

آموزش گام به گام

# برنامه نویسی بانک اطلاعاتی با C#

مرجع کامل

تألیف: مهندس رمضان عباس نژاد



آدرس سایت انتشارات: [www.fanavarinovin.net](http://www.fanavarinovin.net)

---

---

آموزش گام به گام برنامه نویسی  
**بانک اطلاعاتی با**  
**C#**  
**(مرجع کامل)**

---

---

**تالیف**

مهندس رمضان عباس نژادورزی



فن آوری نوین

---

---

سرشناسه	: عباس نژادورزی ، رمضان، ۱۳۴۸ -
عنوان و نام پدیدآور	: آموزش گام به گام برنامه‌نویسی بانک اطلاعاتی با #C (مرجع کامل)/تالیف رمضان عباس نژادورزی.
مشخصات نشر	: بابل: فن آوری نوین، ۱۳۸۸.
مشخصات ظاهری	: ۳۰۴ص: مصور، جدول.
شابک	: ۶۵۰۰۰ ریال: 978-600-91413-2-6
وضعیت فهرست نویسی	: فیپا
موضوع	: سی شارپ (زبان برنامه‌نویسی کامپیوتر)
موضوع	: پایگاه‌های اطلاعاتی -- مدیریت
رده بندی کنگره	: QA۷۳/۷۶/س۹۵ ۲ع ۱۳۸۸
رده بندی دیویی	: ۰۰۵/۱۳۳
شماره کتابشناسی ملی	: ۴۱۰۸۰۹۱



فن آوری نوین

www.fanavarinovin.net

بابل، صندوق پستی ۷۳۴۴۸-۴۷۱۶۷

تلفن: ۰۱۱۱-۲۲۵۶۶۸۷

## آموزش گام به گام برنامه نویسی بانک اطلاعاتی با #C (مرجع کامل)

تألیف: مهندس رمضان عباس نژادورزی

ناشر: فن آوری نوین

چاپ اول: پاییز ۱۳۸۸

جلد: ۲۰۰۰

شابک: ۶-۲-۹۱۴۱۳-۶۰۰-۹۷۸

حروفچینی و صفحه‌آرایی: فن آوری نوین

قیمت: ۶۵۰۰ تومان

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۱۴۴-۶۶۴۰۰۲۲۰

## فهرست مطالب

۱-۱۳-۲. نمونه‌سازی کلاس ..... ۳۶

۱-۱۳-۳. اعضای کلاس ..... ۳۶

### فصل دوم: محیط بانک اطلاعات

۲-۱. تعریف سیستم مدیریت

بانک‌اطلاعات ..... ۴۵

۲-۱-۱. دلایل استفاده از بانک اطلاعات ... ۴۶

۲-۱-۲. طراحی بانک اطلاعاتی ..... ۴۷

۲-۱-۳. نرمال‌سازی داده‌ها ..... ۴۸

۲-۲. بانک اطلاعات SQL Server ..... ۴۸

۲-۳. معرفی بانک اطلاعاتی نمونه ..... ۴۹

۲-۴. ورود به بانک اطلاعاتی SQL

Server ..... ۵۲

۲-۵. تایپ و اجرای دستورات SQL ..... ۵۲

### فصل سوم: ایجاد بانک اطلاعات و جدول

۳-۱. ایجاد بانک اطلاعات و جداول آن .. ۵۵

۳-۱-۱. ایجاد بانک اطلاعات ..... ۵۵

۳-۱-۲. تغییر خواص بانک اطلاعات

موجود ..... ۵۷

۳-۱-۳. حذف بانک اطلاعات ..... ۵۸

۳-۲. اشیای بانک اطلاعات ..... ۵۹

۳-۲-۱. ایجاد جدول با دستور SQL ..... ۶۰

۳-۲-۲. تغییر ساختار جدول با

دستور SQL ..... ۶۲

۳-۲-۳. حذف جدول با دستور SQL ..... ۶۳

۳-۳. دستیابی به بانک اطلاعاتی با

ADO.NET ..... ۶۴

۳-۳-۱. فضای نام System.Data ..... ۶۸

۳-۳-۲. تأمین‌کننده‌های داده ..... ۷۰

۳-۳-۳. کلاس‌های Connection ..... ۷۰

۳-۳-۴. واسط IDbCommand ..... ۷۷

### فصل چهارم: کلاس‌های پایه بانک اطلاعات

۴-۱. کلاس DataSet ..... ۸۸

۴-۱-۱. پر کردن DataSet ..... ۹۱

### فصل اول: آشنایی با زبان C#

۱-۱. زبان C# ..... ۸

۱-۲. فضای نام ..... ۸

۱-۳. انواع داده‌ها ..... ۹

۱-۴. متغیرها ..... ۹

۱-۴-۱. نامگذاری متغیرها ..... ۹

۱-۴-۲. اعلان متغیرها ..... ۹

۱-۴-۳. مقدار دادن به متغیرها ..... ۱۰

۱-۵. ثوابت ..... ۱۰

۱-۶. عملگرها ..... ۱۱

۱-۷. فرم برنامه ..... ۱۲

۱-۷-۱. خواص فرم ..... ۱۲

۱-۷-۲. رویدادهای فرم ..... ۱۳

۱-۷-۳. متدهای فرم ..... ۱۵

۱-۸. کنترل‌ها ..... ۱۵

۱-۸-۱. کنترل label ..... ۱۶

۱-۸-۲. کنترل TextBox ..... ۱۶

۱-۸-۳. کنترل Button ..... ۱۶

۱-۸-۴. کنترل ListBox ..... ۱۷

۱-۸-۵. کنترل ComboBox ..... ۱۸

۱-۸-۶. کنترل CheckBox ..... ۱۹

۱-۸-۷. کنترل CheckedListBox ..... ۱۹

۱-۸-۸. کنترل RadioButton ..... ۱۹

۱-۸-۹. کنترل GroupBox ..... ۲۰

۱-۸-۱۰. کنترل MenuStrip ..... ۲۰

۱-۸-۱۱. کنترل ContextMenuStrip ..... ۲۰

۱-۸-۱۲. کنترل PictureBox ..... ۲۰

۱-۹. ساختارهای کنترلی ..... ۲۱

۱-۹-۱. ساختارهای تصمیم ..... ۲۱

۱-۱۰. ساختارهای تکرار ..... ۲۴

۱-۱۱. مدیریت صفحه‌کلید ..... ۲۵

۱-۱۲. آرایه‌ها ..... ۲۶

۱-۱۳. کلاس‌ها و اشیاء ..... ۳۵

۱-۱۳-۱. تعریف کلاس ..... ۳۵

۱۶۱.....	۶-۵. پرس و جوی فرعی
۱۶۴.....	۶-۶. پرس و جوهای مرکب
۱۶۳.....	۶-۷. کلاس DataAdapter
۱۶۴... ..	۶-۷-۱. ایجاد شیء SqlDataAdapter
۱۶۶.....	۶-۷-۲. بازیابی نتایج پرس و جو
.....	۶-۷-۳. پر کردن DataTable با
۱۶۶.....	DataAdapter
۱۶۶... ..	۶-۷-۴. رویدادهای SqlDataAdapter
۱۶۷.....	۶-۸. کلاس DataReader
.....	۶-۸-۱. سازنده‌های کلاس
۱۶۸.....	SqlDataReader
۱۶۹.....	۶-۸-۲. متد ExecuteNonQuery
۱۷۰.....	۶-۸-۳. متد ExecuteScalar
۱۷۰.....	۶-۹. کاتالوگ

### فصل هفتم: آشنایی پیشرفته SQL Server و ADO.NET

۱۸۱.....	۷-۱. دید
.....	۷-۱-۱. ایجاد دید
۱۸۲.....	۷-۱-۲. تغییر دید
۱۸۲.....	۷-۱-۳. حذف دید
۱۸۳.....	۷-۲. متغیرها
۱۸۴.....	۷-۳. توضیحات
۱۸۴.....	۷-۴. ساختارهای تصمیم
۱۸۴.....	۷-۴-۱. دستور IF ... Else
۱۸۵.....	۷-۴-۲. دستور IF تو در تو
۱۸۶.....	۷-۴-۳. دستور CASE
۱۸۶.....	۷-۵. رویه‌های ذخیره شده
.....	۷-۵-۱. ایجاد رویه‌های ذخیره شده
۱۸۹.....	۷-۵-۲. اجرای رویه ذخیره شده
۱۸۹.....	۷-۵-۳. تغییر رویه ذخیره شده
۱۹۰.....	۷-۵-۴. حذف رویه‌های ذخیره شده
۱۹۰.....	۷-۶. توابع
.....	۷-۶-۱. ایجاد توابع
۱۹۲.....	۷-۶-۲. تغییر تابع
۱۹۲.....	۷-۶-۳. حذف تابع
۱۹۳.....	۷-۷. ارسال پارامترها

### فصل هشتم: تراکنش‌ها

۲۰۱.....	۸-۱. تراکنش چیست؟
۲۰۴.....	۸-۲. ارجاع به تراکنش
۲۰۵.....	۸-۳. کلاس‌های پیاده‌سازی تراکنش
.....	۸-۳-۱. TransactionScope
۲۰۶... ..	۸-۳-۲. CommittableTransaction

