فصل اول

یادآوری و تکمیل مطالب

در کتاب برنامه سازی۱، با مفهوم برنامه، برنامه نویسی و سطوح مختلف زبان‌های برنامه نویسی آشنا شدید، سپس در فصل‌های بعدی کتاب، دستورات مقدماتی و پایه‌ای زبان برنامه نویسی C# را آموختید. همچنین توانستید برنامه‌هایی به زبان C# بنویسید که در محیط کنسول سطر فرمان اجرا شوند. اکنون در این فصل، به یادآوری مطالب آموخته شده قبلی می‌پردازیم و با نوشتن چند برنامه، دستورات پایه‌ای خوانده شده را بررسی و مرور می‌کنیم. در بعضی از برنامه‌ها نیز با ارایه دستورات جدید اطلاعات خود را تکمیل می‌کنید.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ از محیط SharpDevelop برای نوشتن و ترجمه برنامه‌های کنسولی استفاده کند.

۲ــ کاربرد متد TryParse را بیان نماید و از آن استفاده کند.

۳ــ با استفاده از عملگر سه تایی، دستور if را بازنویسی نماید.

۴ــ کلمه کلیدی const را برای تعریف ثابت‌های برنامه به کار بندد.

۵ــ دستورات تکرار متداخل(تودرتو) را در برنامه استفاده نماید.

۶ــ عملکرد دنباله معنی‌دار را بیان نماید و آن را به کار بندد.

۱ــ۱ــ یادآوری

برای مرور آنچه درباره برنامه نویسی با زبان C# آموختیم به یادآوری مباحث مربوط به محیط نگارش و نحوه اجرای برنامه در این محیط می‌پردازیم. یکی از راه‌های ایجاد برنامه به زبان C# نوشتن برنامه در قالب فایل متنی ساده و با ابزاری مانند Notepad است. اگر بخواهیم برنامه ۱ــ۱ را اجرا کنیم، باید آن را به صورت فایل متنی نوشته و با نام Test.cs ذخیره کنیم تا متن برنامه به زبان سی شارپ داشته باشیم.

مثال ۱ــ۱ : برنامه‌ای بنویسید که از لیست زبان‌های نمایش داده شده، زبان سطح میانی را تشخیص دهد و انتخاب نماید.

```
using System;
class Program
{
    static void Main ()
    {
        Console. WriteLine (" which of the following languages are in th middle
languag " );
        Console. WriteLine (" a. C");
        Console. WriteLine (" b. C++");
        Console. WriteLine (" c. C#");
        Console. WriteLine (" d. JAVA");
        Console. Write ("Select Answer (a Or b Or c Or d)? ");
        string choice = Console. ReadLine ( );

        if (choice = = "a")
            Console. WriteLine ("Bravo! Your Answer is correct");
        else if (choice = = "b" | | (choice = = "c" | | (choice = = "d")))
            Console. WriteLine ("Your Answer is incorrect");
        else
        Console. WriteLine ("ERROR");
    }
}
```

برنامه ۱ــ۱ــ تشخیص زبان سطح میانی

اکنون نوبت به ترجمه و تبدیل آن به فایل اجرایی رسیده است. برای ترجمه این برنامه ابتدا وارد

سطر فرمان شده، سپس وارد پوشه‌ای که فایل برنامه ما در آن قرار دارد می‌شویم و فرمان زیر را برای ترجمه برنامه وارد می‌کنیم :

```
CSC  Test.cs
```

در صورتی که برنامه اشکال داشته باشد لیست خطاها ظاهر می‌شود و باید خطاهای احتمالی را برطرف کنیم و در صورت موفقیت فایل اجرایی ساخته می‌شود. برای اجرای برنامه نیز کافی است نام فایل تولید شده را تایپ کرده و کلید Enter را بزنیم.

```
Test.EXE
```

این روش تولید فایل اجرایی برنامه، که از یک ویرایشگر برای نوشتن برنامه و از فایل CSC.EXE برای ترجمه برنامه، استفاده می‌کنند، بسیار وقت گیر است و عیب یابی برنامه‌های بزرگ نیز دشوار است.

۲ــ۱ــ استفاده از محیط توسعه برنامه متمرکز

در برنامه‌های کنسولی ابتدا با استفاده از یک ویرایشگر[۱] متن مانند Notepad، برنامه را نوشته و سپس با پسوند cs. ذخیره کرده و با مترجم C# یعنی برنامه CSC.EXE کامپایل و اجرا می‌کردیم.

روش دیگری که می‌توان برای تولید فایل اجرایی برنامه استفاده کرد، استفاده از یک محیط برنامه‌نویسی متمرکز (IDE[۲]) است که برای نوشتن وعیب‌یابی راحت‌تر برنامه‌ها، طراحی شده است. چندین IDE به وسیلهٔ شرکت‌ها و ارگان‌های مختلف طراحی و به بازار عرضه شده است. بعضی از آنها رایگان و به اصطلاح متن باز[۳] و بعضی دیگر تجاری[۴] هستند. برای انتخاب IDE باید با توجه به سیستم‌عامل کامپیوتر و بودجه مورد نظر اقدام کرد. مثلاً در سیستم‌عامل ویندوز می‌توان از محیط برنامه‌نویسی رایگان و متن باز SharpDevelop یا به اختصار develop#[۵] استفاده نمود (شکل ۱ــ۱).

ویژگی این IDE حجم کمِ آن نسبت به IDE مایکروسافت می‌باشد. برای استفاده از این IDE ابتدا باید نرم افزار Net Framework 4.5 را نصب کنید، سپس نرم افزار متن باز develop# نصب شود.

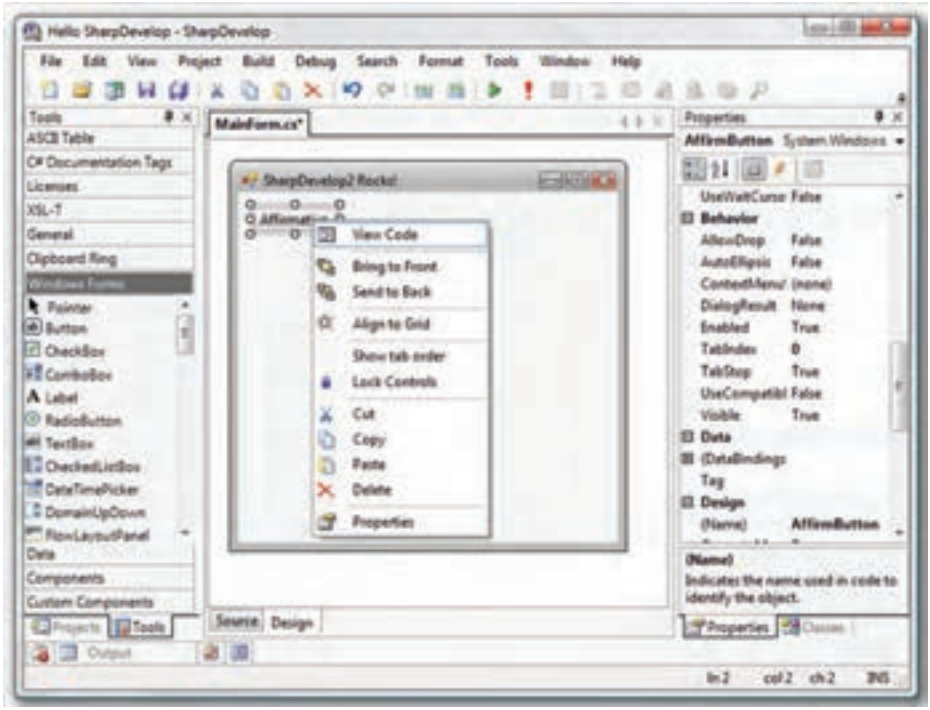
۱ــ Editor

۲ــ Integrated Development Environment

۳ــ Open Source

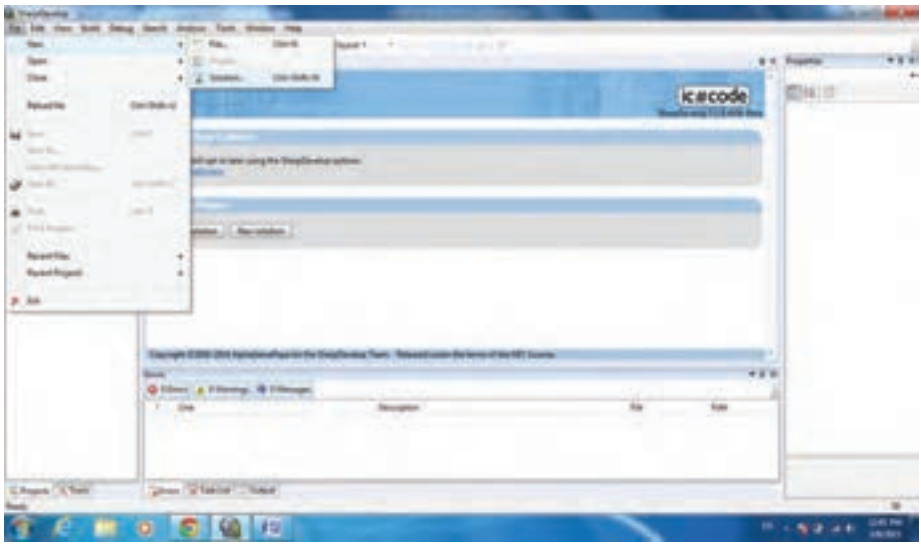
۴ــ Commercial

۵ــ # develop (short for SharpDevelop)



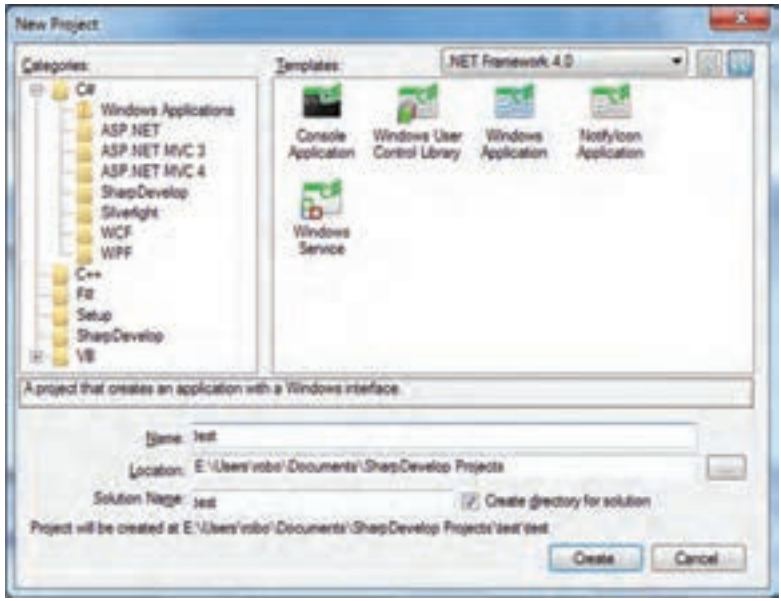
شکل ۱ــ۱ــ محیط برنامه نویسی رایگان و متن باز SharpDevelop

پس از نصب برنامه با کلیک روی آیکون SharpDevelop 5.1 در منوی start برنامه اجرا شده از منوی file گزینه solution را از زیر منوی new انتخاب می‌کنیم.



