بسم الله الرحمن الرحیم

برنامه‌سازی (۲)

رشتهٔ کامپیوتر

گروه تحصیلی کامپیوتر

زمینهٔ خدمات

شاخهٔ آموزش فنی و حرفه‌ای

۰۰۵ جباریه، علیرضا

۲ ب ۲۷۹ج/ برنامه‌سازی (۲) / مؤلف: علیرضا جباریه. ــ تهران: شرکت چاپ و نشر کتاب‌های درسی ایران، ۱۳۹۳.

۱۳۹۳ ۲۱۸ص. ــ (آموزش فنی و حرفه‌ای)

متون درسی رشتهٔ کامپیوتر گروه تحصیلی کامپیوتر، زمینهٔ خدمات.

برنامه‌ریزی و نظارت، بررسی و تصویب محتوا: کمیسیون برنامه‌ریزی و تألیف کتاب‌های درسی رشتهٔ کامپیوتر دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش وزارت آموزش و پرورش.

۱. برنامه‌سازی. الف. ایران. وزارت آموزش و پرورش. دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش. ب. عنوان. ج. فروست.

۱۳۹۳

همکاران محترم و دانش‌آموزان عزیز :

پیشنهـادات و نظرات خود را دربارهٔ محتوای ایـن کتاب بـه نشانی تهران ـ صندوق پستی شمارهٔ ۴۸۷۴/۱۵ دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش، ارسال فرمایند.

پیام‌نگار (ایمیل) **info@tvoccd.sch.ir**

وب‌گاه (وب‌سایت) **www.tvoccd.sch.ir**

وزارت آموزش و پرورش

سازمان پژوهش و برنامه‌ریزی آموزشی

برنامه‌ریزی محتوا و نظارت بر تألیف : **دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش**

نام کتاب : **برنامه سازی(۲) ـ ۴۵۱**

مؤلف : **علیرضا جباریه** و با همکاری **فرزانه شیخی**

اعضای کمیسیون تخصصی : **محمد مشاهری‌فرد ، عسگر قندچی، سیّدحمیدرضا ضیایی، محمدرضایمقانی ، هادی عابدی، ملیحه طزری و حمید احدی**

آماده‌سازی و نظارت بر چاپ و توزیع : **ادارهٔ کلّ نظارت بر نشر و توزیع مواد آموزشی**

تهران : خیابان ایرانشهر شمالی ـ ساختمان شمارهٔ ۴ آموزش و پرورش (شهید موسوی)

تلفن : ۹ـ۸۸۸۳۱۱۶۱، دورنگار : ۸۸۳۰۹۲۶۶، کدپستی : ۱۵۸۴۷۴۷۳۵۹،

وب‌سایت : www.chap.sch.ir

مدیر امور فنی و چاپ : **سیّد احمد حسینی**

طراح جلد : **طاهره حسن‌زاده**

صفحه‌آرا : **راحله زادفتح‌اله**

حروفچین : **کبری اجابتی**

مصحح : **شاداب ارشادی، شهلا دالایی**

امور آماده‌سازی خبر : **سپیده ملک‌ایزدی**

امور فنی رایانه‌ای : **حمید ثابت کلاچاهی، پیمان حبیب‌پور**

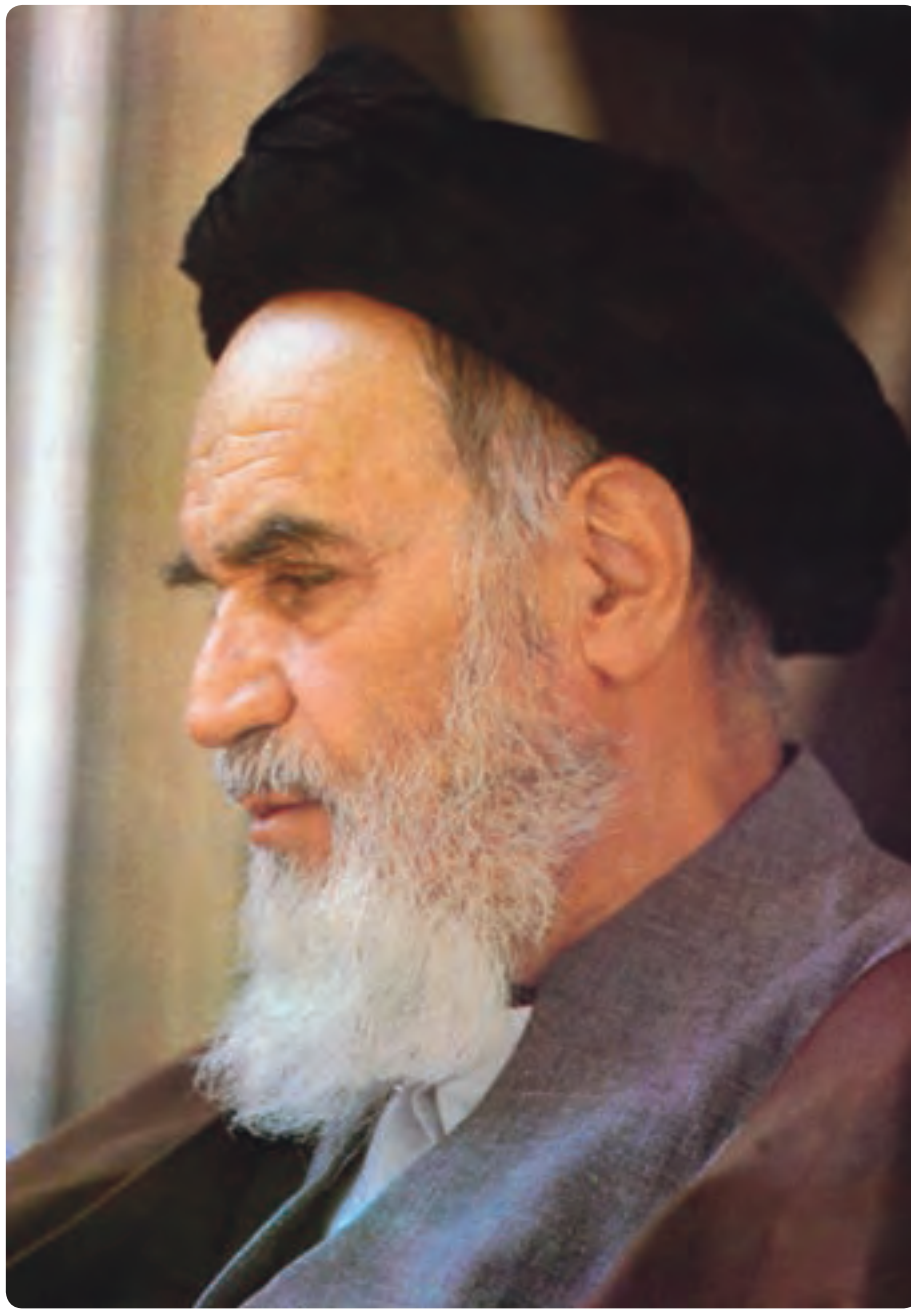
ناشر : **شرکت چاپ و نشر کتاب‌های درسی ایران : تهران ـ کیلومتر ۱۷ جادّهٔ مخصوص کرج ـ خیابان ۶۱ (داروپخش)**

تلفن : ۵ ـ ۴۴۹۸۵۱۶۱، دورنگار : ۴۴۹۸۵۱۶۰، صندوق پستی : ۱۳۹ـ۳۷۵۱۵

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

سال انتشار و نوبت چاپ : **چاپ نهم ۱۳۹۳**

شابک ۳-۱۴۲۳-۰۵-۹۶۴ **ISBN 964-05-1423-3**



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

امام خمینی «قدّس سرّه الشّریف»

فهرست

مقدمه

زبان برنامه‌نویسی Visual Basic یکی از زبان‌های برنامه‌نویسی متداول و محبوب در بین برنامه‌نویسان است. در کتاب برنامه‌سازی ۱ با برخی از جنبه‌های مقدماتی این زبان آشنا شدید. در این کتاب با جنبه‌های پیشرفته این زبان آشنا می‌شوید. یادگیری مفاهیم این کتاب شما را قادر خواهد ساخت تا برنامه‌های کامل‌تر و پیچیده‌تر را بنویسید و امکانات این زبان را به نحو مناسب‌تری به کار ببرید. در این کتاب با منوهای استاندارد، آرایه، کاربرد فرم‌های آماده، گرافیک، روال‌ها و توابع و برخی دیگر از امکانات Visual Basic آشنا شده و با آن‌ها کار خواهید کرد. توصیه می‌شود برای تقویت مهارت‌های خود در زمینهٔ برنامه‌نویسی که یکی از جنبه‌های مهم رشته کامپیوتر است، علاوه بر انجام پروژه‌هایی که در این کتاب ارایه شده است کتاب‌ها و منابع مرتبط را مطالعه نموده و تا حد امکان کد برنامه‌های مختلف را مورد تجزیه و تحلیل قرار دهید.

هدف کلی

شناخت مفاهیم پیشرفته و مباحث تکمیلی زبان برنامه‌نویسی VB و نوشتن برنامه‌های پیشرفته با آن

فصل اول

تصویرها، ماوس و صفحه کلید

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

انواع تصویرهای گرافیکی را شناسایی کرده و آن‌ها را در برنامه‌های خود به‌کار ببرد.

- **از تابع Load Picture برای قراردادن تصویر در کنترل‌های Image و Picture Box استفاده کند.**
- **از رویدادهای صفحه کلید و ماوس در برنامه‌های خود استفاده کند.**
- **مشخصهٔ Key Preview را برای نحوهٔ دریافت ضربات صفحه کلید به وسیلهٔ فرم، مقداردهی کند.**
- **با استفاده از دستور Send Keys کلیدهایی را به برنامه ارسال کند.**

تصویرهای گرافیکی مفاهیم را به‌خوبی بیان می‌کنند. یکی از مشخصه‌های مشترک تمام قالب‌های تصویری، عمق رنگ (Color depth) است (تعداد رنگ‌هایی که تصویر از آن‌ها پشتیبانی می‌کند). عمق رنگ، اغلب با عبارت بیت به کار برده می‌شود. تعداد زیاد بیت‌ها عمق رنگ بیشتر و درنتیجه، تصویری با کیفیت بالا را ارایه می‌کنند. تصویرهایی با عمق رنگ یک بیت، تصویرهایی تک رنگ هستند (تصویرهای سیاه و سفید). تصویرهایی با عمق رنگ ۸ بیت، دارای ۲۵۶ رنگ و اغلب، تصویرهای تخت نامیده می‌شوند. تصویرهایی با عمق ۱۶ بیت، تصویرهای با کیفیت بالا نامیده می‌شوند که دارای ۶۵۵۳۶ رنگ هستند. تصویرهایی با عمق ۲۴ بیت، تصویرهای با رنگ واقعی (true Color) هستند. ویژوال بیسیک از انواع قالب‌های موجود برای تصویرها، از تصویرهای تک‌رنگ گرفته تا تصویرهای با رنگ واقعی حمایت می‌کند (جدول ۱ــ۱).

جدول ۱ــ۱ــ قالب‌های تصویری که به وسیله ویژوال بیسیک پشتیبانی می‌شوند.

نوع	پسوند	توضیحات
GIF images	.gif	Graphic Interchange Format تصویرهایی با ۲۵۶ رنگ یا کمتر
Bitmap image	.bmp	تصویرهایی با عمق رنگ 1bit، 4bit،8bit، 16bit، 24bit
Bitmap image	.dib	تصویرهایی با عمق رنگ 1bit، 4bit،8bit،16bit،24bit
Icon image	.ico	تصویرهایی با ترکیب ۱۶ رنگ، عموماً دارای اندازهٔ ۳۲ در ۳۲ پیکسل که برای نشانه‌ها استفاده می‌شوند.
Curser image	.cur	یک نوع bitmap که عموماً برای اشاره‌گر ماوس استفاده می‌شود.
Windows meta file and Enhanced Windows	.emf و .wmf	تصویرهایی گرافیکی که یک تصویر هندسی را ارایه می‌کنند (مانند خط و دایره).
RLE	.rle	یک نوع تصویر فشرده شده
JPEG image	.jpg	Joint Photographic Experts Group از تصویرهای با رنگ واقعی و تصویرهای تخت پشتیبانی می‌کند.

۱ــ۱ــ کنترل‌های تصویر و کادر تصویر

معمولاً می‌توان تصویرها را با کنترل Image یا کنترل Picture Box به نمایش درآورد. اگرچه می‌توان تصویرها را با استفاده از مشخصهٔ Picture فرم یا سایر شیءهایی که این مشخصه را دارند نیز، نمایش داد. کنترل‌های Image و Picture Box دارای مشخصه‌های شبیه به هم هستند. کنترل PictureBox دارای مشخصه‌ها (مانند Back Color) و روال رویداد (مانند Change) بیشتری نسبت به کنترل image است.

کنترل‌های کادر تصویر (picture box) و تصویر (image)، در اصل عملکرد یکسانی دارند. هر دوی آن‌ها برای نمایش تصویرها و پرونده‌های گرافیکی روی فرم هستند. اما این دو کنترل تفاوت‌هایی هم دارند :

● کنترل کادر تصویر، انعطاف‌پذیری بیشتری دارد و متدهای بیشتری را پشتیبانی می‌کند.

● کنترل تصویر کارایی بهتری دارد و در PCهای با سرعت پایین یا کند بهتر عمل می‌کند. (البته امروزه، دیگر این جنبهٔ قضیه اهمیت چندانی ندارد.)

هر دو کنترل کادر تصویر و تصویر از قالب‌های گرافیکی جدول ۱ــ۱ پشتیبانی می‌کنند.

مهم‌ترین مشخصهٔ این دو کنترل، خاصیت Picture (تصویری که کنترل باید نمایش دهد) است. برای قراردادن تصویر در این کنترل‌ها، در هنگام اجرای برنامه می‌توانید از تابع ()Load Picture استفاده کنید :

```
picPortrait.Pictuer = LoadPicture ("C: \My Photos\ Masque.wmf")
```

دقت کنید که نمی‌توان به‌طور مستقیم یک پرونده تصویری را به خاصیت Picture نسبت داد. تابع ()Load Picture یکی از مهم‌ترین توابع کار با پرونده گرافیکی است. شکل کلی این تابع چنین است :

```
LoadPictuer ([GraphicFileName],[,varSize],[,varColorDepth],[varX,varY])
```

توجه کنید که حتی نام پرونده هم در این تابع اختیاری است و اگر از این تابع بدون مشخص‌کردن نام پرونده استفاده کنید، ویژوال بیسیک کنترل تصویر را پاک خواهد کرد.

باید توجه داشت که متد cls، فقط عبارت‌های نوشته‌شده یا نقاط رسم شده در کنترل تصویر را پاک می‌کند ولی این تابع، می‌تواند عکس درون کنترل تصویر را پاک کند.

آرگومان varSize می‌تواند مقادیر مختلفی بگیرد که آن‌ها را در جدول ۲ــ۱ مشاهده می‌کنید. این آرگومان که اندازهٔ تصویر برای پرونده نشانه یا اشاره‌گر را تعیین می‌کند، یکی از آرگومان‌های مهم این تابع است.

جدول ۲ــ۱ــ مقادیر آرگومان varSize تابع ()Load Picture (فقط برای پرونده‌های نشانه و اشاره‌گر)

ثابت نام‌دار	مقدار	مفهوم
VbLPSmall	۰	نشانه‌های کوچک
VbLPLarge	۱	نشانه‌های بزرگ
VbLPSmallShell	۲	از تنظیمات Display| Appearance مرکز کنترل ویندوز استفاده می‌کند.
VbLPLargeShell	۳	از تنظیمات Display	Appearance مرکز کنترل ویندوز استفاده می‌کند.
VbLPCustom	۴	اندازهٔ نشانه یا آرگومان‌های varX و varY تعیین خواهد شد.

در جدول ۳ــ۱ هم مقادیر آرگومان اختیاری varColor Depth را مشاهده می‌کنید.

جدول ۳ــ۱ مقادیر آرگومان varColorDepth تابع Load Picture() (فقط برای پرونده‌های نشانه یا اشاره‌گر)

ثابت نام‌دار	مقدار	مفهوم
VbLPDefault	۰	بهترین انطباق
VbLPMonochrome	۱	۲ رنگ
VbLPVGAColor	۲	۱۶ رنگ
VbLPColor	۳	۲۵۶ رنگ

کنترل‌های کادر تصویر و تصویر حتی اگر به یک اندازه باشند و تصویر مشابهی را نمایش دهند، به‌طور متفاوتی عکس‌العمل نشان می‌دهند. در کنترل تصویر، قبل از تنظیم کردن خواص Width و Height، باید خاصیت Stretch با True مقداردهی شده باشد. در غیراین صورت، اندازهٔ کنترل به‌طور خودکار به اندازهٔ تصویر موجود در آن درخواهد آمد. اما در مورد کنترل کادر تصویر، فقط بخشی از تصویر که به اندازهٔ کنترل است، نمایش می‌یابد. به عبارت دیگر، کادر تصویر قسمتی از تصویر را نمایش می‌دهد، ولی کنترل تصویر خود به اندازهٔ تصویر درمی‌آید (مگر آن که خاصیت Stretch با True تنظیم شده باشد).

حتی فرم‌ها هم خاصیت Picture را دارند و می‌توانید با تابع Load Picture() تصویرها را به‌طور مستقیم روی فرم قرار دهید. شکل ۲ــ۱ فرمی را که یک تصویر در آن قرار داده شده است، نشان می‌دهد.

تابع load Picture() چه چیزی برمی‌گرداند؟ واضح است، تصویری که نام پرونده آن داده شده است. مشاهده می‌کنید که مقدار برگشتی تابع Load Picture() به خاصیت Picture کنترل تصویر نسبت داده می‌شود، زیرا بدون انجام این کار، کنترل مزبور نمی‌تواند تصویر را نمایش دهد.

هنگامی که ویژوال بیسیک با تابع Load Picture() برخورد کند، وجود یا وجود نداشتن پروندهٔ مشخص شده را بررسی می‌کند (اگر پروندهٔ مزبور روی شبکه باشد، ویژوال بیسیک باید بررسی کند که آیا شما حق استفاده از آن را دارید یا خیر). در پایان هم پروندهٔ تصویر را خوانده و آن را در کنترل تصویر قرار می‌دهد.

شکل ۱ــ۱ــ تصویرها می‌توانند به‌طور مستقیم روی فرم قرار گیرند.

برنامهٔ ۱ــ۱ تصویری را به داخل یک کادر تصویر بار می‌کند. سپس با فشار دکمهٔ Inverse، یک تصویر معکوس از تصویر اصلی ایجاد می‌شود. همچنین برنامه، تصویر معکوس شده را ذخیره می‌کند. روال cmdInvert_Click متد PaintPicture را فراخوانی می‌کند تا معکوس تصویر در کادر تصویر به نمایش درآید. آرگومان اول متد PaintPicture مشخص می‌کند که تصویر رسم شود. آرگومان دوم و سوم مشخص‌کنندهٔ مختصات x و y هستند و تعیین می‌کنند که تصویر در گوشهٔ سمت چپ بالا به نمایش درآید. PaintPicture دارای هفت آرگومان اختیاری برای ایجاد جلوه‌های گوناگون است. تعدادی ازاین آرگومان‌ها بر روی تصویرهای نقش بیتی تأثیر دارند. آرگومان آخر یک مقدار Long است که نحوهٔ انجام عمل را مشخص می‌کند. ثابت vbDstInvert، یک عمل معکوس‌کننده بر روی تصویر نقش بیتی انجام می‌دهد.

برنامهٔ ۱ــ۱ــ معکوس‌کردن یک تصویر

```
1. Private Sub Form_Load ()
2. PicPicture.Picture = Load Picture ("e: \image\cho9\cool.bmp")
3. End Sub
4. Private sub cmdInvert _ Click ( )
5. picPicture.PaintPicture picPicture.Picture, 0, 0, , , , , , , vbDstInvert
6. Save Picture picPicture, "d: \images\cho9\" & "Cool _ invertse. bmp"
7. End Sub
```

متد Save Picture تصویر را بر روی دیسک ذخیره می‌کند. این متد دو آرگومان دریافت می‌کند : یک تصویر و یک رشته، محل و نام پرونده را مشخص می‌کند (شکل ۲ــ۱). (توجه : پرونده تصویری مورد نظر باید در مسیر باشد.)

شکل ۲ــ۱

۲ــ۱ــ ماوس و صفحه کلید

یکی از اساسی‌ترین اصول برنامه‌های ویندوز این است که آن‌ها نسبت به عملیات ماوس عکس‌العمل نشان می‌دهند. هرگاه کاربر عملی با ماوس انجام دهد، ویندوز رویدادی را به برنامه می‌فرستد. هنگام برنامه‌نویسی باید این رویدادها را بررسی کرده و در پاسخ به هرکدام، اقدام مناسب را انجام دهید. البته بعضی از رویدادهای ماوس (به عنوان مثال، کلیک‌کردن روی کادر علامت یا دکمهٔ انتخاب) نیازی به اقدام برنامه‌نویس ندارند، ولی برنامه‌نویس باید رویدادهای دیگر را کنترل کند. هنگامی که کاربر شیئی را روی فرم برنامه جابه‌جا می‌کند یا اطلاعاتی را کپی کرده و در نقطه‌ای دیگر می‌چسباند، خود ویژوال بیسیک بر رویدادها و اعمال ماوس نظارت خواهد کرد.

البته این یکی از استانداردهای ویندوز است که یک برنامه حتی بدون ماوس و فقط به کمک صفحه کلید هم باید بتواند تمام کارایی خود را حفظ کند. با این حال، طبیعت برخی برنامه‌ها به گونه‌ای است که بدون ماوس نمی‌توانند به درستی عمل کنند؛ برنامه‌های طراحی و نقاشی از این قبیل هستند.

کاربر به وسیلهٔ ماوس و صفحه کلید با برنامه‌های ویژوال بیسیک ارتباط برقرار می‌کند. از ماوس برای کلیک‌کردن روی دکمه‌ها، جابه‌جاکردن پنجره‌ها، تغییر اندازهٔ پنجره‌ها، ترسیم نقاط و شکل‌ها و

سایر عملیات استفاده می‌شود. کنترل‌های ویژوال بیسیک می‌توانند کلیک، دوبار کلیک، جابه‌جایی ماوس، و فشار داده‌شدن یا رهاشدن دکمهٔ ماوس را تشخیص دهند. مانند سایر رویدادها، برنامه نویس می‌تواند برای رویدادهای ماوس نیز کد بنویسد. برنامه می‌تواند بررسی کند که کدام دکمهٔ ماوس (چپ، وسط یا راست) فشار داده شده است. ویژوال بیسیک از مفهوم کشیدن و رها کردن (drag and drop) به وسیلهٔ ماوس نیز پشتیبانی می‌کند.

۱ــ۲ــ۱ــ رویدادهای ماوس : ماوس می‌تواند رویدادهای مختلفی تولید کند :

- جابه‌جایی ماوس
- کلیک (click)
- دوبار کلیک (double – click)
- کلیک راست (right – click)
- عملیات کشیدن ــ رها کردن (Drag & Drop)

۲ــ۲ــ۱ــ تغییر شکل اشاره‌گر ماوس : هنگامی که کاربر، ماوس خود را جابه‌جا می‌کند، اشاره‌گر ماوس (Pointer) روی صفحه جابه‌جا می‌شود. در اغلب برنامه‌ها، هنگامی که کاربر کار خاصی با ماوس انجام می‌دهد یا در ناحیهٔ خاصی حرکت می‌کند، اشاره‌گر ماوس تغییر شکل می‌دهد. به‌عنوان مثال، تبدیل اشاره‌گر ماوس به یک ساعت شنی نشان می‌دهد که کار خاصی در حال انجام است و کاربر باید صبر کند.

شکل اشاره‌گر ماوس، تحت کنترل برنامه‌نویس است. در جدول ۴ــ۱ شکل‌های مختلف اشاره‌گر ماوس را مشاهده می‌کنید. برای تغییر شکل اشاره‌گر، هنگامی که ماوس روی یک کنترل خاص است، باید از مشخصهٔ MousePointer آن کنترل استفاده کنید. تقریباً تمام کنترل‌های ویژوال بیسیک مشخصهٔ MousePointer را دارند.

مشخصهٔ MousePointer فرم یا کنترل، شکل اشاره‌گر ماوس را هنگامی که اشاره‌گر وارد ناحیهٔ کنترل می‌شود، تعیین می‌کند. به این منظور، ویژوال بیسیک دارای ثابت‌هایی است که شکل اشاره‌گر ماوس را تغییر می‌دهند (جدول ۴ــ۱). این تغییر شکل در واقع عمل خاصی انجام نمی‌دهد، ولی می‌تواند از لحاظ بصری بیان‌کنندهٔ انجام عمل خاصی باشد. به‌عنوان مثال، در نرم‌افزار وُرد اشاره گر ماوس به شکل I بر این نکته دلالت دارد که کاربر می‌تواند متنی را در داخل سند با کلیک کردن در آن نقطه وارد کند.

ویندوز امکان استفاده از تصویرهای متحرک، در اشاره‌گر ماوس را فراهم می‌کند. مکان‌نماهای متحرک دارای پسوند ANI هستند. مکان‌نماهای متحرک را نمی‌توان در ویژوال بیسیک به‌کار برد.

جدول ۴ــ۱ــ ثابت‌های MousePointer

مقدار	ثابت‌ها	مفهوم
۰	vbDefault	شکل پیش‌فرض اشاره‌گر
۱	vbArrow	علامت معمولی اشاره‌گر 
۲	vbCrosshair	علامت +
۳	vbIbeam	علامت I
۵	vbSizePointer	
۶	vbSizeNESW	
۷	vbSizeNS	
۸	vbSizeNWSE	
۹	vbSizeWE	
۱۰	vbUpArrow	
۱۱	vbHourglass	
۱۲	vbNoDrop	
۱۳	vbArrowHourglass	
۱۴	vbArrowQuestion	
۱۵	vbSizeAll	
۹۹	vbCustom	شکلی که به خاصیت Mouse Icon اشاره می‌کند.

نکته

در MousePointer استفاده از مکان‌نماهای متحرک امکان ندارد.

اشاره‌گر ماوس یک نشانه ۱۶×۱۶ (با پسوند ICO) است که خودتان هم می‌توانید آن را بسازید. برای نمایش این اشاره‌گر، ابتدا باید آن را در خاصیت MouseIcon قرار داده و سپس خاصیت

MousePointer را با Custom–99 مقداردهی کنید. تا زمانی که اشاره‌گر را دوباره تغییر نداده‌اید، این شکل به‌عنوان اشاره‌گر انجام وظیفه خواهد کرد.

وظیفهٔ ایجاد رویدادهای ماوس و ارسال آن‌ها به برنامه، بر عهدهٔ سیستم عامل است. اگر برنامه‌نویس روالی برای این رویدادها ننوشته باشد، برنامه آن‌ها را نادیده خواهد گرفت. در جدول ۵ــ۱ مشاهده می‌کنید که ماوس چه رویدادهایی می‌تواند داشته باشد.

جدول ۵ــ۱ــ رویدادهای ماوس

رویداد	مفهوم
Click	کاربر یکی از دکمه‌های ماوس را کلیک کرده است.
DblClick	کاربر یکی از دکمه‌های ماوس را دو بار کلیک کرده است.
MouseDown	کاربر دکمهٔ ماوس را فشار داده و نگه داشته است.
MouseMove	کاربر ماوس را جابه‌جا کرده است.
MouseUp	کاربر دکمهٔ ماوس را رها کرده است.

تمام رویدادهای ماوس به کنترل‌ها وابسته‌اند و تقریباً تمام کنترل‌ها (و فرم) دارای رویدادهای ماوس هستند، بجز کنترل‌هایی که شکل ظاهری ندارند (مثل Timer). به‌عنوان مثال، اگر فرمی به نام frm Test داشته باشید، کلیک‌کردن روی آن سبب ایجاد رویداد frmTest_Click خواهد شد. در برخی رویدادها لازم است برنامه‌نویس بداند که کاربر کدام دکمهٔ ماوس را کلیک کرده است. این کار فقط در رویدادهای MouseDown و MouseUp امکان‌پذیر است.

اما آیا دوبار کلیک یک رویداد مستقل است یا دو رویداد کلیک پشت سرهم؟ پاسخ این پرسش به دقت کاربر در اجرای دو بار کلیک بستگی دارد. ترتیب رویدادهای ماوس از نظر ایجاد آن‌ها به وسیلهٔ ویندوز چنین است :

۱ــ MouseDown

۲ــ MouseUp

۳ــ Click

۴ــ Dblclick

۵ــ MouseUp

بررسی کنید

دلیل تکراری بودن موارد ۲ و ۵ چیست؟

یعنی، ابتدا یک رویداد MouseDown روی می‌دهد، سپس یک MouseUp و به دنبال آن Click. هنگامی که کاربر دو بار کلیک کند، رویدادهای DblClick و MouseUp به دنبال آن‌ها روی خواهند داد.

رویدادهای MouseDown، MouseMove و MouseUp چهار آرگومان ورودی می‌گیرند :

- **intButton** : دکمه‌ای که فشار داده شده : ۱ برای دکمهٔ چپ، ۲ برای دکمهٔ راست و ۴ برای دکمهٔ وسط (اگر وجود داشته باشد).
- **intShift** : فشار داده شدن دکمه‌های Alt، Ctrl و Shift (همزمان با رویداد) را منعکس می‌کند (۴ برای Alt، ۲ برای Ctrl و ۱ برای Shift).
- **sngX** : مختص افقی اشاره‌گر ماوس در هنگام وقوع رویداد.
- **sngY** : مختص عمودی اشاره‌گر ماوس در هنگام وقوع رویداد.

ویژوال بیسیک فقط بعد از حرکت ماوس به میزان ۱۰ تا ۱۵ twip [۱] (که مقداری بسیار جزئی و کوچک است)، یک رویداد MouseMove تولید می‌کند و برای هر twip این رویداد را ایجاد نخواهد کرد. شکل کلی رویداد MouseDown چنین است :

```
Private Sub imgMouse _ MouseDown (intButton As Integer, intShift As Integer,
sngX As Single, sng Y As Single)
```

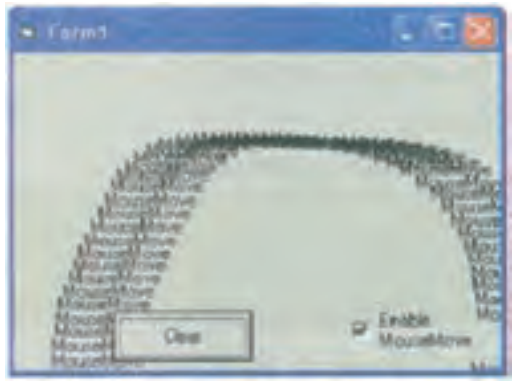
آرگومان‌های sngX و sngY مکان دقیق ماوس را هنگام ایجاد این رویداد برمی‌گردانند. آرگومان intButton (که برای ماوس‌های سه‌دکمه‌ای می‌تواند ۱، ۲ یا ۴ باشد)، کلید فشار داده شده را برمی‌گرداند. البته در اکثر موارد دکمهٔ فشار داده شده چندان مهم نیست، ولی اگر می‌خواهید برنامه با توجه به این که چه دکمه‌ای فشار داده شده است، عکس‌العمل‌های متفاوتی از خود بروز دهد، جای تعیین آن همین رویداد MouseDown است. برنامهٔ ۲ــ۱ برنامه‌ای را نشان می‌دهد که در آن همزمانی کلیدهای Alt، Ctrl یا Shift با رویداد MouseDown بررسی شده است.

۱ــ 1 inch = 2.54 cm = 72 points = 1440 twips

برنامهٔ ۲ــ۱ــ تعیین کلید زده شده همزمان با یک رویداد ماوس

```
1. Private Sub imgMouse_MouseDown (intButton As Integer, intShift As
   Integer, sngX As Single, sngY As Single)
2. Dim intShifState As Integer
3. intsShiftState = intShift And 7      'Special bitwise And
4. Select Case intShiftState
5. Case 1
6.         'Code for Shift combinations
7. Case 2
8.         'Code for Ctrl combinations
9. Case 3
10.    'Code for shift + Alt combinations
11. Case 4
12.    'Code for Alt combinations
13. Case 5
14.        'Code for shift + Alt combinations
15. Case 6
16.        'Code for Ctrl + Alt combinations
17. Case 7
18.        'Code for shift + Ctrl + Alt combinations
19. End Select
20. End Sub
```

در خط ۳ برای بررسی کلید زده شده از And با عدد ۷ (۱۱۱ دودویی) استفاده شده است.



شکل ۳ــ۱

برنامهٔ ۳ــ۱ نحوهٔ استفاده از رویدادهای ماوس را نشان می‌دهد (شکل ۳ــ۱).

برنامهٔ ۳ــ۱ــ استفاده از رویدادهای ماوس

```
1. 'Demonstrating mouse events
2. Option Explicit  'General declaration
3. Private Sub  Form_Load ( )
4.         Call randomize     'Randomize
5. End Sub
6. Private Sub cmdClear_Click ( )
7.         Call Cls  'Clear Form
8. End Sub
9. Private Sub  Form_Click ( )
10.        'Randomly Set Form ForeColor
11.        Select Case  (1+Int (Rnd ( )* 4))
12.          Case 1
13.           ForeColor = vbBlack
14.          Case 2
15.           ForeColor = vbMagenta
16.          Case 3
```

```
17.            ForeColor = vbRed
18.          Case 4
19.            ForeColor = vbBlue
20.    End Select
21. End Sub
22. Private Sub Form_Dblclick ( )
23.        'Randomly Set Form BackColor
24.        Select Case (1 + Int (Rnd( ) )* 4)
25.          Case 1
26.            BackColor = vbWhite
27.          Case 2
28.            BackColor = vbYellow
29.          Case 3
30.            BackColor = vbGreen
31.          Case 4
32.            BackColor = vbCyan
33.    End Select
34.    'Changing ChkMove  BackColor  to Form's  BackColor
35.    chkMove. BackColor = BackColor
36. End Sub
37. Private Sub Form_Mouse Down (Button As Integer,
38.        Shift As Integer, X As Single, Y As single
39.        CurrentX = X  'Set x Coordinate
40.        CurrentY = Y 'Set y Coordinate
41.        Print "MouseDown"
42.       End Sub
43.      Private Sub Form_MouseUp (Button As Integer,
```

```
44.     Shift as Integer, X As Single, Y As Single)
45.     'Reverse Coordinates
46.     Current X = Y
47.     Current Y = X
48.     Print "MouseUp"
49. End Sub
50. Private Sub Form_MouseMove (Button As Integer,
51.     Shift As Integer, X As Single, Y as Single)
52.     'If checked enable Printing Operations
53.     If chkMove. Value = 1 Then
54.       CurrentX = X
55.       Current Y  = Y
56.       Print "MouseMove"
57.    End If
58. End Sub
```

روال Form_MouseDown هنگامی فراخوانی می‌شود که دکمهٔ سمت چپ یا راست ماوس فشار داده شده باشد. مشخصه‌های CurrentX و CurrentY فرم در حالتی که MouseDown رخ دهد با مختصات X و Y تنظیم می‌شوند. این مشخصه‌ها تعیین می‌کنند که متد Print در زمان اجرا، در کدام محل اعمال شود. عبارت "MouseDown" در مکانی که MouseDown رخ داده است، چاپ می‌شود. هنگام رهاشدن دکمهٔ ماوس، روال Form_MouseUp فراخوانی خواهد شد. CurrentX با Y ر CurrentY با X مقداردهی شده است. به این ترتیب عبارت "MouseUP" بر روی "MouseDown" چاپ نمی‌شود.

با کلیک‌کردن روی فرم، روال Form_Click فراخوانی شده و رنگ رویه فرم را به سیاه، ارغوانی، قرمز یا آبی تغییر می‌دهد. با دو بار کلیک سریع بر روی فرم، روال Form_DblClick فراخوانی شده و رنگ زمینه به صورت تصادفی به رنگ های سفید، زرد، سبز یا فیروزه‌ای تغییر می کند. با حرکت‌دادن ماوس بر روی فرم، روال Form_MouseMove فراخوانی می‌شود.

اگر کادر علامت chkMove علامت‌دار شده باشد، عبارت "MouseMove" در موقعیتی که

ماوس قرار دارد، به چاپ می‌رسد. در غیر این صورت، چیزی چاپ نمی‌شود.

۳ــ۲ــ۱ــ دکمه‌های ماوس

آرگومان Button در روال‌های MouseDown، MouseUp و MouseMove تعیین‌کنندهٔ دکمه‌ای از ماوس است که فشار داده شده است. ماوس می‌تواند یک، دو یا سه دکمه داشته باشد.

جدول ۶ــ۱ لیستی از مقادیر ثابت که با آرگومان Button در ارتباط هستند را ارایه می‌کند.

جدول ۶ــ۱ــ ثابت‌های دکمهٔ ماوس

ثابت‌ها	مقدار	توضیحات
vbRightButton	۲	دکمهٔ سمت راست ماوس
vbLeftButton	۱	دکمهٔ سمت چپ ماوس
vbMiddleButton	۴	دکمهٔ وسط ماوس

برنامهٔ مثال زیر، تعیین می‌کند کدام‌یک از دکمه‌های ماوس فشار داده شده است.

مثال ۱ــ۱

فرمی طراحی کنید که دارای یک کنترل Image به نام imgImage باشد، سپس کد زیر را در پنجرهٔ کد این فرم وارد کنید. مسیر پرونده‌ها را متناسب با مسیر آن‌ها در رایانه خودتان انتخاب کنید.

```
1. 'Mouse buttons
2. Options Explicit  'General declaration
3. Private Sub  Form_Load ( )
4.          imgImage.Picture = LoadPicture ("d: \mouse0.gif")
5. End Sub
6. Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
7.    imgImage.Picture = LoadPicture ("d: \mouse" & Button &".gif")
8. End Sub
```

```
9. Private Sub Form_MouseUp (Button As Integer, Shift As Integer, X As
Single, Y As Single)
10.    imgImage.Picture = LoadPicture ("d: \mouse0.gif")
11. End Sub
12. Private Sub imgImage_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y as Single)
13.    imgImage. Picture = LoadPicture ("d: \mouse"& Button & ".gif")
14. End Sub
15. Private Sub imgImage_MouseUp (Button As Integer, Shift As Integer, X
As Single, Y As Single)
16. imgImage.Picture = LoadPicture ("d: \mouse0.gif")
17. End Sub
```

۴ــ۲ــ۱ــ کلیدهای Shift، Ctrl و Alt : می‌توان وضعیت کلیدهای Shift، Ctrl و Alt را در هنگامی که رویدادهایMouseDown، MouseMove یا MouseUp رخ می‌دهند، تعیین کرد. مقداری که نشان‌دهنده وضعیت این کلیدهاست در متغیری به نام Shift ذخیره می‌شود. جدول ۷ــ۱ شامل ثابت‌هایی است که با متغیر Shift در ارتباط هستند. ثابت‌های Shift نیز مانند ثابت‌های Button، می‌توانند حالت ترکیبی داشته باشند.

جدول ۷ــ۱ــ ثابت‌هایی که با متغیر Shift در ارتباط هستند.

ثابت‌ها	مقدار	توضیحات
vbShiftMask	۱	وضعیت کلید Shift
vbCtrlMask	۲	وضعیت کلید Ctrl
vbAltMask	۴	وضعیت کلید Alt

۵ــ۲ــ۱ــ کشیدن و رهاکردن : وقتی کاربر یک شیء را (روی فرم) از نقطه‌ای به نقطهٔ

دیگر می‌کشد، برنامه باید از آن مطلع باشد! کشیدن ــ رهاکردن (drag&drop) عملیاتی است که طی آن کاربر روی یک شیء کلیک کرده، دکمهٔ ماوس را نگه می‌دارد و آن شیء را به نقطه‌ای دیگر می‌برد. به کمک ویندوز، برنامه‌نویسی کشیدن ــ رهاکردن نسبتاً ساده است، چون ویندوز تمام اطلاعات دربارهٔ این فرایند را در اختیار برنامه‌نویس قرار خواهد داد. ویژوال بیسیک دو نوع عملیات کشیدن ــ رهاکردن دارد :

- کشیدن ــ رهاکردن خودکار
- کشیدن ــ رهاکردن دستی

روش اول :

انجام این روش، از طریق خواص کنترل صورت می‌گیرد. تقریباً تمام کنترل‌های ویژوال بیسیک دارای خاصیت DragMode هستند. این خاصیت امکان می‌دهد تا کاربر یک کنترل را با ماوس جابه‌جا کند (وقتی کنترل در حال حرکت است، ویژوال بیسیک فقط قاب آن را نمایش خواهد داد). این وظیفهٔ برنامه‌نویس است که وقتی کاربر کنترل را در نقطه‌ای رها کرد، آن کنترل را به نقطهٔ موردنظر منتقل کند. در حالت خودکار، با اینکه کاربر می‌تواند قاب کنترل را حرکت دهد، ولی انتقال آن بر عهدهٔ برنامه‌نویس است.

رهاشدن کنترل‌ها بر عهدهٔ رویداد DragDrop فرم است. برای اجرای عملیات، فقط باید خاصیت DragMode کنترل با مقدار Automatic–1 مقداردهی شود. روال Form_DragDrop()مسئول نیمهٔ دوم عملیات و انتقال کنترل به محل جدید است.

با آن که ویژوال بیسیک در حالت عادی، هنگام کشیدن و انتقال کنترل‌ها فقط قاب آن‌ها را نشان می‌دهد اما این وضع قابل اصلاح است؛ خاصیت DragIcon به نشانه‌ای اشاره می‌کند که هنگام کشیدن کنترل دیده خواهد شد. بعد از پایان عملیات کشیدن، انتقال کنترل به نقطهٔ جدید در روال Form_DragDrop() صورت خواهد گرفت. به کد زیر که عملیات در آن تکمیل می‌شود، توجه کنید :

```
Private Sub Form_DragDrop (Source As Control, X As Single, Y As Single)
    Source.Move X,Y        'Move to the dropped location
End Sub
```

برای انتقال کنترل از محل اولیه به محل جدید از متد Move استفاده شده است.

نکته

وقتی کاربر کنترلی را از روی کنترل دیگر عبور می‌دهد، رویداد DragOver روی خواهد داد. اگر می‌خواهید کاربر را از رها کردن کنترلی که در حال کشیدن آن است، روی کنترل مزبور منع کنید، در روال رویداد DragOver آن شکل اشاره‌گر را به vbNoDrop تغییر دهید. رویداد DragOver دارای چهار آرگومان است.

Private Sub Form_DragOver (source As Control، x As Single، y As Single، state As Integer)

- کنترل (source)
- مختص افقی ماوس (x)
- مختص عمودی ماوس (y)
- وضعیت کشیدن: ۰ (کنترل وارد محدودهٔ کنترل زیرین می‌شود)، ۱ (کنترل محدودهٔ کنترل زیرین را ترک می‌کند)، ۲ (کنترل در حال عبور از روی کنترل زیرین است). (State)

روش دوم :

- خاصیت DragMode باید با 0–Manual مقداردهی شود.
- کنترل به رویداد MouseDown پاسخ می‌دهد (روش خودکار چنین نیست) و مکان اولیهٔ کنترل می‌تواند ثبت شود.
- عملیات کشیدن در روال رویداد MouseDown شروع خواهد شد.

برای انجام عملیات کشیدن ــ رهاکردن در روال رویداد MouseDown می‌توانید از متد خاصی به نام Drag استفاده کنید. کد زیر نحوهٔ کشیدن یک کنترل تصویر را در حالت دستی نشان می‌دهد :

```
Private Sub imgMouse_MouseDown(Button As Integer, Shift As Integer, X
As Single, Y As Single)
    txtMouse. Text = "Click over the image at "& X &", "& Y imgMouse.Drag
End Sub
```

متد Drag سبب شروع عملیات کشیدن ــ رهاکردن خواهد شد. بدون این متد، روال

() MouseDown قادر به انجام عملیات نخواهد بود. دلیل عمدهٔ استفاده از عملیات دستی، محدود کردن حوزهٔ عملیات مطابق خواستهٔ برنامه‌نویس است (جدول ۸ــ۱).

جدول ۸ــ۱ــ ثابت‌های مورد استفاده در متد Drag

ثابت‌ها	مقدار	توضیحات
vbCancelDrag	۰	عمل کشیدن و رهاکردن لغو می‌شود. روال DragDrop فراخوانی نمی‌شود.
vbBeginDrag	۱	عمل کشیدن و رهاکردن شروع می‌شود. روال DragDrop فراخوانی می‌شود.
vbEndDrag	۲	عمل کشیدن و رهاکردن خاتمه می‌یابد. روال DragDrop فراخوانی می‌شود.

مشخصهٔ DragIcon در مدتی که عمل کشیدن و رهاکردن در حال انجام است تصویری به نمایش درمی‌آورد.

