آموزش
Visual Basic 6.0
(جلد اول)

تهیه و تألیف :
رضا خارائی

خرداد و تیر
۱۳۸۹

تقدیم به همهٔ هم میهنانم

که همیشه دوستشان دارم

و آرزومندم که ایران و ایرانی

همیشه در حال ترقی و پیشرفت باشند...

فهرست مطالب

فهرست مطالب


بسم الله الرحمن الرحيم

پیشگفتار :

با لطف و عنایت خداوند متعال، جلد اول کتاب آموزش ویژوال بیسیک ۶ با توجه به نیاز هموطنان عزیزی که قصد شروع کار در عرصهٔ برنامه نویسی را داشته اند، تهیه شده است. آنچه در این کتاب به نظر شما می رسد، شامل ۹ فصل بوده که دارای مطالب آموزشی، نمونه کد و تمرین می باشد.

این کتاب رایگان است و هرگونه نسخه برداری از مطالب آن به صورت کلی و جزئی مجاز می باشد. این کتاب به این امید تألیف شده که بتواند در ارتقای سطح علمی هموطنان عزیز و علاقه مند به برنامه نویسی مؤثر واقع شود و همانطور که اشاره شد، کپی کردن این جزوه آزاد است، اما تغییر در محتوی آن در انحصار اینجانب می باشد.

جلد دوم این کتاب، شامل مباحث پیشرفتهٔ برنامه نویسی، خواهد بود که این مباحث عبارتند از : گرافیک، چندرسانه ای، کار با API ویندوز، کار با رجیستری، ساخت و استفاده از کلاس ها، ساخت انواع ActiveX، کار با دیتابیسSQL Server، ساخت گزارش با Crystal Report و Socket Programming و... . از این رو، از دانشجویان و استاید محترم تقاضا می شود با ارسال نظرات و پیشنهادات خود در مورد این کتاب و یا اشکالات موجود در محتوی یا متن آن به آدرس kharaee@Gmail.com در بهبود آن سهیم باشند.

توجه داشته باشید که هرگز نمی توانید تنها با خواندن این کتاب و کتابهای آموزشی دیگر، یک برنامه نویس موفق شوید! مگر اینکه علاوه بر علاقه و اطمینان به کارتان، پیوسته تمرین کنید، کدها و پروژه های دیگر را مطالعه کنید که این کار بیشتر از هر کتاب و دوره های آموزشی مفید است و همچنین تحصیلات دانشگاهی تان را ادامه دهید؛ چون درک عمیق تری نسبت به رشته تان به دست خواهید آورد. این دستورالعمل ها، حاصل تجربه و تحقیق بزرگانی چون Peter Norvig (نویسندهٔ کتاب هوش مصنوعی مورد استفاده در اکثر دانشگاه های جهان) بوده که عقیده داشتند حداقل ۱۰ سال لازم است تا یک فرد به تخصص در یک رشته (نه فقط برنامه نویسی) برسد. همچنین، Eric Raymond در کتاب دیکشنری هکرها گفته است : «فقط مطالعه و تحصیلات آکادمیک در رشتهٔ کامپیوتر یک شخص را تبدیل به یک برنامه نویس حرفه ای نمی کند! همانطور که یک فرد فقط با مطالعهٔ رنگ و قلم‌مو نمی تواند نقاش شود.».

در پایان شایسته است، فرصت را مغتنم دانسته و مراتب قدردانی خود را از همراهی و کوشش های تمام عزیزانی که مرا در مراحل زندگی و در زمینهٔ کاری یاری کردند، ابراز نمایم.

رضا خارائی

فصل اول :

آشنایی با ویژوال بیسیک

(Visual Basic)

برنامه نویسی شئ گرا :

برنامه نویسی شیء گرا یعنی Object Oriented Programming یا OOP یک جزء جدایی ناپذیر زبان های برنامه سازی نسل جدید است.

زبان‌های برنامه نویسی شیءگرا، زبان‌هایی هستند که در آن برنامه نویس می‌تواند اشیاء مختلفی را تعریف نماید و از اشیاء تولید شده استفاده نماید. امروزه اکثر زبان‌های دستوری برنامه نویسی از فنون شئ گرایی پشتیبانی می‌کنند.

در شیءگرایی هر شیء (Object) از یک الگو به نام کلاس (Class) درست می شود و شامل دو جزء اصلی است : اعضا (Members) و متدها (Methods). در واقع، در زبان های برنامه سازی شیءگرا شما با الگوها سروکار دارید. یک شیء هنگامی که از یک الگو درست می شود، تمامی اعضا و متدهای آن را به طور کامل دریافت می کند. در تعریف اعضا و متدها باید گفت که اعضا یا Members متغیرهایی هستند که در سراسر کلاس قابل دستیابی هستند و متدها یا Methods قطعه کد هایی بسیار شبیه به تابع هستند که عملیاتی را بر اعضای کلاس با دیگر متغیرها انجام می دهند.

آشنایی با محیط ویژوال بیسیک

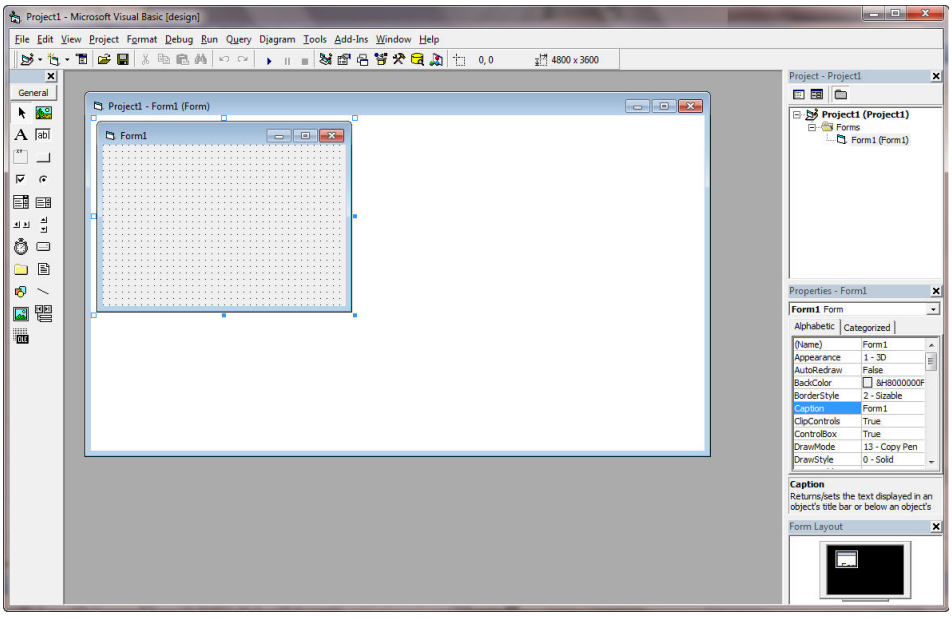
محیط ویژوال بیسیک ساده است. این محیط که یکی از محیط های توسعه یافتهٔ مجتمع یعنی IDE (Integrated Development Environment) است، به برنامه نویسان امکان می دهد که برنامه های تحت ویندوز خود را بدون نیاز به استفاده از برنامه های کاربردی دیگر ایجاد، اجرا و خطایابی کنند.

بعد از اجرای برنامهٔ ویژوال بیسیک (Start->All Programs->Microsoft Visual Studio 6->Microsoft Visual Basic 6.0) کادر محاوره ای New Project به نمایش در می آیدکه این کادر به برنامه نویس امکان انتخاب یکی از انواع برنامه هایی را می دهد که می توان در ویژوال بیسیک ایجاد کرد.

نوع Standard EXE که به طور پیش فرض در این کادر انتخاب شده است، به برنامه نویس امکان می دهد که برنامهٔ اجرایی استانداردی را ایجاد کند. کادر محاوره ای New Project دارای سه زبانه (Tab) است:

- زبانهٔ New : برای ایجاد پروژهٔ جدید
- زبانهٔ Existing : برای باز کردن پروژه ای که از قبل وجود دارد.
- زبانهٔ Recent : لیستی از آخرین پروژه های باز شده یا ایجاد شده را نشان می دهد.

برای باز کردن یک پروژه بر روی آیکن مورد نظر، دابل کلیک کرده یا روی آیکن، کلیک کنید، پس کلید Enter یا دکمهٔ Open را فشار دهید. با باز شدن پروژه کادر محاوره ای بسته شده و وارد محیط IDE می شویم این محیط دارای چندین پنجره، یک نوار منو و یک نوار ابزار است که مشابه نوار منو و ابزار در اکثر برنامه های تحت ویندوز است.

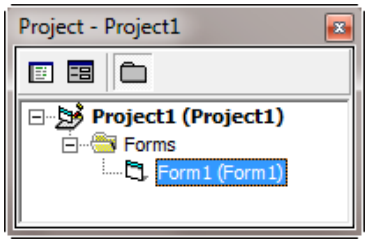


پروژه Standard EXE دارای پنجره های زیر است :

۱. پنجرهٔ پروژه (Project)
۲. پنجرهٔ Form Layout
۳. جعبه ابزار (ToolBox)
۴. پنجرهٔ مشخصه ها (Properties)
۵. پنجرهٔ فرم (Form)

۱. پنجره Project :

پنجره ای است که معمولاً به نام Project Explorer نیز معروف بوده و شامل تمام فایل های مربوط به پروژه است. در صورت عدم مشاهده این پنجره، کلیدهای Ctrl+R را فشار دهید یا از منوی View گزینهٔ Project Explorer را انتخاب کنید.

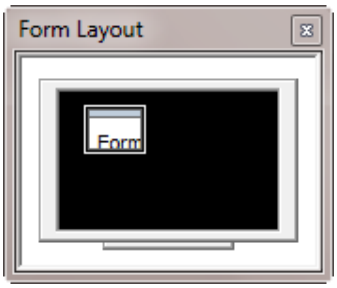


نوار ابزار این پنجره شامل سه دکمه به نام های : ۱- View Code ، ۲- View Object و ۳- Toggle Folders است.

- دکمهٔ View Code پنجره ای را باز می کند که کد (دستورات برنامه) مربوط به پروژهٔ فعال را نمایش می دهد.
- دکمهٔ View Object شکل ظاهری فرم فعال در پنجرهٔ Project را نمایش می دهد.
- دکمهٔ Toggle Folders سبب می شود که با هر بار فشار آن ، پوشهٔ Forms درون این پنجره به صورت متناوب به نمایش درآمده و پنهان شود.

۲. پنجره Form Layout :

این پنجره محل فرم را هنگام اجرای برنامه (Run Time) بر روی صفحهٔ نمایش مشخص می کند. با درگ کردن می توان فرم را در محل جدید خود قرار داد تا در زمان اجرا، در محل مشخص شده ظاهر گردد.



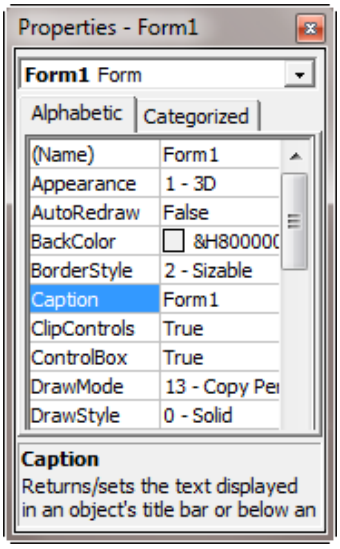
در صورت عدم مشاهده این پنجره، از منوی View گزینهٔ Form Layout Windows را انتخاب کنید.

۳. جعبه ابزار (Tool Box) :

این جعبه شامل شئ هایی است که می توان هر یک از آنها را به تعداد دلخواه بر روی فرم های مربوط به پروژه اضافه کرد. این اشیاء نسبت به نوع پروژه ای که در ابتدای ایجاد پروژهٔ جدید مشخص می شود، متغیر است.

٤. پنجرهٔ مشخصه ها (Properties) :

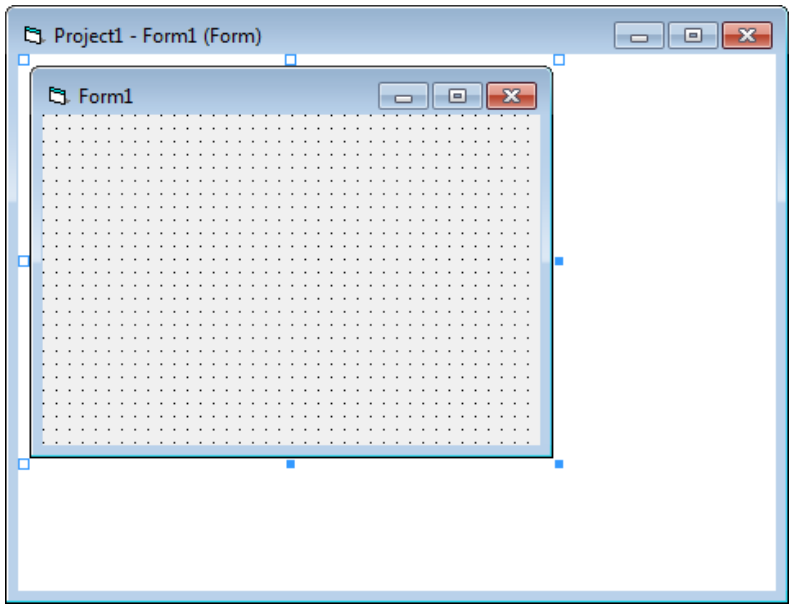
این پنجره ویژگی ها و مشخصه های فرم یا شئ را نشان می دهد. درصورت عدم مشاهدهٔ این پنجره، کلید F4 را فشار دهید یـا از منـوی View گزینهٔ Properties Windows را انتخاب کنید.



همانطور که در شکل مشاهده می کنید در قسمت بالای پنجره، جعبهٔ لیست مانندی وجود دارد که در آن نام شئ یا فرمی که مشخصه های آن در این پنجره آورده شده است، نمایش داده می شود. داخل این لیست نام تمای اشیاء و همچنین نام فرمی که فعال است، آورده شده است. با انتخاب هر شئ یا فرم دیگری از این لیست، مشخصه های مربوط به آن در پنجره نمایش داده می شود.

۵. پنجره فرم ها :

این پنجره، فرم فعال در پنجرهٔ پروژه را با تمام شئ های مربوط به آن ، در یک رابط گرافیکی کاربر (GUI) نشان می دهد.



در ابتدای ایجاد فرم جدید، هیچ شئ ای در آن وجود ندارد. البته در صورتی که دکمهٔ View Code در پنجرهٔ پروژه انتخاب شود یا روی هر شئ یا فرم دابل کلیک شود، پنجرهٔ مربوط به کد، در این قسمت نمایش داده می شود.

انواع فایل ها در ویژوال بیسیک :

۱. **فایل پروژه :** این فایل با پسوند VBP (Visual Basic Project) ذخیره می شود و محتوای آن مشخصات پروژه، نام فایل های فرم و فرم اصلی و... است.

۲. **فایل محیط کاری :** این فایل با پسوند VBW (Visual Basic WorkSpace) ذخیره می شود و محتوای آن، اطلاعات محیط کاری و فرم های پروژه است.

۳. **فایل فرم :** این فایل با پسوند FRM ذخیره می شود و محتوای آن تمام اطلاعات و مشخصات فرم به همراه مشخصات کنترل های موجود در آن است.

٤. **فایل تصاویر :** این فایل با پسوند FRX ذخیره می شود و محتوای تصاویری است که روی فرم یا کنترل های دیگر از آنها استفاده شده است.

۵. **فایل DLL ، OCX** (در فصل نهم با آن آشنا خواهید شد.)

فصل دوم :

داده ها در ویژوال بیسیک

انواع داده ها در ویژوال بیسیک

انواع داده هایی که در ویژوال بیسیک مورد استفاده واقع می شوند به دو دسته کلی تقسیم می گردند :

۱. داده های عددی

۲. داده های غیر عددی

* * *

۱. داده های عددی :

داده های عددی در ویژوال بیسیک می توانند به یکی از صورتهای زیر باشند :

- اعداد صحیح (Integer) : اعدادی هستند که قسمت اعشار ندارند. مانند ۱۲
- اعداد اعشاری (Decimal) : اعدادی هستند که قسمت اعشار دارند. مانند ۱۴,۷

داده های عددی دیگری در ویژوال بیسیک وجود دارند که در جدول زیر مشاهده می کنید :

نوع	مقدار حافظه	محدوده اعداد
Byte	۱ بایت	0 تا 255
Integer	۲ بایت	32768- تا 32767
Long	٤ بایت	تقریباً 2.1E9-/+
Single	۸ بایت	اعداد مثبت : 3.402823E38- تا 1.401298E-45- اعداد منفی : 1.401298E-45 تا 3.402823E38
Double	۸ بایت	اعداد منفی : 1.79769313486232E308- تا 4.94065645841247E- اعداد مثبت : 4.94065645841247E تا 1.79769313486232E308
Currency	۸ بایت	922337203685477.5808- تا 922337203685477 (چهار رقم اعشار برای دقت محاسبات است.)
Decimal	۱۲ بایت	+/- 79228162514264337593543950355 به صورت اعشاری +/- 7.9228162514264337593543950335

همانطور که در جدول فوق ملاحظه می کنید هر نوع داده از نظر تعداد بایت اشغالی با نوع دیگر تفاوت دارد و برنامه نویس برای صرفه جویی در حافظه بهتر است از نوع مناسبی از انواع داده های موجود استفاده نماید. برای مثال برای سابقه کار فرد یا سن آن، بهتر است نوع Byte در نظر گرفته شود.

۲. داده های غیر عددی :

داده های غیر عددی داده هایی هستند که انجام محاسبات ریاضی روی آنها، مورد نظر برنامه نویس نیست و صرفاً می خواهد اطلاعاتی را ذخیره کند و در گزارشات و عملیات منطقی از آنها استفاده نماید.

در جدول زیر داده های غیر عددی نشان داده شده است.

نوع	مقدار حافظه	محدوده
String	طول رشته	از ۱ تا تقریباً ۶۵۴۰۰ نویسه
Date	۸ بایت	از اول ژانویه ۱۰۰ تا ۳۱ دسامبر ۹۹۹۹
Boolean	۲ بایت	True یا False
Variant	طول رشته + ۲۲ بایت	مانند String
Variant	۱٦ بایت	هر عددی تا Double
Object	٤ بایت	معادل شئ تعریف شده

متغیر ها

متغیر (Variable) مکانی در حافظه است که برای نگهداری یک مقدار، مورد استفاده قرار می گیرد. وقتی مقداری را در یک متغیر قرار می دهید، مقدار قبلی آن از بین خواهد رفت. متغیر ها با نامشان شناخته می شوند، بنابراین در یک قسمت از برنامه (روال) استفاده از دو متغیر با یک نام مجاز نیست. متغیر ها بهتر است که قبل از استفاده در برنامه تعریف شوند تا نام و نوع مقداری که می توانند بگیرند مشخص شود. البته ویژوال بیسیک اجازه می دهد که این قاعدهٔ کلی را زیر پا بگذارید!

تعریف متغیر ها

برای اعلان یک متغیر (نام گذاری و تعیین نوع) از کلمهٔ کلیدی Dim استفاده می شود. الزام یا عدم الزام بـه اعـلان متغیرهـا در برنامـه را می توانید با دستور Option Explicit مشخص کنید؛ بدین صورت که اگر این دستور را در بخش General پنجره کدنویسی تایپ کنید، ایـن مسئله الزامی خواهد شد.

شکل کلی اعلان یک متغیر با استفاده از کلمهٔ کلیدی Dim به صورت زیر می باشد :

```
Dim  نام متغیر  As  نوع داده
```

برای مثال، خطوط زیر تعریف متغیرهایی از انواع مختلف را نشان می دهد :

```
Dim A As Integer
Dim B As Long
Dim C As Single
Dim D As Double
Dim E As Currency
Dim F As String
Dim G As Boolean
```

همچنین می توان بعضی از متغیرهای فوق را با استفاده از پسوند نوع داده، تعریف کرد :

```
Dim A%
Dim B&
Dim C!
Dim D#
Dim E@
```

برای تعریف چند متغیر از یک نوع داده نیز می توان به صورت زیر عمل کرد :

```
Dim I,J AS Integer
```

متغیر های تعریف شده در یک روال فقط در همان روال قابل استفاده اند و در هیچ روال دیگـری قابـل دسـترس نیـستند؛ بـه ایـن قبیـل متغیرها، متغیرهای محلی (Local Variable) گفته می شود. متغیرهایی که در قسمت تعاریف ماژول یعنی بخش General تعریف شـوند، در تمام روال های آن ماژول قابل دسترس خواهند بود؛ به این گونه متغیرها، متغیرهای عمومی (Global Variable) می گویند.

نکاتی در رابطه با متغیرها :

۱. نام متغیر باید با یکی از حروف الفبا شروع شود.

۲. استفاده از حروف و اعداد در نام متغیرها مجاز است.

۳. نام یک متغیر می تواند تا ۲۵۵ نویسه طول داشته باشد.

۴. فاصله در نام متغیرها مجاز نیست.

۵. هر چیزی که بین دو علامت "" قرار گیرد، یک رشته است؛ حتی اگر هیچ نویسه ای در آن نباشد.

۶. رشته ای که طول آن صفر باشد، رشتهٔ Null نامیده می شود.

۷. هنگام استفاده از مقادیری که شامل تاریخ و زمان هستند، از علامت # در ابتدا و انتهای این مقادیر، استفاده کنید.

۸. در مقداردهی به متغیرهای منطقی، می توانید از مقادیر Yes و No یا 0 و 1 به جای True و False استفاده کنید.

۹. متغیرهای منطقی در هنگام تعریف به طور خودکار، مقدار False ، متغیرهای تاریخ و ساعت، مقدار 00:00:00 ، متغیرهای عددی، مقدار 0 و متغیرهای رشته ای به صورت یک رشتهٔ خالی ("" یا Null) مقداردهی اولیه می شوند.

۱۰. اگر بخواهیم زمان اعلان یک متغیر از نوع رشته، طول آن ثابت بماند، باید طول آن را نیز مشخص کنیم. برای مثال، در خط زیر متغیر F با طول ثابت ۵ تعریف شده است.

```
Dim F As String * 5
```

حال اگر رشته ای که طول آن بیش از پنج نویسه است، به متغیر F نسبت داده شود، ویژوال بیسیک فقط پنج نویسهٔ اول آن را ذخیره می کند و مابقی را نادیده می گیرد.

مقداردهی به متغیر ها

بعد از اعلان یک متغیر، می توانید با استفاده از دستور انتساب، داده را در آن ذخیره کنید. شکل کلی این دستور چنین است :

```
VarName = Expression
```

که در آن VarName نام یک متغیر بوده و هر چیزی که بتواند یک مقدار تولید کند، عبارت Expression است؛ به عبارت دیگر Expression می تواند یکی از موارد زیر باشد :

- عبارت محاسباتی
- مقدار
- عبارت منطقی یا رشته ای
- مقدار مشخصهٔ یک کنترل
- ترکیبی از عبارت های محاسباتی یا منطقی، متغیرها و مقدار مشخصهٔ کنترل

به مثال های زیر توجه کنید :

```
StrFirstName = "Ali"
A = 123
Count = A + 321
LsDisplay = True
Birthday = #4/1/99#
```

از دستور انتساب برای مقدار دادن به خواص کنترل ها نیز می توان استفاده کرد؛ شکل کلی انجام این کار به صورت زیر است :

```
مقدار = مشخصه.نام کنترل
```

دستورات زیر چند نمونه از مقدار دهی به خواص کنترل ها را با روش انتساب نشان می دهد :

```
Label1.Caption = "Hello World"
Text1.Left = 1024
Command1.Enabled = False
```

اکنون به مثال های زیر توجه کنید :

```
Dim A As String
A = "Hello World"
Label1.Caption = A

Dim B As Integer
B = 1024
Text1.Left = B

Dim C As Boolean
C = False
Command1.Enabled = C
```

دستورات فوق روش تعریف، مقداردهی و استفاده از متغیر ها را نشان می دهد.

ثابت ها (Constants)

متغیرها تنها روش ذخیرهٔ اطلاعات در حافظه نیستند. روش دیگر، استفاده از ثابت هاست. ثابت ها مقادیری هستند که در برنامه ها تعریف می شوند و مورد استفاده قرار می گیرند ولی مقدار آنها در طول برنامه تغییر نمی کند. برای تعریف ثابت ها از کلمهٔ کلیدی Const استفاده می شود :

```
Const نام ثابت As نوع داده = مقدار
```

نام گذاری و روش استفاده از ثابت ها همانند متغیرهاست. به مثال های زیر توجه کنید :

```
Const Num As Integer = 18
Const Name As String = "Reza"
Const Birthday As Date = #17/2/91#
```

عملگرها در ویژوال بیسیک

عملگرها، نمادهایی هستند که اعمالی را روی مقادیر (متغیرها و ثابت‌ها) انجام می دهند. برای مثال عملگر + روی دو عملوند عمل کـرده و آنها را با هم جمع می کند. در ویژوال بیسیک عملگرها به سه دسته تقسیم می شوند :

۱. عملگرهای حسابی

۲. عملگرهای رابطه ای

۳. عملگرهای منطقی

در جدول های زیر انواع عملگرها را به همراه نام و مثال هایی از هر کدام را مشاهده می کنید.

عملگرهاي حسابي		
عملگر	نام	مثال
+	جمع	6 + 2 = 8
-	تفریق	6 – 2 = 4
*	ضرب	6 * 2 = 12
/	تقسیم	6 / 2 = 3
^	توان	6 ^ 2 = 36
Mod	باقیمانده تقسیم	6 Mod 2 = 0
\	تقسیم صحیح	7 \ 3 = 2
+,&	پیوند رشته ها	"Ali" & "Reza" = "AliReza"

عملگرهاي رابطه اي		
عملگر	نام	مثال
>	کوچکتر	6 > 2
=>	کوچکتر یا مساوي	2 => 2
<	بزرگتر	2 < 6
=<	بزرگتر یا مساوي	6 =< 6
<>	نامساوي	2 <> 6

عملگرهاي منطقي		
عملگر	نام	مثال
NOT	نقيض	NOT X
AND	و	X AND Y
OR	يا	X OR Y
XOR	ياي انحصاري	X XOR Y
EQV	هم ارزي	X EQV Y
IMP	مشمول	X IMP Y

جدول درستي عملگرهاي منطقي							
X IMP Y	X EQV Y	X XOR Y	X OR Y	X AND Y	NOT X	Y	X
T	T	F	T	T	F	T	T
F	F	T	T	F	F	F	T
T	F	T	T	F	T	T	F
F	T	F	F	F	T	F	F

عملگرهای منطقی اگر X,Y دارای ارزش درستی یا نادرستی باشند آنگاه عملگرهای مذکور با توجه به درستی و یـا نادرسـتی عملونـدهای X,Y یا درست خواهند بود و یا نادرست.

عملگر NOT : این عملگر نقیض عملوندش است. مثلا اگر X درست (T) باشد آنگاه NOT X دارای ارزش نادرستی (F) است و برعکس.

عملگر AND : نتیجه این عملگر وقتی درست است که هر دو عملوندش درست باشد.

عملگر OR : نتیجه این عملگر وقتی درست است که حداقل یکی از عملوندها دارای ارزش درستی باشد.

عملگر XOR : نتیجه این عملگر وقتی درست است که فقط یکی از عملوندهایش درست باشد.

عملگر EQV : نتیجه این عملگر وقتی درست است که دو عملوند دارای ارزش یکسانی باشند.

عملگر IMP : نتیجه این عملگر وقتی درست است که عملگر سمت چپ درست باشد.

تقدم عملگرد ها

ویژوال بیسیک اعمال ریاضی را به ترتیب خاصی انجام می دهد. جدول زیر ترتیب انجام این عمل را نشان می دهد :

ترتیب	عملگر
۱	^
۲	- (تفریق یکانی)
۳	* ، /
٤	\
٥	Mod
٦	+ ، -

اگر در یک عبارت، عملگرهایی با تقدم یکسان وجود داشته باشند، ویژوال بیسیک محاسبات را از چپ به راست انجام می دهد. به مثال زیر توجه کنید :

```
10 / 2 * 3
```

در این عبارت به دلیل اینکه ضرب و تقسیم دارای تقدم یکسان هستند، ویژوال بیسیک ابتدا تقسیم را انجام داده و سپس حاصل تقسیم را در ۳ ضرب خواهد کرد؛ بدین ترتیب حاصل عبارت فوق ۱۵ خواهد شد.

اگر می خواهید ترتیب انجام محاسبات را تغییر دهید، باید از پرانتز استفاده کنید. به مثال زیر توجه کنید :

```
30 / (2 * 3)
```

در این عبارت ابتدا ۲ در ۳ شرب شده و سپس عدد ۳۰ بر حاصل آن تقسیم می شود. بدین ترتیب حاصل عبارت فوق ۵ خواهد شد.

در پرانتز های تودرتو، ویژوال بیسیک از داخلی ترین پرانتز شروع کرده و رو به بیرون حرکت می کند. به مثال زیر توجه کنید :

```
30 / (2 * (2 + 1))
```

ویژوال بیسیک برای محاسبه عبارت فوق، قبل از هر کاری (۱ + ۲) را محاسبه خواهد کرد.

فصل سوم :

دستورات شرطی در ویژوال بیسیک

قبل از آنکه دستورات شرطی را توضیح دهیم، دو تابع متداول در ویژوال بیسیک را جهت یادآوری بیان می کنیم :

* * *

تابع MsgBox() : یک کادر پیام را به کاربر نشان می هد. این تابع یک آرگومان اجباری (Prompt) و دو آرگومان اختیاری (Style و Title) دارد. آرگومان اول پیامی است که باید کادر پیام نشان دهد و آرگومان دوم تعداد و نوع دکمه های کادر پیام را تعیین خواهد کرد و آرگومان سوم عنوان پنجره کادر پیام خواهد بود. مقدار برگشتی تابع MsgBox() دکمه ای است که کاربر با کلیک کردن روی آن کادر پیام را بسته است و برنامه می تواند با کنترل این مقدار برگشتی (انتخاب کاربر) عکس العمل مناسبی نشان دهد.

شکل کلی تابع MsgBox() چنین است :

```
IntResponse = MsgBox(StrPrompt [, IntStyle] [, StrTitle])
or
MsgBox StrPrompt [, IntStyle] [, StrTitle]
```

آرگومان اول (Prompt) از نوع رشته (String) بوده و اجباری می باشد. اما آرگومان دوم اگر قید نشود، ویژوال بیسیک کادر پیام را فقط با یک دکمهٔ OK نمایش خواهد داد. برای تعیین تعداد و نوع دکمه های کادر پیام می توانید از جدول زیر استفاده کنید :

مقدار	نام ثابت	توضیح
۰	VbOkOnly	دکمهٔ OK
۱	VbOkCancel	دکمه های Ok و Cancel
۲	VbAbortRetryIgnore	دکمه های Abort ، Retry و Ignore
۳	VbYesNoCancel	دکمه های Yes ، No و Cancel
۴	VbYesNo	دکمه های Yes و No
۵	VbRetyCancel	دکمه های Retry و Cancel

همچنین با استفاده از آرگومان دوم، می توان آیکن کادر پیام را تغییر داد. جدول زیر مقادیر و نوع آیکن های کادر پیام را نشان می دهد :

مقدار	نام ثابت	توضیح	آیکن
۱۶	VbCritical	آیکن پیام بحرانی	
۳۲	VbQuestion	آیکن علامت سؤال	
۴۸	VbExclamation	آیکن اخطار	
۶۴	VbInformation	آیکن اطلاعات	

آرگومان سوم نیز از نوع رشته بوده و در صورتی که قید نشود، ویژوال بیسیک از نام پروژه در عنوان کادر پیام استفاده می کند.

