

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 به زبان محاوره‌ای و لزوم یادگیری یک زبان برنامه‌نویسی آشنا می‌شوید.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

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

۱-۱- داده‌ها و اطلاعات^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 به صورت کتی با یک مقدار عددی و یا به صورت کیفی نشان داده می‌شود.

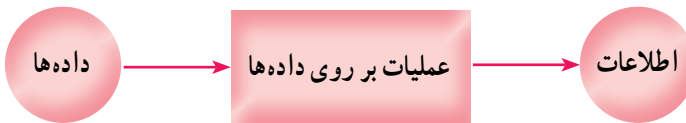
مثلاً آزمون درس ریاضی یک کلاس، نمراتی حاصل می‌شوند که این نمرات به عنوان داده‌ها در نظر گرفته می‌شوند.

۱۷/۵, ۱۴, ۱۹, ۱۱, ۱۸, ...

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

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^۱ دما هستیم که به وسیله 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 به‌سطح 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 ۱-۱- ارتباط بین داده‌ها، عملیات و اطلاعات

با توجه به شکل ۱-۱، مشاهده می‌شود که اطلاعات، حاصل انجام عملیات بر روی داده‌ها است. مجموعه محاسبات و عملیاتی که بر روی داده‌ها صورت می‌گیرد را پردازش^۲ 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.

۱- Sensor

۲- Process

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

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

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

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

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 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 به کار 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 (اتزکتور) به موتور بفرستد. سیستم کامپیوتری مربوطه را ECU (Engine Control Unit) می‌نامند. تصور کنید که اگر سنسورها دچار اشکال شوند چه اتفاقی می‌افتد!

وسایلی که در منزل شما قرار دارد و فکر می‌کنید که حتماً 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 ارایه دهید.

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

به مجموعه دستوراتی که به کامپیوتر می‌فهماند که چه نوع پردازشی را بر روی داده‌ها انجام دهد و همچنین اطلاعات به دست آمده را چگونه نمایش دهد برنامه^۱ 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^۲ می‌گویند. مثلاً برای جمع دو عدد ۵ و ۸ دستورهای زبان ماشین در یک پردازنده شرکت اینتل^۳ 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^۴، باید به زبان ماشین تبدیل شود تا کامپیوتر بتواند آن را بفهمد و اجرا نماید. مترجم خود نیز یک برنامه کامپیوتری می‌باشد که وظیفه آن، ترجمه و تبدیل دستورات یک زبان سطح بالا، به کدهای زبان ماشین می‌باشد.

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

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

۱_ Program

۲_ Machine Language

۳_ Intel

۴_ Compiler

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 نرم افزار 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.

۲-۱- انواع زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی را از نظر این که تا چه اندازه به زبان محاوره‌ای ما نزدیک باشند به صورت 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

۱- Low Leve Language

۲- Assem b y Language

۳- H gh Leve Language

۴- Med um Leve Language

پردازشگر وجود دارد استفاده می‌کنند.

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

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

پاسخ: نرم افزارهای کاربردی آماده، برای تسهیل و انجام امور عادی و روزمره که بین کاربران مشترک است طراحی شده اند و به تدریج و در طی سالیان طولانی با توجه به بازخورد^۲ کاربران تکمیل تر شده اند. مثلاً برنامه صفحه گسترده^۳ (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 فروشگاه خریداری کنیم به سراغ یک خیاط برویم تا یک لباس کاملاً سفارشی^۴ و مطابق با سلیقه و مدل مورد نظر ما درست کند. البته طراحی و تولید یک نرم افزار کاربردی، یک کار سنگین، پیچیده و مستلزم صرف وقت زیادی است و به وسیله‌ی یک گروه از مهندسان با تخصص‌های مختلف طی چند مرحله انجام می‌شود که برنامه‌نویسی تنها یکی از مراحل آن است. البته نگران نباشید ما در این کتاب برنامه‌های کوچک و مقدماتی برای یادگیری زبان برنامه‌نویسی C# می‌نویسیم و در سال بعد شما با مراحل تولید یک نرم افزار و روش‌های طراحی و حل مسئله^۵ آشنا خواهید شد تا بتوانید با استفاده از ابزار برنامه‌نویسی، برنامه‌های کاربردی برای حل یک مشکل در یک سازمان بنویسید و آن را تجربه کنید.

۱- App cat on Software

۲- Feedback

۳- Spreadsheet

۴- Custom zed

۵- Prob em So v ng

نقطه نظر یک برنامه نویس :

من برنامه‌های زیادی را نوشته‌ام ولی تاکنون برنامه‌ای بدون شناسایی و درک کامل مسئله از ابتدا، ننوشته‌ام. حتی برای کاری که بخواهم برای دوستم مجانی انجام دهم، همین روش را به کار می‌گیرم.



شکل ۲-۱- نظر یک برنامه‌نویس

فودآزمایی فصل اول

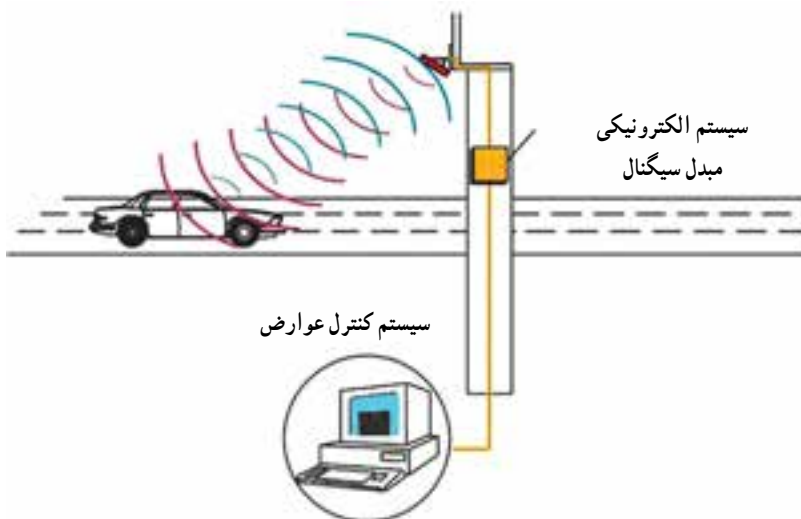
- ۱- در هر یک از موارد زیر داده، اطلاعات و پردازش را مشخص نمایید.
- الف) نرم افزار تهیه کارنامه در مدارس آموزش و پرورش، جمع نمرات و معدل هر دانش آموز را براساس نمراتش، محاسبه و چاپ می کند.
- ب) در یک شرکت مهندسی، حقوق دریافتی کارکنان براساس میزان حضورساعات کار در شرکت محاسبه و پس از کسر مالیات و حق بیمه پرداخت می شود و بر روی فیش چاپ می شود.
- ج) یک روش الکترونیکی در پرداخت عوارض خودرو در هنگام عبور از بزرگراهها، استفاده از تکنولوژی RFID^۱ است. در این روش یک برچسب^۲ مخصوص در روی شیشه خودرو چسبانده می شود که دارای یک کد منحصر به فرد است و اطلاعات شماره پلاک خودرو و کد مربوطه و همچنین میزان اعتبار پولی در هنگام خرید برچسب، در کامپیوتر ثبت می شود. هرگاه 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 می شود (شکل ۳-۱) و در نتیجه شماره پلاک خودرو و میزان اعتبار آن استخراج می شود، عوارض عبور از بزرگراه از اعتبار مربوطه به صورت خودکار کسر می شود (شکل ۴-۱).
- ۲- من چه سطحی از زبان برنامه نویسی هستم؟



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

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

۲- Tag ۲



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

تا کنون زبان‌های زیادی در سطح من تولید شده است، اما تنها برخی از آنها مورد اقبال متخصصان قرار گرفته است. من به زبان محاوره‌ای خیلی نزدیک هستم.
 ۳- توضیحات ستون چپ را با اصطلاح ستون سمت راست تطابق دهید. (یک مورد اضافی است.)

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

- ۴- در یک برنامه، منظور از ورودی همان و منظور از خروجی همان است.
- ۵- در محاسبه عبارت $z \ x^2 \ y$ ، ورودی (ها) و خروجی (ها) کدامند؟
- ۶- زمان ورود و خروج کارکنان یک شرکت با عکس‌برداری از چهره آنان و به کمک کامپیوتر صورت می‌گیرد. در این روش ورودی (داده)، پردازش و خروجی (اطلاعات) را تعیین کنید.
- ۷- چرا لازم است یک زبان برنامه‌نویسی را یاد بگیریم؟

- ۱- با استفاده از جستجو در اینترنت، چند زبان سطح بالای متداول امروزی را پیدا کرده و آنها را نام ببرید.
- ۲- 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.
- ۳- 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 حس‌گرهای آن را شناسایی کنید. همچنین وقتی به منزل برگشتید با استفاده از اینترنت منظور از کلمات دیاگ و فلش کردن که توسط تعمیرکاران استفاده می‌شود را جستجو کرده و متوجه مفهوم آن شوید.

واژگان و اصطلاحات انگلیسی فصل اول

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

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 نام‌گذاری 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:

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

۱-۲- آشنایی با زبان C#

زبان C# یک زبان سطح بالا، شی‌گرا^۱ و همه منظوره است که به وسیله شرکت مایکروسافت هم‌زمان 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

^۱ Object Oriented Language

امکانات آنها را بهبود داده است. تلاش شده است که بهترین ویژگی‌ها گردآوری شود، اما برخلاف زبان جاوا که متن باز است، #C در انحصار و اختیار سازنده آن یعنی شرکت مایکروسافت است. زبان‌های C و Java هر دو به زبان C برمی‌گردند که در سال ۱۹۷۰ ابداع شد و معروفیت آن به دلیل نوشتن سیستم عامل UNIX به وسیله آن بود. زبان C، یک زبان حرفه‌ای است که دست برنامه‌نویس را برای نوشتن برنامه و دسترسی به سخت افزار، باز می‌گذارد و دارای انعطاف 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

BJARNE STROUSTRUP



بیارنه استراس تروپ، زبان C را بر مبنای C و با اضافه کردن ویژگی‌های شی‌گرای در سال ۱۹۷۹ طراحی کرد و در سال ۱۹۸۳ این زبان به نام C منتشر شد.

DENNIS M. RITCHIE



دنيس 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.

ANDERS HEJLSBERG



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

JAMES ARTHUR GOSLING



جیمز آرتزگاسلینگ 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 بدع کرد وی در ابتدا نام Oak که نام درختی در جلوی دفتر کار وی بود، بر این زبان انتخاب کرد، اما بعدها به Java تغییر نام پیدا کرد. زبان جاوا با هدف ساده کردن و بالا بردن امنیت و با شعار برنامه‌ریزی یک بار بنویس و هر جا اجرا کن طراحی شده است.

شکل ۱-۲- پدیدآورندگان زبان‌های برنامه‌نویسی #C، Java، C، C

برای مطالعه

لایه نرم افزاری .NET Framework. شرکت مایکروسافت

این لایه نرم افزاری 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^۱)
- 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 بانک‌های اطلاعاتی می‌باشد.

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

۲-۲- شروع برنامه‌نویسی

همان‌طور که برای تهیه و پخت یک غذا، به مواد اولیه، لوازم آشپزی و دستور پخت^۱ نیاز داریم برای تولید یک برنامه نیز، به یک کامپیوتر یا لپ‌تاپ، لوازم برنامه‌نویسی (یک ویرایشگر متنی^۲ و یک برنامه مترجم) و همچنین به یک الگوریتم نیاز داریم. اگر یک کامپیوتر با سیستم عامل ویندوز^۳ یا بالاتر در دسترس باشد تقریباً تمام مواد اولیه و لوازم مورد نیاز را در اختیار داریم.

^۱ Common Language Runtime

^۲ Rec pe

^۳ Text Ed tor

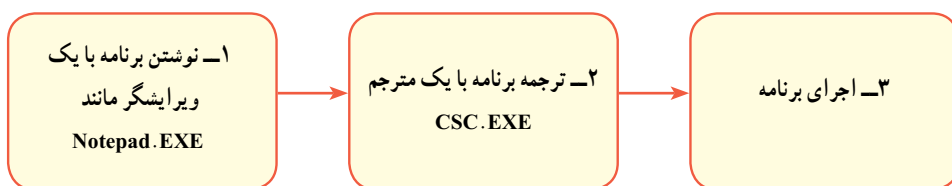
در این صورت با طی مراحل زیر می‌توانیم برنامه‌ای را نوشته، ترجمه کرده و سپس اجرا نماییم.

- ۱- نوشتن برنامه و ذخیره آن با استفاده از یک ویرایشگر مانند برنامه Notepad ویندوز
- ۲- ترجمه برنامه ذخیره شده به وسیله مترجم زبان C# به نام CSC.EXE

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

- ۳- اجرای برنامه ترجمه شده

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



نمودار ۱-۲- مراحل برنامه‌نویسی، ترجمه و اجرا

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

۲-۳- اولین برنامه به زبان C#

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

```
class WelcomeToCSharp
```

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

برنامه ۱-۲- اولین برنامه به زبان C#

این برنامه کوچک فقط یک پیام خوشامدگویی بر روی صفحه نمایش نشان 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 صفحه نمایش ندارد. همان طور که در برنامه Notepad آنچه که می نویسید همگی با یک رنگ نوشته می شود.

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

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

الف) نگاه جزئی تر در حد کلمات و علامت ها

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

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

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

علامت هایی مانند {، }، (،) و نیز در این برنامه دیده می شود که معمولاً برای شروع یا پایان یک قسمت استفاده می گردد.

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

```
class WelcomeToCSharp
{
}
}
```

کلاس برنامه ۱-۲

۱_ Reserved words

۲_ 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 `Main` می‌گوییم که بدنه اجرایی برنامه است هر دستوری که در این قسمت نوشته شود به وسیله 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 ۱-۲

آخرین قسمتی که در برنامه ۱-۲، 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 دستور بالا، پیام خوشامدگویی `Welcome to C#!` بر روی صفحه نمایش، نشان داده می‌شود. آیا شما قبلاً درست حدس زده بودید که برنامه ۱-۲، چه پیامی را بر روی صفحه نمایش، نشان می‌دهد؟!

به وسیله 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 علامت‌های نقل قول بر روی صفحه نمایش، نشان داده نمی‌شوند. بلکه این علامت‌ها برای مشخص کردن شروع و پایان عبارتی است که می‌خواهیم روی صفحه نشان داده شود.

_Method

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.

۲-۴- الگوی یک برنامه ساده به زبان C#

یک برنامه کاربردی نوشته شده به زبان C#، شامل مجموعه‌ای از کلاس‌ها است که هر یک از آنها نیز شامل تعدادی متد هستند. اما در یک برنامه ساده مانند برنامه ۱-۲، 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 ساختار کلی یک برنامه ساده به زبان C# در زیر آمده است. الگوی زیر را به خاطر بسپارید.

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

الگوی یک برنامه ساده به زبان C#

۲-۵- کلاس (class) چیست؟

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

مربوط به یک موضوع است و عملیاتی که می‌توان بر روی آنها انجام داد. در زبان C# گنجینه‌ای از کلاس‌های مختلف و کاربردی، از قبل تعریف شده و آماده وجود دارد که برنامه‌نویس 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.

۱-۵-۲- نحوه تعریف کلاس: در زبان C# این امکان برای برنامه‌نویس فراهم است که کلاس جدیدی را تعریف کند. همان‌طور که در زیر مشاهده می‌کنید از کلمه کلیدی `class` برای تعریف و مشخص کردن یک کلاس جدید استفاده می‌شود. در جلوی کلمه `class`، یک نام دلخواه ذکر می‌گردد که نام کلاس است. مانند `WelcomeToCSharp`

نام کلاس `class`

```
{  
  
    تعریف داده‌ها  
    و عملیات بر روی آنها  
  
}
```

الگوی یک کلاس

نکته

نام گذاری کلاس: نام یک کلاس به وسیله برنامه‌نویس 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 خواندن نام به راحتی انجام شود و تشخیص کلمات آسان باشد، از روش پاسکال استفاده کنید که در آن، اولین حرف هر کلمه با حرف بزرگ نوشته می‌شود.

۶-۲- 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 قرار گیرند، 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 برنامه‌سازی 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 ۱-۲، فقط به تعریف متد Main() می‌پردازیم و از متدهای آماده در زبان C# استفاده خواهیم کرد.

۱-۶-۲- استفاده از متدهای آماده: تعداد زیادی متد در کلاس‌های آماده زبان C# مانند 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 ۱-۲ 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 بنویسید (نمودار ۲-۲).

نام حوزه
System

نام کلاس
Console

نام متد
WriteLine()

نمودار ۲-۲- طریقۀ استفاده از یک متد (System.Console.WriteLine)

به این ترتیب برای استفاده از متد WriteLine() در برنامه ۲-۱ مشاهده می‌کنید که فضای نامی System 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 ۲-۲- استفاده از توضیحات در متن برنامه

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

۱_ Namespace

۲_ Comment

خط دوم 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 ۲-۲، پیامی را که می‌خواهید نمایش داده شود باید بین علامت‌های نقل قول " قرار دهید. برای مثال اگر اسم شما Mohammad است و بخواهید روی صفحه نشان داده شود، باید به صورت زیر بنویسید :

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

اگر بخواهید نام و نام خانوادگی خود را در دو سطر، نمایش دهید، در این صورت می‌توانید دوبار از `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^۱ 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^۲ 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
```

```
*/
```

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

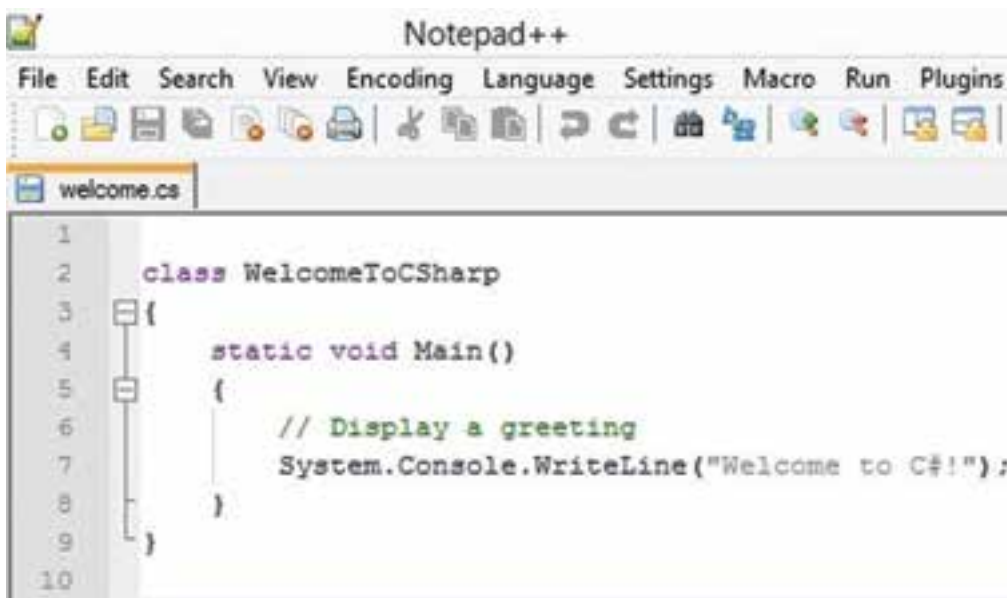
سری به سایت Mono بزنید. www.go-mono.com



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

قدم اول: نوشتن و تایپ برنامه: برای نوشتن یک برنامه ساده مانند برنامه ۱-۲، 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 به صورت رایگان از طریق سایت^۲ آن قابل دانلود است استفاده نماییم. این ویرایشگر به منظور برنامه نویسی به زبان‌های مختلف طراحی شده است به طوری که کلمات رزرو شده، رشته‌ها و توضیحات در این ویرایشگر با رنگ‌های مختلف نشان داده می‌شود (شکل ۳-۲). البته برای استفاده از این ویژگی باید ابتدا از منوی Language، زبان برنامه نویسی مورد نظر خود را C# انتخاب کنید.



شکل ۳-۲- تصویری از محیط ویرایشگر Notepad

^۱_ Text Ed tor

^۲_ <http://notepad.p.us.p.us.org>

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



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

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

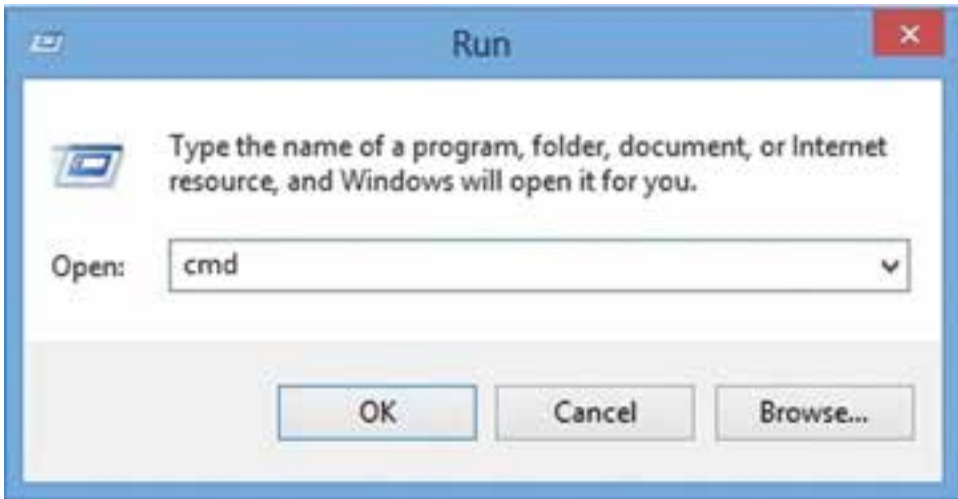
```

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

به حروف کوچک و بزرگ توجه داشته باشید.

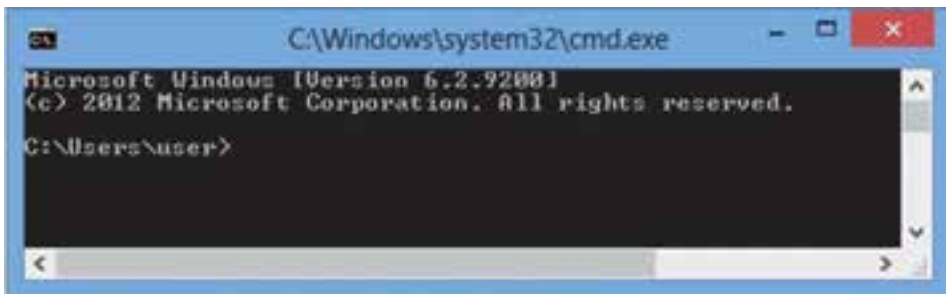
شکل ۲-۴- تصویری از محیط ویرایشگر Notepad

قدم دوم : ترجمه و اجرای برنامه : پس از نوشتن و ذخیره کردن برنامه، لازم است ابتدا برنامه را ترجمه کنیم و اگر اشکالی در تایپ آن وجود دارد آن را برطرف کنیم.
برای ترجمه برنامه مراحل زیر را دنبال کنید.
۱- از طریق گزینه Run فرمان cmd را اجرا کنید، تا وارد پنجره فرمان شویم (شکل ۲-۵).



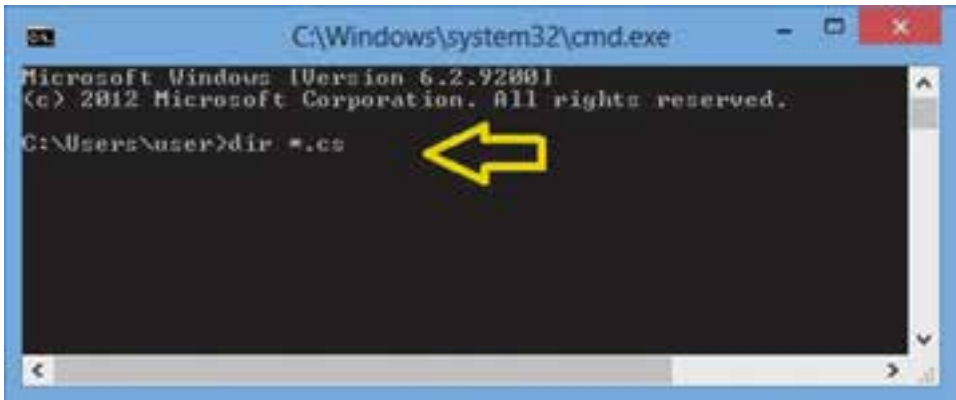
شکل ۲-۵- فرمان cmd

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



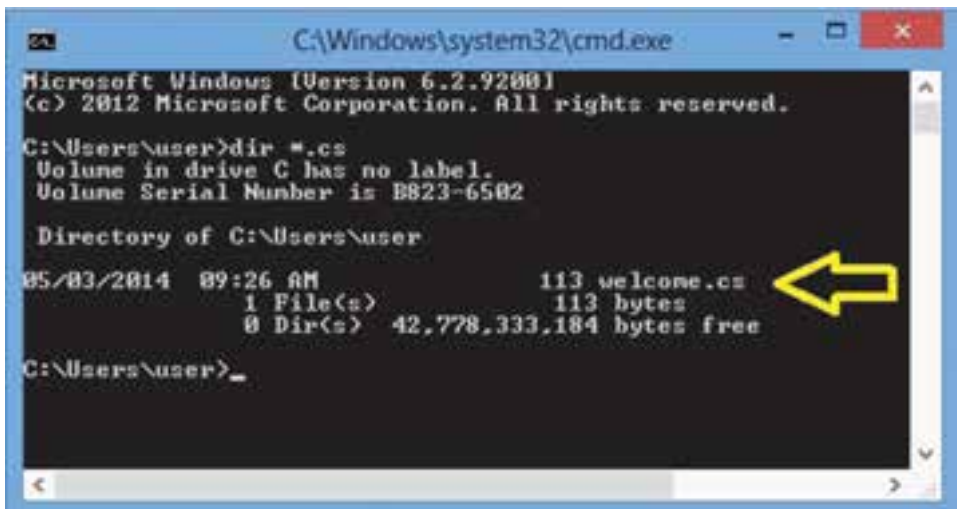
شکل ۲-۶- پنجره command prompt

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



شکل ۲-۷- فرمان Dir

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



شکل ۲-۸- مشخصات فایل برنامه

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

csc filename.cs

```

C:\Windows\system32\cmd.exe

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

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

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

شکل ۹-۲- فرمان ترجمه برنامه

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

```

C:\Windows\system32\cmd.exe

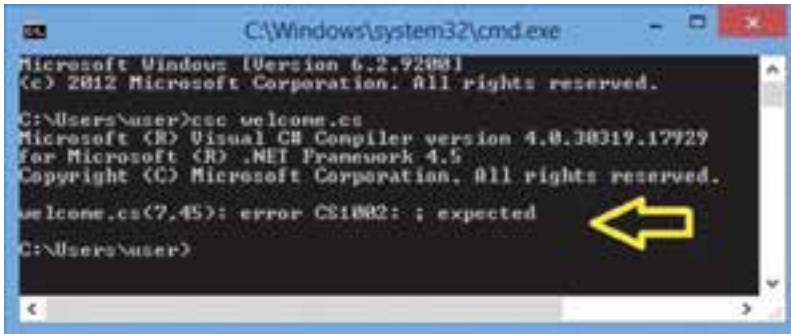
Directory of C:\Users\user
05/03/2014  09:26 AM                113 welcome.cs
               1 File(s)                113 bytes
               0 Dir(s) 42,778,288,128 bytes free.

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

C:\Users\user>
  
```

شکل ۱۰-۲- نتیجه ترجمه برنامه

۷- اگر در تایپ برنامه اشتباهی انجام داده باشید، مترجم نمی‌تواند برنامه را ترجمه کند و در این صورت 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 ۲-۷ ظاهر می‌شود.



شکل ۱۱-۲- پیام خطای مترجم برای نوشتن علامت ؛ در خط ۷ برنامه

در خط آخر این شکل مشاهده می‌کنید که از سمت چپ، ابتدا نام فایل برنامه یعنی welcome.cs و سپس در جلوی آن دو عدد ۷ و ۴۵ نوشته شده است که مکان خطا را در برنامه نشان می‌دهد. عدد اول (۷) شماره سطر و عدد دوم (۴۵) شماره ستون محل خطا در برنامه است. بعد از این دو عدد، کد خطا (error CS1002) و سپس توضیح آن (expected) ذکر شده است. شما باید توضیح 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 وسیله ویرایشگر باز کنید و به سطر ۷ و ستون ۴۵ بروید و علامت نقطه ویرگول را اضافه کنید و سپس برنامه را تحت همان نام قبلی ذخیره کنید و دوباره به پنجره فرمان بازگردید و عمل ترجمه را انجام دهید. اگر 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

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

```

C:\Windows\system32\cmd.exe

C:\Users\Saeed>g++ welcome.cpp
Microsoft (R) Visual C++ Compiler version 4.0.30319.17929
for Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Users\Saeed>dir *.*
    
```

شکل ۱۲-۲- جستجوی فایل ترجمه شده

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

```

C:\Windows\system32\cmd.exe

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

Directory of C:\Users\Saeed

05/03/2014  09:26 AM                113 welcome.cpp
05/03/2014  09:45 AM            3,584 welcome.exe
                2 File(s)              3,697 bytes
                0 Dir(s)      42,777,370,624 bytes free

C:\Users\Saeed>
    
```

شکل ۱۳-۲- لیست فایل های برنامه و ترجمه شده

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

```

C:\Windows\system32\cmd.exe

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

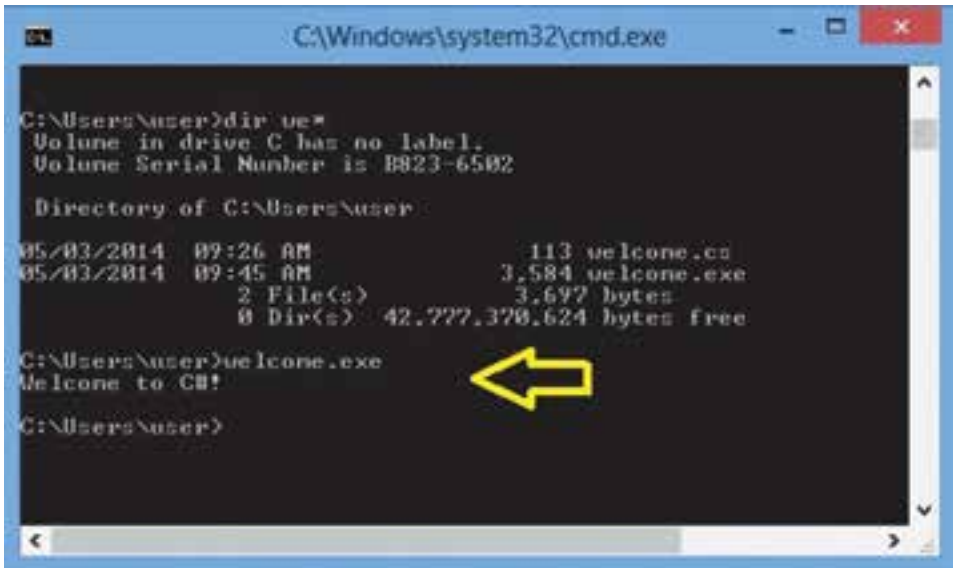
Directory of C:\Users\Saeed

05/03/2014  09:26 AM                113 welcome.cpp
05/03/2014  09:45 AM            3,584 welcome.exe
                2 File(s)              3,697 bytes
                0 Dir(s)      42,777,370,624 bytes free

C:\Users\Saeed>welcome.exe
    
```

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

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



```
C:\Windows\system32\cmd.exe

C:\Users\User>dir ve*
Volume in drive C has no label.
Volume Serial Number is B823-6502

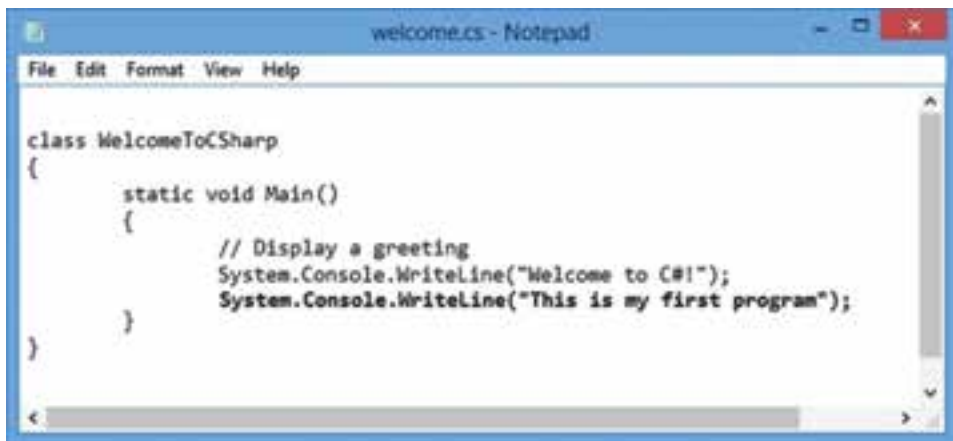
Directory of C:\Users\User:
05/03/2014  09:26 AM                113 welcome.cs
05/03/2014  09:45 AM            3,584 welcome.exe
                2 File(s)                3,697 bytes
                0 Dir(s)         42,777,378,624 bytes free

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

C:\Users\User>
```

شکل ۲-۱۵- نتیجه اجرای برنامه

تبریک می‌گوییم که توانستید اولین برنامه خود را به زبان C# بنویسید و آن را ترجمه و اجرا نمایید.
۱۲- برنامه را با ویرایشگر باز کنید و یک دستور مانند شکل ۲-۱۶ به آن اضافه کنید. (تغییرات 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.



```
welcome.cs - Notepad
File Edit Format View Help

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

شکل ۲-۱۶- اضافه کردن یک دستور به برنامه

۱۳- دوباره برنامه را با ویرایشگر باز کنید و در اولین دستور به جای متد WriteLine() از متد Write() مانند شکل ۲-۱۷ استفاده کنید.



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

شکل ۲-۱۷- استفاده از متد Write()

چه تفاوتی در اجرای برنامه حاصل می شود؟ پیام ها چگونه نشان داده شدند؟
۱۴- برنامه را به حالت اولیه خود بازگردانید و یک دستور نیز به صورت شکل ۲-۱۸ به آن اضافه کنید. برنامه را ذخیره کنید و سپس ترجمه و اجرا نمایید. چه تغییری در خروجی ایجاد می شود؟



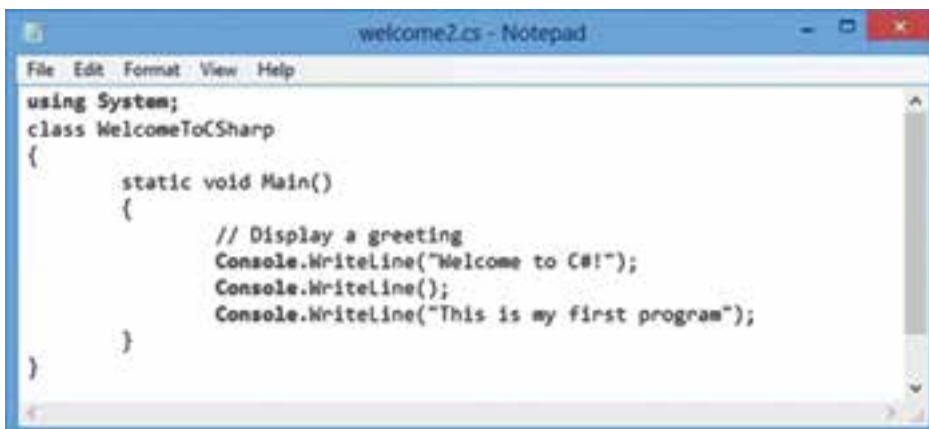
```
class WelcomeToCSharp
{
    static void Main()
    {
        // Display a greeting
        System.Console.WriteLine("Welcome to C#!");
        System.Console.WriteLine();
        System.Console.WriteLine("This is my first program");
    }
}
```

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

۱۵- همان طور که در شکل ۱۸-۲ 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 این صورت می‌توانیم در داخل برنامه، کلمه `System` را از ابتدای دستورات حذف کنیم. برای معرفی فضای نامی از دستور `using` به صورت زیر استفاده می‌کنیم:

؛ فضای نامی `using`

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



```

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

```

شکل ۱۹-۲- استفاده از فضای نام

۱۶- 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 ۲۰-۲). این برنامه را با توجه به نکته زیر تایپ نموده و سپس ترجمه و اجرا نمایید. و به اعداد نشان داده شده بر روی صفحه توجه کنید. آیا محاسبات درست انجام شده است؟

نکته

برای این که 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.

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

```

somecalc.cs - Notepad
File Edit Format View Help
using System ;
class SomeCalculations
{
    static void Main( )
    {
        // Calculates some basic calculations
        Console.WriteLine(15 + 3); // 15 is added by 3 جمع
        Console.WriteLine(15 - 3); // 15 is subtracted by 3 تفریق
        Console.WriteLine(15 * 3); // 15 is multiplied by 3 ضرب
        Console.WriteLine(15 / 3); // 15 is divided by 3 تقسیم
    }
}

```

شکل ۲۰-۲- برنامه چهار عمل اصلی

۱۷- برنامه شکل ۲۰-۲ را با اعداد دیگری آزمایش کنید یعنی به جای ۳ و ۱۵ اعداد دلخواه دیگری مانند ۲۵ و ۴۰ قرار دهید و سپس برنامه را ترجمه و اجرا نمایید. اعدادی که روی صفحه نشان داده می شوند را یادداشت کنید، آیا محاسبات درست انجام شده است؟

۱۸- اگر به خروجی دستور زیر توجه کنید

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

مشاهده خواهید کرد که عدد ۱ چاپ می شود چون تقسیم صحیح و بدون اعشار انجام می شود. در صورتی که 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 به صورت اعشاری بنویسید و سپس برنامه را با همان نام قبلی ذخیره و ترجمه و اجرا نمایید.

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

۱۹- برای این که پیام ها بهتر دیده شوند، می توانیم 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 ۲۱-۲ اضافه کنید. برنامه را تحت نام فعلی ذخیره کنید و سپس ترجمه و اجرا نمایید.

```

using System;
class SomeCalculations
{
    static void Main()
    {
        // Clear console screen
        Console.Clear();
        // Calculates some basic calculations
        Console.WriteLine(40 + 15); // 40 is added by 15
        Console.WriteLine(40 - 15); // 40 is subtracted by 15
        Console.WriteLine(40 * 15); // 40 is multiplied by 15
        Console.WriteLine(40.0 / 25); // 40 is divided by 15
    }
}

```

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

۲۰- پنجره فرمان را ببندید وارد محیط میزکار شوید و با استفاده از Computer وارده پوشه ای شوید که فایل های خود را در آن ذخیره کرده اید. فایل های اجرایی (EXE) تولید شده را پیدا کنید. با دوبار کلیک بر روی آیکن آن، برنامه را اجرا کنید. چه اتفاقی می افتد؟ در یک لحظه صفحه کنسول باز شده و به سرعت بسته می شود و شما فرصت نمی کنید نتایج برنامه را مشاهده کنید. لازم است در انتهای برنامه، دستوری بنویسید که کامپیوتر را به صورت موقتی متوقف کند تا شما بتوانید نتایج محاسبات را ببینید. به این منظور از متد ReadKey() استفاده می کنیم. با اجرای این متد، کامپیوتر منتظر زدن کلیدی باقی می ماند. 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 می کنیم. شکل ۲۲-۲ استفاده از این متدها را نشان می دهد. برنامه چهار عمل اصلی را باز کنید و تغییرات زیر را در آن اعمال و برنامه را ترجمه و اجرا کنید.

```

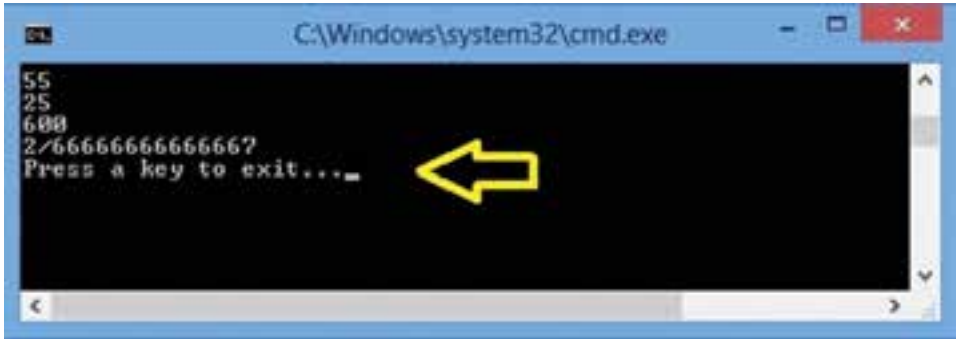
using System;
class SomeCalculations
{
    static void Main()
    {
        // Clear console screen
        Console.Clear();
        // Calculates some basic calculations
        Console.WriteLine(40 + 15); // 40 is added by 15
        Console.WriteLine(40 - 15); // 40 is subtracted by 15
        Console.WriteLine(40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

        Console.WriteLine("Press any key to exit...");
        // Wait untill a key is pressed.
        Console.ReadKey();
    }
}

```

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

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 ۲۳-۲).



شکل ۲۳-۲- استفاده از متد ReadKey() برای دریافت یک کلید و توقف موقتی برنامه

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

۲۱- 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() ;
```

۲۲- 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 ۱-۲ - جعبه رنگ ConsoleColor

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

خودآزمایی فصل دهم

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

۱۰- مثالی از کاربرد دستور using بیاورید.

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

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

ب) صفحه کنسول نمایش را پاک کنید.

پ) اجرای برنامه تا فشردن یک کلید متوقف شود.

ت) صفحه کنسول نمایش با رنگ آبی پاک شود.

تمرینات برنامه نویسی فصل دهم

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

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

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

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

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

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

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

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

۶- با اجرای برنامه زیر چه عبارتی بر روی صفحه نشان داده می‌شود؟ به نظر شما چه ارتباطی

بین علائم { } و { } و { } با اعداد ۱۸ و ۱۵ و ۱۵ و ۱۸ وجود دارد؟

```
Class SomeCalculations
```

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

۷- تمرین زیر به زبان انگلیسی است. آن را با دقت بخوانید و برنامه خواسته شده را بنویسید.

برای زیبایی خروجی از جعبه رنگ ConsoleColor استفاده نمایید.

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

```
/////
| o o |
( | ^ | )
| _ |
```

واژگان و اصطلاحات انگلیسی فصل 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	