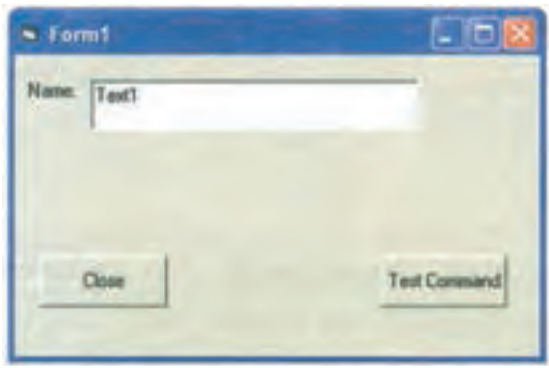
مثال ۲ــ۱

برنامهٔ زیر دارای یک کادر متن به نام Text1، یک برچسب به نام labell و یک کلید به نام cmdTestCommand است که به روش دستی قابل کشیدن هستند.

مراحل کار :

۱ــ برنامهٔ ویژوال بیسیک را اجرا کرده و پروژهٔ جدیدی ایجاد کنید.

۲ــ فرمی به‌صورت شکل ۴ــ۱ طراحی کنید و کد مربوط به دکمهٔ Close آن را نیز بنویسید.



شکل ۴ــ۱

۳ــ روی فرم دو بار کلیک کرده و در رویداد DragDrop فرم، دستورهای زیر را بنویسید :

```
Private Sub Form_DragDrop (Source As Control, X As Single, Y As Single)
        Source.Left = X
        Source.Top = Y
    End Sub
```

توجه

در این رویداد، آرگومان Source یک متغیر است و نام همان کنترلی را که عمل درگ روی آن رخ داده است، برمی‌گرداند. X و Y مختصات محلی است که عمل Drop در آن نقطه رخ داده است. درنتیجه به هنگام Drop دو فرمان فوق، کنترل مورد نظر را در محل رهاشدن ماوس قرار می‌دهند. یعنی گوشهٔ بالای سمت چپ کنترل در مختصات X و Y قرار می‌گیرد.

۴ــ رویدادهای MouseDown سه کنترل دیگر را نیز به‌صورت زیر بنویسید :

```
Private Sub CmdTestCommand_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
        Text1 = "Command Run Draging"
        CmdTestCommand.Drag
End Sub

Private Sub Labell_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
        Text1 = "Label Draging"
        Labell.Drag
End Sub

Private Sub Text1_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
```

```
        Text1 = "TextBox Draging"
        Text1.Drag
End Sub
```

۵ــ برنامه را اجرا کنید و سعی کنید این سه کنترل را جابه‌جا کنید.

مشاهده خواهید کرد که این سه کنترل جابجا می‌شوند ولی پس از رهاکردن دکمهٔ ماوس، گوشهٔ چپ و بالای کنترل در محل رهاشدن ماوس قرار می‌گیرد. برای رفع این اشکال، برنامه را به‌صورت زیر تغییر دهید :

```
Option Explicit
Dim intOldx As Integer, intOldy As Integer
Private Sub CmdClose_Click ()
        End
End Sub

Private Sub CmdTestCommand_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
        Text1 = "Command Run Draging"
        CmdTestCommand. Drag
        intOldx = X
        intOldy = Y
End Sub

Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
        Source.Left = X – intOldx
        Source.Top = Y – intOldy
        End Sub

Private Sub Labell_MouseDown (Button As Integer, Shift As Integer, X As
```

```
Single, Y As Single)
        Text1 = "Label Draging"
        Labell.Drag
        intOldx = X
        intOldy = Y
End Sub

Private Sub Text1_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
        Text1 = "TextBox Draging"
        Text1.Drag
        intOldx = X
        intOldy = Y
End Sub
```

توجه

- متغیرهایی که در بخش General تعریف شوند، در تمام رویدادها قابل دسترس هستند.
- در هر رویداد MouseDown، در انتها، مقادیر X و Y (همان مختصات محلی از کنترل، که ماوس در آن قرار دارد) در دو متغیر intOldx و intOldy ذخیره می‌شود تا در لحظهٔ Drop، این مقادیر از مقادیر X و Y نقطهٔ جدید کم شده و درنتیجه، کنترل به‌درستی سر جای واقعی خود قرار گیرد.

۶ــ برنامه را اجرا کنید و سعی کنید این سه کنترل را دوباره جابه‌جا کنید. مشاهده خواهید کرد که این سه کنترل به درستی جابه‌جا می‌شوند.

درصورتی که بخواهید هنگام کشیدن، شکل اشاره‌گر ماوس عوض شود، لازم است در پایان رویدادهای MouseDown هر سه کنترل، فرمان

نام کنترل.MousePointer=vbSizeAll

را اضافه کنید و برای بازگشت شکل اشاره‌گر به حالت اولیه لازم است در انتهای رویداد Form_DragDrop، فرمان Source.MousePointer=vbArrow را اضافه کنید.

لیست برنامه پس از انجام تغییرات، به‌صورت زیر است :

```
Option Explicit
Dim intOldx As Integer, intOldy As Integer
Private Sub CmdClose_Click()
    End
End Sub
Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
        Source.Left = X – intOldx
        Source.Top = Y – intOldy
        Source.MousePointer = vbArrow
End Sub
Private Sub CmdTestCommand_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
Text1 = "Command Run Draging"
CmdTestCommand.Drag
intOldx = X
intOldy = Y
CmdTestCommand. MousePointer = vbSizeAll
End Sub
Private Sub Labell_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
Text1 = "Label Draging"
Labell.Drag
intOldx = X
intOldy = Y
```

```
Label1.MousePointer = vbSizeAll
End Sub
Private Sub Text1_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
Text1 = "TextBox Draging"
Text1.Drag
intOldx = X
intOldy = Y
Text1.MousePointer = vbSizeAll
End Sub
```

اگر بخواهید از کشیدن خودکار استفاده کنید، لازم است به نکات زیر توجه کنید :

- مشخصهٔ DragMode مربوط به هر کنترل باید روی عدد ۱ یا vbAutomatic قرار گیرد.
- در حالت کشیدن خودکار، رویدادهای MouseDown و MouseUp غیرفعال می‌شوند و فقط رویداد MouseMove قابل استفاده است.
- نیازی به متد Drag نیست و این عمل به‌صورت خودکار انجام می‌شود.
- از مشخصهٔ DragIcon می‌توان برای تعیین شکل اشاره‌گر ماوس به هنگام کشیدن، استفاده کرد.
- رویداد DragDrop هر کنترل، هنگامی فراخوانی می‌شود که ماوس روی آن رها شود. (معادل رویداد MouseUp).

۳ــ۱ــ رویدادهای کلید

هنگامی که کاربر با صفحه کلید کار می‌کند، رویدادهای کلید (key events) ایجاد می‌شوند. با توجه به این‌که فرم یا کنترل فعال باشد پردازش این رویه به‌عهدهٔ آن شیء است. ویژوال بیسیک دارای سه روال رویدادهای کلید است : Keypress، KeyDown و KeyUp. روال‌های KeyPress و KeyDown زمانی که کلیدی فشار داده شود، فراخوانی می‌شوند. هنگام که کلید فشار داده شده رها شود، روال KeyUp فراخوانی خواهد شد. KeyPress متفاوت با KeyDown است. KeyPress نمی‌تواند تشخیص دهد که کدام کلید مانند Shift، Ctrl و Alt و غیره فشار داده شده است.

بسته به این که کدام کلید فشار داده شده باشد، هر سه روال ممکن است فراخوانی شوند. جدول ۹ــ۱ شامل لیستی از ثابت‌هاست که مربوط به رویدادهای کلید هستند.

برنامهٔ ۴ــ۱ نحوهٔ استفاده از رویدادهای کلید را نشان می‌دهد. هنگامی که کلیدی فشار داده شود، keyDown و KeyPress فراخوانی می‌شوند. از keyDown برای چاپ کلید فشار داده شده بر روی فرم و از KeyPress برای نمایش کلید فشار داده شده در نوار عنوان فرم استفاده می‌شود. همچنین KeyDown وضعیت کلیدهای Ctrl ، Shift و Alt را برای تغییر رنگ رویهٔ فرم بررسی می کند. KeyUp رنگ رویهٔ فرم را با رنگ سیاه تنظیم می کند.

جدول ۹ــ۱ــ ثابت‌های مربوط به رویدادهای کلید

ثابت‌ها	مقدار اسکی (ASCII)	توضیحات
vbKeyA–vbKeyZ	65–90	کلیدهای A تا Z
vbKeyNumpad0–vbKeyNumpad9	95–105	اعداد 0 تا 9 بخش عددی صفحه کلید
vbKey0–vbKey9	48–57	کلیدهای عددی 0 تا 9
vbKeyF1–vbKeyF16	112–127	کلیدهای تابعی F1 الی F16
vbKeyDecimal	110	کلید نقطه اعشار (کلید نقطه)
vbKeyBack	8	کلید Backspace
vbKeyTab	9	کلید Tab
vbKeyReturn	13	کلید Enter
vbKeyShift	16	کلید Shift
vbKeyControl	17	کلید Control
vbKeyCapital	20	کلید Caps Lock
vbKeyEscape	27	کلید Esc
vbKeySpace	32	Space bar
vbKeyInsert	45	کلید Insert
vbKeyDelete	46	کلید Delete

برنامهٔ ۴ــ۱ استفاده از رویدادهای کلید

```
1. ‘Demonstrating KeyDown, KeyUp, and, KeyPress
2. Option Explicit          ‘General declaration
3. Dim mTitleString As String    ‘General declaration
4. Private Sub Form_load ()
5.          ‘Store caption value for use in Keypress
6.          mTitleString = Caption & Space$ (5)
7.  End Sub
8. Private Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)
9.          ‘Determine which, if any, of the Shift, Ctrl,
10.         ‘or Alt keys is pressed
11.         Select Case Shift
12.         Case vbShiftMask       ‘Shift
13.         ForeColor = vbYellow
14.         Case vbAltMask         ‘Alt
15.         Fore Color = vbRed
16.         Case vbCtrlMask          ‘Ctrl
17.         ForeColor = vbGreen
18.          Case vbShiftMask + vbAltNask ‘Shift + Alt
19.         ForeColor = vbBlue
20.         Case vbShiftMask + vbCtrlMask          ‘Shift + Ctrl
21.          ForeColor = vbMagenta
22.          Case vbAltMasl + vbCtrlMask       ‘Alt + Ctrl
23.         ForeColor = vbCyan
24.         Case vbAltMask + vbCtrlMask + vbShiftMask ‘All three
25.          Cls
26.         End Select
```

```
27.         'Test for letter key
28.      If KeyCode > = vbKeyA And KeyCode < = vbKeyZ Or KeyCode =
         vbKeySpace Then
29.          Print Chr (KeyCode);    'Print the character
30.      ElseIf KeyCode = vbKeyreturn Then    'Return key
31.          Print    'Print on next line
32.       End If
33. End Sub
34. Private Sub Form_KeyPress (KeyAscii as Integer)
35.              'Update title to display the key Pressed
36.      Caption = mTitleString & "("& Chr (key Ascii) & ")"
37. End Sub
38. Private Sub Form_KeyUp(KeyCode as Integer, Shift As Integer)
39. When key is released, change ForeColor to black
40.   ForeColor = vbBlack
41. End Sub
```

روال KeyDown با استفاده از متغیر Shift، وضعیت کلیدهای Shift، Ctrl و Alt را در زمان رخ دادن رویداد KeyDown مشخص می‌کند. اگر هریک از این کلیدها فشار داده شوند، این رویداد اتفاق می‌افتد و رنگ رویه تغییر پیدا می‌کند. پارامتر KeyCode شامل یک مقدار اسکی است که نشان‌دهندهٔ یک کلید فشار داده شده است. برای تشخیص حرف، از KeyCode استفاده شده است. KeyCode بین حروف کوچک و بزرگ تفاوت قایل می‌شود. از تابع Chr برای تبدیل کدهای اسکی به نویسه استفاده شده است (مثلاً ۶۵ به "A" تبدیل می‌شود). در روال keyPress هم از Chr برای تبدیل مقدار اسکی به رشته استفاده شده است. متغیر KeyAscii بین حروف کوچک و بزرگ تفاوت قایل است. روال KeyPress شامل پارامتر Shift نیست. هرگاه KeyUp مربوط به فرم رخ دهد، دوباره رنگ رویه به رنگ سیاه تغییر می‌یابد. با فشار کلیدهای Shift، Ctrl و Alt نیز رنگ رویه تغییر می‌کند ولی با فشار همزمان هر سه کلید، فرم پاک خواهد شد (شکل ۵ــ۱).

نکته

فقط نویسه‌های A تا Z و فضای خالی روی فرم چاپ می‌شوند و علایم فقط در نوار عنوان فرم مشاهده می‌شوند.

شکل ۵ــ۱

۱ــ۳ــ۱ــ مشخصهٔ KeyPreview : اگر فرمی شامل حداقل یک کنترل قابل مشاهده (در این‌جا کنترلی مد نظر است که مشخصهٔ Visible آن با True تنظیم شده باشد) یا یک کنترل فعال باشد، آن فرم نمی‌تواند به رویدادهای کلید واکنش نشان دهد. برای تغییر این وضعیت، باید مشخصهٔ KeyPreview فرم با True مقداردهی شود تا رویدادهای کلید قبل از آن‌که به وسیلهٔ کنترل‌ها دریافت شوند، به وسیلهٔ فرم دریافت شوند.

معمولاً از KeyPreview هنگامی استفاده می‌شود که چندین کنترل نیاز به یک واکنش در برابر رویدادهای کلید داشته باشند.

برنامهٔ ۵ــ۱ نحوهٔ استفاده از KeyPreview را نشان داده است. برنامه با استفاده از این مشخصه مطمئن می‌شود که فقط رقم‌ها وارد کادر متن می‌شوند.

در روال Form_Load مشخصهٔ KeyPreview با True مقداردهی می‌شود. با این عمل، فرم می‌تواند جلوی رویدادهای کلید را قبل از آنکه به‌صورت خودکار، به روال KeyPress کادر متن یا کادرهای متن انتقال پیدا کنند، بگیرد. هنگامی که کاربر دادهٔ مورد نظرخود را از طریق کادر متن وارد می‌کند، روال KeyPress فرم فراخوانی می‌شود و عمل بررسی بر روی نویسه‌های رقم انجام می‌گیرد (شکل ۶ــ۱).

شکل ۶ــ۱

برنامهٔ ۵ــ۱ استفاده از مشخصهٔ KeyPreview

```
1.  'Demontrating  the KeyPreview property
2.  Option Explicit        'General declaration
3.  Private Sub form_Load ()
4.      'Allow Form to get key events first
5.      Keypreview = True
6.  End Sub
7.  Private Sub Form_KeyPress (KeyAscii As Integer)
8.      'Only allow numeric keys
9.      If KeyAscii>=vbKey0 And KeyAscii<=vbKey9 Then
10.         txtInput.Text = txtInput.Text & Chr (key Ascii)
11.     End If
12.     Ens Sub
13.     Private Sub txtInput_KeyPress (KeyAscii As Integer)
14.         Key Ascii = 0        'Disable event handling
15.     End Sub
16.     Private Sub txtInput_KeyUp (KeyCode As Integer, Shift As Integer)
17.         Selcet Case Int (Rnd() * 3)
18.         Case 0
19.         txtInput.BackColor = vbYellow
20.         Case 1
21.         txtInput.BackColor = vbCyan
22.         Case 2
23.         txtInput.BackColor =  vbRed
24.         End Select
25.  End Sub
```

برای جلوگیری از فراخوانی‌های اضافی روال KeyPress کادر متن، مکانیزم دستیابی به این

روال را به‌صورت KeyAscii=0 غیرفعال کرده‌ایم. هرگاه KeyAscii=0 شود، مجوز عبور کلید فشار داده شده سلب می‌شود ولی در مورد KeyCode چنین نیست. روال KeyUp به‌صورت تصادفی رنگ زمینهٔ متن را تغییر می‌دهد. زمانی که KeyUp غیرفعال نشده باشد، در هر بار رهاکردن کلید، این روال فراخوانی می‌شود.

هرگاه یکی از کلیدهای زیر فشار داده شود، رویداد KeyPress فراخوانی خواهد شد :

- حروف بزرگ یا کوچک
- اعداد
- علایم نقطه‌گذاری
- Backspace و Tab، Enter

رویداد Keypress، اکثر نویسه‌های اسکی را تشخیص می‌دهد (ولی نمی‌تواند نویسه‌های ۰ تا ۳۱ جدول اسکی را تشخیص دهد). رویداد Keypress با فشاردادن کلید فراخوانی می‌شود و اگر کاربر دستش را روی کلید نگه دارد، با تکرار نویسه‌ها این رویداد هم تولید خواهد شد.

همانطورکه می‌دانید، هر رویداد به یک شیء وابسته است. رویداد KeyPress هم به کنترلی که فوکوس را در اختیار دارد، وابسته است. اگر هیچ کنترلی فوکوس نداشته باشد، این رویداد به فرم خواهد رسید.

رویداد KeyPress دارای یک آرگومان از نوع Integer است. رویداد Key Press برای یک کادر متن به صورت زیر است :

```
Private Sub Text_KeyPress (KeyAscii As Integer)
    ‘Code goes here test and respond to keystroke
End Sub
```

آرگومان KeyAscii معادل کد اسکی کلید زده شده خواهد بود و با یک دستور If یا Case Select می‌توان آن را بررسی کرد.

یکی از مهمترین کارهایی که می‌توان با KeyPress انجام داد، تغییردادن کلید زده شده به وسیلهٔ کاربر است، چون کلید زده شده قبل از اینکه به کنترل‌ها برسد از این رویداد عبور خواهد کرد. به‌عنوان مثال، به یاد دارید که وقتی فوکوس در اختیار یک کادر متن است، هر کلیدی که کاربر بزند، بلافاصله در کادر متن ظاهر خواهد شد. اما با رویداد KeyPress می‌توانید روالی بنویسید که کلید زده شده را عوض کند :

```
Private Sub txtTrylt_KeyPress (KeyAscii As Integer)
   ‘ Change any uppercase A to an uppercase B
If KeyAscii = 65 Then    ‘65 is ASCII for A
    Key Ascii = 66        ‘66 is ASCII for B
End If
End Sub
```

این کد سبب می‌شود که هرگاه کاربر A را بزند، کادر متن B را نشان دهد! (حروف دیگر بدون هیچ تغییری نشان داده خواهند شد). لازم به ذکر است که نیازی نیست کد اسکی کلیدها را حفظ کنید چون ویژوال بیسیک برای تمام آن‌ها ثابت‌های نام‌دار تعریف کرده است. به‌عنوان مثال، برای Backspace ثابت vbKeyBack و برای کلید Enter ثابت vbKeyReturn و برای کلید Tab ثابت vbKeyTab (و الی آخر) تعریف شده‌اند. که در جدول ۹ــ۱ آن‌ها را مشاهده کردید.

رویداد KeyDown (که با پایین‌رفتن کلید، تحریک می‌شود) مانند KeyPress است ولی جزییات بیشتری در اختیار برنامه‌نویس قرار می‌دهد. به‌عنوان مثال، رویداد KeyPress برای کلیدهای T یا t کدهای متفاوتی برمی‌گرداند، ولی رویداد KeyDowm برای این دو کلید کد یکسانی برمی‌گرداند و آرگومانی دارد که حالت کلید Shift، Ctrl و Alt را نشان می‌دهد. کار با رویداد KeyPress برای کنترل کلیدهای اسکی ساده‌تر است. بنابراین تا حد امکان از آن استفاده کنید. شکل کلی روال رویداد KeyDown چنین است :

```
Private Sub txtTryIt_KeyDown (KeyCode As Integer, Shift As Integer)
        ‘Keyboard code handler goes here
End Sub
```

آرگومان KeyCode، کد کلیدزده شده و آرگومان Shift، وضعیت کلیدهای Shift، Ctrl و Alt را در خود خواهند داشت. آرگومان KeyCode همیشه معادل حروف بزرگ کلید زده شده است. اگر دقت نکنید، این وضعیت می‌تواند موجب سردرگمی شما شود.

مزیت رویداد KeyDown (علیرغم مشکل فوق) این است که با آن می‌توان تقریباً تمام کلیدهای صفحه کلید را تشخیص داد : کلیدهایی مانند Home، End، کلیدهای فلش‌دار و غیره. آرگومان Shift نوع کلید Shift، Ctrl یا Alt را مشخص خواهد کرد.

وقتی کاربر کلید فشار داده شده را رها کند، رویداد KeyUp فعال خواهد شد. در این رویداد

هم می‌توان وضعیت کلیدهای ترکیبی را بررسی کرد. توجه دارید که هر رویداد KeyPress ترکیبی از رویدادهای KeyDown و KeyUp است.

۲ــ۳ــ۱ــ ارسال ضربات کلید به برنامه : دستور SendKeys می‌تواند ضربات کلید را به برنامه بفرستد. از نظر برنامه، هیچ تفاوتی بین دستور SendKeys و ضربات واقعی صفحه کلید وجود ندارد. شکل کلی این دستور چنین است :

```
SendKeys strKeystrokes [,blnWait]
```

رشتهٔ strKeystrokes، همان نویسه‌هایی است که باید ارسال شوند. اگر آرگومان منطقی blnWait (که معمولاً ذکر نمی‌شود) False باشد (مقدار پیش‌فرض)، روال اجراکنندهٔ دستور SendKeys بلافاصله بعد از ارسال نویسه‌ها مجدداً کنترل برنامه را در دست می‌گیرد. اما اگر این آرگومان True باشد، سیستم تا پردازش کامل نویسه‌ها، کنترل را به روال اجراکنندهٔ دستور SendKeys بازپس نخواهد داد.

برای ارسال نویسه‌های خاص (^، +، ~، (،)، {}، %) باید آن‌ها را داخل یک زوج آکولاد {} قرار دهید. به‌عنوان مثال، اگر می‌خواهید تایپ ۶+۷ را برای برنامه شبیه سازی کنید، باید از دستور SendKeys زیر استفاده کنید :

```
SendKeys "7{+}6"
```

یا برای ارسال کلید Home به برنامه، باید چنین بنویسید :

```
SendKeys "{Home}"
```

تنها کلیدی که نمی‌توان با SendKeys آن را به برنامه‌ها فرستاد، کلید Print Screen است. جدول ۱۰ــ۱ کلیدهای ویژه را نشان می‌دهد.

جدول ۱۰ــ۱

کلیدهای ویژه	عبارت معادل
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}

Down Arrow	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INSERT}or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}

F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

برای استفاده از کلیدهای Shift، Ctrl و Alt باید از جدول ۱۱ــ۱ استفاده کرد :

جدول ۱۱ــ۱

Key	Code
SHIFT	+
CTRL	^
ALT	%

به‌عنوان مثال، "(EC)+" یعنی فشاردادن کلید Shift به همراه کلیدهای E و C و "EC+" به معنی فشاردادن کلید Shift به همراه کلید E و پس از رهاکردن، فشاردادن کلید C است.

برای تکرار یک کلید، لازم است از مدل {key number} استفاده کنیم که در آن Key همان کلید دلخواه و number تعداد دفعات تکرار است. توجه داشته باشید که باید بین Key و number فاصله وجود داشته باشد.

به‌عنوان مثال، {A 5} یعنی فرستادن ۵ بار نویسهٔ A ، {Right 10} یعنی فرستادن کلید جهت‌دار راست به تعداد ۱۰ مرتبه.

دریافت داده‌ها از ورودی و عملیات شرطی

در این مثال با نحوهٔ استفاده از کنترل‌های کادر علامت، دکمهٔ انتخاب و فریم و متدهای Show

و Hide در یک برنامهٔ چندفرمی آشنا خواهید شد. گرفتن اطلاعات از کاربر و کنترل پاسخ‌های وی هم از اهداف آموزشی این مثال است. این مثال به کدنویسی زیادی احتیاج دارد. در این پروژه :

- چند کادر علامت وجود دارد که کاربر با انتخاب آن‌ها می‌تواند تا پرچم کشورهای مربوطه را مشاهده کند.
- کاربر می‌تواند همان وظیفهٔ فوق را با دکمه‌های انتخاب انجام دهد.
- اگر کاربر به یک درخواست، پاسخ صحیح ندهد، با یک کادر پیام به وی اخطار داده می‌شود.
- کاربر می‌تواند نحوهٔ نمایش (بزرگ و کوچک) پرچم‌ها را با یک سری دکمهٔ انتخاب تغییر دهد.

این فرم از پرونده‌های گرافیکی Visual Basic 6 استفاده می‌کند و اگر هنگام نصب Visual Basic 6 آن‌ها را در هارددیسک خود کپی نکرده اید، باید مسیر پرونده را با محل اصلی آن‌ها (CD نصب ویژوال بیسیک) تنظیم کنید.

فرم اولیهٔ برنامه

شکل ۷ــ۱ اولین فرم برنامه را نشان می‌دهد. جدول ۱۲ــ۱ مشخصه‌های کنترل‌های فرم اولیهٔ برنامه را ارایه می‌دهد.

شکل ۷ــ۱

جدول ۱۲ــ۱ــ مشخصه‌های کنترل‌های فرم اولیهٔ برنامه

مشخصه	مقدار
Form Name	FrmSelect
Form caption	Flag Selection
Label Name	lblFlags
Label BorderStyle	1–Fixed Single
Label caption	Flage
Label Font	MS Sans Serif
Label Font Size	24
Label Font Style	Bold
Option Button #1 Name	optCheck
Option Button #1 caption	& Check Boxes
Option Button #2 Name	optOption
Option Button #2 caption	& Option Buttons
Command buttom #1 Name	cmdSelect
Command buttom #1 Caption	Click when & Ready
Command buttom #2 Name	cmdExit
Command buttom #2 Caption	E&xit

کد این فرم، به‌صورت زیر است :

```
1. Private Sub cmdSelect_Click ()
2.     Dim strMsg As String
3.     'Holds message box return value
4. If ((optCheck.Value = False) And (optOption.Value = False)) Then
5. strMsg = MsgBox ("You need to select an option, try again", _
```

```
6.                  vbCritical, "Error")
7.        ElseIf (optCheck.Value = True) Then
8.             frmFlageCheck.Show     'Flage with Check boxes
9.        Else
10.       frmFlagesOpt.Show          'Flage with Option buttons
11.  End If
12.  End Sub
13.  Private Sub Form_Load ()
14.       'Clear each of the option buttons
15.       optCheck.Value = False
16.       optOption.Value = False
17.  End Sub
18.  Private Sub  cmdExit_Click ()
19.       'Stop the program
20.       End
21.  End Sub
```

تحلیل کد فرم اولیه : خطوط ۱۴ تا ۱۸ برنامهٔ فوق، اعمالی هستند که هنگام اجرای برنامه و بارشدن اولین فرم آن انجام می‌شوند (توجه داشته باشید که باید در کادر محاوره‌ای Project | Properties فرم FrmSelect به‌عنوان فرم Startup برنامه تنظیم شده باشد). این روال خاصیت Value هر دو دکمهٔ انتخاب این فرم را False می‌کند تا کاربر مجبور به انتخاب یکی از آن‌ها شود. به عبارت دیگر، در این فرم گزینهٔ پیش‌فرض وجود ندارد.

اگر کاربر بدون انتخاب یکی از این گزینه‌ها، دکمهٔ فرمان cmdSelect را کلیک کند، دستور If خط ۴، متوجه این وضعیت شده و به کاربر پیام می‌دهد که بایستی یکی از گزینه‌ها را انتخاب کند.

اما اگر به درستی عمل کرده باشد و یکی از گزینه‌ها را انتخاب و سپس cmdSelect را کلیک کرده باشد، خط‌های ۷ تا ۹ تعیین می‌کنند که کدام گزینه انتخاب شده است، چون فرمی که باید نمایش داده شود به گزینهٔ انتخاب شده در فرم اولیهٔ برنامه بستگی دارد. برای نمایش فرم‌ها از متد Show استفاده می‌شود. این متد فرمانی است که فقط روی شیء فرم عمل می‌کند و نحوهٔ استفاده از آن مانند سایر متدهاست.

ویژوال بیسیک با دیدن متد Show، فرم مشخص شده را در حافظه بارکرده و نمایش می‌دهد.

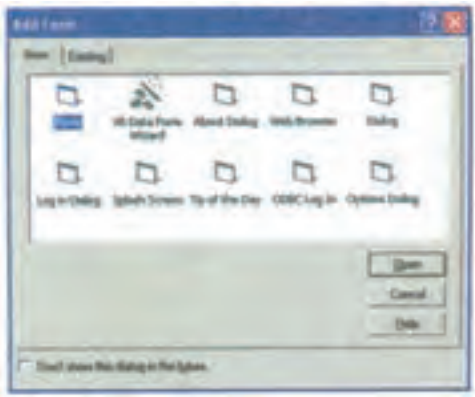
فرم کادر علامت : در شکل ۸ــ۱ فرم دوم برنامه یعنی فرمی که با استفاده از کادر علامت، پرچم‌ها رانمایش می‌دهد، نشان داده شده است.

شکل ۸ــ۱

در این فرم، شش کادر علامت وجود دارد که هریک پرچم کشوری را نمایش می‌دهند. اما برای ایجاد یک فرم جدید در پروژه :

۱ــ داخل پنجرهٔ پروژه کلیک راست کنید.

۲ــ از منوی ظاهر شده، گزینهٔ Add|Form را انتخاب کنید؛ کادر محاوره‌ای ظاهر می‌شود که در آن می‌توانید یک فرم جدید (یا یک فرم موجود) را به پروژه اضافه کنید (شکل ۹ــ۱).



شکل ۹ــ۱

۳ــ روی نشانهٔ Form دو بار کلیک کنید تا فرم جدید به برنامه اضافه شود.

در جدول ۱۳ــ۱ مشخصه‌های کنترل‌های روی این فرم را مشاهده می‌کنید. به یاد داشته باشید که مسیر پرونده مشخصهٔ Picture کنترل‌های تصویر را باید مطابق آنچه در رایانه‌تان وجود دارد، مقداردهی کنید.

جدول ۱۳ــ۱ــ مشخصه‌های کنترل‌های فرم کادر علامت

مشخصه	مقدار
Check box #1 Name	chkBrazil
Check box #1 Caption	& Brazil
Check box #2 Name	chkItaly
Check box #2 Caption	& Italy
Check box #3 Name	chkSpain
Check box #3 Caption	& Spain
Check box #4 Name	chkMexico
Check box #4 Caption	& Mexico
Check box #5 Name	chkportegal
Check box #5 Caption	& portegal
Check box #6 Name	chkNORW
Check box #6 Caption	& NORW
Image #1 Name	imgBrazil
Image #1 Picture	\Progran Files\Microsoft Visual Studio\ common\Graphics\Icons\Flags\FlagBrzl
Image #1Visible	False
Image # 2 Name	ImgItaly
Image #2 Picture	\Program Files\Microsoft Visual Studio\ common\Graphics\Icons\Flags\FlagItaly

Image #2Visible	False
Image #3 Name	ImgSpain
Image #3 Picture	\Progran Files\Microsoft Visual Studio\ common\Graphics\Icons\Flags\FlagPain
Image #3Visible	False
Image #4 Name	imgMexico
Image #4 Picture	\Program Files\Microsoft Visual Studio\ common\Graphics\Icons\Flags\Flagmex
Image #4 Visible	False
Image #5 Name	ImgPortegal
Image #5 Picture	\Program Files\Microsoft Visual Studio\ common\Graphics\Icons\Flags\FlagPort
Image #5 Visible	False
Image #6 Name	ImgNORW
Image #6 Picture	\Program Files\Microsoft Visual Studio\ common\Graphics\Icons\Flags\'FlagsNorw
Image #6 Visible	False
Comm and buttonName	cmdReturn
command button Caption	& Return to selection

حال روی فرم frmFlagCheck دو بار کلیک کرده و کد برنامهٔ ۶ــ۱ را در پنجرهٔ کد این فرم بنویسید. چون با کلیک‌کردن هر کادر علامت، باید تصویر متناظر با آن به‌طور متناوب ظاهر و پنهان شود. برای رویداد Click هر کادر علامت، یک روال اختصاصی نوشته شده است.

برنامهٔ ۶ــ۱ کد فرم کادر علامت

```
1. Private Sub chkBrazil_Click()
2. 'Displays the flag if checked
```

```
3. If chkBrazil.Value = 1 Then
4.         imgBrazil.Visible = True
5. Else
6.         imgBrazil.Visible = False
7. End If
8. End Sub
9. Private Sub chkItaly_Click()
10.        'Displays the flag if checked
11.         If chkItaly.Value = 1 Then
12.            imgItaly.Visible = True
13.         Else
14.            imgItaly.Visible = False
15.  End If
16.  End Sub
17.  Private Sub chkportegal_Click ()
18.        'Displays the flag if checked
19.      If chkportegal.Value = 1 Then
20.         imgportegal.Visible = True
21.     Else
22.        imgportegal.Visible = False
23. End If
24. End Sub
25.  Private Sub chkMexico_Click()
26. 'Displays the flag if checked
27. If chkMexico.Value = 1 Then
28.         imgMexico.Visible = True
29.   Else
```

```
30.      imgMexico. Visible = False
31.  End If
32.  End Sub
33.  Private Sub chkSpain_Click ()
34.  'Displays the flag if checked
35.  If chkSpain.Value = 1 Then
36.      imgSpain.Visible = True
37.  Else
38.      imgSpain.Visible = False
39.  End If
40.  End Sub
41.  Private Sub chkNORW_Click ()
42.  'Displays the flag if checked
43.  If chkNORW.Value = 1 Then
44.      imgNORW.Visible = True
45.  Else
46.      imgNORW.Visible = False
47.      End If
48.      End Sub
49.      Private Sub cmdReturn_Click()
50.          frmFlageCheck.Hide
51.          FrmSelect.Show
52.      End Sub
```

تحلیل کد فرم کادر علامت : در برنامهٔ ۶ــ۱ شش روال رویداد کاملاً مشابه وجود دارد و کار آن‌ها فقط مقداردهی کردن مشخصهٔ Visible کنترل تصویر متناظر با هر کادر علامت است. به دستور If دقت کنید : اگر کاربر روی کادر علامتی که فعال است کلیک کند، قسمت Else اجرا شده وسبب ناپدیدشدن تصویر خواهد شد. هر روال رویداد برای اطلاع از وضعیت فعلی کادر علامت به

مقدار مشخصهٔ Value آن توجه می‌کند : اگر این خاصیت ۱ باشد، تصویر ظاهر و اگر ۰ باشد، تصویر پنهان می‌شود. علاوه‌بر روال‌های کادر علامت، روال cmdReturn_Click نیز در این برنامه مشاهده می‌شود. این روال، فرم کادر علامت را از روی صفحه ناپدید کرده و مجدداً فرم frmSelect را نمایش می‌دهد. وظیفهٔ متد Hide عکس متد Show است.

فرم دکمهٔ انتخاب : در شکل ۱۰ــ۱ فرم سوم پروژه، یعنی فرمی که با دکمه‌های انتخاب، پرچم‌ها را نمایش می‌دهد، نشان داده شده است. این فرم یک امکان دیگر هم در اختیار کاربران قرار می‌دهد : پرچم‌های بزرگ یا پرچم‌های کوچک.

شکل ۱۰ــ۱

این فرم هم مانند فرم کادر علامت، دارای شش دکمهٔ انتخاب (برای هریک از کشورها) است. علاوه‌برآن، این فرم دارای یک فریم با دو دکمهٔ انتخاب برای تعیین اندازهٔ نمایش پرچم‌هاست.

فرم سوم را هم مانند قبل بسازید و مشخصه‌های کنترل‌های آن را مانند جدول ۱۴ــ۱ مقداردهی کنید.

جدول ۱۴ــ۱ــ خواص کنترل‌های فرم دکمهٔ انتخاب

مشخصه	مقدار
Form Name	frmFlagsOpt
Form Caption	Flags
Option button # 1 Name	optBrazil
Option button # 1 Caption	& Brazil
Option button # 1 Value	True
Option button # 2 Name	optItaly
Option button # 2 Caption	& Italy
Option button # 3 Name	optSpain
Option button # 3 Caption	& Spain
Option button # 4Name	optMexico
Option button # 4 Caption	& Mexico
Option button # 5 Name	optportegal
Option button # 5 Caption	& portegal
Option button # 6 Name	optNORW
Option button # 6 Caption	& NORW
Frame Name	fraSize
Frame Caption	Size the flag
Frame Option # 1 Name	optLarge
Frame Option # 1 Caption	& Large
Frame Option # 2 Name	optSmall
Frame Option # 2 Caption	Sma&ll
Image # 1 Name	imgBrazil

Image # 1 Picture	\Program\Microsoft Visual Studio\Common\ Graphics\Icons\Flags\FlgBrzil
Image # 1 Stretch	True
Image # 1 Visible	True
Image # 2 Name	ImgItaly
Image # 2 Picture	\Program\Microsoft Visual Studio\Common\ Graphics\Icons\Flags\Flgitaly
Image # 2 Stretch	True
Image # 2 Visible	False
Image # 3 Name	ImgSpain
Image # 3 Picture	\Program\Microsoft Visual Studio\Common\ Graphics\Icons\Flags\FlgSpain
Image # 3 Stretch	True
Image # 3 Visible	False
Image # 4 Name	ImgMexico
Image # 4 Picture	\Program\Microsoft Visual Studio\Common\ Graphics\Icons\Flags\Flgmex
Image # 4 Stretch	True
Image # 4 Visible	False
Image # 5 Name	Imgportegal
Image # 5 Picture	\Program\Microsoft Visual Studio\Common\ Graphics\Icons\Flags\Flgport
Image # 5 Stretch	True

Image # 5 Visible	False
Image # 6 Name	ImgNORW
Image # 6 Picture	\Program\Microsoft Visual Studio\Common\ Graphics\Icons\Flags\FlgNorw
Image # 6 Stretch	True
Image # 6 Top	4080
Image # 6 Visible	False
Command button Name	Cmdreturn
Command button Caption	& Return to selection

کد فرم frmFlagsOpt را در برنامهٔ ۷ــ۱ مشاهده می‌کنید. این کد کمی طولانی است! ولی بیشترین بخش آن روال‌های تکراری کلیک‌شدن روی دکمه‌های انتخاب است.

برنامهٔ ۷ــ۱ کد فرم دکمهٔ انتخاب

```
1. Private Sub optBrazil_Click()
2.       'Displays the flag if checked
3.    If optSmall.Value = True Then
4.       imgBrazil.Height = 480
5.       imgBrazil.Width = 480
6.    Else    'Large image
7.       imgBrazil.Height = 2800
8.       imgBrazil.Width = 2800
9.    End If
10.   imgBrazil.Visible = True
11.   'Turn off display of all other flags
12.   imgItaly.Visible = False
13.   imgSpain.Visible = False
```

```
14.     imgMexico.Visible = False
15.     imgportegal.Visible = False
16.     imgNORW.Visible = False
17. End Sub
18. Private Sub optItaly_Click()
19.     'Displays the flag if checked
20.     If optSmall.Value = True Then
21.         imgItaly.Height = 480
22.         imgItaly.Width = 480
23.     Else     'Large image
24.         imgItaly.Height = 2800
25.         imgItaly.Width = 2800
26.     End If
27.     imgItaly.Visible = True
28.     'Turn off display of all other flags
29.     imgBrazil.Visible = False
30.     imgSpain.Visible = False
31.     imgMexico.Visible = False
32.     imgportegal.Visible = False
33.     imgNORW.Visible = False
34. End Sub
35. Private Sub opts Pain_Click()
36.     'Display the flag if checked
37.     If optSmall.Value = True Then
38.         imgSpain.Height = 480
39.         imgSpain.Width = 480
40.     Else     'Large image
41.         imgSpain.Height = 2800
```

```
42.          imgSpain.Width = 2800
43.     End If
44.      imgSpain.Visible = True
45.     'Turn off display of all other flags
46.      imgItaly.Visible = False
47.     imgBrazil.Visible = False
48.      imgMexico.Visible = False
49.     imgportegal.Visible = False
50.      imgNORW.Visible = False
51. End Sub
52. Private Sub opMexico_Click()
53.     'Displays the flag if checked
54.     If optSmall.Value = True Then
55.          imgMexico.Height = 480
56.          imgMexico.Width = 480
57.     Else     'Large image
58.          imgMexico.Height = 2800
59.          imgMexico.Width = 2800
60.     End If
61.     imgMexico.Visible = True
62.     'Turn off display of all other flags
63.     imgItaly.Visible = False
64.      imgSpain.Visible = False
65.      imgBrazil.Visible = False
66.     imgportegal.Visible = False
67.     imgNORW.Visible = False
68. End Sub
69. Private Sub optportegal_Click()
```

```
70.      'Displays the flag if checked
71.     If optSmall.Value = True Then
72.           imgportegal.Height = 480
73.           imgportegal.Width = 480
74.     Else    'Large image
75.           imgportegal.Height = 2800
76.           imgportegal.Width = 2800
77.     End If
78.    imgportegal.Visible = True
79.     'Turn off display of all other flags
80.     imgItaly.Visible = False
81.     imgSpain.Visible = False
82.     imgMexico.Visible = False
83.     imgBrazil.Visible = False
84.     imgNORW.Visible = False
85. End Sub
86. Private Sub optNorw_Click()
87.    'Displays the flag if checked
88.     If optSmall.Value = True Then
89.           imgpNORW.Height = 480
90.           imgNORW.Width = 480
91.    Else    'Large image
92.          imgNORW.Height = 2800
93.          imgNORW.Width = 2800
94.    End If
95.     imgNORW.Visible = True
96.     'Turn off display of all other flags
97.     imgItaly.Visible = False
```

```
98.     imgSpain.Visible = False
99.     imgMexico.Visible = False
100.    imgportegal.Visible = False
101.    imgBrazil.Visible = False
102. End Sub
103. Private Sub cmdReturn_Click()
104.    'Return to the selection form
105.    frmFlagsOpt. Hide
106.    frmSelect. Show
107. End Sub
108. Private Sub optSmall_Click()
109.        'Hide all flags shown
110.        'Subsequent flags will be small
111.        imgBrazil.Visible = False
112.        imgItaly.Visible = False
113.        imgSpain.Visible = False
114.        imgMexico.Visible = False
115.        imgportegal.Visible = False
116.        imgNORW.Visible = False
117.        'Reset option buttons
118.        optBrazil.Value = False
119.        optItaly.Value = False
120.        optSpain.Value = False
121.        optMexico.Value = False
122.        optportegal.Value = False
123.        optNORW.Value = False
124. End Sub
125. Private Sub optLarge_Click()
```

```
126.        'Hide all flags shown
127.        'Subsequent flags will be large
128.        imgBrazil.Visible = False
129.        imgItaly.Visible = False
130.        imgSpain.Visible = False
131.        imgMexico.Visible = False
132.        imgportegal.Visible = False
133.        imgNorw.Visible = False
134.        'Reset option buttone
135.        optBrazil.Value = False
136.        optItaly.Value = False
137.        optSpain.Value = False
138.        optMexico.Value = False
139.        optportegal.Value = False
140.        optNORW.Value = False
141. End Sub
```

تحلیل کد فرم دکمهٔ انتخاب : قسمت اعظم کد تکراری است. به خط ۱ تا ۱۷ آن دقت کنید. در اینجا ابتدا روال بررسی می‌کند که آیا تصویر باید بزرگ باشد یا کوچک و برنامه با مقداردهی Height و Width کنترل تصویر، اندازهٔ مناسب را انتخاب می‌کند. ادامهٔ روال به ظاهرکردن تصویر مربوطه (خط ۱۰) و ناپدیدکردن سایر تصویرها (خط‌های ۱۲ تا ۱۶) اختصاص دارد. سایر روال‌ها دقیقاً مشابه این روال هستند.

نکته

در ادامه کتاب خواهید آموخت که با نوشتن توابع می‌توان این کد را ساده‌تر کرد.

خط‌های ۱۰۳ تا ۱۰۷ روال پنهان‌کردن این فرم و ظاهرکردن دوباره فرم اصلی برنامه است. هنگام کلیک‌کردن دکمه‌های OptSmall و OptLarge، برنامه (برای راحتی کار خود) تمام گزینه‌ها را False و تمام پرچم‌ها را ناپدید می‌کند.

خلاصه فصل

در زبان برنامه نویسی ویژوال بیسیک، می‌توان به کمک کنترل‌های Image و PictureBox از تصویرهای با قالب‌های مختلف استفاده کرد. با استفاده از تابع LoadPicture می‌توان تصویری را در مشخصهٔ Picture این کنترل‌ها قرار داد.

زمانی که برنامه‌های کاربردی ایجاد می‌کنید، نیاز است که برنامه نسبت به رویدادهای ماوس و صفحه کلید رفتارهای خاصی داشته باشد. رویدادهای ماوس، به شرح زیر هستند :

- جابه‌جایی ماوس
- کلیک (click)
- دوبارکلیک (double-click)
- کلیک راست (right-click)
- عملیات کشیدن ــ رهاکردن (Drag & Drop)

با استفاده از مشخصهٔ MousePointer می‌توان شکل ظاهری اشاره‌گر ماوس را تغییر داد.