.....	۴-۱-۲. پر کردن DataSet با چندین
.....	۴-۱-۳. تثبیت تغییرات DataSet
.....	۴-۱-۴. دسترسی به جداول DataSet
.....	۴-۲. کلاس DataTable
.....	۴-۲-۱. کلاس DataColumn
.....	۴-۲-۲. کلاس
.....	۴-۳. کنترل DataGridView

### فصل پنجم: ورود و ویرایش داده‌ها

.....	۵-۱. دستورات SQL برای ورود، ویرایش و حذف داده‌ها
.....	۵-۱-۱. دستور INSERT
.....	۵-۱-۲. ویرایش رکوردهای جدولی
.....	۵-۱-۳. حذف رکوردهای جدول
.....	۵-۲. انقیاد داده‌ها
.....	۵-۲-۱. انقیاد کنترل‌های ساده
.....	۵-۲-۲. انقیاد کنترل‌های پیچیده
.....	۵-۳. ۱۳۱
.....	۵-۳-۱. مرتب‌سازی DataView
.....	۵-۳-۲. فیلتر رکوردهای DataView
.....	۵-۳-۳. جستجوی رکورد خاص
.....	۵-۴. ذخیره تصاویر در بانک اطلاعاتی
.....	۵-۴-۱. بازیابی تصاویر از بانک اطلاعاتی
.....	۵-۴-۲. انقیاد کنترل‌ها به فیلد تصاویر

### فصل ششم: بازیابی داده‌ها

.....	۶-۱. پرس و جو
.....	۶-۲. عملگرها در SQL Server
.....	۶-۲-۱. عملگرهای محاسباتی
.....	۶-۲-۲. عملگر انتساب
.....	۶-۲-۳. عملگرهای بیتی
.....	۶-۲-۴. عملگرهای رابطه‌ای
.....	۶-۲-۵. عملگرهای منطقی
.....	۶-۲-۶. عملگرهای یکانی
.....	۶-۲-۷. عملگرهای اتصال رشته‌ای
.....	۶-۲-۸. عملگرهای ویژه
.....	۶-۳. بازیابی اطلاعات از جداول
.....	۶-۴. پیوند جداول
.....	۶-۴-۱. الحاق ضربدری
.....	۶-۴-۲. الحاق متعادل
.....	۶-۴-۳. الحاق نامتعادل
.....	۶-۴-۴. الحاق درونی
.....	۶-۴-۵. الحاق بیرونی

۲۰۷	..... SqlTransaction کلاس	۸-۳-۳
۲۰۹	..... سطح‌های جداسازی تراکنش	۸-۴
۲۰۹	..... نقاط ذخیره	۸-۵

## فصل نهم: گزارش‌گیری با Crystal Reports

۲۲۱	.. امکانات نرم‌افزار Crystal Report	۹-۱
۲۲۱	..... مراحل طراحی گزارش	۹-۲
۲۲۲	..... ایجاد گزارش با ویزارد	۹-۳
۲۲۸	..... بخش‌های گزارش	۹-۴
۲۳۰	..... اضافه کردن فیلد متن به گزارش	۹-۵
۲۳۲	..... اضافه کردن خط به گزارش	۹-۶
۲۳۲	..... اضافه کردن مستطیل به گزارش	۹-۷
۹-۸	افزودن فیلد بانک اطلاعاتی	۹-۸
۲۳۲	..... به گزارش	۹-۹
۲۳۳	..... افزودن تصویر در گزارش	۹-۱۰
۲۳۳	..... اضافه کردن فیلدهای ویژه در گزارش	۹-۱۱
۲۳۴	..... اضافه کردن فرمول به گزارش	۹-۱۲
۲۳۶	..... مرتب‌سازی رکوردهای گزارش	۹-۱۳
۹-۱۳	افزودن کردن گروه در نمایش	۹-۱۳
۲۳۷	..... رکوردها	۹-۱۳-۱
۹-۱۳-۱	انتخاب‌گزینه‌های	۹-۱۴
۲۳۸	..... گروه‌بندی اطلاعات	۹-۱۴
۹-۱۴	افزودن فیلدهای مجموع	۹-۱۵
۲۳۸	..... در گزارش	۹-۱۶
۲۴۰	..... اضافه کردن نمودار به گزارش	۹-۱۷
۲۴۳	.. کنترل Crystal Report Viewer	۹-۱۷
۹-۱۷	فضای نام	۹-۱۷
۲۴۳	..... CrystalDecisions.Shared	۹-۱۷

۹-۱۸	اتصال به بانک اطلاعات از طریق کد	۹-۱۸
۲۴۴	..... Crystal Reports در	۹-۱۹
۲۴۵	..... افزودن پارامتر به گزارش	۹-۲۰
۹-۲۰	نمایش رکوردهای خاص	۹-۲۰
۲۴۶	..... از گزارش	۹-۲۰-۱
۹-۲۰-۱	استفاده از خاصیت	۹-۲۰-۱
RecordSelectionFormula	برای فیلتر	۹-۲۰-۱
۲۴۷	..... رکوردها	۹-۲۰-۲
۹-۲۰-۲	روش اتصال گزارش به	۹-۲۰-۲
۲۴۷	..... DataSet	۹-۲۰-۲

## فصل دهم: پشتیبان‌گیری و بازیابی پشتیبان از بانک اطلاعات

۱۰-۱	پشتیبان‌گیری با دستور BACKUP	۱۰-۱
۲۵۹	..... DATABASE	۱۰-۲
۲۶۱	..... دستور RESTOREDATABASE	۱۰-۳
۲۶۷	..... مدیریت دایرکتوری و فایل	۱۰-۳-۱
۲۶۸	..... کلاس File	۱۰-۳-۱
۲۶۸	..... کلاس Directory	۱۰-۳-۲
۲۶۹	..... کلاس FileStream	۱۰-۳-۳
۲۷۱	..... کلاس GzipStream	۱۰-۳-۴
۲۷۳	..... کنترل SQLDMO	۱۰-۴
۲۷۴	..... اشیای SQLDMO	۱۰-۴-۱
۲۷۵	..... شیء SQLDMO.SQLServer	۱۰-۴-۲
۲۷۷	..... شیء SQLDMO.NameList	۱۰-۴-۳
۲۷۷	..... شیء SQLDMO.DataBase	۱۰-۴-۴
۲۷۷	..... شیء SQLDMO.Backup	۱۰-۴-۵
۲۷۷	..... شیء SQLDMO.Restore	۱۰-۴-۶
۳۰۲	..... منابع	۱۰-۴-۶

## مقدمه

امروزه حجم زیادی از اطلاعات ذخیره و بازیابی می‌شوند. برای جلوگیری از افزونگی داده (تکرار بی‌مورد داده‌ها)، بی‌نظمی و ایجاد سازگاری بین گزارش‌ها از بانک اطلاعات استفاده می‌شود. بانک‌های اطلاعات متعددی وجود دارند که پرکاربردترین و بهترین آنها، سیستم بانک اطلاعات رابطه‌ای است. یکی از بانک‌های اطلاعات رابطه‌ای، **SQL Server** است که ویژگی‌هایی از قبیل کارایی بالا، سهولت یادگیری و استفاده، کارکردن در محیط شبکه، قابلیت دسترسی و امنیت بالا، آن را به عنوان پرکاربردترین بانک اطلاعات جهان تبدیل کرده است. به طوری که ۷۰ درصد کاربران دنیا از این بانک اطلاعات استفاده می‌کنند.

از طرف دیگر، با بانک اطلاعات **SQL Server** نمی‌توان کارهایی از قبیل ایجاد فرم‌های ورود و ویرایش اطلاعات، تهیه گزارش‌ها و غیره را انجام داد. به همین دلیل، برای انجام کارهای مذکور به زبان برنامه‌نویسی نیاز داریم. از آنجایی که **C#** به عنوان یکی از دروس رشته‌های کامپیوتر و **IT** در دانشگاه‌های کشور تدریس می‌شود، این زبان را به عنوان زبان برنامه‌نویسی بانک اطلاعات انتخاب نمودیم.

در این کتاب مطالبی از قبیل ایجاد بانک اطلاعات، جداول، کلاس‌های **C#** برای کار با بانک اطلاعات، ورود، ویرایش، حذف رکوردها، رویه‌های ذخیره شده، پشتیبان‌گیری و بازیابی فایل‌های پشتیبان به صورت فشرده شده، تراکنش‌ها و گزارش‌گیری از بانک اطلاعات بیان گردید.

کتاب حاضر با بهره‌گیری از سال‌ها تجربه در امر تدریس، تألیف کتب کامپیوتر و مهم‌تر از همه برنامه‌نویسی در زمینه بانک اطلاعات تدوین شده است. از ویژگی‌های جالب و برجسته این کتاب، بیان مثال‌های متنوع کاربردی، حل گام به گام آنها و توضیح کامل مثال‌های بیان شده، می‌باشد.

در پایان امیدوارم این اثر مورد توجه جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

کتاب حاوی برنامه‌های است که کد آنها را می‌توانید به صورت رایگان از سایت انتشارات فن‌آوری نوین به آدرس [www.fanavarinovin.net](http://www.fanavarinovin.net) بگیرید.

