

یادآوری و تکمیل مطالب

در کتاب برنامه سازی ۱، با مفهوم برنامه، برنامه نویسی و سطوح مختلف زبان های برنامه نویسی آشنا شدید، سپس در فصل های بعدی کتاب، دستورات مقدماتی و پایه ای زبان برنامه نویسی C# را آموختید. همچنین توانستید برنامه هایی به زبان C# بنویسید که در محیط کنسول سطر فرمان اجرا شوند. اکنون در این فصل، به یادآوری مطالب آموخته شده قبلی می پردازیم و با نوشتن چند برنامه، دستورات پایه ای خوانده شده را بررسی و مرور می کنیم. در بعضی از برنامه ها نیز با ارایه دستورات جدید اطلاعات خود را تکمیل می کنید.

پس از پایان این فصل انتظار می رود که فراگیر بتواند:

- ۱- از محیط SharpDevelop برای نوشتن و ترجمه برنامه های کنسولی استفاده کند.
- ۲- کاربرد متد TryParse را بیان نماید و از آن استفاده کند.
- ۳- با استفاده از عملگر سه تایی، دستور if را بازنویسی نماید.
- ۴- کلمه کلیدی const را برای تعریف ثابت های برنامه به کار بندد.
- ۵- دستورات تکرار متداخل (تودرتو) را در برنامه استفاده نماید.
- ۶- عملکرد دنباله معنی دار را بیان نماید و آن را به کار بندد.

۱-۱- 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 می پردازیم. یکی از راه های ایجاد برنامه به زبان C# نوشتن برنامه در قالب فایل متنی ساده و با ابزاری مانند Notepad است. اگر بخواهیم برنامه ۱-۱ 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 ۱-۱: برنامه ای بنویسید که از لیست زبان های 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.WriteLine ("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 ۱-۱- 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 عیب‌یابی برنامه‌های بزرگ نیز دشوار است.

۲-۱- 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 و عیب‌یابی راحت‌تر برنامه‌ها، طراحی شده است. چندین IDE به وسیله شرکت‌ها و ارگان‌های مختلف طراحی و به بازار عرضه شده است. بعضی از آنها رایگان و به اصطلاح متن‌باز^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 متن‌باز SharpDevelop یا به اختصار #develop^۵ استفاده نمود (شکل ۱-۱).

ویژگی این IDE حجم کم آن نسبت به IDE مایکروسافت می‌باشد. برای استفاده از این IDE ابتدا باید نرم‌افزار Net Framework 4.5 را نصب کنید، سپس نرم‌افزار متن‌باز #develop نصب شود.

