به نام خدا

برنامه سازی ۲

رشته ی کامپیوتر

گروه تحصیلی کامپیوتر

شاخه آموزش فنی و حرفه ای

(اجرای آزمایشی)

۱۳۸۵

وزارت آموزش و پرورش

سازمان پژوهش برنامه ریزی آموزش

برنامه ریزی محتوا و نظارت بر تألیف : دفتر برنامه ریزی و تألیف آموزش های فنی وحرفه ای و کاردانش

اعضا ی کمیسیون تخصصی برنامه ریزی و تألیف: محمدرضا یمقانی، محمد مشاهری فرد، عسگر قندچی، مرتضی حمیدی، فرزانه شیخی، سید حمیدرضا ضیایی، سیدعلیرضا رضازاده، حمیداحدی

نام کتاب : برنامه سازی ۲

مؤلف : علیرضا جباریه

آماده سازی و نظارت برچاپ :

صفحه آرا : مریم نورزاده

طراح جلد:

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

تلفن : ۴- ۶۶۰۲۶۲۴۱ ، دورنگار: ۶۶۰۲۶۲۴۰ ، صندوق پستی ۱۳۴۴۵/۶۸۴

چاپخانه : اتحاد

سال انتشار ونوبت چاپ : ۱۳۸۵ نوبت اول

شابک ۳-۲۳ ۱۴-۰۵-۹۶۴

ISBN 964-05-7423-3

ج

شما عزیزان کوشش کنید که از این وابستگی بیرون آیید و احتیاجات کشور خودتان را بر آورده سازید، از نیروی انسانی ایمانی خودتان غافل نباشید و ازاتکای به اجانب بپرهیزید.

امام خمینی «قدّس سرّه الشّریف»

فهرست مطالب

زبان برنامه نویسی Visual Basic یکی از زبان های برنامه نویسی متداول و محبوب در بین برنامه نویسان است . در کتاب برنامه سازی ۱ با برخی از جنبه های مقدماتی این زبان آشنا شدید در این کتاب با جنبه های پیشرفته این زبان آشنا می شوید. یادگیری مفاهیم این کتاب شمار را قادر خواهد ساخت تا برنامه های کامل تر و پیچیده تر را بنویسید و امکانات این زبان را به نحو مناسب تری به کار ببرید.

در این کتاب با منوهای استاندارد، آرایه، کاربرد فرم های آماده، گرافیک، روال ها و توابع برخی دیگر از امکانات Visual Basic آشنا شده و با آنها کار خواهید کرد.

توصیه می شود برای تقویت مهارت های خود در زمینه برنامه نویسی که یکی از جنبه های مهم رشته کامپیوتر است علاوه بر انجم پروژه هایی که در این کتاب ارایه شدهاست کتاب ها و منابع مرتبط را مطالعه نموده و تا حد امکان که برنامه های مختلف را مورد تجزیه و تحلیل قرار دهید.

هدف کلی: شناخت مفاهیم پیشرفته و مباحث تکمیلی زبان برنامه VB و نوشتن برنامه های پیشرفته با آن

فصل اول

ایجاد منو

هدف های رفتاری: پس از پایان این فصل انتظار می‌رود که هنرجو بتواند:

- با Application Wizard منوهای ساده‌ای را برای برنامه‌های کاربردی ایجاد کند؛
- با Menu Editor منوهای سفارشی و حرفه‌ای را ایجاد کند؛
- منوهای بازشو (میانبر) را ایجاد کند که با کلیک راست کاربر در هر جایی از فرم ظاهر می‌شود؛

۱-۱- منوهای استاندارد ویندوز

هر برنامه‌ای را که بیش از یک عمل انجام می‌دهد می‌توان با افزودن منوهایی کارآمدتر کرد. هنگامی که برنامه‌ای را طراحی می‌کنید هدف، ایجاد ویژگی‌هایی است که کاربرد آن را ساده‌تر کند. منویی که به طور مؤثر طراحی شده باشد، این هدف را برآورده خواهد کرد.

منو متداول‌ترین ویژگی قابل رؤیت برای برنامه‌ی کاربردی است که استفاده از برنامه را ساده‌تر خواهد کرد. با وجود منو، برنامه برای کاربران، طبیعی و آشنا خواهد بود. منویی که به صورت بدون تأثیر (بد) طراحی شده است، سبب سردرگمی کاربران و عدم درک چگونگی کار برنامه می‌شود.

اغلب برنامه‌ها دارای وظایف فایلی هستند که به کاربران امکان می دهد فایل ها را ایجاد و ذخیره کنند. برنامه‌هایی که از ویژگی‌های متداول ویندوز مثل باز کردن و ذخیره‌ی فایل‌ها یا کپی کردن متن استفاده می‌کنند، باید از استانداردهای منوهای ویندوز پیروی کنند.

هدف مایکروسافت ایجاد ویژگی‌های استاندارد در کل رابط گرافیکی کاربر است. این بدین معنی است که اغلب برنامه‌های تحت ویندوز ظاهری مشابه هم دارند. به عنوان مثال، روش دسترسی به گزینه‌ی Print در کل برنامه‌ها مشابه هم است. این استانداردها در چندین مورد مثل سیستم‌های راهنما رعایت می‌شوند.

نکته:

سعی کنید در برنامه‌ها این استاندارد و سنت مایکروسافت را رعایت کنید مگر اینکه دلیل محکمی برای عدم رعایت آنها داشته باشید. متداول‌ترین ویژگی‌های برنامه‌های کاربردی، منوهای File، Edit و Help است که معمولاً File و Edit در سمت چپ نوار منو و Help در سمت راست آن ظاهر می‌شوند.

عناصر خاص منوهای ویندوز که از استاندارد پیروی می‌کنند، در جدول زیر، فهرست شده‌اند.

جدول ۱-۱) موارد متداول منوهای استاندارد ویندوز

ویژگی	*تعریف*
Caption	استفاده از یک یا دو کلمه‌ی خاص
Organization	گزینه‌های منو که براساس وظیفه گروه‌بندی خواهند شد تا حداقل تعداد سطوح را برای دسترسی به هر ویژگی ارایه کنند.
Access Keys	هر گزینه باید یک کلید دسترسی داشته باشد تا از طریق صفحه کلید نیز بتوان به گزینه‌ها دسترسی داشت. در هر قسمت از منو، کلید باید منحصر به فرد باشد و معمولاً حرف اول Caption است.
Shortcut Keys	هر گزینه‌ای از منو که نیاز است در بخشی از برنامه قابل دسترس باشد، باید یک کلید میانبر داشته باشد. هر کلید میانبر می‌تواند فقط مربوط به یک گزینه باشد.
Check Box	گزینه‌هایی که نیاز به فعال و غیرفعال شدن دارند، باید یک کادر علامت داشته باشند تا نشان دهنده‌ی حالت آن باشد. گزینه‌ای که یک کادر محاوره‌ای را باز می‌کند باید دارای ... در انتهای نام خود باشد.

۱-۲ کاربرد Application Wizard

مزیت عمده‌ی Aplication Wizard ایجاد منوهایی است که دارای ویژگی‌های استاندارد ویندوز هستند. به سادگی، می‌توان این ویژگی‌ها که به صورت الگو ارایه شده‌اند را انتخاب کرد. از

Application Wizard نمی‌توان برای اصلاح پروژه‌های موجود استفاده کرد. برای اصلاح این پروژه‌ها از Menu Edior استفاده کنید که در ادامه شرح خواهیم داد.

Application Wizard ابزار مفیدی برای ایجاد برنامه‌ی کاربردی جدید است. این بدین معنی است که این ابزار برای ایجاد برنامه‌ی کاربردی با ویژگی‌های استاندارد مورد استفاده قرار می‌گیرد. در صورتی که می‌خواهید ویژگی‌های دیگری را به آنها اضافه کنید، باید برنامه‌نویسی کنید یا از Menu Editor استفاده کنید. بعد از کلیک کردن روی Finish در Application Wizard، برنامه‌ی اصلی تولید شده و امکان انجام تغییرات با Menu Editor را فراهم می‌کند.

۱-۲-۱ ایجاد یک منوی ساده با Application Wizard

۱- از طریق کادر محاوره‌ای پیش فرض که هنگام شروع Visual Basic 6.0 باز می‌شود یا با انتخاب گزینه ی New Project از منوی File، برنامه‌ی Application Wizard را شروع کنید (شکل ۱-۱).

شکل ۱-۱) روی آیکن Application Wizard برای شروع ویزارد دابل کلیک کنید.

۲- کادر محاوره‌ای Introduction امکان استفاده از پاسخ‌هایی که در طول جلسه‌ی قبلی ذخیره کرده‌اید را فراهم می‌کند (شکل ۱-۲). گزینه های پیش فرض را تغییر نداده و روی Next کلیک کنید.

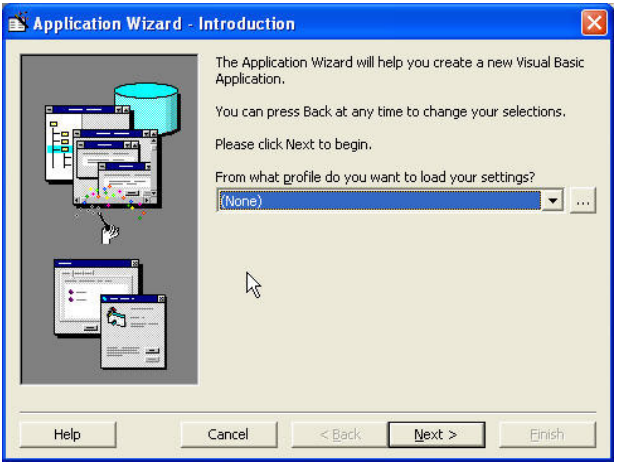
نکته:

با ذخیره‌ی تنظیمات در Application Wizard، می‌توان از وارد کردن دوباره‌ی گزینه‌های ایجاد شده جلوگیری کرد. Application Wizard تنظیمات منو و نوار ابزار را در یک فایل پروفایل (RWP.) ذخیره می‌کند.

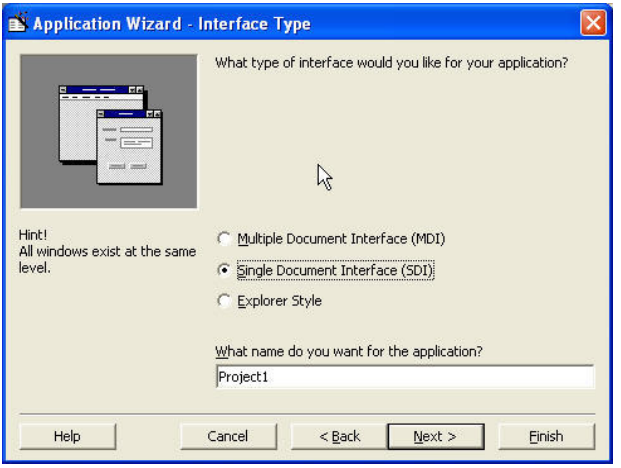
۳- در کادر محاوره‌ای Interface Type، نوع صفحه‌ی آغازین برنامه‌ی کاربردی را انتخاب کنید (شکل ۱-۳). برای این برنامه‌ی کاربردی نمونه، Single Document Interface را انتخاب کنید. نام پروژه پیش فرض را تغییر نداده و Next را انتخاب کنید.

۴- یک منوی پیش فرض براساس استاندارد ویندوز ایجاد می‌شود و می‌توانید آن را تغییر دهید. بعد از انجام تغییرات، روی Next کلیک کنید (شکل ۱-۴).

۵- Application Wizard امکان سفارشی کردن نوار ابزار، فایل منبع، مرورگر، اتصال پایگاه داده و سایر الگوها را فراهم می‌کند. برای این مثال، از این کادرهای محاوره‌ای صرفنظر کرده و پنج بار روی Next کلیک کنید تا به آخرین کادر محاوره‌ای برسید.

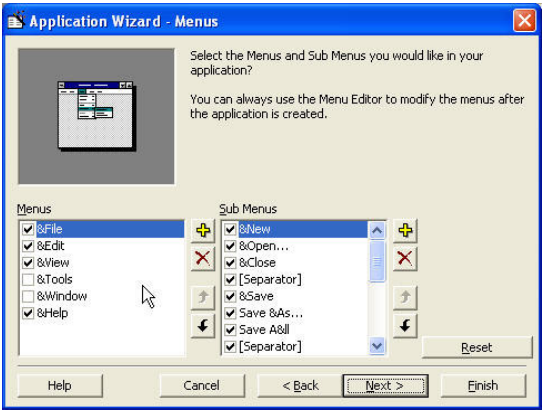


شکل ۱-۲) پروفایل‌ها امکان بارگذاری مجدد گزینه‌ها از جلسات قبلی Application Wizard را فراهم می‌کنند.



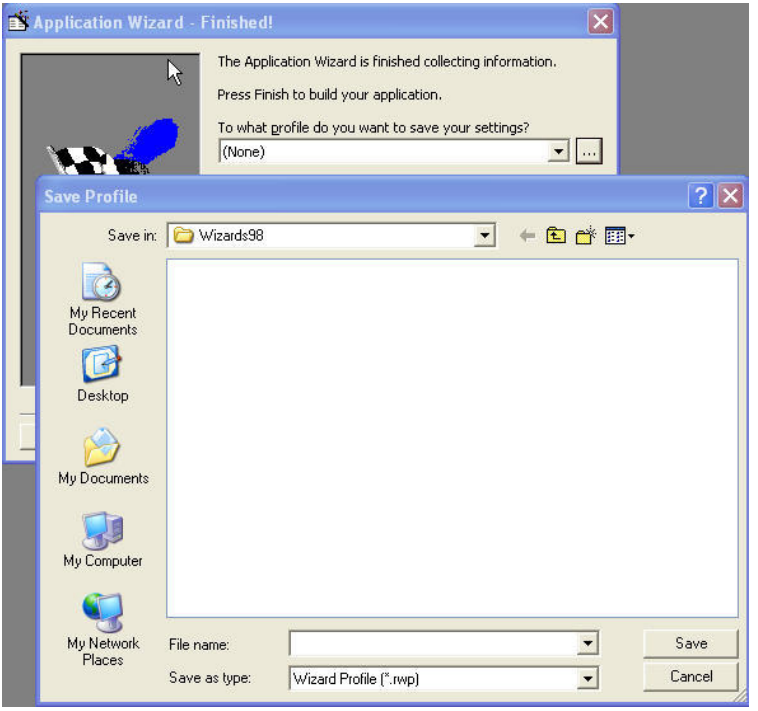
شکل ۱-۳) انتخاب نوع رابط بستگی به چگونگی استفاده از برنامه‌ی کاربردی خواهد داشت.

برنامه‌های کاربردی چندوظیفه‌ای اغلب از MDI استفاده می‌کنند.



شکل ۱-٤) نوع منو و گزینه‌ها را برای برنامه‌ی کاربردی انتخاب کنید.

۶- در آخرین کادر محاوره‌ای، می‌توان پروفایل را ذخیره کرد (شکل ۱-۵). نامی را برای پروفایل وارد کنید که مرتبط با برنامه‌ی کاربردی باشد. بعد از وارد کردن نام، روی Finish کلیک کنید تا Application Wizard کامل شود. (کلیک کردن روی Finish در کادرهای محاوره‌ای قبلی، سبب می‌شود که ویزارد از ذخیره‌ی پروفایل، صرفنظر کند.)



شکل ۱-۵) نام پروفایل را در آخرین صفحه وارد کنید.

۱-۳ کاربرد Menu Editor

Menu Editor امکان ایجاد نوارمنو از ابتدا یا اصلاح منوهای ایجاد شده را فراهم می‌کند. مانند Application Wizard این منوها در بالای فرم ظاهر می‌شوند ولی می‌توان در هر جایی از فرم با کلیک راست نیز به آنها دسترسی پیدا کرد.

مثال ۱-۱) ایجاد یک منوی ساده با Menu Editor

۱- پروژه‌ی جدیدی را باز کنید. پروژه را با نام فایل جدیدی، ذخیره کنید. فرم پیش فرض را تغییر نام داده (با نامی مثل frmMenu) و ذخیره کنید.

نکته:

برای جابه‌جایی بین کادرهای متن در Menu Editor، از کلید Tab یا ماوس استفاده کنید. فشار دادن کلید Enter سبب ایجاد یک منوی جدید می‌شود.

نکته:

قرار دادن علامت امپرسند (&) در Caption منو و قبل از یک نویسه، سبب خواهد شد که آن نویسه به صورت زیر خط‌دار در منو ظاهر شود. با فشار دادن کلید Alt به همراه این نویسه می‌توان این منو را انتخاب کرد. به عنوان مثال، با وجود F زیر خط‌دار در منوی File، کاربران می‌توانند با فشار دادن Alt+F منو را باز کنند.

۲- با کلیک روی دکمه‌ی Menu Editor در نوارابزار استاندارد یا فشار کلیدهای Ctrl+E، برنامه‌ی ویرایشگر منو را باز کنید. اگر فوکوس روی فرم نباشد، آیکن Menu Editor غیرفعال خواهد شد.

۳- در کادر محاوره‌ای Menu Editor و در کادر متن Caption، عبارت File و در کادر متن Name، عبارت mnuFile را وارد کنید.

۴- روی دکمه‌ی فلش راست کلیک کنید. این دکمه برای تورفتگی است.

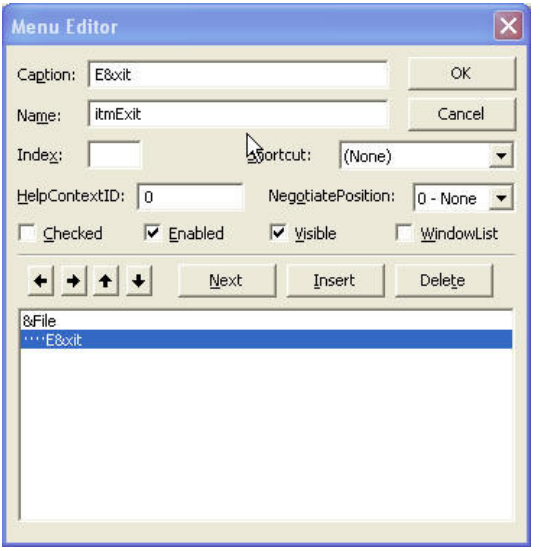
۵- در کادر متن Caption، عبارت E&xit و در کادر متن Name، عبارت itmExit را تایپ کنید. منو مانند کادر محاوره‌ای شکل ۱-۶ ظاهر می‌شود.

۶- روی Ok کلیک کنید.

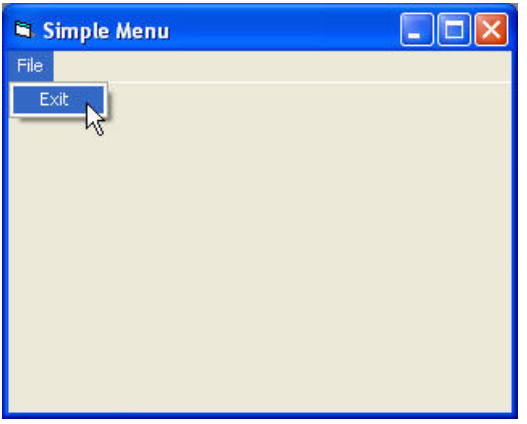
۷- منوی ایجاد شده به فرم اضافه می‌شود (شکل ۱-۷). منوی File را باز کرده و روی Exit کلیک کنید. پنجره‌ی کد با رویداد itmExit – click() ظاهر می‌شود.

۸- دستور Unload Me را به روال رویداد itmExit اضافه کنید (شکل ۱-۸).

۹- کلید F5 را فشار دهید تا کد اجرا شود.

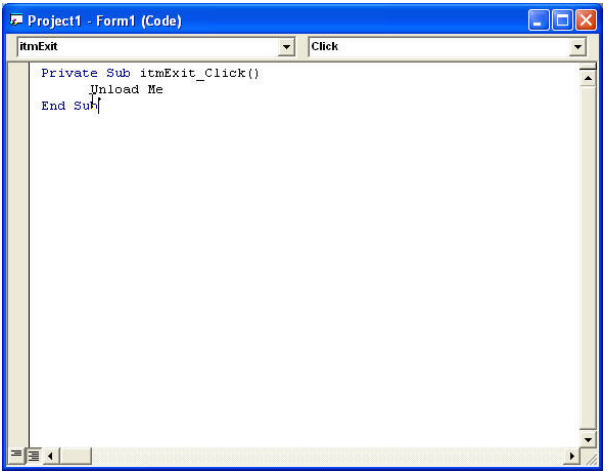


شکل ۱-٦) دکمه‌های فلش، امکان تورفتگی و سازماندهی گزینه‌ها در لیست منو را فراهم می‌کنند.



شکل ۱-۷) یک گزینه در منو شبیه هر کنترل دیگری در ویژوال بیسیک است که دارای مشخصه‌ها و یک رویداد به نام Click است.

در این مثال، یک منوی ساده‌ای را ایجاد کردیم و تکنیک‌های اصلی ایجاد منو را شرح دادیم. برای قطع برنامه در کد مربوط به گزینه‌ی Exit می‌توان از دستور End نیز استفاده کرد.



شکل ۱-۸) برای روال رویداد گزینه‌ها مانند سایر کنترل‌های VB، کدی را بنویسید.

۱-۳-۱ تنظیم مشخصه‌های منو

همانطور که قبلاً نیز ذکر شد، منو کنترلی با مجموعه‌ای از مشخصه‌ها و یک رویداد Click() است. جدول ۱-۲ متداول‌ترین مشخصه‌های مورد استفاده برای شیء Menu را نشان می‌دهد.

جدول ۱-۲) مشخصه‌های متداول شیء Menu

مشخصه	مقدار/ نوع	شرح
Caption	String	متنی که روی نوار منو ظاهر می‌شود.
Checked	Boolean	یک علامت √ قبل از رشته‌ی Caption گزینه قرار می‌دهد.
Enabled	Boolean	در صورتی که True باشد، رشته‌ی Caption به صورت خاکستری و غیرفعال نخواهد بود.
Name	String	نام شیء، فقط در زمان طراحی قابل دسترس است.
Shortcut	N/A	یک ترکیب کلیدی که امکان دسترسی به گزینه را فراهم می‌کند. این مقدار را فقط در زمان طراحی و از طریق فرمتی که در کادر لیست shortcut در

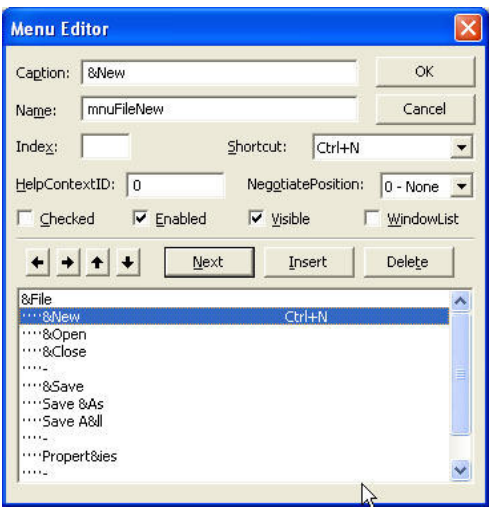
		Menu Editor ظاهر می‌شود، انتخاب کنید.
WindowList	Boolean	یک منوی سطح بالا در فرم MDI ایجاد می‌کند تا لیستی از پنجره‌های باز را نمایش دهد.

مانند هر کنترلی، می‌توان هنگام ایجاد منو یا گزینه‌ی منو، مشخصه‌ی Name را مقداردهی کرد. عدم مقداردهی Name سبب بروز خطا می‌شود. خطای دیگر، ننوشتن رشته‌ای در مشخصه‌ی Caption است که سبب نمایش یک خط خالی خواهد شد.

۱-۳-۲ اضافه کردن کلیدهای دسترسی به گزینه‌های منو

علاوه بر کلیک کردن روی یک گزینه‌ی منو برای اجرای یک وظیفه، می‌توان با استفاده از کلیدهای دسترسی به گزینه‌ها دسترسی پیدا کرد. کلیدهای دسترسی به کاربران امکان انتخاب گزینه‌ها با فشار دادن Alt و سپس حرف تعیین شده به عنوان کلید دسترسی را ارایه می‌دهند. بعد از اینکه منو باز شد، کاربران می‌توانند با فشار دادن کلید دسترسی، گزینه را انتخاب کنند.

برای تعریف یک کلید دسترسی، در کادر متن Caption گزینه قبل از نویسه‌ی مورد نظر، از نویسه‌ی & استفاده کنید (شکل ۱-۹).



شکل ۱-۹) کلیدهای میانبر را قبل از کامپایل کردن برنامه تعیین کنید.

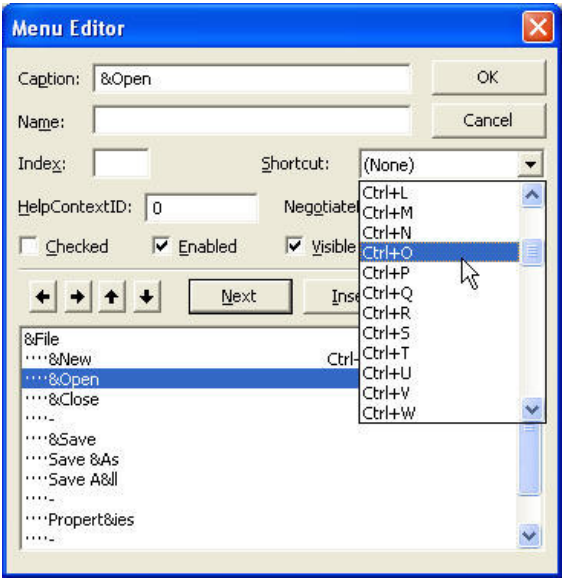
نکته:

از دو کلید دسترسی مشابه در یک بخش منو استفاده نکنید.

کلیدهای دسترسی مطابق با منوها گروه‌بندی می‌شوند و در صورتی می‌توانید از حرف یکسان برای گزینه‌ها استفاده کنید که در منوهای مختلف ظاهر شوند. اگر دو کلید دسترسی مشابه در یک بخش منو قرار دهید، اولین کلید در سلسله مراتب ابتدا اجرا خواهد شد. هنگامی که کلید برای دومین بار فشار داده شود، گزینه‌ی بعدی با همان کلید، انتخاب می‌شود.

۱-۳-۳ اضافه کردن کلیدهای میانبر به گزینه‌ها

از طریق Application Wizard نمی‌توان برای گزینه‌ها کلیدهای میانبر اضافه کرد. ولی انجام این کار در Menu Editor ممکن است استفاده از کلیدهای میانبر، روش دیگری برای اجرای وظایف گزینه‌ها از طریق صفحه کلید است. کلید میانبر مربوط به هر منو را می‌توان در کادر محاوره‌ای Menu Editor از لیست Shortcut انتخاب کرد (شکل ۱-۱۰).



شکل ۱-۱۰) هر برنامه‌ی کاربردی فقط می‌تواند یک نمونه از کلید میانبر را داشته باشد. اگر از Ctrl+N برای New استفاده کرده باشید، Ctrl+N را برای گزینه‌ی Open به کار نبرید. هنگام تعیین کلید میانبر تکراری، VB خطایی را نشان می‌دهد (شکل ۱-۱۱).

شکل ۱-۱۱) هنگامی که روی Ok کلیک می‌کنید، کادر محاوره‌ای Menu Editor خطایی را اعلان می‌کند.

۱-۳-٤ ایجاد منوهای بازشو

دو نوع منو وجود دارد: نوار منویی و منوی بازشو. **نوار منو** با ویزارد ایجاد شده و در ویرایشگر منو ویرایش می‌شود. **منوی بازشو**، منویی است که در هر جایی از فرم ظاهر می‌شود.

همان طور که مشاهده کردید با استفاده از Menu Editor می‌توان منوهایی در فرم ایجاد کرد که این منوها به طور استاندارد و پیش فرض، در سمت چپ بالای فرم ظاهر می‌شوند. اما ممکن است با منوهایی مواجه شده باشید که در موارد مورد نیاز (مثلاً هنگام کلیک راست) ظاهر می‌شوند. برای ایجاد چنین منوهایی نیز از Menu Editor استفاده می‌شود. در حقیقت، در مواقع مورد نیاز می‌توانیم قسمتی از منوی تعریف شده به وسیله‌ی Menu Editor را در محل اشاره‌گر یا در محل مورد نظر فعال کنیم. در VB با استفاده از دستور Popup می‌توان قسمتی از منوها را به همراه زیرمنوهای بعد از آن در ناحیه‌ای از فرم ظاهر کرده و از آن استفاده نمود.

شکل کلی این دستور به صورت زیر است:

PopupMenu MenuName [Flag], [xpos], [ypos]

که دراین ساختار MenuName نام منویی است که می‌خواهیم در فرم نمایش داده شود. هم چنین دو پارامتر اختیاری Ypos و Xpos ، مختصات محل ظاهر شدن منو را تعیین می‌کنند. اگر از این دو مقدار استفاده نشود VB به طور خودکار، مختصات محل فعلی ماوس را به جای این دو متغیر استفاده می‌کند. به عنوان مثال، دستور زیر سبب می‌شود تا منوی Edit، در محل فعلی ماوس نمایش داده شود:

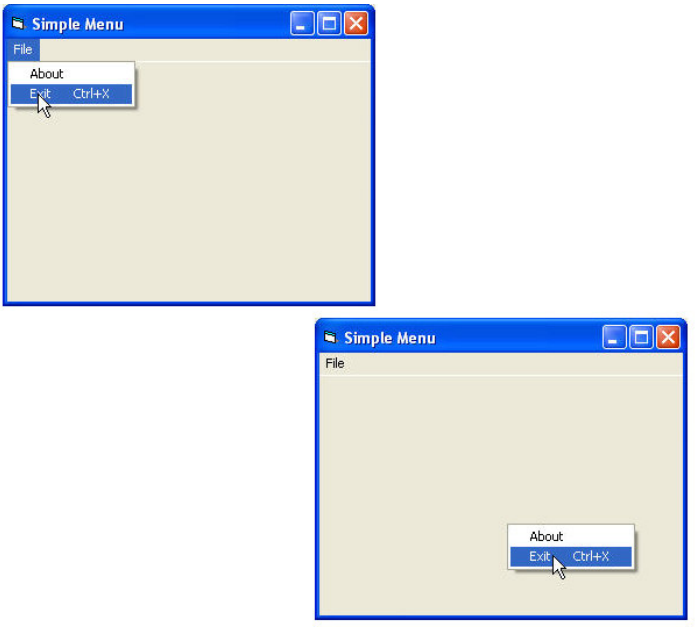
PopupMenu MenuEdit

می‌توانید نمونه‌ی این نوع منوها را تقریباً در همه جای ویندوز مشاهده کنید. در هر قسمت از ویندوز اگر کلیک راست کنید معمولاً منویی ظاهر شده و گزینه‌هایی را ارایه می‌کند. طریقه‌ی ایجاد چنین منوهایی را در مثال فوق آموختید. ولی ممکن است سؤال کنید وقتی روی یک فرم کلیک می‌شود، برنامه چگونه می‌تواند تشخیص دهد که کدام یک از کلیدهای ماوس فشار داده شده است. این مطلب را در کتاب برنامه‌سازی ۱ شرح داده‌ایم که می‌توانید برای یادآوری به فصل ۶ آن کتاب رجوع کنید.

به عنوان مثال، هنگامی که روی دسک‌تاپ ویندوز، کلیک راست می‌کنید، یک منوی بازشو ظاهر می‌شود. در صورتی که بخواهید می‌توانید یک فرمان را هم در نوار و هم منوی بازشو قرار دهید (شکل ۱-۱۲). کد زیر چگونگی انجام این کار را شرح می‌دهد:

```
Private Sub Form_MouseDown (Button As Integer, shift As Integer, X As_
Single,Y As Single)
    If Button = 2 Then
        PopupMenu mnuFile
   End If
End Sub
```

برای اینکه این کد کار کند، نیاز به ایجاد فرمی دارید که شامل یک کنترل Menu به نام mnuFile باشد که حداقل دارای یک گزینه است.



شکل ۱-۱۲) هنگامی که از متد PopupMenu استفاده می‌کنید، فقط گزینه‌های منو ظاهر خواهند شد.

مثال ۱-۲: ایجاد منوهای پیچیده)

اکنون که چگونگی استفاده از Menu Editor را آموختید، یک ویراستار متن ساده‌ای را ایجاد می‌کنیم که از یک سیستم منو استفاده می‌کند.

ویراستار متنی را ایجاد کنید که به کمک گزینه‌های مختلف در منوها، توانایی انجام عملیات زیر را داشته باشد.

- ایجاد فایل جدید
- باز کردن فایل موجود
- ذخیره‌ی فایل
- معکوس کردن قلم متن و تنظیم رنگ زمینه
- شرح حق کپی‌رایت

- خروج از برنامه
- برگرداندن عمل قبلی
- Cut، Copy و Paste متن
- انتخاب کل متن

قبل از شروع کدنویسی، باید ویژگی‌های برنامه را مرور کنید تا سیستم منـو را بـه طـور مناسـب طراحی و گروه‌بندی کنید. اغلب، نوارهای منو با منوی File و Edit شروع می‌شوند.

ترتیب زیر برای ویراستار متن، مناسب خواهد بود.

File Menu	Edit Menu
New	Undo
Open	Cut
Save	Copy
Settings	Paste
About	Select All
Exit	

به گروه بندی سیستم منو توجه کنید. می‌توان این منو را در Menu Editor پیـاده سـازی کـرد. جدول ۱-۳، سلسله مراتب منو و مشخصه‌های Name، Caption و کلیدهای میانبر برای هـر شـیء منو را نشان می‌دهد.

جدول ۱-۳) شیءهای منو برای ویراستار متن

Name	***Caption***	***Level***	***Shortcut***
mnuFile	&File	0	None
itmNew	&New	1	None
itmOpen	&Open	1	None
itmSave	&Save	1	None
sepOne	- (a hyphen)	1	None
itmSettings	Se&ttings	1	None
itmBlackOnWhite	Black On White	2	None
itmWhiteOnBlack	White On Black	2	None
itmAbout	&About	1	None

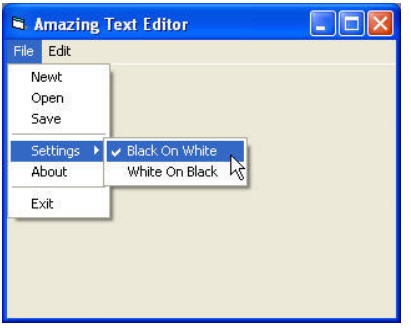
sepTwo	-	1	None
itmExit	E&xit	1	Ctrl+X
mnuEdit	&Edit	0	None
itmUndo	&Undo	1	Ctrl+Z
sepThree	-	1	None
itmCut	Cu&t	1	Ctrl+X
itmCopy	&Copy	1	Ctrl+C
itmPaste	&Paste	1	Ctrl+V
sepFour	-	1	None
itmSelectAll	Select &All	1	Ctrl+A

اضافه کردن خطوط جدا کننده به منوها

نکته:

از خطوط جدا کننده در سطح صفر سلسله مراتب منو نمی‌توان استفاده کرد. باید حداقل در سطح یک تورفتگی باشید.

با تایپ کردن یک خط تیره در کادر متن Caption ویرایشگر منو، یک خط جدا کننده بین گزینه‌ها اضافه کنید. هنگامی که برنامه را اجرا می‌کنید، خطوط جدا کننده در منوها ظاهر می‌شوند (شکل ۱-۱۳). برای جداکننده‌ها از پیشوند Sep استفاده کرده‌ایم.



شکل ۱-۱۳) اگر گزینه‌ای دارای زیرمنویی باشد، در سمت راست آن یک فلش را مشاهده خواهید کرد.

بعضی از گزینه‌ها حالت فعال و غیرفعال دارند که این عمل را می‌توان با نمـایش علامـت ✓ در کنار گزینه به کمک مشخصه‌ی Checked انجام داد. کـد زیـر، چگـونگی انجـام ایـن کـار را بـرای گزینه‌ی رنگ قلم نمایش می‌دهد:

```
01 Private Sub itmBlackOnWhite_Click()
02   `Set the color scheme for Black on White
03   txtMain.BackColor = vbWhite
04   txtMain.ForeColor = vbBlack
05
06   `Set the menu checks accordingly
07   itmBlackOnWhite.Checked = True
08   itmWhiteOnBlack.Checked = False
09 End Sub
```

برش، کپی و برگرداندن به کمک شیء Clipboard

یکی از ویژگی‌های مهم سیستم عامل ویندوز، قابلیت انتقال داده‌ها از یک برنامـه‌ی کـاربردی بـه دیگری از طریق Clipboard است. تمام برنامه‌های کاربردی دارای دسترسی به Clipboard هستند. می‌توان هر شیء ثبت شده در ویندوز از متن ساده گرفته تا مقادیر عددی را در Clipboard ذخیـره کرد.

ویژوال بیسیک امکان دسترسی بـه Clipboard وینـدوز را از طریـق شـیء Clipboard فـراهم می‌کند. این شیء هیچ مشخصه‌ای ندارد ولی دارای چندین متد است. جدول ۱-۴، متدهای مختلـف این شیء را نشان می‌دهد.

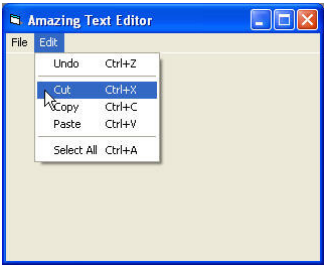
جدول ۱-٤) متدهای Clipboard

متد	شرح
Clear	همه‌ی داده‌ها را از Clipboard پاک می‌کند.
GetData	گرافیکی را از Clipboard برمی‌گرداند.
GetFormat	یک عدد صحیح برمی‌گرداند که اشاره بـه نـوع داده‌هـا در Clipboard است.
GetText	متن ASCII را از Clipboard بازیابی می‌کند.

SetData	گرافیکی را به Clipboard ارسال می‌کند.
SetText	متن ASCII را به حافظه‌ی Clipboard ارسال می‌کند.