عباس نژاددوری

[fanavarienovin@yahoo.com](mailto:fanavarienovin@yahoo.com)



## آشنایی با زبان C#

سازمان‌ها برای نگهداری داده‌ها از بانک اطلاعاتی استفاده می‌کنند. یکی از پرکاربردترین نرم‌افزارهای مدیریت بانک‌های اطلاعات، SQL Server است. از طرف دیگر، بانک اطلاعات به تنهایی نمی‌تواند نیازهای سازمان‌ها را برطرف کند. به همین دلیل با یکی از زبان‌های برنامه‌سازی باید بتوان به بانک اطلاعات متصل شده داده‌های آن را ویرایش، حذف و اضافه نمود. امروزه صدها زبان برنامه‌سازی وجود دارند که از طریق آنها می‌توان به بانک اطلاعات متصل گردید و داده‌های آن را دستکاری نمود. یکی از مهم‌ترین و پرکاربردترین زبان‌های برنامه‌سازی، C# می‌باشد. به همین دلیل در این فصل به طور خلاصه زبان C# را می‌آموزیم.

## ۱-۱. زبان C#

این زبان به همراه نرم‌افزار ویژوال استودیو ارائه شده است. زبان C# از فناوری شیء<sup>۱</sup> و مفهوم شیء‌گرایی<sup>۲</sup> استفاده می‌کند. هر چیزی که در دنیای واقعی وجود دارد، شیء نامیده می‌شود. مثل مردم، ساختمان‌ها، کارخانه‌ها، گیاهان، اتومبیل‌ها، کامپیوترها و غیره. هر شیء از **صفاتی** مانند اندازه، رنگ، وزن و دیگر صفات تشکیل شده است که شکل ظاهری آن را تعیین می‌کنند و رفتارهایی از خودشان نشان می‌دهند، مانند انسان می‌خوابد، گریه می‌کند، می‌خندد، اتومبیل حرکت می‌کند، ترمز می‌نماید و غیره. از آنجایی که زبان C# از فناوری شیء‌گرایی استفاده می‌نماید، امکاناتی در این زبان اضافه شده است که می‌توانید اشیای دنیای واقعی را مدل‌سازی کنید. برای این که C# شیء‌گرایی را پیاده‌سازی کند از مفهوم **کلاس**<sup>۳</sup> استفاده می‌کند. **کلاس**، برای ایجاد انواع جدید به کار می‌رود. هر کلاس شامل داده‌ها و متدهایی است که داده‌ها را دستکاری می‌کنند و سرویس‌هایی را برای مشتریان فراهم می‌نمایند. با مفهوم کلاس و چگونگی پیاده‌سازی آنها در ادامه بیشتر آشنا خواهیم شد.

## ۱-۲. فضای نام

همان‌طور که بیان گردید، زبان برنامه‌نویسی C# از کلاس برای برنامه‌نویسی استفاده می‌کند. کلاس‌ها به دو دسته تقسیم می‌شوند که عبارتند از:

۱. **کلاس‌های آماده:** این کلاس‌ها از قبل نوشته شده‌اند و در کتابخانه FCL ویژوال استودیو وجود دارند.

۲. **کلاس‌هایی که برنامه‌نویس می‌نویسد:** همه کلاس‌های مورد نیاز برنامه‌نویسان از قبل وجود ندارند. بنابراین برنامه‌نویس نیاز دارد، برخی از کلاس‌ها را بنویسد. در ادامه با این کلاس‌ها آشنا خواهید شد.

---

<sup>۱</sup>- Object

<sup>۲</sup>- Object Oriented

<sup>۳</sup>- Class

## ۹ آشنایی با زبان C#

کلاس‌هایی که در کتابخانه FCL وجود دارند، در **فضاهای نام**<sup>۱</sup> مختلف قرار می‌گیرند. برخی از فضای نام‌ها و وظایف آنها در جدول ۱-۱ آمده است. این فضاهای نام به طور خودکار به برنامه C# اضافه می‌شوند. علاوه بر این فضاهای نام، فضاهای نام دیگری وجود دارند که می‌توانید آنها را به برنامه اضافه کنید. برای این منظور باید از دستور using استفاده نمایید. به عنوان مثال، دستورات زیر را ببینید:

```
using System.Data.SqlClient;  
using System.Convert;
```

دستور اول، فضای نام System.Data.SqlClient را به برنامه اضافه می‌کند تا بتوانید از کلاس‌هایی که برای اتصال و دستکاری به بانک اطلاعات SQL Server به کار می‌روند، بهره بگیرید (با این فضای نام در ادامه بیشتر آشنا خواهید شد) و دستور دوم، فضای نام System.Convert را به برنامه اضافه می‌کند تا بتوانید از متدهایی که برای تبدیل انواع داده‌های مختلف به کار می‌روند، استفاده نمایید.

جدول ۱-۱ برخی از فضاهای نام موجود در FCL.	
فضای نام	هدف
System	حاوی کلاس‌های پایه و انواع داده از قبیل double، int، char و غیره است.
System.Data	دارای کلاس‌هایی است که برای دستیابی به داده‌های بانک اطلاعات استفاده می‌شوند.
System.IO	از کلاس‌هایی تشکیل شده است که برای ورودی-خروجی داده‌ها مانند فایل‌ها به کار می‌روند.
System.Drawing	از کلاس‌هایی تشکیل شده است که برای ترسیم اشکال گرافیکی به کار می‌رود.
System.Linq	از کلاس‌هایی تشکیل شده است که برای کار با LINQ به کار می‌روند.

### ۳-۱. انواع داده‌ها

اکثر برنامه‌ها با دریافت داده‌ها، پردازش و استخراج نتایج سر و کار دارند. یعنی، داده‌ها مهم‌ترین بخش برنامه‌نویسی را ایفا می‌نمایند. لذا، باید انواع داده‌هایی که در زبان برنامه‌نویسی وجود دارند، را بیاموزیم. دو نوع داده را می‌توان در زبان C# تعریف نمود که عبارتند از:

۱. داده‌های مقدار ۲. داده‌های مرجع

**داده‌های مقدار**، همان داده‌هایی هستند که توسط انواع اولیه تعریف می‌شوند (بجز داده‌های Object و String). این انواع در جدول ۲-۱ آمده‌اند و **داده‌های مرجع**، تمام انواعی هستند که توسط برنامه‌نویس تعریف می‌شوند یا کلاس‌های هستند که از قبل تعریف شده‌اند. در ادامه پیاده‌سازی کلاس می‌آموزیم.

<sup>۱</sup> - Namespaces

جدول ۱-۲ انواع داده‌های اولیه در C#			
نوع در زبان C#	معادل .NET	اندازه	نگهداری می‌کند.
byte	Byte	۱	مقادیر بدون علامت (۰ تا ۲۵۵)
char	Char	۱	کارکترهای یونیکد
bool	Boolean	۱	مقادیر True یا False
sbyte	Sbyte	۱	مقادیر علامت‌دار (۱۲۸- تا ۱۲۷)
short	Int16	۲	مقادیر صحیح از ۳۲۷۶۸- تا ۳۲۷۶۷
ushort	UInt16	۲	مقادیر صحیح بدون علامت از ۰ تا ۶۵۵۳۵
int	Int32	۴	مقادیر صحیح بین ۲ <sup>۳۱</sup> - تا (۱- ۲ <sup>۳۱</sup> )
uint	UInt32	۴	مقادیر صحیح بدون علامت ۰ تا ۲ <sup>۳۲</sup>
float	Single	۴	اعداد اعشاری ۱۰ <sup>-۴۵</sup> × ۵/۱ ± تا ۱۰ <sup>۳۸</sup> × ۳/۴ ± با ۷ رقم اعشار
double	Double	۸	اعداد اعشاری ۱۰ <sup>-۳۲۴</sup> × ۵/۱ ± تا ۱۰ <sup>۳۰۸</sup> × ۷/۱ ± با ۱۵ تا ۱۶ رقم بعد از اعشار
decimal	Decimal	۸	نقطه اعشار ثابت تا ۲۸ رقم
Long	Int64	۸	مقادیر صحیح ۲ <sup>۶۳</sup> - تا (۱- ۲ <sup>۶۳</sup> )
ulong	UInt64	۸	مقادیر ۰ تا (۱- ۲ <sup>۶۴</sup> )

## ۴-۱. متغیرها

متغیرها نامی برای کلمات حافظه هستند که دارای ویژگی‌های زیر می‌باشند:

- داده‌ها در آنها ذخیره می‌شوند.
  - مقدار آنها در طول اجرای برنامه ممکن است تغییر کند.
  - در یک لحظه خاص فقط یک مقدار را دارند.
- برای استفاده از متغیرها باید نام، نوع و مقدار آنها را تعیین کرد.

### ۴-۱-۱. نامگذاری متغیرها

برای نامگذاری متغیرها در C# می‌توان از ترکیبی از حروف، ارقام و خط ربط (-) استفاده کرد. به طوری که اولین کارکتر آنها رقم نباشد. به عنوان مثال، sum، count1، i\_1 می‌توانند نام‌هایی برای متغیرها باشند، ولی 1count و book!high نمی‌توانند نام متغیرها باشند. زیرا، اولین نام با رقم 1 شروع شد و در دومین نام کارکتر! استفاده گردید که کارکترهای مجاز نمی‌باشند.

