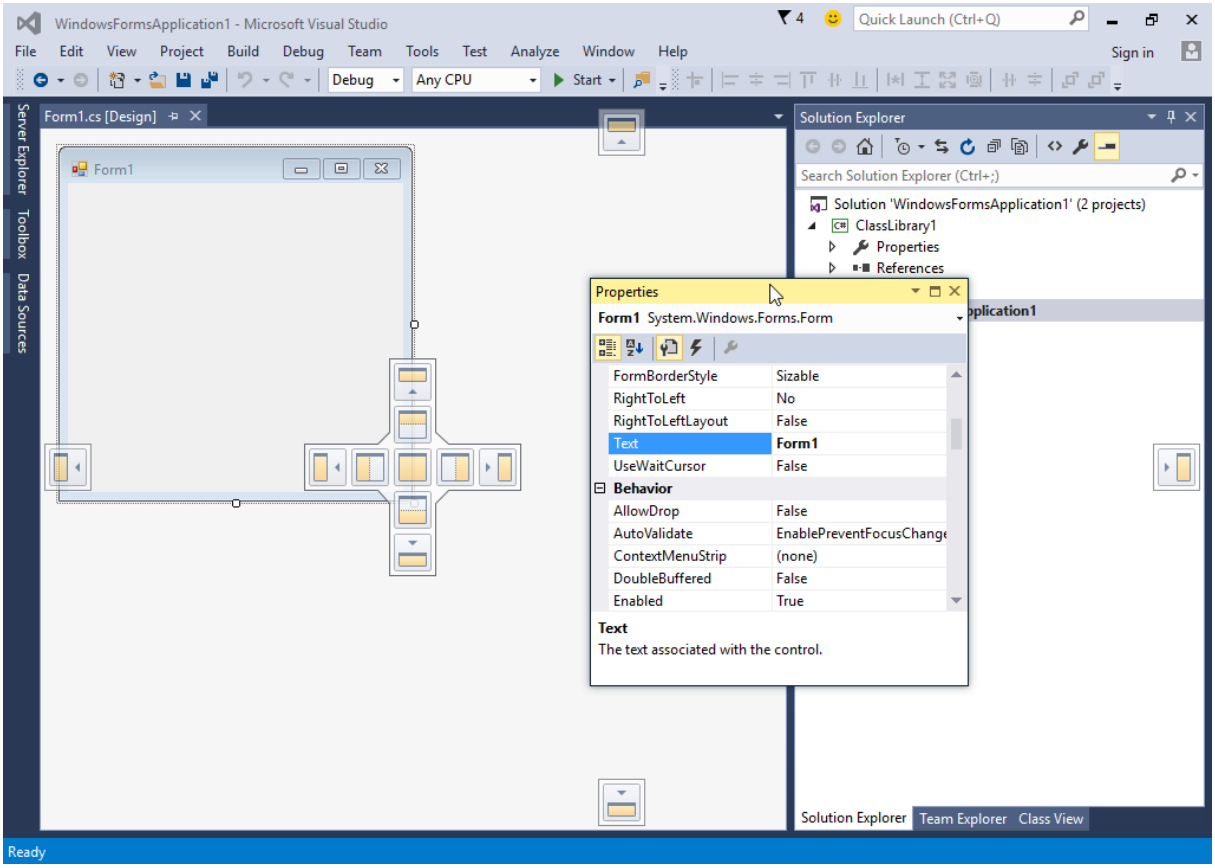
پنجره خواص(Properties) ، خواص و رویدادهای مختلف هر آیتم انتخاب شده اعم از فرم ، فایل ، پروژه و کنترل را نشان می دهد. اگر این پنجره مخفی است، می توانید از مسیر View > Properties Window یا کلید میانبر F4 آنرا ظاهر کنید. در مورد خواص در درسهای آینده مفصل توضیح خواهیم داد. خاصیت ها ، ویژگیها و صفات اشیا را نشان می دهند. به عنوان مثال یک ماشین دارای خواصی مانند رنگ ، سرعت، اندازه و مدل است. اگر یک فرم یا کنترل را در صفحه طراحی و یا یک پروژه یا فایل را در SolutionExplorer انتخاب کنید پنجره خواص مربوط به آنها نمایش داده خواهد شد. این پنجره همچنین دارای رویدادهای مربوط به فرم یا کنترل انتخاب شده می باشد. یک رویداد (event) اتفاقی است که در شرایط خاصی پیش می آید مانند وقتی که بر روی دکمه (button) کلیک و یا متنی را در داخل جعبه متن (text box) اصلاح می کنیم. کمبو باکس (combo box)شکل بالا که با حرف A نشان داده شده است به شما اجازه می دهد که شیء مورد نظرتان (دکمه، فرم و...)را که می خواهید خواص آنرا تغییر دهید انتخاب کنید. این کار زمانی مفید است که کنترلهای روی فرم بسیار کوچک یا به هم نزدیک بوده و انتخاب آنها سخت باشد. در زیر کمبو باکس بالا دکمه های مفیدی قرار دارند .(B) برخی از این دکمه ها در شرایط خاصی فعال می شوند. دکمه اول خاصیت اشیا را بر اساس دسته های مختلفی مرتب می کند. دومین دکمه خواص را بر اساس حروف الفبا مرتب می کند که پیشنهاد می کنیم از این دکمه برای دسترسی سریع به خاصیت مورد نظرتان استفاده کنید. سومین دکمه هم وقتی ظاهر می شود که یک کنترل یا یک فرم را در محیط طراحی انتخاب کنیم. این دکمه به شما اجازه دسترسی به خواص فرم ویا کنترل انتخاب شده را می دهد. چهارمین دکمه (که به شکل یک رعد و برق نمایش داده شده) رویدادهای فرم ویا کنترل انتخاب شده را می دهد. در پایین شکل بالا توضیحات کوتاهی در مورد خاصیت ها و رویداد ها نشان داده می شود. بخش اصلی پنجره خواص(C) شامل خواص و رویدادها است. در ستون سمت چپ نام رویداد یا خاصیت و در ستون سمت راست مقدار آنها آمده است. در پایین پنجره خواص جعبه توضیحات (D) قرار دارد که توضیحاتی درباره خواص و رویدادها در آن نمایش داده می شود.

