

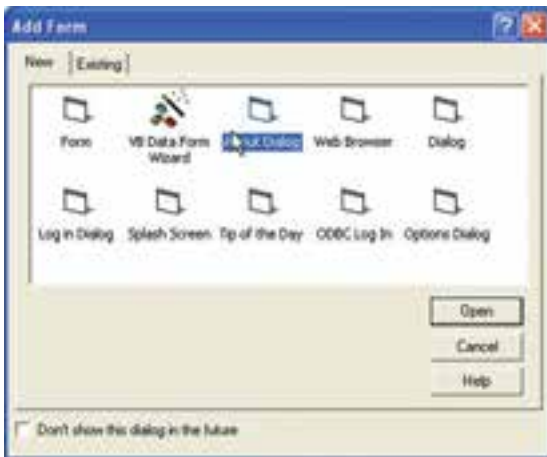
کاربرد فرم‌های آماده

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 **Printer, Color, File** و غیره در برنامه‌های خود استفاده کند؛
- مفهوم رابط چند سندی را شرح دهد؛
- پروژه‌هایی ایجاد کند که از رابط چندسندی استفاده می‌کنند.

۱-۵- 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 آماده‌ی ویژوال بیسیک، می‌توان برای دسترسی به داده‌ها، کادر **About**، صفحه‌های الگو و کادرهای ورود، استفاده کرد.

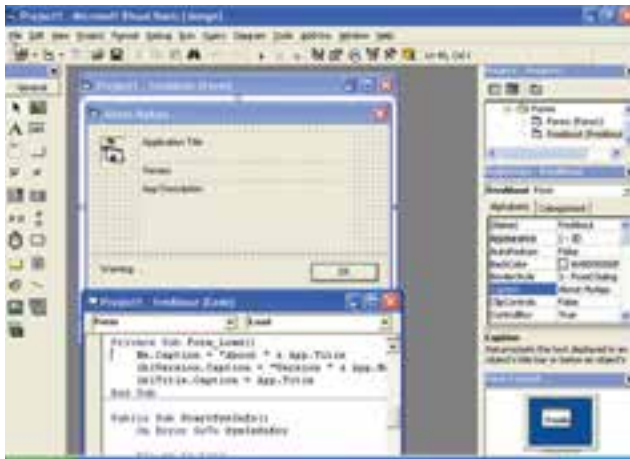
از منوی **Project** گزینه **Add Form** را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود. در این کادر محاوره‌ای، انواع مختلف فرم‌های آماده وجود دارد. فرم مورد نظر را انتخاب کنید (شکل ۱-۵).

شکل ۱-۵- 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 ۵-۲ یک فرم About Dialog را نشان می‌دهد). نه تنها ویژگی‌های 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 ارائه می‌کند که نام برنامه 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 به پروژه جاری و کنترل Me به فرم جاری اشاره می‌کند. کادر محاوره‌ای About کل کد و فراخوانی API^۲ ویندوز را، که برای گزارش اطلاعات سیستم کاربر مورد نیاز است، ارائه می‌کند. (در صورتی که می‌خواهید کد اطلاعات سیستم را مرور کنید، فرم About Dialog را به پروژه جدیدی اضافه کنید و کد تحت دکمه System Info را مشاهده کنید.)



شکل ۵-۲ کادر محاوره‌ای About که به همراه ویژگی‌های بیسیک ارائه می‌شود کدی را که نام برنامه و اطلاعات نسخه و به همین ترتیب اطلاعات سیستم کاربر را گزارش می‌کند، ارائه می‌دهد.

۱- در فصل ۸ توضیح آن آمده است.

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

۲-۵- 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 ارائه می‌کند که به سادگی می‌توان به کمک آن کادرهای محاوره‌ای آماده را برای به دست آوردن اطلاعات کاربر، نمایش داد. این کادرهای محاوره‌ای برای کاربران آشنا هستند و همان‌هایی هستند که خود ویندوز به کار 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 ذخیره اطلاعات را فراهم می‌کند.

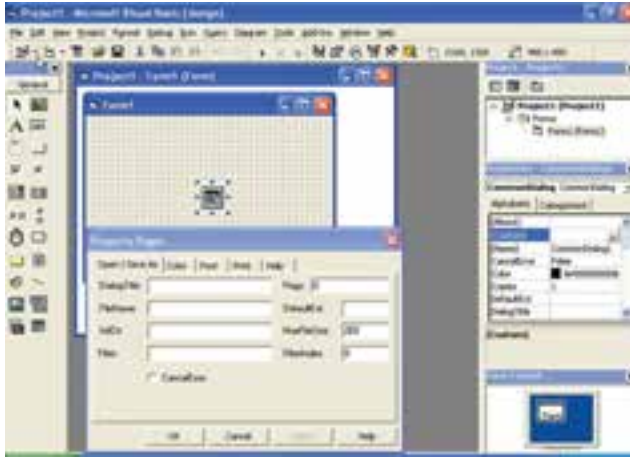
● **Font**، امکان انتخاب قلم و تنظیم مشخصه‌های قلم مورد نظر را فراهم می‌کند.