### ۴-۱-۲. اعلان متغیرها

بعد از این که متغیرها را نامگذاری کردید باید نوع آنها را تعیین نمایید. یعنی، باید تعیین کنید متغیر چه نوع داده‌ای را ذخیره می‌کند. متغیرها به صورت زیر اعلان می‌گردند:

### نام متغیر نوع داده سطح دستیابی

سطح دستیابی، تعیین می‌کند که در چه محدوده‌های می‌توانید به متغیرها دستیابی داشته باشید و مهم‌ترین آنها `public` و `private` هستند. `public` موجب می‌شود تا متغیر را بتوانید در کل کلاس دستیابی داشته باشید و `private` موجب می‌شود متغیر را بتوانید در همان بلاک دستیابی داشته باشید. اگر سطح دستیابی ذکر نشود، `private` در نظر گرفته خواهد شد. در ادامه بیشتر با سطوح دستیابی آشنا خواهید شد. اکنون دستورات زیر را ببینید:

```
int x;
double d;
string s1, s2;
```

دستور اول، متغیر `x` را از نوع صحیح تعریف می‌کند. دستور دوم، متغیر `d` را از نوع `double` و دستور سوم، متغیرهای `s1` و `s2` را از نوع رشته‌ای تعریف می‌نماید.

### ۳-۴-۱. مقدار دادن به متغیرها

برای مقداردهی به متغیرها روش‌های متعددی وجود دارد که عبارتند از:

۱. مقداردهی در زمان اعلان متغیر ۲. پس از تعریف نوع متغیر و با دستور انتساب (=)

۳. دستورات ورودی

به عنوان مثال، دستورات زیر را ببینید:

```
int x = 5 , y = 10;
string s = "good";
```

دستور اول، متغیرهای `x` و `y` را با مقادیر `5` و `10` از نوع `int` تعریف می‌کند و دستور دوم، متغیر رشته‌ای `s` را از نوع `string` تعریف می‌نماید و رشته `"good"` را به آن تخصیص می‌دهد. اکنون دستورات زیر را ببینید:

```
int x;
string s;
x = 10;
s = "C#.NET";
```

دستورات اول و دوم متغیرهای `x` و `s` را به ترتیب از نوع `int` و `string` تعریف می‌کنند. دستور سوم، مقدار متغیر `x` را برابر `10` قرار می‌دهد و دستور چهارم، مقدار متغیر `s` را رشته `C#.NET` تعیین می‌کند. در ادامه با چگونگی مقداردهی متغیرها با دستورات ورودی آشنا خواهید شد.

### ۵-۱. ثوابت

ثوابت، مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند. برای تعریف ثوابت از واژه `const` به صورت زیر استفاده می‌شود:

`const` مقدار ثابت = نام نوع داده

به عنوان مثال، دستورات زیر را ببینید:

```
const float PI = 3.14;
const char ch = '+';
```

دستور اول، ثابت PI را با مقدار 3.14 از نوع اعشاری معرفی می‌کند و دستور دوم، ثابت ch را از نوع کارکتری با مقدار '+' تعریف می‌نماید.

## ۶-۱. عملگرها

عملگرها، نمادهایی هستند که اعمال خاصی را انجام می‌دهند. به عنوان مثال، نماد '-' عملگری است که دو مقدار را از یکدیگر تفریق می‌کند. عملگرها در C# به انواع مختلف تقسیم می‌شوند که در زیر آمده‌اند:

➤ **عملگرهای محاسباتی**، عملگرهایی هستند که عمل محاسبات را بر روی عملوند انجام می‌دهند. این عملگرها در جدول ۱-۳ آمده‌اند.

➤ **عملگرهای رابطهای**، عملگرهایی هستند که دو عملوند را با یکدیگر مقایسه می‌کنند (جدول ۱-۴ را ببینید).

➤ **عملگرهای منطقی**، عملگرهایی هستند که بر روی عبارت منطقی (True یا False) عمل می‌کنند (جدول ۱-۵ را مشاهده کنید).

➤ **عملگرهای ترکیبی**، از ترکیب عملگرهای محاسباتی و علامت = ایجاد می‌شوند (جدول ۱-۶).

➤ **عملگرهای بیتی**، عملگرهایی هستند که برای تست کردن، مقدار دادن یا شیفت دادن و سایر اعمال بر روی عملوندها به کار می‌روند. عملگرهای بیتی در جدول ۱-۷ آمده‌اند.

عملگر	نام	x	y	مثال	نتیجه
-	تفریق و منهای یکانی	10	12	y-x -y	2 -12
+	جمع	17	5	x+y	22
*	ضرب	10	2	x*y	20
/	تقسیم	10	5	x/y	2
%	باقیمانده تقسیم صحیح	10	3	x%y	1
--	کاهش	10	5	--x, y--	9, 4
++	افزایش	10	5	x++, ++y	11, 6

عملگر	نام	x	y	مثال	نتیجه
>	بزرگتر	10	12	x > y	False
<=	بزرگتر یا مساوی	10	7	x >= y	True
<	کوچکتر	10	7	x < y	True
<=	کوچکتر یا مساوی	10	12	x <= y	False
==	تساوی	10	17	x == y	False
!=	نامساوی	10	17	x != y	True

عملگر	نام	x	y	مثال	نتیجه
!	نقیض (not)	10	12	! x	False
&&	و (and)	10	12	x < 12 && y > 10	True
	یا (or)	12	8	x > 12    y < 6	False

عملگر	نام	X	y	مثال	معادل	نتیجه
+=	انتساب جمع	10	12	x += y	x = x + y	x = 22
-=	انتساب تفریق	10	8	x -= y	x = x - y	x = 2
*=	انتساب ضرب	10	12	x *= y	x = x * y	x = 120
/=	انتساب تقسیم	10	2	x /= y	x = x / y	x = 5
%=	انتساب باقیمانده تقسیم	10	4	x %= y	x = x % y	x = 2

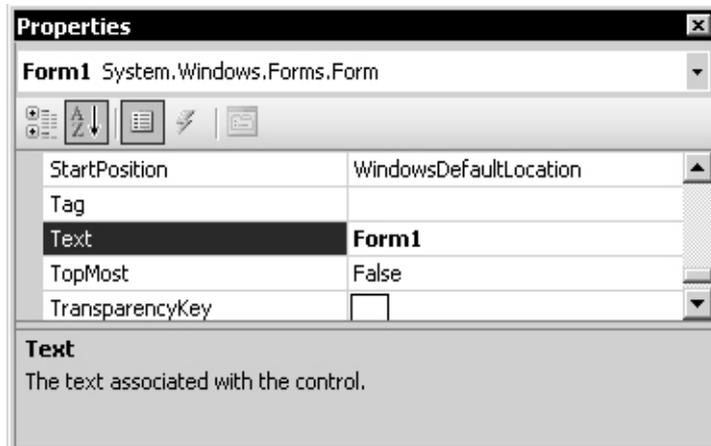
عملگر	نام	x	y	مثال	نتیجه
&	و	10	2	x & y	2
	یا	10	3	x   y	11
^	یا انحصاری	10	3	x ^ y	9
~	نقیض	126	3	~ x	1
>>	شیفت به راست	7	3	x >> y	56
<<	شیفت به چپ	100	2	x << y	25

## ۱-۷. فرم برنامه

فرم برنامه، مکانی است که کنترل‌های برنامه در آن قرار می‌گیرند. هر برنامه بانک اطلاعات در C# حداقل یک فرم دارد. برای استفاده از فرم باید خواص، رویدادها و متدهای آن را بشناسیم. لذا در ادامه به این موضوعات می‌پردازیم.

### ۱-۷-۱. خواص فرم

خواص فرم، شکل ظاهری فرم را تعیین می‌کنند. فرم خواص متعددی دارد. بیان همه این خواص از حوصله این کتاب خارج است. لذا، به تشریح خواص مهم فرم و کنترل‌های دیگر می‌پردازیم. برخی از خواص مهم فرم در جدول ۱-۸ آمده است. برای نمایش خواص فرم، بر روی آن کلیک کنید و سپس کلید F4 را بزنید تا خواص فرم را ببینید (شکل زیر):

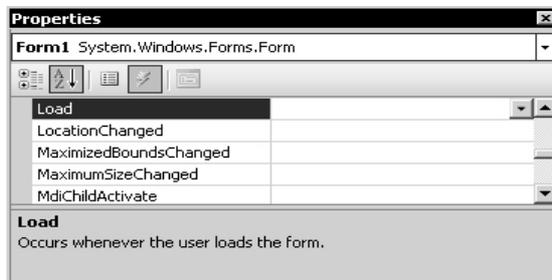


جدول ۱-۸ خواص مهم فرم.	
هدف	خاصیت
	Name نام فرم را تعیین می‌کند. برای اولین فرم، نام فرم Form1 است که می‌توانید آن را تغییر دهید.
	ActiveForm فرم فعال فعلی را تعیین می‌کند.
	Enabled تعیین می‌کند آیا فرم فعال است یا خیر. اگر فرم غیرفعال گردد، به هیچ رویدادی پاسخ نمی‌دهد.
	Font فونت فرم را تعیین می‌کند.
	ForeColor رنگ نوشته‌های روی فرم را تعیین می‌کند.
	Language زبان مورد استفاده در فرم را تعیین می‌کند.
	MinimizeBox تعیین می‌کند آیا دکمه کمینه در عنوان فرم ظاهر شود یا خیر.
	RightToLeft جهت نمایش اطلاعات را تعیین می‌کند (این خاصیت برای زبان فارسی مفید است).
	Size اندازه فرم را تعیین می‌کند.
	Text متنی را تعیین می‌کند که باید در عنوان فرم نمایش داده شود.
	Location محل قرار گرفتن فرم را تعیین می‌کند.