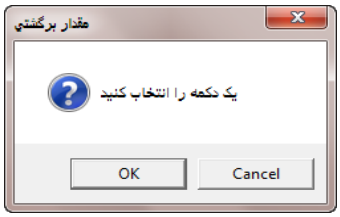
همچنین در جدول زیر مقدار برگشتی هر یک از دکمه های کادر پیام را می توانید مشاهده کنید :

مقدار	نام ثابت	توضیح
۱	vbOk	کاربر Ok را کلیک کرده است.
۲	vbCancel	کاربر Cancel را کلیک کرده است.
۳	vbAbort	کاربر Abort را کلیک کرده است.
۴	vbRetry	کاربر Retry را کلیک کرده است.
۵	vbIgnore	کاربر Ignore را کلیک کرده است.
۶	vbYes	کاربر Yes را کلیک کرده است.
۷	vbNo	کاربرNo را کلیک کرده است.

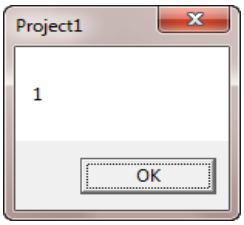
به مثال زیر توجه کنید :

```
Dim Respond As Integer
Respond = MsgBox("یک دکمه را انتخاب کنید", vbOKCancel+vbQuestion, "مقدار برگشتی")
MsgBox Respond
```

دستور فوق، ابتدا کادر زیر را نشان خواهد داد :



و سپس، بعد از انتخاب یک دکمه توسط کاربر (مثلاً دکمهٔ Ok) مقدار متغیر Respond را نشان خواهد داد :



تابع InputBox() : این تابع هم پیامی به کاربر ارائه می دهد و هم امکان می دهد تا کاربر جواب خود را وارد کند. مقدار برگشتی این تابع از نوع Variant است و همیشه می توان آن را به صورت یک رشته به کار برد. شکل کلی این تابه به صورت زیر است :

```
StrAnswer = InputBox(StrPrompt [, StrTitle] [, StrDefault] [, intXpos] [, intYpos])
```

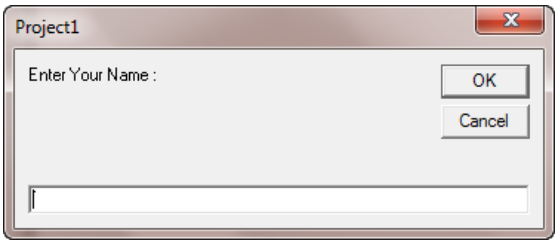
از میان تمام آرگومان های تابع InputBox() فقط اولین آرگومان اجباری است.

- **StrPrompt :** پیام یا پرسشی است که در کادر ورودی مشاهده می شود. حداکثر طول این پیام، می تواند ۱۰۲۴ نویسه باشد.
- **StrTitle :** عنوان پنجرهٔ کادر ورودی است. اگر این آرگومان قید نشود، ویژوال بیسیک به جای آن از نام پروژه استفاده خواهد کرد.
- **StrDefault :** مقداری که به صورت پیش فرض در فیلد ورودی ظاهر خواهد شد.
- **intYpos** و **intXpos :** مختصات ظاهر شدن پنجره کادر ورودی روی صفحه را تعیین می کند. اگر این آرگومان قید نشود، ویژوال بیسیک کادر ورودی را وسط صفحهٔ نمایش قرار خواهد داد.

به مثال زیر توجه کنید :

```
Dim Name As String
Name = InputBox("Enter Your Name :")
MsgBox "Hello " & Name, vbOKOnly+vbInformation, "HelloBox"
```

دستور فوق، ابتدا کادر ورودی زیر را نشان خواهد داد :



و سپس، بعد از وارد کردن یک نام توسط کاربر (مثلاً Reza) کادر پیام زیر را نشان خواهد داد :

* * *

دستور IF

در برنامه نویسی مواردی پیش می آید که بخواهیم دستور یا دستوراتی، هنگامی که شرط خاصی برقرار است، توسط برنامه به اجرا در آید. این مورد در زندگی روزمره نیز دیده می شود؛ به عنوان مثال " اگر فردا باران نیاید، من به کوه خواهم رفت." شرط مورد نظر نیامدن باران است و عملی که قرار است انجام شود، رفتن به کوه می باشد. در ویژوال بیسیک این شرط با استفاده از دستور IF نوشته می شود. شکل کلی این دستور چنین است :

```
If شرط یا شرط ها Then
    دستور یا دستورات بدنه
End If
```

البته اگر دستور بدنهٔ IF فقط یکی باشد، می توان از شکل خلاصه شدهٔ IF استفاده کرد :

```
If شرط یا شرط ها Then دستور
```

به مثال زیر توجه کنید :

```
Dim Number As Integer
Number = InputBox("Enter Number :")
If Number < 10 Then MsgBox "Number Is Small"
```

برنامه فوق، یک عدد را با استفاده از تابع InputBox از کاربر گرفته و اگر کوچکتر از ۱۰ باشد، پیغام "Number Is Small" را نشان می دهد.

کامل کردن IF با Else

در نوع سادهٔ دستور IF ، فقط پردازش حالت درست شرط انجام می شد ولی در نوع کامل IF، حالت نادرست شرط هم پردازش خواهد شد. قسمت Else اجرای حالت نادرست شرط را بر عهده دارد :

```
If شرط یا شرط ها Then
    دستور یا دستورات بدنه
Else
    دستور یا دستورات بدنه
End If
```

به مثال زیر که تکمیل شدهٔ مثال قبلی است، توجه کنید :

```
Dim Number As Integer
Number = InputBox("Enter Number :")
If Number < 10 Then
 MsgBox "Number Is Small"
Else
 MsgBox "Number Is Large"
End If
```

دستور ElseIf (IF متداخل)

اگر یک دستور IF را بعد از Else قرار دهید، به آنها دستورهای IF تودرتو (Nested IF) می گویند. به مثال زیر توجه کنید :

```
Dim Number As Integer
Number = InputBox("Enter Number :")
If Number < 10 Then
 MsgBox "Small"
ElseIf Number > 10 Then
 MsgBox "Large"
Else
 MsgBox "Equality"
End If
```

در مثال فوق، در خط ۵ با دستور ElseIf یک بلوک جدید Else ... If شروع می شود.

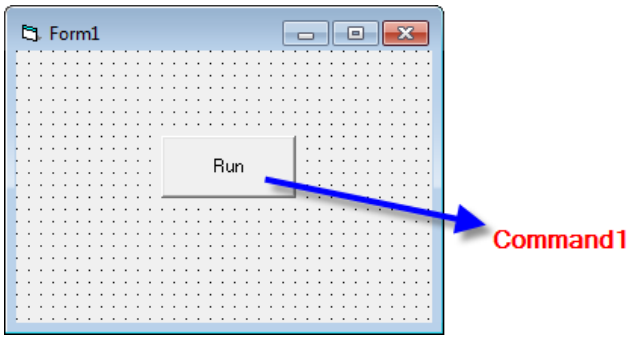
* * *

مثال :

برنامه ای بنویسید که نمرهٔ یک دانش آموز را از ورودی دریافت کرده و سپس بر اساس جدول زیر رتبهٔ وی را نمایش دهد. (توجه داشته باشید که نمرهٔ دانش اموز باید در محدودهٔ صفر تا بیست باشد.)

رتبه	نمره
A	بین ۱۵ تا ۲۰
B	بین ۱۰ تا ۱۵
C	بین ۵ تا ۱۰
D	کمتر از ۵

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



سپس روی دکمهٔ Command1 دابل کلیک کنید و کد های زیر را وارد نمایید :

```
Private Sub Command1_Click()
 Dim N As Integer
 N = InputBox("Enter Student's Number")

 If N > 20 or N < 0 Then
  MsgBox "Number is False"
 Else
   If N < 5 Then
    MsgBox "D"
   ElseIf N > 5 And N <= 10 Then
    MsgBox "C"
   ElseIf N > 10 And N <= 15 Then
    MsgBox "B"
   ElseIf N > 15 Then
    MsgBox "A"
   End If
 End If

End Sub
```

در برنامهٔ فوق، ابتدا با استفاده از تابع InputBox نمرهٔ دانش آموز از ورودی دریافت شده و سپس با استفاده از متد Val به نوع دادهٔ عددی تبدیل می شود و در نهایت درون متغیر N قرار می گیرد. در ادامه، مقدار این متغیر بررسی شده و در صورتی که در محدودهٔ صفر تا بیست نباشد، پیغام مناسب چاپ خواهد شد؛ در غیر این صورت، با یک دستور شرطی جدید مقدار آن متغیر مجدداً و این بار برای تعیین رتبهٔ دانش آموز بررسی می شود و بر اساس مقدار ورودی، نتیجهٔ بررسی با استفاده از تابع MsgBox نمایش داده خواهد شد.

* * *

دستور IIF()

اگر دستور If ... Else چندان پیچیده نباشد، می توان به جای آن از تابع IIF() استفاده کنید که شکل کلی آن به صورت زیر است :

```
IIf (شرط, دستور برای نتیجه درست, دستور برای نتیجه غلط)
```

به عنوان مثال :

```
If Num > 10 Then
  Display = True
Else
 Display = False
End If
```

به دلیل اینکه در هر قسمت If و Else یک دستور وجود دارد، می توان به جای آن از تابع IIF() به شکل زیر استفاده کرد :

```
IIf(Num > 10, Display = True, Display = False)
```

تابع Switch()

این تابع لیستی از عبارت ها را ارزیابی کرده و یک مقدار Variant برمی گرداند یا مقدار مربوط به اولین عبارتی را که درست است، برمی گرداند. شکل کلی این تابع به صورت زیر است :

```
Switch(عبارت ۱, مقدار ۱ [,عبارت ۲ , مقدار ۲] … [, عبارت n, مقدار n])
```

مثال زیر را درنظر بگیرید :

```
Dim Matchup As String
Matchup = Switch(CityName="Tehran", "Persian", CityName="Rome", "Italian")
```

در این مثال، در صورتی که مقدار متغیر CityName برابر با Tehran باشد، متغیر Matchup با Persian مقداردهی می شود و ... .

دستور Select Case

بهترین روش برای بررسی شرایط چندگانه، دستور Select Case است. این دستور شرط های متعدد را انتخاب می کند. شکل کلی این دستور به صورت زیر است :

```
Select Case  عبارت مورد نظر
 Case  عبارت ۱
   یک یا چند دستور
 [Case عبارت ۲
    یک یا چند دستور]
 [Case عبارت nام
   یک یا چند دستور]
 Case Else
   یک یا چند دستور
End Select
```

به مثال زیر توجه کنید :

```
Select Case Text1.Text
 Case "A"
  Label1.Caption = "Perfect"
 Case "B"
   Label1.Caption = "Great"
 Case "C"
   Label1.Caption = "Study harder!"
 Case Else
   Label1.Caption = "Error in Grade!"
End Select
```

ویژوال بیسیک نوع دیگری از Select Case دارد که در آن از عملگر شرطی Is برای بررسی شرط ها استفاده می کند. به مثال زیـر توجـه کنید:

```
Select Case Text1.Text
 Case Is >= 90
  Label1.Caption = "Perfect"
 Case Is >= 80
  Label1.Caption = "Great"
 Case Is >= 70
   Label1.Caption = "Study harder!"
 Case Else
   Label1.Caption = "Error in Grade!"
End Select
```

نوع دیگر بررسی محدوده ها در دستور Select Case، استفاده از عملگر To است. برنامه زیر همان مثال فوق را این بار با استفاده از عملگر To نشان می دهد.

```
Select Case Text1.Text
 Case 0 To 59
  Label1.Caption = "Perfect"
 Case 60 To 69
   Label1.Caption = "Great"
 Case 70 To 79
   Label1.Caption = "Study harder!"
 Case Else
   Label1.Caption = "Error in Grade!"
End Select
```

تابع Choose()

همانطور که تابع ElseIf یک نوع ساده شده دارد (IIF)، تابع Choose() هم نوع ساده شدهٔ دستور Select Case است و بر اساس مقدار اولین آرگومان، یکی از آرگومان های بعدی را انتخاب می کند. این تابع می تواند به تعداد دلخواهی آرگومان داشته باشد. (البته آرگومان های اول و دوم الزامی هستند.) شکل کلی این دستور چنین است :

```
Choose (IndexNull, Expression [, Expression]...)
```

به عنوان مثال، فرض کنید می خواهیم یک لیست کد کالا بسازیم تا وقتی کاربر یک محصول جدید را وارد کرد، برنامه بر اساس کد کالا توضیحات لازم را نشان دهد. بدین منظور میتوانیم از تابع Choose() به زیر استفاده کنیم :

```
Descript = Choose (Text1.Text, "Full Markup", "Mail Order", "Special Order")
```

در این مثال، اگر کاربر نام کد کالا را ۲ وارد کند، مقدار متغیر Descript برابر با "Mail Order" خواهد شد.

* * *

کنترل CheckBox

این کنترل معمولاً برای ارایهٔ انتخاب Yes/No یا True/False به کار برده می شود. کنترل CheckBox می تواند تا چندین انتخاب را به کاربر ارایه دهد و کاربر می تواند یک یا چند CheckBox را انتخاب کند. مهمترین مشخصه این کنترل، Value است که تعیین می کند که آیا کنترل، انتخاب شده است یا نه و آیا این کنترل فعال است یا غیر فعال؟ به طور پیش فرض، این مشخصه با vbUnchecked یا مقدار ۰، مقداردهی می شود. به عنوان مثال وقتی کاربر این کنترل را تیکدار (انتخاب شده) می کند، مقدار مشخصه Value آن ۱ خواهد بود.

جدول زیر، مقادیر و ثابت های معادل آنها در ویژوال بیسیک را برای مشخصه CheckBox نشان می دهد :

String	Value	Constant
Unchecked	0	vbvbUnchecked
Checked	1	vbChecked
Unavailable	2	vbGrayed

کنترل Option Button

این کنترل، برای انتخاب انحصاری یک گزینه از میان چند گزینه به کار می رود. به دکمه های انتخاب (Option Button) ، دکمه های رادیویی (Radio Button) نیز گفته می شود. این کنترل، مانند کنترل قبلی، یک مشخصه Value دارد که فعال یا غیرفعال بودن آن را تعیین می کند. یک کنترل Option Button فقط زمانی غیرفعال می شود که کاربر روی دکمهٔ انتخاب دیگری کلیک کند.

تابع IsNumeric

این تابع برای تشخیص درستی یا نادرستی عدد وارد شده توسط کاربر، بکار می رود و مقدار برگشتی آن از نوع Boolean است. این تابع یک آرگومان را دریافت کرده و اعتبار عددی آن را بررسی می کند. اگر این تابع مقدار False را برگرداند یعنی پارامتر ورودی آن یک عبارت عددی معتبر نیست.

رنگ ها

به وسیله مشخصه ForeColor می توان رنگ قلم و به وسیله مشخصهٔ BackColor می توان رنگ زمینه مربوط به یک شئ را تغییر داد. در ضمن می توان علاوه بر مقدار، از ثابت های نام دار رنگ برای مقداردهی به این مشخصه استفاده کرد.

ثابت	مقدار	توضیحات
vbBlack	&H0	سیاه
vbRed	&HFF	قرمز
vbGreen	&HFF00	سبز
vbYellow	&HFFFF	زرد
vbBlue	&HFF0000	آبی
vbWhite	&HFFFFFF	سفید

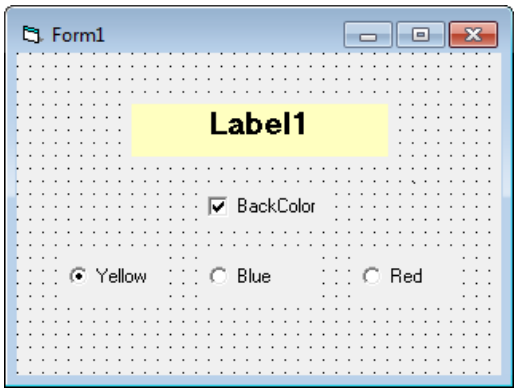
تمرین :

۱- برنامه ای بنویسید که حقوق ناخالص یک کارمند را از ورودی دریافت کرده و با کلیک بر روی دکمهٔ Run، با توجه به جدول زیر میزان مالیات و حقوق خالص را در دو TextBox نمایش دهد.

حقوق ناخالص (H)	مالیات (T)
H <= 10000	T = 0
10000 < H <= 25000	T = (H – 10000) * %5
2500 < H <= 45000	T = (H-25000) * %10 + (25000-10000) * %5
H > 45000	T= (H-45000) * %15 + (45000-25000) * %10 + (25000-10000) * %5

۲- برنامه ای بنویسید که یک عدد را از ورودی دریافت کرده و پس از بررسی اعتبار عددی آن، خروجی مناسب را با استفاده از تابع MsgBox مشخص نماید.

۳- فرمی مانند شکل زیر طراحی کنید. سپس برنامه ای بنویسید که اگر CheckBox تیکدار بود، با انتخاب هر یک از کنترلهای Option رنگ پس زمینهٔ (BackColor) کنترل Label1 تغییر کند ولی اگر CehckBox تیکدار نبود، رنگ قلم (ForeColor) کنترل Label1 تغییر یابد.



فصل چهارم :

حلقه های تکرار در ویژوال بیسیک

در ویژوال بیسیک تعدادی دستور وجود دارد که با استفاده از آن می توان یک یا چند سطر کد را برای چندین بار تکرار کرد. به این دستورها حلقه های تکرار می گویند که خود شامل دو صورت برای تکرار می باشد :

- حلقه های تکرار معین
- حلقه های تکرار نامعین

در حلقه های معین می توان تعداد تکرار حلقه را مشخص کرد اما در نامعین تکرار حلقه تا زمانی که شرط مورد نظر بر قرار شود انجام می گیرد.

حلقه تکرار For

برای اجرای دستورهای مشخصی از برنامه به دفعات معین، از حلقه تکرار For می توان استفاده کرد. هنگامی این حلقه بکار می رود که تعداد دفعات تکرار مشخص باشد. شکل کلی این دستور چنین است :

```
For متغیر = عدد شروع To عدد پایان [Step تعداد پرش]
 مجموع دستور ها
Next [متغیر]
```

قطعه برنامهٔ زیر، اعداد ۱ تا ۱۰ را به صورت صعودی روی فرم چاپ می کند.

```
For X = 1 To 10
 Print X
Next X
```

قطعه برنامه بعدی، اعداد ۱۰ تا ۱ را به صورت نزولی روی فرم چاپ می کند.

```
For Y = 10 To 1 Step -1
 Print Y
Next Y
```

مجموعه دستورات زیر، اعداد زوج بین صفر تا ۱۰ را روی فرم چاپ می کند.

```
For I = 0 To 10 Step 2
 Print I
Next I
```

حلقه تکرار Do

این حلقه چند شکل مختلف دارد.

۱)

```
Do While شرط
  یک یا چند دستور
Loop
```

۲)

```
Do
  یک یا چند دستور
Loop While شرط
```

۳)

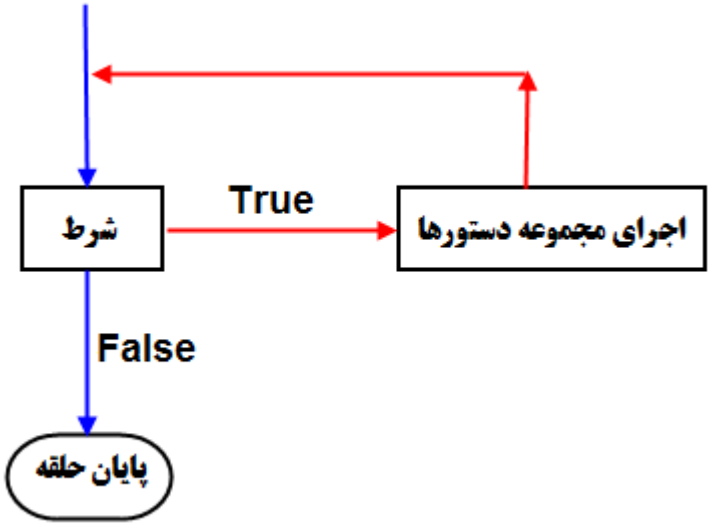
```
Do Until شرط
  یک یا چند دستور
Loop
```

۴)

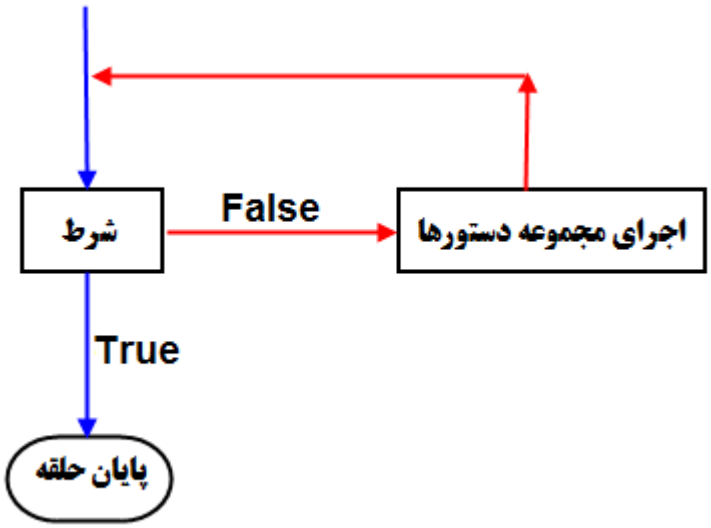
```
Do
  یک یا چند دستور
Loop Until شرط
```

شرط در حلقهٔ Do می تواند هر عبارت یا متغیر منطقی باشد که مقدار True یا False برمی گیرداند. اگر شرط حلقه در ابتدای آن بررسی شود، ممکن است حلقه هرگز اجرا نشود. اما اگر بررسی حلقه در انتهای آن صورت گیرد، حلقه حداقل یک بار اجرا خواهد شد. حلقه هایی که While دارند، تا زمانی که شرط درست است اجرا می شوند، ولی در حلقه های Until، اگر شرط درست باشد، از حلقه خارج می شود.

نمودار گردشی زیر طرز اجرای حلقهٔ Do While/Loop را نشان می دهد :



طرز اجرای حلقهٔ Do Until/Loop نیز در نمودار گردشی زیر مشخص است :



گاهی با بررسی شرط خاصی در دستورات یک حلقهٔ تکرار، نیاز داریم تا قبل از آنکه کار حلقه به پایان برسد، از آن خارج شویم. انجام این کار با استفاده از دستور Exit صورت می گیرد. این دستور می تواند به سه شکل زیر مورد استفاده قرار بگیرد.

```
Exit Sub | For | Do
```

حلقه های متداخل

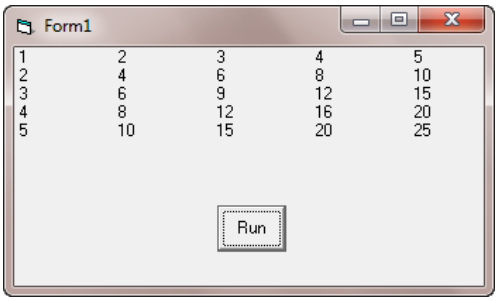
چند حلقه تکرار را می توان در داخل یکدیگر بکار برد و در اینصورت آنها را حلقه های متداخل (Nested Loops) می نامند. حلقه ای که داخلی تر است، سریعتر اجرا می شود؛ زیرا با هر بار اجرای حلقهٔ بیرونی، یک بار اجرا می شود. به عنوان مثال، در قطعه برنامه زیر، حلقه داخلی در مجموع ۴۰ بار اجرا می شود.

```
For Out = 1 To 4
  For In = 1 To 10
    دستورات
  Next In
Next Out
```

به مثال زیر توجه کنید، این برنامه جدول ضرب اعداد ۱ تا ۵ را روی فرم چاپ می کند :

```
Me.Cls
For I = 1 To 5
  For J = 1 To 5
    Print I * J,
  Next J
  Print
Next I
```

خروجی برنامهٔ فوق، بصورت زیر خواهد بود :



نکته :

در ویژوال بیسیک کلمه کلیدی Me اشاره به فرم جاری دارد و به کمک متد Cls می توان متن یا گرافیک موجود روی فرم یا کادر تصویر را در زمان اجرا پاک کرد.

تمرین :

۱- برنامه ای بنویسید که مجموع اعداد ۱ تا ۱۰۰ را به دست آورده و روی فرم چاپ کند.

۲- برنامه ای بنویسید که تعدادی عدد مثبت را از ورودی دریافت کرده و حاصل جمع و میانگین آنها را روی فرم چاپ کند. توجه داشته باشید که تعداد اعداد ورودی مشخص نیست و شرط توقف، ورود عدد صفر یا کوچکتر از صفر است.

۳- برنامه ای بنویسید که خروجی زیر را تولید کند :

```
*
* *
* * *
* * * *
* * * * *
```

فصل پنجم :

آرایه ها در ویژوال بیسیک

آرایه متغیری است دارای چندین خانه (خانه های متوالی حافظه) که تحت یک نام مشترک، یک نوع داده را نگه داری می کند. به هر خانهٔ آرایه یک عنصر گفته می شود و هر عنصر دارای شمارهٔ خاصی که اندیس نامیده می شود، است. برای دسترسی به عناصر آرایه باید از نام آرایه همراه با اندیس آن استفاده کرد. بطور پیش فرض مقدار اندیس اولین عنصر آرایه، صفر می باشد. در ادامه نحوهٔ تعریف و استفاده از آرایه ها را توضیح خواهیم داد.

اعلان آرایه های یک بعدی

پیش از آنکه بتوان از یک آرایه استفاده کرد، باید آن را به صورت زیر اعلان کرد :

```
Dim نام آرایه(تعداد عناصر) As نوع داده
```

و برای مقداردهی عناصر آن از دستور انتساب به صورت زیر استفاده می شود :

```
نام آرایه(اندیس مورد نظر) = مقدار
```

توجه داشته باشید که مقداری که به هر یک از عناصر آرایه نسبت داده می شود، باید از همان نوعی باشد که آرایه تعریف شده است.

به مثال زیر توجه کنید :

```
Dim MyArray(5) As Integer
```

در دستور فوق، یک آرایه ۵ عنصری از نوع عدد صحیح تعریف شده است. برای دسترسی و مقداردهی عناصر این آرایه می توان از حلقهٔ تکرار بصورت زیر استفاده کرد :

```
Dim i AS Integer
For i=0 To 4
 MyArray(i) = InputBox("Enter Number")
Next
```

نکته (۱) :

همانطور که در دستورات بالا، مشاهده می کنید آرایهٔ فوق دارای ۵ عنصر است که اندیس آن از صفر شروع شده و بـه ۴ خـتم مـی شـود. برای اینکه اندیس عناصر یک آرایه از یک شروع شود، باید دستور Option Base 1 را در بخش General تایپ کرد.

نکته (۲) :

علاوه بر دستور Option Base 1 ، می توان آرایه ها را با استفاده از کلید واژهٔ To در داخل اندیس های آن، اعلان کرد. به مثال زیر توجـه کنید:

```
Dim MyAray(1 To 5) As Integer
```

این روش، راه ساده ای برای شروع آرایه از اندیس غیر از صفر است.

نکته (۳) :

به آرایه هایی که هنگام اعلان، تعداد عناصر آنها تعیین می شود، **آرایه های ایستا** و به آرایه هایی که تعداد عناصـر آنهـا در طـول برنامـه تعیین می شود، **آرایه های پویا** می گویند.

تعریف و استفاده از آرایه های پویا

اگرچه تعداد عناصر آرایه هنگام اعلان آن تعیین می شود ولی امکان تغییر اندازهٔ آرایه وجود دارد. برای این کار بایـد از کلیـد واژهٔ ReDim بصورت زیر استفاده کرد :

```
ReDim [Preserve] آرایـه نـام(تـعداد عنـاصر) As نـوع داده
```

نکات کلیدی :

- در تغییر تعداد عناصر آرایه، کلید واژهٔ Preserve اختیاری بوده و امکان نگه داری مقادیر تمام عناصر آرایه را فراهم می کند. به عبارت دیگر، اگر از این کلید واژه استفاده نکنید، مقدار تمام عناصر آرایه از بین میرود.
- نوع داده برای تغییر تعداد عناصر یا تغییر بُعد یک آرایه، اختیاری است و با کلید واژهٔ ReDim نمی تواند تغییر یابد مگر اینکه آرایه از نوع Variant باشد.
- ***نکته بسیار مهم :*** اگر آرایه ای تعریف می کنید که بعداً می خواهید آن را تغییر اندازه دهید، نباید اندازهٔ آن را هنگام اعلان، تعیین کنید.

اکنون، به مثال زیر توجه کنید:

```
Option Base 1
Dim SampleArray() AS Integer
Dim i, j As Integer

Private Sub Command1_Click()
   i = InputBox("Enter Size of Array:")
   ReDim SampleArray(i)
End Sub

Private Sub Command2_Click()   For j = 1 To i
    SampleArray(j) = InputBox("Enter Value:")
   Next
End Sub
```

در برنامهٔ فوق، با کلیک روی Command1 تعداد عناصر آرایه از کاربر دریافت می شود و طول آرایه به همان اندازه تغییر می کند. سپس با کلیک روی Command2 تمام عناصر آرایه توسط کاربر مقداردهی می شود.

نکته ای که در برنامهٔ فوق مشهود است، از بین رفتن مقادیر موجود در آرایه، با هر بار کلیک بر روی Command1 است. در واقع، با هر بار کلیک روی Command1 تمام مقادیر موجود در آرایه که با کلیک بر روی Command2 مقداردهی شده بودند، از بین می رود. برای رفع این مشکل کافی است کد مربوط به Command1_Click() را بصورت زیر تصحیح کنیم :

```
Private Sub Command1_Click()
   i = InputBox("Enter Size of Array:")
   ReDim Preserve SampleArray(i)
End Sub
```

مرتب سازی عناصر آرایه

مرتب کردن داده ها، یکی از عملیات مهم در برنامه های کاربردی است. الگوریتم های متعددی برای مرتب کردن n عنصر آرایه وجود دارد که از جهات مختلف قابل دسته بندی و مقایسه هستند. بر اساس خصوصیات این الگوریتم ها، هر کدام ممکن است در مسائل خاصی، عملکرد بهتری داشته باشند.

الگوریتم مرتب سازی حبابی (Bubble Sort)

فرض کنید می‌خواهیم n داده به صورت صعودی مرتب شوند. عنصر اول را با عنصر دوم مقایسه کرده و در صورتی که عنصر اول بزرگتر باشد، جای آنها را عوض می‌کنیم. همین کار را با عناصر دوم و سوم انجام می‌دهیم و همینطور عناصر سوم و چهارم و الی آخر... .