● **Color**، امکان انتخاب رنگ یا ایجاد رنگ سفارشی برای استفاده در برنامه را ارائه می‌کند.

● **Print**، امکان انتخاب چاپگر و تنظیم چندین پارامتر چاپگر را فراهم می‌کند.

برای استفاده از کنترل Common Dialog، ابتدا باید آن را با انتخاب (Microsoft Common Dialog Control 6.0) از کادر محاوره‌ای Components (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، هر کدام از کادرهای محاوره‌ای مربوط به این کنترل را شرح می‌دهیم. برای هر کادر محاوره‌ای، باید مشخصه‌های کنترل را از طریق پنجره Properties یا کادر محاوره‌ای Property Pages تنظیم کنید. کادر محاوره‌ای Property Pages ساده به مشخصه‌های مورد نیاز برای هر نوع کادر محاوره‌ای را ارائه می‌کند (شکل ۳-۵). برای دسترسی به این کادر محاوره‌ای، در پنجره Properties روی سه نقطه مقابل مشخصه Custom مربوط به کنترل Common Dialog کلیک کنید.



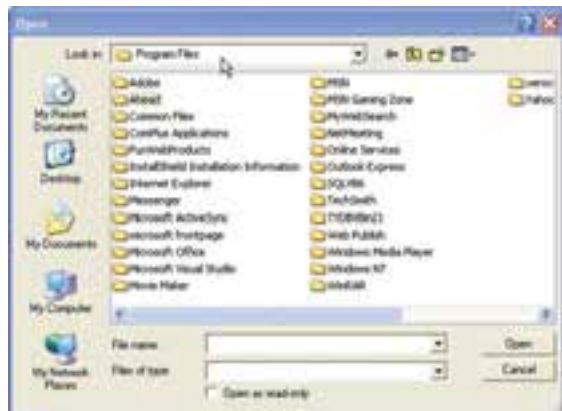
شکل ۳-۵. کنترل Common Dialog در زمان طراحی قابل رؤیت بوده و می‌تواند با استفاده از Property Pages پیکربندی شود.

۳-۵. بازیابی و ذخیره اطلاعات پرونده

کاربرد اصلی کنترل Common Dialog فراهم کردن امکان انتخاب نام پرونده به دو منظور باز کردن و ذخیره کردن پرونده است. حالت باز کردن پرونده، به کاربر امکان تعیین نام پرونده برای بازیابی و استفاده در برنامه را می‌دهد. حالت ذخیره پرونده به کاربر امکان می‌دهد تا نامی را برای ذخیره اطلاعات تحت یک پرونده، فراهم کند. شکل ۴-۵ 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 ۴-۵ را نشان می‌دهد.) از این متد، به صورت زیر استفاده کنید:

CommonDlg1.ShowOpen



شکل ۴-۵. Common Dialog به کادرهای محاوره‌ای پرونده‌ای ویندوز دسترسی دارد.

برای بازکردن کادر محاوره‌ای Save As باید از مند ShowSave استفاده کرد. برای محدود کردن نوع پرونده‌ها مثل پرونده‌های متنی یا سند در کادرهای محاوره‌ای پرونده‌ای، از مشخصه 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. در صورتی که این کار را انجام دهید، ممکن است لیست فایل مورد نظر را به دست 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 *.* به کار ببرید.

اگر از دستور انتساب فوق استفاده کنید، باید فیلتر را داخل زوج کوتیشن (" ") قرار دهید ولی در صورت استفاده از کادر محاوره‌ای Property Pages نیازی به این کار نیست. می‌توان چندین فیلتر را با هم به کار برد و بین این فیلترهای مختلف، باید از علامت پایپ استفاده کرد. مثال زیر را در نظر بگیرید:

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

کادر ترکیبی File of Type در شکل ۴-۵ که به کنترل Common Dialog اعمال شده است نشان می‌دهد. بعد از تنظیم فیلتر، از مشخصه Filename کنترل Common Dialog برای بازیابی نام فایل که کاربر انتخاب کرده است، استفاده کنید:

