آموزش گام به گام

LINQ با C#

(مرجع کامل)

تالیف

مهندس رمضان عباس‌نژادورزی

فن‌آوری نوین

سرشناسه	: عباس‌نژاد ورزی، رمضان، ۱۳۴۸-
عنوان و نام پدیدآور	: آموزش گام به گام LINQ با #C (مرجع کامل)/تالیف رمضان عباس‌نژاد ورزی
مشخصات نشر	: بابل:فن‌آوری نوین، ۱۳۸۹ .
مشخصات ظاهری	: ۲۴۸ ص.:مصور، جدول.
شابک	: ۶۵۰۰۰ریال:978-600-91413-6-4
وضعیت فهرست نویسی	: فیپا
یادداشت کتابنامه	: ص.۲۴۸.
موضوع	: ال. آی. ان. کیو مایکروسافت
موضوع	: اس. کیو. ال (زبان برنامه نویسی کامپیوتر)
موضوع	: زبان‌های پرس و جوی کامپیوتری
موضوع	: سی شارپ (زبان برنامه‌نویسی کامپیوتر)
رده بندی کنگره	: QA۷۶/۷۳/الف ۷۳ ع ۲ ۱۳۸۹
رده بندی دیویی	۰۰۵/۲۶۸
شماره کتابشناسی ملی	۲۰۰۹۰۰۲

فن‌آوری نوین

www.fanavarinovin.net

بابل، کدپستی ۷۳۴۴۸-۴۷۱۶۷ تلفن: ۲۲۵۶۶۸۷-۰۱۱۱

آموزش گام به گام LINQ با #C (مرجع کامل)