وقتی این کار تمام شد بزرگترین عنصر بین داده‌ها به آخرِ لیست می‌رسد . حالا یک بار دیگر از اول این کار را انجام می‌دهیم، اما این بار تا عنصر (n - 1)ام ادامه می‌دهیم (عنصر nام در مرحله اول در جای خودش قرار گرفته)؛ باز هم این کار را تا عنصر (n - 2)ام تکرار می‌کنیم و... . این کار را آنقدر ادامه می دهیم تا اینکه بالاخره داده‌ها مرتب می‌شوند.

* * *

مثال :

برنامه ای بنویسید که با کلیک بر روی دکمهٔ اول، ۱۰ عدد تصادفی ایجاد کرده و ابتدا آنها را درون یک آرایهٔ ۱۰ عنصری قرار دهد و سپس آرایه را درون شئ List1 منتقل کند و با کلیک بر روی دکمهٔ دوم عناصر آرایه را با استفاده از الگوریتم حبابی مرتب کرده و درون List2 قرار دهد.

* * *

قبل از انجام مثال بالا، باید با طرز استفاده از کنترل ListBox و ایجاد اعداد تصادفی در ویژوال بیسیک آشنا شوید:

کنترل ListBox

کنترل ListBox، کنترلی برای لیست کردن اطلاعات است. از مشخصه های مهم این کنترل می توان به مشخصهٔ List اشاره کرد که محتوای این کنترل از طریق آن وارد می شود. مشخصهٔ ListCount که تعداد آیتم های موجود در کنترل را بر می گرداند و ListIndex که شمارهٔ اندیس یک آیتم از ListBox را مشخص می کند، از دیگر مشخصه های مهم این کنترل هستند.

– برای اضافه کردن آیتم به ListBox دوحالت وجود دارد: حالت اول، حالت طراحی فرم است که در آن درون مشخصهٔ List مقادیر رشته ای را وارد می کنیم و حالت دوم، زمان کد نویسی است که از متدی به نام AddItem استفاده می شود. فرم کلی آن به صورت زیر است :

```
Object.AddItem "رشته مورد نظر"
```

Object مقدار مشخصهٔ Name مربوط به یک کنترل ListBox است و "رشته مورد نظر" رشته ای است که می خواهید به ListBox اضافه کنید.

– هر آیتم از ListBox دارای یک اندیس می باشد که این اندیس ها مانند آرایه ها از صفر شروع خواهد شد. به عنوان مثال، برای بدست آوردن مقدار **سومین** آیتم از یک لیست ۵ آیتمی با نام List1 ، از کد زیر استفاده می کنیم :

```
Label1.Caption = List1.List(2)
```

همانطور که اشاره شد، شمارهٔ اندیس آیتم انتخاب شده از لیست، با مشخصه ListIndex مشخص می شود ، مثلاً برای بدست آوردن آیـتمِ انتخاب شده در ListBox، می توان کد زیر را بکار برد :

```
Label1.Caption = List1.List(List1.ListIndex)
```

– برای حذف یک آیتم از لیست، از متد RemoveItem استفاده می شود. در مثال زیر **دومین** آیتم از لیست را حذف می کنیم :

```
List1.RemoveItem(1)
```

– متد Clear نیز برای پاک کردن تمام آیتم های موجود در لیست بکار می رود :

```
List1.Clear
```

ایجاد عدد تصادفی

در ویژوال بیسیک تابعی به نام Rnd وجود دارد که اعداد تصادفی بین صفر تا یک را تولید می کند. این اعداد کاملاً هم تـصادفی نیـستند بلکه از طریق فرمول های خاص ریاضی و با استفاده از جداولی که در کامپیوترهای مختلف متفاوت اند، تولید می شود. دستور زیـر را درون رویداد Load فرم بنویسید :

```
Msgbox Rnd
```

سپس برنامه را چند بار اجرا کرده و ببندید؛ همانطور که مشاهده می کنید، در هر بار اجرای برنامـه، عـدد یکـسانی بـه شـما نمـایش داده می شود. این کار برای تست برنامه و رفع اشکالات برنامه کاملاً مفید است، اما ممکن است شما بخواهید اعداد تولید شده کـاملاً تـصادفی و متفاوت باشند. برای اینکار از روالی به نام Randomize استفاده می شود. در اول رویه مورد نظر که از تابع Rnd استفاده می کنید، دستور زیر را تایپ کنید:

```
Randomize Timer
```

این کار منبع تولید اعداد تصادفی را Timer یا مدت زمان گذشته از نیمه شب بر حسب ثانیه قرار می دهد. چون Timer اعدادی که تولیـد می کند در هر شبانه روز فقط یکبار تکرار می شوند؛ در نتیجه اعداد تولید شده، کاملاً تصادفی و غیر تکراری در هر بار اجرای برنامه هستند.

برای اینکه عدد تولید شده بین صفر و n باشند، باید از دستور زیر استفاده کنید:

```
Rnd * n
```

حال اگر خواستید عدد تولید شده بین ۱ و n باشد این دستور را بکار ببرید:

```
Rnd * (n-1) + 1
```

در بیشتر برنامه ها اعداد تصادفی، در قالب اعداد صحیح مورد استفاده قرار می گیرند. برای این کار باید عدد تصادفی تولید شده را به صورت زیر به عدد صحیح تبدیل کرد :

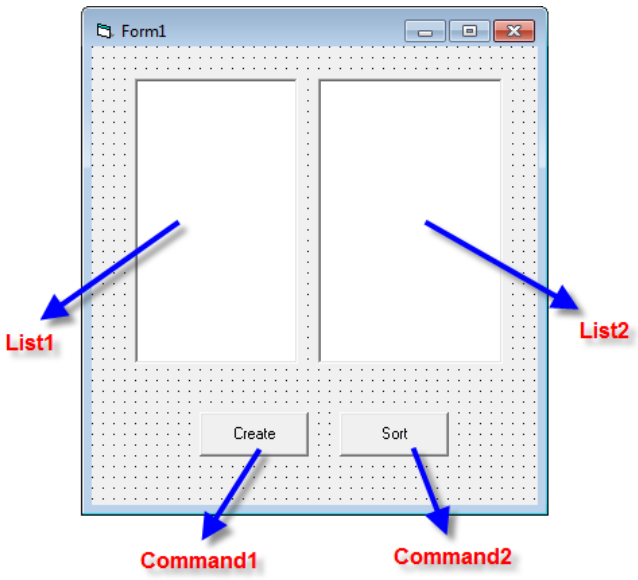
```
int(Rnd * (n-1)) + 1
```

در مثال زیر یک عدد تصادفی بین ۱ تا ۱۰۰ تولید شده و درون متغیر R که از نوع عدد صحیح است، ذخیره می گردد :

```
Dim R As Integer
R = int(Rnd * 99) + 1
```

* * *

حال به ادامه بحث آرایه ها و حل مثال ذکر شده می پردازیم. برای این کار ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



اکنون در بخش General برنامه، یک آرایهٔ ۱۰ عنصری تعریف می کنیم:

```
Dim MyArray(10) As Integer
```

سپس روی دکمهٔ Command1 (Create) دابل کلیک کنید و کد های زیر را تایپ کنید :

```
Private Sub Command1_Click()
 Randomize Timer
 List1.Clear

 Dim x As Integer
 For x = 0 To 9
  MyArray(x) = int(Rnd * 100)
  List1.AddItem (MyArray(x))
 Next x
 List2.Clear
End Sub
```

با اجرای قطعه کد بالا، ابتدا محتوای List1 (با استفاده از متد Clear) از بین می رود. سپس ۱۰ عدد تصادفی ایجاد شده و درون آرایه و کنترل List1 افزوده می شود. در انتها نیز محتوای List2 حذف خواهد شد.

حال روی دکمهٔ Command2 (Sort) دابل کلیک کنید و کد های زیر را تایپ کنید :

```
Private Sub Command2_Click()

 Dim tmp As Integer
 Dim i, j, x As Integer

 For i = 9 To 0 Step -1
  For j = 0 To 9
   If j < 9 Then
    If MyArray(j) > MyArray(j + 1) Then
     tmp = MyArray(j)
     MyArray(j) = MyArray(j + 1)
     MyArray(j + 1) = tmp
    End If
   End If
  Next j
 Next i
```

```
For x = 0 To 9
  List2.AddItem (MyArray(x))
 Next x

End Sub
```

با اجرای دستورات فوق، مرتب سازی آرایه بر اساس الگوریتم مرتب سازی حبابی (Bubble Sort) انجام می شود و نتیجه آن در List2 نمایش داده می شود.

* * *

جستجو در عناصر آرایه

عمل جستجو یکی از مهمترین وظایف برنامه های کامپیوتری می باشد. در این مبحث دو روش جستجو را مورد بررسی قرار می دهیم. نخست، روش جستجوی خطی است که معمولاً در آرایه های نا مرتب مورد استفاده قرار می گیرد و روش دیگر، جستجوی دودویی می باشد که در آرایه های مرتب شده از این شیوه می توانیم استفاده کنیم.

الگوریتم جستجوی خطی (Linear Search)

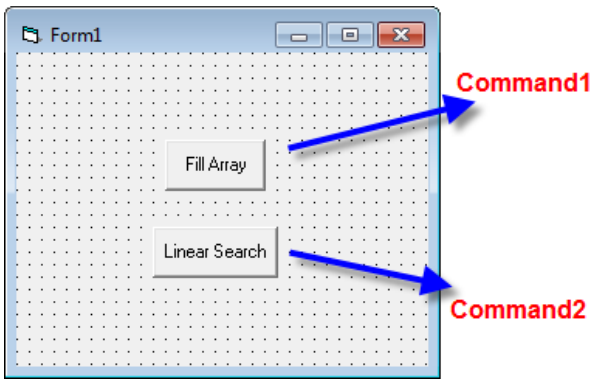
در این روش عنصر مورد جستجو(کلید جستجو)، با هر یک از عناصر آرایه مقایسه می شود. چنانچه با هم برابر بودند؛ جستجو به پایان می رسد و در غیر این صورت، عمل مقایسه با عنصر بعدی آرایه صورت می گیرد. این روند تا یافتن عنصر مورد نظر و یا جستجوی تمامی عناصر آرایه ادامه پیدا می کند.

* * *

مثال :

برنامه ای بنویسید که با کلیک بر روی دکمهٔ اول، ۵ عدد را از ورودی دریافت کرده و درون یک آرایهٔ ۵ عنصری قرار دهد. سپس با کلیک بر روی دکمهٔ دوم مجدداً یک عدد را به عنوان کلید جستجو از ورودی دریافت کرده و با استفاده از الگوریتم جستجوی خطی آن را درون آرایه جستجو کند. خروجی این برنامه ، پیغام مناسبی است که با توجه به نتیجهٔ جستجو، با استفاده از تابع MsgBox نمایش داده می شود.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



در مرحله اول، آرایه را در بخش General به صورت زیر تعریف می کنیم :

```
Dim x(5) As Integer
```

حال روی دکمهٔ Command1 دابل کلیک کنید و کد های زیر را وارد نمایید :

```
Private Sub Command1_Click()
 Dim i As Integer
 For i = 0 To 4
  x(i) = Val(InputBox("Enter Number:"))
 Next
End Sub
```

در قطعه کد فوق، یک حلقه تعریف کرده ایم که به تعداد عناصر آرایه تکرار خواهد شد. همانطور که مشاهده می کنید، از آنجایی که در بخش General دستور Option Base 1 را وارد نکرده ایم، اندیس آرایهٔ x از صفر شروع شده و به تعداد ۵ عنصر یعنی تا اندیس ۴ تعیین می گردد. بنابراین، مقدار شروع حلقه را صفر و مقدار آن را ۴ قرار دادیم. درون حلقه نیز، مقدار هر عنصر از آرایه را با استفاده از تابع InputBox از ورودی دریافت و سپس با استفاده از متد Val به نوع دادهٔ عددی تبدیل می کنیم و درون عناصر آرایه قرار می دهیم.

حال روی دکمهٔ Command2 دابل کلیک کنید و کد های زیر را وارد نمایید :

```
Private Sub Command2_Click()
 Dim srch, j As Integer
 srch = Val(InputBox("Enter Search Value:"))

 For j = 0 To 4
  If x(j) = srch Then
   MsgBox "Value is Found."
   Exit Sub
  End If
 Next

 MsgBox "Value Not Found."
End Sub
```

در این قطعه برنامه، بعد از تعریف متغیرهای مورد نیاز، ابتدا مقدار مورد نظر برای عمل جستجو از ورودی دریافت می شود. سپس یک حلقه تعریف شده است که از ابتدا تا انتهای عناصر آرایه را با مقدار ورودی مقایسه می کند. در صورتی که مقدار ورودی با مقدار عنصری از آرایه برابر باشد، پیغام مناسب ظاهر شده و سپس با استفاده از دستور Exit (خروج زودرس) از روال خارج می شود؛ بنابراین دیگر پیغام دوم (بعد از حلقهٔ تکرار) ظاهر نخواهد شد. اما اگر تمام عناصر آرایه مورد بررسی قرار گرفته و هیچ کدام با مقدار ورودی برابر نباشد، دستور بعد از حلقهٔ تکرار، یعنی نمایش پیغام دوم، اجرا خواهد شد.

استفاده از LBound و UBound

LBound حد پایین و UBound حد بالای یک آرایه را مشخص می کند. مثلاً درون حلقهٔ تکرار که در مثال قبل برای Command2 نوشتیم، می توان کد زیر را نیز بکار برد :

```
For j = LBound(x) To UBound(x)
  ...
Next
```

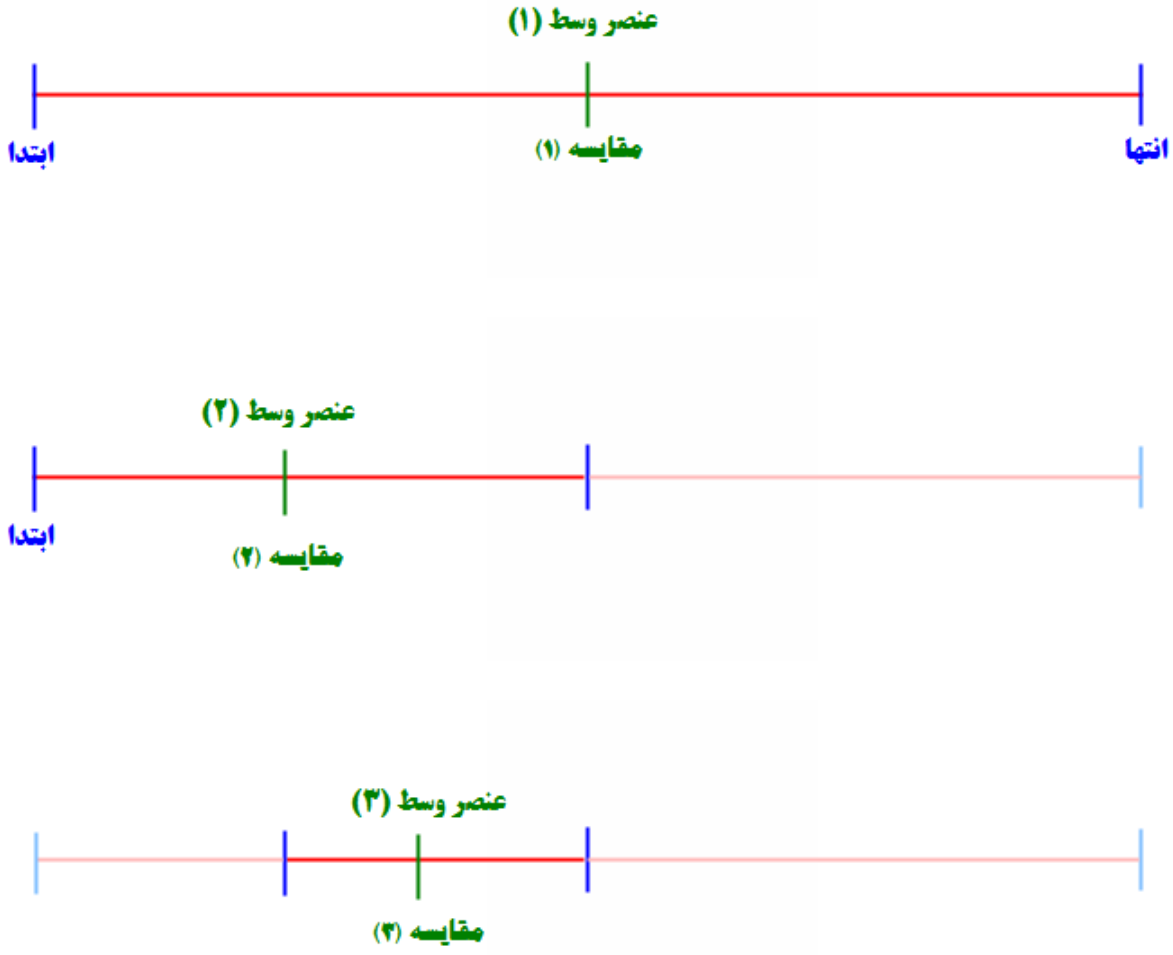
در این صورت LBound(x) برابر با صفر و UBound(x) نیز برابر با چهار خواهد بود.

* * *

الگوریتم جستجوی دودویی (Binary Search)

در آرایه های بزرگ استفاده از روش جستجوی خطی مناسب نیست؛ برای مثال فرض کنید آرایه ای با ۱۰۰۰ عنصر داریم و مقداری که مورد جستجو است در عنصر ماقبل آخر (یعنی عنصر ۹۹۹) قرار دارد، در این صورت تعداد دفعات جستجو زیاد خواهد بود و سبب اتلاف زمان می شود.

روش جستجوی دودویی، روی آرایه های بزرگ و مرتب قابل اجراست. بنابراین، اگر آرایهٔ مورد جستجو مرتب نباشد، ابتدا باید آن را مرتب کرد. در این روش، ابتدا مقدار مورد نظر (کلید جستجو) با عنصر وسط آرایه مقایسه می گردد اگر با این هم برابر باشند جستجو تمام می شود؛ اگر کلید جستجو از عنصر وسط بزرگتر بود جستجو در بخش بالایی آرایه و در غیر اینصورت جستجو در بخش پایینی آن انجام می شود. این رویه تا یافتن عنصر مورد نظر یا بررسی کل عناصر آرایه ادامه می یابد.

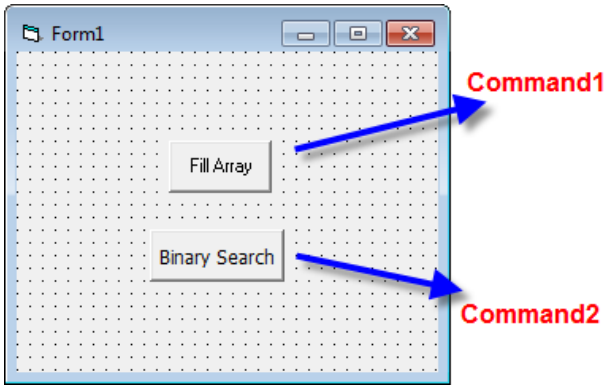


* * *

مثال :

برنامه ای بنویسید که با کلیک بر روی دکمهٔ اول، ۵ عدد تصادفی بین صفر تا ده ایجاد کرده و درون یک آرایهٔ ۵ عنصری قرار دهد. (اندیس آرایه از یک شروع شود) سپس با کلیک بر روی دکمهٔ دوم یک عدد را از ورودی دریافت کرده و با استفاده از الگوریتم جستجوی دودویی آن را درون آرایه جستجو کند. خروجی این برنامه باید در صورت پیدا نشدن مقدار مورد نظر، پیغامِ "یافت نشد" باشد و در صورت پیدا شدن آن، شماره اندیس عنصری که کلید جستجو با آن برابر است، باشد.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



ابتدا در بخش General کد های زیر را تایپ کنید :

```
Option Explicit
Option Base 1
Dim x(5) As Integer
```

سپس روی دکمهٔ Command1 دابل کلیک کنید و کد های زیر را وارد نمایید :

```
Private Sub Command1_Click()
 Randomize Timer

 Dim a As Integer
 For a = LBound(x) To UBound(x)
  x(a) = Rnd * 15
 Next

 Dim i, j, tmp As Integer
 For i = UBound(x) To LBound(x) Step -1
  For j = LBound(x) To i
```

```
    If j = i Then Exit For
    If x(j) > x(j + 1) Then
     tmp = x(j + 1)
     x(j + 1) = x(j)
     x(j) = tmp
    End If
   Next
 Next
End Sub
```

در دستورات بالا، ابتدا آرایه را با اعداد تصادفی پر کرده و سپس آن را قبل از جستجو دودویی مرتب می کنیم.

حال روی دکمهٔ Command2 دابل کلیک کنید و کد های زیر را وارد نمایید :

```
Private Sub Command2_Click()
 Dim key As Integer
 key = Val(InputBox("Enter Search key:"))

 Dim First, Middle, Last As Integer
 First = LBound(x)
 Last = UBound(x)
 Do While First <= Last
  Middle = (First + Last) \ 2
  If key = x(Middle) Then
   MsgBox Middle
   Exit Sub
  ElseIf key < x(Middle) Then
   Last = Middle - 1
  Else
   First = Middle + 1
  End If
 Loop

 MsgBox "یـافت نـشد"
End Sub
```

در قطعه برنامهٔ فوق، ابتدا مقدار مورد نظر برای جستجو، از کاربر پرسیده می شود. سپس بـا اسـتفاده از الگـوریتم جـستجوی دودویـی، آن مقدار در عناصر آرایه جستجو خواهد شد و نهایتاً نتیجه جستجو، با استفاده از تابع MsgBox به کاربر نمایش داده می شود.

* * *

آرایه های دو بُعدی

تا اینجا، با آرایه های یک بعدی آشنا شدید و همانطور که یاد گرفتید، آرایه های یک بعدی مجموعه ای یک سطری از متغیرها می باشد. آرایه های دو بعدی مجموعه ای از متغیرها هستند که تحت عنوان یک نام مشترک و یک نوع داده در سطرها و ستون ها سازماندهی می شوند. در ویژوال بیسیک می توان آرایه هایی تا حداکثر ۶۰ بعد ایجاد کرد.

اعلان آرایه های دوبعدی

برای اعلان یک آرایهٔ دوبعدی به صورت زیر عمل می کنیم:

```
Dim آرایه اسم(تعداد سطرها, تعداد ستون ها) As نوع داده
```

مثال زیر، آرایهٔ دو بعدی از نوع رشته با ۵ سطر و ۳ ستون تحت عنوان Tmp ایجاد می کند:

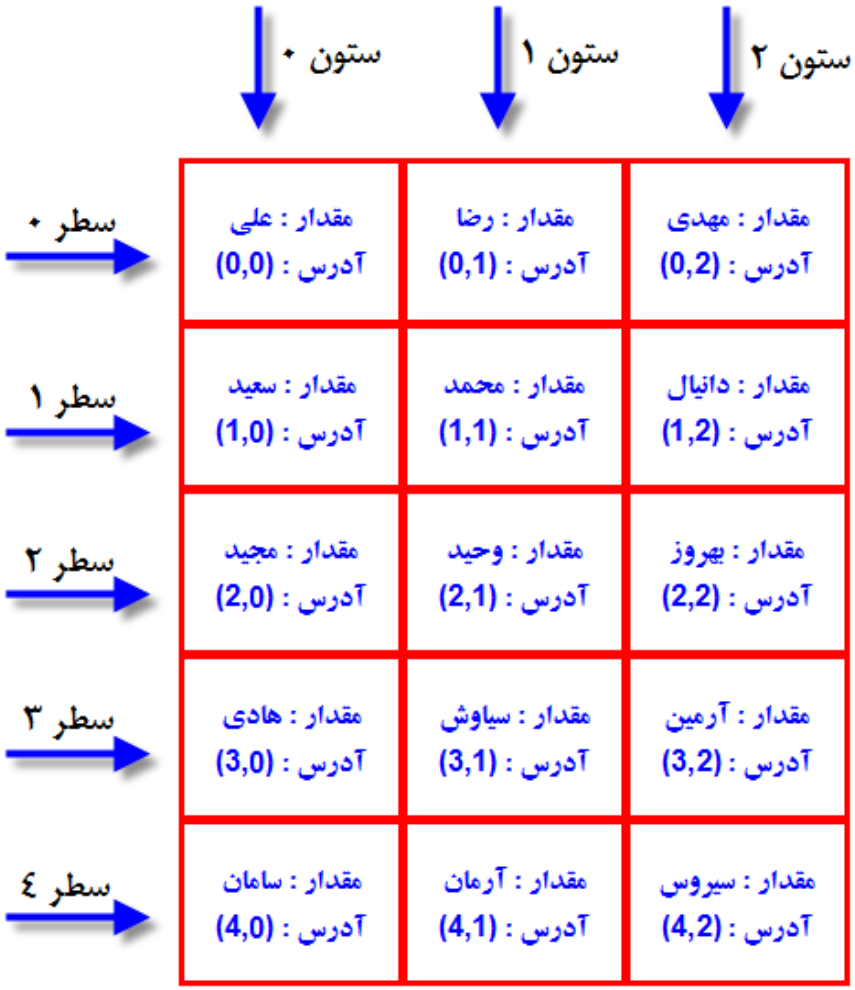
```
Dim Tmp(5, 3) As String
```

برای استفاده از آرایهٔ تعریف شده، باید به صورت زیر استفاده کنید :

```
Tmp(1,2) = InputBox("Enter Name:")
MsgBox Tmp(1,2)
```

در دستورات فوق، ابتدا با استفاده از تابع InputBox یک نام از کاربر پرسیده می شود و سپس مقدار ورودی درون عنصر مورد نظر (سطر دوم، ستون سوم {زیرا اندیس آرایه های دو بعدی نیز از صفر شده می شود.}) از آرایهٔ دو بعدی ذخیره می شود. سپس با استفاده از تابع MsgBox مقدار همان عنصر نمایش داده می شود.

بهتر است برای درک بیشتر، تصویری از ساختمان آرایهٔ دو بعدیِ فوق را در شکل زیر بررسی کنیم :



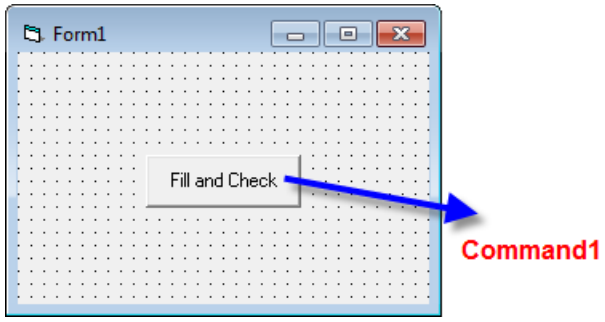
نکته :

می توان از کلید واژهٔ ReDim برای تغییر اندازهٔ آرایهٔ چند بعدی استفاده کرد. برای این کار، اگر از کلید واژهٔ Preserve استفاده مـی کنیـد، باید توجه داشته باشید که فقط آخرین بعد آرایهٔ چند بعدی می تواند تغییر اندازه یابد و تعداد ابعاد نمی تواند تغییر کنند.

مثال :

برنامه ای بنویسید که شماره دانشجویی و نمرهٔ پنج دانشجو را از ورودی دریافت کرده و درون یک آرایهٔ دو بعدی ذخیره کند. سپس شمارهٔ دانشجویی که بالاترین نمره را کسب کرده، در خروجی چاپ کند.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



اکنون کد های زیر را تایپ کنید :

```
Private Sub Command1_Click()
 Dim Info(1 To 5, 1 To 2) As Integer
 Dim i, max As Integer

 max = 1
 For i = 1 To 5
  Info(i, 1) = Val(InputBox("Enter Student's Number"))
  Info(i, 2) = Val(InputBox("Enter Student's Mark"))
  If Info(i, 2) > Info(max, 2) Then max = i
 Next

 MsgBox "Student's Number is " + CStr(Info(max, 1))
End Sub
```

برنامه را ذخیره و اجرا کنید تا پس از وارد کردن مقادیر مورد نیاز، خروجی مطلوب را مشاهده کنید.

تمرین :

۱- برنامه ای بنویسید که یک آرایهٔ پنج عنصری را با استفاده از تابع InputBox مقداردهی کند و سپس با استفاده از الگوریتم جستجوی باینری مقدار وارد شده در Text1 را درون آرایه جستجو کند و نتیجه مطلوب را توسط تابع MsgBox نمایش دهد.

۲- برنامه ای بنویسید که نمرهٔ ۱۰ دانشجو را دریافت کند و نمرهٔ افرادی که رتبهٔ اول، دوم و سوم را کسب کردند، به ترتیب در خروجی چاپ کند.

۳- برنامه ای بنویسید که مقدار متغیر i را از ورودی دریافت کرده و یک آرایهٔ پویا به طول i تعریف کند. سپس مقادیر آرایه را از ورودی خوانده و در آرایه ذخیره کند. نهایتا مقادیر آرایه را درون ListBox قرار دهید.

۴- برنامه ای بنوسید که مقادیر یک ماتریس ۳ × ۳ را دریافت کند و درون یک آرایهٔ دو بعدی ذخیره کند. سپس آرایه را به صورت همان ماتریس روی فرم چاپ کند.

فصل ششم :

روال ها و توابع

مسائلی که تاکنون حل کردیم ساده و کوچک بودند. برای حل مسئله های پیچیده، باید آن را به بخش های کوچک تری تقسیم کرد، به طوری که هر بخش کار خاصی را انجام دهد. برنامه ای را که برای حل بخشی از مسئله نوشته می شود **زیربرنامه** می گویند. این روش دارای مزایایی است که برخی از آن ها عبارتند از :

۱. خوانایی برنامه را بالا می برد، زیرا اهداف کلی برنامه را می توان در برنامه اصلی مشاهده کرد.

۲. رفع اشکال زیربرنامه ها آسان تر است.

۳. از زیربرنامهٔ نوشته شده، در جاهای مختلف برنامه می توان بدون تکرار کدهای آن، به دفعات استفاده کرد.

انواع زیربرنامه

زیربرنامه ها به طور کلی به دو دسته تقسیم می شوند :

- روال (Sub)
- تابع (Function)

مهمترین تفاوتی که بین Function و Sub وجود دارد، این است Function از طریق نام خود یک مقدار برگشتی دارد ولی Sub از این طریق هیچ مقدار برگشتی ندارد.

زیربرنامهٔ روال (Sub)

اگر زیر برنامه ای که می نویسیم، صرفاً وظیفهٔ اجرای یک سری دستورات را داشته و نیاز به مقدار برگشتی ای بعنوان نتیجه نداشته باشد، از روال یا Sub استفاده می کنیم که شکل کلی استفاده از آن به صورت زیر می باشد :

```
[Public|Private]  Sub  نام زیربرنامه(پارامترها)
 دستورات بدنه
End Sub
```

همانطور که مشاهده می کنید، تعریف یک Sub با کلمهٔ کلیدی Public یا Private شروع می شود. این کلید واژه اختیاری بوده و حوزهٔ عملکرد زیربرنامه را مشخص می کند؛ درصورتی که هیچ یک از آنها را ننویسید، ویژوال بیسیک کلید واژهٔ Private را به طور پیش فرض درنظر خواهد گرفت. اگر از Private استفاده کنید روال مورد نظر فقط در فایلی که تعریف می شود، قابل دسترس و استفاده است؛ ولی اگر از Public استفاده کنید، می توانید در فایل های دیگر برنامه نیز آن را بکار ببرید.