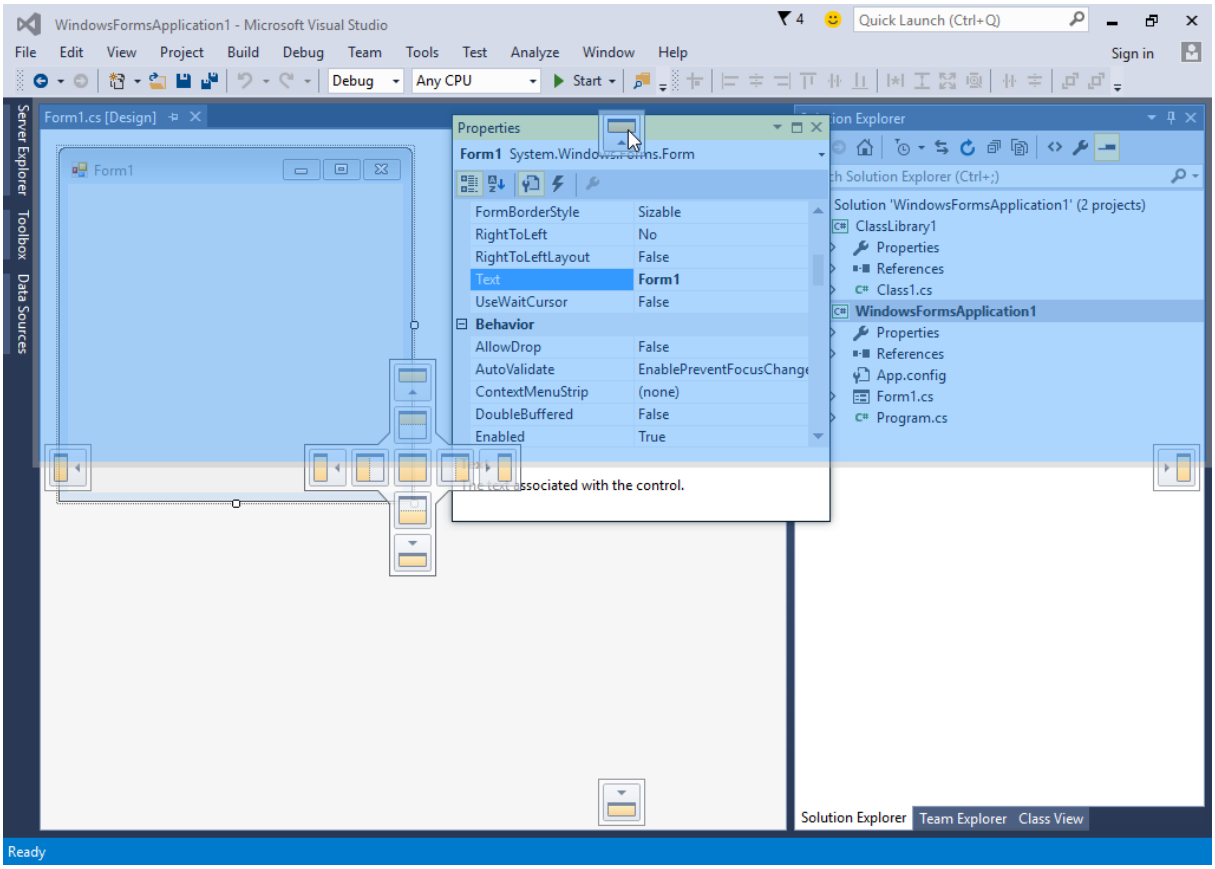
تغییر ظاهر ویژوال استودیو

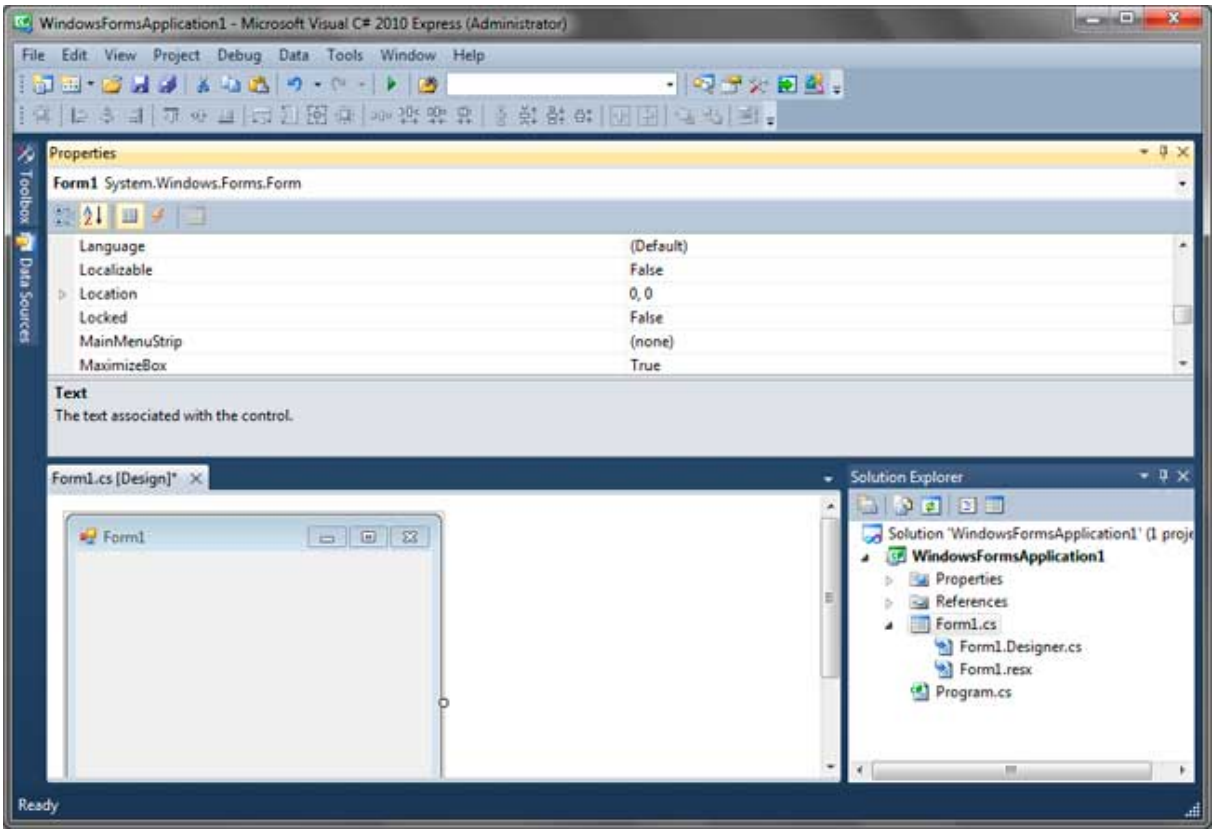
اگر موقعیت پنجره ها و یا ظاهر برنامه ویژوال استودیو را دوست نداشته باشید، می توانید به دلخواه آن را تغییر دهید. برای این کار بر روی نوار عنوان (title bar) کلیک کرده و آنرا می کشید تا پنجره به شکل زیر به حالت شناور در آید:



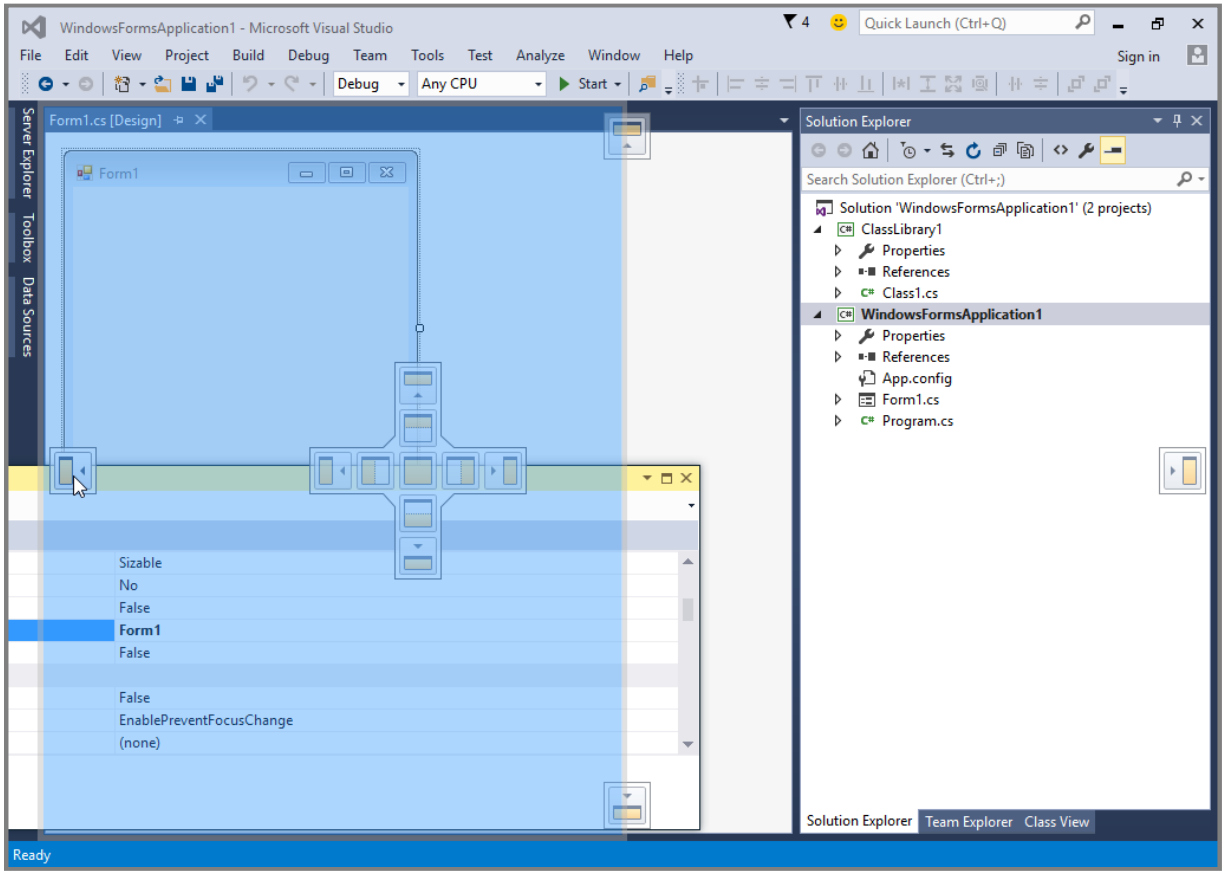
در حالی که هنوز بر روی پنجره کلیک کرده اید و آن را می کشید یک راهنما (فلشی با چهار جهت) ظاهر می شود و شما را در قرار دادن پنجره در محل دلخواه کمک می کند. به عنوان مثال شما می توانید پنجره را در بالاترین قسمت

محیط برنامه قرار دهید، منطقه ای که پنجره قرار است در آنجا قرار بگیرد به رنگ آبی در می آید:

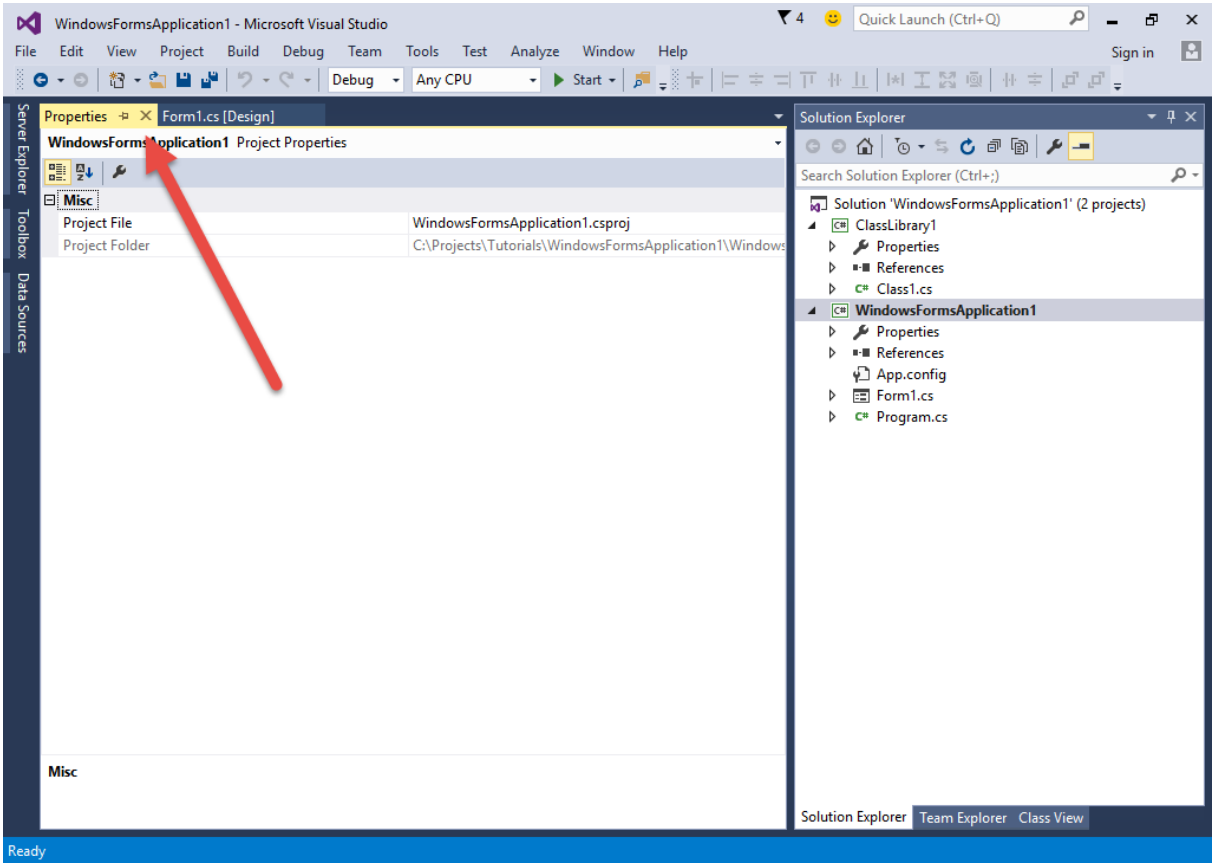




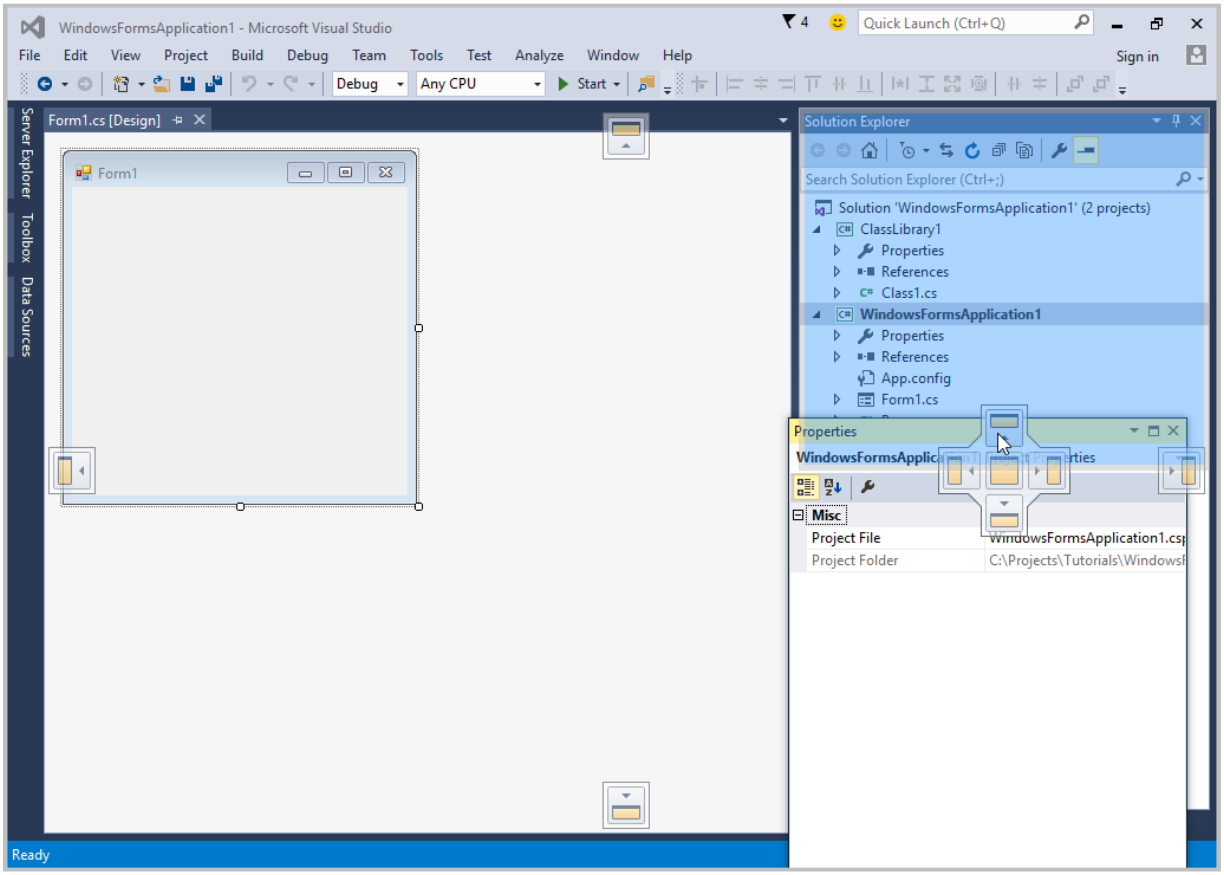
پنجره در قسمت بالای محیط قرار داده شده است. راهنمای صلیب شکل حاوی جعبه های مختلفی است که به شما اجازه می دهد پنجره انتخاب شده را در محل دلخواه محیط ویژوال استودیو قرار دهید. به عنوان مثال پنجره Properties را انتخاب کنید و آنرا به چپ ترین قسمت صلیب در پنجره نمایش داده شده نزدیک و رها کنید، مشاهده می کنید که پنجره مذکور در سمت چپ پنجره Design View قرار می گیرد:



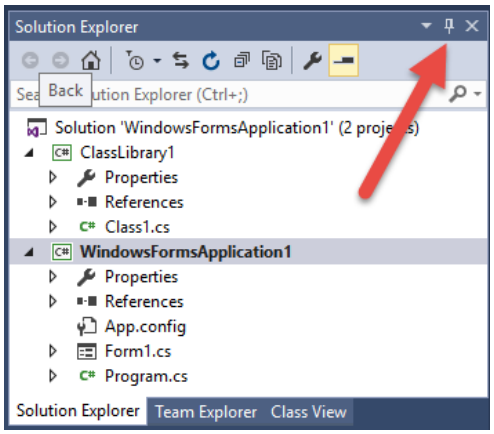
کشیدن پنجره به مرکز صلیب راهنما باعث ترکیب آن با پنجره مقصد می شود که در مثال بالا شما می توانید به عنوان یک تب به پنجره Properties دست پیدا کنید.



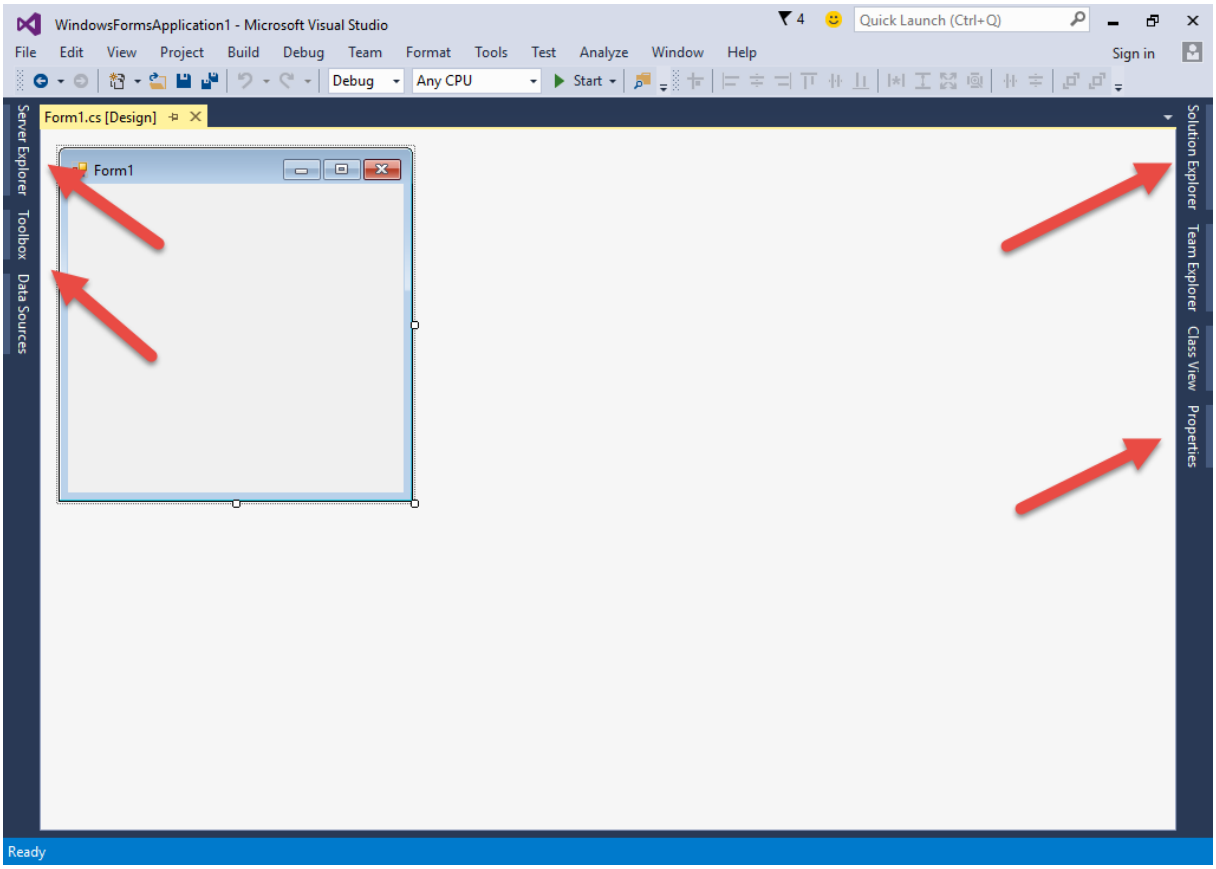
اگر به عنوان مثال پنجره Properties را روی پنجره Solution Explorer بکشید، یک صلیب راهنمای دیگر نشان داده می شود. با کشیدن پنجره به قسمت پایینی صلیب، پنجره Properties زیر پنجره Solution Explorer قرار خواهد گرفت.



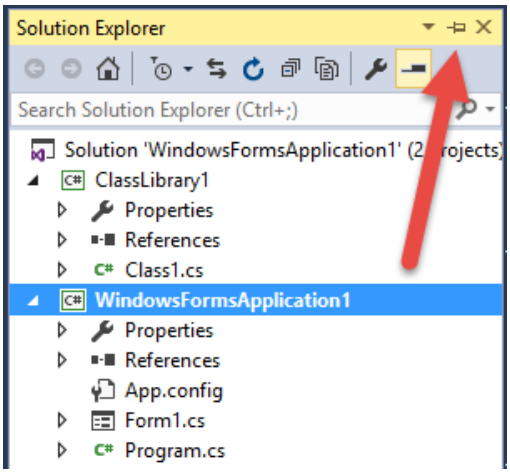
قسمتی از محیط برنامه که می خواهید پنجره در آنجا قرار بگیرد به رنگ آبی در می آید. ویژوال سی شارپ همچنین دارای خصوصیتی به نام **autohide** است که به صورت اتوماتیک پنجره ها را مخفی می کند. هر پنجره دارای یک آیکون سنجاق مانند نزدیک دکمه **close** می باشد.



بر روی این آیکون کلیک کنید تا ویژگی **auto-hide** فعال شود. برای دسترسی به هر یک از پنجره ها می توان با ماوس بر روی آنها توقف یا بر روی تب های کنار محیط ویژوال استودیو کلیک کرد.