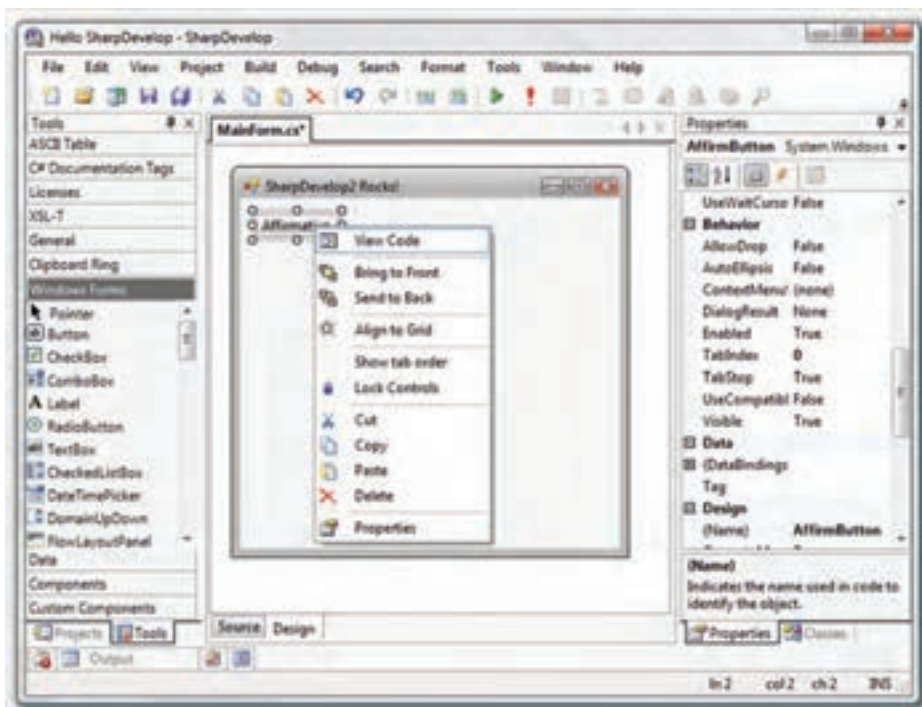
۱- Editor

۲- Integrated Development Environment

۳- Open Source

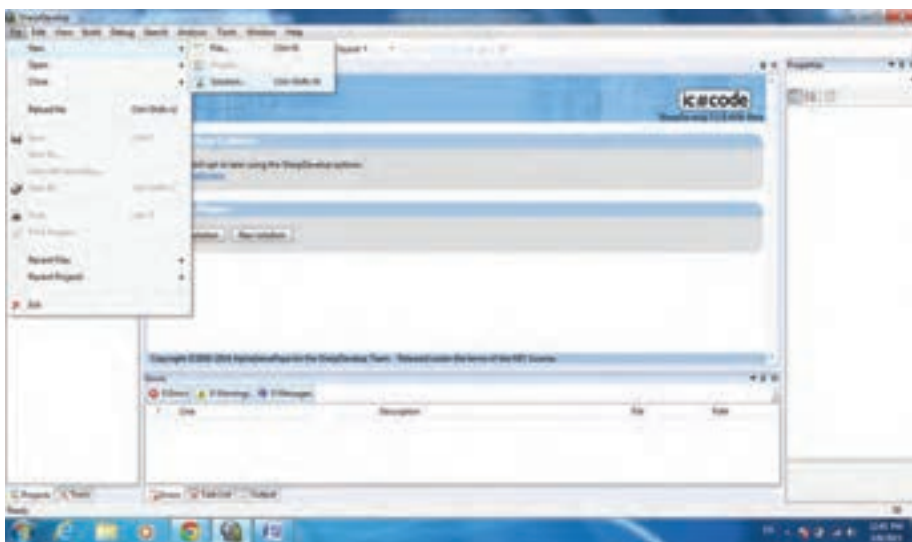
۴- Commercial

۵- # develop (short for SharpDevelop)



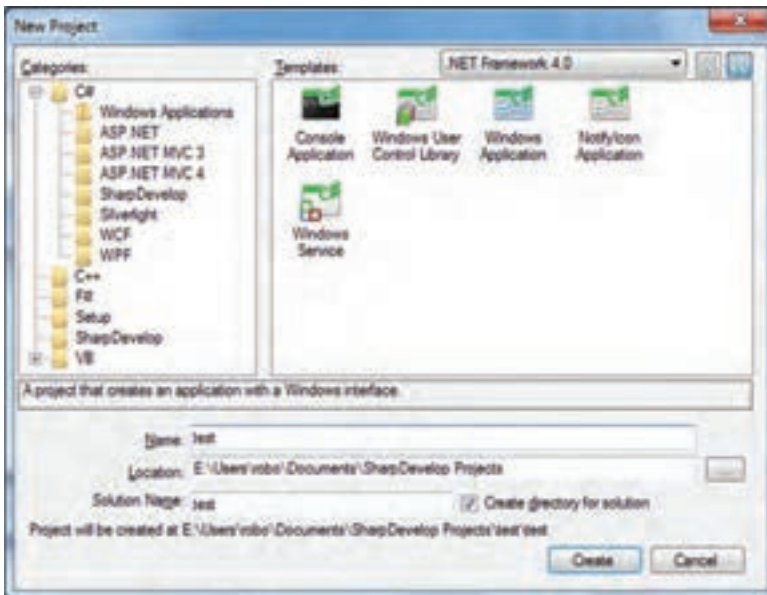
شکل ۱-۱- محیط برنامه نویسی رایگان و متن باز SharpDevelop

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



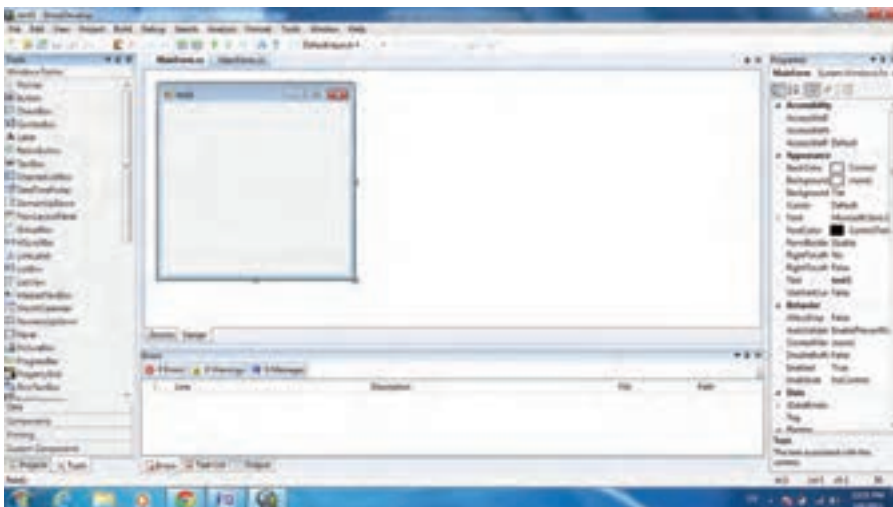
شکل ۱-۲- انتخاب گزینه solution از زیر منوی new

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



شکل ۳-۱- 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.



۵

شکل ۴-۱- فرم اصلی و ابزارهای برنامه نویسی SharpDevelop

در این IDE فرم ابتدایی MainForm نام دارد.

تاکنون از محیط برنامه نویسی ویژوال استودیو اکسپرس^۱ 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^۲ نمایند.