پس از تعیین محدودهٔ قابل دسترس بودن زیربرنامه، از کلید واژهٔ Sub استفاده کرده و سپس پارامترهای مورد نیاز را مشخص می کنیم.

پارامتر ها، همان متغیرهایی هستند که وجود آنها درون زیربرنامه اهمیت دارد. به عبارت دیگر، محاسبه و اجرای دستورات بدنهٔ زیربرنامه، وابسته به مقادیر موجود در این متغیرهاست؛ بنابراین باید این مقادیر را به زیربرنامه ارسال کنیم. برای تعریف پارامتر ها باید به نکات زیر توجه کرد :

۱) پارامترها را باید همانند تعریف یک متغیر، مشخص کرد؛ با این تفاوت که در اینجا از کلید واژهٔ Dim نباید استفاده شود.

۲) اگر زیربرنامه ای که می نویسیم پارامتری نداشت، باید بعد از نامی که برای آن در نظر گرفتیم، باید یک بار پرانتز را باز و سپس ببندیم تا مشخص شود زیربرنامهٔ مورد نظر هیچ پارامتری ندارد.

۳) اگر دو یا چند پارامتر از یک نوع داده داشتید باید هر کدام را به صورت جداگانه تعریف کرده و با استفاده از **","** (کاما) آنها را از هم جدا کنید. برای مثال اگر دو متغیر a و b هر دو از نوع عدد صحیح باشد، نباید آنها را به صورت a,b As Integer تعریف کرد، بلکه هر کدام باید بصورت جدا تعریف شود و بین آنها کاما قرار داد.

۴) هنگام استفاده از پارامترها یکسان بودن نوع و ترتیب آنها بسیار مهم است. برای مثال اگر در زیربرنامه ای دو پارامتر که اولی از نوع رشته و دومی از نوع عدد صحیح است، در نظر گرفته باشیم؛ نباید زمان فراخوانی تابع، ابتدا عدد صحیح و سپس رشته مورد نظر را به زیربرنامه ارسال کنیم. همچنین نباید بیشتر یا کمتر از همان دو پارامتر به زیربرنامه ارسال کنیم.

به طور کلی پارامترها به دو صورت زیر تعریف می شوند :

۱) مقداری (byVal)

۲) ارجاعی (byRef)

وقتی از byVal استفاده کنید مقداری که برای پارامترِ زیربرنامه ارسال می کنید پس از پایان روال، تغییری نمی کند ولی با استفاده از عبارت byRef دقیقاً عکس این موضوع اتفاق خواهد افتاد. در ویژوال بیسیک بطور پیش فرض پارامترها بصورت byRef ارسال می شود.

به مثالهای زیر توجه کنید :

مثال ۱)

```
Private Sub Sample1()
 Dim x As Integer
 x = Val(InputBox("Enter Number"))
 MsgBox "X = " & x
End Sub
```

با اجرای این روال یک مقدار عددی توسط تابع InputBox دریافت شده و سپس با استفاده از تابع MsgBox نمایش داده می شود.

برای استفاده از یک Sub تعریف شده، باید آن را بصورت زیر صدا کنیم :

```
[Call]  اسم روال (لیست پارامترها)
```

مثلاً برای صدا کردن روال تعریف شدهٔ فوق، کد زیر را بکار می بریم:

```
Call Sample1
```

مثال ۲)

```
Private Sub Sample2(X AS Integer)
 Dim y  As Integer
 y = Val(InputBox("Enter Number"))
 x = x + y
 MsgBox "Sum = " & x
End Sub
```

این تابع یک پارامتر را دریافت کرده و آن را با متغیری که در بدنهٔ آن تعریف و مقداردهی شده، جمع می کند و سپس مقدار نهایی آن را با استفاده از تابع MsgBox به کاربر نمایش می دهد. همچنین، برای فراخوانی زیربرنامهٔ فوق به صورت زیر عمل خواهیم کرد :

```
Dim A As Integer
A = Val(InputBox("Enter Number"))
Call Sample2(A)
```

اگر دقت کرده باشید برای تعریف پارامتر در این مثال، کلید واژهٔ byRef یا byVal را بکار نبردیم؛ پس همانطور که قبلاً توضیح داده شد، ویژوال بیسیک نوع پارامتر x را از نوع byRef تعریف می کند. برای اثبات این موضوع می توانید قطعه برنامه فوق را به صورت زیر بنویسیم :

```
Dim A As Integer
A = Val(InputBox("Enter Number"))
Call Sample2(A)
MsgBox "A = " & A
```

در این قطعه کد، پس از فراخوانی روال Sample2 مقدار متغیر A را به وسیلهٔ تابع MsgBox نشان می دهیم تا ببینیم آیا همان مقدار اولیه را دارد یا خیر!؟ با اجرای و تست برنامه متوجه خواهیم شد که مقدار A به همان مقداری که متغیر x در بدنهٔ روال Sample2 دارد، تغییر خواهد کرد که دلیل آن استفاده از نوع byRef در تعریف پارامتر می باشد.

مثال ۳)

```
Private Sub Sample3(ByVal x As Integer, ByRef y As Integer)
 x = x + y
 y = x
 MsgBox "Sum = " & x
End Sub
```

این تابع دو پارامتر دریافت کرده و مجموع آنها را چاپ می کند. ضمناً مقدار هر دو پارامتر، درون این تابع برابر با این مقدار (مجموع دو پارامتر) خواهد شد؛ اما با توجه به اینکه پارامتر اول از نوع byVal بوده، مقدار متغیری که هنگام فراخوانی به زیربرنامه ارسال می شود، پس از اجرای زیربرنامه، تغییر نمی کند؛ بلکه تنها برروی پارامتر دوم که از نوع byRef بوده این تغییر را اعمال خواهد شد. برای اثبات این موضوع می توان ابتدا زیربرنامهٔ فوق را فراخوانی و سپس مقدار متغیرهای ارسال شده به آن را به صورت زیر نمایش داد تا متوجه تغییر یا عدم تغییر هر کدام شویم. برای این کار کدهای زیر را تایپ کنید :

```
Dim A As Integer
A = Val(InputBox("Enter First Number"))
Dim B As Integer
B = Val(InputBox("Enter Second Number"))
Call Sample3(A, B)
MsgBox "A = " & A
MsgBox "B = " & B
```

زیربرنامهٔ تابع (Function)

تابع نیز همانند یک روال عمل می کند با این تفاوت که می تواند از طریق نام خود یک مقدار را به عنوان نتیجه به برنامه فراخوان برگرداند. شکل کلی این دستور به صورت زیر است :

```
[Public|Private]  Function  نام زیربرنامه(پارامترها) As  نوع داده
 دستورات بدنه
End Function
```

در تعریف Function مانند تعریف Sub ابتدا محدودهٔ قابل دسترس بودن تابع را مشخص می کنیم (Private یا Public) ولی بجای کلید واژهٔ Sub از کلید واژهٔ Function استفاده کرده و بعد از آن، نام تابع را تایپ می کنیم. سپس در صورتی که نیاز به پارامتر داشته باشیم، آنها را درون پرانتز تعریف می کنیم. در انتها، از آنجایی که Function حتما یک مقدار برگشتی (از طریق نام تابع) دارد، باید نوع دادهٔ آن را مشخص کنیم. به مثال زیر توجه کنید :

```
Public Function Sum(X AS Integer, Y AS Integer) As Integer
 Sum = X + Y
End Function
```

دقت کنید که **نمی توان** از کلید واژهٔ Call برای فراخوانی یک تابع استفاده کرد؛ بلکه از آنجایی که تابع حتماً یک مقدار برگشتی دارد، باید آن را به مشخصهٔ یک کنترل یا یک متغیر و... نسبت داد. مثلاً به صورت زیر می توانیم از تابع فوق (Sum) استفاده کنیم :

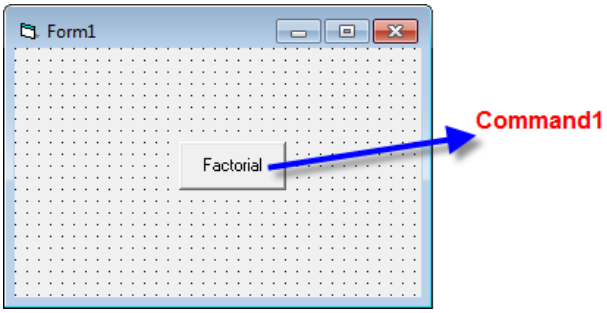
```
Dim A,B As Integer
A = Val(InputBox("Enter First Number"))
B = Val(InputBox("Enter Second Number"))
MsgBox Sum(A, B)
```

* * *

مثال :

برنامه ای بنویسید که با استفاده از تابع، فاکتوریل یک عدد را محاسبه کند و نتیجه را با استفاده از تابع MsgBox به کاربر نمایش دهد.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



ابتدا تابعی به نام Fact برای انجام عمل فاکتوریل می نویسیم :

```
Function Fact(byVal n As Integer)

 Dim i, tmp As Integer
 tmp = 1

 For i = 1 To n
  tmp = tmp * i
 Next

 Fact = tmp

End Function
```

تابع فوق، عدد مورد نظر را به وسیله پارامتر (متغیر n) دریافت کرده و پس از محاسبهٔ فاکتوریل آن، نتیجه را به وسیلهٔ نام خود بر می گرداند. باید توجه داشت که اگر خط آخر (Fact = tmp) را ننویسیم، نتیجهٔ درست به برنامه فراخوان برگردانده نخواهد شد.

برای فراخوانی تابع نوشته شده، روی Command1 دابل کلیک کرده و کد های زیر را تایپ کنید :

```
Private Sub Command1_Click()
 Dim x As Integer
 x = Val(InputBox("Enter Number"))
 MsgBox Fact(x)
End Sub
```

در قطعه برنامه فوق، مقدار متغیر X به وسیلهٔ تابع InputBox دریافت شده و به تابع Fact منتقل می شود؛ سپس نتیجه برگشت داده شده به وسیلهٔ تابع MsgBox به کاربر نشان داده می شود.

تابع بازگشتی

به عبارت ساده، تابعی که خود را فرا بخواند تا به نتیجه نهایی برسد، تابع بازگشتی نامیده می شود. برای درک بهتر این مسئله به مثال زیر توجه کنید :

* * *

فرض کنید، قصد داریم فاکتوریل عدد ۵ را محاسبه کنیم. همانطور که می دانید فاکتوریل عدد ۵ می شود :

$$5 \times 4 \times 3 \times 2 \times 1 = 120$$

پس با کمی دقت می توان گفت که فرمول زیر هم در مورد عبارت فوق صدق می کند :

$$5 \times (\text{فاکتوریل عدد ۴}) = 120$$

زیرا فاکتوریل عدد ۴ همان $4 \times 3 \times 2 \times 1$ است که نتیجهٔ آن ۲۴ می شود. اکنون برای محاسبهٔ فاکتوریل ۴ می توان فرمول زیر را بکار برد :

$$4 \times (\text{فاکتوریل عدد ۳}) = 24$$

و همین طور فرمول زیر را برای فاکتوریل عدد ۳ :

$$3 \times (\text{فاکتوریل عدد ۲}) = 6$$

و باز هم فرمول زیر را برای فاکتور عدد ۲ :

$$2 \times (\text{فاکتوریل عدد ۱}) = 2$$

و نهایتاً فاکتوریل عدد ۱ که نتیجه همان ۱ می شود.

* * *

همانطور که مشاهده کردید، در مثال فوق، برای محاسبهٔ فاکتوریل یک عدد از ضرب همان عدد در فاکتوریل یک واحد کمتر از آن، استفاده کردیم. اگر این فرمول را به صورت یک تابع پیاده سازی کنیم، به آن تابع بازگشتی گفته می شود.

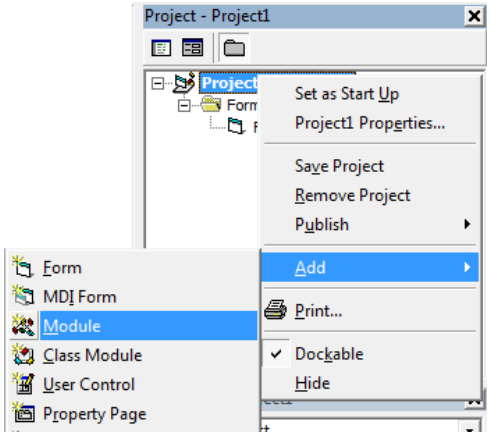
اکنون که مفهوم تابع بازگشتی را درک کردید، به نمونه ای از یک تابع بازگشتی که درواقع پیاده سازی توضیحات فوق است، می پردازیم :

```
Function Fact(n As Integer)

 If n > 1 Then
  Fact = n * Fact(n-1)
 Else
  Fact = 1
 End If
End Function
```

در قطعه کد فوق، مقدار پارامتر (متغیر n) بررسی شده و در صورتی که از عدد یک بزرگتر باشد، مقدار برگشتی تابع حاصل ضرب همان عدد در فاکتوریل یک واحد کمتر از آن، خواهد شد؛ درغیر اینصورت (اگر برابر با یک باشد)، مقدار برگشتی تابع نیز یک می شود.

استفاده از ماژول ها

برای سهولت کار در پروژه های چند فرمی، می توان از ماژول ها (Module) استفاده کرد. برای درک بهتر این موضوع، برنامه ای را در نظر بگیرید که دارای چند فرم بوده و ما قصد داریم چند روال یا تابع را که از قبل نوشته ایم در تمام فرم ها بکار ببریم. در این صورت بهتر است آنها را درون یک ماژول نوشته و در سایر فرم ها فقط آنها را فراخوانی کنیم. برای اینکار در پنجرهٔ Project روی نام پروژه کلیکِ راست و از منوی Add گزینهٔ Module را انتخاب کنید و در کادر باز شده دکمهٔ Open را بزنید.



اکنون در فایل ایجاد شده، می توانید توابع یا روال های مورد نظرتان را بنویسید و در فرم های دیگر فراخوانی کنید. البته باید این نکته را در نظر بگیرید که در تعریف آنها حتما باید از کلید واژهٔ Public استفاده کنید تا در فرم های پروژه قابل دسترس باشند.

تمرین :

۱- برنامه ای بنویسید که دو عدد را از ورودی دریافت کرده و با استفاده از یک تابع بازگشتی، عدد کوچک را به توان عدد بزرگتر برساند.

۲- برنامه ای بنویسید که با استفاده از تابع InputBox یک عدد را از ورودی دریافت کرده و سپس با استفاده از یک تابع، مغلوب آن را روی فرم چاپ کند.

۳- برنامه ای بنویسید که عددی را دریافت از ورودی خوانده و با استفاده از یک زیربرنامهٔ روال، مجموع اعداد ییک تا آن عدد را روی فرم چاپ کند.

فصل هفتم :

کار با فایل ها

آشنایی با فایل ها و انواع آن

داده های مورد نیاز در برنامه هایی که تاکنون نوشته ایم، در متغیرهای معمولی و آرایه ها ذخیره و مورد پردازش قرار گرفته اند. متغیرهای معمولی و آرایه ها همگی در حافظه RAM قرار دارند، لذا پس از خاموش شدن کامپیوتر (و یا قطع جریان برق) و یا خروج از برنامه، داده هایی که در آنها ذخیره شده اند از بین رفته و برای استفاده مجدد از آنها باید دوباره وارد شوند. این کار قطعاً مقرون به صرفه نیست؛ زیرا نه تنها مستلزم صرف وقت زیادی است، بلکه حوصله انجام کار را نیز از کاربر سلب می نماید. برای رفع این مشکل، اطلاعات وارد شده باید بر روی حافظه جانبی (مثل هارد دیسک) ذخیره شوند. چون اطلاعات موجود در حافظه جانبی با قطع جریان برق یا خروج از برنامه، از بین نمی روند و به دفعات مورد استفاده قرار می گیرند. از فایل ها می توان برای ذخیرهٔ اطلاعات با اندازه و قالب متفاوت استفاده کرد. برای این کار، برنامه باید فایلی را باز کرده و اطلاعات مور نظر را در آن ذخیره کند و یا از آن بخواند.

به فایلی که یک برنامه پس از گشودن، اطلاعات را در آن ذخیره می کند، فایل خروجی و به فایلی که پس از گشوده شدن، فقط اطلاعات از آن خوانده می شود، فایل ورودی می گویند. همچنین وقتی یک فایل طوری باز شود که عمل خواندن و نوشتن را بتوان انجام داد، فایل ورودی–خروجی نامیده می شود.

فایل ها در ویژوال بیسیک، از نظر نوع دسترسی و ذخیره سازی اطلاعات به سه دسته تقسیم می شوند :

- ترتیبی (Sequential)
- تصادفی (Random)
- دودویی (Binary)

فایل ترتیبی ساده ترین نوع فایل است اما دارای معایبی نیز می باشد که یکی از آنها کُند بودن کار با این نوع فایل هاست. کار با فایل های تصادفی سرعت بیشتری دارد اما پیچیدگی بیشتری را به برنامه تحمیل می کند. فایل دودویی نوع خاصی از فایل های تصادفی هستند و عموماً کار با این فایل ها برای کامپیوتر ساده تر است، چرا که زبان باینری، زبانی است که کامپیوتر با آن آشناست و در نتیجه سرعت پردازش اطلاعات در این حالت بیشتر خواهد بود.

دستور Open

همانطور که در ابتدای این فصل توضیح دادیم، برای کار با فایل ها (ذخیره و بازیابی اطلاعات) باید ابتدا آن را باز کنیم. برای این کار در ویژوال بیسیک از دستور Open استفاده می شود که شکل کلی آن به صورت زیر است :

```
Open مسیر فایل [For حالت دسترسی] [Access نوع دسترسی] [LockType] As [#]شماره فایل
```

نکاتی در رابطه با دستور Open

۴- مسیر فایل باید آدرس دقیق فایل باشد؛ مانند : "D:\Samples\Project1\MyFile.txt"

۵- حالت دسترسی به فایل می تواند یکی از حالت های موجود در جدول زیر باشد :

حالت	شرح
Append	فایل ترتیبی موجود را برای اضافه کردن داده به انتهای آن باز می کند.
Binary	فایل را به صورت دودویی باز می کند.
Input	فایل ترتیبی را برای خواندن باز می کند.
Output	فایل ترتیبی را برای نوشتن باز می کند.
Random	فایل را برای دسترسی تصادفی باز می کند.

۶- نوع دسترسی می تواند Read، Write یا Read Write باشد.

۷- LockType تعیین می کند که آیا همزمان با این که برنامه روی فایل کار می کند، دیگران می توانند فایل را بخوانند یا خیر؟! مقادیری که پشتیبانی می کند، عبارتند از : Shared، Lock Read، Lock Write و Lock Read Write .

۸- همچنین شماره فایل نیز، شمارهٔ File Handle است که هم می توان آن را به صورت مستقیم مقداردهی کرد (مانند ۱، ۲ و ...) و هم با استفاده از تابع FreeFile() . این تابع، اولین شمارهٔ فایل آزاد را برمی گرداند. دلیل استفاده از شمارهٔ فایل در دستور Open این است که اگر بخواهیم چند فایل را در ویژوال بیسیک همزمان مدیریت کنیم، باید شمارهٔ فایل های باز را بدانیم تا دستورات مربوط به هر فایل به درستی اجرا شوند.

توجه داشته باشید که به جز حالت Input در سایر حالات، اگر فایل وجود نداشته باشد، ویژوال بیسیک آن را ایجاد می کند.

به مثال های زیر توجه کنید :

```
Open "D:\Sample1.txt" For Output As #1
```

فایل Sample1.txt را در درایو D برای نوشتن باز می کند.

```
Dim FileNumber AS Integer
FileNumber = FreeFile()
Open "D:\Sample1.txt" For Input As #FileNumber
```

درون متغیر FileNumber شمارهٔ اولین فایل آزاد را قرار داده و فایل Sample1.txt را برای خواندن باز می کند.

```
Dim FN AS Integer
FN = FreeFile()
Open "D:\Sample1.txt" For Output Lock Write AS #FN
```

فایل را طوری برای نوشتن باز می کند که در هنگام استفاده از فایل توسط برنامه، نمی توان به صورت جداگانه درون آن چیزی نوشت.

دستور Close

هر فایلی که باز شده باید بعد از استفاده بسته شود. دستور بستن فایل به صورت زیر است :

```
Close #شماره فایل۱ [, شماره فایل۲, ..., شماره فایل x]
```

تعداد فایلهایی که می توانید در این دستور قید کنید، محدودیتی ندارد و اگر هیچ شماره ای قید نشود تمام فایل های باز، بسته خواهند شد.

کار با فایل های ترتیبی

در فایل های ترتیبی رکوردها به همان ترتیبی که از ورودی خوانده می شوند، در فایل نوشته می شوند و در هنگام خواندن فایل نیز باید به همان ترتیبی که ذخیره شده اند، خوانده شوند. بزرگترین نقطه ضعف فایل های ترتیبی خواندن و نوشتن است؛ زیرا برای استفاده از این قبیل فایل ها باید تمام بایت های دیگر را هم بخوانید. مثلاً اگر فایلی با ۱۰۰۰ بایت داشته باشیم و فقط بخواهیم یک بایت آن را تغییر دهیم، باید تمام بایت های دیگر را هم، دوباره بخوانیم و بنویسیم. اما فایل های ترتیبی برای ذخیره کردن فایل های متنی کوچک که در آن سرعت چندان مهم نیست، مناسب است.

دستور Print

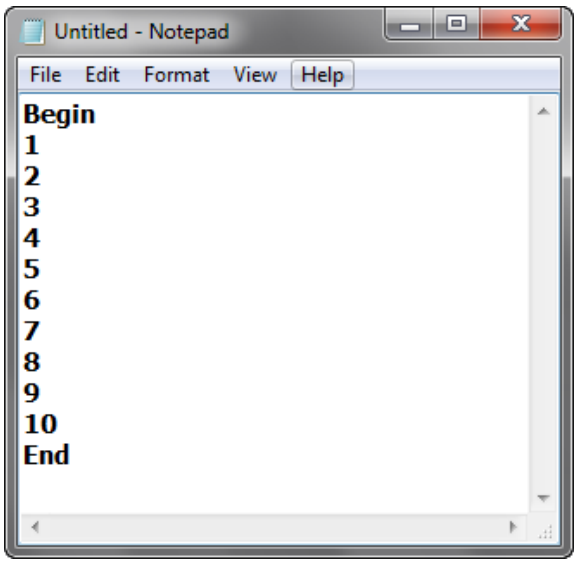
همانطور که در طول این فصل اشاره شد، قبل از استفاده از هر فایلی باید آن را باز کنید. بعد از باز کردن فایل، می توانید اطلاعات را در آن بنویسید. یکی از روش های نوشتن در فایل استفاده از دستور Print است. با این دستور فقط در فایل های ترتیبی می توان نوشت. شکل کلی آن به صورت زیر است :

```
Print #شماره فایل , عبارت عددی یا رشته ای
```

به مثال زیر توجه کنید :

```
Dim i, F As Integer
F = FreeFile()
Open "D:\MyFile.txt" For Output As #F
Print #F, "Begin"
For i = 1 To 10
 Print #F, i
Next
Print #F, "End"
Close #F
```

این قطعه برنامه، فایلی با نام MyFile.txt ایجاد کرده و اعداد ۱ تا ۱۰ را در آن می نویسید. خروجی برنامهٔ فوق به صورت زیر است :



دستور Write

دستور Write فرمان دیگری برای نوشتن در فایل هایی ترتیبی می باشد. شکل کلی آن چنین است :

```
Write #شماره فایل , OutputList
```

که در آن OutputList فهرست متغیرهایی است که باید در فایل مورد نظر نوشته شوند.

به مثال زیر توجه کنید :

```
Dim i, F As Integer
F = FreeFile()
Dim fn, ln As String
fn = InputBox("Enter First Name")
ln = InputBox("Enter Last Name")
Open "D:\List.txt" For Append As #F
Write #F, fn, ln
Close #F
```

قطعه برنامهٔ فوق ابتدا نام و نام خانوادگی یک شخص را از ورودی دریافت کرده و سپس فایل List.txt را در درایو D به گونه ای باز می کند که قادر باشد به انتهای آن اطلاعات مورد نظر را اضافه کند. پس از این کار، اطلاعات دریافت شده را به انتهای فایل اضافه کرده و در نهایت آن را می بندد.

دستور Input و Line Input

بعد از نوشتن اطلاعات در یک فایل، باید بتوانید آن ها را دوباره بازیابی کنید. خواندن یک فایل ترتیبی دقیقاً به همان ترتیب نوشتن آن صورت می گیرد. بازیابی داده ها از فایل خیلی شبیه نوشتن داده ها در آن است. تنها تفاوت این است که به جای استفاده از حالت های Append، Output، Binary یا Random، حالت Input را بکار می بریم. یکی از دستور خواندن فایل های ترتیبی، دستور Input است که شکل کلی آن به صورت زیر می باشد:

```
Input #شماره فایل , متغیر۱, متغیر۲, ..., متغیرn
```

نوع داده متغیرها باید با دستور Print که این مقادیر را در فایل نوشته است، مطابقت داشته باشد اما از آنجایی که داده ها با دستور Write به وسیلهٔ کاما از هم جدا خواهند شد، انطباق Input و Write ضروری نیست.

همچنین، دستور Line Input نیز برای خواندن خط به خط اطلاعات از فایل ها، بکار می رود که شکل کلی آن به صورت زیر است :

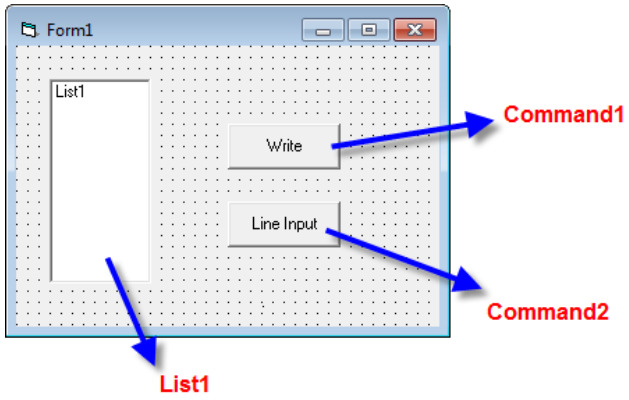
```
Line Input #شماره فایل, متغیر
```

* * *

مثال :

برنامه ای بنویسید که با کلیک بر روی دکمهٔ اول، یک مقدار عددی را از کاربر دریافت کرده و به انتهای یک فایل ترتیبی اضافه کند. سپس با کلیک بر روی دکمهٔ دوم محتوای فایل را درون یک ListBox منتقل کند.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



ابتدا باید اطلاعات را درون فایل بنویسیم، بنابراین روی دکمهٔ Command1 دابل کلیک کنید و کدهای زیر را بنویسید :

```
Private Sub Command1_Click()
 Dim F As Integer
 F = FreeFile()
 Open "D:\File.txt" For Append As #F
 Dim Number As Integer
 Number = Val(InputBox("Enter Number"))
 Write #F, Number
 Close #F
End Sub
```

در قطعه کد بالا، ابتدا فایل مورد نظر را با حالت Append باز کردیم و سپس مقدار متغیر Number را از کاربر گرفته و آن را درون فایل نوشتیم و در نهایت فایل را بستیم.

اکنون کد های زیر را درون Command2 تایپ کنید :

```
Private Sub Command2_Click()
 Dim F As Integer
 F = FreeFile()
 Open "D:\List.txt" For Input As #F
 Dim text As String
 Do While Not EOF(F)
  Line Input #F, text
  List1.AddItem text
 Loop
 Close #F
End Sub
```

قطعه برنامهٔ فوق، فایل مورد نظر را به حالت Input باز کرده و در حلقهٔ تکرار تا زمانی که به آخر فایل نرسیده، آن را خط به خط می خواند و محتوای هر خط را ابتدا درون متغیر Text ذخیره کرده و سپس بلافاصله مقدار ان متغیر را به کنترل List1 اضافه می کند. پس از اتمام کار، فایل را با استفاده از دستور Close می بندد.

* * *

کار با فایل های تصادفی

فایل تصادفی، فایلی است که در هر نقطه از آن بدون رعایت ترتیب همزمان می توان نوشت یا از آن خواند. فایل ترتیبی اعداد را به صورت کد اسکی ذخیره می کند و برای هر رقم یک بایت فضا در نظر می گیرد. اما در فایل تصادفی، اعداد به صورت دودویی ذخیره می شوند؛ لذا اطلاعات فضای کمتری از حافظه جانبی را اشغال می کنند. به عنوان مثال، عدد ۲۰۰۰۰ در فایل ترتیبی ۵ بایت اشغال می کند، در صورتی که فایل تصادفی برای عدد مذکور ۲ بایت در نظر می گیرد. رکورد ها در فایل تصادفی برخلاف فایل ترتیبی دارای طول یکسانی هستند.

علاوه بر این مزیت ها، دلیل اولیهٔ اهمیت فایل های تصادفی، معرفی نوع جدید از انواع داده به وسیله کاربر است که اختصاراً آن را UDT یا User-defined Data Type می نامند. این نوع داده به برنامه نویس کمک می کند تا با ترکیب داده های ذاتی Visual Basic، انواع داده ای جدیدی ایجاد کرده و به عنوان یک رکورد به وسیلهٔ یک متغیر در یک فایل تصادفی ذخیره کند.

نوع دادهٔ تعریف شده به وسیلهٔ کاربر

به این انواع داده، ساختار (Structure) یا رکورد (Record) نیز گفته می شود. برای استفاده از رکوردها، ابتدا باید نوع رکورد را ایجاد کـرده و سپس متغیرهایی را از نوعِ آن تعریف و استفاده کرد. هر نوع رکورد، اجزایی دارد که فیلد نامیده می شوند. برای تعریف رکورد به صـورت زیـر عمل می شود :

```
[Public|Private] Type نـام نـوع داده جدیـد
  فیـلـد۱ As نـوع داده
  فیـلـد۲ As نـوع داده
  :
  فیـلـدn As نـوع داده
End Type
```

نام نوع داده جدید، از قانون نام گذاری متغیرها تبعیت می کند. برای استفاده از نوع داده ای که تعریف می کنید، بایـد یـک متغیـر را از آن نوع اعلان کرده و برای مقداردهی و استفاده از مقادیر موجود در فیلدهای آن از فرمول زیر استفاده کنید :

```
'اعلان متغیر مورد نیاز :
Dim نام متغیر As نوع داده جدید

'مقداردهی فیلد ها :
مقدار = فیلد مورد نظر.نام متغیر

'استفاده از مقادیر :
[نتیجه] = فیلد مورد نظر.نام متغیر
```

در ادامه مثال کاملی از کاربرد رکوردها در برنامه توضیح خواهیم داد.

