تعریف و استفاده آرایه از فرم‌ها و کنترل‌ها را فراگرفته‌اید نحوه تعریف و استفاده آرایه‌ای از اشیایی که از سایر کلاس‌ها به‌وجود می‌آیند نیز به همان صورت امکان‌پذیر است.

مثال ۱۲: می‌خواهیم پروژه employee را به گونه‌ای تنظیم کنیم تا به وسیله آن بتوان اطلاعات ۵ کارمند را ذخیره نمود و در صورت نیاز بتوان اطلاعات یک کارمند را به وسیله کد کارمند مشاهده کرد.

برای انجام این مثال عملیات بعد را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه employee را باز کنید.

۲- پنجره ماژول کد mdlobject را باز کنید و دستورات موجود در آن را به این صورت تغییر دهید:

```
Option Base 1

Public inttotal As Integer

Public employee(5) As New clsemployee
```

در این دستورات یک متغیر عمومی با نام inttotal تعریف شده است این متغیر تعداد کارمندانی را که اطلاعات آن‌ها دریافت شده است نگهداری می‌کند. به علاوه یک آرایه با نام employees با ۵ عضو از کلاس clsemployee نیز تعریف شده است در ضمن با استفاده از دستور Option Base ۱ اندیس شروع در آرایه employees از مقدار ۱ شروع می‌شود.

۳- رویداد Load فرم frmmain را به این صورت تنظیم کنید:

```
Private Sub MDIForm_Load()
```

```
    inttotal = 1
End Sub
```

در این رویداد متغیر شمارنده inttotal روی مقدار ۱ تنظیم می‌شود.

۴- به ماژول کد فرم frmregister بروید و رویداد Click دکمه Save را به این صورت تنظیم کنید:

```
Private Sub cmdsave_Click()
    mdlobject.employees(inttotal).firstname = txtfirstname.Text
    mdlobject.employees(inttotal).lastname = txtlastname.Text
    mdlobject.employees(inttotal).employeeid = Val(txtid.Text)
    mdlobject.employees(inttotal).b_date = CDate(txtbdate.Text)
    inttotal = inttotal + 1
End Sub
```

در این رویداد، هر بار اطلاعات در یکی از اعضای آرایه ذخیره می‌شود و در پایان متغیر شمارنده یک واحد افزایش می‌یابد تا در مرحله بعد داده‌ها در عضو بعدی آرایه ذخیره شوند.

۵- فرم frmdisplay را باز کرده و کنترل‌های آن را مانند شکل ۳-۴۵ تنظیم کنید، سپس خاصیت Enabled کنترل کادر متن txtid را روی مقدار True تنظیم کنید.

شکل ۳-۴۵

۶- به ماژول کد فرم frmdisplay بروید و دستورات آن را به این صورت تنظیم کنید:

```
Private Sub cmddisplay_Click()
   Dim inti As Integer
   For inti = 1 To 5
      If (mdlobject.employees(inti).employeeid = Val(txtid.Text)) Then
         txtfirstname.Text = mdlobject.employees(inti).firstname
         txtlastname.Text = mdlobject.employees(inti).lastname
         txtbdate.Text = Str(mdlobject.employees(inti).b_date)
         Exit Sub
      End If
   Next inti
   MsgBox "The Employee Is Not Exist!", VbMsgBoxStyle.vbCritical, "Not Found"
End Sub
```

در این رویداد با استفاده از یک حلقه For آرایه جستجو می‌شود در هر بار اجرای حلقه، کد کارمندی که در کادر متن tytid دریافت شده است به وسیله دستور شرطی If با کد کارمندی یک عضو آرایه مقایسه می‌شود.

اگر نتیجه بررسی درست باشد سایر مشخصات کارمندی که پیدا شده است در کادرهای متن نمایش داده می‌شود، سپس با اجرای دستور Exit Sub اجرای رویه رویداد خاتمه می‌یابد زیرا با پیدا شدن کارمند موردنظر دیگر نیازی به جستجو و ادامه اجرای حلقه For نمی‌باشد.

هم‌چنین اگر حلقه For کلیه اعضای آرایه را بررسی کند اما کارمند مربوطه موجود نباشد، حلقه بعد از بررسی آخرین عضو خاتمه یافته و یک کادر پیغام، عدم وجود کارمند را در آرایه نشان می‌دهد.

۷- تغییرات را ذخیره کرده و برنامه را اجرا کنید.

۸- اطلاعات چند کارمند را ثبت کنید، سپس با استفاده از فرم Display کد یک کارمند را وارد کنید و روی دکمه فرمان Display کلیک کنید.

این کار را برای سایر کارمندان نیز انجام داده و نتیجه را بررسی کنید، سپس یک کد کارمندی اشتباه را وارد کرده و نتیجه را مجدداً بررسی کنید.

۹- اجرای برنامه را خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه library را به شکلی تنظیم کنید تا بتوان اطلاعات چندین کتاب (حداکثر ۲۰ کتاب) با استفاده از مفهوم آرایه‌ای از اشیا دریافت و ذخیره کرد و در صورت نیاز بتوان اطلاعات هر کتاب دلخواه را به وسیله کد کتاب در کتابخانه جستجو کرد.

کلکسیونی از اشیا

روش مناسب‌تری که برای نگهداری و مدیریت مجموعه‌ای از اشیا قابل استفاده می‌باشد ایجاد کلکسیونی از اشیا است. برای ایجاد یک کلکسیون از اشیا از کلاسی به نام Collection استفاده می‌شود. مزیت کلکسیون‌ها نسبت به آرایه‌ها در این است که کلکسیون‌ها می‌توانند هر تعداد شئ را که بخواهید در خود جای دهند و به‌صورت یک آرایه پویای پیشرفته عمل کنند، در ضمن می‌توان هر شئ را در صورت نیاز از کلکسیون خارج نمود و فضای مورد استفاده آن را آزاد کرد. برای تعریف یک کلکسیون می‌توان به این صورت عمل نمود:

Public یا Private نام کلکسیون As New Collection

وقتی یک کلکسیون از نوع Collection تعریف می‌شود مجموعه‌ای از متدها و خاصیت‌ها برای مدیریت کلکسیون قابل استفاده هستند، این متدها را می‌توانید در جدول ۱-۳ مشاهده کنید.

جدول ۱-۳ متدها و خاصیت‌های شئ کلکسیون

نام متد یا خاصیت	عملکرد
Add	با استفاده از این متد می‌توان یک شئ را به کلکسیون اضافه کرد.
Count	این خاصیت تعداد اشیای موجود در کلکسیون را مشخص می‌کند.
Remove	با استفاده از این متد می‌توان یک شئ را از کلکسیون حذف کرد.

مثال ۱۳: می‌خواهیم پروژه employee را به شکلی تنظیم کنیم تا به وسیله آن بتوان اطلاعات هر تعداد کارمند دلخواه را ذخیره نمود یا با استفاده از کد کارمندی، اطلاعات یک کارمند را مشاهده کرد. برای انجام این مثال عملیات بعد را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه employee را باز کنید.

۲- پنجره ماژول کد mdl object را باز کنید و دستورات آن را حذف کرده و این دستور را داخل آن تایپ کنید:

```
Public colemployees As New Collection
```

این دستور یک کلکسیون با نام colemployees را برای استفاده در سطح تمام پروژه تعریف می‌کند.

۳- به ماژول کد فرم frmmain بروید و رویداد Load آن را حذف کنید.

۴- به ماژول کد فرم frmregister بروید و رویداد Click دکمه Save را به این صورت تنظیم کنید:

```
Private Sub cmdsave_Click()
   Dim employee As clsemployee
   Set employee = New clsemployee
   employee.firstname = txtfirstname.Text
   employee.lastname = txtlastname.Text
   employee.employeeid = Val(txtid.Text)
   employee.b_date = CDate(txtbdate.Text)
   mdlobject.colemployees.Add employee
End Sub
```

در این رویداد ابتدا یک شئ (employee) از روی کلاس clsemployee تعریف می‌شود، سپس خاصیت‌های آن مقداردهی می‌شوند و در مرحله بعد با استفاده از متد Add شئ employee به شئ کلکسیون اضافه می‌شود.

۵- فرم frmdisplay را باز کرده و رویداد Click دکمه فرمان Save را به این صورت تنظیم کنید:

```
Private Sub cmddisplay_Click()
   Dim inti As Integer
   For inti = 1 To colemployees.Count
      If (mdlobject.colemployees(inti).employeeid = Val(txtid.Text)) Then
         txtfirstname.Text = mdlobject.colemployees(inti).firstname
         txtlastname.Text = mdlobject.colemployees(inti).lastname
         txtbdate.Text = Str(mdlobject.colemployees(inti).b_date)
         Exit Sub
      End If
```

```
    Next inti
    MsgBox "The Employee Is Not Exist!", VbMsgBoxStyle.vbCritical, "Not Found"
End Sub
```

در این رویداد با استفاده از یک حلقه For عملیات جستجو در کلکسیون انجام می‌شود. برای مقدار خاتمه حلقه از خاصیت Count کلکسیون Colemployees استفاده شده است این خاصیت تعداد اشیای موجود در کلکسیون را مشخص می‌کند، بنابراین می‌توان از آن به عنوان مقدار خاتمه حلقه استفاده کرد.

در داخل حلقه نیز می‌توان با استفاده از اندیس هر شئ در کلکسیون Colemployees به هریک از اشیای موجود در کلکسیون دسترسی پیدا کرد تا در صورت پیدا شدن کارمند مورد جستجو اطلاعات مربوط به وی نمایش داده شوند.

۶- تغییرات را ذخیره کرده و برنامه را اجرا کنید.

۷- اطلاعات چند کارمند را ثبت کرده و با استفاده از فرم Display اطلاعات ذخیره شده را به وسیله کد کارمند جستجو نمود تا در صورت وجود داشتن، نمایش داده شوند.

۸- اجرای برنامه را خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

برای دسترسی به اشیای ذخیره شده در یک کلکسیون می‌توان به جای For از حلقه For Each استفاده کرد.

شکل کلی نحوه استفاده از این دستور به این صورت است:

```
For Each نام شئ In نام کلکسیون
        دستورات
Next نام شئ
```

در این حلقه احتیاجی به تعیین مقدار شروع و خاتمه نیست و حلقه به‌طور خودکار از اولین شئ در کلکسیون شروع می‌کند و با رسیدن به آخرین شئ در کلکسیون اجرای حلقه پایان می‌یابد. برای دسترسی به اشیای داخل کلکسیون نیز یک شئ از همان کلاسی که اشیای داخل کلکسیون از آن ایجاد شده‌اند تعریف می‌شود و نام آن بعد از کلمه کلیدی Each و Next قرار می‌گیرد. داخل حلقه For Each می‌توان با استفاده از این نام به اعضای کلکسیون دسترسی پیدا کرد. به عنوان مثال برای نمایش و اطلاعات کلیه کارمندان روی فرم می‌توان به این صورت عمل کرد:

```
Dim employee As New clsemployee
 For Each employee In mdlobject.colemployees
    Print employee.employeeid
```

```
Print employee.firstname
Print employee.lastname
Print (employeeb_date)
Next employee
```

در این دستورات ابتدا یک شئ با نام employee از کلاس clsemployee تعریف شده است تا از آن در حلقه For Each برای دسترسی به اشیای داخل کلکسیون Colemployees استفاده شود، سپس حلقه For Each با استفاده از نام شئ employe و نام کلکسیون Colemployee اجرا می‌شود. در این حلقه با استفاده از نام شئ employee ابتدا اطلاعات کارمند اول در کلکسیون نمایش داده می‌شود با رسیدن به دستور Next employee حلقه مجدداً اجرا می‌شود و اطلاعات دومین کارمند را نمایش می‌دهد. به همین ترتیب اجرای حلقه تا رسیدن به آخرین کارمند و نمایش اطلاعات وی ادامه می‌یابد و سپس به‌طور خودکار خاتمه می‌یابد.

مثال ۱۴: می‌خواهیم پروژه employee را به شکلی تنظیم کنیم که برای جستجو و نمایش اطلاعات یک کارمند در فرم Display از حلقه For Each به جای For استفاده شود. برای انجام این مثال عملیات بعد را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه employee را باز کنید.

۲- به پنجره ماژول کد فرم frmdisplay بروید و رویداد Click دکمه فرمان Save را به این صورت تنظیم کنید:

```
Private Sub cmddisplay_Click()
  Dim employee As New clsemployee
  For Each employee In mdlobject.colemployees
    If (employee.employeeid = Val(txtid.Text)) Then
      txtfirstname.Text = employee.firstname
      txtlastname.Text = employee.lastname
      txtbdate.Text = Str(employeeb_date)
      Exit Sub
    End If
  Next employee
  MsgBox "The Employee Is Not Exist!", VbMsgBoxStyle.vbCritical, "Not Found"
```

```
End Sub
```

در این رویداد نیز با استفاده از حلقه For Each عملیات جستجو و نمایش اطلاعات کارمند موردنظر انجام می‌شود.

۳- تغییرات را ذخیره کرده و برنامه را اجرا و آزمایش کنید.

تمرین: پروژه library را به شکلی تنظیم کنید تا بتوان اطلاعات چندین کتاب را با استفاده از مفهوم کلکسیون دریافت و ذخیره کرد و در صورت نیاز بتوان اطلاعات هر کتاب دلخواه را به وسیله کد کتاب در کتابخانه جستجو نمود.

مثال ۱۵: می‌خواهیم پروژه employee را به شکلی تنظیم کنیم تا بتوان اطلاعات یک کارمند را از داخل کلکسیون حذف نمود.

برای انجام این مثال عملیات زیر را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه employee را باز کنید.

۲- یک فرم جدید با نام frmdelete و عنوان Delete مانند شکل ۴۶-۳ به پروژه اضافه کنید و یک کنترل کادر متن با نام txtindex و دو کنترل دکمه فرمان با نام‌های btndelete و btnclose و عنوان‌های Delete و Close روی فرم قرار دهید. سپس فرم را با نام delete ذخیره کنید.

شکل ۴۶-۳

۳- رویداد Click دکمه فرمان btndelete را به این صورت تنظیم کنید:

```
Private Sub cmddelete_Click()
    mdlobject.colemployees.Remove (Val(txtindex.Text))
End Sub
```

در این رویداد با استفاده از متد Remove یک عضو از داخل کلکسیون colemployees حذف می‌شود، متد Remove شماره اندیس یک شئ را در کلکسیون دریافت کرده و آن را حذف می‌کند. در این دستورات اندیس شئ به وسیله

کادر متن txtindex دریافت شده تا متد Remove را با استفاده از آن شئ با توجه به موقعیت قرار گرفتن شئ در کلکسیون حذف کند.

۴- به پنجره فرم frmmain بروید و در نوار منوی فرم یک گزینه جدید با نام mnudelete و عنوان Delete به انتهای منوی information اضافه کنید سپس رویداد گزینه جدید را به این صورت تنظیم کنید:

```
Private Sub mnudelete_Click()
    frmdelete.Show
End Sub
```

در این رویداد با استفاده از متد Show فرم frmdelete نمایش داده می‌شود.

۵- تغییرات را ذخیره کرده و برنامه را اجرا کنید.

۶- اطلاعات چند کارمند را ثبت کنید و بعد با استفاده از گزینه Display از ذخیره شدن اطلاعات اطمینان حاصل کنید

۷- در منوی Information روی گزینه Delete کلیک کنید تا فرم Delete نمایش داده شود، سپس با توجه به ترتیب ثبت اطلاعات، اندیس یک شئ را در کادر متن تایپ کنید و بعد روی دکمه فرمان Delete کلیک کنید.

۸- به فرم Display بازگردید و بررسی کنید آیا اطلاعات کارمند موردنظر حذف شده است یا خیر؟ برای این کار کافی است کد کارمندی را که حذف کرده‌اید در کادر متن مربوطه تایپ کرده و روی دکمه فرمان Display کلیک کنید. در صورتی‌که اطلاعات کارمند موردنظر حذف شده باشد کادر پیغام، عدم وجود آن را نمایش می‌دهد.

۹- عملیات قبل را برای سایر کارمندان نیز تکرار کنید و نتیجه را بررسی نمایید.

۱۰- به اجرای برنامه پایان داده و به پنجره برنامه ویژوال‌بیسیک بازگردید.

۸-۳-۳ دستور With ... End Width

گاهی اوقات برای تنظیم خاصیت‌های یک شئ باید هر بار با استفاده از نام شئ یک خاصیت را مورد استفاده قرار داد. اگر تعداد خاصیت‌های مورد نیاز زیاد باشد لازم است چندین بار نام شئ تکرار شود در این حالت برای جلوگیری از تکرار بی‌مورد نام شئ و ساده‌تر شدن دستورات می‌توان از دستور With استفاده کرد. شکل کلی نحوه استفاده از این دستور به این صورت است:

```
With نام شئ
.
.
.
End With
```

به عنوان مثال برای تنظیم چند خاصیت یک کنترل دکمه فرمان با نام cmdclose می‌توان به این صورت عمل کرد:

```
Private Sub cmddelete_Click()
```

```
    mdlobject.colemployees.Remove (Val(txtindex.Text))
End Sub
```

در این دستورات با استفاده از دستور With نام شئ cmdclose فقط یک‌بار مورد استفاده قرار می‌گیرد و لازم نیست برای دسترسی به هر خاصیت نام شئ مجدداً تکرار شود.

به عنوان مثالی دیگر در پروژه employee و فرم frmregister می‌توان رویداد Click دکمه فرمان cmdsave را با استفاده از دستور With به این صورت تنظیم کرد:

```
Private Sub cmdsave_Click()
    Dim employee As clsemployee
    Set employee = New clsemployee
    With employee
        .firstname = txtfirstname.Text
        .lastname = txtlastname.Text
        .employeeid = Val(txtid.Text)
        .b_date = CDate(txtbdate.Text)
    End With
    mdlobject.colemployees.Add employee
End Sub
```

تمرین: پروژه Library را به شکلی تنظیم کنید تا برای مقداردهی خاصیت‌های اشیا از دستور With … End With استفاده شود.

۹-۳-۳ مرورگر شئ (Object Browser)

مرورگر شئ ابزار دیگری است که در ویژوال‌بیسیک برای مدیریت بهتر اشیای موجود در یک پروژه استفاده می‌شود، با استفاده از این برنامه می‌توانید کلاس‌هایی را که در پروژه ایجاد شده‌اند به همراه فیلدها، خاصیت‌ها، متدها و رویدادهایشان مشاهده کنید.

به‌علاوه می‌توانید ثابت‌های موجود در کتابخانه اشیا و رویه‌های موجود در پروژه را مشاهده کنید یا با استفاده از اجزای موجود در این برنامه، شئ موردنظر را جستجو کنید.

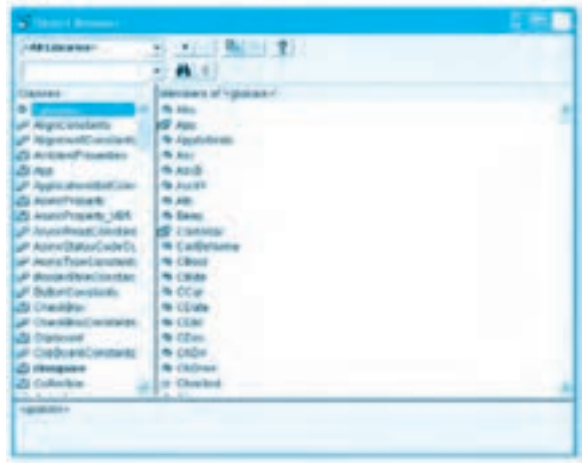
برای دسترسی به برنامه مرورگر شئ می‌توانید یکی از این روش‌ها را انتخاب کنید:

۱- در نوار منوی پنجره ویژوال‌بیسیک ابتدا روی منوی View و سپس روی گزینه Object Browser کلیک نمایید.

۲- کلید F2 را بفشارید.

۳- در پنجره ویژوال‌بیسیک و نوار ابزار آن، روی دکمه  کلیک کنید.

پس از انجام یکی از این روش‌ها پنجره مرورگر شئ مانند شکل ۳-۴۷ نمایش داده می‌شود.



شکل ۳-۴۷

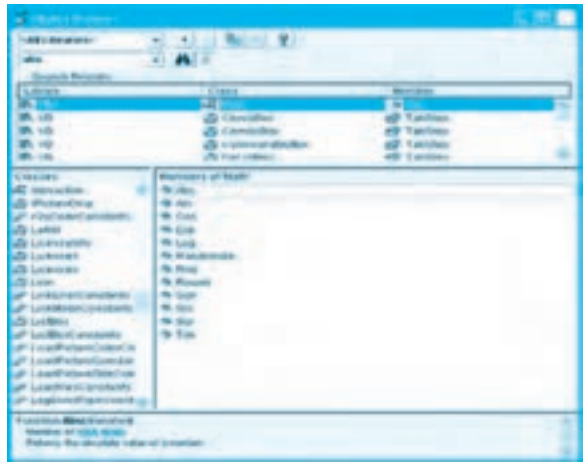
در بخش بالایی و سمت چپ پنجره Object Browser یک کادر لیست ترکیبی بازشو وجود دارد که به شما اجازه می‌دهد تا اجزای موجود در بخش‌های مختلف را براساس جدول ۳-۲ برای نمایش انتخاب کنید.

جدول ۳-۲

عنوان گزینه	توضیح
<All Libraries>	تمام کتابخانه‌های اشیا
نام پروژه	تمام اشیای موجود در پروژه جاری
VB	تمام اشیا و رویه‌های موجود در ویژوال‌بیسیک
VBA	اشیای موجود در فایل کتابخانه‌ای VBA (Visual Basic For Application)
VBRUN	اشیای موجود در فایل کتابخانه‌های VBRUN (اشیا و رویه‌های ویژوال‌بیسیک برای زمان اجرا)

نکته: اگر برنامه Object Browser را در پروژه‌ای که از قبل طراحی شده است اجرا کنید، علاوه بر موارد ارائه شده در جدول ۲-۳، اسامی کلاس‌های موجود در پروژه نیز در کادر لیست ترکیبی بازشو نمایش داده می‌شوند.

یک کادر لیست ترکیبی بازشو نیز در زیر کادر لیست ترکیبی بازشو اول قرار گرفته است، از این کادر لیست می‌توانید برای جستجوی شئ، رویه و هر نوع عبارتی که موردنظر است استفاده کنید. مثلاً برای پیدا کردن تابع Abs می‌توانید عبارت abs را داخل این کادر لیست ترکیبی بازشو تایپ کرده و روی دکمه Find که در کنار آن قرار دارد کلیک کنید (شکل ۴۸-۳). نتایج جستجو در بخش Search Results نمایش داده می‌شود و در قسمت پایین پنجره نیز نحوه استفاده از تابع Abs و عملکرد آن توضیح داده می‌شود.



شکل ۴۸-۳

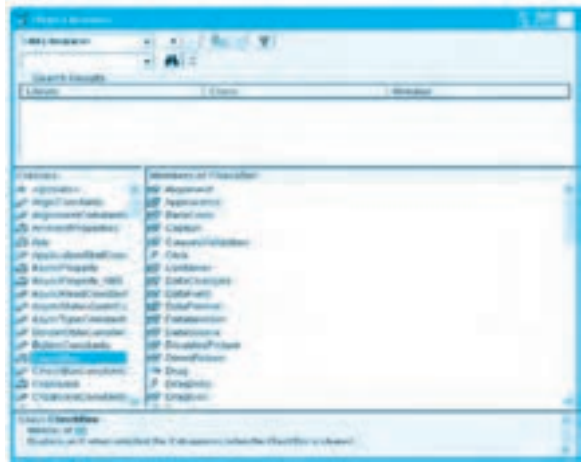
نحوه استفاده از تابع abs و عملکرد آن توضیح داده می‌شود.

در سمت راست دکمه Find دکمه (Show/Hide Search Results) قرار دارد (شکل ۴۸-۳)، اگر روی این دکمه کلیک کنید نتایج جستجو مخفی شده و با کلیک دوباره روی آن، نتایج جستجو نمایش داده می‌شود.

بخش‌های دیگری که در پنجره Object Browser وجود ندارد بخش Classes و Members می‌باشند. در بخش Classes تمام کلاس‌های موجود در کتابخانه‌ها یا پروژه انتخاب شده نمایش داده می‌شوند، اگر در پروژه‌ای یک کلاس ایجاد و کدنویسی شده باشد با رنگ تیره‌تر نمایش داده می‌شود.

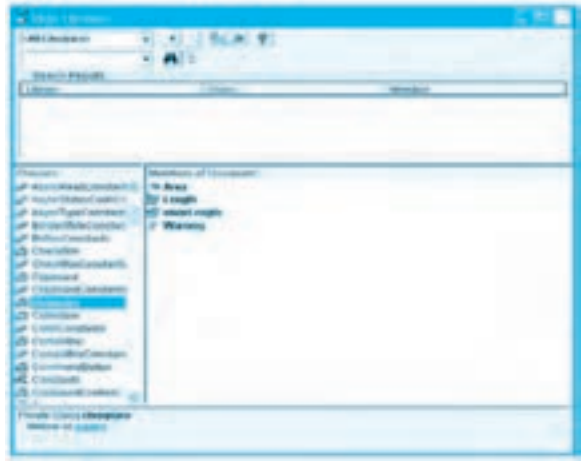
به‌طور پیش‌فرض اولین گزینه در این بخش <globals> است که تمامی اعضای آن نیز در بخش Members

نمایش داده می‌شوند (شکل ۴۷-۳)، اگر در بخش Classes یک کلاس را انتخاب کنید اجزای تشکیل‌دهنده کلاس در بخش Members نمایش داده می‌شوند (شکل ۴۹-۳).



شکل ۴۹-۳

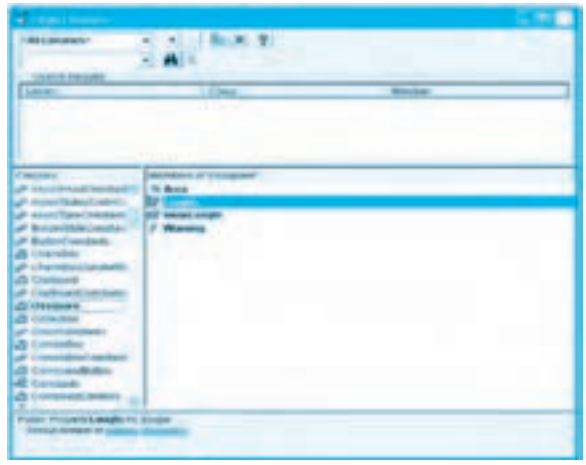
به عنوان مثال اگر پروژه square را باز کرده و برنامه Object Browser را اجرا کنید و در بخش Classes، کلاس clssquare را انتخاب کنید تمام اعضای آن در بخش Members نمایش داده می‌شوند (شکل ۵۰-۳).



شکل ۵۰-۳

در قسمت پایانی پنجره Object Browser یک بخش مستطیل شکل به نام بخش جزییات وجود دارد. زمانی که یک کلاس را در بخش Classes یا یک خاصیت، فیلد، متد یا رویداد را در بخش Members انتخاب می‌کنید،

جزییاتی در رابطه با کلاس یا نحوه تعریف عضو و عملکرد آن در بخش جزییات نمایش داده می‌شود.
به‌علاوه می‌توانید با کلیک روی یک Link که در بخش جزییات نشان داده می‌شود به پروژه یا کلاسی که عضو انتخاب شده در آن قرار دارد راهنمایی شوید.
به عنوان مثال اگر در بخش Classes روی کلاس clssquare کلیک کنید و سپس در بخش Members روی خاصیت Length کلیک کنید، در بخش جزییات تعریف خاصیت Length و در زیر آن یک لینک نمایش داده می‌شود (شکل ۵۱-۳) که اگر روی این لینک کلیک کنید فقط کلاس‌های پروژه نمایش داده می‌شوند (شکل ۵۲-۳).



شکل ۵۱-۳

شکل ۵۲-۳

علاوه بر مواردی که گفته شد می‌توان به دکمه (Go Back) و دکمه (Go Forward) اشاره کرد این دکمه‌ها به ترتیب شما را در سلسله مراتبی که بین کلاس‌ها، خواص، متدها و سایر اجزا حرکت کرده‌اید به عقب یا جلو حرکت می‌دهند.

در کنار این دکمه‌ها دکمه Copy to Clipboard قرار دارد که با استفاده از این دکمه می‌توان از کلاس انتخاب شده در بخش Classes یا عضو انتخاب شده در بخش Members یک کپی مشابه در کلیپ‌برد قرار دارد تا در محل دیگری که موردنظر است آن را Paste کرد به‌علاوه می‌توان اطلاعات موجود در بخش جزییات را با عمل کشیدن انتخاب نمود و سپس با این دکمه یک کپی از آن در کلیپ‌برد قرار داد.

در سمت راست دکمه Copy، دکمه View Definition قرار دارد به وسیله این دکمه نیز می‌توان مستقیماً به محلی که کلاس، خاصیت متد یا رویداد انتخاب شده در آن وجود دارد قرار گرفت.

آخرین دکمه، در پنجره Object Browser دکمه (Help) می‌باشد در صورتی‌که روی این دکمه کلیک کرده یا کلید F1 را بفشارید پنجره راهنمای MSDN نمایش داده می‌شود و در رابطه با کلاس، خاصیت، متد یا رویداد انتخاب شده راهنمایی‌هایی در اختیار شما قرار خواهد گرفت.

تمرین: پروژه rectangle را باز کرده و با استفاده از برنامه Object Browser اعضای آن را مشاهده و بررسی کنید.

تمرین: به وسیله ابزار Object Browser درباره اعضای کلاس‌های Form، Command Button و توابع Chr، Len و Sin به‌دست آورید.

۱۰-۳-۳ نحوه ایجاد ActvieX DLL

ActiveX DLL یکی از امکاناتی است که به وسیله تکنولوژی COM (Component Object Model) ارائه می‌شود و در COM تکامل می‌یابد. در این تکنولوژی‌ها سعی می‌شود تا اجزای تشکیل‌دهنده برنامه‌ها که می‌تواند در پروژه‌های مختلف مورد استفاده قرار گیرند، یک‌بار تهیه و تولید شوند تا تولیدکننده آن یا حتی سایر برنامه‌نویسان بتوانند بدون هیچ زحمتی آن اجزا را در برنامه‌های خود مورد استفاده قرار دهند.

ActiveX DLL (Dynamic Link Library) این امکان را فراهم می‌کند که یک یا مجموعه‌ای از کلاس‌ها را به‌طور مستقل طراحی و ایجاد نمود تا در هر زمان که لازم باشد بتوان از آن‌ها در هر برنامه کاربردی استفاده کرد. با این روش می‌توان یک بار کلاس یا کلاس‌های موردنظر را ایجاد نمود و بارها در پروژه‌های مختلف آن‌ها را به‌کار برد و اشیای خود را از آن‌ها ایجاد کرد و از خاصیت‌ها، متدها و رویدادهای آن‌ها استفاده نمود.

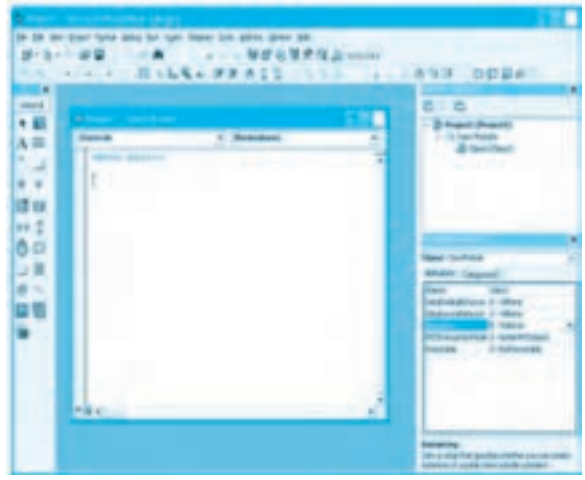
برای ایجاد این نوع از پروژه‌ها به این ترتیب عمل کنید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و در کادرمحاوره New Project آیکن ActiveX DLL را انتخاب کنید

سپس روی دکمه فرمان Open کلیک کنید (شکل ۵۳-۳).

شکل ۵۳-۳

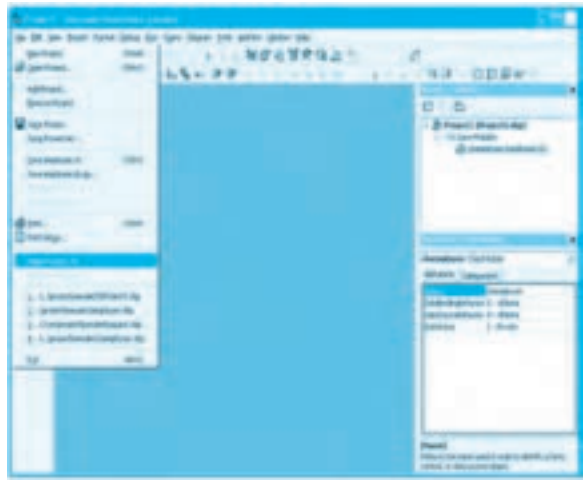
۲- یک پروژه از نوع ActiveX DLL (شکل ۵۴-۳) همراه با یک ماژول کلاس ایجاد می‌شود.



شکل ۵۴-۳

۳- اکنون می‌توان کلاس یا کلاس‌های موردنظر خود را ایجاد نموده و اعضای هریک را تعریف کرد.

۴- پس از طراحی و ساخت کلاس‌ها می‌توانید در پنجره برنامه ویژوال‌بیسیک روی منوی File و سپس گزینه (dll. نام پروژه Make) کلیک کنید (شکل ۵۵-۳) تا فایل DLL ساخته شود.



شکل ۵۵-۳

۵- اکنون می‌توان از فایل DLL تولید شده در سایر پروژه‌ها استفاده کرد.

مثال ۱۶: می‌خواهیم یک پروژه از نوع ActiveX DLL ایجاد کنیم که شامل کلاس clsemployee باشد. برای انجام این مثال این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه جدید از نوع ActiveX DLL ایجاد کنید.

۲- به پنجره ویژوال بروید و روی آیکن Project1 کلیک کنید، سپس نام پروژه را در پنجره خواص روی عبارت employee تنظیم کنید.

۳- دوباره در پنجره پروژه کلیک‌راست کرده و روی گزینه Add و سپس Class Module کلیک کنید.

۴- در کادرمحاوره Add Class Modules روی زبانه Existing کلیک کنید و کلاس clsemployee را در سیستم خود پیدا کرده و آن انتخاب کنید، در پایان روی دکمه فرمان Open کلیک کنید تا کلاس clsemployee به پروژه اضافه شود.

۵- به پنجره ماژول کلاس clsemployee بروید و تمام دستورات آن را با استفاده از گزینه Select All در منوی Edit پنجره ویژوال‌بیسیک انتخاب کنید.

۶- در پنجره ویژوال‌بیسیک روی منوی Edit و سپس گزینه Copy کلیک کنید.

۷- به پنجره ماژول کلاس Class1 بروید و با عمل Paste محتویات کلاس clsemployee را در کلاس Class1 قرار دهید.

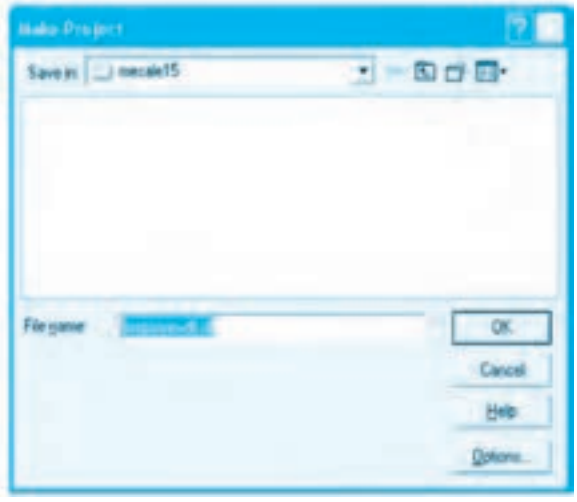
۸- در پنجره پروژه روی آیکن کلاس clsemployee کلیک راست کنید و با استفاده از گزینه Remove employee آن را از پروژه خارج کنید.

۹- در این مرحله در پنجره پروژه آیکن Class1 را انتخاب کنید و خاصیت نام آن را در پنجره روی عبارت clsemployee تنظیم کنید. سپس کلاس جدید را با نام employee ذخیره کنید.

۱۰- پروژه را با نام employeedll ذخیره کنید.

۱۱- در پنجره برنامه ویژوال‌بیسیک ابتدا روی منوی File و سپس گزینه ... Make employeedll.dll کلیک کنید.

یک کادرمحاوره مانند شکل ۵۶-۳ نمایش داده می‌شود، در این کادرمحاوره روی دکمه فرمان OK کلیک کنید تا فایل DLL با نام employeedll.dll در همان پوشه پروژه ایجاد شود.



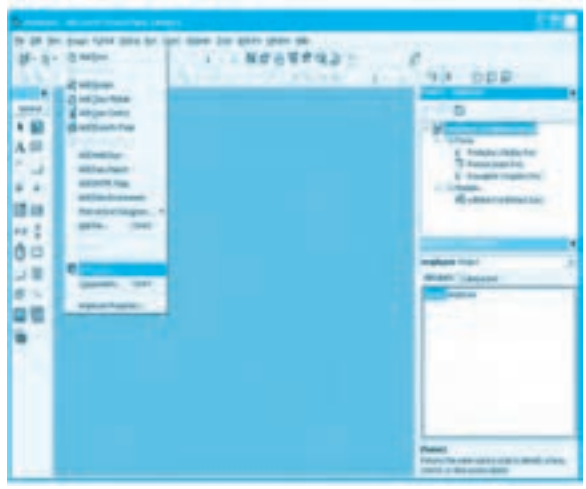
شکل ۵۶-۳

۱۲- پنجره ویژوال‌بیسیک را ببندید.

۱۳- اکنون می‌خواهیم از فایل DLL ایجاد شده در مراحل قبل در یک پروژه دیگر استفاده کنیم، بنابراین پروژه employee را باز کنید.

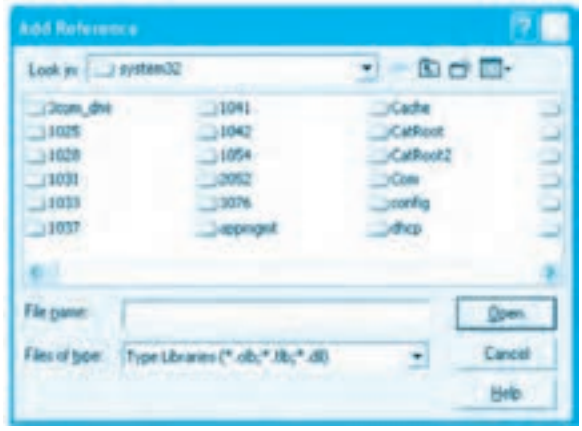
۱۴- در پنجره پروژه روی آیکن کلاس clsemployee کلیک راست کنید و سپس روی گزینه Remove clsemployee کلیک کنید تا کلاس کارمند از پروژه خارج شود.

۱۵- برای استفاده از فایل‌های DLL باید آن‌ها را به پروژه موردنظر اضافه کرد. برای این کار می‌توانید در پنجره ویژوال‌بیسیک روی منوی Project و سپس روی گزینه ... References کلیک کنید (شکل ۳-۵۷).

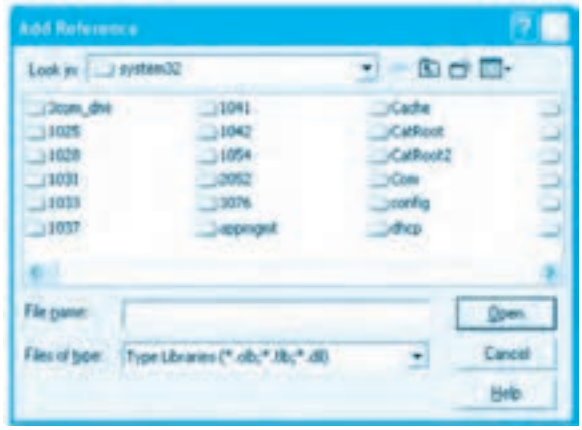


شکل ۳-۵۷

۱۶- در این مرحله کادرمحاوره References نمایش داده می‌شود (شکل ۳-۵۸). در این کادرمحاوره لیستی از فایل‌های DLL که به‌وسیله شرکت مایکروسافت ارائه می‌شوند نمایش داده می‌شوند. روی دکمه Browse کلیک کنید تا کادرمحاوره Add Reference نمایش داده شود (شکل ۳-۵۹).

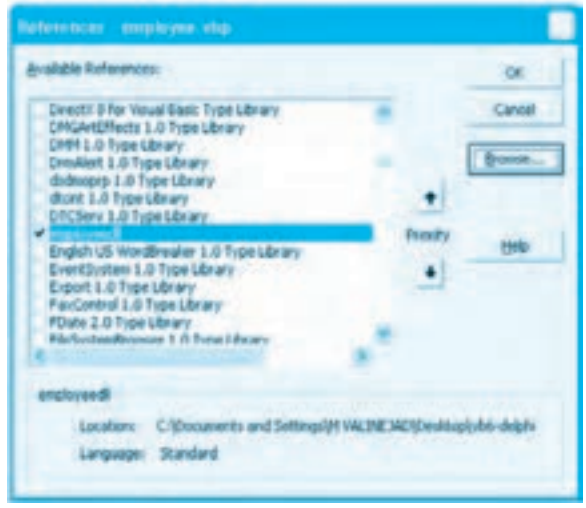


شکل ۳-۵۸



شکل ۳-۵۹

۱۷- با استفاده از کادرمحاوره Add Reference به پوشه پروژه ActiveX DLL خود (employeedll) بروید و فایل employeedll.dll را انتخاب کنید، سپس روی دکمه فرمان Open کلیک کنید. در این مرحله فایل dll در لیست کادرمحاوره References مانند شکل ۳-۶۰ نمایش داده می‌شود.



شکل ۳-۶۰

۱۸- در کادرمحاوره References روی دکمه فرمان OK کلیک کنید. در این مرحله می‌توانید از کلاس موجود در فایل dll استفاده کنید. برای استفاده از کلاس clsemployee می‌توانید به این صورت عمل

کنید:

نام کلاس . نام فایل dll

به عنوان مثال در پروژه employee می‌توان از کلاس clsemployee به این صورت استفاده کرد:

Employeedll.clsemployee

۱۹- به پنجره ماژول فرم‌های frmregister و frmdisplay بروید و دستور

Dim employee As New clsemployee

را به این صورت تغییر دهید:

Dim employee As New employeedll.clsemployee

۲۰- تغییرات را ذخیره کرده و پروژه را اجرا و آزمایش کنید.