MyFileName\$ Commondlg1.FileName

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

- ۱- پروژه جدیدی را شروع کنید.
- ۲- کنترل Common Dialog را همان‌طور که قبلاً شرح داده شد، به جعبه ابزار اضافه کنید.
- ۳- روی فرم کنترل‌های label، Command Button و Common Dialog اضافه کنید.
- ۴- کنترل برجسب را در بالای کنترل دکمه فرمان قرار دهید. هر دو کنترل را به گوشه چپ بالای فرم، منتقل کنید.
- ۵- روی دکمه فرمان، دوبار کلیک کنید تا روال رویداد Command Click ایجاد شود.
- ۶- کد زیر را به این روال اضافه کنید:

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

عملکرد این کد به صورت 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 روی Open کلیک کنند یا هیچ پرونده‌ای را انتخاب نکرده و روی Cancel کلیک کنند. براساس این انتخاب‌ها، ویژوال بیسیک مقداری را در مشخصه FileName کنترل Common Dialog قرار می‌دهد.

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

۴-۵- 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 مشخصه Flags است. این مشخصه به کنترل Common Dialog بیان می‌کند که آیا می‌خواهید قلم‌های صفحه نمایش، قلم‌های چاپگر و یا هر دو را نمایش دهید. مشخصه Flags می‌تواند یکی از سه ثابت زیر را داشته باشد:

جدول ۱-۵- مقادیر مشخصه Flags در کادر محاوره‌ای Font

مقدار	ثابت	مجموعه قلم
1	cdICFScreenFonts	قلم‌های صفحه نمایش
2	cdICFPrinterFonts	قلم‌های چاپگر
3	cdCFBoth	هر دو مجموعه



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

نتیجه

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

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

اطلاعات قلم‌های انتخاب شده در مشخصه‌های کنترل قرار داده می‌شود. جدول ۲-۵ مشخصه‌های کنترل و صفات قلم را نشان می‌دهد.

جدول ۲-۵- مشخصه‌های کنترل که صفات قلم را ذخیره می‌کنند

مشخصه	صفت
FontName	نام قلم
FontSize	اندازه قلم برحسب پوینت
FontBold	آیا ضخیم بودن قلم انتخاب شده است؟
FontItalic	آیا کج بودن 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 به دست آمده و برای تغییر قلم‌ها در کادر متن `txtsample` استفاده می‌شود.

`Commondlg1` is the name of a common dialog

`Commondlg1.ShowFont`

`txtSample.FontName Commondlg1.FontName`

`txtSample.FontSize Commondlg1.FontSize`

`txtSample.FontBold Commondlg1.FontBold`

`txtSample.FontItalic Commondlg1.FontItalic`

`txtSample.FontUnderline Commondlg1.FontUnderline`

`txtSample.FontStrikethru Commondlg1.FontStrikethru`

تمرین ۱-۵: برنامه‌ای بنویسید که دارای کادر متن و دکمه انتخاب فونت باشد و کاربر بتواند نوع و اندازه و Style فونت کادر متن را تعیین کند.

۵-۵- انتخاب رنگ‌ها

کادر محاوره‌ای Color امکان انتخاب رنگ‌هایی برای رویه یا زمینه فرم‌ها یا کنترل‌ها فراهم می‌کند (شکل ۶-۵). کاربران می‌توانند یک رنگ استاندارد را انتخاب کرده یا رنگ سفارشی را ایجاد و انتخاب کنند.



شکل ۶-۵: کادر محاوره‌ای Color امکان انتخاب رنگی برای استفاده در برنامه را به کاربران می‌دهد.

برای فعال کردن کادر محاوره‌ای Color در کنترل Common Dialog، مشخصه 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 مشخصه Color کنترل، ذخیره می‌شود. کد زیر، چگونگی تغییر رنگ زمینه فرم را نشان می‌دهد:

```
CommonDlg.Flags cdLccRGBInit
```

```
CommonDlg.ShowColor
```

تمرین ۲-۵: به مثال ۱-۵ دکمه‌ای برای انتخاب رنگ متن در TextBox

اضافه کنید.

۶-۵- تنظیم گزینه‌های چاپگر

کادر محاوره‌ای Print به کاربران امکان انتخاب چاپگر و تعیین گزینه‌ها برای فرآیند چاپ را ارائه 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 ۷-۵). 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^۱ استفاده کرد.

برنامه‌های کاربردی MDI دارای ظاهر یکسان و هماهنگ بوده و کار کردن با آن‌ها نسبت به برنامه‌های SDI (تک‌سندی) 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.