دستورهای Put و Get

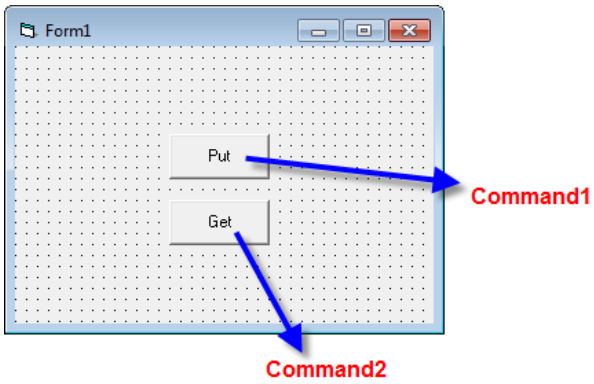
برای خواندن و نوشتن فایل های تصادفی از دو دستور Get و Put استفاده می شـود. ایـن دو دسـتور معـدل دو دسـتور Input و Print در فایل های ترتیبی هستند. ولی تفاوت کوچکی بین این دستورها وجود دارد. در دستورهای Input و Print راهی وجود ندارد تا نقطه ای از فایل برای خواندن یا نوشتن مشخص شود، در حالی که دستورهای Get و Put دارای چنین امکانی هستند :

```
Put #شماره فـایـل, [شماره رکـورد مـوردنـظر,] مـتـغـیـر
Get #شماره فـایـل, [شماره رکـورد مـوردنـظر,] مـتـغـیـر
```

مثال :

یک نوع داده به نام Student تعریف کنید که شامل فیلدهای ID ، FName و LName باشد. سپس برنامه ای بنویسید که با کلیک بر روی دکمهٔ اول، یک متغیر از نوع Student تعریف کرده و ۵ بار فیلدهای آن را مقداردهی و رکورد را درون یک فایل تصادفی ذخیره کنید. سپس با کلیک بر روی دکمهٔ دوم، مقدار فیلدهای ۵ رکورد (ذخیره شده در فایل) را با استفاده از تابع MsgBox نشان دهید.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



ابتدا در بخش General نوع داده مورد نظر را تعریف می کنید :

```
Private Type Student
 ID As Integer
 FName As String * 10
 LName As String * 20
End Type
```

همانطور که مشاهده می کنید، نوع داده ای با نام Student تعریف کرده ایم که دارای سه فیلد می باشد که فیلد ID از نوع Integer و فیلدهای FName و LName هر دو از نوع String هستند که طول آنها به ترتیب ۱۰ و ۲۰ کاراکتر خواهد بود.

نکته :

هنگام نوشتن نوع دادهٔ کاربر در یک فایل (به دلیل اینکه اندازهٔ هر رکورد باید مشخص باشد) فیلدهای رشته ای باید اندازهٔ ثابت داشته باشند.

اکنون بر روی دکمهٔ Command1 دابل کلیک کرده و کدهای زیر را تایپ کنید :

```
Private Sub Command1_Click()
 Dim F As Integer
 F = FreeFile()
 Open "D:\RandFile.txt" For Random As #F

 Dim std As Student
 For i = 1 To 5
  std.ID = Val(InputBox("Enter ID"))
  std.FName = InputBox("Enter First Name")
  std.LName = InputBox("Enter First Name")
  Put #F, i, std
 Next

 Close #F
End Sub
```

در قطعه کد فوق، ابتدا فایل مورد نظر به حالت تصادفی باز می شود. سپس متغیر std از نوع دادهٔ تعریف شده (Student) اعلان و در یک حلقهٔ تکرار، ۵ بار مقداردهی می شود و هر بار در فایل مورد نظر ذخیره می گردد. در انتها فایل نیز بسته خواهد شد.

حال روی Command2 دابل کلیک کرده و کدهای زیر را وارد کنید :

```
Private Sub Command2_Click()
 Dim Rec As Student
 Dim F, i As Integer
 F = FreeFile
 i = 1

 Open "D:\RandFile.txt" For Random As #F

 Do While Not EOF(F)
  Get #F, i, Rec
  i = i + 1
  MsgBox "Student" & Rec.ID & " is " & Rec.FName & Rec.LName
 Loop

End Sub
```

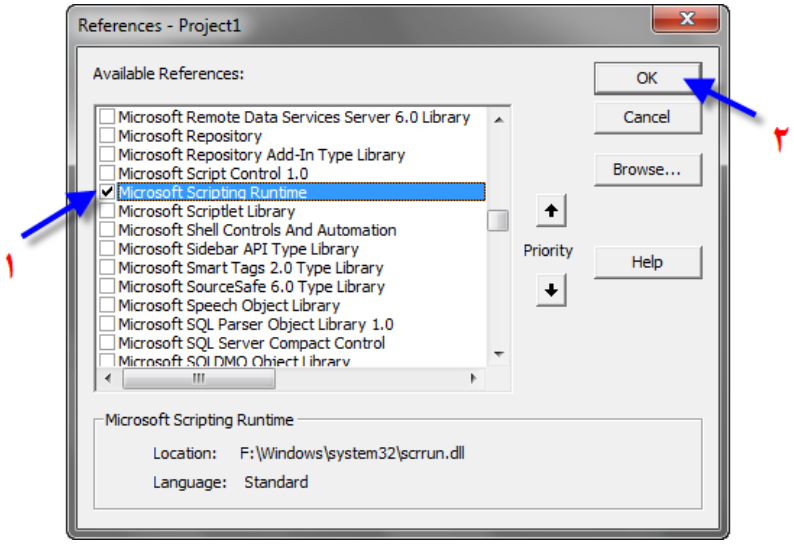
قطعه برنامهٔ فوق، پس از باز کردن فایل مورد نظر، تا زمانی که به آخرین خط نرسیده، محتوای آن را رکورد به رکورد می خواند و درون رکورد Rec که از نوع Student تعریف شده است، ذخیره می کند. سپس بلافاصله مقادیر فیلدهای این رکورد را نمایش می دهد.

کار با File System Object

این شئ که به FSO نیز معروف است، امکان کار کردن با سیستم فایل کامپیوتر (Drive، Folder، File و TextStream) را فراهم می کند. قبل از استفاده از شئ سیستم فایل باید DLL مربوطه را به IDE ویژوال بیسیک اضافه کنید.

فایل DLL، فایلی است که حاوی کتابخانه هایی از Data و Code می باشد که بصورت دینامیک و پویا با برنامه ارتباط برقرار می کند. از مهمترین دلایل استفاده از فایل های DLL، این است که با استفاده از این فایل در برنامه، حجم فایل اجرایی آن کاهش پیدا می کند. در مورد این فایل ها و طرز ساخت آنها در جلد دوم کتاب بطور کامل توضیح خواهیم داد.

برای اضافه کردن DLL مربوط به کار با شئ File System Object ابتدا از منوی Project گزینهٔ References را انتخاب کنید تا کادر مربوطه ظاهر شود. در این کادر گزینهٔ Microsoft Scripting Runtime را یافته و آن را انتخاب کنید. با کلیک روی دکمهٔ OK عملیات را تائید کرده و کادر را ببندید.



در مرحلهٔ بعد باید یک متغیر از نوع شئ File System Object با استفاده از کلید واژهٔ New تعریف کنید تا در زمان اجرا یک شئ جدید از آن ایجاد شود :

```
Dim Fso As New FileSystemObject
```

برای استفاده از متدهای این شئ باید از متغیری که تعریف می کنید، استفاده کنید.

این شئ متدهای زیادی دارد که از متدهای مهم آن، می توان موارد زیر اشاره کرد :

- **متد FileExisting / FolderExists**

این دستورات موجودیت یا عدم موجودیت یک فایل یا پوشه را بررسی می کنند و با توجه به نتیجه بررسی، یک مقدار منطقی (True یا False) را برمی گردانند. به مثال های زیر توجه کنید :

```
Dim Fso As New FileSystemObject
Dim a AS Boolean
a = Fso.FileExisting("D:\MyFolder\MyList.txt")
If a = True Then
 MsgBox "فایل وجود دارد"
Else
 MsgBox "فایل وجود ندارد"
End If
```

قطعه کد فوق، موجودیت فایل MyList.txt را در مسیر D:\MyFolder بررسی کرده و پیغام مناسب را ظاهر می کند. همچنین قطعه کد زیر همین عمل را نسبت به پوشهٔ Sample در مسیر D:\Projects انجام خواهد داد :

```
Dim Fso As New FileSystemObject
Dim b AS Boolean
b = Fso. FolderExists("D:\Projects\Sample")
If b = True Then
 MsgBox "پوشه وجود دارد"
Else
 MsgBox "پوشه وجود ندارد "
End If
```

- **متد CopyFile / CopyFolder**

همانطور که از اسم این متدها پیداست، به منظور کپی کردن یک فایل یا پوشه از آنها استفاده می شود. شکل کلی آن چنین است :

```
(fso متغیر از نوع شئ).CopyFile  مسیر فایل منبع, مسیر فایل مقصد[, بازنویسی]
(fso متغیر از نوع شئ). CopyFolder  مسیر فایل منبع, مسیر فایل مقصد[, بازنویسی]
```

مقداری که برای پارامتر آخر باید در نظر بگیرید مقدار True یا False است. در صورت استفاده از مقدارTrue اگر فایل یـا فولـدر در مـسیر مقصد وجود داشته باشد، آن را حذف و فایل یا فولدر جدید را به جای آن قرار می دهد؛ و در صورت استفاده از False اگر فایـل یـا فولـدر در مسیر مقصد وجود داشته باشند، عمل کپی لغو خواهد شد. مقدار این پارامتر، به صورت پیش فرض True می باشد.

به مثال های زیر توجه کنید :

```
If Fso.FileExists("D:\Project1\Student.txt") = True Then
 Fso.CopyFile "D:\Project1\Student.txt", "D:\Project2\Student2.txt"
 MsgBox "کپي انجام شد"
Else
 MsgBox "فـايـل وجود نـدارد"
End If
```

قطعه کد فوق، ابتدا بررسی می کند که فایل Student.txt در مسیر D:\Project1 وجود دارد یا خیر؟! در صورت وجود داشتن این فایل، یک کپی از آن در مسیر D:\Project2\Student.txt می گیرد؛ و در صورت عدم وجود آن، پیغام مناسب را چاپ می کند. همچنین قطعه کـد زیـر، همین عمل را نسبت به پوشهٔ MyFolder انجام خواهد داد :

```
If Fso.FolderExists("D:\Project1\MyFolder") = True Then
  Fso.CopyFolder "D:\Project1\MyFolder", "D:\Project2\MyFolder2"
  MsgBox "کپي انجام شد"
Else
 MsgBox "پـوشه وجود نـدارد"
End If
```

- **متد DeleteFile / DeleteFolder**

از این دستورات برای حذف یک فایل یا فولدر استفاده می شود. به مثال های زیر توجه کنید :

```
If Fso.FileExists("D:\Project1\Student.txt") = True Then
 Fso.DeleteFile "D:\Project1\Student.txt", "D:\Project2\Student2.txt"
 MsgBox "فـايـل حذف شد"
Else
 MsgBox "فـايـل وجود نـدارد"
End If
```

با اجرای کدهای بالا، در صورت وجود فایل مورد نظر، آن را حذف می کند و در غیر اینصورت پیغام مناسب نمایان می شود. بـرای حـذف یک پوشه نیز می توان کدی مشابهٔ کدهای فوق نوشت و بجای متد DeleteFile از متد DeleteFolder استفاده کرد.

- **متد CreateFolder**

این دستور یک پوشه ایجاد می کند. به مثال زیر توجه کنید:

```
Fso.CreateFolder "D:\Project1\MyFolder"
```

با اجرای این دستور، پوشهٔ MyFolder در مسیر D:\Project1 ساخته می شود.

- **خواندن و نوشتن فایل متنی با استفاده از TextStream**

شئ TextStream محتوای فایل های متنی را به صورت یک رشتهٔ خیلی طولانی می خواند. بنابراین می توان از خواندن خط به خط محتوای فایل متنی، پرهیز کرد. به قطعه کد زیر توجه کنید :

```
Dim Fso As New FileSystemObject
Dim Ts AS TextStream
Dim strData As String

If Not Fso.FileExists("D:\Sample\MyFile.txt") Then
  MsgBox "فایل موجود نیست"
Else
 Set Ts = Fso.OpenTextFile("D:\Sample\MyFile.txt ")
 Do While Not Ts.AtEndOfStream
  strData = strData & Ts.Read(1)
 Loop
 Ts.Close
 MsgBox strData
End If
```

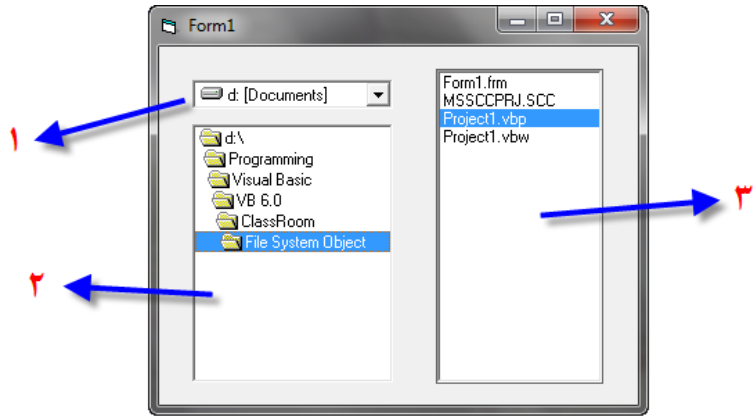
در این قطعه کد، بعد از اعلان متغیرهای مورد نیاز، ابتدا موجودیت فایل مورد نظر بررسی می شود که در صورت عدم وجود فایل، پیغام مناسب نمایان می شود. در غیر اینصورت، متغیر Ts که از نوع شئ TextStream است به وسیلهٔ دستور Set با فایل مورد نظر که به وسیلهٔ شئ FileSystemObject باز می شود، برابر خواهد شد. سپس با استفاده از دستور Read در داخل حلقهٔ تکرار که تا وقتی به انتهای فایل نرسد ادامه خواهد داشت، محتوای فایل مورد نظر خوانده شده و به محتوای متغیر رشته ای (StrData) اضافه می شود. پس از خروج از حلقه نیز مقدار متغیر که برابر با کل متن فایل خواهد بود، نمایش داده خواهد شد.

همچنین می توان از متد CreatTextFile شئ FileSystemObject برای ایجاد یک فایل استفاده کرد، سپس می توان متد Write شئ TeaxtStream را برای نوشتن داده ها در آن فایل به کار برد. کد زیر، چگونگی انجام این کار را نشان می دهد :

```
Dim Fso As New FileSystemObject
Dim Ts AS TextStream
Dim strData As String

Set Ts = Fso.CreatTextFile("D:\Sample\MyFile.txt ", True)
strData = InputBox("Enter Text")
Ts.Write(strData)
Ts.Close
```

کنترل های کار با فایل ها

ویژوال بیسیک برای مدیریت پوشه ها، درایوها و فایل ها دارای سه کنترل خاص می باشد :



۱) کنترل DriveListBox

این کنترل به کابران امکان انتخاب یک درایو را می دهد. این کنترل می تواند تمام درایوهای موجود در سیستم را شناسایی کند. درایو پیش فرض در این کنترل، درایوی است که برنامه از آنجا اجرا شده است. مشخصهٔ متداول این کنترل ، Drive است که نام درایو انتخاب شده در آن قرار می گیرد.

۲) کنترل Directory ListBox

با استفاده از این کنترل، کاربر امکان انتخاب پوشهٔ مورد نظر را خواهد داشت. پوشهٔ پیش فرض این کنترل، پوشه ای است که برنامه از آنجا اجرا شده است. مشخصهٔ متداول این کنترل، Path است که مسیر پوشه انتخاب شده را مشخص می کند.

۳) کنترل File ListBox

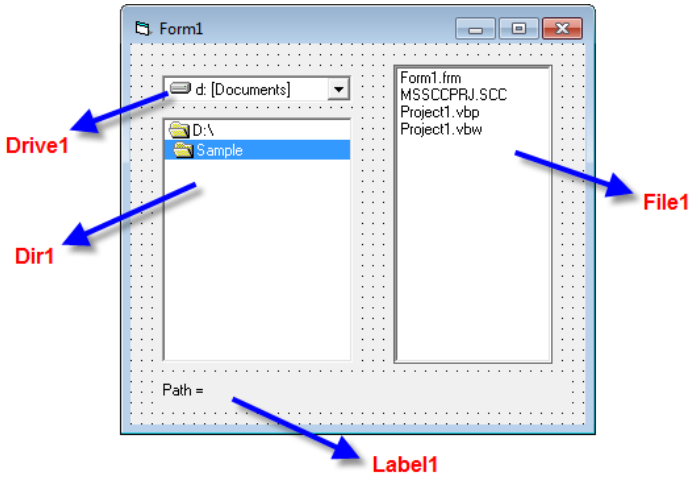
به کمک این کنترل، کاربر می تواند فایل مورد نظرش را انتخاب کند. این کنترل قادر است تمام فایل های موجود در سیستم را شناسایی کند؛ ولی می توان نوع فایل های قابل مشاهده در آن را به فرمت دلخواه تغییر داد. این کنترل، به طور پیش فرض فایل های موجود در پوشه ای که برنامه از آنجا اجرا شده را نمایش خواهد داد. مشخصه های متداول این کنترل عبارتند از :

- **FileName** : نام فایل انتخاب شده.
- **Path** : مسیر فایل انتخاب شده.
- **Pattern** : تعیین نوع فایل های قابل مشاهده در این کنترل.
- **Multi Select** : امکان انتخاب چندین فایل به صورت همزمان.

* * *

مثال :

پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



ابتدا روی Drive1 دابل کلیک کرده و کدهای زیر را تایپ کنید :

```
Private Sub Drive1_Change()
 Dir1.Path = Drive1.Drive
 Label1.Caption = "Path = " & Drive1.Drive
End Sub
```

اکنون روی Dir1 دابل کلیک کرده کدهای زیر را تایپ کنید :

```
Private Sub Dir1_Change()
 File1.Path = Dir1.Path
 Label1.Caption = "Path = " & Dir1.Path
End Sub
```

در نهایت روی File1 دابل کلیک کرده و کدهای زیر را تایپ کنید :

```
Private Sub File1_Click()
 Label1.Caption = "Path = " & File1.Path & "\" & File1.FileName
End Sub
```

اکنون برنامه را اجرا کنید تا عملکرد برنامه را عملاً مشاهده کنید.

* * *

دستورهای فایل

ویژوال بیسیک دارای چند دستور برای کار با فایل ها، پوشه ها و درایوها است. این دستورات را در جدول زیر مشاهده می کنید :

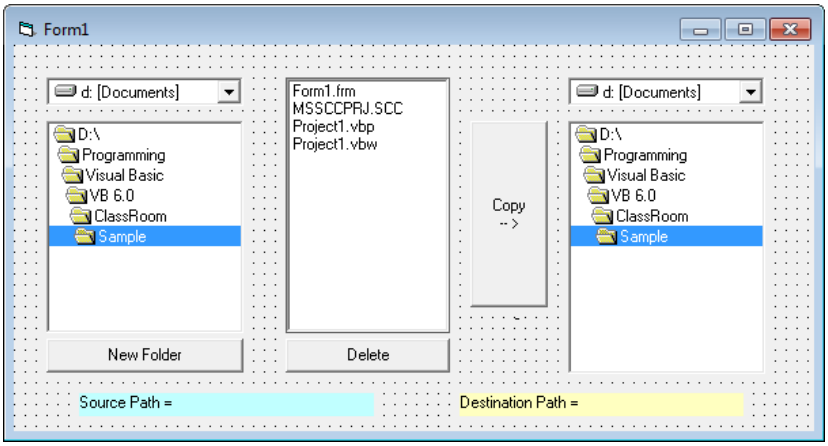
دستور	مفهوم
ChDrive strDrive	به درایور strDrive تغییر درایو می دهد.
ChDir strDirectory	به دایرکتوری (پوشه) strDirectory تغییر دایرکتوری می دهد.
Kill strFileSpec	فایل strFileSpec را پاک می کند.
MkDir strDirectory	دایرکتوری strDirectory را می سازد.
RmDir strDirectory	دایرکتوری strDirectory را حذف می کند.

تمرین :

۱- برنامه ای بنویسید که مجموعه ای از اعداد صحیح را از فایل ترتیبی بخواند و آنها را به صورت صعودی مرتب کرده و آنها را دوباره در همان فایل ذخیره کند.

۲- یک نوع دادهٔ جدید به نام Student تعریف کنید که شامل فیلدهای نام، نام خانوادگی و نمره باشد. سپس داده های ۵ رکورد را گرفته و در یک فایل تصادفی ذخیره کنید. حال با کلیک بر روی دکمهٔ دوم، نام و نام خانوادگی دانشجویی که بیشترین نمره را کسب کرده، نمایش دهید.

۳- برنامه ای که در آخرین مثال این فصل توضیح داده شد را به صورت زیر گسترش دهید.



فصل هشتم :

کار با پایگاه داده و ساخت گزارش

مفاهیم اولیه پایگاه داده

اغلب برنامه های کامپیوتری که امروزه نوشته می شوند به نحوی با داده و اطلاعات مختلف کار می کنند و بیشتر این برنامه ها، داده های مورد نیاز خود را در بانکهای اطلاعاتی رابطه ای نگهداری می کنند. بنابراین در هنگام نوشتن این نوع برنامه ها نیاز دارید که بتوانید در برنامهٔ خود با نرم افزارهای مربوط به این بانکهای اطلاعاتی، مانند SQL Server ، Access و... کار کنید.

زبان ساخت یافت پرس و جو (SQL) مورد استفاده سیستم های پایگاه داده رابطه ای است و از آن برای ایجاد پرس و جوها استفاده می شود. اکثر سیستم های پایگاه داده که مورد استفاده در ویژوال بیسیک هستند، از نوع پایگاه داده رابطه ای هستند. ویژوال بیسیک برنامه نویسان را قادر می سازد که کدهایی بنویسند تا با استفاده از پرس و جوها، به داده های موجود در سیستم های پایگاه داده رابطه ای دسترسی پیدا کنند.

اصولاً هر بانک اطلاعاتی شامل یک و یا چند فایل بزرگ و پیچیده است که داده ها در آن در یک قالب و فرمت ساخت یافته ذخیره می شوند. موتور بانک اطلاعاتی معمولاً به برنامه ای اطلاق می شود که این فایل ها و همچنین داده های درون آنها را مدیریت می کند. در طی این فصل از برنامه Microsoft Access 2003 به عنوان موتور بانک اطلاعاتی استفاده خواهیم کرد و استفاده ازSQL Server را در جلد دوم کتاب توضیح خواهیم داد.

یک فایل بانک اطلاعاتی مربوط به برنامهٔ Access (که پسوند آن نیز mdb. است) معمولا از قسمتهای مختلفی مانند جدولها، پرس‌وجوها، فرم ها، گزارشات، ماکرو ها و ماژول ها تشکیل شده است. به این قسمت های تشکیل دهندهٔ یک بانک اطلاعاتی «اشیاء بانک اطلاعاتی» گفته می شود.

یک جدول شامل مجموعه ای از اطلاعات است که حاوی یک یا چند ستون و نیز یک یا چند ردیف از داده ها است. به هر یک از این ستون ها «فیلد» و به هر ردیف از این اطلاعات یک «رکورد» گفته می شود. در هر جدول، یک رکورد شامل مجموعه ای از فیلدها است که اطلاعات و مشخصه های مربوط به یک نمونه از داده هایی که در آن جدول ذخیره شده است را نشان می دهد. برای مثال فیلدی به نام FName در یک جدول، مشخص کنندهٔ نام یک شخص است که اطلاعات او در آن جدول ذخیره شده است.

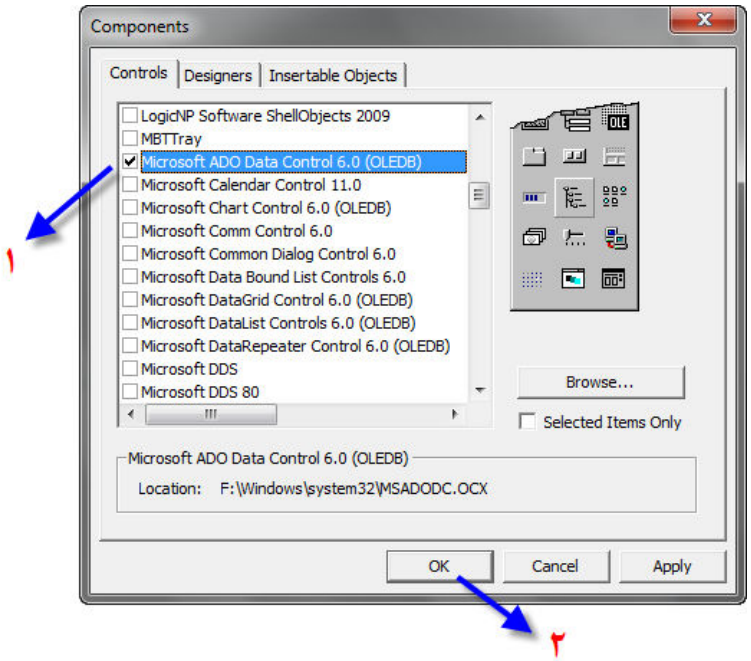
با استفاده از پرس و جوها (Query) می توانیم داده هایی را درون جدول های بانک اطلاعاتی اضافه، آنها را از جدول ها بازیابی و یا تغییراتی را در آنها ایجاد کنیم. این کار با استفاده از دستورات زبان SQL قابل انجام است. همانطور که گفته شد، دستورات این زبان و روش استفاده از آن در ویژوال بیسیک را در جلد دوم کتاب بطور کامل توضیح خواهیم داد.

در ویژوال بیسیک می توان این کارها را بدون استفاده از دستورات SQL انجام داد. برای این کار باید از کنترل Adodc استفاده کنید.

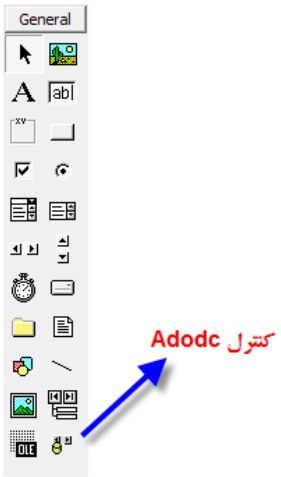
اضافه کردن و پیکربندی کنترل Adodc

ADO مخفف Activex Data Objects می باشد و همه امکانات OLEDB را از طریق object ها ارائه می دهد. OLEDB اتصال واقعی به یک منبع داده را ارایه می کند و همچنین امکان اتصال به منابعی غیر از SQL مانند Email و... را ارائه می دهد. بنابراین برای ارتباط بین برنامه ای که می نویسید و بانک اطلاعاتی Access، باید از ADO استفاده کنید.

کنترل Adodc در واقع نماینده گرافیکی برای استفاده از امکانات ADO است. برای استفاده از این کنترل ابتدا باید آن را به جعبه ابزار ویژوال بیسیک اضافه کنیم . برای این کار، ابتدا از منوی Project گزینهٔ Component را انتخاب کنید تا کادر مربوطه ظاهر شود. در این کادر گزینهٔ Microsoft ADO Data Control 6.0 (OLEDB) را یافته و انتخاب کنید. با کلیک روی دکمهٔ OK عملیات را تائید و کادر را ببندید.



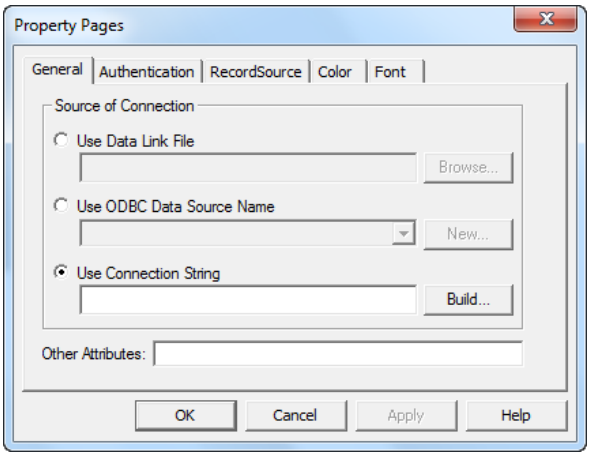
پس از انجام این کار، کنترل مربوطه به لیست ابزار ها اضافه خواهد شد :



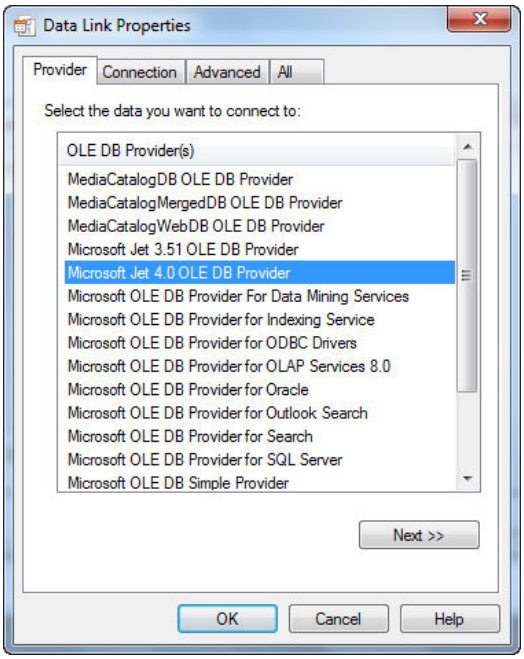
روی کنترل Adodc دابل کلیک کنید تا به فرم اضافه شود. بعد از اضافه شدن کنترل به فرم، باید آن را برای ارتباط با بانک اطلاعاتی پیکربندی کنید. این کار به دو روش قابل انجام است؛ روش اول استفاده از حالت انتخابی و طی چند مرحله بوده و روش دوم با استفاده از کدنویسی است که این روش بیشتر توصیه می شود. اما در هر دو روش مشخصه های StringConnection و RecordSource مربوط به کنترل Adodc باید مقداردهی شوند.

روش اول)

در پنجرهٔ Properties مربوط به کنترل Adodc، روی دکمهٔ موجود در مشخصه Custom کلیک کنید تا پنجرهٔ Property Page ظاهر شود.

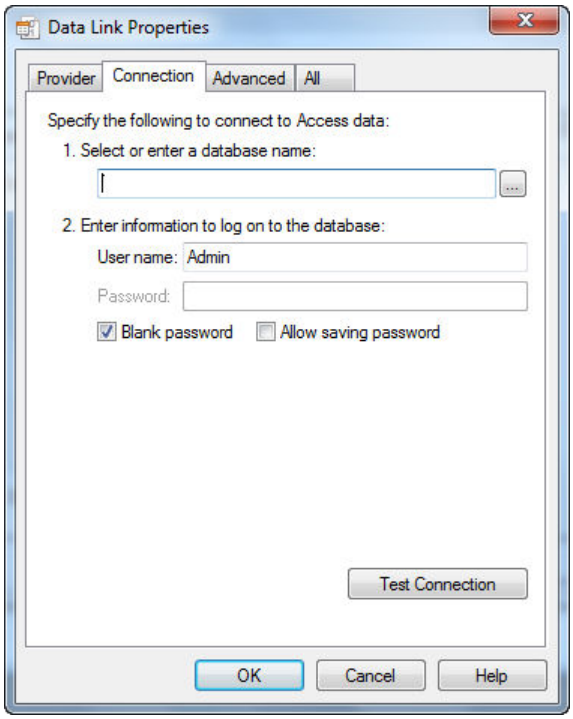


ابتدا در لبهٔ General و مقابل گزینهٔ Use Connection String روی دکمهٔ Build کلیک کنید تا به وسیلهٔ کادرهای محاوره ای ارتباط کنترل Adodc با بانک اطلاعاتی را برقرار کنیم. با کلیک بر روی این گزینه، کادر Data Link Properties نمایان خواهد شد :



در این کادر، لبهٔ Provider لیست Provider های مختلفی را برای اتصال به بانک اطلاعاتی فراهم می کند. این Provider ها کلاس های لازم برای اتصال به یک منبع داده، خواندن اطلاعات، ویرایش، بهنگام سازی و انجام عملیات متفاوت بر روی داده ها را ارائه می نماید. از آنجایی که ما قصد اتصال به یک بانک اطلاعاتی اکسس را داریم، باید گزینهٔ Microsoft Jet 4.0 OLE DB Provider را برای اتصال انتخاب کنیم.

