طراحی سیستم‌های شی گرا با زبان C#

برای رشته‌های:
مهندسی کامپیوتر، فناوری اطلاعات

مهندس رمضان عباس‌نژادورزی
تالیف: مهندس باقر رحیم‌پورکامی
مهندس ابراهیم هاشمیان

برخی از عناوین مهم

- آشنایی با زبان C#
- برنامه‌نویسی تحت کنسول
- ساختارهای کنترل
- پیاده‌سازی متدها
- ارایه‌ها و رشته‌ها
- کلاس‌ها
- وراثت و واسطها
- برنامه‌نویسی تحت فرم
- برنامه‌نویسی بانک اطلاعاتی
- پروژه‌های متعدد با C#
- ارائه و حل مسائل مختلف

طراحی سیستم‌های شیءگرا

با زبان

C#

تالیف

مهندس رمضان عباس‌نژادورزی

مهندس باقر رحیم پورکامی

مهندس ابراهیم هاشمیان

فن‌آوری نوین

سرشناسه : رمضان عباس‌نژاد ورزی - ۱۳۴۸

عنوان و نام پدیدآور : طراحی سیستم‌های شی گرا با زبان #C/ تالیف رمضان عباس‌نژادورزی، باقر رحیم‌پورکامی، سیدابراهیم هاشمیان.

مشخصات نشر : بابل: فن‌آوری نوین، ۱۳۹۱.

مشخصات ظاهری : ۴۳۲ص.: مصور، جدول.

شابک : ۱۳۰۰۰۰ ریال: 978-600-92254-2-2

موضوع : سی شارپ (زبان برنامه‌نویسی کامپیوتر)

موضوع : برنامه‌نویسی شی گرا

شماره افزوده رحیم‌پور کامی، باقر، ۱۳۶۰ -

شماره افزوده هاشمیان، سیدابراهیم، ۱۳۵۷ -

رده بندی کنگره : QA۷۶/۷۳/ س۹۵ع ۲۵۱۳۹۰

رده بندی دیویی : ۰۰۵/۱۳۳

شماره کتابشناسی ملی : ۲۶۶۶۶۳۰

فن‌آوری نوین

www.fanavarienovin.net

بابل، صندوق پستی ۴۷۱۶۷-۷۳۴۴۸ تلفن: ۲۲۵۶۶۸۷-۰۱۱۱

طراحی سیستم‌های شیءگرا با زبان #C

تالیف: مهندس رمضان عباس‌نژادورزی- مهندس باقر رحیم‌پور کامی- مهندس ابراهیم هاشمیان

ویراستار: مهندس هادی‌عباسی

ناشر: فن‌آوری نوین

چاپ اول: بهار ۱۳۹۱

جلد :۲۰۰۰

شابک: ۲ – ۲ – ۹۲۲۵۴ – ۶۰۰ – ۹۷۸

حروفچینی و صفحه‌آرایی: فن‌آوری نوین **قیمت:۱۳۰۰۰ تومان**

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

فهرست مطالب

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 LINQ با C#	فن آوری نوین	ICDL مهارت ۱: مفاهیم پایه اطلاعات	علوم رایانه
تجارت الکترونیکی	فن آوری نوین	ICDL مهارت۲: به کارگیری کامپیوتر و مدیریت	علوم رایانه
امنیت شبکه	فن آوری نوین	ICDL مهارت۳: واژه پردازی به کمک کامپیوتر	علوم رایانه
اصول طراحی پایگاه‌داده	فن آوری نوین	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	آموزش گام به گام برنامه ویژوال بیسیک‌نت	علوم رایانه
آموزش گام به گام برنامه ویژوال بیسیک	علوم رایانه	برنامه نویسی به زبان اسمبلی	علوم رایانه
برنامه نویسی با دلفی	علوم رایانه		

مقدمه

زبان #C در فناوری دات‌نت (.NET) توسط مایکروسافت ارائه شده است که کاملاً شیءگرا است. امروزه اکثر دانشجویان رشته کامپیوتر با این زبان آشنایی دارند. برنامه‌های متعددی از قبیل تحت کنسول، دسک‌تاپ، بانک‌اطلاعاتی، طراحی صفحات وب، WPF، WCF، تحت شبکه و دستگاه‌های موبایل را می‌توانید با زبان #C بنویسید این کتاب تاکید بیشتری به نوشتن برنامه‌های تحت کنسول نموده است.

از طرف دیگر زبان #C به عنوان سرفصل درس برنامه‌سازی پیشرفته در رشته‌ها کامپیوتر، فناوری اطلاعات، ICT و علوم کامپیوتر تدریس می‌شود.

در حال حاضر کتاب‌های زیادی برای زبان برنامه نویسی #C ارائه شده است که جای تقدیر و تشکر دارد. هر یک از این کتاب‌ها نوع خاص از زبان برنامه‌نویسی #C را مورد بررسی قرار می‌دهند. اما، این کتاب تمرکز بیشتری روی برنامه‌های تحت کنسول دارد.

کتاب حاضر با بیان مسائل متعدد تحت کنسول و حل آن‌ها، دانشجویان محترم را با زبان برنامه‌نویسی #C آشنا می‌کند.

کتاب به صورت گام‌به‌گام به جملات کوتا و ساده بیان گردیده است.

برنامه‌های متن کتاب، مسائل‌حل شده و مسائل حل شده در سایت را می‌توانید از سایت انتشارات فن-آوری نوین به آدرس **www.fanavarienovin.net** دریافت نمایید.

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

مولفین

fanavarienovin@yahoo.com

فصل ۱ آشنایی با زبان C#

پیشرفت‌های زبان‌های برنامه‌نویسی نظیر C++ و جاوا، موجب ایجاد مشکلات و نیازمندی‌های جدیدی گردید. ایجاد یکپارچگی اجزا نرم‌افزاری از زبان‌های مختلف برنامه‌نویسی مشکل بود و در نصب این ابزار مشکلات مشترکی وجود داشت. به همین دلیل بود که نسخه جدید قطعات[1] با نرم‌افزارهای قدیمی سازگار نبود. از طرف دیگر، نیاز به برنامه‌های تحت‌وب، موجب گردید تا NET. و زبان برنامه‌نویسی C# ایجاد شود.

C# زبانی است که برنامه نویسان را قادر می‌سازد، به راحتی بتوانند از زبان‌های مختلف در پروژه‌شان استفاده کنند. زیرا، C# ریشه در C، C++ و جاوا دارد و ابزارهای زیادی از آن‌ها را در خود جمع کرده، علاوه‌بر این، ابزارهای جدیدی به آن‌ها اضافه نموده است و قوانین شی‌گرایی به طور کامل در آن پیاده شده است. در ضمن این زبان با قابلیت برنامه‌نویسی ویژوال، امکان ایجاد برنامه‌هایی با استفاده از **محیط توسعه مجتمع** (IDE)[2] را تامین کرده است. با استفاده از IDE، برنامه‌نویس می‌تواند به راحتی برنامه را ایجاد، اجرا، آزمایش (تست) و رفع اشکال (خطایابی) نماید. پس، در زمان برنامه‌نویسی صرفه‌جویی زیادی خواهد شد. روند ایجاد سریع برنامه‌ها با استفاده از IDE، به **توسعه سریع برنامه** (RAD)[3] معروف است.

مزیت دیگر C# استفاده از قطعات تولیدشده در زبان‌های برنامه‌نویسی مختلف در آن است.

۱ – ۱. فرآیند برنامه‌نویسی در دات‌نت

در طراحی یک برنامه، اولین گام تعیین نوع برنامه‌ای است که می‌خواهید آن را ایجاد کنید. در دات‌نت برنامه‌های متعددی از قبیل برنامه تحت کنسول، برنامه‌های تحت ویندوز، برنامه‌های تحت وب، وب سرویس یا انواع دیگری را می‌توان ایجاد کرد. در این کتاب روش ایجاد برنامه‌های تحت کنسول و تحت ویندوز را می‌آموزیم. گام بعدی انتخاب زبان برنامه‌نویسی می‌باشد. این مرحله از اهمیت ویژه‌ای برخوردار است. چون،

---

1 .Components
2. IDE(Integrated Development Environment)
3.RAD(Rapid Application Development)
4. Common Language Runtime

زبان‌های غیردات‌نت امکانات مختلفی را در اختیارتان می‌گذارند. اما، در دات‌نت زبان‌های مختلف به یکدیگر شبیه شده‌اند و امکانات یکسانی را در اختیارتان قرار می‌دهند. چون این زبان‌ها در هنگام اجرا از زبان مشترک زمان اجرا (CLR)[۴] استفاده می‌کنند. بنابراین، در زمان اجرا این مورد که در طراحی برنامه از چه زبانی استفاده شده است، تفاوتی ندارد. از آنجایی که زبان‌های مختلف گرامرهای متفاوتی دارند، بنابراین باید زبانی انتخاب شود تا با گرامر آن آشنا باشید. چون، گرامر زبان #C شبیه زبان C و C++ است به همین دلیل، زبان برنامه‌نویسی و طراحی را #C انتخاب نمودیم.

ویژوال استودیودات‌نت از کامپایلرها و زبان‌های مختلفی تشکیل شده‌است که عبارت‌اند از:

۱. ویژوال بیسیک ۲. ویژوال #C

۳. ویژوال C++ ۴. ویژوال #F

علاوه‌بر مایکروسافت شرکت‌های دیگر نیز برای زبان‌های خود کامپایلرهایی را عرضه کرده‌اند که CLR را به عنوان محیط زمان اجرای نهایی مورد استفاده قرار داده‌اند. برخی از این زبان‌ها عبارت‌انداز: APL، Cobol، Fortran، ML، Mercury، Perl، Python، RPG، Smalltalk و غیره.

گام سوم، نوشتن کد مورد نیاز برنامه می‌باشد. البته برنامه‌های مختلف کدهای متعددی خواهند داشت. در ادامه با انواع برنامه و کدهای پیاده‌سازی آن‌ها آشنا خواهید شد. گام‌چهارم، کامپایل نمودن برنامه می‌باشد. کامپایل موجب می‌شود تا خطاهای برنامه (نحوی و گرامری) رفع شده، برنامه به کد میانی CLR ترجمه شود.

۲ – ۱. مجموعه کتابخانه کلاس دات‌نت Framework

علاوه بر CLR، در مجموعه دات‌نت Framework، بخش دیگری به نام کتابخانه کلاس چارچوب (FCL)[۲] وجود دارد. این بخش شامل هزاران کلاس می‌باشد که هر کدام وظیفه خاصی دارند. مجموعه‌های FCL و CLR به طراحان اجازه می‌دهند که چندین مدل برنامه را طراحی کنند که عبارت‌اند از:

۱. برنامه‌های تحت کنسول در ویندوز، برنامه‌هایی ایجاد می‌کند که نیاز به رابط گرافیکی کاربر ندارند. این برنامه‌ها از رابط خط فرمان استفاده می‌کنند. این نوع برنامه‌ها معمولاً برای نوشتن ابزارهایی نظیر کامپایلر و بعضی از برنامه‌های کاربردی به کار می‌روند (در فصل‌های ۱ تا ۶ این نوع برنامه‌ها را می‌آموزیم).

---

[2].Framework Class Library [2]. Hyper Text Markup Language [3]. Browser [4].Client
[5].Windows Service Control Manager

۲. برنامه‌های تحت ویندوز، برنامه‌هایی هستند که نیاز به رابط گرافیکی کاربر دارند. برنامه‌هایی تحت ویندوز برنامه‌های دسک‌تاپ نیز نامیده می‌شوند. زمانی که به برنامه‌های تحت وب نیاز نباشد، می‌توان از این برنامه‌ها استفاده نمود (در فصل‌های ۷ تا ۹ روش ایجاد برنامه‌های تحت ویندوز را می‌آموزیم).

۳. برنامه‌های تحت وب، برنامه‌هایی ایجاد می‌کنند که مبتنی بر صفحات HTML[۲] هستند. این نوع برنامه‌ها، از طریق سرویس دهنده بانک اطلاعاتی یا چندین وب سرویس، اطلاعات مورد نیازشان را دریافت کرده، پردازش‌های مورد نیاز را برروی آن انجام داده، صفحات مبتنی بر HTML ایجاد می‌نمایند تا این صفحات از طریق مرورگرهای وب[۳] در کامپیوتر سرویس گیرنده[۴] قابل نمایش باشند.

٤. سرویس‌های ویندوز، سرویس‌هایی را می‌توان در دات‌نت ایجاد کرد که توسط مدیر کنترل سرویس ویندوز (SCM)[۵] و نیز دات‌نت Framework قابل کنترل هستند. معمولاً این سرویس‌ها برای تبادل اطلاعات بین برنامه‌های مختلف استفاده می‌شوند.

۵. وب سرویس‌ها، سرویس‌ها یا توابعی هستند که به راحتی از طریق شبکه وب قابل دسترسی و فراخوانی هستند.

٦. قطعات و کتابخانه کلاس، در دات‌نت Framework می‌توان قطعات[۳] و کتابخانه‌هایی با کلاس‌های جدید ایجاد نمود. این قطعات و کتابخانه‌های کلاس‌های جدید را به راحتی می‌توان در برنامه‌های دیگر (حتی به زبان‌های دیگر) استفاده نمود.

۷. و غیره

۳ – ۱. فضای نام

همان‌طور که بیان گردید، در FCL هزاران کلاس وجود دارند. برای دسته‌بندی کلاس‌ها، تمام کلاس‌های مرتبط به هم در یک فضای نام[۲] قرار می‌گیرند. اصلی‌ترین فضای نام، فضای نام System است. این فضای نام، شامل کلاس object و تعدادی کلاس پایه دیگر است. کلاس object یک کلاس پایه است که تمام کلاس‌های FCL از این کلاس مشتق می‌شوند (در فصل ۶ با مفهوم کلاس‌های پایه و مشتق آشنا خواهید شد). کلاس‌هایی که در FCL وجود دارند، **کلاس‌های آماده** نام دارند. علاوه براین کلاس‌ها، برنامه‌نویس می‌تواند کلاس‌های جدیدی را ایجاد کرده و از آن‌ها استفاده کند. چون، ممکن است کلاس موجود در FCL همه نیازهای برنامه‌نویس را برطرف نکند. چگونگی ایجاد این کلاس‌ها را در فصل ۵ و ۶ می‌آموزیم. در این بخش

---

[2].Namespace [3] .Components

می‌خواهیم به کلاس‌های موجود در FCL بپردازیم. کلاس‌های موجود در FCL با توجه به کاربردشان در فضای‌های نام مختلف قرار می‌گیرند. این عمل دو مزیت زیر را برای برنامه‌نویس در پی دارد:

۱. موجب دسته بندی کلاس‌ها می‌شود. یعنی کلاس‌هایی که به هم مرتبط هستند، در یک فضای نام قرار می‌گیرند تا اولاً بتوان به راحتی آن‌ها را به پروژه اضافه نمود و ثانیاً فضاهای نامی که در پروژه به آن‌ها نیازی نیست، به پروژه اضافه نگردند.

۲. علاوه براین می‌توان در کلاس‌های مختلف از نام‌های تکراری استفاده نمود. فضای نام و کلاس‌ها موجب می‌شوند تا نام‌های تکراری از یکدیگر تفکیک شوند.

وقتی برنامه جدیدی ایجاد می‌کنید فضاهای نام جدول ۱ – ۱ به پروژه اضافه می‌شود (البته وقتی که برنامه‌ای از نوع Console Application ایجاد می‌نمایید. اگر برنامه‌هایی با نوع‌های دیگر به برنامه اضافه کنید، ممکن است فضای‌های نام دیگر به پروژه‌تان اضافه شود). علاوه برفضاهای نامی که به طور خودکار به برنامه اضافه می‌شوند، می‌توانید فضاهای نام دیگر را نیز به پروژه‌تان اضافه کنید. برای این منظور می‌توانید از دستور using به صورت زیر استفاد نمایید:

using نام فضای نام ;

به عنوان مثال، دستورات زیر را ببینید:

```
using System.Convert;
using System.Io;
```

دستور اول، فضای نام System.Convert را به برنامه اضافه می‌کند تا بتوانید از متدهایی که برای تبدیل انواع داده‌های مختلف به کار می‌روند، استفاده نمایید (این متدها را در ادامه می‌آموزیم) و دستور دوم، فضای نام System.Io را به پروژه اضافه می‌کند تا بتوانید از کلاس‌هایی که جهت ورودی – خروجی داده‌ها مانند فایل‌ها به کار می‌روند، استفاده نمایید.

چنانچه در ابتدای برنامه با دستور using فضای نام را اضافه نکنید در کلیه مکان‌هایی که می‌خواهید از کلاس‌های موجود در آن فضای نام استفاده نمایید باید مسیر کامل فضای نام را ذکر کنید (به صورت زیر):

```
System.Convert.ToInt32
```

این دستور از متد ()ToInt32 کلاس Convert موجود در فضای نام System استفاده می‌کند.

> هر پروژه جدیدی که ایجاد می‌شود، یک فضای نام جدید همنام با پروژه نیز ایجاد می‌گردد.

۴ – ۱. آموزش زبان‌های برنامه‌نویسی

آموزش زبان‌های برنامه‌نویسی مانند زبان‌های طبیعی زنده دنیا است. یعنی برای آموزش زبان‌های برنامه‌نویسی باید مراحل زیر را انجام داد:

۱. مانند هر زبان طبیعی ابتدا باید علائم تشکیل دهنده زبان را آموخت. به عنوان مثال، زبان فارسی از علائم الف تا ی، ارقام ۰ تا ۹ و علائم خاص مانند ! ، : ، ؟ و غیره تشکیل شده است. هرکدام از این علائم (نمادها) مفهوم خاصی را دارند. زبان #C، نیز از علائم a تا z ، A تا Z ، ۰ تا ۹ ، علائم ویژه نظیر ; ، : ،] ، [، / و غیره تشکیل شده است. ابتدا باید مفاهیم هریک از این علائم را در زبان #C آموخت.

۲. همان‌طور که می‌دانید از ترکیب علائم هر زبان کلمات بوجود می‌آیند. برخی از کلمات دارای معنی و مفهوم هستند و برخی دیگر معنی و مفهوم خاصی ندارند. به عنوان مثال، کلمات **بابا، آب، داد،** در زبان فارسی مفهوم خاصی دارند. ولی کلمات **تپتانم و بکیاپ** مفهوم خاصی ندارند. به کلماتی که در زبان دارای مفهوم خاص هستند، **کلمات کلیدی** می‌گویند. در زبان #C کلمات کلیدی نظیر for، if، else، while و int وجود دارند. در آموزش این زبان ابتدا باید کلمات کلیدی را شناخت. معنی و کاربرد هرکدام از آن‌ها را باید آموخت.

جدول ۱ – ۱ برخی فضاهای نام.

فضای نام	هدف
System	شامل کلاس‌های پایه دات‌نت و انواع داده از قبیل double، int، char و غیره است.
System.Collection.Generic	شامل کلاس‌هایی اصلی کلکسیون در دات‌نت می‌باشد.
System.LINQ	از کلاس‌هایی تشکیل شده است که برای کاربا LINQ [۴] به کار می‌روند.
System.Text	از کلاس‌هایی تشکیل شده است که برای کارکردن برروی متن از قبیل رمزگذاری، رمزگشایی، کلاس StringBuilder و غیره به کار می‌روند.

۳. در هر زبان طبیعی از ترکیب کلمات کلیدی با یک قواعد خاص، جمله ایجاد می‌شود (مانند بابا آب داد). همان‌طور که می‌دانید در زبان فارسی ابتدا فاعل، سپس مفعول و در پایان فعل قرار می‌گیرد. در زبان #C نیز برای ایجاد جملات (دستورات) قواعد خاصی وجود دارد. به عنوان مثال، int برای تعریف داده‌های نوع صحیح به کار می‌رود و به صورت زیر استفاده می‌گردد:

---

[4] تالیف رمضان عباس‌نژاد ورزی انتشارات LINQ به کتاب آموزش گام‌به‌گام LINQ. برای کسب اطلاعات بیشتر در زمینه فناوری نوین مراجعه فرمایید.

int متغیر۱ , متغیر۲n , ... , متغیر;

٤. همان‌طور که می‌دانید، در زبان‌های طبیعی از کنار هم قرار گرفتن جملات مرتبط به هم پاراگراف ایجاد می‌شود. در زبان‌های برنامه‌نویسی نیز با کنار هم قرار دادن دستورات مرتبط به هم، بلاک ایجاد می‌شود. در زبان #C، هر بلاک با } شروع و با { خاتمه می‌یابد.

۵. چند پاراگراف صفحات و فصول را ایجاد خواهند کرد و این روند ادامه می‌یابد تا یک کتاب نوشته شود. در زبان‌های برنامه‌سازی نیز نوشتن برنامه‌ها هم همین روند را دارد. تعدادی بلاک، فایل، و چند فایل مرتبط به هم، برنامه را ایجاد می‌کند.

در ادامه کتاب به آموزش زبان #C با این شیوه می‌پردازیم.

۵ – ۱. شناسه‌ها

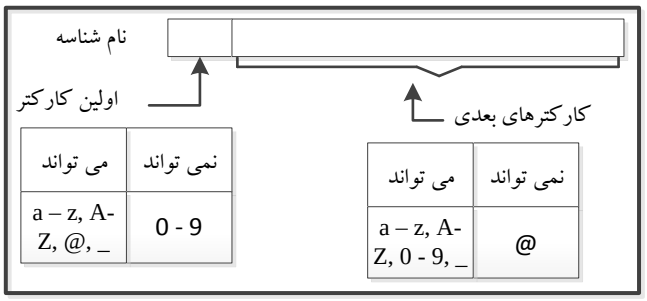
شناسه‌ها[۵]، نام‌هایی هستند که برنامه‌نویس به عناصر #C از قبیل کلاس‌ها[۲]، فضاهای نام[۳]، متدها[۴]، فیلدها[۵]، خواص[۶] و غیره انتخاب می‌کند. به عنوان مثال، (شکل ۱ – ۱) را مشاهده کنید. در این شکل هر کلمه‌ای که در داخل مستطیل قرار دارد، شناسه است. قبل از استفاده از شناسه‌ها باید آن‌ها را نامگذاری نمود. قوانین نامگذاری شناسه‌ها در زیر آمده‌اند (شکل ۱ – ۱). شناسه‌ها در #C با رنگ سبز مشخص می‌گردند.

- کارکترهای الفبایی (A تا Z، a تا z) و خط ربط (_) می‌توانند هر مکان نام شناسه قرار گیرند.

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۱ – ۱ برخی از شناسه‌های در یک برنامه.

[5] .Identifiers [2].Classes [3].Namespaces [4].Methods [5]. Fields [6].Properties



شکل ۲ – ۱ روش نامگذاری شناسه‌ها.

ارقام صفر تا ۹ نمی‌توانند در اولین مکان نام شناسه قرار گیرند، ولی می‌توانند در مکان‌های دیگر نام شناسه مورد استفاده قرار گیرند.

کارکتر @ می‌تواند در اولین مکان نام شناسه قرار بگیرد، اما، نمی‌تواند در مکان‌های دیگر نام شناسه قرار گیرد.

نام شناسه نسبت به حروف بزرگ و کوچک حساس است. یعنی، شناسه‌های myVar و MyVar دو نام مختلف برای دو شناسه در نظر گرفته می‌شوند.

۶ – ۱. کلمات کلیدی

کلماتی که در زبان شناخته شده‌اند و مفهوم خاصی در آن زبان دارند، **کلمات کلیدی**[6] نامیده می‌شوند. برخی از کلمات کلیدی را در (شکل ۳ – ۱) می‌بینید. این کلمات در داخل مستطیل قرار دارند. C# از کلمات کلیدی زیادی تشکیل می‌شود که برخی از آن‌ها را در جدول ۲ – ۱ می‌بینید (کلمات کلیدی در برنامه C# با رنگ آبی مشخص می‌شوند).

[6].Keywords

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۳ – ۱ برخی از کلمات کلیدی در برنامه #C.

جدول ۲ – ۱ کلمات کلیدی #C.

abstract	const	extern	out	int	short	typeof
as	continue	false	override	interface	sizeof	uint
base	decimal	finally	params	internal	stackalloc	ulong
bool	default	fixed	private	is	static	uchecked
break	delegate	float	protected	lock	string	unsafe
byte	do	for	public	long	struct	unshort
case	double	foreach	readonly	namespace	switch	using
catch	else	goto	ref	new	this	virtual
char	enum	if	return	null	throw	void
checked	event	implicit	sbyte	object	true	volatile
class	explicit	in	sealed	operator	try	while
کلمات کلیدی مختص زبان #C						
ascending	by	descending	equals	from	get	group
into	join	let	on	orderby	partial	select
set	value	where	yield			

۷ – ۱. فضای سفید

فضای سفید (whitespace)، کارکترهایی هستند که قابلیت چاپ ندارند. این کارکترها توسط کامپایلر نادیده گرفته می‌شوند، اما برنامه‌نویس برای افزایش خوانایی برنامه از این کارکترها در برنامه‌اش استفاده می‌کند. برخی از این کارکترها عبارت‌اند از : ۱. کارکتر فضای خالی (space) ۲. کارکتر Tab ۳. خط جدید (New Line) و ۴. کلید Enter(carriage return).

۸ – ۱. لیترال‌ها

لیترال‌ها[7]، داده‌هایی هستند که به صورت ثابت در کد برنامه‌تان وارد می‌کنید. لیترال‌ها می‌توانند مقادیر عددی، رشته‌ای (که در بین جفت کتیشن قرار می‌گیرند) یا منطقی (True یا False) باشند. در (شکل ۴ – ۱) برخی از لیترال‌ها را می‌بینید. در این شکل لیترال‌ها در داخل مستطیل قرار دارند.

```
using System;

namespace ProjectNamespace {

    class Program {

        static void Main(string[] args) {

            // perform a calculation
            int x = 10 * 10;

            // print out the result of the calculation
            Console.WriteLine("Result: {0}", x);
        }
    }
}
```

شکل ۴ – ۱ برخی از لیترال‌ها در #C.

۱۸ – ۱. ساختار برنامه #C

برای این که با ساختار برنامه #C آشنا شویم، یک برنامه جدید ایجاد می‌کنیم و از روی برنامه جدید ساختار برنامه را می آموزیم. برای ایجاد برنامه جدید، مثال ۱۳-۱ را ببینید.

مثال ۱۳– ۱. برنامه‌ای که مراحل ایجاد و اجرا یک برنامه در #C را نشان می دهد (هدف این برنامه آشنایی با ایجاد و اجرای برنامه در #C است).

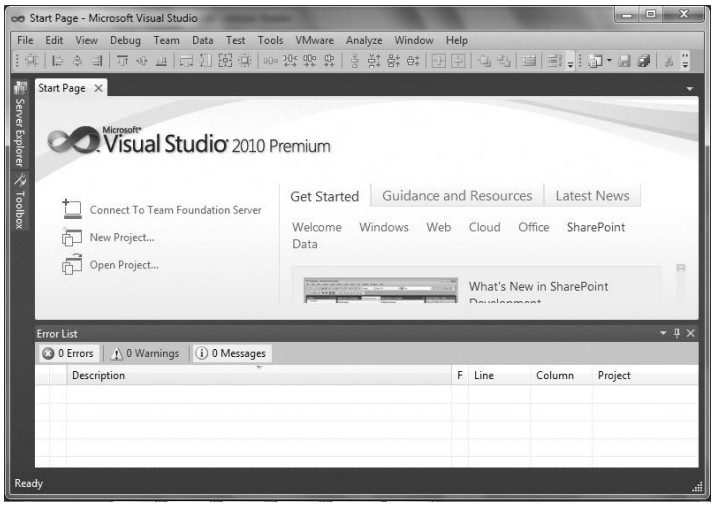
مراحل طراحی و اجرا

۱. نرم افزار #C را اجرا کنید. برای این منظور گزینه زیر را اجرا نمائید:

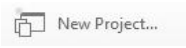
Start/All Programs/Microsoft Visual Studio 2010/Microsoft Visual Studio 2010

۲. اکنون صفحه اول ویژوال استودیو ظاهر می‌شود(شکل۸ – ۱).

[7].Literals

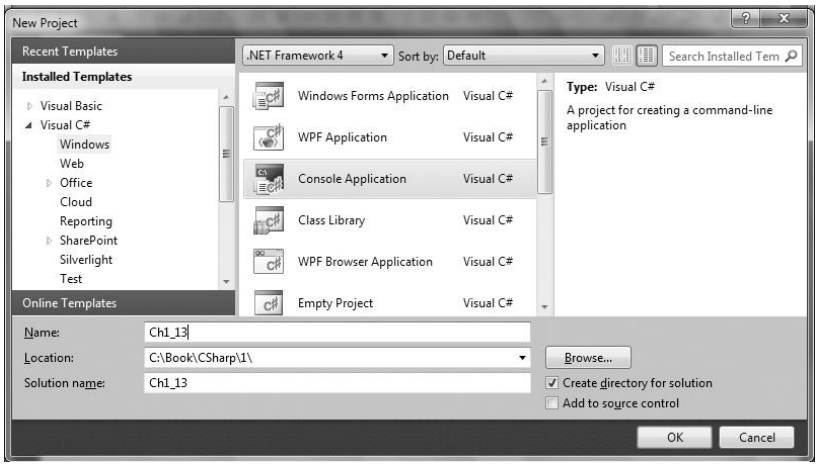


شکل ۸ – ۱ صفحه اول ویژوال استودیو.

۳. گزینه File/New/Project (کلیدهای Ctrl+Shift+N) را اجرا کنید یا آیکن 

را کلیک نمائید. در هر صورت، پنجره New Project ظاهر می‌شود(شکل ۹ – ۱). در این پنجره اطلاعات زیر را انتخاب کنید:

- در سمت چپ، زبان برنامه نویسی و نوع برنامه نویسی را انتخاب کنید. در این برنامه، زبان Visual C# و نوع برنامه نویسی را Windows انتخاب نمایید.
- در پنجره وسط، نوع برنامه C# را انتخاب نمایید. در این برنامه نوع Console Application را انتخاب کنید.
- در جلوی Name، نام پروژه را وارد کنید. در این برنامه Ch1_13 انتخاب شده است.
- در بخش Location مکان ذخیره برنامه را انتخاب کنید. در این برنامه، مسیر C:\Book\Csharp\1\ انتخاب گردیده است.



شکل ۹ – ۱ پنجره New Project.

٤. دکمه OK را کلیک کنید تا برنامه جدید ایجاد شود. دستورات این برنامه به صورت زیر می باشند:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Ch1_13 {
 class Program  {
  static void Main(string[] args)  {
  }
 }
}
```

این برنامه دارای ۵ بخش اصلی است:

۱. بخش فضاهای نام مورد نیاز برنامه، این بخش فضاهای نام مورد نیاز برنامه را معرفی می‌کند. این دستورات با

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

دستور using شروع می‌شوند(دستورات زیر):

فضاهای نام هایی که نیاز ندارید را می‌توانید از برنامه‌تان حذف کنید. در اکثر برنامه‌ها فقط به فضای نام System نیاز است. بنابراین می‌توانید بقیه فضاهای نام را از برنامه حذف نمایید.

در فضای نام System توابع کتابخانه‌ای System نظیر Write()، WriteLine()،Read() و ReadLine() دستورات ورودی و خروجی کنسول وجود دارند. در ادامه با این توابع آشنا خواهید شد.

۲. تعریف فضای نام پروژه، هر برنامه جدیدی که ایجاد می‌کنید، فضای نام جدیدی به نام پروژه ایجاد می‌شود. در این برنامه این فضای نام با دستور زیر ایجاد گردید:

```
namespace Ch1_13
```

۳. تعریف کلاس Program، هر برنامه جدیدی که ایجاد می‌کنید یک کلاس به نام Program ایجاد می‌شود. از طریق این کلاس می‌توان فیلدها، متدها، واسطها [۸] را ایجاد کرد. با مفهوم کلاس در فصل های ۵ و ۶ بیشتر آشنا خواهید شد.کلاس Program با دستور زیر ایجاد گردید:

```
class Program
```

٤. متد Main()، یکی از مهمترین متدهای کلاس Program می‌باشد(وجود متد Main() در تمام برنامه های اجرایی ضروری است). این متد به صورت زیر تعریف شده است:

```
static void Main(string[] args)
```

همان طور که در این دستور مشاهده می‌کنید، متد Main() با کلمه کلیدی static تعریف شده است. این کلمه Modifier نام دارد. کلمه کلیدی static بیان می‌کند که این متد فقط در همین کلاس قابل اجرا می باشد واز طریق هیچ نمونه[۹] دیگری قابل اجرا نمی‌باشد(در فصل‌های ۵ و ۶ با متدهای static بیشتر آشنا خواهید شد). کلمه void در این متد تعیین می‌کند که این متد هیچ مقداری را برگشت نمی دهد. با تعریف متدها در فصل سوم بیشتر آشنا خواهید شد. کلمه args، آرگومان هایی را تعیین می کنند که از طریق خط فرمان می‌توان برای متد Main() ارسال کرد.

۵. دستورات متد Main()، در این برنامه دستورات متد Main() به صورت زیر می‌باشند:

```
{
}
```

چون این برنامه هیچ عملی را انجام نمی‌دهد، بنابراین هیچ دستوری ندارد.

۵. پروژه را ذخیره کنید. برای این منظور، گزینه File/Save All را اجرا نمائید.

٦. پروژه را اجرا کنید. برای این منظور یکی از اعمال زیر را انجام دهید:

- کلید F5 را فشار دهید.
- گزینه Debug/Start Debugging را اجرا نمائید.

در هر صورت پروژه اجرا شده، صفحه سیاهی نمایش داده می‌شود و سریع رد می‌گردد. این صفحه سیاه، صفحه خروجی برنامه نام دارد.

---

[8].Interpace [2].Instance

۱۹ – ۱. دستورات ورودی و خروجی

کنسول ویندوز، خط فرمان ساده ویندوز است که اجازه می‌دهد یک برنامه متنی را نمایش داده و داده ها را از صفحه کلید دریافت کند. برای انجام این کار در C# کلاسی به نام Console (در فضای نام System) وجود دارد که شامل متدهای برای ورودی و خروجی داده در یک کنسول ویندوز است. این کلاس دارای متدهای مختلفی است که متدهای ورودی و خروجی را در ادامه می بینید.

۱ – ۱۹ – ۱. متدهای خروجی

کلاس Console دارای دو متد برای نمایش داده در خروجی است. این دو متد عبارت‌اند از:

جدول ۱۱- ۱ کارکترهای کنترلی.

کارکتر	کد اسکی	هدف
\’	0x0027	کارکتر تک کتیشن (’) را تعیین می کند.
\”	0x0022	کارکتر جفت کتیشن (“) را تعیین می کند.
\\	0x005C	کارکتر بک‌اسلش (Backslash) را تعیین می کند.
\0	0x0000	کارکتر تهی (null) را تعیین می کند.
\a	0x0007	بوق سیستم را به صدا در می‌آورد.
\b	0x0008	کارکترBackspace را تعیین می کند.
\f	0x000C	مکان‌نما را به صفحه بعدی انتقال می‌دهد.
\n	0x000A	مکان‌نما را به سطر بعدی انتقال می‌دهد.
\r	0x000D	کارکتر Carrage return را تعیین می کند.
\t	0x0009	مکان‌نما را به Tab افقی بعدی انتقال می‌دهد.
\v	0x000B	مکان‌نما را به Tab عمودی بعدی انتقال می‌دهد.

متد ()Write، برای نمایش داده در خروجی به کار می‌رود. این متد به صورت زیر استفاده می‌شود:

Console.Write(formatString , subVal0, subVal1, …)

همان طور که در این متد می‌بینید، متد Write چند پارامتر را می پذیرد. این پارامترها عبارت اند از:

۱. پارامتر formatString، رشته فرمت نام دارد. این رشته می‌تواند شامل علامت گذاری‌های جایگزینی[10] باشد. یک علامتگذار جایگزینی، مکانی را در رشته فرمت، علامت گذاری خواهد کرد که از یک مقدار عددی که در داخل }و{ قرار دارد، تشکیل می‌شود. در رشته فرمت می‌توانید از کارکترهای کنترلی استفاده کنید، این کارکترها در جدول ۱۱ – ۱ آمده‌اند.

۲. مقادیر subVal0 , subVal1 و ...، مقادیری هستند که باید جایگزین علامت گذاری‌های جایگزینی در خروجی شوند. این مقادیر جایگزینی شماره دارند که از صفر (۰) شروع می‌شود. یعنی، اولین مقدار دارای شماره صفر، دومین مقدار شماره یک و این روند ادامه می یابد. به عنوان مثال، دستور زیرا را ببینید:

```
Console.WriteLine("Two sample integers are{0}and{1}.",3,6);
```

این دستور خروجی زیر را نمایش می‌دهد:

```
Two sample integers are 3 and 6.
```

همان طور که در این خروجی مشاهده می گردد، مقدار جایگزینی ۳، جایگزین علامت گذار جایگزینی {0} و مقدار جایگزینی 6 جایگزین علامت گذار جایگزینی {1} شده است.

متد WriteLine()، همانند متد Write() است. با این تفاوت که پس از نمایش اطلاعات در خروجی مکان نما را به سطر بعد خروجی انتقال می‌دهد. به عنوان مثال، دستورات زیر را ببینید:

```
Console.Write("One ");
Console.Write("Two ");
```

این دستورات خروجی مقابل را نمایش می‌دهند: One Two

اکنون دستورات زیر را ببینید:

```
Console.WriteLine("One ");
Console.WriteLine("Two ");
```

این دستورات خروجی زیر را نمایش می‌دهند:

```
One
Two
```

متد ReadKey()، برای ایجاد مکث(توقف موقت) در برنامه به کار می‌رود و به صورت زیر استفاده می‌شود:

```
Console.ReadKey();
```

---

[10].Subsutitution

وقتی کنترل اجرای برنامه به این متد برسد، اجرای برنامه به طور موقت قطع خواهد شد تا کاربر کارکتری را فشار دهد. به محض اینکه کاربر کارکتری را فشار داد، اجرای برنامه ادامه می‌یابد. این متد معمولا برای ایجاد توقف به کار می‌رود تا کاربر خروجی برنامه را ببیند.

مثال ۱۴–۱: برنامه که عبارت First output in c# را نمایش می‌دهد.

مراحل طراحی و اجرا

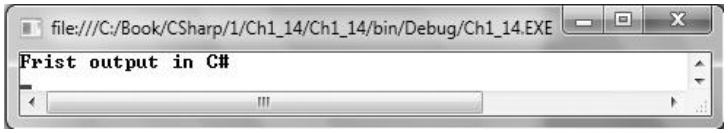
۱. پروژه جدیدی به نام Ch1_14 ایجاد کنید، برای این منظور، گزینه File/New/Project را اجرا نمایید تا پنجره New Project ظاهر شود. در این پنجره، نوع برنامه را Console Application و نام آن را Ch1_14 انتخاب کرده، دکمه OK را کلیک کنید.

۲. به کلاس Program بروید و دستورات آن را به صورت زیر تغییر دهید:

```
using System;
namespace Ch1_14 {
 class Program {
  static void Main(string[] args)  {
    Console.WriteLine("Frist output in C#");
    Console.ReadKey();
  }
 }
}
```

۳. دستور اول داخل متد Main() عبارت First output in C# را نمایش می‌دهد و دستور دوم مکث کوتاهی ایجاد خواهد کرد تا کاربر کلیدی را فشار دهد.

٤. پروژه را ذخیره و اجرا کنید تا خروجی را به شکل زیر مشاهده نمائید.



توجه کنید که زمینه خروجی در حالت عادی مشکی می‌باشد که برای خوانایی کتاب آن را به سفید تغییر دادیم.

۲ – ۱۹ – ۱. متدهای ورودی

در C# برای خواندن اطلاعات ازصفحه کلید متدهای Read() و ReadLine() درکلاس Console وجود دارند. این متدها به صورت زیر به کار می‌روند:

```
Console.Read();
Console.ReadLine();
```

متد ()Read، برای خواندن کارکتر بعدی از ورودی استاندارد به کار می‌رود.

متد ()ReadLine ، برای خواندن یک خط از داده متنی از کاربر به کار می‌رود. کاربر پس از ورود داده باید کلید Enter را فشار دهد.

چنانچه بخواهید داده‌های عددی را از ورودی بخوانید باید آن‌ها را به صورت رشته‌ای خوانده، به عدد تبدیل کنید(تبدیل رشته به عدد را قبلا دیدید).

مثال ۱۵–۱. برنامه‌ای که دو عدد را دریافت کرده، چهارعمل اصلی بر روی دو عدد را انجام می‌دهد.

مراحل طراحی و اجرا

متغیر	هدف
num1	عدد اول
num2	عدد دوم

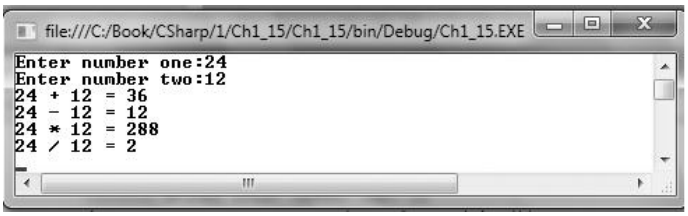
۱. پروژه جدیدی به نام Ch1_15 ایجاد کرده، دستورات کلاس Program.cs را به صورت زیر تغییر دهید:

```
using System;
namespace Ch1_15 {
 class Program  {
  static void Main(string[] args)  {
    Console.Write("Enter number one:");
    int num1 = Convert.ToInt32(Console.ReadLine());
    Console.Write("Enter number two:");
    int num2 = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("{0} + {1} = {2}", num1, num2, num1 + num2);
   Console.WriteLine("{0} - {1} = {2}", num1, num2, num1 - num2);
   Console.WriteLine("{0} * {1} = {2}", num1, num2, num1 * num2);
   Console.WriteLine("{0} / {1} = {2}", num1, num2, num1 / num2);
    Console.ReadKey();
  }
 }
}
```

دستور اول داخل متد ()Main، یک پیغام را نمایش می‌دهد، دستور دوم، رشته‌ای را از صفحه کلید خوانده، به عدد تبدیل می‌کند و در متغیر num1 قرار می‌دهد، دستور سوم، پیغام ورود عدد دوم را به کاربر نمایش می‌دهد، دستور چهارم، رشته‌ای را خوانده به عدد تبدیل می‌نماید و در num2 قرار خواهد داد. دستور پنجم، با متد ()WriteLine، عدد اول ({0} ,num1)، عدد دوم ({1} ,num2) و مجموع دو عدد ({2} ,num1+num2) را نمایش می‌دهد.

دستورات ششم تا هشتم با روش دستور پنجم، حاصل تفریق، ضرب و تقسیم را نمایش می‌دهند.

۲. پروژه را ذخیره و اجرا کنید. دو عدد خواسته شده را وارد نمایید تا خروجی را به شکل زیر مشاهده کنید:



۲۰ – ۱. مسائل حل شده

مثال ۱–۱. برنامه‌ای که حداقل و حداکثر مقدار انواع داده‌های عددی را نمایش می‌دهد (هدف این برنامه آشنایی با خواص Min و Max انواع داده‌های عددی است)

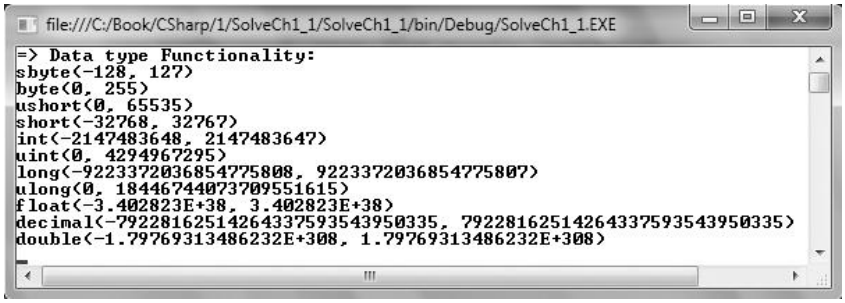
مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh1_1 ایجاد کرده، دستورات کلاس Program.cs، آن را به صورت زیر تغییر دهید:

```
using System;
namespace SolveCh1_1 {
 class Program   {
  static void Main(string[] args)  {
   Console.WriteLine("=> Data type Functionality:");
   Console.WriteLine("sbyte({0}, {1})", sbyte.MinValue,
           sbyte.MaxValue);
   Console.WriteLine("byte({0}, {1})", byte.MinValue,
           byte.MaxValue);
   Console.WriteLine("ushort({0}, {1})", ushort.MinValue,
           ushort.MaxValue);
   Console.WriteLine("short({0}, {1})", short.MinValue,
           short.MaxValue);
   Console.WriteLine("int({0}, {1})", int.MinValue,
           int.MaxValue);
   Console.WriteLine("uint({0}, {1})", uint.MinValue,
           uint.MaxValue);
   Console.WriteLine("long({0}, {1})", long.MinValue,
           long.MaxValue);
   Console.WriteLine("ulong({0}, {1})", ulong.MinValue,
           ulong.MaxValue);
   Console.WriteLine("float({0}, {1})", float.MinValue,
           float.MaxValue);
   Console.WriteLine("decimal({0}, {1})", decimal.MinValue,
           decimal.MaxValue);
   Console.WriteLine("double({0}, {1})", double.MinValue,
           double.MaxValue);
   Console.ReadLine();
  }
```

```
  }
}
```

۲. پروژه را ذخیره و اجرا کنید تا خروجی را به شکل زیر مشاهده کنید:



مثال ۲–۱. برنامه‌ای که نام کاربر را از ورودی می‌خواند و عبارت نام کاربر Welcome را نمایش می‌دهد (هدف این برنامه دریافت و نمایش رشته است)

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh1_2 ایجاد کرده، دستورات کلاس Program.cs، آن را به صورت زیر تغییر دهید:

متغیر	هدف
userName	نام کاربر

```
using System;
namespace SolveCh1_2 {
 class Program  {
  static void Main(string[] args)  {
    Console.Write("Enter user name:");
    string userName = Console.ReadLine();
    Console.WriteLine("Welcome {0}", userName);
    Console.ReadKey();
  }
 }
}
```

۲. پروژه را ذخیره و اجرا کنید. نام کاربر را وارد کرده، کلید Enter را فشار دهید تا خروجی را به شکل زیر مشاهده کنید:

۲۱ – ۱. مسائل حل شده در سایت

۱. برنامه‌ای که عددی دو رقمی را دریافت کرده، مجموع ارقام آن را نمایش می‌دهد.

۲. برنامه‌ای که دو عدد را خوانده، باقی مانده تقسیم عدد اول بر عدد دوم را نمایش می‌دهد.

۳. برنامه‌ای که عددی را خوانده، قدر مطلق آن را نمایش می‌دهد.

٤. برنامه‌ای که سه عدد را خوانده، بزرگترین عدد را نمایش می‌دهد.

٥. برنامه‌ای که عددی را خوانده، اگر رقم یکان بتوان ۲ فرد بود، رقم یکان، وگرنه 0 را نمایش می‌دهد.

٦. برنامه‌ای که x را خوانده، حاصل عبارت $y=3x^4+2x^2$ را نمایش می‌دهد.

۷. برنامه‌ای که نمره دانشجویی را خوانده، اگر نمره بزرگتر از ۱۰ بود، عبارت Passed، وگرنه کلمه Failed را نمایش می‌دهد.

۸. برنامه‌ای که X و Y را خوانده، حاصل عبارت مقابل را نمایش می‌دهد: $Z=X^5 + 2X^4Y^3 -7$

۹. برنامه‌ای که X را خوانده، حاصل عبارت زیر را نمایش می‌دهد: $y = \sqrt{|x| + \sqrt{|x| + \sqrt{|x|}}}$

۱۰. برنامه‌ای که طول و عرض مستطیلی را خوانده، محیط و مساحت آن را نمایش می‌دهد(مساحت مستطیل برابر با طول * عرض می‌باشد و محیط مستطیل برابر با 2* (طول + عرض) است).

۱۱. برنامه‌ای که قاعده و ارتفاع مثلث را خوانده، مساحت مثلث را محاسبه می‌کند و نمایش می‌دهد(مساحت مثلث برابر با ارتفاع ضرب در نصف قاعده است).

۱۲. برنامه‌ای که دو عدد را خوانده، بدون استفاده از متغیر کمکی مقدار آن‌ها را تعویض می‌کند.

۱۳. برنامه‌ای که کارکتری را خوانده، آن را به حروف بزرگ تبدیل می‌نماید و نمایش می‌دهد.

۱٤. برنامه‌ای که X و Y را از ورودی خوانده، مقدار $\log_y(x)$ را نمایش می‌دهد.

۱٥. برنامه‌ای که X را از ورودی خوانده، حاصل e^X را نمایش می‌دهد.

۱٦. برنامه‌ای که X را خوانده (X یک عدد اعشاری است)، بخش صحیح X را نمایش می‌دهد.

۱۷. برنامه‌ای که عدد اعشاری X را خوانده، آن را گرد می‌نماید و نمایش می‌دهد.

۱۸. برنامه‌ای که عدد اعشاری X را خوانده، بخش اعشار آن را نمایش می‌دهد.

۱۹. برنامه‌ای که یک عدد سه رقمی را خوانده، میانگین ارقام آن را نمایش می‌دهد.

۲۰. برنامه‌ای که سن‌تان را به روز خوانده، تعیین می‌کند چند سال، باقی مانده‌اش چند ماه و باقی مانده‌اش چند روز است(هر سال ۳۶۵ روز و هر ماه ۳۰ روز می‌باشد).

۲۲ – ۱. تمرین‌ها

۱. برنامه‌ای بنویسید که شعاع یک دایره را خوانده، محیط، مساحت و قطر آن را نمایش دهد.

$(\lambda = 3.14159)$ (

= قطر دایره 2شعاع *

= محیط دایره 2 * λ شعاع *

= مساحت دایره λ شعاع * شعاع *

۲. سرعت یک دوچرخه سوار در یک جاده سربالایی در مدت یک دقیقه از ۱۰ مایل در ساعت بـه ۲,۵ مایـل در ساعت می‌رسد. برنامه‌ای که شتاب دوچرخه سوار را حساب کرده، و زمانی که دوچرخه سوار می‌ایستد را حسـاب می‌نماید (سرعت اولیه ۱۰ مایل در ساعت فرض شده است). برای حل مساله از فرمول زیر استفاده می‌شود:

$$a = (v_f - v_i)/t$$

سرعت اولیه. v_i، سرعت نهایی و v_f، شتاب، a

۳. چهار ماشین مسابقه در مسافتی به طول یک مایل با یکدیگر مسابقه می‌دهند. برنامه‌ای بنویسیدکه زمان لازم برای هر ماشین که در زیر آمده است را دریافت کند و سرعت هر ماشین را بر مبنای فوت به ثانیه (FPS) و متر بر ثانیه (MPS) محاسبه نامید. هر مایل برابر ۵۲۸۰ فوت و یک کیلومتر برابر با ۳۲۸۲ فوت است.

ثانیه‌ها	دقیقه
۵۲,۸۳	۳
۵۹,۸۳	۳
۰۰,۰۳	۴
۱۶,۲۲	۴

٤. کارخانه‌ای قصد تولید ظروف فلزی را دارد. مساحت این ظروف برابر با مساحت قاعده (λ * شعاع * شعاع) به علاوه مساحت جانبی (λ ۲ * شعاع * ارتفاع ظرف) است. برنامه‌ای بنویسید که شعاع قاعده (radius)، ارتفاع ظرف (height)، ارزش هر سانتی متر مربع از سطح ظرف (cost) و تعداد ظروف تولیدی (quantity) را دریافت کند، قیمت تمام شده هر ظرف و قیمت تمام شده کل ظروف تولیدی را محاسبه کرده، نمایش دهد.

۵. مقداری پول به نیکل و پنی داریم و می‌خواهیم آن ها را به دلار تبدیل کنیم. برنامه‌ای بنویسید که این کار را انجام دهد. برای انجام این کار ابتدا پول را به سنت تبدیل می‌کنیم. یعنی،

پنی + نیکل * ۵ = پول به سنت. سپس، خارج قسمت آن را به ۱۰۰ محاسبه کرده تا دلار بدست آید و باقی مانده تقسیم صحیح آن ۱۰۰، سنتهای باقی را نشان دهد.

٦. برنامه‌ای بنویسید که وزن یک شیء را به پوند گرفته، آن را به گرم و کیلوگرم تبدیل کرده، نمایش دهد. هر پوند معادل ۰/۴۵۳۵۹۲ کیلوگرم است.

۷. اگر قلب انسان به طور متوسط در هر ثانیه یک بار بطپد، برنامه‌ای بنویسید که طول عمر را به سال گرفته و تعیین کند که قلب او چند بار طپیده است (هر سال ۲۳۶۵۲۵روز می‌باشد)

۸. برنامه‌ای بنویسید که طول و عرض زمین مستطیل شکلی را گرفته، سپس، سرعت ماشین چمن زنی را دریافت نماید. زمان مورد نیاز برای چمن زدن این زمین را محاسبه کرده، نمایش دهد.

۹. برنامه‌ای بنویسید که صورت و مخرج دو کسر را گرفته حاصل ضرب، حاصل تقسیم، تفریق و حاصل جمع این کسرها را نمایش دهد. نتایج عبارات را به صورت درصد نیز چاپ نماید.

۱۰. برنامه‌ای بنویسید که نام یک دایناسور و زمانی که می‌زیسته (یعنی چند سال پیش) را خوانده، این زمان را بر حسب ماه، روز و ثانیه محاسبه کرده، چاپ کند (هر سال ۳۶۵/۲۵ روز است).

فصل ۲

ساختارهای کنترلی

در برنامه‌های ساده (برنامه‌هایی که تاکنون در فصل اول نوشته‌اید)، دستورات به صورت پشت سرهم (از اولین دستور به آخرین دستور) اجرا می‌گردند. در برنامه‌های پیچیده و واقعی نیاز است بعضی از دستورات تحت شرایط خاصی اجرا شوند، برخی دیگر از دستورات اجرا نشوند یا برخی از دستورات چندین بار تکرار گردند. برای پیاده‌سازی این برنامه‌ها از ساختارهای کنترلی استفاده می‌شود. ساختار کنترلی دو نوع‌اند که عبارت‌اند از:

۱. ساختارهای تصمیم‌گیری

۲. ساختارهای تکرار

۱ ـ ۲. ساختارهای تصمیم‌گیری

این ساختارها برای مواقعی به کار می‌روند که بخواهید با درست بودن شرط خاصی، مجموعه‌ای از دستورات اجرا گردند. وگرنه (در صورت درست نبودن شرط) برخی دستورات دیگر اجرا شوند. ساختارهای تصمیم در C# عبارت‌اند از: if، switch.

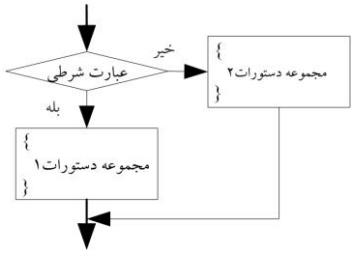
۱ ـ ۱ ـ ۲. ساختار تصمیم if

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

در این ساختار ابتدا شرطی ارزیابی می‌شود، اگر

```
}
```

نتیجه ارزیابی شرط درست (True) باشد، یک یا مجموعه‌ای از دستورات اجرا می‌شوند. وگرنه (در صورت نادرست بودن (False) شرط)، مجموعه دیگری از دستورات اجرا خواهند شد. این ساختار به صورت زیر به کار می‌رود:



در این ساختار، ابتدا **عبارت شرطی داخل پرانتز** ارزیابی می‌شود، اگر نتیجه ارزیابی درست(True) باشد، **مجموعه دستورات ۱** اجرا می‌گردند، در غیر این صورت (نتیجه ارزیابی نادرست (False) باشد)، **مجموعه دستورات ۲** اجرا خواهند شد.

در هنگام استفاده از ساختار if به نکات زیر دقت کنید:

۱. کلمات if و else باید با حروف کوچک نوشته شوند. چون زبان #C یک زبان حساس به متن (case setitive) است. یعنی، بین حروف بزرگ و کوچک فرق قائل می‌شود. بنابراین، تمام کلمات کلیدی را با حروف کوچک تایپ کنید، وگرنه کامپایلر #C خطا می‌گیرد.

۲. در این ساختار شرط می‌تواند مرکب باشد. یعنی، می‌توان با عملگرهای && و || شرط‌های ترکیبی را ایجاد نمود. به عنوان مثال، شرط می‌تواند به صورت زیر بیان گردد:

```
(x > 10 && x < 17)
```

این شرط بررسی می‌کند که x بین ۱۰ تا ۱۷ است یا خیر.

۳. اگر هر یک از بخش‌های if و else فقط یک دستور داشته باشد، کارکترهای } و { را می‌توان حذف کرد (اگر حذف نشود، خطایی از طرف کامپایلر #C اعلان نمی‌گردد). برای افزایش خوانایی برنامه بهتر است کارکترهای } و { را قرار دهید.

۴. در این ساختار می‌توان بخش else را حذف نمود. در این صورت، اگر نتیجه ارزیابی شرط نادرست باشد، **مجموعه دستورات ۱** اجرا نمی‌شوند و کنترل اجرای برنامه به بعد از بلاک بسته (})، if انتقال می‌یابد.

۵. برای تست برابری می‌توان از عملگر == استفاده کرد. چنانچه از عملگر = استفاده شود، از طرف کامپایلر #C پیغام خطا نمایش داده می‌شود.

```
int x = 10;
if (x = 5)
{
  x++;
}
else
{
 x--;
}
```

اجرای این دستور موجب صدور خطا از طرف کامپایلر #C خواهد شد.

۶. همان‌طور که در ساختار if دیدید، بعد از پرانتز بسته if علامت ; قرار نمی‌گیرد. چنانچه برنامه‌نویس علامت ; قرار دهد کامپایلر خطا می‌گیرد (مانند دستور زیر):

```
if (x > 10);
 { x ++;}
```

این دستور موجب صدور خطا از طرف کامپایلر می‌شود.

۷. کامپایلر #C همیشه else را با if قبلی آن متناظر قرار خواهد داد. اگر بخواهید غیر این باشد، باید کدهای مربوط را در داخل { و بلاک { قرار دهید. دستورات زیر را ببینید:

```
if (x == 0)
 if (y == 0)
   Console.WriteLine("x and y = 0");
else
   Console.WriteLine("x != 0");
```

همان‌طور که در این دستورات مشاهده می‌کنید، ابتدا x با صفر مقایسه می‌شود. اگر برابر صفر باشد، سپس، y با صفر مقایسه می‌گردد، اگر برابر صفر باشد، عبارت "x and y = 0" نمایش داده می‌شود. وگرنه، "x != 0" چاپ خواهد شد. برای افزایش خوانایی برنامه بهتر است، دستور به صورت زیر استفاده شود:

دستور زیر را ببینید:

```
if (x == 0)
{
 if (y == 0)
   Console.WriteLine("x and y = 0");
 else
   Console.WriteLine("x != 0");
}
```

```
if (grade >= 60)
   Console.WriteLine("Passed");
else
   Console.WriteLine("Failed");
```

این دستورات اگر نمره(grade) بزرگتر

از ۶۰ باشد، عبارت "Passed"، وگرنه عبارت "Failed" را نمایش می‌دهد.

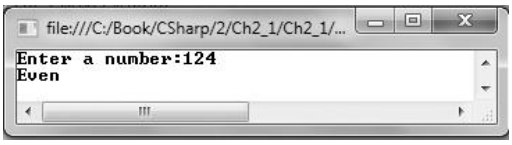
مثال اپ – ۲. برنامه‌ای که عددی را خوانده، تشخیص می‌دهد زوج است یا فرد.

توضیح: در این برنامه ابتدا پیغامی نمایش داده می‌شود تا کاربر عددی را وارد کند. مقدار وارد شده توسط کاربر با متد Parse به عدد تبدیل شده در num قرار می‌گیرد. سپس، شرط num % 2 با صفر مقایسه می‌شود. اگر برابر صفر باشد، کلمه "Even"، وگرنه کلمه "Odd" نمایش داده می‌شود.

متغیر	هدف
num	عدد خوانده شده

```
using System;
namespace Ch2_1 {
 class Program  {
  static void Main(string[] args) {
    Console.Write("Enter a number:");
    int num = int.Parse(Console.ReadLine());
    if (num % 2 == 0)
      Console.WriteLine("Even");
    else
      Console.WriteLine("Odd");
    Console.ReadKey();
  }
 }
}
```

خروجی:



مثال ۲ – ۲. برنامه‌ای که حقوق کارمندی را خوانده، مالیات را محاسبه کرده و نمایش می‌دهد (اگر حقوق بیش از ۴۸۵۰۰۰ تومان باشد، ۱۰درصد مازاد بر ۴۸۵۰۰۰ تومان مالیات، وگرنه مالیات صفر خواهد بود).

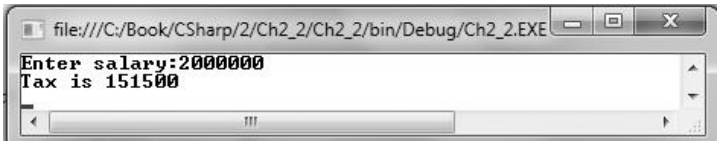
توضیح: در این برنامه ابتدا پیغامی نمایش داده می‌شود تا کاربر حقوق را وارد کند. مقدار وارد شده توسط کاربر با متد Parse به عدد تبدیل شده در salary قرار می‌گیرد. سپس متغیر tax تعریف شده، مقدار اولیه آن صفر در نظر گرفته می‌شود. اگر حقوق (salary) بیشتر از ۴۸۵۰۰۰ تومان باشد، مالیات را حساب کرده در tax قرار می‌دهد و در پایان مقدار tax را نمایش می‌دهد (در این برنامه بخش else حذف شده است).

متغیر	هدف
salary	حقوق
tax	مالیات

```
using System;
namespace Ch2_2 {
 class Program  {
  static void Main(string[] args) {
    Console.Write("Enter salary:");
    long salary = long.Parse(Console.ReadLine());
    long tax = 0;
    if (salary > 485000)
```

```
      tax=(salary - 485000) * 10 /100;
    Console.WriteLine("Tax is {0}", tax);
    Console.ReadKey();
  }
 }
}
```

خروجی:



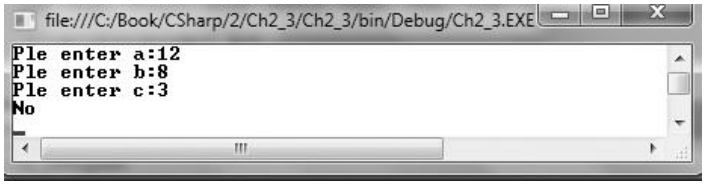
مثال ۳ – ۲. برنامه‌ای که سه عدد را خوانده، تشخیص می‌دهد این سه عدد تشکیل مثلث را می‌دهند یا خیر (سه عدد زمانی تشکیل مثلث را می‌دهند که مجموع هر دو ضلع بیشتر از ضلع سوم باشد). هدف این برنامه آشنایی با عملگر && (و منطقی) است.

توضیح: این برنامه ابتدا اضلاع مثلث را با پیغام مناسب از کاربر دریافت کرده، در متغیرهای a، b و c قرار می‌دهد. اگر a + b > c و b + c > a و a + c > b باشد، آنگاه "Yes"، وگرنه "No" را نمایش می‌دهد.

متغیر	هدف
a	ضلع اول
b	ضلع دوم
c	ضلع سوم

```
using System;
namespace Ch2_3 {
 class Program  {
  static void Main(string[] args) {
    Console.Write("Ple enter a:");
    int a=Convert.ToInt32(Console. ReadLine ());
    Console.Write("Ple enter b:");
    int b = Convert.ToInt32(Console.ReadLine());
    Console.Write("Ple enter c:");
    int c = Convert.ToInt32(Console.ReadLine());
    if (a + b > c && b + c > a && a + c > b)
       Console.WriteLine("Yes");
    else
       Console.WriteLine("No");
    Console.ReadKey();
  }
 }
}
```

خروجی:



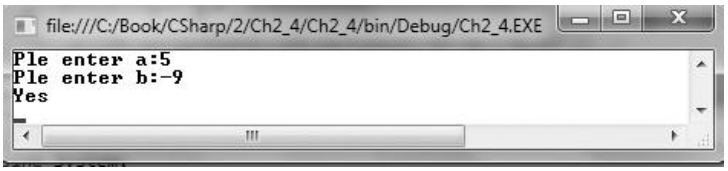
مثال ۴ – ۲. برنامه‌ای که دو عدد را خوانده، اگر هر یک از اعداد منفی باشند، "Yes" وگرنه "No" را نمایش می‌دهد (هدف این برنامه آشنایی با عملگر || (یا منطقی) است).

توضیح: برنامه، ابتدا دو عدد را با پیغام مناسب از کاربر دریافت کرده، در متغیرهای a و b قرار می‌دهـد. سپس اگر $a < 0$ یا $b < 0$ باشد، کلمه "Yes" وگرنه کلمه "No" را نمایش می‌دهد.

متغیر	هدف
a	اولین عدد
b	دومین عدد

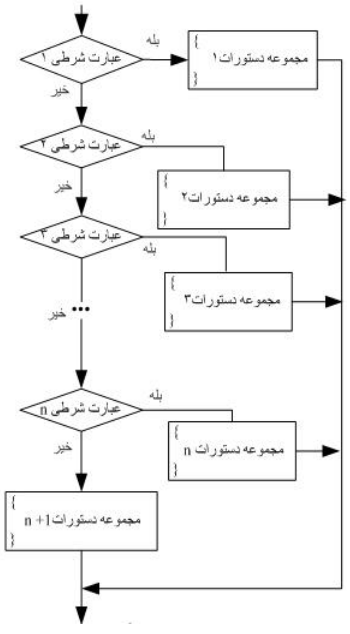
```
using System;
namespace Ch2_4 {
 class Program  {
  static void Main(string[] args) {
    Console.Write("Ple enter a:");
    int a =
Convert.ToInt32(Console.ReadLine());
    Console.Write("Ple enter b:");
    int b = Convert.ToInt32(Console.ReadLine());
    if (a < 0 || b <0)
       Console.WriteLine("Yes");
    else
       Console.WriteLine("No");
    Console.ReadKey();
  }
 }
}
```

خروجی:



۲ ـ ۱ ـ۲ . ساختار if تودرتو

شرط‌هایی که تاکنون دیدید، دو حالته بودند. گاهی ممکن است بخواهید شرط‌های متعدد و ناسازگاری را بررسی کنید. برای این منظور، می‌توانید از ساختار if تودرتو به صورت زیر استفاده نمایید:




```
if (عبارت شرطی ۱)
{
    مجموعه دستورات ۱;
}
else if(عبارت شرطی ۲)
{
    مجموعه دستورات ۲;
}
else if(عبارت شرطی ۳)
{
    مجموعه دستورات ۳;
}
```

```
…
else if(n شرطی عبارت)
{
    n مجموعه دستورات;
}
else
{
   n + 1 مجموعه دستورات;
}
```

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 می‌گردند، دستور if پایان می‌یابد و این روند ادامه می‌یابد. اگر هیچ یک از **عبارات شرطی ۱ تا n** نتیجه **درستی** نداشته باشند، **مجموعه دستورات n + 1**، اجرا می‌گردند.

نکته:	در این ساختار بخش else می‌تواند حذف شود. یعنی، اگر هیچ یک از عبارات شرطی ۱ تا n درست نباشد، کنترل اجرای برنامه به بعد از if انتقال می‌یابد و هیچ یک از مجموعه دستورات ۱ تا n اجرا نمی‌گردند.

مثال ۱۳ – ۲. برنامه‌ای که x و n را از ورودی خوانده، مجموع n جمله سری زیر را نمایش می‌دهد.

$$\frac{1}{-x}-\frac{1-2}{-x+2x^2}+\frac{1-2+3}{-x+2x^2-3x^3}-\dots$$

توضیح: در این برنامه ابتدا x و n را با پیغام مناسب از کاربر دریافت می‌کند. سپس، علامت عدد (sign) را برابر یک قرار می‌دهد، در ادامه مجموع صورت کسر (sum1)، مخرج کسر (sum2) و مجموع کل سری (sum) را برابر صفر قرار می‌دهد و p حاصل توان‌های x را برابر یک قرار می‌دهد و در پایان طبق جدول زیر مقادیر متغیر را محاسبه می‌کند.

i	p	sign	sum1	sum2	Sum

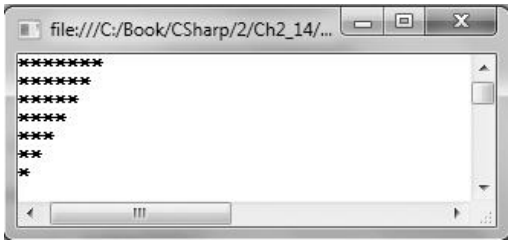
1	x	1	0 + 1	-x	$\frac{1}{-x}$
2	x^2	-1	1 – 2	$-x + 2x^2$	$\frac{1}{-x} - \frac{1-2}{-x+2x^2}$
3	x^3	1	1 – 2 + 3	$-x + 2x^2 - 3x^3$	$\frac{1}{-x} - \frac{1-2}{-x+2x^2} + \frac{1-2+3}{-x+2x^2-3x^3}$

متغیر	هدف
x	عدد ورودی
n	تعداد جملات
sign	علامت
i	nشمارنده از ۱ تا
sum1	مقدار سری صورت مسئله
p	xتوان‌های
sum2	مجموع سری مخرج
sum	مجموع سری کل

```
Using System;
namespace Ch2_13 {
 class Program  {
  static void Main(string[] args) {
   Console.Write("Enter x:");
   int x = Convert.ToInt32
        (Console.ReadLine());
   Console.Write("Enter n:");
   int n = Convert.ToInt32
       Console.ReadLine());
   int sign = 1;
    double sum1 = 0, sum2 = 0,
           sum = 0, p = 1;
   for (int i = 1; i <= n; i++) {
        p *= x;
        sum1 += i * sign;
        sum2 += p * i * sign * -1;
        sum += (double) sum1 / sum2 * sign;
        sign = -sign;
      }
      Console.WriteLine("Sum is {0}", sum);
      Console.ReadKey();
    }
  }
}
```

خروجی:

مثال ۱۴ – ۲. برنامه‌ای که خروجی زیر را نمایش می‌دهد:



توضیح: این برنامه شامل دو حلقه تودرتو است. i شمارنده حلقه خارجی از ۱ تا ۷ را می‌شمارد. چون خروجی ۷ سطر دارد و j شمارنده حلقه داخلی از i تا ۷ می‌شمارد. زیرا برای سطر اول i برابر ۱ است. پس از ۱ تا ۷ می‌شمارد و ۷ ستاره چاپ می‌کند. اما، برای زمانی که i برابر ۲ است j از ۲ تا ۷ می‌شمارد (۶ ستاره نمایش می‌دهد) و این روند ادامه می‌یابد.

متغیر	هدف
i	شمارنده حلقه خارجی از ۱ تا ۷ می‌شمارد
j	تا ۷ می‌شمارد i شمارنده حلقه داخلی از

```
using System;
namespace Ch2_14 {
 class Program   {
  static void Main (string[]args){
   for (int i = 1; i <= 7; i++){
    for(int j= i; j<=7; j++) {
      Console.Write("*");
    }
    Console.WriteLine();
   }
   Console.ReadLine();
  }
 }
}
```

مثال ۱۸ – ۲. برنامه‌ای که تعدادی عدد را خوانده، اعدادی که تمام ارقام آن‌ها برابر هستند، را نمایش می‌دهد. در پایان مجموع اعدادی که تمام ارقام آن‌ها برابرمی‌باشد را نمایش می‌دهد. در این برنامه بعد از ورود هر عدد از کاربر می‌پرسد که آیا ادامه می‌دهد یا خیر.

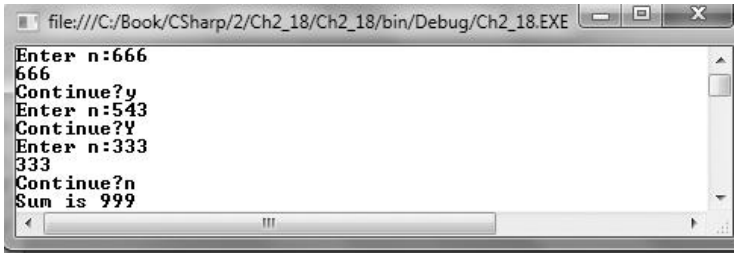
توضیح: در این برنامه ابتدا با یک پیغام عددی را خوانده در n قرار می‌دهد. برای این که عدد n را از دست ندهد، n را در n1 قرار می‌دهد. اکنون، فرض می‌شود تمام ارقام n1 برابر هستند (eq = true). رقم اول را در r1 قرار می‌دهد و سپس با استفاده از یک حلقه تکرار while ارقام n1 را یکی یکی جدا کرده با رقم اول مقایسه می‌کند و این روند ادامه می‌یابد تا حلقه خاتمه یابد. در پایان حلقه چک می‌کند آیا eq برابر true است یا خیر. اگر eq برابر true باشد n را نمایش داده و با sum جمع نموده، در sum قرار می‌دهد و در پایان از کاربر می‌پرسد آیا ادامه می‌دهد یا خیر. اگر کاربر یکی از کارکترهای y یا Y را وارد کند، برنامه ادامه می‌یابد.

متغیر	هدف
r1	رقم اول
r2	رقم فعلی
n	عدد خوانده شده در هر بار
n1	(متغیر کمکی)همان n
eq	آیا همه ارقام عدد برابرند یا خیر
sum	

مجموع اعدادی که ارقام آن‌ها برابرند	

```
using System;
namespace Ch2_18 {
 class Program  {
  static void Main(string[] args) {
   int r1, r2, n, sum = 0;
   char ch;
   do {
    Console.Write("Enter n:");
    n=Convert.ToInt32(Console
.ReadLine());
    int n1 = n;
    bool eq = true;
    r1 = n % 10;
    while (n1 > 0){
      r2 = n1 % 10;
      if (r1 != r2) { eq = false; break; }
      n1 /= 10;
    }
    if (eq == true) {
      Console.WriteLine("{0} ", n);
      sum += n;
    }
     Console.Write("Continue?");
     ch = Convert.ToChar(Console.Read());
     Console.ReadLine();
   }while (ch == 'y' || ch == 'Y');
   Console.Write("Sum is {0}", sum);
   Console.ReadKey();
  }
 }
}
```

خروجی:



۳ – ۲. مسائل حل شده

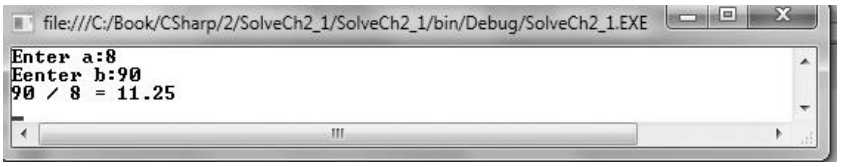
مثال ۱ – ۲. برنامه‌ای که دو عدد را خوانده، عدد بزرگتر را به عدد کوچکتر تقسیم می‌کند.

متغیر	هدف

a	اولین عدد
b	دومین عدد