شکل ۲ــ۱ــ انتخاب گزینه solution از زیرمنوی new

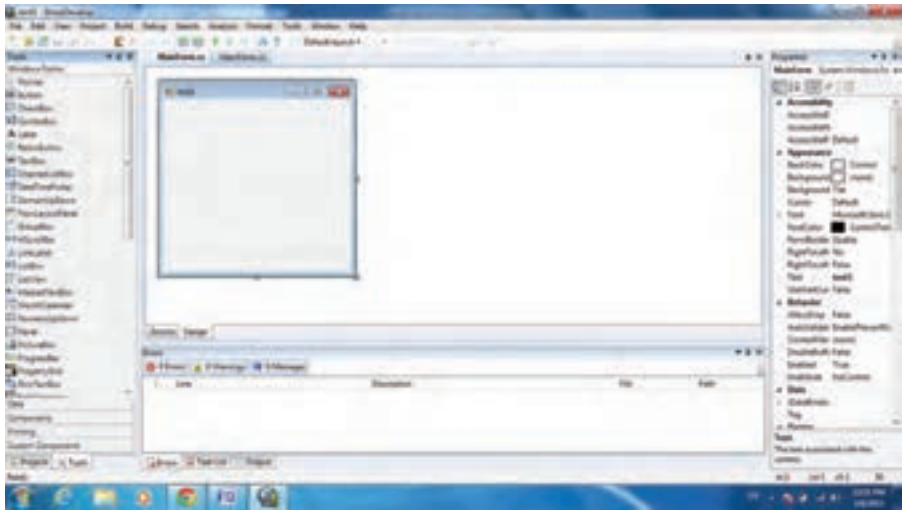
سپس گزینه windows application را انتخاب کرده در قسمت name نامی‌را برای پروژه انتخاب می‌کنیم.



شکل ۳ــ۱ــ تعیین نام پروژه

روی گزینه create کلیک می‌کنیم.

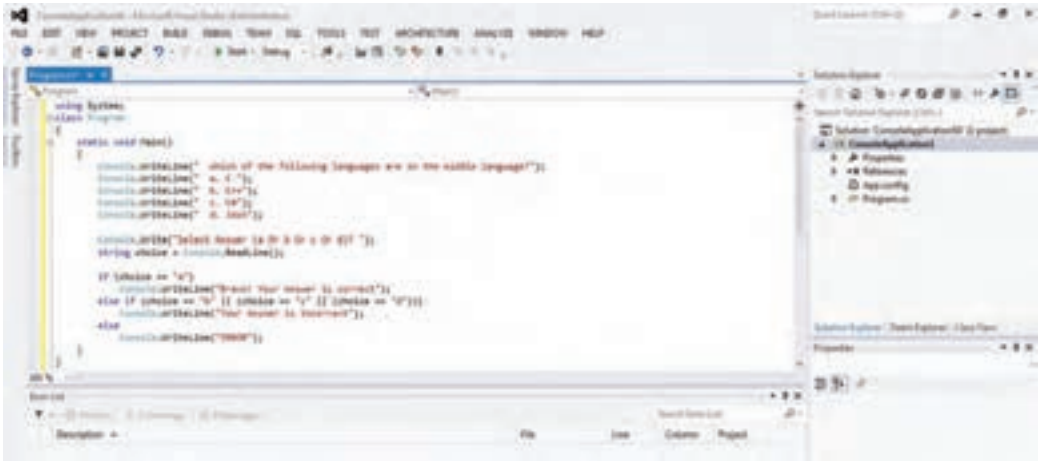
با انتخاب برگه design و برگه tools می‌توانیم به فرم اصلی و ابزارهای برنامه نویسی دسترسی داشته باشیم.



شکل ۴ــ۱ــ فرم اصلی و ابزارهای برنامه نویسی SharpDevelop

در این IDE فرم ابتدایی MainForm نام دارد.

تاکنون از محیط برنامه نویسی ویژوال استودیو اکسپرس[1] نسخه ۲۰۱۲ استفاده می‌کردید که مایکروسافت آن را به طور رایگان از طریق سایت خود عرضه می‌کند، تا برنامه نویسان برای تولید برنامه و ارزیابی این محصول، از آن استفاده کنند و نهایتاً نسخه اصلی ویژوال استودیو را خریداری[2] نمایند.



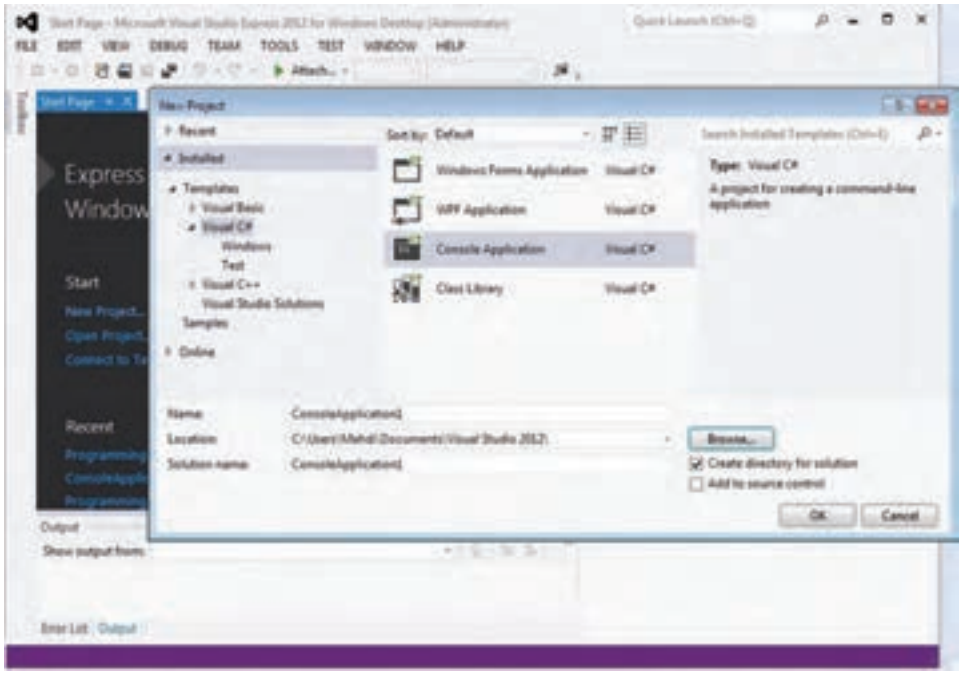
شکل ۵ــ۱ــ محیط Visual Studio

برای نوشتن برنامه‌های این کتاب نیز از وِیژوال استودیو اکسپرس ۲۰۱۲ یا به اختصار VS[3]، استفاده می‌کنیم. در اینجا طریقه نوشتن برنامه و اجرای آن در VS را به اختصار مرور می‌کنیم. برای نوشتن برنامه به زبان C# مانند برنامه ۱ــ۱، ابتدا VS را اجرا کرده، و در پنجره Start page گزینه New Project، گزینه Visual C# و سپس Console Application را انتخاب کنید.

۱ــ Visual Studio Express

۲ــ قیمت ویژوال استودیو ۲۰۱۳ حداقل حدود ۵۰۰ دلار می‌باشد (برای اطلاع از قیمت دقیق و جزئیات به سایت مایکروسافت مراجعه کنید).

۳ــ Visual Studio



شکل ۶ــ۱ــ پنجره پروژه جدید

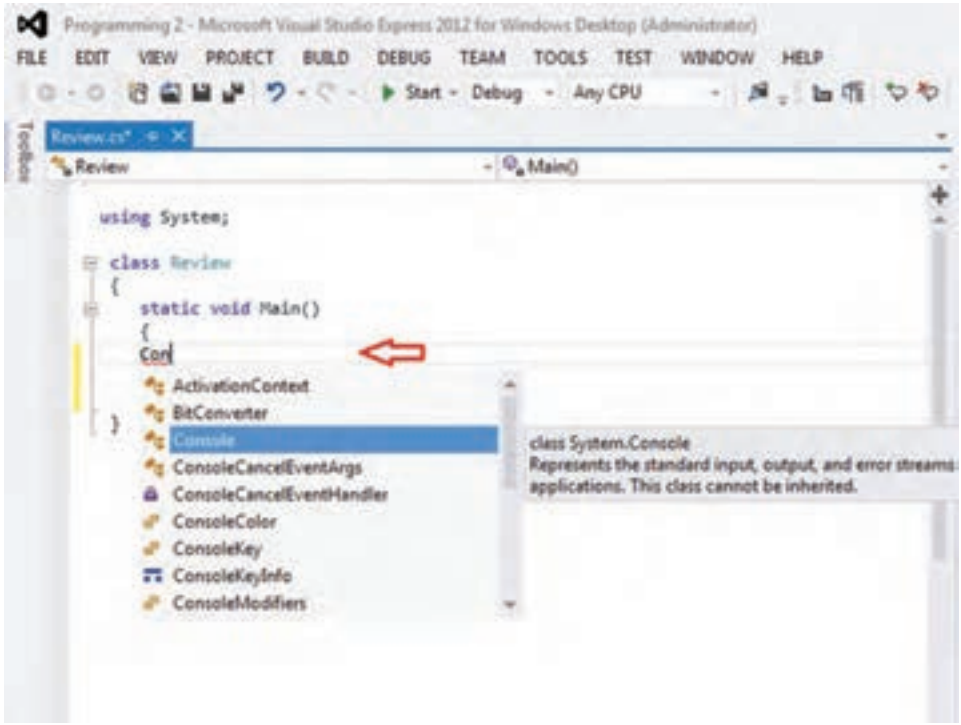
پس از مشخص کردن نام پروژه و مسیری که در آن پروژه ذخیره می‌شود، بر روی OK کلیک می کنیم و در پنجره ویرایشگر برنامه، دستورات را در داخل()Main می‌نویسیم.