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



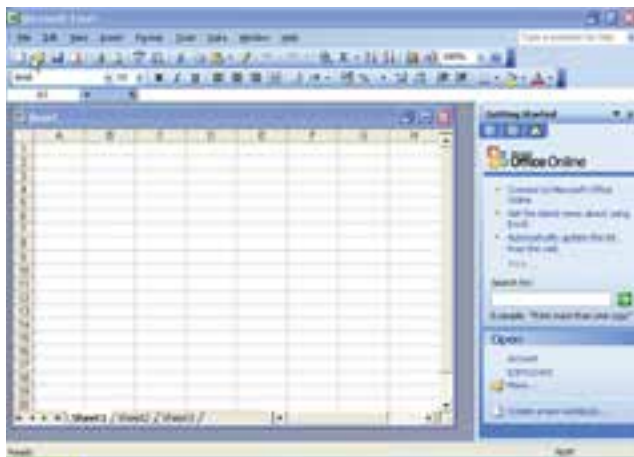
شکل ۷-۵- کادر محاوره‌ای print امکان انتخاب چاپگر و تنظیمات آن را به کاربر می‌دهد.

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

۷-۵- ایجاد برنامه کاربردی چندسندی ساده

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



شکل ۸-۵- Microsoft Excel مثالی از یک برنامه کاربردی MDI است.

ویژوال بیسیک یک شیء MDI به نام MDIForm تعریف کرده است. یک برنامه کاربردی در VB می‌تواند فقط شامل یک شیء MDIForm باشد ولی می‌تواند چندین فرم دیگر داشته باشد که بعضی از آن‌ها فرزند شیء MDIForm بوده و بعضی دیگر پنجره‌های مستقل هستند. پنجره‌های فرزند یک MDIForm منوی خاص خودشان را نمی‌توانند داشته باشند و به جای آن، پنجره‌های فرزند به وسیله 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 پنجره پدر در محلی از منوی پدر، ظاهر می‌شود.

نگاهی

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 ۲-۵

- ۱- پروژه جدیدی را شروع کنید و نام آن را SimplMDI.VBP قرار دهید.
- ۲- فرم پیش‌فرض پروژه را به frmChild تغییر نام دهید. پرونده فرم را با همین نام ذخیره کنید.
- ۳- از منوی Project گزینه 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.
- ۴- مشخصه MDIchild فرم frmChild را با true مقداردهی کنید تا به‌عنوان فرم فرزند mdiMain محسوب شود.
- ۵- کلیدهای Ctrl E را فشار دهید تا Menu Editor باز شود.
- ۶- مانند شکل ۹-۵ منویی را برای فرم mdiMain ایجاد کنید. مقادیر مشخصه Name و Caption منو و گزینه‌های منو را مطابق جدول ۳-۵ تعیین کنید. مطمئن شوید که کادر علامت WindowList برای mnuWindow انتخاب شده باشد.
- ۷- از منوی Project گزینه SimplMDI Properties را انتخاب کنید. در صفحه General کادر محاوره‌ای Project Properties، از لیست بازشوی Startup Object گزینه mdiMain را انتخاب کنید (شکل ۱۰-۵).



شکل ۹-۵- منوهایی را با استفاده از Menu Editor برای یک برنامه کاربردی MDI ایجاد کنید.

جدول ۳-۵- 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 ۱۰-۵- فرم MDI را به عنوان آغازین، تعیین کنید.

۸- کد زیر را برای گزینه‌های منوی برنامه کاربردی بنویسید :

```
Private Sub itmExit Click()
```

```
End
```

```
End Sub
```

```
Private Sub itmAdd Click()
```

```
Dim NewForm As New frmChild
```

ایجاد فرم جدید به صورت پویا :

```
Dim FormNum As Integer
```

Load NewForm

FormNum Forms.Count 1 : تعیین تعداد فرم‌های Child

NewForm.Caption "I am MDI child: " CStr(FormNum)

EndSub

Private Sub itmCasCade Click()

mdiMain. Arrange vbCasCade

EndSub

Private Sub itmTile Clie()

mdiMain. Arrange vbTile Horizontal

EndSub



شکل ۱۱-۵- پروژة SimplMDI فرم‌های فرزند داخل فرم MDI را نشان می‌دهد.

۹- 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 رشته مربوطه اضافه می‌شود تا نشان دهد که چندمین فرم فرزند است. این شماره از مشخصه Forms.Count به‌دست می‌آید و برای این که تعداد فرم‌های فرزند محاسبه شوند، فرم MDI باید از شمارش، کسر شود. به همین دلیل، شماره فرم از تفریق تعداد فرم‌ها و عدد ۱ به‌دست می‌آید.

در این برنامه کاربردی، از متد Arrange شیء MDIForm استفاده شده است. این متد، به طور خودکار پنجره‌های فرزند را داخل پنجره MDI قرار می‌دهد. متد Arrange دارای یک آرگومان است که می‌تواند یکی از مقادیر جدول ۴-۵ را داشته باشد.

جدول ۴-۵- تنظیمات آرایش پنجره‌ها

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

۸-۵- مشخصه‌های Appearance

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

مشخصه AutoShowChildren: اگر مشخصه 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. به طور 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.