ترتیب رویدادهای ماوس از نظر ایجاد آن‌ها به وسیلهٔ ویندوز چنین است :

۱ــ MouseDown

۲ــ MouseUP

۳ــ Click

۴ــ Dblclick

۵ــ MouseUp

رویدادهای KeyDown ،Keypress و KeyUp مربوط به صفحه کلید هستند.

از مشخصهٔ KeyPreview زمانی استفاده می‌کنیم که بخواهیم فرمی با داشتن یک یا چند کنترل، به رویدادهای صفحه کلید پاسخ دهد. معمولاً از KeyPreview هنگامی استفاده می‌شود که چندین کنترل، نیاز به یک واکنش در برابر رویدادهای کلید داشته باشند.

برای ارسال ضربات کلید به برنامه، از دستور SendKeys استفاده کنید.

خودآزمایی

۱ــ برنامه‌ای بنویسید که بدون استفاده از خاصیت PasswordChar بتواند کلمهٔ رمزی را از ورودی بگیرد و آن را نمایش دهد.

۲ــ برنامه‌ای بنویسید که با هربار زدن یک کلید (حداکثر ۴ بار) یک عکس متفاوت را در کنترل Image نشان دهد و پس از زدن آخرین مرتبه، عکس ثابت بماند.

۳ــ برنامهٔ قبل را به گونه‌ای تغییر دهید که پس از رسیدن به آخرین عکس، دوباره به ابتدا بازگردد.

۴ــ برنامه‌ای بنویسید که یک تصویر را نمایش داده و سپس به وسیلهٔ متد PaintPicture آن را کوچک‌تر کرده و در یک کنترل تصویر دیگر نشان دهد (عمل Zoom Out).

۵ــ برنامه‌ای بنویسید که یک تصویر را به وسیلهٔ کنترل Image نمایش داده و قابلیت بزرگ‌نمایی از ۰/۲۵ تا ۴ برابر را برای آن ایجاد کند.

۶ــ برنامهٔ قبل را فقط با PictureBox ایجاد کنید.

۷ــ برنامه‌ای بنویسید که به وسیلهٔ کلیدهای جهت‌دار (Arrow Key) یک برچسب حاوی نام شما را حرکت دهد. میزان حرکت به وسیلهٔ یک کادر متن تعیین شود.

۸ــ برنامهٔ قبل را به گونه‌ای تغییر دهید که به وسیلهٔ کلیدهای PageUP و PageDowm بتوان میزان حرکت را در هر بار دو برابر یا نصف کرد.

۹ــ یکی از راه‌های ساده برای رمزکردن یک رشته، افزودن یا کاهش کد اسکی یک کاراکتر است. به‌عنوان مثال، اگر به کد اسکی هر کاراکتر ۴ واحد اضافه کنیم، کلمهٔ ALI به EMP عوض می‌شود.

برنامه‌ای بنویسید که به وسیلهٔ یک کادر متن، رشته‌ای را دریافت و در همان لحظهٔ ورود، آن را رمز کند. سپس با زدن دکمهٔ Decode و استفاده از فرمان Sendkeys رشتهٔ کادر متن اول را به کادر متن دیگری بفرستد تا در آن رمزگشایی شود (یعنی به شکل اصلی دیده شود).

توجه : از هیچ تابع رشته‌ای نباید استفاده کنید.

۱۰ــ برنامه‌ای بنویسید که تصویری را نمایش دهد که با کلیک روی تصویر، محل آن به‌طور تصادفی تغییر یابد (از فرم خارج نشود).

فصل دوم

ایجاد منو

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

- **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.**

۱ــ۲ــ منوهای استاندارد ویندوز

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

منو متداول‌ترین ویژگی قابل مشاهده برای برنامهٔ کاربردی است که استفاده از برنامه را ساده‌تر خواهد کرد. با وجود منو، محیط 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 عملیات روی پرونده‌ها هستند که به کاربران امکان می‌دهد پرونده‌ها را ایجاد و ذخیره کنند. برنامه‌هایی که از ویژگی‌های متداول ویندوز مثل بازکردن و ذخیرهٔ پرونده‌ها یا کپی کردن متن استفاده می‌کنند، بهتر است از استانداردهای منوهای ویندوز پیروی کنند.

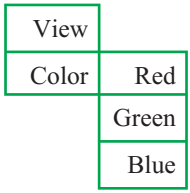
هدف مایکروسافت ایجاد ویژگی‌های استاندارد در کل رابط گرافیکی کاربر است. این بدین معنی است که اغلب برنامه‌های تحت ویندوز ظاهری شبیه هم دارند. به عنوان مثال، روش دسترسی به

گزینهٔ Print در کل برنامه‌ها مشابه هم است. این استانداردها در چندین مورد مثل سیستم‌های راهنما رعایت می‌شوند.

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

جدول ۱ــ۲ــ موارد متداول منوهای استاندارد ویندوز

ویژگی	تعریف
Caption	استفاده از یک یا دو کلمهٔ خاص
Organization	گزینه‌های منو که براساس وظیفه گروه‌بندی خواهند شد تا حداقل تعداد سطوح را برای دسترسی به هر ویژگی ارائه 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 غیرفعال‌شدن دارند، باید یک کادر علامت داشته باشند که نشان‌دهندهٔ حالت آن باشد. گزینه‌ای که یک کادر محاوره‌ای را باز می‌کند باید دارای ... در انتهای نام خود باشد.



Organization تعیین سر منو و زیر منوها براساس وظیفهٔ آن‌ها می‌باشد. در شکل روبرو view سر منو و دارای سطح صفر Color زیر منوی View و داری سطح یک و گزینه‌های Red، Green و Blue زیر منوهای Color و دارای سطح ۲ هستند.

Accesskey برای دسترسی به گزینه‌ها تعریف می‌شود و سبب می‌شود که به جای استفاده از ماوس برای انتخاب منو از صفحه کلید استفاده نماییم (مانند Alt+f برای بازکردن منوی file) ولی Shortcut Key کلید میانبر برای وظیفه‌ای است که در صورت انتخاب منو انجام می‌شود (مانند Ctrl+C برای عمل کپی در نرم‌افزار word)

۲ــ۲ــ ایجاد منو

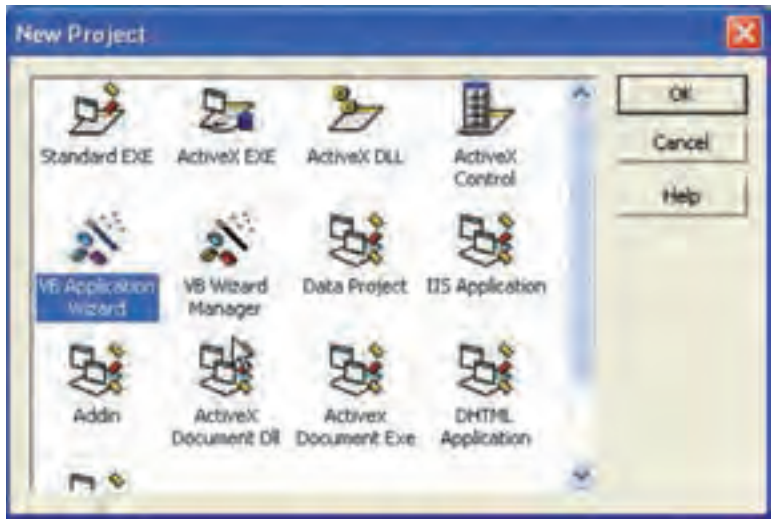
در ویژوال بیسیک برای ایجاد منو دو شیوه وجود دارد که عبارتند از :

- Application wizard
- Menu Editor

Application Wizard ابزار مفیدی برای ایجاد برنامهٔ کاربردی جدید است. این بدین معنی است که این ابزار برای ایجاد برنامهٔ کاربردی با ویژگی‌های استاندارد مورد استفاده قرار می‌گیرد. مزیت عمدهٔ Application Wizard ایجاد منوهایی است که دارای ویژگی‌های استاندارد ویندوز هستند. به سادگی، می‌توان این ویژگی‌ها را که به صورت الگو ارائه شده‌اند انتخاب کرد. 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، برنامهٔ اصلی تولید می‌شود که امکان انجام تغییرات با Menu Editor را فراهم می‌کند.

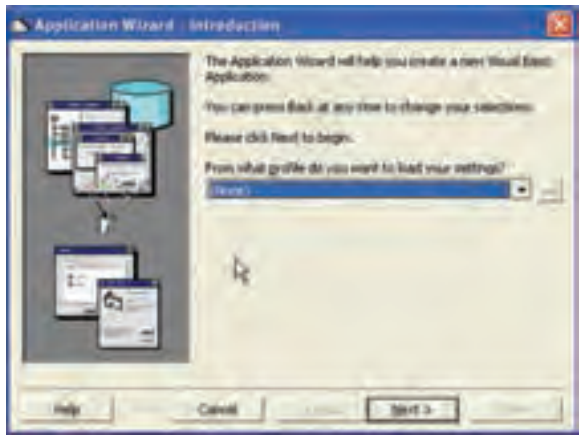
۱ــ۲ــ۲ــ ایجاد یک منوی ساده با Application Wizard

۱ــ از طریق کادر محاوره‌ای پیش‌فرض که هنگام شروع Visual Basic 6.0 باز می‌شود همچنین می‌توانید با انتخاب گزینهٔ New Project از منوی File، برنامهٔ VB Application Wizard را شروع کنید (شکل ۱ــ۲).



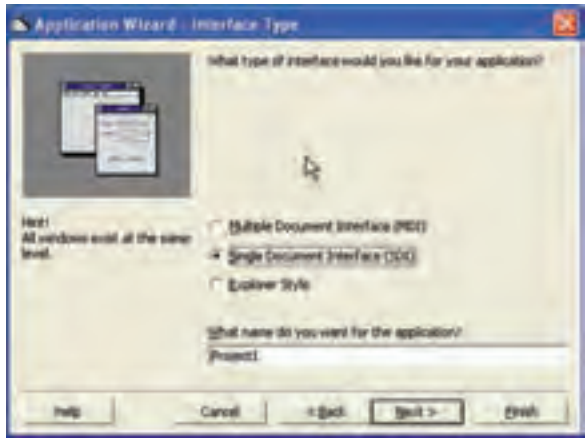
شکل ۱ــ۲ــ روی نشانه Application Wizard برای شروع ویزارد دو بار کلیک کنید.

۲ــ کادر محاوره‌ای Introduction امکان استفاده از پروفایل‌های قبلی را که ذخیره کرده‌اید فراهم می‌کند (شکل۲ــ۲). گزینه‌های پیش‌فرض را تغییر ندهید و روی Next کلیک کنید.



شکل ۲ــ۲ــ پروفایل‌ها امکان بارگذاری مجدد گزینه‌ها از پرونده‌های قبلی Application Wizard را فراهم می‌کنند.

۳ــ در کادر محاوره‌ای Interface Type، نوع صفحهٔ آغازین برنامهٔ کاربردی را انتخاب کنید (شکل۳ــ۲). برای این برنامهٔ کاربردی نمونه، Single Document Interface را انتخاب کنید. نام پروژهٔ پیش‌فرض را تغییر ندهید و Next را انتخاب کنید.



شکل ۳ــ۲ــ انتخاب نوع رابط بستگی به چگونگی استفاده از برنامهٔ کاربردی خواهد داشت. برنامه‌های کاربردی چند وظیفه‌ای اغلب از MDI [۱] استفاده می‌کنند.

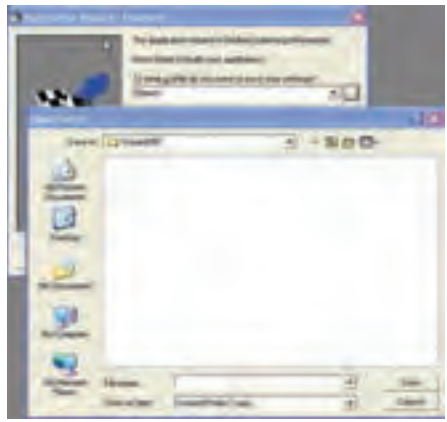
۱ــ توضیح برنامه‌های MDI در فصل سوم آمده است.

۴ــ یک منوی پیش‌فرض بر اساس استاندارد ویندوز ایجاد می‌شود و شما می‌توانید منو و زیرمنوها را تغییر دهید. بعد از انجام تغییرات، روی Next کلیک کنید (شکل ۴ــ۲).

شکل ۴ــ۲ــ نوع منو و گزینه‌ها را برای برنامهٔ کاربردی انتخاب کنید.

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

۶ــ در آخرین کادر محاوره‌ای، می‌توان پروفایل را ذخیره کرد (شکل ۵ــ۲). نامی را برای پروفایل وارد کنید که مرتبط با برنامهٔ کاربردی باشد. بعد از وارد کردن نام، روی Finish کلیک کنید تا Application Wizard کامل شود. (کلیک کردن روی Finish در کارهای محاوره‌ای قبلی، سبب می‌شود که ویزارد از ذخیرهٔ پروفایل، صرف‌نظر کند.)



شکل ۵ــ۲ــ نام پروفایل را در آخرین صفحه وارد کنید.

نکته

پسوند پروفایل RWP است.

۳ــ۲ــ کاربرد Menu Editor

Menu Editor امکان ایجاد نوار منو یا اصلاح منوهای ایجاد شده را فراهم می‌کند. این منوها به طور معمول در بالای فرم قرار می‌گیرد.

روش‌های باز کردن برنامهٔ Menu Editor عبارت‌اند از :

- فشار دادن کلیدهای Ctrl + E
- دکمهٔ Menu Editor در نوار ابزار استاندارد
- انتخاب گزینهٔ Menu Editor از منوی Tools

مثال ۱ــ۲

ایجاد یک منوی ساده

۱ــ پروژهٔ جدیدی را باز کنید.

۲ــ مشخصهٔ Caption فرم را به Simple Menu تغییر دهید.

۳ــ برنامهٔ Menu Editor را باز کنید (توجه کنید که برای باز کردن این برنامه، باید فوکوس روی فرم باشد.)

شکل ۶ــ۲ــ دکمه‌های فلش، امکان تورفتگی و سازماندهی گزینه‌های در لیست منو را فراهم می‌کنند.

۴ــ در کادر محاوره‌ای Menu Editor و در قسمت Caption، عبارت File و در قسمت Name، عبارت mnuFile را وارد کنید.

۵ــ برای ایجاد گزینهٔ این منو روی Next کلیک کنید.

۶ــ در قسمت Caption، عبارت Exit و در قسمت Name، عبارت itmExit را وارد کنید. به دلیل این‌که این منو زیرمنوی (گزینهٔ) منوی File است، روی دکمهٔ فلش راست کلیک کنید (شکل ۶ــ۲).

۷ــ روی Ok کلیک کنید تا منوی ایجاد شده را مشاهده کنید (شکل ۷ــ۲).

۸ــ منوی File را باز کرده و روی Exit کلیک کنید. پنجرهٔ کد با رویداد itmExit_click() ظاهر می‌شود.

۹ــ دستور Unload Me را به روال رویداد itmExit اضافه کنید (شکل ۸ــ۲).

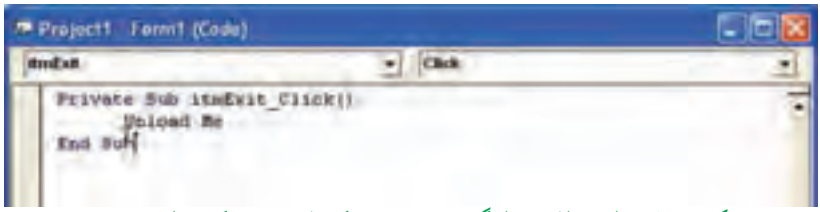
نکته

برای بارگذاری فرم در حافظه از دستور loadMe و برای پاک کردن آن از حافظه از دستور Unload Me استفاده کنید.

ـ منظور از Me فرم جاری است.

۱۰ــ کلید F5 را فشار دهید تا کد اجرا شود.

شکل ۷ــ۲ــ یک گزینه در منو شبیه هرکنترل دیگری در ویژوال بیسیک است که دارای مشخصه‌ها و یک رویداد به نام Click است.



شکل ۸ــ۲ــ برای روال رویداد گزینه‌ها مانند سایر کنترل‌های VB، کدی را بنویسید.

۱ــ۳ــ۲ــ تنظیم مشخصه‌های منو : همان‌طور که قبلاً نیز ذکر شد، منو کنترلی با مجموعه‌ای از مشخصه‌ها و یک رویداد ()Click است. جدول ۲ــ۲ متداول‌ترین مشخصه‌های مورد استفاده برای شیء Menu را نشان می‌دهد.

مانند هر کنترلی، می‌توان هنگام ایجاد منو یا گزینهٔ منو، مشخصهٔ Name را مقداردهی کرد. عدم مقداردهی Name سبب بروز خطا می‌شود. عدم مقداردهی مشخصهٔ Caption سبب نمایش یک خط خالی خواهد شد.

۲ــ۳ــ۲ــ اضافه کردن کلیدهای دسترسی به گزینه‌های منو : علاوه بر کلیک کردن روی یک گزینهٔ منو برای اجرای یک وظیفه، می‌توان با استفاده از کلیدهای دسترسی، به گزینه‌ها دسترسی پیدا کرد. کلیدهای دسترسی به کاربران امکان می‌دهند گزینه‌ها را با فشار دادن Alt و سپس حرف تعیین شده انتخاب کنند. بعد از این‌که منو باز شد، کاربران می‌توانند با فشار دادن کلید دسترسی، گزینه را انتخاب کنند.

جدول ۲ــ۲ــ مشخصه‌های متداول شیء Menu

مشخصه	مقدار / نوع	شرح
Caption	String	متنی که روی نوار منو ظاهر می‌شود.
Checked	Boolean	یک علامت ✓ قبل از رشتهٔ Caption گزینه قرار می‌دهد.
Enabled	Boolean	درصورتی که True باشد، رشتهٔ 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 می‌باشد که به‌عنوان کلید میانبر برای وظیفه‌ای که این گزینه انجام می‌دهد، به‌کار می‌رود. این مقدار را فقط در زمان طراحی و از طریق فرمتی که در کادر لیست Shortcut در Menu Editor ظاهر می‌شود، می‌توانید انتخاب کنید.
WindowList	Boolean	یک منوی سطح بالا در فرم MDI ایجاد می‌کند تا لیستی از پنجره‌های باز را نمایش دهد.[۱]

۱ــ فرم MDI می‌تواند همزمان چند پنجرهٔ باز داشته باشد (مانند نرم‌افزار word) لیست پنجره‌های باز در منویی که خاصیت window List آن true است دیده می‌شود.

برای تعریف یک کلید دسترسی، در کادر متن Caption، قبل از نویسهٔ موردنظر، از نویسهٔ & استفاده کنید (شکل ۹ــ۲).

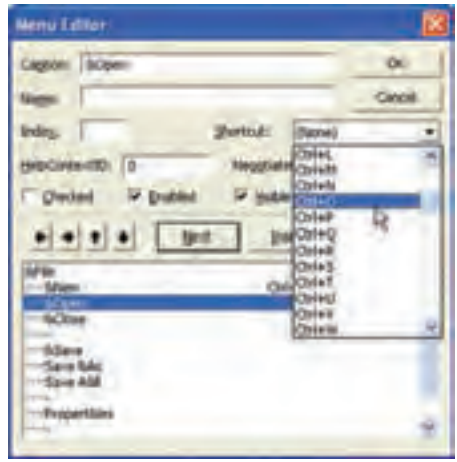
شکل ۹ــ۲ــ کلیدهای میانبر و دسترسی را قبل از کامپایل کردن برنامه تعیین کنید.

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، گزینهٔ بعدی با همان کلید، انتخاب می‌شود.

۳ــ۳ــ۲ــ 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 ۱۰ــ۲).



شکل ۱۰ـ۲

نکته

هر برنامهٔ کاربردی فقط می‌تواند یک نمونه از کلید میانبر را داشته باشد. به‌عنوان مثال، اگر از Ctrl+N برای New استفاده کرده باشید، Ctrl+N را برای گزینهٔ Open به‌کار نبرید. هنگام تعیین کلید میانبر تکراری، VB خطایی را نشان می‌دهد (شکل۱۱ـ۲).

شکل ۱۱ـ۲

نکته

برای گروه‌بندی گزینه‌های یک منو نیاز به خط جداکننده داریم. برای قراردادن این خطاها در بین گزینه‌های منو، در قسمت Caption از کادر محاوره‌ای Menu Editor، خط تیره قرار دهید و نام گزینه را با پیشوندی مثل Sep شروع کنید. توجه داشته باشید که Name خط جداکننده را خالی رها نکنید.

۴ــ۳ــ۲ــ ایجاد منوهای بازشو : دو نوع منو وجود دارد : نوار منویی و منوی بازشو.

نوار منویی به طور ثابت در بالای فرم قرار می‌گیرد اما **منوی بازشو** منویی است که در هر جایی از فرم ظاهر می‌شود.

منوی بازشو در موارد مورد نیاز مانند کلیک راست در محل اشاره‌گر ماوس ظاهر می‌شود. برای ایجاد چنین منوهایی نیز از Menu Editor استفاده می‌شود. در حقیقت، در مواقع مورد نیاز می‌توانیم قسمتی از منوی تعریف شده به‌وسیلهٔ Menu Editor را در محل اشاره‌گر یا در محل مورد نظر فعال کنیم. در VB با استفاده از دستور PopupMenu می‌توان قسمتی از منوها را به همراه زیر منوهای بعد از آن در ناحیه‌ای از فرم ظاهر کرده و از آن استفاده نمود.

شکل کلی این دستور به صورت زیر است :

```
PopupMenu MenuName [Xpos],[Ypos]
```

در این ساختار MenuName نام منویی است که می‌خواهیم در فرم نمایش داده شود. هم‌چنین دو پارامتر اختیاری Ypos و Xpos، مختصات محل ظاهر شدن منو را تعیین می‌کنند. اگر از این دو مقدار استفاده نشود VB به‌طور خودکار از مختصات محل فعلی ماوس 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 نمونهٔ 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 ارائه می‌کند. طریقهٔ ایجاد چنین منوهایی را در مثال فوق آموختید، ولی ممکن است سؤال کنید که : وقتی روی یک فرم کلیک می‌شود، برنامه چگونه می‌تواند تشخیص دهد که کدام یک از کلیدهای ماوس فشار داده شده است؟ نوع کلید ماوس را با بررسی آرگومان ورودی Button در رویدادهای MouseDown و MouseUp می‌توان تعیین کرد.

در صورتی که بخواهید می‌توانید یک فرمان را هم در نوار منویی و هم منوی بازشو قرار دهید (شکل۱۲ــ۲). 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
```

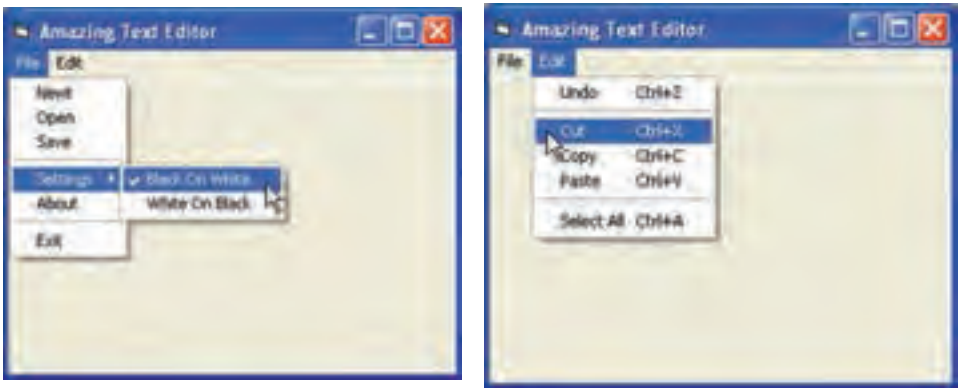
برای این‌که این کد کار کند، نیاز به ایجاد فرمی دارید که شامل یک کنترل Menu به نام mnuFile باشد که حداقل دارای یک گزینه است.

شکل ۱۲ــ۲ــ هنگامی که از متد PopupMenu استفاده می‌کنید، فقط گزینه‌های منو ظاهر خواهند شد.

پژوهش: اگر سر منویی غیرفعال باشد، زیر منوهای آن به چه صورتی مشاهده می‌شوند؟

مثال ۲ــ۲

ایجاد منوهای پیچیده : منویی ایجاد کنید که گزینه‌های زیر در آن وجود داشته باشد :



شکل ۱۳ــ۲

قبل از شروع طراحی، باید ویژگی‌های برنامه را مرور کنید تا سیستم منو را به طور مناسب طراحی و گروه‌بندی کنید. اغلب، نوارهای منو با منوی File و Edit شروع می‌شوند.

File Menu	Edit Menu
New	Undo Ctrl + Z
Open	Cut Ctrl +X
Save	Copy Ctrl +C
Settings	Paste Ctrl +V
About	Select All Ctrl +A
Exit Ctrl + E	

به گروه‌بندی سیستم منو توجه کنید. می‌توان این منو را در Menu Editor پیاده سازی کرد.

جدول ۳ــ۲ سلسله مراتب منو و مشخصه‌های Name ، Caption و کلیدهای میانبر برای هر شیء منو را نشان می‌دهد.

جدول ۳ــ۲ــ شیء‌های منو برای مثال ۲ــ۲

Name	Caption	Level	Shortcut
mnuFile	& File	0	None
itmNew	& New	1	None
itmOpen	& Open	1	None
itmSave	& Save	1	None
sepOne	–	1	None
itmSettings	Se&ttings	1	None
itmBlackOn White	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
sep Three	–	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

۴ــ۲ــ شیء Clipboard

یکی از ویژگی‌های مهم سیستم عامل ویندوز، قابلیت انتقال داده‌ها از یک برنامهٔ کاربردی به برنامهٔ کاربردی دیگر از طریق حافظهٔ Clipboard است. تمام برنامه‌های کاربردی امکان دسترسی به این حافظه را دارند. می‌توان هر شیء موجود در ویندوز (از متن ساده تا یک تصویر) را در این حافظه ذخیره کرد.

در VB می‌توان از طریق شیء Clipboard به این حافظه دسترسی داشت. این شیء به‌طور پیش‌فرض در پروژهٔ Standard EXE قرار دارد و می‌توان از آن استفاده کرد. شیء Clipboard هیچ مشخصه‌ای ندارد ولی دارای چندین متد است. برخی از متدهای Clipboard در جدول ۴ــ۲ ذکر شده است.

جدول ۴ــ۲ــ متدهای شیء Clipboard

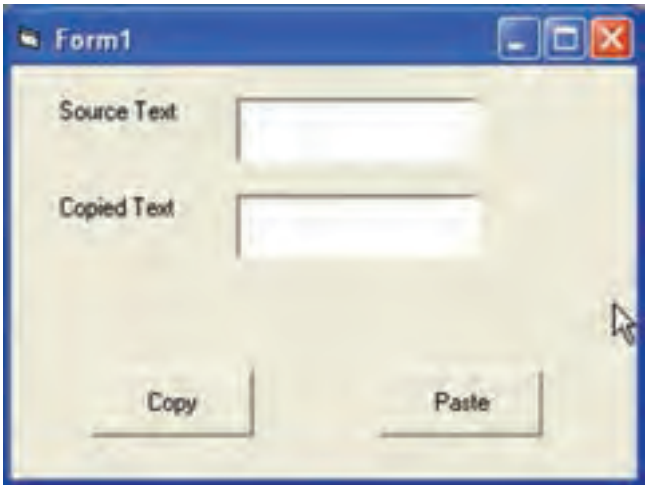
متد	شرح
Clear	تمام داده‌ها را از حافظهٔ Clipboard پاک می‌کند.
GetText	متن ASCII را از حافظهٔ Clipboard بازیابی می‌کند.
SetText	متن ASCII را به حافظهٔ Clipboard ارسال می‌کند.

مثال ۳ــ۲

فرمی ایجاد کنید که شامل دو کادر متن به نام‌های txtMain و txtClip و دو دکمهٔ فرمان به نام‌های cmdCopy و cmdPaste باشد. می‌خواهیم هر چیزی که در txtMain نوشته می‌شود، با انتخاب دکمهٔ Copy به حافظهٔ Clipboard کپی شده و با انتخاب Paste از حافظهٔ Clipboard به کادر متن txtClip کپی شود (شکل ۱۴ــ۲).

کد دکمه‌های فرمان این مثال به صورت زیر خواهد بود :

```
Private Sub cmdCopy_ Click()
    Clipboard.SetText(txtMain.Text)
End Sub
Private Sub cmdPast_Click()
    txtCLip.Text = Clipboard.GetText
End Sub
```



شکل ۱۴ــ۲

در این مثال، کل متن موجود در txtMain با انتخاب دکمهٔ Copy به حافظهٔ Clipboard وارد می‌شود. در صورتی که می‌خواهید فقط قسمتی از متن که انتخاب کرده‌اید به حافظهٔ Clipboard وارد شود، در روال مربوط به دکمهٔ Copy به جای Text از مشخصهٔ SelText استفاده کنید :

```
Clipboard.SetText(txtMain.SelText)
```

نکته

برای انتخاب قسمتی از متن در کادر متن در زمان اجرا، روی نویسهٔ اول کلیک کنید و تا نویسهٔ مورد نظر درگ کنید. انتخاب متن به‌صورت کدنویسی نیز ممکن است. برای انجام این کار، از دو مشخصهٔ SelStart و SelLenght استفاده کنید. SelStart محل شروع انتخاب و SelLenght طول نویسه‌های انتخاب شده را مشخص می‌کنند. به‌عنوان مثال، می‌خواهیم کلمه‌ای را که در کادر متن قرار دارد از نویسهٔ دوم تا پنجم انتخاب کرده و به حافظهٔ Clipboard منتقل کنیم.

```
Text1. Selstart = 1
Text1. SelLenght = 4
Clipboard. SetText (Text 1. SelText)
```

خودآزمایی

۱ــ در چه مواقعی از منوها استفاده می‌شود؟

۲ــ در کادر Menu Editor، گزینهٔ Name به چه منظوری مورد استفاده قرار می‌گیرد؟

۳ــ انواع منوها را نام ببرید.

۴ــ وقتی کاربر از کلید میانبر استفاده می‌کند، کدام رویداد تحریک خواهد شد؟

۵ــ 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.

۶ــ در تمرین ۵، TextBox به فرم اضافه کرده و منویی برای تغییر رنگ متن آن به رنگ‌های زرد و آبی فراهم کنید.

۷ــ به مثال ۳ــ۲ دکمهٔ cut را اضافه کنید و به جای انتقال محتوای txtMain تنها قسمت انتخاب‌شده را cut و past کنید.

فصل سوم

خطایابی و اشکال‌زدایی برنامه

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

- **اصول کار با دستور On Error Goto را بیان کرده و در برنامه‌ها آن را به‌کار ببرد؛**
- **نحوهٔ کار باشیء ERR را شرح داده و در برنامه‌های خود از آن استفاده کند؛**
- **با دستور Resume کار کند و اجرای برنامه‌های قطع شده را از سر بگیرد؛**
- **برنامه‌های خود را اشکال‌زدایی کند.**

۱ــ۳ــ انواع خطاها در برنامه‌نویسی

در برنامه‌نویسی سه نوع خطا وجود دارد :

۱ــ خطای نحوی (syntax error) : شامل خطا در املاء یا محل قرارگیری کلیدواژه‌ها می‌باشد مانند استفاده از دستورIf بدون کلیدواژهٔ then در زبان بیسیک

```
If   a>b
Print a
```

دستور بالا دارای خطای نحوی "Expected: then or Go To" (فاقد then یا GoTo) است.

این خطاها در زمان کامپایل برنامه مشخص می‌شود.

۲ــ خطای زمان اجرا (Runtime error) : خطاهایی که هنگام اجرای دستور رخ می‌دهد و تا قبل از اجرای آن مشخص نمی‌شود.

مثال ۱ــ۳

خطای تقسیم بر صفر

```
i = 0   :  a=5
b= a/i
```

این نوع خطاها در زمان کامپایل قابل تشخیص نیست.

۳ــ خطای منطقی (logical error) : خطاهای منطقی در زمان کامپایل و اجرا مشکلی ایجاد نمی‌کند بلکه از نظر منطقی خروجی برنامه، خروجی مورد نظر نیست و برنامه‌نویس در پیاده‌سازی منطق برنامه و حل مسأله اشتباه کرده است. بنابراین نتیجه مطلوب حاصل نمی‌شود.

```
for   i=2      to 10
        print  i ;
Next
```

هدف تکه کد بالا تولید اعداد زوج کوچکتر از ۱۲ بوده است ولی خروجی آن عبارتست از

2 3 4 5 6 7 8 9 10

زیرا برنامه‌نویس step 2 را فراموش کرده است و باید کد به‌صورت زیر اصلاح شود.

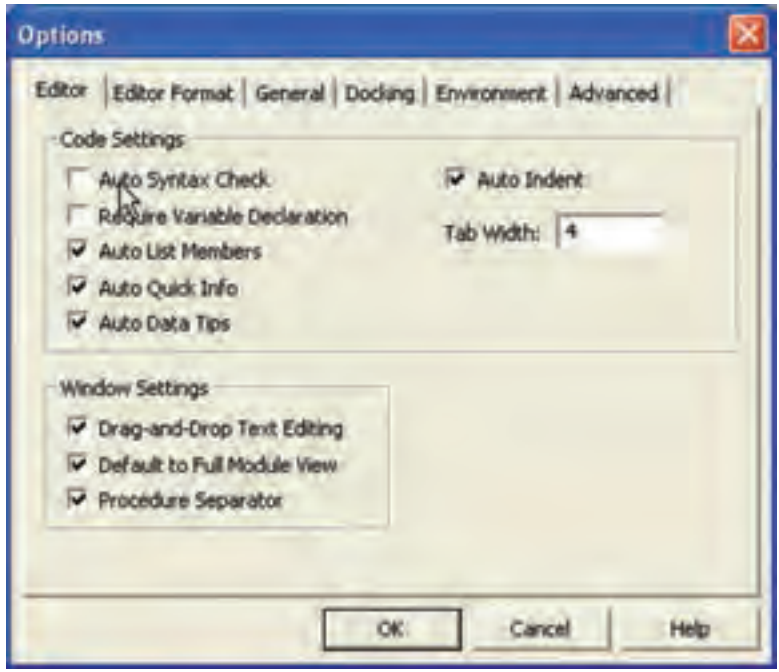
```
for    i=2    to  10   step 2
        print  i ;
Next
```

۲ــ۳ــ رفع اشکال متغیرهای اعلان نشده با Option Explicit

همزمان با کدنویسی، IDE ویژوال بیسیک خطاهای نحوی را که به‌وجود می‌آیند اعلام می‌کند (شکل ۱ــ۳). خطاهای نحوی شامل املاء یا محل قرارگیری کلیدواژه‌ها هستند. به این نوع خطاها می‌توان به سادگی پی برد و آن‌ها را رفع کرد.

شکل ۱ــ۳ــ IDE ویژوال بیسیک دستورات IF بدون کلیدواژهٔ 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 نحوی را اعلام می‌کند. در صورتی که می‌خواهید VB کار را قطع نکند و برای هر خطایی، پیغامی نمایش ندهد، از منوی Tools گزینهٔ Options را انتخاب کرده و کادر علامت Auto Syntax Check را پاک کنید (شکل ۲ــ۳).



شکل ۲ــ۳ــ غیرفعال کردن Auto Syntax Check سبب می‌شود که پیغام‌های خطای نحوی را هنگام تایپ مشاهده نکنید.

پاک کردن این کادر علامت سبب می‌شود که کامپایلر از پیدا کردن خطاها در هنگام تایپ، صرف‌نظر کند.

هنگامی که کد را درون IDE اجرا می‌کنید، ویژوال بیسیک خطاهایی مثل نوع اشتباه و بلاک‌های کد کامل نشده را گزارش می‌کند (شکل ۳ــ۳). فقط در صورتی که Option Explicit تنظیم شده باشد، ویژوال بیسیک کد متغیرهای اعلان نشده را اجرا می‌کند. هنگامی که کلید واژهٔ Option Explicit را در بخش General یک فرم یا مدول وارد می‌کنید، همهٔ متغیرهای کد باید با استفاده از کلیدواژه‌های Public، Private، Dim یا Static به‌طور صریح اعلان شوند.

شکل ۳ــ۳ــ IDE هنگام کامپایل کد، یک بلاک حلقهٔ کامل نشده را گزارش می‌کند.

اشتباهات تایپی ساده منجر به بروز خطاهای بزرگی در کد می‌شوند. به‌عنوان مثال، در کد زیر متغیری با نام intMyNum اعلان شده است ولی در خط ۴ از intMyNim استفاده شده است. به دلیل این که Option Explicit استفاده نشده است، VB به‌طور خودکار متغیر جدیدی به نام intMyNim ایجاد کرده و آن را با صفر مقداردهی می‌کند. (خطای منطقی) شکل ۴ــ۳ نتیجه را نشان می‌دهد.

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

شکل ۴ــ۳ــ اگر از Option Explicit استفاده کرده باشید، پیام خطایی را دریافت خواهید کرد که مشخص می‌کند متغیر intMyNim تعریف نشده است.

۳ــ۳ــ 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 نقطهٔ قطع، محلی در کد است که می‌توان کد را در طول اجرا متوقف کرد. نقطهٔ قطع را می‌توان به چهار روش تعیین کرد :

- کلیک روی خط کد موردنظر و فشار دادن کلید F9
- کلیک روی آیکن Toggle Breakpoint در نوار ابزار Debug
- انتخاب Toggle Break Point از منوی Debug
- کلیک در حاشیهٔ پنجرهٔ Code

نکته

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

هنگامی که یک نقطهٔ قطع را تعیین می‌کنید، توجه کنید که به رنگ قرمز تبدیل می‌شود. هنگامی که کد قطع می‌شود، خط موردنظر کد به رنگ زرد تبدیل می‌شود. همچنین یک فلش در حاشیهٔ چپ پنجرهٔ Code به خط موردنظر اشاره می‌کند (شکل ۵ــ۳).

شکل ۵ ــ۳ــ برای رجوع به نقطهٔ قطع بعدی، کلید F5 را فشار دهید.

همهٔ اشکالات براثر 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، نقطهٔ قطع را تعیین کرده و ناحیهٔ بروز خطا را با استفاده از watches جست‌وجو کنید.

۴ــ۳ــ نمایش مقادیر متغیرها با Watches

با نگاه دوباره به کد و شکل ۳ــ۳ مشاهده می‌کنیم که دارای اشکال است و دلیل آن نیز روشن است. فرض کنید که دلیل بروز این خطا را نمی‌دانید لذا فرصت مناسبی است که از نقاط قطع و watches استفاده کنید.

۱ــ نقطهٔ قطع را برای خطی از کد که کادر پیام را نشان می‌دهد تعیین کنید.

۲ــ برنامه را اجرا کنید.

۳ــ اشاره‌گر ماوس را روی متغیری که می‌خواهید مقدار آن را مشاهده کنید، درگ نمایید و چند لحظه نگه دارید. پنجرهٔ کوچکی با مقدار متغیر ظاهر می‌شود.

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

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

۲ــ در هر نقطهٔ قطع، متغیر مورد نظر را های‌لایت (مشخص) کنید.

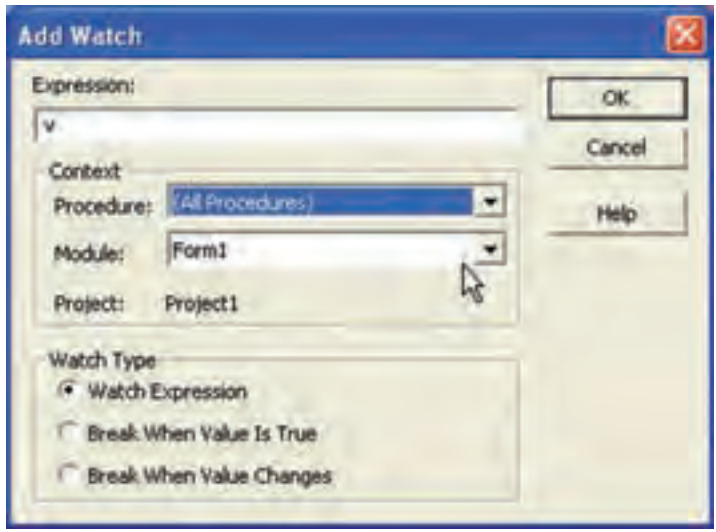
۳ــ کلیک راست کرده و Add watch را از منوی میانبر انتخاب کنید.

۴ــ تنظیمات مناسب را در کادر محاوره‌ای Add watch انجام دهید (شکل۶ــ۳) و سپس روی Ok کلیک کنید.

در پنجرهٔ Watches، قسمت Expression نام متغیر یا مشخصه‌ای نوشته می‌شود که می‌خواهیم مقدار آن را بررسی کنیم. در قسمت Context نام رویداد یا مدولی را که می‌خواهیم مقدار عبارت را در آن بررسی کنیم، می‌نویسیم. در سومین بخش یعنی Watch Type می‌توانید یکی از سه گزینه را انتخاب کنید. با انتخاب اولین گزینه، هنگام رسیدن به نقطهٔ توقف، مقدار Expression را می‌توان در پنجرهٔ Watch مشاهده کرد. انتخاب گزینهٔ دوم سبب می‌شود که در صورت True یا غیرصفر بودن مقدار متغیر کادر Expression، یک نقطهٔ توقف در دستور بعد از آن ایجاد شود. چنانچه گزینهٔ سوم

را انتخاب کنید، اگر مقدار عبارت کادر Expression تغییر کند، یک نقطهٔ توقف در دستور بعدی ایجاد می‌شود.

۵ ــ برای نمایش پنجرهٔ watches (شکل ۷ــ۳)، ازمنوی View گزینهٔ watch window را انتخاب کنید. هنگام اضافه کردن یک watches نیز این پنجره ظاهر می‌شود.



شکل ۶ــ۳ــ کادر محاوره‌ای Add Watch یک ابزار قوی و دارای قابلیت انعطاف است.

شکل ۷ــ۳ــ در حالت Break، مطمئن باشید در محلی از کد که در میدان دید متغیرهای پنجرهٔ watches است، قرار دارید.

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

۱ــ۴ــ۳ــ بررسی خط به خط کد با Step Into و Step Over : می‌توان کد یک برنامه را به صورت خط به خط اجرا و بررسی نمود و اگر خطایی در هر خط وجود دارد آن را رفع کرد. اجرای گام به گام به دو روش انجام می‌شود : Step Into و Step Over.

Step Into را می‌توان با روش‌های زیر اجرا کرد :

- انتخاب گزینهٔ Step Into از منوی Debug
- فشاردادن کلید F8
- کلیک روی نشانهٔ Step Into از نوار ابزار Debug

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.

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 ()From_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 نقطهٔ قطع در خط موردنظر کد قرار داده و سپس کد را به صورت عادی اجرا کنید.

هنگامی که اجرا به نقطهٔ قطع برسد، اجرای گام به گام شروع خواهد شد. برای خروج سریع از روالی که به صورت گام به گام اجرا می‌شود، باید از Step Out استفاده کرد.

Step Out را می‌توان با روش‌های زیر اجرا کرد :

- انتخاب گزینهٔ Step out از منوی Debug
- فشاردادن کلید Ctrl + Shift + F8
- کلیک روی نشانه Step Out از نوار ابزار Debug

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.

Step Over را می‌توان با روش‌های زیر اجرا کرد :

- انتخاب گزینهٔ Step Over از منوی Debug
- فشار دادن کلید Shift + F8
- کلیک روی آیکن Step Over از نوار ابزار Debug

۲ــ۴ــ۳ــ متوقف‌کردن خطوط انتخاب شده : هر بار که یک نقطهٔ قطع را تعیین کنید، تا زمانی که به کد برنگشته و آن نقطهٔ قطع یا همهٔ نقاط قطع را پاک نکرده باشید، در کد باقی می‌ماند. تعداد زیاد نقاط قطع، سبب کند شدن سرعت اشکال‌زدایی می‌شود. این کار را می‌توان با استفاده از Run to Cursor ساده‌تر انجام داد تا کد را در نقاط تعیین شده قطع کند.

Run to Cursor را می‌توان با روش‌های زیر اجرا کرد :

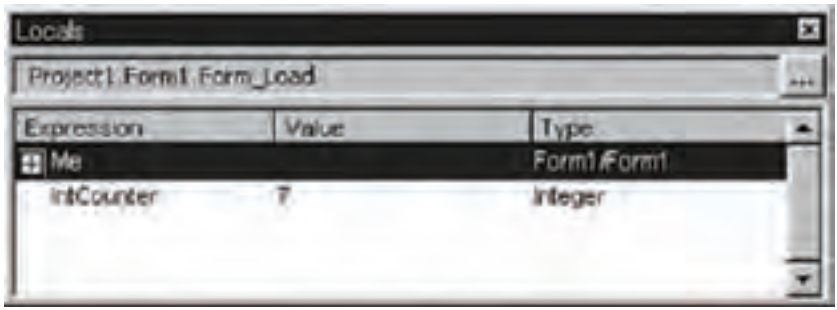
- انتخاب گزینهٔ Run to Cursor از منوی Debug
- فشاردادن کلید Ctrl +F8

روی خطی از کد که می‌خواهید متوقف شود، کلیک کرده و سپس با یکی از روش‌های فوق، Run to Cursor را اجرا کنید. کلید F5 را فشار دهید تا کد اجرا شود. IDE ویژوال بیسیک، اجرای کد را در خطوطی که کلیک کرده‌اید، قطع می‌کند.

۵ــ۳ــ استفاده از ابزارهای اشکال‌زدایی پیشرفته

علاوه بر مشاهدهٔ مقادیر متغیرها و تکنیک‌های اجرای گام به گام، می‌توان از ابزارهای نشان داده شده در شکل‌های ۸ــ۳ تا ۱۱ــ۳ استفاده کرد. ویژوال بیسیک این ابزارها را برای اشکال‌زدایی پیشرفته ارایه می‌کند.

پنجرهٔ Locals (شکل ۸ــ۳) روش ساده‌ای برای مشاهدهٔ تمام متغیرهای موجود در میدان دید جاری است. متغیرها به همراه مقادیرشان در این پنجره فهرست می‌شوند. شیءها دارای یک علامت جمع هستند که می‌توانید روی آن کلیک کنید تا مشخصه‌های شیء را مشاهده کنید. اگر مشخصه، خود شیء دیگری باشد، علامت جمع دیگری را مشاهده خواهید کرد. در این پنجره می‌توان تمام متغیرها را به طور هم زمان و بدون کلیک‌کردن روی هر کدام، مشاهده کرد. برای دسترسی به این پنجره، از منوی View گزینهٔ Locals Window را انتخاب کنید.



شکل ۸ــ۳ــ پنجرهٔ Locals

از پنجرهٔ Immediate : (شکل ۹ــ۳) می‌توان برای آزمایش خطوطی از کد بدون اجرای برنامه استفاده کرد. برای آزمایش این پنجره، کد زیر را در آن وارد کنید :

