بسم الله الرحمن الرحیم

برنامه‌سازی ۱

رشتهٔ کامپیوتر

گروه تحصیلی کامپیوتر

زمینهٔ خدمات

شاخهٔ آموزش فنی وحرفه‌ای

عنوان و نام پدیدآور	:	برنامه‌سازی ۱ : رشته کامپیوتر گروه تحصیلی کامپیوتر زمینه خدمات شاخه آموزش فنی و حرفه‌ای.
مشخصات نشر	:	تهران : شرکت چاپ و نشر کتاب‌های درسی ایران، ۱۳۹۴
مشخصات ظاهری	:	۱۷۶ ص. : مصور.
شابک	:	۹۷۸-۹۶۴-۰۵-۲۳۳۵-۳
وضعیت فهرست‌نویسی	:	فیپای مختصر
یادداشت	:	این مدرک در آدرس http: /opac . nlai . ir قابل دسترسی است.
شناسه افزوده	:	کربلایی، مجید
شناسه افزوده	:	سازمان پژوهش و برنامه‌ریزی آموزشی. دفتر تألیف کتاب‌های درسی فنی وحرفه‌ای و کاردانش
شماره کتاب‌شناسی ملی	:	۳۵۹۳۹۲۱

۱۳۹۴

همکاران محترم و دانش‌آموزان عزیز :

پیشنهادات و نظرات خود را دربارهٔ محتوای این کتاب بـه نشانی تهران ـ صندوق پستی شمارهٔ ۴۸۷۴/۱۵ دفتر تألیف‌کتاب‌های درسی فنی و حرفه‌ای و کاردانش، ارسال فرمایند.

پیام‌نگار (ایمیل) **info@tvoccd.sch.ir**

وب‌گاه (وب‌سایت) **www.tvoccd.sch.ir**

وزارت آموزش و پرورش

سازمان پژوهش و برنامه‌ریزی آموزشی

برنامه‌ریزی محتوا و نظارت بر تألیف : **دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کاردانش**

نام کتاب : **برنامه‌سازی ۱ ـ ۳۵۸/۷۰**

مؤلف : **مجید کربلایی**

اعضای کمیسیون تخصصی : **محمدرضا یمقانی، بتول عطاران، شروین الوندی، مهیار پازوکی، شهناز علیزاده، نیلوفربزرگ‌نیا، سارو آواکیانس، محمدرضا شکرریز، مهناز کارکن، افشین اکبری، سید سعیدرضا سعادت‌یزدی، پردیس پیرایش و زهرا عسگری رکن‌آبادی**

ویراستار علمی : **مهناز کارکن، پردیس پیرایش و زهرا عسگری رکن آبادی**

ویراستار ادبی : **محمدرضا یمقانی**

آماده‌سازی و نظارت بر چاپ و توزیع : **ادارهٔ کل نظارت بر نشر و توزیع مواد آموزشی**

تهران : خیابان ایرانشهر شمالی ـ ساختمان شمارهٔ ۴ آموزش و پرورش (شهید موسوی)

تلفـن : ۹ـ ۸۸۸۳۱۱۶۱، دورنگار : ۸۸۳۰۹۲۶۶، کد پستـی :۱۵۸۴۷۴۷۳۵۹،

وب سایت : www.chap.sch.ir

مدیر امور فنی و چاپ : **لیدا نیک‌روش**

طراح جلد : **ناهید معین‌الرعایا**

صفحه‌آرا : **راحله زادفتح‌اله**

حروفچین : **کبریٰ اجابتی، زهرا ایمانی نصر**

مصحح : **علی مظاهری نظری‌فر، مریم جعفرعلیزاده**

امور آماده‌سازی خبر : **زینت بهشتی‌شیرازی**

امور فنی رایانه‌ای : **طوبیٰ عطایی، سیده شیوا شیخ‌الاسلامی**

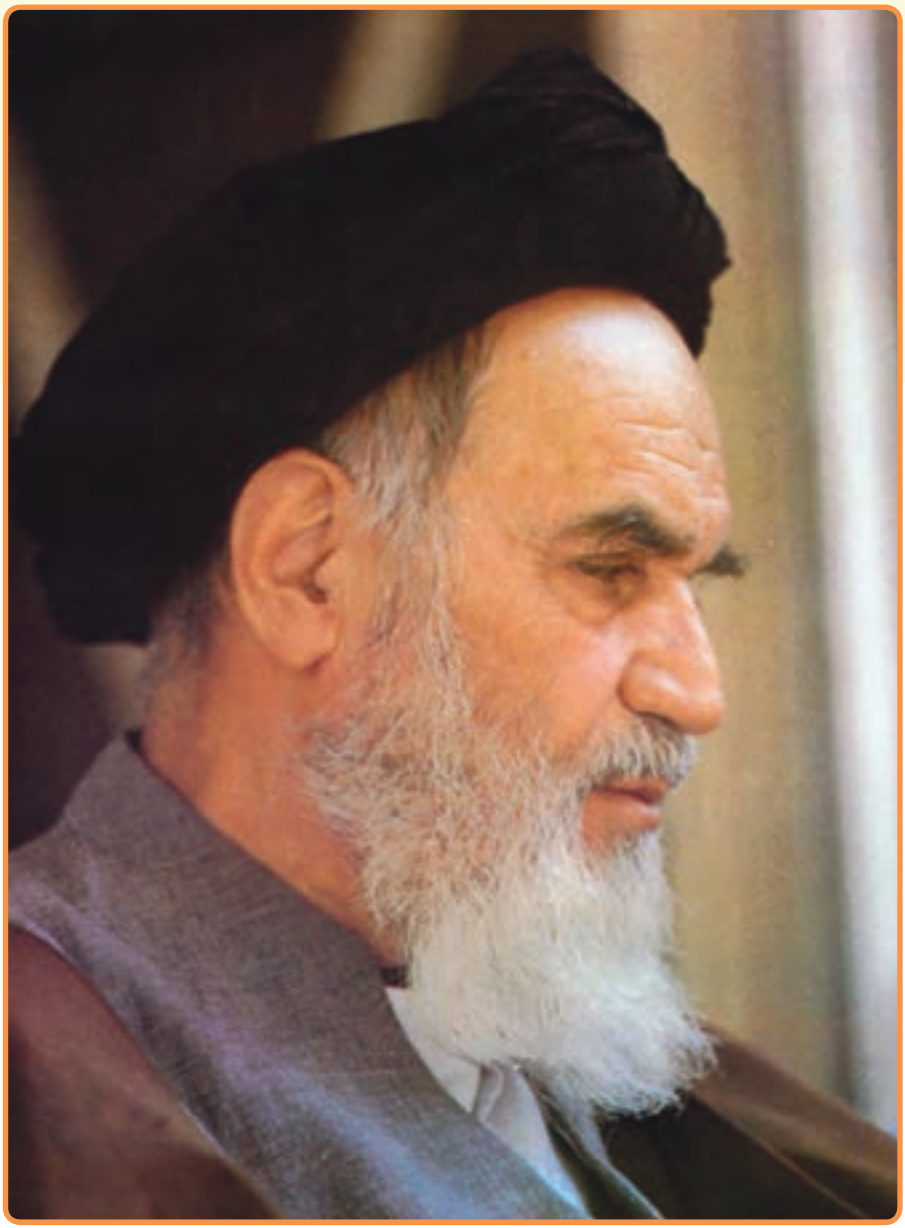
ناشر : **شرکت چاپ و نشر کتاب‌های درسی ایران : تهران ـ کیلومتر ۱۷ جادّهٔ مخصوص کرج ـ خیابان ۶۱ (دارو پخش)**

تلفن : ۵ ـ ۴۴۹۸۵۱۶۱، دورنگار : ۴۴۹۸۵۱۶۰، صندوق پستی : ۱۳۹ـ ۳۷۵۱۵

چاپخانه : **شرکت چاپ و نشر کتاب‌های درسی ایران «سهامی خاص»**

سال انتشار و نوبت چاپ : **چاپ دوم، ۱۳۹۴**

شابک ۳-۲۳۳۵-۰۵-۹۶۴-۹۷۸ ISBN 978-964-05-2335-3



بدانید مادام که در احتیاجات صنایع پیشرفته، دست خود را پیش دیگران دراز کنید و به دریوزگی عمر را بگذرانید، قدرت ابتکار و پیشرفت در اختراعات در شما شکوفا نخواهد شد.

امام خمینی «قدّس سرّه الشّریف»

مقدمه

درس برنامه‌سازی ۱، یک درس پایه برای هنرجویان رشته کامپیوتر فنی و حرفه‌ای است که هدف آن آموزش یک زبان برنامه‌نویسی است. در این کتاب سعی بر آن شده است که زبان برنامه‌نویسی C# با زبانی ساده و قابل فهم برای هنرجویان و دانش‌آموزان بیان شود. در ابتدای هر فصل هدف‌های رفتاری بیان شده است و در انتهای هر فصل به‌جز فصل اول بخش کار در کارگاه درنظر گرفته شده است که باید در کارگاه و به‌صورت عملی انجام شود. خودآزمایی، تمرینات برنامه‌نویسی و جدول واژگان انگلیسی نیز در انتهای هر فصل قرار دارد که هنرجویان عزیز باید آنها را انجام دهند تا به اهداف رفتاری ذکر شده در ابتدای هر فصل برسند.

در فصل اول مفاهیم پایه‌ای پردازش داده‌ها و تقسیم‌بندی زبان‌های برنامه‌نویسی یادآوری می‌شود. در فصل دوم پیدایش زبان C# و ارتباط آن با زبان‌های دیگر بیان شده است و سپس یک برنامه ساده و نحوه ترجمه و اجرای آن از طریق خط فرمان توضیح داده شده است. در فصل سوم با محیط برنامه‌نویسی ویژوال استودیو به‌عنوان ابزاری قدرتمند برای برنامه‌نویسی آشنا می‌شوید. در فصل چهارم متغیر و انواع آن بیان شده است. در فصل پنجم با عبارت‌ها و عملگرهای ریاضی و محاسباتی آشنا می‌شوید. در فصل ششم دستورات شرطی و کنترل اجرای دستورات توضیح داده شده است. و بالاخره در فصل هفتم ساختارهای تکرار و حلقه‌ها بحث می‌شود که توانایی نوشتن برنامه‌های کاربردی را به‌دست می‌آورید. آنچه که از هنرجویان انتظار می‌رود انجام تمرینات بخش کار در کارگاه و حل تمرینات و پاسخ به خودآزمایی‌ها است که با انجام آن زمینه لازم برای یادگیری مطالب کتاب فراهم می‌شود و هنرجویان برای درس برنامه‌سازی ۲ و ۳ در سال بعد آماده می‌شوند. با وجود دقتی که در تألیف این کتاب صورت گرفته است، کتاب عاری از اشتباه نیست. ایده‌ها و پیشنهادات شما باعث بهبود کیفیت کتاب خواهد شد لذا مواردی که از نظر شما بایستی اصلاح شوند را به اطلاع ما برسانید.

مجید کربلایی

فهرست

اول آشنایی با مفاهیم پایه‌ای پردازش داده‌ها

دوم آشنایی با زبان C#

سوم آشنایی با ویژوال استودیو

چهارم آشنایی با انواع داده و متغیرها

پنجم عبارت‌های محاسباتی

ششم دستورهای شرطی

هفتم دستورات تکرار (حلقه‌ها)

۱ 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 نزدیکی به‌زبان محاوره‌ای و لزوم یادگیری یک زبان برنامه‌نویسی آشنا می‌شوید.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

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

۱ــ۱ــ داده‌ها و اطلاعات[۱]

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 به‌صورت کمّی با یک مقدار عددی و یا به‌صورت کیفی نشان داده می‌شود.

مثلاً آزمون درس ریاضی یک کلاس، نمراتی حاصل می‌شوند که این نمرات به‌عنوان داده‌ها درنظر گرفته می‌شوند.

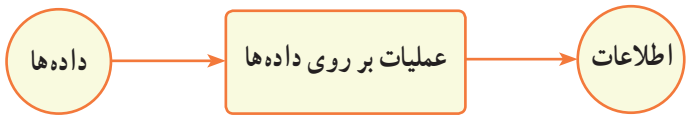
۱۷/۵ , ۱۴ , ۱۹ , ۱۱ , ۱۸ , ...

۱ــ Data & information

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

در بعضی موارد نیز برای به‌دست آوردن داده‌های مربوط به یک موضوع، مجبور به استفاده از لوازم و وسایل مخصوص هستیم، مثلاً در مورد دمای محیط، نیاز به استفاده از یک حسگر[1] دما هستیم که به‌وسیلهٔ آن اندازه دما را به‌دست آوریم. اندازه دمای محیط، یک داده است.

برای اینکه از داده‌ها بتوانیم بهتر استفاده کنیم لازم است بر روی آنها محاسبات و یا به‌طور 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 داده‌ها را، نمرات ریاضی یک کلاس در نظر بگیریم و مجموع آنها را محاسبه و حاصل جمع را بر تعداد نمرات تقسیم کنیم، خارج قسمت به‌دست آمده میانگین یا معدل نمرات است. از روی میانگین نمرات می‌توان به سطح درس ریاضی کلاس پی 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 ۱ــ۱ــ ارتباط بین داده‌ها، عملیات و اطلاعات

با توجه به شکل ۱ــ۱، مشاهده می‌شود که اطلاعات، حاصل انجام عملیات بر روی داده‌ها است. مجموعه محاسبات و عملیاتی که بر روی داده‌ها صورت می‌گیرد را **پردازش**[2] 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.

۱ــ Sensor
۲ــ Process

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

۱ــ صحت داده‌ها : یعنی داده‌ها به درستی گردآوری شده باشند و دادهٔ 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 دانش‌آموزان برابر باشد. و یا در مورد اندازه درجه حرارت محیط، سنسور دما به‌خوبی کار کند و معیوب نباشد و دمای محیط به‌درستی اندازه‌گیری شود.

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

۳ــ 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.

معمولاً از 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 پیرامون ما دستگاه‌های مختلفی قرار دارند که در هر یک از آنها میکروکامپیوتری برای پردازش داده‌ها وجود دارد. مثلاً در یک دستگاه DVD Player با استفاده از لیزر و حسگر، علامت‌های روی دی وی دی به‌عنوان 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 تحویل سوخت به موتور به‌کار می‌رود. داده‌های ورودی این کامپیوتر توسط حسگرهای مختلفی که برای اندازه‌گیری مقدار اکسیژن در هوا، اندازه دمای هوا و اندازه سرعت موتور و غیره تعبیه 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!

کنجکاوی

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

به‌مجموعه دستوراتی که به کامپیوتر می‌فهماند که چه نوع پردازشی را بر روی داده‌ها انجام دهد و همچنین اطلاعات به‌دست آمده را چگونه نمایش دهد برنامه[1] می‌گویند.

دستورات برنامه باید با یک زبان قابل فهم برای کامپیوتر، نوشته شود و با توجه به اینکه سخت‌افزار کامپیوتر بر منطق رقمی (صفر و یک) بنا شده است لذا زبان قابل فهم کامپیوتر دنباله‌ای از کدهای صفر و یک است که به آن زبان **ماشین**[2] می‌گویند. مثلاً برای جمع دو عدد ۵ و ۸ دستورهای زبان ماشین در یک پردازنده شرکت اینتل[3] چنین است :

10111000 00000101 00000000 10111011 00001000 00000000 00000001 11011000

با توجه به اینکه نوشتن دستورات به زبان ماشین وقت‌گیر و دشوار است و زبان ماشین هر پردازنده با پردازنده دیگر متفاوت است، زبان‌های دیگری طراحی شده‌اند که نوشتن 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[4]، باید به زبان ماشین تبدیل شود تا کامپیوتر بتواند آن را بفهمد و اجرا نماید. مترجم خود نیز یک برنامه کامپیوتری می‌باشد که وظیفهٔ آن، ترجمه و تبدیل دستورات یک زبان سطح بالا، به کدهای زبان ماشین می‌باشد.

نوشتن دستورات لازم برای کنترل نحوه کار کامپیوتر، به‌طوری که کامپیوتر بتواند یک کار مشخص را انجام دهد را **برنامه‌نویسی** می‌گویند.

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

۱ــ Program
۲ــ Machine Language
۳ــ Intel
۴ــ Compiler

نکته

به مجموعه دستوراتی که به کامپیوتر می‌فهماند که چه نوع پردازشی را باید بر روی داده‌ها انجام دهد و اطلاعات به‌دست آمده را چگونه نمایش دهد برنامه می‌گویند.

زبان قابل فهم سخت افزار کامپیوتر، زبان ماشین نام دارد و متشکل از دنباله‌ای از کدهای ۰ و ۱ است.

برنامه‌ای که با زبانی غیر از زبان ماشین نوشته شود ابتدا باید توسط یک نرم‌افزار مترجم تبدیل به زبان ماشین شود و سپس برنامه ترجمه شده توسط کامپیوتر اجرا می‌گردد.

مترجم خود یک برنامه کامپیوتری است که وظیفه آن ترجمه برنامه نوشته شده به یک زبان، به کدهای زبان ماشین است.

۲ــ۱ــ انواع زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی را از نظر این‌که تا چه اندازه به زبان محاوره‌ای ما نزدیک باشند به‌صورت زیر دسته‌بندی می‌کنند :

● **زبان‌های سطح پایین**[۱] : زبان‌هایی که به زبان پردازشگر کامپیوتر (CPU) نزدیک باشد و مسلماً از زبان محاوره‌ای ما دور هستند زبان‌های سطح پایین نام دارند. زبان ماشین و زبان اسمبلی[۲] در گروه زبان‌های سطح پایین قرار دارند.

● **زبان‌های سطح بالا**[۳] : زبان برنامه‌نویسی که به زبان محاوره‌ای ما نزدیک باشد، زبان سطح بالا نام دارد. تاکنون صدها زبان برنامه‌نویسی سطح بالا در دانشگاه‌ها و یا شرکت‌های کامپیوتری طراحی و ابداع شده‌اند ولی بعضی از آنها عمومی نشدند و مورد استقبال برنامه‌نویسان قرار نگرفتند 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**[۴] : زبان‌هایی در این دسته قرار می‌گیرند که در آنها دستوراتی برای دسترسی راحت تر به سخت افزار پیش بینی شده باشد و همچنین به زبان عامیانه نزدیک باشند، مانند زبان C. برنامه‌نویسان از این زبان‌ها برای کار با سخت افزار کامپیوتر و برنامه‌ریزی وسایلی که در آنها

۱ــ Low Level Language
۲ــ Assembly Language
۳ــ High Level Language
۴ــ Medium Level 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[1] آماده نظیر Office، آیا لزومی‌دارد که یک زبان برنامه‌نویسی را یاد بگیریم تا یک نرم افزار جدید بسازیم؟

پاسخ: نرم‌افزارهای کاربردی آماده، برای تسهیل و انجام امور عادی و روزمره که بین کاربران مشترک است طراحی شده‌اند و به تدریج و در طی سالیان طولانی با توجه به بازخورد[2] کاربران تکمیل‌تر شده‌اند. مثلاً برنامه صفحه گسترده[3] (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 مرتفع 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[4] و مطابق با سلیقه و مدل مورد نظرما درست کند.

البته طراحی و تولید یک نرم افزار کاربردی، یک کار سنگین، پیچیده و مستلزم صرف وقت زیادی است و به وسیلهٔ 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 نرم‌افزار و روش‌های طراحی و حل مسئله[5] 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.

۱ــ Application Software

۲ــ Feedback

۳ــ Spreadsheet

۴ــ Customized

۵ــ Problem Solving

شکل ۲ــ۱ــ نظر یک برنامه‌نویس

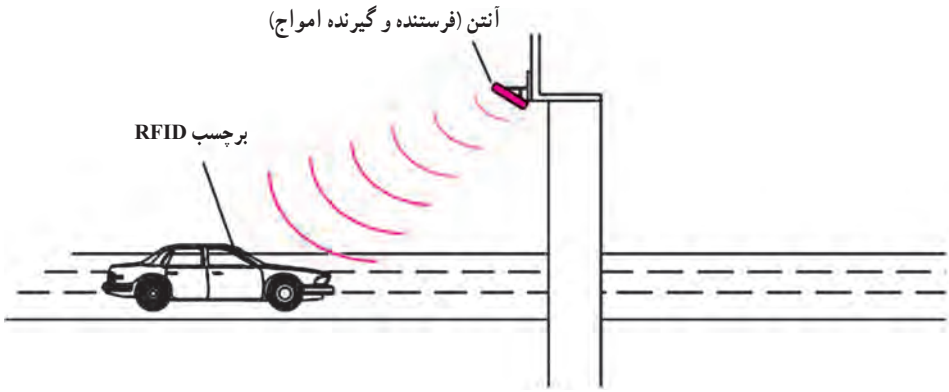
خودآزمایی فصل اول

۱ــ 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 حضورساعات کار در شرکت محاسبه و پس از کسر مالیات و حق بیمه پرداخت می‌شود و بر روی فیش چاپ می‌شود.

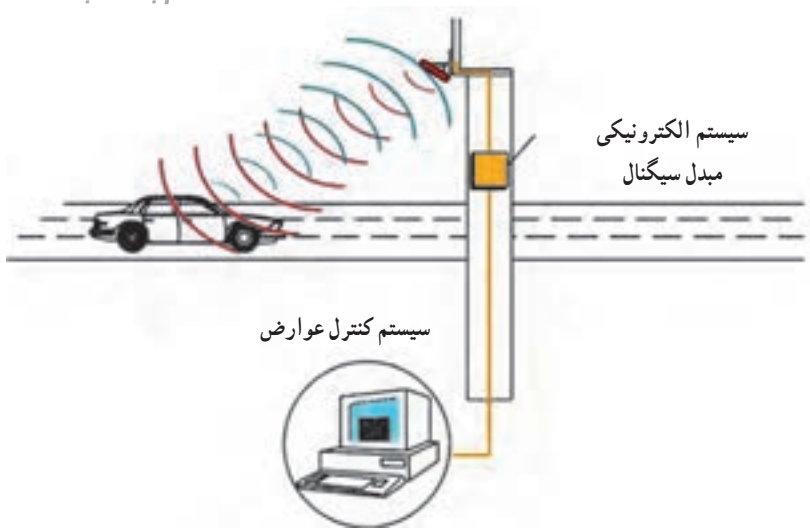
ج) یک روش الکترونیکی در پرداخت عوارض خودرو در هنگام عبور از بزرگراه‌ها، استفاده از تکنولوژی RFID[1] است. در این روش یک برچسب[2] مخصوص در روی شیشه خودرو چسبانده می‌شود که دارای یک کد منحصر به فرد است و اطلاعات شماره پلاک خودرو و کد مربوطه و همچنین میزان اعتبار پولی در هنگام خرید برچسب، در کامپیوتر ثبت می‌شود. هر گاه خودرو بدون توقف از محل پرداخت عوارض عبور کند از طریق حس گرهای امواج رادیویی، کد برچسب خودرو شناسایی شده و این کد به کامپیوتر داده می‌شود (شکل ۳ــ۱) و در نتیجه شماره پلاک خودرو و میزان اعتبار آن استخراج می‌شود، عوارض عبور از بزرگراه از اعتبار مربوطه به صورت خودکار کسر می‌شود (شکل ۴ــ۱).



شکل ۳ــ۱ــ مرحله اول ورود خودرو به گذرگاه عوارض بزرگراه

۱ــ Radio_Frequency Identification شناسایی با امواج رادیویی

۲ــ Tag



شکل ۴ــ۱ــ مرحله دوم تشخیص اطلاعات خودرو و کسر عوارض از اعتبار پولی

۲ــ حدس بزنید چه سطحی از زبان برنامه‌نویسی هستم؟

تاکنون 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.

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

۱ــ برنامه	الف ــ برنامه‌ای برای تبدیل کد سطح بالا به کد قابل فهم برای سخت‌افزار
۲ــ برنامه‌نویسی	ب ــ دنباله‌ای از کدهای صفر و یک
۳ــ مترجم	ج ــ مجموعه‌ای از دستورالعمل‌ها برای پردازش داده‌ها
۴ــ زبان ماشین	د ــ نوشتن دستورات لازم برای حل یک مسئله
۵ــ زبان سطح بالا	هـ ــ زبان برنامه‌نویسی نزدیک به زبان محاوره‌ای
	وــ داده‌ها و ورودی‌های یک برنامه

۴ــ در یک برنامه، منظور از ورودی همان ..... و منظور از خروجی همان ..... است.

۵ــ در محاسبهٔ عبارت $z=x^2+y$، ورودی (ها) و خروجی (ها) کدامند؟

۶ــ 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.

۷ــ چرا لازم است یک زبان برنامه‌نویسی را یاد بگیریم؟

فعالیت

۱ــ با استفاده از جستجو در اینترنت، چند زبان سطح بالای متداول امروزی را پیدا کرده و آنها را نام ببرید.

۲ــ دستگاه‌های الکترونیکی و یا کامپیوتری 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.

۳ــ اگر به اتفاق پدر یا مادر به تعمیرگاه اتومبیل رفتید، فرصت را از دست ندهید و از تعمیرکاران سؤال کنید که دستگاه ECU (توضیح داده شده در قسمت «آیا می‌دانید») در کجای خودرو قرار دارد؟ (در همه خودروهای امروزی حتی مانند پراید نیز وجود دارد. تعجب نکنید.) ECU و حس گرهای آن را شناسایی کنید. همچنین وقتی به منزل برگشتید با استفاده از اینترنت منظور از کلمات دیاگ و فلش‌کردن که توسط تعمیرکاران استفاده می‌شود را جستجو کرده و متوجه مفهوم آن شوید.

واژگان و اصطلاحات انگلیسی فصل اوّل

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	

۲ 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 متد 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 فراگیر بتواند:

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

۱ـ۲ـ آشنایی با زبان C#

زبان C# یک زبان سطح بالا، شی‌گرا[۱] و همه منظوره است که به‌وسیله 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 بر خلاف زبان جاوا که متن باز[۲] است، C# در انحصار و اختیار سازنده آن یعنی شرکت مایکروسافت است. زبان‌های C++

۱ـ Object Oriented Language ۲ـ Open Source

و Java هر دو به زبان C برمی‌گردند که در سال ۱۹۷۰ ابداع شد و معروفیت آن به دلیل نوشتن سیستم‌عامل UNIX به‌وسیله آن بود. زبان C، یک زبان حرفه‌ای است که دست برنامه‌نویس را برای نوشتن برنامه و دسترسی به سخت‌افزار، باز می‌گذارد و دارای انعطاف بسیار 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 به‌کارگیری داده‌ها صورت می‌گیرد تا از اشتباهات دستوری برنامه‌نویس یا کاربر جلوگیری نماید.

برای مطالعه

DENNIS M.RITCHIE

دنیس ریچی متولد ۱۹۴۱ زبان C را بر مبنای زبان B، طی سال‌های ۱۹۶۹ تا ۱۹۷۳ طراحی کرد و همچنین با آقای 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.

BJARNE STROUSTRUP

بیارنه استراس تروپ، 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.

JAMES ARTHUR GOSLING

جیمز آرتگاسلینگ متولد سال ۱۹۵۵، به عنوان پدر زبان جاوا شناخته می‌شود. وی در سال ۱۹۹۴ به همراه تیمش زبان جاوا را بر پایه زبان C و C++ ابداع کرد وی در ابتدا نام Oak که نام درختی در جلوی دفتر کار او بود، برای این زبان انتخاب کرد، اما بعدها به Java تغییر نام پیدا کرد. زبان جاوا با هدف ساده کردن و بالابردن امنیت و با شعار برنامه را یک بار بنویس و هر جا اجرا کن طراحی شده است.

ANDERS HEJLSBERG

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

شکل ۱ــ۲ــ پدید آورندگان زبان‌های برنامه‌نویسی C# , Java , C++ , C

برای مطالعه

لایه نرم‌افزاری .NET Framework. شرکت مایکروسافت

این لایه نرم‌افزاری دارای نسخه‌های مختلفی می‌باشد و همراه با نسخه‌های ویندوز منتشر می‌شود؛ آخرین نسخه آن را می‌توانید از سایت شرکت مایکروسافت دانلود کنید. در حال حاضر (فروردین ۱۳۹۳) نسخه 4.5 ارائه شده است. این لایه شامل موارد زیر است :

- زبان‌های برنامه‌نویسی .NET. مانند C#، J# ، F# و VB
- محیطی برای اجرای برنامه‌ها (CLR[1])
- یک سری ابزارهای برنامه‌نویسی مانند مترجم csc که برنامه‌های C# را به کدهای زبان ماشین ترجمه می‌کند.
- یک سری کتابخانه‌های استاندارد مثل ADO.NET که برای دسترسی به بانک های اطلاعاتی می‌باشد.

زبان برنامه‌نویسی C# یک زبان برنامه‌نویسی سطح بالا، همه منظوره و شیء گرا است که به‌وسیلهٔ 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.

۲ــ۲ــ شروع برنامه‌نویسی

همان‌طور که برای تهیه و پخت یک غذا، به مواد اولیه، لوازم آشپزی و دستور پخت[2] نیاز داریم برای تولید یک برنامه نیز، به یک کامپیوتر یا لپ تاپ، لوازم برنامه‌نویسی (یک ویرایشگر متنی[3] 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.

۱ــ Common Language Runtime

۲ــ Recipe

۳ــ Text Editor

در این صورت با طی مراحل زیر می‌توانیم برنامه‌ای را نوشته، ترجمه کرده و سپس اجرا نماییم.

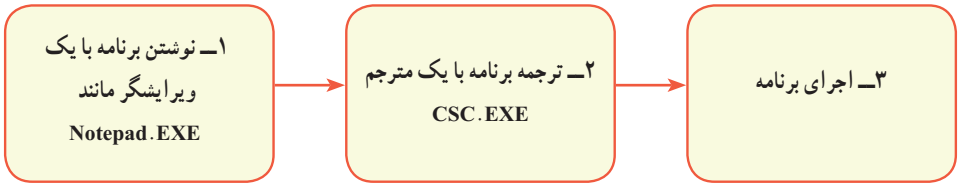
۱ــ نوشتن برنامه و ذخیره آن با استفاده از یک ویرایشگر مانند برنامه Notepad ویندوز

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

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

۳ــ اجرای برنامه ترجمه شده

نمودار ۱ــ۲ این مراحل را به ترتیب از چپ به راست نشان می‌دهد.



نمودار ۱ــ۲ــ مراحل برنامه‌نویسی، ترجمه و اجرا

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

۳ــ۲ــ اولین برنامه به زبان C#

با یک برنامه ساده به زبان C# آشنا می‌شویم. برنامه ۱ــ۲ را در زیرمشاهده کنید.

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

برنامه ۱ــ۲ــ اولین برنامه به زبان C#

این برنامه کوچک فقط یک پیام خوش‌آمدگویی بر روی صفحه نمایش نشان می‌دهد. رنگ‌های کلمات که در این برنامه مشاهده می‌کنید، تنها برای کمک به واضح شدن برنامه برای خواننده به کار

گرفته شده است و تأثیری بر روی برنامه ندارد. همان طور که در برنامه Notepad آنچه که می‌نویسید همگی با یک رنگ نوشته می‌شود.