تألیف: مهندس رمضان عباس‌نژادورزی

نوبت چاپ: چاپ اول

سال چاپ: بهار ۱۳۸۹

شمارگان: ۱۰۰۰ جلد

قیمت: ۶۵۰۰ تومان

نام چاپخانه و صحافی: فرنگاررنگ

شابک: ۴- ۶- ۹۱۴۱۳ - ۶۰۰ - ۹۷۸

نشانی ناشر: بابل چهارراه نواب، کاظم‌بیگی، جنب حسینیه منصور کاظم‌بیگی، طبقه همکف

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

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

فهرست مطالب

فصل اول : آشنایی با زبان #C

۱-۱. زبان #C ۹

۱-۲. فضای نام.....۹

۱-۳. انواع داده‌ها ۱۰

۱-۴. متغیرها ۱۱

۱-٤-۱. نامگذاری متغیرها ۱۱

۱-٤-۲. اعلان متغیرها ۱۱

۱-٤-۳. مقدار دادن به متغیرها ۱۲

۱-۵. ثوابت ۱۲

۱-۶. عملگرها ۱۳

۱-۷. فرم برنامه ۱۴

۱-۷-۱. خواص فرم.....۱٤

۱-۷-۲. رویدادهای فرم ۱۵

۱-۷-۳. متدهای فرم ۱۷

۱-۸. کنترل‌ها ۱۷

۱-۸-۱. کنترل Label ۱۸

۱-۸-۲. کنترل TextBox ۱۸

۱-۸-۳. کنترل Button.....۱۸

۱-۸-٤. کنترل ListBox ۱۸

۱-۸-۵. کنترل ComboBox.....۲۰

۱-۸-٦. کنترل CheckBox ۲۱

۱-۸-۷. کنترل CheckedListBox.....۲۱

۱-۸-۸. کنترل RadioButton ۲۱

۱-۸-۹. کنترل GroupBox ۲۲

۱-۸-۱۰. کنترل MenuStrip.....۲۲

۱-۸-۱۱. کنترل ContextMenuStrip ۲۲

۱-۸-۱۲. کنترل PictureBox ۲۳

۱-۹. ساختارهای کنترلی ۲۳

۱-۹-۱. ساختارهای تصمیم ۲۳

۱-۱۰. ساختارهای تکرار.....۲۶

۱-۱۱. مدیریت صفحه کلید ۲۷

۱-۱۲. آرایه‌ها ۲۸

۱-۱۳. کلاس‌ها و اشیاء ۳۶

۱-۱۳-۱. تعریف کلاس ۳۷

۱-۱۳-۲. نمونه‌سازی کلاس.....۳۷

۱-۱۳-۳. اعضای کلاس.....۳۷

فصل دوم: مبانی بانک اطلاعات SQL Server

۲-۱. تعریف سیستم مدیریت بانک اطلاعات.....۴۶

۲-۱-۱. دلایل استفاده از بانک اطلاعات ٤۷

۲-۱-۲. طراحی بانک اطلاعاتی ٤۸

۲-۱-۳. نرمال‌سازی داده‌ها.....٤۹

۲-۲. بانک اطلاعات SQL Server ۴۹

۲-۳. معرفی بانک اطلاعاتی نمونه.....۵۰

۲-۴. ورود به بانک اطلاعاتی SQL Server ۵۲

۲-۵. تایپ و اجرای دستورات SQL.....۵۳

۲-۶. ایجاد بانک اطلاعاتی ۵۴

۲-٦-۱. تغییر خواص اطلاعاتی موجود ۵۵

۲-٦-۲. حذف بانک اطلاعاتی موجود ٥٦

۷-۲. اشیای بانک اطلاعات ۵۶
۱-۷-۲. ایجاد جدول با دستور SQL .۵۷
۲-۷-۲. تغییر ساختار جدول با
دستور SQL ۶۰
۳-۷-۲. حذف جدول با دستور SQL ۶۱
۸-۲. دستورات SQL برای ورود، ویرایش
و حذف داده‌ها ۶۱
۱-۸-۲. دستور INSERT ۶۲
۲-۸-۲. ویرایش رکوردهای جدول ...۶۳
۳-۸-۲. حذف رکوردهای جدول ۶۴
۹-۲. دستور SELECT.....۶۴
فصل سوم : مفاهیم اولیه و عملگرهای LINQ
۱-۳. مزایای LINQ.....۶۸
۲-۳. معماری LINQ.....۶۸
۳ - ۳ . اسمبلی‌های میانی هسته LINQ ۷۰
۴-۳ . LINQ چطور پیاده‌سازی می‌شود؟..۷۰
۵-۳ . به کارگیری عملگرهای
پرس‌وجوی استاندارد.....۷۰
۶-۳ . تعریف متغیر برای نگهداری
نتیجه پرس‌وجوی LINQ.....۷۱
۷-۳ . ایجاد پرس‌وجو از منابع داده.....۷۱
۱-۷-۳ . ساختار پرس‌وجوی
LINQ.....۷۱
۲-۷-۳ . استفاده از متدهای پرس‌وجو و
عبارات Lambda.....۷۲
۸-۳ . عملگرهای استاندارد پرس‌وجو.....۷۳
فصل چهارم : مدل شیء LINQ to SQL
۱ – ۴. کلاس Data Context.....۱۱۰
۱ – ۱ –٤. نوع قوی DataContext ..۱۱۱
۲ – ۱ –٤. نگاشت جداول ۱۱۲
۳ – ۱ – ٤. نگاشت ستون‌ها ۱۱۲
٤ – ۱ – ٤. نگاشت رابطه‌ها ۱۱٤
۲ – ۴. دستکاری داده‌ها.....۱۱۴
۱ – ۲ – ٤. اضافه کردن رکورد ۱۱۵
۲ – ۲ – ٤. حذف رکورد ۱۱۵
۳ – ۲ –٤ . به روز رسانی رکورد ۱۱۵
۳ – ۴. کنترل DataGridView ۱۱۶
فصل پنجم : قطعه LINQ to Dataset
۱-۵. دستیابی به بانک اطلاعات
با ADO.NET.....۱۲۶
۱-۱-۵. کلاس Connection ۱۲۷
۲-۱-۵. کلاس Command ۱۲۷
۳-۱-۵. کلاس Dataset ۱۲۹
٤-۱-۵. کلاس DataAdapter ۱۳۰
۵-۱-۵. کلاس DataTable ۱۳۱
٦-۱-۵. کلاس DataColumn ۱۳۳
۷-۱-۵. کلاس DataRow ۱۳۳
۸-۱-۵. کلاس DataReader ۱۳٤
۲-۵. مروری بر LINQ to Dataset.....۱۳۵
۳-۵. نیازمندی‌های پروژه LINQ to
Dataset.....۱۳۵
فصل ششم : رویه‌های ذخیره شده و توابع
۱-۶. متغیرها.....۱۴۴
۲-۶. توضیحات ۱۴۵
۳-۶. ساختارهای تصمیم.....۱۴۵
۱-۳-٦. دستور IF … Else ۱٤۵
۲-۳-٦. دستور IF تو در تو.....۱٤٦
۳-۳-٦. دستور CASE ۱٤۷
۴-۶. رویه‌های ذخیره شده.....۱۴۷
۱-٤-٦. ایجاد رویه‌های ذخیره شده .۱٤۸
۲-٤-٦. اجرای رویه ذخیره شده ۱۵۰
۳-٤-٦. تغییر رویه ذخیره شده ۱۵۱