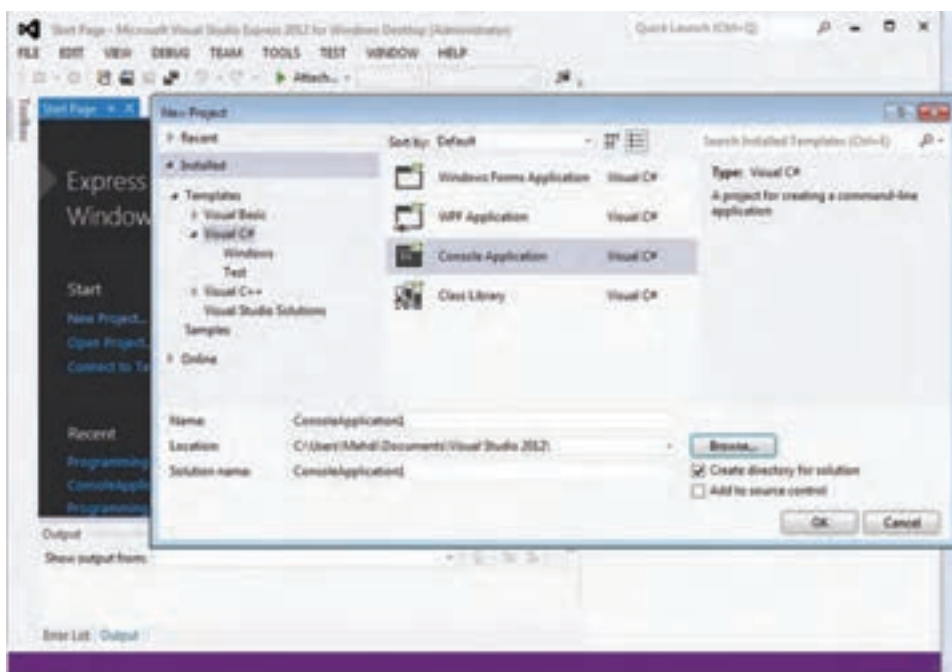
شکل ۵-۱- محیط Visual Studio

برای نوشتن برنامه‌های این کتاب نیز از ویژوال استودیو اکسپرس ۲۰۱۲ یا به اختصار VS^۳، 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 ۱-۱، ابتدا VS را اجرا کرده، و در پنجره Start page گزینه New Project، گزینه Visual C# و سپس Console Application را انتخاب کنید.

۱- Visual Studio Express

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

۳- Visual Studio



شکل ۱-۶- پنجره پروژه جدید

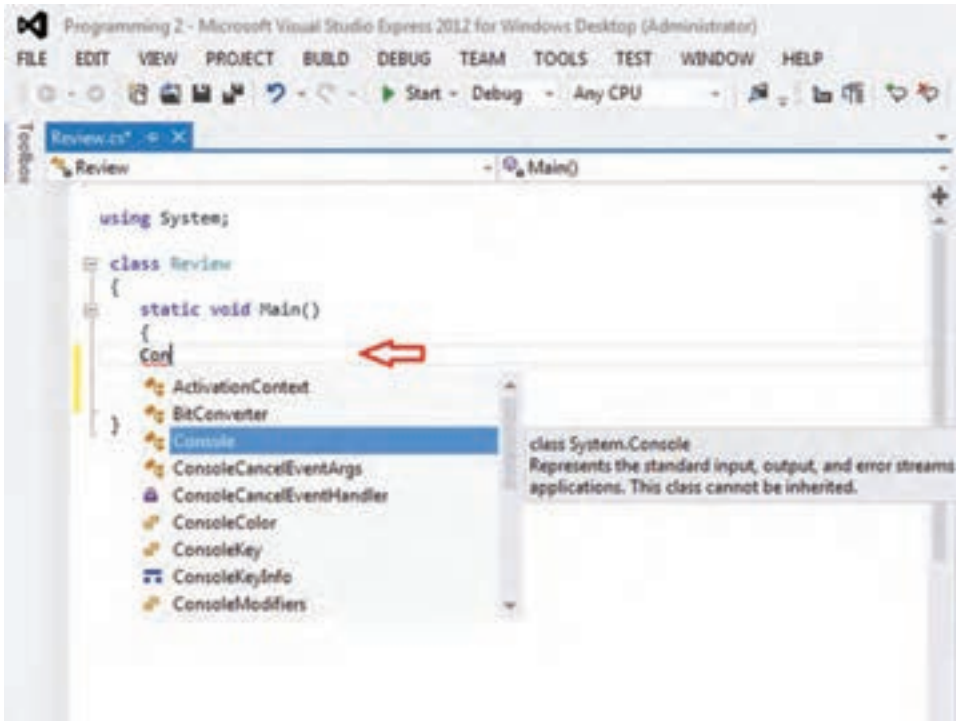
پس از مشخص کردن نام پروژه و مسیری که در آن پروژه ذخیره می‌شود، بر روی OK کلیک می‌کنیم و در پنجره ویرایشگر برنامه، دستورات را در داخل Main() می‌نویسیم.