### ۲-۷-۱. رویدادهای فرم

همان‌طور که بیان کردیم، برنامه‌های ویژوال منتظر رخ دادن رویدادهایی هستند که توسط کاربر انتخاب می‌شوند تا به آنها پاسخ دهند. یعنی، اگر برنامه پاسخ‌گو به رویدادی نوشته شده باشد، در صورت انتخاب آن رویداد توسط کاربر، این برنامه اجرا می‌شود. فرم دارای رویدادهای مختلفی است. برخی از مهم‌ترین آنها در جدول ۱-۹ آمده‌اند.

برای مشاهده رویدادهای فرم، بر روی آن کلیک کنید تا فرم انتخاب شود. اکنون گزینه View/ Properties Window را اجرا نمایید تا پنجره Properties ظاهر شود. در این پنجره دکمه Events را کلیک کنید تا لیست رویدادهای فرم را ببینید (شکل زیر):



جدول ۹-۱ رویدادهای مهم فرم.	
رویداد	هدف
Activated	وقتی که فرم فعال می شود، رخ می دهد.
Click	وقتی رخ می دهد که فرم کلیک گردد.
FormClosed	وقتی رخ می دهد که فرم بسته شود.
FormClosing	وقتی رخ می دهد که فرم در حال بسته شدن باشد. این رویداد قبل از رویداد FormClosed رخ می دهد.
Deactivate	وقتی رخ می دهد که فرم غیرفعال گردد.
DoubleClick	وقتی که فرم کلیک مضاعف شود، رخ می دهد.
Enter	وقتی مکان نما وارد فرم می شود، رخ می دهد.
KeyDown	وقتی کلیدی فشرده شده پایین می رود، رخ می دهد.
KeyPress	وقتی کلیدی فشرده می شود، رخ می دهد. این رویداد قبل از رویداد KeyDown اتفاق می افتد.
KeyUp	وقتی کلید فشرده شده رها می گردد، رخ می دهد.
Leave	وقتی مکان نما فرم را ترک می کند رخ می دهد.
Load	وقتی فرمی باز می شود، رخ می دهد (قبل از نمایش فرم).
MouseDown	وقتی که کلید ماوس فشرده شود، رخ می دهد.
MouseEnter	وقتی مکان نمای ماوس وارد فرم شود، رخ می دهد.
MouseLeave	وقتی مکان نمای ماوس فرم را ترک می کند، رخ می دهد.
MouseUP	وقتی کلید فشرده شده ماوس رها می شود، رخ می دهد.
Move	وقتی فرم شروع به حرکت می کند، رخ می دهد.
Resize	وقتی اندازه فرم تغییر می یابد، رخ می دهد.
TextChanged	وقتی رخ می دهد که متن فرم (کنترل) یا خاصیت Text تغییر کند.
Shown	وقتی که فرم نمایش داده شود، رخ می دهد.
SizeChanged	وقتی رخ می دهد که مقدار خاصیت Size تغییر می یابد.
MouseMove	وقتی رخ می دهد که ماوس روی فرم حرکت می کند.

## ۳-۷-۱. متدهای فرم

متدها، کارهای خاصی را بر روی فرم انجام می‌دهند. برخی از متدهای مهم فرم در جدول ۱-۱۰ آمده‌اند.

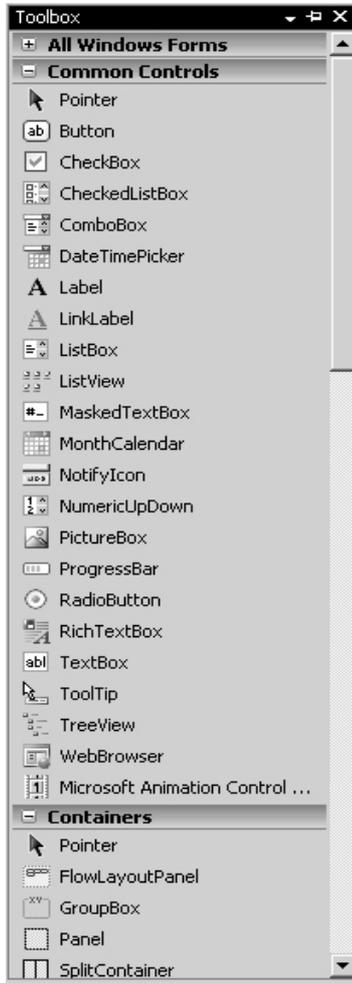
جدول ۱-۱۰ متدهای مهم فرم.	
هدف	متد
فرم را فعال می‌کند.	Active
فرم را می‌بندد.	Close
فرم را حذف کرده از بین می‌برد.	Dispose
مکان‌نما را به فرم مورد نظر منتقل می‌کند.	Focus
فرم را پنهان می‌کند.	Hide
متن عنوان فرم را پاک می‌کند.	ResetText
فرم مخفی شده را آشکار می‌کند.	Show
فرم را بازسازی کرده، اطلاعات آن را دوباره رسم می‌کند.	Refresh
محتویات فرم را به رشته تبدیل می‌نماید.	ToString
موجب اعتبارسنجی فرم می‌شود.	Validate

## ۸-۱. کنترل‌ها

همان‌طور که می‌دانید برای نوشتن برنامه‌ها در C# باید کنترل‌هایی را بر روی فرم قرار دهید (این کنترل‌ها همان قطعات تشکیل دهنده برنامه هستند). خواص آنها را مقداردهی کرده، برنامه پاسخ‌گو به رویدادهای آنها را بنویسید. بنابراین، کنترل‌ها اجزای اصلی برنامه‌های C# را تشکیل می‌دهند. کنترل‌های زیادی در C# وجود دارند. بحث در مورد همه اینها در این کتاب نمی‌گنجد. لذا، در این کتاب برخی از کنترل‌های مهم را بررسی می‌کنیم.<sup>۱</sup> این کنترل را در ادامه خواهید دید. برای اضافه کردن کنترل جدید بر روی فرم، دکمه Toolbox (☒) را کلیک کنید تا کنترل‌های آماده را مشاهده کنید. اکنون جعبه ابزار ظاهر می‌شود (شکل ۱-۱). در این جعبه ابزار، کنترل مورد نظر را کلیک مضاعف کنید تا به فرم اضافه گردد و آن را به مکان دلخواه از فرم انتقال دهید (با کشیدن و رها کردن). من دکمه button1 را به فرم اضافه کردم و به مکان دلخواه انتقال دادم (شکل زیر):



<sup>۱</sup> - برای کسب اطلاعات بیشتر در مورد C# به آموزش گام به گام C# از انتشارات علوم رایانه، تألیف عین‌الله جعفرزادقمی و رمضان عباس‌نژاد مراجعه کنید.



شکل ۱-۱ پنجره ابزار ویژوال استودیو

برای حذف کنترلی از فرم، آن را کلیک کرده تا انتخاب شود. اکنون دکمه Delete را فشار دهید تا کنترل از روی فرم حذف گردد.

### ۱-۸-۱. کنترل Label (A Label)

این کنترل برای نمایش متن‌های غیرقابل ویرایش از قبیل عنوان فیلد، پیام‌های مورد نیاز به کار می‌رود. این کنترل نیز دارای خواص و رویدادهای زیادی است که برخی از مهم‌ترین آنها در زیر آمده است:

**خاصیت AutoSize:** تعیین می‌کند آیا اندازه کنترل با توجه به مقدار خاصیت Text به طور خودکار تغییر کند یا خیر.

**خاصیت TextAlign:** مکان قرار گرفتن متن در کنترل Label را تعیین می‌کند. به عنوان مثال، دستور زیر را ببینید.

```
label1.TextAlign = ContextAlignMent.TopLeft;
```

این دستور، متن را در بالا سمت چپ نمایش می‌دهد.

**رویداد AutoSizeChanged:** وقتی رخ می‌دهد که مقدار خاصیت AutoSize تغییر کند.

**رویداد TextAlignChanged:** وقتی رخ می‌دهد که مقدار خاصیت TextAlign تغییر یابد.

### ۱-۸-۲. کنترل TextBox (abl TextBox)

این کنترل برای دریافت اطلاعاتی از قبیل مقادیر رشته‌ای، عددی و Memo (اطلاعات رشته‌ای طولانی) فیلدها به کار می‌رود. این کنترل نیز مانند کنترل‌های دیگر دارای خواص، رویدادها و متدهای متعددی است. برخی از خواص، رویدادها و متدهای این کنترل در جدول ۱-۱۱ آمده‌اند.

### ۱-۸-۳. کنترل Button (ab Button)

این کنترل‌ها به دکمه فرمان معروف هستند. زیرا با کلیک آن فرمان خاصی (دستورات خاصی) اجرا خواهد شد. خواص، رویدادها و متدهای این کنترل مانند کنترل‌های دیگر است. در ادامه بیشتر با این کنترل آشنا خواهید شد.

