

# کتاب آموزش سی شارپ به زبان فارسی (مقدماتی)

مناسب برای افراد مبتدی که می خواهند  
بهترین زبان برنامه نویسی را فرا بگیرند





## فهرست

### اول آشنایی با مفاهیم پایه‌ای پردازش داده‌ها

۱	۱-۱-داده‌ها و اطلاعات
۵	۱-۲- انواع زبان‌های برنامه‌نویسی
۸	خودآزمایی فصل اول
۱	فعالیت

### دوم آشنایی با زبان C#

۱۲	۲-۱- آشنایی با زبان C#
۱۴	۲-۲- شروع برنامه‌نویسی
۱۵	۲-۳- اولین برنامه به زبان C#
۱۸	۲-۴- الگوی یک برنامه ساده به زبان C#
۱۸	۲-۵- کلاس (Class) چیست؟
۱۹	۲-۶- نحوه تعریف کلاس
۱۹	۲-۷- متدهای چیست؟
۲	۲-۸- استفاده از متدهای آماده
۳۸	خودآزمایی فصل دوم
۳۸	تمرین

۴۱	۳-۱- آشنایی با برنامه ویژوال استودیو
۴۳	۳-۲- ایجاد یک پروژه جدید در ویژوال استودیو
۴۶	۳-۳- معرفی بخش‌های اصلی ویژوال استودیو
۴۶	۳-۳-۱- نوار منو و نوار ابزار
۴۶	۳-۳-۲- پنجره ویرایشگر برنامه
۴۷	۳-۳-۳- پنجره لیست خطاهای (Error list)
۴۷	۳-۳-۴- پنجره (Solution Explorer)
۴۸	۳-۴- برنامه‌نویسی در محیط ویژوال استودیو
۵۲	۳-۵- ترجمه برنامه
۵۳	۳-۶- اجرای برنامه
۵۶	خودآزمایی فصل سوم
۵۶	تمرین

## چهارم آشنایی با انواع داده و متغیرها

۵۹	۴-۱- متغیر چیست؟
۵۹	۴-۲- روش تعریف و ایجاد متغیرها
۶	۴-۳- نوع داده (نوع متغیر)
۶۲	۴-۴- مقداردهی متغیرها
۶۵	۴-۵- نشان دادن محتوای متغیرها بر روی خروجی (صفحه نمایش)
۶۷	۴-۶- نحوه نام‌گذاری متغیرها
۶۹	۴-۷- کار با اعداد اعشاری
۷	۴-۷-۱- فرم نقطه شناور
۷۱	۴-۷-۲- دقت اعداد قابل نمایش در فرم نقطه شناور
۷۱	۴-۸- نوع داده منطقی با بولین (bool)
۷۱	۴-۹- نوع داده حرفی یا کاراکتری char
۷۳	۴-۱۰- نوع داده رشته‌ای (String)
۷۳	۴-۱۱- ۱- متغیر رشته‌ای
۷۳	۴-۱۲- ۲- عملیات بر روی داده‌ها یا متغیرهای رشته‌ای
۷۶	۴-۱۱- دریافت یک رشته از طریق ورودی استاندارد
۸۱	۴-۱۱- دریافت اعداد از طریق ورودی استاندارد
۸۵	خودآزمایی فصل چهارم
۸۸	تمرین

## پنجم عبارت‌های محاسباتی

۹	۱-۵_ عبارت چیست؟
۹۱	۲-۵_ عملگرها ریاضی یا حسابی
۱	۳-۵_ عملگرها افزایشی و کاهشی
۱۱	۴-۵_ عملگرها انتساب
۱۳	خودآزمایی فصل پنجم
۱۶	تمرین

## ششم دستورهای شرطی

۱۹	۱-۶_ عبارت منطقی یا بولین
۱۹	۲-۶_ عملگرها مقایسه‌ای
۱۱۱	۳-۶_ دستور شرطی if
۱۱۹	۴-۶_ دستور شرطی if-else
۱۲۴	۵-۶_ عملگرها منطقی
۱۳	۶-۶_ ساختار if - else پیچیده
۱۳۲	۷-۶_ دستور Switch
۱۳۷	خودآزمایی فصل ششم
۱۳۹	تمرین

## هفتم دستورات تکرار (حلقه‌ها)

۱۴۱	۱-۷_ دستورات تکرار شرطی
۱۴۲	۱-۷_ دستور حلقه شرطی while
۱۴۷	۲-۷_ دستور حلقه شرطی do - while
۱۵۳	۲-۷_ دستور حلقه for
۱۵۹	تمرین



## فصل اول

### آشنایی با مفاهیم پایه‌ای پردازش داده‌ها

در این فصل ابتدا مفاهیم پایه‌ای پردازش داده‌ها شامل : داده‌ها، اطلاعات و پردازش که در کتاب مبانی کامپیوتر خوانده‌اید، یادآوری می‌شود و سپس در ادامه این فصل، با برنامه، زبان برنامه‌نویسی و تقسیم‌بندی زبان‌های برنامه‌نویسی از نظر تزدیکی به زبان محاوره‌ای و لزوم یادگیری یک زبان برنامه‌نویسی آشنا می‌شوید.

پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند :

- ۱- مفهوم داده، اطلاعات و پردازش را بیان کند و آنها را به کار بندد.
- ۲- تعریف برنامه، برنامه‌نویسی، مترجم و انواع زبان‌های برنامه‌نویسی را از نظر سطح تزدیکی به زبان محاوره‌ای بیان کند.
- ۳- در یک برنامه ورودی، خروجی و پردازش را معین کند.
- ۴- هدف از برنامه‌نویسی را بیان نماید.

#### ۱-۱- داده‌ها و اطلاعات<sup>۱</sup>

اغلب مردم دو واژه داده‌ها و اطلاعات را به یک معنی می‌دانند در صورتی که مفهوم و کاربرد این دو واژه با یکدیگر متفاوت است.

داده‌ها، مجموعه‌ای از مقادیر در مورد یک موضوع یا شیء است که به صورت کمی با یک مقدار عددی و یا به صورت کیفی شناس داده می‌شود.

مثلاً آزمون درس ریاضی یک کلاس، نمراتی حاصل می‌شوند که این نمرات به عنوان داده‌ها در نظر گرفته می‌شوند.

۱۷/۵, ۱۴, ۱۹, ۱۱, ۱۸, ...

اسامی دانشآموزان یک کلاس و یا نام شهرهای استان محل سکونت شما مثال‌های دیگری از داده‌ها می‌باشد.

در بعضی موارد نیز برای به دست آوردن داده‌های مربوط به یک موضوع، مجبور به استفاده از لوازم و وسائل مخصوص هستیم، مثلاً در مورد دمای محیط، نیاز به استفاده از یک حسگر<sup>۱</sup> دما هستیم که به وسیله آن اندازه دمای را به دست آوریم. اندازه دمای محیط، یک داده است.

برای اینکه از داده‌ها بتوانیم بهتر استفاده کنیم لازم است بروی آنها محاسبات و یا به طور کلی عملیاتی را انجام دهیم. نتایج حاصل از این عملیات را اطلاعات می‌نامیم که می‌تواند مورد تفسیر و بررسی قرار گیرد و نتیجه بررسی آنها به دانش ختم گردد که دانش می‌تواند مبنای تصمیم‌گیری برای انجام کاری شود. مثلاً اگر داده‌هارا، نمرات ریاضی یک کلاس در نظر بگیریم و مجموع آنها را محاسبه و حاصل جمع را بر تعداد نمرات تقسیم کنیم، خارج قسمت به دست آمده میانگین یا معدل نمرات است. از روی میانگین نمرات می‌توان به سطح درس ریاضی کلاس بی برد. مثلاً اگر این میانگین کم باشد باید کلاس تقویتی ریاضی برای دانشآموزان آن کلاس برگزار کرد. شکل زیر رابطه بین داده‌ها، عملیات و اطلاعات را از چپ به راست نشان می‌دهد.



شکل ۱-۱—ارتباط بین داده‌ها، عملیات و اطلاعات

با توجه به شکل ۱-۱، مشاهده می‌شود که اطلاعات، حاصل انجام عملیات بر روی داده‌ها است. مجموعه محاسبات و عملیاتی که بر روی داده‌ها صورت می‌گیرد را پردازش<sup>۲</sup> می‌نامند. پردازش، گاهی ساده مانند محاسبه مجموع و یا خارج قسمت دو عدد می‌باشد و یا گاهی پیچیده مانند تشخیص شماره پلاک خودرو با استفاده از عکس گرفته شده از خودرو، توسط یک کامپیوتر است.

## نکته

داده‌ها، مقادیر خام و اولیه در مورد یک موضوع هستند.  
اطلاعات، نتایج حاصل از عملیات و محاسبات بر روی داده‌ها می‌باشد.  
مجموعه محاسبات و عملیاتی که بر روی داده‌ها صورت می‌گیرد را پردازش می‌نامند.  
کامپیوتر پردازشگر داده‌ها است.

در فرایند رسیدن از داده‌ها به اطلاعات، نکات زیر باید رعایت گردد:

۱- صحت داده‌ها : یعنی داده‌ها به درستی گردآوری شده باشند و داده اشتباه در بین آنها وجود نداشته باشد. مثلاً در بین نمرات درس ریاضی، نمره منفی (کمتر از صفر) و یا بالاتر از بیست نداشته باشیم و همچنین تعداد نمرات با تعداد داشن آموزان برابر باشد. و یا در مورد اندازه درجه حرارت محیط، سنسور دما به خوبی کار کند و معیوب نباشد و دمای محیط به درستی اندازه گیری شود.

۲- درستی انجام محاسبات : یعنی محاسبات و یا به طور کلی عملیاتی که بر روی داده‌ها صورت می‌گیرد با دقت و بدون اشتباه انجام شوند و در حین انجام عملیات لطمehای به داده‌ها وارد نشود.

۳- روش انجام پردازش : با توجه به هدفی که از گردآوری داده‌ها در نظر داریم باید پردازش مناسب نیز بر روی آنها انجام دهیم تا به اطلاعات مفید برسیم. استفاده از روش‌های بهینه و الگوریتم‌های مناسب در عمل پردازش توصیه می‌شود.

معمول‌آز کامپیوتر برای انجام پردازش بر روی داده‌ها استفاده می‌کنیم چون سرعت کامپیوتر در اجرای عملیات و دقت انجام محاسبات بالا است. همچنین از کامپیوتر برای ذخیره و نگهداری داده‌ها استفاده می‌شود تا بعداً پردازش بر روی آنها صورت گیرد.

## برای مطالعه

در پیامون ما دستگاه‌های مختلفی قرار دارند که در هر یک از آنها میکروکامپیوتری برای پردازش داده‌ها وجود دارد. مثلاً در یک دستگاه DVD Player استفاده از لیزر و حسگر، علامت‌های روی دی وی دی به عنوان داده‌ها خوانده شده و سپس با پردازش بر روی آنها تبدیل به صدا می‌شوند و به بلندگو فرستاده می‌شوند که برای ما قابل شنیدن و با معنی می‌باشد.

در داخل خودروهای امروزی حداقل یک سیستم کامپیوتری وجود دارد که برای کنترل اندازه تحويل سوخت به موتور به کار می‌رود. داده‌های ورودی این کامپیوتر توسط حسگرها مختلفی که برای اندازه گیری مقدار اکسیژن در هوا، اندازه دمای هوا و اندازه سرعت موتور وغیره تعییه شده‌اند تأمین می‌شود و کامپیوتر پس از پردازش داده‌های دریافتی و مقایسه آنها با داده‌های ثابت موجود در حافظه فقط خواندنی (EPROM)، می‌تواند اندازه سوخت و ترکیب آن با هوا را کنترل نماید و از طریق تزریق سوخت (انژکتور) به موتور بفرستد. سیستم کامپیوتری مربوطه را ECU (Engine Control Unit) می‌نامند. تصور کنید که اگر سنسورها دچار اشکال شوند چه اتفاقی می‌افتد!

کندکاوی

وسایلی که در منزل شما قرار دارد و فکر می کنید که احیاناً در آنها یک سیستم کامپیوتری کوچک قرار دارد را شناسایی کنید و تشخیص دهید که داده های ورودی آن دستگاه چیست و خروجی آن دستگاه کدام است. نتایج تحقیق را به کلاس ارایه دهید.

برای اینکه کامپیوتر بداند که چه پردازشی و یا چه عملیاتی را باید بر روی داده‌ها انجام دهد لازم است دستورات مناسب به آن داده شود.

به مجموعه دستوراتی که به کامپیوتر می‌فهماند که چه نوع پردازشی را بر روی داده‌ها انجام دهد و همچنین اطلاعات به دست آمده را چگونه نمایش دهد برنامه<sup>۱</sup> می‌گویند.

دستورات برنامه باید با یک زبان قابل فهم برای کامپیوتر، نوشته شود و با توجه به اینکه سخت افزار کامپیوتر بر منطق رقی (صفر و یک) بنا شده است لذا زبان قابل فهم کامپیوتر دنباله‌ای از کدهای صفر و یک است که به آن زبان ماشین<sup>۳</sup> می‌گویند. مثلاً برای جمع دو عدد ۵ و ۸ دستورهای زبان ماشین در یک پردازنده شرکت اینتل<sup>۴</sup> چنین است:

10111000 00000101 00000000 10111011 00001000 00000000 00000001 11011000

با توجه به اینکه نوشن دستورات به زبان ماشین وقت‌گیر و دشوار است و زبان ماشین هر پردازندۀ با پردازندۀ دیگر متفاوت است، زبان‌های دیگری طراحی شده‌اند که نوشن برنامه به آن زبان‌ها برای ما ساده‌تر از زبان ماشین است. البته پس از اینکه برنامه با زبان دیگری غیر از زبان ماشین نوشته شد، با استفاده از یک مترجم<sup>۳</sup>، باید به زبان ماشین تبدیل شود تا کامپیوتر بتواند آن را بفهمد و اجرا نماید. مترجم خود نیز یک برنامه کامپیوتری می‌باشد که وظیفه آن، ترجمه و تبدیل دستورات یک زبان سطح بالا، به کدهای زبان ماشین می‌باشد.

نوشتن دستورات لازم برای کنترل نحوه کار کامپیوتر، به طوری که کامپیوتر بتواند یک کار مشخص را انجام دهد را برنامه نویسی می‌گویند.

برنامه‌نویس شخصی است که آشنا به دستورات یک زبان برنامه‌نویسی باشد و با به کارگیری صحیح و مناسب دستورات، برنامه‌نویسی کند.

\\\_ Program

## ¶\_ Machine Language

¶\_Inte

¶\_Comp er

## نکته

به مجموعه دستوراتی که به کامپیوتر می‌فهماند که چه نوع پردازشی را باید بر روی داده‌ها انجام دهد و اطلاعات به دست آمده را چگونه نمایش دهد برنامه می‌گویند. زبان قابل فهم سخت افزار کامپیوتر، زبان ماشین نام دارد و مشکل از دنباله‌ای از کدهای ° و ۱ است.

برنامه‌ای که با زبانی غیر از زبان ماشین نوشته شود ابتدا باید توسط یک زم افزار متجم تبدیل به زبان ماشین شود و سپس برنامه ترجمه شده توسط کامپیوتر اجرا می‌گردد. متجم خود یک برنامه کامپیوترا است که وظیفه آن ترجمه برنامه نوشته شده به یک زبان، به کدهای زبان ماشین است.

## ۱-۲- انواع زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی را از نظر این که تا چه اندازه به زبان محاوره‌ای ما تزدیک باشند به صورت زیر دسته‌بندی می‌کنند :

- **زبان‌های سطح پایین<sup>۱</sup>** : زبان‌هایی که به زیان پردازشگر کامپیوتر (CPU) تزدیک باشد و مسلماً از زبان محاوره‌ای ما دور هستند زبان‌های سطح پایین نام دارند. زبان ماشین و زبان اسملی<sup>۲</sup> در گروه زبان‌های سطح پایین قرار دارند.
- **زبان‌های سطح بالا<sup>۳</sup>** : زبان برنامه‌نویسی که به زبان محاوره‌ای ما تزدیک باشد، زبان سطح بالا نام دارد. تاکنون صدها زبان برنامه‌نویسی سطح بالا در دانشگاه‌ها و یا شرکت‌های کامپیوترا طراحی و ابداع شده‌اند ولی بعضی از آنها عمومی نشدن و مورد استقبال برنامه‌نویسان قرار نگرفته‌اند و بعضی از زبان‌ها با آمدن زبان‌های جدید منسوخ شده‌اند. از میان زبان‌های رایج برنامه‌نویسی می‌توان به زبان VB، Java و C# اشاره نمود.

- **زبان‌های سطح میانی<sup>۴</sup>** : زبان‌هایی در این دسته قرار می‌گیرند که در آنها دستوراتی برای دسترسی راحت تر به سخت افزار پیش‌بینی شده باشد و همچنین به زبان عامیانه تزدیک باشند، مانند زبان C. برنامه‌نویسان از این زبان‌ها برای کار با سخت افزار کامپیوتر و برنامه‌ریزی و سایلی که در آنها

۱—Low Leve Language

۲—Assembly Language

۳—High Leve Language

۴—Medium Leve Language

پردازشگر وجود دارد استفاده می‌کنند.

در این کتاب با یکی از زبان‌های برنامه‌نویسی به نام C# (بخوانید C Sharp) آشنا خواهید شد که یک زبان سطح بالا می‌باشد.

قبل از شروع به یادگیری زبان برنامه‌نویسی، ممکن است این سؤال طرح شود که با توجه به وجود نرم افزارهای کاربردی<sup>۱</sup> آماده نظری Office، آیا لزومی دارد که یک زبان برنامه‌نویسی را یاد بگیریم تا لک نرم افزار جدید بسازیم؟

پاسخ: نرم افزارهای کاربردی آماده، برای تسهیل و انجام امور عادی و روزمره که بین کاربران مشترک است طراحی شده اند و به تدریج و در طی سالیان طولانی با توجه به بازخورد<sup>۲</sup> کاربران تکمیل تر شده اند. مثلاً برنامه صفحه گسترده<sup>۳</sup> (ابداع شده در سال ۱۹۷۸) یک نرم افزار کاربردی است که افراد و شرکت‌های بسیار زیادی از آن برای انجام امور روزمره خود استفاده می‌کنند. کاربران این گونه نرم افزارها باید از قابلیت‌هایی که در آن نرم افزار از قبل تدارک دیده شده است استفاده کنند و نیاز خود را با استفاده از آن امکانات مرفوع نمایند. با یادگیری یک زبان برنامه‌نویسی این انعطاف را خواهیم داشت که یک نرم افزار کاربردی مطابق با نیاز خود طراحی و برنامه‌نویسی کنیم. همانطور که ممکن است به جای اینکه یک لباس آماده از فروشگاهی خریداری کنیم به سراغ یک خیاط برویم تا یک لباس کاملاً سفارشی<sup>۴</sup> و مطابق با سلیقه و مدل مورد نظر ما درست کنند. البته طراحی و تولید یک نرم افزار کاربردی، یک کار سنگین، پیچیده و مستلزم صرف وقت زیادی است و به وسیله‌ی یک گروه از مهندسان با تخصص‌های مختلف طی چند مرحله انجام می‌شود که برنامه‌نویسی تنها یکی از مراحل آن است. البته نگران نباشید ما در این کتاب برنامه‌های کوچک و مقدماتی برای یادگیری زبان برنامه‌نویسی C# می‌نویسیم و در سال بعد شما با مراحل تولید یک نرم افزار و روش‌های طراحی و حل مسئله<sup>۵</sup> آشنا خواهید شد تا بتوانید با استفاده از ابزار برنامه‌نویسی، برنامه‌های کاربردی برای حل یک مشکل در یک سازمان بنویسید و آن را تجربه کنید.

---

۱\_App cat on Software

۲\_Feedback

۳\_Spreadsheet

۴\_Custom zed

۵\_Prob em So v ng

نقطه نظر یک برنامه نویس :

من برنامه های زیادی را نوشته ام ولی تاکنون برنامه ای بدون  
شناسایی و درک کامل مسئله از ابتدا، نتوشته ام. حتی برای کاری  
که بخواهم برای دوستم مجانی انجام دهم، همین روش را  
به کار می گیرم.



شکل ۱-۲ - نظر یک برنامه نویس

## خودآزمایی فصل اول

۱- در هر یک از موارد زیر داده، اطلاعات و پردازش را مشخص نمایید.

الف) نرم افزار تهیه کارنامه در مدارس آموزش و پرورش، جمع نمرات و معدل هر دانش آموز را براساس نمراتش، محاسبه و چاپ می کند.

ب) در یک شرکت مهندسی، حقوق دریافتی کارکنان براساس میزان حضور ساعت کار در شرکت محاسبه و پس از کسر مالیات و حق بیمه پرداخت می شود و بر روی فیش چاپ می شود.

ج) یک روش الکترونیکی در پرداخت عوارض خودرو در هنگام عبور از بزرگراه‌ها، استفاده از تکنولوژی RFID است. در این روش یک برچسب<sup>۱</sup> مخصوص در روی شیشه خودرو چسبانده می شود که دارای یک کد منحصر به فرد است و اطلاعات شماره پلاک خودرو و کد مربوطه و همچنین میزان اعتبار پولی در هنگام خرید برچسب، در کامپیوتر ثبت می شود. هر گاه خودرو بدون توقف از محل پرداخت عوارض عبور کند از طریق حسگرهای امواج رادیویی، کد برچسب خودرو شناسایی شده و این کد به کامپیوتر داده می شود (شکل ۱-۳) و در نتیجه شماره پلاک خودرو و میزان اعتبار آن استخراج می شود، عوارض عبور از بزرگراه از اعتبار مربوطه به صورت خودکار کسر می شود (شکل ۱-۴).

۲- من چه سطحی از زبان برنامه نویسی هستم؟

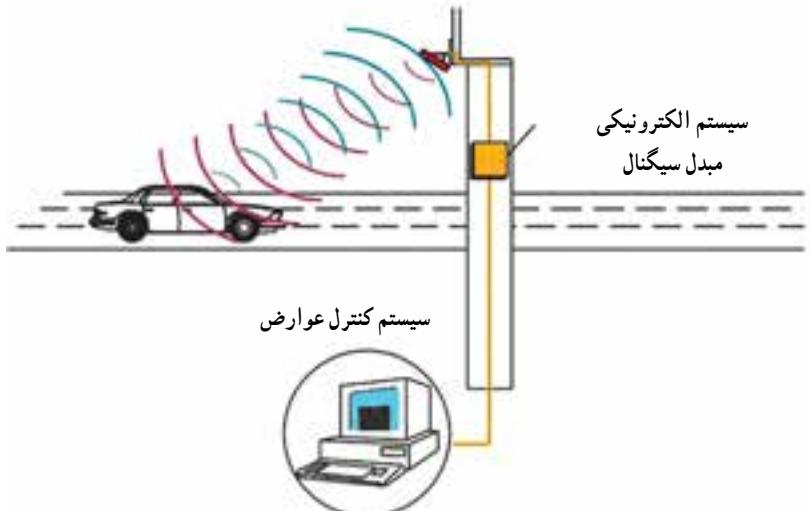
آنتن (فرستنده و گیرنده امواج)



شکل ۱-۳- مرحله اول ورود خودرو به گذرگاه عوارض بزرگراه

۱-Rad o-Frequency Ident f cat on شناسایی با امواج رادیویی

۲- Tag



شکل ۴-۱- مرحله دوم تشخیص اطلاعات خودرو و کسر عوارض از اعتبار پولی

تا کنون زبان‌های زیادی در سطح من تولید شده است، اما تنها برخی از آنها مورد اقبال متخصصان قرار گرفته است. من به زبان محاوره‌ای خیلی نزدیک هستم.

۳- توضیحات ستون چپ را با اصطلاح ستون سمت راست تطابق دهید. (یک مورد اضافی است).

- |   |                  |
|---|------------------|
| الف - برنامه‌ای برای تبدیل کد سطح بالا به کد قابل فهم برای سخت‌افزار  | ۱- برنامه        |
| ب - دنباله‌ای از کدهای صفر و یک   | ۲- برنامه‌نویسی  |
| ج - مجموعه‌ای از دستورالعمل‌ها برای پردازش داده‌ها  | ۳- مترجم         |
| د - نوشتن دستورات لازم برای حل یک مسئله   | ۴- زبان ماشین    |
| ه - زبان برنامه‌نویسی نزدیک به زبان محاوره‌ای   | ۵- زبان سطح بالا |
| و - داده‌ها و ورودی‌های یک برنامه   |                  |
| ۴- در یک برنامه، منظور از ورودی همان ..... و منظور از خروجی همان ..... است.   |                  |
| ۵- در محاسبه عبارت $y = x^z$ ، ورودی (ها) و خروجی (ها) کدامند؟  |                  |
| ۶- زمان ورود و خروج کارکنان یک شرکت با عکس‌برداری از چهره آنان و به کمک کامپیوتر صورت می‌گیرد. در این روش ورودی (داده)، پردازش و خروجی (اطلاعات) را تعیین کنید. |                  |
| ۷- چرا لازم است یک زبان برنامه‌نویسی را یاد بگیریم؟   |                  |

## فعالیت

- ۱- با استفاده از جستجو در اینترنت، چند زبان سطح بالای متدال امروزی را پیدا کرده و آنها را نام ببرید.
- ۲- دستگاه‌های الکترونیکی و یا کامپیوترا که در زندگی ما وارد شده‌اند و همه به آنها وابسته شده‌ایم مانند تلفن همراه و یا کنسول‌های بازی را در نظر بگیرید. با توجه به کاربرد آنها، داده‌های ورودی به آنها و همچنین با توجه به نوع پردازش، اطلاعات خروجی در آنها را شناسایی کنید.
- ۳- اگر به اتفاق پدر یا مادر به تعمیرگاه اتومبیل رفتید، فرست را از دست ندهید و از تعمیرکاران سؤال کنید که دستگاه ECU (توضیح داده شده در قسمت «آیا می‌دانید») در کجا خودرو قرار دارد؟ (در همه خودروهای امروزی حتی مانند پراید نیز وجود دارد. تعجب نکنید.) ECU و حسگرهای آن را شناسایی کنید. همچنین وقتی به منزل برگشتید با استفاده از اینترنت منظور از کلمات دیاگ و فلاش کردن که توسط تعمیرکاران استفاده می‌شود را جستجو کرده و متوجه مفهوم آن شوید.

## واژگان و اصطلاحات انگلیسی فصل اول

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Application Software	
۲	Assembly Language	
۳	Compiler	
۴	Customized	
۵	Data	
۶	Feedback	
۷	High Level Language	
۸	Information	
۹	Installation	
۱۰	Low Level Language	
۱۱	Machine Language	
۱۲	Medium Level Language	
۱۳	Problem Solving	
۱۴	Process	
۱۵	Program	
۱۶	Sensor	
۱۷	Spreadsheet	



## فصل دوه

### آشنایی با زبان C#

در این فصل ابتدا مختصری در مورد پیدایش زبان C# و ارتباط آن با زبان‌های دیگر بیان می‌شود و سپس ساختار یک برنامه به زبان C# معرفی می‌گردد و با دو اصطلاح کلاس و متده به صورت مقدماتی آشنا می‌شوید. در ادامه این فصل روش نام گذاری پاسکال برای انتخاب نام کلاس توضیح داده می‌شود و در انتهای فصل در قسمت کار در کارگاه، اولین برنامه به زبان C# را نوشته و ترجمه و اجرای آن را تجربه خواهید کرد.

#### پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند:

- ۱- قالب کلی یک برنامه ساده در زبان C# را بیان کند.
- ۲- نحوه تعریف یک کلاس و متده در یک برنامه ساده بکار بندد.
- ۳- با استفاده از یک ویرایشگر، برنامه‌ی ساده بنویسد و آن را ذخیره نماید.
- ۴- برنامه ذخیره شده را با استفاده از مترجم در پنجه کنسول ترجمه کرده و سپس اجرا نماید.
- ۵- با استفاده از متدهای مربوط به رنگ، تغییری در خروجی برنامه‌ی خود به وجود آورد.

### ۱-۲- آشنایی با زبان C#

زبان C# یک زبان سطح بالا، شی‌گرا<sup>۱</sup> و همه منظوره است که به وسیله شرکت مایکروسافت هم زمان با پیدایش لایه نرم‌افزاری جدید آن به نام .NET. ابداع و توسعه پیدا کرده است. از نرم‌افزارهای متنوع و گوناگونی از جمله نرم‌افزارهای اداری و برنامه‌های کاربردی تحت وب گرفته تا نرم‌افزارهایی برای تلفن همراه و بازی‌های کامپیوتری، با زبان C# و با استفاده از لایه .NET. تولید می‌شود.

زبان C# شباهت زیادی به زبان‌های C و Java دارد و ویژگی‌هایی را از آنها تقلید کرده، یا بعضی

امکانات آنها را بهبود داده است. تلاش شده است که بهترین ویژگی‌ها گردآوری شود، اما برخلاف زبان جاوا که متن باز<sup>۱</sup> است، C در انحصار اختیار سازنده آن یعنی شرکت مایکروسافت است. زبان‌های C و Java هر دو به زبان C بر می‌گردند که در سال ۱۹۷۰ ابداع شد و معروفیت آن به دلیل نوشتن سیستم عامل UNIX به سیله آن بود. زبان C، یک زبان حرفه‌ای است که دست برنامه‌نویس را برای نوشتن برنامه و دسترسی به سخت افزار، باز می‌گذارد و دارای انعطاف پیش زیادی است، به همین دلیل کمتر اشتباهات منطقی برنامه‌نویس را کنترل می‌نماید. اما در زبان #C، در هنگام ترجمه و همچنین اجرای برنامه دقت زیادی بر روی تطبیق و به کارگیری داده‌ها صورت می‌گیرد تا از اشتباهات دستوری برنامه‌نویس یا کاربر جلوگیری نماید.

## برای مطالعه

BJARNE STROUSTRUP



پیارنه استراس تروب، زبان C را بر مبنای C و با اضافه کردن ویژگی شی گرایی در سال ۱۹۷۹ طراحی کرد و در سال ۱۹۸۳ این زبان به نام C منتشر شد.

DENNIS M. RITCHIE



دنسی ریچی متولد ۱۹۴۱ زبان C را بر مبنای زبان B، طی سال‌های ۱۹۶۹ تا ۱۹۷۳ طراحی کرد و همچنین با آفای Ken Thompson که طراح زبان B بود در طراحی و ایجاد سیستم عامل یونیکس همکاری کرد. وی در سال ۲۰۱۱ در اثر بیماری فوت کرد.

ANDERS HEJLSBERG



اندرس هجلزبرگ مهندس نرم افزار متولد سال ۱۹۶۰ است وی در ساخت و ابداع چندین زبان برنامه نویسی معروف و موفق همکاری داشته است. او سازنده اصلی زبان توربوباسکال و دلفی بود و همچنین مدیر تیم طراحی زبان #C است.

JAMES ARTHUR GOSLING



جیمز آرتور گسلینگ متولد سال ۱۹۵۵، به عنوان پدر زبان جاو شناخته می‌شود. وی در سال ۱۹۹۴ به همراه تیمش زبان جاو را بر پایه زبان C و C بدیع کرد وی در بت نام Oak که نام درختی در جلوی دفتر کار و بود، برای یعنی زبان منتخب کرد، ما بعدها به Java تغییر نام پید کرد. زبان جاو با هدف ساده کردن و بالا بردن منیت و با شعار برنامه ریک بار نویس و هر جا جر کن طرحی شده است.

شکل ۱-۲-۱- پدیدآورندگان زبان‌های برنامه‌نویسی C, C++, Java, C#

## برای مطالعه

### لایه نرم افزاری .NET Framework. شرکت مایکروسافت

این لایه نرم افزاری دارای نسخه های مختلفی می باشد و همراه با نسخه های ویندوز منتشر می شود؛ آخرین نسخه آن را می توانید از سایت شرکت مایکروسافت دانلود کنید. در حال حاضر (فروردين ۱۳۹۳) نسخه ۴.۵ ارائه شده است. این لایه شامل موارد زیر است :

- زبان های برنامه نویسی .NET. مانند C#, J#, F# و VB
- محیطی برای اجرای برنامه ها (CLR)
- یک سری ابزارهای برنامه نویسی مانند مترجم csc که برنامه های C# را به کدهای زبان ماشین ترجمه می کند.
- یک سری کتابخانه های استاندارد مثل ADO.NET که برای دسترسی به بانک های اطلاعاتی می باشد.

زبان برنامه نویسی C# یک زبان برنامه نویسی سطح بالا، همه منظوره و شیء گرا است که به وسیله شرکت مایکروسافت تولید شده است. تیم طراحی این شرکت به سرپرستی آقای Anders Hejlsberg در طی تولید.NET Framework زبان C# را ابداع کرده است. نسخه های مختلف این زبان همراه با.NET. عرضه شده است . نسخه C# 5.0 در ۱۵ آگوست ۲۰۱۲ منتشر شده است.

## ۲-۲- شروع برنامه نویسی

همان طور که برای تهیه و پخت یک غذا، به مواد اولیه، لوازم آشپزی و دستور پخت<sup>۱</sup> نیاز داریم برای تولید یک برنامه نیز، به یک کامپیوتر یا لپ تاپ، لوازم برنامه نویسی (یک ویرایشگر متون<sup>۲</sup> و یک برنامه مترجم) و همچنین به یک الگوریتم نیاز داریم. اگر یک کامپیوتر با سیستم عامل ویندوز ۷ یا بالاتر در دسترس باشد تقریباً تمام مواد اولیه و لوازم مورد نیاز را در اختیار داریم.

<sup>۱</sup>\_ Common Language Runtime

<sup>۲</sup>\_ Recipe

<sup>۳</sup>\_ Text Editor

در این صورت با طی مراحل زیر می توانیم برنامه ای را نوشته، ترجمه کرده و سپس اجرا نماییم.

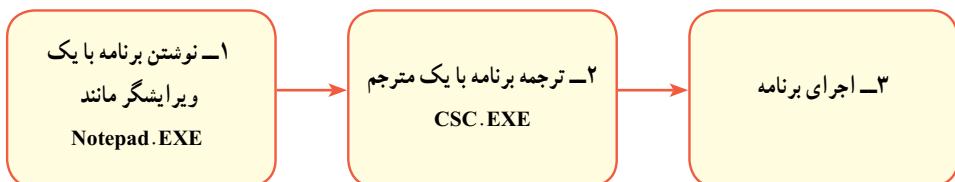
۱- نوشتن برنامه و ذخیره آن با استفاده از یک ویرایشگر مانند برنامه Notepad ویندوز

۲- ترجمه برنامه ذخیره شده به وسیله مترجم زبان C# به نام CSC.EXE

این مترجم با نصب .NET Framework. در روی کامپیوتر قرار می گیرد (در پیوست ۲، نحوه نصب آن توضیح داده شده است).

۳- اجرای برنامه ترجمه شده

نمودار ۱-۲ این مراحل را به ترتیب از چپ به راست نشان می دهد.



نمودار ۱-۲- مراحل برنامه نویسی، ترجمه و اجرا

در انتهای این فصل، در قسمت کار در کارگاه، تمام مراحل بالا را به صورت عملی تجربه خواهید کرد.

## ۲-۲- اولین برنامه به زبان C#

با یک برنامه ساده به زبان C# آشنا می شویم. برنامه ۱-۲ را در زیر مشاهده کنید.

```
class WelcomeToCSharp
{
    static void Main()
    {
        System.Console.WriteLine("Welcome To C#!");
    }
}
```

برنامه ۱-۲- اولین برنامه به زبان C#

این برنامه کوچک فقط یک پیام خوشامدگویی بر روی صفحه نمایش نشان می دهد. رنگ های کلمات که در این برنامه مشاهده می کنید، تنها برای کمک به واضح شدن برنامه برای خواننده به کار

گرفته شده است و تاثیری بر روی صفحه نمایش ندارد. همان طور که در برنامه Notepad آنچه که می نویسید همگی با یک رنگ نوشته می شود.

**سؤال:** آیا با مشاهده این برنامه و بدون اطلاعات قبلی و یا کمک از دیگران، می توانید

حدس بزنید که چه پیامی بر روی صفحه نشان داده می شود؟

برای این که با این برنامه آشنا شویم و یاد بگیریم که چگونه باید به زبان `C#` برنامه بنویسیم از دو

جهنه این برنامه را بررسی خواهیم کرد :

الف) نگاه جزئی تر در حد کلمات و علامت ها

ب) نگاه کلی تر در حد تقسیم بندی یک برنامه به قسمت های مختلف

بانگاهی جزئی تر به برنامه ۱-۲، مشاهده می کنید که این برنامه از تعدادی کلمه و علامت تشکیل شده است. بعضی از کلمات مانند `void`, `class`, `static` کلمات شناخته شده برای زبان `C#` هستند و دارای معنی و مفهوم ثابتی هستند به این نوع کلمات، کلمات کلیدی یا رزرو شده گفته می شود. کلمات رزرو شده<sup>۱</sup> به رنگ آبی در این کتاب نوشته شده اند و به تدریج با آنها آشنا می شوید.

بعضی از کلمات دیگر مانند `WelcomeToCSharp` نامی است که به وسیله برنامه نویس و طبق سلیقه وی انتخاب می شود. به این نام ها شناسه<sup>۲</sup> می گویند. برنامه نویس در انتخاب شناسه ها باید ضوابطی را رعایت کند که در فصل چهارم با آن آشنا می شوید.

علامت هایی مانند {}, {}, ( ) و نیز در این برنامه دیده می شود که معمولاً<sup>۳</sup> برای شروع یا پایان یک قسمت استفاده می گردد.

بانگاهی دیگر و کلی تر به برنامه ۱-۲، مشاهده می کنیم که یک برنامه ساده از یک قسمت کلی به نام کلاس تشکیل شده است که با کلمه کلیدی `class` مشخص می شود و شروع و پایان آن با علامت آکولاد باز و بسته تعیین می گردد. در جلوی کلمه کلیدی `class`، یک نام (شناسه) دلخواه مثلاً `WelcomeToCSharp` نوشته می شود که بیان کننده کار برنامه است. قسمت کلاس برنامه را در شکل زیر مشاهده کنید.

```
class WelcomeToCSharp
```

```
{
```

```
}
```

کلاس برنامه ۱-۲

اگر درون کلاس [WelcomeToCSharp](#) را نگاه کنیم یک قسمت دیگر را خواهیم دید که چنین شروع شده است :

```
static void Main()
```

شروع و پایان این قسمت نیز با علامت‌های آکولاد باز و بسته، مشخص شده است. به این قسمت متند<sup>۱</sup> **Main** می‌گوییم که بدنه اجرایی برنامه است هر دستوری که در این قسمت نوشته شود به وسیله‌ی کامپیوتر به ترتیب اجرا می‌شود. دستورهای برنامه خود را در این قسمت می‌نویسیم.

```
static void Main()
```

```
{  
    System.Console.WriteLine("Welcome To C#!");  
}
```

متند **Main** برنامه ۲-۱

آخرین قسمتی که در برنامه ۲-۱، در داخل متند **Main** قابل تشخیص است، یک دستور اجرایی است و به کامپیوتر اعلام می‌کند که چه باید انجام دهد که در این برنامه، نمایش یک پیام است :

```
System.Console.WriteLine("Welcome To C#!");
```

با اجرای دستور بالا، پیام خوشامدگویی **Welcome to C#!** بر روی صفحه نمایش، نشان داده می‌شود. آیا شما قبل از درست حدس زده بودید که برنامه ۲-۱، چه پیامی را بر روی صفحه نمایش، نشان می‌دهد؟!

به وسیله دستور بالا، هر آنچه که داخل علامت‌های نقل قول **" فرار داشته باشد، بر روی صفحه نمایش نشان داده می‌شود حتی اگر به زبانی غیر از انگلیسی مثلًا فارسی نوشته شده باشد. توجه داشته باشید که خود علامت‌های نقل قول بر روی صفحه نمایش، نشان داده نمی‌شوند. بلکه این علامت‌ها برای مشخص کردن شروع و پایان عبارتی است که می‌خواهیم روی صفحه نشان داده شود.**

---

۱- Method

توجه داشته باشید که زبان C# مانند زبان‌های C و Java نسبت به حروف کوچک و بزرگ حساس است و چنانچه قصد دارید برنامه‌ای را در کامپیوتر وارد کنید به دیکته و نوع حروف کوچک و بزرگ کلمات توجه داشته باشید. مثلاً کلمات static و void باید با حروف کوچک نوشته شود ولی حرف اول کلمه Main باید حرف بزرگ (M) باشد.