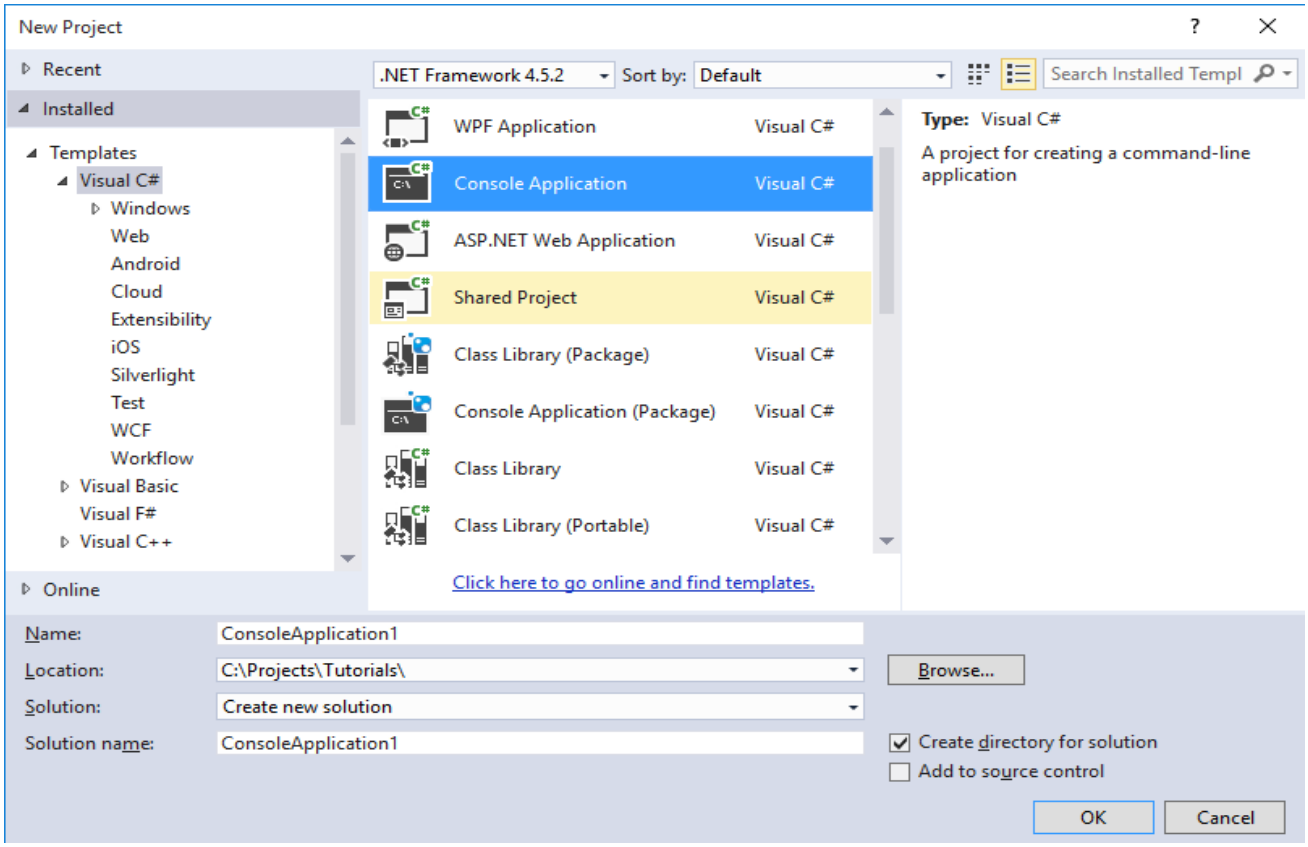
برای غیر فعال کردن این ویژگی در هر کدام از پنجره ها کافیست پنجره را انتخاب کرده و دوباره بر روی آیکون مورد نظر کلیک کنید.



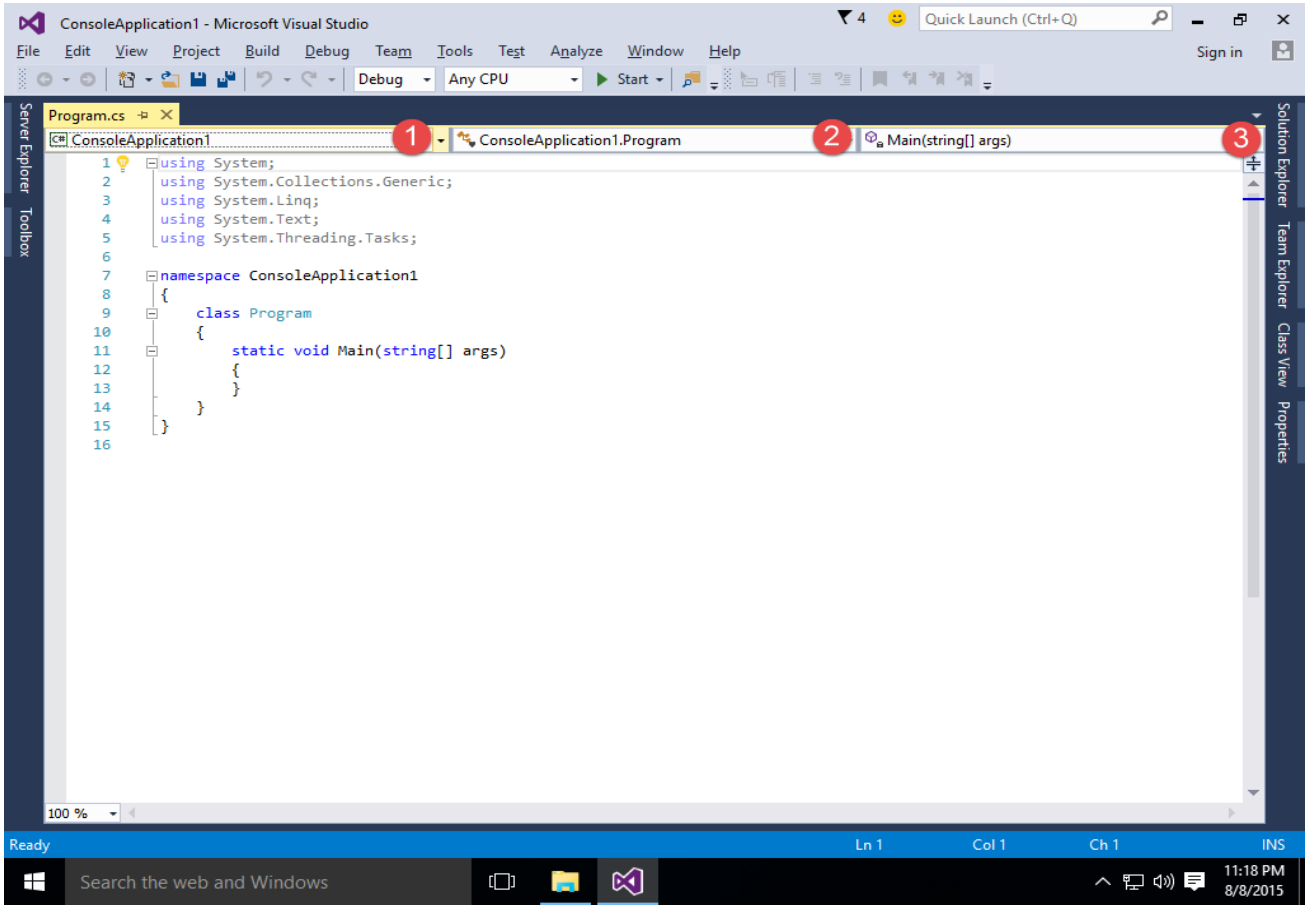
به این نکته توجه کنید که اگر شکل آیکون افقی بود بدین معناست که ویژگی فعال و اگر شکل آن عمودی بود به معنای غیر فعال بود ویژگی auto-hide می باشد.

ساخت یک برنامه ساده

اجازه بدهید یک برنامه بسیار ساده به زبان سی شارپ بنویسیم. این برنامه یک پیغام را در محیط کنسول نمایش می دهد. در این درس می خواهم ساختار و دستور زبان یک برنامه ساده سی شارپ را توضیح دهم. برنامه Visual Studio Community را اجرا کنید. از مسیر File > New Project یک پروژه جدید ایجاد کنید. حال با یک صفحه مواجه می شوید که از شما می خواهد نام پروژه تان را انتخاب و آن را ایجاد کنید(شکل زیر) :



گزینه **Console Application** را انتخاب کنید و نام پروژه تان را **MyFirstProgram** بگذارید. یک **Console Application** برنامه ای تحت داس در محیط وینوز است و فاقد محیط گرافیکی می باشد. بهتر است برنامه خود را در محیط کنسول بنویسید تا بیشتر با مفهوم برنامه نویسی آشنا شوید. بعد از اینکه آموزش مبانی زبان به پایان رسید، برنامه نویسی در محیط ویندوز و بخش بصری سی شارپ را آموزش خواهیم داد. بعد از فشردن دکمه **OK**، ویژوال استودیو یک **solution** در یک فولدر موقتی ایجاد می کند. یک **solution** مجموعه ای از پروژه هاست ،اما در بیشتر تمرینات شامل یک پروژه می باشد. فایل **solution** دارای پسوند **sln** بوده و شامل جزییاتی در مورد پروژه ها و فایلهای وابسته به آن می باشد. پروژه جدید همچنین حاوی یک فایل با پسوند **csproj**. می باشد که آن نیز شامل جزییاتی در مورد پروژه ها و فایلهای وابسته به آن می باشد. حال می خواهیم شما را با محیط کد نویسی آشنا کنیم.



محیط کدنویسی جایی است که ما کدها را در آن تایپ می کنیم. کدها در محیط کدنویسی به صورت رنگی تایپ می شوند در نتیجه تشخیص بخشهای مختلف کد را راحت می کند. منوی سمت چپ شامل(شماره 1) نام پروژه ای که ایجاد کرده اید، منوی وسط (شماره 2) شامل لیست کلاسها ،ساختارها ، انواع شمارشی و...و منوی سمت راست (شماره 3)شامل اعضای کلاسها ،ساختارها ، انواع شمارشی و... می باشد. نگران اصطلاحاتی که به کار بردیم نباشید آنها را در فصول بعد توضیح خواهم داد. همه فایلهای دارای کد در سی شارپ دارای پسوند cs.هستند. در محل کد نویسی کدهایی از قبل نوشته شده که برای شروع شما آنها را پاک کنید و کدهای زیر را در محل کدنویسی بنویسید:

```
namespace MyFirstProgram
{
    class Program
    {
        static void Main()
        {
            System.Console.WriteLine("Welcome to Visual C# Tutorials!");
        }
    }
}
```

ساختار یک برنامه در سی شارپ

مثال بالا ساده ترین برنامه ای است که شما می توانید در سی شارپ بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است. هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. اجازه بدهید هر خط کد را در مثال بالا توضیح بدهیم.