گزینه‌های Cut، Copy و Paste از منوی Edit ویراستار متن (شکل ۱-۱۴)، از شیء Clipboard برای مقداردهی و بازیابی متن از Clipboard استفاده می‌کنند. کد زیر، متن انتخاب شده را به حافظه‌ی Clipboard کپی می‌کند:

```
Private Sub itmCopy_click ()
        Clipboard.SetText txtMain.SelText
End Sub
```



شکل ۱-۱۴) گزینه‌های منوی Edit از کلیدهای میانبر استاندارد ویندوز استفاده می‌کنند.

در این کد، مقدار مشخصه‌ی SelText مربوط به کادر متن بدست آمده و از متد SetText برای ارسال متن به حافظه‌ی Clipboard استفاده می‌شود.

برنامه‌ی زیر، چگونگی بازیابی متن از Clipboard را نشان می‌دهد.

```
01 Private Sub itmPaste_Click()
02  Dim Temp$            `Text from clipboard
03  Dim strLeft As String  `Holds text from left of cursor
04  Dim strRight As String `Holds text from right of cursor
05  Dim strFull As String  `Full text
06
07   `Get the text from the clipboard
08   Temp$ = Clipboard.GetText(vbCFText)
...
32 End Sub
```

انتخاب متن در یک TextBox

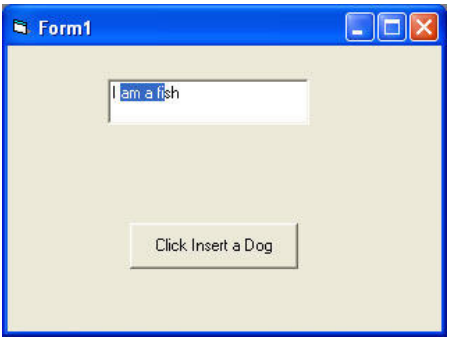
کد Cut و Paste در ویراستار متن، متن انتخاب شده را به Clipboard متنقل می‌کنند. متن کادر متن به وسیله‌ی مشخصه‌ی SetText انتخاب می شود.

کنترل کادر متن دارای چندین ویژگی انتخاب متن است. هنگامی که روی کلمه‌ای دابل کلیک کنید، کادر متن به طور خودکار، نویسه‌های آن کلمه را مشخص (های‌لایت) می‌کند. در صورتی که اشاره‌گر ماوس را روی خطی از متن، درگ کنید، آن خط به طور خودکار، انتخاب می‌شود. هنگامی که در کادر متن کلیک کنید، مکان‌نما بین نویسه‌های مورد نظر ظاهر می‌شود.

کادر متن دارای سه مشخصه‌ی انتخاب است: SelStart، SelLength و SelText.

مقدار SelStart محل شروع رشته در داخل محتوای کادر متن است. مقدار SelLength تعداد نویسه‌های مشخص شده در طول عمل انتخاب است. مقدار SelText نویسه‌های مشخص شده است. کد زیر، چگونگی استفاده از مشخصه‌های SelStart و SelLength برای انتخاب بلاکی از متن را نشان می‌دهد (شکل ۱-۱۵).

```
Private Sub Form_click ()
    Text1.SelStart = 3
    Text1.SelLength = 6
End Sub
```



شکل ۱-۱۵) مشخصه‌های SelStart و SelLength کادر متن را به یک کنترل قوی تبدیل می‌کنند.

نه تنها می‌توان این مشخصه‌ها را خواند بلکه می‌توان آنها را در زمان اجرا مقداردهی کرد. اگر SelText را در زمان اجرا مقداردهی کنید، ویژوال بیسیک به طور خودکار، مقدار رشته‌ی SelText را در محل مکان‌نما و درون کادر متن، درج کرده و متن موجود را جابه‌جا می‌کند:

```
Private Sub Command 1-Click ( )
```

```
    Text1.SelText = “Dog ”
End Sub
```

شکل ۱-۱۶) در صورتی که می‌خواهید در یک کادر متن به طور مکرر، متنی را درج کنید، مشخصه‌ی SelText پیشنهاد مناسبی است.

همچنین کد مربوط به ویراستار متن شامل توابع Right()، Left()، Len() و دستور MsgBox است که با این توابع و دستورها در درس برنامه‌سازی ۱ آشنا شده‌اید.

خودآزمایی و تحقیق

۱. در چه مواقعی از منوها استفاده می‌شود؟
۲. در کادر Menu Editor، گزینه‌ی Name به چه منظوری مورد استفاده قرار می‌گیرد؟
۳. انواع منوها را نام ببرید.
۴. وقتی کاربر از کلید میانبر استفاده می‌کند، کدام رویداد تحریک خواهد شد؟
۵. در پنجره‌ی Menu Editor گزینه‌ی WindowList به چه منظوری استفاده می‌شود؟
۶. تحقیق کنید که اگر سرمنویی غیرفعال شود، زیرمنوهای آن به چه شکلی مشاهده می‌شوند.
۷. فرمی ایجاد کنید که اگر روی آن کلیک راست کنید، منویی باز شود و امکان تغییر رنگ فرم به رنگ‌های قرمز، سبز و آبی را فراهم کند.

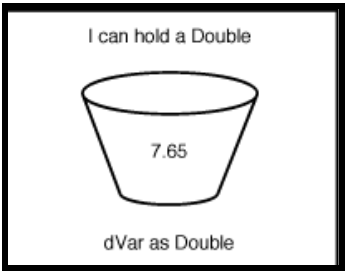
فصل دوم
آرایه‌ها

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

- مفهوم آرایه‌های یک‌بعدی و چندبعدی را شرح داده و از آن‌ها در برنامه‌های خود استفاده کند.
- روش مرتب‌سازی حبابی را شرح داده و عناصر آرایه را به روش حبابی مرتب کند.
- روش‌های جستجوی خطی و دودویی را توضیح داده و برای جستجوی عنصری در بین عناصر آرایه استفاده کند.
- از کنترل‌های ListBox ، ComboBox و Scrolls در برنامه‌های خود استفاده کند.
- مفهوم آرایه‌های کنترلی را شرح دهد.
- آرایه‌های دینامیکی را اعلان و به کار ببرد.

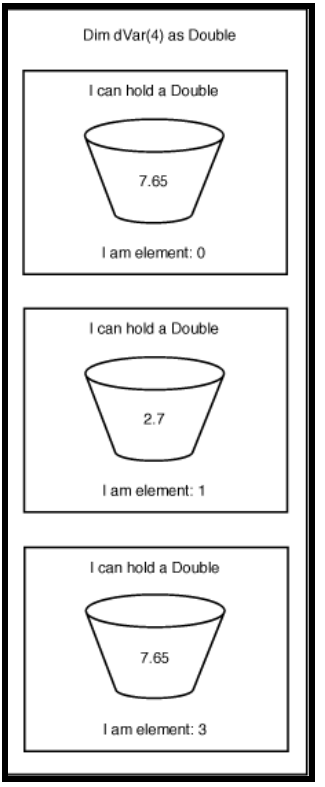
۲-۱ آرایه چیست؟

مجموعه‌ای از متغیرهای مشابه که همگی دارای نام و نوع داده‌ی یکسان هستند را **آرایه** می‌گویند. همانطور که می‌دانید متغیر ظرفی است که یک مقدار را در خود نگه می‌دارد که همیشه این مقدار تغییر می‌کند.



شکل ۲-۱) متغیر، محلی برای نوع داده‌ی خاصی است.

فرض کنید که آرایه مجموعه‌ای از این ظرف‌هاست. هر ظرفی در مجموعه نوع داده‌ی مشابهی با سایر ظرف‌ها نگه می‌دارد و همه‌ی ظرف‌ها دارای نام یکسانی هستند. هر ظرف داخل مجموعه را یک **عنصر** می‌نامند که دارای شماره‌ی خاصی است که محل (موقعیت) عنصر در داخل مجموعه را برمی‌گرداند. اولین عنصر آرایه معمولاً در موقعیت صفر قرار دارد. آرایه‌ها می‌توانند اندازه‌های مختلفی داشته باشند (شکل ۲-۲). یک آرایه ممکن است دارای سه عنصر بوده و دیگری ۳۰ عنصر داشته باشد. حتی ممکن است یک آرایه هیچ عنصری نداشته باشد (امکان اضافه کردن عنصر به آن وجود دارد).



شکل ۲-۲) آرایه مجموعه‌ای از متغیرهاست.

۲-۲ اعلان آرایه

می‌توان یک آرایه را به دو روش اعلان کرد:

- مانند یک متغیر
- با استفاده از کلیدواژه‌ی To

۲-۲-۱ اعلان آرایه مانند یک متغیر

برای اعلان یک آرایه، از شکل کلی زیر استفاده کنید:

```
Dim | Public | Private ArrayName (Subscript) As DataType
```

در این اعلان:

- Dim، Public و Private کلیدواژه‌های اعلان آرایه و تعیین حوزه‌ی عمل آن هستند. درباره‌ی حوزه‌ی عمل در فصل توابع توضیح خواهیم داد. اگر از Dim استفاده کنید، آرایه برای روالی که در آن تعریف شده است، قابل شناسایی خواهد بود. کلمات کلیدی Public و Private را در فصل توابع شرح خواهیم داد.
- ArrayName نام آرایه است که از قوانین نامگذاری متغیرها پیروی می‌کند.

نکته:

هنگام اعلان یک آرایه، اولین عنصر آرایه معمولاً در مکان صفر قرار دارد. می‌توان با نوشتن Option Base 1 در بخش General مدول داخل پروژه، اولین عنصر را در مکان یک قرار داد.

- Subscript مکان آخرین عنصر آرایه را تعیین می‌کند. به دلیل اینکه اولین عنصر آرایه معمولاً در مکان صفر است، بنابراین، اگر آرایه‌ای با اندیس ۶ اعلان کنید، آرایه دارای ۷ مکان و عنصر خواهد بود.
- As کلیدواژه‌ای است که اعلان نوع را مشخص می‌کند.
- DataType نوع داده‌ی معتبر در ویژوال بیسیک است مثل Integer یا Double.

بنابراین، برای اعلان آرایه‌ای از اعداد صحیح با ۵ عنصر، به صورت زیر خواهیم نوشت:

```
Dim iMyArray(4) As Integer
```

برای تعیین یک مقدار برای هر عنصر آرایه‌ی iMyArray، خواهیم داشت:

```
iMyArray(0) = 9
iMyArray(1) = 342
iMyArray(2) = 2746
```

```
iMyArray(3) = 0
iMyArray(4) = 8901
```

برای تغییر مقدار چهارمین عنصر این آرایه از ۰ به ۴۵ ، به صورت زیر عمل می‌کنیم:

```
iMyArray(3) = 45
```

به عنوان مثال، برای اعلان آرایه‌ای از ۹ رشته، از کد زیر استفاده خواهیم کرد:

```
01 Public strMyArray(8) As String
02
03 strMyArray(0) = "I am a pitcher."
04 strMyArray(1) = "I am a catcher."
05 strMyArray(2) = "I play first base."
06 strMyArray(3) = "I play second base."
07 strMyArray(4) = "I play third base."
08 strMyArray(5) = "I play shortstop."
09 strMyArray(6) = "I play left field."
10 strMyArray(7) = "I play center field."
11 strMyArray(8) = "I play right field."
```

۲-۲-۲ اعلان آرایه با کلیدواژه‌ی To

همچنین می‌توان آرایه را با استفاده از کلیدواژه‌ی To در داخل اندیس‌های آرایه اعلان کرد. به عنوان مثال، اگر می‌خواهید آرایه‌ای از ۵ متغیر عدد صحیح ایجاد کنید که اولین عنصر در مکان ۱ و آخرین عنصر در مکان ۵ باشد، از دستور زیر استفاده کنید:

```
Dim iMyArray ( 1 to 5) As Integer
```

این روش، راه ساده‌ای برای شروع آرایه از مکانی غیر از صفر است.

۲-۳ تغییر تعداد عناصر آرایه

اگر چه معمولاً تعداد عناصر آرایه هنگام اعلان آن، تعیین می‌شود ولی امکان تغییر اندازه‌ی آرایه وجود دارد. هنگامی که تعداد عناصر در یک آرایه را تغییر دهید، آن را تغییر بعد می‌دهید. برای انجام این کار، از کلیدواژه‌ی ReDim به شکل زیر استفاده کنید:

```
ReDim [Preserve] ArrayName (Subscript) As DataType
```

- Redim کلیدواژه‌ی ویژوال بیسیک است که مشخص می‌کند آرایه تغییر بعد می‌یابد.
- Preserve یک کلیدواژه‌ی اختیاری است که امکان نگهداری مقادیر تمام عناصر از قبل تعریف شده در آرایه را فراهم می‌کند. اگر از این کلیدواژه استفاده نکنید، مقدار تمام عناصر

برای نوع داده‌های عددی به صفر و برای رشته‌هایی با طول متغیر به رشته‌ای به طول صفر تغییر می‌یابند. رشته‌های با طول ثابت، با صفر پر شده و متغیرهای Variant با EMPTY مقداردهی خواهند شد که بسته به نوع عبارت، به صفر یا رشته‌ای به طول صفر تبدیل می‌شوند.

- ArrayName نام آرایه است.
- Subscript اندیس آخرین خانه‌ی آرایه است.
- As کلیدواژه‌ای است که اعلان نوع را مشخص می‌کند. هنگام تغییر بعد یک آرایه، کلیدواژه‌ی As اختیاری است.
- DataType یک نوع داده‌ی معتبر در ویژوال بیسیک است. هنگام تغییر بعد یک آرایه، DataType اختیاری است و با کلیدواژه‌ی ReDim نمی‌تواند تغییر یابد مگر اینکه آرایه از نوع Variant باشد.

نکته:

پیاده‌سازی واقعی دستور ReDim متفاوت با این شرح است. اگر آرایه‌ای ایجاد می‌کنید که بعداً می‌خواهید آن را تغییر اندازه دهید، نمی‌توانید اندازه‌ی آن را هنگام اعلان، تعیین کنید. بنابراین، در عمل نمی‌توان آرایه‌ی تعریف شده در کد قبلی را به صورت زیر تغییر اندازه داد:

```
ReDim Preserve StrMyArray(9)
StrMyArray (9) = ‘I am the designated hitter.”
```

برای ایجاد آرایه‌ای که بعداً تغییر اندازه خواهید داد، باید ابتدا آرایه را بدون هیچ عنصری، ایجاد کنید. کد زیر، روش مناسب برای ایجاد آرایه‌ای که بعداً تغییر اندازه خواهد یافت را نشان می‌دهد:

```
01 `Create an array without any elements
02 Dim strMyArray() as String
03
04 `Dimension the array for 9 elements
05 ReDim strMyArray(8)
06
07 `Assign values to the array elements
08 strMyArray(0) = "I am a pitcher."
09 strMyArray(1) = "I am a catcher."
10 strMyArray(2) = "I play first base."
11 strMyArray(3) = "I play second base."
```

```
12 strMyArray(4) = "I play third base."
13 strMyArray(5) = "I play shortstop."
14 strMyArray(6) = "I play left field."
15 strMyArray(7) = "I play center field."
16 strMyArray(8) = "I play right field."
17
18 `Add an element and make it so all the values
19 `of the previous elements are kept intact
20 ReDim Preserve strMyArray(9)
21
22 `Assign a value to the new array element
23 strMyArray(9) = "I am the designated hitter."
```

توجه داشته باشید که در کد فوق، اولین دستور ReDim (خط ۵) از کلیدواژه‌ی Preserve استفاده نکرده است و دلیل آن هم این است که در آرایه مقداری وجود ندارد که نگهداری شود. در دومین دستور ReDim در خط ۲۰، کلیدواژه‌ی Preserve خیلی مهم است زیرا عناصر آرایه دارای مقدار هستند که نمی‌خواهیم از دست بروند. اگر از این کلیدواژه در خط ۲۰ استفاده نکنید، آرایه‌ی StrMyArray دارای مقادیر زیر خواهد بود:

```
strMyArray (0) = ""
strMyArray (1) = ""
strMyArray (2) = ""
strMyArray (3) = ""
strMyArray (4) = ""
strMyArray (5) = ""
strMyArray (6) = ""
strMyArray (7) = ""
strMyArray (8) = ""
strMyArray (9) = "I am the designated hitter."
```

۲-٤ آرایه‌های چندبعدی

آرایه‌هایی که تا اینجای فصل ایجاد کردیم، همانطوری که در شکل ۲-۲ مشاهده کردید، آرایه‌های یک بعدی (مجموعه‌ای یک سطری از متغیرها) بودند. در ویژوال بیسیک می‌توان آرایه‌هایی ایجاد کرد که حداکثر ۶۰ بعد داشته باشند، معمولاً آرایه‌های دوبعدی برای اغلب

پروژه‌های برنامه‌نویسی کافی خواهد بود. و به ندرت نیاز به ایجاد آرایه‌های سه بعدی و بیشتر خواهید داشت.

آرایه دوبعدی را شبیه صفحه‌ی بازی دوز در نظر بگیرید. (مجموعه‌ای از ستون‌ها و سطرها که از تداخل آنها خانه‌هایی بوجود می‌آیند). هر خانه دارای موقعیت تعریف شده‌ای است که به صورت شماره ستون و شماره سطر است.

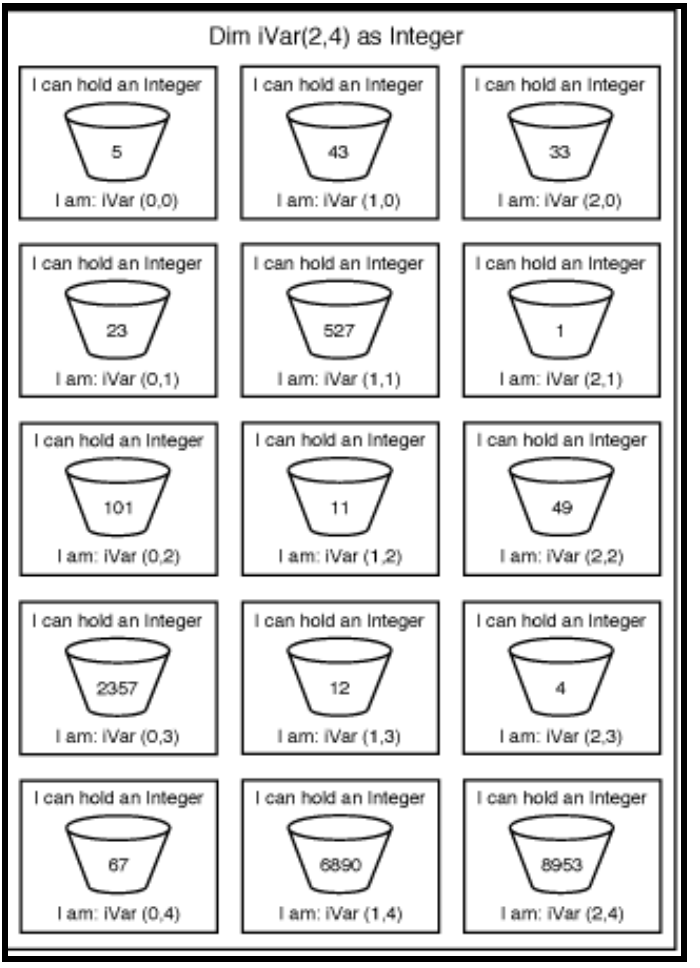
توجه داشته باشید که هر عنصر براساس مختصات ستون و سطر تعریف می‌شود. به عنوان مثال، عنصر iVar(0,0) دارای مقدار ۵ و عنصر iVar(2,2) دارای مقدار ۴۹ است.

برای ایجاد آرایه‌ی دوبعدی، از شکل کلی زیر استفاده کنید:

```
Dim | Public | Private ArrayName (SubscriptOfCols, subscriptOfRows) As DataType
```

با اغلب کلیدواژه‌های این دستور آشنا هستید. دو مورد جدید، عبارتند از:

- SubscriptOfCols – اندیس آخرین ستون آرایه است.
- SubscriptOfRows – اندیس آخرین سطر آرایه است.



شکل ۲-۳) آرایه‌ی دو بعدی را به صورت مجموعه‌ای از ظرف‌ها که در ستون‌ها و سطرها سازماندهی شده‌اند، تصور کنید.

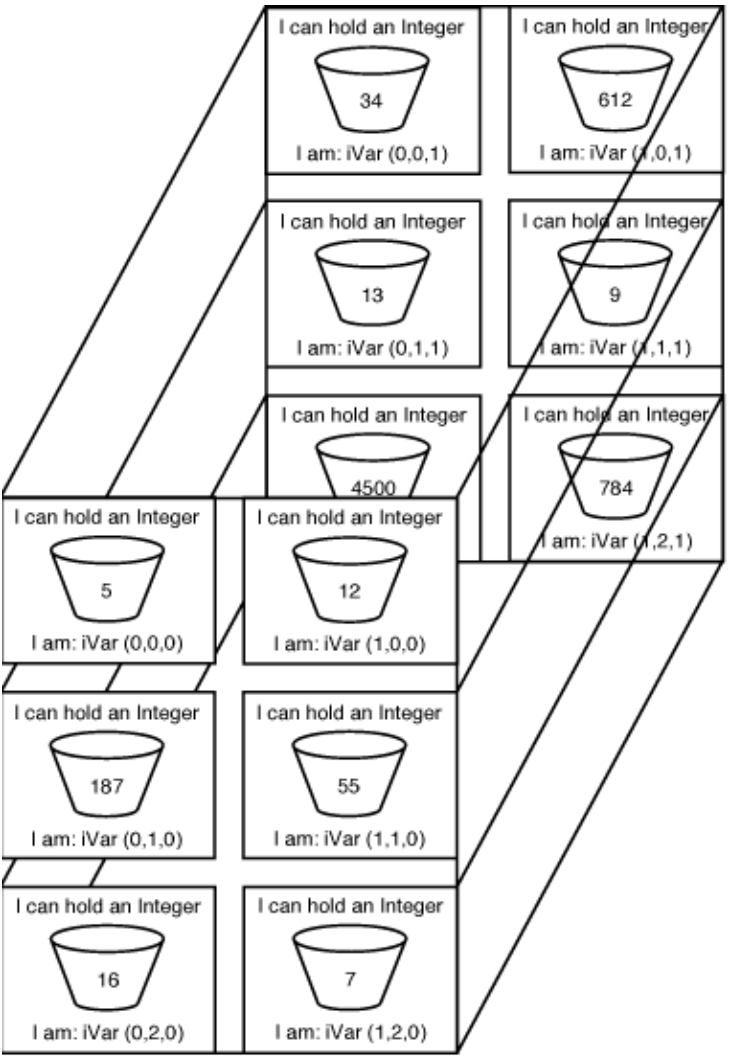
بنابراین، برای اعلان آرایه‌ی شکل ۲-۳، می‌توان نوشت:

```
Dim iVar (2,4) As Integer
```

اگر یک آرایه‌ی دوبعدی را به صورت مستطیل تصور می‌کنید، می‌توانید آرایه‌ی سه بعدی را به شکل یک مکعب مستطیل، در نظر بگیرید. برای اعلان آرایه‌ی سه بعدی از اعداد صحیح، دستور زیر را وارد کنید.

Dim iVar (1, 2, 1) As Integer

شکل ۲-۴، این آرایه‌ی سه بعدی را نشان می‌دهد.



شکل ۲-٤) آرایه‌ی سه بعدی (۱،۲،۱) دارای ۱۲ عنصر است. این آرایه دارای دو ستون و سه سطر به عمق ۲ است.

(تعداد ستون × تعداد سطر× تعداد صفحات عمق) = تعداد عناصر آرایه سه بعدی

در آرایه iVar مقدار تعیین شده برای هر عنصر که در شکل ۲-۴ نشان داده شده است، به صورت زیر خواهد بود:

```
iVar(0,0,0) = 5
iVar(0,1,0) = 187
iVar(0,2,0) = 16
iVar(1,0,0) = 12
iVar(1,1,0) = 55
iVar(1,2,0) = 7
iVar(0,0,1) = 34
iVar(0,1,1) = 13
iVar(0,2,1) = 4500
iVar(1,0,1) = 612
iVar(1,1,1) = 9
iVar(1,2,1) = 784
```

مشابه آرایه‌ی یک بعدی، می‌توان از کلیدواژه‌ی To برای اعلان محدوده‌ی اندیس‌های هر بعد در آرایه‌های چند بعدی نیز استفاده کرد. پس دستور زیر

```
Dim dMyArray (1 To 5, 3 To 8, 3 To 5) As Double
```

یک آرایه سه بعدی از مقادیر اعشاری خواهد بود که دارای ۵ ستون، ۶ سطر و ۳ صفحه‌ی عمق خواهد بود.

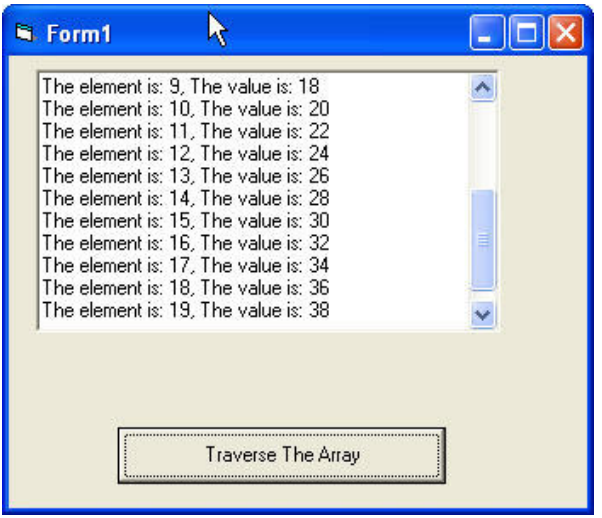
همچنین می‌توان از کلیدواژه‌ی ReDim برای تغییر اندازه‌ی آرایه‌ی چندبعدی استفاده کرد. اگر از کلیدواژه‌ی Preserve استفاده کنید، فقط آخرین بعد آرایه‌ی چندبعدی می‌تواند تغییر اندازه یابد و تعداد ابعاد نمی‌توانند تغییر کنند.

مثال ۲-۱) کاربرد حلقه‌ها برای پیمایش آرایه

می‌توانید از حلقه‌ی For…Next که قبلاً آموخته‌اید، برای پیمایش یک آرایه استفاده کنید. انجام این کار، می‌تواند هنگامی که مقادیر آرایه را تغییر می‌دهید یا گزارش می‌کنید، مفید باشد.

کد زیر، مثالی از کاربرد حلقه‌ی For…Next برای پیمایش آرایه است. این کد مربوط به رویداد Click دکمه‌ای به نام cmdTraverse است که آرایه ای به طول ۲۰ عنصر را ایجاد می‌کند. حلقه‌ی For…Next دوبار استفاده شده است (ابتدا برای تعیین مقادیر عناصر آرایه و سپس پیدا کردن مقدار هر عنصر و ایجاد رشته‌ای که مقدار هر عنصر را گزارش می‌کند). به شکل ۲-۵ توجه کنید.

```
01 Private Sub cmdTraverse_Click()
02 Dim i%
03 Dim iMyArray(19) As Integer
04 Dim BeginMsg$
05 Dim MidMsg$
06 Dim LoopMsg$
07 Dim FullMsg$
08
09 `Assign a value to each element in the array
10 `by using a loop to traverse to each element
11 `in the array.
12 For i% = 0 To 19
13 `Make the value of the element to be
14 `twice the value of i%
15 iMyArray%(i%) = i% * 2
16 Next i%
17
18 `Create the BeginMsg$ string
19 BeginMsg$ = "The element is: "
20 MidMsg$ = ", The value is: "
21 `Go through the array again and make
22 `a string to display
23 For i% = 0 To 19
24 LoopMsg$ = LoopMsg$ & BeginMsg$ & CStr(i%)
25 LoopMsg$ = LoopMsg$ & MidMsg$ & iMyArray(i%)
26
27 `Concatenate the loop message to the
28 `full message. Also add a line break
29 FullMsg$ = FullMsg$ & LoopMsg$ & vbCrLf
30
31 `Clean out the loop message so that
32 `new value next time through the loop
33 LoopMsg$ = ""
34 Next i%
35
36 txtTraverse.Text = FullMsg$
37 End Sub
```



شکل ۲-۵) این برنامه هر عنصر آرایه را با دو برابر شماره‌ی آن عنصر، مقداردهی می‌کند.

کد فوق ساده است. همانطور که می‌دانید حلقه‌ی For…Next یک متغیر شمارنده دارد که در این برنامه مقدار اولیه‌ی آن، با کران پایین آرایه و مقدار نهایی آن، با کران بالای آرایه مقداردهی شده است. هنگامی که حلقه را اجرا می‌کنید، متغیر شمارنده، همیشه در یک عنصر آرایه خواهد بود.

۲-۵ افزودن عنصرها به ListBox و ComboBox

در بین همه‌ی کنترل‌های استاندارد، ListBox و ComboBox مناسب‌ترین کنترل برای گروه‌بندی و لیست‌بندی اطلاعاتی است که کاربران می‌توانند انتخاب کنند. این کنترل‌ها به هم مرتبط بوده و روش کار با آنها یکسان است. تفاوت آنها در این است که ComoboBox از یک ListBox به همراه TextBox (همانطور که از نام آن پیداست) تشکیل شده است و کاربران می‌توانند مقداری را از لیست انتخاب کرده یا مقدار جدیدی را تایپ کنند.

کنترل‌های ListBox و ComboBox می‌توانند لیست‌ها را مدیریت کنند. لیست در ویژوال بیسیک، آرایه‌ای از رشته‌هاست که در مشخصه‌ی List قرار داده شده‌اند. (مشخصه‌ی List برای هر دو کنترل موجود است). عملیاتی که می‌توان با این کنترل‌ها انجام داد، اضافه و حذف کردن رشته‌ها از مشخصه‌ی List است. می‌توان رشته‌ها را در زمان طراحی یا زمان اجرا به مشخصه‌ی List این کنترل‌ها اضافه کرد.

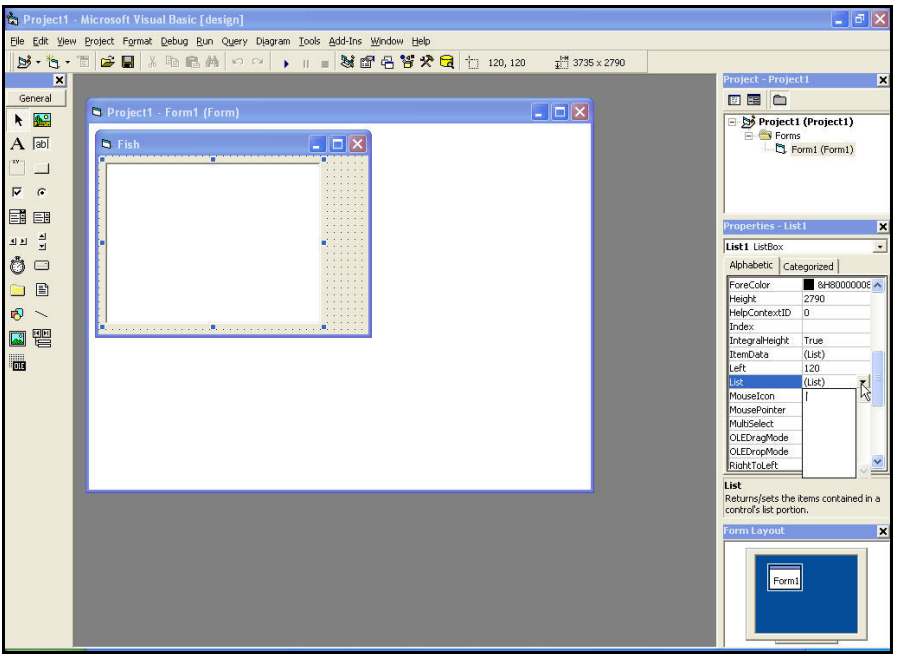
۲-۵-۱ افزودن رشته‌ها به ListBox یا ComoboBox در زمان طراحی

۱- یک کنترل ListBox یا ComboBox به فرم اضافه کنید.

۲- این کنترل را در روی فرم، انتخاب کنید.

۳- در پنجره‌ی Properties، مشخصه‌ی List (شکل ۲-۶) را انتخاب کرده و رشته‌هایی را برای این مشخصه تایپ کنید. برای افزودن چندین خط به مشخصه‌ی List، کلیدهای Ctrl+Enter را برای اضافه کردن خط جدید، فشار دهید (شکل ۲-۷).

در زمان طراحی، رشته‌هایی که به مشخصه‌ی List اضافه کرده‌اید، در ListBox روی فرم، ظاهر خواهند شد (شکل ۲-۸).



شکل ۲-۶) می‌توان رشته‌هایی را در زمان طراحی با وارد کردن مقادیر در مشخصه‌ی List به کنترل‌های ListBox یا ComboBox اضافه کرد.

شکل ۲-۷) برای افزودن چندین عنصر به لیست ListBox در زمان طراحی، کلیدهای Ctrl+Enter را فشار دهید.

شکل ۲-۸) می‌توان لیست ListBox را در زمان طراحی، مشاهده کرد.

برای اضافه کردن رشته‌ای به List کنترل ListBox یا ComoBox در زمان اجرا، از متد AddItem که شکل کلی آن به صورت زیر است، استفاده کنید:

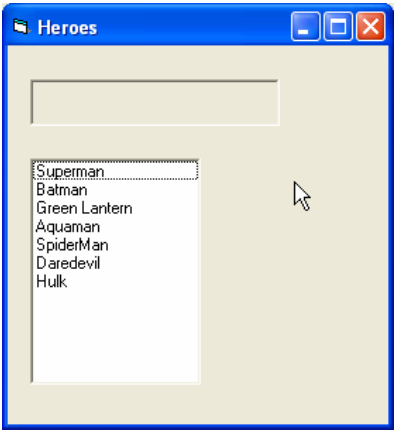
Object.AddItem StringToAdd

- Object مشخصه‌ی Name کنترل ListBox یا ComboBox است.

- AddItem کلیدواژه‌ی ویژوال بیسیک برای متد است.
- StringToAdd رشته‌ای است که می‌خواهید به لیست کنترل اضافه کنید.

کد زیر، چگونگی استفاده از متد AddItem در روال رویداد Load را برای افزودن رشته‌ها به ListBox نشان می‌دهد. شکل ۲-۹، نتیجه‌ی رویداد Load فرم را نشان می‌دهد.

```
01 Private Sub Form_Load()
02 lstHero.AddItem "Superman"
03 lstHero.AddItem "Batman"
04 lstHero.AddItem "Green Lantern"
05 lstHero.AddItem "Aquaman"
06 lstHero.AddItem "SpiderMan"
07 lstHero.AddItem "Daredevil"
08 lstHero.AddItem "Hulk"
09 End Sub
```



شکل ۲-۹) می‌توان از متد AddItem در رویداد Load استفاده کرد تا مشخصه‌ی List کنترل ListBox را مقداردهی کند.

۲-۵-۲ انتخاب عنصرها از لیست

برای آشنایی با چگونگی تعیین مقدار رشته‌ی انتخابی در List کنترل ListBox و CombBox، نیاز دارید بدانید که List آرایه‌ای از رشته‌هاست.

همانطور که می‌دانید برای دسترسی به عنصر دوم آرایه‌ای به نام myArray، دستوری به صورت زیر می‌نویسیم (عنصر اول در خانه ی صفر قرار دارد):

```
MyValue = MyArray (1)
```

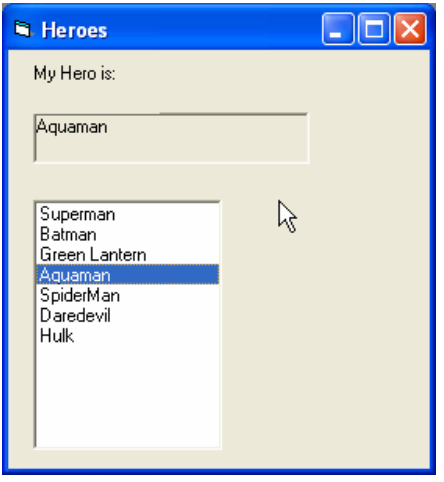
کنترل‌های ListBox و ComboBox نیز از قالب مشابهی استفاده می‌کند. بنابراین برای بدست آوردن دومین رشته در ListBox به نام LsfHero، دستوری به صورت زیر خواهیم نوشت:

```
secondString$ = LstHero.List(1)
```

در هر کنترل ListBox و ComboBox ، شماره‌ی عنصر انتخاب شده در لیست، در مشخصه‌ای به نام ListIndex قرار می‌گیرد. بنابراین، برای تعیین مقدار رشته‌ی انتخاب شده در LstHero، کد زیر را به کار خواهید برد:

```
Private Sub LstHero_Click()
    LbLHero.Caption = LstHero.List(LstHeroIndex)
End Sub
```

هنگامی که کاربر روی رشته‌ای در ListBox کلیک می‌کند، کد اجرا شده و مقدار انتخاب شده در لیست را به مشخصه‌ی Caption برچسبی به نام LbLHero انتساب می‌دهد. شکل ۲-۱۰، رویدادهایی که هنگام انتخاب یک رشته رخ می‌دهند را نشان می‌دهد.



شکل ۲-۱۰) می‌توان برای رویدادهای Click، MouseUp یا MouseDown یک ListBox برنامه نوشت تا مقدار انتخاب شده را بدست آورد.

روش سریع‌تر برای بدست آوردن مقدار یک رشته‌ی انتخاب شده در کنترل‌های ListBox یا ComboBox، استفاده از مشخصه‌ی Text است. به عنوان مثال، می‌توان از کد زیر برای یک ListBox استفاده کرد:

```
Dim strMyStr As String
StrMyStr = List1.Text
```

برای یک ComboBox، از کد زیر استفاده کنید:

```
Dim StrMyStr As String
StrMyStr = Combo1.Text
```

۲-۵-۳ حذف عنصرها از لیست

با استفاده از متد RemoveItem می‌توان رشته‌ای را از لیست ListBox و ComboBox حذف کرد:

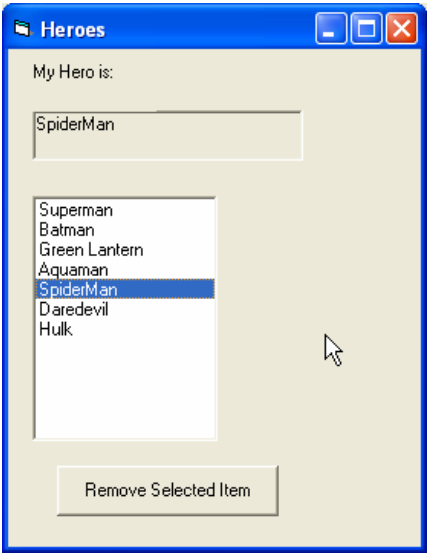
```
Object.RemoveItem Index
```

- Object، مشخصه‌ی Name کنترل ListBox یا ComboBox است.
- RemoveItem کلیدواژه‌ی ویژوال بیسیک برای متدی است که عناصر را از لیست حذف می‌کند.
- Index محل رشته‌ای است که می‌خواهید از لیست حذف کنید. برای حذف عنصر انتخاب شده از لیست، از مشخصه‌ی ListIndex استفاده کنید.