۲۱- اجرای برنامه را پایان دهید و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: یک پروژه ActiveX DLL طراحی کنید که شامل کلاس library باشد، سپس از فایل dll که ایجاد می‌شود به جای کلاس موجود در پروژه library استفاده کنید و به‌وسیله آن اشیای موردنیاز را در پروژه ایجاد کرده و استفاده نمایید.

۳-۳-۱۱ نحوه ایجاد کنترل‌های ActiveX

یکی دیگر از انواع پروژه‌هایی که در ویژوال‌بیسیک می‌توان ایجاد نمود نوع ActiveX Control است. ActiveX Control یکی دیگر از ویژگی‌های تکنولوژی COM می‌باشد که این امکان را برای برنامه‌نویسان فراهم می‌آورد تا بتوانند کنترل‌هایی را به‌صورت مجزا و مستقل طراحی و ایجاد کنند، سپس از آن‌ها در سایر پروژه‌ها استفاده نمایند.

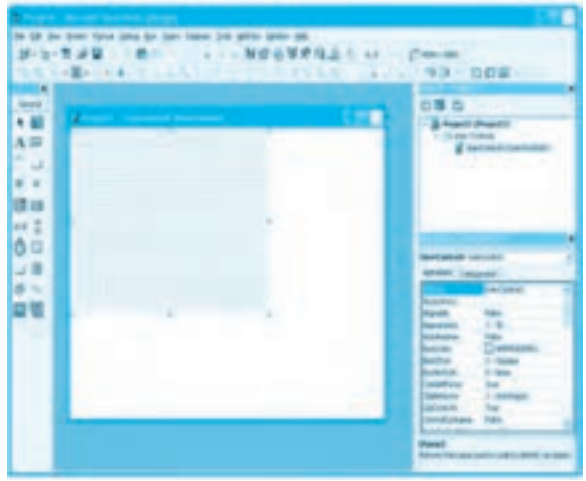
در واقع کنترل‌های ActiveX شبیه به ActiveX DLL هستند با این تفاوت که کنترل‌های ActiveX دارای شکل ظاهری نیز می‌باشند. کنترل‌های ActiveX را می‌توان مانند سایر کنترل‌ها در جعبه ابزار قرار داد و از آن‌ها مانند کنترل‌های ذاتی ویژوال‌بیسیک استفاده کرد. تعداد زیادی از این کنترل‌ها در زمان نصب ویژوال‌بیسیک روی سیستم نصب می‌شوند که می‌توان از آن‌ها به آسانی در هر برنامه کاربردی استفاده نمود. طراحی و ساخت کنترل‌های ActiveX کار مشکل و پیچیده‌ای است اما با استفاده از ابزارهایی که در زبان برنامه‌نویسی ویژوال‌بیسیک ارائه می‌شوند این کار به آسانی امکان‌پذیر است.

برای ایجاد کنترل Actvie‌X به این ترتیب عمل کنید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و در کادرمحاوره New Project آیکن ActiveX Control را انتخاب کنید سپس روی دکمه فرمان Open کلیک کنید (شکل ۶۱-۳).

شکل ۶۱-۳

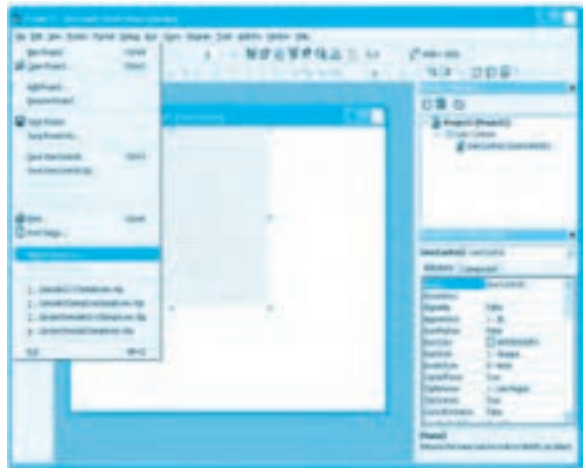
۲- یک پروژه از نوع ActiveX Control مانند شکل ۶۲-۳ ایجاد می‌شود.



شکل ۶۲-۳

در این نوع از پروژه‌ها به جای فرم یا ماژول کلاس، یک کنترل جدید به نام user control در پروژه وجود دارد. این کنترل جدید شبیه به یک فرم بدون حاشیه و کادر اطراف آن می‌باشد، اما این زمینه‌ای است که کنترل جدید روی آن طراحی و ساخته می‌شود. در واقع برای ساخت یک کنترل جدید از یک شئ به نام userControl در ویژوال‌بیسیک استفاده می‌شود.

۳- پس از طراحی و ساخت کنترل جدید می‌توانید در پنجره برنامه ویژوال‌بیسیک روی منوی File و سپس گزینه Make کلیک کنید (شکل ۶۳-۳) تا کنترل جدید به‌صورت یک فایل به نام ocx ساخته شود. نام پروژه ocx.



شکل ۶۳-۳

۴- اکنون می‌توان از فایل ocx تولید شده در سایر پروژه‌ها استفاده کرد.

مثال ۱۷: می‌خواهیم یک کنترل از نوع کادر متن ایجاد کنیم که فقط ارقام ۰ تا ۹ را دریافت می‌کند برای انجام این مثال این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه جدید از نوع ActiveX DLL ایجاد کنید.

۲- به پنجره پروژه بروید و روی آیکن Project1 کلیک کنید، سپس نام پروژه را در پنجره خواص روی عبارت ctlnewtextbox تنظیم کنید.

۳- دوباره به پنجره پروژه بروید و روی آیکن userControl1 کلیک کنید، سپس نام آن را در پنجره خواص روی عبارت newtextbox تنظیم کنید.

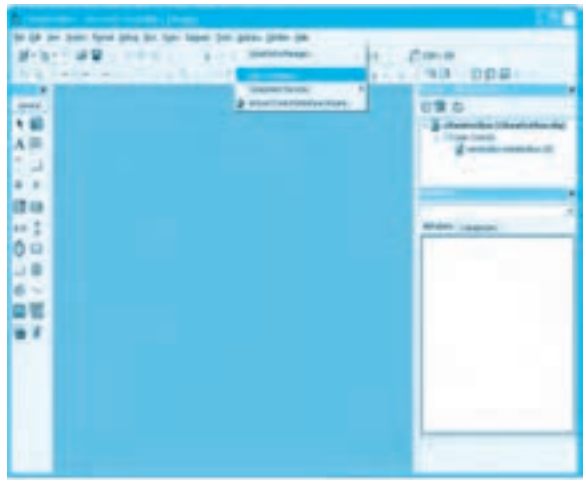
۴- یک کنترل کادر متن با نام txtnumber روی userControl قرار دهید، سپس خاصیت Text آن را خالی کنید. عرض و ارتفاع آن را به ترتیب روی مقادیر ۱۳۰۰ و ۳۰۰ و خاصیت‌های Top و Left آن را نیز روی مقدار صفر تنظیم کنید.

۵- در این مرحله ابعاد userControl را مانند شکل ۳-۶۴ به اندازه کنترل کادر متن کوچک کنید.

شکل ۳-۶۴

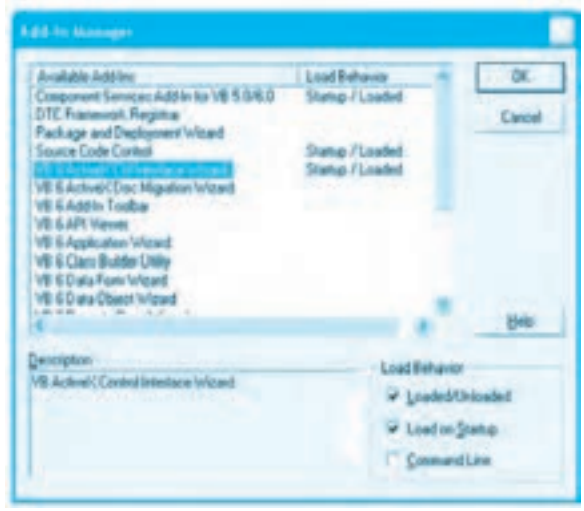
۶- اکنون باید خاصیت‌ها و رویدادها و متدهای کنترل جدید را تنظیم کرده و کنترل جدید را کامل کنید.

برای این کار ابتدا باید ابزار ساخت کنترل ActiveX را در ویژوال‌بیسیک فعال کنید، بنابراین در پنجره برنامه ویژوال‌بیسیک روی منوی Add-Ins و سپس گزینه... Add-In Manager کلیک کنید (شکل ۳-۶۵).



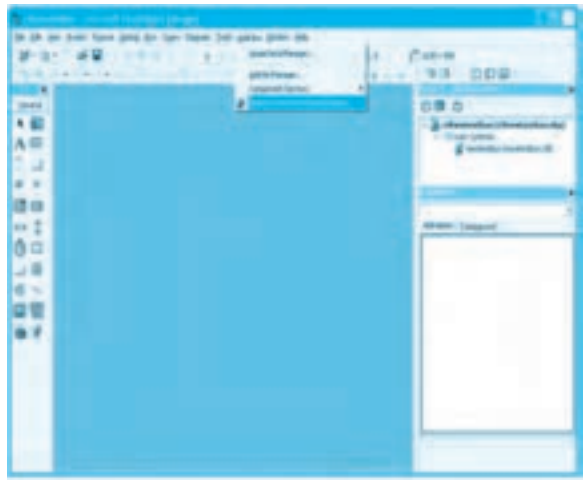
شکل ۳-۶۵

۷- کادرمحاوره Add-In Manager نمایش داده می‌شود. در لیست Available Add-Ins گزینه VB 6 ActiveX Ctrl Interface Wizard را انتخاب کنید، سپس روی کادرهای علامت Loaded/Un loaded و Load on Startup کلیک کنید تا در حالت انتخاب قرار گیرند (شکل ۶۶-۳).



شکل ۶۶-۳

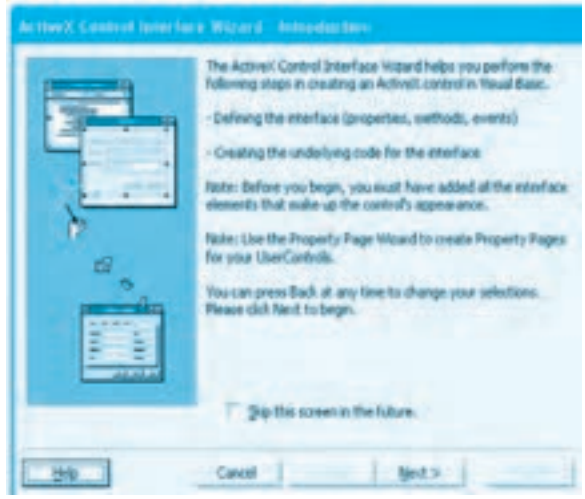
انتخاب کادر علامت Loaded/Un loaded سبب می‌شود تا پس از بسته شدن این کادرمحاوره ابزار ساخت کنترل‌های ActiveX به منوی Add-Ins اضافه شود و انتخاب کادر علامت Load on Startup سبب می‌شود تا در اجرای دوباره برنامه ویژوال‌بیسیک این ابزار به‌طور خودکار به منوی Add-Ins اضافه شود.



شکل ۶۷-۳

۸- در پنجره ویژوال‌بیسیک روی منوی Add-Ins و سپس روی گزینه ... ActiveX Control Interface Wizard کلیک کنید (شکل ۶۷-۳).

۹- در این مرحله یک کادرمحاوره مانند شکل ۶۸-۳ نمایش داده می‌شود که ابزار ساخت کنترل ActiveX را معرفی می‌کند. اگر می‌خواهید این کادرمحاوره را در دفعات بعد مشاهده کنید کادر علامت Skip this screen in the future را انتخاب نمایید.



شکل ۶۸-۳

شکل ۶۹-۳

۱۰- روی دکمه Next کلیک کنید تا به مرحله بعد بروید، در مرحله بعد کادرمحاوره دیگری مانند شکل ۳-۶۹ نمایش داده می‌شود.

در کادرمحاوره Select Interface Members دو کادر لیست وجود دارد: در کادر لیست Available names نام خاصیت‌ها، متدها و رویدادهایی که ممکن است برای کنترل جدید لازم باشد نمایش داده می‌شوند. در کادر لیست Selected names خاصیت‌ها، متدها و رویدادهایی که برای کنترل جدید استفاده می‌شوند نمایش داده می‌شوند. به‌طور پیش‌فرض بعضی از این اعضا برای کنترل جدید انتخاب شده و در کادر لیست Selected names قرار می‌گیرند.

البته شما می‌توانید با دکمه > یک عضو از کادر لیست Available names را به کادر لیست Selected names انتقال دهید یا با دکمه >> کلیه اعضا را منتقل کنید. دکمه‌های < و << عمل انتقال یک یا کلیه اعضا را از کادر لیست Selected names به کادر لیست Available names انجام می‌دهند.

۱۱- در این مرحله خاصیت MaxLength را از کادر لیست Available names انتخاب کرده و سپس روی دکمه > کلیک کنید تا خاصیت MaxLength برای کنترل جدید انتخاب شود.

۱۲- در کادر لیست Selected names خاصیت BackStyle را انتخاب کرده و روی دکمه < کلیک کنید تا این خاصیت از کادر لیست Selected names خارج شود.

۱۳- اکنون در کادر لیست Selected names روی رویداد MouseDown کلیک کنید و سپس کلید Shift را پایین نگه داشته و روی متد Refresh کلیک کنید، در این صورت سه رویداد مربوط به ماوس و متد Refresh انتخاب می‌شوند.

شکل ۳-۷۰

روی دکمه < کلیک کنید تا این اعضا از کادر لیست Selected names خارج شوند.

۱۴- در این مرحله روی دکمه Next کلیک کنید تا به مرحله بعد بروید، در مرحله بعد کادرمحاوره Create Custom Interface Members مانند شکل ۳-۷۰ نمایش داده می‌شود. در این کادرمحاوره می‌توان اعضای جدیدی را برای کنترل جدید تعریف کنید.

روی دکمه Next کلیک کنید تا به مرحله بعد بروید.

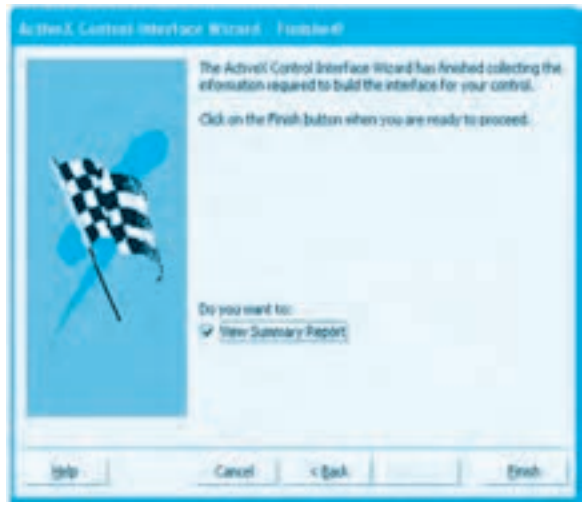
۱۵- اکنون کادرمحاوره Set Mapping نمایش داده می‌شود (شکل ۳-۷۱).

شکل ۳-۷۱

در این کادرمحاوره می‌توان مشخص کرد اعضای انتخاب شده در مرحله قبل برای کدام کنترل از مجموعه کنترل‌های موجود در کنترل جدید مورد استفاده قرار گیرند. در کادر لیست Public Name روی خاصیت BackColor کلیک کنید.

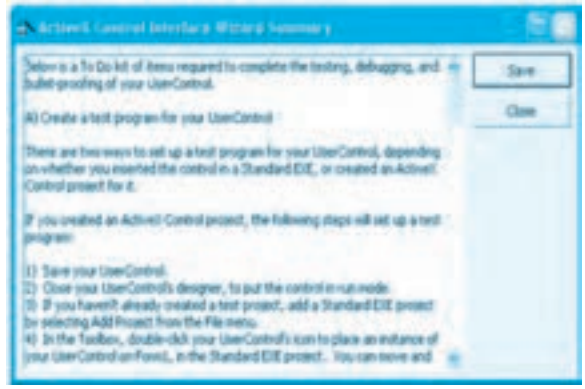
سپس به انتهای کادر لیست رفته، کلید Shift را پایین نگه داشته و روی خاصیت MaxLength کلیک کنید و بعد کادر لیست ترکیبی بازشو Control را باز کرده و نام کنترل کادر متن، یعنی txtnumber را انتخاب کنید، با این عمل کلیه خاصیت‌ها، متدها و رویدادها در اختیار کنترل کادر متن txtnumber قرار می‌گیرند.

۱۶- در این مرحله عملیات ساخت کنترل جدید خاتمه یافته و کادرمحاوره‌ای مانند شکل ۳-۷۲ نمایش داده می‌شود.



شکل ۳-۷۲

اگر در این کادرمحاوره کادر View Summary Report انتخاب شود پس از آن‌که روی دکمه فرمان Finish کلیک کنید، یک پنجره مانند شکل ۳-۷۳ یک راهنما در رابطه با چگونگی تست و آزمایش کنترل جدید در سایر انواع پروژه‌ها نمایش می‌دهد.



شکل ۳-۷۳

۱۷- روی دکمه فرمان Finish کلیک کنید و در صورت نیاز در پنجره Summary روی دکمه فرمان Save کلیک کنید و مطالب ارائه شده را در یک فایل متنی ذخیره کنید. در غیراین‌صورت روی دکمه فرمان Close کلیک کنید تا عملیات خاتمه یابد.

۱۸- اکنون باید کنترل جدید را طوری تنظیم کنیم که فقط ارقام صفر تا ۹ را بپذیرد. بنابراین در پنجره پروژه روی

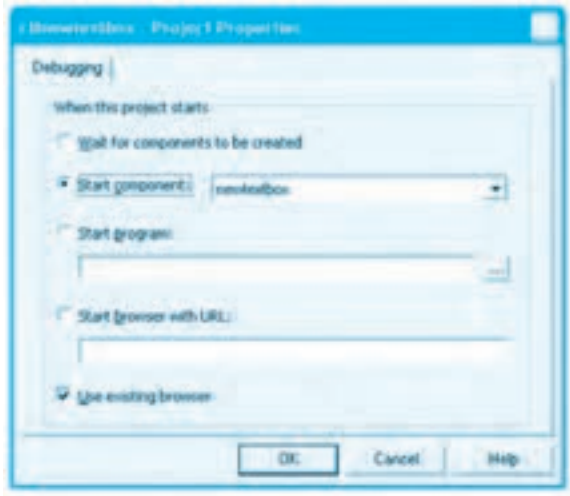
آیکن کنترل جدید (newtextbox) و سپس روی دکمه View Code کلیک نمایید تا پنجره کدنویسی نمایش داده شود. به پنجره کدنویسی بروید و رویداد keypress کنترل کادر متن txtnumber را به این صورت تنظیم کنید:

```
Private Sub txtumber_KeyPress(KeyAscii As Integer)
  If KeyAscii >= 48 And KeyAscii <= 57 or  KeyAscii = 8 Then
     Exit Sub
  Else
     KeyAscii = 0
  End If
End Sub
```

در این رویداد با استفاده از مقدار آرگومان KeyAscii کلید فشرده شده در کادر متن txtnumber به‌وسیله دستور شرطی If بررسی می‌شود.

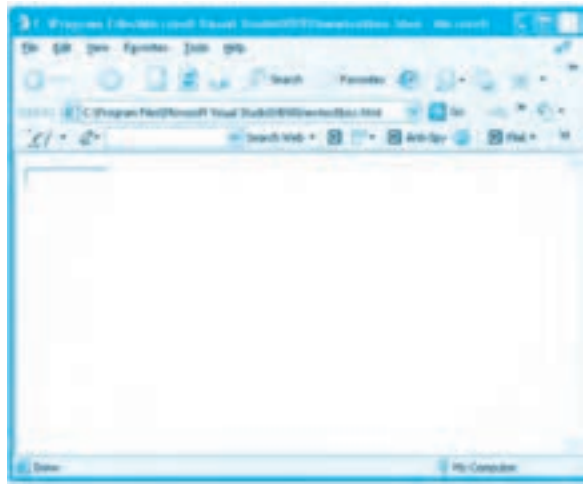
اگر کد اسکی کلید فشرده شده در محدوده کاراکترهای رقمی (۴۸ تا ۵۷) یا کلید BackSpace باشد دستور Exit Sub اجرا می‌شود و بدون این‌که کاری انجام شود اجرای برنامه از رویه رویداد KeyPress خارج می‌شود، اما اگر کاراکترهایی غیر از این کاراکترها تایپ شوند بخش Else دستور If کد اسکی کلید را روی صفر تنظیم می‌کند، در نتیجه کاراکتر تایپ شده دریافت نمی‌شود.

۱۹- تغییرات را ذخیره کرده و پروژه را اجرا کنید. در این مرحله کادرمحاوره‌ای مانند شکل ۳-۷۴ نمایش داده می‌شود.



شکل ۳-۷۴

روی دکمه فرمان OK کلیک کنید. اکنون پروژه اجرا شده و کنترل در پنجره مرورگر اینترنت Internet Explorer نمایش داده می‌شود (شکل ۷۵-۳).



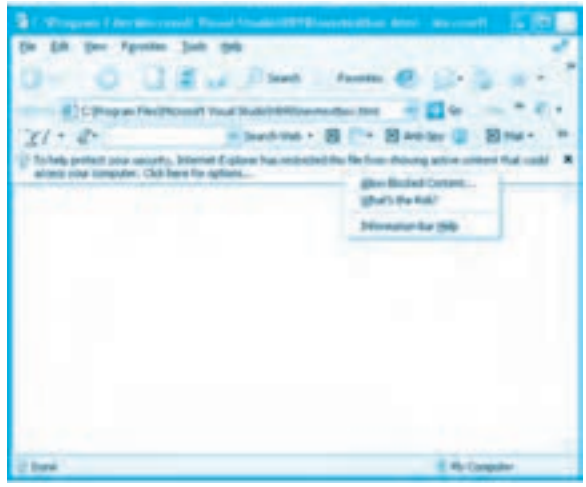
شکل ۷۵-۳

اکنون می‌توانید روی کنترل کادر متن کلیک کرده و آن را آزمایش کنید. چند کاراکتر حرفی و رقمی و سایر کلیدها را فشار دهید و نتیجه را بررسی کنید.

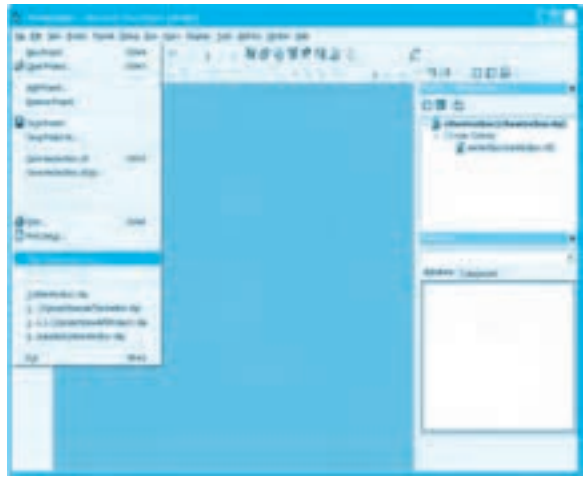
پس از این‌که کنترل را آزمایش و بررسی کردید پنجره مرورگر اینترنت (IE) را ببندید و به پنجره ویژوال‌بیسیک بازگردید و در منوی Run روی گزینه End کلیک کنید تا اجرای پروژه خاتمه یابد.

نکته: اگر هنگام اجرای پروژه ActiveX Control در پنجره مرورگر اینترنت (IE) با پیامی مانند شکل ۷۶-۳ برخورد کردید روی پیام هشداری که نمایش داده می‌شود کلیک راست کنید و روی گزینه ... Allow Blocked Content کلیک کنید و در کادرمحاوره‌ای که نمایش داده می‌شود روی دکمه فرمان Yes کلیک کنید تا اجرای پروژه کامل شود و کنترل در پنجره مرورگر اینترنت (IE) نمایش داده شود.

۲۰- در این مرحله می‌توانید کنترل ActiveX Control خود را برای استفاده در سایر برنامه‌ها کامل کنید، بنابراین در پنجره ویژوال‌بیسیک روی منوی File و سپس گزینه ... Make ctlnewtextbox.ocx کلیک کنید (شکل ۷۷-۳).



شکل ۳-۷۶

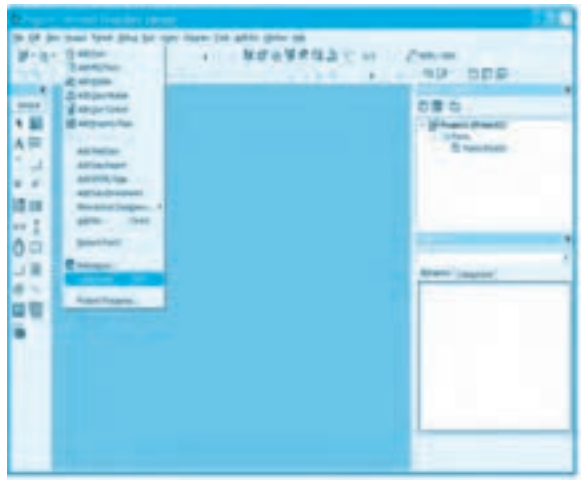


شکل ۳-۷۷

کادرمحاوره Make Project مانند شکل ۳-۷۸ نمایش داده می‌شود، در صورت تمایل می‌توانید نام دیگری را برای کنترل جدید انتخاب کنید، سپس روی دکمه فرمان OK کلیک کنید تا کنترل جدید با پسوند ocx در مسیر تعیین شده ساخته شود.

شکل ۳-۷۸

پس از ساخت کنترل ActiveX می‌توانید از آن در سایر پروژه‌ها مانند پروژه‌هایی از نوع Standard EXE استفاده کنید. برای انجام این کار پس از ایجاد پروژه اصلی (به عنوان مثال Standard EXE) در پنجره ویژوال‌بیسیک روی منوی Project و سپس گزینه ... Components کلیک کنید (شکل ۳-۷۹).

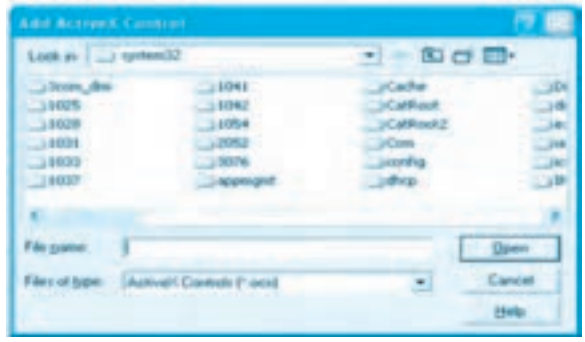


شکل ۳-۷۹

کادرمحاوره‌ای با نام Components نمایش داده می‌شود (شکل ۳-۸۰).

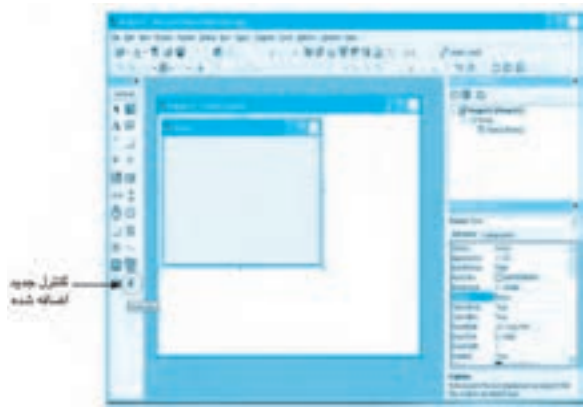
شکل ۸۰-۳

در این کادرمحاوره روی دکمه فرمان ...Browse کلیک کنید. کادرمحاوره Add ActiveX Control نمایش داده می‌شود (شکل ۸۱-۳).



شکل ۸۱-۳

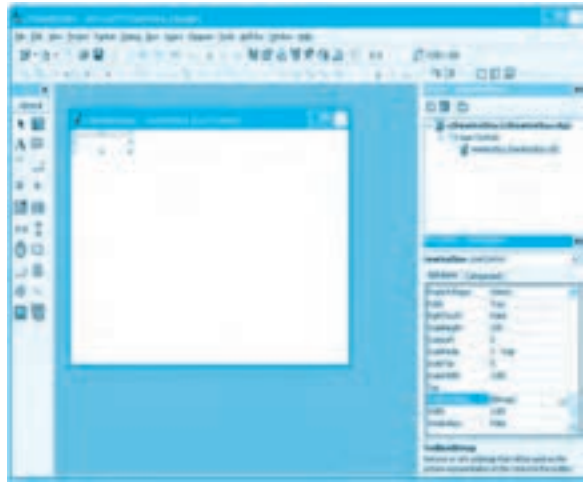
به‌وسیله این کادرمحاوره کنترل جدیدی را که ساخته‌اید در پوشه پروژه ctlnewtxtbox پیدا کرده و روی دکمه فرمان OK کلیک کنید تا کنترل جدید به جعبه ابزار اضافه شود (شکل ۸۲-۳).



شکل ۸۲-۳

اکنون می‌توانید از آن روی هر فرمی که مایل هستید استفاده کنید و خاصیت‌ها و رویدادهای آن را مانند سایر کنترل‌ها تنظیم کنید.

تمرین: یک کنترل کادر متن طراحی کنید که فقط کاراکترهای حرفی را دریافت کند، سپس از کنترل جدید در یک پروژه Standard EXE استفاده کنید.



شکل ۸۳-۳

نکته: اگر بخواهید می‌توانید برای کنترل خود به جای آیکنی که به‌طور پیش‌فرض استفاده شده و در جعبه ابزار نمایش داده می‌شود از یک آیکن دلخواه استفاده کنید. برای این کار لازم است یک فایل از نوع bmp یا jpg با اندازه ۱۵×۱۶ پیکسل تهیه کنید و فایل آیکن را در خاصیت ToolboxBitmap مربوط به UserControl انتخاب کنید (شکل ۸۳-۳). در این صورت پس از اضافه کردن کنترل جدید به پروژه‌های دیگر، آیکن انتخاب شده برای آن در جعبه ابزار ویژوال‌بیسیک نمایش داده می‌شود.

Learn in English

Class Modules

Class modules (.CLS file name extension) are the foundation of object-oriented programming in Visual Basic.

You can write code in class modules to create new objects.

These new objects can include your own customized properties and methods.

Creating Property Procedures

As mentioned previously, public module-level variables in a class module function as properties of an object. However, they>re not very sophisticated. If you need to run code in order to set or return a property>s value, or you want to make a property read-only, you can create a Property procedure. There are three types of Property procedures: Property Get, Property Let, and Property Set procedures. The Property Get procedure returns the current value of a property, whereas the Property Let procedure sets the value. The Property Set procedure assigns an object to an object property.

To create a read-write property, you need to include a pair of Property procedures in the class module. Both procedures must have the same name. If the property stores and returns a scalar value, such as a numeric, text, or date value, you use a Property Let procedure to set the value and a Property Get procedure to retrieve it. If the property stores and returns a reference to an object, you use the Property Set procedure to store the reference and the Property Get procedure to return it.

واژه‌نامه

مساحت	Area
مرورگر	Browser
سازنده	Builder
کلاس	Class
ماژول کلاس	Class Module
کلکسیون	Collection
مؤلفه، جزء	Component
سفارشی کردن	Customize
تعریف	Definition
کپسوله کردن، مخفی کردن	Encapsulation
رویداد	Event
فیلد	Field
بنیاد، اساس	Foondation
مقداردهی در آغاز	Initialize
عضو	Member
اشاره کردن، ذکرکردن	Mention
متد	Method
مدل، طرح	Model
شئ	Object
برنامه‌نویسی به روش شئ‌گرا	Object oriented programming
خاصیت، ویژگی	Property

مستطیل	Rectangle
مرجع، ارجاع	Reference
مربع	Square
ساخت‌یافته	Structural
خلاصه، مختصر	Summary
به پایان رساندن، خاتمه دادن	Terminate
هشدار، اخطار	Warning

خلاصه مطالب

- روش برنامه‌نویسی شئ‌گرا یا oop روش جدیدی در برنامه‌نویسی است که مشکلات برنامه‌نویسی ساخت یافته را برطرف می‌کند.
- اجزای اصلی در برنامه‌نویسی شئ‌گرا عبارتند از: شئ و کلاس.
- کلاس، یک قالب و الگو برای تولید سریع‌تر و راحت‌تر اشیا می‌باشد.
- یک کلاس شامل مجموعه‌ای از داده‌ها، اطلاعات، دستورالعمل‌ها و رفتارها می‌باشد.
- شئ نمونه‌ای است که از روی کلاس ایجاد می‌شود و تمام ویژگی‌ها و رفتارهای آن را کسب می‌کند.
- به هریک از اجزای تشکیل‌دهنده کلاس مانند فیلد، خاصیت، متد و رویداد، یک عضو کلاس می‌گویند.
- فیلدها و خاصیت‌ها اعضایی از کلاس هستند که وظیفه نگهداری داده و اطلاعات را بر عهده دارند.
- متدها و رویدادها اعضایی از کلاس هستند که وظیفه اجرای دستورالعمل‌ها را بر عهده دارند.
- گردآوری و ایجاد مجموعه‌ای از اعضا در یک بخش واحد به نام کلاس را کپسوله کردن یا مخفی کردن اعضا می‌گویند.
- برای ایجاد یک کلاس جدید از ماژول کلاس استفاده می‌شود.
- رویداد Initializer در زمان ایجاد یک شئ از کلاس اجرا می‌شود.
- رویداد Terminate زمانی اجرا می‌شود که شئ ایجاد شده از بین می‌رود.
- ماژول کلاس شبیه به ماژول کد است و شکل ظاهری نداشته و اجزایی مانند کنترل‌ها را در برنمی‌گیرد.
- برای ایجاد ماژول کلاس در نوار منوی پنجره ویژوال‌بیسیک روی منوی Project و سپس روی گزینه Add Class Module کلیک کنید.

- پسوند فایل‌های ماژول کلاس cls است.
- یک شئ را می‌توان در یک ماژول کد یا ماژول فرم یا در یک ماژول کلاس تعریف کرد.
- برای ایجاد یک شئ از یک کلاس از این دستورات استفاده می‌شود:

Dim یا Private یا Public نام شئ As نام کلاس

Set نام شئ = New نام کلاس

- برای تعریف عضو فیلد در یک کلاس می‌توانید در بخش تعاریف ماژول کلاس از این دستور استفاده کنید:

Public نام فیلد As نوع داده

- برای تعریف عضو خاصیت در یک کلاس از رویه‌های Property Get و Property Let استفاده می‌شود.
- برای تعریف عضو متد در یک کلاس می‌توانید از رویه‌های فرعی یا تابعی استفاده کنید.
- برای ایجاد یک کلاس جدید و اعضای آن می‌توانید از Class Builder استفاده کنید.
- با استفاده از مرورگر شئ Object Browser می‌توانید کلاس‌های موجود در یک پروژه و ویژوال‌بیسیک و اعضای آن‌ها را مشاهده کنید و اطلاعاتی در رابطه با آن‌ها به‌دست آورید و به محل تعریف آن‌ها دسترسی پیدا کنید.
- برای ذخیره، نگهداری و مدیریت مجموعه‌ای از اشیا می‌توان از مفهوم آرایه یا کلکسیون استفاده کرد.
- برای ایجاد یک کلکسیون می‌توان از کلاس Collection استفاده نمود.
- متدهای Add و Remove می‌توانند اشیا را به یک کلکسیون اضافه کنند یا خارج نمایند.
- با استفاده از خاصیت Count در یک کلکسیون می‌توان تعداد اشیای موجود در کلکسیون را به‌دست آورد.
- برای دسترسی به اشیای موجود در یک کلکسیون می‌توان از حلقه For یا For Each استفاده کرد.
- برای دسترسی ساده‌تر به مجموعه‌ای از خاصیت‌های یک شئ می‌توان از دستور With ... End With استفاده کرد.
- با پروژه‌های نوع ActiveX DLL می‌توان یک یا مجموعه‌ای از کلاس‌ها را به‌صورت فایل‌های dll مستقل ایجاد و در سایر پروژه‌ها استفاده کرد.
- با پروژه‌های نوع ActiveX Control می‌توان کنترل‌های جدید ایجاد کرد و در سایر پروژه‌ها استفاده نمود.
- پسوند فایل‌های ActiveX Control، ocx می‌باشد.

آزمون نظری

۱- کدام رویه در زمان خواندن مقدار یک خاصیت اجرا می‌شود؟

الف- Property Let ب- Property Set ج- Property Get د- Property

۲- کدام نوع از انواع رویه‌ها برای تعریف متدها استفاده می‌شود؟

الف- رویه‌های تابعی
ب- رویه‌های تابعی و فرعی
ج- رویه‌های فرعی
د- رویه‌های فرعی و تابعی بدون آرگومان

۳- کدام رویداد در زمان ایجاد یک شئ از کلاس اجرا می‌شود؟

الف- Terminate ب- Initial ج- Initialize د- Load

۴- کدام رویه در زمان مقداردهی به یک خاصیت اجرا می‌شود؟

الف- Property ب- Property Let ج- Property Get د- Set

۵- کدام برنامه برای ایجاد کلاس‌ها و اعضای آن‌ها استفاده می‌شود؟

الف- Object Browser
ب- Class Builder
ج- Class Module
د- Class Browser

۶- کدام رویداد در زمان از بین رفتن یک شئ اجرا می‌شود؟

الف- End Class ب- unLoad ج- Terminated د- Terminate

۷- کدام نوع از پروژه‌ها در ویژوال‌بیسیک امکان ایجاد کنترل‌های جدید را فراهم می‌کند؟

الف- ActiveX DLL
ب- ActiveX Control
ج- User Control
د- ActiveX

۸- کدام خاصیت در کلکسیون‌ها تعداد اشیا را در کلکسیون مشخص می‌کنند؟

الف- Count ب- Len ج- Length د- Counter

۹- با کدام دستور امکان دسترسی به خاصیت‌های یک شئ آسان‌تر می‌شود؟

الف- For Each
ب- With
ج- With … End With
د- Select Case

۱۰- کدام‌یک از انواع حلقه برای دسترسی به اشیای ذخیره شده در کلکسیون‌ها مناسب‌تر است؟

الف- For ب- While ج- For Each د- Do While

11- ..... is the Foundation of object oriented Programming in the visual Basic.

a- Module **b**- Class **c**- Class Module **d**- Collection

۱۲- برنامه‌نویسی به روش شئ‌گرا را توضیح دهید و تفاوت آن را با برنامه‌نویسی به روش ساخت

یافته بیان کنید.

۱۳- مفهوم کلاس و شئ را با ذکر مثال توضیح دهید.

۱۴- مفهوم کپسوله کردن را توضیح دهید.

۱۵- نحوه ایجاد یک کلاس جدید را توضیح دهید.

۱۶- نحوه ایجاد شئ از یک کلاس را توضیح دهید.

۱۷- تفاوت بین عضو فیلد و خاصیت را بیان کنید.

۱۸- تفاوت بین عضو متد و رویداد را بیان کنید.

۱۹- نحوه ایجاد عضو فیلد و خاصیت را با ذکر مثال توضیح دهید.

۲۰- نحوه ایجاد عضو متد را با ذکر مثال توضیح دهید.

۲۱- نحوه ایجاد عضو رویداد را با ذکر مثال توضیح دهید.

۲۲- تفاوت بین رویه Property Let و Property Get را توضیح دهید.

۲۳- رویدادهای کلاس را توضیح دهید و کاربرد هریک را بیان کنید.

۲۴- کاربرد Class Builder را بیان کرده و نحوه استفاده از آن را توضیح دهید.

۲۵- کاربرد مرورگر شئ Object Browser را بیان کرده و نحوه استفاده از آن را توضیح دهید.

۲۶- مفهوم کلکسیون و کاربرد آن را بیان کرده و متدها و خاصیت‌های آن را با ذکر مثال توضیح دهید.

۲۷- نحوه استفاده و کاربرد دستور With ... End With را با ذکر مثال توضیح دهید.

۲۸- ذخیره و نگهداری اشیا با استفاده از آرایه را توضیح دهید.

۲۹- مفهوم تکنولوژی Com و Comt را توضیح داده و دلیل استفاده از آن‌ها را بیان کنید.

۳۰- نحوه ایجاد پروژه‌های ActiveX DLL و ActiveX Control را توضیح داده و کاربرد هریک را بیان کنید.

۳۱- نحوه عملکرد دستور For Each را توضیح داده و تفاوت آن را با حلقه For بیان کنید.

آزمون عملی

۱- یک پروژه طراحی کنید که با استفاده از یک کلاس بتواند داده‌ها و اطلاعات مربوط به کارگران یک کارخانه را براساس جدول ۳-۳ دریافت و مدیریت نماید.

الف- امکان دریافت و ذخیره‌سازی اطلاعات هر کارمند وجود داشته باشد.

ب- با استفاده از یک متد بتوان حقوق دریافتی یک کارگر را با استفاده از این فرمول محاسبه کرد.

بیمه - مالیات - حقوق خالص = حقوق دریافتی

ج- با استفاده از یک متد بتوان میزان مالیات حقوق را براساس جدول ۳-۴ محاسبه کرد.

د- امکان مشاهده فیش حقوقی هر کارگر به وسیله شماره پرسنلی وی وجود داشته باشد.

جدول ۳-۳

نام	حقوق خالص
نام خانوادگی	تاریخ استخدام
شماره شناسنامه	وضعیت تأهل
تاریخ تولد	شماره پرسنلی
جنسیت	سمت

جدول ۳-۴

مالیات	حقوق خالص
صفر	**۴۰۰۰۰۰۰ ریال**
۲ درصد	از **۴۰۰۰۰۰۰** ریال تا **۵۰۰۰۰۰۰** ریال
۳ درصد	از **۵۰۰۰۰۰۰** ریال تا **۶۰۰۰۰۰۰** ریال
۴ درصد	از **۶۰۰۰۰۰۰** ریال تا **۷۰۰۰۰۰۰** ریال
۷ درصد	از **۷۰۰۰۰۰۰** ریال و بالاتر

۲- یک کلاس ایجاد کنید که به‌وسیله آن بتوان مساحت و محیط هر مثلث دلخواهی را محاسبه

نمود.

۳- یک کنترل زمان‌سنج طراحی کنید که شامل یک کنترل برچسب و دو کنترل دکمه فرمان با عنوان‌های Start و Stop باشد. با کلیک روی دکمه Start محاسبه زمان از صفر آغاز شده و در کنترل برچسب نمایش داده می‌شود و با کلیک روی دکمه Stop محاسبه زمان خاتمه می‌یابد. سپس کنترل جدید را در یک پروژه Standard EXE مورد استفاده قرار دهید.