سپس روی دکمهٔ Next کلیک کنید تا به لبهٔ Connection هدایت شوید :



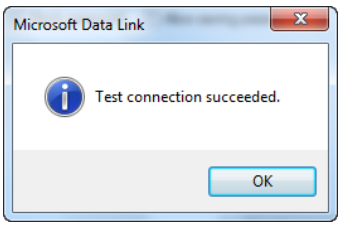
در این برگه (Tab)، ابتدا باید مسیر فایل بانک اطلاعاتی را مشخص کنید. برای اینکار دو راه وجود دارد :

۱. با استفاده از دکمهٔ موجو در جلوی گزینهٔ اول، فایل بانک اطلاعاتی را از مسیری که در آن ذخیره شده است، انتخاب کنید.

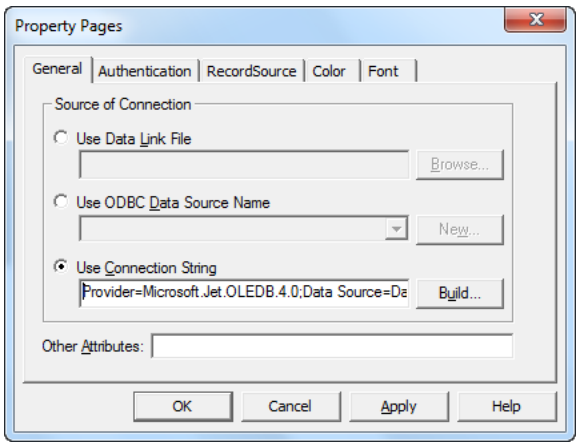
۲. فایل بانک اطلاعاتی را در مسیری که پروژهٔ ویژوال بیسیک را ذخیره کرده اید، کپی کرده و سپس در این قسمت فقط نام و پسوند بانک اطلاعاتی را وارد کنید. (مثلاً Database.mdb)

از آنجا که هر بانک اطلاعاتی اکسس دارای یک User Name پیش فرض به نام Admin است و پسوورد ندارد، از این گزینه صرف نظر می کنیم. البته در صورتی که از بانک اطلاعاتی اکسس که دارای پسورد است، استفاده می کنید، باید گزینهٔ Blank Pasword را غیرفعال کرده و در قسمت مربوطه Password مربوط به بانک را وارد کنید.

در نهایت روی دکمهٔ OK کلیک کنید تا مراحل انجام شده اعمال گردد. البته برای اطمینان از اتصال صحیح، می توانید روی دکمهٔ Test Connection کلیک کنید. پس از کلیک بر روی این دکمه در صورتی که با پیام زیر مواجه شدید، کنترل Adodc به درستی پیکربندی شده است.

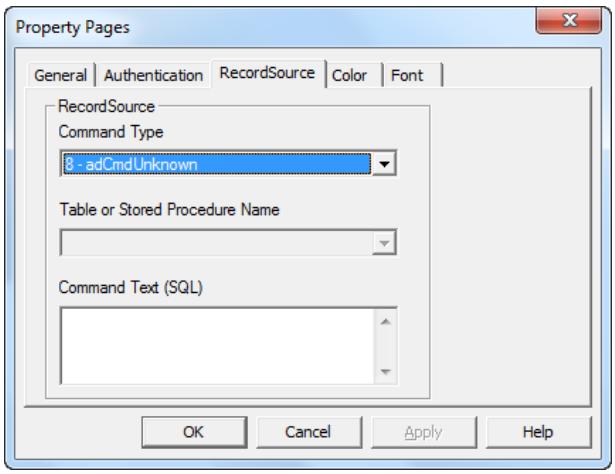


با تایید مراحل انجام شده، به کادر Property Page برمی گردیم که همانطور که مشاهده می کنید، رشتهٔ Connection String مقداردهی شده است.



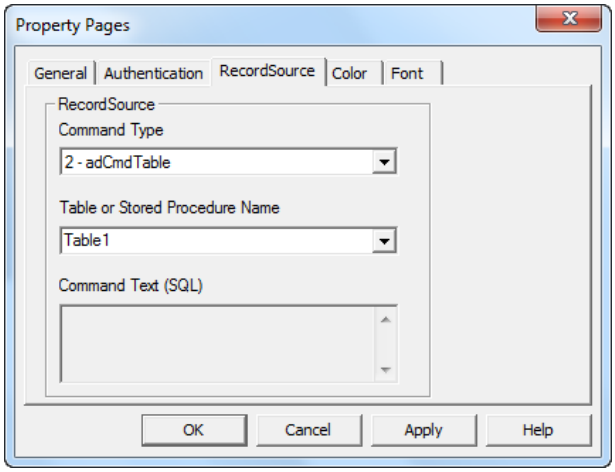
استفاده از این رشته در ارتباط با بانک اطلاعاتی به روش کدنویسی نیز کاملاً ضروری است.

پس از ساخت شدن Connection String باید مشخصهٔ Record Source را برای تعیین جدول و یا ایجاد پرس و جو، مقداردهی کنید. برای این کار در همین پنجره به لبهٔ RecordSource بروید.



در این قسمت، دو گزینه باید حتماً مقداردهی شوند. گزینهٔ اول Command Type است که با توجه به مقداری که به این گزینه می دهید، باید یکی از موارد بعدی یعنی Table or Stored Procedure Name و یا Command Text (SQL) را مقداردهی کنید. برای استفاده از گزینهٔ Command Text باید از دستورات SQL استفاده کنید. همانطور که در ابتدای این فصل بیان کردیم، این دستورات در جلد دوم کتاب بطور کامل توضیح داده خواهد شد. بنابراین در این فصل، از گزینهٔ اول یعنی نام جدول استفاده می کنیم.

برای این کار، ابتدا مقدار Command Type را برابر با 2-adCmdTable قرار می دهیم و پس از فعال شدن گزینهٔ دوم، از بین نام جداول موجود در بانک اطلاعاتی، جدول مورد نظر را انتخاب و سپس روی دکمهٔ OK می کنیم. برای مثال در شکل زیر جدول Table1 انتخاب شده است.



پس از انجام مراحل ذکر شده، کنترل Adodc به بانک اطلاعاتی و جدول مورد نظر، متصل شده و آمادهٔ استفاده خواهد بود.

روش دوم)

این روش بهترین روش برای طراحی برنامه های پایگاه داده می باشد. مهم ترین علت آن، این است که در این روش راحتتر می توان خطاهای احتمالی برنامه را که در هنگام اتصال به بانک اطلاعاتی ممکن است رخ دهد، پیدا کرد و برای کنترل یا رفع آن اقدام نمود.

تمام مراحل توضیح داده شده در روش اول را با استفاده از دستورات زیر می توان پیاده سازی کرد :

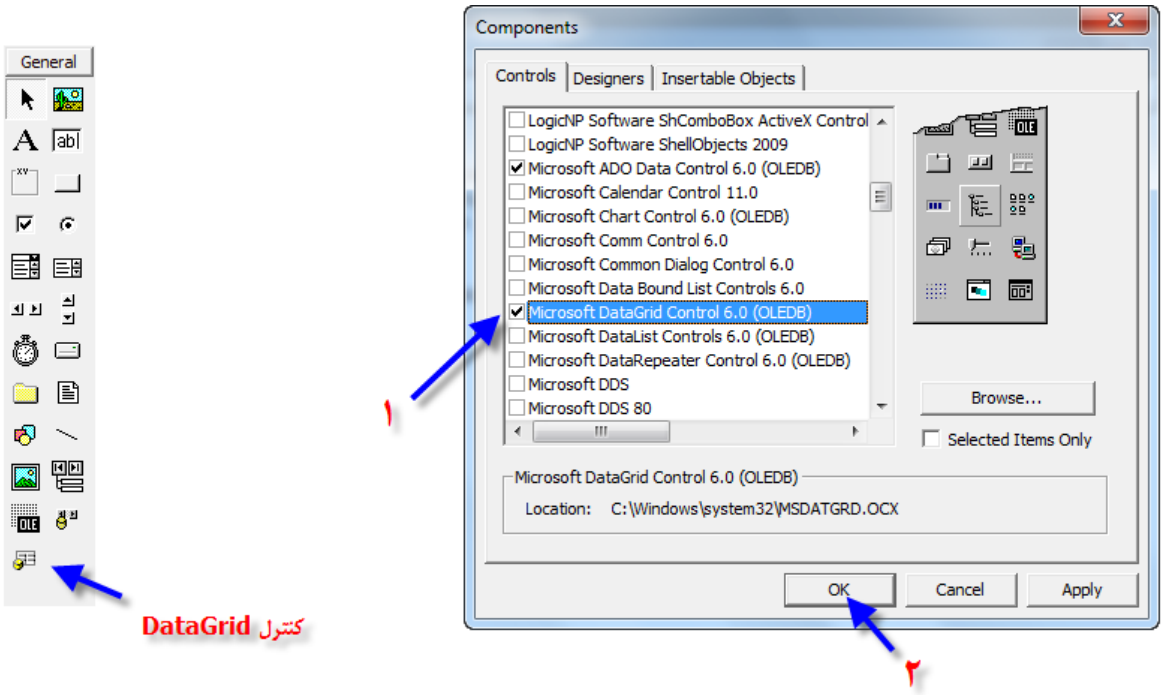
```
Dim DBPath As String
DBPath = App.path & "\Database.mdb"
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DBPath
Adodc1.CommandType = adCmdTable
Adodc1.RecordSource = "Table1"
Adodc1.Refresh
```

توجه کنید که در قطعه کد فوق، دستور App.Path مسیری که برنامه از آنجا اجرا مـی شـود را برمـی گردانـد و سـپس نـام فایـل بانـک اطلاعاتی را به انتهای آن اضافه کرده و درون متغیـر DBPath قـرار خواهـد داد. بعـد از مقـداردهی مشخـصه هـای کنتـرل Adodc ، آن را Refresh می کند تا ارتباط برقرار شود.

کنترل DataGrid و ارتباط آن با Adodc

یکی از کنترل های مهمی که می تواند برای نمایش جدول های بانک اطلاعاتی به کار آید، کنترل DataGrid اسـت. اگـر جـدول بانـک اطلاعاتی را به این کنترل مقید کنید، اطلاعات آن جدول به صورت یک جدول در آن کنترل نمایش داده می شود.

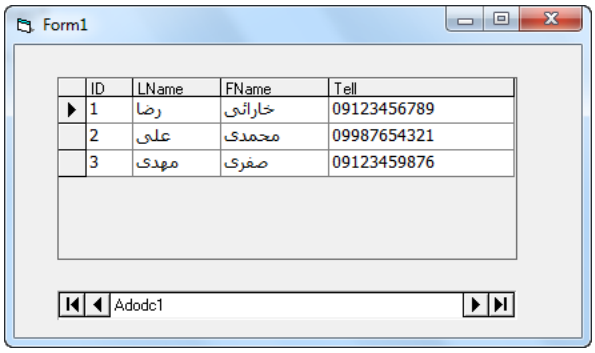
برای استفاده از این کنترل، ابتدا باید آن را به لیست ابزارهـای ویـژوال بیـسیک اضـافه کنیـد. بـرای ایـن کـار از منـوی Project گزینـهٔ Component را انتخاب کرده و در پنجره باز شده گزینهٔ Microsoft DataGrid Control 6.0 را انتخاب و روی دکمهٔ OK کلیک کنید.



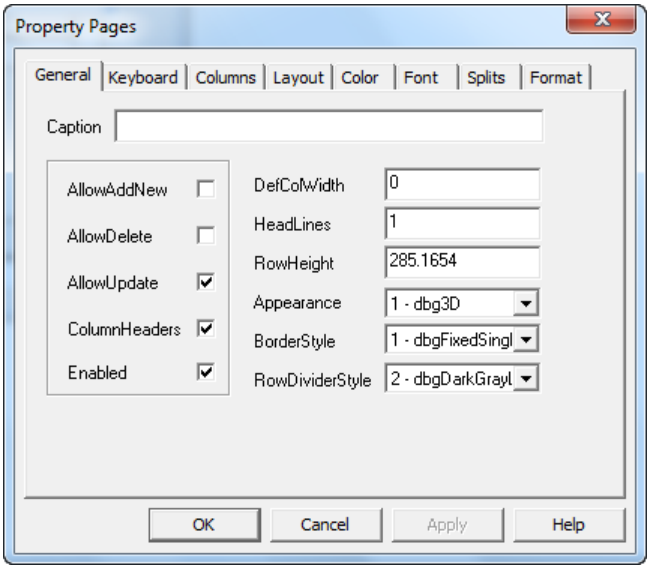
برای برقراری ارتباط بین این کنترل و جدول بانک اطلاعاتی، باید کنترل DataGrid را به کنترل Adodc متصل کرد. این کار نیـز ماننـد پیکربندی کنترل Adodc با استفاده از دو روش امکان پذیر است. در روش کدنویسی فقط کافی است که دسـتور زیـر را بـه انتهـای قطعـه برنامه قبلی (بعد از Adodc1.Refresh) اضافه کنید :

```
Set DataGrid1.DataSource = Adodc1
```

با بکارگیری این دستور، ویژوال بیسیک کنترل DataGrid به طور خودکار پیکربندی می کند تا داد های منبع تعیین شده را نمایش دهد.



به دلیل اینکه تنظیمات پیش فرض همیشه بهترین نیستند، DataGrid به طور کامل قابل پیکربندی بوده و امکان نمایش ستون های خاص، قالب بندی و رنگ هر ستون را فراهم می کند. برای این کار، روی DataGrid کلیک راست کرده و ابتدا گزنهٔ Edit را انتخاب کنید. سپس مجدداً روی آن کلیک راست و با استفاده از گزینهٔ Insert و Delete جدول را به تعداد ستون های مور نظرتان تنظیم کنید. پس از تعیین تعداد ستون های DataGrid با کلیک راست و انتخاب گزینهٔ Properties به پنجرهٔ Property Page بروید.



در این پنجره، می توانید تنظیمات پیشرفته مربوط به کنترل DataGrid را تنظیم کنید. این تنظیمات در برگه های مختلف، تقسیم بندی شده اند که به شرح زیر می باشد :

برگهٔ General : در این برگه می توانید موارد کلی DataGrid را تنظیم کنید که هر کدام در جدول زیر مشخص شده اند :

نام مشخصه	توضیح
Caption	عنوان کلی این جدول را مشخص می کند.
Allow Add New	در صورتی که این متد True باشد در انتهای جدول یک رکورد خالی ایجاد می کند که در صورتی که در آن داده ای وارد نمایید آن داده به لیست اضافه خواهد شد اما اگر این متد False باشد چنین کاری انجام نخواهد شد.
Allow Delete	در صورتی که این متد را فعال کنید می توانید با انتخاب یک رکورد و زدن بر روی دکمه Delete آن رکورد را حذف کنید.
Allow Update	اگر این متد را فعال کنید با اجرای برنامه، تغییراتی را که در داده های DataGrid توسط کاربر اعمال می شود، ثبت می کند.
Column Heardrs	تعیین می کند که ستون های جدول (که همان فیلدها هستند) عنوان داشته باشند.
Enabled	در صورتی که True باشد DataGrid فعال خواهد بود.
RowHeight	ارتفاع سطرها را تعیین می کنید.
Appearance	3D یا Flat بودن DataGrid را مشخص می کند.
BorderStyle	تعیین می کند که حاشیه DataGrid محدود باشد یا خیر.
RowDividerStyle	نوع خط های بین سطر ها را تعیین می کند.
DataMode	تعیین می کند که جدول مقید به داده های کنترل باشد یا نه.

برگهٔ KeyBoard : در این قسمت کلید های فعال روی جدول را می توانید مشخص کنید.

برگهٔ Columns : در این قسمت می توانید تنظیمات مربوط به هر ستون را تعیین کنید که این تنظیمات شامل موارد زیر می باشد :

نام مشخصه	توضیح
Column	ستون جدول را مشخص می کند.
Caption	عنوان هر ستون را مشخص می کند.
DataField	در این قسمت فیلد مربوط به هر ستون را می توانید تعیین کنید.

برگهٔ Layout : در این قسمت می توانید برای هر ستون به طور مجزا خصوصیاتی را اختصاص دهید.

نام مشخصه	توضیح
Locked	در صورتی که این گزینه را برای هر ستون True کنید اطلاعات آن ستون غیر قابل تغییر خواهد بود.
AllowSizing	تعیین می کند که اندازه هر ستون در حال اجرای برنامه قابل تغییر باشد یا نه.
Visible	در صورتی که این گزینه را برای هر ستون False کنید اطلاعات آن ستون نمایش داده نخواهد شد.
Button	در صورتی که این گزینه را برای هر ستون True کنید در کنار آن ستون یک دکمه قرار خواهد گرفت.
DividerStyle	تعیین کننده نوع خط بین هر ستون می باشد.
Alignment	مکان قرار گرفتن متن داخل هر ستون را مشخص می کند.
Width	پهنای هر ستون را مشخص می کند.

برگهٔ Color : در این برگه می توانید رنگ قسمت های مختلف DataGrid را مشخص کنید.

برگهٔ Font : در این برگه می توانید نوع قلم ستون ها و داده های داخل جدول را تغییر دهید.

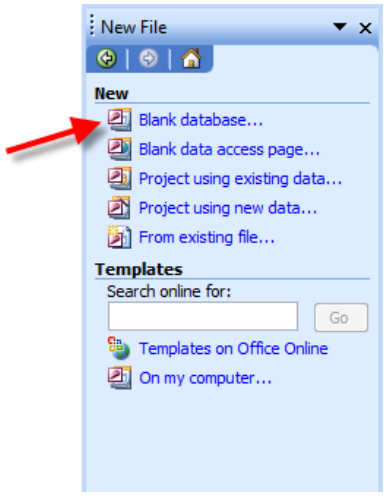
در ادامه مثال کاملی از تنظیمات DataGrid بیان خواهیم کرد.

طراحی یک نمونه از یک برنامهٔ کاربردی با پایگاه داده Access

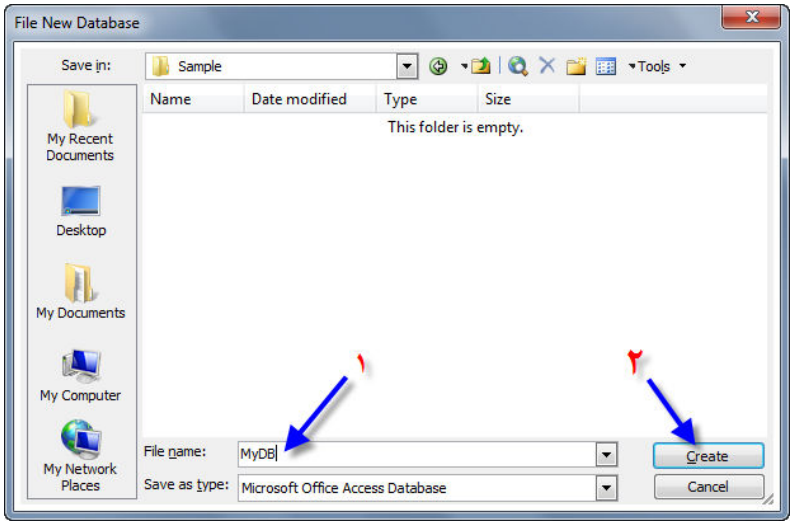
تا اینجا با کنترل ها و نحوهٔ برقراری ارتباط برنامه با بانک اطلاعاتی اکسس آشنا شدید. اما همانطور که می دانید یک برنامهٔ کاربردی، باید امکان اضافه، ویرایش، حذف و جستجو را داشته باشد. برای اینکه کاملاً با روش طراحی و ساخت چنین برنامه هایی آشنا شوید، قصد داریم با توضیح یک مثال از تمام مراحل طراحی و ساخت یک برنامه ساده و کاربردی، انجامِ این کار را آموزش دهیم. بنابراین، مراحل زیر را با دقت انجام دهید :

طراحی بانک اطلاعاتی اکسس

قبل از انجام هرکاری باید یک دیتابیس اکسس طراحی کنید. برای این کار برنامه Microoft Access 2003 را اجرا کرده و از منـوی File گزینهٔ New را انتخاب کنید. در کادری که سمت راست نمایان می شود، گزینهٔ Blank database... را انتخاب کنید.

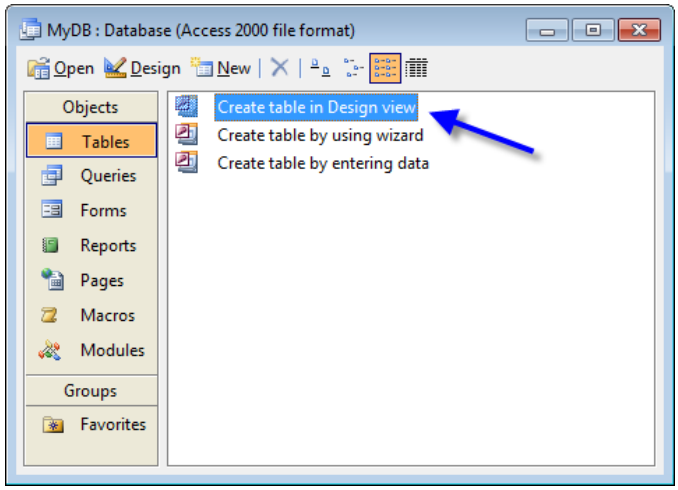


با انتخاب این گزینه، پنجرهٔ File New Database نمایان می شود که باید مسیری مورد نظر را برای ذخیره فایل بانک اطلاعاتی (در ایـن مثال پوشه ای که پروژهٔ ویژوال بیسیک در آن ذخیره شده) انتخاب کنید و سپس در قسمت File Name نام فایل بانـک اطلاعـاتی (در ایـن مثال : MyDB) را تایپ کرده و نهایتاً روی دکمهٔ Create کلیک کنید.

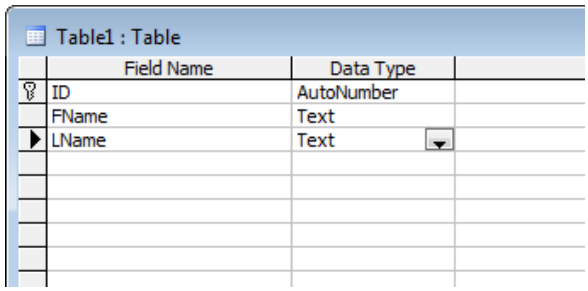


پس از انجام مراحل فوق، فایل بانک اطلاعاتی در مسیر مورد نظر ایجاد می شود. اکنون باید جدول مورد نیاز برنامه را طراحی کنیم.

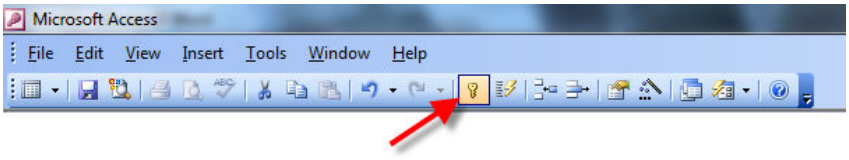
برای طراحی جدول، روی گزینهٔ Create table in Design view دابل کلیک کنید :



پنجره ای باز می شود که می توانید فیلدها و نوع داده های آنها را تعیین کنید. در این پنجره فیلدهای زیر را ایجاد کنید :



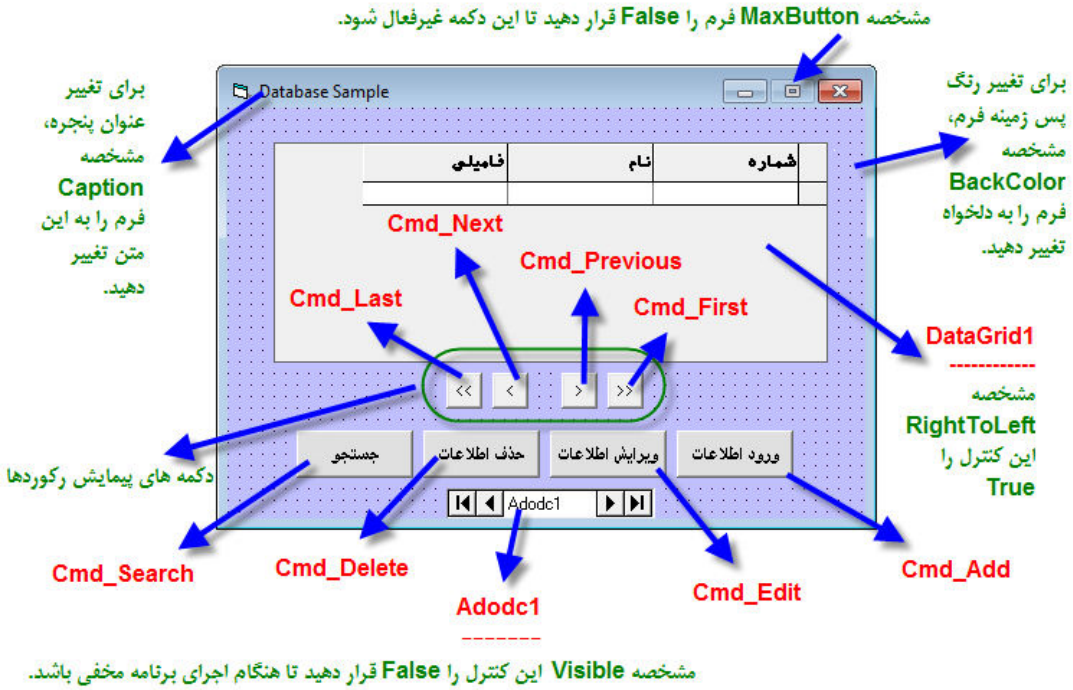
توجه داشته باشید که برای انتخاب فیلد ID به عنوان کلید اصلی جدول، باید ابتدا این فیلد را انتخاب کرده و سپس از منوی ابزارها، آیکن کلید را انتخاب کنید.



اکنون کار طراحی بانک تمام شده است. جدول مورد نظر را با نام Table1 ذخیره کرده و برنامه اکسس را ببندید.

طراحی محیط برنامه

پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :

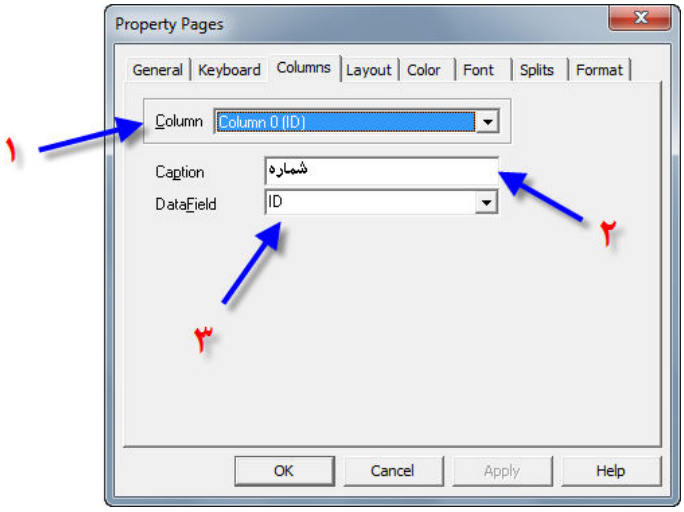


توجه داشته باشید که برای تغییر نام دکمه ها باید مشخصهٔ Name آنها را به موارد فوق، تغییر دهید و برای پیکربندی DataGrid (تعیین تعداد ستونها، تغییر عنوان ستونها، تنظیم ارتفاع و تغییر فونت ستونها و همچنین ارتباط هر ستون به یک فیلد از جدول مورد نظر) باید مراحل زیر را انجام دهید :

روی DataGrid1 کلیک راست کرده و گزینهٔ Edit را انتخاب کنید تا به حالت ویرایش تبدیل شود. مجدداً کلیک راست کنید و روی Insert کلیک کنید تا یک ستون دیگر به جدول اضافه شود.

اکنون برای سایر تنظیمات، روی DataGrid1 کلیک راست کنید و گزینهٔ Properties را انتخاب کنید تا پنجرهٔ Property Page ظاهر شود. در این پنجره ابتدا در برگهٔ General گزینهٔ HeadLines را برابر با ۲ قرار دهید. سپس به برگهٔ Columns رفته و در لیست Column ستون اول را انتخاب کنید. سپس در قسمت Caption عنوان مناسب و در قسمت DataField نام فیلد مورد نظر را وارد کنید. این کار را برای دو ستون دیگر نیز انجام دهید.

شکل زیر، تنظیمات ستون اول را نشان می دهد.



پس از انجام همهٔ این مراحل، کار طراحی فرم اصلی تمام شده و باید کدنویسی های لازم را شروع کنید. دقت داشته باشید که همه مراحل ذکر شده باید به درستی انجام شوند تا در ادامهٔ کار به مشکلی برنخورید.

ارتباط کنترل ها با بانک اطلاعاتی

اکنون، قبل از هر چیز باید ارتباط بین کنترل Adodc1 و بانک اطلاعاتی و همچنین ارتباط کنترل DataGrid1 را با کنترل Adodc1 را برقرار کنیم. در این مثال ما از روش کدنویسی که روش بهتری نسبت به روش ویزاردی است، استفاده می کنیم. بدین منظور روی فرم دابل کلیک کنید و دستورات زیر را تایپ نمایید :

```
Private Sub Form_Load()
Dim DBPath As String
DBPath = App.path & "\MyDB.mdb"
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DBPath
Adodc1.CommandType = adCmdTable
Adodc1.RecordSource = "Table1"
Adodc1.Refresh
Set DataGrid1.DataSource = Adodc1
End Sub
```

رویداد Form_Load یکی از رویدادهای مهم و پرکاربرد فرم می باشد. این رویداد زمانی که فرم در حال بارگذاری شدن است اجرا می شود. درنتیجه، زمانی که این فرم ظاهر می شود ارتباط بین بانک اطلاعاتی و کنترل های Adodc1 و DataGrid1 برقرار خواهد شد.