شکل ۲-۱۱ بهینه‌سازی شده‌ی برنامه‌ی شکل ۲-۱۰ را نشان می‌دهد. یک دکمه برای حذف رشته اضافه شده است که کاربر از ListBox انتخاب می‌کند. کد زیر، چگونگی استفاده از متد RemoveItem را نشان می‌دهد.

```
Private Sub cmdRemove_Click()
        lstHero.RemoveItem (lstHero.ListIndex)
        lstHero.Caption = " "
End Sub
```

هنگامی که عنصری را از ListBox یا ComboBox حذف می‌کنید، مطمئن باشید که مشخصه‌ی Caption برچسب را پاک کنید.



شکل ۲-۱۱) دکمه‌ی فرمان روشی برای حذف عنصری از لیست است.

۲-۵-٤ پاک کردن لیست

در صورتی که می‌خواهید تمام رشته‌های موجود در ListBox یا ComboBox را حذف کنید، از متد Clear استفاده کنید:

```
Object.Clear
```

بنابراین، برای پاک کردن لیست شکل ۲-۱۱، خواهیم نوشت:

```
lstHero.Clear
```

۲-۵-۵ آشنایی با شیوه‌های ComboBox

کنترل‌های ListBox و ComboBox دارای وجوه مشترک زیادی هستند ولی هر کدام دارای محدودیت کاربرد می‌باشند. یک ListBox فضای بیشتری را نسبت به ComboBox اشغال می‌کند و نمی‌توان در آن یک داده‌ی خارج از لیست را انتخاب یا وارد کرد. ComboBox قابلیت انعطاف بیشتری را ارایه می‌دهد و از فضای فرم به صورت کارآمد استفاده می کند.

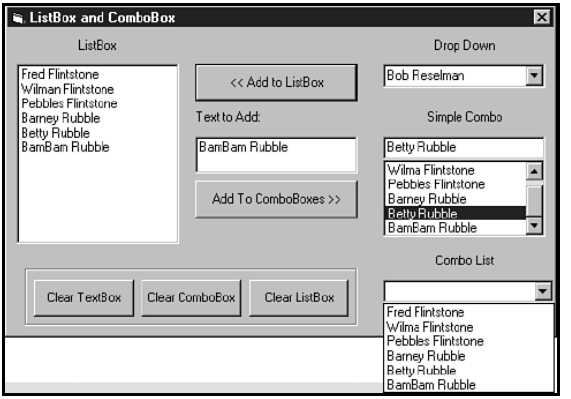
مشخصه‌ی Style کنترل ComboBox امکان تغییر مشخصه‌های عملیاتی و ظاهر کنترل را فراهم می‌کند. جدول ۲-۱، این شیوه‌ها را شرح می‌دهد و شکل ۲-۱۲، شیوه‌های ComboBox اعمال شده به فرم را نشان می‌دهد.

نکته:

هنگامی که یک ComboBox را با شیوه‌ی 1-Simple Combo به فرم اضافه می‌کنید، ComboBox تغییر اندازه یافته و ComboBox مشاهده نمی‌شود. مشخصه‌ی Height را افزایش دهید تا ListBox را نمایش دهد.

جدول ۲-۱) مقادیر مربوط به مشخصه‌ی Style کنترل ComboBox

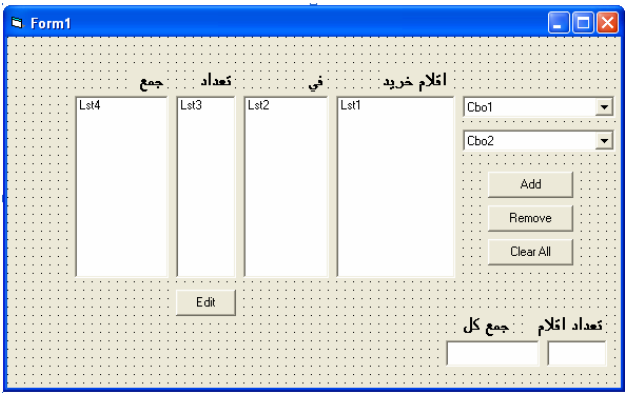
تنظیم	شرح
0 - Drop – Down Combo	یک لیست بازشو. کاربران می‌توانند داده های جدیدی را به ComboBox وارد کنند.
1 – Simple Combo	ترکیبی از TextBox و ListBox که بازشو نیست. کاربران می‌توانند داده‌ها را از ListBox انتخاب کرده یا داده‌ی جدیدی را در TextBox وارد کنند. اندازه‌ی این نوع ComboBox شامل بخش‌های ویرایش و لیست است.
2–Drop–Down List	یک لیست بازشوی فقط خواندنی که کاربران فقط می‌توانند داده‌ها را انتخاب کنند. کاربران نمی‌توانند داده‌ها را وارد کنترل کنند.



شکل ۲-۱۲) توجه کنید که با شیوه‌ی ComboBox بازشو، می‌توان داده‌های جدیدی را در زمان اجرا به کنترل اضافه کرد.

متداول‌ترین شیوه‌های ComboBox شیوه‌های ۰ و ۲ هستند. همانطور که قبلاً نیز بیان شد، هنگامی که می‌خواهید به کاربران امکان اضافه کردن داده‌های جدیدی که در لیست نیستند را ارایه کنید، از شیوه‌ی ComboBox بازشو (۰) استفاده کنید. در صورتی که می‌خواهید کاربران فقط امکان انتخاب داشته باشند، از شیوه‌ی لیست بازشو (۲) استفاده کنید.

مثال ۲-۲: محاسبه قیمت اقلام خریداری شده



شکل ۲-۱۳)

کد این برنامه به صورت زیر خواهد بود:

Private Sub sum()

```
    Dim SngS As Single, SngX As Single
    SngS = 0
    For i = 0 To Lst4.ListCount - 1
        SngX = Lst4.List(i)
        SngS = SngS + Val(SngX)
    Next
    lblsum = SngS
    lbln = Lst1.ListCount
End Sub

Private Sub CmdAdd_Click()
    Dim IntN As Integer, IntM As Integer
    IntN = Val(InputBox("تعداد خرید"))
    If IntN > 0 Then
        IntM = Cbo1.ListIndex
        Lst1.AddItem Cbo1.List(IntM)
        Lst2.AddItem Cbo2.List(IntM)
        Lst3.AddItem IntN
        Lst4.AddItem IntN * Val(Cbo2.List(IntM))
        Call sum
    End If
End Sub
Private Sub Cbo1_Click()
    Cbo2.ListIndex = Cbo1.ListIndex
End Sub
Private Sub CmdClear_Click()
    Dim YN As Integer
    YN = MsgBox("Do you want Clear all", vbYesNo + vbQuestion)
    If YN = vbYes Then
        Lst1.Clear
        Lst2.Clear
        Lst3.Clear
        Lst4.Clear
        Call sum
    End If
End Sub

Private Sub CmdEdit_Click()
    Dim IntN As Integer, IntM As Integer, IntX
    IntX = Lst3.ListIndex
    If IntX <> -1 Then
        IntN = Val(InputBox("تعداد خرید"))
        If IntN > 0 Then
            Lst3.List(Lst3.ListIndex) = IntN
            Lst4.List(IntX) = IntN * Val(Lst2.List(IntX))
            Call sum
        End If
```

```
    End If
End Sub


Private Sub CmdRemove_Click()
    Dim IntX As Integer
    IntX = Lst1.ListIndex
    If IntX >= 0 Then
        Lst1.RemoveItem IntX
        Lst2.RemoveItem IntX
        Lst3.RemoveItem IntX
        Lst4.RemoveItem IntX
    End If
End Sub


Private Sub Form_Load()
    Cbo1.AddItem "خودکار بیک آبی "
    Cbo1.AddItem "خودکار بیک قرمز"
    Cbo1.AddItem "دفتر ۴۰ برگ"
    Cbo1.AddItem " دفتر ۶۰ برگ"
    Cbo1.AddItem " دفتر ۱۰۰ برگ"
    Cbo1.AddItem "کاغذ کلاسور ۱ بسته"
    Cbo1.AddItem "پاک کن کوچک"
    Cbo1.AddItem " پاک کن متوسط"
    Cbo1.AddItem "پاک کن بزرگ"
    Cbo1.AddItem "کلاسور"
    Cbo2.AddItem "1000"
    Cbo2.AddItem "1100"
    Cbo2.AddItem "2500"
    Cbo2.AddItem "3500"
    Cbo2.AddItem "5000"
    Cbo2.AddItem "9000"
    Cbo2.AddItem "1000"
    Cbo2.AddItem "1500"
    Cbo2.AddItem "2500"
    Cbo2.AddItem "12500"
    Cbo1.ListIndex = 0
    Cbo2.ListIndex = 0
    lbln = 0
    lblsum = 0
End Sub
```

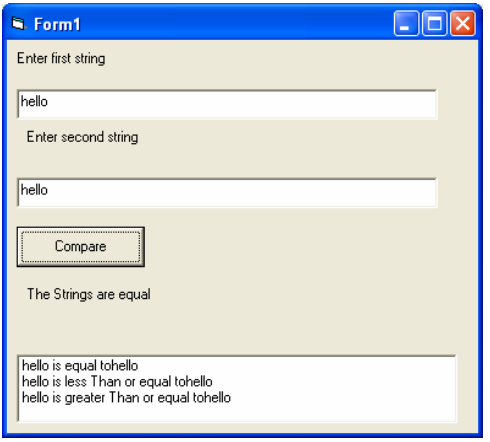
```
Private Sub Lst1_Click()
   Dim IntX As Integer
   IntX = Lst1.ListIndex
   Lst2.ListIndex = IntX
   Lst3.ListIndex = IntX
   Lst4.ListIndex = IntX
End Sub

Private Sub Lst2_Click()
   Dim IntX As Integer
   IntX = Lst2.ListIndex
   Lst1.ListIndex = IntX
   Lst3.ListIndex = IntX
   Lst4.ListIndex = IntX
End Sub

Private Sub Lst3_Click()
   Dim IntX As Integer
   IntX = Lst3.ListIndex
   Lst2.ListIndex = IntX
   Lst1.ListIndex = IntX
   Lst4.ListIndex = IntX
End Sub

Private Sub Lst4_Click()
   Dim IntX As Integer
   IntX = Lst4.ListIndex
   Lst2.ListIndex = IntX
   Lst3.ListIndex = IntX
   Lst1.ListIndex = IntX
End Sub
```

مثال ۲-۳ : مقایسه رشته‌ها

برنامه‌ی زیر از تابع StrComp و عملگرهای مقایسه‌ای (>= ، > ، <= ، < ، <> و =) برای مقایسه دو رشته استفاده می‌کند. رابط کاربر برای این برنامه شامل چندین برچسب (Label)، دو کادر متن (TextBox) و یک دکمه فرمان (Command Button) به همراه یک لیست است. برای مقایسه دو رشته، ابتدا رشته‌ها را در درون دو کادر متن تایپ کرده و سپس بر روی دکمه Compare کلیک نمایید. (دراین حالت روال cmdCompare_Click خط ۴ فراخوانی می‌شود). برنامه سپس به مقایسه دو رشته می‌پردازد و نتیجه با استفاده از تابع StrComp و در پایین دکمه Compare به نمایش در می‌آید. نتایج به دست آمده در لیست پایین پنجره نمایش داده می‌شود.

```
1.  'ComParing String With StrComp,
2.  'Relational and equality opeators
```

```
3.  Option Explicit
4.  Private Sub cmdCompare_Click()
5.  Dim result As Integer
6.  result = StrComp(txtInput1.Text, txtInput2. Text)
7.  If result = -1 Then
8.  LblOutput.Caption = “The first String is less then the Second String”
9.  ElseIf result = 1 Then
10. LblOutput.Caption = “The first String is Grater the second String”
11. Else
12. LblOutput.Caption = “The Strings are equal”
13. End If
14. Call lstOutput.Clear
15. If txtInput1.Text = txtInput2.Text Then
16. Call lstOutput.AddItem(txtInput1.Text & “is equal to” & _
    TxtInput2.Text
17. End If
18. If txtInput1.Text < > txtInput2.Text Then
19. Call lstOutput.AddItem(txtInput1.Text & _
      “is not equal to “ & txtInput2.Text)
20. End If
21. If txtInput1.Text < txtInput2.text Then
22. Call lstOutput.AddItem(txtInput1.Text & _
      “is less Than “&  txtInput2.Text
23. End If
24. If txtInput1.Text > txtInput2.Text Then
25. Call lstOutput. AddItem(txtInput1.text &_
      “is geater Than “& txtInput2. Text)
26. End If
27. If txtInput1.Text <= txtInput2.text Then
28. Call lstOutput.AddItem(txtInput1.Text &_
      “is less Than or equal to” & txtInput2.Text)
29. End If
30. If txtInput1.text >= txtInput2.Text Then
31. Call lstOutput.AddItem(txtInput1.Text &_
      “is greater Than or equal to” & txtInput2.Text)
32. End If
33. End Sub
```



شکل ۲-۱٤)

در خط

result = StrComp(txtInput1.Text, txtInput2.Text)

از تابع StrComp برای مقایسه دو رشته موجود در کادر متن‌های txtInput1 و txtInput2 استفاده شده است. بر مبنای نتیجه result ساختار If/ElseIf/Else در خطوط ۷ تا ۱۱ یک رشته را که مشخص کننده نتیجه است به lblOutput.Caption تخصیص می‌دهد. اگر دو رشته با یکدیگر برابر باشند، تابع StrComp مقدار ۰ را برمی‌گرداند و اگر رشته اول کوچکتر از رشته دوم باشد، مقدار ۱- و اگر رشته اول بزرگتر از رشته دوم باشد، مقدار ۱ برگشت داده می‌شود.

مثال ۲-۴ : صفحه ورودی اطلاعات مشتری

یک فروشگاه ورزشی، می‌خواهد یک صفحه‌ی ورودی برای دریافت اطلاعات مشتریان خود ایجاد کند که اطلاعات موردنیاز زیر را دریافت نماید:

۱. نام
۲. سن
۳. شهر
۴. جنسیت

۵. نوع ورزش (دومیدانی، دوچرخه‌سواری، شنا، اسکی و اسکیت‌سواری)

۶. سطح فعالیت ورزشی (حرفه‌ای، متوسط به بالا، متوسط، مبتدی)

شکل ظاهری فرم به صورت زیر خواهد بود:

شکل ۲-۱۵)

بعد از کامل کردن فرم، اطلاعات مشتری در یک کادر پیغام نمایش داده خواهد شد.

مشخصه‌ها:

Form **frmCustomer**:
BorderStyle = 1 - Fixed Single
Caption = Customer Profile
CommandButton **cmdExit**:
Caption = E&xit
Frame **Frame3**:
Caption = City of Residence
FontName = MS Sans Serif
FontBold = True
FontSize = 9.75
FontItalic = True
ComboBox **cboCity**:
Sorted = True
Style = 1 - Simple Combo
CommandButton **cmdNew**:

Caption = &New Profile
CommandButton **cmdShow**:
Caption = &Show Profile
Frame **Frame4**:
Caption = Athletic Level
FontName = MS Sans Serif
FontBold = True
FontSize = 9.75
FontItalic = True
OptionButton **optLevel**:
Caption = Beginner
Index = 3
OptionButton **optLevel**:
Caption = Intermediate
Index = 2
Value = True
OptionButton **optLevel**:
Caption = Advanced
Index = 1
OptionButton **optLevel**:
Caption = Extreme
Index = 0
Frame **Frame1**:
Caption = Sex
FontName = MS Sans Serif
FontBold = True
FontSize = 9.75
FontItalic = True
OptionButton **optSex**:
Caption = Female
Index = 1
OptionButton **optSex**:
Caption = Male
Index = 0
Value = True
Frame **Frame2**:
Caption = Activities
FontName = MS Sans Serif
FontBold = True

FontSize = 9.75
FontItalic = True

CheckBox **chkAct**:
Caption = In-Line Skating
Index = 5

CheckBox **chkAct**:
Caption = Skiing
Index = 4

CheckBox **chkAct**:
Caption = Swimming
Index = 3

CheckBox **chkAct**:
Caption = Biking
Index = 2

CheckBox **chkAct**:
Caption = Walking
Index = 1

CheckBox **chkAct**:
Caption = Running
Index = 0

TextBox **txtName**:
FontName = MS Sans Serif
FontSize = 12

Label **Label1**:
Caption = Name
FontName = MS Sans Serif
FontBold = True
FontSize = 9.75
FontItalic = True

TextBox **txtAge**:
FontName = MS Sans Serif
FontSize = 12

Label **Label2**:
Caption = Age
FontName = MS Sans Serif
FontBold = True
FontSize = 9.75
FontItalic = True

کد این برنامه به صورت زیر خواهد بود:

```
Option Explicit
Dim Activity As String
Private Sub cmdExit_Click()
End
End Sub

Private Sub cmdNew_Click()
'Blank out name and reset check boxes
Dim I As Integer
txtName.Text = ""
txtAge.Text = ""
For I = 0 To 5
  chkAct(I).Value = vbUnchecked
Next I
End Sub

Private Sub cmdShow_Click()
Dim NoAct As Integer, I As Integer
Dim Msg As String, Pronoun As String
'Check to make sure name entered
If txtName.Text = "" Then
  MsgBox "The profile requires a name.", vbOKOnly + vbCritical, "No Name
Entered"
  Exit Sub
End If
'Check to make sure age entered
If txtAge.Text = "" Then
  MsgBox "The profile requires an age.", vbOKOnly + vbCritical, "No Age
Entered"
  Exit Sub
End If
'Put together customer profile message
Msg = txtName.Text + " is" + Str$(txtAge.Text) + " years old." + vbCr
If optSex(0).Value = True Then Pronoun = "He " Else Pronoun = "She "
Msg = Msg + Pronoun + "lives in " + cboCity.Text + "." + vbCr
Msg = Msg + Pronoun + "is a"
If optLevel(3).Value = False Then Msg = Msg + "n " Else Msg = Msg + " "
```

```
Msg = Msg + Activity + " level athlete." + vbCr
NoAct = 0
For I = 0 To 5
  If chkAct(I).Value = vbChecked Then NoAct = NoAct + 1
Next I
If NoAct > 0 Then
  Msg = Msg + "Activities include:" + vbCr
  For I = 0 To 5
    If chkAct(I).Value = vbChecked Then Msg = Msg + String$(10, 32) +
chkAct(I).Caption + vbCr
  Next I
Else
  Msg = Msg + vbCr
End If
MsgBox Msg, vbOKOnly, "Customer Profile"
End Sub

Private Sub Form_Load()
'Load combo box with potential city names
cboCity.AddItem "شیراز"
cboCity.Text = "شیراز"
cboCity.AddItem "اصفهان"
cboCity.AddItem "تبریز"
cboCity.AddItem "ارومیه"
cboCity.AddItem "اهواز"
cboCity.AddItem "یزد"
cboCity.AddItem "بوشهر"
cboCity.AddItem "زنجان"
cboCity.AddItem "رشت"
cboCity.AddItem "ساری"
cboCity.AddItem "گرگان"
```

```
cboCity.AddItem "مشهد"
cboCity.AddItem "زاهدان"
cboCity.AddItem "کرمان"
Activity = "intermediate"
End Sub

Private Sub optLevel_Click(Index As Integer)
‘Determine activity level
Select Case Index
Case 0
   Activity = "حرفه‌ای"
Case 1
  Activity = "متوسط به بالا"
Case 2
  Activity = "متوسط"
Case 3
  Activity = "مبتدی"
End Select
End Sub
Private Sub txtAge_KeyPress(KeyAscii As Integer)
'Only allow numbers for age
If (KeyAscii >= vbKey0 And KeyAscii <= vbKey9) Or KeyAscii = vbKeyBack
Then
  Exit Sub
Else
  KeyAscii = 0
End If
End Sub
```

مثال ۲-۵ :

در این مثال می‌خواهیم شهر مقصد، محل صندلی و غذای دلخـواه مـسافران هواپیمـا را انتخـاب کنیم. یک کادر لیست، دو کادر ترکیبی (کومبو)، سه برچسب و دو دکمه فرمان روی فرم قرار داده و ظاهر آن را مطابق شکل زیر تنظیم کنید:

شکل ۲-۱۶)

مشخصه‌ها:

Form1:

BorderStyle	1-Fixed Single
Caption	Flight Planner
Name	frmFlight

List1:

Name	lstCities
Sorted	True

Combo1:

Name	cboSeat
Style	2-Dropdown List

Combo2:

Name	cboMeal
Style	1-Simple
Text	[Blank]

(بعد از تنظیم این کادر ترکیبی، آن را تغییر اندازه دهید تا ۴ یا ۵ عنصر را در خود جای دهد.)

Label1:

Caption	Destination City

Label2:

Caption	Seat Location

Label3:

Caption	Meal Preference

Command1:

Caption	&Assign
Name	cmdAssign

Command2:

Caption	E&xit
Name	cmdExit

کد برنامه به صورت زیر خواهد بود:

```
Private Sub Form_Load()
‘Add city names to list box
lstCities.Clear
lstCities.AddItem "شیراز"
lstCities.AddItem "اصفهان"
lstCities.AddItem "تبریز"
lstCities.AddItem "ارومیه"
lstCities.AddItem "اهواز"
lstCities.AddItem "یزد"
lstCities.AddItem "بوشهر"
lstCities.AddItem "زنجان"
lstCities.AddItem "رشت"
lstCities.AddItem "ساری"
lstCities.AddItem "گرگان"
lstCities.AddItem "مشهد"
lstCities.AddItem "زاهدان"
lstCities.AddItem "کرمان"
lstCities.ListIndex = 0
‘Add seat types to first combo box
```

```
cboSeat.AddItem "ردیف اول"
cboSeat.AddItem "وسط"
cboSeat.AddItem "کنار پنجره"
cboSeat.ListIndex = 0
‘Add meal types to second combo box
cboMeal.AddItem "جوجه"
cboMeal.AddItem "برگ"
cboMeal.AddItem "سبزیجات"
cboMeal.AddItem "میوه"
cboMeal.Text = "فرقی نمی‌کند"
End Sub

Private Sub cmdAssign_Click()
‘Build message box that gives your assignment
Dim Message As String
Message = "Destination: " + lstCities.Text + vbCr
Message = Message + "Seat Location: " + cboSeat.Text + vbCr
Message = Message + "Meal: " + cboMeal.Text + vbCr
MsgBox Message, vbOKOnly + vbInformation, "Your Assignment"
End Sub
Private Sub cmdExit_Click()
End
End Sub
```

۲-٦ آرایه کنترلی

در ویژوال بیسیک می‌توان آرایه‌هایی از انواع داده‌ی مختلف ایجاد کرد. همچنین می‌توان آرایه‌ای از کنترل‌ها نیز ایجاد کرد. آرایه‌های کنترلی یک ویژگی ذاتی در ویژوال بیسیک هستند که توانایی و کارایی زبان برنامه‌نویسی را افزایش می‌دهند. می‌توان از یک روال رویداد استفاده کرد تا عمل مشترکی را روی همه‌ی عناصر آرایه‌ی کنترلی انجام دهد. همچنین می‌توان از آرایه‌های کنترلی برای اضافه کردن و حذف پویای کنترل‌ها و فرم‌ها در زمان اجرا استفاده کرد.

تمام کنترل‌های ذاتی را می‌توان به صورت آرایه‌های کنترلی به کار برد. همه‌ی این کنترل‌ها دارای مشخصه‌ی Index هستند که برای شناسایی کنترل خاصی در آرایه‌ی مورد استفاده قرار می‌گیرند.

۲-۶-۱ ایجاد آرایه کنترلی در زمان طراحی

اغلب آرایه‌های کنترلی که ایجاد خواهید کرد، در زمان طراحی ساخته خواهند شد. همزمان با اینکه کنترل‌ها را روی فرم اضافه می‌کنید، نیاز خواهید داشت تا بعضی از آنها را در آرایه‌های کنترلی گروه‌بندی کنید. مثال زیر، چگونگی انجام این کار را بیان می کند.

مثال ۲-۶ : ایجاد آرایه‌ی کنترلی از CommandButtons

۱- پروژه‌ی جدیدی را شروع کنید و نام فرم پیش‌فرض را به frmMain تغییر دهید.

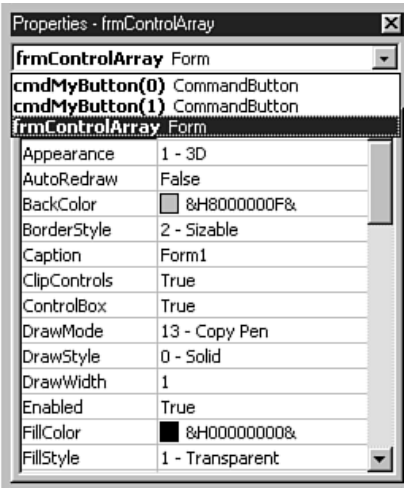
۲- یک دکمه‌ی فرمان به مرکز فرم frmMain اضافه کنید و نام آن را cmdMyButton قرار دهید. مشخصه‌ی Caption این دکمه‌ی فرمان را به Action تغییر دهید.

۳- دکمه‌ی فرمان را انتخاب کنید. از منوی Edit گزینه‌ی Copy را انتخاب کنید تا دکمه‌ی فرمان به حافظه‌ی Clipboard کپی شود.

۴- از منوی Edit گزینه‌ی Paste را انتخاب کنید. یک کادر محاوره‌ای ظاهر شده و از شما سوال می‌کند که آیا می‌خواهید یک آرایه‌ی کنترلی ایجاد کنید؟ روی Yes کلیک کنید تا آرایه‌ی کنترلی ایجاد شود (شکل ۲-۱۷).

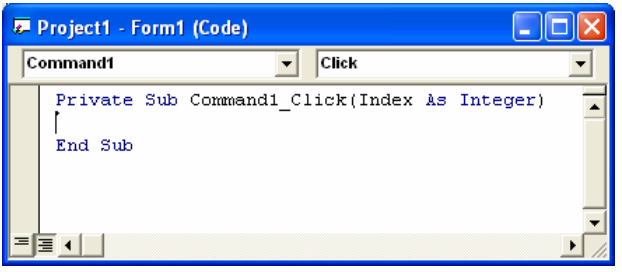
شکل ۲-۱۷) کادر محاوره‌ای تائید ایجاد آرایه‌ی کنترلی

بعد از ایجاد آرایه‌ی کنترلی، اگر به پنجره‌ی Properties رجوع کنید و لیست بازشوی Object را نمایش دهید. مشاهده خواهید کرد که دو دکمه‌ی فرمان با نام یکسان cmdMyButton وجود دارد که هر کدام دارای اندیس خاصی هستند (شکل ۲-۱۸).



شکل ۲–۱۸) هنگامی که کنترل بخشی از آرایه‌ی کنترلی باشد، برای دسترسی به آن باید همیشه به اندیس آن اشاره کرد.

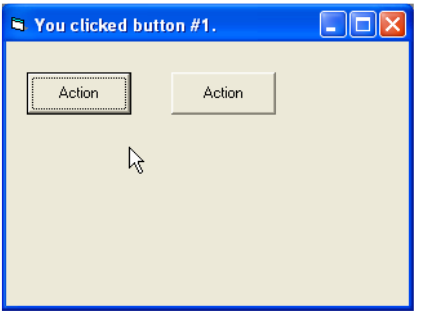
روی دکمه‌ی فرمان دابل کلیک کنید تا روال رویداد Click را مشاهده کنید. این روال رویداد دارای آرگومان Index خواهد بود (شکل ۲–۱۹). این آرگومان یک عدد صحیح است که اندیس کنترل را مشخص می کند. به دلیل اینکه همه‌ی کنترل‌های آرایه‌ی کنترلی نام یکسانی دارند، تفاوت بین آنها با مقدار Index در روال رویداد مشخص می‌شود (۰ اولین کنترل، ۱ دومین کنترل و ۲ سومین کنترل و الی آخر را مشخص می‌کند).



شکل ۲–۱۹) هر زمانی که رویدادی برای کنترلی از یک آرایه‌ی کنترلی رخ دهد، ویژوال بیسیک آرگومان Index را مقداردهی می کند.

کد زیر نشان می‌دهد که روی کدام دکمه‌ی فرمان از آرایه‌ی کنترلی کلیک شده است. شکل ۲-۲۰ اجرای این کد را نشان می‌دهد.

```
01 Private Sub cmdMyButton_Click(Index As Integer)
02 ` Change the form's caption to indicate which
03 ` button in the control array generated an event.
04 Me.Caption = "You clicked button #" & Index & "."
05 End Sub
```



شکل ۲-۲۰) بعد از کلیک کاربر روی دکمه‌ی سمت راست، عنوان فرم مطابق آن تغییر می‌یابد.

۲-۶-۲ افزودن عنصرها به آرایه‌ی کنترلی در زمان اجرا

ایجاد آرایه‌ی کنترلی در زمان طراحی، هنگامی مفید خواهد بود که تعداد کنترل‌های مورد نیاز برای آرایه را بدانید. ولی اگر تعداد کنترل‌هایی که در زمان اجرای برنامه نیاز خواهید داشت را نمی‌دانید، چه کاری باید انجام دهید؟ این مشکل را می‌توان با استفاده از دستور Load برای افزودن کنترل‌ها به آرایه‌ی کنترلی در زمان اجرا، حل کرد.

مثال ۲-۷)

۱- پروژه‌ی جدیدی را شروع کنید. نام فرم را به frmDArray تغییر دهید.

۲- یک دکمه‌ی فرمان به گوشه سمت چپ بالای فرم اضافه کنید و نام آن را cmdCtrlArray قرار دهید.

۳- در پنجره‌ی Properties، مقدار مشخصه‌ی Index دکمه‌ی فرمان را با ۰ مقداردهی کنید.

۴- مشخصه‌ی Caption دکمه‌ی فرمان را به Button #0 تغییر دهید.

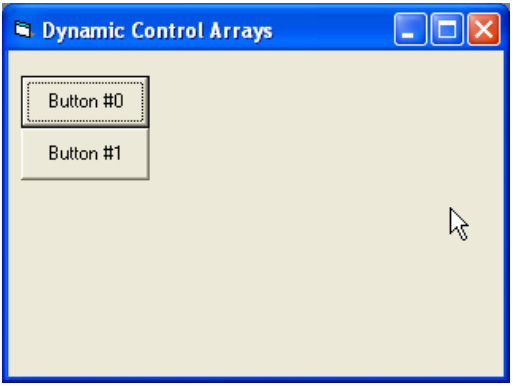
۵- کد زیر را به رویداد Form_Load اضافه کنید.

```
01 Private Sub Form_Load()
```

```
02 `Create a new command button
03 Load cmdCtrlArray(1)
04
05 `Move it directly underneath the old one
06 cmdCtrlArray(1).Left = cmdCtrlArray(0).Left
07 cmdCtrlArray(1).Top = cmdCtrlArray(0).Top + cmdCtrlArray(0).Height
08 cmdCtrlArray(1).Caption = "Button #1"
09
10 `Make the new button visible
11 cmdCtrlArray(1).Visible = True
12
13 End Sub
```

۶- کد را ذخیره کرده و اجرا کنید.

بعد از اجرای کد، برنامه یک دکمه‌ی فرمان جدیدی را ایجاد کرده و زیر اولین دکمه‌ی فرمان قرار می‌دهد (شکل ۲-۲۱).



شکل ۲-۲۱) استفاده از دستور Load دکمه‌ی فرمان دیگری را روی فرم قرار می‌دهد. بقیه‌ی کد محل دقیق کنترل را تعیین می‌کنند.

نکته:

تمام عناصر جدیدی که از آرایه‌ی کنترلی ایجاد می‌کنید دارای مشخصه‌ی Visible با مقدار False هستند. هنگامی که کنترل‌های جدیدی را در زمان اجرا ایجاد می‌کنید، فراموش نکنید خطی در کد

قرار دهید که مشخصه‌ی Visible را با True مقداردهی کند. در غیر این صورت نمی‌توانید کنترل را مشاهده کنید.

کنترل‌های جدیدی که ایجاد می‌کنید دقیقاً تکرار اولین عنصر آرایه‌ی کنترلی هستند. مقادیر همه‌ی مشخصه‌ها به جز Index و Visible یکسان هستند. بنابراین، هنگامی که کنترل جدیدی را ایجاد می‌کنید، این کنترل روی اولین کنترل آرایه قرار می‌گیرد و باید آن را به محل دیگری انتقال دهید که انجام این کار در زمان اجرا با تغییر مشخصه‌ی Left و Top ممکن است.

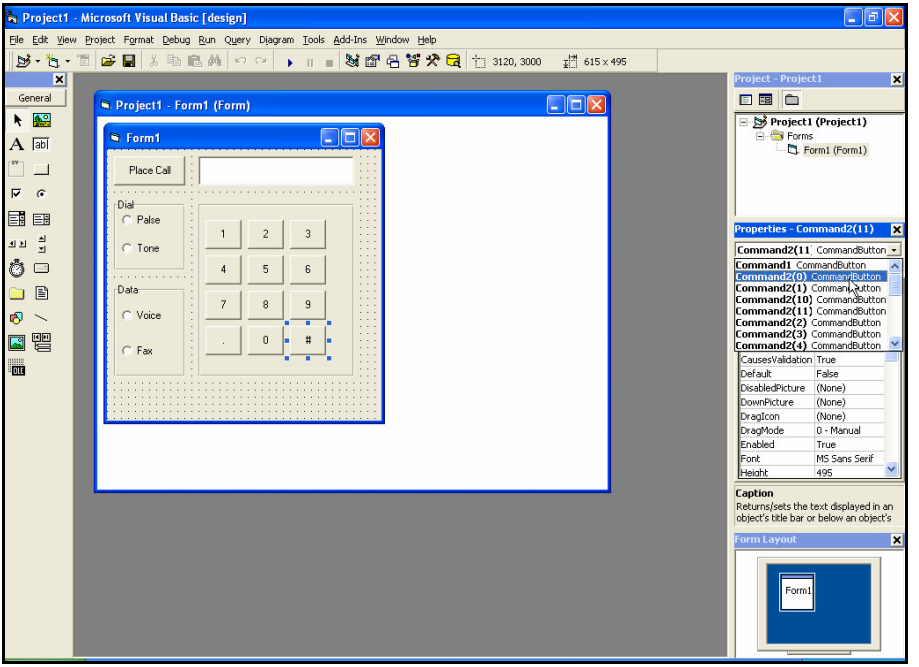
همانطور که در مثال قبل مشاهده کردید، مزیت استفاده از آرایه‌های کنترلی، توانایی داشتن مجری رویداد مشترک است. در این قسمت با استفاده از این ویژگی، برنامه‌ای را ایجاد می‌کنیم که به کاربران امکان وارد کردن چند عدد از طریق شماره‌گیر تلفن برای برقراری تماس را فراهم می‌کند. همچنین کاربران می‌توانند Pulse یا Tone بودن خط تلفن و ارسال نمابر یا تماس تلفنی را انتخاب کنند. این مثال به سادگی طراحی شده است تا چگونگی استفاده از آرایه‌های کنترلی در این برنامه را نشان دهد.

این برنامه یک آرایه‌ی کنترلی از دکمه‌های فرمان را برای مدیریت ورودی کاربر، به کار می‌برد. هر دکمه عنصری از آرایه‌ی کنترلی cmdNum است. در این پروژه اگر از آرایه‌ی کنترلی استفاده نکنید، باید ۱۲ روال رویداد برای برنامه بنویسید. هنگامی که از آرایه‌ی کنترلی استفاده می‌کنید، فقط یک روال رویداد خواهید داشت. برای تشخیص کنترلی که روال رویداد را فعال کرده است، از آرگومان Index استفاده می‌شود (شکل ۲-۲۲).

کد زیر، روال رویداد ()Click مربوط به آرایه‌ی کنترلی را نشان می‌دهد. این روال رویداد، از دستور Select Case برای ارایه‌ی پاسخ‌های مختلف که بستگی به دکمه‌ی کلیک شده دارد، استفاده می‌کند.

```
01 Private Sub cmdNum_Click(Index As Integer)
02 Dim strChar As String
03
04 `Find out which button was clicked by analyzing
05 the Index argument. Depending on which button
06 `you push, set the other string variable accordingly.
07 Select Case Index
```

```
08 `This button has the "*" character
09 Case 10
10 strChar = "*"
11 `This button has the "#" character
12 Case 11
13 strChar = "#"
14 `All the buttons have captions that match
15 `their index value.
16 Case Else
17 strChar = CStr(Index)
18 End Select
19
20 ` Add the new digit to the phone number.
21 lblNumber.Caption = lblNumber.Caption & strChar
22
23 End Sub
```



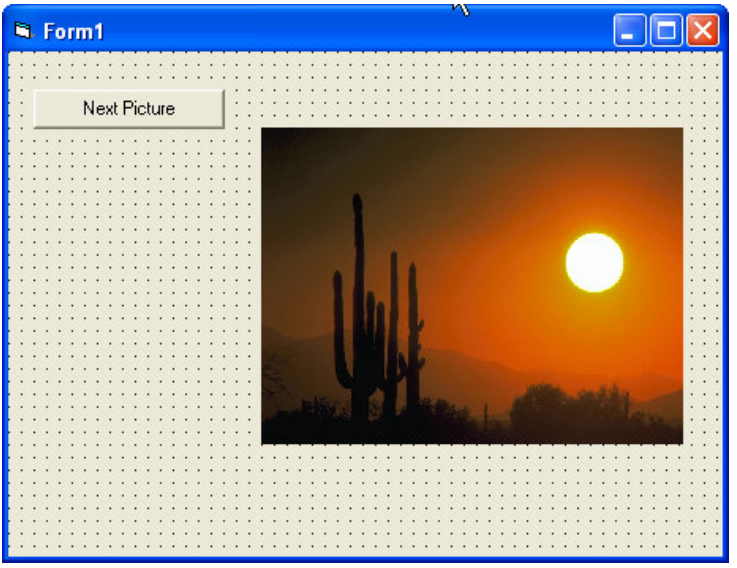
شکل ۲-۲۲) هر عنصری از آرایه‌ی کنترلی در پنجره‌ی Properties به همراه اندیس مربوطه لیست می‌شود.