```
Print 2*3
```

دستور Print مقدار ۶ را ارایه می‌دهد.

شکل ۹ــ۳ــ پنجرهٔ Immediate امکان تایپ کد و اجرای آن را با فشاردادن کلید Enter فراهم می‌کند.

پنجرهٔ Immediate برای دستورهای یک خطی، مناسب است. در این 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 intCounter در آن تعریف شده است، می‌توان خط

زیر را در پنجرهٔ Immediate تایپ و مقدار آن را مشاهده کرد :

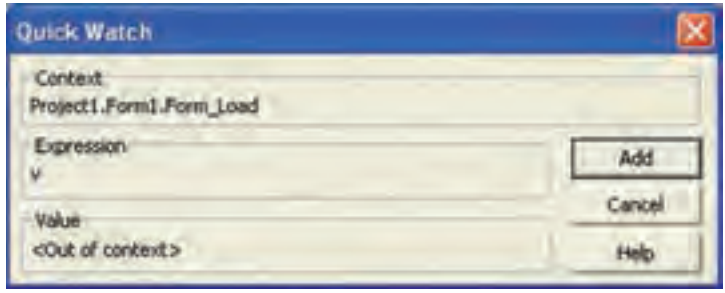
```
Print intCounter
```

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

کادر محاوره‌ای Call Stack در صورتی که از چندین روال و رویداد استفاده می‌کنید، مفید است. این کادر محاوره‌ای (شکل ۱۰ــ۳)، 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 دکمهٔ  از نوار ابزار Debug یا فشاردادن کلیدهای Ctrl+L باز می‌شود. لازم به ذکر است که این عمل فقط در حالت قطع ممکن است.

شکل ۱۰ــ۳ــ کادر محاوره‌ای Call Stack

برای نشان‌دادن **کـادر محاوره‌ای Quick Watch** (شکل ۱۱ــ۳) یک قطع را در کـد قرار دهید، روی متغیری کلیک کنید یا عبارتی را مشخص (های‌لایت) کرده و از منوی Debug گزینهٔ Quick Watch را انتخاب کنید یا از نوار ابزار Debug روی آیکن  کلیک کنید یا کلیدهای Shift+F9 را فشار دهید.



شکل ۱۱ــ۳ــ کادر محاوره‌ای Quick Watch

۶ــ۳ــ کاربرد Find and Replace

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

۷ــ۳ــ ایجاد یک مدیر خطا

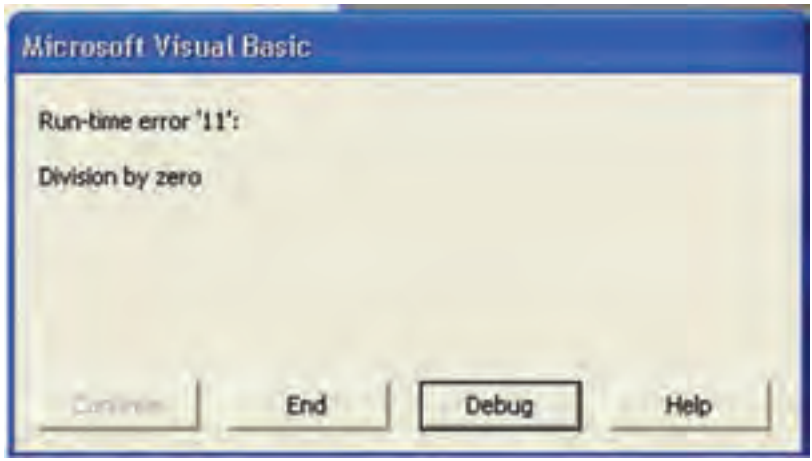
اگر نرم‌افزاری داشته باشید که در طی اجرا خراب می‌شود، احتمالاً برنامه نویس آن بخشی از برنامه را فراموش کرده است : مدیریت خطا. هر برنامه‌ای نیاز دارد که به خطاهایی که در درون خودش رخ می‌دهند، پاسخ دهد. در این قسمت، مشاهده می‌کنید که چگونه از قابلیت‌های مدیریت خطا در ویژوال بیسیک استفاده کنید.

برای درک موضوع مثال زیر را انجام می‌دهیم :

پروژهٔ جدیدی ایجاد نموده و فرمی به صورت زیر طراحی کنید :

```
Private sub Start_Click()
        Print 200/i
endsub
```

حال این پروژه را اجرا کنید. به دلیل عدم تعریف متغیر i، این متغیر از نوع variant و مقدار آن صفر خواهد بود. بنابراین خطای تقسیم بر صفر رخ خواهد داد (شکل ۱۲ــ۳).



شکل ۱۲ــ۳

با انتخاب end در کادر محاوره‌ای شکل فوق، اجرای برنامه قطع می‌شود تا خطا رفع شود و با انتخاب Debug اجرای برنامه موقتاً قطع می‌شود و به خط شامل خطا رفته و آن را با رنگ زرد مشخص می‌کند تا خطا را رفع کنید. برنامهٔ فوق را می‌توان به صورت زیر تغییر داد :

```
Dim h As Integer
Private Sub Start_Click()
    On Error Goto TestError
    Print 200/i
    Print "No Problem"
    Exit Sub
    Test Error:
        h = MsgBox ("Error" & Str(Err. Number) & "=" & _
        Err.Description, VbAbortRetryIgnore)
        If h = VbIgnore Then Resume Next
        If h = VbAbort Then End
        If h = VbRetry Then
            i = 10
            Resume
        End If
End Sub
```

به دلیل اینکه متغیر i مقداردهی نشده، لذا دارای مقدار صفر است و با اجرای خط ۴ این برنامه خطای تقسیم بر صفر رخ می‌دهد و کادر پیامی به شکل زیر ظاهر می‌شود :

شکل ۱۳ــ۳

با انتخاب Abort اجرای برنامه خاتمه می‌یابد. با انتخاب Retry مقدار i غیر صفر می‌شود و دوباره خط ۴ اجرا می‌شود که سبب رخ دادن خطا شده است. با انتخاب Ignore از خطای فعلی چشم‌پوشی شده و کنترل برنامه به خط پنجم منتقل می‌شود. به این ترتیب در زمان اجرا، خطا رفع می‌شود.

در مثال فوق با مفاهیم جدیدی روبه‌رو شدیم :

۱ــ شیء Err : از این شیء برای مدیریت خطاهای زمان اجرا استفاده می‌شود و شامل مشخصه‌هایی است که مهم‌ترین آن‌ها مشخصهٔ Number و Description است. مشخصهٔ Number شمارهٔ خطا و مشخصهٔ Description شرح خطای رخ داده را نگهداری می‌کنند. مشخصهٔ Number عددی Long و مشخصهٔ Description رشته‌ای است.

۲ــ دستور ON error : برای رفع خطاهای زمان اجرا از این دستور استفاده می‌شود. شکل کلی آن به صورت زیر است :

دستور On Error

هرگاه بخواهیم در صورت بروز خطا به محل خاصی از کد رجوع شود، می‌نویسیم :

برچسب On Error Goto

که این برچسب می‌تواند یک رشته یا عدد باشد، البته در صورت عدد بودن مقادیر منفی را نمی‌پذیرد.

اگر بخواهیم در صورت بروز خطا از آن صرف‌نظر کرده و به دستور بعدی رجوع شود، می‌نویسیم : Resume Next و هرگاه بخواهیم در صورت بروز خطا همان خط خطا را دوباره بخواند، می‌نویسیم : Resume

خودآزمایی

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

۲ــ توضیح دهید که اگر روالی یا تابعی فاقد قسمت رسیدگی به خطا باشد و در آن روال یا تابع خطایی به‌وجود آید، چه اتفاقی رخ می‌دهد.

۳ــ چرا رسیدگی به خطاها قبل از عبارت‌های Exit Sub یا Exit Function قرار می‌گیرند؟

فصل چهارم

آرایه‌ها

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

- **مفهوم آرایه‌های یک بعدی و چند بعدی را شرح داده و از آن‌ها در برنامه‌های خود استفاده کند؛**
- **روش مرتب‌سازی حبابی را شرح داده و عناصر آرایه را به‌روش حبابی مرتب کند؛**
- **روش‌های جست‌وجوی خطی و دودویی را توضیح داده و از آن برای جست‌وجوی 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 کند و به‌کار ببرد.**

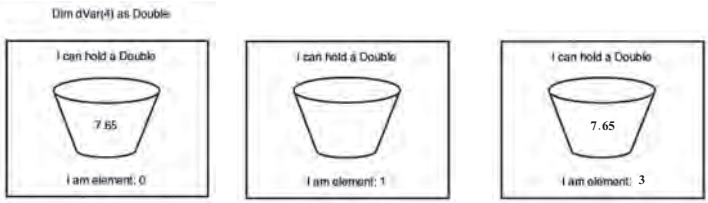
۱ــ۴ــ آرایه چیست؟

همان‌طورکه می‌دانید متغیر ظرفی است که یک مقدار را در خود نگه می‌دارد و با ورود مقدار جدید، مقدار قبلی از بین می‌رود. بعضی مواقع در برنامه‌ها نیاز به نگه‌داری چندین مقدار هم‌نوع داریم. آیا باید چندین متغیر تعریف کنیم؟ واضح است که پاسخ منفی است. راه‌حل این مشکل، استفاده از آرایه است. آرایه تعدادی خانه هم‌جوار و هم‌نام در حافظه است که برای نگهداری مقدارهای هم‌نوع استفاده می‌شود.

آرایه متغیری است با چندین خانه که می‌توان در هر خانهٔ آن، یک مقدارنگه‌داری کرد. به هر خانهٔ

آرایه یک **عنصر** گفته می‌شود. عنصر دارای شمارهٔ خاصی است که محل (موقعیت) عنصر در داخل آرایه را برمی‌گرداند. اولین عنصر آرایه معمولاً در موقعیت صفر قرار دارد. آرایه‌ها می‌توانند اندازه‌های مختلفی داشته باشند (شکل ۲ــ۴). یک آرایه ممکن است دارای سه عنصر بوده و دیگری ۳۰ عنصر داشته باشد. حتی ممکن است یک آرایه هیچ عنصری نداشته باشد (امکان اضافه‌کردن عنصر به آن وجود دارد).

شکل ۱ــ۴ــ متغیر، محلی برای نوع دادهٔ خاصی است.



شکل ۲ــ۴ــ آرایه مجموعه ای از متغیرهاست.

۲ــ۴ــ اعلان آرایه

می‌توان یک آرایه را به دو روش اعلان کرد :

- مانند یک متغیر
- با استفاده از کلید واژهٔ To

۱ــ۲ــ۴ــ 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 حوزهٔ عمل آن هستند. دربارهٔ حوزهٔ 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 کدنویسی، اولین عنصر را در مکان یک قرار داد.

- Subscript مکان آخرین عنصر آرایه را تعیین می‌کند. 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 کلید واژه‌ای است که اعلان نوع را مشخص می‌کند.
- DataType نوع دادهٔ معتبر در ویژوال بیسیک است مثل Integer یا Double و غیره.

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

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

برای تعیین یک مقدار برای هر عنصر آرایهٔ 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 این صورت عمل می‌کنیم :

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

به‌عنوان مثال، برای اعلان آرایه‌ای از ۹ رشته (نام گل‌های مختلف)، از کد زیر استفاده خواهیم کرد :

```
Public strMyArray(8) As String

strMyArray(0) = "Maryam"
```

```
strMyArray(1) = "Roze"
strMyArray(2) = "Niloofar"
strMyArray(3) = "Yakh"
strMyArray(4) = "Yas"
strMyArray(5) = "Banafsheh"
strMyArray(6) = "Zanbagh"
strMyArray(7) = "Magnolia"
strMyArray(8) = "Aftabgardan"
```

۲ــ۲ــ۴ــ اعلان آرایه با کلید واژهٔ To : همچنین می‌توان آرایه را با استفاده از کلید واژهٔ To در داخل اندیس‌های آرایه اعلان کرد. به‌عنوان 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 (حتی مقدار منفی) است.

```
Dim  A (−5,2) As string
```

۳ــ۴ــ تغییر تعداد عناصر آرایه

اگرچه معمولاً تعداد عناصر آرایه هنگام اعلان آن تعیین می‌شود ولی مواردی وجود دارد که هنگام اعلان آرایه تعداد عناصر آن را نمی‌دانیم و یا اینکه می‌خواهیم از این آرایه با اندازه‌های متفاوت در برنامه استفاده کنیم که در این موارد امکان تغییر اندازهٔ آرایه وجود دارد. هنگامی که تعداد عناصر آرایه را تغییر دهید، بُعد آن آرایه تغییر می‌کند. برای انجام این کار، از کلید واژهٔ ReDim به شکل زیر استفاده کنید :

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

- ReDim کلید واژهٔ ویژوال بیسیک است که مشخص می‌کند آرایه تغییر بُعد می‌یابد.
- Preserve یک کلید واژهٔ اختیاری است که امکان نگه‌داری مقادیر تمام عناصر از قبل تعریف شده در آرایه را فراهم می‌کند. اگر از این کلید واژه 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 Null 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 خانهٔ آرایه است.
- As کلید واژه‌ای است که اعلان نوع را مشخص می‌کند. هنگام تغییر بُعد یک آرایه، کلید واژهٔ As اختیاری است.
- DataType یک نوع دادهٔ معتبر در ویژوال بیسیک است. هنگام تغییر بُعد یک آرایه، DataType اختیاری است و با کلید واژهٔ ReDim نمی‌تواند تغییر یابد.

برای مثال اگر بخواهیم پس از دریافت تعداد دانش‌آموزان یک کلاس نام و نام خانوادگی آن‌ها را در آرایه‌ای وارد کنیم، تعداد عناصر آرایه پس از دریافت تعداد دانش‌آموزان کلاس تعیین می‌شود لذا باید هنگام اعلان آرایه تعداد عناصر را تعیین نکنیم.

```
Dim fnam () As String
Count = InputBox ("please Enter count of students")
ReDim fnam (Count)
```

نکته

اگر آرایه‌ای ایجاد می‌کنید که بعداً می‌خواهید آن را تغییر اندازه دهید، اندازهٔ آن را هنگام اعلان، تعیین نکنید. بنابراین، در عمل نمی‌توان آرایهٔ strMyArray را به‌صورت زیر تغییر اندازه داد:

```
Public strMyArray(8)As string
ReDim Preserve StrMyArray (9)
StrMyArray (9) = "Glaiol"
```

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

```
Dim strMyArray() As String
ReDim strMyArray (8)
```

```
strMyArray(0) = "Maryam"
strMyArray(1) = "Roze"
strMyArray(2) = "Niloofar"
strMyArray(3) = "Yakh"
strMyArray(4) = "Yas"
strMyArray(5) = "Banafsheh"
strMyArray(6) = "Zanbagh"
strMyArray(7) = "Magnolia"
strMyArray(8) = "Aftabgardan"
ReDim Preserve strMyArray(9)
strMyArray(9) = "Glaiol"
```

توجه داشته باشید که در کد فوق، اولین دستور ReDim از کلید واژهٔ Preserve استفاده نکرده است و دلیل آن هم این است که در آرایه مقداری وجود ندارد که نگه‌داری شود. در دومین دستور ReDim، کلید واژهٔ Preserve خیلی مهم است زیرا عناصر آرایه دارای مقدار هستند که نمی‌خواهیم از دست بروند. اگر از این کلید واژه در این خط استفاده نکنید، آرایهٔ 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) = "Glaiol"
```

کاربرد حلقه‌ها برای پیمایش آرایه

همان‌طورکه بیان شد، آرایه متغیری با چندین خانه است که در هر خانه می‌توان یک مقدار را نگه داشت. برای مقداردهی، چاپ مقادیر و انجام عملیات روی آرایه، باید دستورات را به تعداد خانه‌های آن تکرار کرد و برای تکرار باید از ساختارهای تکرار استفاده کرد. به دلیل این‌که تعداد دفعات تکرار در آرایه‌ها مشخص است، معمولاً از حلقهٔ For . . . Next استفاده می‌شود که مقدار شمارندهٔ آن از کران پایین تا کران بالای آرایه خواهد بود.

از دستورات LBound و UBound برای تعیین کران پایین و بالای آرایه استفاده می‌شود.

```
LBound (ArrayName)
UBound (ArrayName)
```

مثال ۱ــ۴

فرمی ایجاد کنید که دارای یک کادر متن به نام Textl و یک دکمهٔ فرمان به نام cmdAdd باشد. می‌خواهیم ۱۰ اسم را در کادر متن تایپ کنیم و با کلیک روی دکمهٔ Add آن‌ها را به آرایه اضافه کنیم و بعد از گرفتن دهمین اسم ، کادر متن و دکمهٔ فرمان، غیرقابل رؤیت باشند و تمام اسامی را به همراه طول آن‌ها نمایش دهد.

```
Dim A(9) As String
Dim i As Integer

Private Sub cmdAdd_Click()
        A(i) = Textl. Text  : A مقداردهی عنصر iام آرایه
        Text1. Text = ""  : Text1 پاک‌کردن محتوای
        If i < 9 Then  : آیا تمام ۱۰ عنصر آرایه مقداردهی نشده‌اند؟
            i=i+l
        else
            Textl. Visible = False  ، Textl غیرقابل رؤیت شدن
            cmdAdd. Visible = False  ،cmdAdd غیرقابل رؤیت شدن
            For i = 0 To 9  ، چاپ عناصر آرایه به همراه طول آن‌ها
```

```
            Print A(i), Len(A(i))
        Next
    End If
End Sub
```

شمارهٔ عناصر آرایهٔ A در متغیر سراسری i قرار دارد. لذا مقدار اولیهٔ آن را در رویداد Form–Load تعیین می‌کنیم.

```
Private Sub Form_Load()
    i = 0
End Sub
```

تمرین: برنامه را با کلید f8 خط به خط اجرا کرده و به کمک پنجره watch محتوای متغیر i و آرایهٔ A را مشاهده کنید.

مثال ۲ــ۴

کد زیر، مثال دیگری از کاربرد حلقهٔ For . . . Next برای پیمایش آرایه است. این کد مربوط به رویداد Click دکمه‌ای به نام cmdTraverse است که آرایه‌ای به‌طول ۲۰ عنصر را ایجاد می‌کند. حلقهٔ For . . . Next دوبار استفاده شده است (ابتدا برای تعیین مقادیر عناصر آرایه و سپس پیداکردن مقدار هر عنصر و ایجاد رشته‌ای که مقدار هر عنصر را گزارش می‌کند). به شکل ۳ــ۴ توجه کنید.

```
Private Sub cmdTraverse_Click()
    Dim i As Integer
    Dim iMyArray(19) As Integer
    Dim BeginMsg As String*15
    Dim MidMsg As String*15
    Dim LoopMsg As String*30
    Dim FullMsg As String*500
    For i = 0 To 19  → مقداردهی عناصر آرایه،
        iMyArray(i) = i*2
```

```
        Next i
        BeginMsg = "The element is: "
        MidMsg = ",The value is: "
        For i = 0 To 19      → چاپ مقدار عناصر آرایه ،
            LoopMsg = LoopMsg & BeginMsg & CStr(i)
            LoopMsg = LoopMsg & MidMsg & iMyArray(i)
            FullMsg = FullMsg & LoopMsg & vbCrLf
            LoopMsg =""
        Next i
        txtTraverse. Text = FullMsg
End Sub
```

شکل ۳ــ۴ــ این برنامه هر عنصر آرایه را با دو برابر شمارهٔ آن عنصر مقداردهی می‌کند.

کد فوق ساده است. همان طورکه می‌دانید حلقهٔ For . . . Next یک متغیر شمارنده دارد که در این برنامه مقدار اولیهٔ 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 شمارهٔ در یک عنصر آرایه خواهد بود.

متغیر LooPMsg مجموع رشته‌هایی است که در یک خط قرار می‌گیرند و متغیر FullMsg مجموع رشته‌های تمام خطوط است که در txtTraverse نمایش داده می‌شود. از عملگر & برای اتصال رشته‌ها و از ثابت VbCrlf برای انتقال مکان‌نما به سر خط بعدی استفاده شده است.

۴ــ۴ــ کنترل‌های ListBox و ComboBox

در بین همهٔ کنترل‌های استاندارد، ListBox (کادرلیست) و ComboBox (کادر ترکیبی) مناسب‌ترین کنترل برای گروه‌بندی و لیست‌بندی اطلاعاتی است که کاربران می‌توانند انتخاب کنند. روش کار با این دو کنترل، یکسان است.

لیست در ویژوال بیسیک، آرایه‌ای از رشته‌هاست که در مشخصهٔ List قرار داده شده‌اند. عملیاتی که می‌توان با این کنترل‌ها انجام داد، اضافه و حذف کردن رشته‌ها از مشخصهٔ List است.

ComboBox در واقع ترکیب کنترل‌های ListBox و TextBox است.

تفاوت این دو کنترل عبارت است از :

- در کادرلیست، لیست رشته‌ها معلوم است ولی در کادر ترکیبی، لیست به‌صورت منوی بازشو است.
- در کادرلیست، امکان چندستونی نوشتن وجود دارد ولی در کادر ترکیبی چنین نیست.

مشخصه‌های مهم این دو کنترل عبارت است از :

- List؛ محتوای این دو کنترل از طریق این مشخصه وارد می‌شود. List نام آرایه‌ای‌است که اطلاعات در آن قرار دارد و از نوع رشته‌ای می‌باشد. شماره اولین عنصر آن صفر است ((List (0)
- ListCount؛ تعداد خطوط لیست را برمی‌گرداند.
- ListIndex؛ شمارهٔ خط جاری لیست را برمی‌گرداند.
- Sorted؛ در صورت True بودن، لیست را مرتب می‌کند.

۱ــ۴ــ۴ــ اضافه کردن رشته‌ها به ListBox یا ComboBox

اضافه‌کردن رشته‌ها به دو روش انجام می‌شود.

۱ــ در زمان طراحی فرم : در مشخصهٔ List مقادیر رشته‌ای را وارد می‌کنیم. دقت کنید که باید پس از هر رشته کلیدهای Ctrl+Enter را فشار دهید تا مکان‌نما به سر خط رفته و آمادهٔ دریافت رشتهٔ بعدی شود.

۲ــ در زمان کدنویسی : از متدی به نام AddItem استفاده می‌شود که فرم کلی آن به‌صورت زیر است :

```
Object. AddItem StringToAdd
```

- Object ، مشخصهٔ Name کنترل کادر لیست یا کادر ترکیبی است.

- AddItem، کلید واژهٔ VB برای متد است.
- StringToAdd ، رشته‌ای است که می‌خواهید به لیست کنترل اضافه کنید.

مثال ۳ــ۴

فرض کنید کنترلی به نام List1 داریم و می‌خواهیم نام ۴ گل را در آن وارد کنیم. برای انجام این کار به‌صورت زیر عمل می‌کنیم :

```
Private Sub Form_Load()
    List1 . AddItem "Roze"
    List1 . AddItem "Maryam"
    List1 . AddItem "Yas"
    List1 . AddItem "Laleh"
End Sub
```

۲ــ۴ــ۴ــ انتخاب عنصرها از لیست : برای آشنایی با چگونگی تعیین مقدار رشتهٔ انتخابی در List کنترل ListBox و ComboBox، نیاز دارید بدانید که List آرایه‌ای از رشته‌هاست و عناصر آن به ترتیب با نام List(0) ، List(1)، List(2) ، ... می‌باشد.

همان‌طورکه می‌دانید برای دسترسی به عنصر دوم آرایه‌ای به نام MyArray، دستوری به‌صورت زیر می‌نویسیم (عنصر اول در خانهٔ صفر قرار دارد) :

```
MyValue = MyArray (l)
```

کنترل‌های ListBox و ComboBox نیز از قالب مشابهی استفاده می‌کنند. بنابراین برای به‌دست‌آوردن دومین رشته در ListBox به نام List1، دستوری به‌صورت زیر خواهیم داشت :

```
secondString = List1 . List (1)
```

برای دسترسی به عنصر انتخاب شده از لیست دو روش وجود دارد :

۱ــ در هر کنترل ListBox و ComboBox، شمارهٔ عنصر انتخاب شده در لیست، در مشخصه‌ای به نام ListIndex قرار می‌گیرد. بنابراین، برای تعیین مقدار رشتهٔ انتخاب‌شده در List1، کد صفحه بعد را به‌کار خواهید برد :

List1 . list (شماره عنصر انتخاب شده)

⇒List1 . List(list1 .listIndex)

شماره عنصر انتخاب شده = List1 . ListIndex

در کد زیر هنگامی که کاربر روی رشته‌ای در ListBox کلیک می‌کند، کد اجراشده و مقدار انتخاب‌شده در لیست را به مشخصهٔ Caption برچسبی به نام LabeL1 انتساب می‌دهد. شکل ۴ــ۴ رویدادهایی را که هنگام انتخاب یک رشته رخ می‌دهند نشان می‌دهد.

```
Private Sub List_Click()
    LabeL1 . Caption = List1 . List (List1 . ListIndex)
End Sub
```

شکل ۴ــ۴ــ می‌توان برای رویدادهای Click، MouseUp یا MouseDown یک ListBox کد نوشت تا مقدار انتخاب‌شده را به‌دست آورد.

۲ــ روش سریع‌تر برای به‌دست‌آوردن مقدار یک رشتهٔ انتخاب‌شده در کنترل‌های ListBox یا ComboBox، استفاده از مشخصهٔ Text است. به‌عنوان 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
```

۳ــ۴ــ۴ــ حذف عنصرها از لیست : با استفاده از متد RemoveItem می‌توان رشته‌ای را از لیست ListBox و ComboBox حذف کرد :

```
Object. RemoveItem Index
```

- Object، مشخصهٔ Name کنترل ListBox یا ComboBox است.
- RemoveItem کلیدواژهٔ 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 انتخاب‌شده از لیست، از مشخصهٔ ListIndex استفاده کنید.

شکل ۵ــ۴ بهینه سازی شدهٔ برنامهٔ شکل ۴ــ۴ را نشان می‌دهد. یک دکمه برای حذف رشته‌ای که کاربر از ListBox انتخاب می‌کند، اضافه شده است. کد زیر، چگونگی استفاده از متد RemoveItem را نشان می‌دهد.

```
Private Sub cmdRemove_Click()
    List1. RemoveItem (List1. ListIndex)
End Sub
```

شکل ۵ــ۴ــ دکمهٔ فرمان روشی برای حذف عنصری از لیست است.

۴ــ۴ــ۴ــ پاک‌کردن لیست : درصورتی که می‌خواهید تمام رشته‌های موجود در ListBox یا ComboBox را حذف کنید، از متد Clear استفاده کنید :

```
Object.Clear
```

بنابراین، برای پاک‌کردن لیست شکل ۷ــ۴ خواهیم نوشت :

```
List1.Clear
```

۵ــ۴ــ۴ــ 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 دادهٔ خارج از لیست را انتخاب یا وارد کرد. ComboBox قابلیت انعطاف بیشتری از خود نشان می‌دهد و از فضای فرم به صورت کارآمد استفاده می‌کند.

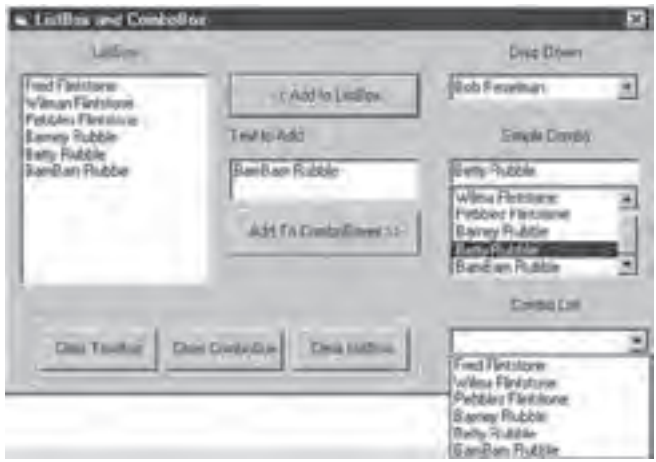
مشخصهٔ Style کنترل ComboBox امکان تغییر مشخصه‌های عملیاتی و ظاهر کنترل را فراهم می‌کند. جدول ۱ــ۴ این شیوه ها را شرح می‌دهد و شکل ۶ــ۴ شیوه‌های ComboBox اعمال شده به فرم را نشان می‌دهد.

نکته

هنگامی که یک ComboBox را با شیوهٔ 1–Simple Combo به فرم اضافه می‌کنید، ComboBox تغییر شکل یافته و به‌صورت ComboBox مشاهده نمی‌شود. مشخصهٔ Height را افزایش دهید تا ListBox آن نمایش یابد.

جدول ۱ــ۴ــ مقادیر مربوط به مشخصهٔ Style کنترل ComboBox

تنظیم	شرح
0 - Drop – Down Combo	یک لیست بازشو. کاربران می‌توانند داده‌های 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 دادهٔ جدیدی را در TextBox وارد کنند. اندازهٔ 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 ۶ــ۴ــ توجه کنید که با شیوهٔ ComboBox بازشو، می‌توان داده‌های جدیدی را در زمان اجرا به کنترل اضافه کرد.

متداول‌ترین شیوه‌های ComboBox شیوه‌های ۰ و ۲ هستند. همان‌طورکه قبلاً نیز بیان شد، هنگامی که می‌خواهید به کاربران امکان اضافه‌کردن داده‌های جدیدی را که در لیست نیستند بدهید از شیوهٔ ComboBox بازشو (۰) استفاده کنید. درصورتی که می‌خواهید کاربران فقط امکان انتخاب داشته باشند، از شیوهٔ لیست بازشو (۲) استفاده کنید.

مثال ۴ــ۴

فرمی به‌صورت زیر ایجاد کنید :

شکل ۷ــ۴

یک کادر لیست و یک کادر ترکیبی به نام‌های Lstflower و Cboprice به همراه دو کادرمتن و یک دکمهٔ فرمان وجود دارد. با کلیک روی دکمهٔ Go متن موجود در هر کادر متن، در کادر لیست مربوطه قرار می‌گیرد. کد مربوط به این مثال به‌صورت زیر خواهد بود :

```
Private Sub cmdGo_Click()
    If txtprice. Text <>"" Then
        Lstprice.AddItem txtprice. Text
        txtprice.Text = ""
    End If
    If txtflower. Text <>"" Then
        cboflower. AddItem txtflower. Text
        txtflower. Text = ""
    End If
End Sub
```

تمرین : مثال ۴ـ۴ را طوری تغییر دهید که:

الف) اگر نامی تکراری وارد شد، به لیست اضافه نکند.

ب) دکمه‌ای برای پاک‌کردن هر دو لیست اضافه کند.

ج) دکمه‌ای اضافه کند که با کلیک آن قیمت گلی تعیین شود که نام آن در txtflowers آمده است. قیمت گل در کنترل Labled نمایش داده شود (راهنمایی: گل و قیمت آن در لیست‌ها دارای شمارهٔ یکسان هستند.)

مثال ۵ــ۴

برنامهٔ زیر از تابع StrComp برای مقایسهٔ دو رشته استفاده می‌کند. رابط کاربر برای این برنامه شامل چندین برچسب (Label)، دو کادر متن (TextBox) و یک دکمهٔ فرمان (Command Button) به همراه یک لیست است. برای مقایسهٔ دو رشته، ابتدا رشته‌ها را در درون دو کادر متن تایپ کرده و سپس بر روی دکمهٔ Compare کلیک نمایید. برنامه با استفاده از تابع StrComp به مقایسهٔ دو رشته

می‌پردازد و نتیجه را به‌صورت عبارتی به لیست پایین پنجره اضافه می‌کند.

```
Dim C As Integer
Private Sub cmdComp_Click()
    C = StrComp(Text1.Text,Text2.Text)
    Select case C
        Case 0 : List1. AddItem   Text1. Text + "=" + Text2. Text
        Case 1 : List1. AddItem   Text1. Text + ">" + Text2. Text
        Case −1 : List1. AddItem   Text1. Text + "<" + Text2. Text
    End Select
End Sub
```

شکل ۸ــ۴

از تابع StrComp() برای مقایسهٔ دو رشته استفاده می‌شود و شکل کلی آن به صورت زیر است :

Variable Name = StrComp(String1, String2 [,Compare])

● **Variable Name** : متغیری است که خروجی این تابع در آن قرار می‌گیرد و نوع آن عددی صحیح است.

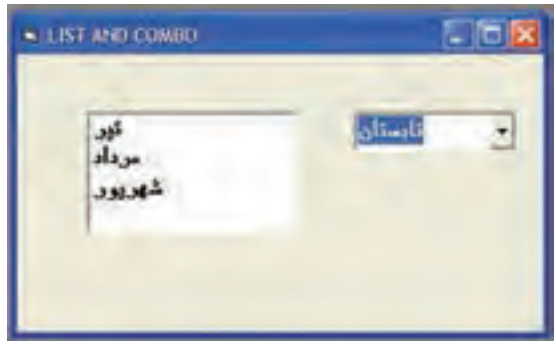
● **String 1, String 2** : دو رشته‌ای هستند که با هم مقایسه خواهند شد. اگر دو رشته با یکدیگر برابر باشند، تابع مقدار صفر را برمی‌گرداند و اگر رشتهٔ اول کوچک‌تر از رشتهٔ دوم باشد، تابع

مقدار ۱- و در صورت بزرگ‌تر بودن رشتهٔ اول از رشتهٔ دوم، مقدار ۱ برگردانده می‌شود (مقایسهٔ رشته‌ها براساس کداسکی انجام می‌شود).

● **Compare** : پارامتری است اختیاری که اگر مقدارش ۱ باشد، بدین معنی است که بین حروف کوچک و بزرگ، تفاوتی قائل نشود ولی اگر صفر باشد، بین این دو نوع حروف تفاوت قائل می‌شود.

مثال ۶ــ۴

فرمی به‌صورت زیر طراحی کنید که با انتخاب نام فصل از یک ComboBox نام ماه‌های آن فصل را در یک ListBox نمایش دهد.



شکل ۹ــ۴

نام کادر ترکیبی را Combo1 و نام کادر لیست را List1 قرار دهید.

قبل از اضافه کردن ماه‌های فصل انتخاب شده به List1 باید محتوای List1 پاک شود (دستور List1.Clear)

```
Private Sub Combol_Click()
    Select case Combol. Text
        Case "بهار"
            Listl. Clear
            Listl. AddItem "فروردین"
            Listl. AddItem "اردیبهشت"
            Listl. AddItem "خرداد"
        Case "تابستان"
```

```
            Listl. Clear
            Listl. AddItem "تیر"
            Listl. AddItem "مرداد"
            Listl. AddItem "شهریور"
        Case "پاییز"
            Listl. Clear
            Listl. AddItem "مهر"
            Listl. AddItem "آبان"
            Listl. AddItem "آذر"
        Case "زمستان"
            Listl. Clear
            Listl. AddItem "دی"
            Listl. AddItem "بهمن"
            Listl. AddItem "اسفند"
    End Select
End Sub
```

مثال ۷ــ۴

در این مثال می‌خواهیم شهر مقصد، محل صندلی و غذای دلخواه مسافران هواپیما را انتخاب کنیم. یک کادر لیست، دو کادر ترکیبی (کومبو)، سه برچسب و دو دکمهٔ فرمان روی فرم قرار داده و ظاهر آن را مطابق شکل صفحهٔ بعد تنظیم کنید :

کد برنامه به‌صورت زیر خواهد بود :

```
Private Sub Form_Load()
    lstCities.Clear
    lstCities.AddItem  "شیراز"
    lstCities. AddItem  "اصفهان"
    lstCities.AddItem  "تبریز"
```

شکل ۱۰ــ۴

```
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          قرار دادن اولین عنصر لیست در حالت انتخاب

cboSeat.AddItem "ردیف اول"
cboSeat.AddItem "وسط"
cboSeat.AddItem "کنار پنجره"
cboSeat.ListIndex = 0
```

```
cboMeal.AddItem "جوجه"
cboMeal.AddItem "برگ"
cboMeal.AddItem "سبزیجات"
cboMeal.AddItem "میوه"

cboMeal. Text = "فرقی نمی کند"
End Sub


Private Sub cmdAssign_Click()
    Dim Message As String
    Message = "مقصد"+ lstCities. Text+vbCr
    Message = Message + "محل صندلی"+ cboSeat. Text+vbCr
    Message = Message + "غذای دلخواه"+ cboMeal. Text+vbCr
    MsgBox Message,vbOKOnly +vbInformation,"سفارش شما"
End Sub


Private Sub cmdExit_Click()
    End
End Sub
```

تمرین : برنامه‌ای بنویسید که الف) نام و نام خانوادگی و نمرات برنامه‌سازی دانش‌آموزان را از طریق دو کادر متن دریافت کرده و در دو کادر لیست قرار دهد. ب) امکان حذف نام دانش‌آموز را به برنامه اضافه کنید و در صورت حذف دانش‌آموز نمره دانش‌آموز نیز حذف شود.

۵ــ۴ــ آرایهٔ کنترلی

در ویژوال بیسیک می‌توان آرایه‌هایی از انواع دادهٔ 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 همهٔ عناصر آرایهٔ 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. همهٔ این کنترل‌ها دارای مشخصهٔ Index هستند که برای شناسایی کنترل خاصی در آرایه مورد استفاده قرار می‌گیرند.

۱ــ۵ــ۴ــ ایجاد آرایهٔ 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 می‌کند.

مثال ۸ــ۴

ایجاد آرایهٔ کنترلی از CommandButtons

۱ــ پروژهٔ جدیدی را شروع کنید و نام فرم پیش فرض را به frmMain تغییر دهید.

۲ــ یک دکمهٔ فرمان به مرکز فرم frmMain اضافه کنید و نام آن را cmdMyButton قرار دهید. مشخصهٔ Caption این دکمهٔ فرمان را به Action تغییر دهید.

۳ــ دکمهٔ فرمان را انتخاب کنید. از منوی Edit گزینهٔ Copy را انتخاب کنید تا دکمهٔ فرمان به حافظهٔ Clipboard کپی شود.

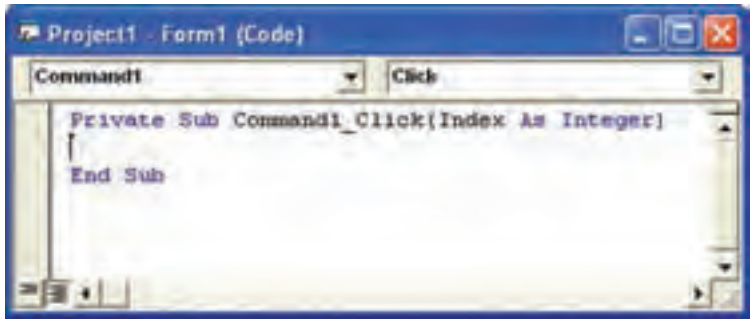
۴ــ از منوی Edit گزینهٔ Paste را انتخاب کنید. یک کادر محاوره‌ای ظاهر شده و از شما سؤال می‌کند که آیا می‌خواهید یک آرایهٔ کنترلی ایجاد کنید؟ روی Yes کلیک کنید تا آرایهٔ کنترلی ایجاد شود (شکل ۱۱ــ۴).

شکل ۱۱ــ۴ــ کادر محاوره‌ای تأیید ایجاد آرایهٔ کنترلی

بعد از ایجاد آرایهٔ کنترلی، اگر به پنجرهٔ Properties رجوع کنید و لیست بازشوی Object را نمایش دهید، مشاهده خواهید کرد که دو دکمهٔ فرمان با نام یکسان cmdMyButton وجود دارد که هرکدام دارای اندیس خاصی هستند (شکل ۱۲ــ۴).

شکل ۱۲ــ۴ــ هنگامی که کنترل بخشی از آرایهٔ کنترلی باشد، برای دسترسی به آن باید همیشه به اندیس آن اشاره کرد.

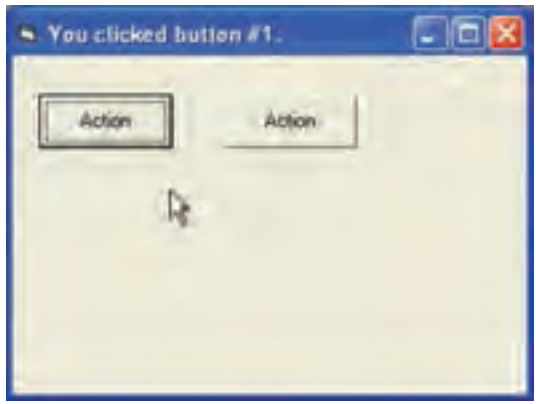
روی دکمهٔ فرمان دوبار کلیک کنید تا روال رویداد Click را مشاهده کنید. این روال رویداد دارای آرگومان Index خواهد بود (شکل ۱۳ــ۴). این آرگومان یک عدد صحیح است که اندیس کنترل را مشخص می‌کند. به دلیل این‌که همهٔ کنترل‌های آرایهٔ کنترلی نام یکسانی دارند، تفاوت بین آن‌ها با مقدار Index در روال رویداد مشخص می‌شود. صفر (۰) اولین کنترل، ۱ دومین کنترل و ۲ سومین کنترل و الی آخر را مشخص می‌کند.



شکل ۱۳ــ۴ــ هر زمانی که رویدادی برای کنترلی از یک آرایهٔ کنترلی رخ دهد، ویژوال بیسیک آرگومان Index را مقداردهی می‌کند.

کد زیر نشان می‌دهد که روی کدام دکمهٔ فرمان از آرایهٔ کنترلی کلیک شده است. شکل ۱۴ــ۴ اجرای این کد را نشان می‌دهد.