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

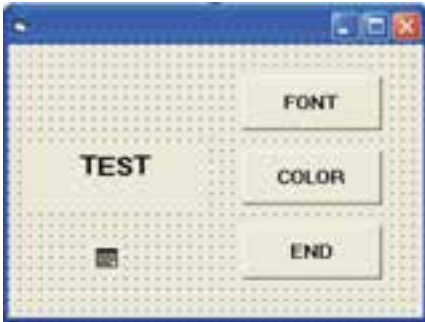
AutoShowChildren را با False مقداردهی کنید و از متد Show استفاده کنید. نخست داده‌ها را در فرم بارگذاری کنید و مطمئن شوید که همهٔ داده‌ها به درستی بارگذاری شده‌اند. در پایان، فرم را نشان دهید. اگر فرم به تجمع داده‌ها وابستگی دارد، این عمل بالاترین قابلیت را ارایه می‌کند.

۱- 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 ۱۲-۵

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



شکل ۱۳-۵

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



شکل ۱۴-۵

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.

۱-۶- آشنایی با Serial Time

ویژوال بیسیک از نوع داده Date استفاده می‌کند که روز را به‌عنوان واحد اصلی زمان به‌کار می‌برد. بنابراین، یک ساعت، $\frac{1}{24}$ و یک ثانیه $\frac{1}{86400}$ روز است. یک هفته را به‌صورت ۷ نمایش می‌دهیم، زیرا یک هفته شامل هفت روز است. نوع داده Date تاریخ را مطابق با قالب زمان کامپیوتر، نمایش می‌دهد؛ یعنی براساس قالب ۱۲ ساعته یا ۲۴ ساعته.

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

تاریخ‌های قبل از ۳۱ دسامبر ۱۸۹۹، با اعداد منفی نمایش داده می‌شوند. به‌عنوان مثال، ۴ جولای سال ۱۷۷۶، روز ۴۵۱۰۳ است یعنی ۴۵۱۰۳ روز قبل از ۳۰ دسامبر ۱۸۹۹.

به محور زیر توجه کنید:



۱- ماه قبل از ژانویه

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

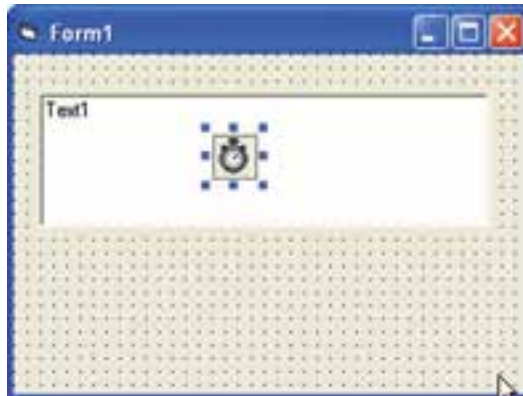
```
Dim MyDate As Date
```

```
MyDate #May 15, 1998#
```

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

۶-۲- 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 فاصله زمانی که تعیین کرده‌اید اجرا می‌کند (شکل ۶-۱). رویداد Timer فراخوانی شده و روال رویدادی که برای TimerName Timer() برنامه نویسی کرده‌اید اجرا می‌شود. Timer Name مقدار مشخصه Name کنترل Timer است.



شکل ۶-۱- کنترل Timer در زمان اجرا قابل رؤیت نخواهد بود.

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

$$24 \times 60 \times 60 = 86400$$

$$86400 \times 2 = 172800 \Rightarrow 2: 57: 46 \text{ Am}$$

$$86400 \times 60 = 5184000$$

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

```
Timer.Interval = 500
```

نکته

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

فاصله زمانی می‌تواند خیلی کوچک مثل هزارم ثانیه، و یا خیلی بزرگ باشد. کوتاه‌ترین فاصله زمانی، ۵۵ میلی‌ثانیه است، زیرا ساعت سیستم در هر ثانیه فقط ۱۸ بار پالس تولید می‌کند. به دلیل این که کنترل Timer رویداد Timer را اجرا می‌کند، دارای مشخصه‌های زیادی نیست (جدول ۱-۶).