اضافه کردن رکورد به جدول

برای افزودن یک رکورد به جدول باید تمام فیلدهای آن را مقداردهی کرد. اما با توجه به اینکه در این مثال فیلد ID از نوع AutoNumber در نظر گرفته شده و به طور خودکار مقداردهی می شود، نیاز به مقداردهی این فیلد نداریم. بنابراین، دستورات زیر را برای دکمهٔ Cmd_Add بنویسید :

```
Private Sub Cmd_Add_Click()
 Dim FirstName, LastName As String
 FirstName = InputBox("Enter First Name")
 LastName = InputBox("Enter Last Name")
 Adodc1.Recordset.AddNew
 Adodc1.Recordset.Fields("FName") = FirstName
 Adodc1.Recordset.Fields("LName") = LastName
 Adodc1.Recordset.Update
End Sub
```

همانطور که ملاحظه می کنید، در قطعه برنامهٔ فوق، ابتدا دو متغیر رشته ای تعریف و توسط کاربر مقداردهی می شوند. سپس با استفاده از متد AddNew از RecordSet یک رکورد بطور موقت به جدول اضافه می شود و با اجرای دو خط بعدی، هر فیلد با مقدار متغیرهای مربوطه مقداردهی شده و در نهایت با استفاده از متد Update، رکورد جدید به جدول افزوده خواهد شد.

توجه داشته باشید که RecordSet رابط اصلی بین برنامه و جدول مورد نظر در بانک اطلاعاتی است و تمام امکانات مورد نیاز برای کار با یک جدول بانک اطلاعاتی مانند : افزودن، ویرایش، حذف، جستجو و... را برای برنامه نویس فراهم می کند.

ویرایش اطلاعات یک رکورد

یکی دیگر از امکانات هر برنامه کاربردی که از پایگاه داده استفاده می کند، این است که برای کاربر امکان ویرایش اطلاعاتی که از قبل وارد کرده و ممکن است دچار تغییراتی شده باشد، فراهم کند. برای این کار ابتدا باید رکورد مورد نظر انتخاب و سپس تغییرات جدید روی آن اعمال گردد. در این مثال، برای انتخاب رکورد کافی است که کاربر روی رکورد مورد نظر در DataGrid کلیک کند؛ با این کار اشاره گر روی آن رکورد قرار می گیرد و دستورات به‌روزرسانی روی آن اعمال خواهد شد.

روی دکمهٔ ویرایش (Cmd_Edit) دابل کلیک کنید و دستورات زیر را تایپ نمایید :

```
Private Sub Cmd_Edit_Click()
 Dim FirstName, LastName As String
 FirstName = InputBox("Enter First Name")
 LastName = InputBox("Enter Last Name")
 Adodc1.Recordset.Update "FName" , FirstName
 Adodc1.Recordset.Update "LName" , LastName
End Sub
```

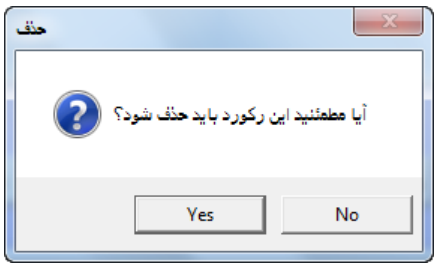
همانطور که ملاحظه می کنید دستور ویرایش رکوردها بسیار ساده است. زیرا متد Update از RecordSet دو پارامتر را دریافت کرده که پارامتر اول نام فیلد مربوطه و پارامتر دوم مقدار جدید فیلد می باشد.

حذف رکورد از جدول

برای این کار نیز ابتدا باید رکورد مورد نظر انتخاب شود. همان گونه که توضیح دادیم، این کار با کلیک بر روی رکورد مورد نظر در کنترل DataGrid توسط کاربر انجام می شود. برای حذف یک رکورد، روی دکمهٔ Cmd_Delete دابل کلیک کرده و کدهای زیر را تایپ نمائید :

```
Private Sub Cmd_Delete_Click()
 Dim Result AS Integer
 Result = MsgBox "آیا مطمئنید این رکورد باید حذف شود؟", vbYesNo+vbQuestion, "حذف"
 If Result = 6 Then  Adodc1.Recordset.Delete
End Sub
```

با اجرای دستورات فوق، ابتدا یک پیغام مانند شکل زیر ظاهر می شود :



با پاسخ دادن به این پیام توسط کاربر، مقداری که در متغیر Result قرار می گیرد، یا ۷ (دکمهٔ No) و یا ۶ (دکمهٔ Yes) خواهد بود. این مقادیر و مقدار سایر دکمه های قابل استفاده در تابع MsgBox، در فصل سوم همین کتاب، توضیح داده شده است. در صورتی که مقدار متغیر Result برابر با ۶ باشد، با متد Delete از RecordSet رکورد انتخاب شده، حذف می شود.

جستجو بین رکوردها

عمل جستجو، یکی از مهمترین و پرکاربردترین قسمتهای یک برنامهٔ کاربردی که دارای پایگاه داده است، محسوب می شود. برنامه ای را تصور کنید که دارای چند صد یا چند هزار رکورد است و قصد داریم اطلاعات یک رکورد خاص را در آن بیابیم؛ کاملاً غیرمنطقی است اگر بخواهیم تمام رکوردها را بررسی کرده تا به رکورد مورد نظر برسیم. عمل جستجو به ما این امکان را می دهد تا اطلاعات جدول را با مقادیری از یک یا چند فیلد خاص (که قصد جستجو کردن بر اساس آنها را داریم) فیلتر کنیم. برای اینکار باید نام فیلد و مقدار مورد نظر را در متد Filter از RecordSet بکار ببریم.

در این مثال قصد داریم جدول Table1 را بر اساس فیلد ID و مقداری که کاربر وارد می کند، فیلتر کنیم. برای اینکار روی دکمهٔ Cmd_Search دابل کلیک کرده و کدهای زیر را تایپ کنید :

```
Private Sub Command4_Click()
 Dim Key As String
 Key = InputBox("Enter ID For Search")
 Adodc1.Recordset.Filter = "ID='" & Key & "'"
End Sub
```

توجه داشته باشید که مقداری که می خواهید فیلد را با آن فیلتر کنید، باید درون تک کوتیشن قرار داده شود. همچنین اگر می خواهید جدول را براساس دو یا چند فیلد، فیلتر کنید باید بین آنها از کاما استفاده کنید. به عنوان مثال، کد زیر جدول را بر اساس دو فیلد نام و نام خانوادگی فیلتر می کند :

```
Private Sub Command4_Click()
 Dim FNameKey, LNameKey As String
 FNameKey = InputBox("Enter First Name For Search")
 LNameKey = InputBox("Enter Last Name For Search")
 Adodc1.Recordset.Filter = "FName='" & FNameKey & "' , FName='" & LNameKey & "'"
End Sub
```

مقداردهی متد Filter، چیزِ خیلی سختی نیست! فرض کنید کاربر مقدار نام را «رضا» و نام خانوادگی را «خارائی» وارد می کند، شما با در نظر گرفتن این موضوع باید کاری کنید که مقدار این متد برابر با متن زیر شود :

```
FName='رضا' , FName='خارائی'
```

با درک این موضوع، نوشتن و درک قطعه کد قبلی چندان مشکل نیست.

پیمایش بین رکوردها

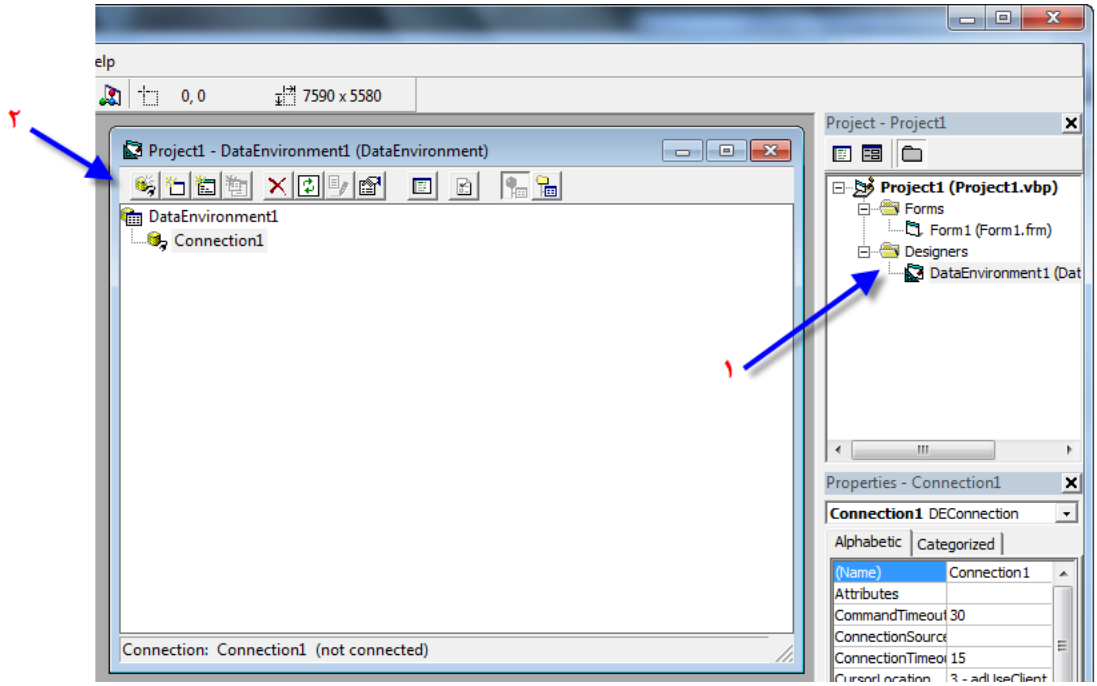
برای حرکت بین رکورد های جدول، به ترتیب روی دکمهٔ های Cmd_First، Cmd_Previous، Cmd_Next و Cmd_Last دابل کلیک کنید و کدهای زیر را وارد نمائید :

```
Private Sub Cmd_First_Click()
 Adodc1.Recordset.MoveFirst
End Sub

Private Sub Cmd_Previous_Click()
 Adodc1.Recordset.MovePrevious
End Sub

Private Sub Cmd_Next_Click()
 Adodc1.Recordset.MoveNext
End Sub

Private Sub Cmd_Last_Click()
 Adodc1.Recordset.MoveLast
End Sub
```

گزارش گیری با استفاده از Data Report

گزارشگیری می تواند یکی از ویژگی هایی باشد که یک برنامه کاربردی را کاملتر و قابل استفاده تر می کند و کاربران به استفاده از این نوع برنامه ها رغبت بیشتری دارند. یعنی به طور کلی امکان گزارشگیری در یک برنامه از مهمترین ویژگی های آن است. از گزارشگیری بیشتر در چاپ استفاده می شود و کاربر می تواند قبل از چاپ، اطلاعاتی را مشاهده و ارزیابی کند. در این کتاب، روش ساخت گزارش معمولی با استفاده از امکانات آمادهٔ ویژوال بیسیک را توضیح خواهیم داد، ولی ابزارهای پیشرفته تری مانند Crystal Repor که توانایی ساخت گزارشهای جالبتر و بهتری را دارا می باشند، وجود دارند که در جلد دوم این کتاب آن را توضیح خواهیم داد.

برای ایجاد یک گزارش معمولی، ابتدا از منوی Project گزینه Add Data Environment را انتخاب کنید. با زدن این گزینه فایلی به پروژه افزوده و پنجره ای به صورت زیر ایجاد می شود :



همانطور که در تصویر فوق می بینید این پنجره از یک لیست درختی تشکیل شده که سرشاخهٔ آن، همان نام رابط (Data Environment) است. شاخهٔ Connections لیست ارتباط ها با پایگاه داده ها را مشخص می کند. برای مثال Connection1 با یک پایگاه داده مرتبط است و حال این که Connection بعدی که ایجاد می کنید می تواند با یک پایگاه داده دیگر ارتباط برقرار کند. برای پیکربندی یک پایگاه داده مراحل زیر را دنبال کنید :

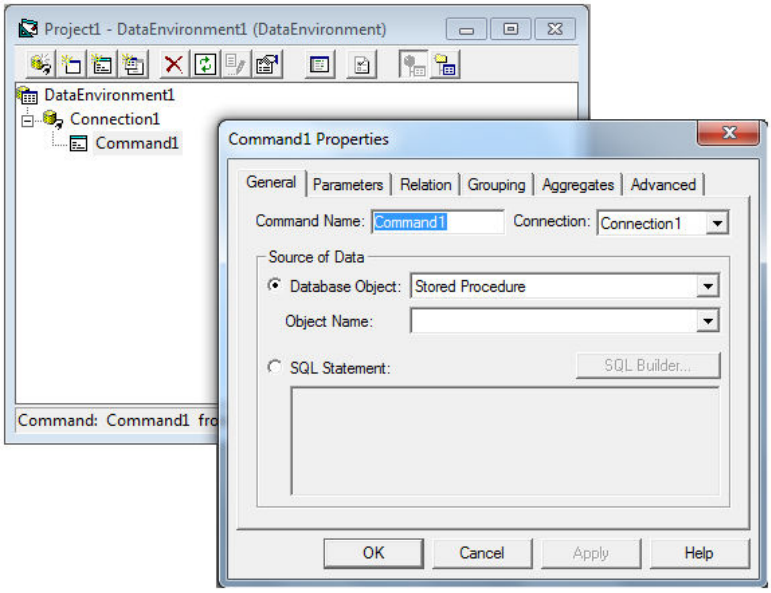
۱) روی Connection1 کلیک راست کرده و گزینهٔ Properties را انتخاب کنید تا کادر Data Link Properties نمایش داده شود.

۲) همانطور که در ابتدای این فصل توضیح دادیم، در این کادر باید ابتدا گزینهٔ Microsoft Jet 4.0 OLE DB Provider را انتخاب کرده و سپس روی دکمهٔ Next کلیک کنید تا برگهٔ دوم این کادر نمایش داده شود.

۳) در برگهٔ دوم این کادر، برای قسمت اول، نام و پسوند فایل بانک اطلاعاتی (در این مثال : MyDB.mdb) را تایپ کنید. برای آگاهی از ارتباط صحیح Connection با بانک اطلاعاتی، دکمهٔ Test Connection را بزنید. در نهایت، روی دکمهٔ OK برای پایان این مرحله کلیک کنید.

با انجام این مراحل، ارتباط Connection1 با پایگاه داده برقرار شده است. حال زمان استفاده از Command است. از Command برای انتخاب جدول و یا ایجاد پرس و جو (Query) استفاده می شوند. برای ایجاد یک Command برای Connection1 روی آن کلیک راست کرده و گزینهٔ Add Command را بزنید. با این کار، Command1 به لیست Command ها اضافه خواهد شد. برای پیکربندی Command1 مراحل یر را انجام دهید :

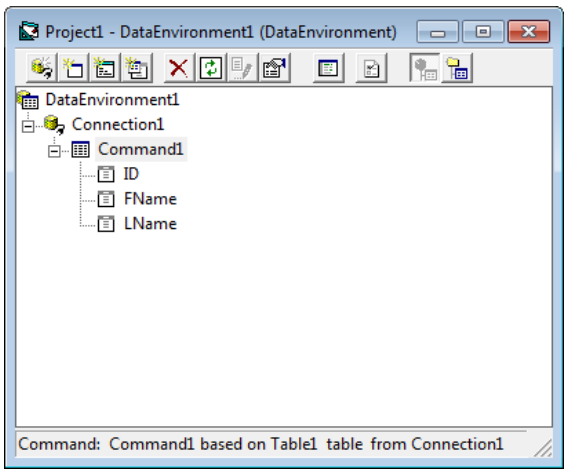
۱) روی Command1 کلیک راست کرده و گزینه Properties را انتخاب کنید تا کادر Command Properties نمایش داده شود :



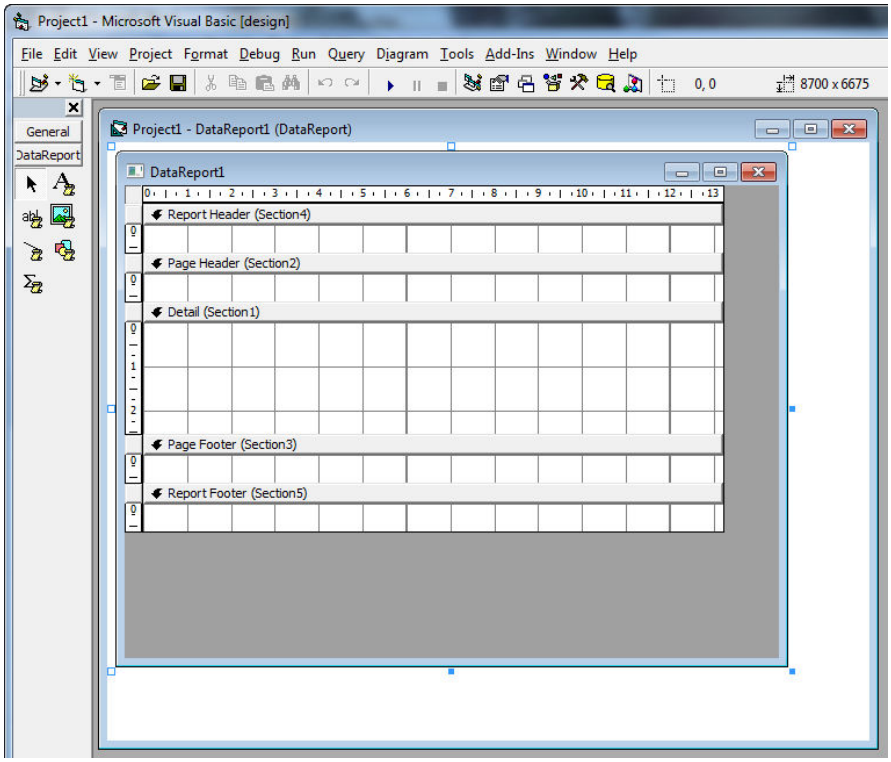
۲) در برگهٔGeneral ، از لیست DataBase Objects گزینهٔ Table را انتخاب کنید تا در لیست Object Name نام جدول های موجود در پایگاه داده ظاهر شود؛ سپس جدول مورد نظر (در این مثال : Table1) را انتخاب کنید.

۳) روی دکمهٔ OK کلیک کنید تا عملیات تایید شده و پنجرهٔ Command1 Properties بسته شود.

پس از انجام مراحل فوق، فیلدهای جدول به شاخهٔ Command اضافه خواهند شد :



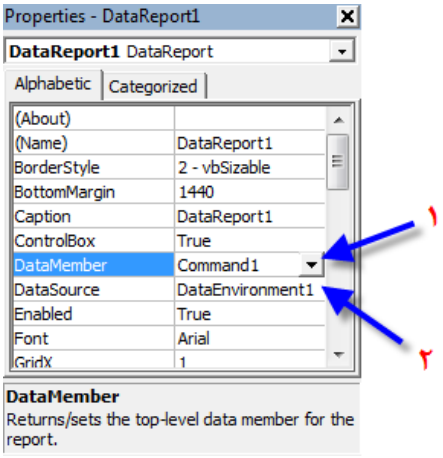
اکنون تنظیمات مربوط به رابط پایگاه داده به پایان رسید. برای ایجاد صفحهٔ طراحی گزارش، از منوی Project گزینهٔ Add Data Report را انتخاب کنید. با این کار، یک پنجرهٔ طراحی گزارش با جعبهٔ ابزار مخصوص، به نام DataReport1 در اختیار شما قرار داده می شود.



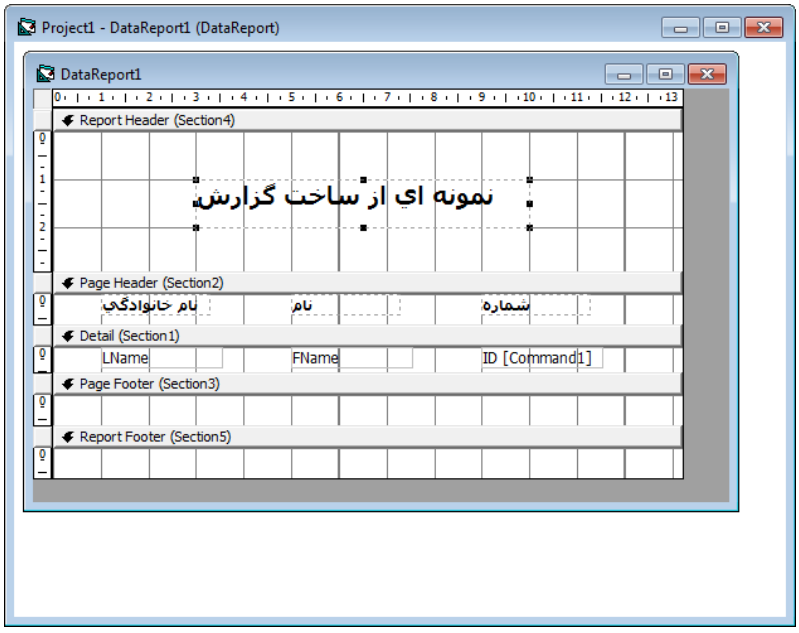
در این پنجره، صفحهٔ گزارش به بخش های مختلفی از جمله Report Header، Page Header، Detail، Page Footer و Report Footer تقسیم شده است. قسمت های Report Header و Report Footer به ترتیب در ابتدا (صفحهٔ اول) و انتهای گزارش (صفحه آخر) درج می شوند و قسمت های Page Header و Page Footer به ترتیب در بالا و پائین هر صفحه از گزارش تکرار می شوند. قسمت Detail نیز محتوای گزارش خواهد بود. همچنین کنترل های قابل استفاده در این پنجره نیز شامل کنترل های زیر می باشند :

کنترل	شرح
RptTextBox	هر داده ای که در زکان اجرا ارایه خواهد شد، نیاز دارد تا در یک RptTextBox قرار داده شود.
RptLine	از این کنترل برای ترسیم خطوط مختلف در روی گزارش استفاده می شود.
RptFunction	این کنترل، توابعی را روی گروه های داده در گزارش اجرا می کند و فق طمی تواند در بخش Page Footer مورد استفاده قرار گیرد. توابعی مثل Row، Count و ... .
RptLabel	از این کنترل برای اضافه کردن متن ایستا به گزارش استفاده می شود.
RptImage	از این کنترل برای اضافه کردن تصویر به گزارش استفاده می شود. در این کنترل فایلهای تصویری با پسوندهای bmp، Gif، Jpg و آیکون ها قابل نمایش هستند.
RptShape	از این کنترل می توان برای اضافه کردن شکل های گرافیکی مختلف به گزارش استفاده کرد که شامل شکل های مستطیل، ایره و... می باشد.

قبل از هر کاری در پنجرهٔ Data Report باید مشخصهٔ DataSource را برابر با نام DataEnvironment و مشخصهٔ Data Member را برابر با نام Command که در پنجرهٔ Data Environment ساخته اید، قراردهید.



حال برای اضافه کردن فیلدهایی که قصد دارید در گزارش درج شوند، هر فیلد را از پنجرهٔ Data Environment گرفته و به بخش Detail گزارش درگ کنید. ویژوال بیسیک بطور خودکار یک کنترل RptTextBox را همراه با RptLabel روی گزارش ترسیم می کند. سپس با قرار دادن یک RptLabel در بخش Report Header و تنظیم مشخصه های Caption و Font گزارشی مانند شکل زیر طراحی کنید :



پس از انجام این کار، طراحی گزارش به پایان رسیده و مرحلهٔ پایانی، دستور نمایش گزارش خواهد بود. برای اینکار به فرم اصلی پروژه (Form1) بازگردید و یک دکمه با نام Cmd_Report روی فرم قرار دهید؛ سپس روی آن دابل کلیک کرده و کد زیر را تایپ نمائید :

```
Private Sub Cmd_Report_Click()
 DataReport1.Show
End Sub
```

برنامه را اجرا کنید و پس از افزودن چند رکورد به وسیلهٔ دکمهٔ «ورود اطلاعات»، روی دکمهٔ cmd_Report کلیک کنید تا پنجرهٔ پیش نمایش گزارش ظاهر شود :

برای قرار دادن شمارهٔ صفحه و تعداد کل صفحات (مثل صفحه ۳ از ۵) در صفحات گزارش، یک کنترل RptLabel در قسمت Page Footer اضافه کرده و می توانید از ثابت هایی که ویژوال بیسیک برای این کار در نظر گرفته است، استفاده کنید. لیست ثابت هایی که در ویژوال بیسیک برای طراحی صفحات گزارش وجود دارد به شرح زیر است :

ثابت	شرح
%p	شماره صفحهٔ جاری
%P	تعداد کل صفحات گزارش
%d	تاریخ جاری (میلادی در قالب کوتاه)
%D	تاریخ جاری (میلادی در قالب بلند)
%t	زمان جاری در قالب کوتاه
%T	زمان جاری در قالب بلند
%i	عنوان گزارش

تمرین :

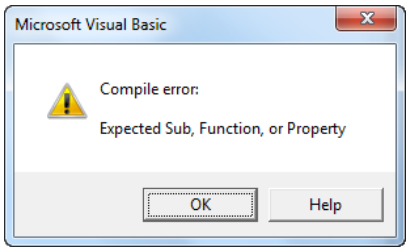
۱- یک برنامه دفترچه تلفن با تمام امکانات (افزودن، ویرایش، حذف، جستجو و چاپ) طراحی کنید.

فصل نهم :

مباحث پایانی

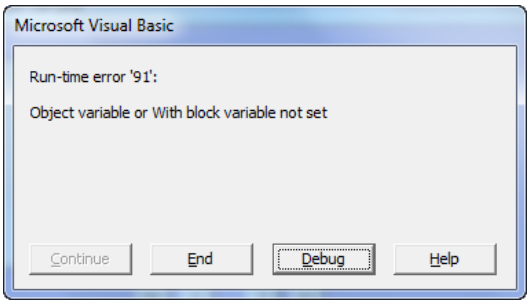
کنترل خطا در برنامه

همزمان با کدنویسی، IDE ویژوال بیسیک خطاهای نحوی (Syntax Error) را که به وجود می آیند، اعلام می کند. این نوع خطاها شامل اشتباه در املاء یا محل قرارگیری کلید واژه ها هستند. برای مثال اگر از دستور If استفاده کرده و آن را با دستور End If نبندید، این خطا به صورت زیر اعلام می شود :



اما گاهی پیش می آید که یک دستور در برنامه ای که می نویسید، در زمان کدنویسی و کامپایل هیچ خطایی نداشته باشد ولی در حین اجرای برنامه ممکن است خطایی تولید کند.

برای مثال فرض کنید که هنگام اضافه کردن یک رکورد در جدول بانک اطلاعاتی، ارتباط برنامه با پایگاه داده قطع شود یا کاربر برای مقدار کلید اصلی، یک مقدار تکراری وارد کند و یا هر خطایی که ممکن است زمان اجرا در این موقع رخ دهد؛ این خطاها معمولاً به صورت زیر اعلام می شود :



برای جلوگیری از بروز چنین خطاهایی بهتر است از دستور On Error Goto استفاده کنید. با استفاده از این روش، اگر زمان اجرای یک دستور در هنگام اجرای برنامه پیغام خطایی رخ دهد، می توانیم برنامه را به خط خاصی از پنجرهٔ کدنویسی هدایت کنیم. شکل کلی استفاده از این دستور به صورت زیر است :

```
On Error Goto برچسب
 .دستوری که ممکن است با اجرای آن خطا رخ دهد
 :
 :
برچسب :
 .دستوری که باید در صورت بروز خطا اجرا شود
```

نام برچسب می تواند یک رشته یا عدد باشد، البته در صورت عدد بودن مقادیر منفی را نمی پذیرد.

به قطعه کد زیر که بعنوان یک مثال کاربردی برای کار با پایگاه داده در فصل قبل بکار بردیم، توجه کنید :

```
Private Sub Cmd_Add_Click()
 Dim FirstName, LastName As String
 FirstName = InputBox("Enter First Name")
 LastName = InputBox("Enter Last Name")

 On Error Goto Err_Control
 Adodc1.Recordset.AddNew
 Adodc1.Recordset.Fields("FName") = FirstName
 Adodc1.Recordset.Fields("LName") = LastName
 Adodc1.Recordset.Update
 MsgBox "اطلاعات با موفقیت ثبت گردید."
 Exit Sub

Err_Control:
 MsgBox "خطایی هنگام ذخیره اطلاعات بوجود آمده است."

End Sub
```

با اجرای این دستورات، اگر زمان اضافه کردن یک رکورد موقت در جدول بانک اطلاعاتی خطایی رخ دهد، برنامه به خط Err_Control هدایت می شود و پیغام مناسب که به صورت فارسی درنظر گرفته شده، ظاهر می گردد. ضمناً با استفاده از این روش، برنامه پایان نمی یابد.

توجه داشته باشید که اگر خطایی رخ ندهد، دیگر نیاز نیست که خط Err_Control اجرا شود. بنابراین باید با استفاده از دستور Exit Sub از روال خارج شویم.

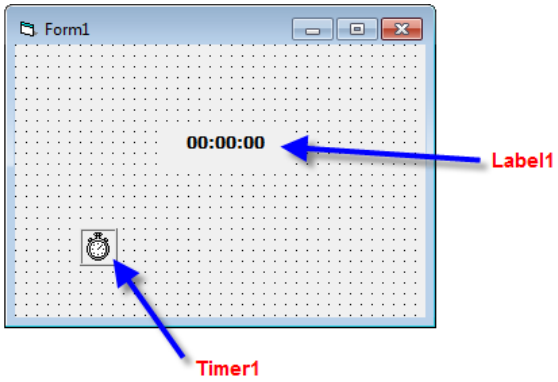
کنترل Timer

اگر می خواهید دستور یا دستوراتی را در فاصلهٔ زمانی مشخص، اجرا کنید، باید از کنترل Timer استفاده کنید. این کنترل در زمان اجرا قابل رؤیت نخواهد بود و فقط در فاصلهٔ زمانی که در مشخصهٔ Interval آن مشخص می کنید، دستوراتی را که در رویداد Timer() آن می نویسید، اجرا خواهد کرد. واحد اندازه گیری مشخصهٔ Interval میلی ثانیه است و بیش از ۶۵۵۳۵ ، مقداری قبول نمی کند. به عبارت دیگر حداکثر فاصلهٔ زمانی که می توانید در این کنترل تنظیم کنید، حدود ۶۵ ثانیه است.

مثال :

فرمی طراحی کنید که با اجرای برنامه، زمان (ساعت) را درون یک Label نشان دهد. بطوری که پیوسته در حال افزایش و نمایش درون Label باشد.

برای نوشتن این برنامه، ابتدا پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



ابتدا مشخصهٔ Interval مربوط به کنترل Timer1 را برابر با ۱۰۰۰ قرار دهید، زیرا هر هزار میلی ثانیه برابر با یک ثانیه است. سپس مشخصهٔ Alignment مربوط به Label1 را 2-Center قرار دهید تا متن آن وسط چین شود. همچنین Caption آن را با 00:00:00 مقداردهی کنید. Font مربوط به این کنترل را نیز می توانید به دلخواه تغییر دهید.

حال روی Timer دابل کلیک کرده و کد زیر را تایپ کنید :

```
Private Sub Timer1_Timer()
 Label1.Caption = CStr(Time)
End Sub
```

برنامه را اجرا کنید. همانطور که ملاحظه می کنید، درون Label1 ساعت به صورت ثانیه به ثانیه نمایش داده می شود. در واقع، تابع Cstr مقدار زمان را با استفاده از تابع Time دریافت کرده و آن را به String تبدیل می کند و درون Caption مربوط به Label1 قرار می دهد. این روال هر هزار میلی ثانیه (یک ثانیه) اجرا می شود.