۴- یک پروژه از نوع ActiveX DLL طراحی و ایجاد کنید که شامل یک کلاس جهت ذخیره‌سازی و نگهداری اطلاعات اتومبیل‌هایی که در یک پارکینگ عمومی وارد و خارج می‌شوند باشد. در این کلاس باید بتوان ساعت ورود، ساعت خروج و هزینه پارکینگ هر اتومبیل را ثبت و محاسبه نمود. سپس از فایل dll به‌دست آمده در یک پروژه از نوع Standard EXE استفاده کنید.

۵- یک کنترل جدید طراحی کنید یک چراغ راهنمایی را با سه رنگ سبز، زرد و قرمز شبیه‌سازی کند.

واحد کار چهارم

هدف جزیی

توانایی خطایابی، خطازدایی و رفع اشکال برنامه‌ها

زمان (ساعت)	
نظری	عملی
۳	۶

هدف‌های رفتاری

پس از مطالعه این واحد کار از فراگیر انتظار می‌رود که:

۱- انواع خطاها و دلیل ایجاد آن‌ها را توضیح دهید.

۲- بتواند خطاهای نوشتاری و منطقی را شناسایی و برطرف کند.

۳- بتواند خطاهای زمان اجرا را با دستور On Error GoTo مدیریت کند.

۴- حالت توقف و ویژگی‌های آن را بداند.

۵- روش‌های ایجاد حالت توقف و خروج از آن را بداند.

۶- بتواند پنجره اجرای فوری دستورات (Immediate Window) را فعال کند و از آن استفاده نماید.

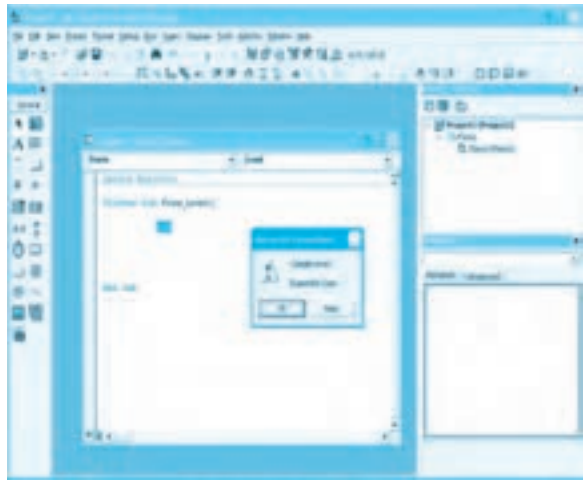
۷- بتواند از Debug Object و متدهای آن استفاده کند.

۸- بتواند از امکانات موجود در منوی Debag برای خطایابی و خطازدایی برنامه‌ها استفاده کند.

کلیات

یکی از مواردی که در پروژه‌های برنامه‌نویسی از اهمیت بسزایی برخوردار است، خطایابی و برطرف کردن خطاهای برنامه می‌باشد. علاوه بر این، برنامه باید توانایی مقابله با خطاهایی را که در هنگام اجرای برنامه توسط کاربران رخ می‌دهد نیز داشته باشد. زبان برنامه‌نویسی ویژوال‌بیسیک در هر دو زمینه امکانات مناسبی در اختیار برنامه‌نویسان قرار می‌دهد. به طور کلی خطاها به سه دسته کلی تقسیم می‌شوند: خطاهای نوشتاری، خطاهای منطقی و خطاهای زمان اجرا.

خطاهای نوشتاری (Syntax Errors) می‌توانند در اثر عدم دقت برنامه‌نویس در زمان تایپ و نوشتن دستورالعمل‌ها ایجاد شوند. ویژوال‌بیسیک این نوع خطاها را بلافاصله پس از تایپ دستورات و فشردن کلید Enter تشخیص می‌دهد و با نمایش کادرهای پیغام و نمایش راهنمایی‌های لازم دستور مربوطه را نیز با رنگ قرمز مشخص می‌کند. این نوع خطاها باید قبل از اجرای برنامه برطرف شوند. به عنوان مثال اگر هنگام استفاده از دستور Select Case به اشتباه آن را به صورت Select Cse تایپ کنید و کلید Enter را بفشارید کادرمحاوره‌ای مانند شکل ۱-۴ نمایش داده می‌شود و اشتباه نوشتاری را اطلاع می‌دهد. در این صورت می‌توانید با کلیک روی دکمه فرمان Help از راهنمای MSDN ویژوال‌بیسیک در صورت نصب آن روی سیستم استفاده کنید یا با کلیک روی دکمه فرمان OK به محل دستور و محلی که خطا وجود دارد، بروید.



شکل ۱-۴

نکته: اگر خطاهای نوشتاری در برنامه برطرف نشود، در هنگام اجرای برنامه خطای ترجمه (Compile error) نمایش داده می‌شود و دستوری که خطا را به وجود آورده است، نمایش داده می‌شود (شکل ۱-۴).

خطاهـای منطقی (Logical Errors) نـاشی از عـدم طـراحی درست برنامه یـا اشـتباه در طـراحی روند منطقی اجـرای برنـامه است. به عنوان مثال اگر اولویت عملگرها در عبارت‌هـای محاسـباتی یا شـرطها رعایت نشـود، نتیجه محاسـبات غیرواقعی بوده یـا بررسـی شـرطهای اسـتفاده شـده نتایج نادرسـتی را بـه وجـود می‌آورند کـه رونـد اجـرای بـرنامه را بـا مشـکل مـواجه می‌کـند یـا عـدم رعـایت قوانین تبدیل داده‌هـا از یک نـوع بـه نوع داده دیگر سبب ایجـاد نتایج غیرقابل پیش‌بینی در بـرنامه‌ها می‌شـود.

نوع دیگر خطاها، یعنی خطاهای زمان اجرا (Runtime Errors) پس از طراحی نرم‌افزار و در زمان استفاده از آن پدیدار می‌شوند. به عنوان مثال می‌توان به خطاهایی که ممکن است در اثر استفاده از حافظه جانبی و دستگاه‌های ورودی، خروجی رخ دهد، اشاره کرد؛ مانند خطای نوشتن یا خواندن روی دیسک معیوب یا ارسال چاپ به یک چاپگر که خاموش بوده یا فاقد کاغذ است.

در چنین مواردی اجرای برنامه به طور ناگهانی قطع شده و کاربر بدون آن‌که اطلاع کافی داشته باشد از برنامه خارج می‌شود.

ویژوال‌بیسیک برای شناسایی و برطرف کردن سه نوع از خطاها امکانات و دستورات مناسبی را ارائه می‌کند.

۴-۱ مدیریت خطاهای زمان اجرا به وسیله دستور On Error GoTo

در ویژوال‌بیسیک برای مدیریت خطاهایی که در زمان اجرای برنامه رخ می‌دهد از دستور On Error GoTo استفاده می‌شود. به عنوان مثال فرض کنید در یک پروژه از نوع Standard EXE مانند شکل ۲-۴ با دو کنترل دکمه فرمان و کادر متن، رویه رویداد Click برای دکمه فرمان Save به این صورت تنظیم شده است:

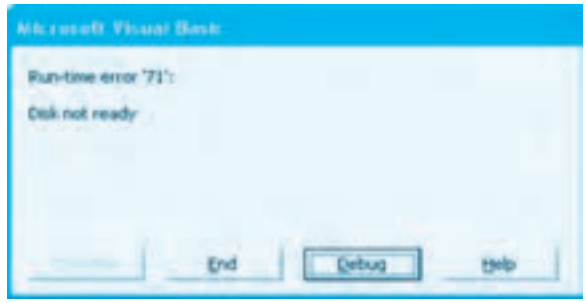
شکل ۲-۴

```
Private Sub cmdsave_Click()
   intrecno = intrecno + 1
   Open "A:\numbers.dat" For Random Access Write As 1 Len = 4
   Dim sngno As Single
   sngno = Val(txtno.Text)
   Put #1, intrecno, sngno
   Close #1
End Sub
```

نکته: می‌توانید از پروژه numbers که در واحدکار دوم ایجاد کرده‌اید، استفاده نمایید.

در صورتی‌که کاربر روی دکمه فرمان cmdsave کلیک کند، دستورات ایجاد و نوشتن در فایل روی فلاپی دیسک موجود در درایو A: اجرا می‌شود، اما در صورتی که فلاپی در درایو موجود نباشد یا به هر دلیل دیگری امکان ایجاد این فایل فراهم نشود برنامه با خطا در زمان اجرا مواجه خواهد شد، اگر فلاپی در درایو موجود نباشد، کادرمحاوره‌ای مانند شکل ۳-۴ نمایش داده می‌شود. در این کادرمحاوره سه دکمه فرمان به صورت فعال دیده می‌شوند.

اگر روی دکمه فرمان Help کلیک کنید، راهنمای ویژوال‌بیسیک شما را در مورد علت خطا راهنمایی خواهد کرد.



شکل ۳-۴

نکته: برای استفاده از راهنمای ویژوال‌بیسیک، باید مجموعه راهنمای MSDN را نصب کرده باشید.

اگر روی دکمه فرمان Debug کلیک کنید، اجرای برنامه به طور موقت متوقف می‌شود و پنجره ویژوال‌بیسیک نمایش داده خواهد شد و دستوری که سبب ایجاد خطا شده است با رنگ زرد مشخص می‌شود. در این حالت برنامه‌نویس می‌تواند بدون قطع روند اجرای برنامه، علت خطا را بررسی کرده و آن را برطرف کند (شکل ۴-۴).

اگر روی دکمه فرمان End کلیک کنید، اجرای برنامه از همان محلی که خطا رخ داده است، متوقف شده و دستورات بعدی اجرا نخواهند شد و پنجره ویژوال‌بیسیک به نمایش درمی‌آید.

شکل ۴-۴

در این‌جا ذکر یک نکته لازم است و آن این‌که کاربرانی که از برنامه استفاده خواهند کرد، هیچ‌گونه آشنایی با زبان برنامه‌نویسی که از آن استفاده کرده‌اند و هم‌چنین نحوه کشف و برطرف کردن خطاهای آن را ندارند، بنابراین وظیفه برنامه‌نویس است که برنامه خود را به‌گونه‌ای تنظیم و طراحی کند که در هنگام برخورد کاربر با خطاهایی که ممکن است، در زمان اجرا رخ دهند، کاربر را به طور صحیح هدایت و راهنمایی کند.

شکل کلی نحوه استفاده از این دستور به این صورت می‌باشد:

On Error GoTo line label

دستوراتی که سبب ایجاد خطا می‌شوند.

line label:

دستورات مدیریت خطا

در این دستور line label می‌تواند یک عبارت یا شماره خط باشد که با کاراکتر کالن (:) همراه است. در صورتی که در زمان اجرای دستوراتی که پس از دستور On Error GoTo قرار می‌گیرند خطایی رخ دهد، اجرای برنامه متوقف نمی‌شود بلکه به دستوراتی که بعد از عبارت یا شماره خط تعیین شده قرار گرفته‌اند، منتقل می‌شود. در این قسمت می‌توان با استفاده از کادرهای پیغام، کاربر را از خطای به وجود آمده باخبر نمود و با توجه به خطای به وجود آمده وی را به شکل مناسب هدایت و راهنمایی کرد یا هر دستوری را که مورد نیاز است در این بخش قرار داد.

اکنون می‌خواهیم این مثال را طوری تنظیم کنیم که بتوان در هنگام اجرای برنامه تصمیم لازم را در صورت نیاز گرفت. برای این کار رویه رویداد قبل را به این شکل تنظیم کنید:

```
Private Sub cmdsave_Click()
   Dim intanswer As Integer
   On Error GoTo ErrorHandler
   intrecno = intrecno + 1
   Open "A:\numbers.dat" For Random Access Write As 1 Len = 4
   Dim sngno As Single
   sngno = Val(txtno.Text)
   Put #1, intrecno, sngno
   Close #1
Exit Sub
ErrorHandler:
    intanswer = MsgBox("ERROR " + Str(Err.Number) + "= " _
          + Err.Description, vbCritical + vbRetryCancel, "DISK ERROR")
   If intanswer = vbCancel Then Resume Next
   If intanswer = vbRetry Then Resume
End Sub
```

در این رویداد پس از تعریف متغیر intanswer از دستور On Error GoTo استفاده شده است. این دستور سبب می‌شود تا در صورتی که خطایی در زمان اجرای دستورات موجود در رویه رخ دهد، اجرای برنامه به دستور بعد از عبارت :Error Handler منتقل شود و برنامه‌نویس می‌تواند با استفاده از عناصری مانند کادرهای پیغام، کاربر را هدایت کند. به عباراتی مانند Error Handler، برچسب خط (line label) می‌گویند. بعد از دستور On Error GoTo دستورات اصلی و در پایان دستورات اصلی رویه، برچسب خط و دستورات

مربوط به کنترل خطای زمان اجرا قرار گرفته‌اند.

حال اگر برنامه را اجرا کرده و روی دکمه فرمان Save، کلیک کنید (بدون آن‌که فلاپی دیسک در درایو موجود باشد) پس از اجرای سه خط اول در رویه رویداد، نوبت به اجرای دستور Open می‌رسد و چون فلاپی در درایو موجود نیست، خطا رخ خواهد داد؛ اما به دلیل استفاده از دستور On Error GoTo اجرای برنامه به اولین دستوری که بعد از برچسب خط، یعنی :Error Handler قرار گرفته است، منتقل می‌شود که در آن از تابع MsgBox جهت راهنمایی کاربر استفاده شده است.

بنابراین کادر پیغامی مشابه شکل ۵-۴ مشاهده می‌کنید. اگر روی دکمه فرمان Cancel کلیک کنید، مقایسه موجود در دستور If بعد از فراخوانی تابع MsgBox نتیجه درست خواهد داشت و در نتیجه دستور Resume Next اجرا می‌شود این دستور سبب می‌شود، اجرای دستوری که سبب ایجاد خطا شده است، متوقف شده و اجرای برنامه به دستور بعد از آن منتقل شود.

شکل ۵-۴

با اجرای دستوراتی که بعد از دستور Open قرار دارند دستور Put اجرا می‌شود اما چون اجرای دستور Open با موفقیت همراه نبوده است این دستور نیز سبب ایجاد خطای دیگری می‌شود و دوباره کادر پیغام دیگری مانند شکل ۶-۴ نمایش داده می‌شود.

اگر در این کادر پیغام روی دکمه فرمان Cancel کلیک کنید دوباره دستور Resume Next اجرا می‌شود و دستور بعدی یعنی دستور Exit Sub اجرا خواهد شد که سبب می‌شود اجرای رویه رویداد خاتمه یابد.

شکل ۶-۴

امـــا اگر در کادر پیغام روی دکمه فرمان Retry کلیک کنید، مقـــایسه موجود در اولین If نتیجـــه نادرست خواهد داشت و دستـــور If بعدی اجرا می‌شود و چون نتیجه مقایسه در این If درست است، دستـــور Resume اجرا می‌شود. این دستور اجرای رویه را به دستـــوری که سبب ایجاد خطا شده است منتقل خواهد کرد و آن را دوباره اجـــرا می‌کند و در صورتی که یک فلاپی دیسک در درایو مربوطه قرار داده شود دستور Open اجرا می‌شود و فایل با موفقیت باز شده و عملیات نوشتن در آن انجام می‌شود، سپس دستور Exit Sub به اجرای رویه خاتمه می‌دهد؛ البته در صورتی که مشکل برطرف نشده باشد مجدداً کادر پیغام، نمایش داده خواهد شد.

در این‌جـا ذکـر ایـن نکته ضـروری اسـت کـه اگـر بـه فـراخوانی تابع MsgBox تـوجه کـنید، می‌بینید کـه دو عبارت Err.Number و Err.Description در فـراخوانی آن اسـتفاده شـده‌اند. در واقع Err یکی از اشـیایی اسـت کـه در ویژوال‌بیسیک بـه منظور مدیریت خطاهـای زمـان اجـرا قـابل اسـتفاده اسـت. این شئ دارای خـواص متعددی اسـت کـه از مهم‌تـرین آن‌ها می‌تـوان بـه خـواص Number و Description اشـاره کـرد. خاصیت Number شـماره خطـایی را کـه رخ داده اسـت، نگهداری می‌کـند و خـاصیت Description می‌تـواند یک توضیح در رابطه بـا خطایی کـه روی داده اسـت در اختیار شـما قـرار دهد.

در صورت نیاز می‌توانید برای مشاهده شماره و توضیح انواع خطاهای زمان اجرای برنامه‌ها به راهنمای MSDN شرکت مایکروسافت مراجعه کنید. در جدول ۱-۴ چند نوع از خطاهای زمان اجرا ارایه شده است.

در پایان به ذکر آخرین نکته در رابطه با دستور On Error می‌پردازیم. از این دستور می‌توان به صورت On Error Resume Next استفاده کرد که در صورت برخورد با یک خطا، اجرای برنامه به دستور بعد از دستوری که سبب بروز خطا شده است، منتقل می‌شود.

هـم‌چنین می‌تـوانید در هـر زمـان کـه بخواهید عملکرد دسـتور On Error GoTo را با دسـتور On Error GoTo ۰ لـغو کنید (البته در صـورت عـدم اسـتفاده از ایـن فـرمـان). در زمـان خروج از رویـه‌ای کـه دسـتور On Error GoTo در آن اسـتفاده شـده اسـت بـه طور خودکـار عملکرد مدیریت خطـا لـغو می‌شـود.

نکته: شئ Err یک متد با نام Clear دارد. اجرای این متد سبب می‌شود کلیه محتویات خاصیت‌های شئ Err پاک شوند. در صورت اجرای دستوراتی از نوع Resume، Exit Sub، Exit Function و On Error این متد را به طور خودکار اجرا می‌کنند.

تمرین: پروژه numbers را به شکلی تنظیم کنید که مدیریت خطای زمان اجرا برای دستورات

رویداد دکمه فرمان sum نیز استفاده شود.

جدول ۱-۴

شماره خطا	پیام خطا	توضیح خطا
۶	Overflow	خطای سرریز شدن حافظه
۷	Out of memory	خطای کمبود حافظه
۱۱	Division by zero	خطای تقسیم بر صفر
۵۳	File not found	خطای عدم وجود فایل
۵۵	File already open	فایل موردنظر برای باز کردن قبلاً باز شده است.
۵۸	File already exists	فایل موردنظر وجود دارد.
۶۱	Disk full	خطای پر شدن دیسک
۷۱	Disk not ready	درایو یا دیسک تعیین شده موجود نیست.
۷۶	Path not found	مسیر تعیین شده موجود نیست.
۵۴	Bad file mode	فایل با حالت (Mode) مناسب برای انجام عملیات باز نشده است.

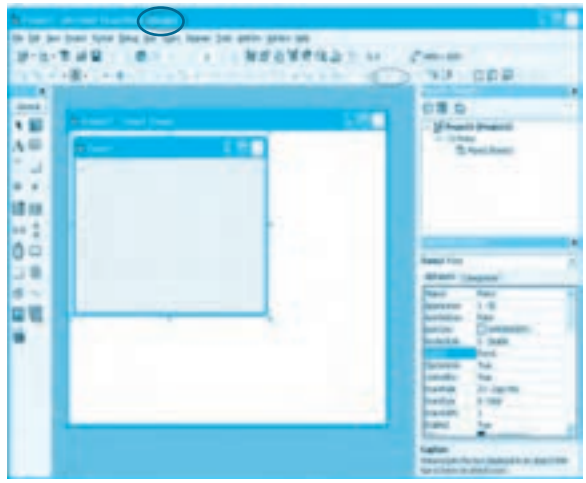
۴-۲ نحوه خطایابی و خطازدایی برنامه‌ها

یکی از توانایی‌های دیگر زبان ویژوال‌بیسیک امکانات فراوان آن در رابطه با اجرای خط به خط برنامه‌ها و پیدا کردن خطاهای منطقی می‌باشد. به‌طورکلی می‌توان از پنجره برنامه ویژوال‌بیسیک در سه حالت (Mode) مختلف استفاده نمود:

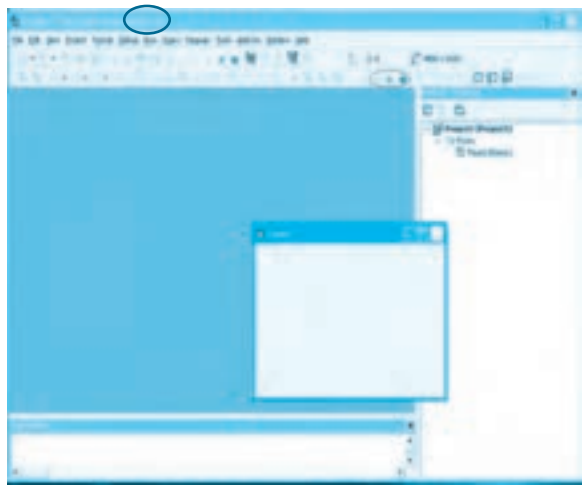
۱- **حالت طراحی:** در این حالت می‌توان پروژه‌ها را طراحی و ایجاد نمود (شکل ۴-۷) و عبارت [design] در نوار عنوان پنجره برنامه ویژوال‌بیسیک نمایش داده می‌شود.

۲- **حالت اجرا:** در این حالت برنامه اجرا شده و رابط گرافیکی برنامه قابل مشاهده و استفاده می‌باشد.

(شکل ۴-۸) و عبارت [run] در نوار عنوان پنجره برنامه ویژوال‌بیسیک نمایش داده می‌شود.



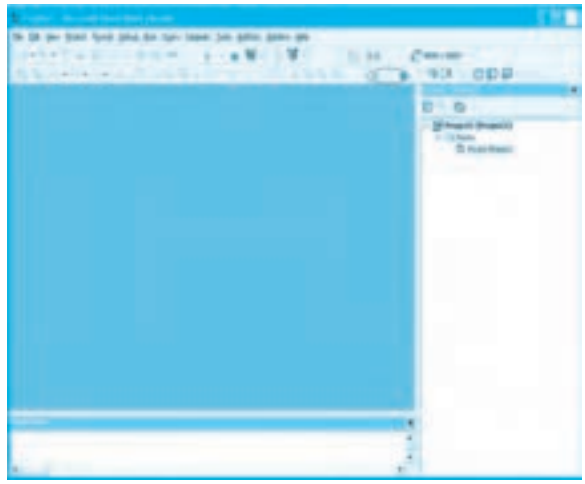
شکل ۷-۴



شکل ۸-۴

۳- حالت توقف: در این حالت برنامه به طور موقت، متوقف می‌شود و برنامه‌نویس می‌تواند برنامه را خط به خط اجرا، بررسی و خطاهای موجود را شناسایی و برطرف کند (شکل ۹-۴).

به علاوه عبارت [break] در نوار عنوان پنجره برنامه ویژوال‌بیسیک نمایش داده می‌شود.



شکل ۹-۴

۴-۲-۱ حالت توقف (Break Mode)

Break Mode حالتی است که اجرای برنامه را به طور موقت متوقف می‌کند و شما علاوه بر پنجره برنامه، به پنجره ویژوال‌بیسیک و محیط طراحی برنامه به طور همزمان دسترسی خواهید داشت؛ در این حالت می‌توانید عملیات زیر را انجام دهید:

۱- تغییر و ویرایش دستورات

۲- مشاهده رویه‌ای که در حال اجرا است.

۳- مشاهده مقادیر متغیرها و عبارات

۴- تغییر مقدار متغیرها

۵- مشاهده یا کنترل دستوراتی که بعد از توقف برنامه اجرا می‌شوند.

۶- اجرای دستورات به صورت جداگانه در پنجره اجرای فوری دستورات (Immediate Window)

۷- مشاهده حالت و شکل ظاهری رابط گرافیکی نرم‌افزار

برای متوقف کردن اجرای یک برنامه و ایجاد حالت توقف، می‌توانید یکی از این روش‌ها را استفاده کنید:

الف- ایجاد یک نقطه توقف (Break Point) در برنامه

ب- استفاده از کلید ترکیبی Ctrl+ Break در هنگام اجرای برنامه

ج- استفاده از دستور Stop در بخش کد برنامه

د- توقف برنامه در هنگامی‌که خطای زمان اجرا رخ می‌دهد.

هـ- استفاده از گزینه‌های Break When Value Is True و Break When Value Changes در کادرمحاوره Add Watch

و- استفاده از گزینه Run To Cursor (یا کلید ترکیبی Ctrl+ F۸) از منوی Debug

ز- انتخاب گزینه Break از منوی Run در نوار منوی پنجره ویژوال‌بیسیک

ح- کلیک روی دکمه ⏸ در نوار ابزار Debug پنجره ویژوال‌بیسیک

در صورتی که بخواهید از حالت توقف (Break Mode) خارج شوید و به اجرای برنامه ادامه دهید می‌توانید یکی از روش‌های زیر را استفاده کنید:

الف- روی منوی Run در پنجره ویژوال‌بیسیک کلیک کنید و سپس گزینه Continue را برگزینید (یا کلید F۵ را بفشارید).

ب- روی دکمه ▶ در بخش RunIcons در نوار ابزار پنجره ویژوال‌بیسیک کلیک کنید.

ج- روی منوی Run در پنجره ویژوال‌بیسیک کلیک کنید، سپس روی گزینه Restart کلیک کنید (یا کلید ترکیبی Shift+F۵ را فشار دهید). این گزینه سبب اجرای برنامه از ابتدا می‌شود.

به عنوان مثال پروژه numbers را باز کنید و آن را اجرا نمایید، سپس روی دکمه ⏸ در نوار ابزار استاندارد در پنجره ویژوال‌بیسیک کلیک کنید (یا یکی از روش‌های گفته شده را استفاده کنید). پس از این عمل برنامه به طور موقت متوقف می‌شود و پنجره ویژوال‌بیسیک در حالت توقف قرار می‌گیرد. برای خروج از این حالت می‌توانید روی دکمه ▶ در نوار ابزار استاندارد در پنجره ویژوال‌بیسیک کلیک کنید و دوباره به حالت اجرا بازگردید.

۲-۲-۴ پنجره اجرای فوری دستورات (Immediate Window)

این پنجره در حالت توقف برنامه به طور خودکار باز می‌شود و در زمان باز شدن هیچ گونه محتویاتی ندارد (شکل ۱۰-۴). شما می‌توانید دستورات خود را در این پنجره تایپ کنید و با فشردن کلید Enter نتیجه اجرای آن‌ها را مشاهده کنید یا دستورات خود را از داخل این پنجره کپی کرده و در پنجره کدنویسی قرار دهید یا بالعکس. به علاوه شما می‌توانید نتیجه محاسبات خود را به وسیله دستور Print و پس از اجرای آن در زیر دستور خود مشاهده کنید. البته باید توجه داشت که از پنجره اجرای فوری دستورات می‌توان در زمان طراحی پروژه نیز استفاده کرد.

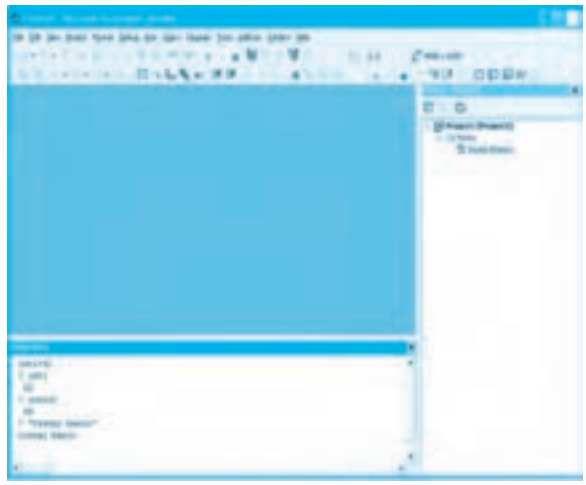
برای این کار می‌توانید در نوار منوی پنجره ویژوال‌بیسیک روی منوی View، سپس روی گزینه Immediate Window کلیک کنید یا در نوار ابزار Debug روی دکمه  کلیک کنید.

برنامه ویژوال‌بیسیک را اجرا کنید و یک پروژه از نوع Standard EXE ایجاد کنید، سپس این عملیات را به ترتیب انجام دهید:

۱- پروژه را اجرا کنید و در نوار ابزار پنجره ویژوال‌بیسیک روی دکمه ⏸ کلیک کنید یا کلید ترکیبی

Ctrl+Break را فشار دهید.

۲- در این مرحله اجرای پروژه متوقف شده و پنجره اجرای فوری دستورات در قسمت پایین پنجره ویژوال‌بیسیک نمایش داده می‌شود (شکل ۱۰-۴).



شکل ۱۰-۴

۳- در پنجره اجرای فوری دستورات کلیک کرده و چند دستور را مانند شکل ۱۱-۴ تایپ کنید. پس از تایپ هر یک از دستورات، کلید Enter را فشار دهید تا نتیجه اجرای آن را در زیر همان دستور مشاهده کنید.

۴- در پایان در نوار ابزار استاندارد پنجره ویژوال‌بیسیک روی دکمه ■ کلیک کنید تا اجرای پروژه خاتمه یابد.

۵- بدون ذخیره‌سازی پروژه از برنامه ویژوال‌بیسیک خارج شوید.

نکته: در پنجره اجرای فوری دستورات می‌توان محتویات متغیرهای برنامه، خاصیت‌های کنترل‌ها و فرم را مشاهده کرد.

۴-۲-۳ Debug Object

Debug Object یکی از اشیای موجود در ویژوال‌بیسیک است که برای متوقف کردن اجرای برنامه در شرایط خاص، ارسال پیام‌ها یا نمایش مقادیر موردنظر در پنجره اجرای فوری دستورات مورد استفاده قرار می‌گیرد.

این شئ دارای دو متد Assert و Print است، متد Assert می‌تواند با بررسی یک مقدار منطقی و مشاهده

مقدار False، اجرای برنامه را موقتاً متوقف سازد، شکل کلی نحوه استفاده از این متد به این صورت است:

Debug.Assert booleanexpression

booleanexpression یک عبارت مقایسه‌ای است که نتیجه آن True یا False است. در صورتی‌که مقدار این عبارت برابر با False باشد، اجرای برنامه متوقف شده و خطی که این متد در آن قرار گرفته است به رنگ زرد نمایش داده می‌شود، اما اگر مقدار این عبارت True باشد اجرای برنامه متوقف نمی‌شود. متد Print می‌تواند عبارت یا مقدار موردنظر شما را در پنجره اجرای فوری دستورات نمایش دهد، شکل کلی متد Print نیز به این صورت است:

Debug.print expression

Expression عبارتی است که در پنجره اجرای فوری دستورات نمایش داده می‌شود.

مثال ۱: می‌خواهیم پروژه numbers را به شکلی تنظیم کنیم که در زمان ورود یک عدد منفی، برنامه به‌طور موقت متوقف شده و پیام مناسبی در پنجره اجرای فوری دستورات نمایش داده شود؛ برای این‌کار این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه numbers را باز کنید.

۲- رویداد دکمه فرمان Save را به این صورت تنظیم کنید:

```
Private Sub cmdsave_Click()
   Dim intanswer As Integer
   On Error GoTo ErrorHandler
   intrecno = intrecno + 1
   Open "C:\numbers.dat" For Random Access Write As 1 Len = 4
   Dim sngno As Single
   sngno = Val(txtno.Text)
   Debug.Assert sngno >= 0
   Debug.Print "your number < 0"
   Put #1, intrecno, sngno
   Close #1
   Exit Sub
ErrorHandler:
    intanswer = MsgBox("ERROR " + Str(Err.Number) + "= " _
          + Err.Description, vbCritical + vbRetryCancel, "DISK ERROR")
```

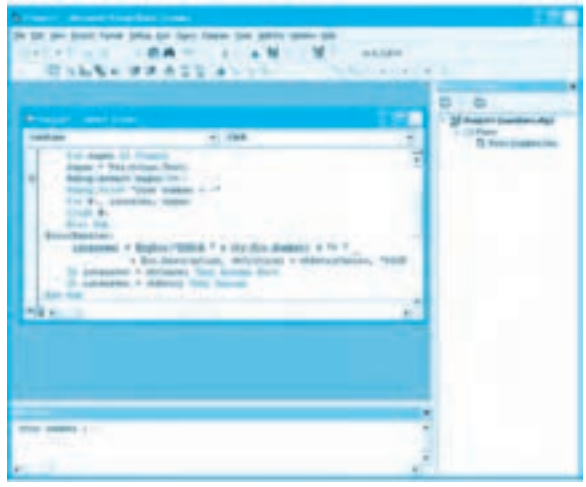
```
    If intanswer = vbCancel Then Resume Next
    If intanswer = vbRetry Then Resume
End Sub
```

در این رویداد از متدهای Debug Object قبل از دستور Put استفاده شده است. متد Debug Object.Assert عدد دریافت شده را بررسی می‌کند اگر عدد منفی باشد نتیجه بررسی شرط sngno > = 0 نادرست (False) می‌شود در نتیجه اجرای برنامه متوقف می‌شود و متد Debug Object Print پیام تعیین شده را نمایش می‌دهد. اما اگر عدد صفر یا مثبت باشد نتیجه بررسی شرط sngno > = 0 درست (True) می‌شود؛ بنابراین اجرای برنامه ادامه یافته و پیامی هم نمایش داده نخواهد شد.

۳- برای آن‌که در هنگام اجرای دستور Open خطای زمان اجرا به‌وجود نیاید در رویداد دکمه‌های فرمان Save و Sum نام درایو را در دستور Open روی یک درایو مناسب مانند C: تنظیم کنید.

۴- تغییرات را ذخیره و پروژه را اجرا کرده و یک عدد مثبت تایپ کنید و روی دکمه فرمان Save کلیک کنید. عدد وارد شده ذخیره می‌شود.

۵- اکنون یک عدد منفی وارد کرده و روی دکمه فرمان Save کلیک کنید؛ در این حالت برنامه متوقف شده و خطی که متد Assert در آن قرار دارد با رنگ زرد نمایش داده می‌شود به‌علاوه پیام your number < 0 نیز در پنجره اجرای فوری دستورات نمایش داده می‌شود(شکل ۱۱-۴).



شکل ۱۱-۴

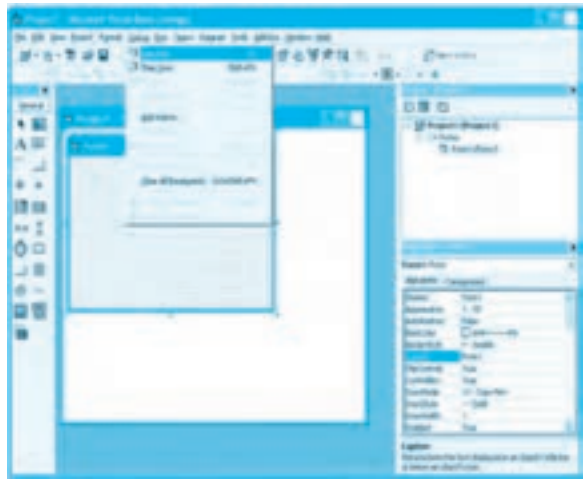
۶- در این مرحله می‌توانید اجرای برنامه را با کلیک روی دکمه ▶ یا فشردن کلید F5 ادامه داده یا با کلیک روی دکمه ■ خاتمه دهید.

۷- به اجرای پروژه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه numbers را به شکلی تنظیم کنید تا هنگام خواندن اعداد از فایل و محاسبه مجموع آن‌ها اگر عدد منفی از فایل خوانده شود برنامه با استفاده از شئ Debug به‌طور خودکار متوقف شده و پیام مناسبی نمایش داده شود.

Debug Menu ۴-۲-۴

ابزار دیگری که در ویژوال‌بیسیک جهت خطایابی و آزمایش درستی برنامه‌ها مورد استفاده قرار می‌گیرد در منوی Debug در پنجره ویژوال‌بیسیک قرار دارد. گزینه‌های متعددی در این منو وجود دارند که به منظور انجام عملیات خطایابی و خطازدایی برنامه بسیار مفید هستند (شکل ۱۲-۴). در این‌جا به توضیح عملکرد هریک از این گزینه‌ها می‌پردازیم.



شکل ۱۲-۴

گزینه Step Into

Step Into اولین گزینه در منوی Debug است که برای اجرای خط به خط یک پروژه مناسب است (شکل ۱۲-۴). با انتخاب این گزینه در هر مرتبه یک خط از خطوط پروژه اجرا می‌شود و خط بعدی با رنگ زرد (که در سمت چپ آن نیز یک فلش با رنگ زرد وجود دارد) نشان داده خواهد شد.

این گزینه در زمانی‌که پنجره ویژوال‌بیسیک در حالت طراحی یا توقف قرار دارد، قابل استفاده و در حالت اجرای برنامه غیرفعال است.

نکته: فشردن کلید **F8** معادل گزینه **Step Into** است.

نکته: در صورت فعال بودن نوار ابزار Debug در پنجره ویژوال‌بیسیک می‌توانید از دکمه در این نوار ابزار استفاده کنید.

به منظور فعال کردن نوار ابزار Debug در مکان خالی از نوار منو یا سایر نوارهای ابزار موجود کلیک راست کنید و از منویی که ظاهر می‌شود، گزینه Debug را انتخاب کنید.

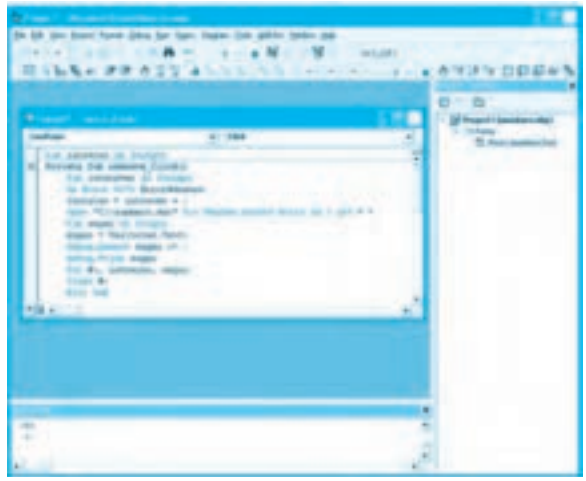
مثال ۲: می‌خواهیم پروژه numbers را با استفاده از گزینه Step Into خط به خط اجرا کرده و نحوه اجرای دستورات را بررسی و مشاهده کنیم؛ برای این کار این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه numbers را باز کنید.

۲- در پنجره ویژوال‌بیسیک ابتدا روی منوی Debug و سپس روی گزینه Step Into کلیک کنید (یا کلید F8 را فشار دهید).

۳- برنامه اجرا شده و فرم آن نمایش داده می‌شود؛ یک عدد در کادر متن وارد کنید و روی دکمه فرمان Save کلیک کنید. در این لحظه اجرای برنامه به‌طور موقت قطع می‌شود و اولین خط از رویداد Click دکمه فرمان Save با رنگ زرد نمایش داده می‌شود (شکل ۱۳-۴).

۴- چند بار دیگر کلید F8 را بفشارید تا در هر بار یک خط از رویداد اجرا شود به این صورت می‌توانید نحوه اجرا و عملکرد هر خط از دستورات را به‌طور دقیق بررسی کنید و از صحت عملکرد آن‌ها اطمینان حاصل کنید یا مشکلات احتمالی را پیدا نموده و برطرف کنید.



شکل ۱۳-۴

۵- در نوار استاندارد پنجره ویژوال‌بیسیک روی دکمه کلیک کنید تا اجرای برنامه ادامه یابد.

۶- اجرای برنامه را متوقف کنید و به پنجره ویژوال‌بیسیک بازگردید.

گزینه Step Over

این گزینه پس از گزینه Step Into قرار گرفته است.

عملکرد این گزینه مانند گزینه Step Into است با این تفاوت که از این گزینه زمانی استفاده می‌شود که یک رویه فراخوانی می‌شود و نیازی به اجرای خط به خط دستورات رویه فراخوانی شده نیست.

این گزینه تمام رویه را به‌طور یکجا اجرا کرده و اجرای برنامه را در خط بعد از فراخوانی متوقف می‌کند، پس می‌توانید به‌وسیله این گزینه یا گزینه Step Into یا سایر روش‌ها به اجرای برنامه ادامه دهید.

نکته: کلید ترکیبی Shift+F8 معادل گزینه Step Over است.

نکته: در صورت فعال بودن نوار ابزار Debug در پنجره ویژوال‌بیسیک می‌توانید از دکمه در این نوار ابزار نیز استفاده کنید.

مثال ۳: می‌خواهیم پروژه numbers را با استفاده از گزینه Step Into اجرا کرده و نحوه اجرای دستورات را بررسی کنیم.

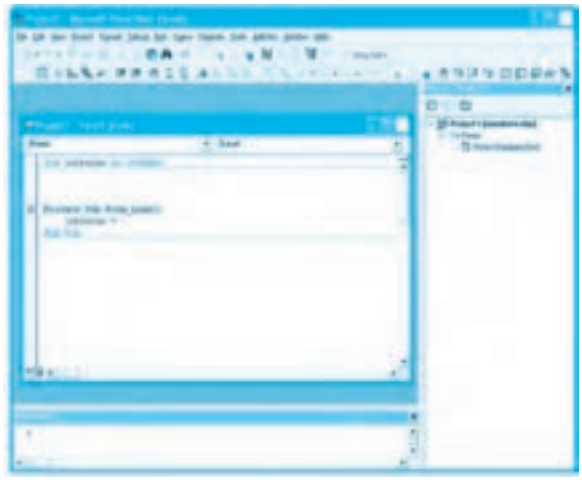
برای این کار این عملیات را به ترتیب انجام دهید:

۱- پروژه numbers را باز کنید و رویداد Load فرم را به این صورت تنظیم کنید:

```
Private Sub Form_Load()
   intrecno = 0
End Sub
```

در این رویداد متغیر intrecno مقدار اولیه صفر قرار داده می‌شود.

۲- تغییرات را ذخیره کرده و برنامه را با استفاده از گزینه Step into اجرا کنید. در این مرحله برنامه در اولین خط از رویداد Load فرم متوقف می‌شود (شکل ۱۴-۴).



شکل ۱۴-۴

۳- دو بار دیگر کلید F8 را فشار دهید، اکنون فرم برنامه مشاهده می‌شود. یک عدد در کادر متن تایپ کرده و روی دکمه فرمان OK کلیک کنید.

این‌بار اجرای برنامه در اولین خط از رویداد کلیک دکمه فرمان Save متوقف می‌شود. چند بار دیگر با استفاده از کلید F8 دستوراتی از این رویداد را اجرا کنید و نتیجه را بررسی نمایید، سپس اجرای برنامه را خاتمه دهید.

۴- اکنون برنامه را با استفاده از گزینه Step Over اجرا کنید. در این حالت برنامه دوباره در اولین خط از رویداد Load فرم متوقف می‌شود. دوبار دیگر روی دکمه  در نوار ابزار Debug کلیک کنید تا فرم برنامه در زمان اجرا نمایش داده شود.