## ۲-۴- الگوی یک برنامه ساده به زبان C#

یک برنامه کاربردی نوشته شده به زبان C#، شامل مجموعه‌ای از کلاس‌ها است که هر یک از آنها نیز شامل تعدادی متدهستند. اما در یک برنامه ساده مانند برنامه ۱-۲، تنها یک کلاس وجود دارد که در آن نیز فقط یک متده است. این متده نام Main() است و نقطه آغاز اجرای برنامه است و الگوریتم خود را برای اجرا در آن می‌نویسیم. الگو یا ساختار کلی یک برنامه ساده به زبان C# در زیر آمده است. الگوی زیر را به خاطر بسپارید.

```
یک نام دلخواه class
{
    static void Main()
    {
        دستورات مربوط به انجام یک کار
    }
}
```

الگوی یک برنامه ساده به زبان C#

## ۲-۵- کلاس (class) چیست؟

کلاس یک مفهوم اساسی در برنامه‌نویسی شی‌گرا است که در کتاب برنامه‌سازی ۲ به تفصیل بحث می‌شود. در اینجا اگر بخواهیم به طور ساده در مورد معنی و مفهوم کلاس صحبت کنیم، باید بگوییم که کلاس به عنوان یک قالب یا الگویی می‌باشد که در آن داده‌هایی تعریف می‌شود. این داده‌ها

مربوط به یک موضوع است و عملیاتی که می‌توان بر روی آنها انجام داد. در زبان C# گنجینه‌ای از کلاس‌های مختلف و کاربردی، از قبیل تعریف شده و آماده وجود دارد که برنامه نویس کافی است آنها را بشناسد و در برنامه استفاده نماید. یک کلاس آماده در زبان C# است که عملیات مختلف ورودی و یا خروجی (بر روی صفحه نمایش و یا صفحه کلید) در آن تعریف شده است.

**۱-۵-۲- نحوه تعریف کلاس :** در زبان C# این امکان برای برنامه نویس فراهم است که کلاس جدیدی را تعریف کند. همان طور که در زیر مشاهده می‌کنید از کلمه کلیدی class برای تعریف و مشخص کردن یک کلاس جدید استفاده می‌شود. در جلوی کلمه class، یک نام دلخواه ذکر می‌گردد که نام کلاس است. مانند [WelcomeToCSharp](#)

نام کلاس class

{

تعریف داده‌ها

و عملیات بر روی آنها

}

الگوی یک کلاس

## نکته

**نام گذاری کلاس :** نام یک کلاس به وسیله برنامه نویس نام گذاری می‌شود. سعی کنید یک نام با معنی و مطابق با کار برنامه انتخاب کنید. ممکن است این نام از چند کلمه تشکیل شده باشد. بین کلمات ناید فاصله بگذارید ولی برای این که خواندن نام به راحتی انجام شود و تشخیص کلمات آسان باشد، از روش پاسکال استفاده کنید که در آن، اولین حرف هر کلمه با حرف بزرگ نوشته می‌شود.

## ۶-۲- متدهای کلاس

همان طور که گفته شد در داخل کلاس، عملیات بر روی داده‌ها و یا الگوریتم انجام یک کار تعریف می‌شود. متدهای مجموعه‌ای از دستورات است که برای انجام یک کار لازم است. هر متدهای مطابق با عملکردش نام گذاری می‌شود و همچنین دارای یک جفت پرانتز باز و بسته است که در آن ممکن است ورودی‌هایی ذکر شود که برای انجام کار لازم است.

در برنامه‌های زبان C#، ممکن است متدهای زیادی تعریف و یا مورد استفاده قرار گیرند، اما حتماً باید متدى به نام Main() تعریف شده باشد که نقطه آغاز اجرای برنامه است و اجرای یک برنامه از اولین دستور داخل آن شروع می‌شود.

کلمات static و void در قالب کلی متى Main() ویژگی‌های متى را مشخص می‌کنند که در کتاب برنامه سازی ۲ با آن آشنا می‌شوید.

```
static void Main()
```

```
{
```

```
    ; دستور شماره ۱
```

```
    ; دستور شماره ۲
```

```
    ; دستور شماره ۳
```

ادامه دستورات

```
}
```

قالب کلی متى (Main())

در این کتاب مانند برنامه ۱-۲، فقط به تعریف متى Main() می‌پردازیم و از متدهای آماده در زبان C# استفاده خواهیم کرد.

۱-۶-۲- استفاده از متدهای آماده : تعداد زیادی متى در کلاس‌های آماده زبان C# مانند وجود دارد که هر یک از آنها، برای انجام کاری در نظر گرفته شده است. مثلاً متى WriteLine() از کلاس Console برای نشان دادن پیام روی صفحه نمایش در نظر گرفته شده است که در برنامه ۱-۱ از آن استفاده کردیم :

```
System.Console.WriteLine("Welcome To C#!");
```

همان‌طور که برای آدرس دادن منزل خود به دیگران، نام منطقه، خیابان، کوچه و شماره پلاک را ذکر می‌کنید، برای استفاده از یک متى نیز باید نام فضا یا حوزه، نام کلاس و سپس نام متى را مشخص کنید و برای جدا کردن آنها از یکدیگر، علامت نقطه بین آنها بنویسید (نمودار ۲-۲).

نام حوزه  
System

نام کلاس  
Console

نام متدها  
WriteLine()

نمودار ۲-۲- طریقه استفاده از یک متدها (System.Console.WriteLine())

به این ترتیب برای استفاده از متدها WriteLine() در برنامه ۱-۲ مشاهده می‌کنید که فضای نامی<sup>۱</sup> System و نام کلاس Console و در آخر نام متدها نوشته شده است که با علامت نقطه از یکدیگر جدا شده‌اند. متدهای دیگری نیز در کلاس Console وجود دارد که در این فصل با برخی از آنها آشنا می‌شوید.

برنامه زیر برای نمایش دو پیام بر روی صفحه نوشته شده است :

```
class WelcomeToCSharp
{
    static void Main()
    {
        System.Console.WriteLine("Welcome To C#!");
        // Insert a blank line
        System.Console.WriteLine();
        System.Console.WriteLine("This is my first program.");
    }
}
```

برنامه ۲-۲- استفاده از توضیحات در متن برنامه

برنامه ۲-۲ مانند برنامه ۱-۲ است با این تفاوت که سه خط دیگر به متدها Main() اضافه شده است. خط اول فقط یک توضیح<sup>۲</sup> برای خواننده برنامه و یا برنامه نویسان می‌باشد و توضیح می‌دهد که خط بعدی برنامه چه عملی را انجام می‌دهد. نشانه توضیحات علامت // (دو بار کلید /) است و مترجم با دیدن این علامت متوجه می‌شود که این خط یک توضیح است؛ بنابراین آن را به زبان ماشین ترجمه نمی‌کند.

خط دوم یک دستور اجرایی است :

System.Console.WriteLine();

در این خط از متد ( ) استفاده شده است با این تفاوت که داخل پرانتز، خالی است. اجرای این دستور سبب می شود که روی صفحه نمایش یک سطر خالی ایجاد شود. دستور آخر، پیام This is my first program. را روی صفحه نمایش نشان می دهد.

با توجه به برنامه ۲-۲، پیامی را که می خواهید نمایش داده شود باید بین علامت های نقل قول " قرار دهید. برای مثال اگر اسم شما Mohammad است و بخواهید روی صفحه نمایش نشان داده شود، باید به صورت زیر بنویسید :

System.Console.WriteLine("Mohammad");

اگر بخواهید نام و نام خانوادگی خود را در دو سطر، نمایش دهید، در این صورت می توانید دو بار از متد ( ) استفاده کنید. مثلا برای نمایش "Mohammad" و "Ghasemi" چنین می نویسیم :

System.Console.WriteLine("Mohammad");

System.Console.WriteLine("Ghasemi");

حروف، علامت ها و عبارتی که ما بین علامت های نقل قول نوشته می شود را رشته<sup>۱</sup> می نامند. این حروف می توانند فارسی، انگلیسی یا به هر زبانی باشد. مانند نام "Mohammad" و یا یک رمز عبور "Mehran2014". هر یک از حروف و علامت ها را نیز یک کاراکتر<sup>۲</sup> می نامند. برای مثال Mohammad از ۸ کاراکتر و رمز عبور Mehran2014 از ۱۰ کاراکتر تشکیل شده است. چنان چه فاصله در رشته وجود داشته باشد، فاصله نیز یک کاراکتر محسوب می شود.

## نکته

برای درج توضیحات در برنامه، اگر یک خط باشد از علامت // و چنان چه چند خط باشد از علامت /\* توضیحات \*/ استفاده کنید :

// Display a greeting message

/\*

FileName: welcome.cs ... Date : 05\_07\_2014

Display a greeting message

\*/

## برای مطالعه

زبان برنامه‌نویسی C# از لایه نرم افزاری.NET استفاده می‌کند و مایکروسافت.NET را برای ویندوز طراحی کرده و به این ترتیب روی ویندوز کار می‌کند و اگر کسی بخواهد بر روی سیستم عامل دیگری، برنامه C# را اجرا کند باید لایه نرم افزاری مطابق با.NET را بر روی آن سیستم داشته باشد. خوشبختانه برنامه‌هایی مانند Mono وجود دارند که بر روی سیستم عامل‌های دیگری غیر از ویندوز نیز نصب می‌شود و دارای مترجم C# می‌باشد و به این وسیله برنامه‌های نوشته شده به این زبان را بر روی سیستم‌های دیگر غیر از ویندوز می‌توان ترجمه و اجرا کرد. برنامه Mono برای سیستم عامل‌های زیر وجود دارد :

Android, BSD, iOS, Linux, OS X, Windows, Solaris and UNIX

سری به سایت Mono بزنید.

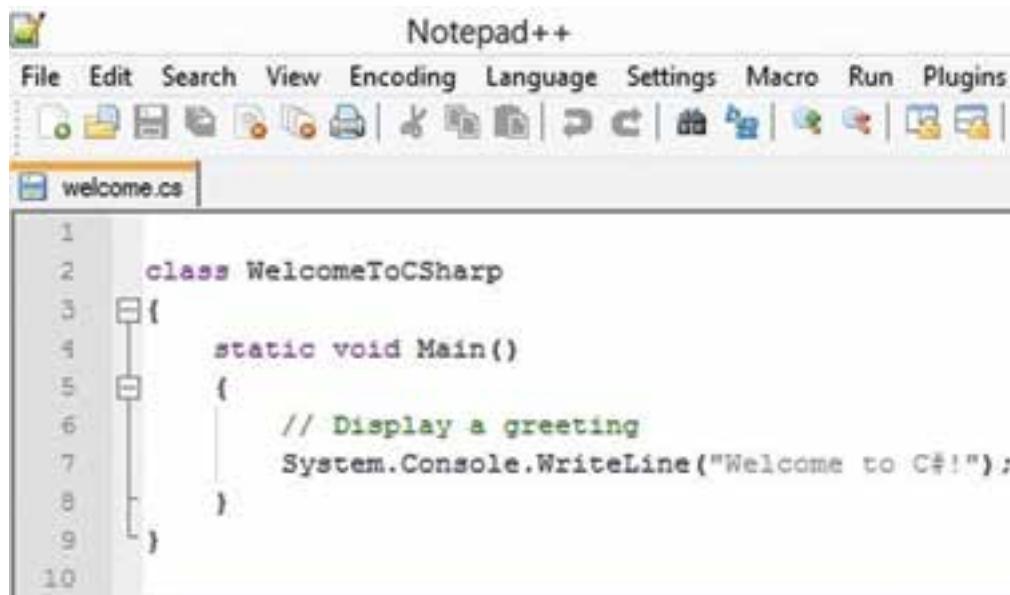
[www.go-mono.com](http://www.go-mono.com)



شکل ۲-۲- برنامه نویسی C# بر روی سیستم عامل غیر از ویندوز

قدم اول: نوشتن و تایپ برنامه: برای نوشتن یک برنامه ساده مانند برنامه ۲-۱، که در این فصل مورد بررسی قرار گرفت، نیاز به یک ویرایشگر متنی<sup>۱</sup> است. یک ویرایشگر متنی قادر است حروف، کلمات و آنچه را که تایپ می‌کنید بدون در نظر گرفتن اطلاعات نوع فونت، اندازه حروف و رنگ در یک فایل ذخیره کند. برنامه Notepad یک ویرایشگر ساده است که همراه با سیستم عامل ویندوز در روی کامپیوتر نصب می‌شود.

علاوه بر برنامه Notepad، می‌توانیم از ویرایشگر دیگری مانند برنامه Notepad که به صورت رایگان از طریق سایت<sup>۲</sup> آن قابل دانلود است استفاده نماییم. این ویرایشگر به منظور برنامه نویسی به زبان‌های مختلف طراحی شده است به طوری که کلمات رزرو شده، رشته‌ها و توضیحات در این ویرایشگر با رنگ‌های مختلف نشان داده می‌شود (شکل ۲-۳). البته برای استفاده از این ویرایشگر باید ابتدا از منوی Language، زبان برنامه نویسی مورد نظر خود را #C انتخاب کنید.



شکل ۲-۳- تصویری از محیط ویرایشگر Notepad

۱- Text Editor

۲- <http://notepad-plus-plus.org>

توجه داشته باشید که از برنامه Word استفاده نکنید که در این صورت، کدهای اضافی مربوط به صفحه‌بندی، رنگ و فونت را نیز به فایل شما اضافه می‌کند که مترجم در هنگام ترجمه برنامه، انتظار آنها را ندارد و دچار مشکل می‌شود.



● در هنگام تایپ دستورات، صرفنظر از این که از چه ویرایشگری (Notepad, Notepad#) استفاده می‌کنید، باید دقت داشته باشید که زبان C# نسبت به حروف کوچک و بزرگ حساس است؛ بنابراین برنامه را دقیقاً مانند کتاب تایپ کنید (شکل ۲-۴).

پس از آن که برنامه ۱-۲ را تایپ کردیم، باید آن را ذخیره نماییم. برای این منظور به منوی File بروید و گزینه... Save As... را انتخاب کنید. در هنگام ذخیره فایل دقت کنید که فایل را در کجا و در چه مسیری ذخیره می‌نمایید و نام فایل مناسبی را برای آن انتخاب کنید و پسوند آن را cs. قرار دهید (شاید بهتر باشد یک پوشه در یکی از درایوها به نام خودتان بسازید و فایل را در آن ذخیره کنید). در این تمرین نام فایل را welcome.cs قرار دهید.

```
File Edit Format View Help
```

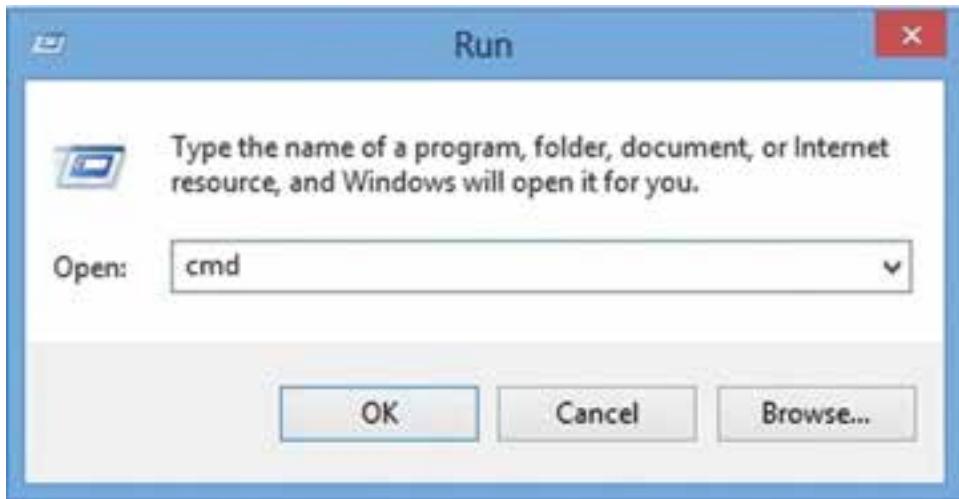
```
class WelcomeToCSharp
{
    static void Main()
    {
        // Display a greeting
        System.Console.WriteLine("Welcome to C#!");
    }
}
```

به حروف کوچک و بزرگ توجه داشته باشید.

شکل ۲-۴- تصویری از محیط ویرایشگر Notepad

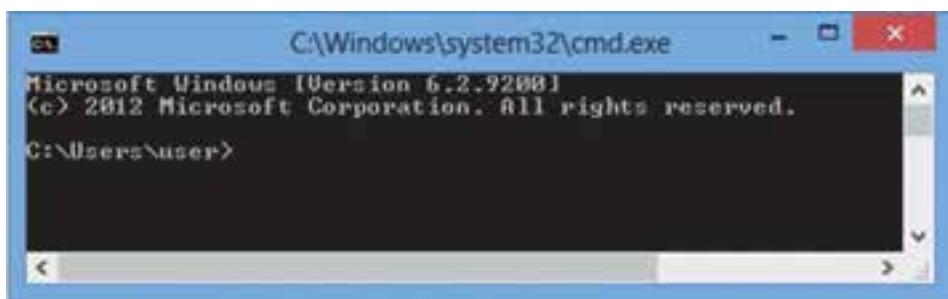
قدم دوم : ترجمه و اجرای برنامه : پس از نوشتن و ذخیره کردن برنامه، لازم است ابتدا برنامه را ترجمه کنیم و اگر اشکالی در تایپ آن وجود دارد آن را برطرف کنیم.  
برای ترجمه برنامه مراحل زیر را دنبال کنید.

۱- از طریق گزینه Run فرمان cmd را اجرا کنید، تا وارد پنجره فرمان شویم (شکل ۲-۵).



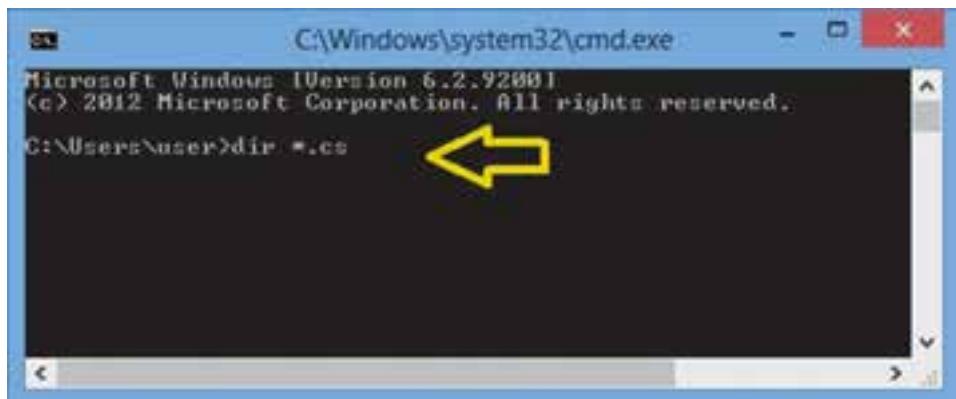
شکل ۲-۵- فرمان cmd

۲- پنجره فرمان ظاهر می شود (نوشته ها در شکل، ممکن است با آنچه در پنجره کامپیوتر شما دیده می شود متفاوت باشد) (شکل ۲-۶).



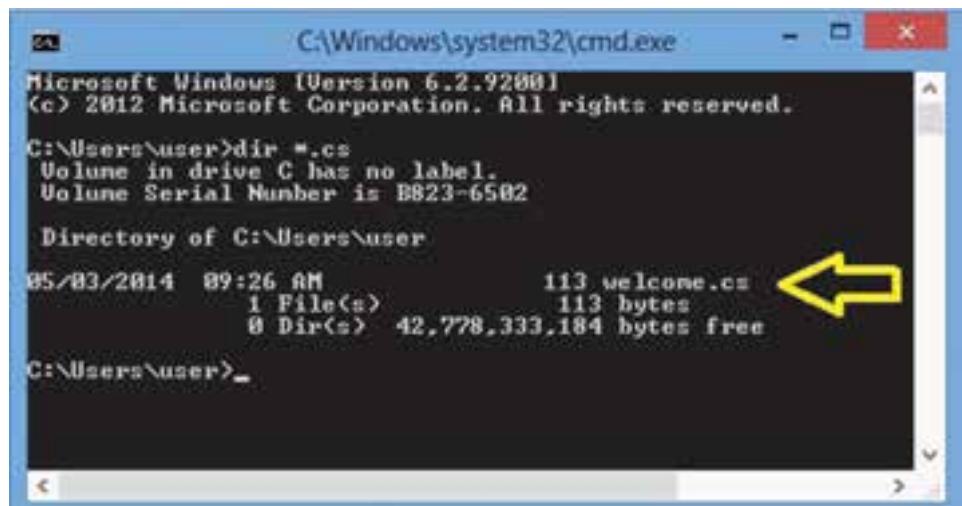
شکل ۲-۶- پنجره command prompt

**۳** در پنجره Command Prompt استفاده می‌کنیم و با توجه به این که پسوند cs می‌باشد، با تایپ فرمان زیر از وجود فایل برنامه مطمئن می‌شویم (شکل ۲-۷). اگر فایل را پیدا نکردید باید وارد پوشه‌ای شوید که برنامه را در آنجا ذخیره کرده‌اید. به این ترتیب از دستور cd برای وارد شدن به پوشه موردنظر خود استفاده کنید. شاید دستور .. نیز برای وقتی که می‌خواهید به یک پوشه بالاتر بروید مفید باشد.



شکل ۲-۷ - فرمان dir

**۴** اگر مرحله قبل را به درستی انجام دهید، باید شکل ۲-۸ را مشاهده کنید.



شکل ۲-۸ - مشخصات فایل برنامه

**۵** توجه داشته باشید اگر برنامه **.Net Framework** بر روی کامپیوتر شما قبلاً نصب شده باشد برنامه ای به نام **csc.exe** برای ترجمه برنامه های زبان **C#** در اختیار دارد. پس از یافتن فایل خود، با استفاده از این مترجم، برنامه خود را ترجمه نمایید. در پنجره فرمان از دستور زیر استفاده کنید : منظور از **filename.cs** نام فایل مورد نظر شما است.

csc filename.cs

```
C:\Users\user>dir *.cs
Volume in drive C has no label.
Volume Serial Number is B823-6582

Directory of C:\Users\user

05/03/2014  09:26 AM           113 welcome.cs
               1 File(s)      113 bytes
               0 Dir(s)  42,778,288,128 bytes free

C:\Users\user>csc welcome.cs
```

شكل ۲-۹— فرمان ترجمه برنامه

**۶** با اجرای دستور بالا مترجم شروع به ترجمه برنامه می کند و اگر همه کارها به درستی انجام شده باشد (برنامه **.NET**). قبلاً نصب شده باشد، تایپ دستور **csc** را درست انجام داده باشید، فایل **csc** به درستی در مسیر جستجوی سیستم عامل (path) معرفی شده باشد و برنامه هیچ خطایی نداشته باشد.) شکل ۲-۱۰ را مشاهده خواهید کرد.

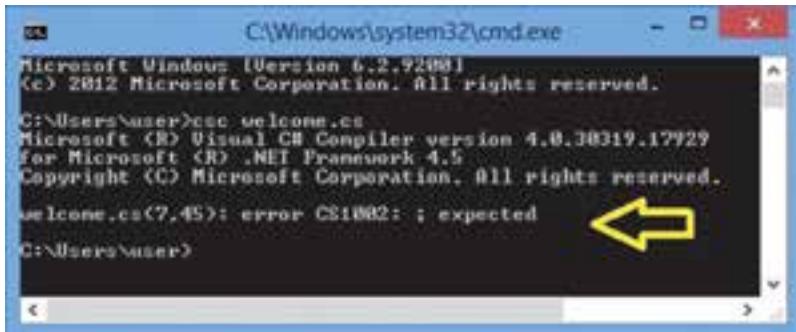
```
C:\Windows\system32\cmd.exe
Directory of C:\Users\user
05/03/2014  09:26 AM           113 welcome.cs
               1 File(s)      113 bytes
               0 Dir(s)  42,778,288,128 bytes free

C:\Users\user>csc welcome.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Users\user>
```

شكل ۲-۱۰— نتیجه ترجمه برنامه

**۷**- اگر در تایپ برنامه اشتباهی انجام داده باشد، مترجم نمی‌تواند برنامه را ترجمه کند و در این صورت خطای خطاها را گزارش می‌دهد. برای مثال اگر فراموش کرده باشید علامت نقطه ویرگول را در انتهای دستور (System.Console.WriteLine()) بنویسید، با ترجمه برنامه خطای شکل ۷-۲ ظاهر می‌شود.



```
C:\Windows\system32\cmd.exe
Microsoft Windows (Version 6.2.9200)
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\user>csc welcome.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

welcome.cs(7,45): error CS1002: ; expected
C:\Users\user>
```

شکل ۷-۲- پیام خطای مترجم برای نوشتن علامت ; در خط ۷ برنامه

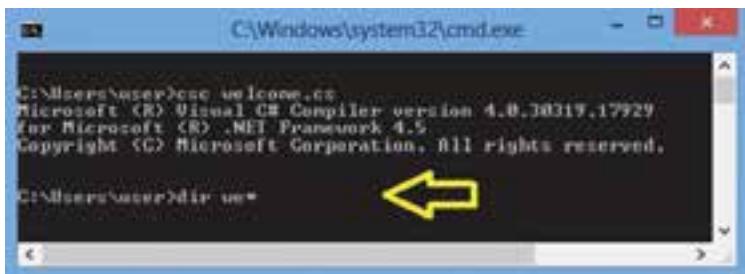
در خط آخر این شکل مشاهده می‌کنید که از سمت چپ، ابتدا نام فایل برنامه یعنی welcome.cs و سپس در جلوی آن دو عدد ۷ و ۴۵ نوشته شده است که مکان خطا را در برنامه نشان می‌دهد. عدد اول (۷) شماره سطر و عدد دوم (۴۵) شماره ستون محل خطا در برنامه است. بعد از این دو عدد، کد خطا (error CS1002) و سپس توضیح آن (expected) (;) ذکر شده است. شما باید توضیح خط را با دقت بخوانید و معنی آن را درک کنید تا بتوانید اشکال برنامه را برطرف کنید. در این مثال توضیح خط چنین است : علامت ; فراموش شده برای برطرف کردن خطای بالا لازم است برنامه را به وسیله ویرایشگر باز کنید و به سطر ۷ و ستون ۴۵ بروید و علامت نقطه ویرگول را اضافه کنید و سپس برنامه را تحت همان نام قبلی ذخیره کنید و دوباره به پنجره فرمان بازگردید و عمل ترجمه را انجام دهید. اگر خطای رخ نداد به مرحله بعدی بروید و گرنه باید دوباره عملیات رفع اشکال را تکرار کنید.

### نکته

اشتباه معمول برنامه نویسان در هنگام نوشتن برنامه، فراموش کردن علامت نقطه ویرگول در انتهای دستورات است. در این صورت خطای صادره از طرف مترجم چنین است :

error CS1002: ; expected

**۸** در صورت ترجمه موفق برنامه، فایل جدیدی ساخته می‌شود. با دستور Dir از وجود آن مطمئن می‌شویم (شکل ۲-۱۲).



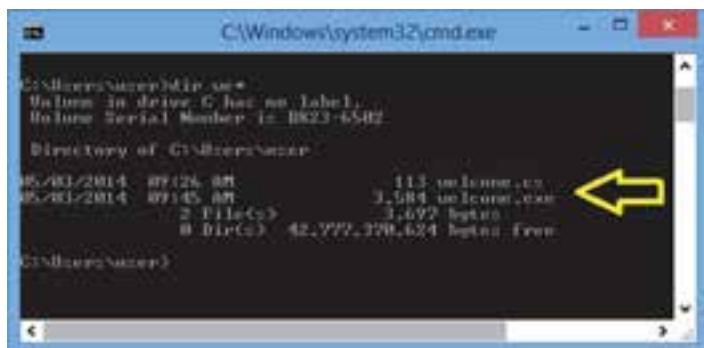
```
C:\Windows\system32\cmd.exe
C:\Users\user> welcome.cs
Microsoft (R) Visual C# Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Users\user>dir *.*

```

شکل ۲-۱۲- جستجوی فایل ترجمه شده

**۹** همان طور که در شکل ۲-۱۳ مشاهده می‌کنید، فایل جدیدی با همان نام welcome و لی با پسوند EXE ساخته شده است.



```
C:\Windows\system32\cmd.exe
C:\Users\user>dir *.*
Volume in drive C has no label.
Volume Serial Number is 8823-6502

Directory of C:\Users\user

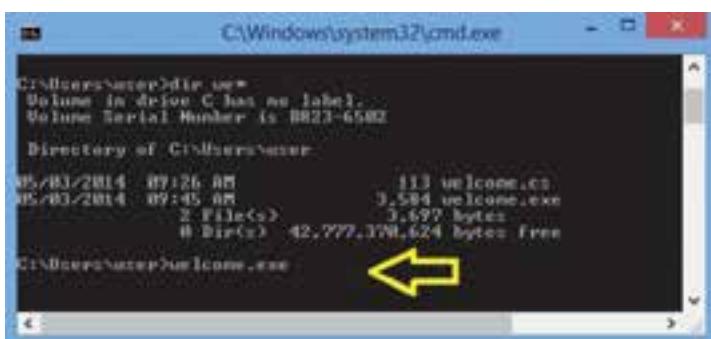
05/03/2014  09:26  AM           113 welcome.cs
05/03/2014  09:45  AM          3,584 welcome.exe
               2 File(s)        3,697 bytes
               0 Dir(s)   42,777,378,624 bytes Free

C:\Users\user>

```

شکل ۲-۱۳- لیست فایل‌های برنامه و ترجمه شده

**۱۰** بعد از اینکه فایل اجرایی ساخته شد، می‌توانید برنامه را اجرا نمایید. کافی است نام آن را در پنجره فرمان بنویسید و کلید Enter را بزنید (شکل ۲-۱۴).



```
C:\Windows\system32\cmd.exe
C:\Users\user>dir *.*
Volume in drive C has no label.
Volume Serial Number is 8823-6502

Directory of C:\Users\user

05/03/2014  09:26  AM           113 welcome.cs
05/03/2014  09:45  AM          3,584 welcome.exe
               2 File(s)        3,697 bytes
               0 Dir(s)   42,777,378,624 bytes Free

C:\Users\user>welcome.exe

```

شکل ۲-۱۴- اجرای فایل ترجمه شده یا برنامه اجرایی

۱۱- نتیجه اجرای برنامه به صورت شکل ۲-۱۵ خواهد بود.

C:\Windows\system32\cmd.exe

```
C:\Users\user>dir user
Volume in drive C has no label.
Volume Serial Number is B823-6502

Directory of C:\Users\user

05/03/2014  09:26 AM           113 welcome.cs
05/03/2014  09:45 AM          3,684 welcome.exe
               2 File(s)        3,697 bytes
               0 Dir(s)   42,222,378,624 bytes free

C:\Users\user>welcome.exe
Welcome to C#!
C:\Users\user>
```

A yellow arrow points from the text "نتیجه اجرای برنامه" to the output of the command "welcome.exe".

شکل ۲-۱۵- نتیجه اجرای برنامه

تبrik می‌گوییم که توانستید اولین برنامه خود را به زبان #C بنویسید و آن را ترجمه و اجرا نمایید.

۱۲- برنامه را با ویرایشگر باز کنید و یک دستور مانند شکل ۲-۱۶ به آن اضافه کنید. (تغییرات

بارنگ تیره مشخص شده است).

برنامه را تحت همان نام قبلی ذخیره کنید و مراحل ترجمه و اجرا (مرحله ۵ به بعد) را تکرار کنید.

welcome.cs - Notepad

```
File Edit Format View Help
```

```
class WelcomeToCSharp
{
    static void Main()
    {
        // Display a greeting
        System.Console.WriteLine("Welcome to (#!");
        System.Console.WriteLine("This is my first program");
    }
}
```

شکل ۲-۱۶- اضافه کردن یک دستور به برنامه

۱۳— دوباره برنامه را با ویرایشگر باز کنید و در اولین دستور به جای متده WriteLine() از متده Write() استفاده کنید.



```
File Edit Format View Help
welcome.cs - Notepad
class WelcomeToCSharp
{
    static void Main()
    {
        // Display a greeting
        System.Console.WriteLine("Welcome to C#!");
        System.Console.WriteLine("This is my first program");
    }
}
```

شکل ۲-۱۷— استفاده از متده Write()

چه تفاوتی در اجرای برنامه حاصل می شود؟ بیام ها چگونه نشان داده شدند؟

۱۴— برنامه را به حالت اولیه خود بازگردانید و یک دستور نیز به صورت شکل ۲-۱۸ به آن اضافه کنید. برنامه را ذخیره کنید و سپس ترجمه و اجرا نمایید. چه تغییری در خروجی ایجاد می شود؟



```
File Edit Format View Help
welcome.cs - Notepad
class WelcomeToCSharp
{
    static void Main()
    {
        // Display a greeting
        System.Console.WriteLine("Welcome to C#!");
        System.Console.WriteLine();
        System.Console.WriteLine("This is my first program");
    }
}
```

شکل ۲-۱۸— استفاده از متده WriteLine() برای ایجاد یک خط خالی

**۱۵** همان طور که در شکل ۲-۱۸ مشاهده می کنید، برای هر بار استفاده از متده WriteLine() مجبور هستیم کلمات System و Console را ذکر کنیم. برای کوتاه کردن این دستورات می توانیم در ابتدای برنامه، فضای نامی<sup>۱</sup> System را معرفی کنیم که در آن کلاس Console تعریف شده است. در این صورت می توانیم در داخل برنامه، کلمه System را از ابتدای دستورات حذف کنیم. برای معرفی فضای نامی از دستور using به صورت زیر استفاده می کنیم :

فضای نامی using

برنامه قبلی را باز کرده و تغییرات زیر را اعمال کنید و آن را اجرا نمایید (شکل ۲-۱۹).

```
welcome2.cs - Notepad
File Edit Format View Help
using System;
class WelcomeToCSharp
{
    static void Main()
    {
        // Display a greeting
        Console.WriteLine("Welcome to C#!");
        Console.WriteLine();
        Console.WriteLine("This is my first program");
    }
}
```

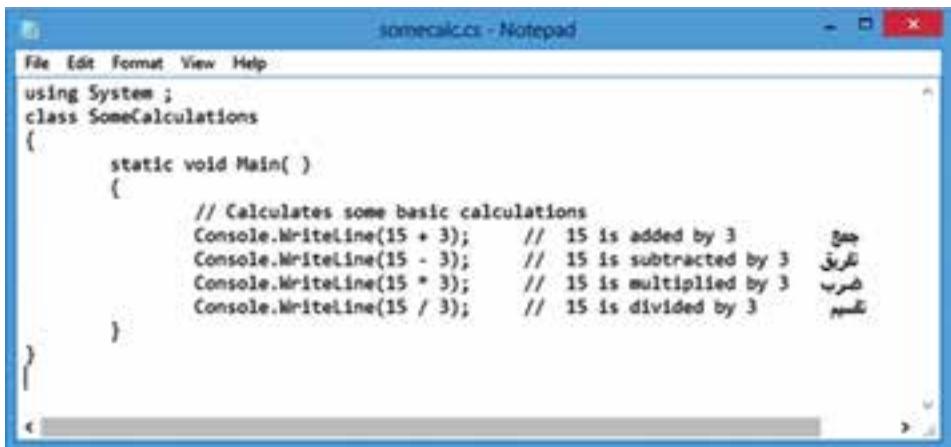
شکل ۲-۱۹—استفاده از فضای نام

**۱۶** از متده WriteLine() علاوه بر نمایش یک پیام می توان تیجه یک عبارت ریاضی را نیز نمایش داد. برنامه ای برای انجام چهار عمل اصلی بر روی دو عدد نوشته شده است (شکل ۲-۲۰). این برنامه را با توجه به نکته زیر تایپ نموده و سپس ترجمه و اجرا نمایید. و به اعداد نشان داده شده برروی صفحه توجه کنید. آیا محاسبات درست انجام شده است؟

### نکته

برای این که سریع تر بتوانید این برنامه را تایپ کنید، یکی از برنامه های قبلی خود را با ویرایشگر باز کنید و فقط نام کلاس و دستورات داخل متده Main را تغییر دهید و سپس تحت نام جدیدی آن را ذخیره (Save As...) کنید. توضیحات برنامه می تواند به زبان فارسی نیز باشد.

۱- برای توضیح بیشتر به ضمیمه ۴ در انتهای کتاب مراجعه کنید.



#### شكل ٢٠— برنامه حفار عمل اصلی

**۱۷** برنامه شکل ۲-۲ را با اعداد دیگری آزمایش کنید یعنی به جای ۳ و ۱۵ اعداد دلخواه دیگری مانند ۲۵ و ۴۰ قرار دهید و سپس برنامه را ترجمه و اجرا نمایید. اعدادی که روی صفحه شان داده می‌شوند را یادداشت کنید، آیا محاسبات درست انجام شده است؟

-۱۸- اگر به خروجی دستور زیر توجه کنید

```
Console.WriteLine(40 / 25);
```

مشاهده خواهید کرد که عدد ۱ چاپ می‌شود چون تقسیم صحیح و بدون اعشار انجام می‌شود. در صورتی که جواب صحیح و دقیق عدد  $1.\overline{6}$  است. برای این که کامپیوتر را مجبور به انجام عمل تقسیم اعشاری کنیم، لازم است دست کم یکی از اعداد را به صورت اعشاری بنویسیم. یعنی عدد  $4.$  را به صورت  $4.$   $0$  یا عدد  $25.$   $0$  را به صورت  $25.$   $0$  بنویسید. حال برنامه را در ویرایشگر باز کرده و یکی از اعداد در دستور تقسیم را به صورت اعشاری بنویسید و سپس برنامه را با همان نام قبلی ذخیره و ترجیحات اجرا نماید.

```
Console.WriteLine(40.0 / 25);
```

**۱۹**- برای این که پیام ها بهتر دیده شوند، می توانیم ابتدا با استفاده از متدها Clear نمایش را پاک کنیم. به این منظور برنامه را با ویرایشگر باز کنید و متدها ذکور را به برنامه طبق شکل اضافه کنید. برنامه را تحت نام **فعلی ذخیره کنید و سپس ترجمه و اجرا نمایید.**

```
using System;
class SomeCalculations
{
    static void Main()
    {
        // Clear console screen
        Console.Clear();
        // Calculates some basic calculations
        Console.WriteLine(40 + 15);      // 40 is added by 15
        Console.WriteLine(40 - 15);      // 40 is subtracted by 15
        Console.WriteLine(40 * 15);      // 40 is multiplied by 15
        Console.WriteLine(40.0 / 15);    // 40 is divided by 15
    }
}
```

شکل ۲-۲۱—استفاده از متد Clear() برای پاک کردن صفحه

۲۰. پنجره فرمان را بیندید وارد محیط میزکار<sup>۱</sup> شوید و با استفاده از Computer وارد پوشاهی شوید که فایل‌های خود را در آن ذخیره کرده‌اید. فایل‌های اجرایی (EXE) تولید شده را پیدا کنید. با دوبار کلیک بر روی آیکن آن، برنامه را اجرا کنید. چه اتفاقی می‌افتد؟

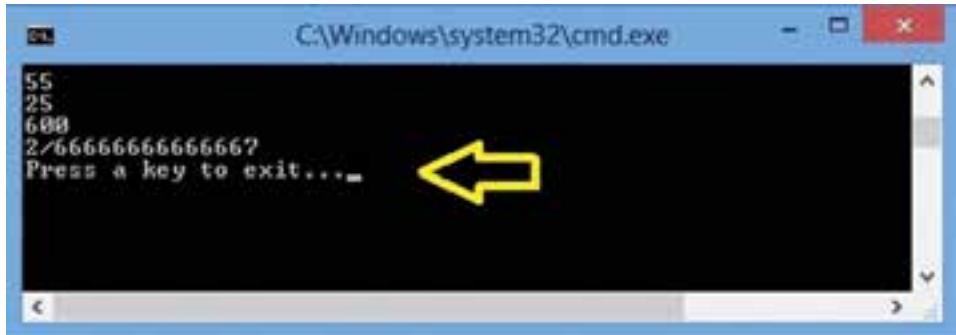
در یک لحظه صفحه کنسول باز شده و به سرعت بسته می‌شود و شما فرصت نمی‌کنید نتایج برنامه را مشاهده کنید. لازم است در انتهای برنامه، دستوری بنویسید که کامپیوتر را به صورت موقتی متوقف کند تا شما بتوانید نتایج محاسبات را ببیند. به این منظور از متد ReadKey() استفاده می‌کنیم. با اجرای این متد، کامپیوتر منتظر زدن کلیدی باقی می‌ماند. البته برای این که کاربر بداند که چرا اجرای برنامه متوقف شده است و کامپیوتر منتظر دریافت چه چیزی است، یک پیام با استفاده از متد Write() روی صفحه چاپ می‌کنیم. شکل ۲-۲۲ استفاده از این متدها را نشان می‌دهد. برنامه چهار عمل اصلی را باز کنید و تغییرات زیر را در آن اعمال و برنامه را ترجمه و اجرا کنید.

```
using System;
class SomeCalculations
{
    static void Main()
    {
        // Clear console screen
        Console.Clear();
        // Calculates some basic calculations
        Console.WriteLine(40 + 15);      // 40 is added by 15
        Console.WriteLine(40 - 15);      // 40 is subtracted by 15
        Console.WriteLine(40 * 15);      // 40 is multiplied by 15
        Console.WriteLine(40.0 / 15);    // 40 is divided by 15

        Console.Write("Press any key to exit..."); // Wait until a key is pressed.
        Console.ReadKey();
    }
}
```

شکل ۲-۲۲—استفاده از متد Clear() برای پاک کردن صفحه

با اجرای برنامه، مشاهده خواهد کرد که نتایج محاسبات و پیام... Press any key to exit... را روی صفحه نشان داده می‌شود و سپس کامپیوتر متوقف می‌شود و با زدن یک کلید برنامه پایان می‌باید (شکل ۲-۲۳).



شکل ۲-۲۳—استفاده از متده ReadKey برای دریافت یک کلید و توقف موقتی برنامه

کار با رنگ‌ها : جعبه رنگ ۱۶‌تایی در `C#` است. نام این رنگ‌ها در جدول ۲-۱ آمده است.

۲۱—ویرگی `BackgroundColor` رنگ زمینه کنسول را نشان می‌دهد.

مثال : با استفاده از جعبه رنگ `ConsoleColor` رنگ زمینه را آبی کنید :

```
Console.BackgroundColor ConsoleColor.Blue;
```

با اجرای دستور بالا تغییری در رنگ زمینه مشاهده نمی‌شود. برای تغییر رنگ زمینه لازم است از متده `Clear` کلاس `Console` استفاده کنیم. دستور زیر را در خط بعدی بنویسید.

```
Console.Clear();
```

۲۲—ویرگی `ForegroundColor` رنگ قلم کنسول را نشان می‌دهد.

مثال : کلمه IRAN را با رنگ قرمز روی صفحه بنویسید :

```
Console.ForegroundColor ConsoleColor.Red;
```

```
Console.WriteLine("IRAN");
```

جدول ۲-۱ - جعبه رنگ ConsoleColor

نمونه رنگ	نام رنگ	نام رنگ در ConsoleColor
Black	مشکی	Black
DarkBlue	سورمه ای	DarkBlue
DarkGreen	سبز تیره	DarkGreen
DarkCyan	فیروزه ای تیره	DarkCyan
DarkRed	قرمز تیره	DarkRed
DarkMagenta	بنفش	DarkMagenta
DarkYellow	زرد تیره	DarkYellow
DarkGray	خاکستری تیره	DarkGray
Blue	آبی	Blue
Green	سبز	Green
Cyan	فیروزه ای	Cyan
Red	قرمز	Red
Magenta	صورتی	Magenta
Yellow	زرد	Yellow
White	سفید	White
Gray	خاکستری	Gray

## خودآزمایی فصل دوھ

- ۱- قالب کلی یک برنامه به زبان C# چگونه است؟
- ۲- در هر برنامه به زبان C#، دست کم یک ..... تعریف می شود که در داخل آن، یک متند به نام ..... وجود دارد که اجرای برنامه از آن نقطه آغاز می گردد.
- ۳- در زبان C#، انتهای یک دستور با چه علامتی مشخص می شود؟
- ۴- تحقیق کنید که در زبان های C، C و Java علامت پایان دستور چیست؟
- ۵- علامت توضیحات در زبان C# کدام است؟
- ۶- روش قراردادی پاسکال برای نوشتن نام یک کلاس چیست؟
- ۷- برای نشان دادن یک پیام یا یک عبارت بر روی صفحه نمایش از چه دستوری استفاده می کنید؟
- ۸- تفاوت بین دو متده WriteLine() و Write() را بیان کنید.
- ۹- با استفاده از متده WriteLine()، دستور مناسبی برای ایجاد یک سطر خالی بین نوشه ها در خروجی بنویسید.

۱۰- مثالی از کاربرد دستور using بیاورید.

۱۱- برای هر یک از خواسته های زیر، دستور یا دستورات مربوطه را بنویسید :

الف) نام شما در خروجی بارنگ زرد نمایش داده شود.

ب) صفحه کنسول نمایش را پاک کنید.

پ) اجرای برنامه تا فشردن یک کلید متوقف شود.

ت) صفحه کنسول نمایش با رنگ آبی پاک شود.

## تمرینات برنامه نویسی فصل دوھ

- ۱- برنامه ای بنویسید که نام، نام خانوادگی و نام مدرسه شما را روی صفحه نمایش دهد.
- ۲- برنامه تمرین ۱ را تغییری دهید تا اطراف نام شما یک کادر مانند شکل زیر رسم نماید مثلاً اگر اسم شما محمد است. روی صفحه چنین نمایش داده شود :

MOHAMMAD

۳- برنامه‌ای بنویسید که حرف انگلیسی نام شما را با استفاده از علامت \* نشان دهد. مثلاً اگر اسم شما حمید است حرف انگلیسی H را با استفاده از علامت \* به صورت زیر نشان دهد.

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

یادآوری : هنگامی که می‌خواهید برنامه‌ای بنویسید سعی کنید از برنامه‌ای که قبلاً ذخیره کرده‌اید استفاده کنید. برنامه را در یک ویرایشگر باز و سپس تغییرات لازم را ایجاد و ذخیره نمایید.

۴- برنامه‌ای بنویسید که تعداد روزهای عمر شما را با استفاده از ضرب سن شما در عدد ۳۶۵ محاسبه روی صفحه نشان دهد. مثلاً اگر سن شما ۱۶ است عبارت  $16 \times 365$  را محاسبه و نمایش دهد.

```
System.Console.WriteLine(16 * 365);
```

۵- با توجه به تمرین ۴ تعداد سال‌های کمیسه را نیز حساب کنید و برنامه را تغییر دهید.  
(راهنمایی : برای محاسبه تعداد سال کمیسه، خارج قسمت سن بر عدد ۴ را حساب کنید.)  
۶- با اجرای برنامه زیر چه عبارتی بر روی صفحه نشان داده می‌شود؟ به نظر شما چه ارتباطی بین علائم { } و { } و { } با اعداد ۱۸ و ۱۵ و ۱۵ وجود دارد؟

```
Class SomeCalculations
```

```
{
    Static void Main()
    {
        System.Console.WriteLine("{0} {1} {2}", 18, 15, 18);
    }
}
```

۷- تمرین زیر به زبان انگلیسی است. آن را با دقت بخوانید و برنامه خواسته شده را بنویسید.  
برای زیبایی خروجی از جعبه رنگ ConsoleColor استفاده نمایید.

Write a program that prints a face, using text characters, hopefully better looking than this one.

```
|||||
| o o |
(| ^ |)
|_ __|
```

## واژگان و اصطلاحات انگلیسی فصل دو

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	C Sharp Compiler	
۲	Calculate	
۳	Calculation	
۴	Character	
۵	Class	
۶	Command Prompt	
۷	Comment	
۸	Common Language Runtime	
۹	Console	
۱۰	Display a greeting message	
۱۱	Expected	
۱۲	Main	
۱۳	Method	
۱۴	Namespace	
۱۵	Object Oriented Language	
۱۶	Open Source	
۱۷	Pascal case	
۱۸	Path	
۱۹	Press any key to exit	
۲۰	Reserved words	
۲۱	String	
۲۲	Text Editor	



## فصل سوم

### آشنایی با ویژوال استودیو

در فصل قبلی با ساختار کلی یک برنامه C# آشنا شدید و برنامه‌های ساده را با استفاده از یک ویرایشگر، نوشته و ذخیره کردیم. سپس فایل برنامه را از طریق پنجره فرمان<sup>۱</sup> با استفاده از مترجم زبان C# (برنامه csc.exe) ترجمه کرده و فایل اجرایی EXE. تولید شد. در آخر، فایل EXE را اجرا کردیم. این روش برای نوشتند و تولید برنامه‌های کوچک، خوب است اما اگر برنامه‌ای که می‌نویسیم بزرگ باشد، به کارگیری این روش کمی دشوار و زمان برخواهد بود. به خصوص عیب‌یابی و اشکال زدایی آن بسیار وقت گیر است. در این فصل با برنامه‌ای آشنا خواهیم شد که همه ابزارها و لوازم مورد نیاز یک برنامه نویس در آن گردآوری شده است و کار برنامه‌نویسی را آسان می‌کند.

#### پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند :

- ۱- IDE را تعریف کند و برای آن مثال بیاورد.
- ۲- مزایای استفاده از IDE و کاربرد VS را بیان نماید.
- ۳- یک برنامه جدید در محیط VS از نوع کنسول ایجاد کند، آن را ذخیره، ترجمه و اجرا نماید.

### ۱-۳- آشنایی با ویژوال استودیو<sup>۲</sup>

تایپ برنامه در یک ویرایشگر، ورود به پنجره فرمان، ترجمه کردن، عیب‌یابی و اشکال زدایی برنامه، همگی عملیاتی بودند که در فصل قبلی انجام دادید و کمی وقت گیر و پر زحمت بود، چون از یک محیط باید وارد محیط دیگری می‌شدید. برای این که راحت‌تر بتوانیم برنامه‌نویسی کنیم لازم است

<sup>۱</sup> – Command Prompt

<sup>۲</sup> – Visual Studio

از محیط استفاده کنیم که همه ابزارها و لوازم مورد نیاز برنامه‌نویسی در آن گردآوری و متمرکز شده باشد. به چنین محیط برنامه‌نویسی که در آن می‌توان تمام مراحل برنامه‌نویسی، ترجمه، اشکال یابی و سرانجام اجرا را انجام داد،<sup>۱</sup> IDE گفته می‌شود که به معنای محیط تولید برنامه متمرکز می‌باشد. یعنی همه ابزارها و امکانات لازم برای تولید برنامه در یک جا گردآوری شده است.

شرکت مایکروسافت یک IDE بسیار پیشرفته برای برنامه‌نویسی فراهم کرده است که با کمک آن می‌توان راحت‌تر برنامه بنویسیم و ترجمه و اجرا کنیم. نام این نرم افزار ویژوال استودیو است. ویژوال استودیو یک محیط برنامه‌نویسی بسیار قوی برای تولید برنامه‌های کاربردی تحت ویندوز و بر پایه .Net Framework می‌باشد. ویژوال استودیو از چند زبان برنامه‌نویسی نظیر C#، C و VB پشتیبانی می‌کند. در این محیط علاوه بر تایپ برنامه، می‌توان برنامه را ترجمه، عیب‌یابی و سرانجام اجرا کرد. لایه نرم افزاری.NET Framework 4.5. به همراه ویژوال استودیو ۲۰۱۲ عرضه شد است. در حال حاضر<sup>۲</sup> VS 2013 عرضه شده است. خوبی‌خانه شرکت مایکروسافت همراه با عرضه ویژوال استودیوی تجاری، یک نسخه رایگان از این نرم افزار را تحت عنوان ویژوال استودیو اکسپرس<sup>۳</sup> نیز عرضه می‌کند که می‌توانید آن را از روی سایت شرکت مایکروسافت<sup>۴</sup> دانلود نمایید.



شکل ۱-۳- اجرای ویژوال استودیو اکسپرس ۲۰۱۲

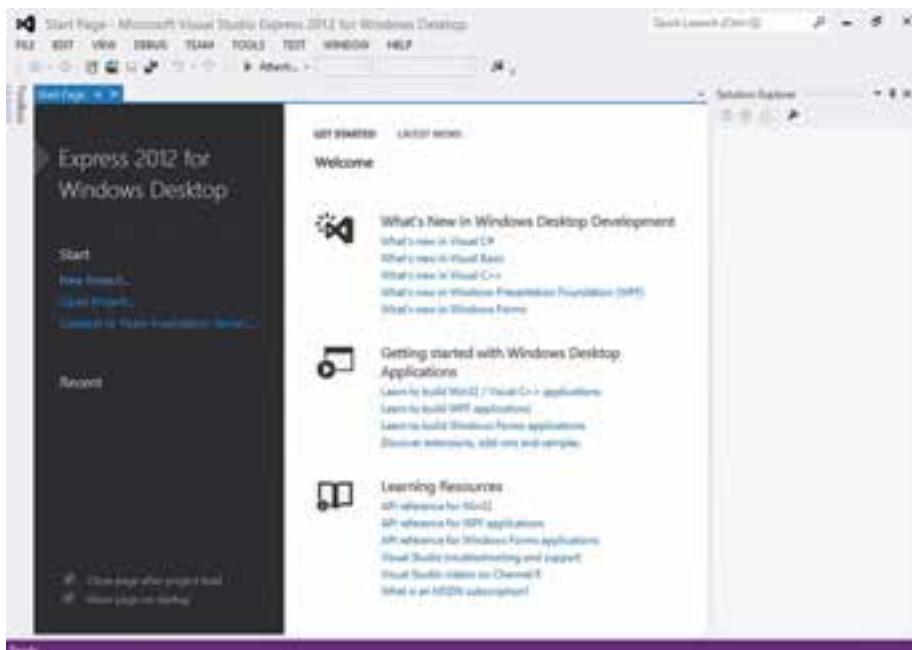
۱- Integrated Development Environment

۲- اردیبهشت ۹۳

۳- Visual Studio Express

۴- <http://microsoft.com/vs/visual-studio-downloads>

ما در این کتاب با 2012 Visual Studio Express کار می کنیم و از این به بعد در سرتاسر کتاب از مخفف VS برای بیان کلمه ویژوال استودیو استفاده می کنیم. اگر نرم افزار 2012 VS را در اختیار ندارید می توانید از نسخه های قدیمی تر نیز استفاده کنید فقط شکل ظاهری آن ممکن است کمی متفاوت باشد.



شکل ۳-۲- پنجره آغازین ویژوال استودیو اکسپرس ۲۰۱۲

### ۳-۲- ایجاد یک پروژه جدید در ویژوال استودیو

با اجرای برنامه VS، صفحه شروع (Start Page) مطابق با شکل ۳-۳ ظاهر می شود، از این صفحه می توانید یک پروژه یا برنامه جدید (New Project...) بسازید و یا برنامه های قبلی خود را باز (...) Open Project... کنید.

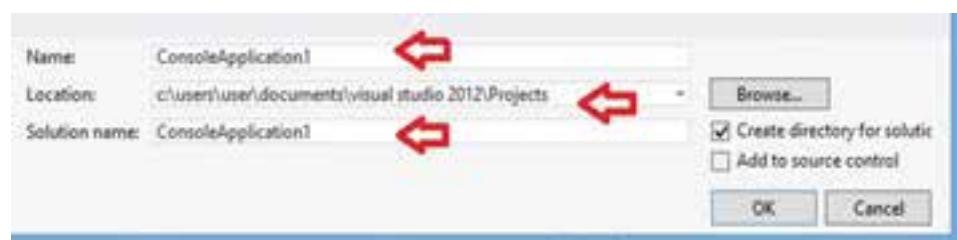
برای ایجاد یک برنامه جدید به زبان C# که در محیط کنسول کار می کند ابتدا روی گزینه (New Project...) در لیست سمت چپ کلیک کنید. پس از ظاهر شدن پنجره، Visual C# را انتخاب کنید (شکل ۳-۳- مرحله ۱) و در وسط صفحه بر روی گزینه Console Application کلیک کنید (شکل ۳-۳ مرحله ۲).



شکل ۳-۳- پنجره پروژه جدید

در قسمت Name باید نام پروژه را تایپ کنید که به صورت پیش فرض ConsoleAppliaction1 برای آن در نظر گرفته شده است. مناسب است نامی مطابق با هدف برنامه ای که می نویسید انتخاب کنید چون این نام به هیچ وجه گویا و روشن نیست.

همان طور که در شکل ۳-۴ مشاهده می کنید، در قسمت Location مسیر ذخیره پروژه نشان داده شده است. هنگامی که VS را نصب می کنید در داخل My Documents یک پوشه به نام Projects ساخته می شود که در داخل آن نیز یک پوشه فرعی به نام ConsoleApplication1 ایجاد می شود. در داخل این پوشه، پروژه های شما به طور پیش فرض ذخیره می شود. اگر بخواهید می توانید پروژه خود را در مسیر دیگری ذخیره کنید. مسیر دلخواه خود را با کلیک بر روی دکمه Browse مشخص کنید.

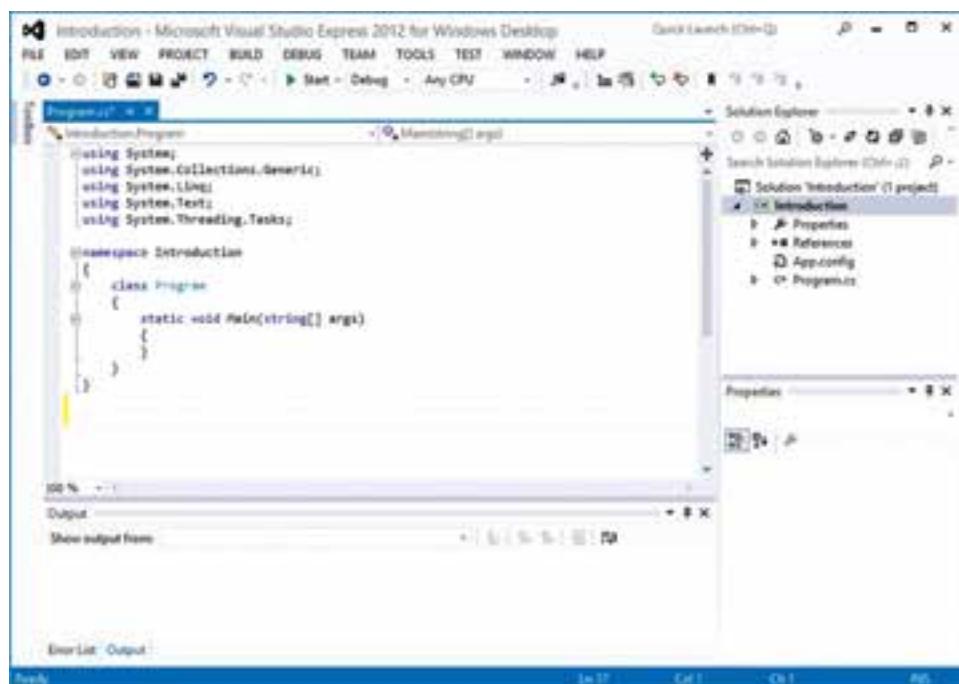


شکل ۳-۴- تعیین نام برای یک پروژه جدید

بعد از مشخص کردن محل پروژه، باید نامی برای Solution name در نظر بگیرید که نام پوشه‌ای را معین می‌کند که فایل‌های مربوط به یک چند پروژه در آن نگهداری می‌شود. معمولاً نام Solution با نام پروژه یکسان انتخاب می‌شود.

در حال حاضر چون هدف ما آشنایی با VS است نام Introduction را در قسمت Name می‌نویسیم و این نام برای Solution نیز انتخاب می‌شود. حال پس از تعیین نام‌ها، بر روی کلید کلیک کنید. در این صورت یک پوشه با نام مذکور در مسیر پیش فرض ساخته می‌شود. درون این پوشه چندین فایل و یک پوشه فرعی با نامی که برای پروژه انتخاب کردید (در مثال ما Introduction) ساخته می‌شود.

پس از چند لحظه مشاهده خواهید کرد که در IDE پنجره‌های مختلفی نشان داده می‌شود و در یکی از پنجره‌ها صورت کلی یک برنامه C# به صورت آماده مانند شکل ۳-۵ نشان داده می‌شود.

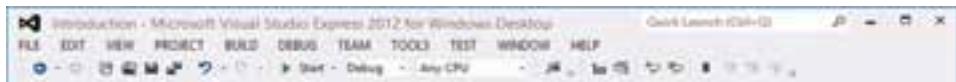


شکل ۳-۵- محیط تولید برنامه (IDE) ویزوال استودیو

قبل از این که به برنامه نویسی پردازیم ابتدا به صورت مختصر با بخش‌های مختلف محیط IDE آشنا می‌شویم.

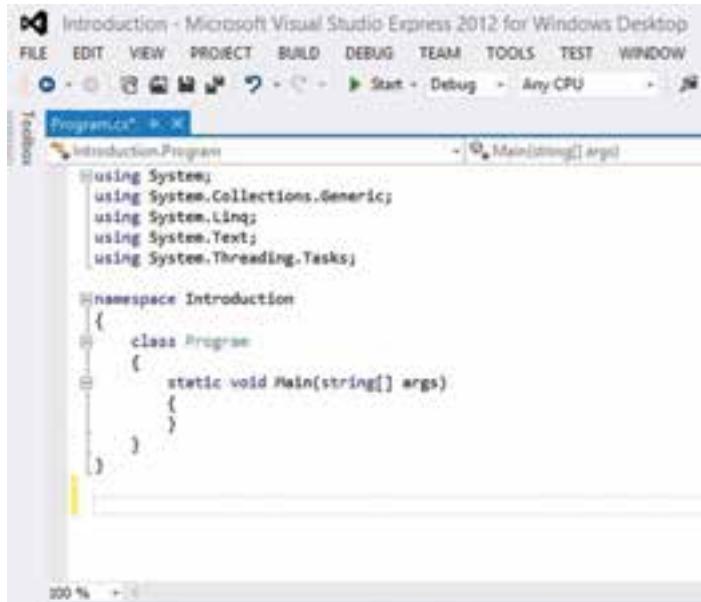
### **۳-۳- معرفی بخش‌های اصلی ویژوال استودیو**

**۳-۳-۱** نوار منو و نوار ابزار<sup>۱</sup>: مانند بیشتر نرم افزارها، در قسمت بالای صفحه، منوهای مختلف VS و در زیر آن ابزارهای پر کاربرد مطابق با شکل ۳-۶ دیده می شود. به تدریج با این منوها و بزرگ آشنا شنایم می شویم.



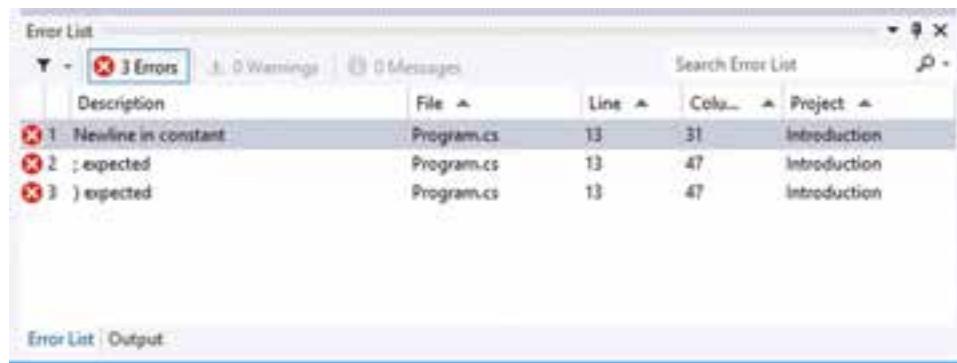
شكل ٦-٣- نوار منو و نوار ابزار محیط ویژوال استودیو

**۲-۳-۳-۲** پنجره ویرایشگر برنامه : در شکل ۷-۳ پنجره ویرایشگر برنامه نشان داده شده است. در این پنجره متن برنامه نگهداری می شود و می توانید چندین برنامه را هم زمان به صورت باز داشته باشید. در پنجره نویسی، این پنجره کاربرد زیاد دارد.



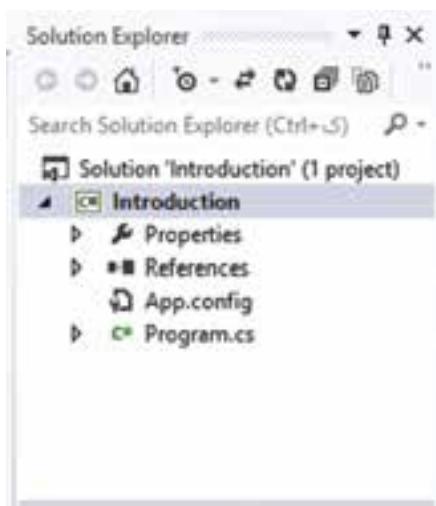
شکل ۷-۳- پنجره ویرایشگر برنامه

**۳-۳-۳ پنجره لیست خطاها (Error List) :** در صورتی که برنامه اشکال تایی یا ساختاری داشته باشد، خطاها و اشکالات برنامه در این پنجره مانند شکل ۳-۸ لیست می‌شوند. پس از ترجمه برنامه باید به این پنجره نگاه کنیم تا خطاها را احتمالی برنامه را متوجه شویم.



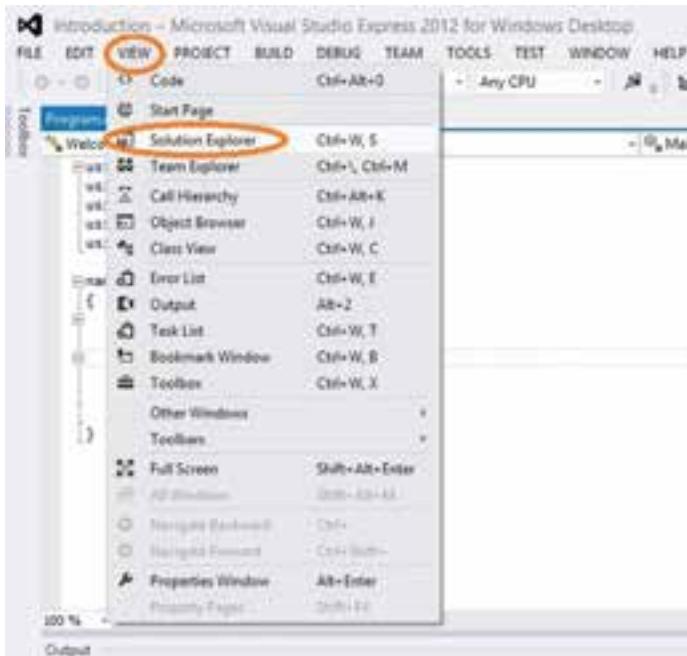
شکل ۳-۸- پنجره لیست خطاها

**۳-۳-۴ پنجره Solution Explorer :** سمت راست صفحه، پنجره ای قرار دارد که ساختار پروژه و تمام فایل‌های موجود در آن را نشان می‌دهد. اگر پروژه‌ای باز نباشد، محتوای این پنجره خالی است. ما نام این پنجره را مرورگر پروژه می‌نامیم و به وسیله آن به تمام اجزای پروژه دسترسی داریم.



شکل ۳-۹- پنجره Solution Explorer

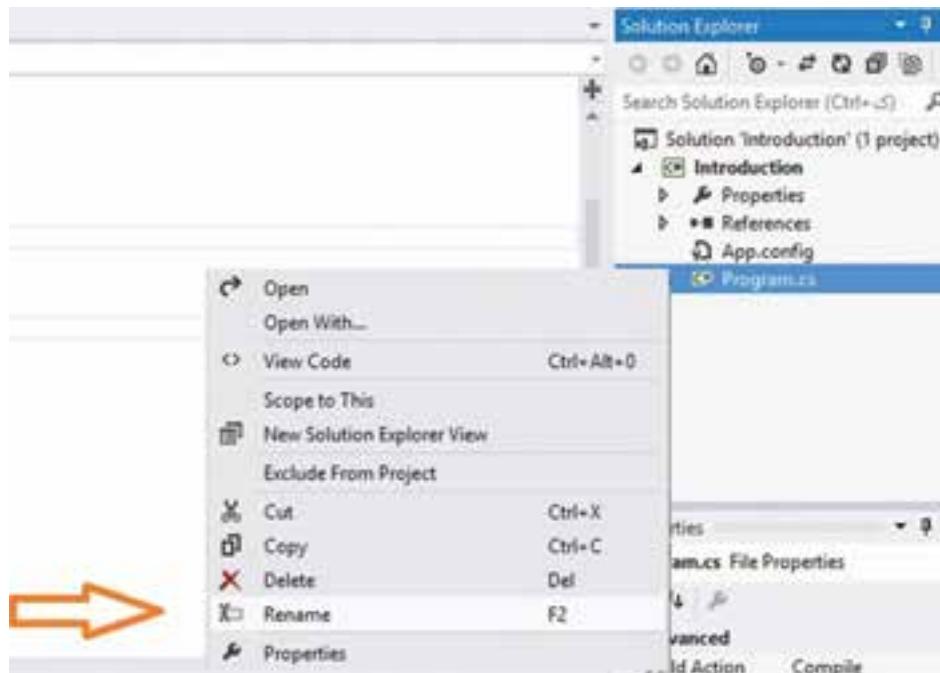
اگر پنجره مروارگر پروژه باز نیست، و آن را مشاهده نمی‌کنید از منوی View در بالای صفحه استفاده کنید. در این منو، نام تمام پنجره‌ها در شکل ۳-۱۰ مشاهده می‌شود، روی گزینه Solution Explorer کلیک کنید تا این پنجره دیده شود.



شکل ۳-۱۰- ظاهر کردن پنجره Solution Explorer

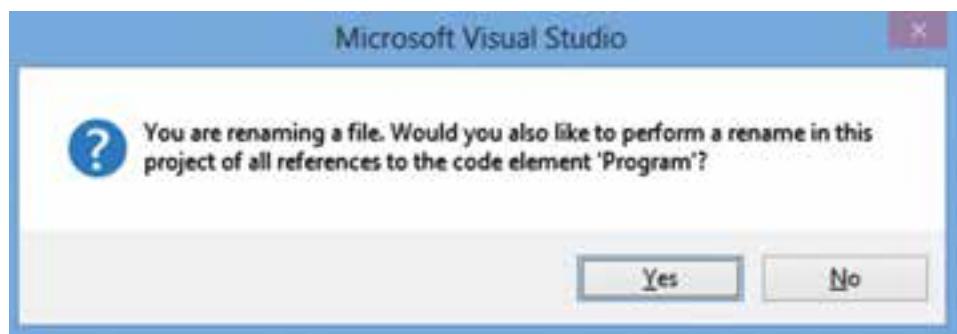
#### ۴-۳- برنامه‌نویسی در محیط ویژوال استودیو

بعد از ایجاد یک پروژه جدید و وارد شدن به محیط برنامه‌نویسی (IDE)، اگر به پنجره کاوشگر Solution توجه کنید، داخل شاخه پروژه Introduction، یک فایل به نام program.cs وجود دارد. (پسوند cs). نشان‌دهنده برنامه به زبان C Sharp می‌باشد) این فایل، فایل متن برنامه است و به طور خودکار تولید شده است. محتوای این فایل در پنجره ویرایشگر برنامه نشان داده شده است. نام این فایل را می‌توانید مطابق با عملکرد برنامه تغییر دهید و یک نام مناسب برای آن انتخاب کنید که با دیدن نام فایل به عملکرد آن بی‌بیرید. در این مثال می‌خواهیم برنامه‌ای بنویسیم که یک پیام خوشامدگویی مانند فصل قبل نمایش دهد بنابراین برای تغییر نام فایل Program.cs روی آن کلیک راست کرده و نام دلخواه خود مثلاً Welcome.cs را وارد می‌کنیم (شکل ۱۱-۳).



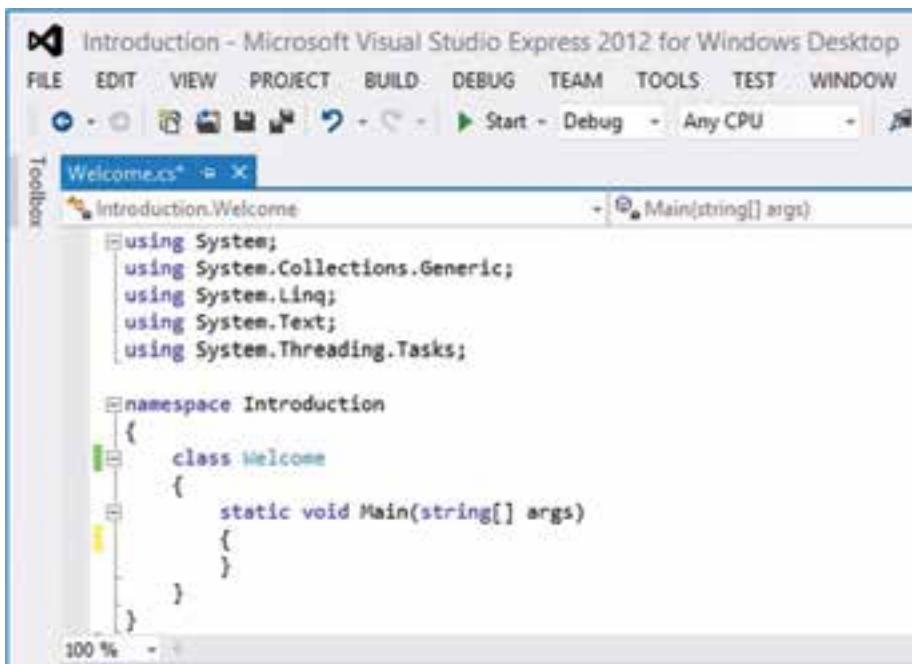
شکل ۱۱-۳- تغییر نام برنامه

پس از تغییر نام و زدن کلید Enter پنجره‌ای مانند شکل ۱۱-۲ باز می‌شود و سؤال می‌شود «آیا این تغییر نام در تمام مکان‌هایی که به این نام رجوع می‌شود نیز اعمال شود؟» اگر پاسخ مثبت یعنی Yes را انتخاب کنید مشاهده خواهید کرد که در متن برنامه، نام کلاس که قبلاً Program بود به Welcome تغییر نام می‌دهد.



شکل ۱۱-۳- تغییر نام در تمام قسمت‌هایی که به Program رجوع می‌کند

حال به پنجره کد، باز می‌گردیم، همان طور که در شکل ۱۳-۳ ملاحظه می‌کنید ساختار کلی یک برنامه به وسیله VS برای شما آماده می‌شود که تعدادی دستور در آن موجود است.



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

namespace Introduction
{
    class Welcome
    {
        static void Main(string[] args)
        {
        }
    }
}
```

شکل ۱۳-۳- پنجره ویرایشگر برنامه جدید

در چند خط بالای برنامه، راهنمایی‌هایی برای مترجم یعنی دستورات using نوشته شده است و جلوی هر کدام، یک فضای نامی ذکر شده است. فضای نامی System برای شما آشنا است زیرا در فصل قبل دستور using System را در بالای برنامه نوشتیم و این کار سبب شد که استفاده و نوشتن متدهای مربوط به Console در برنامه ساده‌تر شود. در حال حاضر می‌توانید دستورات دیگر using را پاک کنید چون فعلًاً به کلاسی غیر از Console نیاز نداریم.

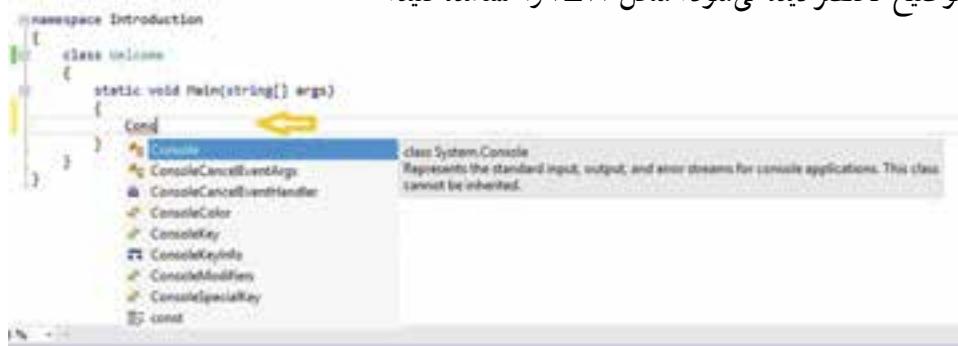
بعد از دستورات using، دستور namespace به همراه نام پروژه (Introduction) نوشته شده است و علامت‌های آکولاد باز و بسته کل برنامه را دربرگرفته است. با دستور namespace یک فضای نامی جدید تعریف می‌شود که برای سازماندهی و دسته‌بندی پروژه‌های بزرگ مورد استفاده دارد. فعلًاً به آن کار نداریم ولی لازم نیست آن را پاک کنید، بنابراین تغییری روی آن انجام ندهید. در داخل این فضای نامی، دستور class و سپس در داخل آن متاد Main نوشته شده است. نام کلاس Welcome است مگر این که در مرحله قبل نام برنامه را تغییر نداده باشید که در این صورت

نام کلاس Program است. متد Main() نیز در داخل کلاس دیده می شود ولی داخل پرانتزهای آن عبارت string[] args نوشته شده است. می توانید این عبارت را از داخل پرانتزها پاک کنید تا برنامه ساده‌تر شود همان‌طور که در فصل قبل داخل پرانتزها خالی بود.

داخل متد Main، بین آکو لادها فضا ایجاد کنید (این کار را با کلیک کردن در بین دو علامت آکولاد و سپس با زدن کلید Enter انجام دهید) تا بتوانید دستورات مورد نظر خود را بنویسید. مثلاً دستورات زیر را بنویسید :

```
Console.WriteLine("Hello World");
Console.WriteLine("Welcome to C#");
```

هنگامی که مشغول تایپ برنامه هستید باید تفاوت قابل ملاحظه‌ای را با روش فصل قبل که برنامه را در محیط Notepad ویندوز و یا در برنامه NotePad می‌نوشته احساس کنید. اولاً کلمات با توجه به نوع آنها رنگی نوشته شده‌اند، مثلاً کلمات کلیدی با رنگ آبی نشان داده شده‌اند و ضمناً در هنگام تایپ برنامه به محض نوشتن یک حرف کلمه Console لیستی از کلمات مشابه نمایش داده می‌شود. با تایپ چند حرف دیگر، کلمه Console در لیست نشان داده می‌شود و در کنار آن یک توضیح مختصر دیده می‌شود. شکل ۳-۱۴ را مشاهده کنید.



شکل ۳-۱۴- کیک در تایپ دستورات در محیط ویژوال استودیو

هرگاه کلمه مورد نظر شما در لیست دیده شد و نوار آبی رنگ روی آن قرار گرفت می‌توانید تایپ آن کلمه را کنید و کلید Enter را بزنید. در این صورت کلمه به طور کامل تایپ می‌شود و می‌توانید دنباله دستور را بنویسید. این کار باعث می‌شود سرعت شما در تایپ برنامه به طور چشمگیری افزایش یابد. اگر در تایپ یک دستور غلط املایی داشته باشید و یا قوانین برنامه‌نویسی زبان C# را رعایت نکنید در این صورت، یک خط قرمز رنگ، زیر کلمه یا مکانی که در آن اشتباه وجود دارد کشیده می‌شود و به شما یادآوری می‌کند که در آن مکان یک خطأ وجود دارد و شما باید آن را برطرف کنید.

A screenshot of the Visual Studio IDE. The code editor shows the following C# code:

```
namespace Introduction
{
    class Welcome
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

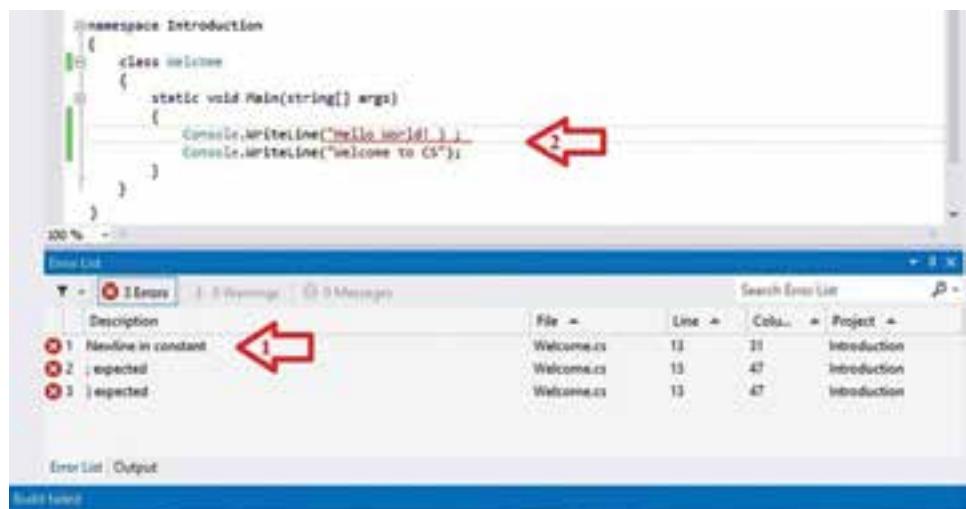
A red arrow points upwards from the code editor towards the status bar at the bottom of the window.

شکل ۳-۱۵- نشان دادن خطأ در تایپ برنامه در محیط ویژوال استودیو

مثلاً در شکل ۳-۱۵ مشاهده می کنید که یک خط قرمز در زیر عبارت Hello World! کشیده است. با کمی دقیق متوجه می شوید که علامت نقل قول انتهای پیام فراموش شده است! چون پیام ها به عنوان یک رشته باید بین علامت های نقل قول قرار داشته باشند.

### ۳-۵- ترجمه برنامه

بعد از نوشتن دستورات بالا، برای ترجمة برنامه، کلید F6 را بزنید. اگر برنامه خطأ داشته باشد، خطاهای در پنجه Error List دیده می شود. برای مثال شکل ۳-۱۶ خطاهای برنامه را در حالتی نشان می دهد که فراموش کرداید انتهای رشته را با علامت " مشخص نمایید. در این شکل، پنجه Error List چند خطأ را نشان می دهد؟



شکل ۳-۱۶- مرحله رفتن به محل خطأ در برنامه

شاید این مسئله تا حدودی تعجبآور باشد که شما در تایپ برنامه، یک علامت نقل قول را فراموش کرده‌اید، اما در پنجره Error List، سه خطأ گزارش می‌شود. بنابراین در ترجمه یک برنامه، انتظار گزارش تعداد خطای زیاد را داشته باشید. در حالت بروز خطا، باید با خواندن توضیح خطا و شماره خط برنامه که گزارش می‌شود، اقدام به رفع اشکال برنامه کنید. همواره از خطای اول شروع کنید، بنابراین در پنجره Error List، روی خطای اول دابل کلیک کنید (شکل ۳-۱۶ مرحله ۱) تا به طور مستقیم به پنجره ویرایشگر برنامه و محل خطای مزبور هدایت شوید (شکل ۳-۱۶ مرحله ۲). در این صورت اقدام به رفع اشکال کنید و سپس برنامه را دوباره ترجمه کنید. اگر خطای گزارش شد باز سراغ خطای اول بروید و برنامه را تصحیح کنید.

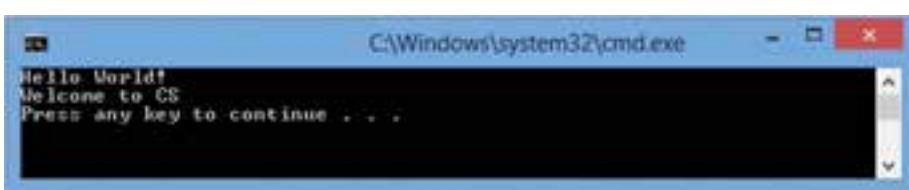
پس از رفع اشکالات برنامه اگر دوباره عمل ترجمه را انجام دهید، در پنجره خروجی (Output) (Shکل ۳-۱۷) موفق را مانند شکل ۳-۱۷ مشاهده خواهید کرد.



شکل ۳-۱۷- پنجره خروجی در حالت بدون خطا

### ۶-۳- اجرای برنامه

بعد از ترجمه صحیح و بدون خطا، می‌توانید برنامه را اجرا کنید. برای اجرای برنامه از دکمه **Start** = بر روی نوار ابزار یا از ترکیب کلیدی **Ctrl F5** استفاده کنید. با اجرای برنامه صفحه کنسول با سه پیام مانند شکل ۳-۱۸ دیده می‌شود. دو پیام خوش آمدگویی، همان‌هایی هستند که شما نوشته‌اید. اما پیام سوم ... Press any key to continue... به طور خودکار به وسیله VS در صفحه کنسول اضافه می‌شود تا صفحه کنسول بسته نشود و شما فرصت دیدن پیام‌ها را داشته باشید. با زدن یک کلید دلخواه، پنجره کنسول بسته می‌شود زیرا دستور دیگری در برنامه وجود ندارد.



شکل ۳-۱۸- نتیجه اجرای برنامه در صفحه کنسول

## کار در کارگاه: آشنایی با محیط VS (ویژوال استودیو)

در این قسمت، مراحل ایجاد یک پروژه، ترجمه و اجرای برنامه را تجربه خواهید کرد.

۱- برنامه VS را نصب کنید. (از پیوست شماره ۱ در انتهای کتاب استفاده کنید.)

۲- برنامه VS را اجرا کنید.

۳- در صفحه آغازین، یک پروژه جدید بسازید. زبان `C#` را انتخاب و یک برنامه کاربردی

جدید Console Application ایجاد کنید. (به توضیحات ابتدای این فصل مراجعه کنید.)

۴- یک نام مناسب برای پروژه انتخاب کنید. مثلاً Introduction

۵- با محیط IDE ویژوال استودیو آشنا شوید و پنجره هایی که در این فصل معرفی شده بودند،

مانند پنجره Solution Explorer را شناسایی کنید.

۶- در محیط IDE، پنجره ویرایشگر برنامه را شناسایی کنید. ساختار یک برنامه باید دیده شود.

۷- وارد پنجره ویرایشگر برنامه شوید و دو خط زیر را مطابق با آنچه در درس گفته شد در داخل

متد Main اضافه کنید.

```
Console.WriteLine("Hello World");
```

```
Console.WriteLine("Welcome to C#");
```

۸- برنامه را ترجمه و سپس اجرا کنید.

۹- علامت نقطه ویرگول را از انتهای یکی از دستورات حذف کنید و دوباره برنامه را ترجمه

کنید و به پنجره لیست خطاهای دقت کنید. چند خطأ گزارش شد؟

۱۰- در پنجره لیست خطاهای، روی توضیح اولین خطأ دوبار کلیک کنید تا به ویرایشگر برنامه و محل خطأ بروید و اشکال را برطرف کنید.

۱۱- برنامه را ترجمه و اجرا کنید.

۱۲- علامت نقل قول قبل از حرف H در پیام Hello World! را حذف کنید و دوباره برنامه

را ترجمه کنید و به پنجره لیست خطاهای دقت کنید. چند خطأ گزارش شد؟

۱۳- در پنجره لیست خطاهای، روی توضیح اولین خطأ دوبار کلیک کنید تا به ویرایشگر برنامه و محل خطأ بروید و اشکال را برطرف کنید و سپس برنامه را ترجمه و اجرا کنید.

۱۴- در داخل متد Main()، دستورات زیر را اضافه کنید.

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

```

{
    Console.BackgroundColor = ConsoleColor.Blue;
    Console.Clear();
    Console.WriteLine("Hello World!");
    Console.WriteLine("Welcome to CS");
    Console.ReadKey();
}

```

۱۵— متدها از کلاس کنسول برای ایجاد یک صدا یا صوت در برنامه استفاده می‌شود این متدها در پایان برنامه به صورت زیر اضافه کنید تا یک صوت به مدت یک ثانیه ایجاد کند.

```

Console.WriteLine("Welcome to CS");
Console.Beep();
Console.ReadKey();

```

اگر بخواهید مدت زمان نواختن صدا و همچنین فرکانس (زیر و بم) صدا را تغییر دهید، کافی است در داخل پرانتز متدها، اعدادی را به ترتیب زیر بنویسید :

(فرکانس بر حسب هertz ، مدت زمان بر حسب میلی ثانیه )  
مثالاً برای نواختن یک صدا با فرکانس ۵۰۰ هertz به مدت ۲ ثانیه ( ۲۰۰۰ میلی ثانیه ) دستور زیر را می‌نویسیم :

```
Console.Beep(500 , 2000 );
```

توجه داشته باشید که مقدار فرکانس را باید در محدوده مناسبی بنویسید زیرا گوش انسان فقط قادر است اصواتی با فرکانس حدود ۲۰۰ تا ۱۰۰۰۰ هertz را خوب بشنود. صدای بم دارای فرکانس کم و صدای ریز فرکانس بالا دارند.

۱۶— با قرار دادن اعداد مختلف به عنوان فرکانس در متدها، محدوده شنوازی گوش خود را آزمایش کنید.

## فودآزمایی فصل سوم

- ۱- نرم افزار Visual Studio یک ..... بسیار پیشرفته برای برنامه‌نویسی به زبان C# است.
- ۲- IDE مخفف کلمات ..... است و یانگر یک محیط برنامه‌نویسی است که می‌توان در آن برنامه را تایپ کرد.
- ۳- در محیط VS علاوه بر تایپ برنامه، می‌توان برنامه را .....، عیب‌یابی و سرانجام ..... کرد.
- ۴- در محیط VS پس از ترجمه برنامه، خطاهای احتمالی آن در کدام پنجره دیده می‌شوند؟
- ۵- در پنجره Solution Explorer محیط VS، چه اطلاعاتی نشان داده می‌شود؟
- ۶- برنامه‌نویسی در محیط ویژوال استودیو چه برتری نسبت به استفاده از یک ویرایشگر دارد؟
- ۷- ترجمه برنامه در محیط VS چه برتری نسبت به پنجره فرمان واستفاده از متترجم CSC.EXE دارد؟
- ۸- از چه متدی برای ایجاد یک صوت استفاده می‌کنید؟ مدت زمان و فرکانس نواختن صوت را چگونه تعیین می‌کنید؟

## تمرینات برنامه‌نویسی فصل سوم

(این تمرینات در محیط ویژوال استودیو انجام شود)

- ۱- برنامه‌ای بنویسید که نام و نام خانوادگی و تاریخ تولد شما را به زبان فارسی نمایش دهد.
- ۲- برنامه‌ای بنویسید که ۳ صوت با فرکانس‌های ۵۰۰ و ۶۰۰ و ۷۰۰ هرتز را هر یک به مدت نیم ثانیه پشت سرهم ایجاد کند.
- ۳- برنامه‌ای بنویسید که اطلاعات هر سطر جدول زیر را در یک خط نمایش دهد.

نام برنامه	آدرس سایت
NotePad++	<a href="http://notepad-plus-plus.org">http://notepad-plus-plus.org</a>
Visual Studio	<a href="http://microsoft.com/visualstudio/downloads">http://microsoft.com/visualstudio/downloads</a>
NET Framework 3.5	<a href="http://www.microsoft.com/en-us/download/details.aspx?id=22">http://www.microsoft.com/en-us/download/details.aspx?id=22</a>
DirectX	<a href="http://www.microsoft.com/en-us/download/details.aspx?id=35">http://www.microsoft.com/en-us/download/details.aspx?id=35</a>

- ۴- با استفاده از متدهای تغییر رنگ زمینه و رنگ قلم خروجی برنامه تمرین شماره ۳ را به صورت دلخواه رنگی کنید.

## واژگان و اصطلاحات انگلیسی فصل سوم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Beep	
۲	Command Prompt	
۳	Console Application	
۴	Integrated Development Environment	
۵	Introduction	
۶	Location	
۷	New Project	
۸	Open Project	
۹	Solution Name	
۱۰	Start Page	
۱۱	Toolbar	
۱۲	Visual Studio	
۱۳	Visual Studio Express	

## فصل چهارم

### آشنایی با انواع داده‌ها و متغیرها

برنامه‌هایی که تاکنون نوشته‌ایم، به نشان دادن یک پیام یا حاصل یک عبارت بر روی صفحه نمایش محدود می‌شد، اما در برنامه‌های کاربردی با داده‌ها و مقادیر مختلف سروکار داریم و باید بر روی آن عملیاتی را انجام دهیم. بعضی از این مقادیر مانند تاریخ تولد یک شخص یا نمره یک دانش‌آموز از قبل مشخص نیستند. مقدار این نوع داده‌ها باید در هنگام اجرای برنامه، ابتدا از کاربر دریافت شوند و در مکانی از حافظه کامپیوتر نگهداری شوند و در ادامه برنامه و در جریان پردازش، مورد استفاده قرار گیرند. چه حافظه‌ای برای نگهداری داده‌ها در هنگام پردازش مناسب است؟

در این فصل با متغیرها آشنا می‌شویم که برای نگهداری موقتی داده‌ها در برنامه مورد استفاده قرار می‌گیرند. همچنین برای نگهداری اطلاعات و نمایش آنها بر روی صفحه نمایش از متغیرها استفاده می‌کنیم.

#### پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند :

- ۱- متغیر را تعریف کند و انواع متغیر را در برنامه‌های خود به کار بیندد.
- ۲- انواع داده‌ها را نام ببرد و تفاوت کاربرد هر یک را توضیح دهد.
- ۳- میزان حافظه و محدوده انواع داده‌ها را بیان کند.
- ۴- متغیرها را به طور صحیح در برنامه اعلان کند و آنها را مقداردهی نماید.
- ۵- شکل نمایش نقطه‌شناور را توضیح دهد و اعداد اعشاری را در این قالب بنویسد.
- ۶- از متدهای ReadLine() برای دریافت داده‌های یک برنامه از ورودی استفاده کند.
- ۷- بر روی رشته دریافتی از ورودی، تغییراتی داده و سپس نمایش دهد.
- ۸- از متدهای Parse() برای تبدیل یک رشته به یک عدد استفاده کند.

## ۱-۴- متغیر چیست؟

در هر کامپیوتر حافظه‌های مختلفی وجود دارد که هر یک برای انجام کار خاصی پیش‌بینی شده است. یک نوع از حافظه کامپیوتر که قادر است داده‌ها را نگهداری کند و به سرعت قابل دسترسی است، حافظه موقتی RAM<sup>۱</sup> است. از اطلاعات درون حافظه RAM، در هر لحظه می‌توان با اطلاع شد و یا در صورت لزوم محتویات آن را تغییر داد یا مقدار جدیدی را در آن ذخیره کرد.

با توجه به مطالب گفته شده، لازم است در یک برنامه، یک یا چند مکان (بایت) از حافظه RAM کامپیوتر برای نگهداری موقتی داده‌ها یا نتایج حاصل از پردازش مورد استفاده قرار گیرد. در زبان‌های برنامه‌نویسی به این مکان‌ها، متغیر<sup>۲</sup> گفته می‌شود زیرا می‌توان محتوای آنها را در طول اجرای برنامه تغییر داد.

### نکته

متغیر : مکانی از حافظه RAM کامپیوتر است که برای نگهداری موقتی داده‌ها یا اطلاعات استفاده می‌شود

متغیر را مانند یک ظرف در نظر بگیرید. در آشیخانه ظروف متعددی با شکل ظاهری، اندازه و جنس مختلفی وجود دارد که هر یک برای نگهداری یک نوع غذا یا مایعات استفاده می‌شود که گنجایش آن را داشته باشد. در یک برنامه نیز برای نگهداری هر یک از داده‌ها با توجه به نوع و بزرگی داده، باید از متغیر مناسبی استفاده کنیم که بتواند داده را نگهداری کند.

## ۲-۴- روش اعلان (تعريف) و ایجاد متغیرها

قبل از اینکه بتوانید مقداری را در یک متغیر ذخیره کنید باید متغیری را ایجاد کنید که قادر باشد آن مقدار را به درستی ذخیره نماید. در هنگام ایجاد متغیر باید نوع متغیر را مشخص نمایید. در زبان C# برای ایجاد و مشخص کردن نوع متغیر، از الگوی زیر استفاده می‌شود.

نام متغیر نوع داده

`int a;`

دستور زیر را در نظر بگیرید :

در این دستور متغیر `a` از نوع عدد صحیح اعلان می‌شود. کلمه `int` نوع متغیر را مشخص می‌کند که قادر است اعداد صحیح را در خود نگهداری کند و `a` نام متغیر است. نام متغیر به وسیله برنامه‌نویس انتخاب می‌شود که بهتر است نام و نوع آن مطابق با داده‌ای باشد که مقداردهی می‌شود.

### ۳-۴- نوع داده<sup>۱</sup> (نوع متغیر)

نوع متغیر به طور کلی ۳ ویژگی را مشخص می‌کند :

۱- گنجایش یا ظرفیت متغیر : مثلاً نوع `int` چهار بایت است.

۲- نوع اطلاعاتی که در متغیر می‌توان ذخیره کرد : مثلاً در متغیر نوع `int` فقط اعداد صحیح و بدون ممیز قابل نگهداری است.

۳- چه عملیاتی را می‌توان بر روی آن انجام داد : مثلاً عملیات ریاضی معمول را می‌توان روی اعداد نوع `int` انجام داد.

در زبان C# علاوه بر نوع داده `int`، انواع دیگری از داده‌ها نیز دسته بندی و گروه بندی شده‌اند و نحوه نمایش یا نگهداری آنها در حافظه و عملیاتی که می‌توان بر روی آنها انجام داد از قبل مشخص و تعریف شده است و برای هر دسته یا گروه از داده‌ها، یک نام انتخاب شده است که به آن نوع داده اولیه<sup>۲</sup> یا درون ساخته می‌گویند. جدول ۴-۱ انواع داده و مشخصات هر یک را نشان می‌دهد.

برای مثال در جدول ۴-۱ نوع داده `sbyte` را در نظر بگیرید. این نوع داده، اعداد صحیح و بدون ممیز در محدوده ۱۲۷ تا ۱۲۸ را شامل می‌شود که یک بایت حافظه را اشغال می‌کند و بر روی آنها می‌توان عملیات ریاضی را انجام داد. اگر در یک برنامه، متغیری از نوع `sbyte` را استفاده کنیم، قادر خواهیم بود به عنوان مثال عدد ۷۸ را در آن ذخیره کنیم. اما نمی‌توان عدد ۲۰۰ و یا عدد ۱/۵ را در آن نگهداری کرد. همچنین نوع داده `byte` اعداد صحیحی فقط در محدوده ۰ تا ۲۵۵ را شامل می‌شود که در یک بایت قرار می‌گیرد. در این نوع داده فقط اعداد مثبت یا بدون علامت<sup>۳</sup> قابل نمایش می‌باشند.

۱ - Data Type

۲ - Representation

۳ - Primitive Data Type or Built In Data Type

۴ - Unsigned numbers

## جدول ۱-۴

نوع داده	کاربرد نوع داده	مقدار حافظه (بايت)	کمترین مقدار	بیشترین مقدار
sbyte	اعداد صحیح		28	27
byte	اعداد صحیح مثبت		0	255
short	اعداد صحیح	2	32768	32767
ushort	اعداد صحیح مثبت	2	0	65535
int	اعداد صحیح	4	2 47483648	2 47483647
uint	اعداد صحیح مثبت	4	0	4294967295
long	اعداد صحیح	8	9223372036854778508	9223372036854778507
ulong	اعداد صحیح مثبت	8	0	844674407370955 6 5
float	اعداد اعشاری	4	$3.402823 \times 10^{38}$	$3.402823 \times 10^{38}$
double	اعداد اعشاری با دقت زیاد	8	.797693 3486232× $10^{308}$	.797693 3486232× $10^{308}$
decimal	اعداد صحیح بزرگ اعداد اعشاری با دقت بسیار زیاد	6	79228162514264337593543950335 $7.9 \times 10^{28}$	79228162514264337593543950335 $7.9 \times 10^{28}$
bool	مقدار منطقی		false	true
char	یک حرف یا علامت (کراکتر)	2	0	65535
string	رشته		-	-
object	آدرس یک داده		-	-

دستور ; **age** متغیری به نام **age** ایجاد می کند که این متغیر بسیار کوچک و به ظرفیت یک بایت است و می تواند یکی از اعداد صفر تا ۲۵۵ را در خود ذخیره کند.  
 اگر بخواهید چند متغیر از یک نوع را تعریف کنید کافی است بعد از ذکر نوع داده، نام متغیرها را با علامت ویرگول از یکدیگر جدا کنید. مثلاً برای تعریف دو متغیر برای نگهداری حداقل و حداکثر درجه حرارت از دستور زیر استفاده می کنیم :

**Sbyte minTemp , maxTemp ;**

هر نوع داده، مجموعه‌ای از مقادیر به همراه مجموعه‌ای از عملیات را مشخص می‌کند.

برای اعداد صحیح و بدون ممیز نوع داده‌های زیر استفاده می‌شود :

`sbyte`, `byte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`

و برای اعداد اعشاری می‌توانید از نوع داده‌های `float` و `double` استفاده کنید. نوع داده `float` برای اعداد اعشاری با دقت حداقل ۷ رقم اعشار استفاده می‌شود. در صورتی که ارقام عدد بیش از آن باشد عدد گرد می‌شود. مثلاً عدد  $123/45689$  به صورت عدد  $123/4568$  قابل نگهداری است. نوع داده `double` برای اعداد اعشاری بسیار بزرگ و یا بسیار کوچک مانند جرم و بار الکتریکی یک الکترون و با دقت زیاد ۱۵ رقم استفاده می‌شود.



در زبان برنامه‌نویسی `C#`، قبل از اینکه بتوانید داده‌ای را در یک متغیر ذخیره کنید باید متغیر را ایجاد (یا اعلان) کنید و در هنگام ایجاد کردن یک متغیر، باید نوع متغیر(نوع داده) را مشخص نمایید. مثال ;

#### ۴—۴— مقداردهی متغیرها

پس از تعریف یا ایجاد متغیر، می‌توانید در آن، مقداری را با توجه به نوع متغیر ذخیره کنید. توجه داشته باشید که در یک متغیر همواره فقط یک مقدار نگهداری می‌شود و با ذخیره کردن داده جدید در یک متغیر، مقدار قبلی آن از بین می‌رود. مقداردهی متغیرها به چند روش صورت می‌گیرد. با دستور زیر مستقیماً مقداری در متغیر قرار می‌گیرد به این دستور، دستور انتساب<sup>۱</sup> می‌گویند.

**مقدار = نام متغیر**

دستورات زیر را در نظر بگیرید :

`byte` age ;

`age` ۱۶ ;

متغیر `age` از نوع عدد صحیح اعلان شده و با عدد ۱۶ مقداردهی شده است.

در هنگام تعریف یا ایجاد متغیر نیز می‌توانید آن را مستقیماً مقداردهی کنید که به آن مقداردهی اولیه<sup>۱</sup> می‌گویند. الگوی آن چنین است :

### مقدار = نوعداده ناممتغیر

بنابراین دو دستور قبل را با الگوی بالا جایگزین می‌کنیم :

`byte age 16;`

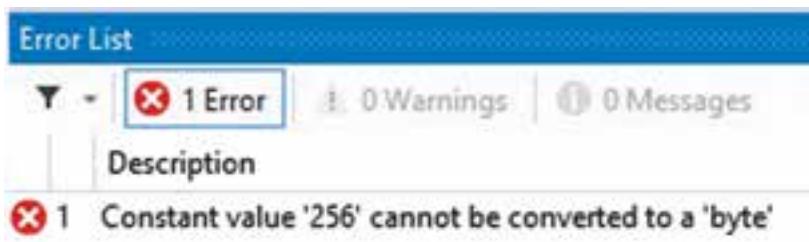
### نکته

- ۱- برای مشخص کردن اعداد مثبت نیازی به قراردادن علامت در پشت عدد نیست.
- ۲- در بین ارقام عدد باید ویرگول قرار دهید تا ارقام عدد، دسته بندی و جدا شوند.
- ۳- اگر عددی را بخواهید در داخل یک متغیر ذخیره کنید که خارج از ظرفیت و گنجایش آن متغیر باشد، مترجم متوجه آن شده و اجازه نمی‌دهد.

مثلاً دستور انتساب زیر را در نظر بگیرید :

`byte age 256;`

با توجه به ظرفیت متغیر `age` که حداقل عدد ۲۵۵ است، در هنگام ترجمه این دستور، خطای شکل ۱-۴ ظاهر می‌شود که شرح آن چنین است : «مقدار ثابت ۲۵۶ را نمی‌تواند به یک `byte` تبدیل شود».



شكل ۱-۴- خطأ در انتساب عدد صحيح ۲۵۶ در يك متغير نوع byte

<sup>۱</sup> In t a ze

در یک برنامه به زبان C# می‌توانید اعداد صحیح را در مبنای ۱۶ نیز بنویسید. برای این منظور قبل از عدد مورد نظر از پیشوند ۰X یا ۰x استفاده کنید که نشانه اعداد مبنای ۱۶ می‌باشد. مثلاً:

```
byte portValue = 0x1B;  
ushort portAddress = 0X00FF;
```

با اجرای این دستورات در متغیر portValue عدد ۲۷ و در متغیر portAddress عدد ۲۵۵ قرار می‌گیرد.<sup>۱</sup>

برای مشخص کردن انواع عددی دیگر از نشانه‌های جدول ۴-۲ استفاده می‌شود که در انتهای عدد ذکر می‌شود.

جدول ۴-۲ - نشانه‌های نوع اعداد ثابت

مثال	نشانه	نوع عدد
125U	u یا U	عدد صحیح مثبت
1700L	l یا L	عدد صحیح بزرگ
250000UL	UL	عدد صحیح بزرگ مثبت
2.5f	f یا F	عدد اعشاری با دقت معمولی
12.75d	d یا D	عدد اعشاری با دقت زیاد
12345678M	m یا M	عدد بسیار بزرگ

## نکته

اگر در برنامه، یک عدد اعشاری بدون نشانه بنویسید این عدد به عنوان عدد اعشاری با دقت زیاد در نظر گرفته می‌شود.

برای ذخیره اعداد اعشاری باید از متغیرهای نوع float یا double استفاده کنید. مثلاً برای نگهداری نمرات درسی (معمولًاً با دو رقم اعشار) یا اعداد گنگ مانند  $\pi$  باید از چنین متغیرهایی استفاده کرد. دستور زیر را در نظر بگیرید:

```
double PI = 3.141592653589793238;
```

در این دستور برای نگهداری عدد  $\pi$  متغیر PI با دقت زیاد اعلان و مقداردهی شده است.

$$1. (1B) = (1 \times 16) + 11 = 27$$

$$(FF) = (15 \times 16) + 15 = 255$$

## نکته

برای ذخیره اغلب داده‌ها مانند نمره یک درس، متغیر نوع float مناسب است.

اگر چه می‌توانید از متغیر نوع double نیز استفاده کنید ولی حافظه اشغالی این متغیر دو برابر متغیر نوع float است.

دستورات زیر را در نظر بگیرید:

```
float myPhysicMark;  
myPhysicMark = 17.75f;
```

در دستورات بالا برای ذخیره نمره درس فیزیک متغیری اعلان و مقداردهی شده است.

**سؤال:** در دستور اتساب، بعد از عدد اعشاری ۱۷.۷۵ حرف f نوشته شده است که شانه اعداد اعشاری با دقت معمولی است. آیا می‌توانید حرف f را نویسید؟

## نکته

در زبان C# هر عدد اعشاری داخل برنامه، به وسیله مترجم به عنوان نوع double در نظر گرفته می‌شود. بنابراین اگر بخواهید یک عدد ممیزی را در یک متغیر نوع float ذخیره کنید مترجم خطای هشدار می‌دهد. برای جلوگیری از این مسئله باید از متغیرهای نوع double در هنگام کار با اعداد اعشاری استفاده کنید و یا اینکه در جلوی اعداد اعشاری حرف F یا f را بنویسید تا مترجم، این عدد را به عنوان یک عدد نوع float در نظر بگیرد.

## ۴—۵— نشان دادن محتوای متغیرها بر روی صفحه نمایش

معمولًاً در برنامه‌ها لازم است محتوای متغیرها که شامل داده‌ها و یا نتایج پردازش با اطلاعات بر روی صفحه نشان داده شود تا کاربر از آنها آگاه شود. بدین منظور از متند Write() یا WriteLine() استفاده می‌کنیم که در فصل‌های قبلی برای نمایش یک پیام یا حاصل یک عبارت به کار گرفته شد. مثلاً برای نشان دادن محتوای متغیر age دستور زیر را می‌نویسیم:

```
byte age = 16;  
System.Console.WriteLine( age );
```

با توجه به اینکه در متغیر age عدد ۱۶ قرار دارد با اجرای دستور بالا، این عدد روی صفحه کنسول نشان داده می‌شود.

اگر شخص دیگری غیر از شما، این عدد را روی صفحه مشاهده کند، شاید متوجه نشود که این عدد چیست و شاید عدد ۱۶ را به عنوان نمره در نظر بگیرد. بنابراین بهتر است قبل از نمایش هر عدد، یک پیام (رشته) نیز نشان داده شود و به صورت کوتاه و مختصر منظور و مفهوم عددی را که قرار است روی صفحه نشان داده شود بیان کند. بنابراین دستور بالا را به صورت زیر می‌نویسیم :

```
System.Console.WriteLine("My age is " age );
```

با اجرای این دستور، عبارت زیر روی صفحه نشان داده می‌شود :

```
My age is 16
```

علامت در دستور بالا، به معنای عمل جمع ریاضی نیست بلکه به منظور کنار هم قرار دادن<sup>۱</sup> این دو مقدار (رشته‌ها) استفاده شده است. همان طور که در دستور زیر نیز از علامت استفاده شده است :

```
System.Console.WriteLine("I am " age "years old.");
```

با اجرای این دستور، عبارت زیر روی صفحه نشان داده می‌شود :

```
I am 16 years old.
```

مثال ۱-۴- استفاده از چند متغیر صحیح و اعشاری در برنامه ۱-۱، نشان داده شده است :

```
class VariableDemo
{
    static void Main()
    {
        // Declare some integer numbers variables
        int a = 10, b = 20, c;
        c = a - b;
        Console.WriteLine("a = " + a);
        Console.WriteLine("b = " + b);
        Console.WriteLine("a - b = " + c);
        // Declare some real numbers variables
        float lowPI = 3.141592653589793238f;
        double highPI = 3.141592653589793238;
```

```

// Print the results on the console
Console.WriteLine("Float PI is: "    lowPI);
Console.WriteLine("Double PI is: "   highPI);

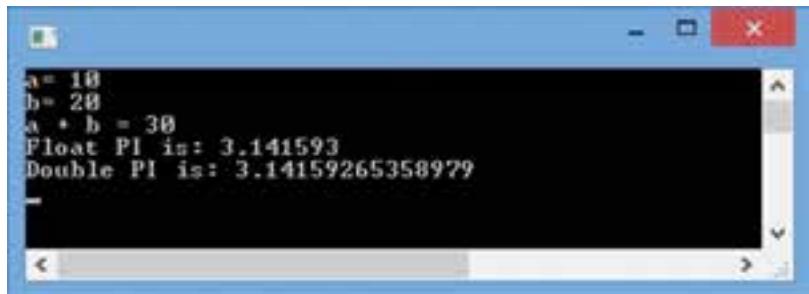
Console.ReadKey();
}

}

```

برنامه ۴-۱—تعريف و مقداردهی و نمایش محتوای متغیرها

در برنامه ۱-۴، سه متغیر  $a$ ,  $b$ ,  $c$  از نوع عدد صحیح تعریف شده‌اند و در متغیر  $c$  نتیجه حاصل جمع دو عدد  $a$  و  $b$  قرار می‌گیرد. در دو متغیر اعشاری  $lowPI$  و  $highPI$  عدد  $\pi$  با دقت‌های مختلف نگهداری شده است.



شکل ۴-۲—خروجی برنامه ۱-۴

## ۶-۴—نحوه نام‌گذاری متغیرها

همان طور که پدر و مادر برای انتخاب یک نام خوب و مناسب برای فرزند خود، وقت زیادی می‌گذارند و نکاتی از جمله زیبایی نام و با معنا بودن را رعایت می‌کنند و همچنین سعی می‌کنند که این نام قبلًاً در خانواده و یا تزدیکان انتخاب نشده باشد، به همان صورت برنامه نویس نیز برای متغیرها باید یک نام صحیح، بامعا و غیرتکراری در محدوده آن را انتخاب کند این کار باید با حوصله انجام شود و نام انتخایی نباید با نام‌های دیگر یکسان باشد.

در زبان C# در نام گذاری متغیرها، رعایت موارد زیر **الزامی** است :

۱- استفاده از حروف الفبا، اعداد و کاراکتر زیرخط، **مجاز** است.

۲- نام متغیر **نمی تواند** با عدد شروع شود.

۳- نام انتخابی **نمی تواند** با کلمات کلیدی یا رزرو شده باشد.

۴- استفاده از علامت فاصله و خط تیره در نام متغیر **مجاز نیست**.

در انتخاب نام متغیرها، **بهتر** است نکات زیر رعایت شود :

- نام با معنی و با توجه به کاربرد متغیر در برنامه انتخاب شود. مانند woodLength

- از نام‌های مخفف استفاده نکید چون خواندن آنها مشکل است. مانند crntStdnt

- اولین حرف نام متغیر را با حروف کوچک شروع کنید و اگر نام متغیر از چند کلمه تشکیل شده، برای خوانایی، حرف اول کلمات بعدی را با حروف بزرگ بنویسید. به این روش نوشتن نام، کوهان شتری<sup>۱</sup> می‌گویند.

چند نمونه نام متغیر دو کلمه‌ای به روش کوهان شتری را ملاحظه می‌کنید :

fileName , userName , notFound , localIP

روش دیگری برای نام گذاری متغیرها به نام روش مجارستانی<sup>۲</sup> به وسیله آفای چارلز سیمونی<sup>۳</sup> ابداع شده که در ابتدای نام متغیر، مخفف نوع داده ذکر می‌شود که یک روش شناخته شده و معروف برای نام گذاری متغیرها است.

چند نمونه نام متغیر دو کلمه‌ای، به روش مجارستانی را ملاحظه می‌کنید :

IntNumber, LngSalary, BlnStatus

در این کتاب از روش کوهان شتری برای نام گذاری متغیرها استفاده شده است.

## نکته

با توجه به حساسیت زبان C# به حروف کوچک و بزرگ، در نام گذاری متغیرها به این نکته دقت کنید که متغیر a و A مستقل هستند.

۱— Came Notat on

۲— Hungar an Notat on

۳— Char es S mony

در جدول زیر تعدادی نام متغیر و علت مجاز یا غیرمجاز بودن این نام‌ها می‌بینید.

### جدول ۴-۳- نمونه متغیرهای مجاز و غیر مجاز

نام متغیر	توضیح
1a	غیر مجاز، نام متغیر نباید با عدد شروع شود
a1	مجاز
employee Salary	غیر مجاز، بین کلمات نباید فاصله وجود داشته باشد
First	مجاز
Hello!	غیر مجاز، علامت تعجب نباید در نام وجود داشته باشد
payRate	مجاز
one+two	غیر مجاز، علامت + در یک نام نباید قرار داشته باشد
Conversion	مجاز
counter 1	مجاز
2nd	غیر مجاز، نام نمی‌تواند با عدد شروع شود

در دستورات زیر، چند نمونه از اعلان و مقدار دهنده متغیرها را مشاهده می‌کنید :

```
تعريف متغير برای نگهداري سرعت خودرو با مقداردهی اوليه // int speed = 70;
تعريف سه متغير برای اضلاع مثلث float a,b,c; // Triangle sides
تعريف يك متغير برای نگهداري مساحت مثلث float triangleArea; // triangleArea
متغيري برای نگهداري بار الکتریکی يك جسم double electricalCharge; // electricalCharge
```

### ۴-۷- کار با اعداد اعشاری

در فیزیک و شیمی و یا به طور کلی در علوم، با اعداد بسیار کوچک و بسیار بزرگ سروکار داریم. اگر بخواهید عدد اعشاری بسیار کوچک و یا بسیار بزرگی را در یک متغیر ذخیره کنید، می‌توانید آن را به صورت کوتاه با روشنی شبیه نماد علمی بنویسید. برای اینکه با این روش آشنا شوید ابتدا لازم است روش نماد علمی را یادآوری کنیم.

در روش نماد علمی، هر عدد از ۲ بخش تشکیل می‌شود که با علامت ضرب از یکدیگر جدا شده اند. بخش اول یک عدد اعشاری بین ۱ تا ۹ است ( فقط یک رقم صحیح دارد) که به آن مانتیس می‌گویند و قسمت دوم که به صورت توانی از عدد  $10^{-n}$  است که به آن نما گفته می‌شود جدول ۴-۴).

#### جدول ۴-۴- مثال‌هایی از فرم نماد علمی

فرم معمولی	فرم نماد علمی
4380000	$4.38 \times 10^6$
0000265	$2.65 \times 10^{-5}$
47 9832	$47.9832 \times 10^1$
1000000	$1 \times 10^7$
-5600	$-5.6 \times 10^3$

۱-۷-۴- فرم نقطه شناور : در زبان C# از یک فرم نماد علمی برای نمایش اعداد اعشاری استفاده می‌شود که به آن فرم نقطه شناور<sup>۱</sup> گفته می‌شود. در این فرم مانند نماد علمی، عدد از دو بخش مانیس و نما تشکیل شده است که با حرف E از یکدیگر جدا شده اند. در این فرم، توان  $10^0$  بعد از حرف E نوشته می‌شود و خبری از علامت ضرب بین دو قسمت نیست (جدول ۴-۵).

#### جدول ۴-۵- مثال‌هایی از نمایش اعداد در فرم نقطه شناور

عدد	نمایش عدد در فرم نقطه شناور
75 924	$7.5924E1$
0 18	$1.8E-1$
0 0000453	$4.53E-5$
-1 482	$-1.482E0$
7800 0	$7.8E3$

بار الکتریکی یک الکترون  $1.602 \times 10^{-19}$  کولن است که در متغیرهای زیر ذخیره شده است می‌توانید

این عدد بسیار کوچک را در یک متغیر نوع **double** یا **float** به صورت نقطه شناور ذخیره کنید

**Double** electricalCharge = 1.602E -19;

**Float** electricalCharges = 1.602E -19F;

**سؤال:** کدامیک از دستورات بالا را ترجیح می‌دهید؟ چرا؟

**۲-۷-۴**— دقت اعداد قابل نمایش در فرم نقطه شناور : حداکثر تعداد ارقام غیر صفر و با معنی مانیس عدد را، دقت عدد می نامند. دقت اعداد نوع `float` ۶ یا ۷ رقم و اعداد نوع `double` ۱۵ رقم است.

### نکته

به غیر از میزان حافظه مصرفی و محدوده اعداد قابل نمایش در نوع داده های `float` و `dou-`ble، میزان دقت این دو نوع داده نیز با یکدیگر متفاوت است.

### ۸-۴- نوع داده منطقی یا بولین<sup>۱</sup> (`bool`)

در انتهای جدول ۱-۴ نوع داده منطقی یا بولین (`bool`) را مشاهده می کنید این نوع داده فقط شامل دو مقدار درست (`true`) و نادرست (`false`) است. متغیرهایی که از این نوع داده تعریف و ایجاد می شوند، قادرند یکی از دو مقدار `true` و `false` را پذیرند که با حروف کوچک انگلیسی نوشته می شوند. دستورات زیر متغیر `response` را اعلان و با `false` مقدار دهی اولیه می کند. سپس محتوای متغیر بر روی صفحه نمایش چاپ می شود.

```
bool response = false;  
System.Console.WriteLine(response);
```

### ۹-۴- نوع داده حرفی یا کاراکتری (`char`)

کاراکتر عبارت است از یک حرف الفباء یا یک علامت و یا ت莎ههایی مانند آنچه که در روی دکمه های صفحه کلید مشاهده می کنید. در کامپیوتر برای هر دکمه صفحه کلید یک کد عددی در نظر گرفته می شود و در واقع هنگامی که یک کلید را فشار می دهید کدی متناظر با آن کلید تولید و این کد به صورت دنباله ای از صفر و یک در حافظه کامپیوتر ذخیره می شود. یک کاراکتر را می توانید با کد آن مشخص کنید و یا علامت آن را در بین علامت '' (تک کوتشن) قرار دهید.  
چند نمونه از کاراکترها را در زیر مشاهده می کنید.

'A' , 'a' , '&' , '\$' , '+' , ''

- در داخل علامت‌ها فاصله (Space) نیز به عنوان یک کاراکتر در نظر گرفته می‌شود.
- در داده کاراکتری، فقط یک کاراکتر باید بین علامت<sup>۱۰</sup> وجود داشته باشد.

توجه داشته باشید که در زبان برنامه‌نویسی C#، نوع داده char به منظور کار با داده‌های کاراکتری پیش‌بینی شده است. اگر بخواهید یک کاراکتر را در یک متغیر ذخیره کنید باید متغیری از نوع داده char تعریف کنید.

گنجایش این متغیر، دو بایت است و کد کاراکتر را نگهداری می‌کند.

`char` : نام متغیر

در دستور زیر، متغیری به نام ch از نوع char تعریف و حرف A در آن ذخیره شده است.

`char ch 'A';`

متغیر ch یک متغیر دو بایتی است که در آن کد کاراکتر نگهداری می‌شود. این کد دو بایتی طبق استاندارد یونیکد (Unicode) است. در استاندارد یونیکد، کد هر کاراکتر عددی بین ۰ تا ۶۵۵۳۵ است و تمام نشانه‌ها، علامت و حروف الفباء زبان‌های مختلف کشورها به وسیله این استاندارد کدبندی شده است. این کدبندی مستقل از سیستم عامل، زبان برنامه‌نویسی و سخت افزار است.

در برنامه می‌توانید به جای قرار دادن کاراکتر در علامت<sup>۱۱</sup> از کد آنها استفاده کنید، چون کاراکترها فقط محدود به آنچه که بر روی صفحه کلید قرار دارد نیستند. بنابراین با داشتن کد هر کاراکتر می‌توانید آن را در برنامه استفاده کنید. معمولاً برای سادگی، این کد را در مبنای ۱۶ ذکر می‌کنند. با توجه به اینکه در کدبندی یونیکد، از دو بایت استفاده می‌شود و هر ۴ بیت یک رقم مبنای ۱۶ است، برای نمایش این کد در مبنای ۱۶ از یک عدد ۴ رقمی استفاده می‌شود. مثلاً کد کاراکتر A عدد ۶۵ در مبنای ۱۰ است. معادل این کد در مبنای ۱۶ عدد ۴۱ است. این عدد را در داخل علامت<sup>۱۲</sup> قرار می‌دهیم و برای مشخص کردن این عدد به عنوان کد کاراکتر، قبل از آن، علامت `\u` یا `\x` را می‌نویسیم مانند الگوی زیر:

`\u4e0e` کد ۴ رقمی

در دستور زیر، متغیر ch اعلان و حرف A در آن ذخیره می‌شود. از صفرهای اضافی قبل از عدد، برای تکمیل کد به صورت ۴ رقمی استفاده شده است.

`char ch '\u0041'; // Same as Char ch 'A'`

## ۱۰-۴- نوع داده رشته‌ای (String)

نوع داده `char`, تنها برای نگهداری یک کاراکتر مناسب است. برای هنگامی که داده‌ها, مانند نام یک شخص، بیش از یک کاراکتر است باید از نوع داده رشته‌ای (`string`) استفاده کنیم. یک رشته شامل تعدادی حروف و کاراکتر است که در بین جفت کوتیشن `" "` قرار گرفته است. مثلاً `"Mohammad"` یک داده رشته‌ای شامل ۸ کارکتر است.

۱۱-۴- متغیر رشته‌ای: برای نگهداری داده‌های رشته‌ای در برنامه، باید متغیر رشته‌ای تعریف کنید. متغیرهای رشته‌ای قادرند آدرس محلی که یک داده رشته‌ای وجود دارد را نگهداری کنند یا به عبارت ساده، قادرند داده‌های رشته‌ای را ذخیره کنند. بنابراین با متغیری از نوع رشته، قادر خواهیم بود به داده‌های رشته‌ای دسترسی داشته باشیم.  
دستور زیر یک متغیر رشته‌ای به نام `name` را اعلان می‌کند.

```
string name;
```

و با دستور انتساب زیر، می‌توانید رشته `"Mohammad"` را در متغیر `name` ذخیره کنید و در طول برنامه به آن دسترسی داشته باشید :

```
name = "Mohammad";
```

**سؤال:** آیا می‌توانید دو دستور بالا را با یک دستور جایگزین کنید؟

۱۲-۴- عملیات بر روی داده‌ها یا متغیرهای رشته‌ای : عملیات مختلفی بر روی رشته‌ها می‌توان انجام داد، یکی از عملیات‌های معمول و کاربردی، الحاق یا کنارهم قرار دادن رشته‌ها است. برای الحاق دو رشته از علامت `+` استفاده می‌شود.  
قطعه کد زیر را در نظر بگیرید. در این کدها، محتوای متغیر رشته‌ای `name` با رشته `Welcome` الحاق شده و حاصل در متغیر `message` قرار می‌گیرد.

```
string name = "Mohammad";
```

```
string message = "Welcome" + name;
```

```
System.Console.WriteLine(message);
```

نتیجه خروجی چنین خواهد بود :

```
WelcomeMohammad
```

**سؤال:** اگر بخواهید خروجی به صورت خوانا `Welcome Mohammad` شود یعنی بین دو کلمه یک فاصله قرار گیرد، چه تغییری در دستورات بالا ایجاد می‌کنید؟

با توجه به این که در زبان C#، علامت `#` برای عمل جمع ریاضی و هم برای الحاق رشته‌ها استفاده می‌شود، در به کارگیری این علامت در برنامه باید دقت کافی داشته باشد.

## کار در کارگاه ۱

**مثال ۲-۴:** برنامه زیر مانند برنامه ۱-۴ برای محاسبه مجموع دو عدد `a` و `b` نوشته شده است با این تفاوت که حاصل جمع در متغیری ذخیره نشده بلکه روی صفحه نمایش، نشان داده می‌شود. به خط آخر این برنامه توجه کنید. آیا به نظر شما با اجرای این برنامه، عدد ۲۵ به عنوان حاصل جمع نشان داده می‌شود؟ چرا؟

```
class VariableDemo
{
    static void Main()
    {
        // Declare two integer variables
        int a, b;

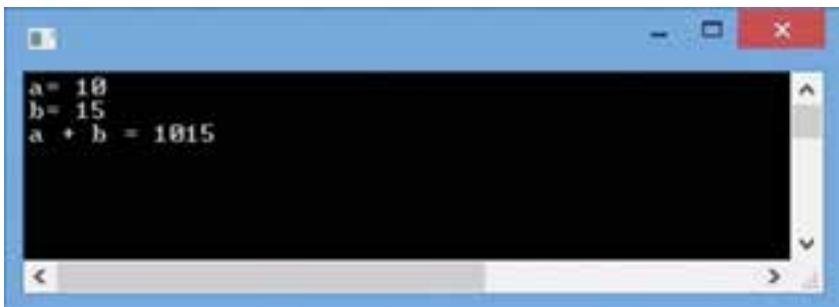
        a = 10;
        b = 15;

        Console.WriteLine("a = " + a);
        Console.WriteLine("b = " + b);

        // What is displayed?
        Console.WriteLine("a + b = " + a + b); ←
    }
}
```

برنامه ۲-۴—دقت در استفاده از علامت `=` در هنگام کار با اعداد و رشته‌ها

در خط آخر برنامه ۴-۲، علامت دو بار استفاده شده است که هر دو علامت، عمل الحاق رشته را انجام می‌دهند . خروجی این برنامه مطابق شکل ۴-۳ است :

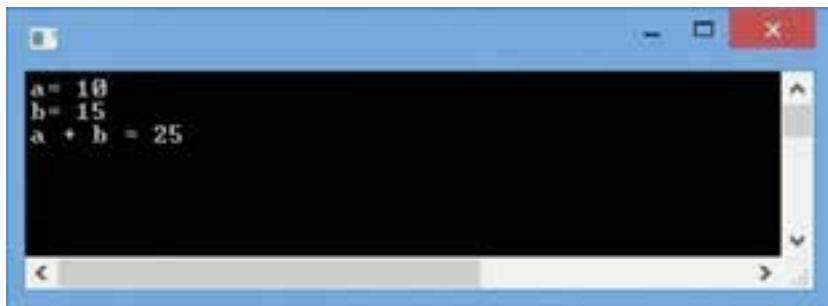


شکل ۴-۳ - خروجی برنامه ۴-۲

برای رفع اشکال برنامه ۴-۲، خط آخر را به صورت زیر بازنویسی می‌کنیم :

```
Console.WriteLine("a  b  " (a  b));
```

با تصحیح خط آخر، نتیجه اجرای برنامه شکل ۴-۴ خواهد شد :



شکل ۴-۴ - خروجی برنامه ۴-۲ پس از تصحیح

**سؤال:** با توجه به خروجی برنامه ۴-۴ چرا پرانتز سبب تغییر مقدار خروجی شد؟

## ۱۱-۴- دریافت رشته

تاکنون داده‌های مشخص و ثابتی رادر داخل برنامه استفاده کردیم. این داده‌ها به وسیله برنامه‌نویس درون برنامه تعیین شده بود. حال می‌خواهیم برنامه‌های خود را کاربردی کنیم و داده‌ای را از کاربر دریافت کنیم. برای این منظور از متدهای ReadLine() استفاده می‌کنیم که به کاربر اجازه می‌دهد تا داده مورد نظر خود را از طریق صفحه کلید وارد کند.

متدهای ReadLine() مانند مذکور که تاکنون خوانده ایم در کلاس `Console` تعریف شده است و در فضای نامی `System` قرار دارد. بنابراین به صورت زیر استفاده می‌شود :

```
System.Console.ReadLine();
```

کامپیوتر با اجرای این متدهای موقوف شده و منتظر دریافت داده می‌شود. کاربر می‌تواند داده مورد نظر خود را تایپ کند و در پایان دکمه Enter را بزنند که در این صورت، داده به صورت یک رشته در حافظه ذخیره می‌شود. اگر رشته دریافتی را با دستور انتساب در یک متغیر رشته‌ای ذخیره کنیم، داده وارد شده، در برنامه قابل دسترسی خواهد بود.

برای مثال می‌خواهیم نام و نام خانوادگی یک شخص را از کاربر سؤال کرده و در برنامه استفاده کنیم. برای این منظور ابتدا دو متغیر رشته‌ای به نام `name` و `family` از نوع رشته‌ای اعلان می‌کنیم و سپس از متدهای ReadLine() برای دریافت نام و نام خانوادگی به صورت زیر استفاده می‌نماییم :

```
string name, family;  
name = System.Console.ReadLine();  
family = System.Console.ReadLine();
```

### نکته

متدهای ReadKey() و ReadLine() شبیه متد ReadKey() است با این تفاوت که متد ReadLine() فقط منتظر دریافت یک کلید می‌شود اما در متد ReadLine() تا هنگامی که کلید Enter زده نشده است کامپیوتر منتظر می‌ماند.

توجه داشته باشید وقتی کامپیوتر منتظر دریافت داده است کاربر باید بداند که چه داده‌ای را لازم است وارد کند (نام، نمره، سن) بنابراین لازم است قبل از استفاده از متد ReadLine() یک دستور برای نمایش یک پیام و توضیحی کوتاه در مورد اینکه کامپیوتر منتظر دریافت چه داده‌ای است در برنامه نوشته شود. از متد Write() بدین منظور استفاده می‌کنیم.

مثالاً برای دریافت نام کاربر دستورات زیر را می‌نویسیم :

```
string name;  
System.Console.WriteLine("Enter your name: ");  
name = System.Console.ReadLine();
```

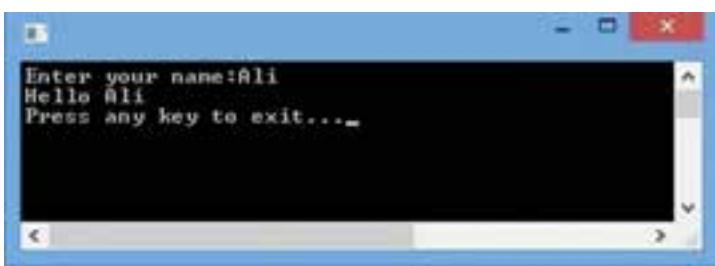
## کار در کارگاه ۲

مثال ۳-۴ : نام کاربر از ورودی دریافت شده و خطاب به او پیام خوشامدگویی اعلام شود.

```
using System;  
class HelloYourName  
{  
    static void Main()  
    {  
        string name;  
        Console.WriteLine("Enter your name: ");  
        name = Console.ReadLine();  
        Console.WriteLine("Hello " + name);  
        Console.WriteLine("Press any key to exit...");  
        Console.ReadKey();  
    }  
}
```

برنامه ۳-۴ - خوشامدگویی به کاربر

اگر فرض کنید که کاربر، نام Ali را وارد کند، خروجی برنامه به صورت شکل ۵-۴ خواهد بود :



شکل ۵-۴ - خروجی برنامه ۳-۴

**مثال ۴-۴ :** می خواهیم به برنامه ۳-۴، دستوراتی اضافه کنیم که علاوه بر دریافت نام کاربر، نام خانوادگی وی نیز سوال شود و سپس نام و نام خانوادگی را در یک خط نمایش دهد.

در برنامه ۳-۴، کافی است یک متغیر رشته‌ای به نام family تعریف کرده و از متد() برای دریافت نام خانوادگی استفاده کنیم. برای نمایش نام و نام خانوادگی در یک خط نیز، از علامت برای الحاق رشته‌ها استفاده می‌کنیم (کدهای برجسته، تغییرات جدید هستند).

```
using System;  
class HelloYourName  
{  
    static void Main()  
    {  
        string name, family;  
        Console.WriteLine("Enter your name: ");  
        name = Console.ReadLine();  
  
        Console.WriteLine("Enter your family: ");  
        family = Console.ReadLine();  
  
        Console.WriteLine("Hello " + name + " " + family);  
  
        Console.WriteLine("Press any key to exit...");  
        Console.ReadKey();  
    }  
}
```

برنامه ۴-۴- تکمیل برنامه خوشنامدگویی به کاربر

**سؤال:** خروجی برنامه تغییر یافته چه تفاوتی با شکل ۴-۵ دارد؟

**مثال ۴-۵ :** می خواهیم برنامه ای بنویسیم که دو عدد دلخواه از کاربر دریافت کند و مجموع آن ها را حساب کرده و روی صفحه نمایش، نشان دهد.

برای دریافت داده ها از کاربر، از متده ReadLine() مانند مثال های قبلی استفاده می کنیم.

داده های دریافتی به وسیله این متده، در قالب رشته در حافظه ذخیره می شوند، بنابراین برای دسترسی به آن ها باید از متغیر های رشته ای استفاده کنیم.

```
using System;
class GetNumbers
{
    static void Main()
    {
        string firstNumber, secondNumber;

        Console.WriteLine("Enter a number: ");
        firstNumber = Console.ReadLine();

        Console.WriteLine("Enter another number: ");
        secondNumber = Console.ReadLine();

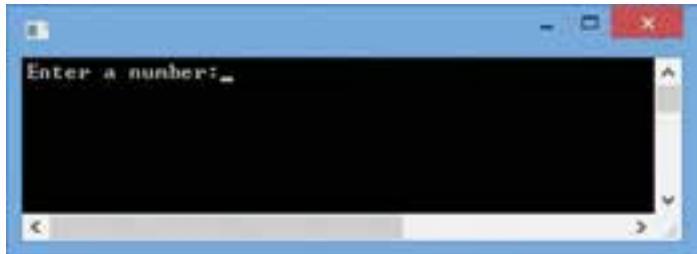
        Console.WriteLine("Total " + (firstNumber + secondNumber));

        Console.WriteLine("Press any key to exit... ");
        Console.ReadKey();
    }
}
```

**برنامه ۴-۵**— اولین تلاش برای دریافت داده های عددی

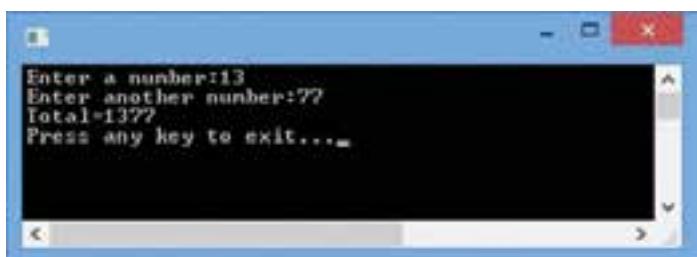
با اجرای این برنامه، پنجره ای ظاهر می شود که از کاربر خواسته می شود که یک عدد وارد کند

(شکل ۴-۶).



شکل ۴-۶- خروجی برنامه ۵-۴ دریافت یک عدد

پس از وارد کردن یک عدد و زدن دکمه Enter، عدد دیگری خواسته می‌شود. فرض کنید اعداد ۱۳ و ۷۷ نو سط کاربر وارد شود (شکل ۴-۶).



شکل ۴-۷- خروجی برنامه ۵-۴

**سوال:** کاربر با وارد کردن دو عدد ۱۳ و ۷۷ انتظار داشت که مجموع آنها یعنی عدد ۹۰ روی صفحه نشان داده شود اما به جای آن، عدد ۱۳۷۷ نشان داده شد. چرا؟

همان طور که بیان شد متده ReadLine() داده دریافتی را به صورت یک رشته در حافظه ذخیره می‌کند و در برنامه ۵-۴ از متغیرهای رشته‌ای firstNumber و secondNumber برای دسترسی به داده‌های ورودی استفاده کردیم. بنابراین علامت در دستور زیر عمل الحق دو رشته مثلاً "13" و "77" را انجام می‌دهد و طبیعی است که نباید انتظار عمل جمع ریاضی داشته باشیم.

**۱۱-۴- دریافت اعداد :** با توجه به این که داده‌های دریافتی به وسیله متده ReadLine() همواره به صورت رشته تحویل داده می‌شود باید به وسیله دستوری، رشته دریافتی را به عدد تبدیل کنیم. بنابراین به متده نیاز داریم که رشته شامل ارقام را به ارزش عددی تبدیل کند تا بتوانیم روی آنها محاسبات ریاضی انجام دهیم.

خوب‌بختانه برای انواع داده‌های عددی، متده به نام Parse() از قبیل تعریف شده است که می‌تواند از یک رشته شامل ارقام، معادل عددی آن را بدست آورد. مثلاً برای تجزیه رشته "259" به ارزش عددی، با توجه به این که درون رشته، یک عدد صحیح قرار دارد، از متده Parse() مربوط به نوع داده int استفاده می‌کنیم :

```
int.Parse("259");
```

### نکته

به عمل بررسی کاراکتر به کاراکتر یک رشته، برای جدا کردن و بدست آوردن یک مقدار با معنی، تجزیه کردن<sup>۱</sup> می‌گویند.

حاصل اجرای این متده، عدد ۲۵۹ است که باید در یک متغیر نوع صحیح ذخیره شود. بنابراین استفاده مفید از این متده به صورت زیر خواهد بود :

```
int a;  
a = int.Parse("259");  
می‌توانید دو دستور بالا را با دستور زیر جایگزین نمایید :  
int a = int.Parse("259");
```

اگر رشته‌ای حاوی عدد اعشاری باشد باید از متده Parse() مربوط به نوع داده اعشاری مثلاً float یا double استفاده کنید. مثلاً برای تبدیل رشته "2.50" به عدد 2.5 از دستورات زیر استفاده می‌کنیم :

```
float b;  
b = float.Parse("2.50");
```

با استفاده از متده Parse() می‌توانیم رشته دریافتی که به وسیله متده ReadLine() از کاربر گرفته شده است را به عدد تبدیل کنیم به شرط اینکه حاوی اعداد باشد.

```
string input;  
float number;  
input = Console.ReadLine();  
number = float.Parse(input);
```

---

```
1.Parse
```

همچنین می‌توانید متدهای ReadLine() و Parse() استفاده کنید که در این صورت نیازی به متغیر رشته‌ای نیست:

float number;

number = float.Parse(Console.ReadLine());

**سؤال:** آیا می‌توانید دو دستور بالا را، باز هم خلاصه‌تر کنید؟

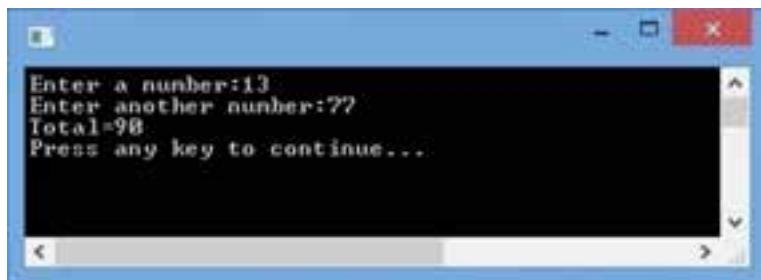
### کار در کارگاه ۳

مثال ۶\_۴ : با تکمیل برنامه ۶\_۵ مجموع دو عدد دریافتی را چاپ نمایید.

```
using System;  
class GetNumbers  
{  
    static void Main()  
    {  
        string input;  
        float firstNumber, secondNumber;  
  
        Console.WriteLine("Enter a number: ");  
        input = Console.ReadLine();  
        firstNumber = float.Parse(input);  
  
        Console.WriteLine("Enter another number: ");  
        input = Console.ReadLine();  
        secondNumber = float.Parse(input);  
  
        Console.WriteLine("Total " + (firstNumber + secondNumber));  
  
        Console.WriteLine("Press any key to continue...");  
        Console.ReadKey();  
    }  
}
```

برنامه ۶\_۴ – دریافت دو عدد و محاسبه مجموع

نتیجه اجرای برنامه در شکل ۴-۸ نشان داده شده است :



شکل ۴-۸- خروجی برنامه ۴-۶

برنامه‌های زیر را در محیط VS ایجاد کنید.

۱- دستورات زیر را در داخل متد Main() برای شناسایی انواع متغیرها بنویسید. با اضافه کردن دستورات WriteLine()، محتوای متغیرها را بر روی صفحه نمایش، نشان دهید.

```
// Declare and initialize some variables
```

```
// Use long suffix.
```

```
long aLongNumber = 10000L;
```

```
// Use double suffix.
```

```
double aDoubleNumber = 123.764D;
```

```
// Use float suffix.
```

```
float aFloatNumber = 100.50F;
```

```
// Use unsigned suffix.
```

```
uint anUnsignedNumber = 1000U;
```

```
// Use decimal suffix.
```

```
decimal aDecimalNumber = 4000.1234M;
```

```
// Use unsigned suffix and long suffix.
```

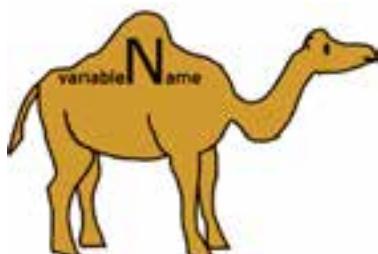
```
ulong anUnsignedLong = 10002000300040005000UL;
```

- ۲** برنامه شماره ۴-۴ (خوشامدگویی به کاربر) را با داده‌های مختلف (نام خود، نام همکلاسی‌ها) آزمایش کنید.
- ۳** برنامه شماره ۶-۴ (دربیافت دو عدد و محاسبه مجموع) را با اعداد صحیح و اعشاری آزمایش کنید.

## خودآزمایی فصل چهارم

- ۱- چه نوع حافظه کامپیوتر برای نگهداری حجم کمی از داده‌ها در طول اجرای یک برنامه مناسب است؟
- ۲- در زبان‌های برنامه‌نویسی، مکانی از حافظه برای نگهداری موقتی داده و اطلاعات ..... نامیده می‌شود.
- ۳- منظور از نوع داده چیست؟
- ۴- نوع متغیر چه ویژگی‌هایی را نشان می‌دهد؟
- ۵- تفاوت‌های بین نوع داده float و double را نام ببرید.
- ۶- برای نگهداری هر یک از داده‌های زیر در برنامه، یک متغیر مناسب تعریف کنید.
  - الف) سن افراد
  - ب) درجه حرارت محیط اتاق
  - پ) حقوق کارمند

ج) وضعیت خاموش و روشن بودن یک لامپ
- ۷- تحقیقی کوتاه بر روی روش نام گذاری مجارستانی (Hungarian Notation) با استفاده از اینترنت داشته باشید. با توجه به محیط‌های برنامه‌نویسی (IDE) پیشرفته، مانند ویزوال استودیو، نشان دهید که دیگر نیازی به ذکر نوع داده در ابتدای نام متغیر که در روش مجارستانی استفاده می‌شود، وجود ندارد.
- ۸- شکل زیر چه روشی را برای نام گذاری متغیرها نشان می‌دهد؟ نام این روش چیست؟



- ۹- چرا در هنگام ترجمه دستور زیر، خطأ ظاهر می‌شود؟ چگونه این خطأ را برطرف می‌کنید?  
`short value 66000 / 2;`

۱- در جدول زیر، کدامیک از نام‌های متغیر، غیر مجاز است و یا مناسب نیستند. آنها را تصویح کنید. (اولین ردیف جدول برای شما پاسخ داده شده است).

نام متغیر	کاربرد متغیر با توجه به معنی آن	نام متغیر	نام پیشنهادی
		مجاز نامناسب غیرمجاز	شما
networkOK	وضعیت شبکه	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
29yesitsme		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
myCurrentID		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
intValue		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
8%.tax		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
woodLength		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
glassArea		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
width		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
height		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
MySalary		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
employeeSalary		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	

۱۱- کدامیک از داده‌های زیر یک کاراکتری محسوب می‌شود؟ برای پاسخ خود دلیل بیاورید.

'! ', 'abc', '+' , '!' , '&' , '\', '\"' , '.\''

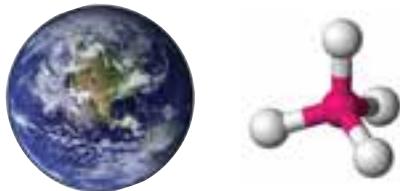
۱۲- در زبان C#، برای دریافت داده از ورودی، از متدهای `Console.ReadLine()` و `Console.Read()` استفاده می‌کنیم.

۱۳- اعداد زیر را در متغیرهای مناسب با استفاده از روش نقطه شناور جای دهید.

الف) عمر زمین بر حسب سال  $4,600,000$ .

ب) فاصله زمین تا خورشید  $149,600,000$  متر است.

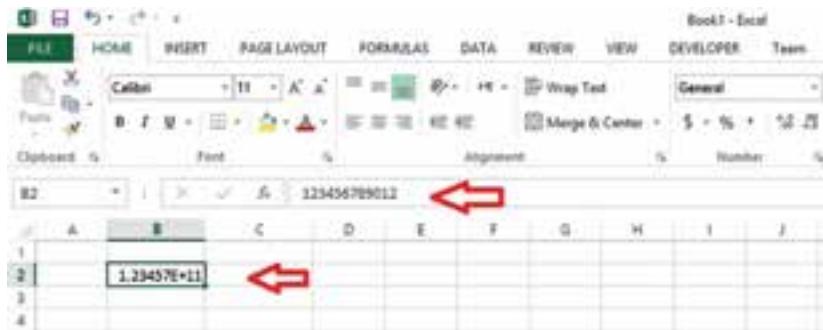
پ) اندازه جرم یک اتم کربن (جرم اتمی) برابر با  $166 \times 10^{-26}$  kg است.



۱۴- با توجه به اینکه لازم نیست مانیسین  $\circ$  تا ۱ باشد، جدول زیر را تکمیل کنید.

عدد	نمایش عدد در فرم نقطه شناور	گونه دیگری از فرم نقطه شناور
75 924	7 5924E1	
0 18	1 8E-1	
0 0000453	4 53E-5	
-1 482	-1 482E0	
7800 0	7 8E3	

۱۵- در برنامه های صفحه گسترده، اعداد بزرگ به صورت نقطه شناور نیز نمایش داده می شوند. وارد برنامه اکسل شوید و عدد بزرگی را وارد نموده و کلید Enter را بزنید. چه عددی روی صفحه نشان داده می شود؟ پس از مشاهده فرم نقطه شناور، سعی کنید با تغییر فرمت سلول مربوطه، عدد را به صورت معمولی نشان دهید.



۱۶- کدام یک از دستورات زیر، می‌تواند مجموع دو عدد a و b را به طور صحیح نشان دهد؟  
نتیجه اجرای هر یک از دستورات را نیز بنویسید.

```
int a = 10, b = 20;
System.Console.WriteLine("a" + "b");
System.Console.WriteLine("a + b");
System.Console.WriteLine(a + b);
System.Console.WriteLine("a + b" + a + b);
```

۱۷- سؤال زیر به زبان انگلیسی است. آن را خوانده و پاسخ صحیح را انتخاب کنید.  
The C# method that prints a line of output on the screen and then positions the cursor on the next line is .....

- A) `println()`
- B) `DisplayLine()`
- C) `WriteLine()`
- D) `Write()`

## تمرینات برنامه‌نویسی فصل چهارم

- ۱- برنامه‌ای بنویسید که دو عدد از کاربر دریافت نماید و حاصل جمع و حاصل تفریق آنها را نمایش دهد. (اعداد ورودی ممکن است صحیح و یا اعشاری باشد).
- ۲- برنامه‌ای بنویسید که نام و نام خانوادگی و سن کاربر را دریافت کند و سپس اطلاعات دریافتی را با رنگ‌های دلخواه روی صفحه نمایش، نشان دهد.
- ۳- با استفاده از یک برنامه ساده شامل متدهای `WriteLine()`، رشته‌ها یا کاراکترهای جدول زیر را نمایش دهید تا بتوانید معادل آن‌ها را پیدا کرده و جدول را کامل کنید.

کد حرف یا رشته	معادل	کد حرف یا رشته	معادل
'\u0007'		"0\u0020"	
"b\u0061ck"		"C\u0023"	
'\u000a\'		'\u0040'	
"12" + "8"		'\u0030'	

## واژگان و اصطلاحات انگلیسی فصل چهارم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Assignment	
۲	Boolean	
۳	Built-In Data Type	
۴	Camel Notation	
۵	Concatenate	
۶	Floating point notation	
۷	Hungarian Notation	
۸	Initialize	
۹	Integer Numbers	
۱۰	Precision	
۱۱	Primitive Data Type	
۱۲	Random Access Memory	
۱۳	Representation	
۱۴	Significant digits	
۱۵	Unsigned numbers	
۱۶	Variable	

### عبارت‌های محاسباتی

یکی از توانایی‌های کامپیوتر قدرت و سرعت بالا در انجام محاسبات است. انجام بیش از یک میلیارد عمل جمع بر روی اعداد نوع صحیح در کمتر از یک ثانیه، بسیار شگفت آور است. کامپیوترها با استفاده از چنین توانایی قادر هستند عملیات و پردازش مورد نظر در برنامه را با سرعت بر روی داده‌ها انجام دهند. از جمله پردازش‌هایی که از کامپیوتر استفاده می‌شود، جستجوی داده مورد نظر در بین داده‌ها یا عمل مرتب سازی داده‌ها به ترتیب خاصی است. برای انجام چنین پردازش‌هایی لازم است ابتدا با انواع عبارت‌ها و محاسبات در زبان #C آشنا شویم و سپس از آنها در برنامه‌های خود استفاده کنیم.

**پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند:**

- ۱- عملوند، عملگر و عبارت محاسباتی را تعریف کند و آنها را به درستی به کارگیرد.
- ۲- عملگرهای ریاضی را نام ببرد و در عبارت محاسباتی از آنها استفاده کند.
- ۳- حاصل عبارت محاسباتی را با استفاده از جدول تقدم عملگرها به دست آورد.
- ۴- کاربرد عملگرهای افزایشی، کاهشی و انتساب را بیان نماید.

#### ۱-۵- عبارت<sup>۱</sup> چیست؟

در درس ریاضی با عبارت‌های محاسباتی مختلفی مانند عبارت‌های زیر آشنا شدید.

$$8 \quad 3*5$$

$$9 - 7.25$$

$$5.6*3 \quad 5.6$$

$$\frac{y}{6} \quad 25 \quad x$$

در این عبارت‌ها، علامت نشانه عمل جمع و علامت \* نشانه عمل ضرب است. به این علامت‌ها که بیانگر انجام یک عمل بر روی اعداد و داده‌ها هستند، عملگر<sup>۱</sup> گفته می‌شود. مثلاً عملگر \* در عبارت  $3*5$  ۳ و ۵ عمل ضرب را انجام می‌دهد و همچنین عملگر در عبارت بالا بر روی عدد ۸ و نتیجه حاصلضرب یعنی عدد ۱۵، عمل جمع را انجام می‌دهد. به اعدادی که یک عملگر بر روی آنها عملی را انجام می‌دهد عملوند<sup>۲</sup> می‌گویند. اعداد ۳ و ۵ عملوندهای عملگر ضرب و عدد ۸ و ۱۵ عملوندهای عملگر جمع هستند.

هر یک از عملگرهای ضرب و جمع بر روی دو عدد عمل می‌کنند و به عبارتی دارای دو عملوند هستند به این عملگرهای، عملگرهای دوتایی<sup>۳</sup> گفته می‌شود. عملگر در عبارت  $7.25 \times 9$  عملگر تفریق است که آن نیز یک عملگر دوتایی است و حاصل تفریق ۷.۲۵ از ۹ را محاسبه می‌کند. اما عملگر قرینه در عبارت  $x$ ، فقط دارای یک عملوند  $x$  است و آن را قرینه می‌کند. این عملگر یک عملگر یکتایی<sup>۴</sup> است.

### نکته

یک عبارت از تعدادی عملگر و عملوند تشکیل شده است و دارای یک حاصل یا نتیجه می‌باشد. نتیجه یا حاصل یک عبارت ممکن است عددی یا غیرعددی باشد.

## ۲-۵- عملگرهای ریاضی یا حسابی<sup>۵</sup>

اگر در عبارتی بیش از یک عملگر وجود داشته باشد ابتدا عملگری عمل خود را انجام می‌دهد که اولویت<sup>۶</sup> بالاتری نسبت به دیگری داشته باشد. مثلاً اولویت عملگر ضرب بیش از اولویت عملگر جمع است. چنانچه دو یا چند عملگر دوتایی، با اولویت یکسان در یک عبارت وجود داشته باشد ابتدا عملگر سمت چپ انجام می‌شود. به عبارت دیگر از سمت چپ به راست، عملگرهای به ترتیب انجام می‌شوند که به آن «شرط پذیری چپ»<sup>۷</sup> می‌گویند.

در جدول ۵-۱ لیست عملگرهای ریاضی را به ترتیب اولویت مشاهده می‌کنید. عملگر قرینه اولویت بالاتری نسبت به بقیه عملگرهای ریاضی دارد و عملگرهای ریاضی دارند. ولی کمترین اولویت را در بین عملگرهای ریاضی دارند.

۱- Operator

۲- Operand

۳- Binary Operator

۴- Unary Operator

۵- Arithmetic

۶- Precedence

۷- Left-Associativity

## جدول ۱-۵- عملگرهای ریاضی

نوع عملگر	مثال	نشانه	نام عملگر	اولویت
یکتایی	-5	-	قرینه	۱
دو تابی	12 * 36	*	ضرب	
	25 4	/	تقسیم	۲
	23 % 5	%	باقیمانده تقسیم	
دو تابی	75 + 14	+	جمع	۳
	29 - 36	-	تفريق	

عملکرد عملگرهای جمع و تفريق و ضرب مانند عملکرد آنها در ریاضیات است اما عملگر تقسیم با توجه به نوع عملوند هایش می تواند تقسیم صحیح و بدون ممیز و یا تقسیم اعشاری و ممیزی انجام دهد. مثلاً در عبارت  $2 / 9$  چون عملوند ها اعداد صحیح هستند بنابراین تقسیم بدون ممیز و صحیح انجام خواهد شد که نتیجه آن عدد ۴ است . اما در عبارت  $2 / 0.9$  یا در عبارت  $2.0 / 9$  ، چون حداقل یکی از عملوند ها، اعشاری است بنابراین تقسیم به صورت اعشاری انجام می شود که حاصل عبارت عدد  $4 / 5$  است.

در جدول ۱-۵، عملگر جدیدی نیز به نام باقیمانده تقسیم که با نشانه٪ مشخص می شود، مشاهده می کنید. به وسیله این عملگر می توانیم باقیمانده تقسیم یک عدد بر عدد دیگر را با توجه به خارج قسمت صحیح و بدون اعشار به دست آوریم. مثلاً در تقسیم عدد  $23$  بر عدد  $5$ ، خارج قسمت بدون اعشار عدد  $4$  است بنابراین باقیمانده عدد  $3$  است.

$$23 / 5 \quad 4$$

$$23 \% 5 \quad 3$$

23	5
20	4
<hr/>	
3	

برای تغییر دادن اولویت عملگرهای از علامت های پرانتز استفاده می شود. مثلاً در عبارت زیر ابتدا عمل جمع و سپس عمل ضرب انجام می شود.

$$5.6 * (3 \quad 6.5)$$

اگر چند پرانتز تو در تو نیز وجود داشته باشد، ابتدا داخلی ترین پرانتز انجام می شود.

در جدول ۲-۵، چند نمونه از عبارت‌های ریاضی نشان داده شده است.

جدول ۲-۵—مثالی از عبارت‌های ریاضی

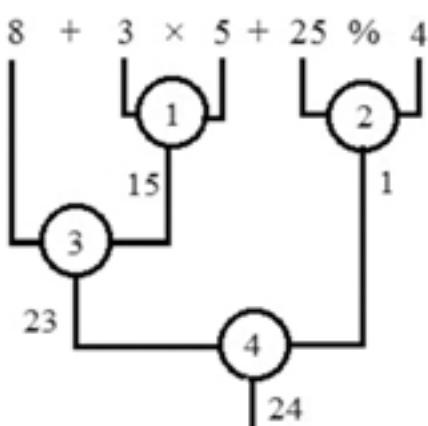
عبارت	حاصل عبارت	نوع عبارت
$175 / 31$	5	صحیح
$175 \% 31$	20	صحیح
$75 / 2$	3.75	اعشاری
$75 \% 2$	1.5	اعشاری
$36 / 20$	18.0	اعشاری
$36 \% 0.2$	0	صحیح
$2541389 / 10$	254138	صحیح
$2541389 \% 10$	9	صحیح

عبارت محاسباتی زیر را در نظر می‌گیریم و سپس نوع آن را تعیین می‌کنیم.

$$8 \ 3 * 5 \ 25 \% 4$$

در این عبارت بیش از یک عملگر وجود دارد، بنابراین ابتداعملگری که دارای اولویت بالاتر است، انجام می‌شود. چون اولویت عملگرهای  $*$  و  $\%$  بالاتر از عملگر  $+$  است بنابراین ابتدا این دو

عملگر انجام می‌شود و از طرفی چون این دو عملگر دارای اولویت یکسان هستند، برطبق شرکت‌پذیری چپ، ابتدا عملگر سمت چپ یعنی  $*$  و سپس عملگر  $\%$  انجام می‌شود. بر همین اساس در مورد عملگرهای جمع نیز ابتدا عملگر سمت چپ و سپس عملگر بعدی انجام می‌شود (نمودار ۵-۱).



نمودار ۵-۱—ترتیب اجرای عملگرها

در برنامه‌ها، معمولاً حاصل با نتیجهٔ یک عبارت را در یک متغیر نگهداری می‌کنند. البته نوع متغیری که حاصل یک عبارت، در آن قرار می‌گیرد باید با نوع عبارت، سازگار باشد. مانند ظرفی در آشپزخانه که بخواهیم در آن غذا یا نوشیدنی بربزیم باید گنجایش مناسب آن غذا را داشته باشد.

قوانین زیر، باید به وسیلهٔ برنامه نویس، در هنگام انتساب یک عبارت به یک متغیر، رعایت شود، در غیر اینصورت با پیام خطای مترجم مواجه می‌شویم. مترجم زبان #C روى اين قوانين سخت گير<sup>۱</sup> است زيرا می‌خواهد از اشتباهاات برنامه نویسان جلوگيري نماید، اين يكى از ویژگي هاي زيان #C است.  
**۱** اگر حاصل یک عبارت عدد صحیح باشد بسته به اندازه و بزرگی عدد، می‌تواند در یک متغیر نوع صحیح که گنجایش آن مساوی یا بزرگ‌تر از حاصل عبارت باشد جای گیرد.  
مثالاً حاصل عبارت 31 / 175 عدد 5 است این عدد کوچک می‌تواند در تمام متغیرهای نوع صحیح زیر قرار گیرد.

sbyte , byte , short , ushort , int , uint , long , ulong

ولی در عبارت 10 / 2541389 چون حاصل عبارت عدد 254138 است که عدد صحیح بزرگی است فقط در متغیرهای نوع **ulong** ، **uint** ، **long** قابل نگهداری است.  
**۲** اگر حاصل یک عبارت از نوع صحیح باشد می‌تواند در یک متغیر نوع اعشاری نیز ذخیره شود، اما با این تفاوت که اعداد بزرگ (**long**) فقط با ۷ رقم دقت (در نوع **float**) و یا با ۱۵ رقم دقت (در نوع **double**) ذخیره می‌شود و بقیه ارقام عدد، گرد می‌شود.  
مثالاً در دستور زیر حاصل عبارت 10 / 1234567890 در متغیر اعشاری ذخیره می‌شود ولی به دلیل اینکه عدد بزرگی است به صورت گرد شده در متغیر ذخیره می‌شود (شکل ۲-۵).

float number = 1234567890 / 10;

Console.WriteLine("Number " + number);



شکل ۲-۵- نتیجه اجرای عملگر تقسیم

همان طور که در شکل ۵-۲ مشاهده می‌کنید مقداری که در متغیر number وجود دارد به صورت نماد علمی نشان داده می‌شود که اگر آن را به صورت معمولی تبدیل کنیم خواهیم داشت :

$$1.234568E\ 08 \rightarrow 1.234568 \times 10^8 \quad 123456800$$

اگر این مقدار را با حاصل عبارت اولیه مقایسه کنیم خواهیم دید که عدد با ۷ رقم گرد شده است :

حاصل عبارت 123456789

مقدار ذخیره شده در متغیر 123456800

۳- اگر حاصل یک عبارت از نوع اعشاری باشد **نمی‌تواند** به طور ضمنی، در یک متغیر نوع صحیح جای داده شود، فقط می‌تواند در یک متغیر اعشاری (نوع float و یا double) جای گیرد.

۴- اگر حاصل یک عبارت از نوع اعشاری double باشد، فقط در متغیر نوع double می‌تواند جای می‌گیرد. **چرا؟**

حاصل تقسیم یک عدد اعشاری بر یک عدد صحیح، عددی اعشاری است و مترجم آن را از نوع double در نظر می‌گیرد. تقسیم زیر را در نظر بگیرید :

219.5 / 14

با توجه به اینکه عدد 219.5 اعشاری است، بنابراین تقسیم نوع اعشاری انجام می‌شود و نتیجه عبارت از نوع double خواهد بود.

برای نگهداری نتیجه این عبارت، باید ابتدا متغیر مناسبی را تعریف کنیم و سپس با دستور انتساب حاصل عبارت را در آن ذخیره کنیم. با توجه به اینکه حاصل عبارت از نوع double است بنابراین باید متغیر نیز اعشاری و از نوع double باشد.

دستور زیر متغیری برای ذخیره معدل داش آموز تعریف می‌کند تا نتیجه تقسیم را در آن ذخیره نماید.

Double meanScore;

اکنون می‌توانید مقدار عبارت را در متغیر مذبور با استفاده از دستور انتساب مقداردهی کنید.

Double meanScore 219.5 / 14 ;

جایگزینی دو دستور بالا با یک دستور به صورت زیر خواهد بود :

Double meanScore 219.5 / 14 ;

با اجرای دستور قبل عمل تقسیم انجام شده و حاصل تقسیم یعنی عدد 14.6786714285714

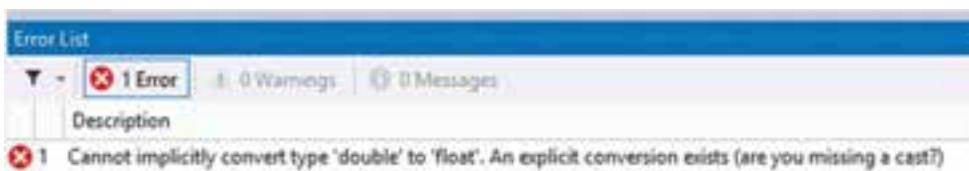
در متغیر meanScore ذخیره می‌شود.

## سؤال: آیا لازم است معدل نمرات با چنین دقتی (۱۵ رقم) ذخیره شود؟

با توجه به اینکه معمولاً معدل نمرات با دو رقم اعشار بیان می‌شود، بنابراین در این موارد بهتر است از متغیر نوع float استفاده کنیم. در این صورت لازم است نوع عبارت محاسبه معدل نیز float باشد. برای اینکه نوع عبارت float شود می‌توانید از مترجم بخواهید عدد 219.5 را یک عدد نوع float در نظر بگیرد بدین منظور حرف f یا F را بعد از عدد می‌نویسیم 219.5 F بنابراین دستور زیر را می‌نویسیم:

```
float meanScore = 219.5f / 14;
```

با اجرای دستور بالا، عدد ۱۵/۶۷۸۵۷ در متغیر meanScore ذخیره می‌شود. اگر فراموش کنید که در دستور بالا، حرف f را بنویسید مترجم خطأ می‌دهد. چون در سمت راست علامت اتساب، یک عبارت از نوع داده double است ولی در سمت چپ یک متغیر از نوع float است که ظرفیت کمتری نسبت به double دارد. در چنین حالتی در محیط VS یک خط قرمز زیر عبارت کشیده می‌شود و مترجم خطای را صادر می‌کند (شکل ۳-۵).



شکل ۳-۵-خطای تبدیل نوع double به نوع float

شکل ۳-۵ بیان می‌کند که:

«مترجم نمی‌تواند نوع داده double را به طور ضمنی و خودکار به نوع float تبدیل کند.» باید به طور صریح و واضح از مترجم بخواهید عمل تبدیل نوع را انجام دهد.

## نکته

اگر در برنامه‌ای با عبارت‌ها و اعداد اعشاری با دقت حداقل ۷ رقم سر و کار دارید و می‌خواهید از متغیرهای نوع float استفاده کنید، باید پس از هر عدد اعشاری، یک حرف f یا F قرار دهید تا مترجم آن عدد را به عنوان نوع float در نظر بگیرد. اما اگر از متغیرهای نوع double استفاده می‌کنید دیگر نیازی به نوشتن حرف f نیست.

```
float meanMark = 219.5f / 14;
```

## کار در کارگاه

مثال ۱-۵: مجموع و معدل نمرات درسی را طبق جدول ۳-۵ محاسبه کنید.

جدول ۳-۵- نمرات درسی یک دانشآموز

نام درس	نمره درس	تعداد واحد <sup>۱</sup>
فیزیک	۱۷/۵	۳
شیمی	۱۹	۲
ریاضی	۱۴/۵	۴
ادبیات	۱۸	۲
ورزش <sup>۲</sup>	۱۹	۱

به دلیل این که می خواهیم روی داده ها پردازش انجام دهیم بهتر است ابتدا نمرات را داخل متغیرهای مناسب ذخیره کنیم :

`float physicMark = 17.5F, chemistryMark = 19, mathMark = 14.5F;`

`float literacyMark = 18, PEMark = 19;`

به همین صورت تعداد واحدها را نیز داخل متغیرهای مناسب قرار می دهیم :

`int physicCredit = 3, chemistryCredit = 2, mathCredit = 4;`

`int literacyCredit = 2, PECredit = 1;`

حال می خواهیم مجموع نمرات را به دست آوریم. با توجه به اینکه تعداد واحد هر درس مختلف است نمی توانیم به سادگی نمرات را با یکدیگر جمع کنیم بلکه باید هر نمره را در تعداد واحد درسی مربوطه ضرب کنیم و سپس حاصل ضرب های به دست آمده را با یکدیگر جمع کنیم و نتیجه را در یک متغیر قرار دهیم. بنابراین متغیری به نام `totalMark` برای این منظور تعریف و حاصل عبارت محاسباتی مربوطه را در آن قرار می دهیم :

`float totalMark ;`

`totalMark = (physicMark * physicCredit)`

`(chemistryMark * chemistryCredit)`

`(mathMark * mathCredit)`

`(literacyMark * literacyCredit)`

`(PEMark * PECredit);`

## سؤال:

این عبارت محاسباتی نیازی به پرانتز گذاری ندارد. چرا؟ در اینجا فقط برای بالا بدن خوانایی عبارت یا برنامه از پرانتزها استفاده کرده‌ایم.

پس از به دست آوردن مجموع نمرات که در متغیر totalMark، قرار گرفته است، مجموع واحدها را نیز حساب می‌کنیم و در یک متغیر به نام totalCredit قرار می‌دهیم.

```
int totalCredit = physicCredit + chemistryCredit + mathCredit + literacyCredit + PECredit;
```

در این لحظه می‌توانیم با تقسیم مجموع نمرات بر مجموع واحدها، معدل را به دست آوریم و

در یک متغیر ذخیره کنیم.

```
float average = totalMark / totalCredit;
```

برای نمایش مقادیر متغیرها از دستورات زیر استفاده می‌کنیم :

```
Console.WriteLine("Total mark: " + totalMark);
```

```
Console.WriteLine("Total credit: " + totalCredit);
```

```
Console.WriteLine("Average: " + average);
```

با توجه به توضیحات بالا برنامه مورد نظر چنین خواهد بود :

```
using System;
```

```
class Average
```

```
{
```

```
    static void Main()
```

```
{
```

```
    float physicMark = 17.5F, chemistryMark = 19, mathMark = 14.5F;
```

```
    float literacyMark = 18, PEMark = 19;
```

```
    int physicCredit = 3, chemistryCredit = 2, mathCredit = 4;
```

```
    int literacyCredit = 2, PECredit = 1;
```

```
    float totalMark;
```

```
    totalMark = (physicMark * physicCredit)
```

```
        (chemistryMark * chemistryCredit)
```

```
        (mathMark * mathCredit)
```

```
        (literacyMark * literacyCredit)
```

```
        (PEMark * PECredit);
```

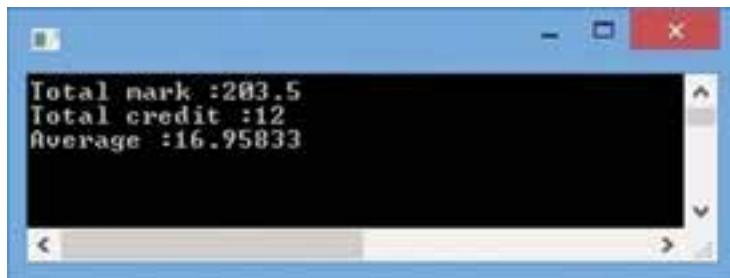
```

int totalCredit = physicCredit + chemistryCredit + mathCredit + literacyCredit
+ PECredit;
float average = totalMark / totalCredit;
Console WriteLine("Total mark: " + totalMark);
Console WriteLine("Total credit: " + totalCredit);
Console WriteLine("Average: " + average);
Console ReadKey();
}
}

```

برنامه ۱\_۵—محاسبه معدل نمرات یک داش آموز

خروجی برنامه چنین است :



شکل ۴\_۵—خروجی برنامه ۱\_۵

به مثال ۱\_۵ دستوراتی اضافه کنید که نمرات درسی را از ورودی دریافت کند. این برنامه را در محیط VS بنویسید و آن را ذخیره، ترجمه و اجرا نمایید.

### ۳-۵- عملگرهای افزایشی و کاهشی<sup>۱</sup>

در زبان برنامه نویسی #C، عملگرهای یکتایی و -- به ترتیب برای افزایش و کاهش مقدار یک متغیر به اندازه یک واحد در نظر گرفته شده است. البته این عملگرها از زبان C وارد این زبان شده اند و برای کوتاه شدن و تایپ کمتر دستورات، ابداع شده اند که در زبان های دیگر امروزی مانند جاوا نیز وجود دارند (جدول ۴-۵).

جدول ۴-۵- (عملگرهای افزایشی و کاهشی)

نام عملگر	نوع عملگر	نشانه	مثال
افزایش	یکتایی	+	value یا value
کاهش	یکتایی	-	value یا value

اکنون می توانیم اضافه کردن یک نمره به درس ریاضی را با استفاده از عملگر افزایشی انجام

دهیم :

عملگر افزایشی قبل از نام متغیر قرار گرفته است // ++mathMark ;

یا

عملگر افزایشی بعد از نام متغیر قرار گرفته است // mathMark++ ;

عملگر افزایشی یا کاهشی را می توانید قبل از نام متغیر و یا بعد از نام متغیر ذکر کنید که در هر دو حالت سبب می شود مقدار متغیر به اندازه یک واحد تغییر کند. تفاوت بین این دو حالت را در سؤالات ۱۱ و ۱۲ خودآزمایی ببینید.

## ۴-۵- عملگر های انتساب

در جدول ۵-۵ عملگر های انتساب دیده می شود.

نمانه	نوع عملگر
,	انتساب
,	انتساب
*	انتساب
/	انتساب
%	انتساب
&	انتساب
,	انتساب
,	انتساب
^	انتساب
<<	انتساب
>>	انتساب

با عملگر و کاربرد آن در فصل قبل آشنا شدید که برای ذخیره کردن یک مقدار در یک متغیر استفاده می شود. دستورات زیر را در نظر بگیرید.

```
int x ,y ;  
x = 6;  
string helloString = "Hello World";  
y = x;
```

در دستور آخر، مقداری که در متغیر x قرار دارد (عدد ۶) به متغیر y منتسب می شود و y نیز برابر ۶ می شود.

### نکته

حاصل عبارتی که دارای عملگر انتساب است، مقدار داده یا متغیری است که در سمت راست عملگر واقع شده است.

x = 6; یعنی x ————— 6

مثالاً در دستور زیر

```
int z = x = 9;
```

چند عملگر انتساب وجود دارد و از سمت راست به ترتیب انجام می شوند. ابتدا عدد ۹ در متغیر x قرار می گیرد و سپس حاصل عبارت که عدد ۹ است در متغیر y کپی شده و سپس این مقدار در متغیر z کپی می شود. یعنی در عبارتی که عملگر های انتساب وجود دارد این عملگرها از سمت راست به چپ انجام می شوند (شرکت پذیر راست) بر خلاف عملگر های ریاضی مشابه که از چپ به راست انجام می شونند.

```
int z = y = x = 9;
```

اگر بخواهیم یک نمره، به نمره درس ریاضی در مثال ۱-۵ اضافه کنیم، با توجه به اینکه نمره هر درس را در یک متغیر ذخیره کردیم، دستور لازم برای افزایش یک واحد به متغیر مربوطه را بنویسید.

mathMark 1

متغیر mathMark حاوی نمره درس ریاضی است. برای افزایش یک نمره از دستور انتساب

زیر استفاده می‌کنیم :

mathMark mathMark 1 ;

با اجرای این دستور انتساب، ابتدا کامپیوتر عبارت سمت راست علامت را محاسبه می‌کند.

بدین منظور محتوای متغیر mathMark که برابر با  $\frac{۱۴}{۵}$  است با عدد یک جمع می‌شود و حاصل عبارت یعنی  $\frac{۱۵}{۵}$  را در متغیر سمت چپ، جایگزین مقدار قبلی آن می‌کند. بنابراین از این به بعد محتوای متغیر مذبور  $\frac{۱۵}{۵}$  خواهد بود که به این ترتیب یک واحد افزایش یافته است.

با دستوری مشابه، می‌توانیم مقدار یک متغیر را کاهش دهیم مثلاً اگر بخواهید دو واحد از

مجموع تعداد واحدها کم کنید دستور زیر را می‌نویسیم :

totalCredit totalCredit 2 ;

در این دستور نیز ابتدا عبارت سمت راست محاسبه و نتیجه در متغیر سمت چپ قرار می‌گیرد.

در جدول ۵-۵، علاوه بر عملگر ، عملگرهای دیگر انتساب همراه با یک عملگر ریاضی وجود دارند. عملگرهای و و \* هر یک، علاوه بر عمل انتساب، یک عمل ریاضی را نیز روی یک عملوند مشترک انجام می‌دهند تا تایپ دستورات خلاصه شود.

فرض کنید بخواهید محتوای متغیر x را سه برابر کنید در این صورت از دستور زیر استفاده می‌کنید :

x \*= 3;

**سؤال:** معادل دستور ;\*= 3 x چه دستوری است؟

**مثال ۲-۵:** با اجرای دستورات زیر چه عددی بر روی خروجی نشان داده می‌شود؟

int x = 3;

int y = 4;

x \*= y;

Console.WriteLine(x);

در دستور ; y \*= x حاصل ضرب متغیر x در متغیر y محاسبه می‌شود (برابر ۱۲) و در متغیر

x قرار می‌گیرد. بنابراین عدد ۱۲ در روی صفحه نمایش داده می‌شود.

**سؤال:** معادل دستور ; y \*= x چه دستوری است؟

## فودآزمایی فصل پنجم

۱- عبارت، عملگر و عملوند را با ذکر یک مثال مشخص کنید.

۲- عبارت ریاضی  $\frac{5x}{y+3} + 4x$  را با توجه به جدول ۱-۵ (اولویت عملگرها) به صورت یک عبارت قابل قبول در #C، بنویسید.

۳- مترجم در ترجمه کدام یک از دستورات زیر، خطای می دهد؟ دلیل خود را ذکر کنید.

- 1) `int number 127.9;`
- 2) `float number 12 2.3;`
- 3) `ushort myMark 1300;`
- 4) `sbyte totalScore 250;`

۴- اگر در متغیر `number` یک عدد صحیح قرار داشته باشد، دستوری بنویسید که رقم یکان آن را بدست آورده و در خروجی نشان دهد.

۵- در دستورات زیر، نمودار اجرای عملگرها را مانند شکل ۱-۵ در ابتدای این فصل، رسم کنید و حاصل را به دست آورید.

```
int n 600 10 * 9 % 4;  
System.Console.WriteLine( 12 * 2 % 8 - 3 * 2 );
```

۶- برای انجام عبارت‌های ستون سمت راست، از چه عملگری در ستون چپ استفاده می‌کنید؟

- |                                |     |        |
|--------------------------------|-----|--------|
| ۱- کاهش ۶ واحد از یک متغیر     | %   | - الف- |
| ۲- افزایش یک واحد به یک متغیر  | - ب | -      |
| ۳- کاهش یک واحد از یک متغیر    | - ج | -      |
| ۴- افزایش n واحد به یک متغیر   | - د | -      |
| ۵- محاسبه رقم یکان یک عدد صحیح | - ه | -      |

۷- خروجی قطعه کد زیر را به دست آورید. اعداد نشان داده شده در خروجی، مانند چه دنباله‌ای از اعداد در ریاضیات است؟

```
n 0;  
d 4;  
a 5;  
Console.WriteLine(a d * n);  
n 1;  
Console.WriteLine(a d* n);  
n 1;  
Console.WriteLine( a d * n );  
n 1;  
Console.WriteLine( a d * n );
```

در متدهای `WriteLine()` بالا، با چه فرضی فضای نامی `System` در ابتدای هر خط، ذکر شده است؟

- ۸- اگر متغیر `Z` را یک بار با عدد ۵ و بار دیگر با عدد ۳ مقداردهی کنیم،  
دستور `Console.WriteLine(z)` چه مقداری را در نهایت چاپ می‌کند؟ توضیح دهید؟
- ۹- برنامه‌ای بنویسید که دو عدد از ورودی دریافت نماید و مقادیر درون دو متغیر را جابجا کند.  
(از مثال جا بجایی محتوای دو لیوان حاوی شیر و نوشابه کمک بگیرید)
- ۱۰- دستورات زیر بر روی مقادیر متغیرهای `a,b` چه عملی را انجام می‌دهد؟  
پاسخ را در جدول `Trace` وارد کنید.

```
int a 5;  
int b 10;  
a a b;  
b a b;  
a a b;
```

a	b
5	10
a	b
10	5
a	b

۱۱- نتیجه اجرای دستورات زیر چیست؟

```
int a = 5;  
int b = a;  
Console.WriteLine(a);  
Console.WriteLine(b);
```

۱۲- نتیجه اجرای دستورات زیر چیست؟ نتیجه این تمرین را با تمرین قبلی مقایسه کنید.

```
int a = 5;  
int b = a ;  
Console.WriteLine(a);  
Console.WriteLine(b);
```

۱۳- سوال زیر به زبان انگلیسی است آن را خوانده و پاسخ‌های صحیح را مشخص کنید.

Which of the following is the correct way to initialize the variables i and j to a value 10 each?

- A) int i = 10;  
    int j = 10;
- B) int i = 10, j = 10;
- C) int i ,j ;  
    i = 10; j = 10;
- D) int i,j = 10;
- E) int i = j = 10;

## تمرینات برنامه‌نویسی فصل پنجم

۱- نیروی جاذبه در کره ماه  $\frac{1}{4}$  نیروی جاذبه در زمین است. برنامه‌ای بنویسید که وزن یک شخص در روی زمین را سؤال کرده و سپس وزن وی در کره ماه را حساب کند.

۲- اگر محیط<sup>۱</sup> یک مربع<sup>۲</sup> برابر ۱۸ سانتی متر باشد برنامه‌ای بنویسید که اطلاعات زیر را روی

صفحه نشان دهد :

الف) اندازه هر ضلع<sup>۳</sup> مربع بر حسب سانتی متر

ب) اندازه مساحت<sup>۴</sup> مربع بر حسب سانتی متر مربع

۳- برنامه‌ای بنویسید که محیط یک مربع را سؤال کند و سپس اندازه ضلع و اندازه مساحت مربع را محاسبه نماید.

۴- سال ۱۴۰۰ شمسی را در نظر بگیرید. تعداد روزهای سپری شده تا این سال چند روز است؟

تعداد ساعت‌های سپری شده چقدر است؟ برنامه‌ای بنویسید که شماره سال را از کاربر سؤال کند و سپس تعداد روز و ساعت سپری شده را در متغیرهای مناسبی قرار دهد و سپس نمایش دهد.

۵- یک فروشگاه پوشک، اجناس خود را با ۱۵ درصد تخفیف، حراج کرده است، برنامه‌ای بنویسید که مبلغ قبل از تخفیف جنس را از ورودی دریافت کند و سپس قیمت بعد از تخفیف را محاسبه و نمایش دهد.

۶- برای یک دستگاه خودپرداز برنامه‌ای بنویسید که یک عدد را به عنوان مبلغ یک اسکناس دریافت کند و آن را به اسکناس‌های کوچکتر خرد<sup>۵</sup> کند. مثلاً بتواند یک اسکناس ۱۰۰ هزار ریالی را به دو اسکناس ۵۰۰۰۰ ریالی و یا ۱۰ اسکناس ۱۰۰۰۰ ریالی خرد کند.

(راهنمایی : از عملگرهای % و / استفاده کنید.)



۱- Per meter

۲- Square

۳- S de

۴- Area

۵- Change

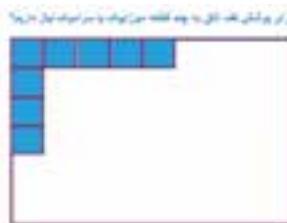
- ۷- برنامه‌ای بنویسید که مقلوب یک عدد صحیح دو رقمی دریافتی از کاربر را نمایش دهد.
- ۸- برنامه‌ای بنویسید که تاریخ تولد هر فرد را بر حسب ما، روز و سال دریافت نماید و سپس تعداد روزهای عمر وی را حساب کند. هر سال  $365$  روز و هر ماه  $30$  روز است.

۹- (این سوال نیاز به برنامه‌نویسی ندارد و فقط یک محاسبه ساده است.)



می‌خواهیم کف یک اتاق  $3 \times 4$  متری را سرامیک کنیم. اگر هر قطعه سرامیک به شکل مربع و به اندازه  $50 \times 50$  سانتی متر باشد به چند قطعه سرامیک نیاز داریم؟  
اگر اندازه آن برابر  $40 \times 40$  سانتی متر باشد آن وقت به چه تعداد سرامیک نیاز داریم؟

- ۱۰- برنامه‌ای به نام Simple Tiling Calculator بنویسید که اندازه طول (Width) و عرض (Length) یک اتاق را پرسد و سپس اندازه یک سرامیک مربع شکل را سؤال نماید. تعداد سرامیک‌های لازم برای پوشش کف را محاسبه و تعیین کند. توجه داشته باشید که نتیجه به دست آمده از محاسبات باید عدد صحیح باشد. مثلاً عددی مانند  $17/25$  باید به عدد  $18$  گرد شود.



- ۱۱- مدیر یک ساختمان مسکونی قصد دارد، هزینه برق عمومی ساختمان، را بر اساس تعداد نفرات ساکنین تقسیم کند. تعداد نفرات هر خانواده طبق جدول زیر است، برنامه‌ای بنویسید تا به مدیر ساختمان در بدست آوردن هزینه برق هر خانواده کمک کند. برنامه باید یک عدد به عنوان هزینه برق از کاربر دریافت کند و سپس با استفاده از اطلاعات جدول، هزینه برق هر خانوار را محاسبه و نمایش دهد.

شماره واحد	تعداد نفرات خانواده
۱	۳
۲	۴
۳	۲
۴	۵
۵	۲



## فصل ششم

### دستورهای شرطی

در بیشتر برنامه‌های کاربردی، لازم است مقدار داده‌ها را بررسی و مقایسه کنیم. سپس بر اساس نتیجه حاصل از بررسی، دستور یا دستورهایی را اجرا کنیم. به عبارت دیگر، در برنامه‌ها لازم است بتوانیم بر اساس مقدار داده‌ها، تصمیم گیری کنیم. برنامه‌هایی که تاکنون نوشته‌ایم تمام دستورهای داخل متدهای Main()، پشت سرهم و به نوبت اجرا می‌شوند. اکنون در این فصل با دستورهای شرطی آشنا می‌شویم که به وسیله آنها اجرای دستورها و پردازش آنها، کنترل می‌شوند و رفتار برنامه بر اساس وضعیت داده‌ها تغییر می‌کند.

#### پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند:

- ۱- انواع عملگرهای مقایسه‌ای را توضیح دهد و آنها را در برنامه استفاده کند.
- ۲- ساختار دستور if-else را در برنامه‌های شرطی خود استفاده کند.
- ۳- مفهوم عملگرهای منطقی را شرح دهد و انواع آنها را نام ببرد.
- ۴- از عملگرهای منطقی به طور صحیح در برنامه‌ها استفاده نماید.
- ۵- با استفاده از جدول عملگرهای منطقی، نتیجه عبارتها را ترکیبی منطقی و مقایسه‌ای را به دست آورد.
- ۶- اولویت عملگرها را با استفاده از جدول تقدم عملگرها در حل عبارات ترکیبی به کار بندد.
- ۷- دستور switch را در برنامه‌های خود به کار گیرد.
- ۸- با ترکیب دستورهای شرطی، برنامه‌هایی با ساختار پیچیده شرطی بنویسد.

## ۱-۶\_ عبارت منطقی یا بولین<sup>۱</sup>

در فصل چهارم در قسمت معرفی انواع داده‌ها با نوع داده منطقی، آشنا شدید. این نوع داده، فقط دارای دو مقدار true (درست) یا false (نادرست) است. عبارت منطقی نیز عبارتی است که حاصل آن فقط یکی از دو مقدار true یا false است.

مثالاً عبارت  $12 > 10$  یک عبارت منطقی است، که نتیجه آن false است چون  $12 > 10$  نیست. ولی در عبارت  $0 > x$  به شرط اینکه  $x$  عددی مثبت باشد نتیجه عبارت true است.

در دو مثال بالا از عملگرهای مقایسه‌ای < یا > استفاده شده است، عملگرهای مقایسه‌ای دیگری نیز وجود دارند که در جدول ۱-۶، مشاهده می‌شود. این عملگرها شبیه عملگرهایی است که در ریاضیات استفاده می‌شود. معمولاً در عبارت‌های منطقی از عملگرهای مقایسه‌ای استفاده می‌شود.

## ۲-۶\_ عملگرهای مقایسه‌ای<sup>۲</sup>

همان طور که از نام این عملگرها مشخص است، برای مقایسه داده‌ها استفاده می‌شوند و نتیجه آنها یکی از دو مقدار true یا false است. عملگرهای مقایسه‌ای می‌توانند دو عدد صحیح یا اعشاری و یا دو داده کاراکتری و یا رشته‌ای را با یکدیگر مقایسه کنند.

علامت بعضی از عملگرهای مقایسه‌ای در زبان C#، با علامت ریاضی آنها کمی متفاوت است مثلاً برای بررسی مساوی یا برابر بودن دو مقدار، از علامت استفاده می‌شود (دو بار علامت =) و یا برای بررسی مخالف یا نابرابر بودن دو مقدار، از علامت ! استفاده می‌شود.

در جدول ۱-۶ هر یک از عملگرهای مقایسه‌ای به همراه علامت ریاضی آنها نشان داده شده است.

۱\_ Boolean Expressions

۲\_ Comparison Operators

## جدول ۱-۶\_عملگرهای مقایسه‌ای

مثال	نشانه عملگر در ریاضی	عملگر در زبان C#	نام عملگر
delta = 0	=	==	برابری
name ! “AMIN”	≠	!=	نامساوی
max < number	<	<	کوچکتر
x < a	≤	<=	کوچکتر یا مساوی
temperature > 25	>	>	بزرگتر
(a b) > c	≥	>=	بزرگتر یا مساوی

مثال ۱-۶ : در برنامه ۱-۶ حاصل چند عبارت منطقی بر روی صفحه نشان داده می‌شود :

```

using System;
class Expression
{
    static void Main()
    {
        int weight = 700;
        Console.WriteLine(weight > 500); // True
        char gender = 'm';
        Console.WriteLine(gender == 'f'); // False
        double colorWaveLength = 1.630;
        Console.WriteLine(colorWaveLength > 1.621); // True
        Console.WriteLine('B' > 'A'); // True
        Console.ReadKey();
    }
}

```

برنامه ۱-۶\_عبارت‌های منطقی

حاصل عبارت‌های منطقی را می‌توانید بر روی صفحه خروجی نمایش دهید و یا در داخل متغیرهایی از نوع bool ذخیره کنید.

### ۳-۶- دستور شرطی<sup>۱</sup> if

در درس مبانی کامپیوتر، با دستور شرطی «اگر» آشنا شدید و به وسیله آن می‌توانستید اجرای دستورها را کنترل کنید. در زبان برنامه‌نویسی C# از دستور if (با حروف کوچک نوشته می‌شود) برای کنترل اجرای دستورها و بررسی شرط، استفاده می‌شود. ساختار کلی دستور if به صورت زیر است :

;if (عبارت منطقی)

; دستور

دستور شرطی if از سه بخش تشکیل شده است : کلمه رزرو شده if ، عبارت منطقی داخل پرانتز و دستوری که در صورت درست (true) بودن نتیجه عبارت، اجرا خواهد شد. توجه داشته باشید که پس از کلمه if، یک جفت پرانتز وجود دارد و عبارتی از نوع منطقی که برای بررسی و مقایسه داده است، داخل پرانتز نوشته می‌شود. در خط بعدی، دستوری که می‌خواهیم در صورت درست بودن عبارت منطقی اجرا شود، با تورفتگی می‌نویسیم و در انتهای آن علامت ; رابه منزله پایان دستور if قرار می‌دهیم.

در دستورات زیر نمونه‌ای از به کار گیری دستور if را می‌بینید.

if (mark < 10)

Console.WriteLine("Failed");

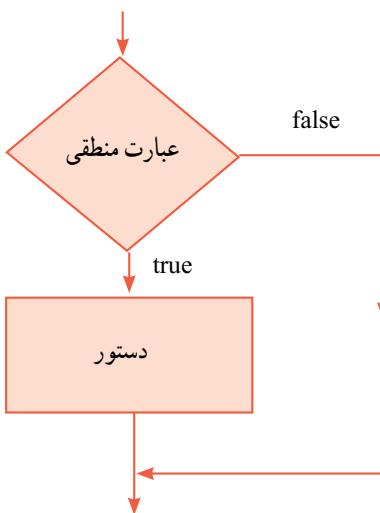
**سؤال:** اگر در متغیر mark، نمره ۸/۵ قرار داشته باشد، به نظر شما با اجرای دستورهای بالا، چه پیامی بر روی صفحه نمایش، ظاهر می‌شود؟ برای نمره ۱۸ چطور خواهد بود؟

به علامت نقطه ویرگول در دستور if توجه کنید. بعد از علامت پرانتز علامت نقطه ویرگول نگذارید، زیرا دستور if هنوز تمام نشده است. بلکه علامت نقطه ویرگول باید در انتهای دستور نوشته شود.  
نقطه ویرگول ندارد

if (عبارت منطقی)

; دستور

فلوچارت دستور if ساده به صورت زیر است :

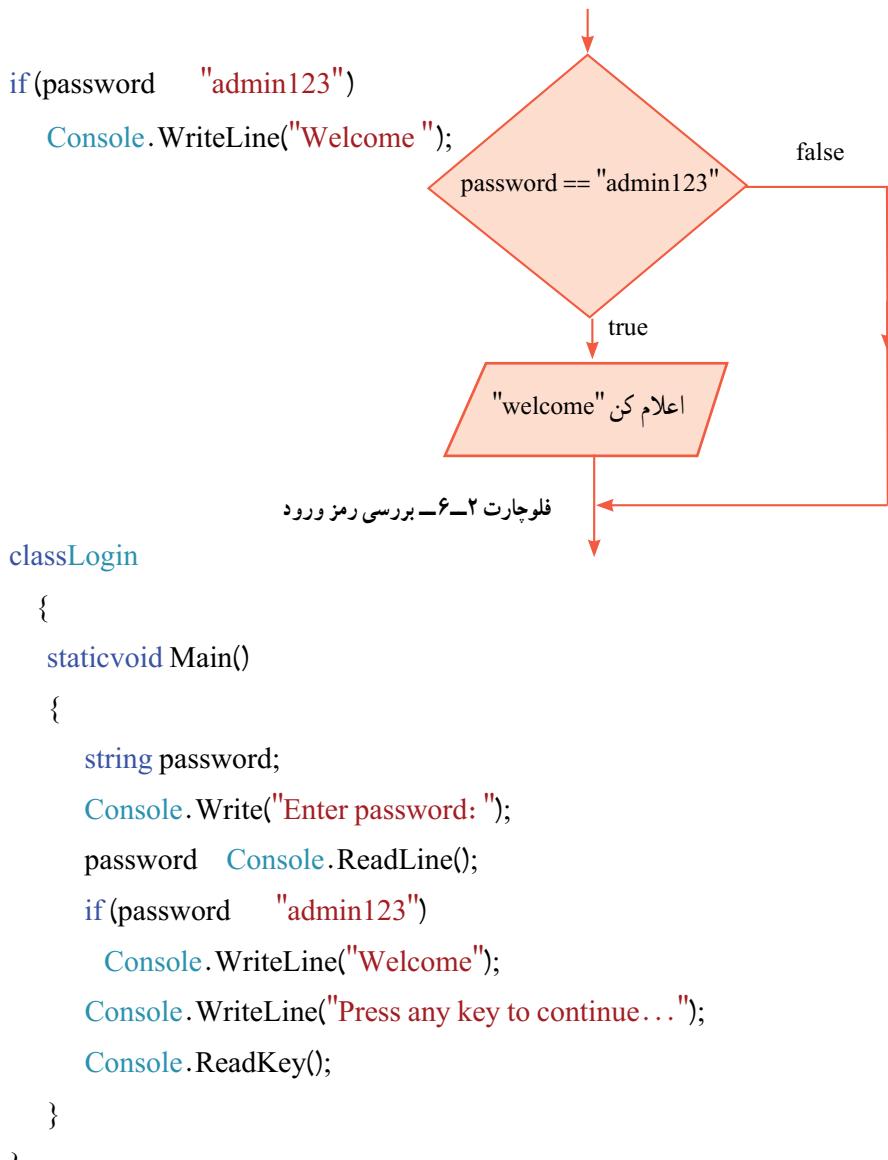


فلوچارت ۱-۶- دستور if

وقتی که کامپیوتر، در حال اجرای برنامه است با رسیدن به دستور if، ابتدا مقدار عبارت را محاسبه می‌کند. در صورتی که ارزش عبارت true باشد، دستوری که بعد از if قرار دارد، اجرا می‌شود و در صورتی که مقدار عبارت false باشد، دستور مربوطه اجرا نمی‌شود.

**مثال ۶-۲ :** می خواهیم برنامه ای بنویسیم که رمز ورود را از کاربر دریافت کند و در صورت صحیح بودن پیام مناسب چاپ نماید (رمز صحیح admin123).

الگوریتم یا روش انجام کار : ابتدا رمز را از کاربر دریافت کرده و در متغیر password قرار می دهیم. اگر رمز وارد شده مساوی admin123 بود پیام Welcome را نمایش می دهیم. دستور شرطی این برنامه در زیر آورده شده است.

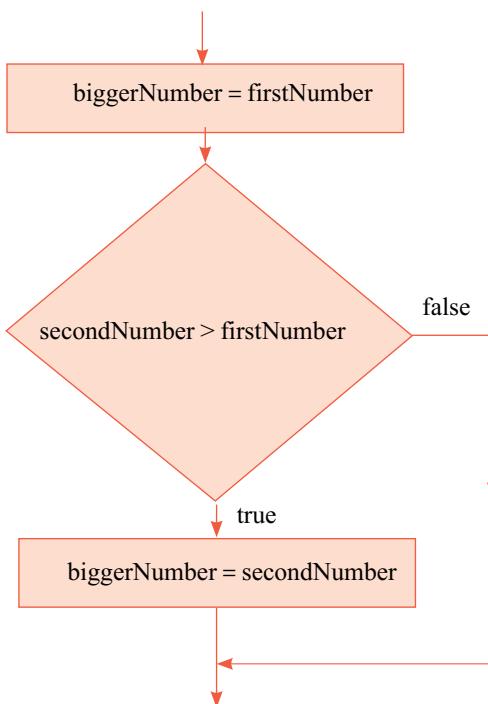


## کار در کارگاه ۱

۱- برنامه ۳-۶ را در محیط VS بنویسید، ترجمه و اجرا نمایید و به ازای ورودی‌های مختلف برنامه را آزمایش کنید.

۲- مثال ۳-۶: می‌خواهیم برنامه‌ای بنویسیم، که عدد بزرگتر را از بین دو عدد دریافتی کاربر، پیدا کند.

الگوریتم یا روش انجام کار: ابتدا اعداد مورد نظر را از کاربر دریافت کرده و در دو متغیر firstNumber و secondNumber قرار می‌دهیم. برای پیدا کردن بزرگ‌ترین عدد، ابتدا اولین عدد را به عنوان biggerNumber می‌گیریم. اگر این عدد همچنان بزرگ‌تر بود نیازی به انجام کاری نیست اما اگر عدد دوم، بزرگ‌تر بود، لازم است فرض اولیه خود را تصحیح کنیم و عدد دوم را به عنوان عدد biggerNumber در نظر بگیریم. به این ترتیب در متغیر biggerNumber عدد دوم را قرار می‌دهیم. در انتها نیز، مقدار موجود در biggerNumber را نمایش می‌دهیم. اگر بیش از دو عدد در اختیار داشتیم و بخواهیم عدد بزرگ‌تر را پیدا کنیم، باز هم از همین روش یا الگوریتم می‌توانیم استفاده کنیم. فلوچارت ۳-۶ را مشاهده کنید.



فلوچارت ۳-۶- پیدا کردن عدد بزرگتر از بین دو عدد

```

class FindMaximum
{
    static void Main()
    {
        string input;
        int firstNumber , secondNumber ;

        Console.WriteLine("Enter a number: ");
        input = Console.ReadLine();
        firstNumber = int.Parse(input);

        Console.WriteLine("Enter another number: ");
        input = Console.ReadLine();
        secondNumber = int.Parse(input);

        int biggerNumber = firstNumber;
        if(secondNumber>firstNumber)
            biggerNumber = secondNumber;
        Console.WriteLine("The maximum number is: " + biggerNumber);

        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}

```

برنامه ۳—۶ برنامه تشخیص عدد بزرگتر از بین دو عدد

توجه داشته باشید که در نوشتن دستورهای بالا، همه دستورها به جز خط بعد از دستور if، زیر هم نوشته می شوند.

**سؤال:** به نظر شما چرا دستور biggerNumber = secondNumber; از ابتدای خط شروع نشده است و به اندازه یک tab به سمت داخل نوشته شده است؟

دلیل این کار، این است که می‌خواهیم نشان دهیم که این دستور، یک دستور عادی نیست بلکه اجرای این دستور، وابسته به ارزش عبارت منطقی دستور if است. بنابراین همواره سعی کنید دستور مربوط به دستور if را کمی داخل تر بنویسید تا برنامه خوانا و واضح باشد.

اگر چه رعایت نکردن این قاعده، تاثیری در ترجمه و اجرا ندارد و اگر حتی تمام دستورها را پشت سر هم و در یک خط طولانی نیز بنویسید باز برنامه، ترجمه می‌شود اما اشکال زدایی و یا گسترش برنامه بسیار سخت خواهد بود و درک برنامه نیز مشکل می‌شود.

**مثال ۴-۶ :** عددی صحیح، داخل متغیری به نام number قرار دارد و می‌خواهیم زوج بودن آن را پس از تشخیص، با پیامی مناسب اعلام کنیم.

الگوریتم یا روش انجام کار : می‌دانیم عددی زوج است که رقم یکان آن یکی از اعداد ۰ و ۲ و ۴ و ۶ و ۸ باشد. از طرفی می‌دانیم که عدد زوج بر ۲ بخش پذیر است یعنی باقیمانده تقسیم آن بر عدد ۲ صفر است. از یکی از این دو روش می‌توانیم به سادگی زوج بودن عدد را تشخیص دهیم. در این مثال، از روش بخش پذیری بر ۲ استفاده می‌کنیم. ابتدا باقیمانده تقسیم عدد بر ۲ را حساب می‌کنیم، اگر باقیمانده برابر صفر بود پیامی را روی صفحه نمایش می‌دهیم تا نشان دهد که عدد زوج است.

```
int remainder = number % 2;  
if (remainder == 0)  
    Console.WriteLine(number + " is Even.");
```

اگر در متغیر number، عددی زوج مانند ۱۶ باشد، حاصل باقیمانده تقسیم آن عدد بر ۲، صفر خواهد شد و در این صورت نتیجه عبارت منطقی در دستور if، مقدار true خواهد بود. بنابراین دستور نمایش پیام اجرا می‌شود و اعلام می‌کند که عدد زوج است.

اما اگر در دستورهای بالا، عددی که در متغیر number قرار دارد، عددی فرد مانند ۱۵ باشد، حاصل باقیمانده تقسیم آن عدد بر ۲ عدد یک خواهد شد و در این صورت نتیجه عبارت منطقی در دستور if، مقدار false خواهد بود. بنابراین دستور نمایش پیام، اجرا نمی‌شود و چیزی روی صفحه نشان داده نمی‌شود.

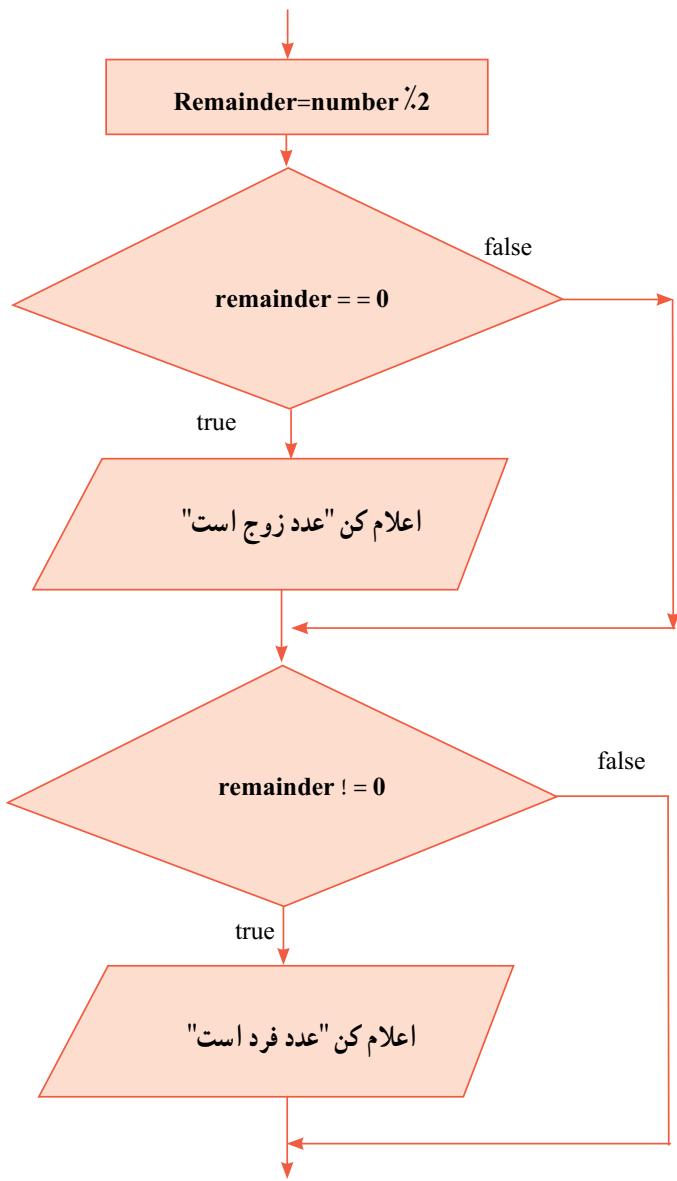
یکی از اشتباهات رایج برنامه نویسان در نوشتن عبارت منطقی دستور if، استفاده از علامت به جای است. در این صورت، مترجم متوجه بروز چنین اشتباهی می‌شود و علاوه بر کشیدن خط قرمز در زیر عبارت مورد نظر، خطای را صادر می‌کند (شکل ۱-۶).



شکل ۱-۶—اشتباه رایج برنامه نویسان و بروز خطای مترجم

**مثال ۱-۶ :** اکنون می‌خواهیم دستورهای مربوط به مثال ۱-۴ را طوری توسعه دهیم که اگر در متغیر number، عددی فرد وجود داشت، آن را شناسایی کرده، و پیام مناسب اعلام کند.

**الگوریتم یا روش انجام کار :** برای انجام این کار می‌توانیم یک دستور if به دستورهای قبلی اضافه کنیم که برای تشخیص عدد فرد، استفاده شود. اگر باقیمانده تقسیم عدد بر ۲، برابر یک شود آن عدد فرد است. بنابراین پیام "عدد فرد است" نمایش داده می‌شود. از آنجا که حاصل باقیمانده هر عددی بر ۲ عدد صفر یا یک است، بنابراین به جای شرط مساوی یک، از شرط نامساوی صفر استفاده می‌کنیم. فلوچارت ۱-۶ را مشاهده کنید.



فلوچارت ۴-۶ - تشخیص زوج و فرد با دو دستور if

مطابق با فلوچارت ۴-۶، دستورهای لازم برای انجام کار، چنین خواهد بود :

```
int remainder = number % 2;  
if (remainder == 0)  
    Console.WriteLine(number + " is Even");  
else  
    Console.WriteLine(number + " is Odd");
```

## ۴-۶- دستور شرطی if-else

در مثال ۴-۶، برای اینکه عدد زوج یا فرد را تشخیص دهیم از باقیمانده تقسیم عدد بر ۲ استفاده کردیم. در فلوچارت شکل ۴-۶، تصمیم گیری بر اساس باقیمانده تقسیم صورت می‌گیرد که عدد صفر یا مخالف است. در چنین مواردی از دستور شرطی if-else می‌توان استفاده کرد. شکل کلی این دستور چنین است :

عبارت منطقی if  
؛ دستور شماره ۱  
else  
؛ دستور شماره ۲

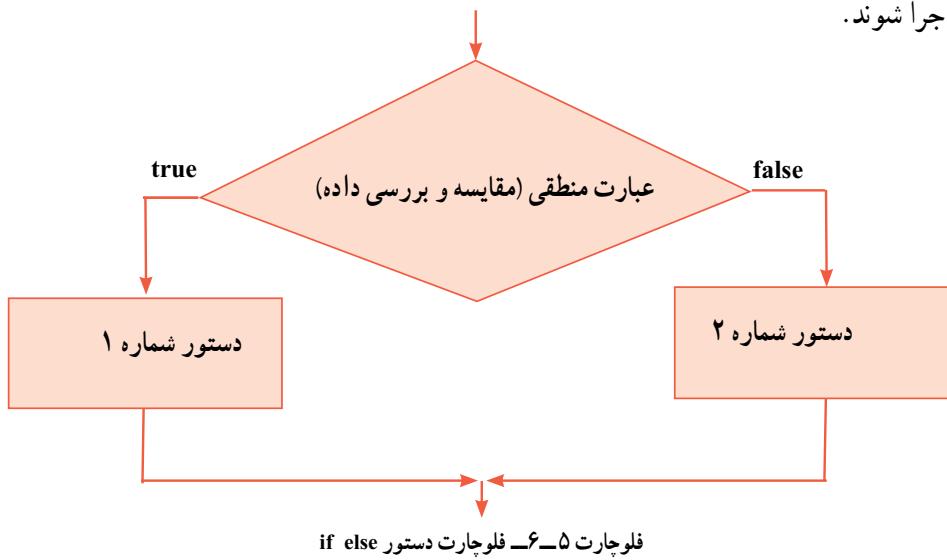
اگر نتیجه عبارت منطقی true باشد، دستور شماره ۱ که مربوط به قسمت if است اجرا می‌شود و در غیر اینصورت، یعنی اگر نتیجه عبارت منطقی false باشد، دستور شماره ۲ که مربوط به قسمت else است، اجرا می‌شود.

### نکته

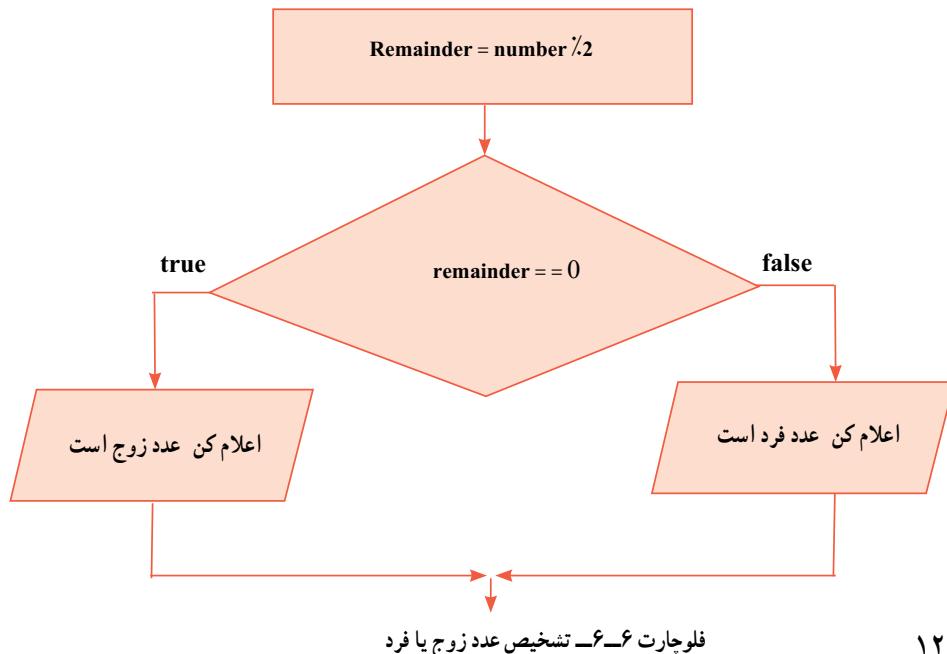
دستور if else را خوانا بنویسید یعنی :

دستورهای شماره ۱ و ۲ که مربوط به قسمت if یا قسمت else است را با کمی تورفتگی بنویسید تا معلوم شود که هر کدام متعلق به چه قسمتی است. مراقب باشید که بعد از کلمه رزرو شده else، علامت نقطه ویرگول قرار ندهید.

دستور if-else به صورت فلوچارت ۵-۶ در زیر نشان داده شده است. توجه داشته باشید که همواره فقط یکی از دو دستور شماره ۱ یا شماره ۲ اجرا می‌شود و محل است که هر دو دستور باهم اجرا شوند.



**مثال ۶-۶ :** در این مثال، دستورهای تشخیص زوج یا فرد بودن عدد را با استفاده از دستور if-else، طبق فلوچارت ۶-۶ می‌نویسیم.



```

int remainder = number % 2;
if (remainder == 0)
    Console.WriteLine(number + " is Even");
else
    Console.WriteLine(number + " is Odd");

```

روشی که در این مثال استفاده شد، نسبت به روشی که در مثال ۶-۴ به کار گرفته شد را با یکدیگر مقایسه کنید. کدام یک بهتر است؟

## کار در کارگاه ۲

**مثال ۶-۷ :** می خواهیم برنامه ای بنویسیم که حقوق دریافتی (خالص) یک کارمند را محاسبه نماید. هر کارمند دارای یک حقوق ثابت است که با توجه به میزان تحصیلات و تجربه کاری معین می شود. از حقوق تمام کارمندان، ۷ درصد به عنوان حق بیمه کسر می گردد. همچنین کارکنانی که حقوق آنها بیش از ده میلیون ریال باشد، مالیات به اندازه ۵ درصد کسر می گردد (مالیات به مازاد برد ۰ میلیون تعلق می گیرد).

**الگوریتم یا روش انجام کار :** فلوچارت ۶-۷ عملیات محاسبه حقوق را نشان می دهد.  
 با توجه به فلوچارت، ابتدا حقوق ثابت کارمند دریافت می شود. حق بیمه آن محاسبه و سپس حقوق با عدد ده میلیون ریال مقایسه می شود. چنانچه حقوق بیش از ده میلیون ریال باشد، آن گاه مقدار مالیات محاسبه و بر روی صفحه نشان داده می شود. در غیر این صورت مالیات صفر در نظر گرفته می شود. با کمی دقیق به قسمت شرط این فلوچارت متوجه می شویم که در صورت برقراری شرط، دو دستور باید اجرا شود. در حالی که در مثال های قبلی تنها یک دستور اجرا می شد. برای مشخص کردن این دو دستور در برنامه، باید آنها را در بین علامت های آکولاد باز و بسته قرار دهیم. با این کار یک بلاک<sup>۱</sup> می سازیم.

{

دستور محاسبه مالیات

دستور نمایش مقدار مالیات

}

**بلاکی شامل دو دستور**

## بلاک چیست؟

به تعدادی دستور که داخل علامت‌های آکولاد باز و بسته قرار داشته باشند بلاک گفته می‌شود.

```
{
    ;
    ;
    ;
}
```

● بلاک می‌تواند خالی باشد یعنی بین علامت‌های آکولاد، هیچ دستوری وجود نداشته باشد.

● بلاک می‌تواند فقط شامل یک دستور باشد.

● برای خوانا و واضح شدن یک بلاک، دستورهای داخل بلاک را با تورفتگی می‌نویسیم تا به راحتی در برنامه مشخص شوند.



فلوچارت ۷-۶— محاسبه حقوق دریافتی کارمندان یک شرکت

```

class Salary
{
    static void Main()
    {
        long income, tax, insurance, net;
        string input;
        Console.WriteLine("Enter income: ");
        input = Console.ReadLine();
        income = long.Parse(input);
        insurance = income * 0.07;
        if (income > 10000000)
        {
            tax = (income - 10000000) * 0.05;
            Console.WriteLine("Tax " + tax);
        }
        else
            tax = 0;
        Console.WriteLine("Insurance " + insurance);
        net = income - insurance - tax;
        Console.WriteLine("Net " + net);

        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}

```

برنامه ۴-۶- محاسبه حقوق خالص کارمندان یک شرکت (مثال ۷-۶)

**سؤال:** در مثال ۷-۶، برای محاسبه مالیات، ابتدا تفاوت حقوق با ده میلیون ریال محاسبه شده و سپس در پنج درصد ضرب شده است. به این روش محاسبه، مالیات مازاد گفته می‌شود. به نظر شما چرا مستقیماً حقوق در پنج درصد ضرب نمی‌شود؟

## ۶-۵ عملگرهای منطقی<sup>۱</sup>

عملگر منطقی && یک عملگر دوتایی و دارای دو عملوند است. در هنگام اجرای آن، ابتدا نتیجه عملوند سمت چپ، به وسیله کامپیوتر محاسبه و درستی یا نادرستی آن مشخص می‌شود. اگر ارزش عملوند سمت چپ false باشد نتیجه عملگر && نیز false خواهد بود. اما اگر نتیجه عملوند سمت چپ true باشد، آنگاه عملوند سمت راست محاسبه می‌شود که بر اساس ارزش آن، نتیجه عملگر && طبق جدول ۶-۲ مشخص می‌شود.

جدول ۶-۲ - نتیجه عملگر &&

ارزش عملوند سمت چپ	ارزش عملوند سمت راست	نتیجه عملگر &&
false	false	false
false	true	false
true	false	false
true	true	true

مثال ۶-۸: فرض کنید نمره درسی یک دانش آموز در متغیر mark قرار دارد. با بررسی قبولی یا مردودی، پیام مناسب چاپ کنید.

الگوریتم یا روش انجام کار: در این مثال دو محدوده از اعداد، باید بررسی شود. بنابراین به دو دستور if نیاز داریم که در صورت درست بودن عبارت منطقی مربوط به هر یک از آنها، پیامی بر روی صفحه نمایش داده شود. برای این مثال از فلوچارت استفاده نمی‌کنیم و به جای آن مستقیماً دستورها را می‌نویسیم. شکل زیر ساختار دستورها را نشان می‌دهد. عبارت‌های داخل پرانتز به زبان فارسی است که باید به زبان #C نوشته شوند:

(نمره کمتر از ده) if

Console.WriteLine("Failed"); // مردود

(نمره از ده تا بیست) if

Console.WriteLine("Passed"); // قبول

عبارت (نمره کمتر از ده) در زبان C#، به صورت عبارت منطقی ( $mark < 10$ ) نوشته می‌شود در این عبارت ساده در صورتی که نمره (mark) عددی کمتر از ده، مثلاً ۸ باشد نتیجه عبارت true است و در نتیجه دستور WriteLine() اجرا شده و پیام «Failed» را نمایش می‌دهد.

```
if (mark < 10)  
    Console.WriteLine("Failed");
```

اما عبارت (نمره از ده تا بیست) به سادگی عبارت قبلی در زبان C نوشته نمی‌شود، بلکه این عبارت را باید با ترکیب دو عبارت «نمره بزرگتر یا مساوی ۱۰» و همچنین «نمره کوچکتر یا مساوی ۲۰» بسازید.

( $mark > 10$  و  $mark < 20$ )

بین دو عبارت، «و» قرار دارد که برای ترکیب دو عبارت منطقی استفاده می‌شود، به طوریکه هر گاه نتیجه هر دو عبارت true باشد، نتیجه کل عبارت نیز درست است ولی اگر نتیجه حداقل یکی از عبارت‌ها false باشد، نتیجه کل عبارت نیز نادرست است. «و» در واقع یک عملگر منطقی است. بنابراین ترکیب دو عبارت قبلی با استفاده از عملگر `&&` به صورت زیر است:

```
(mark > 10) && (mark < 20)
```

در این عبارت مرکب، به ازای ۱۴ mark، نتیجه هر دو عبارت سمت چپ و راست، برابر true است. بنابراین نتیجه کل عبارت نیز true است. اما به ازای ۹ mark نتیجه عبارت false است بنابراین نتیجه کل عبارت نیز false خواهد بود.

حال دستور if مربوط به وضعیت قبولی دانش آموز چنین خواهد بود:

```
if ((mark > 10) && (mark < 20))  
    Console.WriteLine("Passed");
```

دستورهای کامل به صورت زیر است:

```
if (mark < 10)  
    Console.WriteLine("Failed");  
if ((mark > 10) && (mark < 20))  
    Console.WriteLine("Passed");
```

علاوه بر عملگر `&&`، عملگرهای دیگری نیز در زبان C، برای ایجاد عبارت‌های مرکب وجود دارد، که علامت و عملکرد آنها به ترتیب اولویت، در جدول ۳-۶ نشان داده شده است.

### جدول ۳-۶- عملگرهای منطقی

اولویت	نام عملگر	علامت	عملکرد
۱	نقیض	!	ارزش عملوند را معکوس می‌کند
۲	و	&&	تنها در صورتی که هر دو عملوند true باشند نتیجه این عملگر نیز true خواهد بود در غیر این صورت false است
۳	یا		اگر حداقل یکی از عملوندها true باشند نتیجه این عملگر نیز true خواهد بود
۴	یا انحصاری <sup>۱</sup>	<sup>۸</sup>	اگر ارزش عملوندها مخالف یکدیگر باشد، نتیجه این عملگر true خواهد بود

### جدول ۴-۶- حالت‌های مختلف در عملگرهای منطقی

نتیجه عملگر <sup>۸</sup>	ارزش عبارت سمت راست	ارزش عبارت سمت چپ	ارزش عبارت سمت چپ
false	false	false	false
true	true	false	false
true	false	true	false
false	true	true	true

### کار در کارگاه ۳

مثال ۶-۹: می‌خواهیم برنامه‌ای بنویسیم که عددی را دریافت کند و تشخیص دهد که مضرب پنج است یا خیر.

الگوریتم یا روش کار: برای تشخیص اعدادی که مضرب پنج هستند، از دو روش می‌توان استفاده می‌شود. یک روش، بررسی بخش پذیری بر پنج است، مانند روشی که برای تشخیص اعداد زوج در مثال ۵-۶، استفاده کردیم. روش دیگر، بررسی رقم یکان عدد است که در این مثال، می‌خواهیم از آن استفاده کنیم. در این روش، رقم یکان عدد را جدا می‌کنیم. اگر رقم یکان برابر پنج یا برابر صفر بود، آن گاه عدد، مضرب پنج است. به عبارت مرکب در دستور if، توجه کنید که از عملگر استفاده شده است.

۱- Exc us ve or (XOR)

```

class CheckNumbers
{
    static void Main()
    {
        string input;
        int number;

        Console.WriteLine("Enter a number: ");
        input = Console.ReadLine();
        number = int.Parse(input);

        int firstDigit = number % 10;
        if ((firstDigit == 0) || (firstDigit == 5))
            Console.WriteLine("The number is a multiple of 5.");

        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}

```

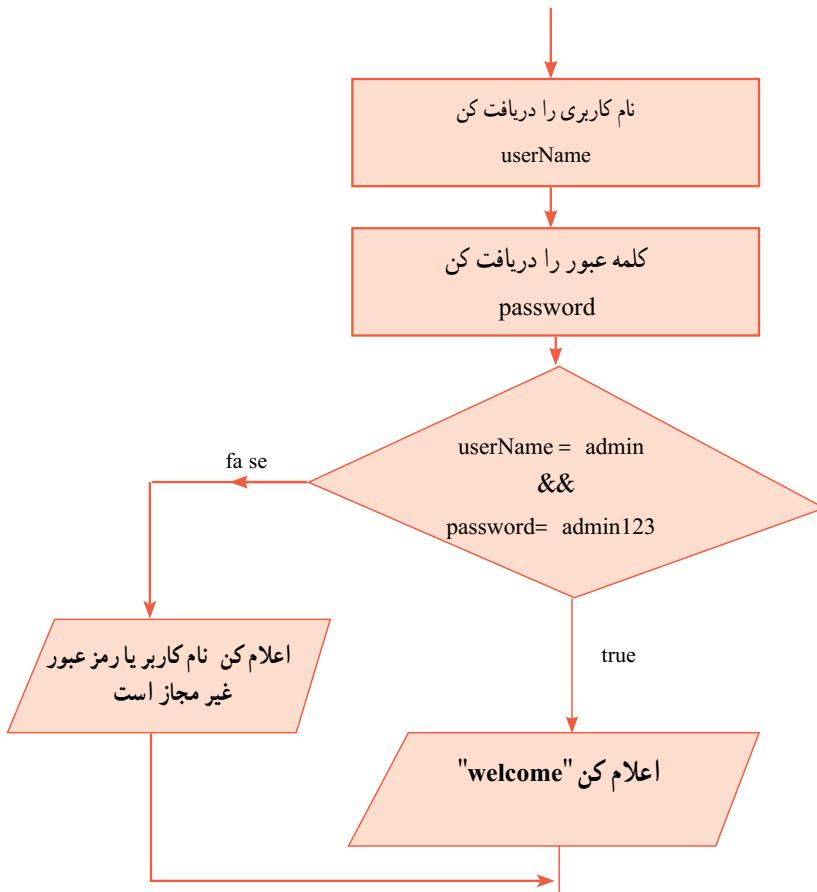
برنامه ۵\_۶— برنامه تشخیص عدد مضرب پنج (مثال ۹\_۶)

**سؤال:** خروجی برنامه به ازای اعداد ورودی ۹۰ و ۳۷ چیست؟ چه پیامی روی صفحه ظاهر می‌شود؟

**سؤال:** چه دستورهایی به برنامه اضافه می‌کنید، تا به ازای اعداد ورودی غیر مضرب ۵ مانند ۳۷، خروجی برنامه واضح باشد؟

مثال ۱۰\_۶ : می‌خواهیم مثال ۳\_۶ را تکمیل کنیم به صورتی که شخص خاصی بتواند از آن استفاده نماید. بنابراین با اجرای برنامه، باید نام کاربری و کلمه رمز دریافت شده و درستی ورودی‌ها در برنامه بررسی شود.

الگوریتم یا روش کار: در این مثال، ابتدا نام کاربری و کلمه عبور را از کاربر دریافت کرده و آنها را در دو متغیر ذخیره می‌کنیم. سپس محتوای متغیرها را با نام کاربری و کلمه عبور از قبل تعریف شده، مقایسه می‌نماییم. اگر اطلاعات وارد شده، صحیح بود پیام خوشامد گویی نمایش داده می‌شود. در غیر این صورت، برنامه پیام می‌دهد که اطلاعات وارد شده صحیح نیست (فلوچارت ۶-۸).



فلوچارت ۶-۸- دریافت نام کاربری و کلمه عبور

در فلوچارت ۶-۸، فرض شده است که کاربر مجاز، دارای نام کاربری «admin» و کلمه عبور «admin123» است. برای بررسی اطلاعات وارد شده از یک عبارت منطقی مرکب، در دستور if-else استفاده می‌کنیم. برنامه شماره ۶-۶ بر اساس این فلوچارت نوشته شده است:

```

class Login
{
    static void Main()
    {
        string userName, password;

        Console Write("Enter username: ");
        userName = Console ReadLine();

        Console Write("Enter password: ");
        password = Console ReadLine();

        if ((userName == "admin") && (password == "admin123"))
            Console WriteLine("Welcome Admin ");
        else
            Console WriteLine("Invalid username or password!");

        Console WriteLine("Press any key to continue   ");
        Console.ReadKey();
    }
}

```

برنامه ۶\_۶\_ دریافت نام کاربری و کلمه عبور کاربر (مثال ۱۰\_۶)

**مثال ۱۱\_۶ :** می خواهیم برنامه ای بنویسیم که با توجه به عدد دریافتی از کاربر، نام یکی از فصل های سال را نمایش دهد (۱ : بهار، ۲ : تابستان، ...).

الگوریتم و روش انجام کار : ابتدا عدد دریافتی را در متغیر number قرار می دهیم. سپس با مقایسه عدد دریافتی با شماره های ۱ تا ۴ نام فصل مربوطه را نمایش می دهیم.

## ۶-۶ ساختار if\_else پیچیده

در این ساختار در صورتی که عبارت منطقی ۱ درست (true) نباشد، عبارت منطقی ۲ بررسی می‌شود و در صورتی که هیچ یک از عبارات منطقی ۱، ۲ و ۳ (درست) true نباشد، دستورات شماره ۴ انجام می‌شود.

عبارت منطقی ۱)

{

; دستورات شماره ۱

}

عبارت منطقی ۲)

{

; دستورات شماره ۲

}

عبارت منطقی ۳)

{

; دستورات شماره ۳

}

else

{

; دستورات شماره ۴

{

در مثال ۱۱-۶ چنین ساختاری مورد استفاده قرار می‌گیرد.

```

class CheckSeason
{
    static void Main(string[] args)
    {
        string input;
        int number;
        Console Write("Enter a number: ");
        input = Console ReadLine();
        number = int Parse(input);

        if (number == ۱)
            Console WriteLine("Spring");
        else if (number == ۲)
            Console WriteLine("Summer");
        else if (number == ۳)
            Console WriteLine("Fall");
        else if (number == ۴)
            Console WriteLine("Winter");
        else
            Console WriteLine("Invalid Number! ");

        Console WriteLine("Press any key to continue   ");
        Console.ReadKey();
    }
}

```

برنامه ۶— دریافت شماره فصل و نمایش نام فصل (مثال ۱۱—۶)

## ۷-۶- دستور switch

در مواردی که بخواهیم حالت‌های مختلف یک عبارت را بررسی و بر اساس آن دستورهایی را اجرا کنیم، از دستور switch استفاده می‌کنیم. ساختار کلی این دستور را می‌بینید.

switch (عبارت)

{

case ۱:

; دستور ۱

break;

case : ۲

; دستور ۲

break;

:

:

default:

; دستورهای دیگر

break;

}

ساختار کلی دستور switch

در جلوی کلمه رزرو شده switch، عبارتی در داخل پرانتز نوشته می‌شود که بر اساس حاصل آن، تصمیم گیری و اجرای دستورها کنترل می‌شود. مقادیری که حاصل عبارت با آن‌ها مقایسه می‌شود، هر یک در جلوی کلمه رزرو شده case نوشته می‌شود. اگر حاصل عبارت با یک مقدار case برابر باشد، آن گاه دستور یا دستورهای جلوی case تا رسیدن به کلمه رزرو شده break اجرا می‌شود. اگر حاصل عبارت با هیچ کدام از مقادیر case برابر نشد، آن گاه دستورهای قسمت default اجرا می‌شوند.

آنکه لادهای باز و بسته محدوده عملیات شروع و پایان دستور switch را معین می‌کند. نوع عبارتی که داخل پرانتز دستور switch نوشته می‌شود، نمی‌تواند اعشاری باشد. اما عبارت‌های حرفی، رشته‌ای و انواع داده صحیح می‌تواند استفاده شود.

**مثال ۱۲-۶:** اکنون می‌خواهیم مثال ۱۱-۶ را با استفاده از دستور switch بازنویسی نماییم.

```
class CheckSeason
{
    static void Main()
    {
        string input;
        Console.WriteLine("Enter a number: ");
        input = Console.ReadLine();
        switch (input)
        {
            case "1":
                Console.WriteLine("Spring");
                break;
            case "2":
                Console.WriteLine("Summer");
                break;
            case "3":
                Console.WriteLine("Fall");
                break;
            case "4":
                Console.WriteLine("Winter");
                break;
            default:
                Console.WriteLine("Invalid Number!");
                break;
        }
        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}
```

برنامه ۷-۶— دریافت شماره فصل و نمایش نام فصل (مثال ۱۲-۶)

**۱۳-۶** مثال : در اینجا می‌خواهیم مثال ۲-۶ را توسعه دهیم. تشخیص سه کاربر مختلف، با دریافت نام کاربری و کلمه عبور، انجام شود. در این برنامه، نام کاربری با دستور switch و کلمه عبور با دستور if بررسی شده است.

```
class SwitchDemo
{
    static void Main(string[] args)
    {
        string userName, password;
        Console.WriteLine("Enter username: ");
        userName = Console.ReadLine();
        Console.WriteLine("Enter password: ");
        password = Console.ReadLine();
        switch (userName)
        {
            case "admin":
                if (password == "admin123")
                    Console.WriteLine("Welcome Manager.");
                else
                    Console.WriteLine("Wrong password!");
                break;
            case "accountant":
                if (password == "acc123")
                    Console.WriteLine("Welcome accountant.");
                else
                    Console.WriteLine("Wrong password!");
        }
    }
}
```

```

        break;

    case "sales":
        if (password == "sales123")
            Console.WriteLine("Welcome Manager!");
        else
            Console.WriteLine("Wrong password!");
        break;

    default:
        Console.WriteLine("Invalid username!");
        break;
    }
}

Console.WriteLine("Press any key to continue... ");
Console.ReadKey();
}
}

```

برنامه ۷-۶— دریافت نام کاربری و رمز عبور کاربران (مثال ۱۳-۶)

اگر به ساختار دستور switch توجه کنید، پس از هر دستور case، یک دستور break نوشته شده است. از کلمه رزرو شده break برای خاتمه دادن به یک case استفاده می‌شود. اگر دستور break نوشته نشود، مترجم برای جلوگیری از اشتباه برنامه نویس، خطای می‌دهد<sup>۱</sup>. برای هر case، می‌توان بیش از یک دستور نوشت و نیازی به بلاک ندارد.

**۲- مثال ۱۴-۶** : با استفاده از قطعه برنامه زیر، متن کامل برنامه را در محیط VS بنویسید.  
در این قطعه برنامه از کاربر سؤالی پرسیده می‌شود، کاربر در پاسخ به سؤال، کلمه‌ای را وارد می‌کند. اگر کاربر کلمه Yes و یا کلمه maybe را وارد نماید، هر دو یک نتیجه را خواهد داشت و پیام Great! بر روی صفحه نشان داده می‌شود.

۱- در زبان C یا C، می‌توانید دستور break را تنویسید که در این صورت بعد از پایان اجرای دستورات یک case، دستورات بعدی نیز اجرا می‌شود.

```

Console.WriteLine("Do you enjoy C# ? (yes/no/maybe): ");
string input = Console.ReadLine();
switch (input)
{
    case "yes":
    case "maybe":
        Console.WriteLine("Great!");
        break;
    case "no":
        Console.WriteLine("Too bad!");
        break;
}

```

قطعه برنامه ۸\_۶\_۶ کاربرد دستور switch بدون break

**۳** برنامه‌ای بنویسید که یک عدد را بگیرد و زوج یا فرد بودن آن را تشخیص دهد.

**۴** برنامه‌ای بنویسید که نمره‌دانش آموز را دریافت کند و پیامی برای قبولی یا مردودی او صادر نماید.

**۵** برنامه شماره ۱۱\_۶ و ۱۲\_۶ را نوشته و ترجمه کنید. به ازای ورودی‌های مختلف آن را

اجرا کنید.

**۶** اگر کاربر در هنگام اجرای مثال ۱۴\_۶، کلمه دیگری به غیر از کلمات تعیین شده بنویسد، خروجی چه خواهد بود؟

**۷** می‌خواهیم اگر کاربر کلمه دیگری غیر از کلمات تعیین شده در مثال ۱۳\_۶، را وارد کرد، دستور زیر اجرا گردد، چه تغییری در داخل دستور switch انجام می‌دهید؟

```
Console.WriteLine("I'm sorry, I don't understand that!");
```

## خودآزمایی فصل ششم

۱- دستورهای زیر برای کنترل محتوای number نوشته شده است. با تغییراتی در عبارت منطقی و دستور if، همین کنترل را با استفاده از عملگر `&&` انجام دهید :

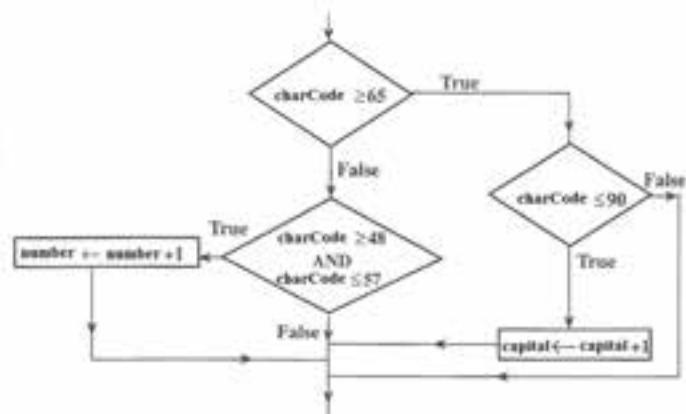
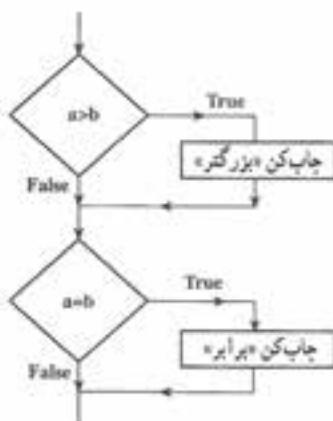
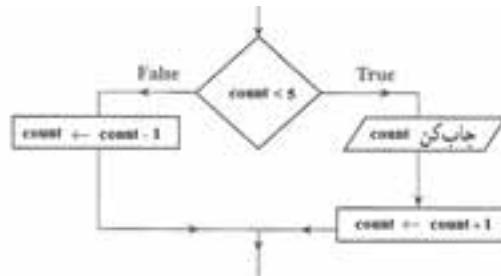
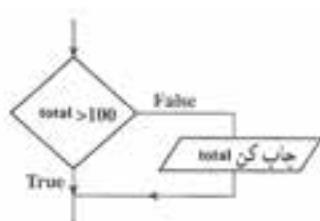
```
if ((number > 10) && (number < 0))
```

```
    Console.WriteLine("The number should between zero to ten.");
```

```
else
```

```
    Console.WriteLine("Good job!");
```

۲- دستورهای شرطی مطابق با فلوچارت‌های زیر بنویسید :



۳- بدون استفاده از دستور if، دستور زیر را بنویسید.

```
if (number < 100)
```

```
    number =;
```

```
else
```

```
    number = 1;
```

۴- در هر یک از دستورهای زیر، با استفاده از عملگرهای منطقی، تعداد دستورهای if را کاوش دهید.

(الف)

```
if(score > 0)  
    if(score < 100)  
        Console.WriteLine("Score " score);
```

(ب)

```
if(choice == 1)  
    Console.Clear();  
if(choice == 2)  
    Console.Clear();  
Console.WriteLine("Starting...");
```

۵- اگر در متغیر char، یک کاراکتر قرار داشته باشد، با استفاده از دستورهای شرطی معین کنید که در داخل متغیر، چه نوع کاراکتری (حروف بزرگ، حروف کوچک و یا ارقام) قرار دارد؟

۶- در هر یک از دستورهای شرطی زیر، دستور WriteLine() در چه صورت اجرا می‌شود؟

(الف)

```
if(number > 0)  
    if(number < 10)  
        failed ;  
    else  
        Console.WriteLine(number);
```

(ب)

```
if(number > 0)  
{  
    if(number < 10)  
        failed ;  
}  
else  
    Console.WriteLine(number);
```

## تمرینات برنامه‌نویسی فصل ششم

- ۱- برنامه‌ای بنویسید که عددی را به عنوان شماره فصل از ورودی دریافت کند. نام ماههای آن فصل را در خروجی چاپ نماید. برای ورودی غیر مجاز، پیام مناسب نشان داده شود.
  - ۲- برنامه‌ای بنویسید که سن کاربر را دریافت کند و تعداد روزهای عمر او را نشان دهد. عدد وارد شده توسط کاربر در برنامه کنترل شود. سن کاربر نمی‌تواند منفی و یا بزرگتر از عددی مانند  $15^{\circ}$  باشد.
  - ۳- برنامه‌ای بنویسید که عددی را از ورودی دریافت کند و تعیین کنده این عدد مضرب  $6$  است یا خیر؟ از روش بخش‌پذیری هم زمان بر  $2$  و  $3$  استفاده نماید.
  - ۴- برنامه‌ای بنویسید که عددی را از ورودی دریافت کند و زوج بودن آن را تشخیص دهد. برای بخش‌پذیری عدد ورودی بر  $2$ ، از روش بررسی رقم یکان شامل  $0, 2, 4, 6, 8$  استفاده کنید.
  - ۵- برنامه‌ای بنویسید که یک عدد یک رقمی دریافت کند و کلمه منتظر با آن عدد را روی صفحه نمایش دهد مثلاً اگر عدد  $5$  از ورودی دریافت شد، برنامه کلمه FIVE را نمایش دهد. این برنامه را برای عدد دو رقمی توسعه دهید.
  - ۶- برنامه‌ای بنویسید که میزان مصرف برق در یک ماه را بر حسب کیلووات سوال نماید و سپس بهای برق مصرفی یک ماه را، بر اساس جدول زیر محاسبه کند و مبلغ آن را بر حسب ریال نمایش دهد. مثلاً اگر مصرف برق  $18^{\circ}$  کیلووات در ماه باشد در این صورت  $10^{\circ}$  کیلووات آن با مبلغ  $1364$  ریال محاسبه شده و  $8^{\circ}$  کیلووات با قیمانده آن، با مبلغ  $1426$  ریال محاسبه می‌شود :
- $(10 \times 1364) + (8 \times 1426)$  بهای برق مصرفی

نرخ (ریال)	پله‌های مصرف $3^{\circ}$ روزه (kwh)
۱۳۶۴	صرف تا ۱
۱۴۲۶	مازاد بر ۱ تا ۲
۱۴۸۸	مازاد بر ۲ تا ۳
۱۵۵	مازاد بر ۳ تا ۴
۱۷۲۶	مازاد بر ۴ تا ۵
۱۹۸۴	مازاد بر ۵ تا ۶
۲۲۲۲	مازاد بر ۶

- ۷- برنامه‌ای بنویسید که سه عدد از ورودی دریافت کند و عدد بزرگتر را تشخیص داده و آن را نمایش دهد.

۸- برنامه‌ای بنویسید که شماره روز را دریافت و نام روز را چاپ کند. مثلاً اگر اولین روز سال جاری چهارشنبه بوده است، با ورود عدد ۱، چهارشنبه، با ورود عدد ۲، پنج شنبه و ... چاپ شود. از دستور switch استفاده کنید.

۹- برنامه‌ای بنویسید که دمای هوا را از ورودی دریافت نماید. بر طبق جدول زیر، پیام مناسب را چاپ نماید.

خروجی	ورودی
بسیار سرد	< دما
سرد	$\leq$ دما <
معتل	$\leq$ دما > ۱
گرم	۱ $\leq$ دما > ۲
بسیار گرم	۲ $\leq$ دما > ۳

۱۰- برنامه‌ای بنویسید که شماره رنگ را از ورودی دریافت نماید. طبق جدول زیر، رنگ زمینه صفحه نمایش را به رنگ مربوطه تغییر دهد.

رنگ صفحه نمایش	ورودی(شماره رنگ)
مشکی	
آبی	۱
قرمز	۲
سبز	۳

۱۱- فرض کنید آزمونی دارای ۳۰ سؤال چهارگزینه‌ای باشد. برنامه‌ای بنویسید که تعداد پاسخ‌های درست و تعداد پاسخ‌های نادرست یک شرکت‌کننده در این آزمون را دریافت کرده، نمره‌ی وی و درصد پاسخ‌گویی به سوالات را محاسبه نماید. (با این فرض که هر پاسخ غلط  $\frac{1}{3}$  نمره منفی دارد). مثلاً اگر داشت آموز ۱۷ پاسخ درست و ۶ پاسخ غلط داشته باشد، نمره‌ی وی برابر ۱۵ می‌باشد (از ۳۰ نمره) و درصد پاسخ‌گویی وی، برابر ۵۰٪ است.

۱۲- آیا برنامه تمرین ۱۱، کنترلی بر روی صحّت داده‌های ورودی برنامه دارد؟ مثلاً کنترل می‌کند که مجموع تعداد پاسخ‌های صحیح و غلط کمتر یا مساوی تعداد کل سوالات آزمون باشد؟  
۱۳- با تکمیل تمرین ۱۱ از بروز چنین خطاهایی در ورودی جلوگیری کنید.

## فصل هفتم

### دستورات تکرار (حلقه‌ها)

در بعضی از برنامه‌های کاربردی باید یک عمل چندین بار تکرار شود. مثلاً اگر بخواهیم میانگین یا معدل نمرات درس زبان انگلیسی یک کلاس را محاسبه کنیم، باید نمرات تمام دانشآموزان کلاس را از ورودی دریافت کرده و با یکدیگر جمع کنیم. در این مثال، عمل دریافت نمره از ورودی و عمل جمع زدن نمره‌ها، به تعداد دانشآموزان کلاس، باید تکرار گردد. نوشتن چنین برنامه‌هایی با دستورات تکراری، خسته کننده و طولانی و گاهی غیرممکن خواهد بود. در زبان‌های برنامه‌نویسی از جمله زبان C#، دستورات ایجاد حلقه، برای کوتاه کردن تعداد دستورات برنامه، پیش‌بینی شده‌اند. به وسیله این دستورات، برنامه‌نویس می‌تواند، عملیات و پردازش‌های تکرار شونده را فقط یک بار بنویسد و کامپیوتر آنها را به دفعات، تکرار کند. در این فصل با انواع دستورات حلقه و کاربرد آنها، آشنا می‌شویم.

پس از پایان این فصل انتظار می‌رود که فرآگیر بتواند:

- ۱- کاربرد حلقه در برنامه را توضیح دهد.
- ۲- دستورات ایجاد حلقه را نام ببرد و تفاوت هر یک را بیان کند.
- ۳- عملکرد و کاربرد دستور حلقه while را توضیح دهد.
- ۴- عملکرد و کاربرد دستور حلقه for را توضیح دهد.
- ۵- برنامه‌های کاربردی با حلقه تکرار بنویسد.

#### ۱-۷- دستورات تکرار شرطی

فرض کنید، می‌خواهیم برنامه‌ای بنویسیم که فقط، افراد خاصی مجاز به استفاده از آن باشند. بدین منظور، در ابتدای برنامه، نام کاربری و کلمه عبور را سؤال می‌کنیم. اگر کاربر توانست اطلاعات خواسته شده را به طور صحیح وارد کند، به قسمت‌های بعدی برنامه هدایت می‌شود و در غیر این صورت، مجدداً نام کاربری و کلمه عبور درخواست می‌شود.

در چنین برنامه‌هایی، عمل دریافت اطلاعات، ممکن است تکرار گردد. تکرار دستورات یک برنامه، بسته به نوع الگوریتم آن، می‌تواند با دفعات معین و یا نامعین باشد. زمانی که تعداد دفعات نامشخص است، توقف و یا تکرار بستگی به برقراری یک شرط دارد. در این گونه موارد از دستورات حلقه شرطی مانند do-while یا while استفاده می‌کنیم. اگر تعداد دفعات تکرار مشخص باشد، مثلاً حداقل ۳ بار نام کاربری و کلمه عبور دریافت گردد، از دستور حلقه معین for استفاده می‌شود.

## ۷-۱ - دستور حلقه شرطی while : ساختار کلی دستور while، در زیر نشان داده شده است :

(عبارت منطقی) while

; دستور

دستور while از سه بخش تشکیل شده است :

- ۱ - کلمه رزرو شده while
- ۲ - عبارت منطقی در داخل پرانتز
- ۳ - دستوری که در صورت درست بودن نتیجه عبارت، اجرا خواهد شد.

مثال ۱-۷ : نمونه‌ای از به کار گیری دستور while چنین است :

```
int x = 1;  
while (x < 100)  
    Console.WriteLine("x " + x);
```

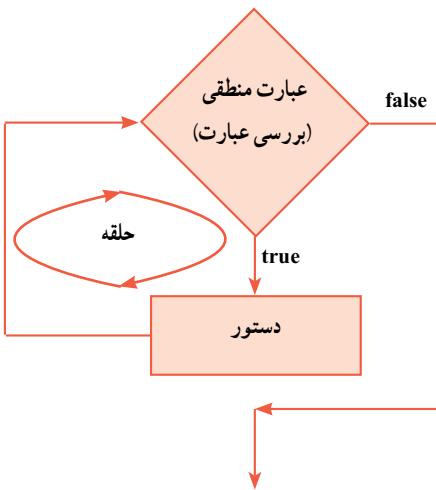
قطعه برنامه ۱-۷-۱ از یک حلقه

**سؤال:** به نظر شما خروجی این دستورات چیست؟ (چه اعدادی روی صفحه نمایش، نشان داده می‌شود؟)

## نکته

به علامت نقطه ویرگول در دستور while توجه کنید. بعد از علامت پرانتز علامت نقطه ویرگول نگذارید، زیرا دستور while هنوز تمام نشده است. علامت نقطه ویرگول باید در انتهای دستور نوشته شود.

نقطه ویرگول ندارد  
(عبارت منطقی) while ; دستور



فلوچارت ۱-۷- دستور while

هنگامی که کامپیوتر در حال اجرای برنامه است، با رسیدن به دستور while، ابتدا مقدار عبارت را بررسی می‌کند. در صورتی که مقدار عبارت true باشد، دستور (یا بلاک) نوشته شده بعد از while، اجرا می‌شود. پس از آن، دوباره مقدار عبارت محاسبه می‌شود و تا زمانی که ارزش آن true باشد، دستور مذکور، اجرا خواهد شد. در این حالت می‌گوییم حلقه<sup>۱</sup> ایجاد شده است (فلوچارت ۱-۷). دستور یا دستوراتی که مکرر اجرا می‌گردند در بدنه حلقه<sup>۲</sup> قرار دارد. اگر در ارزیابی عبارت، مقدار false حاصل شود، دستورات بدنه حلقه دیگر اجرا نخواهند شد. برنامه از حلقه خارج می‌شود و دستورات بعدی اجرا می‌شوند.

### نکته

اگر بخواهید بیش از یک دستور تکرار گردد، باید آنها را به صورت یک بلاک بنویسید. یعنی آنها را در داخل علامت‌های آکولاد باز و بسته قرار دهید.

### کار در کارگاه ۱

- ۱- مثال ۱-۷ را به صورت یک برنامه کامل در VS تایپ و اجرا نمایید.
- ۲- اعداد حلقه را طوری تغییر دهید تا خروجی اعداد دو رقمی شود.
- ۳- اعداد حلقه را طوری تغییر دهید تا خروجی اعداد سه رقمی و به صورت تزولی شود.

۱- Loop

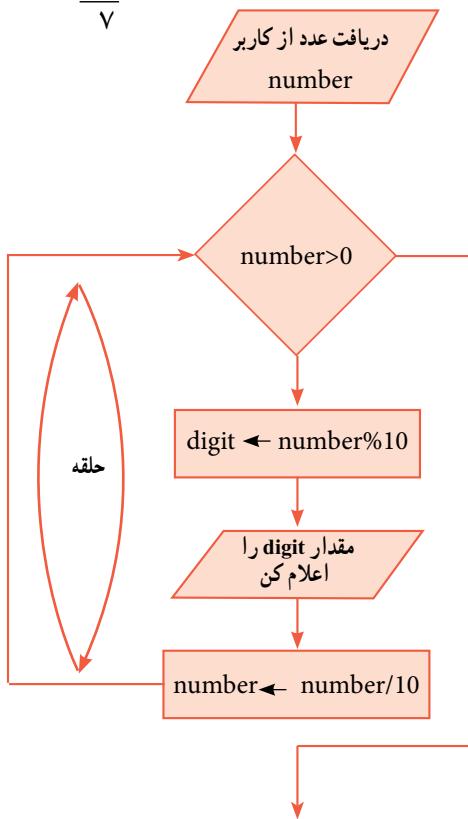
۲- Loop Body

**مثال ۷-۲** : می خواهیم برنامه ای بنویسیم که ارقام یک عدد را جدا نموده و آن ها را نمایش دهد.  
**الگوریتم یا روش انجام کار** : با توجه به آن که رقم یکان هر عدد، باقیمانده تقسیم آن عدد صحیح بر  $10^{\circ}$  است، کافی است عدد دریافتی را بر  $10^{\circ}$  تقسیم و باقیمانده آن را نمایش دهیم. به عنوان مثال اگر عدد دریافتی ۵۷۶ باشد باقیمانده تقسیم آن بر عدد  $10^{\circ}$ ، عدد ۶ است که رقم یکان عدد است.

$$\begin{array}{r} 576 \\ \hline 7 \end{array} \quad \begin{array}{r} 1^{\circ} \\ | \\ 57 \\ \hline 57 \\ \hline 0 \end{array}$$

اگر دوباره خارج قسمت بدست آمده یعنی ۵۷ را بر عدد  $10^{\circ}$  تقسیم کنیم، خواهیم داشت :

$$\begin{array}{r} 57 \\ \hline 7 \end{array} \quad \begin{array}{r} 1^{\circ} \\ | \\ 5 \\ \hline 5 \\ \hline 0 \end{array}$$



اگر به باقیمانده تقسیم بالا توجه کنید، متوجه می شوید که عدد ۷، رقم ده گان عدد دریافتی است. به همین ترتیب ادامه می دهیم و عمل تقسیم را تکرار کرده و باقیمانده تقسیم را به دست می آوریم.

$$\begin{array}{r} 5 \\ \hline 5 \end{array} \quad \begin{array}{r} 1^{\circ} \\ | \\ 5 \\ \hline 0 \end{array}$$

baciymande تقسیم بالا را در نظر بگیرید. در اینجا نوشتیم آخرین رقم عدد یعنی ۵ را نیز جدا کنیم.

با دقت در عملیات فوق متوجه می شویم که عمل تقسیم، عملی تکراری است و تا زمانی انجام می شود که مقسوم آن بزرگتر از صفر باشد (فلوچارت ۷-۲).

فلوچارت ۷-۲- جدا کردن ارقام عدد

مطابق با فلوچارت شکل ۷-۲، برنامه را می‌نویسیم :

```
class Numbers
{
    static void Main()
    {
        int number, digit ;
        string input;
        Console.WriteLine("Enter a number: ");
        input = Console.ReadLine();
        number = int.Parse(input);
        while (number > 0)
        {
            digit = number % 10;
            Console.WriteLine(digit);
            number /= 10;
        }
        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}
```

برنامه ۷-۲— جدا کردن ارقام یک عدد صحیح

**سؤال:** در برنامه ۷-۲، یک بلاک شامل سه دستور، در داخل حلقه قرار دارد که تکرار می‌شود. آن بلاک و دستورهای داخل آن را مشخص کنید.

**سؤال:** برنامه را برای عدد ۳۸۵ در جدول، Trace کنید.

input	number	digit	خروجی
385			
۱۴۵			

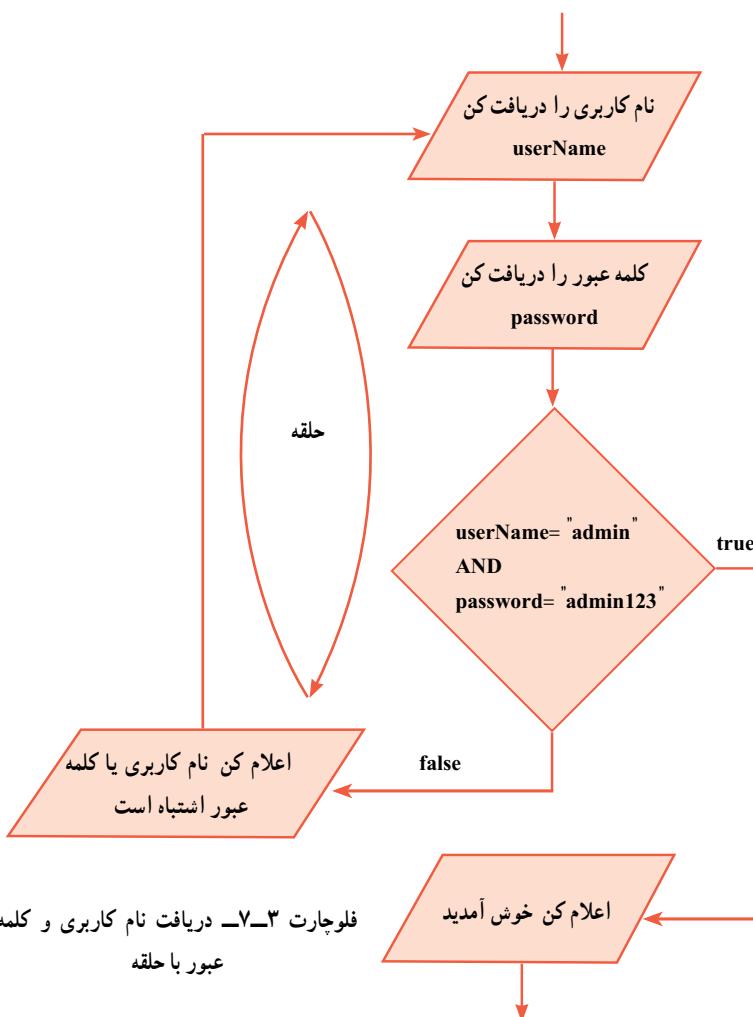


**سؤال:** آیا ممکن است که دستورات داخل حلقه، اصلاً اجرا نشوند؟ به ازای چه

مقادیری این اتفاق می‌افتد؟

**مثال ۷-۳:** می‌خواهیم برنامه‌ای بنویسیم که نام کاربری و رمز عبور را سؤال نماید و اگر کاربر اطلاعات خواسته شده را به درستی وارد نکرد، دوباره سؤال شود.

**الگوریتم یا روش انجام کار :** در فصل ششم در مثال ۱۰-۶، با روش دریافت و بررسی نام کاربری و کلمه عبور آشنا شدیم. در این مثال از حلقه برای تکرار عملیات استفاده می‌کنیم. در صورتی که کاربر اطلاعات را به طور صحیح وارد نکرد، باید دوباره عمل دریافت و بررسی اطلاعات تکرار شود. فلوچارت ۷-۳، حلقه و عملیات تکراری را نشان می‌دهد.



فلوچارت ۷-۳ - دریافت نام کاربری و کلمه  
عبور با حلقه

اگر فلوچارت ۳-۷ را با دقت بررسی کنید، متوجه می‌شوید که در این مثال، ابتدا دستورات داخل حلقه اجرا می‌شوند و سپس شرط ادامه تکرار، بررسی می‌شود. در حالی که در مثال قبلی، ابتدا شرط بررسی می‌شد و سپس در صورت برقراری شرط، دستورات داخل حلقه اجرا می‌شد. در زبان برنامه‌نویسی C#، دستور حلقه do-while برای این گونه مسایل در نظر گرفته شده است، که در این قسمت به شرح آن می‌پردازیم.

۲-۱ دستور حلقه شرطی do-while : شکل کلی دستور do-while به صورت زیر است :

do

; دستور

while (عبارت منطقی) ;

دستور do-while از چهار بخش تشکیل شده است :

۱- کلمه رزرو شده do

۲- دستور داخل حلقه

۳- کلمه رزرو شده while

۴- عبارت منطقی داخل پرانتز، که در صورت درست بودن آن، دستور داخل حلقه تکرار می‌شود.

مثال ۴-۷ : نمونه‌ای از به کارگیری دستور do-while چنین است :

```
int x = 1;  
do  
    Console.WriteLine("x = " + x);  
    while (x < 100);
```

قطعه برنامه ۳-۷-۱ مثالی از یک حلقه

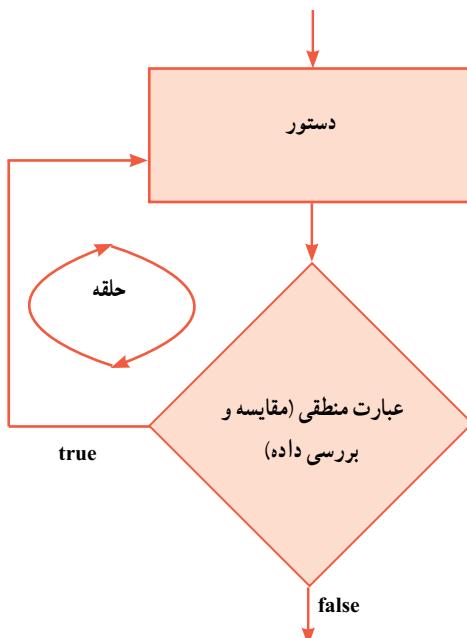
**سوال:** به نظر شما خروجی این دستورات چیست؟ (چه اعدادی روی صفحه نمایش، نشان داده می‌شود؟)

نکته

اگر بخواهید بیش از یک دستور در حلقه قرار گیرد، باید آنها را در یک بلاک قرار دهید.

به محل نوشتن علامت ; در دستور do-while توجه کنید. این علامت بعد از عبارت منطقی باید نوشته شود.

کامپیوتر با رسیدن به دستور do-while، ابتدا دستور داخل حلقه را اجرا می‌کند که از کلمه do شروع می‌شود و سپس با رسیدن به کلمه while، مقدار عبارت منطقی را ارزیابی می‌نماید. اگر حاصل عبارت true باشد، آنگاه به قسمت do برمی‌گردد و دستور بدنه حلقه اجرا می‌شود. تازمانی که حاصل عبارت true است حلقه تکرار می‌شود. اگر حاصل ارزیابی عبارت false شود، دیگر به کلمه do برمی‌گردد و کنترل برنامه به خط بعد از while واگذار شده و دستورات دیگر برنامه اجرا می‌شوند. فلوچارت ۷-۴، دستور do-while را شان می‌دهد.



فلوچارت ۷-۴—دستور do-while

## کار در کارگاه ۲

برنامه مربوط به فلوچارت ۷-۳ (دریافت نام کاربری و کلمه عبور)، را با استفاده از حلقه do-while بنویسید.

```

class LoginLoop
{
    static void Main(string[] args)
    {
        string userName, password;
        bool loginFlag;
        do
        {
            Console.WriteLine("Enter username: ");
            userName = Console.ReadLine();
            Console.WriteLine("Enter password: ");
            password = Console.ReadLine();
            if ((userName == "admin") && (password == "admin123"))
                loginFlag = true;
            else
            {
                loginFlag = false;
                Console.WriteLine("Wrong username or password!. Try again.");
            }
        } while (!loginFlag);
        Console.WriteLine("Welcome Admin.");
        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}

```

برنامه ۷— دریافت نام کاربری و رمز عبور در داخل حلقه

در برنامه ۷-۴، پیش از یک دستور در داخل حلقه قرار دارد، بنابراین از علامت‌های آکولاد باز و بسته برای ایجاد یک بلاک استفاده شده است که بین کلمات do و while قرار دارند. در داخل بلاک ابتدا نام کاربری و کلمه عبور دریافت شده است و سپس درستی آنها توسط دستور if، بررسی شده است. برای کنترل حلقه (یا شرط تکرار حلقه) از یک متغیر منطقی به نام loginFlag استفاده شده است. اگر کاربر اطلاعات نام کاربری و کلمه عبور را صحیح وارد کند، در این متغیر مقدار true قرار می‌گیرد. اما اگر کاربر اطلاعات نادرست وارد کند، در این متغیر مقدار false قرار می‌گیرد.  
به عبارت منطقی کنترل حلقه که پس از کلمه while نوشته شده است دقت کنید :

```
{ while (!loginFlag);
```

در عبارت منطقی، از عملگر نقیض استفاده شده است. بنابراین تا زمانی که مقدار متغیر loginFlag برابر false است، حلقه تکرار می‌گردد. هر گاه کاربر، اطلاعات صحیح را وارد کند در متغیر loginFlag مقدار true قرار گرفته و در نتیجه حاصل عبارت منطقی false شده و حلقه دیگر تکرار نمی‌شود. در نتیجه دستور بعد از While اجرا می‌شود که نمایش یک پیام خوشامد گویی است.

```
Console.WriteLine("Welcome Admin.");
```

**سؤال:** چرا در قسمت else از علامت‌های آکولاد باز و بسته استفاده شده است، اما در قسمت if، چنین نیست؟



مثال ۷-۷ : می‌خواهیم یک بازی حدس عدد، ایجاد کنیم. این بازی بین دو بازیکن به شرح زیر صورت می‌گیرد. بازیکن اول عددی را برای خود در نظر می‌گیرد و بازیکن دوم باید آن عدد را حدس بزند. بازیکن اول در طول بازی، راهنمایی لازم را در اختیار بازیکن دوم قرار می‌دهد تا عدد بالاتر یا پایین تری را حدس بزند.

الگوریتم یا روش انجام کار : با توجه به شرح بازی، ابتدا عدد مورد نظر بازیکن اول را سؤال کرده و در یک

متغیر (number) ذخیره می‌کنیم. سپس از بازیکن دوم می‌خواهیم تا عددی که بازیکن اول وارد کرده است را حدس بزند. عدد دریافتی از بازیکن دوم (guess)، باید با عدد مورد نظر بازیکن اول مقایسه شود که در این صورت سه حالت رخ می‌دهد :

حالت اول : guess < number است که در این صورت پیام «آفرین درست حدس زدید» اعلان شود.

حالت دوم : guess < number است که در این صورت پیام «برو بالا» اعلان شود.

حالت سوم : guess > number است که در این صورت پیام «برو پایین» اعلان شود.

تا زمانی که بازیکن دوم، عدد را درست حدس نزد است، عملیات دریافت عدد از بازیکن دوم و مقایسه آن باید تکرار شود. بنابراین در این برنامه نیاز به یک حلقه می‌باشد. از آن جا که عمل حدس زدن عدد، حداقل یک بار به وسیله بازیکن دوم انجام می‌شود، به نظر شما از چه دستور حلقه باید استفاده کنیم؟

فلوچارت ۷-۵، لازم برای انجام این بازی را نشان می‌دهد. آن را به ازای ورودی‌های مختلف دنبال کنید.



فلوچارت ۷-۵—بازی حدس عدد

بر اساس فلوچارت فوق، برنامه ۷-۵ را می‌نویسیم.

```

class GussTheNumber
{
    static void Main(string[] args)
    {
        string input;
        int number, guess;
        Console.WriteLine("Player 1: Think a number (1100_): ");
        input = Console.ReadLine();
        number = int.Parse(input);
        Console.Clear(); // clear console screen
        do
        {
            Console.WriteLine("Player 2: Guess the number (1100_): ");
            input = Console.ReadLine();
            guess = int.Parse(input);
            if (guess < number)
                Console.WriteLine("Incorrect, Go Up.");
            else
                if (guess > number)
                    Console.WriteLine("Incorrect, Go Down.");
                else
                    Console.WriteLine("Well done!, it's correct.");
        } while (guess != number);
        Console.WriteLine("Press any key to continue... ");
        Console.ReadKey();
    }
}

```

do-while دایل

در برنامه ۷-۵، از حلقه do-while استفاده شده است. به عبارت منطقی در دستور while توجه کنید :

```
{ while (guess != number);
```

اگر حدس بازیکن دوم، مخالف عدد مورد نظر بازیکن اول باشد، نتیجه عبارت true است و حلقه تکرار می‌شود. تنها در صورتی که هر دو عدد با یکدیگر برابر باشند، نتیجه عبارت false خواهد شد و حلقه قطع شده و دستور بعد از while اجرا می‌شود. توجه کنید که چون دستورات داخل حلقه بیش از یک دستور است، در داخل یک بلاک قرار گرفته اند.

گسترش برنامه : معمولاً برنامه‌ای که نوشته می‌شود، ایده‌آل و کامل نیست. امکان اضافه کردن ویژگی و قابلیت‌های جدید در هر برنامه وجود دارد. در برنامه ۷-۵ نیز، می‌توان امکاناتی را اضافه کرد. به عنوان مثال، قابلیت امتیازدهی را به برنامه اضافه می‌کنیم. این امتیاز باید متناسب با تعداد دفعاتی باشد، که بازیکن دوم تلاش می‌کند تا عدد مورد نظر بازیکن اول را پیدا کند. در قسمت کار در کارگاه، در انتهای این فصل، چنین امکانی را به برنامه ۷-۵ اضافه می‌کنیم.

## ۷-۲ دستور حلقه for

در مقدمه این فصل، مثالی در مورد محاسبه میانگین نمرات درسی یک کلاس بیان شد که در آن، عملیات دریافت نمرات و محاسبه مجموع آنها، باید به تعداد دانش آموزان یک کلاس تکرار شود. در چنین برنامه‌هایی که در آن تعداد تکرار دستورات معین است، بهتر است از دستور حلقه for استفاده کنیم که برای این منظور در زبان C# پیش‌بینی شده است. برای آشنایی با این دستور با یک مثال ساده جهت نمایش اعداد ۱ تا ۱۰ شروع می‌کنیم.

مثال ۷-۶ : می‌خواهیم اعداد طبیعی از ۱ تا ۱۰ را روی صفحه نمایش، نشان دهیم. از دستور for به صورت زیر استفاده می‌کنیم :

```
for (int i = 1; i < 10; i++)
```

```
    Console.WriteLine(i);
```

قطعه برنامه ۷-۶ - استفاده از حلقه for برای نمایش اعداد طبیعی ۱ تا ۱۰

دستور (WriteLine) در این مثال در داخل حلقه قرار دارد که در هر بار تکرار حلقه، مقدار متغیر *i* روی صفحه نمایش، نشان داده می‌شود. مقدار متغیر *i* چقدر است؟

برای پاسخ به این سؤال، به داخل پرانتز، در جلوی دستور for، توجه کنید. سه قسمت در داخل پرانتز، قابل تشخیص است. در قسمت اول، یک متغیر از نوع صحیح به نام i با مقدار اولیه ۱ تعریف شده است. قسمت دوم، یک عبارت منطقی ( $10 < i$ ) است و قسمت سوم نیز یک دستور انتساب افزایشی ( $i++$ ) است.

جزیيات اجرای دستورات بالا چنین است: در دستور for، ابتدا عدد ۱ در متغیر i قرار می‌گیرد و سپس عبارت منطقی محاسبه می‌شود و چون ( $10 < i$ ) است، نتیجه عبارت true است. بنابراین دستور داخل حلقه اجرا می‌شود و در نتیجه عدد ۱ در روی صفحه نمایش داده می‌شود. سپس دستور انتساب افزایشی انجام می‌شود یعنی مقدار متغیر به اندازه یک واحد افزایش می‌یابد.

دوباره عبارت منطقی محاسبه می‌شود و چون هنوز نتیجه عبارت درست است ( $1 < 10$ ) در نتیجه دستور داخل حلقه اجرا شده و عدد ۲ بر روی صفحه نشان داده می‌شود. این عملیات تکرار می‌شود تا زمانی که نتیجه عبارت نادرست شود یعنی ( $10 < 10$ ) که در این صورت حلقه قطع می‌شود. بنابراین اعداد ۱ تا ۱۰ روی صفحه نمایش، نشان داده می‌شود.

با توجه به مثال، شکل کلی دستور for چنین است:

(تغییر مقدار متغیر؛ عبارت منطقی؛ مقدار اولیه **نام متغیر**) for  
؛ دستور

کلمه for، یک کلمه رزرو شده است. متغیری که در حلقه for استفاده می‌شود، هر نام مجازی می‌تواند داشته باشد. این متغیر به نام شمارنده<sup>۱</sup> معروف می‌باشد. چون نقش شمارش تعداد تکرار حلقه را به عهده دارد. شمارش می‌تواند به صورت صعودی یا تزولی انجام شود. بدیهی است در حالت تزولی، تغییر مقدار متغیر باید به صورت کاهشی باشد.

شکل‌های دیگر حلقه مثال ۶-۷ در زیر آورده شده است.

```
for (int i = 1; i < 10; Console.WriteLine(i), i++);
```

**سؤال:** خروجی این دستور را بررسی کنید.

```
for (int i = 1; i < 10; i++, Console.WriteLine(i));
```

**سؤال:** خروجی این دستور را بررسی کنید.

**مثال ۷-۷** : در دستور زیر از یک حلقه نزولی استفاده شده است. به هر سه قسمت داخل پرانتز دقت کنید.

```
for (int i = 100 ; i > 1 ; i -- )
```

```
    Console.WriteLine( i);
```

قطعه برنامه ۷-۷—استفاده از حلقه **for** کاھشی

**سؤال :** به نظر شما خروجی این دستورات چیست؟ (چه اعدادی روی صفحه نمایش،  
شان داده می‌شود؟)

کاربرد **break** در ساختار **for** : قطعه برنامه زیر را در نظر بگیرید :

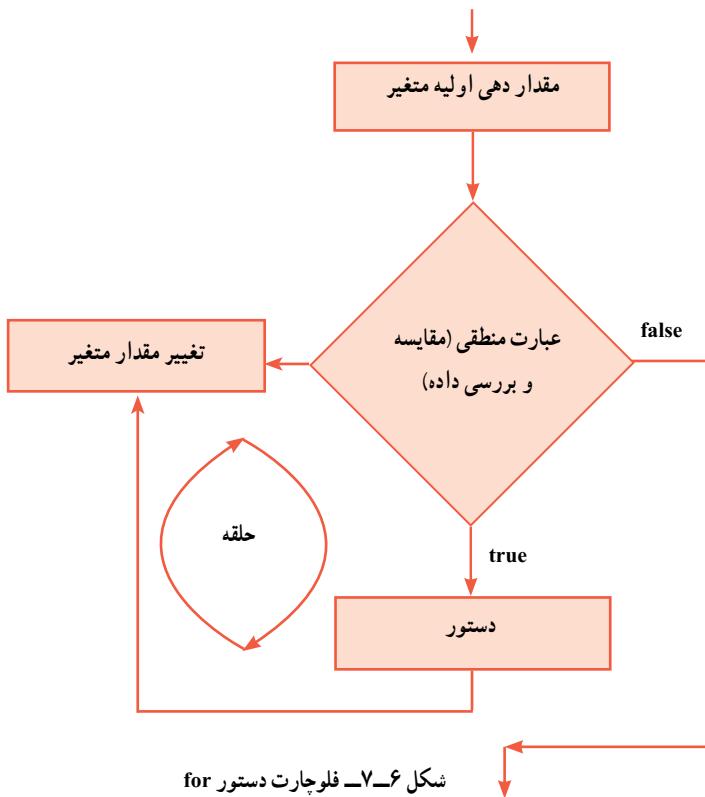
```
int i;  
  
for(i = 20 ; i > 1 ; i -- )  
  
{  
    if(i == 5)  
        break;  
  
    Console.WriteLine(i);  
  
}  
  
Console.WriteLine("broke out of loop at i of " + i);
```

در این قطعه برنامه از دستور **break** برای خروج از حلقه در صورت برقراری شرط  $(i == 5)$  استفاده شده است. این حلقه اعداد ۲۰ تا ۶ را به صورت نزولی چاپ می‌کند و با رسیدن به عدد ۵ توسط دستور **break** حلقه خاتمه می‌یابد.

توجه داشته باشید، که هیچ کدام از سه قسمت داخل پرانتز، در دستور **for** اجباری نیستند. حتی دستور **for**، می‌تواند به صورت زیر نوشته شود که در این صورت، یک حلقه تمام نشدنی و بی‌نهایت ایجاد می‌شود.

```
for(;;)  
    Console.WriteLine("Infinite Loop! ");
```

**شكل ۷-۶**، فلوچارت دستور **for** را نشان می‌دهد. در این فلوچارت، دستور ایجاد و مقداردهی اولیه متغیر، در داخل حلقه قرار ندارد و تنها یک بار در ابتدا اجرا می‌شود.



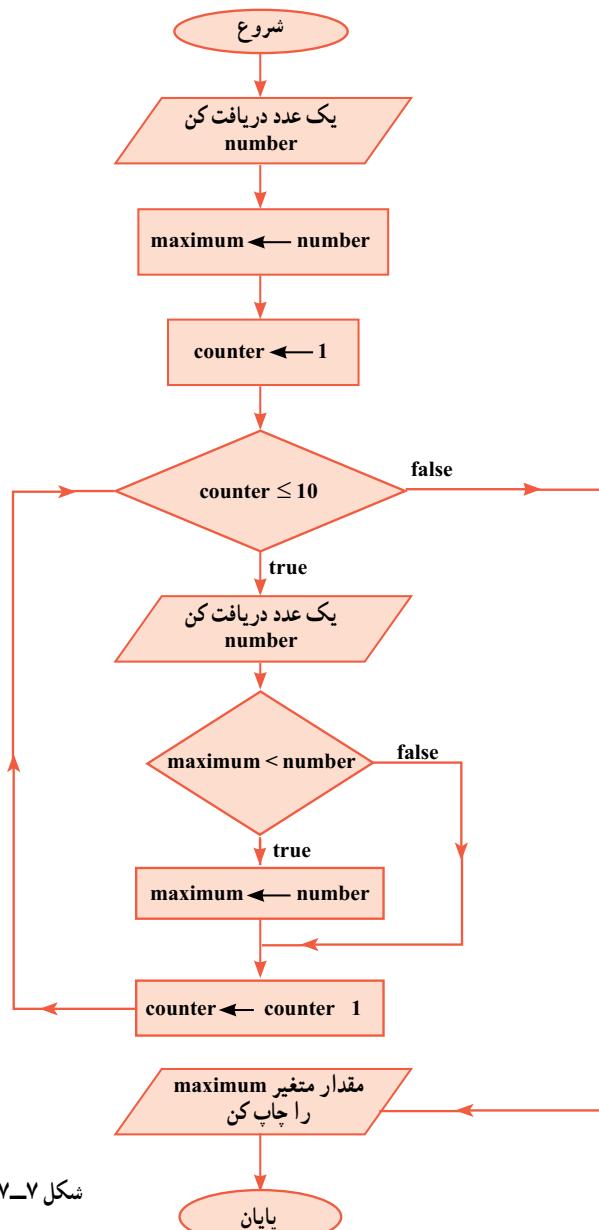
## نکته

به علامت ( ; ) در دستور for توجه کنید. بعد از علامت پرانتز علامت ( ; ) نگذارید، زیرا دستور for هنوز تمام نشده است. علامت ; باید در انتهای دستور نوشته شود.  
 (تغییر مقدار متغیر ; عبارت منطقی ; مقدار اولیه نام متغیر) for  
 ; دستور ←

**مثال ۶-۷:** می‌خواهیم ده عدد از ورودی دریافت کرده، بزرگ‌ترین آن را تشخیص داده و نمایش دهیم.

الگوریتم یا روش انجام کار : برای پیدا کردن بزرگ‌ترین عدد، از روشی که در فصل ششم در مثال ۶-۳، توضیح داده شد، استفاده می‌کنیم. شکل ۶-۷، فلوچارت پیدا کردن بزرگ‌ترین عدد، از بین ده عدد را نشان می‌دهد. در ابتدای این فلوچارت، اولین عدد دریافت می‌شود و آن را به عنوان

بزرگترین عدد در متغیر maximum ذخیره می‌کنیم. سپس در داخل حلقه، بقیه اعداد دریافت و با مقدار متغیر maximum مقایسه می‌شوند. هر جا که عدد بزرگتری دریافت شد، در متغیر maximum ذخیره می‌شود. چون حلقه باید به تعداد ۹ بار تکرار گردد، از یک متغیر به نام counter برای شمارش تعداد تکرار حلقه استفاده شده است، که با هر بار تکرار حلقه، یک واحد به آن اضافه می‌شود.



شكل ۷-۷ – فلوچارت پیدا کردن بزرگترین عدد

برنامه ۷-۸ بر اساس فلوچارت شکل ۷-۷ نوشته شده است.

```
class FindMaximum
{
    static void Main(string[] args)
    {
        string input;
        int number, maximum;
        Console.WriteLine("Enter a number: ");
        input = Console.ReadLine();
        maximum = int.Parse(input); // Suppose first number is maximum.

        for (int i = 2; i < 10; i++)
        {
            Console.WriteLine("Enter a number: ");
            input = Console.ReadLine();
            number = int.Parse(input);
            if (maximum < number) // found bigger number
                maximum = number;
        }

        Console.WriteLine("The maximum number is " + maximum);
        Console.WriteLine("Press any key to continue . . .");
        Console.ReadKey();
    }
}
```

دریافت دومین عدد تا  
دهمین در حلقه

دریافت اولین عدد

نامه ۸-۷-ساده که دن بزگت بیان عدد از سه تا هشت

## کار در کارگاه ۳

- ۱- قطعه برنامه ۳-۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.
- ۲- برنامه ۵-۷ را نوشته و اجرا کنید. این بازی را با هم گروه خود انجام دهید.
- ۳- به برنامه ۵-۷، امکان امتیازدهی را اضافه نمایید. برای این منظور، با استفاده از یک متغیر، تعداد دفعات تلاش بازیکن دوم را شمارش کرده و در انتهای برنامه، محتوای آن را نشان دهید. یک دستور انتساب افزایشی در داخل حلقه می‌تواند تعداد دفعات را شمارش کند.
- ۴- قطعه برنامه‌های ۶-۷ و ۷-۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.
- ۵- برنامه ۸-۷ را نوشته و اجرا کنید. ده عدد مختلف وارد کرده و نتیجه را مشاهده کنید.
- ۶- با تغییراتی در برنامه ۸-۷، کوچکترین عدد را پیدا کنید. نام متغیر minimum را برای کوچک‌ترین عدد استفاده کنید.
- ۷- با استفاده از بند ۸ و ۹، برنامه‌ای بنویسید که هم بزرگ‌ترین و هم کوچک‌ترین عدد را از بین ده عدد پیدا کند.
- ۸- به بند ۱°، دستوری اضافه کنید تا فاصله بین بزرگ‌ترین عدد و کوچک‌ترین عدد را پیدا کند.

## تمرینات برنامه‌نویسی فصل هفتم

- ۱- برنامه‌ای بنویسید که اعداد زوج از ۳ تا ۲۱ را به صورت تزولی نشان دهد. خروجی برنامه را روی عدد ۱۶ متوقف کنید.
- ۲- برنامه‌ای بنویسید که یک عدد صحیح را دریافت کند و سپس اعداد فرد از ۱ تا آن عدد را چاپ نماید.
- ۳- برنامه‌ای بنویسید که یک عدد صحیح را دریافت کند و مقلوب آن را محاسبه و نمایش دهد. مثلاً اگر عدد ۵۲۹ وارد شد برنامه عدد ۹۲۵ را نمایش دهد.
- ۴- برنامه‌ای بنویسید که نمرات درس انگلیسی دانش آموزان یک کلاس ۱۵ نفری را سؤال نماید و سپس اطلاعات زیر را نمایش دهد :  
الف) بالاترین نمره کلاس

ب) کمترین نمره کلاس

پ) فاصله بین کمترین و بیشترین نمره

ت) مجموع نمرات کلاس

ث) میانگین یا معدل نمرات کلاس

۵- اجرای دستورات زیر سبب نمایش چه اعدادی می‌شود؟

```
int a = 1, b = 1, c = 0;  
while (a < 30)  
{  
    Console.WriteLine(a);  
    c = a - b;  
    a = b;  
    b = c;  
}
```

۶- در یک بازی دو نفره، هفت چوب کبریت قرار دارد. هر یک از بازیکنان می‌توانند در نوبت خود یک یا دو یا حداکثر سه چوب کبریت بردارند. بازیکنی که آخرین چوب کبریت را بردارد، بازنده است. برنامه‌ای بنویسید که این بازی را بین دو بازیکن اجرا کند. در هر مرحله بازی، تعداد چوب کبریت‌های باقیمانده را چاپ کنید. سپس از هر بازیکن، تعداد چوب کبریت‌هایی که مایل است بردارد را سؤال نموده و باقیمانده را چاپ کنید.

## پیوست ۱ : نصب Visual Studio

شرکت مایکروسافت نرم افزاری به نام Visual Studio ساخته است که برای برنامه نویسی استفاده می شود این نرم افزار شامل مترجم زبان برنامه نویسی، یک ویرایشگر<sup>۱</sup> و یک اسکالیاب می باشد. این نرم افزار در چند نسخه با ویژگی های مختلف عرضه شده است. همچنین یک نسخه رایگان به نام Visual Studio Express Edition عرضه کرده است که برای شروع کار برنامه نویسی مناسب است. دیگر نرم افزار رایگان، نرم افزار Microsoft .Net Framework است که یک دسته ابزار مختلف به صورت فرمان (که در خط فرمان باید تایپ شوند) فراهم می کند که برای ترجمه و اجرای برنامه نوشته شده به زبان #C می تواند مورد استفاده قرار گیرد. بنابراین شما برای برنامه نویسی و اجرای برنامه ها دو راه دارید :

- ۱- استفاده از Visual Studio که نیاز به تهیه و نصب نرم افزار VS دارد که در ادامه این ضمیمه طریقه نصب آن بیان شده است.
- ۲- استفاده از ابزارهای خط فرمان : نیاز به نصب .NET. دارد که در پیوست ۲ طریقه نصب آن را خواهید دید.

## پیوست ۲ : نصب Net Framework

برای نصب Net Framework. مراحل زیر را دنبال کنید :

- ۱- از طریق موتور جستجو گری نظری گوگل، برنامه Net Framework را جستجو کرده و وارد سایت شرکت مایکروسافت شوید یا به آدرس زیر مراجعه کنید :  
<http://www.microsoft.com/net>
- ۲- در صفحه مربوط به دانلود برنامه .Net Framework روی نسخه .NET. موردنظر جهت دانلود کلیک کنید.

<sup>۱</sup>\_Ed tor

مشکلات احتمالی که ممکن است در ترجمه یا اجرای برنامه پیش آید:

الف) اشکال در ترجمه برنامه

- ۱- اگر در هنگام ترجمه برنامه با پیام خطای زیر روبرو شدید احتمال دارد که برنامه .NET. بر روی سیستم شما نصب نشده باشد لذا باید این برنامه را نصب کنید. (به ضمیمه ۲ رجوع شود.)
- ۲- اگر علیرغم نصب برنامه .NET. همچنان خطای پیدا نکردن مترجم csc رخ می دهد، حتماً در مسیر جستجو (Path)، فولدر حاوی برنامه csc معرفی نشده است. فولدر حاوی متوجه معمولاً در مسیر زیر قرار دارد :

C:/ Windows / Microsoft.NET/ Framework/ v4.0.30319

البته بسته به نسخه .NET. شماره های v4.0.30319 و ۴.۰.۳۰۳۱۹ متفاوت می باشد. این شماره مربوط به نسخه .NET4.5 است.

برای اضافه کردن فولدر مربوطه به مسیر جستجو، عملیات زیر را انجام دهید :

اگر سیستم عامل کامپیوتر شما ویندوز ۷ یا وینتاو ۸ باشد بعد است :

- ۱- در میز کار روی آیکن Computer کلیک راست کنید و سپس Properties را انتخاب کنید.

۲- در سمت چپ صفحه روی گزینه Advanced System Setting کلیک کنید :

۳- در پنجره System Properties روی گزینه Environment Variables کلیک کنید.

۴- با کلیک بر روی متغیر Path آن را انتخاب کرده و سپس روی کلید Edit کلیک کنید.

- ۵- در پنجره ای که ظاهر می گردد مسیر برنامه های مورد جستجو نوشته شده است مسیر NET. را اضافه کنید.

ب) اشکال در اجرای برنامه

اگر در هنگام ترجمه برنامه با اشکال زیر روبرو شدید ممکن است یکی از دو اشتباه زیر را انجام داده باشید :

۱- نام فایل را درست تایپ نکرده اید.

۲- پسوند فایل برنامه را در بعد از نام فایل نوشته اید.

## پیوست ۴ : لیست کلمات کلیدی و رزرو شده

abstract	as	base	bool
break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	in (generic modifier)	int	interface
internal	is	lock	long
namespace	new	null	object
operator	out	out (generic modifier)	override
params	private	protected	public
readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc
static	string	struct	switch
this	throw	true	try
typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual
void	volatile	while	

