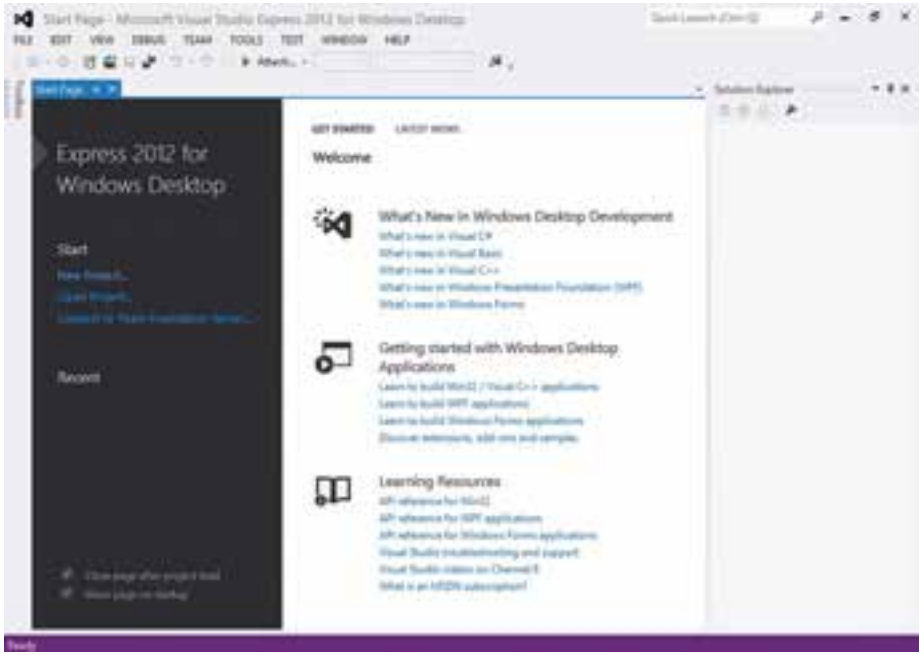
ما در این کتاب با Visual Studio Express 2012 کار می کنیم و از این به بعد در سرتاسر کتاب از مخفف VS برای بیان کلمه ویژوال استودیو استفاده می‌کنیم. اگر نرم افزار VS 2012 را در اختیار ندارید می‌توانید از نسخه‌های قدیمی‌تر نیز استفاده کنید فقط شکل ظاهری آن ممکن است کمی‌متفاوت باشد.

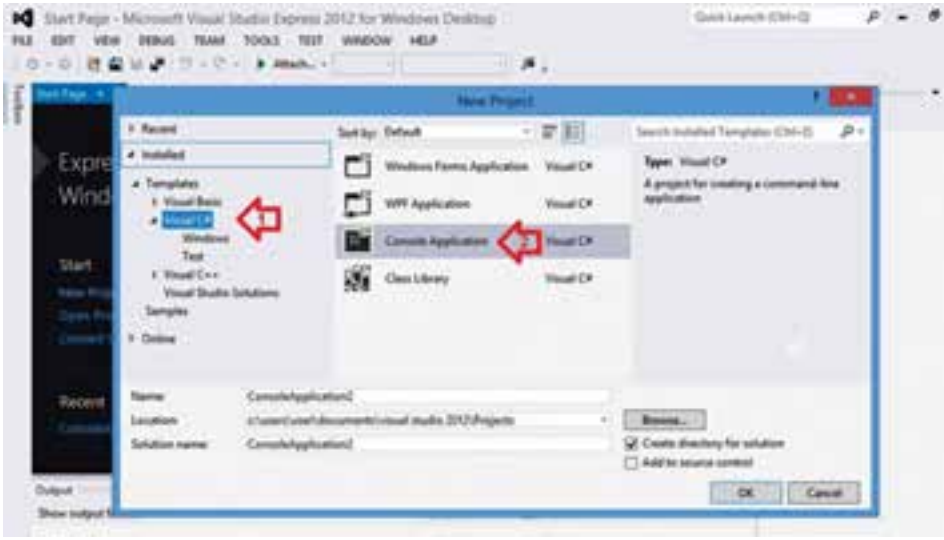


شکل ۲ــ۳ــ پنجره آغازین ویژوال استودیو اکسپرس ۲۰۱۲

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.