٤-٤-٦. حذف رویه‌های ذخیره شده ١٥٢
٥-٦. توابع ١٥٢
١-٥-٦. ایجاد توابع ١٥٢
٢-٦-٦. تغییر تابع ١٥٣
٣-٥-٦. حذف تابع ١٥٤
٦-٦. نگاشت توابع و رویه‌ها
ذخیره شده ١٥٤
١-٦-٦. نگاشت یک شیء در LINQ به
یک رویه ذخیره شده با تابع در
بانک اطلاعات ١٥٥
٢-٦-٦. معرفی پارامترها و رویه ذخیره
شده LINQ ١٥٥
٣-٦-٦. تعریف دستوراتی که رویه
ذخیره شده را اجرا می‌کنند.....١٥٥
٤-٦-٦. برگشت مقادیر.....١٥٦
٥-٦-٦. نگاشت رویه‌های ذخیره شده
برای نتایج چندگانه.....١٥٧
٦-٦-٦. نگاشت و فراخوانی توابع
تعریف شده توسط کاربر ١٥٨
فصل هفتم : LINQ وXML
١-٧. ساختار داده‌های XML.....١٦٦
٢-٧. صفات.....١٦٧
٣-٧. بخش‌های تعریف XML.....١٦٧
٤-٧. کلاس‌های LINQ to XML.....١٦٨
١-٤-٧. کلاس XElement.....١٦٨
٢-٤-٧. کلاس XAttribute.....١٧٣
٣-٤-٧. کلاس XDocument.....١٧٣
٥-٧. مفاهیم برنامه‌نویسی
LINQ to XML.....١٧٥
١-٥-٧. بار کردن سند XML
موجود.....١٧٦
٢-٥-٧. ذخیره کردن XML از طریق
LINQ to XML.....١٧٨
٣-٥-٧. دستکاری XML.....١٧٨
٤-٥-٧. اضافه کردن عناصر.....١٧٨
٥-٥-٧. به روز رسانی عناصر.....١٧٩
٦-٥-٧. حذف عناصر.....١٨١
٦-٧. کار کردن با صفات.....١٨٣
١-٦-٧. اضافه کردن صفت ١٨٣
٢-٦-٧. بازیابی صفت ١٨٤
٣-٦-٧. حذف صفت ١٨٥
٧-٧. رویدادهای LINQ to XML.....١٩٥
٨-٧. رویدادهای SQL to XML.....١٩٨
فصل هشتم : تراکنش‌ها و برخوردها
١-٨. ارجاع به تراکنش ٢٠٤
٢-٨. همزمانی خوش‌بینانه ٢٠٥
١-٢-٨. کلاس ChangeConflictException
.....٢٠٧
٢-٢-٨. متد Reslove.....٢٠٧
٣-٨. روش‌های پیاده‌سازی تراکنش.....٢١٣
١-٣-٨. خاصیت Transaction شیء
DataContext ٢١٣
٢-٣-٨. کلاس
TransactionScope ٢١٤
٣-٣-٨. کلاس
CommittableTransaction ٢١٤
فصل نهم: گزارش گیری با Microsoft Report
١-٩. امکانات نرم‌افزار Microsoft Report
.....٢٢٢
٢-٩. مراحل طراحی گزارش ٢٢٢
٣-٩. ایجاد گزارش با ویزارد.....٢٢٣

۴-۹. اضافه کردن گزارش جدید ۲۲۹
۱-۴-۹. بخش‌های گزارش.....۲۳۰
۲-۴-۹. اضافه کردن فیلد متن
به گزارش ۲۳۱
۳-۴-۹. اضافه کردن خط ۲۳۱
۴-۴-۹. اضافه کردن مستطیل ۲۳۲
۵-۴-۹. اضافه کردن تصویر به گزارش ۲۳۲
۵-۹. نمایش پنجره DataSources.....۲۳۴
۶-۹. ارسال پارامتر به گزارش ۲۳۴
پیوست الف: امکانات طراحی LINQ.. ۲٤۰
پیوست ب: ابزار SQLMetal..... ۲٤٥
منابع :..... ۲٤۸