```
Private Sub cmdMyButton_Click(Index As Integer)
    Me.Caption = "You clicked button # " & Index & "."
End Sub
```



شکل ۱۴ــ۴ــ بعد از کلیک کاربر روی دکمهٔ سمت راست، عنوان فرم مطابق آن تغییر می‌یابد.

همانطورکه در مثال بالا دیدید روال رویدادها برای تمام عناصر آرایه کنترلی مشترک است (یک روال رویداد کلیک برای دو دکمه در مثال بالا) و برای تشخیص اینکه رویداد رخ داده مربوط به کدام عنصر آرایهٔ کنترلی است روال رویدادهای آرایه‌های کنترلی دارای آرگومان ورودی Index است که مشخص‌کنندهٔ شمارهٔ عنصری از آرایهٔ کنترلی است که رویداد برای آن رخ داده است.

۲ــ۵ــ۴ــ اضافه‌کردن عنصرها به آرایهٔ کنترلی در زمان اجرا : ایجاد آرایهٔ کنترلی در زمان طراحی هنگامی مفید خواهد بود که تعداد کنترل‌های مورد نیاز برای آرایه را بدانید. ولی اگر تعداد کنترل‌هایی را که در زمان اجرای برنامه نیاز خواهید داشت نمی‌دانید، چه کاری باید انجام دهید؟ این مشکل را می‌توان با استفاده از دستور Load برای افزودن کنترل‌ها به آرایهٔ کنترلی در زمان اجرا، حل کرد.

مثال ۹ــ۴

فرض کنید یک دکمهٔ فرمان به عنوان اولین عنصر آرایهٔ کنترلی داریم و می‌خواهیم در هنگام اجرا عنصر بعدی آن را ایجاد کنیم. مراحل کار به صورت زیر خواهد بود :

۱ــ پروژهٔ جدیدی را شروع کنید. نام فرم را به frmDArray تغییر دهید.

۲ــ یک دکمهٔ فرمان به گوشهٔ سمت چپ بالای فرم اضافه کنید و نام آن را cmdCtrlArray قرار دهید.

۳ــ در پنجرهٔ Properties، مقدار مشخصهٔ Index دکمهٔ فرمان را با صفر (۰) مقداردهی کنید.

۴ــ مشخصهٔ Caption دکمهٔ فرمان را به Button#0 تغییر دهید.

۵ــ کد زیر را به رویداد Form_Load اضافه کنید.

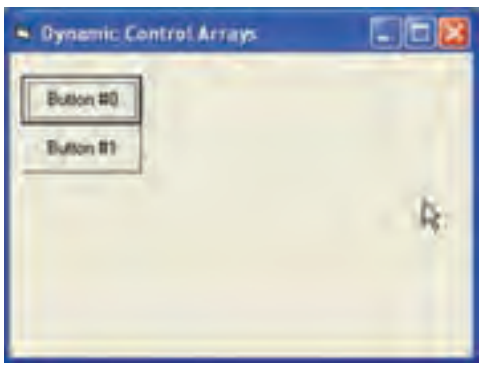
```
Private Sub Form_Load()
    Load cmdCtrlArray(1)
    cmdCtrlArray(1).Left = cmdCtrlArray(0). Left
    cmdCtrlArray(1).Top = cmdCtrlArray(0). Top + cmdCtrlArray (0).Height
    cmdCtrlArray(1).Caption = " Button # 1"
    cmdCtrlArray(1).Visible = True
End Sub
```

۶ــ کد را ذخیره کرده و اجرا کنید.

بعد از اجرای کد، برنامه یک دکمهٔ فرمان جدیدی را ایجاد می‌کند و آن را زیر اولین دکمهٔ فرمان قرار می‌دهد (شکل ۱۵ــ۴).

نکته

تمام عناصر جدیدی که از آرایهٔ کنترلی در زمان اجرا ایجاد می‌کنید دارای مشخصهٔ Visible با مقدار False هستند. هنگامی که کنترل‌های جدیدی را در زمان اجرا ایجاد می‌کنید، فراموش نکنید خطی در کد قرار دهید که مشخصهٔ Visible را با True مقداردهی کند. در غیر این صورت نمی‌توانید کنترل را مشاهده کنید.



شکل ۱۵ــ۴ــ استفاده از دستور Load دکمهٔ فرمان دیگری را روی فرم قرار می‌دهد. بقیهٔ کد محل دقیق کنترل را تعیین می‌کند.

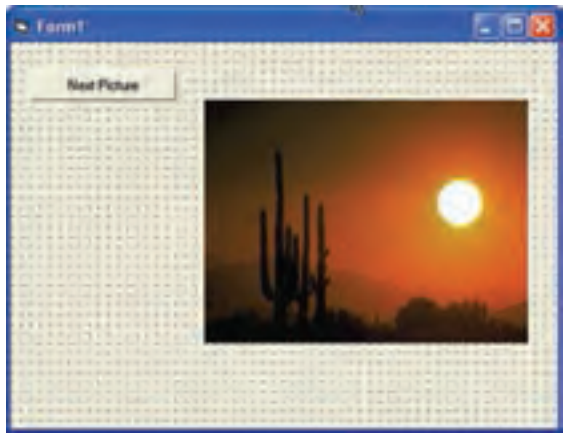
کنترل‌های جدیدی که ایجاد می‌کنید دقیقاً تکرار اولین عنصر آرایهٔ کنترلی هستند. مقادیر همهٔ مشخصه‌ها به جز Index و Visible یکسان هستند. بنابراین، هنگامی که کنترل جدیدی را ایجاد می‌کنید، این کنترل روی اولین کنترل آرایه قرار می‌گیرد و باید آن را به محل دیگری انتقال دهید که انجام این کار در زمان اجرا با تغییر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

همان‌طور که در مثال قبل مشاهده کردید، مزیت استفاده از آرایه‌های کنترلی، توانایی داشتن مجری رویداد مشترک است.

مثال ۱۰ــ۴

آرایه‌ای ۵ عنصری از تصاویر داریم که می‌خواهیم با هر بار کلیک روی دکمهٔ Next Picture تصویر بعدی نمایش داده شود.

۱ــ پروژه‌ای به صورت زیر ایجاد کنید (شکل ۱۶ــ۴). (این پروژه دارای ۵ عنصر تصویر است که روی هم قرار گرفته‌اند)



شکل ۱۶ــ۴

۲ــ کد زیر را برای فرم وارد کنید :

```
Dim cnt As Byte,i As Byte
Private Sub Cmdpic_Click()
    imgl(cnt). Visible = False
```

```
If cnt < 4 Then cnt = cnt + 1 else  Cnt = 0
img1(cnt). Visible = True
End Sub
```

در Form_load ویژگی visible تمام تصاویر به جز تصویر اول (img1(0)) را False می‌کنیم.

```
Private Sub Form_Load()
For i=1 to 4
Img1(i). Visible = False
NEXT i
cnt = 0
End Sub
```

۳ــ پروژه را اجرا کنید.

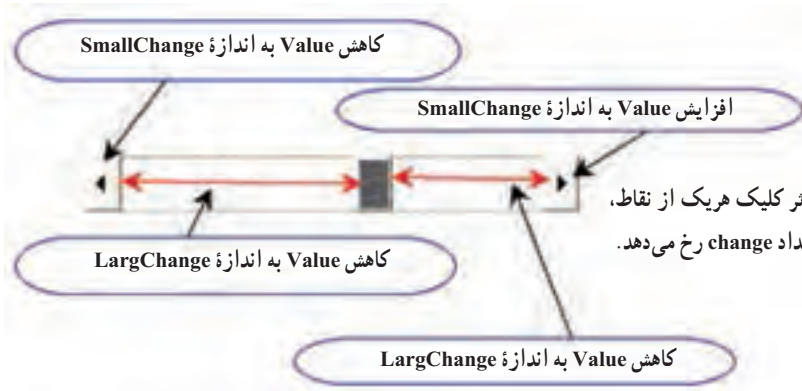
۶ــ۴ــ کاربرد کنترل‌های Scroll Bar

کنترل‌های نوار لغزان استاندارد (HscrollBar و VscrollBar) امکان تغییر داده‌ها یا جابه‌جایی در محدوده‌ای از مقادیر را با کلیک‌کردن روی دکمه های Up و Down یا با جابه‌جایی دکمهٔ لغزان، فراهم می‌کند. کنترل‌های نوار لغزان دارای چند مشخصهٔ خاص هستند که در جدول ۲ــ۴ با آن‌ها آشنا خواهید شد.

جدول ۲ــ۴ــ مشخصه‌های خاص کنترل‌های HscrollBar و VscrollBar

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

کنترل‌های نوار لغزان دارای دو رویداد Change و Scroll هستند. هرگاه مقدار Value تغییر کند بر روی فلش‌ها و یا فضای بین فلش‌ها کلیک کنیم و یا دکمه لغزان جابه‌جا شود رویداد Change رخ می‌دهد. هنگام حرکت‌دادن دکمهٔ لغزان چندین رویداد Scroll رخ می‌دهد و به محض متوقف‌شدن دکمه یک رویداد Change رخ می‌دهد.



شکل ۱۷ــ۴ــ میزان تغییرات value دراثر کلیک نقاط مختلف در ScrollBar

مثال ۱۱ــ۴

پروژه‌ای ایجاد کنید که با تغییر نوارهای لغزان افقی و عمودی، مقدار آن‌ها را در کادر متن‌های جداگانه نمایش دهد.

۱ــ پروژهٔ جدیدی به نام SmpScroll. vbp ایجاد کنید. نام فرم را frmMain قرار دهید.

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

۳ــ دو کنترل به کادر متن اضافه کنید؛ نام یکی را txtNS و دیگری را txtWE قرار دهید. مشخصهٔ Text هر دو کنترل را یک رشتهٔ خالی قرار دهید (شکل ۱۸ــ۴).

۴ــ مشخصه‌های کنترل‌های VscrollBar و HscrollBar را مطابق جدول ۳ــ۴ مقداردهی کنید.

جدول ۳ــ۴ــ تنظیم مشخصه‌های کنترل‌های ScrollBar

مشخصه	مقدار
Min	0
Max	20
SmallChange	1
LargeChange	2

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

نمایش مقدار مشخصهٔ Value نوار لغزان افقی در کادر متن txtWE

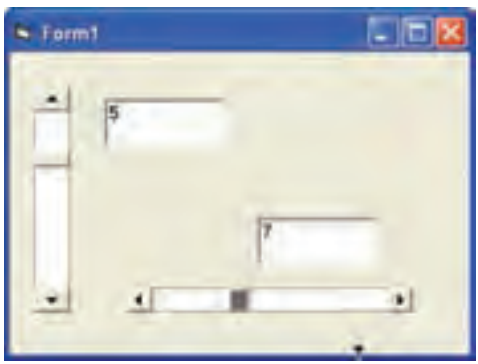
```
Private Sub hscrWE_Change()
    txtWE.Text = CStr(hscrWE.Value)
End Sub
```

نمایش مقدار مشخصهٔ Value نوار لغزان عمودی در کادر متن txtNS

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

۶ــ پروژه را ذخیره کرده و اجرا کنید.

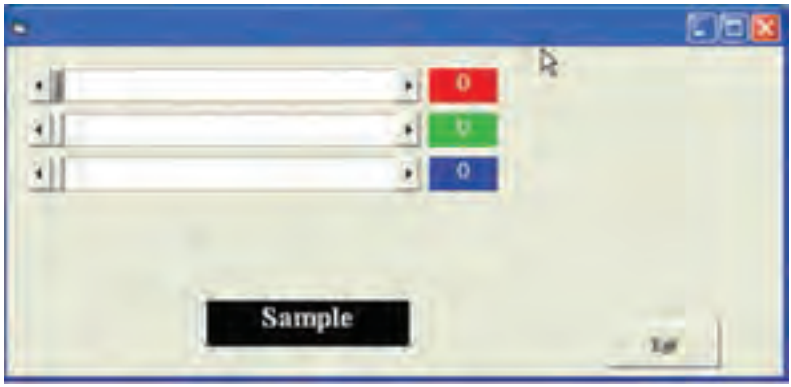
هنگامی که پروژه را اجرا کردید، با کلیک روی فلش‌های کنترل‌های HscrollBar و VscrollBar، مقدار کادر متن مربوطه یک واحد تغییر می‌کند (براساس مقدار مشخصهٔ SmallChange). ولی اگر بین فلش و دکمهٔ لغزان کلیک کنید، مقدار کادر متن مربوطه، دو واحد تغییر می‌کند (براساس مقدار مشخصهٔ LargeChange) (شکل ۱۸ــ۴).



شکل ۱۸ــ۴ــ به دلیل این‌که مشخصهٔ Max با ۲۰ مقداردهی شده است، هنگام جابه‌جایی نوار لغزان، مقادیر نشان داده شده در کادرهای متن تا ۲۰ می‌رسند.

مثال ۱۲ــ۴

فرمی به صورت شکل ۱۹ــ۴ ایجاد کنید. با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی و سبز می‌توان رنگ برچسب را تغییر داد.



شکل ۱۹ــ۴

مقدار Value هر نوار لغزان روی برچسبی که کنار آن قرار دارد نشان داده می‌شود. و رنگ زمینهٔ برچسب Label1 براساس مقدار Value هر سه نوار لغزان تعیین می‌شود.

کد مربوطه را به فرم اضافه کنید :

```
Dim r,g,b As Byte
Private Sub Command1_Click()
    End
End Sub

Private Sub Form_Load()
    LabeL1 . BackColor = RGB(0, 0, 0)
End Sub

Private Sub HScroll l_Change()
    r = HScroll1 . Value
```

```
        g = HScroll2.Value
        b = HScroll3.Value
        Lblr.Caption = r
        Lblg.Caption = g
        Lblb.Caption = b
        LabeL1.BackColor = RGB(r,g,b)
End Sub
Private Sub HScroll1_Scroll()
        r = HScroll1.Value
        g = HScroll2.Value
        b = HScroll3.Value
        Lblr.Caption = r
        Lblg.Caption = g
        Lblb.Caption = b
        LabeL1. BackColor = RGB(r, g, b)
End Sub

Private Sub HScro112_Change()
        r = HScroll1.Value
        g = HScroll2.Value
        b = HScroll3.Value
        Lblr.Caption = r
        Lblg.Caption = g
        Lblb.Caption = b
        LabeL1. BackColor = RGB(r, g, b)
End Sub
```

```
Private Sub HScroll2_Scroll()
        r = HScroll1 . Value
        g = HScroll2 .  Value
        b = HScroll3 .  Value
        Lblr . Caption = r
        Lblg . Caption = g
        Lblb . Caption = b
        LabeL1 .  BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll3_Change()
        r = HScroll1 . Value
        g = HScroll2 .  Value
        b = HScroll3 .  Value
        Lblr . Caption = r
        Lblg . Caption = g
        Lblb . Caption = b
        LabeL1 .  BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll3_Scroll()
        r = HScroll1 . Value
        g = HScroll2 . Value
        b = HScroll3 . Value
        Lblr . Caption = r
        Lblg . Caption = g
        Lblb . Caption = b
```

```
    Labell. BackColor = RGB(r, g, b)
End Sub
```

تمرین ۱: در مثال ۱۲ـ۴ کنترل‌های نوار لغزان و برچسب‌های کنار آن را به‌صورت آرایه‌های کنترلی طراحی کنید.

تمرین ۲: مثال ۱۲ـ۴ را به گونه‌ای تغییر دهید که رنگ زمینه از طریق یک کادر لیست انتخاب شود و شدت تمام رنگ‌ها با یک Scrollbar تعیین شود.

۷ـ۴ـ مرتب‌سازی عنصرهای آرایه‌ها

مرتب‌کردن داده‌ها، یکی از عملیات مهم در برنامه‌های کاربردی است. الگوریتم‌های متعددی برای مرتب‌کردن 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 مرتب‌شونده، بخشی از هر عنصر که مقایسه و مرتب‌سازی براساس آن بخش انجام می‌شود، «کلید» نام دارد. در یک الگوریتم مرتب‌سازی همواره دو عمل «مقایسه» و «تعویض» انجام می‌گیرد.

شرکت مخابرات برحسب نام خانوادگی افراد، و در صورت تکراری‌بودن نام، شماره تلفن‌ها را در کتابچهٔ راهنمای تلفن مرتب می‌کند که با این روش، جست‌وجوی شمارهٔ تلفن فرد مورد نظر آسان‌تر خواهد شد.

یکی از روش‌های مرتب‌سازی عناصر آرایه روش حبابی (Bubble Sort) است. در این روش عناصر آرایه دو به دو مقایسه می‌شوند و عنصر کوچکتر قبل از عنصر بزرگتر قرار می‌گیرد. فرض کنید که A لیستی از n عدد باشد. می‌خواهیم عنصار A را به‌صورت افزایشی (صعودی) مرتب کنیم یعنی

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

برای شروع، عناصر A را دو به دو مقایسه می‌کنیم. A(0) با A(1) اگر A(0) بزرگتر از A(1) بود، محتوای آن‌ها را جابجا می‌کنیم. سپس A(1) با A(2) مقایسه می‌شود و به همین ترتیب A(2) با A(3)، A(3) با A(4)، ... تا A(n−1) با A(n). تا اینجا یک مرحله از مقایسه‌ها انجام شده و باید این مراحل را تکرار کنیم تا تمام عناصر مرتب شود. حداکثر تعداد این مراحل برای مرتب‌شدن کل آرایه (n−1) می‌باشد.

برای جابه‌جایی محتوای دو خانه از متغیر سومی باید استفاده کنیم.

در تکه کد زیر برای جابجاکردن محتوای دو خانه از متغیر inttemp استفاده شده است.

```
If intarray A(intComp) > intarray A (intComp+1) then
    inttemp = intarray A(intComp)
```

```
intarray A (intComp) = int array A(intComp+1)
intarray A(intComp+1) = inttemp
End If
```

مقایسهٔ دو عنصر کنارهم و در صورت لزوم جابجاکردن محتوای آن‌ها

عمل مقایسهٔ دو به دو در اولین مرحله به تعداد n−۱ بار انجام می‌شود و در هر مرحله یکی از عناصر مرتب می‌شود لذا از تعداد مقایسه‌های دو به دو یکی کم می‌شود، به گونه‌ای که در مرحله اول تعداد مقایسه‌ها n−۱، در مرحله دوم تعداد مقایسه‌ها n−۲، ... می‌باشد. لذا تعداد مقایسه‌های هر مرحله به شمارهٔ آن مرحله بستگی دارد و اگر شماره مرحله را در متغیر intpass قرار دهیم تعداد مقایسه‌ها در هر مرحله n−intpass می‌باشد که n تعداد عناصر آرایه (کران بالای آرایه) است لذا تکه کد زیر عمل مقایسه در هر مرحله را نشان می‌دهد.

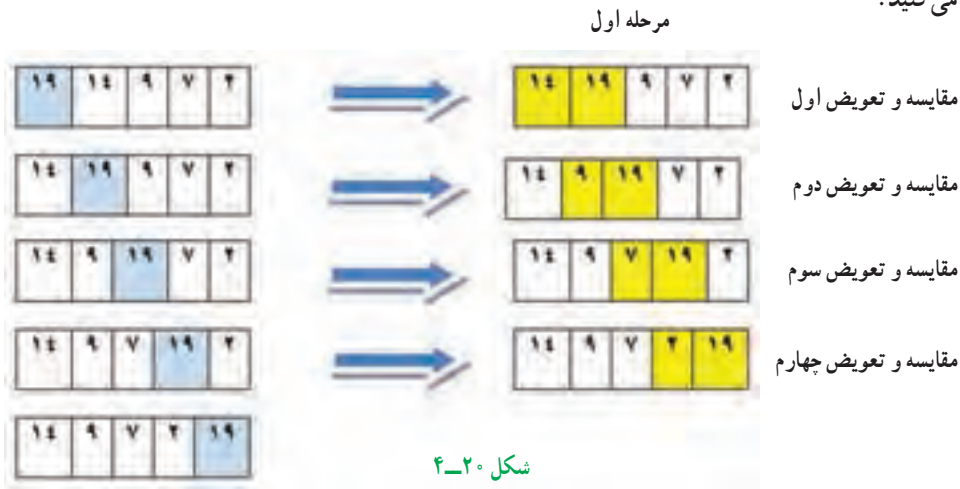
```
For intComp = LBound (intarray A) To UBound (intarray A) – intPass
```

برای مثال آرایه A به‌صورت :

A(1)	A(2)	A(3)	A(4)	A(5)
۱۸	۱۴	۹	۷	۲

را در نظر بگیرید.

در شکل ۲۰ــ۴ نتیجه اجرای قطعه کد قبل برای intpass=1 را روی آرایه A مشاهده می‌کنید.



شکل ۲۰ــ۴

نتیجه آرایه پس از مرحله اول و مرتب‌شدن عنصر آخر آرایه (عدد ۱۹)

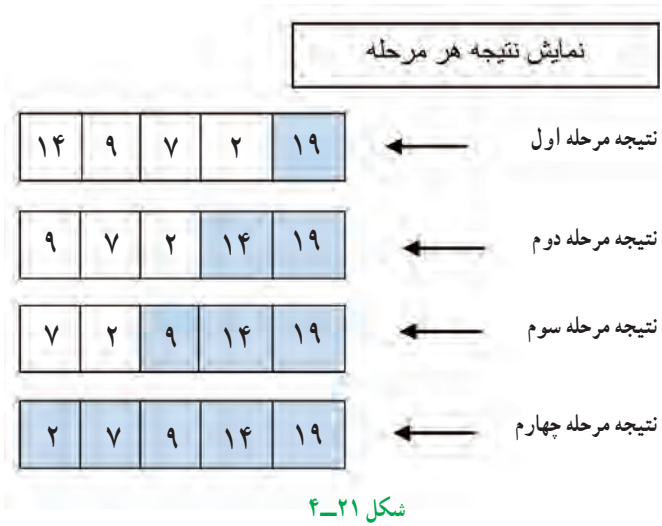
برای مرتب‌شدن تمام عناصر آرایه، این مراحل حداکثر n−۱ بار باید تکرار شود لذا تکه کد قبل باید درون یک حلقه For قرار گیرد که متغیر این حلقه شماره مرحله را نگهداری می‌کند.

۱ــ For intpass = LBound (intarray A) To UBound (intaray A)

تکه کد مربوط به اجرای هر مرحله

Next

خروجی آرایه A پس از هر مرحله در شکل ۲۱ــ۴ آمده است.



شکل ۲۱ــ۴

همانطورکه در جدول مشاهده می‌کنید در هر مرحله یکی از عناصر آرایه مرتب می‌شود. در مرحله اول A(۵) (۱۸)، در مرحله دوم A(۴) (۱۴)، در مرحله سوم A(۳) (۹) و در مرحله چهارم A(۱) و A(۲) (۲،۷) با هم مرتب می‌شوند. از آن‌جا که دو عنصر A(۱) و A(۲) با هم مرتب می‌شوند تعداد مراحل از تعداد عناصر آرایه یکی کمتر است.

مثال ۱۳ــ۴

پروژه‌ای ایجاد کنید که با کلیک روی دکمهٔ Create Data ده (۱۰) عدد صحیح تصادفی در یک کادر لیست قرار بگیرند، سپس با کلیک روی دکمهٔ Sort این اعداد مرتب شده و در کادر لیست دوم کپی شوند.

برنامهٔ زیر، مقادیر آرایهٔ mArray را به صورت صعودی مرتب می‌کند. تکنیکی که ما در این برنامه استفاده کرده‌ایم، Bubble Sort (مرتب‌سازی حبابی) است. روال Bubble Sort عمل مرتب‌سازی را انجام می‌دهد.

شکل ۲۲ــ۴

شکل ۲۳ــ۴

```
Option Explicit
Option Base 1  اندیس اولین عنصر آرایه یک می‌باشد
```

```
Dim intarrayA(10) As Integer
Private Sub cmdGenerate_Click ()
    Dim inti As Integer
    lstOriginal.Clear        خالی کردن کادر لیست اولیه
    For inti  = LBound(intarrayA) To UBound(intarrayA)
    مقداردهی عناصر آرایه با اعداد تصادفی کوچکتر از ۱۰۰
        intarrayA(inti) = Int (Rnd*100)
        lstOriginal.AddItem intarrayA(inti)
    قراردادن عناصر آرایه در کادر لیست اولیه
    Next
    lstSorted.Clear                    lstSorted خالی کردن کادر لیست
    cmdSort. Enabled = True
End Sub

Private Sub cmdSort_Click()
    Dim intpass As Integer, intcomp As Integer
    Dim inttemp As Integer
    Dim inti As Integer
    IstSorted.Clear        حلقهٔ شمارنده مراحل
    For intpass = LBound(intarrayA) To UBound (intarrayA) – 1
    حلقهٔ مقایسهٔ دو به دوی اعداد
        For intcomp = LBound (intarrayA) To UBound(intarrayA) – intpass
    جابه‌جاکردن عناصر آرایه در صورت لزوم
            If  intarrayA(intcomp) > intarrayA(intcomp + 1) Then
                inttemp = intarrayA(intcomp)
                intarrayA(intcomp) = intarrayA(intcomp + 1)
                intarrayA(intcomp + l) = inttemp
```

```
                End If
            Next
        Next
                    اضافه‌کردن عناصر آرایهٔ مرتب‌شده به کادر لیست LSt sorted
        For inti = LBound(intarrayA) To UBound (intarrayA)
            lstSorted.AddItem intarrayA(inti)
        Next
End Sub

Private Sub cmdExit_Click()
        End
End Sub
Private Sub Form_Load()
        Randomize
        cmdSort.Enabled = False
End Sub
```

۸ــ۴ــ جست‌وجو

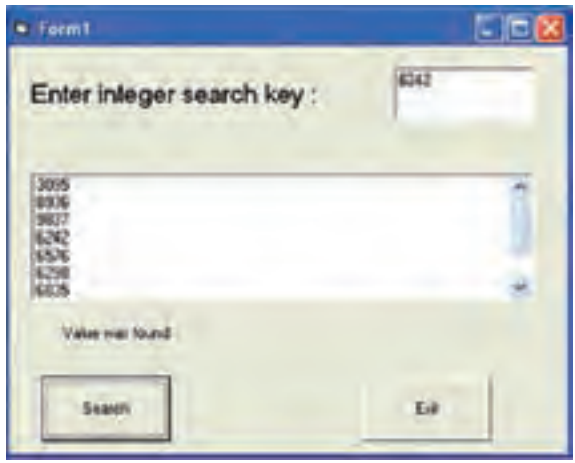
در اغلب موارد می‌خواهید با مقادیر زیادی از داده‌ها که در آرایه‌ها ذخیره شده‌اند، کار کنید و ممکن است تعیین مقداری که در آرایه وجود دارد، ضروری باشد. عملیاتی که به منظور یافتن عضوی در آرایه صورت می‌گیرد به نام جست‌وجو (Searching) معروف است.

۱ــ۸ــ۴ــ جست‌وجوی خطی : جست‌وجوی خطی در مورد آرایه‌های کوچک یا آرایه‌های مرتب نشده خوب عمل می‌کند ولی در مقابل آرایه‌های بزرگ و مرتب کارایی بهینه ندارد.

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

مثال ۱۴ــ۴

برنامه‌ای بنویسید که ۱۰ عدد حداکثر۴رقمی تولید کند و در صورت یافتن کلید جست‌وجو شمارهٔ عنصر و در صورت نیافتن پیغام مناسب نمایش دهد.



شکل ۲۴ــ۴ــ جست‌وجوی خطی

```
Option Explicit
Option Base 1
Dim intarrayA(10) As Integer

Private Sub cmdExit_Click()
    End
End Sub

Private Sub cmdSearch_Click()
    Dim intkey As Integer
    Dim inti As Integer
    Dim intindex As Integer
```

```
    Dim blnfount As Boolean
    lblresult.Caption = " "
    intkey = Val(txtkey)
    blnfound = False
                                        مقایسهٔ عناصر آرایه با کلید جست‌وجو
    For inti = LBound(intarrayA)To UBound(intarrayA)
        If intarrayA(inti) = intkey Then
            blnfound = True
            intindex = inti
            Exit For
        End If
    Next
    If blnfound = True Then
        lblResult = "Item=" & intindex
    Else
        lblResult = "Not Found"
    End If
End Sub

Private Sub Form_Load()
    Dim inti As Integer
    Call Randomize
    Call lstData.Clear
    lblResult = " "
    txtkey = " "
                    *** LstData تولید اعداد تصادفی و اضافه‌کردن به کادر لیست ***
```

```
    For inti = LBound(intarrayA) To UBound(intarrayA)
        intarrayA(inti) = Int(Rnd*10000)
        lstData. AddItem intarrayA (inti)
    Next
End sub
```

۲ــ۸ــ۴ــ جست‌وجوی دودویی : این روش جست‌وجو برای آرایه‌های بزرگ مناسب است. به دلیل این‌که در این نوع آرایه‌ها با روش جست‌وجوی خطی، تعداد دفعات جست‌وجو زیاد خواهد بود که سبب اتلاف زمان می‌شود.

روش جست‌وجوی دودویی، روی آرایه‌های مرتب قابل اجراست. بنابراین، در صورتی که آرایه موردنظر نامرتب باشد، ابتدا باید آن را مرتب کنید.

فرض کنید آرایه به صورت صعودی مرتب شده است. برای جست‌وجوی دودویی، ابتدا عنصر میانی (وسط) آرایه را به‌دست آورده و با کلید جست‌وجو مقایسه می‌کنیم. در این لحظه سه حالت ممکن است رخ دهد :

- عنصر میانی با کلید جست‌وجو مساوی است. در این حالت جست‌وجو موفق بوده و خاتمه می‌یابد و اندیس عنصر میانی به عنوان مکان عنصر مورد جست‌وجو برگردانده می‌شود.
- عنصر میانی از کلید جست‌وجو بزرگ‌تر است. بنابراین از تمام عناصر قبل از آن نیز بزرگ‌تر خواهد بود و می‌توان جست‌وجو را در نیمهٔ بالایی (سمت راست عنصر میانی) ادامه داد.
- عنصر میانی از کلید جست‌وجو کوچک‌تر است. بنابراین از تمام عناصر بعد از آن نیز کوچک‌تر خواهد بود و می‌توان جست‌وجو را در نیمهٔ پایینی (سمت چپ عنصر میانی) ادامه داد.

به مثال زیر توجه کنید :

در لیست ۲۴،۷،۸،۱۹ نیمه پایینی ۱۹ ۸ ↓ ۷ ۴ ۲ نیمه بالایی

۱ــ جست‌وجوی عدد ۷ : عنصر میانی مساوی کلید جست‌وجو است (۷=۷).

۲ــ جست‌وجوی عدد ۴ : عنصر میانی از کلید جست‌وجو بزرگتر است پس در نیمه بالایی به دنبال عدد می گردیم (۴ < ۷).

۳ــ جست‌وجوی عدد ۱۹ : عنصر میانی از کلید جست‌وجو کوچکتر است لذا در نیمه پایینی به دنبال عدد می‌گردیم. (۱۹ > ۷).

در این روش، جست‌وجو تا زمانی ادامه می‌یابد که کران پایین بزرگ‌تر از کران بالا باشد و اگر

چنین حالتی رخ دهد، می‌گوییم جست‌وجو ناموفق بوده است.

به عنوان مثال، فرض کنید می‌خواهیم محل نامی را در دفترچهٔ تلفن (یا واژه‌ای را در فرهنگ لغت) پیدا کنیم. واضح است که برای انجام چنین کاری از جست‌وجوی خطی استفاده نمی‌کنیم و به جای آن وسط دفترچه را باز کرده و تعیین می‌کنیم که جست‌وجو در کدام نیمهٔ دفترچه است. سپس نیمهٔ موردنظری را از وسط بازکرده و تعیین می‌کنیم که اسم موردنظر در کدام یک چهارم می‌باشد، این کار را روی یک چهارم مورد نظر نیز انجام داده و تعیین می‌کنیم که اسم در کدام یک هشتم می‌باشد و الی آخر. این کار را تا زمانی ادامه می‌دهیم که عنصر موردنظر را پیدا کنیم. این الگوریتم سریع‌تر از الگوریتم جست‌وجوی خطی است، زیرا تعداد عناصر در حال کم شدن است.

مثال ۱۵ــ۴

فرض کنید آرایه‌ای از اعداد مرتب به‌صورت زیر وجود دارد و می‌خواهیم از روش جست‌وجوی دودویی برای پیدا کردن محل یک عنصر استفاده کنیم.

List: 2,5,7,14,26

Search Key: 14

Beg = 0, End = 4, Mid = Int [(Beg+End)/2] = [(0+4)/2] =2

List(2) = 7

Beg کران پایین، End کران بالا و Mid وسط آرایه است. بعد از به‌دست آوردن عنصر میانی آرایه (یعنی عدد ۷) آن را با کلید جست‌وجو (عدد ۱۴) مقایسه می‌کنیم. چون ۱۴ از ۷ بزرگ‌تر است، پس از اعداد قبل از آن نیز (اعداد ۲ و ۵) بزرگ‌تر است. بنابراین جست‌وجو را باید در نیمهٔ دوم آرایه انجام دهیم. برای انجام این کار، کران پایین آرایه به‌صورت زیر تغییر می‌کند :

Beg = Mid + 1 = 2 + 1 = 3, End = 4, Mid = [(3+4)/2] = 3

List(3) = 14

با تغییر کران پایین، دوباره عنصر میانی لیست جدید را به‌دست می‌آوریم و با کلید جست‌وجو مقایسه می‌کنیم. در این حالت کلید جست‌وجو با عنصر میانی برابر است. بنابراین می‌توان گفت که جست‌وجو موفق بوده و مکان کلید در آرایه ۳ است.

تمرین : در مثال فوق، جست‌وجوی عدد ۳ را به صورت گام به گام انجام دهید.

این روش جست‌وجو فقط مخصوص لیست‌هایی است که از قبل مرتب شده باشند. در این روش به‌جای آن‌که عمل جست‌وجو و مقایسه از اولین عنصر لیست آغاز شود ابتدا عنصر وسط لیست با کلید مقایسه می‌گردد و درنتیجه سه حالت اتفاق می‌افتد :

۱ــ کلید مساوی با مقدار عنصر وسط لیست است.

۲ــ کلید بزرگ‌تر از مقدار عنصر وسط لیست است.

۳ــ کلید کوچک‌تر از مقدارعنصر وسط لیست است.

در حالت اول کار جست‌وجو تمام شده است.

ولی اگر لیست صعودی باشد در حالت دوم محال است که کلید در نیمهٔ بالایی لیست باشد. پس نیمهٔ پایین لیست را جست‌وجو می‌کنیم.

حالت سوّم مشابه حالت دوّم است با این تفاوت که کلید نمی‌تواند در نیمه پایینی لیست باشد و باید نیمه بالای لیست را جستجو کنیم.

در این روش با یک عمل مقایسه نصف عناصر لیست را کنار زده‌ایم. درواقع در یک لیست ۱۰۰،۰۰۰ عنصری با یک عمل مقایسه ۵۰،۰۰۰ عنصر لیست را کنار می‌گذاریم.

با مقایسهٔ بعدی مطمئن می‌شویم کلید در ۲۵،۰۰۰ عنصر دیگر لیست وجود ندارد. یعنی هر بار لیست نصف می‌شود مثلاً اگر لیستی ۸۰۰،۰۰۰ عنصر داشته باشد، حداکثر ۲۰ مقایسه کافی است تا کلید پیدا شود.

مثال ۱۶ــ۴

برنامه‌ای بنویسید که ۱۰ عدد تصادفی دو رقمی تولید کند سپس هر عددی را وارد کردیم موقعیت آن عدد را در لیست جست‌وجو کند و جواب بدهد. (جست‌وجوی دودویی)

شکل ۲۵ــ۴

```
Option Explicit

Private Sub cmdCreate _ Click()
        Dim inti As integer
        Dim intnum As Integer
        Lstnumber.Clear
         txtkey = " "
        '*** ۱۰≤intNum<۱۰۰ تولید ۱۰ عدد تصادفی دورقمی ***
             For inti =1 To 10
           Intnum = Int(Rnd * 90) + 10
           Lstnumber.AddItem intnum
       Next
End Sub
Private Sub cmdExit_Click()
        End
End Sub
Private Sub cmdsearch_Click()
        Dim intkey As Integer
        Dim intLow As Integer
        Dim intHigh As Integer
        Dim intMid As Integer
        Dim intPosition As Integer
        Dim blnfound As Integer
        lblPosition.Caption = " "
        intkey = Val (txtkey)
        blnfound = False
        intLow = 0:  intHigh = 9
```

```
    Do While intLow <= intHigh And Not blnfound
        intMid = (intLow + intHigh) \ 2
        If intkey < Val (Istnumber.List (intMid) Then
            intHigh = intMid − 1
        ElseIf intkey > Val (Istnumber.List (intMid)) Then
            intLow = intMid + 1
        EIse
        blnfound = True
        intPosition = intMid
        End If
    Loop
    If blnfound = True Then
            lblPosition = "Item = " & intPosition
    Else
            lblPosition = "Not Found"
    End If
End Sub

Private Sub From_Load()
    Randomize Timer
End Sub
```

۹ــ۴ــ آرایه‌های چندبعدی

آرایه‌هایی که تا اینجای فصل ایجاد کردیم، همان‌طورکه در شکل ۲ــ۴ مشاهده کردید، آرایه‌های یک‌بعدی (مجموعه‌ای یک‌سطری 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 به‌ندرت نیاز به ایجاد آرایه‌های سه‌بعدی و بیشتر خواهید داشت.

آرایهٔ دوبعدی را شبیه صفحهٔ بازی دوز در نظر بگیرید. (مجموعه‌ای از ستون‌ها و سطرها که از تداخل آن‌ها خانه‌هایی به‌وجود می‌آیند). هر خانه دارای موقعیت تعریف‌شده‌ای به‌صورت شمارهٔ سطر و شمارهٔ ستون است.

توجه داشته باشید که هر عنصر براساس مختصات سطر و ستون تعریف می‌شود. به‌عنوان مثال، عنصر iVar(0,0) دارای مقدار ۵ و عنصر iVar (2,2) دارای مقدار ۴۹ است (شکل ۲۶ــ۴).

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

```
Dim | Public | Private ArrayName (SubscriptOfRows, SubscriptOfCols) As Data  Type
```

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

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

شکل ۲۶ــ۴ــ آرایهٔ دوبعدی را به‌صورت مجموعه‌ای از ظرف‌ها که در ستون‌ها و سطرها سازماندهی شده‌اند، تصور کنید.

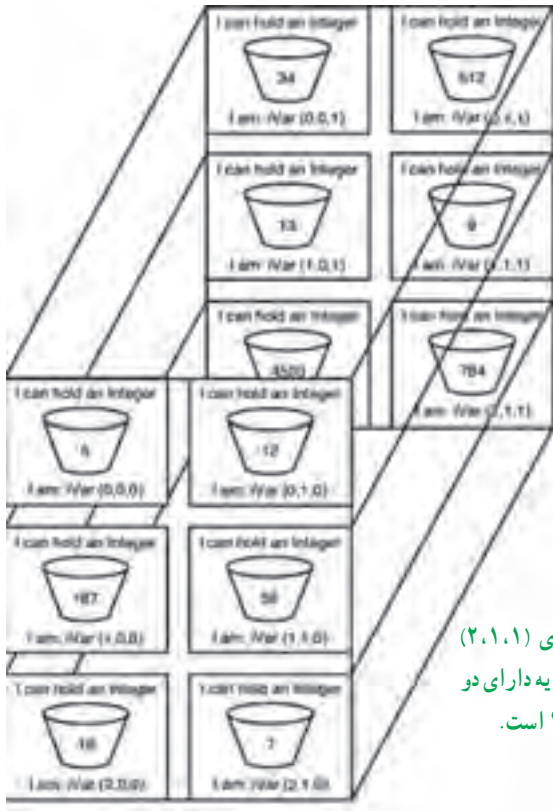
بنابراین، برای اعلان آرایهٔ شکل ۲۶ــ۴ می‌توان نوشت :

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

اگر یک آرایهٔ دوبعدی را به‌صورت مستطیل تصور کنید، می‌توانید آرایهٔ سه‌بعدی را به شکل یک مکعب مستطیل در نظر بگیرید. برای اعلان آرایهٔ سه‌بعدی از اعداد صحیح، دستور زیر را وارد کنید :

```
Dim iVar (2,1,1) As Integer
```

شکل ۲۷ــ۴ این آرایهٔ سه‌بعدی را نشان می‌دهد.



شکل ۲۷ــ۴ــ آرایهٔ سه‌بعدی (۲،۱،۱) دارای ۱۲ عنصر است. این آرایه دارای دو ستون و سه سطر به عمق ۲ است.

(تعداد صفحات عمق × تعداد سطر × تعداد ستون) = تعداد عناصر آرایهٔ سه‌بعدی

در آرایهٔ iVar مقدار تعیین‌شده برای هر عنصر که در شکل ۲۷ــ۴ نشان داده شده است، به‌صورت زیر خواهد بود :