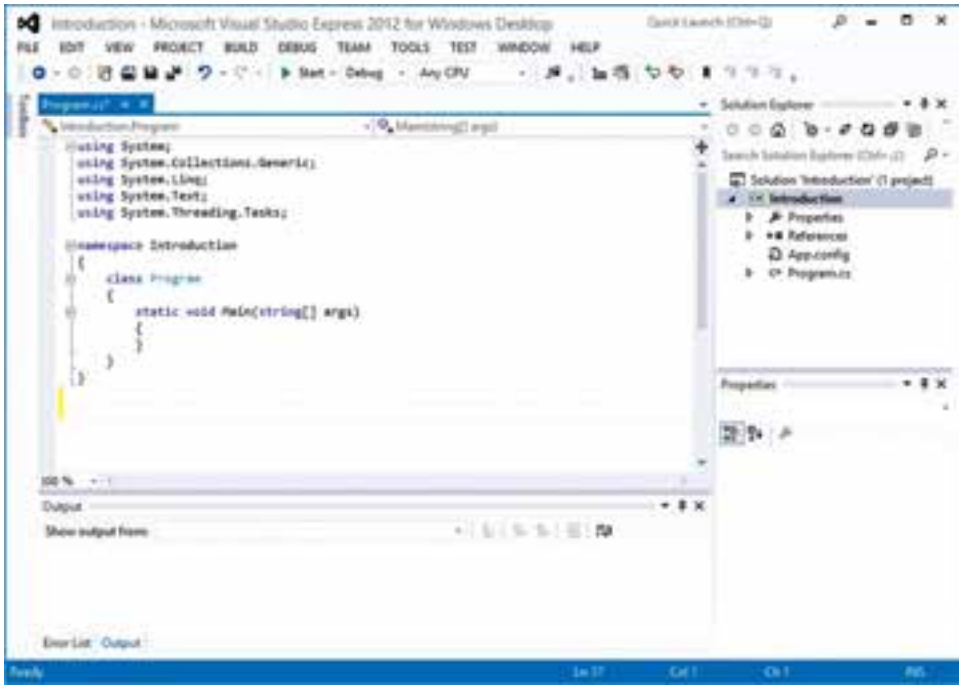
همان‌طور که درشکل ۴ــ۳ مشاهده می‌کنید، در قسمت Location مسیر ذخیره پروژه نشان داده شده است. هنگامی که VS را نصب می‌کنید در داخل My Documents یک پوشه به نام Visual Studio ساخته می‌شود که در داخل آن نیز یک پوشه فرعی به نام Projects ایجاد می‌شود. در داخل این پوشه، پروژه‌های شما به‌طور پیش فرض ذخیره می‌شود. 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

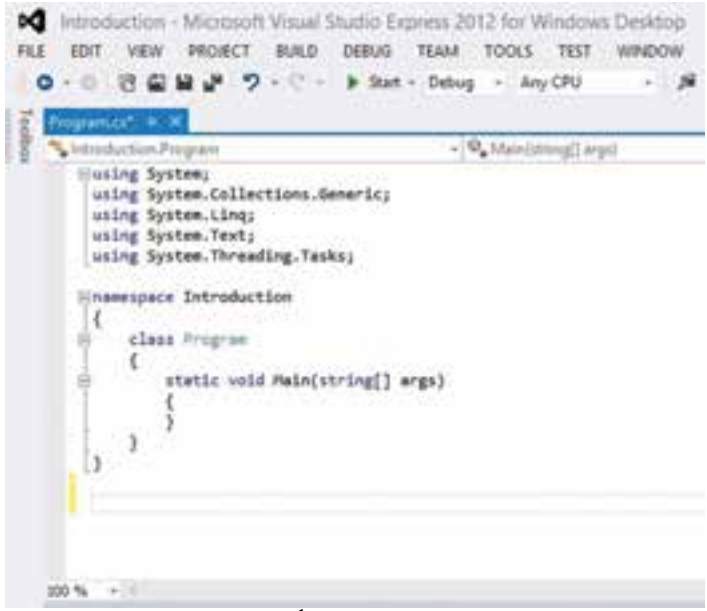
قبل از این که به برنامه‌نویسی بپردازیم ابتدا به صورت مختصر با بخش‌های مختلف محیط IDE آشنا می شویم.

۳ــ۳ــ معرفی بخش‌های اصلی ویژوال استودیو

۱ــ۳ــ۳ــ نوار منو و نوار ابزار[۱] : مانند بیشتر نرم افزارها، 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 وِیژوال استودیو

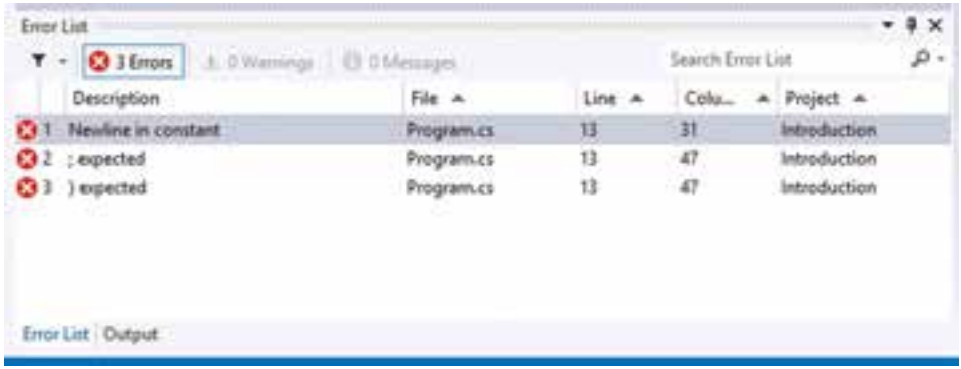
۲ــ۳ــ۳ــ پنجره ویرایشگر برنامه : در شکل ۷ــ۳ پنجره ویرایشگر برنامه نشان داده شده است. در این پنجره متن برنامه نگهداری می‌شود و می‌توانید چندین برنامه را هم زمان به صورت باز داشته باشید. در برنامه‌نویسی، این پنجره کاربرد زیاد دارد.



شکل ۷ــ۳ــ پنجره ویرایشگر برنامه

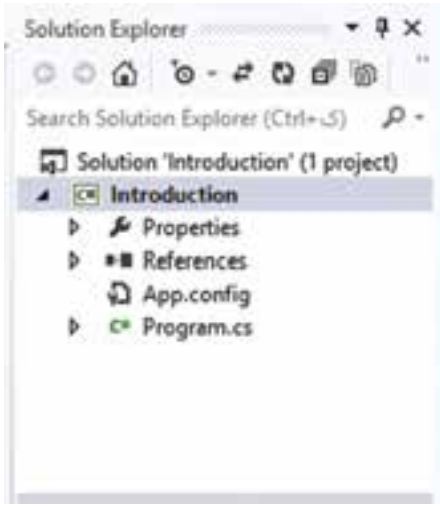
۱ــToo Bar

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 ۸ــ۳ لیست می‌شوند. پس از ترجمهٔ برنامه باید به این پنجره نگاه کنیم تا خطاهای احتمالی برنامه را متوجه شویم.



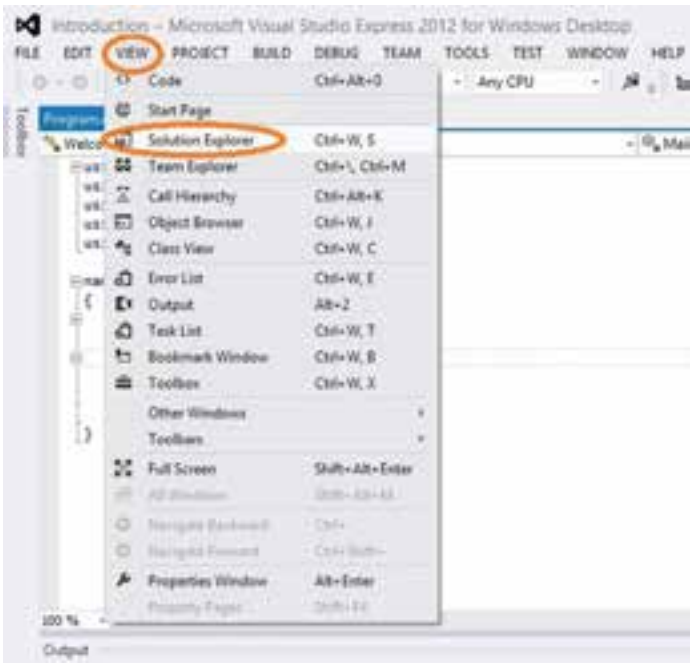
شکل ۸ــ۳ــ پنجره لیست خطاها

۴ــ۳ــ۳ــ پنجره (Solution Explorer) : سمت راست صفحه، پنجره ای قرار دارد که ساختار پروژه و تمام فایل‌های موجود در آن را نشان می‌دهد. اگر پروژه‌ای باز نباشد، محتوای این پنجره خالی است. ما نام این پنجره را مرورگر پروژه می‌نامیم و به وسیلهٔ آن به تمام اجزای پروژه دسترسی داریم.



شکل ۹ــ۳ــ پنجره Solution Explorer

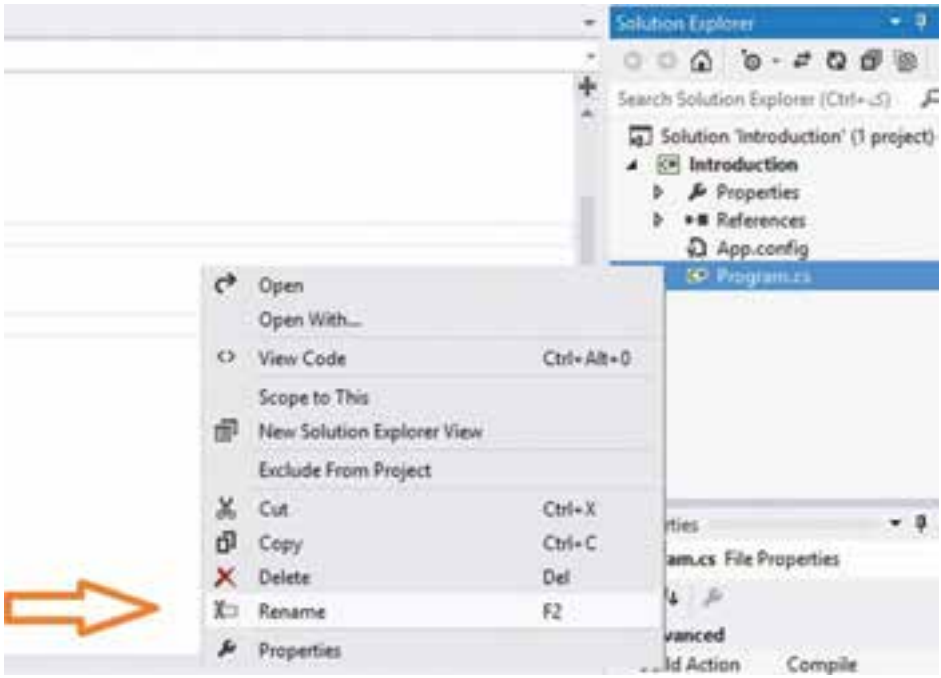
اگر پنجرهٔ مرورگر پروژه باز نیست، و آن را مشاهده نمی‌کنید از منوی View در بالای صفحه استفاده کنید. در این منو، نام تمام پنجره‌ها در شکل ۱۰ــ۳ مشاهده می‌شود، روی گزینه Solution Explorer کلیک کنید تا این پنجره دیده شود.



شکل ۱۰ــ۳ــ ظاهر کردن پنجره Solution Explorer

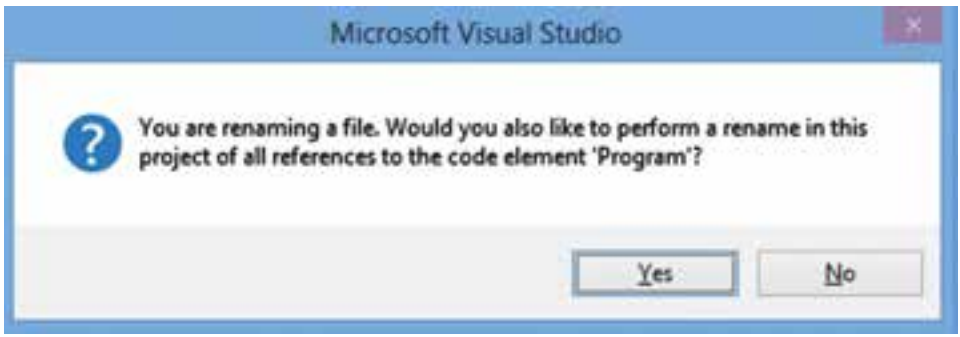
۴ــ۳ــ برنامه‌نویسی در محیط ویژوال استودیو

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 خوشامدگویی مانند فصل قبل نمایش دهد بنابراین برای تغییر نام فایل Program.cs روی آن کلیک راست کرده و نام دلخواه خود مثلاً Welcome.cs را وارد می‌کنیم (شکل ۱۱ــ۳).



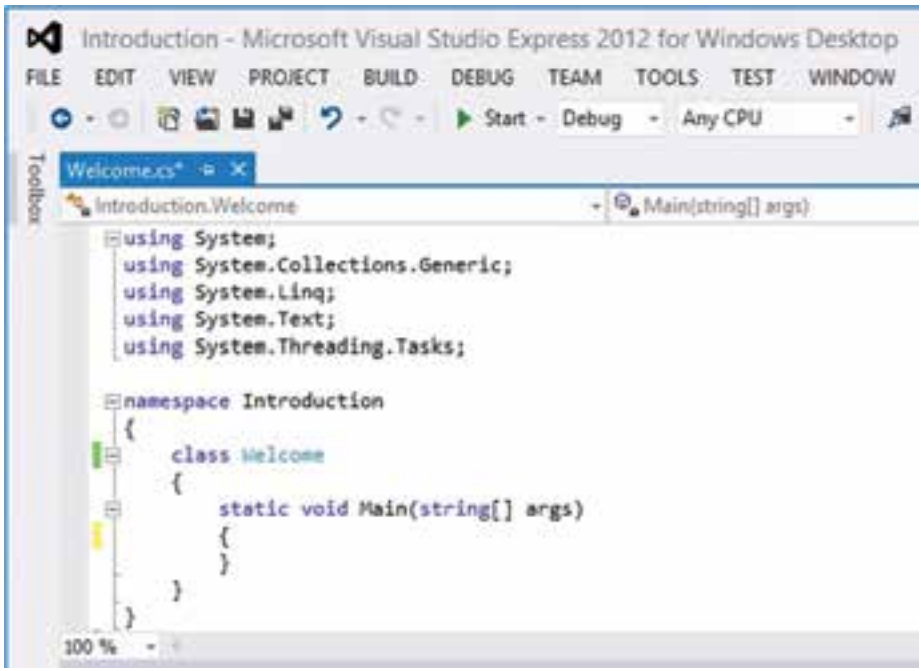
شکل ۱۱ــ۳ــ تغییر نام برنامه

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 قبلا 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 ا ست.



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.

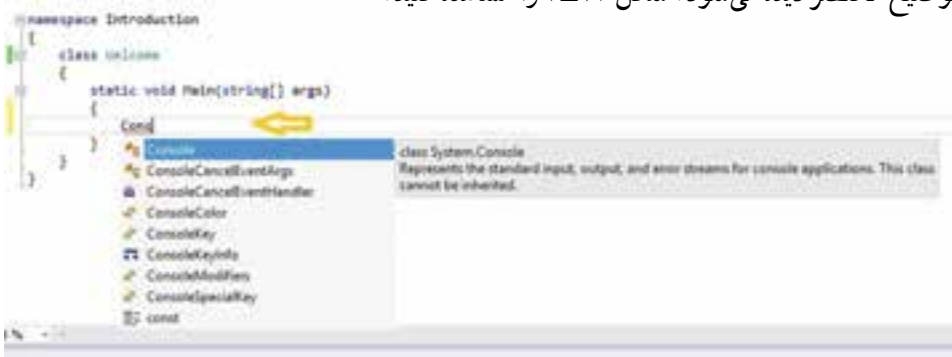
در داخل این فضای نامی، دستور class و سپس در داخل آن متد Main نوشته شده است. نام کلاس Welcome است مگر این که در مرحله قبل نام برنامه را تغییر نداده باشید که در این صورت

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 :

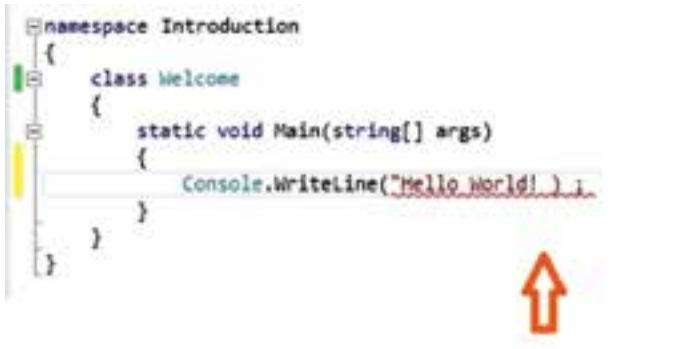
```
Console.WriteLine("Hello World");
Console.WriteLine("Welcome to C#");
```

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.

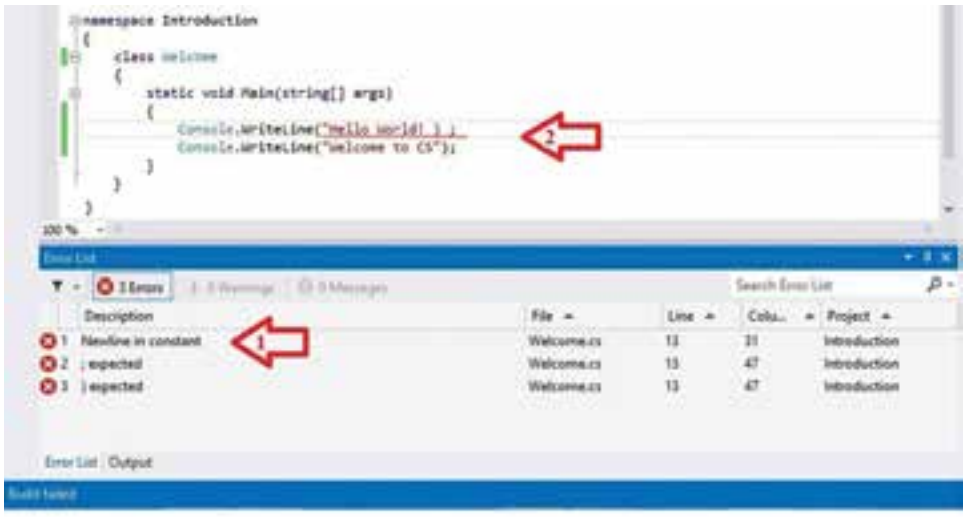


شکل ۱۵ــ۳ــ نشان دادن خطا در تایپ برنامه در محیط ویژوال استودیو

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 F6 را بزنید. اگر برنامه خطا داشته باشد، خطاها در پنجره List Error 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 دکمه ▶ Start - بر روی نوار ابزار 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. (از پیوست شماره ۱ در انتهای کتاب استفاده 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(string[] args)
```

```
{
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 ( فرکانس برحسب هرتز , مدت زمان برحسب میلی ثانیه);
```

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. صدای بم دارای فرکانس کم و صداهای ریز 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 VSعلاوه 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 به زبان فارسی 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 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	
۱	Start Page	
۱۱	Toolbar	
۱۲	Visual Studio	
۱۳	Visual Studio Express	

فصل چهارم

آشنایی با انواع داده‌ها و متغیرها

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

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

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ متغیر را تعریف کند و انواع متغیر را در برنامه‌های خود به کاربندد.
۲ــ انواع داده‌ها را نام ببرد و تفاوت کاربرد هر یک را توضیح دهد.
۳ــ میزان حافظه و محدودهٔ انواع داده‌ها را بیان کند.
۴ــ متغیرها را به‌طور صحیح در برنامه اعلان کندو آن‌ها را مقداردهی نماید.
۵ــ شکل نمایش نقطه شناور را توضیح دهد و اعداد اعشاری را در این قالب بنویسد.
۶ــ ازمتد ()ReadLine برای دریافت داده‌های یک برنامه از ورودی استفاده کند.
۷ــ بر روی رشته دریافتی از ورودی، تغییراتی داده و سپس نمایش دهد.
۸ــ 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[۱] 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[۲] 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 Var ab e

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 گروه بندی 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 Representat on

۳ ــ Pr m t ve Data Type or Bu t In Data Type

۴ ــ Uns gnex numbers

جدول ۱ــ۴

	نوع داده	کاربرد نوع داده	مقدار حافظه (بایت)	کمترین مقدار	بیشترین مقدار
عددی	sbyte	اعداد صحیح		28	27
	byte	اعداد صحیح مثبت		0	255
	sho t	اعداد صحیح	2	32768	32767
	usho t	اعداد صحیح مثبت	2	0	65535
	nt	اعداد صحیح	4	2 47483648	2 47483647
	u nt	اعداد صحیح مثبت	4	0	4294967295
	ong	اعداد صحیح	8	9223372036854778508	9223372036854778507
	u ong	اعداد صحیح مثبت	8	0	844674407370955 6 5
	f oat	اعداد اعشاری	4	3.402823×0^{38}	3.402823×0^{38}
	doub e	اعداد اعشاری با دقت زیاد	8	$.797693\ 3486232 \times 0^{308}$	$.797693\ 3486232 \times 0^{308}$
	dec ma	اعداد صحیح بزرگ اعداد اعشاری با دقت بسیار زیاد	6	79228162514264337593543950335 7.9×0^{28}	79228162514264337593543950335 7.9×0^{28}
غیرعددی	boo	مقدار منطقی		fa se	t ue
	char	یک حرف یا علامت (کراکتر)	2	0 کد کراکتر مطابق با سیستم Un code	65535 کد کراکتر مطابق با سیستم Un code
	str ng	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

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 اعشار 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 دادهdouble 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 Ass gnment

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.

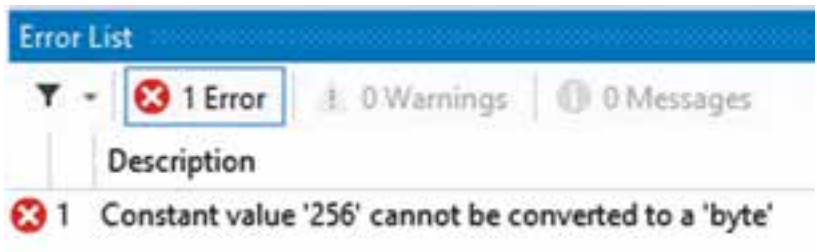
۳ــ اگر عددی را بخواهید در داخل یک متغیرذخیره کنید که خارج از ظرفیت و گنجایش آن متغیر باشد، مترجم متوجه آن شده و اجازه نمی‌دهد.

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

۱ــ In t a ze

در یک برنامه به زبان C# می‌توانید اعداد صحیح را در مبنای ۱۶ نیز بنویسید. برای این منظور قبل از عدد مورد نظر از پیشوند 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	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.

۵ــ۴ــ 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   3.141592653589793238;
```

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 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[۱] 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[۲] به وسیلهٔ آقای چارلز سیمونیی[۳] 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 به این نکته دقت کنید که متغیر a و A مستقل هستند.

۱ــ Came Notat on

۲ــ Hungar an Notat on

۳ــ Char es S mony

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

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 مقدار دهی 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).