محیط VS بسیار قدرتمند و دارای ابزارهای مختلفی برای نوشتن، اجرا و عیب‌یابی برنامه است. یکی از امکانات ارزشمند در هنگام نوشتن برنامه، IntelliSense^۱ 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^۲، دستور مورد نظر، در برنامه به طور خودکار قرار می‌گیرد و لازم نیست بقیه نام دستور را تایپ کنید. این امکان علاوه بر اینکه سبب می‌شود سرعت تایپ برنامه افزایش یابد، باعث می‌شود دستورات با املای صحیح و بدون خطا نوشته شود.

۱- Intelligent Sense

۲- Space Bar

همان طور که در شکل ۷-۱ مشاهده می‌کنید، برنامه‌نویس قصد تایپ نام کلاس Console را دارد که به محض تایپ حرف C، لیست IntelliSense به طور خودکار^۱ باز می‌شود و نام‌هایی را نشان می‌دهد که با حرف C شروع می‌شوند و اگر برنامه‌نویس 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.



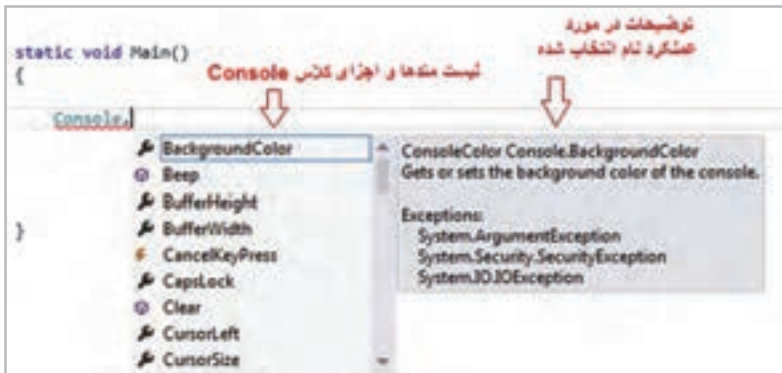
شکل ۷-۱- پنجره کدنویسی

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

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

نکته

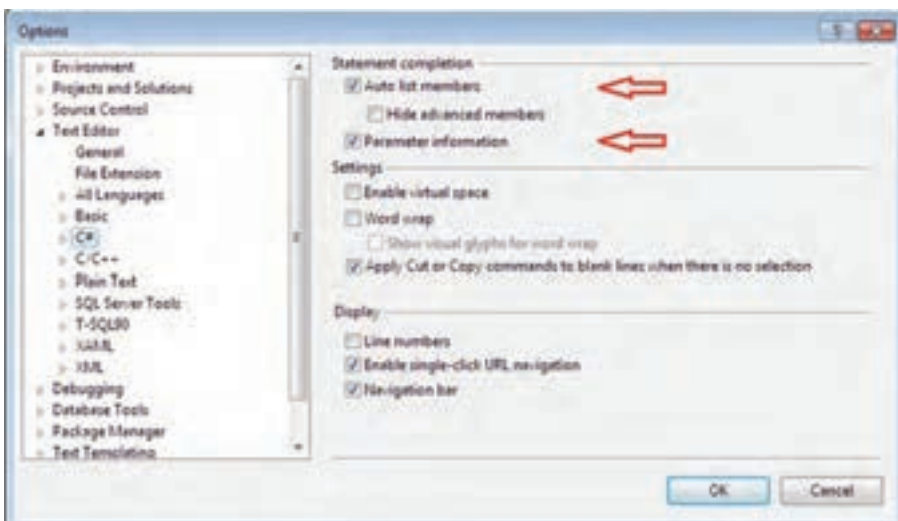
لیست مربوط به IntelliSense را در داخل متن برنامه، می توانید با زدن کلید ترکیبی CTRL+SPACE ظاهر کنید.



شکل ۸-۱- منوی IntelliSense

نکته

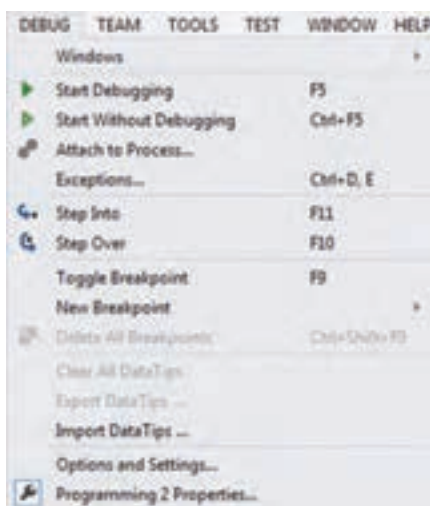
اگر امکان IntelliSense فعال نیست و از آن نمی توانید استفاده کنید، باید از طریق منوی Tools گزینه Options وارد قسمت Text Editor شوید و سپس در بخش C#, گزینه `Auto list members` و `Parameter information` را فعال کنید.