```
iVar(0,0,0) = 5
iVar(1,0,0) = 187
```

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

مشابه آرایهٔ یک‌بعدی، می‌توان از کلید واژهٔ To برای اعلان محدودهٔ اندیس‌های هر بعد در آرایه‌های چندبعدی نیز استفاده کرد. پس دستور زیر :

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

یک آرایهٔ سه‌بعدی از مقادیر اعشاری خواهد بود که دارای ۵ سطر، ۶ ستون و ۳ صفحهٔ عمق خواهد بود.

همچنین می‌توان از کلید واژهٔ ReDim برای تغییر اندازهٔ آرایهٔ چندبعدی استفاده کرد. اگر از کلید واژهٔ Preserve استفاده کنید، فقط آخرین بعد آرایهٔ چندبعدی می‌تواند تغییر اندازه یابد و تعداد ابعاد نمی‌توانند تغییر کنند.

خودآزمایی

۱ــ ۱۰ دانش‌آموز و ۳ نمره از هر یک را از ورودی دریافت کنید، سپس نام هر دانش‌آموز را به همراه معدل سه نمرهٔ او نمایش دهید. در پایان نام دانش‌آموزانی که بالاترین و پایین‌ترین معدل را دارند اعلام کنید.

۲ــ برنامه‌ای بنویسید که عناصر یک ماتریس ۴×۴ را دریافت کند. سپس این ماتریس را روی فرم نشان دهد و با پیغامی اعلان کند که آیا عناصر قطر اصلی ماتریس با هم برابر هستند یا نه.

۳ــ برنامه‌ای بنویسید که عناصر یک ماتریس ۴×۴ را دریافت کند. سپس عددی را گرفته، آن را در ماتریس جست‌وجو کند و تعداد تکرار آن را نمایش دهد.

۴ــ برنامه‌ای بنویسید که عناصر یک ماتریس ۳×۳ را دریافت کند. سپس این ماتریس را نشان داده و اعلام کند بالا مثلثی است یا پایین مثلثی.

۵ــ برنامه‌ای بنویسید که نام و نمره ریاضی ۸ نفر را گرفته و اعلام کند رتبه دوم کیست.

فصل پنجم

کاربرد فرم‌های آماده

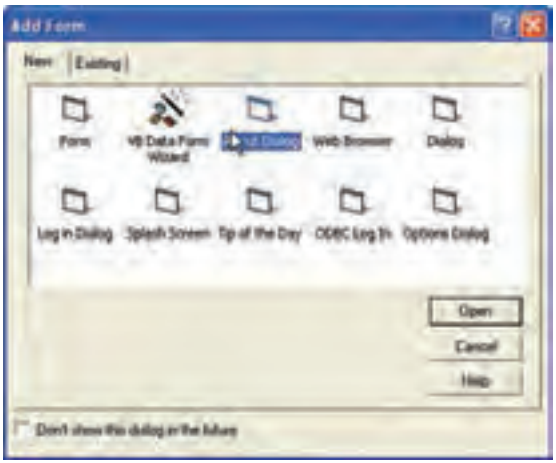
هدف‌های رفتاری: پس از آموزش این فصل، هنرجو می‌تواند:

- **دلیل استفاده از فرم‌های آماده را بیان کند؛**
- **از انواع مختلف فرم‌ها و کادرهای محاوره‌ای آماده مانند File، Color، Printer و غیره در برنامه‌های خود استفاده کند؛**
- **مفهوم رابط چند سندی را شرح دهد؛**
- **پروژه‌هایی ایجاد کند که از رابط چندسندی استفاده می‌کنند.**

۱ــ۵ــ 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 آمادهٔ ویژوال بیسیک، می‌توان برای دسترسی به داده‌ها، کادر About، صفحه‌های الگو و کادرهای ورود، استفاده کرد.

از منوی Project گزینهٔ Add Form را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود. در این کادر محاوره‌ای، انواع مختلف فرم‌های آماده وجود دارد. فرم موردنظر را انتخاب کنید (شکل ۱ــ۵).



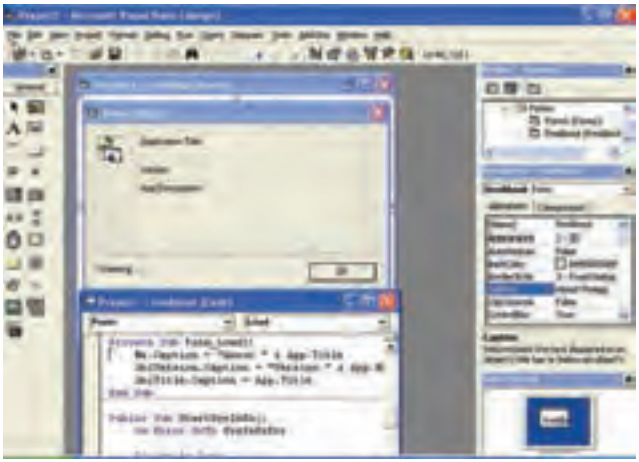
شکل ۱ــ۵ــ می‌توان فرم‌های آماده یا Form Wizard را در کادر محاوره‌ای Add Form انتخاب کرد.

در صورتی که می‌خواهید کادر محاوره‌ای About را به پروژه اضافه کنید، در کادر محاوره‌ای Add Form روی About Dialog کلیک کنید (شکل ۲ــ۵ یک فرم About Dialog را نشان می‌دهد). نه‌تنها ویژوال بیسیک، فرم را به پروژه اضافه می‌کند، بلکه بخشی از کدهایی را که وظایف اصلی فرم را دربر می‌گیرند نیز درج می‌کند. About Dialog کدی را ارایه می‌کند که نام برنامهٔ کاربردی و اطلاعات نسخه را همان‌طور که در کد زیر مشاهده می‌کنید گزارش می‌کند :

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

در این کد کنترل APP[1] به پروژه جاری و کنترل Me به فرم جاری اشاره می‌کند.

کادر محاوره‌ای About کل کد و فراخوانی API[2] ویندوز را، که برای گزارش اطلاعات سیستم کاربر موردنیاز است، ارایه می‌کند. (در صورتی که می‌خواهید کد اطلاعات سیستم را مرور کنید، فرم About Dialog را به پروژهٔ جدیدی اضافه کنید و کد تحت دکمهٔ System Info را مشاهده کنید.)



شکل ۲ــ۵ــ کادر محاوره‌ای About که به همراه ویژوال بیسیک ارایه می‌شود کدی را که نام برنامه و اطلاعات نسخه و به همین ترتیب اطلاعات سیستم کاربر را گزارش می‌کند، ارایه می‌دهد.

۱ــ در فصل ۸ توضیح آن آمده است.

۲ــ در برنامه‌سازی ۳ با انواع توابع API آشنا خواهید شد.

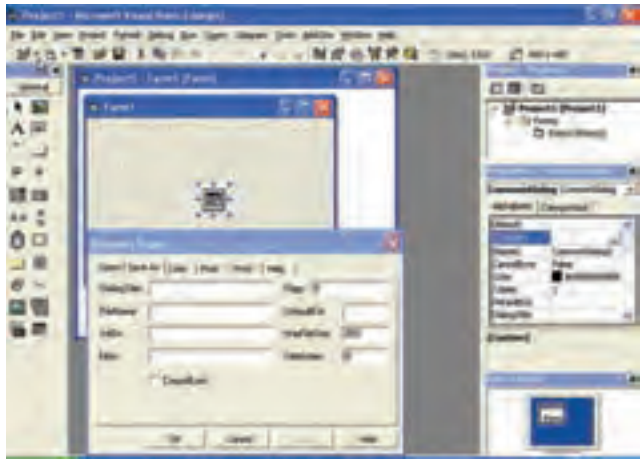
۲ــ۵ــ کنترل Common Dialog

بعضی مواقع ممکن است بخواهید برنامه‌ای بنویسید که کاربران بتوانند نام فایل را تعیین کنند، قلم‌ها و رنگ‌ها را انتخاب کنند و چاپگر را کنترل نمایند. اگرچه می‌توانید کادرهایی محاوره‌ای ایجاد کنید که این وظایف را مدیریت کنند ولی نیاز به انجام این کار نیست. ویژوال بیسیک، کنترل Common Dialog را ارایه می‌کند که به سادگی می‌توان به کمک آن کادرهای محاوره‌ای آماده را برای به‌دست آوردن اطلاعات کاربر، نمایش داد. این کادرهای محاوره‌ای برای کاربران آشنا هستند و همان‌هایی هستند که خود ویندوز به‌کار می‌برد. با استفاده از کنترل Common Dialog، به چهار کادر محاوره‌ای ویندوز دسترسی خواهید داشت :

- **File (Save, Open)**، به کاربر امکان انتخاب فایل برای باز شدن یا انتخاب نام فایل برای ذخیرهٔ اطلاعات را فراهم می‌کند.
- **Font**، امکان انتخاب قلم و تنظیم مشخصه‌های قلم موردنظر 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 ازکنترل Common Dialog، ابتدا باید آن را با انتخاب (MicrosoftCommon Dialog Control 6.0) از کادر محاوره‌ایComponents (انتخاب Components از منوی Project) به پروژه اضافه کنید. بعد از اضافه کردن کنترل Common Dialog به جعبه ابزار، روی کنترل کلیک و شبیه کنترل‌های دیگر، روی فرم ترسیم کنید. این کنترل، مانند یک نشانه روی فرم ظاهر می‌شود و در زمان اجرا، قابل رؤیت نیست. ولی هنگامی که کد، Common Dialog را فراخوانی کند، کادر محاوره‌ای خاصی ظاهر می‌شود.

در ادامه، هرکدام از کادرهای محاوره‌ای مربوط به این کنترل را شرح می‌دهیم. برای هر کادر محاوره‌ای، باید مشخصه‌های کنترل را از طریق پنجرهٔ Properties یا کادرمحاوره‌ای Property Pages تنظیم کنید. کادر محاوره‌ای Property Pages دسترسی ساده به مشخصه‌های موردنیاز برای هرنوع کادر محاوره‌ای را ارایه می‌کند (شکل ۳ــ۵). برای دسترسی به این کادر محاوره‌ای، در پنجرهٔ Properties روی سه نقطهٔ مقابل مشخصهٔ Custom مربوط به کنترل Common Dialog کلیک کنید.



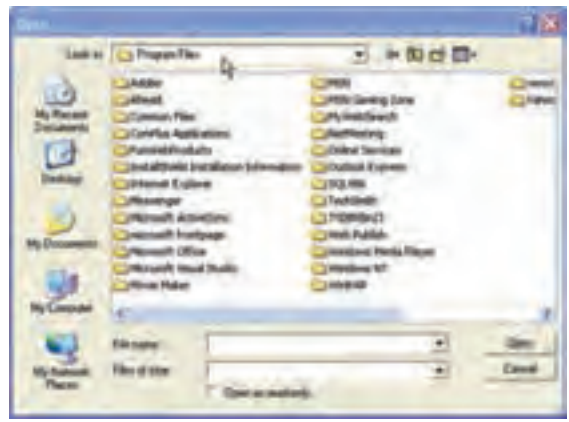
شکل ۳ــ۵ــ کنترل Common Dialog در زمان طراحی قابل رؤیت بوده و می‌تواند با استفاده از Property Pages پیکربندی شود.

۳ــ۵ــ بازیابی و ذخیرهٔ اطلاعات پرونده

کاربرد اصلی کنترل Common Dialog فراهم کردن امکان انتخاب نام پرونده به دو منظور بازکردن و ذخیره‌کردن پرونده است. حالت باز کردن پرونده، به کاربر امکان تعیین نام پرونده برای بازیابی و استفاده در برنامه را می‌دهد. حالت ذخیرهٔ پرونده به کاربر امکان می‌دهد تا نامی را برای ذخیرهٔ اطلاعات تحت یک پرونده، فراهم کند. شکل ۴ــ۵ کادر محاوره‌ای با مؤلفه‌های اصلی را نشان می‌دهد.

کادرهای محاوره‌ای Open و Save شبیه هم هستند. برای باز کردن یک پرونده موجود، از متد ShowOpen کنترل Common Dialog استفاده کنید. (این متد، کادر محاوره‌ای نشان داده شده در شکل ۴ــ۵ را نشان می‌دهد.) از این متد، به صورت زیر استفاده کنید :

```
CommonDlg1.ShowOpen
```



شکل ۴ــ۵ــ Common Dialog به کادرهای محاوره‌ای پرونده‌ای ویندوز دسترسی دارد.

برای بازکردن کادر محاوره‌ای Save As باید از متد ShowSave استفاده کرد.

برای محدود کردن نوع پرونده‌ها مثل پرونده‌های متنی یا سند در کادرهای محاوره‌ای پرونده‌ای، از مشخصهٔ Filter کنترل Common Dialog استفاده کنید. این مشخصه را می‌توان در زمان طراحی از طریق کادر محاوره‌ای Property 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. درصورتی که این کار را انجام دهید، ممکن است لیست فایل مورد نظر را به‌دست نیاورید.

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

اگر از دستور انتساب فوق استفاده کنید، باید فیلتر را داخل زوج کوتیشن (" ") قرار دهید ولی در صورت استفاده از کادر محاوره‌ایProperty Pages نیازی به این کار نیست.

می‌توان چندین فیلتر را با هم به‌کار برد و بین این فیلترهای مختلف، باید ازعلامت پایپ استفاده کرد. مثال زیر را درنظر بگیرید :

Commondlg1. Filter ="Text Documents | *.txt | All Files (*.*) | *.*"

کادر ترکیبی File of Type در شکل ۴ــ۵ کد فوق را که به کنترل Common Dialog اعمال شده است نشان می‌دهد. بعد از تنظیم فیلتر، از مشخصهٔ Filename کنترل Common Dialog برای بازیابی نام فایلی که کاربر انتخاب کرده است، استفاده کنید :

MyFileName$ = Commond1g1.FileName

مثال ۱ــ۵

می‌خواهیم از کادر محاوره‌ای File برای انتخاب و باز کردن یک پرونده استفاده کنیم. مراحل کار به صورت زیر خواهد بود :

۱ــ پروژهٔ جدیدی را شروع کنید.

۲ــ کنترل Common Dialog را همان‌طور که قبلاًشرح داده شد، به جعبه ابزار اضافه کنید.

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

۴ــ کنترل برچسب را در بالای کنترل دکمهٔ فرمان قرار دهید. هر دو کنترل را به گوشهٔ چپ بالای فرم، منتقل کنید.

۵ ــ روی دکمهٔ فرمان، دوبار کلیک کنید تا روال رویداد Commandl_Click ایجاد شود.

۶ــ کد زیر را به این روال اضافه کنید :

```
01 CommonDialogl.Filter ="All Files (*.*) | *.*"
02 CommonDialogl.ShowOpen
03 If CommonDialogl. FileName<>"" Then
04    LabeLl.Caption = CommonDialogl.FileName
05 Else
06    LabeL1. Caption = "No file selected"
07 End If
```

عملکرد این کد به صورت زیر است :

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

خط ۲، کادر محاوره‌ای Open را نمایش می‌دهد. کاربران می‌توانند پرونده‌ای را انتخاب کرده و رویOpen کلیک کنند یا هیچ پرونده‌ای را انتخاب نکرده و روی Cancel کلیک کنند. براساس این انتخاب‌ها، ویژوال بیسیک مقداری را در مشخصهٔ FileName کنترل Common Dialog قرار می‌دهد.

خطوط ۳ تا ۷، مقدار مشخصهٔ FileName را بررسی می‌کنند و در صورت انتخاب پرونده، نام آن در مشخصهٔ Caption برچسب قرار می‌گیرد و در غیر این‌صورت، پیغام No File Selected نوشته می‌شود.

۴ــ۵ــ انتخاب اطلاعات قلم

تنظیم کنترل Common Dialog برای نمایش کادر محاوره‌ای Font شبیه نمایش کادرهای محاوره‌ای پرونده‌ای است.

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

جدول ۱ــ۵ــ مقادیر مشخصه Flags در کادر محاوره‌ای Font

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

شکل ۵ــ۵ــ کادر محاوره‌ای Font امکان انتخاب قلم‌ها را به کاربر می‌دهد.

نکته

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

می‌توان مقدار مشخصهٔ Flags را از محیط طراحی با استفاده از کادر محاوره‌ای Property Pages یا از درون برنامه با دستور انتساب تعیین کرد. بعد از تعیین مشخصهٔ 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 انتخاب‌شده در مشخصه‌های کنترل قرار داده می‌شود. جدول ۲ــ۵ مشخصه‌های کنترل و صفات قلم را نشان می‌دهد.

جدول ۲ــ۵ــ مشخصه‌های کنترل که صفات قلم را ذخیره می‌کنند

مشخصه	صفت
FontName	نام قلم
FontSize	اندازهٔ قلم برحسب پوینت
FontBold	آیا ضخیم‌بودن قلم انتخاب شده است؟
FontItalic	آیا کج‌بودن قلم انتخاب شده است؟
FontUnderline	آیا قلم زیرخط‌دار شده است؟
FontStrikethru	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 به‌دست آمده و برای تغییر قلم‌ها در کادر متن txtsample استفاده می‌شود.

```
Commondlg1 is the name of a common dialog
Commondlg1.ShowFont
txtSample.FontName = Commondlgl.FontName
txtSample.FontSize = Commondlgl.FontSize
txtSample.FontBold = Commondlgl.FontBold
txtSample.FontItalic = Commondlgl.FontItalic
txtSample.FontUnderline = Commondlg1.FontUnderline
txtSample.FontStrikethru = Commondlgl.FontStrikethru
```

تمرین ۱ــ۵: برنامه‌ای بنویسید که دارای کادر متن و دکمهٔ انتخاب فونت باشد و کاربر بتواند نوع و اندازه و Style فونت کادر متن را تعیین کند.

۵ــ۵ــ انتخاب رنگ‌ها

کادر محاوره‌ای Color امکان انتخاب رنگ‌هایی برای رویه یا زمینهٔ فرم‌ها یا کنترل‌ها فراهم می‌کند (شکل۶ــ۵). کاربران می‌توانند یک رنگ استاندارد را انتخاب کرده یا رنگ سفارشی را ایجاد و انتخاب کنند.

شکل ۶ــ۵ــ کادر محاوره‌ای Color امکان انتخاب رنگی برای استفاده در برنامه را به کاربران می‌دهد.

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

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

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

```
CommonDlg1.Flags = cdLccRGBInit
CommonDlg1.ShowColor
```

```
MyForm.BackColor = CommonDlg1.Color
```

تمرین ۲ــ۵: به مثال ۱ـ۵ دکمه‌ای برای انتخاب رنگ متن در TextBox اضافه کنید.

۶ــ۵ــ تنظیم گزینه‌های چاپگر

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

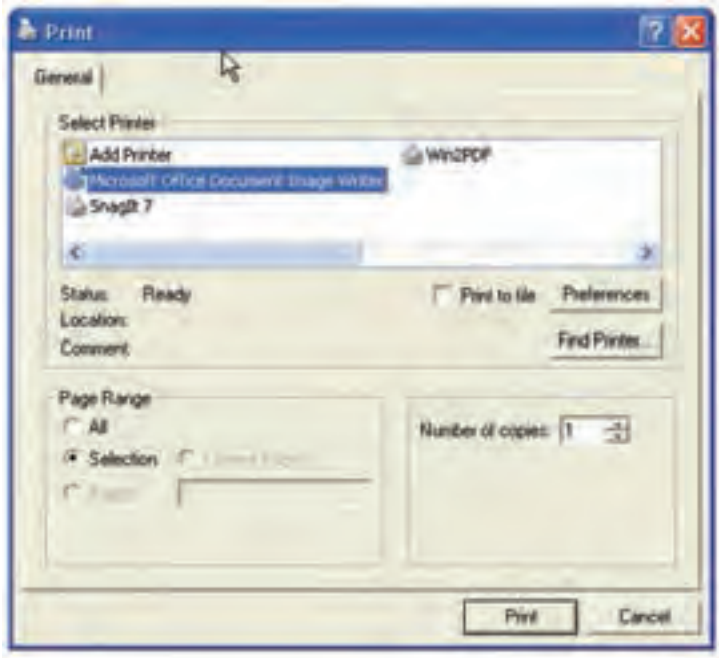
برای اجرای کادر محاوره‌ای Print، باید متد ShowPrinter را اجرا کرد (شکل ۷ــ۵). کاربران می‌توانند در این کادر محاوره‌ای چاپگر را از لیست Name انتخاب کنند که شامل تمام چاپگرهای نصب شده در ویندوز است. زیر لیست Name خط Status قرار دارد که وضعیت فعلی چاپگر انتخاب شده را بیان می‌کند.

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

با استفاده از کادر محاوره‌ای Print تنها می‌توان چاپگر و نحوهٔ چاپ را انتخاب کرد ولی داده‌ها برای چاپ به چاپگر ارسال نمی‌شود برای ارسال داده‌ها به چاپگر باید از شیء Printer[1] استفاده کرد.

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 همدیگرند. اغلب برنامه‌نویسان ترجیح می‌دهند که از برنامه‌های کاربردی MDI استفاده کنند. برنامه‌های NotePad و Pbrush نمونه‌هایی از برنامه‌های تک‌سندی هستند.

۱ــ شیء Printer و متدهای آن در برنامه‌سازی ۳ شرح داده شده است.



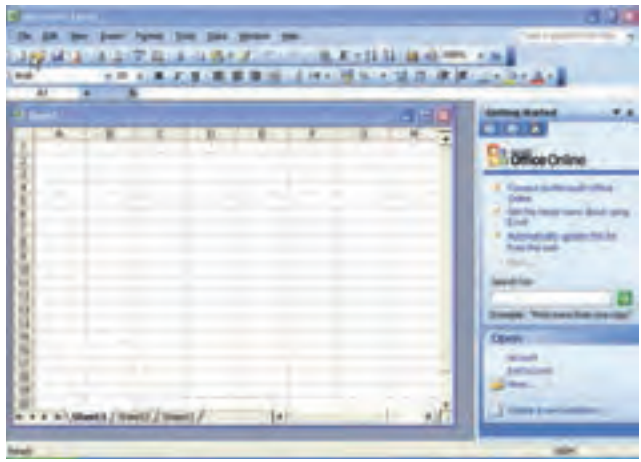
شکل ۷ــ۵ــ کادر محاوره‌ای **print** امکان انتخاب چاپگر و تنظیمات آن را به کاربر می‌دهد.

کد زیر چگونگی استفاده از کادر محاوره‌ای print را نشان می‌دهد. در این کد تنظیمات انجام شده توسط کاربر روی برچسب LblPrint نمایش داده می‌شود.

```
Dim BeginPage, EndPage, NumCopies, I, strs
CommonDlg1. ShowPrinter 'Display Print dialog box
'Get user-selected values
BeginPage = CommonDlg1. FromPage
EndPage = CommonDlg1. Topage
NumCopies = CommonDlg1. Copies
'display user-selected values
strs = "begin page: "& BeginPage
strs = strs & "end page: " & EndPage
strs = strs & "num copies: " & NumCopies
LblPrint. Caption = strs
```

۷ــ۵ــ ایجاد برنامهٔ کاربردی چندسندی ساده

MDI سرنام کلمات Multiple Document Interface (رابط چند سندی) است. یک برنامهٔ MDI (چندسندی)، برنامه‌ای است که می‌تواند هم‌زمان چندین سند باز داشته باشد و روی آن‌ها کار کند. Microsoft Word و Microsoft Excel مثال‌هایی از برنامهٔ کاربردی چندسندی هستند.



شکل ۸ــ۵ــ Microsoft Excel مثالی از یک برنامهٔ کاربردی MDI است.

ویژوال بیسیک یک شیء MDI به نام MDIForm تعریف کرده است. یک برنامهٔ 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 به‌وسیلهٔ 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 پنجرهٔ پدر در محلی از منوی پدر، ظاهر می‌شود.

نکته

کنترل‌های محدودی مانند Common Dialog، Picture Box و Timer می‌توانند به‌طور مستقیم روی فرم MDI قرار بگیرند. بنابراین، برای قرار دادن کنترل‌های دیگر روی فرم MDI باید یک کنترل Picture Box روی فرم قرار داد و سپس کنترل‌های موردنظر را داخل آن استفاده کرد.

پژوهش: بررسی کنید که آیا کنترل‌های دیگری وجود دارند که بتوان به‌طور مستقیم روی فرم MDI قرار داد؟

مثال ۲ــ۵

۱ــ پروژهٔ جدیدی را شروع کنید و نام آن را SimplMDI.VBP قرار دهید.

۲ــ فرم پیش‌فرض پروژه را به frmChild تغییر نام دهید. پرونده فرم را با همین نام ذخیره کنید.

۳ــ ازمنوی Project گزینهٔ Add MDI Form را انتخاب کنید تا یک فرم MDI به پروژه اضافه شود. فرم MDI را به mdiMain تغییر نام دهید و با همین نام ذخیره کنید.

۴ــ مشخصهٔ MDIchild فرم frmChild را با true مقداردهی کنید تا به‌عنوان فرم فرزند mdiMain محسوب شود.

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

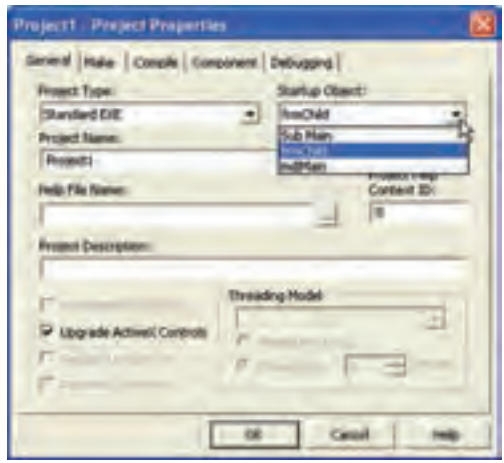
۶ــ مانند شکل ۹ــ۵ منویی را برای فرم mdiMain ایجاد کنید. مقادیر مشخصهٔ Name و Caption منو و گزینه‌های منو را مطابق جدول ۳ــ۵ تعیین کنید. مطمئن شوید که کادر علامت WindowList برای mnuWindow انتخاب شده باشد.

۷ــ از منوی Project گزینهٔ SimplMDI Properties را انتخاب کنید. در صفحهٔ General کادر محاوره‌ای Project Properties، از لیست بازشوی Startup Object گزینهٔ mdiMain را انتخاب کنید (شکل ۱۰ــ۵).

شکل ۹ ــ۵ــ منوهایی را با استفاده از Menu Editor برای یک برنامهٔ کاربردی MDI ایجاد کنید.

جدول ۳ــ۵ــ مشخصات منوی SimplMDI

Cation	Name	Indent
&File	mnuFile	0
&Add	itmAdd	1
&Exit	itmExit	1
&Window	mnuWindow	0
&CasCade	itmCasCade	1
&Tile	itmTile	1



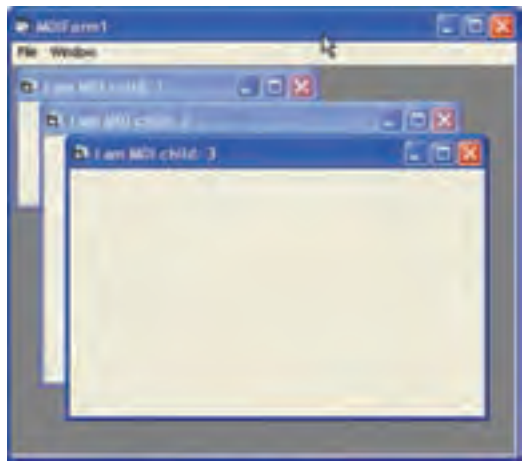
شکل ۱۰ــ۵ــ فرم MDI را به‌عنوان آغازین، تعیین کنید.

۸ ــ کد زیر را برای گزینه‌های منوی برنامهٔ کاربردی بنویسید :

```
Private Sub itmExit–Click()
    End
End Sub
Private Sub itmAdd_Click()
Dim NewForm As New frmChild
Dim FormNum As Integer
```

ایجاد فرم جدید به‌صورت پویا :

```
Load NewForm
FormNum = Forms.Count-1                    : Child تعیین تعداد فرم‌های
NewForm.Caption = "I am MDI child: " + CStr(FormNum)
End Sub
Private Sub itmCasCade_Click()
    mdiMain. Arrange  vbCasCade
End Sub
Private Sub itmTile_Clike()
mdiMain. Arrange vbTile Horizontal
EndSub
```



شکل ۱۱ــ۵ــ پروژهٔ SimplMDI فرم‌های فرزند داخل فرم MDI را نشان می‌دهد.

۹ــ برنامه را اجرا کنید.

روال رویداد itmAdd_Click() به‌طور پویا فرم جدیدی را با استفاده از عملگر New ایجاد می‌کند. Caption فرم جدید دارای یک شماره است (FormNum) که به انتهای رشتهٔ مربوطه اضافه می‌شود تا نشان دهد که چندمین فرم فرزند است. این شماره ازمشخصهٔ Forms.Count به‌دست می‌آید و برای این که تعداد فرم‌های فرزند محاسبه شوند، فرم MDI باید از شمارش، کسر شود. به همین دلیل، شمارهٔ فرم از تفریق تعداد فرم‌ها و عدد ۱ به‌دست می‌آید.

در این برنامهٔ کاربردی، از متد Arrange شیء MDIForm استفاده شده است. این متد، به‌طور خودکار پنجره‌های فرزند را داخل پنجرهٔ MDI قرار می‌دهد. متد Arrange دارای یک آرگومان است که می‌تواند یکی از مقادیر جدول ۴ــ۵ را داشته باشد.

جدول ۴ــ۵ــ تنظیمات آرایش پنجره‌ها

ثابت	مقدار	شرح
vbCasCade	0	تمام فرم‌های فرزند MDI را که به حداقل تبدیل نشده‌اند پشت سرهم قرار می‌دهد.
vbTileHorizontal	1	تمام فرم‌های فرزند MDI را که به حداقل تبدیل نشده‌اند به‌صورت کاشی‌های افقی می‌چیند.
vbTileVertical	2	تمام فرم‌های فرزند MDI را که به حداقل تبدیل نشده‌اند به صورت کاشی‌های عمودی می‌چیند.
vbArrangeIcons	3	آیکن‌های مربوط به MDI حداقل شده را مرتب می‌کند.

۸ــ۵ــ مشخصه‌های Appearance

فرم‌های سه‌بعدی برای کاربران مناسب‌تر است. با تغییر مشخصهٔ Appearance فرم MDI به 1، می‌توان فرم‌های سه‌بعدی ایجاد کرد. اگر مقدار این مشخصه 0 باشد فرم به‌صورت تخت ظاهر می‌شود (0–Flat، 1–3D)

مشخصهٔ AutoShowChildren : اگر مشخصهٔ AutoShowChildren دارای مقدار false باشد و 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.

به‌طور پیش‌فرض AutoShowChildren دارای مقدار True است و به همین دلیل 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.

نکته

اگر فرمی دارید که داده‌ها را در درون خودش بارگذاری می‌کند، مشخصهٔ

AutoShowChildren را با False مقداردهی کنید و از متد Show استفاده کنید. نخست داده‌ها را در فرم بارگذاری کنید و مطمئن شوید که همهٔ 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.

خودآزمایی

۱ــ فرمی مانند شکل روبه‌رو طراحی کنید، طوری که با انتخاب هر فرم از منوی Form ، فرم مربوطه باز شود و با انتخاب هر رنگی از منوی Color فرم جاری تغییر رنگ یابد و منوی Exit نیز سبب خروج از برنامه شود.

شکل ۱۲ــ۵

۲ــ فرمی مانند شکل روبه‌رو طراحی کنید به نحوی که با کلیک روی دکمهٔ Font بتوان خصوصیات ظاهری قلم برچسب، و با انتخاب Color بتوان رنگ برچسب را از طریق کادرهای محاوره‌ای مربوطه تغییر داد.

شکل ۱۳ــ۵

۳ــ فرمی مانند شکل روبه‌رو طراحی کنید به نحوی که با انتخاب اندازه و حالت و نام قلم، نوشتهٔ روی برچسب تغییر کند.

شکل ۱۴ــ۵

فصل ششم

زمان و تاریخ

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

- **در برنامه‌های خود از عنصر زمان استفاده کند؛**
- **برنامه‌هایی بنویسد که تاریخ و ساعت را به‌طور کامل مدیریت کنند؛**
- **انواع قالب‌ها را توضیح دهد و به‌کار گیرد.**

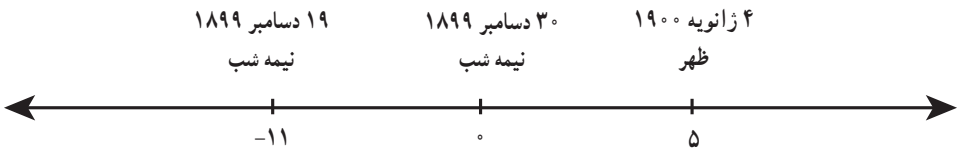
۱ــ۶ــ آشنایی با Serial Time

ویژوال بیسیک از نوع دادهٔ Date استفاده می‌کند که روز را به‌عنوان واحد اصلی زمان به‌کار می‌برد. بنابراین، یک ساعت، $\frac{۱}{۲۴}$ و یک ثانیه $\frac{۱}{۸۶۴۰۰}$ روز است. یک هفته را به‌صورت ۷ نمایش می‌دهیم، زیرا یک هفته شامل هفت روز است. نوع دادهٔ Date تاریخ را مطابق با قالب زمان کامپیوتر، نمایش می‌دهد؛ یعنی براساس قالب ۱۲ ساعتی یا ۲۴ ساعتی.

در تقویم میلادی اولین روز، اول ژانویهٔ سال است ولی ویژوال بیسیک، اولین روز را ۳۱ دسامبر[۱] 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 می‌شوند. به‌عنوان مثال، ۴ جولای سال ۱۷۷۶، روز ۴۵۱۰۳- است یعنی ۴۵۱۰۳ روز قبل از ۳۰ دسامبر ۱۸۹۹.

به محور زیر توجه کنید :



۱ــ ماه قبل از ژانویه

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

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

نوع دادهٔ Date می‌تواند به هر نوع دادهٔ دیگری تبدیل شود. به عنوان مثال، P.M May 15,1998 2 به صورت یک عدد اعشاری 35390.58333 از نوع Double خواهد بود. هر چیزی که در سمت چپ نقطهٔ اعشار است، نمایانگر روز بوده و هر چیزی که در سمت راست نقطهٔ 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 به‌کار روند. به‌عنوان مثال، 0.12345 زمان 2: 57: 46 A.m را نشان می‌دهد[۱].

۲ــ۶ــ آشنایی با کنترل Timer

در ویژوال بیسیک، کنترل Timer امکان پی‌گیری زمان را فراهم می‌کند. فرض کنید که Timer ساعتی است که یک رویداد قابل برنامه نویسی را در یک فاصلهٔ زمانی که تعیین کرده‌اید اجرا می‌کند (شکل ۱ــ۶). رویداد Timer فراخوانی شده و روال رویدادی که برای TimerName_Timer() برنامه‌نویسی کرده‌اید اجرا می‌شود. Timer Name مقدار مشخصهٔ Name کنترل Timer است.

شکل ۱ــ۶ــ کنترل Timer در زمان اجرا قابل رؤیت نخواهد بود.

۱ــ هر روز ۲۴ ساعت و هر ساعت ۶۰ دقیقه و هر دقیقه ۶۰ ثانیه است.

۰/۱۲۳۴۵ × ۲۴ = ۲/۹۶۲۸

۰/۹۶۲۸ × ۶۰ = ۵۷/۷۶۸ ⇒ ۲: ۵۷: ۴۶ Am

۰/۷۶۸ × ۶۰ = ۴۶/۰۸

می‌توان فاصلهٔ زمانی را که طی آن رویداد Timer اجرا می‌شود با تعیین مقداری برای مشخصهٔ Interval کنترل Timer تنظیم کرد. واحد اندازه‌گیری مشخصهٔ Interval میلی‌ثانیه است، بنابراین اگر می‌خواهید رویداد Timer در نیم ثانیه اجرا شود، کد زیر را بنویسید :

```
Timerl.Interval = 500
```

نکته

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

فاصلهٔ زمانی می‌تواند خیلی کوچک مثل هزارم ثانیه، و یا خیلی بزرگ باشد. کوتاه‌ترین فاصلهٔ 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 ۱ــ ۶).

جدول ۱ــ ۶ــ مشخصه‌های کنترل Timer

مشخصه	شرح
Name	نام کنترل. اگر یک کنترل Timer داشته باشید، پیش‌فرض Timerl است و الی آخر.
Enabled	کنترل Timer را فعال یا غیر فعال می‌کند. پیش‌فرض، True یا On است.
Index	محل کنترل Timer را هنگامی که به‌عنوان عنصری در آرایه‌ای از کنترل‌های Timerباشد تعیین می کند.
Interval	فاصلهٔ زمانی را که رویداد Timer اجرا خواهد شد تعیین می‌کند (برحسب میلی‌ثانیه).
Left	موقعیت گوشهٔ سمت چپ کنترل Timer را تعیین می‌کند.
Tag	شبیه یک متغیر درونی در کنترل است که داده‌های موردنیاز را ذخیره کرده و تا پایان برنامه به صورت پایدار نگه‌داری می‌کند.
Top	موقعیت گوشهٔ بالای کنترل Timer را تعیین می‌کند.

نکته

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

مثال ۱ــ۶

فرمی طراحی کنید که دارای یک برچسب باشد و با اجرای برنامه این برچسب از سمت چپ به سمت راست فرم حرکت کند و با رسیدن به انتهای سمت راست ازحرکت بایستد.

برای این برنامه نیاز به یک Label و یک Timer داریم.

```
Private Sub Form_Load()
    Timer1.Interval = 300
    Label1.Caption = "Move"
End Sub
Private Sub Timer1_Timer()
    If Label1. Left<Form1.Width–Label1. Width Then
        Label1. Left = Label1. Left +100
    Else
        Timer1. Enabled = False   متوقف‌کردن حرکت برچسب در انتهای سمت راست
    End If
End Sub
```

تمرین: در مثال ۱ـ۶، دو دکمهٔ فرمان به نام Start و End قرار دهید که با کلیک روی Start حرکت شروع و با کلیک روی End خاتمه یابد.

۳ــ۶ــ کاربرد توابع Date، Time و Now

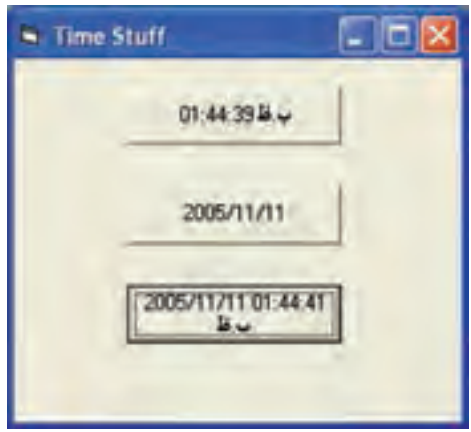
اگرچه می‌توان برای رویداد Timer برنامه‌ای نوشت، ولی نمی‌توان زمان اجرا را با این کنترل تنظیم کرد. برای این که Timer زمان را گزارش کند باید ساعت سیستم را بررسی کند.

از تابع Time برای به‌دست آوردن زمان سیستم و از تابع Date برای به‌دست آوردن تاریخ سیستم استفاده کنید. در صورتی که می‌خواهید هم‌زمان و هم تاریخ را از ساعت سیستم به‌دست آورید، از تابع Now استفاده کنید.

مثال ۲ــ۶

برای مشاهدهٔ چگونگی عملکرد این توابع، پروژه‌ای را ایجاد کرده و سه دکمهٔ 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 ۲ــ۴ داده‌هایی را که هر تابع برمی‌گرداند نشان می‌دهد.

```
Private Sub cmdTime_Click()
        cmdTime.Caption = CStr(Time)
End Sub
Private Sub cmdDate_Click()
        cmdDate.Caption = CStr(Date)
End Sub
Private Sub cmdNow_Click()
        cmdNow.Caption = CStr(Now)
End Sub
```



شکل ۲ــ۶ــ توابع Time، Date و Now یک نوع دادهٔ Variant(Date) را برمی‌گردانند که با تابع CStr() به رشته تبدیل می‌شوند.

مثال ۳ــ۶

استفاده از Timer برای ایجاد یک برنامهٔ 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 نوار ابزار ظاهر می‌شود. مراحل انجام کار به صورت زیر است :

۱ــ پروژهٔ Standard EXE جدیدی را شروع کنید.

۲ــ یک کنترل label و یک کنترل Timer روی فرم قرار دهید (کنترل Timer را می‌توانید در هر جایی قرار دهید، زیرا قابل رؤیت نخواهد بود.)

۳ــ نام فرم را frmClock و نام برچسب را lblTimeDisplay قرار دهید. نام کنترل Timer را تغییر ندهید (همان نام پیش‌فرض Timerl را حفظ کنید).

۴ــ مشخصهٔ BorderStyle فرم را به ۱–Fixed single تغییر دهید. مشخصهٔ MinButton فرم را True قرار دهید.

۵ــ کد زیر را برای روال رویداد ()Form_Load بنویسید.

```
Private Sub Form_Load()
    lblTimeDisplay.Top = frmClock.ScaleTop
    lblTimeDisplay.Left = frmClock.ScaleLeft
    lblTimeDisplay.Width = frmClock.ScaleWidth
    lblTime Display.Height = frmClock. ScaleHeight
End Sub
```

خطوط ۲ تا ۵ برای بزرگ کردن برچسب به اندازهٔ فرم است.

برای تعیین محل و ابعاد کنترل‌ها از مشخصه‌های Top، left، width و Height استفاده می‌کنیم. می‌توان علاوه‌بر چهار مشخصه فوق از مشخصات ScaleTop، Scaleleft، ScaleWidth و ScaleHeight استفاده کرد. Scalewidth و ScaleHeight به ترتیب پهنا و ارتفاع کنترل را تقسیم‌بندی می‌کنند. برای مثال Scalewidth = 100 پهنای کنترل را به ۱۰۰ قسمت مساوی تقسیم می‌کند و واحد اندازه‌گیری افقی را مشخص می‌نماید به همین ترتیب ScaleHeight=50 ارتفاع کنترل را به ۵۰ قسمت مساوی تقسیم کرده و واحد اندازه‌گیری عمودی را مشخص می‌کند.

از چهار مشخصهٔ ScaleTop، Scaleleft، ScaleWidth و ScaleHeight در مواردی استفاده

می‌کنیم که بخواهیم شیء همیشه در موقعیت ثابتی از فرم قرار داشته باشد. در کد (Form–load) بالا می‌خواهیم اندازه برچسب همیشه به اندازهٔ فرم باشد یعنی با تغییر اندازهٔ فرم، اندازه برچسب هم تغییر کند.

برای مشخص کردن واحد اندازه‌گیری ابعاد کنترل از خصوصیت ScaleMode استفاده می‌کنیم که مقادیر آن در جدول ۲ــ۶ آمده است.

جدول ۲ــ۶

ثابت‌ها	مقادیر	شرح
vbUser	۰	هرگاه حداقل یکی از مشخصات ScaleTop، ScaleWidth، ScaleHeight و ScaleLeft توسط برنامه‌نویس مقداردهی شود.
vbTwips	۱	Twip (پیش‌فرض) هر اینچ ۱۴۴۰ توئیپ و هر سانتیمتر ۵۶۷ توئیپ است.
vbPoints	۲	Point هر اینچ ۷۲ پوینت است.
vbPixels	۳	Pixel
vbCharacters	۴	Character (هر واحد افقی ۱۲۰ twips و هر واحد عمودی ۲۴۰ twips است.)
vbInches	۵	اینچ
vbMillimeters	۶	میلی‌متر
vbCentimeters	۷	سانتی‌متر

مقادیر ScaleTop و Scaleleft مساوی صفر است جز حالتی که ScaleMode مساوی صفر (vbUser) باشد.

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

۷ــ کد مربوط به روال Timerl_Timer() را به صورت زیر بنویسید.

```
Private Sub Timerl_Timer()
    If frmClock.WindowState = vbNormal Then
        lblTimeDisplay. Caption = CStr(Time)
        frmClock.Caption = Format(Date," Long Date")
```

```
    Else
        frmClock.Caption = CStr(Time)
    End If
End Sub
```

۸ ــ برنامه را ذخیره کرده و آن را اجرا کنید (شکل ۳ــ۶).

شکل ۳ــ۶ــ تنظیم مشخصهٔ MaxButton با مقدار False دکمهٔ به حداکثررسانی را غیرفعال می‌کند.

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

فرم‌ها دارای سه مشخصهٔ MinButton، MaxButton و ControlBox هستند که روی سه دکمهٔ سمت راست نوار عنوان تأثیر می‌گذارند. MinButton و MaxButton برای فعال و غیرفعال کردن دکمه‌های Minimize و Maximize مورد استفاده قرار می‌گیرند. این مشخصه‌ها فقط در زمان طراحی و هنگامی که مشخصهٔ BorderStyle فرم دارای مقدار 1–Fixed Single یا 2–Sizeable باشد، قابل دسترسی هستند.

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

نکته

هنگام استفاده از مشخصهٔ Window State دقت کنید. ممکن است به‌طور اتفاقی روی فرمی که Border Style آن Fixed Single است، مشخصهٔ Window State را با Maximized مقداردهی کنید. نتیجهٔ 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 برنامهٔ ساعت، مقدار مشخصهٔ Window State فرم را برای تعیین به حداقل رسانی فرم،بررسی می‌کند. مشخصهٔ Window State دارای سه مقدار است :

0– Normal(vbNormal)

1– Minimized (vbMinimized)

2– Maximized (vbMaximized)

می‌توان مقدار این مشخصه را خواند و تعیین کرد که آیا پنجره در اندازهٔ عادی، تمام صفحه یا در نوار وظیفه است. همچنین می‌توان مشخصهٔ Window State 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.

۴ــ۶ــ کاربرد تابع Format()

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

MyString = Format (Expression[,Format_String [,First Day Of Week [,First Week Of Year]]])

- My String نام متغیر رشته‌ای است که خروجی تابع در آن قرار می‌گیرد.
- Format نام تابع است.
- Expression عبارتی است که می‌خواهیم این تابع آن را به قالب موردنظر تبدیل کند.

Format_String الگوی رشته‌ای است که به تابع بیان می‌کند که می‌خواهد رشتهٔ خروجی به چه صورتی ظاهر شود.

- FirstDayOfWeek یک ثابت اختیاری است که اولین روز هفته را تعیین می‌کند. پیش‌فرض، Sunday است ولی اگر می‌خواهید روز شنبه باشد، باید آن را بنویسید (Saturday).
- First Week Of Year یک عبارت ثابت اختیاری است که اولین هفتهٔ سال را تعیین می‌کند. به‌طور پیش فرض، اول ژانویه، اولین هفتهٔ سال است.

کلید کار کردن با تابع Format()، آشنایی با پارامتر Format_String است. جدول ۲ــ۶ چگونگی استفاده از تنظیمات این تابع را برای تغییر ظاهر رشته‌های تاریخ و زمان ارایه می‌کند.

نکته

باید پارامتر Format_String را بین زوج کوتیشن " " قرار دهید، زیرا تابع این پارامتر را به صورت رشته به‌کار می‌برد. به‌عنوان مثال، دستور My String$ =Format(.50،percent) خطایی را تولید خواهد کرد ولی این دستور به‌صورت My String$ =Format(.50،"percent") صحیح خواهد بود.

جدول ۲ــ۶ــ کاربرد تابع Format () برای زمان و تاریخ

Format – String	مثال	نتیجه
“Long Date”	Format(36000, “Long Date”)	Friday, July24,1998
“Medium Date”	Format(36000, “Medium Date”)	24 – Jul–98
“Short Date”	Format(36000, “Short Date”)	7/24/98
“Long Time”	Format(0.874, “Long Time”)	8: 58: 34 p.m.
“Medium Time”	Format(0.874, “Medium Time”)	8: 58 p.m.
“Short Time”	Format(0.874, “Short Time”)	20: 58

جدول ۳ــ۶ چگونگی استفاده از تابع Format()برای کار کردن با مقادیر عددی برای نمایش به صورت رشتهٔ دلخواه را نشان می‌دهد.

جدول ۳ــ۶ــ 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
"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. به‌عنوان مثال، اگر می‌خواهید مطمئن شوید که ورودی کاربر همیشه دو رقم اعشار داشته باشد، از قالب ("###,##0.00",235.6)Format استفاده کنید که مقدار 235.60 را برمی‌گرداند.

۵ــ۶ــ محاسبهٔ 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 :

```
My Long = Date Diff (Interval, Start_date, End_Date [,First Day of Week
[,First Week Of Year]])
```

- MyLong نام متغیری است که خروجی تابع در آن قرار می‌گیرد و از نوع Long است.
- DateDiff نام تابع است.
- Interval رشته‌ای است که فاصلهٔ زمانی براساس آن، تفاوت تاریخ را اندازه‌گیری خواهد کرد (جدول ۴ــ۶).

جدول ۴ــ۶ــ مقادیر مختلف پارامتر InterVal مربوط به تاریخ 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

- Start_Date تاریخ شروع است (از نوع Date).
- End_Date تاریخ پایان است (از نوع Date).
- FirstDayOf Week، یک ثابت اختیاری است که اولین روز هفته است.
- FirstWeekOfYear، عبارت ثابت اختیاری است که اولین هفتهٔ 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 رشتهٔ پارامتر Interval تعیین می‌شود (جدول ۴ــ۶).

نکته

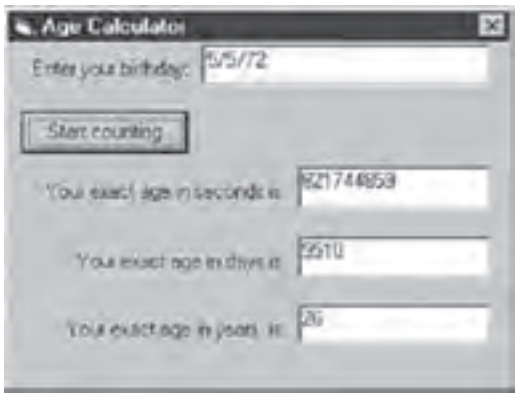
()IsDate یک تابع VB است که یک رشته یا مقدار تاریخ را بررسی می‌کند. اگر رشته یا تاریخ شبیه یک تاریخ معتبر باشد، تابع ()IsDate مقدار True را برمی‌گرداند و در غیراین‌صورت مقدار False را برمی‌گرداند. به عنوان مثال، IsDate("September 16 1998") و IsDate(#9/16/98#) مقدار True را برمی‌گرداند ولی IsDate("Birthday")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

تابع CDate ورودی خود را به نوع Date تبدیل می‌کند لذا هرگاه رشته‌ای از نظر تابع IsDate شبیه یا یک تاریخ معتبر است می‌توان با استفاده از تابع CDate آن را به قالب تاریخ (نه شبیه به تاریخ) تبدیل کرد.

مثال ۴ــ۶

برنامه‌ای بنویسید که تاریخ تولد کاربر را دریافت و در صورت معتبر بودن، اعلام کند که چند سال، چند روز و چند ثانیه از آن تاریخ گذشته است.

پروژه‌ای مطابق شکل ۴ــ۶ ایجاد کنید (نام پروژه را DateDiff.vbp قرار دهید).



شکل ۴ــ۶ــ استفاده از تابع ()Format خواندن مقادیر روز و ثانیه‌ها را ساده‌تر می‌کند.

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

بعد از این که کاربر کادر پیام را بست، مکان‌نما به کادر متن برگشته و متن نادرست مشخص شده و از روال خارج می‌شود.

در روال رویداد Timer1_Timer() که هر نیم‌ثانیه (Interval = 500) اجرا می‌شود، کنترل Timer با استفاده از تابع Now، تاریخ و ساعت سیستم را به‌دست می‌آورد. روال رویداد از تاریخ DateDiff() سه بار استفاده می‌کند (یک بار برای به‌دست آوردن اختلاف زمان برحسب ثانیه و سپس روز و در پایان سال). مقادیر برگشتی در متغیرهای LngYear و LngSec قرار داده می‌شود. کد زیر، رویدادهای مربوط به این برنامه را ارائه می‌کند :

```
01    Private Sub cmdStart_Click()
02
05    If IsDate(txtBDate. Text)Then                    بررسی اعتبار ورودی
09    Bdate = CDate(txtBDate.Text)                     تبدیل به فرمت تاریخ
10    Else
12    MsgBox "You must enter a proper date!",vbCritical, "Data error"
14    txtBDate.SetFocus                                انتقال فوکوس به کادر متن
16    txtBDate.SelStart = 0
18    txtBDate.SelLength = Len(txtBDate.Text)  Highlight کردن متن داخل textbox
20    Exit Sub
21    End If
24    Timer1.Enabled = True                            فعال کردن تایمر
25    End Sub
```

```
26
27      Private Sub Timerl_Timer()
28      Dim LngSec As Long
29      Dim LngDay As Long
30      Dim LngYear As Long
31
33       LngSec = DateDiff("s", BDate, Now)          چند ثانیه از تاریخ تولد گذشته
34       LngDay = DateDiff("d", BDate, Now)          چند روز از تاریخ تولد گذشته
36      LngYear = DateDiff("yyyy", BDate, Now)       چند سال از تاریخ تولد گذشته

39
42      lblAgeSecs.Caption = CStr(LngSec)
43      lblAgeDays.Caption = CStr(LngDay)
44      lblAgeYears.Caption = CStr(LngYear)
45      End Sub
```

۶ــ۶ــ کاربرد متغیرهای ایستا به همراه Timer

فرض کنید 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، هربار که روال قطع شود، متغیر از حوزهٔ عمل خود خارج شده و با صفر مقداردهی می‌شود.