۵- یک عدد در کادر متن تایپ کنید و روی دکمه فرمان Save کلیک کنید. تمام دستورات رویداد Click دکمه فرمان Save بدون آن‌که وارد رویداد شوید در یک مرحله اجرا می‌شود.

۶- اجرای برنامه را متوقف کرده و به پنجره ویژوال‌بیسیک بازگردید.

گزینه Step Out

با انتخـــاب این گزینه از منـــوی Debug تمام دستورات باقیمانده از رویه در حال اجرا، به‌طور همزمان و یکجا اجرا شده و اجرای برنامه به محل فراخوانی رویه منتقل می‌شود. استفاده از این گزینه در مواقعی مناسب است که به وسیله گزینه Step Into چند دستور را بررسی کرده و نیازی به اجرای دستورات باقیمانده نداشته باشید. بنابراین این کار سرعت عملیات خطایابی و برطرف کردن اشکالات را زیاد می‌کند.

نکته: کلید ترکیبی **Ctrl+Shift+F8** معادل گزینه **Step Out** است.

نکته: در صورت فعال بودن نوار ابزار **Debug** در پنجره ویژوال‌بیسیک می‌توانید از دکمه  در این نوار ابزار استفاده کنید.

مثال ۴: می‌خواهیم پروژه numbers را با استفاده از گزینه‌های Step Into و Step Out اجرا کنیم و نحوه ایجاد دستورات را بررسی نماییم. برای این کار این عملیات را به ترتیب انجام دهید:

۱- پروژه numbers را باز کنید و آن را با استفاده از گزینه Step Into اجرا کنید.

۲- برنامه در اولین خط از رویداد Load فرم متوقف می‌شود.

۳- دو بار دیگر کلید F8 را فشار دهید و در فرم برنامه یک عدد در کادر متن تایپ کنید و روی دکمه فرمان Save کلیک کنید.

۴- پس از توقف دوباره برنامه در رویداد Click دکمه فرمان Save، دوباره با استفاده از گزینه Step Into چند دستور از مجموعه دستورات رویداد را اجرا کنید.

۵- در این مرحله روی دکمه  در نوار ابزار Debug کلیک کنید، اکنون تمام دستورات باقیمانده در رویداد در یک مرحله اجرا می‌شوند و بدون هیچ توقفی رویداد پایان می‌یابد.

۶- به اجرای برنامه پایان داده و به پنجره ویژوال‌بیسیک بازگردید.

گزینه Run To Cursor

یکی دیگر از گزینه‌های موجود در منوی Debug گزینه Run To Cursor است.

این گزینه در مواقعی استفاده می‌شود که لازم است دستورات تا محل معینی اجرا شوند. برای اجرای برنامه تا دستور موردنظر، در پنجره کدنویسی، مکان‌نما را در خطی که دستور موردنظر در آن قرار گرفته است، قرار دهید و سپس گزینه Run To Cursor را از منوی Debug انتخاب کنید. زمانی‌که اجرای برنامه به دستوری که مکان‌نما در آن قرار گرفته است، برسد، دستور مربوط با رنگ زرد و یک فلش زرد رنگ در سمت چپ آن نمایش داده می‌شود. می‌توانید از این گزینه هم در شروع اجرای برنامه و هم در زمان رسیدن برنامه به یک نقطه توقف استفاده کنید.

نکته: کلید ترکیبی **Ctrl+F8** معادل گزینه **Run To Cursor** است.

مثال ۵: می‌خواهیم پروژه numbers را با استفاده از گزینه Run To Cursor به شکلی اجرا کنیم که با رسیدن به دستور Open در رویداد Click دکمه فرمان Save برنامه متوقف شود. برای انجام این مثال این

عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه numbers را باز کنید.

۲- به رویداد Click دکمه فرمان Save بروید و مکان‌نما را در خطی که دستور Open وجود دارد قرار دهید.

۳- در نوار منوی پنجره ویژوال‌بیسیک روی منوی Debug و سپس گزینه Run To Cursor کلیک کنید. در این مرحله فرم برنامه در زمان اجرا نمایش داده می‌شود، در کادر متن یک عدد تایپ کرده و روی دکمه فرمان Save کلیک کنید.

۴- اکنون اجرای برنامه ادامه می‌یابد و با رسیدن به دستور Open متوقف می‌شود.

۵- مکان‌نما را در خطی که دستور Put قرار دارد ببرید و دوباره برنامه را با گزینه Run To Cursor اجرا کنید. این‌بار دستورات برنامه تا رسیدن به دستور Put اجرا می‌شوند و سپس برنامه در این محل متوقف می‌گردد.

به وسیله کلید F5 اجرای رویداد را ادامه داده و سپس با کلیک روی دکمه ■ در نوار ابزار استاندارد یا نوار ابزار Debug به اجرای پروژه پایان دهید.

گزینه ... Add Watch

گاهی اوقات پیش می‌آید که با برطرف کردن خطاهای نوشتاری و منطقی مشکلات برطرف نمی‌شوند و خروجی موردنظر به‌دست نمی‌آید. در این حالت می‌توان بررسی مقادیر متغیرها، عبارات، ثابت‌ها و خواص کنترل‌ها و اشیا، نحوه اجرای برنامه و نتیجه عملیات محاسباتی و پردازش‌های گوناگون را مشاهده کنید. گزینه ... Add Watch در منوی Debug این امکان را به آسانی فراهم می‌آورد.

پس از انتخاب این گزینه کادرمحاوره Add Watch مطابق شکل ۱۵-۴ نمایش داده می‌شود. در این کادرمحاوره اجزای مختلفی وجود دارند که به توضیح هریک از آن‌ها می‌پردازیم.

شکل ۱۵-۴

در این کادرمحاوره و در کادر متن Expression می‌توانید نام متغیر و خاصیت یا ثابتی که می‌خواهید مقدار آن را در طی اجرای برنامه مشاهده کنید، بنویسید. در بخش Context دو کادر لیست قرار دارند که می‌توانید از کادرهای لیست ترکیبی بازشو Procedure و Module به ترتیب نام رویه و ماژولی را که عبارت موردنظر در آن قرار گرفته است، انتخاب کنید.

البته می‌توانید در این لیست‌ها به ترتیب مقادیر All Procedures یا All Modules را انتخاب کنید تا تمام رویه‌ها یا ماژول‌های موردنظر قرار بگیرند. در زیر این لیست‌ها نیز نام پروژه نمایش داده می‌شود.

در بخش Watch Type نیز سه دکمه انتخاب قرار گرفته است. اگر دکمه انتخاب Watch Expression را برگزینید در صورت ایجاد یک نقطه توقف در برنامه (Break Point)، مقدار عبارتی را که در کادر متن Expression نوشته شده است، در پنجره Watches برنامه ویژوال‌بیسیک مشاهده خواهید کرد و هر زمان مقدار عبارت مربوطه تغییر کند این تغییرات را بلافاصله در پنجره Watches برنامه ویژوال‌بیسیک مشاهده خواهید کرد. در صورت انتخاب دکمه انتخاب Break When Value Is True، اگر مقدار عبارتی که در کادر متن Expression نوشته شده است درست (True) یا هر مقداری بجز صفر باشد، یک نقطه توقف در دستور بعدی ایجاد شده و بلافاصله پنجره ویژوال‌بیسیک به همراه پنجره Watches نمایش داده می‌شود.

توجه داشته باشید که این گزینه برای عبارات و متغیرهای رشته‌ای مناسب نیست. اگر دکمه انتخاب Break When Value Changes را برگزینید در صورتی‌که مقدار عبارتی که در کادر متن Expression نوشته شده است در طول اجرای برنامه تغییر کند، برنامه در دستور بعدی متوقف می‌شود و بلافاصله پنجره ویژوال‌بیسیک به همراه پنجره Watches نمایش داده می‌شود.

در پایان پس از تعیین تنظیمات موردنظر، روی دکمه فرمان OK در کادرمحاوره Add Watch کلیک کنید.

نکته: برای اضافه کردن یک متغیر، خاصیت یا عبارت جدید در پنجره Watches می‌توانید در پنجره Watches کلیک راست کنید و در منویی که نمایش داده می‌شود روی گزینه Add Watch ... کلیک کنید.

نکته: برای دسترسی به کادرمحاوره Add Watch در زمان طراحی یا هنگام توقف برنامه در زمان اجرا می‌توانید در نوار ابزار Debug روی دکمه کلیک کنید یا در پنجره ویژوال‌بیسیک روی منوی View و سپس گزینه Watch Window کلیک کنید.

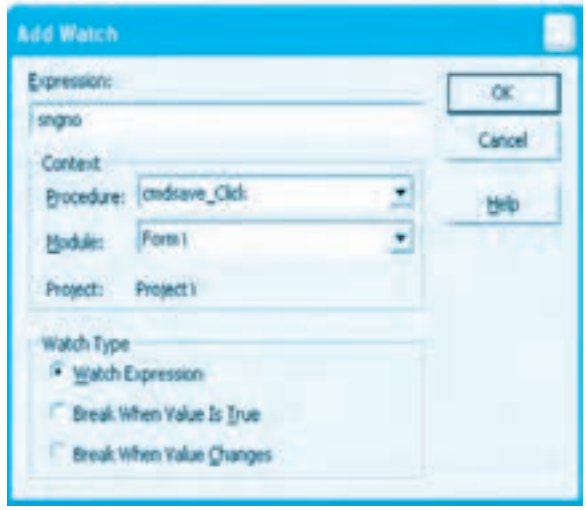
نکته: برای حـــذف یک متغیر، خـاصیت یا عبـارت از پنجره Watches، در آن پنجـره روی عبـارت موردنظر کلیک راست کرده و سپس در منویی که نمایـش داده می‌شـود روی گزینه Delete Watch کلیک کنید یا پس از انتخاب عبـارت مـوردنظر در پنجره Watches، کلیـد Delete را در صفحه کلید فشار دهید.

مثال ۶: می‌خواهیم پروژه numbers را با استفاده از ابزار موجود در ویژوال‌بیسیک خط به خط اجرا کرده و به‌طور همزمان محتویات متغیرها و خاصیت‌های اشیا را مشاهده و بررسی کنیم. برای این کار این عملیات را به ترتیب انجام دهید:

۱- پروژه numbers را باز کنید.

۲- با استفاده از گزینه Step Into برنامه را اجرا کنید تا وارد رویداد Click دکمه فرمان Save شوید.

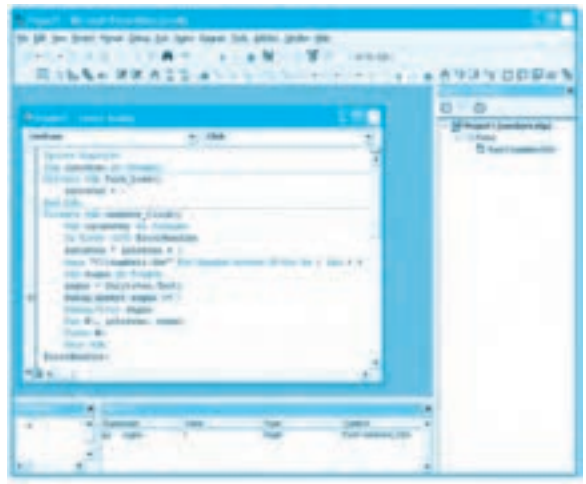
۳- در نوار منوی پنجره ویژوال‌بیسیک روی منوی Debug و سپس گزینه ... Add Watch کلیک کنید تا کادرمحاوره Add Watch نمایش داده شود. در این کادرمحاوره و در کادر متن Expression نام متغیر sngno را تایپ کرده و از کادر لیست ترکیبی بازشو Procedure، رویه cmdsave_Click را انتخاب کنید (شکل ۱۶-۴)، سپس روی دکمه فرمان OK کلیک نمایید.



شکل ۱۶-۴

۴- در این مرحله پنجره Watch در پایین پنجره ویژوال‌بیسیک و در کنار پنجره اجرای فوری دستورات نمایش داده می‌شود (شکل ۱۷-۴) و داخل آن نام متغیر sngno قابل مشاهده می‌باشد. در این پنجره علاوه بر ستون

Expression که نام متغیر را نشان می‌دهد ستون Value، مقدار ذخیره، ستون Type نوع داده و ستون Context نام رویه و ماژولی را که متغیر موردنظر در آن قرار دارد، مشخص می‌کنند.



شکل ۱۷-۴

نکته: در صورتی‌که پنجره Watch در هنگام اجرای برنامه نمایش داده نشود می‌توانید با استفاده از گزینه Watch Window در منوی View پنجره ویژوال‌بیسیک آن را نمایش دهید.

۵- مکان‌نما را در خطی که دستور sngno = Val (txtno.Text) در آن قرار دارد ببرید و با استفاده از گزینه Run To Cursor برنامه را تا این دستور اجرا کنید و مقدار متغیر sngno را در پنجره Watches بررسی کنید. سپس کلید F8 را فشار دهید.

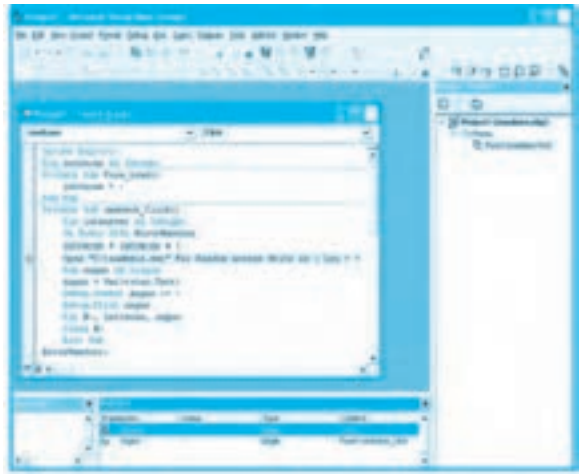
۶- اکنون دوباره در پنجره Watch مقدار متغیر sngno را بررسی کنید. مقدار این متغیر از صفر به مقداری که در کادر متن تایپ نموده‌اید تغییر کرده است. بنابراین با این روش می‌توان مقدار متغیرها و خاصیت‌های اشیا را در هنگام اجرای خط به خط برنامه بررسی نمود و محلی را که خطا در آن به‌وجود می‌آید، شناسایی و برطرف کرد.

۷- به اجرای پروژه خاتمه دهید و به کادرمحاوره Add Watch بروید، سپس در کادر متن Expression نام متغیر intrecno را تایپ کنید و در کادر لیست ترکیبی بازشوی Procedure گزینه (All Procedures) را انتخاب کنید و در بخش Watch Type روی دکمه انتخاب Break When Value Changes کلیک کنید و بعد روی دکمه فرمان OK کلیک کنید و بعد روی دکمه فرمان کلیک نمایید (شکل ۱۸-۴).

شکل ۱۸-۴

در این مرحله متغیر intrecno به پنجره Watches اضافه می‌شود.

۸- پروژه را اجرا کنید و یک عدد در کادر متن فرم برنامه تایپ کرده سپس روی دکمه Save کلیک کنید. در این مرحله اجرای برنامه با رسیدن به دستور Open در رویداد cmdsave_Click به‌طور موقت متوقف می‌شود (شکل ۱۷-۴). چون دستور intrecno=intrecno+1 که قبل از دستور Open قرار دارد مقدار این متغیر را تغییر می‌دهد، بنابراین با توجه به تنظیماتی که کادرمحاوره Add Watch برای متغیر intrecno در نظر



شکل ۱۹-۴

گرفته است اجرای برنامه در دستور بعد یعنی دستور Open متوقف می‌شود و سطری که در پنجره Watch دارای متغیر intrecno می‌باشد با یک نوار به رنگ آبی نمایش داده می‌شود. (شکل ۱۹-۴).

۹- اجرای پروژه را خاتمه دهید و به کادرمحاوره Add Watch بروید، سپس در کادر متن Expression عبارت intrecno > 2 را تایپ کنید و در کادر لیست ترکیبی بازشوی Procedure گزینه (All Procedures) را انتخاب کنید و در بخش Watch Type روی دکمه انتخاب Break When Value Is True کلیک کنید و بعد روی دکمه فرمان OK کلیک نمایید (شکل ۲۰-۴).

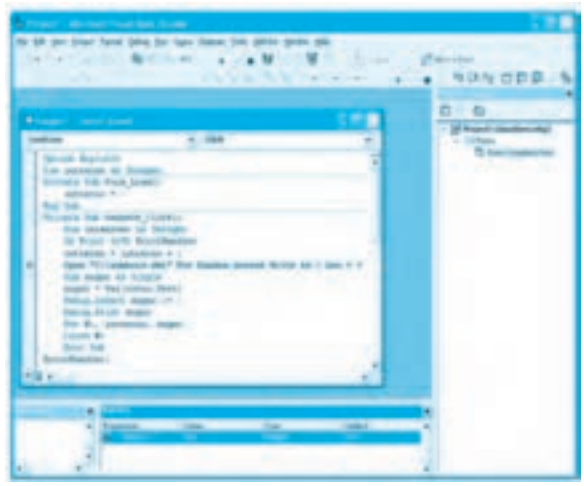
شکل ۲۰-۴

در این مرحله عبارت intrecno > 2 به پنجره Watches اضافه می‌شود.

۱۰- در پنجره ویژوال‌بیسیک به پنجره Watches بروید و روی نام متغیر intrecno کلیک راست کنید و در منوی بازشویی که نمایش داده می‌شود روی گزینه Delete Watch کلیک کنید تا متغیر intrenco از پنجره Watch حذف شود.

۱۱- عبارت sngno را نیز از پنجره Watches حذف کنید.

۱۲- پروژه را با استفاده از کلید F5 اجرا کنید و اعداد خود را وارد کرده و روی دکمه Save کلیک کنید. وقتی سومین عدد را تایپ می‌کنید و روی دکمه Save کلیک می‌نمایید اجرای برنامه در رویداد cmdsave_Click و دستور Open متوقف می‌شود. چون در این لحظه مقدار متغیر intrecno، برابر با ۳ می‌باشد، حاصل عبارت intrecno > 2 برابر با True می‌شود و با توجه به تنظیمات انجام شده در کادرمحاوره Add Watch، برنامه باید در این مرحله متوقف گردد. در این حالت عبارت intrecno > 2 نیز در پنجره Watches با رنگ آبی نمایش داده می‌شود (شکل ۲۱-۴). مقدار این عبارت را در پنجره Watches بررسی کنید.



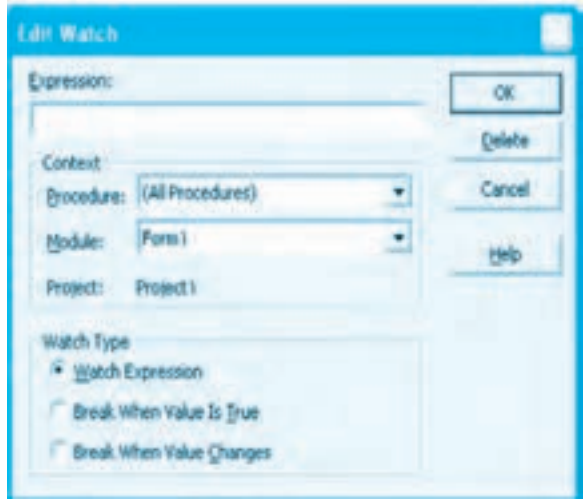
شکل ۴-۲۱

۱۳- اجرای پروژه را پایان دهید و به پنجره ویژوال‌بیسیک بازگردید.

گزینه ... Edit Watch

درصورتی‌که بخواهید عبارات موجود در پنجره Watches و تنظیمات مربوط به آن را ویرایش یا حذف کنید یا تنظیمات مربوط به آن را تغییر دهید، گزینه ... Edit Watch را از منوی Debug در پنجره ویژوال‌بیسیک انتخاب کنید.

در صورت انتخاب این گزینه کادرمحاوره Edit Watch به منظور ویرایش عبارت موردنظر و تغییر تنظیمات مربوط به آن، نمایش داده می‌شود (شکل ۴-۲۲).



شکل ۴-۲۲

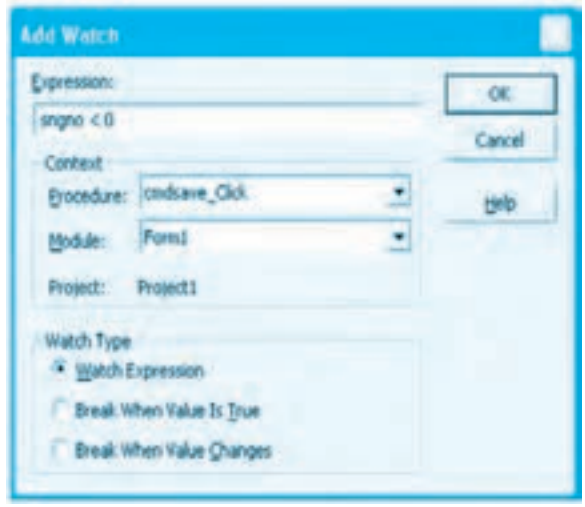
این کادرمحاوره شبیه به کادرمحاوره Add Watch است، تنها تفاوتی که این کادرمحاوره با کادرمحاوره Add Watch دارد، دکمه فرمان Delete است که به‌وسیله آن می‌توان عبارت موردنظر را از پنجره Watches حذف کرد.

نکته: کلید ترکیبی Ctrl+W معادل گزینه ... Edit Watch در منوی Debug است.

نکته: برای دسترسی به کادرمحاوره Edit Watch می‌توانید در پنجره Watches روی عبارت موردنظر کلیک راست کنید و از منویی که ظاهر می‌شود گزینه ... Edit Watch را انتخاب کنید.

مثال ۷: پروژه numbers را باز کنید و این عملیات را به ترتیب انجام دهید:

۱- به کادرمحاوره Add Watch بروید و آن را مطابق شکل ۲۳-۴ تنظیم کنید.



شکل ۲۳-۴

۲- پروژه را اجرا کرده و چند عدد مثبت در فایل ذخیره کنید، سپس یک عدد منفی وارد نمایید؛ برنامه در این لحظه متوقف می‌شود. به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید.

۳- در پنجره ویژوال‌بیسیک روی منوی Debug و سپس گزینه ... Edit Watch کلیک کنید.

۴- در کادرمحاوره Edit Watch به کادر متن Expression بروید و عبارت sngno < 0 را به‌صورت sngno <= 0 تنظیم کنید و سپس روی دکمه فرمان OK کلیک کنید.

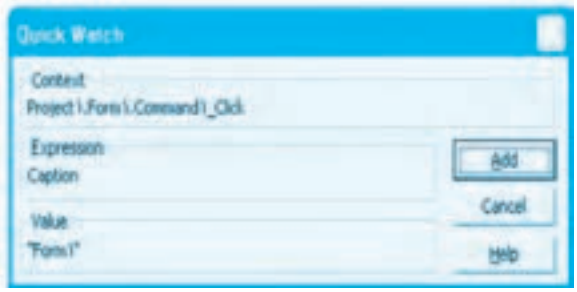
۵- پروژه را اجرا کرده و نحوه اجرا و عکس‌العمل آن را برای عدد مثبت، منفی و صفر بررسی کنید.

۶- اجرای پروژه را خاتمه دهید و عبارت‌های موجود در پنجره Watches را با استفاده از دکمه فرمان Delete در کادرمحاوره Edit Watch حذف کنید.

گزینه ... Quick Watch

در صورتی‌که بخواهید مقدار یک متغیر یا خاصیت را بدون استفاده از پنجره Watches مشاهده کنید، روی این گزینه در منوی Debug کلیک کنید. این گزینه در زمانی که یک برنامه در حالت توقف قرار دارد، قابل استفاده است.

برای استفاده از این گزینه، ابتدا باید پروژه را به گونه‌ای اجرا کنید که در مرحله‌ای از اجرا متوقف شود، سپس در پنجره کد فرم یا ماژول موردنظر مکان‌نما را روی یکی از کاراکترهای نام متغیر یا خاصیت قرار دهید و گزینه Quick Watch ... را از منوی Debug انتخاب کنید. کادرمحاوره Quick Watch مانند شکل ۴-۲۴ نمایش داده می‌شود.



شکل ۴-۲۴

در این کادرمحاوره سه بخش وجود دارد: در بخش Context نام پروژه، ماژول و رویه‌ای که متغیر یا خاصیت مربوطه در آن قرار دارد و در بخش Expression نام متغیر و خاصیت موردنظر که می‌خواهید مقدار آن بررسی شود، نمایش داده می‌شود و در بخش Value مقداری که در متغیر یا خاصیت مربوطه ذخیره شده است، قابل مشاهده خواهد بود.

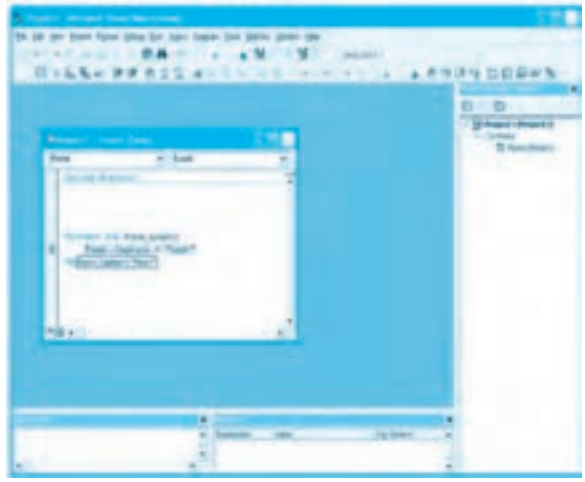
علاوه بر این سه بخش، دکمه فرمان Add نیز اجازه می‌دهد در صورت لزوم عبارت موردنظر را به پنجره Watches اضافه کنید.

نکته: کلید ترکیبی Shift+F9 معادل گزینه ... Quick Watch در منوی Debug است.

نکته: در صورت فعال بودن نوار ابزار Debug در پنجره ویژوال‌بیسیک، می‌توانید از دکمه  در این نوار ابزار استفاده کنید.

نکته: در صورتی‌که بخواهید محتویات یک خاصیت را که همراه با نام کنترل یا فرم تایپ شده است به وسیله گزینه ... Quick Watch ببینید، ابتدا نام خاصیت و نام کنترل را با هم انتخاب کنید. این کار را به‌وسیله عمل کشیدن ماوس یا استفاده از کلید ترکیبی Shift + → یا Shift + ← انجام دهید.

نکته: در صورتی‌که در زمان ایجاد یک نقطه توقف در زمان اجرای یک پروژه اشاره‌گر ماوس را روی نام یک متغیر، خاصیت یا عبارت موردنظر نگه دارید، می‌توانید مقدار متغیر یا خاصیت مربوطه را به‌صورت یک Tooltip مشاهده کنید (شکل ۲۵-۴).



شکل ۲۵-۴

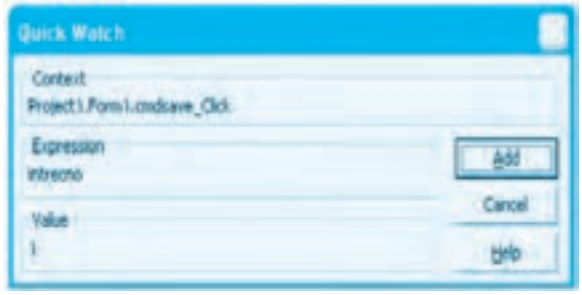
مثال ۸: پروژه numbers را باز کنید و این عملیات را به ترتیب انجام دهید:

۱- پروژه را با استفاده از کلید F۸ به‌صورت خط به خط اجرا کنید تا وارد رویداد cmdsave_Click شوید.

۲- در این رویداد اشاره‌گر ماوس را روی نام متغیر intrecno قرار دهید و کمی صبر کنید. پس از چند لحظه مقدار متغیر intrecno در یک کادر، نمایش داده می‌شود.

۳- مکان‌نما را به دستور Open منتقل کنید و با استفاده از گزینه Run To Cursor برنامه را تا این دستور اجرا نمایید.

۴- در بالای دستور Open نام متغیر intrecno را با عمل کشیدن انتخاب کنید، سپس در پنجره ویژوال‌بیسیک روی منوی Debug و بعد روی گزینه ... Quick Watch کلیک کنید. کادرمحاوره Quick Watch مانند شکل ۲۶-۴ نمایش داده می‌شود. مقدار متغیر intrecno را مشاهده و بررسی کنید.



شکل ۲۶-۴

در این مرحله با استفاده از دکمه Add در این کادرمحاوره متغیر intrecno را به پنجره Watch اضافه کرده و نتیجه را بررسی کنید.

۵- برنامه را با استفاده از کلید F8 خط به خط اجرا کرده و چند عدد دیگر را وارد و ذخیره نمایید و در هر مرحله مقدار متغیرهای intrecno و sngno را به‌وسیله Quick Watch بررسی نمایید.

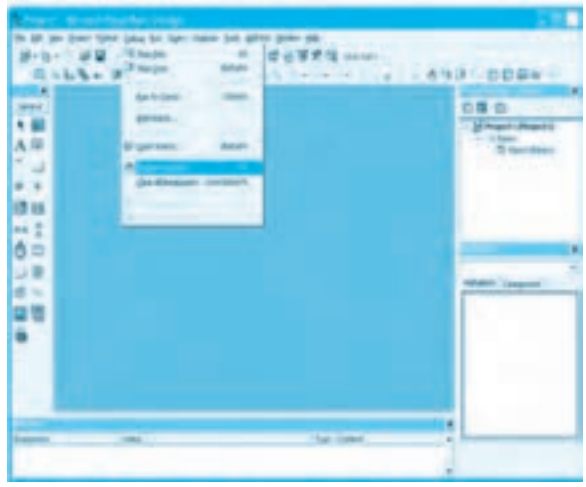
۶- اجرای پروژه را خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

گزینه Toggle Breakpoint

روش دیگری که برای ایجاد توقف در زمان اجرای یک پروژه مناسب است استفاده از نقطه توقف (Breakpoint) می‌باشد.

به منظور ایجاد یک نقطه توقف در هنگام اجرای یک پروژه، ابتدا مکان‌نما را به دستور و خط موردنظر در پنجره کد منتقل کنید، سپس در پنجره ویژوال‌بیسیک روی منوی Debug و سپس گزینه Toggle BreakPoint کلیک کنید.

دستور مربوطه با رنگ زرشکی نمایش داده می‌شود و یک دایره توپر با همین رنگ در سمت چپ دستور مشاهده خواهد شد (شکل ۲۷-۴).



شکل ۲۷-۴

وقتی در زمان اجرای یک پروژه به دستوری که به‌وسیله این گزینه علامت‌گذاری شده است، برسید اجرای پروژه متوقف شده و شما می‌توانید از سایر گزینه‌ها و امکانات موجود در منوی Debug استفاده کنید.

نکته: کلید F9 معادل گزینه Toggle BreakPoint در منوی Debug است.

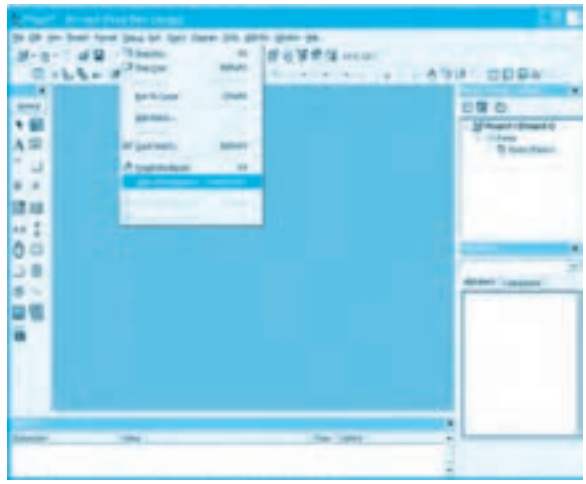
نکته: در صورت فعال بودن نوار ابزار Debug در پنجره ویژوال‌بیسیک می‌توانید از دکمه در این نوار ابزار استفاده کنید.

نکته: **برای ایجاد یک نقطه توقف می‌توانید اشاره‌گر ماوس را به نوار عمودی که در سمت چپ پنجره کد قرار گرفته است منتقل کنید و به موازات دستور موردنظرتان کلیک کنید.**

نکته: **برای از بین بردن یک نقطه توقف می‌توانید یکی از روش‌هایی را که برای ایجاد یک نقطه آن به‌کار می‌رود، استفاده کنید.**

گزینه Clear All Breakpoints

درصورتی‌که بخواهید کلید نقاط توقفی را که ایجاد کرده‌اید، حذف کنید، از گزینه Clear All Breakpoints در منوی Debug استفاده کنید. البته می‌توانید از کلید ترکیبی Ctrl+Shift+F9 نیز استفاده کنید (شکل ۲۸-۴).

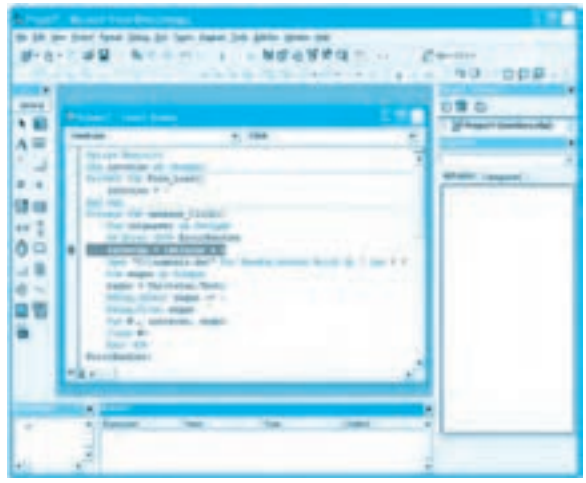


شکل ۲۸-۴

مثال ۹: پروژه numbers را باز کنید و این عملیات را به ترتیب انجام دهید:

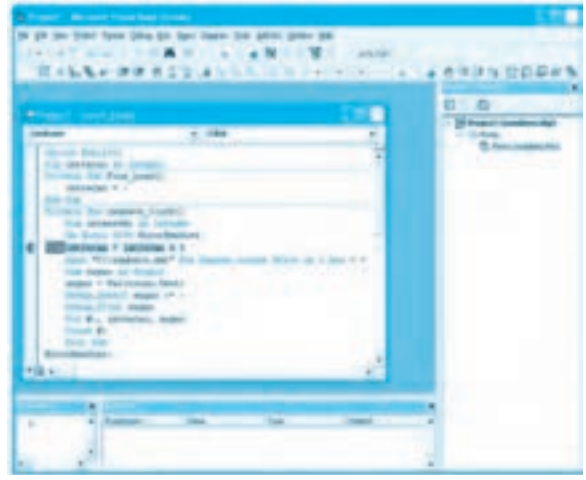
۱- به رویداد cmdsave_Click بروید و مکان‌نما را در کنار دستور intrecno = intrecno+1 قرار دهید.

۲- در پنجره ویژوال‌بیسیک ابتدا روی منوی Debug و سپس گزینه Toggle Breakpoint کلیک کنید. در این مرحله، این دستور مانند شکل ۲۹-۴ به عنوان یک نقطه توقف علامت‌گذاری می‌شود.



شکل ۲۹-۴

۳- پروژه را به‌وسیله کلید F5 یا دکمه ▶ در نوار ابزار استاندارد اجرا کنید. یک عدد در کادر متن تایپ کرده و روی دکمه فرمان Save کلیک کنید. در این مرحله اجرای برنامه در دستوری که علامت‌گذاری شده است، متوقف می‌شود (شکل ۳۰-۴).



شکل ۳۰-۴

۴- مقدار متغیرهای intrecno و sng را با استفاده از پنجره Watches یا Quick Watch بررسی نمایید.

۵- اجرای برنامه را با کلید F5 ادامه دهید و یک عدد دیگر وارد و ذخیره کنید. با کلیک روی دکمه فرمان Save اجرای برنامه دوباره در همان دستوری که علامت‌گذاری کرده‌اید، متوقف می‌شود. به این صورت می‌توانید برنامه را در هر نقطه‌ای که می‌خواهید متوقف و با ابزارهای موجود اجرای برنامه را ادامه دهید یا مقدار متغیرها و خاصیت‌های اشیا را کنترل و بررسی نمایید.

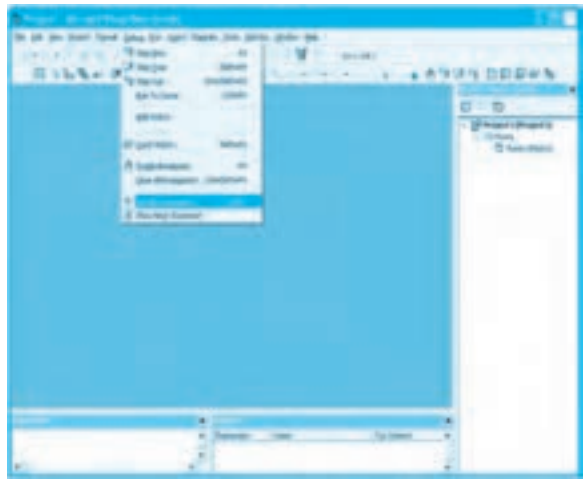
۶- به اجرای پروژه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

در پنجره ویژوال‌بیسیک روی منوی Debug و سپس گزینه Clear All Breakpoints کلیک کنید تا نقطه توقف تعیین شده از بین برود. هم‌چنین می‌توانید مکان‌نما را در نقطه توقف تعیین شده قرار داده و کلید F9 را فشار دهید تا نقطه توقف از بین برود.

تمرین: در پروژه numbers چند نقطه توقف دیگر در رویدادهای cmdsave_Click و cmdsum_Click ایجاد کنید و برنامه را اجرا نموده و در زمان توقف در هر نقطه مقدار متغیرهای استفاده شده در آن رویداد را بررسی کنید.

گزینه Set Next Statement

گاهی اوقات لازم است در هنگام توقف در اجرای یک پروژه و انجام بررسی‌های مورد نیاز، تعدادی از دستورات تا رسیدن به یک دستور معین اجرا نشوند. گزینه Set Next Statement از منوی Debug اجازه می‌دهد تا اجرای پروژه از دستور جاری تا رسیدن به دستور تعیین شده، اجرا نگردد (شکل ۳۱-۴).



شکل ۳۱-۴

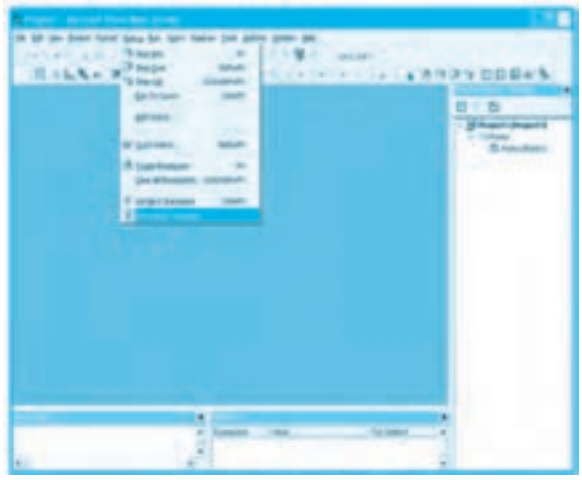
برای تعیین دستور موردنظر فقط کافی است که پس از توقف، مکان‌نما را به دستور موردنظر انتقال دهید و روی این گزینه در منوی Debug کلیک کنید. دستور مربوطه با رنگ زرد به همراه یک فلش در سمت

چپ نمایش داده خواهد شد. اگر در این حالت اجرای پروژه ادامه یابد دستورات تا دستوری که با رنگ زرد مشخص شده است، اجرا نمی‌شوند.

نکته: کلید ترکیبی Ctrl+F9 معادل گزینه Set Next Statement در منوی Debug است.

گزینه Show Next Statement

هنگامی که اجرای پروژه متوقف می‌شود ممکن است وقتی در پنجره کد، بین دستورات مختلف حرکت کنید گزینه Show Next Statement، مکان‌نما را از موقعیت جاری به دستوری که در مرحله بعد آن را اجرا می‌کند، منتقل کند (شکل ۳۲-۴).



شکل ۳۲-۴

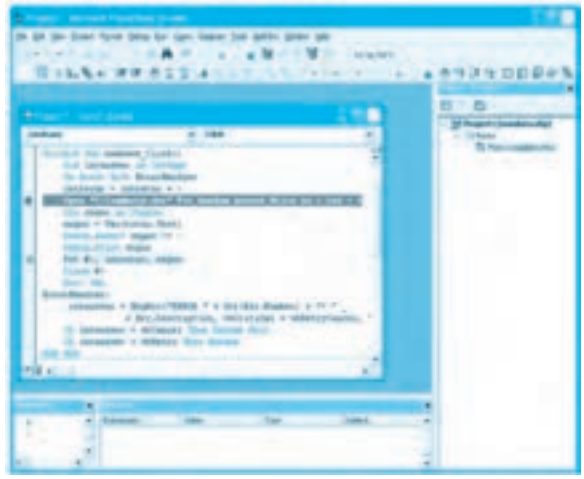
مثال ۱۰: پروژه numbers را باز کرده و این عملیات را به ترتیب انجام دهید:

۱- به رویداد Click دکمه فرمان Save بروید و دستور Open را به عنوان یک نقطه توقف علامت‌گذاری کنید.

۲- پروژه را با کلید F5 اجرا کنید و پس از تایپ یک عدد در کادر متن پنجره برنامه روی دکمه Save کلیک کنید.

۳- در این مرحله، اجرای برنامه با رسیدن به دستور Open متوقف می‌شود.

۴- مکان‌نما را به دستور Put منتقل کرده و در پنجره ویژوال‌بیسیک روی منوی Debug و سپس گزینه Set Next Statement کلیک کنید. اکنون دستور Put با رنگ زرد همراه با یک فلش در سمت چپ آن نمایش داده می‌شود (شکل ۳۳-۴).



شکل ۳۳-۴

۵- در این مرحله اگر مقدار متغیر sngno را بررسی کنید متوجه می‌شوید هیچ تغییری در این متغیر به‌وجود نیامده است چون از دستور Open تا دستور Put هیچ‌یک از دستورات اجرا نشده‌اند. حتی اگر در این وضعیت کلید F8 را برای اجرای دستور Put فشار دهید، خطای زمان اجرا رخ می‌دهد زیرا دستور Open اجرا نشده است تا فایل را برای نوشتن داده‌ها که به‌وسیله دستور Put انجام می‌شود باز کند.

۶- به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید و پروژه را دوباره با استفاده از کلید F5 اجرا کنید. یک عدد در کادر متن پنجره برنامه تایپ کرده و روی دکمه فرمان Save کلیک کنید تا اجرای برنامه در دستور Open متوقف شود.

۷- در پنجره کد به رویداد Click دکمه Sum بروید و مکان‌نما را در یکی از دستورات این رویداد قرار دهید، سپس در پنجره ویژوال‌بیسیک روی منوی Debug و بعد روی گزینه Show Next Statement کلیک کنید.