### ۱-۸-۴. کنترل ListBox (ListBox)

این کنترل برای نمایش لیستی از اشیاء به کار می‌رود. در این کنترل می‌توان اطلاعاتی از قبیل نام افراد، لیست کالاها و غیره را نمایش داد. این کنترل نیز مانند کنترل‌های دیگر دارای خواص، رویدادها و متدهای زیادی است که برخی از پرکاربردترین آنها در جدول ۱-۱۲ آمده‌اند.

جدول ۱-۱۱ خواص، رویدادها و متدهای کنترل TextBox.	
هدف	خاصیت
رشته‌ای از نوع آرایه است که تعداد خط‌های TextBox را تعیین می‌کند.	Lines
حداکثر طول متن را تعیین می‌کند. به عنوان مثال، اگر در این خاصیت ۵۰ را وارد کنید، کاربر حداکثر می‌تواند ۵۰ کاراکتر را وارد کند.	MaxLength
تعیین می‌کند آیا TextBox می‌تواند چند سطر اطلاعات را بپذیرد یا خیر.	MultiLines
تعیین می‌کند که آیا محتویات TextBox فقط خواندنی باشد یا خیر. اگر اطلاعات TextBox فقط خواندنی باشد، این کنترل مانند Label عمل می‌کند.	ReadOnly
تعیین می‌کند در هنگام ورود اطلاعات در TextBox آیا اطلاعات به صورت کلمه عبور ظاهر شوند (اطلاعات اصلی مخفی گردند یا خیر).	PasswordChar
هدف	رویداد
زمانی رخ می‌دهد که خاصیت MultiLines تغییر یابد.	MultiLineChanged
زمانی رخ می‌دهد که خاصیت ReadOnly تغییر یابد.	ReadOnlyChanged
هدف	متد
محتویات TextBox را پاک می‌کند.	Clear
اثر اجرای فرمان Clear انجام شده را خنثی می‌کند.	ClearUndo
متن انتخاب شده در TextBox را در حافظه موقت کپی می‌کند.	Copy
متن انتخاب شده در TextBox را به حافظه موقت انتقال می‌دهد.	Cut
متن حافظه موقت را در مکان فعلی کپی می‌نماید.	Paste
تأثیر آخرین فرمان انجام شده را خنثی می‌کند.	Undo
برای انتخاب متن TextBox به کار می‌رود.	Select
کل متن TextBox را انتخاب می‌کند.	SelectAll
کلیه متن انتخاب شده TextBox را از حالت انتخاب خارج می‌کند.	DeselectAll

اکنون دستورات زیر را ببینید:

```
listBox1.Clear();
listBox1.Sorted = true;
listBox1.Items.Add ("one");
listBox1.Items.Insert ("zero" , 0);
listBox1.Items.RemoveAt (1);
```

دستور اول، گزینه‌های listBox1 را حذف می‌کند. دستور دوم، گزینه‌های listBox1 را به صورت مرتب شده نمایش می‌دهد. دستور سوم، گزینه one را به listBox1 اضافه می‌کند. دستور چهارم، گزینه "zero" را قبل از گزینه "one" اضافه می‌نماید و دستور پنجم، مقدار مکان دوم listBox1 را حذف می‌کند.

جدول ۱-۱۲ خواص، متدها و رویدادهای کنترل ListBox.	
هدف	خاصیت
گزینه‌هایی را تعیین می‌کند تا بر روی کنترل نمایش داده شوند.	Items
تعیین می‌کند آیا کنترل ListBox می‌تواند چند ستونی باشد یا خیر.	MultiColumn
تعیین می‌کند آیا در کنترل ListBox می‌توان چند گزینه را انتخاب کرد. مقادیر این خاصیت می‌تواند None (انتخاب گزینه امکان‌پذیر نیست)، One (فقط یک گزینه را می‌توان انتخاب کرد)، MultiSimple (چند گزینه را می‌توان انتخاب کرد) و MultiExtended (با کلیدهای Ctrl و Shift می‌توان چند گزینه را انتخاب کرد) را بپذیرد.	SelectionMode
نام منبع داده (بانک اطلاعات) را تعیین می‌کند.	DataSource
رشته‌ای را تعیین می‌نماید که باید در خاصیت DataSource نمایش داده شود.	ValueMember
تعیین می‌کند آیا اطلاعات ListBox مرتب شده باشد یا خیر.	Sorted
لیستی است که گزینه‌های انتخاب شده را نگهداری می‌کند.	SelectedItems
هدف	رویداد
وقتی خاصیت DataSource تغییر یابد، رخ می‌دهد.	DataSourceChanged
وقتی خاصیت DisplayMember تغییر کند، رخ می‌دهد.	DisplayMemberChanged
وقتی خاصیت ValueMember تغییر یابد، رخ می‌دهد.	ValueMemberChanged
هدف	متد
برای اضافه کردن گزینه‌ای به انتهای ListBox به کار می‌رود.	Add
گزینه‌ای را در مکان خاص ListBox اضافه می‌کند.	Insert
تعداد گزینه‌های ListBox را می‌شمارد.	Count
مقداری را از ListBox حذف می‌کند.	Remove
مقداری را از مکان خاصی از ListBox حذف می‌کند.	RemoveAt
مقداری را دریافت کرده اندیس مکان آن را برمی‌گرداند.	IndexOf

### ۵-۸-۱. کنترل ComboBox

این کنترل ترکیبی از یک کنترل TextBox و ListBox است که برای تایپ یا انتخاب گزینه‌ای به کار می‌رود. خواص، رویدادها و متدهای این کنترل مانند کنترل ListBox است.

### ۶-۸-۱. کنترل CheckBox ( CheckBox )

این کنترل برای تعریف گزینه‌هایی با دو انتخاب از قبیل مرد یا زن، جانباز یا غیرجانباز، متأهل یا مجرد به کار می‌رود. این کنترل را می‌توان به فیلدهای منطقی در بانک اطلاعات انقیاد<sup>۱</sup> کرد. در ادامه چگونگی انقیاد این کنترل را به فیلدهای منطقی می‌بینید. خواص و رویدادهای کنترل CheckBox در زیر آمده‌اند:

**خاصیت CheckAlign:** محل قرار گرفتن ✓ (تیک) را در کنار مربع تعیین می‌کند.

**خاصیت Checked:** تعیین می‌کند آیا کنترل CheckBox انتخاب شده است یا خیر (✓ نمایش داده شده است یا خیر).

**خاصیت CheckState:** یکی از سه حالت CheckBox را تعیین می‌کند که می‌تواند مقادیر Unchecked (انتخاب نشده)، Checked (انتخاب شده) یا Indeterminate (غیرفعال) را بپذیرد.

**خاصیت TreeState:** تعیین می‌کند آیا کنترل CheckBox دارای سه حالت است یا خیر.

**رویداد CheckedChanged:** وقتی رخ می‌دهد که خاصیت Checked تغییر یابد.

**رویداد CheckStateChanged:** وقتی رخ می‌دهد که خاصیت CheckState تغییر یابد.

### ۷-۸-۱. کنترل CheckedListBox ( CheckedListBox )

این کنترل ترکیبی از CheckBox و ListBox است. یعنی هر یک از گزینه‌های ListBox دارای حالت انتخاب CheckBox می‌باشند. خواص و رویدادهای این کنترل مانند رویدادهای CheckBox و ListBox است. ولی دو متد زیر به این کنترل اضافه شده است.

**متد SetItemsChecked:** برای مقداردهی به خاصیت Checked این کنترل به کار می‌رود. به عنوان مثال، دستور زیر را ببینید.

```
CheckedListBox1.SetItemChecked (3, true);
```

این دستور حالت چهارمین گزینه انتخاب شده CheckListBox1 را تیک‌دار می‌کند (اندیس گزینه‌ها از صفر شروع می‌شود).

**متد SetItemsCheckState:** مقدار خاصیت CheckState گزینه خاصی را تعیین می‌کند.

### ۸-۸-۱. کنترل RadioButton ( RadioButton )

این کنترل، دکمه‌های رادیویی را ایجاد می‌کند که می‌توان فقط یک گزینه (یکی از آنها) را انتخاب کرد. یعنی، برای ایجاد دکمه‌هایی به کار می‌رود که دارای گزینه‌های ناسازگار هستند. چنانچه بخواهید چند گروه از RadioButton داشته باشید، آنها را باید در کنترل‌های GroupBox مختلف اضافه کنید.

<sup>۱</sup> - Bind

### ۹-۸-۱. کنترل GroupBox

این کنترل برای ایجاد گروه‌های متعدد کنترل‌ها به کار می‌رود. اگر بخواهید چند کنترل را در یک گروه قرار دهید، باید آنها را به یک کنترل GroupBox اضافه کنید. خواص، متدها و رویدادهای این کنترل مانند کنترل‌های دیگر است.