شکل ۹-۱- 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 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 ۱۰-۱- منوی Debug برای اجرای برنامه

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



شکل ۱۱-۱- گزینه Start در نوار ابزار محیط VS

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

۱-۳-۱ یادآوری دستورات زبان برنامه نویسی C#

۱-۳-۱-۱ 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 می‌شود.

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

۱-۳-۲-۱ 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: `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 ۱-۲: 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 می‌کنیم.

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

برنامه ۱-۲-۱- 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 ۳-۱- نمایش سن سال جاری

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

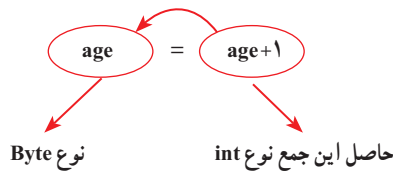
```
age = age + 1;
```

Cannot implicitly convert type 'int' to 'byte'. An explicit conversion exists (are you missing a cast?)

شکل ۱۲-۱- 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: $age = age + 1$ 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 به طور 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 شماست 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 تبدیل نوع استفاده می‌کنیم (برنامه ۴-۱). 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 ۴-۱- نوع دیگری از برنامه نمایش سن سال جاری

در تبدیل نوع داده باید به نوع داده‌های مقصد و مبدأ توجه کرد. نوع داده مقصد باید بتواند مقدار داده 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 ۳-۱: 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 ۵-۱- نمایش کدهای اسکی کاراکترهای C و #

مثال ۴-۱: 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 وسیله 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 ۴-۱- خوشامدگویی به کاربر

در داخل پرانتز در متد `WriteLine()` برنامه ۴-۱، از علامت `+` برای اتصال و الحاق رشته‌ها استفاده شده است. عملگر `+` هنگامی که با رشته‌ها و یا متغیرهای رشته‌ای به کار می‌رود، عمل الحاق یا کنار هم قرار دادن مقدار رشته‌ها را انجام می‌دهد.

۳-۱- الگوی 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

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 :

Console.WriteLine("{0} plus {1} is equal to {2}", 17, 45, 17+45);

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 نحوه نمایش داده را مشخص می‌کند. مثلاً برای نمایش اعداد اعشاری با دو رقم اعشار (مثل میانگین نمره) از کاراکتر F استفاده می‌شود. در پیوست ۲، لیست کامل کاراکترهای الگوی نمایش برای انواع داده‌ها وجود دارد.

مثال ۵-۱ : در برنامه ۷-۱ 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.

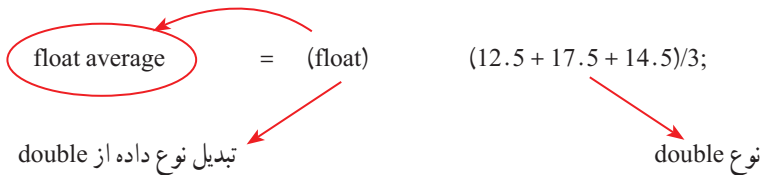

```

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);
    }
}

```

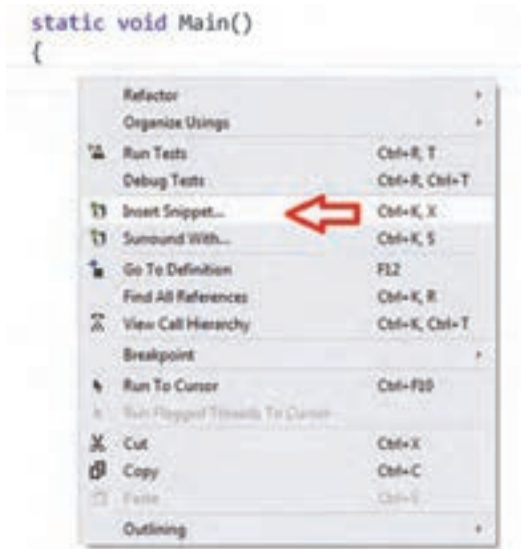
برنامه ۷-۱- نمایش عدد اعشاری با دو رقم اعشار و با تراز راست



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 تبدیل نوع استفاده شده است. این عمل با استفاده از کلمه کلیدی (float) در این خط از برنامه انجام شده است.

۴-۳-۱ یادآوری دستورات شرطی

۴-۳-۱-۱ 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، در صورت 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 برنامه‌نویسی، امکاناتی را فراهم کرده‌اند که نیاز به حفظ کردن تمامی جزئیات و تاپ 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 ۱۳-۱).



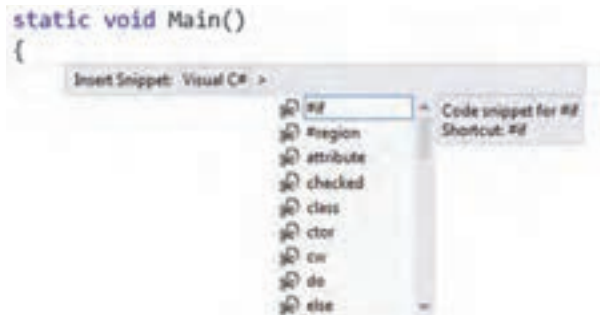
شکل ۱۳-۱- گزینه Insert Snippet



شکل ۱۴-۱- انتخاب گزینه Visual C# از Insert Snippet

در این صورت منوی دیگری باز می شود و از آن گزینه Visual C# را انتخاب کنید (شکل ۱۴-۱).