۸- در این مرحله مکان‌نما به‌طور خودکار دوباره به دستور Open در رویداد Click دکمه فرمان Save منتقل می‌شود و این دستور با رنگ زرد قابل مشاهده است.

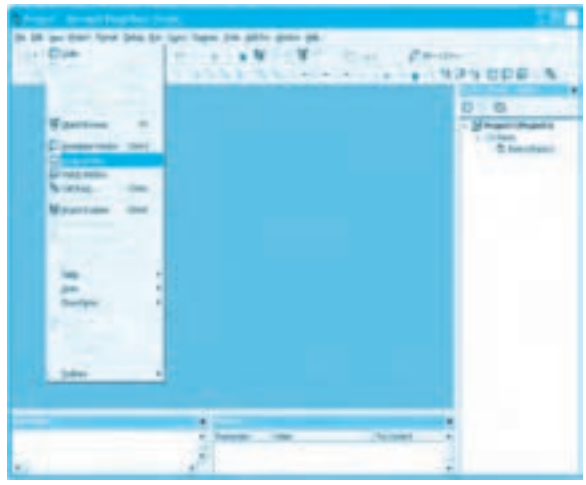
۹- به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید.

۴-۲-۵ پنجره Locals

با استفاده از این پنجره می‌توان مقدار کلیه متغیرها و خاصیت‌های اشیا را در هر ماژول و رویه‌ای که در حال اجرا می‌باشد به‌طور خودکار مشاهده و بررسی نمود.

برای دسترسی به این پنجره در پنجره ویژوال‌بیسیک ابتدا روی منوی View و سپس روی گزینه Locals Window کلیک کنید.

(شکل ۳۴-۴)؛ پنجره Locals با عنوان Locals و با شباهت زیاد با پنجره Watches نمایش داده می‌شود.

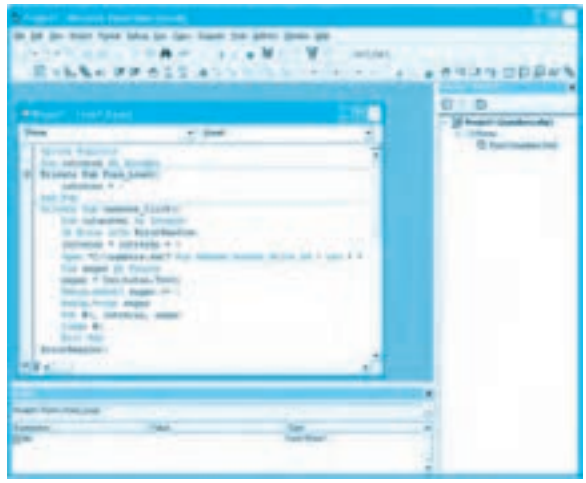


شکل ۳۴-۴

مثال ۱۱: پروژه numbers را باز کنید و این عملیات را به ترتیب انجام دهید:

۱- در پنجره ویژوال‌بیسیک روی منوی Debug و سپس روی گزینه Locals Window کلیک کنید.

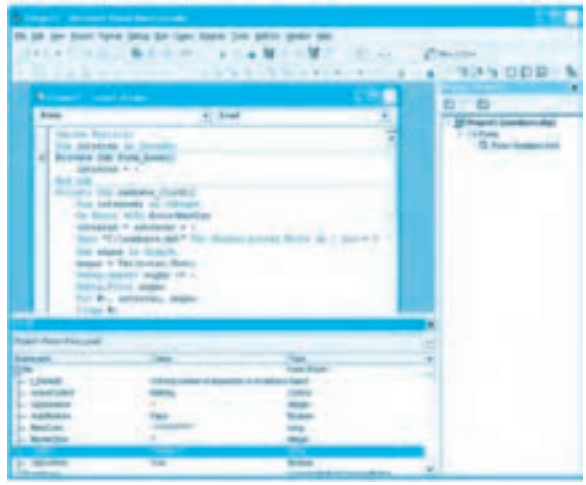
۲- پروژه را با فشردن کلید F8 اجرا کنید؛ در این زمان خط اول از رویداد Load فرم با رنگ زرد نمایش داده می‌شود و پنجره Locals در پایین پنجره ویژوال‌بیسیک قابل مشاهده خواهد بود (شکل ۳۵-۴).



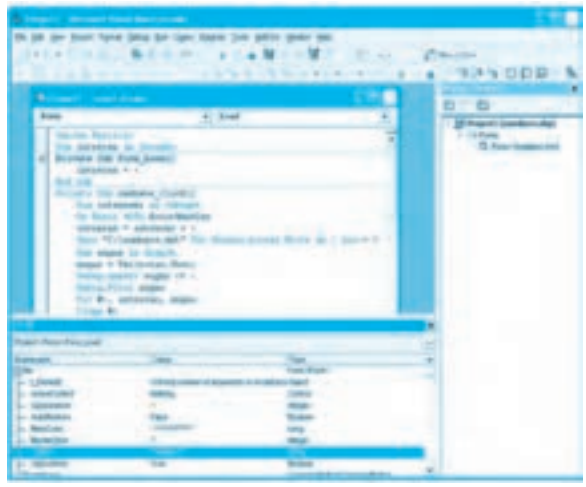
شکل ۳۵-۴

در این مرحله پنجره Locals کلمه Me را در ستون Expression نمایش می‌دهد، اگر روی علامت + در کنار این کلمه کلیک کنید مانند شکل ۴-۳۶ خاصیت‌های فرم برنامه را به همراه مقادیر هریک از آن‌ها نمایش می‌دهد.

اکنون روی دکمه - در کنار کلمه Me کلیک کنید تا ساختار درختی به حالت اول بازگردد.



شکل ۴-۳۶



شکل ۴-۳۷

۳- کلید F8 را چندبار فشار دهید، یک عدد دلخواه در کادر متن در پنجره برنامه تایپ کنید و روی دکمه فرمان Save کلیک کنید. پس از ورود به رویداد Click دکمه فرمان Save، با استفاده از کلید F8 برنامه را خط به خط اجرا کنید و محتویات پنجره Locals را مشاهده نمایید. در این مرحله علاوه بر خاصیت‌های فرم، متغیرهای intanswer و sngno نیز در پنجره Locals نمایش داده می‌شوند (شکل ۳۷-۴).

۴- اجرای برنامه را با استفاده از کلید F8 ادامه دهید تا دستور Exit Sub اجرا شود. در این لحظه اسامی متغیرها به‌طور خودکار از پنجره Locals خارج می‌شوند.

Learn in English

On Error GoTo Statement

The **On Error** statement syntax can have any of the following forms:

Statement	Description
On Error GoTo line	Enables the error-handling routine that starts at line specified in the required line argument. The line argument is any line label or line number. If a run-time error occurs, control branches to line, making the error handler active. The specified line must be in the same procedure as the On Error statement; otherwise, a compile-time error occurs.
On Error Resume Next	Specifies that when a run-time error occurs, control goes to the statement immediately following the statement where the error occurred where execution continues. Use this form rather than On Error GoTo when accessing objects.
On Error GoTo 0	Disables any enabled error handler in the current procedure.

If you don>t use an On Error statement, any run-time error that occurs is fatal; that is, an error message is displayed and execution stops.

An «enabled» error handler is one that is turned on by an On Error statement; an «active» error handler is an enabled handler that is in the process of handling an error. If an error occurs while an error handler is active (between the occurrence of the error and a Resume, Exit Sub, Exit Function, or Exit Property statement), the current procedure>s error handler can>t handle the error. Control returns to the calling procedure. If the calling procedure has an enabled error handler, it is activated to handle the error. If the calling procedure>s error handler is also active, control passes back through previous calling procedures until an enabled, but inactive, error handler is found. If no inactive, enabled error handler is found, the error is fatal at the point at which it actually occurred. Each time the error handler passes control back to a calling procedure, that procedure becomes the current procedure. Once an error is handled by an error handler in any procedure, execution resumes in the current procedure at the point designated by the Resume statement.

Error-handling routines rely on the value in the Number property of the Err object to determine the cause of the error. The error-handling routine should test or save relevant property values in the Err object before any other error can occur or before a procedure that might cause an error is called. The property values in the Err object reflect only the most recent error. The error message associated with Err.Number is contained in Err.Description.

On Error Resume Next causes execution to continue with the statement immediately following the statement that caused the run-time error, or with the statement immediately following the most recent call out of the procedure containing the On Error Resume Next statement. This statement allows execution to continue despite a run-time error. You can place the error-handling routine where the error would occur, rather than transferring control to another location within the procedure. An On Error Resume Next statement becomes inactive when another procedure is called, so you should execute an On Error Resume Next statement in each called routine if you want inline error handling within that routine.

On Error GoTo ۰ disables error handling in the current procedure. It doesn>t specify line ۰ as the start of the error-handling code, even if the procedure contains a line numbered ۰. Without an On Error GoTo ۰ statement, an error handler is automatically disabled when a procedure is exited.

واژه‌نامه

Assert	دفاع‌کردن، آزادکردن
Break Mode	حالت توقف
Breakpoint	نقطه توقف
Compile error	خطای کامپایل (ترجمه)
Debug	خطازدایی
Description	توضیح
Design Mode	حالت طراحی
Determine	تصمیم گرفتن، تعیین کردن
Expression	عبارت
Immediate Window	پنجره اجرای فوری دستورات
Local	محلی، موضعی
Logical error	خطای منطقی
Mode	حالت
Quick	سریع
Run Mode	حالت اجرا
Runtime error	خطای زمان اجرا
Runtimes	روتین، (بخشی از کد)
Rely	اعتماد کردن، تأکید کردن
Relevant	وابسته، مربوط، مناسب
Reflect	منعکس کردن، برگرداندن
Step	گام، مرحله

خطای نوشتاری	Syntax error
مشاهده کردن	Watch
مدیریت خطا	error- handling
زمان ترجمه	Compile- Time
اجرا	Execute
رخ دادن	Occur
واقعه، اتفاق	occurence

خلاصه مطالب

- انواع خطا در ویژوال‌بیسیک عبارتند از: خطاهای نوشتاری، منطقی و زمان اجرا.
- به منظور مدیریت خطاهای زمان اجرا از دستور On Error GoTo استفاده می‌شود.
- شئ Err به منظور مدیریت خطاهای زمان اجرا استفاده می‌شود که دارای دو خاصیت Number و Description است.
- خاصیت Number از شئ Err، شماره خطایی را که رخ داده است، مشخص می‌کند.
- خاصیت Description از شئ Err، یک توضیح در مورد خطایی که روی داده است، ارائه می‌کند.
- دستور Resume Next اجرای برنامه را به دستور بعد از محلی که خطا رخ داده است، منتقل می‌کند.
- دستور Resume دستوری را که سبب ایجاد خطا شده است، مجدداً اجرا می‌کند.
- دستور On Error GoTo ۰ عملکرد دستور On Error GoTo را غیرفعال می‌کند.
- در ویژوال‌بیسیک در هنگام تایپ دستورات، بعضی از خطاهای نوشتاری بلافاصله پس از تایپ اعلام می‌شوند.
- به منظور ادامه اجرای یک برنامه پس از ایجاد حالت توقف یکی از روش‌های زیر مناسب است:

  الف– انتخاب گزینه Continue از منوی Run یا استفاده از دکمه  در نوار ابزار پنجره ویژوال‌بیسیک.

  ب– استفاده از گزینه Restart در منوی Run و اجرای برنامه از ابتدا.
- پنجره اجرای فوری دستورات (Immediate Window) را می‌توان در حالت توقف یا طراحی پروژه باز

نمود و امکان تایپ و اجرای دستورات را به‌طور مستقل فراهم کرد.

- از Debug Object می‌توانید برای ایجاد حالت توقف در برنامه استفاده کنید.
- Debug Object دارای دو متد Assert و Print است.
- متد Assert می‌تواند با بررسی یک عبارت منطقی و مشاهده مقدار False اجرای برنامه را متوقف کند.
- متد Print می‌تواند عبارت یا مقدار موردنظر را در پنجره اجرای فوری دستورات نمایش دهد.
- از گزینه‌های موجود در منوی Debug در پنجره ویژوال‌بیسیک برای کشف و برطرف کردن خطاهای برنامه و بررسی و آزمایش درستی برنامه و اجرای دستورات استفاده می‌شود.
- از گزینه Step Into در منوی Debug یا کلید F8 برای اجرای خط به خط یک برنامه استفاده می‌شود.
- از گزینه Step Over در منوی Debug (یا کلید ترکیبی Shift+F8) برای اجرای خط به خط برنامه استفاده می‌شود با این تفاوت که رویه‌های فراخوانی شده را به‌طور یکجا اجرا می‌کند.
- از گزینه Step Out در منوی Debug (یا کلید ترکیبی Ctrl+Shift+F8) برای اجرای تمامی دستورات باقیمانده در یک رویه در حال اجرا استفاده می‌شود.
- گزینه Run To Cursor از منوی Debug (یا کلید ترکیبی Ctrl+F8)، دستورات را تا محلی که مکان‌نما در آن قرار دارد، اجرا می‌کند.
- با استفاده از گزینه Add Watch در منوی Debug می‌توان مقدار متغیرها، خواص و هر عبارتی را در پنجره Watches در حالت توقف مشاهده کرد.
- با استفاده از گزینه Edit Watch در منوی Debug (یا کلید ترکیبی Ctrl+W) می‌توان متغیر، خواص و عبارت موجود در پنجره Watches را تغییر داد و ویرایش یا حذف کرد.
- گزینه Quick Watch از منوی Debug (کلید ترکیبی Shift+F9) امکان مشاهده مقدار متغیرها، خواص و عبارات را در حالت توقف و اضافه کردن آن‌ها به پنجره Watches فراهم می‌کند.
- درصورتی‌که یک دستور به عنوان نقطه توقف (Breakpoint) علامت‌گذاری شود، وقتی اجرای برنامه به آن دستور برسد اجرای برنامه به‌طور موقت متوقف می‌شود و برنامه در حالت Break قرار می‌گیرد.
- با استفاده از گزینه Toggle Breakpoint از منوی Debug (یا کلید F9) می‌توان یک نقطه توقف ایجاد کرد.
- می‌توانید عملیات زیر را در حالت توقف (Break Mode) انجام دهید:

الف- تغییر و ویرایش دستورات

ب- مشاهده رویه فعالی که فراخوانی شده است.

ج- مشاهده مقادیر متغیرها و خواص کنترل‌ها و فرم‌ها و تغییر مقدار آن‌ها

د- مشاهده و بررسی دستوراتی که بعد از نقطه توقف اجرا می‌شوند.

هـ - اجرای دستورات در پنجره اجرای فوری دستورات

و- مشاهده حالت و شکل ظاهری رابط گرافیکی نرم‌افزار

- روش‌های ایجاد یک نقطه توقف در یک برنامه عبارتند از:

الف- ایجاد یک Breakpoint

ب- استفاده از کلید ترکیبی Ctrl+Break در هنگام اجرای برنامه یا استفاده از دستور Stop در بخش کد برنامه

ج- توقف برنامه در هنگام رخ دادن یک خطای زمان اجرا

د- استفاده از گزینه‌های موجود در کادرمحاوره Add Watch

هـ - استفاده از گزینه Run To Cursor از منوی Debug

و- استفاده از گزینه Break از منوی Run یا استفاده از دکمه  در نوار ابزار پنجره ویژوال‌بیسیک

- گزینه Clear All Breakpoints از منوی Debug (یا کلید ترکیبی Ctrl+Shift+F9) تمام نقاط توقف را از بین می‌برد.
- گزینه Set Next Statement از منوی Debug (یا کلید ترکیبی Ctrl+F9) امکان انتقال اجرای برنامه از نقطه توقف تا رسیدن به یک دستور معین را فراهم می‌کند.
- گزینه Show Next Statement از منوی Debug، مکان‌نما را از موقعیت جاری به دستوری که در مرحله اجرا قرار دارد، منتقل می‌کند.
- پنجره Local در هر لحظه محتویات متغیرها و مقدار خاصیت‌های اشیایی را که در ماژول و رویه‌ای در حال اجرا می‌باشند، به‌طور خودکار نمایش می‌دهد.

آزمون نظری

۱- کدام گزینه برای مشاهده سریع مقدار یک متغیر مناسب است؟

الف- Add Watch ب- Step Over ج- Edit Watch د- Quick Watch

۲- کدام گزینه برای اجرای دوباره دستوری که خطای زمان اجرا را ایجاد کرده است، به‌کار می‌رود؟

الف- Resume Next ب- Debug ج- Resume د- On Error GoTo

۳- کدام گزینه برای دسترسی به شماره خطایی که رخ داده است، به‌کار می‌رود؟

الف- Error ب- Err ج- Number د- Description

۴- از کدام گزینه برای ایجاد حالت توقف استفاده نمی‌شود؟

الف- Ctrl+F8 ب- استفاده از دستور Stop

ج- Ctrl+F2 د- استفاده از دکمه ⏸ در نوار ابزار استاندارد ویژوال‌بیسیک

۵- از کدام پنجره برای اجرای دستورات و مشاهده عملکرد آن‌ها خارج از برنامه استفاده می‌شود؟

الف- Watches ب- Local ج- Immediate د- Debug

۶- کدام گزینه برای اجرای دستوری که بعد از دستور تولیدکننده خطای زمان اجرا قرار دارد مناسب است؟

الف- Err ب- Resume Next ج- Resume د- On Error GoTo

۷- کدام گزینه برای اضافه کردن یک متغیر یا خاصیت به پنجره Watches مناسب است؟

الف- Add Watch ب- Edit Watch ج- Quick Watch د- گزینه‌های الف و ج

۸- کدام گزینه برای اجرای خط به خط برنامه در تمام ماژول و رویه‌ها مناسب‌تر است؟

الف- Step Into ب- Step Over ج- Step Out د- Run To Cursor

۹- کدام گزینه برای اجرای برنامه تا یک دستور معین مناسب است؟

الف- Step Out ب- Run To Cursor ج- Toggle Breakpoint د- گزینه‌های ب و ج

۱۰- کدام پنجره برای مشاهده خودکار مقدار متغیرها و خاصیت‌های اشیا در تمام سطح برنامه مناسب است؟

الف- Watches ب- Locals ج- Immediate د- Quick Watch

11- ..... disable error handling in the current procedure.

a- On Error GoTo Next b- On Error GoTo Resume c- On Error GoTo Line d- On Error GoTo 0

۱۲- انواع خطاها و دلیل به‌وجود آمدن آن‌ها را توضیح دهید.

۱۳- نحوه شناسایی و برطرف کردن خطاهای نوشتاری را توضیح دهید.

۱۴- نحوه شناسایی و برطرف کردن خطاهای منطقی را توضیح دهید.

۱۵- نحوه شناسایی و برطرف کردن خطاهای زمان اجرا را با ذکر مثال توضیح دهید.

۱۶- برنامه ویژوال‌بیسیک در چه حالت‌هایی قابل استفاده است؟ هر حالت را با ویژگی‌های آن توضیح دهید.

۱۷- نقطه توقف و نحوه ایجاد، استفاده و مدیریت آن را توضیح دهید.
۱۸- عملکرد گزینه‌های Step Into، Step Out و Step Over و تفاوت آن‌ها را با یکدیگر بیان کنید.
۱۹- نحوه استفاده از پنجره Watches را توضیح دهید.
۲۰- نحوه استفاده از گزینه Quick Watch در منوی Debug پنجره ویژوال‌بیسیک را توضیح دهید.
۲۱- کاربرد پنجره‌های اجرای فوری دستورات و Locals را بیان کنید.
۲۲- کاربرد گزینه‌های Run To Cursor، Set Next Statement و Show Next Statement را بیان کنید.

آزمون عملی

۱- پروژه‌ای طراحی کنید که کاربر توانایی مشاهده محتویات درایوها و پوشه‌ها و ایجاد و حذف آن‌ها را داشته باشد به‌علاوه در هنگام اجرای برنامه خطاهایی مانند خالی بودن درایو فلاپی یا نادرست بودن مسیر تعیین شده و مانند آن‌ها را به شکل مناسب شناسایی و مدیریت کند و با نمایش کادرهای پیغام مناسب کاربر را راهنمایی و هدایت نماید.
۲- پروژه طراحی شده در تمرین قبل را به وسیله گزینه‌های موجود در منوی Debug خطایابی و آزمایش کنید.
۳- یک پروژه از نوع Standard EXE به همراه یک فرم و یک کنترل دکمه فرمان طراحی کنید، سپس دستورات زیر را در رویه رویداد Click دکمه فرمان بنویسید و عملیات زیر را انجام دهید.

```
Dim i As Integer
Dim j As Integer
Do While (i <= 4)
   j = 1
   Do While (j <= 3)
      Print "*";
      j = j + 1
   Loop
   Print
   i = i + 1
Loop
```

الف- پروژه را با گزینه‌های Step Into، Step Over، Step Out و Return To Cursor اجرا کنید و مقدار متغیرهای i و j را در هنگام اجرای دستورات کنترل کنید.

ب- دو نقطه توقف در رویه رویداد Click دکمه فرمان ایجاد کنید و دوباره پروژه را با کنترل مقدار متغیرهای i و j اجرا کنید.

واحد کار پنجم

هدف جزیی

توانایی استفاده از کنترل‌های پیشرفته

زمان (ساعت)	
نظری	عملی
۴	۸

هدف‌های رفتاری

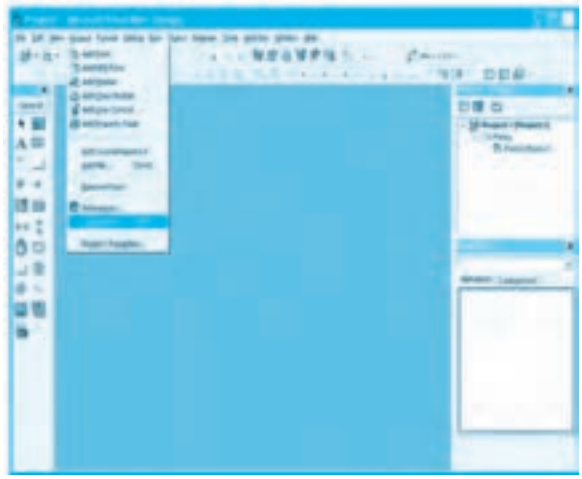
پس از مطالعه این واحد کار از فراگیر انتظار می‌رود که:

۱- بتواند کنترل‌های ActiveX را به پروژه‌ها اضافه کرده و استفاده نماید.

۲- بتواند از کنترل‌های FlatScrollbar، ImageList، ImageCombo، Slider، UpDown، RichTextBox، MSChart، Toolbar، FlexGrid، CommonDialog استفاده کند.

کلیات

علاوه بر کنترل‌های ذاتی، ویژوال‌بیسیک مجموعه‌ای از کنترل‌های از نوع ActiveX را نیز در اختیار برنامه‌نویسان قرار می‌دهد تا با استفاده از آن‌ها رابط‌های گرافیکی زیباتر و مناسب‌تری تهیه نمایند. برای دسترسی به این کنترل‌ها لازم است آن‌ها را به جعبه ابزار ویژوال‌بیسیک اضافه کنید؛ برای این کار می‌توانید در پنجره ویژوال‌بیسیک ابتدا روی منوی Project و سپس روی گزینه Components... کلیک کنید یا از کلید ترکیبی Ctrl+T استفاده نمایید (شکل ۱-۵).



شکل ۱-۵

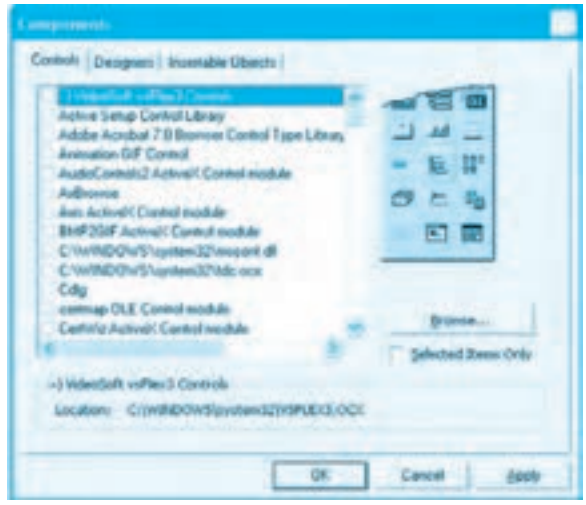
در این مرحله کادرمحاوره Components مانند شکل ۲-۵ نمایش داده می‌شود. در این کادرمحاوره می‌توانید در زبانه Controls و از کادر لیست موجود در این زبانه، کنترل خود را پیدا کرده، روی کادر علامت مربوط به آن کلیک کنید و سپس با کلیک روی دکمه فرمان OK یا Apply آن را به جعبه ابزار اضافه کنید.

در صورتی که بخواهید از سایر کنترل‌های ActiveX که به وسیله خود یا سایر افراد تهیه شده‌اند استفاده کنید، می‌توانید با دکمه فرمان Browse... در این زبانه کنترل خود را پیدا کرده و آن را به جعبه ابزار اضافه کنید.

۱-۵ کنترل FlatScrollBar

ممکن است تاکنون با نوارهای پیمایش یا همان ScrollBar که در برنامه‌ها و پنجره‌های ویندوز کاربرد زیادی دارند، آشنا شده باشید.

از نوارهای پیمایش برای مشاهده محتویات یک پنجره یا متن یا تصویری که تمام آن را نمی‌توان به‌طور همزمان مشاهده نمود، استفاده می‌شود.

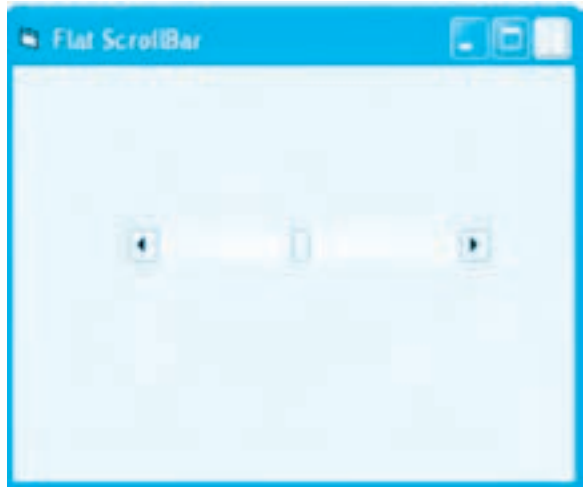


شکل ۲-۵

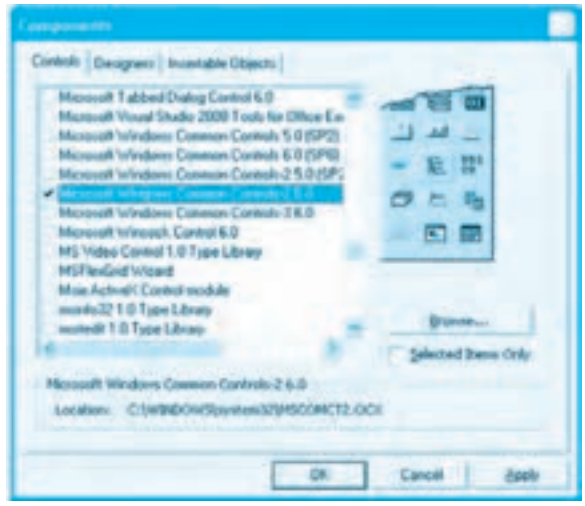
همان‌طورکه در شکل ۳-۵ مشاهده می‌کنید این کنترل از دو دکمه مثلثی شکل و یک دکمه مستطیل شکل متحرک در بین دکمه‌های مثلثی تشکیل شده است.

کاربر می‌تواند با کلیک روی دکمه‌های مثلثی و کشیدن دکمه مستطیل یا کلیک روی نقطه‌ای خالی از نوار کنترل، روی متن، تصویر یا محتویات پنجره حرکت کند.

برای اضافه کردن این کنترل به جعبه ابزار گزینه Microsoft Windows Common Controls-2 6.0 را در کادرمحاوره Components انتخاب کنید (شکل ۴-۵).



شکل ۳-۵



شکل ۴-۵

پس از اضافه کردن این Component به جعبه ابزار، ۵ کنترل جدید به جعبه ابزار ویژوال‌بیسیک اضافه می‌شود که یکی از آن‌ها کنترل نوار پیمایش می‌باشد.

این کنترل علاوه بر رویدادها و خاصیت‌های مشترکی که با سایر کنترل‌ها دارد، دارای رویدادها و خاصیت‌های ویژه‌ای نیز می‌باشد که به بررسی آن‌ها می‌پردازیم.

شکل ۵-۵

خاصیت Appearance

به وسیله این خاصیت می‌توان شکل ظاهری کنترل را تعیین کرد، اگر مقدار این خاصیت برابر با صفر (fsb3D) باشد، کنترل به‌صورت سه بعدی و اگر مقدار آن برابر با ۱ (fsbFlat) باشد، کنترل به‌صورت دو بعدی یا مسطح نمایش داده می‌شود و اگر مقدار آن برابر با ۳ (fsbTrack3D) باشد، کنترل در حالت عادی به‌صورت دو بعدی و وقتی اشاره‌گر ماوس روی دکمه‌های کنترل قرار گیرد، دکمه مربوطه به‌صورت سه بعدی نمایش داده می‌شود (شکل ۶-۵). نحوه استفاده از این خاصیت به این صورت است:

FlatScrollBar نام کنترل.Appearance [=integer]

شکل ۶-۵

integer می‌تواند یکی از مقادیر عددی یا ثابت رشته‌ای را که توضیح داده شد، کسب کند. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت Arrow

به وسیله این خاصیت می‌توان دکمه‌های مثلثی شکل ابتدا و انتهای کنترل را فعال یا غیرفعال کرد. اگر مقدار این خاصیت برابر صفر (CC2Both) باشد هر دو دکمه قابل استفاده می‌باشند و اگر مقدار این خاصیت برابر با ۱ (CC2LeftUp) باشد، فقط دکمه سمت چپ و اگر برابر با ۲ (CC2RightDown) باشد فقط دکمه سمت راست در کنترل فعال و قابل استفاده است. نحوه استفاده از این خاصیت به این صورت است:

FlatScrollBar نام کنترل.Arrows [=integer]

integer می‌تواند یک عدد صحیح یا یک ثابت رشته‌ای از سه مقدار ارایه شده برای خاصیت Arrow باشد. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت‌های LargeChange و SmallChange

به وسیله این دو خاصیت می‌توانید مقدار تغییرات را در هنگام استفاده از کنترل معین کنید. خاصیت LargeChange مقدار تغییرات را هنگامی که کاربر روی مکانی از نوار پیمایش (بجز دکمه‌ها) کلیک کرده،

معین می‌کند و خاصیت SmallChange مقدار تغییرات را هنگامی که کاربر روی دکمه‌های مثلثی کلیک کرده است، تعیین می‌نماید. نحوه استفاده از این دو خاصیت به این صورت است:

FlatScrollBar نام کنترل.LargeChange [=number]

FlatScrollBar نام کنترل.SmallChange [=number]

number یک عبارت عددی از نوع صحیح است که میزان تغییرات را معین می‌کند. اگر از این بخش استفاده نشود، خاصیت خوانده می‌شود. به عنوان مثال دستورات زیر باعث می‌شوند که در صورت کلیک روی نوار پیمایش مقدار تغییرات به اندازه ۱۰ واحد و اگر از دکمه‌های مثلثی استفاده شود، مقدار تغییرات هر بار به اندازه سه واحد باشد.

FlatScrollBar. LargeChange=10

Flat ScrollBar.SmallChange=3

خواص Max و Min

این دو خاصیت مقدار حداکثر (در زمان رسیدن دکمه متحرک به انتهای نوار پیمایش) و مقدار حداقل (در زمان رسیدن دکمه متحرک به ابتدای نوار پیمایش) را در نوار پیمایش معین می‌کنند. نحوه استفاده از این دو خاصیت به این صورت است:

FlatScrollBar نام کنترل.Max [= value]

FlatScrollBar نام کنترل.Min [= value]

value یک عبارت عددی است که مقدار حداکثر و حداقل را معین می‌کند. اگر از این بخش استفاده شود، مقدار خاصیت خوانده می‌شود.

خاصیت Orientation

به وسیله این خاصیت می‌توان نوار پیمایش را به‌صورت افقی یا عمودی نمایش داد. نحوه استفاده از این خاصیت به این صورت است:

FlatScrollBar نام کنترل.Orientation [= integer]

integer یک عبارت عددی یا ثابت رشته‌ای است که اگر مقدار آن صفر (CC2Orientation Horizontal) باشد کنترل به‌صورت افقی و در صورتی که مقدار آن برابر ۱ (CC2Orientation Vertical) باشد، کنترل به‌صورت عمودی نمایش داده می‌شود. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت Value

این خاصیت مقدار فعلی را در نوار پیمایش معین می‌کند. نحوه استفاده از این خاصیت به این صورت است:

FlatScrollBar نام کنترل.Value [= integer]

integer یک عبارت عددی از نوع صحیح است که مقدار حرکت در نوار پیمایش را معین می‌کند. اگر از این

بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

رویداد Change

این رویداد وقتی رخ می‌دهد که مقدار خاصیت Value در کنترل تغییر کند.

رویداد Scroll

عملکرد این رویداد مشابه رویداد Change است، اما با این تفاوت که این رویداد فقط زمانی اجرا می‌شود که کاربر دکمه مستطیل متحرک را در نوار پیمایش، به وسیله عمل کشیدن جابه‌جا کند. در این صورت با حرکت دکمه مستطیلی به وسیله عمل کشیدن بلافاصله و همزمان با عمل کشیدن دستورات موجود در رویداد اجرا می‌شوند.

مثال ۱: می‌خواهیم یک پروژه طراحی کنیم که به وسیله آن بتوان هر تصویر دلخواهی را مشاهده نمود؛ برای این کار این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه با یک فرم و کنترل‌های آن مانند شکل ۷-۵ ایجاد کنید و عرض و ارتفاع فرم را روی مقدار ۵۰۰۰ و نام آن را روی frmpicture تنظیم کنید سپس یک کنترل منو مانند شکل ۶-۵ روی فرم قرار دهید و گزینه‌های آن را تنظیم کنید.

شکل ۷-۵

۲- یک کنترل Image روی فرم قرار دهید و خاصیت‌های نام آن را روی عبارت imgpicture و Stretch آن را روی مقدار True تنظیم کنید.

۳- در این مرحله دو کنترل از نوع FlatScrollBar مانند شکل ۷-۵ روی فرم قرار دهید و خاصیت‌های آن‌ها را به ترتیب روی عبارت‌های fsbhorizontal و fsbvertical تنظیم کنید.

۴- رویداد Click گزینه Open را به این صورت تنظیم کنید:

```
Private Sub mnuopen_Click()
   Dim strpicture As String
   strpicture = InputBox("Enter Path Name :", "open")
   imgpicture.Picture = LoadPicture(strpicture)
   imgpicture.Stretch = False
   fsbhorizontal.Min = 0
   fsbhorizontal.Max = imgpicture.Width
   fsbhorizontal.LargeChange = 10
   fsbhorizontal.SmallChange = 5
   fsbhorizontal.Value = 0
   fsvvertical.Min = 0
   fsvvertical.Max = imgpicture.Height
   fsvvertical.LargeChange = 10
   fsvvertical.SmallChange = 5
   fsvvertical.Value = 0
End Sub
```

در این رویداد ابتدا با نمایش یک کادر ورود داده، نام و مسیر فایل دریافت می‌شود سپس تصویر با استفاده از تابع Loadpicture در کنترل تصویر نمایش داده می‌شود و برای آن‌که تصویر با اندازه واقعی مشاهده شود خاصیت Stretch کنترل تصویر روی مقدار False تنظیم می‌شود. در مرحله خاصیت Min کنترل fsbhorizontal روی مقدار صفر و خاصیت Max آن روی مقدار خاصیت Width کنترل تصویر تنظیم می‌شود تا حداکثر میزان حرکت در نوار پیمایش افقی با توجه به عرض تصویر تنظیم شود. سپس مقدار خاصیت Large Change نوار پیمایش افقی روی مقدار ۱۰ و مقدار خاصیت SmallChange آن روی مقدار ۵ تنظیم می‌شود و در پایان نیز خاصیت Value کنترل که موقعیت اولیه دکمه مستطیلی را مشخص می‌کند روی مقدار صفر نمایش داده می‌شود تا تصویر در ابتدا از گوشه چپ نمایش داده شود. به همین ترتیب خاصیت‌های کنترل نوار پیمایش عمودی نیز روی مقادیر مربوطه تنظیم می‌شوند.

۵- اکنون رویدادهای Change و Scroll دو کنترل نوار پیمایش را به این صورت تنظیم کنید:

```
Private Sub fsbhorizontal_Change()
  imgpicture.Left = -fsbhorizontal.Value
End Sub
```

```
Private Sub fsbhorizontal_Scroll()
  imgpicture.Left = -fsbhorizontal.Value
End Sub

Private Sub fsvvertical_Change()
   imgpicture.Top = -fsvvertical.Value
End Sub
Private Sub fsvvertical_Scroll()
   imgpicture.Top = -fsvvertical.Value
End Sub
```

در این رویدادها برای آن‌که عمل پیمایش روی تصاویر امکان‌پذیر باشد مقدار خاصیت Value کنترل‌ها در خاصیت Left و Top کنترل تصویر قرار می‌گیرد؛ این عمل باعث می‌شود تا در صورت حرکت در هر یک از کنترل‌های نوار پیمایش با توجه به مقدار خاصیت‌های SmallChange و LargeChange تصویر به سمت پایین یا چپ و برعکس حرکت کند. کنترل نوار پیمایش افقی حرکت به سمت چپ و راست را با استفاده از خاصیت Left کنترل تصویر و کنترل نوار پیمایش عمودی حرکت به سمت بالا و پایین را با استفاده از خاصیت Top کنترل تصویر انجام می‌دهند.

۶- پروژه و فرم را با نام Showimage ذخیره کرده و سپس پروژه را اجرا کنید. روی گزینه Open در منوی فایل کلیک کرده، نام و مسیر یک فایل تصویر از نوع Bmp یا Jpg را در کادر ورود داده تایپ کنید و سپس روی دکمه فرمان OK کلیک کنید.

۷- پس از نمایش تصویر در صورتی‌که تمام تصویر قابل مشاهده نباشد با استفاده از نوارهای پیمایش افقی و عمودی تصویر را پیمایش کنید.

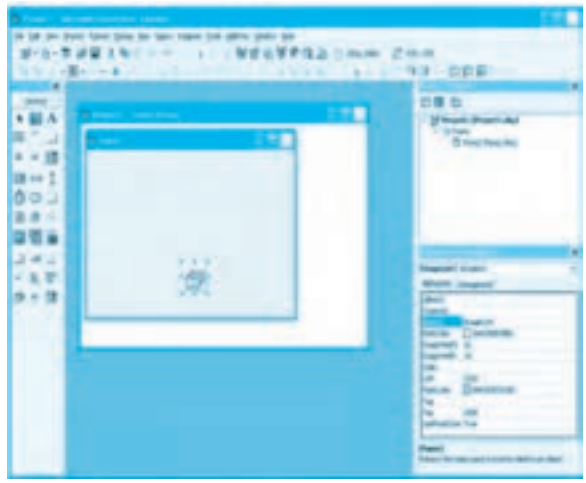
برای این کار می‌توانید روی دکمه‌های مثلثی در دو طرف کنترل‌های نوار پیمایش کلیک کنید یا عمل کشیدن را روی دکمه مستطیلی انجام دهید یا روی مکانی از نوار پیمایش کلیک کنید و نتیجه را در حالت‌های مختلف بررسی نمایید.

۸- به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه Showimage را به شکلی تنظیم کنید که اگر ابعاد تصویر از ابعاد فرم کوچک‌تر باشد نوارهای پیمایش نمایش داده نشوند به‌علاوه برنامه این امکان را داشته باشد تا کاربر بتواند مقدار تغییرات بزرگ و کوچک در نوارهای پیمایش را مطابق میل خود تنظیم کند.

۵-۲ کنترل ImageList

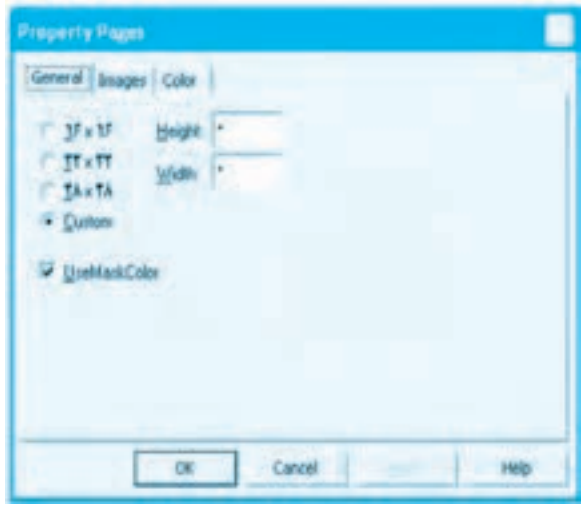
این کنترل به تنهایی کاربردی ندارد و معمولاً ایجاد و طراحی کنترل‌های دیگری نظیر کنترل‌های ToolBar، ImageCombo و غیره مورد استفاده قرار می‌گیرند. این کنترل می‌تواند مجموعه‌ای از تصاویر را در خود نگهداری کند تا در صورت لزوم این تصاویر در سایر کنترل‌ها مورد استفاده قرار گیرند (شکل ۵-۸). این کنترل هنگام اجرای برنامه دیده نمی‌شود و شامل هیچ رویدادی نیست و مهم‌ترین خاصیت آن، خاصیت Custom است که به وسیله آن می‌توانید تصاویر خود را انتخاب و به تصاویر موجود در کنترل اضافه کنید.



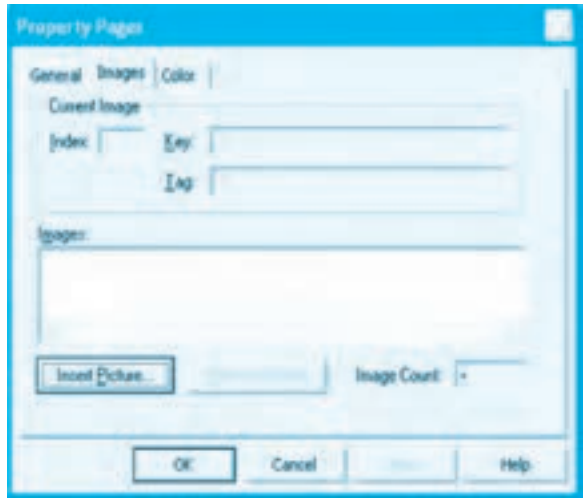
شکل ۵-۸

برای اضافه کردن این کنترل به جعبه ابزار ویژوال‌بیسیک در نوار منوی ویژوال‌بیسیک روی منوی Project کلیک کنید و گزینه Component را انتخاب نمایید و از کادرمحاوره‌ای که ظاهر می‌شود زبانه Control و سپس گزینه Microsoft Windows Common Controls 6.0 (SP6) را انتخاب کنید و روی دکمه OK کلیک نمایید. ۹ کنترل جدید به جعبه ابزار ویژوال‌بیسیک اضافه می‌شوند (شکل ۵-۹). برای قرار دادن تصاویر در کنترل، ابتدا در پنجره خواص روی دکمه ... در روبه‌روی خاصیت Custom کنترل ImageList کلیک کنید، در این صورت کادرمحاوره‌ای مانند شکل ۵-۱۰ ظاهر خواهد شد. اگر در این کادرمحاوره در زبانه Images روی دکمه فرمان Insert Picture... کلیک کنید (شکل ۵-۱۱)، کادرمحاوره دیگری به نام Select picture ظاهر می‌شود که به وسیله آن می‌توانید تصاویر مورد نیاز خود را پیدا کرده و با کلیک روی دکمه فرمان Open آن را به لیست تصاویر در کنترل اضافه کنید.