جدول ۱-۶ - مشخصه‌های کنترل Timer

مشخصه	شرح
Name	نام کنترل اگر یک کنترل Timer داشته باشید، پیش‌فرض Timer 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 به عنوان عنصری در آرایه‌ای از کنترل‌های Timer باشد تعیین می‌کند
Interval	فاصله زمانی را که رویداد Timer اجرا خواهد شد تعیین می‌کند (برحسب میلی‌ثانیه)
Left	موقعیت گوشه سمت چپ کنترل Timer را تعیین می‌کند
Tag	شبهه یک متغیر درونی در کنترل است که داده‌های مورد نیاز را ذخیره کرده و تا پایان برنامه به صورت پایدار نگه‌داری می‌کند
Top	موقعیت گوشه بالای کنترل Timer را تعیین می‌کند

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

مثال ۱-۶

فرمی طراحی کنید که دارای یک برجسب باشد و با اجرای برنامه این برجسب از سمت چپ به سمت راست فرم حرکت کند و با رسیدن به انتهای سمت راست از حرکت 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 'متوقف کردن حرکت برجسب در انتهای سمت راست
```

```
    End If
```

```
End Sub
```

تمرین: در مثال ۱-۶، دو دکمه فرمان به نام `Start` و `End` قرار دهید که با کلیک روی `Start` حرکت شروع و با کلیک روی `End` خاتمه یابد.

۶-۳- کاربرد توابع `Date`، `Time` و `Now`

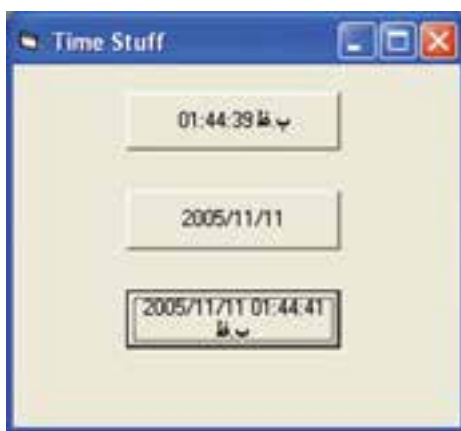
اگرچه می‌توان برای رویداد `Timer` برنامه‌ای نوشت، ولی نمی‌توان زمان اجرا را با این کنترل تنظیم کرد. برای این که `Timer` زمان را گزارش کند باید ساعت سیستم را بررسی کند.

از تابع Time برای به دست آوردن زمان سیستم و از تابع Date برای به دست آوردن تاریخ سیستم استفاده کنید. در صورتی که می خواهید هم زمان و هم تاریخ را از ساعت سیستم به دست آورید، از تابع Now استفاده کنید.

مثال ۲-۶

برای مشاهده چگونگی عملکرد این توابع، پروژه‌ای را ایجاد کرده و سه دکمه 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 ۲-۴ 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 ۲-۶- توابع Time، Date، و Now یک نوع داده Variant (Date) را برمی‌گردانند که با تابع CStr() به رشته تبدیل می‌شوند.

استفاده از **Timer** برای ایجاد یک برنامه 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:

- ۱- پروژه Standard EXE جدیدی را شروع کنید.
- ۲- 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).
- ۳- نام فرم را `frmClock` و نام برجسب را `lblTimeDisplay` قرار دهید. نام کنترل `Timer` را تغییر ندهید (همان نام پیش فرض `Timer1` را حفظ کنید).
- ۴- مشخصه `BorderStyle` فرم را به `Fixed single` تغییر دهید. مشخصه `MinButton` فرم را `True` قرار دهید.
- ۵- 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
    lblTimeDisplay.Height = frmClock.ScaleHeight
```

```
End Sub
```

خطوط ۲ تا ۵ برای بزرگ کردن برجسب به اندازه فرم است. برای تعیین محل و ابعاد کنترل‌ها از مشخصه‌های `Height`، `width`، `left`، `Top` استفاده می‌کنیم. می‌توان علاوه بر چهار مشخصه فوق از مشخصات `ScaleWidth`، `Scaleleft`، `ScaleTop` و `ScaleHeight` استفاده کرد. `ScaleHeight` و `Scalewidth` به ترتیب پهنا و ارتفاع کنترل را تقسیم‌بندی می‌کنند. برای مثال `Scalewidth = 100` پهناهای کنترل را به 100 قسمت مساوی تقسیم می‌کند و واحد اندازه‌گیری افقی را مشخص می‌نماید به همین ترتیب `ScaleHeight = 50` ارتفاع کنترل را به 50 قسمت مساوی تقسیم کرده و واحد اندازه‌گیری عمودی را مشخص می‌کند. از چهار مشخصه `ScaleHeight`، `ScaleWidth`، `Scaleleft`، `ScaleTop` 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 اندازه فرم باشد یعنی با تغییر اندازه فرم، اندازه برچسب هم تغییر کند. برای مشخص کردن واحد اندازه‌گیری ابعاد کنترل از خصوصیت ScaleMode استفاده می‌کنیم که مقادیر آن در جدول ۲-۶ آمده است.