در خط 1 فضای نام **(namespace)** تعریف شده است که شامل کدهای نوشته شده توسط شما است و از تداخل نامها جلوگیری می کند. در باره فضای نام در درسهای آینده توضیح خواهیم داد.

در خط **2** آکولاد **({)** نوشته شده است. آکولاد برای تعریف یک بلوک کد به کار می رود. سی شارپ یک زبان ساخت یافته است که شامل کدهای زیاد و ساختارهای فراوانی می باشد. هر آکولاد باز**({)** در سی شارپ باید دارای یک آکولاد بسته **(})** نیز باشد. همه کدهای نوشته شده از خط **2** تا خط 10 یک بلوک کد یا بدنه فضای نام است.

در خط **3** یک کلاس تعریف شده است. در باره کلاسها در فصلهای آینده توضیح خواهیم داد. در مثال بالا کدهای شما باید در داخل یک کلاس نوشته شود. بدنه کلاس شامل کد های نوشته شده از خط **4** تا **9** می باشد.

در خط **5** متد **Main** یا متد اصلی تعریف شده است. هر متد شامل یک سری کد است که وقتی اجرا می شوند که متد را صدا بزنیم. درباره متد و نحوه صدا زدن آن در فصول بعدی توضیح خواهیم داد. متد **Main** نقطه آغاز اجرای برنامه است. این بدان معناست که ابتدا تمام کدهای داخل متد **Main** و سپس بقیه کدها اجرا می شود. در باره متد **Main** در فصول بعدی توضیح خواهیم داد. متد **Main** و سایر متدها دارای آکولاد و کدهایی در داخل آنها می باشند و وقتی کدها اجرا می شوند که متدها را صدا بزنیم. هر خط کد در سی شارپ به یک سیمیکولن **(;)** ختم می شود. اگر سیمیکولن در آخر خط فراموش شود برنامه با خطا مواجه می شود. مثالی از یک خط کد در سی شارپ به صورت زیر است :

```
System.Console.WriteLine("Welcome to Visual C# Tutorials!");
```

این خط کد پیغام **Welcome to Visual C# Tutorials!** را در صفحه نمایش نشان می دهد. از متد **()WriteLine** برای چاپ یک رشته استفاده می شود. یک رشته گروهی از کاراکترها است که به وسیله دابل کوتیشن **(")** محصور شده است. مانند :"**Welcome to Visual C# Tutorials!**". یک کاراکتر می تواند یک حرف،عدد، علامت یا ... باشد. در کل مثال بالا، نحوه استفاده از متد **WriteLine** است، که در داخل کلاس **Console** که آن نیز به نوبه خود در داخل فضای نام **MyFirstProgram** قرار دارد را نشان می دهد. توضیحات بیشتر در درسهای آینده آمده است. سی شارپ فضای خالی و خطوط جدید را نادیده می گیرد. بنابراین شما می توانید همه برنامه را در یک خط بنویسید. اما اینکار خواندن و اشکال زدایی برنامه را مشکل می کند. یکی از خطاهای معمول در برنامه نویسی فراموش کردن سیمیکولن در پایان هر خط کد است. به مثال زیر توجه کنید :

```
System.Console.WriteLine(
"Welcome to Visual C# Tutorials!");
```

سی شارپ فضای خالی بالا را نادیده می گیرد و از کد بالا اشکال نمی گیرد. اما از کد زیر ایراد می گیرد :

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

```
"Welcome to Visual C# Tutorials!");
```

به سیمیکولن آخر خط اول توجه کنید. برنامه با خطای نحوی مواجه می شود، چون دو خط کد مربوط به یک برنامه هستند و شما فقط باید یک سیمیکولن در آخر آن قرار دهید. همیشه به یاد داشته باشید که سی شارپ به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال **MAN** و **man** در سی شارپ با هم فرق دارند. رشته ها و توضیحات از این قاعده مستثنی هستند که در درسهای آینده توضیح خواهیم داد. مثلا کدهای زیر با خطا مواجه می شوند و اجرا نمی شوند :

```
system.console.writeline("Welcome to Visual C# Tutorials!");
SYSTEM.CONSOLE.WRITELINE("Welcome to Visual C# Tutorials!");
sYsTem.cONsoLe.wRItelIne("Welcome to Visual C# Tutorials!");
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می کند. اما کد زیر کاملا بدون خطا است :

```
System.Console.WriteLine("WELCOME TO VISUAL C# TUTORIALS!");
```

همیشه کدهای خود را در داخل آکولاد بنویسید.

```
{
   statement1;
}
```

این کار باعث می شود که کدنویسی شما بهتر به چشم بیاید و تشخیص خطاها راحت تر باشد. یکی از ویژگیهای مهم سی شارپ نشان دادن کدها به صورت تو رفتگی است. بدین معنی که کدها را به صورت تو رفتگی از هم تفکیک می کند و این در خوانایی برنامه بسیار موثر است.

ذخیره پروژه و برنامه

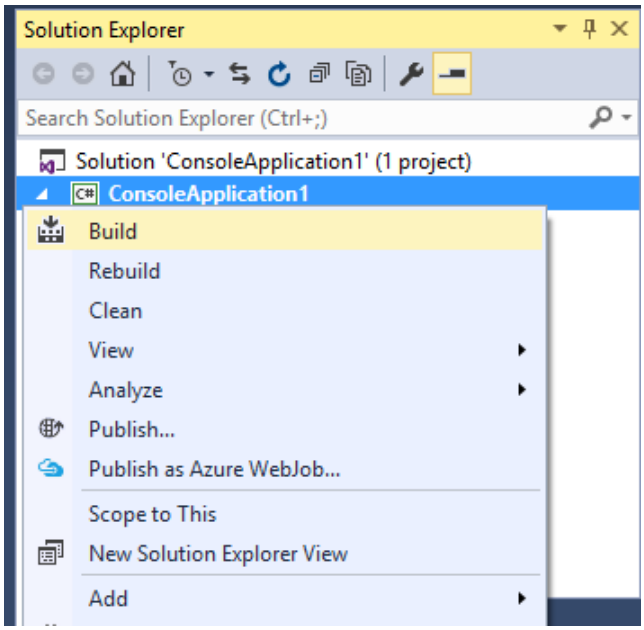
برای ذخیره پروژه و برنامه می توانید به مسیر **File > Save All** بروید یا از کلیدهای میانبر **Ctrl+Shift+S** استفاده کنید. همچنین می توانید از قسمت **Toolbar** بر روی شکل  کلیک کنید.

برای ذخیره یک فایل ساده می توانید به مسیر **File > Save (FileName)** بروید یا از کلیدهای میانبر **Ctrl+S** استفاده کنید. همچنین می توانید از قسمت **Toolbar** بر روی شکل  کلیک کنید.

برای باز کردن یک پروژه یا برنامه از منوی **File** گزینه **Open** را انتخاب می کنید یا بر روی آیکون  در **toolbar** کلیک کنید. سپس به محلی که پروژه در آنجا ذخیره شده میروید و فایلی با پسوند **sln** یا پروژه با پسوند **csproj**. را باز می کنید.

کامپایل برنامه

قبلا یاد ذکر شد که کدهای ما قبل از اینکه آنها را اجرا کنیم ابتدا به زبان میانی مایکروسافت ترجمه می شوند. برای کامپایل برنامه از منوی **Debug** گزینه **Build Solution** را انتخاب می کنید یا دکمه **F6** را بر روی صفحه کلید فشار می دهیم. این کار همه پروژه های داخل **solution** را کامپایل میکند. برای کامپایل یک قسمت از **solution** به **Solution Explorer** می رویم و بر روی آن قسمت راست کلیک کرده و از منوی باز شوند گزینه **build** را انتخاب می کنید. مانند شکل زیر:



اجرای برنامه

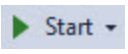
وقتی ما برنامه مان را اجرا می کنیم سی شارپ به صورت اتوماتیک کدهای ما را به زبان میانی مایکروسافت کامپایل می کند. دو راه برای اجرای برنامه وجود دارد :

اجرا همراه با اشکال زدایی **(Debug)**

اجرا بدون اشکال زدایی **(Non-Debug)**

اجرای بدون اشکال زدایی برنامه، خطاهای برنامه را نادیده می گیرد. با اجرای برنامه در حالت **Non-Debug** سریعا برنامه اجرا می شود و شما با زدن یک دکمه از برنامه خارج می شوید. در حالت پیش فرض حالت **Non-Debug** مخفی است و برای استفاده از آن می توان از منوی **Debug** گزینه **Start Without Debuging** را انتخاب کرد یا از دکمه های ترکیبی **Crl + F5** استفاده نمود:

```
Welcome to Visual C# Tutorials!
Press any key to continue . . .
```

به این نکته توجه کنید که پیغام **Press any key to continue...** جز خروجی به حساب نمی آید و فقط نشان دهنده آن است که برنامه در حالت **Non-Debug** اجرا شده است و شما می توانید با زدن یک کلید از برنامه خارج شوید. دسترسی به حالت **Debug Mode** آسان تر است و به صورت پیشفرض برنامه ها در این حالت اجرا می شوند. از این حالت برای رفع خطاها و اشکال زدایی برنامه ها استفاده می شود که در درسهای آینده توضیح خواهیم داد. شما همچنین می توانید از **break points** و قسمت **Help** برنامه در مواقعی که با خطا مواجه می شوید استفاده کنید. برای اجرای برنامه با حالت **Debug Mode** می توانید از منوی **Debug** گزینه **Start Debugging** را انتخاب کرده و یا دکمه **F5** را فشار دهید. همچنین می توانید بر روی شکل  در **toolbar** کلیلک کنید. اگر از حالت **Debug Mode** استفاده کنید برنامه نمایش داده شده و فورا ناپدید می شود. برای جلوگیری از این اتفاق شما می توانید از کلاس و متد **System.Console.ReadKey()** برای توقف برنامه و گرفتن ورودی از کاربر جهت خروج از برنامه استفاده کنید. (درباره متدها در درسهای آینده توضیح خواهیم داد.)