شکل ۹-۵



شکل ۱۰-۵



شکل ۱۱-۵

هم‌چنین می‌توانید با انتخاب تصویر موردنظر در کادرمحاوره Property Pages و کلیک روی دکمه فرمان Remove Picture، تصویر مربوطه را از لیست تصاویر حذف کنید، هر یک از تصاویر به ترتیب ورود به لیست، شماره‌ای دریافت می‌کنند که از عدد یک آغاز شده و در کادر متن Index در کادرمحاوره Property Pages نمایش داده می‌شوند. این عدد در زمان انتخاب تصاویر از کنترل ImageList، تصویر موردنظر را معین می‌کند.

نکته: برای تنظیم ابعاد تصاویر در کنترل ImageList می‌توانید در پنجره خواص، خاصیت (Custom) را پیدا کرده و روی دکمه ... روبه‌روی این خاصیت کلیک کنید تا کادرمحاوره Property Pages مانند شکل ۱۰-۵ نمایش داده شود. در زبانه General در این کادرمحاوره می‌توانید روی یکی از دکمه‌های انتخاب کلیک کرده و اندازه موردنظر خود را انتخاب کنید.

نکته: کنترل ImageList در زمان اجرا، نمایش داده نمی‌شود.

مهم‌ترین خاصیت این کنترل خاصیت ListImage است؛ به وسیله این خاصیت و با استفاده از متد Add در آن می‌توان تصاویر موردنظر خود را با تابع Loadpicture به کنترل اضافه کرد.

نحوه استفاده از این متد به این صورت است:

نام کنترل Image List - List Images Add „ Load Picturel ("نام و مسیر فایل تصویر")

به عنوان مثال این دستورات سه تصویر از نوع این ico را در کنترل ImageList قرار می‌دهد.

```
ImageList1.ListImages.Add , , LoadPicture("c:\BICYCLE.ICO")
ImageList1.ListImages.Add , , LoadPicture("c:\PLANE.ICO")
ImageList1.ListImages.Add , , LoadPicture("c:\ROCKET.ICO")
```

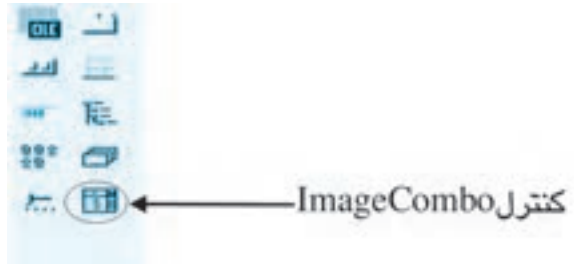
۵-۳ کنترل ImageCombo

اگر گاهی اوقات بخواهید اسامی موجود در کادرهای لیست را همراه با یک تصویر نمایش دهید، کنترل‌هایی نظیر ListBox و ComboBox چنین امکانی را فراهم نمی‌کنند.

به وسیله کنترل Image Combo می‌توانید چنین کادر لیست‌هایی را ایجاد کنید، البته برای نمایش تصاویر در این کنترل باید از یک کنترل ImageList استفاده کرد (شکل ۱۲-۵). برای آن‌که این کنترل را به جعبه ابزار اضافه کنید ابتدا روی منوی Project در پنجره ویژوال‌بیسیک و سپس روی گزینه Components کلیک کنید.

در کادرمحاوره‌ای که ظاهر می‌شود روی زبانه Controls و بعد از آن روی کادر علامت Microsoft Windows Common Controls 6.0 (SP6)کلیک کنید، در پایان روی دکمه فرمان OK کلیک کنید. این کنترل به همراه چند کنترل دیگر از جمله کنترل ImageList به جعبه ابزار اضافه می‌شوند (شکل ۱۳-۵).

شکل ۱۲-۵



شکل ۱۳-۵

در این‌جا خاصیت‌ها و رویدادهای این کنترل بررسی می‌شود.

خاصیت ImageList

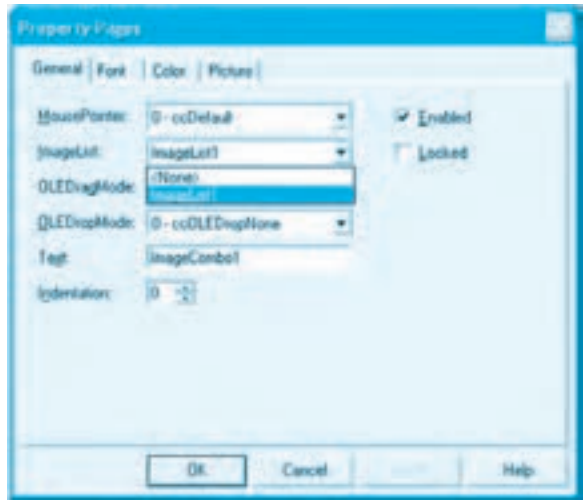
برای آن‌که بتوانید کادر لیستی از تصاویر ایجاد کنید باید به وسیله این خاصیت یک کنترل ImageList را که حاوی یک یا مجموعه‌ای از تصاویر باشد، انتخاب کنید. برای انجام این عمل می‌توانید از خاصیت ImageList در کنترل ImageCombo استفاده کنید؛ نحوه استفاده از این خاصیت به این صورت است:

نام کنترل ImageList = ImageList.نام کنترل ImageCombo

به عنوان مثال این دستور تصاویر موجود در کنترل ImageList را در اختیار کنترل ImageCombo 1 قرار می‌دهد.

```
ImageCombo1.ImageList = ImageList1
```

نکته: برای تنظیم این خاصیت در زمان طراحی می‌توانید در پنجره خواص، خاصیت (Custom) را پیدا کرده و روی دکمه ... که روبه‌روی آن قرار دارد، کلیک کنید. کادرمحاوره Property Pages مانند شکل ۱۴-۵ نمایش داده می‌شود. در این کادرمحاوره در زبانه General لیست بازشو ImageList را باز کنید و نام کنترل ImageList موردنظر را انتخاب کرده و در پایان روی دکمه OK کلیک کنید.



شکل ۱۴-۵

خاصیت ComboItems

به وسیله این خاصیت می‌توان تصاویر موردنظر خود را که قبلاً در یک کنترل ImageList ذخیره شده‌اند به کادر لیست کنترل اضافه نمود. این خاصیت از نوع کلکسیون می‌باشد و شامل متد Add است که برای اضافه کردن تصاویر به کادر لیست به‌کار می‌رود.

نحوه استفاده از این متد به این صورت است:

ImageList اندیس تصویر در کنترل و "عنوان تصویر" ,, Combo Items Add ,کنترل نام ImageCombo

به عنوان مثال این دستورات چهار تصویر را از یک کنترل ImageList به کنترل ImageCombo اضافه می‌کنند.

```
ImageCombo1.ImageList = ImageList1
ImageCombo1.ComboItems.Add , , "BICYCLE", 1
ImageCombo1.ComboItems.Add , , "PLANE", 2
ImageCombo1.ComboItems.Add , , "ROCKET", 3
ImageCombo1.ComboItems.Add , , "CARS", 4
```

در این دستورات ابتدا کنترل ImageList1 به عنوان تأمین‌کننده تصاویر برای کنترل ImageCombo معرفی می‌شود، سپس با متد Add خاصیت ComboItems این تصاویر یکی‌یکی به کنترل ImageCombo اضافه می‌شوند. آرگومان اول و دوم در متد Add اختیاری است. آرگومان سوم عنوان هر تصویر را مشخص می‌کند و آرگومان چهارم اندیس تصویری را که در کنترل ImageList قرار دارد برای نمایش در کنترل ImageCombo تعیین می‌کند.

نکته: اگر تصویری که شماره اندیس آن در متد Add تعیین می‌شود در کنترل ImageList انتخاب شده نباشد عنوان بدون تصویر به کادر لیست تصاویر اضافه می‌شود.

خاصیت Locked

این خاصیت از نوع منطقی است و وقتی مقدار آن برابر با True باشد، کاربر توانایی تایپ مقادیر خود را در کادر لیست تصویر نخواهد داشت و اگر مقدار آن روی False تنظیم شده باشد، کاربر می‌تواند مقادیر موردنظر خود را در کنترل تایپ کند. نحوه استفاده از آن به این صورت می‌باشد.

نام کنترل ImageCombo.Locked [= Boolean]

Boolean یک مقدار منطقی است که می‌تواند True یا False باشد اگر این بخش حذف شود، مقدار خاصیت خوانده می‌شود. به عنوان مثال این دستور کنترل ImageCombo1 را به شکلی تنظیم می‌کند تا امکان تایپ در آن وجود نداشته باشد.

```
ImageCombo1.Locked = True
```

خاصیت SelectedItem

این خاصیت عنوان تصویری را که کاربر انتخاب کرده است، مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل ImageCombo.SelectedItem

به عنوان مثال این دستور عنوان تصویر انتخاب شده در کنترل ImageCombo1 را در کنترل برچسب نمایش

می‌دهد.

```
Label1.Caption = ImageCombo1.SelectedItem
```

خاصیت Index

این خاصیت شماره ترتیب تصویری را که کاربر از کادر لیست تصاویر انتخاب کرده است، بازمی‌گرداند. شماره‌ها از یک (برای اولین گزینه) آغاز شده و به ترتیب گزینه‌ها افزایش می‌یابد. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل ImageCombo.SelectedItem.Index

به عنوان مثال این دستور شماره تصویری را که در کنترل ImageCombo1 انتخاب شده است، در کنترل برچسب نمایش می‌دهد.

```
Label1.Caption = ImageCombo1.SelectedItem.Index
```

خاصیت Text

این خاصیت نام گزینه‌ای را که کاربر از لیست انتخاب کرده یا مقداری را که در کادر متن کنترل لیست تصویر تایپ شده است، مشخص می‌کند.

نحوه استفاده از این خاصیت به این صورت است:

ImageCombo.Text [=**String**]

String یک عبارت از نوع رشته‌ای است که عنوان تصویر را در کادر لیست تصاویر مشخص می‌کند. اگر این بخش حذف شود، مقدار خاصیت خوانده می‌شود؛ به عنوان مثال این دستورات عنوان تصویر انتخاب شده را در کنترل برچسب نمایش می‌دهد.

```
ImageCombo1.Text = "BICYCLE"
Label1.Caption = ImageCombo1.Text
```

رویداد Change

این رویداد عملکردی مشابه رویداد Change در کنترل کادر متن دارد و وقتی کاراکتری در کنترل کادر لیست تصویر تایپ می‌شود، اجرا می‌شود.

رویداد GotFocus

این رویداد زمانی اجرا می‌شود که فوکوس به کنترل می‌رسد به عنوان مثال در این دستورات این رویداد در کنترل ImageCombo۱ به شکلی تنظیم شده است تا با دریافت فوکوس به وسیله این کنترل تصویر پیش‌فرض تنظیم شود.

```
Private Sub ImageCombo1_GotFocus()
```

```
    ImageCombo1.Text = "BICYCLE"
End Sub
```

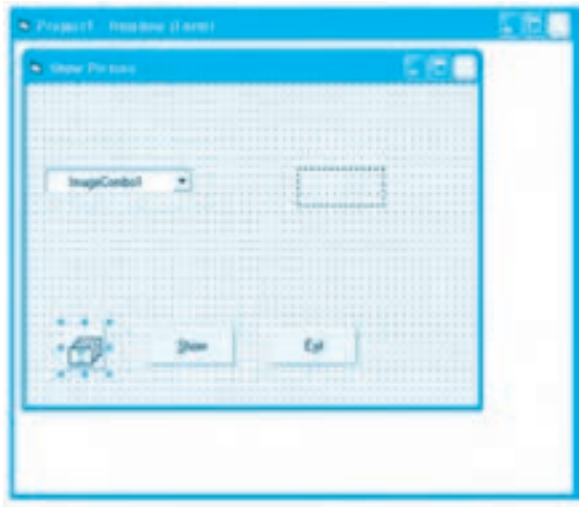
رویداد LostFocus

این رویداد زمانی اجرا می‌شود که کنترل فوکوس خود را از دست می‌دهد؛ به عنوان مثال در این دستورات، این رویداد به شکلی تنظیم شده است تا با انتقال فوکوس عنوان تصویر انتخاب شده در کنترل برچسب نمایش داده شود.

```
Private Sub ImageCombo1_LostFocus()
    Label1.Caption = ImageCombo1.Text
End Sub
```

مثال ۲: می‌خواهیم پروژه‌ای طراحی کنیم تا کاربر بتواند تصاویر خود را از کادر لیست تصویر انتخاب کرده و در یک کنترل تصویر آن را مشاهده کند؛ برای انجام این مثال این عملیات بعد را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه با یک فرم با نام frmshow و عنوان Show Picture مانند شکل ۱۵-۵ ایجاد کنید.



شکل ۱۵-۵

۲- یک کنترل تصویر (Image) با نام imgpicture و دو کنترل دکمه فرمان با نام‌های cmdshow و cmdexit مانند شکل ۱۵-۵ روی فرم قرار داده و تنظیم نمایید.

شکل ۱۶-۵

۳- یک کنترل ImageList با نام ilspictures روی فرم قرار داده و با استفاده از خاصیت Custom چند تصویر را در آن قرار دهید (شکل ۱۶-۵).

نکته: در صورتی‌که بخواهید از تصاویر این مثال استفاده کنید، می‌توانید به پوشه Industry در مسیر زیر بروید و فایل‌های آن را در پوشه پروژه خود کپی کنید با فرض آن‌که برنامه ویژوال‌استودیو در درایو C: نصب شده است.

C:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Industry

۴- یک کنترل ImageCombo با نام imgcbopictures روی فرم قرار دهید.

۵- رویداد Load را به این صورت تنظیم کنید:

```
Private Sub cmdshow_Click()
   Dim strpath As String
   Select Case imgcbopictures.SelectedItem.Index
      Case Is = 1
         strpath = "BICYCLE.ICO"
      Case Is = 2
         strpath = "PLANE.ICO"
```

```
    Case Is = 3
      strpath = "ROCKET.ICO"
    Case Is = 4
      strpath = "CARS.ICO"
  End Select
  imgpicture.Picture = LoadPicture(strpath)
End Sub
```

در این رویداد ابتدا کنترل ilspictures به عنوان کنترل تأمین‌کننده تصاویر برای کنترل کادر لیست تصویر انتخاب می‌شود. سپس با استفاده از چند متد Add تصاویر به کادر لیست تصویر اضافه می‌شوند.

در مرحله بعد خاصیت Locked روی مقدار True تنظیم می‌شود تا امکان تایپ در کادر متن کنترل کادر تصویر وجود نداشته باشد و دستور آخر، تصویر پیش‌فرض را برای نمایش در کنترل کادر تصویر انتخاب می‌کند. در این دستور از کلکسیون ComboItems استفاده می‌شود. در این کلکسیون می‌توان با شماره اندیس هر تصویر در کنترل به خاصیت‌های هر عضو در کنترل کادر لیست دسترسی پیدا نمود. هر عضو در این کلکسیون دارای خاصیتی به نام Selected می‌باشد اگر این خاصیت برای هر عضو تنظیم شود تصویر مربوط به همان عضو به‌طور پیش‌فرض در کنترل نمایش داده می‌شود، بنابراین دستور آخر سومین تصویر را در زمان نمایش فرم به عنوان گزینه پیش‌فرض نمایش می‌دهد.

۶- اکنون رویداد کلیک دکمه فرمان Show را به این صورت تنظیم کنید:

```
Private Sub cmdshow_Click()
  Dim strpath As String
  Select Case imgcbopictures.SelectedItem.Index
    Case Is = 1
      strpath = "BICYCLE.ICO"
    Case Is = 2
      strpath = "PLANE.ICO"
    Case Is = 3
      strpath = "ROCKET.ICO"
    Case Is = 4
      strpath = "CARS.ICO"
  End Select
```

```
    imgpicture.Picture = LoadPicture(strpath)
End Sub
```

در این رویداد با استفاده از دستور Select Case و خاصیت index تصویر انتخاب شده، نام تصویر در متغیر strpath ذخیره می‌شود سپس تابع LoadPicture با استفاده از این متغیر، تصویر را در کنترل تصویر نمایش می‌دهد.

نکته: در صورتی که فایل‌های تصویر در محل دیگری غیر از پوشه پروژه باشند، لازم است نام و مسیر فایل‌ها به‌طور کامل نوشته شوند.

۷- پروژه را با نام ShowPictures ذخیره و سپس آن را اجرا کنید. در این لحظه فرم نمایش داده شده و سومین تصویر به عنوان تصویر پیش‌گزیده در کنترل کادر لیست تصویر نمایش داده می‌شود.

۸- در این مرحله کادر لیست تصویر را انتخاب کرده و یکی از تصاویر را به دلخواه انتخاب و روی دکمه فرمان Show کلیک کنید. اکنون تصویر انتخاب شده در کنترل تصویر قابل مشاهده می‌باشد (شکل ۱۷-۵).

شکل ۱۷-۵

۹- به اجرای پروژه پایان دهید و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه showpictures را به شکلی تنظیم کنید تا در هنگام اجرای پروژه تصویر پیش‌گزیده در

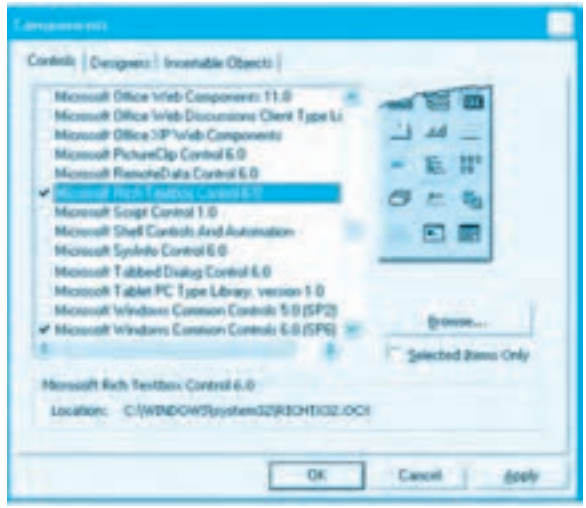
کنترل کادر لیست تصویر در کنترل تصویر نمایش داده شود به‌علاوه وقتی کنترل کادر لیست تصویر فوکوس خود را از دست می‌دهد به‌طور خودکار تصویر انتخاب شده در کنترل تصویر نمایش داده می‌شود. در ضمن می‌توان نام و مسیر فایل انتخاب شده را در یک کنترل برچسب مشاهده کرد.

۵-۴ کنترل RichTextBox

کنترل RichTextBox یکی دیگر از کنترل‌های ویژوال‌بیسیک است که اجازه کار روی داده‌های متنی را فراهم می‌کند. این کنترل اجازه ورود و ویرایش داده‌های متنی را با امکانات بیشتری نسبت به کنترل TextBox در اختیار کاربر قرار می‌دهد. به عنوان مثال می‌توان انواع قالب‌بندی‌های متن مثل حالت Bold، Italic، Font، اندازه، رنگ قلم و غیره را به بخشی از متن موجود در کنترل اعمال کرد.

علاوه بر این به وسیله این کنترل می‌توان فایل‌هایی با قالب‌بندی RTF و متنی را باز و ذخیره کرد.

برای اضافه کردن کنترل RichTextBox به جعبه ابزار، کادرمحاوره Components را باز کنید و گزینه Microsoft RichTextbox Control 6.0 را انتخاب کنید (شکل ۱۸-۵) تا کنترل RichTextBox به جعبه ابزار ویژوال‌بیسیک اضافه شود.



شکل ۱۸-۵

این کنترل دارای خاصیت‌ها و متدهای ویژه‌ای است که در این‌جا بررسی می‌شود.

خاصیت FileName

به وسیله این خاصیت می‌توانید نام و مسیر فایل موردنظر را که می‌خواهید محتویات آن را در کنترل نمایش دهید، تعیین کنید. نحوه استفاده از این خاصیت به این صورت است:

RichTextBox نام کنترل.Filename [= path]

path یک عبارت رشته‌ای است که بیانگر مسیر و نام فایل موردنظر می‌باشد. اگر از این بخش استفاده نشود، مقدار خاصیت خوانده می‌شود. به عنوان مثال این دستور محتویات فایل letter.txt را در کنترل نمایش می‌دهد.

```
RichTextBox1.FileName = "c:\letter.txt"
```

خاصیت Locked

این خاصیت از نوع منطقی می‌باشد. اگر مقدار آن روی True تنظیم شود محتویات کنترل، قابل ویرایش نیست و در صورتی‌که مقدار آن روی False تنظیم شود، متن موجود در کنترل قابل ویرایش است. نحوه استفاده از این خاصیت به‌صورت زیر است:

RichTextBox نام کنترل.Locked [= boolean]

boolean یک مقدار منطقی است که می‌تواند True یا False باشد. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود. به عنوان مثال این دستور امکان ویرایش متن در کنترل RichTextBox را از بین می‌برد.

```
RichTextBox1.Locked = True
```

خاصیت MaxLength

این خاصیت حداکثر تعداد کاراکترها را در کنترل معین می‌کند. مقدار پیش‌فرض صفر است که با توجه به میزان حافظه سیستم، حداکثر تعداد کاراکترها معین می‌شود. نحوه استفاده از این خاصیت به‌صورت زیر است:

RichTextBox نام کنترل.MaxLength [= long]

long یک مقدار عددی از نوع اعداد صحیح بلند (LongInteger) می‌باشد. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت MultiLine

این خاصیت از نوع منطقی است. اگر مقدار این خاصیت روی True تنظیم شده باشد، کنترل توانایی دریافت و نمایش متن را در چندین خط دارد، ولی اگر مقدار این خاصیت روی False تنظیم شود از یک خط جهت نمایش متن استفاده می‌شود. فقط در زمان طراحی قابل تنظیم می‌باشد.

خاصیت ScrollBars

این خاصیت می‌تواند نوارهای پیمایش افقی و عمودی را در کنترل فعال یا غیرفعال کند. این خاصیت فقط در زمان طراحی قابل تنظیم است و یکی از مقادیر عددی یا رشته‌ای را مطابق جدول ۱-۵ می‌پذیرد.

نکته: قبل از استفاده از این خاصیت به منظور نمایش نوارهای پیمایش، خاصیت MultiLine را روی مقدار True تنظیم کنید.

جدول ۱-۵ مقادیر مربوط به خاصیت ScrollBars

مثال	ثابت عددی	ثابت رشته‌ای
کنترل بدون نوار پیمایش	۰	rtfNone
کنترل با نوار پیمایش افقی	۱	rtfHorizontal
کنترل با نوار پیمایش عمودی	۲	rtfVertical
کنترل با هر دو نوار پیمایش	۳	rtfBoth

خاصیت TextRTF

این خاصیت شبیه به خاصیت Text در کنترل TextBox است و متن موجود در کنترل را تنظیم و نگهداری می‌کند. نحوه استفاده از این خاصیت به‌صورت زیر است:

نام کنترل RichTextBox.TextRTF [= string]

string یک عبارت رشته‌ای است که به متن موجود در کنترل اشاره می‌کند. درصورتی‌که این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

متد LoadFile

به وسیله این متد می‌توان محتویات یک فایل متنی یا RTF را در کنترل نمایش داد. نحوه استفاده از این متد به‌صورت زیر است:

نام کنترل RichTextBox.LoadFile pathname , filetype

pathname یک عبارت رشته‌ای است که به نام و مسیر فایل اشاره می‌کند و filetype می‌تواند یکی از مقادیر موجود در جدول ۲-۵ باشد.

جدول ۲-۵ مقادیر مربوط به filetype

توضیح	ثابت عددی	ثابت رشته‌ای
فایل RTF (پیش‌فرض)	۰	rtfRTF
فایل متنی	۱	rtfText

به عنوان مثال این دستور محتویات فایل letter.txt را در کنترل بارگذاری کرده و نمایش می‌دهد.

```
RichTextBox1.LoadFile "c:\letter.txt", rtfText
```

متد SaveFile

به وسیله این متد می‌توان محتویات موجود در کنترل را به‌صورت یک فایل RTF یا متنی روی دیسک ذخیره کرد. نحوه استفاده از این خاصیت به‌صورت زیر است:

نام کنترل RichTextBox.SaveFile pathname , filetype

آرگومان pathname یک عبارت رشته‌ای است که به نام و مسیر فایل اشاره می‌کند و filetype می‌تواند یکی از مقادیر موجود در جدول ۲-۵ باشد.

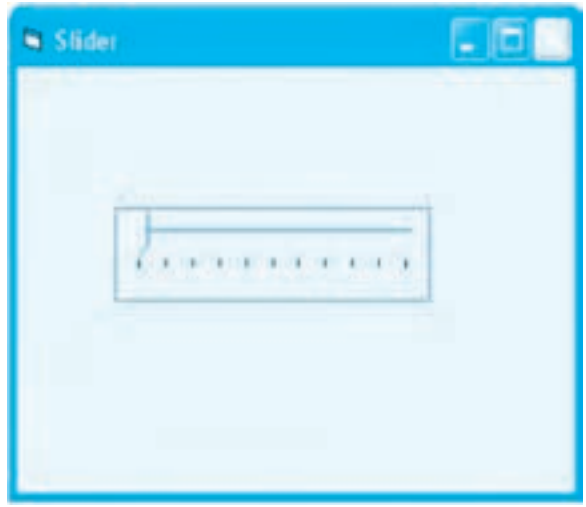
به عنوان مثال این دستور محتویات کنترل RichTextBox را در فایل test.txt در ریشه درایو d:\ ذخیره می‌کند:

```
RichTextBox1.SaveFile "d:\test.txt", rtfText
```

تمرین: پروژه‌ای از نوع MDI و با نام myeditor طراحی کنید که به وسیله کنترل RichTextBox بتوان یک فایل متنی جدید ایجاد نمود یا یک فایل متنی را باز کرد، ویرایش نمود و تغییرات را ذخیره کرد.

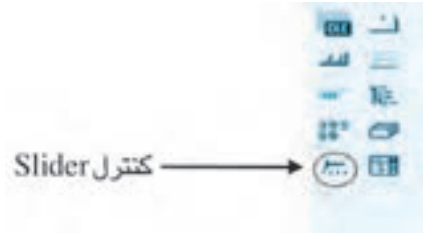
۵-۵ کنترل Slider

یکی دیگر از کنترل‌های ActiveX در ویژوال‌بیسیک، کنترل Slider است (شکل ۱۹-۵). به وسیله این کنترل می‌توانید مقادیر موردنظر خود را برای بخش‌های مختلف برنامه، تنظیم کنید مانند کنترل‌های Slider که در کادرمحاوره Display Properties ویندوز در زبانه Settings برای تنظیم وضوح صفحه‌نمایش استفاده می‌شود. برای تنظیم مقدار در این کنترل می‌توانید روی کنترل و درجات تقسیم‌بندی کلیک کنید یا به وسیله فشردن کلیدهای Page Up یا Page Down مقدار آن را تغییر دهید، علاوه بر این می‌توانید این کار را به وسیله کلیدهای جهت‌دار چپ، راست یا انجام عمل کشیدن روی دکمه تنظیم مقدار روی کنترل انجام دهید.



شکل ۱۹-۵

برای اضافه کردن این کنترل به جعبه ابزار در کادرمحاوره Components گزینه Microsoft Windows Common Controls 6.0 (SP6)را انتخاب کنید تا کنترل Slider به جعبه ابزار ویژوال‌بیسیک اضافه شود (شکل ۲۰-۵).



شکل ۲۰-۵

این کنترل دارای خواص مشابهی مانند کنترل‌های نوار پیمایش است، به‌علاوه دارای خواص متد و رویدادهای ویژه‌ای نیز می‌باشد که بررسی می‌شوند.

خاصیت BorderStyle

به وسیله این خاصیت می‌توان یک کادر در اطراف کنترل نمایش داد.

نحوه استفاده از این خاصیت به این صورت است:

نام کنترل Slider.BorderStyle [= value]

اگر value مقدار ۰ یا ccNone باشد، کنترل بدون کادر و اگر مقدار آن ۱ یا ccFixed Single باشد کنترل به همراه یک کادر، نمایش داده می‌شود. اگر این بخش حذف شود مقدار خاصیت خوانده می‌شود. به عنوان مثال این دستور سبب می‌شود کنترل Slider1 همراه با یک کادر نمایش داده شود.

```
Slider1.BorderStyle =ccFixedSingle
```

خاصیت LargeChange

این خاصیت میزان تغییرات را هنگامی که کاربر در مکانی روی کنترل، عمل کلیک انجام می‌دهد یا کلیدهای Page Up یا Page Down را می‌فشارد، تعیین می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل Slider.LargeChange [= number]

number یک عبارت عددی از نوع صحیح بلند (Long) است که میزان تغییرات را در کنترل تعیین می‌کند. اگر این بخش حذف شود، مقدار خاصیت خوانده می‌شود.

خاصیت SmallChange

این خاصیت میزان تغییرات را هنگامی که کاربر دکمه تنظیم مقدار روی کنترل را می‌کشد یا کلیدهای جهت‌دار چپ یا راست را می‌فشارد، تعیین می‌کند. نحوه استفاده از این خاصیت به این صورت است:

Slider نام کنترل.SmallChange [= number]

number یک عبارت عددی از نوع صحیح است که میزان تغییرات را معین می‌کند. اگر این بخش حذف شود، مقدار خاصیت خوانده می‌شود.

خاصیت Max

این خاصیت بیانگر حداکثر مقداری است که در کنترل قابل استفاده است و با رسیدن دکمه متحرک کنترل به انتهای کنترل به‌دست می‌آید، نحوه استفاده از این خاصیت به این صورت است:

Slider نام کنترل.Max [= value]

value یک عبارت عددی از نوع صحیح است که می‌تواند بین ۳۲۷۶۷- تا ۳۲۷۶۷ باشد. اگر این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت Min

این خاصیت بیانگر حداقل مقداری است که در کنترل قابل استفاده است و با رسیدن دکمه متحرک کنترل به ابتدای کنترل به‌دست می‌آید. نحوه استفاده از این خاصیت به‌صورت زیر است:

Slider نام کنترل.Min [= value]

value یک عبارت عددی از نوع صحیح است که می‌تواند بین ۳۲۷۶۷- تا ۳۲۷۶۷ باشد. اگر این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت TickFrequency

این خاصیت فاصله بین درجه‌بندی‌ها را در کنترل معین می‌کند. به عنوان مثال اگر دامنه تغییرات بین صفر تا ۱۰۰ باشد و مقدار این خاصیت روی ۲ تنظیم شده باشد با حرکت از یک درجه‌بندی به درجه‌بندی بعدی، دو واحد به مقدار قبلی اضافه خواهد شد. نحوه استفاده از این خاصیت به این صورت است:

Slider نام کنترل.TickFrequency [= number]

number یک عبارت عددی است. اگر این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت TickStyle

این خاصیت نحوه قرار گرفتن و نمایش درجه‌بندی‌ها را روی کنترل تعیین می‌کند.
نحوه استفاده از این خاصیت به این صورت است:

Slider نام کنترل.TickStyle[= number]

number یک عدد صحیح یا ثابت رشته‌ای است. مقادیر مربوط به این خاصیت در جدول ۳-۵ ارایه شده‌اند. اگر این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

جدول ۳-۵ مقادیر مربوط به خاصیت TickStyle

ثابت رشته‌ای	ثابت عددی	توضیح
sldBottmRight	۰	نمایش درجه‌بندی در پایین کنترل
sldTopLeft	۱	نمایش درجه‌بندی در بالای کنترل
sldBoth	۲	نمایش درجه‌بندی در بالا و پایین کنترل
sldNoTicks	۳	کنترل بدون نمایش درجه‌بندی

به عنوان مثال این دستور کنترل Slider 1 را به شکلی تنظیم می‌کند که درجه‌بندی در هر دو طرف کنترل نمایش داده شود.

```
Slider1.TickStyle = sldBoth
```

خاصیت Value

این خاصیت مقدار فعلی را در کنترل Slider تعیین می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل Slider.Value [= integer]

integer یک عبارت عددی از نوع صحیح است که مقدار حرکت را در کنترل معین می‌کند. اگر این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

متد GetNumTicks

این متد تعداد خطوط درجه‌بندی بین مقدار حداقل و حداکثر را در کنترل باز می‌گرداند، نحوه استفاده از این متد به‌صورت زیر است:

نام کنترل Slider.GetNumTicks

به عنوان مثال این دستور تعداد خطوط درجه‌بندی عدد ۱۱ در کنترل Slider شکل ۱۹-۵ را روی فرم نمایش می‌دهد.

```
Print Slider1.GetNumTicks
```

رویداد Change

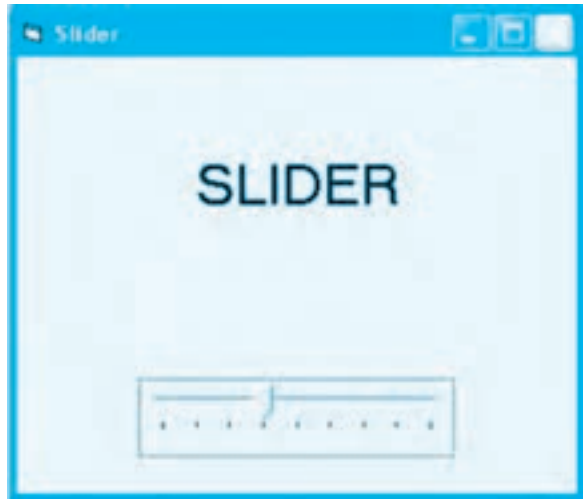
این رویداد وقتی اجرا می‌شود که مقدار خاصیت Value کنترل Slider تغییر کند.

رویداد Scroll

عملکرد این رویداد مشابه رویداد Change است؛ تنها تفاوتی که بین این دو رویداد وجود دارد در این است که این رویداد زمانی اجرا می‌شود که کاربر به وسیله کلیدهای جهت‌دار چپ و راست یا با انجام عمل کشیدن روی دکمه متحرک کنترل مقدار مورد نظر خود را انتخاب کند.

مثال ۳: می‌خواهیم پروژه‌ای طراحی کنیم که بتوان در آن اندازه یک عبارت را به وسیله کنترل Slider تنظیم نمود. برای انجام این مثال این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه با یک فرم و کنترل‌های آن مانند شکل ۲۱-۵ ایجاد کنید.



شکل ۲۱-۵

۲- یک کنترل برچسب با نام lbltext و یک کنترل Slider با نام sldsize مانند شکل ۲۱-۵ روی فرم قرار دهید و تنظیم کنید.

۳- رویداد Load فرم را به این صورت تنظیم کنید:

```
Private Sub Form_Load()
    sldsize.Max = 50
    sldsize.Min = 10
    sldsize.SmallChange = 2
    sldsize.LargeChange = 5
    sldsize.TickFrequency = 5
```

```
    sldsize.Value = 20
    lbltext.FontSize = 20
    lbltext.Alignment = vbCenter
End Sub
```

در این رویداد ابتدا حداکثر و حداقل اندازه قلم مورد استفاده در خاصیت‌های Max و Min کنترل sldsize قرار داده می‌شود تا دامنه تغییرات قلم در کنترل برچسب با استفاده از این مقادیر کنترل شود دستور سوم تعداد و فاصله خطوط درجه‌بندی را با توجه به مقدار حداقل یعنی ۱۰ و مقدار حداکثر یعنی ۵۰ تنظیم می‌کند، سپس خاصیت‌های SmallChange و LargeChange به ترتیب روی مقادیر ۲ و ۵ تنظیم می‌شوند تا میزان تغییرات کوچک و بزرگ در کنترل مشخص شوند.

در مرحله بعد خاصیت Value کنترل sldsize روی مقدار ۲۰ تنظیم می‌شود تا در زمان نمایش فرم دکمه متحرک روی مقدار ۲۰ تنظیم شود. سپس همین مقدار در خاصیت FontSize کنترل برچسب قرار می‌گیرد تا اندازه قلم در متن نمایشی با مقداری که کنترل Slider نمایش می‌دهد، یکسان باشد و در پایان خاصیت Alignment کنترل برچسب روی مقدار vbCenter تنظیم می‌شود تا متن نمایشی در زمان تغییر اندازه قلم همواره در وسط کنترل نمایش داده شود.

۴- اکنون رویداد Change و Scroll کنترل sldSize را به این صورت تنظیم کنید:

```
Private Sub sldsize_Change()
    lbltext.FontSize = sldsize.Value
End Sub

Private Sub sldsize_Scroll()
    lbltext.FontSize = sldsize.Value
End Sub
```

در این رویدادها نیز با استفاده از خاصیت Value مقدار تعیین شده در کنترل اسلایدر در خاصیت FontSize کنترل برچسب قرار می‌گیرد تا با هر روشی که مقدار در کنترل Slider تغییر کند، اندازه قلم تنظیم شود.

۵- پروژه و فرم را با نام Slider ذخیره نمایید، سپس پروژه را اجرا کنید.

۶- روی کنترل Slider با استفاده از کلیدهای Page Up و Page Down، End و Home اندازه متن را تغییر دهید و نتیجه را بررسی کنید.

۷- اکنون با استفاده از عمل کشیدن روی دکمه متحرک کنترل Slider یا کلیک روی نوار درجه‌بندی آن، اندازه متن را تنظیم کرده و نتیجه را بررسی کنید.

۸- به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه‌ای طراحی کنید که به وسیله آن بتوان ابعاد یک تصویر را با استفاده از دو کنترل Slider تنظیم نمود. یک کنترل Slider عرض و دیگری ارتفاع تصویر را تنظیم کند.

۵-۶ کنترل UpDown

این کنترل یکی دیگر از کنترل‌های ActiveX ویژوال‌بیسیک است که به وسیله دکمه‌های مثلثی روی آن می‌توان یک مقدار را افزایش یا کاهش داد. به عنوان مثال می‌توان تنظیم اندازه قلم یا تعیین مدت زمان لازم برای فعال شدن محافظ صفحه نمایش و مانند آن‌ها را نام برد. این کنترل معمولاً به کنترل دیگری مربوط می‌شود تا تغییرات مقادیر در کنترل UpDown روی کنترل دوم اعمال شود، به کنترل دوم، BuddyControl می‌گویند (شکل ۲۲-۵).

شکل ۲۲-۵

برای اضافه کردن این کنترل به جعبه ابزار در کادرمحاوره Components گزینه Microsoft Windows Common Controls-2 5.0 (SP2)را انتخاب کنید تا کنترل UpDown به جعبه ابزار ویژوال‌بیسیک اضافه شود (شکل ۲۳-۵).

شکل ۲۳-۵

این کنترل علاوه بر خاصیت‌های مشترکی که با سایر کنترل‌ها دارد، دارای خاصیت‌ها و رویدادهای منحصر به‌فرد دیگری نیز می‌باشد که در این‌جا بررسی می‌شوند.

خاصیت BuddyControl

به وسیله این خاصیت کنترل UpDown به کنترلی که نام آن به این خاصیت نسبت داده شده است، متصل می‌شود. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.BuddyControl [= **value**]

value نام کنترلی است که کنترل UpDown به آن متصل می‌شود.

خاصیت Alignment

این خاصیت نحوه قرار گرفتن کنترل UpDown را نسبت به کنترل Buddy تعیین می‌کند. در صورتی‌که بخواهید این کنترل در سمت چپ کنترل Buddy قرار بگیرد، مقدار این خاصیت را روی مقدار ۱ یا cc2AlignmentRight و در صورتی که بخواهید این کنترل در سمت راست کنترل Buddy قرار بگیرد، مقدار این خاصیت را روی مقدار صفر یا cc2AlignmentRight تنظیم کنید. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.Alignment [= **value**]

value می‌تواند یکی از مقادیر cc2AlignmentRight یا cc2AlignmentRight باشد. اگر از این بخش استفاده نشود، مقدار خاصیت خوانده می‌شود. به عنوان مثال این دستور کنترل UpDown را در سمت چپ کنترل Buddy قرار می‌دهد.

```
UpDown1.BuddyControl = Text1
UpDown1.Alignment = cc2AlignmentLeft
```

خاصیت BuddyProperty

این خاصیت، تعیین می‌کند که چه خاصیتی از کنترل Buddy با کنترل UpDown، منطبق و همگام شود. این خاصیت باید در زمان طراحی و پس از تنظیم خاصیت BuddyControl تنظیم شود. در صورتی‌که این خاصیت تنظیم نشود، خاصیت پیش‌فرض در کنترل Buddy مورد استفاده قرار می‌گیرد.

خاصیت Value

این خاصیت مقدار کنونی را در کنترل UpDown تعیین می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.Value [= **integer**]

integer یک عبارت عددی از نوع صحیح است که مقدار تعیین شده توسط کنترل را معین می‌کند. اگر این بخش حذف شود مقدار خاصیت خوانده می‌شود.

خاصیت Increment

این خاصیت میزان افزایش مقدار خاصیت Value را در کنترل UpDown تعیین می‌کند. از این خاصیت زمانی استفاده می‌شود که کاربر روی دکمه‌های کنترل UpDown کلیک کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.Increment [= value]

value یک مقدار عددی از نوع صحیح است که میزان افزایش مقدار خاصیت Value را در کنترل معین می‌کند. اگر از این بخش استفاده نشود، مقدار خاصیت خوانده می‌شود.

خاصیت Max

این خاصیت بیانگر حداکثر مقداری است که کنترل UpDown می‌تواند از آن استفاده کند. نحوه استفاده از این خاصیت به صورت زیر است:

نام کنترل UpDown.Max [= value]

value یک مقدار عددی از نوع صحیح است که حداکثر مقدار در کنترل UpDown را تعیین می‌کند. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت Min

این خاصیت حداقل مقداری را که کنترل UpDown می‌تواند کسب کند، تعیین می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.Min [= value]

value یک مقدار عددی از نوع صحیح است که حداقل مقدار در کنترل UpDown را تعیین می‌کند. اگر از این بخش استفاده نشود، مقدار خاصیت خوانده می‌شود.