اگر مقدار مشخصه‌ی Index دکمه‌های فرمان با مقدار مشخصه‌ی Caption آنها دارای یک مقدار باشند، می‌توانید کد فوق را به صورت ساده‌ی زیر بنویسید:

```
01 Private Sub cmdNum_Click(Index As Integer)
02 lblNumber.Caption = lblNumber.Caption & cmdNum(Index).Caption
03 End Sub
```

مثال ۲-۸)

۱- پروژه‌ای به صورت زیر ایجاد کنید (شکل ۲-۲۳).



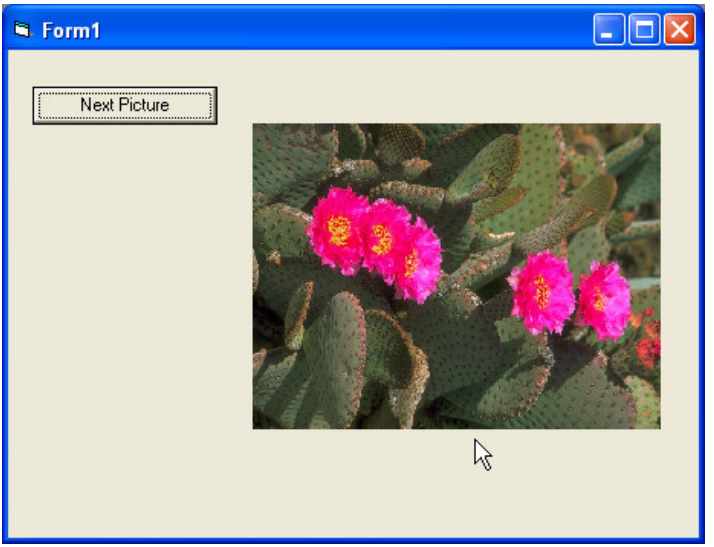
شکل ۲-۲۳)

۲- کد زیر را برای فرم وارد کنید:

```
'Image Array Example
Dim cnt
Private Sub Cmdpic_Click()
    img1(cnt).Visible = False
    cnt = cnt + 1
    If cnt > 4 Then cnt = 0
```

```
    img1(cnt).Visible = True
End Sub

Private Sub Form_Load()
    img1(0).Visible = True
    img1(1).Visible = False
    img1(2).Visible = False
    img1(3).Visible = False
    img1(4).Visible = False
    cnt = 0
End Sub
```

۳- پروژه را اجرا کنید (شکل ۲-۲۴).



شکل ۲-۲۴)

۲-۷ کاربرد کنترل‌های Scroll Bar

کنترل‌های نوار لغزان استاندارد (HscrollBar و VscrollBar) امکان تغییر داده‌ها یا جابه‌جایی در محدوده‌ای از مقادیر را با کلیک کردن روی دکمه‌های Up و Down یا با جابه‌جایی دکمه‌ی لغزان، فراهم می‌کند. کنترل‌های نوار لغزان دارای چند مشخصه‌ی خاص هستند که در جدول ۲-۲ با آنها آشنا خواهید شد.

جدول ۲-۲) مشخصه‌های خاص کنترل‌های HscrollBar و VScrollBar

مشخصه	شرح
Min	کمترین مقدار کنترل را هنگامی که دکمه‌ی لغزان در بالا یا سمت چپ نوار لغزان قرار دارد، تعیین می‌کند. مقدار پیش‌فرض، صفر است ولی اعداد منفی را نیز می‌توان به کار برد.
Max	بیشترین مقدار ممکن کنترل را هنگامی که دکمه‌ی لغزان در پایین یا سمت راست نوار لغزان قرار دارد، تعیین می‌کند. مقدار پیش‌فرض، ۳۲۷۶۷ است.
Value	موقعیت دکمه‌ی لغزان را نسبت به مشخصه‌های Min و Max تعیین می‌کند.
LargChange	مقدار تغییر مشخصه‌ی Value را هنگامی که کاربران بین دکمه‌ی لغزان و فلش کلیک می‌کنند، تعیین می‌کند.
SmallChange	مقدار تغییر مشخصه‌ی Value را هنگامی که کاربران روی فلش‌ها کلیک می‌کنند، تعیین می‌کند.

مثال۲-۹)

۱- پروژه‌ی جدیدی به نام SmpScrll.vbp ایجاد کنید. نام فرم را frmMain قرار دهید. مشخصه‌ی Caption فرم را به Simple Scroll Bars تغییر دهید.

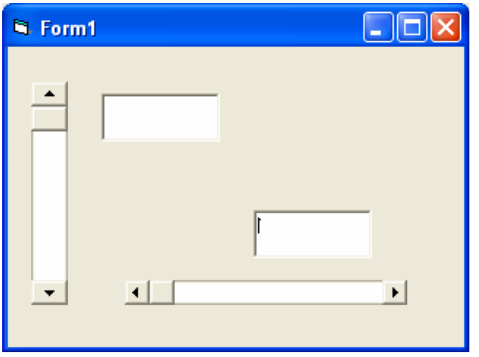
۲- کنترل‌های VscrollBar و HscrollBar را به فرم اضافه کنید. نام کنترل VscrollBar را به vscrNS و کنترل HscrollBar را به hscrWE تغییر دهید.

۳- دو کنترل کادر متن اضافه کنید. نام یکی را txtNS و دیگری را txtWE قرار دهید. مشخصه‌ی Text هر دو کنترل را یک رشته‌ی خالی قرار دهید (شکل ۲-۲۵).

۴- مشخصه‌های کنترل‌های VscrollBar و HscrollBar را مطابق جدول ۲-۳ مقداردهی کنید.

جدول ۲-۳) تنظیم مشخصه‌های کنترل‌های ScrollBar

مشخصه	مقدار
Min	0
Max	20
SmallChange	1
LargeChange	2



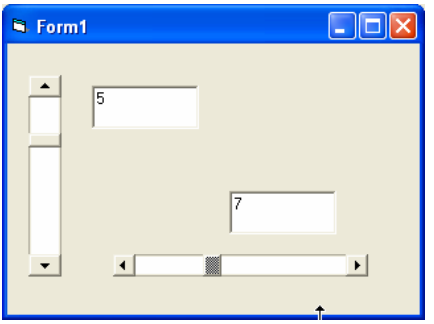
شکل ۲-۲۵) تنها تفاوت کنترل‌های HscrollBar و VscrollBar در جهت آنها است.

۵- کد زیر را در بخش General فرم frmMain اضافه کنید:

```
Private Sub hscrWE_Change()
   txtWE.Text = CStr(hscrWE.Value)
End Sub
Private Sub vscrNS_Change()
   txtNS.Text = CStr(vscrNS.Value)
End Sub
```

۶- پروژه را ذخیره کرده و اجرا کنید.

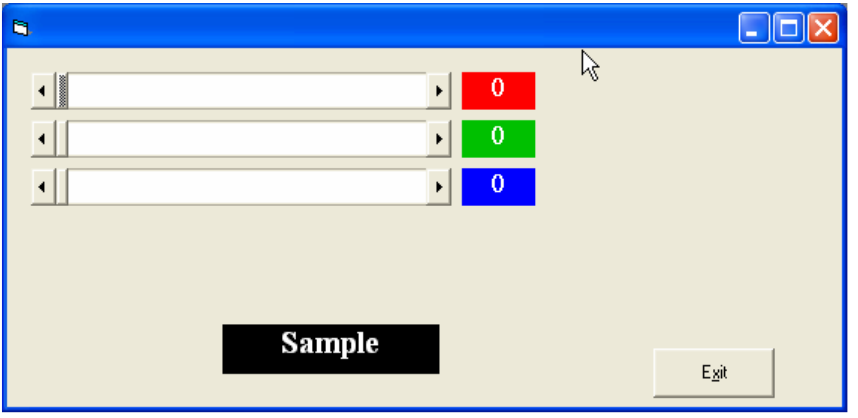
هنگامی که پروژه را اجرا کردید، با کلیک روی فلش‌های کنترل‌های HscrollBar و VscrollBar، مقدار کادر متن مربوطه یک واحد تغییر می‌کند (براساس مقدار مشخصه‌ی SmallChange). ولی اگر بین فلش و دکمه‌ی لغزان کلیک کنید، مقدار کادر متن مربوطه، دو واحد تغییر می‌کند (براساس مقدار مشخصه‌ی LargeChange) (شکل ۲-۲۶).



شکل ۲-۲۶) به دلیل اینکه مشخصه‌ی Max با ۲۰ مقداردهی شده است، هنگام جابه‌جایی نوار لغزان، مقادیر نشان داده شده در کادرهای متن تا ۲۰ می‌رسند.

مثال ۲-۱۰)

فرمی به صورت شکل ۲-۲۷ ایجاد کنید. با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی و سبز می‌توان رنگ برچسب را تغییر داد.



شکل ۲-۲۷)

۲- کد مربوطه را به فرم اضافه کنید:

```
Private Sub Command1_Click()
    End
End Sub

Private Sub Form_Load()
    Label1.BackColor = RGB(0, 0, 0)
```

```
End Sub

Private Sub HScroll1_Change()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll1_Scroll()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll2_Change()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub
Private Sub HScroll2_Scroll()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub
```

```
Private Sub HScroll3_Change()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll3_Scroll()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub
```

۸-۲ مرتب سازی عنصرهای آرایه‌ها

مرتب کردن داده‌ها یکی از مهمترین عملیات در برنامه‌های کاربردی است. الگوریتم‌های متعددی برای مرتب کردن n عنصر وجود دارد که از جهات مختلف قابل دسته‌بندی و مقایسه هستند. براساس خصوصیات این الگوریتم‌ها، هر کدام ممکن است در مسائل خاصی، عملکرد بهتری داشته باشند.

در عناصر مرتب شونده، بخشی از هر عنصر که مقایسه و مرتب سازی براساس آن بخش انجام می شود، «کلید» نام دارد. الگوریتم‌هایی که ترتیب عناصر با کلید مساوی را حفظ می‌کنند، «الگوریتم متعادل» نام دارند. در یک الگوریتم مرتب سازی همواره دو عمل «مقایسه» و «تعویض» انجام می‌گیرد و تعداد این اعمال مرتبه‌ی اجرایی الگوریتم را مشخص می‌کند.

الگوریتم‌ها به شیوه‌های مختلف نظیر تعویض مرتب عناصر، انتخاب عنصر مناسب و ساخت درخت، درج عناصر به صورت مرتب شده ادغام آرایه های کوچک و مرتب و مرتب‌سازی در یک مبنا، عمل مرتب‌سازی را انجام می‌دهند.

شرکت مخابرات برحسب نام خانوادگی افراد و در صورت تکراری بودن با نام، شماره تلفن‌ها را در کتابچه‌ی راهنمای تلفن مرتب می‌کند که با این روش، جستجوی شماره‌ی تلفن فرد مورد نظر آسان‌تر خواهد شد.

فرض کنید که A لیستی از n عدد باشد. **مرتب کردن** A عملیات آرایش مجدد عناصر A به صورت افزایشی است که در آن:

$A(0) < A(1) < A(2) < \ldots < A(N)$

به عنوان مثال، فرض کنید لیست اصلی A به صورت زیر باشد :

8,4,19,2,7,13,5,16

بعد از مرتب‌کردن، لیست به صورت زیر تبدیل خواهد شد :

2,4,5,7,8,13,16,19

فرض کنید که لیستی از اعداد A(N),…,A(1),A(0) در حافظه قرار دارد. الگوریتم مرتب‌سازی حبابی (Bubble Sort) به صورت زیر کار می‌کند :

مرحلهٔ اول : A(0) و A(1) را به ترتیب موردنظر مرتب کنید، طوری که A(0)<A(1) باشد. سپس A(1) و A(2) را مقایسه کرده و A(1)<A(2) قرار دهید. بعد A(2) و A(3) را مقایسه کرده و A(2)<A(3) قرار دهید. این کار تا زمانی که به مقایسهٔ A(N-1) و A(N) برسید ادامه داده و A(N-1)<A(N) قرار دهید.

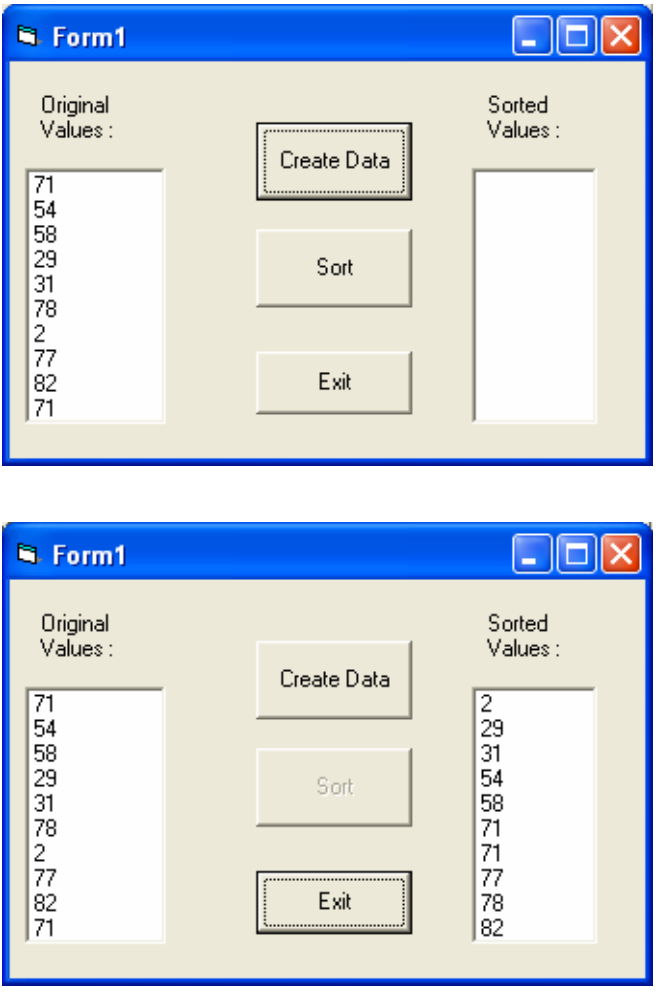
بعد از n – ۱ مرحله، لیست به صورت افزایشی مرتب خواهد شد.

مثال ۲-۱۱)

برنامه‌ی زیر، مقادیر آرایه mArray را به صورت صعودی مرتب می‌کند. تکنیکی که ما در برنامه استفاده کرده‌ایم، Bubble Sort (مرتب سازی حبابی) است. روال BubbleSort عمل مرتب‌سازی را انجام می‌دهد. برای استفاده بهینه از این روال (BubbleSort) آن را در مدول کد قرار داده‌ایم.

```
'Bubble Sort
Option Explicit
Option Base 1
Public mArray(10) As Integer
Private Sub cmdGenerate_Click()
```

```
    Dim x As Integer
    Call lstOriginal.Clear                  'Clear data
    Erase mArray                            'Clear array
    ‘Generate numbers
    For x = LBound(mArray) To UBound(mArray)
        mArray(x) = 1 + Int(100 * Rnd())
        Call lstOriginal.AddItem(mArray(x))
    Next x
    Call lstSorted.Clear            'Clear listBox
    cmdSort.Enabled = True      'Enable Sort button
End Sub

Private Sub cmdSort_Click()
    Dim pass As Integer, compare As Integer
    Dim hold As Integer
    Dim x As Integer
    Call lstSorted.Clear      ‘Clear ListBox
    For pass = 1 To (UBound(mArray) –1)
        For compare = 1 To (UBound(mArray) –1)
            If mArray(Compare) > mArray(Compare + 1) Then
            hold = mArray(Compare)
            mArray(Compare) = mArray(Compare + 1)
            mArray(Compare + 1) = hold
            End If
        Next compare
    Next Pass
    For x = 1 To UBound(mArray)
        Call lstSorted.AddItem(mArray(x))
    Next x
    cmdSort.Enabled = False
End Sub
Private Sub cmdExit_Click()
    End
End sub
```



شکل ۲-۲۸)

۲-۹ جستجوی خطی

در اغلب موارد می‌خواهید با مقادیر زیادی از داده‌ها که در آرایه‌ها ذخیره شده‌اند، کار کنید و ممکن است تعیین مقداری که در آرایه وجود دارد، ضروری باشد. عملیاتی که به منظور یافتن عضوی در آرایه صورت می‌گیرد به نام جستجو (Searching) معروف است. جستجوی خطی در مورد آرایه‌های کوچک یا آرایه‌های مرتب نشده خوب عمل می‌کند ولی در مقابل آرایه‌های بزرگ و

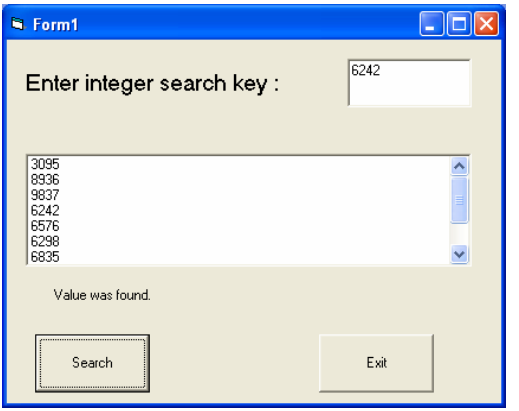
مرتب کارایی بهینه ندارد. اگر آرایه مرتب شده باشد استفاده از تکنیک دیگری که بهینه‌تر است، مناسب خواهد بود. این تکنیک، جستجوی دودویی (Binary Search) نام دارد.

در جستجوی خطی، هر عضو آرایه با مقداری که کلید جستجو (search key) نامیده می‌شود، مقایسه می‌شود. در این روش جستجو، به طور میانگین، برنامه کلید جستجو را با نصف اعضای آرایه مورد مقایسه قرار می‌دهد. برای تعیین این که مقدار مورد نظر در آرایه وجود ندارد، برنامه باید کلید جستجو را با تمام اعضای آرایه مقایسه کند.

مثال ۲-۱۲: روش جستجوی خطی)

```
'Demonstrating a linear search
Option Explicit
Option Base 1
Dim mArray(10) As Integer
Private Sub cmdSearch_Click()
    Dim searchKey As Integer            'Value to Search for
    Dim element = -1 As Integer         'Index of Value
    Dim x As Integer
    lblresult.Caption = " "
    searchKey = txtkey.Text
    For x = LBound(mArray) To UBound(mArray)
        If mArray(x) = searchKey Then
            element = x             'Return index
        End If
    Next x
    If element < > –1 Then
        lblresult.Caption = “Value was found.”
    Else
        lblresult.Caption = “Value was Not found.”
    End If
End Sub

Private Sub Form_Load()
    Dim x As Integer
    Call Randomize
    Call lstData.Clear
    lblResult.Caption = “ “
    ‘Genrate Some random data
```

```
    For x = LBound(mArray) To UBound(mArray)
        mArray(x) = 1+ Int(10000 * Rnd())
        Call lstData.AddItem(mArray(x))
    Next x
End sub

Private Sub cmdExit_Click()
    End
End sub
```



شکل ۲-۲۹) جستجوی خطی

۲-۱۰ جستجوی دودویی

فرض کنید که DATA یک آرایه‌ی خطی است که شامل اعداد با ترتیب صعودی یا حروف به ترتیب الفبایی است. یک الگوریتم جستجوی بهینه به نام **جستجوی دودویی** وجود دارد که می‌تواند برای پیدا کردن محل LOC عنصر ITEM در DATA مورد استفاده قرار گیرد.

به عنوان مثال، فرض کنید می‌خواهیم محل نامی را در دفترچه‌ی تلفن (یا واژه‌ای را در فرهنگ لغت) پیدا کنیم. واضح است که برای انجام چنین کاری از جستجوی خطی استفاده نمی‌کنیم و به جای آن وسط دفترچه را باز کرده و تعیین می‌کنیم که جستجو در کدام نیمه‌ی دفترچه است. سپس نیمه‌ی موردنظری را از وسط باز کرده و تعیین می‌کنیم که اسم موردنظر در کدام یک چهارم می‌باشد، این کار را روی یک چهارم موردنظر نیز انجام داده و تعیین می‌کنیم که اسم در کدام یک

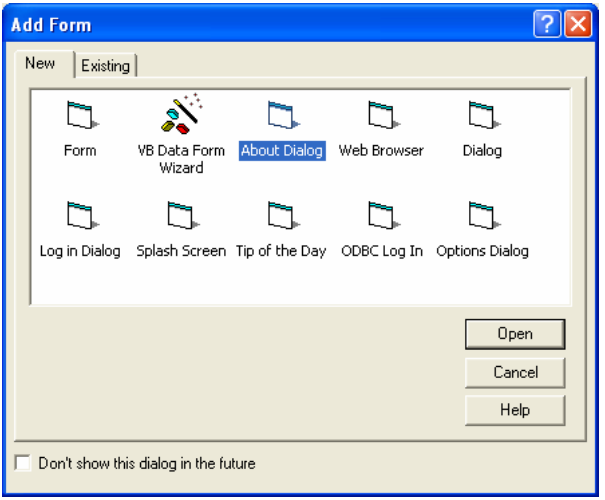
هشتم می‌باشد والی آخر. این کار را تا زمانی ادامه می‌دهیم که عنصر موردنظر را پیدا کنیم. این الگوریتم سریعتر از الگوریتم جستجوی خطی است، زیرا تعداد عناصر، در حال کم شدن است. مثال مربوط به جستجوی دودویی در فصل ششم آورده شده است.

فصل سوم

کاربرد فرم‌های آماده

ویژوال بیسیک دارای تعدادی فرم آماده (از پیش تعریف شده) است که استفاده از آنها سبب صرفه‌جویی در زمان کدنویسی می‌شود. از فرم‌های آماده ویژوال بیسیک، می‌توان برای دسترسی به داده‌ها، کادر About ، صفحه‌های الگو و کادرهای ورود، استفاده کرد.

از منوی Project گزینه‌ی Add Form را انتخاب کنید تا کاد محاوره‌ای مربوطه باز شود. در این کادر محاوره‌ای، انواع مختلف فرم‌های آماده وجود دارند. فرم مورد نظر را انتخاب کنید (شکل ۳-۱).



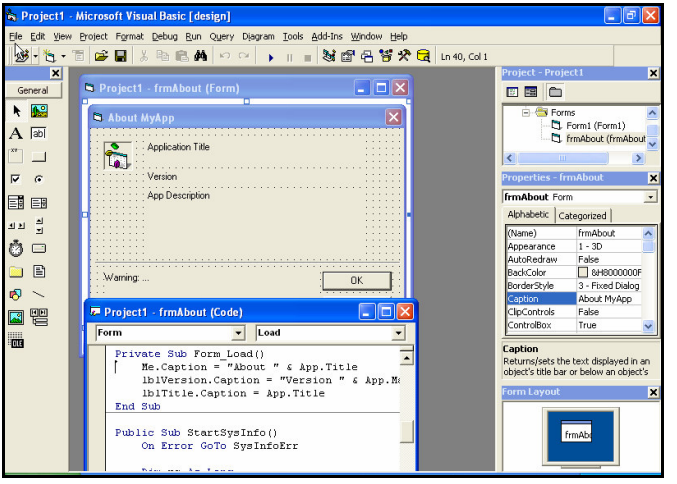
شکل ۳-۱) می‌توان فرم‌های آماده یا Form Wizard را در کادر محاوره‌ای Add Form انتخاب کرد.

در صورتی که می‌خواهید کادر محاوره‌ای About را به پروژه اضافه کنید، در کادر محاوره‌ای Add Form روی About Dialog کلیک کنید (شکل ۳-۲ یک فرم About Dialog) را نشان می‌دهد. نه تنها ویژوال بیسیک، فرم را به پروژه اضافه می‌کند، بلکه بخشی از کدهایی که وظایف

اصلی فرم را در برمی‌گیرند نیز درج می‌کند. About Dialog کدی را ارایه می‌کند که نـام برنامـه‌ی کاربردی و اطلاعات نسخه را همانطور که در کد زیر مشاهده می‌کنید، گزارش می کند:

```
Private sub Form_load( )
    Me.caption = “About” & App.Title
    LblVersion.caption=“Version”App.Major&“.”&App.Minor&“.”&App.Revisi
on
    LbLTitle.Caption = App.Title
End Sub
```

کادر محاوره‌ای About کل کد و فراخوانی API ویندوز کـه بـرای گـزارش اطلاعـات سیـستم کاربر مورد نیاز است را ارایه می‌کند. (در صورتی که می‌خواهید کد اطلاعات سیستم را مرور کنیـد، فرم About Dialog را به پروژه‌ی جدیـدی اضـافه کنیـد و کـد تحـت دکمـه‌ی System Info را مشاهده کنید.)



شکل ۳-۲) کادر محاوره‌ای About که به همراه ویژوال بیسیک ارایه می‌شود کدی که نام برنامه و اطلاعات نسخه و به همین ترتیب اطلاعات سیستم کاربر را گزارش می‌کند، ارایه می‌دهد.

۳-۱ کنترل Common Dialog

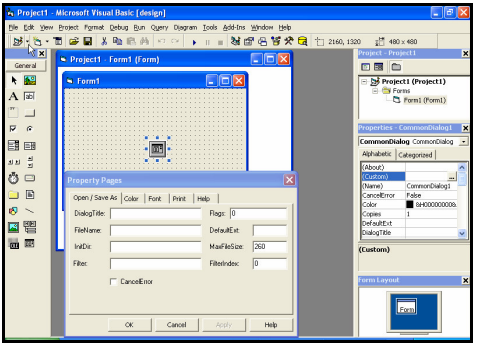
بعضی مواقع این احتمال وجود دارد که بخواهید برنامه‌ای بنویسید که کاربران بتوانند نام فایـل را تعیین کنند، قلم‌ها و رنگ‌ها را انتخاب کنند و چاپگر را کنترل نمایند. اگر چـه مـی‌توانیـد کادرهـای

محاوره‌ای ایجاد کنید که این وظایف را مدیریت کنند ولی نیاز بـه انجـام ایـن کـار نیـست. ویـژوال بیسیک، کنترل Common Dialog را ارایه می‌کند که به سادگی می‌توان کادرهای محـاوره‌ای آمـاده را برای بدست آوردن اطلاعات کاربر، نمایش داد. این کادرهای محاوره‌ای برای کاربران آشنا هستند و همان‌هایی هستند که خود ویندوز به کار می‌بـرد. بـا ا سـتفاده از کنتـرل Common Dialog، بـه چهار کاد محاوره‌ای ویندوز دسترسی خواهید داشت:

- **File**، به کاربر امکان انتخاب فایل بـرای بـاز شـدن یـا انتخـاب نـام فایـل بـرای ذخیـره‌ی اطلاعات را فراهم می‌کند.
- **Font**، امکان انتخاب قلم و تنظیم مشخصه‌های قلم مورد نظر را فراهم می‌کند.
- **Color**، امکان انتخاب رنگ یا ایجاد رنگ سفارشی برای استفاده در برنامه را ارایه می‌کند.
- **Print**، امکان انتخاب چاپگر و تنظیم چندین پارامتر چاپگر را فراهم می‌کند.

برای استفاده از کنترل Common Dialog، ابتدا باید آن را با انتخاب Microsoft Common Dialog Control 6.0 از کـادر محـاوره‌ای Components (انتخـاب Components از منـوی Project) به پروژه اضافه کنید. بعد از اضافه کردن کنترل Common Dialog به جعبـه ابـزار، روی کنترل کلیک و شبیه کنترل‌های دیگر، روی فرم ترسیم کنید. این کنترل، مانند یـک آیکـن روی فـرم ظاهر می شود و در زمان اجرا، قابل رؤیت نیـست. ولـی هنگـامی کـه کـد، Common Dialog را فراخوانی کند، کادر محاوره‌ای خاصی ظاهر می‌شود.

در ادامه هر کدام از کادرهای محاوره‌ای مربوط به این کنترل را شرح می‌دهـیم. بـرای هـر کـادر محاوره‌ای، باید مشخصه‌های کنترل را از طریق پنجره‌ی Properties یا کادر محـاوره‌ای Property Pages تنظیم کنید. کادر محاوره‌ای Property Pages دسترسی ساده به مشخـصه‌هـای مـورد نیـاز برای هر نوع کادر محاوره‌ای را ارایه می‌کند (شکل ۳–۳). برای دسترسی به این کادر محـاوره‌ای، در پنجره‌ی Properties روی سه نقطـه مقابـل مشخـصه‌ی Custom مربـوط بـه کنتـرل Common Dialog کلیک کنید.



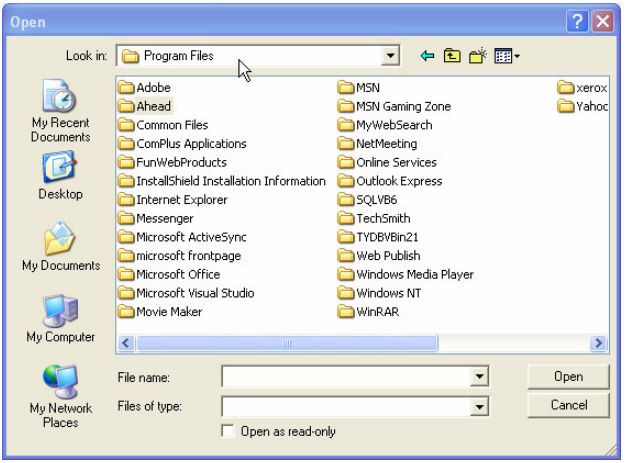
شکل ۳-۳) کنترل Common Dialog در زمان طراحی قابل رؤیت بوده و می‌تواند با استفاده از کنترل Property Pages پیکربندی شود.

۳-۲ بازیابی اطلاعات فایل با کادر محاوره‌ای File

کاربرد کلیدی کنترل Common Dialog ارایه‌ی نام فایل به وسیله‌ی کاربران به دو روش است: باز کردن و ذخیره‌ی فایل. حالت باز کردن فایل، به کاربر امکان تعیین نام فایل برای بازیابی و استفاده در برنامه را می‌دهد. حالت ذخیره‌ی فایل به کاربر امکان می‌دهد تا نامی را برای ذخیره‌ی اطلاعات تحت یک فایل، فراهم می‌کند. شکل ۳-۴، کادر محاوره‌ای با مؤلفه‌های اصلی را نشان می‌دهد.

کادرهای محاوره‌ای Open و Save شبیه هم هستند. برای باز کردن یک فایل موجود، از متد ShowOpen کنترل Common Dialog استفاده کنید. (این متد، کادر محاوره‌ای نشان داده شده در شکل ۳-۴ را نشان می‌دهد.) از این متد، به شکل زیر استفاده کنید:

```
CommonDlg1.ShowOpen
```



شکل ۳–٤) Common Dialog به کادرهای محاوره‌ای فایلی ویندوز دسترسی دارد.

برای باز کردن کادر محاوره‌ای Save As باید از متد Show Save استفاده کرد.

برای محدود کردن نوع فایل‌ها مثل فایل‌های متنی یا سند در کادرهای محاوره‌ای فایلی، از مشخصه‌ی Filter کنترل Common Dialog استفاده کنید. این مشخصه را می‌توان در زمان طراحی از طریق کادرمحاوره‌ای Proprty Pages یا در زمان اجرا با دستور زیر، تنظیم کرد:

ControlName.Filter = "*description* | *filtercond*"

- ControlName، نام تعیین شده برای کنترل Common Dialog است.
- Filter، نام مشخصه است.

نکته:

دقت کنید که قبل یا بعد از علامت پایپ (|) ، فضای خالی قرار ندهید. در صورتی که این کار را انجام دهید، ممکن است لیست فایل مورد نظر را بدست نیاورید.

- description، شرحی از انواع فایل‌هایی است که باید نشان داده شوند. مثال‌هایی از این شرح، عبارتند از: "Text Files"، "Word Documents" و "All Files" ، خط عمودی (علامت پایپ) باید وجود داشته باشد.
- Filtercond، فیلتر واقعی فایل‌هاست. این فیلتر را به صورت‌های *.txt، *.doc و *.* به کار ببرید.

اگر از دستور انتساب فوق استفاده کنید، باید فیلتر را داخل زوج کوتیش قرار دهید ولی در صورت استفاده از کاد محاوره‌ای Property Pages نیازی به این کار نیست.

می‌توان چندین فیلتر را با هم به کار برد ولی این فیلترهای مختلف، باید از علامت پایپ استفاده کرد. مثال زیر را در نظر بگیرید:

```
cdlgGetFile.Filter = "Text Documents ¦ *.txt ¦ All Files (*.*) ¦ *.*"
```

کاد ترکیبی File Type در شکل ۳–۴، کد فوق که به کنترل Comon Dialog اعمال شده است را نشان می‌دهد. بعد از تنظیم فیلتر، از مشخصه‌ی Filename کنترل Common Dialog برای بازیابی نام فایلی که کاربر انتخاب کرده است، استفاده کنید:

```
MyFileName$ = dlgGetFile.FileName
```

مثال ۳–۱ :

۱- پروژه‌ی جدیدی را شروع کنید.

۲- کنترل Common Dialog را همانطور که قبلاً شرح داده شد، به جعبه ابزار اضافه کنید.

۳- روی فرم کنترل‌های label، Common Button و Common Dailog اضافه کنید.

۴- کنترل برچسب را در بالای کنترل دکمه‌ی فرمان قرار دهید. هر دو کنترل را به گوشه چپ بالای فرم، منتقل کنید.

۵- روی دکمه‌ی فرمان، دابل کلیک کنید تا روال رویداد Command1_Click ایجاد شود.

۶- کد زیر را به این روال اضافه کنید:

```
01 CommonDialog1.Filter = "All Files (*.*)|*.*"
02 CommonDialog1.ShowOpen
03   If CommonDialog1.FileName <> "" Then
04      Label1.Caption = CommonDialog1.FileName
05   Else
06      Label1.Caption = "No file selected"
07   End If
```

عملکرد این کد به صورت زیراست:

- خط ۱، فیلتری را برای نمایش تمام فایل‌ها در یک پوشه، اعمال می‌کند.

- خط ۲، کادر محاوره‌ای Open را نمایش می‌دهد. کاربران می‌توانند فایلی را انتخاب کرده و روی Open کلیک کنند یا هیچ فایلی را انتخاب نکرده و روی Cancel کلیک کنند. براساس این انتخاب‌ها، ویژوال بیسیک مقداری را در مشخصه‌ی Filename کنترل Common Dialog قرار می‌دهد.
- خطوط ۳ تا ۷، مقدار مشخصه‌ی FileName را بررسی می‌کنند و در صورت انتخاب فایل، نام آن در مشخصه‌ی Caption برچسب قرار می گیرد و در غیر این صورت، پیغام No File Selected نوشته می شود.

۳-۳ انتخاب اطلاعات قلم با کادر محاوره‌ای Font

تنظیم کنترل Common Dialog برای نمایش کادر محاوره‌ای Font شبیه نمایش کادرهای محاوره‌ای فایلی است. در حقیقت می‌توان از کنترل Common Dialog برای مدیریت عملیات فایل، قلم، رنگ و چاپگر استفاده کرد.

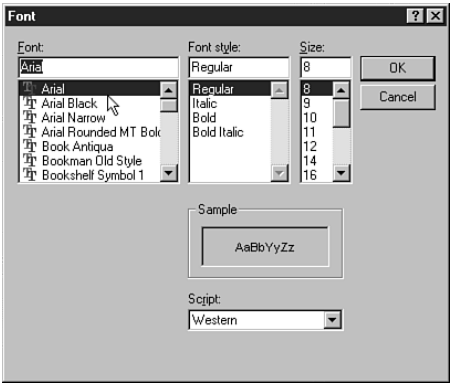
نکته:

اگر از کنترل Common Dialog برای انتخاب قلم‌ها استفاده می‌کنید و مقداری را برای مشخصه‌ی Flags تنظیم نکرده‌اید، پیغام خطایی را دریافت خواهید کرد مبنی بر اینکه هیچ قلمی نصب نشده است.

اولین گام در استفاده از کنترل Common Dialog برای مدیریت قلم‌ها، مقداردهی مشخصه‌ی Flags است. این مشخصه به کنترل Common Dialog بیان می‌کند که آیا می‌خواهید قلم‌های صفحه نمایش چاپگر یا هر دو را نمایش دهید. مشخصه‌ی Flags می‌تواند یکی از سه ثابت زیر را داشته باشد:

مجموعه قلم	ثابت	مقدار
قلم‌های صفحه نمایش	cdlCFScreenFonts	1
قلم‌های چاپگر	cdlCFPrinterFonts	2
هر دو مجموعه	cdCFBoth	3

شکل ۳-۵، کادر محاوره‌ای Font که فقط شامل قلم‌های صفحه نمایش است را نشان می‌دهد.



شکل ۳–۵) کادر محاوره‌ای Font امکان انتخاب قلم‌ها به کاربر را می‌دهد.

می‌توان مقدار مشخصه‌ی Flags رااز محیط طراحی با استفاده از کادر محاوره‌ای Property Pages یا از درون برنامه با دستور انتساب، تعیین کرد. بعد از تعیین مشخصه‌ی Flags، می‌توان کادر محاوره‌ای Font را با متد ShowFont اجرا کرد که دارای شکل کلی مانند متد Show Open است که قبلاً شرح داده شده است.

اطلاعات قلم‌های انتخاب شده در مشخصه‌های کنترل قرار داده می‌شود. جدول ۳–۱، مشخصه‌های کنترل و صفات قلم را نشان می‌دهد.

جدول ۳–۱) مشخصه‌های کنترل که صفات قلم را ذخیره می‌کنند.