مقدمه

امروزه کد و داده دو عنصر اصلی نرم‌افزار هستند. یکی از بخش‌های بسیار مهم نرم‌افزار، نوشتن کدهایی برای دستیابی به داده است. از آنجائی که منابع داده‌ای مختلف از قبیل بانک‌های اطلاعاتی متعدد، فایل‌های XML و آرایه‌ها وجود دارند، یکی از مهم‌ترین مشکلات برنامه‌نویسان، این است که هر یک از منابع داده از یک مدل خاص استفاده می‌کند. بنابراین، برنامه‌نویس باید داده‌های خود را از مدلی به مدل دیگر تبدیل کند تا بتواند از آن‌ها استفاده نماید. این تبدیلات نه تنها زمان‌بر است، بلکه برنامه‌نویس باید کار کردن با ابزارهای مختلف را یاد بگیرد. به همین دلیل، شرکت مایکروسافت تکنولوژی LINQ (Language Integrated Query) را در دات‌نت اضافه نموده است. به زبان ساده می‌توان گفت LINQ، واسطی است که بین زبان برنامه‌نویسی و منبع داده قرار می‌گیرد و ارتباط بین آن‌ها را برقرار می‌کند. یعنی، نیازی نیست با تغییر منبع داده، برنامه نوشته شده در زبان برنامه‌نویسی تغییر یابد.

در این کتاب، مقدمه‌ای از #C، بانک اطلاعاتی SQL Server، عملگرهای استاندارد LINQ، قطعات LINQ to SQL و LINQ to Dataset آمده است. همچنین، این کتاب به موضوعاتی از قبیل چگونگی نگاشت رویه‌های ذخیره شده و توابع به LINQ، ارتباط بین LINQ و XML، چگونگی پیاده‌سازی تراکنش در LINQ و ارتباط بین Microsoft Report و LINQ پرداخته است. در پیوست امکانات طراحی کلاس در LINQ و ابزار SqlMetal بیان گردیده است.

در پایان امیدوارم این اثر نیز مورد توجه اساتید و دانشجویان عزیز قرار گیرد.

کتاب حاوی برنامه‌هایی است که کد آن‌ها را می‌توانید به صورت رایگان از سایت انتشارات فن‌آوری نوین به آدرس www.fanavarinovin.net بگیرید.

عباس نژادورزی
fanavarienovin@yahoo.com

دیگر آثار مولف			
نام کتاب	انتشارات	نام کتاب	انتشارات
حل مسائل C (مرجع کامل)	فن آوری نوین	آموزش گام به گام Crystal Report	علوم رایانه
حل مسائل C++ (مرجع کامل)	فن آوری نوین	آشنایی با شبکه GSM	علوم رایانه
آموزش گام به گام برنامه نویسی بانک اطلاعات با C# (مرجع کامل)	فن آوری نوین	آموزش گام به گام سیستم عامل لینوکس	علوم رایانه
حل مسائل C# (مرجع کامل)	فن آوری نوین	خود آموز اکسس	علوم رایانه
حل مسائل پاسکال (مرجع کامل)	فن آوری نوین	آموزش گام به گام Word	علوم رایانه
آموزش گام به گام برنامه نویسی بانک اطلاعات با ویژوال بیسیک‌نت (مرجع کامل)	فن آوری نوین	آموزش گام به گام اکسل	علوم رایانه
آموزش گام به گام برنامه ویژوال بیسیک	علوم رایانه	ICDL مهارت ۱: مفاهیم پایه اطلاعات	علوم رایانه
آموزش گام به گام برنامه ویژوال بیسیک‌نت	علوم رایانه	ICDL مهارت۲: به کارگیری کامپیوتر و مدیریت	علوم رایانه
برنامه نویسی به زبان اسمبلی	علوم رایانه	ICDL مهارت۳: واژه پردازی به کمک کامپیوتر	علوم رایانه
برنامه نویسی با دلفی	علوم رایانه	ICDL مهارت٤: صفحات گسترده	علوم رایانه
آموزش گام به گام دلفی‌نت	علوم رایانه	ICDL مهارت٥: پایگاه داده	علوم رایانه
آموزش گام به گام C#.NET	علوم رایانه	ICDL مهارت٦: ارائه مطلب	علوم رایانه
آموزش گام به گام VisualC++.NET	علوم رایانه	ICDL مهارت۷: اطلاعات و ارتباطات	علوم رایانه
آموزش گام به گام برنامه نویسی با ویژوال C++	علوم رایانه	آموزش گام به گام FLASH MX	علوم رایانه
آموزش گام به گام J#.NET	علوم رایانه	درس و کنکور برنامه‌نویسی به زبان C	علوم رایانه
آموزش گام به گام SQL Server	علوم رایانه	برنامه سازی سیستم	علوم رایانه
آموزش گام به گام SQL	علوم رایانه	پرسش‌های چهار گزینه‌ای پاسکال	علوم رایانه
رهیافت و پرسش‌های چهار گزینه‌ای C	علوم رایانه	آموزش گام به گام VC++	علوم رایانه
رهیافت و پرسش‌های چهار گزینه‌ای C++	علوم رایانه	کارور رایانه ۱	علوم رایانه
تست و پرسش‌های چهار گزینه‌ای C	علوم رایانه	کارور رایانه ۲	علوم رایانه
مبانی فناوری اطلاعات	علوم رایانه		