```
using System;
namespace SolveCh2_1 {
 class Program       {
  static void Main(string[] args) {
   Console.Write("Enter a:");
   int a = Convert.ToInt32(Console.ReadLine());
   Console.Write("Eenter b:");
   int b = Convert.ToInt32(Console.ReadLine());
   if (a > b)
     Console.WriteLine("{0} / {1} = {2}",a, b, (float) a / b);
   else
     Console.WriteLine("{0} / {1} = {2}", b, a, (float) b / a);
   Console.ReadKey();
  }
 }
}
```

خروجی:



مثال ۲ – ۲. برنامه‌ای که a و x را خوانده، حاصل معادله زیر را نمایش می‌دهد:

اگر $x >= a$ باشد $x + x^2$ وگرنه x

متغیر	هدف
a	عدد ورودی اول
x	عدد وردی دوم

```
using System;
namespace solveCh2_2 {
 class Program       {
  static void Main(string[] args) {
   Console.Write("Enter a:");
   int a =
Convert.ToInt32(Console.ReadLine());
   Console.Write("Eenter x:");
   int x = Convert.ToInt32(Console.ReadLine());
   if (x >= a)
     Console.WriteLine("{0}+ {0}^ 2 / {0} = {1}", x, x + x* x);
   else
     Console.WriteLine("x  = {0}", x);
   Console.ReadKey();
  }
 }
}
```

خروجی:

مثال ۳ – ۲. برنامه‌ای که حقوق کارمندی را خوانده، با روش زیر مالیات را محاسبه می‌کند و نمایش می‌دهد:

اگر حقوق کمتر یا مساوی ۴۸۵۰۰۰ تومان باشد، از پرداخت مالیات معاف است (مالیات ۰ در نظر گرفته می‌شود). وگرنه، اگر حقوق از ۴۸۵۰۰۰ تا ۱۰۰۰۰۰۰ تومان باشد، مالیات برابر با ۱۰/۱۰۰ × (۴۸۵۰۰۰ – حقوق) است. وگرنه، اگر حقوق از ۱۰۰۰۰۰۰ تا ۲۵۰۰۰۰۰ تومان باشد، مالیات برابر با ۱۰/۱۰۰ × (۴۸۵۰۰۰ – ۱۰۰۰۰۰۰) + ۱۰۰ / ۱۵ × (۱۰۰۰۰۰۰ – حقوق) است. اما اگر حقوق بالای ۲۵۰۰۰۰۰ تومان باشد، مالیات برابر است با:

۱۰/۱۰۰ × (۴۸۵۰۰۰ – ۱۰۰۰۰۰۰) + ۱۵/۱۰ × (۱۰۰۰۰۰۰ – ۲۵۰۰۰۰۰) + ۱۰۰ / ۲۰ × (۲۵۰۰۰۰۰ – حقوق)

متغیر	هدف
salary	حقوق
tax	مالیات

```
using System;
namespace solveCh2_3 {
 class Program       {
  static void Main(string[] args)  {
   Console.Write("Enter salary:");
   int salary = Convert.ToInt32(Console.ReadLine());
   int tax = 0;
   if (salary <= 485000) {
     tax = 0;
   }
   else if (salary <= 1000000){
     tax = (salary - 485000) * 10 / 100;
   }
   else if (salary <= 2500000){
     tax=(1000000- 485000)*10/100+(salary-1000000)*15/100;
   }
   else {
      tax = (1000000 - 485000) * 10 / 100 + (2500000 -
         1000000) * 15 / 100 + (salary - 2500000) * 20 / 100;
   }
   Console.Write("Tax is {0}", tax);
   Console.ReadKey();
  }
 }
}
```

خروجی:

مثال ۱۳– ۲. برنامه‌ای که عددی را خوانده، در هر مرحله عدد و در پایان حاصل ضرب ارقام فرد آن را نمایش می‌دهد.

توضیح: در این برنامه برای تعیین رقم یکان از عملگر ٪ استفاده می‌شود (یعنی n % 10، رقم یکان عدد می‌باشد).

متغیر	هدف
n multiply digit	عدد خوانده شده و عدد در هر مرحله حاصل ضرب ارقام فرد عدد رقم یکان عدد در هر مرحله

```
using System;
namespace SolveCh2_13 {
 class Program  {
  static void main(string[]args){
    int n, multiply = 1;
    Console.Write("Enter a
                    number:");
    n = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("   Number       Digit ");
    Console.WriteLine("=========  =========");
    while (n > 0)  {
      int digit = n % 10;
      Console.WriteLine("{0}\t\t{1}", n, digit);
      if (digit % 2 == 1) multiply *= digit;
      n /= 10;
    }
    Console.WriteLine("\nMultiply is {0}", multiply);
    Console.ReadKey();
  }
 }
}
```

خروجی:

٤ – ٢. مسائل حل شده درسایت

۱. برنامه‌ای که دو عدد را خوانده، باقی مانده تقسیم صحیح عدد بزرگتر بر عدد کوچکتر را نمایش می‌دهد.

۲.برنامه‌ای که عددی را خوانده تشخیص می‌دهد بر ۶ بخش‌پذیر است یا خیر (عددی بر ۶ بخش‌پذیر است که بر ۲ و ۳ بخش‌پذیر باشد).

۳. برنامه‌ای که عددی را خوانده، اگر عدد بزرگتر از صفر باشد، "Positive"، اگر عدد صفر باشد، "Zero"، وگرنه "Negative" را نمایش می‌دهد.

۴. برنامه‌ای که عددی یک رقمی را خوانده معادل لاتین آن را نمایش می‌دهد، اگر عدد یک رقمی نباشد، پیغام خطا را نمایش خواهد داد.

۵. برنامه‌ای که دو عدد را خوانده تعیین می‌کند باقی مانده تقسیم عدد اول بر عدد دوم زوج است یا خیر.

۶.برنامه‌ای که عددی را خوانده، قدرمطلق آن را نمایش می‌دهد. قدرمطلق عدد به صورت زیر تعیین می‌شود:
اگر عدد کوچکتر از صفر باشد، منفی عدد را نمایش می‌دهد. اگر عدد برابر صفر باشد، صفر را نمایش خواهد داد.
وگرنه، همان عدد را نمایش می‌دهد.

۷. برنامه‌ای که ضرایب یک معادله درجه ۲ (a، b و c) را خوانده، ریشه‌های آن را محاسبه می‌کند و نمایش می‌دهد. برای محاسبه ریشه‌های معادله درجه ۲ به صورت زیر عمل می‌شود. ابتدا دلتا (delta) به صورت زیر محاسبه شود:

$$delta = b^2 - 4 * a * c$$

اگر delta برابر صفر باشد، معادله دو ریشه دارد و ریشه‌های delta، کوچکتر از صفر باشد، معادله ریشه ندارد. اگر delta برابرند و به صورت زیر محاسبه می‌شوند(x1 , x2 آن :)

$$x1 = x2 = -b / (2 * a)$$

اگر delta ، بزرگتر از صفر باشد، معادله دارای دو ریشه است که به صورت زیر محاسبه می‌شوند:

$$x1 = \frac{-b + \sqrt{delta}}{2*a} \quad \text{و} \quad x2 = \frac{-b - \sqrt{delta}}{2*a}$$

۸. برنامه‌ای که نمره دانشجویی که بین ۰ تا ۱۰۰ است را خوانده، اگر نمره کوچکتر مساوی ۷۰ باشد، عبارت "Fail"، اگر نمره بین ۷۰ تا ۸۰ باشد؛ عبارت "Good"، اگر نمره بین ۸۰ تا ۹۰ باشد، عبارت "Very Good"، اگر نمره بین ۹۰ تا ۱۰۰ باشد، "Excellent"، وگرنه "Invalid number" را نمایش می‌دهد.

۹. برنامه‌ای که عددی را از وردی خوانده، مانند خروجی زیر را چاپ می‌کند (اگر عدد وارد شده

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

۳ باشد، خروجی مقابل تولید می‌شود):

۱۰. برنامه‌ای که عددی را از وردی خوانده، بزرگترین رقم آن را نمایش می‌دهد.

۱۱. برنامه‌ای که عددی را خوانده، با استفاده از حلقه while مجموع مکعبات ۱ تا عدد خوانده شده را نمایش می‌دهد. به عنوان مثال، اگر عدد ۳ وارد شود، عبارت به صورت $۳۶ = ۳^۳ + ۲^۳ + ۱^۳$ را نمایش دهد.

۱۲. برنامه‌ای که n را خوانده، مجموع n جمله سری زیر را نمایش می‌دهد:

$$1-\frac{1+3}{2+4}+\frac{1+3+5}{2+4+6}-\frac{1+3+5+7}{2+4+6+8}+...$$

۱۳. برنامه‌ای که عدد فرد n را خوانده خروجی زیر را نمایش می‌دهد (اگر n، 5 وارد شود، خروجی مقابل را نمایش می‌دهد):

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

۱۴. برنامه‌ای که یک رشته را دریافت کرده، اگر کاربر یکی از کلمات C# یا c# را وارد کند، پیغام "Good Choice, C# is a fine language" را نمایش می‌دهد. اما اگر کاربر یکی از کلمات VB یا vb را وارد کند، عبارت "VB: OOP, multithreading" را نمایش خواهد داد. در غیر این صورت عبارت "well…goodluck with that and more" را نمایش می‌دهد (در این برنامه از switch استفاده شود).

۱۵. برنامه‌ای که مقادیر درجه سیلسیوس (سانتی‌گراد) مقادیر ۰ تا ۹۹ فارنهایت را نمایش می‌دهد. این برنامه هر ۲ مقدار را در یک سطر نمایش خواهد داد.

۱۶. برنامه‌ای که عددی را خوانده، اعداد مضرب ۳ تا عدد خوانده شده را نمایش می‌دهد.

۱۷. برنامه‌ای که عددی را خوانده اعداد فرد از عدد خوانده شده تا یک را نمایش می‌دهد.

۱۸. برنامه‌ای که یک عدد، یک کارکتر و عدد دیگری را دریافت می‌کند. اگر کارکتر ورودی یکی از حروف A، a و + باشد، حاصل جمع عدد را نمایش می‌دهد. اگر کارکتر ورودی یکی از حروف S، s یا – باشد، حاصل تفریق دو عدد را نمایش خواهد داد. اما، اگر حرف وارد شده M، m یا * باشد، حاصل ضرب دو عدد را نمایش خواهد داد. وگرنه، اگر حرف وارد شده D، d یا / باشد، حاصل تقسیم دو عدد را نمایش می‌دهد. اگر کارکتر وارد شده، هیچ یک از حروف بیان شده نباشد، "Error!!!" را نمایش می‌دهد (این برنامه با switch نوشته شده است).

۱۹. برنامه‌ای که عددی را خوانده، میانگین ارقام فرد و زوج را جداگانه نمایش می‌دهد.

۲۰. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

```
2    4    6    8    10   12
6    8    10   12   14   16
```

```
10   12   14   16   18   20
14   16   18   20   22   24
```

۲۱. برنامه‌ای که مختصات نقطه‌ای مانند (x , y) و r (شعاع دایره) را دریافت کرده، تعیین می‌کند نقطه‌ای مانند (a و b) چه وضعیتی نسبت به دایره دارد.

۲۲. برنامه‌ای که n را خوانده، حاصل عبارت $\sum_{i=1}^{n} x^i$ را نمایش می‌دهد.

۲۳. برنامه‌ای که n را خوانده، حاصل عبارت $\sum_{i=1}^{n} x^n$ را نمایش می‌دهد.

۲۴. برنامه‌ای که x و y را خوانده، سپس تعیین می‌کند (x , y) درچه مکانی از دستگاه مختصات هستند (روی محور، مبدا و یا یکی از ناحیه‌های (ربع‌های) دستگاه). اگر x = 0 و y = 0 باشد، نقطه در مبدا مختصات است. اگر x = 0 و y != 0 باشد، نقطه روی محور y است، اگر y = 0 و x != 0 باشد، نقطه روی محور xها قرار دارد، وگرنه، نقطه روی یکی از ناحیه‌ها قرار دارد.

۲۵. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

```
1    3    5    7    9    13
5    7    9    11   13   17
9    11   13   15   17   21
13   15   17   19   21   23
```

۲۶. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

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

۲۷. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

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

۲۸. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

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

۲۹. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

۳۰. برنامه‌ای که خروجی زیر را نمایش می‌دهد:

100	200	300	400	500	600
150	250	350	450	550	650
200	300	400	500	600	700
250	350	450	550	650	750
300	400	500	600	700	800
350	450	550	650	750	850
400	500	600	700	800	900
450	550	650	750	850	950
500	600	700	800	900	1000

۳۱. برنامه‌ای که x، a و n را خوانده، حاصل عبارت $F(x, a, n) = a^x + a^{2x} + \ldots + a^{nx}$ را نمایش می‌دهد.

۵ – ۲. تمرین‌ها

۱۱. برنامه‌ای بنویسید که عدد فرد n را از ورودی خوانده، مجموع سری زیر را نمایش دهد:

$$1-3+5-7+....\pm N$$

۱۲. برنامه‌ای بنویسید که y و x دو عدد صحیح را گرفته، با استفاده از عملگر ضرب x^y را محاسبه کرده، نمایش دهد.

۱۳. برنامه‌ای بنویسید که سه عدد صحیح را از ورودی خوانده، اگر a زوج باشد مجموع مکعبات آن سه عدد را نمایش می‌دهد، وگرنه، مجموع مربعات آن سه عدد را نمایش دهد.

۱٤. برنامه‌ای بنویسید که تعدادی عدد را خوانده و تشخیص دهد عدد دارای رقم 0 است یا خیر. برای خروج از برنامه کاربر باید ۱- را وارد کند.

۱۵. برنامه‌ای بنویسید که خروجی زیر را نمایش دهد:

1	64
2	16
4	8
8	4
16	2
64	1

۱٦. برنامه‌ای بنویسید که شماره هزینه تعدادی خانوار را به روز خوانده، هزینه ماهیانه، سالانه آن‌ها را نمایش دهد. سپس، تعیین کند کدام خانوار کمترین هزینه وکدام خانوار بیشترین هزینه را دارد. برای خروج از برنامه به جای شماره خانوار عدد ۹۹- وارد شود.

۱۷. برنامه‌ای بنویسید که عددی را خوانده حاصل ضرب ارقام غیر صفر آن را نمایش دهد.

۱۸. برنامه‌ای بنویسید که تمام اعداد سه رقمی‌که حاصل ضرب ارقام آن‌ها بزرگتر از نصف خودشان است را نمایش دهد.

۱۹. برنامه‌ای بنویسید که n را از ورودی خوانده، سپس، ساعات اضافه کار n کارمند را خوانده و سه کارمندی که کمترین اضافه کار را دارند، چاپ نماید (بدون استفاده از آرایه و مرتب کردن).

۲۰. یک شرکت بیمه، به بیمه گذاران خود سود سالانه می‌دهد. میزان سود سالانه برابر با ۴/۵٪ میزان سرمایه گذاری‌شده می‌باشد. برنامه‌ای بنویسید که سرمایه بیمه گذار را دریافت کند و سود بیمه گذار را برای ۸ سال

مختلف حساب نماید (بدون پرداخت سود به بیمه گذار، یعنی سود به سرمایه سالانه اضافه گردد). سرمایه بیمه گذار را در پایان ۸ سال، نمایش دهد.

۲۱. برنامه‌ای بنویسید که سن و جنسیت تعدادی افراد را خوانده، تعداد و درصد مردانی که سن آنها بین ۱۸ تا ۲۰ است را محاسبه می‌کند و نمایش دهد. اگر جنسیت M یا m وارد گردید، این فرد مرد است (برای خاتمه برنامه به جای سن ۱- وارد می‌شود).

۲۲. شرکت مخابرات ایران، برای مکالمات راه دور از نرخ‌های زیر استفاده می‌کند:

- تمام مکالماتی که بین ۲۳ تا ۸ صبح انجام می‌شود، نرخ مکالمه ۵۰٪ محاسبه می‌شود.
- برای مکالماتی که در روزهای تعطیل (روز ۷) انجام شود، نرخ مکالمه ۷۵٪ محاسبه می‌شود.
- برای بقیه ساعات روزهای دیگر مکالمه به طور کامل حساب شود.
- نرخ هر پالس ۴۶ ریال می‌باشد.
- به کلیه مکالمات ۴٪ مالیات تعلق می‌گیرد.

برنامه‌ای بنویسید که برای تعدادی مشترک، ساعات شروع مکالمه (از صفر تا ۲۳ ساعت)، تعداد پالس های مکالمه، روز انجام مکالمه را دریافت کند، مبلغ ناخالص مکالمه، مالیات، مبلغ خالص، مجموع مالیات و مجموع مبلغ خالص را محاسبه و نمایش دهد (برای خاتمه کار به جای ساعت شروع ۹۹- را وارد نماید).

۲۳. نرخ هر متر مکعب آب با توجه به نوع مصرف از قبیل مصارف خانگی، تجاری یا صنعتی فرق می‌کند. برنامه‌ای بنویسید که میزان مصرف آب به متر مکعب و نوع مصرف تعدادی مشترک را گرفته مبلغ قبض آب آنها را محاسبه و چاپ کند. برای محاسبه صورتحساب آب به روش زیر عمل نماید:

- اگر نوع مصرف حرف H باشد (مصرف خانگی)، به ازای هر ۱۰۰ متر مکعب مصرف ۵۰۰ ریال دریافت می‌شود.
- اگر نوع مصرف حرف I باشد (مصرف صنعتی تا چهارمیلیون متر مکعب)، به ازای هر ۱۰۰۰ متر مکعب آب، ۷۵۰ ریال و به ازای هر متر مکعب بیش از آن ۰/۰۰۰۲۵ به مبلغ قبض اضافه خواهد شد.
- اگر نوع مصرف F باشد (مصرف تجاری)، در صورتی مصرف آب تا دو میلیون متر مکعب باشد، به ازای هر ۱۵۰ متر مکعب ۶۰۰ ریال و به ازای هر متر مکعب بیش از دو میلیون ۰/۰۰۰۰۴ به مبلغ قبض اضافه خواهد شد.

نمره	نوع دانشجو
۱۸ تا ۲۰	ممتاز
۱۵ تا ۱۸	خوب

۱۲ تا ۱۵	معمولی
زیر ۱۲	ضعیف

برای خاتمه برنامه، کاربر به جای مصرف آب عدد منفی وارد می‌نماید.

۲٤. برنامه‌ای بنویسید که تعدادی نمره امتحانی دانشجو بین ۰ تا ۲۰ را از ورودی خوانده(اگر کاربر نمره زیر صفر وارد نماید، برنامه خاتمه یابد).

سپس بر اساس جدول مقابل تعداد و درصد دانشجویان ممتاز، خوب، معمولی و ضعیف را نمایش دهد:

۲۵. برنامه‌ای بنویسید که تمام اعداد چهار رقمی را چاپ کند که مجموع ارقام آن‌ها عددی فرد است.

۲٦. برنامه‌ای بنویسید که تمام اعداد چهار رقمی را نمایش دهد که حاصل ضرب ارقام آن‌ها عددی تام است(عددی تام است، که مجموع مضرب‌های اعداد کوچکتر از خودش برابر خودش می‌باشد).

۲۷. برنامه‌ای بنویسید که تعداد صفرهای کل اعداد چهار رقمی را شمارد.

۲۸. برنامه‌ای بنویسید که سه عدد را خوانده، اعداد بین اعداد اول و دوم که مجموع ارقام آن‌ها برابر عدد سوم باشد را نمایش دهد. به عنوان مثال، اگر کاربر ۵، ۱۰۰۰ و ۱۵ را وارد کند، برنامه باید تمام اعداد از ۵ تا ۱۰۰۰ که مجموع ارقام آن‌ها برابر ۱۵ باشد را نمایش دهد. (مثل ۷۸، ۸۷، ۹۶، ۶۹ و ...)

۲۹. برنامه‌ای بنویسید که دو عدد را خوانده، اعداد اولی که بین این دو عدد قرار دارند را نمایش دهد.

۳۰. فرض کنید بخواهید از بانک ۱۰۰۰۰۰۰۰ ریال وام با بهره ۱۵٪ با مدت بازپرداخت ۱۸ ماهه دریافت کنید، بهره وام به صورت زیر محاسبه می‌شود:

$$\text{بهره وام} = \frac{\text{نرخ بهره} \times \text{تعداد اقساط}}{12} + \frac{\text{مبلغ پرداختنی وام}}{12} = \frac{۱۵ / ۰ \times ۱۸ \times ۱۰۰۰۰۰۰۰}{12} = ۲۲۵۰۰۰۰$$

مبلغ بهره (۱۲۲۵۰۰۰۰)ریال به متقاضی پرداخت می‌گردد. حال چنانچه متقاضی ۱۰۰۰۰۰۰۰ریال نیاز داشته باشد، چقدر وام باید به او پرداخت شود. برنامه‌ای که مبلغ مورد نیاز متقاضی، تعداد اقساط و مبلغ بهره را دریافت می‌کند، سپس وامی که باید به متقاضی پرداخت شود و قسط هر ماه را چاپ می‌کند:

توضیح: مبلغ وام پرداختنی به صورت زیر محاسبه می‌گردد:

$$\text{مبلغ متقاضی} = \frac{\text{درصد نرخ بهره} \times \text{تعداد اقساط} \times \text{مبلغ وام پرداختنی}}{12} - \text{مبلغ وام پرداختی} = \text{مبلغ متقاضی} = \text{مبلغ بهره وام} - \text{مبلغ وام پرداختی}$$

$$\text{مبلغ کل وام} = \text{مبلغ متقاضی} \times ۱۲ = \text{درصد نرخ بهره} \times \text{تعداد اقساط} \times \text{مبلغ وام} - \frac{\text{مبلغ متقاضی} \times ۱۲}{(\text{نرخ بهره} \times \text{تعداد اقساط} - ۱۲)} \times ۱۲ \text{مبلغ وام}$$

تذکر: برنامه تا هر زمان که کاربر بخواهد ادامه می‌یابد.

فصل ۳

متدها و پیاده‌سازی آن‌ها

برنامه‌های کامپیوتری که در دنیای واقعی مورد استفاده قرار می‌گیرند، خیلی بزرگتر و پیچیده‌تر از برنامه‌هایی هستند که در فصل‌های ۱ و ۲ آمده‌اند. تجربه نشان داده است که بهترین راه توسعه، نگهداری و استفاده از برنامه‌ها شکستن آن‌ها به قسمت‌های کوچکتر می‌باشد. یعنی برنامه‌نویس برنامه را به مجموعه‌ای از #Cفعالیت‌هایی تقسیم می‌کند که هر فعالیت وظیفه خاصی دارد. این فعالیت‌ها **ماژول**[11] نام دارند. ماژول‌ها در همان **متدها**[2] و **کلاس‌ها**[3] نام دارند. در این فصل با متدها و چگونگی ایجاد آن‌ها آشنا خواهید شد و در ادامه مفهوم کلاس و چگونگی پیاده‌سازی آن‌ها را می‌آموزیم.

همان‌طور که بیان گردید، متدها به برنامه نویس این امکان را می‌دهند تا بتواند برنامه‌ها را به بخش‌های کوچکتری تقسیم کند. این تقسیم بندی برنامه‌ها به بخش‌های کوچکتر دارای مزایای زیر است:

۱. خوانایی برنامه افزایش می‌یابد و همچنین، تست و رفع اشکال برنامه‌ها آسان‌تر خواهد شد. چون برنامه به بخش‌های کوچکتری تقسیم می‌شوند.

۲. می‌توان از متدهای نوشته شده استفاده مجدد[4] نمود. نمونه‌ای از استفاده مجدد متدها را در فصل‌های اول و دوم دیدید (متد Convert.ToInt32()).

۳. متدها، امکان کار گروهی را فراهم می‌کنند. زیرا، پس از این که برنامه را به قسمت‌های کوچکتر تقسیم شد، هر یک از اعضای گروه می‌توانند برروی یک بخش از برنامه کار کنند.

۴. از متدهای نوشته شده دیگران می‌توان استفاده کرد.

۵. متدها را می‌توان در فضای نام قرار داده و دربرنامه‌های دیگر از آن‌ها استفاده نمود.

---

[11]. Module [2]. Methods [3].Classes [4].Reuse

۱ – ۳. انواع متدها

دو نوع متد وجود دارد:#Cدر زبان برنامه‌نویسی

۱.متدهای کتابخانه‌ای

۲.متدهایی که برنامه‌نویس می‌نویسد.

۱ – ۱ – ۳. متدهای کتابخانه‌ای

متد کتابخانه‌ای متدهایی هستند که همراه کامپایلر وجود دارند. این متدها، **متدهای عمومی** نام دارنـد، زیـرا ReadKey() ،Write() کاربردهـای زیـادی دارنـد. برخـی از متـدهای کتابخانـه‌ای را تـاکنون دیدیـد نظیـر WriteLine() و غیره ToChar() ،ToInt64() ،ToInt32() قرار دارند یا متدهای Console که در فضای نام قرار دارند. این متدها را Math قرار دارند. متدهای ریاضی و محاسباتی در فضای نام Convertکه در فضای نام در جدول ۱ – ۳ می‌بینید. برای استفاده از این متدها کافی است به صورت زیر عمل کنید:

(لیست پارامترها) نام متد . نام کلاس;

جدول ۱ – ۳ متدهای ریاضی تعریف شده در کلاس Math.

نام متد	الگوی متد	هدف	مثال
Abs(x)	int Abs (int)	را برمی‌گرداندxقدر مطلق	Abs(-5) را ۵، برمی‌گرداند
Acos(x)	double Acos(double)	به رادیان را برمی‌گرداند-xآرک کسینوس	Acos(0.2)، مقدار ۱,۳٦۹٤٤ را برمی‌گرداند
Asin(x)	double Asin(double)	به رادیان را برمی‌گرداند-xآرک سینوس	Asin(0.2)، مقدار ۰,۲۰۱۳۵۸ را برمی‌گرداند
Atan(x)	double Atan(double)	به رادیان را برمی‌گرداند-xآرک تانژانت	Atan(0.2)، مقدار ۰,۱۹۷۳۹٦ را برمی‌گرداند
Ceil(x)	double Ceil(double)	کوچکترین عدد صحیح بزرگتر یا را برمی‌گرداندxمساوی	Ceil(13.2)، مقدار ١٤ را برمی‌گرداند
Cos(x)	double Cos(double)	به رادیان را برمی‌گرداندxکسینوس	Cos(2.0)، مقدار -۰,٤١٦١٤٧ رابرمی‌گرداند
Exp(x)	double Exp(double)	را برمی‌گرداندxتابع نمایی	Exp(0.2)، مقدار

			۷,۳۸۹۰۶ را برمی‌گرداند
Sin(x)	double Sin(double)	به رادیان را برمی‌گرداند x سینوس	مقدار Sin(0.2)، ۰,۹۰۹۲۹۷ را برمی‌گرداند
Floor(x)	double Floor(double)	بزرگترین مقدار صحیح کوچکتر یا مساوی x را برمی‌گرداند	مقدار Floor(12.14)، ۱۲ را برمی‌گرداند
Sqrt(x)	double Sqrt(double)	جذر x را برمی‌گرداند	مقدار Sqrt(4.0)، ۲ را برمی‌گرداند
Pow(x, n)	double Pow(double)	x به توان n را برمی‌گرداند	مقدار Pow(2.0,5.0)، ۳۲,۰ را برمی‌گرداند
Tan(x)	double Tan(double, double)	به رادیان را برمی‌گرداند x تانژانت	مقدار Tan(2)، ۲,۱۸۵۰۴- را برمی‌گرداند
Log(x)	double Log(double)	لگاریتم طبیعی x را برمی‌گرداند	مقدار Log(2)، ۰,۶۹۳۱۴۷ را برمی‌گرداند
$Log_{10}(x)$	double Log_{10}(double)	لگاریتم عمومی x در پایه 10، را برمی‌گرداند	مقدار $Log_{10}(2)$، ۰,۳۰۱۰۳ را برمی‌گرداند

پارامترها، اطلاعاتی هستند که باید به متد ارسال شوند. به عنوان مثال، دستورات زیر را ببینید:

Math.Abs (-7.5) ⇨ 7.5
Math.Ceiling (19.2) ⇨ 10.0
Math.Max (12.3,112.7) ⇨ 112.7
Math.Sqrt (81.0) ⇨ 9.0

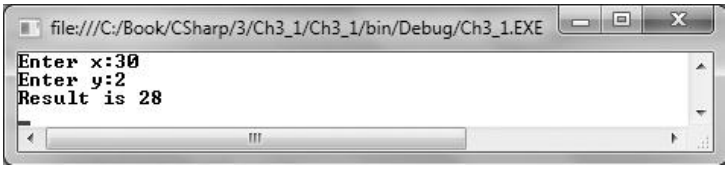
مثال ۱ – ۳. برنامه‌ای که x و y را خوانده، حاصل $\sqrt{|x-y|^{|y|}}$ را محاسبه کرده، نمایش می‌دهد (هدف این برنامه آشنایی با متدهای Sqrt()، Pow() و Abs() در کلاس Math است).

متغیر	هدف
x	عدد ورودی
y	عدد ورودی

```
using System;
namespace Ch3_1 {
 class Program  {
  static void Main(string[] args)  {
   Console.Write("Enter x:");
   double x=Convert.ToDouble(Console.ReadLine());
   Console.Write("Enter y:");
   double y = Convert.ToDouble(Console.ReadLine());
   Console.WriteLine("Result is {0}", Math.Sqrt(Math.Pow
          (Math.Abs(x - y), Math.Abs(y))));
   Console.ReadKey();
  }
 }
```

خروجی:

```
}
```



۲ – ۱ – ۳. متدهایی که برنامه‌نویس می‌نویسد

زبان C# دارای متدها و کلاس‌های زیادی است. اما، این متدها و کلاس‌ها پاسخ‌گوی تمام نیازهای برنامه نویس نیستند. بنابراین برنامه‌نویس باید بتواند متدها و کلاس‌های جدیدی بنویسد و از آن‌ها در برنامه‌هایش استفاده کند. در این بخش روش نوشتن متدها را می‌آموزیم و در فصل‌های بعد روش ایجاد کلاس‌های جدید را می‌بینید. برای استفاده از متدها در برنامه باید دو کار انجام شود:

۱. نوشتن متد ۲. صدا زدن[۱۲](فراخوانی) متد

نوشتن متد

برای نوشتن متد باید سطح دسترسی، نوع، نام و لیست پارامترهای متد را تعیین کرد. ساختار تعریف متد به صورت زیر می‌باشد:

```
سطح دسترسی  نوع متد  نام متد (لیست پارامترها)
{
    بدنه متد;
}
```

سطح دسترسی متد می‌تواند private یا public باشد. چنانچه سطح دسترسی متد private باشد، در خارج از این کلاس نمی‌توان از متد استفاده نمود. ولی اگر سطح دسترسی متد public باشد، در کلاس‌های دیگر نیز می‌توان از این متد استفاده نمود. چنانچه سطح دسترسی متد ذکر نشود، private در نظر گرفته می‌شود. در این فصل سطح دسترسی متدها را private در نظر می‌گیریم. بنابراین، در سطح دسترسی متدها چیزی ذکر نمی‌کنیم.

[12] .Call

نوع متد، نوع مقداری را تعیین می‌کند که متد برگشت می‌دهد. این نوع می‌تواند یکی از انواع اولیه تعریف شده در C# نظیر int، double و غیره یا انواعی باشد که برنامه نویس تعریف می‌کند. متدها از لحاظ مقداری را که برمی‌گردانند به سه نوع تقسیم می‌شوند:

۱. **متدهایی که هیچ مقداری را برنمی‌گردانند.** نوع این متدها باید void تعیین شود (مثال ۱ – ۳ را ببینید).

۲. **متدهایی که فقط یک مقدار را برمی‌گردانند.** نوع مقداری که این متدها برمی‌گردانند، قبل از نام متد قرار می‌گیرد. برخی از این متدها عبارت‌اند از:

۱. متدی که تعیین می‌کند عددی اول است یا خیر. این متد یک مقدار true (در صورت اول بودن عدد) یا false (در صورت اول نبودن عدد) برمی‌گرداند.

۲. متدی که کوچکترین مقدار بین دو عدد را برمی‌گرداند. نوع مقداری که متد برمی‌گرداند به نوع اعداد بستگی دارد.

۳. متدی که یک کد اسکی را به کارکتر تبدیل کرده، برمی‌گرداند. نوع این متد char خواهد بود.

۴. و غیره.

متدهایی که یک مقدار را برگشت می‌دهند، این عمل را از طریق نام متد با دستور return به صورت‌های زیر انجام می‌دهند:

return مقدار;

return (عبارت);

ساختار اول، یک مقدار را برگشت می‌دهد. به عنوان مثال، دستورات زیر را ببینید:

```
return true;
return    12.5;
```

دستور اول، مقدار true، اما دستور دوم مقدار12.5 را برمی‌گرداند.

ساختار دوم نتیجه ارزیابی یک عبارت را برمی‌گرداند. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
return (a == b && b == c);
return ((a < b) ? a : b);
```

دستور اول، اگر مقادیر a، b و c برابر باشند، true وگرنه، false را برمی‌گرداند. اما، دستور دوم کوچکترین مقدار بین a و b را برگشت خواهد داد.

۳. **متدهایی که چندین مقدار را برگشت می‌دهند.** این متدها، مقادیر را از طریق پارامترها برگشت می‌دهند. در ادامه چگونگی برگشت چندین مقدار از طریق پارامترها را خواهید دید.

- **نام متد**، برای دسترسی به متد به کار می‌رود و از قوانین نامگذاری شناسه‌ها و متغیرها پیروی می‌کند.
- **پارامترهای متد**، اطلاعاتی هستند که در هنگام صدازدن (فراخوانی) متدها باید به آن‌ها ارسال گردند.

متدها می‌توانند از لحاظ تعداد پارامترهایی که می‌پذیرند، به سه دسته تقسیم شوند:

۱. **متدهایی که هیچ پارامتری ندارند.** این متدها برای چاپ پیغام مشخصی به کار می‌روند.

۲. **متدها ممکن است یک پارامتر داشته باشند.** در این صورت باید نوع و نام پارامتر در داخل پرانتز تعیین گردد. نمونه‌هایی از این متدها در زیر آمده‌اند:

- متدی که عددی را دریافت کرده، تعیین می‌کند اول است یا خیر. پارامتر این متد، عددی می‌باشد که باید تعیین شود اول است یا خیر.
- متدی که رشته‌ای را دریافت کرده، تعداد ارقام ۰ تا ۹ آن را می‌شمارد. پارامتر این متد، رشته‌ای که باید تعداد ارقام آن شمارش شود، می‌باشد.
- متدی که عددی را دریافت کرده، حاصل ضرب ارقام آن را برمی‌گرداند. پارامتر این متد، عددی که حاصل ضرب ارقام آن باید برگردانده شود، می‌باشد.

۳. **متدها ممکن است بیش از یک پارامتر داشته باشند.** در این صورت، نوع و نام پارامترها در داخل پرانتز قرار می‌گیرند. پارامترها با کاما از هم جدا می‌گردند. برخی از این متدها را در زیر می‌بینید:

- متدی که دو عدد را دریافت کرده، مجموع اعداد تام بین این دو عدد را برمی‌گرداند. در این متد، دو عدد به عنوان پارامتر می‌باشند.
- متدی که سه عدد را به عنوان پارامتر دریافت کرده، آن‌ها را مرتب می‌کند.
- متدی که دو مقدار را دریافت کرده، اولین مقدار را به توان دومین مقدار می‌رساند.

۷ – ۳. مسائل حل شده

مثال ۱ – ۳. برنامه‌ای که عددی را خوانده، حاصل ضرب ارقام بالای ۵ را از طریق یک متد محاسبه نموده، نمایش می‌دهد.

توضیح: تابع بازگشتی این برنامه به صورت زیر تعریف می‌شود:

$$mulG5(n) = \begin{cases} 1 & \text{اگر } n == 0 \\ mulG5(n/10) & \text{وگرنه ، اگر } n \% 10 > 5 \\ n \% 10 * mulG5(n/10) & \text{وگرنه} \end{cases}$$

متد	متغیر	هدف

mulG5	n	پارامتر ورودی(عدد)
Main	n	عدد خوانده شده

```
using System;
namespace SolveCh3_1 {
 class Program  {
  static long mulG5(int n){
    if (n == 0) return 1;
    else if (n % 10 <= 5) return mulG5(n / 10);
    else return (n % 10 * mulG5(n / 10));
  }
  static void Main(string[] args)  {
    Console.Write("Enter n: ");
    int n= Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Result is {0}", mulG5(n));
    Console.ReadKey();
  }
 }
}
```

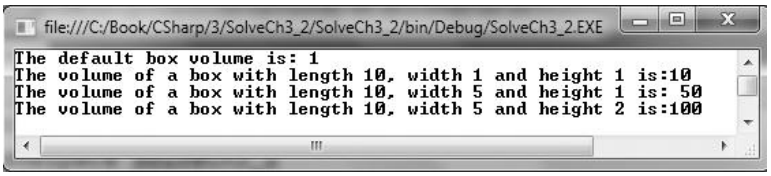
خروجی:

مثال ۲ – ۳. برنامه‌ای که اطلاعات جعبه‌ای از قبیل طول، عرض و ارتفاع را خوانده با استفاده از یک متد، حجم جعبه را نمایش می‌دهد (هدف این برنامه آشنایی با پارامترهای با مقدار پیش‌فرض است).

متد	متغیر	هدف
boxVolume	length width height	طول جعبه عرض جعبه ارتفاع جعبه

```
using System;
namespace SolveCh3_2 {
 class Program  {
  static int boxVolume(int
length=1,int width=1,int height=1){
    return length * width * height;
  }
  static void Main(string[] args)  {
   Console.WriteLine("The default box volume is: {0}",
                     boxVolume());
   Console.WriteLine("The volume of a box with length 10,
           width 1 and height 1 is:{0}", boxVolume(10));
   Console.WriteLine("The volume of a box with length 10,
           with 5 and height 1 is: {0}", boxVolume(10, 5));
   Console.WriteLine("The volume of a box with length 10,
           width 5 and height 2 is:{0}",boxVolume(10,5, 2));
   Console.ReadKey();
  }
 }
}
```

خروجی:



مثال ۳ – ۳. برنامه‌ای که x را خوانده، با متدهای همنام مقدار عبارت $F(x) = x^2 - 2^x + 4$ را محاسبه کرده نمایش می‌دهد.

متد	متغیر	هدف
F	x	پارامترصحیح یا اعشاری
Main	x1 x2	عدد صحیح ورودی عدد اعشاری ورودی

```
using System;
namespace SolveCh3_3 {
 class Program  {
  static float F(float x)  {
    return(x*x - 2 * x + 4);
  }
  static int F(int x)  {
     return ( x * x - 2 * x + 4);
  }
  static void Main(string[] args)  {
     Console.Write("Enter a integer number:");
     int x1 = Convert.ToInt32(Console.ReadLine());
     Console.WriteLine("F({0}):{1}", x1, F(x1));
     Console.Write("Enter a float number:");
     float x2 = Convert.ToSingle(Console.ReadLine());
     Console.WriteLine("F({0}):{1}", x2, F(x2));
     Console.ReadKey();
  }
 }
}
```

خروجی:

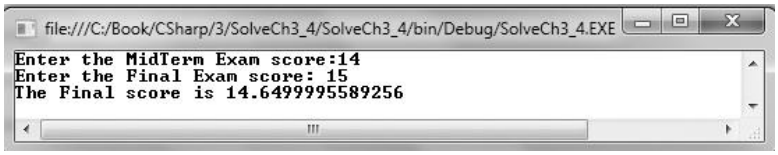
مثال ۴ – ۳. برنامه‌ای که نمره میان‌ترم و پایان‌ترم دانشجویی را گرفته با استفاده از یک متد نمره نهایی دانشجو را محاسبه نموده، نمایش می‌دهد (نمره نهایی دانشجو برابر با 0/65 * نمره پایان ترم + 0/35 * نمره میان‌ترم است).

متد	متغیر	هدف
CalcScore	midTermپارامتر	نمره میان ترم
	finalپارامتر	نمره نهایی
	MIDTERM_WEIGHT ثابت	تاثیر نمره میان‌ترم در نمره کل

	FINAL_WEIGHT ثابت	تاثیر نمره پایان ترم در نمره کل
Main	midTermExam finalExam finalScore	نمره میان ترم نمره پایان ترم نمره کل

```
using System;
namespace SolveCh3_4 {
 class Program  {
  static double CalcScore(double midTerm, double final) {
    const double MIDTERM_WEIGHT = 0.35f;
    const double FINAL_WEIGHT = 0.65f;
    return(MIDTERM_WEIGHT * midTerm + FINAL_WEIGHT * final);
  }
  static void Main(string[] args)  {
    Console.Write("Enter the MidTerm Exam score:");
    double midTermExam=Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter the Final Exam score: ");
    double finalExam = Convert.ToDouble(Console.ReadLine());
    double finalScore = CalcScore(midTermExam, finalExam);
    Console.WriteLine("The Final score is {0}", finalScore);
    Console.ReadKey();
  }
 }
}
```

خروجی:



مثال ۵ – ۳. برنامه‌ای که عددی را خوانده، اگر عدد خوانده شده فرد باشد، مجموع و حاصل ضرب عدد فرد تا آن عدد، وگرنه مجموع و حاصل ضرب اعداد زوج تا آن عدد را محاسبه کرده، نمایش می‌دهد.

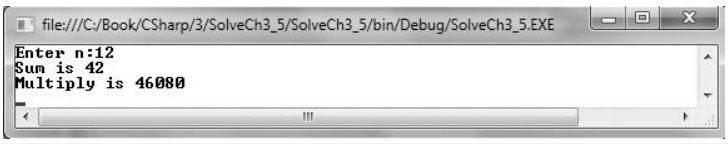
توضیح: متد ()sum، برای محاسبه مجموع اعداد زوج یا فرد از یک مقدارتا پارامترش به کار می‌رود. این متد به صورت معمولی نوشته شده است. اما متد mul به صورت بازگشتی پیاده‌سازی شده است که حاصل ضرب اعداد زوج یا فرد تا پارامترش را برمی گرداند.

متد	متغیر	هدف
همه متدها	n	عدد خوانده شده یا پارامتر متدها

Sum	i s	تا ۱ با گام کاهش ۲شمارنده از مجموع اعداد تولید شده فرد یا زوج

```
using System;
namespace SolveCh3_5 {
 class Program {
  static int sum(int n){
   int s = 0;
   for(int i=n; i>=1; i-= 2)
         s += i;
  return s;
  }
  static long mul(int n)  {
    if (n < 1) return 1;
    else return (n * mul(n - 2));
  }
  static void Main(string[] args)  {
     Console.Write("Enter n:");
     int n = Convert.ToInt32(Console.ReadLine());
     Console.WriteLine("Sum is {0}", sum(n));
     Console.WriteLine("Multiply is {0}", mul(n));
     Console.ReadKey();
  }
 }
}
```

خروجی:



مثال ۱۷– ۳. برنامه‌ای که با یک تابع، منویی با گزینه‌های a تا f را نمایش می‌دهد و سپس کاربر با ورود یکی از حروف، مجموع یکی از سری‌های زیر را محاسبه می‌کند:

$$a)\ x-\frac{x^2}{2!}+\frac{x^3}{3!}-\frac{x^4}{4!}+\ldots\pm\frac{x^n}{n!} \qquad b)\ x-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots\pm\frac{x^n}{n!}$$

$$c)\ \frac{x^2}{2!}-\frac{x^4}{4!}+\frac{x^6}{6!}-\ldots\pm\frac{x^n}{n!}$$

$$d)\ \frac{1}{1!}-\frac{1+2}{2!}+\frac{1+2+3}{3!}-\frac{1+2+3+4}{4!}+\frac{1+2+3+4+5}{5!}-\ldots\pm\frac{1+2+3+\ldots+n}{n!}$$

$$e)\ \frac{1}{1!}-\frac{2}{2!}+\frac{1+3}{3!}-\frac{2+4}{4!}+\frac{1+3+5}{5!}-\frac{2+4+6}{6!}+\ldots \qquad f)\ exit$$

توضیح: برای حل این مسئله توابع زیر نوشته شده‌اند:

۱. متدی که از کاربر یکی از کارکترهای a تا f را می‌پذیرد. چنانچه کاربر کارکتری غیر از این کارکترها را وارد کند، مجدداً از کاربر این کارکترها را درخواست می‌کند. این تابع، input نام دارد.

۲. متدی که n را دریافت کرده، فاکتوریل n را محاسبه می‌کند.

۳. متدی که x و n را دریافت کرده، x به توان n را برمی‌گرداند.

☒**متد sum()**، سه پارامتر start (عدد شروع)، end (عدد پایان) و step (گام افزایش) را دریافت کرده، اعداد از start تا end را با گام افزایش step جمع کرده، برمی‌گرداند. به عنوان مثال، اگر این تابع به صورت‌های زیر فراخوانی گردد، نتایج زیر را برمی‌گرداند:

sum (1, 5, 1) = 1 + 2 + 3 + 4 + 5 = 15
sum (1, 7, 2) = 1 + 3 + 5 + 7 = 16
sum (0, 10, 2) = 0 + 2 + 4 + 6 + 8 + 10 = 30

☒**متد sumA_C()**، چهار پارامتر x (عدد خوانده شده)، start (شروع توان یا فاکتوریل)، end (پایان توان یا فاکتوریل) و step (گام افزایش توان یا فاکتوریل) را دریافت کرده، یکی از سری‌های a، b یا c را محاسبه می‌کند.

متد	متغیر	هدف
input	ch	کارکتری که از ورودی دریافت می‌شود
sum	پارامتر start پارامتر end step i s	عدد شروع عدد پایان گام افزایش شمارنده حلقه مجموع اعداد از start تا end با گام افزایش step
pow	پارامتر x پارامتر n i p	عددی که باید به توان n برسد توان عدد x شمارنده x به توان n
fact	پارامتر n i f	عددی که باید فاکتوریل آن حساب شود شمارنده فاکتوریل عدد n
sumA_C	پارامتر x پارامتر start پارامتر end	عدد خوانده شده توان یا فاکتوریل شروع توان یا فاکتوریل پایان

	پارامتر step sign s i	گام افزایش توان یا فاکتوریل علامت (یکی در میان مثبت و منفی) مجموع سری شمارنده
sumD	پارامتر n s sign i	عدد خوانده شده مجموع علامت شمارنده
sumE	پارامتر n s sign i	عدد خوانده شده مجموع علامت شمارنده
Main	x n s ch	عدد خوانده شده عدد خوانده شده مجموع کارکتر خروجی input

☒ **متد sumD()**، پارامتر n را دریافت کرده، مجموع n جمله سری d را محاسبه می‌کند و نتیجه را برمی‌گرداند.

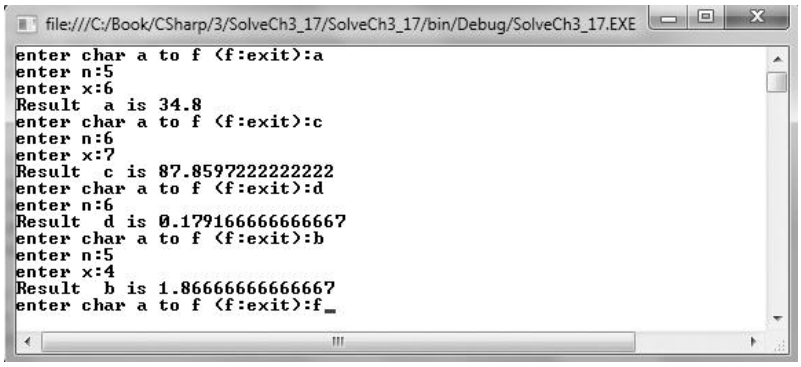
☒ **متد sumE()**، پارامتر n را دریافت کرده، مجموع n جمله سری e را برمی‌گرداند.

اگر کاربر کارکتر f را وارد کند، برنامه خاتمه می‌یابد.

```
using System;
namespace SolveCh3_17 {
 class Program  {
  static  char input()   {
    char ch;
    while( true )    {
      Console.Write("enter char a to f (f:exit):");
      ch =Convert.ToChar(Console.Read());
      Console.ReadLine();
      if ( ch >= 'a' && ch <= 'f') break;
    }
     return ch;
  }
  static int sum(int start, int end, int step)  {
    int s = 0;
    for(int i=start ; i <= end; i+= step)   {
     s += i;
    }
    return s;
  }
  static double  pow(double x, int n)  {
    double p = 1.0;
    for(int i=1; i <= n; i++)    {
      p *= x;
    }
    return p;
  }
  static long  fact(int n)  {
    long f = 1;
    for(int i=2; i<=n; i++)    {
     f *= i;
    }
    return f;
```

```
  }
  static double sumA_C(double x,int start,int end,int step){
    double s = 0, sign = 1;
    for ( int  i = start; i<= end; i += step)  {
      s += (double)sign * pow( x, i) /fact(i);
      sign = -sign;
    }
    return s;
  }
  static double sumD(int n)  {
   double s = 0, sign = 1;
    for ( int  i = 1; i<= n; i ++)  {
      s += (double)sign * sum (1, i, 1) /fact(i);
      sign = -sign;
    }
    return s;
  }
  static double sumE(int n)  {
    double s = 0, sign = 1;
    for ( int  i = 1; i <= n; i ++)  {
      s += (double)sign * sum (i % 2, i, 2) /fact(i);
      sign = -sign;
    }
    return s;
  }
  static void Main(string[] args) {
    double x = 0, s = 0;
    int n;
    for (; ; )  {
      char ch = input();
      if (ch == 'f') break;
      Console.Write("enter n:");
      n = Convert.ToInt32(Console.ReadLine());
      if (ch == 'a' || ch == 'b' || ch == 'c')  {
        Console.Write("enter x:");
        x = Convert.ToInt32(Console.ReadLine());
        if (ch == 'a') s = sumA_C(x, 1, n, 1);
        else if (ch == 'b') s = sumA_C(x, 1, n, 2);
        else s = sumA_C(x, 2, n, 2);
      }
      else if (ch == 'd') s = sumD(n);
      else s = sumE(n);
      Console.Write("Result  {0} is {1}", ch, s);
    }
  }
 }
}
```

خروجی:


file:///C:/Book/CSharp/3/SolveCh3_17/SolveCh3_17/bin/Debug/SolveCh3_17.EXE
enter char a to f (f:exit):a
enter n:5
enter x:6
Result a is 34.8
enter char a to f (f:exit):c
enter n:6
enter x:7
Result c is 87.8597222222222
enter char a to f (f:exit):d
enter n:6
Result d is 0.179166666666667
enter char a to f (f:exit):b
enter n:5
enter x:4
Result b is 1.86666666666667
enter char a to f (f:exit):f

۸ – ۳. مسائل حل شده در سایت

۱. برنامه‌ای که عددی را خوانده، با استفاده از یک تابع، توان ۲ عدد خوانده شده را محاسبه می‌کند و نمایش می‌دهد.

۲. برنامه‌ای که نمره را به صورت عددی از ۰ تا ۱۰۰ خوانده، توسط تابعی مقدار حرفی معادل آن را برمی‌گرداند و نمایش می‌دهد (اگر نمره بین ۸۰ تا ۱۰۰ باشد، حرف A، اگر نمره بین ۶۰ تا ۸۰ باشد، حرف B، چنانچه نمره بین ۵۰ تا ۶۰ باشد، حرف C، وگرنه حرف F را نمایش می‌دهد).

۳. برنامه‌ای که سن فردی را به سال، ماه، روز دریافت کرده، توسط توابعی به روز، ساعت، دقیقه و ثانیه تبدیل کرده و برمی‌گرداند و نمایش می‌دهد (هر سال، ۳۶۵ روز و هر ماه ۳۰ روز است).

۴. برنامه‌ای که حقوق کارمند را خوانده، توسط تابعی مالیات برحقوق کارمند را محاسبه کرده، نمایش می‌دهد (مالیات از طریق جدول مقابل محاسبه می‌شود):

از	تا	درصدمالیات
0	483000	0
483001	600000	%10
600001	1000000	%15
1000001	2000000	%20
2000001	99999999	%30

۵. برنامه‌ای که دو عدد را خوانده، بزرگترین و کوچکترین مقسوم‌علیه مشترک آن‌ها را نمایش می‌دهد (بزرگترین مقسوم‌علیه مشترک، بزرگترین عددی است که بر a و b بخش‌پذیر باشد و کوچکترین مقسوم‌علیه مشترک، برابر است با:

بزرگترین مقسوم‌علیه / a * b = کوچکترین مقسوم‌علیه مشترک

۶. برنامه‌ای که حاصل جمع ارقام زوج، عددی را از طریق تابع بازگشتی محاسبه کرده، نمایش می‌دهد.

۷. برنامه‌ای که n را خوانده و با استفاده از یک تابع n بار کارکتر x را ۱۰ مرتبه نمایش می‌دهد.

۸. برنامه‌ای که اعداد اول بین ۱ تا ۱۰۰ را نمایش می‌دهد. برای تعیین این که عدد اول است یا خیر از روش زیر استفاده می‌شود:

- اگر عدد کوچکتر از ۲باشد، اول نیست.
- وگرنه، اگر عدد کوچکتر از ۴ باشد، اول است.
- وگرنه، اگر عدد زوج باشد، اول نیست.
- وگرنه، اگر هیچ یک از شرط‌های بیان شده برقرار نباشد، بر اعداد فرد از ۳ تا نصف خودش تقسیم می‌گردد، اگر برهیچ عدد فرد بخش‌پذیر نباشد، اول است.

۹. برنامه‌ای که با استفاده از یک متد مساحت مربع یا مستطیل را نمایش می‌دهد (هدف این برنامه آشنایی با متدهای همنام است).

۱۰. متدی به نام SquareOfAsterisks() که پارامتر side را گرفته، با کارکتر * مربعی رسم می‌کند که هر ضلع آن به تعداد side ستاره دارد. برنامه‌ای که از این متد استفاده می‌کند.

۱۱. برنامه‌ای که قیمت کالا و درصد تخفیف را دریافت کرده، به تابعی ارسال کند، و این تابع میزان تخفیف کالا را برگردانده در برنامه میزان تخفیف نمایش داده شود.

۱۲. برنامه‌ای که میزان موجودی حساب بانکی را به همراه درصد بهره سالانه دریافت کرده، تعیین می‌کند پس از چند سال موجودی حساب (بدون برداشت بهره و هیچ مبلغی از حساب) به مبلغ خاصی می‌رسد (موجودی انتهای هر سال توسط تابعی محاسبه گردد).

۱۳. برنامه‌ای که n و k را خوانده با استفاده از متدی به نام C جایگشت C(n, k) را به صورت زیر محاسبه کرده، نمایش می‌دهد(برنامه تا زمانی ادامه می‌یابد که کاربر n و k را برابر وارد کند). جایگشت k و n به صورت زیر محاسبه می‌شود:

$$C(n,k) = \frac{n}{1} \times \frac{n-1}{2} \times \frac{n-2}{3} \times \ldots \times \frac{n-k}{i}$$

۱۴. برنامه‌ای که **n** را خوانده حاصل عبارت مقابل را نمایش می‌دهد: 1! + 2! + 3! + . . . + n!

این برنامه تابعی برای محاسبه فاکتوریل و تابع دیگری برای محاسبه مجموع دارد.

۱۵. برنامه‌ای که **n** را خوانده حاصل عبارت مقابل را نمایش می‌دهد: $\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \ldots + \frac{n}{n!}$

این برنامه تابعی برای محاسبه فاکتوریل و تابع دیگری برای محاسبه مجموع دارد.

۱۶. برنامه‌ای که **x** و **n** را خوانده، حاصل عبارت مقابل را نمایش می‌دهد: $\frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$

این برنامه توابعی برای محاسبه فاکتوریل، توان (x^n) و مجموع دارد.

۱۷. برنامه‌ای که **x** و **n** را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$\frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{1\times 3} + \frac{x^4}{2\times 4} + \frac{x^5}{1\times 3\times 5} + \ldots$$

۱۸. برنامه‌ای که **x** و **n** را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$\frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{1\times3} - \frac{x^4}{2\times4} + \frac{x^5}{1\times3\times5} - \frac{x^6}{2\times4\times6} + \dots$$

۱۹. برنامه‌ای که دو مقدار عددی را خوانده، سپس توسط تابعی به عدد اول یک واحد اضافه نماید و از عدد دوم یک واحد کم کرده آن‌ها را در برنامه اصلی نمایش می‌دهد (در این برنامه دو تابع نوشته شده است. تابع اول، **incdec** که پارامترها را با مقدار فراخوانی می‌کند. در این صورت تغییرات پارامترهای مجازی برای پارامترهای واقعی ارسال نمی‌گردد. اما، تابع **incdec1** دو پارامتر نوع ارجاع را می‌پذیرد و مقادیر تغییر یافته آن‌ها را به برنامه اصلی برمی‌گرداند).

۲۰. برنامه‌ای که **x** و **k** را خوانده، لگاریتم **x** در مبنای **k** را با یک تابع محاسبه کرده، نمایش می‌دهد.

۲۱. برنامه‌ای که عددی را خوانده ریشه دوم عدد را نمایش می‌دهد.

۲۲. متدی به نام DisplayFancyMessage() که سه پارامتر را دریافت کرده، پیغام خاصی را نمایش می‌دهد. پارامتر اول، رنگ نوشته پیغام، پارامتر دوم، رنگ زمینه نوشته و پارامتر سوم پیغام مورد نظر است، برنامه‌ای که از این متد استفاده می‌نماید و دو پیغام را نمایش می‌دهد.

۲۳. برنامه‌ای که با استفاده از یک متد area مساحت مربع و مستطیل را نمایش می‌دهد (هدف این برنامه نوشتن متدهای همنام است).

۲۴. برنامه‌ای که عددی را خوانده، با استفاده از یک متد بازگشتی مجموع اعداد کوچکتر مساوی عدد خوانده شده را محاسبه می‌نماید و نمایش می‌دهد.

۲۵. برنامه‌ای که عددی را خوانده، با استفاده از یک متد معمولی (sum)، مجموع ارقام عدد و یک متد بازگشتی (mul)، حاصل ضرب ارقام عدد را برمی‌گرداند و نمایش می‌دهد.

۲۶. برنامه‌ای که دو عدد n و k را خوانده با استفاده از متدی به نام Prem() (به معنی است) جایگشت P(n , k) را با فرمول زیر محاسبه کرده نمایش می‌دهد:

$$\mathbf{P(n , k) = n \times (n - 1) \times (n - 2) \times \dots \times (n - k + 2) \times (n - k + 1)}$$

۲۷. برنامه‌ای که با روش بازگشتی مقدار مشتق nام را محاسبه می‌کند: $\frac{d(n)}{d_x n}(e^{ax})$

اگر از تابع e^{ax} مشتق بگیریم، رابطه بازگشتی زیر به دست می‌آید: $(e^{ax})n = a \times (e^{ax})^{n-1}$

۲۸. برنامه‌ای که عدد n را خوانده، با استفاده از متد معمولی (sumEvenDigit)، حاصل جمع ارقام زوج و متد بازگشتی (mulOddDigit)، حاصل ضرب ارقام فرد را محاسبه کرده، برمی‌گرداند و برنامه حاصل این دو متد را نمایش می‌دهد.

۲۹. برنامه‌ای که ۱۵ جمله دنباله زیر را به کمک متد بازگشتی نمایش می‌دهد:

$F(1) = 2$ اگر $n == 1$

$F(n) = 2 * F(n-1) + 1$ وگرنه،

۳۰. برنامه‌ای که حاصل عبارت $f(x) = 5x^2 - 3x + 4$ را برای مقادیر صحیح و اعشاری x با دو متد همنام محاسبه می‌کند.

۳۱. برنامه‌ای که قاعده و ارتفاع مثلث را خوانده با استفاده از دو متد همنام مساحت مثلث را حساب می‌کند (مساحت مثلث برابر با قاعده ضرب در نصف ارتفاع است). ارتفاع و قاعده می‌توانند دو عدد صحیح یا دو عدد اعشاری باشند.

۳۲. برنامه‌ای که شعاع کره‌ای (r) را گرفته، مساحت (سطح) و حجم آن را با استفاده ار یک متد (فقط یک متد) محاسبه کرده، به برنامه اصلی برمی‌گرداند و نمایش می‌دهد. برای محاسبه سطح و حجم کره از فرمول‌های زیر استفاده می‌شود:

سطح کره $= 3.1415 * r^2 * 4$ حجم کره $= 3.1415 * r^3 * 4/3$

۳۳. برنامه‌ای که یک کارکتر را خوانده، بدون استفاده از عملگر if (با عملگر XOR) حروف کوچک را به بزرگ یا حروف بزرگ را به کوچک تبدیل می‌کند.

۳۴. برنامه‌ای که عدد n را خوانده با استفاده از تابع بازگشتی S(n)، حاصل عبارت زیر را محاسبه کرده نمایش می‌دهد:

$$S(n) = \begin{cases} 1 & \text{اگر } n \text{ برابر ۱ باشد} \\ \sum_{i=1}^{n} i^2 = S(n-1) + n^2 & \text{وگرنه} \end{cases}$$

۳۵. برنامه‌ای که حداکثر چهار عدد اعشاری را به عنوان پارامترهای ورودی یک متد دریافت کرده، متد میانگین آن‌ها را محاسبه کرده، برمی‌گرداند (هدف از این برنامه، آشنایی با پارامترهای با مقدار پیش‌فرض برای متد است).

۳۶. برنامه‌ای که x و n را خوانده، حاصل عبارت زیر را نمایش می‌دهد: $F(x,n) = \sum_{k=1}^{n} \frac{x^n}{k!}$

۸ – ۳. تمرین

۳۱. متدی بازگشتی بنویسید که x , y را خوانده، حاصل عبارت زیر را برگرداند:

اگر x یا y کوچکتر از صفر باشند F(x, y) = x – y. در غیر این صورت F(x , y) = F (x-1, y) +F (x, y-1)، برنامه‌ای که از این متد بازگشتی استفاده می‌کند.

۳۲. متدی بازگشتی **بنویسید** که n را خوانده، تمام اعداد مضرب ۳ کوچکتر از آن را چاپ کند. برنامه‌ای که از این متد استفاده نماید.

۳۳. متدی بازگشتی **بنویسید** که دو عدد را به عنوان پارامتر دریافت کند و تمام اعداد فرد بین دو عدد را چاپ نماید. برنامه‌ای **بنویسید** که از این متد استفاده نماید.

۳٤. موسسه‌ای در نظر دارد به حقوق کارمندانش x درصد اضافه نماید. متدی **بنویسید** که ضریب افزایش حقوق و حقوق کارمندانش را به عنوان پارامتر دریافت کند. سپس حقوق جدید را بر گرداند. برنامه‌ای **بنویسید** که برای n کارمند از این متد استفاده کند.

۳۵. متدی **بنویسید** که یک عدد را گرفته، علامت عدد (+، – یا ۰) را بر گرداند. برنامه‌ای **بنویسید** که از این متد استفاده کند.

۳٦. متدی **بنویسید** که دو عدد را گرفته و تعیین کند چند عدد اول بین این دو عدد وجود دارند. برنامه، متدی برای تعیین عدد اول دارد. برنامه‌ای **بنویسید** که از این توابع استفاده کند.

۳۷. متدی **بنویسید** که تاثیر نیروی جاذبه بر اجسام در حال سقوط را محاسبه کند. برنامه‌ای **بنویسید** که زمان را دریافت کرده (برحسب ثانیه) و این تاثیر را محاسبه و چاپ کند. این برنامه باید ارتفاع شی سقوط کننده در هر ثانیه را نمایش دهد.

$$\text{فاصله} = \frac{1}{2} \times g \times +2$$

زمان = t ، $g = 9.80665$ و فاصله = ارتفاع

۳۸. برنامه‌ای **بنویسید** که عددی را خوانده، به روش بازگشتی حاصل ضرب ارقام آن را برگرداند.

۳۹. برنامه‌ای **بنویسید** که عددی را خوانده، به روش بازگشتی حاصل ضرب اعداد مضرب ۳ کوچکتر مساوی آن عدد را نمایش دهد.

٤۰. اگر طول دو ضلع (B , C) از مثلث و زاویه بین این دو ضلع Alpha باشد، می‌توانیم طول ضلع سوم را به صورت زیر بدست آوریم:

$$A^2 = B^2 + C^2 - 2 * B * C * Cos(Alpha)$$

برنامه‌ای بنویسید که B ، C و Alpha را خوانده، ضلع A را محاسبه کند و نمایش دهد.

٤١. فرض کنید اساتید دارای حقوق کمتر از ۷۰۰۰۰۰ تومان، ۷ درصد اضافه حقوق می‌گیرند، ولی اساتید بین ۷۰۰/۰۰۰ تومان تا ۱/۰۰۰/۰۰۰ تومان ٤ درصد و اساتید با حقوق بیشتر ۳ درصد اضافه حقوق می‌گیرند، متدی بنویسید که اضافه حقوق کارمند را محاسبه نماید. برنامه‌ای که حقوق اساتید را خوانده اضافه حقوق و حقوق اساتید را محاسبه کند.

٤٢. ریشه دوم یک عدد را با تکرار محاسبات فرمول زیر می‌توان محاسبه کرد:

$$NG = 0.5 \times (LG + N / LG)$$

NG، تخمین بعدی و LG، تخمین فعلی است. متدی که دارای سه پارامتر باشد، پارامتر اول، یک عدد حقیقی، پارامتر دوم، مقدار تخمینی ریشه دوم این عدد و پارامتر سوم، مقدار خروجی عدد محاسبه شده است.

در این متد، مقدار NG محاسبه می‌گردد. سپس تفاضل NG و LG محاسبه می‌شود، اگر این تفاضل برابر صفر گردد، کار متد خاتمه می‌یابد. در غیر این صورت، مقدار LG در NG قرار می‌گیرد و این محاسبات ادامه می‌یابد. برنامه‌ای بنویسید که تعدادی عدد را خوانده، ریشه دوم آن را محاسبه کند. برای خاتمه برنامه به جای عدد مقدار منفی وارد شود.

فصل ۵

برنامه‌نویسی مبتنی برشیء: کلاس‌ها

برنامه هایی که در فصل ۱ تا ۴ دیده اید، همه شامل یک کلاس به نام Program بودند. این کلاس یک متد به نام Main() دارد که تمام برنامه‌ها در داخل آن نوشته می شوند. گاهی نیاز است کلاس‌های جدیدی ایجاد کنید. در این فصل ایجاد کلاس جدید را می‌آموزیم. ساختار کلاس را در (شکل ۱- ۵) می بینید.

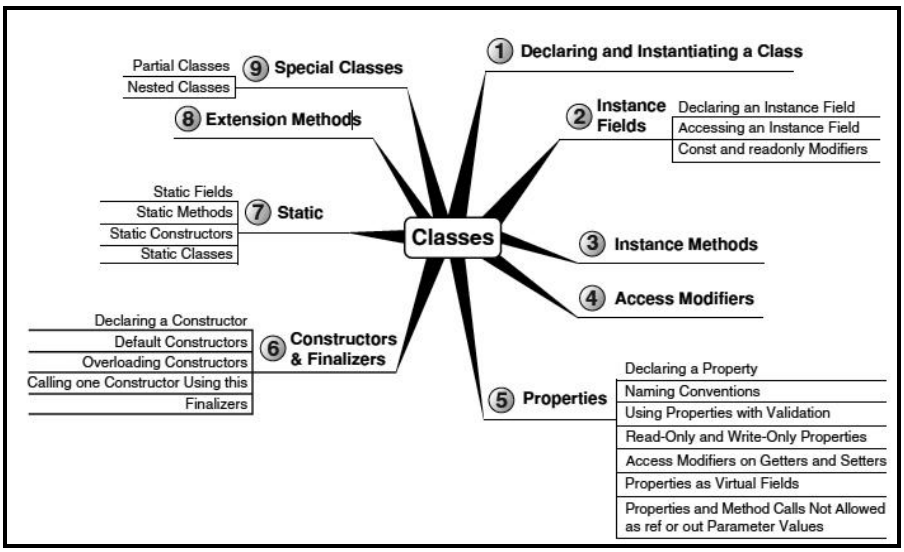
۱ – ۵. کلاس‌ها

دنیای واقعی پر از اشیا است. کافی است به اطراف خودتان نگاه کنید، اطرافتان پر از اشیایی نظیر هواپیما، انسان‌ها، حیوانات، ماشین‌ها، ساختمان‌ها، موبایل‌ها، کامپیوترها و غیره می‌باشد. در زبان‌های برنامه‌نویسی ساخت‌یافته نظیر C، پاسکال، برنامه‌نویسان برروی اعمال[۱۳](کارها یا وظایف (Tasks)) به جای شیء[۲] تمرکز می‌کردند (برنامه نویسان در دنیای واقعی زندگی می‌کنند، اما تفکر آن‌ها شیءگرایی نیست). این امر موجب می‌گردد تا برنامه‌های نوشته شده دارای مشکلات زیر باشند:

۱. واحدهای تشکیل دهنده برنامه نمی‌توانند به سادگی نشان دهنده اشیای دنیای واقعی باشند. لذا، بخش‌های تشکیل دهنده برنامه قابلیت **استفاده مجدد**[۳] زیادی نخواهند داشت.

---

13 .Actions 2.Object 3.Reuse



شکل ۱ – ۵ مطالب مورد نیاز برای کار با کلاس.

۲. بخش‌های تشکیل‌دهنده برنامه اشیای دنیای واقعی را مدل‌سازی نمی‌کنند. لذا، پیچیدگی افزایش می‌یابد.

۳. امکان تغییرپذیری داده‌ها و توابع سخت است.

۴. نگهداری و پشتیبانی برنامه‌ها مشکل‌تر خواهد شد.

اما، در **برنامه نویسی شی‌گراء**، کلاس‌ها ایجاد می‌شوند.

چون کلاس‌ها، اشیای دنیای واقعی را مدل‌سازی می‌کنند. لذا، مزایای زیر را در پی‌خواهند داشت:

۱. **قابلیت استفاده مجدد افزایش می‌یابد**، زیرا، اشیاء را به راحتی می‌توان در برنامه‌های مختلف استفاده نمود.

۲. **به کارگیری اشیاء راحت‌تر خواهد بود**، زیرا، این اشیا، اشیای دنیای واقعی را مدل‌سازی خواهند کرد. بنابراین، موجب کاهش پیچیدگی خواهند شد.

۳. **قابلیت نگهداری اشیاء راحت‌تر است**. در ادامه خواهید دید که هر یک از اشیاء به طور مستقل نگهداری می‌شوند. لذا، تغییرپذیری، توسعه و نگهداری آن‌ها آسان‌تر خواهد شد.

همان‌طور که بیان گردید، کلاس‌ها برای مدل‌سازی اشیای دنیای واقعی به کار می‌روند. اشیای دنیای واقعی یکسری ویژگی هایی از قبیل رنگ، وزن، اندازه، نام، جنس و غیره دارند که شکل ظاهری اشیاء را تعیین می‌کنند. این ویژگی‌ها **صفات اشیاء** نام دارند. صفات اشیاء همان اعضای داده‌ای آن‌ها می‌باشند. اشیاء علاوه برصفات، یکسری رفتارها نیز از خودشان نشان می‌دهند. این رفتارها، **تابع عضو (متد)** نام دارند. متدها، عملیاتی هستند که برروی اعضای داده‌ای قابل اجرا هستند.

کلاس‌ها، الگوهایی برای اشیاء با صفات مشترک و رفتارهای یکسان می‌باشند. به عنوان مثال، کلاس ماشین خواصی مانند رنگ، وزن، تعداد درها، تعداد چرخ‌ها و غیره دارد که شکل ظاهری ماشین را تعیین می‌کند. ماشین ها علاوه برصفات یکسری متدهای از قبیل روشن شدن، خاموش شدن، سرعت گرفتن و ترمزکردن دارند. بنابراین، برای استفاده از ماشین باید از متدهایش استفاده نمود. برای استفاده از کلاس‌ها دو کار باید انجام گردد که عبارت‌اند از:

۱. تعریف کلاس‌ها ۲. نمونه‌سازی از کلاس‌ها

۱-۱-۵. تعریف کلاس‌ها

بیان گردید که کلاس‌ها برای مدل‌سازی اشیاء به کار می‌روند. برای تعریف کلاس ابتدا باید اعضای داده‌ای آن را مشخص کرد. سپس اعضای متدی آن را نوشت. تعریف کلاس در C# با واژه class به صورت زیر انجام می‌شود:

```
سطح دسترسی کلاس   class   نام کلاس {
    اعضای کلاس
}
```

سطح دسترسی کلاس محدوده دسترسی و استفاده از کلاس را تعیین می‌کند که می‌تواند internal یا public باشد. اگر در سطح دسترسی کلاس چیزی ذکر نشود، internal در نظر گرفته می‌شود. در این صورت از کلاس می‌توان در فضای نامی که کلاس تعریف شده است استفاده نمود. اما، اگر سطح دسترسی کلاس public تعیین شود، علاوه بر فضای نامی که کلاس در آن تعریف شده است، در فضای نام دیگر نیز می‌توان از کلاس استفاده کرد. نام کلاس، از قانون نام‌گذاری شناسه‌ها پیروی می‌کند و اعضای تشکیل‌دهنده کلاس، متغیرها و متدهایی هستند که کلاس از آن‌ها تشکیل می‌شود. اعضای تشکیل‌دهنده کلاس عبارت‌اند از:

۱ . ثابت‌ها ۲. فیلدها ۳. خواص ۴. سازنده‌ها

۵ . مخرب‌ها ۶. متدها ۷. ایندکسرها ۸. delegateها

در ادامه در مورد هر یک از اعضای کلاس به طور مفصل بحث خواهیم کرد.

برای اضافه کردن (تعریف کلاس) جدید مراحل زیر را انجام دهید:

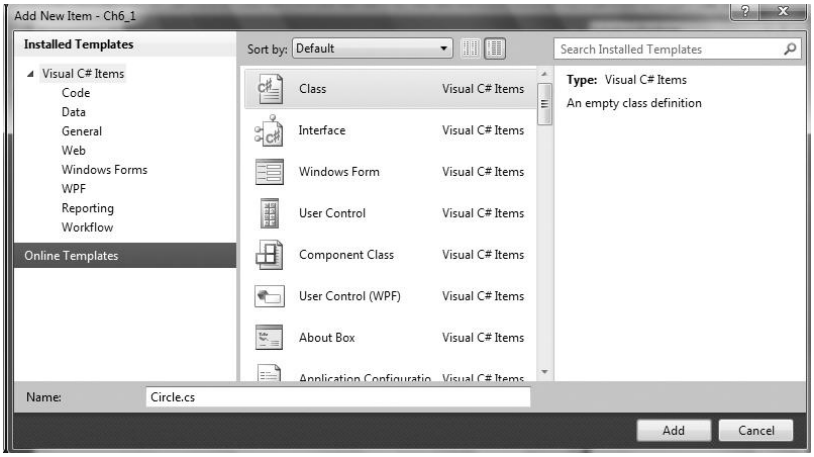
۱. پروژه‌ایی را ایجاد کنید که می‌خواهید به آن کلاس جدید اضافه نمایید (من پروژه Ch5_1 را ایجاد کردم).

۲. گزینه Project / Add Class را اجرا کنید تا پنجره Add New Project ظاهر شود (شکل ۲ – ۵). در بخش Name نام کلاس را انتخاب کنید (من Circle.cs انتخاب کردم).

۳. دکمه Add را کلیک کنید تا کلاس جدید اضافه شود. اکنون دستورات کلاس را به صورت زیر می‌بینید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Ch6_1  {
 class Circle    {
 }
}
```

در این دستورات بخش اول، فضاهای نام پیش‌فرض قرار دارند. چنانچه به فضای نامی نیاز نداشته باشید می‌توانید آن را حذف کنید. در بخش بعد، فضای نام آمده است (فضای نامی که این کلاس در آن تعریف می‌باشد). و در بخش آخر، تعریف کلاس را می‌بینید. Ch5_1شده است



برای ایجاد کلاس.Add New Itemشکل ۲ – ۵ پنجره جدید

سطح دسترسی کلاس internal است. چون، چیزی ذکر نگردید. بنابراین کلاس فقط می‌تواند در فضای نامی که در آن تعریف شده است (اینجا Ch5_1) استفاده شود. نام کلاس Circle است. برای این کلاس هیچ عضوی تعریف نگردید. البته برخی از اعضای کلاس object را به ارث می‌برد. در ادامه اعضایی که این کلاس از کلاس object به ارث می‌برد، خواهید دید.

هر کلاس جدیدی که ایجاد می‌شود، برخی از اعضای کلاس object را به ارث می‌برد.

۲ – ۱ – ۵. نمونه‌سازی کلاس‌ها

همان‌طور که در فصل اول دیدید، برای استفاده از انواعی نظیر int، double، char و ... باید متغیرهایی از این انواع تعریف می‌گردید، برای استفاده از کلاس باید نمونه‌ای (شیایی) از کلاس ایجاد کنید. برای این منظور، به صورت زیر عمل می‌شود:

نام کلاس () = new نام نمونه نام کلاس;

نام نمونه، از قانون نامگذاری متغیرها پیروی می‌کند. به عنوان مثال، دستورات زیر را در کلاس Program (فایل Program.cs) تایپ کنید:

```
Circle C1 = new Circle();
```

این دستور نمونه‌ای از کلاس Circle به نام C1 ایجاد می‌کند. دسترسی به اعضای کلاس به صورت‌های زیر می‌باشد:

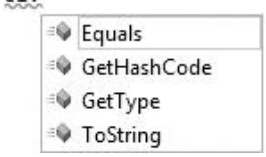
1. **نام عضو . نام نمونه**
2. **نام عضو . نام کلاس**

ساختار اول، برای دسترسی به اعضای معمولی کلاس به کار می‌رود و ساختار دوم برای دسترسی به اعضای ثابت و نوع static به کار می‌رود که در ادامه خواهید دید.

دستورات مقابل را در کلاس Program تایپ کنید:

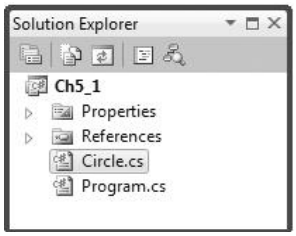
```
C1.
```

اکنون شکل مقابل را مشاهده خواهید کرد:



همان‌طور که در این شکل می‌بینید، با توجه به این که هنوز عضو جدیدی برای کلاس تعریف نکردیم، کلاس دارای چهار عضو متدی به نام‌های Equals() (برابری دو شی را بررسی می‌کند)، GetHashCode() (کد شیء را برمی‌گرداند)، GetType() (نوع شی را برمی‌گرداند) و ToString() (نام فضای نام و نام کلاس را برمی‌گرداند) است. کلاس Circle این متدها را از کلاس object به ارث برده است. با مفهوم وراثت در فصل ۷ آشنا خواهید شد.

پنجره Solution Explorer بعد از اضافه کردن کلاس جدید



پس از اضافه کردن کلاس جدید، فایل جدیدی به نام کلاس و پسوند cs به پروژه اضافه می‌شود. این فایل را می‌توانید در پنجره Solution Explorer ببینید. برای این منظور، گزینه View / Solution Explorer را اجرا کنید تا پنجره Solution Explorer را مشاهده نمایید (شکل ۳– ۵). همان طور که در این شکل می‌بینید، این برنامه از دو کلاس Program.cs و Circle.cs تشکیل شده است. برای نمایش کد هر یک از این فایل‌ها برروی آن کلیک مضاعف کنید.

شکل ۳ – ۵ پنجره Solution Explorer.

۲ – ۵. اعضای کلاس

همان‌طور که دیدید، برای کلاس باید اعضای آن را پیاده‌سازی نمود. کلاس‌ها معمولاً از دو نوع عضو تشکیل می‌شود که عبارت‌انداز:

اعضای داده‌ای، اعضایی که مقادیر مورد نیاز کلاس را نگهداری می‌کنند. این اعضا می‌توانند ثابت یا قابل تغییرباشند. به عنوان مثال، کلاس دایره دارای دو عضو داده‌ای است. عضو داده‌ای اول PI است که عدد 3.14159 می‌باشد و عضو داده‌ای دوم شعاع (r) است که شعاع دایره در آن قرار می‌گیرد. در ادامه اعضای داده‌ای ثابت و متغیر را می‌آموزیم.

اعضای متدی، اعمالی را تعیین می‌کنند که کلاس می‌خواهد انجام دهد. اعضای متدی می‌توانند سازنده‌ها، مخرب‌ها و متدهای کلاس باشند. به عنوان مثال، دایره دارای متدهایی جهت محاسبه محیط و مساحت می‌باشد. بنابراین کلاس می‌تواند اعضای زیر را داشته باشد:

۱. ثابت‌ها ۲. فیلدها ۳. خواص ۴. سازنده‌ها ۵. مخرب‌ها ۶. متدها ۷. و غیره

مثال ۳ – ۵. برنامه‌ای که کلاسی طراحی می‌کند که اطلاعات زیر را دارد:

۱. شماره استادی ۲. نام استاد ۳. نام خانوادگی استاد

٤. ساعات تدریس ۵. مبلغ پرداختی به ازای هر ساعت تدریس

در این کلاس مبلغ حق‌التدریس استاد را محاسبه می‌کند که به صورت زیر محاسبه می‌شود:

مبلغ پرداختی به ازای هر ساعت تدریس * ساعت تدریس = مبلغ حق‌التدریس

برنامه‌ای که از این کلاس استفاده می‌کند

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام Ch5_3 ایجاد کنید.

۲. کلاس Teacher را به پروژه اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
using System;
namespace Ch5_3 {
 class Teacher  {
   private string id;
   private string firstName;
   private string lastName;
   private int hour;
   private int payPerOneHour;
   public string ID    {
     get    {   return id;   }
     set    {   id = value;  }
   }
   public string FirstName    {
     get   {    return firstName;  }
     set   {   firstName = value;  }
   }
   public string LastName   {
      get   {   return lastName;   }
      set   {   lastName = value;  }
   }
   public int Hour   {
      get   {   return hour;       }
      set   {
       if (value > 0) hour = value;
       else hour = 0;
      }
   }
   public int PayPerOneHour    {
     get    { return payPerOneHour;}
     set    {
       if (value > 0) payPerOneHour = value;
       else payPerOneHour = 0;
     }
   }
   public Teacher() { }
   public Teacher(string i, string f, string l, int h, int p) {
    ID = i;FirstName = f;LastName = l;Hour = h;PayPerOneHour=p;
   }
   public long Payment()  { return hour * payPerOneHour; }
  }
}
```

این کلاس اعضای زیر را دارد:

- **فیلد id**، شماره استادی را نگهداری می کند.

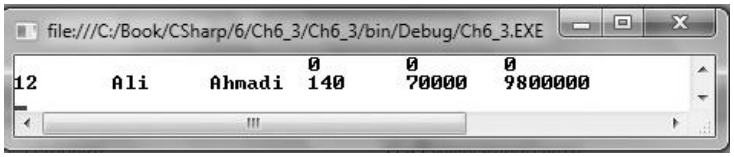
- **خاصیت ID**، برای بازیابی و مقداری دهی به فیلد id به کار می‌رود.
- **فیلد firstName**، نام استاد را نگهداری می کند.
- **خاصیت FirstName**، برای بازیابی و مقداردهی به فیلد firstName به کار می‌رود.
- **فیلد lastName**، نام‌خانوادگی استاد را نگهداری می‌نماید.
- **خاصیت LastName**، برای بازیابی و مقداردهی به فیلد lastName استفاده می‌شود.
- **فیلد hour**، برای نگهداری ساعات تدریس استاد به کار می‌رود.
- **خاصیت Hour**، جهت بازیابی و مقداردهی فیلد hour به کار می‌رود. متد set از ورود داده‌های منفی برای فیلد hour جلوگیری می کند.
- **فیلد payPerOneHour**، مبلغ پرداختی به ازای هر ساعت تدریس استاد را ذخیره می کند.
- **خاصیت payPerOneHour**، جهت بازیابی و مقداردهی فیلد payPerOneHour به کار می‌رود و متد set آن از ورود مبلغ منفی در فیلد payPerOneHour جلوگیری خواهد کرد.
- **سازنده بدون پارامتر ()Teacher**، سازنده پیش‌فرض را پیاده‌سازی می کند که به فیلدهای عددی مقدار اولیه صفر، رشته‌ای مقدار اولیه null را می‌دهد.
- **سازنده با پارامتر (...)Teacher**، پارامترهایی را می گیرد و به خواص تعیین شده در کلاس تخصیص می‌دهد.
- **متد ()Payment**، مبلغ حق‌التدریس پرداختی به استاد را محاسبه کرده برمی گرداند.

۳. به کلاس Program برگردید و دستورات آن را به صورت تغییر دهید:

```
using System;
namespace Ch5_3 {
 class Program  {
  static void Main(string[] args)  {
   Teacher t1 = new Teacher();
   Console.WriteLine(t1.ID+"\t"+t1.FirstName+"\t"+t1.LastName+"\t"+
      t1.Hour.ToString()+"\t"+t1.PayPerOneHour.ToString()+"\t"+
      t1.Payment() .ToString());
   Teacher t2 = new Teacher("12","Ali","Ahmadi", 140, 70000);
   Console.WriteLine(t2.ID + "\t" + t2.FirstName + "\t" + t2.LastName +
         "\t" + t2.Hour.ToString() + "\t" + t2.PayPerOneHour.ToString()
         + "\t" + t2.Payment().ToString());
   Console.ReadKey();
  }
 }
}
```

دستور اول داخل متد ()Main، یک نمونه به نام t1 از نوع Teacher ایجاد می‌کند. برای ایجاد این نمونه سازنده پیش‌فرض را فراخوانی می‌نماید تا مقدار اولیه null را برای فیلدهای id، firstName و lastName و مقدار اولیه صفر را برای فیلدهای hour و PayPerOneHour تعیین کند. دستور دوم، از طریق خواص نمونه t1 مقدار فیلدهای آن را نمایش می‌دهد. دستور سوم، نمونه t2 را ایجاد می‌کند. برای ایجاد نمونه t2 سازنده با پارامتر کلاس Teacher را فراخوانی می‌کند و به فیلدهای id، firstName، LastName، hour و payPerOneHour به ترتیب مقادیر اولیه "۱۲"، "Ali"، "Ahmadi"، ۱۴۰ و ۷۰۰۰۰ را تخصیص خواهد داد و دستور چهارم، مقادیر فیلدها را از طریق خواص آن‌ها نمایش می‌دهد.

۴. پروژه را ذخیره و اجرا کنید تا خروجی زیر را ببینید:



۱۱ ـ ۵. مسائل حل شده

مثال ۱ – ۵. برنامه‌ای که شکل مربع را پیاده‌سازی می‌کند. برای این منظور کلاس Square را ایجاد می‌نماید. متغیر خصوصی side را در این کلاس ایجاد می‌کند که توسط متدهای set و get بتوان به آن دسترسی یافت. کلاس باید شامل دو متد سازنده باشد. یکی از سازنده‌ها آرگومان ندارد و متد سازنده دیگر مقدار side را باید به عنوان آرگومان بپذیرد.

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh5_ 1 ایجاد کنید.

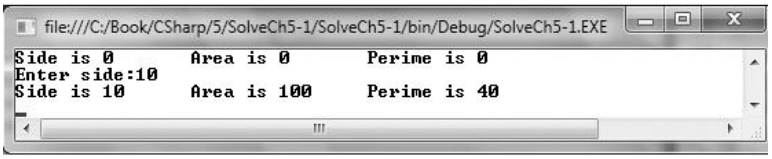
۲. دستورات کلاس Square را به پروژه اضافه کنید ودستورات آن را به صورت زیر تغییر دهید:

```
using System;
namespace SolveCh5_1 {
 class Square        {
  private int side;
  public int Side    {
    get { return side; }
    set { side = (value > 0) ? value : 0; }
  }
  public Square() { }
  public Square(int side) { Side = side;   }
  public int Area() { return side * side; }
  public int Perime() { return 4 * side;  }
 }
}
```

۳. به کلاس Program بروید و دستورات آن را به صورت زیر تغییر دهید:

```
using System;
namespace SolveCh5_1  {
 class Program        {
 static void Main(string[] args)  {
   Square s1 = new Square();
   Console.WriteLine("Side is {0}\tArea is {1}\tPerime is {2}",
          s1.Side, s1.Area().ToString(), s1.Perime().ToString());
    Console.Write("Enter side:");
   Square s2 = new Square(Convert.ToInt32(Console.ReadLine()));
   Console.WriteLine("Side is {0}\tArea is {1}\tPerime is {2}",
          s2.Side, s2.Area().ToString(), s2.Perime().ToString());
    Console.ReadKey();
  }
 }
}
```

۴. پروژه را ذخیره و اجرا کنید تا خروجی زیر را ببینید:



مثال ۲ – ۵. برنامه‌ای که کلاسی به نام SavingAccount ایجاد می‌کند. این کلاس از متغیر استاتیکی به نام annualInterestRate برای ذخیره سازی نرخ سود همه حساب‌ها استفاده می‌کند. هر شی از این کلاس شامل متغیر خصوصی به نام SavingBalance می‌باشد که مقدار پس‌انداز جاری سپرده را در خود نگه می‌دارد. متدی به نام CalculateModifyInterest() دارد که سود ماهانه را با فرمول زیر محاسبه می‌نماید:

(SavingBalance * annualInterestRate)/12

سپس سود را به SavingBalance اضافه می‌نماید. کلاس متد استاتیکی به نام ModifyInterestRate که به مقدار فیلد annualInterestRate یک مقدار جدید تخصیص می‌دهد. برنامه‌ای که از این کلاس استفاده می‌کند. در این برنامه دو شیء به نام‌های saver1 و saver2 ایجاد می‌کند و به هر یک تراز ۲۰۰۰۰۰ دلار و ۳۰۰۰۰۰ دلار نسبت می‌هد. مقدار annualInterestRate را ٤٪ قرار می‌دهد و سپس سود ماهانه را حساب می‌کند و تراز را برای saver1 و saver2 محاسبه کرده، نمایش می‌هد. سپس مقدار annualInterestRate را ۵٪ قرار می‌دهد و سود ماه بعد را حساب می‌نماید و تراز جدید را برای هر یک از حساب‌ها نشان می‌دهد (تراز همان فیلد savingBalance است).

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh5_2 ایجاد کنید.

۲. کلاس SavingAccount را به پروژه اضافه کنید ودستورات آن را به صورت زیر تغییر دهید:

```
using System;
namespace SolveCh5_2 {
 class SavingAccount {
  private double savingBalance;
```

```
  public static double annualInterestRate;
  public double SavingBalance   {
    get { return savingBalance; }
    set { savingBalance = (value > 0) ? value: 0; }
  }
  public SavingAccount()  { }
  public SavingAccount(double saving)   {
    SavingBalance = saving;
  }
  public double  CalculateModifyInterest()  {
    return SavingBalance * annualInterestRate / 12;
  }
  public static void ModifyInterestRate(double rate)   {
    annualInterestRate = rate;
  }
 }
}
```

۳. به کلاس program بروید ودستورات آن را به صورت زیر تغییر دهید:

```
using System;
namespace SolveCh5_2 {
 class Program       {
  static void Main(string[] args)  {
   SavingAccount Saver1 = new SavingAccount(200000);
   SavingAccount Saver2 = new SavingAccount(300000);
   SavingAccount.ModifyInterestRate(0.04);
   Saver1.SavingBalance+=Saver1.CalculateModifyInterest();
   Saver2.SavingBalance+=Saver2.CalculateModifyInterest();
   Console.WriteLine("Saving Balance1:{0}\t\tSaving  Balance2:{1}",
         Saver1.SavingBalance, Saver2.SavingBalance);
   SavingAccount.ModifyInterestRate(0.05);
   Saver1.SavingBalance+=Saver1.CalculateModifyInterest();
   Saver2.SavingBalance+=Saver2.CalculateModifyInterest();
   Console.WriteLine("Saving Balance1:{0}\t\tSaving Balance2:{1}",
          Saver1.SavingBalance, Saver2.SavingBalance);
   Console.ReadKey();
  }
 }
}
```

۴. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:

۱۲ – ۵. مسائل حل شده در سایت

مثال ۱ – ۵. برنامه‌ای که نقطه شروع و پایان خطی را دریافت می‌کند، طول خط را محاسبه کرده، نمایش می‌دهد. این برنامه سه کلاس زیر را دارد:

۱. کلاس Point، اطلاعات نقطه‌ای را نگهداری می‌کند. این کلاس دارای اعضای زیر است:

- فیلدهای x و y مختصات نقطه را نگهداری می‌کنند.
- خواص X و Y برای بازیابی و مقدار دهی به فیلدهای x و y به کار می‌روند.

۲. کلاس Line، اطلاعات خط را نگهداری می‌کند. این کلاس فیلدهای زیر را دارد:

- فیلدهای starting، ending، نقطه شروع و پایان خط را نگهداری می‌کنند که از نوع کلاس Point هستند و len طول خط را نگهداری می‌کند.

۳. کلاس Program، متدی به نام Main() دارد. در این متد نمونه‌ای از کلاس Line به نام myLine ایجاد می‌شود. از طریق خواص X و Y به فیلدهای starting و ending نمونه myLine مقدار تخصیص می‌یابد. سپس طول خط حساب شده، در فیلد len قرار می‌گیرد و نقطه شروع، پایان و طول خط نمایش داده می‌شود.

مثال ۲ – ۵. برنامه‌ای که مجموع مساحت‌ها و محیط‌های چند مربع را نمایش می‌دهد(در این برنامه تعداد مربع‌ها و اضلاع آن‌ها به صورت تصادفی تولید می‌گردند). این برنامه کلاس‌های زیر را دارد:

۱. کلاس Square، اعضای زیر را دارد:

- **فیلد side**، ضلع مربع را نگهداری می‌کند.
- **خاصیت Side**، برای بازیابی و مقدار دهی فیلد side به کار می‌رود.
- **سازنده ()Square**، به فیلد Side از طریق خاصیت Side مقدار اولیه می‌دهد.
- **متد ()Area**، مساحت مربع را محاسبه می‌نماید.
- **متد ()Perime**، محیط مربع را محاسبه می‌کند.
- **متد ()ToString**، ضلع مربع، محیط و مساحت را با پیغام مناسب برمی‌گرداند.

۲. کلاس Program، متدی به نام Main() دارد. این متد، یک نمونه به نام r از نوع Random ایجاد می‌کند، یک عدد تصادفی تولید کرده، در n قرار می‌دهد، به تعداد عدد تصادفی تولید شده(یعنی،n)، نمونه از کلاس Square ایجاد می‌کند(آرایه‌ای از Square به نام s ایجاد می‌کند، متغیرهای totalPerime و totalArea را تعریف کرده، مقدار اولیه صفر را در آن‌ها قرار می‌دهد در حلقه تکرار، یک نمونه از Square ایجاد می‌کند، به عنصر آرایه (S[i]) تخصیص می‌دهد، محیط و مساحت نمونه را محاسبه می‌کند و با

متغیرهای totalArea، totalPerime جمع می‌کند. با فراخوانی متد ()ToString، ضلع مربع، محیط و مساحت آن را نمایش می‌دهد. در پایان، مجموع مساحت‌ها و محیط‌ها را نمایش خواهد داد.

مثال ۳ – ۵. برنامه‌ای که استفاده از delegate را نشان می‌دهد. این برنامه کلاس‌های زیر را دارد:

- **کلاس SimpleMath**، متدهای ()Subtract، ()Add، ()Mul، ()Div و ()Mod را دارد که دو پارامتر را دریافت می‌کنند که به ترتیب حاصل تفریق، مجموع، حاصل ضرب، حاصل تقسیم و حاصل باقی‌مانده تقسیم صحیح آن‌ها را بر می‌گردانند.
- **کلاس Program**، یک delegate به نام BinaryMath را تعریف می‌کند که یک متد با دو پارامتر از نوع int را می‌پذیرد و نتیجه را به صورت int برمی‌گرداند. متد ()DisplayDelegateInfo دارد که پارامتری به نام delObj از نوع Delegate را دریافت کرده، اطلاعات آن را بر می گرداند. متد ()Main، پیغامی را نمایش داده، نمونه‌ای به نام b از نوع BinaryMath (delegate) ایجاد می‌نماید. در ادامه، از طریق نمونه b، متدهای ()Div، ()Mul، ()Subtract، ()Add و ()Mod را فراخوانی می‌کند.

مثال ۴ – ۵. برنامه‌ای که اندازه پارچه را به یارد دریافت کرده، به متر تبدیل می‌کند(هر متر 1.196 یارد است). این برنامه کلاس‌های زیر را دارد:

- **کلاس Convert**، برای تبدیل اندازه پارچه از یارد به متر به کار می‌رود.
- **فیلد yards** ، اندازه پارچه را به یارد نگهداری می‌کند. خاصیت Yards، برای بازیابی و مقداردهی فیلد yards استفاد می‌شود.
- **سازنده ()Convert**، به فیلد yards از طریق خاصیت yards مقدار اولیه می‌دهد.
- **متد()toMeter**، اندازه پارچه را به متر تبدیل می‌کند.
- **کلاس Program**، دارای متد ()Main است. این متد، نمونه‌ایی به نام c از نوع Convert ایجاد کرده، اندازه پارچه را به یارد دریافت می‌کند، با فراخوانی متد ()toMeter آن را به متر تبدیل می‌کند و نمایش می‌دهد.

مثال ۵ – ۵. برنامه‌ای که قاعده و ارتفاع مثلثی را دریافت کرده، محیط و مساحت مثلث را حساب می‌کند. محیط و مساحت مثلث به صورت زیر محاسبه می‌گردد:

$$\text{محیط مثلث} = \text{قاعده} + \text{ارتفاع} + \sqrt{(\text{قاعده})^2 + (\text{ارتفاع})^2}$$

$$\text{مساحت مثلث} = \text{قاعده} \times \text{ارتفاع} \ / \ ۲$$

این برنامه دو کلاس زیر را دارد:

۱. کلاس Traingle، نوع مثلث را پیاده سازی می‌کند که اعضای زیر را دارد:

- **فیلدهای base1 و height**، به ترتیب قاعده و ارتفاع مثلث را نگهداری می‌کنند.
- **خواص Base1 و Height** ، برای بازیابی و مقدار دهی به فیلدهای base1 و height استفاده می‌شوند.
- **سازنده ()Traingle**، به فیلدهای base1 و height از طریق خواص Base1 و Height مقدار اولیه تخصیص می‌دهد. متدهای ()Area و ()Perimeter به ترتیب مساحت و محیط مثلث را حساب می‌کنند.

۲. کلاس Program، متدی به نام ()Main دارد. این متد، ابتدا نمونه‌ای به نام t1 از نوع کلاس Triangle ایجاد می‌کند. قاعده و ارتفاع مثلث را با پیغام‌های مناسب از ورودی خوانده، در خواص Base1 و Height نمونه t1 قرار می‌دهد. در پایان با فراخوانی متدهای ()Area و ()Perimeter محیط و مساحت مثلث را نمایش می‌دهد.

مثال ۶ – ۵. برنامه‌ای که زمان را دریافت کرده و آن را به صورت Am (قبل از ظهر) یا Pm (بعدازظهر) نمایش می‌دهد. این برنامه کلاس‌های زیر را دارد:

۱. کلاس Time، اعضای زیر را دارد:

- **فیلدهای hour، minute و second**، به ترتیب مقادیر ساعت، دقیقه و ثانیه را نگهداری می‌کنند.
- **خواص Houre، Minute و Second**، به ترتیب مقدار فیلدهای hour، minute و second را بازیابی می‌کنند و به این فیلدها مقدار می‌دهند(متد ()set چک می‌کند تا زمان نادرست وارد نشود).
- **سازنده ()Time**، مقادیر ساعت (h)، دقیقه (m) و ثانیه (s) را به عنوان پارامتر دریافت کرده، از طریق خواص Hour، Minute و Second به فیلدها مقدار اولیه می‌دهند.
- **متد ()ToString**، زمان را به فرمت‌های Am یا Pm تبدیل کرده، برمی‌گرداند.

۲. کلاس Program ، متد ()Main این کلاس، دو نمونه به نام‌های t1 و t2 ایجاد کرده، با فراخوانی ()ToString بر روی این نمونه، آن را با فرمت‌های Am یا Pm نمایش می‌دهد.

مثال ۷ – ۵. برنامه‌ای که شماره کارمندی، نام، تاریخ استخدام و حقوق ناخالص تعدادی کارمند را مقدار دهی می‌کند، مالیات، بیمه و حقوق خالص را محاسبه می‌کند. در پایان مجموع حقوق خالص را نمایش می‌دهد. مالیات، بیمه و حقوق خالص به صورت زیر محاسبه می‌گردند.

بیمه = حقوق ناخالص * ۷ / ۱۰۰

اگر حقوق بیشتر از ۴۸۵۰۰۰۰ ریال باشد، آنگاه مالیات برابر است با

مالیات = حقوق ناخالص *۱۰/ ۱۰۰

وگرنه، مالیات صفر در نظر گرفته می‌شود.

بیمه – مالیات – حقوق ناخالص = حقوق خالص

این برنامه کلاس‌های زیر را دارد:

۱. کلاس‌های Date، اعضای زیر را دارد:

- **فیلدهای year، month و day**، سال، ماه، و روز را نگهداری می‌کند.
- **خواص Year، Month و Day**، مقدار فیلدهای year، month و day را بازیابی کرده، یا به آن‌ها مقدار می‌دهند.
- **سازنده Date()**، سال (y)، ماه (m) و روز (D) را به عنوان پارامتر دریافت کرده، از طریق خواص Year، Month و Day به فیلدهای year، month و day تخصیص می‌دهد.

۲. **کلاس Employee**، اعضای زیر را دارد:

- **فیلدهای code، name، empDate و salary**، به ترتیب مقادیر شماره کارمندی، نام، تاریخ استخدام و حقوق را نگهداری می‌کنند. البته دقت کنید که empDate نمونه‌ای از نوع کلاس date است.
- **خواص Code، Name و Salary**، به ترتیب برای مقدار دهی یا بازیابی مقدار فیلدهای code، name و Salary به کار می‌روند.
- **سازنده Employee()**، اطلاعات مورد نیاز کارمند را از طریق پارامتر دریافت کرده، در فیلدهای کلاس Employee قرار می‌دهد.
- **متدهای calInsurance()، calTax() و calPayment()**، به ترتیب بیمه، مالیات و حقوق خالص را محاسبه کرده، برمی‌گردانند.

۳. کلاس Program، دارای متد Main() است. در این متد سه نمونه از کلاس Employee ایجاد شده، در آرایه Emp قرار می‌دهد. سپس با سازنده Employee()، به عناصر آرایه اطلاعات سه کارمند را تخصیص خواهد داد. در ادامه، با حلقه تکرار foreach اطلاعات هر کارمند را نمایش می‌دهد و حقوق خالص او را با totalPayment جمع می‌کند. در پایان، مقدار totalPayment را نمایش می‌دهد.

مثال ۸ – ۵. برنامه‌ای که شعاع بزرگ (طولی) و شعاع کوچک (عرضی) بیضی را دریافت کرده، مساحت بیضی را با فرمول زیر محاسبه می‌کند:

$$\text{مساحت} = \pi * \text{شعاع بزرگ} * \text{شعاع کوچک}$$

این برنامه دو کلاس زیر را دارد:

کلاس Ellipse، فیلدهای width (شعاع عرضی)، length(شعاع طولی) و سازنده ()Ellipse (مقدار شعاع عرضی و طولی را به عنوان پارامتر می‌گیرد، درفیلدهای width و length قرار می‌دهد)، متد ()Area (جهت محاسبه مساحت بیضی) و متد ()ToString (شعاع عرضی، طولی و مساحت بیضی را بر می‌گرداند) را دارد.

کلاس Program، در متد ()Main این کلاس، شعاع عرضی و طولی را از ورودی دریافت می‌کند، نمونه‌ای به نام e از کلاس Ellipse ایجاد می‌کند و با فراخوانی متد ()ToString در متد ()WriteLine اطلاعات بیضی را نمایش می‌دهد.

۱۳ – ۵. تمرین

۱. کلاسی برای اعداد کسری بنویسید که امکانات زیر را داشته باشد:

- سازنده‌ای که از مخرج صفر برای کسر جلوگیری کند.
- متدی برای جمع دو کسر دارد.
- متدی برای تفریق دو کسر دارد.
- متدی برای ضرب دو کسر دارد.
- متدی برای ساده کردن دو کسر دارد.
- متد ()ToString را مجددا پیاده‌سازی می‌نماید تا یک عدد کسری را به صورت رشته برگرداند.

۲. کلاسی به نام Date بنویسید که دارای ویژگی‌های زیر باشد:

- سال، ماه و روز را بپذیرد.
- سازنده‌ای بنویسید که اگر ماه بین ۱ تا ۶ است، کاربر نتواند روز را بیشتر از ۳۱ وارد نماید. ولی، اگر ماه بین ۷ تا ۱۱ است، کاربر نتواند روز را بیشتر از ۳۰ وارد نماید و اگر ماه ۱۲ بود، کاربر نتواند روز را بیشتر از ۲۹ وارد کند. در این سازنده از ورود سال، ماه و روز منفی جلوگیری کند.
- متد ()Equals را مجددا پیاده‌سازی کنید تا دو تاریخ را با یکدیگر مقایسه کند.
- متدی به نام ()nextDay بنویسید که تاریخ را یک روز اضافه کند.
- متدی بنویسید که حاصل جمع دو تاریخ را برگرداند.
- متد ()ToString را مجددا پیاده‌سازی کنید تا تاریخ را به فرمت روز/ ماه/ سال تبدیل کرده، برگرداند (سال چهارم رقم، ماه دوم رقم و روز دوم رقم).

فصل ۷ برنامه‌های کاربردی با فرم

برنامه‌هایی که تاکنون نوشته شده، همه از نوع Console Application بودند. اکثر برنامه‌های واقعی و بزرگ دارای فرم‌های متعددی هستند که کاربران می‌توانند از طریق آن‌ها داده را وارد کنند. بنابراین باید بتوانید برنامه‌هایی که دارای فرم هستند را ایجاد کنید. دراین فصل ابتدا برنامه‌های نوع Forms Application را خواهیم پرداخت. سپس برخی از کنترل‌های ساده، اضافه کردن و روش استفاده از آن‌ها را در فرم می‌آموزیم.

۷–۱. مراحل نوشتن برنامه‌های ویندوزی

برای نوشتن برنامه‌های ویندوزی باید مراحل زیر را انجام دهید:

۱. تعیین کنید در برنامه به چه کنترل‌ها (اشیا) نیاز است. سپس اشیا مورد نیاز را به برنامه اضافه نمایید.

۲. خواص هر یک از کنترل‌ها را تعیین نمایید. همان‌طور که بیان گردید، خواص کنترل، شکل ظاهری کنترل را تعیین می‌کنند.

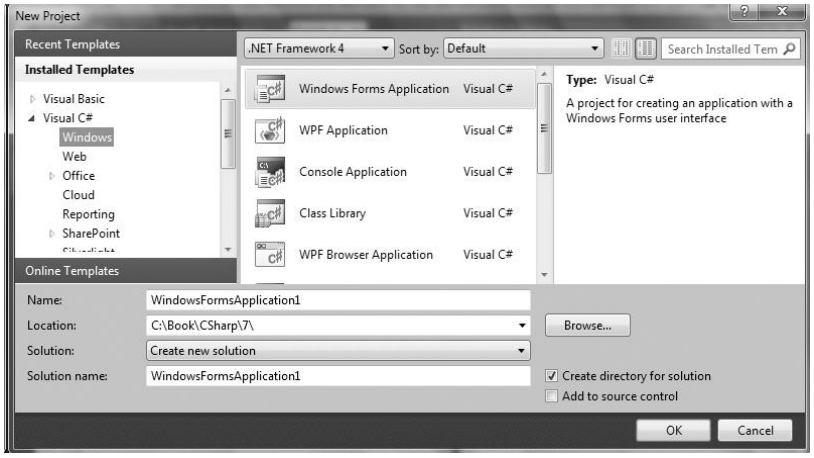
۳. تعیین شود هر کنترل به چه رویدادهایی باید پاسخ دهد، سپس برنامه پاسخ‌گو به رویدادهای کنترل‌ها را بنویسید.

۴. پروژه را خطایابی (کامپایلر) و اجرا نمایید.

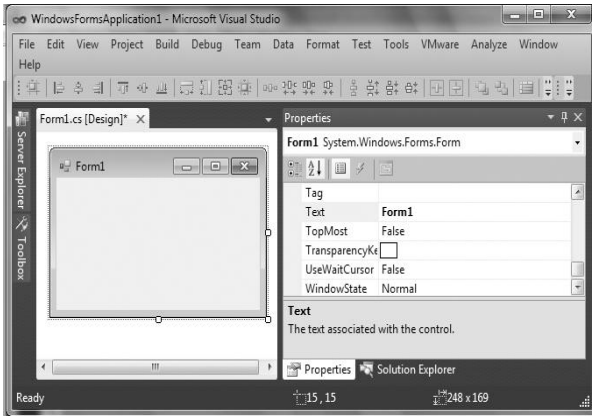
۷–۲. ایجاد برنامه جدید و اضافه کردن کنترل‌ها به فرم

همان‌طور که بیان گردید، یکی از انواع برنامه‌ها در #C، برنامه‌های Windows Forms Application می‌باشد. این برنامه‌ها پس از ایجاد آن، شامل یک فرم هستند که کنترل‌های مورد نیاز برنامه در آن قرار می‌گیرند. برای ایجاد چنین برنامه‌ای مراحل زیر را اجرا کنید:

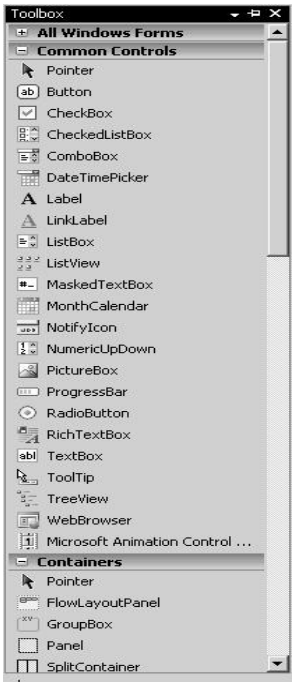
۱. گزینه File/New/Project (Ctrl + Shift + N) را اجرا کنید تا پنجره New Project ظاهر شود (شکل ۱ – ۷). در این پنجره، در بخش زبان، Visual C# را انتخاب کنید. در بخش نوع برنامه، آیکن Windows Forms Application را انتخاب کرده، بخش Name و Location آن را تعیین کنید (در این برنامه فعلاً بخش Name و Location آن را تغییر ندادیم) و دکمه OK را کلیک کنید تا برنامه ایجاد شود. فرم ایجاد شده برنامه را در (شکل ۲ – ۷) می‌بینید.



شکل ۱ – ۷ پنجره New Project.



شکل ۲ – ۷ پنجره Form.



۲. گزینه View / Toolbox را کلیک کنید تا لیست کنترل‌ها را ببینید (شکل ۳ – ۷).

۳. برروی کنترل‌های مورد نظر کلیک مضاعف کنید تا آن‌ها به فرم

انتقال یابند. به عنوان مثال، در این برنامه یک کنترل Label، یک کنترل TextBox و دو کنترل Button را با کلیک مضاعف به فرم اضافه نمودیم.

۴. مکان کنترل‌ها را با کشیدن و رها کردن تنظیم کنید تا شکلی مانند زیر ظاهر شود:

شکل ۳ – ۷ پنجره Toolbox.