جدول ۲-۶

شرح	مقادیر	ثابت‌ها
هرگاه حداقل یکی از مشخصات ScaleHeight، ScaleWidth، ScaleTop و ScaleLeft توسط برنامه‌نویس مقداردهی شود		vbUser
Twip (پیش فرض) هر اینچ ۱۴۴ توئپ و هر سانتیمتر ۵۶۷ توئپ است	۱	vbTwips
Point هر اینچ ۷۲ پوینت است	۲	vbPoints
Pixel	۳	vbPixels
Character (هر واحد افقی ۱۲ twips و هر واحد عمودی ۲۴ twips است)	۴	vbCharacters
اینچ	۵	vbInches
میلی‌متر	۶	vbMillimeters
سانتی‌متر	۷	vbCentimeters

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

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

۷- کد مربوط به روال Timer() Timer1 را به صورت زیر بنویسید.

```
Private Sub Timer1_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- برنامه را ذخیره کرده و آن را اجرا کنید (شکل ۳-۶).



شکل ۳-۶- تنظیم مشخصه MaxButton با مقدار False دکمه به حداکثرسانی را غیرفعال می‌کند.

برنامه 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 مشخصه MinButton با True، دکمه به حداقل رسانی را روی نوار عنوان نمایش می‌دهد و امکان به حداقل رسانی فرم را فراهم می‌کند.

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

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

نتیجه

هنگام استفاده از مشخصه Window State ممکن است به طور اتفاقی روی فرمی که Border Style آن Fixed Single است، مشخصه Window State را با Maximized مقداردهی کنید. نتیجه 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 برنامه ساعت، مقدار مشخصه Window State فرم را برای تعیین به حداقل رسانی فرم، بررسی می کند. مشخصه Window State دارای سه مقدار است :

- 0 Normal(vbNormal)
- 1 Minimized (vbMinimized)
- 2 Maximized (vbMaximized)

می توان مقدار این مشخصه را خواند و تعیین کرد که آیا پنجره در اندازه عادی، تمام صفحه یا در نوار وظیفه است. همچنین می توان مشخصه 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.

۴-۶- کاربرد تابع Format()

تابع Format() یکی از توابع قوی ویژوال بیسیک است که امکان کنترل روش نمایش رشته ها را فراهم می کند. این تابع معمولاً برای نمایش مقادیر تاریخ/زمان و اعداد استفاده می شود ولی می تواند رشته ها را نیز به صورت مورد نظر 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 رشته ای است که خروجی تابع در آن قرار می گیرد.
- Format نام تابع است.
- Expression عبارتی است که می خواهیم این تابع آن را به قالب مورد نظر تبدیل کند.
- Format String الگوی رشته ای است که به تابع بیان می کند که می خواهد رشته خروجی به چه صورتی ظاهر شود.

● FirstDayOfWeek یک ثابت اختیاری است که اولین روز هفته را تعیین می کند. پیش فرض، Sunday است ولی اگر می خواهید روز شنبه باشد، باید آن را بنویسید (Saturday).

● First Week Of Year یک عبارت ثابت اختیاری است که اولین هفته سال را تعیین می کند. به طور پیش فرض، اول ژانویه، اولین هفته سال است.

کلید کار کردن با تابع Format()، آشنایی با پارامتر Format String است. جدول ۲-۶ چگونگی استفاده از تنظیمات این تابع را برای تغییر ظاهر رشته های تاریخ و زمان ارایه می کند.

نگه

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

جدول ۲-۶- کاربرد تابع `Format()` برای زمان و تاریخ

نتیجه	مثال	Format - String
Friday, July 24, 1998	<code>Format(36000, "Long Date")</code>	"Long Date"
24 - Jul-98	<code>Format(36000, "Medium Date")</code>	"Medium Date"
7/24/98	<code>Format(36000, "Short Date")</code>	"Short Date"
8: 58: 34 p m	<code>Format(0 874, "Long Time")</code>	"Long Time"
8: 58 p m	<code>Format(0 874, "Medium Time")</code>	"Medium Time"
20: 58	<code>Format(0 874, "Short Time")</code>	"Short Time"

جدول ۳-۶- چگونگی استفاده از تابع `Format()` برای کار کردن با مقادیر عددی برای نمایش به صورت رشته دلخواه را نشان می‌دهد.

جدول ۳-۶- کاربرد تابع `Format()` برای اعداد

Format String	Example	Result
"General Number"	<code>Format(36000, "General Number")</code>	36000
"Currency"	<code>Format(36000, "Currency")</code>	\$36,000.00
"Fixed"	<code>Format(36000, "Fixed")</code>	36000.00
"Standard"	<code>Format(36000, "Standard")</code>	36,000.00
"Percent"	<code>Format(36000, "Percent")</code>	3600000.00%
"Scientific"	<code>Format(36000, "Scientific")</code>	3.60E+04
"Yes/No"	<code>Format(36000, "Yes/No")</code>	Yes
"True/False"	<code>Format(36000, "True/False")</code>	True
"On/Off"	<code>Format(36000, "On/Off ")</code>	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") استفاده کنید که مقدار 235.60 را برمی‌گرداند.

۵-۶ محاسبه 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 فاصله زمانی براساس آن، تفاوت تاریخ را اندازه‌گیری خواهد کرد (جدول ۴-۶).

جدول ۴-۶ 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 هفته 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 رشته پارامتر Interval تعیین می‌شود (جدول ۴-۶).

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 غیراین صورت مقدار False را برمی‌گرداند. به عنوان مثال، IsDate("September 16 1998") و IsDate("#9/16/98#") مقدار True را برمی‌گرداند ولی IsDate("Birthday") مقدار False را برمی‌گرداند.

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 ۴-۶

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 ۴-۶ ایجاد کنید (نام پروژه را DateDiff.vbp قرار دهید).

شکل ۴-۶ 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 دکمه Start Counting کلیک می‌کند. روال رویداد Click مربوط به دکمه، ورودی را بررسی کرده و در صورتی که شبیه یک رشته 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 Timer1_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