مشخصه	صفت
FontName	نام قلم
Font size	اندازه قلم برحسب پوینت
Font Bold	آیا سیاه بودن قلم انتخاب شده است؟
Font Italic	آیا کج بودن قلم انتخاب شده است؟
Font Underline	آیا قلم زیر خط‌دار شده است؟
Font Strikethru	آیا روی قلم خط کشیده شده است؟

اطلاعات قلم می‌تواند برای تعیین قلم هر نوع کنترلی در برنامه یا حتی شیء Printer مورد استفاده قرار گیرد. کد زیر نشان می‌دهد که اطلاعات قلم چگونه بدست آمده و برای تغییر قلم‌ها در کادر متن txtsample استفاده می‌شود.

```
`cdlGetFont is the name of a common dialog
cdlGetFont.ShowFont
txtSample.FontName = cdlGetFont.FontName
txtSample.FontSize = cdlGetFont.FontSize
txtSample.FontBold = cdlGetFont.FontBold
txtSample.FontItalic = cdlGetFont.FontItalic
txtSample.FontUnderline = cdlGetFont.FontUnderline
txtSample.FontStrikethru = cdlGetFont.FontStrikethru
```

۳-٤ انتخاب رنگ‌ها با کادرمحاوره‌ای Color

کادر محاوره‌ای Color امکان انتخاب رنگ‌هایی برای رویه یا زمینه‌ی فرم‌ها یا کنترل‌ها فراهم می‌کند (شکل ۳–۶). کاربران می‌توانند یک رنگ استاندارد را انتخاب کرده یا رنگ سفارشی را ایجاد و انتخاب کنند.

شکل ۳–٦) کادرمحاوره‌ای Color امکان انتخاب رنگی برای استفاده در برنامه را به کاربران می‌دهد.

برای فعال کردن کادر محاوره‌ای Color در کنترل Common Dialog، مشخصه‌ی Flags آن را با ثابت cdLccRGBInit مقداردهی و سپس متد Show Color را فراخوانی کنید.

هنگامی که کاربران رنگی را از کادر محاوره‌ای انتخاب می‌کنند، مقدار رنگ در مشخصه‌ی Color کنترل، ذخیره می‌شود.

کد زیر، چگونگی تغییر رنگ زمینه‌ی فرم را نشان می‌دهد:

```
Common Dlg1.Flags = cdLccRGBInit
CommonDLG1.ShowColor
MyForm.BackColor = CommonDlg1.Color
```

۳-۵ تنظیم گزینه‌های چاپگر با کادرمحاوره‌ای Print

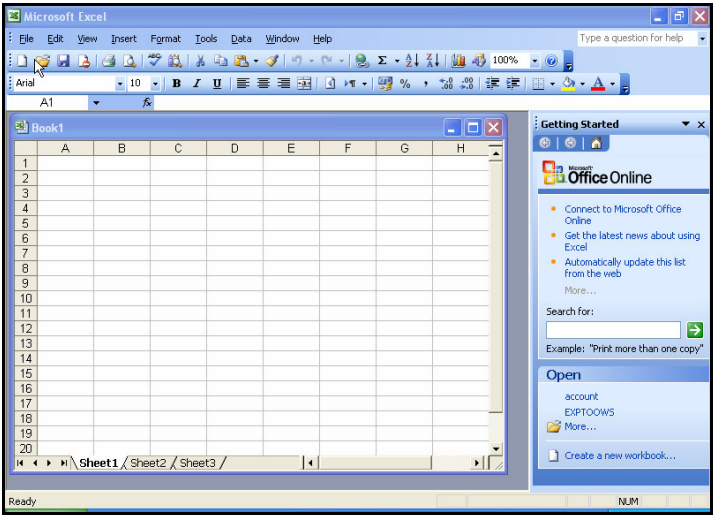
کادر محاوره‌ای Print به کاربران امکان انتخاب چاپگر و تعین گزینه‌ها برای فرآیند چاپ را ارایه می‌کند. این گزینه‌ها شامل تعیین کل صفحات، محدوده‌ای از صفحات یا بخش انتخاب شده را برای چاپ است. همچنین گزینه‌هایی برای تعیین تعداد کپی‌های چاپی و چاپ در یک فایل نیز وجود دارد.

برای اجرای کادر محاوره‌ای Print، کاربران می‌توانند چاپگر را از لیست Name انتخاب کنند که شامل تمام چاپگرهای نصب شده در ویندوز است. زیر لیست Name خط Status قرار دارد که وضعیت فعلی چاپگر انتخاب شده را بیان می‌کند.

کادر محاوره‌ای Print اطلاعات دریافتی از کاربران را در مشخصه‌های کاد محاوره‌ای برمی‌گرداند. مشخصه‌های FromPage و To Page، صفحات شروع و پایان خروجی چاپی را بیان می‌کند. مشخصه‌ی Copies تعداد کپی‌هایی که کاربران می‌خواهند چاپ کنند را نشان می‌دهد.

۳-٦ ایجاد برنامه‌ی کاربردی چند سندی ساده

MDI سرنام کلمات Multiple Document Interface (رابط چند سندی) است. یک برنامه‌ی MDI (چند سندی)، برنامه‌ای است که همه‌ی پنجره‌های آن در داخل یک پنجره باز می‌شوند. Microsoft Word و Microsoft Excel مثال‌هایی ازبرنامه‌ی کاربردی چند سندی هستند.



شکل ۳–۷) Microsoft Excel مثالی از یک برنامه‌ی کاربردی MDI است.

ویژوال بیسیک یک شیء MDI به نام MDIForm تعریف کرده است. یک برنامه‌ی کاربردی در VB می‌تواند **فقط** شامل یک شیء MDIForm باشد ولی می‌تواند چندین فرم دیگر داشته باشد که بعضی از آنها فرزند شیء MDIForm بوده و بعضی دیگر پنجره‌های مستقل هستند. پنجره‌های فرزند یک MDIForm منوی خاص خودشان را نمی‌توانند داشته باشند و به جای آن، پنجره‌های فرزند به وسیله‌ی منوی فرم MDI پدرشان کنترل می‌شوند. اگر منویی را به فرم فرزند MDI اضافه کنید، در زمان اجرا داخل فرم فرزند MDI قابل رؤیت نخواهد بود. منوی فرزند فعال در پنجره‌ی پدر در محلی از منوی پدر، ظاهر می‌شود.

مثال ۳– ۲)

۱– پروژه‌ی جدیدی را شروع کرده و نام آن را SimplMDI.VBP قرار دهید.

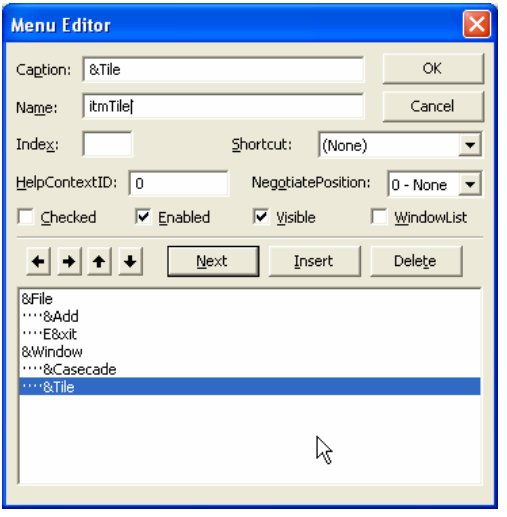
۲– فرم پیش فرض پروژه را به frmChild تغییر نام دهید. فایل فرم را با همین نام ذخیره کنید.

۳– از منوی Project گزینه‌ی Add MDI Form را انتخاب کنید. تا یک فرم MDI به پروژه اضافه شود. فرم MDI را به mdiMain تغییر نام و با همین نام ذخیره کنید.

۴- مشخصه‌ی MDIchild فرم frmChild را با true مقداردهی کنید تا به عنوان فرم فرزند mdiMain محسوب شود.

۵- کلیدهای Ctrl+E را فشار دهید تا Menu Editor باز شود.

۶- مانند شکل ۳–۸، منویی را برای فرم mdiMain ایجاد کنید. مقادیر مشخصه‌ی Name و Caption منو و گزینه‌های منو را مطابق جدول ۳–۲ تعیین کنید. مطمئن شوید که کادر علامت WindowList برای mnuWindow انتخاب شده باشد.

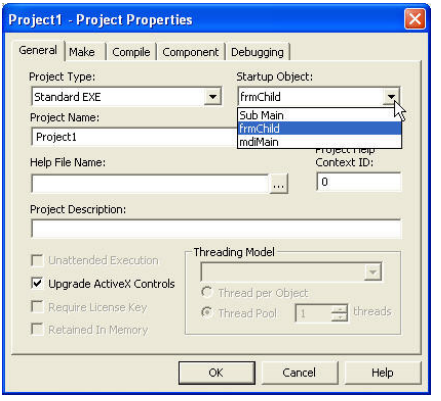


شکل ۳–۸) منوهایی را با استفاده از Menu Editor برای یک برنامه‌ی کاربردی MDI ایجاد کنید.

جدول ۳–۲) مشخصات منوی SimplMDI

Caption	Name	Indent
&File	mnuFile	0
&Add	itmAdd	1
E&xit	itmExit	1
&Window	mnuWindow	0
&CasCade	itmCasCade	1
&Tile	itmTile	1

۷- از منـوی Project گزینـه‌ی SimplMDI Properties را انتخـاب کنیـد. در صـفحه‌ی General کـادر محـاوره‌ای Project Properties، از لیـست بازشـوی Startup Object گزینه‌ی mdiMain را انتخاب کنید (شکل ۳–۹).

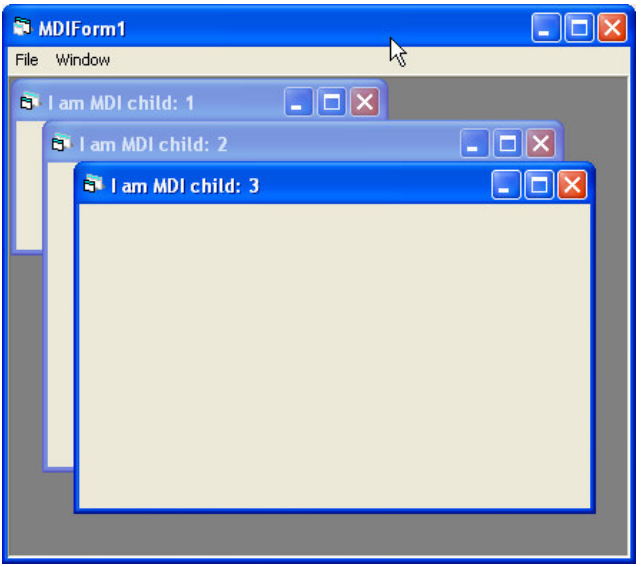


شکل ۳-۹) فرم MDI را به عنوان آغازین، تعیین کنید.

۸- کد زیر را برای گزینه‌های منوی برنامه‌ی کاربردی، بنویسید:

```
01 Private Sub itmAdd_Click()
02 Dim NewForm As New frmChild ` Declare another new form
03 Dim FormNum%
04 `Add it to the new form
05 Load NewForm
06 `Get a number for the new form, less the MDI parent
07 FormNum% = Forms.Count - 1
08 `Set its caption
09 NewForm.Caption = "I am MDI child: " + CStr(FormNum%)
10 End Sub
```

۹- برنامه را اجرا کنید.



شکل ۳-۱۰) پروژه‌ی SimplMDI فرم‌های فرزند داخل فرم MDI را نشان می‌دهد.

روال رویداد itmAdd_click() به طور پویا فرم جدیدی را با استفاده از عملگر New ایجاد می‌کند. Caption فرم جدید دارای یک شماره است (FormNum%) که به انتهای رشته ی مربوطه اضافه می‌شود تا نشان دهد که چندمین فرم فرزند است. این شماره از مشخصه‌ی Forms.Count بدست می‌آید و برای اینکه تعداد فرم‌های فرزند محاسبه شوند، فرم MDI باید از شمارش، کسر شود. به همین دلیل، شماره‌ی فرم از تفریق تعداد فرم‌ها و عدد ۱ بدست می‌آید.

در این برنامه‌ی کاربردی، از متد Arrange شیء MDIForm استفاده شده است. این متد، به طور خودکار پنجره‌های فرزند را داخل پنجره‌ی MDI قرار می‌دهد. متد Arrange دارای یک آرگومان است که می‌تواند یکی از مقادیر جدول ۳-۳ را داشته باشد.

جدول ۳-۳) تنظیمات آرایش پنجره‌ها

ثابت	مقدار	شرح
vbCasCade	0	تمام فرم‌های فرزند MDI که به حداقل تبدیل نشده‌اند را پشت سرهم قرار می‌دهد.

vbTileHorizontal	1	تمام فرم‌های فرزند MDI که به حداقل تبدیل نشده‌اند را به صورت کاشی‌های افقی می‌چیند.
vbTileVertical	2	تمام فرم‌های فرزند MDI که به حداقل تبدیل نشده‌اند را به صورت کاشی‌های عمودی می‌چیند.
vbArrangeIcons	3	آیکن‌های مربوط به MDI حداقل شده را مرتب می‌کند.

۳-۷ مشخصه‌های Appearance

کاربران به فرم‌های سه‌بعدی عکس‌العمل مثبتی دارند. با تغییر مشخصه‌ی Appearance فرم MDI به 1-3D، می‌توان فرم‌های سه‌بعدی ایجاد کرد.

۳-۷-۱ مشخصه‌ی AutoShowChildren

نکته:

اگر فرمی دارید که داده‌ها را در درون خودش، بارگذاری می‌کند، مشخصه‌ی AutoShowChildren را با False مقداردهی کنید و از متد Show استفاده کنید. ابتدا داده‌ها را در فرم بارگذاری کنید. ثانیاً مطمئن شوید که تمام داده‌ها به درستی بارگذاری شده‌اند. در پایان، فرم را نشان دهید. اگر فرم به تجمع داده‌ها وابستگی دارد، این عمل بالاترین قابلیت را ارایه می‌کند.

به طور پیش فرض، مشخصه‌ی AutoShowChildren دارای مقدار Flase است. به همین دلیل، اگر فراموش کنید که در کد از متد Show برای نمایش فرم بعد از بارگذاری آن استفاده کنید، کاربران برای نمایش فرم، دچار مشکل خواهند شد.

اگر AutoShowChildren را با True مقداردهی کنید، هنگام استفاده از دستور Load فرم فرزند MDI قابل رؤیت خواهد بود. انجام این کار، سبب صرفه‌جویی در زمان برنامه‌نویسی شده و توانایی کد برنامه را افزایش می‌دهد.

برنامه‌های کاربردی MDI دارای ظاهر یکسان و هماهنگ بوده و کار کردن با آنها نسبت به برنامه‌های SDI (تک سندی) ساده‌تر خواهد بود. در برنامه‌های SDI تمام پنجره‌ها مستقل از همدیگر هستند. اغلب برنامه‌نویسان ترجیح می‌دهند که از برنامه‌های کاربردی MDI استفاده کنند.

فصل چهارم

زمان و تاریخ

۴-۱ آشنایی با Serial Time

ویژوال بیسیک از نوع داده‌ی Date استفاده می‌کند که روز را به عنوان واحد اصلی زمان به کار می‌برد. بنابراین، یک ساعت، $\frac{1}{24}$ روز است و یک ثانیه $\frac{1}{86400}$ روز است. یک هفته را به صورت ۷ نمایش می‌دهیم، زیرا یک هفته شامل هفت روز است. نوع داده‌ی Date تاریخ را مطابق با قالب زمان کامپیوتر، نمایش می‌دهد: براساس قالب ۱۲ ساعتی یا ۲۴ ساعتی.

در تقویم میلادی اولین روز، اول ژانویه سال ۰۰۰۰ است ولی ویژوال بیسیک، اولین روز را ۳۱ دسامبر سال ۱۸۹۹ فرض می‌کند. بنابراین، دومین روز، اول ژانویه سال ۱۹۰۰ است و ۱۲ ژوئن سال ۱۹۶۸، روز ۲۵۰۰۱ است.

تاریخ‌های قبل از ۳۱ دسامبر ۱۸۹۹، با اعداد منفی نمایش داده می شوند. به عنوان مثال، ۴ جولای سال ۱۷۷۶، روز ۴۵۱۰۳- است یعنی ۴۵۱۰۳ روز قبل از ۳۰ دسامبر ۱۸۹۹.

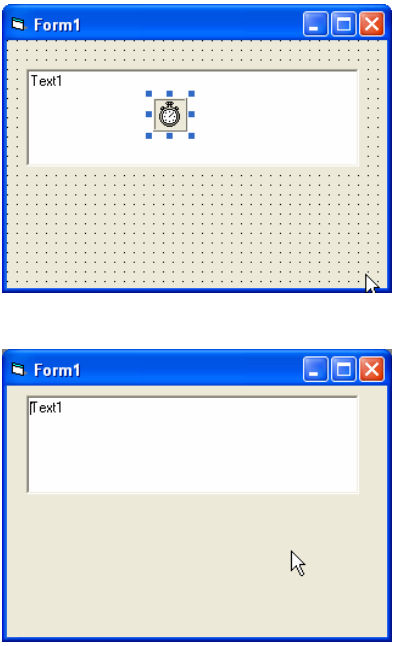
می‌توان مقادیر تاریخ را با قرار دادن بین دو علامت عددی (#)، در متغیرهای Date ذخیره کرد. به عنوان مثال:

```
Dim MyDate As Date
MyDate = #May 15, 1998#
```

نوع داده‌ی Date می‌تواند به هر نوع داده‌ی دیگری تبدیل شود. به عنوان مثال، 2 P.M May 15, 1998 به صورت یک عدد اعشاری 35390.58333 از نوع Double خواهد بود. هر چیزی که در سمت چپ نقطه‌ی اعشار است، نمایانگر روز بوده و هر چیزی که در سمت راست نقطه‌ی اعشار باشد، نمایانگر زمان یا بخشی از روز است: ساعت، دقیقه، ثانیه و میلی‌ثانیه. توجه داشته باشید که مقادیر اعشاری کوچکتر از ۱ می‌توانند فقط برای تعیین زمان، به کار روند. به عنوان مثال، 0.12345 زمان 2:57:46 a.m را ارایه می‌کند.

۴-۲ آشنایی با کنترل Timer

در ویژوال بیسیک، کنترل Timer امکان پی‌گیری زمان را فراهم می‌کند. فرض کنید کـه Timer ساعتی است که یک رویداد قابل برنامه‌نویسی را در یک فاصله‌ی زمانی که تعیین کرده‌اید، اجرا مـی کنـد (شـکل ۴-۱). رویـداد Timer فراخـوانی شـده و روال رویـدادی کـه بـرای TimerName_Timer() برنامه‌نویسی کرده‌اید، اجـرا مـی‌شـود. TimerName مقـدار مشخـصه‌ی کنترل Timer است.



شکل ۴-۱) کنترل Timer در زمان اجرا قابل رؤیت نخواهد بود.

می‌توان فاصله‌ی زمانی که رویداد Timer اجرا می‌شود را بـا تعیـین مقـداری بـرای مشخـصه‌ی Interval کنترل Timer تنظیم کرد. واحد اندازه‌گیری مشخصه‌ی Interval میلی‌ثانیه است، بنابراین اگر می‌خواهید رویداد Timer در نیم‌ثانیه اجرا شود، کد زیر را بنویسید:

```
Timer1.Interval = 500
```

نکته:

حداکثر مقدار مشخصه‌ی Interval، می‌تواند ۶۵،۵۳۵ باشد. این بدین معنی است که حداکثر فاصله‌ی زمانی که می‌توانید تنظیم کنید، ۶۵/۵ ثانیه است. برای استفاده از فاصله‌های زمانی طولانی‌تر مثل ۱۰ دقیقه، باید ۱۰ فاصله‌ی زمانی ۶۰۰۰۰ میلی‌ثانیه را قرار دهید.

فاصله‌ی زمانی می‌تواند خیلی کوچک مثل هزارم ثانیه یا خیلی بزرگ باشد. کوتاهترین فاصله‌ی زمانی، ۵۵ میلی‌ثانیه است، زیرا ساعت سیستم در هر ثانیه فقط ۱۸ بار پالس تولید می‌کند.

به دلیل اینکه کنترل Timer رویداد Timer را اجرا می‌کند، دارای مشخصه‌های زیادی نیست (جدول ۴-۱).

جدول ۴-۱) مشخصه‌های کنترل Timer

مشخصه	شرح
Name	نام کنترل. اگر یک کنترل Timer داشته باشید، پیش فرض Timer1 است و الی آخر.
Enabled	کنترل Timer را فعال یا غیرفعال می‌کند. پیش فرض، True یا On است.
Index	محل یک کنترل Timer خاص در داخل آرایه‌ی کنترل را گزارش می‌کند.
Interval	فاصله‌ی زمانی که رویداد Timer اجرا خواهد شد را تعیین می‌کند (برحسب میلی‌ثانیه).
Left	موقعیت گوشه‌ی سمت چپ کنترل Timer را تعیین می‌کند.
Tag	مشخصه‌ی Tag شبیه یک متغیر درونی در کنترل است که داده‌های مورد نیاز را ذخیره کرده و تا پایان برنامه به صورت پایدار نگهداری می‌کند.
Top	موقعیت گوشه‌ی بالای کنترل Timer را تعیین می‌کند.

نکته:

مشخصه‌های Left و Top نامربوط هستند، زیرا کنترل Timer در زمان اجرا روی فرم نمایش داده نمی‌شوند.

۴-۳ کاربرد توابع Time، Date و Now

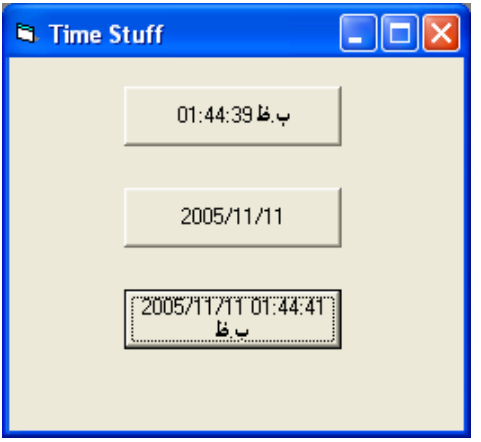
اگر چه می‌توان برای رویداد Timer برنامه‌ای نوشت، ولی نمی‌توان زمان اجرا را با این کنترل، تنظیم کرد. برای اینکه Timer زمان روز را گزارش کند، باید ساعت سیستم را بررسی کند.

هنگامی که به ساعت سیستم دسترسی دارید، از تابع Timer برای بدست آوردن زمان سیستم و از تابع Date برای بدست آوردن تاریخ سیستم استفاده کنید. در صورتی که می‌خواهید هم زمان و هم تاریخ را از ساعت سیستم بدست آورید، از تابع Now استفاده کنید.

مثال ۴-۱)

برای مشاهده‌ی چگونگی عملکرد این توابع، پروژه‌ای را ایجاد کرده و سه دکمه‌ی فرمان به فرم اضافه کنید. این دکمه‌ها را به ترتیب cmdTime، cmdDate و cmdNow نامگذاری کنید و کد زیر را برای روال رویداد هر دکمه اضافه کنید. شکل ۴-۲، داده‌هایی که هر تابع برمی‌گرداند را نشان می‌دهد.

```
Private Sub cmdTime_Click()
`Get the time and convert it to a string
cmdTime.Caption = CStr(Time)
End Sub

Private Sub cmdDate_Click()
`Get the date and convert it to a string
cmdDate.Caption = CStr(Date)
End Sub

Private Sub cmdNow_Click()
`Get the date and time. Convert them to a string
cmdNow.Caption = CStr(Now)
End Sub
```



شکل ۴-۲) توابع Time، Date و Now یک نوع داده‌ی Variant (Date) را برمی‌گردانند که با تابع CStr() به رشته تبدیل می‌شوند.

مثال ۴-۲) استفاده از Timer برای ایجاد یک برنامه‌ی ساعت

این برنامه زمان اجرا را روی فرم و تاریخ جاری را در نوار عنوان فرم نشان می‌دهد. هنگامی که فرم به حداقل اندازه رسیده باشد، زمان اجرا در عنوان فرم در نوارابزار ظاهر می‌شود. مراحل انجام کار، به صورت زیر است:

۱- پروژه‌ی Stanard EXE جدیدی را شروع کنید.

۲- یک کنترل label و یک کنترل Timer روی فرم قرار دهید (کنترل برچسب را بزرگ کنید تا کل فرم را بپوشاند. کنترل Timer را می‌توانید در هر جایی قرار دهید، زیرا قابل رؤیت نخواهد بود.)

۳- نام فرم را frmClock و نام برچسب را lblTimeDisplay قرار دهید. نام کنترل Timer را تغییر ندهید (همان نام پیش فرض Timer1 را حفظ کنید).

۴- مشخصه‌ی BorderStyle فرم را به single 1-Fixed تغییر دهید. مشخصه‌ی MinButton فرم را True قرار دهید.

۵- کد زیر را برای روال رویداد Form_Load() بنویسید:

```
01 Private Sub Form_Load()
02 `Set the position and size of the Label control
```

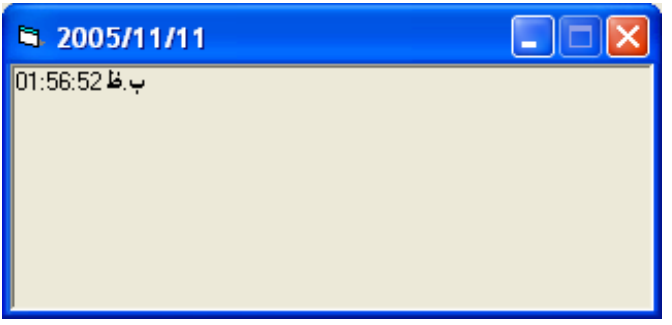
```
03 `to the form's client area.
04 lblTimeDisplay.Top = ScaleTop
05 lblTimeDisplay.Left = ScaleLeft
06 lblTimeDisplay.Width = ScaleWidth
07 lblTimeDisplay.Height = ScaleHeight
08 End Sub
```

۶- مشخصه‌ی Interval کنترل Timer را با ۵۰۰ (نیم ثانیه) مقداردهی کنید. مقدار مشخصه‌ی Enabled کنترل Timer را به True تغییر دهید.

۷- کد مربوط به روال Timer1_Timer() را به صورت زیر بنویسید:

```
01 Private Sub Timer1_Timer()
02 `If the form is diplayed as a window, show
03 `the time in the client area and the date
04 `in the window's title bar.
05 If frmClock.WindowState = vbNormal Then
06 lblTimeDisplay.Caption = CStr(Time)
07 frmClock.Caption = Format(Date, "Long Date")
08 Else
09 `If the form is minimized into the Task Bar,
10 `set the caption to show the time. This will
11 `make the time appear in the Task Bar.
12 frmClock.Caption = CStr(Time)
13 End If
14 End Sub
```

۸- برنامه را ذخیره کرده و آن را اجرا کنید (شکل ۴-۳).



شکل ۴-۳) تنظیم مشخصه‌ی MaxButton با مقدار False دکمه‌ی به حداکثررسانی را غیرفعال می‌کند.

برنامه‌ی ساعت، از چندین چیز استفاده می‌کند که ممکن است قبلاً مشاهده نکرده باشید. تابع Format() که قالب نمایش تاریخ را به صورت روز، ماه و سال کامل تبدیل می‌کند. (در ادامه راجع به این تابع توضیح خواهیم داد.) همچنین تنظیم مشخصه‌ی MinButton با True، دکمه‌ی به حداقل‌رسانی را روی نوار عنوان نمایش می‌دهد و امکان به حداقل‌رسانی فرم را فراهم می‌کند.

فرم‌ها دارای سه مشخصه‌ی MinButton، MaxButton و ContrlBox هستند که روی سه دکمه‌ی سمت راست نوار عنوان تأثیر می‌گذارند. MinButton و MaxButton برای فعال و غیرفعال کردن دکمه‌های Minmize و Maxmize مورد استفاده قرار می‌گیرند. این مشخصه‌ها فقط در زمان طراحی و هنگامی که مشخصه‌ی BorderStyle فرم دارای مقدار 1-Fixed Single یا 2-Sizeable باشد، قابل دسترسی هستند.

مشخصه‌ی ControlBox منوی کنترل سمت چپ نوار عنوان را فعال یا غیرفعال می‌کند. فقط در صورتی می‌توان این مشخصه را فعال کرد که مشخصه‌ی BorderStyle فرم دارای یکی از مقادیر 1-Fixed Single، 2-Sizable یا 3-Fixed Dialog باشد.

نکته:

هنگام استفاده از مشخصه‌ی Window State دقت کنید. ممکن است به طور اتفاقی روی فرمی که Border Style آن Fixed Single است، مشخصه‌ی Window State را با Maxmized مقداردهی کنید. نتیجه‌ی این کار، فرمی است که کل صفحه را در برمی‌گیرد و نمی‌توان آن را تغییر اندازه داد. کاربر نیز روی این فرم کنترلی ندارد.

همچنین برنامه‌ی ساعت، مقدار مشخصه‌ی WindowState فرم را برای تعیین به حداقل‌رسانی فرم، بررسی می‌کند. مشخصه‌ی WindowState دارای سه مقدار است: 0-Normal (vbNormal) ، 1-Minimized (vbMinimized) و 2-Maximized (vbMaximized). می‌توان مقدار این مشخصه را خواند تا تعیین کرد که آیا پنجره در اندازه‌ی عادی، تمام صفحه یا در نوار وظیفه است؟ همچنین می‌توان مشخصه‌ی WindowState را تعیین کرد تا پنجره را به صورت تمام صفحه تبدیل کند، آن را به حالت عادی برگرداند یا در نوار وظیفه با حداقل اندازه نشان دهد.

۴-۴ کاربرد تابع Format()

تابع Format() یکی از توابع قوی ویژوال بیسیک است که امکان کنترل روش نمایش رشته‌ها را فراهم می‌کند. تابع معمولاً برای نمایش مقادیر تاریخ/زمان و اعداد استفاده می‌شود ولی می‌تواند رشته‌ها را نیز به صورت مورد نظر تبدیل کند. شکل کلی این تابع، به صورت زیر است:

```
MyString$=Format(Expression[,Format_String[,FirstDayOfWeek[,FirstWeekYe
ar]]])
```

- MyString$ مقدار برگشتی است.
- Format نام تابع است.
- Expression عبارتی است که یک رشته، تاریخ یا مقدار عددی را برمی‌گرداند.
- Format_String الگوی رشته‌ای است که به تابع بیان می‌کند می‌خواهید رشته‌ی خروجی، به چه صورتی ظاهر شود.
- FirstDayOfWeek یک ثابت اختیاری است که اولین روز هفته را تعیین می‌کند. پیش فرض، Sunday است ولی اگر می‌خواهید روز شنبه باشد، باید آن را بنویسید (Saterday).
- FirstWeekOfYear یک عبارت ثابت اختیاری است که اولین هفته‌ی سال را تعیین می‌کند. به طور پیش فرض، اول ژانویه، اولیه هفته‌ی سال است.

کلید کار کردن با تابع Format()، آشنایی با پارامتر Format_String است. جدول ۴-۲، چگونگی استفاده از تنظیمات این تابع برای تغییر ظاهر رشته‌های تاریخ و زمان را ارایه می‌کند.

نکته:

باید پارامتر Format_String را بین زوج کوتیشن قرار دهید، زیرا تابع این پارامتر را به صورت رشته به کار می‌برد. به عنوان مثال، دستور MyString$ = Format(.50,percent) خطایی را تولید خواهد کرد ولی این دستور به صورت MyString$=Format(.50,"percent") صحیح خواهد بود.

جدول ۴-۲) کاربرد تابع Format() برای زمان و تاریخ

Format_String	مثال	نتیجه

Friday, July 24, 1998 24 – Jul – 98 7/24/98 8:58:34 p.m. 8:58 p.m. 20:58	Format (36000, “Long Date”) Format (36000, “Medium Date”) Format (36000, “Short Date”) Format (0.874, “Long Time”) Format (0.874, “Medium Time”) Format (0.874, “Short Time”)	“Long Date” “Medium Date” “Short Date” “Long Time” “Medium Time” “Short Time”

جدول ۴-۳، چگونگی استفاده از تابع Format() برای کار کردن با مقادیر عددی برای نمایش به صورت رشته‌ی دلخواه را نشان می‌دهد.

جدول ۴-۳) کاربرد تابع Format() برای اعداد

Format String	***Example***	***Result***
"General Number"	Format(36000, "General Number")	36000
"Currency"	Format(36000, "Currency")	$36,000.00
"Fixed"	Format(36000, "Fixed")	36000.00
"Standard"	Format(36000, "Standard")	36,000.00
"Percent"	Format(36000, "Percent")	3600000.00%
"Scientific"	Format(36000, "Scientific")	3.60E+04
Format String	Example	Result
"Yes/No"	Format(36000, "Yes/No")	Yes
"True/False"	Format(36000, "True/False")	True
"On/Off"	Format(36000, "On/Off")	On

همچنین می‌توان قالب‌های رشته‌ای دیگری را نیز به کار برد. به عنوان مثال، اگر می‌خواهید مطمئن شوید که ورودی کاربر همیشه دو رقم اعشار داشته باشد، از قالب Format (235.6, "###,##0.00)" استفاده کنید که مقدار 235.60 را برمی‌گرداند.

۴-۵ محاسبه‌ی اختلاف تاریخ‌ها

هنگامی که نیاز دارید، میزان زمان بین دو تاریخ را بدانید، از تابع ()DateDiff استفاده کنید. تابع ()DateDiff دارای شکل کلی زیر است:

MyLong=DateDiff(Interval,Start_date,End_Date[,FirstDayOfWeek[,FirstWeekOfYear]])

- MyLong مقدار برگشتی بوده و از نوع Long خواهد بود.
- DateDiff نام تابع است.
- Interval رشته‌ای است که فاصله‌ی زمانی براساس آن، تفاوت تاریخ را اندازه‌گیری خواهد کرد (جدول ۴-۴).
- Start_Date تاریخ شروع است (از نوع Date).
- End_Date تاریخ پایان است (از نوع Date).
- FiratDayOfWeek، یک ثابت اختیاری است که اولین روز هفته است.
- FirstWeekOfYear، عبارت ثابت اختیاری است که اولین هفته‌ی سال را تعیین می‌کند.

تابع ()DateDiff اولین تاریخ (Start_date) را منهای دومین تاریخ (End_Date) می‌کند. بعد از تفریق، تابع عددی از نوع Long را برمی‌گرداند که اختلاف بین این تاریخ‌هاست. واحد اندازه‌گیری این اختلاف، براساس مقدار رشته‌ی پارامتر Interval تعیین می‌شود (جدول ۴-۴).

جدول ۴-۴) مقادیر مختلف پارامتر Interval مربوط به تابع ()DateDiff

Value	Interval	Usage	Return Value
"yyyy"	Year	DateDiff("yyyy", "7/4/76", "7/4/86")	10
"q"	Quarter	DateDiff("q", "7/4/76", "7/4/86")	40
"m"	Month	DateDiff("m", "7/4/76", "7/4/86")	120
"y"	Day of year	DateDiff("y", "7/4/76", "7/4/86")	3652
"d"	Day	DateDiff("d", "7/4/76", "7/4/86")	3652
"w"	Weekday	DateDiff("w", "7/4/76", "7/4/86")	521
"ww"	Week	DateDiff("ww", "7/4/76", "7/4/86")	521
"h"	Hour	DateDiff("h", "7/4/76", "7/4/86")	87648
"n"	Minute	DateDiff("n", "7/4/76", "7/4/86")	5258880

"s"	Second	DateDiff("s", "7/4/76", "7/4/86")	315532800

نکته:

IsDate() یک تابع VB است که یک رشته یا مقدار تاریخ را بررسی می‌کند. اگر رشته یا تاریخ شبیه یک تاریخ معتبر باشد، تابع IsDate() مقدار True را برمی‌گرداند و در غیر این صورت. مقدار false را برمی‌گرداند. به عنوان مثال، IsDate("September 16, 1998") و Isdate(#9/16/98#) مقدار True را برمی‌گرداند ولی IsDate("Birthday") مقدار False را برمی‌گرداند.

مثال ۴-۳)

پروژه‌ای مطابق شکل ۴-۴ ایجاد کنید (نام پروژه را DateDff.vbp قرار دهید).

این پروژه چگونگی استفاده از تابع DateDiff() را برای گزارش سن شخص برحسب روز و سال، شرح می‌دهد. کاربر، تاریخ تولد خودش را در کادر متن وارد می‌کند و سپس روی دکمه‌ی Start Counting کلیک می‌کند. داخل روال رویداد Click مربوط به دکمه، کد برنامه، ورودی را بررسی کرده و در صورتی که شبیه یک رشته‌ی تاریخی باشد، با استفاده از تابع IsDate() مقدار True را برمی‌گرداند. اگر رشته معتبر باشد، متن تاریخ تولد به تاریخ تبدیل شده و در متغیری به نام gf-dtBirthday ذخیره می‌شود، سپس Timer فعال می‌شود. در صورتی که رشته معتبر نباشد، پیغام خطایی ظاهر می‌شود.

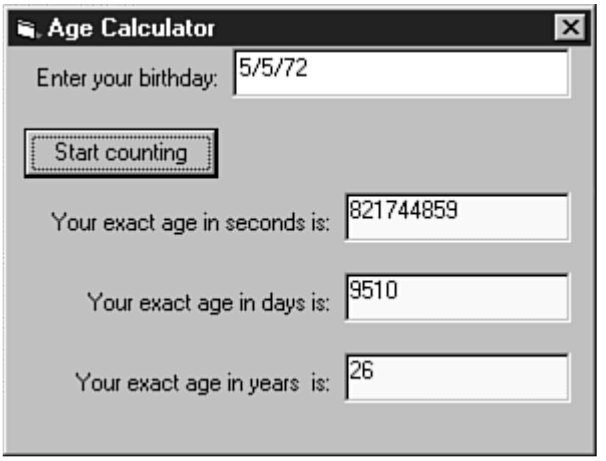
بعد از اینکه کاربر کادر پیام را بست، مکان‌نما به کادر متن برگشته و متن نادرست مشخص شده و از روال خارج می شود.