در این صورت منویی ظاهر می شود که دستورات و کلمات رزرو شده زبان سی شارپ را نشان می دهد (شکل ۱۵-۱).



شکل ۱۵-۱- 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 ۱۵-۱). بعد از ظاهر شدن دستور انتخابی، می‌توانیم عبارت داخل پرانتز را به دلخواه تغییر دهیم (شکل ۱۶-۱).

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

شکل ۱۶-۱ ساختار دستور if

کار در کارگاه ۲

مثال ۱۶-۱: برنامه‌ای بنویسید که نمره 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. داده دریافتی به صورت یک رشته به وسیله این متد تحویل داده می‌شود. مانند "18.75" بنابراین به یک متغیر رشته‌ای برای ذخیره 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 ۸-۱- معتبر بودن نمره

مثال ۷-۱: مثال قبل را توسعه دهید به طوری که 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 برنامه زیر، دستور 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 ۸-۱ درج کنیم؟

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 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 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.

۲-۴-۱- 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 می توان 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 اول ؟ (عبارت منطقی)

کامپیوتر در هنگام برخورد با عملگر سه تایی ابتدا حاصل عبارت منطقی را محاسبه می کند، 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 می شود؟

مثال ۸-۱: در برنامه ۹-۱ 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 ۹-۱ تشخیص زوج و فرد بودن اعداد

۳-۴-۳-۱: 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 ۹-۱: برنامه ۱۰-۱ به وسیله 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 اگر عدد وارد شده با هر کدام 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 ۱۰-۱- تغییر رنگ صفحه کنسول

۵-۳-۱- 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-while` و دستور `for` آشنا شدید. در این قسمت با ذکر چند مثال، انواع دستورات تکرار را یادآوری می‌کنیم.

۵-۳-۱- 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 ۱-۱۰ : 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 متغیر کم 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 ۱۱-۱- نمایش اعداد متوالی به صورت کاهشی تا صفر

سؤال! اگر علامت‌های آکولاد باز و بسته در حلقه 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 ۱۱-۱: 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 :

۱- تعداد کاراکتر رشته دریافتی را شمارش کنیم که همان ارقام عدد هستند. در فصل سوم با این روش تعداد ارقام عدد را حساب می کنیم.

۲- مانند مثال های قبلی، رشته دریافتی را به عدد تبدیل کرده، و سپس با استفاده از تقسیم های متوالی به 10^0 تعداد دفعات تقسیم را می شماریم که همان تعداد ارقام عدد است. در برنامه ۱۲-۱ 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 ۱۲-۱- نمایش شمارش ارقام

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

۱- البته علامت عدد و یا فاصله را باید در نظر داشت و به عنوان تعداد ارقام آنها را محسوب نکرد.

۵-۲-۳-۱- دستور **for** : برای اجرای دستور یا دستورات به تعداد دفعات معین، از دستور **for** استفاده می‌کنیم.

مثال ۱-۱۲ : در برنامه مثال قبل (چاپ اعداد به صورت کاهشی)، به جای دستور **while** از دستور **for** استفاده کنید.

مثال ۱-۱۳ : برنامه‌ای بنویسید که نمرات درس برنامه‌سازی 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 ۱-۱۳- محاسبه مجموع و معدل ۱۰نمره

۱-۴ استفاده از مقدار ثابت^۱ در برنامه

مثال ۱-۴: برنامه مثال قبل را برای یک کلاس با ۱۵ نفر دانش آموز 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 حلقه for را به ۱۵ تغییر می دهیم. 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 می شود و مجموع نمرات به درستی حساب می شود اما میانگین نمرات به اشتباه حساب می شود. برای جلوگیری از چنین اشتباهاتی و همچنین توسعه راحت برنامه می توان 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 می شود. معمولاً نامی که برای اعداد ثابت تعریف می شود با حروف بزرگ نوشته می شود تا در برنامه مشخص باشد که این شناسه، یک مقدار ثابت است. نحوه تعریف ثابت ها، مانند تعریف متغیرها می باشد با این تفاوت که در ابتدای تعریف آنها کلمه const قرار دارد.