خاصیت Orientation

این خاصیت نحوه قرار گرفتن دکمه‌های کنترل را تعیین می‌کند. اگر مقدار این خاصیت روی مقدار صفر یا cc2OrientationVertical تنظیم شود، دکمه‌ها به‌صورت عمودی و اگر روی مقدار یک یا cc2OrientationHorizontal تنظیم شود، دکمه‌ها به‌صورت افقی قرار می‌گیرند. نحوه استفاده از این کنترل به این صورت است:

نام کنترل UpDown.Orientation [= value]

value می‌تواند یکی از مقادیر cc2OrientationVertical یا cc2OrientationHorizontal باشد.

خاصیت SyncBuddy

به وسیله این خاصیت می‌توان خاصیت Value کنترل UpDown را با خاصیتی از کنترل Buddy که در خاصیت BuddyProperty انتخاب شده است، منطبق و هماهنگ کرد. این خاصیت از نوع منطقی است و

در صورت تنظیم آن روی مقدار True با افزایش یا کاهش مقدار خاصیت Value، خاصیت انتخاب شده در کنترل Buddy نیز تغییر می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.SyncBuddy [= value]

value یک مقدار منطقی است که می‌تواند True یا False باشد. در صورت انتخاب مقدار True خاصیت انتخاب شده از کنترل Buddy با مقدار خاصیت Value در کنترل UpDown هماهنگ می‌شود. اگر از این خاصیت استفاده نشود مقدار خاصیت خوانده می‌شود.

نکته: این خاصیت پس از تنظیم خاصیت BuddyProperty به‌طور خودکار روی مقدار True تنظیم می‌شود.

خاصیت Wrap

این خاصیت از نوع منطقی است. اگر مقدار این خاصیت روی True تنظیم شود، پس از رسیدن مقدار خاصیت Value در کنترل UpDown به مقدار حداکثر (Max) و عبور از آن، مقدار خاصیت Value روی مقدار تعیین شده در خاصیت Min تنظیم می‌شود. همین‌طور با رسیدن به مقدار حداقل (Min) و عبور از این محدوده، مقدار خاصیت Value روی مقدار تعیین شده در خاصیت Max تنظیم می‌شود. به عبارت دیگر امکان حرکت از محدوده حداکثر به حداقل و برعکس به‌وجود می‌آید.

درصورتی‌که این خاصیت روی مقدار False تنظیم شود، با رسیدن به مقدار Max و Min افزایش یا کاهش مقدار خاصیت Value متوقف می‌شود. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل UpDown.Wrap [= Boolean]

Boolean یک مقدار منطقی است که می‌تواند True یا False باشد. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

رویداد Change

این رویداد وقتی اجرا می‌شود که مقدار خاصیت Value کنترل UpDown تغییر کند.

رویداد DownClick

این رویداد وقتی اجرا می‌شود که کاربر روی دکمه سمت چپ یا دکمه پایینی کنترل UpDown کلیک کند.

رویداد UpClick

این رویداد وقتی اجرا می‌شود که کاربر روی دکمه سمت راست یا دکمه بالایی کنترل UpDown کلیک کند.

مثال ۴: می‌خواهیم پروژه‌ای طراحی کنیم که بتوان در آن اندازه یک عبارت را به وسیله کنترل

UpDown تنظیم نمود؛ برای انجام این مثال این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه با یک فرم و کنترل‌های آن مانند شکل ۲۴-۵ تنظیم کنید.

شکل ۲۴-۵

۲- یک کنترل برچسب با نام lb۱text یک کنترل کادر متن با نام txtsize و یک کنترل UpDown با نام updsize مانند شکل ۲۴-۵ روی فرم قرار دهید و تنظیم کنید.

۳- در پنجره طراحی فرم، کنترل updsize را انتخاب کنید سپس خاصیت BuddyControl آن را روی نام کنترل کادر متن یعنی txtsize تنظیم کنید تا بین دو کنترل ارتباط و وابستگی به وجود آید.

۴- در این مرحله روی دکمه ▼ که روبه‌روی خاصیت BuddyProperty قرار دارد کلیک کرده و از کادر لیستی که نمایش داده می‌شود، خاصیت Text را انتخاب کنید. اکنون خاصیت SyncBuddy به‌طور خودکار روی مقدار True تنظیم می‌شود.

با تنظیم این دو خاصیت دو کنترل با هم ارتباط کامل پیدا می‌کنند، به شکلی که هرگونه تغییر در مقدار خاصیت Value کنترل updsize در کنترل کادرمتن txtsize قابل مشاهده می‌باشد.

۵- اکنون رویداد Load فرم را به این صورت تنظیم کنید تا خاصیت کنترل‌ها در زمان اجرا در هنگام بارگذاری فرم در حافظه تنظیم شوند.

```
Private Sub Form_Load()
   updsize.Alignment = cc2AlignmentRight
   updsize.Increment = 5
```

```
    updsize.Wrap = True
    updsize.Max = 60
    updsize.Min = 10
    updsize.Value = 20
    lbltext.FontSize = 20
    lbltext.Alignment = vbCenter
    txtsize.Locked = True
End Sub
```

در این رویداد ابتدا خاصیت Alignment کنترل UpDown طوری تنظیم می‌شود تا این کنترل در سمت راست کنترل کادر متن قرار بگیرد، سپس خاصیت Increment روی مقدار ۵ تنظیم می‌شود تا در زمان کلیک روی دکمه‌های مثلثی کنترل UpDown هر بار ۵ واحد به اندازه قلم بعدی اضافه یا کم شود. دستور بعد خاصیت Wrap را روی مقدار True تنظیم می‌کند تا با رسیدن خاصیت Value به مقدار حداکثر (Max)، مقدار خاصیت Value روی مقدار حداقل (Min) تنظیم شود و برعکس. در مرحله بعد مقادیر حداقل و حداکثر تنظیم می‌شوند و سپس مقدار پیش‌گزیده در کنترل UpDown در خاصیت Value روی مقدار۲۰ قرار داده شده و برای هماهنگی بین اندازه قلم در کنترل برچسب و کنترل UpDown مقدار خاصیت FontSize برچسب نیز روی ۲۰ تنظیم می‌شود و در پایان نیز خاصیت Alignment کنترل برچسب و خاصیت Locked در کنترل کادر متن روی مقادیر vbCenter و True تنظیم می‌شوند تا متن نمایشی همواره در وسط کنترل برچسب نمایش داده شود و در کادر متن فقط امکان مشاهده اندازه قلم وجود داشته باشد.

۶- در این مرحله رویداد Change کنترل UpDown را به این صورت تنظیم کنید تا با تغییر مقدار خاصیت Value در کنترل UpDown اندازه قلم در کنترل برچسب تنظیم شود.

```
Private Sub updsize_Change()
    lbltext.FontSize = updsize.Value
End Sub
```

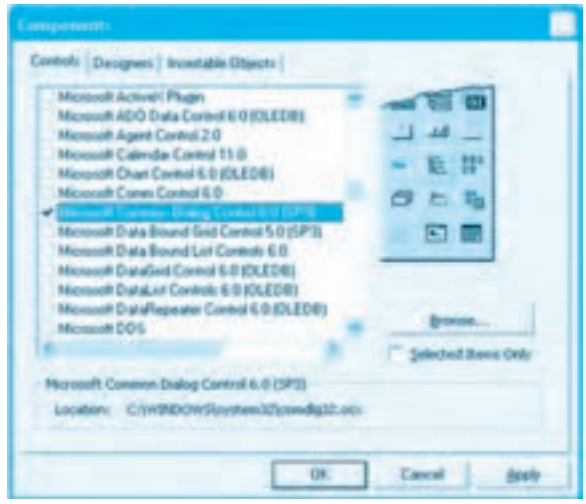
۷- فرم و پروژه را با نام updown ذخیره کرده و سپس آن را اجرا کنید و بعد روی دکمه‌های مثلثی کنترل UpDown کلیک کنید تا اندازه قلم در کنترل برچسب کم و زیاد شود.

۸- پس از آزمایش برنامه، به اجرای آن پایان دهید و به پنجره ویژوال‌بیسیک بازگردید.

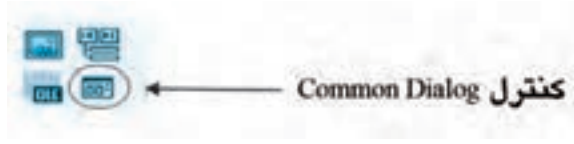
تمرین: پروژه updown را به شکلی تنظیم کنید که به جای استفاده از رویداد Change از رویدادهای UpClick و DownClick کنترل UpDown برای تغییر اندازه قلم در کنترل برچسب استفاده شود.

۵-۷ کنترل CommonDialog

به وسیله این کنترل می‌توانید انواع کادرمحاوره را مانند کادرهای محاوره در ویندوز ایجاد کنید. این کنترل امکان ایجاد کادرهای محاوره برای باز کردن و ذخیره‌سازی فایل‌ها، انجام عملیات چاپ، انتخاب رنگ‌ها و فونت‌ها و نمایش راهنما را فراهم می‌آورد. آیکن این کنترل به‌طور پیش‌فرض در جعبه ابزار دیده نمی‌شود. برای اضافه کردن این کنترل به جعبه ابزار ویژوال‌بیسیک در کادرمحاوره Components گزینه Microsoft Common Dialog Control 6.0 (SP3) را انتخاب کنید (شکل ۵-۲۵) تا کنترل CommonDialog به جعبه ابزار ویژوال‌بیسیک اضافه شود (شکل ۵-۲۶).



شکل ۵-۲۵



شکل ۵-۲۶

پس از اضافه شدن آیکن کنترل، می‌توانید آن را مانند سایر کنترل‌ها به فرم‌های خود اضافه کنید. این کنترل هنگام اجرای برنامه، روی فرم مشاهده نمی‌شود و فقط زمانی کادرهای محاوره نمایش داده می‌شوند که متدهای مربوط به کادرهای محاوره فراخوانی شوند. این کنترل به وسیله متدهای زیر می‌تواند کادرهای مربوطه را نمایش دهد. این متدها را در جدول ۵-۴ مشاهده می‌کنید.

جدول ۴-۵ متدهای کنترل CommonDialog

نام کادرمحاوره	نام متد
کادرمحاوره باز کردن فایل‌ها	ShowOpen
کادرمحاوره ذخیره‌سازی فایل‌ها	ShowSave
کادرمحاوره رنگ‌ها	ShowColor
کادرمحاوره فونت	ShowFont
کادرمحاوره چاپگر	ShowPrinter
کادرمحاوره راهنما	ShowHelp

۵-۷-۱ نحوه ایجاد کادرهای محاوره باز کردن و ذخیره‌سازی فایل‌ها

برای ایجاد این‌گونه از کادرهای محاوره ابتدا یک کنترل CommonDialog روی فرم اضافه کنید و سپس به وسیله متدهای ShowOpen و ShowSave کادرهای محاوره مربوطه را نمایش دهید. کنترل CommonDialog در این دو حالت خاصیت‌هایی را ارایه می‌کند که در این‌جا بررسی می‌شوند:

خاصیت FileName

این خاصیت نام و مسیر فایلی را که توسط کاربر انتخاب شده است، نگهداری می‌کند و به وسیله آن نام فایل انتخاب شده در کادرمحاوره به‌دست می‌آید. به عنوان مثال این دستور نام و مسیر فایل انتخاب شده در کادرمحاوره Open را نشان می‌دهد.

```
Print CommonDialog1.FileName
```

خاصیت Filter

این خاصیت می‌تواند نوع فایل‌هایی را که باید در کادرمحاوره نمایش داده شوند، تعیین کند. به عنوان مثال اگر بخواهیم فایل‌هایی با پسوند txt و doc نمایش داده شوند، خاصیت Filter به این صورت تنظیم می‌شود:

```
CommonDialog۱.Filter = "Text Files (*.txt)|*.txt|Documents (*.doc)|*.doc"
```

نکته: اگر در کادرهای محاوره Open یا Save کاربر روی دکمه فرمان Open یا Save کلیک کند، در صورتی‌که فایلی را انتخاب کرده باشد نام و مسیر آن فایل را می‌توان به وسیله خاصیت FileName مورد دسترسی قرار داد اما اگر کاربر روی Cancel کلیک کند، خاصیت FileName یک رشته خالی را ارایه می‌کند.

خاصیت FilterIndex

در صورتی‌که بیش از یک فایل را به وسیله خاصیت Filter برای نمایش در کادرمحاوره تعیین کرده باشید، به‌طور پیش‌فرض اولین نوع فایل در خاصیت Filter در کادرمحاوره در نظر گرفته می‌شود. به وسیله خاصیت FilterIndex می‌توانید گزینه پیش‌فرض را تغییر دهید.

شکل کلی نحوه استفاده از این خاصیت به این صورت است:

نام کنترل CommonDialog.FilterIndex [= number]

number مقدار عددی است که به شماره ترتیب نوع فایل‌های مربوطه در خاصیت Filter اشاره می‌کند. اگر این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

به عنوان مثال این دستور دومین نوع فایلی را که در خاصیت Filter کنترل CommonDialog مشخص شده است برای نمایش فایل‌ها در نظر می‌گیرد.

```
CommonDialog1.FilterIndex = 2
```

خاصیت FileTitle

به وسیله این خاصیت می‌توان به نام فایل انتخاب شده دسترسی پیدا کرد. این خاصیت فقط نام فایل را بدون مسیر آن مشخص می‌کند. به عنوان مثال این دستور نام فایل انتخاب شده در کادرمحاوره Open را نمایش می‌دهد.

```
Print CommonDialog1.FileTitle
```

خاصیت InitDir

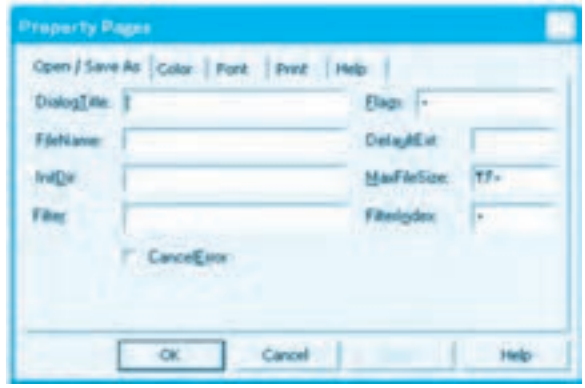
زمانی‌که کادرمحاوره Open یا Save As نمایش داده می‌شود، اسامی فایل‌ها و پوشه‌های مسیر جاری در دیسک نمایش داده می‌شود. در صورتی‌که بخواهید مسیر ویژه‌ای را برای کادرمحاوره در نظر بگیرید، می‌توانید مسیر موردنظر را در این خاصیت ذخیره کنید. شکل کلی نحوه استفاده از این خاصیت به‌صورت زیر است:

نام کنترل CommonDialog.InitDir [= string]

string یک عبارت رشته‌ای است که مسیر موردنظر را تعیین می‌کند. در صورتی‌که از این بخش استفاده نشود، مقدار خاصیت خوانده می‌شود به عنوان مثال این دستور مسیر پیش‌گزیده را برای کادرمحاوره روی ریشه درایو D: تنظیم می‌کند.

نکته: خواص فوق را می‌توانید از طریق کادرمحاوره Property Pages نیز تنظیم کنید. برای دسترسی به کادرمحاوره مزبور، در پنجره خواص، کنترل CommonDialog را انتخاب کرده و روی خاصیت (Custom) آن کلیک کنید، سپس روی دکمه ... کلیک کنید، کادرمحاوره Property Pages نمایش داده خواهد شد. در ادامه روی زبانه Open /Save As کلیک کنید و مقادیر موردنظر را برای هر خاصیت در این پنجره و در بخش مربوط به هر خاصیت بنویسید.

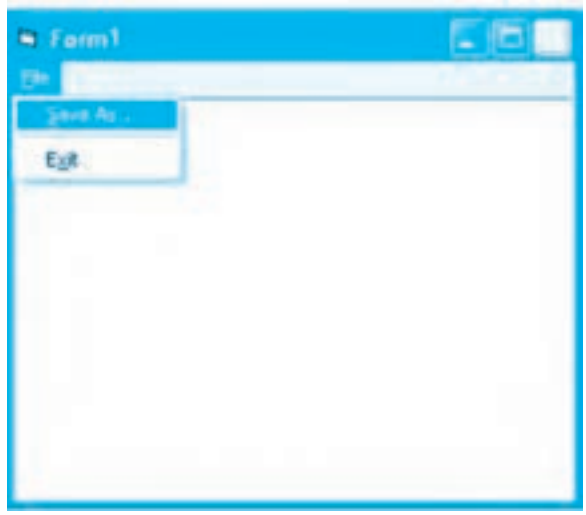
```
CommonDialog1.InitDir = "D:\"
```



شکل ۵-۲۷

مثال ۵: می‌خواهیم پروژه‌ای طراحی کنیم که به وسیله آن بتوان یک فایل متنی جدید ایجاد کرد و محتویات آن را با نام و در مسیر دلخواهی ذخیره نمود. برای انجام این مثال، این عملیات را به ترتیب انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و یک پروژه از نوع Standard EXE همراه با یک فرم، منو و کادر متن مانند شکل ۵-۲۸ ایجاد کنید. به شکلی که کادر متن تمام سطح فرم را بپوشاند. سپس یک کنترل از نوع Common Dialog روی فرم قرار دهید و نام آن را روی Comd Save as تنظیم کنید.



شکل ۵-۲۸

۲- خاصیت MultiLine کنترل کادر متن را روی مقدار True تنظیم کنید.

۳- رویداد Click گزینه mnusaveas گزینه ... Save As را به این صورت تنظیم کنید.

```
Private Sub mnusaveas_Click()
   Dim myfso As New FileSystemObject
   Dim myfile As TextStream
   comdsaveas.Filter = _
                  "All Files (*.*)|*.*|Text Files (*.txt)|*.txt"
   comdsaveas.FilterIndex = 2
   comdsaveas.ShowSave
   If comdsaveas.FileName <> "" Then
      Set myfile = myfso.OpenTextFile _
                        (comdsaveas.FileName, ForWriting, True)
      myfile.Write (txttext.Text)
      myfile.Close
   End If
End Sub
```

در این رویداد ابتدا دو دستور اول، اشیای مورد نیاز برای ذخیره‌سازی فایل متنی را تعریف می‌کنند سپس خاصیت Filter کنترل کادرمحاوره برای نمایش فایل‌های متنی یا همه فایل‌ها در یک مسیر تنظیم می‌شود و بعد با تنظیم خاصیت FilterIndex روی مقدار ۲ نوع فایل متنی به‌طور پیش‌فرض برای نمایش فایل‌ها در کادرمحاوره انتخاب می‌شوند.

در مرحله بعد کادرمحاوره ذخیره‌سازی با متد ShowSave نمایش داده می‌شود. اگر پس از نمایش کادرمحاوره Save نام فایلی برای ذخیره‌سازی تعیین شود و روی دکمه فرمان Save کلیک شود دستورات موجود در دستور شرطی If فایل را باز کرده و محتویات کادر متنی txttext را در آن ذخیره می‌کنند اما اگر روی دکمه فرمان Cancel کلیک شود خاصیت FileName یک رشته خالی خواهد بود در نتیجه بررسی شرط در دستور If نادرست بوده و بنابراین محتویات کادر متن ذخیره نمی‌شود.

۴- پروژه و فرم را با نام texteditor ذخیره کرده و پروژه را اجرا کنید.

۵- در کادر متن، اطلاعاتی را تایپ نموده و در منوی File برنامه روی گزینه ... Save As کلیک کنید و در کادرمحاوره Save یک مسیر دلخواه را انتخاب نموده و پس از تایپ نام فایل روی دکمه Save کلیک کنید و نتیجه را بررسی نمایید و همین عملیات را با دکمه Cancel در کادرمحاوره Save انجام دهید و نتیجه را

بررسی نمایید.

۶- اجرای پروژه را خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

۵-۷-۲ نحوه ایجاد کادرمحاوره قلم (Font)

برای ایجاد این‌گونه از کادرهای محاوره ابتدا یک کنترل CommonDialog روی فرم اضافه کنید و سپس به وسیله متد ShowFont کادرمحاوره مربوط به قلم را ایجاد کنید (شکل ۲۹-۵).

شکل ۲۹-۵

کنترل CommonDialog در این حالت این خاصیت‌ها را ارایه می‌کند:

خاصیت Color

این خاصیت رنگ قلم را تنظیم می‌کند. می‌توانید این خاصیت را از پنجره خواص یا از طریق کدنویسی تنظیم کنید. شکل کلی نحوه استفاده از این خاصیت به این صورت است:

نام کنترل CommonDialog.Color [= number]

number یک عبارت عددی است که رنگ قلم را معین می‌کند. می‌توانید از مقادیر ثابت عددی، رشته‌ای یا توابع مربوط به رنگ‌ها استفاده کنید.

در صورت عدم استفاده از این بخش رنگ فعلی قلم خوانده می‌شود.

خاصیت Flags

به وسیله این خاصیت می‌توانید کادرمحاوره Font را با توجه به نیاز خود نمایش دهید. شکل کلی نحوه استفاده از این خاصیت به این صورت است:

نام کنترل CommonDialog.Flags [= value]

value مقدار ثابتی است که می‌تواند یکی از مقادیر جدول ۵-۵ را کسب کند. اگر از این بخش استفاده نشود مقدار خاصیت خوانده می‌شود.

جدول ۵-۵ مقادیر مربوط به خاصیت Flags

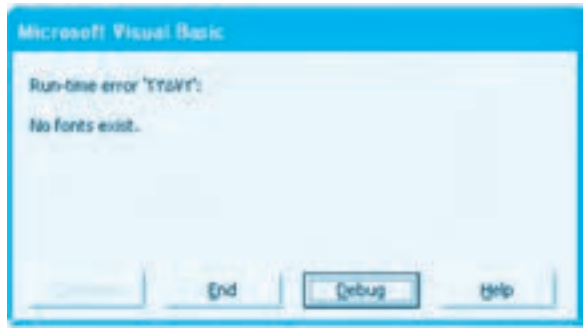
ثابت رشته‌ای	توضیح
cdlCFApply	دکمه فرمان Apply در کادرمحاوره نمایش داده می‌شود.
cdlCFBoth	نام قلم‌های مربوط به چاپگر و صفحه نمایش را در کادرمحاوره نمایش می‌دهد.
cdlCFEffects	امکان انتخاب رنگ، خط زیر و سایر جلوه‌ها را امکان‌پذیر می‌کند.
cdlCFPrinterFonts	فقط نام قلم‌های مربوط به چاپگر را در کادرمحاوره نمایش می‌دهد.
cdlCFScreenFonts	فقط نام قلم‌های مربوط به صفحه نمایش را در کادرمحاوره نمایش می‌دهد.

به عنوان مثال این دستور سبب می‌شود کلیه قلم‌های چاپگر و صفحه نمایش در کادرمحاوره Font نمایش داده شوند.

```
CommonDialog1.Flags = cdlCFBoth
```

نکته: قبل از فراخوانی متد ShowFont مقدار خاصیت Flags را روی یکی از سه مقدار cdlCFBoth، cdlCFPrinterFonts یا cdlCFScreenFonts تنظیم کنید در غیر این صورت پیام‌های خطایی مانند شکل‌های ۵-۳۰ و ۵-۳۱ ظاهر خواهد شد.

شکل ۵-۳۰



شکل ۳۱-۵

خواص مربوط به جلوه‌های ویژه در قلم‌ها

به وسیله چهار خاصیت ارایه شده در جدول ۶-۵ می‌توانید به قلم‌های خود جلوه‌های ویژه‌ای را اضافه کنید.

جدول ۶-۵

نام خاصیت	توضیح
FontBold	قلم پررنگ
FontItalic	قلم به‌صورت مایل
FontStrikethru	قلم با خط وسط در هر کاراکتر
Fontunderline	قلم با خط زیر در هر کاراکتر

شکل کلی نحوه استفاده از این چهار خاصیت به این صورت است:

نام خاصیت.نام کنترل CommonDialog [= **boolean**]

مقدار boolean یک عبارت منطقی True یا False است که فعال یا غیرفعال بودن جلوه مربوطه را معین می‌کند. اگر مقدار هر یک از خواص فوق True باشد جلوه مربوطه فعال و در غیر این صورت غیرفعال خواهد بود. در صورت عدم استفاده از مقدار boolean، مقدار فعلی خاصیت باز خوانده می‌شود. به عنوان مثال این دستورات تمام جلوه‌های ویژه فونت مانند Bold، Italic و Underline را در کادرمحاوره به‌طور پیش‌گزیده انتخاب می‌کند.

```
CommonDialog1.FontBold = True
```

```
CommonDialog1.FontItalic = True
CommonDialog1.FontUnderline = True
```

نکته: برای استفاده از جلوه‌های ویژه قلم لازم است قبل از نمایش کادرمحاوره Font خاصیت Flags آن روی مقدار cdlCFEffects تنظیم شود.

خاصیت FontName

وقتی در کادرمحاوره Font نام یک قلم انتخاب شود، نام قلم انتخاب شده در این خاصیت نگهداری می‌شود. شکل کلی نحوه استفاده از این خاصیت به‌صورت زیر است:

CommonDialog نام کنترل.FontName [= Font]

در صورتی‌که بخواهید قلم موردنظر در کادرمحاوره به عنوان قلم پیش‌فرض انتخاب شود نام قلم را در این خاصیت ذخیره کنید. بخش Font یک عبارت رشته‌ای است که استفاده از آن اختیاری می‌باشد و در صورت عدم استفاده از این بخش، نام قلم انتخاب شده در کادرمحاوره به‌دست می‌آید. به عنوان مثال این دستور قلم Times New Roman را به عنوان قلم پیش‌گزیده در کادرمحاوره Font انتخاب می‌کند.

```
CommonDialog1.FontName = "Times New Roman"
```

خاصیت FontSize

به وسیله این خاصیت می‌توان اندازه قلم را در کادرمحاوره‌ای به‌طور پیش‌فرض انتخاب کرد یا اندازه قلم انتخاب شده توسط کاربر را پس از بسته شدن کادرمحاوره به‌دست آورد. شکلی کلی نحوه استفاده از این خاصیت به صورت زیر است:

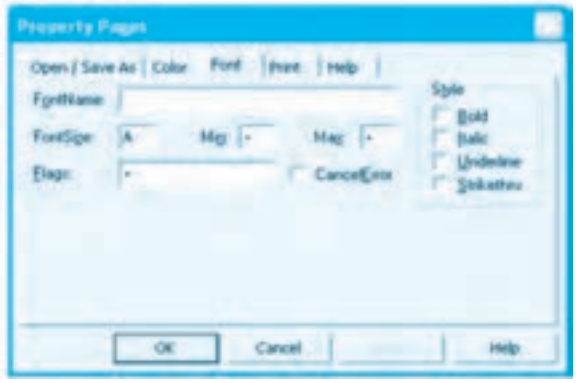
CommonDialog نام کنترل.FontSize [= points]

points یک عبارت عددی است که اندازه پیش‌فرض را برای قلم در کادرمحاوره تعیین می‌کند. در صورت عدم استفاده از این بخش اندازه قلم انتخاب شده به‌دست می‌آید.

نکته: خاصیت‌های کادرمحاوره Font را می‌توانید در زمان طراحی و با استفاده از خاصیت (Custom) و کادرمحاوره Property Pages در زبانه Font تنظیم کنید (شکل ۳۲-۵).

مثال ۶: می‌خواهیم پروژه texteditor را به گونه‌ای تنظیم کنیم تا با استفاده از یک کادرمحاوره Font بتوانیم قلم و ویژگی‌های آن را در کادر متن تنظیم کنیم. برای انجام این مثال این عملیات را به ترتیب

انجام دهید:



شکل ۵-۳۲

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه texteditor را باز کنید.

۲- در این مرحله نوار منو را مانند شکل ۵-۳۳ تنظیم کنید سپس یک کنترل CommonDialog دیگر با نام Comdfont روی فرم قرار دهید.

شکل ۵-۳۳

۳- اکنون رویداد Load فرم را به این صورت تنظیم کنید:

```
Private Sub Form_Load()
```

```
    comdfont.DialogTitle = "Font"
    comdfont.Flags = cdlCFBoth + cdlCFEffects
    comdfont.Color = vbBlack
    comdfont.FontSize = 12
End Sub
```

در این رویداد مقادیر اولیه برای نمایش کادرمحاوره Font مانند رنگ، اندازه و نوع قلم‌ها تنظیم می‌شوند به‌علاوه با استفاده از خاصیت DialogTitle عنوان کادرمحاوره Font نیز به شکل مناسب تنظیم می‌شود.

۴- در این مرحله رویداد Click گزینه ... Font را به این صورت تنظیم کنید:

```
Private Sub mnufont_Click()
    comdfont.ShowFont
    txttext.FontName = comdfont.FontName
    txttext.FontSize = comdfont.FontSize
    txttext.FontBold = comdfont.FontBold
    txttext.ForeColor = comdfont.Color
    txttext.FontItalic = comdfont.FontItalic
    txttext.FontUnderline = comdfont.FontUnderline
    txttext.FontStrikethru = comdfont.FontStrikethru
End Sub
```

در این رویداد ابتدا با استفاده از متد ShowFont کادرمحاوره Font نمایش داده می‌شود سپس با استفاده از خاصیت‌های مربوط به کادرمحاوره Font تنظیمات کاربر به خاصیت‌های مشابه در کنترل کادر متن منتقل می‌شود.

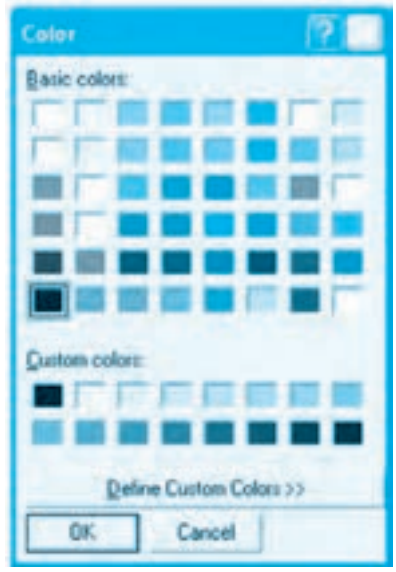
۵- تغییرات را ذخیره کرده و پروژه را اجرا کنید سپس متنی را در کادر متن برنامه تایپ کنید و با استفاده از گزینه Font در منوی Options ویژگی‌های قلم را در متن به دلخواه خود تنظیم کنید این عمل را چند بار تکرار کرده و تفاوت دکمه فرمان OK و Cancel را در کادرمحاوره Font بررسی کنید.

۶- به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید.

تمرین: پروژه texteditor را به شکلی تنظیم کنید تا در زمان نمایش کادرمحاوره Font، تنظیمات آن با ویژگی‌های فعلی متن هماهنگ و یکسان باشد به‌علاوه در تمام پروژه از یک کنترل CommonDialog استفاده شود.

۵-۷-۳ نحوه ایجاد کادرمحاوره رنگ (Color)

برای ایجاد این‌گونه کادرهای محاوره ابتدا یک کنترل CommonDialog روی فرم اضافه کنید و سپس به وسیله متد ShowColor کادرمحاوره مربوط به رنگ را نمایش دهید (شکل ۳۴-۵).



شکل ۳۴-۵

کنترل CommonDialog در این حالت این خاصیت‌ها را ارائه می‌کند.

خاصیت Color

این خاصیت شماره رنگ انتخاب شده به وسیله کاربر را نگهداری می‌کند. شکل کلی نحوه استفاده از این خاصیت به این صورت است:

نام کنترل CommonDialog.Color [=number]

number یک عبارت عددی است که رنگ پیش‌فرض را در کادرمحاوره تعیین می‌کند و می‌تواند یک مقدار ثابت عددی یا یک ثابت رشته‌ای باشد. در صورت عدم استفاده از آن، رنگ انتخابی کاربر در کادرمحاوره به دست می‌آید.

خاصیت Flags

به وسیله این خاصیت می‌توانید کادرمحاوره رنگ را با توجه به نیاز خود تنظیم کنید و نمایش دهید. شکل کلی نحوه استفاده از این خاصیت به این صورت است:

نام کنترلCommonDialog.Flags [=Value]

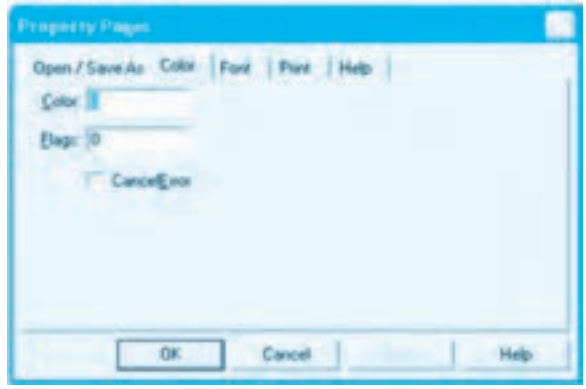
Value مقدار ثابتی است که می‌تواند یکی از مقادیر موجود در جدول ۷-۵ باشد.

جدول ۷-۵ مقادیر مربوط به خاصیت Flags

ثابت رشته‌ای	توضیح
cdlCCFullOpen	کادرمحاوره رنگ به طور همزمان همراه با بخش تعریف رنگ نمایش داده می‌شود. در صورت عدم استفاده از این ثابت، کاربر باید روی دکمه Define Custom Colors >> کلیک کند تا بخش تعریف رنگ فعال شود.
cdlCCPreventFullOPen	امکان استفاده از دکمه Define Custom Colors>> جهت نمایش بخش تعریف رنگ وجود نخواهد داشت.
cdlCCHelpButton	دکمه Help در کادرمحاوره نمایش داده می‌شود.

مثال ۷: می‌خواهیم پروژه texteditor را به شکلی تنظیم کنیم تا به وسیله کادرمحاوره رنگ بتوان رنگ زمینه را در کنترل کادر متن تنظیم کرد. برای انجام این مثال این عملیات را انجام دهید:

نکته: خاصیت‌های کادرمحاوره رنگ را می‌توانید در زمان طراحی و با استفاده از خاصیت (Custom) و کادرمحاوره Property Pages تنظیم کنید (شکل ۳۵-۵).



شکل ۳۵-۵

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه texteditor را باز کنید.

۲- در این مرحله یک گزینه جدید با عنوان ... Color به منوی Options اضافه کنید، سپس یک کنترل CommonDialog جدید با نام comdcolor روی فرم قرار دهید.

۳- رویداد Click گزینه ... Color را به این صورت تنظیم کنید:

```
Private Sub mnucolor_Click()
   comdcolor.Flags = cdlCCFullOpen
   comdcolor.ShowColor
   txttext.BackColor = comdcolor.Color
End Sub
```

در این رویداد ابتدا خاصیت Flags کادرمحاوره به شکلی تنظیم می‌شود تا در زمان نمایش، کادرمحاوره همراه با بخش تعریف رنگ باز شود؛ در مرحله بعد دستور ShowColor اجرا می‌شود تا کادرمحاوره رنگ نمایش داده شود و پس از آن‌که یک رنگ در کادرمحاوره انتخاب شد و روی دکمه فرمان OK کلیک کردیم، خاصیت Color کنترل کادرمحاوره در خاصیت BackColor کنترل کادر متن قرار می‌گیرد تا رنگ زمینه آن مطابق با رنگی که انتخاب شده است، تنظیم شود.

۴- تغییرات را ذخیره کنید، سپس برنامه را اجرا کرده و عملکرد آن‌را بررسی نمایید.

۵- به اجرای پروژه خاتمه دهید و به پنجره ویژوال‌بیسیک بازگردید.

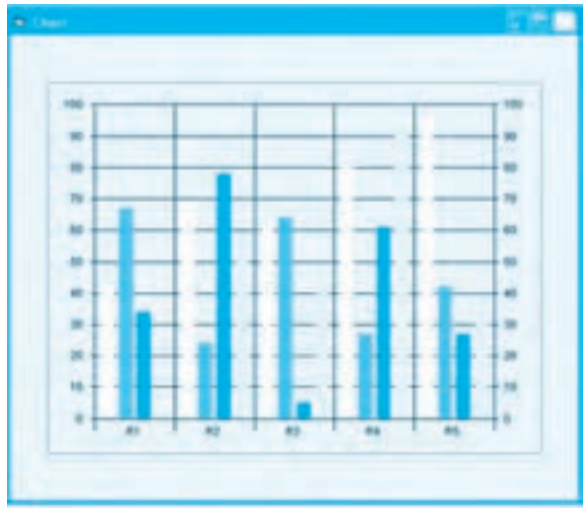
تمرین: پروژه texteditor را طوری تنظیم کنید که از یک کنترل CommonDialog برای تمام کادرهای محاوره استفاده شود.

۵-۷-۴ نحوه ایجاد کادرمحاوره راهنما (Help)

برای ایجاد یک کادرمحاوره راهنما علاوه بر استفاده از یک کنترل CommonDialog و متد ShowHelp لازم است تا مطالب بیشتری در مورد نحوه ایجاد و طراحی فایل‌های راهنما و چگونگی استفاده از آن‌ها را بدانید. ارائه مطالب فوق از بحث این کتاب خارج است.

۵-۸ کنترل نمودار (MSChart)

با استفاده از این کنترل می‌توانید انواع نمودارها را به صورت گرافیکی نمایش دهید (شکل ۳۶-۵).



شکل ۵-۳۶

برای اضافه کردن این کنترل به جعبه ابزار در کادرمحاوره Components گزینه Microsoft Chart Control 6.0 (OLEDB) را انتخاب کنید (شکل ۵-۳۷) تا کنترل MSChart به جعبه ابزار ویژوال‌بیسیک اضافه شود (شکل ۵-۳۸).

شکل ۵-۳۷

شکل ۳۸-۵

این کنترل علاوه بر خاصیت‌های مشترکی که با سایر کنترل‌ها دارد دارای خاصیت‌های ویژه‌ای نیز می‌باشد که در این‌جا بررسی می‌شوند.

خاصیت ChartData

این خاصیت که داده‌های لازم برای ترسیم نمودار در کنترل نمودار را مشخص می‌کند البته داده‌ها را باید به صورت یک آرایه در اختیار این خاصیت قرارداد. نحوه استفاده از این خاصیت به این صورت است.

Ms Chart نام کنترل . Chart Data = data

data یک آرایه دوبعدی است که داده‌های موردنظر برای استفاده در کنترل نمودار را نگهداری می‌کند.

خاصیت ChartType

این خاصیت نوع نمودار را مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

MSChart نام کنترل.ChartType [=type]

type می‌تواند یکی از مقادیر ارائه شده در جدول ۸-۵ باشد. اگر از آن استفاده نشود مقدار خاصیت خوانده می‌شود.

به عنوان مثال این دستور نوع نمودار را در کنترل نمودار به صورت دایره‌ای دوبعدی تنظیم می‌کند.

MSChart1.chartType = VtChChartType2dPie

خاصیت ColumnCount

این خاصیت تعداد ستون‌هایی را که برای نمایش در هر یک از مقادیر داده‌ای که برای نمودار تعیین می‌شود، مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

MSChart نام کنترل.ColumnCount [=Count]

Count یک مقدار عددی صحیح و مثبت است که تعداد ستون‌ها را مشخص می‌کند و اگر از آن استفاده

نشود، مقدار خاصیت خوانده می‌شود.

جدول ۸-۵ مقادیر مربوط به خاصیت ChartType

نوع نمودار	ثابت رشته‌ای
میله‌ای سه بعدی	VtChChartType3dBar
میله‌ای دو بعدی	VtChChartType2dBar
خطی سه بعدی	VtChChartType3dLine
خطی دو بعدی	VtChChartType2dLine
سطحی سه بعدی	VtChChartType3dArea
سطحی دو بعدی	VtChChartType2dArea
پله‌ای سه بعدی	VtChChartType3dStep
پله‌ای دو بعدی	VtChChartType2dStep
ترکیبی سه بعدی	VtChChartType3dCombination
ترکیبی دو بعدی	VtChChartType2dCombination
دایره‌ای دو بعدی	VtChChartType2dPie
نقطه‌ای دو بعدی	VtChChartType2dXY

خاصیت RowCount

این خاصیت تعداد سطرهایی را که برای هر ستون از نمودار در نظر گرفته می‌شود، مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

MSChart نام کنترل.RowCount [=Count]

Count یک مقدار عددی صحیح و مثبت است که تعداد ستون‌ها را مشخص می‌کند. اگر از آن استفاده نشود، مقدار خاصیت خوانده می‌شود.

خاصیت Column

این خاصیت شماره ستون داده را در کنترل نمودار مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

MSChart نام کنترل.Column [=Col]

Col یک مقدار عددی صحیح و مثبت است که شماره ستون را برای داده تعیین می‌کند. اگر از آن استفاده نشود مقدار خاصیت خوانده می‌شود.

خاصیت Row

این خاصیت شماره سطر را برای ستون جاری مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

MSChart نام کنترل.Row [=num]

num یک مقدار عددی صحیح و مثبت است که شماره سطر در ستون جاری را مشخص می‌کند. اگر از آن استفاده نشود، مقدار خاصیت خوانده می‌شود.

خاصیت ShowLegend

این خاصیت از نوع منطقی می‌باشد اگر مقدار آن روی True تنظیم شود راهنمای نمودار نمایش داده می‌شود و در غیر این صورت راهنمای نمودار نمایش داده نمی‌شود.

نحوه استفاده از این خاصیت به این صورت است:

MSChart نام کنترل.ShowLegend [=boolean]

boolean یک مقدار منطقی است و می‌تواند True یا False باشد. اگر از آن استفاده نشود، مقدار خاصیت خوانده می‌شود.

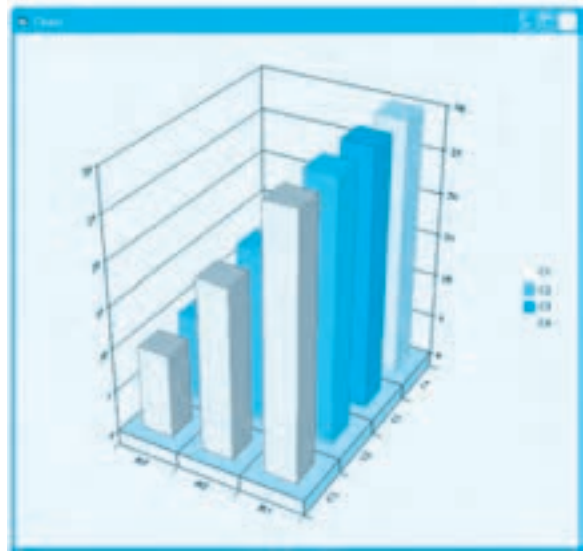
به عنوان مثال این دستورات یک نمودار سه بعدی با ۴ ستون و ۳ سطر همراه با داده‌های آن‌ها را تنظیم می‌کند و نمایش می‌دهند.

```
MSChart۱.chartType = VtChChartType3dBar
MSChart۱.ColumnCount = 4
MSChart۱.RowCount = 3
For Column = 1 To 4
  For Row = 1 To 3
```

```
        MSChart1.Column = Column
        MSChart1.Row = Row
        MSChart١.Data = Row * 10
    Next Row
Next Column
MSChart1.ShowLegend = True
```

در این دستورات ابتدا نـوع نمودار بـا استفاده از خاصیت ChartType بـه صورت میله‌ای سه بعدی تنظیم می‌شود سپس تعداد ستون‌ها و سطرها بـه ترتیب روی مقادیر ۳ و ۴ تنظیم می‌شوند.