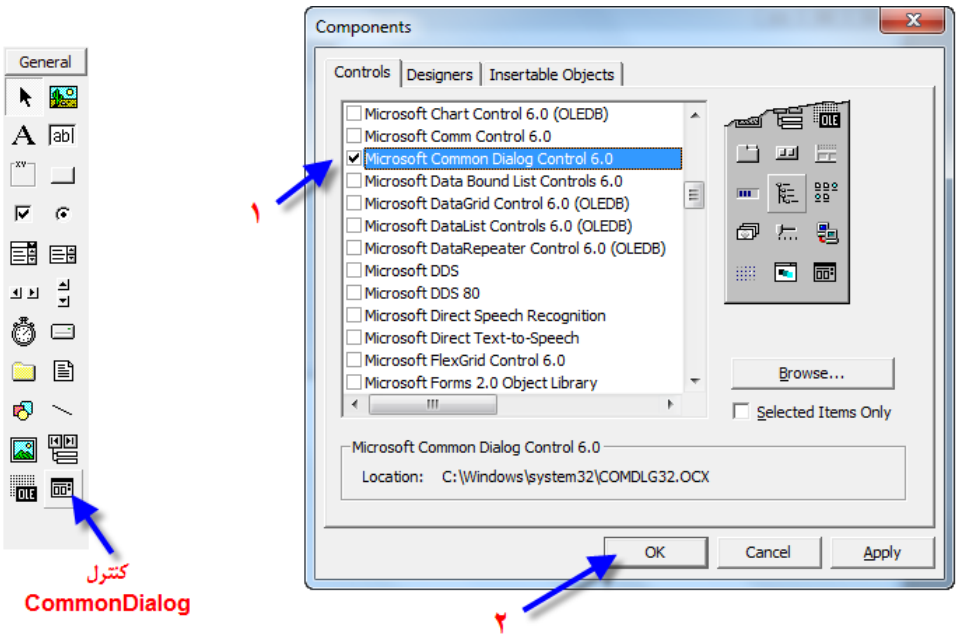
همچنین برای نمایش تاریخ (بصورت میلادی) می توانید از دستور CStr(Date) استفاده کنید.

کنترل CommonDialog

بعضی مواقع ممکن است بخواهید برنامه ای بنویسید که کاربران بتوانند مسیر یک فایل را تعیین کنند، نوع و یا رنگ قلم یک کنترل را به مورد دلخواه خود تغییر دهند و یا چاپگر را کنترل کنند. ویژوال بیسیک کادرهای محاوره ای آماده ای را توسط کنترل Common Dialog برای برنامه نویسان ارائه می دهد. این کنترل نیز مانند کنترل Timer زمان اجرا قابل رؤیت نیست. با استفاده از این کنترل می توانید به چهار کادر محاوره ای ویندوز دسترسی داشته باشید :

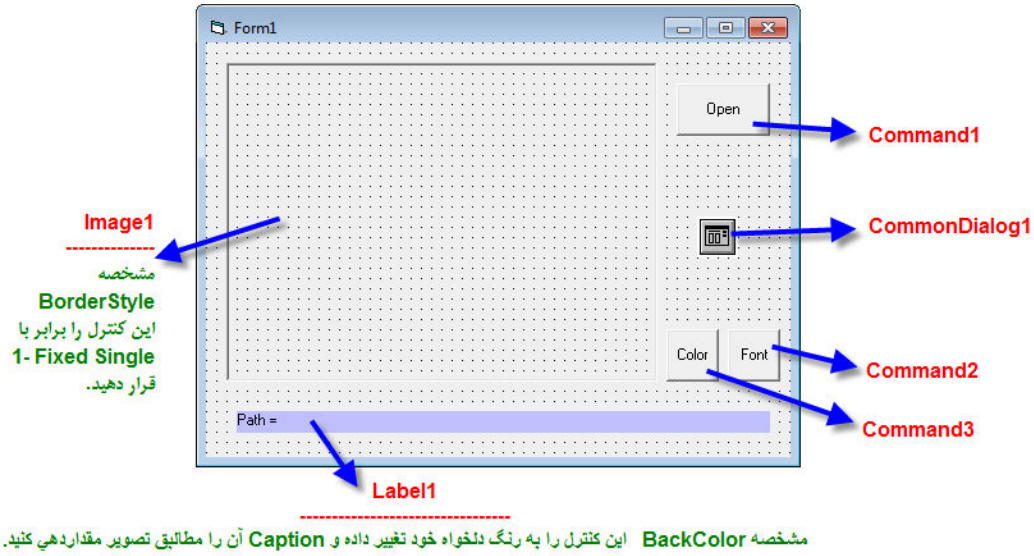
- **File (Open و Save) :** به کاربر امکان انتخاب فایل برای باز شدن یا انتخاب نام فایل برای ذخیرهٔ اطلاعات را فراهم می کند.
- **Font :** امکان انتخاب قلم و تنظیم مشخصه های قلم مورد نظر را فراهم می کند.
- **Color :** امکان انتخاب رنگ یا ایجاد رنگ سفارشی برای استفاده در برنامه را ارایه می کند.
- **Print :** امکان انتخاب چاپگر و تنظیم چندین پارامتر چاپگر را فراهم می کند.

برای استفاده از این کنترل، ابتدا باید آن را به پروژه اضافه کنید. برای این کار از منوی Project گزینهٔ Components را انتخاب کرده و در پنجرهٔ باز شده، گزینهٔ Microsoft Common Dialog Control 6.0 را انتخاب و روی دکمهٔ Ok کلیک کنید تا این ابزار به لیست ابزارهای پروژه اضافه شود.



مثال :

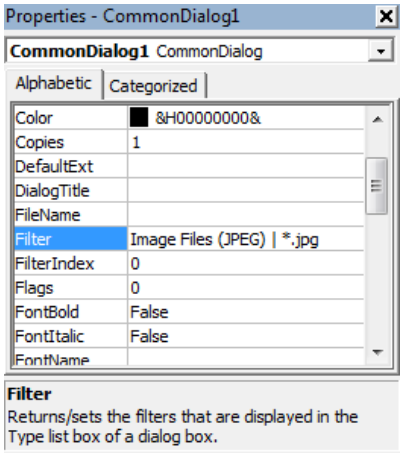
پروژهٔ جدیدی را در ویژوال بیسیک ایجاد کرده و فرمی مانند شکل زیر طراحی کنید :



قصد داریم که کاربر با کلیک بر روی دکمهٔ Open یک فایل تصویری با پسوند jpg را توسط کادر محاوره ای انتخاب کند و برنامه تصویر مربوطه را در کنترل Image1 نمایش داده و مسیر فایل را در Label1 نمایش دهد.

برای این کار پس از تنظیم مشخصهٔ BorderStyle مربوط به کنترل Image، مقدار مشخصهٔ Stretch این کنترل را نیز برابر با True کنید. مشخصهٔ BorderStyle برای تنظیم حاشیه کنترل بوده و مشخصهٔ Stretch برای آن است که تصویر انتخاب شده در ابعاد کنترل Image1 نمایش داده شود (درغیر اینصورت پس از انتخاب تصویر، اندازهٔ کنترل Image1 به اندازهٔ تصویر مورد نظر، تغییر پیدا خواهد کرد).

برای آن که در کادر محاوره ای، فقط فایل های تصویری قابل رؤیت و انتخاب باشند، کنترل CommonDialog1 را انتخاب کرده و مشخصهٔ Filter را مانند شکل زیر مقداردهی کنید :

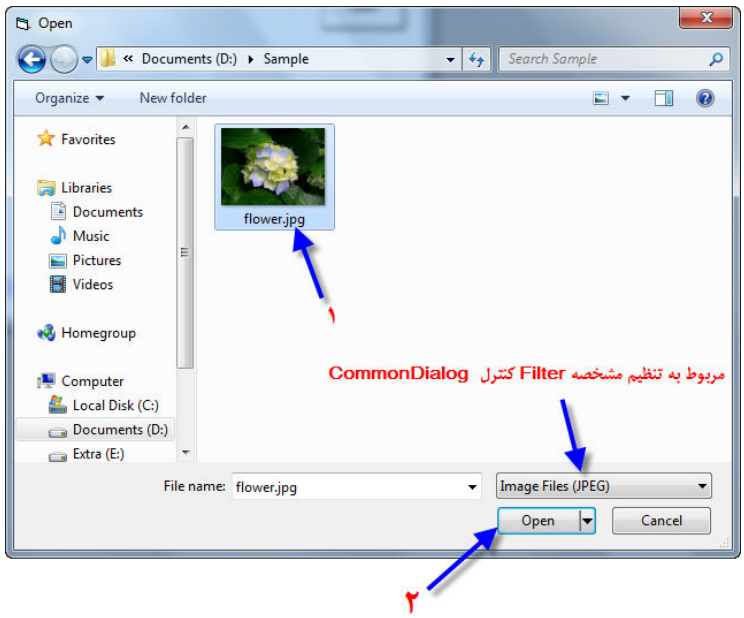


اکنون باید برای نمایش کادر محاوره ای و سپس بدست آوردن مسیر فایل انتخاب شده و نهایتاً نمایش تصویر آن در کنترل Image1 برنامه بنویسید. برای این کار روی دکمهٔ Command1 دابل کلیک کرده و کدهای زیر را تایپ کنید :

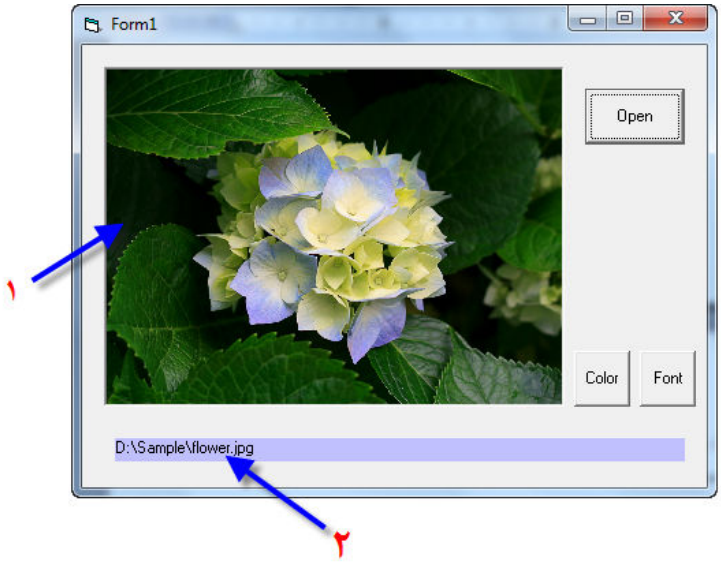
```
Private Sub Command1_Click()
 CommonDialog1.ShowOpen
 If CommonDialog1.FileName <> "" Then
  Image1.Picture = LoadPicture(CommonDialog1.FileName)
  Label1.Caption = CommonDialog1.FileName
 End If
End Sub
```

با اجرای دستورات فوق، ابتدا کادر Open مربوط به کنترل Common Dialog نمایان می شود. در صورتی که کاربر دکمهٔ Cancel را در این کادر انتخاب کند، مقدار مشخصهٔ FileName این کنترل یک رشتهٔ خالی خواهد بود. اما اگر فایلی را انتخاب کرده و دکمهٔ Open را بزند، مقدار این مشخصه برابر با FileName مربوط به آن فایل خواهد شد. بنابراین، با استفاده از دستور شرطی، این عمل را بررسی کرده و در صورتی که فایلی انتخاب کرده باشد، ابتدا آن را در کنترل Iamge1 نمایش داده و سپس مسیر فایل را در Label1 نشان می دهیم. برای نمایش یک تصویر باید مشخصهٔ Picture مربوط به کنترل Image را برابر با فایل مربوطه قرار داد که این عمل در کدنویسی با استفاده از تابع LoadPicture() انجام می شود.

برنامه را تا همین جا یک بار اجرا کنید، همانطور که می بینید با کلیک بر روی دکمهٔ Open کادر محاوره ایِ Open نمایان می شود و به کاربر فقط امکان انتخاب فایل هایی با پسوند jpg را می دهد.



یک فایل تصویری را انتخاب کنید تا ببینید برنامه چگونه عمل می کند :



اکنون قصد داریم با استفاده از دکمه های Font و Color، همان مشخصه های مربوط به Label1 را تنظیم کنیم. برای این کار ابتدا کنترل Common Dialog را انتخاب کرده و سپس مقدار مشخصهٔ Flags را برابر با ۱ قرار دهید. این مشخصه به کنترل بیان می کند که قلم های صفحه نمایش، قلم های چاپگر و یا هر دو را نمایش دهد. این مشخصه را می توانید با یکی از مقادیر زیر مقداردهی کنید :

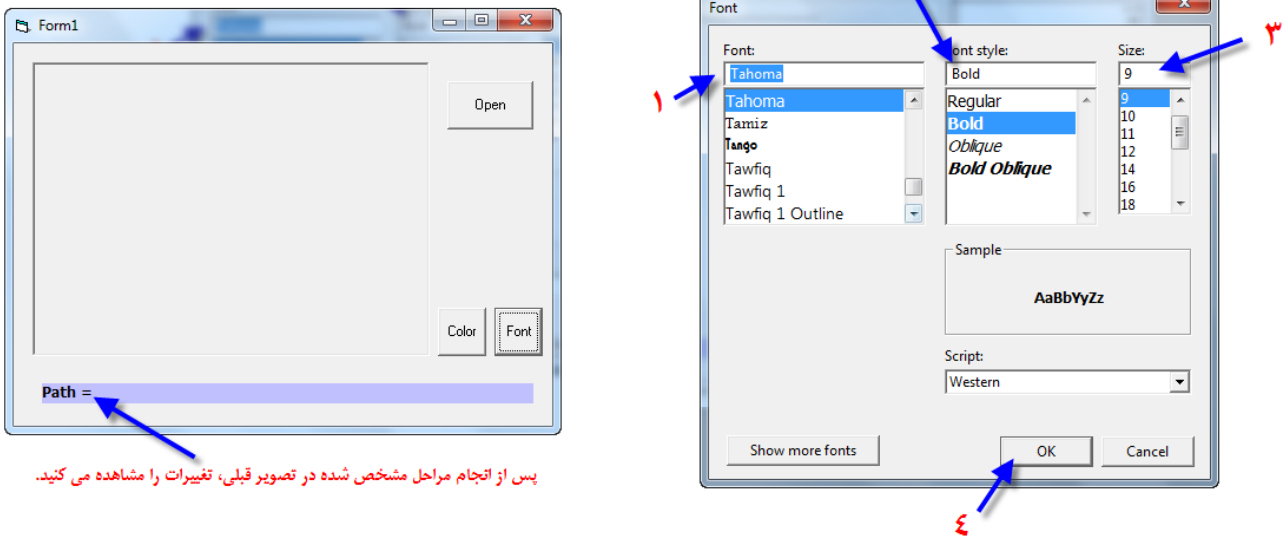
مجموعهٔ قلم	ثابت	مقدار
قلم های صفحهٔ نمایش	cdlCFScreenFonts	۱
قلم های چاپگر	cdlCFPrinterFonts	۲
هر دو مجموعه	cdCFBoth	۳

اکنون روی Command2 دابل کلیک کرده و کدهای زیر را وارد نمائید :

```
Private Sub Command2_Click()
 CommonDialog1.ShowFont
 On Error Resume Next
 Label1.FontName = CommonDialog1.FontName
 Label1.FontSize = CommonDialog1.FontSize
 Label1.FontBold = CommonDialog1.FontBold
 Label1.FontItalic = CommonDialog1.FontItalic
End Sub
```

توجه داشته باشید که خط دوم برای کنترل خطا بکار برده شده است. این دستور باعث می شود که اگر کاربر هیچ نوع قلمی را انتخاب نکرد و درواقع دکمهٔ Cancel را در کادر محاوره ای Font انتخاب کرد، از اجرای دستورات بعدی صرف نظر شود.

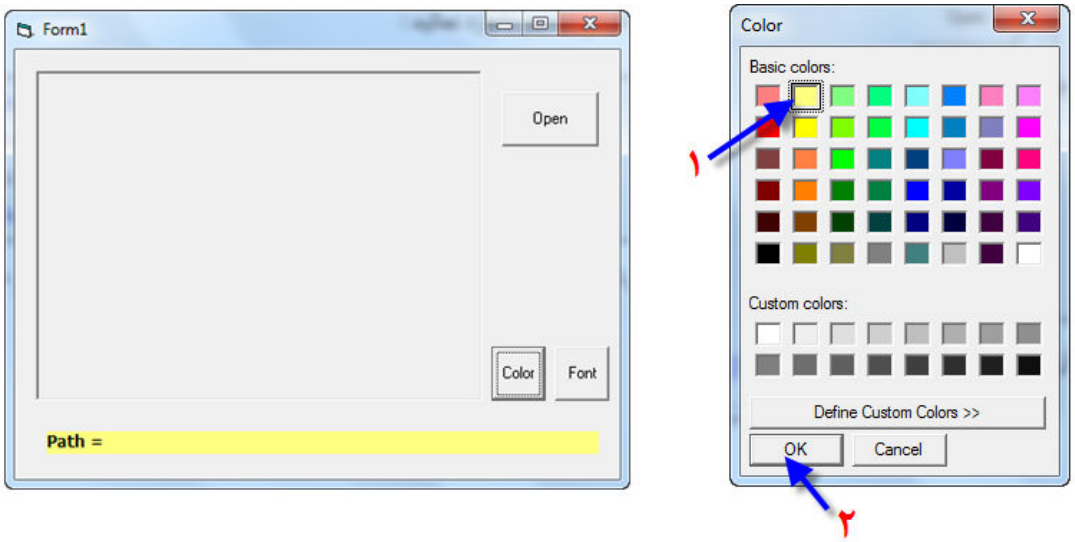
برنامه را مجدداً اجرا و تست کنید تا طرز عملکرد برنامه را مشاهده کنید :



حال روی دکمهٔ Command3 دابل کلیک کنید و دستورات زیر را وارد نمائید :

```
Private Sub Command3_Click()
 CommonDialog1.ShowColor
 Label1.BackColor = CommonDialog1.Color
End Sub
```

با اجرای این دستورات، کادر انتخاب رنگها نمایان شده و در صورت انتخاب رنگ مورد نظر، رنگ پس زمینهٔ Label1 نیز تغییر خواهد کرد.

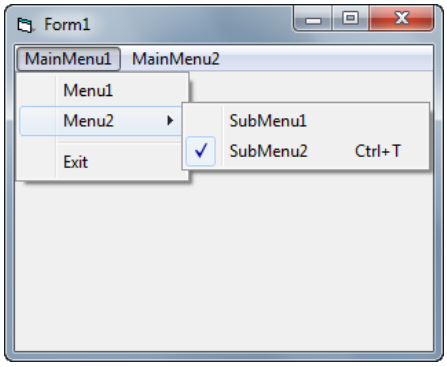


طراحی منوها

داشتن منوهای مناسب و سپردن حداقل کار به کاربر، از جمله مسائلی است که به طراحی یک سیستم که استفاده از آن آسان باشد، کمک می کند. در این بخش، روش های طراحی انواع منوها را در ویژوال بیسیک شرح خواهیم داد. منوها به دو نوع تقسیم می شوند :

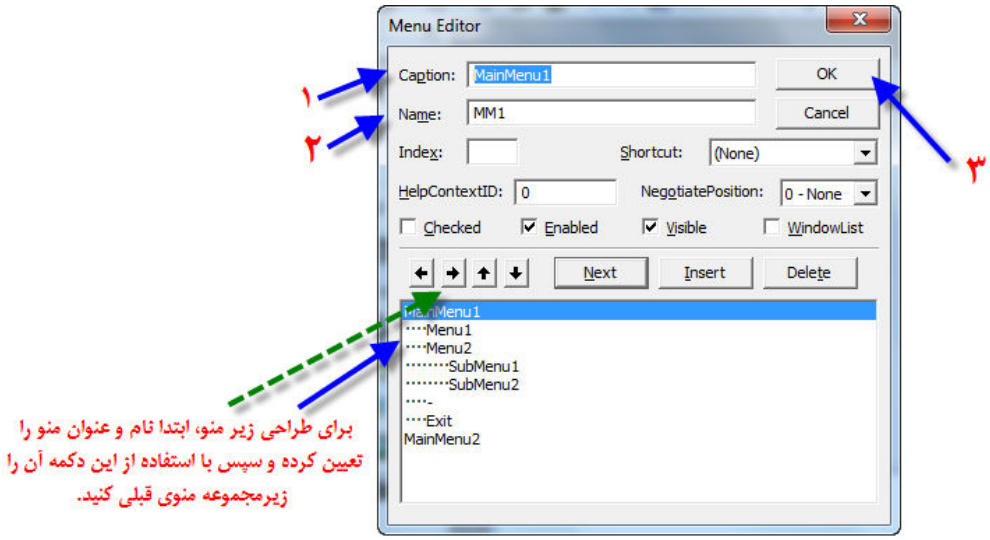
۱) Main Menu

این منوها ثابت بوده و معمولاً در بالای فرم قرار می گیرند. برای توضیح چگونگیِ ساخت این منوها، مراحل طراحی منوی زیر را دنبال می کنیم :



ابتدا با انتخاب Menu Editor از منوی Tools و یا فشار دادن کلیدهای Ctrl+E، پنجرهٔ Menu Editor را باز کنید.

برای هر منو حتماً باید یک عنوان (Caption) و یک نام (Name) در نظر بگیرید. عنوان منو برای نمایش در آیتم های منو مورد استفاده قرار می گیرد و نام منو در زمان کدنویسی استفاده می شود. بنابراین، منوها را به ترتیب زیر طراحی کنید :



فقط برای گزینهٔ SubMenu2 گزینهٔ Checked را علامت بزنید و از لیست Shortcut گزینهٔ Ctrl+T را انتخاب کنید. با علامتدار کردن گزینهٔ Checked برای هر منو یک علامت تیک کنار آن قرار می گیرد. همچنین با مشخص کردن یک کلید میانبر برای هر منو، با فشار دادن آنها در زمان اجرا، دستورات آن منو اجرا خواهد شد.

جهت کد نویسی برای هر منو، کافی است در زمان طراحی فرم، روی منوی مورد نظر کلیک کنید تا به پنجرهٔ کدنویسی هدایت شوید. برای مثال، در پروژهٔ فوق، پس از بستن پنجرهٔ Menu Editor روی منوی Exit کلیک کنید و دستور زیر را وارد کنید تا با کلیک روی این منو در زمان اجرا، برنامه بسته شود :

```
Private Sub MExit_Click()
 End
End Sub
```

۲) Poppup Menu

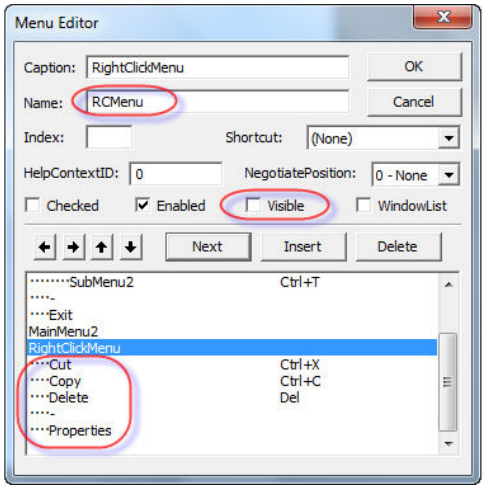
این منوها در هر جایی از فرم می توانند ظاهر شوند. بیشترین موارد استفاده از این نوع منوها، زمان کلیک راست در محل خاصی از فرم است. برای ایجاد چنین منوهایی نیز از Menu Editor استفاده می شود. در واقع، در زمان مورد نیاز می توان قسمتی از منوی تعریف شده به وسیلهٔ Menu Editor را در محل اشاره گر یا در محل مورد نظر ظاهر کرد. در ویژوال بیسیک با استفاده از دستور PopupMenu می توان قسمتی از منوها را به همراه زیر منوهای آن ظاهر کرده و از آن استفاده نمود. شکل کلی این دستور چنین است :

```
PopupMenu نـام مـنو, [Xpos] [, Ypos]
```

توجه داشته باشید که Xpos و Ypos دو مقدار اختیاری و عددی هستند که مختصات محل ظاهر شدن منو را تعیین می کنند. اگر این دو مقدار را ننویسید، VB بطور خودکار از مختصات محل فعلی ماوس به جای این دو متغیر استفاده می کند.

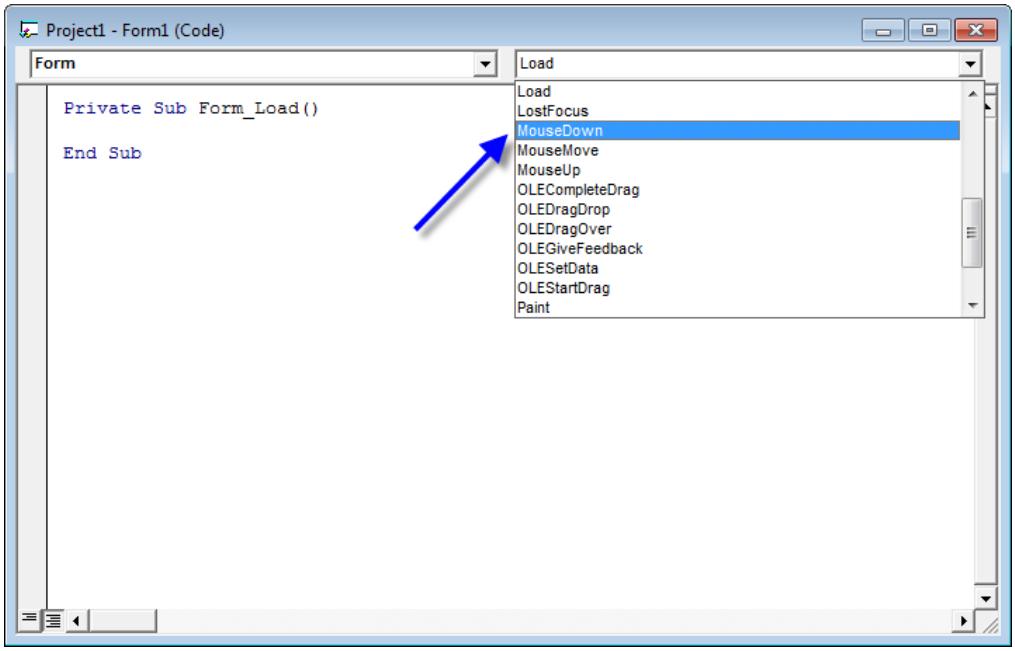
مثال) فرض کنید قصد داریم به برنامهٔ قبلی که MainMenu را طراحی کرده بودیم، امکان ظاهر شدنِ منوی Popup، اضافه کنیم.

برای این کار ابتدا باید به پنجرهٔ Menu Editor رفته و منوی مورد نظر را طراحی کنید :



توجه داشته باشید که مشخصهٔ Visible فقط منوی اصلی باید غیرفعال باشد؛ زیرا قصد داریم این منو در Main Menu نشان داده نشود. بلکه فقط با کلیک راست بر روی فرم ظاهر گردد. مشخصهٔ Name این منو را نیز برابر با RCMenu قرار دهید. پس از تعریف نام و تنظیم مشخصهٔ Visible منوی اصلی، باید بقیه منو ها را بصورت زیرمنو ایجاد کنید، زیرا فقط زیر منوهای آن (منظور زیرمنوهای RCMenu است) در PopupMenu نمایان خواهند شد.

پس از طراحی منوها در پنجرهٔ Menu Editor، رو فرم دابل کلیک کنید تا به پنجرهٔ کدنویسی هدایت شوید. در این پنجره، از لیست رویدادها (Events) گزینهٔ MouseDown را انتخاب کنید. این رویداد زمانی رخ می دهد که هر یک از دکمه های ماوس روی فرم فشار داده شود.



پس از این کار بدنهٔ این رویداد به پنجره کدنویسی اضافه می شود :

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

End Sub
```

همانطور که ملاحظه می کنید، این رویداد دارای چندین پارامتر است. این پارامتر ها عبارتند از :

Button : مشخص می کند که دکمهٔ چپ ماوس زده شده است یا دکمهٔ راست. اگر دکمهٔ چپ باشد مقدار این پارامتر ۱ و اگر دکمهٔ راست باشد، مقدار آن ۲ خواهد بود.

Shift : وضعیت کلیدهای Ctrl، Shift و Alt را در زمان رخ دادن این رویداد، بررسی می کند. مقادیر این پارامتر ممکن است هر یک از موارد زیر باشد :

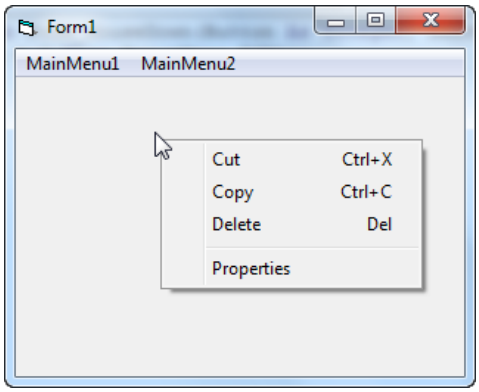
کلید مورد نظر	مقدار
هیچکدام	۰
Shift	۱
Ctrl	۲
Ctrl + Shift	۳
Alt	۴
Alt + Shift	۵
Alt + Ctrl	۶

X و Y نیز مختصات موقعیت فعلی ماوس را نشان می دهند.

اکنون که با این رویداد و پارامترهای آن آشنا شدید، به برنامه باز گردید و دستورات زیر را در آن تایپ کنید :

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
 If Button = 2 Then PopupMenu RCMenu
End Sub
```

این کد بررسی می کند که اگر کاربر روی فرم کلیک راست کرد، PopupMenu طراحی شده (زیرمنوهای RCMenu) ظاهر گردد :



همانطور که ملاحظه می کنید، RCMenu قابل رؤیت نیست، اما زیرمنوهای آن به عنوان Popup به نمایش درآمده اند.

ذخیره پروژهٔ و ساخت فایل اجرایی (EXE)

در ویژوال بیسیک، مانند سایر نرم افزارها باید پروژه هایی که انجام می دهید، ذخیره کنید تا بعداً قابل بازیابی باشند. برای ذخیرهٔ پروژه و فایل های مربوطه در ویژوال بیسیک از منوی فایل، گزینهٔ Save Project را انتخاب کنید. با این کار در اولین بار، یک کادر محاوره ای نمایان شده و از شما مسیر و نام فایل پروژه را برای ذخیره سازی می پرسد. پس از تعیین نام و مسیر ذخیرهٔ فایل پروژه، فایل مربوطه در آنجا ذخیره خواهد شد.

با این حال، برای هر بار اجرای برنامه ای که نوشته اید، باید ابتدا ویژوال بیسیک اجرا و سپس پروژه تان را در آن باز کنید تا بتوانید برنامه تان را اجرا نمائید. به جای این کار بهتر است یک فایل اجرایی از برنامه خود بسازید و فقط آن را اجرا کنید.

برای این کار از منوی File گزینهٔ Make ProjectName.exe (که ProjectName نام پروژه تان خواهد بود) را انتخاب کرده و مسیر و نام فایل اجرایی را مشخص کنید تا ویژوال بسیک یک فایل اجرایی (با پسوند exe) از پروژتان بسازد.

پایان