درون روال رویداد Timer1_Timer() که هر نیم‌ثانیه (Interval = 500) اجرا می‌شود، کنترل Timer با استفاده از تابع Now، تاریخ و ساعت سیستم را بدست می‌آورد. روال رویداد از تابع DateDiff() دوبار استفاده می‌کند (یک بار برای بدست آوردن اختلاف بین زمان جاری و زمان تاریخ تولد براساس روز و بار دیگر برای بدست آوردن نتایج تاریخ‌ها برحسب سال). مقادیر برگشتی در متغیرهای LYourAgeInDays و LYourAgeInYears قرار داده می‌شود. کد زیر، رویدادهای مربوط به این برنامه را ارایه می‌کند:

```
01 Private Sub cmdStart_Click()
02
```

```
`Check to make sure that the string "looks"
`like a date.
If IsDate(txtBDate.Text) Then
`If it is a date, convert the text to a date data
`type and assign it to the global birthday date
`variable
gf_dtBirthday = CDate(txtBDate.Text)
Else
`If it isn't, then report an error
MsgBox "You must enter a proper date!", vbCritical, "Data error"
`Set the cursor back to the textbox
txtBDate.SetFocus
`Set the cursor to the beginning of the text box
txtBDate.SelStart = 0
`Highlight the erroneous text
txtBDate.SelLength = Len(txtBDate.Text)
`Leave the sub
Exit Sub
End If

`Turn on the timer
Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
Dim lYourAgeInSecs As Long
Dim lYourAgeInDays As Long
Dim lYourAgeInYears As Long

`Calculate the date difference in seconds
lYourAgeInSecs = DateDiff("s", gf_dtBirthday, Now)

`Calculate the date difference in days
lYourAgeInDays = DateDiff("d", gf_dtBirthday, Now)

`Calculate the date difference in years
lYourAgeInYears = DateDiff("yyyy", gf_dtBirthday, Now)

`Report the date differences
lblAgeSecs.Caption = CStr(lYourAgeInSecs)
lblAgeDays.Caption = CStr(lYourAgeInDays)
```

```
44 lblAgeYears.Caption = CStr(lYourAgeInYears)
45 End Sub
```



شکل ٤-۴) استفاده از تابع Format() خواندن مقادیر روز و ثانیه‌ها را ساده‌تر می‌کند.

۴-۶ کاربرد متغیرهای ایستا به همراه Timer

فرض کنید که می‌خواهید برنامه‌ای بنویسید که عملی را در هر نیم‌ثانیه تا ۱۰ بار انجام دهد. برای انجام این کار، نیاز دارید متغیری ایجاد کنید که تعداد دفعات اجرای رویداد Timer را نگه دارد. اگر یک متغیر شمارنده داخل رویداد Timer ایجاد کنید، هر بار که روال قطع شود، متغیر از حوزه‌ی عمل خود خارج شده و با صفر مقداردهی می شود.

```
01 Sub Timer1_Timer()
02 If g_TimeLimit% > 10 Then
03 MsgBox "Attempts exceeded!"
04 Else
05 `Attempt to do something
06 End if
07
08 g_TimeLimit% = g_TimeLimit% + 1
09
10 End Sub
```

اگر چه این کد، کار خواهد کرد ولی بهینه نیست، زیرا ایجاد متغیر عمومی آن را مستقل از روال می‌کند. اگر بخواهید Timer را از کد حذف کنید، این متغیر عمومی، بدون استفاده خواهد ماند.

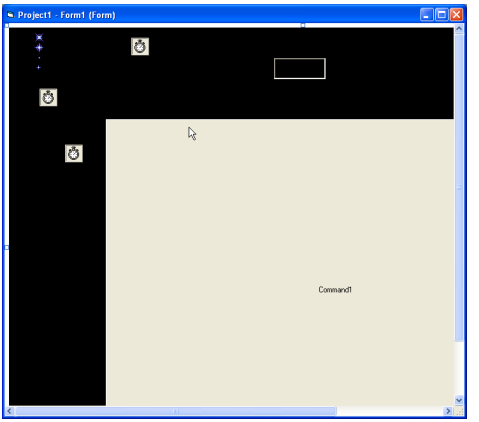
یک راهکار بهتر، تعریف متغیر شمارنده با کلید واژه‌ی Static است. کد زیـر، چگـونگی انجـام این کار را نشان دهد:

```
01 Private Sub Timer1_Timer()
02 `Make a static variable that will retain value even
03 `after the event procedure terminates
04 Static i%
05
06 `Report the present value of i%
07 frmMain.Caption = "Present value of i%: " & CStr(i%)
08
09 `Report the running time
10 lblTime.Caption = Format(Time, "Long Time")
11
12 `Check to see if the counter has exceeded 10 loops
13 If i% >= 10 Then
14 `If the counter is exceeded, send a notice
15 lblLimitNotice.Caption = "Limit Exceeded!"
16 Else
17 `If not, keep the reporting label blank
18 lblLimitNotice.Caption = ""
19 End If
20
21 `Increment the counter
22 i% = i% + 1
23 End Sub
```

هنگامی که یک متغیر Static ایجاد می‌کنید، مقدار آن حتی بعـد از خـروج از روالـی کـه در آن تعریف شده است، حفظ می‌شود. مزیت انجام این کار، این است که متغیر درونی کنترلی کـه بـه آن وابسته است، کپسوله‌سازی می‌شود. مقدار متغیر بدون در نظر گرفتن وضعیت روالی که در آن ایجـاد شده است، پایدار باقی می‌ماند.

مثال ٤-٤)

۱- پروژه‌ای مطابق شکل ۴-۵ ایجاد کنید و کد مربوطه را وارد کنید.



شكل ٤-٥)

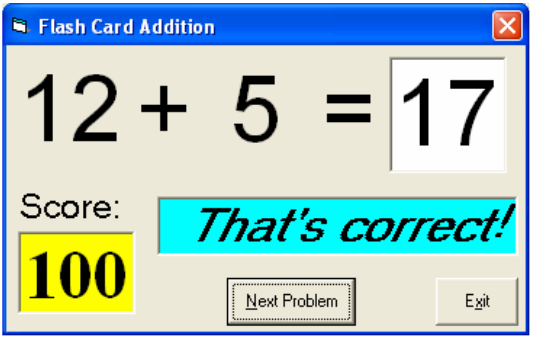
```
Const NumberOfSnow = 40
Const MaxSpeed = 3
Dim x, y, n
Dim StarCnt
Dim N_Snow(1 To NumberOfSnow, 1 To 4)

Private Sub Form_Activate()
   Command1.Top = 112
   Command1.Left = 152
   Command1.Width = 731
   Command1.Height = 537
   x = 400
   y = 100
For i = 0 To 3
   Imgstar(i).Top = 130
   Imgstar(i).Left = 450
   Imgstar(i).Visible = False
Next
Imgstar(0).Visible = True
Randomize Timer
For i = 1 To NumberOfSnow
   N_Snow(i, 1) = Int(Rnd * (Command1.Width - 20)) + Command1.Left
   N_Snow(i, 2) = Command1.Top - Int(Rnd * 100)
   N_Snow(i, 3) = Int(Rnd * MaxSpeed + 1)
   N_Snow(i, 4) = Int(Rnd * 7) + 5
```

```
Next
Me.ForeColor = &HFFF0F0
Me.Font.Name = "symbol"
StarCnt = 0
End Sub

Private Sub Form_Click()
    End
End Sub

Private Sub startimer_Timer()
      Imgstar(StarCnt).Visible = False
      StarCnt = StarCnt + 1
      If StarCnt = 4 Then StarCnt = 0
      Imgstar(StarCnt).Visible = True
End Sub

Private Sub Timer1_Timer()
  Me.Picture = Picture1.Picture
   For i = 1 To NumberOfSnow
   Me.CurrentX = N_Snow(i, 1)
   Me.CurrentY = N_Snow(i, 2)
   Me.FontSize = N_Snow(i, 4)
   If Me.CurrentY > Command1.Top Then Me.Print Chr(&HB7)
   N_Snow(i, 2) = N_Snow(i, 2) + N_Snow(i, 3)
   If N_Snow(i, 2) > Command1.Height + Command1.Top - 15 Then
      N_Snow(i, 1) = Int(Rnd * (Command1.Width - 20)) + Command1.Left
      N_Snow(i, 2) = Command1.Top
      N_Snow(i, 3) = Int(Rnd * MaxSpeed + 1)
   End If
   If x < Command1.Left Then
      x = Command1.Left + Command1.Width \ 2
      y = Command1.Top
   End If
   Next
End Sub
```

۲- پروژه را اجرا کنید (شکل ۴-۶).

شکل ٤-٦)

مثال ۴-۵ : محاسبه‌ی جمع‌های تصادفی)



شکل ۴-۷)

مشخصه‌ها :

Form **frmAdd**:
BorderStyle = 1 - Fixed Single
Caption = Flash Card Addition
CommandButton **cmdNext**:
Caption = &Next Problem
Enabled = False
CommandButton **cmdExit**:
Caption = E&xit
TextBox **txtAnswer**:
FontName = Arial

FontSize = 48
MaxLength = 2

Label **lblMessage**:
Alignment = 2 - Center
BackColor = &H00FFFF00& (Cyan)
BorderStyle = 1 - Fixed Single
FontName = MS Sans Serif
FontBold = True
FontSize = 24
FontItalic = True

Label **lblScore**:
Alignment = 2 - Center
BackColor = &H0000FFFF& (Yellow)
BorderStyle = 1 - Fixed Single
Caption = 0
FontName = Times New Roman
FontBold = True
FontSize = 36

Label **Label1**:
Alignment = 2 - Center
Caption = Score:
FontName = MS Sans Serif
FontSize = 18

Label **Label4**:
Alignment = 2 - Center
Caption = =
FontName = Arial
FontSize = 48

Label **lblNum2**:
Alignment = 2 - Center
FontName = Arial
FontSize = 48

Label **Label2**:
Alignment = 2 - Center
Caption = +
FontName = Arial
FontSize = 48

Label **lblNum1**:
Alignment = 2 - Center

FontName = Arial
FontSize = 48

کد این برنامه به صورت زیر خواهد بود:

```
Option Explicit
Dim Sum As Integer
Dim NumProb As Integer, NumRight As Integer

Private Sub cmdExit_Click()
End
End Sub
Private Sub cmdNext_Click()
'Generate next addition problem
Dim Number1 As Integer
Dim Number2 As Integer
txtAnswer.Text = ""
lblMessage.Caption = ""
NumProb = NumProb + 1
'Generate random numbers for addends
Number1 = Int(Rnd * 21)
Number2 = Int(Rnd * 21)
lblNum1.Caption = Format(Number1, "#0")
lblNum2.Caption = Format(Number2, "#0")
'Find sum
Sum = Number1 + Number2
cmdNext.Enabled = False
txtAnswer.SetFocus
End Sub
Private Sub Form_Activate()
Call cmdNext_Click
End Sub

Private Sub Form_Load()
Randomize Timer
NumProb = 0
NumRight = 0
End Sub

Private Sub txtAnswer_KeyPress(KeyAscii As Integer)
```

```
Dim Ans As Integer
'Check for number only input and for return key
If (KeyAscii >= vbKey0 And KeyAscii <= vbKey9) Or KeyAscii = vbKeyBack
Then
  Exit Sub
ElseIf KeyAscii = vbKeyReturn Then
'Check answer
  Ans = Val(txtAnswer.Text)
  If Ans = Sum Then
    NumRight = NumRight + 1
    lblMessage.Caption = "That's correct!"
  Else
    lblMessage.Caption = "Answer is "+Format(Sum, "#0")
  End If
  lblScore.Caption = Format(100*NumRight/NumProb,"##0")
  cmdNext.Enabled = True
  cmdNext.SetFocus
Else
  KeyAscii = 0
End If
End Sub
```

مثال ٤-٦ : شمارش معکوس

۱- پروژه‌ای ایجاد کرده و یک کنترل Timer و یک برچسب روی فرم آن قرار دهید.

۲- مشخصه‌ی Caption برچسب را 2:00:00 قرار دهید (شکل۴-۸).

شکل ٤-٨)

۳- کد زیر را وارد کنید:

```
Option Explicit
Private AlarmTime As Date
```

```
Private Sub Form_Load()
   AlarmTime = DateAdd("h", 2, Now)
End Sub

Private Sub Timer1_Timer()
Dim txt As String
   txt = Format$(AlarmTime - Now, "h:mm:ss")
   Label1.Caption = txt
End Sub
```

۴- پروژه را اجرا کنید.

خودآزمایی

۱. درهر یک از دستورات زیر مقدار ذخیره شده در متغیر strS را مشخص کنید.

a. strS = Format (“74135”, “&&&&&-&&&&”)
b. strS = Format(d, “h ampm”)
c. strS = Format(12345.67, “######.###”)

فصل ششم
روال‌ها و توابع

۶-۱ استفاده از روال‌ها در ویژوال بیسیک

تا اینجا با مفاهیمی مثل Subs و Functions آشنا شده‌اید. روال‌های رویدادی مثل Click() و Load() از نوع Subs هستند و در ویژوال بیسیک توابعی مثل LoadPicture() و Len() نیز وجود دارند که قبلاً با آنها کار کرده‌اید.

ویژوال بیسیک، یک زبان برنامه‌نویسی روالی است. بعد از نامگذاری یک بلاک کد، می‌توان آن را فراخوانی و اجرا کرد. به عبارت دیگر، می‌توان چند خط کد نوشت و آن را در یک بلاک قرار داده و نامی به آن اختصاص داد. سپس بلاک کد را هنگام نیاز فراخوانی کرد. این بلاک کد تقریباً شبیه برنامه‌ای در داخل برنامه‌ی دیگر است. این برنامه‌های کوچک که داخل برنامه‌های بزرگ هستند را در صورتی که مقدار برگردانند، "توابع" و در غیر این صورت "Subs" می‌نامند.

برنامه‌نویسی با این مفاهیم، سال‌هاست که ادامه دارد (در حقیقت subs مخفف subroutine است.) و کدنویسی را ساده‌تر، سریع‌تر و کارامدتر می‌کنند. همچنین استفاده از این مفاهیم، امکان نوشتن کدهای کپسوله‌سازی شده و قابل استفاده مجدد را فراهم می‌کنند. کپسوله‌سازی به سادگی متدها و مشخصه‌های یک شیء را با هم در یکجا و پشت یک رابط عمومی قرار می‌دهد.

۶-۲ ایجاد و فراخوانی یک Sub ساده

نکته:

Subs امکان تغییر ساده‌ی کد را فراهم می‌کند. اگر کدی که نیاز به استفاده‌ی مکرر آن دارید، کد را در یک sub قرار دهید. سپس اگر نیاز به تغییر کد داشته باشید، به سادگی می‌توانید به sub رجوع کرده و تغییرات را اعمال کنید. اگر کد را در یک sub قرار ندهید، مجبور خواهید بود که به هر نمونه‌ای از کد در برنامه رجوع کرده و تغییرات مورد نیاز را اعمال کنید که انجام تغییرات مؤثر و کارامد با این روش، مشکل خواهد بود.

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

```
[private ¦ public] sub    subName()
خطوطی از کد ....
End Sub
```

- [private ¦ public] کلیدواژه‌های اختیاری هستند که حوزه‌ی عمل sub را تعریف می‌کنند.
- Sub کلیدواژه‌ای است که نوع روال را تعیین می‌کند.
- SubName نامی است که برای روال تعیین می‌شود.
- End Sub کلیدواژه‌هایی هستند که پایان بلاک کد را مشخص می‌کنند.

کد زیر، مثالی از یک sub ساده است:

```
Public Sub DataNotFound( )
    MgBox "Data Not Found" , vbInformation
End Sub
```

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

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

```
Private Sub itmOpen_Click( )
    Call DataNotFound
End Sub
```

۶-۳ ایجاد Subs با استفاده از Add Procedure

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

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

نکته:

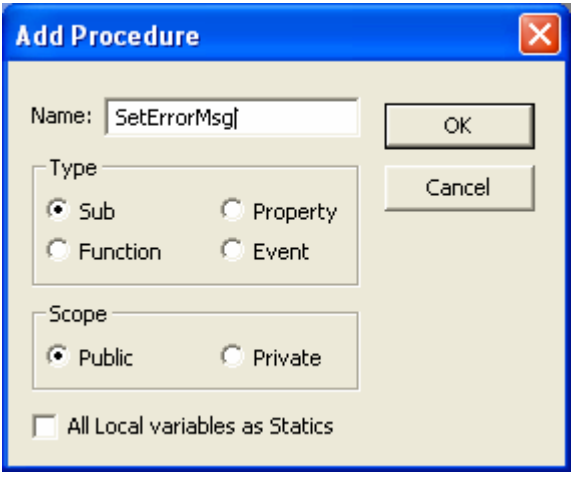
برای فعال کردن گزینه‌ی Add Procedure باید در پنجره‌ی Code فرم یا مدول موردنظر باشید.

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

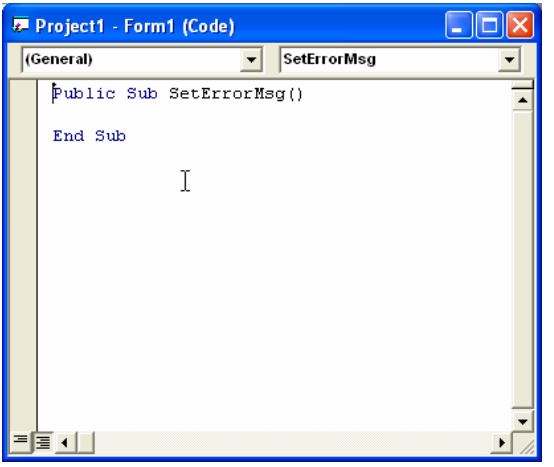
۱- از منوی Tools گزینه‌ی Add Procedure را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود.

۲- نامی را برای sub وارد کنید (شکل ۶-۱).

۳- روی Ok کلیک کرده و بلاک کد را به فرم یا مدول اضافه کنید (شکل ۶-۲).



شکل ٦-١) کادر محاوره‌ای Add Procedure امکان ایجاد Subs و توابع برای انواع پروژه‌های VB را فراهم می‌کند.



شکل ٦-٢) sub جدیدی را در بخش General فرم یا مدول بدست خواهید آورد.

بعد از اینکه بلاک کد را با کادر محاوره‌ای Add Procedure ایجاد کردید، کد روال را داخل بلاک کد اضافه کنید. بعد از End Sub کدی را وارد نکنید، انجام این کار سبب بروز خطا در زمان کامپایل خواهد شد.

٦-٤ ایجاد یک تابع ساده

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

```
[Private ¦ Public] Function FunctionName() As DataType
        خطوطی از کد ....
    FunctionName = ReturnValue
End Function
```

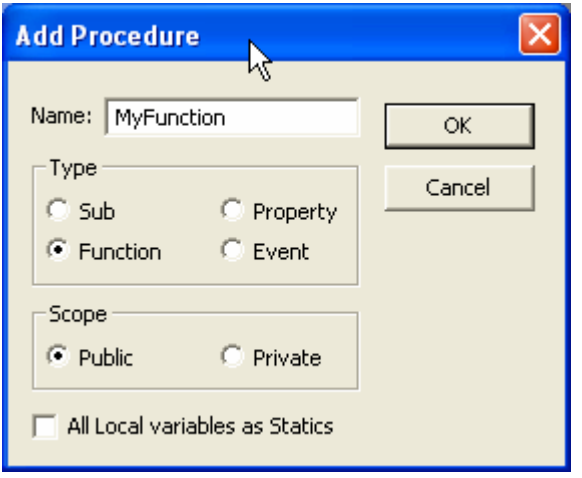
- Private ¦ Public کلیدواژه‌های اختیاری هستند که حوزه‌ی عمل تابع را تعریف می‌کنند.
- Function کلیدواژه‌ای است که مشخص می‌کند، روال از نوع تابع است.
- FunctionName نام تابع است.
- As کلیدواژه‌ای برای تعیین نوع داده است.
- DataType نوع داده‌ی مقداری است که تابع برمی‌گرداند.
- ReturnValue مقداری است که به وسیله‌ی تابع برگردانده می‌شود.
- End Function کلیدواژه‌هایی هستند که پایان بلاک کد را مشخص می‌کنند.

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

```
01 Public Function GetNumber() As Integer
02 Dim a%
03 Dim b%
04 Dim c%
05 `Assign values to some variables
06 a% = 7
07 b% = 12
08
09 `Add them together
10 c% = a% + b%
```

```
11
12 `Pass the result out of the function by assigning
13 `it to the function name.
14 GetNumber = c%
15 End Function
```

تابع را نیز می‌توان مانند Sub با همان دو روش به فرم یا مدول اضافه کرد (شکل ۶-۳).



شکل ٦-٣) تابع را در کادر محاوره‌ای Add Procedure اضافه کنید.

٦-٥ ارسال آرگومان‌ها به Subs و Functions

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

```
EndDay (NumOne As Integer, strName As String) As Integer
```

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

کد زیر، تابعی به نام GetGreaterNum() را نشان می‌دهد که بزرگترین مقدار بین دو عدد دریافتی را محاسبه و برمی‌گرداند.

```
01 Public Function GetGreaterNum(NumOne As Integer, _
NumTwo As Integer) As Integer
02 `If the first number is greater than the second
03 If NumOne > NumTwo Then
04 `return the first number
05 GetGreaterNum = NumOne
06 Else
07 `if not, return the second number
08 GetGreaterNum = NumTwo
09 End If
10 End Function
```

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

```
01 Private Sub cmdGreaterNum_Click()
02 Dim i%
03 Dim j%
04 Dim RetVal%
05
06 `Get the input in txtNumOne and convert it to an integer
07 i% = CInt(txtNumOne.Text)
08
09 `Get the input in txtNumTwo and convert it to an integer
10 j% = CInt(txtNumTwo.Text)
11
12 RetVal% = GetGreaterNum(i%, j%)
13
14 `Take the result from the function, convert it to a
15 `string and assign it to the caption of the button.
16 cmdGreaterNum.Caption = CStr(RetVal%)
17 End Sub
```

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

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

و تابع را با استفاده از کد زیر فراخوانی می‌کنید،

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

این فراخوانی، خطایی را تولید می‌کند. "D56R" از نوع رشته‌ای است ولی تابع برای آرگومان دوم انتظار داده‌ای از نوع Double را دارد.

همچنین تعداد آرگومان‌ها نیز باید یکسان باشند. به عنوان مثال، فرض کنید تابعی دارید که به صورت زیر تعریف شده است:

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

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

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

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

ممکن است آرگومانی را با استفاده از کلیدواژه‌ی Optional که قبل از آرگومان هنگام اعلان تابع قرار می‌گیرد، به صورت اختیاری تعیین کرد. آرگومان‌های اختیاری باید از نوع Variant باشند.

٦-٥-١ کاربرد آرگومان‌های نام‌دار

می‌توان از آرگومان‌های نام‌دار برای ارسال ساده‌تر مقادیر به روال‌ها استفاده کرد. یک آرگومان نام‌دار، نام رشته‌ای از آرگومان در روال است. به عنوان مثال، اگر تابعی به نام EndDay() دارید که دارای دو آرگومان از نوع Integer است و به صورت زیر تعریف شده است:

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

برای ارسال مقداری به تابع با استفاده از آرگومان‌های دارای نام، از اسامی آرگومان‌ها استفاده کرده و با استفاده از نویسه‌های =: مقداری را برای آنها تعیین کنید. بنابراین، برای ارسال واقعی مقادیر در تابع EndDay() با استفاده از آرگومان‌های نام‌دار، به صورت زیر عمل کنید:

```
X = EndDay( NumOne : =3, NumTwo :- 4)
```

٦-٦ خروج از روال‌ها

بعضی مواقع قبل از پایان روال، نیاز به ترک آن دارید. می‌توانید این کار را با کلیدواژه‌ی Exit انجام دهید. کد زیر، تابع ExitEarly() را نشان می‌دهد که دو آرگومان می‌گیرد. یک عدد صحیح

برای تعیین حد بالای حلقه و یک عدد صحیح دیگر که نـشانه‌ای بـرای خـروج از تـابع در صـورت برقراری شرط خاصی است.

```
01 Public Function ExitEarly(iLimit As Integer, _
iFlag As Integer) As Integer
02 Dim i%
03 Dim Limit%
04 Dim Flag%
05
06 `Assign the limit argument to a local variable
07 Limit% = iLimit
08
09 `Assign the state argument to local variable
10 Flag%= iFlag
11
12 `Run a For...Next loop to Limit%
13 For i% = 0 To Limit%
14
15 `If the passed in state is one
16 If Flag% = 1 Then
17
18 `Check to see if i% equals half the value of
19 `the Limit variable
20 If i% = Limit% / 2 Then
21
22 `If it does, pass out the value of i%
23 `at that point
24 ExitEarly = i%
25
26 `Terminate the function; there is no
27 `reason to go on
28 Exit Function
29 End If
30 End If
31 Next i%
32
33 `If you made it this far, the flag variable does not
34 `equal one, so pass the value of i% out of the
35 `function by assigning the value of i% to the
```

```
36 `function name.
37 ExitEarly = i%
38
39 End Function
```

٦-٧ آشنایی با حوزه عمل

حوزه‌ی عمل (میدان دید)، قابلیت دو متغیر مختلف هستند که دارای نام یکسان بوده و مقادیر مختلفی را نگهداری می‌کنند و دارای دوره‌ی حیات متفاوتی هستند. کد زیر، دو تابع ()EndDay و ()Bar را نشان می‌دهد:

```
01 Public Function EndDay() as Integer
02 Dim x as Integer
03 Dim y as Integer
04
05 x = 2
06 y = 7
07 EndDay = x + y
08 End Function
09
10 Public Function Bar() as Integer
11 Dim x as Integer
12 Dim y as Integer
13
14 x = 12
15 y = 34
16 Bar = x * y
17 End Function
```

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

۸-۶ مستندسازی روال‌ها

مستندسازی روال‌ها به سایر برنامه‌نویسان امکان می‌دهد که به طور کامل از برنامه‌ی شما استفاده کرده و در صورت لزوم اشکالات آن را رفع کنند.

تمام روال‌ها دارای یک سرآیند خواهند بود. سرآیند (header) بخشی از کد است که در ابتـدای بلاک کد آورده شده و توضیحاتی را درباره‌ی روال ارایه می‌دهد. کد زیر همـان تـابع ExitEarly() است که به صورت حرفه‌ای مستندسازی شده است:

```
01 Public Function ExitEarly(iLimit As Integer, _
02 iFlag As Integer) As Integer
03 `****************************************
04 `Sub/Function: ExitEarly
05 `
06 `Arguments: iLimit The upper limit of the For..Next Loop
07 ` iFlag An integer indicating early exit from
08 ` the function. 1 = Exit.
09 ` Other values are ignored.
10 `
11 `Return: The value of the For...Next loop counter
12 `
13 `Remarks: This function is used to demonstrate the way
14 ` to use arguments within a function
15 `
16 `Programmer: Bob Reselman
17 `
18 `History: Created 4/20/98
19 `
20 `Copyright 1998, Macmillan Publishing
21 `****************************************
22
23 Dim i% `Counter variable
24 Dim Limit% `Internal variable for the upper limit of the
25 `For...Next loop
26 Dim Flag% `Internal variable for the exit flag
27
28 `Assign the limit argument to a local variable
29 Limit% = iLimit
```

```
30
31 `Assign the state argument to local variable
32 Flag% = iFlag
33
34 `Run a For...Next loop to Limit%
35 For i% = 0 To Limit%
36
37 `If the passed in state is one
38 If Flag% = 1 Then
39
40 `Check to see if i% equals half the value of
41 `the Limit variable
42 If i% = Limit% / 2 Then
43
44 `If it does, pass out the value of i%
45 `at that point
46 ExitEarly = i%
47
48 `Terminate the function; there is no reason
49 `to go on
50 Exit Function
51 End If
52 End If
53 Next i%
54
55 `If you made it this far, the state variable does not
56 `equal one, so pass the value of i% out of the function
57 `by assigning the value of i% to the function name.
58 ExitEarly = i%
59
60 End Function
```

۹-٦ تعیین نقطه‌ی ورودی با SubMain()

به طور پیش فرض، هنگامی که پروژه‌ای را در VB شروع می‌کنید، اولین فرم ایجاد شده فرمی خواهد بود که پروژه در آغاز فراخوانی می‌کند. این کار در صورتی که یک فرم در پروژه دارید، ممکن است ولی اگر دارای پروژه‌ای با چندین فرم هستید، می‌توانید فرم‌های دیگر را از داخل اولین فرم، بارگذاری (فراخوانی) کنید:

```
Private Sub Form-Load( )
    Load frmAnotherForm
End Sub
```

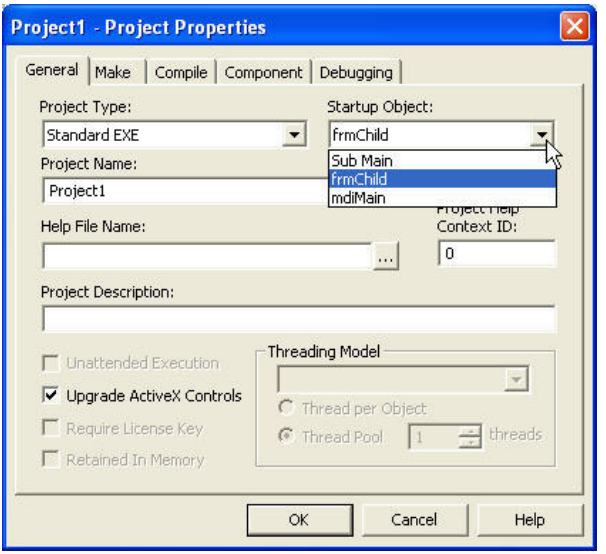
این روش زمانی مفید است که تعداد فرم‌ها محدود باشند.

برای پروژه‌هایی که دارای هیچ فرمی نیستند (مثل برنامه‌های اینترنتی که در سمت سرور کار می‌کنند)، چیزی برای بارگذاری وجود ندارد. چه کاری باید برای نقطه‌ی شروع (نقطه‌ی ورودی) برنامه انجام دهید؟ ویژوال بیسیک، یک نقطه شروع غیر مبتنی بر فرم را برای برنامه ارایه می‌کند (روال Sub Main()). Sub Main() روال خاصی است که به وسیله‌ی ویژوال بیسیک به عنوان روال شروع هر پروژه‌ای رزرو شده است. Sub Main() باید در یک مدول اعلان شود و برای هر پروژه فقط می‌توان یک Sub Main() در نظر گرفت. مراحل زیر، چگونگی تعیین این روال به عنوان نقطه‌ی شروع را نشان می‌دهند:

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

۲- در لیست بازشوی Startup Object گزینه‌ی Sub Main از زبانه‌ی General کادر محاوره‌ای را انتخاب کنید.

۳- روی Ok کلیک کنید.



شکل ٦-٤) می‌توان Sub Main() یا هر فرم دیگری را به عنوان شیء شروع در پروژه انتخاب کرد.

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

```
01 Sub Main()
02 `Use the Show method to display both
03 `forms upon startup
04 frmMain.Show
05 frmOther.Show
06
07 `Report that all forms are shown
08 MsgBox "Everything shown"
09 End Sub
```

کد مربوط به بعضی از نسخه‌های Sub Main() می‌تواند ساده باشد ولی برای بعضی از روال‌های Sub Main() می‌تواند پیچیده باشد. کد زیر نشان می‌دهد که Sub Main() چگونه برای اجرای روال شروعی که سایر روال‌ها را فراخوانی می‌کند، مورداستفاده قرار می‌گیرد. این کد مربوط به روال Sub Main() برنامه‌ی VBSchedular است.

```
01 Sub Main()
02 `Load the form and let it run the code in the
03 `Form_Load event handler
04 Load frmMain
05 `Intialize the contact list combo box
06 Call InitComboAsDb(frmMain.cboName, frmMain.DataMain)
07 `Fill the appointment list with today's appoinments
08 Call GetDailyAppointments(CDbl(Cdate (frmMain.FormDateString())), _
frmMain.lstSchedule, frmMain.DataMain, gAppointmentDelta%)
09 `Show the main form
10 frmMain.Show
11 `Set the mouse pointer back to an arrow
12 frmMain.MousePointer = 0
13 End Sub
```

مثال ٦-١) مثال ٢-١٠را به کمک یک روال بنویسید:

```
Private Sub setcolor()
    r = HScroll1.Value
    g = HScroll2.Value
    b = HScroll3.Value
    Lblr = r
    Lblg = g
    Lblb = b
    Label1.BackColor = RGB(r, g, b)
End Sub
Private Sub Command1_Click()
    End
End Sub


Private Sub Form_Load()
    Label1.BackColor = RGB(0, 0, 0)
End Sub
```

```
Private Sub HScroll1_Change()
   Call setcolor
End Sub
Private Sub HScroll1_Scroll()
   Call setcolor
End Sub
Private Sub HScroll2_Change()
   Call setcolor
End Sub
Private Sub HScroll2_Scroll()
   Call setcolor
End Sub
Private Sub HScroll3_Change()
   Call setcolor
End Sub
Private Sub HScroll3_Scroll()
   Call setcolor
End Sub
```

۱۰-٦ توابع تشخیص نوع داده

این توابع نوع داده‌ی آرگومان را مشخص می‌کنند. برنامه‌ها معمولاً با داده‌های مختلفی سر و کار دارند، ولی گاهی برنامه‌نویس از قبل نمی‌تواند حدس بزند که با چه نوع داده‌ای سر و کار خواهد داشت. مثلاً، قبل از آنکه محاسبه‌ای انجام دهید باید مطمئن شوید که داده‌ها از نوع عددی هستند. در جدول ۶-۱ توابع Is...() را مشاهده می‌کنید؛ آرگومان این توابع همگی از نوع Variant است.

جدول ٦-۱) توابع تشخیص نوع داده.

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

در برنامه‌ی زیر، چگونگی استفاده از تابع IsEmpty() نشان داده شده است.

```
1: 'Code that tests the Is() functions
```

```
2: Dim var1 As Variant, var2 As Variant
3: Dim var3 As Variant, var4 As Variant
4: Dim intMsg As Integer          'MsgBox return
5: ' Fill variables with sample values to test
6: var1 = 0          'Zero value
7: var2 = Null       'Null value
8: var3 = ""         'Null string
9: ' Call each Is() function
10: If IsEmpty(var1) Then
11:       intMsg = MsgBox("var1 is empty", vbOKOnly)
12: End If
13: If IsEmpty(var2) Then
14:          intMsg = MsgBox("var2 is empty", vbOKOnly)
15: End If
16: If IsEmpty(var3) Then
17:          intMsg = MsgBox("var3 is empty", vbOKOnly)
18: End If
19: If IsEmpty(var4) Then
20:          intMsg = MsgBox("var4 is empty", vbOKOnly)
21: End If
```

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

```
var4 is empty
```

چون تمام متغیرهای دیگر مقدار گرفته‌اند (حتی Null هم یک مقدار محسوب می‌شود). برای تست کردن Null می‌توانید از IsNull() استفاده کنید. توجه داشته باشید که شرط زیر

```
If (varA = Null) Then …
```

حتی اگر متغیر varA واقعاً Null باشد، True نخواهد شد. در این موارد تنها راه‌حل استفاده از تابع IsNull() است. به قطعه کد زیر توجه کنید:

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

در اینجا برنامه قبل از ادامه‌ی کار، خالی نبودن یکی از فیلدهای برنامه را تست می‌کند.

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

- Empty (به ۰ تبدیل می‌شود)
- اعداد صحیح (Integer)
- اعداد صحیح بلند (Long)
- اعداد اعشاری (Single)
- اعداد اعشاری با دقت مضاعف (Double)
- واحد پول (Cueerncy)
- تاریخ
- رشته (اگر شبیه یک عدد باشد)

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

```
1: Dim varAge As Variant
2: Dim intMsg As Integer          'MsgBox( ) return
3: varAge = InputBox("How old are you?", "Get Your Age")
4: If IsNumeric(varAge) Then
5:   intMsg = MsgBox("Thanks!", vbOKOnly)
6: Else
7:     intMsg = MsgBox("What are you trying to hide?", _
       vbOKOnly+vbQuestion)
8: Ens If
```

درستی پاسخ کاربر در خط ۴ تست می‌شود.

اگر می‌خواهید نوع یک متغیر را بدانید، باید از تابع VarType() استفاده کنید. جدول ۶-۲ مقادیر برگشتی این تابع را نشان می‌دهد.

جدول ٦-۲) مقادیر برگشتی تابع VarType()

نوع داده	ثابت نام‌دار	مقدار برگشتی
Empty	vbEmpty	0
Null	vbNull	1
Integer	vbInteger	2
Long	vbLong	3

Single	vbSingle	4
Double	vbDouble	5
Currency	vbCurrency	6
Date	vbDate	7
String	vbString	8
Object	vbObject	9
یک مقدار خطا	vbError	10
Boolean	vbBoolean	11
Variant	vbVariant	12
یک شیء دسترسی داده	vbDataObject	13
Decimal	vbDecimal	14
Byte	vbByte	17
یک آرایه	vbArray	8192

در برنامه‌ی زیر، با دستور Select Case نوع داده‌ی ارسال شده به تابع مشخص شده است.

```
1: Private Sub PrntType(varA)      'Variant if you don't specify otherwise
2:  Dim intMsg As Integer          'MsgBox() return
3: Select Case VarType(varA)    'VarType() returns an integer
4:        Case 0
5:           intMsg = MsgBox("The argument is Empty")
6:        Case 1
7:           intMsg = MsgBox("The argument is null")
8:        Case 2
9:           intMsg = MsgBox("The argument is Iteger")
10:       Case 3
11:          intMsg = MsgBox("The argument isLong")
12:       Case 4
13:          intMsg = MsgBox("The argument is Single")
14:       Case 5
15:          intMsg = MsgBox("The argument is Double")
16:       Case 6
17:          intMsg = MsgBox("The argument is Currency")
18:       Case 7
19:          intMsg = MsgBox("The argument is Date")
20:       Case 8
21:          intMsg = MsgBox("The argument is String")
22:       Case 9
23:          intMsg = MsgBox("The argument is Object")
24:       Case 10
25:          intMsg = MsgBox("The argument is Error")
26:       Case 11
27:          intMsg = MsgBox("The argument is Boolean")
```

```
28:        Case 12
29:           intMsg = MsgBox("The argument is a Variant array")
30:        Case 13
31:           intMsg = MsgBox("The argument is a Data Access Object")
32:        Case 14
33:           intMsg = MsgBox("The argument is Decimal")
34:        Case 17
35:           intMsg = MsgBox("The argument is Byte")
36:        Case Else
37:           intMsg = MsgBox("The argument is an Array")
38: End Select
39: End Sub
```

۶-۱۱ توابع تبدیل نوع

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

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

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

```
intA1 = Ctnt (8.5)        'Stores an 8 in intA1
intA2=Cint (8.5001)       'Stores an 9 in intA2
```

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

جدول ۶-۳) توابع تبدیل نوع

تابع	مفهوم
CBool()	آرگومان خود را به نوع Boolean تبدیل می‌کند.
CByte()	آرگومان خود را به نوع Byte تبدیل می‌کند.
CCur()	آرگومان خود را به نوع Currency تبدیل می‌کند.
CDate()	آرگومان خود را به نوع Date تبدیل می‌کند.
CDbl()	آرگومان خود را به نوع Double تبدیل می‌کند.