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

برنامه ۱-۴ 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++) ↔
```

```
{
```

^۱_ Constant

```

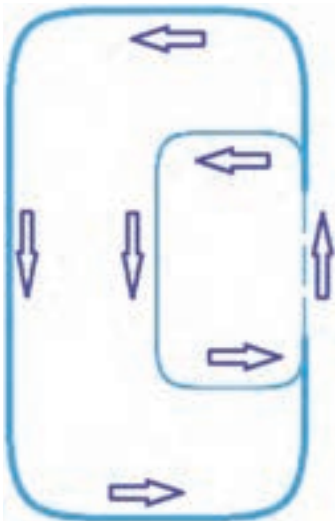
Console. Write ( " Enter a mark: " );
input = Console. ReadLine ( );
number = float. Parse (input);
total = total + number;
}
average = total / SIZE ; ←
Console. WriteLine ( "Total : " + total);
Console. WriteLine ( "Average : " + average);
}
}

```

برنامه ۱۴-۱- امکان تغییر راحت برنامه برای دریافت ۱۵ نمره

سؤال: اگر بخواهید برنامه ۱۴-۱ برای محاسبه میانگین نمرات پنج دانش آموز به کار رود، چه تغییری در آن اعمال می کنید؟

۵-۱- حلقه های متداخل (تو در تو)



شکل ۱۷-۱- پیست دومیدانی
مشابه حلقه های تو در تو

در بعضی از مواقع، یک دستور تکرار را در داخل دستور تکرار دیگری به کار می بریم. به عبارت دیگر هنگامی که در داخل یک حلقه، حلقه دیگری قرار داشته باشد، حلقه های تو در تو یا متداخل^۱ نامیده می شوند.

حلقه های تو در تو را مانند پیست دومیدانی در نظر بگیرید، حال دونه ای 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؟

مثال ۱۵-۱: در برنامه ۱۵-۱، از دو دستور for استفاده شده است که یکی در داخل دیگری

قرار دارد و بنابراین حلقه تودرتو را تشکیل می دهند.

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 می دهد. کامپیوتر به ازای هر بار اجرای حلقه

بیرونی، حلقه داخلی را به طور کامل (پنج بار) انجام می دهد.

```
using System;
```

```
class Review
```

```
{
```

```
    static void Main ( )
```

حلقه بیرونی

```
    {
```

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

```
        {
```

```
            for (int j = 1; j <= 5; j++)
```

حلقه داخلی

```
                Console. Write ("*");
```

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

```
            }
```

```
        }
```

```
    }
```

برنامه ۱۵-۱- حلقه تودرتو

در حلقه داخلی دستور نمایش رشته "*" قرار دارد، بنابراین با اجرای حلقه داخلی، پنج بار

رشته "*" در کنار هم در روی یک خط در صفحه نمایش به صورت زیر چاپ شود:

```
*****
```

از آنجا که حلقه بیرونی، ده بار تکرار می شود بنابراین طرح زیر، روی صفحه نمایش نقش می بندد:

```
*****
```

```
*****
```

```
*****
```

```
*****
```

۱_ Outer Loop

۲_ 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؟

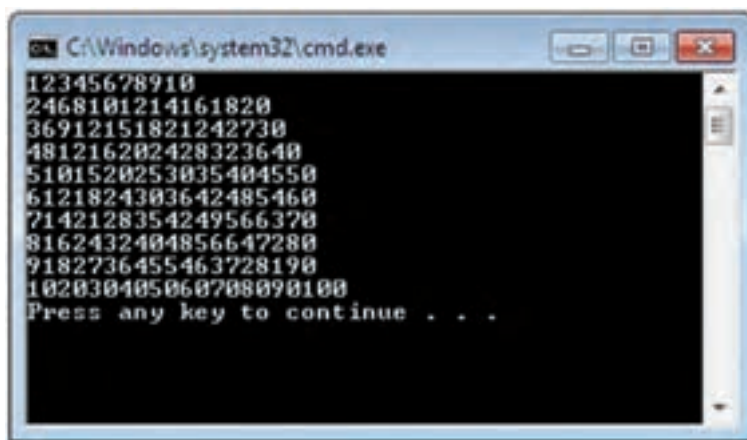
کار در کارگاه ۵

مثال ۱۶-۱: 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 ۱۶-۱- جدول ضرب اعداد ۱ تا ۱۰

نتیجه اجرا یا خروجی برنامه فوق چنین خواهد بود :



```
C:\Windows\system32\cmd.exe
12345678910
2468101214161820
36912151821242730
481216202428323640
5101520253035404550
6121824303642485460
7142128354249566370
8162432404856647280
9182736455463728190
102030405060708090100
Press any key to continue . . .
```

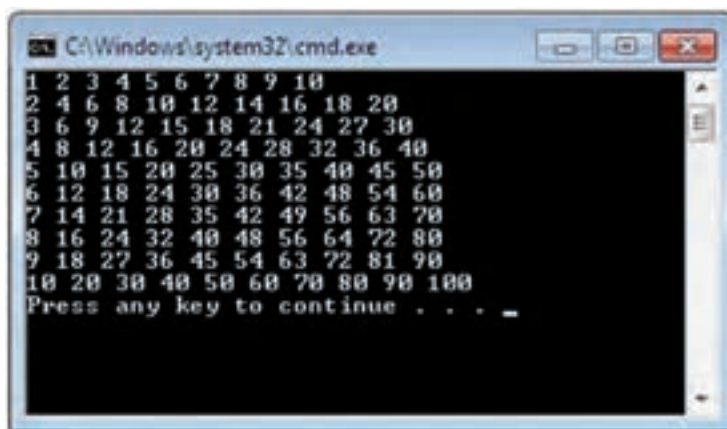
شکل ۱۸-۱- خروجی برنامه جدول ضرب

همان طور که مشاهده می کنید 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 اضافه کردن 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 :