محیط VS بسیار قدرتمند و دارای ابزارهای مختلفی برای نوشتن، اجرا و عیب‌یابی برنامه است. یکی از امکانات ارزشمند در هنگام نوشتن برنامه، IntelliSense[1] است که در هنگام تایپ دستورات برنامه، با نوشتن چند حرف از نام دستور (نام کلاس، متد،...)، منو یا لیستی ظاهر می‌شود که در آن نام متدها و دستورات مرتبط نشان داده شده‌اند. در این لیست می‌توانید با کلیدهای فلش بالا و پایین، حرکت کنید و سپس کلمه مورد نظر خود را برگزینید. با زدن کلید Tab یا کلید فاصله[2]، دستور مورد نظر، در برنامه به طور خودکار قرار می‌گیرد و لازم نیست بقیه نام دستور را تایپ کنید. این امکان علاوه‌بر اینکه سبب می‌شود سرعت تایپ برنامه افزایش یابد، باعث می‌شود دستورات با املای صحیح و بدون خطا نوشته شود.

۱ــ Intelligent Sense

۲ــ Space Bar

همان‌طور که در شکل ۷ ــ ۱ مشاهده می‌کنید، برنامه‌نویس قصد تایپ نام کلاس Console را دارد که به محض تایپ حرف C، لیست Intelli Sense به طور خودکار[1] باز می‌شود و نام‌هایی را نشان می‌دهد که با حرف C شروع می‌شوند و اگر برنامه نویس مشغول تایپ دو حرف دیگر شود، در لیست Intellisense به نام کامل Console می‌رسد و می‌تواند بقیه کلمه را تایپ نکند و تنها با زدن کلید Tab یا فاصله، کلمه Console را در برنامه بگنجاند.



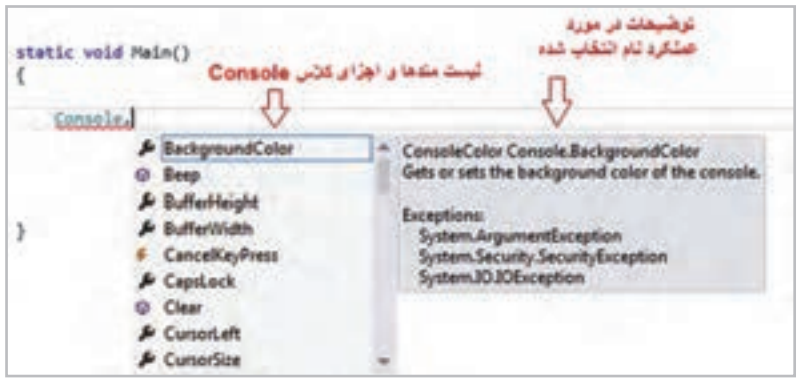
شکل ۷ ــ ۱ــ پنجره کدنویسی

لیست کلماتی که به وسیلهٔ IntelliSense نشان داده می‌شود، مرتبط با کلمه‌ای است که برنامه‌نویس در حال تایپ آن است. مثلاً هنگامی که برنامه نویس علامت نقطه را پس از کلاس Console تایپ می‌کند، لیستی از متدهای این کلاس ظاهر می‌شود.

۱ــ ممکن است این امکان غیرفعال شده باشد که در این‌صورت به توضیح ارائه شده در نکته شماره ۲ مراجعه کنید.

نکته

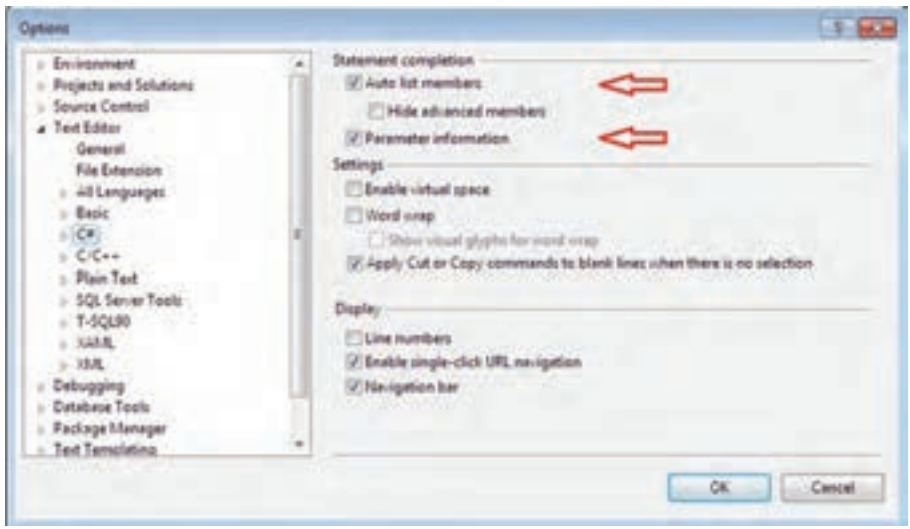
لیست مربوط به IntelliSense را در داخل متن برنامه، می‌توانید با زدن کلید ترکیبی CTRL+SPACE ظاهر کنید.



شکل ۸ــ۱ــ منوی IntelliSense

نکته

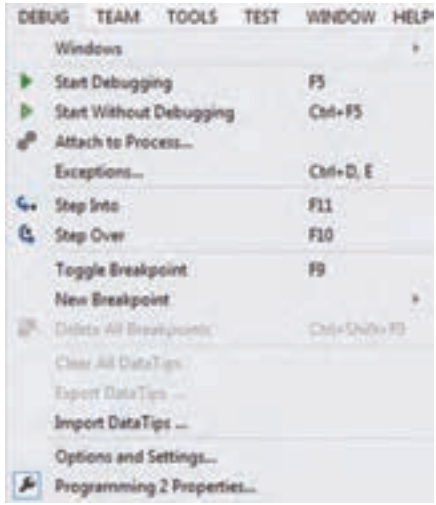
اگر امکان IntelliSense فعال نیست و از آن نمی‌توانید استفاده کنید، باید از طریق منوی Tools گزینه Options وارد قسمت Text Editor شوید و سپس در بخش C#، گزینه Auto list members و Parameter information را فعال کنید.



شکل ۹ــ۱ــ فعال یا غیر فعال کردن امکان IntelliSense

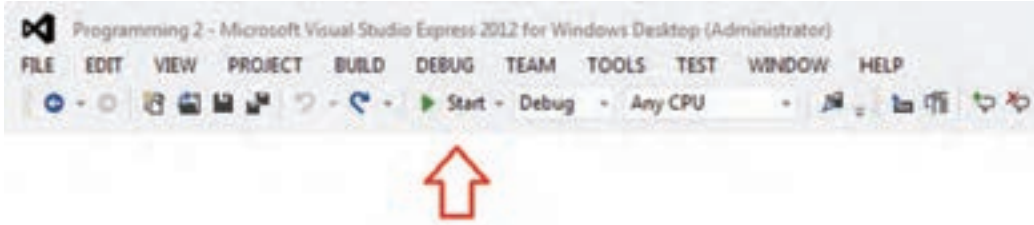
پس از پایان تایپ دستورات، برنامه را ترجمه و اجرا کنید. بدین منظور منوی Debug را باز کرده، و از آن می‌توانید گزینه‌های Start Debugging و یا Start Without Debugging را انتخاب و یا از کلیدهای میانبر آنها CTRL+F5 و یا F5 برای اجرای برنامه استفاده کنید.

سؤال: تفاوت این دو گزینه در اجرای برنامه چیست؟



شکل ۱۰ــ۱ــ منوی Debug برای اجرای برنامه

همچنین می‌توانید در نوار ابزار بالای صفحه، روی کلید Start کلیک کنید.



شکل ۱۱ــ۱ــ گزینه Start در نوار ابزار محیط VS

در دنباله این فصل در قسمت کار درکارگاه عملیات ذکر شده را در محیط VS انجام خواهید داد.

۳ــ۱ــ یادآوری دستورات زبان برنامه نویسی C#

۱ــ۳ــ۱ــ دستورات ورودی و خروجی

همان‌طور که به خاطر دارید برای دریافت داده‌ها از متد ReadLine() و همچنین برای نمایش اطلاعات بر روی صفحه نمایش از متد WriteLine() استفاده می شود.

سؤال: متد دیگری برای نمایش اطلاعات روی صفحه نمایش، متد Write () است این متد با متد WriteLine() چه تفاوتی دارد؟

۲ــ۳ــ۱ــ تبدیل نوع داده : به خاطر دارید که برای نگهداری داده‌ها در برنامه، از متغیرها استفاده می‌شود. هر متغیر مانند ظرفی است که می‌تواند مقداری را در خود نگهداری کند. اندازه و گنجایش هر متغیر توسط نوع داده معین می‌شود. انواع مختلفی از داده‌ها در زبان C# وجود دارند که برای اعداد صحیح، اعداد اعشاری، حروف و کلمات مورد استفاده قرار می‌گیرند. مانند : long ،int ،double ،char ،string.

قبل از اینکه بخواهیم از یک متغیر در برنامه استفاده کنیم، بایستی نام متغیر، گنجایش آن و نوع داده‌ای را که می‌تواند نگهداری کند مشخص کنیم.

کار در کارگاه ۱

مثال ۲ــ۱ : برنامه‌ای بنویسید که سن شما را بر حسب سال نشان دهد.

با توجه به اینکه سن عدد کوچکی است می‌توانیم آن را از نوع byte تعریف کنیم. سپس با پیامی مقدار این متغیر را چاپ می کنیم.

```
using System;
class Review
{
    static void Main ()
    {
        byte age = 16;
        Console. WriteLine ("My age was " + age);
        }
}
```

برنامه ۲ــ۱ــ نمایش سن

در برنامه بالا علاوه بر نمایش یک رشته مقدار یک متغیر یعنی عدد ۱۶ نیز نشان داده می‌شود. حتماً به یاد دارید که علامت + هنگامی که حداقل با یک رشته به کار می‌رود عمل الحاق رشته‌ها را انجام می‌دهد. بنابراین محتوای متغیر به یک رشته تبدیل شده و به رشته "My age Was" الحاق می‌شود.