فصل ۱ 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 استودیونت ارائه شده است. زبان #C از فناوری شیء[1] و مفهوم شی‌گرایی[2] 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 شی‌گرایی استفاده می‌نماید، امکاناتی در این زبان اضافه شده است که می‌توانید اشیای دنیای واقعی را مدل‌سازی کنید. برای این که #C شی‌گرایی را پیاده‌سازی کند از مفهوم **کلاس**[3] 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 استودیونت وجود دارند.

1- Object 2- Object Oriented 3- Class

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.

کلاس‌هایی که در کتابخانه FCL وجود دارند، در **فضاهای نام**[1] 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 کارمی‌روند، 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 می‌روند.
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 بی‌آموزیم. 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 و یا کلاس‌هایی هستند که از قبل تعریف شده‌اند. در ادامه پیاده‌سازی کلاس را می‌آموزیم.

1- Namespaces

جدول۲-۱ 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	٤	مقادیر صحیح بین $-۲^{۳۱}$ تا $(۲^{۳۱}-۱)$
uint	Uint32	٤	مقادیر صحیح بدون علامت ۰ تا $۲^{۳۲}$
float	Single	٤	اعداد اعشاری $\pm ۱/۵\times ۱۰^{-۴۵}$ تا $\pm ۳/۴\times ۱۰^{۳۸}$ با ۷ رقم اعشار
double	Double	۸	اعداد اعشاری $\pm ۵/۰\times ۱۰^{-۳۲۴}$ تا $\pm ۱/۷\times ۱۰^{۳۰۸}$ با ۱۵ تا ۱٦ رقم بعد از اعشار
decimal	Decimal	۸	نقطه اعشار ثابت تا ۲۸ رقم
long	Int64	۸	مقادیر صحیح $-۲^{۶۳}$ تا $(۲^{۶۳}-۱)$
ulong	Uint64	۸	مقادیر ۰ تا $(۲^{۶۴}-۱)$

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 دارند.

برای استفاده از متغیرها باید نام، نوع و مقدار آنها را تعیین کرد.

۱-۴-۱. 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 hioh!book نمی‌توانند نام متغیرها باشند. زیرا، اولین نام با رقم 1 شروع شد و در دومین نام از کارکتر ! استفاده گردید که کارکتر مجاز نمی‌باشد.

۲-۴-۱. 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:

سطح دستیابی نوع داده نام متغیر ;

سطح دستیابی، تعیین می‌کند که در چه محدوده‌ای می‌توانید به متغیرها دستیابی داشته باشید و مهم‌ترین آنها public و private هستند. public موجب می‌شود تا بتوانید متغیر در کل کلاس دستیابی داشته باشید و private موجب می‌شود تا بتوان به متغیر 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 ۱۰ 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 نوع داده نام = مقدار ثابت ;

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 عملوندها انجام می‌دهند. این عملگرها در جدول ۳-۱ آمده‌اند.
- **عملگرهای رابطه‌ای**، عملگرهایی هستند که دو عملوند را با یکدیگر مقایسه می‌کنند (جدول ٤-۱ را ببینید).
- **عملگرهای منطقی**، عملگرهایی هستند که بر روی عبارات منطقی (True یا False) عمل می‌کنند (جدول ٥-۱ را مشاهده کنید).
- **عملگرهای ترکیبی**، از ترکیب عملگرهای محاسباتی و علامت = ایجاد می‌شوند (جدول٦-۱).
- **عملگرهای بیتی**، عملگرهایی هستند که برای تست کردن، مقداردهی یا شیفت دادن و سایر اعمال بر روی عملوندها به کار می‌روند. عملگرهای بیتی در جدول ۷-۱ آمده‌اند.

جدول ۳-۱ عملگرهای محاسباتی.

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--	4 , 9
++	افزایش	10	5	x++ , ++y	6 , 11

جدول ۴-۱ عملگرهای رابطه‌ای.