```
namespace MyFirstProgram
{
    class Program
    {
        static void Main()
        {
            System.Console.WriteLine("Welcome to Visual C# Tutorials!");
            System.Console.ReadKey();
        }
    }
}
```

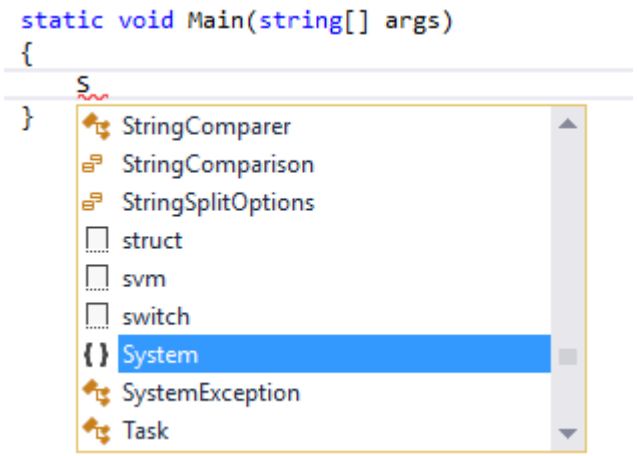
حال برنامه را در حالت **Debug Mode** اجرا می کنیم. مشاهده می کنید که برنامه متوقف شده و از شما در خواست ورودی می کند، به سادگی و با زدن دکمه **Enter** از برنامه خارج شوید. من از حالت **Non-Debug** به این علت استفاده کردم تا نیازی به نوشتن کد اضافی **Console.ReadKey()** نباشد. از این به بعد هر جا ذکر شد که برنامه را اجرا کنید برنامه را در حالت **Non-Debug** اجرا کنید. وقتی به مبحث استثناءها رسیدیم از حالت **Debug** استفاده می کنیم. حال با خصوصیات و ساختار اولیه سی شارپ آشنا شدید در درسهای آینده مطالب بیشتری از این زبان برنامه نویسی قدرتمند خواهید آموخت.

استفاده از IntelliSense

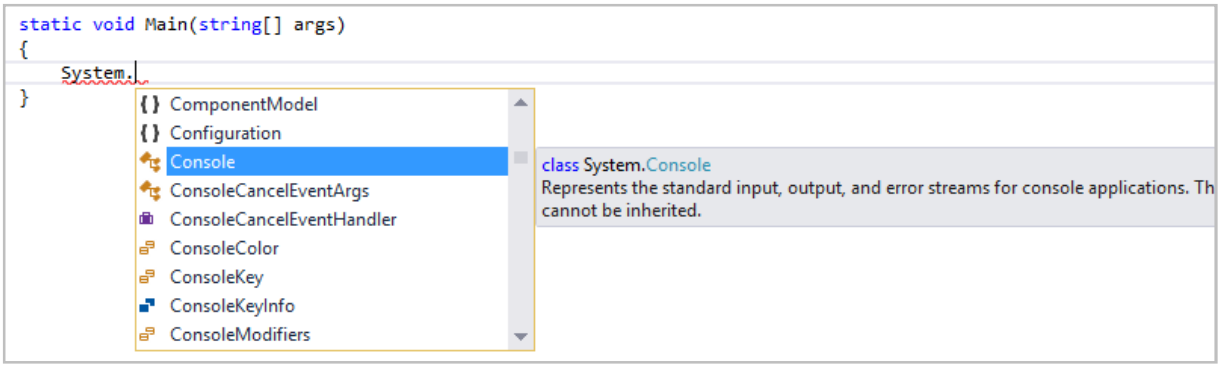
شاید یکی از ویژگیهای مهم **Visual Studio** ، اینتل لایسنس **(IntelliSense)** باشد. **IntelliSense** ما را قادر می سازد که به سرعت به کلاسها و متدها و....دسترسی پیدا کنیم. وقتی که شما در محیط کدنویسی حرفی را تایپ کنید، **IntelliSense**فورا فعال می شود. کد زیرا را در داخل متد **Main** بنویسید.

```
System.Console.WriteLine("Welcome to Visual C# Tutorials!");
```

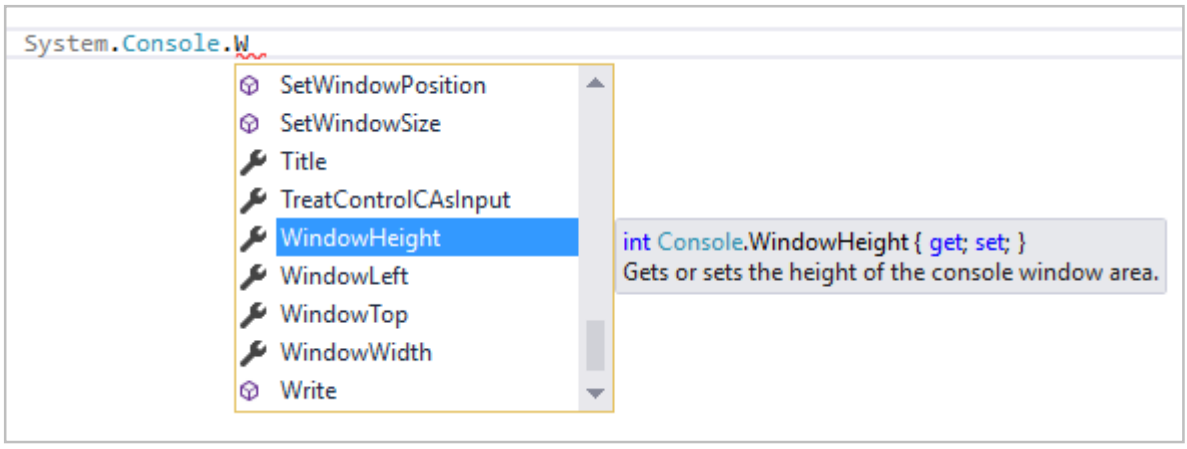
اولین حرف را تایپ کنید تا **IntelliSense**فعال شود.



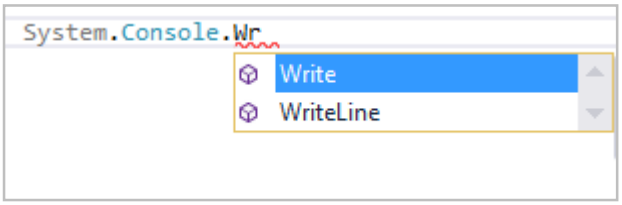
IntelliSense لیستی از کلمات به شما پیشنهاد می دهد که بیشترین تشابه را با نوشته شما دارند. شما می توانید با زدن دکمه **tab** گزینه مورد نظرتان را انتخاب کنید. با تایپ نقطه (.) شما با لیست پیشنهادی دیگری مواجه می شوید.



اگر بر روی گزینه ای که می خواهید انتخاب کنید لحظه ای مکث کنید توضیحی در رابطه با آن مشاهده خواهید کرد مانند شکل بالا. هر چه که به پایان کد نزدیک می شوید لیست پیشنهادی محدود تر می شود. برای مثال با تایپ حرف **W**، **IntelliSense** فقط کلماتی را که دارای حرف **W** هستند را نمایش می دهد.



با تایپ حرف های بیشتر لیست محدودتر شده و فقط دو کلمه را نشان می دهد.



اگر **IntelliSense** نتواند چیزی را که شما تایپ کرده اید پیدا کند هیچ چیزی را نمایش نمی دهد. برای ظاهر کردن **IntelliSense** کافیست دکمه ترکیبی **Ctrl+Space**را فشار دهید. برای انتخاب یکی از متدهایی که دارای چند حالت هستند، می توان با استفاده از دکمه های مکان نما (بالا و پایین) یکی از حالت ها را انتخاب کرد. مثلا متد **Writeline()** همانطور که در شکل زیر مشاهده می کنید دارای **19** حالت نمایش پیغام در صفحه است.

IntelliSense به طور هوشمند کدهایی را به شما پیشنهاد می دهد و در نتیجه زمان نوشتن کد را کاهش می دهد.

رفع خطاها

بیشتر اوقات هنگام برنامه نویسی با خطا مواجه می شویم. تقریبا همه برنامه هایی که امروزه می بینید، حداقل از داشتن یک خطا رنج می برند. خطا ها می توانند برنامه شما را با مشکل مواجه کنند . در سی شارپ سه نوع خطا وجود دارد:

خطای کامپایلری

این نوع خطا از اجرای برنامه شما جلوگیری می کند. این خطا ها شامل خطای دستور زبان می باشد. این بدین معنی است که شما قواعد کد نویسی را رعایت نکرده اید. یکی دیگر از موارد وقوع این خطا هنگامی است که شما از چیزی استفاده می کنید که نه وجود دارد و نه ساخته شده است. حذف فایلها یا اطلاعات ناقص در مورد پروژه ممکن است باعث به وجود آمدن خطای کامپایلری شود. استفاده از برنامه بوسیله برنامه دیگر نیز ممکن است باعث جلوگیری از اجرای برنامه و ایجاد خطای کامپایلری شود.

خطاهای منطقی

این نوع خطا در اثر تغییر در یک منطق موجود در برنامه به وجود می آید. رفع این نوع خطاها بسیار سخت است چون شما برای یافتن آنها باید کد را تست کنید. نمونه ای از یک خطای منطقی برنامه ای است که دو عدد را جمع می کند ولی حاصل تفریق دو عدد را نشان می دهد. در این حالت ممکن ات برنامه نویس علامت ریاضی را اشتباه تایپ کرده باشد.

استثناء

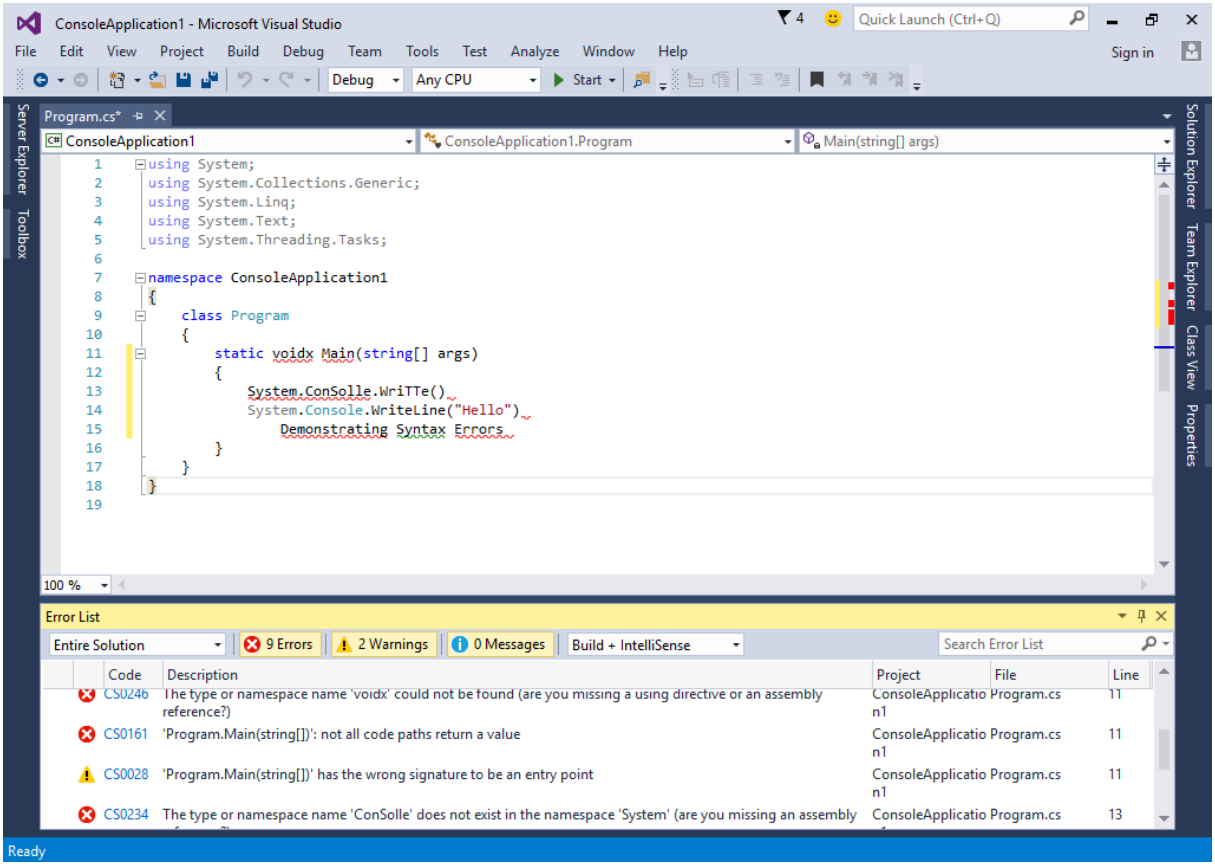
این نوع خطاها هنگامی رخ می دهند که برنامه در حال اجراست. این خطا هنگامی روی می دهد که کاربر یک ورودی نامعتبر به برنامه بدهد و برنامه نتواند آن را پردازش کند. ویژوال استودیو و ویژوال سی شارپ دارای ابزارهایی برای پیدا کردن و برطرف کردن خطاها هستند. وقتی در محیط کدنویسی در حال تایپ کد هستیم یکی از ویژگیهای ویژوال استودیو تشخیص خطاهای ممکن قبل از اجرای برنامه است. زیر کدهایی که دارای خطای کامپایلری هستند خط قرمز کشیده می شود.