سؤال: متغیر نوع byte برای نگهداری اعداد صحیح کوچک استفاده می‌شود. آیا محدوده نگهداری آنها را به خاطر دارید؟

برای نمایش سن شما در سال جاری، تغییراتی در برنامه بالا اعمال می‌کنیم که علاوه بر سن سال قبل، سن سال جاری را نیز نمایش دهد. بدین منظور بعد از دو دستور بالا، یک واحد به متغیر age، اضافه می‌کنیم و مجدداً محتوای متغیر را به همراه یک پیام مناسب چاپ می‌کنیم.

```
using System;
class Review
{
    static void Main ()
    {
        byte age = 16;
        Console. WriteLine ("My age was " + age);
        age = age + 1;
        Console.WriteLine ("Now my age is " + age);
    }
}
```

برنامه ۳ــ۱ــ نمایش سن سال جاری

اما با دقت در متن برنامه متوجه می‌شویم که مترجم از دستور افزایش مقدار متغیر به اندازهٔ یک واحد خطا می‌گیرد. اگر علامت ماوس را روی خط قرمز رنگ ببریم توضیح خطا به صورت زیر نوشته می‌شود :

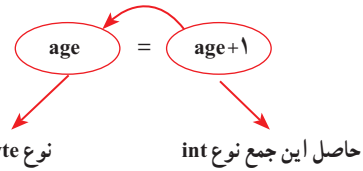
```
age = age + 1;
Cannot implicitly convert type 'int' to 'byte'. An explicit conversion exists (are you missing a cast?)
```

شکل ۱۲ــ۱ــ خطای صادر شده

سؤال: منظور از خطای قبل چیست؟ با استفاده از عملگرهای افزایشی و کاهشی برنامه را بازنویسی کنید.

اکنون می‌خواهیم علت بروز مشکل در دستور; age = age +۱ در برنامه بالا را بررسی کنیم. عملگر + دارای دو عملوند است. یکی از عملوندها، متغیر age از نوع byte و عملوند دیگر عدد ۱ که عددی صحیح است و مترجم به طور پیش فرض اعداد صحیح را از نوع int در نظر می‌گیرد.

ظرفیت داده int، چهار بایت و ظرفیت داده byte یک بایت است و نمی‌توان محتوای int را به‌طور مستقیم درون داده‌های کوچک‌تر از خودش جای داد. بنابراین حاصل عمل جمع، عددی از نوع صحیح int خواهد بود و هنگامی که بخواهید یک مقدار int را در یک متغیر کوچک از نوع byte قرار دهید این مقدار بزرگ‌تر از گنجایش ظرف مورد نظر شما است و بنابراین مترجم خطا اعلام می‌کند.



برای برطرف کردن اشکال، می‌توانید از مترجم بخواهید که حاصل جمع را به نوع byte تبدیل کند تا در متغیر age جای بگیرد. برای این منظور از تبدیلِ نوع[۱] استفاده می‌کنیم (برنامه ۴ــ۱).

شکل کلی تبدیل نوع داده چنین است :

عبارت مبدأ (نوع داده مقصد)

```
using System;
class Review
{
    static void Main ( )
    {
        byte age = 16;
        Console. WriteLine ("My age was " + age);
        age = (byte) (age + 1);
        Console. WriteLine ("Now My age is " + age);
    }
}
```

برنامه ۴ــ۱ــ نوع دیگری از برنامه نمایش سن سال جاری

۱ــ Type Cast

نکته

در تبدیل نوع داده باید به نوع داده‌های مقصد و مبدأ توجه کرد. نوع داده مقصد باید بتواند مقدار دادهٔ مبدأ را در خود جای دهد، زیرا تبدیل نوع داده دارای قوانینی است که در پیوست ۱ جدول مربوط به آن آورده شده است.

مثال ۳ــ۱ : برنامه‌ای بنویسید که با استفاده از تبدیل نوع، عدد را به صورت کاراکتر و همچنین کاراکتر را به صورت عدد تبدیل کرده و نمایش دهد.

الگوریتم یا روش انجام کار : در این برنامه، در متد () Write محتوای متغیر aNumber را که عدد صحیح 67 قرار داده‌ایم، با استفاده از تبدیل نوع (char)، به عنوان کد اسکی یک کاراکتر در نظر گرفته می‌شود که در این صورت، معادل با حرف انگلیسی بزرگ C است.

همچنین محتوای متغیر کاراکتری ch و کاراکتر 'C' با استفاده از تبدیل نوع (int) به یک عدد صحیح که همان کد کاراکتر است تبدیل شده، و با استفاده از متد () WriteLine نمایش داده می‌شوند.

```
using System;
class Review
{

    static void Main ( )
    {
       int aNumber = 67;
       Console. Write ( (char) aNumber);

       char ch = "# ";
       Console. WriteLine (ch);

       Console. WriteLine ("Code of C letter :  " + (int) ' C ');
       Console. WriteLine ("Code of  # :  " + (int) ch);
    }
}
```

برنامه ۵ــ۱ــ نمایش کدهای اسکی کاراکترهای C و #

مثال ۴ــ۱ : برنامه‌ای بنویسید که نام کاربر را سؤال نماید و یک پیام خوشامدگویی به وی اعلام کند.

الگوریتم یا روش انجام کار : در این برنامه برای دریافت نام کاربر، از متد ReadLine() استفاده می‌کنیم و نام کاربر را در یک متغیر ذخیره می‌نماییم. نوع متغیر باید از نوع رشته‌ای String باشد که بتوان یک رشته یا نام کاربر را در آن نگهداری کرد. سپس به وسیلهٔ متد WriteLine () نام کاربر به همراه پیام خوشامدگویی را در خروجی نمایش می‌دهیم. به برنامه زیر دقت کنید.

```
using System;
class Review
{
    static void Main ( )
    {
        Console. Write ("Your Name: ");
        string userName = Console . ReadLine ( );
        Console. WriteLine ("Hi " + userName + " ,Welcom Back to C#! ");
    }
}
```

برنامه ۶ــ۱ــ خوشامد گویی به کاربر

در داخل پرانتز در متد WriteLine () برنامه ۶ــ۱، از علامت + برای اتصال و الحاق رشته‌ها استفاده شده است. عملگر + هنگامی‌که با رشته‌ها و یا متغیرهای رشته‌ای به کار می‌رود، عمل الحاق یا کنار هم قرار دادن مقدار رشته‌ها را انجام می‌دهد.

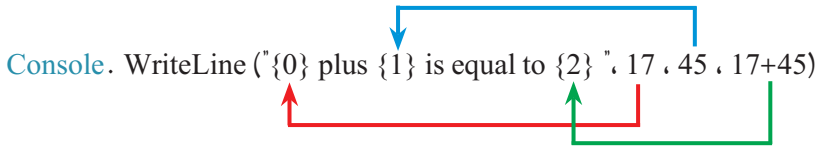
۳ــ۳ــ۱ــ الگوی[۱] جای گذاری در رشته : به جای استفاده از عملگر + می‌توان در متد WriteLine () از الگوی جای گذاری مطابق با روش زیر استفاده کرد.

```
Console. WriteLine ("Hi " {0}, Welcome back to CSharp!" , userName);
```

در این روش، به جای {0}، مقدار متغیر userName قرار می‌گیرد. در واقع {0} جای قرارگیری متغیر را در داخل رشته نشان می‌دهد. اگر بخواهیم مقدار متغیر یا عبارت دیگری را نیز در

۱ــ Formatting

داخل رشته معین کنیم شماره‌های دیگری را بین علامت {} قرار می‌دهیم. مثلاً برای جای عبارت دوم عدد 1 در بین علامت {} استفاده می‌کنیم. مانند دستور زیر که در آن مقدار سه عبارت در رشته با شماره‌های 0 و 1 و 2 مشخص شده است :



علاوه بر محل قرارگیری یک متغیر یا عبارت در یک رشته، می‌توانیم الگو و طریقه نمایش عبارت و همچنین تراز چپ و راست را نیز در صورت نیاز معین کنیم. منظور از عدد تراز تعداد فضای خالی است که در صفحه نمایش، جهت نمایش مقدار متغیر یا عبارت اختصاص داده می‌شود. به طور کلی الگوی جای گذاری چنین است :

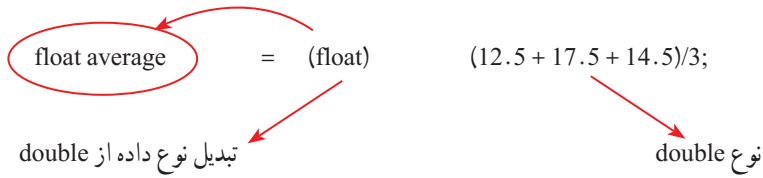
{الگوی نمایش : عدد تراز، شماره}

اگر عدد تراز منفی باشد، مقدار موردنظر در فضای اختصاص یافته چپ چین می‌شود و اگر عدد تراز مثبت باشد، مقدار موردنظر در فضای اختصاص یافته راست چین می‌شود. الگوی نمایش از تعدادی کاراکتر تشکیل می‌شود که هر یک نحوهٔ نمایش داده را مشخص می‌کند. مثلاً برای نمایش اعداد اعشاری با دو رقم اعشار(مثل میانگین نمره) از کاراکتر F استفاده می‌شود. در پیوست۲، لیست کامل کاراکترهای الگوی نمایش برای انواع داده‌ها وجود دارد.

مثال ۵ــ۱ : در برنامه ۷ــ۱ میانگین سه نمره محاسبه شده است و سپس به دو صورت معمولی و همچنین با دو رقم اعشار نشان داده شده است. در هر دو تراز راست با فضای ۱۰ کاراکتر در نظر گرفته شده است.

```
using System;
class Program
{

    static void Main ( )
    {
       float average = (float) (12.5 + 17.5 +14.5) /3;
       Console. Writeline ("Average = {0,10}" , average);
       Console. Writeline ("Average = {0,10: F}" , average);

    }
}
```

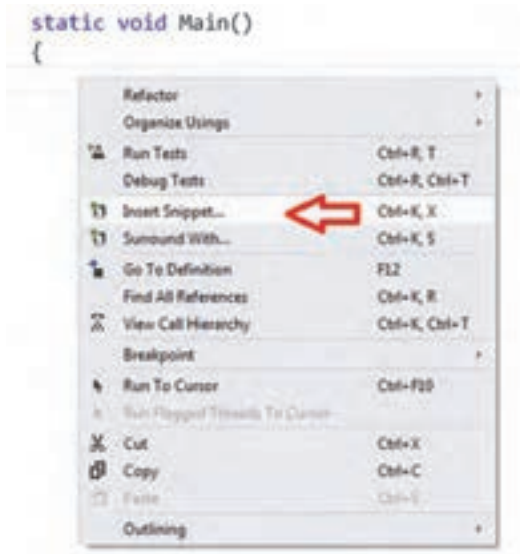
برنامه ۷ــ۱ــ نمایش عدد اعشاری با دو رقم اعشار و با تراز راست



به خاطر دارید پیش فرض برای اعداد اعشاری، نوع double است و در این خط از برنامه به جای استفاده از پسوند F یا f از تبدیلِ نوع استفاده شده است. این عمل با استفاده از کلمه کلیدی (float) در این خط از برنامه انجام شده است.

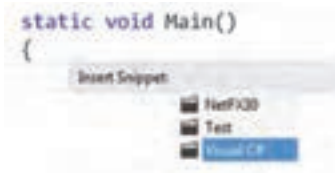
۴ــ۳ــ۱ــ یادآوری دستورات شرطی

۱ــ۴ــ۳ــ۱ــ دستور if–else : از دستور if–else برای تصمیم‌گیری در اجرای دستورات استفاده می‌شود. در جلوی دستور if یک عبارت منطقی ساده و یا مرکب به عنوان شرط نوشته می‌شود، درصورت درست بودن شرط، دستور یا دستورات بعد از این عبارت اجرا می‌شود، در غیر این صورت دستور یا دستورات پس از else اجرا می‌شود. زبان‌های برنامه نویسی، امکاناتی را فراهم کرده‌اند که نیاز به حفظ کردن تمامی جزئیات و تایپ دستورات نیست، به عنوان مثال می‌توانید در پنجره کد نویسی کلیک راست کرده و از منویی که ظاهر می‌شود گزینه Insert Snippet را انتخاب کنید (شکل ۱۳ــ۱).



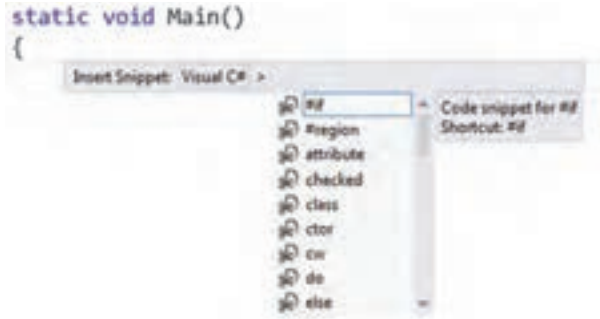
شکل ۱۳ــ۱ــ گزینه Insert Snippet

در این صورت منوی دیگری باز می‌شود و از آن گزینه Visual C# را انتخاب کنید (شکل ۱۴ــ۱).



شکل ۱۴ــ۱ــ انتخاب گزینه Visual C# از Insert Snippet

در این صورت منویی ظاهر می‌شود که دستورات و کلمات رزروشدهٔ زبان سی شارپ را نشان می‌دهد (شکل ۱۵ــ۱).



شکل ۱۵ــ۱ــ انتخاب دستورات برنامه نویسی در گزینه انتخاب شده

به عنوان مثال اگر بخواهیم دستور if را در این لیست پیدا کنیم، حرف i را تایپ می‌کنیم تا به سرعت if در لیست یافت شود. اکنون می‌توانید روی آن کلیک کنید (شکل ۱۵ــ۱). بعد از ظاهر شدن دستور انتخابی، می‌توانیم عبارت داخل پرانتز را به دلخواه تغییر دهیم (شکل ۱۶ــ۱).

```
static void Main()
{
    if (true)
    {

    }
```

شکل ۱۶ــ۱ــ ساختار دستور if

کار در کارگاه ۲

مثال ۶ــ۱ : برنامه‌ای بنویسید که نمرهٔ یک درس را از ورودی دریافت کند و تشخیص دهد آیا این نمره معتبر است یا خیر.

الگوریتم یا روش انجام کار : در این مثال باید عددی از کاربر گرفته شود. متد ReadLine() برای دریافت داده‌ها از ورودی استفاده می‌شود. دادهٔ دریافتی به صورت یک رشته به وسیلهٔ این متد تحویل داده می‌شود. مانند "18.75" بنابراین به یک متغیر رشته‌ای برای ذخیرهٔ آن نیاز داریم. اگرچه کاربر عددی به عنوان نمره وارد می‌کند اما این عدد به صورت رشته‌ای از ارقام، ذخیره می‌شود که قابلیت محاسبات عددی ندارد. بنابراین باید این داده رشته‌ای را به یک عدد تبدیل کنیم. متد Parse() نیز برای تبدیل یک رشته به عدد استفاده می‌شود.

اکنون در شرط دستور if مقدار نمره را بررسی می‌کنیم اگر نمره کمتر از صفر یا بیشتر از ۲۰ باشد نمره غیر مجاز است و باید پیام "نمره غیر مجاز است" بر روی صفحه نشان داده شود. شرط دستور if در این مثال، یک شرط مرکب است که از دو قسمت تشکیل شده است و با کلمه "یا" به یکدیگر مربوط می‌شود. در زبان C# از عملگر || به عنوان عملگر "یا" استفاده می‌شود. در صورتی که شرط برقرار نباشد کنترل برنامه به قسمت else انتقال می‌یابد و دستور مربوط به آن اجرا می‌شود.

```
using System;
class Review
{
    static void Main ( )
    {
        float  number ;
        string input ;
        Console. Write ("Enter a mark: ");
        input = Console. ReadLine ( );
        number = float. parse (in put);
        if ( (number<0) | | (number>20) )
            Console. WriteLine ("Invalid Mark!");
        else
            Console. WriteLine ("Mark is in valid range");
    }
}
```

برنامه ۸ــ۱ــ معتبر بودن نمره

مثال ۷ــ۱ : مثال قبل را توسعه دهید به طوری‌که قبولی یا مردودی را نیز معین کند.

الگوریتم یا روش انجام کار : برای تشخیص قبولی یا مردودی، باید یک دستور if به برنامه قبل اضافه کنیم. اگر نمره کمتر از ۱۰ باشد پیام "شما نمره قبولی نیاوردید. تلاش بیشتری کنید." بر روی صفحه نشان داده شود و در غیر این صورت (در قسمت else آن)، پیام "شما قبول شدید." نمایش داده شود.

قطعه برنامهٔ زیر، دستورِ if جدیدی است که باید به برنامه اضافه شود.

```
if ( (number <10)
    Console. WriteLine ("You failed, try harder next. ");
else
    Console. WriteLine ("You passed. ");
```

سؤال: دستور if جدید را در چه قسمتی از برنامه ۸ ــ۱درج کنیم؟

نکته

اگر کاربر در هنگام ورود داده‌های عددی، کاراکتری غیر از عدد وارد کند، متد ()Parse در تبدیل آن به عدد، دچار خطا شده، و برنامه به طور ناگهانی قطع[1] می‌شود و خطایی ظاهر می‌شود. برای جلوگیری از این مشکل سه راه پیش رو داریم :

۱. پس از دریافت رشته حاوی عدد از کاربر، و قبل از استفاده از متد () Parse، کاراکترهای وارد شده را مورد بررسی قرار دهیم و اگر فقط شامل عدد بود، آن را با استفاده از متد Parse به عدد تبدیل کنیم این روش را در فصل مربوط به رشته‌ها مورد بررسی قرار خواهیم داد.

۲. خطایی که ممکن است متد(Parse) در جریان تبدیل اعداد تولید کند، را کنترل و پیش‌بینی کنیم و برای آن دستورات مناسبی در برنامه قرار دهیم. این روش را در فصل کنترل خطا[2] مورد بررسی قرار می‌دهیم.

۳. به جای استفاده از متد() Parse، از متد () TryParse استفاده شود. این متد اگر تبدیل انجام شود مقدار تبدیل شده را به ورودی دوم خود می‌ریزد و اگر موفق به تبدیل عدد نشود خطایی را نمایش نمی‌دهد و برنامه را قطع نمی‌کند بلکه با تولید یک مقدار منطقی False، مشکل در تبدیل عدد را به اطلاع برنامه‌نویس می‌رساند و برنامه‌نویس با دریافت این مقدار متوجه می‌شود که کاربر عدد را صحیح وارد نکرده است. به عنوان مثال اگر قطعه کد زیر را داشته باشیم :

```
int x;
if (int. Tryparse (str, out x) )
    Console. WriteLine ("your number is: " + x);
else
   Console. WriteLine ("wrong number");
```

اگر متغیر str برابر با متن "8" باشد مقدار x برابر 8 می‌شود و مقدار true به عنوان خروجی TryParse، تولید شده و متن "your number is 8" نمایش داده می‌شود. اما اگر مقدار متغیر str برابر با "a" باشد متد TryParse مقدار false را بر می‌گرداند و در نتیجه متن "wrong number" نمایش داده می‌شود.

۱ــ Crash

۲ــ Handling Exceptions

۲ــ۴ــ۳ــ۱ــ عملگر سه تایی یا علامت سؤال : دستور if را در قطعه برنامه زیر در نظر بگیرید :

```
value ++ ;
if (value > 20)
    max = 20 ;
else
    max = value ;
```

دستورات بالا را می‌توان به صورت خلاصه با استفاده از عملگر علامت سؤال به صورت زیر نوشت :

```
value ++ ;
max = (value > 20) ? 20 : value ;
```

شکل کلی به کار گیری این عملگر به صورت زیر است :

مقدار دوم : مقدار اوّل ? (عبارت منطقی)

کامپیوتر در هنگام برخورد با عملگر سه تایی ابتدا حاصل عبارت منطقی را محاسبه می‌کند، اگر حاصل عبارت برابر درست (True) باشد، نتیجه این عملگر برابر مقدار اول خواهد بود و در صورتی که حاصل عبارت منطقی، برابر نادرست (False) باشد، حاصل این عملگر برابر مقدار دوم است.

سؤال: به نظر شما چرا عملگر ? عملگر سه تایی نامیده می‌شود؟

مثال ۸ــ۱ : در برنامه ۹ــ۱ به جای دستور if از عملگر سه تایی استفاده کنید.

```
using System;
class Review
{
    static void Main ( )
    {
        int number;
        string input , output ;

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

        if (number % 2 = = 0)
            output = "Even number";
        else
            output = "Odd number";
        Console. WriteLine ("{0} is an {1}." , number, output) ;
    }
}
```

برنامه ۹ــ۱ــ تشخیص زوج و فرد بودن اعداد

۳ــ۴ــ۳ــ۱ــ دستور switch : برای مواقعی که بخواهیم حالت‌های مختلف یک عبارت را بررسی کرده و بر اساس آن دستورات را اجرا کنیم از دستور switch استفاده می‌کنیم.

کار در کارگاه ۳

مثال ۹ــ۱ : برنامهٔ ۱۰ــ۱ به وسیلهٔ یکی از هنرجویان نوشته شده است. کار این برنامه را بررسی می‌کنیم :

الگوریتم یا روش انجام کار : در این مثال ابتدا یک عدد از کاربر دریافت شده و در یک متغیر ذخیره می‌شود سپس اجرای برنامه وارد دستور switch خواهد شد و عدد دریافتی را با caseهای مختلف از ۱ تا ۳ مقایسه می‌کند اگرعدد وارد شده با هرکدام از caseها برابر باشد دستور مربوط به case اجرا

می‌شود و رنگ زمینه تغییر می‌کند. اما اگر عدد وارد شده با هیچ یک از دستورات case برابر نبود، دستور default اجرا خواهد شد. دستورات مذکور در داخل یک حلقه for قرار دارد که دوبار تکرار می‌شود.