```
        Dim i As Integer
01      Private Sub Timer1_Timer()
02          Form1.Caption = "i is: " & Cstr(i)
03          If I > 10 Then Timer1.Enabled = False
04          i = i + 1
05      End Sub
```

اگرچه 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.

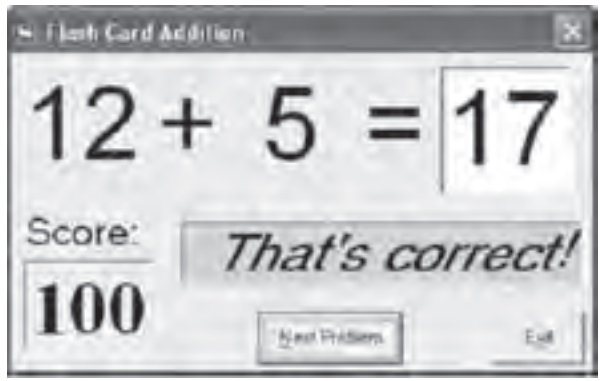
یک راهکار بهتر، تعریف متغیر شمارنده با کلید واژهٔ Static است. برای انجام این کار در کد فوق، بین خطوط ۱ و ۲ کد زیر را بنویسید :

```
Static i As Integer
```

هنگامی که یک متغیر Static ایجاد می‌کنید، مقدار آن حتی بعد از خروج از روالی که در آن تعریف شده است، حفظ می‌شود. مزیت انجام این کار، این است که متغیر درون کنترلی که به آن وابسته است، کپسوله‌سازی می‌شود. مقدار متغیر بدون درنظرگرفتن وضعیت روالی که در آن ایجاد شده است، پایدار باقی می‌ماند.

مثال ۵ــ۶

فرمی به‌صورت زیر (شکل۵ــ۶) ایجاد کنید. می‌خواهیم با کلیک روی دکمهٔ Next Problem به‌طور خودکار ۲ عدد تصادفی (بین صفر تا ۲۰) تولید شود و حاصل‌جمع آن‌ها را کاربر تایپ کند. در صورتی که پاسخ مثبت بود، به امتیاز Score ، ۵ امتیاز اضافه شود و پیغام صحیح بودن بدهد. اگر پاسخ نادرست بود، ۵ امتیاز کسر کند و با پیغام خطا پاسخ درست را اعلان کند (توجه کنید که امتیاز از ۱۰۰ بیشتر نباشد). با کلیک روی Next این عملیات تکرار شود.



شکل ۵ــ۶

کد این برنامه به‌صورت زیر جمع خواهد بود :

```
Option Explicit
Dim Sum As Interger
```

```
Dim NumProb As Integer, NumRight As Integer

Private Sub cmdExit_Click()
        End
End Sub
Private Sub cmdNext_Click()
    Dim Number1 As Integer
    Dim Number2 As Integer
    txtAnswer.Text =" "
    lblMessage. Caption = " "
    Number 1 = Int(Rnd *21)                    تولید عدد تصادفی اول بین ۰ و ۲۱
    Number2 = Int(Rnd *21)                     تولید عدد تصادفی دوم بین ۰ و ۲۱
    LbINum1. Caption = Number1                 نمایش عدد اول
    LbINum2.Caption = Number2                  نمایش عدد دوم
    Sum = Number1 + Number2                    محاسبهٔ حاصل جمع دو عدد
    cmdNext.Enabled = False
    txtAnswer. SetFocus        انتقال فوکوس به کادر متن txtAnswer برای دریافت جواب
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
If KeyAscii = 13 Then                                   در صورت فشرده‌شدن کلید Enter
    If Val(txtAnswer.Text) = Sum Then   بررسی محتوای txtAnswer با حاصل جمع واقعی
    lblMessage.Caption = "That's correct!"
    If Val (lblScore.Caption) < 100 Then lblScore.Caption = lblScore.
    Caption +5                                  به روزرسانی امتیاز با اضافه‌کردن به میزان ۵
Else
    lblMessage.Caption = "Answer is " +Sum                  نمایش پاسخ صحیح
    IblScore.Caption = lblScore.Caption – 5   به روزرسانی امتیاز با کم‌کردن به میزان ۵
End If
    cmdNext. Enabled = True
    cmdNext.SetFocus
End If
End Sub
```

تمرین: شمارش معکوس

پروژه‌ای مطابق شکل ۶ـ۶ ایجاد کنید و یک کنترل Timer و یک برچسب روی فرم قرار دهید تا شمارش معکوس را شبیه‌سازی کند.

شکل ۶ـ۶

خودآزمایی

۱ــ در هریک از دستورات زیر مقدار ذخیره‌شده در متغیر strS را مشخص کنید.

الف) strS = Format(d, "hampm")

ب) strS = Format(12345.67, "######.### ")

۲ــ فرمی تهیه کنید به این شرح که، با انتخاب تاریخ، در قسمت پایین تقویم، تاریخ نمایش داده شود. با کلیک روی دکمهٔ مقایسه، تاریخ با تاریخ روز مقایسه شود و پیغام دهد که آیا تاریخِ گذشته یا آینده است؟

۳ــ به مثال ۱۰ــ۶ دکمهٔ CmdTimer اضافه کنید که با کلیک آن تصاویر به طور خودکار نمایش داده شوند (هر ۱۰ ثانیه یک تصویر) و با کلیک دوبارهٔ این دکمهٔ نمایش خودکار متوقف شود.

۴ــ برنامه‌ای بنویسید که از ۱ تا ۱۰ را بشمارد و فاصلهٔ بین هر عدد با عدد بعدی ۳ دقیقه باشد (هر سه دقیقه به شماره قبلی یکی اضافه کند) برنامه را یکبار با استفاده از متغیر عمومی و بار دیگر با استفاده از متغیر ایستا بنویسید.

۵ــ برنامه‌ای بنویسید که تاریخ تولد دو دانش‌آموز را دریافت کرده و اختلاف سنی آن‌ها را برحسب ثانیه، روز و سال نمایش دهد.

فصل هفتم

گرافیک

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

- **از دستورات و توابع گرافیکی در برنامه‌های خود استفاده کند؛**
- **شکل‌های مختلفی را با نوشتن کدهای خاصی، ترسیم نماید.**

استفاده از هنر گرافیک در پروژه‌ها سبب می‌شود که فرم‌ها گویاتر و جذّاب‌تر شوند. علاوه بر این، کاربران نیز می‌توانند با استفاده از گرافیک‌ها عملیات خاصی را به سادگی انجام دهند.

۱ــ۷ــ اضافه کردن گرافیک به فرم

با استفاده از کنترل‌های Picture Box و Image می‌توان تصویری را به فرم اضافه کرد. با وجود این که درفصل اول با این دو کنترل و تفاوت‌های آن‌ها آشنا شدید ولی در این‌جا این تفاوت‌ها را به‌گونهٔ دیگری مورد بررسی قرار می‌دهیم :

الف ــ Picture Box ، برای گروه‌بندی کنترل‌های دیگر می‌تواند استفاده شود ولی Image این قابلیت را ندارد.

ب ــ Picture Box ، در صورت بزرگ بودن تصویر، قسمتی از آن را که به اندازهٔ کادرش است نشان می‌دهد و مشخصه‌ای به نام AutoSize دارد که در صورت True بودن، می‌تواند کادر Picture Box را به اندازهٔ واقعی تصویر بزرگ یا کوچک کند ولی کنترل Image در صورت بزرگ بودن تصویر، به‌طور پیش‌فرض کادرش را به اندازهٔ تصویر بزرگ می‌کند و دارای مشخصه‌ای به‌نام Stretch است که در صورت True بودن، قبل از قرار دادن تصویر، می‌تواند تصویر را به اندازهٔ کادر خود تبدیل کند.

درج تصویر در هر دو کنترل، مشابه بوده و با یکی از روش‌های صفحهٔ بعد انجام می‌شود :

- **در زمان طراحی فرم** : با استفاده از مشخصهٔ Picture می‌توان تصویر دلخواه را در این کنترل‌ها درج کرد.
- **در زمان کدنویسی** : با استفاده از تابع LoadPicture() می‌توان تصویر را در این کنترل‌ها قرار داد که شکل کلی آن به‌صورت زیر است :

```
ControlName.Picture = LoadPicture(StrFilePath)
```

- ControlName نام کنترل تصویری است.
- LoadPicture نام تابع است.
- StrFilePath رشته‌ای است که مسیر و نام پرونده گرافیکی موردنظر روی حافظهٔ جانبی را تعیین می‌کند.

نکته

اگر به جای StrFilePath رشتهٔ خالی قرار دهید، تصویر جاری حذف می‌شود.

در زمان کدنویسی می‌توان تصویر یک کنترل را به کنترل دیگری نسبت داد. به‌عنوان مثال، کد زیر تصویر Image2 را به Image1 نیز نسبت می‌دهد :

```
Image1.Picture = Image2.Picture
```

نکته

روش‌های درج تصویر فوق، فقط مختص این دو کنترل نیستند و می‌توان در تمام کنترل‌هایی که مشخصهٔ Picture دارند (مانند Form ، Command Button و ...) از این روش‌ها برای درج تصویر استفاده کرد.

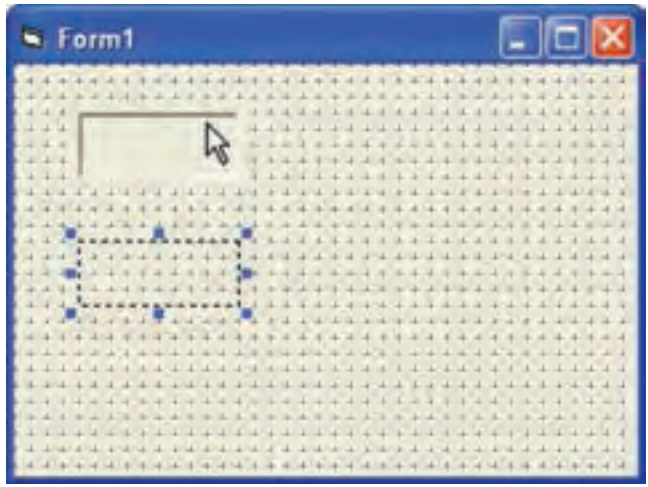
مثال ۱ــ۷

۱ــ پروژهٔ Standard EXE را باز کرده و آن را smplGrfx نام‌گذاری کنید. نام پیش‌فرض فرم را به frmMain تغییر دهید و مشخصهٔ Caption آن را با Simple Graphics مقداردهی کنید.

۲ــ یک کنترل Picture Box روی فرم اضافه کنید. نام این کنترل را picMain قرار دهید.

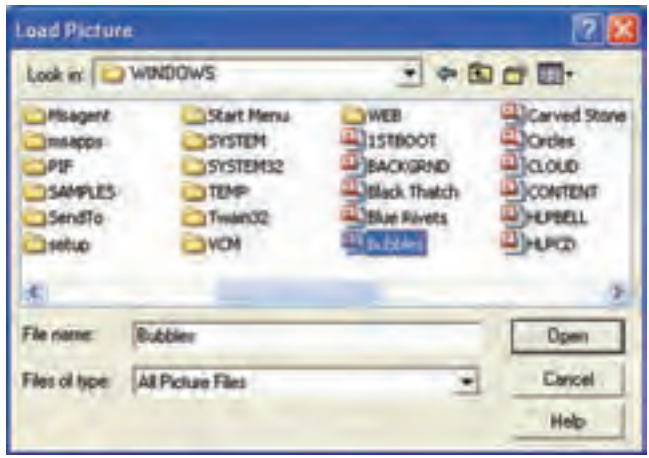
۳ــ یک کنترل Image به فرم اضافه کنید و نام آن را imgMain قرار دهید.

۴ــ کنترل‌های فوق را مطابق شکل ۱ــ۷ تغییر اندازه داده و جابه‌جا کنید.



شکل ۱ــ۷ــ مقدار پیش‌فرض مشخصهٔ BorderStyle مربوط به کنترل PictureBox برابر Single Fixed 1– و کنترل Image برابر با None – 0 است.

۵ــ کادر تصویر picMain را انتخاب کنید. از پنجرهٔ Properties در مقابل مشخصهٔ Picture روی علامت سه نقطه کلیک کنید تا کادر محاوره‌ای Load Picture باز شود (شکل۲ــ۷).



شکل ۲ــ۷ــ پوشهٔ Windows موجود در سیستم، انواع مختلفی از فایل‌های گرافیکی را ارائه می‌کند.

نکته

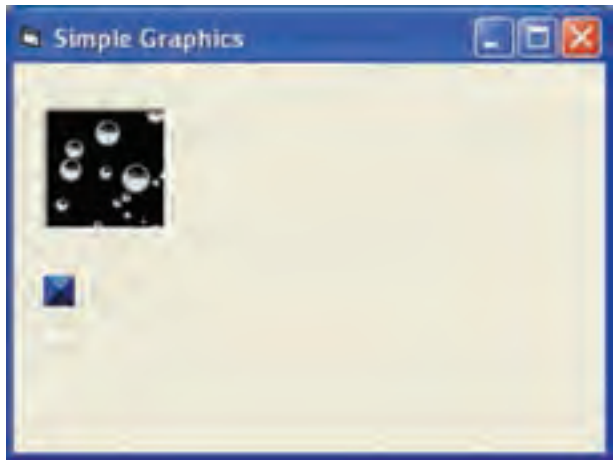
Bubbles.bmp یک تصویر نقش بیتی است که همراه ویندوز ارائه می‌شود. اگر به هر دلیلی این پرونده در مسیر مورد نظر نبود، می‌توانید هر پرونده نقش بیتی یا نشانه، metafile، jPEG و GIF را انتخاب کنید. همهٔ این قالب‌ها را می‌توان در PictureBox قرار داد.

۶ــ از پوشهٔ Windows فایل Bubbles.bmp را انتخاب کنید. هنگامی که روی دکمهٔ Open کلیک کنید، این تصویر نقش بیتی در کادر تصویر ظاهر خواهد شد.

۷ــ مقدار مشخصهٔ AutoSize کادر تصویر picMain را با True مقداردهی کنید. انجام این کار، سبب می‌شود که کادر تصویر به اندازهٔ تصویر، بزرگ شود.

۸ــ کنترل Image را انتخاب کرده و مقدار مشخصهٔ picture آن را پرونده تصویری Triangles.bmp قرار دهید (این فایل نیز در پوشهٔ Windows قرار دارد). کنترل Image مشخصهٔ AutoSize ندارد.

۹ــ پروژه را ذخیره کرده و اجرا کنید (شکل ۳ــ۷).



شکل ۳ــ۷ــ کنترل Image از مشخصهٔ AutoSize پشتیبانی نمی‌کند و به‌طور خودکار ابعاد این کنترل مطابق با تصویر، تغییر اندازه پیدا می‌کند.

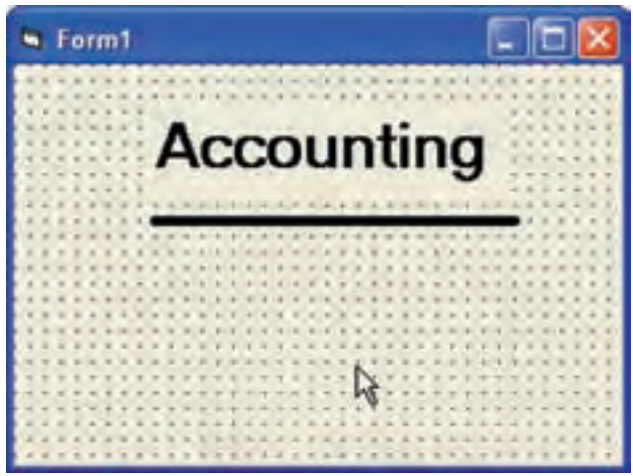
پژوهش: بررسی کنید که چه کنترل‌هایی برای گروه‌بندی کنترل‌ها استفاده می‌شود.

۲ــ۷ــ کنترل‌های ترسیمی

جعبه ابزار Visual Basic دارای دو کنترل ترسیمی است :

- کنترل خط، که بین دو نقطه یک خط راست رسم می‌کند.
- کنترل شکل، که روی فرم اشکال هندسی رسم می‌کند.

۱ــ۲ــ۷ــ ترسیم خط : خط راست یکی از اشکالی است که کاربردهای فراوانی می‌تواند داشته باشد، حتی اگر برنامه دارای جنبه‌های گرافیکی هم نباشد. در شکل ۴ــ۷ یک فرم را که در آن از خط استفاده شده ملاحظه می‌کنید؛ این فرم می‌تواند فرم معرفی یک برنامه باشد.

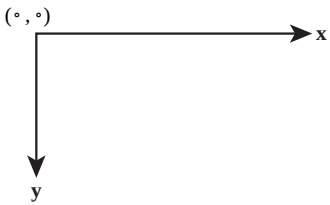


شکل ۴ــ۷ــ با خط می‌توان روی قسمت‌های مهم فرم تأکید کرد.

اگر روی کنترل خط (Line) در جعبه ابزار دو بار کلیک کنید، Visual Basic یک خط راست در وسط فرم قرار خواهد داد. این خط دارای دو دستگیرهٔ تغییر اندازه در دو انتهای خود است که با آن‌ها می‌توانید اندازه و جهت خط را تغییر دهید. مهم‌ترین خواص خط عبارتند از :

- Border Color : رنگ خط
- Border Style : شکل ظاهری خط (به جدول ۱ــ۷ و شکل ۵ــ۷ نگاه کنید.)
- Border Width : پهنای خط بر حسب نقطه ($\frac{۱}{۷۲}$ اینچ)
- Y1،X1، X2، Y2 : مختصات دو سر خط (یک خط با دو نقطهٔ انتهایی آن تعریف می‌شود.)

محورهای مختصات به‌صورت روبه‌رو است. مبدأ مختصات گوشه بالا و سمت چپ فرم می‌باشد.



نکته

اگر BorderWidth بزرگ‌تر از ۱ شود. BorderStyle فقط حالت خط پیوسته خواهد داشت.

جدول ۱ــ۷ــ مقادیر خاصیت Border Style کنترل خط

ثابت نام‌دار	مفهوم
0 _Transparent	رنگ‌زمینهٔ فرم از ورای خط دیده خواهد شد.
1 – Solid	خط پیوسته
2 – Dash	خط‌چین
3 – Dot	نقطه‌چین
4 – Dash – Dosh –Dash	خط ـ نقطه ـ خط
5 – Dash – Dot – Dot–Dash	خط ـ نقطه ـ نقطه ـ خط

شکل ۵ــ۷ــ شکل ظاهری خط را خاصیت Border Style تعیین می‌کند.

۲ــ۲ــ۷ــ ترسیم شکل : با کنترل شکل (Shape) می‌توانید اشکال مختلف هندسی روی فرم خود رسم کنید. شکلی که رسم می‌شود توسط خاصیت Shape کنترل معین می‌شود :

- 0–Rectangle : مستطیل
- 1–Square : مربع
- 2–Oval : بیضی
- 3–Circle : دایره
- 4–Rounded Rectangle : مستطیل با گوشه‌های گرد
- 5–Rounded Square : مربع با گوشه‌های گرد

کنترل Shape، علاوه بر خاصیت Shape، چند خاصیت مهم دیگر هم دارد که آن‌ها را در جدول۲ــ۷ ملاحظه می‌کنید.

جدول ۲ــ۷ــ خواص کنترل Shape

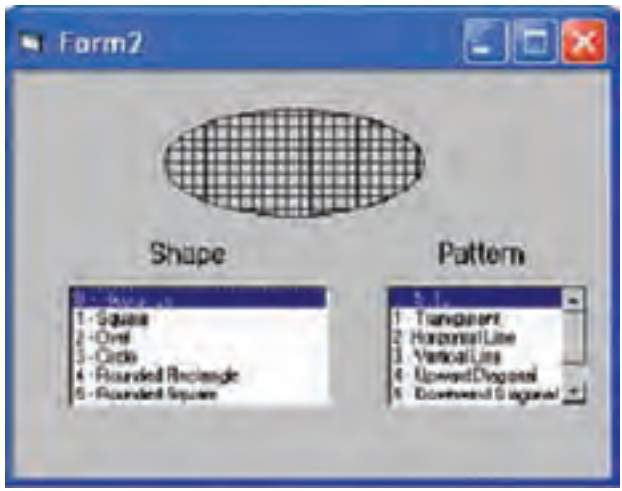
خاصیت	مفهوم
Shape	نوع شکل ترسیمی را مشخص می‌کند.
Back Style	(0/1) شفافیت شکل رسم شده را معین می‌کند.
Border Color	رنگ خط دور شکل
Border Style	ظاهر خط دور شکل
Border Width	ضخامت حاشیهٔ شکل (برحسب twips)
Fill Color	رنگ طرح داخلی شکل (به خاصیت Fill Style نگاه کنید.)
Fill Style	طرح داخل شکل (شکل ۶ــ۷ هشت طرح ممکن را نشان می‌دهد.)
Height	ارتفاع شکل
Width	عرض شکل

شکل ۶ــ۷ــ هشت طرح موجود برای خاصیت Fill Style کنترل Shape

مثال ۲ــ۷

فرمی طراحی کنید که با استفاده از دو کادر لیست، نوع شکل و شیوهٔ هاشور داخل شکل را انتخاب کنید.

شکل ۷ــ۷ اجرای این برنامه را نشان می‌دهد. با انتخاب گزینه‌ای از دو کادر لیست شکل و طرح می‌توانید ظاهر شکل موجود روی فرم را عوض کنید.



شکل ۷ــ۷ــ با انتخاب شکل و طرح نتیجهٔ کار را ببینید.

مراحل زیر را دنبال کنید :

۱ــ یک پروژهٔ جدید باز کنید.

۲ــ کنترل‌های موجود در شکل ۷ــ۷ را روی فرم قرار دهید.

۳ــ کد زیر را به برنامه اضافه کنید :

```
Private Sub Frm_Load()
```

اضافه‌کردن عناصر لیست کنترل lstShape ، که شامل انواع شکل‌هاست.

```
    lstShape.AddItem "0 – Rectangle"
    lstShape.AddItem "1 – Square"
    lstShape.AddItem "2 – Ovel"
    lstShape.AddItem "3 – Circle"
    lstShape.AddItem "4 – Rounded Rectangle"
    lstShape.AddItem "5 – Rounded Square"
```

اضافه‌کردن عناصر لیست کنترل lstpattern، که شامل انواع Styleهای پرکردن درون شکل است.

```
    lstPattern.AddItem "0 – Solid"
    lstPattern.AddItem "1 – Transparent"
    lstPattern.AddItem "2 – Horizontal Line"
    lstPattern.AddItem "3 – Vertical Line"
    lstPattern.AddItem "4 – Upward Diagonal"
    lstPattern.AddItem "5 – Downward Diagonal"
    lstPattern.AddItem "6 – Cross"
    lstPattern.AddItem "7 – Diagonal Cross"
    lstShape.ListIndex = 0
```

انتخاب اولین عنصر لیست lstshape،

```
    lstPattern.ListIndex = 0
```

انتخاب اولین عنصر لیست lstpattern،

```
End Sub
Private Sub lstpattern_Click()
```

```
    shpSample.FillStyle = lstPattern.ListIndex      پرکردن شکل با Style انتخاب شده
End Sub

Private Sub lstshap_Click()
    shpSample.Shape = lstShape.ListIndex      تغییر شکل براساس شکل انتخاب شده
End Sub

Private Sub mnuFileExit_Click()
End
    End Sub
```

۴ــ برنامه را اجرا کنید. گزینه‌هایی را در دو جعبه لیست Shape و Pattern انتخاب کرده و نتیجه را مشاهده کنید.

پژوهش: کنترل‌های Line و Shape چه رویدادهایی دارند؟

مثال ۳ــ۷

فرمی به شکل زیر طراحی کنید که دارای دو نوار لغزان برای تغییر شکل و هاشور آن باشد.

شکل ۸ــ۷

از Hscrooll1 برای تغییر شکل استفاده می‌شود لذا مقدار آن از ۰ تا ۵ تغییر می‌کند.
از Hscrooll2 برای تعیین شیوهٔ پرکردن داخل شکل استفاده می‌شود و مقدار آن از ۰ تا ۷ تغییر می‌کند.

```
Private Sub Form_Load()
   Hscrooll1.Min = 0
   Hscrooll1.Max = 5
   Hscrooll2.Min = 0
   Hscrooll2.Max = 7
   Shapel.Shape = 0                    شکل مستطیل با رنگ سیاه پر شده است
   Shapel.FillStyle = 0
End Sub ()
Private Sub Hscroll1_Change()
   Shapel.FillStyle = Hscroll1.Value
```

مقدار Value جدید شیوهٔ پرکردن داخل شکل را تعیین می‌کند.

```
End Sub
Private Sub Hscroll2_Change()
Shapel.Shape = Hscroll2. Value      مقدار Value جدید شکل جدید را تعیین می‌کند.
End Sub
```

۳ــ۷ــ متدهای ترسیمی

گرافیک به خط و اشکال هندسی محدود نیست. با Visual Basic قادرید نقطه به نقطهٔ صفحهٔ نمایش را در اختیار خود گرفته و کنترل کنید.

۱ــ۳ــ۷ــ متد Pset : متدی که با آن می‌توانید پیکسل‌ها را دستکاری کنید Pset نام دارد. با متدهای ترسیمی، دیگر نیازی به کنترل‌های گرافیکی ندارید. این متدهای ترسیمی را روی کنترل جعبه تصویر هم می‌توانید اعمال کنید.

شکل کلی متد Pset چنین است :

```
frmName.Pset [Step] (intX, intY), [color]
```

● (intX, intY) مختصات نقطه‌ای است که می‌خواهیم روشن شود.

مختصات این نقطه را می‌توان به دو روش تعیین کرد.

۱ــ به‌صورت مطلق بدون Step

۲ــ به‌صورت نسبی با استفاده از Step که مختصات نقطه را با توجه به فاصلهٔ آن از موقعیت فعلی مکان‌نما تعیین می‌کند.

مثال :

```
frmDraw.pset (100, 200)
frmDraw.psetstep (20, 20)
```

دستور اول برای روشن‌کردن پیکسل (۱۰۰,۲۰۰) است و دستور دوم روشن‌کردن پیکسل (۱۲۰,۲۲۰)که (۲۰+۱۰۰,۲۰۰+۲۰) می‌باشد.

● color، کد رنگی است که با آن نقطهٔ موردنظر روشن می‌شود.

نقطهٔ (۰،۰) در متد Pset گوشهٔ چپ بالای فرم است (البته این وضعیت را با خواص ScaleX و ScaleY می‌توانید تغییر دهید.) دستور زیر پیکسل (۱۰۰،۲۰۰) را روشن می‌کند :

```
frmDraw.Pset (100, 200)     'Turn on a Pixel
```

رنگ این پیکسل همان رنگ ForeColor فرم خواهد بود. با تعیین یک عدد هگزادسیمال یا یکی از ثابت‌های Visual Basic (از قبیل vbBlack، vbRed، vbBlue، vbGreen، vbYellow، vbMagenta، vbCyan یا vbWhite) برای آرگومان Color متد Pset می‌توانید رنگ نقطه را هم تنظیم کنید. دستور زیر نقطهٔ (۱۰۰،۲۰۰) را به رنگ زمینهٔ فرم درمی‌آورد :

```
frmDraw.Pset (100,200), frmDraw.BackColor  'Turn off a Pixel
```

همان‌طورکه گفته شد، آرگومان Step باعث تغییر در تعیین مکان پیکسل‌ها خواهد شد. به کمک Step می‌توانید مکان یک پیکسل را نسبت به نقطهٔ قبلی تنظیم کنید. به‌دستور زیر توجه کنید :

```
frmDraw. Pset Step (300,350) 'Relative Pixel Location
```

با این دستور پیکسل (۵۵۰، ۴۰۰) روشن خواهد شد (۲۰۰+۳۵۰، ۱۰۰+۳۰۰). توجه کنید که (۲۰۰،۱۰۰) مختصات نقطهٔ قبلی است.

با متد Pset خط هم می‌توان رسم کرد؛ به مثال زیر توجه کنید :

```
For intX = 1 To 100
   frmDraw.Pset (intX, 250)
```

```
Next intX
```

رسم خطی افقی که شامل نقاطی با y=۲۵۰ است و x آن‌ها از ۱ تا ۱۰۰ تغییر می‌کند.

۲ــ۳ــ۷ــ متد Line : برای رسم خط، متد مستقلی وجود دارد که آن را در زیر ملاحظه می‌کنید :

```
frmName.Line [Step] (intX1, intY1)–[Step] (intX2, intY2), [Color] , [B | BF]
```

- در این دستور دو زوج، نقطهٔ مختصات ابتدا و انتهای خط هستند. با آرگومان Step می‌توانید مکان نقطه‌ها را نسبت به نقطه‌های قبلی محاسبه کنید. دستور زیر یک خط از پیکسل (۱۰۰،۱۰۰) به پیکسل (۱۵۰، ۱۵۰) رسم می‌کند.

```
FrmDraw. Line (100,100) – (150, 150)
```

- آرگومان Color رنگ خط را مشخص می‌کند (اگر میل ندارید خط با رنگ ForeColor فرم رسم شود).
- برای رسم مستطیل می‌توانید از گزینهٔ B در متد Line استفاده کنید.

```
frmDraw. Line (100,100) – (150, 150),  ,B
```

در این دستور، زوج نقاط مختصات گوشهٔ چپ ــ بالا و راست ــ پایین مستطیل خواهند بود : کامای اضافه را فراموش نکنید! زیرا محل آرگومان رنگ در دستور line را تعیین می‌کند و اگر فراموش شود B به‌عنوان رنگ در نظر گرفته می‌شود.

- برای رنگ کردن داخل مستطیل رسم شده می‌توانید از گزینهٔ BF استفاده کنید :

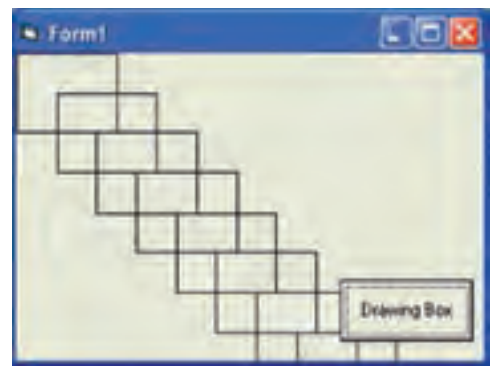
```
frmDraw. Line (35,40) – (150, 175), vbGreen, BF    'A Green Box
```

دستور فوق یک مستطیل سبز رنگ رسم خواهد کرد. اگر یکی از گوشه‌های مستطیل خارج از محدودهٔ فرم قرار گیرد، Visual Basic آن را فقط تا لبه‌های فرم رسم خواهد کرد، و حتی اگر بعد از رسم مستطیل فرم را بزرگ کنید باز هم قسمت پنهان مستطیل را نخواهید دید مگر آن که مشخصه Auto Redraw آن را با True مقدار دهی کنید.

مثال ۴ــ۷

در کد صفحهٔ بعد، روالی را مشاهده می‌کنید که چند مستطیل از سمت چپ ــ بالای فرم به سمت گوشهٔ راست ــ پایین آن رسم می‌کند؛ به شکل ۹ــ۷ نگاه کنید.

```
1:  Private Sub cmdBoxes_Click()
2:          Dim intStartX As Integer
3:          Dim intStartY As Integer
4:          Dim intLastX As Integer
5:          Dim intLastY As Integer
6:          Dim intCTr as Integer
7:          intStartX =  0
8:          intStartY =  0
9:          intLastX =  1000
10:         intLastY =  800
11:             رسم ۲۰ مستطیل
12: For intCtr = 1 To 20
13:         frmBoxes.Line (intStart X, intStartY) – (intLastX, intLastY),  ,B
                                                              رسم مستطیل
14:
15:         'Prepare for next set of boxes          تعیین مختصات مستطیل بعدی
16:         intStartX = intStartX + 400
17:         intStartY = intStartY + 400
18:         intLastX = intLastX + 400
19:         intLastY = intLastY + 400
20: Next intCtr
21:
22: End Sub
```



شکل ۹ــ۷ــ رسم مستطیل با متد Line

۳ــ۳ــ۷ــ متد Circle : متد Circle متدی است که می‌توانید به کمک آن دایره و بیضی رسم کنید :

frmDraw.Circle [Step] [intX, intY), sngRadius, [Color], , , sngAspect

(کاماهای اضافی مربوط به آرگومان‌های پیشرفته‌تر هستند که در این کتاب آن‌ها را مورد بحث قرار نمی‌دهیم؛ فقط اشاره می‌کنیم که حتماً باید آن‌ها را قبل از آرگومان sngAspect قید کنید.)

- intX و intY مختصات مرکز دایره هستند.
- sngRadius شعاع دایره (برحسب پیکسل) است.
- آرگومان Step باعث تعیین مکان نسبی مرکز دایره خواهد شد.

دستور زیر دایره‌ای به مرکز (۲۰۰،۳۰۰) شعاع ۱۰۰ پیکسل رسم خواهد کرد :

frmDraw.Circle (300,200), 100

اگر رنگی در آرگومان Color قید نشود، این متد از رنگ ForeColor فرم استفاده خواهد کرد. برای رسم بیضی‌می‌توانید

- از آرگومان sngAspect (که نسبت قطر عمودی به قطر افقی آن است) استفاده کنید. اگر این آرگومان ۱ باشد (حالت پیش‌فرض) یک دایره رسم خواهد شد، اما اگر عددی کوچک‌تر یا بزرگ‌تر از ۱ باشد دایره (به ترتیب) به صورت بیضی افقی یا عمودی تغییر خواهد کرد. در شکل ۱۰ــ۷ بیضی‌هایی را که دو دستور زیر رسم می‌کنند، مشاهده می‌کنید :

frmDraw.Circle (1000, 1250), 400, , , , 0.2

frmDraw.Circle (1750, 1250), 400, , , , 4

شکل ۱۰ــ۷ــ شکل بیضی را آرگومان **SnyAspect** تعیین می‌کند.

پژوهش: با استفاده از کدام پارامترها در متد Circle می‌توان بخشی از دایره را رسم کرد؟

۴ــ۷ــ رنگ‌آمیزی

همان‌طور که در کار با اغلب کنترل‌ها مشاهده کردید، می‌توان رنگ رویه و زمینهٔ آن‌ها را تغییر داد. برای تعیین رنگ، روش‌های متفاوتی وجود دارد که از جمله عبارت است از :

- **استفاده از تابع QBColor** : به کمک این تابع می‌توان رنگ‌های مورد استفاده در QBasic را به کار برد. شکل کلی این تابع به صورت QBColor(n) است. مقدار متغیر n می‌تواند از صفر تا ۱۵ باشد.

مثال ۵ــ۷

دایره‌ای به شعاع ۱۰۰ و با رنگ قرمز ترسیم کنید.

```
Circle(100, 200), 100, QBColor(4)
```

- **استفاده از ثابت‌های نام‌دار ویژوال بیسیک** : این ثابت‌ها معمولاً با پیشوند vb و سپس نام رنگ ظاهر می‌شوند. مانند vbGreen, vbRed و ...

مثال ۶ــ۷

نقطه‌ای در مختصات (۱۰۰،۲۰۰) را با رنگ قرمز ترسیم کنید :

```
Pset (100,200), vbRed
```

- **استفاده از تابع RGB** : این تابع با ترکیب سه رنگ قرمز، سبز و آبی، رنگ‌های جدیدی را تولید می‌کند. شکل کلی تابع به صورت زیر است :

```
RGB(ColorRed, ColorGreen, ColorBlue)
```

مقدار هر سه پارامتر تابع فوق، از صفر تا ۲۵۵ می‌تواند باشد.

مثال ۷ــ۷

خطی به طول ۱۰۰ و به صورت عمودی با رنگ قرمز ترسیم کنید :

```
Line (100,200) – (200,200), RGB (255,0,0)
```

این تابع به دلیل این که می‌تواند ۲۵۶×۲۵۶×۲۵۶ رنگ تولید کند، نسبت به روش‌های قبلی کامل‌تر است.

- استفاده از کدهای رنگی که در پنجرهٔ Properties در مقابل مشخصه‌های رنگی وجود دارد.

مثال ۸ــ۷

نقطه‌ای با رنگ قرمز در مختصات (۲۰۰ و ۱۰۰) ترسیم کنید :

```
Pset (100,200), &HFF&
```

این روش نیز از نظر تعداد رنگ‌هایی که تولید می‌کند، شبیه تابع RGB است.

مثال ۹ــ۷

فرمی به شکل زیر طراحی کنید که با درگ کردن روی کادر تصویر بتوان هر چیزی نوشت (شبیه تخته سیاه).

شکل ۱۱ــ۷

۱ــ منویی با ساختار زیر ایجاد کنید :

File

— New —

Exit

Caption	Name
&File	mnuFile
&New	itmNew
	sepone
E&xit	itmExit

۲ــ یک کادر تصویر به نام PicBoard روی فرم قرار دهید.

۳ــ کد زیر را تایپ کرده و پروژه را اجرا کنید :

```
Private Sub Form_Load()
   PicBoard.ForeColor = QBColor(15)        تعیین رنگ قلم
   PicBoard.BackColor = QBColor(0)         تعیین رنگ زمینه
   PicBoard.DrawWidth = 3                  تعیین ضخامت قلم
End Sub
Private Sub PicBoard_MouseMove (button As Integer, Shift As Integer, x
As Single, y As Single)
   If Button = 1 Then
         PicBoard.Pset (x,y)          رسم نقطه در محل حرکت ماوس
   End If
End Sub
Private Sub itmNew_Click()
   PicBoard.Cls      پاک کردن تخته
End Sub
Private Sub itmExit_Click()
   End
End Sub
```

مثال ۱۰ــ۷

ترسیم دایره‌های تصادفی

شکل ۱۲ــ۷

```
Private Sub CmdClear_Click()
   Picl. Cls          Picl  پاک‌کردن محتوای کنترل
End Sub
Private Sub CmdDraw_Click()
   Dim Intx As Integer, Inty As Integer, IntR As Integer, Intc As Integer
   Randomize Timer به منظور تولید اعداد تصادفی متفاوت در هر بار اجرای برنامه
   For i = 1 To 50
   Intx = Int(Rnd* Pic1. ScaleWidth)
                              تولید عدد تصادفی بین ۰ و پهنای کنترل Picl برای x
   Inty = Int(Rnd* Pic1. ScaleHeight)
                              تولید عدد تصادفی بین ۰ و ارتفاع کنترل Picl برای y
   IntR = Int(Rnd* 40)        تولید عدد تصادفی بین ۰ و ۴۰ برای شعاع دایره
   Intc = Int(Rnd* 16)        تولید عدد تصادفی بین ۰ و ۱۶ برای رنگ
   Picl. Circle (Intx, Inty), IntR, QBColor (Intc)          رسم دایره
Next
```

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

اگر بخواهیم در رویداد Form–load از متدهای ترسیمی استفاده کنیم باید مشخصهٔ AutoReDraw فرم را True کنیم این مشخصه سبب می‌شود که پس از قرارگرفتن فرم در حافظه تمام ترسیمات دوباره انجام شود. در حالت پیش‌فرض این مشخصه مساوی False است و پس از load شدن فرم چیزی روی فرم رسم نشده و تمام ترسیمات باید دوباره رسم شود.

```
Private Sub Form_Load()
   Picl. AutoRedraw = True
   Picl. BackColor = QBColor(15)
   Picl. ScaleMode = vbPixel
   Picl. TabStop = False
End Sub
```

از کلید Tab می‌توان برای انتقال فوکوس به کنترل‌های روی فرم استفاده کرد. اگر بخواهیم از انتقال فوکوس به کنترل با کلید Tab جلوگیری کنیم خصوصیت Tab Stop آن کنترل را مساوی False قرار می‌دهیم (پیش‌فرض این خصوصیت True است.)

مثال ۱۱ــ۷

چیدن کاشی‌وار یک تصویر روی فرم

۱ــ پروژهٔ جدیدی به‌صورت زیر ایجاد کنید(شکل ۱۳ــ۷) و تصویر موردنظر را برای کنترل Picture تعیین کنید.

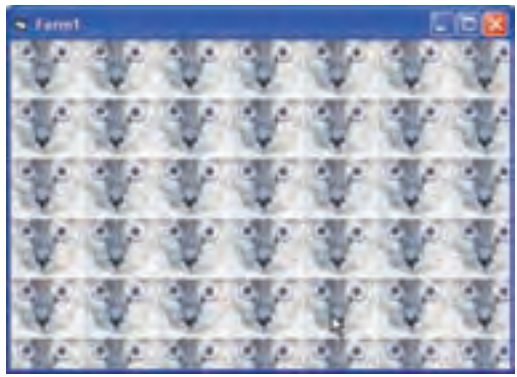
شکل ۱۳ــ۷

۲ــ کد مربوطه را وارد کنید :

```
Option Explicit
Private Sub Form_Paint()
```

```
Dim wid As Single
Dim hgt As Single
Dim x As Single
Dim y As Single
  wid = Picture1.ScaleWidth                    به‌دست‌آوردن پهنای Picture1
  hgt = Picture1.ScaleHeig                     به‌دست‌آوردن ارتفاع Picture1
  y = 0
  Do While y < Me.ScaleHeight    تا زمانی که y از ارتفاع فرم کوچکتر است
  x = 0
رسم یک سطر از Picture ها
  Do While x < Me.ScaleWidth     تا زمانی که x از پهنای فرم کوچکتر است
PaintPicture Picture1.Picture,x,y, wid, hgt رسم Picture جدید در مکان تعیین شده
  x = x + wid                                  تعیین x بعدی
  Loop
تمام شدن رسم یک سطر
  y = y + hgt                                  تعیین y بعدی
  Loop
End Sub
```

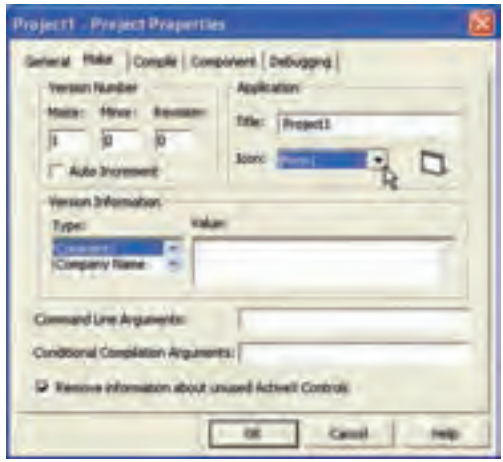
۳ـ پروژه را اجرا کنید (شکل ۱۴ـ۷).



شکل ۱۴ـ۷

۵ــ۷ــ ایجاد نشانه فرم

برنامه‌های ویندوز معمولاً دارای آیکن الصاق شده‌ای در داخل پرونده اجرایی هستند تا برنامه را به‌طور گرافیکی، نمایان کنند. هنگامی که برنامه‌ای را ایجاد می‌کنید، VB نشانه فرم را به آن اضافه می‌کند. مگر این‌که بخواهید این نشانه را تغییر دهید (شکل ۱۵ــ۷).



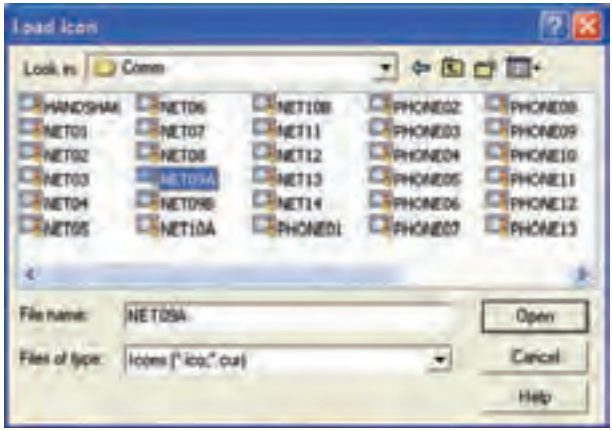
شکل ۱۵ــ۷ــ مشخصه‌های منحصر به فرد برنامه را در کادر محاوره‌ای Project Properties تعیین کنید.

با تعیین نشانه‌ای برای مشخصهٔ Icon یک فرم، می‌توان نشانه سفارشی را به پرونده اجرایی برنامه اختصاص داد. ویژوال بیسیک شامل چندین نشانه است که در پوشهٔ VB\Graphic\Icons\ قرار دارند.

مراحل زیر، چگونگی تغییر نشانه فرم را بیان می‌کنند :

۱ــ مشخصهٔ Icon فرم را انتخاب کنید.

۲ــ روی دکمهٔ سه نقطه کلیک کنید تا کادر محاوره‌ای Load Icon باز شود، نشانه موردنظر را انتخاب کنید (شکل ۱۶ــ۷).



شکل ۱۶ــ۷ــ در ویندوز، پرونده‌های نشانه، در همهٔ نماها تصاویر خودشان را نشان می‌دهند.

۳ــ روی دکمهٔ Open کلیک کنید تا نشانه به فرم اضافه شود.

اگر پروژه فقط دارای یک فرم باشد، اضافه کردن نشانه سفارشی به مشخصهٔ Icon فرم، به‌طور خودکار همان نشانه را برای برنامه تعیین می‌کند. اگر پروژه دارای چندین فرم باشد، باید نشانه برنامهٔ کاربردی را از کادر محاوره‌ای Project Properties تعیین کنید.

نکته

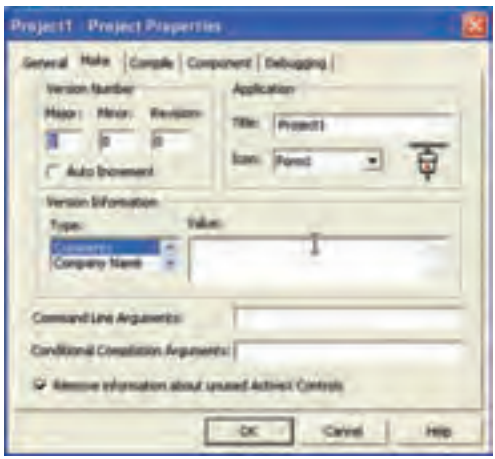
نمی‌توان نشانه‌ای را در VB یا Paint ویندوز ایجاد کرد. ولی CD ویژوال بیسیک دارای برنامه‌ای برای ایجاد نشانه است. این برنامه Microsoft Image Editor (imagedit.exe) نام دارد که در مسیر Common\Tools\Imagedit قرار دارد. اگر این برنامه را نمی‌توانید پیدا کنید، احتمالاً آن را نصب نکرده‌اید، برنامهٔ نصب ویژوال بیسیک را دوباره اجرا کرده و آن را نصب کنید.

تعیین نشانه برای برنامهٔ کاربردی

۱ــ نشانه‌ای را برای هر فرم در برنامهٔ کاربردی تعیین کنید.

۲ــ کادر محاوره‌ای Project Properties را بازکرده و زبانهٔ Make را انتخاب کنید.

۳ــ از لیست بازشوی Icon، فرمی را که دارای نشانه موردنظر برای استفاده به‌عنوان نشانه برنامه است انتخاب کنید (شکل ۱۷ــ۷).



شکل ۱۷ــ۷ــ می‌توان اغلب مشخصه‌های پروژه را در کادر محاوره‌ای Project Properties تعیین کنید.

خودآزمایی

۱ــ برنامه‌ای بنویسید که ۷۰ نقطه در محل‌های تصادفی و با رنگ و پهنای تصادفی ترسیم کند.

۲ــ برنامه‌ای بنویسید که مجموعه‌ای از خطوط با طول‌ها و رنگ‌های متفاوت ترسیم نماید.

۳ــ برنامه‌ای بنویسید که مجموعه‌ای از هشت دایرهٔ متحدالمرکز با فاصلهٔ ۱۰ پیکسل از یکدیگر ترسیم نماید.

۴ــ برنامه‌ای بنویسید که یک شبکهٔ توری (GRID) بر روی تصویری در یک Picture box ترسیم نماید.

۵ــ برنامه‌ای بنویسید که به کاربر امکان جابه‌جا کردن خط با مشخص کردن مقادیر X1، Y1، X2، Y2 را بدهد.

۶ــ برنامه‌ای بنویسید که کاربر بتواند با درگ‌کردن روی فرم، خطی را رسم کند و نقطهٔ شروع و پایان خط ضخیم‌تر از بقیهٔ نقاط باشد.

۷ــ برنامه‌ای بنویسید که تصویری را وسط فرم قرار دهد و سپس بارش برف را روی آن طراحی کنید.

فصل هشتم

تولید بسته‌های نرم‌افزاری

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

- **پروژه‌های خود را اجرایی کند؛**
- **برای پروژه‌های خود، یک پرونده قابل نصب (Setup) ایجاد کند.**

۱ــ۸ــ کارکردن با اطلاعات نسخه

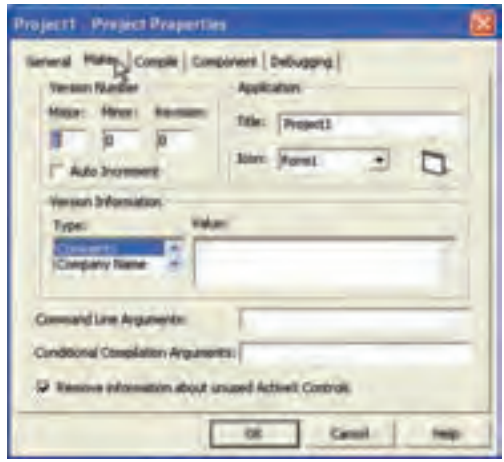
یکی از اولین موارد حرفه‌ای که می‌توان به برنامهٔ کاربردی اضافه کرد، درج اطلاعات عمومی متداول برنامهٔ کاربردی، شامل نام شرکت، شمارهٔ نسخه و سایر اطلاعات مشابه است. ویژوال بیسیک امکان ذخیرهٔ تمام این اطلاعات را با استفاده از شیء App (یک شیء تعریف شده در ویژوال بیسیک که نیاز به ایجاد آن برای برنامهٔ کاربردی ندارید) فراهم می‌کند.

اغلب مشخصه‌های شیء App برای ارایهٔ اطلاعات عمومی برنامهٔ کاربردی، مورد استفاده قرار می‌گیرند. جدول ۱ــ۸ مشخصه‌های متداول را نشان می‌دهد.

جدول ۱ــ۸ــ مشخصه‌های متداول شیء App

مشخصه	شرح
Comments	رشته‌ای را برمی‌گرداند که شامل توضیحاتی دربارهٔ برنامهٔ کاربردی است. در زمان اجرا فقط خواندنی است.
Company Name	نام شرکت یا تولیدکنندهٔ برنامهٔ کاربردی را برمی‌گرداند. در زمان اجرا فقط خواندنی است.
EXEName	نام پرونده اجرایی را بدون پسوند برمی‌گرداند (فقط خواندنی).
FileDescription	رشته‌ای که به طور خلاصه هدف برنامهٔ کاربردی را شرح می‌دهد. در زمان اجرا فقط خواندنی است.
HelpFile	فایل راهنمای مربوط به برنامهٔ کاربردی را تعیین می‌کند. در زمان اجرا خواندنی/ نوشتنی است.
Legal Copyright	رشتهٔ حق کپی‌رایت را برمی‌گرداند. از Character Map برای افزودن نمادهای ویژه در این کادر استفاده کنید. در زمان اجرا فقط خواندنی است.
Legal Trademarks	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	شمارهٔ اصلی نسخه را برمی‌گرداند (به عنوان مثال، ۴ در ۳، ۴). در زمان اجرا فقط خواندنی است.
Minor	شمارهٔ فرعی نسخه را برمی‌گرداند (به عنوان مثال، ۳ در ۳، ۴). در زمان اجرا فقط خواندنی است.
Path	فهرستی را که برنامهٔ کاربردی در آن قرار دارد برمی‌گرداند. در زمان اجرا فقط خواندنی است.
PrevInstance	اگر نمونه‌ای از برنامهٔ کاربردی در حال اجرا باشد، مقداری را برمی‌گرداند. در زمان اجرا فقط خواندنی است.
Product Name	نام محصول برنامهٔ کاربردی را برمی‌گرداند. در زمان اجرا فقط خواندنی است.
Revision	شمارهٔ بازنگری برنامهٔ کاربردی را برمی‌گرداند. در زمان اجرا فقط خواندنی است.

می‌توان از این مشخصه‌ها برای ارایهٔ اطلاعات مهم دربارهٔ برنامهٔ کاربردی موردنظر، استفاده کرد. این مشخصه‌ها در کادر محاوره‌ای Project Properties تعیین می‌شوند (شکل ۱ــ۸).



شکل ۱ــ۸ــ برای دسترسی به کادر محاوره‌ای Project Properties، گزینهٔ Project Name Properties (نام پروژه) را از منوی Project انتخاب کنید.

می‌توان مقادیر این مشخصه‌ها را در زمان اجرا از داخل کد خواند. همچنین می‌توان مقداری را برای مشخصه‌های اطلاعات نسخهٔ شیء App تعیین کرد که انجام این کار، با کلیک راست روی پرونده اجرایی و انتخاب Properties ممکن است (شکل ۲ــ۸).

شکل ۲ــ۸ــ برای مشاهدهٔ مشخصه‌های نسخه، زبانهٔ Version را از Properties انتخاب کنید.

نکته

مقادیر مشخصه‌های اطلاعات نسخهٔ شیء App درون قالب دودویی پرونده اجرایی اضافه می‌شوند و در زمان اجرا نمی‌توان آن‌ها را تغییر داد.

```
Private Sub cmdCopyright_Click()
    lblMain.Caption = App.LegalCopyright
End Sub

Private Sub cmdPath_Click()
    lblMain.Caption = App.Path
End Sub

Private Sub cmdProductName_Click()
    lblMain.Caption = App.ProductName
End Sub

Private Sub cmdVersionNum_Click()
    Dim strVerNum
    str VerNum = CStr(App. Major) & "." & CStr(App. Minor) & "."_
    & CStr(App.Revision)
    lblMain.Caption = strVerNum
End Sub
```

۲ــ۸ــ کامپایل‌کردن 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 پرونده‌های گرافیکی بودند که با IDE ویژوال بیسیک به‌وجود می‌آمدند. اکنون زمان آن است که این پرونده‌ها را به یک پرونده اجرایی تبدیل کنید که به طور مستقل از

IDE اجرا خواهد شد. این فرایند را **کامپایل کردن کد** یا **ساخت پرونده 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، پرونده اجرایی به‌وجود آمده به صورت کد مفسری اجرا می‌شود.

نکته

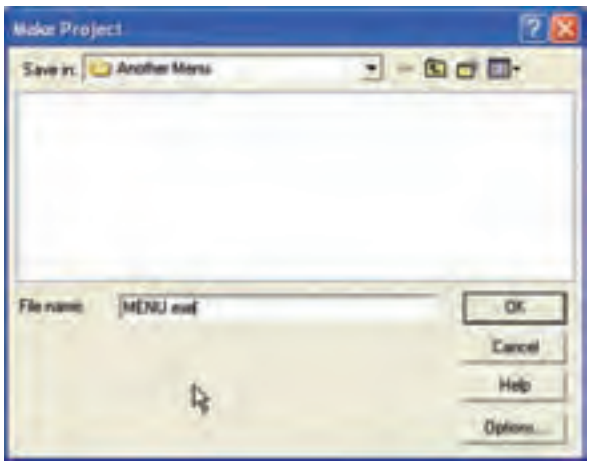
اندازهٔ پرونده اجرایی native code نسبت به P_code بزرگ‌تر است. بنابراین، اگر می‌خواهید کوچک‌ترین پرونده اجرایی را ارایه دهید، باید از P-code استفاده کنید. ولی اگر می‌خواهید که سریع‌ترین را ارایه دهید، باید native code را به وجود آورید.

اگر کد را به صورت native code کامپایل کنید، پرونده‌های 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 وسیلهٔ EXE به طور متفاوت مورد دسترسی و استفاده واقع می‌شوند).

۱ــ۲ــ۸ــ کامپایل کد به Standard EXE

۱ــ پروژه‌ای را که می‌خواهید کامپایل کنید باز کنید.

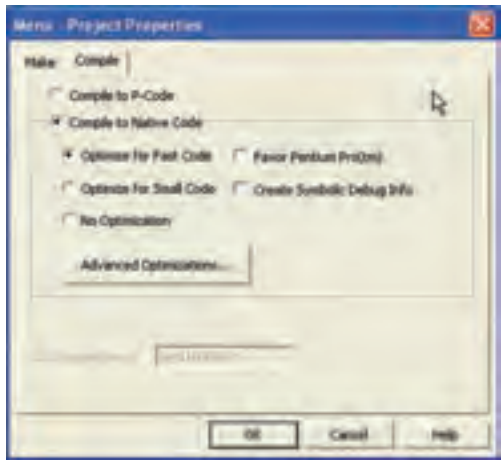
۲ــ از منوی File گزینهٔ Make Project Name.exe را انتخاب کنید. کادر محاوره‌ای Make Project ظاهر می‌شود (شکل ۳ــ۸).



شکل ۳ ــ۸ــ نام پرونده اجرایی را در این کادر محاوره‌ای وارد کنید. برای تغییر بعضی از مشخصه‌های شیء App روی دکمهٔ Options کلیک کنید.

۳ــ در صورتی که می‌خواهید، نام پرونده اجرایی را در کادر متن File Name تغییر دهید، روی دکمهٔ Options کلیک کنید تا کادر محاوره‌ای Properties باز شود. در زبانهٔ Compile یکی از دو روش کامپایل P–code یا Native code را انتخاب کنید.

۴ــ در کادرهای محاوره‌ای Project Properties و Make Project روی Ok کلیک کنید تا کد کامپایل شود.

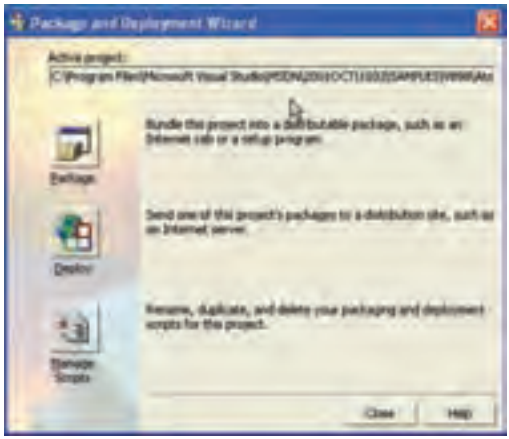


شکل ۴ــ۸ــ می‌توان چندین گزینه برای کامپایل Native code انتخاب کرد. به‌طور کلی، انتخاب کد سریع‌تر، پرونده اجرایی بزرگ‌تری را ایجاد می‌کند.

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

۳ــ۸ــ کاربرد Package and Deployment Wizard

از این ابزار می‌توان برای ایجاد بسته‌های نرم‌افزاری که نصب می‌شوند (Setup) از هر نوع برنامهٔ کاربردی در ویژوال بیسیک استفاده کرد. برای انجام این کار، Package and Deployment Wizard را شروع کنید (شکل ۵ــ۸). این ابزار باید در زیر منوی Visual Basic از منوی Start لیست شده باشد، زیرا به طور پیش فرض، نصب می‌شود. در صورتی که نیست، باید آن را روی سیستم نصب کنید.



شکل ۵ــ۸ــ ابزار Package and Deployment Wizard

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

۱ــ پرونده پروژه را در بالای کادر محاوره‌ای انتخاب کنید.

نکته

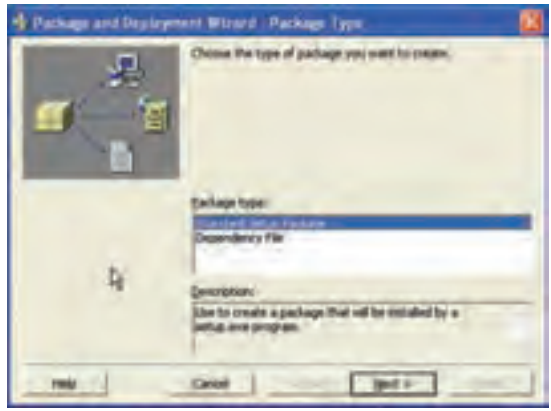
برای این مراحل، می‌توانید از هر پروژه‌ای که در ویژوال بیسیک دارید یا از نمونه‌های خود ویژوال استفاده کنید. ما از پروژهٔ ATM موجود در ویژوال بیسیک استفاده کرده‌ایم.

۲ــ برای ساخت یک برنامهٔ نصب (setup)، روی دکمهٔ Package کلیک کنید.

۳ــ در صورتی که برنامهٔ کاربردی را کامپایل نکرده باشید، ویزارد از شما کامپایل برنامه را سؤال می‌کند (شکل ۶ــ۸). برای ادامه روی دکمهٔ Compile کلیک کنید.

شکل ۶ــ۸ــ ویزارد نیاز به پرونده اجرایی در داخل برنامه Setup دارد، بنابراین به طور خودکار پرونده پروژهٔ برنامهٔ کاربردی را کامپایل می‌کند.

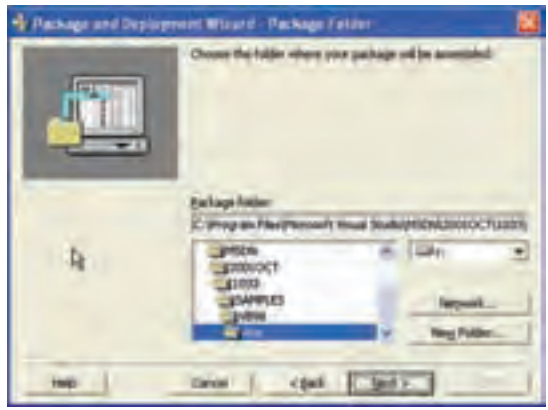
۴ــ بعد از کامپایل برنامهٔ کاربردی، از شما خواهد پرسید که چه نوع بستهٔ نرم‌افزاری می‌خواهید (شکل ۷ــ ۸). گزینهٔ Standard Setup Package را انتخاب کرده و روی OK کلیک کنید.



شکل ۷ــ ۸ــ نوع بستهٔ نرم‌افزاری را که می‌خواهید با ویزارد ایجاد کنید انتخاب کنید.

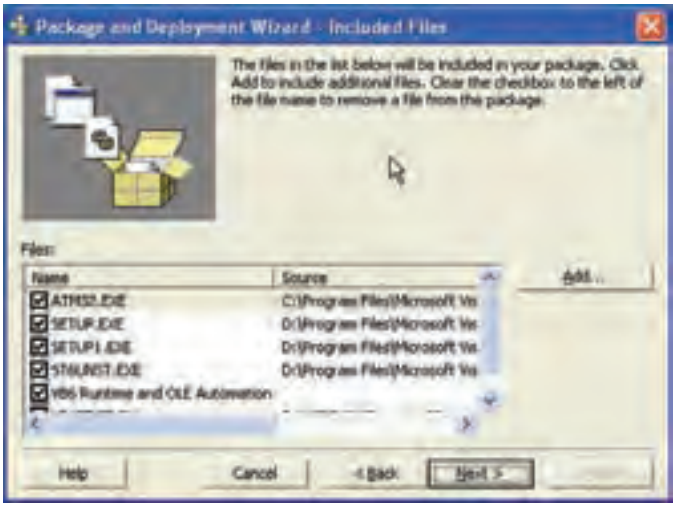
پرونده‌های فلاپی دیسک یا CD-ROM را در این مرحله برای ویزارد ارایه نکنید، زیرا قبل از کامل شدن فرایند، ویزارد چندین بار پرونده‌ها را تغییر می‌دهد. پرونده‌ها را روی هارد دیسک ذخیره کنید و بعد از پایان فرایند آن‌ها را روی CD کپی کنید.

۵ ــ محل پرونده‌های نصب را، که ویزارد ایجاد خواهد کرد، تعیین کنید. این پرونده‌ها به رسانهٔ مقصد که ممکن است دیسک یا CD باشد، کپی شوند. فهرستی را انتخاب کرده (شکل ۸ــ۸) و برای ادامه روی Next کلیک کنید.



شکل ۸ ــ۸ــ ویزارد نیاز به محلی برای نگهداری موقتی پرونده‌های موردنظر برای نصب دارد.

۶ــ همان‌طور که قبلاً نیز بیان شد، ویژوال بیسیک پرونده‌های زیادی به غیر از پرونده اجرایی دارد. کادر محاوره‌ای بعدی (شکل ۹ــ۸)، پرونده‌های مورد نیاز برای نصب را به همراه پرونده اجرایی لیست می‌کند. اگر پرونده‌های دیگری (مثل پرونده‌های راهنما) دارید که باید نصب شوند، می‌توانید آن‌ها را در این مرحله اضافه کنید. روی دکمهٔ next کلیک کنید.



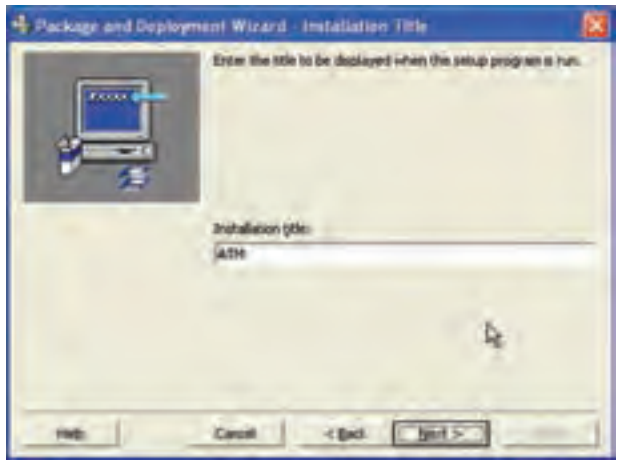
شکل ۹ــ۸ــ این کادر محاوره‌ای، لیست کاملی از پرونده‌های موردنیاز برای ایجاد برنامهٔ کاربردی را که روی کامپیوتر دیگری اجرا می‌شود نشان می‌دهد.

نکته

پرونده‌هایی را که ویزارد تولید خواهد کرد پرونده‌های CAB (کابینت) می‌نامند. این پرونده‌ها نوع خاصی از پرونده‌های آرشیوی هستند که به وسیلهٔ مایکروسافت طراحی شده و خیلی شبیه به پرونده Zip هستند. اگر از نرم‌افزار Winzip نسخهٔ ۷,۰ یا بالاتر استفاده می‌کنید، می‌توانید محتوای پرونده‌های CAB را مشاهده کنید.

۷ــ مرحلهٔ بعدی در ویزارد تعیین اندازهٔ رسانهٔ مقصد است. اگر قصد دارید برنامهٔ کاربردی را روی فلاپی دیسک‌ها قرار دهید، بزرگ‌ترین پرونده‌ای که ویزارد می‌تواند تولید کند به اندازهٔ دیسک خواهد بود. برای کپی پرونده‌ها روی فلاپی دیسک، Single Cab را انتخاب کرده و روی Next کلیک کنید تا ادامه یابد.

۸ــ عنوان برنامهٔ کاربردی را تعیین کنید (شکل ۱۰ــ۸). این عنوان در طول نصب نمایش داده خواهد شد. عنوان مناسبی را وارد کرده و برای ادامه روی Next کلیک کنید.

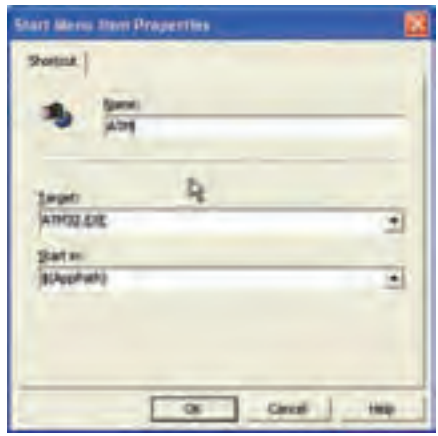


شکل ۱۰ــ۸ــ عنوان برنامهٔ کاربردی را در این کادر محاوره‌ای وارد کنید.

۹ــ برای برنامهٔ کاربردی که استفاده می‌کنید، نیاز دارید نشانه‌ای در منوی Start داشته باشید. مرحلهٔ بعدی ویزارد (شکل ۱۱ــ۸) یک روش منحصر به فرد برای تعیین گروه‌هایی از نشانه‌هاست. تنظیمات پیش‌فرض، ایجاد گروه با نام برنامهٔ کاربردی و سپس ایجاد نشانه‌ای برای شروع برنامه است.

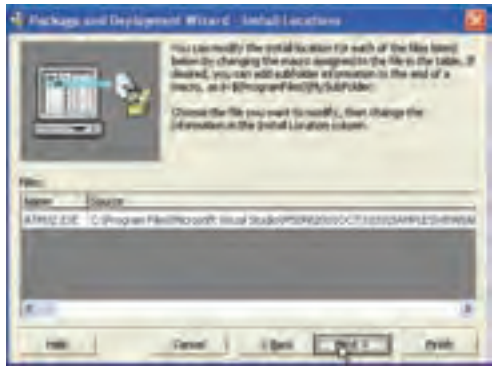
شکل ۱۱ــ ۸ ــ نشانه‌ها و گروه‌های موردنظر را برای برنامهٔ کاربردی انتخاب کنید.

به دلیل این‌که برنامهٔ کاربردی فقط دارای یک آیکن است، به‌طور استاندارد نشانه آن تحت All Programs ایجاد می‌شود. ابتدا روی گروه ATM و بعد روی دکمهٔ Remove کلیک کنید. سپس روی دکمهٔ New Item کلیک کرده و نام برنامهٔ کاربردی را وارد کنید، آن‌گاه در کادر محاوره‌ای که ظاهر می‌شود، روی Ok کلیک کنید (شکل ۱۲ــ۸). بعد از پایان اضافه کردن گروه‌ها و نشانه‌ها روی Next کلیک کنید.



شکل ۱۲ــ۸ــ گزینه‌های اصلی را برای هر نشانه‌ای که می‌خواهید ایجاد کنید تعیین کنید.

۱۰ــ کادر محاوره‌ای بعدی (شکل ۱۳ــ۸) امکان تغییر محل نصب هر پرونده‌ای را که به‌وسیلهٔ سیستم موردنیاز نیست فراهم می‌کند. تمام پرونده‌های 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.



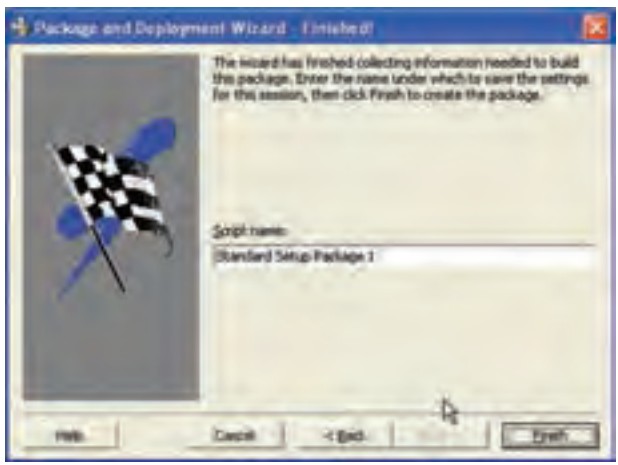
شکل ۱۳ــ۸ــ اگر نیاز به تغییر فهرست نصب برای پرونده‌هایی که اضافه کرده‌اید دارید، این کادر محاوره‌ای امکان انجام این کار را فراهم می‌کند.

۱۱ــ پرونده‌های اصلی (مثل DLLها و OCXها)، پرونده‌های اشتراکی هستند. اگر هرکدام از این پرونده‌ها را به‌عنوان بخشی از نصب اضافه کنید، آن‌ها به صورت اشتراکی نشانه‌گذاری می‌شوند. بنابراین، هنگامی که کاربران برنامهٔ کاربردی را حذف (Uninstal) کنند، پرونده‌های به اشتراک گذاشته شده قبل از حذف، اصلاح می‌شوند. کادر محاوره‌ای نشان داده شده درشکل ۱۴ــ۸ امکان نشانه‌گذاری همهٔ پرونده‌های اشتراکی را فراهم می‌کند. روی Next کلیک کنید.

شکل ۱۴ــ۸ــ پرونده‌های اشتراکی را در این کادر محاوره‌ای نشانه‌گذاری کنید.

۱۲ــ نامی را تعیین کنید (شکل ۱۵ــ ۸) و روی دکمهٔ Finish کلیک کنید تا بستهٔ نرم‌افزاری نصب ایجاد شود.

۱۳ــ بعد از این‌که ویزارد ایجاد بستهٔ نرم‌افزاری نصب را خاتمه داد، گزارشی را با چند پیام مهم تولید می‌کند. گزارش را بخوانید و سپس درگزارش و ویزارد روی Close کلیک کنید.



شکل ۱۵ــ۸ــ آخرین مرحله، ویزارد امکان ارایهٔ نامی برای کاربرد بعدی را فراهم می‌کند.

خودآزمایی و پژوهش

۱ــ پروژه‌ای را به صورت Setup شده به هنر آموز خود تحویل دهید.

۲ــ مقایسه‌ای بین یکی از برنامه‌های

Package and deployment, Setup factory, Install sheild, Wyse installer

از نظر امکانات ساخت برنامه نصب انجام دهید.

۳ــ برنامه نصب پروژه پایانی خود را با استفاده از یکی از سه برنامه معرفی شده در خودآزمایی ۲ به‌صورت setup درآورید.

پروژه ۱: طراحی یک مسابقه

در این مسابقه ۸ سؤال به همراه پاسخ‌هایشان روی فرم وجود دارد. ابتدا سؤال موردنظر را انتخاب می‌کنیم و سپس پاسخ مناسب آن را. اگر پاسخ درست انتخاب شده باشد، سؤال و پاسخ هر دو از روی فرم حذف می‌شود ولی اگر پاسخ اشتباه انتخاب شده باشد، سؤال و پاسخ به همان صورت اولیه باقی می‌ماند. یک زمان‌سنج در پایین فرم وجود دارد تا زمان باقی‌مانده را اعلام کند. اگر به تمامی سؤالات در زمان موردنظر پاسخ درست دهید پیام «آفرین...» می‌دهد واگر زمان به صفر برسد و هنوز سؤال بدون پاسخ مانده باشد پیام «متأسفم ...» می‌دهد.

* ما این مسابقه را با ۸ سؤال طراحی کرده‌ایم ولی می‌توانید آن را به مسابقه‌ای با سؤالات بیش‌تر تعمیم دهید.

کنترل‌های موردنیاز :

۱ــ دو کنترل label برای عنوان‌های «سؤالات» و «نتایج»

۲ــ ۸ کنترل Command برای سؤالات و ۸ کنترل Command برای پاسخ آن‌ها

۳ــ یک کنترل Timer برای نگهداری زمان روی فرم قرار می‌دهیم.

۴ــ یک کنترل label در پایین صفحه برای نمایش زمان باقیمانده.

تحلیل برنامه

وقتی تعدادی کنترل یکسان داریم که مثل هم کار می‌کنند بهتر است آن‌ها را به صورت آرایه کنترلی استفاده کنیم. به این ترتیب آرایه کنترلی برای سؤالات و آرایه کنترلی دیگری برای پاسخ‌ها خواهیم داشت و برای حل بهتر این برنامه، Command های با اندیس یکسان را برای سؤال و پاسخ مرتبط با هم قرار می‌دهیم. فقط دقت کنید چیدمان Command ها طوری باشد که سؤال و جواب مرتبط با هم مقابل یکدیگر نباشند.

طراحی برنامه

یک کنترل Command را روی فرم آورده، ظاهر آن را تنظیم کنید و مشخصهٔ Name آن را cmdq می‌نویسیم و سپس ۷ کپی از آن می‌گیریم و اندیس این آرایه کنترلی صفر تا هفت خواهد بود. (سؤالات)

یک کنترل Command دیگر روی فرم

آورده و مثل قبلی عمل می‌کنیم فقط مشخصهٔ Name آن را CmdA قرار می‌دهیم. (پاسخ‌ها)

یک کنترل Timer را روی فرم آورده و مشخصهٔ Interval آن را 500 می‌دهیم.

یک کنترل label به نام Label1 در پایین فرم قرار داده و مشخصهٔ Caption آن را 60 می‌دهیم.

کدنویسی

۱ــ سؤالات و پاسخ آن‌ها را طرح کرده و روی Commandها قرار می‌دهیم. این عمل را در Form–load انجام می‌دهیم.

```
Private sub Form–load()
k=0
Cmdq(0).Caption = "کنترلی که قادر است تصویر را به اندازه خود درآورد"
Cmdq(1).Caption = "پسوند فایل پروژه چیست"
Cmdq(2).Caption = "کدام کنترل هنگام اجرا دیده نمی‌شود"
Cmdq(3).Caption = "کدام خصوصیت در تمام کنترل‌ها وجود دارد"
Cmdq(4).Caption = "کدام کنترل رویداد ندارد"
Cmdq(5).Caption = "کدام کنترل برای گروه‌بندی کنترل‌هاست"
Cmdq(6).Caption = "تابع چیست"
Cmdq(7).Caption = "یک متد گرافیکی"
CmdA(0).Caption = "ImageImag"
CmdA(1).Caption = "vbp"
CmdA(2).Caption = "Timer"
CmdA(3).Caption = "name"
CmdA(4).Caption = "Shape"
CmdA(5).Caption = "Frame"
CmdA(6).Caption = "روالی که مقداری را بر می‌گرداند"
CmdA(7).Caption = "pset"
end sub
```

۲ــ می‌خواهیم هرگاه روی سؤالی کلیک شد اندیس آن را در جایی نگهداری کنیم تا بتوانیم آن را با اندیس پاسخ مقایسه کنیم.

```
Private sub cmdq−click(index as integer)
        i = index
end sub
private Sub cmdA_Click (index as integer)
        if i = index then
                cmdq(i).visible = false
                cmdA(index).visible = false
                k = k+1
        end if
end sub
```

۳ــ در رویداد CmdA−Click متغیری به نام k می‌بینیم که تعداد پاسخ‌هایی که خصوصیت visible آن False است را می‌شمارد تا هرگاه تعداد آن به ۸ رسید یعنی به همه سؤالات، پاسخ درست داده شده است و این شرط اعلام پیام برنده شدن در مسابقه است که در رویداد Timer از Timer1 قرار می‌دهیم.

```
Private sub Timer1−timer()
        If k = 8 Then
                Timer1. enabled = False
                msgbox (« آفرین… »)
                end
        end if
        If label1. Caption>0 then
                label1.Caption = label1.Caption−1
        else
                Timer1. enabled = False
                msgbox («متأسفم… »)
                end
        end if
```

end sub

۴ــ در این برنامه متغیرهای i و k استفاده شده پس باید این متغیرها را تعریف کنیم برای این کار در قسمت general می‌نویسیم :

Dim I as integer

Dim K as integer

پروژه ۲: PaintBrush

فرمی به شکل زیر ایجاد کنید

کد این پروژه به صورت زیر خواهد بود :

```
Dim IntOldX As Integer, IntOldY As Integer, LngColor As Long
Dim IntLineSize As Byte

Private Sub CboLineSize_Click()
IntLineSize = Val(CboLineSize.Text)
Pic1.DrawWidth = IntLineSize
End Sub

Private Sub CmdClear_Click()
    Pic1.Cls
End Sub

Private Sub CmdExit_Click()
    Unload Me
End Sub

Private Sub CmdPallete_Click()
    CommonDialog1.ShowColor
    LngColor = CommonDialog1.Color
    LBLCurrentColor.BackColor = LngColor
End Sub

Private Sub CmdSave_Click()
    Dim StrFn As String
    With CommonDialog1
    .Filter = "BitMap Format|*.bmp"
```

```
    .ShowSave
    StrFn = UCase(.FileName)
    If StrFn<>"" Then
    SavePicture Pic1. Image, StrFn
    End If
    End With
End Sub

Private Sub Form_Load()
    Dim IntI As Integer
    Pic1.AutoRedraw = True
    Pic1.BackColor = QBColor(15)
    Pic1.ScaleMode = vbPixels
    Pic1. MousePointer = vbCrosshair
    'Set Color Box
    For IntI = 0 To 15
        LBLColor(IntI).BackColor = QBColor(IntI)
    Next
    'Set CurrentColor
    LngColor = 0
    LBLCurrentColor.BackColor = QBColor (LngColor)
    'Fill Draw Width 1 to 5 in ComboBox
    For IntI =1 To 5
        CboLineSize.AddItem IntI
    Next
    'Set DrawWidth to 1
    CboLineSize.ListIndex = 0
```

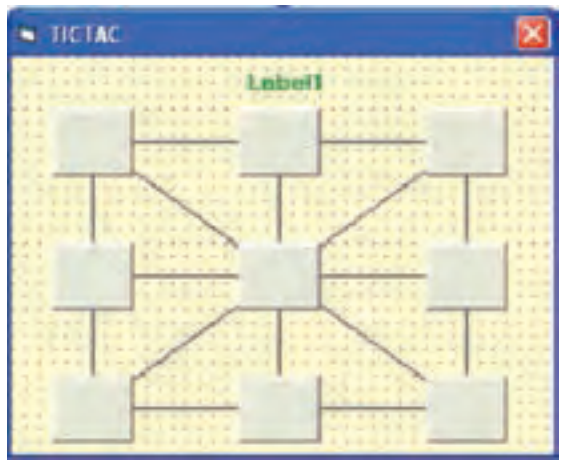
```
    IntLineSize = Val(CboLineSize.Text)
    Pic1.DrawWidth = IntLineSize
    'Set Drawing Line Option
    OptLine.Value = True
End Sub

Private Sub LBLColor_Click(Index As Integer)
    LngColor = LBLColor(Index).BackColor
    LBLCurrentColor.BackColor=LngColor
End Sub

Private Sub Picl_MouseDown(Button As Integer, Shift As Integer,X As Single,
Y As Single)
    IntOldX = X
    IntOldY= Y
End Sub
Private Sub Picl_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
Dim IntI As Integer, IntR As Integer, IntA As Integer
Dim SngPI As Single
    SngPI = 4* Atn(1)
    Randomize Timer
    If Button = 1 Then
        If OptLine.Value = True Then
            Pic1.Line(IntOldX,IntOldY)-(X,Y),LngColor
            IntOldX = X
            IntOldY = Y
```

```
        ElseIf OptSpray.Value = True Then
            For IntI= 1 To 10
                IntA = Rnd*2*SngPI
                IntR = Rnd*15
                Pic1.PSet (X+IntR*Cos(IntA), Y+IntR*Sin(IntA)),LngColor
            Next
        End If
    End If
End Sub
```

پروژه ۳: بازی XO




```
Dim StrPlayer As String

Private Sub CLEAR()
Dim Inti As Variant
For Each Inti In Controls
    If TypeOf Inti Is CommandButton Then
```

```
        Inti.Caption = ""
    End If
    If TypeOf Inti Is Line Then
        Inti. BorderWidth = 1
        Inti.BorderColor = RGB(0, 0, 0)
    End If
Next Inti
End Sub

Private Sub CHECK()
a = cmd(0).Caption + cmd(1).Caption + cmd(2).Caption
If a ="XXX" Then Line1.Border Width = 3: Line1.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line1.Border Width = 3: Line1.BorderColor = RGB(0, 255,
0):
MsgBox "Playaer O Win": CLEAR

a = cmd(3).Caption + cmd(4).Caption + cmd(5).Caption
If a ="XXX" Then Line2.Border Width = 3: Line2.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line2.Border Width = 3: Line2.BorderColor = RGB(0, 255,
0):
MsgBox "Playaer O Win": CLEAR

a = cmd(6).Caption + cmd(7).Caption + cmd(8).Caption
```

```
If a ="XXX" Then Line3.Border Width = 3: Line3.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line3.Border Width = 3: Line3.BorderColor = RGB(0, 255,
0):
MsgBox "Playaer O Win": CLEAR

a = cmd(0).Caption + cmd(3).Caption + cmd(6).Caption
If a ="XXX" Then Line4.BorderWidth = 3: Line4.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line4.BorderWidth = 3: Line4.BorderColor = RGB(0, 255, 0):
MsgBox "Playaer O Win": CLEAR

a = cmd(1).Caption + cmd(4).Caption + cmd(7).Caption
If a ="XXX" Then Line5.Border Width = 3: Line5.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line5.BorderWidth = 3: Line5.BorderColor = RGB(0, 255,
0):
MsgBox "Playaer O Win": CLEAR

a = cmd(2).Caption+cmd(5).Caption + cmd(8).Caption
If a ="XXX" Then Line6.BorderWidth = 3: Line6.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line6.BorderWidth = 3: Line6.BorderColor = RGB(0, 255,
```

```
0):
MsgBox "Playaer O Win": CLEAR

a = cmd(0).Caption + cmd(4).Caption + cmd(8).Caption
If a ="XXX" Then Line7.BorderWidth = 3:  Line7.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line7.BorderWidth = 3: Line7.BorderColor = RGB(0, 255,
0):
MsgBox "Playaer O Win": CLEAR

a = cmd(2).Caption + cmd(4).Caption + cmd(6).Caption
If a ="XXX" Then Line8.BorderWidth = 3:  Line8.BorderColor = RGB(255, 0,
0):
MsgBox "Playaer X Win": CLEAR
If a ="OOO" Then Line8.BorderWidth = 3:  Line8.BorderColor = RGB(0, 255,
0):
MsgBox "Playaer O Win": CLEAR
End Sub

Private Sub cmd_Click(Index As Integer)
If cmd(Index).Caption = "" Then
    cmd(Index).Caption = StrPlayer
    If StrPlayer = "X" Then
        StrPlayer = "O"
        LBL1.Caption = "PLAYER2 O"
    Else
```

```
        StrPlayer = "X"
        LBL1.Caption ="PLAYER1 X"
    End If
CHECK
End If
End Sub

Private Sub Form_Load()
StrPlayer = "X"
LBL1.Caption ="PLAYER1 X"
CLEAR
End Sub
```

توضیحات این پروژه را به عهدهٔ هنرجویان واگذار می‌کنیم.

پروژه‌های پیشنهادی

۱ــ با کمک و راهنمایی هنرآموز درس پروژه PaintBrush را کامل کنید.

۲ــ بازی منچ را شبیه‌سازی کنید.

۳ــ یک پازل تصویری طراحی و امکان مرتب نمودن آن را فراهم کنید.