```
namespace ConsoleApplication1
{
    class Program
    {
        static voidx Main(string[] args)
        {
            System.ConSolle.WriTTe()
            System.Console.WriteLine("Hello")
                Demonstrating Syntax Errors
        }
    }
}
```

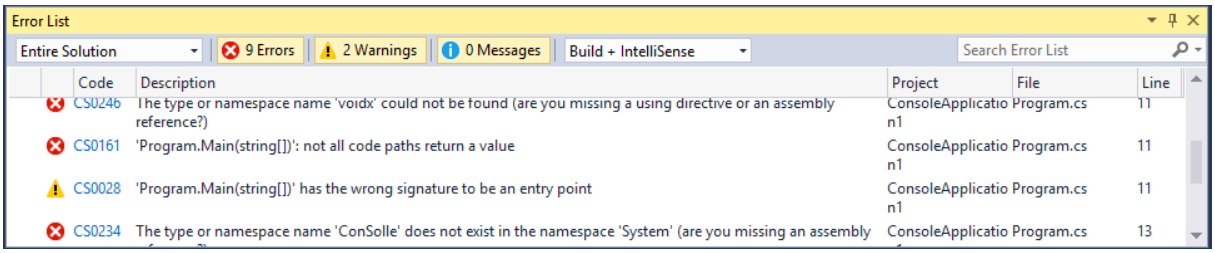
هنگامی که شما با ماوس روی این خطوط توقف کنید، توضیحات خطا را مشاهده می کنید. شما ممکن است با خط سبز هم مواجه شوید که نشان دهنده اخطار در کد است، ولی به شما اجازه اجرای برنامه را می دهند. به عنوان مثال ممکن است شما یک متغیر را تعریف کنید ولی در طول برنامه از آن استفاده نکنید. (در درس های آینده توضیح خواهیم داد).

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int number;
        }
    }
}
```

در باره رفع خطاها در آینده توضیح بیشتری می دهیم. ErrorList (لیست خطاها) که در شکل زیر با فلش قرمز نشان داده شده است به شما امکان مشاهده خطاها ، هشدارها و رفع آنها را می دهد. برای باز کردن Error List می توانید به مسیر View> Error List بروید.



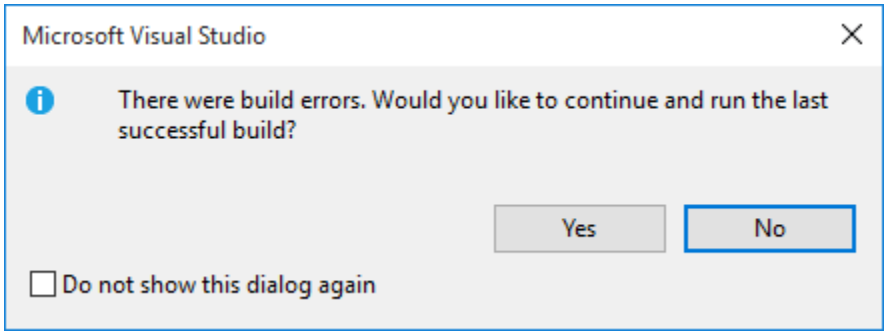
همانطور که در شکل زیر مشاهده می کنید هرگاه برنامه شما با خطا مواجه شود لیست خطاها در Error List نمایش داده می شود.



در شکل بالا تعدادی خطا همراه با راه حل رفع آنها در Error List نمایش داده شده است. Error List دارای چندین ستون است که به طور کامل جزییات خطاها را نمایش می دهند.

ستون	توضیحات
Description	توضیحی درباره خطا
File	فایلی که خطا در آن اتفاق افتاده است
Line	شماره خطی از فایل که دارای خطاست
Column	ستون یا موقعیت افقی خطا در داخل خط
Project	نام پروژه ای که دارای خطاست.

اگر برنامه شما دارای خطا باشد و آن را اجرا کنید با پنجره زیر روبرو می شوید:



مربع کوچک داخل پنجره بالا را تیک بزنید چون دفعات بعد که برنامه شما با خطا مواجه شود، دیگر این پنجره به عنوان هشدار نشان داده نخواهد شد. با کلیک بر رو ی دکمه Yes برنامه با وجود خطا نیز اجرا می شود. اما با کلیک بر روی دکمه NO اجرای برنامه متوقف می شود و شما باید خطاهای موجود در پنجره Error List را بر طرف نمایید. یکی دیگر از ویژگیهای مهم پنجره Error List نشان دادن قسمتی از برنامه است که دارای خطاست. با یک کلیک ساده بر روی هر کدام خطاهای موجود در پنجره Error List ، محل وقوع خطا نمایش داده می شود.

خطایابی و برطرف کردن آن

در جدول زیر لیست خطاهای معمول در پنجره Error List و نحوه برطرف کردن آنها آمده است : کلمه Sample ،جانشین نامهای وابسته به خطاهایی است که شما با آنها مواجه می شوید و در کل یک کلمه اختیاری است:

خطا	توضیح	راه حل
; expected	در پایان دستور علامت سیمیکان (;) قرار نداده اید	اضافه کردن یک سیمیکالن (;)

خطا	توضیح	راه حل
The name ‘sample’ does not exist in the current context.	کلمه sample در کد شما نه تعریف شده و نه وجود دارد	کلمه sample را حذف یا تعریف کنید.
Only assignment, call, increment, decrement, and new object expressions can be used as a statement.	کد جز دستورات سی شارپ نیست	دستور را حذف کنید
Use of unassigned local variable ‘sample’	متغیر sample مقدار دهی اولیه نشده	قبل از استفاده از متغیر آن را مقدار دهی اولیه کنید
The type or namespace name ‘sample’ could not be found (are you missing a using directive or an assembly reference?)	نوع یا فضای نام متغیر sample تعریف نشده است	باید یک کلاس یا فضای نام ،به نام sample ایجاد کنید
‘MyMethod()’: not all code paths return a value	بدین معنات که متد MyMethod() که به عنوان متدی با مقدار برگشتی در نظر گرفته شده در همه قسمتهای کد دارای مقدار برگشتی نیست.	مطمئن شوید که متد در همه قسمتهای کد دارای مقدار برگشتی است
Cannot implicitly convert type ‘type1′ to ‘type2′	متغیر type2 نمی تواند به متغیر type1 تبدیل شود.	با استفاده از متدهای تبدیل انواع به هم ،دو متغیر را یکسان کنید.

نگران یادگیری کلمات به کار رفته در جدول بالا نباشید چون توضیح آنها در درسهای آینده آمده است.