سؤال: آیا با مشاهده این برنامه و بدون اطلاعات قبلی و یا کمک از دیگران، می‌توانید حدس بزنید که چه پیامی بر روی صفحه نشان داده می‌شود؟

برای این‌که با این برنامه آشنا شویم و یاد بگیریم که چگونه باید به زبان C# برنامه بنویسیم از دو جنبه این برنامه را بررسی خواهیم کرد :

الف) نگاه جزیی‌تر در حد کلمات و علامت‌ها

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

با **نگاهی** جزیی‌تر به برنامه ۱ــ۲، مشاهده می‌کنید که این برنامه از تعدادی کلمه و علامت تشکیل شده است. بعضی از کلمات مانند class،static و void کلمات شناخته شده برای زبان C# هستند و دارای معنی و مفهوم ثابتی هستند به این نوع کلمات ، کلمات کلیدی یا رزرو شده گفته می‌شود. کلمات رزرو شده[1] به رنگ آبی در این کتاب نوشته شده‌اند و به تدریج با آنها آشنا می‌شوید.

بعضی از کلمات دیگر مانند WelcomeToCSharp نامی است که به‌وسیله برنامه‌نویس و طبق سلیقه وی انتخاب می‌شود. به این نام‌ها شناسه[2] می‌گویند. برنامه نویس در انتخاب شناسه‌ها باید ضوابطی را رعایت کند که در فصل چهارم با آن آشنا می‌شوید.

علامت‌هایی مانند }، {،)، (و " نیز در این برنامه دیده می‌شود که معمولاً برای شروع یا پایان یک قسمت استفاده می‌گردد.

با **نگاهی دیگر و کلی‌تر** به برنامه ۱ــ۲، مشاهده می‌کنیم که یک برنامه ساده از یک قسمت کلی به نام کلاس تشکیل شده است که با کلمه کلیدی class مشخص می‌شود و شروع و پایان آن با علامت آکولاد باز و بسته تعیین می‌گردد. در جلوی کلمه کلیدی class، یک نام (شناسه) دلخواه مثلاً WelcomeToCSharp نوشته می‌شود که بیان کننده کار برنامه است. قسمت کلاس برنامه را در شکل زیر مشاهده کنید.

```
class WelcomeToCSharp
{

}
```

کلاس برنامه ۱ــ۲

۱ــ Reserved words ۲ــ Identifier

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 **متد[1] Main می‌گوییم که بدنه اجرایی برنامه است هر دستوری که در این قسمت نوشته شود به وسیلهٔ کامپیوتر به ترتیب اجرا می‌شود . دستورهای برنامه خود را در این قسمت می‌نویسیم.**

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

متد Main برنامه ۱ــ۲

آخرین قسمتی که در برنامه ۱ــ۲، در داخل متد 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 دستوربالا، پیام خوش‌آمدگویی Welcome to C#!بر روی صفحه نمایش، نشان داده می‌شود. آیا شما قبلاً درست حدس زده بودید که برنامه ۱ــ۲، چه پیامی‌را بر روی صفحه نمایش، نشان می‌دهد؟!

به وسیلهٔ دستور بالا، هر آنچه که داخل علامت‌های نقل قول " " قرار داشته باشد، بر روی صفحه نمایش نشان داده می‌شود حتی اگر به زبانی غیر از انگلیسی مثلاً فارسی نوشته شده باشد. توجه داشته باشید که خود علامت های 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.

۱ــ Method

نکته

توجه داشته باشید که زبان C# مانند زبان‌های C، 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.

۴ــ۲ــ الگوی یک برنامه ساده به زبان C#

یک برنامهٔ کاربردی نوشته شده به زبان C#، شامل مجموعه‌ای از کلاس‌ها است که هر یک از آنها نیز شامل تعدادی متد هستند. اما در یک برنامه ساده مانند برنامه ۱ــ۲، تنها یک کلاس وجود دارد که در آن نیز فقط یک متد به نام Main( ) تعریف می‌شود که نقطه آغاز اجرای برنامه است و الگوریتم خود را با رعایت قوانین زبان C# در آن می‌نویسیم. الگو یا ساختارکلی یک برنامه ساده به زبان C# در زیر آمده است. الگوی زیر را به‌خاطر بسپارید.

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

الگوی یک برنامه ساده به زبان C#

۵ــ۲ــ کلاس (class) چیست؟

کلاس یک مفهوم اساسی در برنامه‌نویسی شی گرا است که در کتاب برنامه سازی۲ 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 قبل تعریف شده و آماده وجود دارد که برنامه نویس کافی است آنها را بشناسد و در برنامه استفاده نماید. Console 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.

۱ــ۵ــ۲ــ نحوه تعریف کلاس : در زبان C# این امکان برای برنامه نویس فراهم است که کلاس جدیدی را تعریف کند. همان طور که در زیرمشاهده می کنید از کلمه کلیدی class برای تعریف و مشخص کردن یک کلاس جدید استفاده می‌شود. در جلوی کلمه class، یک نام دلخواه ذکر می گردد که نام کلاس است. مانند WelcomeToCSharp

```
class نام کلاس
{
        تعریف داده‌ها
        و عملیات بر روی آنها
}
```

الگوی یک کلاس

نکته

نام گذاری کلاس : نام یک کلاس به وسیلهٔ 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 نباید فاصله بگذارید ولی برای این که خواندنِ 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.

۶ــ۲ــ متد چیست؟

همان‌طور که گفته شد در داخل کلاس، عملیات بر روی داده‌ها و یا الگوریتم انجام یک کار تعریف می‌شود. متد مجموعه‌ای از دستورات است که برای انجام یک کار لازم است. هر متد 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 قرارگیرند، اما حتماً باید متدی 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 متد Main() ویژگی‌های متد 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 متد Main()

در این کتاب مانند برنامه ۱ــ۲، فقط به تعریف متد Main() می‌پردازیم و از متدهای آماده در زبان C# استفاده خواهیم کرد.

۱ــ۶ــ۲ــ استفاده از متدهای آماده : تعداد زیادی متد در کلاس‌های آماده زبان C# وجود دارد که هر یک از آنها، برای انجام کاری در نظر گرفته شده است. مثلاً متد WriteLine() از کلاس Console برای نشان دادن پیام روی صفحه نمایش در نظر گرفته شده است که در برنامه ۱ــ۲ 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 متد نیز باید نام فضا یا حوزه، نام کلاس و سپس نام متد را مشخص کنید و برای جدا کردن آنها از یکدیگر، علامت نقطه بین آنها قرار دهید (نمودار ۲ــ۲).

نمودار ۲ــ۲ــ طریقه استفاده از یک متد (System.Console.WriteLine)

به این ترتیب برای استفاده از متد ()WriteLine در برنامه ۱ــ۲ مشاهده می‌کنید که فضای نامی System[1] و نام کلاس Console و در آخر نام متد 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 ۲ــ۲ــ استفاده از توضیحات در متن برنامه

برنامه ۲ــ۲ مانند برنامه ۱ــ۲ است با این تفاوت که سه خط دیگر به متد ()Main اضافه شده است. خط دوم این برنامه فقط یک توضیح[2] برای خواننده برنامه می‌باشد و توضیح می‌دهد که خط بعدیِ 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.

۱ــ NameSpace

۲ــ Comment

خط سوم یک دستور اجرایی است :

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

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

با توجه به برنامه ۲ــ۲، پیامی را که می‌خواهید نمایش داده شود باید بین علامت های نقل قول " " قرار دهید. برای مثال اگر اسم شما Mohammad است و بخواهید روی صفحه نشان داده شود، باید به‌صورت زیر بنویسید :

```
System.Console.WriteLine ("Mohammad");
```

اگر بخواهید نام و نام خانوادگی خود را در دو سطر، نمایش دهید، در اینصورت می‌توانید دو بار از متد WriteLine() استفاده کنید. مثلاً برای نمایش "Mohammad " و " Ghasemi" چنین می‌نویسیم :

```
System.Console.WriteLine("Mohammad");
System.Console.WriteLine("Ghasemi");
```

حروف، علامت‌ها و عبارتی که ما بین علامت‌های نقل قول نوشته می‌شود را **رشته**[1] می‌نامند. این حروف می‌تواند فارسی، انگلیسی یا به هر زبانی باشد. مانند نام "Mohammad" و یا یک رمز عبور "Mehran2014". هر یک از حروف و علامت‌ها را نیز یک **کاراکتر**[2] می‌نامند. برای مثال Mohammad از ۸ کاراکتر و رمز عبور Mehran2014 از ۱۰ کاراکتر تشکیل شده است. چنانچه فاصله در رشته وجود داشته باشد، فاصله نیز یک کاراکتر محسوب می‌شود.

نکته

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

```
// Display a greeting message
/*
    FileName: welcome.cs ... Date : 05_07_2014
    Display a greeting message
*/
```

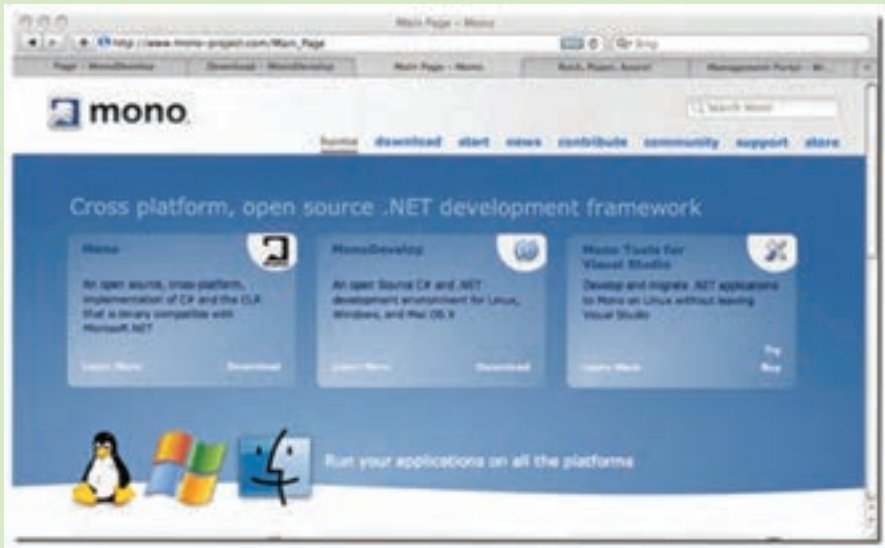
۱ــ String ۲ــ Character

برای مطالعه

زبان برنامه‌نویسی #C از لایه نرم‌افزاری .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 نرم‌افزاری 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

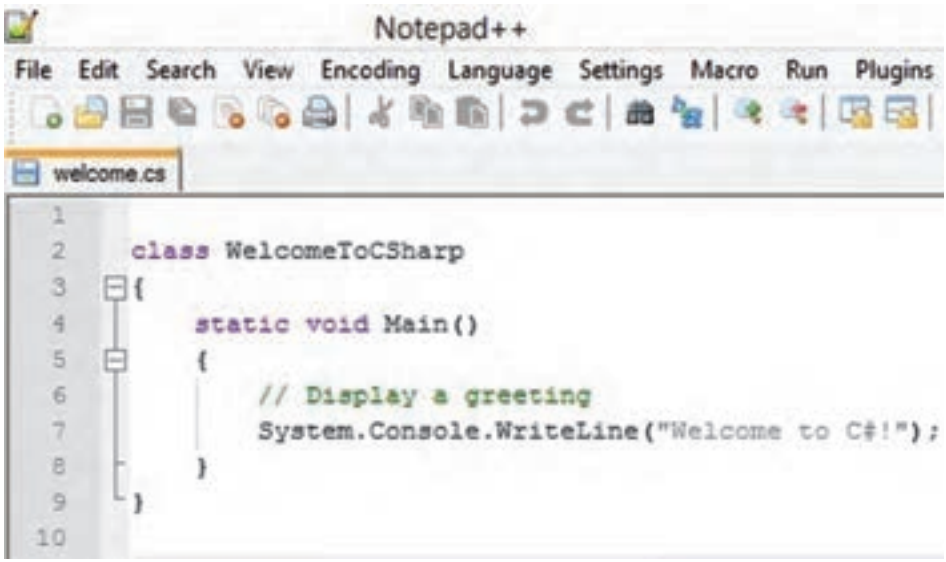


شکل ۲ــ۲ــ برنامه نویسی #C بر روی سیستم عامل غیر از ویندوز

کار در کارگاه: برنامه نویسی به زبان C#

قدم اول : نوشتن و تایپ برنامه : برای نوشتن یک برنامه ساده مانند برنامه ۱ــ۲، که در این فصل مورد بررسی قرار گرفت، نیاز به یک ویرایشگر متنی[۱] 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 به‌صورت رایگان از طریق سایت[۲] آن قابل دانلود است استفاده نماییم. این ویرایشگر به‌منظور برنامه‌نویسی به زبان‌های مختلف طراحی شده است به‌طوری که کلمات رزرو شده، رشته‌ها و توضیحات در این ویرایشگر با رنگ های مختلف نشان داده می‌شود (شکل ۳ــ۲). البته برای استفاده از این ویژگی باید ابتدا از منوی Language، زبان برنامه‌نویسی موردنظر خود را C# انتخاب کنید.



شکل ۳ــ۲ــ تصویری از محیط ویرایشگر Notepad++

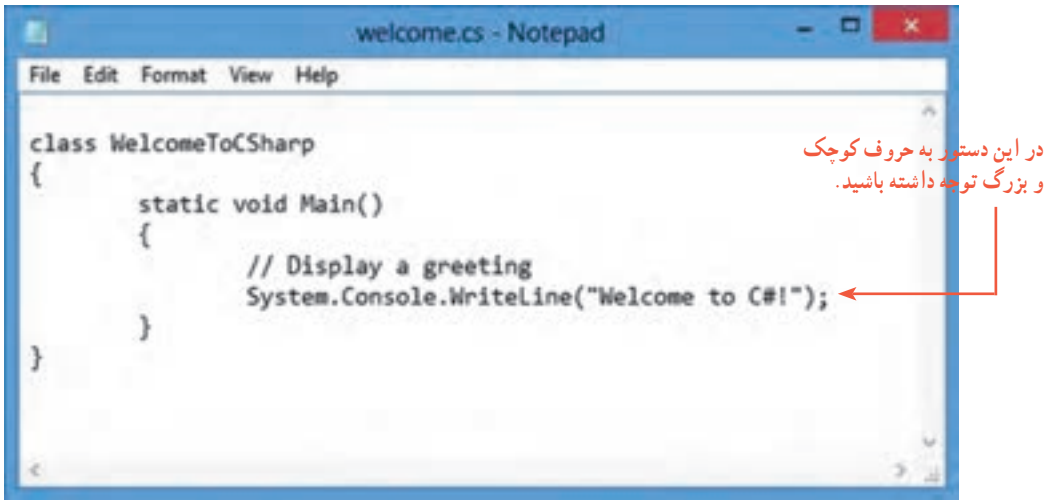
۱ــ Text Editor

۲ــ 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 اینکه 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 ۴ــ۲).

پس از آنکه برنامه ۱ــ۲ 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 قراردهید.

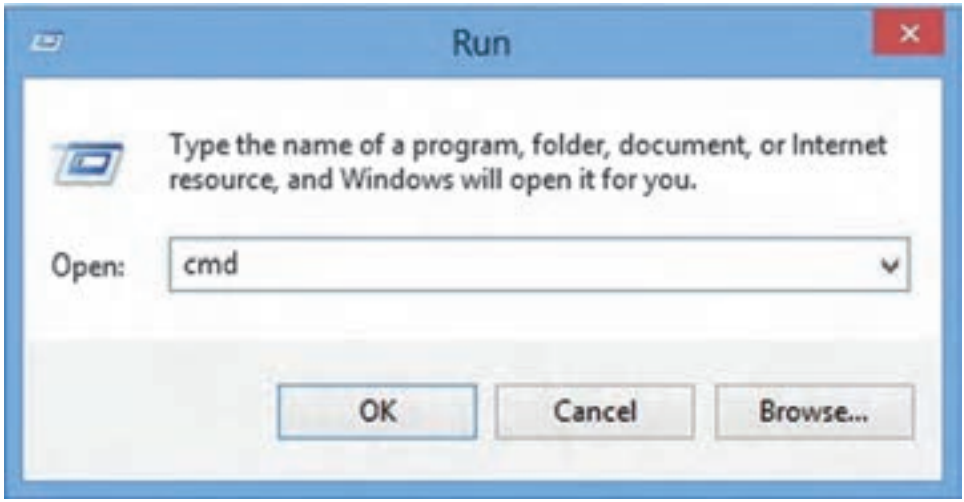


شکل۴ــ۲ــ تصویری از محیط ویرایشگر Notepad

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.

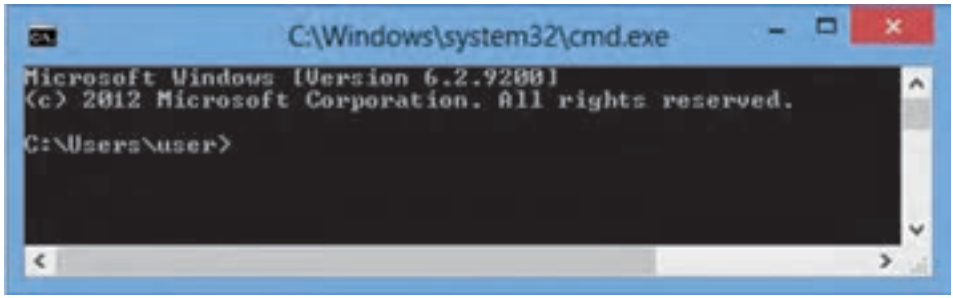
برای ترجمه برنامه مراحل زیر را دنبال کنید.

۱ــ از طریق گزینه Run فرمان cmd را اجرا کنید، تا وارد پنجره فرمان شویم (شکل ۵ــ۲).



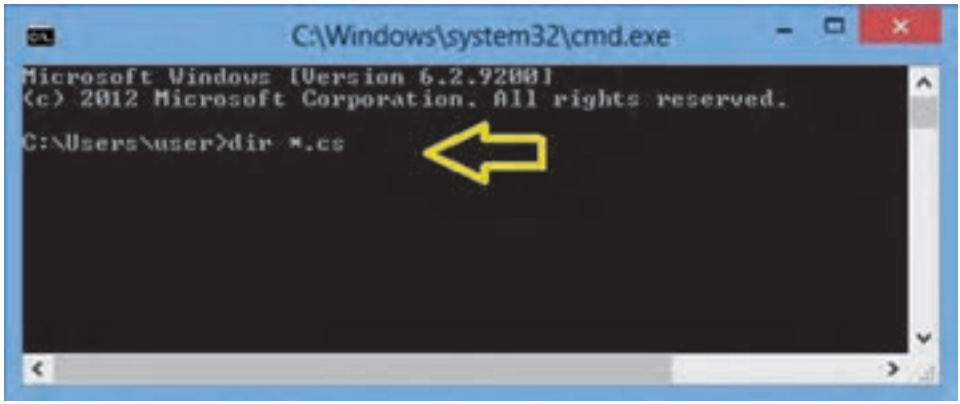
شکل ۵ــ۲ــ فرمان cmd

۲ــ پنجره فرمان ظاهر می‌شود (نوشته‌ها در شکل، ممکن است با آنچه در پنجره command prompt کامپیوتر شما دیده می‌شود متفاوت باشد) (شکل ۶ــ۲).



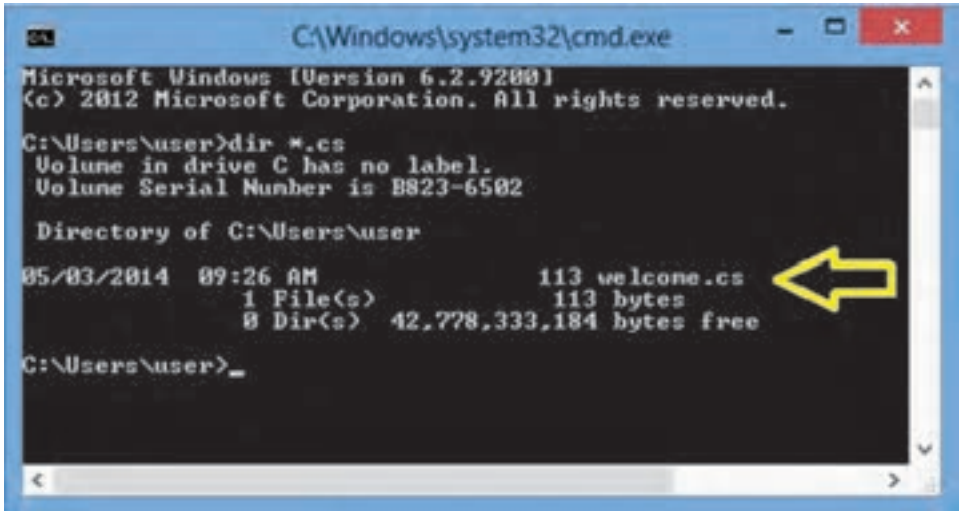
شکل ۶ــ۲ــ پنجره command prompt

۳ــ در پنجره Command Prompt از فرمان Dir استفاده می‌کنیم و با توجه به این‌که پسوند فایل cs می‌باشد، با تایپ فرمان زیر از وجود فایل برنامه مطمئن می‌شویم (شکل ۷ــ۲). 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 cd.. نیز برای وقتی که می‌خواهید به یک پوشه بالاتر بروید مفید باشد.



شکل ۷ــ۲ــ فرمان Dir

۴ــ اگر مرحله قبل را به درستی انجام دهید، باید شکل ۸ــ۲ را مشاهده کنید.

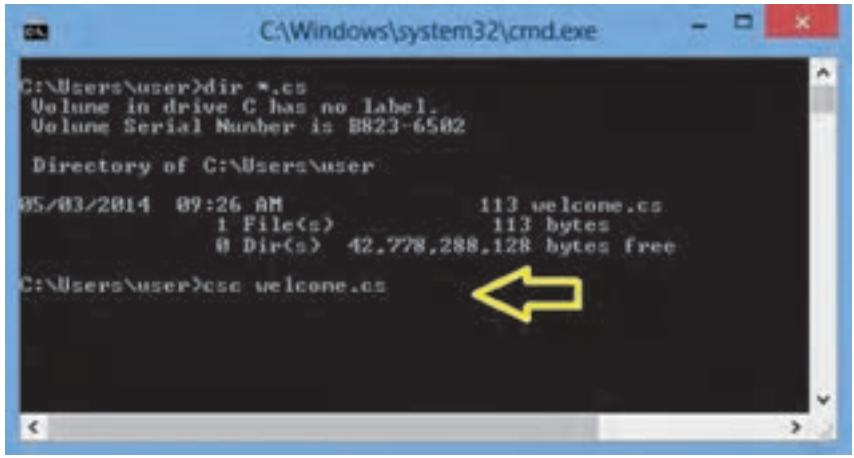


شکل ۸ــ۲ــ مشخصات فایل برنامه

۵ــ توجه داشته باشید اگر برنامه Net Framework. بر روی کامپیوتر شما قبلاً نصب شده باشد برنامه‌ای به نام csc.exe[1] برای ترجمه برنامه‌های زبان C# در اختیار دارید. پس از یافتن فایل خود، با استفاده از این مترجم، برنامه خود را ترجمه نمایید. در پنجره فرمان از دستور زیر استفاده کنید :

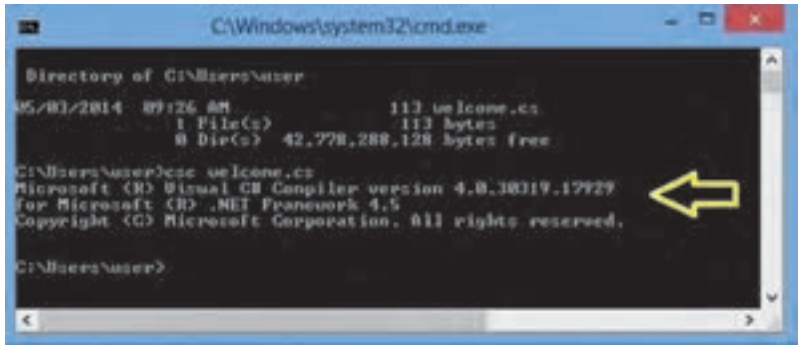
```
csc filename.cs
```

منظور از filename.cs نام فایل مورد نظر شما است.



شکل ۹ــ۲ــ فرمان ترجمه برنامه

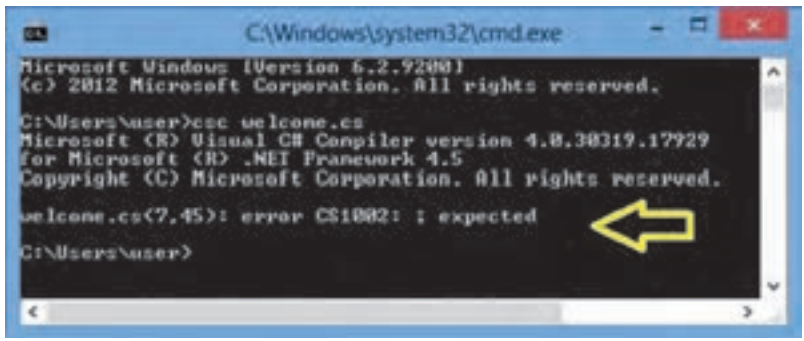
۶ــ با اجرای دستور بالا مترجم شروع به ترجمه برنامه می‌کند و اگر همه کارها به درستی انجام شده باشد (برنامه NET. قبلاً نصب شده باشد، تایپ دستور csc را درست انجام داده باشید، فایل csc به درستی در مسیر جستجوی سیستم عامل (path) معرفی شده باشد و برنامه هیچ خطایی نداشته باشد.) شکل ۱۰ــ۲ را مشاهده خواهید کرد.



شکل ۱۰ــ۲ــ نتیجه ترجمه برنامه

۱ــ C Sharp Compiler

۷ــ اگر در تایپ برنامه اشتباهی انجام داده باشید، مترجم نمی‌تواند برنامه را ترجمه کند و در این‌صورت خطا یا خطاهای برنامه 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 ۷ــ۲ ظاهر می‌شود.



شکل ۱۱ــ۲ــ پیام خطای مترجم برای ننوشتن 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 خطا را با دقت بخوانید و معنی آن را درک کنید تا بتوانید اشکال برنامه را برطرف کنید. در این مثال توضیح خطا چنین است : علامت ; فراموش شده

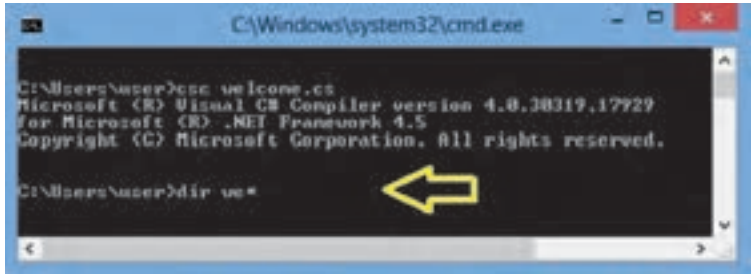
برای برطرف کردن خطای بالا لازم است برنامه را به وسیلهٔ ویرایشگر باز کنید و به سطر ۷ و ستون ۴۵ بروید و علامت نقطه ویرگول را اضافه کنید و سپس برنامه را تحت همان نام قبلی ذخیره کنید و دوباره به پنجره فرمان بازگردید و عمل ترجمه را انجام دهید. اگر خطایی رخ نداد به مرحله بعدی بروید وگرنه باید دوباره عملیات 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 :

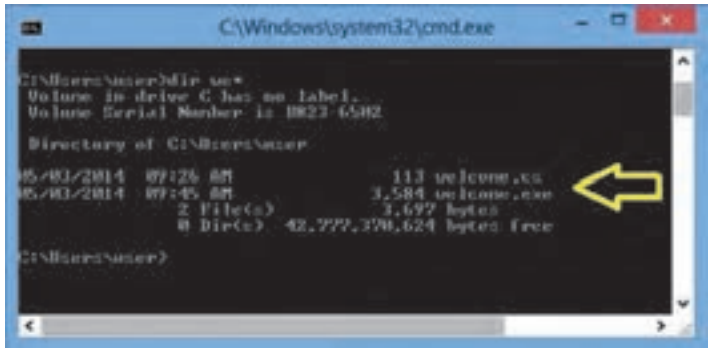
error CS1002: ; expected

۸ــ در صورت ترجمه موفق برنامه، فایل جدیدی ساخته می‌شود. با دستور Dir از وجود آن مطمئن می‌شویم (شکل ۱۲ــ۲).



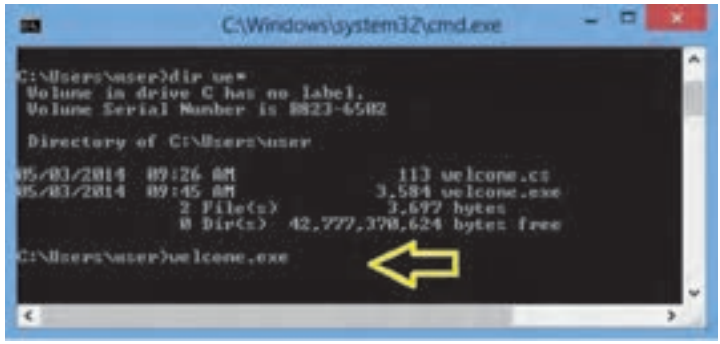
شکل ۱۲ــ۲ــ جستجوی فایل ترجمه شده

۹ــ همان طور که در شکل ۱۳ــ۲ مشاهده می‌کنید، فایل جدیدی با همان نام welcome ولی با پسوند EXE ساخته شده است.



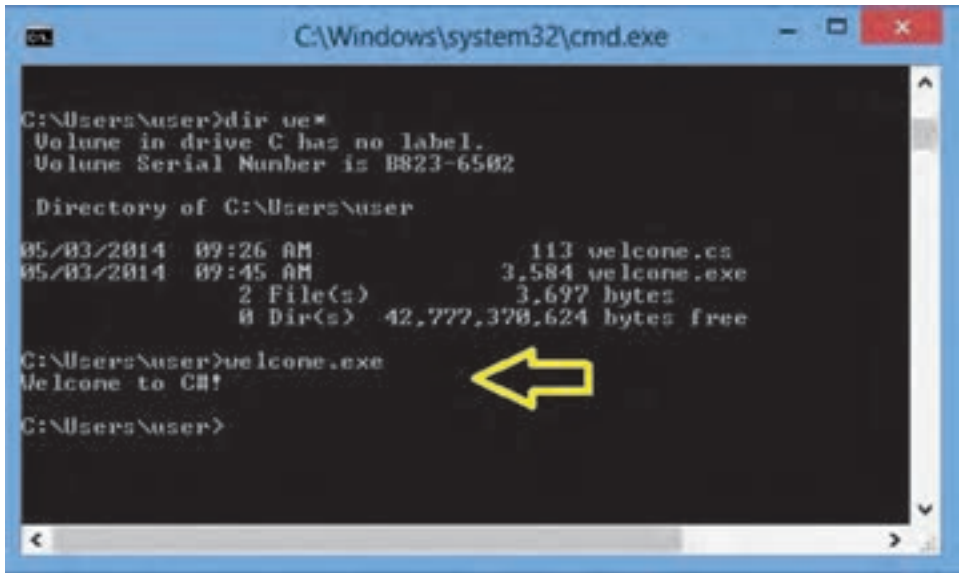
شکل ۱۳ــ۲ــ لیست فایل های برنامه و ترجمه شده

۱۰ــ 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 ۱۴ــ۲).



شکل۱۴ــ۲ــ اجرای فایل ترجمه شده یا برنامه اجرایی

۱۱ــ نتیجه اجرای برنامه به‌صورت شکل ۱۵ــ۲ خواهد بود.

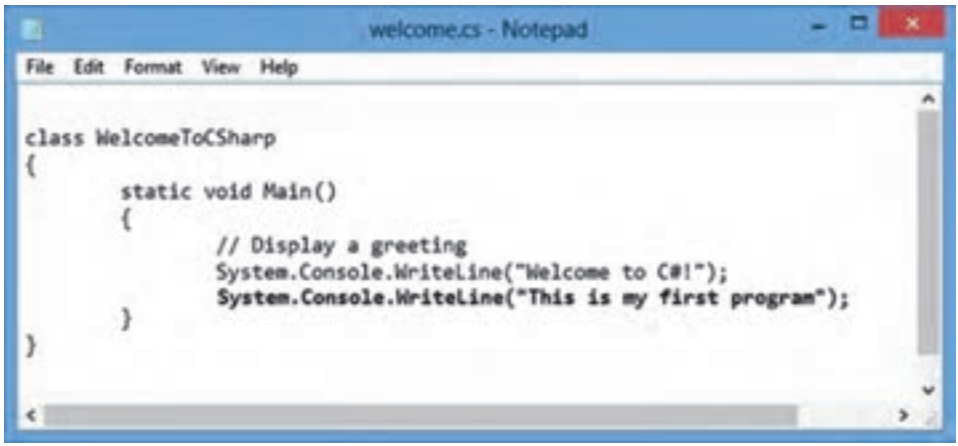


شکل ۱۵ــ۲ــ نتیجه اجرای برنامه

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

۱۲ــ برنامه را با ویرایشگر باز کنید و یک دستور مانند شکل ۱۶ــ۲ به آن اضافه کنید. (تغییرات با رنگ تیره مشخص شده است.)

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



شکل ۱۶ــ۲ــ اضافه کردن یک دستور به برنامه

۱۳ــ دوباره برنامه را با ویرایشگر باز کنید و در اولین دستور به جای متد ()WriteLine از متد ()Write مانند شکل ۱۷ــ۲ استفاده کنید.

شکل۱۷ــ۲ــ استفاده از متد ()Write

چه تفاوتی در اجرای برنامه حاصل می‌شود؟ پیام‌ها چگونه نشان داده شدند؟

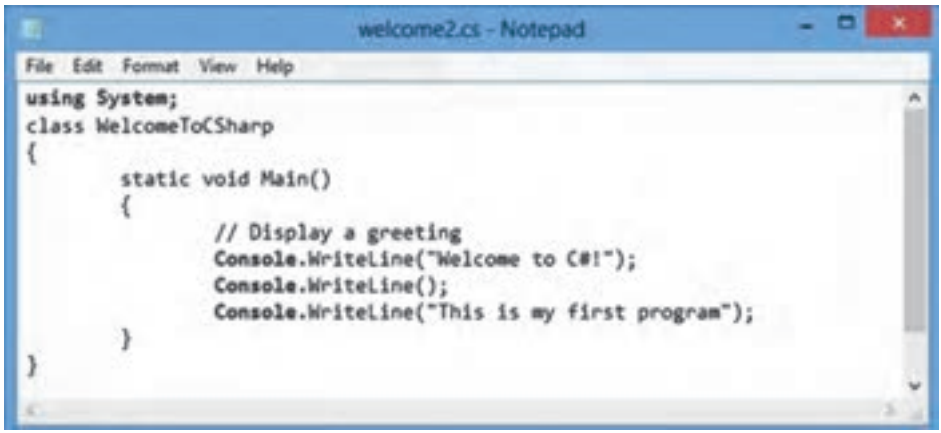
۱۴ــ برنامه را به حالت اولیه خود بازگردانید و یک دستور نیز به‌صورت شکل ۱۸ــ۲ به آن اضافه کنید. برنامه را ذخیره کنید و سپس ترجمه و اجرا نمایید. چه تغییری در خروجی ایجاد می‌شود؟

شکل ۱۸ــ۲ــ استفاده از متد ()WriteLine برای ایجاد یک خط خالی

۱۵ــ همان‌طور که در شکل ۱۸ــ۲ مشاهده می‌کنید، برای هر بار استفاده از متد WriteLine() مجبور هستیم کلمات System و Console را ذکر کنیم. برای کوتاه کردن این دستورات می‌توانیم در ابتدای برنامه، فضای نامی System را معرفی کنیم که در آن کلاس Console تعریف شده است. در این‌صورت می‌توانیم در داخل برنامه، کلمه System را از ابتدای دستورات حذف کنیم. برای معرفی فضای نامی از دستور using به‌صورت زیر استفاده می‌کنیم :

```
using فضای نامی ;
```

برنامه قبلی را باز کرده و تغییرات زیر را اعمال کنید و آن را اجرا نمایید (شکل ۱۹ــ۲).

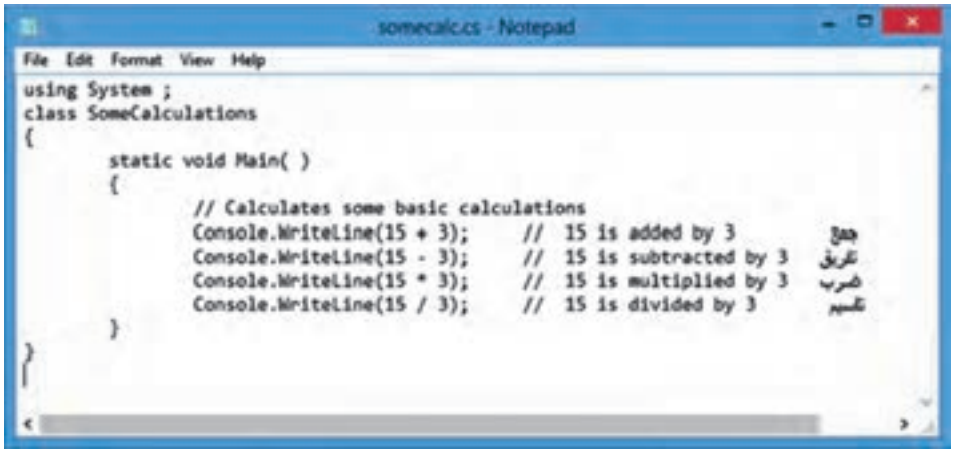


شکل ۱۹ــ۲ــ استفاده از فضای نام

۱۶ــ از متد WriteLine() علاوه بر نمایش یک پیام می‌توان نتیجه یک عبارت ریاضی را نیز نمایش داد. برنامه‌ای برای انجام چهار عمل اصلی بر روی دو عدد نوشته شده است (شکل ۲۰ــ۲). 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 این‌که سریع‌تر بتوانید این برنامه را تایپ کنید، یکی از برنامه‌های قبلی خود را با ویرایشگر باز کنید و فقط نام کلاس و دستورات داخل متد Main را تغییر دهید و سپس تحت نام جدیدی آن را ذخیره (Save As...) کنید. توضیحات برنامه می‌تواند به زبان فارسی نیز باشد.



شکل ۲۰ــ۲ــ برنامهٔ چهار عمل اصلی

۱۷ــ برنامه شکل ۲۰ــ۲ 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 نشان داده می‌شوند را یادداشت کنید، آیا محاسبات درست انجام شده است؟

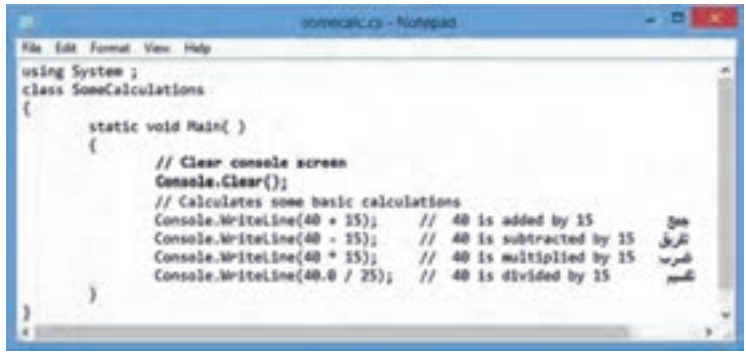
۱۸ــ اگر به خروجیِ دستور زیر توجه کنید

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

مشاهده خواهید کرد که عدد ۱ چاپ می‌شود چون تقسیم صحیح و بدون اعشار انجام می‌شود. در صورتی‌که جواب صحیح و دقیق عدد ۱/۶ 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 ۴۰ را به صورت ۴۰/۰ یا عدد ۲۵ را به صورت ۲۵/۰ بنویسید. حال برنامه را در ویرایشگر باز کرده و یکی از اعداد در دستور تقسیم را به‌صورت اعشاری بنویسید و سپس برنامه را با همان نام قبلی ذخیره و ترجمه و اجرا نمایید.

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

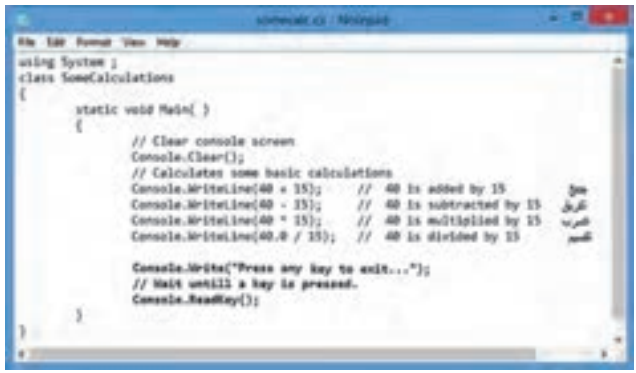
۱۹ــ برای این که پیام‌ها بهتر دیده شوند، می‌توانیم ابتدا با استفاده از متد Clear() صفحه نمایش را پاک کنیم. به این منظور برنامه را با ویرایشگر باز کنید و متد مذکور را به برنامه طبق شکل ۲۱ــ۲ اضافه کنید. برنامه را تحت نام فعلی ذخیره کنید و سپس ترجمه و اجرا نمایید.



شکل ۲۱ــ۲ــ استفاده از متد ()Clear برای پاک کردن صفحه

۲۰ــ پنجره فرمان را ببندید وارد محیط میزکار[1] شوید و با استفاده از Computer وارد پوشه‌ای 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؟

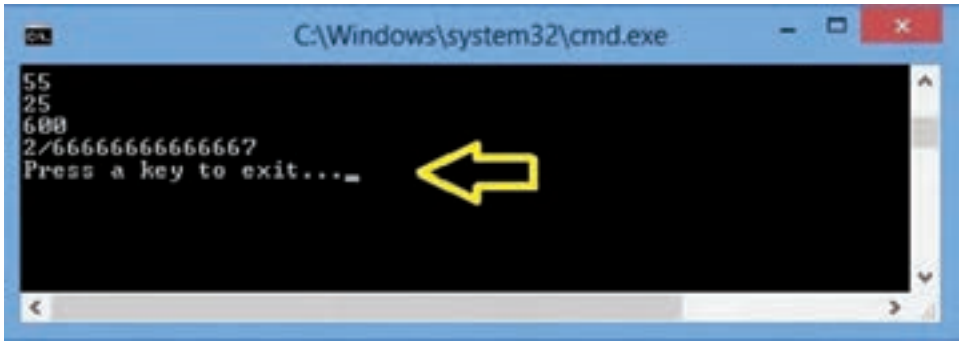
در یک لحظه صفحه کنسول باز شده و به سرعت بسته می‌شود و شما فرصت نمی‌کنید نتایج برنامه را مشاهده کنید. لازم است در انتهای برنامه، دستوری بنویسید که کامپیوتر را به‌صورت موقتی متوقف کند تا شما بتوانید نتایج محاسبات را ببینید. به این منظور از متد ()ReadKey استفاده می‌کنیم. با اجرای این متد، کامپیوتر منتظر 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 ۲۲ــ۲ 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.



شکل ۲۲ــ۲ــ اضافه کردن متد ()ReadKey به برنامه شکل ۲۱ــ۲

۱ــ Desktop

با اجرای برنامه، مشاهده خواهید کرد که نتایج محاسبات و پیام ...Press any key to exit روی صفحه نشان داده می‌شود و سپس کامپیوتر متوقف می‌شود و با زدن یک کلید برنامه پایان می‌یابد (شکل ۲۳ــ۲).



شکل ۲۳ــ۲ــ استفاده از متد ReadKey() برای دریافت یک کلید و توقف موقتی برنامه

کار با رنگ‌ها : ConsoleColor جعبهٔ رنگ ۱۶تایی در C# است. نام این رنگ‌ها در جدول ۱ــ۲ آمده است.

۲۱ــ ویژگی 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 متد Clear کلاس Console استفاده کنیم. دستور زیر را در خط بعدی بنویسید.

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

۲۲ــ ویژگی 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 ۱ــ۲ ــ جعبهٔ رنگ ConsoleColor

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

خودآزمایی فصل دوم

۱ــ قالب کلی یک برنامه به زبان C# چگونه است؟

۲ــ درهر برنامه به زبان C#، دست‌کم یک ..... تعریف می‌شود که در داخل آن، یک متد به نام ..... وجود دارد که اجرای برنامه از آن نقطه آغاز می‌گردد.

۳ــ در زبان C#، انتهای یک دستور با چه علامتی مشخص می‌شود؟

۴ــ تحقیق کنید که در زبان‌های C، C++ و Java علامت پایان دستور چیست؟

۵ــ علامت توضیحات در زبان C# کدام است؟

۶ــ روش قراردادی پاسکال برای نوشتن نام یک کلاس چیست؟

۷ــ برای نشان دادن یک پیام یا یک عبارت بر روی صفحه نمایش از چه دستوری استفاده می‌کنید؟

۸ــ تفاوت بین دو متد ()WriteLine و ()Write را بیان کنید.

۹ــ با استفاده از متد ()WriteLine، دستور مناسبی برای ایجاد یک سطر خالی بین نوشته‌ها در خروجی بنویسید.

۱۰ــ مثالی از کاربرد دستور using بیاورید.

۱۱ــ برای هر یک از خواسته‌های 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 برنامه‌نویسی فصل دوم

۱ــ برنامه‌ای بنویسید که نام، نام خانوادگی و نام مدرسه شما را روی صفحه نمایش دهد.

۲ــ 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 |
- - - - - - - - - - -
```

۳ــ 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 * به‌صورت زیر نشان دهد.

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

یادآوری : هنگامی‌که 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.

۴ــ 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 ۱۶ × ۳۶۵ را محاسبه و نمایش دهد.

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

۵ــ با توجه به تمرین ۴ تعداد سال‌های کبیسه را نیز حساب کنید و برنامه را تغییر دهید.

(**راهنمایی :** برای محاسبه تعداد سال کبیسه، خارج قسمت سن بر عدد ۴ را حساب کنید.)

۶ــ با اجرای برنامه زیر چه عبارتی بر روی صفحه نشان داده می‌شود؟ به نظر شما چه ارتباطی بین علایم {۰} و {۱} و {۲} با اعداد ۱۸ و۱۵ و ۱۵+۱۸ وجود دارد؟

```
class SomeCalculations
{
static void Main( )
   {
   System.Console.WriteLine("{0} + {1} = {2}" , 18 , 15 , 18+15);
   }
}
```

۷ــ 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 دوم

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	

۳ 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[۱] 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 نوشتن و تولید برنامه‌های کوچک، خوب است اما اگر برنامه‌ای که می‌نویسیم بزرگ باشد، به کارگیری این روش کمی‌دشوار و زمان‌بر خواهد بود. به خصوص عیب‌یابی و اشکال‌زدایی آن بسیار وقت‌گیر 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 فراگیر بتواند:

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

۳ــ۱ــ آشنایی با ویژوال استودیو[۲]

تایپ برنامه در یک ویرایشگر، ورود به پنجره فرمان، ترجمه کردن، عیب یابی و اشکال زدایی برنامه، همگی عملیاتی بودند که در فصل قبلی انجام دادید وکمی‌وقت گیر و پر زحمت بود، چون از یک محیط باید وارد محیط دیگری می‌شدید. برای این که راحت‌تر بتوانیم برنامه‌نویسی کنیم لازم است از محیطی استفاده کنیم که همه ابزارها و لوازم مورد نیاز برنامه‌نویسی در آن گردآوری و متمرکز شده

۱ــ Command Prompt
۲ــ Visual Studio

باشد. به چنین محیط برنامه‌نویسی که در آن می‌توان تمام مراحل برنامه‌نویسی، ترجمه، اشکال یابی و سرانجام اجرا را انجام داد، IDE[1] گفته می‌شود که به معنای محیط تولید برنامهٔ 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 می‌توانیم 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 شده است. در حال حاضر[2] VS 2013 عرضه شده است. خوشبختانه شرکت مایکروسافت همراه با عرضه ویژوال استودیوی تجاری، یک نسخه رایگان از این نرم افزار را تحت عنوان ویژوال استودیو اکسپرس[3] نیز عرضه می‌کند که می‌توانید آن را از روی سایت شرکت مایکروسافت[4] دانلود نمایید.

شکل ۱ــ۳ــ اجرای ویژوال استودیو اکسپرس ۲۰۱۲

---

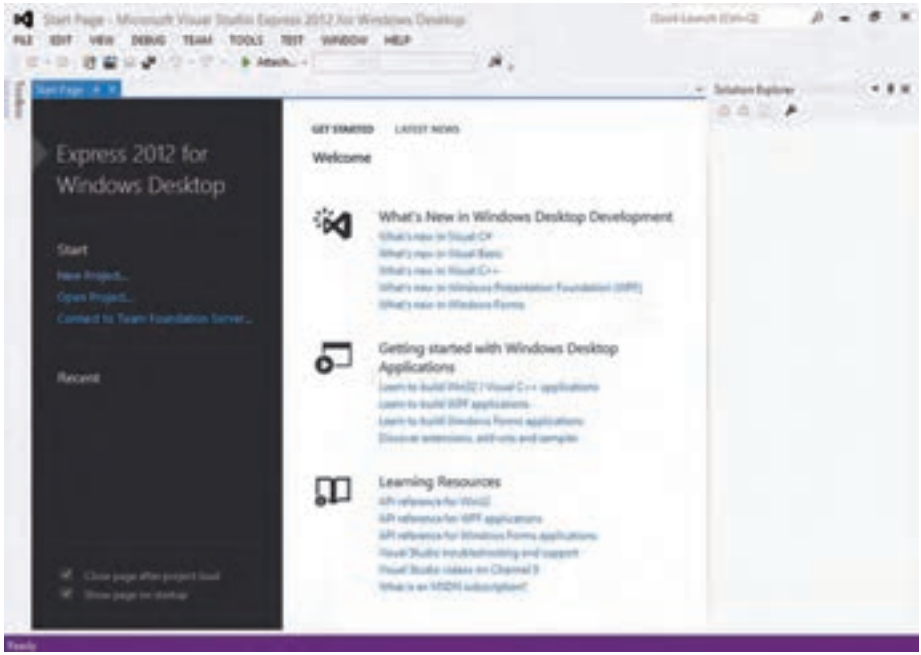
۱ ــ Integrated Development Environment

۲ ــ اردیبهشت ۹۳

۳ ــ Visual Studio Express

۴ ــ http: //microsoft.com/visualstudio/downloads

ما در این کتاب با Visual Studio Express 2012 کار می‌کنیم و از این به بعد در سرتاسر کتاب از مخفف VS برای بیان کلمه ویژوال استودیو استفاده می‌کنیم. اگر نرم افزار VS 2012 را در اختیار ندارید می‌توانید از نسخه‌های قدیمی‌تر نیز استفاده کنید فقط شکل ظاهری آن ممکن است کمی‌متفاوت باشد.

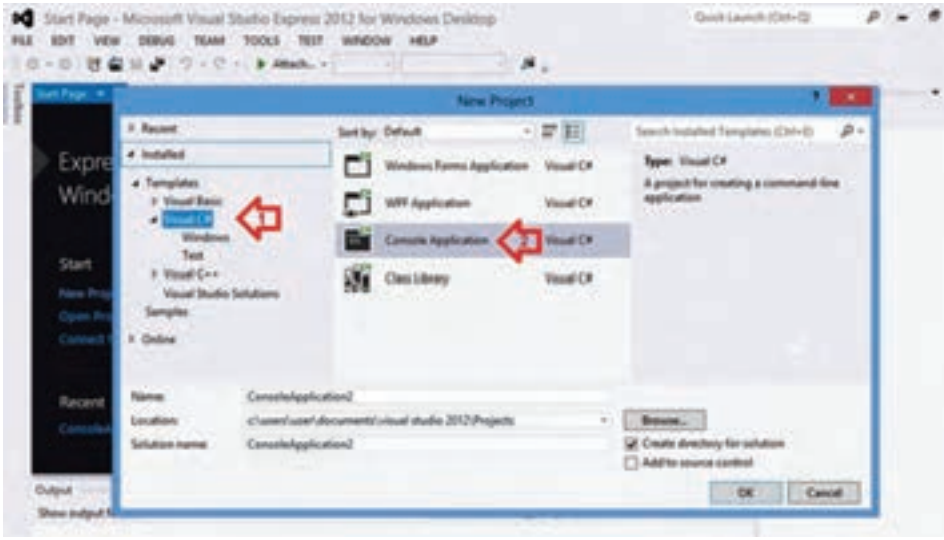


شکل ۲ــ۳ــ پنجره آغازین ویژوال استودیو اکسپرس ۲۰۱۲

۲ــ۳ــ ایجاد یک پروژه جدید در ویژوال استودیو

با اجرای برنامهٔ VS، صفحهٔ شروع (Start Page) مطابق با شکل ۳ــ۳ ظاهر می‌شود، از این صفحه می‌توانید یک پروژه یا برنامه جدید (New Project...) بسازید و یا برنامه‌های قبلی خود را باز(Open Project...) کنید.

برای ایجاد یک برنامهٔ جدید به زبان C# که در محیط کنسول کار می‌کند ابتدا روی گزینه (New Project...) در لیست سمت چپ کلیک کنید. پس از ظاهر شدن پنجره، Visual C# را انتخاب کنید (شکل۳ــ۳ــ مرحله ۱) و در وسط صفحه بر روی گزینهٔ Console Application کلیک کنید (شکل ۳ــ۳ مرحله۲).



شکل ۳ــ۳ــ 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 برنامه‌ای که می‌نویسید انتخاب کنید چون این نام به هیچ وجه گویا و روشن نیست.

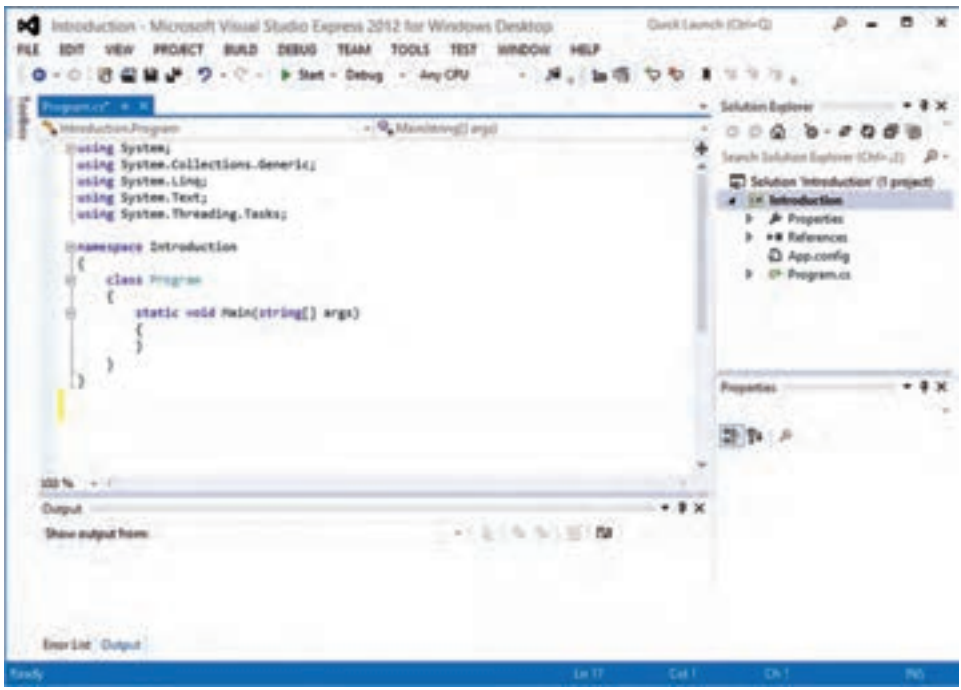
همان‌طور که درشکل ۴ــ۳ مشاهده می‌کنید، در قسمت Location مسیر ذخیره پروژه نشان داده شده است. هنگامی که VS را نصب می‌کنید در داخل My Documents یک پوشه به نام Visual Studio ساخته می‌شود که در داخل آن نیز یک پوشه فرعی به نام Projects ایجاد می‌شود. در داخل این پوشه، پروژه‌های شما به‌طور پیش فرض ذخیره می‌شود. اگر بخواهید می‌توانید پروژه خود را در مسیر دیگری ذخیره کنید. مسیر دلخواه خود را با کلیک بر روی دکمه Browse مشخص کنید.

شکل ۴ــ۳ــ تعیین نام برای یک پروژه جدید

بعد از مشخص کردن محل پروژه، باید نامی‌برای Solution name در نظر بگیرید که نام پوشه‌ای را معین می‌کند که فایل‌های مربوط به یک یا 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 می شود. حال پس از تعیین نام‌ها، بر روی کلید Ok 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 نامی‌که 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 ۵ــ۳ نشان داده می‌شود.



شکل ۵ــ۳ــ محیط تولید برنامه (IDE) وِیژوال استودیو

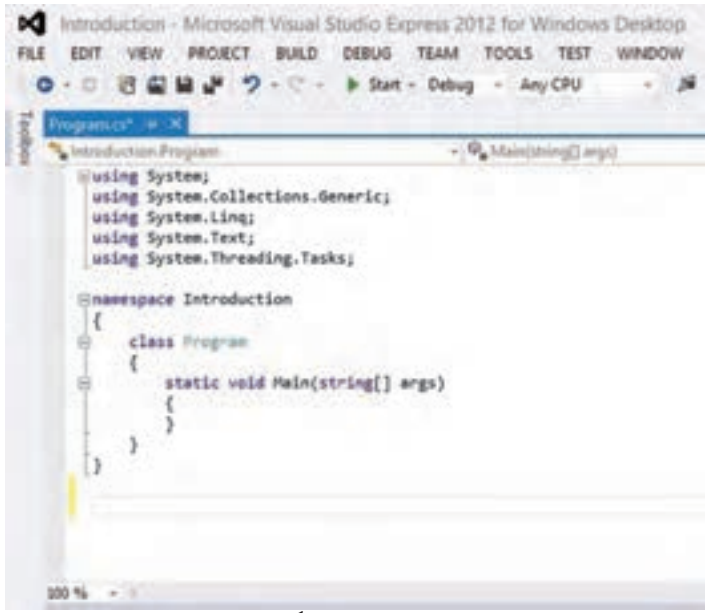
قبل از این که به برنامه‌نویسی بپردازیم ابتدا به صورت مختصر با بخش‌های مختلف محیط IDE آشنا می‌شویم.

۳ــ۳ــ معرفی بخش‌های اصلی ویژوال استودیو

۱ــ۳ــ۳ــ نوار منو و نوار ابزار[۱] : مانند بیشتر نرم‌افزارها، در قسمت بالای صفحه، منوهای مختلف VS و در زیر آن ابزارهای پرکاربرد مطابق با شکل ۶ــ۳ دیده می‌شود. به تدریج با این منوها وابزارها آشنا می‌شوید.

شکل ۶ــ۳ــ نوار منو و نوار ابزار محیط ویژوال استودیو

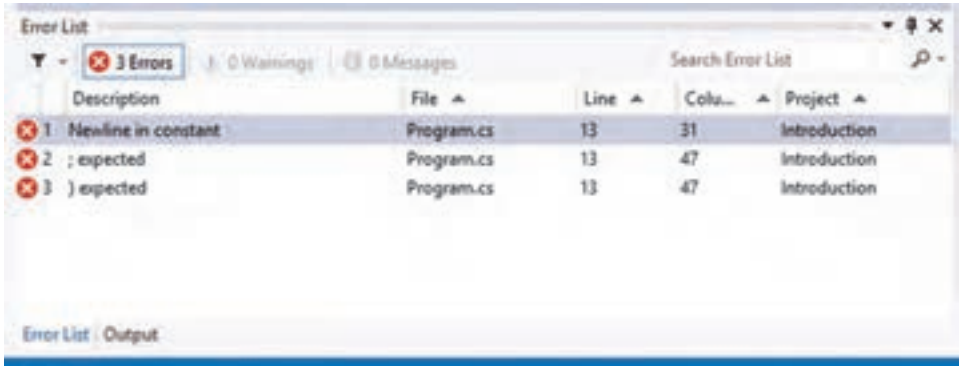
۲ــ۳ــ۳ــ پنجره ویرایشگر برنامه : در شکل ۷ــ۳ پنجره ویرایشگر برنامه نشان داده شده است. در این پنجره متن برنامه نگهداری می‌شود و می‌توانید چندین برنامه را هم زمان به صورت باز داشته باشید. در برنامه‌نویسی، این پنجره کاربرد زیاد دارد.



شکل ۷ــ۳ــ پنجره ویرایشگر برنامه

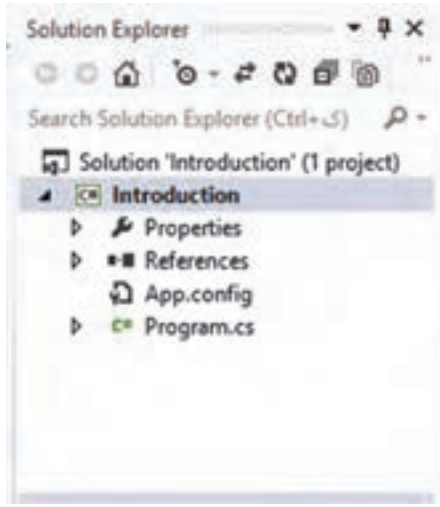
۱ــTool Bar

۳ــ۳ــ۳ــ پنجره لیست خطاها (Error List) : در صورتی که برنامه اشکال تایپی یا ساختاری داشته باشد، خطاها و اشکالات برنامه در این پنجره مانند شکل ۸ ــ۳ لیست می‌شوند. پس از ترجمهٔ برنامه باید به این پنجره نگاه کنیم تا خطاهای احتمالی برنامه را متوجه شویم.



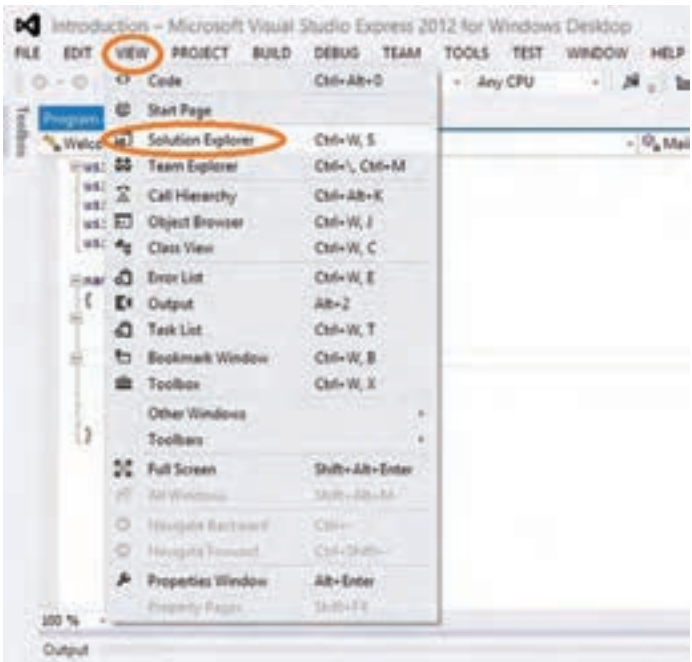
شکل ۸ ــ۳ــ پنجره لیست خطاها

۴ــ۳ــ۳ــ پنجره (Solution Explorer) : سمت راست صفحه، پنجره‌ای 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 وسیلهٔ آن به تمام اجزای پروژه دسترسی داریم.



شکل ۹ــ۳ــ پنجره Solution Explorer

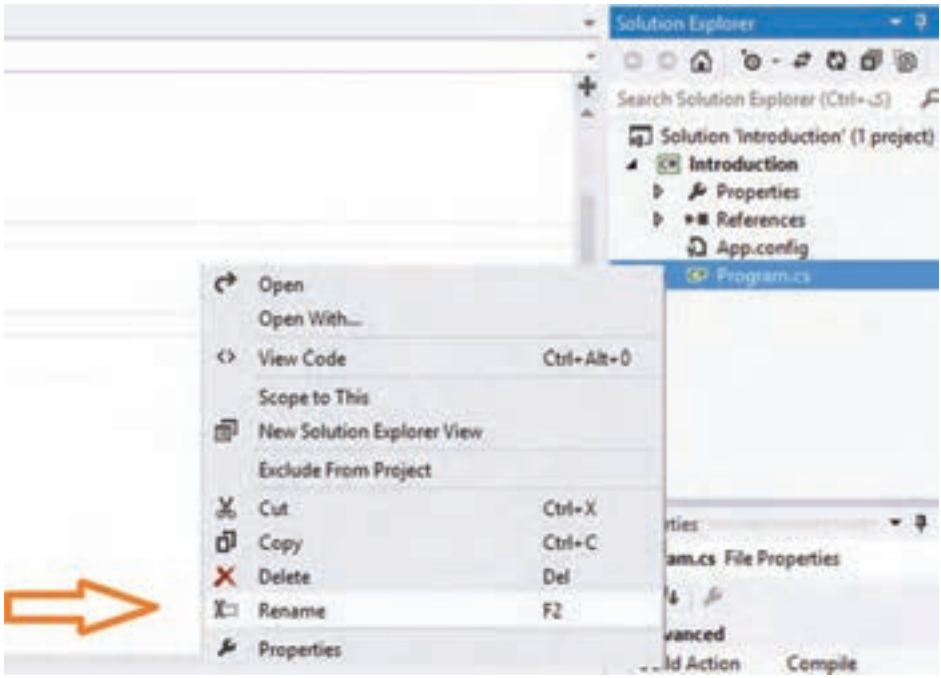
اگر پنجرهٔ مرورگر پروژه باز نیست، و آن را مشاهده نمی‌کنید از منوی View در بالای صفحه استفاده کنید. در این منو، نام تمام پنجره‌ها در شکل ۱۰ــ۳ مشاهده می‌شود، روی گزینه Solution Explorer کلیک کنید تا این پنجره دیده شود.



شکل ۱۰ــ۳ــ ظاهر کردن پنجره Solution Explorer

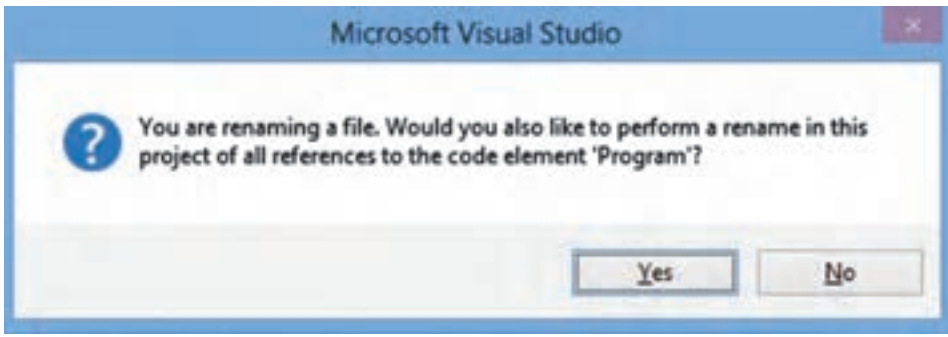
۴ــ۳ــ برنامه‌نویسی در محیط ویژوال استودیو

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 به‌طور 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 پی‌ببرید. در این مثال می‌خواهیم برنامه‌ای بنویسیم که یک پیام خوش‌آمدگویی 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 ۱۱ــ۳).



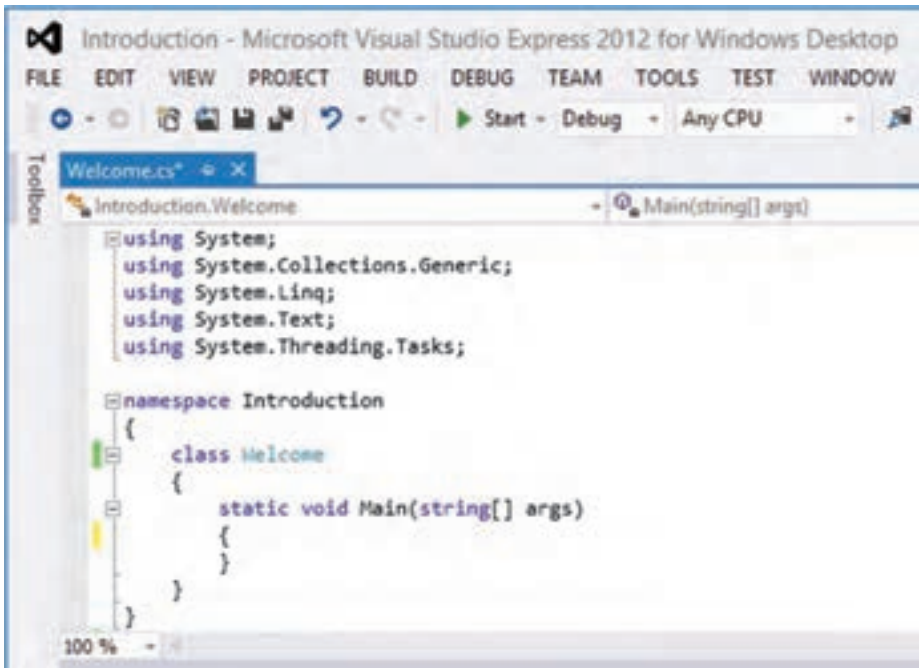
شکل ۱۱ــ۳ــ تغییر نام برنامه

پس از تغییر نام و زدن کلید Enter پنجره‌ای مانند شکل ۱۲ــ۳ 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 ۱۲ــ۳ــ تغییر نام در تمام قسمت‌هایی که به **Program** رجوع می‌کند

حال به پنجره کد، باز می‌گردیم، همان‌طور که در شکل ۱۳ــ۳ ملاحظه می‌کنید ساختار کلی یک برنامه به وسیلهٔ VS برای شما آماده می‌شود که تعدادی دستور در آن موجود است.



شکل ۱۳ــ۳ــ 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 فعلاً 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 نامی‌جدید تعریف می‌شود که برای سازماندهی و دسته‌بندی پروژه‌های بزرگ مورد استفاده دارد. فعلاً به آن کار نداریم ولی لازم نیست آن را پاک کنید، بنابراین تغییری روی آن انجام ندهید.

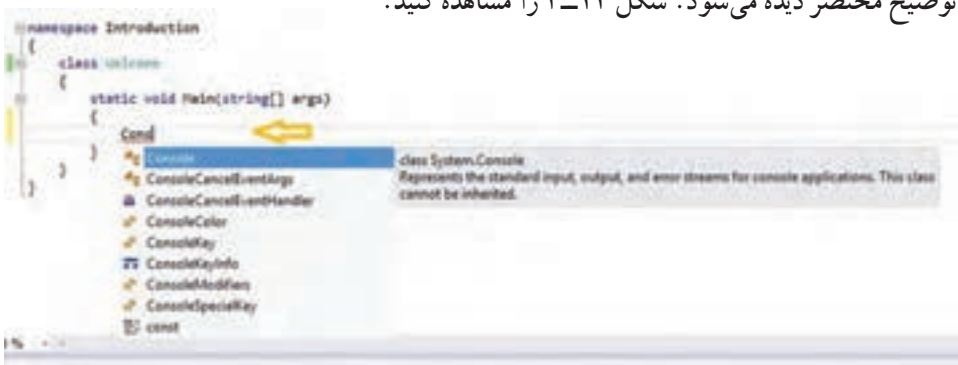
در داخل این فضای نامی، دستور class و سپس در داخل آن متد Main نوشته شده است. نام کلاس Welcome است مگر این که در مرحله قبل نام برنامه را تغییر نداده باشید که در این صورت

نام کلاس Program است. متد ()Main نیز در داخل کلاس دیده می‌شود ولی داخل پرانتزهای آن عبارت string[] args نوشته شده است. می‌توانید 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 آکولادها 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 :

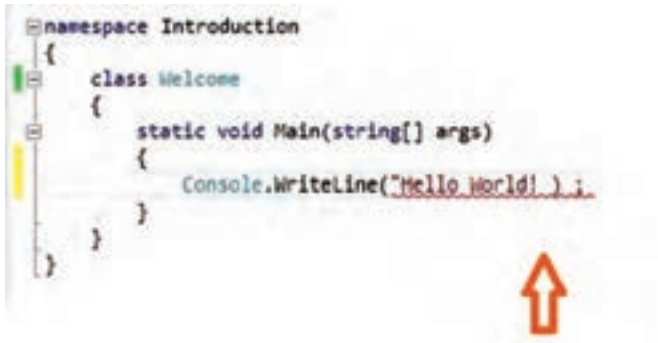
```
Console.WriteLine("Hello World");
Console.WriteLine("Welcome to C#");
```

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 می‌نوشتید 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 ۱۴ــ۳ را مشاهده کنید.



شکل ۱۴ــ۳ــ کمک 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 رها کنید و کلید Enter را بزنید. در این صورت کلمه به‌طور کامل تایپ می‌شود و می‌توانید دنبالهٔ 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 خطا وجود دارد و شما باید آن را برطرف کنید.

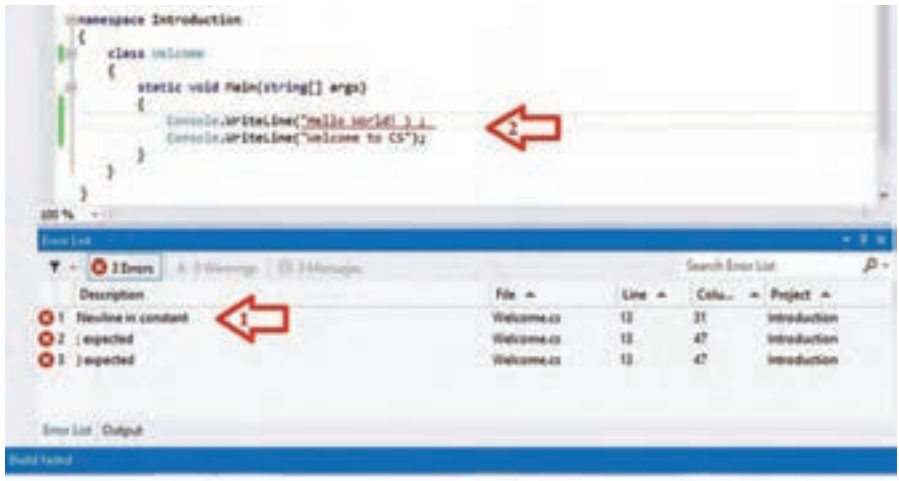


شکل ۱۵ــ۳ــ نشان دادن خطا در تایپ برنامه در محیط ویژوال استودیو

مثلاً در شکل ۱۵ــ۳ مشاهده می‌کنید که یک خط قرمز در زیر عبارت !Hello World کشیده شده است. با کمی‌دقت متوجه می‌شوید که علامت نقل قول انتهای پیام فراموش شده است! چون پیام‌ها به عنوان یک رشته باید بین علامت‌های نقل قول قرار داشته باشند.

۵ــ۳ــ ترجمه برنامه

بعد از نوشتن دستورات بالا، برای ترجمهٔ برنامه، کلید F7 یا F6 را بزنید. اگر برنامه خطا داشته باشد، خطاها در پنجره Error List دیده می‌شود. برای مثال شکل ۱۶ــ۳ خطاهای برنامه را در حالتی نشان می‌دهد که فراموش کرده‌اید انتهای رشته را با علامت " مشخص نمایید. در این شکل، پنجرهٔ Error List چندخطا را نشان می‌دهد؟



شکل ۱۶ــ۳ــ مراحل رفتن به محل خطا در برنامه

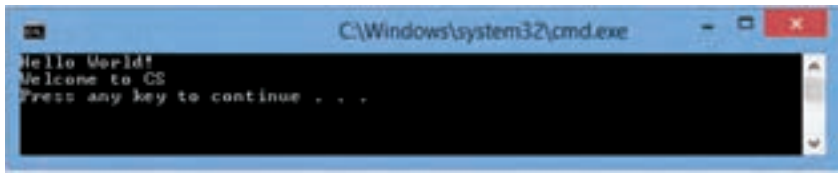
شاید این مسئله تا حدودی تعجب‌آور باشد که شما در تایپ برنامه، یک علامت نقل قول را فراموش کرده‌اید، اما در پنجره Error List، سه خطا گزارش می‌شود. بنابراین در ترجمه یک برنامه، انتظار گزارش تعداد خطای زیاد را داشته باشید. در حالت بروز خطا، باید با خواندن توضیح خطا و شماره خط برنامه که گزارش می‌شود، اقدام به رفع اشکال برنامه کنید. همواره از خطای اول شروع کنید، بنابراین در پنجره Error List، روی خطای اول دابل کلیک کنید (شکل ۱۶ــ۳ مرحله ۱) تا به‌طور مستقیم به پنجره ویرایشگر برنامه و محل خطای مزبور هدایت شوید (شکل ۱۶ــ۳ مرحله ۲). در این صورت اقدام به رفع اشکال کنید و سپس برنامه را دوباره ترجمه کنید. اگر خطایی 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 پنجرهٔ خروجی (Output) پیام ترجمهٔ بدون خطا و موفق را مانند شکل ۱۷ــ۳ مشاهده خواهید کرد.

شکل ۱۷ــ۳ــ پنجره خروجی در حالت بدون خطا

۶ــ۳ــ اجرای برنامه

بعد از ترجمه صحیح و بدون خطا، می‌توانید برنامه را اجرا کنید. برای اجرای برنامه از منوی DEBUG گزینه Start Without Debugging و یا از ترکیب کلیدی Ctrl + F5 استفاده کنید. با اجرای برنامه صفحه کنسول با سه پیام مانند شکل ۱۸ــ۳ دیده می‌شود. دو پیام خوش آمدگویی، همان‌هایی هستند که شما نوشته‌اید. اما پیام سوم Press any key to continue... به‌طور خودکار به وسیلهٔ 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 ۱۸ــ۳ــ 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.

۱ــ برنامه VS را نصب کنید.

۲ــ برنامه VS را اجرا کنید.

۳ــ 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.)

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

۵ــ با محیط IDE ویژوال استودیو آشنا شوید و پنجره‌هایی که در این فصل معرفی شده بودند، مانند پنجره Solution Explorer را شناسایی کنید.

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

۷ــ وارد پنجره ویرایشگربرنامه 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#");
```

۸ــ برنامه را ترجمه و سپس اجرا کنید.

۹ــ علامت نقطه ویرگول را از انتهای یکی از دستورات حذف کنید و دوباره برنامه را ترجمه کنید و به پنجره لیست خطاها دقت کنید. چند خطا گزارش شد؟

۱۰ــ در پنجره لیست خطاها، روی توضیح اولین خطا دوبار کلیک کنید تا به ویرایشگر برنامه و محل خطا بروید و اشکال را برطرف کنید.

۱۱ــ برنامه را ترجمه و اجرا کنید.

۱۲ــ علامت نقل قول قبل از حرف H در پیام Hello World! را حذف کنید و دوباره برنامه را ترجمه کنید و به پنجره لیست خطاها دقت کنید. چند خطا گزارش شد؟

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

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

```
static void Main()
```

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

۱۵ــ متد Beep() از کلاس کنسول برای ایجاد یک صدا یا صوت در برنامه استفاده می‌شود این متد را 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 متد Beep()، اعدادی را به ترتیب زیر بنویسید :

```
Console.Beep (مدت زمان برحسب میلی ثانیه , فرکانس برحسب هرتز);
```

مثلاً برای نواختن یک صدا با فرکانس ۵۰۰ هرتز 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 هرتز را خوب بشنود. صداهای بم فرکانس کم و صداهای زیر فرکانس بالا دارند.

۱۶ــ با قرار دادن اعداد مختلف به عنوان فرکانس در متد Beep()، محدوده شنوایی گوش خود را آزمایش کنید.

خودآزمایی فصل سوم

۱ــ نرم افزار Visual Studio یک..... بسیار پیشرفته برای برنامه‌نویسی به زبان C# است.

۲ــ IDE مخفف کلمات..... است و بیانگر یک محیط برنامه‌نویسی است که می‌توان در آن برنامه را تایپ کرد.

۳ــ در محیط VS علاوه بر تایپ برنامه، می‌توان برنامه را.....، عیب‌یابی و سرانجام ..... کرد.

۴ــ در محیط VS پس از ترجمه برنامه، خطاهای احتمالی آن در کدام پنجره دیده می‌شوند؟

۵ــ در پنجره Solution Explorer محیط VS، چه اطلاعاتی نشان داده می‌شود؟

۶ــ برنامه‌نویسی در محیط ویژوال استودیو چه برتری نسبت به استفاده از یک ویرایشگر دارد؟

۷ــ ترجمه برنامه در محیط VS چه برتری نسبت به پنجره فرمان و استفاده از مترجم CSC.EXE دارد؟

۸ــ 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)

۱ــ برنامه‌ای بنویسید که نام و نام خانوادگی و تاریخ تولد شما را نمایش دهد.

۲ــ برنامه‌ای بنویسید که ۳ صوت با فرکانس‌های ۵۰۰ و ۶۰۰ و ۷۰۰ هرتز را هر یک به مدت نیم ثانیه پشت سرهم ایجاد کند.

۳ــ برنامه‌ای بنویسید که اطلاعات هر سطر جدول زیر را در یک خط نمایش دهد.

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

۴ــ 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 واژه 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	

۴ 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 فراگیر بتواند:

- متغیر را تعریف کند و انواع متغیر را در برنامه‌های خود به کاربندد.
- انواع داده‌ها را نام ببرد و تفاوت کاربرد هر یک را توضیح دهد.
- میزان حافظه و محدودهٔ انواع داده‌ها را بیان کند.
- متغیرها را به‌طور صحیح در برنامه اعلان کند و آن‌ها را مقداردهی نماید.
- شکل نمایش نقطه شناور را توضیح دهد و اعداد اعشاری را در این قالب بنویسد.
- ازمتد ()ReadLine برای دریافت داده‌های یک برنامه از ورودی استفاده کند.
- بر روی رشته دریافتی از ورودی، تغییراتی داده و سپس نمایش دهد.
- از متد ()Parse برای تبدیل یک رشته به یک عدد استفاده کند.

۱ــ۴ــ متغیر چیست؟

در هر کامپیوتر حافظه‌های مختلفی وجود دارد که هر یک برای انجام کار خاصی پیش‌بینی شده است. یک نوع از حافظهٔ کامپیوتر که قادر است داده‌ها را نگهداری کند و به سرعت قابل دسترسی است، حافظه موقتی RAM[1] 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[2] 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 آشپزخانه 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.

۲ــ۴ــ روش اعلان (تعریف) 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;
```

دستور زیر را در نظر بگیرید :

```
int a;
```

۱ــ Random Access Memory

۲ــ Variable

در این دستورمتغیر 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 وسیلهٔ برنامه نویس انتخاب می‌شود که بهتر است نام و نوعِ آن مطابق با داده‌ای باشد که مقداردهی می‌شود.

۳ــ۴ــ نوع داده[۱] (نوع متغیر)

نوع متغیربه طور کلی ۳ ویژگی را مشخص می‌کند :

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

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

۳ــ 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 گروه‌بندی شده‌اند و نحوه نمایش یا نگهداری[۲] 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[۳] یا درون ساخته می‌گویند. جدول ۱ــ۴ انواع داده و مشخصات هر یک را نشان می‌دهد.

برای مثال در جدول ۱ــ۴ نوع داده sbyte را در نظر بگیرید. این نوع داده، اعداد صحیح و بدون ممیز در محدوده ۱۲۸- تا ۱۲۷+ 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[۴] قابل نمایش می‌باشند.

---

۱ ــ Data Type

۲ ــ Representation

۳ ــ Primitive Data Type or Built- In Data Type

۴ ــ Unsigned numbers

جدول ۱ــ۴ــ انواع داده

	نوع داده	کاربرد نوع داده	مقدار حافظه (بایت)	کمترین مقدار	بیشترین مقدار
عددی	sbyte	اعداد صحیح	1	−128	127
	byte	اعداد صحیح مثبت	1	0	255
	short	اعداد صحیح	2	−32768	32767
	ushort	اعداد صحیح مثبت	2	0	65535
	int	اعداد صحیح	4	− 2147483648	2147483647
	uint	اعداد صحیح مثبت	4	0	4294967295
	long	اعداد صحیح	8	−9223372036854778508	9223372036854778507
	ulong	اعداد صحیح مثبت	8	0	18446744073709551615
	float	اعداد اعشاری	4	-3.402823×10^{38}	3.402823×10^{38}
	double	اعداد اعشاری با دقت زیاد	8	$-1.79769313486232 \times 10^{308}$	$1.79769313486232 \times 10^{308}$
	decimal	اعداد صحیح بزرگ اعداد اعشاری با دقت بسیار زیاد	16	$-79228162514264337593543950335$ -7.9×10^{28}	$79228162514264337593543950335$ $+7.9 \times 10^{28}$
غیر عددی	bool	مقدار منطقی	1	false	true
	char	یک حرف یا علامت (کراکتر)	2	0 کد کراکتر مطابق با سیستم Unicode	65535 کد کراکتر مطابق با سیستم Unicode
	string	رشته		–	–
	object	آدرس یک داده		–	–

دستور byte age ; متغیری به نام age ایجاد می‌کند که این متغیر بسیار کوچک و به ظرفیت یک بایت است و می‌تواند 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 می‌کنیم :

```
sbyte minTemp , maxTemp ;
```

نکته

اگر متغیر مقدار دهی نشود خطا می‌گیرد. 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 حداکثر ۷ رقم استفاده می‌شود. در صورتی که ارقام عدد بیش از آن باشد عدد گرد می‌شود. مثلاً عدد ۱۲۳/۴۵۶۷۸۹ به صورت عدد ۱۲۳/۴۵۶۸ 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.

نکته

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 float mark;

۴ــ۴ــ 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 مستقیما مقداری در متغیر قرار می‌گیرد به این دستور، دستور انتساب[۱] می‌گویند.

نام متغیر = مقدار;

دستورات زیر را در نظر بگیرید :

```
byte  age ;
age  =  16 ;
```

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

۱ــ Assignment

در هنگام تعریف یا ایجاد متغیر نیز می‌توانید آن را مستقیما مقداردهی کنید که به آن **مقداردهی اولیه**[1] می‌گویند. الگوی آن چنین است :

نام‌متغیر نوع‌داده = مقدار;

بنابراین دو دستور قبل را با الگوی بالا جایگزین می‌کنیم :

```
byte age = 16 ;
```

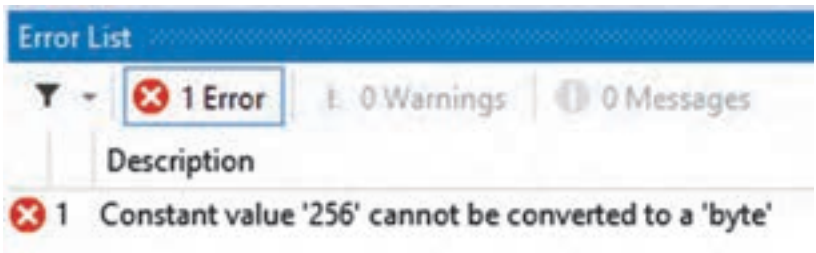
نکته

۱ــ برای مشخص کردن اعداد مثبت نیازی به قراردادن علامت + در پشت عدد نیست.
۲ــ در بین ارقام عدد نباید ویرگول قرار دهید تا ارقام عدد، دسته بندی و جدا شوند.
۳ــ اگر عددی را بخواهید در داخل یک متغیرذخیره 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 حداکثر عدد ۲۵۵ است، در هنگام ترجمه این دستور، خطای شکل ۱ــ۴ ظاهر می‌شود که شرح آن چنین است :
«مقدار ثابت ۲۵۶ را نمی‌تواند به یک byte تبدیل شود».



شکل ۱ــ۴ــ خطا در انتساب عدد صحیح ۲۵۶ در یک متغیر نوع byte

۱ ــ Initialize

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 0x یا 0X 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[۱].

برای مشخص کردن انواع عددی دیگر از نشانه‌های جدول ۲ــ۴ استفاده می‌شود که در انتهای عدد ذکر می‌شود.

جدول ۲ــ۴ــ نشانه‌های نوع اعداد ثابت

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

نکته

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

برای ذخیره اعداد اعشاری باید ازمتغیرهای نوع float یا double استفاده کنید. مثلاً برای نگهداری نمرات درسی(معمولاً با دو رقم اعشار) یا اعداد گنگ مانند π باید از چنین متغیرهایی استفاده کرد.

دستور زیر را در نظر بگیرید :

```
double PI = 3.141592653589793238;
```

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

---

۱ . (1B) = (1×16)+11=27 (FF)=(15×16) + 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 حافظهٔ اشغالی این متغیر دو برابر متغیر نوع float است.

دستورات زیر را در نظر بگیرید :

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

در دستورات بالا برای ذخیرهٔ نمرهٔ درس فیزیک متغیری اعلان و مقداردهی شده است.

سؤال: در دستور انتساب، بعد از عدد اعشاری 17.75 حرف f نوشته شده است که نشانه اعداد اعشاری با دقت معمولی است. آیا می‌توانید حرف f را ننویسید؟

نکته

در زبان C# هر عدد اعشاری داخل برنامه، به وسیلهٔ مترجم به عنوان نوع double در نظر گرفته می‌شود. بنابراین اگر بخواهید یک عدد ممیزی را در یک متغیر نوع float ذخیره کنید مترجم خطا یا 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.

۵ـ۴ـ نشان دادن محتوای متغیرها بر روی صفحه نمایش

معمولاً 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 متد 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 هرعدد، 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[1] 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 ۱ــ۴ــ استفاده از چند متغیر صحیح و اعشاری در برنامه ۱ــ۴ ، نشان داده شده است :

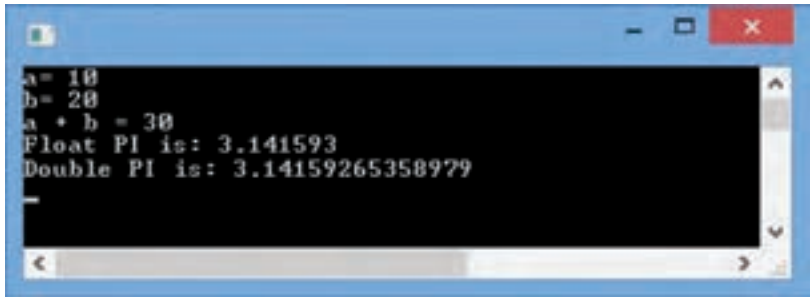
```
class VariableDemo
{
   static void Main()
    {
        // Declare some integer numbers variables
        int a = 10 , b = 20 ,c ;
        c = a + b;
        Console.WriteLine("a= " + a);
        Console.WriteLine("b= " + b );
        Console.WriteLine("a + b = " + c);
        // Declare some real numbers variables
        float lowPI = 3.14159265358979238f;
        double highPI = 3.14159265358979238;
```

۱ــ Concatenate

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


            Console.ReadKey();
        }
    }
```

برنامه ۱ــ۴ــ تعریف و مقداردهی و نمایش محتوای متغیرها

در برنامه ۱ــ۴، 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 π با دقت‌های مختلف نگهداری شده است.



شکل ۲ــ۴ــ خروجی برنامه ۱ــ۴

۶ــ۴ــ نحوه نام گذاری 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 قبلاً در خانواده و یا نزدیکان انتخاب نشده باشد، به همان صورت برنامه نویس نیز برای متغیرها باید یک نام صحیح، بامعنا و غیرتکراری در محدوده آن را انتخاب کند این کار باید با حوصله انجام شود و نام انتخابی نباید با نام‌های دیگر یکسان باشد.

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

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

۲ــ نام متغیر نمی‌تواند با عدد شروع شود.

۳ــ نام انتخابی نمی‌تواند با کلمات کلیدی و یا رزرو شده باشد.

۴ــ استفاده ازعلامت 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 نکنید 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 روشِ نوشتن نام، کوهان شتری[1] می‌گویند.

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

fileName , userName , notFound , localIP

روش دیگری برای نام‌گذاری متغیرها به‌نام روش مجارستانی[2] به وسیلهٔ آقای چارلز سیمونیی[3] 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 نام‌گذاری متغیرها استفاده شده است.

نکته

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

در جدول صفحه بعد تعدادی نام متغیر و علت مجاز یا غیرمجاز بودن این نام‌ها را می‌بینید.

۱ــ Camel Notation

۲ــ Hungarian Notation

۳ــ Charles Simonyi

جدول ۳ــ۴ــ 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 مقداردهی متغیرها را مشاهده می‌کنید :

```
int speed = 70 ;   // تعریف متغیر برای نگهداری سرعت خودرو با مقداردهی اولیه
float  a , b , c ;          //  Triangle sides تعریف سه متغیر برای اضلاع مثلث
float  triangleArea ;     //  تعریف یک متغیر برای نگهداری مساحت مثلث
double electricalCharge;  //  متغیری برای نگهداری بار الکتریکی یک جسم
```

۷ــ۴ــ 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 روشی شبیه نماد علمی‌بنویسید. برای اینکه با این روش آشنا شوید ابتدا لازم است روش نماد علمی‌را 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 ۱۰ است که به آن **نما** گفته می‌شود (جدول ۴ــ۴).

جدول ۴ــ۴ــ مثال‌هایی از فرم نماد علمی

فرم نماد علمی	فرم معمولی
4.38×10^{6}	4380000
2.65×10^{-5}	.0000265
4.79832×10^{1}	47.9832
1×10^{7}	10000000
-5.6×10^{3}	−5600

۱ــ۷ــ۴ــ فرم نقطه شناور : در زبان C# از یک فرم نماد علمی‌برای نمایش اعداد اعشاری استفاده می‌شود که به آن **فرم نقطه شناور**[۱] گفته می‌شود. در این فرم مانند نماد علمی، عدد از دو بخش مانتیس و نما تشکیل شده است که با حرف E از یکدیگر جدا شده‌اند. در این فرم، توان ۱۰ بعد از حرف E نوشته می‌شود وخبری از علامت ضرب بین دو قسمت نیست (جدول ۵ــ۴).

جدول ۵ــ۴ــ مثال‌هایی از نمایش اعداد در فرم نقطه شناور

نمایش عدد در فرم نقطه شناور	عدد
7.5924E1	75.924
1.8E−1	0.18
4.53E − 5	0.0000453
−1.482E0	−1.482
7.8E3	7800.0

بار الکتریکی یک الکترون $۱/۶۰۲ \times ۱۰^{-۱۹}$ 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 بالارا ترجیح می‌دهید؟ چرا؟

۱ــ Floating point notation

۲ــ۷ــ۴ــ دقت اعداد قابل نمایش در فرم نقطه شناور : حداکثر تعداد ارقام غیر صفر و با معنی مانتیس عدد را، دقت عدد می‌نامند. دقت اعداد نوع float ۷ رقم و اعداد نوع double ۱۵ رقم است.

نکته

به غیر از میزان حافظه مصرفی و محدوده اعداد قابل نمایش در نوع داده‌های float و double، میزان دقت این دو نوع داده نیز با یکدیگر متفاوت است.

۸ــ۴ــ نوع داده منطقی یا بولین[۱] (bool)

در انتهای جدول ۱ــ۴ نوع داده منطقی یا بولین (bool) را مشاهده می‌کنید این نوع داده فقط شامل دو مقدار درست (true) و نادرست (false) است. متغیرهایی که از این نوع داده تعریف و ایجاد می‌شوند، قادرند یکی از دو مقدار true و false را بپذیرند که با حروف کوچک انگلیسی نوشته می‌شوند.

دستورات زیر متغیر response را اعلان و با false مقدار دهیِ اولیه می‌کند. سپس محتوای متغیر بر روی صفحه نمایش چاپ می‌شود.

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

۹ــ۴ــ نوع داده حرفی یا کاراکتری char

کاراکترعبارت است از یک حرف الفباء یا یک علامت و یا نشانه‌هایی مانند آنچه که در روی دکمه‌های صفحه کلید مشاهده می‌کنید. در کامپیوتر برای هر دکمه صفحه کلید یک کدِ عددی در نظر گرفته می‌شود و در واقع هنگامی‌که یک کلید را فشار می‌دهید کدی متناظر با آن کلید تولید و این کد به صورت دنباله‌ای از صفر و یک در حافظه کامپیوتر ذخیره می‌شود. یک کاراکتر را می‌توانید با کد آن مشخص کنید و یا علامت آن را در بین علائم '' (تک کوتیشن) قرار دهید.

چند نمونه از کاراکترها را در زیر مشاهده می‌کنید.

```
'A'   ,   'a'   ,   '&'   ,   '$'   ,   '+'   ,   ''
```

۱ــ Boolean

نکته

- در داخل علامت‌ها فاصله (Space) نیز به عنوان یک کاراکتر در نظر گرفته می‌شود.
- در داده کاراکتری، فقط یک کاراکتر باید بین علائم '' 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 کاراکترعددی بین ۰ تا ۶۵۵۳۵ است و تمام نشانه‌ها، علائم و حروف الفباء زبان‌های مختلف کشورها به وسیلهٔ 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 علائم '' از کد آنها استفاده کنید، چون کاراکترها فقط محدود به آنچه که بر روی صفحه کلید قرار دارد نیستند. بنابراین با دانستن 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 علائم '' قرار می‌دهیم و برای مشخص کردن این عدد به عنوان کد کاراکتر، قبل از آن ، علامت \u یا \x را می‌نویسیم مانند الگوی زیر :

```
'\u کدچهارقمی'
```

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'
```

۱۰ــ۴ــ 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 دادهٔ رشته‌ای شامل ۸ کاراکتر است.

۱ــ۱۰ــ۴ــ 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 رشته ای 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 متغیرname 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؟

۲ــ۱۰ــ۴ــ عملیات بر روی داده‌ها یا متغیرهای رشته‌ای: عملیات مختلفی بر روی رشته‌ها می‌توان انجام داد، یکی از عملیات 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 متغیرmessage 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؟

نکته

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

کار در کارگاه ۱

مثال ۲ــ۴ : برنامه زیر مانند برنامه ۱ــ۴ 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 ۲ــ۴ــ دقت در استفاده از علامت + در هنگام کار با اعداد و رشته‌ها

در خط آخر برنامه ۲ــ۴، علامت + دو بار استفاده شده است که هر دو علامت، عمل الحاق رشته را انجام می‌دهند . خروجی این برنامه مطابق شکل ۳ــ۴ است :

شکل ۳ــ۴ــ خروجی برنامه ۲ــ۴

برای رفع اشکال برنامه ۲ــ۴، خط آخر را به صورت زیر بازنویسی می‌کنیم :

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

با تصحیح خط آخر، نتیجه اجرای برنامه شکل ۴ــ۴ خواهد شد :

شکل ۴ــ۴ــ خروجی برنامه۲ــ۴ پس از تصحیح

سؤال: با توجه به خروجی برنامه ۲ــ۴ چرا پرانتز سبب تغییر مقدار خروجی شد؟

۱۱ــ۴ــ دریافت رشته

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

متد ReadLine() مانند متدهایی که تاکنون خوانده‌ایم در کلاس Console تعریف شده است و در فضای نامی System قرار دارد. بنابراین به‌صورت زیر استفاده می‌شود :

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

کامپیوتر با اجرای این متد متوقف شده و منتظر دریافت داده می‌شود. کاربر می‌تواند داده مورد نظر خود را تایپ کند و در پایان دکمهEnter را بزند 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 متد ReadLine() برای دریافت نام و نام خانوادگی به صورت زیر استفاده می‌نماییم :

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

نکته

متد ReadLine() شبیه متد ReadKey() است با این تفاوت که متد ReadKey() 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.

مثلاً برای دریافت نام کاربر دستورات زیر را می‌نویسیم :

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

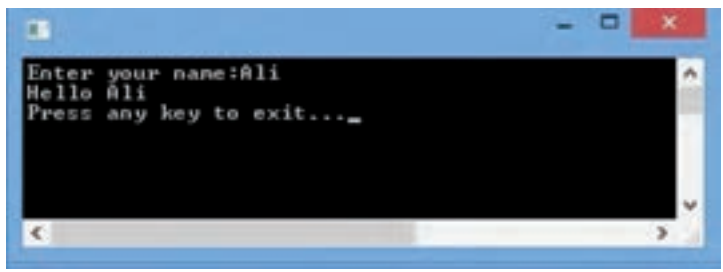
کار در کارگاه ۲

مثال ۳ــ۴ : نام کاربر از ورودی دریافت شده و خطاب به او پیام خوشامدگویی اعلام شود.

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

برنامه ۳ــ۴ــ خوش‌آمدگویی به کاربر

اگر فرض کنید که کاربر، نام Ali را وارد کند، خروجی برنامه به صورت شکل ۵ــ۴ خواهد بود :



شکل ۵ــ۴ــ خروجی برنامه ۳ــ۴

مثال ۴ــ۴ : می‌خواهیم به برنامه ۳ــ۴، دستوراتی اضافه کنیم که علاوه بر دریافت نام کاربر، نام خانوادگی وی نیز سؤال شود و سپس نام ونام خانوادگی را در یک خط نمایش دهد.

در برنامه ۳ــ۴، کافی است یک متغیر رشته‌ای به نام family تعریف کرده و از متد()ReadLine 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.Write("Enter your name: ");
        name = Console.ReadLine();

        Console.Write("Enter your family: " );
        family = Console.ReadLine();

        Console.WriteLine("Hello "+ name +" "+ family);

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

برنامه ۴ــ۴ــ تکمیل برنامه خوش‌آمدگویی به کاربر

سؤال: خروجی برنامه تغییر یافته چه تفاوتی با شکل ۵ــ۴ دارد؟

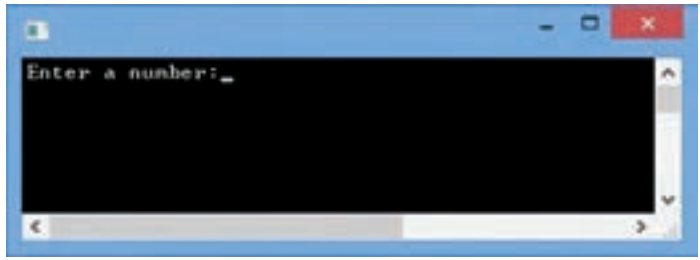
مثال ۵ــ۴ : می‌خواهیم برنامه‌ای بنویسیم که دو عدد دلخواه از کاربر دریافت کند و مجموع آن‌ها را حساب کرده و روی صفحه نمایش، نشان دهد.

برای دریافت داده‌ها از کاربر، از متد ReadLine() مانند مثال‌های قبلی استفاده می‌کنیم. داده‌های دریافتی به وسیلهٔ این متد، در قالب رشته در حافظه ذخیره می‌شوند، بنابراین برای دسترسی به آن ها باید از متغیرهای رشته‌ای استفاده کنیم.

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

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

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

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

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

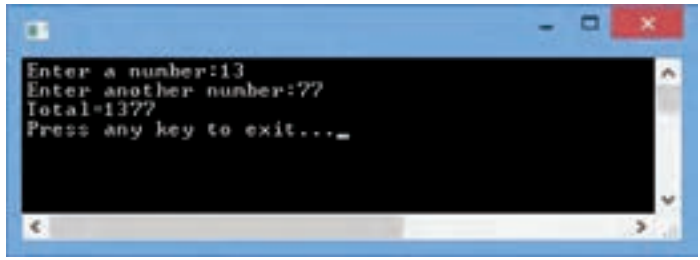
برنامه ۵ــ۴ــ اولین تلاش برای دریافت داده‌های عددی

با اجرای این برنامه، پنجره‌ای ظاهر می‌شود که از کاربر خواسته می‌شود که یک عدد وارد کند (شکل ۶ــ۴).



شکل ۶ــ۴ــ خروجی برنامه ۵ــ۴ دریافت یک عدد

پس از وارد کردن یک عدد و زدن دکمه Enter، عدد دیگری خواسته می‌شود. فرض کنید اعداد ۱۳ و ۷۷ توسط کاربر وارد شود (شکل ۶ــ۴).



شکل ۷ــ۴ــ خروجی برنامه ۵ــ۴

سؤال: 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 متد ReadLine() داده دریافتی را به صورت یک رشته در حافظه ذخیره می‌کند و در برنامه ۵ــ۴ از متغیرهای رشته‌ای firstNumber و secondNumber برای دسترسی به داده‌های ورودی استفاده کردیم. بنابراین علامت + در دستور زیرعمل الحاق دو رشته مثلاً "13" و "77" را انجام می‌دهد و طبیعی است که نباید انتظار عمل جمع ریاضی داشته باشیم .

۱ــ۱۱ــ۴ــ دریافت اعداد : با توجه به این که داده‌های دریافتی به وسیله متد ReadLine() همواره به صورت رشته تحویل داده می‌شود باید به وسیله دستوری، رشته دریافتی را به عدد تبدیل کنیم. بنابراین به متدی نیاز داریم که بتواند یک رشته شامل ارقام را به ارزش عددی تبدیل کند تا بتوانیم روی آنها محاسبات ریاضی انجام دهیم.

خوشبختانه برای انواع داده‌های عددی، متدی به نام Parse() از قبل 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 متد 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**[1] می‌گویند.

حاصل اجرای این متد، عدد ۲۵۹ است که باید در یک متغیر نوع صحیح ذخیره شود. بنابراین استفاده مفید از این متد 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 متد 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 متد Parse() می‌توانیم رشته دریافتی که به وسیلهٔ متد 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);
```

۱ـ Parse

همچنین می‌توانید متد ReadLine() را مستقیماً در متد 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؟

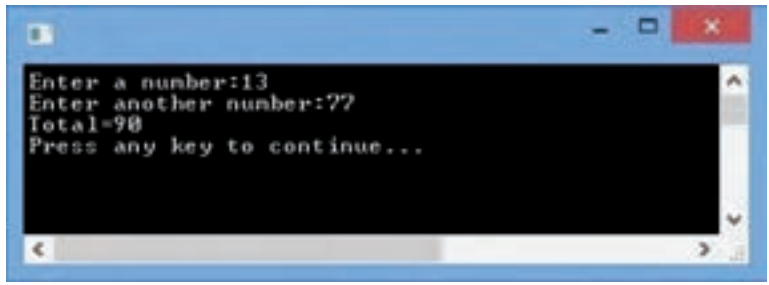
کار در کارگاه ۳

مثال ۶ــ۴ : با تکمیل برنامه ۵ــ۴ مجموع دو عدد دریافتی را چاپ نمایید.

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

        Console.Write("Enter a number: ");
        input = Console.ReadLine();
        firstNumber = float.Parse(input);

        Console.Write("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 ۶ــ۴ــ دریافت دو عدد و محاسبه مجموع

نتیجه اجرای برنامه در شکل ۸ــ۴ نشان داده شده است :



شکل ۸ــ۴ــ خروجی برنامه ۶ــ۴

برنامه‌های زیر را در محیط VS ایجاد کنید.

۱ــ 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;
```

۲ــ برنامه شماره ۴ــ۴ (خوش‌آمدگویی به کاربر) را با داده‌های مختلف (نام خود، نام همکلاسی‌ها) آزمایش کنید.

۳ــ برنامه شماره ۶ــ۴ (دریافت دو عدد و محاسبه مجموع) را با اعداد صحیح و اعشاری آزمایش کنید.

خودآزمایی فصل چهارم

۱ــ چه نوع حافظه کامپیوتر برای نگهداریِ حجمِ کمی‌از داده‌ها در طول اجرای یک برنامه مناسب است؟

۲ــ در زبان‌های برنامه‌نویسی، مکانی از حافظه برای نگهداری موقتی داده واطلاعات ..... نامیده می‌شود.

۳ــ منظور از نوع داده چیست؟

۴ــ نوع متغیر چه ویژگی‌هایی را نشان می‌دهد؟

۵ــ تفاوت‌های بین نوع داده float و double را نام ببرید.

۶ــ 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
ت) ظرفیت یک‌هارد دیسک امروزی
ث) جمعیت یک کشور
ج) وضعیت خاموش و روشن بودن یک لامپ

۷ــ تحقیقی کوتاه بر روی روش نام گذاری مجارستانی (Hungarian Notation) با استفاده از اینترنت داشته باشید. با توجه به محیط‌های برنامه‌نویسی (IDE) پیشرفته، مانند ویژوال استودیو، نشان دهید که دیگر نیازی به ذکر نوع داده در ابتدای نام متغیر که در روش مجارستانی استفاده می‌شود، وجود ندارد.

۸ــ شکل زیر چه روشی را برای نام گذاری متغیرها نشان می‌دهد؟ نام این روش چیست؟

۹ــ چرا در هنگام ترجمه دستور زیر، خطا ظاهر می‌شود؟چگونه این خطا را برطرف می‌کنید؟

```
short value = 66000 / 2 ;
```

۱۰ــ در جدول زیر، کدامیک از نام‌های متغیر، غیر مجاز است و یا مناسب نیستند. آنها را تصحیح کنید. (اولین ردیف جدول برای شما پاسخ داده شده است.)

نام متغیر	کاربرد متغیر یا توجه به معنی آن	نام متغیر			نام پیشنهادی شما
		مجاز	نامناسب	غیرمجاز	
networkOK	وضعیت شبکه	■	□	□	
29yesitsme		□	□	□	
myCurrentID		□	□	□	
intValue		□	□	□	
8%tax		□	□	□	
woodLength		□	□	□	
glassArea		□	□	□	
width		□	□	□	
height		□	□	□	
MySalary		□	□	□	
employeeSalary		□	□	□	

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

'۰۸' , '۱۲' , '۱' , '&' , '!' , '+' , 'abc' , '!='

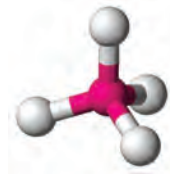
۱۲ــ در زبان C#، برای دریافت داده از ورودی، از متد ..... و برای نمایش اطلاعات در خروجی، از متد..... استفاده می‌کنیم.

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

الف) عمر زمین بر حسب سال ۴,۶۰۰,۰۰۰,۰۰۰

ب) فاصله زمین تا خورشید ۱۴۹,۶۰۰,۰۰۰,۰۰۰ متر است.

پ) اندازه جرم یک اتم کربن (جرم اتمی) برابر با ۱۶۶kg۰/۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰

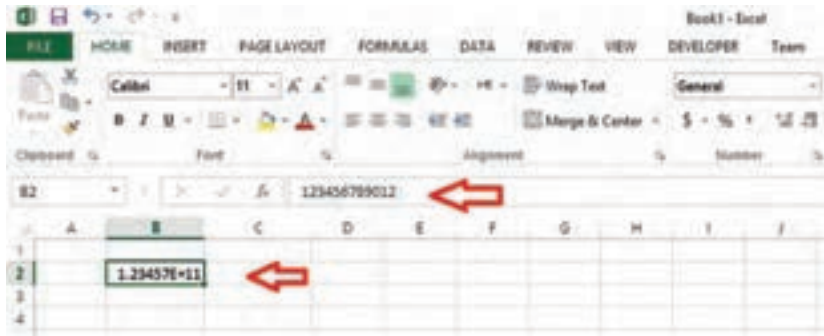


۱۴ــ با توجه به اینکه لازم نیست مانتیس بین ۰ تا ۱ باشد، جدول زیر را تکمیل کنید.

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	

۱۵ــ در برنامه‌های صفحه گسترده، اعداد بزرگ به صورت نقطه شناور نیز نمایش داده می‌شوند. وارد برنامه اکسل شوید و عدد بزرگی را وارد نموده و کلید Enter را بزنید. چه عددی روی صفحه نشان داده می‌شود؟

پس از مشاهده فرم نقطه شناور، سعی کنید با تغییر فرمت سلول مربوطه، عدد را به صورت معمولی نشان دهید.



۱۶ــ کدام‌یک از دستورات زیر،می‌تواند مجموع دو عدد a و b را به‌طور 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);
System.Console.WriteLine("a + b"+(a + b));
```

۱۷ــ 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()
C) WriteLine()
B) DisplayLine()
D) Write()

تمرینات برنامه‌نویسی فصل چهارم

۱ــ 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.)

۲ــ 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.

۳ــ با استفاده از یک برنامه ساده شامل متد 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\u00200"	
"b\u0061ck"		"C\u0023"	
'\u000a'		'\u0040'	
"12" + "8"		'\u0030'	

واژگان و اصطلاحات انگلیسی فصل چهارم

ردیف	واژه انگلیسی	معنی واژه 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	

۵ 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 دادهٔ 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 فراگیر بتواند:

۱ــ عملوند، عملگر و عبارت محاسباتی را تعریف کند و آنها را به درستی به کارگیرد.

۲ــ عملگرهای ریاضی را نام ببرد و در عبارت محاسباتی از آنها استفاده کند.

۳ــ حاصل عبارت محاسباتی را با استفاده از جدول تقدم عملگرها به دست آورد.

۴ــ کاربرد عملگرهای افزایشی، کاهشی و انتساب را بیان نماید.

۱ــ۵ــ عبارت[۱] چیست؟

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

$8+3*5$

$9-7.25$

$5.6*3+5.6$

$-\frac{y}{6}+25-x$

۱ــ Expression

در این عبارت‌ها، علامت + نشانهٔ عمل جمع و علامت * نشانهٔ عمل ضرب است. به این علامت‌ها که بیانگر انجام یک عمل بر روی اعداد و داده‌ها هستند، **عملگر**[۱] گفته می‌شود. مثلاً عملگر * در عبارت 5*3+8 بر روی اعداد 3 و 5 عمل ضرب را انجام می‌دهد و همچنین عملگر + در عبارت بالا بر روی عدد 8 و نتیجه حاصلضرب یعنی عدد 15، عمل جمع را انجام می‌دهد. به اعدادی که یک عملگر بر روی آنها عملی را انجام می‌دهد **عملوند**[۲] می‌گویند. اعداد 3 و 5 عملوندهای عملگر ضرب و عدد 8 و 15 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 عملگرها، عملگرهای **دوتایی**[۳] گفته می‌شود. عملگر - در عبارت 7.25 - 9 عملگر تفریق است که آن نیز یک عملگر دوتایی است و حاصل تفریق 7.25 از 9 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**[۴] 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.

۲ــ۵ــ عملگرهای ریاضی یا حسابی[۵]

اگر در عبارتی بیش از یک عملگر وجود داشته باشد ابتدا عملگری عمل خود را انجام می‌دهد که اولویت[۶] 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، عملگرها به ترتیب انجام می‌شوند که به آن «**شرکت‌پذیری چپ**[۷]» می‌گویند.

در جدول ۱ــ۵ لیست عملگرهای ریاضی را به ترتیب اولویت مشاهده می‌کنید. عملگر قرینه اولویت بالاتری نسبت به بقیه عملگرهای ریاضی دارد و عملگرهای جمع و تفریق دارای اولویت یکسان ولی کمترین اولویت را در بین عملگرهای ریاضی دارند.

۱ــ Operator
۲ــ Operand
۳ــ Binary Operator
۴ــ Unary Operator
۵ــ Arithmetic
۶ــ Precedence
۷ــ Left_Associative

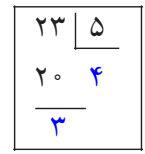
جدول ۱ــ۵ــ عملگرهای ریاضی

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

عملکرد عملگرهای جمع و تفریق و ضرب مانندعملکرد آنها در ریاضیات است اما عملگر تقسیم با توجه به نوع عملوندهایش می‌تواند تقسیم صحیح و بدون ممیز و یا تقسیم اعشاری و ممیزی انجام دهد. مثلاً در عبارت 9 / 2 چون عملوندها اعداد صحیح هستند بنابراین تقسیم بدون ممیز و صحیح انجام خواهد شد که نتیجه آن عدد ۴ است . اما در عبارت 9.0 / 2 یا در عبارت 9 / 2.0 ، چون حداقل یکی از عملوندها، اعشاری است بنابراین تقسیم به صورت اعشاری انجام می‌شود که حاصل عبارت عدد ۴/۵ است.

در جدول ۱ــ۵، عملگر جدیدی نیز به نام باقیمانده تقسیم که با نشانهٔ % مشخص می‌شود، مشاهده می‌کنید. به وسیلهٔ این عملگر می‌توانیم باقیماندهٔ تقسیم یک عدد بر عدد دیگر را با توجه به خارج قسمت صحیح و بدون اعشار به دست آوریم. مثلاً در تقسیم عدد ۲۳ بر عدد ۵، خارج قسمت بدون اعشار عدد ۴ است بنابراین باقیمانده عدد ۳ است.

```
23 / 5 = 4
23 % 5 = 3
```



برای تغییر دادن اولویت عملگرها، از علامت‌های پرانتز استفاده می‌شود. مثلاً در عبارت زیر ابتدا عمل جمع و سپس عمل ضرب انجام می‌شود.

```
5.6 *(3 + 6.5)
```

اگر چند پرانتز تو در تو نیز وجود داشته باشد، ابتدا داخلی‌ترین پرانتز انجام می‌شود.

در جدول ۲ــ۵، چند نمونه از عبارت‌های ریاضی نشان داده شده است.

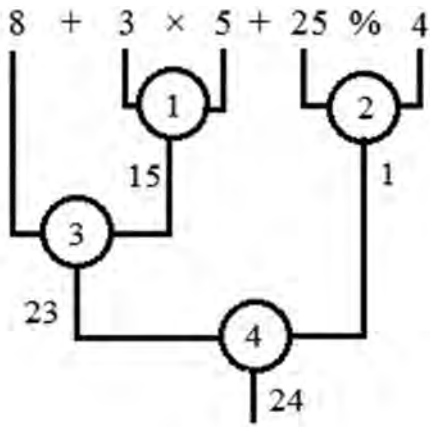
جدول ۲ــ۵ــ مثالی از عبارت‌های ریاضی

نوع عبارت	حاصل عبارت	عبارت
صحیح	5	175 / 31
صحیح	20	175 ٪ 31
اعشاری double	3.75	7.5 / 2
اعشاری double	1.5	7.5 ٪ 2
اعشاری double	18.0	36 / 2.0
صحیح	0	36 ٪ 2
صحیح	254138	2541389 / 10
صحیح	9	2541389 ٪ 10

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 ابتدا عملگری 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 شرکت‌پذیریِ 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 ۱ــ۵).



نمودار ۱ــ۵ــ ترتیب اجرای عملگرها

در برنامه‌ها، معمولاً حاصل یا 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 بریزیم 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# روی این قوانین سخت گیر[1] است زیرا می‌خواهد از اشتباهات برنامه نویسان جلوگیری نماید، این یکی از ویژگی‌های زبان C# است.

۱ــ اگر حاصل یک عبارت عدد صحیح باشد بسته به اندازه و بزرگی عدد، می‌تواند در یک متغیرِ نوعِ صحیح که گنجایش آن مساوی یا بزرگ‌تر از حاصل عبارت باشد جای گیرد.

مثلاً حاصل عبارت 31 / 175 عدد 5 است این عدد کوچک می‌تواند در تمام متغیرهای نوع صحیح زیر قرار گیرد.

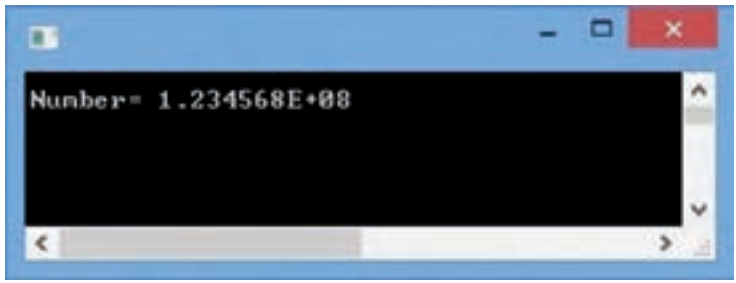
sbyte , byte , short , ushort , int , uint , long , ulong

ولی در عبارت 2541389 / 10 چون حاصل عبارت عدد 254138 است که عدد صحیح بزرگی است فقط در متغیرهای نوع ulong , long , uint , int قابل نگهداری است.

۲ــ 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.

مثلاً در دستور زیر حاصل عبارت 1234567890 / 10 در متغیر اعشاری ذخیره می‌شود ولی به دلیل اینکه عدد بزرگی است به صورت گرد شده در متغیر ذخیره می‌شود (شکل ۲ــ۵).

```
float number = 1234567890 /  10;
Console.WriteLine("Number=" + number);
```



شکل ۲ــ۵ــ نتیجهٔ اجرای عملگر تقسیم

۱ــ Strict type checking language

همان طور که در شکل ۲ــ۵ 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 :

$1.234568E+08 \rightarrow 1.234568 \times 10^{8} = 123456800$

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

۳ــ 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.

۴ــ 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 نتیجهٔ 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 ذخیرهٔ معدل دانش‌آموز تعریف می‌کند تا نتیجهٔ تقسیم را در آن ذخیره نماید.

```
double meanScore;
```

اکنون می‌توانید مقدار عبارت را در متغیر مزبور 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 15.6786714285714 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 خطا می‌دهد. چون در سمت راست علامت انتساب، 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 خطایی را صادر می‌کند (شکل ۳ــ۵).

شکل ۳ــ۵ــ خطای تبدیل نوع double به نوع float

شکل ۳ــ۵ 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 حداکثر 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 به‌عنوان 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 ۱ــ۵ : مجموع و معدل نمرات درسی را طبق جدول ۳ــ۵ محاسبه کنید.

جدول ۳ــ۵ــ نمرات درسی یک دانش‌آموز

نام درس	نمره درس	تعداد واحد[1]
فیزیک	۱۷/۵	۳
شیمی	۱۹	۲
ریاضی	۱۴/۵	۴
ادبیات	۱۸	۲
ورزش[2]	۱۹	۱

به دلیل این که می‌خواهیم روی داده‌ها پردازش انجام دهیم بهتر است ابتدا نمرات را داخل متغیرهای مناسب ذخیره کنیم :

```
float physicMark = 17.5F, chemistryMark=19, mathMark=14.5F;
float litracyMark=18, PEMark=19;
```

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

```
int physicCredit=3, chemistryCredit=2, mathCredit=4;
int literacyCredit=2, PECredit=1;
```

حال می‌خواهیم مجموع نمرات را به دست آوریم. با توجه به اینکه تعداد واحد هر درس مختلف است نمی‌توانیم به سادگی نمرات را با یکدیگر جمع کنیم بلکه باید هر نمره را در تعداد واحد درسی مربوطه ضرب کنیم و سپس حاصل ضرب‌های 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 می‌دهیم :

۱ــ Credit Hour

۲ــ Physical Education

```
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 بردن خوانایی عبارت یا برنامه از پرانترها 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 :

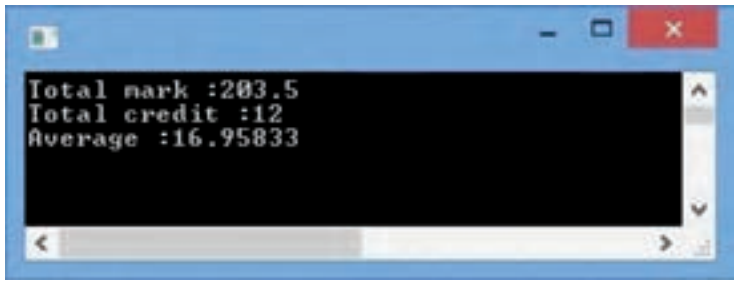
```
Console.WriteLine("Total mark: " + totalMark);
Console.WriteLine("Total credit: " + totalCredit);
Console.WriteLine("Average: " + average);
```

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 litracyMark = 18, PEMark = 19;
        int physicCredit = 3, chemistryCredit = 2, mathCredit = 4;
        int literacyCredit = 2, PECredit = 1;
        float totalMark;
        totalMark = (physicMark * physicCredit) +
        (chemistryMark * chemistryCredit) +
        (mathMark * mathCredit) +
        (litracyMark * 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 ۱ــ۵ــ محاسبه معدل نمرات یک دانش آموز

خروجی برنامه چنین است :



شکل ۴ــ۵ــ خروجی برنامه ۱ــ۵

به مثال ۱ــ۵ 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.

۳ــ۵ــ عملگرهای افزایشی[۱] و کاهشی[۲]

در زبان برنامه‌نویسی C#، عملگرهای **یکتایی** ++ و -- به ترتیب برای افزایش و کاهش مقدار یک متغیر به اندازهٔ یک واحد در نظر گرفته شده است. البته این عملگرها از زبان C وارد این زبان شده‌اند و برای کوتاه شدن و تایپ کمتر دستورات، ابداع شده‌اند که در زبان‌های دیگر امروزی مانند جاوا نیز وجود دارند (جدول ۴ــ۵).

جدول ۴ــ۵ــ (عملگرهای افزایشی و کاهشی)

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

۱ــ Increment ۲ــ Decrement

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

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

یا

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

عملگر افزایشی یا کاهشی را می توانید قبل از نام متغیر و یا بعد از نام متغیر ذکر کنید که درهر دو حالت سبب می‌شود مقدار متغیر به اندازهٔ یک واحد تغییر کند. تفاوت بین این دو حالت را در سؤالات ۱۱و۱۲ خودآزمایی ببینید.

۴ــ۵ــ عملگرهای انتساب

در جدول ۵ــ۵ عملگرهای انتساب دیده می‌شود.

نوع عملگر	نشانه
انتساب	= , += , -= , *= , /= , %= &= , |= , ^= , <<= , >>=

با عملگر = و کاربرد آن در فصل قبل آشنا شدید که برای ذخیره کردن یک مقدار در یک متغیر استفاده می‌شود. 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 (عدد۶) به متغیر y منتسب می‌شود و y نیز برابر ۶ می‌شود.

نکته

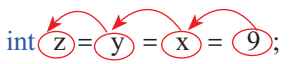
حاصل عبارتی که دارای عملگر انتساب است، مقدار داده یا متغیری است که در سمت راست عملگر واقع شده است.

x=6; یعنی x ←— 6

مثلاً در دستور زیر

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

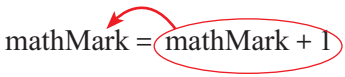
چند عملگر انتساب وجود دارد و از سمت راست به ترتیب انجام می‌شوند. ابتدا عدد ۹ در متغیر x قرار می‌گیرد و سپس حاصل عبارت که عدد ۹ است در متغیر y کپی شده و سپس این مقدار در متغیر z کپی می‌شود. یعنی در عبارتی که عملگرهای انتساب وجود دارد این عملگرها از سمت راست به چپ انجام می‌شوند (شرکت پذیر راست) بر خلاف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



اکنون اگر بخواهیم یک نمره، به نمره درس ریاضی در مثال ۱ــ۵ اضافه کنیم، با توجه به اینکه نمرهٔ هر درس را در یک متغیر ذخیره کردیم، دستور لازم برای افزایش یک واحد به متغیر مربوطه را بنویسید.

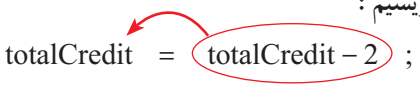
```
mathMark+=1
```

متغیر mathMark حاوی نمره درس ریاضی است. برای افزایش یک نمره از دستور انتساب زیر استفاده می کنیم :



با اجرای این دستور انتساب، ابتدا کامپیوتر عبارت سمت راستِ علامت = را محاسبه می کند. بدین منظور محتوای متغیر mathMark که برابر با ۱۴/۵ است با عدد یک جمع می‌شود و حاصل عبارت یعنی ۱۵/۵را در متغیر سمت چپ، جایگزین مقدار قبلی آن می کند. بنابراین از این به بعد محتوای متغیر مزبور 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 :



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

در جدول ۵ــ۵، علاوه بر عملگر =، عملگرهای دیگر انتساب همراه با یک عملگر ریاضی وجود دارند. عملگرهای =+ و =- 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 می کنید :

```
x *= 3;
```

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

مثال ۲ــ۵ : 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 x*= y ; 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 x*= y ; چه دستوری است؟

خودآزمایی فصل پنجم

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

۲ــ عبارت ریاضی $۴x + \frac{۵x}{y+۳}$ را با توجه به جدول ۱ــ۵ (اولویت عملگرها) به صورت یک عبارت قابل قبول در C#، بنویسید.

۳ــ مترجم در ترجمه کدام یک از دستورات زیر، خطا می‌دهد؟ دلیل خود را ذکر کنید.

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

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

۵ــ در دستورات زیر، نمودار اجرای عملگرها را مانند شکل ۱ــ۵ 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) ;
```

۶ــ برای انجام عبارت‌های ستون سمت راست، از چه عملگری در ستون چپ استفاده می کنید؟

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

۷ــ 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 ()WriteLineبالا، با چه فرضی فضای نامی System در ابتدای هر خط، ذکر نشده است؟

۸ــ اگر متغیر z را یک‌بار با عدد ۵ و بار دیگر با عدد ۳ مقداردهی کنیم، دستور Console.Writeline(z); چه مقداری را در نهایت چاپ می‌کند؟ توضیح دهید؟

۹ــ برنامه‌ای بنویسید که دو عدد از ورودی دریافت نماید و مقادیر درون دو متغیر را جابجا کند.
(از مثال جابجایی محتوای دو لیوان حاوی شیر و نوشابه کمک بگیرید)

۱۰ــ 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

۱۱ــ نتیجه اجرای دستورات زیر چیست؟

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

۱۲ــ 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);
```

۱۳ــ سؤال 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;
```

F)
```
int i, j;
i = j = 10;
```

تمرینات برنامه‌نویسی فصل پنجم

۱ــ نیروی جاذبه در کرهٔ ماه $\frac{۱}{۶}$ 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.

۲ــ اگر محیط[۱] یک مربع[۲] برابر ۱۸ سانتی متر باشد برنامه‌ای بنویسید که اطلاعات زیر را روی صفحه نشان دهد :

الف) اندازه هر ضلع[۳] مربع بر حسب سانتی متر

ب) اندازه مساحت[۴] مربع بر حسب سانتی متر مربع

۳ــ برنامه‌ای بنویسید که محیط یک دایره را سؤال کند و سپس اندازهٔ شعاع و اندازهٔ مساحت دایره را محاسبه نماید.

۴ــ سال ۱۳۰۰ شمسی را در نظر بگیرید. تعداد روزهای سپری شده از این سال تا امروز چقدر 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.

۵ــ یک فروشگاه پوشاک، 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.

۶ــ برای یک دستگاه خودپرداز برنامه‌ای بنویسید که یک عدد را به عنوان مبلغ یک اسکناس دریافت کند و آن را به اسکناس‌های کوچک‌تر خرد[۵] 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.)

۱ــ Perimeter

۲ــ Square

۳ــ Side

۴ــ Area

۵ــ Change

۷ــ برنامه‌ای بنویسید که مقلوب یک عدد صحیح دو رقمی دریافتی از کاربر را نمایش دهد.

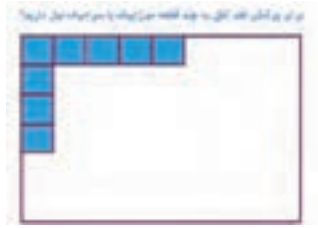
۸ــ برنامه‌ای بنویسید که تاریخ تولد هر فرد را بر حسب ماه، روز و سال دریافت نماید و سپس تعداد روزهای عمر وی را حساب کند. هر سال ۳۶۵ روز و هر ماه ۳۰ روز است.

۹ــ (این سؤال نیاز به برنامه‌نویسی ندارد و فقط یک محاسبه ساده است.)

می‌خواهیم کف یک اتاق ۳ ×۴ متری را سرامیک کنیم. اگر هر قطعه سرامیک به شکل مربع و به اندازهٔ ۵۰ سانتی متر باشد به چند قطعه سرامیک نیاز داریم؟

اگر اندازهٔ آن برابر ۴۰ سانتی متر باشد آن وقت به چه تعداد سرامیک نیاز داریم؟

۱۰ــ برنامه‌ای به نام Simple Tiling Calculator بنویسید که اندازه طول (Width) و عرض (Length) یک اتاق را بپرسد و سپس اندازهٔ یک سرامیک مربع شکل را سؤال نماید. تعداد سرامیک‌های لازم برای پوشش کف را محاسبه و تعیین کند.



۱۱ــ مدیر یک ساختمان مسکونی قصد دارد، هزینهٔ برق عمومی‌ساختمان، 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.

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

واژگان و اصطلاحات انگلیسی فصل پنجم

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Area	
۲	Arithmetic	
۳	Binary Operator	
۴	Change	
۵	Credit Hour	
۶	Decrement	
۷	Expression	
۸	Increment	
۹	Operand	
۱۰	Operator	
۱۱	Perimeter	
۱۲	Physical Education	
۱۳	Precedence	
۱۴	Side	
۱۵	Square	
۱۶	Strict type Checking language	
۱۷	Unary Operator	

۶ 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، تصمیم‌گیری کنیم. برنامه‌هایی که تاکنون نوشته‌ایم تمام دستورهای داخل متد ()Main، پشت سرهم و به نوبت اجرا می‌شدند. اکنون در این فصل با دستورهای شرطی آشنا می‌شویم که به وسیلهٔ 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 فراگیر بتواند:

۱ــ انواع عملگرهای مقایسه‌ای را توضیح دهد و آنها را در برنامه استفاده کند.

۲ــ ساختار دستور if و if_else را در برنامه‌های شرطی خود استفاده کند.

۳ــ مفهوم عملگرهای منطقی را شرح دهد وانواع آنها را نام ببرد.

۴ــ ازعملگرهای منطقی به طور صحیح در برنامه‌ها استفاده نماید.

۵ــ با استفاده از جدول عملگرهای منطقی، نتیجهٔ عبارتهای ترکیبی منطقی و مقایسه‌ای را به‌دست آورد.

۶ــ اولویت عملگرها را با استفاده از جدول تقدم عملگرها در حل عبارات ترکیبی به‌کاربندد.

۷ــ دستور switch را در برنامه‌های خود به کارگیرد.

۸ــ باترکیب دستورهای شرطی، برنامه‌هایی با ساختار پیچیده شرطی بنویسد.

۱ــ۶ــ عبارت منطقی یا بولین[۱]

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.

مثلاً عبارت 12 < 10 یک عبارت منطقی است، که نتیجهٔ آن false است چون 12 کوچک‌تر از 10 نیست. ولی در عبارت x > 0 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 ۱ــ۶، مشاهده می‌شود. این عملگرها شبیه عملگرهایی است که در ریاضیات استفاده می‌شود. معمولاً درعبارت‌های منطقی از عملگرهای مقایسه‌ای استفاده می‌شود.

۲ــ۶ــ عملگرهای مقایسه‌ای[۲]

همان طور که از نام این عملگرها مشخص است، برای مقایسهٔ داده‌ها استفاده می‌شوند و نتیجهٔ 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 کمی‌متفاوت 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 ۱ــ۶ هر یک از عملگرهای مقایسه‌ای به همراه علامت ریاضی آنها نشان داده شده است.

۱ــ Boolean Expression

۲ــComparison Operators

جدول ۱ــ۶ــ عملگرهای مقایسه‌ای

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

مثال ۱ــ۶ : در برنامه ۱ــ۶ 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' + 1); // True
        Console.ReadKey();
    }
}
```

برنامه ۱ــ۶ــ عبارت‌های منطقی

نکته

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

۳ــ۶ــ دستور شرطی[۱] if

در درس مبانی کامپیوتر، با دستور شرطی «اگر» آشنا شدید و به وسیلهٔ 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 شدهٔ if ، عبارت منطقی داخل پرانتز و دستوری که در صورت درست (true) بودنِ نتیجه عبارت، اجرا خواهد شد.

توجه داشته باشید که پس از کلمهٔ if، یک جفت پرانتز وجود دارد و عبارتی از نوع منطقی که برای بررسی و مقایسهٔ داده است، داخل پرانتز نوشته می‌شود. در خط بعدی، دستوری که می‌خواهیم در صورتِ درست بودنِ عبارت منطقی اجرا شود، با تورفتگی می‌نویسیم و در انتهای آن علامت ; را به منزله پایان دستور if قرار می‌دهیم.

در دستورات زیر نمونه‌ای از به کارگیری 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؟

۱ــ Conditional Statement

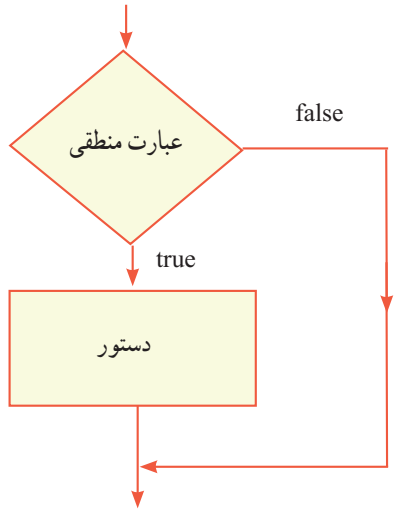
نکته

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 :



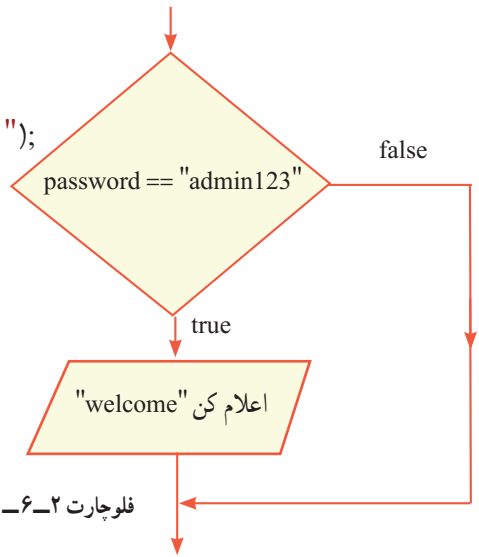
فلوچارت ۱ــ۶ــ دستور if

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 ۲ــ۶ : می‌خواهیم برنامه‌ای 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 می‌دهیم. اگر رمز وارد شده مساوی admin123 بود پیام Welcome را نمایش می‌دهیم. دستور شرطی این برنامه در زیر آورده شده است.

```
if (password == "admin123")
    Console.WriteLine("Welcome ");
```



فلوچارت ۲ــ۶ــ بررسی رمز ورود

```
class Login
  {
   static void Main()
   {
      string password;
      Console.Write("Enter password: ");
      password = Console.ReadLine();
      if (password == "admin123")
        Console.WriteLine("Welcome");
      Console.WriteLine("Press any key to continue...");
      Console.ReadKey();
   }
}
```

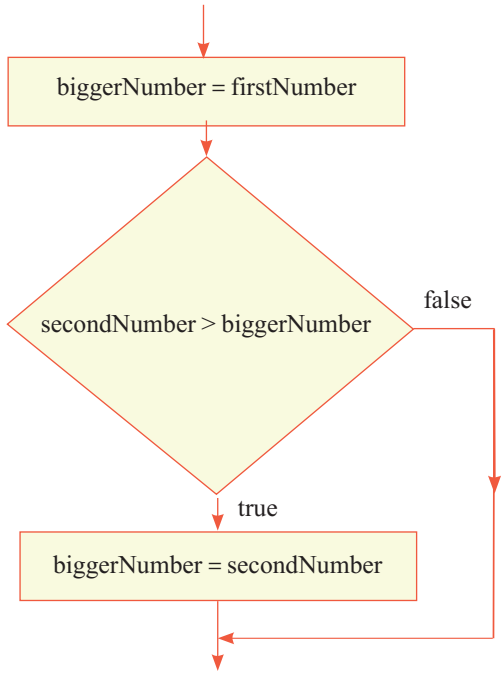
برنامه ۲ــ۶ــ بررسی رمز ورود

کار در کارگاه ۱

۱ــ برنامه ۲ــ۶ را در محیط VS بنویسید، ترجمه و اجرا نمایید و به ازای ورودی‌های مختلف برنامه را آزمایش کنید.

۲ــ **مثال ۳ــ۶** : می‌خواهیم برنامه‌ای بنویسیم که عدد بزرگ‌تر را از بین دو عدد دریافتیِ 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 عدد بزرگ‌تر فرض می‌کنیم و داخل یک متغیر مثلاً به نام biggerNumber قرار می‌دهیم. آن گاه مقدار این متغیر(فرض خود) را با عدد بعدی مقایسه می‌کنیم. 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 بزرگ‌تر در نظر بگیریم. به این ترتیب درمتغیر 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 ۳ــ۶ را مشاهده کنید.



فلوچارت ۳ــ۶ــ پیدا کردن عدد بزرگ‌تر از بین دو عدد

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

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

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

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

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

دریافت عدد اول

دریافت عدد دوم

برنامه ۳ــ۶ــ 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 می‌شوند.

سؤال: به نظر شما چرا دستور biggerNumber = secondNumber; از ابتدای خط شروع نشده است و به اندازهٔ 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 کمی‌داخل تر بنویسید تا برنامه خوانا و واضح باشد.

اگرچه رعایت نکردن این قاعده، تأثیری 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 ۴ــ۶ : عددی صحیح، داخل متغیری به نام number قرار دارد و می‌خواهیم زوج بودن آن را پس از تشخیص، با پیامی‌مناسب اعلام کنیم.

الگوریتم یا روش انجام کار : می‌دانیم عددی زوج است که رقم یکان آن یکی از اعداد ۰ و۲ و۴ 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 باقیماندهٔ تقسیم عدد بر ۲ را حساب می‌کنیم، اگر باقیمانده برابر صفر بود پیامی‌را روی صفحه نمایش می‌دهیم تا نشان دهد که عدد زوج است.

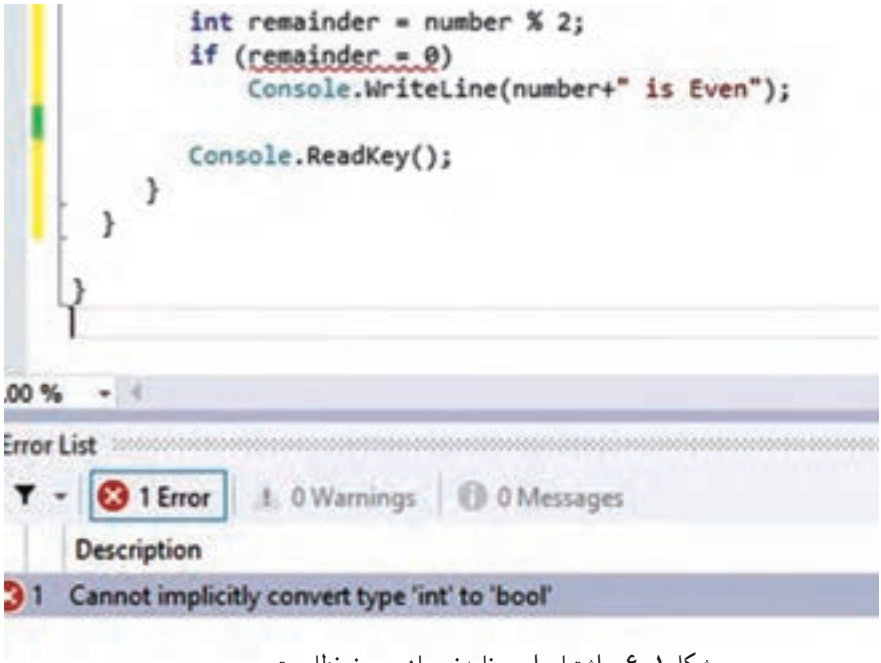
```
int remainder = number % 2;
if (remainder == 0)
    Console.WriteLine(number+" is Even.");
```

اگر در متغیر number، عددی زوج مانند ۱۶ باشد، حاصل باقیماندهٔ تقسیم آن عدد بر ۲، صفر خواهد شد و در این صورت نتیجهٔ عبارت منطقی در دستور if، مقدار true خواهد بود. بنابراین دستورِ 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 باقیماندهٔ تقسیم آن عدد بر ۲ عدد یک خواهد شد و در این صورت نتیجه عبارت منطقی در دستور if، مقدار false خواهد بود. بنابراین دستورِ نمایش پیام، اجرا نمی‌شود و چیزی روی صفحه نشان داده نمی‌شود.

نکته

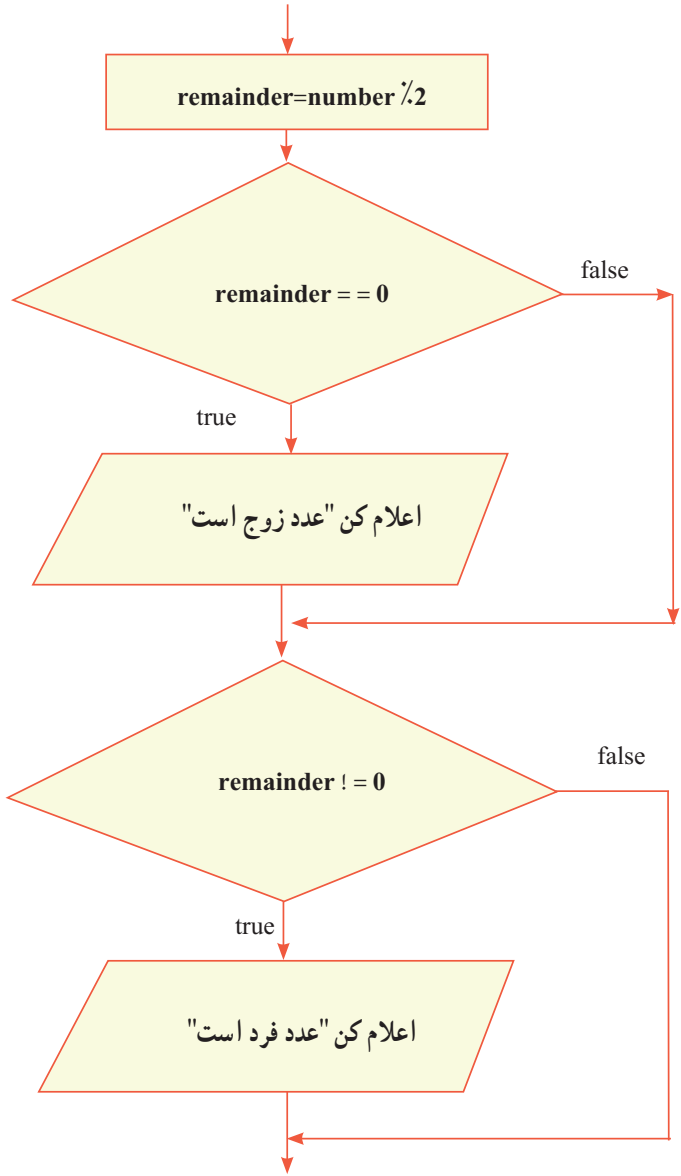
یکی از اشتباهات رایج برنامه نویسان در نوشتن عبارت منطقی دستور if، استفاده از علامت = به جای == است. در این صورت، مترجم متوجه بروز چنین اشتباهی می‌شود و علاوه بر کشیدن خط قرمز در زیر عبارت مورد نظر، خطایی را صادر می‌کند (شکل ۱ــ۶).



شکل ۱ــ۶ــ اشتباه رایج برنامه نویسان و بروز خطای مترجم

مثال ۵ــ۶ : اکنون می‌خواهیم دستورهای مربوط به مثال ۴ــ۶ 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 باقیماندهٔ تقسیم عدد بر ۲، برابر یک شود آن عدد فرد است. بنابراین پیام "**عدد فرد است**"نمایش داده می‌شود. از آنجا که حاصل باقیماندهٔ تقسیم هر عددی بر ۲ ، صفر یا یک است، بنابراین به جای شرط مساوی یک، از شرط نامساوی صفر استفاده می‌کنیم. فلوچارت ۴ــ۶ را مشاهده کنید.



فلوچارت ۴ــ۶ــ تشخیص زوج و فرد با دو دستور if

مطابق با فلوچارت ۴ـ۶، دستورهای لازم برای انجام کار، چنین خواهد بود :

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

۴ـ۶ـ دستور شرطی if_else

در مثال ۴ـ۶، برای اینکه عدد زوج یا فرد را تشخیص دهیم از باقیمانده تقسیم عدد بر ۲ استفاده کردیم. در فلوچارت شکل ۴ـ۶، تصمیم‌گیری بر اساس باقیماندهٔ تقسیم صورت می‌گیرد که عدد صفر یا مخالف صفر است. در چنین مواردی از دستور شرطی if_else می‌توان استفاده کرد. شکل کلی این دستور چنین است :

```
if (عبارت منطقی)
    دستور شماره ۱ ;
else
    دستور شماره ۲ ;
```

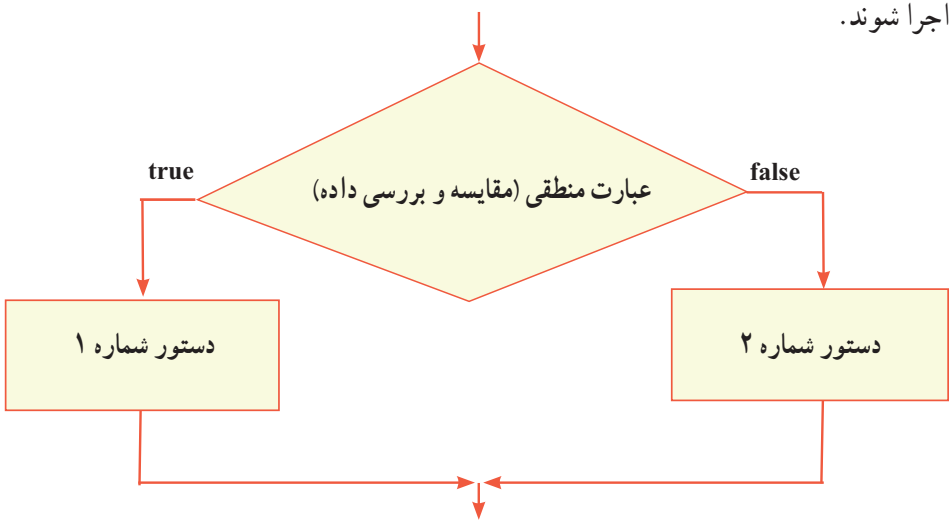
اگر نتیجه عبارت منطقیtrue باشد، دستور شماره ۱ که مربوط به قسمت if است اجرا می‌شود و در غیر این‌صورت، یعنی اگر نتیجه عبارت منطقیfalse باشد، دستور شماره ۲ که مربوط به قسمت else است،اجرا می‌شود.

نکته

دستور if-else را خوانا بنویسید یعنی :

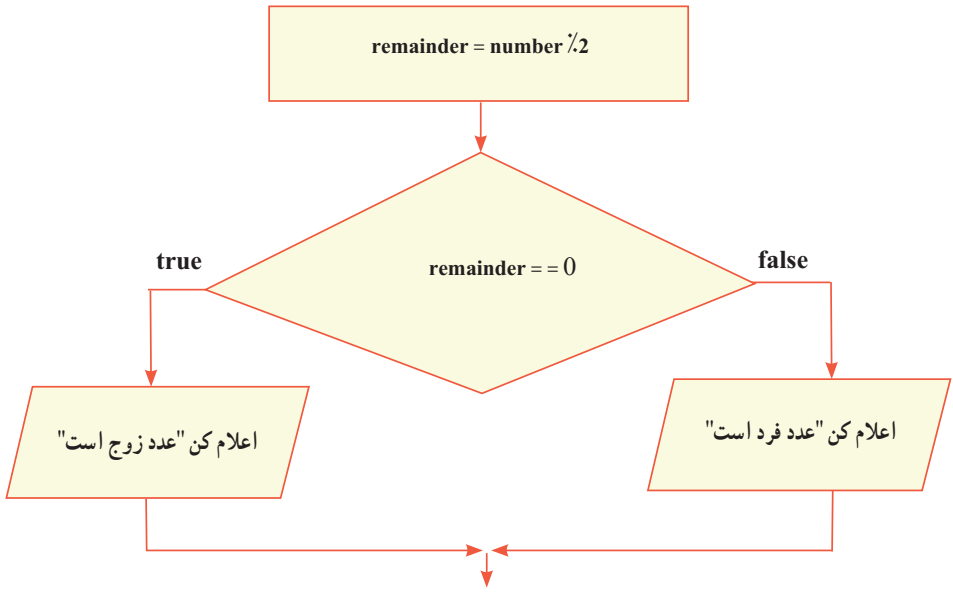
دستورهای شماره ۱ و ۲ که مربوط به قسمت if یا قسمت else است را با کمی تورفتگی بنویسید تا معلوم شود که هرکدام متعلق به چه قسمتی است. مراقب باشید که بعد از کلمه رزرو شده else، علامت نقطه ویرگول قرار ندهید.

دستور if_else به صورت فلوچارت ۵ـ۶ در زیر نشان داده شده است. توجه داشته باشید که همواره فقط یکی از دو دستور شماره ۱ یا شماره ۲ اجرا می‌شود و محال است که هر دو دستور با هم اجرا شوند.



فلوچارت ۵ـ۶ـ فلوچارت دستور if-else

مثال ۶ـ۶ : در این مثال، دستورهای تشخیص زوج یا فرد بودن عدد را با استفاده از دستور if_else، طبق فلوچارت ۶ـ۶ می‌نویسیم.



فلوچارت ۶ـ۶ـ تشخیص عدد زوج یا فرد

```
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 ۴ــ۶ به کار گرفته شد را با یکدیگر مقایسه کنید. کدام یک بهتر است؟

کار در کارگاه ۲

مثال ۷ــ۶ : می‌خواهیم برنامه‌ای 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 می‌شود. از حقوق تمام کارمندان، ۷ درصد به عنوان حق بیمه کسر می‌گردد. همچنین کارکنانی که حقوق آنها بیش از ده میلیون ریال باشد، مالیات به اندازه ۵ درصد کسر می‌گردد (مالیات به مازاد بر ده میلیون تعلق می‌گیرد).

الگوریتم یا روش انجام کار : فلوچارت ۷ــ۶ عملیات محاسبه حقوق را نشان می‌دهد.

با توجه به فلوچارت، ابتدا حقوق ثابت کارمند دریافت می‌شود. حق بیمه آن محاسبه و سپس حقوق با عدد ده میلیون ریال مقایسه می‌شود. چنانچه حقوق بیش از ده میلیون ریال باشد، آن‌گاه مقدار مالیات محاسبه و بر روی صفحه نشان داده می‌شود. در غیر این صورت مالیات صفر در نظر گرفته می‌شود. با کمی‌دقت به قسمت شرط این فلوچارت متوجه می‌شوید که در صورت برقراری شرط، دو دستور باید اجرا شود. در حالی‌که در مثال‌های قبلی تنها یک دستور اجرا می‌شد. برای مشخص کردن این دو دستور در برنامه، باید آنها را در بین علامت‌های آکولاد باز و بسته قرار دهیم. با این کار یک **بلاک**[1] می‌سازیم.

```
{
    دستور محاسبه مالیات
    دستور نمایش مقدار مالیات
}
```

بلاکی شامل دو دستور

۱ــ Block

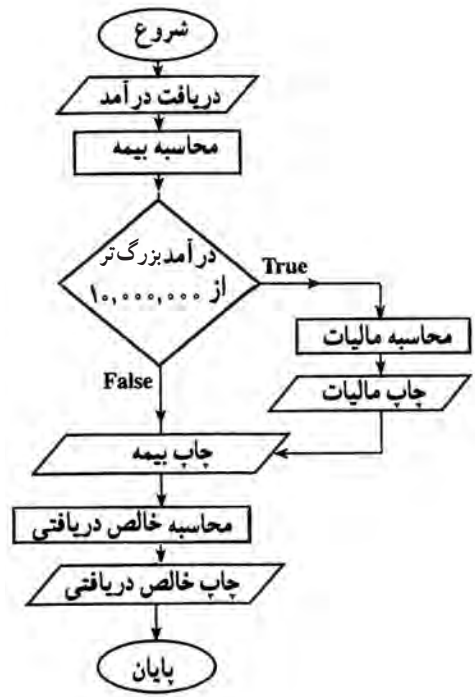
نکته

بلاک چیست؟

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

```
{
    دستور ;
    دستور ;
    دستور ;
}
```

- 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.



فلوچارت۷ــ۶ــ محاسبه حقوق دریافتی کارمندان یک شرکت

```
class Salary
{
    static void Main()
    {
        long income;
        double tax, insurance, net;
        string input;
        Console.Write("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 ۴ــ۶ــ محاسبه حقوق خالص کارمندان یک شرکت (مثال ۷ــ۶)

سؤال: در مثال ۷ــ۶، 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؟

مثال ۸ــ۶ : فرض کنید نمره پایانی درس برنامه‌سازی یک دانش آموز در متغیر mark قرار دارد. با بررسی قبولی یا مردودی، پیام مناسب چاپ کنید.

الگوریتم یا روش انجام کار : با توجه به اینکه نمره قبولی ۱۲ می‌باشد، در صورتی‌که نمره در محدودهٔ ۱۲ تا ۲۰ قرار داشته باشد دانش‌آموز قبول و در صورتی‌که نمره در محدودهٔ ۰ و کمتر از ۱۲ است دانش‌آموز مردود محسوب می‌شود.

به زبان ریاضی محدودهٔ ۱۲ تا ۲۰ به شکل زیر است :

$$۱۲ \leq mark \leq ۲۰$$

سؤال: مفهوم این عبارت ریاضی چیست و چه ورودی‌هایی برای متغیر mark در این محدوده قرار می‌گیرند؟

اگر عبارت ریاضی فوق را بشکنیم، به دو عبارت ساده زیر خواهیم رسید :

$mark \leq ۲۰$ و همچنین $۱۲ \leq mark$

و یا

$mark \leq ۲۰$ و همچنین $mark \geq ۱۲$

سؤال: اصطلاح «و همچنین» که ما آن را از این به بعد به‌طور ساده «و» می‌نامیم، چه مفهومی دارد؟ عبارت ترکیبی قبلی را برای mark = ۱۴ بررسی می‌کنیم :

با توجه به اینکه ۱۴ از ۱۲ بزرگ‌تر است، نتیجه عملگر مقایسه‌ای $mark \geq ۱۲$ مقدار true خواهد شد و همچنین با توجه به این‌که ۱۴ از ۲۰ کوچک‌تر است، نتیجه عملگر مقایسه‌ای $mark \leq ۲۰$ نیز مقدار true می‌باشد. بنابراین عبارت ترکیبی بالا مقدار true خواهد شد.

عبارت فوق در برنامه نویسی را یک «عبارت ترکیبی منطقی» می‌نامیم و در سی شارپ به شکل زیر می‌نویسیم :

```
(mark >= ۱۲) && (mark <= ۲۰)
```

سؤال: جدول Trace زیر را به ازای مقادیر داده شده تکمیل کنید.

mark	(mark >= ۱۲)	(mark <= ۲۰)	(mark >= ۱۲) && (mark <= ۲۰)
۲۲			
۱۰			

بنابراین دستور if مربوط به وضعیت قبولی دانش‌آموز به صورت زیر است :

```
if ((mark >= ۱۲) && (mark <= ۲۰))
Console.WriteLine("Passed");
```

سؤال: با توجه به دستور قبل، دستور مربوط به وضعیت مردودی را بنویسید.

۵ــ۶ــ عملگرهای منطقی[۱]

عملگر منطقی && یک عملگر دوتایی و دارای دو عملوند است. در هنگام اجرا، ابتدا نتیجه عملوند سمت چپ، به وسیلهٔ 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 و ارزش عبارت ترکیبی به دست می‌آید. علاوه بر عملگر &&، عملگرهای منطقی دیگری نیز در زبان C#، برای ایجاد عبارت‌های مرکب وجود دارد، که علامت و عملکرد آنها به ترتیب اولویت، در جدول ۲ــ۶ نشان داده شده است.

جدول ۲ــ۶ــ عملگرهای منطقی

اولویت	نام عملگر	علامت	عملکرد
۱	نقیض	!	ارزش عملوند را معکوس می‌کند.
۲	و	&&	تنها در صورتی که هر دو عملوندtrue 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[۲]	^	اگر ارزش عملوندها مخالف یکدیگر باشد، نتیجه این عملگر true خواهد بود.

۱ــ Logical operator

۱ــ Exclusive or (XOR)

جدول ۳ــ۶ــ حالت‌های مختلف در عملگرهای منطقی

نتیجه عملگر &&	نتیجه عملگر ^	نتیجه عملگر ||	ارزش عبارت سمت راست	ارزش عبارت سمت چپ
false	false	false	false	false
false	true	true	true	false
false	true	true	false	true
true	false	true	true	true

سؤال: در صورتی که هنرستان شما بخواهد به هنرجویان پایه دوم که معدل بالای ۱۸ دارند جایزه دهد چه عملگر منطقی و چه شرطی را می‌نویسید؟ جدول trace مشابه سؤال قبل تهیه کنید.

کار در کارگاه ۳

مثال ۹ــ۶ : 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 کرد. یک روش، بررسی بخش پذیری بر پنج است، مانند روشی که برای تشخیص اعداد زوج در مثال ۵ــ۶ 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.

```
class CheckNumbers
{
    static void Main()
    {
```

```
        string input;
        int number;

        Console.Write("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 ۵ــ۶ــ برنامه تشخیص عدد مضرب پنج (مثال ۹ــ۶)

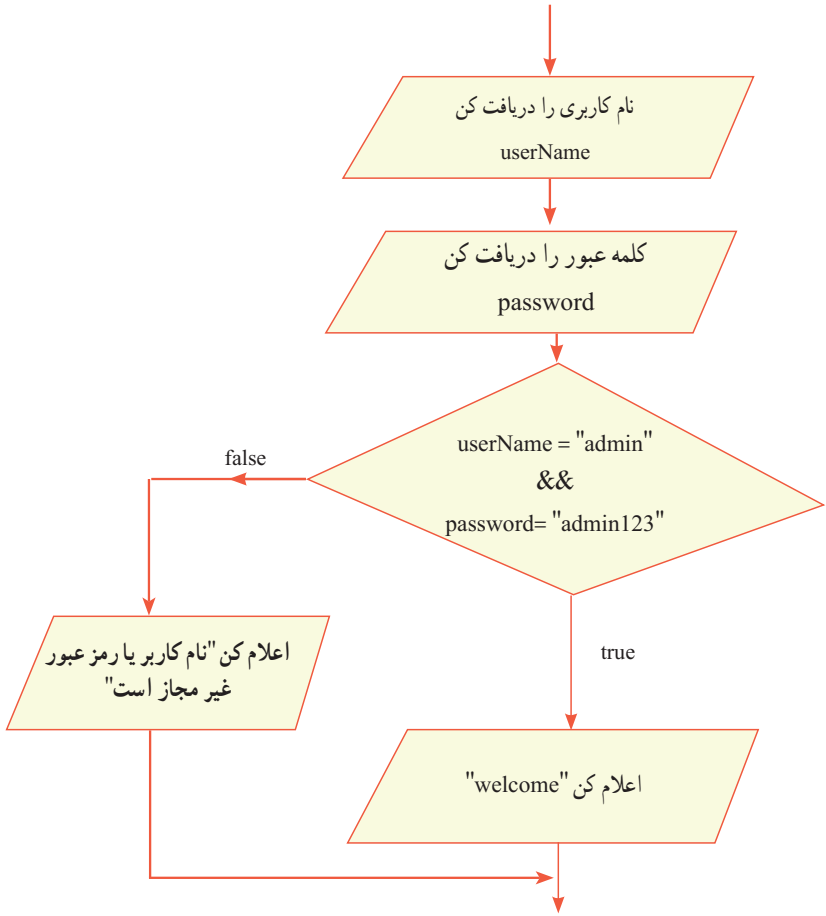
سؤال: خروجی برنامه به ازای اعداد ورودی ۹۰ و ۳۷ چیست؟ چه پیامی‌روی صفحه ظاهر می‌شود؟

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

مثال ۱۰ــ۶ : می‌خواهیم مثال ۳ــ۶ 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 خوش‌آمدگویی نمایش داده می‌شود.

در غیر این صورت، برنامه پیام می‌دهد که اطلاعات وارد شده صحیح نیست (فلوچارت ۸ــ۶).



فلوچارت ۸ــ۶ــ دریافت نام کاربری و کلمه عبور

در فلوچارت ۸ــ۶، فرض شده است که کاربرِ مجاز، دارای نام کاربری «admin» و کلمهٔ عبور «admin123» است. برای بررسی اطلاعات وارد شده از یک عبارت منطقی مرکب، در دستور if_else استفاده می‌کنیم. برنامه شماره ۶ــ۶ 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 ۶ــ۶ــ دریافت نام کاربری و کلمه عبور کاربر (مثال ۱۰ــ۶)

مثال ۱۱ــ۶ : می‌خواهیم برنامه‌ای بنویسیم که با توجه به عدد دریافتی از کاربر، نام یکی از فصل‌های سال را نمایش دهد (۱ : بهار، ۲ : تابستان، . . .)

الگوریتم و روش انجام کار : ابتدا عدد دریافتی را در متغیر number قرارمی‌دهیم. سپس با مقایسهٔ عدد دریافتی با شماره‌های ۱ تا ۴ نام فصل مربوطه را نمایش می‌دهیم.

برای نوشتن شرط‌های این برنامه، نیاز به مقایسه‌هایی داریم که شرط برنامه را پیچیده و به هم مرتبط می‌کنند.

۶ــ۶ ــ ساختار if_else پیچیده

در این ساختار هرگاه یکی از عبارت‌های منطقی if درست باشد، دستور مربوط به همان if اجرا شده و دیگر سایر شرط‌ها بررسی نمی‌شود و در نتیجه ساختار سریع‌تر اجرا می‌شود. زیرا نیازی ندارد به بررسی شرط‌های اضافی بپردازد که قطعاً نادرست هستند. در ساختار زیر در صورتی که عبارت منطقی۱ 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.

```
if (عبارت منطقی۱)
{
   دستورات شماره ۱ ;
 }
else if (عبارت منطقی۲)
{
    دستورات شماره ۲ ;
}
else if (عبارت منطقی۳)
{
    دستورات شماره ۳ ;
}
else
{
   دستورات شماره ۴ ;
{
```

در مثال ۱۱ــ۶ چنین ساختاری مورد استفاده قرار می‌گیرد.

```
class CheckSeason
{
    static void Main
    {
        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 ۷ــ۶ــ دریافت شماره فصل و نمایش نام فصل (مثال ۱۱ــ۶)

۷ــ۶ــ 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 مقدار۱:
        دستور۱;
        break;
    case مقدار۲:
        دستور۲;
        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، تصمیم‌گیری 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.

آکولادهای 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، رشته ای و انواع داده صحیح می‌تواند استفاده شود.

مثال ۱۲ــ۶ : اکنون می‌خواهیم مثال ۱۱ــ۶ را با استفاده از دستور switch بازنویسی نماییم.

```
class CheckSeason
{
    static void Main()
    {
        string input;
        Console.Write("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 ۸ــ۶ــ دریافت شماره فصل و نمایش نام فصل (مثال ۱۲ــ۶)

کار در کارگاه ۴

۱ــ مثال ۱۲ــ۶ را طوری بنویسید که به‌ازای دریافت شمارهٔ هر فصل تغییر رنگ زمینه صفحه نمایش را متناسب با فصل انجام دهد.

۲ــ مثال ۱۳ــ۶ : در اینجا می‌خواهیم مثال ۲ــ۶ را توسعه دهیم. تشخیص سه کاربر مختلف، با دریافت نام کاربری، 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.Write("Enter username: ");
        userName = Console.ReadLine();
        Console.Write("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");
            else
                Console.WriteLine("Wrong password!");
            break;
        default:
            Console.WriteLine("Invalid username!");
            break;
    }
    Console.WriteLine("Press any key to continue...");
    Console.ReadKey();
  }
}
```

برنامه ۹ــ۶ــ دریافت نام کاربری و رمز عبور کاربران (مثال ۱۳ــ۶)

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 اشتباهِ برنامه‌نویس، خطا می‌دهد[1]. برای هر case، می‌توان بیش از یک دستور نوشت و نیازی به بلاک ندارد.

۳ــ مثال ۱۴ــ۶ : 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.

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

```
Console.Write("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 ۱۰ــ۶ــ کاربرد دستور switch بدون default

سؤال: اگر در ورودی حروف بزرگ وارد کنیم، عملکرد Switch چگونه خواهد بود؟

گسترش برنامه : برنامه را طوری تغییر دهید که نسبت به دریافت حروف کوچک و بزرگ حساس نباشد.

برای این کار از متد زیر کمک بگیرید.

```
input.To Lower
```

۴ــ برنامه‌ای بنویسید که یک عدد را بگیرد و زوج یا فرد بودن آن را تشخیص دهد.

۵ــ برنامه‌ای بنویسید که نمرهٔ دانش‌آموز را دریافت کند و پیامی برای قبولی یا مردودی او صادر نماید.

۶ــ برنامه شماره ۱۱ــ۶ و ۱۲ــ۶ را نوشته و ترجمه کنید. به ازای ورودی‌های مختلف آن را اجرا کنید.

۷ــ اگر کاربر در هنگام اجرای مثال ۱۴ــ۶، کلمه دیگری به غیر از کلمات تعیین شده بنویسد، خروجی چه خواهد بود؟

۸ــ می‌خواهیم اگر کاربر کلمه دیگری غیر از کلمات تعیین شده در مثال ۱۳ــ۶، را وارد کرد، دستور زیر اجرا گردد، چه تغییری در داخل دستور switch انجام می‌دهید؟

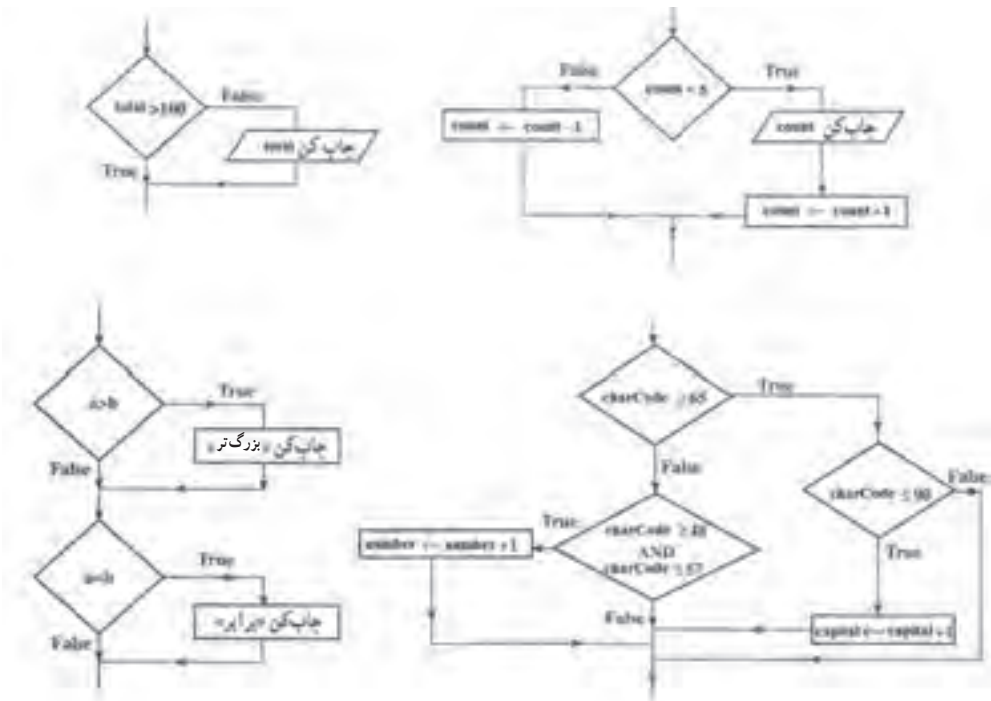
```
Console.WriteLine("I'm sorry, I don't understand that!");
```

خودآزمایی فصل ششم

۱ــ 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!");
```

۲ــ دستورهای شرطی مطابق با فلوچارت‌های زیر بنویسید :



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

```
if (number < 100)
    number++;
```

۴ــ در هر یک از دستورهای زیر، با استفاده از عملگرهای منطقی، تعداد دستورهای if را کاهش 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("Stating...");
```

۵ــ 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؟

۶ــ 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

۱ــ برنامه‌ای بنویسید که عددی را به عنوان شمارهٔ فصل از ورودی دریافت کند. نام ماه‌های آن فصل را در خروجی چاپ نماید. برای ورودی غیر مجاز، پیام مناسب نشان داده شود.

۲ــ 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 ۱۵۰ باشد.

۳ــ برنامه‌ای بنویسید که عددی را از ورودی دریافت کند و تعیین کند که این عدد مضرب ۶ است یا خیر؟ از روش بخش‌پذیری هم زمان بر ۲و۳ استفاده نمایید.

۴ــ برنامه‌ای بنویسید که عددی را از ورودی دریافت کند و زوج بودن آن را تشخیص دهد. برای بخش پذیری عدد ورودی بر ۲، از روش بررسی رقم یکان شامل ۰، ۲، ۴، ۶، ۸ استفاده کنید.

۵ــ برنامه‌ای بنویسید که یک عدد یک رقمی دریافت کند و کلمه متناظر با آن عدد را روی صفحه نمایش دهد مثلاً اگر عدد ۵ از ورودی دریافت شد، برنامه کلمه FIVE را نمایش دهد.

۶ــ برنامه‌ای بنویسید که میزان مصرف برق در یک ماه را بر حسب کیلو وات سؤال 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 ۱۸۰ کیلو وات در ماه باشد در این صورت ۱۰۰ کیلو وات آن با مبلغ ۱۳۶۴ ریال محاسبه شده و ۸۰ کیلو وات باقیمانده آن، با مبلغ ۱۴۲۶ ریال محاسبه می‌شود :

بهای برق مصرفی = (۱۰۰×۱۳۶۴) + (۸۰ × ۱۴۲۶)

پله‌های مصرف ۳۰ روزه (kwh)	نرخ (ریال)
مصرف ۰ تا ۱۰۰	۱۳۶۴
مازاد بر ۱۰۰ تا ۲۰۰	۱۴۲۶
مازاد بر ۲۰۰ تا ۳۰۰	۱۴۸۸
مازاد بر ۳۰۰ تا ۴۰۰	۱۵۵۰
مازاد بر ۴۰۰ تا ۵۰۰	۱۷۳۶
مازاد بر ۵۰۰ تا ۶۰۰	۱۹۸۴
مازاد بر ۶۰۰	۲۲۳۲

۷ــ برنامه‌ای بنویسید که سه عدد از ورودی دریافت کند و عدد بزرگ‌تر را تشخیص داده و آن را نمایش دهد.

۸ــ 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.

۹ــ برنامه‌ای بنویسید که دمای هوا را از ورودی دریافت نماید. بر طبق جدول زیر، پیام مناسب را چاپ نماید.

ورودی	خروجی
دما < ۰	بسیار سرد
۰ < دما ≤ ۱۰	سرد
۱۰ < دما ≤ ۲۰	معتدل
۲۰ < دما ≤ ۳۰	گرم
دما > ۳۰	بسیارگرم

۱۰ــ برنامه‌ای بنویسید که شماره رنگ را از ورودی دریافت نماید. طبق جدول زیر، رنگ زمینهٔ صفحه نمایش را به رنگ مربوطه تغییردهد.

ورودی(شماره رنگ)	رنگ صفحه نمایش
۰	مشکی
۱	آبی
۲	قرمز
۳	سبز

۱۱ــ 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، نمرهٔ وی و درصد پاسخ‌گویی به سؤالات را محاسبه نماید. (با این فرض که هر پاسخ غلط $\frac{۱}{۳}$ نمرهٔ منفی دارد).
مثلاً اگر دانش‌آموز ۱۷ پاسخ درست و ۶ پاسخ غلط داشته باشد، نمرهٔ وی برابر ۱۵ می‌باشد (از ۳۰ نمره) و درصد پاسخ‌گویی وی، برابر ۵۰٪ است.

۱۲ــ آیا برنامهٔ 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 سؤالات آزمون باشد؟

۱۳ــ با تکمیل تمرین ۱۱ از بروز چنین خطاهایی در ورودی جلوگیری کنید.

واژگان و اصطلاحات انگلیسی فصل ششم

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Block	
۲	Boolean Expression	
۳	Comparison Operators	
۴	Conditional Statement	
۵	Exclusive or (XOR)	
۶	Logical Operator	

۷ 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. به‌وسیلهٔ 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 فراگیر بتواند:

- کاربرد حلقه در برنامه را توضیح دهد.
- دستورات ایجاد حلقه را نام ببرد و تفاوت هر یک را بیان کند.
- عملکرد و کاربرد دستور حلقه while را توضیح دهد.
- عملکرد و کاربرد دستور حلقه for را توضیح دهد.
- برنامه‌های کاربردی را با حلقه تکرار بنویسد.
- در برنامه‌های خود break و Continue را در صورت لزوم به‌کار بندد.
- ضرورت استفاده از حلقه‌های متداخل را توضیح دهد و آنها را در برنامه‌های خود به‌کار بندد.

۱ــ۷ــ 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 این‌صورت، 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 while یا do_while استفاده می‌کنیم. اگر تعداد دفعات تکرار مشخص باشد، مثلاً حداکثر ۳ بار نام کاربری و کلمه عبور دریافت گردد، از دستور حلقه معین for استفاده می‌شود.

۱ــ۱ــ۷ ــ دستور حلقه شرطی while : ساختار کلی دستور while، در زیر نشان داده شده است :

```
while (عبارت منطقی)
    دستور ;
```

دستور while از سه بخش تشکیل شده است :

۱ــ کلمه رزرو شده while

۲ــ عبارت منطقی در داخل پرانتز

۳ــ دستوری که در صورت درست بودنِ نتیجه عبارت، اجرا خواهد شد.

مثال ۱ــ۷ : نمونه‌ای از به‌کار گیری دستور while چنین است :

```
int x=1;
while (x < 100)
    Console.WriteLine("x=" + x++);
```

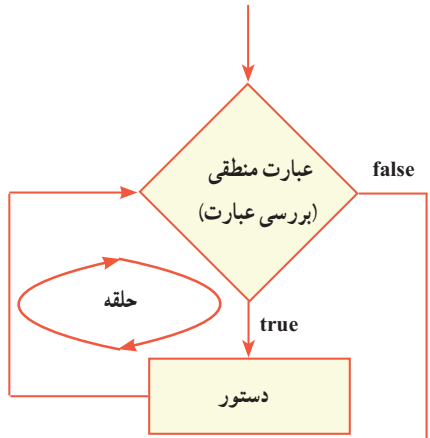
قطعه برنامه ۱ــ۷ــ مثالی از یک حلقه

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.

```
while (عبارت منطقی)      نقطه ویرگول ندارد
    دستور ;
```



فلوچارت ۱ــ۷ــ 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، دستورمذکور، اجرا خواهد شد. در این حالت می‌گوییم **حلقه**[۱] ایجاد شده است (فلوچارت ۱ــ۷).

دستور یا دستوراتی که مکرر اجرا می‌گردند در بدنه حلقه[۲] 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 به‌صورت یک بلاک بنویسید. یعنی آنها را در داخل علامت‌های آکولاد باز و بسته قرار دهید.

کار در کارگاه ۱

۱ــ مثال ۱ــ۷ را به صورت یک برنامه کامل در VS تایپ و اجرا نمایید.

۲ــ اعداد حلقه را طوری تغییر دهید تا خروجی اعداد دو رقمی شود.

۳ــ اعداد حلقه را طوری تغییر دهید تا خروجی اعداد سه رقمی و به صورت نزولی شود.

۱ــ Loop ۲ــ Loop Body

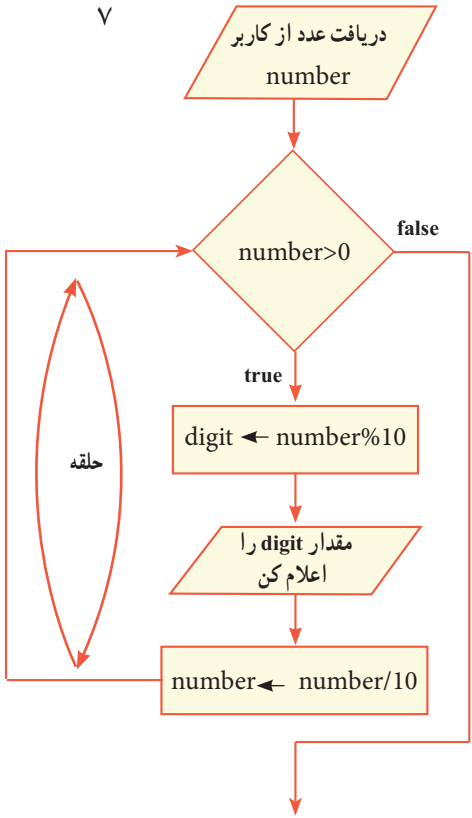
مثال ۲ــ۷ : می‌خواهیم برنامه‌ای بنویسیم که ارقام یک عدد را جدا نموده و آنها را نمایش دهد.

الگوریتم یا روش انجام کار : با توجه به آن که رقم یکان هر عدد، باقیمانده تقسیم آن عدد صحیح بر ۱۰ است، کافی است عدد دریافتی را بر ۱۰ تقسیم و باقیمانده آن را نمایش دهیم. به عنوان مثال اگر عدد دریافتی ۵۷۶ باشد باقیمانده تقسیم آن بر عدد ۱۰، عدد ۶ است که رقم یکان عدد است.

$$\begin{array}{r|l} ۵۷۶ & ۱۰ \\ \hline ۵۷۰ & ۵۷ \\ \hline ۶ & \end{array}$$

اگر دوباره خارج قسمت بدست آمده یعنی ۵۷ را بر عدد ۱۰ تقسیم کنیم، خواهیم داشت :

$$\begin{array}{r|l} ۵۷ & ۱۰ \\ \hline ۵۰ & ۵ \\ \hline ۷ & \end{array}$$



فلوچارت ۲ــ۷ــ جدا کردن ارقام عدد

اگر به باقیمانده تقسیم بالا توجه کنید، متوجه می‌شوید که عدد ۷، رقم ده گان عدد دریافتی است. به همین ترتیب ادامه می‌دهیم و عمل تقسیم را تکرار کرده و باقیمانده تقسیم را به‌دست می‌آوریم.

$$\begin{array}{r|l} ۵ & ۱۰ \\ \hline ۰ & ۰ \\ \hline ۵ & \end{array}$$

باقیمانده تقسیم بالا را در نظر بگیرید. در اینجا توانستیم آخرین رقم عدد یعنی ۵ را نیز جدا کنیم.

با دقت در عملیات فوق متوجه می‌شویم که عمل تقسیم، عملی تکراری است و تا زمانی انجام می‌شود که مقسوم آن بزرگتر از صفر باشد (فلوچارت ۲ــ۷).

مطابق با فلوچارت شکل ۲ــ۷، برنامه را می‌نویسیم :

```
class Numbers
{
    static void Main()
    {
        int number, digit ;
        string input;
        Console.Write("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 ۲ــ۷ــ جدا کردن ارقام یک عدد صحیح

سؤال: در برنامه ۲ــ۷، یک بلاک شامل سه دستور، در داخل حلقه قرار دارد که تکرار می‌شود. بلاک و دستورهای داخل آن را مشخص کنید.

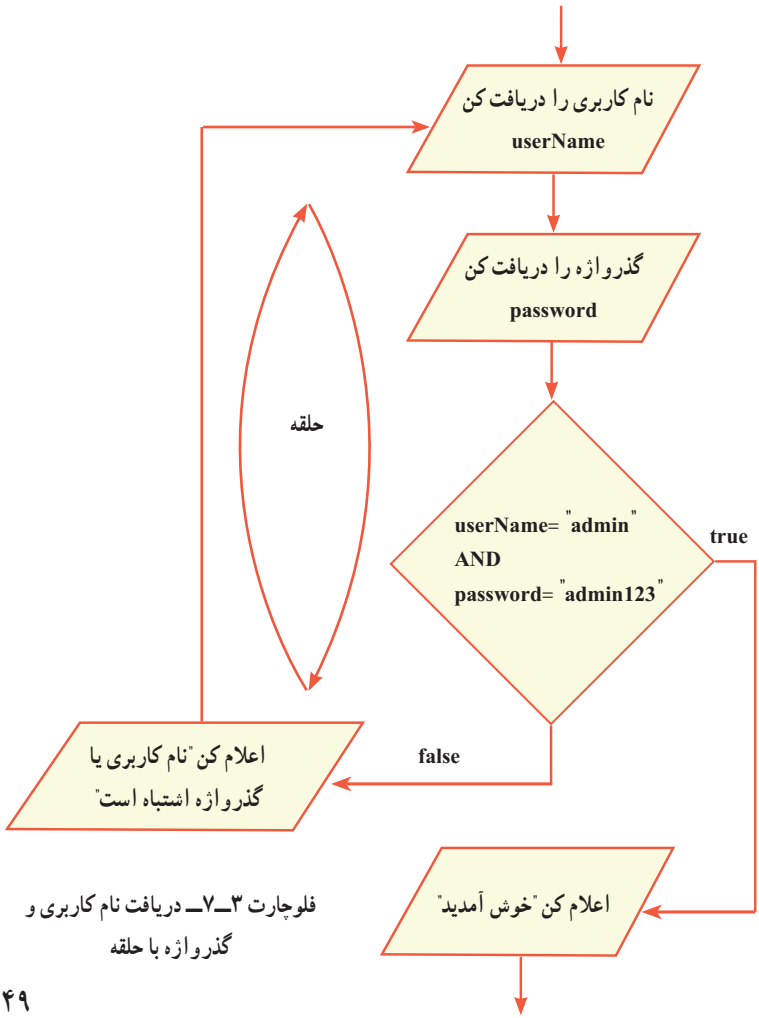
سؤال: برنامه را برای عدد ۳۸۵ در جدول، Trace کنید.

input	number	digit	خروجی
385			

سؤال: آیا ممکن است که دستورات داخل حلقه، اصلاً اجرا نشوند؟ به ازای چه مقادیری این اتفاق می‌افتد؟

مثال ۳ــ۷ : 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 ۱۰ــ۶، با روش دریافت و بررسی نام کاربری و گذرواژه آشنا شدید. در این مثال از حلقه برای تکرار عملیات استفاده می‌کنیم. در صورتی که کاربر اطلاعات را به‌طور صحیح وارد نکرد، باید دوباره عمل دریافت و بررسی اطلاعات تکرار شود. فلوچارت ۳ــ۷، حلقه و عملیات تکراری را نشان می‌دهد.



فلوچارت ۳ــ۷ــ دریافت نام کاربری و گذرواژه با حلقه

اگر فلوچارت ۳ــ۷ 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.

۲ــ۱ــ۷ دستور حلقه شرطی do_while : شکل کلی دستور do_while به‌صورت زیر است :

```
do
    دستور;
while (عبارت منطقی) ;
```

دستور do_while از چهار بخش تشکیل شده است :

۱ــ کلمه رزرو شده do

۲ــ دستور داخل حلقه

۳ــ کلمه رزرو شده while

۴ــ عبارت منطقی داخل پرانتز، که در صورت درست بودن آن، دستور داخل حلقه تکرار می‌شود.

مثال ۴ــ۷ : نمونه‌ای از به‌کارگیری دستور do_while چنین است :

```
int x=1;
do
    Console.WriteLine("x=" + x++);
while (x < 10);
```

قطعه برنامه ۳ــ۷ــ مثالی از یک حلقه

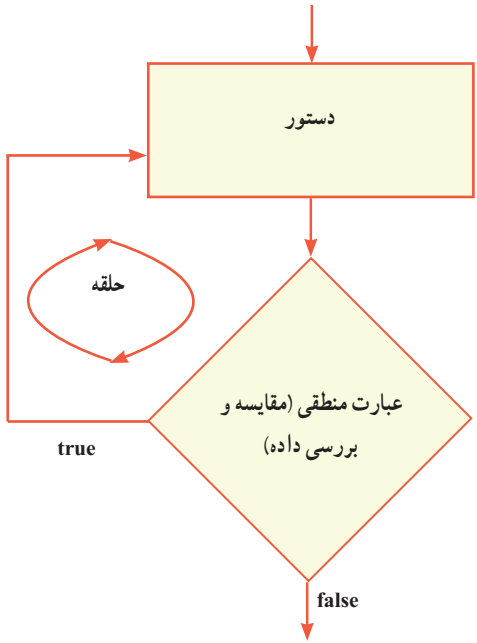
سؤال: به نظر شما خروجی این حلقه چیست؟ جدول Trace حلقه را ترسیم کنید.

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. تا زمانی که حاصل عبارت true است حلقه تکرار می‌شود. اگر حاصل ارزیابی عبارت false شود، دیگر به کلمه do برنمی‌گردد و کنترل برنامه به خط بعد از while واگذار شده و دستورات دیگرِ برنامه اجرا می‌شود. فلوچارت ۴ـ۷، دستور do_while را نشان می‌دهد.



فلوچارت ۴ـ۷ـ دستور do_while

کار در کارگاه ۲

برنامه مربوط به فلوچارت ۳ـ۷ (دریافت نام کاربری و گذرواژه)، را با استفاده از حلقه do_while بنویسید.

```
using System;
class LoginLoop
{
    static void Main(string[] args)
    {
        string userName, password;
        bool loginFlag;
        do
        {
            Console.Write("Enter username: ");
            userName = Console.ReadLine();
            Console.Write("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();
    }
}
```

حلقه do...while

برنامه ۴ـ۷ـ دریافت نام کاربری و رمز عبور در داخل حلقه

در برنامه ۴ــ۷، بیش از یک دستور در داخل حلقه قرار دارد، بنابراین ازعلامت‌های 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 گذرواژه 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 گذرواژه 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 ۵ــ۷ : 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 :

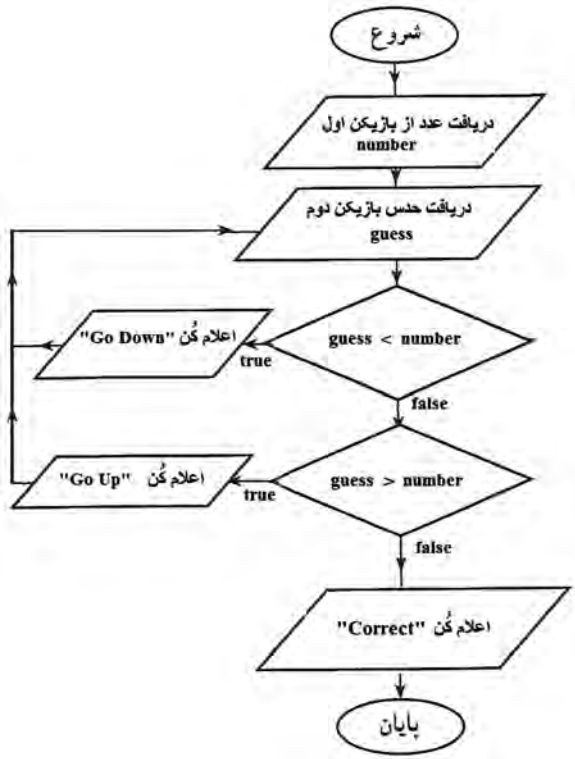
حالت اول : guess = number است که در این صورت پیام «آفرین درست حدس زدید» اعلان شود.

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 نزده 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 وسیلهٔ بازیکن دوم انجام می‌شود، به نظر شما از چه دستور حلقه باید استفاده کنیم؟

فلوچارت ۵ــ۷، لازم برای انجام این بازی را نشان می‌دهد. آن را به ازای ورودی‌های مختلف دنبال کنید.



فلوچارت ۵ــ۷ــ بازی حدس عدد

بر اساس فلوچارت فوق، برنامه ۵ــ۷ را می‌نویسیم.

```
class GuessTheNumber
{
    static void Main(string[] args)
    {
        string input;
        int number, guess;
        Console.Write("Player 1: Think a number (1-100): ");
        input = Console.ReadLine();
        number = int.Parse(input);
        Console.Clear(); // clear console screen
        do
        {
            Console.Write("Player 2: Guess the number (1-100): ");
            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();
    }
}
```

حلقه do...while

برنامه ۵ــ۷ــ بازی حدس عدد

در برنامه ۵ــ۷، 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 ۵ــ۷ نیز، می‌توان امکاناتی را اضافه کرد. به‌عنوان 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 ۵ــ۷ اضافه می‌کنیم.

۲ــ۷ــ 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 ۶ــ۷ : 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 ۶ــ۷ــ استفاده از حلقه for برای نمایش اعداد طبیعی ۱ تا ۱۰

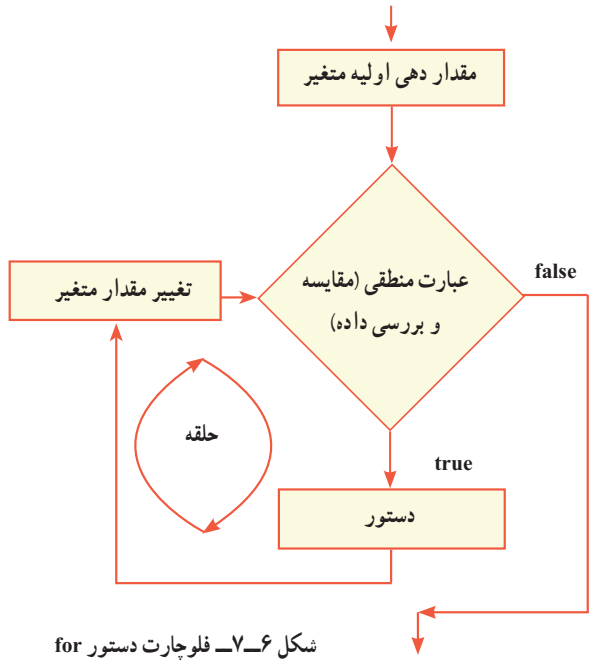
دستور ()WriteLine در این مثال در داخل حلقه قرار دارد که در هر بار تکرار حلقه، مقدار متغیر i روی صفحه نمایش، نشان داده می‌شود. مقدار متغیر i چقدر است؟

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 اولیهٔ ۱ تعریف شده است. قسمت دوم، یک عبارت منطقی (i <= 10) است و قسمت سوم نیز یک دستور انتساب افزایشی (i++) است.

جزییات اجرای دستورات بالا چنین است : در دستور for، ابتدا عدد ۱ در متغیر i قرار می‌گیرد و سپس عبارت منطقی محاسبه می‌شود و چون (i<=10) 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 (۲<=۱۰) در نتیجه دستور داخل حلقه اجرا شده و عدد ۲ بر روی صفحه نشان داده می‌شود. این عملیات تکرار می‌شود تا زمانی‌که نتیجه عبارت نادرست شود یعنی (۱۱<=۱۰) که در این‌صورت حلقه قطع می‌شود. بنابراین اعداد ۱ تا ۱۰ روی صفحه نمایش، نشان داده می‌شود.

شکل ۶ــ۷، 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.



شکل ۶ــ۷ــ فلوچارت دستور for

با توجه به مثال، شکل کلی دستور for چنین است :

for (تغییر مقدار متغیر ; عبارت منطقی ; مقدار اولیه = نام متغیر)
; 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[1] معروف می‌باشد. چون نقشِ شمارش تعداد تکرار حلقه را به عهده دارد. شمارش می‌تواند به‌صورت صعودی یا نزولی انجام شود. بدیهی است در حالت نزولی، تغییر مقدار متغیر باید به‌صورت کاهشی باشد.

شکل‌های دیگر حلقه مثال ۶ــ۷ 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 و تفاوت آن را با دستور قبل بیان نمایید.

مثال ۷ــ۷ : 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 ۷ــ۷ــ استفاده از حلقه for کاهشی

سؤال: به نظر شما خروجی این حلقه چیست؟

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.

for (تغییر مقدار متغیر ; عبارت منطقی ; مقدار اولیه = نام متغیر)
; دستور ←

۱ــ Counter

۱ــ۲ــ۷ــ کاربرد break در ساختار for : جدول Trace مربوط به این حلقه را رسم نمایید.

مثال ۸ــ۷ : می‌خواهیم برنامه‌ای برای ATM[1] بنویسیم که سه بار اجازه ورود گذرواژه را به کاربر بدهد و در صورتی که هر سه بار گذرواژه اشتباه وارد شود، کارت توسط دستگاه عابربانک با صدور پیغامی ضبط می‌شود.

الگوریتم یا روش انجام کار : دستگاه عابر بانک از شما سه بار گذرواژه ورود می‌گیرد. در صورتی که مقدار ورودی صحیح باشد، شما با پیغام خوش‌آمدگویی مواجه می‌شوید و با دستور break از حلقه خارج می‌شوید و در صورتی که مقدار ورودی درست نباشد، تا ۳ بار اجازه وارد کردن دارید وگرنه کارت با پیغامی ضبط می‌شود. در این برنامه برای دریافتِ ۳ بار گذرواژه، از حلقه for استفاده شده است. هر بار که گذرواژه از ورودی دریافت می‌شود، با گذرواژه اصلی مقایسه می‌شود و در صورت صحیح بودن علامتی به نام متغیر flag مقدار true می‌گیرد. این متغیر در آغاز برنامه مقدار false دارد و تا زمانی که گذرواژه درست وارد نشود همچنان false می‌ماند.

---

۱ ــ automatic teller machine

```
using System;
class ATMGetPass
{
        static void Main ()
        {
                int i;
                string mainPassword = "1404", password;
                bool flag = false;
                        for (i = 3; i >=1; i--)
                        {
                                Console.Write("Enter Password: ");
                                password = Console.ReadLine();
                                Console.WriteLine();
                                if (password == mainpassword)
                                {
                                        flag = true;
                                        Console.BackgroundColor = ConsoleColor.DarkGreen;
                                        Console.WriteLine(" That is right...");
                                        Console.Beep();
                                        break;   // خروج از حلقه
                                }
                                else
                                        Console.WriteLine(" PLZ try again...");
        }
        if (flag = false)
        {
                Console.BackgroundColor = ConsoleColor.Red;
                Console.WriteLine(" Your card will no longer work!!!");
        }
    }
}
```

برنامه ۸ــ ۷ ــ دریافت گذرواژه برای ATM

سؤال: چرا ترجیح می‌دهیم در این برنامه از متغیر رشته‌ای حاوی رشته‌ایِ از ارقام، برای دریافت گذرواژه استفاده کنیم؟

سؤال: به‌طور کلی چه زمانی از متغیرهای عددی و چه موقع از متغیر رشته‌ای رقمی در برنامه کمک می‌گیریم؟

گسترش برنامه : متغیر گذرواژه اصلی را به جای عدد، با حروف مقداردهی نمایید و برنامه را طوری تغییر دهید که نسبت به دریافت حروف کوچک و بزرگ حساس نباشد. برای این کار، از متدهای ToUpper() و ToLower() در شرط استفاده کنید.

سؤال: عملکرد متدهای ToUpper() و ToLower() را توضیح دهید.

نکته

توجه داشته باشید، که هیچ کدام از سه قسمت داخل پرانتز، در دستور for اجباری نیستند. حتی دستور for، می‌تواند به‌صورت زیر نوشته شود که در این صورت، یک حلقه تمام نشدنی و بی‌نهایت ایجاد می‌شود.

```
for ( ; ; )
 Console.WriteLine("Infinite Loop! ");
```

۲ــ۲ــ۷ــ کاربرد دستور continue در ساختار for :

مثال ۹ــ۷ــ همه شما بازی هپ را می‌شناسید و بارها برای تمرین جدول ضرب در دوران ابتدایی این بازی را انجام داده‌اید. در این برنامه می‌خواهیم بازی هپ را برای عدد ۵ شبیه‌سازی کنیم.

الگوریتم یا روش انجام کار : در بازی هپ، هرجا که به مضرب عدد تعیین شده می‌رسیم باید پیغام هپ را چاپ کنیم. برای این کار، حلقه‌ای را تا مثلاً ۵۰ در نظر می‌گیریم تا هرجا به مضرب ۵ رسید، هپ با رنگ دیگری چاپ شود. برای رسیدن به مضرب ۵ از شرطی با عملگر % برای باقیمانده کمک می‌گیریم. در نگاه اول برنامه با آنچه تاکنون آموخته‌اید قابل نوشتن است.

```
using System;
class hop
{
        static void Main (string [ ] args)
        {
                for (int i = 1; i<=20; i++)
                {
                        if (i ٪ 5 ==0)
                        {
                                Console.ForegroundColor = ConsoleColor.Yellow;
                                Console.WriteLine("hop");
                                Console.ForegroundColor = ConsoleColor.Gray;
                        }
                        Console.Write(" {0}" , i);
                }
                Console.ReadKey();
        }
}
```

برنامه ۹ــ ۷ــ بازی هپ

سؤال: اشکال برنامه فوق چیست؟

کار در کارگاه ۳

۱ــ قطعه برنامه ۳ــ۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.

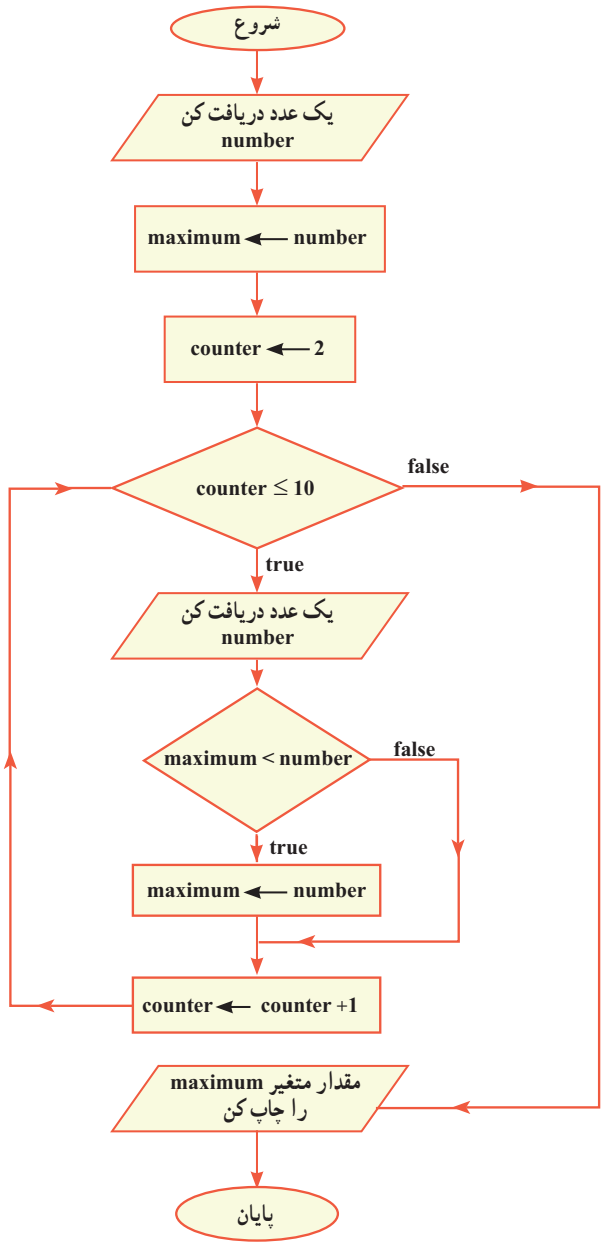
۲ــ به برنامه ۵ــ۷، امکان امتیازدهی را اضافه نمایید. برای این منظور، با استفاده از یک متغیر، تعداد دفعات تلاش بازیکن دوم را شمارش کرده و در انتهای برنامه، محتوای آن را نشان دهید. یک دستور انتساب افزایشی در داخل حلقه می‌تواند تعداد دفعات را شمارش کند.

۳ــ قطعه برنامه‌های ۶ــ۷ و ۷ــ۷ را به یک برنامه کامل تبدیل کرده و آن را اجرا کنید.

در مثال ۹ــ۷ نباید هم مضرب عدد ۵ و هم هپ هر دو چاپ شوند. در واقع به جای مضرب باید هپ چاپ شود. بنابراین باید برنامه را در رسیدن به مضرب ۵ متوقف کنیم و از گام بعدی حلقه، تکرار را از سر بگیریم. در واقع زمانی از دستور پرش continue استفاده می‌کنیم که خواهان متوقف شدن یکی از مراحل تکرار و شروع بلافاصله مرحله بعدی تکرار هستیم. دستور continue مرحله فعلی را متوقف و مرحله بعدی تکرار را شروع می‌کند. در برنامه هپ کافیست دستور ;continue را قبل از بسته شدن بلاک if قرار دهید.

مثال ۱۰ــ۷ : 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 ۳ــ۶، توضیح داده شد، استفاده می‌کنیم. شکل ۷ــ۷، 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.



شکل ۷ــ۷ ــ فلوچارت پیدا کردن بزرگ‌ترین عدد

برنامه ۱۰ــ۷ بر اساس فلوچارت شکل ۷ــ۷ نوشته شده است.

```
class FindMaximum
{
    static void Main(string[] args)
    {
        string input;
        int number, maximum;
        Console.Write("Enter a number: ");
        input = Console.ReadLine();
        maximum = int.Parse(input); // Suppose first number is maximum.

        for (int i=2; i<=10 ; i++)
        {
            Console.Write("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 اولین عدد

دریافت دومین عدد تا دهمین در حلقه

برنامه ۱۰ــ۷ــ پیدا کردن بزرگ‌ترین عدد از بین ده عدد

کار در کارگاه ۴

۱ــ با تغییراتی در برنامه ۱۰ــ۷، کوچکترین عدد را پیدا کنید. نام متغیر minimum را برای کوچک‌ترین عدد استفاده کنید.

۲ــ برنامه ۱۰ــ۷ را طوری بنویسید که هم بزرگترین و هم کوچک‌ترین عدد را از بین ده عدد پیدا کند.

۳ــ به برنامهٔ قبلی دستوری اضافه کنید تا فاصله بین بزرگ‌ترین عدد و کوچک‌ترین عدد را پیدا کند.

۳ــ۷ــ حلقه‌های متداخل

درون حلقه ممکن است دستور یا دستوراتی وجود داشته باشند. دستور یا دستورات داخل حلقه به تعداد تکرار حلقه، انجام می‌شود. بنابراین اگر حلقه‌ای مثلاً ۵ بار تکرار شود، دستورات داخل آن ۵ بار انجام می‌شوند.

دستور داخل حلقه ممکن است خودش، یک حلقه باشد. بنابراین حلقه‌هایی تودرتو خواهیم داشت که حلقهٔ داخلی به تعدادِ دفعاتِ تکرارِ حلقهٔ بیرونی، تکرار می‌شود.

سؤال: مثال‌های زیر را ببینید و تعداد تکرار هرکدام را مشخص کنید.

```
for (int i = 1;  i<=10;  i++)
Console.WriteLine(i);
```

```
for (int i = 1; i<=5; i++)
{
        input = Console.ReadLine();
        num = int.Parse(input);
        sum = sum + num;
}
```

```
for (int i = 5; i>=1; i--)
        for (int j = 1; j<=3; j++)
```

```
for (int i = 1; i<=2; i++)
{
        j=4;
        do
        {
                Console.WriteLine("{1},{1}", i , j);
                j-=1;
        } while (j >=1);
}
```

سؤال: تفاوت میان تعداد تکرار و دستورات درون حلقه را در مثال‌های (۱) تا (۴) با همکلاسان خود به بحث بگذارید.

جدول Trace مثال شماره (۳) به عنوان نمونه آورده شده است. این جدول را تکمیل کنید.

I	J	خروجی
۵	۱	
	۲	
	۳	
۴	۱	
	۲	
	۳	

اکنون فرض کنید می‌خواهیم برای یک دونده در مسابقه دو شبیه سازی انجام دهیم. در این مسابقه هر دونده باید دور تا دور ۳ زمینِ مسابقه بدود که البته هر زمین را نیز باید ۵ بار به طور کامل، دور بزند.

سؤال: پیشنهاد می‌کنید کدام‌یک از حلقه‌های قبلی را با اندکی تغییر برای این مسابقه به کار ببریم؟

سؤال: در زندگی روزمره با چه مثال‌هایی از حلقه‌های تودرتو یا متداخل روبرو هستید؟

مثال ۱۱ــ۷ــ در این برنامه قصد داریم یک ساعت دیجیتال را با ساعت و دقیقه شبیه سازی کنیم.

الگوریتم یا روش انجام کار : ابتدا بخش نمایش دقیقه را کدنویسی می‌کنیم. برای نمایش دقیقه نیاز به حلقه‌ای داریم که ۶۰ بار کار کند (چرا؟). این بخش با حلقهٔ بسیار ساده قابل کدنویسی است.

```
for (int  min = 0;  min<60;  min++)
        Console.WriteLine("{0}", min);
```

با اجرای کد بالا دقیقه‌های یک ساعت شبیه سازی می‌شود. در صورتی که بخواهیم ۱۲ ساعت را شبیه سازی کنیم کافیست دستورات بالا را ۱۲ بار تکرار کنیم. بنابراین دستورات را درون حلقه‌ای می‌نویسیم که ۱۲ بار تکرار می‌شود.

```
for (int  hour = 1;  hour <= 12;  hour++)
        for (int  min = 0;  min<60;  min++)
                Console.WriteLine("{0}", min);
```

با تغییراتی در دستور WriteLine می‌توانیم ساعت را هم نمایش دهیم. بنابراین برنامه به شکل زیر خواهد بود :

```
using System;
class clock
{
static void Main()
{
for (int  hour = 1;  hour <= 12;  hour++)
        for (int  min = 0;  min<60;  min++)
                Console.WriteLine("{1}: {1}" , hour , min);
 }
}
```

برنامه ۱۱ــ۷ــ نمایش ساعت دیجیتال با استفاده از حلقه‌های متداخل

سؤال: برنامه را طوری تغییر دهید که قبل از تغییر ساعت شمار، ایست موقتی با دریافت کلید از کاربر داشته باشد.

خودآزمایی فصل هفتم

۱ــ درستی یا نادرستی هر عبارت را تعیین کنید.

الف) در حلقهٔ while دستورات، حداقل یک‌بار تکرار می‌شوند.

ب) عبارت منطقی حلقه do در جلوی آن نوشته می‌شود.

ج) ما می‌توانیم حلقه for با تکرار ۵ را با while هم بازنویسی کنیم.

د) برای خروج از حلقه دستور break به ما کمک می‌کند.

هـ) اگر در انتهای for علامت ; قرار دهیم خطا رخ می‌دهد.

۲ــ جاهای خالی را با عبارات مناسب تکمیل کنید.

الف) معمولاً برای تکرار معین از حلقه ..... کمک می‌گیریم.

ب) برای این‌که حلقه for به صورت بی نهایت تکرار شود، به شکل ..... استفاده می‌کنیم.

ج) در حلقهٔ do عبارت منطقی در جلوی ..... نوشته می‌شود.

د) تعداد تکرار حلقه‌های متداخل از ..... تکرار هر حلقه به دست می‌آید.

۳ــ حلقهٔ زیر را با استفاده از for بنویسید.

```
int i = 11;
do
{
        Console.WriteLine(i);
{ while (i < = 99)
        i = 11+;
```

۴ــ حلقهٔ سؤال قبل را trace کنید و در یک جمله بنویسید چه کاری انجام می‌دهد؟

۵ــ 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;
}
```

۶ــ خروجی این قطعه برنامه را به دست آورید.

```
for (int i = 1; i<=3; i++)
{
        for (int j = 1; j<=3; j++)
                if(i==j)
                        Console.Write(1);
                else
                        Console.Write(0);
        Console.WriteLine( );
}
```

تمرینات برنامه‌نویسی فصل هفتم

۱ــ برنامه‌ای بنویسید که اعداد زوج از ۳ تا ۲۱ را به صورت نزولی نشان دهد. برنامه را طوری تغییر دهید که خروجی برنامه روی عدد ۱۶ متوقف شود.

۲ــ برنامه‌ای بنویسید که یک عدد صحیح را دریافت کند و سپس اعداد فرد از ۱ تا آن عدد را چاپ نماید.

۳ــ 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.

۴ــ 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

۵ــ 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.

۶ــ برنامه‌ای بنویسید که نمرات ۱۰ دانش‌آموز را که هر کدام ۵ درس دارند دریافت و مجموع و معدل هر دانش‌آموز را محاسبه و چاپ نماید.

۷ــ برنامه‌ای بنویسید که جدول ضرب ۱۰×۱۰ را روی صفحه نمایش دهد.

واژگان و اصطلاحات انگلیسی فصل هفتم

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Counter	
۲	Loop	
۳	Loop Body	
۴	ATM	
۵	Flag	

پیوست ها

پیوست ۱: نصب Visual Studio

شرکت مایکروسافت نرم‌افزاری به نام Visual Studio ساخته است که برای برنامه‌نویسی استفاده می‌شود. این نرم‌افزار شامل مترجم زبان برنامه‌نویسی، یک ویرایشگر[۱] و یک اشکال‌یاب می‌باشد. این نرم‌افزار در چند نسخه با ویژگی‌های مختلف عرضه شده است. همچنین یک نسخه رایگان به نام Visual Studio Express Edition عرضه کرده است که برای شروع کار برنامه‌نویسی مناسب است. دیگر نرم‌افزار رایگان، نرم‌افزار Microsoft . Net Framework است که یک دسته ابزار مختلف به‌صورت فرمان (که در خط فرمان باید تایپ شوند) فراهم می‌کند که برای ترجمه و اجرای برنامه نوشته شده به زبان C# می‌تواند مورد استفاده قرار گیرد. بنابراین شما برای برنامه‌نویسی و اجرای برنامه‌ها دو راه دارید :

۱ــ استفاده از Visual Studio نیاز به تهیه و نصب نرم‌افزار VS دارد که در ادامه این ضمیمه طریقه نصب آن بیان شده است.

۲ــ استفاده از ابزارهای خط فرمان : نیاز به نصب NET. دارد که در پیوست ۲ طریقه نصب آن را خواهید دید.

۱ــ Editor

پیوست ۲: نصب Net Framework

برای نصب Net Framework. مراحل زیر را دنبال کنید :

۱ــ از طریق موتور جست‌وجوگری نظیر گوگل، برنامه Net Framework. را جست‌وجو کرده و وارد سایت شرکت مایکروسافت شوید یا به آدرس زیر مراجعه کنید :

http: //www.microsoft.com/net

۲ــ در صفحه مربوط به دانلود برنامه Net Framework. روی نسخه NET. موردنظر جهت دانلود کلیک کنید.

پیوست ۳

مشکلات احتمالی که ممکن است در ترجمه یا اجرای برنامه پیش آید :

الف) اشکال در ترجمه برنامه

۱ــ اگر در هنگام ترجمه برنامه با پیام خطای زیر روبه‌رو 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.)

۲ــ اگر علی‌رغم نصب برنامه NET. همچنان خطای پیدا نکردن مترجم csc رخ می‌دهد، حتماً در مسیر جستجو (Path)، فولدر حاوی برنامه csc معرفی نشده است. فولدر حاوی متوجم معمولاً در مسیر زیر قرار دارد :

C: / Windows / Microsoft.NET/ Framework/ v4.0.30319

البته بسته به نسخه NET. شماره‌های v4.0.30319 متفاوت می‌باشد. این شماره مربوط به نسخه NET4.5 است.

برای اضافه کردن فولدر مربوطه به مسیر جست‌وجو، عملیات زیر را انجام دهید :

اگر سیستم عامل کامپیوتر شما ویندوز ۷ یا ویستا به بعد است :

۱ــ در میزکار روی آیکن Computer کلیک راست کنید و سپس Properties را انتخاب کنید.

۲ــ در سمت چپ صفحه روی گزینه Advanced System Setting کلیک کنید :

۳ــ در پنجره System Properties روی گزینه Environment Variales کلیک کنید.

۴ــ با کلیک برروی متغیر Path آن را انتخاب کرده و سپس روی کلید Edit کلیک کنید.

۵ــ در پنجره‌ای که ظاهر می‌گردد مسیر برنامه‌های مورد جست‌وجو نوشته شده است مسیر NET. را اضافه کنید.

ب) اشکال در اجرای برنامه

اگر در هنگام ترجمه برنامه با اشکال زیر روبه‌رو شدید ممکن است یکی از دو اشتباه زیر را انجام داده باشید :

۱ــ نام فایل را درست تایپ نکرده‌اید.

۲ــ پسوند فایل برنامه را در بعد از نام فایل ننوشته‌اید.

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	