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	True
= =	تساوی	10	17	x = = y	False
!=	نامساوی	10	17	x != y	True

جدول ۵-۱ عملگرهای منطقی.

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

جدول ۶-۱ عملگرهای ترکیبی.

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

جدول ۷-۱ عملگرهای بیتی.

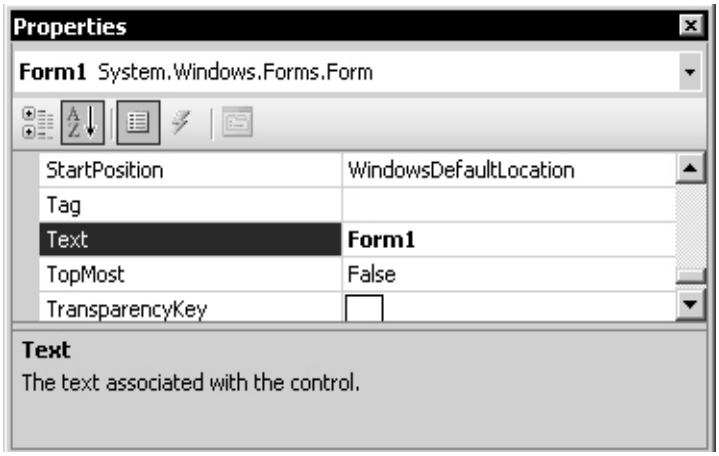
عملگر	نام	x	y	مثال	نتیجه
&	و	10	2	x & y	2
|	یا	10	3	x | y	11
^	یا انحصاری	10	3	x ^ y	9
~	نقیض	126	3	~ x	1
>>	شیفت به راست	56	3	x >> y	7
<<	شیفت به چپ	25	2	x << y	100

۷-۱. فرم برنامه

فرم برنامه، مکانی است که کنترل‌های برنامه در آن قرار می‌گیرند. هر برنامه 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 ۸-۱ آمده است. برای نمایش خواص فرم، بر روی آن کلیک کنید و سپس کلید F4 را بزنید تا خواص فرم را ببینید (شکل زیر):



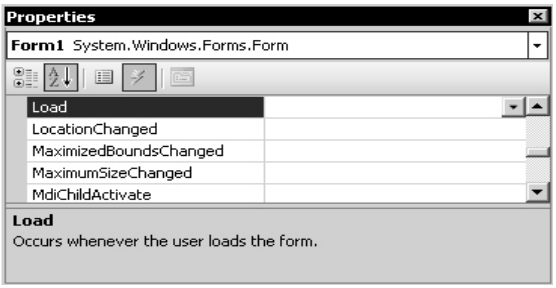
جدول ۸-۱ خواص مهم فرم.

خاصیت	هدف
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	تعیین‌می‌کند آیا فرم فعال است یا خیر. اگر فرم غیرفعال گردد، به‌هیچ‌رویدادی‌پاسخ نمی‌دهد.
Font	فونت فرم را تعیین می‌کند.
ForeColor	رنگ‌نوشته‌های‌روی‌فرم 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	جهت‌نمایش‌اطلاعات‌را تعیین‌می‌کند(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 ۹-۱ 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 می‌شود، رخ می‌دهد.
Click	وقتی رخ می‌دهد که فرم کلیک گردد.
FormClosed	وقتی رخ می‌دهد که فرم بسته شود.
FormClosing	وقتی رخ می‌دهد که فرم در حال بسته شدن باشد. این رویداد قبل از رویداد FormClosed رخ می‌دهد.
Deactivate	وقتی رخ می‌دهد که فرم غیرفعال گردد.
DoubleClick	وقتی که فرم کلیک مضاعف شود، رخ می‌دهد.
Enter	وقتی مکان‌نما وارد فرم می‌شود، رخ می‌دهد.
KeyDown	وقتی کلیدی فشرده شده پایین می‌رود، رخ می‌دهد.
KeyPress	وقتی کلیدی فشرده می‌شود، رخ می‌دهد. این رویداد قبل از رویداد KeyDown اتفاق می‌افتد.
KeyUP	وقتی کلید فشرده شده رها می‌گردد، رخ می‌دهد.
Leave	وقتی مکان‌نما فرم را ترک می‌کند رخ می‌دهد.
Load	وقتی فرمی باز می‌شود، رخ می‌دهد (قبل از نمایش فرم).
MouseDown	وقتی کلید ماوس فشرده شود، رخ می‌دهد.
MouseEnter	وقتی مکان‌نمای ماوس وارد فرم شود، رخ می‌دهد.
MouseLeave	وقتی مکان‌نمای ماوس فرم را ترک می‌کند، رخ می‌دهد.
MouseUP	وقتی کلید فشرده شده ماوس رها می‌شود، رخ می‌دهد.
Move	وقتی فرم شروع به حرکت می‌کند، رخ می‌دهد.
Resize	وقتی اندازه فرم تغییر می‌یابد، رخ می‌دهد.
TextChanged	وقتی رخ می‌دهد که متن فرم (کنترل) یا خاصیت Text تغییر کند.
Shown	وقتی که فرم نمایش داده شود، رخ می‌دهد.
SizeChanged	وقتی رخ می‌دهد که مقدار خاصیت Size تغییر می‌یابد.
MouseMove	وقتی رخ می‌دهد که ماوس روی فرم حرکت می‌کند.

۳-۷-۱. متدهای فرم

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

جدول ۱۰-۱ متدهای مهم فرم.

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

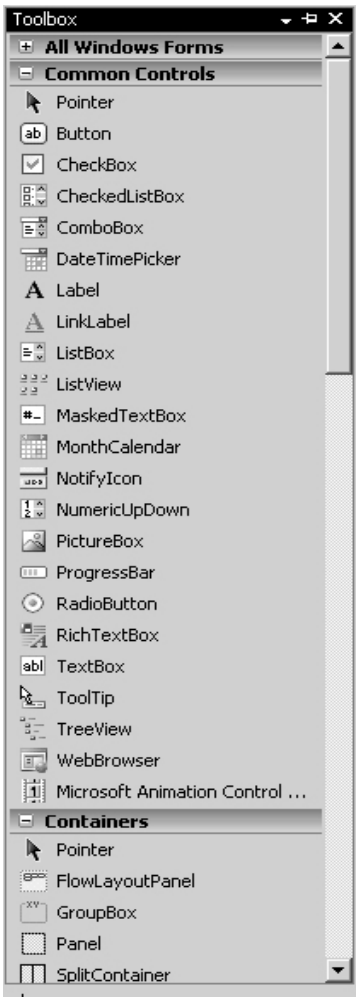
۸-۱. کنترل‌ها

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 اجزا 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.[۱]

این کنترل‌ها 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):

۱- برای کسب اطلاعات بیشتر در مورد #C به آموزش گام به گام #C از انتشارات علوم رایانه، تألیف عین‌الله جعفرنژادقمی و رمضان عباس‌نژادورزی مراجعه کنید.

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.



شکل ۱-۱ پنجره ابزار ویژوال استودیونت

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.

۴-۸-۱. 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 ۱۲-۱ آمده‌اند.

جدول ۱۱-۱ خواص، رویدادها و متدهای کنترل TextBox.

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

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 ۱۲-۱ خواص، متدها و رویدادهای کنترل ListBox.

خاصیت	هدف
Items	گزینه‌هایی را تعیین می‌کند تا بر روی کنترل نمایش داده شوند.
MultiColumn	تعیین می‌کند آیا کنترل ListBox می‌تواند چند ستونی باشد یا خیر.
SelectionMode	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 Shift و Ctrl می‌توان چند گزینه را انتخاب کرد) را بپذیرد.
DataSource	نام منبع داده (بانک اطلاعات) را تعیین می‌کند.
ValueMember	رشته‌ای را تعیین می‌نماید که باید در خاصیت DataSource نمایش داده شود.
Sorted	تعیین می‌کند آیا اطلاعات ListBox مرتب شده باشد یا خیر.
SelectedItems	لیستی است که گزینه‌های انتخاب شده را نگهداری می‌کند.
رویداد	**هدف**
DataSourceChanged	وقتی خاصیت DataSource تغییر یابد، رخ می‌دهد.
DisplayMemberChanged	وقتی خاصیت DisplayMember تغییر کند، رخ می‌دهد.
ValueMemberChanged	وقتی خاصیت ValueMember تغییر یابد، رخ می‌دهد.
متد	**هدف**
Add	برای اضافه کردن گزینه‌ای به انتهای ListBox به کار می‌رود.
Insert	گزینه‌ای را در مکان خاص ListBox اضافه می‌کند.
Count	تعداد گزینه‌های ListBox را می‌شمارد.
Remove	مقداری را از ListBox حذف می‌کند.
RemoveAt	مقداری را از مکان خاصی از ListBox حذف می‌کند.
Indexof	مقداری را دریافت کرده اندیس مکان آن را برمی‌گرداند.

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[1] کرد. برخی از 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.SetItemsChecked (3, true);
```

این دستورحالت 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.

1- Bind

۹–۸–۱. کنترل GroupBox (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 (MenuStrip)

این کنترل برای ایجاد منو به کار می‌رود. چنانچه این کنترل را به فرم اضافه کنید. شکل ۲–۱ 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 ۲–۱ اضافه کردن منو.

۱۱–۸–۱. کنترل ContextMenuStrip (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 (PictureBox)

این کنترل برای نمایش تصاویری از قبیل نمادهای گرافیکی، تصاویر بیت نگاشت[1]، شبه فایل، آیکن‌ها و غیره به کار می‌رود. برخی از خواص و متدهای این کنترل در زیر آمده‌اند:

1- 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 PictureBox1 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 (تصویر را در اندازه واقعی 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 در 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 ساختارها دو نوع‌اند که عبارتند از:

۱. ساختارهای تصمیم

۲. ساختارهای تکرار

۹-۱-۱. 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:

```
if  (عبارت شرطی)
   {
        مجموعه دستورات۱ ;
   }
else
   {
        مجموعه دستورات ۲ ;
   }
```

در این ساختار، ابتدا **عبارت شرطی داخل پرانتز** ارزیابی می‌گردد، اگر نتیجه ارزیابی درست باشد، **مجموعه دستورات۱** اجرا می‌شوند، وگرنه **مجموعه دستورات۲** 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 شرطی۱)
 {
        مجموعه دستورات ۱ ;
 }
else if     (عبارت شرطی۲)
 {
        مجموعه دستورات ۲ ;
 }
        ⋮
else if   (عبارت شرطیn)
 {
        مجموعه دستوراتn ;
 }
else
 {
        مجموعه دستورات n+1 ;
 }
```

در این ساختار، اگر نتیجه **عبارت شرطی۱**، درست باشد، **مجموعه دستورات۱** اجرا می‌شوند و کنترل اجرای برنامه به بعد از { مربوط به else انتقال می‌یابد. وگرنه، اگر **نتیجه عبارت شرطی۲**، درست باشد، **مجموعه دستورات۲** 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> :
        دستورات۲ ;
        break ;
        :
case <مقدارn> :
        دستورات n ;
        break ;
default :
        دستورات n+1 ;
        break ;
    }
```

در این ساختار، ابتدا عبارت داخل پرانتز switch، ارزیابی می‌شود. اگر نتیجه ارزیابی این عبارت، برابر با **مقدار۱** باشد، **دستورات۱** اجرا خواهند شد و دستور break ساختار switch را خاتمه می‌دهد. اگر نتیجه ارزیابی عبارت برابر **مقدار ۱** نباشد، با **مقدار۲** مقایسه می‌گردد، اگر برابر این مقدار باشد، **دستورات۲** 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 or (||) 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:

for (مقدار اولیه اندیس حلقه ; شرط حلقه ; گام حرکت) {
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 شیء‌ای را تعیین می‌کند که این رویداد بر روی آن رخ داده است.
- **e**: از نوع KeyEventArgs است و 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 کاربر 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 با یک نام و از یک نوع داده را به صورت پشت سر هم در حافظه ذخیره کنید. برای این منظور باید متغیرهای **اندیس‌دار** یا **آرایه**ها را تعریف نمایید. آرایه با توجه به تعداد اندیس‌هایشان به انواع مختلف تقسیم می‌شوند. آرایه‌ای با یک اندیس را **آرایه یک بعدی**، آرایه دارای دو اندیس را **آرایه دو بعدی** و آرایه‌ای 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

ا علان آرایه یک بعدی به صورت زیر انجام می‌شود:

نوع آرایه [ ] نام آرایه ;

این دستور، شکل زیر را ایجاد می‌کند:

نام آرایه [ ] نوع آرایه

?

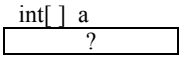
برای تخصیص فضا به آرایه باید نمونه‌هایی از آن را ایجاد کنید. این کار با دستور new و به صورت زیر انجام می‌شود:

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

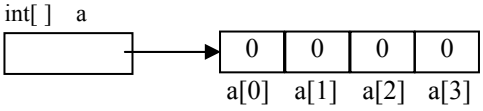
اکنون دستورات زیر را در نظر بگیرید:

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

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



دستور دوم، آرایه‌ای با ٤ 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 (۰) 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 [۰] + a [۱];
```

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 (۱۲+۱۰) 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.Length;
```

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).