در مرحله بعد با استفاده از دو حلقه For برای هر سطر و ستون مشخص، یک داده که از حاصلضرب شماره سطر در عدد ۱۰ به وجود می‌آید در خاصیت Data مربوطه ذخیره می‌شود و در پایان با تنظیم خاصیت ShowLegend روی مقدار True راهنمای نمودار نیز نمایش داده می‌شود. پس از اجرای این دستورات نمودار مانند شکل ۳۹-۵ نمایش داده می‌شود.

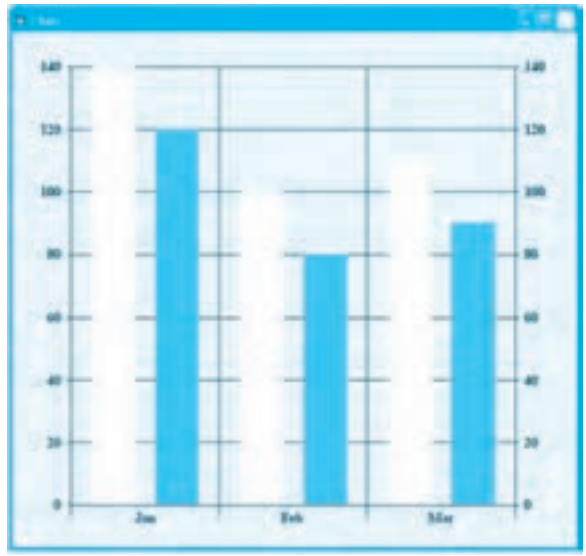


شکل ۳۹-۵

به عنوان مثال دیگر این دستورات میزان بارندگی دو شهر را در ۳ ماه اول سال به صورت یک نمودار میله‌ای دو بعدی نمایش می‌دهند.

```
Dim arrData(۱ To 3, 1 To 3)
arrData(1, 1) = "Jan"
arrData(2, 1) = "Feb"
arrData(3, 1) = "Mar"
arrData(1, 2) = 140
arrData(2, 2) = 100
arrData(3, 2) = 110
arrData(1, 3) = 120
arrData(2, 3) = 80
arrData(3, 3) = 90
MSChart1.ChartData = arrData
MSChart1.chartType = VtChChartType2dBar
```

در این دستورات از یک آرایه دو بعدی با سه سطر و سه ستون به عنوان منبع داده برای کنترل نمودار استفاده شده است پس از مقداردهی اعضای این آرایه، نام آرایه در اختیار خاصیت Data کنترل نمودار قرار می‌گیرد و سپس نوع نمودار از نوع میله‌ای دوبعدی تنظیم می‌شود. پس از اجرای این دستورات نمودار مانند شکل ۴۰-۵ نمایش داده می‌شود.



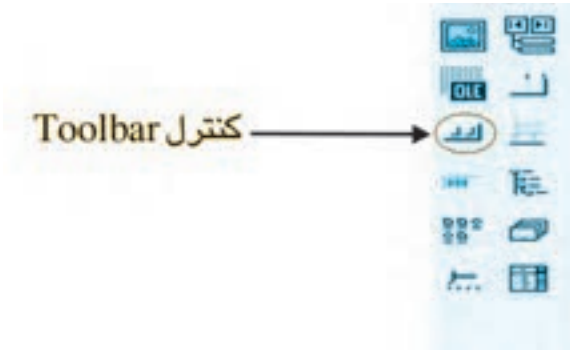
شکل ۴۰-۵

تمرین: پروژه‌ای طراحی کنید که میزان آلودگی هوای دو شهر را در دوازده ماه از سال دریافت کرده و به صورت نمودار میله‌ای سه بعدی نمایش دهد.

۵-۹ کنترل نوار ابزار Toolbar

از این کنترل در اغلب برنامه‌ها برای دسترسی آسان‌تر کاربران به عملکردهای موجود در برنامه استفاده می‌شود به عنوان مثال می‌توان به نوارهای ابزاری که در ویژوال‌بیسیک از آن‌ها استفاده می‌شود، اشاره کرد مانند نوار ابزار استاندارد، نوار ابزار Debug، نوار ابزار Edit و غیره.

برای اضافه کردن این کنترل به جعبه ابزار در کادرمحاوره Components گزینه Microsoft Windows Common Control 6.0 (SP6) را انتخاب کنید تا کنترل نوار ابزار به جعبه ابزار ویژوال‌بیسیک اضافه شود (شکل ۴۱-۵).

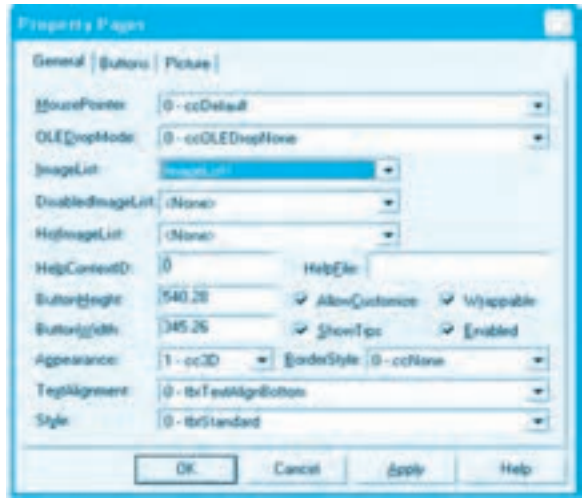


شکل ۴۱-۵

برای ایجاد دکمه‌ها در این کنترل بهتر است در زمان طراحی و در پنجره خواص و روبه‌روی خاصیت Custom روی دکمه کلیک کنید تا کادرمحاوره Property Pages نمایش داده شود؛ در این کادرمحاوره در زبانه General و در لیست بازشو ImageList می‌توانید نام کنترل ImageList موردنظر را که از تصاویر موجود در آن برای دکمه‌های کنترل نوار ابزار استفاده می‌شود، انتخاب کنید (شکل ۴۲-۵)

سپس به زبانه Buttons در این کادرمحاوره بروید و با کلیک روی دکمه فرمان Insert Button دکمه‌های موردنظر را روی کنترل نوار ابزار ایجاد کنید (شکل ۴۳-۵). در این زبانه می‌توانید در کادر متن Caption عنوان دکمه و در کادر متن Image شماره اندیس تصویر موردنظر در کنترل ImageList را انتخاب کنید البته در لیست بازشوی Style نیز می‌توانید نوع دکمه‌ها را مطابق جدول ۹-۵ انتخاب

کنید.



شکل ۴۲-۵

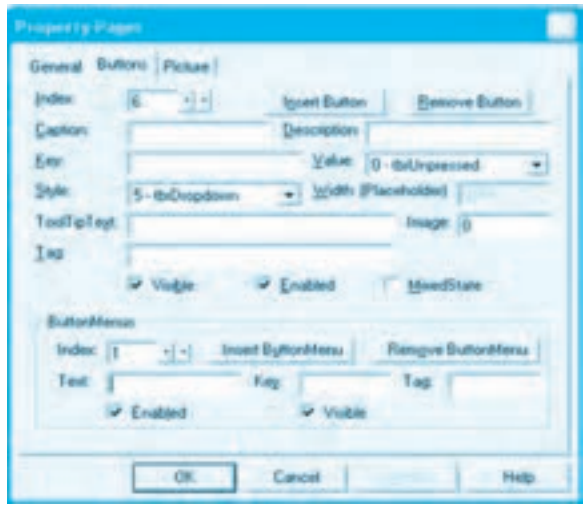
جدول ۹-۵ مقادیر مربوط به خاصیت Style دکمه‌ها در کنترل نوار ابزار

نوع دکمه	ثابت عددی	ثابت رشته‌ای
دکمه معمولی	۰	tbrDefault
می‌توان با کلیک روی دکمه آن را در حالت انتخاب یا عدم انتخاب قرار داد.	۱	tbrCheck
از این نوع می‌توان برای گروهی از دکمه‌ها که در هر لحظه یکی از آن‌ها در حالت انتخاب باشند، استفاده کرد.	۲	tbrButtonGroup
از این نوع برای ایجاد فاصله بین دکمه‌ها استفاده می‌شود.	۳	tbrSeparator
مانند tbrSeparator است با این تفاوت که عرض آن قابل تنظیم می‌باشد.	۴	tbrPlaceholder
از این نوع برای دکمه‌هایی که دارای لیست بازشویی از گزینه‌ها می‌باشند، استفاده می‌شود.	۵	tbrPropDown

نکته: در صورتی که بخواهید در نوار ابزار دکمه‌هایی از نوع tbrDropDown ایجاد کنید، پس از ایجاد دکمه اصلی با استفاده از دکمه فرمان Insert ButtonMenu در بخش ButtonMenus در زبانه Buttons گزینه‌های آن را ایجاد کنید و عنوان گزینه‌ها را در کادر متن Text در همین بخش تایپ کنید (شکل ۴۴-۵).

نکته: به هر دکمه در کنترل در نوار ابزار یک شماره اندیس (Index) اختصاص داده می‌شود که به وسیله آن قابل شناسایی می‌باشد. اندیس‌ها از شماره یک شروع شده و به ترتیب ادامه می‌یابند.

نکته: به هر گزینه در دکمه از نوع tbrDropDown یک شماره اندیس (Index) اختصاص داده می‌شود که به وسیله آن قابل شناسایی می‌باشد. اندیس‌ها از شماره یک شروع شده و به ترتیب ادامه می‌یابند.



شکل ۴۳-۵

این کنترل علاوه بر خاصیت‌های مشترکی که با سایر کنترل‌ها دارد، دارای خاصیت‌ها و رویدادهای ویژه‌ای نیز می‌باشد که در این‌جا بررسی می‌شوند.

خاصیت Align

این خاصیت موقعیت قرار گرفتن کنترل نوار ابزار را روی فرم تعیین می‌کند. نحوه استفاده از این خاصیت به این شکل است:

نام کنترل نوار ابزار.Align [=number]

number یک مقدار عددی صحیح است که می‌تواند یکی از مقادیر جدول ۱۰-۵ باشد. اگر از آن استفاده نشود، مقدار خاصیت خوانده می‌شود.

جدول ۱۰-۵ مقادیر مربوط به خاصیت Align

ثابت رشته‌ای	ثابت‌عددی	توضیح
vbAlignNone	۰	در فرم‌های SDI می‌توان از این مقدار استفاده کرد و موقعیت و اندازه کنترل را به طور دلخواه تنظیم نمود. این مقدار در فرم‌های MDI قابل استفاده نیست.
vbAlignTop	۱	کنترل نوار ابزار در بالای فرم نمایش داده می‌شود.
vbAlignBotton	۲	کنترل نوار ابزار در پایین فرم نمایش داده می‌شود.
vbAlignLeft	۳	کنترل نوار ابزار در سمت چپ فرم نمایش داده می‌شود.
vbAlignRight	۴	کنترل نوار ابزار در سمت راست فرم نمایش داده می‌شود.

خاصیت‌های ButtonHeight و ButtonWidth

وقتی خاصیت Align روی مقدار vbAlignNone تنظیم شود این دو خاصیت به ترتیب می‌توانند ارتفاع و عرض دکمه‌ها را مشخص کنند البته بهتر است این کار را با تنظیم اندازه تصاویر در کنترل ImageList انجام دهید.

خاصیت ShowTips

این خاصیت از نوع منطقی می‌باشد. اگر مقدار آن روی True تنظیم شود هنگام توقف اشاره‌گر ماوس روی هر دکمه یک متن راهنما (ToolTip) نمایش داده می‌شود. در صورت تنظیم این خاصیت روی مقدار True لازم است هنگام ایجاد دکمه متن راهنما در کادرمحاوره Property Pages و در کادر متن ToolTip Text تایپ شود.

اگر مقدار این خاصیت روی False تنظیم شود، متن‌های راهنما نمایش داده نمی‌شوند. این خاصیت را فقط

می‌توان در زمان طراحی برنامه به وسیله پنجره خواص تنظیم نمود.

خاصیت Style

این خاصیت حالت گرافیکی کنترل نوار ابزار و دکمه‌های آن را مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل نوار ابزار.Style [=integer]

integer یک مقدار عددی است. اگر از مقدار عددی صفر یا ثابت رشته‌ای tbrFlat استفاده شود کنترل و دکمه‌های آن به صورت مسطح و اگر از مقدار عددی یک یا ثابت رشته‌ای tbrStandard استفاده شود، کنترل و دکمه‌های آن به صورت سه بعدی نمایش داده می‌شوند.

خاصیت TextAlignment

در صورتی که هنگام ایجاد هر دکمه، عبارتی برای عنوان آن تعیین کرده باشید با این خاصیت می‌توانید محل نمایش آن را در دکمه مشخص کنید. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل نوار ابزار.TextAlignment [=integer]

integer یک مقدار عددی است اگر از مقدار عددی صفر یا ثابت رشته‌ای tbrTextAlignButtom استفاده شود. عنوان در پایین هر دکمه و اگر روی مقدار عددی یک یا ثابت رشته‌ای tbrTextAlignRight استفاده شود، عنوان در سمت راست هر دکمه نمایش داده می‌شود. اگر از این بخش استفاده نشود، مقدار خاصیت خوانده می‌شود.

خاصیت Wrappable

این خاصیت از نوع منطقی می‌باشد و اگر مقدار آن روی True تنظیم شود در صورتی‌که تعداد دکمه‌ها بیش از عرض کنترل باشد دکمه‌ها را در چند سطر نمایش می‌دهد اما اگر روی مقدار False تنظیم شود تمام دکمه‌ها در یک سطر نمایش داده می‌شوند. بهتر است در زمانی که تعداد دکمه‌ها زیادتر از عرض کنترل می‌باشد مقدار این خاصیت روی True تنظیم شود.

رویداد ButtonClick

این رویداد زمانی اجرا می‌شود که عمل کلیک روی دکمه‌های کنترل نوار ابزار انجام شود. این رویداد دارای یک آرگومان با نام Button است که به وسیله آن می‌توان دکمه کلیک شده را شناسایی کرد. در این آرگومان می‌توان با استفاده از خاصیت دیگری به نام Index شماره اندیس دکمه کلیک شده را به دست آورد. به عنوان مثال این رویداد به شکلی تنظیم شده است تا در صورت کلیک روی اولین دکمه در کنترل نوار ابزار یک کادر پیغام نمایش داده شود.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)

  If Button.Index = 1 Then MsgBox "Button1 is  Clicked."
```

```
End Sub
```

رویداد ButtonMenuClick

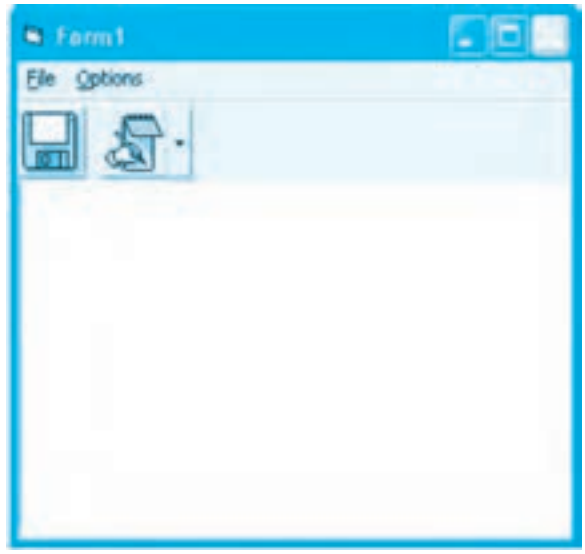
این رویداد زمانی اجرا می‌شود که روی یکی از گزینه‌های دکمه‌ای که از نوع tbrDropDown می‌باشد عمل کلیک انجام شود. این رویداد نیز مانند رویداد ButtonClick دارای یک آرگومان است. آرگومان ButtonMenu در این رویداد می‌تواند گزینه‌ای را که روی آن کلیک شده است، مشخص کند. این آرگومان یک خاصیت با نام Index دارد که شماره اندیس گزینه کلیک شده را تعیین می‌کند؛ به عنوان مثال این رویداد به گونه‌ای تنظیم شده است تا در صورت کلیک روی گزینه دوم در دکمه‌ای از نوع tbrDropDown یک کادر پیغام را نمایش دهد.

```
PrivateSub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As MSComctlLib.ButtonMenu)

  If ButtonMenu.Index = 2 Then MsgBox "Buttonmenu2 is  Clicked."

End Sub
```

مثال ۸: می‌خواهیم پروژه texteditor را به شکلی تنظیم کنیم که عملیات ذخیره فایل و انجام تنظیمات قلم و رنگ زمینه کادر متن به وسیله نوار ابزار نیز امکان‌پذیر باشد برای انجام این مثال این عملیات را انجام دهید:

۱- برنامه ویژوال‌بیسیک را اجرا کرده و پروژه texteditor را باز کنید.

۲- یک کنترل ImageList با نام ilspicture و یک کنترل نوار ابزار با نام tblstandard مانند شکل ۴۴-۵ روی فرم قرار دهید.



شکل ۴۴-۵

۳- با استفاده از کادرمحاوره Property Pages در کنترل ilspicture دو تصویر مانند شکل ۴۴-۵ و با ابعاد ۳۲×۳۲ قرار دهید.

۴- کادرمحاوره Property Pages را در کنترل نوار ابزار tblstandard باز کنید و در زبانه General از لیست بازشو ImageList کنترل ilspicture را برای تأمین تصاویر در کنترل نوار ابزار انتخاب کنید، سپس به زبانه Buttons بروید و مطابق جدول ۱۱-۵ دکمه‌های آن را ایجاد کنید.

جدول ۱۱-۵ خاصیت‌های دکمه‌های نوار ابزار

دکمه / نام خاصیت	دکمه اول	دکمه دوم	دکمه سوم
Index	۱	۲	۳
ToolTipText	Save As …	----------	Options
Image	۱	۰	۲
Style	۰	۳	۵

۵- پس از ایجاد دکمه سوم در زبانه Buttons به بخش ButtonMenus بروید و دو گزینه مطابق جدول ۱۲-۵ برای دکمه سوم ایجاد کنید.

جدول ۱۲-۵ خاصیت‌های گزینه‌های دکمه سوم نوار ابزار

گزینه / نام خاصیت	گزینه اول	گزینه دوم
Index	۱	۲
Text	Font …	Color …

۶- در این مرحله رویداد ButtonClick کنترل نوار ابزار را به این صورت تنظیم کنید:

```
Private Sub tblstandard_ButtonClick(ByVal Button As MSComctlLib.Button)
```

```
    If Button.Index = 1 Then Call mnusaveas_Click
End Sub
```

در این رویداد با استفاده از یک دستور شرطی، اندیس دکمه‌ای که کلیک شده است، بررسی می‌شود. اگر روی دکمه اول کلیک شده باشد، اندیس آن یک است بنابراین رویه رویداد mnusaveas فراخوانی می‌شود تا همان عملیات که با کلیک روی گزینه ... Save As انجام می‌شد، اجرا شود.

۷- در این مرحله رویداد ButtonMenuClick کنترل نوار ابزار را به این صورت تنظیم کنید:

```
Private Sub tblstandard_ButtonMenuClick(ByVal ButtonMenu As MSComctlLib.
ButtonMenu)
    Select Case ButtonMenu.Index
        Case 1
            Call mnufont_Click
        Case 2
            Call mnucolor_Click
    End Select
End Sub
```

در این رویداد با استفاده از یک دستور Select Case و خاصیت Index آرگومان ButtonMenu دکمه کلیک شده بررسی می‌شود.

اگر روی گزینه اول کلیک شود دستور اولین Case اجرا شده و کادرمحاوره Font نمایش داده می‌شود اما اگر روی گزینه دوم کلیک شود دستور دومین Case اجرا شده و کادرمحاوره Color نمایش داده خواهد شد. در این دو حالت رویه‌های رویداد mnufont_Click و mnuColor_Click فراخوانی می‌شوند تا از نوشتن دوباره دستورات جلوگیری به عمل آید.

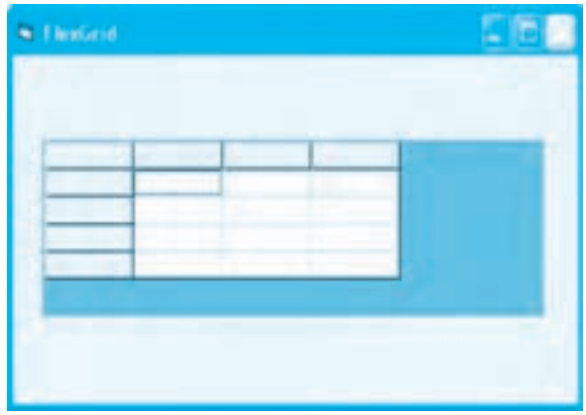
۸- تغییرات را ذخیره کرده و پروژه را اجرا نمایید سپس عملکرد دکمه‌ها را بررسی و نتیجه را با حالتی که از منوها استفاده می‌شود مقایسه کنید.

۹- به اجرای پروژه خاتمه داده و به پنجره ویژوال‌بیسیک بازگردید.

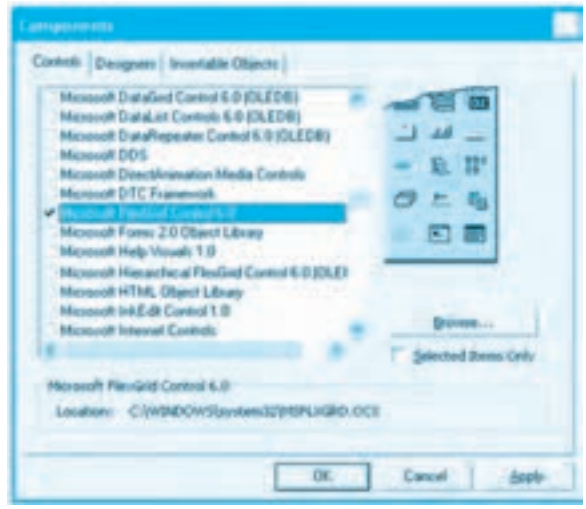
تمرین: پروژه texteditor را به شکلی تنظیم کنید که برای باز کردن فایل‌ها بتوان از یک دکمه در نوار ابزار استفاده کرد؛ به علاوه یک دکمه دیگر نیز برای ایجاد یک سند جدید به نوار ابزار اضافه کنید.

۵-۱۰ کنترل شبکه MSFlexGrid

به وسیله این کنترل می‌توان هر نوع جدول را به شکل مناسب نمایش داد (شکل ۵-۴۵). برای اضافه کردن این کنترل به جعبه ابزار در کادرمحاوره Components گزینه Microsoft Flex Grid Control 6.0 را انتخاب کنید (شکل ۵-۴۶) تا کنترل شبکه به جعبه ابزار ویژوال‌بیسیک اضافه شود (شکل ۵-۴۷).



شکل ۵-۴۵



شکل ۵-۴۶

شکل ۴۷-۵

این کنترل دارای تعدادی ستون و سطر است. به طور پیش‌فرض این کنترل شبکه دارای یک سطر و ستون ثابت می‌باشد که با رنگ تیره‌تر قابل مشاهده هستند. از سطرها و ستون‌های ثابت معمولاً برای نمایش عناوین و از سایر سطرها و ستون‌ها برای نمایش داده‌ها استفاده می‌شود. این کنترل علاوه بر خاصیت‌های مشترکی که با سایر کنترل‌ها دارد دارای خاصیت‌های ویژه‌ای نیز می‌باشد که در این‌جا بررسی می‌شوند.

خاصیت Cols

این خاصیت تعداد کل ستون‌ها را در کنترل شبکه مشخص می‌کند نحوه استفاده از این خاصیت به این صورت است:

نام کنترل شبکه.Cols [=value]

value یک مقدار عددی صحیح از نوع Long می‌باشد و تعداد کل ستون‌ها را مشخص می‌کند و اگر از آن استفاده نشود، تعداد ستون‌ها به دست می‌آید.

خاصیت Rows

این خاصیت تعداد سطرها را در کنترل شبکه مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل شبکه.Rows [=value]

value یک مقدار عددی صحیح از نوع Long می‌باشد و تعداد کل سطرها را مشخص می‌کند و اگر از آن استفاده نشود تعداد سطرها به دست می‌آید.

خاصیت FixedCols

این خاصیت تعداد ستون‌های ثابت را در کنترل شبکه مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل شبکه.FixedCols [=value]

value یک مقدار عددی صحیح از نوع Long می‌باشد و تعداد ستون‌های ثابت را مشخص می‌کند و اگر از

آن استفاده نشود، تعداد ستون‌های ثابت به دست می‌آید.

خاصیت FixedRows

این خاصیت تعداد سطرهای ثابت را در کنترل شبکه مشخص می‌کند. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل شبکه.FixedRows [=value]

value یک مقدار عددی صحیح از نوع Long می‌باشد و تعداد سطرهای ثابت را مشخص می‌کند و اگر از آن استفاده نشود، تعداد سطرهای ثابت به دست می‌آید.

خاصیت Col و Row

این دو خاصیت به ترتیب شماره ستون و شماره سطر خانه جاری را مشخص می‌کنند. نحوه استفاده از این دو خاصیت به این صورت است:

نام کنترل شبکه.Col [=value]

نام کنترل شبکه.Row [=value]

value یک مقدار عددی صحیح از نوع Long می‌باشد و اگر از آن استفاده نشود شماره سطر و ستون خانه جاری به دست می‌آید. با این دو خاصیت می‌توان یک خانه از جدول را در کنترل شبکه مورد دسترسی قرار داد. این دو خاصیت در زمان طراحی قابل تنظیم نمی‌باشند.

خاصیت Text

به وسیله این خاصیت می‌توان محتویات یک خانه در کنترل شبکه را تنظیم کرد یا خواند. برای دسترسی به یک خانه می‌توان از خاصیت‌های Col و Row استفاده کرد. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل شبکه.Text [=string]

string یک عبارت رشته‌ای است که مقدار موردنظر برای ذخیره‌سازی در یک خانه از کنترل شبکه را مشخص می‌کند. اگر از آن استفاده نشود، مقدار ذخیره شده در خانه موردنظر به دست می‌آید.

خاصیت CellAlignment

این خاصیت می‌تواند محتویات یک خانه را در کنترل شبکه از هر دو طرف (عرضی و ارتفاعی) به شکل مناسب نمایش دهد. نحوه استفاده از این خاصیت به این صورت است:

نام کنترل شبکه.CellAlignment [=value]

value می‌تواند یکی از مقادیر ارائه شده در جدول ۱۳-۵ باشد که اگر از آن استفاده نشود مقدار خاصیت خوانده می‌شود.

جدول ۱۳-۵ مقادیر مربوط به خاصیت CellAlignment

ثابت رشته‌ای	ثابت عددی	توضیح
FlexAlignLeftTop	۰	محتویات خانه از سمت چپ و بالای خانه قرار می‌گیرد.
FlexAlignLeftCenter	۱	محتویات خانه از سمت چپ و در وسط خانه قرار می‌گیرد.
FlexAlignLeftBottom	۲	محتویات خانه از سمت چپ و در پایین خانه قرار می‌گیرد.
FlexAlignCenterTop	۳	محتویات خانه در وسط و بالای خانه قرار می‌گیرد.
FlexAlignCenterCenter	۴	محتویات خانه از هر دو طرف در وسط قرار می‌گیرد.
FlexAlignCenterBottom	۵	محتویات خانه در وسط و پایین خانه قرار می‌گیرد.
FlexAlignRightTop	۶	محتویات خانه در سمت راست و بالای خانه قرار می‌گیرد.
FlexAlignRightCenter	۷	محتویات خانه در سمت راست و وسط خانه قرار می‌گیرد.
FlexAlignRightBottom	۸	محتویات خانه در سمت راست و پایین خانه قرار می‌گیرد.
FlexAlignGeneral		اگر محتویات خانه رشته‌ای باشد از سمت چپ و وسط و اگر عددی باشد از سمت راست و وسط در خانه قرار می‌گیرند.

رویداد LeaveCell

این رویداد قبل از آن‌که خانه جاری تغییر کند، اجرا می‌شود.

رویداد EnterCell

این رویداد زمانی اجرا می‌شود که خانه جاری در کنترل تغییر می‌کند. این رویداد بعد از رویداد LeaveCell اجرا می‌شود.

رویداد RowColChange

این رویداد زمانی اجرا می‌شود که خانه جاری در کنترل تغییر می‌کند. این رویداد پس از رویدادهای LeaveCell و EnterCell اجرا می‌شوند.

به عنوان مثال این دستورات میزان بارندگی در سه سال اخیر را در سه شهر در یک کنترل شبکه نمایش می‌دهند.

```
flxgrdcity.Cols = 4
flxgrdcity.Rows = 4
flxgrdcity.FixedCols = 1
flxgrdcity.FixedRows = 1
flxgrdcity.Col = 0
flxgrdcity.Row = 1
flxgrdcity.Text = "Tehran"
flxgrdcity.Col = 0
flxgrdcity.Row = 2
flxgrdcity.Text = "Tabriz"
flxgrdcity.Col = 0
flxgrdcity.Row = 3
flxgrdcity.Text = "Kashan"
flxgrdcity.Col = 1
flxgrdcity.Row = 0
flxgrdcity.CellAlignment = 4
flxgrdcity.Text = 1386
flxgrdcity.Col = 2
flxgrdcity.Row = 0
flxgrdcity.CellAlignment = 4
```

flxgrdcity.Text = 1387

flxgrdcity.Col = 3

flxgrdcity.Row = 0

flxgrdcity.CellAlignment = 4

flxgrdcity.Text = 1388

flxgrdcity.Col = 1

flxgrdcity.Row = 1

flxgrdcity.CellAlignment = 4

flxgrdcity.Text = 50

flxgrdcity.Col = 2

flxgrdcity.Row = 1

flxgrdcity.CellAlignment = 4

flxgrdcity.Text = 70

flxgrdcity.Col = 3

flxgrdcity.Row = 1

flxgrdcity.CellAlignment = 4

flxgrdcity.Text = 75

flxgrdcity.Col = 1

flxgrdcity.Row = 2

flxgrdcity.CellAlignment = 4

flxgrdcity.Text = 40

flxgrdcity.Col = 2

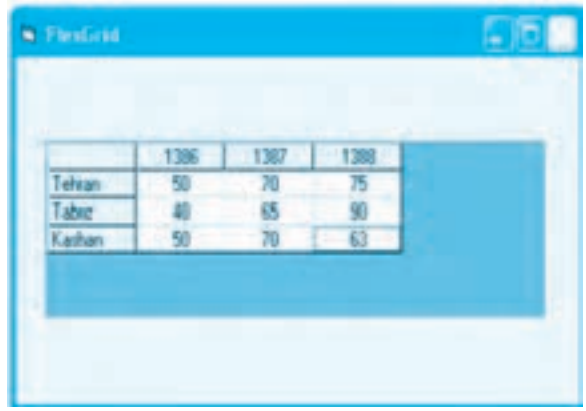
flxgrdcity.Row = 2

flxgrdcity.CellAlignment = 4

```
flxgrdcity.Text = 65
flxgrdcity.Col = 3
flxgrdcity.Row = 2
flxgrdcity.CellAlignment = 4
flxgrdcity.Text = 90
flxgrdcity.Col = 1
flxgrdcity.Row = 3
flxgrdcity.CellAlignment = 4
flxgrdcity.Text = 50
flxgrdcity.Col = 2
flxgrdcity.Row = 3
flxgrdcity.CellAlignment = 4
flxgrdcity.Text = 70
flxgrdcity.Col = 3
flxgrdcity.Row = 3
flxgrdcity.CellAlignment = 4
flxgrdcity.Text = 63
```

در این دستورات ابتدا تعداد سطرها و ستون‌ها با خاصیت‌های Cols و Rows مقداردهی می‌شوند. با توجه به این‌که بارندگی ۳ سال برای ۳ شهر مدنظر است بنابراین لازم است تا ۴ سطر و ستون در کنترل شبکه ایجاد شود و در مرحله بعد برای نمایش سال و نام شهرها یک سطر و یک ستون ثابت نیز در نظر گرفته می‌شوند سپس با استفاده از خاصیت Col، Row و Text نام شهرها در خانه‌های ستون اول (ستون شماره صفر) و سطر دوم (سطر شماره ۱) به بعد قرار می‌گیرند و به همین صورت سال‌ها نیز در خانه‌های سطر اول و ستون‌های اول و دوم و سوم، ذخیره می‌شوند. البته در این سطر و ستون‌ها خاصیت CellAlignment روی مقدار ۴ تنظیم می‌شود تا محتویات خانه‌ها در سطر اول در وسط هر خانه قرار بگیرند در مرحله بعد نیز با همین شیوه، اطلاعات مربوط به شهر تهران در سطر اول و ستون‌های اول تا سوم قرار داده می‌شوند و به همین شکل برای سایر شهرها مقادیر ثبت می‌شود.

پس از اجرای این دستورات کنترل شبکه مانند شکل ۴۸-۵ نمایش داده می‌شود.



شکل ۴۸-۵

تمرین: پروژه‌ای طراحی کنید تا به وسیله آن بتوان اسامی چند شهر دلخواه را همراه با میزان متوسط بارندگی در آن‌ها در ۵ سال اخیر دریافت کرد و در آرایه‌ای ذخیره نمود، سپس محتویات آرایه را به طور مناسب در یک کنترل شبکه نمایش داد. در ضمن در برنامه امکان تهیه نمودار از این داده‌ها نیز وجود داشته باشد.

Learn in English

CommonDialog Control

The CommonDialog control provides a standard set of dialog boxes for operations such as opening and saving files, setting print options, and selecting colors and fonts. The control also has the ability to display help by running the Windows Help engine.

Syntax CommonDialog Remarks

The CommonDialog control provides an interface between Visual Basic and the routines

in the Microsoft Windows dynamic-link library Commdlg.dll. To create a dialog box using this control, Commdlg.dll must be in your Microsoft Windows SYSTEM directory.

You use the CommonDialog control in your application by adding it to a form and setting its properties. The dialog displayed by the control is determined by the methods of the control. At run time, a dialog box is displayed or the help engine is executed, when the appropriate method is invoked; at design time, the CommonDialog control is displayed as an icon on a form. This icon can>t be sized.

The CommonDialog control can display the following dialogs using the specified method.

Method	**Dialog Displayed**
ShowOpen	Show Open Dialog Box
ShowSave	Show Save As Dialog Box
ShowColor	Show Color Dialog Box
ShowFont	Show Font Dialog Box
ShowPrinter	Show Print or Print Options Dialog Box
ShowHelp	Invokes the Windows Help Engine

The CommonDialog control automatically provides context sensitive help on the interface of the dialog boxes by clicking:

- The What>s This help button in the title bar then clicking the item for which you want more information.

- The right mouse button over the item for which you want more information then selecting the What>s This command in the displayed context menu.

واژه‌نامه

Appearance	ظاهر
Arrow	پیکان، فلش
Buddy	رفیق، یار
Common	مشترک
Filter	صاف کردن، جدا نمودن
Flag	پرچم
Frequency	فرکانس، تناوب، تکرار
Increment	افزایش
Legend	راهنما
Orientation	جهت
Slider	لغزنده، سرخورنده
Tick	درجه‌بندی، علامت
Wrap	پوشاندن، پیچیدن

خلاصه مطالب

- برای ایجاد کادرهای محاوره بازکردن فایل‌ها، ذخیره‌سازی فایل‌ها، رنگ‌ها، قلم و راهنما از کنترل CommonDialog استفاده می‌شود.
- برای استفاده از قابلیت‌های نوار پیمایش در برنامه‌ها از کنترل‌های FlatScrollBar استفاده می‌شود.
- برای ایجاد لیستی از تصاویر به منظور استفاده از آن‌ها در کنترل‌هایی مانند ImageCombo و Toolbar از کنترل ImageList استفاده می‌شود.
- برای ایجاد لیستی از تصاویر از کنترل ImageCombo استفاده می‌شود.
- برای نمایش، دریافت، ویرایش و ذخیره‌سازی داده‌های متنی می‌توان از کنترل RichTextBox استفاده

کرد.

- به وسیله کنترل UpDown می‌توان مقدار مربوط به خاصیت‌های کنترل‌های دیگر را تنظیم کرد یا از آن برای تنظیم مقدار یک متغیر استفاده نمود.
- برای نمایش انواع نمودار می‌توان از کنترل MSChart استفاده کرد.
- از کنترل Slider برای تنظیم یک مقدار معین استفاده می‌شود.
- برای نمایش و مدیریت داده‌ها به صورت جدول از کنترل FlexGrid استفاده می‌شود.
- از کنترل Toolbar برای ایجاد نوارهای ابزار استفاده می‌شود.

آزمون نظری

۱- کدام کنترل برای نمایش لیستی از تصاویر مناسب است؟

الف- ImageList ب- ListBox ج- ComboBox د- ImageCombo

۲- کدام خاصیت در کنترل UpDown مقدار تعیین شده توسط آن‌را در اختیار خاصیت کنترل متصل به آن قرار می‌دهد؟

الف- Buddy ب- Text ج- BuddyProperty د- SyncBuddy

۳- با کدام کنترل می‌توان فایل‌های متنی را به طور مستقیم مشاهده یا ذخیره نمود؟

الف- TextBox ب- RichTextBox ج- ListBox د- UpDown

۴- کدام رویداد در کنترل نوار پیمایش برای ایجاد تغییرات با اندازه کم مناسب است؟

الف- Change ج- LostFocus ب- Scroll د- GotFount

۵- کدام خاصیت در کنترل Slider نحوه قرار گرفتن درجه‌بندی‌ها را معین می‌کند؟

الف- TickFrequency ب- TickStyle

ج- Value د- Scroll

۶- کدام‌یک از رویدادهای کنترل شبکه (FlexGrid) در زمان تغییر سلول جاری زودتر از سایر رویداد اجرا می‌شود؟

الف- EnterCell ب- LeaveCell ج- RowColChange د- GotFocus

۷- با کدام نوع دکمه در کنترل نوار ابزار Toolbar می‌توان یک دکمه همراه با چند گزینه ایجاد نمود؟

الف- tbrCheck ب- tbrButtonGroup

ج- tbrPlaceholder د- tbrDropdown

۸- کدام خاصیت در کنترل MSChart می‌تواند سبب نمایش راهنما شود؟

الف- ShowHelp ب- ShowLegend ج- ShowChart د- ChartType

۹- کدام خاصیت در کادرمحاوره قلم قبل از نمایش آن باید تنظیم شود؟

الف- Flags ب- Font ج- FontBold د- Flag

۱۰- کدام کنترل برای ایجاد انواع کادرهای محاوره مناسب است؟

الف- ListBox ب- Common

ج- CommonDialog د- DTPicker

11- The ..... Control Provides Standard Set of dialog boxes such as opening and saving files color , and font and help.

a- Slider **b**- UpDown **c**- Toolbar **d**- CommonDialog

۱۲- نحوه اضافه کردن کنترل‌های ActiveX را به پروژه‌ها بیان کنید.

۱۳- نحوه اضافه کردن، کاربرد و خاصیت‌ها، متدها و رویدادهای این کنترل‌ها را با ذکر مثال توضیح دهید.

الف- ImageList **ب**- ImageCombo **ج**- UpDown **د**- Slider

ه- RichTextBox **و**- Toolbar **ز**- FlatScroll **ح**- MSChart

۱۴- نحوه ایجاد انواع کادرهای محاوره را به طور کامل با ذکر مثال توضیح دهید و تفاوت‌های آن‌ها را بیان کنید.

۱۵- کاربرد و نحوه استفاده از کنترل FlexGrid را بیان کنید و خاصیت‌ها و رویدادهای آن‌را توضیح دهید.

آزمون عملی

۱- پروژه‌ای طراحی کنید که به وسیله آن بتوان یک متن به صورت چشمک‌زن نمایش داد و با استفاده از یک کنترل UpDown بتوان سرعت چشمک‌زدن آن‌را تنظیم کرد.

۲- یک پروژه طراحی کنید که به وسیله آن بتوان تصاویر دلخواه خود را در یک کادر لیست تصویر قرار داد و با انتخاب تصویر در کادر لیست، تصویر انتخاب شده را روی فرم مشاهده کرد. به علاوه به وسیله دو کنترل Slider بتوان ابعاد تصویر را تغییر داد.

۳- پروژه‌ای طراحی کنید که شرایط زیر را دارا باشد:

الف- اطلاعات دانش‌آموزان یک کلاس را که شامل نام، نام خانوادگی و معدل هر یک می‌باشد در یک فایل ذخیره کند.

ب- اطلاعات ذخیره شده در فایل را خوانده و در یک کنترل FlexGrid به شکل مناسب نمایش دهد.

ج- یک نمودار میله‌ای از معدل و نام و نام خانوادگی کلیه دانش‌آموزان به صورت دو بعدی یا سه بعدی نمایش دهد.

د- کلیه امکانات با استفاده از نوار منو، نوار ابزار و به صورت MDI قابل استفاده باشد.

پاسخنامه

واحدکار اول

۱- د)	۲- ب)	۳- ب)	۴- د)
۵- الف)	۶- ج)	۷- ج)	۸- ج)
۹- ج)	۱۰- ج)	۱۱- ج)	۱۲- الف)
۱۳- د)	۱۴- الف)	۱۵- ب)	b -16)

واحدکار دوم

۱- الف)	۲- ج)	۳- الف)	۴- ج)
۵- ب)	۶- ب)	۷- ج)	c - 8)

واحدکار سوم

۱- ب)	۲- ج)	۳- ج)	۴- ب)
۵- ب)	۶- د)	۷- ب)	۸- الف)
۹- ب)	۱۰- ب)	c -11)	

واحدکار چهارم

۱- د)	۲- ج)	۳- ج)	۴- ج)
۵- ج)	۶- ب)	۷- د)	۸- الف)
۹- د)	۱۰- ب)	d -11)	

واحدکار پنجم

۱- د)	۲- د)	۳- ب)	۴- ب)
۵- ب)	۶- ب)	۷- د)	۸- ب)
۹- الف)	۱۰- ج)	b -11)	

منابع

۱- راهنمای MSDN

2- Desktop Applications for Microsoft Visual Basic 6.0 (MSpress)