```
C:\Windows\system32\cmd.exe
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
Press any key to continue . . .
```

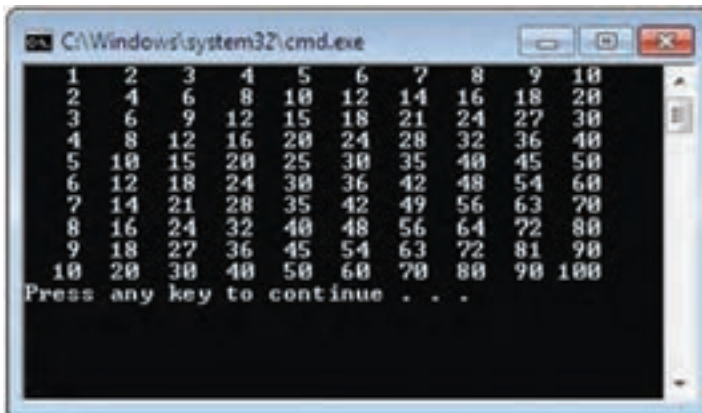
شکل ۱۹-۱- جدا کردن اعداد جدول ضرب

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^۱ نمایش از قبل پیش بینی شده است. اعداد جدول ضرب دو رقمی هستند به جز حالت خاص ضرب ۱۰ در ۱۰ است که حاصل برابر ۱۰۰ و یک عدد ۳ رقمی است 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:



```
C:\Windows\system32\cmd.exe
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
Press any key to continue . . .
```

شکل ۲۰-۱. خروجی نهایی برنامه جدول ضرب

۱-۶- کاراکتر ۲

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

۱- Format String

۲- Back Slash Character

به این مجموعه کاراکترها که کاراکتر اول \ و دومی کاراکتر دیگری است، دنباله فرار^۱ گفته می‌شود. یکی از کاربردهای این دنباله، نمایش علامت^۲ در یک رشته است. اگر بخواهیم جمله‌ای را همراه با علامت^۳ 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^۴ 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 معنی دار نشان داده شده‌اند.

جدول ۱-۱- دنباله معنی دار

دنباله	عملکرد
\a	ایجاد یک بوق هشدار ^۲
\b	حذف یک کاراکتر (Backspace)
\n	ایجاد یک خط خالی (New Line)
\t	ایجاد یک فاصله افقی زیاد tab
\'	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.

مثال ۱-۱۷: در برنامه ۱-۱۷ دنباله‌های معنی‌دار استفاده شده‌اند. خروجی هر دستور به صورت 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 ۱-۱۷- نمایش دنباله معنی‌دار

خودآزمایی فصل اول

الف) درستی یا نادرستی هر عبارت را تعیین کنید.

- ۱- مقدار یک شناسه ثابت را در طول برنامه می توان تغییر داد.
- ۲- برای بررسی حالات مختلف یک عبارت و اجرای دستورات بر اساس آن از دستور switch استفاده می کنیم.
- ۳- برای نمایش کاراکترهای خاص مثل " وسط یک رشته از دنباله معنی دار استفاده می شود.
(ب) جاهای خالی را با عبارت مناسب پر کنید.
- ۴- لیستی از نام متدها و دستورات که هنگام تایپ نمایش داده می شود نام دارد.
- ۵- برای نمایش منوی دستورات و کلمات رزرو شده سی شارپ از فرمان استفاده می شود.
- ۶- برای تعریف شناسه یا نام ثابت از کلمه کلیدی استفاده می شود.
- ۷- IDE ساده و کم حجم، برای نوشتن و ترجمه برنامه های سی شارپ نام دارد.
(ج) به سؤالات زیر پاسخ دهید.
- ۸- تفاوت متد Parse و Try parse را بنویسید.
- ۹- قطعه برنامه زیر را Trace کنید و در ماتریس زیر جای گذاری کنید.

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

—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

- ۱۰- در اجرای قطعه کد زیر پیام خطا داده می شود. به نظر شما مشکل چیست؟ با تغییر کد
int x = 14.35;
مشکل را برطرف کنید.

۱۱- 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!");
```

تمرینات برنامه‌نویسی فصل اول



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

۲- 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);  
  
}  
}
```

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

```
1 = 1  
1+2 = 3  
1+2+3 = 6  
1+2+3+4 = 10  
1+2+3+4+5 = 15  
1+2+3+4+5+6 = 21  
1+2+3+4+5+6+7 = 28  
1+2+3+4+5+6+7+8 = 36  
1+2+3+4+5+6+7+8+9 = 45  
1+2+3+4+5+6+7+8+9+10 = 55
```


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

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

طرح (۱)
طرح (۲)
طرح (۳)
طرح (۴)

۵- 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);
```

۶- 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 $2+7 = 4+5 \rightarrow 9 = 9$

(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.

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	