CDec()	آرگومان خود را به نوع Decimal تبدیل می‌کند.
CInt()	آرگومان خود را به نوع Integer تبدیل می‌کند.
CLng()	آرگومان خود را به نوع Long تبدیل می‌کند.
CSng()	آرگومان خود را به نوع Single تبدیل می‌کند.
CStr()	آرگومان خود را به نوع String تبدیل می‌کند.
CVar()	آرگومان خود را به نوع Variant تبدیل می‌کند.

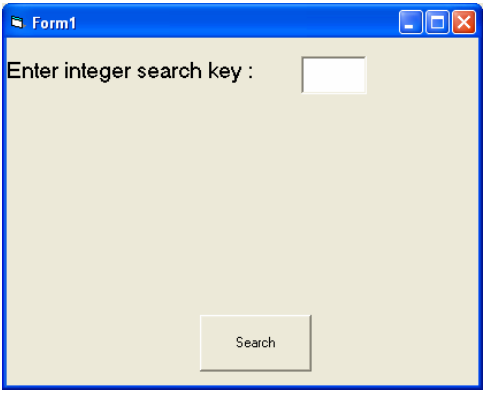
مثال ٦-٢: جستجوی دودویی)

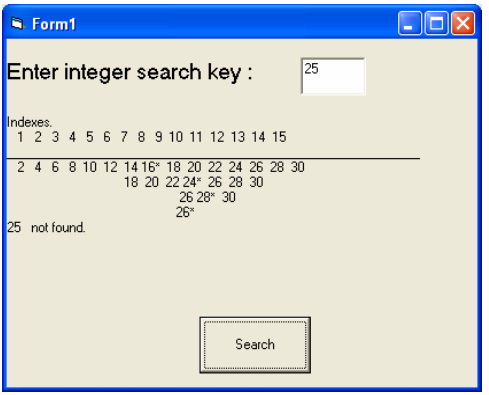
در برنامه زیر از تابع BinarySearch استفاده شده است. اگر Searchkey با عضو وسط (middle) آرایه، برابر نباشد، اندیس low یا high تغییر پیدا کرده و زیر آرایه کوچکتر جستجو می‌شود. اگر مقدار SearchKey کوچکتر از عضو وسط middle باشد، اندیس high با middle-1 مقداردهی می‌شود و عمل جستجو از low تا middle ادامه می‌یابد ولی اگر مقدار SearchKey بزرگتر از عضو وسط باشد، عمل جستجو از عضو middle+1 تا high ادامه پیدا می‌کند. برنامه از آرایه‌ای با ۱۵ عضو استفاده می‌کند. روال PrintHeader اندیس‌های آرایه را چاپ می‌کند و روال PrintRow هر زیر آرایه را در هر بار جستجوی دودویی چاپ خواهد کرد. عضو وسط (middle) در هر زیر آرایه را با علامت ستاره * تعیین می‌کنیم تا مشخص شود که SearchKey با کدامیک از اعضای آرایه در حال مقایسه شدن است.

```
1     'Demonstrating a binary search
2     Option Explicit                 'General declaration
3     Option Base 1                   'General declaration
4     Dim mArray(15) As Integer       'General declaration
5     Dim mLowBound As Integer        'General declaration
6     Dim mUpperBound As Integer      'General declaration

7     Private Sub Form_Load()
8     Dim x As Integer
9     mLowBound = LBound(mArray)
10    mUpperBound = UBound(mArray)
11    'Generate Some array data
12    For x = mLowBound To mUpperBound
13    mArray(x) = 2 * x
```

```
14      Next x
15      End sub

16      Private Sub cmdSearch_Click()
17      Dim x As Integer
18      Call Cls
19      'Print blanks so printing does not
20      'Print behind Label and TextBox
21      For x = 1 To 5
22      Print
23      Next x
24      Call BinarySearch()
25      End sub

26      Private Sub BinarySearch()
27      Dim middle As Integer
28      Dim low As Integer, high As Integer
29      Dim searchKey As Integer
30      low = mLowbound
31      high = mUpperBound
32      Call PrintHeader
33      SearchKey = txtkey.Text
34      Do While (low < = high)
35      Middle = (low + high) \2
36      Call PrintRow(low, middle, high)
37      If (searchKey = mArray(middle)) Then
38      Print "Found" & searchKey & "in"& "index" & middle
39      Exit sub
40      ElseIf searchKey < mArray(middle) Then
41      High =middle –1
42      Else
43      Low = middle + 1
44      End If
45      Loop
46      Print Searchkey & "not found."
47      End Sub
48      Private Sub PrintHeader()
49      Dim x As Integer
50      Print "Indexes."
```

```
51      For x = mLowBound To mUpperBound
52      Print Format$(x, "!@@@@");
53      Next x
54      Print
55      For x = mLowBound To 4 * mUpperBound
56      Print "_";
57      Next x
58      Print
59      End sub

60 Private Sub PrintRow(low As Integer, middle As Integer, High As
        Integer)
61      Dim x As Integer
62      For x = mLowBound To mUpperBound
63      If(x <low Or x > high) Then
64      Print Space$(4);
65      ElseIf (x = middle) Then
66      Print Format$(mArray(x) & "*", "!@@@@");
67      Else
68      Print Format$(mArray(x), "! @@@@");
69      End If
70      Next x
71      Print
72      End Sub
```





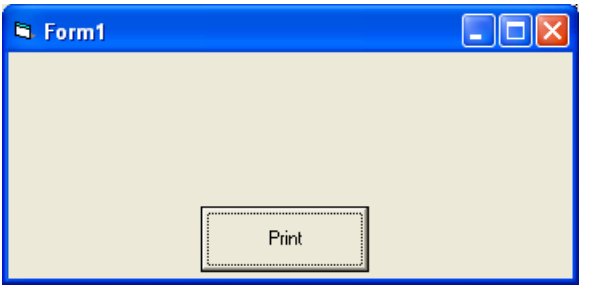
شکل ٦-٥) جستجوی دودویی

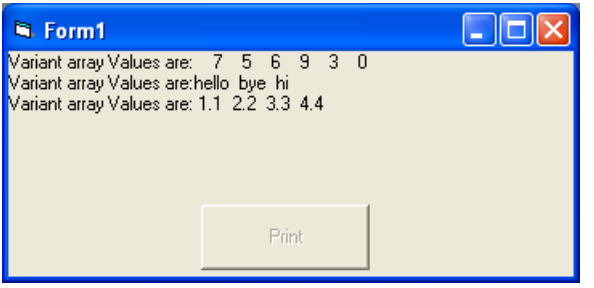
٦-١٢ تابع Array

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

```
1.  'Demonstrating function Array
2.  Option Explicit          'General declaration
3.  Option Base 1            'General declaration
4.  Private sub cmdPrint_Click()
5.  Dim v As Variant, x As Integer
6.  v = Array(7, 5, 6, 9, 3, 0)          'Returens Variant array
7.  Print "Variant array Values are: ";
8.  For x = LBound(v) To UBound(v)
9.  Print Format$(v(x), " @@@ ");
10. Next x
11. Print
12. V = Array("hello" , "bye" , "hi")    'Return variant array
13. Print "Variant array Values are:";
14. For x = LBound(v) To UBound(v)
15. Print v(x) Space$(2) ;
16. Next x
17. Print
18. V = Array(1.1 , 2.2 , 3.3 , 4.4)
```

```
19. Print “Variant array Values are: “;
20. For x = LBound(v) To UBound(v)
21. Print v(x) & Space$(2);
22. Next x
23. cmdPrint.Enabled = False
24. End Sub
```





شکل ۱۱-۳) استفاده از تابع Array()

فصل هفتم

خطایابی و اشکالزدایی برنامه

هدف‌های رفتاری :

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

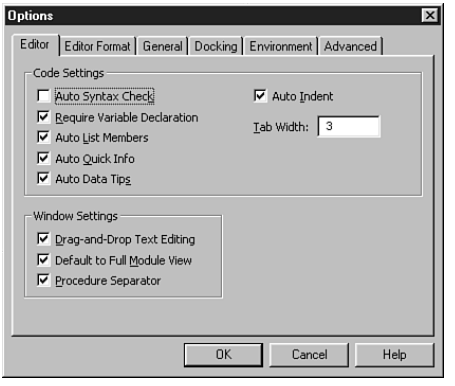
۱. اصول کار با دستور On Error Goto را بیان کرده و در برنامه‌ها آن را به کار ببرد.
۲. نحوه‌ی کار با شیء ERR را شرح داده و در برنامه‌های خود از آن استفاده کند.
۳. با دستور Resume کار کند و اجرای برنامه‌های قطع شده را از سر بگیرد.
۴. برنامه‌های خود را اشکالزدایی کند.

۷-۱ رفع اشکال متغیرهای اعلان نشده با Option Explicit

همزمان با کدنویسی، IDE ویژوال بیسیک خطاهای نحوی که بوجود می‌آیند را اعلام می‌کند (شکل ۷-۱). خطاهای نحوی شامل املاء یا محل قرارگیری کلیدواژه‌ها هستند. به این نوع خطاها می‌توان به سادگی پی برد و رفع کرد.

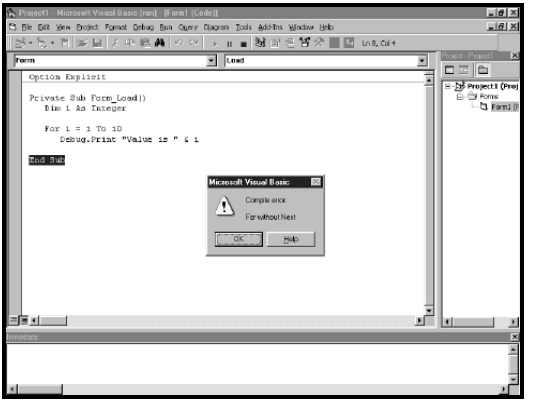
شکل ۷-۱) IDE ویژوال بیسیک دستورات If بدون کلیدواژه Then را هنگام تایپ شناسایی می‌کند. ولی اگر فقط کلیدواژه‌های End-If اشتباه باشند، هنگام کامپایل کد، اعلام خواهد شد.

همزمان با تایپ، VB خطاهای املایی را اعلام می‌کند. در صورتی که می‌خواهید VB کار را قطع نکند و برای هر خطایی، پیغامی نمایش دهد، از منوی Tools گزینه‌ی Options را انتخاب کرده و کادر علامت Auto Syntax Check را پاک کنید (شکل ۲-۵). پاک کردن این کادر علامت سبب می‌شود که کامپایلر از پیدا کردن خطاها در هنگام تایپ، صرفنظر کند.



شکل ۷-۲) غیرفعال کردن Auto Syntax Check سبب می‌شود که پیغام‌ها خطای نحوی را هنگام تایپ مشاهده کنید.

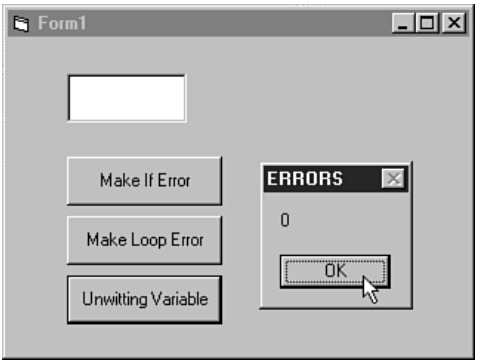
هنگامی که کد را درون IDE اجرا می‌کنید، ویژوال بیسیک خطاهایی مثل نوع اشتباه و بلاک‌های کد کامل نشده را گزارش می‌کند (شکل ۷-۳). فقط در صورتی که Option Explicit تنظیم شده باشد، ویژوال بیسیک کد متغیرهای اعلان نشده را اجرا می‌کند. هنگامی که کلید واژه‌ی Option Explicit را در بخش General یک فرم یا مدول وارد می‌کنید، همه‌ی متغیرهای کد باید با استفاده از کلیدواژه‌های Public، Private، Dim یا Static به طور صریح اعلان شوند.



شکل ۷-۳) IDE هنگام کامپایل کد، یک بلاک حلقه‌ی کامل نشده را گزارش می‌کند.

اشتباهات تایپی ساده منجر به بروز خطاهای بزرگی در کد می‌شوند. به عنوان مثال، در کد زیر متغیری با نام intMyNum اعلان شده است ولی در خط ۶، از intMyNim استفاده شده است. به دلیل اینکه Option Explicit استفاده نشده است، VB به طور خودکار متغیر جدیدی به نام intMyNim ایجاد کرده و آن را با صفر مقداردهی می‌کند. شکل ۴-۵، نتیجه را نشان می‌دهد. اگر Option Explicit استفاده شده بود، IDE هنگام تایپ کد، این خطا را اعلام می‌کرد.

```
01 Private Sub cmdUnWit_Click()
02 Dim intMyNum As Integer
03
04 intMyNum = 2 + 2
05
06 MsgBox CStr(intMyNim)
07 End Sub
```



شکل ۷-٤) اگر از Option Explicit استفاده کرده باشید، پیام خطایی را دریافت خواهید کرد که مشخص می‌کند متغیر intMyNim تعریف نشده است.

۷-۲ بررسی قطعات کد با BreakPoint

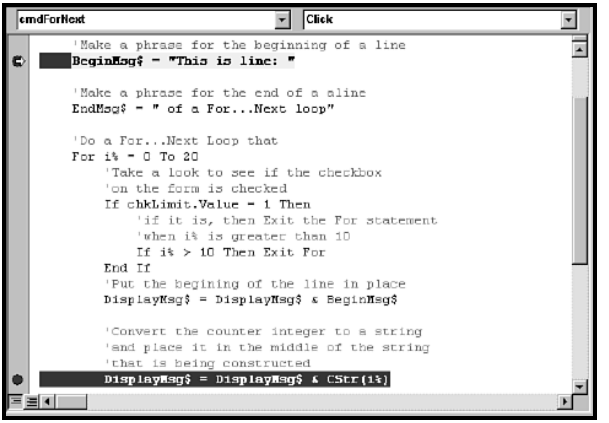
می‌توان کد ویژوال بیسیک را در هر نقطه‌ای از اجرا متوقف کرده و آن را با نقاط قطع (breakpoints) بررسی کرد. یک نقطه‌ی قطع، محلی در کد است که می‌توان کد را در طول اجرا متوقف کرد. نقطه‌ی قطع را می‌توان به چهار روش، تعیین کرد:

- کلیک روی خط کد مورد نظر و فشار دادن کلید F9
- کلیک روی آیکن Breakpoint در نوار ابزار استاندارد
- فعال کردن نقطه قطع از منوی Debug
- کلیک در حاشیه‌ی پنجره‌ی Code

نکته:

برای پاک کردن تمام نقاط قطع در کد، از منوی Debug گزینه‌ی Clear All Breakpoints را انتخاب کنید. همچنین می‌توان تمام نقاط قطع را با فشار دادن کلیدهای Ctrl+Shift+F9 پاک کرد.

هنگامی که یک نقطه قطع را تعیین می‌کنید، توجه کنید که به رنگ قرمز تبدیل می‌شود. هنگامی که کد قطع می‌شود، خط مورد نظر کد به رنگ زرد تبدیل می شود. همچنین یک فلش در حاشیه‌ی چپ پنجره‌ی Code به خط مورد نظر اشاره می‌کند (شکل ۷-۵).



شکل ۷–۵) برای رجوع به نقطه قطع بعدی، کلید F5 را فشار دهید.

همه‌ی اشکالات بر اثر نحو کد به وجود نمی‌آیند و اغلب اشکالات که سبب بروز خطا می‌شوند به دلیل منطق کد یا اشکال طراحی است. پیدا کردن این نوع اشکالات، مشکل است. از نقاط قطع می‌توانید برای محدود کردن کدی که سبب بروز اشکال شده است، استفاده کنید. بعد از تعیین خطی از کد که می‌دانید اشکال رخ داده است، نقطه‌ی قطع را تعیین کرده و ناحیه‌ی بروز خطا را با استفاده از watches جستجو کنید.

۷–۳ نمایش مقادیر متغیرها با Watches

با نگاه دوباره به کد و شکل ۷–۳، مشاهده می‌کنیم که دارای اشکال است و دلیل آن نیز روشن است. فرض کنید که دلیل بروز این خطا را نمی‌دانید، فرصت مناسبی است که از نقاط قطع و watches استفاده کنید.

۱– نقطه‌ی قطع را برای خطی از کد که کادر پیام را نشان می‌دهد، تعین کنید.

۲– برنامه را اجرا کنید.

۳– اشاره‌گر ماوس را روی متغیری که می‌خواهید مقدار آن را مشاهده کنید، درگ نمایید و چند لحظه نگه دارید. پنجره‌ی کوچکی با مقدار متغیر ظاهر می شود.

در پنجره‌ی watches می‌توان تغییر مقادیر یک متغیر را مشاهده کرد. برای اضافه کردن متغیرهای دیگری به پنجره‌ی watch مراحل زیر را انجام دهید:

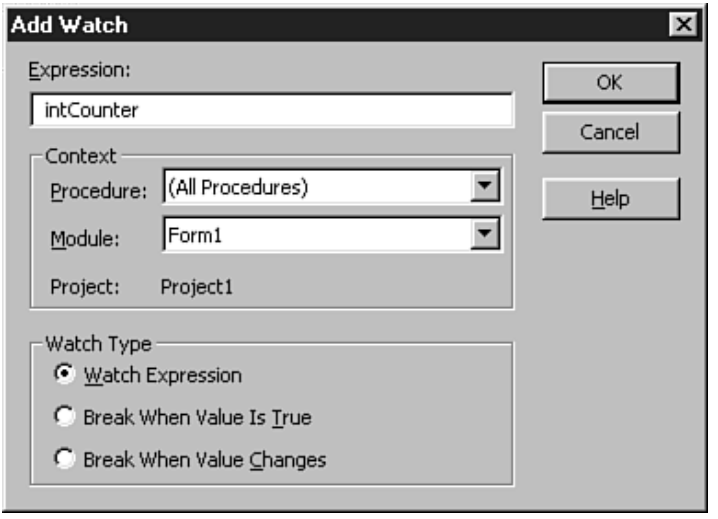
۱- یک یا دو نقطه قطع برای متغیرهای مورد نظر تعیین کرده و کلید F5 را فشار دهید.

۲- درهر نقطه قطع، متغیر موردنظر را های‌لایت (مشخص) کنید.

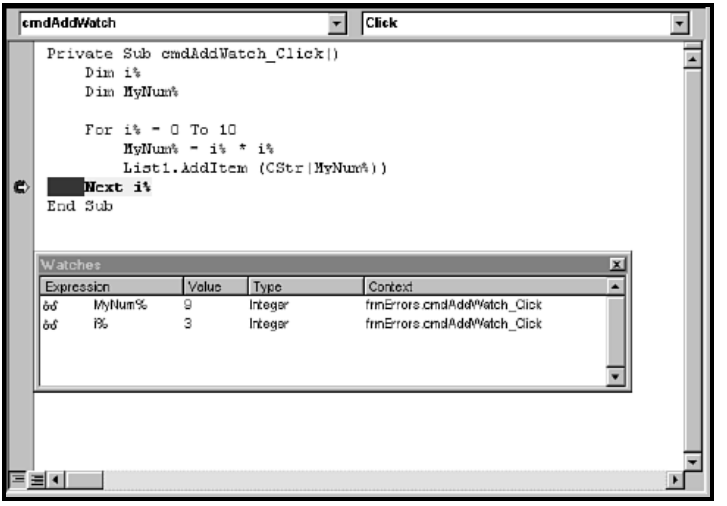
۳- کلیک راست کرده و Add watch را از منوی میانبر انتخاب کنید.

۴- تنظیمات مناسب را در کادر محاوره‌ای Add watch انجام دهید (شکل ۷-۶) و سپس روی Ok کلیک کنید.

۵- برای نمایش پنجره‌ی watches (شکل ۷-۷)، از منوی View گزینه‌ی watch window را انتخاب کنید. هنگام اضافه کردن یک watches نیز این پنجره ظاهر می‌شود.



شکل ۷-٦) کادر محاوره‌ای Add Watch یک ابزار قوی و دارای قابلیت انعطاف است.



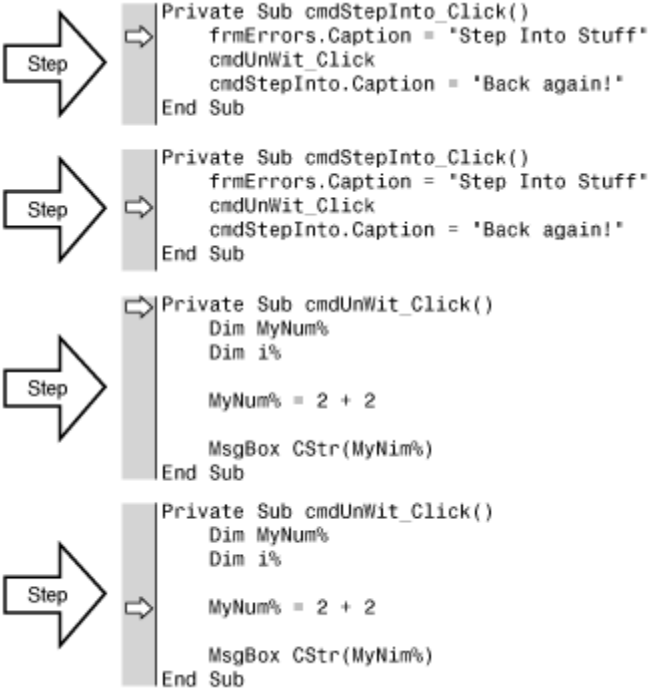
شکل ۷-۷) در حالت Break، مطمئن باشید در محلی از کد که در میدان دید متغیرهای پنجره‌ی watches است، قرار دارید.

پی‌گیری چندین متغیر مستقل که به طور پیوسته تغییر می‌یابند، می‌تواند خیلی مشکل باشد. پنجره‌ی watches ابزار مؤثری برای پی‌گیری مقادیر در عملیات پویا مثل حلقه‌ها یا آرایه‌ها است.

۷-۳-۱ بررسی خط به خط کد با Step Into و Step Over

می‌توان کد یک برنامه را به صورت خط به خط اجرا و بررسی کرد تا اگر خطایی در هر خط وجود دارد را متوجه شده و رفع کرد. اجرای گام به گام به دو روش انجام می‌شود: Step into و Step over.

هنگام استفاده از روش step into، اگر در خطی از کد یک روال دیگری فراخوانی شود، اجرای گام به گام وارد روال جدید خواهد شد (شکل ۷-۸).



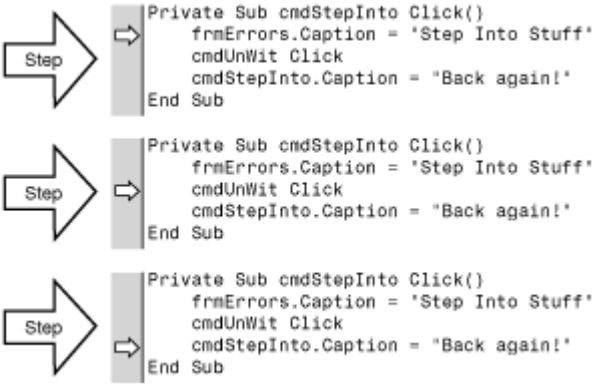
شکل ۷–۸) می‌توان با فشار دادن کلید F8 یا انتخاب گزینه‌ی step Into از منوی Debug کد را به صورت گام به گام اجرا کرد.

اگر کد را به صورت اجرای گام به گام شروع کنید، ممکن است بعضی مواقع از اینکه هیچ چیزی رخ نداده است، متعجب شوید. در صورتی که هیچ کدی در روال رویداد Form_Load() نداشته باشید، هیچ رویدادی اجرا نمی‌شود. به خاطر داشته باشید که ویندوز یک سیستم عامل رویدادگرا است و رویدادی باید برای کد رخ دهد تا اجرا شود. تنها کدی که به صورت پیش فرض شروع به کار می‌کند، کد مربوط به روال‌های Form_Initialize()، Form_Load() یا Sub Main() است. اگر هیچ کاری داخل این روال‌ها نباشد، برنامه تا زمانی که رویدادی رخ ندهد، هیچ کاری انجام نمی‌دهد.

برای اجرای گام به گام کد، بهتر است یک نقطه‌ی قطع در خط مورد نظر کد قرار داده و سپس کد را به صورت عادی اجرا کنید.

هنگامی که اجرا به نقطه قطع برسد، اجرای گام به گام شروع خواهد شد. برای خـروج سـریع از روالی که به صورت گام به گام اجرا می‌شود، کلیدهای Ctrl+Shift+F8 را فـشار داده یـا از منـوی Debug گزینه‌ی Step out را انتخاب کنید.

هنگامی که از روش step over برای اجرای گام به گام کد استفاده می‌کنید، در صورت برخورد با خطی که روال دیگری را فراخوانی می‌کند، وارد روال فراخوانی شده نخواهید شد و فقط این خط کد به صورت یک خط منفرد اجرا خواهد شد (شکل ۷-۹).



شکل ۷-۹) در صورتی که نمی‌خواهید کد درون یک روال رویداد مثل cmdUnWit_Click() را اشکالزدایی کنید، کلیدهای Shift+F8 را فشار دهید تا از روش step over استفاده کنید.

ممکن است اجرای این تکنیک اشکالزدایی از طریق نوارابزار Debug سریعتر و سـاده‌تـر باشـد (شکل ۷-۱۰).

شکل ۷-۱۰) نوارابزار Debug می تواند یک نوار ابزار شناور باشد یا می‌توان آن را به ناحیه‌ی نوارابزارها درگ کرد تا ثابت شود.

۷-۳-۲ متوقف کردن خطوط انتخاب شده با Run to Cursor

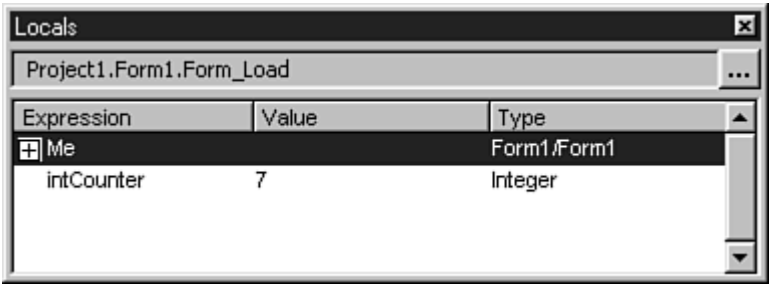
هر بار که نقطه‌ی قطه‌ای را تعیین کنید، تا زمانی که به کد برنگشته و آن نقطه‌ی قطع یـا همـه‌ی نقاط قطع را با انتخـاب گزینـه‌ی Clear (Ctrl+Shift+F9) All Breakpoints از منـوی Debug پاک نکرده باشید، در کد باقی می‌ماند. تعداد زیاد نقاط قطع، سبب کنـد شـدن سـرعت اشـکالزدایی می‌شود. این کار را می‌توان با استفاده از Run to Cursor ساده‌تر انجام داد تا کد را در نقاط تعیـین شده قطع کند.

روی خطی از کد که می‌خواهید متوقف شود، کلیک کرده و سپس کلیـدهای Ctrl+F8 را فـشار دهید یا از منوی Debug گزینه‌ی Run to Corsor را انتخاب کنید. کلید F5 را فشار دهیـد تـا کـد اجرا شود. IDE ویژوال بیسیک، اجرای کد را در خطوطی که کلیک کرده‌اید، قطع می کند.

۷-٤ استفاده از ابزارهای اشکالزدایی پیشرفته

علاوه بر مشاهده‌ی مقادیر متغیرها و تکنیک‌های اجرای گام به گام، می‌تـوان از ابزارهـای نـشان داده شده در شکل‌های ۷-۱۱ تا ۷-۱۴ استفاده کرد. ویژوال بیسیک این ابزارها را بـرای اشـکالزدایی پیشرفته ارایه می‌کند.

پنجره‌ی Locals (شکل ۷-۱۱) روش ساده‌ای برای مشاهده‌ی تمام متغیرهای موجود در میـدان دید جاری است. متغیرها به همراه مقادیرشان در این پنجره فهرست می شـوند. شـیءها دارای یـک علامت جمع هستند که می‌توانید روی آن کلیک کنید تا مشخصه‌هـای شـیء را مـشاهده کنیـد. اگـر مشخصه، خود شیء دیگری باشد، علامت جمع دیگـری را مـشاهده خواهیـد کـرد. در ایـن پنجـره می‌توان تمام متغیرها را به طور همزمان و بدون کلیک کردن روی هر کدام، مشاهده کرد.



شکل ۷-۱۱) پنجره‌ی Locals تمام متغیرهای موجود در میدان دید جاری را به همراه مقادیرشان نشان می‌دهد. برای دسترسی به این پنجره، از منوی View گزینه‌ی Locals window را انتخاب کنید.

از پنجره‌ی Immediate (شکل ۷-۱۲) می‌توان برای آزمایش خطوطی از کد بدون اجرای برنامه استفاده کرد. برای آزمایش این پنجره، کد زیر رادر آن وارد کنید:

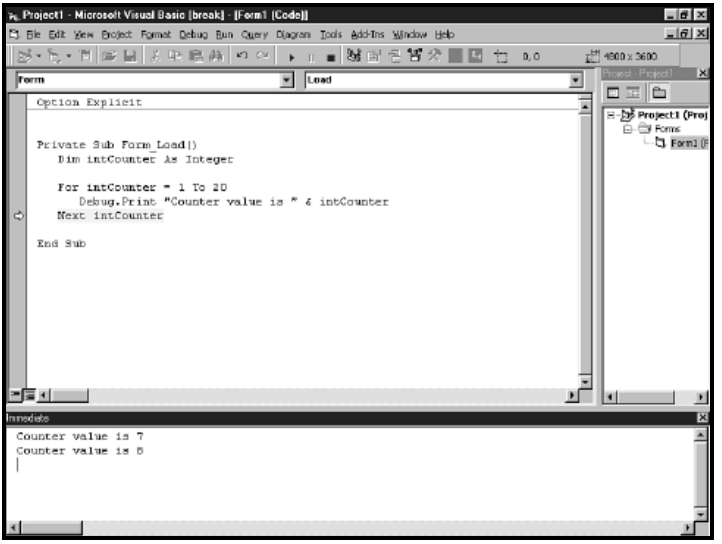
```
Print 2*3
```

دستور Print مقدار ۶ را ارایه می‌دهد.

پنجره‌ی Immediate برای دستورهای یک خطی، مناسب است. دراین پنجره نمی‌توان متغیر جدیدی را اعلان کرد، ولی می‌توان از هر متغیری که در میدان دید است، استفاده کرد. به عنوان مثال، اگر برنامه در یک روالی قطع شده است که متغیر int Counter در آن تعریف شده است، می‌توان خط زیر را در پنجره‌ی Immediate تایپ کرده و مقدار آن را مشاهده کرد:

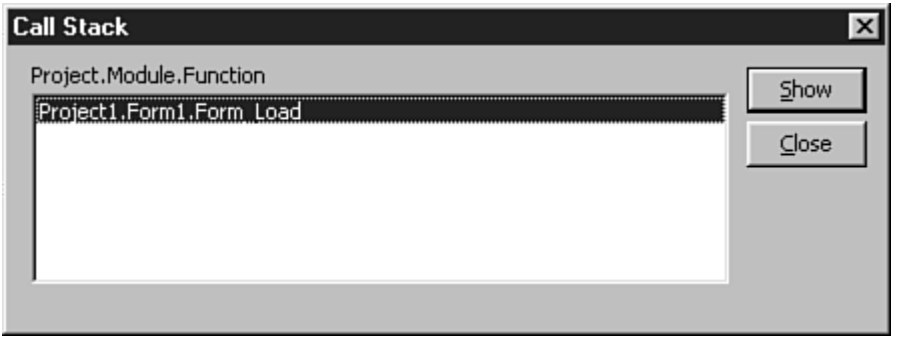
```
Print intCounter
```

همچنین می‌توان مقادیر را در پنجره ی Immediate تغییر داده و متدهایی را روی شیءها اجرا کرد. هر چیزی که نیاز به یک خط کد دارد را در این پنجره می توان اجرا کرد.



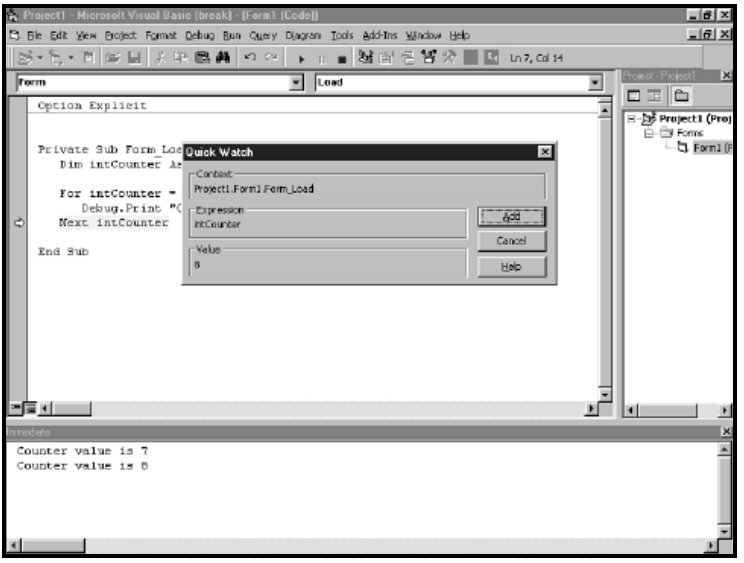
شکل ۷-۱۲) پنجره‌ی Immediate امکان تایپ کد و اجرای آن را با فشار دادن کلید Enter فراهم می‌کند.

کادر محاوره‌ای Call stack در صورتی که از چندین روال و رویداد استفاده می‌کنید، مفید است. این کادر محاوره‌ای (شکل ۷-۱۳)، تمام روال‌ها و توابع فعال را نشان می‌دهد. عنصر لیست شده در بالای کادر محاوره‌ای، روال جاری است و خط زیر آن، خط فراخوانی آن است و الی آخر. این کادر محاوره‌ای با کلیک کردن روی دکمه‌ی Call Stack از نوار ابزار Debug یا با فشار دادن کلیدهای Ctrl+L باز می‌شود.



شکل ۷-۱۳) کادر محاوره‌ای Call stack تمام روال‌های فعال را نشان می‌دهد. این عمل فقط در حالت قطع ممکن است.

برای نشان دادن کادر محاوره‌ای Quick watch (شکل ۷-۱۴)، یک قطع را در کد قرار دهید، روی متغیری کلیک کنید یا عبارتی را مشخص (های‌لایت) کرده و از منوی Debug گزینه‌ی Quick watch را انتخاب کنید.



شکل ۷-۱٤) همچنین می‌توان کادر محاوره‌ای Quick watch را با فشار دادن کلیدهای Shift+F9 باز کرد.

۷-۵ کاربرد Find and Replace

در کتاب بسته‌های نرم‌افزاری ۱ و در بخش آموزش Word با ابزاری به نام Find and Replace آشنا شدید که رشته‌ای را در متن سند جستجو کرده و رشته‌ی دیگری را جایگزین آن می‌کند. در پنجره‌ی کد ویژوال بیسیک نیز می‌توان از این ویژگی استفاده کرد که چگونگی انجام این کار را قبلاً آموخته‌اید.

۷-٦ طراحی برنامه‌های کاربردی برای اشکالزدایی

اگر برنامه‌ی دارید که نیاز به برنامه‌نویسی زیادی دارد، روش‌های اشکالزدایی که تا به اینجا آموختید، مناسب نخواهند بود. به جای استفاده از نقاط قطع و watch از اشکالزدایی شرطی استفاده کنید (مثال زیر را در نظر بگیرید). کد زیر، مجموع اعداد ۱ تا ۲۰ را به دست آورده و نتیجه را نمایش می‌دهد.

```
01 Private Sub Form_Load()
02 Dim intCounter As Integer
03 Dim intSum As Integer
```

```
04 intSum = 0
05
06 For intCounter = 1 To 20
07 intSum = intSum + intCounter
08 Next intCounter
09 MsgBox "Sum is " & intSum & "."
10
11 End Sub
```

فرض کنید که از درست کار کردن این کد مطمئن نیستید. یک روش ساده برای آزمایش کد، اضافه کردن دستور Debug.print بین خطوط ۷ و ۸ کد فوق به صورت زیر است:

```
Debug.print “value:” i &”, New sum:” & intSum
```

هنگام اجرای برنامه نتیجه‌ی دستور Debug.print در پنجره‌ی Immediate نمایش داده خواهد شد. هنگام کامپایل این برنامه دستورهای Debug.print کاری انجام نمی‌دهند، زیرا پنجره‌ی Immediate برای چاپ وجود ندارد. به همین دلیل، بهتر است از روش بهتری به نام اشکالزدایی شرطی استفاده کنید. برای انجام این کار، کد فوق را به صورت زیر اصلاح کنید:

```
01 Private Sub Form_Load()
02 Dim intCounter As Integer
03 Dim intSum As Integer
04 intSum = 0
05
06 For intCounter = 1 To 20
07 intSum = intSum + intCounter
08
09 #If DEBUG_ON Then
10 Debug.Print "Value: " & intCounter & ", New Sum: " _
& intSum
11 #End If
12
13 Next intCounter
14 MsgBox "Sum is " & intSum & "."
15
16 End Sub
```

برای اینکه این کد به طور مناسب کار کند، خط زیر را به بخش اعلان فرم، اضافه کنید:

```
#Const DEBUG_ON = True
```

۷-۷ ایجاد یک مدیر خطا

اگر نرم‌افزاری داشته باشید که در طی اجرا خراب می‌شود، احتمالاً برنامه‌نویس آن بخشی از برنامه را فراموش کرده است: مدیریت خطا. هر برنامه‌ای نیاز دارد که به خطاهایی که در درون خودش رخ می‌دهند، پاسخ دهد. در این قسمت، مشاهده می‌کنید که چگونه از قابلیت‌های مدیریت خطا در ویژوال بیسیک استفاده کنید. همچنین به عنوان مثال، یک مدیریت خطای ساده‌ای را ایجاد خواهیم کرد که می‌تواند برای برنامه‌های کاربردی، بسط پیدا کند:

۱- پروژه‌ی جدیدی را شروع کنید.