> برای حذف کنترل می‌توانید آن را کلیک کرده تا انتخاب شود. سپس، کلید Del را فشار دهید و برای برگرداندن کنترل حذف شده گزینه Edit / Undo را اجرا کنید.

۵. خواص کنترل‌ها را تعیین کنید (در ادامه تنظیم خواص کنترل‌ها را می‌آموزیم).

۶. رویدادهایی که کنترل‌ها باید به آن‌ها پاسخ دهند را بنویسید. در ادامه روش اضافه کردن رویداد برای کنترل‌ها و فرم را می‌آموزیم.

۷. پروژه را ذخیره کنید. برای این منظور گزینه File / Save All را کلیک کنید.

۸. پروژه را اجرا کنید. برای انجام این کار، گزینه Debug /Start Debugging (کلید F5) را اجرا کنید.

جدول ۱-۷ 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.
BackColor	رنگ زمینه فرم را تعیین می‌کند.
BackgroundImage	تصویر زمینه فرم را تعیین می‌نماید.
BackgroundImageLayout	نحوه قرار گرفتن تصویر در فرم را تعیین می‌کند. اگر Stretch تعیین شود، تصویر کل فرم را می‌پوشاند.
Cursor	شکل مکان نما را تعیین می‌کند. وقتی کرسر به فرم انتقال یافت، آن مکان نمایاید نمایش داده شود.
CancelButton	دکمه‌ایی را تعیین می‌کند که اگر کاربر کلید Esc را فشاردهد، رویداد Click آن دکمه اجرا می‌شود.
ControlBox	تعیین می‌کند که آیا دکمه‌های بستن، بیشینه، کمینه و منوی فرم نمایش داده شود یا خیر.
Icon	آیکن فرم را تعیین می‌کند. وقتی فرم کمینه گردید، این آیکن در نوار وظیفه ویندوز نمایش داده می‌شود.
MaximizeBox	تعیین می‌کند که آیا دکمه بیشینه فرم نمایش داده شود یا خیر.
Opacity	شفافیت فرم را تعیین می‌کند که عددی بین صفر تا ۱۰۰ است.
RightToLeftLayout	جهت نمایش Layout دکمه‌ها منوی فرم را تعیین می‌کند (چپ به راست باشد یا راست به چپ).
WindowState	نحوه نمایش فرم را تعیین می‌کند که مقادیر Normal(عادی)، Minimized (کمینه) و Maximized (بیشینه) را می‌پذیرد.

جدول ۲-۷ 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.

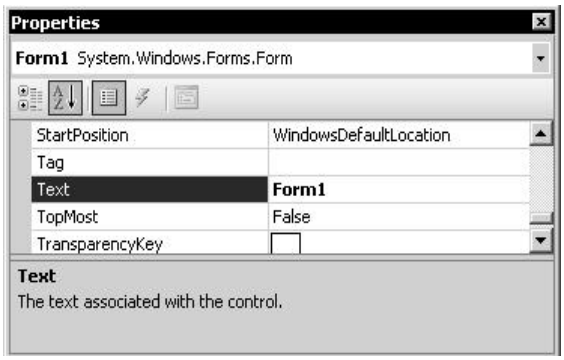
Enter	وقتی مکان‌نما وارد فرم می‌شود، رخ می‌دهد.
KeyDown	وقتی کلید فشرده شده پایین باشد، رخ می‌دهد.
KeyPress	وقتی کلیدی فشرده می‌شود، رخ می‌دهد. این رویداد قبل از رویداد KeyDown اتفاق می‌افتد.
KeyUP	وقتی کلید فشرده شده رها می‌گردد، رخ می‌دهد.
Leave	وقتی مکان‌نما فرم را ترک می‌کند رخ می‌دهد.
Load	وقتی 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. فرم برنامه

فرم برنامه، مکانی است که کنترل‌های برنامه در آن قرار می‌گیرند. هر برنامه تحت ویندوز 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.

۱-۳-۷. خواص فرم

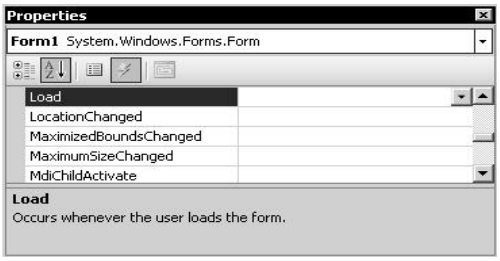
خواص فرم، شکل ظاهری فرم را تعیین می‌کنند. فرم خواص متعددی دارد. بیان همه این خواص از حوصله این کتاب خارج است. لذا، به تشریح خواص مهم فرم و کنترل‌های دیگر می‌پردازیم. برخی از خواص مهم فرم در جدول ۲-۷ آمده است. برای نمایش خواص فرم، بر روی آن کلیک کنید و سپس کلید F4 را بزنید تا پنجره خواص را ببینید (شکل زیر):



۲-۳-۷. رویدادهای فرم

همان‌طور که بیان شده، برنامه‌های ویژوال منتظر رخ دادن رویدادهایی هستند که توسط کاربر انتخاب می‌شوند تا به آنها پاسخ دهند. یعنی، اگر برنامه پاسخ‌گو به رویداد نوشته شده باشد، در صورت انتخاب آن رویداد توسط کاربر، این برنامه اجرا می‌شود. فرم دارای رویدادهای مختلفی است. برخی از مهم‌ترین رویدادهای فرم در جدول ۳-۷ آمده‌اند.

برای مشاهده رویدادهای فرم، بر روی آن کلیک کنید تا فرم انتخاب شود. سپس 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 ۳-۷ متدهای مهم فرم.

متد	هدف
Active	فرم را فعال می‌کند.
Close	فرم را می‌بندد.
Dispose	فرم را حذف کرد و 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.
Equals	تعیین می‌کند آیا فرم با شیای برابر است یا خیر.

۳-۳-۷. متدهای فرم

متدها، کارهای خاصی را بر روی فرم انجام می‌دهند. برخی از متدهای مهم فرم، در جدول ۳-۷ آمده‌اند.

مثال ۱-۷. برنامه‌ای که عنوان فرم برنامه را "اولین برنامه" تعیین می‌کند و با کلیک برروی فرم رنگ زمینه فرم آبی خواهد شد. چنانچه مکان‌نما از فرم خارج شود، رنگ فرم سبز می‌گردد و با ورود مکان‌نما به فرم رنگ آن قرمز خواهد شد (هدف این برنامه، آشنایی با فرم، اضافه کردن رویدادهای مختلف به فرم است).

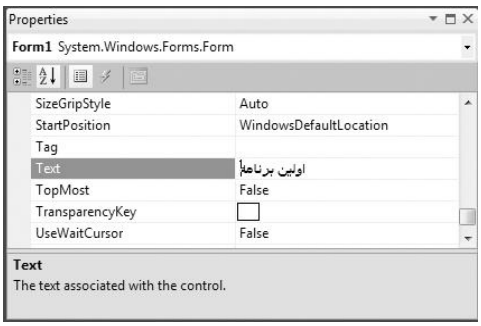
مراحل طراحی و اجرا

۱. گزینه File/New/Project را اجرا کنید تا پنجره New Project ظاهر شود (شکل ۱-۷). در این پنجره اطلاعات زیر را انتخاب کنید:

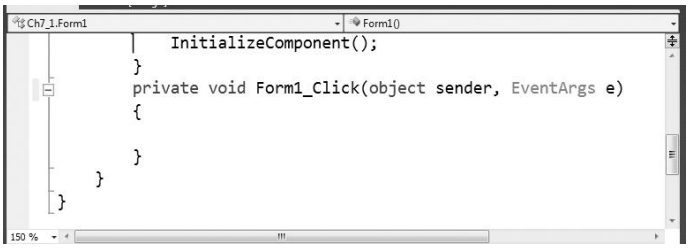
```
Windows Forms Application
Name: Ch7-1
Location:C:\Book\CSharp\7
```

دکمه OK را کلیک کنید.

۲. Text آن " اولین برنامه" را تایپ کنید (شکل زیر):



۳. فرم برنامه را کلیک کنید. کلید F4 را فشاردهید تا پنجره Properties ظاهر شود. در پنجره Propertis، دکمه  را کلیک کنید تا لیست رویدادهای فرم را ببینید. رویداد Click را پیدا کرده و جلوی رویداد Click، کلیک مضاعف کرده تا رویداد Form1_Click به برنامه اضافه شود (شکل زیر):



دستورات رویداد Form1_Click را بین } و { تایپ کنید. چون می‌خواهید با کلیک روی فرم رنگ زمینه فرم آبی شود، دستورات زیر را در این رویداد تایپ کنید:

```
BackColor = Color.Blue;
```

۴. اکنون، دستور Form1_Click به صورت زیر تغییر می‌یابد:

```
private void Form1_Click(object sender, EventArgs e)
{
    BackColor = Color.Blue;
}
```

وقتی که فرم کلیک شود این دستورات اجرا می‌شوند تا رنگ زمینه فرم آبی گردد.

۵. گزینه View/Designer را اجرا کنید تا طراح فرم ظاهر شود. کلید F4 را فشار دهید تا پنجره Properties را مشاهده کنید.

۶. رویداد MouseLeave را پیدا کرده، جلوی آن را کلیک مضاعف کنید تا رویداد Form1_MouseLeave به برنامه اضافه گردد. اکنون دستورات آن را به صورت زیر تغییر دهید:

```
private void Form1_MouseLeave(object sender, EventArgs e)
{
    BackColor = Color.Green;
}
```

این رویداد موجب می‌شود تا اگر اشاره‌گر ماوس از فرم خارج شود، رنگ زمینه فرم سبز خواهد شد.

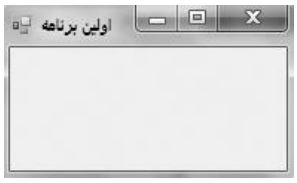
۷. گزینه View/Designer را کلیک کنید تا طراح فرم ظاهر شود. در پنجره خواص، رویداد MouseEnter را پیدا کرده، جلوی آن کلیک مضاعف کرده تا رویداد Form1_MouseEnter به برنامه اضافه شود. اکنون دستورات آن را به صورت زیر تغییر دهید:

```
private void Form1_MouseEnter(object sender, EventArgs e)
{
    BackColor = Color.Red;
}
```

این رویداد موجب می‌شود اگر مکان‌نما وارد فرم گردد، رنگ زمینه فرم قرمز شود.

۸. پروژه را ذخیره کنید. برای این منظور، گزینه File/Save All را اجرا کنید.

۹. پروژه را اجرا کنید. برای این منظور کلید F5 را فشار دهید تا شکل ۴ – ۷ ظاهر شود. برروی فرم کلیک کنید تا رنگ زمینه فرم آبی شود (مانند شکل ۵ – ۷).



شکل ٤ – ۷ خروجی مثال ۱ – ۷ پس از اجرا.

شکل ٥ – ۷ خروجی مثال ۱ – ۷ پس از کلیک برروی فرم.

مکان‌نما را به بیرون از فرم انتقال دهید. اکنون رنگ زمینه فرم سبز خواهد شد و با انتقال مکان‌نما به درون فرم، رنگ زمینه فرم به قرمز تغییر می‌یابد. دکمه Close را کلیک کنید تا اجرای برنامه خاتمه یابد.

۸–۲–۷. 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 CheckedListBox1 را تیک‌دار می‌کند (اندیس گزینه‌ها از صفر شروع می‌شود).

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

مثال ۵– ۷. برنامه‌ای که اعمال زیر را انجام می‌دهد:

- توسط متدی ۲۰ عدد تصادفی بین ۱ تا ۲۰ به listBox اضافه می‌نماید (نام ListBox را به عنوان پارامتر دریافت می‌کند).

- عددی را خوانده، توسط متد دیگری کلیه وقوع عدد را در یک ListBox حذف می‌کند (نام ListBox را از طریق پارامتر دریافت می‌کند).
- چنانچه ListBoxای کلیک مضاعف گردد، کلیه اطلاعات انتخاب شده آن به ListBox دیگر انتقال می‌یابد (ListBox اول و دوم را از طریق پارامتر به متد ارسال می‌کند).

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام Ch7_5 از نوع Windows Forms Application ایجاد کنید.

۲. یک کنترل Label به فرم اضافه کرده، خاصیت Text آن را **"عدد"** تعیین کنید.

۳. یک کنترل TextBox به فرم اضافه نمایید.

۴. یک کنترل CheckBox به فرم اضافه کرده، خاصیت Text آن را عبارت **"لیست یک یا دو"** تعیین کنید.

۵. دو کنترل ListBox به فرم اضافه کنید.

۶. سه کنترل Button به فرم اضافه کرده، خاصیت Text آن‌ها را عبارت **"ایجاد اعداد"**، **"حذف"** و **"خروج"** تعیین کنید.

۷. گزینه View/Code را اجرا کنید تا کد برنامه را ببینید. بعد از بلاک { initializeComponent متد FillListBox() را به صورت زیر اضافه کنید:

```
private void FillListBox(ListBox list1)
{
    Random r = new Random();
    list1.Items.Clear();
    for(int i=1; i<=20; i++)
     list1.Items.Add(r.Next(1, 20));
}
```

این متد list1 را از نوع ListBox به عنوا ن پارامتر دریافت کرده، ۲۰ عدد تصادفی تولید می‌کند و در list1 اضافه می‌نماید.

۸. متد MoveListBox() را بعد از متد FillListBox() به صورت زیر تعریف کنید:

```
private void MoveListBox(ListBox list1, ListBox list2)
{
 if (list1.SelectedItems.Count > 0)  {
  object[] selectedItem=new object[list1.SelectedItems.Count];
  list1.SelectedItems.CopyTo(selectedItem, 0);
  foreach (object i in selectedItem) {
    list1.Items.Remove(i);
    list2.Items.Add(i);
  }
```

```
  }
}
```

این دستورات، اگر گزینه‌ای از list1 انتخاب شده باشد، آن را به list2 اضافه می‌نماید. برای این منظور، ابتدا گزینه‌های انتخاب شده list1 را در آرایه selectedItem قرار می‌دهد و سپس عناصر آرایه را یکی یکی در list2 اضافه کرده، از list1 حذف می‌کند.

۹. متد RemoveAll() را بعد از متد MoveListBox() به صورت زیر تعریف کنید:

```
private void RemoveAll(ListBox list1, int value)
{
 do {
     int index = list1.Items.IndexOf(value);
     if (index == -1) break;
     list1.Items.RemoveAt(index);
    } while (true);
}
```

این متد list1 و value را به عنوان پارامتر دریافت کرده، کلیه تکرار value را در list1 حذف می‌کند. برای این منظور در داخل یک حلقه بی‌نهایت، با متد IndexOf(value) را در list1 جستجو می‌کند، اگر پیدا نکرد، حلقه خاتمه می‌یابد، وگرنه مکان value را در index قرار می‌دهد و با RemoveAt آن را از listBox1 حذف می‌نماید.

۱۰. دکمه "**ایجاد اعداد**" را کلیک مضاعف کرده، دستورات رویداد button1_Click آن را به صورت زیر تغییر دهید:

```
private void button1_Click(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
       FillListBox(listBox1);
    else
      FillListBox(listBox2);
}
```

اگر checkBox1 انتخاب شده باشد، با فراخوانی متد FillListBox()، ۲۰ عدد تصادفی تولید کرده، به ListBox1 اضافه می‌کند، وگرنه ۲۰ عدد تصادفی تولید کرده، به listBox2 اضافه می‌نماید.

۱۱. دکمه "**حذف**" را کلیک مضاعف کرده، دستورات رویداد button2_Click() را به صورت زیر تغییر دهید:

```
private void button2_Click(object sender, EventArgs e)
{
    int delItem = Convert.ToInt32(textBox1.Text);
    if (checkBox1.Checked == true)
       RemoveAll(listBox1, delItem);
```

```
    else
        RemoveAll(listBox2, delItem);
}
```

این دستورات ابتدا عددی را خوانده، در delItem قرار می‌دهد. سپس، اگر CheckBox1 انتخاب شده باشد، با فراخوانی متد ()RemoveAll، مقدار delItem را از listBox1 حذف می‌کند، وگرنه delItem را از ListBox2 حذف می‌نماید.

۱۲. رویداد DoubleClick را به listBox1 اضافه کنید. برای این منظور، listBox1 را انتخاب کرده، کلید F4 را فشار دهید تا پنجره خواص ظاهر شود. دکمه Events را کلیک کنید تا رویدادهای listBox1 ظاهر شود. رویداد DoubleCkick را پیدا کرده، جلوی آن را کلیک مضاعف نمایید و دستورات آن را به صورت زیر تغییر دهید:

```
private void listBox1_DoubleClick(object sender,EventArgs e)
{
    MoveListBox(listBox1, listBox2);
}
```

این دستورات، رویداد ()MoveListBox را فراخوانی می‌کنند تا گزینه‌های انتخاب شده listBox1 به listBox2 منتقل شوند.

۱۳. رویداد ()DoubleClick را مانند مرحله ۱۲ برای listBox2 اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
private void listBox2_DoubleClick(object sender,EventArgs e)
{
    MoveListBox(listBox2, listBox1);
}
```

این دستورات، متد ()MoveListBox را فراخوانی می‌کنند تا گزینه‌های انتخاب شده listBox2 به listBox1 منتقل شوند.

۱۴. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد ()Form1_Load آن را به صورت زیر تغییر دهید:

```
private void Form1_Load(object sender, EventArgs e)
{
   listBox1.SelectionMode = SelectionMode.MultiSimple;
   listBox2.SelectionMode = SelectionMode.MultiSimple;
}
```

این دستورات موجب می‌شوند تا کاربر بتواند چند گزینه listBox1 و listBox2 را انتخاب کند.

۱۵. دکمه **"خروج"** را کلیک کرده و دستورات رویداد button2_Click را به صورت زیر تغییر دهید:

```
private void button3_Click(object sender, EventArgs e)
{
     Close();
}
```

۱۶. پروژه را ذخیره و اجرا کنید. دکمه **"ایجاد اعداد"** را کلیک نمایید تا (شکل ۱۳ – ۷) ظاهر شود. ۱۵ را جلوی عدد وارد کرده، دکمه حذف را کلیک کنید تا کلیه عدد ۱۵ از listBox2 حذف شوند (شکل ۱۴ – ۷)، عدد ۲ و ۱۹ را کلیک مضاعف کنید تا به listBox1 انتقال یابند (شکل ۱۵ – ۷).

شکل ۱۳ – ۷ خروجی پس از کلیک روی دکمه ایجاد اعداد.

شکل ۱۴ – ۷ خروجی پس از حذف تمام مقادیر ۱۵ از listBox2.

شکل ۱۵ – ۷ انتقال مقادیر ۲ و ۱۹ با کلیک مضاعف.

۴–۷. مسائل حل شده

مثال ۱– ۷. برنامه‌ای که با حرکت ماوس برروی فرم، مختصات فعلی اشاره‌گر (مکان‌نما) ماوس را در عنوان فرم نمایش می‌دهد.

مراحل طراحی و اجرا

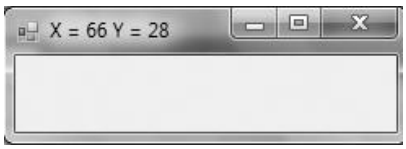
۱. پروژه جدیدی به نام SolveCh7_1 از نوع Windows Forms Application ایجاد کنید.

۲. رویداد MouseMove را به فرم اضافه کنید. برای این منظور، کلید F4 را فشار دهید. دکمه Events را کلیک کرده تا رویدادهای فرم را ببینید. رویداد MouseMove را پیدا نموده جلوی آن را کلیک مضاعف نمایید و دستورات آن را به صورت زیر تغییر دهید:

```
private void Form1_MouseMove(object sender,MouseEventArgs e)
{
  Text="X="+e.X.ToString() + "\t" + " Y = " + e.Y.ToString();
}
```

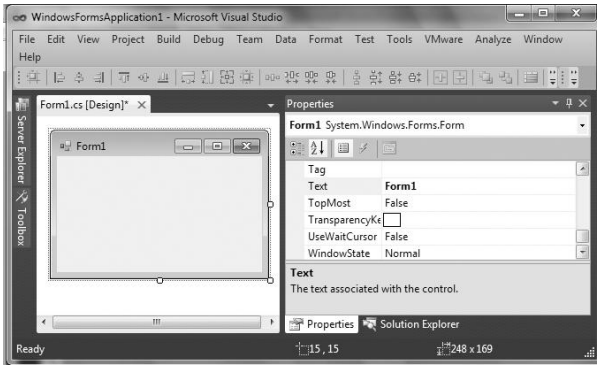
این دستورات، با حرکت ماوس روی فرم مکان فعلی مکان‌نما را برروی عنوان فرم نمایش می‌دهند.

۳. پروژه را ذخیره و اجرا کنید. مکان‌نما را برروی فرم حرکت دهید تا مکان اشاره‌گر فرم را برروی عنوان فرم ببینید (مانند شکل ۲۴ – ۷).



شکل ۲۴ – ۷ نمونه خروجی مثال ۱ – ۷.

مثال ۲– ۷. برنامه‌ای که کنترل‌ها و منوی عنوان فرم را غیر فعال می‌کند و چنانچه اندازه فرم تغییر یابد، عنوان فرم به عبارت " در حال تغییر اندازه فرم" تغییر می‌یابد و با فشردن کلید ESC فرم بسته خواهد شد.



مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh7_2 از نوع Windows Forms Application ایجاد کنید.

۲. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Load آن را به صورت زیر تغییر دهید:

```
private void Form1_Load(object sender, EventArgs e)
{
    ControlBox = false;
}
```

این دستورات کنترل‌ها و منوی عنوان فرم را پنهان می‌کنند.

۳. رویداد ResizeBegin را برای فرم اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
private void Form1_ResizeBegin(object sender, EventArgs e)
{
    Text = "در حال تغییر اندازه فرم";
}
```

۴. رویداد KeyPress را برای فرم اضافه کرده، دستورات رویداد آن را به صورت زیر تغییر دهید:

```
private void Form1_KeyPress(object sender,KeyPressEventArgs e)
{
  if ((int)e.KeyChar == 27) Close();
}
```

این دستورات چنانچه کاربر برروی فرم کلید ESC را فشار دهد، برنامه خاتمه می‌یابد.

۵. پروژه را ذخیره و اجرا کنید. اندازه فرم را تغییر دهید تا خروجی به (شکل ۲۵ – ۷) تغییر یابد. همان‌طور که در این فرم می‌بینید، دکمه‌های کنترل و منوی عنوان فرم حذف شده است. برای خروجی از برنامه کلید ESC را فشار دهید.

شکل ۲۵ – ۷ نمونه خروجی مثال ۲ – ۷.

مثال ۳– ۷. برنامه‌ای که در یک TextBox فقط اعداد منفی یا مثبت را بپذیرد. اگر کاربر عدد منفی وارد نمود، قدرمطلق آن را محاسبه کرده و سپس تمام اعداد کامل (تام) قبل از عدد خوانده شده را به یک ListBox اضافه می‌نماید (عدد ی کامل است که مجموع مقسوم علیه‌های کوچکتراز خودش برابر با خودش باشد).

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh7_3 از نوع Windows Forms Application ایجاد کنید.

۲. یک کنترل Label، یک کنترل TextBox، یک کنترل ListBox و دو کنترل Button به فرم اضافه کنید. خاصیت Text کنترل Label را "عدد" تعیین کنید و خاصیت Text دو دکمه را به ترتیب "نمایش" و "خروج" تعیین نمایید.

۳. رویداد KeyPress را برای کنترل textBox1 اضافه کرده، دستورات رویداد آن را به صورت زیر تغییر دهید:

```
private void textBox1_KeyPress(object sender,
     KeyPressEventArgs e)
 {
  if (textBox1.Text.Length == 0 && e.KeyChar == '-')
        e.Handled = false;
   else if (!(char.IsControl(e.KeyChar) || char.IsDigit(e.KeyChar)))
```

```
        e.Handled = true;
}
```

این دستورات موجب می‌شوند تا کاربر فقط بتواند اعداد منفی (کارکترهای به جز -، ارقام و کنترل) را نمی‌پذیرد. البته کارکتر منفی را وقتی می‌پذیرد که اولین کارکتر textBox1 باشد.

۴. دکمه "**نمایش**" را کلیک مضاعف کرده، دستورات رویداد button1_Click آن را به صورت زیر تغییر دهید:

```
private void button1_Click(object sender, EventArgs e)
{
    int n = Convert.ToInt32(textBox1.Text);
    if (n < 0) n = -n;
    listBox1.Items.Clear();
    for (int i = 1; i <= n; i ++)   {
       int sum = 0;
       for (int j = 1; j <= i / 2; j++)
       if (i % j == 0) sum += j;
       if(sum == i)   listBox1.Items.Add(i);
   }
}
```

این دستورات ابتدا مقدار موجود در textBox1 را به عدد تبدیل می‌کنند. اگر عدد منفی باشد آن را به عدد مثبت تبدیل خواهند کرد. سپس، اعداد تام از ۱ تا عدد خوانده شده را پیدا کرده به listBox1 اضافه می‌کنند.

۵. دکمه "**خروج**" را کلیک مضاعف کرده، دستورات رویداد button2_Click را به صورت زیر تغییر دهید:

```
private void button2_Click(object sender, EventArgs e)
{
    Close();
}
```

۶. پروژه را ذخیره و اجرا کنید. عدد ۹۹- را وارد کرده و دکمه "**نمایش**" را کلیک کنید تا خروجی را به (شکل ۲۶ – ۷) مشاهده کنید (در هنگام ورود به جزء کارکترهای صفر تا ۹، کارکتر کنترلی و – نمی‌توانید کارکتر دیگری وارد نمایید).

شکل ۲۶ – ۷ نمونه خروجی مثال ۳ – ۷,

۵–۷. مسائل حل شده در سایت

۱. برنامه‌ای که بردار اطراف فرم را در ComboBoxای اضافه می‌کند (مقادیر خاصیت FormBorderStyle) و کاربر می‌تواند از طریق ComboBox مقدار خاصیت را برای فرم انتخاب کند.

۲. برنامه‌ای که مکان شروع فرم (مقدار خاصیت FormStartPosition) را در ComboBoxای اضافه می‌کند و کاربر می‌تواند مکان قرار گرفتن فرم را انتخاب کند.

۳. برنامه‌ای که از طریق سه کنترل CheckBox حالت‌های بیشینه، کمینه و نمایش یا عدم نمایش منوها و دکمه‌های روی فرم را فعال یا غیر فعال می‌کند.

۴. برنامه‌ای که رنگ‌های مختلف را به CheckListBox اضافه می‌کند، به طوری که کاربر می‌تواند برخی از رنگ‌ها را انتخاب کند. سپس از انتخاب رنگ‌های مورد نظر، اگر کاربر دکمه‌ای را فشار دهد، رنگ‌های انتخاب شده توسط کاربر را نمایش می‌دهد (هدف این برنامه آشنایی با کنترل CheckListBox و متدهای آن است).

۵. برنامه‌ای که اعمال زیر را انجام می‌دهد.

- تعدادی عدد تصادفی تولید می‌کند و در listBox1 یا listBox2 اضافه می‌کند (این عمل را با متد FillListBox انجام می‌دهد که نام ListBox را به عنوان پارامتر دریافت می‌نماید).
- تمام داده (Value) موجود در یک ListBox (پارامتر list1) را حذف می‌کند (این عمل از طریق متد RemoveAll() انجام می‌شود).
- با کلیک مضاعف داده‌های یک listBox (پارامتر list1) به ListBox دیگر (پارامتر list2) انتقال می‌دهد. این عمل از طریق متد MoveListBox() انجام می‌شود.

۶. برنامه‌ای که با استفاده از دو CheckBox کاربر می‌تواند حالت‌های MultiLine یا ReadOnly بودن یک TextBox را انتخاب کند.

۷. برنامه‌ای که nامین تکرار مقدار عددی m را در یک ListBox پیدا کرده و نمایش می‌دهد (این برنامه اعداد را به صورت تصادفی تولید می‌کند و به ListBox اضافه می‌نماید).

۸. برنامه‌ای که nامین عدد m را از ListBox حذف می‌کند (این برنامه، برای پیدا کردن nامین مقدار عدد m در listBox، متد Find() را پیاده‌سازی کرده است).

۹. برنامه‌ای که مکان‌های عناصر انتخاب شده یک ListBox را به ListBox دیگر اضافه می‌کند. (این برنامه متد addSelectedIndex() را پیاده‌سازی می‌کند که list1 و list2 را از نوع ListBox به عنوان پارامتر دریافت می‌کند).

۱۰. برنامه‌ای که تعدادی عدد تصادفی تولید کرده، در کنترل‌های ListBox، ComboBox و CheckListBox اضافه می‌کند. سپس با استفاده از متد ()RemoveAll کلیه گزینه‌های کنترل‌های CheckListBox، ComboBox و ListBox را حذف می‌کند (این برنامه سه متد همنام به نام ()RemoveAll را پیاده‌سازی کرده است که پارامترهای مختلف از نوع ListBox، ComboBox و CheckListBox را دریافت می‌کنند).

۱۱. برنامه‌ای که تعدادی گزینه به یک CheckListBox اضافه می‌کند و سپس توسط دکمه‌ای تمام گزینه‌های CheckListBox را انتخاب می‌نماید.

۱۲. برنامه‌ای که تعدادی گزینه به یک CheckListBox اضافه کرده، سپس توسط دکمه‌ای انتخاب گزینه‌ها را معکوس می‌کند (یعنی، گزینه‌های انتخاب شده را از حالت انتخاب خارج کرده، گزینه‌هایی که انتخاب نشده باشند را انتخاب می‌نماید).

۶-۷. تمرین

۱. برنامه‌ای بنویسید که سه کنترل Button را به فرم اضافه کرده، با کلیک دکمه button1، اندازه فرم حداکثر گردد. با کلیک دکمه button2، اندازه فرم به حالت نرمال برگردد و با کلیک دکمه button3، اندازه فرم کمینه می‌شود. چنانچه مکان‌نمای ماوس از محدوده‌ی فرم خارج شود، رنگ زمینه فرم آبی گردد و اگر مکان‌نما به محدوده فرم برگردد (وارد شود)، رنگ زمینه فرم قرمز شود.

۲. برنامه‌ای بنویسید که اعمال زیر را انجام دهد:

- عبارت " فرم اصلی" را در سمت راست عنوان فرم نمایش دهد.
- با کلیک روی فرم، عنوان فرم را به سمت چپ برده و با کلیک دیگر عنوان به سمت راست برود و این روند ادامه یابد.
- دارای دو کنترل TextBox و شش کنترل Button باشد.
- با کلیک button1، اطلاعات textBoxt2 به textBox1 اضافه شود.
- با کلیک دکمه button2، اطلاعات textBox1 به textBox2 اضافه گردد.
- با کلیک دکمه button3، اطلاعات انتخاب شده کنترل textBox2 حذف گردد.
- با کلیک دکمه button4، رنگ زمینه فرم به سبز تغییر یابد.
- با کلیک دکمه button5، رنگ زمینه فرم به خاکستری تبدیل شود.

- باکلیک دکمه button6، یک تصویر در زمینه فرم قرار گیرد.

۳. برنامه‌ای که اعمال زیر را انجام دهد:

- یک کنترل Label به فرم اضافه کند.
- اگر فرم کلیک گردد، عبارت Form Clicked برروی label1 و عنوان فرم نوشته شود.
- اگر مکان‌نما فرم را ترک کند، رنگ زمینه label سفید و رنگ زمینه فرم آبی شود.
- اگر مکان‌نما وارد label شود، رنگ زمینه فرم قرمز و رنگ نوشته label، سفید شود.
- اگر بر روی فرم کلیدی فشرده شود، بر روی عنوان فرم و label عبارت Key Pressed نوشته شود.
- اگر اندازه فرم تغییر یابد، در عنوان فرم و label1 عبارت Resizing با رنگ سبز نوشته شود.
- وقتی مکان‌نما بر روی فرم حرکت می‌کند رنگ زمینه فرم سبز شده و در عنوان فرم و label1 عبارت Mouse Moved نوشته شود.
- اگر label کلیک گردید، برنامه خاتمه یابد.

فصل ۸ ایجاد برنامه‌های پیشرفته کاربردی در فرم

در فصل ۷ با برنامه‌های تحت ویندوز دارای فرم آشنا شدید. در آن فصل روش افزودن کنترل‌ها به فرم، تغییر خواص، ایجاد رویدادها و فراخوانی متدهای آن‌ها را دیدید. همچنین وظایف بعضی از کنترل‌ها را آموختید. در این فصل برخی از کنترل‌های دیگر را می‌بینید. در ادامه این کنترل‌ها، خواص، رویدادها و متدهای آن ها را مشاهده خواهید کرد.

۸-۱. کنترل Timer (Timer)

این کنترل اجازه می‌دهد تا رویداد TimerTick را در فواصل زمانی منظم احضار کنید. خواص، متدها و رویدادهای کنترل Timer در زیر آمده‌اند:

- **خاصیت Interval**، فواصل زمانی بین رویدادهای Tick را به میلی ثانیه تعیین می‌کند. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
timer1.Interval = 2000;
```

این دستور موجب می‌شود تا هر دو ثانیه رویداد Tick اتفاق افتد.

- **خاصیت Enabled**، تعیین می‌کند که آیا این کنترل رویداد Tick را احضار کند یا خیر. اگر مقدار این خاصیت true باشد، کنترل به رویداد Tick پاسخ خواهد داد، وگرنه به رویداد Tick پاسخ نمی‌دهد.
- **متد ()Start**، برای شروع کار Timer به کار می‌رود.
- **متد ()Stop**، برای خاتمه دادن به کار Timer به کار می‌رود.

۸-۲. کنترل ProgressBar (ProgressBar)

این کنترل میزان عملیات انجام شده کاری را به صورت ویژوالی نمایش می‌دهد. به عنوان مثال، در هنگام کپی نمودن فایل‌ها در ویندوز درصد پیشرفت کار را می‌بینید. درصد پیشرفت کار کپی در ویندوز توسط کنترل ProgressBar نمایش داده می‌شود. برخی از خواص کنترل ProgressBar را در زیر می‌بینید:

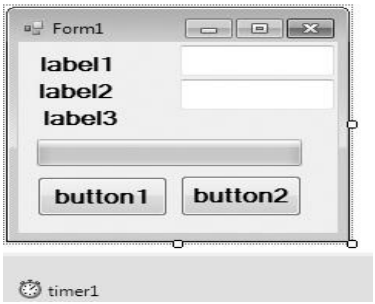
خاصیت Maximum، حداکثر مقدار کنترل ProgressBar را تعیین می‌کند. وقتی مقدار خاصیت Value کنترل برابر این مقدار شود، درصد پیشرفت کار کنترل ProgressBar کامل می‌شود.

خاصیت Minimum، حداقل مقدار کنترل ProgressBar را تعیین می‌کند. وقتی مقدار خاصیت Value برابر این مقدار باشد، درصد پیشرفت کار صفر خواهد بود.

خاصیت Step، گام (میزان) افزایش مقدار خاصیت Value را تعیین می‌کند. مقدار این خاصیت می‌تواند بین خواص Maximum و Minimum باشد.

مثال ۱–۸. برنامه‌ای که نام کاربر و کلمه عبور را دریافت کرده، نحوه احراز هویت کاربر را با یک کنترل ProgressBar و Timer نشان می‌دهد(هدف این برنامه آشنایی با کنترل‌های Timer و ProgressBar است).

مراحل طراحی و اجرا



۱. پروژه جدیدی به نام Ch8_1 از نوع Windows Forms Application ایجاد کنید.

۲. سه کنترل Label، دو کنترل TextBox، یک کنترل Timer، یک کنترل ProgressBar ودو کنترل Button به فرم اضافه کنید.

۳. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Form1_Load را به صورت زیر تغییر دهید:

```
private void Form1_Load(object sender, EventArgs e)
{
 textBox2.PasswordChar = '*';
 label3.Visible = false;
 label1.Text = "Use name:";
 label2.Text = "Password:";
 button1.Text = "login";
 button2.Text = "Exit";
}
```

این دستورات، خواص کنترل‌های روی فرم را مقدار اولیه می‌دهند.

٤. کنترل timer1 را کلیک مضاعف کرده تا رویداد timer1_Tick به پروژه اضافه شود. اکنون دستورات آن را به صورت زیر تغییر دهید:

```
private void timer1_Tick(object sender, EventArgs e)
{
  progressBar1.Visible = true;
```

```
    progressBar1.Value = progressBar1.Value + 5;
    label3.Visible = true;
    label3.Text = "Please Wait While we are checking Authentication...";
    if (progressBar1.Value == progressBar1.Maximum) {
    if ((textBox1.Text=="fanavarienovin")&&(textBox2.Text == "123456"))
        label3.Text = "Welcome!!.";
    else
        label3.Text = "Sorry!! Username or Password is Wrong.";
        timer1.Enabled = false;
        progressBar1.Visible = false;
        progressBar1.Enabled = false;
        progressBar1.Value = ;
        label3.Visible = true;
  }
}
```

این دستورات ProgressBar را فعال می‌کنند، ۵ واحد به مقدار Value کنترل ProgressBar اضافه کرده، label3 را فعال می‌کنند و یک پیغام را در آن نمایش می‌دهند. سپس مقدار Value کنترل ProgressBar را با مقدار Maximum خودش چک می‌کند. اگر این دو مقدار برابر باشند، مقدار وارد شده در کنترل‌های textBox1 و textBox2 را با مقدار نام کاربر(fanavarienovin) و کلمه عبور (123456) چک می‌کند، اگر نام کاربر و کلمه عبور صحیح وارد شده باشند، عبارت" !!Welcome" ، وگرنه عبارت "Sorry !! User name or Password is Wrong" را در label3 نمایش می‌دهد. در پایان، progressBar1 و timer1 را غیرفعال کرده، مقدار Value کنترل progressBar1 را صفر تعیین می‌کند و خاصیت Visible کنترل label3 را به true تغییر می‌دهد تا label3 و محتوی آن نمایش داده شود.

۵. دکمه button1 را کلیک مضاعف کرده، دستورات رویداد Click را به صورت زیر تغییر دهید:

```
private void button1_Click(object sender, EventArgs e)
{
    timer1.Enabled = true;
}
```

این دستورات timer1 را فعال می‌کنند.

٦. دکمه button2 را کلیک مضاعف کرده، دستورات رویداد Click آن را به صورت زیر تغییر دهید:

```
private void button2_Click(object sender, EventArgs e)
{
   Close();
}
```

۷. پروژه را ذخیره و اجرا کنید. در textBox1 مقدار fanavarienovin و در textBox2 مقدار 1234 را وارد کرده و دکمه Login را کلیک کنید.

اکنون پیشرفت احراز هویت را به شکل مقابل می‌بینید:

پس از چند لحظه عبارت زیر بر روی lable13 ظاهر می‌شود:

Sorry!! Username or Password is Wrong.

اکنون در textBox2 مقدار 123456 را وارد نموده، دکمه Login کلیک نمائید تا عبارت زیر بر روی label3 نمایش داده شود:

Welcome!!.

۳-۸. کنترل TrackBar(TrackBar)

این کنترل، یک واسط ساده را فراهم می‌کند تا کاربر بتواند یک مقدار در یک بازه را انتخاب کند. این کنترل مقدار را توسط لغزنده‌ای مشخص می‌کند که می‌توان با ماوس یا کلیدهای صفحه کلید، لغزنده را حرکت داد تا مقدار خاصیت Value آن را تغییر کند. برخی از خواص این کنترل در زیر آمده‌اند.

- **خاصیت LargeChange**، تعداد مکان هایی که لغزنده [۱۴] در پاسخ به کلیک‌های ماوس و فشردن کلیدهای Page Up و Page Down حرکت می‌کند.
- **خاصیت Maximum**، حداکثر مقدار را برای کنترل TrackBar، تعیین می‌کند.
- **خاصیت Minimum**، حداقل مقدار را برای کنترل TrackBar تعیین می‌کند.
- **خاصیت SmallChange**، تعداد مکان‌هایی را تعیین می‌کند که لغزنده در پاسخ به فشردن کلیدهای جهت نما حرکت می‌کند.
- **خاصیت TickFrequancey**، تعداد مکان‌های بین علامت‌های تیک روی کنترل TrackBar را تعیین می‌کند.
- **خاصیت TrickStyle**، سبک تیک ها روی TrackBar را تعیین می‌کند.
- **خاصیت Value**، مقدار برگشت داده شده توسط TrackBar است.

مثال ۲-۸. برنامه‌ای که شفافیت فرم را ازطریق کنترل TrackBar تغییر می‌دهد(هدف این برنامه آشنایی با کنترل TrackBar است).

مراحل طراحی و اجرا

۱.پروژه جدیدی به نام Ch8_2 ایجاد کنید(نوع پروژه

1.Slider

را Windows Forms Application انتخاب کنید).

۲. یک کنترل TrackBar به فرم اضافه نمایید.

۳. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Form1_Load را به صورت زیر تغییر دهید:

```
private void Form1_Load(object sender, EventArgs e)
{
    BackColor = Color.Blue;
}
```

این متد رنگ زمینه فرم را آبی انتخاب می‌کنند.

٤. رویداد Scroll را برای کنترل trackBar1 اضافه کرده، دستورات رویداد trackBar1_Scroll را به صورت زیر تغییر دهید:

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
  if (trackBar1.Value >= 5)
   this.Opacity = (double)trackBar1.Value * 10 / 100;
}
```

این دستورات، اگر مقدار خاصیت Value کنترل trackBar1 بیشتر از ۵ باشد، آن را به درصد با نوع double تبدیل کرده، در خاصیت Opacity فرم قرار می‌دهد تا شفافیت فرم تغییر یابد(مقادیر زیر ۵ فرم را محو می‌کنند).