### ۱۰-۸-۱. کنترل MenuStrip

این کنترل برای ایجاد منو به کار می‌رود. چنانچه این کنترل را به فرم اضافه کنید. شکل ۱-۲ ظاهر می‌شود که می‌توانید گزینه‌های منو را در بخش Type Here تایپ کنید. در مثال ۱-۱، چگونگی اضافه کردن منو را خواهید دید. این کنترل دارای خواص زیر است:

- ✚ **خاصیت Enabled:** تعیین می‌کند آیا گزینه منو فعال باشد یا غیرفعال.
- ✚ **خاصیت Checked:** تعیین می‌کند آیا گزینه منو می‌تواند دارای علامت تیک (✓) باشد یا خیر.
- ✚ **خاصیت ShortCut:** کلید میانبری را برای گزینه منو تعیین می‌کند.
- ✚ **خاصیت Text:** عنوان گزینه منو را تعیین می‌کند، ولی اگر در این خاصیت مقدار - را وارد کنید، گزینه منو به عنوان یک جدا کننده در نظر گرفته می‌شود.



شکل ۱-۲ اضافه کردن منو.

### ۱۱-۸-۱. کنترل ContextMenuStrip

این کنترل همانند کنترل MenuStrip می‌باشد، با این تفاوت که برای ایجاد منوی میانبر به کار می‌رود. منو میانبر، منویی است که با کلیک راست بر روی کنترل نمایش داده می‌شود و می‌توان گزینه‌های آن را اجرا نمود. خواص، رویدادها و متدهای این کنترل مانند کنترل MenuStrip است.

### ۱۲-۸-۱. کنترل PictureBox

این کنترل برای نمایش تصاویری از قبیل نمادهای گرافیکی، تصاویر بیت نگاشت<sup>۱</sup>، شبه فایل، آیکن‌ها و غیره به کار می‌رود. برخی از خواص و متدهای این کنترل در زیر آمده‌اند:

<sup>۱</sup> - Bitmap

🚩 **خاصیت Image:** تصویری را تعیین می‌کند که باید در کنترل PictureBox نمایش داده شود. به عنوان مثال، دستورات زیر را ببینید:

```
Image img1 = new Bitmap ("D:\\1.gif");
PictureBox1.Image = (Image) img1;
```

این دستورات، فایل 1.gif ریشه درایو D را در کنترل PictureBox نمایش می‌دهند.

🚩 **خاصیت SizeMode:** نحوه و اندازه نمایش تصویر را در PictureBox تعیین می‌کند و می‌تواند مقادیر زیر را بپذیرد:

۱. مقدار **Normal:** اندازه تصاویر را تغییر نمی‌دهد (اندازه واقعی آن را نمایش می‌دهد).

۲. مقدار **StretchImage:** تصویر را به اندازه PictureBox بزرگ یا کوچک می‌کند.

۳. مقدار **AutoSize:** کنترل PictureBox را به اندازه تصویر تغییر می‌دهد.

۴. مقدار **CenterImage:** تصویر را در وسط کنترل PictureBox نمایش می‌دهد.

• **متد Save:** برای ذخیره تصویر موجود را کنترل PictureBox بر روی حافظه جانبی به کار می‌رود. به عنوان مثال، دستور زیر را ببینید:

```
PictureBox1.Image.Save ("test.gif");
```

تصویر موجود در PictureBox1 را در فایل test.gif ذخیره می‌کند.

## ۹-۱. ساختارهای کنترلی

در برنامه‌های ساده، دستورات برنامه به صورت پشت سر هم (از اولین دستور به آخرین دستور) اجرا می‌شوند. گاهی نیاز است بعضی از دستورات چندین بار اجرا شوند، تحت شرایط خاصی اجرا شده یا اجرا نگردند. برای پیاده‌سازی چنین برنامه‌هایی از ساختارهای کنترلی استفاده می‌شود. این ساختار دو نوع‌اند که عبارتند از:

۱. ساختارهای تصمیم

۲. ساختارهای تکرار

### ۱-۹-۱. ساختارهای تصمیم

این ساختارها برای حالتی به کار می‌روند که بخواهید تحت شرایطی مجموعه‌ای از دستورات اجرا شوند یا برخی دیگر اجرا نگردند. ساختارهای تصمیم در C# عبارتند از: `if` و `switch`.

#### ساختار تصمیم if

این ساختار یک عبارت شرطی را ارزیابی می‌کند، در صورتی که نتیجه ارزیابی شرط دارای ارزش درست باشد، مجموعه‌ای از دستورات اجرا می‌شوند، وگرنه، مجموعه دیگری از دستورات اجرا خواهند شد. این ساختار به صورت زیر به کار می‌رود:

## ۲۳ آشنایی با زبان C#

```
if (عبارت شرطی)
{
    ; مجموعه دستورات ۱
}
else
{
    ; مجموعه دستورات ۲
}
```

در این ساختار، ابتدا عبارت شرطی داخل پرانتز ارزیابی می‌گردد، اگر نتیجه ارزیابی درست باشد، مجموعه دستورات ۱ اجرا می‌شوند، وگرنه مجموعه دستورات ۲ اجرا خواهند شد.

در این ساختار به نکات زیر توجه کنید:

✚ در هر یک از بخش‌های if و else اگر تعداد دستورات یکی باشد، بلاک‌های { و } را می‌توانید حذف کنید.

✚ در این ساختار می‌توان بخش else را حذف کرد. در این صورت، اگر نتیجه ارزیابی عبارت شرطی نادرست باشد، دستورات بعد از { if اجرا خواهند شد.

✚ در این ساختار، عبارت شرطی می‌تواند با عملگرهای منطقی از قبیل && یا ||، شرط‌های مختلف را ترکیب کرد.

### ساختار if تودرتو

گاهی ممکن است بخواهید شرط‌های متعددی را بررسی کنید. برای این منظور، می‌توانید از ساختار if تودرتو استفاده کنید. این ساختار به صورت زیر به کار می‌رود:

```
if (عبارت شرطی ۱)
{
    ; مجموعه دستورات ۱
}
else if (عبارت شرطی ۲)
{
    ; مجموعه دستورات ۲
}
:
else if (عبارت شرطی n)
{
    ; مجموعه دستورات n
}
else
{
    ; مجموعه دستورات n+1
}
```

در این ساختار، اگر نتیجه عبارت شرطی ۱، درست باشد، مجموعه دستورات ۱ اجرا می‌شوند و کنترل اجرای برنامه به بعد از { مربوط به else انتقال می‌یابد. وگرنه، اگر نتیجه عبارت شرطی ۲، درست باشد، مجموعه دستورات ۲ اجرا می‌شوند و دستور if خاتمه می‌یابد و این روند ادامه می‌یابد و اگر هیچ یک از عبارت‌های شرطی نتیجه درستی نداشته باشند، مجموعه دستورات n+1، اجرا خواهند شد.

### ساختار switch

ساختار تصمیم تودرتو، موجب کاهش خوانایی برنامه خواهد شد. برای رفع این مشکل، می‌توان از ساختار switch استفاده نمود. این ساختار به صورت زیر به کار می‌رود:

```
switch (عبارت) {
  case <مقدار۱> :
    دستورات ۱ ;
    break ;
  case <مقدار۲> :
    دستورات ۲ ;
    break ;
    :
  case <مقدار n> :
    دستورات n ;
    break ;
  default :
    دستورات n+1 ;
    break ;
}
```

در این ساختار، ابتدا عبارت داخل پراگم switch، ارزیابی می‌شود. اگر نتیجه ارزیابی این عبارت، برابر با مقدار ۱ باشد، دستورات ۱ اجرا خواهند شد و دستور break ساختار switch را خاتمه می‌دهد. اگر نتیجه ارزیابی عبارت برابر مقدار ۱ نباشد، با مقدار ۲ مقایسه می‌گردد، اگر برابر این مقدار باشد، دستورات ۲ اجرا می‌شوند و ساختار switch خاتمه می‌یابد و این روند ادامه می‌یابد. اگر نتیجه ارزیابی عبارت برابر هیچ یک از مقادیر ۱ تا n نباشد، دستورات n+1 اجرا خواهند شد.

در ساختار switch باید به نکات زیر دقت کنید:

- ✚ مقادیر موجود در case‌های دستور switch نمی‌توانند با هم مساوی باشند.
- ✚ اگر در یک case دستور break ذکر نشود، این مقدار case با مقدار case بعدی (||) می‌گردد.
- ✚ دستور switch فقط مساوی بودن را بررسی می‌کند. ولی، ساختار if هر یک از شرط‌های منطقی (عملگرهای منطقی) را بررسی می‌نماید.

## ۱۰-۱. ساختارهای تکرار

برای انجام کارهای تکراری در برنامه از ساختارهای تکرار استفاده می‌شود. در C# حلقه‌های تکرار متعددی وجود دارند که برخی از آنها عبارتند از: `for`، `while`، `do while` و `foreach`.

### ساختار تکرار `for`

این ساختار برای حالتی به کار می‌رود که تعداد تکرار از قبل مشخص باشد. این ساختار به صورت زیر به کار می‌رود:

```
{ (گام حرکت ; شرط حلقه ; مقدار اولیه اندیس حلقه) for
```

```
; مجموعه دستورات بدنه حلقه
```

```
}
```

در این ساختار، ابتدا مقدار اولیه در اندیس حلقه قرار می‌گیرد. سپس شرط حلقه تست می‌گردد، اگر شرط دارای ارزش درستی باشد، مجموعه دستورات بدنه حلقه اجرا می‌گردند و گام حرکت به اندیس حلقه اضافه می‌شود. در ادامه شرط حلقه تست می‌گردد و این روند ادامه می‌یابد. تا زمانی که شرط حلقه نقض نگردد، حلقه ادامه می‌یابد. به عنوان مثال، دستورات زیر را ببینید.

```
int sum = 0;
for (i = 1 ; i <= 100 ; i++)
{
    sum += i;
    listBox1.Items.Add (i.ToString());
}
textBox1.Text = sum.ToString();
```

این دستورات، اعداد ۱ تا ۱۰۰ را به `listBox1` اضافه کرده مجموع آنها را در `textBox1` نمایش می‌دهند.