۲- در روال رویداد Form-Load، کد زیر را اضافه کنید:

```
Dim intTest As Integer
IntTest = 100/0
```

۳- پروژه را اجرا کنید.

به دلیل اینکه تقسیم بر صفر، خطاست، ویژوال بیسیک برنامه را قطع کرده و پیام خطایی را تولید می‌کند. چند روش برای رفع خطاهای زمان اجرا وجود دارد. اولین روش، استفاده از دستور On Error Resume Next است. این دستور به عنوان بخشی از مدیریت خطا سبب می‌شود که ویژوال بیسیک از خطا صرفنظر کرده و کار را ادامه دهد. این نوع مدیریت خطا هنگامی مفید است که نیازی به رفع خطا نداشته باشید. به عنوان مثال، اگر خطایی هنگام خروج از برنامه رخ دهد، مدیریت خطا نیازی به رفع خطا ندارد زیرا به هر روشی می‌توان از برنامه خارج شد. کد فوق را به صورت زیر اصلاح کنید:

```
Dim intTest as Integer
On Error Resume Next
Inttest = 100/0
Debug.print “program is past the error.”
```

به دلیل اینکه مدیریت خطا ندارید، ویژوال بیسیک خطا را تشخیص داده و از خطی که سبب بروز خطا شده است، صرفنظر می‌کند. در پنجره ی Immediate، نتیجه‌ی دستور Print را مشاهده خواهید کرد.

دراین حالت، مدیریت خطا از خرابی برنامه جلوگیری نمی‌کند. زیرا عدم رفع خطا سبب خواهد شد که خطاهای دیگری در برنامه رخ دهند. اگر در ادامه‌ی برنامه نیاز به مقدار متغیر intTest داشته باشید، ممکن است کار نکند و خطاهای بزرگتری بوجود آیند که خطاهای پشت سرهم (cascading error) نامیده می‌شوند.

خطاهای پشت سرهم مانند خراب شدن یک اتومبیل در بزرگراه است که به دلیل عدم کنترل اتومبیل‌های پشت سر آن، با اتومبیل خراب تصادف کرده و در نهایت سبب بسته شدن بزرگراه می‌شوند.

مثال ۱-۷)

این مثال به کاربران امکان باز کردن درایو CD-ROM را می‌دهد:

```
01 Private Sub Form_Load()
02   Dim intTest As Integer
03   Dim intRet As Integer
04   On Error GoTo EH
05   intTest = 5 / 0
06   Exit Sub
07
08 EH:
09   If Err.Number = 11 Then
10     MsgBox "Division by zero error occurred.", _
        vbCritical
11     End
12   ElseIf Err.Number = 71 Then
13     intRet = MsgBox("That drive is not ready.", _
        vbExclamation + vbAbortRetryIgnore)
14     Select Case intRet
15     Case vbRetry
16       Resume
17     Case vbAbort
18       End
19     Case vbIgnore
20       Resume Next
21     End Select
22   `
```

```
23  ` more conditions follow
24  `
25  End If
26 End Sub
```

ابتدا دستور On Error از دستور GoTo استفاده می‌کند تا در صورت بروز خطا به کدام خط منتقل شود. در این حالت، کنترل به بلاکی از کد که با برچسب EH مشخص شده است، منتقل می‌شود. برچسب‌ها معمولاً برای مدیریت خطا استفاده می‌شوند. این بخش از مدیریت خطا (خطوط ۹ تا ۲۵) مشخصه‌ی Number شیء Err را بررسی می‌کند. این عدد تعیین می‌کند که چه خطایی رخ داده است.

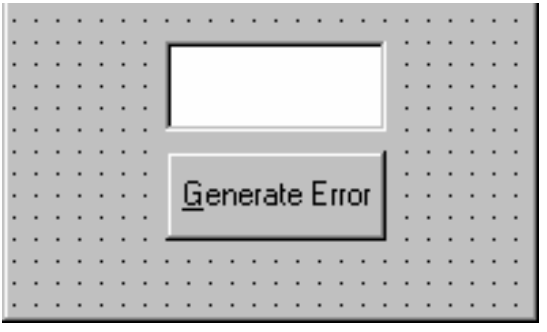
مثال ۷-۲: تولید خطا)

۱. پروژه‌ی جدیدی را شروع کنید. یک کادر متن و یک دکمه‌ی فرمان اضافه کنید.

۲. مشخصه‌های فرم و کنترل‌ها را به صورت زیر تنظیم کنید:

Form1:		
	BorderStyle	1-Fixed Single
	Caption	Error Generator
	Name	frmError
Command1:		
	Caption	Generate Error
	Default	True
	Name	cmdGenError
Text1:		
	Name	txtError
	Text	[Blank]

فرم شبیه شکل زیر خواهد بود:



۳. کد زیر را برای این فرم، بنویسید:

```
Private Sub cmdGenError_Click()
    On Error GoTo HandleErrors
    Err.Raise Val(txtError.Text)
    Err.Clear
Exit Sub
HandleErrors:
Select Case MsgBox(Error(Err.Number), vbCritical + vbAbortRetryIgnore,
"Error Number" + Str(Err.Number))
 Case vbAbort
  Resume ExitLine
 Case vbRetry
  Resume
 Case vbIgnore
  Resume Next
End Select
ExitLine:
Exit Sub
End Sub
```

شماره و پیغام خطاهای تولید شده در این برنامه به صورت زیر خواهد بود:

Error Number	**Error Description**
6	Overflow
9	Subscript out of range
11	Division by zero
13	Type mismatch
16	Expression too complex
20	Resume without error
52	Bad file name or number
53	File not found
55	File already open
61	Disk full
70	Permission denied
92	For loop not initialized

خودآزمایی

۱- برنامه‌ای بنویسید که لیستی از انواع خطاها که غالباً رخ می‌دهند را ارایه کند.

۲- توضیح دهید که اگر روالی یا تابعی فاقد قسمت رسیدگی بـه خطـا باشـد و در آن روال یـا تابع خطایی بوجود آید، چه اتفاقی رخ می‌دهد.

۳- چرا رسیدگی به خطاها قبل از عبارات Exit Sub یا Exit Function قرار می‌گیرند؟

فصل هشتم

تولید بسته‌های نرم‌افزاری

هدف‌های رفتاری :

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

۵. ویژگیهای فایل‌های با دسترسی تصادفی را بیان کند.
۶. اصول تعریف رکوردها را بیان کرده و انجام دهد.
۷. اصول باز کردن فایل‌ها با دستور Open و در حالات مختلف را بیان نماید.
۸. فایل‌های با دسترسی تصادفی را به صورت ترتیبی بخواند.
۹. چگونگی نوشتن، خواندن و بستن فایل را بیان کند و انجام دهد.

۸-۱ کار کردن با اطلاعات نسخه

یکی از اولین موارد حرفه‌ای که می‌توان بـه برنامـه‌ی کـاربردی اضـافه کـرد، ارایـه‌ی اطلاعـات عمومی متداول برنامه‌ی کاربردی است که شامل نام شرکت، شماره نسخه و سـایر اطلاعـات مـشابه است. ویژوال بیسیک امکان ذخیره‌ی تمام این اطلاعـات را بـا اسـتفاده از شـیء App (یـک شـیء تعریف شده در ویژوال بیسیک که نیاز به ایجاد آن برای برنامه‌ی کاربردی ندارید) فراهم می‌کند.

اغلب مشخصه‌های شیء App برای ارایه اطلاعات عمومی برنامه‌ی کاربردی، مورد استفاده قرار می‌گیرند. جدول ۸-۱ مشخصه‌های متداول را نشان می‌دهد.

جدول ۸-۱) مشخصه‌های متداول شیء App

مشخصه	شرح
Comments	رشته‌ای را برمی‌گرداند کـه شـامل توضـیحاتی دربـاره‌ی برنامـه‌ی

	کاربردی است. در زمان اجرا فقط خواندنی است.
Company Name	نام شرکت یا تولیدکننده‌ی برنامه‌ی کاربردی را برمی‌گرداند. در زمان اجرا فقط خواندنی است.
EXEName	نام فایل اجرایی را بدون پسوند برمی‌گرداند (فقط خواندنی).
FileDescription	رشته‌ای که به طور خلاصه هدف برنامه‌ی کاربردی را شرح می‌دهد. در زمان اجرا فقط خواندنی است.
HelpFile	فایل راهنمای مربوط به برنامه‌ی کاربردی را تعیین می‌کند. در زمان اجرا خواندنی/نوشتنی است.
LegalCopyright	رشته‌ی حق کپی رایت را برمی‌گرداند. از Character Map برای افزودن نمادهای ویژه دراین کادر استفاده کنید. در زمان اجرا فقط خواندنی است.
LegalTrademarks	اطلاعات علامت تجاری را در صورت نیاز برمی‌گرداند. از Character Map برای اضافه کردن نمادهای ویژه استفاده کنید. در زمان اجرا فقط خواندنی است.
Major	شماره‌ی اصلی نسخه را برمی‌گرداند (به عنوان مثال، ۴ در ۴,۳). در زمان اجرا فقط خواندنی است.
Minor	شماره‌ی فرعی نسخه را برمی‌گرداند (به عنوان مثال، ۳ در ۴,۳). در زمان اجرا فقط خواندنی است.
Path	فهرستی که برنامه‌ی کاربردی در آن قرار دارد را برمی‌گرداند. در زمان اجرا فقط خواندنی است.
PrevInstance	اگر نمونه‌ای از برنامه‌ی کاربردی در حال اجرا باشد، مقداری را برمی‌گرداند. در زمان اجرا فقط خواندنی است.
Product Name	نام محصول برنامه‌ی کاربردی را برمی‌گرداند. در زمان اجرا فقط خواندنی است.

Revision	شماره‌ی بازنگری برنامه‌ی کاربردی را برمی‌گرداند. در زمان اجـرا فقط خواندنی است.

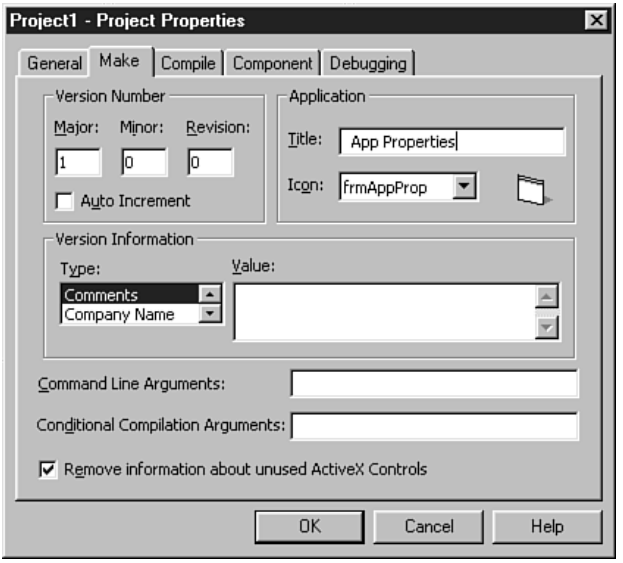
می‌توان از این مشخصه‌ها برای ارایه اطلاعات مهم درباره‌ی برنامه‌ی کاربردی که به کار می‌برید، استفاده کرد. این مشخصه‌ها در کادر محاوره‌ای Project Properties تعیین می‌شوند (شکل ۸-۱).

می‌توان مقادیر این مشخصه‌ها را در زمان اجرا از داخل کد خواند. همچنین می‌تـوان مقـداری را برای مشخصه‌های اطلاعات نسخه‌ی شیء App تعیین کرد که انجام این کار را با کلیک راست روی فایل اجرایی و انتخاب Properties ممکن است (شکل ۸-۲).

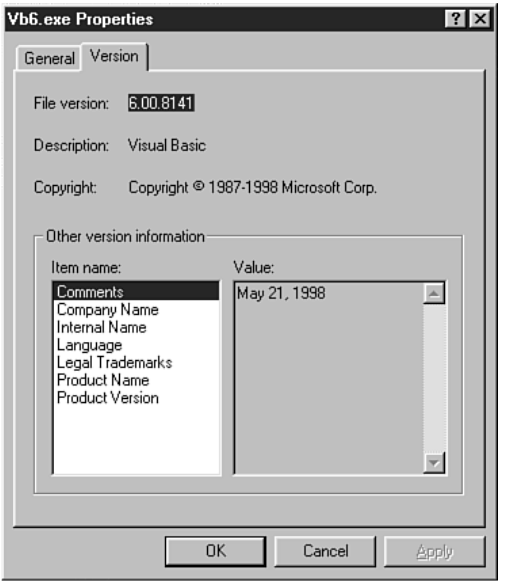
نکته:

مقادیر مشخصه‌های اطلاعات نسخه‌ی شیء App درون قالب دودویی فایل اجرایی اضافه می‌شـوند و در زمان اجرا نمی‌توان آنها را تغییر داد.

```
01 Private Sub cmdCopyright_Click()
02 lblMain.Caption = App.LegalCopyright
03 End Sub
04
05 Private Sub cmdPath_Click()
06 lblMain.Caption = App.Path
07 End Sub
08
09 Private Sub cmdProductName_Click()
10 lblMain.Caption = App.ProductName
11 End Sub
12
13 Private Sub cmdVersionNum_Click()
14 Dim strVerNum
15
16 strVerNum = CStr(App.Major) & "." _
& CStr(App.Minor) & "." _
& CStr(App.Revision)
17
18 lblMain.Caption = strVerNum
19 End Sub
```



شکل ۸-۱) برای دسترسی به کادر محاوره‌ای Project Properties، گزینه‌ی ProjectName Properties (نام پروژه) را از منوی Project انتخاب کنید.



شکل ۸-۲) برای مشاهده‌ی مشخصه‌های نسخه، زبانه‌ی Version را از Windows Explorer انتخاب کنید.

۸-۲ کامپایل کردن پروژه

بعد از تعیین مقادیر مشخصه‌های شیء App پروژه، می‌توان کد را کامپایل کرد. پروژه‌هایی که تا این مرحله از درس ایجاد کرده‌اید، دارای متن و فایل‌های گرافیکی بودند که با IDE ویژوال بیسیک بوجود می‌آمدند. اکنون زمان آن است که این فایل‌ها را به یک فایل اجرایی تبدیل کنید که به طور مستقل از IDE اجرا خواهد شد. این فرایند را **کامپایل کردن کد** یا **ساخت فایل اجرایی** می‌نامند.

ویژوال بیسیک از دو قالب برای کامپایل کد پشتیبانی می‌کند: P-code یا native code. هنگامی که کد را به P-code کامپایل می‌کنید، فایل اجرایی بوجود آمده به صورت کد مفسری اجرا می‌شود.

نکته:

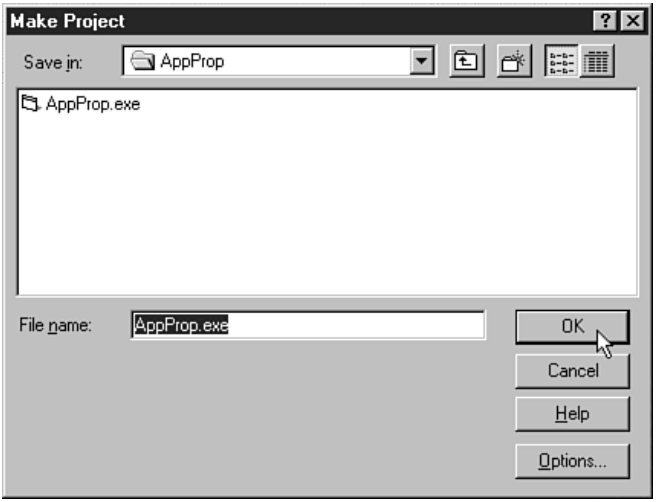
اندازه‌ی فایل اجرایی native code نسبت به P-code بزرگتر است. بنابراین، اگر می‌خواهید کوچکترین فایل اجرایی را ارایه دهید، باید از P-code استفاده کنید. ولی اگر می‌خواهید که سریعترین را ارایه دهید، باید native code رابوجود آورید.

اگر کد را به صورت native code کامپایل کنید، فایل‌های پروژه به کد دودویی کارآمدتری تبدیل می‌شوند که از تمام قابلیت‌های پردازنده استفاده می‌کند. این کد سریعتر اجرا خواهد شد. ولی native code هنوز هم به DLLهای زمان اجرا نیاز دارد (تنها تفاوت این است که DLLها به وسیله‌ی EXE به طور متفاوت مورد دسترسی و استفاده واقع می‌شوند).

۸-۲-۱ کامپایل کد به Standard EXE

۱- پروژه‌ای که می‌خواهید کامپایل کنید را باز کنید.

۲- از منوی File گزینه‌ی *ProjectName.exe* Make را انتخاب کنید. کادر محاوره‌ای Make Project ظاهر می‌شود (شکل ۳-۸).

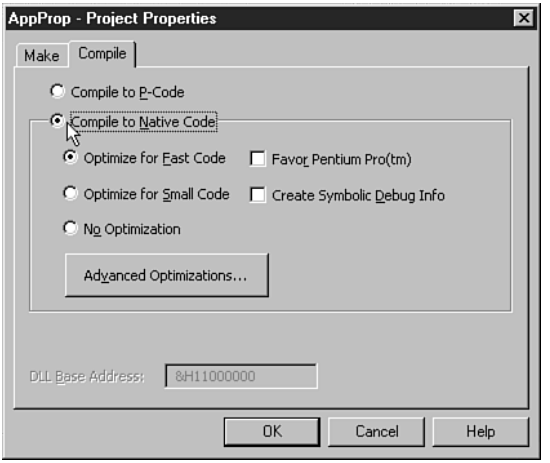


شکل ۸-۳) نام فایل اجرایی را در این کادر محاوره‌ای وارد کنید. برای تغییر بعضی از مشخصه‌های شیء App روی دکمه‌ی Options کلیک کنید.

۳- در صورتی که می‌خواهید نام فایل اجرایی را در کادر متن File Name تغییر دهید.

۴- روی دکمه‌ی Options کلیک کنید تا کادرمحاوره‌ای Properties باز شود. در زبانه‌ی Compile یکی از دو روش کامپایل P-code یا Native code را انتخاب کنید.

۵- در کادرهای محاوره‌ای Project Properties و Make Project روی Ok کلیک کنید تا کد کامپایل شود.



شکل ۸-٤) می‌توان چندین گزینه برای کامپایل Native code انتخاب کرد. به طور کلی، انتخاب کد سریعتر، فایل اجرایی بزرگتری را ایجاد می‌کند.

کامل کردن این فرایند، یک فایل اجرایی را تولید می‌کند که می‌توان خارج از IDE ویژوال بیسیک اجرا کرد. با این وجود برنامه هنوز برای ارایه آماده نیست. برای تحویل برنامه‌ی کاربردی، نیاز به اجرای Application Setup Wizard برای فایل اجرایی دارید که امکان اجرا روی سیستمی که ویژوال بیسیک نصب نشده است را فراهم می‌کند.

۸-۳ کاربرد Package and Deployyment Wizard

ازاین ابزار می‌توان برای ایجاد بسته‌های نرم‌افزاری که نصب می‌شوند (Setup) از هر نوع برنامه‌ی کاربردی در ویژوال بیسیک استفاده کرد. برای انجام این کار، Package and Deployment Wizard را شروع کنید (شکل ۸-۵). این ابزار باید در زیر منوی Visual Basic از منوی Start لیست شده باشد، زیرا به طور پیش‌فرض، نصب می‌شود. در صورتی که نیست، باید آن را روی سیستم نصب کنید.

شکل ۸-۵) ابزار Package and Deployment Wizard

مراحل ایجاد یک بسته‌ی نرم‌افزاری که نصب می‌شود، به صورت زیر است:

۱- فایل پروژه را در بالای کادرمحاوره‌ای انتخاب کنید.

نکته:

برای این مراحل، می‌توانید از هر پروژه‌ای که در ویژوال بیسیک دارید یا از نمونه‌های خـود ویـژوال بیسیک استفاده کنید. ما از پروژه‌ی ATM موجود در ویژوال بیسیک استفاده کرده‌ایم.

۲- برای ساخت یک برنامه‌ی نصب (setup)، روی دکمه‌ی Package کلیک کنید.

۳- در صورتی که برنامه‌ی کاربردی را کامپایل نکرده باشید، ویزارد از شـما کامپایـل برنامـه را سوال می‌گند (سوال ۸-۶). برای ادامه روی دکمه‌ی Compile کلیک کنید.

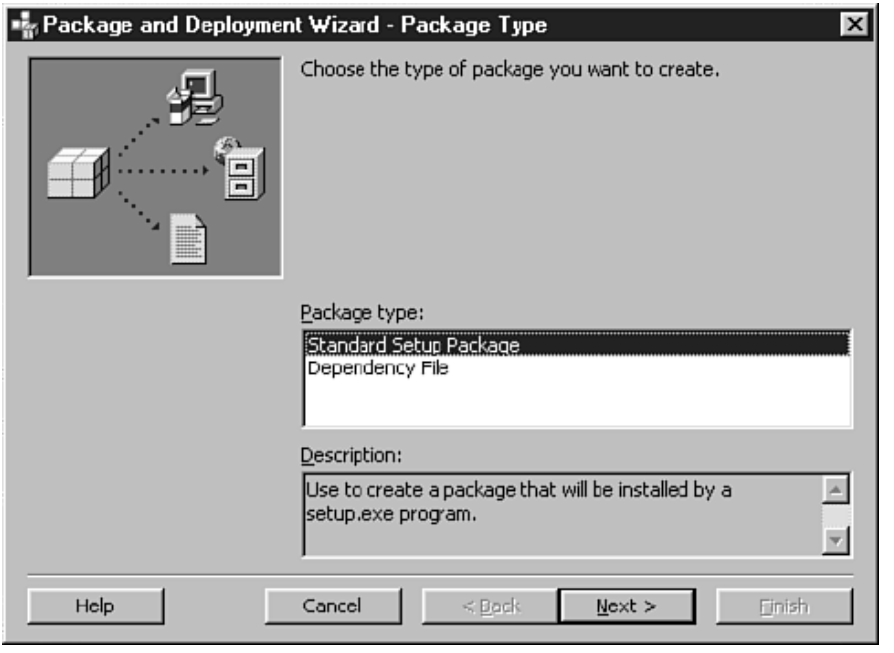
شکل ۸-٦) ویزارد نیاز به فایل اجرایی در داخل برنامه‌ی Setup دارد، بنابراین به طور خودکار فایل پروژه‌ی برنامه‌ی کاربردی را کامپایل می‌کند.

۴- بعد از کامپایل برنامه‌ی کاربردی، از شما خواهد پرسید که چه نوع بسته‌ی نرم‌افزاری می‌خواهید (شکل ۸-۷). گزینه‌ی Standard Setup Package را انتخاب کرده و روی Ok کلیک کنید.

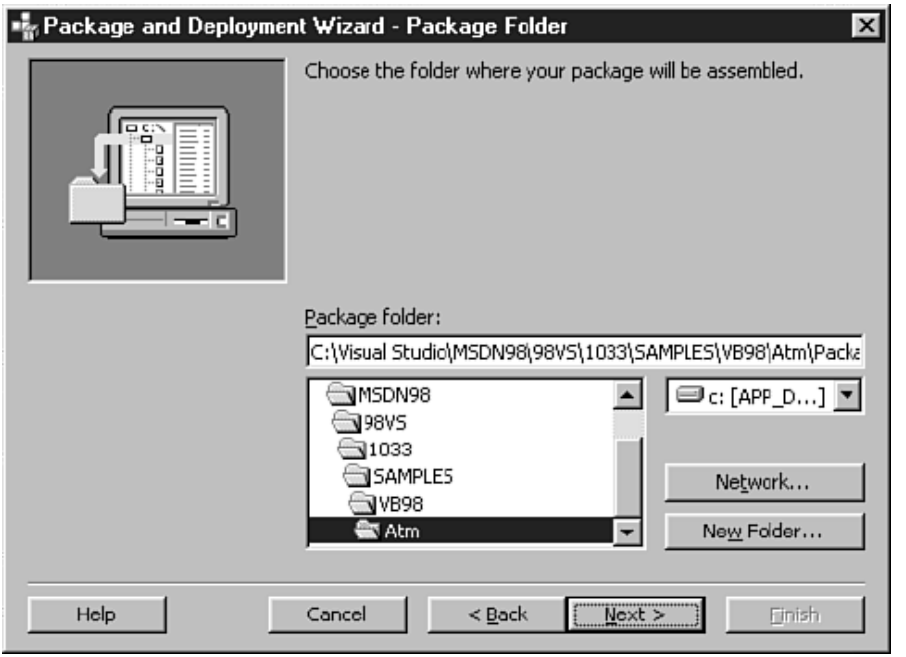
نکته:

فایل‌های فلاپی دیسک یا CD-ROM را در این مرحله برای ویزارد ارایه نکنید، زیرا قبل از کامل شدن فرایند، ویزارد چندین بار فایل‌ها را تغییر می‌دهد. فایل‌ها را روی هارددیسک ذخیره کنید و بعد از پایان فرایند آنها را روی CD کپی کنید.

۵- محل فایل‌های نصب که ویزارد ایجاد خواهد کرد را تعیین کنید. این فایل‌ها به رسانه‌ی مقصد که ممکن است دیسک یا CD باشد، کپی شوند. فهرستی را انتخاب کرده (شکل ۸-۸) و برای ادامه روی Next کلیک کنید.



شکل ۸–۷) نوع بسته ی نرم‌افزاری که می‌خواهید با ویزارد ایجاد کنید را انتخاب کنید.



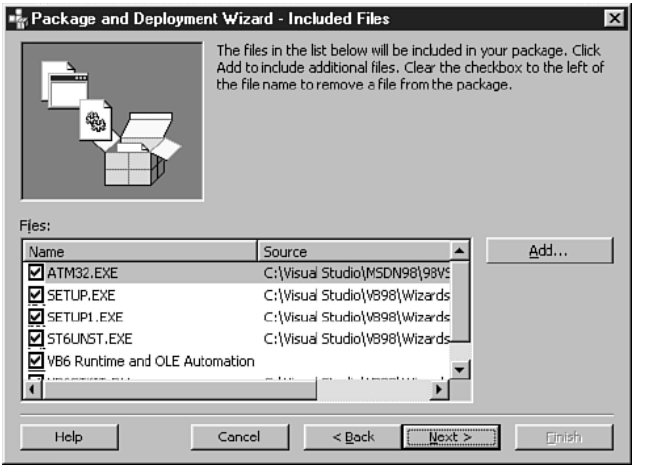
شکل ۸–۸) ویزارد نیاز به محلی برای نگهداری موقتی فایل‌های مورد نظر برای نصب دارد.

۶- همانطور که قبلاً نیز بیان شد، ویژوال بیسیک فایل‌های زیادی به غیر از فایل اجرایی دارد. کادرمحاوره‌ای بعدی (شکل ۸–۹)، فایل‌های مورد نیاز برای نصب را به همراه فایل اجرایی، لیست می‌کند. اگر فایل‌های دیگری (مثل فایل‌های راهنما) دارید که باید نصب شوند، می‌توانید آنها را در این مرحله اضافه کنید. روی دکمه ی Next کلیک کنید.

نکته:

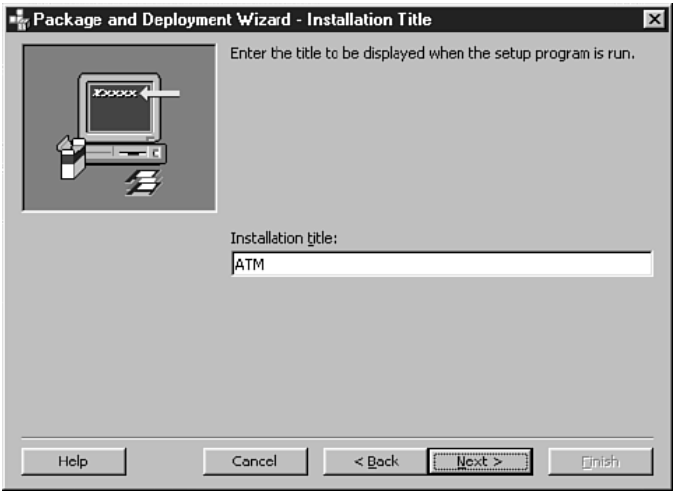
فایل‌هایی که ویزارد تولید خواهد کرد را فایل‌های CAB (کابینت) می‌نامند. این فایل‌ها نوع خاصی از فایل‌های آرشیوی هستند که به وسیله‌ی مایکروسافت طراحی شده و خیلی شبیه به فایل Zip هستند. اگر از نرم‌افزار Winzip نسخه‌ی ۷,۰ یا بالاتر استفاده می‌کنید، می‌توانید محتوای فایل‌های CAB را مشاهده کنید.

۷- مرحله‌ی بعدی در ویزارد تعیین اندازه‌ی رسانه‌ی مقصد است. اگر قصد دارید برنامه‌ی کاربردی را روی فلاپی دیسک‌ها قرار دهید، بزرگترین فایلی که ویزارد می‌تواند تولید کند به اندازه دیسک خواهد بود. برای کپی فایل‌ها روی فلاپی دیسک، Single Cab را انتخاب کرده و روی Next کلیک کنید تا ادامه یابد.



شکل ۸-۹) این کادرمحاوره‌ای، لیست کاملی از فایل‌های مورد نیاز برای ایجاد برنامه‌ی کاربردی که روی کامپیوتر دیگری اجرا می‌شود را نشان می‌دهد.

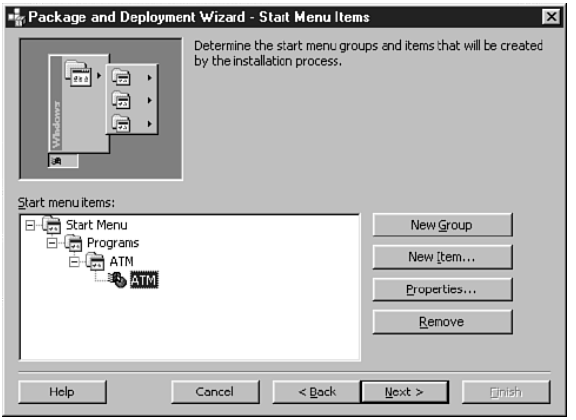
۸- عنوان برنامه‌ی کاربردی را تعیین کنید (شکل ۸-۱۰). این عنوان در طول نصب نمایش داده خواهد شد. عنوان مناسبی را وارد کرده و برای ادامه روی Next کلیک کنید.



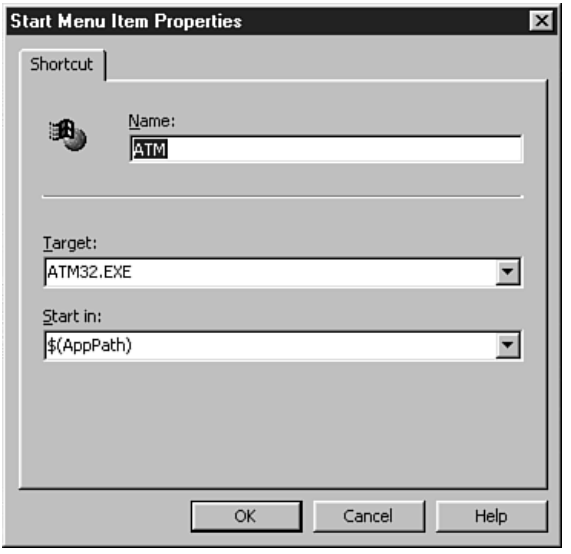
شکل ۸-۱۰) عنوان برنامه‌ی کاربردی در این کادرمحاوره‌ای را وارد کنید.

۹- برای برنامه‌کاربردی که استفاده می‌کنید، نیاز دارید آیکنی در منوی Start داشته باشید. مرحله‌ی بعدی ویزارد (شکل ۸-۱۱)، یک روش منحصر به فرد برای تعیین گروه‌هایی از آیکن‌هاست. تنظیمات پیش فرض، ایجاد گروه با نام برنامه‌ی کاربردی و سپس ایجاد آیکنی برای شروع برنامه است.

به دلیل اینکه برنامه‌ی کاربردی فقط دارای یک آیکن است، به طور استاندارد آیکن آن تحت All Programs ایجاد می‌شود. روی گروه ATM و سپس دکمه‌ی Remove کلیک کنید. سپس روی دکمه‌ی New Item کلیک کرده و نام برنامه‌ی کاربردی را وارد کنید و سپس در کادرمحاوره‌ای که ظاهر می‌شود، روی Ok کلیک کنید (شکل ۸-۱۲). بعد از پایان اضافه کردن گروه‌ها و آیکن‌ها روی Next کلیک کنید.



شکل ۸-۱۱) آیکن‌ها و گروه‌های موردنظر را برای برنامه‌ی کاربردی انتخاب کنید.

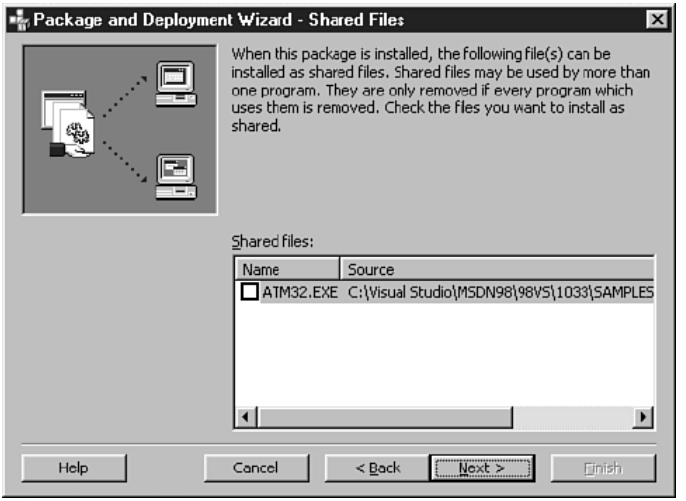


شکل ۸-۱۲) گزینه‌های اصلی برای هر آیکنی که می‌خواهید ایجاد کنید را تعیین کنید.

۱۰- کادر محاوره‌ای بعدی (شکل ۸-۱۳) امکان تغییر محل نصب هر فایلی که به وسیله‌ی سیستم مورد نیاز نیست را فراهم می‌کند. تمام فایل‌های سیستمی به طور خودکار در فهرست windows\system نصب می‌شوند، مگر اینکه محل دیگری را تعیین کنید. به دلیل اینکه فهرست تعیین شده برای این مثال صحیح است، روی Next کلیک کنید.

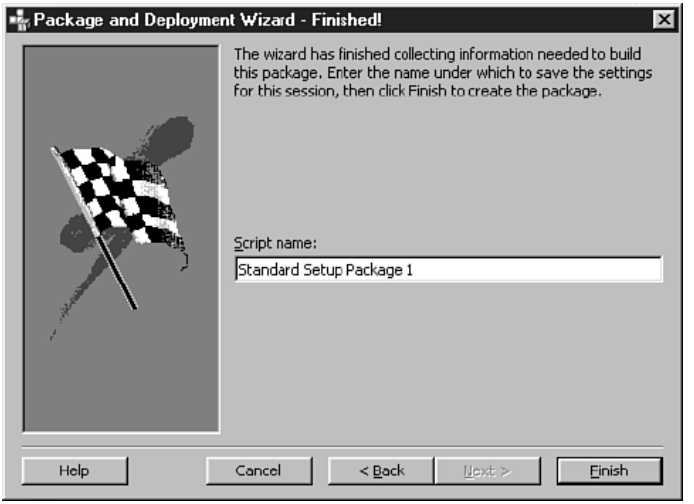
۱۱- فایل‌های اصلی (مثل DLLها و OCXها)، فایل‌های اشتراکی هستند. اگر هرکدام از این فایل‌ها را به عنوان بخشی از نصب اضافه کنید، آنها به صورت اشتراکی نشانه‌گذاری می‌شوند. بنابراین، هنگامی که کاربران برنامه‌ی کاربردی را حذف (Uninstal) کنند، فایل‌های به اشتراک گذاشته شده قبل از حذف، اصلاح می‌شوند. کادرمحاوره‌ای نشان داده شده در شکل ۸-۱۴ امکان نشانه‌گذاری همه‌ی فایل‌های اشتراکی را فراهم می‌کند. روی Next کلیک کنید.

شکل ۸-۱۳) اگر نیاز به تغییر فهرست نصب برای فایل‌هایی که اضافه کرده‌اید دارید، این کادر محاوره‌ای امکان انجام این کار را فراهم می‌کند.



شکل ۸–۱٤) فایل‌های اشتراکی را در این کادر محاوره‌ای، نشانه‌گذاری کنید.

۱۲– نامی را تعیین کنید (شکل ۸–۱۵) و روی دکمه‌ی Finish کلیک کنید تا بسته‌ی نـرم‌افـزاری نصب ایجاد شود.



شکل ۸–۱۵) آخرین مرحله‌ی ویزارد امکان ارایه‌ی نامی برای کاربرد بعدی را فراهم می‌کند.

۱۳- بعد از اینکه ویزارد ساخت بسته‌ی نرم افزاری نصب را خاتمه داد، گزارشی را با چند پیام مهم تولید می‌کند. گزارش را بخوانید و سپس در گزارش و ویزارد روی Close کلیک کنید.