سؤال: آیا می‌توانید برنامه را به صورت دیگری بهتر از آنچه که نوشته شده بنویسید؟

```
using System;
class Program
{
    static void Main ( )
    {
        for (int i = 1; i < = 2; i++)
        {
            int n = int. Parse(Console.ReadLine () );
            switch (n)
            {

                case 1:
                    Console. BackgroundColor = ConsoleColor.Red;
                    Console.Clear();
                    break;
                case 2:
                    Console. BackgroundColor = ConsoleColor.Green;
                    Console.Clear();
                    break;
                case 3:
                    Console. BackgroundColor = ConsoleColor.Blue;
                    Console.Clear();
                    break;
                default:
                    Console. WriteLine ("ERROR");
                    break;
            }
        }
    }
}
```

برنامه ۱۰ــ۱ــ تغییر رنگ صفحه کنسول

۵ــ۳ــ۱ــ یادآوری دستورات تکرار یا حلقه

در مواقعی که می‌خواهیم یک یا چند دستور، بیش از یک بار انجام شوند، به جای اینکه آنها را چندین بار تایپ و یا کپی کنیم، می‌توانیم از دستورات تکرار یا حلقه استفاده نماییم. در کتاب برنامه‌سازی ۱ با انواع دستورات تکرار مانند while، do ــ while و دستور for آشنا شدید. در این قسمت با ذکر چند مثال، انواع دستورات تکرار را یادآوری می‌کنیم.

۱ــ۵ ــ۳ــ۱ــ دستور while : برای مواقعی که یک یا چند دستور باید تا زمانی که حاصل یک عبارت منطقی درست است اجرا شوند، از دستور while استفاده می‌شود.

کار در کارگاه ۴

مثال ۱۰ــ۱ : برنامه‌ای بنویسید که یک عدد را از کاربر دریافت کند و اعداد کمتر از آن تا عدد یک را نشان دهد. مثلاً اگر عدد ۱۰ وارد شد، اعداد ۱۰، ۹، ... تا عدد ۱ به صورت کاهشی یا نزولی نمایش داده شوند.

الگوریتم یا روش انجام کار : در این مثال، ابتدا یک عدد از کاربر دریافت می‌کنیم و آن را در یک متغیر ذخیره می‌نماییم. سپس باید محتوای این متغیر را روی صفحه نشان دهیم و پس از آن، یک واحد از متغیرکم کنیم و مجدداً عملیات نمایش محتوای متغیر و کاهش مقدار آن باید تکرار شود تا زمانی که به عدد یک برسیم.

برای این منظور از دستور تکرار while استفاده می‌کنیم. برنامه زیر را مشاهده کنید :

```
using System;
class Review
{
    static void Main ( )
    {
        int  number;
        string input;
        Console. WriteLine (" Counting down to one" ) ;
        Console. Write (" Enter a number (1–100): " ) ;
        input = Console. ReadLine ( );
        number = int. Parse (input);
        while (number > 0)
```

```
        {
            Console. WriteLine (number);
            number – –;
        }
    }
}
```

برنامه ۱۱ــ۱ــ نمایش اعداد متوالی به صورت کاهشی تا صفر

سؤال: اگر علامت‌های آکولاد باز و بسته در حلقهٔ بالا حذف شوند پس از اجرای برنامه، چه اتفاقی می‌افتد؟

در قطعه برنامه زیر به دستورات تکرار شونده توجه کنید :

```
while (number > 0)
{
    Console. WriteLine (number);
    number – –;
}
```

اگر مایل باشید، می‌توانید دستورات داخل حلقه while را در یک دستور خلاصه کنید. یعنی :

```
while (number > 0)
        Console. WriteLine (number – –);
```

در داخل پرانتز متغیر number به همراه عملگر کاهشی نوشته شده است. چون عملگر – –، بعد از نام متغیر ذکر شده است، ابتدا مقدار متغیر چاپ می‌شود و سپس مقدار آن یک واحد کاهش می‌یابد.

مثال ۱۱ــ۱ : برنامه‌ای بنویسید که یک عدد را از کاربر دریافت کند و تعداد ارقام آن را شمارش و نشان دهد.

الگوریتم یا روش انجام کار : ابتدا باید داده‌ای را از کاربر دریافت کنیم و با توجه به اینکه متد() ReadLine داده دریافتی را به صورت یک رشته تحویل می‌دهد، برای شمارش تعداد ارقام

عدد، به دو روش زیر می توانیم عمل کنیم :

۱ــ تعداد کاراکتر رشته دریافتی را شمارش کنیم که همان ارقام عدد هستند[۱]. در فصل سوم با این روش تعداد ارقام عدد را حساب می‌کنیم.

۲ــ مانند مثال‌های قبلی، رشته دریافتی را به عدد تبدیل کرده، و سپس با استفاده از تقسیم‌های متوالی به ۱۰ تعداد دفعات تقسیم را می‌شماریم که همان تعداد ارقام عدد است. در برنامه ۱۲ــ۱ از این روش استفاده شده است.

```
using System;
class Review
{
    static void Main ( )
    {
        int  number , remainder, digits=0;
        string input;
        Console. Write (" Enter a number: " ) ;
        input = Console. ReadLine ( );
        number = int. Parse (input);
        while (number > 0)
        {
            remainder = number % 10 ;
            digits ++ ;
            number = number / 10 ;
        }
        Console. WriteLine ("The number has {0} digits" , digits);
    }
}
```

برنامه ۱۲ــ۱ــ نمایش شمارش ارقام

سؤال: آیا این برنامه برای اعداد منفی نیز کار می‌کند؟ اگر پاسخ خیر است با کمترین تغییر در شرط دستور while کاری کنید که برای اعداد منفی نیز برنامه به درستی کار کند.

۱ــ البته علامت عدد و یا فاصله را باید در نظر داشت و به‌عنوان تعداد ارقام آنها را محسوب نکرد.

۲ــ۵ ــ۳ــ۱ــ دستور for : برای اجرای دستور یا دستورات به تعداد دفعات معین، از دستور for استفاده می‌کنیم.

مثال ۱۲ــ۱ : در برنامه مثال قبل (چاپ اعداد به صورت کاهشی)، به جای دستور while از دستور for استفاده کنید.

مثال ۱۳ــ۱ : برنامه‌ای بنویسید که نمرات درس برنامه سازی ۲ مربوط به یک کلاس ۱۰ نفری را دریافت کند و مجموع و میانگین نمرات را حساب کند.

الگوریتم یا روش انجام کار : در این مثال چون عمل دریافت نمرات و محاسبه مجموع باید ده بار تکرار گردد از یک حلقه استفاده می‌کنیم که در داخل حلقه عملیات مربوطه را قرار می‌دهیم. اگرچه از هر نوع حلقه می‌توانیم برای این منظور استفاده کنیم اما در این مثال حلقه for را به کار می‌گیریم.

```
using System;
class Review
{
    static void Main ( )
    {
        float number, total =0, average;
        string input;
        for (int i = 0; i < 10; i++)
        {
            Console. Write (" Enter a mark: " ) ;
            input = Console. ReadLine ( );
            number = float. Parse (input);
            total = total + number;
        }
        average = total / 10 ;
        Console. WriteLine ("Total : " + total);
        Console. WriteLine ("Average : " + average);
    }
}
```

برنامه ۱۳ــ۱ــ محاسبه مجموع و معدل ۱۰نمره

۴ــ۱ــ استفاده از مقدار ثابت[۱] در برنامه

مثال ۱۴ــ۱ : برنامه مثال قبل را برای یک کلاس با ۱۵ نفر دانش‌آموز تغییر(توسعه) دهید.

الگوریتم یا روش انجام کار : در این حالت لازم است به جای دریافت ۱۰ عدد، پانزده عدد دریافت کنیم، بنابراین حلقه تکرار for باید به تعداد ۱۵ بار اجرا شود. برای این منظور کافی است عدد ۱۰ را در حلقهٔ for را به ۱۵ تغییر می‌دهیم. همچنین باید مجموع نمرات را به تعداد نمرات دریافتی یعنی عدد ۱۵ تقسیم کنیم. به این ترتیب باید در انتهای برنامه مجموع نمرات (total) را به عدد ۱۵ تقسیم کنیم.

اما فرض کنید که برنامه نویس فراموش کند عدد ۱۰ دوم را تغییر دهد در این صورت با اجرای برنامه پانزده نمره دریافت می‌شود و مجموع نمرات به درستی حساب می‌شود اما میانگین نمرات به اشتباه حساب می شود. برای جلوگیری از چنین اشتباهاتی و همچنین توسعه راحت برنامه می‌توان در ابتدای کلاس یک شناسه به عنوان یک ثابت تعریف کرد و عدد ۱۰ را به آن نسبت داد سپس در سرتاسر برنامه هر جا که به عدد ۱۰ نیاز بود از آن شناسه استفاده کرد.

برای تعریف شناسه یا نام ثابت از کلمه کلیدی const استفاده می‌شود. معمولاً نامی‌که برای اعداد ثابت تعریف می‌شود با حروف بزرگ نوشته می‌شود تا در برنامه مشخص باشد که این شناسه، یک مقدار ثابت است. نحوهٔ تعریف ثابت‌ها، مانند تعریف متغیرها می‌باشد با این تفاوت که در ابتدای تعریف آنها کلمه const قرار دارد.

```
const int SIZE = 15;
```

برنامه ۱۴ــ۱ را در نظر بگیرید :

```
using System;
class Review
{
    const int SIZE = 15;  ⇦
    static void Main ( )
    {
        float number, total =0, average;
        string input;
        for (int i = 0; i < SIZE; i++)  ⇦
        {
```

۱ــ Constant

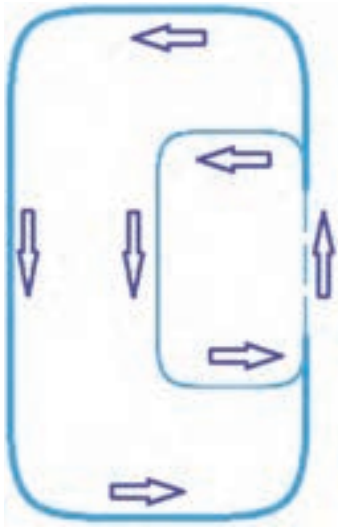
```
            Console. Write (" Enter a mark: ") ;
            input = Console. ReadLine ( );
            number = float. Parse (input);
            total = total + number;
        }
        average = total / SIZE ;
        Console. WriteLine ("Total : " + total);
        Console. WriteLine ("Average : " + average);
    }
}
```

برنامه ۱۴ــ۱ــ امکان تغییر راحت برنامه برای دریافت ۱۵ نمره

سؤال: اگر بخواهید برنامهٔ ۱۴ــ۱ برای محاسبه میانگین نمرات پنج دانش‌آموز به کار رود، چه تغییری در آن اعمال می‌کنید؟

۵ــ۱ــ حلقه‌های متداخل (تو در تو)

در بعضی از مواقع، یک دستور تکرار را در داخل دستور تکرار دیگری به کار می‌بریم. به عبارت دیگر هنگامی که در داخل یک حلقه، حلقه دیگری قرار داشته باشد، حلقه‌های تودرتو یا متداخل[۱] نامیده می‌شوند.



شکل ۱۷ــ۱ــ پیست دومیدانی مشابه حلقه‌های تودرتو

حلقه‌های تودرتو را مانند پیست دومیدانی در نظر بگیرید، حال دونده‌ای را در نظر بگیرید که طبق دستور مربی، باید به ازای هر بار دویدن در پیست بزرگ، مجبور باشد که در داخل پیست کوچک نیز پنج بار طناب بزند. اگر مربی وی از او بخواهد به تعداد ۱۰ بار در پیست بزرگ با همان شرط ذکر شده بدود، در مجموع، چند بار در پیست کوچک طناب زده است؟

۱ــ Nested Loops

مثال ۱۵ــ۱ : در برنامه ۱۵ــ۱، از دو دستور for استفاده شده است که یکی در داخل دیگری قرار دارد و بنابراین حلقه تودرتو را تشکیل می‌دهند.

الگوریتم یا روش انجام کار : در این برنامه، دستور for با متغیر شمارنده i، حلقه بیرونی[۱] و دستور for با متغیر شمارنده j، حلقه داخلی[۲] را تشکیل می‌دهد. کامپیوتر به ازای هر بار اجرای حلقه بیرونی، حلقه داخلی را به طور کامل (پنج بار) انجام می‌دهد.

```
using System;
class Review
{
    static void Main ( )
    {
        for (int i = 1; i <= 10; i++)
        {
            for (int j = 1; j <= 5 ; j++)
                Console. Write (" *" ) ;
            Console. WriteLine ();
        }
    }
}
```

حلقه بیرونی

حلقه داخلی

برنامه ۱۵ــ۱ــ حلقه تو در تو

در حلقه داخلی دستور نمایش رشته "*" قرار دارد، بنابراین با اجرای حلقه داخلی، پنج بار رشته "*" در کنارهم در روی یک خط در صفحه نمایش به صورت زیر چاپ شود :

*****

از آنجا که حلقه بیرونی، ده بار تکرار می‌شود بنابراین طرح زیر، روی صفحه نمایش نقش می‌بندد :

*****
*****
*****
*****

۱ــ Outer Loop

۲ــ Inner Loop

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

سؤال: نقش دستور ;() Console . WriteLine در این برنامه چیست؟

سؤال: برای اینکه در ابتدای هر خط در طرح فوق، شماره سطر را چاپ کنیم، چه باید کرد؟

کار در کارگاه ۵

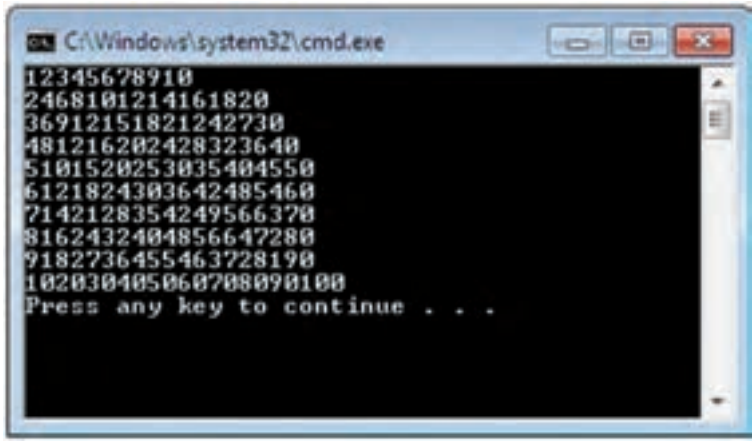
مثال ۱۶ــ۱ : برنامه‌ای بنویسید که جدول ضرب اعداد ۱ تا ۱۰، را روی صفحه نشان دهد.

الگوریتم یا روش انجام کار : برای نمایش جدول ضرب اعداد ۱ تا ۱۰، کافی است در برنامه قبل به جای نمایش رشته "*" حاصل عبارت i*j را محاسبه و روی صفحه چاپ کنیم.

```
using System;
class Review
{
    static void Main ( )
    {
        for (int i = 1; i < = 10; i++)
        {
            for (int j = 1; j <= 10; j++)
                Console. Write ("{0}", i* j) ;
            Console. WriteLine ();
        }
    }
}
```

برنامه ۱۶ــ۱ــ جدول ضرب اعداد ۱ تا ۱۰

نتیجه اجرا یا خروجی برنامه فوق چنین خواهد بود :

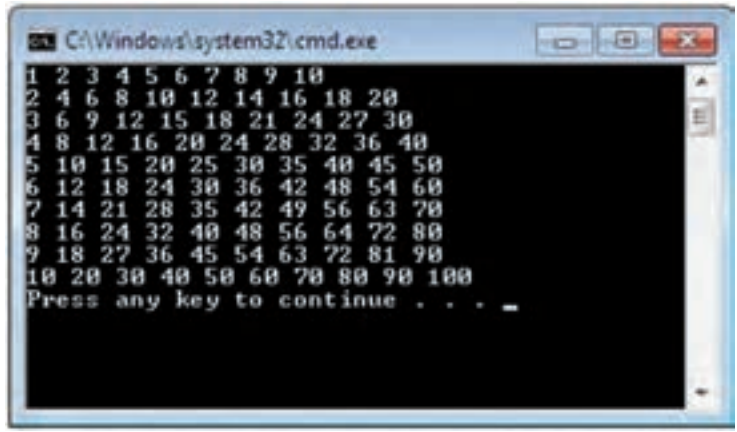


شکل ۱۸ــ۱ــ خروجی برنامه جدول ضرب

همان‌طور که مشاهده می‌کنید هیچ فاصله‌ای بین اعداد وجود ندارد و اعداد قابل تشخیص نیستند. برای رفع این مشکل باید بین هر عدد فاصله چاپ کنیم. یک روش ابتدایی اضافه‌کردن یک یا دو فاصله برای هر عدد است که در متد ( ) Write اعمال کنیم :

```
Console. Write ("{0} ", i*j) ;
```

در این صورت، نتیجه اجرا یا خروجی برنامه چنین خواهد شد :



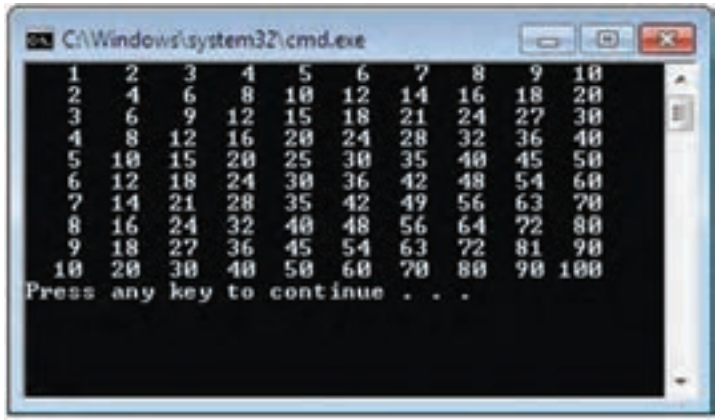
شکل ۱۹ــ۱ــ جدا کردن اعداد جدول ضرب

در این حالت خواندن اعداد بهتر شد و می‌توان اعداد را از یکدیگر تمیز داد. اما شکل کلی خروجی چندان جالب نیست و اعداد در زیر یکدیگر قرار ندارند.

خوشبختانه در زبان C#، راه حل چنین مشکلی در نظر گرفته شده است و امکان شکل یا الگوی[1] نمایش از قبل پیش بینی شده است. اعداد جدول ضرب دو رقمی‌هستند به جز حالت خاص ضرب ۱۰ در ۱۰ است که حاصل برابر ۱۰۰ و یک عدد ۳ رقمی‌است بنابراین اگر برای چاپ هر عدد ۳ مکان و یک مکان نیز به عنوان فاصله در نظر بگیریم، در مجموع فضایی به اندازه ۴ کاراکتر، حداقل فضای لازم برای جداسازی اعداد است. در علامت{0} بعد از عدد صفر، تعداد فضای اختصاص داده شده را به صورت زیر مشخص می‌کنیم :

```
Console. Write ("{0,4}", i*j) ;
```

در این صورت خروجی برنامه چنین خواهد بود :



شکل ۲۰ــ۱ــ خروجی نهایی برنامه جدول ضرب

۶ــ۱ــ کاراکتر \[2]

در زبان‌های برنامه نویسی، از جمله زبان C#، کاراکتر \، یک کاراکتر خاص و معنی‌دار است. هر گاه این کاراکتر، در رشته‌ای دیده شود، کاراکتر بعدی آن اثر و عملکرد خاصی، جدای از شکل ظاهری آن دارد، به عبارت دیگر کاراکتر \، سبب تغییر عملکرد کاراکتر بعدی می‌شود. به همین دلیل

۱ــ Format String

۲ــ Back Slash Character

به این مجموعه کاراکترها که کاراکتر اول \ و دومی کاراکتر دیگری است، دنباله فرار[1] گفته می‌شود.

یکی از کاربردهای این دنباله، نمایش علامت" در یک رشته است. اگر بخواهیم جمله‌ای را همراه با علامت " روی صفحه نمایش چاپ کنیم به مشکل برخورد می‌کنیم. چون معنای معمول این کاراکتر، نشانه شروع و پایان یک رشته است و در وسط یک رشته نمی‌تواند قرار گیرد. لذا باید معنی معمول آن را تغییر دهیم. بنابراین از دنباله "\ برای بیان این کاراکتر استفاده می‌کنیم. :

```
Console. WriteLine ("C# is pronounced \"see Sharp\" " ) ;
                                  ⇧دنباله معنی دار⇧
```

سؤال: خروجی دستور بالا چیست؟

در جدول زیر تعدادی از دنباله‌های معنی‌دار نشان داده شده‌اند.

جدول ۱ــ۱ــ دنباله معنی‌دار

عملکرد	دنباله
ایجاد یک بوق هشدار[2]	\a
حذف یک کاراکتر(Backspace)	\b
ایجاد یک خط خالی(New Line)	\n
ایجاد یک فاصله افقی زیاد tab	\t
ایجاد یک تک کوتیشن (')	\'
ایجاد یک دابل کوتیشن (")	\"
ایجاد یک Back slash (\)	\\

نکته

زبان C# روی حروف کوچک و بزرگ حساس است و این شامل دنباله‌های جدول بالا نیز می‌شود.

۱ــ Escape Sequence ۲ــ Alarm

مثال ۱۷ــ۱ : در برنامه ۱۷ــ۱ دنباله‌های معنی‌دار استفاده شده‌اند. خروجی هر دستور به‌صورت توضیح در جلوی آن نشان داده شده است.

```
using System;
class Escape Sequence
{
static void Main ( )
{
Console. WriteLine ("Escape Sequence") ;              // Enscape Sequence
Console. WriteLine ("\t Escape Sequence") ;           //          Enscape Sequence

Console. WriteLine ("\n Escape Sequence") ;           //
                                                      // Enscape Sequence
Console. WriteLine ("\" Escape Sequence") ;           // " Enscape Sequence
Console. WriteLine ("\' Escape Sequence") ;           // ' Enscape Sequence
Console. WriteLine (" Escape\ b Sequence") ;          // ' Enscape Sequence
Console. WriteLine ("\ \  Escape Sequence") ;         // \ Enscape Sequence
Console. WriteLine ("\ ~ Escape \b Sequence") ;       // Error
```

برنامه ۱۷ــ۱ــ نمایش دنباله معنی دار

خودآزمایی فصل اول

الف) درستی یا نادرستی هر عبارت را تعیین کنید.

۱ــ مقدار یک شناسه ثابت را در طول برنامه می‌توان تغییر داد.

۲ــ برای بررسی حالات مختلف یک عبارت و اجرای دستورات بر اساس آن از دستور switch استفاده می‌کنیم.

۳ــ برای نمایش کاراکترهای خاص مثل " وسط یک رشته از دنباله معنی دار استفاده می‌شود.

ب) جاهای خالی را با عبارت مناسب پر کنید.

۴ــ لیستی از نام متدها و دستورات که هنگام تایپ نمایش داده می‌شود ..... نام دارد.

۵ــ برای نمایش منوی دستورات و کلمات رزرو شده سی شارپ از فرمان ..... استفاده می‌شود.

۶ــ برای تعریف شناسه یا نام ثابت از کلمه کلیدی ..... استفاده می‌شود.

۷ــ IDE ساده و کم حجم، برای نوشتن و ترجمه برنامه‌های سی شارپ..... نام دارد.

ج) به سؤالات زیر پاسخ دهید.

۸ــ تفاوت متد Parse و Try parse را بنویسید.

۹ــ قطعه برنامه زیر را Trace کنید و در ماتریس زیر جای‌گذاری کنید.

```
for (int i=1; i<=4; i++)
 {
    for (int j=4; j>=1; j--)
     Console.WriteLine( (i=j) ? 1: 0);
   Console.WriteLine();
 }
```


۱۰ــ در اجرای قطعه کد زیر پیام خطا داده می‌شود. به نظر شما مشکل چیست؟ با تغییر کد مشکل را برطرف کنید.

```
int x = 14.35;
```

۱۱ــ عملکرد هر یک از سه دستور زیر را بررسی کنید. آیا خروجی این سه دستور یکسان است؟

الف)

```
Console. WriteLine ("Hi" + userName +" , welcome back to CSharp!");
```

ب)

```
Console. WriteLine ("Hi" {0}, welcome back to CSharp!" , userName);
```

ج)

```
Console. WriteLine ("Hi" {0}, {1}", userName, "welcome back to CSharp!" );
```

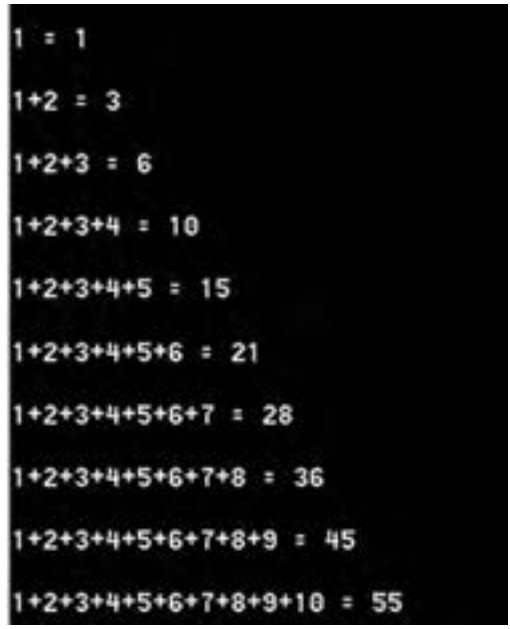
۱ــ برنامه‌ای بنویسید که شکل روبرو را روی صفحه نمایش دهد.

۲ــ برنامه زیر سه عدد را دریافت می‌کند و عدد بزرگ‌تر را نمایش می‌دهد. از عملگر ۳ تایی یا عملگر علامت سؤال به جای دستورات if استفاده کنید.

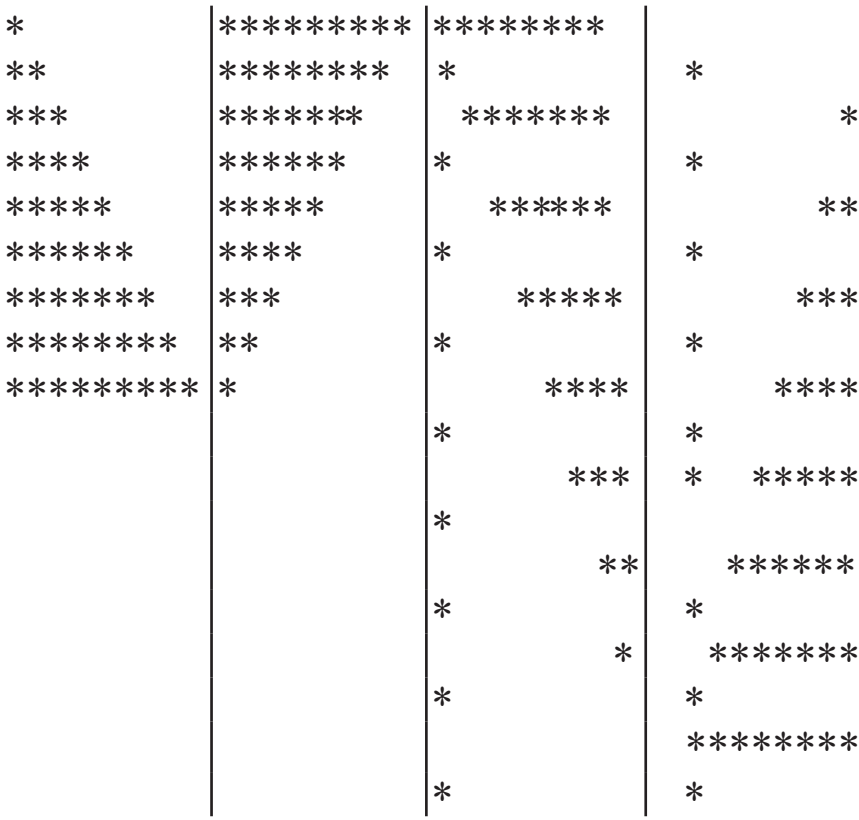
```
using System;
class Review
{
    static void Main ( )
    {
        float a, b, c, max;
        string input;
        Console. WriteLine ("This program finds the maximum number.");
        Console. Write ("Enter first number: ");
        input = Console. ReadLine ( );
        a = float. Parse (input);
        Console. Write ("Enter second number: ");
        input = Console. ReadLine ();
        b = float. Parse (input);

        Console. Write ("Enter third number: ");
        input = Console. ReadLine ();
        c = float. Parse (input);

        max =a;
        if (max < b)
```

```
        max = b;
    if (max < c)
        max = c;
    Console. WriteLine ("Maximum number is " + max);

    }
}
```

۳ــ برنامه‌ای بنویسید که با استفاده از حلقه‌های تو در تو، مجموع اعداد طبیعی را طبق شکل روبرو محاسبه و نمایش دهد.



۴ــ برنامه‌هایی بنویسید که هر یک بتواند یکی از طرح‌های زیر را نمایش دهد.



۵ــ خروجی اجرای دستورات زیر چیست؟

```
string columns = "Column 1\tColumn 2\tColumn 3" ;
string rows = "Row 1\r\nRow 2\r\nRow 3";
Console. WriteLine (columns);
Console. WriteLine (rows);
```

۶ــ تمرین زیر به زبان انگلیسی است، معنی و مفهوم آن را درک کرده، سپس آن را حل کنید.

Lucky Numbers:

a) Write a program to find and print all four_digit numbers of the type abcd, where: a+b = c+d

Example: 2745 is a lucky number, where 2+7 = 4+5 → 9 = 9

(Hint: Use four nested for_loops _ one for each digit as seen below)

```
using System;
class Program
{
    static void Main (string [ ] args)
    {
        Console. WriteLine ("Four Digits Lucky numbers: ");
        for (int a = 1; a<= 9; a++)
            for (int b = 0;  b <= 9; b++)
                for (int c = 0; c <= 9; c++)
                    for (int d = 0; d < = 9; d++)
                        if  (                )
                            Console. WriteLine ("{0}{1} {2} {3}", a, b, c, d);
    }
}
```

b) How many lucky numbers are printed? Add a statement to your program to calculate how many lucky numbers are there.

فعالیت

متن زیر از مستندات MSDN با موضوع حلقه‌های تکرار، برداشت شده است. آن را با کمک هم کلاسی خود ترجمه کنید و به کلاس ارایه نمایید.

A loop is a statement, or set of statements, that are repeated for a specified number of times or until some condition is met. The type of loop you use depends on your programming task and your personal coding preference. One main difference between C# and other languages, such as C++, is the foreach loop, designed to simplify iterating through arrays or collections.

واژگان و اصطلاحات انگلیسی فصل اول

ردیف	واژه انگلیسی	معنی به فارسی
۱	Alarm	
۲	Asterisk	
۳	Back Slash Character	
۴	Command Prompt	
۵	Comment	
۶	Commercial	
۷	Constant	
۸	Counter	
۹	Crash	
۱۰	Disk Operating System	
۱۱	Escape Sequence	
۱۲	Exception	
۱۳	Format String	
۱۴	Handling Exceptions	
۱۵	Inner Loop	
۱۶	Integrated Development Environment	
۱۷	Intelligent Sense	
۱۸	Nested Loops	
۱۹	Open Source	
۲۰	Outer Loop	
۲۱	Source	
۲۲	Space Bar	