### ساختار تکرار `while`

این ساختار برای زمانی به کار می‌رود که تعداد تکرار از قبل مشخص نباشد و به صورت زیر استفاده می‌شود:

```
while (شرط)
```

```
{
```

```
; مجموعه دستورات بدنه حلقه
```

```
}
```

در این ساختار، ابتدا شرط داخل پرانتز ارزیابی می‌شود. اگر شرط درست باشد، مجموعه دستورات بدنه حلقه اجرا می‌شوند و شرط مجدداً ارزیابی می‌گردد و این روند تا زمانی که شرط ارزش درستی داشته باشد، ادامه می‌یابد و به محض اینکه شرط نقض (نادرست) شود، حلقه خاتمه می‌یابد. چنانچه مجموعه دستورات بدنه حلقه، یک دستور باشد، می‌توانید `{` و `}` را حذف نمایید.

**ساختار تکرار do while**

این ساختار همانند ساختار while است. با این تفاوت که دستورات بدنه حلقه حداقل یک بار اجرا می‌گردند. چون، شرط در انتهای حلقه ارزیابی (تست) می‌شود. این ساختار به صورت زیر به کار می‌رود:

```
do
{
    ; مجموعه دستورات بدنه حلقه
}
while (شرط);
```

**دستور break**

این دستور موجب خروج از حلقه تکرار یا switch می‌شود و به صورت زیر به کار می‌رود:

```
break;
```

**دستور continue**

این دستور، کنترل اجرای برنامه را به ابتدای حلقه تکرار برمی‌گرداند و به صورت زیر به کار می‌رود:

```
continue;
```

به عنوان مثال، دستورات زیر را ببینید:

```
listBox1.Items.Clear();
for (int i = 1; i <= 10; i++)
{
    if (i < 5) continue;
    listBox1.Items.Add (i.ToString());
}
```

این دستورات، مقادیر ۵ تا ۱۰ را به listBox1 اضافه می‌کنند. زیرا، اگر i کوچک‌تر از ۵ باشد، کنترل اجرای برنامه به ابتدای حلقه برمی‌گردد.

**۱۱-۱. مدیریت صفحه کلید**

کنترل‌ها علاوه بر این که به رویدادهای ماوس پاسخ می‌دهند، می‌توانند به رویدادهای صفحه کلید از قبیل KeyPress، KeyDown و KeyUp پاسخ دهند. رویداد KeyPress قبل از رویداد KeyDown و رویداد KeyDown قبل از رویداد KeyUp اتفاق می‌افتد. رویداد KeyPress، زمانی اتفاق می‌افتد که کاربر کلیدی (به جز کلیدهای Tab، مکان‌نما و کلیدهایی که کداسکی آنها بین ۰ تا ۳۱ است) را فشار دهد. این رویداد دو پارامتر دارد که عبارتند از:

**Sender:** از نوع Objcet است و شی‌ای را تعیین می‌کند که این رویداد بر روی آن رخ داده است.

**e:** از نوع KeyEventArgs دارای ساختاری می‌باشد که اطلاعات کلید فشرده شده از قبیل KeyChar (مقدار کارکتر فشرده شده) و Handled (گرداننده رویداد KeyPress) را نگهداری می‌کند.

➡ **رویداد KeyDown:** وقتی کاربری کلیدی را فشار می‌دهد، این رویداد اتفاق می‌افتد (این رویداد به کلیدهای مکان‌نما، Tab و کارکترهایی که کد آنها بین ۰ تا ۳۱ باشد، نیز پاسخ می‌دهد). این رویداد همچنین مانند رویداد KeyPress دارای دو پارامتر sender و e است. ساختار پارامتر e در جدول ۱۳-۱ آمده است.

➡ **رویداد KeyUp:** زمانی رخ می‌دهد که کلید فشرده شده را رها کنید. پارامترهای این رویداد مانند رویداد KeyDown است.

جدول ۱۳-۱ ساختار پارامتر e مربوط به رویدادهای KeyUp و KeyDown.	
خاصیت	هدف
Alt	آیا کلید Alt فشرده شده است یا خیر؟
Handled	گرداننده رویداد KeyDown و KeyUp را تعیین می‌کند.
Ctrl	آیا کلید Ctrl فشرده شده است یا خیر؟
KeyCode	کداسکی کارکتر فشرده شده را مشخص می‌کند.
KeyData	داده مربوط به کارکتر فشرده شده را تعیین می‌کند.
KeyValue	کارکتر فشرده شده را مشخص می‌نماید.
Shift	آیا کلید Shift فشرده شده است یا خیر؟

## ۱۲-۱. آرایه‌ها

تاکنون متغیرهایی که در برنامه‌ها استفاده کردیم، یک مقدار را ذخیره می‌کردند. گاهی نیاز است چندین مقدار از یک نوع داده را ذخیره کنید. برای این منظور باید متغیرهای **اندیس‌دار** یا **آرایه** را تعریف نمایید. آرایه با توجه به تعداد اندیس‌هایشان به انواع مختلف تقسیم می‌شوند. آرایه‌ایی با یک اندیس را **آرایه یک بعدی**، آرایه دارای دو اندیس را **آرایه دو بعدی** و آرایه‌ایی که n اندیس داشته باشد، **آرایه n بعدی** نام دارد. برای استفاده از آرایه دو کار باید انجام شود:

۱. اعلان آرایه

۲. تخصیص فضا به آرایه

برای اعلان آرایه یک بعدی به صورت زیر عمل می‌شود:

نام آرایه [ ] نوع آرایه

این دستور، شکل زیر را ایجاد می‌کند:

نام آرایه [ ] نوع آرایه

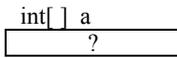
برای تخصیص فضا به آرایه باید نمونه‌هایی از آن را ایجاد کنید. این کار با دستور new و به صورت زیر انجام می‌شود:

؛ [تعداد عناصر آرایه] نوع آرایه new = نام آرایه

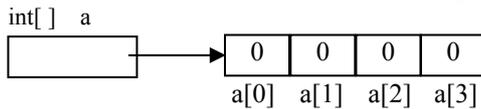
اکنون دستورات زیر را در نظر بگیرید:

```
int[] a;  
a = new int[4];
```

دستور اول، آرایه `a` را از نوع `int` تعریف می‌کند (شکل زیر):



دستور دوم، آرایه‌ای با ۴ عنصر تعریف کرده آدرس شروع آن را به `a` تخصیص می‌دهد (مانند شکل زیر):



در هنگام استفاده از آرایه به نکات زیر توجه داشته باشید:

۱. اندیس آرایه از صفر (+) شروع می‌شود.
۲. تعداد عناصر آرایه را می‌توان در زمان اجرا تعیین کرد. یعنی، آرایه‌ای به صورت پویا تعریف نمود. به عنوان مثال، دستورات زیر را ببینید:

```
int count = Convert.ToInt32 (textBox1.Text);
int[] a = new int [count];
```

این دستورات، آرایه‌ای به نام `a` تعریف می‌کنند که تعداد عناصر آن از طریق کنترل `textBox1` تعیین می‌گردد.

۳. همان‌طور که می‌دانید، آرایه متغیری اندیس‌دار است. بنابراین، برای بازیابی و مقداردهی (دستیابی) به عناصر آن از اندیس به صورت زیر استفاده می‌شود:

#### اندیس آرایه نام آرایه

اکنون دستورات زیر را ببینید:

```
Int[] a = new int[3];
a [0] = 10;
a [1] = 12;
a [2] = a [1] + a [0];
```

دستور اول، آرایه‌ای به نام `a` با ۳ عنصر تعریف می‌کند. دستور دوم، اولین عنصر آرایه (اندیس صفر) را برابر ۱۰ قرار می‌دهد. دستور سوم، دومین عنصر آرایه را برابر ۱۲ قرار داده و چهارمین دستور، مجموع اولین عنصر و دومین عنصر (۱۰+۱۲) را در سومین عنصر آرایه قرار می‌دهد.

۴. برای تعیین تعداد عناصر آرایه از خاصیت `Length` استفاده می‌شود. دستورات زیر را ببینید:

```
Int[] a = new int [5];
Label1.Text = a.Lenght;
```

دستور اول، آرایه‌ای با ۵ عنصر تعریف می‌کند. دستور دوم، ۵ را در `label1` نمایش می‌دهد (تعداد عناصر آرایه `a` را با خاصیت `Length` تعیین می‌کند).

۵. در هنگام تعریف آرایه، عناصر آرایه از نوع عددی با صفر مقداردهی می‌شوند. عناصر آرایه از نوع `boolean` با `false` مقداردهی خواهند شد و عناصر آرایه از نوع ارجاع با `تهی`<sup>۱</sup> مقداردهی می‌گردند. ولی می‌توان هنگام تعریف آرایه به آنها مقدار داد. برای این منظور باید به صورت زیر عمل نمایید:

<sup>۱</sup>- Null