۵. پروژه را ذخیره و اجرا کنید تا خروجی مقابل را ببینید.

اکنون مقدار TrackBar را تغییر دهید تا خروجی را به شکل همان طور که در خروجی می‌بینید، شفافیت فرم تغییر کرده است، به طوری که نوشته‌های زیر فرم نیز نمایش داده شده‌اند.

۴-۸. کنترل MaskedTextBox (MaskedTextBox)

کنترل MaskedTextBox، تغییر یافته کنترل TextBox است که اجازه ورود داده‌ها با الگوی مشخص را می‌دهد. یعنی، در این کنترل می‌توان داده‌هایی با فرمت تاریخ، تلفن وغیره را وارد نمود. برخی از خواص این کنترل در زیر آمده‌اند:

- **خاصیت AllowPromptAsInput**، تعیین می‌کند آیا کارکتر اعلان (Prompt) ورودی معتبر است یا خیر؟

- **خاصیت BeepOnError**، تعیین می‌کند وقتی که کارکتر نامعتبر وارد شد، بوق سیستم به صدا درآید یا خیر؟
- **خاصیت CutCopyMaskFormat**، تعیین می‌کند آیا لیترال‌ها و کارکترهای اعلان شامل کپی یا برش (Cut) متن شوند یا خیر؟
- **خاصیت HidePromptOnLeave**، تعیین می‌کند وقتی که مکان نما MaskedTextBox را ترک می‌کند آیا کارکترهای اعلان مخفی شوند یا خیر؟
- **خاصیت InsertKeyMode**، حالت درج در متن را برای MaskedTextBox تعیین می‌کند.
- **خاصیت Mask**، قالبی را تعیین می‌کند که کنترل باید در آن قالب متن را دریافت نماید. در ادامه، تنظیم این خاصیت را می بینید.
- **خاصیت PromptChar**، کارکتر استفاده شده برای اعلان را تعیین می‌کند.
- **خاصیت ResetOnPrompt**، تعیین می‌کند که چگونه کارکتر ورودی با کارکترهای اعلان یکسان اداره شود.
- **خاصیت ResetOnSpace**، تعیین می‌کند چگونه باید کارکتر فضای خالی ورودی اداره شوند.
- **خاصیت SkipLitrals**، تعیین می‌کند که آیا لیترال‌ها در قالب رد شوند یا دوباره وارد گردند.
- **خاصیت TextMaskFormat**، تعیین می‌کند آیا کارکترهای لیترال و اعلان در خاصیت Text آورده شوند و توسط آن برگشت داده شوند یا خیر؟
- **خاصیت MaskedTextBox**، قالب (فرمت) دریافت رشته را تعیین می‌کند. برای این منظور از کارکترهای خاصی استفاده می کند که برخی از آن‌ها را در جدول ۱-۸ می بینید. بعضی از فرمت‌های تعیین شده و ورودی آن‌ها در جدول ۲-۸ آمده‌اند.

۵-۸. کنترل ToolTip(ToolTip)

این کنترل اطلاعاتی را برای کاربر فراهم می‌کند، وقتی که مکان نمای ماوس روی کنترل حرکت کرد آن را نمایش می‌دهد. برخی از خواص کنترل ToolTip در زیر آمده است:

- **خاصیت Active**، تعیین می‌کند که آیا کنترل فعال است یا خیر.
- **خاصیت AutomaticDelay**، مدت تاخیر خودکار کنترل را تعیین می‌کند.

خاصیت AutoPopDelay، مدت زمانی را تعیین می‌کند که مقدار Text این کنترل بر روی کنترل‌های دیگر باید قابل رویت باشد.

خاصیت InitialDelay، مدت زمانی را تعیین می‌کند که قبل از نمایش ToolTip باید بگذرد.

خاصیت ShowAlways، تعیین می‌کند که آیا همیشه یک پنجره ToolTip نمایش داده شود(حتی وقتی که کنترل پدر غیر فعال باشد) یا خیر.

خاصیت ToolTipTitle، عنوان کنترل ToolTip را تعیین می‌کند.

جدول ۱-۸ کارکترهای تعیین فرمت خاصیت Mask.

کارکتر	هدف
0	یک رقم اجباری ۰ تا ۹ را تعیین می‌کند (حتما باید یک رقم وارد شود).
9	یک رقم اختیاری بین ۰ تا ۹ را تعیین می‌کند.
#	یک رقم اختیاری بین ۰ تا ۹، + و- را تعیین می‌کند.
L	یک کارکتر اجباریَ A تاَ Z یاَ a تاَ z را تعیین می‌کند.
?	یک کارکتر اختیاری بینَ A تاَ Z یاَ a تاَ z را تعیین می‌کند.
&	یک کارکتر اجباری را تعیین می‌کند. اگر AsciiOnly مقدار true را داشته باشد، مانند L عمل می‌کند.
C	یک کارکتر اختیاری را تعیین می‌کند. اگر AsciiOnly، مقدار True را داشته باشد، مانند[?] عمل می‌کند.
A, a	یک کارکتر الفبایی عددی اختیاری را تعیین می‌کند. اگر AsciiOnly مقدار True را داشته باشد، آن فقط کارکترهایَ A تاَ Z یاَ a تاَ z را می‌پذیرد.
/	جداکننده تاریخ را تعیین می‌کنند.
$	نماد پولی را تعیین می‌کند
<	شیفت بالا را تعیین می‌کند. تمام کارکترهای بعد از آن به حروف کوچک تبدیل خواهند شد.
>	شیفت پائین را تعیین می‌کند. تمام کارکترهای بعد از آن به حروف بزرگ تبدیل خواهند شد.
|	کارکترهای <، > قبلی را غیر فعال می‌کند.
\	را رد می‌کند.\استفاده شود، کارکتر \یک کارکتر فرمت در داخل کارکتر لیترال رد می‌کند. اگر از

جدول ۲-۸ نمونه فرمت های تعیین شده و ورودی آن‌ها.		
فرمت	ورودی	چگونگی نمایش
(999)-000-0000	1234567890	(123)-456-7890
0000/00/00	13901101	1390/11/01
99,999.00	1234567	$12,345.67
LL>|LLL<LL	abcdABCD	AbCdABcd

خاصیت UseAnimation، تعیین می‌کند آیا یک افکت متحرک باید نمایش داده شود یا خیر؟

خاصیت UseFading، تعیین می‌کند یک افکت محو شونده فعال شود یا خیر؟

۲۴-۸. مسائل حل شده

مثال ۱-۸. برنامه‌ای که یک نمودار دایره‌ای رسم می‌کند(هدف این برنامه آشنایی بیشتر با گرافیک است)

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh8_1 ایجاد کنید.

۲. آرایه arrWeight را به صورت زیر تعریف کنید:

```
int[] arrWeight;
```

arrWeight، آرایه ای است که نمودار دایره‌ای بر اساس مقادیر آن رسم می‌شود.

۳. متد DrawSliceChart() را به صورت زیر تعریف کنید:

```
private void DrawSliceChart(PaintEventArgs e, int[] arrWeight)
{
  int numberOfSections = arrWeight.Length;
  int x0 = 30;
  int y0 = 30;
  int radius = 60;
  int startAngle = 0;
  int sweepAngle = 0;
  int[] height = new int[numberOfSections];
  int total = SumOfArray(arrWeight);
  Random rnd = new Random();
  SolidBrush brush = new SolidBrush(Color.Aquamarine);
  Pen pen = new Pen(Color.Black);
  for (int i = 0; i < numberOfSections; i++) {
    brush.Color = Color.FromArgb(rnd.Next(200, 255),
    rnd.Next(255), rnd.Next(255), rnd.Next(255));
    if (i == numberOfSections - 1)
       sweepAngle = 360 - startAngle;
    else
       sweepAngle = (360 * arrWeight[i]) / total;
            e.Graphics.FillPie(brush, x0 - height[i], y0 -
     height[i], 2 * radius, 2 * radius, startAngle, sweepAngle);
```

```
    e.Graphics.DrawPie(pen, x0 - height[i], y0 - height[i], 2   *
    radius, 2 * radius, startAngle, sweepAngle);
     startAngle += sweepAngle;
     brush.Color = Color.FromKnownColor(KnownColor.Black);
  }
}
```

این متد پارامتر e و آرایه arrWeight را دریافت کرده، نمودارهای دایره‌ای را با مقادیر آرایه رسم می‌نماید.

٤. متد SumOfArray() را به صورت زیر تعریف کنید:

```
private static int SumOfArray(int[] intArray)
{
  int sum = 0;
  for (int i = 0; i < intArray.Length; i++)
     sum += intArray[i];
  return sum;
}
```

این متد آرایه ای را به عنوان پارامتر دریافت کرده، مجموع عناصر آن را برمی‌گرداند.

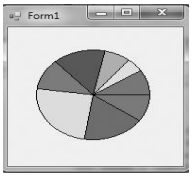
٥. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Form1_Load را به صورت زیر تغییر دهید:

```
private void Form1_Load(object sender, EventArgs e)
{
  arrWeight = new int[] { 13, 24, 33, 15, 20, 10, 7, 11 };
}
```

این دستورات عناصر آرایه را مقدار اولیه می‌دهند.

٦. رویداد Paint را برای فرم اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
  DrawSliceChart(e, arrWeight);
}
```



این دستورات متد DrawSliceChart() را فراخوانی می‌کنند.

٧. پروژه را ذخیره و اجرا کنید تا خروجی مقابل را ببینید:

مثال ۲–۸. برنامه‌ای که عبارت "فناوری نوین" را بر روی فرم حرکت می‌دهد (هدف این برنامه آشنایی بیشتر با کنترل Timer است).

مراحل طراحی و اجرا

١. پروژه جدیدی به نام Solve8_2 ایجاد کنید.

۲. یک کنترل Label و یک کنترل Timer به فرم اضافه کنید. خاصیت Text کنترل Label را "فناوری نوین" وارد نمایید.

۳. ناحیه خالی فرم را کلیک مضاعف کرده، دستورات رویداد Form1_Load را به صورت زیر تغییر دهید:

```
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Enabled = true;
}
```

این دستورات timer1 را فعال می‌کنند.

٤. کنترل timer1 را کلیک مضاعف کرده، دستورات رویداد Tick آن را به صورت زیر تغییر دهید:

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (label1.Right >= this.Right ) label1.Left = 10;
    label1.Left += 2;
}
```

این دستورات اگر label1 به انتهای فرم رسیده باشد، آن را به ابتدای فرم انتقال می‌دهند، وگرنه مکان سمت چپ label1 را دو واحد اضافه می‌کنند.

۵. پروژه را ذخیره و اجرا کنید تا حرکت عبارت "فناوری نوین" را بر روی فرم مشاهده کنید.

مثال ۳–۸. برنامه‌ای که تعداد کارکترها و کلمات یک کنترل RichTextBox را می‌شمارد (هدف این برنامه آشنایی بیشتر با کنترل RichTextBox است).

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh8_3 ایجاد کنید.

۲. دو کنترل RadioButton، یک کنترل RichTextBox به فرم اضافه کرده، خاصیت Text کنترل‌های RadioButton را به ترتیب تعداد کارکتر و تعداد کلمات تعیین کنید.

۳. یک Label به فرم اضافه کنید. خاصیت Text آن را خالی نمایید.

٤. متدهای CountWords()، CountCharacters() و UpdateDisplay() را به صورت زیر تعریف کنید:

```
private int CountCharacters(string text)
{
   return text.Length;
}
private int CountWords(string text)
{
```

```
  if (richTextBox1.Text == string.Empty)
    return 0;
  string[] strWords = text.Split(' ');
    return strWords.Length;
}
private void UpdateDisplay()
{
  if (radioButton2.Checked == true)
   label1.Text = " کلمه " + CountWords(richTextBox1.Text);
  else
   label1.Text= " کارکتر "+ CountCharacters(richTextBox1.Text);
}
```

متد ()CountCharacters، تعداد کارکترها را برمی‌گرداند، متد ()CountWords، تعداد کلمات را برگشت می‌دهد و متد ()UpdateDisplay، اطلاعات روی فرم را به روز می‌نماید.

۵. دکمه "تعداد کارکتر" را کلیک مضاعف کرده، دستورات رویداد CheckedChanged آن را به صورت زیر تغییر دهید:

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
   UpdateDisplay();
}
```

این دستورات با کلیک "تعداد کارکتر" نوشته‌های روی فرم را به روز می‌کنند.

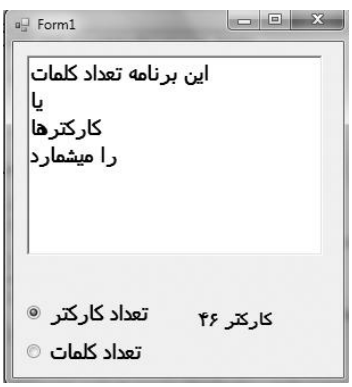
٦. دکمه"تعداد کلمات" را کلیک مضاعف کرده، دستورات رویداد CheckedChanged آن را به صورت زیر تغییر دهید:

```
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    UpdateDisplay();
}
```

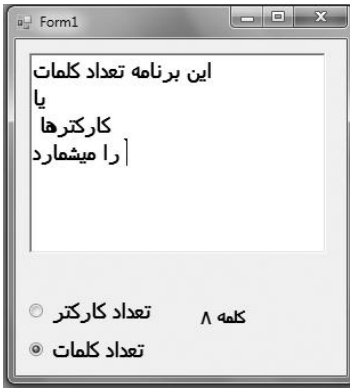
۷. کنترل richTextBox1 را کلیک مضاعف کرده، دستورات رویداد TextChanged آن را به صورت زیر تغییر دهید:

```
private void richTextBox1_TextChanged(object sender, EventArgs e)
{
    UpdateDisplay();
}
```

۸. پروژه را ذخیره و اجرا کرده، متنی را در داخل کادر تایپ نمائید و "تعداد کارکتر" را کلیک کنید تا خروجی مقابل را ببینید:



اکنون "تعداد کلمات" را کلیک کرده تا خروجی زیر را ببینید:



مثال ۴–۸. برنامه‌ای که اطلاعات کنترل‌هایی از قبیل ListBox و TextBox را از یک فرم به فرم دیگر انتقال می‌دهد(هدف این برنامه آشنایی با انتقال اطلاعات از یک فرم به فرم دیگر از طریق فیلدهای استاتیک می‌باشد).

مراحل طراحی و اجرا

۱. پروژه جدیدی به نام SolveCh8_4 ایجاد کنید.

۲. یک کنترل ListBox، یک کنترل TextBox و یک کنترل Button به فرم اضافه کنید. خاصیت Text کنترل Button را عبارت Form2 تعیین نمایید و اعداد ۱ تا 5 را به خاصیت Items کنترل ListBox اضافه کنید.

۳. با گزینه Project /Add Windows Form ، فرم جدیدی اضافه کرده، یک کنترل ListBox به آن اضافه کنید.

٤. با گزینه Project/Add Class کلاس جدیدی به نام Common به پروژه اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace SolveCh8_4    {
   static class Common  {
        public static ListBox listCommon=new ListBox();
        public static string txtCommon;
    }
}
```

این دستورات کلاس Common را به صورت static تعریف می‌کنند که دو فیلد از نوع static برای انتقال اطلاعات بین فرم دارد.

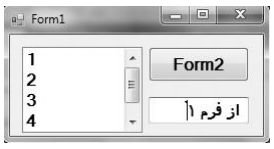
۵. به Form1 بروید و دکمه Form2 را کلیک مضاعف نمائید. اکنون دستورات زیر را در رویداد Click آن تایپ کنید:

```
private void button1_Click(object sender, EventArgs e)
{
  Common.listCommon.Items.Clear();
  Common.listCommon.Items.AddRange(listBox1.Items);
  Common.txtCommon = textBox1.Text;
  (new Form2()).Show();
}
```

را پاک می‌کند، دستور دوم، گزینه های Common از کلاس listCommonدستور اول، اطلاعات را در فیلد textBox1 اضافه می‌کند، دستور سوم، مقدار Common از کلاس listCommon را به listBox1 را نمایش می‌دهد. Form2 قرار می‌دهد و دستور چهارم، Common کلاس txtCommon

٦. به Form2 بروید و ناحیه خالی آن را کلیک مضاعف کرده، دستورات رویداد Form2_Load را به صورت زیر تغییر دهید:

```
private void Form2_Load(object sender, EventArgs e)
{
   listBox1.Items.Clear();
   listBox1.Items.AddRange(Common.listCommon.Items);
   Text = Common.txtCommon;
}
```



در فرم ۲ را حذف می‌کند، دستور listBox1دستور اول، گزینه‌های listBox1 را به Common از کلاس listCommonدوم، گزینه‌های Common از کلاس txtCommonاضافه می‌کند و دستور سوم، مقدار را در عنوان فرم ۲ قرار می‌دهد.

۷. پروژه را ذخیره و اجرا کنید. در textBox1 عبارت" از فرم 1 " را تایپ کرده تا شکل مقابل ظاهرشود:

دکمه Form2 را کلیک کرده تا خروجی زیر را ببینید:

همان طور که در این خروجی می‌بینید، اطلاعات ListBox

از فرم ۱ به Form2 انتقال یافته است و متن تایپ شده در کنترل

TextBox فرم ۱ در عنوان فرم ۲ آمده است.

۲۵–۸. مسائل حل شده در سایت

۱. برنامه‌ای که نمودار ستونی بر روی فرم رسم می‌کند این برنامه از بخش‌های زیر تشکیل شده است:

- **آرایه alWeight**، آرایه‌ای که مقادیر ستون‌های نمودار در آن قرار دارند.
- **متد DrawBarChart()**، رسم ستون‌های نمودار را با توجه به مقادیر آرایه alWeight انجام می‌دهد.
- **متد SumOfArray()**، مجموع مقادیر عناصر آرایه را برمی‌گرداند.
- **متد MaxValue()**،آرایه‌ای را به عنوان پارامتر دریافت کرده، بزرگترین مقدار عناصر آن را برمی‌گرداند.
- **متد Form1_Paint()**، متد DrawBarChart() را فراخوانی می‌کند.

۲. برنامه‌ایی که تصویری را بر روی PictureBox باز کرده، آن را چرخش می‌دهد. این برنامه از کلاس‌های زیر تشکیل شده است:

- **کلاس utilities**، این کلاس متدهایی از نوع static به نام RotateImage دارد که برای چرخش تصویر به کار می‌رود.
- کلاس Form دارای اعضای زیر است:
- **متغیر image**، تصویر را تعیین می‌کند و angle، میزان چرخش تصویر می‌باشد.
- **متد onKeyDown()**، رویداد KeyDown را مجدداً پیاده سازی می‌کند که با کلیدهای جهت نما بتوان چرخش تصویر را انجام داد.
- **متد ImageRotate()**، پارامترهای angle، pb(pictureBox) و img را دریافت کرده، img موجود در pb را به اندازه angle چرخش می‌دهد.
- **متد button1_Click()**، فایل تصویر را از طریق openFileDialog1 باز کرده، در pictureBox1 قرار می‌دهد.
- **متد numericUpDown1_ValueChanged()**، با توجه به مقدار numericUpDown1، تصویر را چرخش می‌دهد (با تغییر مقدار numericUpDown1 چرخش تصویر مجدداً انجام می‌شود)
- **متد Form1_Load()**، خواص اولیه پروژه را مقداردهی می‌کند.

۳. برنامه‌ایی که کرنومتر تایمر (زمان) را برروی فرم نمایش می‌دهد. ایـن برنامـه امکـان قطع نمـایش کرنـومتر را دارد. این برنامه متدهای زیر را دارد:

- **متد timer1_Tick()**، کرنومتر (زمان سپری شده) را برروی label1 نمایش می‌دهد.
- **متد Button1_Click()**، اگر timer1 فعال باشد، آن را غیر فعال خواهد کرد، وگرنه تاریخ و زمان فعلی را در da قرار می‌دهد، زمان را فعال می‌کند.
- **متد Form1_Load()**، خواص Text کنترل‌های روی فرم را تنظیم می‌کند.

۴. برنامه‌ای که زمان و تاریخ را برروی فرم نمایش می‌دهد. این برنامه متدهای زیر را دارد:

- **متد Form1_Load**، خاصیت InterVal کنترل timer1 را ۱۰۰۰ تعیین می‌کند و timer1 را فعال می‌نماید.
- **متد timer1_Tick()**، تاریخ و زمان فعلی را در عنوان فرم نمایش می‌دهد.

۵. برنامه‌ای که نمودار از نوع Pie رسم می‌کند. این برنامه متدهای زیر را دارد:

- **متد MaxValue()**، آرایه‌ای به نام intArray را به عنوان پارامتر دریافت کرده، بزرگتـرین مقـدار آن را برمـی-گرداند.
- **متد DrawPieChart()**، پارامتر e و آرایه alWeight (وزن Pieهایی که باید رسم شـوند) را دریافـت کـرده، نمودار از نوع Pie را رسم نماید.
- **متد Form1_Paint()**، بافراخوانی متد DrawPieChart نمودار را رسم می‌کند.
- **متد Form1_Load()**، به عناصر آرایه alWeight مقدار می‌دهد.

۶. برنامه‌ای که امکان کشیدن و رها کردن مقداری را از یـک TextBox بـه TextBox دیگـر را مـی‌دهـد. ایـن برنامه از متدهای زیر تشکیل شده است:

- **متد textbox_DragDrop()**، متغیر txt را تعریف کرده، آدرس Sender را در آن قرار مـی‌دهـد و اطلاعـات Text پارامتر e را در آن قرار می‌دهد.
- **متد textbox_DragEnter()**، اگر خاصیت Data پارامتر e دارای مقدار باشـد، مقـدار DragDropEffects را در Effect پارامتر e کپی می‌کند، وگرنه Effect پارامتر e را None در نظر می‌گیرد.
- **متد textbox_MouseDown()**، آدرس Sender را در txt قرار می‌دهد. اطلاعات txt را انتخـاب مـی‌کنـد و متد DoDragDrop را با پارامترهای txt.Text و DragDropEffexts.Copy فراخوانی می‌کند.

- **متد Form1_Load()**، رویدادهای dragDrop، MouseDown و DragEnter را برای کنترل‌های textBox1 و textBox2 اضافه کرده، خاصیت AllowDrop کنترل‌های textBox1 و textBox2 را True تعیین می‌کند تا در کنترل‌های textBox1 و textBox2 امکان کشیدن و رها کردن وجود داشته باشد.

۷. برنامه‌ای که جهت حرکت کلیدهای ← و → را تغییر می‌دهد (این برنامه از متد textBox1_KeyDown تشکیل شده است که جهت حرکت کلیدها را تغییر می‌دهد).

۸. برنامه‌ای که تصویری را در یک textBox قرار می‌دهد. این برنامه دارای متد button1_Click است که یک کنترل PictureBox در زمان اجرا ایجاد می‌کند، کادر openFileDialog() را نمایش می‌دهد تا کاربر تصویری را انتخاب کند، سپس تصویر انتخاب شده در خاصیت Image کنترل Pic (از نوع PicturBox) قرار می‌گیرد. در پایان، کنترل‌های موجود در textBox1 را پاک کرده، کنترل Pic را به آن اضافه می‌نماید.

۹. برنامه‌ای که خانه‌های مشبک برروی فرم رسم می‌کند. این برنامه دارای متد Form1_Paint() است که خطوط عمودی و افقی با قلمی به رنگ قرمز با عرض ۲ رسم می‌کند که از برخورد این خطوط، خانه‌های مشبک تولید می‌گردد.

۱۰. برنامه‌ای که منحنی برروی فرم رسم می‌کند (هدف این برنامه آشنایی با متد DrawCurve() می‌باشد). این برنامه دارای متد Form1_Paint() است که در آن قلمی با رنگ آبی به ضخامت ۳ ایجاد می‌شود، شیء گرافیک ایجاد شده مختصات نقاط مورد نیاز برای رسم منحنی مقدار می‌گیرند، با فراخوانی متد DrawCurve() منحنی رسم می‌شود و شیء obj حذف می‌گردد.

۱۱. برنامه‌ای که امکان انتخاب فونت‌های پررنگ (Bold)، کج (Italic)، زیرخط دار (UnderLine)، اندازه فونت و قرار گرفتن داده در وسط را برای کنترل RichTextBox فراهم می‌کند. این برنامه از متدهای زیر تشکیل شده است:

- **متد button1_Click()**، فونت پررنگ را تعریف کرده، به متن انتخاب شده در RichTextBox تخصیص می‌دهد.
- **متد button2_Click()**، فونت ایتالیک را تعریف کرده، به متن انتخاب شده در RichTextBox تخصیص می‌دهد.
- **متد button3_Click()**، فونت زیر خط دار را تعریف کرده، به متن انتخاب شده در RichTextBox تخصیص می‌دهد.

متد button4_Click()، متن انتخاب شده، اگر در وسط RichTextBox1 باشد، آن را به سمت چپ انتقال می‌دهد وگرنه، آن را به وسط RichTextBox منتقل خواهد کرد.

متد numericUpDoown_Validating()، اندازه فونت را انتخاب کرده و فونت موجود را به این اندازه تغییر خواهد داد. سپس، اطلاعات انتخاب شده را با فونت جدید تغییر می‌دهد.

۲۶-۸. تمرین

۱. برنامه‌ای بنویسید که اعمال زیر را انجام دهد:

- کاربر بتواند کامپیوتر را Shutdown، log off یا Restart نماید.
- کاربر بتواند لیست پردازش‌ها (Process) را ببیند و یکی از آن‌ها را حذف کند.
- کاربر بتواند فایل اجرایی خاصی را انتخاب کرده، اجرا نماید.

۲. برنامه‌ای بنویسید که امکان تغییر تصویر دسک‌تاپ کامپیوتر را فراهم کند.

۳. برنامه‌ای بنویسید که ارتباط با پورت سری کامپیوتر را برقرار کند و اطلاعاتی را برای پورت‌ها بفرستد.

۴. برنامه‌ای بنویسید که اعمال زیر را انجام دهد:

- تصویری را باز کند. تصویر را لایه گذاری کرده یا شفافیت رنگ آن را تغییر دهد. تصویر را ذخیره نماید و تصویر اولیه را برگرداند.
- از صفحه کامپیوتر عکس بگیرد.
- رنگ‌ها را ترکیب کند و رنگ‌های ترکیب شده را نمایش دهد.
- برنامه‌ای بنویسید که درایوهای موجود در سیستم را در یک comboBox نمایش دهد. کاربر بتواند درایو مورد نظر را انتخاب کرده، اطلاعات آن را ببیند.

۵. برنامه‌ای بنویسید که آیکن خاصی را به طور تصادفی بر روی نقاط مختلف صفحه نمایش جابه‌جا کند.

۶. برنامه بنویسید که فرم خوش‌آمدگویی طراحی کند و آن را برای چند لحظه نمایش دهد.

۷. برنامه‌ای بنویسید که اعمال زیر را انجام دهد:

- کنترلی را از یک فرم بکشد و بر روی فرم دیگر رها کند. در فرمی که کنترل بر روی آن رها می‌گردد، مشخصات کنترل رها شده (Draged) را نمایش دهد.
- مشخصات کنترلی که مکان‌نمای ماوس بر روی آن رفته است را در نوار وضعیت (statustBar) فرم اصلی نمایش دهد.

۸. برنامه‌ای بنویسید که اعمال زیر را انجام دهد:

- دکمه‌های با حالت مختلف رسم کند.
- چک باکس‌های (checkBox) مختلف ترسیم نماید.
- فرمی به شکل دایره رسم کند.
- رنگ زمینه کلیه فرم‌های باز را تغییر دهد.

۹. برنامه‌ای بنویسید که چگونگی ایجاد و ذخیره کلیدهای میانبر (ShortCut) را در C# نشان دهد.

۱۰. برنامه‌ای بنویسید که اعمال زیر را انجام دهد:

- ایستگاه کاری (کامپیوتر فعلی) را قفل کند.
- سطل بازیافت را خالی نماید.
- زمان اجرای بخشی از برنامه را محاسبه کند.
- برنامه را مجددا اجرا کند.
- بر روی دسک‌تاپ ویندوز متنی را بنویسد.
- شفافیت کنترلی را تغییر دهد.

۱۱. برنامه‌ای بنویسید که متنی را با جلوه ویژه نمایش دهد.

۱۲. برنامه‌ای بنویسید که اعمال زیر را بر روی تصویر خاصی انجام دهد:

- تصویری را بار کند.
- تصویر را چرخش دهد.
- وسط یک کنترل بنویسد.
- اندازه تصویر را تغییر دهد.
- تصویر را کپی نماید.

امروزه سازمان‌ها، مؤسسات، ادارات و شرکت‌ها با حجم عظیمی از داده‌ها سر و کار دارند. به عنوان مثال، فرض کنید بخواهید اطلاعات مربوط به مکالمات شرکت مخابرات یکی از استان‌ها را نگهداری کنید. به طوری که در یک سال حدود ۱۵ میلیارد رکورد جمع‌آوری می‌گردد. نگهداری، پردازش و بازیابی این حجم اطلاعات از طریق فایل‌های معمولی زمان‌بر است. برای جلوگیری از تکرار بی‌مورد داده‌ها (افزونگی داده‌ها)، ایجاد سازگاری بین گزارش‌ها (جلوگیری از بی نظمی) و صرفه‌جویی در میزان حافظه، به کارگیری بانک اطلاعات به صورت یک ضرورت درآمده است. یعنی، بدون استفاده از بانک اطلاعات نمی‌توان اطلاعات مربوط به مکالمات تلفن ثابت یک استان را نگهداری و ذخیره کرد. از طرف دیگر، اکثر برنامه‌های کاربردی که با داده‌ها سر و کار دارند، داده‌ها را در بانک اطلاعات ذخیره می‌نمایند و از طریق بانک اطلاعات آن را پردازش می‌کنند.

بانک‌های اطلاعات متعددی وجود دارند که از جمله می‌توان Access، SQL Server، Oracle، DB2 و MySQL را نام برد. هر یک از این بانک‌های اطلاعات کاربرد خاصی دارند. در بین این بانک‌ها، بانک اطلاعات SQL Server از محبوبیت خاصی برخوردار است. زیرا، حدود ۷۰ درصد از کاربران دنیا از این بانک اطلاعات استفاده می‌کنند. به همین خاطر، در این فصل ابتدا با مفاهیم بانک اطلاعات و دلایل استفاده از آن آشنا خواهیم شد. سپس، یک بانک اطلاعات نمونه را مثال می‌زنیم و در ادامه فصل، این بانک اطلاعات را ایجاد کرده، اعمال مختلف را بر روی آن انجام می‌دهیم.

۱–۹. تعریف سیستم مدیریت بانک اطلاعات

سیستم مدیریت بانک اطلاعات، مکانیزیم نگهداری رکوردها[15] است. یعنی، بانک اطلاعات مخزنی برای نگهداری از داده‌ها است که کاربران آن می‌توانند اعمال زیر را انجام دهند:

۱. افزودن جداول خالی به بانک اطلاعات

۲. افزودن رکوردهایی به جداول بانک اطلاعات

۳. تغییر ساختار جداول

---

[15]- **Records** [2]- **Query**

٤. حذف رکوردهای بانک اطلاعات

۵. تغییر داده‌های موجود دربانک اطلاعات

٦. اجرای پرس‌وجو[۱۶] بر روی جداول

به عبارت ساده‌تر، سیستم مدیریت بانک اطلاعات، سیستمی کامپیوتری است که هدف آن ذخیره و بازیابی داده‌ها می‌باشد. بانک اطلاعات داده را پردازش نموده به اطلاعات تبدیل کرده، آن را بازیابی می‌نماید.

۲-۹. طراحی بانک اطلاعاتی

بانک‌های اطلاعاتی مختلفی وجود دارند که مهم‌ترین آنها بانک اطلاعات رابطه‌ای است. اطلاعات در بانک اطلاعات رابطه‌ای، بین جداول مختلف توزیع می‌گردند تا ذخیره‌سازی و بازیابی اطلاعات بهینه‌تر شود. یعنی، از افزونگی داده‌ها جلوگیری می‌گردد، بی‌نظمی را کاهش می‌دهد و داده‌های تهی را نیز کاهش خواهد داد. این زمانی اتفاق می‌افتد که بانک اطلاعات خوب طراحی گردد. بنابراین، هرچه بانک اطلاعات بهتر طراحی شود، ابزار مهمی برای مدیریت اطلاعات شخصی، تجاری یا اداری است. ولی، چنانچه بانک اطلاعات بد طراحی گردد، ارزش چندانی نخواهد داشت. پس، هرچه وقت بیشتری برای طراحی و تحلیل داده‌ها صورت گیرد، نتیجه بهتری بدست می‌آید. زمانی که طراحی کامل بررسی گردید، به راحتی می‌توان بانک اطلاعات را ایجاد نمود.

فرآیند طراحی با تحلیل کارهایی شروع می‌گردد که برای بانک اطلاعات نیاز داریم. در این فرآیند، ابتدا باید مشخص کنید سیستم چه کاری باید انجام دهد. با کاربران مصاحبه کرده تا به خواسته‌های آن‌ها پی ببرید. توجه داشته باشید که فرآیند طراحی، فرآیندی تکراری است. وقتی کاربران می‌خواهند از سیستم جدید استفاده کنند، راجع به ویژگی‌های آن فکر می‌کنند، مثل فرم‌های ورود داده‌ها، پرس‌وجوهای خاص، و فیلدهای محاسباتی.

از طرف دیگر، طراحی باید در یک نقطه خاتمه یابد و توسعه بانک اطلاعات شروع گردد. در این صورت، خواسته‌های جدید سیستم را می‌توانید در نسخه‌های بعدی سیستم منظور کنید. فرآیند طراحی بانک اطلاعات را می‌توان به مراحل زیر تقسیم کرد که هر مرحله دارای هدف خاصی است:

۱. **تعیین خواسته‌های کاربران:** در این مرحله، با کاربران مصاحبه می‌گردد. فرم کاربران بررسی می‌شود تا خواسته‌های آنها از بانک اطلاعات مشخص گردد.

۲. **توزیع داده‌ها در جداول:** یکی از مهم‌ترین بخش‌های طراحی، توزیع داده‌های جداول است. زیرا، چنانچه طراحی جداول خوب باشد، از تکرار بی‌مورد، بی‌نظمی و ورود داده‌های تهی در جداول جلوگیری می‌کند. این کار با نرمال‌سازی جداول اتفاق می‌افتد که در ادامه نرمال‌سازی را می‌بینید.

۳. فیلدهای هر رکورد را در هر جدول مشخص نمایید.

٤. برای هر جدول کلید اولیه و کلیدهای کاندید را تعیین کنید تا تضمین شود که هیچ دو رکورد یکسان نباشند.

۵. ارتباط بین جداول را تعیین کنید تا بتوانید پرس‌وجوها را از چند جدول تهیه کنید.

٦. طراحی را با کاربران مرور کنید تا مطمئن شوید، بانک اطلاعات طراحی شده نیازهای کاربران را برطرف می‌کند.

۷. جداول بانک اطلاعات را ایجاد کرده، داده‌ها را در آنها وارد نمایید.

۸. کارایی بانک اطلاعات را تحلیل کنید و بانک اطلاعات طراحی شده را بهینه نمایید تا بتوانید سریع‌تر نتایج پرس‌وجوها را دریافت کنید.

۳-۹. معرفی بانک اطلاعاتی نمونه

در این بخش یک بانک اطلاعاتی نمونه را معرفی می‌نماییم. سپس، مراحل طراحی و پیاده‌سازی این بانک اطلاعاتی را با هم مرور می‌کنیم. بانک اطلاعاتی که برای این کتاب در نظر گرفته شده است، اطلاعات مربوط به شهرها و استان‌ها را نگهداری می‌نماید. برای طراحی این بانک اطلاعاتی، مراحل زیر باید انجام شود:

۱. **تعیین اهداف بانک اطلاعاتی:** در این مرحله باید فرم‌ها، نمودارها، گزارشات و موارد دیگر مورد نیاز کاربران تعیین شود. در این مرحله، همچنین باید فرم‌ها، گزارشات و غیره طراحی، پیاده‌سازی با کاربران مرور شوند.

۲. **توزیع داده‌ها:** مهم‌ترین مرحله طراحی بانک اطلاعاتی، تعیین چگونگی توزیع داده است. در این مرحله باید تعیین گردد، چه داده‌هایی در چه جداولی توزیع شود. در این بخش باید تعداد جداول بانک اطلاعاتی، فیلدهای هر یک از جداول و نوع فیلدها تعیین شود. در این بانک اطلاعاتی نمونه چند جدول در نظر گرفته شده است که برخی از آنها عبارت‌اند از:

- **جدول Group1:** اطلاعات مربوط به گروه‌ها از قبیل کد گروه و نام گروه را نگهداری می‌کند.
- **جدول Student:** اطلاعات مربوط به دانشجویان را نگهداری می‌کند (جدول۱-۹ را ببینید).
- **جدول Clg:** اطلاعات دانشکده‌ها را نگهداری می‌کند(جدول۱-۹ را مشاهده کنید).

۳. تعیین ساختار و فیلدهای جدول: بعد از این که فیلدهای جدول مشخص گردید، باید تعیین شود نوع هر فیلد چیست، آیا فیلد می‌تواند Null باشد یا خیر، چه فضایی از حافظه را اشغال می‌کند. در این بخش باید تعیین کنیم هر جدول از چند فیلد تشکیل شده است و هر فیلد دارای چه خواصی (نام، اندازه، نوع و محدودیت‌ها) است. فیلدهای بانک اطلاعاتی نمونه در جدول۱-۹ آمده‌اند.

۴. تعیین فیلد کلید اولیه[۱۷] جداول: یکی از بخش‌های مهم طراحی بانک اطلاعاتی تعیین فیلد کلید اولیه هر جدول است. فیلد کلید اولیه سه خاصیت دارد که عبارت‌اند از:

۱. فیلد کلید اولیه نمی‌تواند تهی (Null) باشد. یعنی، اگر فیلدی را به عنوان کلید اولیه انتخاب کنید، به طور خودکار محدودیت NOT NULL به آن تخصیص می‌یابد.

۲. فیلد کلید اولیه (یا کلید) تضمین می‌کند که هیچ دو رکوردی در جدول برای آن فیلد دارای مقدار یکسانی نمی‌باشند. یعنی، فیلد کلید از ورود رکوردهای تکراری در جدول جلوگیری می‌کند.

۳. اطلاعات جدول براساس فیلد کلید اولیه مرتب می‌شوند. به عنوان مثال، در بانک اطلاعاتی نمونه، در جدول Student، فیلد stNo، فیلد کلید اولیه است. زیرا، برای هیچ دو دانشجو مقدار این فیلد برابر نمی‌باشد، یا به عبارت دیگر، هیچ دو دانشجو شماره دانشجویی یکسانی ندارند. ولی، در جدول Group1، فیلد groupCode فیلد کلید اولیه است.

۵. تعیین ارتباط بین جدول‌ها: یکی از ویژگی‌های اولیه بانک اطلاعات رابطه‌ای، ارتباطی است که جداول می‌توانند با یکدیگر داشته باشند. ارتباط بین جداول موجب می‌شود تا از افزونگی (تکرار بی‌مورد داده‌ها در جدول) جلوگیری شود. ارتباط بین جدول Gropu1 و Student با استفاده از فیلد GroupCode برقرار می‌شود.

۶. تعیین اشیای دیگر بانک اطلاعات: بانک اطلاعاتی از اشیای متعددی تشکیل شده است. مهم‌ترین شیء بانک اطلاعاتی جدول است. جدول برای نگهداری داده‌های بانک اطلاعاتی به کار می‌رود. اشیای دیگری از قبیل دیدها[۱۸]، توابع[۲]، رویه‌های ذخیره شده[۳]، کرسرها[۴]، تریگرها[۵] و غیره در بانک اطلاعاتی وجود دارند.

در ادامه کتاب با شیء Table آشنا خواهید شد و چگونگی ایجاد، حذف و ویرایش آن را در بانک می‌آموزیم.

---

[17]- Primary Key

[18] .Views [2]- Functions [3]- Stored Procedures [4]- Cursors [5]- Triggers [6].Database Management System

۹–۹. دستیابی به بانک اطلاعات با ADO.NET

بیان گردیده که بانک اطلاعاتی همان فایل‌های کامپیوتری (فایل‌های داده و کارنامه که در فصل ۲ دیدید) هستند که برنامه کاربردی با استفاده از سیستم مدیریت بانک اطلاعات (DBMS[۱۹]) آن را پردازش می‌کند. امّا، برای این که برنامه کاربردی با سیستم مدیریت بانک اطلاعات ارتباط برقرار کند، نیاز به واسط[۲۰] نرم افزاری دارد. یکی از واسط‌های نرم افزاری بسیار مهم ADO.NET[۲] می‌باشد. این واسط امکان ارتباط با بانک اطلاعاتی رابطه‌ای و سایر منابع داده را فراهم می‌آورد. برخی از ویژگی‌های ADO.NET از قبیل بی اتصال بودن[۳]، قابلیت برنامه‌نویسی، کارایی، توسعه‌پذیری و غیره، این تکنولوژی را از سایر تکنولوژی‌های دیگر متمایز ساخته است. در این تکنولوژی کلاس‌های متعددی برای کار با بانک‌های اطلاعات مختلف فراهم شده‌اند که برخی از آن‌ها را در ادامه می‌بینید. از آنجایی که در این کتاب بانک اطلاعاتی SQL Server بررسی می‌گردد، به کلاس‌های مربوط به کار با بانک اطلاعاتی SQL Server می‌پردازیم.

۱–۹–۹. کلاس Connection

اولین قدم در عملیات بانک اطلاعاتی، برقراری ارتباط بین برنامه کاربردی و سرویس‌دهنده بانک اطلاعات است. برای این منظور می‌توانید از کلاس Connection استفاده کنید. کلاس‌های مختلفی برای برقراری اتصال با بانک اطلاعاتی به کار می‌روند که برخی از آن‌ها عبارت‌اند از:

۱- SqlConnection ۲- oledbConnection

۳- oracleConnection ٤- OdbcConnection

برای استفاده از شیء SqlConnection باید نمونه‌ای از آن ایجاد کرده، پارامترهای رشته اتصال آن را مقداردهی کنید و آن را باز نمایید. پارامترهای رشته اتصال همان پارامترهایی مانند UserID (حساب کاربری)، Password (کلمه عبور)، Data Source (منبع داده)، Initial Catalog (نام بانک اطلاعاتی) و غیره می‌باشند. این پارامترها را در کلاس DataContext دیدید. برای ایجاد نمونه‌ای از شیء SqlConnection به صورت زیر عمل کنید:

```
String conStr = "Data Source = TCI;Intial Catalog =
                Student; Integrated Security=SSPI;"
SqlConnection con = new SqlConnection(conStr);
```

---

1 .Database Management System

[1].Interface [2].Activex Data Objects [3].ConnectionLess

این دستورات، شیء اتصالی به نام con از نوع SqlConnection تعریف می‌کند. این شیء می‌تواند به بانک اطلاعاتی Student در سرویس دهنده TCI وصل شود. برای برقراری اتصال (بازکردن اتصال) باید از متد ()Open به صورت زیر استفاده کنید:

```
con.Open();
```

این دستور، اتصال به بانک اطلاعاتی را از طریق شیء اتصال con برقرار می‌کند.

بعد از این که کارتان با بانک اطلاعاتی خاتمه یافت، باید اتصال به آن را قطع کنید. برای این منظور می‌توانید از متد ()Close به صورت زیر استفاده نمایید:

```
con.Close();
```

۲-۹-۹. کلاس Command

این کلاس برای اجرای دستورات SQL یا رویه‌های ذخیره شده از طریق #C بر روی بانک اطلاعات به کار می‌رود. کلاس‌های متعددی برای Command وجود دارند که برخی از آن‌ها عبارت‌اند از :

جدول ۳ - ۹ خواص و متدهای مهم شیء SqlCommand.

خاصیت	هدف
CommandText	دستور SQL یا نام رویه ذخیره شده‌ای را تعیین می‌کند که می‌خواهید بر روی بانک اطلاعات اجرا گردد.
CommandType	نوع دستوری را تعیین می‌کند که در CommandText قرار گرفته است.
Connection	شیء اتصالی (Connectionای) را تعیین می‌کند که شیء SqlCommand از طریق آن باید دستورات را اجرا کند.
Parameters	کلکسیون پارامترهای ارسال شده به شیء SqlCommand را تعیین می‌کند.
متد	**هدف**
Cancel	فرمان در حال اجرا را لغو می‌کند.
ExecuteNonQuery()	شیء SqlCommand را بر روی بانک اجرا می‌کند و تعداد رکوردهایی را برمی‌گرداند که تحت تاثیر اجرای این دستور قرار گرفته‌اند.
ExecuteScalar()	دستور موجود در خاصیت CommandText شیء SqlCommand را اجرا کرده اولین فیلد را برمی‌گرداند.
ExecuteReader()	دستور موجود در شیء SqlCommand را اجرا کرده، مجموعه‌ای از رکوردها را برمی‌گرداند که قابل ویرایش نیستند (فقط خواندنی هستند).

۱. کلاس SqlCommand ۲. کلاس OledbCommand

۳. کلاس odbcCommand ٤. کلاس OracleCommand

برای کار با کلاس SqlCommand، نمونه‌ای از آن ایجاد کرده، خواص آن را مقداردهی نمایید و از متدهای آن استفاده کنید تا دستورات موجود در نمونه بر روی بانک اطلاعات اجرا شوند. برخی از خواص و متدهای مهم کلاس SqlCommand در جدول ۳ – ۹آمده است. برای استفاده از شیء SqlCommand مراحل زیر را انجام دهید:

۱. با دستور زیر یک شیء از نوع SqlCommand ایجاد کنید:

```
SqlCommand  cmd = new  SqlCommand();
```

۲. خاصیت Connection آن را به صورت زیر مقداردهی کنید:

```
cmd.Connection = con ;
```

con، شیءای از نوع SqlConnection می‌باشد که قبلاً ایجاد گردید.

۳. خاصیت CommandText آن را به صورت زیر مقداردهی نمایید:

```
cmd.CommandText = "SELECT  *  FROM Group1" ;
```

این دستور، برای بازیابی اطلاعات جدول Group1 به کار می‌رود. خاصیت CommandText می‌تواند نام رویه ذخیره شده یا نام جدول باشد.

۴. خاصیت CommandType آن را مقداردهی کنید. این خاصیت می‌تواند یکی از مقادیر CommandType.Text (اگر مقدار CommandText یک دستور Sql باشد)، CommandType.StoredProcedure (اگر مقدار خاصیت CommandText نام یک رویه ذخیره شده باشد) و CommandType.TableDirect (اگر مقدار خاصیت CommandText، نام یک جدول باشد) را بپذیرد. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
cmd.CommandType = CommandType.Text ;
```

این دستور، نوع دستور cmd را از نوع Text (دستور SQL) در نظر می گیرد.

۵. دستور موجود در شیء Cmd را اجرا نمایید. برای این منظور می‌توانید از متدهای ExecuteNonQuery()، ExecuteScalar()، ExecuteReader() و دیگر متدها استفاده کنید. عملکرد این متدها را در جدول ۳ – ۹ دیدید. به عنوان مثال، دستور زیر را ببینید:

```
cmd.ExecuteNonQuery() ;
```

این دستور، دستور موجود در cmd را اجرا کرده اطلاعات جدول City را بازیابی می‌نماید.

۳-۹-۹. کلاس Dataset

این کلاس هسته ADO.NET است. یک Dataset، از مجموعه‌ای از جداول و اطلاعاتی درباره آن‌ها تشکیل شده است و نتایج پرس‌وجو از بانک اطلاعات را نگهداری می‌نماید. یکی از ویژگی‌های جالب Dataset این است که بدون این که به منبع داده متصل باشیم، می‌توانیم از آن‌ها استفاده کنیم. کلاس Dataset از کلاس‌های زیر تشکیل می‌گردد:

۱. **کلاس** DataTable، اطلاعات جدول بانک اطلاعاتی را نگهداری می‌کند.

۲. **کلاس** DataRelation، برای ایجاد ارتباط بین جداول بانک اطلاعات به کار می‌رود.

۳. **کلاس** DataView، اطلاعات موجود در DataTable را به شیوه‌های مختلف نمایش می‌دهد، به طوری که می‌توان اطلاعات جدول را فیلتر یا مرتب نمود.

۴. **کلاس** DataAdapter، پلی بین Dataset و منبع داده (بانک اطلاعات) برقرار می‌کند. برای بازیابی اطلاعات از بانک اطلاعات و قرار دادن در Dataset و ثبت تغییرات انجام شده در Dataset از کلاس DataAdapter استفاده می‌گردد.

برای پرکردن Dataset مراحل زیر را انجام دهید:

۱. یک شیء اتصال به بانک اطلاعات تعریف کنید. برای این منظور می‌توانید از کلاس SqlConnection به صورت زیر استفاده نمایید:

```
SqlConnection con = new SqlConnection("رشته اتصال");
```

۲. دستور یا دستورات Sqlایی را تعریف کنید که می‌خواهید داده‌ها را از جداول بانک اطلاعات بازیابی کند (مانند دستور زیر):

```
String SQL = "SELECT  *  FROM Student";
```

۳. نمونه‌ای از کلاس SqlDataAdapter تعریف کنید. به عنوان مثال، دستورات زیر را ببینید:

```
SqlDataAdapter da = new  SqlDataAdapter(SQL,Con) ;
```

۴. شیء Dataset را ایجاد کنید (دستور زیر):

```
Dataset  ds = new Dataset();
```

۵. متد Fill را روی نمونه شیء SqlDataAdapter اجرا نمایید تا Dataset را پر کنید. این دستور به صورت زیر به کار می‌رود:

```
da.Fill(ds,"Tale1") ;
```

پس از این که اطلاعات از بانک اطلاعات بازیابی گردید و در Dataset قرار گرفت، اطلاعات Dataset را می‌توان بدون اتصال به بانک اطلاعاتی ویرایش نمود. در پایان باید بتوانید تغییرات اعمال شده در Dataset را بر روی بانک اطلاعات اعمال کنید. برای این منظور می‌توانید از متد ()Update استفاده نمایید. همان‌طور که بیان شده، Dataset می‌تواند شامل تعدادی جدول باشد، برای دسترسی به جدول بانک اطلاعات می‌توانید از خاصیت Table به همراه اندیس آن یا نام جدول استفاده کنید. به عنوان مثال، دستورات زیر را ببینید:

```
ds.Table[0]
ds.Table["Table1"]
ds.Table[3]
```

دستور اول، اولین جدول ds و دستور دوم، جدول Table1 از ds را بازیابی خواهد کرد و دستور سوم، چهارمین جدول ds را بازیابی می‌کند.

مثال ۱ – ۹. پروژه‌ای که جدول بانک اطلاعات نمونه (جداول موجود در جدول ۱ – ۹) را ایجاد می‌کند، اعمالی از قبیل افزودن، حذف، ویرایش رکوردها را برروی آن‌ها انجام می‌دهد. در این برنامه امکان جستجو براساس فیلدهای خاص وجود دارد.

مراحل طراحی واجرا

۱. پروژه جدیدی به نام Ch9 _ 1 ایجاد کنید.

۲. کلاس جدیدی به نام Database به پروژه اضافه کرده، دستورات آن را به صورت زیر تغییر دهید:

```
using System;
using System.Data.SqlClient; // 1- added
using System.Windows.Forms;  // 2- added
using System.Data;           // 3-added
namespace Ch9_1   {
 static class Database     {
  public static string conStr="Data Source=1- VAIO\\YOUSEF;Initial
        Catalog=Student;Integrated Security=True";
   public static bool executeQuery(string SQL) {
    try  {
          SqlConnection conn = new SqlConnection(conStr);
          conn.Open();
          SqlCommand cmd = new SqlCommand(SQL, conn);
          cmd.ExecuteNonQuery();
          return true;
        }
    catch { return false; }
   }
   public static void fillDataGrid(DataGridView dataGrid1,
          string SQL) {
          SqlConnection con = new SqlConnection(conStr);
          con.Open();
```

```
        SqlDataAdapter da = new SqlDataAdapter(SQL, con);
        DataSet ds = new DataSet();
        da.Fill(ds);
        dataGrid1.DataSource = ds.Tables[0];
}
public static void showMessage(bool correct, string
        message1, string message2){
        if (correct == true)
          MessageBox.Show(message1);
        else
          MessageBox.Show(message2);
}
public static void comboBoxFill(string tableName, string
        fieldName, ComboBox comboBox1){
        SqlConnection conn = new SqlConnection(conStr);
        DataSet ds = new DataSet();
        conn.Open();
        string SQL = "SELECT " + fieldName.Trim() + " FROM "
                     + tableName + " ORDER BY " + fieldName;
        SqlDataAdapter da = new SqlDataAdapter(SQL, conn);
        ds.Clear();
        da.Fill(ds, tableName);
        comboBox1.DataSource = ds.Tables[0].DefaultView;
        comboBox1.DisplayMember = fieldName;
        conn.Close();
}
public static object comboBoxReturn(string fieldName1,
        string tableName, string fieldName2, string value){
        SqlConnection conn = new SqlConnection(conStr);
        conn.Open();
        string SQL = "SELECT  " + fieldName1 + " FROM " +
          tableName +" Where " + fieldName2 +" = '" + value + "'";
        SqlCommand cmd = new SqlCommand(SQL, conn);
        cmd.CommandType = CommandType.Text;
        object _codeReturn = cmd.ExecuteScalar();
        conn.Close();
        return _codeReturn;
}
public static void nextControl(KeyEventArgs e){
        switch (e.KeyCode){
             case Keys.Down:
             case Keys.Enter:
                  SendKeys.Send("{TAB}");
                  break;
             case Keys.Up:
                  SendKeys.Send("+{TAB}");
                  break;
 }
```

```
    }
  }
}
```

این کلاس دارای اعضای زیر است:

- **فیلد conStr**، عضوی است که رشته اتصال را نگهداری می‌کند.
- **متد executeQuery()**، دستور SQLای که باید اجرا شود را به عنوان پارامتر گرفته (پارامتر SQL)، آن را برروی بانک اطلاعاتی اجرا می‌کند. اگر دستور به طور صحیح اجرا شود، ture، وگرنه false را برمی‌گرداند.
- **متد fillDataGrid()**، نتیجه یک دستور SQL را در یک DataGridView نمایش می‌دهد. این متد DataGridView و دستور SQL را از طریق پارامترهای dataGrid1 و SQL دریافت می‌کند.
- **متد comboBoxFill()**، ComboBox خاصی (پارامتر comboBox1) را با یک فیلد (پارامتر fieldName) از جدول مشخص (پارمتر tableName) پرمی‌کند.
- **متد comboBoxReturn()**، رشته‌ای (پارامتر value) را به عنوان پارامتر دریافت می‌کند. مقدار آن را در در فیلد اول (پارامتر fieldName2) یک جدول (پارامتر tableName) جستجو کرده، مقدار فیلد دوم (پارامتر fieldName1) را برمی‌گرداند.
- **متد nextControl()**، اطلاعات کلید فشرده شده را از طریق پارامتر e دریافت کرده، اگر یکی از کلیدهای Enter و ↓ فشره شد باشد، مکان به کنترل بعدی انتقال می‌یابد. ولی چنانچه کلید ↑ فشارداده شود، مکان‌نما به کنترل قبلی منتقل خواهد شد.

۳. یک کنترل MenuStrip به فرم اضافه کرده، دو منو به نام‌های **ایجاد جدول** و **دستکاری جداول** به منو اضافه کنید. گزینه‌های **ایجاد جدول گروه، ایجاد جدول دانشجو، ایجاد جدول دانشکده و خروج** را به منوی **ایجاد جدول** اضافه نمایید و گزینه‌های **جدول گروه، جدول دانشجو و جدول دانشکده** را به منوی دستکاری داده اضافه کنید.

۴.کنترل textBox2 را کلیک مضاعف کرده، دستورات رویداد TextChanged() آن را به صورت زیر تغییر دهید:

```
private void textBox2_TextChanged(object sender, EventArgs e){
 string SQL = "SELECT clgNo 'شماره' , clgName نام, bossName
         'نام رئیس'  FROM Clg";
 SQL += " WHERE clgName LIKE '%" + textBox2.Text + "%'";
 Database.fillDataGrid(dataGridView1, SQL);
}
```

این دستورات با توجه به تغییر نام دانشکده اطلاعات dataGridView1 را فیلتر می‌کنند.

۵.کنترل textBox3 را کلیک مضاعف کرده، دستورات رویداد ()TextChanged آن را به صورت زیر تغییر دهید:

```
private void textBox3_TextChanged(object sender, EventArgs e){
 string SQL = "SELECT clgNo 'شماره' , clgName نام, bossName
        'نام رئیس'  FROM Clg";
 SQL += " WHERE clgName LIKE '%" + textBox3.Text + "%'";
 Database.fillDataGrid(dataGridView1, SQL);
}
```

این دستورات با توجه به تغییر نام رئیس، اطلاعات dataGridView1 را فیلتر می‌کنند.

۶. پروژه را ذخیره و اجرا کنید تا شکل ۵ – ۹ ظاهر شود. با گزینه‌های "**ایجاد جدول گروه**"، "**ایجاد جدول دانشجو**" و "**ایجاد جدول دانشکده**" از منوی "**ایجاد جداول**" جدول‌های بانک اطلاعاتی را ایجاد کنید. در اینجا، برای ایجاد جدول گروه، برروی گزینه ایجاد جدول گروه کلیک کنید تا شکل ۶ – ۹ ظاهر شود.

دوباره گزینه "**ایجاد جدول گروه**" را مجدداً کلیک نمایید تا خروجی را به شکل ۷ – ۹ ببینید. دکمه OK را کلیک کنید. گزینه "**جدول دانشجو**" را کلیک مضاعف کرده، تا شکل ۸ – ۹ را مشاهده کنید. اطلاعات شکل ۸ – ۹ را وارد کرده، دکمه اضافه را کلیک کنید تا رکورد به جدول Student اضافه شود تا شکل مقابل ظاهر شود:

دکمه OK را کلیک کنید تا شکل ۹ – ۹ را ببینید (این اطلاعات در dataGridView1 اضافه شده‌اند). با دکمه‌های دیگر و گزینه‌های دیگر کار کنید تا بتوانید رکوردها را "**ویرایش**" و "**حذف**" کنید یا آن‌ها را نمایش دهید. البته همه این کارها را می‌توانید برای "**جدول گروه**" و "**جدول دانشکده**" انجام دهید.

شکل ۵ – ۹ اولین خروجی مثال ۱ – ۹ (فرم اصلی).

شکل ۶ – ۹ ایجاد جدول گروه.

شکل ۷ – ۹ خطا در ایجاد جدول.

شکل ۸ – ۹ فرم ورود اطلاعات جدول دانشجو.

شکل ۹ – ۹ رکورد اضافه شده در جدول دانشجو.

پیوست پروژه‌های تکمیلی

۱. پروژه‌ایی که پینگ‌پنگ را پیاده‌سازی می‌کند. این پروژه شامل موارد زیر می‌باشد:

- از Struct برای نمایش ساختار دایره، مستطیل و سایر اشکال هندسی مورد استفاده می‌کند.
- از کلاس به عنوان قالب (الگوی) برای نمایش دایره، مستطیل و سایر اشکال هندسی مورد نظر استفاده می‌کند.
- از آرایه برای ذخیره اشکال ایجاد شده و حرکت همه آن‌ها در زمین بازی استفاده می‌کند.

۲. پروژه‌ایی که بازی xo را پیاده‌سازی می‌کند. در این پروژه مباحث زیر بررسی می‌گردد:

- متد Click برای دریافت کلیک کاربر و انجام عکس‌العمل پیاده‌سازی شده است.
- متد Win جهت بررسی زمین بازی و یافتن بازیکن برنده پیاده‌سازی شده است.
- ایجاد Button در زمان اجرا انجام شده است.
- رویدادهای مربوط به Buttonها در زمان اجرا فراخوانی گردید.

۳. برنامه‌ایی که بازی شطرنج را پیاده‌سازی می‌کند. در این پروژه مفاهیم زیر بحث و بررسی می‌شود:

- کلاس Stone، کلاس پایه می‌باشد.
- کلاس Pawn، کلاس مشتق است.
- سازنده‌های کلاس پایه و مشتق پیاده‌سازی شده‌اند.
- واسطی (Interface) به نام IStone دارد که شامل متدی جهت بررسی امکان حرکت مهره می‌باشد. این واسط توسط کلاس Pawn به ارث برده می‌شود و این متد پیاده‌سازی می‌شود.
- عملگر == را تعریف مجدد نموده است تا در صورت یکی بودن رنگ مهره‌ها امکان حرکت وجود نداشته باشد.
- Delegate و Event (رویداد) را پیاده‌سازی کرده است تا به ازای حرکت هر مهره این رویداد ایجاد شده، کیش یا مات را بررسی نماید.
- آرایه‌ایی از Buttonها برای رسم زمین بازی ایجاد نموده است.
- متد Genericای به نام ShowStone ایجاد کرده است که با دریافت یک کلاس از جنس مهره‌ها نام مهره جابه‌جا شده را نمایش می‌دهد.

۴. پروژه‌ایی که وب کمپ را راه‌اندازی کرده و تصویر آن را برروی صفحه نمایش انتقال می‌دهد.

۵. پروژه‌ایی که امکان کشیدن و رها کردن عناصر یک ListBox به ListBox دیگر را فراهم می‌کند. در این برنامه همچنین می‌توان انتقال اطلاعات TextBox را به ListBox را انجام داد (با کشیدن و رها کردن). گزینه‌ها

در TextBox با کاما از یکدیگر جدا می‌شوند. هدف این پروژه آشنایی بیشتر با منوها، کنترل‌ها ToolTip، MenuStrip و پیاده‌سازی رویدادهای MouseDown، MouseMove، MouseUp، DragEnter و DragDrop برای کنترل‌های مختلف است.

۶.پروژه‌ایی که بانک اطلاعاتی پویای ایجاد می کند. این برنامه دارای اهداف زیر است:

- شناسایی سرویس‌دهنده‌های بانک‌های اطلاعاتی موجود در شبکه یا سیستم فعلی به صورت دستی یا خودکار.
- ایجاد بانک اطلاعاتی جدید به طوری که اطلاعات بانک اطلاعاتی را از کاربر دریافت می کند.
- ایجاد جداول جدید و ویرایش ساختار جداول در بانک اطلاعاتی ایجاد شده یا بانک‌های اطلاعاتی موجود.
- افزودن رکورد به جداول، حذف رکوردهای جداول و ویرایش رکوردهای موجود در جداول.
- ایجاد و ویرایش رویه‌های ذخیره شده.

منابع

۱. عباس نژاد، رمضان، آموزش گام به گام بانک اطلاعاتی با C# ، بابل انتشارات فن آوری نوین ، ۱۳۸۸.

۲. عباس نژاد، رمضان، حل مسائل C# ، بابل انتشارات فن آوری نوین ، ۱۳۸۸.

۳. عباس نژاد، رمضان، شمعلی زاده، علی ، رحیم پور، باقر ، آموزش گام به گام برنامه نویسی با C++، بابل انتشارات فن آوری نوین ، ۱۳۹۰.

4. Blackburn, Peter D., ADO.NET Examples and Best Practices for C# Programmers (Second Edition), Apress, 2002.

5. 1. Anders Hejlsberg,Mads Torgersen,Scott Wiltamuth,Peter Golde, "**The C# Programming Language Fourth Edition"**, Addison Wesley, 2011

6. 2. Daniel M. Solis, **"Illustrated C# 2010"**,Apress,2010.

7. 3. Allen Jones and Adam Freeman**," Visual C# 2010 Recipes A Problem Solution Approach"**, Apress, 2010

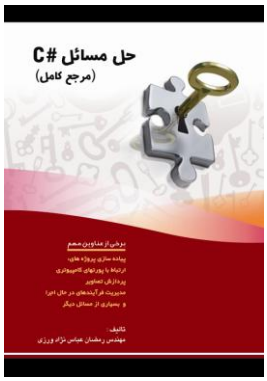
8. 4. Andrew Troelsen**," Pro C# 2010 and the .NET 4 Platform Fifth Edition"**,Apress, 2010.

9. 5. James Foxall**," SamsTeachYourself 24 in Hours Visual C#® 2008 Complete Starter Kit",**Sams, 2008

10. ۶. Herbert Schildt**," C# 4.0:The Complete Reference"**, McGraw-Hill, 2010.

11. 7. Matthew A, Stoecker, Steve Stein,**" MCTS.Exam 70-505 Microsoft .NET Framework 3.5 Windows Froms Application Development"**, Microsoft Press, 2009.

12. 8. Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner,**" C# 4 and .NET 4"**, Wiley, 2010



عنوان برخی از برنامه‌های این کتاب در زیر آمده است:

- حرکت بین TextBoxها با کلیدهای جهت‌نما، اعتبار سنجی TextBoxهـا بـرای دریافـت اطلاعـات فقط عددی، فقط حرفی، اعداد اعشاری و Email، انتخاب گزینه‌های ListBox و انتقال گزینه‌های انتخاب شده ListBox دیگر.
- کشیدن و رها کردن یک تصویر بر روی PictureBox، انتقال تصویر خاصی به TextBox، محو کردن فرم به مرور زمان.
- چرخش به سمت چپ یا راست یک آرایه، پیدا کردن اشتراک، اجتماع و تفاضل دو آرایه، پیاده‌سـازی رزرو بلیط هواپیمایی، پیاده‌سازی تبدیل جملات زبان انگلیسی به زبان خوکی، پیاده‌سازی تبدیل عـدد بـه معادل فارسی آن، پیاده‌سازی تبدیل تاریخ به معادل فارسی آن (خواندن تاریخ به فارسـی)، تبـدیل تـاریخ شمسی به میلادی و بر عکس، پیاده‌سازی بازی دوز، پیاده‌سازی کلاس اعداد کسری، پیـاده‌سـازی کـلاس تاریخ با امکان جمع، تفریق، تاریخ و غیره، پیاده‌سازی بازی Taxman.
- پیاده‌سازی کلاس‌های اشکال مختلف از قبیل شکل مربع، مستطیل برای آموزش مفاهیم کـلاس‌هـای پایـه، مشـتق abstract، virtual، override، پیـاده‌سـازی کـلاس‌هـای مشـتق، پایـه، abstract، virtual، override، پیاده‌سازی برنامه‌ای برای Shutdown، log off و Restart کردن کامپیوتر، پیاده‌سـازی اجـرای برنامه اجرایی خاص از طریق C#، پیاده‌سازی برنامه‌ای برای تغییر دسکتاپ کامپیوتر، پیاده‌سازی برنامـه‌ای

برای ارتباط با پورت‌ها، خواندن و نوشتن اطلاعات در پورت، پیاده‌سازی برنامه‌ای برای لایه گذاری تصاویر، پیاده‌سازی برنامه‌ای جهت کنترل CD برای باز کردن و بستن درایور CD.

- پیاده‌سازی برنامه‌ای برای فشرده سازی و خارج کردن فایل از حالت فشرده، پیاده‌سازی برنامه‌ای جهت نمایش درایوهای موجود در سیستم و نمایش خواص درایو انتخاب شده، پیاده‌سازی کپی یک دایرکتوری در دایوکتوری دیگر به صورت بازگشتی، پیاده‌سازی حرکت تصادفی توپ و یک آیکن در صفحه، پیاده‌سازی رسم فرم به شکل دایره یا بیضی.
- پیاده‌سازی برنامه تغییر رنگ زمینه فرم‌های باز، پیاده‌سازی برنامه ایجاد Shortcut از طریق #C، پیاده‌سازی برنامه‌هایی برای قفل کردن ایستگاه کاری، نوشتن متنی روی دسکتاپ، تغییر شفافیت کنترل خاص و اجرای مجدد برنامه، پیاده‌سازی برنامه‌ای برای نوشتن متن با جلوه ویژه، پیاده‌سازی برنامه‌ای برای کپی، چرخش و تغییر اندازه تصاویر و ۱۰ها برنامه دیگر.

فصل‌های کتاب:

فصل اول: بیان مقدمه‌ای بر C#، مفاهیم کلاس‌ها و چگونگی پیاده‌سازی آن.

فصل دوم: بیان تعریف بانک اطلاعاتی، مزایا و معایب و چگونگی اتصال به بانک اطلاعاتی SQL Server 2008.

فصل سوم: روش ایجاد بانک اطلاعاتی، جداول و تغییر جداول با دستورات SQL و مفاهیمی مانند ADO.NET، کلاس‌های Connection و Command (مانند SqlConnection و SqlCommand).

فصل چهارم: بیان کلاس‌هایی مانند DataSet، DataTable، DataRow و DataColumn.

فصل پنجم: بیان دستورات SQL مانند Insert، Update و Delete برای دستکاری داده، انقیاد داده‌ها به کنترل‌های روی فرم، چگونگی پیمایش رکوردهای جدول بانک اطلاعاتی و کلاس DataView.

فصل ششم: بیان مفهوم بازیابی داده از یک بانک اطلاعات (دستور SELECT، چگونگی پیوند جداول و پرس‌وجوهای متداخل)، مفهوم کاتالوگ در بانک اطلاعات، کلاس‌های DataAdapter و DataReader

فصل هفتم: بیان مفهوم و چگونگی ایجاد دید، توابع و رویه‌های ذخیره شده، چگونگی استفاده از رویه‌های ذخیره شده در C#.

فصل هشتم: بیان مفهوم تراکنش، خواص آن و پیاده‌سازی تراکنش با زبان C#

فصل نهم: بیان، مفهوم Crystal Report یا Microsoft Report و چگونگی استفاده از ارسال پارامترها از طریق فرم برای گزارش، چگونگی اتصال در زمان اجرا به بانک اطلاعاتی در Microsoft Reports، چگونگی اتصال به Dataset از طریق Microsoft Report و چگونگی نمایش رکوردهای خاص (رکوردهایی که دارای شرط خاص باشند) در Microsoft Report.

☒ **فصل دهم:** بیان پشتیبان‌گیری و بازیابی پشتیبان از طریق زبان ویژوال بیسیک‌نت یا C#. در این فصل، سه نوع پشتیبان‌گیری و بازیابی پشتیبان پیاده‌سازی گردید که عبارت‌اند از :

۱. پشتیبان‌گیری از طریق Backup Database و بازیابی پشتیبان از طریق دستور Restore Database.

۲. پشتیبان‌گیری و فشرده‌سازی اطلاعات به طوری که در جاهای دیگر قابل بازیابی و خواندن نیست (فقط از طریق همین برنامه می‌توان بازیابی را انجام داد).

۳. چگونگی Attach و Detach کردن بانک اطلاعات از طریق C#.