```

۶-۶ کاربرد متغیرهای ایستا به همراه Timer

فرض کنید می‌خواهید برنامه‌ای بنویسید که عملی را در هر نیم‌ثانیه تا 10° بار انجام دهد. برای انجام این کار، نیاز دارید متغیری ایجاد کنید که تعداد دفعات اجرای رویداد Timer را نگه دارد. اگر یک متغیر شمارنده داخل رویداد Timer ایجاد کنید، هر بار که روال قطع شود، متغیر از حوزه 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 واژه 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 در نظر گرفتن وضعیت روالی که در آن ایجاد شده است، پایدار باقی می‌ماند.

مثال ۵-۶

فرمی به صورت زیر (شکل ۵-۶) ایجاد کنید. می‌خواهیم با کلیک روی دکمه Next Problem به طور خودکار ۲ عدد تصادفی (بین صفر تا ۲۰) تولید شود و حاصل جمع 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 ۵-۶

کد این برنامه به صورت زیر جمع خواهد بود :

Option Explicit

Dim Sum As Integer

Dim NumProb As Integer, NumRight As Integer

Private Sub cmdExit Click()

End

End Sub

Private Sub cmdNext Click()

Dim Number1 As Integer

Dim Number2 As Integer

txtAnswer.Text ""

lblMessage.Caption ""

Number1 = Int(Rnd * 21)

تولید عدد تصادفی اول بین ۰ و ۲۱

Number2 = Int(Rnd * 21)

تولید عدد تصادفی دوم بین ۰ و ۲۱

LblNum1.Caption = Number1

نمایش عدد اول

LblNum2.Caption = Number2

نمایش عدد دوم

Sum = Number1 + Number2

محاسبه 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 فشردن 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 اضافه کردن به میزان 5

Else

lblMessage.Caption = "Answer is " & Sum نمایش پاسخ صحیح

lblScore.Caption = lblScore.Caption - 5 به روزرسانی امتیاز با کم کردن به میزان 5

End If

cmdNext.Enabled = True

cmdNext.SetFocus

End If

End Sub

تمرین: شمارش معکوس

پروژه‌ای مطابق شکل ۶-۶ ایجاد کنید و یک کنترل Timer و یک برجسب روی

فرم قرار دهید تا شمارش معکوس را شبیه‌سازی کند.



شکل ۶-۶

- ۱- در هر یک از دستورات زیر مقدار ذخیره شده در متغیر strS را مشخص کنید.
 - الف) `strS Format(d, "hampm")`
 - ب) `strS Format(12345.67, "#####.### ")`
- ۲- فرمی تهیه کنید به این شرح که، با انتخاب تاریخ، در قسمت پایین تقویم، تاریخ نمایش داده شود. با کلیک روی دکمه مقایسه، تاریخ با تاریخ روز مقایسه شود و پیغام دهد که آیا تاریخ گذشته یا آینده است؟
- ۳- به مثال ۱۰-۶ دکمه CmdTimer اضافه کنید که با کلیک آن تصاویر به طور خودکار نمایش داده شوند (هر ۱۰ ثانیه یک تصویر) و با کلیک دوباره این دکمه نمایش خودکار متوقف شود.
- ۴- برنامه ای بنویسید که از ۱ تا ۱۰ را بشمارد و فاصله 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.
- ۵- برنامه ای بنویسید که تاریخ تولد دو دانش آموز را دریافت کرده و اختلاف سنی آنها را برحسب ثانیه، روز و سال نمایش دهد.