توضیحات

وقتی که کدی تایپ می کنید شاید بخواهید که متنی جهت یادآوری وظیفه آن کد به آن اضافه کنید. در سی شارپ (و بیشتر زبانهای برنامه نویسی) می توان این کار را با استفاده از توضیحات انجام داد. توضیحات متونی هستند که توسط کامپایلر نادیده گرفه می شوند و به عنوان بخشی از کد محسوب نمی شوند. هدف اصلی از ایجاد توضیحات خوانایی و تشخیص نقش کدهای نوشته شده توسط شما ، برای دیگران است. فرض کنید که می خواهید در مورد یک کد خاص، توضیح بدهید، می توانید توضیحات را در بالای کد یا کنار آن بنویسید.

از توضیحات برای مستند سازی برنامه هم استفاده می شود. در برنامه زیر نقش توضیحات نشان داده شده است :

```
1   namespace CommentsDemo
2   {
3       class Program
4       {
5           public static void Main(string[] args)
6           {
7               // This line will print the message hello world
8               System.Console.WriteLine("Hello World!");
9           }
10      }
11  }
```

در خط 7 یک توضیح ساده (تک خطی) نشان داده شده است. توضحات بر دو نوعند، توضیحات تک خطی و توضیحات چند خطی :

```
// single line comment

/* multi
   line
   comment */
```

توضیحات تک خطی همانگونه که از نامش پیداست برای توضیحاتی در حد یک خط به کار می روند. این توضیحات با علامت // شروع می شوند و هر نوشته ای که در سمت راست آن قرار بگیرد جز توضیحات به حساب می آید. این نوع توضیحات معمولا در بالا یا کنار کد قرار می گیرند. اگر توضیح در باره یک کد به بیش از یک خط نیاز باشد از توضیحات چند خطی استفاده می شود. توضیحات چند خطی با */ شروع و با /* پایان می یابند. هر نوشته ای که بین این دو علامت قرار بگیرد جز توضیحات محسوب می شود. نوع دیگری از توضیحات ، توضیحات XML نامیده می شوند. این نوع با سه اسلش (///) نشان داده می شوند. از این نوع برای مستند سازی برنامه استفاده می شود و در درس های آینده در مورد آنها توضیح خواهیم داد.

کاراکترهای کنترلی

کاراکترهای کنترلی، کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می شوند و به دنبال آنها یک حرف یا عدد می آید و یک رشته را با فرمت خاص نمایش می دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می توان از کاراکتر کنترلی \n استفاده کرد :

```
System.Console.WriteLine("Hello\nWorld!");
```

```
Hello
World!
```

مشاهده کردید که کامپایلر بعد از مواجهه با کاراکتر کنترلی \n نشانگر موس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می دهد. متد **WriteLine()** هم مانند کاراکتر کنترلی \n یک خط جدید ایجاد می کند ، البته بدین صورت که در انتهای رشته یک کاراکتر کنترلی \n اضافه می کند :

```
System.Console.WriteLine("Hello World!");
```

کد بالا و کد زیر هیچ فرقی با هم ندارند :

```
System.Console.Write("Hello World!\n");
```

متد **Write()** کارکردی شبیه به **WriteLine()** دارد با این تفاوت که نشان گر موس را در همان خط نگه می دارد و خط جدید ایجاد نمی کند. جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می دهد :

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	چاپ فضای خالی	\0
حرکت به صورت عمودی	\v	صدای بیپ	\a
چاپ کاراکتر یونیکد	\u	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی از بک اسلش(\) استفاده می کنیم. از آنجاییکه \ معنای خاصی به رشته ها می دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم :

```
System.Console.WriteLine("We can print a \\ by using the \\\\ escape sequence.");
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\ ،نشان دادن مسیر یک فایل در ویندوز است :

```
System.Console.WriteLine("C:\\Program Files\\Some Directory\\SomeFile.txt");
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (") برای نشان دادن رشته ها استفاده می کنیم برای چاپ آن از "\استفاده می کنیم :

```
System.Console.WriteLine("I said, \"Motivate yourself!\".");
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (') از \' استفاده می کنیم :

```
System.Console.WriteLine("The programmer\'s heaven.");
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می شود :

```
System.Console.WriteLine("Left\tRight");
Left    Right
```

هر تعداد کاراکتر که بعد از کاراکتر کنترلی \r بیایند به اول سطر منتقل و جایگزین کاراکترهای موجود می شوند :

```
System.Console.WriteLine("Mitten\rK");
Kitten
```

مثلا در مثال بالا کاراکتر K بعد از کاراکتر کنترلی \r آمده است. کاراکتر کنترلی حرف K را به ابتدای سطر برده و جایگزین حرف M می کند.

برای چاپ کاراکترهای یونیکد می توان از \u استفاده کرد. برای استفاده از \u ،مقدار در مبنای 16 کاراکتر را درست بعد از علامت \u قرار می دهیم. برای مثال اگر بخواهیم علامت کپی رایت (©) را چاپ کنیم باید بعد از علامت \u مقدار 00A9 را قرار دهیم مانند :

```
System.Console.WriteLine("\u00A9");
©
```

برای مشاهده لیست مقادیر مبنای 16 برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید :
http://www.ascii.cl/htmlcodes.htm

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می دهد. بیشترین خطا زمانی اتفاق می افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می کند.

علامت @

علامت @ به شما اجازه می دهد که کاراکترهای کنترلی را رد کرده و رشته ای خوانا تر و طبیعی تر ایجاد کنید. وقتی از کاراکترهای کنترلی در یک رشته استفاده می شود ممکن است برای تایپ مثلا یک بک اسلش (\ ) به جای استفاده از دو علامت \\ از یک \ استفاده کرده و دچار اشتباه شوید.

این کار باعث به وجود آمدن خطای کامپایلری شده و چون کامپایلر فکر می کند که شما می خواهید یک کاراکتر کنترلی را تایپ کنید، کاراکتر بعد از علامت \ را پردازش می کند و چون کاراکتر کنترلی وجود ندارد خطا به وجود می آید. به مثال زیر توجه کنید :

```
System.Console.WriteLine("I want to have a cat\dog as a birthday present."); //Error
```

با وجودیکه بهتر است در مثال بالا از اسلش (/) در cat/dog استفاده شود ولی عمدا از بک اسلش (\ ) برای اثبات گفته بالا استفاده کرده ایم. کامپایلر خطا ایجاد می کند و به شما می گوید که کاراکتر کنترلی \d قابل تشخیص نیست، چون چنین کاراکتر کنترلی وجود ندارد. زمانی وضعیت بدتر خواهد شد که کاراکتر بعد از بک اسلش کاراکتری باشد که هم جز یک کلمه باشد و هم جز کاراکترهای کنترلی. به مثال زیر توجه کنید :

```
System.Console.WriteLine("Answer with yes\no:");
Answer with yes
o
```

استفاده از علامت @ برای نادیده گرفتن کاراکترهای کنترلی

استفاده از علامت @ زمانی مناسب است که شما نمی خواهید از علامت بک اسلش برای نشان دادن یک کارکتر کنترلی استفاده کنید. استفاده از این علامت بسیار ساده است و کافی است که قبل از رشته مورد نظر آن را قرار دهید.

```
System.Console.WriteLine(@"I want to have a cat\dog as a birthday present.");
I want to have a cat\dog as a birthday present.
```

از علامت @ معمولا زمانی استفاده می شود که شما بخواهید مسیر یک دایرکتوری را به عنوان رشته داشته باشید. چون دایرکتوری ها دارای تعداد زیادی بک اسلش هستند و طبیعتا استفاده از علامت @ به جای دابل بک اسلش (\\) بهتر است.

```
System.Console.WriteLine(@"C:\Some Directory\SomeFile.txt");
C:\Some Directory\SomeFile.txt
```

اگر بخواهید یک دابل کوتیشن چاپ کنید به سادگی می توانید از دو دابل کوتیشن استفاده کنید.

```
System.Console.WriteLine(@"Printing ""double quotations""...");
Printing "double quotations"...
```

از به کار بردن علامت @ و کاراکترهای کنترلی به طور همزمان خودداری کنید چون باعث چاپ کاراکتر کنترلی در خروجی می شود.

استفاده از علامت @ برای نگهداری از قالب بندی رشته ها

یکی دیگر از موارد استفاده از علامت @ چاپ رشته های چند خطی بدون استفاده از کاراکتر کنترلی \n است. به عنوان مثال برای چاپ پیغام زیر :

```
C# is a great programming language and
it allows you to create different
kinds of applications.
```

یکی از راه های چاپ جمله بالا به صورت زیر است :

```
Console.WriteLine("C# is a great programming language and\n" +
"it allows you to create different\n" +
"kinds of applications.");
```

به نحوه استفاده از \n در آخر هر جمله توجه کنید. این کاراکتر همانطور که قبلا مشاهده کردید خط جدید ایجاد می کند و در مثال بالا باعث می شود که جمله به چند خط تقسیم شود. از علامت + هم برای ترکیب رشته ها استفاده می شود. راه دیگر برای نمایش مثال بالا در چندین خط استفاده از علامت @ است

```
Console.WriteLine(@"C# is a great programming language and
it allows you to create different
kinds of applications.");
```

در این حالت کافیست که در هر جا که می خواهید رشته در خط بعد نمایش داده شود دکمه Enter را فشار دهید

متغیرها

متغیر مکانی از حافظه است که شما می توانید مقادیری را در آن ذخیره کنید. می توان آن را به عنوان یک ظرف تصور کرد که داده های خود را در آن قرار داده اید. محتویات این ظرف می تواند پاک شود یا تغییر کند. هر متغیر دارای یک نام نیز هست. که از طریق آن می توان متغیر را از دیگر متغیر ها تشخیص داد و به مقدار آن دسترسی پیدا کرد. همچنین دارای یک مقدار می باشد که می تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد. مقدار متغیر می تواند تهی نیز باشد. متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده ای که در آن ذخیره می شود یکی است. متغیر دارای عمر نیز هست که از روی آن می توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد. و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است. ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می کنیم. هنگامی که یک برنامه ایجاد می کنیم احتیاج به یک مکان برای ذخیره داده ،مقادیر یا داده هایی که توسط کاربر وارد می شوند داریم. این مکان همان متغیر است. برای این از کلمه متغیر استفاده می شود چون ما می توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار